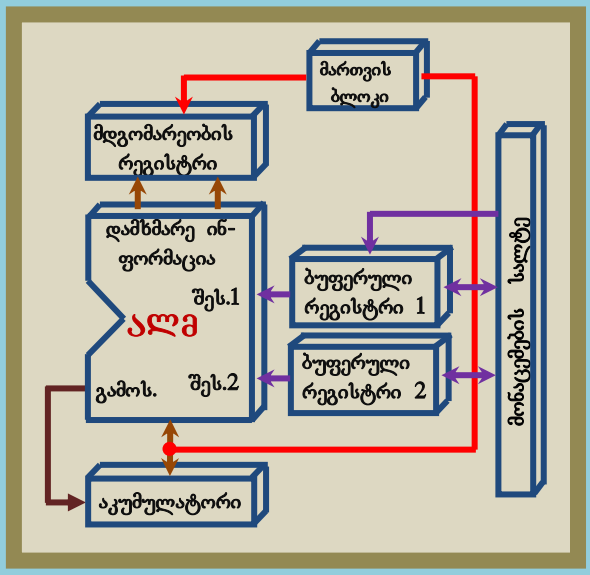
ალექსანდრე დუნდუა

ტრანსპორტზე მიკროპროცესორული ტექნიკის გამოყენების საფუძვლები



«ტექნიკური უნივერსიტეტი»

ადამიანის მიერ საუკუნეზე მეტი ხნის წინათ გამოგონებულ პირველ რელეურ მმართველ მოწყობილობებს წარმოადგენს სიგნალიზაციის, ცენტრალიზაციისა და ბლოკირების მოწყობილობები, რომლებსაც დღეს სარკინიგზო ავტომატიკისა და ტელემექანიკის მოწყობილობებსაც უწოდებენ.

მიკროპროცესორული საელემენტო ბაზის გამოყენება საშუალებას გვაძლევს დავამუშავოთ სარკინიგზო ავტომატიკისა და ტელემექანიკის მნიშვნელოვნად გაფართოებული ფუნქციური შესაძლებლობების მქონე თვისობრივად ახალი სისტემები.

უმაღლესი კლასის მეშვიდე სერიის ავტომობილ BMW–ში ფუნქციონირებს 60-ზე მეტი მიკროკონტროლერი.

სათანადო ჰიპერბოლის გამოყენების შემთხვევაში თანამედროვე ავიაგამანადგურებელი შეიძლება სპეციალიზებული რთული მიკროპროცესორული სისტემის გარე მოწყობილობად მივიჩნიოთ.

Alexander Dundua
Professor of the Georgian
Technical University

ალექსანდრე დუნდუა. ტრანსპორტზე მიკროპროცესორული ტექნიკის გამოყენების საფუძვლები. დამხმარე სახელმძღვანელო უმაღლესი სკოლის სტუდენტებისათვის. თბილისი: «ტექნიკური უნივერსიტეტი»; *2017* წელი. - *344* გვ.

განხილულია ტრანსპორტზე მიკროპროცესორული ტექნიკის გამოყენების საკითხები, მოცემულია: ძირითადი ცნობები მიკროპროცესორებისა და მიკროკომპიუტერების შესახებ, ელექტრონულ-გამოთვლითი ტექნიკის საინფორმაციო-ლოგიკური საფუძვლები, სხვადასხვა სირთულის მიკროპროცესორული სისტემების ორგანიზების პრინციპები, მათი ფუნქციონირების ალგორითმები და აგრეთვე ლოგიკური მოწყობილობების პროგრამული რეალიზების მეთოდები.

დანართებში მოცემულია მიკროპროცესორულ ტექნიკაში გამოყენებული ინგლისური ტერმინოლოგია და აბრევიატურები, აგრეთვე განხილულია ელექტრონიკის სხვადასხვა საკითხი.

განკუთვნილია «ტრანსპორტის», «საგზაო ინჟინერიისა» და «საზღვაო მეცნიერებების» საგანმანათლებლო პროგრამათა ბაკალავრებისათვის; შეიძლება გამოყენებული იქნეს აგრეთვე «მექანიკის ინჟინერიისა და ტექნოლოგიის» საგანმანათლებლო პროგრამის ბაკალავრების, ტექნიკური პროფილის სპეციალისტების, მაგისტრანტებისა და მიკროპროცესორული ტექნიკით დაინტერესებული ნებისმიერი პირის მიერ.

რეცენზენტები: პროფესორი *მერაბ გოცაძე*

შპს «თბილისის სატრანსპორტო კომპანიის» სიგნალიზაციისა და კავშირგაბმულობის სამსახურის სამგზავრო ავტომატიკის დისტანციის უფროსი *გიორგი რევაზიშვილი*

ტექსტის აწყობა, გრაფიკული გაფორმება და გარეკანი *ალექსანდრე დუნდუასი*

სიყვარულით ჩემს შვილებს - ვერიკოსა და ალექსანდრეს!

წინასიტყვაობის მაგივრ

ანუ მმართველი მოწყობილობების აგებისა და ტრანსპორტზე მათი გამოყენების საკითხისათვის

შესასვლელი (მმართველი) ზემოქმედებების შედეგად ნახტომისებურად გადანაცვლებადი ელემენტის შემცველი პირველი ავტომატურ მოწყობილობა, რომელიც ახდენდა გამოსასვლელთან მიერთებული ელექტრული მოწყობილობების კომუტირებას (ჩართვას, გამორთვას, გადართვას), ***1835*** წელს გამოიგონა ***ჯ. ჰენრიმ***. მოქმედების პრინციპის მიხედვით იგი წარმოადგენდა კონტაქტის ელექტრომაგნიტურ გადამრთველს და იმ ფუნქციის გამო, რისთვისაც იგი პირველად იქნა გამოყენებული, მიიღო სახელწოდება ***relay*** (***რელე***), რაც გულისხმობს დასვენებული ცხენით დაღლილი საფოსტო ცხენის შეცვლის ანალოგიით ძლიერი დენით შესუტებული დენის შეცვლას.

რელე უწყვეტი ან დისკრეტული შესასვლელი ზემოქმედების შედეგად ნახტომისებურად ცვლის საკუთარი კონტაქტური ან უკონტაქტო გამოსასვლელის ელექტროფიზიკურ მდგომარეობას. შესასვლელი ზემოქმედების გავლენით გამოსასვლელის ელექტროფიზიკური მდგომარეობის ნახტომისებურად ცვლილებას ***რელეური მახასიათებელი*** ეწოდება.

გამოსასვლელის ელექტროფიზიკური მდგომარეობის ნახტომისებურად ცვლა რელეს საშუალებას აძლევს მოახდინოს მის გამოსასვლელზე (გამოსასვლელებზე) მიერთებული ელექტრული წრედების კომუტირება (ჩართვა, გამორთვა, გადართვა). რელეს რეალურ კონსტრუქციებს აქვს მართვის ***კვაზინახტომისებური მახასიათებელი*** და ამიტომ იგი ამოქმედდება არა მყისიერად, არამედ მმართველი სიგნალის მიმართ გარკვეული დაყოვნებით. მინიმალური რაოდენობის კავშირის ხაზებით დაკავშირებული მინიმალური რაოდენობის დეტა-

ლებისაგან რეალიზებულ უმარტივეს რელეს ***რელეური ელემენტი*** ეწოდება. რელეური ელემენტებით შეიძლება ავაგოთ რთული რელეები და ***რელეური მოწყობილობები.***

ჰენრის მიერ გამოგონებული რელე წარმოადგენდა ელექტრომაგნიტურ მოწყობილობას. გამოგონებიდან ***40*** წლის განმავლობაში იგი მარტო ტელეგრაფიაში გამოიყენებოდა და მხოლოდ ***1878*** წლიდან დაიწყეს მისი გამოყენება იმ პერიოდისათვის ინოვაციურად მიჩნეულ ***სატელეფონო ტექნიკაში***. ამის შემდეგ მალე ელექტრომაგიტური რელეები გამოყენებული იქნა ელექტროტექნიკური, რადიოტექნიკური და ხელსაწყოთსამშენებლო მოწყობილობებშიც, რომლებშიც იგი ელექტრული წრედების საკომუტაციო ფუნქციებს ასრულებდა. ყველაზე მოგვიანებით რელეს გამოყენება დაიწყეს ***მმართველი მოწყობილობების*** ასაგებად.

მართვა წარმოადგენს წინასწარ განსაზღვრული მიზნის მიღწევისათვის გამიზნული მოქმედებების ერთობლობას. მათი შესრულება მოითხოვს აზროვნებითი და პრაქტიკული საქმიანობის ჰარმონიული ერთობლიობას და მეტად შრომატევადია. მისი შემსუბუქების მიზნით ადამიანს ბუნებრივად გაუჩნდა ***მმართველი მოწყობილობების*** აგების მიზანი. მართვის ტექნიკური მოწყობილობების აგების დროს წამოჭრილი პრობლემების გადასაწყვეტად ***1937*** წელს ***ნორბერტ ვინერმა*** შეკრიბა მეცნიერები, რომელთა ძალისხმევით მოგვიანებით წარმოშვა ახალი მეცნიერება – ***კიბერნეტიკა***. ფიზიოლოგები პირველად შეეჯახნენ ინჟინრული ტიპის იდეებს. მეცნიერების ერთმანეთისაგან მნიშვნელოვნად დაშორებულ სფეროებში მომუშავე სპეციალისტების ურთიერთობამ გამოიღო ნაყოფი და ***1943*** წელს ამერიკელმა ნეიროფიზიოლოგმა ***უ. მაკ-კალოხმა*** და მისმა მოწაფემ ***უ. პიტსმა*** [20]:

▲ დაამუშავეს ***ნეირონის,*** როგორც უმარტივესი პროცესორული ელემენტის მოდელი;

▲შემოგვთავაზეს ლოგიკური და არითმეტიკული ოპერაციების შესრულების უნარის მქონე ***ნეირონული ქსელის*** სტრუქტურა;

▲ გამოთქვეს დასაბუთებული ვარაუდი იმის შესახებ, რომ მათ მიერ შემოთავაზებულ ნეირონულ ქსელს აქვს სწავლის, სახეების ამოცნობისა და მიღებული ინფორმაციის განზოგადების უნარი.

მაკ-კალოხისა და ***ვალტერ პიტსის*** მიერ დაწყებული კვლევები ლოგიკურად დაასრულა ამერიკელმა მათემატიკოსმა ***სტივენ კოულ***

კლინმა, რომელმაც ნეირონული მოდელი აღწერა ***რეგულარულ სიმრავლედ*** წოდებული საკუთარი მათემატიკური სისტემის გამოყენებით. მის საპატივცემლოდ რეგულარულ სიმრავლეთა ალგებრას ***კლინის ალგებრა*** ეწოდა. აღნიშნული ალგებრის რეგულარული გამოსახულების საფუძველზე აკადემიკოსმა ***ვ. მ. გლუშკოვმა*** შემოგვთავაზა სასრული ავტომატების აბსტრაქტული სინთეზის ძირითადი ალგორითმი [**5**]. ამ ალგორითმის ფორმალიზებული ვარიანტი ჩვენ მიერ იქნა დამუშავებული, რომელიც ***2005 -2008*** წლებში მიმოსვლის გზათა ***სანკტ-პეტერბურგის*** სახელმწიფო უნივერსიტეტის სამეცნიერო შრომების კრებულებში გამოქვეყნდა [**6;7**]. მისი გადამუშავებული ვარიანტი ***ДАНАРТ 4***-შია მოყვანილი.

ჯონ ფონ ნეიმანმა მმართველი მოწყობილობების ასაგებად ნეირონული არასაიმედო ელემენტების გამოყენების შესაძლებლობების შესწავლისას დაასკვნა, რომ სტრუქტურაში საკმაო დიდი რაოდენობის ელემენტების შეტანით შეიძლება უზრუნველყოფილიყო აღნიშნული მოწყობილობის საიმედოდ ფუნქციონირება **[23].**

ნეიმანის მიერ დაწყებული კვლევა განაგრძო ***კლოდ შენონმა,*** რომელმაც ციფრული მოწყობილობის ასაგებად აბსტრაქტული ნეირონული ელემენტების ნაცვლად გამოიყენა რელეები.

რელეები ***ნეიმანს*** შემთხვევით არ აურჩევია. მან ჯერ კიდევ **1938** წელს გამოაქვეყნა რელეური და გადამრთველი სქემებისადმი მიძღვნილი ნაშრომი, რომელშიც დაასაბუთა ***XIX*** საუკუნის შუა პერიოდში ინგლისელი მათემატიკოსის ***ჯორჯ ბულის*** მიერ დამუშავებული ***ლოგიკის ალგებრის*** უდიდესი პრაქტიკული მნიშვნელობა [**22**].

აღნიშნული ალგებრა თითქმის მთელი საუკუნის განმავლობაში ითვლებოდა პრაქტიკული ღირებულებებისაგან დაცლილ წმინდა თეორიული მნიშვნელობის შრომად. ***შენონმა*** გვიჩვენა ასეთი შეხედულების მცდარობა. მან დაგვანახა ლოგიკურ ფუნქციებსა და რელეურ მოწყობილობებს შორის არსებული ურთიერთცალსახა შესაბამისობა: ნებისმიერ ლოგიკურ ფუნქციას შეეთანადება გარკვეული რელეური მოწყობილობა და პირიქით, ნებისმიერი რელეური მოწყობილობა ახდენს გარკვეული ლოგიკური ფუნქციის რეალიზებას. ***ვლადიმერ მაიაკოვსკის*** ერთ-ერთ ცნობილი ლექსის სტრიქონების პერიფრაზირებას თუ მოვახდენთ, შეგვიძლია ზემოთ ფორმულირებული პოსტულატი ასე ჩამოვაყალიბოთ: ვამბობთ ლოგიკურ ფუნქციას – რელეური

მოწყობილობა გვაქვს წარმოდგენილი, ვამბობთ რელეურ მოწყობილობას და ვგულისხმობთ ამ მოწყობილობით რეალიზებულ ლოგიკურ ფუნქციას! სხვა სიტყვებით რომ ვთქვათ, ***რელეური მოწყობილობის მათემატიკურ მოდელს წარმოადგენს ლოგიკური ფუნქცია.***

სწორედ რელეური მოწყობილობების ზემოთ აღნიშნული გამოკვლევამ ითამაშა გადამწყვეტი როლი იმაში, რომ ***კლოდ შენონმა 1956*** წელს ***ედუარდ ფორესტ მურთან*** ერთად მმართველი მოწყობილობების ასაგებად ***ნეიმანის*** მიერ გამოყენებული აბსტრაქტული ნეირონული ელემენტების რეალურ ეკვივალენტებად სწორედ რელეები შეირჩია. მიღებულმა შედეგება გვიჩვენა მკვლევრების მიერ მიღებული გადაწყვეტილების სისწორე. აღმოჩნდა, რომ რელეების გამოყენებისას მნიშვნელოვნად მცირდება მოწყობილობაში შესატანი დამატებითი ელემენტების რაოდენობა. არასაიმედო ნეირონების გამოყენებისას მოწყობილობის საიმედო ფუნქციონირებისათვის თუ მასში არსებული ელემენტების რაოდენობა დაახლოებით ***60000***-ჯერ უნდა გაზრდილიყო, არასაიმედო რელეების გამოყენებისას აღნიშნული რაოდენობის მხოლოდ ***67***-ჯერ გაზრდაც საკმარისი იყო [21].

აღნიშნული გამოკვლევის საფუძველზე დადგინდა, რომ საიმედო მმრთველი მოწყობილობების ასაგებად გამოსაყენებელ ყველაზე საუკეთესო საელემენტო ბაზას იმ პერიოდისათვის ***ელექტრომაგნიტური რელეები*** წარმოადგენდა.

ინტუიციური მიხვედრილობის წყალობით რელეების გამოყენებით იქნა რეალიზებული ***პირველი მმართველი მოწყობილობები.*** მასაჩუსეტის უნივერსიტეტის პროფესორის ***სემუელ კოლდუელის*** (***Samuel H. Calduell***) აზრით ასეთ მოწყობილობებს წარმოადგენდა სიგნალიზეციის, ცენტრალიზაციისა და ბლოკირების (***სცბ***-ს) სისტემები [10], რომლებსაც დღეს ***სარკინიგზო ავტომატიკისა და ტელემექანიკის სისტემებსაც*** უწოდებენ.

რკინიგზის ტრანსპორტს მმართველი მოწყობილობების გამოყენების პირველობა ერგო არა მისი შემქმნელი პირების ახირებულობის, არამედ მათ მიერ ობიექტური აუცილებლობის გაცნობიერების გამო. ჯერ კიდევ ***1825*** წელს გახსნილ ***სტოქტონ-დარლინგტონის*** პირველ რკინიგზაზე სულ რაღაც ***24*** კმ/სთ სიჩქარით მოძრავ პირველ შემადგენლობას წინ მიუძღოდა მხედარი, რომელსაც ხელში ეჭირა ალამი წარწერით ***«The private danger is the public good»*** («***კერძო ხიფათი,***

საზოგადოებრივი კეთილდღეობა»). ლოზუნგის ჭეშმარიტების პრაქტიკულ დადასტურებას დიდი ხანი არ დასჭირვებია. ***1830*** წელს ***ლივერპულსა და მანჩესტერის*** დამაკავშირებელ რიგით მეორე რკინიგზის გახსნისას სარკინიგზო მშენებლობის თაგამოდებულმა მომხრემ, ინგლისის პარლამენტის წევრმა ***ჰუკინსონმა*** ვერ შენიშნა მატარებლის გაგზავნის სიგნალი (ამ პერიოდში ორთქლმავალს არ გააჩნდა საყვირი) და ორთქმავლის ბორბლებქვეშ მოჰყვა. ამ რკინიგზის ხაზის გახსნიდან რამდენიმე დღის შემდეგ მოხდა მეორე უბედური შემთხვევა: ქალაქებს ***ლისტერსა*** და ***სვენინგტონს*** შორის მოძრავი მატარებელი დაეჯახა ლისტერის ბაზარში მიმავალ ზეთითა და კვერცხებით დატვირთულ საზიდარს, რომელიც გადადიოდა სარკინიგზო გადასასვლელზე. აღნიშნულმა შემთხვევამ დიდი მღელვარება გამოიწვია და რკინიგზის დირექციამ იმ დღესვე მოიწვია თათბირი, რომელზედაც ***ჯორჯ სტეფენსონის*** წინადადებით გადაწყდა ***ლივერპულ-მანჩესტერის*** მთელ უბანზე გარკვეული მანძილის დაშორებით განელაგებინათ მესიგნალეები. ისინი დღისით ალმებით, ხოლო ღამით ფარნებით გადასცემდნენ მატარებელთა მოძრაობის ნებადამრთავ და ამკრძალავ სიგნალებს.

1841 წელს ბრიტანეთში გამოჩნდა პირველი უძრავი სიგნალები – ***სემაფორები*** (ბერძ, ***Sema*** -ნიშანი, - ***psoros***-მზიდი), რომლებიც დღევანდელი შუქნიშნის მექანიკური წინაპარია. ისინი მოძრავი ფრთებით უჩვენებდა მემანქანეს თავისუფალია თუ არა გზა, ხოლო ამ ფრთებს სპეციალური ბაგირის მეშვეობით მართავდნენ მესიგნალეები. მათი ავტომატურად მართვის იდეა ***XIX*** საუკუნის შუა პერიოდში დაიბადა და მისი პირველი რეალიზაცია ***1859*** წელს საფრანგეთის ქალაქებს ***პარიზსა*** და ***სან-ჟერმენის*** სადგურებს შორის არსებულ უბანზე არსებული სემაფორებისათვის განახორციელა გამომგონებელმა ***ჟან ბარანოვსკიმ***. ამის შემდეგ დაიწყო მატარებელთა უსაფრთხოდ მოძრაობისათვის სიგნალიზაციის, ცენტრალიზაციისა და ბლოკირების მოწყობილობების დამუშავებისა და გამოყენების ეპოქა.

მატარებელთა უსაფრთხოდ მოძრაობის ორგანიზებისათვის დაახლოებით ***1860*** წელს დიდი ბრიტანეთის რკინიგზებზე გამოჩნდა პირველი ***მექანიკური ცენტრალიზაციები. XIX*** საუკუნეში ბრიტანეთში ფუნქციონირებდა მექანიკური ცენტრალიზაციების დამამზადებელი რამდენიმე კომპანია, რომელთაგანაც გამოირჩეოდა ***Saxby & Farmer***

კომპანია. გერმანიაში ***XX*** საუკუნის დასაწყისში ფართოდ იყო გავრცელებული ***Einheit*** მოდელის მექანიკური ცენტრალიზაცია. მექანიკური ცენტრალიზაციებიდან რელეურ ცენტრალიზაციებზე გადასვლის პროცესი ძალიან ნელა მიმდინარეობდა. ამ მიმართულებით გადადგმულ პირველ ნაბიჯებად შეიძლება ჩაითვალოს დაახლოებით ***1870*** წელს გამოგონებული ***Blockfeld*** სახელწოდების ბლოკ-აპარატი და სარელსო წრედები. ორივე ამ მოწყობილობამ ისეთი ფუნქციების რეალიზება გახადა შესაძლებელი, რომელთა შესრულება მექანიკური საშუალებებით შეუძლებელი იყო. ამან მნიშვნელოვნად გააფართოვა მექანიკური ცენტრალიზაციათა ტექნიკური შესაძლებლობები.

1900 წლიდან დაწყებულ შემდგომ ეტაპისათვის დამახასიათებელია ნაწილობრივ ელექტრული და ნაწილობრივ მექანიკური ფუნქციების მქონე (ე. წ. ***ელექტრომექანიკური***) ცენტრალიზაციების დამუშავება.

რელეური ტექნოლოგიის გამოყენებით პირველი მთლიანად ელექტრული ცენტრალიზაციები დამუშავდა და დაინერგა ორ მსოფლიო ომებს შორის არსებულ პერიოდში. მეორე მსოფლიო ომის შემდგომ პირველ ორ ათწლეულში მოხდა მათი სრულყოფა და ისინი გადაიქცა მთელ მსოფლიოში გავრცელებულ ტექნოლოგიად. ***დღეისათვის ექსპლუატირებადი ცენტრალიზაციების უმრავლესობა რელეურ ცენტრალიზაციას წარმოადგენს.***

სარკინიგზო მმართველი მოწყობილობების მომდევნო ეტაპს საფუძველი ჩაეყარა ***1978*** წელს, როდესაც ***გეტებორგის*** (***შვეცია***) სარკინიგზო სადგურზე დაინერგა შვედური ფირმა ***Ericson***-ის მიერ დამუშავებული ***JZN-850*** სახელწოდების პირველი ***მიკროპროცესორული ცენტრალიზაცია.*** აქედან დაიწყო სარკინიგზო ავტომატიკისა და ტელემექანიკის ***მიკროპროცესორული სისტემებით რელეური სისტემების შეცვლის პროცესი.***

ბოლო **30** წლის განმავლობაში მიკროპროცესორული და გამოთვლითი ტექნიკის ბაზაზე აგებული სარკინიგზო ავტომატიკისა და ტელემექანიკის სისტემები თანდათან მკვიდრდება სარკინიგზო პრაქტიკაში. მათი წარმოებით დაკავებულია არაერთი ცნობილი ფირმა, რომლებიც ინერგება არა მარტო განვითარებულ, არამედ მრავალ განვითარებად ქვეყანაშიც.

სპეციალისტები მიიჩნევენ, რომ ***სცბ***-ს ტრადიციულ სისტემებს აქვს კარგი (**80** წლამდე) ხანგამძლეობა [**16**]. ამიტომ რკინიგზაზე ახალ სისტემათა დანერგვის ტემპები ჩვეულებრივ მაღალი არ არის. ისედაც დაბალ ტემპებს კიდევ უფრო ანელებს მიკროელექტრონულ სისტემათა უსაფრთხოების პრობლემების გადაწყვეტის პროცესში წამოჭრილი სიძნელეები. ამიტომ მსოფლიოს რკინიგზებზე დღესაც ერთდროულად ფუნქციონირებს ***სცბ***-ს სხვადასხვა თაობისა და მოდიფიკაციის მრავალი მოწყობილობა. მაგალითად, შვეიცვარიაში გასული საუკუნის ბოლოს მუშაობდა **163** მექანიკური, **260** ელექტრომექანიკური, **413** რელეური და მხოლოდ ერთი მიკროპროცესორული ცენტრალიზაცია [**16**]. ანალოგური სიტუაციაა მრავალ სხვა ქვეყანაშიც.

საქართველოს რკინიგზის სადგურები და გადასარბენები, ბათუმის სადგურისა და მისთან მიმდებარე გადასარბენების გამოკლებით, მთლიანად ავტომატიკის რელეური სისტემებითაა აღჭურვილი. ისინი თანდათანობით უნდა ჩაანაცვლოს მიკროპროცესორულმა სისტემებმა. მათთვის დამახასიათებელი ლოგიკური ამოცანების გადაწყვეტისათვის მისადაგებული ***ბინარული პროგრამის ბლოკ-სქემის*** აგების ჩვენ მიერ დამუშავებული ფორმალური მეთოდი ***2005*** წელს იქნა გამოქვეყნებული ***სანკტ-პეტერბურგის*** ზემოთ აღნიშნული უნივერსტეტის სამეცნიერო შრომების კრებულში [**6**]. მისი რადენადმე სახეშეცვლილი ვარიანტი ***12.4*** პარაგრაფში გვაქვს მოყვანილი.

მიკროპროცესორული ტექნიკის გამოყენების გარეშე არ დარჩენილა ***საავტომობილო წარმოებაც.*** თანამედროვე ავტომობილში მიკროპროცესორული სისტემები მართავს ძრავას, მუხრუჭებს, მოსახვევებში ავტომობილის მოძრაობას; ავარიის დროს მათ მოჰყავს სამუშაო მდგომარეობაში უსაფრთხოების ბალიშები; ისინი ადევნებს თვალყურს მანქანაში არსებული დასაჯდომებისა და უკანა ხედვის სარკეების მდგომარეობას, იცავს მანქანას გატაცებისა და მოსრიალებისაგან, უზრუნველყოფს ავტომობილის გამონაბოლქვ აირებში მავნე აირების იმ დონემდე შემცირებას, რომელსაც მოითხოვს მრავალი ქვეყნის კანონმდებლობა, შესაძლებელს ხდის საწვავის ეკონომიურად ხარჯვას და ა.შ.

ვიმედოვნებთ, რომ მოცემული ნაშრომი ქართველ სტუდენტებს თანამედროვე სატრანსპორტო მიკროპროცესორული ტექნოლოგიების

ათვისებაში ქმედით დახმარებას გაუწევს. სიამოვნებით მივიღებთ მისი შემდგომი სრულყოფის მიზნით გამოთქმულ საქმიან მოსაზრებებს. ვეცდებით მასში არსებული შესაძლო ხარვეზი მომავალში აღმოვფხვრათ. ნაშრომში ყურადღება ძირითადად მიკროპროცესორულ ტექნიკაზე გვაქვს გამახვილებული; მიკროპროცესორული ტექნიკის გამოყენებით სარკინიგზო სისტემების აგების სპეციფიკური საკითხები საკმაოდ ვრცლად [3, გვ. 285-343] გვაქვს გადმოცემული.

დასასრულს სასიამოვნო მოვალეობად ვთვლი მადლობა გადავუხადო ***რომანოზ (რომა) ხომასურიძეს*** სხვადასხვა სახის დახმარებისათვის, რომელიც ძალიან ხშირად მჭირდებოდა მოცემულ წიგნზე მუშაობის პერიოდში.

I დანართში მოყვანილია მიკროპროცესორულ ლიტერატურაში გამოყენებული ინგლისური ტერმინები და აბრევიატურები ***II დანართში*** პოპულარული ენითაა გადმოცემული დიოდებისა და ტრანზისტორების აგებულებისა და ფუნქციონირების პრინციპები. ***III დანართში*** დახასიათებულია ლოგიკური ელემენტების ძირითად ოჯახები. ***IV დანართში*** მოყვანილია ციფრული მოწყობილობების აბსტრაქტული სინთეზის ორიგინალური ალგორითმი, რომელიც სახელმძღვანელოს ავტორის მიერ ***2005-2008*** წლებში იქნა დამუშავებული მმართველი ლოგიკური მოწყობილობების პროგრამული და აპარატურული რეალიზაციისათვის.

I თავი
საწყისი ცნობები

1.1 პროცესორიდან – მიკროპროცესორამდე

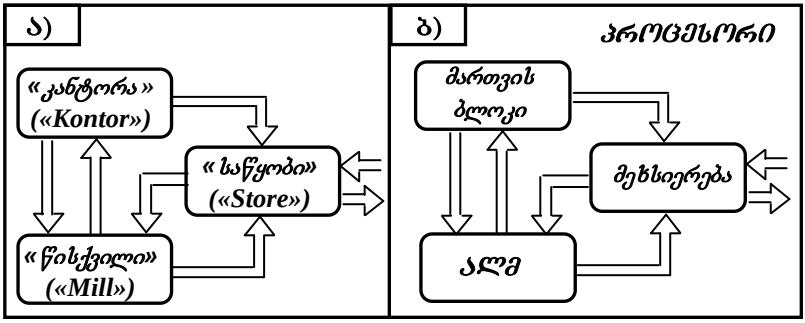
სტანდარტიზაციის საერთაშორისო ორგანიზაციის (***ISO***-ს) მიერ ფორმულირებული განსაზღვრების მიხედვით ***პროცესი*** ეწოდება შესასვლელი მონაცემების გამოსასვლელ მონაცემებად გარდამქმნელ ურთიერთდამოკიდებულ მოქმედებათა ერთობლიობას. ვინაიდან ანალოგურ ფუნქციას ასრულებს ნებისმიერი პროგრამაც, ამიტომ ***პროცესი ფაქტობრივად რეალიზების პროცესში მყოფ პროგრამას წარმოადგენს.***

ერთმანეთისაგან შეიძლება განვასხვაოთ ***ბუნებრივი და ხელოვნური პროცესები.*** პირველი მათგანის ინიცირებას იწვევს ადამიანისაგან დამოუკიდებელი ბუნებრივი მოვლენები, ხოლო მეორე მათგანს – ადამიანი ან მის მიერ კონსტრუირებული ხელოვნური მოწყობილობები. ხელოვნური პროცესის მაინიცირებელ მოწყობილობას, ბუნებრივია, ***პროცესორი*** შეიძლება ვუწოდოთ.

პროცესორის რეალიზების იდეა ეკუთვნის ინგლისელ მეცნიერსა და კონსტრუქტორს ***ჩარლზ ბებიჯს. 1822*** წელს მან შექმნა საანგარიშო მანქანა, რომელიც ფიქსირებული მოქმედებების შემსრულებელი სპეციალიზებული გამომთვლელი მანქანის წინაპრად შეიძლება მივიჩნიოთ. რეგისტრებში გარკვეული საწყისი მონაცემების ჩატვირთვის შემდეგ იგი ახდენდა მოცემული სახის მრავალწევრის ტაბულირებას. ამ მანქანას ბებიჯმა ***სხვაობითი მანქანა*** უწოდა, რადგან ფუნქციათა ტაბულირებისათვის იგი იყენებდა სასრული სხვაობების მეთოდს. მას ჰქონდა სამი ნაკლი: ***1)*** კონკრეტული ფუნქციის ტაბულირებისას მიღებულ ბოლო სხვაობაზე დამოკიდებულებით ადამიანს თვითონ უნდა მიეთითებინა მანქანისათვის ფუნქციონირების მომდევნო გზა; ***2)*** ფუნქციის ტაბულირებისათვის საჭირო მონაცემები მანქანაში ადამიანს თავად უნდა შეეტანა; ***3)*** მანქანა სასრული სხვაობის მეთოდზე დაფუძნებულ მხოლოდ ერთადერთ პროგრამას ასრულებდა.

ბებიჯს დაებადა ზემოთ ჩამოთვლილი ნაკლისაგან თავისუფალი მანქანის აგების იდეა. ასეთ უნივერსალურ მანქანას ადამიანის ჩაურე-

ვლად უნდა შეძლებოდა: ***1)*** კონკრეტული პროგრამის შესრულებისას მიღებულ საშუალედო შედეგზე დამოკიდებულებით მუშაობის მომდევნო გზის დამოუკიდებლად არჩევა; ***2)*** ნებისმიერი პროგრამის შესასრულებლად საჭირო მონაცემების დამოუკიდებლად მოძიება; ***2)*** რამდენიმე პროგრამის რეალიზება;



ნახ. 1.1. ბებიჯის ანალიზური მანქანისა (***ა***) თანამედროვე კომპიუტერის პროცესორის (***ბ***) სქემები

ზემოთ აღნიშნული იდეის განხორციელებისათვის ბებიჯმა გადაწყვიტა დაემუშავებინა გამოთვლების პროცესის მმართველი მოწყობილობა. მას უნდა ჰქონოდა ბრძანებებისა და მონაცემების შესანახი სპეციალური ***საწყობი***, ანუ ***store*** (ნახ.***1.1,ა***) და პროცესების (ბრძანებების) უშუალოდ შემსრულებელ მოწყობილობა, რომელსაც ბებიჯმა ***mill***, ანუ ***წისქვილი*** უწოდა. საწყობში ბრძანებები და მონაცემები ავტომატურად ***შეტანის სპეციალურ მოწყობილობას*** უნდა შეეტანა, ხოლო წისქვილის მიერ ფორმირებული «მზა პროდუქტი» საწყობში უნდა გადატვირთულიყო. მომხმრებელს ამ პროდუქტის მიღება სპეციალური ***გამოტანის მოწყობილობით*** უნდა შეძლებოდა. საწყობიდან წისქვილში ბრძანებებისა და მონაცემების გადატანისა და წისქვილის მუშაობის პროცესებისათვის უნდა ეხელმძღვანელა სპეციალური მოწყობილობას, რომლისთვისაც ბებიჯს სახელი არ დაურქმევია, მაგრამ, ლოგიკურად თუ ვიმსჯელებთ, მას სახელად ***კანტორა*** შეეფერება.

საწყობს თუ რეგისტრებისაგან შემდგარ ***ოპერატიულ მეხსიერებას***, წისქვილს – ***არითმეტიკულ-ლოგიკურ მოწყობილობასა (ალგ)*** და კანტორას - ***მართვის მოწყობილობას*** ვუწოდებთ, მაშინ ბებიჯის შემოთავაზებული მოწყობილობა ***1.1,ბ*** ნახაზზე ნაჩვენებ კვანძის სახეს

მიიღებს, რომელსაც თანამედროვე კომპიუტერული სისტემის პროცესორის ანალოგიური სტრუქტურა აქვს.

ადრეულ კომპიუტერში პროცესორი ***ხისტად ჩაშენებულ*** საკმაოდ რთულ მოწყობილობას წარმოადგენდა. მაგალითად, გერმანელი ინჟინრის ***კონრად ცუზეს*** (***1910-1995***) მიერ ***1941*** წლის დეკემბერში კონსტრუირებულ ***Z3*** კომპიუტერში პროცესორი აგებული იყო ***2400*** რელეს გამოყენებით, რომელთაგანაც არითმეტიკული მოწყობილობაზე მოდიოდა ***600***, მეხსიერებაზე - ***1400*** და მართვის ბლოკზე – ***600*** რელე.

ინტეგრალური სქემის გამოყენებით აგებულმა და სპეციალურ კორპუსში მოთავსებულმა პროცესორმა კომპიუტერისაგან განცალკევებულად არსებობის უნარი შეიძინა. მიკროზომების გამო მას ***მიკროპროცესორის*** ეწოდა. პროცესორი და მიკროპროცესორი ფაქტობრივად ანალოგურ ფუნქციებს ასრულებს, ამიტომ ტერმინებს ***«პროცესორსა»*** და ***«მიკროპროცესორს»*** ხშირად სინონიმებადაც მიიჩნევენ.

მსოფლიოში პირველი მიკროპროცესორის შექმნის ისტორია საკმაოდ ყურადსაღებია. ***1969*** წლის ზაფხულში კალკულატორების ახალი ოჯახის დამუშავებაზე მომუშავე იაპონურ კომპანია «***Busicom***»-ს დასჭირდა რამდენიმე ათასი ტრანზისტორის შემცველი ინტეგრალური სქემა და მის დასამზადებლად მიმართა მეხსიერებისთვის განკუთვნილი მიკროსქემების დამზადების სფეროში სახელმოხვეჭილ ფირმა "***Intel***"-ს. ერთობლივი პროექტის დამუშავებაში ამ უკანასკნელმა ჩართო საკუთარი ინჟინერი ***მ. ჰოფი.*** იგი გაეცნო ***Busicom***-ის საქმიანობასა და მას შესთავაზა ალტერნატიული იდეა: ***12*** რთული სპეციალიზებული ინტეგრალური მიკროსქემების ნაცვლად შეექმნათ ერთი დაპროგრამებადი უნივერსალური მიკროსქემა, რომელსაც შემდეგ მიკროპროცესორი ეწოდა. ***ჰოფის*** იდეამ გაიმარჯვა და ფირმა «***Intel***»-მა მიიღო შეკვეთა მსოფლიოში პირველი ***მიკროპროცესორი*** დაემზადებინა. იდეის რეალიზაცია ადვილი არ აღმოჩნდა და დამხმარედ ***1970*** წლის ბოლოს საქმეში ჩართეს იტალიელი ფიზიკოსი და ელექტრონიკოსი ***ფედერიკო ფადჟინი*** (***Federico Faggin, 1942*** წ.). აღსანიშნავია, რომ მოგვიანებით ***ფადჟინმა*** დაარსა ფირმა ***«Zilog»*** და შექმნა შემდგომში მრავალ კომპიუტერში მომუშავე შესანიშნავი ***8***-თანრიგიანი პროცესორი ***Z80***. ***ფ. ფადჟინმა 9*** თვის განმავლობაში პროცესორი აღწერის ეტაპიდან კრისტალში რეალიზებულ

დონემდე მიიყვანა. ***1971*** წლის ***15*** ნოემბერს გამოსული ***"Intel 4004"*** სახელწოდების მსოფლიოში პირველი მიკროპროცესორი შეიცავდა ***2300*** ტრანზისტორს და მისი მუშაობის სიხშირე ***108*** კილოჰერცის ტოლი იყო. კორპორაცია ***Intel***-ის დამაარსებელისა და დირექტორთა საბჭოს საპატიო დირექტორის ***გორდონ ერლ მურის*** (***Gordon Earle Moore***) მიერ ემპირული დაკვირვების საფუძველზე აღმოჩენილი კანონის (ე. წ. ***მურის კანონის***) თანამედროვე ფორმულირების თანახმად ***ინტეგრალური სქემის კრისტალზე განთავსებული ტრანზისტორების რაოდენობა ყოველ 24 თვეში ორმაგდება***. ხშირად ციტირებადი ***18*** თვიანი ინტერვალი დაკავშირებულია ***Intel***-ის თანამშრომლის ***დ. ჰაუსის პროგნოზთან***, რომლის თანახმადაც ტრანზისტორების რაოდენობისა და თითოეული ამ ტრანზისტორის სისწრაფის ზრდის გამო მიკროპროცესორის მწარმოებლურობა ყოველ ***18*** თვეში ორმაგდება.

გორდონ მურის ზემოთმოყვანილი კანონის მოქმედების შესაბამისად ***2011*** წელს დამუშავებული ***Core I7***-ს სახელწოდების პროცესორში არსებული ტრანზისტორების რაოდენობამ მილიარდ ***160*** მილიონს მიაღწია. დღეისათვის არსებული მონაცემების თანახმად აღნიშნული რაოდენობა სამი მილიარდის ფარგლებშია, ოღონდ ამ რაოდენობის ზუსტი განსაზღვრა უკვე თვით მიკროპროცესორების მწარმოებელ ფირმებსაც უჭირს.

მოგვიანებით ***გორდონ მურმა*** იწინასწარმეტყველა, რომ მისი კანონი ***2015*** წლისთვის დაკარგავდა ძალას, ოღონდ მისი ეს წინასწარმეტყველება არ გამართლდა. ***გორდონ მურის*** კანონი თუ ***2025*** წლამდე დარჩება სამართლიანი, მაშინ მომავლის კომპიუტერს წამში უფრო მეტი გამოთვლების ჩატარება შეეძლება, ვიდრე ადამიანის ტვინს, რომელიც ***86*** მილიარდი ნეირონისაგან შედგება.

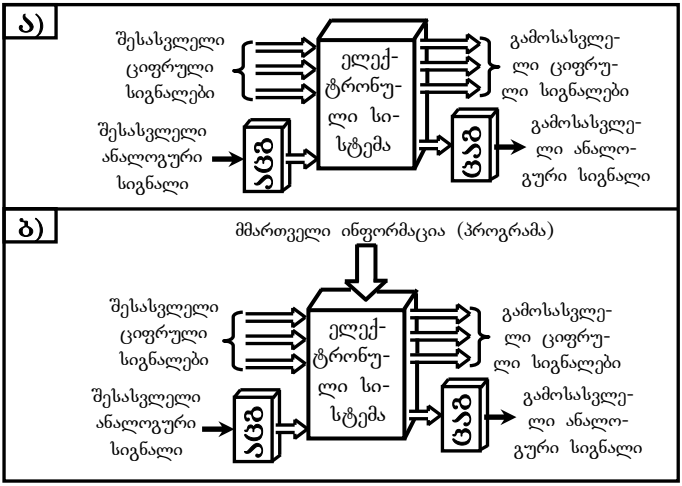
1.2. სისტი და მოქნილი ელექტრონული სისტემები

ელექტრონული სისტემა შეიძლება ვუწოდოთ ინფორმაციის დამამუშავებელ ნებისმიერ ელექტრონული კვანძს, ბლოკს, ხელსაწყოს ან კომპლექსს. მისი კერძო შემთხვევას წარმოადგენს მიკროპროცესორული სისტემა, რომელიც შესასვლელი სიგნალების გადამუშავების საფუძველზე გამოიმუშავებს და გასცემს გამოსასვლელ სიგნალებს

(ნახ. ***1.2,ა***). შესასვლელ და გამოსასვლელ სიგნალებად შეიძლება გამოყენებული იქნეს როგორც ანალოგური, ისე ციფრული სიგნალები. სისტემის შიგნით შეიძლება ყოველგვარი ცვლილების გარეშე მოხდეს სიგნალების (ინფორმაციის) შენახვა და დაგროვება.

მიკროპროცესორულ სისტემას მხოლოდ ციფრების სახით წარმოდგენილი სიგნალების დამუშავება შეუძლია, ე.ი. იგი ***ციფრული სისტემების*** კლასს მიეკუთვნება. აღნიშნულის გამო ანალოგურ შესასვლელ სიგნალებს იგი ანალოგურ-ციფრული ***აცგ*** გარდამქმნელის (იხ. ***ნახ. 1.2,ა***) მეშვეობით ციფრულ სიგნალებად (კოდებად) გარდაქმნის.

შესასვლელი სიგნალების დამუშავების შედეგად ფორმირებული ციფრული გამოსასვლელი სიგნალები ასეთივე სახით გაიცემა სისტემიდან, მაგრამ თუ სისტემას ანალოგური გამოსასვლელი სიგნალების გაცემა მოეთხოვება, მაშინ ფორმირებულ ციფრულ სიგნალებს ანალოგურ სიგნალებად გარაქმნის ციფრულ ანალოგური ***ცაც*** გარდამქმნელი (იხ. ***ნახ.1.2,ა***).



ნახ.1.2. ა) *ელექტრონული სისტემა;* ***ბ)*** *დაპროგრამებადი (უნივერსალური) ელექტრონული სისტემა.*

ტრადიციული ციფრული სისტემის დამახასიათებელი თავისებურებაა ის, რომ მასში ინფორმაციის დამუშავებისა და შენახვის ალგორითმი ***სისტადაა დაკავშირებული*** სისტემის სქემოტექნიკასთან. ეს

ნიშნავს, რომ ამ ალგორითმების შეცვლა მხოლოდ სისტემის სტრუქტურის, სისტემაში შემავალი ელექტრონული კვანძებისა და/ან ამ კვანძებს შორის არსებული კავშირების შეცვლითაა შესაძლებელი. მაგალითად, შეკრების დამატებითი ოპერაციის შესასრულებლად სისტემას კიდევ ერთი სუმატორი უნდა დავუმატოთ. ცხადია, რომ სისტემის ექსპლუატაციის პროცესში ამის გაკეთება პრაქტიკულად შეუძლებელია; ამისათვის საჭიროა იქნეს ორგანიზებული სისტემის ხელახლა დაპროექტების, დამზადებისა და გამართვის ახალი საწარმოო პროცესი. ამიტომ ტრადიციულ ციფრულ სისტემას ხშირად ***«ხისტი ლოგიკის» მქონე სისტემას*** უწოდებენ.

«ხისტი ლოგიკიანი» ნებისმიერი სისტემა მხოლოდ და მხოლოდ ერთი ან (ძალიან იშვიათად) ძალიან მსგავსი რამდენიმე ამოცანის შესასრულებლად აიგება. მოცემულ შემთხვევევაში ***ამოცანა*** ეწოდება ელექტრონული სქემების მიერ შესასრულებელ ფუნქციებს. ასეთ სისტემას აქვს როგორც ღირსებები, ისე ნაკლოვანებები.

"ხისტი ლოგიკიანი" სისტემათა ღირსებებია: ***1) აპარატურული სიჭარბის არარსებობა***, ე.ი. სისტემის თითოეული ელემენტი აუცილებლად სრული დატვირთვით მუშაობს (ამ სისტემის გონივრულად დაპროექტების შემთხვევაში) ***2) მაქსიმალური სწრაფმოქმედება. სწრაფმოქმედება*** წარმოადგენს სისტემის მიერ საკუთარი ფუნქციების შესრულების სიჩქარის მაჩვენებელს. ამას განაპირობებს ის გარემოება, რომ ინფორმაციის დამუშავების ალგორითმების შესრულების სიჩქარე განისაზღვრება მხოლოდ ცალკეული ლოგიკური ელემენტების სისწრაფითა და ინფორმაციის გატარების გზათა შერჩეული სქემით, ხოლო ლოგიკურ ელემენტებს დროის მოცემულ მომენტში ყოველთვის მაქსიმალური სწრაფმოქმედება აქვს.

«ხისტი ლოგიკიანი» სისტემათა ნაკლს წარმოადგენს ის, რომ ყოველი ახალი ამოცანისათვის საჭიროა მისი ხელახლა დაპროექტება და დამზადება. ესაა ხანგრძლივი და ძვირი პროცესი, რომლის შესასრულებლად მაღალკვალიფიციური სპეციალისტებია საჭირო.

აღნიშნული ნაკლის დაძლევა ისეთი სისტემის აგებითაა შესაძლებელი, რომელსაც ერთი ალგორითმიდან მეორე ალგორითმზე გადაწყობა აპარატურის შეუცვლელად შეუძლია. ამ შემთხვევაში ჩვენ შეგვეძლება ამ სისტემას ესა თუ ის ალგორითმი პროგრამად წოდებული მმართველი ინფორმაციის სახით მივაწოდოთ (ნახ.***1.2,ბ***).

ასეთ შემთხვევაში ტადიციული სისტემა გარდაიქმნება ***უნივესალურ, მოქნილ სიტემად.*** მიკროპროცესორული სისტემა სწორედ ასეთი სისტემების კლასს მიეკუთვნება.

ნებისმიერი უნივერსალობა აუცილებლად ზრდის სიჭარბეს. მართლაც, მაქსიმალურად რთული ამოცანის გადაწყვეტა გაცილებით უფრო მეტ საშუალებებს მოითხოვს, ვიდრე უმარტივესი ამოცანის გადაწყვეტა. უნივერსალური სისტემის სიჭარბე ისეთი უნდა იყოს, რომ მას შეეძლოს ყველაზე რთული ამოცანის გადაწყვეტა. ასეთი სისტემა მარტივი ამოცანის გადაწყვეტისას მთელი ძალით არ იმუშავებს – მისი მთელი რესურსის გამოყენება საჭირო არ იქნება. ამასთანავე რაც უფრო მარტივი იქნება გადასაწყვეტი ამოცანა, მით უფრო მეტი იქნება სიჭარბე და მით უფრო ნაკლებად იქნება გამართლებული უნივესალობა. სიჭარბის გაზრდა ზრდის სისტემის ღირებულებას, მოხმარებულ სიმძლავრეს, ამცირებს საიმედოობას და ა.შ.

უნივერსალობა მნიშვნელოვნად ამცირებს სისტემის სწრაფმოქმედებასა. შეუძლებელია უნივერსალური სისტემის იმგვარად ოპტიმიზება, რომ იგი მაქსიმალურად სწრაფად ხსნიდეს ყველა ამოცანას. ზოგადი წესის თანახმად ***უნივერსალობისა და მოქნილობის გაზრდისას მცირდება სწრაფმოქმედება.*** უფრო მეტიც, ბუნებაში არ არსებობს ისეთი (თუნდაც უმარტივესი) ამოცანა, რომელსაც მაქსიმალურად სწრაფად გადაწყვეტს უნივერსალური სისტემა.

საბოლოოდ შეგვიძლია დავასკვნათ, რომ «ხისტი ლოგიკიანი» სისტემის გამოყენება მაშინ არის გამართლებული, როდესაც ხანგრძლივად უცვლელია გადასაწყვეტი ამოცანა, მაქსიმალურად მარტივია ინფორმაციის დამუშავების ალგორითმები და მოითხოვება მაღალი სწრაფმოქმედება. უნივერსალური (დაპროგრამებადი) სისტემების გამოყენება კი პირიქით, მაშინაა კარგი, როდესაც გადასაწყვეტი ამოცანები ხშირად იცვლება, რთულია ინფორმაციის დამუშავების ალგორითმები და მაღალი სწრაფმოქმედება ძალიან მნიშვნელოვანი არ არის. მაშასადამე, ნებისმიერი სისტემა თავის ადგილზეა კარგი.

ცხოვრებას განუწყვეტლივ შეაქვს კორექტივები არსებულ რეალობაში და ასეთ მდგომარეობაა სისტემების შერჩევაშიც. ბოლო ათი წლის განმავლობაში ძალიან გაიზადა უნივერსალური (მიკროპროცესორული) სისტემების სწრაფმოქმედება და მკვეთრად შემცირდა მათ ასაგებად გამოყენებული მიკროსქემების ღირებულება. ამან წარმოშვა

«ხისტი ლოგიკიან» სისტემათა გამოყენების არეალის შემცირების ტენდენცია, რომელიც მომავალში კიდევ უფრო გაღრმავდება.

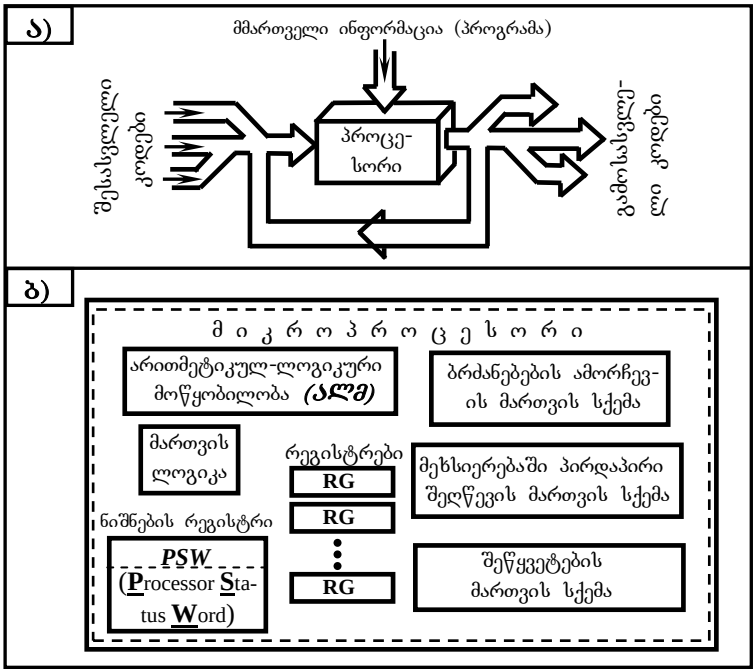
1.3. უნივერსალური მიკროპროცესორული სისტემების ფუნქციონირების თავისებურებები

უნივერსალურ მიკროპროცესორულ სისტემებში ინფორმაციის დამუშავების პროცესს მთლიანად მიკროპროცესორი ასრულებს. სისტემის დანარჩენი კვანძები მხოლოდ ისეთი დამხმარე პროცესების შესრულებითაა დაკავებული, როგორიცაა ინფორმაციის (მათ შორის მმართველი ინფორმაციის ანუ პროგრამის) შენახვა, გარე მოწყობილობებთან და მომხმარებლებთან კავშირის დამყარება და ა.შ. ასე რომ, პროცესორი პრაქტიკულად მთლიანად ცვლის ტრადიციული სისტემებისათვის აუცილებელ «ხისტ ლოგიკას». იგი ასრულებს არითმეტიკულ და ლოგიკურ ოპერაციებს, შინაგან რეგისტრებში დროებით შეინახავს კოდებს, მიკროპროცესორული სისტემის კვანძებს შორის გადააგზავნის მათ და ა.შ. მიკროპროცესორის მიერ შესასრულებელი ასეთი ელემენტარული ოპერაციების რაოდენობა რამდენიმე ასეულს აღწევს. შეიძლება შეგვექმნას ისეთი შთაბეჭდილება, რომ იგი სისტემის ტვინს წარმოადგენს. ქვემოთ დავინახავთ, რომ ეს წარმოდგენა რამდენადმე გაზვიადებულია.

მიკროპროცესორის თავისებურებას წარმოადგენს ის, რომ იგი ყველა ოპერაციას ასრულებს მიმდევრობით, ანუ რიგითობის დაცვით. მიუხედავად იმისა, რომ ბოლო წლებში გამოჩნდა ზოგიერთი ოპერაციების პარალელურად შემსრულებელი მიკროპროცესორები, აგრეთვე მრავალპროცესორული სისტემები, რომლებშიც რამდენიმე პროცესორი ერთ ამოცანას პარალელურად ასრულებს, მაგრამ, როგორც ამბობენ, ნებისმიერი გამონაკლისი მაინც ზოგადი კანონზომიერების დამადასტურებელი არგუმენტია.

ოპერაციების მიმდევრობით შესრულება, ***ერთი მხრივ***, უდავოდ დადებით მოვლენად უნდა ჩაითვალოს, რადგან სწორედ მისი საშუალებითაა შესაძლებელი მხოლოდ ერთმა პროცესორმა შეასრულოს ნებისმიერი, მათ შორის ძალიან რთული, ალგორითმი. ***მეორე მხრივ***, ოპერაციების მიმდევრობითი შესრულებაა იმის მიზეზი, რომ ალგორითმის შესრულების ხანგრძლივობა ალგორითმის სირთულეზეა დამო-

კიდებული: მარტივი ალგორითმი რთულ ალგორითმზე უფრო სწრაფად სრულდება. ე.ი. მიკროპროცესორულ სისტემას შეუძლია გააკეთოს ყველაფერი, ოღონდ იგი ძალიან სწრაფად ვერ მუშაობს, რადგან ყველა საინფორმაციო ნაკადს ერთადერთ კვანძში – მიკროპროცესორში უხდება გავლა (ნახ. ***1.3,ა***). ტრადიციულ ციფრულ სისტემას კი ადვილად შეუძლია ყველა საინფორმაციო ნაკადი პარალელურად შეასრულოს: ამას იგი აღწევს საკუთარი სქემის გართულების წყალობით.



ნახ.1.3. *ა) საინფორმაციო ნაკადები მიკროპროცესორულ სისტემაში;*
ბ) უმარტივესი მიკროპროცესორის სტრუქტურა.

მიკროპროცესორს, რომელსაც მრავალი ოპერაციის შესრულება ხელეწიფება, მოცემულ მომენტში შესასრულებელ ოპერაციას იგებს მმართველი ინფორმაციის ანუ პროგრამის მეშვეობით. ***პროგრამა*** წარმოადგენს კოდების ერთობლიობას, რომლებითაც დაშიფრულია ბრძანებები (ინსტრუქციები). ამ კოდების გაშიფვრით იგებს პროცესორი, თუ რა უნდა შეასრულოს მან. პროგრამას ადგენს ადამიანი,

ხოლო მიკროპროცესორი ამ პროგრამის უსიტყვო შემსრულებელია: იგი თავის თავს არ აძლევს არავითარი ინიციატივის გამოჩენის უფლებას (თუ, რა თქმა უნდა, წესივრულ მდგომარეობაშია). სწორედ ამის გამო აღვნიშნავდით ზემოთ, რომ გაზვიადებულია მიკროპროცესორის ტვინთან შედარება: იგი მხოლოდ ადამიანის მიერ შედგენილი ალგორითმის უტყვი შემსრულებელია. ამ ალგორითმს იგი მხოლოდ მაშინ დაარღვევს, როდესაც სისტემაში წარმოიშობა რაიმე მტყუნება. ალგორითმიდან ნებისმიერი გადახრის მიზეზი მიკროპროცესორში ან მიკროპროცესორულ სისტემაში მომხდარი მტყუნებაა.

პროცესორის მიერ შესასრულებელ ბრძანებათა ერთობლიობა წარმოქმნის ***ბრძანებათა სისტემას.*** პროცესორის ბრძანებათა სისტემის სტრუქტურა და მოცულობა განსაზღვრავს მის სწრაფმოქმედებას, მონილობასა და მოხერხებულ გამოყენებას. პროცესორის ბრძანებათა რაოდენობა შეიძლება რამდენიმე ათეულიდან რამდენიმე ასეულამდე იცვლებოდეს. ბრძანებათა სისტემა შეიძლება გამიზნული იყოს შეზღუდული წრის ან მაქსიმალურად ფართო წრის ამოცანების გადასაწყვეტად. პირველ შემთხვევაში საქმე გვაქვს ***სპეციალიზებულ,*** ხოლო მეორე შემთხვევაში – ***უნივერსალურ პროცესორებთან.*** ბრძანებათა კოდებს შეიძლება განსხვავებული რაოდენობის თანრიგები ჰქონდეს (ისინი შეიძლება შეიცავდეს ერთ ან რამდენიმე ბაიტს). თითოეული ბრძანების შესრულებისათვის გარკვეული დროა საჭირო, ამიტომ პროგრამის მთლიანად შესასრულებლად საჭირო დრო არა მარტო პროგრამაში არსებული ბრძანებების რაოდენობაზე, არამედ მასში გამოყენებულ ბრძანებათა სახეებზეა დამოკიდებული.

ბრძანებების შესასრულებლად პროცესორის სტრუქტურაში არსებობს შინაგანი რეგისტრები, არითმეტიკულ-ლოგიკური მოწყობილობა (***ალმ, ALU*** – ***Arithmetic Logic Unit***), მულტიპლექსორები, ბუფერები და სხვა კვანძები. ყველა კვანძის მუშაობა სინქრონიზდება პროცესორის გარეგანი საერთო ტაქტური სიგნალით. ე. ი. პროცესორი საკმაოდ რთული ციფრული მოწყობილობაა (ნახ. ***1.3,ბ***).

საინფორმაციო მიკროპროცესორული სისტემების შემქმნელებისათვის პროცესორის შინაგანი სტრუქტურის წვრილმანების შესახებ ინფორმაცია მაინცდამაინც მნიშვნელოვანი არ არის. მან პროცესორი უნდა განიხილოს როგორც ***«შავი ყუთი»*** რომელიც შესასვლელი და მმართველობითი კოდების საპასუხოდ ასრულებს გარკვეულ ოპერა-

ციებს და გასცემს საპასუხო გამოსასვლელ სიგნალებს. მომხმარებლმა აუცილებლად უნდა იცოდეს ბრძანებათა სისტემა, პროცესორის მუშაობის რეჟიმი და აგრეთვე გარესამყაროსთან პროცესორის ურთიერთმოქმედების წესები, რასაც ***ინფორმაციის გაცვლის პროტოკოლი ეწოდება.*** პროცესორის შინაგანი სტრუქტურის შესახებ საჭიროა ვიცოდეთ მხოლოდ ის, რაც აუცილებელია ამა თუ იმ ბრძანებისა და რეჟიმის ამოსარჩევად.

1.4. სქემოტექნიკის ძირითადი საკითხები

მიკროპროცესორული სისტემის დაპროექტებისათვი აუცილებელია სქემოტექნიკის ძირითადი საკითხების ცოდნა, ამიტომ მოცემულ პარაგრაფში მოკლედ განვიხილავთ ამ საკითხებს.

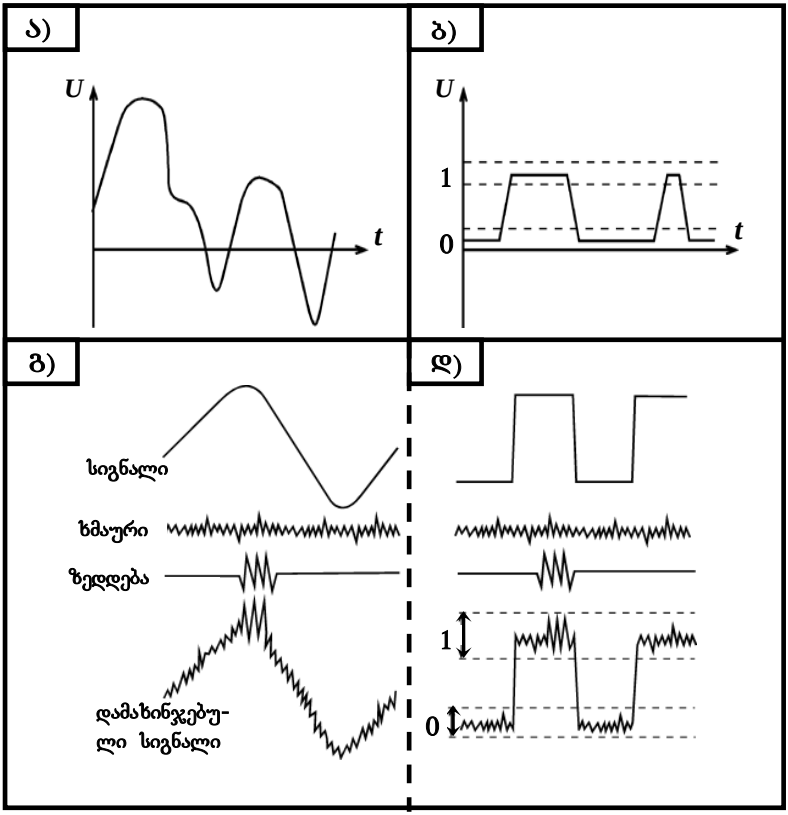
1.4.1. ძირითადი ცნობები ელექტრული სიგნალების შესახებ

ელექტრული სიგნალი ეწოდება დროში ცვლად ელექტრულ სიდიდეს (მაგალ. ძაბვას, დენს, სიმძლავრეს). მთელი ელექტრონიკა ძირითადად ელექტრულ სიდიდეებს იყენებს, თუმცა ბოლო პერიოდში სულ უფრო და უფრო ხშირად დაიწყო შუქოვანი სიგნალების გამოყენება, რომელიც სინათლის დროში ცვლად ინტენსივობას წარმოადგენს.

განასხვავებენ ანალოგურ და ციფრულ სიგნალებს. ***ანალოგური სიგნალი*** ეწოდება სიგნალს, რომელსაც შეუძლია განსაზღვრულ ფარგლებში ნებისმიერი მნიშვნელობის მიღება (ნახ. ***1.4,ა***). სახელწოდება «ანალოგური» გულისხმობს, რომ სიგნალი ფიზიკური სიდიდის ანალოგურად, ე.ი. უწყვეტად, იცვლება. ანალოგურ სიგნალების გამომყენებელ მოწყობილობებს ***ანალოგური მოწყობილობები*** ეწოდება. ***ციფრული სიგნალი*** ეწოდება სიგნალს, რომელსაც მხოლოდ ორი (ზოგჯერ – სამი) მნიშვნელობის მიღება შეუძლია, ამასთანავე, ნებადართულია ამ მნიშვნელობებიდან გარკვეული გადახრა (ნახ. ***1.4,ბ***). ციფრული სიგნალების გამომყენებელ მოწყობილობებს ციფრული (დისკრეტული) მოწყობილობები ეწდება.

ანალოგურ სიგნალებსა და მათ გამომყენებელ მოწყობილობებს აქვს მნიშვნელოვანი ნაკლი. კერძოდ, ისინი განიცდის მრავალნაირი პარაზიტული სიგნალების - ხმაურის, დაბრკოლებების, ზედდებების - გავლენას. ***ხმაური*** წარმოადგენს ნებისმიერი ელექტრონული მოწყობილობის შიგა ქაოტურ სუსტ სიგნალს. ***დაბრკოლებები*** და ***ზედდებები*** – გარედან ელექტრონულ

სისტემაში შემოსული და სასარგებლო სიგნალის დამამახინჯებელი სიგნალებია.



ნახ.1.4. *ანალოგური (**ა**) და ციფრული სიგნალები; გარე ზემოქმედებები ანალოგურ (**გ**) და ციფრულ (**დ**) სიგნალებზე*

ანალოგურმა სიგნალმა შეიძლება მიიღოს ნებისმიერი მნიშვნელობა (ნებადართულია ნებისმიერი მნიშვნელობა, ამიტომ მისი მნიშვნელობა უწყვეტად იცვლება (ნახ.***1.4,გ***), რაც მეტად სახიფათოა. ანალოგური სიგნალის ყოველი გარდაქმნა, ყოველი საშუალედო შენახვა, მანძილზე ყოველი გადაცემა პერმანენტულად აუარესებს ანალოგურ სიგნალს (ცვლის მის ჭეშმარიტ მნიშვნელობას) და საბოლოოდ შეიძლება მისი სრული განადგურებაც გამოიწვიოს.

ანალოგური სიგნალებისაგან განსხვავებით ***ციფრულ სიგნალებს*** მხოლოდ ორი მნიშვნელობის მიღება შეუძლია (მისთვის მხოლოდ ორი მნიშვნელობაა

ნებადართული). დასაშვებ ფარგლებში შესაძლო მცირეოდენი გადახრები ვერ ამახინჯებს ციფრულ სიგნალებს (ნახ.*1.3,დ*). სწორედ ამიტომაა შესაძლებელი: *1)* მოვახდინოთ ციფრული სიგნალების გაცილებით რთული და მრავალსაფეხურიანი გარდაქმნები; *2)* გაცილებით უფრო დიდხანს შევინახოთ უდანაკარგოდ; *3)* ანალოგურ სიგნალებზე გაცილებით მაღალი ხარისხით გადავცეთ შორ მანძილზე.

გარდა ზემოთ აღნიშნულისა, შესაძლებელია ციფრული მოწყობილობების აბსოლუტურად ზუსტად გაანგარიშება და სხვადასხვა სიტუაციაში მათი შესაძლო ქცევების წინასწარმეტყველება. ციფრული მოწყობილობები გაცილებით ნელა ბერდება, რადგან მათი პარამეტრების მცირეოდენი ცვლილებები მათ ფუნქციონირებაზე არ აისახება. ადვილია ციფრული მოწყობილობების დაპროექტება და გამართვა. სწორედ ამ ღირსებებმა უზრუნველყეს ციფრული ელექტრონიკის სწრაფი განვითარება.

მოკლედ განვიხილოთ ანალოგური და ციფრული მოწყობიბლობების დადებითი და უარყოფითი მხარეები. ციფრული მოწყობილობის მნიშვნელოვანი ნაკლია ის, რომ ციფრული სიგნალის აღსაქმელად საჭიროა მან გარკვეული მინიმალური დროის განმავლობაში მაინც შეინარჩუნოს ნებადართული დონე. ანალოგურ სიგნალს კი შეუძლია ნებისმიერი მნიშვნელობა მიიღოს უსასრულოდ მცირე დროის განმავლობაში. სხვა სიტყვებით რომ ვთქვათ ანალოგური სიგნალი განსაზღვრულია უწყვეტ დროში (ე. ი. დროის ნებისმიერ მომენტში), ხოლო ციფრული სიგნალი – დისკრეტულ დროში (ე.ი. დროის მხოლოდ ფიქსირებულ მომენტებში). ამიტომ ანალოგური მოწყობილობების მაქსიმალური სიჩქარე პრინციპულად ყოველთვის აღემატება ციფრული მოწყობილობების იმავე პარამეტრს. ციფრული მოწყობილობებისაგან განსხვავებით, ანალოგურ მოწყობილობებს შეუძლიათ უფრო სწრაფად ცვლად სიგნალებთან მუშაობა. ანალოგურ მოწყობილობებს ციფრულ მოწყობილობებზე უფრო სწრაფად შეუძლია დაამუშაოს და/ან გადასცეს ინფორმაცია. გარდა ამისა, ანალოგური სიგნალი, ინფორმაციის გადაცემის თვალსაზრისით, ციფრულ სიგნალზე უფრო ტევადია. ამიტომ იმ მოცულობის ინფორმაციის გადასაცემად, რომელსაც შეიცავს ერთი ანალოგური სიგნალი, ხშირად საჭიროა რამდენიმე (*4*-დან *16*-მდე) ციფრული სიგნალი იქნეს გამოყენებული.

1.4.2. ციფრული მიკროსქემების შესასვლელები და გამოსასვლელები

ციფრული მიკროსქემების დამზადების ტექნოლოგია და სქემოტექნიკა განსაზღვრავს მათი შესასვლელებისა და გამოსასვლელების მახასიათებ-

ლებსა და პარამეტრებს. ციფრული მოწყობილობათა შემქმნელებისათვის კი ნებისმიერი მიკროსქემა წარმოადგენს მხოლოდ «შავ ყუთს», რომლის შიგა ნაწილის ცოდნა არაა აუცილებელი. მნიშვნელოვანია, მას მხოლოდ ზუსტად ჰქონდეს წარმოდგენილი, თუ როგორ მოიქცევა მოცემულ კონკრეტულ მოწყობილობაში ჩართული მიკროსქემა, სწორად შეასრულებს თუ არა იგი მასზე დაკისრებულ ფუნქციას. ყველაზე მეტად გავრცელდა ციფრული მიკროსქემების აგების შემდეგი ორი ტექნოლოგია: ***ტტლ***-ტექნოლოგია (ტრანზისტორულ-ტრანზისტორული ლოგიკის გამომყენებელი ტექნოლოგია) ან ***ტტლშ***-ტექნოლოგია (შოტკის დიოდებიანი ***ტტლ***-ტექნოლოგია); ***2.*** «ლითონ-ჟანგეულ-ნახევარგამტარული» სტრუქტურის კომპლემენტარული ტრანზისტორებიანი ტექნოლოგია (ე. წ. ***კლშნ***-ტექნოლოგია).

მიკროსქემების შესასვლელი და გამოსასვლელი კასკადების ორგანიზება დამოკიდებულია აღნიშნული მიკროსქემების წარმოდგენის დონეებზე. განასხვავებენ მიკროსქემების წარმოდგენის შემდეგი დონეებს: ***1. პირველი დონე,*** რომელსაც ***მიკროსქემის ლოგიკური მოდელის აგების დონე*** ეწოდება; ***2. მეორე დონე,*** რომელსაც მიკროსქემისათვის დამახასიათებელი დროითი შეყოვნების განსაზღვრის დონე ეწოდება; ***3. მესამე დონე, რომელსაც მიკროსქემის ელექტრონული მოდელის აგების დონე ეწოდება.*** ამ დონეზე მიკროსქემის ასაგებად გამოიყენება ისეთი ელექტრონული კომპონენმტები, როგორიცაა ტევადობები, ინდუქციურობები, რეზისტორები და ა.შ.

დასაწყისში განვიხილოთ ***მიკროსქემათა შესასვლელები.***

მიკროსქემების წარმოდგენის ***პირველ და მეორე დონეებზე*** საერთოდ საჭირო არ არის რაიმე ვიცოდეთ მიკროსქემების შესასვლელების შესახებ. შესასვლელი განიხილება, როგორც როგორც უსასრულოდ დიდი წინაღობა, რომელიც გავლენას ვერ ახდენს მიკროსქემასთან მიერთებულ გამოსასვლელებზე.

მიკროსქემების წარმოდგენის ***მესამე, ელექტრული მოდელის დონეზეც*** არაა საჭირო რაიმე ვიცოდეთ როგორც მიკროსქემის შიგა აგებულების, ისე შესასვლელების სქემოტექნიკის შესახებ. საკმარისია ვიცოდეთ, რომ შესასვლელზე ლოგიკური ***0***-ის ტოლი სიგნალის მიწოდებისას ამ შესასვლელიდან გამოდის $I_{I.L}$–ზე არაუმეტესი, ხოლო ლოგიკური ***1***-ის ტოლი სიგნალის მიწოდებისას – $I_{I.H}$–ზე არაუმეტესი დენი. მიკროსქემის სწორად მუშაობისათვის საკმარისია, რომ ლოგიკური ***0***-ის შესაბამისი შესასვლელი სიგნალის ძაბვის დონე $U_{I.L}$–ზე, ხოლო ლოგიკური ***1***-ის შესაბამისი შესასვლელი სიგნალის ძაბვის დონე – $U_{I.H}$–ზე ნაკლები იყოს.

განსაკუთრებულ შემთხვევას მიეკუთვნება სიტუაცია, როდესაც რომელიმე შესასვლელი არ არის მიერთებული არც ერთ გამოსასვლელთან; ასეთ შესასვლელს ეწოდება ***დაკიდებული შესასვლელი***. ასეთი შესასვლელის არ-

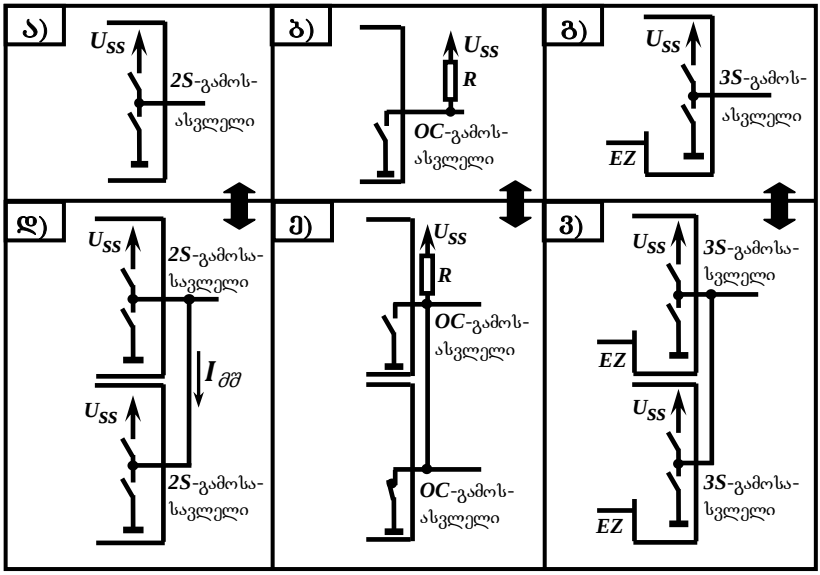
სებობისას შეიძლება მიკროსქემამ არ იმუშაოს, ან იმუშაოს არასტაბილურად. ამიტომ იმაზე დამოკიდებულებით, თუ როგორი ლოგიკური დონის არსებობაა საჭირო, რეკომენდებულია გამოუყენებელი შესასვლელებს მივუერთოთ მიკროსქემის კვების U_{SS} ძაბვა ან საერთო სადენი (მიწა). ***ტტლ***-ტექნოლოგიით რეალიზებული ზოგიერთი მიკროსქემისათვის საჭიროა კვების ძაბვა მივუერთოთ არა უშუალოდ, არამედ დაახლოებით ***1*** კოლოომი წინაღობიანი რეზისტორით (***20***-მდე შესასვლელისათვის საკმარისია ერთი ასეთი რეზისტორის გამოყენება).

ტტლ მიკროსქემის მიუერთებელ შესასვლელებზე ფორმირდება ***1,5-1,6*** ვოლტამდე ძაბვა, რომელსაც ზოგჯერ ***დაკიდებულ პოტენციალს*** უწოდებენ. ჩვეულებრივ ამ დონეს მიკროსქემა ლოგიკურ ***1***-ად აღიქვამს, მაგრამ ყოველთვის ამ იმედზე ყოფნა არ შეიძლება. ***კლშნ*** მიკროსქემების დაკიდებულ შესასვლელებზე წარმოქმნილი პოტენციალი მიკროსქემამ შეიძლება აღიქვას როგორც ლოგიკურ ***1***-ად, ისე ლოგიკურ ***0***-ადაც. მუდმივია მხოლოდ ის მოთხოვნა, რომ ***ნებისმიერი დაკიდებული შესასვლელი სადღაც უნდა იყოს მიერთებული.*** მიუერთებლად შეიძლება დავტოვოთ ***ტტლ***-ტექნოლოგიით დამზადებული მიკროსქემის მხოლოდ ის დაკიდებული შესასვლელი, რომლის მდგომარეობას მიკროსქემის მოცემული ჩართვის დროს მნიშვნელობა არა აქვს. ეს არ შეიძლება ***კლშნ***-ტექნოლოგიით დამზადებული მიკროსქემისათვის! ***მიკროსქემების გამოსასვლელები*** შესასვლელებისაგან პრინციპულად განსხვავდება იმით, რომ მათი თავისებურებების გაითვალსწინება მიკროსქემების წარმოდგენის პირველსა და მეორე დონებზეც.

არსებობს შემდეგი სამი სახის გამოსასვლელი კასკადი, რომლებიც თვისობრივად განსხვავდება ერთმანეთისაგან როგორც საკუთარი მახასიათებლებით, ისე გამოყენების სფეროებითაც: ***1. სტანდარტული ანუ ორი მდგომარეობიანი გამოსასვლელი***, რომელსაც ხშირად უწოდებენ ***2S*** სახის, ხოლო იშვიათად – ***ტტლ*** სახის გამოსასვლელს; ***2. ღია კოლექტორიანი გამოსასვლელი***, რომელსაც ***OC*** სახის გამოსასვლელს უწოდებენ; ***3. სამი მდგომარეობიანი ანუ გამორთვის შესაძლებლობიანი გამოსასვლელი***, რომელსაც ***3S*** სახის გამოსასვლელს უწოდებენ.

▲ სტანდარტულ ***2S*** გამოსასვლელს აქვს მხოლოდ ორი, კერძოდ, ლოგიკური ***0***-სა და ლოგიკური ***1***-ის შესაბამისი მდგომარეობა. ***ორივე ეს მდგომარეობა აქტიურია***, ე. ი. ორივე ამ მდგომარეობის დროს გამოსასვლელი I_{OL} და I_{OH} დენის სიდიდე შეიძლება მნიშვნელოვანი იყოს. მიკროსქემის წარმოდგენის პირველ და მეორე დონეზე შეიძლება ჩავთვალოთ, რომ ასეთი გამოსასვლელი შედგება მიმდევრობით შეერთებული ორი ამომრთველისაგან (იხ.ნახ.***1.5ა***), რომლებიც მორიგეობით შეირთვება; ზედა ამომ-

რთველის ჩართვას შეესაბამება ლოგიკური ***1***, ხოლო ქვედა ამომრთველის ჩართვას – ლოგიკური ***0***.



ნახ. 1.5. *ციფრული მიკროსქემების სამი სახის გამოსასვლელი ხაზისა (**ა,ბ,გ**) და მათი გაერთიანების (**დ,ე,ვ**) სქემები მიკროსქემების წარმოდგენის პირველ და მეორე დონეებზე*

▲ ღია კოლექტორიან ***OC*** გამოსასვლელსაც ორი მდგომარეობა აქვს, რომელთაგანაც აქტიურია მხოლოდ ლოგიკური ***0***-ის შესაბამისი მდგომარეობა: ამ მდგომარეობაშია უზრუნველყოფილი დიდი გამოსასვლელი I_{OL} დენის არსებობა. მეორე მდგომარეობა დაიყვანება იმაზე, რომ გამოსასვლელი მთლიანად გამორთულია მასთან შესაერთებელი გამტარისაგან. ეს მეორე მდგომარეობა შეიძლება გამოვიყენოთ ლოგიკური ***1***-სათვის, ოღონდ ***OC*** გამოსასვლელს და კვების ძაბვას შორის უნდა ჩავრთოთ დაახლოებით ***100*** ომი წინაღობის მქონე ***R*** რეზისტორი, რომელსაც ***pulup*** (ამწევ), ანუ ***სადატვირთო რეზისტორს*** უწოდებენ. მიკროსქემის წარმოდგენის პირველ და მეორე დონეებზე ასეთი გამოსასვლელი შეიძლება წარმოდგენილი იქნეს ერთი ამომრთველის სახით (ნახ. ***1.5,ბ***), რომლის ჩართულ მდგომარეობას შეესაბამება ლოგიკური ***0***-ის სიგნალი, ხოლო განრთულ მდგომარეობას – გათიშული, პასიური მდგომარეობა. ***R*** რეზისტორის სიდიდეზე დამოკიდებულია ***0*** მდგომარეობიდან მდგომარეობა ***1***-ში გადასვლისათვის საჭირო დრო-

ის სიდიდე. ეს უკანასკნელი გავლენას ახდენს t_{LH} დაყოვნებაზე, მაგრამ რეზისტორების ჩვეულებრივ გამოყენებადი ნომინალებისათვის აღნიშნული გავლენა მნიშვნელოვანი არ არის.

▲ სამი მდგომარეობიანი ***3S*** გამოსასვლელი ძალიან ჰგავს სტანდარტულ (***2S)*** გამოსასვლელს, ოღონდ ორ აქტიურ მდგომარეობას ემატება მესამე პასიური მდგომარეობა, რომლის დროსაც გამოსასვლელი შეიძლება მომდევნო სქემისაგან განრთულად ჩავთვალოთ. მიკროსქემის წარმოდგენის პირველ და მეორე დონეზე ასეთი გამოსასვლელი შეიძლება ჩავთვალოთ ორი ამომრთველისაგან შემდგარ გამოსასვლელად (ნახ.***1.5,ვ***), რომლებიც შეიძლება არა მარტო თანამიმდევრობითად ჩაირთოს და მოახდინოს ლოგიკური ***0***-ისა და ***1***-ის ფორმირება, არამედ ერთდროულადაც გაითიშოს. ამ მესამე მდგომარეობას ეწოდება ***მაღალიმპენდასური*** ანუ ***Z–მდგომარეობა***. მესამე ***Z***–მდგომარეობაში გადასაყვანად გამოიყენება სპეციალური მმართველი შესასვლელი, რომელსაც აღინიშნება, როგორც ***OE*** (***O****utput* ***E****nable* - გამოსასვლელის ნებართვა) ან ***EZ*** (***E****nable* ***Z****-state*).

აქტიური და პასიური მდგომარეობებიანი გამოსასვლელების მოხერხებულად გაერთიანებისათვის სტანდარტული ***2S*** გამოსასვლელის გარდა დამუშავებული იქნა კიდევ ორი, კერძოდ ***OS*** და ***3S*** გამოსასვლელები. მოწყობილობის შესასვლელზე თუ საჭიროა რიგრიგობით იქნეს მიწოდებული სიგნალები მეზობელი მოწყობილობის ორი სხვადასხვა გამოსასვლელიდან (ნახ. ***1.5,დ***), მაშინ ამ გამოსასვლელებად არ გამოდგება ***2S*** გამოსასვლელი; შეიძლება ***OC*** (ნახ. ***1.5,ე***) ან ***3S*** (ნახ.***1.5,ვ***) გამოსასვლელის გამოყენება.

ორი ან რამდენიმე ***2S*** შესასვლელის გაერთიანებისას (ნახ.***1.5,დ***) დასაშვებია წარმოიშვას სიტუაცია, როდესაც ერთ-ერთი გამოსასვლელი შეეცდება გასცეს ლოგიკური ***1***, ხოლო მეორე – ლოგიკური ***0*** სიგნალი. ამ შემთხვევაში ლოგიკური ***1***-ის გამცემ ზედა ჩაკეტილ ამომრთველისა და ლოგიკურ ***0***-ის გამცემი ქვედა ჩაკეტილ ამომრთველს შორის არსებულ ხაზში გაივლის მოკლედ შერთვის დაუშვებლად დიდი $I_{\partial\partial}$ დენი. ეს არის ავარიული სიტუაცია, რომლის დროსაც ზუსტად არ არის განსაზღვრული მისაღები გამოსასვლელი ლოგიკური სიგნალის დონე – იგი შეიძლება აღქმული იქნეს როგორც ***0***-ად, ისე ***1***-ად. მოკონფლიქტო გამოსასვლელები შეიძლება მწყობრიდან გამოვიდეს და დაარღვიოს როგორც მიკროსქემის, ისე მთლიანად სქემის მუშაობა.

მიკროსქემების წარმოდგენის მესამე დონეზე აუცილებელია გავითვალისწინოთ, რომ გამოსასვლელებში არსებული ამომრთველები (იხ. ნახ.***1.5,ა,ბ,გ) წარმოადგენს არა უბრალოდ ტუმბლერებს (როგორც ეს გვქონდა*** პირველი ორი დონის დროს), არამედ საკუთარი სპეციფიკური მახასიათებლების მქონე ტრანზისტორულ გასაღებებს. ხშირ შემთხვევაში საკმარისია ვიცოდეთ

მოცემული გამოსასვლელი თუ როგორ დენს წარმოქმნის ლოგიკური ***0***-ისა და ლოგიკური ***1***-ის დროს. ლოგიკური ***0***-ის დროს წარმოშობილი დენი აღვნიშნოთ $I_{0.L}$, ხოლო ლოგიკური ***1***-ის დროს წარმოშობილი დენი $I_{0.H}$ სიმბოლოთი. ამ დენების სიდიდეები მოცემულ შესასვლელთან მიერთებული ყველა მიკროსქემების შესასვლელი დენის ჯამს არ უნდა აჭარბებდეს. მიკროსქემის ერთ გამოსასვლელთან მიერთებებული ელემენტების რაოდენობა განისაზღვრება ამ მიკროსქემის ***განშტოების კოეფიციენტით*** ანუ ***სადატვირთო უნარით.*** არსებობს ჩვეულებრივი და ამაღლებული სადატვირთო უნარის მქონე მიკროსქემები. მეორე სახის მიკროსქემის სადატვირთო უნარი ორჯერ და უფრო მეტად აღემატება პირველი სახის მიკროსქემის სადატვირთო უნარს. ***3S*** სახის გამოსასვლელებიან მიკროსქემებს აქვს ამაღლებული სადატვირთო უნარი (ე. ი. შეუძლია უფრო მაღალი გამოსასვლელი დენების ფორმირება). ***2S*** და ***OC*** გამოსასვლელიან მიკროსქემებს შეიძლება ჰქონდეს როგორც ჩვეულებრივი, ისე ამაღლებული სადატვირთო უნარი.

მესამე დონეზე გარდა ზემოთ აღნიშნულისა, აუცილებელია გავითვალისწინოთ, თუ რა სიდიდის გამოსასვლელი და ძაბვების ფორმირება შეუძლია მიკროსქემას. ***OC***-ს გამოსასვლელები გათვალისწინებულია ლოგიკური ***1***-ის როგორც ჩვეულებრივ (U_{0H}=U_{SS}=***5 V***), ისე ლოგიკური ***1***-ის ამაღლებულ (***30*** ვოლტამდე) ძაბვაზე. ბოლო შემთხვევაში ამ გამოსასვლელის გარე რეზისტორი მიერთებულია ამაღლებული ძაბვის წყაროსთან (***ნახ. 1.5,ა,ბ,გ***).

1.4.3. სალტური კავშირების ორგანიზების საფუძვლები

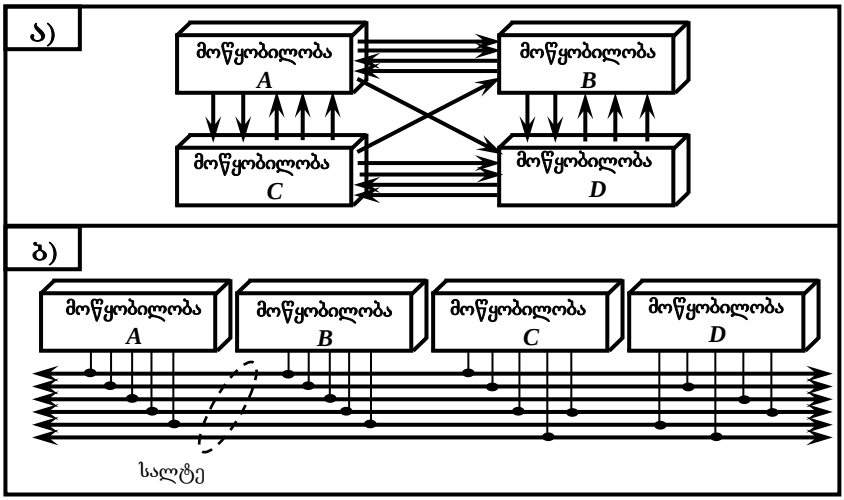
გარკვეულ სისტემაში შემავალი ელემენტების დასაკავშირებლად შეიძლება გამოყენებული იქნეს კავშირგაბმულობის კლასიკური და სალტური სტრუქტურები. განვიხილოთ ორივე მათგანი.

კავშირის კლასიკური სტრუქტურის დროს (ნახ. ***1.6,ა***) მოწყობილობებს შორის ყველა სიგნალი და კოდი კავშირის განცალკევებული არხით გადაიცემა. სისტემაში შემავალი თითოეული მოწყობილობა საკუთარ სიგნალებსა და კოდებს სხვა მოწყობილობებისაგან დამოუკიდებლად გადასცემს. ასეთ სისტემაში ვიღებთ კავშირის ბევრ ხაზსა და ინფორმაციის გაცვლის მრავალფეროვან პროტოკოლებს.

კავშირების სალტური სტრუქტურის დროს (ნახ. ***1.6,ბ***) მოწყობილობებს შორის ყველა სიგნალი კავშირის ერთი და იმავე არხით, ოღონდ სხვადასხვა დროს გადაიცემა (ამას ***მულტიპლექსური გადაცემა*** ეწოდება). ამის შე-

დეგად მნიშვნელოვნად მცირდება კავშირის არხების რაოდენობა და მარტივდება ინფორმაციის გაცვლის წესები (პროტოკოლები). სიგნალების გადასაცემ კავშირის ხაზის ერთობლიობას ***სალტე*** (ინგ. ***bus***) ეწოდება.

კავშირების სალტური სტრუქტურის დროს საინფორმაციო ნაკადები ადვილად გაიგზავნება საჭირო მიმართულებით, მაგალითად, შესაძლებელია ერთ პროცესორში მათი გატარება, რაც ძალიან მნიშვნელოვანია მიკროპროცესორული სისტემისათვის. სამაგიეროდ, სალტური სტრუქტურის დროს ინფორმაცია კავშირის არხებში მიმდევრობითად (ერთმანეთის მიყოლებით და არა ერთდროულად) გადაიცემა, რაც კავშირების კლასიკურ სტრუქტურასთან შედარებით ამცირებს სისტემის სწრაფმოქმედებას.



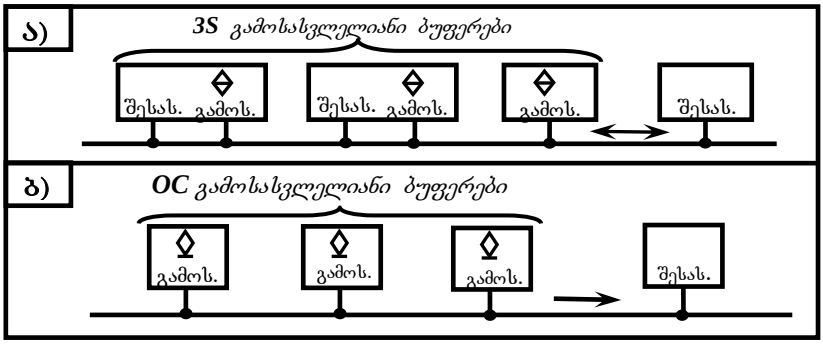
ნახ. 1.6. *კავშირის კლასიკური (**ა**) და სალტური (**ბ**) სტრუქტურები*

კავშირების სალტური სტრუქტურის დიდი ღირსებაა ის, რომ სალტესთან მიერთებულმა ყველა მოწყობილობამ ერთი და იმავე წესებით (ერთი და იმავე პროტოკოლით) უნდა მიიღოს და გადასცეს ინფორმაცია. ამიტომ შესაძლებელია მოვახდინოთ სალტით ინფორმაციის გაცვლაში მონაწილე ყველა მოწყობილობის უნიფიცირება.

კავშირების სალტური სტრუქტურის მნიშვნელოვანი ნაკლია ის, რომ კავშირის ხაზებთან მოწყობილობები პარალელურადაა მიერთებული. ამიტომ ნებისმიერი მოწყობილობის ნებისმიერ მტყუნებას, რომელიც აზიანებს კავშირის ხაზს, შეუძლია მთელი სისტემა მწყობრიდან გამოიყვანოს. ამ მიზეზის გამო საკმაოდ რთულია კავშირის სალტური სტრუქტურის მქონე სისტემის

გამართვა და ჩვეულებრივ იგი სპეციალური მოწყობილობის გამოყენებით უნდა მოხდეს.

არსებობს კავშირის შემდეგი სამი სახის ხაზი: ერთმიმართული, ორმხრივმიმართული და მულტიპლექსური ხაზი. 1) ***ერთმიმართული*** ხაზები სიგნალებს მხოლოდ ერთი მიმართულებით ატარებს. ასეთი ხაზების მეშვეობით შეიძლება მიკროსქემის ერთი გამოსასვლელი რამდენიმე შესასვლელს დავუკავშიროთ. ამ გამოსასვლელისათვის ზემოთ განხილული სამივე კასკადის გამოყენებაა შესაძლებელი. ***2) ორმხრივმიმართულ ხაზებში*** სიგნალები ორი ურთიერთსაწინააღმდეგო მიმართულებით ვრცელდება. მათთან შეიძლება მიკროსქემების ***3S*** და ***ОС*** ტიპის რამდენიმე გამოსასვლელი მივაერთოთ (ნახ.***1.7,ა***), ამიტომ ასეთ ხაზებთან მიერთებულ ყველა მიკროსქემას ***3S*** ან ***ОС*** ტიპის გამოსასვლელები აქვს.



ნახ.1.7. *ორმხრივმიმართული (ა) და მულტიპლექსური (ბ) ხაზები*

მიკროსქემის გამოსასვლელის ტიპი სპეციალური ნიშნით აღინიშნება: ***3S*** გამოსასვლელიანი მიკროსქემაზე გვაქვს გადახაზული რომბის (იხ.ნახ.***1.7,ა***), ხოლო ***ОС*** გამოსასვლელიან მიკროსქემაზე – ქვემოთ გახაზული რომბის (იხ.***1.7,ბ***) გამოსახულება. ***2S*** გამოსასვლელიან მიკროსქემაზე (ბუფერზე) არავითარი დამატებითი აღნიშვნა არ არსებობს. ორმხრივმიმართული ხაზებით გაერთიანებულ მიკროსქემებს შორის კონფლიქტების ასაცილებლად საჭიროა მუშაობდეს მხოლოდ ერთი აქტიური გამოსასვლელი, ხოლო დანარჩენი გამოსასვლელები იმყოფებოდეს მესამე (მაღალიმპენდასურ) მდგომარეობაში.

მულტიპლექსირება ერთი და იმავე ხაზით სხვადასხვა დროს სხვადასხვა სიგნალების გადაცემას ნიშნავს. მულტიპლექსირების ძირითადი დანიშნულება შემაერთებელი ხაზების საერთო რაოდენობის შემცირებაა. ორმხრივმიმართული ხაზი აუცილებლად მულტიპლექსირებულია, ერთმიმართული ხაზი

კი შეიძლება იყოს ან არ იყოს მულტიპლექსირებული. ორივე შემთხვევაში მას შეიძლება მივუერთოთ რამდენიმე გამოსასვლელი, რომელთაგანაც დროის მოცემულ მომენტში მხოლოდ ერთი გამოსასვლელია აქტიურ მდგომარეობაში. დანარჩენი გამოსასვლელები ამ დროს გამოირთვება (გადადის პასიურ მდგომარეობაში). ორმხრვმიმართული ხაზებისაგან განსხვავებით, ბუფერების საფუძველზე აგებულ მულტიპლექსურ ხაზს მხოლოდ ერთი შესასვლელი, მაგრამ აუცილებლად რამდენიმე ***3S*** ან ***OS*** გამოსასვლელი უნდა მივუერთოთ (ნახ.***1.7,ბ***). მულტიპლექსირებული ხაზები შეიძლება ავაგოთ არა მარტო ბუფერებით, არამედ მულტიპლექსორების მიკროსქემებითაც.

II თავი
პროცესორის მიერ შესასრულებელი არითმეტიკული და ლოგიკური ოპერაციები

«ნებისმიერი საგანი შეიძლება რიცხვის სახით წარმოვადგინოთ»;

«სამყაროს მართავენ რიცხვები; სამყაროში ყველაფერი არის რიცხვი».

პითაგორა

2.1. რიცხვული ფორმით ინფორმაციის წარმოდგენის საკითხისათვის

წინა საუკუნის ბოლო წლებიდან მსოფლიოს ეკონომიკაში სოკოკოებივით ამოიზარდა ***ავტომატიზებული საწარმოები***, რომლებშიც საწარმოო პროცესის ყველა ძირითადი და დამხმარე ოპერაციებს საწარმოო პროცესის მართვის სრული ავტომატიზაციის ფონზე ასრულებს ავტომატური მანქანები (დაზგები, ჩარხები). ***მიკროპროცესორული სისტემა*** მეტაფორულად შეიძლება ისეთ ავტომატიზებულ საწარმოოდ მივიჩნიოთ, რომელშიც ინფორმაციის სახის წარმოდგენილი ნედლეულის ავტომატიზებულად დამუშავების გზით იწარმოება თვისობრივად ახალი ინფორმაცია, რომელიც ამ წარმოების მიერ დამზამზადებულ პროდუქციას წარმოადგენს. ამ უკანასკნელის მისაღებად ნედლეულის (ძველი ინფორმაციის) უშუალო დამუშავებით დაკავებულია პროცესორი. მან რომ თავისი ფუნქციის შესრულება შეძლოს, აუცილებელია ინფორმაცია წარმოდგენილი იყოს ისეთი ფორმით, რომლის გადამუშავება შესაძლებელია.

ზემოთ აღნიშნულიდან წამოიჭრა ისეთი ფორმის განსაზღვრის ამოცანა, რომელიც, ***უპირველეს ყოვლისა***, შეიძლება მიეცეს ნებისმიერი სახის ინფორმაციას და, ***მეორე მხრივ***, ამ ფორმით წარმოდგენილი ინფორმაციის გადამუშავება შეეძლოს პროცესორს. მოცემული

ამოცანის გადაწყვეტისას კიდევ ერთხელ დადასტურდა იმ მტკიცების ჭეშმარიტება, რომლის თანახმადაც ნებისმიერი ახალი კარგად დავიწყებული ძველია. ჯერ კიდევ ჩვენს წელთაღრიცხვამდე ***VI-V*** საუკუნეში ძველი ბერძენი ფილოსოფოსი ***პითაგორე სამოსელი*** (ჩვ.წ.აღ ***570-490*** წწ.) ამტკიცებდა, რომ ნებისმიერი საგანი შეიძლება რიცხვის სახით წარმოვადგინოთ; ოღონდ ეს წარმოდგენა პითაგორეს მიხედვით თუ მისტიური ბუნების იყო, ***XX*** საუკუნის პირველ ნახევარში იგი მეცნიერული ფორმით ჩამოყალიბდა.

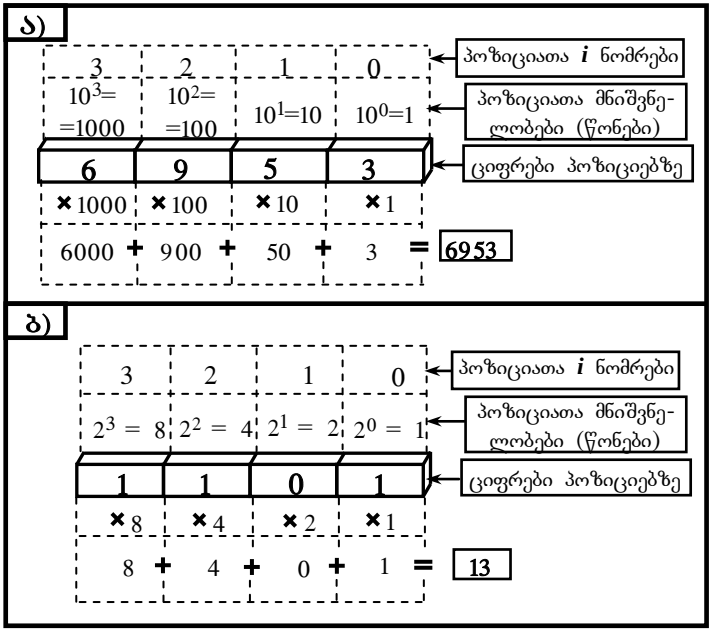
ადამიანის ტვინი წარმოქმნილია დაახლოებით ***86 მილიარდი*** ნეირონისაგან, რომლის უხეშ ტექნიკურ ანალოგად შეიძლება ***ტრანზისტორი*** მივიჩნიოთ. მაშასადამე, ადამიანის ტვინი შეიძლება ურთულეს ბუნებრივ ციფრულ სისტემად ჩავთვალოთ (შედარებისათვის აღვნიშნავთ, რომ კომპანია ***Intel***–ს მიერ დამზადებული ყველაზე მძლავრი ***Tukwila*** სახელწოდების მიკროპროცესორი მხოლოდ ***ორ მილიარდზე*** მეტ ტრანზისტორს შეიცავს). ნეირონი შეიძლება იმყოფებოდეს მშვიდ ან აგზნებულ მდგომარეობაში, რომლებიც პირობითად შესაბამისად ***0***-ითა და ***1***-ით შეიძლება აღვნიშნოთ.

ადამიანის გრძნობის ორგანოებზე გარე საგნის ზემოქმედებებისას წარმოიშობა გარკვეული ნეიროსიგნალები. ისინი ზემოქმედებს თავის ტვინის გარკვეულ ნეირონებზე და მათ მდგომარეობა ***0***-იდან გადაიყვანს მდგომარეობა ***1***-ში; ***0***-ებისა და ***1***-ების ერთობლიობა წარმოადგენს ორობით რიცხვს, ე.ი. ტვინში ბუნებრივად წარმოიქმნება ორობითი რიცხვი, რომელიც ზემოთ აღნიშნული საგნის აბსტრაქტულ სახეს წარმოადგენს. მაშასადამე, ჩვენ მიერ აღქმული ნებისმიერი ინფორმაცია თავის ტვინში ფიქსირდება გარკვეული ორობითი რიცხვის სახით. ვინაიდან უმაღლესად ორგანიზებული ბუნებრივი სისტემა – ტვინი – ჩვენს გარშემო არსებულ ინფორმაციას ორობითი რიცხვების სახით აღიქვამს, ამიტომ შეგვიძლია ვამტკიცოთ, რომ ***თვლის ორობითი სისტემა ბუნებრივი სისტემაა, რომელიც ნებისმიერი აღქმადი ინფორმაციის წარმოდგენის საშუალებას იძლევა.*** თვლის ნებისმიერი სხვა სისტემა, მათ შორის ჩვენთვის ბავშვობიდანვე შესისხლხორცებული ათობითი სისტემა, ადამიანის მიერ ხელოვნურად შექმნილი სისტემაა. მოხდენილად შენიშნა ცნობილმა საბჭოთა მატემატიკოსმა ***ნ. ნ. ლუზინმა***, რომ «თვლის ათობითი სისტემის უპირატესობები არა მათემატიკური, არამედ ზოოლოგიურია. ხელებზე ათის

ნაცვლად რვა თითი რომ გვქონოდა, კაცობრიობა დაიწყებდა თვლის რვაობითი სისტემის გამომყენებას».

2.2. ორობითი რიცხვები

თვლის სისტემა ეწოდება რიცხვების ჩაწერის სიმბოლურ მეთოდს, რომელიც წერითი სიმბოლოებით (ციფრებით) რიცხვების წარმოდგენის საშუალებას იძლევა. განასხვავებენ თვლის პოზიციურ და არაპოზიციურ სისტემებს. ***თვლის პოზიციურ სისტემებში*** ციფრების მნიშვნელობა იცვლება რიცხვში ამ ციფრის მიერ დაკავებულ პოზიციაზე დამოკიდებულებით, ხოლო ***არაპოზიციურ სისტემებში*** პოზიციის ცვლილება არ ცვლის ციფრის მნიშვნელობას.



ნახ.2.1 *ათობითი (ა) და ორობითი (ბ) რიცხვების აგების პრინციპის მაილუსტრირებელი ნახაზები*

თვლის პოზიციური სისტემები ერთმანეთისაგან განსხვავდება რიცხვების ჩასაწერად მათში გამოყენებული ციფრების (წერითი სიმბო-

ლოების) რაოდენობის მიხედვით. ***g*** რაოდენობის ციფრების მქონე თვლის სისტემას ***g***–ობითი სისტემა ეწოდება, სადაც ***g*** ორზე არანაკლები (***g≤2***) ნატურალური რიცხვია. აღნიშნულ ***g*** რიცხვს თვლის სისტემის ფუძეს უწოდებენ; მაშასადამე, სისტემის ***თვლის ფუძე*** ეწოდება სისტემაში გამოყენებული ციფრების რაოდენობის მაჩვენებელ ორზე არანაკლებ ნატურალურ რიცხვს.

თვლის g-ობითი სისტემა ეწოდება ***g*** რაოდენობის წერითი ნიშნების დახმარებით რიცხვების სიმბოლური ჩაწერის მეთოდს. ***g***-ობითი სისტემით ჩაწერილ რიცხვებს ***g-ობითი რიცხვები*** ეწოდება.

ა)

ათობითი	ორობითი	ათობითი	ორობითი
10 1	8 4 2 1	10 1	8 4 2 1
0	0	8	1 0 0 0
1	1	9	1 0 0 1
2	1 0	0	1 0 1 0
3	1 1	1 1	1 0 1 1
4	1 0 0	1 2	1 1 0 0
5	1 0 1	1 3	1 1 0 1
6	1 1 0	1 4	1 1 1 0
7	1 1 1	1 5	1 1 1 1

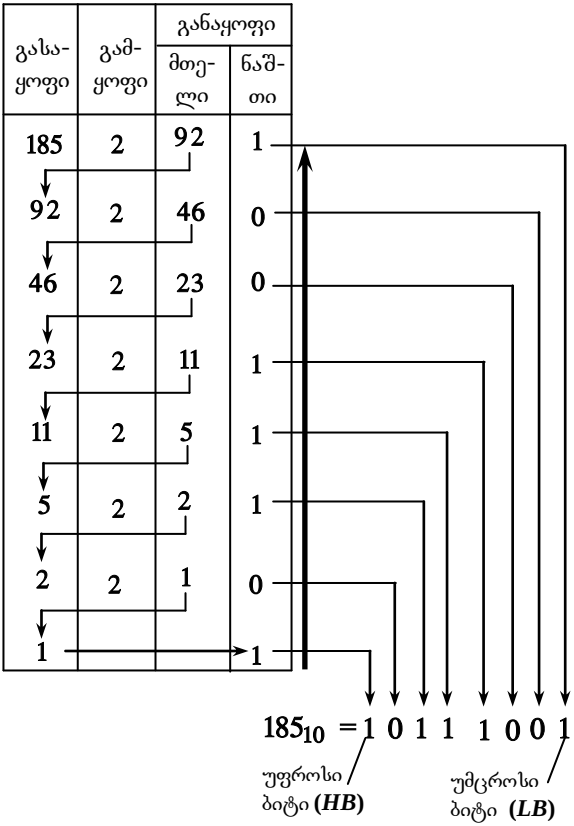
ბ)

ფუძის ხარისხი	2^7		2^6		2^5		2^4		2^3		2^2		2^1		2^0
პოზიციის მნიშვნელობა	128		64		32		16		8		4		2		1
ორობითი	**1**		**1**		**0**		**1**		**1**		**0**		**0**		**1**
ათობითი	128	+	64	+	0	+	16	+	8	+	0	+	0	+	1 = 217

ნახ.2.2. *ათობითი რიცხვები და მათი ორობითი ეკვივალენტები (**ა**), ორობით-ათობითი გარდაქმნა (**ბ**)*

ყოველდღიურ საქმიანობაში ჩვენ ***10***-ობით რიცხვებს ვიყენებთ. იგი ***0,1,...9*** ციფრებითაა შედგენილი (მათი რაოდენობაა ***10***), რომლებსაც

ათობითი ციფრები ანუ ***დიჭები*** ეწოდება (***D**ecimal dig**IT***). ***შ.2.1,ა*** ნახაზზე ნაჩვენებია, რომ, მაგალითად ***10***-ობითი ***6953*** რიცხვი შედგება ***6, 9, 5, 3*** ციფრებისაგან, რომელთა მნიშვნელობები დამოკიდებულია მათ მიერ დაკავებული პოზიციის ***i*** ნომერზე და გამოთვლება როგორც $g^i=10^i$; აღნიშნულის გამო მოყვანილი ციფრების მნიშვნელობები შესაბამისად ექვსი ათასეულის, ცხრა ასეულის, ხუთი ათეულისა და სამი ერთეულის ტოლია. საბოლოოდ რიცხვის მნიშვნელობაა ექვსიათასს პლუს ცხრაასი, პლუს ორმოცდაათი, პლუს სამი.

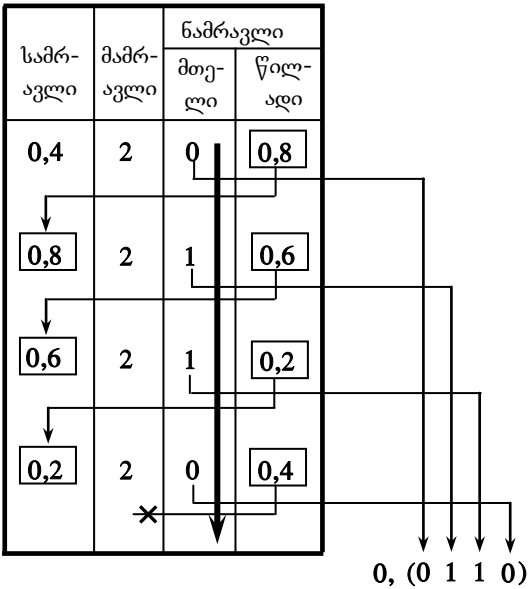


ნახ.2.3. *მთელი ათობითი რიცხვის ორობით რიცხვად გარდაქმნა*

ანალოგიურად გამოითვლება ნებისმიერი ***g***–ობითი რიცხვის მნიშვნელობა. შედარებით დაწვრილებით განვიხილოთ თვლის ***2***-ობითი

სისტემა. მისი ფუძეა ***g=2***, ე.ი. შეიცავს ორ ციფრს (წერით სიმბოლოს) – ***0***-სა და ***1***-ს. მათ ბიტები (***B**inary dig**IT***) ეწოდება. ციფრულ ელექტრონულ სისტემებში ბიტი ***0*** წარმოიდგინება ***LOW*** (დაბალი), ხოლო ბიტი ***1*** – ***HIGH*** (მაღალი) ძაბვით, თუმცა პრინციპულად შესაძლებელია პირიქითაც მოხდეს. ***2.1,ბ*** ნახაზზე მოყვანილია პირველი ოთხი ორობითი პოზიციის ათობითი მნიშვნელობები. ამავე ნახაზიდან ჩანს, რომ ორობითი ***1101*** (წაიკითხება ასე: ერთი, ერთი, ნული, ერთი) რიცხვს შეესაბამება ათობითი რიცხვი ***13.***

2.2,ა ნახაზზე მოყვანილია ***0***-დან ***15***-ის ჩათვლით ათობითი რიცხვები და მათი ორობითი ეკვივალენტები. კომპიუტერის სფეროში მუშაობის მსურველებმა ისინი ზეპირად უნდა იცოდნენ.



ნახ.შ.2.4. *ათწილადის ორობით ეკვივალენტად გარდაქმნა.*

ორობითი ***11011001*** (წაიკითება ასე: ერთი, ერთი, ნული, ერთი, ერთი, ნული, ნული, ერთი) რიცხვის ათობით ეკვივალენტად გარდაიქმნება ***2.2.,ბ*** ნახაზზე მოყვანილი პროცედურის საშუალებით. ნახაზზე თითოეული პოზიციის ათობითი მნიშვნელობები (წონები) ამ პოზიციებში არსებული ბიტების ზემოთაა მითითებული. მათზე პოზიციებში არსებული ბიტების გამრავლებითა და მიღებული ნამრავ-

ლების შეკრებით მიიღება განხილული ორობითი რიცხვის შესაბამისი ათობითი რიცხვი: ***128 + 64 +0 +16 +8 +0 +0 +1 = 117.***

რიცხვის ჩანაწერში თვლის სისტემის ფუძე ინდექსით არის მითითებული. ამგვარად, 11011001_2 რიცხვი ორობითია (ანუ, მისი ფუძეა ***2***), ხოლო რიცხვი 217_{10} – ათობითია. ***2.2,ბ*** ნახაზის თანახმად $11011001_2 = 217_{10}$.

მთელი ათობითი, მაგალითად, 185, რიცხვის ორობით რიცხვად გარდაქმნისათვის ჯერ იგი უნდა გავყოთ ***2***-ზე და ცალ-ცალკე ჩავწეროთ მიღებული მთელი ნაწილი და ნაშთი, როგორც ეს ***2.3*** ნახაზზეა ნაჩვენები. მიღებული მთელი ნაწილი ნახაზზე ნაჩვენები ისრის შესამისად გადავანაცვლოთ გასაყოფის ადგილზე; აღწერილი ოპერაცია მანამ გავიმეოროთ, სანამ მიღებული განაყოფის მთელი ნაწილი ***1***-ის (იხ. ***2.3*** ნახაზის ბოლოსწინა მწკრივი) ან ***0***-ის ტოლი არ გახდება. ფორმირებული ნაშთების ქვემოდან ზემოთ ამოწერით მივიღებთ საძებნ ორობით რიცხვს. განხილულ მაგალითში მიღებული ნაშთების ქვემოდან ზემოთ ამოწერით მიიღება ორობით 10111001_2 რიცხვი, რომელიც 185_{10} რიცხვის ეკვივალენტურია, ე. ი. $185_{10} = 10111001_2$.

ათწილადის ორობითი ეკვივალენტის პოვნისას ***2***-ზე გაყოფის ოპერაცია იცვლება ***2***-ზე გამრავლების ოპერაციით. მაგალითად, $0,4_{10}$-ის ორობითი ეკვივალენტის საპოვნელად $0,4_{10}$-ს ვამრავლებთ ***2***-ზე (ნახ.***შ.2.4***), რაც გვაძლევს მთელ ***0***-სა და წილად ***0,8***-ს. შემდეგ წილადი ***0,8*** ნახაზზე ნაჩვენები ისრის შესაბამისად გადაგვაქვს სამრავლის ადგილზე და ხელახლა ვამრავლებთ ***2***-ზე. აღწერილ პროცესს ვიმეორებთ მანამ, სანამ წილადების სვეტში არ მივიღებთ ***0***-ს, ან ისეთ ***ათწილადს***, რომელიც უკვე გვქონდა სამრავლად გამოყენებული. პირველ შემთხვევაში ვიღებთ არაპერიოდულ, ხოლო მეორე შემთხვევაში – პერიოდულ ორობით წილადს. ჩვენს მაგალითში მეოთხე მწკრივში ნამრავლ ათწილადად მივიღეთ უკვე გამოყენებული ***0,4,*** რის გამოც გამრავლების პროცესს ვწყვეტთ (პროცესის გაგრძელებისას შედეგები გამეორდებოდა). საძებნი ორობითი წილადის გამოსახულების მისაღებად ***მთელი*** ნამრავლები უნდა ამოვწერთ ზემოდან ქვემოთ; განხილულ მაგალითში (იხ.ნახ***2.4***) ვიღებთ $0,(0110)_2$-ს ე. ი. $0,8_{10} = 0,(0110)_2$.

$185,4_{10}$ რიცხვის ორობითი ეკვივალენტის პოვნისათვის ჯერ უნდა ვიპოვოთ 185_{10}-ის, ხოლო შემდეგ – $0,4_{10}$-ის ორობითი ეკვივალენტები და მიღებული შედეგები გავაერთიანოთ, ე. ი. მივიღებთ შემდეგ შედეგს: $185,4_{10} = 10111001,(0110)_2$.

2.3. თექვსმეტობითი რიცხვები

ტიპური მიკრო ელექტრონული გამომთვლელი მანქანის (მიკრო-კონტროლერის) მეხსიერებას შეუძლია შეინახოს ჩვენ მიერ ზემოთ მოტანილი ორობითი 11011001_2 რიცხვი. ნულებისა და ერთების ასეთი გრძელი მიმდევრობა ძნელია დასამახსოვრებლად და მოუხერხებელია კლავიატურიდან შესატანად. 11011001_2 რიცხვი შეგვეძლო გარდაგვექმნა ათობითად, რაც მოგვცემდა 185_{10}-ს, მაგრამ გადაყვანის პროცესი დიდ დროს დაგვაკარგვინებდა (იხ. ნახ. *3.2.3*).

მიკროინფორმატიკის სისტემების უმრავლესობა 11011001_2-ის მსგავსი რიცხვების დამახსოვრებისა და გამოყენების გამარტივებისათვის, იყენებს რიცხვების ჩაწერის *16*-ობით ფორმას.

2.5,ა ნახაზზე მოყვანილია ეკვივალენტური ათობითი, ორობითი და თექვსმეტობითი რიცხვები. შევნიშნავთ, რომ თითოეული თექვსმეტობითი სიმბოლო (ციფრი) შეიძლება წარმოვადგინოთ ოთხი ბიტის ერთადერთი ერთობლიობით (ე.წ. *ტეტრადათი*). მაგალითად, ორ-ობით **10011011** რიცხვში ბიტების პირველ **1001** ოთხეულს, ანუ ტეტრადას შეესაბამება თექვსმეტობითი 9_{16} ციფრი, ხოლო მომდევნო **1011** ტეტრადას – $\mathbf{B}_{16}$ ციფრი; **1001** და **1011** ტეტრადების შესაბამისად 9_{16} და - $\mathbf{B}_{16}$ ციფრებით შეცვლისას განხილული ორობითი რიცხვი გარდაიქმნება თექვსმეტობით რიცხვად, ე.ი. $10011011_2 = 9\mathbf{B}_{16}$.

ზემოთ აღნიშნულიდან გამოდის შემდეგი: ***ორობითი რიცხვის თექვსმეტობით რიცხვებად გარდასაქმნელად*** ორობითი რიცხვი, დაწყებული უმცროსი ბიტიდან, დავყოთ ოთხი ბიტის შემცველ ჯგუფებად. არასრული ჯგუფის მიღებისას ბიტების წინ *0*-ების მიწერით მასში ბიტების რაოდენობა შევავსოთ ოთხამდე. მიღებული ოთხეულები (ტრიადები) შევცვალოთ *2.5,ა* ნახაზზე მოყვანილი *16*-ბითი ციფრებით. მაგალითისათვის განვიხილოთ ორობითი რიცხვი 1110101_2. მარცხნიდან მარჯვნივ ოთხეულებად დაყოფისას პირველად მიიღება *1010*, ხოლო შემდეგ *11*; ამ უკანასკნელის ოთხამდე შევსებისათვის

ა)

10-ობითი	16-ობითი	ორობითი				10-ობითი	16-ობითი	ორობითი			
		8	4	2	1			8	4	2	1
0	0	0	0	0	0	8	8	1	0	0	0
1	1	0	0	0	1	9	9	1	0	0	1
2	2	0	0	1	0	10	A	1	0	1	0
3	3	0	0	1	1	11	B	1	0	1	1
4	4	0	1	0	1	12	C	1	1	0	0
5	5	0	1	0	1	13	D	1	1	0	1
6	6	0	1	1	0	14	E	1	1	1	0
7	7	0	1	1	1	15	F	1	1	1	1

ბ)

ფუძის ხარისხი	16^3		16^2		16^1		16^0	
პოზიციის მნიშვნელობა	4096		256		16		1	
16-ობითი	2		D		4		F	
ათობითი	$2 \times 4096 = 8192$	+	$13 \times 256 = 3328$	+	$4 \times 16 = 64$	+	$15 \times 1 = 15$	= 11599

გ)

გასაყოფი	გამყოფი	განაყოფი	
		მთელი	ნაშთი
15797_{10}	16	987_{10}	$5_{10} = 5_{16}$
987_{10}	16	61_{10}	$11_{10} = B_{16}$
61_{10}	16	3_{10}	$13_{10} = D_{16}$
3_{10}	16	0_{10}	$3_{10} = 3_{16}$

$15797_{10} = 3\ D\ B\ 5_{16}$

ნახ.2.5. *10-ობითი, 16-ობითი და 2-ობითი ეკვივალენტები (ა); 16-ობითი რიცხვის 2-ობითად (ბ) და 10-ობითს - 16-ობითი გარდაქმნა (გ)*

წინ უნდა მივუწეროთ ***00***, ე.ი. ***11 = 0011***. მაშასადამე 1111010_2 რიცხვი გადაიქცევა 001111010_2 რიცხვად: $111010_2 = 0011\ 1010_2$; ***2.5,ა*** ნახაზის თანახმად $0011_2=3_{16}$, ხოლო $1010_2=A_{16}$, ამიტომ $111010_2 = 3A_{16}$.

თექვსმეტობითი ჩაწერა ფართოდ გამოიყენება ორობითი რიცხვების წარმოსადგენად, ამიტომ ***2.5,ა*** ნახაზზე მოცემული ცხრილი უნდა ზეპირად უნდა გვახსოვდეს.

თექვსმეტობითი რიცხვის ათობით რიცხვებად გარდაქმნის პროცედურა ილუსტრირებულია ***2.5,ბ*** ნახაზზე, რომელზედაც თექვსმეტობით რიცხვად აღებულია $2D4F_{16}$. პირველი ოთხი თექვსმეტთანრიგიანი ციფრის პოზიციათა მნიშვნელობები მარცხნიდან მარჯვნივ შესაბამისად არის ***4096***, ***256***, ***16*** და ***1***. ნახაზის თანახმად $2D4F_{16}$ რიცხვის ეკვივალენტური ათობითი რიცხვი უნდა შეიცავდეს ***15*** (E_{16})– ერთს, ოთხ თექვსმეტს, ცამეტ (D_{16}) ორასორმოცდათექვსმეტსა და ორ ოთხიათასოთხმოცდათექვსმეტს.

$2D4F_{16}$ რიცხვის თითოეული ციფრი მრავლდება მის მიერ დაკავებული პოზიციის მნიშვნელობაზე, ანუ წონაზე და მიღებული შედეგები იკრიბება, რის შედეგადაც ვიღებთ 11599_{10}-ს ე.ი. $2D4F_{16} = 11599_{10}$

ათობითი რიცხვის თექვსმეტობით რიცხვებად გარდაქმნის პროცედურა ილუსტრირებულია ***2.5,გ*** ნახაზზე, რომელზეც ათობით რიცხვად აღებულია 15797_{10}. ***პირველ მწკრივში*** 15797_{10} იყოფა ***16***-ზე, რომლის შედეგადაც მიღებულია 987_{10} მთელი და 5_{10} ანუ 5_{16} ნაშთი. პირველი მთელი ნაწილი (***987***) ***მეორე მწკრივში*** ხდება გასაყოფი და ხელახლა იყოფა ***16***-ზე. მიიღება ***61*** მთელი და 11_{10} ანუ B_{16} ნაშთი; ***მესამე მწკრივში*** გასაყოფი ***61*** იყოფა ***16***-ზე; მიიღება ***3*** მთელი და 13_{10} ანუ D_{16} ნაშთი. მეოთხე მწკრივში გასაყოფი ***3*** იყოფა ***16***-ზე; მიიღება ***0*** მთელი და 3_{10} ანუ 3_{16} ნაშთი. რადგან მთელი ნაწილი ***0***-ია, პროცედურა მთავრდება. ნაშთების ქვემოდან ზემოთ მიმდევრობით ამოწერისას მივიღებთ საძებნ $3DB5_{16}$ შედეგს, ე. ი. $15797_{10}=3DB5_{16}$

2.4. რვაობითი რიცხვები

რვაობითი ჩაწერა თექვსმეტობითი ჩაწერის მსგავსად ორობითი რიცხვების წარმოდგენისათვის გამოიყენება. ორობითი სისტემა წარმოადგენს ***0***-დან დაწყებული ***7***-ის ჩათვლით დამთავრებული ***8*** ციფრის შემცველ სისტემას, რომლის ფუძეა ***8***. ***შ.2.6,ა*** ნახაზზე მოცემულია რამდენიმე ათობითი, რვაობითი და ათობითი რიცხვი.

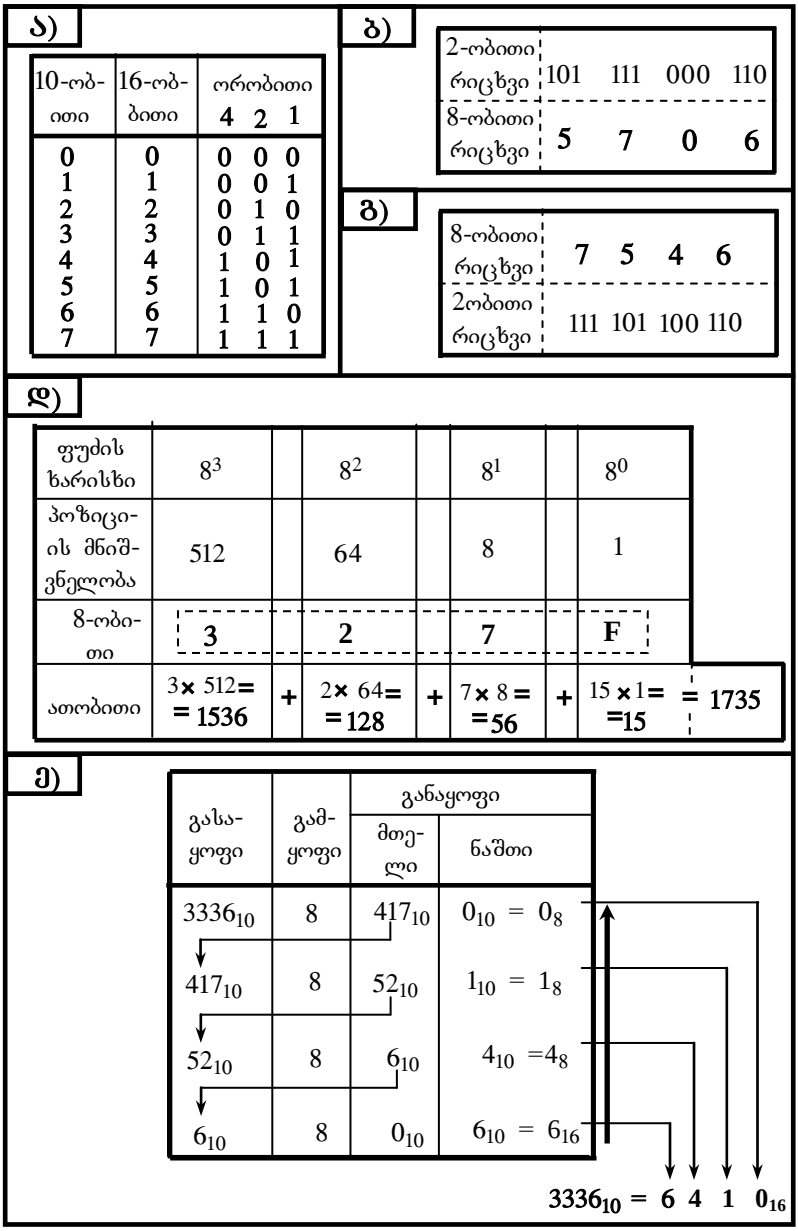
2.6,ა ნახაზზე ნაჩვენებია რვაობითი ციფრების შესაბამისი ორობითი რიცხვები.

ორობითი 1011110000110_2 რიცხვი გარდავქმნათ ეკვივალენტურ რვაობით რიცხვად. ამისათვის უმცროსი ბიტიდან დაწყებული იგი დავყოთ სამეულებად ანუ ***ტრიადებად***. ბოლო ოპერაციისას მიღებულ ჯგუფში არასაკმარისი რაოდენობის ციფრების არსებობისას იგი სამამდე უნდა გავზარდოთ ბიტებისათვის მარცხნივ საჭირო რაოდენობის ***0***-ების მიწერით. ამის შემდეგ ***შ.2.6,ა*** ნახაზზე მოცემული ცხრილის გამოყენებით თითოეული ტრიადა შევცვალოთ ეკვივალენტური რვაობითი ციფრით, რის შედეგადაც მივიღებთ საძებნ რვაობით 5706_8 რიცხვს, ე.ი. $1011110000110_2=5706_8$ (ნახ.***2.6,ბ***)

რვაობითი 7546_8 გარდავქმნათ მის ეკვივალენტურ ორობით რიცხვად. თითოეული რვაობითი ციფრი შევცვალოთ ორობითი ტრიადით და მივიღებთ, რომ $7546_8 = 1111011001100_2$ (ნახ. ***2.6,გ***).

2.6,დ ნახაზზე მოყვანილია რვაობითი $327F_8$ რიცხვის ათობითი ფორმით ჩაწერის კლასიკური პროცედურა. აქ პირველი ოთხი ***8***-ობითი პოზიციის მნიშვნელობები (წონები) მარცხნიდან მარჯვნივ შესაბამისად არის ***512, 64, 8*** და ***1***. ამ მაგალითში გვაქვს ***15*** ერთეული. ***7*** რვაეული, ***2*** სამოცდაოთხეული და ***3*** ხუთასთორმეტეული. მათი შეკრებით ვიღებთ შედეგს: ***1536 + 128 + 56 + + 15 = 1735.***

დასასრულს, ათობითი 3336_{10} რიცხვის რვაობით ეკვივალენტად გარდაქმნის პროცედურა ***2.6,ე*** ნახაზზეა ნაჩვენები. ***პირველ მწკრივში 3336*** გაყოფილია ***8***-ზე; მიღებულია მთელი ნაწილი ***417*** და $0_{10}=0_8$ ნაშთი. მიღებული მთელი ***417*** ნაწილი ***მეორე მწკრივში*** განხილულია გასაყოფად და ხელახლადაა გაყოფილი ***8***-ზე. მიღებულია მთელი ნაწილი ***52*** და $1_{10}=1_8$ ნაშთი. ***მესამე მწკრივში 52*** გაყოფილია ***9***-ზე. მიღებულია მთელი ნაწილი ***6*** და $4_{10}=4_8$ ნაშთი. ***მე-***



ნახ.2.6. *10-ობითი, 8-ობითი და 2-ობითი ეკვივალენტები (**ა**), 2-ობითი რიცხვის 8-ობითად (**ბ**), 8-ობითის – 2-ობითად (**გ**) და 10-ობითის – 8-ობითად (**დ**) გარდაქმნა. 10-ობითი მთელი რიცხვის 8-ობით რიცხვად გარდაქმნა (**ე**).*

ოთხე მწკრივში მთელი ნაწილი ***6*** გაყოფილია ***8***-ზე. მიღებულია მთელი ნაწილი ***0*** და $4_{10}=0_8$ ნაშთი.

რადგან მთელი ნაწილი ***0***-ის ტოლი გახდა, პროცედურა მთავრდება. მიღებული ნაშთების ქვემოდან ზემოთ მიმდევრობით ამოწერით ფორმირებულია 6410_8 შედეგი, ე.ი. $3336_{10} = 6410_8$.

მიკროპროცესორებისა და მიკროელექტრული გამომთვლელი მანანების უმრავლესობა ამუშავებს ***4***-ის ჯერადი რაოდენობის ბიტებისაგან შედგენილ ჯგუფებს. ამიტომ თექვსმეტობითი ჩაწერა უფრო ხშირად გამოიყენება, ვიდრე რვაობითი ჩაწერა; სამაგიეროდ რვაობითი ჩაწერა უფრო მოსახერხებელია მაშინ, როდესაც ბიტების რაოდენობა ***3***-ის ჯერადია (მაგალითად, როცა გამოიყენება 12-ბიტიანი ჯგუფები).

2.5. ორობით-ათობითი რიცხვები

ბიტების (ორობითი ციფრების) საშუალებით ათობითი (რაციონალური) რიცხვების წარმოდგენისათვის გავრცელებულია დიტების (ათობითი ციფრების) კოდირება ოთხნიშნა ორობითი კოდების კოდური სიტყვებით. აღნიშნულ ოპერაციას ათობითი რიცხვების ორობითი კოდირება ანუ –***BCD***-კოდირება (ინგლ. ***B****inar****y****-****C****oded* ***D****ecimal* – ორობით-ათობითი კოდირება) ეწოდება, რომლის შედეგადაც მიიღება ე.წ. ***ორობით-ათობითი რიცხვები***.

BCD-კოდირებისათვის შეიძლება გამოვიყენოთ სხვადასხვა კოდის კოდური სიტყვები. ასეთი კოდებიდან ყველაზე ხშირად გამოიყენება ***8421-*** და ***+3***-კოდები, რომელთა კოდური სიტყვებით ათობითი ***0,...,9*** ციფრების კოდირების მიღებული ვარიანტი ***2.7,ა*** ნახაზზე მოყვანილ ცხრილშია ნაჩვენები. ამ ცხრილის თანახმად, მაგალითად, ათობითი ციფრი ***3*** კოდირებულია ***8421***-კოდის კოდური ***0011*** სიტყვით და ***+3***-კოდის კოდური ***0110*** სიტყვით.

ათობითი 3875_{10} რიცხვის ***8421***-კოდისა და ***+3***-კოდის გამოყენებით ჩაწერისას მისი თითოეული ციფრი კოდირდება ზემოთ აღნიშნული კოდების კოდური სიტყვებით და მიიღება (ნახ.***2.7,ბ***):

$3875_{10} = 0011100001111_{8421} = 0110101110101000_{+3}$.

8421-კოდის საშუალებით წარმოდგენილი ორობით-ათობითი ***0101001001110011*** რიცხვისა და ***+3***-კოდის საშუალებით წარმო-

ა)

ათობითი ციფრები	*BCD* კოდირებისათვის გამოსაყენებელი კოდები							
	8	4	2	1	+3 კოდი			
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

ბ)

ათობითი რიცხვი	3	8	7	5
8421-BCD კოდი	0011	1000	0111	0101
+3 - BCD კოდი	0110	1011	1010	1000

გ)

8421-BCD კოდი	0101	0010	0101	0011
+3 - BCD კოდი	1000	0101	1001	0110
ათობითი რიცხვი	5	2	6	3

ნახ.2.7. *დიტების კოდირება 8421- და +3-კოდის კოდური სიტყვებით (**ა**); ათობითი რიცხვის ორობით ათობით (**ბ**) და ორობით-ათობითი რიცხვების ათობით რიცხვებად (**გ**) გარდაქმნა.*

დგენილი ორობით-ათობითი ***1000010110010110*** რიცხვის ათობით რიცხვად გარდაქმნისათვის ამ რიცხვების თითოეული ტრიადა (ოთხეული) იცვლება შესაბამისი ათობითი ციფრით (დიტით) და მიიღება (ნახ.***2.7,გ***);

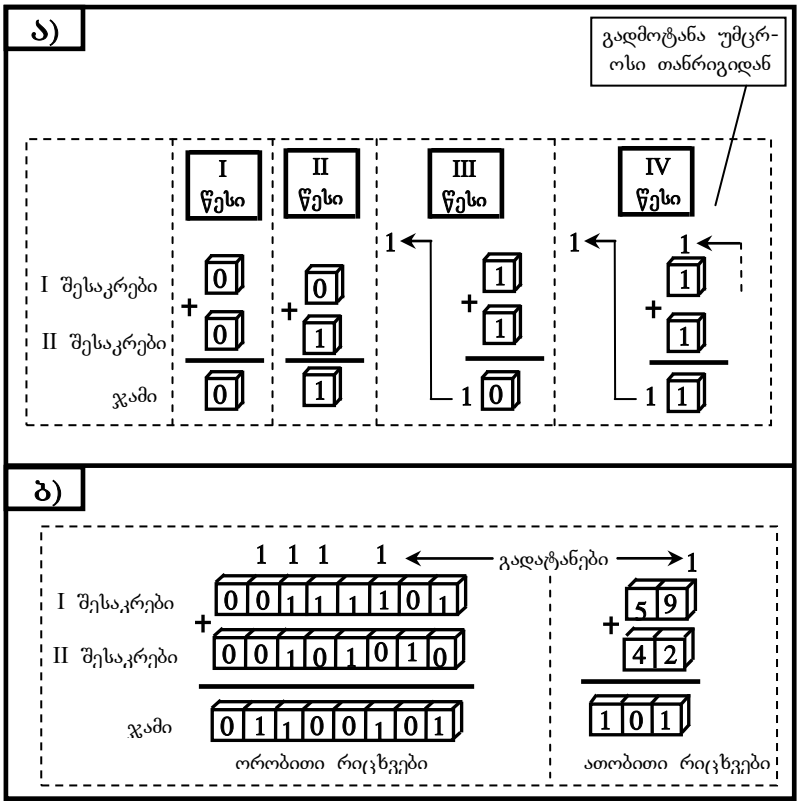
$$0101001001110011_{8421} = 1000010110010110_{+3} = 5263_{10}$$

მიკროპროცესორებს შეუძლია წმინდა ორობითი რიცხვების შეკრება, თუმცა მათ აქვთ მიღებულ ჯამის ათობით-ორობით რიცხვებად

გარდაქმნის ბრძანებები. ეს უკანასკნელები საჭიროებისას ზემოთ აღწერილი პროცედურების გამოყენებით შეიძლება ადვილად გარდაიქმნას ათობით რიცხვებად.

2.6. ორობითი არითმეტიკა

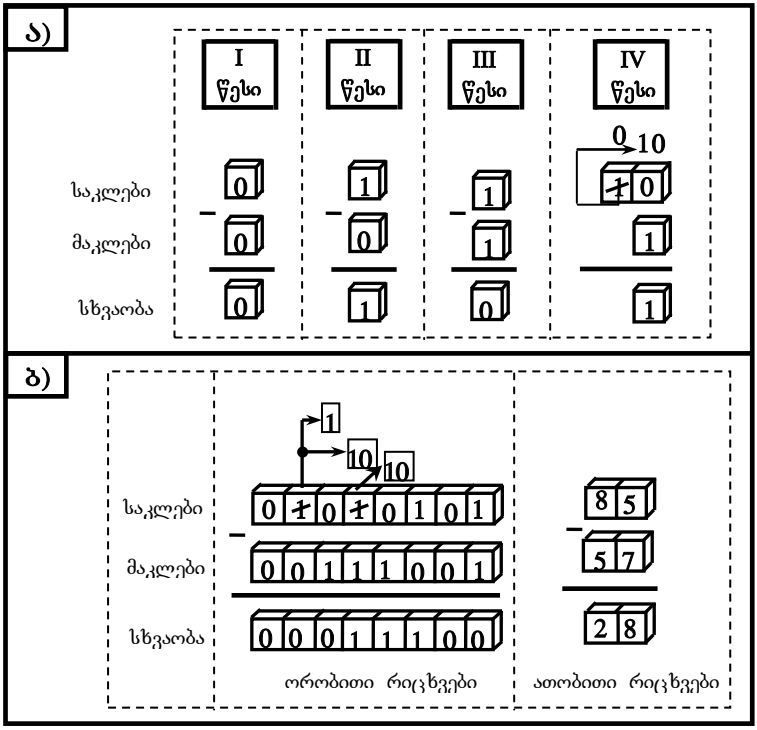
ორობითი რიცხვების შეკრების, გამოკლებისა და გამრავლების ოპერაციები ისევე სრულდება, როგორც ანალოგიური ოპერაციები ათობით რიცხვებზე.



ნახ.2.8. *ორობითი შეკრების წესები (ა) და მაგალითი (ბ)*

მიკროპროცესორების უმრავლესობას აქვს ორობითი რიცხვების შეკრებისა (***ADD***) და გამოკლების (**SUB**) ბრძანებები, მაგრამ არსე-

ბობს ისეთი პროცესორებიც (მაგალითად, ***I8086*** და ***8088***), რომლებსაც დამატებით აქვთ გამრავლების (***MUL***) და გაყოფის (***DIV***), ბრძანებებიც.



ნახ.2.9. *ორობითი გამოკლების წესები (**ა**) და მაგალითი (**ბ**)*

2.8,ა ნახაზზე მოყვანილია ორობითი შეკრების წესები. ***I*** და ***II წესების*** აზრი ნათელია. ***III წესის*** თანახმად ***1+1=10***, მიღებულ ჯამში არსებული მაღალი ნიშნადობის მქონე ციფრი ***1*** გადაიტანება უახლოეს უფროს თანრიგში. ***IV წესი*** გვიჩვენებს, რომ ***1+1+1=11***. ამ შემთხვევაში იკრიბება არა მარტო ***I, II*** შესაკრებებად არსებული ციფრები ***1***, არამედ უმცროსი თანრიგიდან გადმოტანილი მესამე ციფრი ***1***-იც. მათი შეკრებით მიიღება ჯამი ***1*** და გადატანა ***1***.

2.8,ბ ნახაზზე მოყვანილია ორობითი ***00111101*** და ***00101010*** რიცხვების შეკრების მაგალითი. თვალსაჩინოებისათვის ამ რიცხვების ათობითი ეკვივალენტები ნახაზის მარჯვენა მხარესაა შეკრებილი.

ორობითი გამოკლების წესები ***2.9,ა*** ნახაზზეა მოყვანილი. პირველი სამი წესი ათობითი შეკრების წესების ანალოგურია. ***IV*** წესი უფროსი თანრიგიდან (რომლის წონაა ***2***) მოითხოვს სესხს. ეს სესხი, რომელიც საკლების როლს ასრულება, არის ***10***. მისგან ***1***-ის გამოკლებით მიიღება სხვაობა ***1.***

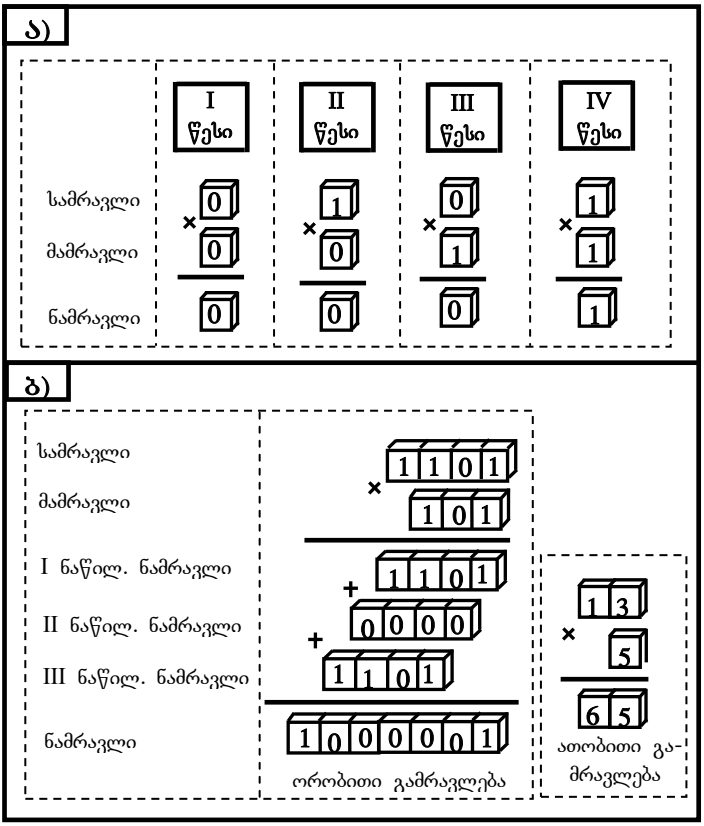
01010101_2 რიცხვს გამოვაკლოთ ***00110001_2*** რიცხვი (ნახ.***2,9,ბ***). სვეტებში, რომელთა თანრიგების წონებია ***1,2*** და ***4,*** გამოკლების ოპერაციები ადვილად შესასრულებადია. სვეტში, რომლის წონაა ***8,*** უნდა შესრულდეს ოპერაცია ***0-1.*** მის შესასრულებლად სვეტიდან, რომლის წონაა ***16*** ნასესხებია ***1***, რომელიც სვეტ ***8***-ში გადმოტანისას იღებს სახეს ***10_2***. რის გამოც ამ სვეტში ***IV*** წესის შესაბამისად სრულდება ოპერაცია ***10-1=1***. (იხ.ნახ.***2.9,ბ***).

რთულადაა საქმე სვეტში, რომლის წონაა ***16.*** ამ სვეტიდან სესხად ***1***-ის წაღების შემდეგ, რაც ზემოთ განვახორციელეთ, საკლებად დარჩა ***0*** და ამ სვეტშიც უნდა შესრულდეს ოპერაცია ***0-1***. ზემოთ განხილული შემთხვევის ანალოგურად საჭიროა ***1*** ვისესხოთ სვეტიდან, რომლის წონაა ***32***, მაგრამ ამ სვეტშიც საკლებად დგას ***0***. ამიტომ სვეტმა ***32***-მა უნდა ისესხოს ***1*** სვეტ ***64***-დან და შემდეგ ეს ნასესხები გადასცეს სესხად სვეტ ***16***-ს. ამ ოპერაციების შესრულების შემდეგ სვეტ ***16***-ში საკლებად აღმოჩნდება რიცხვი ***10_2*** და შესრულდება ოპერაცია ***10-1=1***, ხოლო სვეტ ***32***-ში საკლებად აღმოჩნდება ციფრი ***1*** და იქ შესრულდება ოპერაცია ***1-1=0***.

სვეტ ***64*** ზემოთ აღნიშნული სესხების გაცემის შემდეგ საკლებად დარჩება ***0*** და მასში შესრულდება ოპერაცია ***0-0=0***. იგივე ოპერაცია შესრულდება სვეტ ***64***-ში, და საბოლოოდ მივიღებთ, რომ (იხ. ნახ. ***2.9,ბ***):

01010101-00111001=00011100.

ორობითი გამრავლების წესები (ნახ. ***2.10,ა***) ემთხვევა ათობითი გამრავლების წესებს. ამ წესების თანახმად, მამრავლი თუ ***0***-ის ტოლია, ნამრავლში ვიღებთ ***0***-ს, ხოლო თუ მამრავლი ***1***-ის ტოლია, ნამრავლი ემთხვება სამრავლს.



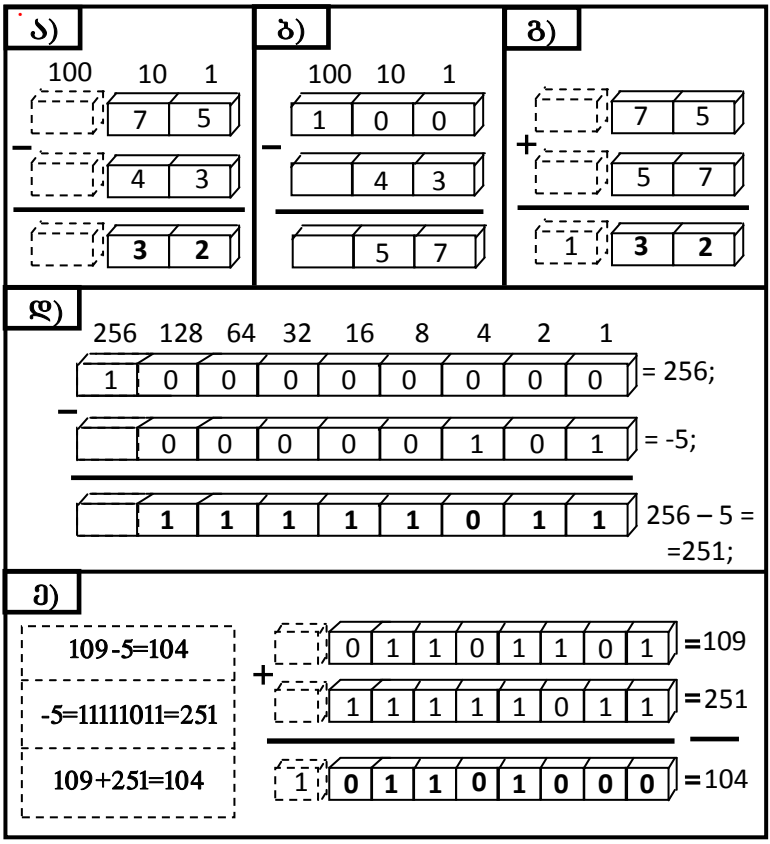
ნახ.2.10. *ორობითი გამრავლების წესები (ა) და მაგალითი (ბ)*

ორობითი 1101_2 რიცხვის 101_2 რიცხვზე გამრავლების მაგალითი ***2.10,ბ*** ნახაზზეა მოყვანილი. ნახაზის მარჯვენა მხარეზე ანალოგური ოპერაცია შესრულებულია ათობით ეკვივალენტებზეც.

2.7. დამატებითი კოდების რაობა

სხვადასხვა ნიშნის მქონე რიცხვების შესაკრებად პროცესორმა დიდი აბსოლუტური მნიშვნელობის მქონე რიცხვს უნდა გამოაკლოს პატარა აბსოლუტური მნიშვნელობის მქონე რიცხვი და ჯამს მიანიჭოს დიდი აბსოლუტური მნიშვნელობის მქონე რიცხვის ნიშანი. დადე-

ბითი რიცხვიდან უარყოფითი რიცხვის გამოკლების ოპერაცია შეიძლება შეიცვალოს მისთვის უარყოფითი რიცხვის ***დამატებითი კოდის*** (რომელიც დადებით რიცხვს წარმოადგენს) მიმატების ოპერაციით. განვიხილოთ, თუ რა განაპირობებს ამას. თვალსაჩინოებისათვის გამოვიყენოთ ჩვენთვის ცნობილი ათობითი რიცხვები. ამასთანავე სიმარტივისათვის ავიღოთ მათი ორნიშნა წარმომადგენლები. მიღებული შედეგი შეიძლება ადვილად განზოგადდეს ნებისმიერი სიგრძის ათობით რიცხვებზე.



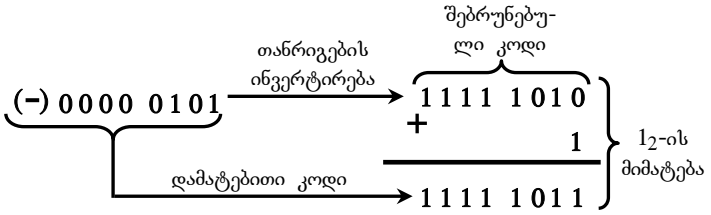
ნახ.2.11.*. სხვადასხვანიშნიანი ათობითი რიცხვების შეკრება (ა), ათობითი დამატებითი კოდის პოვნა (ბ) და მისი გამოყენება (გ), ორობითი დამატებითი კოდის პოვნა (დ) და მისი გამოყენება (ე)*

ორნიშნა ათობითი რიცხვების ჩასაწერად გამოსაყენებელ ორთანრიგიანი რეგისტრის პირობით გამოსახულებას დავუმატოთ პუნქტირით გამოსახული ***გადავსების აბსტრაქტული თანრიგი*** (ნახ.***2.11,ა***). რეგისტის როგორც რეალური, ასევე აბსტრაქტული თანრიგების წონები მათ ზემოთ მივუთითოთ. ასეთი რეგისტრების გამოყენებით შესრულებული ***75 – 43 =32*** ოპერაცია ***2.11,ა*** ნახაზზეა მოყვანილი.

ორნიშნა 43 რიცხვის ***დამატებითი კოდი*** ეწოდება ისეთ რიცხვს, რომელიც უნდა დავუმატოთ მას უმცირესი ***სამნიშნა*** რიცხვის ანუ ***100***-ის მისაღებად, რომელიც გადავსების აბსტრაქტული თანრიგის წონასაც წარმოადგენს; ე. ი ***43***–ს დამატებითი კოდი იქნება ***100 - - 43=75***. აბსოლუტური სიდიდით იგი საწყის ***-43*** რიცხვს, რომელიც ჩვენს მაგალითში ასრულებს მაკლების როლს, ***100***-ერთეულით.

საკლებ ***75***-იდან ***43***-ის გამოკლების ოპერაციას თუ შევცვალოთ მისთვის ***43***-ის დამატებითი კოდის ანუ ***57***-ის მიმატების ოპერციით, რომელიც ***43***-ს აღემატება ***100***-ით (იხ.ნახ.***4.11,გ***), მიიღება ასევე ***100***-ით მეტი შედეგი ***132***; ამ უკანასკნელში არსებული ზედმეტი ასეულის მაჩვენებელი ციფრი ***1*** გადავა გადავსების აბსტრაქტულ თანრიგში და დაიკარგება, რეალურ თანრიგებში კი დაგვრჩება ჭეშმარიტი შედეგი ***32***. მაშასადამე, საკლებიდან მაკლების გამოკლების ოპერაცია შეიცვალა საკლებზე მაკლების დამატებითი კოდის მიმატებით. აქედან გამომდინარე, შეიძლება დავასკვნათ: ***გამოკლების ოპერაცია შეიძლება შევცვალოთ საკლებისათვის მაკლების დამატებითი კოდის მიმატების ოპერაციით***; უფრო ზოგადად ***სხვადასხვანიშნიანი რიცხვების შეკრების ოპერაცია შეიძლება შევცვალოთ ორი დადებითი რიცხვის შეკრების ოპერაციით, რომელთაგანაც მეორე დადებითი რიცხვის როლს ასრულებს უარყოფითი რიცხვის დამატებითი კოდი.***

ათობითი რიცხვებისათვის დამატებითი კოდების პოვნის ამოცანის ფორმალიზება რამდენადმე რთულია. საწინააღმდეგო სურათი გვაქვს ორობითი რიცხვების შემთხვევაში. ორობითი რიცხვისათვის დამატებითი კოდის პოვნისათვის ჩასატარებელი ოპერაციების სქემა ***2.12*** ნახაზზეა მოყვანილი, რომლის შესაბამისად შეიძლება ჩამოვაყალიბოთ ორობითი რიცხვებისათვის დამატებითი კოდების პოვნის შემდეგი ალგორითმი:



***ნახ.2.2.12** . ორობითი რიცხვის შებრუნებული და დამატებითი კოდების მიღება*

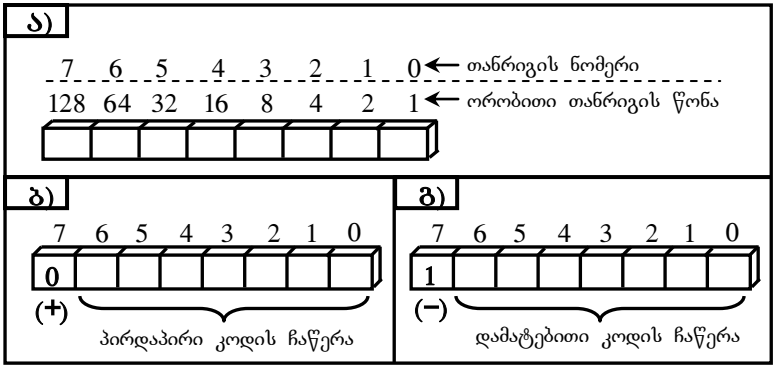
1. მოხდეს მოცემული ორობითი რიცხვის (ე.ი. მისი თითოეული ბიტის) ***ინვერსირება***, რის შედეგადაც მიიღება ამ რიცხვის ***შებრუნებული კოდი***;

2. პუნქტ ***1***-ში მიღებულ შებრუნებულ კოდს დაემატოს ***1,*** ე. ი. მოხდეს მისი ***ინკრემენტირება.***

3. პუნქტ ***2***-ში მიღებული რიცხვი წარმოადგენს საწყისი ორობითი რიცხვის დამატებით კოდს;

4. ალგორითმის დასასრული.

როგორც ალგორითმიდან ჩანს, ორობითი რიცხვის დამატებითი კოდის საპოვნელად პროცესორმა უნდა მოახდინოს ამ რიცხვის ინვერსირება და ინკრემენტირება. რადგან ნებისმიერი პროცესორისათვის ინვესირებისა და ინკრემენტირების ოპერაციები ადვილად შესასრულებელი ოპერაციებია, ამიტომ მათთვის დამატებითი კოდების ფორ-



***ნახ.შ.2.2.13.** ორობითი თანრიგების განლაგება (**ა**); ნიშნის ბიტში **0**-ის არსებობით დადებითი რიცხვების იდენტიფიცირება (**ბ**); ნიშნის ბიტში **1**-ის არსებობით უარყოფითი რიცხვების იდენტიფიცირება (**გ**);*

მირების ამოცანა მარტივ ამოცანათა კლასს მიეკუთვნება. პროცესორი სხვადასხვანიშნიანი რიცხვებზე ოპერაციების ჩატარების საჭიროებისას იგი იყენებს ჩვენ მიერ ზემოთ განხილულ დამატებით კოდებს. ეს ამარტივებს აღნიშნული ოპერაციების ჩატარებისათვის საჭირო აპარატურულ საშუალებებს.

მიკროპროცესორში ან მიკროპროცესორის მეხსიერებაში უნიშნო *8*-თანრიგიანი რიცხვების შესანახად გამოყენებული *8*-თანრიგიანი რეგისტრის პირობითი გამოსახულება *2.13,ა* ნახაზზეა მოყვანილი. რეგისტრის თანრიგები ნახაზზე დანომრილია და თითოეულ თანრიგთან მითითებულია მისი წონა. კერძოდ, მე-*7* თანრიგის წონაა *128,* მე-*6* თანრიგის – *64* და ა.შ.

ნიშნიანი *8*-თანრიგიანი რიცხვების შესანახად გამოყენებული *8*-თანრიგიანი რეგისტრების სტრუქტურები *2.13,ბ,გ* ნახაზებზეა მოყვანილი. დადებითი რიცხვების ჩასაწერად გამოიყენება *2.13,ა* ნახაზზე, ხოლო უარყოფითი რიცხვების ჩასაწერად – *2.13,ბ* ნახაზზე ნაჩვენები რეგისტრი. ორივე შემთხვევაში რეგისტრის მე-*7* თანრიგი გამოიყენება რიცხვის ნიშნის მაჩვენებელი ბიტის ჩასაწერად. როგორც ნახაზიდან ჩანს, პლუს ნიშანს შეესაბამება ბიტი *0*, ხოლო მინუს ნიშანს – ბიტი *1*. აღნიშნულის გამო *8*-თანრიგიან რეგისტრში შეიძლება ჩაიწეროს *7*-თანრიგიანი ორობითი რიცხვი.

ზემოთ აღნიშნულის თანახმად თუ რეგისტრში არსებული ნიშნის ბიტი *0*-ის ტოლია, მაშინ მასში ჩაწერილია დადებითი რიცხვი, ხოლო თუ *1*-ის ტოლია, მაშინ – უარყოფითი რიცხვი.

2.13,ბ ნახაზის თანახმად *8*-თანრიგიან რეგისტრში დადებითი *7*-თანრიგიანი რიცხვების ჩასაწერად თავისუფალ თანრიგებში ჩაიწერება ***პირდაპირ კოდში*** წარმოდგენილი ეს რიცხვები, ანუ უშუალოდ დადებითი *7*-თანრიგიანი ორობითი რიცხვები. რეგისტრში *01001001* ბიტების არსებობა ნიშნავს, რომ მასში ჩაწერილია $+73_{10}$ (*64*+*8*+*1*), ხოლო მასში *01111111* ბიტების არსებობა – რომ მასში ჩაწერილია 127_{10} (*64*+*32*+16+8+4+*2*+1). ეს უკანასკნელი არის ის უდიდესი დადებითი რიცხვი, რომელსაც შეიძლება შეიცავდეს განხილული *8*-თანრიგიანი რეგისტრი.

2.13,ვ ნახაზის თანახმად *8*-თანრიგიან რეგისტრში უარყოფითი *7*-თანრიგიანი რიცხვების ჩასაწერად თავისუფალ თანრიგებში ჩაიწერება ამ რიცხვების შესაბამისი დამატებითი კოდები.

2.1 ცხრილში დადებითი და უარყოფითი რიცხვების დამატებითი კოდები. როგორც ვხედავთ, ნებისმიერი დადებითი რიცხვის უფროს თანრიგში დგას ციფრი *0*, ხოლო დანარჩენ თანრიგებში მდგარი ბიტები წარმოქმნის ორობით რიცხვს. ნებისმიერი უარყოფითი რიცხვის უფროს თანრიგში დგას *1*. განვიხილოთ *2.1* ცხრილის *+0* სტრიქონი: დამატებით კოდში *+0* ჩაიწერება როგორც *00000000*.

ცხრ.2.1. ნიშნიანი ათობითი რიცხვები და მათი წარმოდგენა დამატებითი კოდით

ათობითი როცხვები	ნიშნებიანი რიცხვების წარმოდგენა და შენიშვნები	
+127	0111 1111	დადებითი რიცხვები პირდაპირი ორობითი რიცხვებივით წარმოიდგინება
⋮	⋮	
+9	0000 1001	
+8	0000 1000	
+7	0000 0111	
+6	0000 0110	
+5	0000 0101	
+4	0000 0100	
+3	0000 0011	
+2	0000 0010	
+1	0000 0001	
+0	0000 0000	
-1	1111 1111	უარყოფითი რიცხვები დამატებითი კოდის სახით წარმოიდგინება
-2	1111 1110	
-3	1111 1101	
-4	1111 1100	
-5	1111 1011	
-6	1111 1010	
-7	1111 1001	
-8	1111 1000	
-9	1111 0111	
⋮	⋮	
-128	1000 0000	

უახლოეს ქვედა სტრიქონში ვხედავთ, რომ *-1* დამატებით კოდში ჩაიწერება როგორც *11111111*. ცხრილიდან ჩანს, რომ რეგისტრში ჩა-

ცხრ. 2.2. -10_{10} რიცხვის დამატებითი კოდის გამოთვლა

ოპერანდი	ჩასატარებელი ოპერაცია	ოპერაციის რეალიზება
-10_{10}	ოპერანდი ნიშნის გარეშე ჩაიწეროს 8-ნიშნა ორობითი რიცხვის სახით	00001010
00001010	ოპერანდი წარმოდგენილი იქნეს შებრუნებული კოდის სახით	11110101
11110101	მოხდეს ოპერანდის ინკრემენტირება (მას დაემატოს 1)	11110101 + 1 = 11110110
11110110	მიღებული ოპერანდი გავუტოლოთ საწყისი ოპერანდის დამატებით კოდს	11110110 არის -10_{10} რიცხვის დამატებითი კოდი

ცხრ.შ.2.3. 11110000 რიცხვის ათობითი ეკვივალენტის პოვნა

ოპერანდი	ჩასატარებელი ოპერაცია	ოპერაციის რეალიზება
00001111	ოპერანდი ჩაიწეროს შებრუნებული კოდის სახით	11110000
11110000	მოხდეს ოპერანდის ინკრემენტირება (მას დაემატოს 1)	00001111 + 1 = 00010000
00010000	დავწეროთ ოპერანდის ათობითი ეკვივალენტი	16_{10}
-16_{10}	ოპერანდი ავიღოთ უარყოფითი ნიშნით	-16_{10}

ჩაწერილი უდიდესი უდიდესი რიცხვი *-128*-ის ტოლია და მას შეესატყვისება დამატებითი კოდი *10000000.*

2.1 ცხრილში მოყვანილია დამატებითი კოდები *-1*-დან *-9* რიცხვამდე. *2.2* ცხრილში მოყვანილია -10_{10} რიცხვის დამატებითი კოდის პოვნის პროცედურა. აღნიშნული ცხრილის თანახმად -10_{10} რიცხვის დამატებითი კოდია *11110110*. შევნიშნავთ რომ ნიშნის მე-*7* თანრიგში არსებული ბიტი *1* იმას ნიშნავს, რომ მიღებული (*11110110*) რიცხვი უარყოფითია.

განვსაზღვროთ ორობითი *11110000* რიცხვის შესაბამისი ათობითი ეკვივალენტი. *2.3* ცხრილში მოყვანილია ზემოთ დამოწმებული რიცხვის ათობით რიცხვად გარდაქმნის პროცედურა. პროცედურის პირველ ეტაპზე მიღებულია დადებითი ორობითი $00010000_2 = 16_{10}$ რიცხვი, მაგრამ ვინაიდან დამატებითი კოდის მთავარი ბიტი *1*-ის ტოლია, ამიტომ მას მიენიჭა უარყოფითი ნიშანი. საბოლოოდ ვიღებთ, რომ $11110000 = -16_{10}$.

2.8. შეკრებისა და გამოკლების ოპერაციების პროცესორული შესრულების თავისებურებები

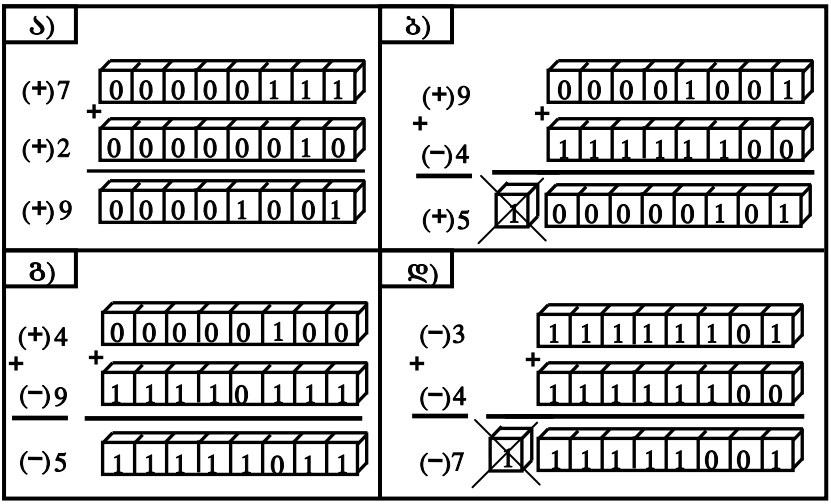
პროცესორების უმრავლესობას აქვს ორობითი რიცხვების შეკრების (***ADD***) და გამოკლების (**SUB**) ბრძანებები. მათში გამრავლებისა და გაყოფის ოპერაციები შეკრების, გამოკლებისა და ძვრის ოპერაციების დახმარებით სრულდება. გამრავლების (***MUL***) და გაყოფის (***DIV***) სპეციალური ბრძანებები მხოლოდ ზოგიერთ (მაგალითად, ***Intel8086*** და ***Intel8088***) მიკროპროცესორშია გამოყენებული. ამიტომ მოცემულ პარაგრაფში ყურადღებას სწორედ შეკრებისა და გამოკლების ოპერაციების პროცესორული შესრულების თავისებურებებზე გავამახვილებთ.

შეკრებისა და გამოკლების არითმეტიკული ოპერაციების შესრულებისას პროცესორი დადებით რიცხვებს აღიქვამს პირდაპირი, ხოლო უარყოფით რიცხვებს – დამატებითი კოდების სახით. პროცესორის მიერ ამ ოპერაციების შესრულების თავისებურებებს გავეცნოთ კონკრეტულ ამოცანათა გადაწყვეტის მაგალითით.

შეკრების ოპერაციები (ნახ.*2.14*).

▲***ამოცანა 1.*** შეიკრიბოს დადებითი რიცხვები *7* და *2*.

ამოცანის გადაწყვეტა. ნაპოვნი იქნათ მოცემული დადებითი რიცხვების პირდაპირი ორობითი კოდები. *7*-ს შეესაბამება პიდაპირი კოდი ***00000111***, ხოლო *2*-ს – ***00000010*** (იხ.ცხრ.2.1). აღნიშნული კოდების შეკრებით მიიღება პირდაპირი კოდი ***00001001*** (ნახ.***2.14,ა***), რომელსაც შეესაბამება რიცხვი *9* (იხ. ***ცხრ.2.1***), ე.ი. *7+2=9*, რაც სწორია.



ნახ.2.14. *შეკრების ოპერაციების პროცესორული შესრულება*

▲***ამოცანა 2***. შეიკრიბოს რიცხვები *9* და *-4*.

ამოცანის გადაწყვეტა. რიცხვ *9*-ს შეესაბამება პირდაპირი კოდი ***00001001***, ხოლო რიცხვ *-4*-ს – დამატებითი კოდი ***11111100*** (იხ. ცხრ.***2.1***). ამ ოპერანდების შეკრებით მიიღება *9*-თანრიგიანი ორობითი რიცხვი ***100000101***, რომლის უმაღლეს თანრიგში მდგარი ბიტი *1* გადადის გადავსების ვირტუალურ თანრიგში და იკარგება (***2.14,ბ*** ნახაზზე იგი გადახაზულია). შედეგის რეგისტრში გვრჩება ორობითი რიცხვი **00000101**, რომლის უმაღლეს თანრიგში დგას ბიტი *0*, ე. ი. იგი წარმოადგენს პირდაპირი კოდს, რომელსაც შეესაბამება ათობითი რიცხვი *5* (იხ.ცხრ.***2.1***). ე. ი. *9+(-4)=5*, რაც სწორია.

▲***ამოცანა 3.*** შეიკრიბოს რიცხვები ***4*** და ***-9***.

ამოცანის გადაწყვეტა. რიცხვ ***4***-ს შეესაბამება პირდაპირი კოდი ***00000100***, ხოლო რიცხვ ***-9***-ს – დამატებითი კოდი ***11110111*** (იხ. ცხრ.***2.1***). ამ ოპერანდების შეკრებით მიიღება დამატებითი კოდი ***111111011***, რომელსაც შეესაბამება ათობითი რიცხვი ***-5*** (იხ.ცხრ.***2.1***). მაშასადამე ***4+(-9)=-5***, რაც სწორია.

▲***ამოცანა 4***. შეიკრიბოს რიცხვები ***-3*** და ***-4***.

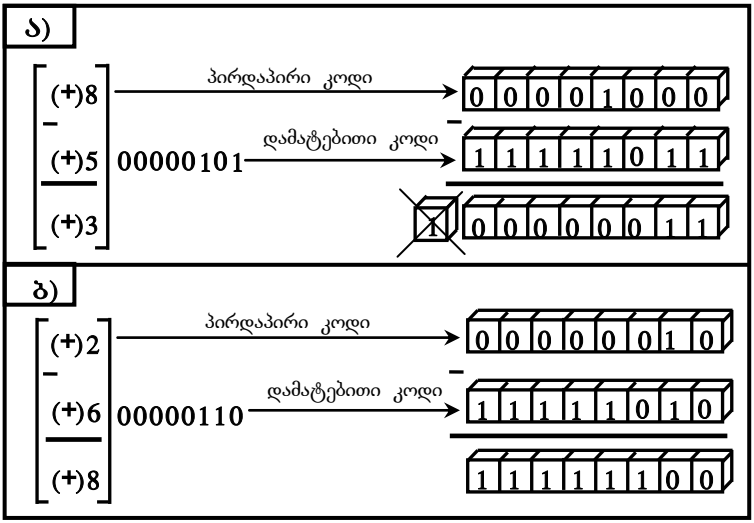
ამოცანის გადაწყვეტა. უარყოფით რიცხვებს ***-3***-სა და ***-4***-ს ***შესაბამისად შეესაბამებათ დამატებითი კოდები 11111101*** და ***11111100***, რომელთა შეკრებით მიიღება ***9***-თანრიგიანი ორობითი რიცხვი ***111111001.*** ამ უკანასკნელის უმაღლეს თანრიგში მდგარი ბიტი ***1*** გადადის გადავსების ვირტუალურ თანრიგში და იკარგება: ***2.14,ბ*** ნახაზზე იგი გადახაზულია. შედეგის რეგისტრში გვრჩება ორობითი რიცხვი ***11111001***, რომელიც წარმოადგენს ***-7***-ის დამატებით კოდს (იხ. ცხრ. ***2.1***). ე.ი. ***-3+(-4)=-7,*** რაც სწორია.

გამოკლების ოპერაციები

(ნახ.***2.15***).

▲***ამოცანა 5.*** შესრულდეს გამოკლების ოპერაცია ***8-5***.

ამოცანის გადაწყვეტა. საკლებ ***8***-ს შეესაბამება ორობითი რიცხვი (პირდაპირი კოდი) ***00001000***, ხოლო მაკლებ ***5***-ს – ორობითი რიცხვი (პირდაპირი კოდი). ***00001001***. ვიპოვოთ მაკლებ ***00001001***-ის დამატებითი კოდი. მისი ინვერსირებით მიიღება შებრუნებული კოდი ***11110110,*** ამ უკანასკნელის ინკრემენტირებით, ე.ი. მასზე ***1***-ის მიმატებით კი - საძებნი დამატებით კოდი – ***11110111***-ს. ამოცანის გადასაწყვეტად საკლების პირდაპირ ***00001000***, კოდს უნდა დავუმატოთ მაკლების დამატებითი ***11110111*** კოდი, რის შედეგადაც მიიღება ***9***–თანრიგიანი ორობითი რიცხვი ***100000011*** (ნახ.***2.15,ა***). ამ რიცხვის უმაღლს თანრიგში მდგარი ბიტი ***1*** გადადის გადავსების ვირტუალურ თანრიგში და იკარგება (***2.15,ა*** ნახაზზე იგი გადახაზულია). შედეგის რეგისტრში გვრჩება პირდაპირი (რადგან მის უმაღლეს თანრიგში დგას ბიტი ***0***) კოდი ***00000011***, რომელსაც შეესაბამება ათობითი რიცხვი ***3*** (იხ.ცხრ.***2.1***). მაშასადამე, ***8-5=3***, რაც სწორია.



ნახ.2.15. *გამოკლების ოპერაციების პროცესორული შესრულება*

▲***ამოცანა 5.*** შესრულდეს გამოკლების ოპერაცია ***2-6***.

ამოცანის გადაწყვეტა. საკლებ ***2***-ს შეესაბამება ორობითი რიცხვი (პირდაპირი კოდი) ***00000010***, ხოლო მაკლებ ***6***-ს – ორობითი რიცხვი ***00000110***. ვიპოვოთ მაკლებ ***00000110***-ის დამატებითი კოდი. მისი ინვერსირებით მიიღება შებრუნებული კოდი ***11111001*** ამ უკანასკნელის ინკრემენტირებით, ე.ი. მასზე ***1***-ის მიმატებით, კი საძებნი დამატებით კოდი – ***11111010***. ამოცანის გადასაწყვეტად საკლების პირდაპირ ***00000010*** კოდს უნდა დავუმატოთ მაკლების დამატებითი ***11111010***. კოდი. მიიღება ორობითი რიცხვი ***11111100*** (ნახ.***2.15,ა***). რადგან ამ რიცხვის უმაღლს თანრიგში დგას ბიტი ***1,*** ამიტომ იგი წარმოადგენს დამატებით კოდს. მას შეესაბამება ათობითი რიცხვი ***-4*** (იხ.ცხრ.***2.1***). ე.ი. ***2-6 =-4,*** რაც სწორია.

2.9. ელემენტალური ლოგიკური ფუნქციები

ლოგიკა აზროვნების პროცესის შემსწავლელი მეცნიერებაა. აღნიშნულიდან გამომდინარე პროცესორის მიერ ლოგიკური ფუნქციების რეალიზება ადამიანის აზროვნებასთან მიახლოებული ქმედებების შემსრულებელი ინტელექტუალური სისტემების შექმნისაკენ გადადგმული უმნიშვნელოვანესი ნაბიჯია. უშუალოდ ***ლოგიკური ოპერაციების ფორმულირებამდე*** გავეცნოთ ***ლოგიკური ფუნქციის ცნებას***.

მსჯელობების ურთიერთდაკავშირებით შესაძლებელია მივიღოთ ახალი მსჯელობა (იხილეთ ზემოთ). გარდა ამისა, ***ორნიშნა ფორმალურ ლოგიკას*** აინტერესებს არა მსჯელობის შინაარსი, არამედ მისი ჭეშმარიტობა და მცდარობა. გარკვეული მსჯელობა აღვნიშნოთ y_i სიმბოლოთი და ჩავთვალოთ $x_i=0$, თუ იგი მცდარია და $x_i=1$ – თუ იგი ჭეშმარიტია. ე.ი. x ორობითი ცვლადია: $x_i \in \{0;1\}$. ორობით ცვლადს ***ლოგიკური ცვლადი*** ეწოდება.

ლოგიკურ x_i ცვლადებზე დამოკიდებულ $y = f(x_1,\ldots, x_n)$ ფუნქციას, რომლისთვისაც სამართლიანია გამოსახულება $y, x_i \in \{0,1\}, i = 1,\ldots,n,$ ***ლოგიკური ფუნქცია*** ეწოდება.

ლოგიკური ფუნქციების განსაზღვრების ცხრილურ ფორმას წარმოადგენს 2^n რაოდენობის მწკრივის მქონე ***ჭეშმარიტობის ცხრილი***, სადაც n არის ლოგიკური ფუნქციის არგუმენტების რაოდენობა. მის მარცხენა ნაწილში ჩამოწერილია არგუმენტების მნიშვნელობათა ყველა ნაკრები, ხოლო მარჯვენა ნაწილში – ფუნქციის მნიშვნელობები ამ ნაკრებებზე.

ცხრ.2.4. *n=0,...,5 არგუმენტებზე დამოკიდებული ლოგიკური ფუნქციების N რაოდენობები*

n	***0***	***1***	***2***	***3***	***4***	***5***
N	*2*	*4*	*16*	*256*	*65536*	*4 294 967 296*

განასხვავებენ ფიქციურ და არსებით არგუმენტებს. ფიქციური არგუმენტი ეწოდება ისეთ არგუმენტს, რომელის მნიშვნელობის ცვლილება ვერ ცვლის ფუნქციის მნიშვნელობას. არსებითი არგუმენტი ისეთი არგუმენტია, რომლის მნიშვნელობის ცვლილება ცვლის ფუნქციის მნიშვნელობას.

ფიქციური არგუმენტების შემცველ ლოგიკურ ფუნქციას **გადაგვარებული ლოგიკური** ფუნქცია ეწოდება. რადგან ფიქციური არგუმენტები ვერ ცვლის ლოგიკური ფუნქციის მნიშვნელობას, ამიტომ შეგვიძლია ისინი ფუნქციიდან გამოვრიცხოთ, რის შედეგადაც ფუნქცია გადაიქცევა ***არაგადაგვარებულ ლოგიკურ ფუნქციად***

n არგუმენტზე დამოკიდებული ლოგიკური ფუნქციების ***N*** რაოდენობა განისაზღვრება ფორმულით:

$$N = 2^{2^n}.$$

2,4 ცხრილში მოცემულია ***n=0,..,5*** არგუმენტებზე დამოკიდებული ლოგიკური ფუნქციების ***N*** რაოდენობები. განვიხილოთ ისინი.

1) n=0 არგუმენტებზე დამოკიდებული ფუნქციების რაოდენობაა ***2***. ესენია კონსტანტა ***0*** (***y=0***) და კონსტანტა ***1*** (***y=1***).

2) n=1 არგუმენტებზე დამოკიდებული ფუნქციების რაოდენობაა ***4***. ამ ფუნქციებიდან ორი მათგანი გადაგვარებულია, ე. ი. ფიქციურია მათში არსებული ერთადერთი არმენტები, რომელთა გამორიცხვით ისინი გადაიქცევა ***0*** რაოდენობის არგუმენტზე დამოკიდებულ ლოგიკურ ფუნქციებად, ე. ი. კონსტანტა ***0***-ად და კონსტანტა ***1***-ად. მაშასადამე, არსებობს ერთ არგუმენტზე დამოკიდებული მხოლოდ ორი არაგადაგვარებული ლოგიკური ფუნქცია. მათი სახელწოდებები, ჭეშმარიტობის ცხრილები და მათემატიკური გამოსახულებები ***2.5*** ცხრილშია მოცემული, საიდანაც ჩანს, რომ უარყოფის ლოგიკური ფუნქცია იღებს არგუმენტის საწინააღმდეგო მნიშვნელობას, ხოლო გამეორების ლოგიკური ფუნქცია იმეორებს არგუმენტის მნიშვნელობას.

3) n=2 არგუმენტებზე დამოკიდებული ფუნქციების რაოდენობაა ***16*** (იხ. ცხრ.***2.4***), რომელთაგან გადაგვარებულია ***6*** ფუნქცია. ამ ფუნქციებიდან ორ მათგანში ფიქციურია ორივე, ხოლო დანარჩენ ოთხში – თითო-თითო არგუმენტი. ფიქციური ორი არგუმენტის შემცველი გადაგვარებული ფუნქციებიდან ფიქტიური არგუმენტების გამორიცხვის შედეგად მიიღება ***0*** არგუმენტზე დამოკიდებული ლოგიკური ფუნქციები, ანუ კონსტანტა ***0*** და კონსტანტა ***1***.

ფიქციური ერთი არგუმენტის შემცველი ოთხი გადაგვარებული ფუნქციებიდან:

▲ ორი მათგანი ისეთია, რომ მათგან ფიქციური არგუმენტების გამორიცხვით მიიღება არაგადაგვარებული ფუნქციები, რომელთაგან-

ცხრ.2.5, *ერთ და ორ არგუმენტებზე დამოკიდებული გადაუგვარებელი ლოგიკური ფუნქციების ჭეშმარიტობის ცხრილები და მათემატიკური გამოსახულებები*

სახელწოდება	ჭეშმარიტობის ცხრილი	მათემატიკური გამოსახულება	სახელწოდება	ჭეშმარიტობის ცხრილი	მათემატიკური გამოსახულება
1)უარყოფის ფუნქცია 2)**არა** ფუნქცია	x | y 0 | 1 1 | 0	$y = \overline{x}$	1)გამეორების ფუნქცია	x | y 0 | 0 1 | 1	$y = x$
1)დიზიუნქცია 2)ლოგიკური შეკრება 3)**ან** ფუნქცია	x_1 x_2 | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	$y = x_1 \vee x_2 =$ $= x_1 + x_2.$	ეკვივალენტობა	x_1 x_2 | y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1	$y = x_1 \approx x_2$
1)კონიუნქცია 2) ლოგიკური გამრავლება 3)**და** ფუნქცია	x_1 x_2 | y 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	$y = x_1 \,\&\, x_2 =$ $= x_1 \wedge x_2 =$ $x_1 x_2$	პირდაპირი იმპლიკაცია	x_1 x_2 | y 0 0 | 1 0 1 | 0 1 0 | 1 1 1 | 1	$y = x_1 \rightarrow x_2$
1)არაერთმიშვნელიანობა 2)გამომრიცხველი **ან** ფუნქცია 3)2-ის მოდულით შეკრება	x_1 x_2 | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	$y = x_1 \oplus x_2$	უკუიმპლიკაცია	x_1 x_2 | y 0 0 | 1 0 1 | 0 1 0 | 1 1 1 | 1	$y = x_1 \leftarrow x_2$
1)დიზიუნქციის უარყოფა 2)**ან-არა** ფუნქცია 3) **შების** ფუნქცია 4)**პირსის** ისარი	x_1 x_2 | y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	$y = \overline{x_1 \vee x_2} =$ $= \overline{x_1 + x_2} =$ $= \overline{x_1}\,\overline{x_2} =$ $= x_1 \downarrow x_2.$	პირდაპირი იმპლიკაციის უარყოფა	x_1 x_2 | y 0 0 | 0 0 1 | 1 1 0 | 0 1 1 | 0	$y = \overline{x_1 \rightarrow x_2}$
1)კონიუნქციის უარყოფა 2)**და-არა** ფუნქცია 3)**შეფერის** ფუნქცია	x_1 x_2 | y 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	$y = \overline{x_1 \,\&\, x_2} =$ $= \overline{x_1 \wedge x_2} =$ $\overline{x_1}\ \overline{x_2}\ .$	უკუიმპლიკაციის უარყოფა	x_1 x_2 | y 0 0 | 1 0 1 | 0 1 0 | 1 1 1 | 1	$y = \overline{x_1 \leftarrow x_2}$

აც ერთი იმეორებს x_1 არგუმენტის ($y = x_1$), ხოლო მეორე - x_2 არგუმენტის ($y=x_2$) მნიშვნელობას, ე.ი. ისინი ერთ ცვლადზე დამოკიდებული გამეორების ფუნქციებია.

▲ დანარჩენი ორი კი ისეთია, რომ მათგან ფიქციური არგუმენტების გამორიცხვით მიიღება არაგადაგვარებული ფუნქციები, რომელთაგანაც ერთი იღებს x_1 არგუმენტის, ხოლო მეორე - x_2 არგუმენტის მნიშვნელობის შებრუნებულ მნიშვნელობას. ისინი ერთ ცვლადზე დამოკიდებულ უარყოფის ფუნქციებია ფუნქციებია და შესაბამისად აღინიშნება ასე: $y = \overline{x_1}$ და $y = \overline{x_2}$.

დანარჩენი ***10*** ფუნქცია არ შეიცავს ფიქციურ არგუმენტებს, ე.ი. ისინი ***ორ არგუმენტზე დამოკიდებული არაგადაგვარებული ლოგიკური ფუნქციებია.*** მათი სახელწოდებები, ჭეშმარიტობის ცხრილები და მათემატიკური გამოსახულებები ***2.5*** ცხრილშია მოყვანილი.

აღნიშნული ლოგიკური ფუნქციები შეიძლება შემდეგ ორ ქვეჯგუფად დავყოთ:

▲ ***სკალარული (არაორიენტირებული) ტიპის ლოგიკური ფუნქციების ქვეჯგუფი***. მასში გაერთიანებულია შემდეგი ექვსი ლოგიკური ფუნქცია: დიზიუნქციის, კონიუნქციის, ***2***-ის მოდულით შეკრების, დიზიუნქციის უარყოფის, კონიუნქციის უარყოფისა და ეკვივალენტობის ფუნქცია.

აღნიშნულ ჯგუფში შემავალი ფუნქციების მნიშვნელობები არგუმენტების გადანაცვლებით ფუნქციის მნიშვნელობა არ იცვლება, ე. ი. სრულდება ტოლობა $f(x_1, x_2) = f(x_2, x_1)$.

▲ ***ვექტორული (ორიენტირებული) ტიპის ლოგიკური ფუნქციების ქვეჯგუფი.*** მასში გაერთიანებულია შემდეგი ოთხი ლოგიკური ფუნქცია: პირდაპირი იმპლიკაციის, უკუიმპლიკაციის, პირდაპირი იმპლიკაციის უარყოფისა და უკუიმპლიკაციის ლოგიკური ფუნქციები.

აღნიშნულ ჯგუფში შემავალი ფუნქციების მნიშვნელობა იცვლება არგუმენტების გადანაცვლებით, ე. ი. სრულდება უტოლობა $f(x_1, x_2) \neq f(x_2, x_1)$.

n = 1,2 არგუმენტებზე დამოკიდებულ არაგადაგვარებულ ლოგიკურ ფუნქციებს ***ელემენტალური ლოგიკური ფუნქციები*** ეწოდება.

4) n=3;4 **არგუმენტებზე დამოკიდებული ფუნქციების** რაოდენობა შესაბამისად არის ***65556*** და ***4294967296*** (იხ.***ცხრ. 2.4***). მათი განხილვა არაა საჭირო, რადგან ელემენტალური ლოგიკური ფუნქციების სიმრავლიდან შეიძლება ამოვირჩიოთ ისეთი ფუნქციები, რომელთა დახმარებითაც შევძლებთ გამოვსახოთ ნებისმიერი სირთულის ლოგიკური ფუნქცია. ასეთი ფუნქციებისაგან შემდგარ სისტემას

ელემენტალური ლოგიკური ფუნქციების ფუნქციურად სრული სისტემა ეწოდება. ფუნქციურად სრულია შემდეგი სისტემები: S_1 = (ინვერსია, დიზიუნქცია, კონიუნქცია); S_2 = {დიზიუნციის უარყოფა}, S_2 = {კონიუნქციის უარყოფა}.

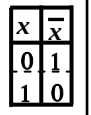
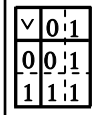
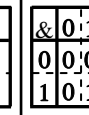
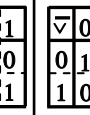
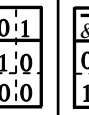
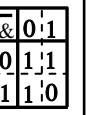
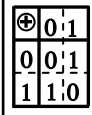
S_1 სისტემის გამოყენებისას ნებისმიერი სირთულის ლოგიკური ფუნქცია შეიძლება ინვერსიის, დიზიუნქციისა და კონიუნქციის მეშვეობით გამოვსახოთ. S_2 სისტემა ნებისმიერი ლოგიკური ფუნქციის დიზიუნქციის უარყოფის ფუნქციით, ხოლო S_3 სისტემა – კონიუნქციის უარყოფის ფუნქციით გამოსახვის საშუალებას გვაძლევს.

S_1 სისტემის ღირსებაა ის, რომ ადვილია მის მიერ გამოსახული ლოგიკური ფუნქციების გარდაქმნა, ხოლო პრაქტიკულად უფრო მოსახერხებელია S_2 და S_3 სისტემებში შემავალი ფუნქციებით გამოსახული ლოგიკური ფუნქციის მარეალიზებელი მოწყობილობის სინთეზი. ამიტომ გამოიყენება ასეთი მიდგომა: პრაქტიკულად სარეალიზაციო ლოგიკური ფუნქცია დასაწყისში გამოისახება S_1 სისტემაში შემავალი ელემენტებით და იგი სათანადოდ გარდაიქმნება. მიღებული ფუნქცია გამოისახება S_2 ან S_3 სისტემაში შემავალი ლოგიკური ფუნქციით და აიგება მიღებული ფუნქციის მარეალიზებელი მოწყობილობა.

2.10. ლოგიკური ოპერაციები და მათი რეალიზება

ელემენტალური ლოგიკური ფუნქციის მარეალიზებელ დისკრეტულ მოწყობილობას ***ლოგიკური ელემენტი***, ხოლო ამ ელემენტის მიერ შესრულებულ ოპერაციას – ***ლოგიკური ოპერაცია*** ეწოდება. ზოგიერთი ლოგიკური ელემენტის პირობითი აღნიშვნა და მის მიერ რეალიზებული ლოგიკური ოპერაცია ***2.16*** ნახაზზეა მოყვანილი. ***არა***-ფუნქციის შესაბამისი ლოგიკური ოპერაცია წარმოადგენს ***უნარულ*** (ერთოპერანდიან) ***ოპერაციას***, რომელიც თითოეული ბიტის მნიშვნელობას ცვლის საწინააღმდეგო მნიშვნელობით. დანარჩენი ოპერაციები ბინარული ოპერაციებია, რომლებიც ***2.16*** ნახაზზე მოყვანილი ცხრილების შესაბამისად სრულდება.

ნებისმიერი დისკრეტული მოწყობილობა ახდენს გარკვეული რა-

ლოგიკური ფუნქცია	***არა*** ფუნქცია	***ან*** ფუნქცია	***და*** ფუნქცია	***ან-არა*** ფუნქცია	***და-არა*** ფუნქცია	***mod***-ით შეკრება	ეკვივალენტობა
ლოგიკური ელემენტი							
ლოგიკური ოპერაცია							

ნახ.2.16 *ლოგიკური ფუნქციები, ლოგიკური* ელემენტების პირობითი აღნიშვნები და ლოგიკური ოპერაციები.

ოდენობის ლოგიკური ფუნქციის რეალიზებას. ნებისმიერი ეს ფუნქცია შეიძლება გამოისახოს ელემენტალური ლოგიკური ფუნქციებით, რომელთა რეალიზებას ასრულებს ლოგიკური ელემენტები. მაშასადამე, ნებისმიერი დისკრეტული მოწყობილობა შეიძლება ავაგოთ ლოგიკური ელემენტებით, ე. ი. ლოგიკური ელემენტები დისკრეტული მოწყობილობების ასაგებად საჭირო საელემენტო ბაზაა.

ლოგიკური ოპერაციების რეალიზება შეიძლება პროგრამულადაც. ამ შემთხვევაში ბინარული ოპრაციის თითოეული ბიტი ერთმანეთს უდარდება და მოცემული ოპერაციის შესაბამისი ცხრილის შესაბამისად განისაზღვრება შედეგის შესატყვისი ბიტის მნიშვნელობა.

2.17,ა,ბ,გ,დ,ე,ვ ნახზების ზედა ნაწილში მოცემულია ორი ბიტის შედარების წესები, ხოლო ქვედა ნაწილში ნაჩვენებია, რომ ***10110010*** და ***00111001*** ბაიტებზე:

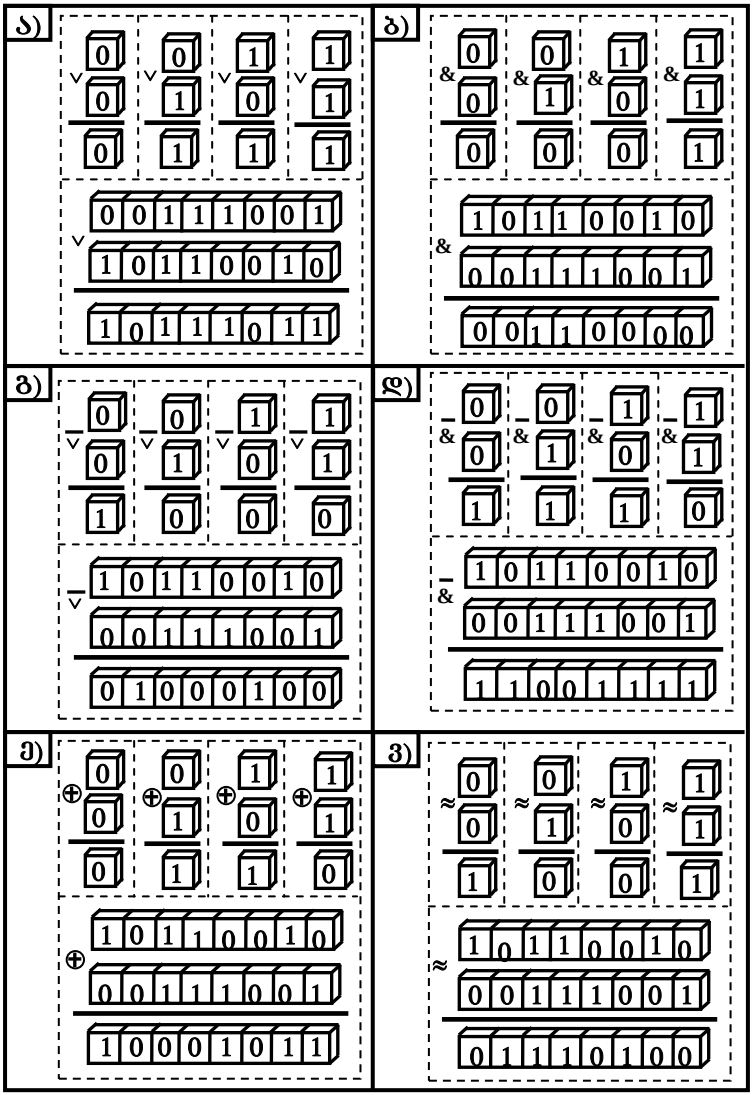
▲ ***და***-ოპერაციის ჩატარებით მიიღება ბაიტი ***10111011*** ბაიტი (ნახ.***2.17,ა***);

▲ ***ან***-ოპერაციის ჩატარებით მიიღება ბაიტი ***00110000*** ბაიტი (ნახ.***2,17,ბ***);

▲ ***ან-არა*** ოპერაციის ჩატარებით მიიღება ბაიტი ***01000100*** ბაიტი (ნახ.***2.17,გ***);

▲ ***და-არა***-ოპერაციის ჩატარებით მიიღება ბაიტი ***110001111*** ბაიტი (ნახ.***2,17,დ***);

▲ ***2-ის მოდულით შეკრების***-ოპერაციის ჩატარებით მიიღება ბაიტი ***10001011*** ბაიტი (ნახ.***2.17,ე***);



ნახ.2.17. ლოგიკური ოპერაციების შესრულება ბიტებსა და ბაიტებზე

▲ ***ეკვივალენტობის***-ოპერაციის ჩატარებით მიიღება ბაიტი ***10000100*** ბაიტი (ნახ.***2.17,დ***).

2.11. რეგისტრებში ორობითი სიტყვების ძვრის ოპერაციები

ოპერაციებს, რომლებიც საშუალებას გვაძლევს რეგისტრში ჩაწერილი ორობითი კოდი თითო-თითო ბიტებით დავძრათ მარცხნივ (უფროსი ბიტის მხარეზე) ან მარჯვნივ (უმცროსი ბიტის მხარეზე), ***ძვრის ოპერაციები*** ეწოდება და ისინი მიეკუთვნება ლოგიკური ოპერაციების ჯგუფს. მათი გამოყენება საშუალებას გვაძლევს:

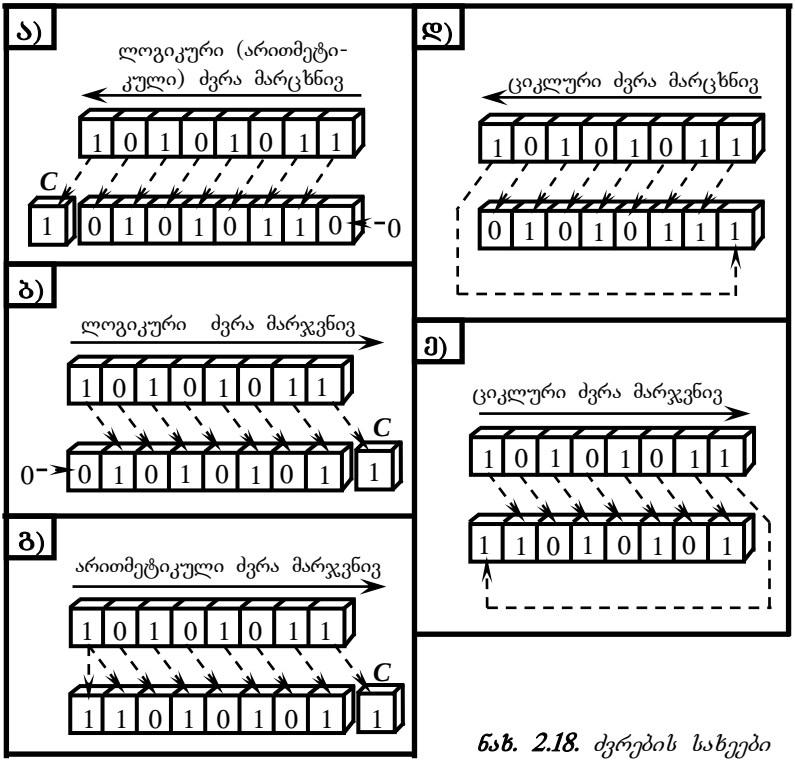
1) ცალ-ცალკე დავამუშაოთ თითოეული ბიტი; ***2***) რიცხვი სწრაფად გავამრავლოთ ან გავყოთ თვლის იმ სისტემის ფუძის ხარისხებზე, რომელიც გამოყენებულია ამ რიცხვის წარმოსადგენად. კერძოდ, ***თვლის ორობითი სისტემისას*** (რომელიც პროცესორულ სისტემაშია გამოყენებული) ძვრების საშუალებით ოპერანდად გამოყენებული ორობითი რიცხვი შეგვიძლია სწრაფად გავამრავლოთ და გავყოთ ***2***-ის ხარისხებზე (***2***-ზე, ***4***-ზე, ***8***-ზე, ***16***-ზე, ***32***-ზე და ა.შ.).

ზემოთ აღნიშნულის გამო დამპროგრამებლები ძალიან აფასებენ ძვრების ყველა სახესხვაობას და ფართოდ იყენებენ მათ.

ძვრის ოპერაციის იდეა ძალიან მარტივია: კოდის (ორობითი რიცხვის) ყველა ბიტი ერთდროულად გადაიძვრება მარცხენა ან მარჯვენა მეზობელ თანრიგებში. ამ დროს სპეციფიკურ მდგომარეობაში არის ორობითი რიცხვის ორი განაპირა (უმცროსი და უფროსი) ბიტი, რომლებსაც «მეზობელი» არ ჰყავს. განსაზღვრულობისათვის გავარჩიოთ ორობითი რიცხვის მარცხნივ ძვრა (ნახ ***2.18,ა***). ყველაზე უმცროსი ბიტისათვის (ნახაზზე იგი მარჯვნიდან განაპირაა) მონაცემებს ვერსაიდან ვერ ავიღებთ, ამიტომ მასში უბრალოდ ***0*** შეიტანება. ყველაზე უფროსი (მარცხენა განაპირა ბიტი უნდა დაიკარგოს, რადგან მისი შესანახი ადგილი არ არსებობს. მონაცემები რომ არ დაიკარგოს, ამ თანრიგის შიგთავსი კოპირდება სპეციალურ ***C*** ბიტში (პროცესორის მდგომარეობის ***PSW*** რეგისტრში), რომელსაც მუშაობაში იყენებს პროცესორი.

განხილული ტიპის ძვრას უწოდებენ ***ლოგიკურ ძვრას.*** იგი შეიძლება გამოვიყენოთ სწრაფი გამრავლებისა და გაყოფისათვის. გან-

ვიხილოთ, მაგალითად, ***8***-თანრიგიანი ორობითი რიცხვი ***00100110,*** რომლის შესაბამისი ათობითი რიცხვია 38_{10}; ამ რიცხვის ***მარცხნივ***



ნახ. 2.18. *ძვრების სახეები*

ერთი ბიტით ძვრის შემდეგ მივიღებთ რიცხვს ***01001100***, რომლის შესაბამისი ათობითი რიცხვია 76_{10} და იგი ორჯერ აღემატება ძვრამდელი ორობითი რიცხვის შესაბამის ათობით 38_{10} რიცხვს. ეს შემთხვევით არ მომხდარა: თვლის ***N***-ობით სისტემის ფუძე ***N***–ის ტოლია და თუ მასში წარმოდგენილ რიცხვს მარჯვნივ ***0***-ს მივუწერთ, იგი ***N***-ჯერ იზრდება, ხოლო ჩვენს შემთხვევაში ***N=2***.

ორობითი მთელი ***დადებითი რიცხვის*** მარჯვნივ ერთი ბიტით ძვრის დროს ნებისმიერი ***ლუწი რიცხვი*** ზუსტად ***2***-ჯერ მცირდება. ***კენტი რიცხვის*** დროს ნარჩუნდება მთელი ნაწილი, ხოლო ნაშთი იკარგება. მაგალითად, $00100011=35_{10}$ რიცხვის მარჯვნივ ერთი ბიტით დროს მიიღება $00010001=17_{10}$.

ახლა განვიხილოთ, რა ხდება ***უარყოფითი რიცხვებისათვის.*** ორობით სისტემაში უარყოფითი რიცხვები შეიძლება წარმოდგენილი იქნეს სხვადასხვა ფორმით, კერძოდ, პირდაპირი, შებრუნებული და დამატებითი კოდების სახით. მიკროპროცესორულ სისტემებში უარყოფით რიცხვებს წარმოადგენენ დამატებითი კოდების სახით, რადგან ასეთი ფორმით მათი წარმოდგენა სხვა ღირსებებთან ერთად შეკრების ფორმით გამოკლების ოპერაციის შესრულების საშუალებასაც იძლევა.

როგორც ***2.1*** ცხრილიდან ჩანს, უარყოფითი ***«-8»*** რიცხვის შესაბამის ***8***-ბიტურ დამატებით კოდს წარმოადგენს ***1111 1000.*** მისი ***მარცხნივ ერთი ბიჯით ძვრისას*** მიიღება რიცხვი ***1111 0000,*** რომელიც წარმოადგენს ***«-16»*** რიცხვის დამატებით კოდს, ე.ი. მნიშვნელობა ზემოთ განხილული შემთხვევის ანალოგურად გაორმაგდა. ე. ი. ზემოთ აღმოჩენილი კანონზომიერება დაცულია!

აღნიშნული რიცხვის მარჯვნივ ერთი ბიჯით დაძვრისას ზემოთ აღმოჩენილი კანონზომიერების ძალით ***2***-ჯერ უნდა შემცირდეს, მაგრამ ეს არ ხდება. მართლაც, მარჯვნივ ერთი ბიჯით დაძვრით მიიღება რიცხვი ***0111 1100***, რომელიც დადებითი (და არა უარყოფითი) რიცხვის კოდია! საქმე ის არის, რომ უარყოფითი რიცხვების მარჯვნივ დაძვრის დროს დადებითი რიცხვების მარჯვნივ დაძვრისაგან განსხვავებით უფროსი თანრიგი უნდა შევავსოთ არა ***0***-ით, არამედ ***1***-ით. შექმნილი მდგომარეობიდან გამოსასვლელის საპოვნელად შემოტანილი იქნა ძვრის კიდევ ერთი ნაირსახეობა – ***არითმეტიკული ძვრა.*** იგი ლოგიკური ძვრისაგან მხოლოდ იმით განსხვავდება, რომ მისი უფროსი (ნიშნის) ბიტი არ იცვლება, ე. ი. რიცხვის ნიშანი უცვლელი რჩება.

ჩვენ მიერ ზემოთ განხილულ ***1111 1100*** კოდს მარჯვნივ ერთი ბიტით არითმეტიკულად დაძვრისას მიიღება ***1111 1100***, რომელიც არის ***"-4"*** რიცხვის დამატებითი კოდი (იხ. ცხრ. ***2.1***).

2.12. ლოგიკური ნიღბები

1857 წელს დამუშავებული იქნა სავაჭრო ფლოტისათვის განკუთვნილი კოდური სიგნალების სისტემა, რომელიც ***18 ასოს*** მეშვეობით სპეციფიკური ინფორმაციის მანძილზე გადაცემის საშუალებას

იძლეოდა. აღნიშნულ სისტემას დიდი ხნის განმავლობაში იყენებდნენ და ის შუქურმა სიგნალიზაციამ მთლიანად მხოლოდ ***2012*** წელს შეცვალა.

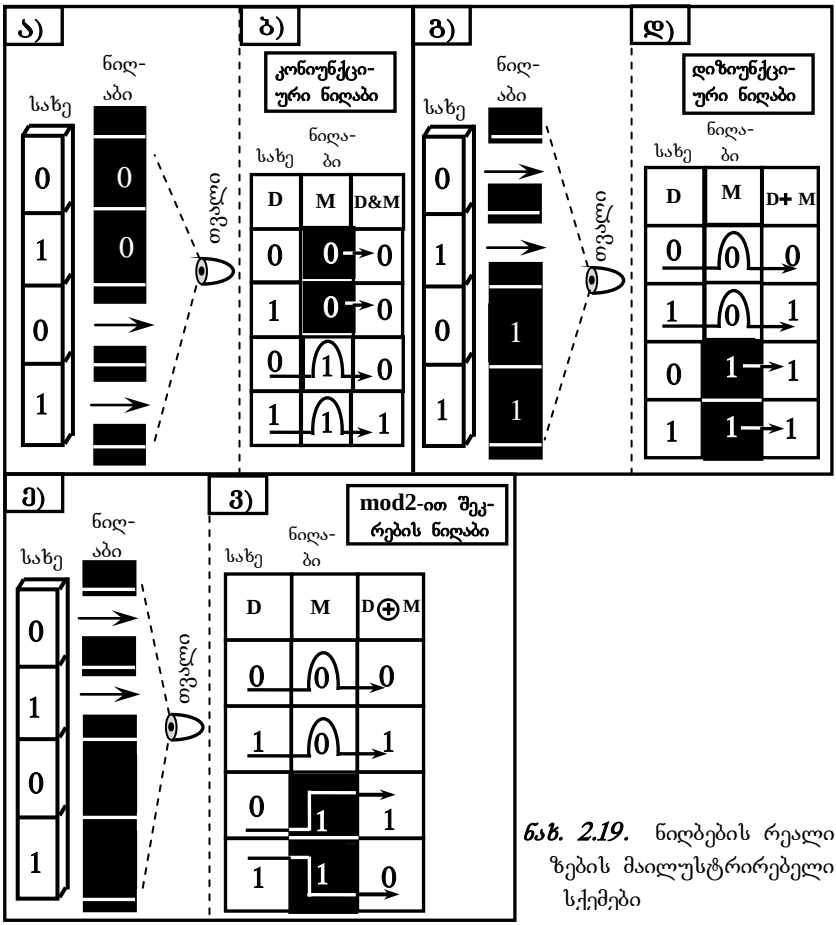
XVIII საუკუნეში იტალიაში წარმოიშვა სპეციფიკური სახის ბალი, რომლის მონაწილეებს ***ნიღბები*** ეკეთათ. ასეთ ბალს უწოდეს იტალიური სიტყვა «***maschera***»-დან ნაწარმოები სახელი ***"მასკარადი"*** (***რედაქტორის შენიშვნა***: აღნიშნული ტერმინის არაბულ-ქართულ შესატყვისს ***"მასხარა"*** წარმოადგენს). ნიღაბზე არსებობდა თვალისა და პირისათვის განკუთვნილი ამონაჭრები, ხოლო დანარჩენ ნაწილზე დახატული იყო ის სახე, რომლითაც სურდა წარდგომოდა სხვებს კონკრეტული ნიღბის მფლობელი.

მოდერნიზებული სახის ალამები და ნიღბები გამოყენებული იქნა მიკროპროცესორულ სისტემაში სპეციფიკური ოპერაციების ორგანიზებისათვის. განვიხილოთ ისინი.

მიკროპროცესორულ სისტემაში შემოიტანეს სპეციალური ***2*** ბაიტის სიგრძის მქონე «მდგომარეობის რეგისტრი», რომლის თითოეულ ბიტს ეწოდა «***ალამი***». ამგვარად მოდერნიზებული ***ალმის დაყენება*** გულისხმობს ბიტის მნიშვნელობად ***«1»***-ის ჩაწერას, ხოლო ***ჩამოგდება*** – ბიტის მნიშვნელობად ***«0»***-ის ჩაწერა. აღნიშნული ალმები (ბიტები) გამოიყენება სხვადასხვა სახის გადასვლის ბრძანებების რეალიზებისათვის.

რაც შეეხება ტერმინ ***«ნიღაბს»,*** იგი უწოდეს კონსტანტას, რომელიც განსაზღვრავს მრავალთანრიგიანი რიცხვის ბიტების იმ არეს, სადაც გამოყენებული უნდა იქნეს გარკვეული არითმეტიკული თუ ლოგიკური ოპერაცია. მისი საშუალებით გარკვეული ოპერაციის ჩატარებისას შეიძლება ზოგიერთი ბიტის (ისევე, როგორც მასკარადის დროს სახის გარკვეული ნაწილების) დაფარვა, ან როგორც პირვანდელი, ასევე ინვერსირებული სახით მათი გამოჩენა.

2.19,ა ნახაზზე ნაჩვენებია შემთხვევა, რომლის დროსაც ***0101*** კოდიან რეგისტრზე ანუ სახეზე, ჩამოფარებულია ნიღაბი. რეგისტრის ზედა თანრიგებს ფარავს ნიღაბი, რომელზეც დაწერილია ციფრები ***0;0***, ხოლო ქვედა ორ თანრიგთან არსებობს ამონაჭრები, საიდანაც თვალი ხედავს ციფრებს ***01***. ამის გამო საბოლოოდ თვალი რეგისტრიდან ამოიკითხავს კოდს ***0001.***



ნახ. 2.19. ნიღბების რეალიზების მაილუსტრირებელი სქემები

ნიღაბზე დაწერილ ***0;0*** ბიტებს თუ უცვლელად დავტოვებთ, ხოლო ნიღბის ამონაჭრებს შევცვლით ბიტებით ***1;1***, მაშინ მივიღებთ, რომ განხილულ ***რეალურ ნიღაბს*** შეესაბამება ***ლოგიკური ლოგიკური ნიღაბი 0011*** (იხ. ***1.9,ა*** ნახაზის მარჯვენა მხარე), რომელიც თავის ფუნქციას ასრულებს რეგისტრში არსებულ კოდზე ლოგიკური გამრავლების (კონიუნქციის) მეშვეობით. მართლაც:

$$0101 \;\&\; 0011 = 0001.$$

აღნიშნულის გამო ***0011*** ნიღაბს ეწოდება ***კონიუნქციური ნიღაბი.*** კონიუნქციურ ნიღაბს თუ ჩამოვაფარებთ რომელიმე რეგისტრს, მაშინ

მოხდება ამ რეგისტრის იმ თანრიგებში არსებული ბიტების «ჩამოგდება», რომლებზეც აფარებულია კონიუნქციური ნიღბის ***0***-ის ტოლი ბიტები, ე.ი. იგი გამოიყენება გარკვეული ბიტების «ჩამოსაგდებად». ***2.19,ბ*** ნახაზზე ილუსტრირებულია კონიუნქციური ***M=0011*** ნიღბის საშუალებით ***D=0101*** სახის ზედა ორი მაღალი ბიტის «ჩამოგდების» შემთხვევა.

2.19,გ ნახაზზე ნაჩვენებია შემთხვევა, რომლის დროსაც «სახეზე» ჩამოფარებულია ნიღაბი, რომლის ზედა ნაწილები ამოჭრულია, ხოლო ქვემო ნაწილებზე დაწერილია ციფრები ***1;1.*** ამის გამო თვალი ხედავს კოდს ***0111***.

რეალური ნიღბის ამონაჭრებს თუ შევცვლით «***0***»-ებით, ხოლო მასზე არსებულ ნაწერებს შევინარჩუნებთ, მივიღებთ ლოგიკურ ნიღბის ***0011*** კოდს (ნახ.***2.19,გ***); სახე ***0101***-თან მისი დიზიუნქციისას ვიღებთ თვალის მიერ ზემოთ დანახულ ***0111*** რიცხვს. დიზიუნქციის ოპერაციის შესრულების საჭიროების გამო მოცემულ ლოგიკური ნიღბის კოდს ***დიზიუნქციური ნიღაბი*** ეწოდება. დიზიუნქციურ ნიღაბს თუ ჩამოვაფარებთ რომელიმე რეგისტრს, მაშინ მოხდება ამ რეგისტრის იმ თანრიგებში არსებული ბიტების «დაყენება», რომლებზეც აფარებულია დიზიუნქციური ნიღბის ***1***-ის ტოლი ბიტები, ე.ი. იგი გამოიყენება გარკვეული ბიტების «დასაყენებლად». ***2.19,ბ*** ნახაზზე ილუსტრირებულია კონიუნქციური ***M=0011*** ნიღბის საშუალებით ***D=0101*** სახის ზედა ორი მაღალი ბიტის «დაყენების» შემთხვევა.

2.19,ე,ვ ნახაზებზე ნაჩვენებია ე.წ. ***mod.2-ის ნიღბის*** გამოყენების შემთხვევა. ***mod.2-ით შეკრების ნიღაბს*** თუ ჩამოვაფარებთ რომელიმე რეგისტრს, მაშინ მოხდება ამ რეგისტრის იმ თანრიგებში არსებული ბიტების ინვერსირება, რომლებზეც აფარებულია დიზიუნქციური ნიღბის ***1***-ის ტოლი ბიტები, ე. ი. იგი გამოიყენება გარკვეული ბიტების ინვერსირებისათვის.

მაშასადამე: 1) **კონიუნქციური ნიღაბი** გამოიყენება გარკვეული ბიტების ჩამოსაგდებად; 2) **დიზიუნქციური ნიღაბი** გამოიყენება გარკვეული ბიტების დასაყენებლად; 3) ***mod.2-ით შეკრების ნიღაბი*** გამოიყენება გარკვეული ბიტების ინვერსერებისათვის.

III თავი
ციფრული მოწყობილობების ლოგიკური რეალიზების საფუძვლები

კარგ თეორიაზე უფრო პრაქტიკული არაფერი არ არის.
გ. რ. კირჰოფი (*1821-1887*)

3.1. ლოგიკის ალგებრის ცნება

ალგებრას, რომელშიც მზიდად წოდებულ ***G*** სიმრავლეს წარმოქმნის ორობითი ცვლადები, ხოლო სიგნატურას (ოპერაციების ნაკრებს) - კონიუნქცია, დიზიუნქცია და ინვერსია, ***ლოგიკის*** (***ორობითი***) ***ალგებრა*** ეწოდება. ხშირად მას ამ ალგებრის შემქმნელი ***ჯორჯ ბულის*** საპატივცემლოდ ***ბულის ალგებრასაც*** უწოდებენ.

ცხრ.3.1. ლოგიკის ალგებრის ძირითადი კანონები

1). ასოციურობის კანონი: $x_1(x_2\,x_3) = (x_1x_2)\,x_3$; $(x_1+x_2)+\,x_3 = x_1+(x_2+x_3)$.	**5)** წინააღმდეგობრობის კანონი: $x\,\bar{x}= 0$.	**11)** დიზიუნქციის მიმართ კონიუნქციის დისტრიბუციულობის კანონი: $x_1(x_2\,x_3) = x_1x_2 + x_1x_3$.
2) კომუტატურობის კანონი $x_1x_2 = x_2x_3$; $x_1+x_2= x_1+x_2$.	**6)** დე მორგანის კანონი: $\overline{x_1\,x_2} = \bar{x}_1+\bar{x}_2$; $\overline{x_1+\,x_2} = \bar{x}_1\,\bar{x}_2$.	**12)** კონიუნქციის მიმართ დიზიუნქციის დისტრიბუციულობის კანონი: $x_1+ (x_2\,x_3) = (x_1+x_2)(x_1+x_3)$.
3) ორმაგი უარყოფის კანონი: $\bar{\bar{x}} = x$.	**7)** გამორიცხული მესამის კანონი: $x+\bar{x}=1$	
4) იდემპოტენტურობის კანონი $xx= x$; $x+x= x$.	**8)** შეწებების კანონი: $xy+x\bar{y} =x$ $(x+y)(x+\bar{y}) = x$	**13)** შეგრაგვნის კანონი: $x+\bar{x}y =x+y$; $x(\bar{x}+y)=xy$
	9) $x+0=x$; $x\cdot0=0$	
	10) $x+1=1$; $x\cdot1=x$	

საყოველთაოდ ცნობილი სასკოლო ალგებრის ანალოგურად ლოგიკის ალგებრაშიც არსებობს გარკვეული კანონები, რომელთაგანაც

ზოგი მათგანი ემთხვევა სასკოლო ალგებრის კანონებს, ზოგი კი არა. კომპაქტურობისათვის ლოგიკის ალგებრის ყველა ძირითადი კანონი ***3.1*** ცხრილის სახით გვაქვს მოყვანილი. მათი ცოდნა აუცილებელია ლოგიკური ფუნქციების ანალიზური გამოსახულებათა გარდაქმისათვის (გამარტივებისათვის, ან კონკრეტული სიტუაციაზე დამოდებულებით მათთვის სასურველი ფორმის მისაცემად).

3.2. ლოგიკური ფუნქციები და მათი წარმოდგენის ფორმები

ლოგიკურ x_i ცვლადებზე დამოკიდებულ $y = f(x_1,\ldots, x_n)$ ფუნქციას, რომლისთვისაც სამართლიანია გამოსახულება $y, x_i \in \{0,1\}, i = 1,\ldots,n$, ***ლოგიკური ფუნქცია*** ეწოდება. არსებობს ლოგიკური ფუნქციების წარმოდგენის ცხრილური, ანალიზური, კოორდინატული და რიცხვითი ფორმები [*1,2*].

ცხრილურ ფორმას წარმოადგენს ჭეშმარიტობის ცხრილი (იხ. **§2.9**). იგი უმარტივესი ფორმაა, რომლიდანაც მიიღება დანარჩენი ფორმები.

ლოგიკური ფუნქციის ანალიზური ფორმებს წარმოადგენს დიზიუნქციური სრულყოფილი ნორმალური ფორმა (***ДСНФ***) და კონიუნქციური სრულყოფილი ნორმალური ფორმა (***КСНФ***).

▲ ***დიზიუნქციური სრულყოფილი ნორმალური ფორმის მიღების A_1 ალგორითმი***: ***1)*** ჭეშმარიტობის ცხრილიდან ამოვიწეროთ არგუმენტთა ის ნაკრებები, რომლებზეც ფუნქცია **1**-ის ტოლ მნიშვნელობებს იღებს; ***2)*** შევადგინოთ არგუმენტების მნიშვნელობათა ამოწერილ თითოეულ ნაკრებში შემავალი არგუმენტების კონიუნქციები. ამისათვის, თუ ნაკრებში არგუმენტის მნიშვნელობა **1**-ის ტოლია, იგი კონიუნქციაში უცვლელად, ხოლო თუ **0**-ის ტოლია, მაშინ – ინვერსირებული სახით შევიტანოთ; ***3)*** პუნქტ *2*-ში ფორმირებული კონიუნქციები შევაერთოთ დიზიუნქციათა ნიშნებით.

A_1 ალგორითმის მეორე პუნქტში შედგენილ კონიუნქციებს ***ერთიანის კონსტიტუენტები*** ანუ ***მინიტერმები*** ეწოდება. მაშასადამე, ***ДСНФ წარმოადგენს ერთიანების კონსტიტუენტთა (მინიტერმთა) დიზიუნქციას.***

განვიხილოთ ***3*** ცვლადზე დამოკიდებული ლოგიკური ფუნქცია, რომელიც «**1**»-ის ტოლ მნიშვნელობას იღებს მხოლოდ მაშინ, როდესაც შესასვლელი ცვლადების უმრავლესობა (ორი ან სა-მი ასეთი ცვლადი) «**1**»-ის ტოლია. ასეთ ლოგიკურ ფუნქციას ***მაჟორიტარული ლოგიკური ფუნქცია***; მისი ჭეშმარიტობის ცხრილი ***3.1,ა*** ნახაზზეა მოცემული. აღნიშნული ჭეშმარიტობის ცხრილისათვის შევასრულოთ A_1 ალგორითმი: ***1)*** 011; 101; 110; 111. ***2)*** $\overline{x_1}\,x_2\,x_3$; $x_1\,\overline{x_2}\,x_3$; $x_1\,x_2\,\overline{x_3}$; $x_1\,x_2\,x_3$. ***3)*** იხილეთ ნახაზი ***3.1***,ბ.

ა)

X_{10}	x_1	x_2	x_3	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

ბ) $y(x_1,x_2,x_3) = \overline{x_1}\,x_2\,x_3 + x_1\,\overline{x_2}\,x_3 + x_1\,x_2\,\overline{x_3} + x_1\,x_2\,x_3;$

გ) $y(x_1,x_2,x_3) = (x_1+x_2+x_3)(x_1+x_2+\overline{x_3})(x_1+\overline{x_2}+x_3)(\overline{x_1}+x_2+x_3);$

დ) $y(x_1,x_2,x_3) = \overline{x_1}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}\,\overline{x_2}\,x_3 + \overline{x_1}\,x_2\,\overline{x_3} + x_1\,\overline{x_2}\,\overline{x_3};$

ე) $y(x_1,x_2,x_3) = (x_1+\overline{x_2}+\overline{x_3})(\overline{x_1}+x_2+\overline{x_3})(\overline{x_1}+\overline{x_2}+x_3)(\overline{x_1}+\overline{x_2}+\overline{x_3})$

ვ) $y(x_1,x_2,x_3) = \sum(3,5,6,7)$

ზ) $y(x_1,x_2,x_3) = \prod(0,1,2,4)$

ნახ.3.1 *მაჟორიტარული ლოგიკური ფუნქციის წარმოდგენის ცხრილური (**ა**), ანალიზური (**ბ,გ,დ,ე**) და რიცხვითი (**ვ,ზ**) ფორმები.*

A_1 ალგორითმის რეალიზების ზემოთ მოყვანილ მეორე პუნქტში მოყვანილია განხილული ლოგიკური ფუნქციისათვის შედგენილი ერთიანის კონსტიტუენტები ანუ მინიტერმები, ხოლო ***3.1,ბ*** ნახაზზე მოყვანილია აღნიშნული ფუნქციის ***დსნფ.***

▲ ***კონიუნქციური სრულყოფილი ნორმალური ფორმის*** (***კსნფ***-ის) ***მიღების A_2 ალგორითმი***: ***1)*** ჭეშმარიტობის ცხრილიდან ამოვიწეროთ არგუმენტთა ის ნაკრებები, რომლებზაც ფუნქცია **0**-ის ტოლ მნიშვნელობებს იღებს; ***2)*** შევადგინოთ არგუმენტების მნიშვნელობათა ამორჩეულ თითოეულ ნაკრებში შემავალი არგუმენტების დიზიუნქციები. ამისათვის თუ ნაკრებში არგუმენტის მნიშვნელობა **0**-ის ტოლია, იგი დიზიუნქციაში უცვლელად, ხოლო თუ **1**-ის ტოლია, მაშინ - ინვერსირებული სახით გადმოვიწეროთ; ***3)*** პუნქტ ***2***-ში ფორმირებული დიზიუნქციები შევაერთოთ კონიუნქციათა ნიშნებით.

ზემოთ განხილული ალგორითმის მეორე პუნქტში შედგენილ დიზიუნქციებს ***ნულების კონსტიტუენტები*** ანუ ***მაქსიტერმები*** ეწოდება. ზოგიერთი ავტორი ნულების კონსტიტუენტებს ***ანტიკონსტიტუენტებადაც*** მოიხსენიებს. მაშასადამე, *** КНФ წარმოადგენს ნულების კონსტიტუენტთა (ანტიკონსტიტუენტთა, მაქსიტერმთა) კონიუნქციას.***

3.1,ა ნახაზზეა მოცემული ჭეშმარიტობის ცხრილისათვის შევასრულოთ A_2 ალგორითმი: *1)* 000; 001; 010; 100. *2)* $(x_1+x_2+x_3)$; $(x_1+x_2+\overline{x_3})$; $(x_1+\overline{x_2}+x_3)$; $(\overline{x_2}+x_1+x_0)$. *3)* იხილეთ ნახაზი ***3.1***,გ.

A_2 ალგორითმის რეალიზების ზემოთ მოყვანილ მეორე პუნქტში მოყვანილია განხილული ლოგიკური ფუნქციისათვის შედგენილი ნულების კონსტიტუენტები (მაქსიტერმები), ხოლო ***3.1,გ*** ნახაზზე – აღნიშნული ფუნქციის ***КНФ***.

ზოგჯერ მოსახერხებელია გამოვიყენოთ ***ДНФ***-ისა და ***КНФ***-ის ინვერსიული ფორმები.

▲ ***ДНФ***-ის ***ინვერსიული ფორმის*** მისაღებად A_1 ალგორითმის პირველ პუნქტში ჭეშმარიტობის ცხრილიდან უნდა ამოვწეროთ არგუმენტთა ის ნაკრებები, რომლებზეც ფუნქცია **0**-ის ტოლ მნიშვნელობებს იღებს, ხოლო დანარჩენი ორი პუნქტი უცვლელად შევასრულოთ. მისი გამოყენებისას ჩვენი მაგალითისათვის მივიღებთ ***3.1,დ*** ნახაზზე მოყვანილ გამოსახულებას, რომელიც ჩვენი ფუნქციის ***ДНФ***-ს ინვერსირებული ფორმაა.

▲ ***КНФ***-ის ***ინვერსიული ფორმის*** მისაღებად A_2 ალგორითმის პირველ პუნქტში ჭეშმარიტობის ცხრილიდან უნდა ამოვწეროთ არგუმენტთა ის ნაკრებები, რომლებზეც ფუნქცია **1**-ის ტოლ მნიშვნელობებს იღებს, ხოლო დანარჩენი ორი პუნქტი უცვლელად შევასრულოთ. მისი გამოყენების შემთხვევაში ჩვენი მაგალითისათვის მივიღებთ ***3.1,ე*** ნახაზზე მოყვანილ გამოსახულებას, რომელიც ჩვენი ფუნქციის ***КНФ***-ს ინვერსირებული ფორმაა.

▲ხშირად გამოსახულებების შესამოკლებლად ლოგიკურ ფუნქციებს წარმოდგენენ ათობითი რიცხვების მიმდევრობათა სახით, რომლებიც წარმოადგენს ზემოთ განხილული და ალგორითმების მეორე პუნქტში არსებული ორობითი რიცხვების ათობით ეკვივალენტებს. მიღებულ გამოსახულებებს ***ДНФ***-სა და ***КНФ***-ის შესაბამის ***რიცხვით ფორმებს*** უწოდებენ, ისინი შესაბამისად ***3.1,ე,ზ*** ნახაზებზეა მოყვანილი.

▲ განვიხილოთ მინიტერმები. ისინი წარმოადგენს შესაბამისი ლოგიკური ფუნქციის პირდაპირი ან ინვერსირებული ფორმით წარმოდგენილი არგუმენტების კონიუნქციას. ***ელემენტალური კონიუნქცია*** ეწოდება კონიუნქციას, რომელშიც ნებისმიერი ცვლადი მხოლოდ ერთხელ შედის. აგების ძალით მინიტერმი ელემენტალური კონიუნქციაა. ***ელემენტალური კონიუნქციის რანგი*** ეწოდება მასში შემავალი ცვლადების რაოდენობას. ელემენტალური კონიუნუნქციების დიზიუნქციას ***ლოგიკური ფუნქციის ნორმალური ფორმა*** (***ДНФ***) ეწოდება. ***ДНФ*** არის ***n*** არგუმენტზე დამოკიდებული ლოგიკური ფუნქციის გამოსახვა ისეთი ***ДНФ***-ის სახით, რომელიც მხოლოდ ***n*** რანგის ელემენტალური კონიუნქციებისაგან შედგება. ***n*** რანგის ორ ელემენტალური კონიუნქციას თუ აქვს Ax_i და $A\bar{x}_i$ სახე, სადაც ***A*** წარმოადგენს ***n-1*** რანგის ელემენტალურ კონიუნქციას, მაშინ მათი დიზიუნქცია შეიძლება ***A*** კონიუნქციით შეიცვალოს. მართლაც: $Ax_i + A\bar{x}_i = A\,(x_i + \bar{x}_i) = A$. ამ დროს თითქოს ორი ელემენტალური კონიუნქცია შეეწება ერთმანეთს და ერთ კონიუნქციად გადაიქცა. ამიტომ მოყვანილ გარდაქმნას ***შეწებების წესსაც*** უწოდებენ. ***ДНФ***-ში შეწებების წესების გამოყენების გზით მიიღება ***ДНФ***-ად.

ლოგიკურ ფუნქციას აქვს ერთადერთი ***ДНФ***, რომლიდანაც შეიძლება მიღებული იქნეს რამდენიმე ***ДНФ***. ***ДНФ***-ისა და ***ДНФ***-ის რანგები ეწოდება მათში შემავალი მინიტერმების რანგების ჯამს. ზემოთ აღნიშნულის გამო ***ДНФ***-ის რანგი აღემატება მისგან მიღებული ნებისმიერი ***ДНФ***-ის რანგს. სხვადასხვა ***ДНФ***-ების რანგები შეიძლება ერთმანეთისაგან განსხვავდებოდეს. მათ შორის არსებულ უდაბლესი რანგის ***ДНФ***-ს ეწოდება იმ ***ДНФ***-ის მინიმალური ფორმა, რომლისგანაც იყო იგი მიღებული და აღინიშნება როგორც ***МДНФ***. ***ДНФ***-იდან ***МДНФ***-ის მიღების პროცესს ***ლოგიკური ფუნქციის მინიმიზაცია*** ეწოდება. მის შესახებ კონკრეტული მაგალითის ფონზე განვიხილავთ მომდევნო პარაგრაფში.

3.3. ციფრული მოწყობილობების ლოგიკური რეალიზება

ნებისმიერ ლოგიკურ ფუნქციას შეესაბამება გარკვეული ციფრული მოწყობილობა, რომელიც ახდენს ამ ფუნქციის რეალიზებას. სხვა სი-

ტყვებით რომ ვთქვათ, ნებისმიერი ლოგიკური ფუნქცია წარმოადგენს გარკვეული ციფრული მოწყობილობის მათემატიკურ მოდელს. მათემატიკური მოდელის მეშვეობით ციფრული მოწყობილობის აგების ამოცანას ***სინთეზის ამოცანა*** ეწოდება. არსებობს სინთეზის ამოცანის გადაწყვეტის მკაცრად ფორმულირებული ალგორითმი, რომლის დეტალური განხილვა ცდება ჩვენი სახელმძღვანელოს ფარგლებს. მიუხედავად ამისა, კონკრეტული ამოცანის გადაწყვეტის მაგალითზე აღნიშნული ალგორითმის შესახებ ზოგადი წარმოდგენის ქონა, ჩვენი აზრით, აუცილებელია ზოგადად ინჟინრული თვალსაწიერის გასაფართოებლად. აღნიშნულის გამო მოცემულ პარაგრაფში ავაგებთ ზემოთ მოყვანილი მაჟორიტარული ლოგიკური ფუნქციის მარეალიზებელ ციფრულ მოწყობილობას. ამისათვის საჭიროა შემდეგი ოპერაციების შესრულება.

1. სიტყვიერად აღიწეროს სარეალიზებელი ლოგიკური ფუნქცია.

სამ არგუმენტზე დამოკიდებული მაჟორიტარული ლოგიკური ფუნქცია სიტყვიერად ***3.2.*** პარაგრაფში გვაქვს აღწერილი;

2. სიტყვიერი აღწერის საფუძველზე შედგეს სარეალიზებელი ლოგიკური ფუნქციის ჭეშმარიტობის ცხრილი.

სარეალიზებელი ლოგიკური ფუნქციის ჭეშმარიტობის ცხრილი ***3.1,ა*** ნახაზზეა მოცემული;

3. ჭეშმარიტობის ცხრილის მეშვეობით განისაზღვროს სარეალიზაციო ლოგიკური ფუნქციის ДსნФ (ან ***КсნФ***).

ჩვენ მიერ განხილული ***3*** ცვლადზე დამოკიდებული მაჟორიტარული ლოგიკური ფუნქციის ***ДსნФ 3.1,ბ*** ნახაზზეა მოყვანილი.

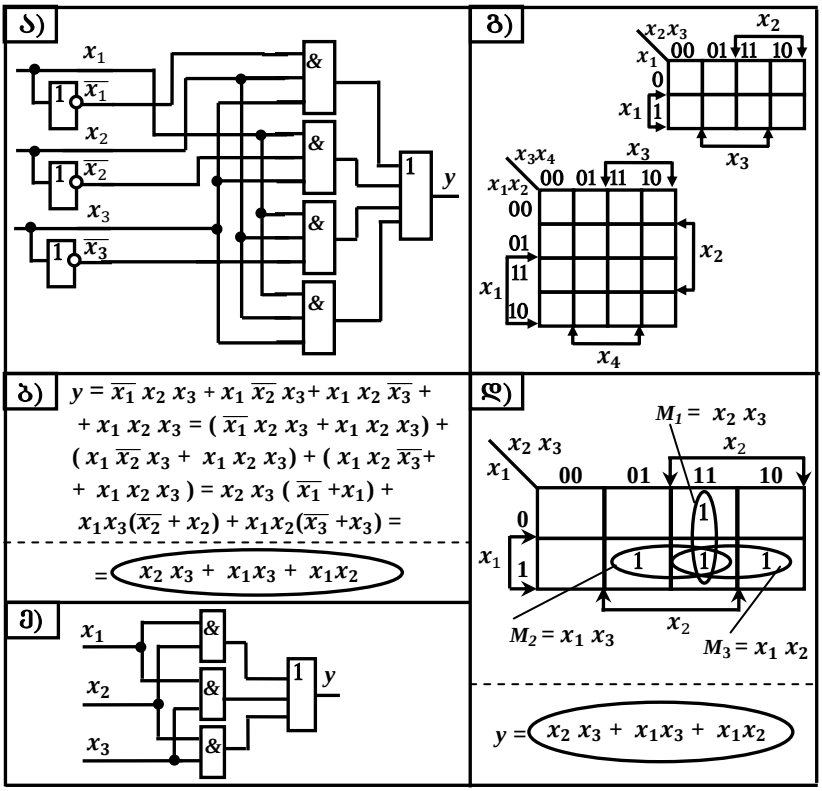
4. არამინიმიზებული ციფრული მოწყობილობის აგებისათვის გადავიდეთ მე-5, ხოლო მინიმიზებული ციფრული მოწყობილობის აგებისათვის – მე-7 პუნქტზე;

5. ДсნФ-ში შემავალი მინიტერმები რეალიზდეს კონიუნქტორებით, რომელთა შესასვლელების რაოდენობა მინიტერმებში არსებული ცვლადების რაოდენობის ტოლია.

განხილული ***ДсნФ*** შეიცავს ***4*** მინიტერმს, რომელთაგანაც თითოეული შეიცავს ***3*** ლოგიკურ ცვლადს (იხ. ნახ.***3.1,ბ***). სამშესასვლელიანი კონიუნქტორების მეშვეობით ოთხივე მინიტერმი რეალიზებულია ***3.2,ა*** ნახაზზე მოყვანილ სქემაზე;

6. კონიუნქტორების მიერ ფორმულირებული ლოგიკური ცვლადები ლოგიკურად შეიკრიბოს დიზიუნქტორის მეშვეობით, რომლის შესასვლელების რაოდენობა ზემოთ აღნიშნული კონიუნქტორების რაოდენობის ტოლია და მოხდეს პუნქტ 9-ზე გადასვლა.

განხილულ მაგალითში გვაქვს ოთხი კონიუნქტორი, რომელთა გამოსასვლელებზე ფორმირებული ლოგიკური ცვლადების ლოგიკურ შეკრებას ახდენს **4** შესასვლელიანი დიზიუნქტორი (ნახ.***3.2,ა***);



ნახ.3.2. *მაჟორიტარული* $y = \overline{x_1}\, x_2\, x_3 + x_1\, \overline{x_2}\, x_3 + x_1\, x_2\, \overline{x_3}$ *ფუნქციის აპარატურული რეალიზება*

7. მოხდეს მოცემული ДНФ-ს მინიმიზება, ე.ი მის საფუძველზე ნაპოვნი იქნეს МДНФ.

არსებობს ***МДНФ***-ის პოვნის ანალიზური და მატრიცული მეთოდები. თავის მხრივ არსებობს ლოგიკური ფუნქციის მინიმიზების უშ-

უალო და ფორმალიზებული მეთოდებიც. ***უშუალო მეთოდის*** გამოყენებისას ლოგიკური ალგებრის კანონების (იხ.***ცხრ.3.1***) გამოყენებით ვახდენთ ***ᲓᲜᲤ***–ის გამარტივებას, როგორც ეს ***3.2,ბ*** ნახაზზეა მოყვანილი. ეს ფორმა შრომატევადია; მისი გამოყენება მოითხოვს მაღალი პროფესიონალიზმსა და დიდ პრაქტიკულ გამოცდილებას.

მინიმიზაციის ამოცანა მარტივდება სხვადასხვა ფორმალიზებული მეთოდების გამოყენებით: შესაძლებელია შევადგინოთ მათი შესაბამისი პროგრამები და ამოცანა გადავწყვიტოთ კომპიუტრის გამოყენებით. ერთ-ერთი ასეთი ფორმალიზებული მეთოდია, მაგალითად, ***ქვაინისა*** და ***მაკ-კლასკის*** მიერ შემოთავაზებული მეთოდი. იგი ჩვენ [*1*]-ში გვაქვს განხილული და მას აქ არ შევეხებით.

ᲓᲜᲤ–ის პოვნის მატრიცული მეთოდის დროს გამოიყენება ე.წ. კარნოს ბარათები. აღნიშნული მეთოდი ფართოდ გამოიყენება მაშინ, როდესაც არგუმენტების რაოდენობა **4...5**-ს არ აჭარბებს. ***კარნოს ბარათი*** ეწოდება ცხრილს, რომლის უჯრედების რაოდენობა n ცვლადზე დამოკიდებული ფუნქციისათვის 2^n-ის ტოლია, ამასთანავე თითოეულ მინიტერმს ბარათზე შეესაბამება საკუთარი უჯრედი. ***3.2,გ*** ნახაზზე მოყვანილია **3**- გა **4**-ცვლადზე დამოკიდებული ლოგიკური ფუნქციების შესაბამისი კარნოს ბარათები.

ᲓᲜᲤ-ს სახით წარმოდგენილი ლოგიკური ფუნქციის კარნოს ბარათის მეშვეობით გამოსახვისათვის კარნოს ბარათის იმ უჯრედებში, რომელთა შესაბამის მინიტერმებს შეიცავს ***ᲓᲜᲤ***, უნდა ჩავწეროთ **1**, ხოლო დანარჩენი უჯრები დავტოვოთ შეუვსებლად. ***3.1,ა*** ნახაზზე მოყვანილი განსახილველი ლოგიკური ფუნქცია კარნოს ბარათის მეშვეობით ***3.2,დ*** ნახაზზეა მოყვანილი.

ამავე ნახაზზე ნაჩვენებია კარნოს ბარათის მეშვეობით ლოგიკური ფუნქციის ***ᲓᲜᲤ***–ის პოვნის პროცესი. მისი განხილვა სცილდება ჩვენი სახელმძღვანელის ფარგლებს, მაგრამ იგი დეტალურად გვაქვს განხილული [*1.2*]-ში, რომლებსაც შეუძლია გაეცნოს დაინტერესებულ პირს.

8. ᲓᲜᲤ რეალიზდეს ისე, როგორც მე-5 პუნქტში მოხდა ᲓᲜᲤ-ის რეალიზება, ოღონდ მინიტერმების ნაცვლად ავიღოთ ***ᲓᲜᲤ***–ს ელემენტალური კონიუნქციები (ნახ.***3.2,ე***);

9. ალგორითმის დასასრული.

3.2,ა,ე ნახაზებზე მოყვანილი სქემებით ერთიდაიგივე ლოგიკური ფუნქციაა რეალიზებული, ოღონდ ***3.2,ა*** ნახაზზე მოხდენილია ამ ფუნქციის ***ДНФ***-ის, ხოლო ***3.2,ე*** ნახაზზე - ***ДСНФ***–ის რეალიზება. როგორც ნახაზიდან ჩანს, ***ДСНФ***–ის გამოყენებისას მნიშვნელოვნად მარტივდება სქემის სტრუქტურა.

დასასრულს, აღნიშნავთ, რომ ***ლოგიკური ფუნქციის მინიმიზაციის პრობლემის აქტუალობა*** დამოკიდებულია შესაბამისი ციფრული მოწყობილობის ასაგებად გამოყენებულ საელემენტო ბაზაზე. კერძოდ, იგი მეტად აქტუალურია საელემენტო ბაზად ცალკეული ელექტრონული ხელსაწყოების (ტრანზისტორების, დიოდების, ტირისტორების და ა.შ) ან მცირე ინტეგრალური სქემების გამოყენების დროს და ნაკლებად აქტუალურია ამ მიზნით საშუალო და დიდი ინტეგრალური სქემების გამოყენებისას.

ციფრული მოწყობილობის რეალიზებისათვის, უპირველეს ყოვლისა, ფორმირებული უნდა იყოს დავალება, რომელშიც განსაზღვრული იქნება აღნიშნული მოწყობილობის ფუნქციები.

დავალების ფორმირე-ბისათვის შეიძლება გამოყენებული იქნეს ბუნებრივი სალაპარაკო ენა, მაგრამ აღნიშნული ენის მრავალმნიშვნელიანობა და სიჭარბე აძნელებს დავალების კომპაქტურად და ცალსახად ფორმირებას. აღნიშ-ნული პრობლემის დასაძლევად გამოიყენება სხვადასხვა მიდგომა. ზემოთ დავალება ფორმირებული იყო ჭეშმარიტობის ცხრილის (იხ.***ნახ.3.1,ა***) მეშვეობით. უნივერსალურ მიდგომად უნდა ჩაითვალოს დავალების ფორმირება ***რეგულარული გამოსახულების*** სახით [5,6,7,19,25]. რეგულარული გამოსახულების მეშვეობით ლოგიკური მოწყობილობის რეალიზებისათვის დამახასიათებელი პრობლემა ფორმირებულია ***IV ᲓᲐᲜᲐᲠᲗᲨᲘ.*** აღნიშნული პრობლემა დასაძლევად ჩვენ მიერ დამუშავებული იქნა ციფრული მოწყობილობების აბსტრაქტული სინთეზის ორიგინალური ალგორითმი [3]; იგი რამდენადმე ცდება მოცემულ სახელმძღვანელოში განსახილველ საკითხებს, მაგრამ მისი გამოყენება მნიშვნელოვნად აადვილებს მიკროპროცესორული მოწყობილობების კონსტრუქტორის საქმიანობას. აღნიშნულის გამო მიზანშეწონილად ჩავთვალეთ იგი წარმოგვედგინა ***IV ᲓᲐᲜᲐᲠᲗᲨᲘ.***

IV თავი
მიკროპროცესორული სისტემის ტიპური ციფრული მოწყობილობები

მიკროპროცესორული სისტემის სტრუქტურა შედგება ცალკეული ელემენტების, კვანძების, ბლოკებისა და მოწყობილობებისაგან. ***ელემენტები*** ამუშავებს ინფორმაციის ბიტების შესაბამის ცალკეულ ელექტრულ სიგნალებს. ***კვანძები*** ამუშავებს სიგნალთა ჯგუფების მეშვეობით წარმოქმნილ ცალკეული საინფორმაციო სიტყვებს, ხოლო ***ბლოკი*** - ამ სიტყვების ცალკეულ მიმდევრობებს; სიტყვების ურთიერთდაკავშირებულ მიმდევრობათა რეალიზებას ახდენს ***მოწყობილობა.***

ერთმანეთისაგან განასხვავებენ ანალოგურ და ციფრულ მოწყობილობებს, რომელთაგანაც პირველი განკუთვნილია ანალოგური (უწყვეტი), ხოლო მეორე – ციფრული (წყვეტილი ანუ დისკრეტული) სიგნალების დასამუშავებლად. ჩვენ მხოლოდ ციფრულ მოწყობილობებს განვიხილავთ.

ერთმანეთისაგან განასხვავებენ კომბინაციურ (ერთტაქტურ) და მიმდევრობით (მრავალტაქტურ) ციფრულ მოწყობილობებს. ამ უკანასკნელებს ხშირად ***სასრულ ავტომატებსაც*** უწოდებენ.

კომბინაციური ეწოდება ციფრულ მოწყობილობას, რომლის გამოსასვლელ სიგნალთა მნიშვნელობებს ცალსახად განსაზღვრავს (დროის მოცემულ ან წინა მომენტში) შესასვლელი სიგნალების მნიშვნელობები. ***მიმდევრობითი ციფრული მოწყობილობის*** გამოსასვლელი სიგნალების მნიშვნელობები დამოკიდებულია არა მარტო შესასვლელი საგნალების მნიშვნელობაზე, არამედ თავად მოწყობილობის შინაგან მდგომარეობაზეც. აღნიშნულის გამო კომბინაციურ მოწყობილობებს ხშირად ***მეხსიერების არმქონე მოწყობილობებს***, ხოლო მიმდევრობით მოწყობილობებს – ***მეხსიერებიან მოწყობილობებსაც*** უწოდებენ.

მოცემული თავის მიზანია კომპიუტერული სისტემის სტრუქტურის შემადგენელი ძირითადი ერთეულების მოკლედ განხილვა.

4.1. კომბინაციური ლოგიკურ მოწყობათა ტიპური ფუნქციური კვანძები

<u>მულტიპლექსორები და დემულტიპლექსორები</u>. ***მულტიპლექსორი*** ეწოდება ინფორმაციის რამდენიმე წყაროდან ერთ გამოსასვლელ არხში მონაცემების მართულად გადამცემ კომბინაციურ ლოგიკურ მოწყობილობას. მას აქვს ერთი გამოსასვლელი და ორი სახის, კერძოდ, საინფორმაციო და სამისამართო შესასვლელები. ***საინფორმაციო შესასვლელებზე*** მიეწოდება კონკრეტული ინფორმაციის გამომსახველი ორობითი კოდი. ***სამისამართო შესასვლელებზე*** მიწოდებული კოდით განისაზღვრება დროის მოცემულ მომენტში, თუ რომელი საინფორმაციო შესასვლელი უნდა მიუერთდეს გამოსასვლელს. ვინაიდან ***n***–თანრიგიან ორობით კოდს 2^n–ის რაოდენობის მნიშვნელობის მიღება შეუძლია, ამიტომ ***n*** რაოდენობის ***სამისამართო შესასვლელის*** მქონე მულტიპლექსორს 2^n რაოდენობის ***საინფორმაციო შესასვლელი*** უნდა ჰქონდეს.

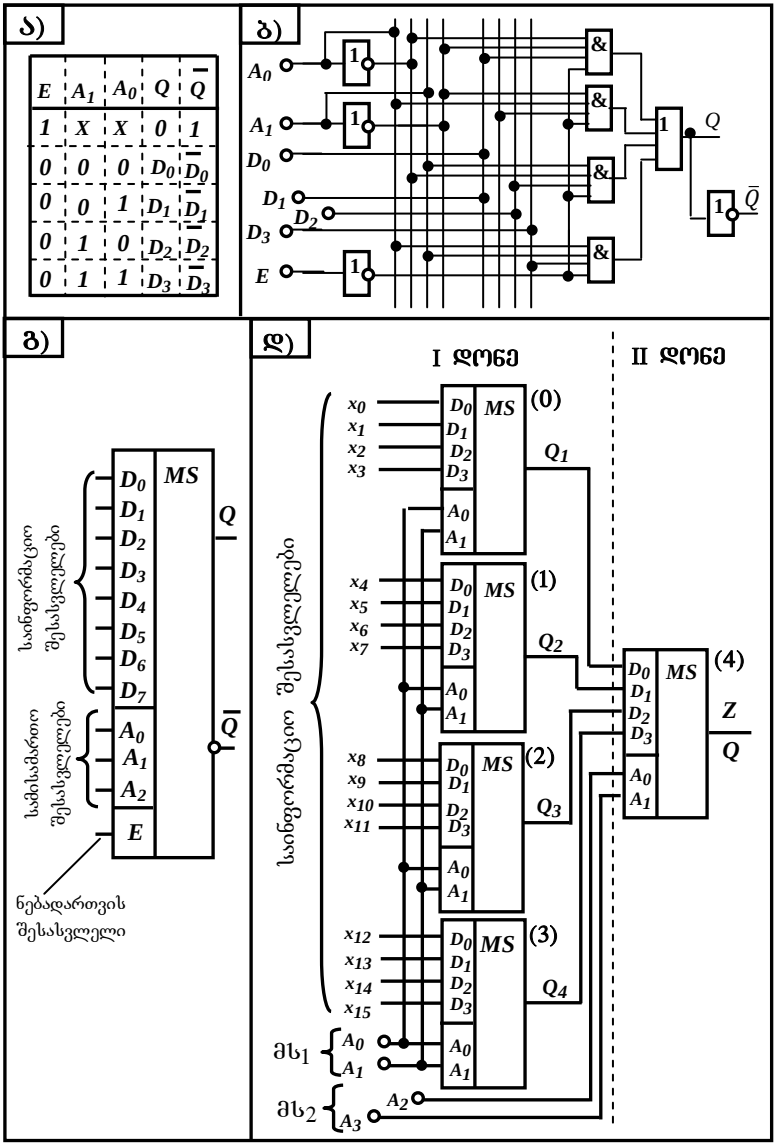
ორი სამისამართო შესასვლელის მქონე მულტიპლექსორის მუშაობის ამსახავი ჭეშმარიტობის ცხრილი ***4.1,ა*** ნახაზზეა მოცემული. ამ ცხრილიდან ჩანს, რომ მულტიპლექსორს აქვს დამატებითი ინვერსიული $\overline{Q}$ გამოსასვლელი და მუშაობის ნებადართვის ***E*** შესასვლე-ლი.

მუშაობის ნებადართვის ***E*** შესასავლელზე აქტიური ლოგიკური (***E=1***) სიგნალის მიწოდებისას მულტიპლექსორის გამოსასვლელი სიგნალი მუდმივია და მისი მნიშვნელობა არაა დამოკიდებული შესასვლელ სიგნალებზე.

მულტიპლექსორის მუშაობის აღმწერ ლოგიკურ ფუნქციას აქვს სახე: $Q = D_0\overline{A}_1\overline{A}_0\overline{E} + D_1\overline{A}_1A_0\overline{E} + D_2A_1\overline{A}_0\overline{E}$.

მოცემული ლოგიკური ფუნქციის მარეალიზებელი მულტიპლექსორის ლოგიკური სქემა ***4.2,ბ*** ნახაზზეა ნაჩვენები.

წარმოება ინტეგრალური მიკროსქემის სახით უშვებს ტიპურ მულტიპლექსორებს, რომლებსაც აქვს ***8*** საინფორმაციო და ***3*** სამისა-



ნახ. 4.1. მულტიპლექსორის: *ჭეშმარიტობის ცხრილი (ა); ლოგიკური სქემა (ბ); პირობითი გამოსახულება (გ); ხის (იერარქიის) ლოგიკური სქემა (დ).*

მართო შესასვლელი. მისი პირობითი გამოსახულება ***4.1,ვ*** ნახაზზეა მოყვანილი. იგი ფუნქციონირებს შემდეგნაირად. სამისამართო შესასვლელებზე მიწოდებული ორობითი $A_2A_1A_0$ მისამართით განისაზღვრება ის საინფორმაციო $D_i \in \{D_0, D_1, \dots, D_7\}$ შესასვლელი, რომელიც მიუერთდება მულტიპლექსორის Q გამოსასვლელს.

8-ზე მეტი საინფორმაციო შესასვლელების მქონე მულტიპლექსორის საჭიროებისას საჭიროა ტიპური ***8*** საინფორმაციო შესასვლელის მქონე მულტიპლექსორებისაგან ავაგოთ ე.წ. ***მულტიპლექსორული ხე.*** სიმარტივისათვის აღნიშნული ხის ასაგებად გამოვიყენოთ პირობითი ***4*** საინფორმაციო და ***2*** სამისამართო შესასვლელის მქონე მულტიპლექსორები. მისი სტრუქტურული სქემა ***4.1,დ*** ნახაზზეა მოყვანილი. როგორც ამ ნახაზიდან ჩანს, აღნიშნულ ხის სტრუქტურაში მულტიპლექსორები ორ დონეზეა განთავსებული: ***I*** დონეზე განთავსებულია (***0***), (***1***), (***2***) და (***3***) ნომრის, ხოლო ***II*** დონეზე – (***4***) ნომრის მქონე ***4*** საინფორმა-ციო შესასვლელის მქონე მულტიპლექსორები. მულტიპლესორული ხის სამისამართო ***მს1*** შესასვლელით ამოირჩევა ***I*** დონის მულტიპლექსორი, რომელთანაც მიერთებულია ამოსარჩევი საინფორმაციო არხი; ეს უკანასკნელი კი სამისამართო ***მს2*** შესასვლელის საშუალებით ამოირჩევა. სიცხადისათვის განვიხილოთ შემდეგი მაგალითი:

დავუშვათ, რომ საჭიროა მულტიპლექსორული ხის ***Z*** გამოსასვლელს მიუერთდეს ***(2)*** ნომრის მქონე მულტიპლექსორის საინფორმაციო D_3 არხთან დაკავშირებული საინფორმაციო x_{11} არხი. ამისათვის საჭიროა სამისამართო ***მს1*** შესასვლელზე მივაწოდოთ ორობითი ***10*** მისამართი, რისი მეშვეობითაც ამოირჩევა ***(2)*** ნომრის მქონე მულტიპლექსორი. ამის შემდეგ ***მს2*** საინფორმაციო შესასვლელზე უნდა მივაწოდოთ ორობითი ***11*** მისამართი, რომლითაც ამოირჩევა ***(2)*** ნომრის მქონე მულტიპლექსორის საინფორმაციო D_3 შესასვლელი და იგი მიუერთდება ზემოთ აღნიშნულ ***Z*** გამოსასვლელს. ვინაიდან ეს საინფორმაციო D_3 შესასვლელი საინფორმაციო x_{11} არხთანაა დაკავშირებული, დასმული ამოცანა წარმატებით იქნება გადაწყვეტილი. საბოლოოდ შეიძლება ვთქვათ, რომ x_{11} არხის ჯამური ორობითი მისამართი განხილული შემთხვევის დროს ***1011***-ის ტოლია.

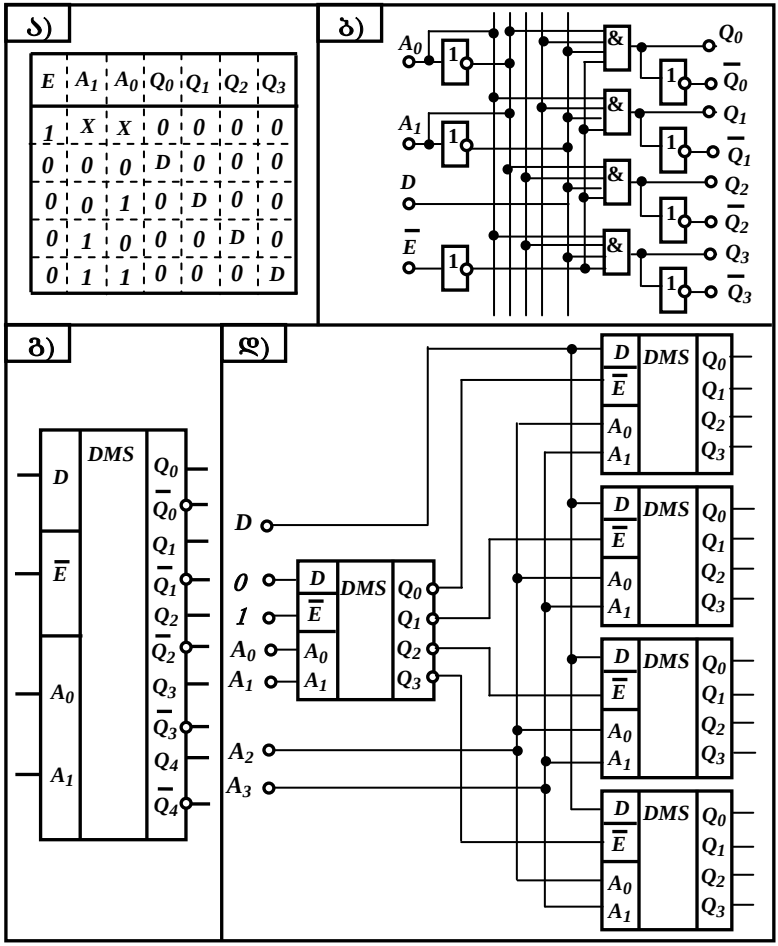
რამდენიმე წყაროდან მოსული ინფორმაცია ერთი არხის მეშვეობით დროში დანაწევრებულად გადაცემისათვის, მულტიპლექსორების გარდა საჭიროა გვქონდეს შებრუნებული დანიშნულების მოწყობილობები, რო-

მლებიც ერთი არხიდან მიღებულ ინფორმაციებს რამდენიმე მიმღებს გაუნაწილებს. ამ ამოცანას წყვეტს ე. წ. დემულტიპლექსორი.

დემულტიპლექსორი ეწოდება ინფორმაციის ერთი წყაროდან რამდენიმე გამოსასვლელ არხში მონაცემების მართულად გადამცემ ლოგიკურ მოწყობილობას.

ა)

E	A_1	A_0	Q_0	Q_1	Q_2	Q_3
1	X	X	0	0	0	0
0	0	0	D	0	0	0
0	0	1	0	D	0	0
0	1	0	0	0	D	0
0	1	1	0	0	0	D



ნახ. 4.2. *დემულტიპლექსორის: ჭეშმარიტობის ცხრილი (**ა**); ლოგიკური სქემა (**ბ**); პირობითი გამოსახულება (**გ**); ხის (იერარქიის) ლოგიკური სქემა (**დ**).*

ზოგადად დემულტიპლექსორს აქვს ერთი საინფორმაციო და 2^n გამოსასვლელიანი n რაოდენობის სამისამართო შესასვლელი. მისი მუშაობის აღმწერი ჭეშმარიტობის ცხრილი ***4.2,ა*** ნახაზზეა მოყვანილი. მასში E წარმოადგენს მუშაობის ნებადართვის სიგნალს, A_1,A_0 - სამისამართო შესასვლელებს. ხოლო $Q_0 \ldots Q_3$ - გამოსასვლელებს. მოცემულ ცხრილს შეესაბამება ლოგიკური ფუნქციების შემდეგი სისტემა:

$$\left.\begin{aligned} Q_0 &= D\bar{A}_1\bar{A}_0\bar{E} = \bar{D}\downarrow A_1\downarrow A_0\downarrow E; \\ Q_1 &= D\bar{A}_1A_0\bar{E} = \bar{D}\downarrow A_1\downarrow \bar{A}_0\downarrow E; \\ Q_2 &= DA_1\bar{A}_0\bar{E} = \bar{D}\downarrow \bar{A}_1\downarrow A_0\downarrow E; \\ Q_3 &= DA_1A_0\bar{E} = \bar{D}\downarrow \bar{A}_1\downarrow \bar{A}_0\downarrow E; \end{aligned}\right\}$$

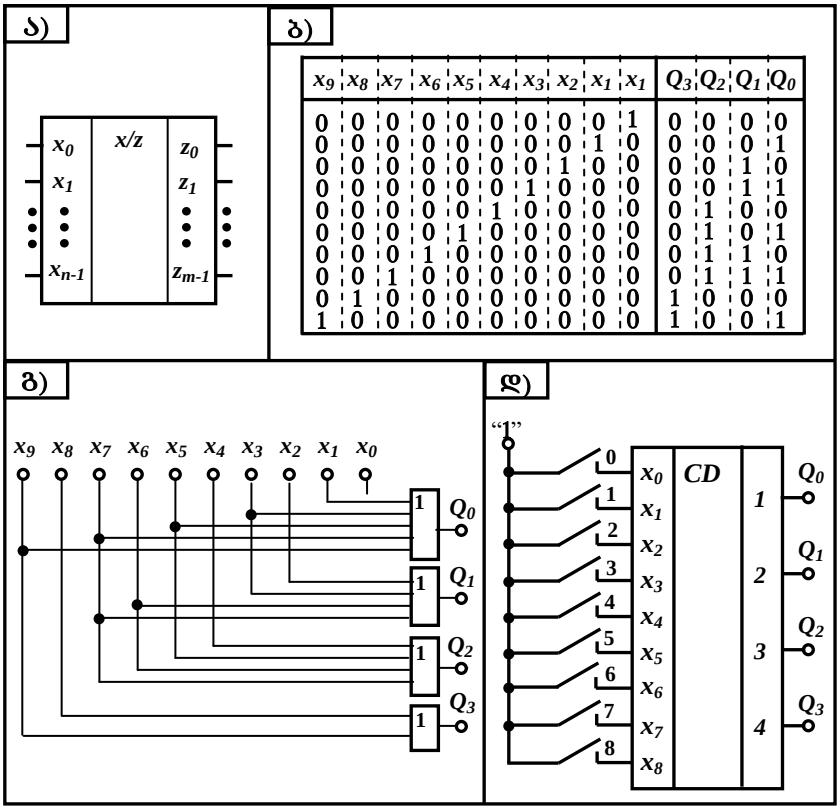
4.2,ბ ნახაზზე მოყვანილია მოცემული სისტემის მარეალიზებელი დემულტიპლექსორორის ლოგიკური სქემა, ხოლო ***4.2,გ*** ნახაზზე – მისი პირობითი გამოსახულება. სქემის საფუძველზე გამოსასვლელი გამომყვანების რაოდენობის გაზრდის საჭიროებისას აუცილებელია ავაგოთ ***დემულტიპლექსირული ხე***. იგი ზუსტად მულტიპლექსირული ხის მსგავსად აიგება (ნახ. 4***.2,დ***). ამ დროს პირველი დონის დემულტიპლექსორი იმართება სამისამართო სიტყვის უმცროსი თანრიგებით, ხოლო მეორე დონის დემულტიპლექსორი – აღნიშნული სიტყვის უფროსი თანრიგებით.

დემულტიპლექსირული ხის აგებისას აუცილებელია გავითვალისწინოთ მუშაობის ნებადართვისათვის (სტრობირებისათვის) საჭირო შესასვლელი.

<u>კოდების გარდამქმნელი.</u> ციფრულ ტექნიკაში ინფორმაცია სხვადასხვანაირად კოდირდება. კომპიუტერში, მაგალითად, ოპერაციების შესასრულებლად ორობითი კოდის რამდენიმე სახესხვაობა (პირდაპირი, შებრუნებული, დამატებითი, ორობით-ათობითი და ა.შ. კოდები) გამოიყენება. კავშირგაბმულობის ხაზებით ინფორმაციის გადაცემისას უნდა გამოვიყენოთ სხვა სახის კოდები, რომლებიც, მაგალითად, ამცირებს შეცდომების წარმოშობის ალბათობას, ხოლო მათი წარმოშობის შემთხვევაში ამ შეცდომების გასწორების შესაძლობას იძლევა. მსგავსი სახის კოდებს მიეკუთვნება ლუწობის ან კენტობის შემმოწმებელი კოდები, ჰემინგის კოდები, მუდმივწონიანი (კერძოდ, **5C2**) კოდები.

ზემოთ აღნიშნულიდან გამომდინარე, ყოველთვის წამოიჭრება გარვეული სახის კოდის სხვა სახის კოდად გარდაქმნის საჭიროება.

აპარატურულ დონეზე ამ ამოცანას წყვეტს კოდების გარდამქმნელად წოდებული კომბინაციური მოწყობილობები.



ნახ. 4.3. კოდების გარდამქმნელის *პირობითი გამოსახულება* ***(ა);*** *შიფრატორის ჭეშმარიტობის ცხრილი* ***(ბ);*** *ათობითი რიცხვების* ***შიფრატორის*** *ლოგიკური სქემა* ***(გ);*** *კლავიატურიდან ინფორმაციის შეტანის მოწყობილობა* ***(დ).***

კოდის გარდამქმნელი ეწოდება კოდირების სახეობის შემცვლელ კომბინაციურ მოწყობილობას. მისი მუშაობაც აღიწერება ჭეშმარიტობის ცხრილით, რომელიც ერთმანეთს შეუსაბამებს ამ მოწყობილობის შესასვლელებზე მიწოდებულ და გამოსასვლელებზე წარმოქმნილ კოდურ სიტყვებს. ზოგადად შეიძლება ერთმანეთს არ ემთხვეოდეს აღნიშნულ სიტყვათა სიგრძეები (სიტყვების წარმომქმნე-

ლი ბიტების რაოდენობები). მთავარია ცხრილი მეშვეობით დამყარდეს აღნიშნული სიტყვების ურთიერთცალსახა შესაბამისობა. კოდების გარდამქმნელის პირობითი გამოსახულება, რომლითაც იგი პრინციპულ სქემებზე აღინიშნება, ***4.3,ა*** ნახაზზეა მოყვანილი. ინტეგრალური სქემების სახით გამოშვებული კოდების გარდამქმნელის მაგალითს წარმოადგენს ორობითი ინფორმაციის ორობით-ათობით ინფორმაციად გარდამქმნელი. კოდის გარდამქმნელთა კერძო შემთხვევებს წარმოადგენს შიფრატორები და დეშიფრატორები.

<u>შიფრატორები და დეშიფრატორები.</u> ***შიფრატორი*** ანუ ***კოდერი*** ეწოდება მოწყობილობას, რომელიც ათობით რიცხვებს გარდაქმნის ორობით რიცხვებად. შესასვლელებს მიმდევრობით მიეწოდება ათობითი რიცხვების გამომსახველ სიგნალთა ერთობლიობები. თითოეული ეს ერთობლიობა გარდაიქმნება შესაბამისი ორობითი რიცხვის გამომსახავ სიგნალთა ერთობლიობად. ნათქვამის თანახმად, შიფრატორს თუ აქვს n რაოდენობის გამოსასვლელი, მაშინ მისი შესასვლელების რაოდენობა 2^n–ზე ნაკლები არ უნდა იყოს. 2^n გამოსასვლელისა და n შესასვლელის მქონე შიფრატორს ***სრული შიფრატორი*** ეწოდება, ხოლო 2^n-ზე ნაკლები შესასვლელებიან შიფრატორს - ***არასრული შიფრატორი***.

შიფრატორის მუშაობა განვიხილოთ ***0***-დან **9**-მდე ათობითი რიცხვის ორობით-ათობით კოდებად გარდამქმნელის მაგალითზე. ამ შემთხვევის შესაბამისი ჭეშმარიტობის ცხრილი ***4.3,ბ*** ნახაზზეა მოცემული, რომლიდანაც მიიღება შემდეგი ლოგიკური ფუნქციათა შემდეგი სისტემა:

$$\left.\begin{aligned} Q_3 &= x_8 + x_9; \\ Q_2 &= x_4 + x_5 + x_6 + x_7; \\ Q_3 &= x_2 + x_3 + x_6 + x_7; \\ Q_3 &= x_1 + x_3 + x_5 + x_7 + x_9. \end{aligned}\right\}$$

მოცემული სისტემა განსაზღვრავს შიფრატორის მუშაობას. მის მიხედვით სინთეზირებული შიფრატორი ***4.3,გ*** ნახაზზეა მოყვანილი.

ნახაზიდან ჩანს, რომ, განხილული ტიპის შიფრატორის x_0 შესასვლელზე სიგნალი არ მიეწოდება. რომელიმე შესასვლელზე სიგნალის არარსებობა ამ შესასვლელზე სიგნალ ***0***-ის არსებობად აღიქმება.

შიფრატორი ძირითადად გამოიყენება კლავიატურიდან ციფრულ სისტემაში პირველადი ინფორმაციის შესატანად. ნებისმიერ კლავიშ-

ზე თითის დაჭერისას შიფრატორის შესაბამის შესასვლელს მიეწოდება ლოგიკური «1» სიგნალი, რომელიც შიფრატორის გამოსასვლელზე ორობით-ათობით კოდად გარდაიქმნება. ინფორმაციის შეტანის ერთ-ერეთი ვარიანტი ***4.3,დ*** ნახაზზეა მოყვანილი.

დეშიფრატორი ანუ ***დეკოდერი*** ეწოდება თვლის ორობითი სისტემიდან რიცხვების ათობით სისტემაში გადამყვან კომბინაციურ ლოგიკურ მოწყობილობას. განსაზღვრების თანახმად დეშიფრატორი კოდების გარდამქმნელთა კლასს მიეკუთვნება. ამ შემთხვევაშიც თითოეულ შესასვლელ ორობით რიცხვს შეუთანადდება მოწყობილობის გარკვეულ გამოსასვლელზე ფორმირებული სიგნალი. ამგვარად, დეშიფრატორი ასრულებს შიფრატორის მიერ შესრულებული ოპერაციის შებრუნებულ ოპერაციას. n რაოდენობის შესასვლელებისა და m რაოდენობასა გამოსასვლელების მქონე დეშიფრატორს ეწოდება ***სრული დეშიფრატორი,*** თუ სრულდება ტოლობა $m = 2^n$; ***არასრული დეშიფრატორის*** დროს $m < 2^n$.

დეშიფრატორის მუშაობა აღიწერება შიფრატორის ჭეშმარიტობის ცხრილის (იხ. ***ნახ. 4.3,ბ***) ანალოგური ჭეშმარიტობის ცხრილით, ოღონდ მასში ადგილებს ერთმანეთს უცვლის შესასვლელი და გამოსასვლელები. ამ ცხრილის შესაბამისად გამოსასვლელი სიგნალი "***1***"-ის ტოლი ცვლადების მხოლოდ ერთადერთ ნაკრებზე ხდება, ამიტომ დეშიფრატორის მუშაობას აღწერს ლოგიკური ფუნქციების შემდეგი სისტემა:

$$\left.\begin{aligned} x_0 &= \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1}\,\overline{Q_0};\\ x_1 &= \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1}\,Q_0;\\ x_2 &= \overline{Q_3}\,\overline{Q_2}\,Q_1\,\overline{Q_0};\\ &\vdots \end{aligned}\right\}$$

სადაც Q_i არის მოწყობილობის i–ურ გამოსასვლელზე არსებული ლოგიკური ცვლადის მნიშვნელობა.

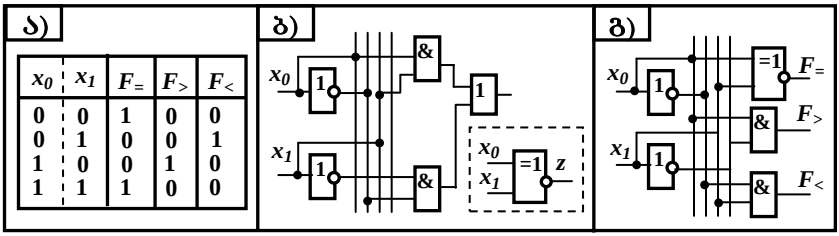
<u>ციფრული კომპარატორი</u> ეწოდება ორობითი კოდების სახით წარმოდგენილი რიცხვების ერთმანეთთან შემდარებელ კომბინაციურ ლოგიკურ მოწყობილობას.

კომპარატორის შესასვლელების რაოდენობას განსაზღვრავს შესადარებელი რიცხვების თანრიგიანობა. კომპარატორის გამოსასვლელზე ჩვეულებრივ ფორმირდება შემდეგი სამი სიგნალი:

- $F_=$ - «შესადარებელი კოდების რიცხვითი ეკვივალენტები ერთმანეთის ტოლია»;

- $F_>$ - «პირველი კოდის რიცხვითი ეკვივალენტი აღემატება მეორე კოდის რიცხვით ეკვივალენტს»;

- $F_<$ - «პირველი კოდის რიცხვითი ეკვივალენტი ნაკლებია მეორე კოდის რიცხვით ეკვივალენტზე».



ნახ. 4.4. *ერთთანრიგიანი* ***კომპარატორის*** *ჭეშმარიტობის ცხრილი* ***(ა);*** *გამომრიცხავი* ***ანა-არა*** *ელემენტის ლოგიკური სქემა და პირობითი გასახულება* ***(ბ);*** *კომპარატორის ლოგიკური სქემა* ***(გ).***

ორი ერთთანრიგიანი კოდების შედარების დროს კომპარატორის მუშაობას აღწერს ***4.4,ა*** ნახაზზე მოყვანილი ჭეშმარიტობის ცხრილი.

ჭეშმარიტობის ცხრილის ანალიზი გვიჩვენებს, რომ შესასვლელი სიგნალების ნებისმიერი კომბინაციის დროს კომპარატორის გამოსასვლელზე შეიძლება ფორმირდეს მხოლოდ ერთი აქტიური (ერთეულოვანი) სიგნალი. ამიტომ შესასვლელი კოდების ნებისმიერი თანრიგიანობისას საკმარისია შესასვლელი სიგნალების გამოყენებით წარმოიქმნას ნებისმიერი ორი გამოსასვლელი სიგნალი. მესამე სიგნალი ყოველთვის შეიძლება მივიღოთ ორი ცნობილი სიგნალის მეშვეობით.

ჭეშმარიტობის ცხრილიდან მიიღება ლოგიკური ფუნქციების შემდეგი სისტემა:

$$\left.\begin{aligned} F_= &= \overline{x_1}\,\overline{x_0} + x_1 x_0 = \overline{F_<}\;\overline{F_>}; \\ F_< &= \overline{x_1} x_0 = \overline{F_=}\;\overline{F_>}; \\ F_> &= x_1 \overline{x_0} = \overline{F_=}\;\overline{F_<}. \end{aligned}\right\}$$

აპარატურული დანახარჯების თვალსაზრისით, მოყვანილი გამოსახულებების ანალიზს გვიჩვენებს, რომ შესასვლელი ცვლადების

გამოყენებისას მოსახერხებელი იქნება განვსაზღვროთ $F_>$ და $F_<$ სიგნალთა მნიშვნელობები, ხოლო $F_=$ სიგნალი განვიხილოთ როგორც მათი ფუნქცია; ვინაიდან $F_=$ სიგნალის განმსაზღვრელ ფუნქციას ციფრულ ტექნიკაში უფრო მეტი დამოუკიდებელი მნიშვნელობა აქვს, ამიტომ საჭიროა იგი უფრო დაწვრილებით განვიხილოთ. $F_=$-ის გამოსახულებას ეწოდება გამომრიცხავი ***ან-არა*** ფუნქცია, ანუ ორის მოდულით შეკრების ინვერსია. ***და***, ***ან***, ***არა*** ელემენტების გამოყენებით ამ ოპერაციის რეალიზების მაგალითი და მისი პირობითი გამოსახულება ***4.4,ბ*** ნახაზზეა მოყვანილი. ***4.4,გ*** ნახაზზე მოცემულია ***4.4,ა*** ნახაზზე მოყვანილი ჭეშმარიტობის ცხრილის მიხედვით აგებული ლოგიკური მოწყობილობის სტრუქტურული სქემა.

<u>არითმეტიკულ-ლოგიკური მოწყობილობა (ალმ)</u> ეწოდება მიკროპროცესორული სისტემის კვანძს, რომელიც ასრულებს ინფორმაციის დამუშავებისათვის საჭირო არითმეტიკულ და ლოგიკურ ოპერაციებს. აღნიშნული ოპერაციების უმრავლესობა სრულდება აპარატურულად, მცირე ნაწილი კი – პროგრამულად.

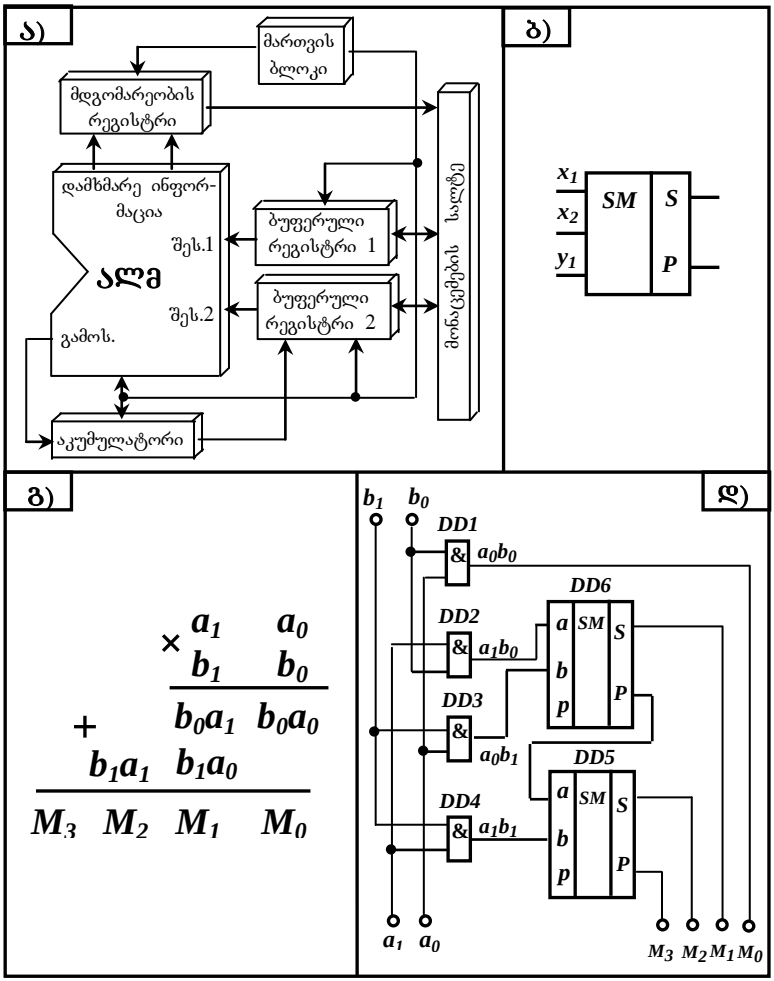
ალმ-ის მიერ რეალიზებულ ოპერაციებს შორის ყველაზე მნიშვნელოვანია არითმეტიკული შეკრებისა და გამრავლების ოპერაციები. მათ შესრულებაზე დახარჯული დროის ხანგრძლივობა მიკროპროცესორული სისტემის ფუნქციონირების შესაფასებელ ერთ-ერთ ძირითად პარამეტრად ითვლება.

შესასვლელ ცვლადებზე არითმეტიკული და ლოგიკური ოპერაციების შესრულებისათვის საჭიროა ***ალმ***-ში მათი შეტანა; ამიტომ ***ალმ***-ის სტრუქტურას ემატება როგორც საწყისი მონაცემების, ისე მათზე ოპერაციების ჩატარებით მიღებული შუალედური შედეგების შემნახავი დამხმარე მოწყობილობები. ასეთი მოწყობილობების ფუნქციებს ასრულებს ***დამატებითი რეგისტრები.***

4.5,ა ნახაზზე მოყვანილია დამატებითი რეგისტრებიანი ***ალმ***–ის სქემის ერთ-ერთი ვარიანტი. არსებითად ეს სქემა მიკროპროცესორის გამარტივებული სქემაა.

ალმ-ში მონაცემების შესატანად, როგორც წესი, არსებობს პორტების (გამომყვანების) ორი, ხოლო მისგან მონაცემების გამოსატანად – ერთი ჯგუფი. ***4.5,ა*** ნახაზზე შესასვლელი პორტების ჯგუფები აღნიშნულია როგორც **«შეს.1»** და **«შეს.2 »,** ხოლო გამოსასვლელი პორტების ჯგუფი – როგორც **«გამოს»**. გარდა ამისა, **დამხმარე ინ-**

ფორმაციით მომარაგებისათვის ***ალგ***-ში გათვალისწინებულია პორტების კიდევ ერთი ჯგუფიც (იხ. ნახ. ***4.5,ა***).



ნახ. 4.5. *გარე რეგისტრებთან* ***ალგ***-*ის მიერთება* ***(ა);*** *სუმატორის (ამჯამავის) პირობითი აღნიშვნა* ***(ბ);*** *ორთანრიგიანი რიცხვების გამრავლების სქემა* ***(გ);*** *მატრიცული მამრავებელის სტრუქტურული სქემა* ***(დ).***

შესასვლელი პორტების ორივე ჯგუფი აღჭურილია მონაცემების დროებითად შესანახი საკუთარი ***ბუფერული რეგისტრით***. თითოეულ ბუფერულ რეგისტრში შეინახება ინფორმაციის ერთი სიტყვა, რომლის თანრიგების რაოდენობა მოწყობილობის კონკრეტულ ტიპზეა დამოკიდებული. ერთ-ერთი შესასვლელი პორტი მონაცემებს უშუალოდ მონაცემების სალტიდან იღებს, ხოლო მეორე პორტს მონაცემები შეუძლია მიიღოს როგორც უშუალოდ მონაცემების სალტიდან, ისე ***აკუმულატორად*** წოდებულ სპეციალიზებული რეგისტრიდანაც. ამ უკანასკნელი რეგისტრის შესასვლელთან გამოსასვლელი პორტია მიერთებული (იხ. ნახ. ***4.5,ა***).

არაერთ შემთხვევაში აკუმულატორს გააჩნია მონაცემების სალტესთან მისაერთებელი მეორე შესასვლელიც. ამიტომ ზოგადად აკუმულატორში შეიძლება ინახებოდეს როგორც უკვე ჩატარებული ოპერაციის შედეგად მიღებული, ასევე მონაცემების სალტით გადაცემული მონაცემებიც. ***ალმ***-ის მუშაობის შესახებ ***დამხმარე ინფორმაციის*** მიღებისათვის განკუთვნილი პორტების (გამომყვანების) ჯგუფი მიერთებულია «მდგომარეობის რეგისტრად» წოდებულ სპეციალურ რეგისტრთან. ამ რეგისტრს ზოგჯერ «ინდიკატორსაც» უწოდებენ. მის თანრიგებში ინახება სამომსახურეო ინფორმაცია უკანასკნელად შესრულებული ოპერაციის შედეგის შესახებ. მაგალითად, მასში შეიძლება ინახებობოდეს ინფორმაცია იმის შესახებ, რომ აკუმულატორი ჩამოყრილია, ბოლო ოპერაციის შესრულებისას მიღებული იქნა უარყოფითი შედეგი და ა.შ.

ოპერაციის ტიპზე დამოკიდებულებით ***ალმ*** შეიძლება ამუშავებდეს მონაცემების ერთ ან ორ სიტყვას; პირველ შემთხვევაში, იგი გამოიყენებს ერთ, ხოლო მეორე შემთხვევაში – ორივე შესასვლელ პორტს. მაგალითად, არითმეტიკული შეკრების ოპერაციის შესრულებისას იგი იყენებს ორივე, ხოლო შებრუნებული კოდის მისაღებად – მხოლოდ ერთ პორტს. ოპერაციის შედეგი ყოველთვის ჩაიწერება აკუმულატორში.

სხვადასხვა კლასის მოწყობილობებისათვის ***ალმ***-ის მიერ რეალიზებული ოპერაციათა კონკრეტული ჩამონათვალი შეიძლება საკმაოდ მრავალფეროვანი იყოს. ამ მრავალფეროვნებაში მაინც შეიძლება გამოიყოს ოპერაციები, რომელსაც ასრულებს ნებისმიერი სახის ***ალმ.*** ასეთი ოპერაციებია არითმეტიკული შეკრება და გამოკლება, ლოგი-

კური გამრავლება, შეკრება და *2*-ის მოდულით შეკრება, ინვერსია, სხვადასხვა მიმართულებით (მარცხნივ ან მარჯვნივ) ძვრა, დადებითი და უარყოფითი ზრდა (ინკრემენტი და დეკრემენტი). ეს ოპერაციები მხოლოდ ***ალმ***–ში არსებული აპარატურული საშუალებებით (ლოგიკური ელემენტებით) სრულდება, რის გამოც მას ელემენტალურ ოპერაციებსაც უწოდებენ. უფრო რთული ოპერაციები, როგორიცაა, მაგალითად, არითმეტიკული გამრავლება და გაყოფა, როგორც წესი, პროგრამულად (მიკროპროგრამული ხერხით) სრულდება.

ალმ მიეკუთვნება კომბინაციურ მოწყობილობებს, რადგან მას არ გააჩნია მეხსიერების საკუთარი ელემენტები და მისი გამოსასვლელი სიგნალები მხოლოდ შესასვლელი სიგნალების კომბინირებით მიიღება. მის მიერ კონკრეტული ელემენტარული ოპერაციის შესასრულებლად დახარჯული დრო სიგნალის გავრცელების დაყოვნების დროზეა დამოკიდებული, ე.ი. განისაზღვრება გამოყენებული საელემენტო ბაზის სიხშირული თვისებებითა და რეალიზებული ლოგიკურ ფუნქციების თავისებურებებით.

კომპიუტერის მუშაობის ანალიზმა გვიჩვენა, რომ მის მიერ შესრულებული ოპერაციების ***50%***-ს შეადგენს ***არითმეტიკული გამრავლება***, ხოლო დაახლოებით ***45%***-ს – ***არითმეტიკული შეკრება.*** აქედან ხდება ნათელი, თუ რატომ ითვლება მიკროპროცესორული სისტემის ერთ-ერთ ძირითად პარამეტრად როგორც შეკრების, ისე გამრავლების ოპერაციის შესასრულებლად საჭირო დრო. პირველი მათგანი განსაზღვრავს გამოყენებული საელემენტო ბაზის, ხოლო მეორე მათგანი – გამოყენებული ალგორითმების სრულყოფილობას. მოცემულ ნაშრომში ჩვენ განვიხილავთ მხოლოდ აპარატურული ხერხით ლოგიკური და არითმეტიკული ოპერაციების შესასრულებლად საჭირო ლოგიკური სქემების აგებასთან დაკავშირებულ ძირითად საკითხებს.

<u>სუმატორი</u> წარმოადგენს ორობითი კოდების სახით წარმოდგენილ რიცხვებზე შეკრების არითმეტიკული ოპერაციის შემსრულებელ კომბინაციურ ლოგიკურ მოწყობილობას.

სუმატორები ***ალმ***-ის ერთ-ერთი ძირითადი მოწყობილობებია. ტერმინი «სუმატორი» მოიცავს უმარტივესი ლოგიკური სქემებიდან დაწყებულ და ურთულესი ციფრული კვანძებით დამთავრებულ მოწყობილობათა ფართო სპექტრს. მათთვის საერთოა ის, რომ ნებისმიერი

მათგანი არითმეტიკულად კრებს ორობითი სახით წარმოდგენილ რიცხვებს. ამ რიცხვების თითოეული თანრიგისათვის გათვალისწინებულია ***3*** შესასვლელის მქონე ერთ თანრიგიანი მაჯამებელი ელემენტი (ნახ.***4.5,ბ***). ორი შესასვლელი განკუთვნილია მოცემული თანრიგის x_1 და x_2 შესაკრებებისათვის, ხოლო მესამე შესასვლელი – მეზობელი უმცროსი თანრიგიდან გადმოსული შესაკრებ **«1»**-ისათვის. განხილულ ელემენტს აქვს ***S*** და ***P*** გამოსასვლელი, რომელთაგანაც ***S*** გამოსასვლელიდან გაიცემა საძებნი x_1+x_2 ჯამი, ხოლო ***P*** გამოსასვლელიდან - უფროს თანრიგში გადასატანი ბიტი.

რამდენიმე თანრიგიანი ორობითი რიცხვების შესაკრებად გამოიყენება **«1»**-ის მიმდევრობითი გადატანიანი რამდენიმე სუმატორული ელემენტი. მიმდევრობითი გადატანიანი სუმატორების სწრაფმოქმედებას ზღუდავს ყველა ელემენტში გადატანების მიმდევრობით გადაცემისათვის საჭირო დრო. ამ მაჩვენებლის გასაუმჯობესებლად გამოიყენება პარალეური გადატანიანი სუმატორები.

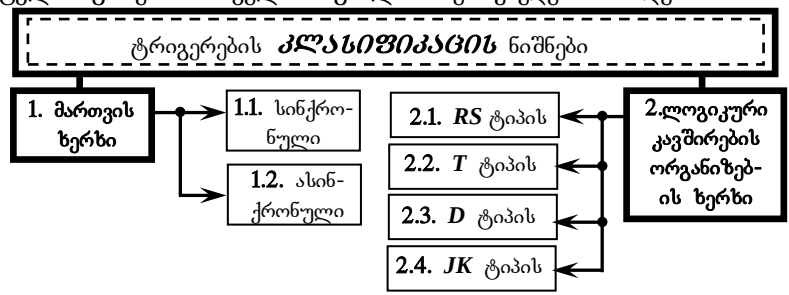
არითმეტიკული გამრავლებისა და გაყოფის ოპერაციები მიკროპროცესორულ სისტემაში ტრადიციულად პროგრამული ხერხით სრულდება. ტექნოლოგიების სფეროში ბოლო წლებში მიღწეული წარმატებების შემწეობით მოხერხდა დამუშავებულიყო აპარატურული ხერხით ამ ოპერაციების შემსრულებელი ინტეგრალური სქემები. მათმა გამოყენებამ მნიშვნელოვნად აამაღლა მიკროპროცესორული სისტემების სწრაფმოქმედება.

აპარატურული მამრავლებელი ფაქტურად ახდენს თანამამრავლთა თანრიგების კერძო ნამრავლების შეჯამებაზე დაფუძნებულ ტრადიციული ალგორითმის რეალიზებას. აღნიშნულის საილუსტრაციოდ განვიხილოთ ორთანრიგიანი ორობითი a_1a_0 და b_1b_0 კოდების გამრავლების (ნახ. ***4.5,ვ***) მაგალითი. მისი მარეალიზებელი მოწყობილობის სქემა ***4.5,დ*** ნახაზზეა მოყვანილი. მამრავლთა თანრიდების კერძო ნამრავლები ფორმირდება ***DD1…DD4*** აბრევიატურებით აღნიშნული ***ᲓᲔ***-ელემენტებით. მათ აჯამებს ***DD5*** და ***DD6*** სუმატორები. მოყვანილ სტრუქტურას ეწოდება ***მატრიცული მამრავლებელი ბლოკი.***

4.2. ციფრული ავტომატები (ტრიგერები, რეგისტრები, მთვლელები)

ციფრულ ავტომატებში მეხსიერების ელემენტებად გამოიყენება სხვადასხვა მოდიფიკაციის მიკროელექტრონული ტრიგერი.

ტრიგერი ეწოდება შესასვლელი სიგნალის ორი (ლოგიკური **«1»**-ისა და ლოგიკური **«0»**-ის ტოლი) მდგრადი მნიშვნელობის მაფორმირებელ მოწყობილობას, რომელიც ამ მნიშვნელობების ურთიერთშეცვლას გარე მმართველი სიგნალის ზემოქმედებით ახდენს.



ნახ. 4.6. *ტრიგერების კლასიფიკაცია*

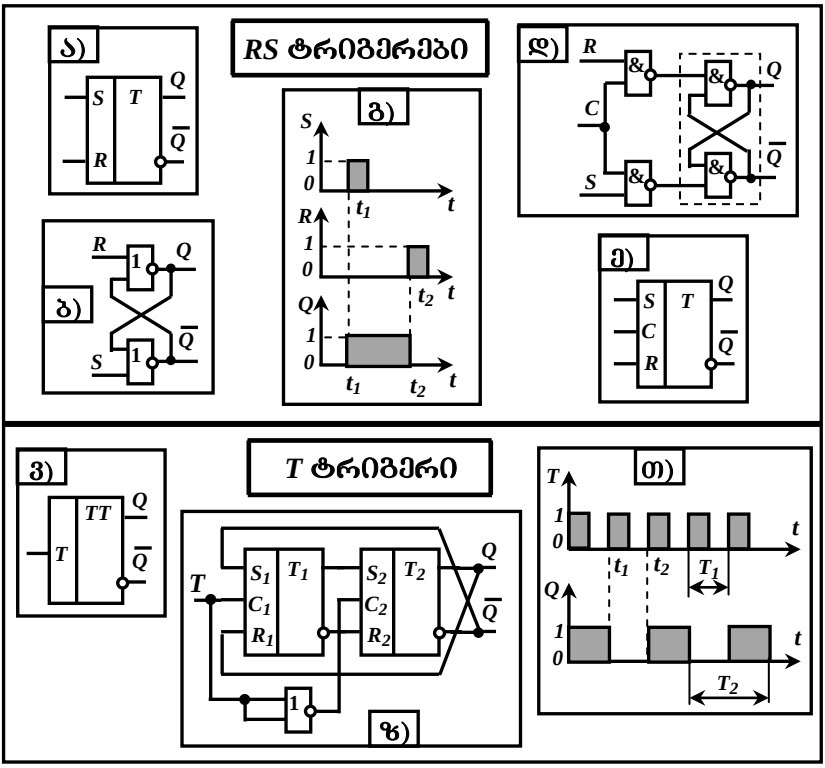
არსებობს სხვადასხვა სახის *ტრიგერი.* მათი კლასიფიკაცია ***4.6*** ნახაზზეა მოცემული. ელექტრონულ სქემებში გამოყენებულ პირობით გრაფიკულ გამოსახულებებში ტრიგერების იდენტიფიცირება მარჯვენა ზედა ნაწილში მოთავსებული ასო ***T***-ს (ნახაზები: ***4.7,ა,გ,ე; 4.8,ა***) მეშვეობით ხდება. ტრიგერის გამოსასვლელები აღინიშნება ***Q*** ასოებით. ამ დროს **«0»** გამოსასვლელის აღნიშვნისათვის ეს ასო ინვერსიის ნიშნით, ე.ი $\overline{Q}$ (იხ. ნახ. ***4.7***; ***4.8***) სახით აიღება.

მოკლედ განვიხილოთ ტრიგერების ***4.6*** ნახაზზე ჩამოთვლი ნაირსახეობები.

ასინქრონულ RS-ტრიგერს (ნახ.***4.7,ბ***)გააჩნია ორი საინფორმაციო შესასვლელი, რომელთაგანაც ერთ-ერთი აღნიშნულია ასოთი ***S*** (ინგ. ***Set*** ანუ «დაყენება»), ხოლო მეორე – ასოთი ***R*** (ინგ ***Reset*** ანუ «ჩამოყრა»). მისი პირდაპირი გამოსასვლელი აღნიშნულია როგორც ***Q***, ხოლო ინვერსირებული გამოსასვლელი – როგორც $\overline{Q}$.

ასინქრონული ***RS*** ტრიგერი მიიღება დადებითი ჯვარედინი კავშირებით დაკავშირებული ორი ლოგიკური ***ან-არა*** ელემენტით (ნახ.

4.7,ბ). მისი მუშაობის პრინციპის მაჩვენებელი დროითი დიაგრამა ***4.7,გ*** ნახაზზეა მოყვანილი.



ნახ. 4.7. *ასინქრონული* ***RS ტრიგერის :*** *პირობითი გამოსახულება* ***(ა);*** *ფუნქციური სქემა* ***(ბ);*** *დროითი დიაგრამა* ***(გ);*** *სინქრონული* ***RS ტრიგერის:*** *ფუნქციური სქემა* ***(დ);*** *პირობითი გამოსახულება* ***(ე).*** ***T ტრიგერის:*** *პირობითი აღნიშვნა* ***(ვ);*** *ფუნქციური სქემა* ***(ზ);*** *დროითი დიაგრამა* ***(თ).***

S შესასვლელზე ლოგიკური ***«1»***-ის მიეწოდებისას (იხ.ნახ. ***4.7,ა***), ე.ი. როდესაც ***S=1*** (***4.7,გ*** ნახაზზე ნაჩვენები t_1 მომენტის დროს) ინვერსიულ $\overline{Q}$ გამოსასვლელზე წარმოიქმნება ლოგიკური ***«0».*** აღნიშნული სიგნალი უკუკავშირის წრედით მიეწოდება ***ან-არა*** ელემენტის ერთ-ერთ (ჩვენს შემთხვევაში ქვედა) შესასვლელს. ამ ელემენტის მეორე ***R*** შესასვლელზე ლოგიკური ***«0»***სიგნალის არსებობის გამო გვექნება: ***Q=1***. ტრიგერის ეს მდგომარეობა დიდი ხნის განმა-

ვლობაში შენარჩუნდება და არ შეიცვლება ***S*** შესასვლელზე არსებული სიგნალის შეცვლის დროსაც მანამ, სანამ ***R*** შესასვლელზე შენარჩუნებული იქნება ლოგიკური **«0»** (მდგომარეობა, რომლის დროსაც ***S=1*** და ***R=1,*** მოცემული ტრიგერისათვის ***აკრძალულია***, ვინაიდან შესასვლელი სიგნალების ამ კომბინაციის დროს ტრიგერის გამოსასვლელის მდგომარეობის წინასწარმეტყველება შეუძლებელია). ამგვარად, ***RS***-ტრიგერი იმახსოვრებს ***S*** შესასვლელზე ლოგიკური **«1»** სიგნალის მიწოდების ფაქტს მანამ, სანამ ***R*** შესასვლელზე არ წარმოიქმნება ჩამოყრის **«1»** სიგნალი (ე.ი. ***4.7,ბ*** ნახაზზე ნაჩვენები დროის t_2 მომენტამდე).

R=1-ის დროს ***Q*** გამოსასვლელზე წარმოიქმნება სიგნალი **«0»,** ე.ი. ***Q=0,*** ხოლო შესაბამისი უკუკავშირით უზრუნველყოფილი იქნება $\overline{Q}=1$ ტოლობა და ა.შ.

<u>სინქრონულ RS ტრიგერს</u> (ანუ ***<u>RST ტრიგერს</u>***) (ნახ.***4.7,დ***), აქვს დამატებითი ***C*** შესასვლელი (ინგ. ***<u>C</u>****lock*, ქართ. "საათი"). მასზე მიეწოდება სინქრონიზაციის იმპულსები. სინქრონული ტრიგერი მიიღება ასინქრონული ***RS*** ტრიგერის შესასვლელთან დამატებითი ორი ***და-არა*** ელემენტის მიერთებით (იხ. ნახ. ***4.7,დ***).

სინქრონიზაციის ***C***-შესასვლელზე სიგნალის არარსებობისას (ე.ი. როდესაც ***C=0***) ***R*** და ***S*** შესასვლელები საკუთარი ტრიგერიდან (იგი პუნქტირულ მართკუთხედშია ჩასმული) განრთული აღმოჩნდება და ამ შესასვლელებზე სიგნალების ცვლილებას აღნიშნული ტრიგერის მდგომარეობის შეცვლა არ შეუძლია.

ზემოთ აღნიშნულ შესასვლელზე ნებადამრთველი სიგნალის გაჩენისას, ე. ი. როდესაც ***C=1***, შესასვლელი ***და-არა*** ელემენტები ასრულებს ინვერტორების ფუნქციას. ამ დროს ტრიგერის მდგომარეობას ასინქრონული ***RS*** ტრიგერის ანალოგურად ცალსახად განსაზღვრავს ***R***- და ***S***- შესასვლელებზე არსებული სიგნალების მნიშვნელობები.

აღსანიშნავია, რომ განხილულ ტრიგერს საკუთარი მნიშვნელობის შეცვლა შეუძლია იმ ინტერვალის ნებისმიერ მომენტში, რომლის განმავლობაშიც ***C=1***. ასეთ ტრიგერს ეწოდება ***სინქრონიზაციის სტატიკური შესასვლელის მქონე ტრიგერი.***

პრაქტიკაში ძალიან ძალიან გავრცელებულია ***სინქრონიზაციის დინამიკური (იმპულსური) შესასვლელის მქონე ტრიგერები.*** ასეთი

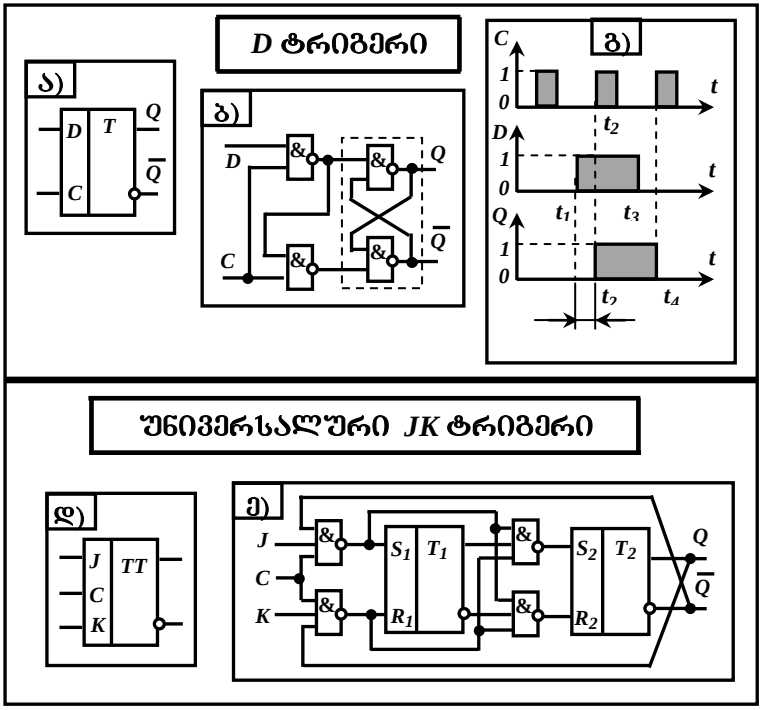
ტრიგერის თავისებურებაა ის, რომ ერთი მდგომარეობიდან მეორეში მისი გადართვა მხოლოდ შესასვლელი ***C*** სიგნალის ცვლილების პერიოდშია შესაძლებელი; გადართვა ხდება სინქრონიზაციის იმპულსის წინა ან უკანა ფრონტით. ასეთი გადაწყვეტა მნიშვნელოვნად ამაღლებს ტრიგერული მოწყობილობების დაბრკოლებამდგრადობას, რადგან ამ დროს მაქსიმალურად მცირდება ინფორმაციის გადაწერისათვის საჭირო ინტერვალი. მუშაობის მითითებული რეჟიმის რეალიზებისათვის საჭიროა ***4.7,დ*** ნახაზზე არსებული დამატებითი ***და-არა*** ელემენტები შევცვალოთ დამხმარე ***RS***-ტრიგერებით, რის შედეგადაც მიიღება ე. წ. ***სამი ტრიგერის სქემა***.

<u>T ტრიგერი</u> (ინგ. ***<u>T</u>****umbler* – ტუმბლერი, გადამრთველი) ფართოდ გამოიყენება ციფრულ მთვლელებში. ამ ტრიგერში (ნახ.***4.7,ე***) არსებობს ***T*** შესასვლელი, რომელზეც (იმპულსის) ყოველი ზემოქმედება იწვევს ტრიგერს ერთი მდგომარეობიდან მეორეში გადართვას. გადართვების რაოდენობა ***T***-შესასვლელზე მოსული იმპულსების რაოდენობის ტოლია, რის გამოც მას ***მთვლელი შესასვლელის მქონე ტრიგერსაც*** უწოდებენ.

T ტრიგერების უმრავლესობას საფუძვლად უდევს ორსაფეხუროვანი ტრიგერის სქემა; ეს უკანასკნელი წარმოქმნილია გადაჯვარედინებული უკუკავშირებით მოცული სინქრონული ორი ***RST*** ტრიგგერით, რომლებიც მიმდევრობითაა შეერთებული (ნახ.***4.7,ვ***). გარდა ამისა, ამ ***RST*** ტრიგერთა ***C*** შესასავლები ერთმანეთთან ***არა***-ელემენტითაა შეერთებული და მათი გაერთიანებით წარმოქმნილია მთლიანად ტრიგერის ***T*** შესასვლელი (ნახ.***4.7,ზ***). ამგვარად C_1 შესასვლელზე ყოველი ზემოქმედება შესასვლელი ხაზიდან განრთავს მეორე ***RST*** ტრიგერს. T_2 ტრიგერთან უკუკავშირის წრედით T_1 ტრიგერის შეერთების გამო ამ უკანასკნელის თითოეული იძულებითი გადართვა იწვევს T_2 ტრიგერის «გადაყირავებას» (მდგომარეობის შეცვლას). ***T*** ტრიგერის მუშაობის პრინციპი ილუსტრირებულია ***T*** შესასვლელსა და ***Q*** გამოსასვლელზე მიმდინარე პროცესების დროითი დიაგრამით (ნახ. ***4.7,თ***)

<u>D ტრიგერი</u> (ინგ. ***<u>D</u>****elay* – შეყოვნება), რომლის პირობითი გამოსახულება ***4.8,ა*** ნახაზზეა მოყვანილი, ***Q*** გამოსასვლელზე სიგნალს წარმოქმნის ***D*** შესასვლელზე მმართველი ზემოქმედებიდან გარკვეული დროის გასვლის შემდეგ (ე.ი. დაყოვნებით). ამისათვის მას აქვს ***C***

შესასვლელი, რომლის (სტატიკურად ან დინამიკურად) აგზნება საშუალებას იძლევა ტრიგერი გადაირთოს ***D*** შესასვლელზე არსებული სიგნალის შესაბამისად. ***D*** ტრიგერი მიიღება ***RS***-ტრიგერისაგან. თუ შესასვლელზე გამოვიყენებთ ერთნაირი ტიპის ლოგიკურ ***და-არა*** ელემენტებს (ნახ.***4.8,ბ***). პროცესების დროითი დიაგრამები ***4.8,გ*** ნახაზზეა მოყვანილი.



ნახ. 4.8. D ტრიგერის: *პირობითი გამოსახულება* ***(ა)****; ფუნქციური სქემა* ***(ბ)****; დროითი დიაგრამა* ***(გ)****;* ***უნივერსალური JK ტრიგერის:*** *პირო-ბითი გამოსახულება* ***(დ)****; ფუნქციური სქემა* ***(ე)****.*

<u>უნივერსალური JK ტრიგერს</u>, რომლის პირობითი გამოსახულება ***4.8,ა*** ნახაზზეა მოყვანილი, გააჩნია ორი საინფორმაციო (***J*** და ***K***) და ერთი მასინქრონიზებელი (***C***) შესასვლელი. იგი აგებულია უკუკავშირებიანი ორი ***RST*** ტრიგერით (ნახ.***4.8,ე***).

JK-ტრიგერს თვლიან ***უნივერსალურ ტრიგერად,*** ვინაიდან მისი შესასვლელი მომჭერების სხვადასხვანაირად გადართვის გზით შეიძ-

ლება ყველა სხვა სახის ტრიგერი მივიღოთ. მიკროელექტრონული შესრულების დროს ***JK*** ტრიგერს გააჩნია გაფართოებული შესასვლელი ლოგიკა, რომლის წყალობითაც ასეთი მრავალფუნქციური ტრიგერი შეიძლება ერთი და იმავე სერიის სხვადასხვა მიკროსქემაში გამოვიყენოთ.

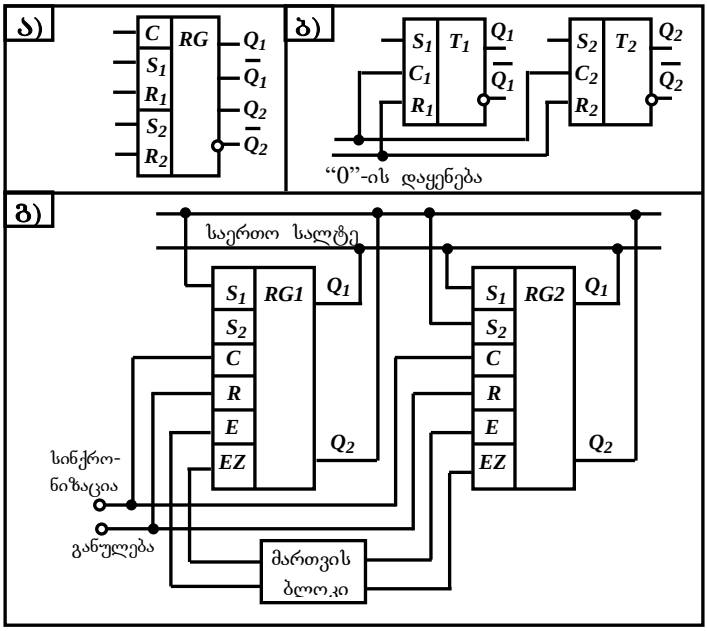
<u>რეგისტრები.</u> ორობითი ფორმით წარმოდგენილი ინფორმაციის შესანახად განკუთვნილ მოწყობილობას ***რეგისტრს*** უწოდებენ. იგი მიიღება **«0»**-ის ან **«1»**-ის შემნახველი ტრიგერების ურთიერთდაკავშირებით. ტრიგერების რაოდენობა განსაზღვრავს რეგისტრის თანრიგიანობას. რეგისტრებს შეუძლია მართვის სისტემის მეშვეობით შეასრულოს ინფორმაციის მიღების, გადაცემისა და გარდაქმნის ოპერაციებიც.

შესასრულებელ ფუნქციებზე დამოკიდებულებით განასხვავებენ ***პარალელურ და ძვრის რეგისტრებს.***

RST ტრიგერებით აგებული უმარტივესი ორთანრიგიანი პარალელური რეგისტრის პირობითი გამოსახულება ***4.9,ა*** ნახაზზე, ხოლო მისი ფუნქციური სქემა - ***4.9,ბ*** ნახაზზეა მოყვანილი. ორობითი კოდის სახით ინფორმაციის ჩასაწერად თავდაპირველად ყველა ***R***- და ***S***-შესასვლელებზე სათანადო სიგნალების მიწოდების გზით ტრიგერები უნდა გავანულოთ. ამის შემდეგ შეიძლება ტრიგერების მდგომარეობა არ შევცვალოთ (**«0»**-ის შესანარჩუნებლად) ან შევცვალოთ ***S***-შესასვლელელის მეშვეობით (**«1»**-ის ჩასაწერად). რეგისტრში არსებული ინფორმაცია შეიძლება წავიკითხოთ რეგისტრის ***Q***–გამოსასვლელებიდან.

რეგისტრის უმნიშვნელოვანესი პარამეტრებია ***თანრიგების რაოდენობა*** და ***სწრაფმოქმედება***. დაბალი სწრაფმოქმედების დროს იზღუდება სისტემის მართვის ტაქტური იმპულსების მაქსიმალური სიხშირე, რომელიც უზრუნველყოფს ინფორმაციის ჩაწერას, წაკითხვასა და უმარტივეს დამუშავებას. ინფორმაცია უმარტივესად მუშავდება ძვრის რეგისტრებში ინფორმაციის (შიგთავსების) მარცხნივ და/ან მარჯვნივ ძვრის გზთ (ძვრის მიმართულება, როგორც ***§2.11***-შია ნაჩვენები, დამოკიდებულია იმაზე, შიგთავსი იყოფა თუ მრავლდება 2^n,***n*** **=0,1,2,…** მნიშვნელობაზე). ***ძვრის რეგისტრებში*** ჩვეულებრივ გამოიყენება მიმდევრობით შეერთებული ***D*** ტრიგერები, რომლებსაც აქვს შემდეგი სამი შესასვლელი: საინფორმაციო (***D*** შესასვლელი), ძვრის (***C***

შესასვლელი) და დასაყენებელი (***R*** შესასვლელი). ამ დროს მთელი მოწყობილობის ერთდროულად მართვისათვის გაერთიანებულია ყველა ტრიგერის ***C***- და ***R***-შესასვლელები. ძვრის რეგისტრი შეიძლება ავაგოთ ***RS***-ტრიგერებითაც, ოღონდ ამ შემთხვევაში თითოეული თანრიგისათვის დროში დაძრული ტაქტური იმპულსებით მართვადი ორი ტრიგერის გამოყენება მოგვიწევს. ამ შემთხვევაში ჩნდება შესასვლელიდან გამოსასვლელისაკენ ჩასაწერი სიგნალის თანრიგობრივად გადაადგილების შესაძლებლობა.



ნახ. 4.9. *ორთანრიგიანი* ***რეგისტრის*** *პირობითი გამოსახულება* ***(ა)*** *და ფუნქციური სქემა* ***(ბ);*** *საერთო სალტესთან რამდენიმე რეგისტრის მიერთების სქემა* ***(გ);***

რთულ ციფრულ მოწყობილობებში ცალკეულ რეგისტრებს შორის ინფორმაცია, როგორც წესი, ყველა რეგისტრის შესაბამისი თანრიგების შესასვლელებისა და გამოსასვლელების შემაერთებელი საერთო სალტით გაიცვლება (ნახ.***4.9,გ***). ამ დროს თითოეულ რეგისტრს აქვს ჩაწერის ნებადართვის ***E*** შესასვლელი და შესაბამისი გამოსასვლე-

ლი გამომყვანებიდან მისი ტრიგერების განმრთველი ***EZ*** შესასვლელი. ამ სქემაში ინფორმაციის გადაცემის კონკრეტული გზა ამოირჩევა მართვის ბლოკიდან შესაბამისი ნებადამრთველი სიგნალების მიწოდებით. მაგალითად, ***RG2*** რეგისტრიდან ***RG1*** რეგისტრში ინფორმაციის გადასაწერად ნებადამრთველი სიგნალები საჭიროა მივაწოდოთ ***RG2*** რეგისტრის ***EZ*** შესასვლელსა და ***RG1*** რეგისტრის ***E*** შესასვლელს. ამ შემთხვევაში საერთო სალტესთან მიერთებული იქნება მხოლოდ ***RG2*** რეგისტრის გამოსასვლელები და ***RG1*** რეგისტრის შესასვლელები. ამიტომ სინქრონიზაციის იმპულსით მოხდება ინფორმაციის მოთხოვნილი გადაწერა.

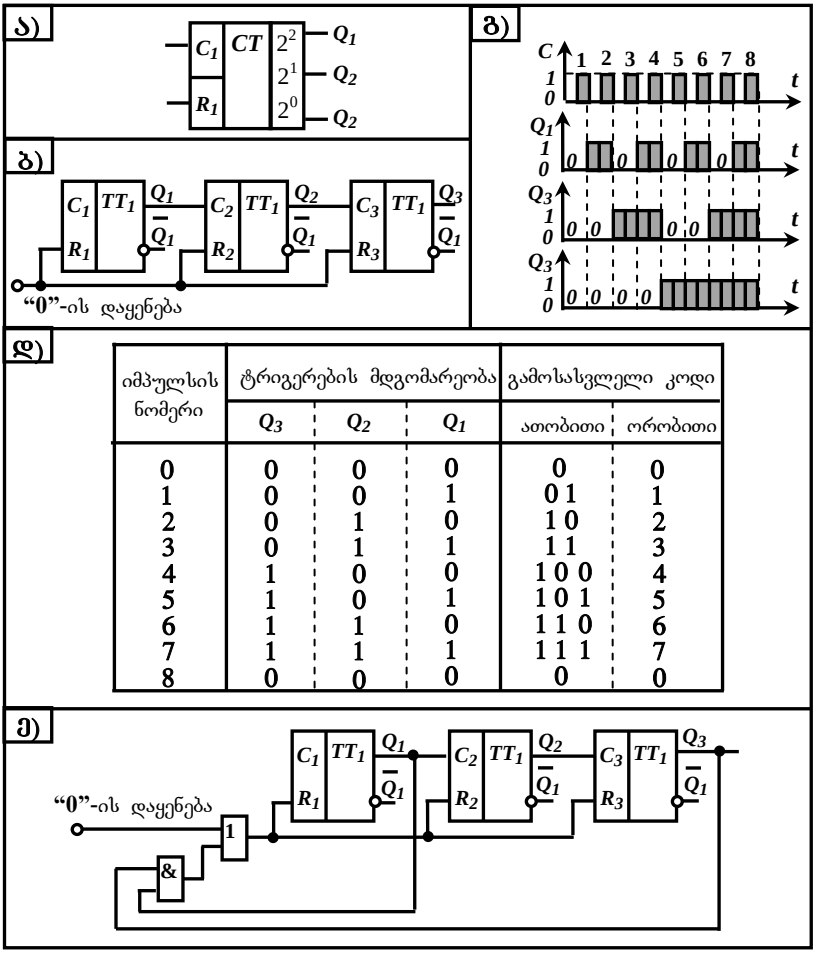
<u>იმპულსების ციფრული მთვლელები</u>. ***ციფრული მთვლელები*** ეწოდება ტრიგერებით აგებულ ფუნქციურ მოწყობილობას, რომელიც ითვლის მის შესასვლელებზე მოსულ იმპულსებს. ჩვეულებრივ ორობითი კოდის სახით ფორმირებული თვლის შედეგი შეიძლება წაკითხული იქნას ან მოხდეს მისი შენახვა მთვლელის ტრიგერში. საჭიროებისამებრ თვლის შედეგი შესასვლელზე იმპულსის თითოეული მისვლის შემდეგ წაიკითხება. მთვლელს ორობითი კოდის სახით წარმოდგენილ 2^{n-1}-ზე არაუმეტეს რიცხვამდე თვლა შეუძლია, სადაც ***n*** მიმდევრობით შეერთებული ტრიგერების რაოდენობაა. შესასვლელზე მოსული იმპულსების რაოდენობა თუ არ იზღუდება, მაშინ ყოველი 2^{n-1} რაოდენობის იმპულსის მოსვლის შემდეგ მთვლელი ბრუნდება საწყის ნულოვან მდგომარეობაში და ხელახლა იწყებს თვლას. ასეთი ტიპის მთვლელს ეწოდება ***კვლავდამთვლელი*** ან ***გადამთვლელი*** შეიძლება ვუწოდოთ.

ციფრული მთვლელები შეიძლება უმარტივესი ტრიგერული და ლოგიკური მიკროსქემებით ავაგოთ. არსებობს მაღალი დონის ინტეგრაციის ***ერთი მიკროსქემის*** სახით დამზადებული მრავალთანრიგიანი უნივერსალური მთვლელებიც.

მიზნობრივი დანიშნულების მიხედვით არსებობს შემდეგი ტიპის მთვლელები: ***1)*** ამჯამავი (პირდაპირ ითვლის იმპულსებს); ***2)*** გამომკლები (მაქსიმალური მნიშვნელობიდან ახდენს ნულამდე უკუთვლას); ***3)*** რევერსიული (ახდენს როგორც პირდაპირ-, აისე უკუთვლასაც).

მთვლელის ფუნქციონირების პრინციპი განვიხილოთ ***T*** ტრიგერებით აგებული ***3***-თანრიგიანი მთვლელის მაგალითზე. მისი პირობითი

აღნიშვნა და ფუნქციური სქემა ***4.10,ა,ბ*** ნახაზებზეა მოყვანილი. მუშაობის დაწყების წინ მთვლელის გასანულებლად გამოიყენება სპეციალური სალტე ***«0»-ის დაყენება*** (იხ.ნახ.***4.10,ბ***), რომელთანაც მიერთებულია ტრიგერების ყველა ***R*** შესასვლელი.



იმპულსის ნომერი	ტრიგერების მდგომარეობა			გამოსასვლელი კოდი	
	Q_3	Q_2	Q_1	ათობითი	ორობითი
0	0	0	0	0	0
1	0	0	1	0 1	1
2	0	1	0	1 0	2
3	0	1	1	1 1	3
4	1	0	0	1 0 0	4
5	1	0	1	1 0 1	5
6	1	1	0	1 1 0	6
7	1	1	1	1 1 1	7
8	0	0	0	0	0

ნახ. 4.10. *3-თანრიგიანი* ***მთვლელის:*** *პირობითი გამოსახულება* ***(ა),*** *ფუნქციური სქემა* ***(ბ),*** *დროითი დიაგრამა* ***(გ)*** *და მისი მუშაობის პროცესის მაილუსტრირებელი ცხრილი* ***(დ);*** *4-მდე თვლის უნარის მქონე ორობითი მთვლელი* ***(ე)***

4.10,გ ნახაზზე მოყვანილი დიაგრამიდან ჩანს, რომ მთვლელის ***C***-შესასვლელზე თვლის იმპულსების გამოჩენისას მიმდევრობით გადაირთვება სქემაში არსებული ტრიგერები; ამასთანავე ყოველი მომდევნო ტრიგერის გადართვის პერიოდი წინა პერიოდთან შედარებით ორმაგდება. გადართულ ტრიგერთა ***Q***-გამოსასვლელებზე სიმბოლო ***"1"***-ის მიწერით (დროით დიაგრამზე ეს მდგომარეობები თალხი ფერითაა აღნიშნული) შეიძლება მოვახდინოთ მთვლელის მდგომარეობათა იმგვარი სისტემატიზება როგორც ეს ***4.10,დ*** ნახაზზე მოყვანილ ცხრილშია ნაჩვენები. მის თანახმად ტრიგერები მდგომარეობა **«1»**-ში მას შემდეგ გადაირთვება, როდესაც წინა ტრიგერის მდგომარეობა **«1»** შეიცვლება მდგომარეობა **«0»**-ით. ეს ნიშნავს, რომ მითითებულ რეჟიმში ფორმირდება შემდეგი ტრიგერის გადამყირავებელი გადატანის სიგნალი.

მითითებული ცხრილიდან ისიც ჩანს, რომ ***3***-თანრიგიანი მთვლელის ***გადათვლის მოდული***, ე. ი. მთვლელის განულებებს შორის არსებულ მდგომარეობათა რაოდენობა, $2^3 = 8$-ის ტოლია. ამ დროს პირველი ტრიგერის წონაა 2^0, მეორე ტრიგერის – 2^1, ხოლო მესამე ტრიგერის – 2^2. აღნიშნული «წონები» მითითებულია ***3***-თანრიგიანი მთვლელის პირობითი გამოსახულების მარჯვენა (დამხმარე) ზონაში (იხ.ნახ. ***4.9,ა***).

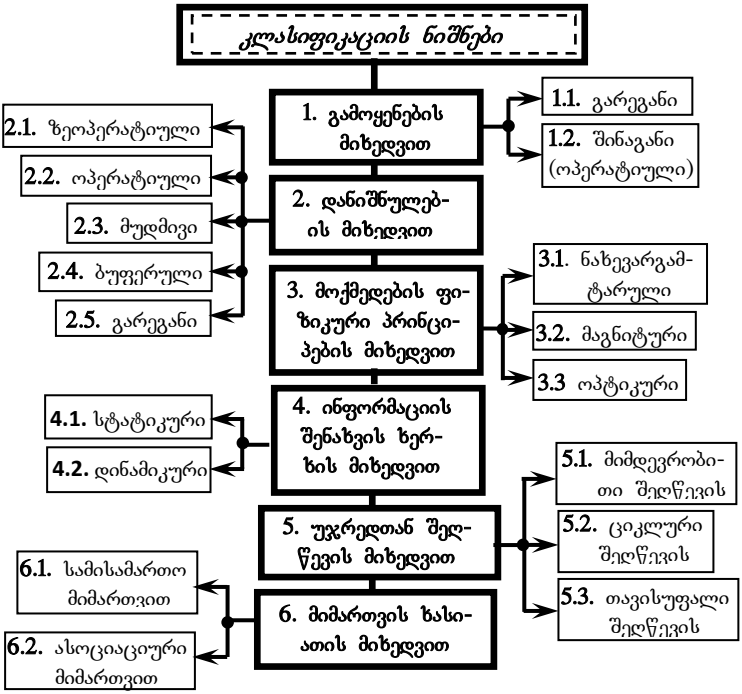
4-მდე თვლის უნარის მქონე იმპულსების ორობითი მთვლელის დასაპროექტებლად ***3***-თანრიგიანი ორობითი მთვლელის ზემოთ განხილულ სქემას უნდა დავუმატოთ უკუკავშირი, რომელიც ყოველი მე-***5*** იმპულსის მოსვლისას გაანულებს სქემაში არსებულ ტრიგერებს და მთვლელი თავიდან დაიწყებს თვლას (ნახ. ***4.9,ე***).

4.3. მიკროპროცესორული სისტემის მეხსიერების კლასიფიკაცია

მიკროპროცესორული სისტემის მეხსიერება აღნიშნული სისტემის ფუნქციური ნაწილია, რომელიც განკუთვნილია მონაცემების ჩასაწერად, შესანახად და გასაცემად. აღნიშნულის შესაბამისად განასხვავებენ მეხსიერების მუშაობის ჩაწერის, შენახვისა და წაკითხვის რეჟიმებს. ინფორმაციის დამხსომებელ მოწყობილობაში (***დმ***-ში) ჩაწერას ან მისგან ამოკითხვას სხვაგვარად ***დმ-თან მიმართვასაც*** უწოდებენ. მეხსიერების სწრაფმოქმედება განისაზღვრება ***დმ***-თან მიმართვის ოპერაციის ხანგრძლივობით. ***ინფო-***

რმაციის ჩაწერის შემთხვევაში **ᲓᲛ**-თან მიმართვის $t_{\text{მიმ}}$ დრო მიიღება ინფორმაციის $t_{\text{ძ}}$ მოძებნის, ადრე ჩაწერილი ინფორმაციის $t_{\text{წჟ}}$ წაშლისა და მის ნაცვლად ახალი რიცხვის $t_{\text{ჩწ}}$ ჩაწერის დროების შეკრებით:

$$t_{\text{მიმ}} = t_{\text{ძ}} + t_{\text{წჟ}} + t_{\text{ჩწ}}.$$



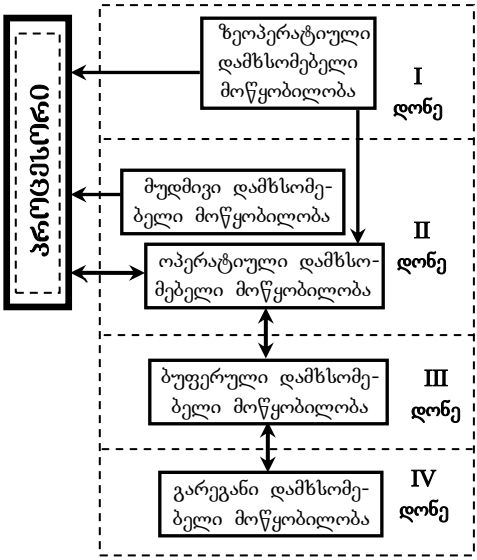
ნახ. 4.11. *დამხსომებელი მოწყობილობების კლასიფიკაცია*

ინფორმაციის წაკითხვის შემთხვევაში **ᲓᲛ**-თან მიმართვის $t_{\text{მიმ}}$ დრო მიიღება ინფორმაციის $t_{\text{ძ}}$ მოძებნის, მისი $t_{\text{წკ}}$ წაკითხვისა და წაკითხული კოდების $t_{\text{აღდგ}}$ აღდგენის დროების შეკრებით (წაკითხული ინფორმაციის აღდგენა საჭიროა იმ შემთხვევაში, თუ წაკითხვის პროცესის დროს ზიანდება ჩაწერილი ინფორმაცია):

$$t_{\text{მიმ}} = t_{\text{ძ}} + t_{\text{წკ}} + t_{\text{აღდგ}}.$$

დამხსომებელი მოწყობილობების კლასიფიკაცია ***4.11*** ნახაზზეა მოცემული. შევნიშნავთ, რომ სამისამართო მიმართვის მქონე მეხსიერებას ხშირად ***სამისამართო ამორჩევის მეხსიერებასაც*** უწოდებენ. ასევე ასოციაციური მიმართვის მეხსიერება ცნობილია ***ასოციაციური ამორჩევის მეხსიერების*** სახელწოდებითაც.

ერთდროულად მაღალი რომ იყოს მიკროპროცესორულ სისტემის როგორც საინფორმაციო ტევადობა, ისე სწრაფმოქმედებაც, მისი მეხსიერება იერარქიულად უნდა იყოს აგებული. მეხსიერების იერაქიულობა ზრდის მეხსიერების ტევადობას და ამცირებს მეხსიერებასთან მიმართვის დროს. შესაძლებელი ხდება გავზარდოთ მეხსიერებაში შესანახი მონაცემების რაოდენობა, რაც რთული ამცანების გადასაწყვეტადაა საჭირო. ***დმ***-ის სტრუქტურის იერარქიულად აგებისას ინფორმაციის ნაკადების ლოგიკური ორგანიზაცია



ნახ. 4.12. *მიკროპროცესორულ სისტემათა დამხსომებელი მოწყობილობების იერარქია*

ისეთი უნდა იყოს, რომ კომპიუტერული სისტემის მთლიანი საინფორმაციო ველი წარმოიდგინებოდეს შინაგანი აბსტრაქტული ***დმ***-ს სახით, რომელსაც ექნება ერთიანი სამისამართო სივრცე. ასეთ აბსტრაქტულ ***დმ***-ს ***ვირტუალურ*** (*წარმოდგენით*) ***დმ***-ს უწოდებენ. მისი უჯრედები მისამართდება აბსტრაქტული მათემატიკური მისამართებით, რისთვისაც გამოიყენება გვერდობრივი ცხრილები. იერარქიულ სტრუქტურაში შეიძლება გამოვყოთ მეხსიერების შემდეგი დამხსომებელი მწყობილობები (ნახ. ***4.12***):

1. ზეოპერატიული დამხსომებელი მოწყობილობები. მათ ***ადგილობრივ მეხსიერებასაც*** უწოდებენ, რომელსაც აქვს პროცესორის თანაზომადი სწრაფმოქმედება. ზეოპერატიული ***დმ***-ის ტევადობა ჩვეულებრივად რამდენიმე ათეული სიტყვიდან რამდენიმე ათას სიტყვამდე, ხოლო მათდამი მიმართვის დრო – რამდენიმე მეათედი მიკროწამიდან რამდენიმე მეასედ მიკროწამამდე იცვლება. ისინი განკუთვნილია პროგრამის მიმდინარე ბრძანებათა გარკვეული მიმდევრობების შესანახად. ზეოპერატიული ***დმ***-ები იმ შემთხვევაში გამოიყენება, როდესაც პროცესორის სწრაფმოქმედებას ზღუდავს ოპერატიული დამხსომებელი მოწყობილობების სწრაფმოქმედება.

2. ***ოპერატიული დამხსომებელი მოწყობილობები*** (***RAM***), რომლებს ***ძირითად მეხსიერებასაც*** უწოდებენ. აღნიშნული სახის დამსომებელ მოწყობილობა არაა სპეციალიზებული ინფორმაციის ან მარტო ჩაწერაზე, ან მისგან ინფორმაციის მარტო წაკითხვაზე: ***თავისუფლად*** შეიძლება ამოვირჩიოთ მუ-

შაობის ორივე რეჟიმი; ამიტომ ხშირად მას ***თავისუფალი შემწვევის მეხსიერებას,*** ანუ ***RAM***-ს (***R****andom* ***A****ccess* ***M****emory*) უწოდებენ. რადგან განხილული სახის მეხსიერებისათვის რეალიზებადია როგორც ჩაწერის, ისე წაკითხვის რეჟიმი, ***წაკითხვა/ჩაწერის მეხსიერება,*** ანუ ***RWM***-ს (***R***e-ad/***W****rite* ***M****emor*) უწოდებენ. მაშასადამე, შემოკლებით ოპერატიულ მეხსიერებას შეიძლება ***RAM*** ან ***RWM*** ვუწოდოთ. ჩვენ გამოვიყენებთ პირველ მათგანს.

ოპერატიული ***დმ*** გამოიყენება მიმდინარე გამოთვლებისათვის საჭირო მონაცემებისა და პროგრამების, აგრეთვე ისეთი პროგრამების შესანახად, რომლებზედაც საჭიროა მოხდეს სწრაფი გადასვლა გამოთვლითი პროცესის მიმდინარეობისას შეწყვეტების წარმოშობის დროს. თანამედროვე მიკროპროცესორული სისტემების ოპერატიულ დამხსომებელ მოწყობილობებში შეიძლება შევინახოთ რამდენიმე ათასი სიტყვიდან რამდენიმე ასეულ ათას სიტყვამდე; მათდამი მიმართვის დრო მიკროწამის გარკვეული ნაწილიდან რამდენიმე მიკროწამამდე იცვლება. ოპერატიული დამხსომებელი მოწყობილობები პროცესორთან შეიძლება როგორც უშუალოდ, ისე ზეოპერატიული დამხსომებელი მოწყობილობების მეშვეობით, იყოს დაკავშირებული. ოპერატიულ და ზეოპერატიულ დამხსომებელ მოწყობილობათა ***მეხსიერების ელემენტებად*** გამოიყენება ნახევარგამტარული ელემენტები, თხელი მაგნიტური ფირები, ფერიტული გულარები და ა.შ.

3. ***მუდმივი დამხსომებელი მოწყობილობები*** (***ROM***) განკუთვნილია მხოლოდ გამოთვლების პროცესში უცვლელი ინფორმაციის, მაგალითად, მუდმივად გამოყენებადი პროგრამების, მონაცემების, ფუნქციათა ცხრილების, მიკროპროგრამების და ა.შ, შენახვისა და წაკითხისათვის. მათში ინფორმაცია მხოლოდ ერთხელ (დამზადების პროცესში) ჩაიწერება და მუშაობის დროს შესაძლებელია მხოლოდ მისი წაკითხვა. ამიტომ მას მხოლოდ ***წაკითხვისათვის განკუთვნილ მეხსიერება***, ანუ ***ROM***-ს (***R****ead-****O****nly* ***M****emory*) უწოდებენ. რომელსაც ჩვენ მუდმივი მეხსიერების შემოკლებით აღნიშვნისათვის გამოვიყენებთ. მუდმივი დამხსომებელი მოწყობილობების აპარატურული სირთულე ჩამოუვარდება ოპერატიული დამხსომებელი მოწყობილობების აპარატურულ სირთულეს.

4. ***ბუფერული დამხსომებელი მოწყობილობები მოწყობილობები*** (***ბდმ***) გამოიყენება სხვადასხვა სწრაფმოქმედების მქონე, მაგალითად, ოპერატიულ და გარე დამხსომებელ მოწყობილობებს შორის ინფორმაციის გაცვლისას მონაცემების საშუალედო შენახვისათვის. ტევადობისა და სწრაფმოქმედების მიხედვით ბუფერულ დამხსომებელ მოწყობილობებს უკავია საშუალედო ადგილი ოპერატიულ და გარე დამხსომებელ მოწყობილობებს შორის. მათი აგებისათვის შეიძლება გამოყენებული იქნეს ნახევარგამტარული ელემენტები, ფერიტული გულარები და მაგნიტური დისკები.

5. გარე დამხსომებელი მოწყობილობები (**ᲒᲓᲛ**) ინფორმაციის დიდი მასივების შესანახად გამოიყენება. მათში ერთდროულად შესანახი მონაცემების ***მოცულობა*** შეიძლება აჭარბებდეს ასაობით მილიონ სიტყვებს, ***მათდამი მიმართვის დრო*** კი იცვლება რამდენიმე მილიწამებდან წამის რამდენიმე მეათედამდე. გარე დამხსომებელ მოწყობილობებში შენახული მონაცემები გამოთვლით პროცესში უშუალოდ არ გამოიყენება: ისინი გარე მოწყობილობებიდან მიეწოდება ოპერატიულ დამხსომებელ მოწყობილობებს. გარე დამხსომებელ მოწყობილობებად ხშირად გამოიყენება მაგნიტური ლენტები, მოქნილი და ხისტი მაგნიტური დისკები, ცილინდრული მაგნიტური დომენების შემცველი მიკროსქემები, აგრეთვე ოპტიკური დისკები.

ნახევარგამტარულ დამხსომებელ მოწყობილობის ელემენტს საფუძვლად უდევს ტრიგერის ბისტაბულური სქემა, რომელიც შედგება გადაჯვარედინებული კავშირებიანი ლოგიკური ელემენტებისა და ინფორმაციის ჩაწერა/წაკითხისათვის განკუთვნილი წრედებისაგან. ასეთი ელემენტების ღირსებაა ის რომ როგორც მათი, ისე ამ ელემენტების ბაზაზე ორგანიზებული ელექტრონული მოწყობილობების (რეგისტრების, დეშიფრატორების და ა.შ.) დამზადება ერთიანი ტექნოლოგიური პროცესითაა შესაძლებელი.

დამხსომებელი ელემენტების მრავალფეროვნებით აიხსნება ის გარემოება, რომ მრავალფეროვანია როგორც ***ᲓᲘᲡ***-ებად (დიდ ინტგრალურ სქემებად) აღნიშნული ელემენტების, ისე დამგროვებლებად ***ᲓᲘᲡ***-ების გაერთიანების ხერხები.

დამხსომებელი ელემენტების კლასიფიცირება შეიძლება მოვახდინოთ ფიზიკო-ტექნოლოგიური, სქემოტექნიკური და სისტემოტექნიკური ნიშნებით. მოკლედ განვიხილოთ ისინი.

ა) ფიზიკო-ტექნოლოგიურად განასხვავებენ, ერთი მხრივ, ***ბიპოლარული*** ტრანზისტორებით, ხოლო, მეორე მხრივ, ლითონ-ჟანგეულ-ნახევარგამტარული ანუ ***ᲚᲫᲜ***-ტრანზისტორებით აგებულ დამხსომებელ ელემენტებს. პირველი სახის ტრანზისტორერები ***დანართ 3***-ში, ხოლო მეორე სახის ტრანზისტორები - ***დანართ 4***-ში გვაქვს დაწვრილებით განხილული აქ კი მათ ზოგადად დავახასიათებთ.

ბიპოლარულ ტრანზისტორებს საფუძვლად უდევს ნახევარგამტარულ ბლოკში მიმდინარე მოვლენები. აღნიშნული ბლოკი შედგება ***n*** და ***p*** ტიპის გამტარობის ურთიერთმონაცვლე ***3*** არესაგან. ამის შედეგად მასში წარმოქმნილია ორი ***p-n*** გადასასვლელი. გადასასვლელები ერთმანეთისაგან დაშორებულია მანძილით, რომელიც არაძირითადი მზიდების – ელექტრონებისა და «ხვრელების» დიფუზიურ სიგრძეზე ნაკლებია. ბიპოლარული ტექნოლოგიის გამოყენებით დამუშავებულია დამხსომებელი ელემენტების დიდი ჯგუფი,

რომლებიც აგებულია ერთემიტერული და მრავალემიტერული ტრანზისტორებით, ტირისტორებით, ინჟექციური სქემებით.

ლშნ-ტრანზისტორებს (***უნიპოლარულ ტრანზისტორებს***) საფუძვლად უდევს ორ ელექტროდს შორის მოქცეულ ნახევარგამტარული მასალის გამტარობის მართვის პრინციპი; მართვა ხორციელდება ელექტრული ველით, რის გამოც ***ლშნ***-ტრანზისტორებს ***ველის ტრანზისტორებსაც*** უწოდებენ. ***ლშნ***-ტრანზისტორი აქტიური ხელსაწყოა, რომელშიც დენს გრძივი ელექტრული ველის ზემოქამედებით წარმოქმნის მუხტების ძირითადი მზიდები, ხოლო დენის სიდიდეს მართავს მმართველ ელექტროდზე მოდებული განივი ელექტრული ველი. არსებობს ორი სახის, კერძოდ, ***n***– და ***p***–ტიპის არხებიანი ***ლშნ***-ტრანზისტორები. კრისტალში მათი ჩალაგების სიმჭიდროვე ***10⁶*** ბიტს აღწევს.

ბ) ***სქემოტექნიკურად*** დამხსომებელ ელემენტებს განასხვავებენ:

▲ მათში არსებული ტრანზისტორების რაოდენობის მიხედვით;

▲ ინფორმაციის ამისაღებად საჭირო სამისამართო და თანრიგობრივი სალტების რაოდენობისა და ფუნქციების მიხედვით (ჩაწერის, წაკითხვისა და კვების ძაბვის მიწოდების შეთასებული ან დანაწევრებებული ფუნქციები);

▲ ტრიგერში არსებული სადატვირთო წინაღობების ხასიათის მიხედვით (წრფივი, არაწრფივი, გადართვადი დატვირთვიანი დამხსომებელი ელემენტები);

▲ თანრიგობრივ სალტეებთან დამხსომებელი ელემენტების კავშირის ხასიათის მიხედვით (უშუალო ან საგასაღებო ელემენტის მეშვეობით განხორციელებული კავშირებიანი დამხსომებელი ელემენტები).

გ) ***სისტემოტექნიკურად*** დამხსომებელ მოწყობილობებს განასხვავებენ:

▲ შენახვის პრინციპის მიხედვით (სტატიკური, დინამიკური და კვაზისტატიკური დამხსომებელი ელემენტები);

▲ წაკითხვის პრინციპის მიხედვით (დამხსომებელი ელემენტები, რომლებიდანაც წაკითხვის შემდეგ არ ნადგურდება წაკითხული ინფორმაცია ან აღნიშნული ინფორმაცია ნადგურდება და საჭირო ხდება მისი აღდგენა).

▲ დამხსომებელი ელემენტებიდან წაკითხული სიგნალის ფორმისა და სახის მიხედვით (დამხსომებელი ელემენტები, რომლებიდანაც წაიკითხება ერთპოლარული, ორპოლარული, პარაფაზული ***ფორმის***, იმპულსური დენის ან იმპულსური ძაბვის ***სახის*** სიგნალები).

V თავი
მიკროპროცესორის ანატომია

5.1. პროცესორის აგების პრინციპები

მიკროპროგრამული მართვის პრინციპი. ციფრულ სისტემებში (მოწყობილობებში) ინფორმაციაზე ოპერაციებს ასრულებს ***მიკროპროცესორული მართვის პრინციპის*** გამოყენებით აგებული ***პროცესორი.*** აღნიშნული პრინციპი შემდეგი ხუთი დებულებით გამოიხატება:

1) ინფორმაცია წარმოიდგინება ცალკეული ***სიტყვების*** (ორობითი კოდების) სახით, რომლებზეც გარკვეული სახის ოპერაციებს ასრულებს პროცესორი;

2) საინფორმაციო სიტყვებზე (ორობით კოდებზე) პროცესორის მიერ ჩასატარებელი ნებისმიერი ოპერაცია წარმოადგენს რთულ ოპერაციას; იგი შედგება მიმდევრობითად შესასრულებელ ელემენტალურ მოქმედებებისაგან, რომლებსაც ***მიკროპერაციები*** ეწოდება.

3) მიკროოპერაციების შესრულების თანამიმდევრობის დასაცავად გამოიყენება ***ლოგიკური პირობები***; ისინი ლოგიკური ***1***-ის ან ***0***-ის სახით ასახავს ***პროცესორის მდგომარეობას***, რომელშიც იგი გადადის ყოველი მიკროოპერაციის შესრულების შემდეგ.

4) პროცესორში ოპერაციების შესრულების პროცესი აღიწერება მიკროოპერაციებისა და ლოგიკური პირობების ტერმინების მეშვეობით წარმოდგენილი ***ალგორითმის*** ფორმით, რომელსაც ***მიკროპროგრამა*** ეწოდება;

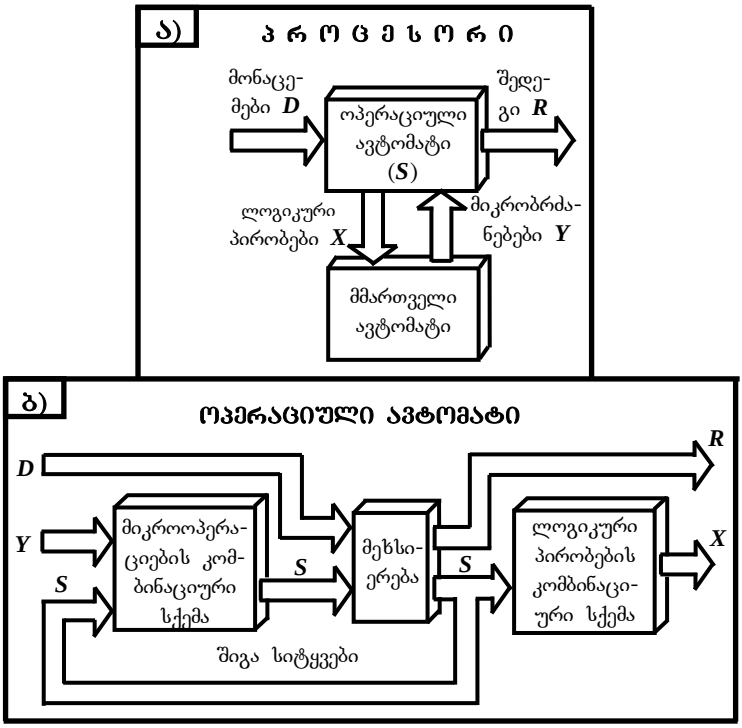
5) მიკროპროგრამა გამოიყენება პროცესორის ფუნქციის წარმოდგენის ფორმად, რომლის საფუძველზეც განისაზღვრება ***პროცესორის სტრუქტურა*** და ***დროში მისი ფუნქციონირების წესი***.

პროცესორის სტრუქტურა. სტრუქტურულ-ფუნქციონურად პროცესორი იყოფა ორ ნაწილად: ოპერაციულ და მმართველ ავტომატად (ნახ.***5.1***).

ოპერაციული ავტომატი განკუთვნილია: ***1)*** შესასვლელი (***D***), გამოსასვლელი (***R***) და შინაგანი (S) სიტყვათა სიმრავლის შესანახად;

2) ***R*** შედეგის მიღებისათვის საჭირო მიკროოპერაციათა ნაკრების შესასრულებლად; ***3)*** მაუწყებელი სიგნალების ***X*** სიმრავლის ფორმირებისათვის, რომლის თითოეული ელემენტი გარკვეულ ლოგიკურ პირობებთან იგივდება.

ოპერაციული ავტომატით რეალიზებადი მიკროოპერაციების ინიციცირებას ახდენს მმართველი $Y = \{y_1 \ldots y_N\}$ სიგნალების სიმრავლე, რომლის ყოველი წევრი შეესაბამება გარკვეულ მიკროოპერაციას.



ნახ. 5.1. *პროცესორის სტრუქტურა (**ა**), ოპერაციული ავტომატის სტრუქტურა (**ბ**)*

ოპერაციული ავტომატი თავის მხრივ შედგება (ნახ.***5.1,ბ***) ***მეხსიერებისა*** და ***ორი კომბინაციური სქემისაგან***, რომელთაგანაც ერთი წარმოქმნის ***მიკროოპერაციებს,*** ხოლო მეორე – ***ლოგიკურ პირობებს***.

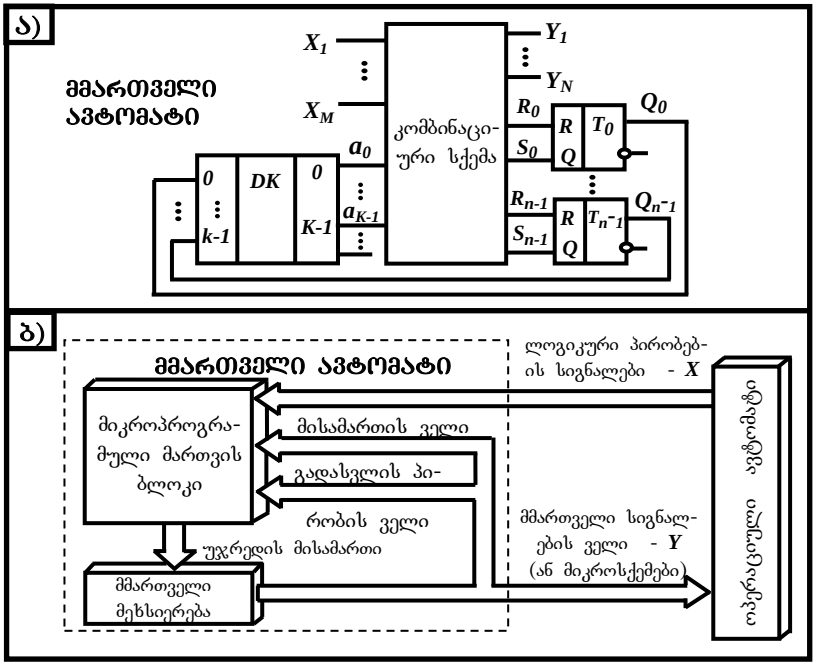
მმართველი ავტომატი (იხ.ნახ.***5.1,ა***) წარმოქმნის მიკროპროგრამით განსაზღვრულ და ლოგიკურ ***X*** პირობათა მნიშვნელობების შესაბამის მმართველ სიგნალებს, რომლებიც შედის ***Y*** სიმრავლეში. მიკროპროგრამების პაკეტის შესრულებისას მიკროპროცესორის შესასვლელებს მიმდევრობით მიეწოდება ამა თუ იმ მიკროპროგრამის შესაბამისი ოპერაციების კოდები. აღნიშნულ შესასვლელებს შეიძლება ლოგიკური პირობების გარე სიგნალებიც მიეწოდოს, ხოლო მისი გამოსასვლელებიდან მიღებული იქნეს გარე მოწყობილობის მმართველი სიგნალები.

მმართველი ავტომატის სტრუქტურა მნიშვნელოვნადაა დამოკიდებული იმაზე, თუ რა პრინციპითაა აგებული მოცემული ავტომატი. ***სქემური ლოგიკის პრინციპის*** გამოყენებით აგებულ მმართველ ავტომატებში მმართველი ***Y*** სიგნალების საჭირო მიმდევრობას გამოიმუშავებს ლოგიკური ელემენტებისაგან აგებული სქემა, რომელსაც უმეტესწილად აქვს მცირე და საშუალო ხარისხის ინტეგრაციის მქონე ინტეგრალური მიკროსქემის სახე. ***დაპროგრამებადი ლოგიკის პრინციპის*** გამოყენებისას მმართველი ***Y*** სიგნალები გამომუშავდება მუდმივ დამხსომებელ მოწყობილობებში ან დაპროგრამებად ლოგიკურ მატრიცებში შენახული მიკროპროგრამის მეშვეობით.

განასხვავებენ ***სპეციალიზებულ*** და ***უნივერსალურ*** პროცესორებს. პირველი მათგანი განკუთვნილია გარკვეული არითმეტიკული თუ ლოგიკური ოპერაციების შესასრულებლად, ხოლო მეორე მათგანს აქვს ფუნქციონირების მეტად ფართო სფერო. მოცემულ პარაგრაფში ყურადღებას ***სპეციალიზებულ პროცესორებზე*** გავამახვილებთ, ხოლო მომდევნო პარაგრაფში უფრო ვრცლად განვიხილავთ უნივერსალური პროცესორების ფუნქციონირების საკითხებს.

სქემური ლოგიკის პრინციპის გამოყენებით მმართველი ავტომატის სინთეზის პროცესი რამდენიმე ეტაპისაგან შედგება. ***პირველ ეტაპზე*** აიგება იმ ოპერაციათა ალგორითმების გრაფ-სქემები, რომელებიც უნდა მართოს აღნიშნულმა ავტომატმა. ***მეორე ეტაპზე*** აიგება მმართველი ავტომატის გადასვლების გრაფი, რომელიც განსაზღვრავს მმართველი ავტომატის ფუნქციონირების კანონსა და სტრუქტურას. ამისათვის ფიქსირდება და კოდირდება ავტომატის მდგომარეობები: ავტომატის თითოეულ მდგომარეობას შეუთანადდება ტრიგერების ***Q***-გამოსასვლელთა მდგომარეობებით წარმოქმნილი გარკვეული კოდური

კომბინაცია. კოდის თანრიგების n რაოდენობა ემთხვევა ტრიგერების რაოდენობას და აკმაყოფილებს $K \le 2^n$ პირობას, სადაც K ავტომატის მდგომარეობათა რაოდენობაა.



ნახ. 5.2. სქემური ლოგიკისა (ა) და დაპროგრამებადი ლოგიკის (ბ) მმართველი ავტომატის სტრუქტურული სქემა

მმართველი ავტომატის სტრუქტურული ავტომატის სქემა (ნახ. *5.2,ა*) შეიცავს:

1) $T_1,\ldots,T_{n-1}$ ტრიგერებს, რომლებიც წარმოქმნის გამოსასვლელი $Q_0,\ldots,Q_{n-1}$ სიგნალების დახმარებით ავტომატის მიმდინარე მდგომარეობის ფიქსირებისათვის საჭირო ინფორმაციის შემნახველ რეგისტრს;

2) **DC** დეშიფრატორს, რომელიც n-თანრიგიანი $Q_0,\ldots,Q_{n-1}$ კოდს გარდაქმნის ავტომატის მდგომარეობათა $a_0,\ldots,a_{K-1}$ სიგნალებად;

3) კომბინაციურ სქემას, რომელიც შესასვლელი $X_1,\ldots,X_M$, $a_0,\ldots,a_{K-1}$ სიგნალებით გამოიმუშავებს ოპერატიული ავტომატისათვის

განკუთვნილ $Y_1,\ldots,Y_N$, ხოლო ტრიგერებისათვის – $S_0,\ldots,S_{n-1}$, $Q_0,\ldots,Q_{n-1}$ სიგნალებს.

ამის შემდეგ დგება თითოეული განსახილველი ალგორითმისათვის მმართველი ავტომატის კომბინაციური სქემის აგებულების განსზღვრის ამოცანა, რომლის განხილვა სცდება ჩვენი წიგნის ჩარჩოებს. ხაზს გავუსვამთ მხოლოდ იმ ფაქტს, რომ ***სქემური ლოგიკის მქონე მმარ-თველი ავტომატი*** ოპერაციული ავტომატის მუშაობისათვის აუცილე-ბელი მმართველი სიგნალების მიმდევრობის ფორმირებას აპარატურუ-ლი საშუალებების დახმარებით ახდენს.

რამდენიმე სიტყვით შევეხოთ სხვა პრინციპით აგებულ მმართვე-ლი ავტომატს, რომლის დროსაც უწყვეტ მმართველ სიგნალებს წაროშობს მმართველი მეხსიერების უჯრედებში შენახული მიკრო-პროგრამა.

მმართველი სიგნალების $Y = (y_1, y_2,\ldots)$ ერთობლიობა თითოეულ ტაქტურ პერიოდში წარმოქმნის ***მიკრობრძანებას***. გარკვეული ოპერა-ციის შესასრულებლად განკუთვნილი მიკრობრძანებების მიმდევრობას ***მიკროპროგრამა*** ეწოდება. ასეთ შემთხვევაში ***ოპერაციის შესრულება*** დაიყვანება მმართველი მეხსიერებიდან მიკროპროგრამის შემადგენელი მიკრობრძანებების ამოღებასა და მმართველი Y სიგნალების დახმარე-ბით ოპერაციული ავტომატისათვის მის გადაცემამდე. ***მმართველ მეხ-სიერებაში*** შეინახება სხვადასხვა ოპერაციების შესრულებისათვის განკუთვნილი მრავალი მიკროპროგრამა. ესა თუ ის მიკროპროგრამა ამოირჩევა ოპერატიული მეხსიერებიდან შემოსული ბრძანების დახმა-რებით. ამორჩეული მიკროპროგრამა რეალიზდება მმართველი მეხსი-ერების უჯრედებიდან მიკრობრძანების მიკრობრძანებების მიმდევრო-ბითი წაკითხვის გზით. მმართვის ასეთი პრინციპის დროს თითო-ეული ტაქტის დროს განისაზღვრება მმართველი მეხსიერების იმ უჯრედის მისამართი, რომლიდანაც უნდა მოხდეს მიკროპროგრამის მორიგი მიკრობრძანების წაკითხვა. მიკროპროგრამის მიკრობრძანება შეიცავს მთელ რიგ ***ველებს***. თითოეულ ველს ეთმობა გარკვეული რაოდენობის თანრიგები. ველების ერთობლიობას ეწოდება ***მიკრო-ბრძანების ფრაგმენტი.*** მიკროპროგრამის შემადგენელი მიკრობრძანებ-ის ფორმატში, როგორც წესი, გაითვალისწინება შემდეგი სამი სახის ველი: ***1)*** მმართველი სიგნალების ველი, რომელიც ეთმობა ოპერაცი-ული ავტომატის მართვისათვის განკუთვნილ Y მიკრობრძანებას; ***2)***

გადასვლის პირობის ველი, რომელშიც მიეთითება მოცემულ მიკრობრძანებაზე გადასვლა ხდება რაიმე პირობის შესრულებისას თუ ნებისმიერ შემთხვევაში; პირველ შემთხვევის დროს გვაქვს ე.წ. ***პირობითი გადასვლა***, ხოლო მეორე შემთხვევის დროს კი – ***უპირობო გადასვლა***. პირობითი გადასვლის დროს განისაზღვრება ***ლოგიკური პირობა*** X_i, რომლის შესრულებისას ხდება გადასვლა; ***3) მისამართის ველი***, რომელშიც მიეთითება მიკროპროგრამაში შემავალი სასურველი მიკრობრძანების ***საორიენტაციო მისამართი***. ზოგადად მისამართი ლოგიკურ პირობებზეა დამოკიდებული. გადასვლის სახესა და ლოგიკური პირობის შესრულება-არშესრულებაზე დამოკიდებულით ნარჩუნდება ან მოდიფიცირდება (იცვლება) ზემოთ აღნიშნული საორიენტაციო მისამართი.

დაპროგრამებადი ლოგიკის მქონე მმართველი ავტომატის განზოგადებული სტრუქტურა ***5.2,ბ*** ნახაზზეა გამოსახული. იგი მმართველი მეხსიერების გარდა შეიცავს მრიგი მიკრობრძანების წარმომქმნელი ფუნქციის შემსრულებელ მიკროპროგრამული მართვის ბლოკს.

მისამართის ველების მდგომარეობაზე, მიმდინარე მიკრობრძანების გადასვლის პირობასა და ოპერაციული ავტომატის მიერ გაცემული ლოგიკური პირობების სიგნალთა მნიშვნელობაზე დამოკიდებულებით მიკროპროგრამული მართვის ბლოკში ფორმირდება მეხსიერების იმ უჯრედის მისამართი, რომელშიცაა შენახული შესასრულებელი მიკროპროგრამის მორიგი მიკრობრძანება. მომდევნო ტაქტურ პერიოდში მიკრობრძანება ამოიკითხება მმართველი მეხსიერებიდან. მმართველ სიგნალთა ველის თანრიგები გადაეცემა ოპერაციულ ავტომატს, ხოლო მისამართის ველისა და გადასვლის პირობათა ველის თანრიგები – მიკროპროგრამული მართვის ბლოკს; ოპერაციული ავტომატი შეასრულებს მოცემულ Y_i მიკრობრძანებას, ხოლო მიკროპროგრამების მართვის ბლოკი ჩამოაყალიბებს მომდევნო მიკრობრძანების მისამართს. პროცესი გაგრძელდება მიკროპროგრამის მთლიანად შესრულებამდე. ვინაიდან მმართველი ავტომატის სტრუქტურა სტანდარტულია, ამიტომ კონსტრუქტორების მთელი ძალისხმევა გადატანილია მუდმივ დამხსომებელ მოწყობილობაში ჩასაწერი მიკროპროგრამის შედგენაზე, რომლის განხილვაც სცდება ჩვენი წიგნის ჩარჩოებს.

5.2. უნივერსალური 8-თანრიგიანი პროცესორის ზოგადი სტრუქტურა

ოპერაციული და მმართველი ავტომატების სახით წარმოდგენილი ***სპეციალიზებული პროცესორების*** სტრუქტურა, რომელიც წინა პარაგრაფში განვიხილეთ, თვალსაჩინო წარმოდგენას გვაძლევს გარკვეული ოპერაციების შესასრულებლად განკუთვნილი პროცესორების აგებისა და ფუნქციონირების პრინციპების შესახებ. მათგან განსვავებით ***უნივერსალური პროცესორები*** გაცილებით ბევრ სხვადასხვა ციფრულ კვანძებსა და ინფორმაციის გასაცვლელად განკუთვნილ არხებს შეიცავს და ამიტომ მათ გაცილებით ფართო ფუნქციური შესაძლებლობები აქვს. მიკროპროცესორების აგებულებისა და ინფორმაციის მიკროპროცესორული დამუშავების უფრო ნათლად გაგებისათვის მიზანშეწონილია განვიხილოთ უმარტივესი ***8***-თანრიგიანი პროცესორი (ნახ.***5.3***).

პროცესორების მათი დანიშნულებაა ბრძანებათა საკუთარი სისტემით გათვალისწინებული ოპერაციების შესრულება. პროგრამის რეალიზებისათვის მიკროპროცესორული სისტემის ცენტრალურმა პროცესორმა უნდა შეასრულოს შემდეგ ფუნქციები:

1) ოპერატიულ მეხსიერებაში შენახული ბრძანებებისა და მონაცემების ***მისამართის ფორმირება***;

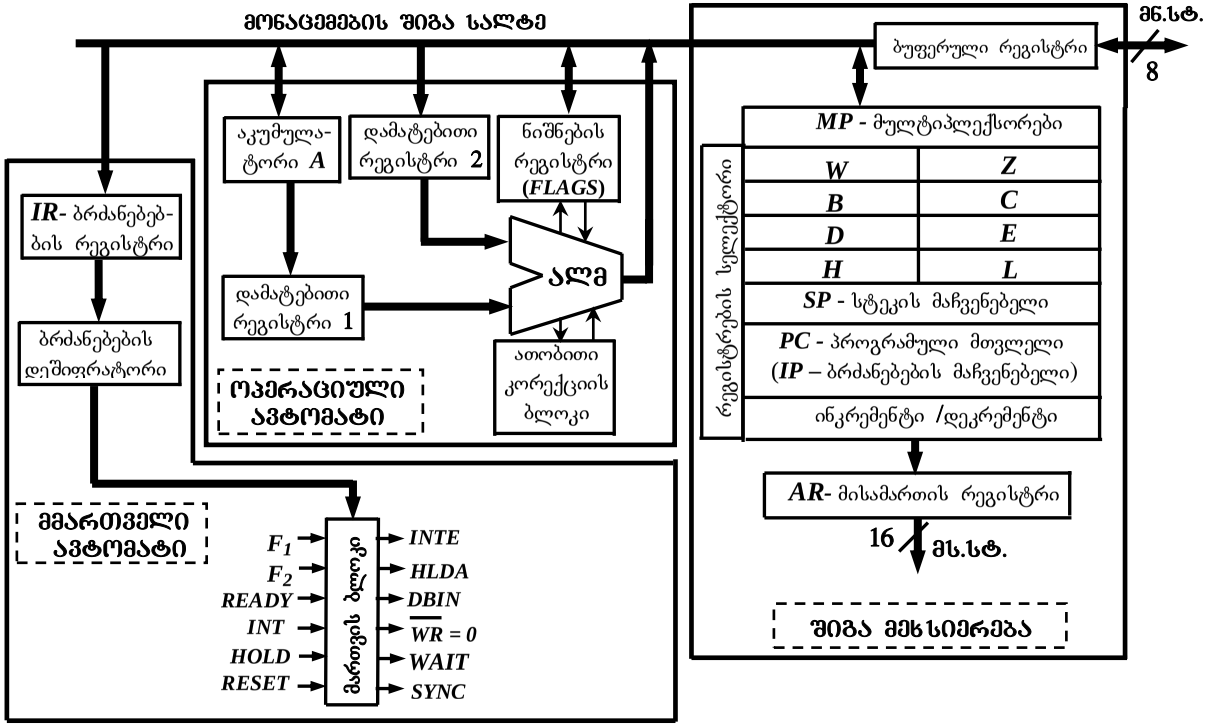
2) მეხსიერებიდან ბრძანებების ***ამოკრება*** და მათი ***გაშიფვრა***;

3) ძირითადი (ოპერატიული) მეხსიერებიდან ***მონაცემების მიღება***, მათზე ბრძანების კიდით განსაზღვრული არითმეტიკული, ლოგიკური და სხვა ***ოპერაციების ჩატარება*** და გარე მოწყობილობებისათვის ან მეხსიერებისათვის დამუშავებული ***მონაცემების გადაცემა***;

4) შინაგანი კვანძების, გარე მოწყობილობებისა და მეხსიერების ნორმალური ფუნქციონირებისათვის საჭირო მდგომარეობის (ლოგიკური პირობის), მმართველი და დროითი ***სიგნალების ფორმირება***;

5) შესრულებულ ოპერაციათა შედეგების, მისამართების, მდგომარეობის ფორმირებული სიგნალებისა და სხვა მონაცემების ***დროებითი შენახვა***;

6) გარე მოწყობილობებიდან შემოსული მოთხოვნის სიგნალების მიღება და მათი ***მომსახურება***.



ნახ.5.3.უნივერსალური 8-თანრიგიანი პროცესორის სტრუქტურული სქემა

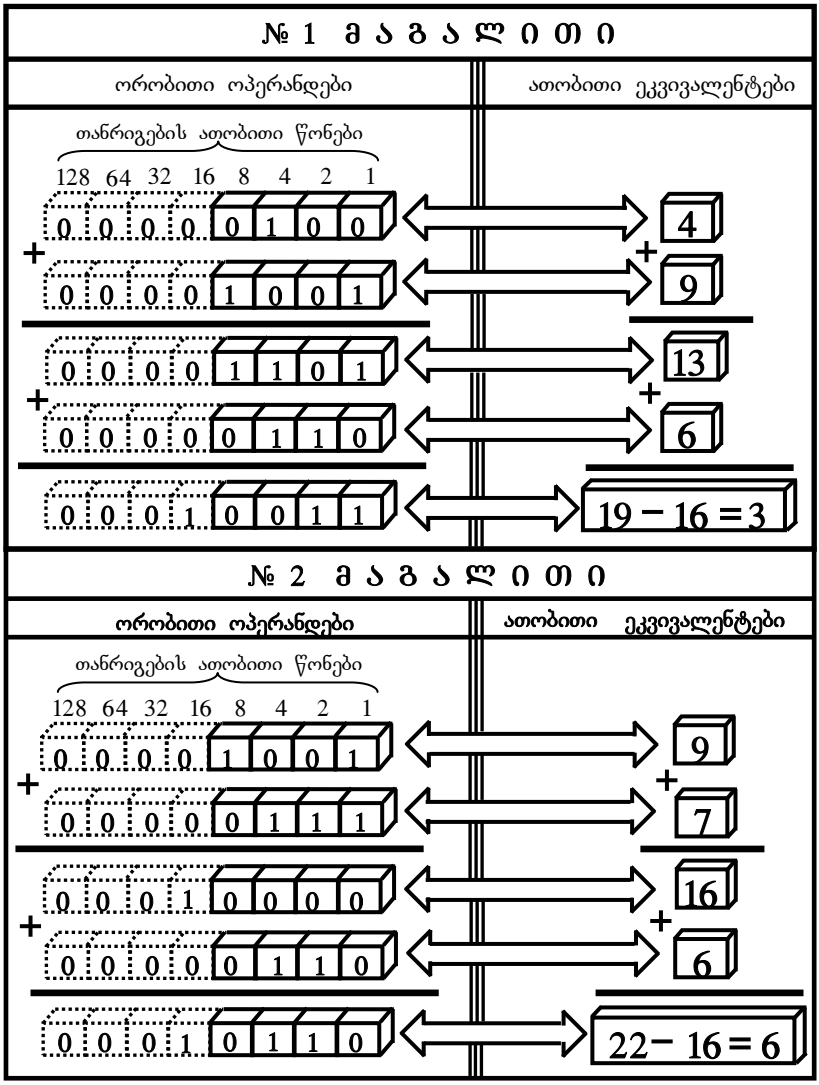
პროცესორის აგებულება შევისწავლოთ ***5.3*** ნახაზზე მოყვანილი სტრუქტურული სქემის მაგალითზე. აღნიშნული ნახაზიდან ნათლად ჩანს, რომ პროცესორი ***შემადგენელი ძირითადი ნაწილებია: 1) ოპერაციული ავტომატი,*** რომლიც შეიცავს ***ალმ***–ით აღნიშნულ არითმეტიკულ-ლოგიკურ მოწყობილობასა და მის მომსახურე რეგისტრებს (***A*** აკუმულატორსა და ორ დამატებით რეგისტრს), ათობითი კორექციის ბლოკს და ლოგიკური პირობების მაფორმირებელ ნიშნების რეგისტრს; ***2) მმართველი ავტომატი,*** რომელიც შეიცავს ბრძანებების რეგისტრს (***I****nstruction* ***R****egistr-****IR***), დეშიფრატორსა და მართვის ბლოკს; ***3) შიგა მეხსიერება,*** რომელიც შეიცავს საერთო დანიშნულების (***W,Z,B,C,D,E,H,L***) რეგისტრებს, სტეკის მაჩვენებელს (***S****tack* ***P****ointer-****SP***), პროგრამულ მთვლელს (***P****rogram* ***C****ounter–****PC***; მას ბრძანებების მაჩვენებელს – ***P****ointer of* ***I****nstruction*-ს, ანუ ***PC***-საც უწოდებენ), მისამართის რეგისტრსა (***A****ddress* ***R****egister-****AR***) და ბუფერულ რეგისტრს; ***4) კომუნიკაციის საშუალებები,*** რომლებსაც მიეკუთვნება შიგა სალტეები, მონაცემების სალტე (***მ.ს.ტ***), მისამართის სალტე (***მს.ს.ტ***) და მართვის სალტე.

მოკლედ განვიხილოთ მიკროპროცესორის თითოეული სტრუქტურული ერთეული.

<u>1.არითმეტიკულ-ლოგიკური მოწყობილობა ანუ ალმ</u> იგი საშუალებას გვაძლევს ***8***-თანრიგიან ოპერანდებზე შევასრულოთ შემდეგი ოპერაციები: ***ა)*** ორი ორობითი ოპერანდის არითმეტიკული შეკრება, რომლის დროსაც გადატანა გადაეცემა ან არ გადაეცემა უფროს თანრიგს და გამოკლება, რომლის დროსაც სესხი გადმოიტანება ან არ გადმოიტანება უმცროს თანრიგში; ***ბ)*** დიზიუნქცია, კონიუნქცია, ორის მოდულით შეკრება და შედარება; ***გ)*** ოთხი სახის ციკლური ძვრა (იხ.***§2.11***); ***დ***) არითმეტიკული ოპერაციები ათობით რიცხვებზე.

ოპერაციების შესრულებისას ***ალმ***-ს ერთ-ერთი ოპერანდი მიეწოდება ***A აკუმულატორისა*** და ***დამტებით რეგისტრ 1***-ის, ხოლო მეორე ოპერანდი – ***დამატებით რეგისტრ 2***-ის გავლით. ციკლური ძვრები სრულდება მხოლოდ ***A აკუმულატორის*** შიგთავსზე. აკუმულატორშივე თავსდება ***ალმ***–ში შესრულებულ ოპერაციათა შედეგები.

<u>2.ათობითი კორექციის ბლოკი</u> გამოიყენება პროცესორის მიერ ათობითი რიცხვების შეკრების პროცესში წამოჭრილი კორექტირების



ნახ.5..4. *ათობითი კორექციის ორი მაგალითი*

პროცესის რეალიზებისათვის. ათობითი რიცხვების შესაკრებად ათობითი რიცხვის თითოეული ციფრი, როგორც წესი, წარმოიდგინება ***8421*** კოდის (იხ.ნახ.***2.7,ა***) ოთხნიშნა სიტყვებით (ნახევარბაიტებით,

ტეტრადებით). ნახევარბაიტები არითმეტიკის წესებით იკრიბება. კორექციის აუცილებლობა მაშინ წამოიჭრება, როდესაც ჯამი ***9***-ზე მეტია. კორექტირებისათვის მიღებულ ჯამს უნდა დაემატოს ათობითი რიცხვ ***6***-ის შესაბამისი ორობითი ***0110*** რიცხვი. ამას განაპირობებს ის გარემოება, რომ ხუთნიშნა ორობითი რიცხვის უფროსი (მეხუთე) თანრიგის წონაა ***16*** ათობითი ერთეული, ხოლო ორნიშნა ათობითი რიცხვის უფროსი თანრიგის წონა ***10***-ის ტოლია, ე.ი. აღნიშნული წონების სხვაობა ***6***-ის ტოლია.

შეკრების შედეგად თუ მიიღება ***10***-დან ***15***-მდე რიცხვი, მაშინ 6_{10} (0110_2) რიცხვის მიმატება მე-***5*** თანრიგში გააჩენს ***1***-ს. ეს ბიტი უფროს ნახევარბაიტში გადასვლით «წაიღებს დანამატს, ე.ი.***6***-ს» და დატოვებს სწორ შედეგს. კორექციის ეს შემთხვევა (როდესაც შეკრების შედეგად მიღებულია რიცხვი ***13***) ***5.4,ა*** ნახაზზეა ილუსტრირებული.

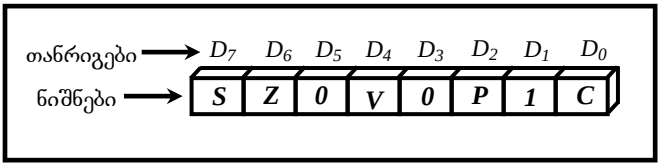
შეკრების შედეგად თუ მიიღება ***16***-დან ***18***-მდე რიცხვი, მაშინ მე-***5*** თანრიგში ჩნდება ***1,*** რომლის მნიშვნელობა (გადატანამდე) შეესაბამება ათობით რიცხვ ***16***-ს. უფროს ნახევარბაიტში ამ ***1***-ანის გადატანის შემდეგ მისი მნიშვნელობა ხდება ***10***-ის ტოლი, ე.ი. ***6*** ასთობითი ერთეულით მცირდება, რაც მოითხოვს მის კორექტირებას (ე.ი. ***6***-ის მიმატებას). კორექციის ეს შემთხვევა (როდესაც შეკრების შედეგად მიღებულია რიცხვი ***16***) ***5.4,ბ*** ნახაზზეა ილუსტრრებული

<u>3.ნიშნების (ალმების) რეგისტრი.</u> ***ალმ*** უშუალოდაა დაკავშირებული ***ნიშნების (ალმების) 5-თანრიგიან რეგისტრთან***, რომელშიც ფიქსირდება ზოგიერთი არითმეტიკული და ლოგიკური ოპერაციების შესრულების შედეგები. იგი შეიცავს შემდეგ ***5*** ტრიგერს: ***ა) გადატანის ტრიგერს,*** რომელიც გამოიმუშავებს ***C=1*** სიგნალს შეკრების ან ძვრის ოპერაციების შესრულებისას უფროსი თანრიგიდან ***1***-ის გადატანის წარმოშობისის დროს; ***ბ) დამატებითი გადატანის ტრიგერს***, რომელიც გამოიმუშავებს ***V=1*** სიგნალს ორობით-ათობით კოდებზე ოპერაციის ჩატარებისას მესამე (ე.ი. უმცროსი ნახევარბაიტის უფროსი) თანრიგიდან გადასატანი ***1***-ის წარმოშობის დროს; ***გ) ნულის ტრიგერს***, რომელიც გამოიმუშავებს ***Z=1*** სიგნალს ოპერაციის შედეგად ***0***-ის მიღების დროს; ***დ) ნიშნის ტრიგერს***, რომელიც გამოიმუშავებს ***S=1*** სიგნალს დამატებით კოდით წარმოდგენილი ოპერანდის უფროსი თანრიგის ***1***-ზე ტოლობის, ე. ი. უარყოფითი რიცხვის

მიღების დროს; ***ე) ლუწობის ტრიგერს***, რომელიც გამო-იმუშავებს ***P=1*** სიგნალს ოპერაციის შედეგში ლუწი რაოდენობის ***1***-ის არსებობის დროს.

ჩამოთვლილი ტრიგერები პროგრამაში უზრუნველყოფს პირობით გადასვლებს. მაგალითად, თუ წინა ოპერაციის შესრულების შედეგი ***0***-ის ტოლია, მაშინ ნულის ტრიგერი გადადის მდგომარეობა ***1***-ში (***Z=1***) და მოხდება პროგრამის სხვა ნაწილზე გადასვლა. მონაცემების სალტით გადაცემის დროს ბაიტში ***ნიშნების რეგისტრის*** თანრიგების განაწილება ***5.5*** ნახაზზეა მოყვანილი.

4.რეგისტრები. რეგისტრებთან, მათ შორის ***ბრძანებების მთვლელსა*** და ***სტეკის მაჩვენებელთან***, შეღწევა ***რეგისტრების სელექტორის*** დახმარებით ***მულტიპლექსორებითაა*** შესაძლებელი.



ნახ.5.5. ბაიტში ნიშნების რეგისტრების თანრიგების განაწილება

დასამუშავებელი მონაცემების შენახვისას საერთო დანიშნულების რეგისტრები თამაშობს ***აკუმულატორების***, ხოლო ოპერანდთა მისამართების შენახვისას – ***მაჩვენებლების*** როლს. პროგრამის შესრულებისას ***B,C,D,E,H,L*** რეგისტრები შეიძლება გამოვიყენოთ როგორც დამოუკიდებელ ***8***-თანრიგიან რეგისტრებად, ასევე ***16***-თანრიგიანი რეგისტრულ ***BC, DE, HL*** წყვილებადაც. მათ მოიხსებიებენ ***16***-თანრიგიან ***B,D,H*** რეგისტრის სახელითაც, რომელშიც შენახულია ***16***-თანრიგიანი რიცხვის პირველი ბაიტი. ***16***-თანრიგიანი ***H*** რეგისტრი გამოიყენება ***სამისამართო რეგისტრადაც***: ირიბი რეგისტრული დამისამართების დროს იგი ინახავს ძირითადი მეხსიერებიდან შემომავალ ***შემსრულებლობით მისამართს***.

H და ***Z*** რეგისტრები პროგრამულად მიუწვდომელია და მხოლოდ შიგა მიკროპროცესორული ბრძანებებისათვის გამოიყენება. მათში შეინახება ბრძანების მეორე და მესამე ბაიტები

მიკროპროცესორსა და გარე მოწყობილონებს შორის ინფორმაცია გაიცვლება ორმხრივმიმართული ***ბუფერული რეგისტრით***, ხოლო მეხსიერება და გარე მოწყობილობები დამისამართდება ***16***-თანრიგიანი

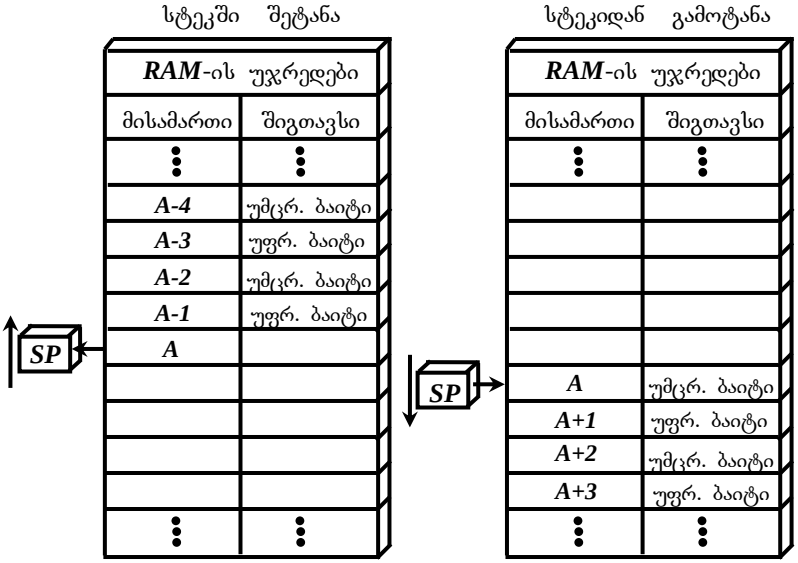
მისამართის რეგისტრის მეშვეობით. ბუფერული და მისამართის რეგისტრების თავისებურებაა ის, რომ გარდა ლოგიკური ***0***-ისა და ***1***-ის ტოლი მდგომარეობებისა მათში გათვალისწინებულია მესამე მდგომარეობაც, რომლის დროსაც უსასრულოდ დიდია რეგისტრების გამოსასვლელი წინაღობა. ამ მდგომარეობაში მიკროპროცესორის არსებობისას გარე მოწყობილობები მუშაობს ***მეხსიერებასთან პირდაპირი დაშვების*** რეჟიმში.

<u>5. პროგრამული მთვლელი</u> (***P**rogram **C**onter* – ***PC***), ანუ ***პრძანებათა მაჩვენებელი*** (***P**ointer of **I**nstruction* – ***PI***) განკუთვნილია მეხსიერებაში იმ უჯრედის მისამართის მისათითებლად, რომელშიც ბრძანების მორიგი ბაიტია განთავსებული (ბრძანებებისათვის გამოიყენება ***3***-ბაიტური ფორმატი). ამიტომ ბრძანების ყოველი მორიგი ბაიტის ამორჩევის შემდეგ ***ინკრემენტ-დეკრემენტის სქემა*** პროგრამული მთვლელის შიგთავსს ზრდის ერთი ერთეულით. რომელიმე ბრძანების ამოკრების წინ ბრძანებების მთვლელში შეიტანება მისი პირველი ბაიტის მისამართი. ***3***-ბაიტური ბრძანების ამოკრებისას პროგრამული მთვლელის შიგთავსი სამჯერ იზრდება. მისამართების ცვლილების ჩვეულებრივი მიმდევრობა შეიძლება შეიცვალოს. ამისათვის გათვალისწინებულია პროგრამის იმ ნაწილის საწყისი მისამართის შეტანის შესაძლებლობა, რომელიც მოცემულ მომენტში უნდა შესრულდეს. პროგრამის ამ ნაწილს ***ქვეპროგრამა*** ეწოდება.

<u>6. სტეკის მაჩვენებელი</u> (***S**tack **P**ointer-* ***SP***) მეხსიერების ***სტეკად*** წოდებული განსაკუთრებული ნაწილის სწრაფად დამამისამართებელი ***16***-თანრიგიან რეგისტრია. ***სტეკური მეხსიერება*** კომპიუტერის ძირითადი მეხსიერების განსაკუთრებული სახეა, რომელიც შეწყვეტების მომსახურებისათვის გამოიყენება. მიკროპროცესორის მიერ ქვეპროგრამის დამუშავების პერიოდში სტეკში შეინახება შეწყვეტილ მეხსიერებაში დაბრუნების მისამართი, აგრეთვე აკუმულატორისა და ნიშნების რეგისტრის შიგთავსები.

სტეკთან გაცვლა ხდება ***2***-ბაიტური სიტყვებით, რომელთა შესანახად გამოიყენება ორი მეზობელი უჯრედი. სტეკის ***SP*** მაჩვენებელში ***A*** მისამართის შენახვის შემთხვევისათვის სტეკით გაცვლის თავისებურება ***5.6*** ნახაზზეა ილუსტრირებული. სტეკში ინფორმაციის შეტანისას ***SP*** რეგისტრის შიგთავსი ***1***-ით მცირდება და ***A-1*** მისამართის მქონე უჯრედში ჩაიწერება უფროსი ბაიტი; ამის შემდეგ ხე-

ლახლა ***1***-ით მცირდება ***SP*** რეგისტრის შიგთავსი და ***A-2*** მისამამართის მქონე უჯრედში ჩაიწერება უმცროსი ბაიტი და ა.შ. სტეკის მიერ დაკავებულ უკანასკნელი უჯრედის ***A*** მისამართი შენახულია სტეკის ***SP*** მაჩვენებელში. სტეკიდან ინფორმაციის გამოტანა იწყება იმ უჯრედიდან, რომლის მისამართი ***SP***-შია შენახული (ჩვენს შემთხვევაში იქ ინახება ***A*** მისამართი). პირველი უმცროსი თანრიგის გამოტანის შემდეგ ***SP***–ს შიგთავსი ***1***-ით იზრდება და ***A-1*** მისამართის მქონე უჯრედიდან გამოიტანება უფროსი ბაიტი. უფროსი ბაიტის წაკითხის შემდეგ ხელახლა ***1***-ით იზრდება ***SP***–ს შიგთავსი და ა.შ. ე.ი. ***SP*** არ ჩერდება წაკითხულ უჯრედზე, იგი მიუთითებს სტეკის მწვერვალს. ამგვარად სტეკური მეხსიერება მუშაობს პრინციპით – ***პირველად ამოიღება უკანასკნელად შესული მონაცემები.***



ნახ.5.6. *სტეკში შესრულებადი ოპერაციები*

7. მმართველი ავტომატი. შესასრულებელი ბრძანების პირველი ბაიტი ***ბრძანებების რეგისტრში*** ჩაიწერება. ***ბრძანებების დეშიფრატორში*** ფორმირდება სიგნალები, რომელთა მოქმედებით ***მართვის ბლოკში*** ამუშავდება მოთხოვნილი ოპერაციის შემსრულებელი მიკროპროგრამა. ბრძანებების ნაკრებით განსაზღვრული ოპერაციის მიკროპრო-

გრამები მუდმივ მეხსიერებაშია "ჩაკერებული". ***მართვის ბლოკის*** შესასვლელებს მიეწოდება: ***ა) T*** პერიოდის მქონე ტაქტური F_{1}, F_1 იმპულსების ორი არაგადამფარავი F_{1}, F_1 მიმდევრობა; ***ბ)*** მიკროპროცესორთან ინფორმაციის გასაცვლელად გარე მოწყობილობებისა და მეხსიერების მზადყოფნის სიგნალი ***READI; გ)*** მთავარი პროგრამის შესრულების შეწყვეტისა და ამ შეწყვეტის მომსახურე ქვეპროგრამის შესრულებაზე გადასვლის შესახებ გარე მოწყობილობებიდან შემოსული შეკითხვის სიგნალი ***INT***; ***დ)*** სალტეების მიტაცების შესახებ გარე მოწყობილობიდან შემოსული სიგნალი ***HOLD***; იგი, როგორც წესი, საჭიროა მეხსიერებასან პირდაპირი შეღწევის არხით ინფორმაციის გაცვლის ორგანიზებისათვის; ***ე)*** მიკროპროცესორის საწყისი დაყენებისათვის საჭირო ჩამოყრის სიგნალი ***RESET***.

მართვის ბლოკის გამოსასვლელზე წარმოიშვება პროცესორის შიგა მოწყობილობების მართვის სიგნალები (***5.3*** ნახაზზე ისინი ნაჩვენები არ არის) და სიგნალები, რომლებითაც იმართება გარე მოწყობილობები. კერძოდ, გარე მოწყობილობების მართვისათვის წარმოიშობა სიგნალები: ***ა)*** სინქრონიზაციის სიგნალი ***SYNC***; იგი თითოეული სამანქანო ციკლის დაწყებას გვიჩვენებს; ***სამანქანო ციკლი*** წარმოადგენს დროის მონაკვეთს, რომელსაც ხარჯავს გარე მოწყობილობა ან მეხსიერება მეხსიერებასთან ერთხელ მიმართვის დროს; ***ბ)*** მონაცემების მისაღებად პროცესორის მზადყოფნის მადასტურებელი სიგნალი ***DBIN***; ***გ)*** ლოდინის რეჟიმში პრიცესორის ყოფნის მადასტურებელი სიგნალი ***WAIT***; ***დ)*** სალტეების მიტაცების სიგნალი ***HLDA***; იგი სალტეების მაღალომურ მდგომარეობაში ყოფნას ადასტურებს, რაც პროცესორის გვერდის ავლით მეხსიერებასთან გარე მოწყობილობების მიმართვას ხდის შესაძლებელს; ***ე)*** შეწყვეტის ნებადართვის სიგნალი ***INTE***; იგი ადასტურებს მართვის ბლოკში არსებული ***შეწყვეტის ნებადამრთველი ტრიგერის*** ლოგიკური *1*-ის მდგომარეობაში ყოფნას, რაც შეკითხვის სიგნალების მიღებას ხდის შესაძლებელს; ***ვ)*** გაცემის $\overline{WR}$=0 სიგნალი; იგი გვიჩვენებს, რომ მეხსიერებაში ჩასაწერად ან გარე მოწყობილობებისათვის გადასაცემად საჭირო ინფორმაცია პროცესორმა გამოატანა სალტეს.

8. მიკროპროცესორის ინტერფეისი. ინფორმაცია პროცესორის კვანძებს შორის ***მონაცემების შიგა 8***-თანრიგიანი ***სალტით***, ხოლო გარე კვანძებს შორის – ***მონაცემების 8***-თანრიგიანი ***სალტითა*** (***მნ. სტ***-ით) და ***პუფერული რეგისტრით*** გაიცვლება. მეხსიერებისა და გარე მოწყობილობების დამისამართებისათვის გამოიყენება ***მისამართების 16 თანრიგიანი სალტედ*** და ***მისამართის რეგისტრი.***

VI თავი
მიკროპროცესორული სისტემის ფუნქციონირების საკითხები

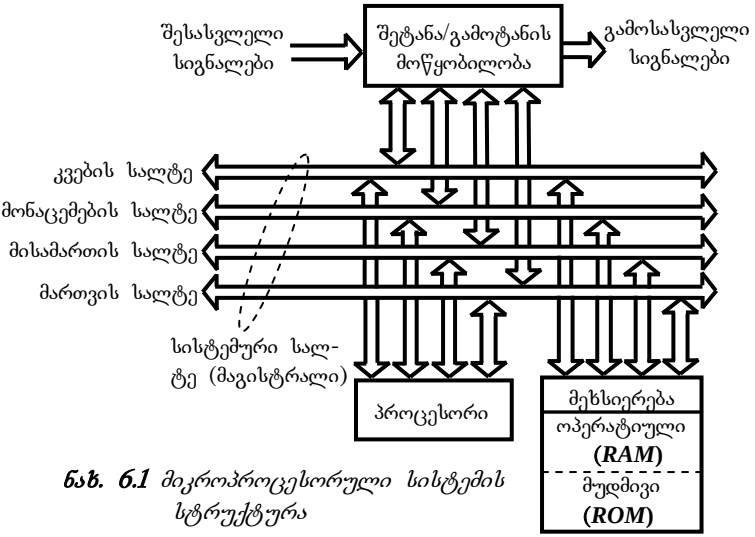
6.1. მიკროპროცესორული სისტემის ზოგადი დახასიათება

მიკროპროცესორული სისტემა ეწოდება ერთი ან რამდენიმე მოწყობილობების შემდგარ ფუნქციურად დასრულებულ ნაკეთობას, რომელთაგანაც ძირითადია მიკროპროცესორი და/ან მიკროკონტროლერი [11]. მისი ტიპური სტრუქტურა ნახ***6.1***.-ზეა მოყვანილი. იგი შედგება შემდეგი სამი ძირითადი მოწყობილობისაგან: ***1) პროცესორი; 2) მეხსიერება***, რომელიც მოიცავს ***ოპერატიულ*** (***RAM***) მეხსიერებასა და მუდმივ (***ROM***) მეხსიერებას. უკანასკნელი გამოიყენება მონაცემებისა და პროგრამების შესანახად); ***3) შეტანა/გამოტანის*** (***I/O*** – *Input/Output Devices*) ***მოწყობილობა,*** რომელიც გამოიყენება გარე მოწყობილობებთან მიკროპროცესორის დასაკავშირებლად, მიკროპროცესორის მიერ შესასვლელი სიგნალების მისაღებად და გამოსასვლელი სიგნალების გასაცემად. ***სიგნალის მიღებისას*** ეს სიგნალი შეიტანება მიკროპროცესორში და თვლიან რომ ***მიკროპროცესორმა წაიკითხა*** (Read) იგი; ***სიგნალს გაცემისას*** ეს სიგნალი გაიტანება გარე მოწყობილობაში და თვლიან, რომ გარე მოწყობილობამ ***ჩაწერა*** (***Write***) იგი.

მიკროპროცესორული სისტემის მოწყობილობები გაერთიანებულია საერთო ***სისტემური სალტით*** (***სისტემური მაგისტრალით***), რომელიც შედგება შემდეგი ოთხი სალტისაგან: ***1)*** მისამართის სალტე (***Address Bus***); ***2)*** მონაცემთა სალტე (***Data Buss***); ***3)*** მართვის სალტე (***Control Buss***); ***4)*** კვების სალტე (***Pover Bus***).

მისამართის სალტე გამოიყენება იმ მოწყობილობის მისამართის (ნომრის) განსასაზღვრავად, რომელთანაც მოცემულ მომენტში უნდა გაცვალოს ინფორმაცია პროცესორმა. მიკროპროცესორულ სისტემაში თითოეულ მოწყობილობასა (მიკროპროცესორის გარდა) და მეხსიერების თითოეულ უჯრედს მინიჭებული აქვს საკუთარი მისამართი

(ნომერი). როდესაც პროცესორი მისამართის სალტეზე გამოიტანს რომელიმე მისამართის კოდს, მოწყობილობისათვის, რომელსაც მინიჭებული აქვს ეს მისამართი, ნათელი ხდება, რომ მან უნდა გაუცვალოს ინფორმაცია პროცესორს. მისამართების სალტე შეიძლება იყოს ერთმხრივ ან ორმხრივად მიმართული.



***ნახ. 6.1** მიკროპროცესორული სისტემის სტრუქტურა*

მონაცემების სალტე წარმოადგენს მიკროპროცესორული სისტემის ყველა მოწყობილობას შორის საინფორმაციო კოდების გადასაცემად გამოყენებულ მთავარ სალტეს. ინფორმაციის გადაგზავნაში ჩვეულებრივ მონაწილეობს პროცესორი. იგი მონაცემების კოდს აგზავნის რომელიმე მოწყობილობაში ან მეხსიერების რომელიმე უჯრედში და პირიქით – მონაცემების კოდს იღებს რომელიმე მოწყობილობიდან ან მეხსიერების უჯრედიდან. თუმცა შესაძლებელია ინფორმაციის გადაცემა პროცესორის მონაწილეობის გარეშე მოხდეს მოწყობილობებს შორის. მონაცემების სალტე ყოველთვის ორმხრივაა მიმართული.

მართვის სალტე მისამართისა და მონაცემების სალტეებისაგან განსხვავებით შედგება განცალკევებული მმართველი სიგნალებისაგან. ინფორმაციის გაცვლის პროცესში თითოეულ ამ სიგნალს აქვს საკუთარი ფუნქცია. ზოგიერთი სიგნალი გამოიყენება გადასაცემი ან მისაღები მონაცემების ***სტრობირებისათვის*** (ე. ი. ისინი განსაზღვრავს

დროის მომენტებს, რომელთა დროსაც კოდი გამოტანილია მონაცემების სალტეზე). დანარჩენი სიგნალები შეიძლება გამოყენებული იქნეს მონაცემების მიღების დასადასტურებლად, ყველა მოწყობილობის საწყის მდგომარეობაში მოსაყვანად, მოწყობილობების ტესტირებისათვის და ა. შ. მართვის სალტე შეიძლება იყოს ერთმხრივ ან ორმხრივად მიმართული.

კვების სალტე გამოიყენება არა საინფორმაციო სიგნალების გადასაგზავნად, არამედ სისტემის კვებისათვის. იგი შედგება კვების ხაზებისა და საერთო სადენისაგან. მიკროპროცესორულ სისტემაში შეიძლება იყოს კვების ერთი (ხშირად ***+5*** ვოლტიანი) ან რამდენიმე (***-5***, ***+12*** და ***-12*** ვოლტიანი) წყარო. კვების თითოეულ ძაბვას შეესაბამება კავშირის საკუთარი ხაზი. ამ ხაზებთან მოწყობილობები პარალელურად არის მიერთებული.

მიკროპროცესორულ სისტემაში ***შესასვლელი კოდის შესატანად*** პროცესორი მისამართის სალტით მიმართავს შეტანა-გამოტანის საჭირო მოწყობილობას და მონაცემების სალტით იღებს შესასვლელ ინფორმაციას. მიკროპროცესორული სისტემიდან ***გამოსასვლელი კოდის (სიგნალის) გამოსატანად*** პროცესორი მისამართის სალტით მიმართავს შეტანა/გამოტანის საჭირო მოწყობილობას და მონაცემების სალტით მას გადასცემს გამოსასვლელ ინფორმაციას.

ინფორმაციის რთული მრავალსაფეხუროვანი დამუშავების დროს პროცესორმა შეიძლება საშუალედო შედეგები სისტემურ ოპერატიულ მეხსიერებაში შეინახოს. მეხსიერების ნებისმიერ უჯრედთან მიმართვისათვის პროცესორს მისამართის სალტეზე გამოაქვს ამ უჯრედის მისამართი და მონაცემების სალტით მასში გადასცემს, ან მისგან იღებს საინფორმაციო კოდს. ოპერატიულ ან მუდმივ მეხსიერებაში მმართველი კოდებიცაა (პროცესორის მიერ შესასრულებელი ბრძანებებიცაა) მოთავსებული, რომელთა წაკითხვა პროცესორს მისამართების სალტეზე არსებული მისამართების შესაბამისად მონაცემების სალტით შეუძლია. ***მუდმივი მეხსიერებაში*** ძირითადად შენახულია მიკროპროცესორული სისტემის საწყისი დამუშავების პროგრამა, რომელიც კვების ყოველი ჩართვის შემდეგ სრულდება. მასში ინფორმაცია ერთხელ და სამუდამოდ შეიტანება.

ამგვარად, მიკროპროცესორულ სისტემაში საინფორმაციო და ბრძანებების კოდები სალტეების მეშვეობით მიმდევრობით, რიგითობის

დაცვით, გადაიცემა. ეს განაპირობებს მიკროპროცესორული სისტემის შედარებით დაბალ სისწრაფეს. აღნიშნულ სისწრაფეს ზღუდავს არა იმდენად პროცესორის ნელმოქმედება (რაც ასევე მნიშვნელოვანია) და მაგისტრალით ინფორმაციის გაცვლის დაბალი სიჩქარე, არამედ ის, რომ ინფორმაცია მიმდევრობით გადაიცემა.

მნიშვნელოვანია გავითვალისწინოთ ისიც, რომ ***შეტანა/გამოტანის მოწყობილობებს,*** რომლებიც «ხისტი ლოგიკის» გამოყენებითაა აგებული, შეიძლება დავაკისროთ მიკროპროცესორული სისტემის მიერ შესასრულებელი ფუნქციების ნაწილის შესრულება; მაშასადამე, შეგვიძლია აღნიშნული ფუნქციების ერთი ნაწილი შევასრულოთ აპარატურულად, მეორე ნაწილი კი – პროგრამულად. ამ დროს უნდა გავახსოვდეს, რომ ***აპარატურული რეალიზაცია*** აჩქარებს ფუნქციების შესრულების სისწრაფეს, მაგრამ არასაკმარისად მოქნილია და ზრდის სისტემის ღირებულებას; ***პროგრამული რეალიზაცია,*** პირიქით, ამცირებს ფუნქციების შესრულების სისწრაფეს, მაგრამ უზრუნველყოფს მაღალ მოქნილობას და არ ზრდის სისტემის ფასს. მაშასადამე, ფუნქციები აპარატურულად და პროგრამულად რეალიზებისათვის ისე უნდა გავყოთ, რომ საჭირო მოქნილობისა და ღირებულების შენარჩუნების პირობებში მაქსიმალურად ამაღლდეს ფუნქციების შესრულების სწრფმოქმედება. აღნიშნულის გამო უნდა მოხდეს აპარატურული და პროგრამული ფუნქციების ოპტიმალური კომბინირება.

ზოგჯერ ***შეტანა/გამოტანის მოწყობილობები*** თავად შეიცავს საკუთარ პროცესორს, ე. ი. თვითონვეა მცირე მიკროპროცესორული სისტემა. მასში პროგრამული ფუნქციების გადატანით შეგვიძლია განვტვირთოთ ძირითადი სისტემის ცენტრალური პროცესორი.

6.2. მიკროპროცესორული სისტემის მუშაობის რეჟიმები

მიკროპროცესორული სისტემა, როგორც აღვნიშნეთ, უზრუნველყოფს მუშაობის მაღალ მოქნილობას, რის გამოც მისი საშუალებით ნებისმიერი ამოცანის გადაწყვეტაა შესაძლებელი. ამ მოქნილობას განაპირობებს ის გარემოება, რომ სისტემის ფუნქციების რეალიზებას პროცესორი პროგრამის (პროგრამული უზრუნველყოფის, ანუ ***software***–ს). აპარატურა (აპარატურული უზრუნველყოფა – ***hardwa-***

re) ნებისმიერი ამოცანისათვის უცვლელია. მეხსიერებაში პროგრამის ჩაწერით შეიძლება ვაიძულოთ მიკროპროცესორული სისტემა შეასრულოს მოცემული აპარატურით მხარდაჭერილი ნებისმიერი ამოცანა. ამასთანავე, ***კავშირების სალტური ორგანიზაცია*** საშუალებას გვაძლევს საკმაოდ იოლად შევცვალოთ აპარატურული მოდულები, მაგალითად, შეიძლება ძველი მეხსიერება შევცვალოთ დიდი მოცულობის ან უფრო სწრაფმოქმედი ახალი მეხსიერებით, სისტემას დავუმატოთ შეტანა/გამოტანის მოწყობილობა ან მოვახდინოთ არსებული მოწყობილობის მოდერნიზება, პროცესორი შევცვალოთ უფრო მძლავრი პროცესორით. ყოველივე ზემოთ აღნიშნული საშუალებას გვაძლევს დამატებით გავზარდოთ სისტემის მოქნილობა და სისტემისადმი წაყენებული მოთხოვნების ნებისმიერი ცვლილებების პირობებშიც უზრუნველვყოთ სისტემის საჭირო ხანგამძლეობა.

მიკროპროცესორული სისტემის მოქნილობას ამაღლების კიდევ ერთი წყაროა ***მუშაობის რეჟიმების შერჩევის*** შესაძლებლობა. ეს უკანასკნელი ფაქტობრივად იგივეა, რაც სისტემის ***მაგისტრალში ინფორმაციის გაცვლის რეჟიმის*** შერჩევის შესაძლებლობა.

ნებისმიერ მიკროპროცესორულ სისტემას (მათ შორის - კომპუტერსაც) აქვს მაგისტრალში ინფორმაციის გაცვლის შემდეგი სამი რეჟიმი: ***1)*** ინფორმაციის პროგრამული გაცვლის რეჟიმი; ***2)*** შეწყვეტებით (***Interrupts***-ის) ინფორმაციის გაცვლის რეჟიმი და ***3)*** მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლის რეჟიმი (***Direct Memori Access***-ს, ანუ ***DMA***-ს) ინფორმაციის გაცვლის რეჟიმი.

ინფორმაციის პროგრამული გაცვლის რეჟიმი ნებისმიერი მიკროპროცესორული სისტემის ძირითადი რეჟიმია, რომლის გარეშე ინფორმაცია ვერც ერთი სხვა რეჟიმით ვერ გაიცვლება. ამ რეჟიმში სისტემური მაგისტრალის ერთადერთი ბატონ-პატრონი (ანუ მავალებელი, ***Master***) არის პროცესორი. ინფორმაციის გაცვლის ნებისმიერი ოპერაციის მაინიცირებელი პროცესორია და თითოეული მათგანი პროგრამის მიერ დადგენილი თანამიმდევრობის მკაცრი დაცვით სრულდება.

პროცესორი მეხსიერებიდან კითხულობს (ირჩევს) ბრძანებათა კოდებს და ასრულებს მათ. შესრულების პერიოდში მეხსიერებიდან ან შეტანა/გამოტანის მოწყობილობებიდან წაიკითხავს მონაცემებს, დამუშავებს მათ, მონაცემებს ჩაწერს მეხსიერებაში ან გადასცემს მათ

შეტანა/გამოტანის მოწყობილობებს. პროგრამაში შემავალ ბრძანებებს პროცესორი კითხულობს წრფივად, ციკლურად ან ნახტომით.

წრფივი წაკითხვის შემთხვევაში პროცესორი ბრძანებებს კითხულობს თანამიმდევრულად, პროგრამაში მათი განლაგების რიგითობის დაცვით;

ციკლური წაკითხვა შეიძლება შევადაროთ ტექსტის დაზეპირების მიზნით ამ ტექსტში შემავალი გარკვეული მონაკვეთის რამდენჯერმე განმეორებით წაკითხვას. პროცესორი ბრძანებების გარკვეულ თანამიმდევრობას რამდენჯერმე კითხულობს. ***ციკლიდან გამოსვლა*** ნიშნავს, გარვეული რაოდენობის განმეორებითი პროცესის შემდეგ ტექსტის პროგრამის მომდევნო ბრძანებების წაკითხვაზე გადასვლას. პროცესორი თუ ვერ ახერხებს ციკლიდან გამოსვლას და უსასრულოდ აგრძელებს ბრძანებათა გარკვეული ერთობლიობის კითხვის პროცესს, მაშინ ამბობენ, რომ მოხდა მისი ***ჩაციკვლა.***

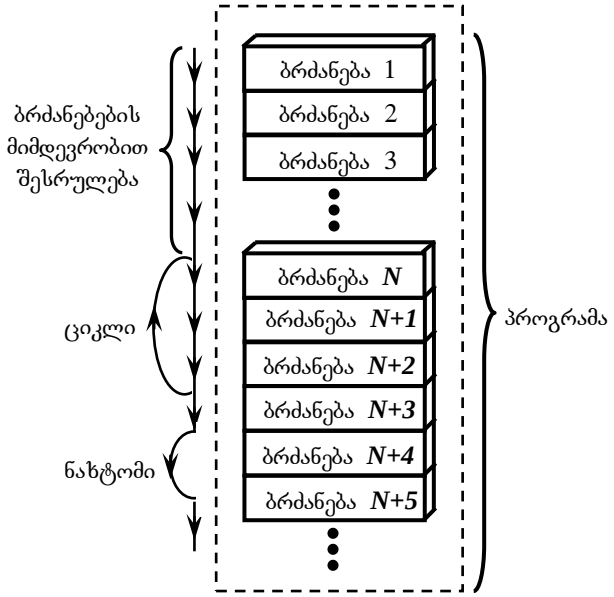
ნახტომით წაკითხვისას პროცესორი გარკვეული ბრძანების წაკითხვის შემდეგ ერთი ან რამდენიმე ბრძანების იქით მდგარი ბრძანების წაკითხვაზე გადადის.

წაკითხვის სამივე ხერხისათვის დამახასიათებელია ის, რომ პროგრამის წაკითხვის პროცესი უწყვეტია და ამ პროცესს სრულად აკონტროლებს პროცესორი. პროცესორი არ რეაგირებს პროგრამასთან დაუკავშირებელ არც ერთ გარე მოვლენაზე (ნახ.***6.2***). მაგისტრალში ყველა სიგნალს მოცემულ შემთხვევაში აკონტროლებს პროცესორი.

შეწყვეტებით ინფორმაციის გადაცემა მაშინ გამოიყენება, როდესაც საჭიროა მიკროპროცესორულმა სისტემამ რაღაც გარეგან მოვლენაზე, გარედან შემოსულ გარკვეულ სიგნალზე მოახდინოს რეაგირება. კომპიუტერის შემთხვევაში გარე მოვლენა შეიძლება იყოს კლავიატურის კლავიშზე თითის დაჭერა, ან ლოკალური ქსელიდან მონაცემთა პაკეტის შემოსვლა. კომპიუტერმა რეაგირება უნდა მოახდინოს ამაზე, კერძოდ, ეკრანზე გამოიტანოს სიმბოლო ან წაიკითხოს ქსელიდან მიღებული პაკეტი და დაამუშაოს იგი.

გარე მოვლენაზე რეაგირება ზოგადად შეიძლება მოხდეს: ***1)*** ამ მოვლენის დადგომის ფაქტზე მუდმივი ***პროგრამული კონტროლის განხორციელების გზით*** (ამ მეთოდს ეწოდება ალმის გამოკითხვის

ანუ ***Polling***-ის მეთოდი); ***2) შეწყვეტის დახმარებით***, რაც ნიშნავს მიმდინარე პროგრამის შესრულებიდან ექსტერნულად აუცილებელი

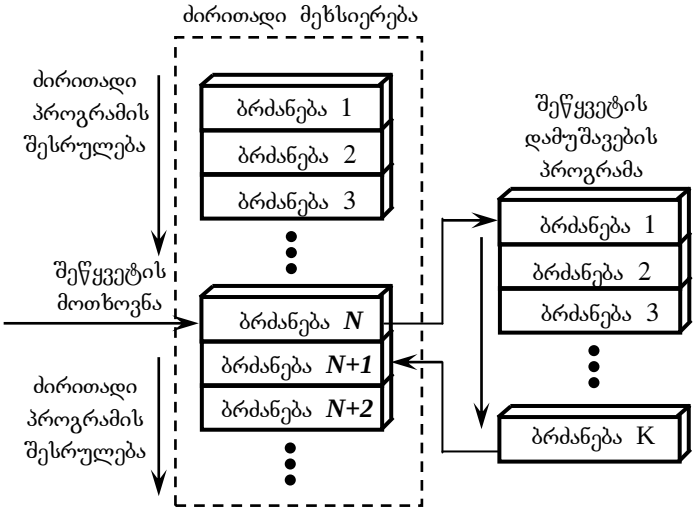


ნახ. 6.2. *ინფორმაციის პროგრამული გაცვლა*

პროგრამის შესრულებაზე პროცესორის იძულებით გადაყვანას; ***3)*** მეხსიერებასთან პირდაპირი დაშვების გზით. ამ დროს ინფორმაციის გაცვლაში პროცესორი არ მონაწილეობს: იგი გამორთულია სისტემური მაგისტრალიდან და მას ჩამორთმეული აქვს შემოსულ ექსტერნულ პროგრამასა და მეხსიერებას შორის შუამავლის ფუნქციის შესრულების ფუნქცია. აღნიშნული პროგრამა პირდაპირ დაიშვება მეხსიერებასთან და თვითონ ახორციელებს მასთან კავშირს.

სამივე მეთოდის ილუსტრირება მოვახდინოთ შემდეგი მარტივი მაგალითით. დავუშვათ, რომ საუზმისათვის ქურაზე ასადუღებლად დავდგით რძე. ნათელია, რომ რძის ადუღებაზე სასწრაფოდ უნდა მოვახდინოთ რეაგირება. ამის ორგანიზების პირველი გზაა რძისთვის მუდმივად თვალყურის დევნება, რომლის დროსაც სხვა საქმისთვის დრო არ დაგვრჩება. ამას ეთანადება ***ალმის გამოკითხვიანი პროგრამული რეჟიმი***. მეორე გზაა ქვაბზე გადამწოდის დაყენება, რომელიც რძის ადუღებისას მოგვცემს ბგერით სიგნალს. ამ შემთხვევაში რძის

ადუღებამდე შეგვეძლება გულდამშვიდებით შევასრულოთ სხვა საქმე: ხმოვანი სიგნალის გაგონებისთანავე გამოვრთავთ ქურას, თუმცა ქურის გამორთვამდე რაღაც მცირე დრო დაგვეკარგება მიმდინარე სამუშაოდან გამოსასვლელად. ამიტომ ჩვენი რეაქცია პირველ შემთხვევაზე უფრო დაგვიანებული იქნება. აღნიშნულ შემთხვევას ეთანადება ***ინფორმაციის გაცვლა შეწყვეტის გამოყენებით.*** არსებობს მესამე გზაც, რომლის დროსაც ქვაბზე დაყენებული გადამწოდი რძის ადუღების ფაქტს ჩვენ კი არ შეგვატყობინებს, არამედ უჩვენოდ გამორთავს ქურას. უკანასკნელი შემთხვევა შეიძლება შევადაროთ ინფორმაციის გაცვლას ***მეხსიერებაში პირდაპირი დაშვების გზით,*** თუმცა შედარება რამდენადმე უხეშია, რადგან ჩვენ (რომელიც პროცესორის როლს ვთამაშობთ) მთლად უსაქმოდ არ ვრჩებით: ვასრულებთ სხვა სამუშაოს.



ნახ. 6.3 *შეწყვეტის მომსახურება*

მიკროპროცესორულ სისტემაში რეალიზებული რომ იყოს ალმის გამოკითხვის პირველი შემთხვევა, საჭიროა იმ მოწყობილობაზე მიერთებულ შეტანა/გამოტანის მოწყობილობიდან, რომლის ქცევაზეც საჭიროა მოხდეს სასწრაფო რეაგირება, პროცესორი ინფორმაციას მუდმივად უნდა კითხულობდეს.

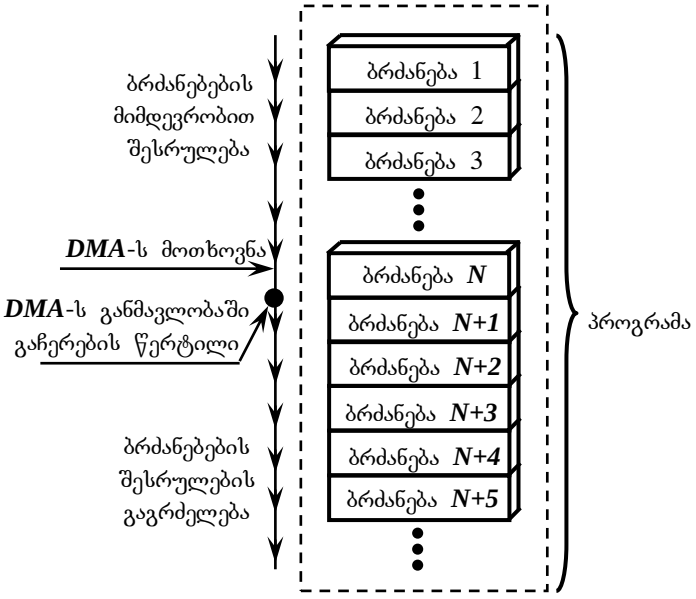
შეწყვეტით ინფორმაციის გაცვლის მეორე შემთხვევის დროს პროცესორი როგორც კი გარე მოწყობილობიდან მიიღებს შეწყვეტის, ანუ ***IRQ*** მოთხოვნას (***IRQ - Interrupt ReQuest*** «მოთხოვნა შეწყვეტაზე»), დაასრულებს მიმდინარე ბრძანების შესრულებას და გადადის შეწყვეტის დამუშავების პროგრამაზე. ამ უკანასკნელის დასრულების შემდეგ იგი უბრუნდება საწყის პროგრამას იმ წერტილიდან, რომელშიც მოხდა შეწყვეტა (ნახ.***6.3***)

ინფორმაციის პროგრამულად გაცვლის რეჟიმის ანალოგურად ***შეწყვეტით ინფორმაციის გაცვლის რეჟიმის დროსაც მთელ სამუშაოს ასრულებს პროცესორი.*** გარე მოვლენა მას აიძულებს ყურადღება მხოლოდ დროებით გადაიტანოს შეწყვეტის პროგრამაზე. გარე მოვლენაზე რეაქცია შეწყვეტით ინფორმაციის დამუშავების დროს უფრო შეყოვნებულია ინფორმაციის პროგრამულად დამუშავებასთან შედარებით. ამ უკანასკნელივით მაგისტრალზე ყველა სიგნალი პროცესორს გამოაქვს, ე. ი. იგი აკონტროლებს მაგისტრალს.

შეწყვეტების მომსახურებისათვის სისტემაში ზოგჯერ შეიტანება შეწყვეტების კონტროლერის სპეციალური მოდული, რომელიც ინფორმაციის გაცვლაში არ მონაწილეობს. მისი ამოცანაა მიკროპროცესორს გაუადვილოს მუშაობა გარე მოწყობილობებიდან შემოსულ მოთხოვნებთან. ამ კონტროლერს ჩვეულებრივ მართავს პროცესორი სისტემური სალტის მეშვეობით.

შეწყვეტებით ინფორმაციის გაცვლის რეჟიმი არ ზრდის მიკროპროცესორული სისტემის მუშაობის სიჩქარეს. მისი გამოყენება საშუალებას გვაძლევს უარი ვთქვათ გარე მოვლენის ალმის მუდმივი გამოკითხვაზე, რაც გარე მოვლენის მოხდენამდე პროცესორს საშუალებას აძლევს დაკავდეს კონკრეტული ამოცანების შესრულებით.

მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლის რეჟიმი (DMA) წინა ორი რეჟიმისაგან დიამეტრალურად განსხვავებული რეჟიმია, რომლის დროსაც ინფორმაცია სისტემური სალტით გაიცვლება პროცესორის მონაწილეობის გარეშე. გარე მოწყობილობა, რომელიც საჭიროებს მომსახურებას, აცნობებს პროცესორს, რომ აუცილებელია ***DMA*** რეჟიმი. საპასუხოდ პროცესორი დაასრულებს მიმდინარე ბრძანების შესრულებას და ყველა სალტიდან გამოირთვება, რითაც მომთხოვნი მოწყობილობისათვის ცნობილი ხდება, რომ შეიძლება დაიწყოს ***DMA*** რეჟიმში ინფორმაციის გაცვლა.

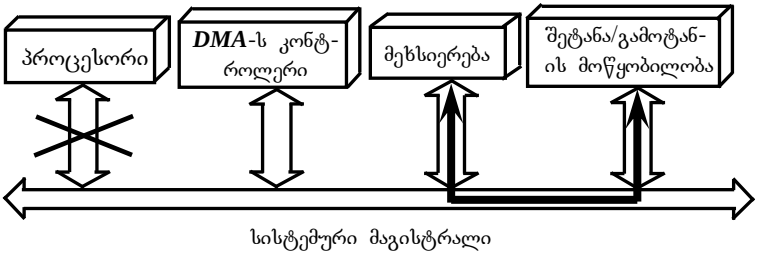


***ნახ. 6.4. DMA** –ს მომსახურება*

DMA ოპერაცია დაიყვანება შეტანა/გამოტანის მოწყობილობიდან მეხსიერებაში ან პირიქით ინფორმაციის გადაგზავნამდე. ინფორმაციის გადაგზავნის დამთავრების შემდეგ პროცესორი ძირითად (საწყის) პროგრამას უბრუნდება იმ წერტილში, სადაც მოხდა მისი შეწყვეტა (ნახ. ***6.4***). ეს ჰგავს შეწყვეტების რეჟიმის მომსახურებას, ოღონდ იმ განსხვავებით, რომ პროცესორი წყვეტს ინფორმაციის გაცვლის პროცესში მონაწილეობას. შეწყვეტების შემთხვევის მსგავსად ***DMA***-ს დროსაც გარე მოვლენაზე რეაქცია პროგრამული რეჟიმის დროს არსებულ რეაქციაზე გაცილებით დაბალია.

ნათელია, რომ ამ შემთხვევაში სისტემაში საჭიროა შევიტანოთ დამატებითი მოწყობილობა (***DMA***-ს კონტროლერი), რომელიც პროცესორის ყოველგვარი მონაწილეობის გარეშე სისტემური სალტით ინფორმაციას სრულფასოვნად გაცვლის. ამასთანავე, პროცესორმა აღნიშნულ ***DMA***-ს კონტროლერს წინასწარ უნდა შეატყობინოს, თუ საიდან უნდა აიღოს მან ინფორმაცია და/ან სად უნდა მოათავსოს იგი. ***DMA***-ს კონტროლერი შეიძლება მივიჩნიოთ სპეციალიზებულ

პროცესორად, რომელიც ცენტრალური პროცესორისაგან განსხვავებით თვითონ არ მონაწილეობს ინფორმაციის გაცვლაში, მაგრამ შეუძლია ინფორმაციის მიღება და გაცემა (ნახ. ***6.5***).



ნახ.6.5. *საინფორმაციო ნაკადები* ***DMA*** *რეჟიმში*

შესაძლებელია ***DMA*** კონტროლერი შედიოდეს შეტანა/გამოტანის ერთ (ან რამდენიმე) მოწყობილობის სტრუქტურაში, რომელსაც (ან რომლებსაც) სჭირდება ***DMA*** რეჟიმი.

თეორიულად მეხსიერებაში პირდაპირი შეღწევის რეჟიმმა შეიძლება ინფორმაციის უფრო სწრაფად გაცვალოს, ვიდრე პროგრამულმა რეჟიმმა, რადგან ***DMA***-ის სპეციალიზებულ კონტროლერს შეუძლია მონაცემები ცენტრალურ პროცესორზე უფრო სწრაფად გადასცეს; მაგრამ პრაქტიკულად ამ უპირატესობის რეალიზება ყოველთვის ვერ ხერხდება. ***DMA*** რეჟიმში გაცვლის სიჩქარე ჩვეულებრივ მაგისტრალის შესაძლებლობებითაა შეზღუდული. გარდა ამისა, ***DMA*** კონტროლერის რეჟიმების პროგრამულად დასახვის აუცილებლობამ შეიძლება ***DMA*** რეჟიმში მონაცემების უფრო სწრაფად გადაგზავნის შედეგად მიღებული მოგება შეიძლება გააქარწყლოს. ამიტომ ***DMA რეჟიმი იშვიათად გამოიყენება.***

სისტემაში თუ უკვე არსებობს დამოუკიდებელი ***DMA*** კონტროლერი, მაშინ მთელ რიგ შემთხვევაში შეიძლება მნიშვნელოვნად გავამარტივოთ ***DMA*** რეჟიმში მომუშავე შეტანა/გამოტანის აპარატურა. ალბათ, ეს არის ***DMA*** რეჟიმის უდავო ერთადერთი უპირატესობა.

6.3. მიკროპროცესორული სისტემების არქიტექტურა

მიკროპროცესორული სისტემების უმრავლესობას საფუძვლად უდევს ***ჯონ ფონ ნეიმანის, არტურ ბიორკსისა*** და ***ჰერმან გოლდსტაინის*** მიერ ***1946*** წელს ფორმულირებული შემდეგი პრინციპები:

1. პროგრამული მართვის პრინციპი (პროგრამა შედგება იმ ბრძანებების ნაკრებისაგან, რომლებსაც პროცესორი ერთმანეთის მიყოლებით ავტომატურად ასრულებს);

2. მეხსიერების ერთგვაროვნობის პრინციპი (პროგრამები და მონაცემები ერთსა და იგივე მეხსიერებაშია შენახული; ბრძანებებზე შეიძლება ისეთივე მოქმედებები შევასრულოთ, როგორც მონაცემებზე);

3. მისამართიანობის პრინციპი (ძირითადი მეხსიერება სტრუქტურულად შედგება დანომრილი უჯრედებისაგან).

ზემოთჩამოთვლილი პრინციპების დაცვით აგებულ მიკროპროცესორულ სისტემაზე ამბობენ, რომ მას აქვს კლასიკური, ანუ ***ფონ ნეიმანისეული*** (***პრისტონული***, იმ ქალაქის საპატივცემლოდ, სადაც მოღვაწეობდა ***ფონ ნეიმანი***) ***არქიტექტურა***.

ბუნებრივად იბადება კითხვა იმის შესახებ, თუ ***რას ეწოდება მიკროპროცესორული სისტემის არქიტექტურა***. იგი შეიძლება განისაზღვროს ვიწრო უტილიტარული და ზოგად ტექნიკური თვალსაზრისით.

ვიწრო უტილიტარული თვალსაზრისით ***მიკროპროცესორული სისტემის არქიტექტურა*** ამ სისტემის იმ თვისებების ერთობლიობაა, რომლებიც მნიშვნელოვანია მომხმარებლისათვის.

ზოგად ტექნიკური თვალსაზრისით ***მიკროპროცესორული სისტემის არქიტექტურა*** განისაზღვრება როგორც ამ სისტემის აპარატურული და პროგრამული უზრუნველყოფის ერთობლიობა, რომელიც ინფორმაციის ციფრული დამუშავების საშუალებას იძლევა. უფრო დაწვრილებით, მიკროპროცესორული სისტემის არქიტექტურა ეწოდება აღნიშნული სისტემის გარკვეულ დონეზე აღწერას, რომელიც მოიცავს დაპროგრამების სამომხმარებლო შესაძლებლობების, ბრძანებათა და დამისამართებათა სისტემების, მეხსიერების ორგანიზაციის აღწერასაც. არქიტექტურა განსაზღვრავს მიკროპროცესორული სისტემის მოქმედების პრინციპს, საინფორმაციო კავშირებსა და ძირი-

თადი ლოგიკური კვანძების (პროცესორის, ოპერატიული და გარე დამხსომებელი მოწყობილობების, პერიფერიული მოწყობილობების) ურთიერთშეერთებებს. სხვადასხვა მიკროპროცესორული სისტემების არქიტექტურათა ერთნაირობა უზრუნველყოფს მათ სამომხმარებლო ურთიერთშეთავსებადობასაც.

აქამდე ჩვენ განვიხილავდით მხოლოდ ***ფონ ნეიმანისეული*** (პრისტონული) ***არქიტექტურის*** მქონე მიკროპროცესორულ სისტემას. მონაცემებისა და ბრძანებების გადასაგზავნად მის სტრუქტურაში ერთადერთი სალტის არსებობის გამო (ნახ. ***ნ.ნ,ა***) მას ხშირად ***ერთსალტურ სისტემასაც*** უწოდებენ.

არსებობს მიკროპროცესორული სისტემის ალტერნატიული არქიტექტურა – მონაცემებისა და ბრძანებების ***დაცალკევებული სალტეებიანი*** (***ორსალტიანი***, ანუ ***ჰარვარდული***) ***არქიტექტურა.*** ამ არქიტექტურის დროს სისტემაში მონაცემებისათვის და ბრძანებებისათვის თავთავიანთი მეხსიერება არსებობს (ნახ.***ნ.ნ,ბ***). მათთან პროცესორი ინფორმაციას განცალკევებული სალტეებით ცვლის.

განვიხილოთ ორივე არქიტექტურის როგორც დადებითი, ისე უარყოფითი მხარეები.

პრისტონული, ანუ ფონ ნეიმანისეული (საერთო სალტიანი) არქიტექტურა მარტივია, იგი პროცესორისაგან არ მოითხოვს ორივე სალტის ერთდროულად მომსახურებასა და მათი საშუალებით ინფორმაციის გაცვლის გაკონტროლებას. მონაცემებისა და ბრძანებებისათვის საერთო მეხსიერების არსებობა საშუალებას იძლევა, მოქნილად განაწილდეს მისი მოცულობა მონაცემებსა და ბრძანებებს შორის. მაგალითად, ზოგჯერ გამოიყენება დიდი და რთული პროგრამა, ხოლო შესანახი მონაცემების რაოდენობა მცირეა და პირიქით, საჭიროა მარტივი პროგრამა, მაგრამ მისი მეშვეობით მუშავდება დიდი მოცულობის მონაცემები. მონაცემებსა და ბრძანებებს მეხსიერების მოცულობა მოქნილად შეიძლება გაუნაწილდეს. მთავარია მხოლოდ ის, რომ პროგრამა და მონაცემები ერთად მოთავსდეს სისტემაში. ასეთი არქიტექტურის მქონე სისტემებში, როგორც წესი, გამოიყენება საკმაოდ დიდი (ათობითი და ასობითი მეგაბაიტის ტოლი) მოცულობის მეხსიერება, რაც რთული ამოცანების გადაწყვეტის საშუალებას იძლევა.

გაცილებით რთულია ჰარვარდული არქიტექტურა, იგი აიძულებს პროცესორს ერთდროულად კოდების ორ ნაკადთან იმუშაოს და ინ-

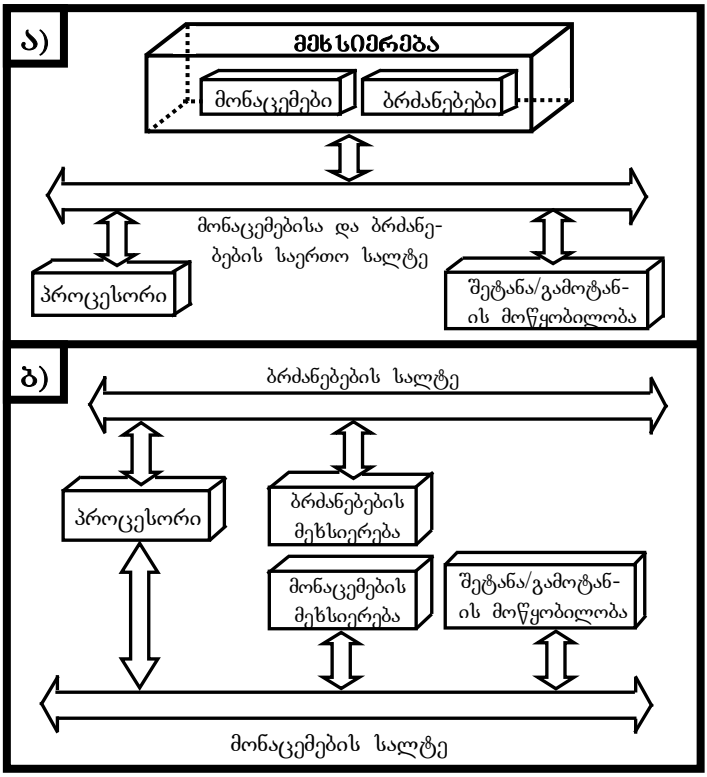
ფორმაცია ორ სალტეში ერთდროულად გაცვალოს. პროგრამა შესაძლებელია მხოლოდ ბრძანებების მეხსიერებაში, ხოლო მონაცემები – მონაცემების მეხსიერებაში მოვათავსოთ. ასეთი ვიწრო სპეციალიზაცია ზღუდავს სისტემის მიერ გადასაწყვეტი ამოცანების არეალს, რადგან მეხსიერების მოქნილი გადანაწილების საშუალებას არ იძლევა. ამ შემთხვევაში მონაცემების მეხსიერებისა და ბრძანებების მეხსიერების მოცულობები ძალიან დიდი არ არის, ამიტომ მოცემული არქიტექტურიანი სისტემები შესაძლებელია გამოვიყენოთ ნაკლებად რთული ამოცანების გადასაწყვეტად.

ჰარვარდული (ორსალტიანი) არქიტექტურის ძირითადი ღირსება, უპირველესად ყოვლისა, არის მისი სწრაფმოქმედება.

საქმე ისაა, რომ ერთადერთი სალტის დროს პროცესორი იძულებულია ამ სალტით როგორც მიიღოს (მეხსიერებიდან ან შეტანა/გამოტანის მოწყობილობებიდან) ასევე გადააგზავნოს (მეხსიერებაში ან შეტანა/გამოტანის მოწყობილობაში) მონაცემები, აგრეთვე ამავე სალტის მეშვეობით უნდა წაიკითხოს ბრძანებები მეხსიერებაში. ბუნებრივია, რომ მაგისტრალში ერთდროულად ეს გადაგზავნები ვერ განხორციელდება. თანამედროვე პროცესორებს ბრძანებებისა შესრულების პროცესისა და სისტემურ სალტესთან ინფორმაციის გაცვლის ციკლების ურთიერთშეთავსება შეუძლია. კონვეიერული ტექნოლოგიებისა და სწრაფი კეშ-მეხსიერების გამოყენება მას საშუალებას აძლევს შედარებით ნელმომქმედ სისტემურ მეხსიერებასთან დააჩაროს ურთიერთმოქმედების პროცესი. ტაქტური სიხშირის ამაღლება და პროცესორების სტრუქტურის სრულყოფა ბრძანების შესრულების ხანგრძლივობას ამცირებს. მაგრამ სისტემის სწრაფმოქმედების შემდგომი გაზრდა მხოლოდ მონაცემების გადაგზავნებისა და ბრძანებების წაკითხვის შეთავსებით, ე. ი. ორსალტიან (ჰარვარდულ) არქიტექტურაზე გადასვლითაა შესაძლებელი.

ორსალტიანი (ჰარვარდული) არქიტექტურა გადასაწყვეტი ამოცანების შესაბამისად სალტეთა სტუქტურების (სხვადასხვა კოდის თანრიგების, ინფორმაციის გაცვლის წესის, სიჩქარის და ა.შ.) ოპტიმალურად შერჩევის საშუალებას გვაძლევს. ამიტომ სხვა პირობების ერთნაირობისას ორსალტურ (ჰარვარდულ) არქიტექტურაზე გადასვლა ამაღლებს მიკროპროცესორული სისტემის მუშაობის საჩქარეს,

თუმცა ზრდის აპარატურულ დანახარჯებსა და პროცესორის სტრუქტურას.



ნახ.6.6. *მიკროპროცესორული სისტემის ფონ ნეიმანისეული ანუ პრისტონული (ა) და ჰარვარდული (ბ) არქიტექტურა*

ორსალტიანი არქიტექტურის უპირატესობები ყველაზე მარტივად რეალიზდება ერთი მიკროსქემის ფარგლებში. ამ შემთხვევაში მნიშვნელოვნად მცირდება ამ არქიტექტურის ნაკლოვანებების ზეგავლენა. ამიტომ იგი გამოიყენება მიკროკონტროლერებში, რომლებიც ნაკლებად რთული ამოცანების სწრაფად გადაწყვეტის საშუალებას გვაძლევს.

6.4. მიკროპროცესორულ სისტემათა ტიპები

გამოყენების ფართო დიაპაზონისა და წაყენებული მოთხოვნების მრავალფეროვნების გამო დამუშავებულია ურთიერთგანსხვავებული სიმძლავრეების, სტრუქტურული თავისებურებებისა და შესაძლებლობების მქონე შემდეგი ტიპის მიკროპროცესორული სისტემები: ***1) მიკროკონტროლერები*** – უმარტივესი მიკროპროცესორული სისტემები, რომლებშიც სისტემის ყველა კვანძი ან მათი უმრავლესობა ერთი მიკროსქემის სახით არის რეალიზებული; ***2) კონტროლერები*** – განცალკევებული მოდულების სახით რეალიზებული მმართველი მიკროპროცესორული სისტემები; ***3) მიკროკომპიუტერები*** – გარე მოწყობილობებთან შეუღლების განვითარებული საშუალებებიანი შედარებით მძლავრი მიკროპროცესორული სისტემები; ***4) კომპიუტერები*** (მათ შორის – პერსონალური) – ყველაზე მძლავრი და უნივერსალური მიკროპროცესორული სისტემები.

ზემოთ ჩამოთვლილი ტიპის მიკროპროცესორულ სისტემების ურთიერთგამიჯვნა ხშირად საკმაოდ ძნელია, თუმცა არსებობს მათ შორის გარკვეული განსხვავებები.

მიკროკონტროლერები ისეთი უნივერსალური მოწყობილობებია, რომლებიც პრაქტიკულად გამოიყენება არა დამოუკიდებლად, არამედ უფრო რთული მიკროპროცესორული სისტემის, მათ შორის კონტროლერების, შემადგენლობაში. მიკროკონტროლერის სალტე მომხმარებლისაგან დაფარულია და მოთავსებულია მიკროსქემის შიგნით. შეზღუდულია მიკროკონტროლერებთან გარე მოწყობილობების მიერთების შესაძლებლობები. მიკროკონტროლერების გამოყენებით აგებული მოწყობილობები ერთი კონკრეტული ამოცანის გადასაწყვეტადაა განკუთვნილი.

კონტროლერები, როგორც წესი, აიგება გარკვეული ერთი ამოცანის, ან ერთმანეთთან დაახლოებული ამოცანების ჯგუფის გადასაწყვეტად. შეუძლებელია მათ მივუერთოთ დამატებითი კვანძები და მოწყობილობები; მაგალითად, დიდი მეხსიერება, შეტანა/გამოტანის საშუალებები და ა.შ. მათი სისტემური სალტე უმეტესწილად დაფარულია მომხმარებლისათვის. კონტროლერის სტრუქტურა მარტივია და მაქსიმალური სწრაფმოქმედების ნიშნითაა ოპტიმიზებული. უმეტესწილად შესასრულებელი პროგრამები მუდმივ მეხსიერებაშია შენახული და უცვლელია. კონსტრუქციულად კონტროლერები გამოდის ერთდაფური ვარიანტის სახით.

მიკროკომპიუტერებს კონტროლერებზე უფრო ღია სტრუქტურა აქვს: მათ სისტემურ სალტეზე რამდენიმე დამატებითი მოწყობილობის მიერთებაა შესაძლებელი. მიკროკომპიუტერები მოთავსებულია სისტემური მაგისტრალის მომხმარებლისათვის ხელმისაწვდომი გასართიან კორპუსში. მათ შეიძ-

იძლება ჰქონდეს ინფორმაციის შენახვის მაგნიტური მზიდიანი საშუალებები (მაგალითად, მაგნიტური დისკები) და მომხმარებელთან კავშირის საკმაოდ განვითარებული საშუალებები (ვიდეომონიტორი, კლავიატურა). მიკროკომპიუტერები გათვლილია ფართო წრის ამოცანების გადასაწყვეტად, ოღონდ კონტროლერებისაგან განსხვავებით, საჭიროა ყოველ ახალ ამოცანასთან მისი ხელახლა მისადაგება. მიკროკომპიუტერების მიერ შესასრულებელი პროგრამების შეცვლა ძალიან ადვილია.

კომპიუტერები და მათი გავრცელებული სახეები ***პერსონალური კომპიუტერები*** ყველაზე გავრცელებული უნივერსალური მიკროპროცესორული სისტემებია. მათში აუცილებლად გათვალისწინებულია მოდერნიზციისა და ახალ მოწყობილობებთან მიერთების შესაძლებლობა. მათი სისტემური სალტე ხელმისაწვდომია მომხმარებლისათვის. კომპიუტერთან გარე მოწყობილობები შეიძლება კავშირის რამდენიმე (ზოგჯერ ***10***-მდე) ჩაშენებული პორტით მივაერთოთ. კომპიუტერს ყოველთვის აქვს მომხმარებელთან კავშირის მეტად განვითარებული სისტემა, დიდი მოცულობის ინფორმაციის შესანახი საშუალება, საინფორმაციო ქსელებით სხვა კომპიუტერებთან დაკავშირების საშუალებები. მრავალფეროვანია კომპიუტერების გამოყენების სფერო, რომელიც ყველასათვის ცნობილია.

ზემოთ ჩამოთვლილი თითოეული მიკროპროცესორული სისტემის დახმარებით შეიძლება ნებისმიერი ამოცანის გადაწყვეტა. მაგრამ მისი ამორჩევის დროს უნდა ვისწრაფოდეს თავი ავარიდოთ სიჭარბეს და მოცემული ამოცანისათვის უზრუნველვყოთ აუცილებელი მოქნილობა.

ახალი მიკროპროცესორული სისტემების დამუშავებისას დღეისათვის უფრო ხშირად (დაახლოებით ***80%*** შემთხვევაში) ირჩევენ მიკროკონტროლერებს. ამ დროს მიკროკონტროლერებს იყენებენ დამოუკიდებლად (უმნიშვნელო აპარატურის დამატებით) ან შეტანა/გამოტანის განვითარებული საშუალებების მქონე უფრო რთულ კონტროლერებთან ერთად.

მიკროპროცესორებისა და მიკროპროცესორული კომპლექტების ბაზაზე აგებულ კლასიკურ მიკროპროცესორულ სისტემებს წარმოებები იშვიათად უშვებს მათი დამუშავებისა და გამართვის სირთულის გამო. ასეთი ტიპის სისტემებს ძირითადად მაშინ იყენებენ, როდესაც მოთხოვნებს ვერ აკმაყოფილებს ტიპური მიკროკონტროლერები.

დღეს მნიშვნელოვანი ადგილი უკავია პერსონალური კომპიუტერის საფუძველზე აგებულ მიკროპროცესორული სისტემებს. შეუღლების დამატებითი მოწყობილობებით აღჭურვილ პერსონალურ კომპიუტერებს ***სათანადო პირობების შექმნის შემთხვევაში*** ნებისმიერი ამოცანის გადაწყვეტა შეუძლია, ოღონდ ხშირად ძნელია აღნიშნული პირობების უზრუნველყოფა.

VII თავი
ინფორმაციის გაცვლის ორგანიზაცია

7.1. მიკროპროცესორულ სისტემათა სალტეებით ინფორმაციის გაცვლის ციკლები

მიკროპროცესორული სისტემების აპარატურული ნაწილის ასაგებად, რომლის გარეშეც მათში ვერ იმუშავებს ვერცერთი პროგრამული უზრუნველყოფა, საჭიროა ვიცოდეთ ამ სისტემათა სალტეებით ინფორმაციის გაცვლის ორგანიზების პრინციპები.

პირველი მიკროპროცესორების არსებობის ***40***-ზე მეტი წლის განმავლობაში გამომუშავებული იქნა ინფორმაციის გაცვლის გარკვეული წესები, რომლებიც უნდა დავიცვათ ახალი მიკროპროცესორული სისტემების დამუშავებისათვის. ეს წესები ძალიან რთული არ არის, მაგრამ სამუშაოს წარმატებულად შესასრულებლად მათი ცოდნა და განუხრელი დაცვაა საჭირო. პრაქტიკამ გვიჩვენა, რომ ***სალტეებით ინფორმაციის გაცვლის პრინციპების ცოდნა კონკრეტული მიკროპროცესორების თავისებურებების ცოდნაზე უფრო მნიშვნელოვანია.*** სტანდარტული სისტემური მაგისტრალის სიცოცხლისუნარიანობა აღემატება ამა თუ იმ პროცესორის ანალოგურ პარამეტრს. ახალი პროცესორების შემქმნელები ორიენტაციას იღებენ უკვე არსებული სტანდარტის მაგისტრალებზე. უფრო მეტიც, სრულიად განსხვავებული პროცესორების გამოყენებით აგებული ზოგიერთი სისტემები ერთსა და იმავე სისტემურ სალტეს გამოიყენებს. მაშასადამე, ***მიკროპროცესორული სისტემების აგების პროცესში მაგისტრალი უმნიშვნელოვანეს როლს ასრულებს.***

ინფორმაცია მიკროპროცესორულ სისტემებში ინფორმაციის გაცვლის ციკლებში გაიცვლება. ***ინფორმაციის გაცვლის ციკლი*** ეწოდება დროის ინტერვალს, რომლის განმავლობაშიც სალტეში ინფორმაციის გაცვლის ერთი ელემენტარული ოპერაცია სრულდება. კერძოდ, პროცესორიდან მეხსიერებაში ან შეტანა/გამოტანის მოწყობილობიდან – პროცესორში მონაცემების კოდი ამ ციკლებში გადაიგზავნება. ერთი

ციკლის ფარგლებში შეიძლება მონაცემების რამდენიმე კოდი მონაცემთა მთელი მასივებიც კი გადაიგზავნოს. მაგრამ ეს იშვიათად ხდება.

განასხვავებენ ინფორმაციის გაცვლის შემდეგი ორი ტიპის ციკლს: *1) **ჩაწერის ციკლს,*** რომლის დროსაც პროცესორი ჩაწერს (გამოიტანს) ინფორმაციას; **2) *წაკითხვის (შეტანის) ციკლს,*** რომლის დროსაც პროცესორი კითხულობს (შეიტანს) ინფორმაციას.

ზოგიერთ მიკროპროცესორულ სისტემებში არსებობს ***«წაკითხვა-მოდიფიკაცია-ჩაწერის»*** ან ***«შეტანა-პაუზა-გამოტანის»*** ციკლიც. ამ ციკლებში პროცესორი ჯერ მეხსიერებიდან ან შეტანა/გამოტანის მოწყობილობიდან ამოიკითხავს ინფორმაციას, შემდეგ თავისებურად გარდაქმნის მას და ხელახლა ჩაწერს იმავე მისამართზე. მაგალითად, პროცესორმა შეიძლება მეხსიერების უჯრედიდან ამოიკითხოს კოდი, გაზარდოს იგი ერთი ერთეულით და ხელახლა ჩაწეროს იმავე უჯრედში. ასეთი ტიპის ციკლის არსებობა-არაარსებობა პროცესორის თავისებურებაზეა დამოკიდებული.

განსაკუთრებილ ადგილს იკავებს ***მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლის ციკლი*** (სისტემაში თუ არის გათვალისწინებული მეხსიერემასთან ინფორმაციის უსუალოდ გაცვლის შესაძლებლობა) და ***შეწყვეტის მოთხოვნისა და ამ მოთხოვნის დაკმაყოფილების ციკლი*** (შეწყვეტის არსებობის შემთხვევაში). თითოეული ციკლის განმავლობაში ინფორმაციის გამცვლელი მოწყობილობები საინფორმაციო და მმართველ სიგნალებს ერთმანეთს მკაცრად განსაზღვრული წესის, ანუ ***ინფორმაციის გაცვლის პროტოკოლის*** დაცვით აწვდის. გაცვლის ციკლის ხანგრძლივობა შეიძლება იყოს მუდმივი ან ცვლადი, მაგრამ იგი ყოველთვის შედგება სისტემის ტაქტური სიხშირის რამდენიმე პერიოდისაგან, ე. ი. იდეალური შემთხვევის დროსაც პროცესორის მიერ ინფორმაციის როგორც წაკითხვის, ისე ჩაწერის სიხშირე რამდენჯერმე მცირეა სისტემის ტაქტურ სიხშირეზე.

ბრძანებათა კოდები სისტემის მეხსიერებიდანაც წაკითხვის ციკლების დახმარებით წაიკითხება. ამიტომ ***ერთსალტური არქიტექტურის შემთხვევაში*** სისტემურ მაგისტრალში ურთიერთმონაცვლეობს ბრძანებების წაკითხვისა და გადაგზავნის (წაკითხვის, ჩაწერის) ციკლები, მაგრამ დამოუკიდებლად იმისა, მონაცემები გაიგზავნება თუ ბრძანებები, უცვლელი რჩება ინფორმაციის გაცვლის პროტოკოლები. ***ორსალტური არქიტექტურის შემთხვევაში*** ბრძანებების წაკითხვისა

და მონაცემების ჩაწერის ან წაკითხვის ციკლები სხვადასხვა სალტეებშია გადანაწილებული და ისინი შეიძლება ერთდროულად შესრულდეს.

7.2. მიკროპროცესორულ სისტემათა სალტეები

ინფორმაციის გაცვლის ციკლების თავისებურებათა განხილვამდე გავეცნოთ მიკროპროცესორული სისტემების სხვადასხვა სალტეების შემადგელობასა და გავარკვიოთ მათი დანიშნულება.

მიკროპროცესორული სისტემის სისტემური მაგისტრალი (სისტემური სალტე), როგორც აღვნიშნეთ, შედგება სამი ძირითადი, კერძოდ, მონაცემების, მისამართისა და მართვის სალტისაგან.

მონაცემების სალტე – ძირითადი სალტეა, რომლისთვისაც იქმნება მთელი სისტემა. მისი თანრიგების (კავშირების არხების) რაოდენობა განსაზღვრავს ინფორმაციის გაცვლის სიჩქარეს, ეფექტურობასა და ბრძანებათა შესაძლო რაოდენობას.

მიმართულების სალტე ყოველთვის ორმხრივმიმართულია. რადგან ინფორმაცია ორივე მიმართულებით შეიძლება გადაიცეს. ამ სალტის ხაზების გამოსასვლელ კასკადად ხშირად გამოიყენება ***3-მდგომარეობიანი (3S ტიპის) გამოსასვლელი კასკადი*** (იხ.***§1.4.2***).

მონაცემების სალტეს ჩვეულებრივ აქვს ***8, 16, 32*** ან ***64*** თანრიგი. მათი საშუალებით ინფორმაციის გაცვლის ერთი ციკლის განმავლობაში შესაბამისად გადაიცემა ***1, 2, 4*** და ***8*** ბიტის ტოლი ინფორმაცია. მონაცემების სალტის თანრიგიანობა ემთხვევა მთელი მაგისტრალის თანრიგიანობასაც. მაგალითად, ***32***-თანრიგიან სისტემურ სალტეზე საუბრისას იგულისხმება, რომ მისამართების სალტეც ***32***-***თანრიგიანია.***

მისამართის სალტე – მნიშვნელობით მეორეა, რომელიც განსაზღვრავს მიკროპროცესორული სისტემის მოსალოდნელ სირთულეს ანუ მეხსიერების დასაშვებ მოცულობას; სწორედ ამ უკანასკნელზეა დამოკიდებულია პროგრამის შესაძლო სიდიდე და მონაცემების ის მოცულობა, რომლის დამახსოვრებაცაა შესაძლებელი. მისამართის ***N*** თანრიგიანმა სალტემ შეიძლება 2^N რაოდენობის ობიექტი დაამისამართოს. მაგალითად, მისამართის ***16***-თანრიგიანი სალტის შემთხვევაში გვაქვს ***65536*** მისამართი. მისამართის სალტის თანრიგიანობა ჩვეულებრივ ***4***-ის ჯერადია და შეიძლება მიაღწიოს ***32***-ს ან ***64***-ს.

მისამართის სალტე შეიძლება იყოს ერთმხრივ მიმართული (მხოლოდ პროცესორით მაგისტრალის მართვის დროს) ან ორმხრივმიმართული (სხვა მოწყობილობისათვის, მაგალითად, მეხსიერებასთან პირდაპირ დაშვების კონტროლერისათვის, მაგისტრალის დროებით მართვის გადაცემის შესაძლებლობის დროს). ამ სალტის ხაზების გამოსასვლელ კასკადად ხშირად გამოიყენება ***3-მდგომარეობიანი (3S ტიპის) გამოსასვლელი კასკადი,*** ან ჩვეულებრივი ***ტტლ (2--მდგომარეობიანი, 2S ტიპის) კასკადი*** (იხ.***§1.4.2***).

ნახ. 7.1. *მისამართისა და მონაცემების სალტეთა მულტიპლექსირება*

მაგისტრალის კავშირების არხთა საერთო რაოდენობის შესამცირებლად ხშირად ახდენენ მისამართისა და მომაცემების სალტეების ***მულტიპლექსირებას,*** რომლის დროსაც მისამართებიცა და მონაცემებიც კავშირის ერთი და იმავე არხით, ოღონდ დროის სხვადასხვა მომენტებში გადაიცემა (ციკლის დასაწყისში გადაიცემა მისამართი, ბოლოში კი – მონაცემები). ამ მომენტების ფიქსირებისათის (ანუ, როგორც ამბობენ, ***სტრობირებისათვის***) მართვის სალტით გადაიცემა სპეციალური სიგნალი. ნათელია, რომ მულტიპლექსირებული სალტის გამოყენებისას მცირდება ინფორმაციის გაცვლის სიჩქარე და იზრდება გაცვლის ციკლის ხანგრძლივობა (ნახ. ***7.1***). მისამართისა და მონაცემების სალტეების ტიპებზე დამოკიდებულებით არსებობს მულტიპლექსირებული და არამულტიპლექსირებული მაგისტრალები.

ზოგიერთ მულტიპლექსირებულ მაგისტრალში მისამართის ერთი კოდის შემდეგ მონაცემების რამდენიმე კოდის (ან მონაცემების მასივის) მისამართი გადაიცემა. ეს მნიშვნელოვნად ამაღლებს მაგიტრალის სწრაფმოქმედებას. ზოგჯერ მაგისტრალში გამოიყენება ნაწილობ-

რივი მულტიპლექსირება ანუ მონაცემების თანრიგთა ერთი ნაწილი გადაიცემა მისამართების არამულტიპლექსური, მეორე ნაწილი კი – მულტიპლექსური ხაზებით.

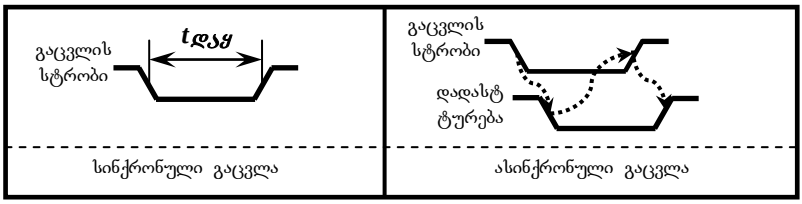
მართვის სალტე. დამხმარე სალტეა, რომელშიც ცირკულირებს მიმდინარე ციკლის ტიპის განმსაზღვრელი და ციკლის ურთიერთტოლი ნაწილების დროითი მომენტების მაფიქსირებელი მმართველი სიგნალები. გარდა ამისა, მმართველი სიგნალები მეხსიერების ან შეტანა/გამოტანის (შემსრულებელი) მოწყობილობების (***slave***-ს) მუშაობასთან ათანხმებს პროცესორის (ან მაგისტრალის სხვა მფლობელის, მავალებლის, ***master***-ის) მუშაობას.

მართვის სიგნალები იშვიათად გადაიცემა დადებითი, ხოლო უმეტესწილად - უარყოფითი ლოგიკის გამოყენებით. სალტის ხაზები შეიძლება იყოს როგორც ერთ- ასევე ორმხრივმიმართული. მათთვის გამოიყენება სხვადასხვა ტიპის გამოსასვლელი კასკადები; კერძოდ, ***2S ტიპის კასკადი*** გამოიყენება ერთმიმართული ხაზებისათვის, ხოლო ***3S ტიპის კასკადი*** – ორმხრივმიმართული ხაზებისათვის; ამ უკანასკნელებისათვის შეიძლება ***ღია კოლექტორიანი*** (***OC***-ტიპის)***გამოსასვლელი კასკადის*** გამოყენებაც (იხ.***§1.4.2***).

უმთავრესი მმართველი სიგნალებია გაცვლის ***სტრობები***, რომლებიც წარმოადგენს პროცესორის მიერ ფორმირებულ და დროის იმ მომენტების განმსაზღვრელ სიგნალებს, რომელთა დროსაც მონაცემების სალტეში გადაიგზავნება და გაიცვლება მონაცემები. უმეტესწილად მაგისტრალში გაცვლის შემდეგი ორი სახის სტრობი გამოიყენება: ***1) ჩაწერის (გამოტანის) სტრობი;*** იგი დროის იმ მომენტს განსაზღვრავს, როდესაც შემსრულებელ მოწყობილობას შეუძლია პროცესორის მიერ მონაცემების სალტეზე გამოტანილი მონაცემების მიღება; ***2) წაკითხვის (შეტანის) სტრობი;*** იგი დროის იმ მომენტს განსაზღვრავს, როდესაც შემსრულებელმა მოწყობილობამ მონაცემების სალტეზე უნდა გამოიტანოს პროცესორის მიერ წასაკითხი მონაცემების კოდი.

ამავე დროს დიდი მნიშვნელობა აქვს ციკლის ფარგლებში პროცესორი თუ როგორ ამთავრებს გაცვლას, რომელ მომენტში მიიღებს იგი გაცვლის სტრობს. შესაძლებელია ამის გადაწყვეტის ორი შემდეგი გზა (ნახ. *7.2*): ***1) სინქრონული გაცვლის*** გზა; ამ დროს პროცესორი მონაცემების გაცვლას ამთავრებს ***დაყოვნების*** ერთხელ

და სამუდამოდ დადგენილი ***tდაყ***-ს ტოლი ინტერვალის შემდეგ, ე. ი. შემსრულებელი მოწყობილობის ინტერესის გაუთვალისწინებლად; ***2) ასინქრონული გაცვლის*** გზა; ამ დროს პროცესორი გაცვლას ამთავრებს მხოლოდ შემსრულებელი მოწყობილობის მიერ გამოგზავნილი ოპერაციის დამთავრების მაუწყებელი სპეციალური სიგნალის მიღების შემდეგ. ამას ეწოდება «ხელის ჩამორთმევის» ანუ «***hands-hake***»-ს რეჟიმი.



ნახ. 7.2. *ინფორმაციის სინქრონული და ასინქრონული გაცვლა*

სინქრონული გაცვლის ღირსებაა გაცვლის უფრო მარტივი პროტოკოლი, მცირე რაოდენობის მმართველი სიგნალების არსებობა. ***ნაკლია*** შემსრულებლის მიერ ოპერაციის დასრულების გარანტიის არარსებობა, აგრეთვე ის, რომ შემსრულებელს უნდა ჰქონდეს მაღალი სწრაფმოქმედება.

ასინქრონული გაცვლის ღირსებაა მონაცემების გადაგზავნის უფრო მაღალი საიმედოობა, სხვადასხვა სწრაფმოქმედების შემსრულებლებთან მუშაობის შესაძლებლობა. ***ნაკლია*** თითოეული მომხმარებლის მიერ დადასტურების სიგნალის ფორმირების აუცილებლობა, რაც წარმოშობს დამატებით აპარატურული ხარჯების საჭიროებას.

გაცვლის აღნიშნული სახეების სწრაფმოქმედების ცალსახად შეფასება შეუძლებელია. ***ინფორმაციის ასინქრონული გაცვლის*** დროს საჭიროა დამატებითი დრო სიგნალის გამომუშავება-გადაცემისა და პროცესორის მიერ მისი დამუშავებისათვის. ***სინქრონული გაცვლის დროს*** ხელოვნურად უნდა გავზარდოთ ***გაცვლის სტრობის*** ხანგრძლივობა იმისათვის, რომ დიდი რაოდენობის შემსრულებელმა მოწყობილობებმა პროცესორის ტემპში მოასწროს ინფორმაციის გაცვლა. აღნიშნულის გამო ზოგჯერ მაგისტრალში გათვალისწინებულია როგორც სინქრონული, ისე ასინქრონული გაცვლის შესაძლებლობა. ამ დროს სინქრონული გაცვლა არის ძირითადი და საკმაოდ

სწრაფი, ხოლო ასინქრონული გაცვლა მხოლოდ ნელმოქმედი შემსრულებლებისათვის გამოიყენება.

გაცვლის გამოყენებულ სახეზე დამოკიდებულებით განასხვავებენ მიკროპროცესორული სისტემების სინქრონულ და ასინქრონულ მაგისტრალებს.

7.3. ინფორმაციის გაცვლის ციკლები

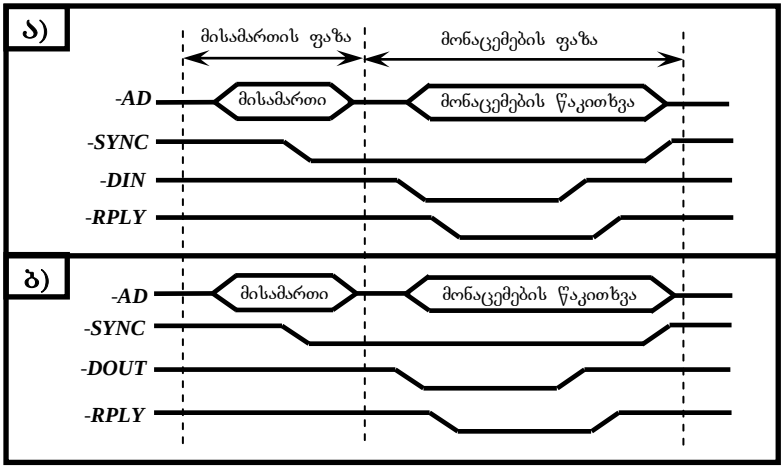
7.3.1. ინფორმაციის პროგრამული გაცვლა

ინფორმაციის პროგრამული გაცვლის პროცესი სიმარტივისათვის კერძო ორი ტიპური მაგალითის ფორმით განვიხილოთ. პირველ მაგალითში გაცვლისათვის გამოვიყენებთ ფირმა ***DEC***–ის (***Digital Equipment Corporation***) მიერ დამუშვებულ ***Q***-სალტეს, ხოლო მეორე მაგალითში - ფირმა ***IBM***-ის მიერ შემოთავაზებულ სინქრონულ არამულტიპლექსირებულ ***ISA (Industrial Standard Architecture)*** სალტეს**.** აღვნიშნავთ, რომ ***Q***–სალტე ფართოდ გამოიყენება როგორც მიკროკომპიუტერებში, ისე სამრეწველო კონტროლერებში, ხოლო ***ISA***-სალტე - პერსონალურ კომპიუტერებში. ორივე მაგალითში სიგნალის სახელწოდების წინ «***მინუსი» ნიშნის*** არსებობა ნიშნავს რომ აქტიურ სიგნალს აქვს დაბალი, ხოლო პასიურ სიგნალს – მაღალი დონე; **«*მინუსი» ნიშნის არარსებობისას*** აქტიურ სიგნალს აქვს მაღალი, ხოლო პასიურ სიგნალს – დაბალი დონე.

I მაგალითი. პროგრამული გაცვლა Q-სალტის გამოყენებით. ***Q***–სალტით ინფორმაციის გაცვლის პროცესში არსებული წაკითხვისა (შეტანისა) და ჩაწერის (გამოტანის) ციკლების გამარტივებული დროითი დიაგრამები ***7.3*** ნახაზზეა მოყვანილი.

ინფორმაციის გაცვლის ციკლის დასაწყისში (***მისამართის ფაზაში***) პროცესორი მისამართი/მონაცემების (***AD***) სალტეზე გამოიტანს მისამართის კოდს. ამ სალტეზე გამოიყენება ***უარყოფითი ლოგიკა***. ***AD*** სალტეზე სიგნალების საშუალო დონე იმას აღნიშნავს, რომ დროის მოცემულ ინტერვალებში არსებული სიგნალების მდგომარეობა არაა მნიშვნელოვანი. მისამართის ***სტრობირებისათვის*** გამოიყენება ასევე პროცესორის მიერ გამოტანილი უარყოფითი -***SYNC*** სიგნალი. მისი წინა (უარყოფითი) ფრონტი გვიჩვენებს, რომ ***AD*** სალტეზე გამოტა-

ნილია საჭირო (მოქმედი) მისამართი. მისამართის ფაზა ერთნაირია ჩაწერისა და წაკითხვის ციკლებში.



***ნახ. 7.3. Q-** სალტით წაკითხვისა (**ა**) და ჩაწერის (**ბ**) ციკლი*

საკუთარ კოდის მიღების (ამოცნობის) შემდეგ შეტანა/გამოტანის მოწყობილობა ან მეხსიერება (შემსრულებელი) დაიწყებს მზადებას ინფორმაციის გასაცვლელად. სტრობირების ***-SYNC*** სიგნალის დაწყებიდან (მისი უარყოფითი ფრონტიდან) გარკვეული დროის გავლის შემდეგ პროცესორი მოხსნის მისამართს და დაიწყებს ***მონაცემების ფაზას.***

წაკითხვის ციკლის მონაცემების ფაზაში (ნახ. ***7.3,ა***) პროცესორი გამოიტანს მონაცემების წაკითხვის სტრობის ***-DIN*** სიგნალს, რომლის საპასუხოდ იმ მოწყობილობამ, რომელსაც მიმართავს პროცესორი (შემსრულებელი) უნდა გამოიტანოს მონაცემების (წასაკითხი მონაცემების) კუთვნილი კოდი. იმავდროულად ამ მოწყობილობამ გაცვლის დასამოწმებელი ***-RPLY*** სიგნალით უნდა დაადასტუროს ოპერაციის შესრულება.

შემსრულებელ მოწყობილობებს შორის რომ არ წარმოიშვას კონფლიქტები, ***-RPLY*** სიგნალისათვის გამოსასვლელ კასკადად გამოიყენება ღია კოლექტორიანი (***OC ტიპის***, იხ.***§1.4.2***) კასკადი. პროცესორი ***-RPLY*** სიგნალის მიღების შემდეგ ამთავრებს მუშაობას. ამისათვის იგი მოხსნის ***-DIN*** და ***-SYNC*** სიგნალებს. შემსრულებე-

ლმა მოწყობილობამ ***-DIN*** სიგნალის მოხსნის საპასუხოდ ***AD*** სალტედან უნდა მოხსნას მონაცემების კოდი და დაასრულოს დადასტურების ***-RPLY*** სიგნალი.

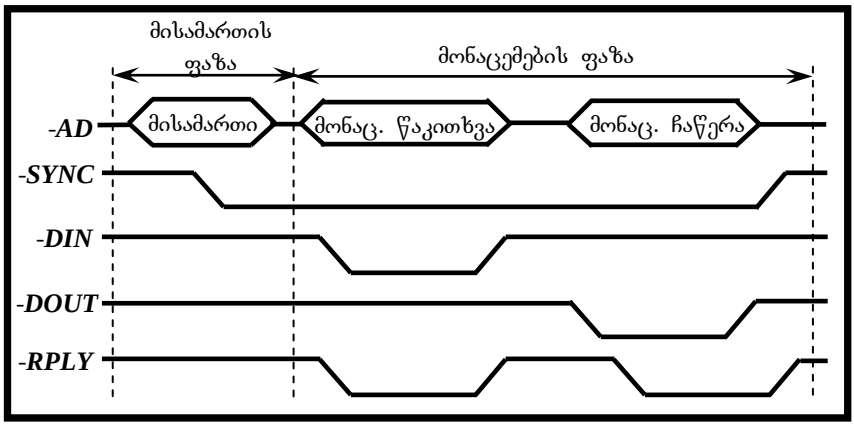
ჩაწერის ციკლის მონაცემების ფაზაში (ნახ.***7.3,ბ***) პროცესორი ***AD*** სალტეზე გამოიტანს ჩასაწერი მონაცემების კოდს, რომელსაც თან ახლავს მონაცემების ჩაწერის სტრობის უარყოფით ***-DOUT*** სიგნალი. შემსრულებელმა მოწყობილობამ ამ სიგნალის მიხედვით პროცესორიდან უნდა მიიღოს მონაცემებ და წარმოშვას ინფორმაციის გაცვლის დამადასტურებელ ***-RPLY*** სიგნალი. სიგნალ ***-RPLY***-ის მიღების შემდეგ პროცესორი ამთავრებს ინფორმაციის გაცვლის ციკლს. ამისათვის იგი ***AD*** სალტიდან მოხსნის მონაცემების კოდსა და ***-DOUT*** სიგნალს. ამ უკანასკნელის მოხსნის საპასუხოდ შემსრულებელმა მოწყობილობამ უნდა დაამთავროს მადასტურებელი ***–RPLY*** სიგნალი; და ბოლოს, პროცესორი მოხსნის ***-SYNC*** სიგნალს.

ზემოთ აღნიშნულის თანახმად, განხილული სალტით მისამართი გადაიცემა სინქრონულად (მომხმარებლის მიერ მისი მიღების დაუდასტურებლად), ხოლო მონაცემები – ასინქრონულად, ე. ი. შემსრულებელმა მოწყობილობამ აუცილებლად უნდა დაადასტუროს რომ მან გასცა ან მიიღო მონაცემები. დადგენილი დროის განმავლობაში დადასტურების ***-RPLY*** სიგნალის არარსებობას პროცესორი ავარიულ სიტუაციად აღიქვამს. პრინციპულად შესაძლებელია მისამართის ასინქრონულად გადაცემაც. ამით გაიზრდება გაცვლის საიმედოობა, მაგრამ შეიძლება შემცირდეს გაცვლის სიჩქარე.

Q-bus მაგისტრალზე წაკითხვისა და ჩაწერის ციკლების გარდა გამოიყენება «***შეტანა-პაუზა -გამოტანის***» («***მოდიფიცირება-ჩაწერა***») ტიპის ციკლი. მისი გამარტივებული დიაგრამა ***7.4*** ნახაზზეა წარმოდგენილი.

ამ ციკლში ***სამისამართო ფაზა*** ზუსტად ისევე სრულდება, როგორც წაკითხვისა (შეტანის) და ჩაწერის (გამოტანის) ციკლებში. ***მონაცემების ფაზაში*** კი პროცესორი ჯერ წაიკითხავს სამისამართო ფაზაში მითითებულ მისამართზე არსებულ მონაცემს, დაამუშავებს მას და შემდეგ მას იმავე მისამართზე ჩაწერს. წაკითხვისათვის გამოიყენება წაკითხვის ***-DIN*** სტრობი, ხოლო ჩაწერისათვის – ჩაწერის ***-DOUT*** სტრობი. შემსრულებელი მოწყობილობა ***-DIN*** სიგნალის საპასუხოდ საკუთარ მონაცემებს გაიტანს ***AD*** სალტეზე, ხოლო

-DOUT სიგნალის საპასუხოდ იგი მიიღებს მონაცემებს ***AD*** სალტედან. წაკითხვისა და ჩაწერის ციკლების ანალოგურად შემსრულებელი მოწყობილობა მის მიერ თითოეული ოპერაციის შესრულებას ადასტურებს ***-RPLY*** სიგნალით. «შეტანა-პაუზა-გამოტანის» ციკლის სიდიდე აღემატება წაკითხვისა და ჩაწერის ცალ-ცალკე აღებული ციკლების სიდიდეებს, მაგრამ ჩამოუვარდება ამ ციკლების ჯამურ სიდიდეს (რადგან მისთვის მხოლოდ ერთი სამისამართო ფაზაა საკმარისი). პროცესორი «შეტანა-პაუზა-გამოტანის» ციკლის დასაწყისში გამოიმუშავებს ***–SYNC*** სიგნალს და მას შეინახავს მთელი ციკლის დამთავრებამდე.

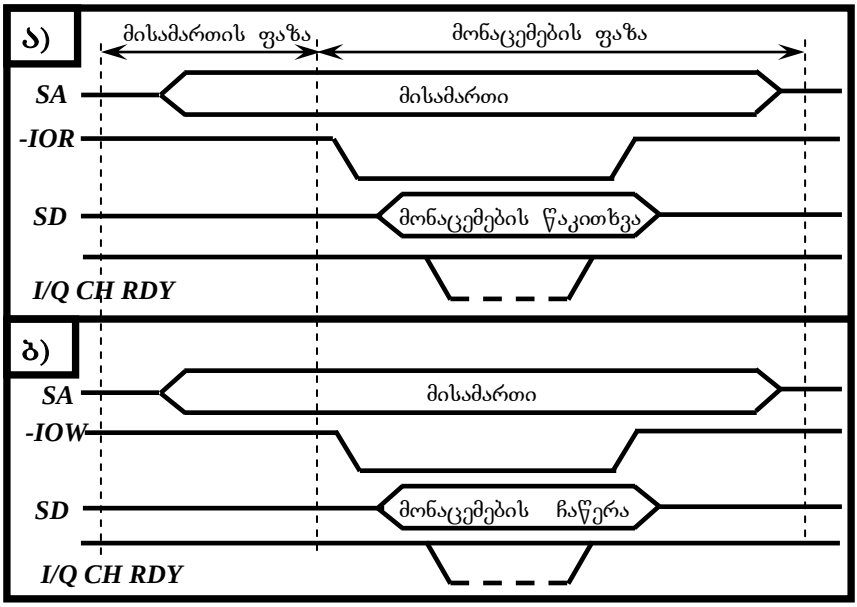


ნახ. 7.4. *ციკლი "შეტანა-პაუზა-გამოტანა" Q-სალტით*

II მაგალითი. პროგრამული გაცვლა ISA სალტის გამოყენებით. ***ISA*** სალტით შეტანა/გამოტანის მოწყობილობაში ჩაწერისა ამ მოწყობილობიდან წაკითხვის გამარტივებული ციკლები ***7.5*** ნახაზზეა მოყვანილი.

ორივე ციკლი იწყება პროცესორის (მავალებლის) მიერ მისამართის ***SA*** სალტეზე მისამართის კოდის გამოტანით (ამ სალტეზე ლოგიკა დადებითია). მისამართი ***SA*** სალტეზე რჩება ციკლის დამთავრებამდე. ორივე ციკლისათვის ერთნაირი ***მისამართის ფაზა*** მთავრდება მონაცემების გაცვლის ***-IOR*** ან ***-IOW*** სტრობის დაწყებისას. მისამართის ფაზის განმავლობაში შემსრულებელმა მოწყობილობამ უნდა მიიღოს მისამართის კოდი და ამოიცნოს ან ვერ ამოიცნოს იგი. ამო-

ცნობის შემთხვევაში შემსრულებელი მოწყობილობა დაიწყებს ინფორმაციის გაცვლისათვის მზადებას.



***ნახ. 7.5. ISA** –ს სალტით შეტანა/გამოტანის მოწყობილობიდან წაკითხვისა (**ა**) და მასში ჩაწერის (**ბ**) ციკლი*

წაკითხვის ციკლის მონაცემების ფაზაში (ნახ. ***7.5,ა***) პროცესორი გამოიტანს შეტანა/გამოტანის მოწყობილობიდან მონაცემების წაკითხვის ***-IOR*** სიგნალს. ამის საპასუხოდ შემსრულებელმა მოწყობილობამ მონაცემის ***SD*** სალტეზე უნდა გასცეს მონაცემების საკუთარი კოდი (წასაკითხი მონაცემები). მონაცემების სალტეზე ლოგიკა დადებითია. დადგენილი დროის გავლის შემდეგ პროცესორი მოხსნის გაცვლის ***-IOR*** სტრობს, რომლის შემდეგ ***SA*** სალტიდანაც მოიხსნება მისამართის კოდი. ციკლი მთავრდება შემსრულებლის სწრაფმოქმედების გაუთვალისწინებლად.

ზემოთ აღწერილი პროცესი მიმდინარეობს ***ინფორმაციის სინქრონული გაცვლის დროს***, მაგრამ ***ISA*** მაგისტრალზე ინფორმაციის ასინქრონული გაცვლაცაა გათვალისწინებული. ამისათვის გამოიყენება არხის (მაგისტრალის) მზადყოფნობის ***I/Q CH RDY*** სიგნალი. შემ-

სრულებელ მოწყობილობებს შორის კონფლიქტების წარმოშობის თავიდან ასაცილებლად აღნიშნული სიგნალის გამოსასვლელ კასკადად გამოიყენება ***OC-ტიპის კასკადი*** (იხ.***§1.4.2***). სინქრონული გაცვლის დროს ***I/Q CH RDY*** სიგნალი ყოველთვის დადებითია, მაგრამ ნელმოქმედ შემსრულებელ მოწყობილობას, რომელიც პროცესორის ტემპით ვერ მუშაობს, შეუძლია ეს სიგნალი მოხსნას, ე. ი. იგი ნულოვანი გახადოს ინფორმაციის გაცვლის სტრობის დაწყებისთანავე. ასეთი სიტუაციის დროს პროცესორი გაახანგრძლივებს გაცვლის სტრობს და ***I/Q CH RDY*** სიგნალის ხელახლა დადებითად გახდომამდე გადაავადებს ციკლის დამთავრებას. სიმარტივისათვის შეიძლება ჩავთვალოთ, რომ შემსრულებელი მოწყობილობა მოცემულ შემთხვევაში წარმოქმნის გაცვლის დამთავრებისათვის მოუმზადებლობის მაჩვენებელ უარყოფით სიგნალს. ამ სიგნალის არსებობის განმავლობაში ჩერდება მაგისტრალზე გაცვლის პროცესი.

Q-bus მაგისტრალზე ასინქრონული გაცვლისაგან ***ISA*** მაგისტრალზე იგივე გაცვლა შემდეგი ნიშნით განსხვავდება. ***Q-bus***-ის დროს დადასტურების სიგნალი აუცილებელია, ხოლო ***ISA***-ს შემთხვევაში მოუმზადებლობის შესახებ ნიშანი შემსრულებელი არ წარმოქმნის თუ იგი შეძლებს პროცესორის ტემპით მუშაობას. სამაგიეროდ ციკლის ყოველი დასრულების მომენტში პროცესორი ***Q-bus***-ის შემთხვევაში ყოველთვის დარწმუნებულია, რომ შემსრულებელმა მოწყობილობამ მოასწრო მოთხოვნილი ოპერაციის შესრულება, ხოლო ***ISA***-ს შემთხვევაში არა.

ჩაწერის ციკლის მონაცემების ფაზაში (ნახ.***7.5,ბ***) პროცესორი შეტანა/გამოტანის მოწყობილობაში ჩაწერის ***–IOW*** სტრობის თანხლებით მონაცემის ***SD*** სალტეზე გამოიტანს ჩასაწერი მონაცემების კოდს. ამ სიგნალის მიღების შემდეგ შემსრულებელმა მოწყობილობამ ***SD*** სალტედან უნდა შეიტანოს ჩასაწერი მონაცემების კოდი. ეს თუ პროცესორი მუშაობის ტემპით ვერ განახორციელა, მაშინ აღნიშნულ მოწყობილობას შეუძლია ***-IOW*** სიგნალის წინა ფრონტის მიღების შემდეგ საჭირო დროით მოხსნას ***I/Q CH RDY*** სიგნალი, რის შედეგადაც პროცესორი გადაავადებს ჩაწერის ციკლის დამთავრებას.

განხილული მაგალითები ვერ ამოწურავს ნახსენები მაგისტრალებით ინფორმაციის გაცვლის ყველა დეტალს. მათი საშუალებით მხოლოდ გაცვლის ძირითადი პრინციპებია ილუსტრირებული.

7.3.2. შეწყვეტით ინფორმაციის გაცვლა

შეწყვეტის რეჟიმში ინფორმაცია იმავე პრინციპებით გაიცვლება როგორც პროგრამული გაცვლის დროს, მაგრამ მას ახასიათებს მთელი რიგი სპეციფიკური თავისებურება.

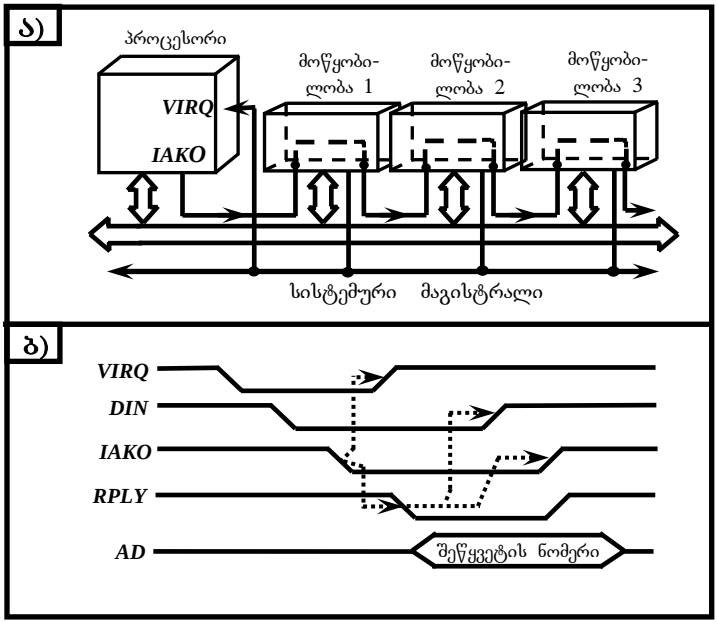
მიკროპროცესორულ სისტემებში ძირითადად არსებობს შემდეგი ორი სახის შეწყვეტები: ***1) ვექტორული შეწყვეტა,*** რომელის დროსაც აუცილებელია მაგისტრალით წაკითხვის ციკლის ორგანიზება; ***2) რადიალური შეწყვეტა,*** რომელთა დროსაც მაგისტრალით რაიმე ციკლის ორგანიზება საჭირო არ არის.

ზოგადად, მიკროპროცესორულ სისტემებში ხშირია შეწყვეტა, ამიტომ პროცესორს სჭირდება ჰქონდეს ინფორმაცია კონკრეტული შეწყვეტის ***ნომრის*** (ანუ, როგორც ხშირად ამბობენ, ***ვექტორის მისამართის***) შესახებ.

ვექტორული შეწყვეტის დროს შეწყვეტის ნომრის კოდი პროცესორს შეწყვეტის გამომთხოვი შეტანა/გამოტანის მოწყობილობისაგან გადაეცემა. ამისათვის პროცესორი ჩაატარებს მაგისტრალით წაკითხვის პროცესს და მონაცემების სალტით იღებს შეწყვეტის ნომრის კოდს. შეწყვეტის გამომთხოვმა მოწყობილობამ ვინაიდან არ იცის, მიაქცევს თუ არა მას ყურადღებას პროცესორი, ამიტომ მოცემულ ციკლში არ გამოიყენება მისამართის სალტე. ამ შემთხვევაში შეტანა/გამოტანის ყველა მოწყობილობისათვის საკმარისია მაგისტრალში შეწყვეტის მხოლოდ ერთი გამოთხოვა არებობდეს. ასეა, მაგალითად, ***Q-bus*** მაგისტრალში,

შეწყვეტებში მონაწილე სიგნალების ***Q-bus*** მაგისტრალში გავრცელების სქემა ***7.6,ა*** ნახაზზე, ხოლო შეწყვეტის ციკლის მოთხოვნისა და მიწოდების გამარტივებული დროითი დიაგრამა ***7.6,ბ*** ნახაზზეა მოყვანილი.

შეწყვეტა გამოითხოვება უარყოფითი ***-VIRQ*** სიგნალით, რომლის ფორმირება შეუძლია შეწყვეტის მსურველ ნებისმიერ მოწყობილობას. მათ შორის კონფლიქტის წარმოშობის თავიდან ასაცილებლად ***-VIRQ*** სიგნალისათვის გამოსასვლელ კასკადად გამოიყენება ***ღია კოლექტორიანი გამოსასვლელი OC კასკადი.*** სიგნალ ***-VIRQ***-ს მიღების შემდეგ პროცესორი მიმდინარე ბრძანების შესრულების დამთავრების შემდეგ მოწყობილობას მისცემს შეწყვეტის ნებას; ამი-



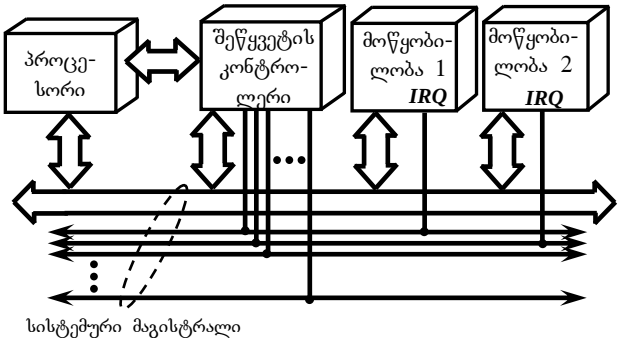
ნახ.7.6. *ვექტორულ შეწყვეტებში მონაწილე სიგნალების* **Q-bus** *მაგისტრალში გავრცელების სქემა (**ა**) და აღნიშნული შეწყვეტის მოთხოვნა/მიწოდების ციკლის გამარტივებული დროითი დიაგრამა (**ბ**)*

სათვის ის გამოიტანს მონაცემების წაკითხვის **–*DIN*** და შეწყვეტის ნებადართვის **-*IAKO*** სიგნალს. უკანასკნელი სიგნალი შეეცდება მიმდევრობით გაიაროს პოტენციალურად შეწყვეტის გამომთხოვი ყველა მოწყობილობა, მაგრამ მას გზას გადაუკეტავს შეწყვეტის რეალურად გამომთხოვი მოწყობილობა. ამის შედეგად აღმოჩნდება, რომ თუ შეწყვეტა ერთდროულად გამოითხოვა რამდენიმე მოწყობილობამ, სიგნალს მიიღებს პროცესორთან ახლო მდგომი მხოლოდ ერთი მათგანი. კონფლიქტების გადაწყვეტის ასეთ მექანიზმს ზოგჯერ ***გეოგრაფიულ პრიორიტეტს*** (ან ***მწკივობრივ პრიორიტეტს***, **Daisy Chain**-ს) უწოდებენ. შეწყვეტის გამომთხოვმა მოწყობილობამ **-*IAKO*** სიგნალის მიღების შემდეგ უნდა მოხსნას საკუთარი **-*VIRQ*** სიგნალი.

აღნიშნულის შემდეგ პროცესორი შეასრულებს შეწყვეტის ნომრის წაკითხვის ***უმისამართოდ წაკითხვის ციკლს.*** მოწყობილობამ, რომელსაც გამოეყო შეწყვეტა, **–*DIN*** და **-*IAKO*** სიგნალების მიღების

შემდეგ მისამართი/მონაცემების ***AD*** სალტეზე უნდა გასცეს შეწყვეტის ნომერი (შეწყვეტის მისამართის კოდი) და გაიტანოს დადასტურების –***RPLY*** სიგნალი. პროცესორი წაიკითხავს ამ კოდს და –***DIN, -IAKO*** სიგნალების მოხსნის გზით დაასრულებს უმისამართოდ წაკითხვის ციკლს.

რადიალური შეწყვეტის დროს მაგისტრალში შეწყვეტის მოთხოვნის ხაზების რაოდენობა შესაძლო სხვადასხვა შეწყვეტის რაოდენობის ტოლია. მაშასადამე, შეწყვეტის სარგებლობის მსურველი შეტანა/გამოტანი თითოეული მოწყობილობა საკუთარი განცალკევებული არხით აგზავნის შეწყვეტის გამოთხოვის სიგნალს. პროცესორი შეწყვეტის ნომერს ამოიცნობს იმ ხაზის ნომრის მიხედვით, რომლიდანაც იგი იღებს შეწყვეტის სიგნალს. ამ დროს არ არსებობს მაგისტრალით ინფორმაციის გაცვლის არავითარი ციკლი. რადიალური შეწყვეტის დროს სისტემაში ჩვეულებრივ ჩაირთვება შეწყვეტის მოთხოვნის სიგნალების დამამუშავებელი კონტროლერი. სწორედ ასეა ორგანიზებული შეწყვეტები, მაგალითად, ***ISA*** მაგისტრალში.



ნახ.7.7. *მაგისტრალზე რადიალური შეწყვეტების ორგანიზებისათვის საჭირო შეწყვეტების სტრუქტურა*

მაგისტრალის დროს შეწყვეტით ინფორმაციის გაცვლაში მონაწილე მოწყობილობებს შორის არსებული კავშირების გამარტივებული სქემა ***7.7*** ნახაზზეა მოყვანილი. პროცესორი კონტროლერთან ურთიერთობას ამყარებს როგორც მაგისტრალით (მუშაობის რეჟიმის დასასახად), ისე უმაგისტრალოდაც (შეწყვეტათა მოთხონების დამუშავების დროს). შეწყვეტათა გამოთხოვათა ***IRQ*** სიგნალები მაგისტრალის ყველა მოწყობილობას შორის ნაწილდება. ***IRQ***-ის თითო-

ეულ ხაზზე მოდის ერთი მოწყობილობა. რადგან ამ ხაზებში კონფლიქტური სიტუაციები არ წარმოიქმნება, ამიტომ მათთვის გამოიყენება ***2S ტიპის გამოსასვლელი კასკადი*** (იხ.***§1.4.2***) შეწყვეტის მოთხოვნას წარმოადგენს ***IRQ*** სიგნალის წინა, დადებითი ფრონტი. რამდენიმე მოწყობილობიდან შეწყვეტის მოსვლის შემთხვევაში მათი შესრულების რიგითობას განსაზღვრავს კონტროლერი.

ორივე სახის შეწყვეტას აქვს როგორც დადებითი, ისე უარყოფითი მხარეები. კერძოდ, ***ვექტორული შეწყვეტა*** სისტემას აძლევს უფრო მეტ მოქნილობას, მაგრამ ამ დროს უმისამართო წაკითხვის ციკლების მომსახურებისათვის შეწყვეტის მომთხოვნ თითოეულ მოწყობილობაში საჭიროა დამატებითი აპარატურული კვანძების არსებობა. ***რადიალური შეწყვეტის*** რაოდენობა სისტემაში ძალიან ბევრი არ არის (მათი რაოდენობა ***1***-დან ***16***-მდე მერყეობს). ასეთი სახის შეწყვეტის დროს სისტემაში აუცლებლად უნდა არსებობდეს შეწყვეტის კონტროლერი. თითოეული რადიალური შეწყვეტა სისტემური მაგისტრალის მართვის სალტეში დამატებითი ხაზის არსებობას მოითხოვს, სამაგიეროდ რადიალურ შეწყვეტასთან მუშაობა ადვილია, რადგან ყველაფერი დაიყვანება ერთადერთი ***IRQ*** სიგნალის გამომუშავებაზე და ინფორმაციის გაცვლის არავითარი ციკლი მაგისტრალში საჭირო არ არის.

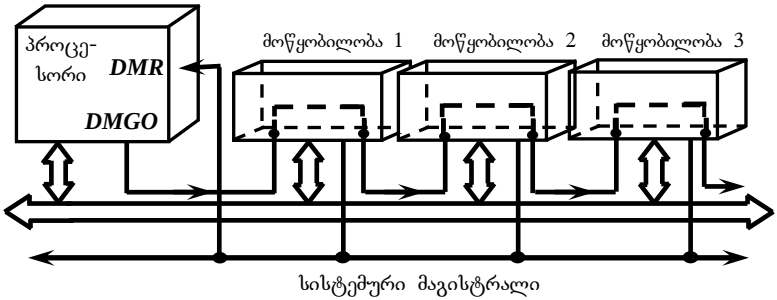
7.3.3. მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლა

მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლის (***DMA***) რეჟიმის ციკლები იმავე წესებით სრულდება, რა წესებითაც სრულდება პროგრამული გაცვლისა და შეწყვეტების მიწოდების ციკლები.

DMA-რეჟიმში გაცვლის დაწყებამდე მოწყობილობამ, რომელსაც ***DMA*** სჭირდება, უნდა მოითხოვოს და მიიღოს იგი. ***DMA***-ს მოთხოვისა და მიღების პროცედურა ძალიან ჰგავს შეწყვეტის მოთხოვნისა და მიღების პრცედურას. ორივე შემთხვევაში მომსახურების მომთხოვნმა მოწყობილობამ მოთხოვნის სიგნალი უნდა გაუგზავნოს პროცესორს; ოღონდ ***DMA***-ს შემთხვევაში პროცესორმა მომთხოვნ მოწყობილობას იგი სპეციალური სიგნალების დახმარებით უნდა მიაწოდოს, რადგან უშუალოდ გაცვლის განმავლობაში პროცესორი მა-

გისტრალისაგან გამორთულია. ***რადიალურ შეწყვეტის*** დროს პროცესორს არ მოეთხოვება მისი მიწოდება.

Q-bus მაგისტრალის შემთხვევაში ***DMA***-ს მოთხოვნა და მიწოდება შეწყვეტების მოთხოვნისა და მიწოდების მსგავსად ორგანიზდება. ***DMA***-ში მონაწილე მოწყობილობათა კავშირების გამარტივებული სტრუქტურა ***7.8*** ნახაზზეა ნაჩვენები. ***DMA***-ს გამოთხოვა/მიწოდების დროითი დიაგრამა ძალიან ჰგავს შეწყვეტების გამოთხოვა/მიწოდების დროით დიაგრამას (იხ. ნახ. ***7.6,ბ***).

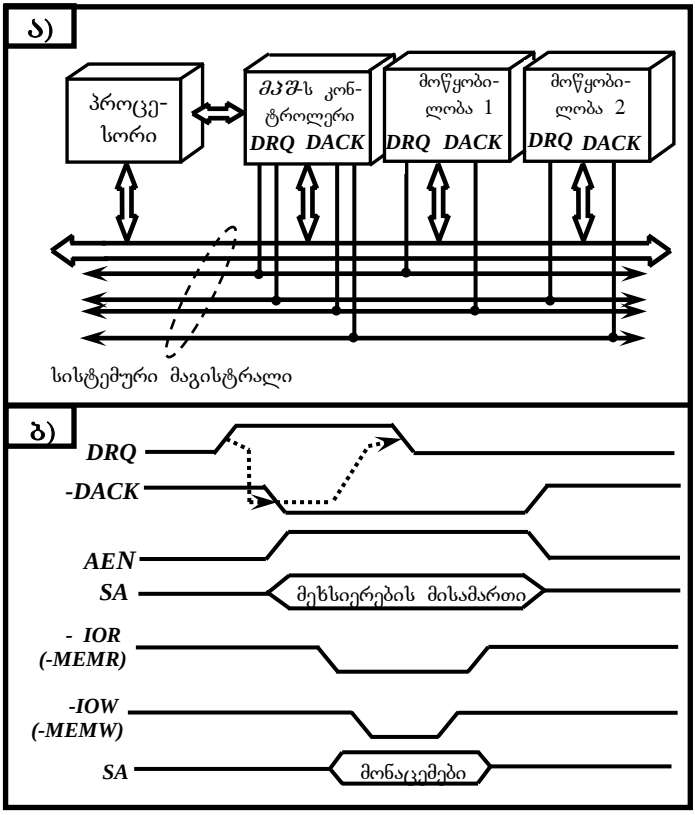


ნახ.7.8. Q-bus *მაგისტრალის დროს* ***მპშ****-ს გამოთხოვა/მიწოდების-პროცესში მონაწილე მოწყობილობათა კავშირების სტრუქტურა*

DMA-ს მოთხოვნის ***DMR*** სიგნალი საერთო ხაზით გადაეცემა ყველა მოწყობილობას, რომელსაც სჭირდება ***DMA***. ამ ხაზზე გამოიყენება ***OC ტიპის გამოსასვლელი კასკადი*** (იხ.***§1.4.2***). პროცესორი ***-DMR*** სიგნალის მიღების შემდეგ გასცემს ***DMA***-ს მიწოდების ***DMGO*** სიგნალს, რომელიც ***IAKO*** სიგნალის (იხ.ნახ.***7.6,ა***) ანალოგურია. ეს სიგნალიც ცდილობს, მიმდევრობით გაიაროს ყველა მოწყობილობაში, მაგრამ მას გზას გადაუკეტავს და დაისაკუთრებს პროცესორთან ახლოს მდებარე მოწყობილობა (***გეოგრაფიული პრიორიტეტი***). ამის შემდეგ ***DMA***-ს მიმღები მოწყობილობა პროგრამული გაცვლის ციკლების ანალოგურად ჩაატარებს მაგისტრალით გაცვლის ციკლებს. ***DMA***-ს ციკლებში ინფორმაცია წაიკითხება მეხსიერებიდან და ჩაიწერება შეტანა/გამოტანის მოწყობილობაში ან პირიქით – წაიკითხება შეტანა/გამოტანის მოწყობილობიდან და გადაიცემა მეხსიერებაში.

ISA მაგისტრალის შემთხვევაში ***DMA***-ს მოთხოვნა/მიწოდება ძალიან ჰგავს ***რადიალური შეწყვეტების*** რეალიზაციას (ნახ. ***7.9,ა***).

სისტემაში ზუსტად ისევე არსებობს ***DMA***-ს კონტროლერი. მასში თავს იყრის ***DMA***-ს გამოთხოვის ***-DRQ*** სიგნალები და მისგან გამოდის ***DMA***-ს მიწოდების ***-DACK*** სიგნალები. ***DMA***-ს თითოეულ არხთან (და სიგნალების წყვილთან) მიერთებულია ***DMA***-ს გამომთხოვი მხოლოდ ერთი მოწყობილობა. არხისათვის (სიგნალებისათვის) გამოიყენება ***2S გამოსასვლელი კასკადი*** (იხ.***§1.4.2***). მოწყობილობა, რომელმაც ინფორმაცია მეხსიერებასთან უშუალოდ უნდა გაცვალოს, აგზავნის მოთხოვნის ***-DRQ*** სიგნალს და საპასუხოდ იღებს მიწოდების ***-DACK*** სიგნალს. ამის შემდეგ ***DMA***-ს კონტროლერი ასრულებს შეტანა/გამოტანის მოწყობილობასა და მეხსიერებას შორის ინფორმაციის გაცვლის ციკლს.



ნახ. 7.9. *მაგისტრალის დროს* ***DMA****-ს მოთხოვნა/მიწოდების კავშირების სტრუქტურა (****ა****) და* ***DMA****-ს ციკლი (****ბ****)*

ISA მაგისტრალის შემთხვევაში ***DMA***-ს ციკლების გამარტივებული დროითი დიაგრამა ***7.9,ბ*** ნახაზზეა ნაჩვენები.

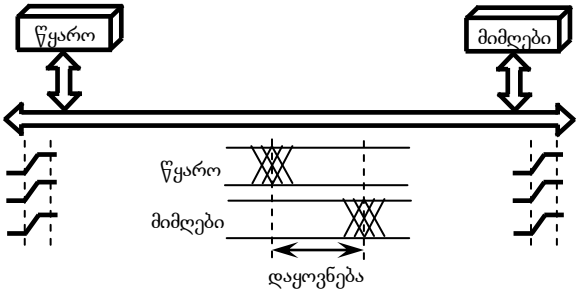
ISA მაგისტრალის შემთხვევაში მეხსიერებაში ჩაწერისათვის გამოიყენება (***-MEMW***) სტრობი, ხოლო შეტანა/გამოტანის მაგისტრალში ჩასაწერად - (***-IOW***) სტრობი.; ასევე, მეხსიერებიდან წასაკითხად გამოიყენება (***-MER***) სტრობი, ხოლო შეტანა/გამოტანის მოწყობილობიდან წასაკითხად - (***-IOR***) სტრობი. ეს საშუალებას იძლევა მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლის ერთ ციკლის განმავლობაში შევასრულოთ ინფორმაციის მეხსიერებიდან წაკითხვისა და შეტანა/გამოტანის მოწყობილობაში მისი ჩაწერის, ან შეტანა/გამოტანის მოწყობილობიდან ინფორმაციის წაითხვისა და მეხსიერებაში მისი ჩაწერის ოპერაციები. ამ დროს მისამართის სალტეზე გამოიტანება მეხსიერების მისამართი, ხოლო შეტანა/გამოტანის მისამართი შეიცვლება ერთადერთი ***AEN*** სიგნალით. ბუნებრივია, რომ მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლის რეჟიმში შეტანა/გამოტანის მხოლოდ ის მოწყობილობა მონაწილეობს, რომელიც წინასწარ მოითხოვა და ამის საფუძველზე მიიღო ***DMA***. ასეთი გამარტივებული დამისამართების გამო შეტანა/გამოტანის მოწყობილობებს შორის არავითარი კონფლიქტი არ წარმოიშობა.

7.4. მაგისტრალში სიგნალების გავრცელების პროცესი

მაგისტალებსა და სალტეებში ინფორმაციის გაცვლის ორგანიზების დროს უნდა გავითვალისწინოთ როგორც სალტეებში სიგნალების გავრცელებასთან, ისე თავად სალტეების ბუნებასთან დაკავშირებული რამდენიმე უმნიშვნელოვანესი მომენტი. საწინააღმდეგო შემთხვევაში სისტემაში შემავალი ციფრული მოწყობილობების მთელი ლოგიკის უშეცდომოდ დაპროექტების მიუხედავად მიკროპროცესორული სისტემა ან უბრალოდ არ იმუშავებს, ან იმუშავებს არამდგრადად.

მიკროპროცესორული სისტემის სალტე (მაგისტრალი) მიკროსქემაში თუ არ არის ჩაშენებული და გარედანაა გაყვანილი, მაშინ საჭიროა გავითვალისწინოთ გრძელ ხაზებში სიგნალების გავლის თავისებურებები. მიუხედავად იმისა, რომ მაგისტრალი ძალიან გრძელი არ არის (მისი სიგრძე ორ ათეულ სანტიმეტრს არ აჭარბებს), ზემოთ აღნიშნული თავისებურებების გათვალისწინება მაინც აუცილებელია

გაცვლის სინქრონიზებისათვის. მაგისტრალში სიგნალების გავრცელებაზე ზემოქმედებს შემდეგი ფაქტორები:



ნახ. 7.10. *სალტეში სიგნალების გავრცელების პროცესი*

1) მაგისტრალის ხაზებში სიგნალების გავრცელების დაყოვნებათა ჯამური სიდიდე; ***2)*** სალტის სხვადასხვა ხაზში სიგნალების ურთიერთგანსხვავებული დაყოვნებით გავრცელება; ***3)*** კავშირის ხაზებზე სიგნალების არაერთდროული გამოტანა; ***4***) მაგისტრალში გამავალ სიგნალთა ფრონტების დამახინჯება; ***5)*** კავშირის ხაზების ბოლოებიდან სიგნალების არეკვლა (ნახ. ***7.10***).

ყველა ამ ფაქტორის გასათვალისწინებლად გაცვლის სტანდარტული მაგისტრალებისა და პროტოკოლების შემქმნელები ყოველთვის ითვალისწინებენ ინფორმაციის გაცვლაში მონაწილე სიგნალებს შორის აუცილებელ დაყოვნებებს. ამასთანავე ამ დაყოვნებებს ისე ირჩევენ, რომ სიგნალის მიმღებ მოწყობილობას რჩებოდეს ამ სიგნალის დასამუშავებლად საკმარისი დრო. ყოველი ახალი მაგისტრალის დამუშავებისას ზემოთ აღნიშნულის გათვალისწინება სავალდებულოა.

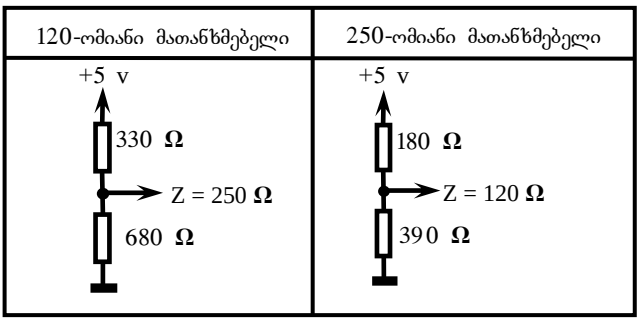
აღნიშნულიდან ნათელია, რომ დაყოვნებების შემცირების გზით რომელიმე სტანდარტული პროტოკოლის «მოდერნიზება» ძალიან სახიფათოა. ზუსტად ასევე სახიფათოა პროტოკოლის შეუცვლელად მაგისტრალის დაგრძელების მცდელობა, რომლის დროსაც იზრდება ხაზებსა და სალტეებში სიგნალის გავრცელების დაყოვნება. ამგვარი «მოდერნიზაციისადმი» განსაკუთრებით მგრძნობიარეა სინქრონული მაგისტრალები, რომლებშიც სავალდებულო არ არის თითოეული ოპერაციის შესრულების დადასტურება.

მაგალითად, ***გაცვლის ციკლში მისამართის ფაზის ხანგრძლივობა*** შემდეგნაირად ამოირჩევა. სამისამართო ფაზის განმავლობაში მისა-

მართის კოდის ყველა თანრიგის ყველა, თუნდაც პროცესორის მიერ არა ერთდროულად ფორმირებული, სიგნალი სალტის საკუთარი სადენებით უნდა მივიდეს შემსრულებელ მოწყობილობამდე; ხოლო ამ უკანასკნელმა უნდა მიიღოს და დაამუშაოს მისამართის ეს კოდი (ე.ი. უცხო მისამართისაგან უნდა განასხვაოს საკუთარი მისამართი). პროცესის გარანტირებულად შესრულებისათვის სამისამართო ფაზის რეალური ხანგრძლივობა რამდენადმე უნდა აღემატებოდეს მინიმალურად აუცილებელ ხანგრძლივობას.

ზუსტად ასევე, ***წაკითხვის ციკლში მონაცემების ფაზის ხანგრძლივობა*** ისე უნდა შეირჩეს, რომ მის განმავლობაში: ***1)*** შემსრულებელმა მოწყობილობამ მოასწროს წაკითხვის სტრობის მიღება და მონაცემების სალტეზე წაკითხული მონაცემების გატანა; ***2)*** აღნიშნული კოდი მივიდეს პროცესორამდე; ***3)*** პროცესორმა მოასწროს მისი წაიკითხვა.

ამის შემდეგ პროცესორი მოხსნის წაკითხვის სტრობს, სიგნალის ეს უკანა ფრონტი დაყოვნებით მივა შემსრულებელ მოწყობილობამდე, რომელიც ასევე დაყოვნებით მოხსნის მონაცემების საკუთარ კოდს. ანალოგიური პროცესი მიმდინარეობს ***ჩაწერის ციკლშიც.***



ნახ. 7.11. *Q-bus მაგისტრალის ბოლო მათანხმებლები*

მაგისტრალში გავრცელებული სიგნალების ფორმის გასაუმჯობესებლად ზოგჯერ მაგისტრალის ხაზის ბოლოებში გამოიყენება ბოლო მათანხმებლები (ტერმინატორები). მათი გამოყენება მაშინაა განსაკუთრებით მნიშვნელოვანი, როდესაც მაგისტრალის დასაშვები სიგრძე რამდენიმე მეტრს აღემატება. მაგალითად, ***Q-bus*** მაგისტრალის შემთხვევაში გამოიყენება **120**-ომიანი და ***220***-ომიანი მათანხმებლები (ნახ.***7.11***).

მათანხმებლების ჩართვის დროს იცვლება მაგისტრალის ხაზებზე მომუშავე გადამცემების სადატვირთვო უნარი. ***ISA*** მაგისტრალში მსგავსი მათანხმებლები არ გამოიყენება, თუმცა ზოგიერთ ხაზებზე მიერთებულია რეზისტორები, რომელთა მეორე ბოლოები მიერთებულია კვების სალტესთან (ასეთებია ხაზები, რომელთა გამოსასვლელ კასკადებად გამოიყენება ***OC ტიპის კასკადები,*** იხ.***§1.4.2***).

მაგისტრალს შეიძლება მიუერთდეს გამოსასვლელი დენის მომხმარებელი რამდენიმე მოწყობილობა, ამიტომ მაგისტრალის ხაზებზე მომუშავე გადამცემთა გამოსასვლელმა კასკადებმა უნდა უზრუნველყოს მაღალი გამოსასვლელი დენები. მაგისტრალური გადამწოდების მიერ მოთხოვნილი ***გამოსასვლელი დენების*** ტიპური სიდიდეები იცვლება ***20***-დან ***30 მილიამპერამდე*** ფარგლებში. იმავდროულად გადამცემების გადატვირთვის გამოსარიცხად საჭიროა მცირე იყოს მაგისტრალურ მიმღებებთა შესასვლელი დენები. მაგისტრალური მიმღებების ***შესასვლელი დენები 0,2-დან 0,8 მილიამპერამდე*** ფარგლებში უნდა იცვლებოდეს.

VIII თავი
მაგისტრალის მოწყობილობათა ფუნქციები

მიკროპროცესორული სისტემის მაგისტრალთან მიერთებული ძირითადი მოწყობილობებია პროცესორი, მეხსიერება და შეტანა/გამოტანის მოწყობილობები. მოკლედ განვიხილოთ თითოეული მათგანის ფუნქციები, აგრეთვე აგებისა და მაგისტრალთან მიერთების პრინციპები.

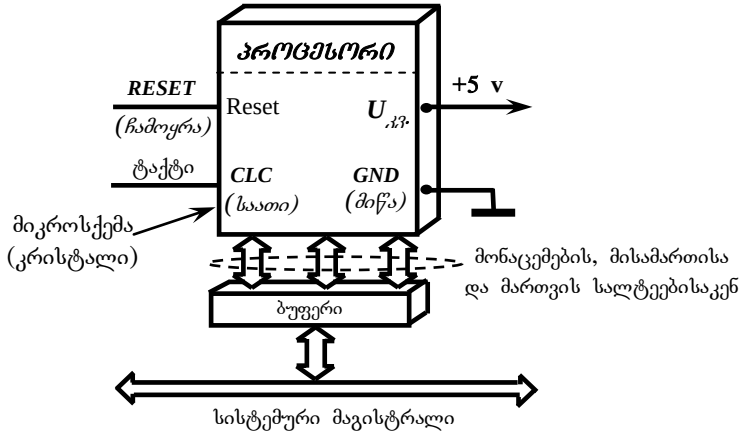
8.1. მიკროპროცესორი და მისი ფუნქციები

მიკროპროცესორს (ნახ.***8.1***), რომლის რაობასაც ზემოთ გავეცანით, ადრეულ წლებში რამდენიმე მიკროსქემისაგან წარმოქმნილი კომპლეტის სახე ჰქონდა, მაგრამ დღეისათვის ამ სახით მას პრაქტიკულად არ აგებენ. თანამედროვე მიკროპროცესორი წარმოადგენს მისამართის, მონაცემებისა და მართვის სალტეებთან დაკავშირერებული სამი, აგრეთვე სხვა დამხმარე ფუნქციების შესასრულებლად აუცილებელი სხვადასხვა დამატებითი გამომყვანის მქონე ერთ მიკროსქემას. გამომყვანთა რაოდენობის შესამცირებლად ზოგჯერ ახდენენ სიგნალებისა და სალტეების მულტიპლექსირებას.

პროცესორის უმნიშვნელოვანესი პარამეტრებია მისი მონაცემებისა და მისამართის სალტეთა თანრიგიანობა და მართვის სალტეში არსებული მმართველი სიგნალების რაოდენობა. მონაცემების სალტის თანრიგიანობა განსაზღვრავს სისტემის ***მუშაობის სიჩქარეს,*** მისამართის სალტის თანრიგიანობა – სისტემის ***დასაშვებ სირთულეს***, ხოლო მართვის ხაზების რაოდენობა – სისტემის სხვა მოწყობილობებთან პროცესორის მიერ ***ინფორმაციის გაცვლის რეჟიმების მრავალფეროვნებასა და ეფექტურობას***.

პროცესორი ტაქტირებადი მოწყობილობაა და ამიტომ სამი ძირითადი სალტისათვის განკუთვნილი გამომყვანების გარდა მას ყოველთვის აქვს გარე ***ტაქტური სიგნალის*** ანუ კვარცული (***CLK***) რეზონატორის მისაერთებლად განკუთვნილი ერთი ან ორი გამომყვანიც.

რაც უფრო დიდია პროცესორის ტაქტური სიჩქარე, მით უფრო სწრაფად მუშაობს იგი, ე. ი. მით უფრო სწრაფად ასრულებს ბრძანე-



ნახ. 8.1. *პროცესორის ჩართვის ბლოკური სქემა*

ბებს. აღსანიშნავია რომ პროცესორის სისწრაფეს არა მარტო ტაქტური სიხშირე, არამედ მისი სტრუქტურის თავისებურებებიც განსაზღვრავს. თანამედროვე პროცესორები ბრძანებათა დიდ უმრავლესობას ერთი ტაქტის განმავლობაში ასრულებს და მათ აქვს რამდენიმე ბრძანების პარალელურად შესრულებისათვის საჭირო საშუალებებიც. პროცესორის ტაქტური სიხშირე პირდაპირ და ხისტად არ განსაზღვრავს მაგისტრალში ინფორმაციის გაცვლის სიჩქარეს, რადგან ამ უკანასკნელს ზღუდავს მაგისტრალში სიგნალის დაყოვნებით გავრცელება და ამ დროს წარმოშობილი დამახინჯებები. მაშასადამე, ტაქტური სიხშირე განსაზღვრავს პროცესორის არა გარეგან, არამედ შინაგან სიჩქარეს. ზოგჯერ ტაქტურ სიხშირეს აქვს ქვედა და ზედა ზღვრები. ზედა საზღვარზე გადამეტებამ შეიძლება გადაახუროს პროცესორი და წარმოშვას (რეგულარულადაც, რაც ყველაზე უფრო არასასურველია) ამოვარდნები. ამიტომ ამ სიხშირის ცვლილებას ძალიან ფრთხილად უნდა მოვეკიდოთ.

პროცესორში არსებული კიდევ ერთი მნიშვნელოვანი სიგნალია საწყისი ***ჩამოყრის*** (ჩამოგდების) ***RESET სიგნალი.*** კვების ჩართვის, ავარიული სიტუაციის, ან პროცესორის «ჩაკიდების» დროს ამ სიგნა-

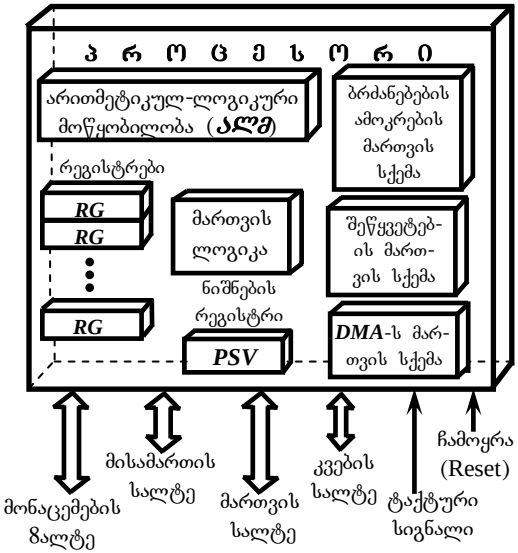
ის მიწოდება იწვევს პროცესორის ინიციალიზაციას (ვარგის მდგომარეობაში მოყვანას), აიძულებს მას შეუდგეს საწყისი ამუშავების პროგრამის შესრულებას. ავარიული სიტუაცია შეიძლება გამოიწვიოს კვების წრედებსა და «მიწაში» არსებულმა დაბრკოლებებმა, მეხსიერებაში მომხდარმა ამოვარდნებმა, გარე მაიონიზებელმა გამოსხივებამ და მრავალმა სხვა მიზეზმა. ამის შედეგად პროცესორმა შეიძლება დაკარგოს პროგრამის შესრულებაზე კონტროლო და რომელიღაც მისამართთან გაჩერდეს. სწორედ ამ მდგომარეობიდან გამოსაყვანად გამოიყენება საწყისი ჩამოყრის ***RESET*** სიგნალი. საწყისი ჩამოყრის ამ შესასვლელით შეიძლება პროცესორს ვაცნობოთ დადგენილ ზომაზე უფრო მეტად კვების ძაბვის შემცირების ფაქტი. ასეთ შემთხვევაში პროცესორი დაიწყებს მნიშვნელოვანი მონაცემების შემანარჩუნებელი პროგრამის შესრულებას. თავისი არსით ეს შესასვლელი რადიალური შეწყვეტის ნაირსა-ხეობად შეიძლება მივიჩნიოთ.

ზოგჯერ მიკროპროცესორის მიკროსქემას განსაკუთრებული სიტუაციების დასამუშავებლად (მაგალითად, გარე ტაიმერიდან შეწყვეტისათვის) შეიძლება ჰქონდეს რადიალური შეწყვეტების კიდევ ერთი-ორი შესასვლელიც.

თანამედროვე პროცესორის კვების სალტეს ჩვეულებრივ აქვს კვების ერთი (***+5*** ან ***+3*** ვოლტის ტოლი) ძაბვა და საერთო სადენი («მიწა»). ადრეულ პროცესორებს არც თუ ისე იშვიათად კვების რამდენიმე ძაბვა სჭირდებოდა. ზოგიერთ პროცესორებში გათვალისწინებულია დაბალი ენერგომოხმარების რეჟიმი. პროცესორების თანამედროვე, განსაკუთრებით მაღალი ტაქტური სიხშირის მქონე, მიკროსქემები ჩვეულებრივ საკმაოდ დიდ სიმძლავრეს მოიხმარს. ამიტომ კორპუსის ნორმალური სამუშაო ტემპერატურის შესანარჩუნებლად მათზე ხშირად რადიატორები, ვენტილატორები, ხოლო ზოგჯერ – სპეციალური მიკრომაცივრებიცაა დაყენებული.

მაგისტრალს პროცესორი უერთდება საჭიროებისამებრ დემულტიპლექსირებისა და მაგისტრალის სიგნალების ელექტრული ბუფერირების უზრუნველმყოფი ***ბუფერული მიკროსქემებით.*** ზოგჯერ სისტემური მაგისტრალში მიღებული ინფორმაციის გაცვლის პროტოკოლი არ ემთხვევა პროცესორის სალტეებში გაცვლისათვის გამოყენებულ პროტოკოლს; ამ დროს ბუფერული მიკროსქემები ერთმანეთთან ათანხმებს აღნიშნულ პროტოკოლებს. ზოგჯერ მიკროპროცე-

სორულ სისტემაში გამოიყენება რამდენიმე (სისტემური და ლოკალური) მაგისტრალი. ასეთ შემთხვევაში თითოეული მაგისტრალისათვის გამოიყენება საკუთარი ბუფერული კვანძი. ასეთი სტრუქტურა, მაგალითად, დამახასიათებელია ***პერსონალური კომპიუტერებისათვის.***



ნახ. 8.2. *მიკროპროცესორის შინაგანი სტრუქტურის გამარტივებული სქემა*

კვების ჩართვის შემდეგ პროცესორს მიეწოდება ***საწყისი ამუშავების პროგრამის*** პირველი ბრძანება და იგი შეუდგება მუდმივ (ენერგოდამოუკიდებელ) მეხსიერებაში წინასწარ ჩაწერილი ამ პროგრამის შესრულებას. მისი დამთავრების შემდეგ პროცესორი გადავა მუდმივ ან ოპერატიულ მეხსიერებაში არსებული ძირითადი პროგრამის შესასრულებლად, რისთვისაც იგი პირველი ბრძანებიდან დაწყებული რიგითობის დაცვით ამ პროგრამის ყველა ბრძანებას ასრულებს. ძირითადი პროგრამის შესრულების პროცესს იგი შეიძლება დროდადრო მოაცდინოს გარე შეწყვეტებმა ***DMA***–ს (იხ.***§6.2***) მოთხოვნებმა.

მეხსიერებიდან ბრძანებებს პროცესორი ამოკრებს მაგისტრალის მიხედვით წაკითხვების ციკლების დახმარებით. აუცილებლობის შემთხვევაში პროცესორი: ***1)*** ჩაწერის ციკლებით მონაცემებს ჩაწერს

მეხსიერებაში ან შეტანა/გამოტანის მოწყობილობებში; ***2)*** წაკითხვის ციკლებით ამოიკითხავს მონაცემებს მეხსიერებიდან ან შეტანა/გამოტანის მოწყობილობებიდან.

ამგვარად ნებისმიერი პროცესორის ძირითადი ფუნქციებია: **1)** შესასრულებელი ბრძანებების ამორჩევა (წაკითხვა); **2)** მეხსიერებიდან ან შეტანა/გამოტანის მოწყობილობიდან მონაცემების შეტანა (წაკითხვა); **3)** მეხსიერებაში ან შეტანა/გამოტანის მოწყობილობაში მონაცემების გატანა (ჩაწერა); **4)** მონაცემების (ოპერანდების) დამუშავება, მათ შორის მათზე არითმეტიკულ-ლოგიკური ოპერაციების შესრულება; **5)** მეხსიერების დამისამართება, ე. ი. მეხსიერების იმ მისამართის გაცემა, რომელთანაც უნდა გაიცვალოს ინფორმაცია; **6**) შეწყვეტებისა და ***DMA***-ს რეჟიმის დამუშავება.

8.2 ნახაზზე მოცემულია მიკროპროცესორის გამარტივებული სტრუქტურული სქემა. გავეცნოთ მასში მოყვანილი კვანძების ძირითად ფუნქციებს.

ბრძანებების ამოკრების მართვის სქემა განკუთვნილია მეხსიერებიდან ბრძანებების წასაკითხად და მათი გაშიფვრისათვის. ადრეულ მიკროპროცესორებში უკვე ამოღებული ბრძანების შესრულებისა და იმავდროულად მომდევნო ბრძანების ამოღების ოპერაციები ურთიერთშეუთავსებადები იყო და ამიტომ ისინი ერთდროულად ვერ სრულდებოდა. მოგვიანებით ***16***-თანრიგიან პროცესორებში გამოჩნდა წინა ბრძანების შესრულების პერიოდში რამდენიმე მომდევნო ბრძანების ამორჩევის შესაძლებლობის უზრუნველმყოფი ე.წ. ***ბრძანებათა კონვეიერი***. ამ დროს ორი პროცესი სრულდება პარალელურად, რაც ამაღლებს პროცესორის სწრაფმოქმედებას. ***კონვეიერი*** პროცესორის მომცრო შინაგანი მეხსიერებაა, რომელშიც პირველი შესაძლებლობისთანავე (გარე სალტის განთავისუფლებისთანავე) ჩაიწერება მოცემული ბრძანების მომდევნოდ შესასრულებელი რამდენიმე ბრძანება. მათ მიკროპროცესორი ისეთივე თანამიმდევრობით წაიკითხავს, რა თანამიმდევრობითაც ისინი არის ჩაწერილი კონვეიერში. ამის გამო კონვეიერი წარმოადგენს ***FIFO*** («**F**irst **I**n – **F**irst **O**ut», ანუ «პირველი შევიდა – პირველი გამოვიდა») ტიპის მეხსიერებას.

მოცემული ბრძანების შესრულების შემდეგ პროცესორი თუ გადადის არა მომდევნო, არამედ დაშორებით მდგარ ბრძანებაზე, მაშინ

კონვეიერი არ გამოგვადგება, მასში წინასწარ ჩაწერილი ბრძანებები გამოუყენებელი რჩება და ისინი უნდა ჩამოიყაროს. საბედნიეროდ ასეთი შემთხვევების რაოდენობა დიდი არ არის.

კონვეიერის იდეის განვითარების შედეგად დამუშავდა ***პროცესორის შინაგანი კეშმეხსიერება,*** რომელიც იმ პერიოდში, როდესაც პროცესორი გარკვეული ბრძანებების დამუშავებითაა დაკავებული, ივსება მომავალში შესასრულებელი ბრძანებებით. რაც უფრო დიდია კეშ-მეხსიერება, მით უფრო ნაკლებია იმის ალბათობა, რომ შეიძლება გადასვლის ბრძანების გამოჩენის დროს საჭირო გახდეს მისი შიგთავსის ჩამოყრა. ცხადია, რომ შინაგან მეხსიერებაში არსებულ ბრძანებებს პროცესორი გარე მეხსიერებაში არსებულ ბრძანებებზე უფრო სწრაფად ასრულებს. ***კეშ-ეხსიერებაში*** მოცემულ მომენტში დასამუშავებელი მონაცემებიც შეიძლება იქნეს შენახული, რაც აგრეთვე აჩქარებს მუშაობას. მაღალი სწრაფმოქმედების უზრუნველსაყოფად თანამედროვე პროცესორებში გამოიყენება: **1)** ბრძანებების ამოკრებისა და მათი გაშიფვრის ურთიერთშეთავსება; **2)** რამდენიმე ბრძანების ერთდროულად გაშიფვრა; **3)** ბრძანებათა რამდენიმე პარალელური კონვეიერი; **4)** გადასვლათა ბრძანებების წინასწარმეტყველება და ზოგიერთი სხვა მეთოდი.

არითმეტიკურ-ლოგიკური მოწყობილობა (ალმ, ALU) განკუთვნილია პროცესორის მიერ მიღებული ბრძანების შესაბამისად ინფორმაციის დასამუშავებლად. დამუშავების მიზანი შეიძლება იყოს სხვადასხვა ლოგიკური ან არითმეტიკული ოპერაციების შესრულება. ამ დროს რეალიზებადი ბრძანებით განისაზღვრება, თუ რომელ კოდებზე უნდა შესრულდეს ოპერაციები და სად უნდა მოთავსდეს მიღებული შედეგები. ბრძანებით თუ მოითხოვება მონაცემების დაუმუშავებლად გადაგზავნა, მაშინ მის შესრულებაში ***ალმ*** არ მონაწილეობს.

ალმ-ის სწრაფმოქმედება მნიშვნელოვანწილად განსაზღვრავს პროცესორის მწარმოებლურობას. ამასთანავე, მნიშვნელოვანია არა მარტო იმ ტაქტური სიგნალის სიხშირე, რომლითაც ხდება ***ალმ***–ის ტაქტირება, არამედ ამა თუ იმ ბრძანების შესასრულებლად საჭირო ტაქტების რაოდენობაც. მწარმოებლურობის ასამაღლებლად კონსტრუქტორი ცდილობს უზრუნველყოს ერთი ტაქტის განმავლობაში ბრძანების შესრულება და მაქსიმალურად გაზრდოს ტაქტური სიხში-

რე. მწარმოებლურობის ამაღლების ***ერთ-ერთი გზაა ალმ*** მიერ შესასრულებელი ბრძანებების რაოდენობის შემცირება, რაც რეალიზებულია ბრძანებების შემცირებულ ნაკრებზე მომუშავე (ე.წ. ***RISC*** ანუ "***R***educed ***I***nstruction ***S***et ***C***omputer") პროცესორებში. პროცესორის მწარმოებლურობის მიღწევის ***მეორე გზაა*** მასში პარალელურად მომუშავე რამდენიმე ***ალმ***-ს გამოყენება.

რაც შეეხება მცურავ წერტილიან რიცხვებზე შესასრულებელ, აგრეთვე სხვა სპეციალურ რთულ ოპერაციებს, ადრეულ პროცესორებში მათი რეალიზება ხდებოდა უფრო მარტივი ბრძანებათა მიმდევრობის, სპეციალური ვეპროგრამების გამოყენებით; შემდეგ დამუშავდა სპეციალური გამომთვლელები – მათემატიკური თანაპროცესორები, რომლებიც ასეთ ოპერაციების შესრულებისას ცვლიდა ძირითად პროცესორს. თანამედროვე მიკროპროცესორებში მათემატიკური თანაპროცესორები შემადგენელ ნაწილს წარმოადგენს.

პროცესორის RG რეგისტრები ფაქტობრივად სხვადასხვა კოდების (მონაცემების, მისამართების, ბრძანებების) დროებით შესანახად განკუთვნილი მეხსიერების ძალიან სწრაფი უჯრედებია. ვინაიდან მათზე ოპერაციები მაქსიმალურად სწრაფად სრულდება, ამიტომ სასურველია მიკროპროცესორში რაც შეიძლება ბევრი რეგისტრის განთავსება. პროცესორის სწრაფმოქმედებაზე ძალიან დიდ გავლენას ახდენს რეგისტრთა თანრიგიანობა. რეგისტრებისა და ***ალმ***-ის თანრიგიანობას ეწოდება ***პროცესორის შინაგანი თანრიგიანობა***, რომელიც შეიძლება არ დაემთხვეს მის ***გარეგან თანრიგიანობას.***

შინაგანი რეგისტრების დანიშნულების განსასაზღვრავად არსებობს ორი მიდგომა. კომპანია ***Intel***–ის მიერ მხარდაჭერილი ***პირველი მიდგომის*** დროს თითოეული რეგისტრისათვის განისაზღვრება მკაცრად განსაზღვრული ფუნქცია. ასეთი მიდგომა, ერთი მხრივ, ამარტივებს პროცესორის ორგანიზებას და ამცირებს ბრძანების შესრულების დროს, მაგრამ, მეორე მხრივ, ამცირებს მოქნილობას, ხოლო ზოგჯერ პროგრამის შესრულების პროცესსაც ანელებს. მაგალითად, ზოგიერთ არითმეტიკული ოპერაციებს ასრულებს და შეტანა/გამოტანის მოწყობილობებთან ინფორმაციის ცვლის ***აკუმულატორად*** წოდებული ერთადერთი რეგისტრი, რის გამოც ზოგიერთი პროცედურების ჩასატარებ-

ლად ზოგჯერ საჭირო ხდება რეგისტრებს შორის რამდენიმე დამატებითი გადაგზავნის პროცედურის შესრულება.

მეორე მიდგომის დროს ყველა (ან თითქმის ყველა) რეგისტრი თანაბარუფლებიანია. ასეთი მიდგომა გამოყენებულია, მაგალითად, ფირმა ***DEC***-ის მიერ გამოშვებულ ***16***-თანრიგიან ***T-11*** პროცესორში. ამ დროს მიიღწევა მაქსიმალური მოქნილობა, მაგრამ რამდენადმე რთულდება პროცესორის სტრუქტურა.

არსებობს საშუალედო მიდგომაც, კერძოდ, ფირმა ***Motorola***-ს მიერ შემოთავაზებულ ***MS68000*** პროცესორში რეგისტრების ერთი ნახევარი გამოიყენება მონაცემებისათვის და ისინი ურთიერთშენაცვლებადებია, ხოლო ასევე ურთიერთშენაცვლებადი მეორე ნახევარი მისამართებისთვისაა გამოყენებული.

■ ***ნიშნების PSW რეგისტრი (მდგომარეობის რეგისტრი)*** ზემოთ განხილული სხვა რეგისტრებისაგან განსხვავებით განკუთვნილია არა რაიმე მონაცემის ან მისამართის, არამედ ***პროცესორის მდგომარეობის მაჩვენებელი სიტყვის*** შესანახად, რომლის ინგლისური აბრევიატურაა ***PSW*** (***Processor Status Word***). ამ სიტყვის თითოეულ ბიტს ალამი ეწოდება და იგი შეიცავს ინფორმაციას წინა ბრძანების შესრულების შესახებ; მაგალითად, არსებობს ე.წ. «***ნულოვანი შედეგის ბიტი (ალამი)***», რომელიც მაშინ დამყარდება («გამოიფინება»), როდესაც წინა ბრძანების შესრულების შედეგად ნული იქნება მიღებული და გაიწმინდება («ჩამოიხსნება») მაშინ, როდესაც ბრძანების შედეგად მიღებული იქნება ნულისაგან განსხვავაბული შედეგი. ამ ბიტებს (ალმებს) იყენებს პირობითი გადასვლის ბრძანებები, მაგალითად «***ნულოვანი შედეგის შემთხვევაში გადასვლის ბრძანება***». აღნიშნული რეგისტრი შეიძლება შეიცავდეს ე. წ «***მართვის ალმებსაც***», რომელთა დანიშნულებაა განსაზღვროს. გარკვეული ბრძანებების შესრულების რეჟიმები.

■ ***შეწყვეტების მართვის სქემის*** დანიშნულებაა: ***1)*** დაამუშაოს პროცესორთან მოსული შეწყვეტის მოთხოვნა; ***2)*** განსაზღვროს შეწყვეტის დამუშავების პროგრამის სათავის მისამართი (შეწყვეტის ვექტორის მისამართი); ***3)*** მიმდინარე ბრძანების დამთავრების შემდეგ მეხსიერებაში (სტეკში) შეინახოს პროცესორის რეგისტრთა მიმდინა-

რე შიგთავსები; ***4)*** უზრუნველყოს შეწყვეტის დამუშავების პროგრამაზე გადასვლა;

შეწყვეტის დამუშავების პროგრამის დამთავრების შემდეგ მეხსიერებიდან (სტეკიდან) აღდგება შინაგანი რეგისტრების მნიშვნელობები და პროცესორი უბრუნდება შეწყვეტილ ძირითად პროგრამას. სტეკის შესახებ დაწვრილებით ვისაუბრებთ შემდეგ განყოფილებაში.

■ ***მეხსიერებასთან ინფორმაციის უშუალოდ გაცვლის მმართველი სქემა*** განკუთვნილია გარე სალტეებიდან პროცესორის დროებით გამოსართველად და მისი მუშაობის მანამადე შესაჩერებლად, სანამ ***DMA***-ს მომთხოვნი მოწყობილობა მეხსიერებასთან უშუალოდ ინფორმაციის გაცვლის უფლებით სარგებლობს..

■ ***მართვის ლოგიკის*** მიერ სრულდება: ***1)*** პროცესორის კვანძების ურთიერთზემოქმედებები; ***2***) მონაცემების გადაგზავნები; ***3)*** გარე სიგნალებთან პროცესორის სინქრონიზება; ***3)*** ინფორმაციის შეტანისა და გამოტანის პროცედურები.

პროცესორის მუშაობის პერიოდში ბრძანებების ამოკრების სქემა მეხსიერებიდან მიმდევრობით ამოირჩევს შესასრულებელ ბრძანებებს; მონაცემების დამუშავების საჭიროების წარმოშობ‌ისას ერთვება ***ასმ.*** ამ უკანასკნელის შესასვლელებს დასამუშავებელი მონაცემები შეიძლება მიეწოდოს მეხსიერებიდან ან შინაგანი რეგისტრებიდან. შინაგან რეგისტრებში ინახება მეხსიერებაში განთავსებული დასამუშავებელი მონაცემების მისამართებიც. ***ასმ***-ში მონაცემების დამუშავების შედეგები ცვლის ნიშნების რეგისტრის მდგომარეობას და ჩაიწერება შინაგან მეხსიერებაში ან მეხსიერებაში (მონაცემების წყაროცა და მიმღებიც ბრძანების კოდშია მითითებული). აუცილებლობის შემთხვევაში ინფორმაცია შეიძლება გადაიწეროს მეხსიერებიდან (ან შეტანა/გამოტანის მოწყობილობიდან) შინაგან რეგისტრში ან პირიქით, შინაგანი რეგისტრიდან - მეხსიერებაში (ან შეტანა/გამოტანის მოწყობილობაში).

ნებისმიერი მიკროპროცესორის შინაგანი რეგისტრი ასრულებს შემდეგ ორ სამომსახურო ფუნქციას: ***1)*** განსაზღვრავს მეხსიერების იმ უჯრედის მისამართს, რომელშიც მოთავსებულია მოცემულ მომენტში შესასრულებელი ბრძანება (ესაა ***ბრძანებათა მთვლელის*** ანუ ***ბრძანებ-***

ათა მაჩვენებლის ფუნქცია); ***2)*** განსაზღვრავს სტეკის მიმდინარე მისამართს (ესაა ***სტეკის მაჩვენებლის*** ფუნქცია).

სხვადასხვა მიკროპროცესორში თითოეული ამ ფუნქციის შესასრულებლად შეიძლება გამოიყოს ერთი ან ორი შინაგანი რეგისტრი. ზემოთ აღნიშნული ორი (ბრძანებებისა და სტეკის) მაჩვენებლები ერთმანეთისაგან განსხვავდება არა მარტო საკუთარი სპეციფიკური, სამოსახურო, სისტემური დანიშნულებით, არამედ შიგთავსის შეცვლის განსაკუთრებული ხერხებითაც. პროგრამებს მათი შიგთავსების შეცვლა მხოლოდ უკიდურესი საჭიროების დროს შეუძლია, რადგან ამ დროს დაშვებულმა შეცდომამ შეიძლება დაარღვიოს კომპიუტერის მუშაობა, გამოიწვიოს მისი «დაკიდება» («გაჭედვა») და გააფუჭოს მეხსიერების შიგთავსი.

ბრძანებების მაჩვენებლის (მთვლელის) შიგთავსი შემდეგნაირად იცვლება. სისტემის მუშაობის დასაწყისში (კვების ჩართვისას) მასში ერთხელ და სამუდამოდ შეიტანება დადგენილი მნიშვნელობა. ესაა საწყისი ამუშავების პროგრამის პირველი მისამართი. მეხსიერებიდან თითოეული მორიგი ბრძანების ამოღების შემდეგ ბრძანების ფორმატსა და მიკროპროცესორის ტიპზე დამოკიდებულებით ერთი ან ორი ერთეულით ავტომატურად იზრდება (ინკრემენტირდება) ბრძანებების მაჩვენებლის მნიშვნელობა; ე. ი. მომდევნო ბრძანება ამორჩეული იქნება მეხსიერების რიგით მომდევნო მისამართიდან. მეხსიერების მისამართების მიმდევრობით გადარჩევის დამრღვევი ***გადასვლის ბრძანების*** შესრულებისას ბრძანებების მაჩვენებელში იძულებით ჩაიწერება ახალი მნიშნელობა – მეხსიერების ახალი მისამართი, რომლიდანაც დაწყებული ბრძანებათა მისამართები მიმდევრობის დაცვით ამოირჩევა. ქვეპროგრამის გამოძახებისა და ქვეპროგრამიდან დაბრუნების დროს ან შეწყვეტის დამუშავების დაწყებისა და მისი დამთავრების დროს ბრძანებათა მაჩვენებლის შიგთავსები ანალოგურად შეიცვლება.

ზოგადად სტეკზე და, კერძოდ, სტეკის მაჩვენებელზე ქვემოთ ვილაპარაკებთ..

8.2. მეხსიერების ფუნქციები

მიკროპროცესორული სისტემის მეხსიერება დროებით ან მუდმივად შეინახავს მონაცემებისა და ბრძანებების ორობით კოდებს. მეხსიერების მოცულობით განისაზღვრება არა მარტო ის, თუ რამდენად რთული ალგორითმების რეალიზებას შეძლებს სისტემა, არამედ გარკვეულწილად მთლიანად სისტემის სწრაფმოქმედებაც. ***მეხსიერების მოდულები*** შედგება ოპერატიული ან მუდმივი მეხსიერების მიკროსქემებისაგან. ბოლო წლებში მიკროპროცესორული სისტემებში ხშირად გამოიყენება ე. წ. ***ფლეშმეხსიერება*** (***flesh memory***), რომელიც წარმოადგენს შიგთავსის მრავალჯერადად ჩაწერის უნარის მქონე ენერგოდამოუკიდებელ მეხსიერებას.

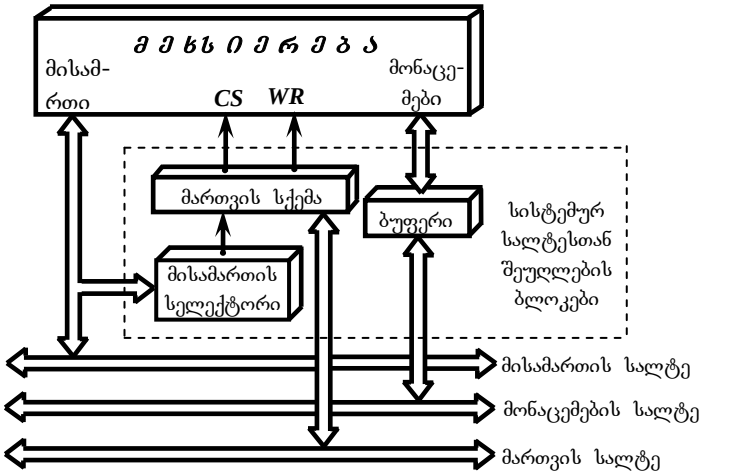
ორობითი კოდების სახით წარმოდგენილი ინფორმაცია უშუალოდ შეინახება მეხსიერების უჯრედებში, რომელთა რაოდენობა ***2***-ის ჯერადია. მისამართების ***N*** თანრგიანი სალტის გამოყენებისას მეხსიერების უჯრედების მაქსიმალური რაოდენობა 2^N -ის ტოლია. ***მეხსიერების მოცულობას*** ყველაზე ხშირად მისი უჯრედების თანრიგიანობაზე დამოუკიდებლად ბაიტებით ზომავენ. ზოგადად ***ტერმინ ბაიტით*** აღინიშნება ინფორმაციის ერთი სიმბოლოს კოდირებისათვის გამოყენებული ორობითი რიცხვის ბიტების რაოდენობა. იყო დრო, როდესაც ამისთვის გამოიყენებოდა ***5***-ბიტური ორობითი რიცხვები და მაშინ ბაიტად ითვლებოდა ***5*** ბიტი; დღეისათვის აღნიშნული მიზნისათვის ***8***-ბიტურ ორობითი რიცხვები გამოიყენება და ამიტომ ბაიტი ეწოდება ***8***-ს. გამოიყენება მეხსიერების მოცულობის უფრო მსხვილი ერთეულებიც; მაგალითად, ***კილობაიტი*** (**კბ**), რომელიც უდრის $2^{10} = 1024$ ბაიტს; ***მეგაბაიტი*** (**მბ**), რომელიც უდრის $2^{20} = 1048576$ ბაიტს (დაახლოებით 10^6 მილიონ ბაიტს); ***გიგაბაიტი*** (**გბ**), რომელიც უდრის $2^{30} = 1073741824$ ბაიტს (დაახლოებით უდრის მილიარდ ბაიტს); ***ტერაბაიტი*** (**ტბ**), რომელიც უდრის $2^{40} = 1099511627776$ ბაიტს (დახლოებით ტრილიონ ბაიტს); ***პეტაბაიტი*** (**პბ**), რომელიც უდრის $2^{50} = 1024$ ტერაბაიტს (დაახლოებით კვადრილიონ ბაიტს) და ა.შ.

მეხსიერებაში თუ ***65536*** უჯრედია, რომელთაგანაც თითოეული ***16*** თანრიგიანია, მაშინ ამბობენ, რომ მეხსიერების მოცულობა ***128***

კილობაიტის ტოლია. მიკროპროცესორული სისტემის მეხსიერების უჯრედების ერთობლიობას სისტემის ***მეხსიერების სივრცე*** ეწოდება.

სისტემურ მაგისტრალთან მეხსიერების მოდულის მისაერთებლად გამოიყენება ***შეუღლების ბლოკები***, რომელთა შემადგენლობაში შედის მისამართის დეშიფრატორი (სელექტორი), მაგისტრალის მმართველი სიგნალების დამამუშავებელი სქემა და მონაცემების ბუფერები (ნახ. ***8.3***)

მიკროპროცესორული სისტემის მუშაობის პროცესში პერმანენტულად მყარდება კავშირი სისტემულ სალტესა და მეხსიერებას შორის. ***ოპერატიული მეხსიერების*** შემთხვევაში ეს კავშირები მყარდება წაკითხვისა და ჩაწერის, ხოლო ***მუდმივი მეხსიერების*** შემთხვევაში - მხოლოდ წაკითხვის ციკლებში. სისტემის სტრუქტურა შეიცავს მეხსიერების რამდენიმე მოდულს, რომელთაგანაც თითოეულისათვის მეხსიერებაში გამოყოფილია ფიქსირებული რაოდენობის უჯრდებისაგან წარმოქმნილი გარკვეული ***მიდამო***. ამ მიდამოს განსაზღვრავს მოდულში არსებული ***მისამართის სელექტორი*** (იხ.ნახ.***8.3***). საჭირო მომენტებში მეხსიერების მუშაობის ნებადართვის ***CS*** და მეხსიერებაში ჩაწერის ნებადართვის ***WR*** სიგნალებს გამოიმუშავებს მართვის სქემა. მონაცემების ბუფერები მონაცემებს გადასცემს მეხსიერებიდან მაგისტრალში ან პირიქით – მაგისტრალიდან მეხსიერებაში.

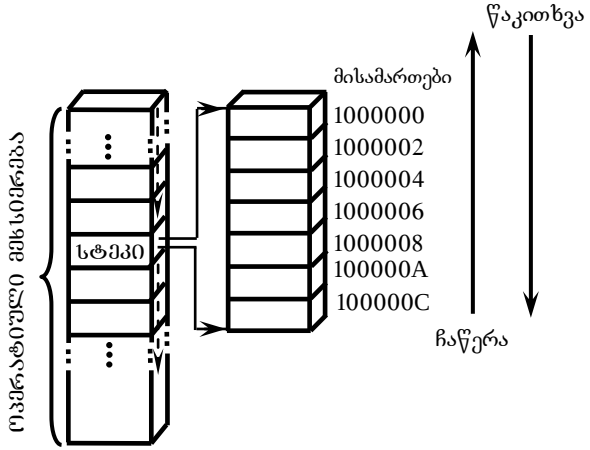


ნახ. 8.3. *მეხსიერების მოდულის გამარტივებული სტრუქტურა*

მიკროპროცესორული სისტემის მეხსიერების სივრცეში ჩვეულებრივ გამოყოფილია სპეციალური ფუნქციების შემსრულებელი რამდენიმე განსაკუთრებული მიდამო.

საწყისი ამუშავების პროგრამის მეხსიერების მიდამო ყოველთვის მუდმივ დამხსომებელ მოწყობილობაში ან ფლეშმეხსიერებაშია ორგანიზებული. კვების წყაროს ჩართვის ან ***RESET*** სიგნალის დახმარებით ჩამოყრის შემდეგ პროცესორი სწორედ ამ მიდამოდან იწყებს მუშაობას.

სტეკის მეხსიერების მიდამო ანუ უბრალოდ ***სტეკი*** ოპერატიული მეხსიერების ნაწილია, რომელიც განკუთვნილია ***LIFO*** (***"Last In – First Out"*** ე. ი. "შევიდა ბოლო, გამოვიდა პირველი") რეჟიმში მონაცემების დროებით შესანახად. სხვა ოპერატიული მეხსიერებისაგან განსხვავებით ტეკის დამისამართების ხერხი უცვლელია. სტეკში ნებისმიერი რიცხვის (კოდის) ჩაწერისას აღნიშნული რიცხვი ჩაიწერება მისამართზე, რომელიც მიიღება სტეკის მაჩვენებლის რეგისტრის შიგთავსის ერთი (ან ორი) ერთეულით წინასწარ შემცირების (***დეკრემენტირების***) შედეგად. ამის გამო აღმოჩნდება, რომ უკანასკნელად ჩაწერილი რიცხვი თავდაპირველად, ხოლო პირველად ჩაწერილი რიცხვი – ბოლოში იქნება წაკითხული. ასეთ მეხსიერებას ეწოდება ანუ ***სავაზნის ტიპის მეხსიერება*** (ავტომატის სავაზნიდან თავდაპირველად ამოიღება



ნახ. 8.4 *სტეკის მუშაობის პრინციპის ილუსტრირება*

მასში უკანასკნელად ჩადებული ვაზნა). სტეკის მოქმედების პრინციპი ***8.4*** ნახაზზეა მოყვანილი (მეხსიერების უჯრე-დების მისამართები პირობითადაა შერჩეული.

დავუშვათ, რომ სტეკის მიმდინარე მდგომარეობაა ***1000008*** და მასში უნდა ჩავწეროთ ორი რიცხვი (სიტყვა). პირველი სიტყვის ჩაიწერება ***1000006*** (ჩაწერის წინ სტეკის მაჩვენებელი ორით მცირება), ხოლო მეორე – ***1000004*** მისამართიან უჯრედში. ამის შემდეგ თუ ამ ორ სიტყვას წავიკითხავთ, პირველად წაიკითხება ***1000004*** მისამართიან უჯრედში ჩაწერილი სიტყვა, რომლის შემდეგ სტეკის მაჩვენებელი გახდება ***1000006***. შემდეგ წაიკითხება ***1000006*** უჯრედში არსებული სიტყვა და სტეკის მაჩვენებელი ***1000008***-ს ტოლი გახდება. ყველაფერი დაბრუნდა საწყის მდგომარეობაში. პირველად წაკითხული იქნა მეორედ ჩაწერილი, ხოლო შემდეგ - პირველად ჩაწერილი სიტყვა.

ასეთი დამისამართება აუცილებლია ***მრავალჯერადად ჩალაგებული პროგრამების*** შემთხვევაში. დავუშათ, რომ სრულდება ძირითადი პროგრამა, რომლიდანაც საჭიროა გამოძახებული იქნეს ***1***-ლი ქვეპროგრამა. ამ ქვეპროგრამის რეალიზების დაწყებამდე შესაწყვეტი ძირითადი პროგრამის მონაცემები და მისი მომსახურე შიგა რეგისტრების შიგთავსები შეინახება (გადაიწერება) სტეკში და იქ იქნება შენახული მანამ, სანამ ბოლომდე არ შესრულდება გამოძახებული ქვეპროგრამა. ამის შემდეგ იგი დაბრუნდება (წაიკითხება, ამოიღება) სტეკიდან. საჭირო თუ გახდა ***1***-ლი ქვეპროგრამიდან მე-***2*** ქვეპროგრამის გამოძახება, მაშინ იგივე ოპერაცია ჩატარდება ***1***-ლი ქვეპროგრამის მონაცემებისა და შინაგანი რეგისტრების შიგთავსებისთვისაც. ნათელია, რომ მე-***2*** ქვეპროგრამის შიგნით სტეკის ნაპირთან (საიდანაც იწყება წაკითხვა) იქნება ***1***-ლი ქვეპროგრამის მონაცემები, ხოლო უფრო ღრმად იქნება ძირითადი პროგრამის მონაცემები. ამ დროს სტეკიდან წაკითხვისას ავტომატურად იქნება დაცული ინფორმაციის წაკითხვის საჭირო თანამიმდევრობა. იგივე იქნება მაშინაც, როდესაც გაცილებით მაღალი იქნება ასეთი ჩალეგებების დონე. აქედან ნათლად ჩანს, რომ სასურველია მონაცემები, რაც შეიძლება ღრმად შევინახოთ, რათა ისინი სწრაფად არ ამოვიდეს ზედაპირზე.

სტეკთან ინფორმაციის გაცვლისათვის ნებისმიერი პროცესორის ბრძანებათა სისტემაში გათვალისწინებულია სტეკში ჩაწერის ***PUSH***

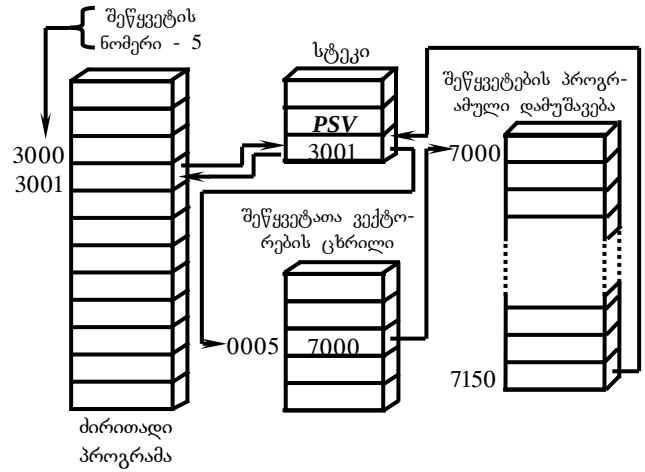
და სტეკიდან წაკითხვის ***POP*** ბრძანებები. სტეკში შეიძლება გადავმალოთ არა მარტო პროცესორების ყველა შინაგანი რეგისტრის, არამედ ***ნიშნების რეგისტრის*** შიგთავსიც (პროცესორის მდგომარეობის ***PSW*** სიტყვაც) შევინახოთ. ეს საშუალებას მოგვცემს, მაგალითად, ქვეპროგრამიდან დაბრუნებისას გავაკონტროლოთ ამ ქვეპროგრამის უშუალოდ გამოძახების დროს შესასრულებელი უკანასკნელი ბრძანების შესრულების შედეგი. პროგრამებსა და ქვეპროგრამებს შორის მონაცემების მოხერხებულად ურთიერთგაცვლისათვის სტეკში შეიძლება მონაცემებიც შევინახოთ. ზოგადად, მეხსიერებიდან სტეკს რაც უფრო დიდი მიდამო დაეთმობა, მით უფრო გაუადვილდება მუშაობა დამპროგრამებელს და მით უფრო რთული პროგრამების შესრულებას შეძლებს მიკროპროცესორული სისტემა.

მეხსიერების შემდეგი სპეციალური მიდამოა ***შეწყვეტების ვექტორთა ცხრილი.***

შეწყვეტის ცნება ზოგადად საკმაოდ მრავალმნიშვნელოვანია. ***შეწყვეტად ითვლება*** არა მარტო გარე მოწყობილობის მოთხოვნის მომსახურება, არამედ პროცესორის მუშაობის თანამიმდევრობის ნებისმიერი დარღვევაც. მაგალითად, შეწყვეტა შეიძლება ისეთი არითმეტიკული ოპერაციის არაკორექტული შესრულებისათვის იყოს გათვალისწინებული, როგორიცაა ნულზე გაყოფა. არსებობს აგრეთვე ***პროგრამული შეწყვეტაც,*** რომლის დროსაც ძირითად პროგრამაში წინასწარაა გათვალისწინებული გარკვეულ ქვეპროგრამაზე გადასვლის ბრძანება და მისი შესრულების შემდეგ ხდება ძირითად პროგრამაზე დაბრუნება. პროგრამულ და ნამდვილ შეწყვეტებს ის აქვს საერთო, რომ ორივე შემთხვევაში ერთნაირად ხდება ძირითადი პროგრამიდან ქვეპროგრამაზე და პირიქით გადასვლის პროცესები.

ნებისმიერი შეწყვეტა მუშავდება შეწყვეტათა ვექტორების (მაჩვენებლების) ცხრილის მეშვეობით. უმარტივეს შემთხვევაში ამ ცხრილში არსებობს შეწყვეტების დამუშავების პროგრამების დასაწყისთა მისამართები, რომლებსაც ***ვექტორები ეწოდება***. ცხრილი შეიძლება საკმაოდ გრძელი იყოს (რამდენიმე ასეულ ელემენტს შეიცავდეს). ჩვეულებრივად შეწყვეტების ვექტორთა მიდამო მეხსიერების სივრცის დასაწყისში (მეხსიერების დაბალი მისამართებიან უჯრედებში) განთავსდება. თითოეული ვექტორის (ანუ თითოეული ვექტორის საწყისი ელემენტის) მისამართი წარმოადგენს შეწყვეტის ნომერს.

აპარატურული შეწყვეტების დროს შეწყვეტის ნომერი განისაზღვრება შეწყვეტის მომთხოვნი მოწყობილობით (***ვექტორული შეწყვეტის დროს***) ან შეწყვეტის მომთხოვნი ხაზის ნომრით (***რადიალური შეწყვეტის დროს***). აპარატურული შეწყვეტის მიმღები მოწყობილობა დაასრულებს მიმდინარე ბრძანების შესრულებას და მიმართავს მეხსიერებაში არსებული შეწყვეტების ვექტორთა ცხრილის იმ სტრიქონს, რომელიც განისაზღვრება მოთხოვნილი შეწყვეტის ნომრით. პროცესორი წაიკითხავს ამ სტრიქონის შიგთავსს (შეწყვეტის ვექტორის კოდს) და გადავა მეხსიერების ამ ვექტორით მოცემულ მისამართზე. ამ მისამართიდან დაწყებული მეხსიერებაში განთავსებული უნდა იყოს მოცემული ნომრიანი შეწყვეტის დამუშავების პროგრამა. შეწყვეტის დამუშავების პროგრამის ბოლოში აუცილებლად უნდა იყოს განთავსებული შეწყვეტიდან გამოსვლის ბრძანება, რომლის შესრულების შემდეგ პროცესორი დაუბრუნდება შეწყვეტილ ძირითად პროგრამას და გააგრძელებს მის შესრულებას. შეწყვეტების დამუშავების პროგრამის შესრულების პერიოდში პროცესორის პარამეტრები სტეკში უნდა იყოს შენახული.



ნახ.8.5. *შეწყვეტის დამუშავების გამარტივებული ალგორითმი*

დავუშვათ, რომ პროცესორი ასრულებდა ძირითადი პროგრამის იმ ბრძანებას, რომლის ნომერი მეხსიერებაში ვთქვათ იყო ***3000*** (ნახ. ***8.5***). ამ მომენტში მან მიიღო მოთხოვნა შეწყვეტიდან, რომლის ნო-

მერი იყო ***5***. პროცესორი დაამთავრებს იმ ბრძანების შესრულებას, რომლის მისამართი მეხსიერებაში იყო ***3000***. შემდეგ სტეკში შეინახავს ბრძანებების მთვლელის მიმდინარე ნომერს (***3001***-ს) და ***PSW***–ს მიმდინარე მნიშვნელობას. ამის შემდეგ მეხსიერების ნომერ ***5***-დან წაიკითხავს შეწყვეტის ვექტორის კოდს. დავუშვათ, ეს კოდი იყოს ***7000***-ის ტოლი. პროცესორი მეხსიერებაში ***7000*** მისამართზე გადავა და ამ მისამართიდან დაიწყებს შეწყვეტის პროგრამის დამუშავებას. შეიძლება, რომ ეს პროგრამა მთავრდებოდეს მისამართ ***7150***-ზე. ამ მისამართზე მისვლის შემდეგ პროცესორი უბრუნდება შეწყვეტილი პროგრამის შესრულებას. ამისათვის მას სტეკიდან ამოაქვს მისამართის მნიშვნელობა (ჩვენს შემთხვევაში **3001**), რომელზედაც შეწყდა ძველი პროგრამა და იმ მომენტისათვის არსებული ***PSW.*** პროცესორი წაიკითხავს ბრძანებას მისამართ ***3001***-დან

ავარიულ სიტუაციაში წარმოშობილი შეწყვეტა ზუსტად ასევე დამუშავდება, ოღონდ ამ შემთხვევაში შეწყვეტის ვექტორის მისამართი (ვექტორების ცხრილში სტრიქონის ნომერი) ხისტად იქნება მიბმული ავარიული სიტუაციის მოცემულ ტიპთან.

შეწყვეტების ორგანიზების ასეთი, პირველი შეხედვით საკმაოდ რთული ორგანიზაცია დამპროგრამებელს საშუალებას აძლევს ადვილად შეცვალოს შეწყვეტების დამუშავების პროგრამები, ისინი მეხსიერების ნებისმიერ სფეროში განათავსოს და ნებისმიერი ზომისა და სირთულის გახადოს.

შეწყვეტის დამუშავების პროგრამის შესრულების პერიოდში შეიძლება მოვიდეს შეწყვეტაზე ახალი მოთხოვნა. ამ შემთხვევაში იგი დამუშავდება ზუსტად ისევე, როგორც ეს ზემოთ აღვწერეთ, ოღონდ ძირითად პროგრამად ჩაითვლება წინა შეწყვეტის დამუშავების პროგრამა. ამას ეწოდება ***შეწყვეტების მრავალჯერადი ჩალაგება.*** სტეკის მექანიზმი ასეთი მრავალჯერადი ჩალაგების უპრობლემოდ მომსახურების საშუალებას იძლევა, რადგან სტეკიდან პირველად ამოიღება უკანასკნელად შენახული კოდი, ე.ი. მოცემული შეწყვეტის დამუშავებიდან გადასვლა ხდება წინა შეწყვეტაზე.

შევნიშნავთ, რომ უფრო რთული შემთხვევების დროს შეწყვეტების ვექტორთა ცხრილში შეიძლება განთავსებული იყოს არა შეწყვეტების დამუშავების პროგრამების დასაწყისის მისამართები, არამედ შეწყვეტების ე. წ. ***დესკრიპტორები*** (აღმწერები). მიუხედავად ასეთი

სახეცვლილებისა, ამ დესკრიპტორის დამუშავების შედეგად საბოლოოდ მაინც შეწყვეტის დამუშავების პროგრამის დასაწყისის მისამართი მიიღება.

მიკროპროცესორული სისტემის მეხსიერების კიდევ ერთი სპეციალური მიდამოში ორგანიზდება ***სისტემურ სალტესთან მიერთებული მოწყობილობების მეხსიერება.*** მეხსიერების ასეთი უბანი არც თუ ისე ხშირად გვხვდება, მაგრამ ზოგჯერ მისი არსებობა მეტად მოსახერხებელია. ე. ი. პროცესორი იღებს შეტანა/გამოტანის ან სისტემურ სალტესთან მიერთებული რომელიმე სხვა მოწყობილობის მეხსიერებასთან იმგვარი ურთიერთობის დამყარების საშუალებას, როგორი ურთიერთობაც აქვს მას დამყარებული საკუთარ სისტემურ მეხსიერებასთან. ჩვეულებრივ ამისათვის მეხსიერების სივრცეში გამოყოფილი სარკმელი არცთუ ისე დიდია.

მეხსიერების სივრცის ყველა დანარჩენ ნაწილის დანიშნულება უნივერსალურია. მათში განთავსდება როგორც მონაცემები, ასევე პროგრამები (რა თქმა უნდა, ერთსალტური, ანუ ფონ ნეიმანისეული არქიტექტურის შემთხვევაში). ზოგჯერ მეხსიერების ეს სივრცე გამოიყენება ყოველგვარი საზღვრებისაგან თავისუფალი ერთიანი სივრცის სახით. სხვა შემთხვევებში კი მეხსიერების სივრცე შეიძლება დაყოფილი იყოს ***სეგმენტებად*** და ჰქონდეს სეგმენტის დასაწყისის მისამართისა და სეგმენტის დადგენილი ზომის პროგრამულად შეცვლის შესაძლებლობა.

ზემოთ აღნიშნულ ***ორივე მიდგომას*** აქვს როგორც დადებითი, ისე უარყოფითი მხარეები. მაგალითად ***სეგმენტების გამოყენება*** პროგრამების ან მონაცემების მიდამოს დაცვის საშუალებას იძლევა, მაგრამ სეგმენტების საზღვრებმა შეიძლება გაართულოს დიდი პროგრამებისა და მონაცემთა მასივების განთავსება.

დასასრულს შევჩერდებით მეხსიერების მისამართებისა და შეტანა/გამოტანის მოწყობილობების მისამართების გამოჯვნის პრობლემაზე. არსებობს ამ პრობლემის გადაწყვეტისადმი შემდეგი ორი ძირითადი მიდგომა: **1)** საერთო სამისამართო სივრცეში შეტანა/გამოტანის მოწყობილობების მისამართებისათვის სპეციალური მიდამოს გამოყოფა; ***2)*** მეხსიერების სამისამართო სივრცისაგან შეტანა/გამოტანის მოწყობილობების სამისამართო სივრცის სრული გამოჯვნა.

პირველი მიდგომა იმითაა კარგი, რომ შეტანა/გამოტანის მოწყობილობებთან მიმართვისას პროცესორმა შეიძლება გამოიყენოს მეხსიერებასთან საურთიერთებლო ბრძანებები, მაგრამ ამ დროს შეტანა/გამოტანის მოწყობილობების სამისამართო სივრცის ტოლი სიდიდით მცირდება მეხსიერების სამისამართო სივრცე. მაგალითად, ***16***-თანრიგიანი სამისამართო სალტის დროს შეიძლება მხოლოდ ***64*** კილობაიტის რაოდენობის მისამართების არსებობა. ამათგან ***54*** კილობაიტის რაოდენობის მისამართი გამოიყოფა მეხსიერების სამისამართო სივრცისათვის, ხოლო ***8*** კილობაიტი სივრცე – შეტანა/გამოტანის მოწყობილობების სამისამართო სივრცისათვის.

მეორე მიდგომის ღირსება ის არის, რომ მეხსიერება იკავებს მიკროპროცესორული სისტემის მთელ სამისამართო სივრცეს. შეტანა/გამოტანის მოწყობილობებთან ურთიერთობისათვის გამოიყენება სპეციალური ბრძანებები და მაგისტრალზე ინფორმაციის გაცვლის სპეციალური სტრობები. სწორედ ასე გაკეთებული, მაგალითად, ***პერსონალურ კომპიუტერებში;*** მაგრამ მოცემულ შემთხვევაში შეტანა/გამოტანის მოწყობილობებთან ურთიეთობის შესაძლებლობები საერთო მეხსიერების დროს ურთიერთობის შესაძლებლობასთან შედარებით მნიშვნელოვნად შეზღუდულია.

8.3. ვირტუალური მეხსიერების კონცეფცია და მეხსიერების გვერდობრივი ორგანიზაციის პრინციპი

მიკროპროცესორული სისტემის მეხსიერება, რომელსაც ***ჩარლზ ბებიჯმა*** საწყობი (***Store***) უწოდა, პროგრამებისა და მონაცემების სახით წარმოდგენილი ინფორმაციის შესანახად («დასაწყობებისათვის») შექმნილი საშუალებაა. ზოგადად არსებობს მისი შემდეგი ორი სახე:

▲ ოპერატიული მიზნებისათვის განკუთვნილი მცირე მოცულობის ინფორმაციის შესანახად გამოიყენება ***ძირითადი მეხსიერება***, რომელიც რეალიზებულია ოპერატიული დამხსომებელი მოწყობილობებისა და ***მუდმივი დამხსომებელი მოწყობილობების*** სახით;

▲ დიდი მოცულობის ხანგრძლივად შესანახი დისკური, ლენტური და სხვა სახის დამგროვებლები.

ადრეულ მიკროპროცესორულ სისტემებში მეხსიერების ერთი ნაწილი განთავსდებოდა მიკროპროცესორულ ბლოკში, მეორე ნაწილი კი აღნიშნული ბლოკის გარეთ; ამიტომ პირველ მათგანს უწოდებდნენ ***შიგა***, ხოლო მეორე მათგანს – ***გარე მეხსიერებას.*** ამ დროს შიგა მეხსიერება წარმოადგენდა ელექტრონულ და მაგნიტურ (ფერიტულ გულარებიან) მოწყობილობებს, გარე მეხსიერება კი მაგნიტური დოლების გამოყენებით დამზარებულ მოძრავ მზიდებიან განცალკევებულ მოწყობილობებს. დღეისათვის მეხსიერების მოწყობილობების უმრავლესობა (პერსონალური კომპიუტერების შემთხვევაში კი მთლიანად) ***სისტემურ ბლოკშია*** მოთავსებული. მიუხედავად ამისა, შენარჩუნებული იქნა შიგა და გარე მოწყობილობებად მეხსიერების დაყოფის ტრადიცია, ამასთანავე:

▲ ***შიგა (ოპერატიული და მუდმივი) მეხსიერება*** ეწოდება მუშაობის დროს მიკროპროცესორის მიერ უშუალოდ მოხამრებადი ინფორმაციის შემნახველ მეხსიერებას, რომელიც მიკროსქემების სახით მზადდება. მას ხშირად ***ძირითად მეხსიერებასაც*** უწოდებენ.

▲ ***გარე (ჩვეულებრივ, დისკური) მეხსიერება***, რომელში არსებულ ინფორმაციის გამოყენებისათვის მიკროპროცესორს სჭირდება «შუამავალი», რომელიც ამ ინფორმაციას გადმოაკოპირებს ოპერატიულ (შიგა) მეხსიერებაში, ე. ი. პროცესორი ამ ინფორმაციას უშუალოდ ვერ მოიხმარს. შუამავლის როლს თამაშობს სპეციალური პროგრამების (***დრაივერების***) მეშვეობით ფუნქციონირებადი ***კონტროლერები***.

შიგა (ძირითად) მეხსიერებას მიეკუთვნება ოპერატიული და მუდმივი მეხსიერება. ***ოპერატიულ მეხსიერებაში*** შეინახება მოცემულ მომენტში მიკროპროცესორის მიერ რეალიზებადი სამომხმარებლო პროგრამები და მონაცემები. მუდმივ მეხსიერება გამოიყენებს შეტანა-გამოტანის ბაზურ სისტემაში (ე. წ. ***BIOS***-ში) შემავალი დამხმარე ქვეპროგრამები და იმ დაფებისათვის (მაგალითად, ვიდეოადაპტერისათვის) განკუთვნილი დრაივერები, რომლებიც აქტიურდება ამუშავების საწყის ეტაპზე. არსებობს მუდმივი მეხსიერების მრავალი სახესხვაობა (***ROM, PROM, EPROM, EEPROM, Flash Mrmory, FRAM***), რომლებიც ერთმანეთისაგან აგებულებისა და გამოყენების სფეროების მიხედვით განსხვავდება.

§2.4-ში განხილული გვაქვს მეხსიერების კლასიფიცირებასთან დაკავშირებული ზოგიერთი ძირითადი საკითხი და მათ ახლა არ შევე-

ხებით, აქ ყურადღებას გავამახვილებთ მხოლოდ მეხსიერების ლოგიკური ორგანიზაციის ისეთ სპეციფიკურ საკითხებზე, როგორებიცაა ვირტუალური მეხსიერების კონცეფცია და მეხსიერების გვერდობრივი ორგანიზაციის პრინციპი.

მეხსიერების უჯრედების დამისამართებისათვის გამოიყენება მისამართების ***16***-თანრიგიანი სალტე (იხ.ნახ.***5.3***). მისი საშუალებით შეიძლება დამისამართდეს მეხსიერების $2^{16}=65536$ უჯრედი. მაშასადამე, მას შეესაბამება ***შესაძლო სამისამართო სივრცე*** ამ უკანასკნელს წარმოქმნის მისამართები, რომელთა ათობითი ეკვივალენტების ***0,1,2,...,65535.***

ძირითადი მეხსიერება შეიცავს ***4096*** უჯრედს. ამ უჯრედების მისამართები, რომელთა ათობითი ეკვივალენტებია ***0,1,2,....4095,*** წარმოქმნის ***ფიზიკურ სამისამართო სივრცს.***

ადრეულ წლებში ფიზიკურ სამისამართო სივრცის და შესაძლო სამისამართო სივრცი მოცულობები ერთმანეთის ტოლი იყო, ე.ი. არსებობდა ურთიერთცალსახა შესაბამისობა შესაძლო და ფიზიკურ მისამართებს შორის. როგორც ვხედავთ, შესაძლო სამისამართო სივრცის მოცულობამ მნიშვნელოვნად ჩამოიტოვა უკან ფიზიკური სამისამართო სივრცე და დაირღვა ზემოთ აღნიშნული ურთიერთშესაბამისობა. შესაძლო სამისამართო სივრცეს, რომლის მოცულობა აღემატება ფიზიკურ სამისამართო სივრცეს, ***ვირტუალური სამისამართო სივრცე*** ეწოდა. წარმოქმნილი შეუსაბამობისაგან თავის დასაღწევად ვირტუალური (შესაძლო) სამისამართო სივრცე დაიყო ***სასარგებლო*** (***0***-დან ***4095***-მდე) და ***უსარგებლო მისამართებად*** (***4096***-დან ***65535***-მდე). დროის ნებისმიერ მომენტში შეიძლება კავშირი დამყარებულიყო მეხსიერების მხოლოდ პირველ ***4096*** (***0***-დან ***4095***-მდე მისამართის მქონე) უჯრედთან. უსარგებლო მისამართებთან შეცდომით მიმართვისას პროცესორი შეცდომის წარმოშობას აფიქსირებდა, კერძოდ, გამოიმუშავებდა არარსებულ მეხსიერებასთან მიმართვის სიგნალს.

ვიტუალური მეხსიერების კონცეფცია წარმოადგენს რეალური მეხსიერების მისამართების სივრცეში ვირტუალური სამისამართო სივრცის მისამართების (რომლისთვისაც შეეძლო მიემართა პროცესორს) ასახვას (ნახ.***8.6***). ამ დროს იგულისხმება, რომ:

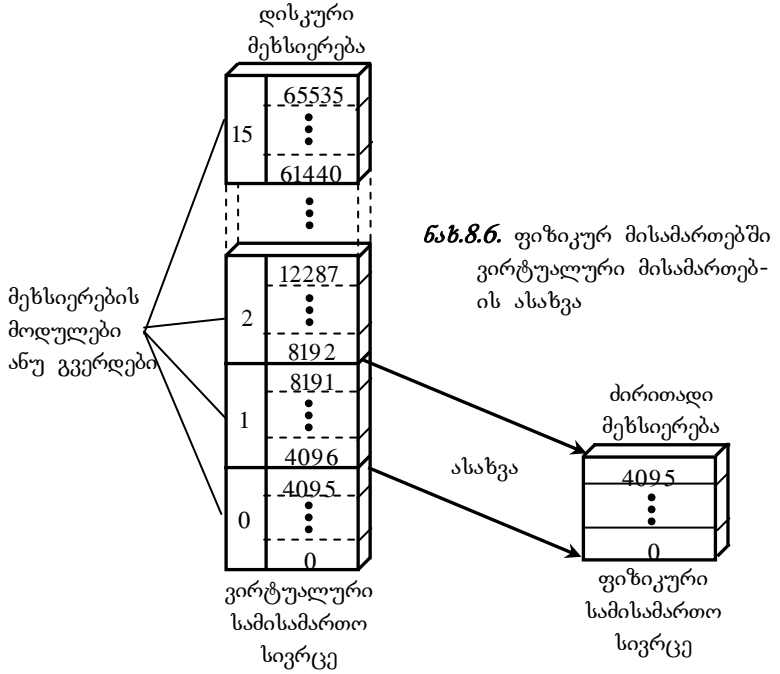
▲ გარდა ძირითადი მეხსიერებისა, მიკროპროცესორულ სისტემაში არის გარე (დისკური) მეხსიერება, რომელშიც შენახულია პროგრამა;

▲ სისტემას აქვს ასახვის განმახორციელებელი საშუალებები.

ვირტუალური მეხსიერების მოქმედების მექანიზმი ასეთია:

▲ ***0***-დან ***4095***-მდე მისამართებიანი მოდული ***1*** გადაკოპირდება ძირითად მეხსიერებაში და მოხდება მისი რეალიზება;

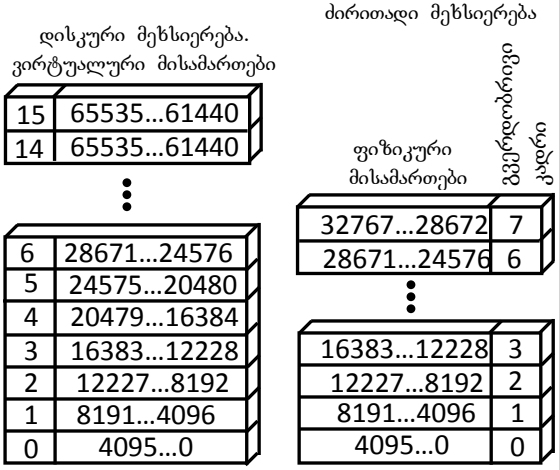
▲ მოდულ ***1***-მა თუ არ განიცადა ცვლილებები, ძირითად მეხსიერებაში გადაკოპირდება ***4096***-დან ***8191***-მდე მისამართებიანი მოდული ***2*** და შესრულდება; საწინააღმდეგო შემთხვევაში საჭიროა დასაწყისში დისკზე გადაიგზავნოს (დაბრუნდეს) მოდული ***1.***



ნახ.8.6. ფიზიკურ მისამართებში ვირტუალური მისამართების ასახვა

ვირტუალური მეხსიერების ზემოთ განხილული მოდულური სტრუქტურა განეკუთვნება ე. წ. ერთგანზომილებიან მეხსიერებას. ***ერთგანზომილებიანი მეხსიერება*** ეწოდება ისეთ მეხსიერებას, რომლის სამისამართო სივრცე შედგება ***0***-დან გარკვეულ მაქსიმალურ

მნიშვნელობამდე წრფივად (მიმდევრობითად) მზარდი მისამართებისაგან. ასეთ სტრუქტურას ეწოდება ***მეხსიერების გვერდობრივად ორგანიზებული სტრუქტურა***, ხოლო მეხსიერებაში შემავალ მოდულებს – ***მეხსიერების გვერდები***. რამდენიმე ერთგანზომილებიანი სამისამართო სივრცეთა ერთობლიობა წარმოქმნის მრავალგანზომილებიან სამისამართო სივრცეს. მრავალგანზომილებიანი სამისამართო სივრცის გამოყენებით ვირტუალური მეხსიერება გვაძლევს მეხსიერების ორგანიზების სპეციფიკურ სახეს, რომელსაც უწოდებენ ***მეხსიერების სეგმენტურ სტრუქტურას,*** ხოლო ამ სტრუქტურის მიღების პროცესს – ***მეხსიერების სეგმენტირებას.*** სეგმენტური სტრუქტურის ერთეულს წარმოადგენს ***სეგმენტი,*** რომელიც თავისი არსით ერთგანზომილებიანი მეხსიერებაა. გვერდებისაგან განსხვავებით სეგმენტებს შეიძლება ჰქონდეს განსხვავებული ზომის სამისამართო ზომები.



ნახ. 8.7. მეხსიერების გვერდობრივი ორგანიზაციის ილუსტრაცია

მეხსიერების სეგმენტირების საკითხს მეტნაკლებად სრულად **§9. 2.2.**-ში შევეხებით, ხოლო აქ მხოლოდ ***მეხსიერების გვერდობრივი ორგანიზების პრინციპს*** განვიხილავთ. ამ პრინციპის თანახმად ვირტუალური სამისამართო სივრცე თანაბარი ზომის გვერდებისაგან შედგება. ფიზიკური სამისამართო სივრცეც იყოფა გვერდის ტოლ ცალკეულ ნაწილებად. თითოეულ ასეთ ნაწილს ***ეწოდება გვერდო-***

ბრივი კადრი. 8.3 ნახაზზე მოყვანილი ძირითადი მეხსიერება მხოლოდ ერთ გვერდობრივ კადრს შეიცავს. რამდენიმე, კერძოდ, რვა გვერდობრივი კადრის მაგალითი ***8.7*** ნახაზზეა მოყვანილი, რომლის თანახმადაც:

▲ვირტუალური სამისამართო სივრცის ზომაა ***64*** კილობაიტი და იგი დაყოფილია ***16*** გვერდად; თითოეულის გვერდის ზომაა ***4*** კილობაიტი;

▲ძირითადი მეხსიერების ფიზიკური სამისამართო სივრცის ზომაა ***32*** კილობაიტი და იგი შეიცავს ***8*** გვერდობრივ კადრს, რომელთაგანაც თითოეულის ზომაა ***4*** კილობაიტი.

მეხსიერების გვერდობრივი ორგანიზების დროს ***16***-თანრიგიანი ვირტუალური მისამართი აისახება ძირითადი მეხსიერების ***15***-თანრიგიან ფიზიკურ მისამართში. მეხსიერების ***სეგმენტირების საკითხს*** მეტნაკლებად დაწვრილებით ***§9.2.2***-ში შევეხებით.

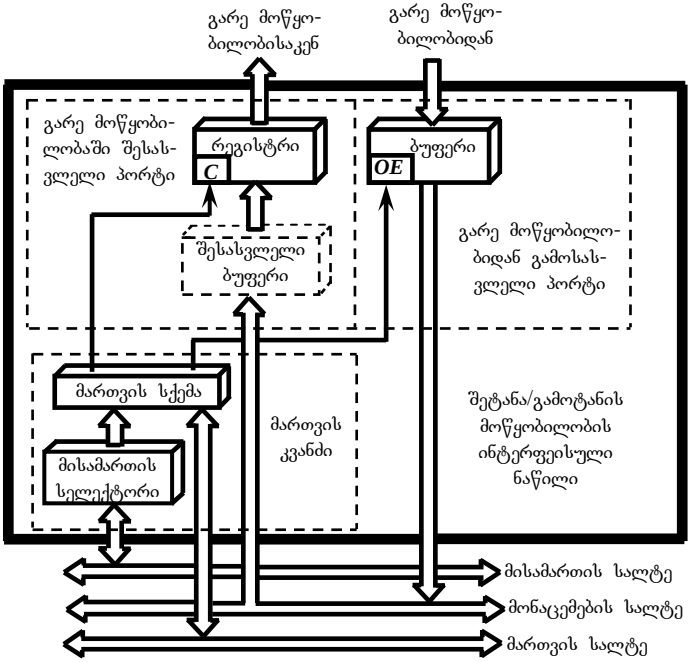
8.4. შეტანა/გამოტანის მოწყობილობათა ფუნქციები

შეტანა/გამოტანის მოწყობილობები მაგისტრალთან თითქმის იგივე პრინციპებით ცვლის ინფორმაციას, როგორც მეხსიერება. ინფორმაციის გაცვლის თვალსაზრისით ყველაზე მნიშვნელოვანი განსხვავებაა ის, რომ მეხსიერების მოდულს სისტების სამისამართო სივრცეში გააჩნია ბევრი (რამდენიმე ათეულ მილიონამდე) მისამართი, ხოლო შეტანა/გამოტანის მოწყობილობას ჩვეულებრივ მცირე რაოდენობის (ათამდე), ხოლო ზოგჯერ მხოლოდ ერთადერთი მისამართი.

მეხსიერების მოდულები ინფორმაციას ცვლის მხოლოდ მაგისტრალთან და პროცესორთან, ხოლო შეტანა/გამოტანის მოწყობილობები დამატებით ურთიერთზემოქმედებს ციფრული ან ანალოგური სახის გარე მოწყობილობებთანაც. ამიტომ მეხსიერების მოდულებთან შედარებით გაცილებით მრავალფეროვანია შეტანა/გამოტანის მოწყობილობები. ხშირად მათ ისეთი განსხვავებული სახელებითაც მოიხსენებენ, როგორიცაა: შეუღლების მოწყობილობები, კონტროლერები, გაფართოების ბარათები, ინტერფეისული მოდულები და ა.შ.

შეტანა/გამოტანის მოწყობილობები მაგისტრალთან ერთნაირი პრინციპით ცვლის; ამიტომ მაგისტრალთან შეუღლების კვანძებიც ერთნაირი პრინციპების გამოყენებითაა რეალიზებული. შეტანა/გამოტანის

მოწყობილობის, უფრო ზუსტად, მისი საინტერფესიო ნაწილის გამარტივებული სტრუქტურა ***8.8*** ნახაზზეა ნაჩვენები. მეხსიერების მოდულის მსგავსად იგიც აუცილებლად შეიცავს მისამართის სელექტორის, მონაცემების სტრობების დამუშავების მართვის სქემებსა და მონაცემთა ბუფერებს.



ნახ. 8.8. *შეტანა/გამოტანის უმარტივესი მოწყობილობის სტრუქტურა*

შეტანა/გამოტანის უმარტივესი მოწყობილობები გარე მოწყობილობებისათვის მონაცემების კოდს პარალელური ფორმატით გაიტანს და მათგან ამავე ფორმატით მიიღებს მას. შეტანა/გამოტანის ასეთ მოწყობილობებს ხშირად ***შეტანა/გამოტანის პარალელურ პორტებსაც*** უწოდებენ. ისინი ყველაზე უნივერსალურია, ე. ი. აკმაყოფილებს დიდი რაოდენობის გარე მოწყობილობებთან შეუღლების მოთხოვნებს; ამიტომ მათ ხშირად მიკროპროცესორული სისტემის სტუქტურაში სტანდარტულ მოწყობილობებად გამოიყენებენ. ჩვეულებრივ პარალელურ პორტები შედის მიკროკონტროლერების შემადგენლობაში. მიკ-

როკონტროლერი სწორედ პარალელური პორტებით უკავშირდება გარე სამყაროს.

უმარტივეს შემთხვევაში ***შესასვლელ პორტს (შეტანის პორტს)*** წარმოადგენს პარალელური რეგისტრი, რომელშიც პროცესორს შეუძლია ჩაწეროს ინფორმაცია. ***გამოსასვლელ პორტად (გამოტანის პორტად)*** ჩვეულებრივ გამოიყენება ერთმხივ მიმართული ბუფერი, რომლის მეშვეობითაც პროცესორს შეუძლია გარე მოწყობილობიდან წაიკითხოს ინფორმაცია. სწორედ ასეთი პორტებია ნაჩვენები ***8.8*** ნახაზზე მოყვანილ სქემაზე. პორტი შეიძლება ორმხრივმიმართულიც იყოს, ე. ი. შეასრულოს როგორც შესასვლელი, ისე გამოსასვლელი პორტის ფუნქცია. ასეთ შემთხვევაში პროცესორი იმ მისამართიდან წაიკითხავს ინფორმაციას, რომელ მისამართზედაც ახდენს ინფორმაციის ჩაწერას. ეს გარემოება გარე მოწყობილობასთან დამაკავშირებელი შესასვლელი და გამოსასვლელი ხაზების გაერთიანებისა და ორმხრივიმართული ხაზის ფორმირების საშუალებას გვაძლევს.

მაგისტრალის მხრიდან მიმართვის შემთხვევაში ***მისამართის სელექტორი*** ამოიცნობს შეტანა/გამოტანის მოცემული მოწყობილობისათვის მინიჭებულ მისამართს. ***მართვის სქემა*** ინფორმაციის გაცვლის მაგისტრალური სტრობების საპასუხოდ გასცემს გაცვლის შინაგან სტრობებს. მონაცემების შესასვლელი ***ბუფერი*** შეტანა/გამოტანის ამ მოწყობილობასთან ელექტრულად შეათანხმებს მონაცემების სალტეს (შესაძლებელია ბუფერი არც არსებობდეს). რეგისტრში მონაცემების სალტიდან მოსული მონაცემები ***C*** სიგნალით ჩაიწერება და გაიტანება გარე მოწყობილობაში (იხ. ***ნახ.8.8***). პორტიდან წაკითხვის ციკლში მონაცემების გამოსასვლელი ***ბუფერი*** მონაცემებს გარე მოწყობილობიდან გადასცემს მაგისტრალის მონაცემების სალტეზე.

შეტანა/გამოტანის უფრო რთული მოწყობილობების (შეულღების მოწყობილობების) შემადგენლობაში შედის შინაგანი ***ბუფერული ოპერატიული მეხსიერება*** და, უფრო მეტიც, მათ შეიძლება საკუთარი ***მიკროკონტროლერიც*** ჰქონდეს.

შეტანა/გამოტანის თვითოეულ მოწყობილობას მიკროპროცესორული სისტემის სამისამართო სივრცეში გამოეყოფა საკუთარი მისამართი. გამორიცხული უნდა იყოს მისამართების დუბლირება, რასაც თვალი უნდა ადევნოს მიკროპროცესორული სისტემის დამმუშავებელმა და მომხმარებელმაც.

შეტანა/გამოტანის მოწყობილობებმა შეიძლება ინფორმაცია არა მარტო პროგრამულად, არამედ შეწყვეტების შემწეობითაც გაცვალოს. ასეთ შემთხვევაში ისინი გარე მოწყობილობიდან შემოსულ შეწყვეტის მომთხოვნ სიგნალს გარდაქმნის მოცემული მაგისტრალისათვის აუცილებელ შეწყვეტის მოთხოვნის სიგნალად (ან სიგნალების მიმდევრობად ***ვექტორული შეწყვეტის დროს***). მეხსიერებასთან ინფორმაციის უშუალოდ გასაცვლელად შეტანა/გამოტანის მოწყობილობამ მაგისტრალზე უნდა გასცეს ***DMA***-ს მოთხოვნა და იმუშაოს მოცემული მაგისტრალისთვის მიღებულ ციკლებში.

მიკროპროცესორულ სისტემებში, როგორც წესი, გამოჰოფენ შეტანა/გამოტანის მოწყობილობების შემდეგ სამ ჯგუფს: ***1)*** მომხმარებლის ინტეფეისის მოწყობილობებს (რომელიც განკუთვნილია მომხმარებლის მიერ ინფორმაციის შესატანად და მომხმარებლისათვის ინფორმაციის გამოსატანად); ***2)*** ინფორმაციის ხანგრძლივად შენახვისათვის განკუთვნილ მოწყობილობებს; ***3)*** ტაიმერულ მოწყობილობებს.

მომხმარებლის ინტერფეისისათვის განკუთვნილი ***შეტანის მოწყობილობებს*** მიეკუთვნება კლავიატურის, ტუმბლერების, ცალკეული ღილაკების, მაუსის, ტრეკბოლის, ჯოისტიკისა და ა. შ. კონტროლერები. მომხმარებლის ინტერფეისისათვის განკუთვნილი ***გამოტანის მოწყობილობებს*** მიეკუთვნება შუქდიოდური ინდიკატორების, ტაბლოს, სხვადასხვა სახის ეკრანის და ა.შ. კონტროლერები. უმარტივესი მმართველი კონტროლერების ან მიკროკონტროლერების შემთხვევაში ეს საშუალებები შეიძლება არ არსებობდეს, ხოლო რთულ მიკროპროცესორულ სისტემებში ისინი აუცილებლად გამოიყენება. მოცეცემულ შემთხვევაში გარე მოწყობილობის როლს ასრულებს ადა-მიანი.

ინფორმაციის ხანგრძლივად შენახვისთვის განკუთვნილი შეტანა/გამოტანის მოწყობილობები უზრუნველყოფს მიკროპროცესორული სისტემის შეუღლებას კომპაქტური ან მაგნიტური დისკების დისკოსატარებთან, აგრეთვე მაგნიტურ ლენტზე დამგროვებლებთან.

ტაიმერული სისტემები შეტანა/გამოტანის სხვა მოწყობილობებისაგან იმით განსხვავდება, რომ მათ შეიძლება არ ჰქონდეს გარე მოწყობილობებთან მისაერთებელი გამომყვანები. ისინი იმისათვისაა განკუთვნილი, რომ მიკროპროცესორულმა სისტემამ შეძლოს: დაიცვას

დასახული დროითი ინტერვალების, თვალყური ადევნოს რეალურ დროს, აითვალის იმპულსები და ა. შ. ნებისმიერი ტაიმერი ეფუძნება ურთიერთამუშავების უნარის მქონე კვარცულ ტაქტურ გენერატორსა და მრავალთანრიგა ორობით მთვლელებს. პროცესორს შეუძლია ტაიმერში ჩაწეროს ტაქტური სიხშირის გაყოფის კოეფიციენტები და დასათვლელი იმპულსების რაოდენობა, დასახოს ტაიმერის მთვლელების მუშაობის რეჟიმი; რაც შეეხება წაკითხვას, იგი კითხულობს მთვლელების გამოსასვლელ კოდებს. შესაძლებელია ტაიმერის ყველა ფუნქცია პროგრამულადაც შესრულდეს, ამიტომ შეიძლება ზოგიერთ სისტემებში ისინი არც არსებობდეს; მაგრამ სისტემაში ტაიმერის ჩართვა უფრო რთული ამოცანების გადაწყვეტისა და უფრო ეფექტური ალგორითმების აგების საშუალებას იძლევა.

IX თავი
პროცესორის ფუნქციონირების საფუძვლები

9.1. ზოგადი საკითხები

პროცესორის ანუ ***ჩარლზ ბებიჯისეული*** წისქვილის (**mill**-ის), ძირითადი ფუნქციაა შეასრულოს ორობითი კოდების სახით წარმოდგენილ მონაცემებზე არითმეტიკული და ლოგიკური ოპერაციების ჩასატარებლად საჭირო ბრძანებები. აღნიშნული ***ბრძანებების სისტემა*** ჩამოგავს ლოგიკური ელემენტების ***ჭეშმარიტობის ცხრილს*** (ლოგიკური მიკროსქემების ***მუშაობის რეჟიმების ცხრილს***); კერძოდ, ისინი განსაზღვრავს პროცესორის რეაქციას მის შესასვლელებზე მოქმედი სიტყვებზე, ე. ი. პროცესორის მუშაობის ლოგიკას.

მიკროპროცესორული სისტემის დამუშავების პროცესში პროგრამების შედგენა უმნიშვნელოვანესი და ხშირად მეტად შრომატევადი ეტაპია. ***ეფექტური პროგრამების შესადგენად*** აუცილებელია, თუნდაც ზოგადი წარმოდგენა მაინც გვქონდეს პროცესორის მიერ გამოყენებულ ***ბრძანებათა სისტემაზე***. კომპაქტური და სწრაფი პროგრამებისა და ქვეპროგრამების შედგენა ***ასემბლერის ენაზეა*** შესაძლებელი. მისი გამოყენებისათვის პროცესორის ბრძანებათა სისტემის ცოდნაა საჭირო; ეს იმითაა განპირობებული, რომ ასემბლერი წარმოადგენს ადამიანის მიერ ადვილად აღსაქამელი (სიმბოლური) ფორმით ბრძანებათა კოდების ჩაწერის ფორმატს.

პროგრამული უზრუნველყოფის დამუშავებისათვის არსებობს სხვადასხვა პროგრამული საშუალებაც, კერძოდ მაღალი დონის დაპროგრამების ენები (პასკალი, სი და ა. შ.); მათ გამოსაყენებლად პროცესორის ბრძანებების ცოდნა საჭირო არ არის, მაგრამ ბრძანებათა სისტემისა და ასემბლერის ენის ცოდნა ნებისმიერი მიკროპროცესორული სისტემის პროგრამული უზრუნველყოფის ზოგიერთი უმნიშვნელოვანესი ნაწილების ეფექტურობის რამდენჯერმე გაზრდის საშუალებას იძლევა. ამიტომ მოცემულ თავში განვიხილავთ პროცე-

სორების უმრავლესობაში გამოყენებული ძირითადი ტიპის ბრძანებებსა და მათი გამოყენების თავისებურებებს.

პროცესორის მეხსიერებიდან ამოსარჩევი (წასაკითხი) თითოეული ბრძანება უახლოესი რამდენიმე ტაქტის განმავლობაში განსაზღვრავს პროცესორის ქცევის ალგორითმს. ***ბრძანების კოდი*** პროცესორს განუსაზღვრავს: *1)* რომელ ***ოპერანდებზე*** (ე.ი. მონაცემებზე) რა ოპერაცია უნდა შეასრულოს; *2)****საიდან უნდა აიღოს*** ბრძანების შესასრულებლად საჭირო ინფორმაცია; *3)* მიღებული შედეგი (თუ ეს საჭიროა) სად უნდა მოათავსოს.

ბრძანების კოდ შეიძლება ერთი ან რამდენიმე ბაიტისაგან შედგებოდეს; პროცესორისთვის ბრძანების კოდის პირველივე ბაიტის ან სიტყვის წაიკითხისთანავე ხდება ნათელი, აღნიშნული კოდის თუ რამდენი ბაიტი უნდა წაიკითხოს მან. ბრძანების კოდი პროცესორში გაიშიფრება და გარდაიქმნება პროცესორის ცალკეული კვანძების მიერ შესასრულებელი ***მიკროოპერაციების ნაკრებად***. მისი ცოდნა მიკროპროცესორული სისტემის შემქმნელისათვის ძალიან მნიშვნელოვანი არ არის. მისთვის მხოლოდ ამა თუ იმ ბრძანების შედეგის ცოდნაა მნიშვნელოვანი.

9.2. ოპერანდების დამისამართება

პროცესორის ბრძანებათა უდიდესი ნაწილის ფუნქციონირება მონაცემების კოდებთან (ოპერანდებთან) არის დაკავშირებული. ზოგიერთი მათგანი გადაამუშავებს ***შესასვლელ*** (ერთ ან ორ) ***ოპერანდს***, სხვები კი გამოიმუშავებს (უმეტესწილად, ერთადერთ) ***გამოსასვლელ ოპერანდს.*** შესასვლელ ოპერანდებს ***წყარო ოპერანდები,*** ხოლო გამოსასვლელ ოპერანდებს – ***მიმღები ოპერანდები*** ეწოდება. შესასვლელი და გამოსასვლელი ოპერანდების ყველა ეს კოდი სადღაც უნდა იყოს განთავსებული. ისინი შეიძლება განთავსდეს პროცესორის შიგა რეგისტრებში (ყველაზე მოსახერხებელი და სწრაფი ვარიანტი), სისტემურ მეხსიერებაში (ყველაზე გავრცელებული ვარიანტი), ან შეტანა/გამოტანის მოწყობილობაში (ყველაზე იშვიათი შემთხვევა). ოპერანდების ადგილმდებარეობას განსაზღვრავს ბრძანების კოდი. ამასთანავე არსებობს სხვადასხვა მეთოდი იმისა, თუ საიდან უნდა იქნეს აღებული ***შესასვლელი ოპერანდი,*** და სად უნდა მოთავსდეს ***გა-***

მოსასვლელი ოპერანდი. ამ მეთოდებს ეწოდება ***დამისამართების მეთოდები***. დამისამართების ამორჩეული მეთოდის ეფექტურობა მნიშვნელოვანწილად განსაზღვრავს მთლიანად მიკროპროცესორის მუშაობის ეფექტურობას.

9.2.1. დამისამართების მეთოდები

სხვადასხვა პროცესორებში დამისამართების მეთოდების რაოდენობა ***4***-დან ***16***-მდე იცვლება. განვიხილოთ დღეისათვის მიკროპროცესორების უმრავლესობაში გამოყენებული დამისამართების მეთოდების რამდენიმე ტიპი.

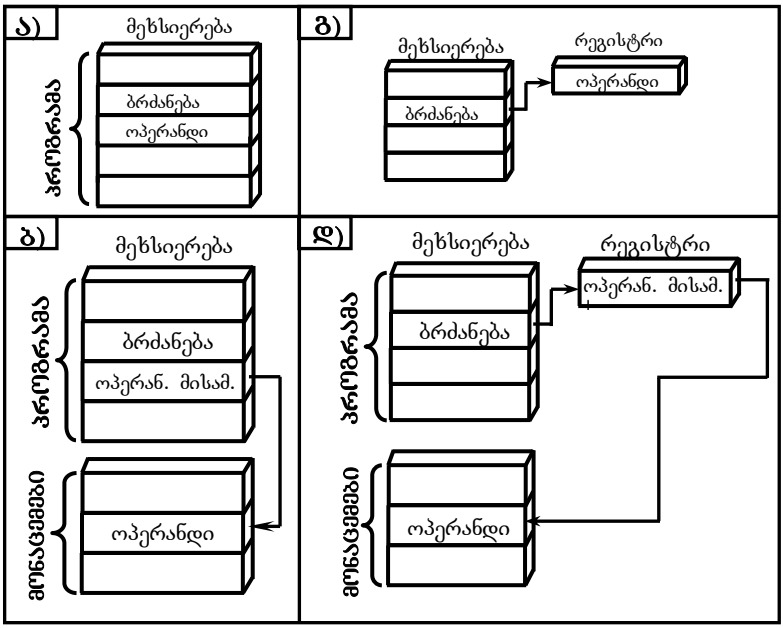
უშუალო დამისამართების დროს (ნახ. ***9.1,ა***) შესასვლელი ოპერანდი მეხსიერებაში უშუალოდ ბრძანების კოდის შემდეგ არის განთავსებული. ოპერანდს ჩვეულებრივ წარმოადგენს კონსტანტა, რომელიც სადღაც უნდა გადაიგზავნოს, ან რომელიმე შიგა რეგისტრის შიგთავსს მიემატოს და ა. შ. მაგალითად, შეიძლება არსებობდეს პროცესორის რომელიმე შინაგანი რეგისტრის შიგთავსისასათვის რიცხვ ***8***-ის მიმატების ბრძანება. ეს რიცხვი ***8*** მეხსიერებაში განთავსდება პროგრამაში არსებული შეკრების ბრძანების შემდეგ არსებულ მისამართზე.

პირდაპირი (ანუ აბსოლუტური) დამისამართების დროს (ნახ.***9.1,ბ***) მეხსიერებაში განთავსებულ ბრძანების კოდს უშუალოდ მოსდევს არა ოპერანდი, არამედ ამ ოპერანდის ადგილმდებარეობის განმსაზღვრელი მისამართი. მაგალითად, განვიხილოთ მეხსიერების იმ უჯრედის გაწმენდის (შიგთავსის განულების) ბრძანება, რომლის მისამართია ***1000000***; ეს მისამართი განთავსდება უშუალოდ აღნიშნული ბრძანების კოდის შემდეგ.

რეგისტრული დამისამართების დროს (ნახ.***9.1,გ***) ოპერანდი თავსდება პროცესორის შიგა რეგისტრში. ასეთი შეიძლება იყოს, მაგალითად, ერთი რეგისტრიდან მეორეში რიცხვის გადაგზავნის ბრძანება, რომლითაც რიცხვი გადაიცემა რეგისტრ ***1***-დან რეგისტრი ***2***-ში. ორივე რეგისტრის ნომერი (***1*** და ***2***) განისაზღვრება გადაგზავნის ბრძანების კოდით.

ირიბულ-რეგისტრული (ანუ ირიბული) დამისამართების დროს პროცესორის შიგა რეგისტრში თავსდება არა თავად ოპერანდი, არა-

მედ მეხსიერებაში ამ ოპერანდის მისამართი (ნახ. ***9.1,დ***). მაგალითად, ასეთია მეხსიერების იმ უჯრედის გაწმენდის ბრძანება, რომლის მისამართი მოთავსებულია ნულოვან რეგისტრში. ამ რეგისტრის ნომერი (0) განისაზღვრება გაწმენდის ბრძანების კოდით.



ნახ. 9.1. *უშუალო დამისამართება (ა), პირდაპირი დამისამართება (ბ), რეგისტრული დამისამართება (გ), ირიბი დამისამართება (დ)*

იშვიათად გვხვდება დამისამართების შემდეგი ორი მეთოდი:

1. ***ავტოინკრემენტალური დამისამართება*** ძალიან ჰგავს ირიბ დამისა-მართებას და მისგან მხოლოდ იმით განსხვავდება, რომ ბრძანების შესრულების შემდეგ გამოყენებული შიგთავსი ერთი ან ორი ერთე-ულით იზრდება. დამისამართების ეს მეთოდი ძალიან მოსახერხებე-ლია, მაგალითად, მეხსიერებაში არსებული მონაცემთა მასივში შემა-ვალი კოდების დასამუშავებლად. კონკრეტული კოდის დამუშავების შემდეგ რეგისტრში გაჩენილი ახალი მისამართი უკვე გვიჩვენებს მა-სივის მომდევნო დასამუშავებელ კოდს. მოცემულ შემთხვევაში რომ ირიბი დამისამართება გამოგვეყენებინა, მაშინ

დაგვჭირდებოდა დამტებითი ბრძანებით გაგვეზარდა ამ რეგისტრის შიგთავსი.

2. ***ავტოდეკრემენტალური დამისამართება*** ჰგავს ავტოინკრემენტალურ დამისამართებას, ოღონდ ამორჩეული რეგისტრის შიგთავსი ბრძანების შესრულების წინ ერთი ან ორი ერთეულით კი არ იზრდება, არამედ მცირდება. ეს დამისამართებაც მონაცემთა მასივების დასამუშავებლადაა მოსახერხებელი. ავტოინკრიმენტალური და ავტოდეკრიმენტალური დამისამართებები სტეკური ტიპის მეხსიერების (იხ. ***ნახ. 8.4***) მეხსიერების ორგანიზების საშუალებას იძლევა.

დამისამართების სხვა გავრცელებული მეთოდებიდან შეიძლება ვახსენოთ ***ინდექსური მეთოდები,*** რომელთა დროსაც ოპერანდის მისამართი გამოითვლება რეგისტრის შიგთავსისა და მოცემული კონსტანტას (ინდექსის) შეკრების გზით. ამ კონსტანტას კოდი მეხსიერებაში უშუალოდ ბრძანების კოდის შემდეგ არის განთავსებული.

დამისამართების ამა თუ იმ მეთოდის ამორჩევაზე მნიშვნელოვანწილადაა დამოკიდებული ბრძანების შესრულების ხანგრძლივობა. ***ყველაზე სწრაფია რეგისტრული დამისამართება***, რადგან ამ დროს საჭირო არ არის მაგისტრალში ინფორმაციის გაცვლის დამატებითი ციკლები. დამისამართება თუ მოითხოვს მეხსიერებასთან მიმართვას, მაშინ ამ მიმართვისათვის საჭირო ციკლების ხანგრძლივობათა გამო გახანგრძლივდება ბრძანების შესრულებაც. ნათელია, რომ რაც უფრო ბევრ შინაგან რეგისტრს შეიცავს პროცესორი, მით უფრო ხშირად შეგვეძლება გამოვიყენოთ რეგისტრული დამისამართება და, აქედან გამომდინარე, სისტემა მით უფრო სწრაფად იმუშავებს.

9.2.2. მეხსიერების სეგმენტირება

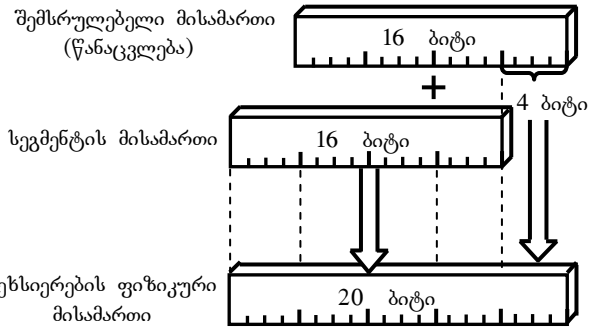
მეხსიერების სეგმენტირება (იხ.***§ 8.3***) პირველად განხორციელდა ***1978*** წელს ფირმა ***Intel***-ის მიერ დამუშავებულ პირველ ***16***-თანრიგიან ***I8086*** მიკროპროცესორში. მისი დამუშავების აუცილებლობა განაპირობა იმ გარემოებამ, რომ აღნიშნული ფირმის მიერ დამუშავებული პროცესორების შინაგანი რეგისტრები ***16***-თანრიგიანია, ხოლო ფიზიკური მისამართი კი ***20***-თანრიგიანი. ***16***-თანრიგიანი მისამართი მხოლოდ ***64*** კილობაიტიანი მოცულობის მეხსიერების გამოყენების საშუალებას გვაძლევს, რაც აშკარად არასაკმარისია. სწორედ ამ სი-

ტუაციიდან გამოსვლისათვის დამუშავდა მეხსიერების სეგმენტირების მეთოდი. იმ პერიოდში ფირმა ***Motorola***–ს მიერ გამოშვებულ ***MC68000*** მიკროპროცესორს კი ჰქონდა ***32***-თანრიგიანი შინაგანი რეგისტრები, ამიტომ მისთვის მეხსიერების სეგმენტირების პრობლემა არ წამოჭრილა.

მოკლედ განვიხილოთ ფირმა ***Intel***-ის მიერ გამოშვებულ სხვადასხვა მიკროპროცესორებში მეხსიერების სეგმენტირების პრინციპები.

I8086 პროცესორში მეხსიერება ასე არის სეგმენტირებული. სისტემის მთელი მეხსიერება წარმოდგენილია არა უწყვეტი სივრცის, არამედ მოცემული ზომის (***64*** კილობაიტის ტოლი) რამდენიმე ნაკვეთის სახით, რომელთა მდებარეობები მეხსიერების სივრცეში შეიძლება პროგრამული გზით შეიცვალოს.

მეხსიერების მისამართების კოდების შესანახად გამოიყენება არა ცალკეული რეგისტრები, არამედ რეგისტრთა წყვილები, რომლებიც შედგება: ***1)*** სეგმენტის დასაწყისის მისამართის (ე.ი. მეხსიერებაში სეგმენტის მდებარეობის) განმსაზღვრელი ***სეგმენტური რეგისტრისაგან***; ***2)*** სეგმენტის შიგნით სამუშაო მისამართის მდებარეობის განმსაზღვრელი ***მაჩვენებელი რეგისტრისაგან*** (***წანაცვლების რეგისტრისაგან***).

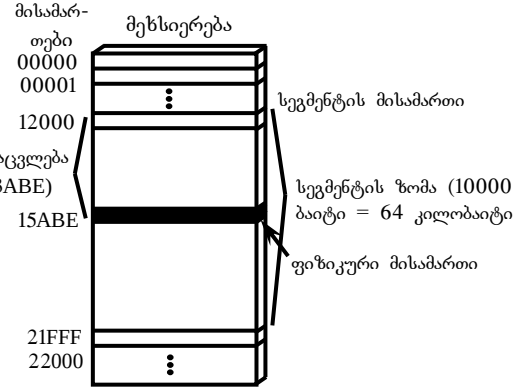


ნახ. 9.2. *სეგმენტის მისამართისა და წანაცვლებისაგან მეხსიერების ფიზიკური მისამართის ფორმირება*

მისამართის გარე სალტეზე გამოსატანი მეხსიერების ***20-თანრიგიანი ფიზიკური მისამართი*** წარმოიქმნება ისე, როგორც ეს ***9.2*** ნახაზზეა ნაჩვენები, ე. ი. წანაცვლებისა და სეგმენტის ***4*** ბიტით წანაც-

ვლებული მისამართის შეკრების გზით. ამის შედეგად მიღებული ფიზიკური მისამართის მეხსიერებაში განთავსების ადგილი ***9.3*** ნახაზზეა ნაჩვენები.

სეგმენტი შეიძლება დაიწყოს მეხსიერების მხოლოდ ***16***-ბაიტურ საზღვარზე (რადგან სეგმენტის მისამართის არსებითად აქვს ოთხი უმცროსი ნულოვანი თანრიგი, როგორც ***9.2*** ნახაზიდან ჩანს), ე.ი. ***16***-ის ჯერადი მისამართიდან. სეგმენტების ამ დასაშვებ საზღვრებს ეწოდება ***პარაგრაფების საზღვრები.***

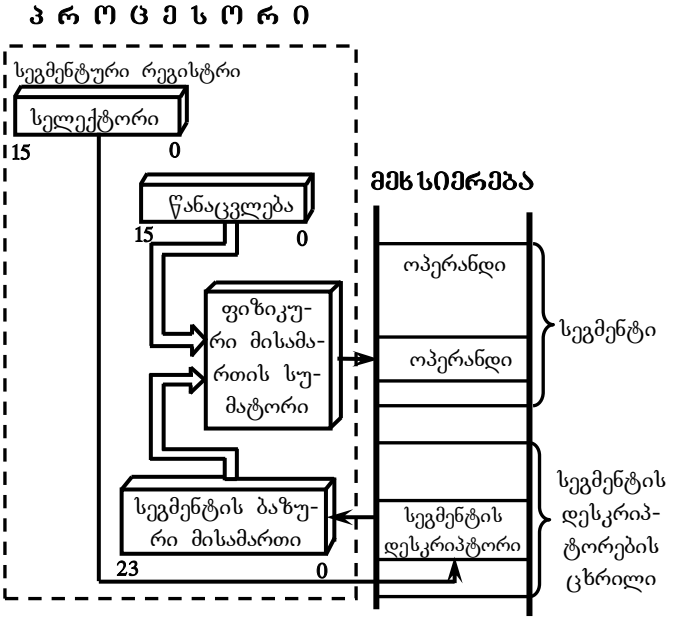


ნახ. 9.3. *სეგმენტში ფიზიკური მისამართი (კოდებად გამოყენებულია* ***16****-თანრიგიანი რიცხვები)*

მიკროპროცესორ ***Intel 80286***-ში მეხსიერების სეგმენტირების უფრო რთული მეთოდები გამოიყენება. მის ე. წ. დაცულ რეჟიმში მეხსიერების მისამართი გამოითვლება ***9.4*** ნახაზზე მოყვანილი სქემის შესაბამისად.

მოცემული შემთხვევის დროს სეგმენტურ რეგისტრში შეინახება არა სეგმენტების ბაზისური (საწყისი) მისამართი, არამედ მეხსიერებაში იმ მისამართების განმსაზღვრელი სელექტორების კოდები, რომლებშიცაა შენახული სეგმენტების დესკრიპტორები (ე. ი. აღმწერები). მეხსიერების დესკრიპტორიან სფეროს ეწოდება ***დესკრიპტორების ცხრილი.*** სეგმენტის თითოეული დესკრიპტორი შეიცავს სეგმენტის ბაზისურ მისამართს, სეგმენტის ზომას (რომელიც იცვლება ***1***-დან ***64*** კილობაიტამდე) და მის ატრიბუტებს. სეგმენტის ბაზისური

მისამართი ***24***-ბიტიანია, რაც ფიზიკური მეხსიერების ***16*** მეგაბაიტის დამისამართებას უზრუნველყოფს.



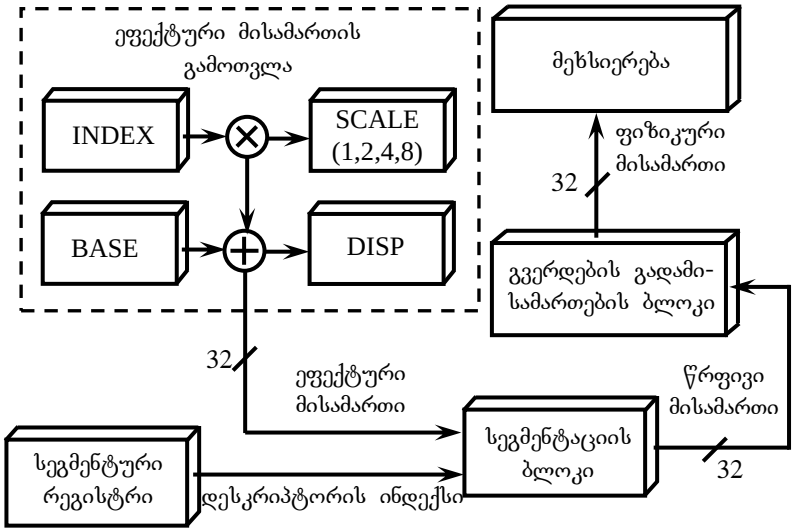
ნახ.9.4. *პროცესორ* ***Intel 80286*** *–ის მეხსიერების დამისამართება დაცულ რეჟიმში*

ამგვარად, მეხსიერების ფიზიკური მისამართის გამომთვლელ სუმატორს მიეწოდება არა სეგმენტური რეგისტრის შიგთავსი, როგორც ეს ხდებოდა წინა შემთხვევაში, არამედ დესკრიპტორების ცხრილიდან ***სეგმენტის ბაზისური მისამართი.***

■ სეგმენტირების კიდევ უფრო რთული მეთოდი გამოიყენება და ***Intel***-ის მიერ გამოშვებულ ***I80386*** და მომდევნო მოდელის პროცესორებში. ეს მოდელი ***9.5*** ნახაზზეა ილუსტრირებული.

მეხსიერების მისამართი (ფიზიკური მისამართი) სამ ეტაპად გამოითვლება. დასაწყისში ***სამი კომპონენტის,*** კერძოდ, ბაზის (*Base*), ინდექსისა (*Index*) და წანაცვლების (*Displacement*) შეკრებით გამოითვლება ***32***-თანრიგიანი ე. წ. ***ეფექტური მისამართი.*** ამ დროს შესაძლებელია მასშტაბზე (***Scale***) ინდექსის გამრავლება.

გავეცნოთ ამ კომპონენტებს. ***1) წანაცვლება*** წარმოადგენს ბრძანებაში ჩართულ ***8-***, ***16-*** ან ***32-***თანრიგიან რიცხვს; ***2) ბაზა*** (***Base***) პროცესორის ***ბაზისური რეგისტრის*** შიგთავსია. ჩვეულებრივ იგი გამოიყენება გარკვეული მასივის დასაწყისის მისათითებლად; ***3) ინდექსი*** (***Index***) პროცესორის ინდექსური რეგისტრის შიგთავსია. ჩვეულებრივ იგი გამოიყენება მასივის ერთ-ერთი ელემენტის ამოსარჩევად; ***4) მასშტაბი*** (***Scale***) იმ ბრძანების კოდში მითითებული ***1***-ის, ***2***-ის, ***4***-ის ან ***8***-ის ტოლი მამრავლი, რომელზეც მრავლდება ინდექსი სხვა კომპონენტებთან მის შეკრებამდე.



ნახ.9.5. *პროცესორ* ***Intel 80386*** *–ის ფიზიკური მისამართის ფორმირება დაცულ რეჟიმში*

ამის შემდეგ სეგმენტაციის სპეციალური ბლოკი გამოითვლის ***32***-თანრიგიან ***წრფივ მისამართს***, რომელიც მიიღება სეგმენტური რეგისტრიდან აღებული ბაზისური მისამართისა და ეფექტური მისამართის შეკრებით. და ბოლოს, სეგმენტური გადამისამართების ბლოკის მიერ წრფივი მისამართის გარდაქმნით წარმოიქმნება ***32***-ბიტური ***ფიზიკური მისამართი***; კერძოდ, წრფივ მისამართს იგი გადათარგმნის ოთხ-ოთხ კილობაიტიან ***ფიზიკურ გვერდად.***

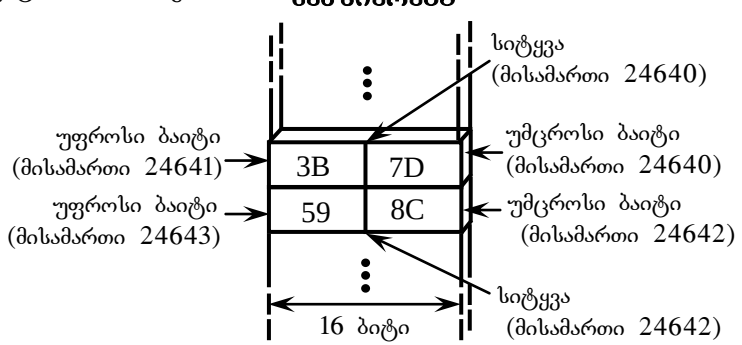
ნებისმიერ შემთხვევაში სეგმენტირება მეხსიერებაში ერთ ან რამდენიმე სეგმენტს გამოყოფს მონაცემებისათვის და ამდენივე სეგმენტს

– პროგრამებისათვის. ერთი სეგმენტიდან მეორეზე გადასვლა სეგმენტური რეგისტრის შიგთავსის შეცვლაზე დაიყვანება. ზოგჯერ ეს ძალიან მოსახერხებელია, მაგრამ სეგმენტირებულ მეხსიერებასთან მუშაობა უწყვეტ, არასეგმენტირებულ მეხსიერებასთან მუშაობაზე უფრო რთულია, რადგან საჭიროა სეგმენტთა საზღვრებისათვის, მათ აღმწერებისათვის, გადამრთველებისათვის და ა.შ. თვალყურის დევნება.

9.2.3. ბაიტებისა და სიტყვების დამისამართება

16 ან ***32*** თანრიგის მქონე ბევრ პროცესორს შეუძლია მეხსიერებაში არა მარტო (***16***–თანრიგიანი ან ***32***-თანრიგიანი) მთელი სიტყვის, არამედ მასში შემავალი ცალკეული ბაიტების დამისამართება, რისთვისაც თითოეული ბაიტს საკუთარი მისამართი გააჩნია.

16-თანრიგიანი პროცესორების შემთხვევაში მეხსიერებაში თითოეულ 16-თანრიგიან სიტყვას მიკუთვნებული აქვს ლუწი მისამართი, ხოლო მასში შემავალ ბაიტებს შეიძლება ჰქონდეს როგორც ლუწი, ისე კენტი მისამართები.



ნახ. 9.6. *სიტყვებისა და ბაიტების დამისამართება*

მაგალითისათვის განვიხილოთ მეხსიერების ***16***-თანრიგიანი უჯრედი, რომლის მისამართია ***24640*** და რომელშიც შენახულია სიტყვა ***3B7D*** (ნახ.***9.6***). მთლიანად ამ სიტყვაზე მიმართვისათვის პროცესორი გამოიტანს მისამართ ***24640***-ს და გამოიყენებს სიტყვის დამამისამართებელ ბრძანებას; მას თუ სურს მიმართოს ამ სიტყვის უმცროს ***7D*** ბაიტს, მაშინ გამოიტანს იმავე ***24640*** მისამართს, ოღონდ გამოიყ-

ენებს არა სიტყვის, არამედ ბაიტის დამამისამართებელ ბრძანებას. სიტყვის უფროს **3B** ბაიტთან მიმართვისათვის პროცესორი უკვე გამოიტანს **24641** მისამართსა და გამოიყენებს ბაიტის დამამისამართებელ ბრძანებას. რადგან **24640** და **24641** მისამართები უკვე გამოვიყენეთ, ამიტომ **598C** შიგთავსიან მომდევნო უჯრისათვის უნდა გამოვიყენოთ მომდევნო **24642 და 24643** მისამართები.

მაგისტრალზე ბაიტებისა და სიტყვების გაცვლის ციკლების ერთმანეთისაგან განსასხვავებლად მართვის სალტეში გამოიყენება ბაიტების გაცვლის მაჩვენებელი სპეციალური სიგნალი. ბაიტებთან სამუშაოდ ბრძანებათა სისტემაში გაითვალისწინება სპეციალური ბრძანება, ან გამოიყენება დამისამართების გარკვეული მეთოდები.

9.3. პროცესორის რეგისტრები

პროცესორის **შინაგანი რეგისტრი** სამომსახურო ინფორმაციის დროებით შესანახად განკუთვნილი მცირე **ზომის ზოპერატიული მეხსიერებაა**. სხვადასხვა პროცესორებში რეგისტრების რაოდენობა **6-8**-დან რამდენიმე ათეულამდე იცვლება. **განასხვავებენ სპეციალიზებულ და უნივერსალურ რეგისტრებს.** ძირითადად სპეციალიზებული რეგისტრებია ბრძანებების მთვლელი რეგისტრი, მდგომარეობის (**PSW**) რეგისტრი და სტეკის მაჩვენებელი რეგისტრი; დანარჩენები შეიძლება იყოს როგორც უნისავერსალური, ისე სპეციალიზებული. განვიხილოთ სხვადასხვა სახის მიკროპროცესორი.

- ფირმა **DEC**–ას **16**-თანრიგიან პროცესორ **T-11**–ში არის საერთო დანიშნულების **8** და მდგომარეობის ერათადერთი რეგისტრი. ყველა რეგისტრი **16**-თანრიგიანია. **საერთო დანიშნულების რეგისტრებიდან** (**სდრ**-ებიდან) ერთი გამოიყენება **ბრძანების მთვლელად**, ხოლო მეორე – **სტეკის მაჩვენებლად.** დანარჩენი რეგისტრი სრულად ურთიერთშენაცვლებადია, ე. ი. უნივერსალური დანიშნულებისაა და შეუძლია შეინახოს როგორც მონაცემები, ისე მისამართებიც (მაჩვენებლებიც), ინდექსებიც და ა. შ. მოცემული პროცესორის მეხსიერების მაქსიმალური დასაშვები მოცულობა **64** კილობაიტია (მეხსიერების მისამართი **16**-თანრიგიანია).
- **Motorola**–ს **16**-თანრიგიან პროცესორ **MC68000**–ში შემდეგი **19** რეგისტრია: მდგომარეობის **16**-თანრიგიანი რეგისტრი, ბრძანებე-

ბის მთვლელი ***32***-თანრიგიანი რეგისტრი, მისამართის ***32***-თანრიგიანი ***9*** და მონაცემების ამდენივე თანრიგიანი ***8*** რეგისტრი. დამისამართებადი მეხსიერების მაქსიმალური დასაშვები მოცულობაა ***16*** მეგაბაიტი (მისამართის გარე სალტე **24**-თანრიგიანია). მონაცემების რვავე და მისამართის შვიდივე რეგისტრი ურთიერთშენაცვლებადია.

■ პროცესორ ***I8086*** პროცესორში არსებულ თითოეულ რეგისტრი სპეციალური დანიშნულება და შეუძლია ერთმანეთი მხოლოდ ნაწილობრივ ან საერთოდ ვერ შეცვალოს. დაწვრილებით განვიხილოთ ამ პროცესორის რამდენიმე თავისებურება.

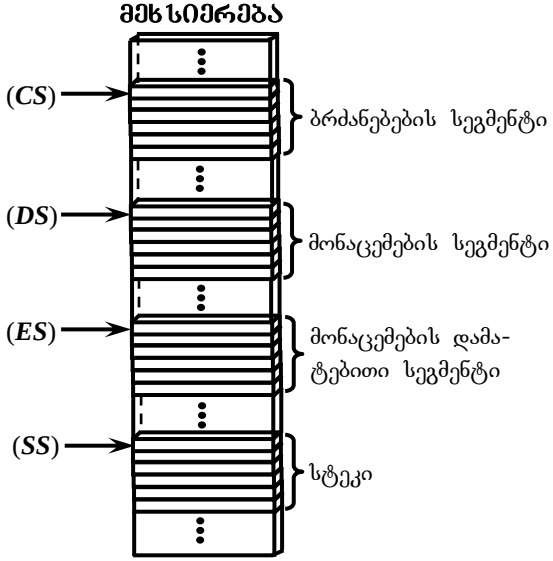
პროცესორს აქვს ***16***-თანრიგიანი ***14*** რეგისტრი. ამათგან ოთხი (***AX,BX,CX,DX***) მონაცემებისთვისაა განკუთვნილი; თითოეულ მათგანს ოპერანდებისა და ოპერაციათა შედეგების შენახვის გარდა საკუთარი შემდეგი სპეციფიკური დანიშნულება: ***1) AX***-რეგისტრი მონაწილეობს გამრავლების, გაყოფისა და შეტანა/გამოტანის მოწყობილობებთან ინფორმაციის გაცვლის პროცესებში; ***2) BX***-რეგისტრი მისამართის გამოთვლებში ბაზური რეგისტრის ფუნქციას ასრულებს; ***3***) ***CX***-რეგისტრი გამოიყენება ციკლების მთვლელად; ***4) DX***-რეგისტრი მონაწილეობს შეტანა/გამოტანის მისამართის განსაზღვრაში.

შესაძლებელია მონაცემების რეგისტრების ორივე ბაიტი იქნეს გამოყენებული (მაგალითად, ***AX*** რეგისტრში ამ ბაიტებს აქვს საკუთარი სახელწოდება: უმცროსი ბაიტის სახელია ***AL***, ხოლო უფროსი ბაიტის სახელი - ***AN***).

პროცესორის მომდევნო ოთხი შინაგანი რეგისტრს წარმოადგენს ე.წ. ***სეგმენტური რეგისტრები***. რომელთაგანაც თითოეული მათგანი სამუშაო სეგმენტებიდან ერთ-ერთი მათგანის მდგომარეობას განსაზღვრავს (ნახ. ***9.7***): ***1) CS*** (***Code Segment***) რეგისტრი შეესაბამება იმ ბრძანებების სეგმენტს, რომლებიც სრულდება მოცემულ მომენტში; ***2) DS*** (***Data Segment***) რეგისტრი შეესაბამება იმ მონაცემების სეგმენტს, რომელსაც მოცემულ მომენტში იყენებს პროცესორი; ***3) ES*** (***Extra Segment***) რეგისტრი შეესაბამება მონაცემების დამატებით სეგმენტს; ***4) SS*** (***Stack Segment***) რეგისტრი შეესაბამება სტეკის სეგმენტს.

მეხსიერების სივრცის ოპტიმალური გამოყენებისათვის პრინციპულად შესაძლებელია ყველა ამ სეგმენტმა ერთმანეთი გადაფაროს. მაგალითად, თუ პროგრამა მოიცავს მხოლოდ სეგმენტის ნაწილს, მა-

შინ მონაცემების სეგმენტი შეიძლება დაიწყოს არა პროგრამის მთელი სეგმენტის დამთავრების შემდეგ, არამედ პროგრამის დამთავრებისთანავე (***16*** ბაიტი სიზუსტით).



ნახ. 9.7. *მეხსიერებაში ბრძანებების, მონაცემებისა და სტეკის სეგმენტები*

პროცესორის შემდეგი ხუთი რეგისტრი (***SP – Stack Pointer, BP-Base Pointer, SI – Source Index, DI – Destination Index, IP – Instuction Pointer***), გამოიყენება მაჩვენებლებად (ე.ი. სეგმენტის ფარგლებში განსაზღვრავს წანაცვლებას). მაგალითად, ბრძანებების მთვლელი წარმოიშვება ***CS*** და ***IP*** რეგისტრების წყვილით, ხოლო სტეკის მაჩვენებელი - ***SP*** და ***SS*** რეგისტრების წყვილით. ***SI***, ***DI*** რეგისტრები გამოიყენება სტრიქონულ ოპერაციებში, ე.ი. ერთი ბრძანებით მეხსიერების რამდენიმე უჯრედის მიმდევრობით დამუშავების დროს.

უკანასკნელი ***FLAGS*** რეგისტრი წარმოადგენს პროცესორის მდგომარეობის (***PSW***) რეგისტრს. მასში არსებული ***16*** თანრიგიდან გამოიყენება შემდეგი ცხრა თანრიგი (ნახ. ***9.8.***): ***CF*** – (***Carry Flag***) – არითმეტიკული ოპერაციების დროს ***გადატანის ალამი; PF*** – (***Part Flag***) – ***შედეგის სილუწის ალამი; AF*** – (Auxiliar ***Flag***) – ***დამატებითი გადატანის; ZF*** – (***Zero Flag***) – ***ნულოვანი შედეგის ალამი; SF*** - (***Sign***

F**lag*) – ***ნიშნის ალამი (ემთხვევა შედეგის უფროს ბიტს); ***TF*** – (***T**rap **F**lag*) – ***ბიჯური რეჟიმის ალამი*** (გამოიყენება გამართვის დროს; ***IF*** – (***I**nterrupt-enable **F**lag*) – აპარატურული შეწყვეტების ნებადართვის ალამი; ***DF*** – (***D**irection **F**lag*) – სტრიქონული ოპერაციების დროს მიმართულების ალამი; ***OF*** – (***O**verflow **F**lag*) – გადავსების ალამი.

ნახ. 9.8. 8086 *პროცესორის მდგომარეობის რეგისტრი*

მდგომარეობის რეგისტრის უჯრედებში ბიტების დაუყენება და ბიტებისაგან უჯრედების გასუფთავება ხდება წინა ბრძანების შესრულების შედეგად მიღებულ შედეგზე დამოკიდებულებით და მათ გამოიყენებს პროცესორის ზოგიერთი ბრძანება. უჯრედებში ბიტები შეიძლებს დააყენოს ან წაშალოს პროცესორის სპეციალური ბრძანებამ (მომდევნო პარაგრაფში განვიხილავთ პროცესორის ბრძანებათა სისტემას).

ბევრ მიკროპროცესორში გამოყოფილია ***აკუმულატორად*** (ე.ი. დამგროვლებლად) წოდებული სპეციალური რეგისტრი. ამ დროს, როგორც წესი, მხოლოდ ამ რეგისტრის მეშვეობით ხდება ურთიერთზემოქმედება შეტანა/გამოტანის მოწყობილობებთან. ზოგჯერ მასში თავსდება ნებისმიერი ბრძანების შესრულების შედეგი. მაგალითად ***8086*** რეგისტრში მონაცემების ***AX*** რეგისტრი შეიძლება ჩაითვალოს თავისებურ აკუმულატორად, რადგან იგი აუცილებლად მონაწილეობს გამრავლებისა და გაყოფის ბრძანების შესრულებაში, აგრეთვე მხოლოდ მისი გავლით შეიძლება გადავგზავნოთ მონაცემები შეტანა/გამოტანის მოწყობილობაში და აგრეთვე ამ მოწყობილობიდან. სპეციალური რეგისტრ-აკუმულატორის გამოყოფა ამარტივებს პროცესორის სტრუქტურას და აჩქარებს პროცესორის შიგნით გადაგზავნებს, მაგრამ ზოგჯერ ანელებს სისტემის მუშაობას, რადგან ინფორმაციის მთელმა ნაკადმა ერთადრთ რეგისტრ-აკუმულატორში უნდა გაიაროს. ასეთი პრობლემა არ წამოიჭრება მაშინ, როდესაც ყველა რეგისტრი ურთიერთშეცვლადია.

X თავი
პროცესორების ბრძანებათა სისტემა

10.1. ზოგადი ცნობები

პროცესორის ბრძანებათა სისტემა შემავალი ბრძანებების სახეები, მათი შესასრულებელი ფუნქციები და თავისებურებები ***10.1*** ცხრილშია მოყვანილი.

ცხრილი 10.1. მიკროპროცესორის ბრძანებები

ბრძანებათა სახეები	**შესასრულებელი ფუნქციები**	**ბრძანებების თვისებურებები**
მონაცემთა გადაგზავნის ბრძანებები	ოპერანდების გადაგზავნა (გადაკოპირება)წყაროდან (Source) მიმღებში (Destination)	წყაროდ და მიმღებად გამოიყენება პროცესორის შიგა რეგისტრები, მეხსიერების ან შეტანა/გამოტანის უჯრედები; ***ალგ*** ამ დროს არ გამოიყენება
არითმეტიკული ბრძანებები	შეკრების, გამოკლების, გამრავლების, გაყოფის, 1-ით გაზრდის (ინკრემენტირების). 1-ით შემცირების (დეკრიმენტირების) ოპერაციები	ბრძანებებს სჭირდება ერთი ან ორი შესასვლელი ოპერანდი; ისინი წარმოქმნის ერთ გამოსასვლელ ოპერანდს
ლოგიკური ბრძანებები	კონიუნქციის, დიზიუნქციის, ინვერსიის, ***2***-ის მოდულით შეკრების, გაწმენდის, სხვადასხვა სახის (მარჯვნივ, მარცხნივ, არითმეტიკული, ციკლური) ძვრის ოპერაციები	- " -
გადასვლათა ბრძანებები	უზრუნველყოფს ქვეპრორამაზე გადასვლას, მისგან დაბრუნებას, პროგრამათა ფრაგმენტების გამოტოვებას, წარმოქმნის ციკლებს, განშტოებებს და ა.შ.	ყოველთვის ცვლის ბრძანებათა მთვლელის შიგთავსს; არის უპირობო ან პირობითი; იძლევა ინფორმაციის რთული ალგორითმების აგების საშუალებას

თითოეული შესრულებული ბრძანების შედეგის შესაბამისად პროცესორის მდგომარეობის რეგისტრზე (***PSW***-ზე) დაყენდება ბიტი, ან დაიცლება იგი ბიტისაგან; ***PSW***–ში ალმების შეცვლა ნებისმიერ

ბრძანებას არ შეუძლია. ისინი კონკრეტული პროცესორის თავისებურებების შესაბამისად განისაზღვრება.

სხვადასხვა პროცესორის ბრძანებების სისტემები არსებითი ურთიერთგანსხვავებულობის მიუხედავად ერთმანეთს ძალიან ჰგავს. სხვადასხვა პროცესორების ბრძანებათა რაოდენობებიც სხვადასხვაა. მაგალითად, ***MC68000*** პროცესორს აქვს ***61*** ხოლო, ***8086*** პროცესორს – ***133*** ბრძანება. თანამედროვე მძლავრი პროცესორის ბრძანებათა რაოდენობა რამდენიმე ასეულს აღწევს. არსებობს ბრძანებათა შემცირებული ნაკრების მქონე პროცესორებიც (ე.წ. ***RISC*** პროცესორები; ***Reducid Instuction Set Computer***), რომლებშიც ბრძანებათა რაოდენობის მაშემცირების ხარჯზე გაზრდილია ეფექტურობა და მუშაობის სიჩქარე.

უფრო დაწვრილებით განვიხილოთ პროცესორის ბრძანებათა ***10.1*** ცხრილში მოყვანილი ოთხივე ჯგუფის თავისებურებები.

10.2. მონაცემების გადაგზავნის ბრძანებები

ნებისმიერი პროცესორის მრძანებათა სისტემებში ძალიან მნიშვნელოვანი ადგილი უკავია მონაცემების გადაგზავნის ბრძანებებს. ისინი ასრულებს შემდეგ უმნიშვნელოვანეს ფუნქციებს: ***1)*** პროცესორის შინაგან რეგისტრებში ჩატვირთავს (ჩაწერს) შიგთავსებს; ***2)*** პროცესორის შინაგანი რეგისტრების შიგთავსებს შეინახავს მეხსიერებაში; ***3)*** მეხსიერების ერთი არედან შიგთავსს გადააკოპირებს მეორე არეში; ***4)*** შეტანა/.გამოტანის მოწყობილობებში ჩაწერს ან მისგან ამოიკითხავს ინფორმაციას.

ზოგიერთ (მაგალითად, ***T-11*** ტიპის) პროცესორში ყველა ამ ფუნქციას ოპერანდების დამისამართების სხვადასხვა მეთოდით ასრულებს ერთადერთი ***MOV*** (***MOVB***) ბრძანება.

სხვა პროცესორებში, გარდა ***MOV*** ბრძანებისა, ჩამოთვლილი ფუნქციების შესასრულებლად გამოიყენება კიდევ რამდენიმე ბრძანება. მაგალითად, რეგისტებში ჩატვირთვისათავის შეიძლება გამოყენებული იქნეს ***ჩატვირთვის ბრძანებები***, ამასთანავე სხვადასხვა რეგისტრისათვის გამოყენებული უნდა იქნეს სხვადასხვა ბრძანება (მათი აღნიშნები შედგენილია სიტყვა ***LOAD***-ის გამოყენებით). ხშირად სტეკში ჩასატვირთად ან სტეკიდან ამოსაღებად გამოიყენება სპეციალური ბრძანებები (***PUSH*** -*"შეინახე სტეკში"*, ***POP*** - *"ამოიღე სტეკიდან"*). ისინი

გადაგზავნისას ახდენს ავტოინკრემენტირებას და დეკრემენტირებას (აღნიშნული პროცესები პროცესორში ცხადი სახით რომც არ იყოს გათვალისწინებული).

ზოგიერთი (მაგალითად, ***8086***) პროცესორის ბრძანებათა სისტემაში მონაცემების სტრიქონული (ანუ მწკრივული) გადაგზავნისათვის არსებობს სპეციალური ***MOVS*** ბრძანება. იგი გადააგზავნის არა ერთი სიტყვას ან ბაიტს, არამედ დასახული რაოდენობის სიტყვებსა და ბაიტებს (***MOVSB***), ე. ი. მაგისტრალში წარმოშობს ინფორმაციის გაცვლის არა ერთ, არამედ რამდენიმე ციკლს. ამ დროს მეხსიერების მისამართი, რომელთანაც ხდება ურთიერთზემოქმედება, თითოეული მიმართვის შემდეგ იზრდება (ან მცირდება) ***1***-ით ან ***2***-ით. ე. ი. არაცხადი სახით გამოიყენება ავტოინკრემენტული ან ავტოდეკრემენტული დამისამართება.

ზოგიერთ (მაგალითად, ***8086***) პროცესორში სპეციალურად გამოიყოფა შეტანა/გამოტანის მოწყობილობებთან გაცვლის ფუნქციები. ***IN*** ბრძანება გამოიყენება შეტანა/გამოტანის მოწყობილობიდან ინფორმაციის წასაკითხად (შეტანისათვის), ხოლო ***OUT*** ბრძანება – პირიქით აღნიშნულ მოწყობილობაში ინფორმაციის ჩასაწერად (გამოსატანად). ამ შემთხვევაში ინფორმაცია გაიცვლება რეგისტრ-აკუმულატორსა და შეტანა/გამოტანის მოწყობილობას შორის. ამავე ოჯახის (***80286*** პროცესორიდან დაწყებული) უფრო მოწინავე პროცესორების ბრძანებათა სისტემებში დამატებულია სტრიქონული (მწკრივული) შეტანის ***INS*** და სტრიქონული გამოტანის ***OUTS*** ბრძანებები. ამ ბრძანებებით მონაცემების მთელი მასივი გადაიგზავნება მეხსიერებიდან შეტანა/გამოტანის მოწყობილობაში (***OUTS***) ან პირიქით – შეტანა/გამოტანის მოწყობილობიდან – მეხსიერებაში (***INS*** ბრძანებით). თითოეული მიმართვის შემდეგ მეხსიერების მისამართი იზრდება ან მცირდება (***MOVS*** ბრძანების ანალოგურად).

მონაცემების გადაგზავნის ბრძანებებს მიეკუთვნება ***ინფორმაციის გაცვლის ბრძანებებიც*** (მათი სახელწოდებები წარმოიქმნება სიტყვისაგან ***Exchange***). შეიძლება გაითვალისწინებოდეს ინფორმაციის გაცვლა შინაგან რეგისტრებს შორის, ერთი რეგისტრის ორ ნახევარს შორის (***SWAR***) ან რეგისტრსა და მეხსიერების უჯრედს შორის.

10.3. არითმეტიკული ბრძანებები

არითმეტიკული ბრძანებები ოპერანდების კოდებს ორობით ან ორობით-ათობით კოდებად განიხილავს. ისინი შემდეგ ხუთ ძირითად ჯგუფად იყოფა: *1.* ფიქსირებულ მძიმიანი არითმეტიკული (შეკრების, გამოკლების, გამრავლებისა და გაყოფის) ოპერაციების ჩამტარებელ ბრძანებები; *2.* მცურავ მძიმიანი არითმეტიკული (შეკრების, გამოკლების, გამრავლებისა და გაყოფის) ოპერაციების ჩამტარებელი ბრძანებები; *3.* ჩამოყრის (განულების) ბრძანებები; *4.* ინკრემენტისა და დეკრემენტის ბრძანებები; *5.* შედარების ბრძანებები.

მოკლედ განვიხილოთ ამ ჯგუფებში შემავალი ბრძანებები.

■ ***ფიქსირებული მძიმიანი ბრძანებები*** პროცესორის რეგისტრებში ან მეხსიერებაში არსებულ კოდებს ჩვეულებრივ ორობით რიცხვებად განიხილავს. შეკრების (***ADD***) ბრძანება გამოითვლის ორი კოდის ჯამს. გამოკლების (***SUB***) ბრძანება გამოითვლის ორი კოდის სხვაობას. გამრავლების (***MUL***) ბრძანება გამოითვლის ორი კოდის ნამრავლს (შედეგის თანრიგიანობა ორჯერ აღემატება თანამამრავლთა თანრიგიანობას). გაყოფის (***DIV***) ბრძანება ერთი კოდის მეორეზე გაყოფის შედეგად მიღებულ განაყოფს გამოითვლის. ნებისმიერ ამ ბრძანებას შეუძლია ოპერაცია ჩაატაროს როგორც ნიშნიან, ისე უნიშნო რიცხვებზე.

■ ***მცურავი მძიმიანი (წერტილიანი) ბრძანებები*** იყენებს ხარისხისა და მანტისით რიცხვების წარმოდგენის ფორმატს (ჩვეულებრივ ეს რიცხვები იკავებს მეხსიერების ორ მეზობელ უჯრედს). თანამედროვე მძლავრ პროცესორებში მცურავი წერტილიანი ბრძანებების ნაკრებში ოთხი არითმეტიკული მოქმედების შესასრულებელი ბრძანებების გარდა შედის სხვა ბრძანებებიც, რომლითაც გამოითვლება ტრიგონომეტრიული, ლოგარითმული, აგრეთვე ბგერისა და გამოსახულების დასამუშავებლად საჭირო სხვა რთული ფუნქციები.

■ ***გაწმენდის (CLR) ბრძანებები*** განკუთვნილია რეგისტრში ან მეხსიერების უჯრედში ნულოვანი კოდის ჩასაწერად. ეს ბრძანებები შეიძლება შეიცვალოს ნულოვანი კოდის გადაგზავნის ბრძანებებით, მაგრამ ჩამოგდების სპეციალური ბრძანებები ჩვეულებრივ გადაგზავნის ბრძანებებზე უფრო სწრაფად მუშაობს. ჩამოგდების ბრძანებებს

ხშირად ლოგიკური ბრძანებების ჯგუფს მიაკუთვნებენ და აღნიშნული ბრძანებების განხილვისას კიდევ ერთხელ დავუბრუნდებით მათ.

- ***ინკრემენტის*** (ერთიით გაზრდის, ***INC***) და ***დეკრემენტის*** (ერთით შემცირების, ***DEC***) ***ბრძანებებიც*** ძალიან მოსახერხებელია. ისინი შეიძლება შეიცვალოს ერთის მიმატების ან გამოკლების ბრძანებებით, მაგრამ ამ უკანასკნელ ოპერაციებზე სწრაფად სრულდება ინკრემენტისა და დეკრემენტის ბრძანებები. ისინი მოითხოვს ერთადერთ შესასვლელ ოპერანდს, რომელიც იმავდროულად გამოსასვლელი ოპერანდიცაა.
- ***შედარების*** (***CMP***) ***ბრძანება*** ორი შესასვლელი ოპერანდის შესადარებლად გამოიყენება. არსებითად იგი გამოითვლის ამ ორი ოპერანდის სხვაობას, ოღონდ გამოსასვლელ ოპერანდის წარმოქმნის ნაცვლად გამოკლების შედეგზე დამოკიდებულებით პროცესორის მდგომარეობის (***PSW***) რეგისტრში არსებულ გარკვეულ ბიტებს. შედარების ბრძანების მომდევნო (ჩვეულებრივ, ***გადასვლის***) ***ბრძანება*** გაანალიზებს პროცესორის მდგომარეობის რეგისტრში არსებულ ბიტებს და მათ მნიშვნელობაზე დამოკიდებულებით შეასრულებს გარკვეულ მოქმედებას (***გადასვლის ბრძანებებზე §10.5***-ში ვისაუბრებთ). ზოგიერთ (მაგალითად, ***8086*** და მასთან შეთავსებად) პროცესორში გამოიყენება მეხსიერებაში არსებული ოპერანდების ორი მიმდევრობის მწკრივულად შედარების ბრძანებები.

10.4. ლოგიკური ბრძანებები

ლოგიკური ბრძანებები ოპერანდებზე ასრულებს ლოგიკურ (ბიტურ) ოპერაციებს, ე. ი. ოპერანდების კოდებს განიხილავს არა ერთიან რიცხვად, არამედ ცალკეული ბიტების ნაკრებად. ამით განსხვავდება ისინი არითმეტიკული ოპერაციებისაგან. ლოგიკური ბრძანებები შემდეგ ძირითად ოპერაციებს ასრულებს: ***1)*** ლოგიკურ ***და*** ოპერაციას, ლოგიკურ ***ან*** ოპერაციას, ***2***-ის მოდულით შეკრების ოპერაციას; ***2)*** ლოგიკურ, არითმეტიკულ და ციკლურ ძვრებს; ***3)*** ბიტებისა და ოპერანდების შემოწმებას; ***4)*** პროცესორის მდგომარეობის (***PSW***) რეგისტრში ბიტების (ალმების) დაყენებასა და ჩამოგდებას.

ლოგიკური ოპერაციების ბრძანებები საშუალებას იძლევა ორი შესასვლელი ოპერანდისაგან ცალკეული ბიტების სახით გამოითვალოს ძირითადი ლოგიკური ფუნქციები.

და (***AND***) ოპერაცია გამოიყენება მოცემული ბიტების იძულებით ჩამოსაყრელად. ამისათვის ერთ ოპერანდად გამოიყენება ***ნიღბის კოდი***, რომლის იმ თანრიგებში ბიტებად, რომელთა ჩამოყრა მოითხოვება, ბიტებად ნულებია დაყენებული (იხ. ***ნახ. 2.19,ბ***).

ან (***OR***) ოპერაცია გამოიყენება მოცემული ბიტების იძულებით დასაყენებლად. ამისათვის ერთ ოპერანდად გამოიყენება ***ნიღბის კოდი,*** რომლის იმ თანრიგებში, რომლებშიც ერთიანის დაყენება მოითხოვება, ბიტებად ერთიანებია დაყენებული (იხ. ***ნახ. 2.19,დ***). .

ორის მოდულით შეკრების (***XOR***) ოპერაცია გამოიყენება მოცემული ბიტების ინვერსირებისათვის. ამისათვის ერთ ოპერანდად გამოიყენება ***ნიღბის კოდი***, რომლის იმ თანრიგებში, რომელთა ინვერსირება მოითხოვება, ბიტებად ერთიანებია დაყენებული (იხ. ***ნახ. 2.19,ვ***).

რეგისტრებში ორობითი კოდების ძვრის ოპერაციები დატალურად გვაქვს განხილული ***§2.11***-ში.

10.5. გადასვლათა ბრძანებები

გადასვლათა ბრძანებები განკუთვნილია ყველა შესაძლო ციკლების, განშტოებების, ქვეპროგრამების გამოძახებებისა და ა. შ. ორგანიზებისათვის, ე. ი. ისინი არღვევს ***ძირითადი პროგრამის*** მიმდევრობით შესრულების რეჟიმს. აღნიშნული ბრძანებები რეგისტრმთვლელში ჩაწერს ახალ მნიშვნელობას და ამით აიძულებს პროცესორს გადავიდეს არა მომდევნო, არამედ მეხსიერებაში არსებული ნებისმიერ სხვა ბრძანებაზე. არსებობს ძირითადი ბრძანების იმ წერტილზე დაბრუნე-ბის ბრძანებებიც, რომლიდანაც მოხდა გადასვლა. მომავალში თუ გვაქვს განზრახული ძირითად პროგრამზე დაბრუნება, მაშინ ამ პრო-გრამიდან ყოველი გადახვევის წინ პროცესორის მიმდინარე პარამე-ტრები სტეკში უნდა იქნეს შენახული; ასეთი განზრახვის არარსებო-ბისას სტეკში პარამეტრების შენახვა საჭირო არ არის.

განასხვავებენ გადასვლის შემდეგი ორი სახის ბრძანებებს: **1)** უპირობო გადასვლის ბრძანებებს; ***2)*** პირობითი გადასვლის ბრძანებებს.

ამ ბრძანებათა სახელწოდებებში გამოიყენება სიტყვები ***Branch*** (განშტოება) და ***Jump*** (ნახტომი). გავეცნოთ ორივე მათგანს.

■ ***უპირობო გადასვლათა ბრძანებები*** ახალ მისამართზე უპირობო გადასვლას მოითხოვს. მათი საშუალებით მითითებული სიდიდით ხდება მიმდინარე პროგრამაში წინ ან უკან წანაცვლება, ან მეხსიერების მითითებულ მისამართზე გადასვლა. წანაცვლების სიდიდე ან მისამართის ახალი მნიშვნელობა შესასვლელი ოპერანდის სახითაა წარმოდგენილი.

■ ***პირობით გადასვლათა ბრძანებები*** გადასვლას იწვევს არა ყოველთვის, არამედ მხოლოდ მოცემული პირობის შესრულებისას. ასეთ პირობებად გამოიყენება პროცესორის მდგომარეობის ***PSW*** რეგისტრში ალმების მდგომარეობები, ე. ი. ***გადასვლის პირობაა*** ალმის მნიშვნელობის შემცვლელი წინა ოპერაციის შედეგი. სხვადასხვა პროცესორებში ამ პირობათა რაოდენობა ***4***-დან ***16***-მდეა. მათი რამდენიმე მაგალითია: ***1***) «გადასვლა, ***0***-ზე ტოლობისას»; ***2)*** «გადასვლა ***1***-ზე ტოლობისას»; ***3***) «გადასვლა გადავსებისას»; ***4)*** «გადასვლა გადავსება არარსებობისას»; ***5)*** «გადასვლა ***0***-ზე მეტობისას»; ***6)*** «გადასვლა ***0***-ზე ნაკლებობის ან ტოლის დროს» და ა.შ.

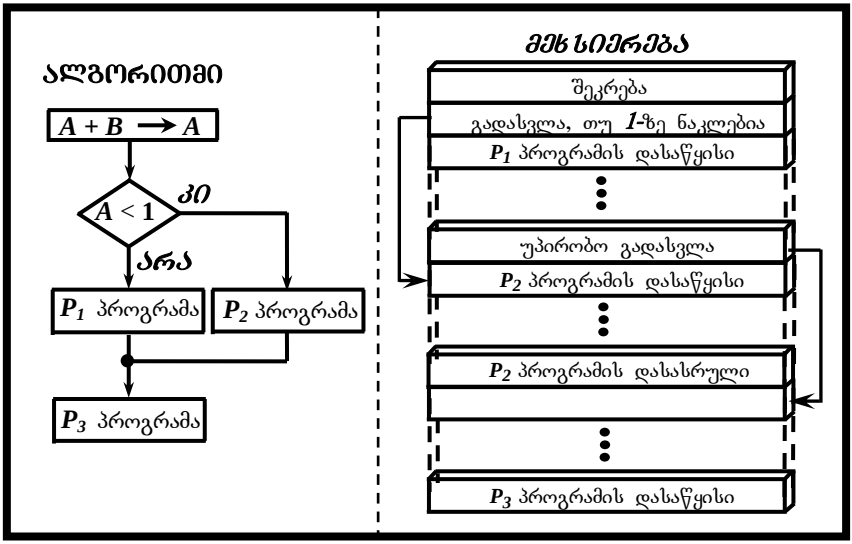
გადასვლის პირობის შესრულებისას ბრძანებათა რეგისტრმთვლელში ჩაიტვირთება ახალი მნიშვნელობის ბრძანება, საწინააღმდეგო შემთხვევაში აღნიშნული რეგისტრის შიგთავსი უბრალოდ გაიზრდება და პროცესორი დაიწყებს მომდევნო ბრძანების შესრულებას.

გადასვლის პირობის შესამოწმებლად პირობითი გადასვლის ბრძანების (ან, შესაძლებელია, რამდენიმე ბრძანების) წინ სპეციალურად სრულდება შედარების ***CMP*** ბრძანება. მაგრამ ალმები შეიძლება ***დააყენოს*** (***1***-ის ტოლი გახადოს), მაგალითად, მონაცემების გადამგზავნმა, არითმეტიკულმა ან ლოგიკურმა ნებისმიერმა ბრძანებამ.

პირობითი და უპირობო გადასვლების რამდენიმე ბრძანების ერთობლივად გამოყენება პროცესორს საშუალებას აძლევს განაშტოვოს ნებისმიერი სირთულის ალგორითმი. მაგალითისათვის **10.1** ნახაზზე ნაჩვენებია ***ორად განშტოებული***, ხოლო **10.3** ნახაზზე - ***სამად განშტოებული პროგრამა***, რომლებიც ბოლოში ხელახლა ერთიანდება.

მიმდინარე პროგრამის დასრულების შემდეგ გადასვლის წერტილში ხელახლა დაბრუნების უზრუნველმყოფი ***გადასვლის ბრძანებები***

დამხმარე პროგრამების – ე.წ. ***ქვეპროგრამების*** შესასრულებლად გამოიყენება. მათ ***ქვეპროგრამების გამოძახების ბრძანებებსაც*** უწოდებენ და ***CALL*** მნემოკოდით აღნიშნავენ. ქვეპროგრამების გამოყენება ამარტივებს და უფრო ლოგიკურს ხდის ძირითად ბრძანებას. ამით ეს უკანასკნელი მოქნილი, ადვილად დასაწერი და ადვილადვე გასამართი ხდება. ამავე დროს ქვეპროგრამების ზრდის პროგრამის შესრულების ხანგრძლივობას და ამიტომ მათი გამოყენების დროს საჭიროა დავიცვათ ზომიერება.

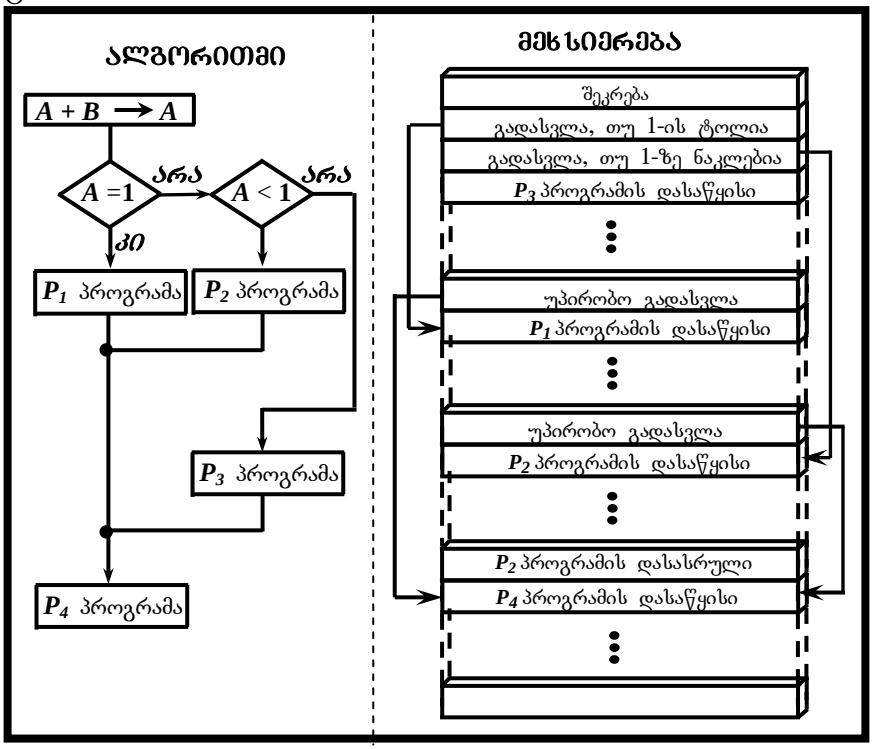


ნახ. 10.1. *ორად განშტოებული პროგრამის რეალიზება*

დაბრუნების გამთვალისწინებელი გადასვლის ყველა ბრძანება ასრულებს ***უპირობო გადასვლას*** (ისინი არ ამოწმებს არავითარ ალამს). სამაგიეროდ მათ სჭირდება ერთი შესასვლელი ოპერანდი, რომელსაც შეუძლია მიუთითოს როგორც ახალი მისამართის აბსოლუტური მნიშვნელობა, ასევე მისამართის მიმდინარე მნიშვნელობისათვის მისამატებელი წანაცვლება. ***ბრძანებათა მთვლელის*** მიმდინარე მნიშვნელობა (***მიმდინარე მისამართი***) გადასვლის შესრულების წინ შეინახება სტეკში.

ქვეპროგრამის გამოძახების წერტილში (გადასვლის წერტილში) უკუდაბრუნებისათვის გამოიყენება დაბრუნების სპეციალური (***RET*** ან

RTS) ბრძანება. იგი სტეკიდან ამოიტანს გადასვლის ბრძანების მისამართის მნიშვნელობას და მას ჩაწერს ბრძანებათა მთვლელ რეგისტრში.



ნახ. 10.2. *სამად განშტოებული პროგრამის რეალიზება*

დაბრუნების უნარის მქონე გადასვლის ბრძანებებს შორის განსაკუთრებული ადგილი უკავია ე. წ. ***შეწყვეტათა*** (***INT***) ***ბრძანებებს***. ისინი შესასვლელ ოპერანდად მოითხოვს შეწყვეტის ნომერს (ვექტორის მისამართს). მათი მომსახურება ზუსტად ისევე ხდება, როგორც აპარატურული შეწყვეტების დროს, ე. ი. მოცემული გადასვლისათვის პროცესორი მიმართავს შეწყვეტათა ვექტორების ცხრილის, და შეწყვეტის ნომრის მიხედვით მისგან მიიღებს მეხსიერების იმ უჯრედის მისამართს, რომელზეც უნდა მოხდეს გადასვლა. შეწყვეტის გამოძახების მისამართი და პროცესორის მდგომარეობის (***PSW***) რეგისტრის შიგთავსი შეინახება სტეკში. ***PSW***-ს შენახვაა დაბრუნებით გადას-

ვლის ბრძანებებისაგან შეწყვეტების ბრძანებების მნიშვნელოვანი განმასხვავებლი ნიშანი.

გამოსაყენებლად შეწყვეტების ბრძანებები ხშირად დაბრუნების უნარის მქონე ბრძანებებზე უფრო მოსახერხებელია. ***შეწყვეტების ვექტორთა ცხრილის*** შედგენის შემდეგ მას შეიძლება მივმართოთ საჭიროების ყოველი წარმოშობის დროს. ***შეწყვეტის ნომერი*** შეესაბამება ქვეპროგრამის ნომერს, ე. ი. ქვეპროგრამის მიერ შესასრულებელი ფუნქციის ნომერს. ამიტომ შეწყვეტების ბრძანებებს უფრო ხშირად იყენებენ პროცესორთა ბრძანებების სისტემებში, ვიდრე დაბრუნების უნარის მქონე გადასვლების ჩვეულებრივ ბრძანებებს.

შეწყვეტის ბრძანებით გამოძახებული ქვეპროგრამიდან დასაბრუნებლად გამოიყენება შეწყვეტიდან დაბრუნების (***IRET*** ან ***RTI***) ბრძანება. ამ ბრძანებას სტეკიდან ამოაქვს მასში შენახული როგორც ბრძანებათა მთვლელის, ისე პროცესორის მდგომარეობის ***PSW*** რეგისტრის შიგთავსი.

ჩვენ ზემოთ განვიხილეთ პროცესორებში ყველაზე ხშირად გამოყენებადი ბრძანებები. რა თქმა უნდა, კონკრეტულ პროცესორებს ხშირად აქვს მრავალი სხვა ბრძანებაც. მიგვაჩნია, რომ მათი შესწავლა აღნიშნული პროცესორების გამოყენების პროცესში არის უფროა მოსახერხებელი.

10.6. პროცესორის სწრაფმოქმედება

მთელი მიკროპროცესორული სისტემის მუშაობის ეფექტურობის განმსაზღვრელი ერთ-ერთი უმნიშვნელოვანესი მახასიათებელია ***პროცესორის სწრაფმოქმედება.*** იგი იმდენად მრავალ ფაქტორზეა დამოკიდებული, რომ სხვადასხვა ფირმის მიერ დამუშავებული ან სხვადასხვა დანიშნულების პროცესორის თუნდაც ერთ ოჯახში შემავალი პროცესორების ერთმანეთთან შედარებაც, ძალიან რთულია.

გამოვყოთ პროცესორის სწრაფმოქმედებაზე გავლენის მომხდენი უმნიშვნელოვანესი ფაქტორები.

ტაქტური სიხშირე. სწრაფმოქმედება, უპირველეს ყოვლისა, პროცესორის ტაქტურ სიხშირეზეა დამოკიდებული. პროცესორის შიგნით ყველა ოპერაცია სრულდება სინქრონულად, ტაქტირდება ერთიანი ტაქტური სიგნალით. რაც უფრო მაღალია ტაქტური სიხშირე,

მით უფრო სწრაფად მუშაობს პროცესორი; გარკვეული პროცესორის ტაქტური სიხშირის ორჯერ გაზრდა ამდენჯერვე ამცირებს ამ პროცესორის მიერ ბრძანებების შესრულებაზე დახარჯულ დროს.

■ ***ტაქტების რაოდენობა.*** სხვადასხვა პროცესორე ერთსა და იმავე ბრძანებების შესასრულებლად სხვადასხვა რაოდენობის ტაქტები სჭირდებათ; ბრძანების შესასრულებლად საჭირო ტაქტების რაოდენობა შეძლება ერთიდან ათამდე და, შესაძლებელია, ასამდეც იცვლებოდეს. არსებობს ისეთი პროცესორებიც, რომლებსაც მიკრო ოპერაციების დაპარალელების ხარჯზე ბრძანების შესრულებას ტაქტის გარკვეულ ნაწილშიც ასწრებს.

■ ***ბრძანებების სირთულე და დამისამართების მეთოდები.*** ბრძანების შესრულებაზე დახარჯული ტაქტების რაოდენობა დამოკიდებულია ამ ბრძანების სირთულესა და დამისამართების მეთოდებზე. მაგალითად, ყველაზე სწრაფად (მცირე რაოდენობის ტაქტებში) სრულდება შინაგან რეგისტრებში მონაცემთა გადაგზავნის ბრძანებები, ყველაზე ნელა კი (მეტი რაოდენობის ტაქტებში) - მეხსიერებაში შენახულ მცურავ მძიმიან რიცხვებზე რთული არიმეტიკული მოქმედებათა ჩატარების ბრძანებები.

ცხრილი 10.1. ზოგიერთი პროცესორის პარამეტრები

პროცესორი	*8085*	*8086*	*6800*	*68000*
ფ ი რ მ ა	***Intel***		***Motorola***	
თანრიგიანობა	***8***	***16***	***8***	***16***
ბრძანებების რაოდენობა	***80***	***133***	***72***	***61***
ტაქტური სიხშირე, მგჰც	***3***	***5***	***1***	***8***
მოკლე ოპერაციების შესრულების დრო, მკწმ	***1,3***	***0,4***	***2,0***	***0,3***

პროცესორების ***მწარმოებლურობის*** რაოდენობრივი შეფასებისათვის ზომის ერთეულად ადრე გამოიყენებოდა ***MIPS*** (***Mega Instuksion Per Sekond)***, რომელიც გვიჩვენებს, წამში თუ რამდენი მილიონი ოპერაციის შესრულება შეუძლია მათ. ბუნებრივია, რომ მიკროპროცესორების დამამზადებლები ცდილობდნენ, გამოეყენებინათ ყველაზე სწრაფი ბრძანებები. ასეთი მიდგომა არც თუ ისეთი დიდად მოსახერხებე-

ლი აღმოჩნდა. მოგვიანებით მცურავ წერტილიან რიცხვებზე მოქმედებების ჩატარების დროს მწარმოებლურობის შესაფასებლად შემოღებული იქმა ზომის ახალი ერთეული – ***FLOPS*** (***FL****oating point* ***O****peration* ***P****er* ***S****econd*), რომელიც გვიჩვენებს მცურავ მძიმიან რიცხვებზე წამში შესრულებული ოპერაციების რაოდენობას. იგი თავისი არსით იგი ვიწროსპეციალურია, რადგან ზოგიერთ სისტემაში საერთოდ არ გამოიყენება მცურავი მძიმიანი რიცხვები.

პროცესორის სწრაფმოქმედების მაჩვენებელი სხვა ანალოგური მაჩვენებელია მოკლე (სწრაფი) ოპერაციების შესრულების დრო. მაგალითისათვის, ***10.1*** ცხრილში წარმოდგენილია რამდენიმე ***8-*** და ***16-***თანრიგიანი პროცესორების სწრაფმოქმედების მაჩვენებლები. დღეისათვის ***MIPS***-ის მსგავსად ეს მაჩვენებელიც პრაქტიკულად არ გამოიყენება.

ბრძანებების შესრულების დრო სწრაფმოქმედების განმსაზღვრელი მნიშვნელოვანი, მაგრამ არა ერთადერთი ფაქტორია. დიდი მნიშვნელობა აქვს პროცესორის ***ბრძანებათა სისტემის სტრუქტურასაც.*** მაგალითად, ზოგიერთი პროცესორისათვის გარკვეული ოპერაციის შესასრულებლად ერთი ბრძანებაა საკმარისი, მაგრამ სხვა პროცესორს ამისათვის რამდენიმე ბრძანება სჭირდება. ზოგიერთ პროცესორებს აქვს ერთი ტიპის, ხოლო მეორეს – სხვა ტიპის ამოცანების სწრაფად გადამწყვეტი ***ბრძანებათა სისტემა.*** მნიშვნელოვანია აგრეთვე ის, თუ დამისამართების როგორი სისტემაა გამოყენებული მოცემულ პროცესორში, არსებობს თუ არა მეხსიერების სეგმენტირება, შეუძლია თუ არა მას შეტანა/გამოტანის მოწყობილობებთან ურთიერთობა და ა.შ.

სისტემის სწრაფმოქმედებაზე დიდ გავლენას ისიც ახდენს, თუ როგორ ურთიერთობს პროცესორი ბრძანებებისა და მონაცემების მეხსიერებასთან, აქვს თუ არა მას ერთმანეთს შეუითავსოს უკვე ამოკრებილი ბრძანებების შესრულებისა და მეხსიერებიდან ახალი ბრძანებების ამოკრების პროცესები.

ზოგადად სისტემის სწრაფმოქმედებას პროცესორის თანრიგიანობაც განსაზღვრავს. ***8***-თანრიგიანი პროცესორი ***16***-თანრიგიან პროცესორზე უფრო ნელა გადააგზავნის და დაამუშავებს მონაცემთა დიდ მასივებს, ***16***-თანრიგიანი პროცესორი კი ***65536***-ზე უფრო დიდ ციფრებს ***32***-თანრიგიან პროცესორზე უფრო ნელა დაამუშავებს.

მაღალი სირთულის ამოცანების გადაწყვეტისას მიკროპროცესორული სისტემის სწრაფმოქმედება ***სისტემური მეხსიერების საერთო მოცულობაზეა*** დამოკიდებული. ამას განაპირობებს ის გარემოება, რომ მცირე მოცულობის სისტემური მეხსიერების დროს მიკროპროცესორულ სისტემა იძულებულია მონაცემები გარე მეხსიერებაში (მაგალითად, მაგნიტურ დისკზე) შეინახოს, რაც მნიშვნელოვნად (რამდენიმე ხარისხით) ანელებს მუშაობას. პროცესორისათვის ასევე მნიშვნელოვანია სამისამართო სალტის თანრიგიანობაც.

ზემოთ მოყვანილი ფაქტების გამო შეიძლება დავასკვნათ, რომ პირობითია პროცესორთა მწარმოებლურობის მაჩვენებლები. ისინი მხოლოდ ირიბად ახასიათებს მოცემული პროცესორის ბაზაზე აგებული სისტემის სწრაფმოქმედებას. მიუხედავად ამისა, პროცესორების მწარმოებელი ზოგიერთი ფირმა მაინც გვაწვდის საკუთარი პროდუქტების რაოდენობრივ მაჩვენებლებს. ისინი განსაზღვრულია სხვადასხვა ბრძანებების ამა თუ იმ თანაფარდობით შემცველი სპეციალური სატესტო პროგრამების შესრულების მაგალითზე.

ცხრილი 10.2. პროცესორთა მწარმოებლურობის ***iCMOP*** და და ***iCMOP Index 2.0*** ინდექსები

პროცესორი	*iCMOP Index*	პროცესორი	*iCMOP Index 2.0*
i486SX-25	*100*	*Pentium-100*	*90*
i386DX-33	*56*	*Pentium-120*	*100*
i486DX-33	*136*	*Pentium-150*	*114*
i486DX2-66	*297*	*Pentium-200*	*142*
i486DX4-100	*435*	*Pentium MMX-160*	*160*
Pentium-60	*510*	*Pentium MMX-233*	*203*
Pentium-100	*815*	*Pentium Pro-200*	*220*
Pentium-133	*1110*	*Pentium II-266*	*303*

მაგალითად ფირმა ***Intel***–მა ***1992*** წელს ***32***-თანრიგიანი მიკროპროცესორების მწარმოებლურობის ურთიერთშედარებისათვის ზომის საკუთარი ერთეული ***iCMOP Index*** (***I****ntel* ***CO****mparative* ***M****icroprocessor* ***P****erformance*) შემოგვთავაზა. ამ მაჩვენებლის გამოსათვლელად გამოიყენება მთელი და მცურავი წერტილის მქონე რიცხვების, აგრეთვე

გრაფიკისა და ვიდეოს დამამუშავებელი ***16***- და ***32***-თანრიგიანი ბრძანებების ნაკრები, ხოლო საბაზისო პროცესორად შერჩეულია ***i486SX-25*** ტიპის პროცესორი, რომლის ინდექსად მიღებულია რიცხვი ***100.*** ცხრილ ***10.2***-ის მარცხენა ნაწილში მოყვანილია ფირმის ზოგიერთი პროცესორისათვის გამოთვლილი ***iCMOP*** ინდექსები. როგორც ცხრილიდან ჩანს, უფრო განვითარებული არქიტექტურის გამო ***486*** ოჯახის პროცესოცესორები ყოველთვის უფრო სწრაფია ***386*** ოჯახის პროცესორებზე, ხოლო ნებისმიერი ***Pentium***–ი სწრაფია **486** ოჯახის ნებისმიერ პროცესორზე. ***ტაქტური სიხშირე***, რომელიც ცხრილში პროცესორის სახელწოდებისთან დეფისითაა დაკავ-შირებული, მწარმოებლურობას მხოლოდ ერთი ოჯახის ფარგლებში განსაზღვრავს.

ფირმა ***Intel***–მა ***1996*** წელს შემოგვთავაზა მეორე მაჩვენებელი - ***iCMOP Index 2.0***. რომლის გამოსათვლელად უკვე არ გამოიყენება ***16***–თანრიგიანი ბრძანებები; სამაგიეროდ შეტანილია მულტიმედიური ტესტი, ხოლო ***100***-ს ტოლი ინდექსიან საბაზისო პროცესორად ***i486SX-25***–ს ნაცვლად აღებულია პროცესორი ***Pentium-120.***

ფირმა ***Intel*** -ის მიერ გამოშვებული ზოგიერთი პროცესორისათვის გამოთვლილი ***iCMOP Index 2.0*** მაჩვენებელი ***10.2*** ცხრილის მარჯვენა ნაწილშია მოყვანილი.

ზემოთ მოყვანილი და მსგავსი მაჩვენებლების ღირებულება არ არის ძალიან მაღალი. კონკრეტული კომპიუტერისა და სხვადასხვა პროცესორისათვის მაჩვენებელთა სიდიდემ შეიძლება მოგვაწოდოს სრულიად ობიექტური მონაცემები, რომელთა საშუალებითაც შეგვეძლება, მაგალითად, შევაფასოთ უფრო მძლავრი პროცესორით მოცემული პროცესორის შეცვლის მიზნშეწონილება, მაგრამ გასაშუალებული ***iCMOP*** მაჩვენებელით ზუსტად ვერ დავადგენთ სხვადასხვა ტიპის გამოყენებაზე ორიენტირებული სხვადასხვა ამოცანის გადაწყვეტისას, როგორ იმუშავებს მოცემული პროცესორი.

XI თავი
ზოგადი ცნობები მიკროკონტროლერების შესახებ

11.1. მიკროკონტროლერის რაობა

მიკროკონტროლერი წარმოადგენს სხვადასხვა ელექტრონული მოწყობილობების მართვისათვის განკუთვნილ სპეციალურ მიკროსქემას. მიკროკონტროლერის «დაიბადა» ***1971*** წელს, როდესაც ქვეყანას მოევლინა მისი უდიდებულესობა – საერთო დანიშნულების მიკროპროცესორი.

მიკროკონტროლერების შემქმნელებმა ხორცი შეასხეს მახვილგონივრულ იდეას – ერთ კორპუსში გააერთიანეს პროცესორი, მეხსიერება და პერიფერიული მოწყობილობები. მიუხედავად იმისა, რომ უკანასკნელ ხანს მიკროკონტროლერების ყოველწლიური წარმოება მრავალჯერ აღემატება მიკროპროცესორების წარმოებას, მათზე მოთხოვნილება მაინც არ მცირდება.

მიკროკონტროლერებს უშვებს ათობით კომპანია. ამასთანავე ისინი აწარმოებს არა მარტო თანამედროვე ***32***-თანრიგიან, არამედ ***16-*** და ***8***-ბიტურ (***i8051***-ის ანალოგურ) მიკროკონტროლერებსაც. მათი სიმრავლე დაყოფილია ცალკეულ ოჯახებად. თითოეული ოჯახის შიგნით შეიძლება შევხვდეთ სრულიად ერთნაირ მოდელს, რომლებიც ერთმანეთისაგან განსხვავდება ცენტრალური პროცესორის მუშაობის სიჩქარითა და მეხსიერების მოცულობით. საქმე ისაა, რომ მიკროკონტროლერები უპირატესად გამოიყენება ჩაშენებულ სისტემებში, სათამაშოებში, ჩარხებში, მასობრივ სამეცნიერო ტექნიკაში, სარკინიგზო ავტომატიკის მმართველ მოწყობილობებში, თანამედროვე ავტომობილებში, საოჯახო ავტომატიკაში – იქ, სადაც საჭიროა არა პროცესორის სიმძლავრე, არამედ უფრო საჭიროა ფასსა და საკმარის ფუნქციურობას შორის წონასწორობის დაცვა. სწორედ ამიტომაა დღემდე მოთხოვნადი ძველი (***8***-თანრიგიანი) ტიპის მიკროკონტროლერები; მათ ავტომატურად კარების გაღებისა და გაზონების მორწყვიდან დაწყებული «გონიერი სახლის» სისტემაში ინტეგრაციით დამთავრებული, ბევრი რამ შეუძლია. ამასთანავე ბაზარზე მრავლადაა წამში ასობით მილიონ ოპერაციის ჩამტარებელი და მრავალფეროვანი პერიფერიული მოწყობილობებით «კბილებამდე აღჭურვილი» უფრო მძლავრი მიკროკონტროლერებიც. მათთვის შესატყვისი ამოცანები არსებობს. ამგვარად, კონსტრუქტორმა ჯერ უნდა შეაფასოს

ამოცანა და მხოლოდ ამის შემდეგ უნდა შეირჩიოს მისთვის მიკროკონტროლერი.

დღეისათვის არსებობს ***i8051*** მიკროკონტროლერთან შეთავსებადი ***200***-ზე მეტი მოდიფიკაციისა და უამრავი სხვა ტიპის მიკროკონტროლერები. კონსტრუქტორებს შორის დიდი პოპულარობით სარგებლობს ფირმა ***Microcip Technology***-ის მიერ დამუშავებული ***PIC*** და ***Atmel***-ის მიერ დამუშავებული ***AVR*** ტიპის ***8***-თანრიგიანი, ***TI*** ფირმის მიერ დამუშავებული ***MSP430*** ტიპის ***16***-თანრიგიანი, ***ARM Limited*** ფირმის მიერ დამუშავებული ***ARM*** ტიპის ***32***-თანრიგიანი და ა.შ. მიკროკონტროლერები.

მიკროკონტროლერი ერთდროულად არის როგორც პროგრამულად მართვადი რთული მოწყობილობა, ისე ელექტრონული ხელსაწყოც (მიკროსქემაც), რის გამოც მის შესაფასებლად დიდი რაოდენობის პარამეტრები გამოიყენება. სახელწოდებაში არსებული წინსართი «***მიკრო***» აღნიშნავს, რომ იგი მზადდება მიკროელექტრონული ტექნოლოგიით.

მუშაობის პროცესში მიკროკონტროლერი მეხსიერებიდან ან შეტანის პორტიდან ამოიკითხავს ბრძანებებს და ასრულებს მათ. ეს ნიშნავს, რომ თითოეული ბრძანება განისაზღვრება მიკროკონტროლერის არქიტექტურაში ჩადებული ბრძანებათა სისტემით და ***ბრძანების კოდის შესრულება*** მიკროსქემის შიგა ელემენტების მიერ გარკვეული ***მიკროოპერაციების ჩატარებით*** გამოიხატება.

მიკროკონტროლერები სხვადასხვა ელექტრონული და ელექტრული მოწყობილობების მოქნილად მართვის საშუალებას გვაძლევს. ზოგიერთი მოდელის მიკროკონტროლერები იმდენად მძლავრია, რომ შეუძლია გადართოს რელე (მაგალითად, ნაძვის ხის გირლანდაში).

მიკროკონტროლერები, როგორც წესი, მუშაობს არა განცალკევებულად, არამედ ჩაირჩილება სქემაში, რომელთანაც მის გარდა მიერთებულია ეკრანი, საკლავიატურო შესასვლელები, სხვადასხვა გადამწოდი და ა.შ.

მიკროკონტროლერებისათვის განკუთვნილმა პროგრამებმა შეიძლება იმათი ყურადღება მიიპყროს, რომლებიც ყოველი ბიტის მომჭირნედ გამოყენებას ანიჭებს უპირატესობას, რადგან მიკროკონტროლერებში არსებული მეხსიერების მოცულობა ***2***-დან ***128*** კილობაიტის ფარგლებშია. ძალიან მცირე მოცულობის მეხსიერების დროს გვიხდება პროგრამა ***ასემბლერის*** ან ***ფორტის*** ენებზე დავწეროთ. აღნიშნული მოცულობის სიდიდე «ოდნავი ამოსუნთქვის» საშუალებას თუ მოგვცემს, მაშინ შეგვეძლება ***ბეისიკის ან პასკალის სპეციალური ვერსიებიც*** გამოვიყენოთ, თუმცა უმჯობესია ორიენტაცია ***C*** ენაზე ავიღოთ. საბოლოოდ დაპროგრამებამდე მიკროკონტროლერი უნდა «გავტესტესტოთ» სპეციალურ აპარატურული ან პროგრამული ემულატორებში.

დაისმის კითხვა: ***მიკროპროცესორი და მიკროკონტროლერი ერთი და იმავე მოწყობილობის სხვადასხვა დასახელებებია თუ განსხვავებული მოწყობილობები?***

მიკროპროცესორი ნებისმიერი გამომთვლელი მანქანის ცენტრალური მოწყობილობაა, რომელიც დამზადებულია ინტეგრალური ტექნოლოგიით. სახელწოდებაც მიგვითითებს იმაზე, რომ სწორედ მასში მიმდინარეობს გამოთვლითი პროცესები. არცთუ თანამედროვე და მძლავრ გამომთვლელ სისტემად გარდაქმნისათვის მას უნდა დავუმატოთ გარე მოწყობილობები, უპირველეს ყოვლისა, ოპერატიული მეხსიერება და ინფორმაციის შეტანა/გამოტანის პორტები.

რაც შეეხება ***მიკროკონტროლერს,*** მას საკუთარ სტრუქტურაშივე აქვს როგორც პროცესორი, ასევე ***RAM, ROM*** და პერიფერიულ მოწყობილობათა მთელი ნაკრები; ე. ი. იგი უკვე «ფეხზე მყარად დამდგარი» გამომთვლელი სისტემაა. თავდაპირველად მას ერთკრისტალურ გამომთვლელ მანქანასაც უწოდებდნენ, ოღონდ მოგვიანებით კონტროლერად «გადანათლეს»; ამით ხაზი გაუსვეს იმ ფაქტს, რომ მიკროკონტროლერისათვის მნიშვნელოვანია არა გამოთვლითი ფუნქციის შესრულება, არამედ ***მართვის პროცესში*** მის მიერ შეტანილი წვლილი (ინგლ. ტერმინი ***Control*** ნიშნავს მართვას)

11.2. მიკროკონტროლერების კლასიფიკაცია და სტრუქტურა

დღეისათვის არსებული მიკროკონტროლერები იყოფა შემდეგ სამ ძირითად ჯგუფად: ***1)*** ჩასაშენებელ ***8***-თანრიგიან მიკროკონტროლერებად; ***2)*** ***6***- და ***32***-თანრიგიან მიკროკონტროლერებად; ***3***) ციფრულ სასიგნალო (***DSP***) მიკროკონტროლერებად.

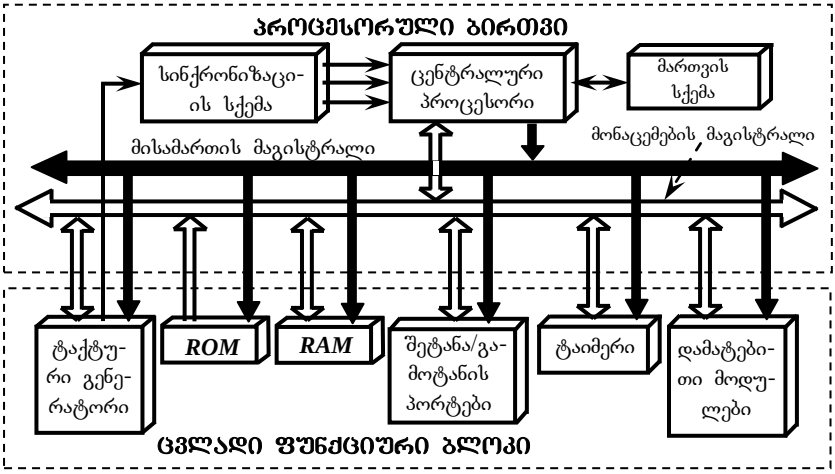
მიკროკონტროლერთა მრავალრიცხოვან ოჯახში ყველაზე მეტად გავრცელებულია ***8***-თანრიგიანი მიკროკონტროლერები. განვითარების პროცესში მათ გაიარეს შედარებით სუსტი პერიფერიის მქონე კონტროლერებიდან დროის რეალურ მასშტაბში მომუშავე მრავალფუნქციური კონტროლერებისაკენ მიმავალი რთული გზა. განსაკუთრებით წარმატებით ისინი გამოიყენება ისეთი რეალური ობიექტების მართვისათვის, რომელთა ფუნქციონირების ალგორითმებში მრავლადაა გამოყენებული ლოგიკური ფუნქციები. ამას განაპირობებს ის გარემოება, რომ ლოგიკური ფუნქციების რეალიზების სასწრაფე პროცესორის თანრიგიანობაზე არ არის დამოკიდებული. ***სარკინიგზო ავტომატიკისა და ტელემექანიკის სისტემების*** მიერ რეალიზებული ფუნქციების აბსოლუტური უმრავლესობა ლოგიკური ფუნქციებია; ამიტომ აღნიშნული თანა-

მედროვე სისტემების ასაგებად გამოყენებულ უმნიშვნელოვანეს საელემენტო ბაზას წორედ *8*-თანრიგიანი მიკროპროცესორები გვევლინება.

8-თანრიგიანი მიკროკონტროლერების პოპულარობის ზრდას ხელი ისეთი ცნობილი ფირმების მიერ გამოშვებული ნაკეთობათა ნომენკლატურის გაფართოებამაც შეუწყო, როგორიცაა ***Motorola, Microchip, Intel, Zilog, Atmel*** და სხვები.

თანამედროვე *8*-თანრიგიან მიკროკონტროლერების განმასხვავებელი ნიშნებია:

▲ ***მოდულური ორგანიზაცია***, რომლის დროსაც ძირითადი პროცესორული ბირთვის (ცენტრალური პროცესორის) ბაზაზე პროექტდება მიკროკონტროლერების ოჯახი, რომელშიც შემავალი წევრების მეხსიერებები, პერიფერიულ მოდულთა ნაკრებები და სინქრონიზციის სიხშირეები ერთმანეთისაგან განსხვავდება.



ნახ. 11.1 *მიკროკონტროლერის მოდულური სტრუქტურა*

▲ ***დახურული არქიტექტურა***; მისთვის დამახასიათებელია მიკროკონტროლერის კორპუსზე მისამართისა და მონაცემების ხაზების გამომყვანთა არარსებობა; ე. ი. იგი წარმოადგენს მონაცემების ჩაკეტილად დამამუშავებელ სისტემას, რომლის შესაძლებლობების გაზრდა მისამართისა და მონაცემების პარალელური სალტეების გამოყენების გზით არ არის გათვალისწინებული.

▲ ტიპური პერიფერიული ფუნქციონალური მოდულების (ტაიმერების, მიმდევრობითი ინტერფეისის კონტროლერების, ანალურ-ციფრული გარდამქმნელებისა და სხვათა) გამოყენება. სხვადასხვა მწარმოებლის მიერ გამოშ-

შვებული აღნიშნული მოდულები უმნიშვნელო განსხვავებით ასრულებს დავალებულ ალგორითმებს.

▲ ***პერიფერიული მოდულების მუშაობის რეჟიმების რაოდენობის გაფართოების შესაძლებლობა;*** აღნიშნული რეჟიმები დაისახება მიკროკონტროლერის სპეციალურ ფუნქციათა რეგისტრების ინიციალიზაციის პროცესში.

ავების მოდულური პრინციპის დროს ერთი ოჯახში შემავალ სხვადასხვა მოდელის მიკროკონტროლერს აქვს ერთნაირი პროცესორული ბირთვი და ცვლადი ფუნქციური ბლოკი. მოდულური მიკროკონტროლერის სტრუქტურა ***11.1*** ნახაზზეა მოყვანილი.

პროცესორული ბირთვი შეიცავს: ***1)*** ცენტრალურ პროცესორს; ***2)*** მისასამართს. მონაცემებისა და მართვის სალტეებიდან შემდგარ მართვის შიგა მაგისტრალს; ***3)*** მიკროკონტროლერის სინქრონიზაციის სქემას; ***4)*** მიკროკონტროლერის მუშაობის რეჟიმების მართვის სქემას; აღნიშნულ რეჟიმებში შედის დაბალი ენერგომოხმარების რეჟიმი, საწყისი ამუშავების (ჩამოგდების) რეჟიმი და ა.შ.

ცვლად ფუნქციურ ბლოკში გაერთიანებულია ტაქტური გენერატორების მოდული, სხვადასხვა ტიპისა და მოცულობის მეხსიერების მოდულები, შეტანა/გამოტანის პორტები, ტაიმერები. შედარებით მარტივ მიკრონტროლერებში შეწყვეტების დამუშავების მოდული პროცესორულ ბირთვშია შეტანილი, უფრო რთულ მიკროკონტროლერებში იგი ცვლად ფუნქციურ ბლოკში განვითარებული შესაძლებლობების მოდულის სახითაა ფორმირებული. აღნიშნულ ბლოკში შეიძლება შედიოდეს ისეთი დამატებითი მოდულებიც, როგორიცაა ძაბვის კომპარატორები, ანალოგურ-ციფრული ***ᲐᲪᲛ*** გარდამქმნელები და სხვა. თითოეული მოდული შინაგანი კონტროლერული მაგისტრალის პროტოკოლის გათვალისწინებით პროექტდება მიკროკონტროლერში სამუშაოდ. მოცემული მიდგომა საშუალებას იძლევა ერთი ოჯახის ფარგლებში შეიქმნეს მრავალფეროვანი სტრუქტურის მიკროკონტროლერები.

11.3. მიკროკონტროლერის პროცესორული ბირთვი

▌ **<u>მიკროკონტროლერის პროცესორული ბირთვის სტრუქტურა</u>**. ***პროცესორული ბირთვი*** (***MCU - <u>M</u>icroprocessor <u>C</u>ore <u>U</u>nit***) წარმოადგენს მიკროკონტროლერის საფუძველს. იგი ასრულებს ყველა გამოთვლით ოპერაციას და, იმავდროულად, მართავს სქემის ყველა დანარჩენი ნაწილის მუშაობას. სისტემური სალტეებით პროცესორული ბირთვი მონაცემებს ცვლის როგორც მეხსიერებასთან, ისე ყველა ფუნქციურ ბლოკთან. მიკროპროცესორული ბირთვის თანრიგიანობა მთლიანად მიკროკო-

ნტროლერის თანრიგიანობას განსაზღვრავს. დღეისათვის ყველაზე გავრცელებულია ***8***-თანრიგიანი მიკროკონტროლერები. გარდა ამ კონტროლერებისა, მარტივ სქემებში ფართოდ გამოიყენება ***4***-ბიტური, ხოლო რთულ მაღალმწარმოებლურ სისტემებში – ***16***- და ***32***-თანრიგიანი მიკროკონტროლერები.

თანამედროვე ***8*** თანრიგიანი მიკროკონტროლერების პროცესორულ ბირთვში გამოიყენება ბრძანებათა სრული სისტემის (***C****omplicated* ***I****nstruction* ***S****et* ***C****omputer*) მარეალიზებელი ***CISC***-არქიტექტურის პროცესორი, ან ბრძანებათა შემოკლებული სისტემის (***R****educed* ***I****nstruction* ***S****et* ***C****omputer*) მარელიზებელი ***RISC***-არქიტექტურის პროცესორი. განვიხილოთ ორივე მათგანი.

▲ ***CISC-პროცესორები*** იყენებს დამისამართების განვითარებული შესაძლებლობების მქონე ბრძანებათა დიდ ნაკრებს, რის გამოც აღნიშნული პროცესორიანი მიკროკონტროლერის გამოყენების შემთხვევაში საჭირო ოპერაციის შესასრულებლად ბრძანებათა სისტემიდან შესაძლებელია ყველაზე შესაფერისი ბრძანების ამორჩევა. ***8***-თანრიგიან მიკროკონტროლერის შემთხვევაში ***CISC-არქიტექტურიან პროცესორს*** შეძლება ჰქონდეს ერთბაიტიანი, ორბაიტიანი და სამბაიტიანი (იშვიათად - ოთხბაიტიანი) ფორმატის ბრძანებები. ამ დროს ბრძანებათა სისტემა ***არაორთოგონალური***ა, რაც იმას ნიშნავს, რომ ყველა ბრძანებას არ შეუძლია პროცესორის ნებისმიერი რეგისტრისათვის დამისამართების ნებისმიერი ხერხი გამოიყენოს. შესასრულებელი ბრძანება მეხსიერებიდან რამდენიმე ციკლის განმავლობაში ცალკეული ბაიტების სახით ამოიკრიფება. ბრძანების შესასრულებლად შეიძლება ***1***-დან ***12***-მდე რაოდენობის ციკლი იყოს საჭირო.

CISC-არქიტექტურიანი პროცესორებია: ***ა)*** ფირმა ***Intel***–ის მიერ დამუშავებული ***MCS-51*** ბირთვიანი მიკროკონტროლერები, რომლებსაც დღეს უშვებს მრავალი მწარმოებელი. ბ) ფირმა ***Motorola***–ის მიერ გამოშვებული მიკროკონტროლერთა ***HCO5, HCO8*** , ***HCO11*** ოჯახები და სხვები.

▲ ***RISC***-არქიტექტურიან პროცესორებში მინიმუმამდეა შემცირებული შესასრულებელ ბრძანებათა ნაკრები. რთული ოპერაციების რეალიზებისათვის ბრძანებებათა კომბინირებაა საჭირო. ამ დროს ყველა ბრძანება ფიქსირებული სიგრძისაა (შეიცავს ***12, 14*** ან ***16*** ბიტს). მეხსიერებიდან ბრძანების ამოსაღებად და მის შესასრულებლად სინქრონიზაციის თითო-თითო ციკლია (ტაქტია) საკმარისი. ***RISC***-პროცესორის ბრძანებათა სისტემა პროცესორის ყველა რეგისტრის თანაბარუფლებიანი გამოყენების საშუალებას იძლევა, რაც ოპერაციების შესრულების დროს დამატებით მოქნილობას უზნველყოფს.

RISC-პროცესორებს მიეკუთვნება: ***ა)*** ფირმა ***Atmel***-ის მიერ გამოშვებული ***AVR*** ტიპის მიკროკონტროლერები ***ბ)*** ფირმა ***Microchip***-ის მიერ გამოშვებული ***PIC16***, ***PIC17*** ტიპის მიკროკონტროლერები და სხვები.

შინაგანი მაგისტრალის ერთი და იმავე ტაქტური სიხშირის დროს ***CISC***-პროცესორიან მიკროკონტროლერებთან შედარებით ***RISC***-პროცესორებიან მიკროკონტროლერებს, ერთი შეხედვით, უფრო მაღალი მწარმოებლურობა უნდა ჰქონდეს, მაგრამ სინამდვლეში მწარმოებლურობის შეფასების საკითხი მეტად რთულია და არაცალსახა გადაწყვეტა აქვს.

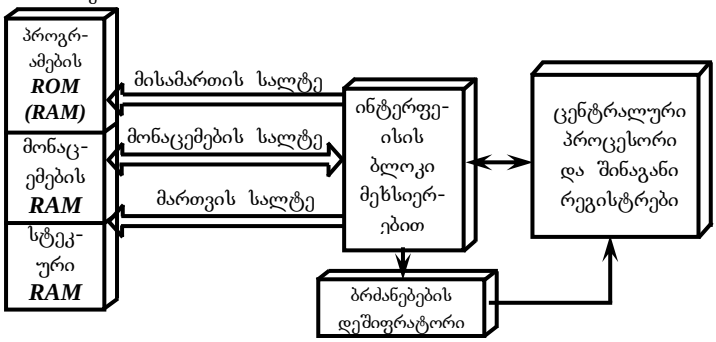
ჯერ ერთი, სავსებით კორექტული არ არის ბრძანებების შესრულების ხანგრძლივობის მიხედვით განსხვავებული (***RISC***, ***CISC***) სისტემის მიკროკონტროლერების მწარმოებლურობის შედარება. ჩვეულებრივ მიღებულია, რომ მიკროპროცესორისა და მიკროკონტროლერის ***მწარმოებლურობა შეფასდეს*** რეგისტრიდან რეგისტრში ერთი წამის განმავლობაში შესრულებული გადაგზავნის ოპერაციების რაოდენობის მიხედვით. ***CISC***-პროცესორიან მიკროკონტროლერში რეგისტრიდან რეგისტრში გადაგზავნის დრო ***1***-დან ***3***-ციკლამდეა, და თითქოს მისი მწარმოებლურობა უნდა ჩამოუვარდებოდეს ***RISC***-პროცესორიანი მიკროკონტროლერის მწარმოებლურობას, მაგრამ ***RISC***-პროცესორის დროს ბრძანებათა სისტემის ორთოგონალობის შენარჩუნების პირობებში ბრძანებათა ფორმატის შემოკლებისაკენ სწრაფვამ შეზღუდა ერთ ბრძანებისათვის მისაწვდომი რეგისტრების რაოდენობა. მაგალითად, ***PIC16*** მიკროკონტროლერის ბრძანებათა სისტემით შესაძლებელია ოპერაციის შედეგი გადაიგზავნოს მხოლოდ წყარო-რეგისტრ **f**–ში ან სამუშაო რეგისტრ ***W***-ში. ამიტომ ამ რეგისტრებისაგან განსხვავებულ რეგისტრში ერთ-ერთ ხელმისაწვდომი რეგისტრიდან შიგთავსის გადასაგზავნად ***ორი ბრძანების გამოყენება*** დაგვჭირდება. ამის აუცილებლობა ხშირად მაშინ წარმოიშობა, როდესაც მიკროკონტროლერის ერთ-ერთ პორტში საერთო დანიშნულების რომელიმე რეგისტრის შიგთავსია გადასაგზავნი. მაშასადამე, ბრძანებათა რთული სისტემა ზოგჯერ ოპერაციის შესრულების უფრო ეფექტური ხერხის რეალიზების საშუალებას გვაძლევს.

მეორეც, რეგისტრიდან რეგისტრში გადაგზავნის სიჩქარის მიხედვით მიკროკონტროლერის მწარმოებლურობის შეფასება არ ითვალიწინებს მართვის კონკრეტულად რეალიზებადი ალგორითმის თავისებურებებს. მაგალითად, ავტომატიზებული მართვის სწრაფმოქმედი მოწყობილობების შექმნისას ძირითადი ყურადღება უნდა დაეთმოს გადაცემის სხვადასხვა ფუნქციის რეალიზებისას გამრავლებისა და გაყოფის ოპერაციების შესრულების ხანგრძივობას, ხოლო საყოფაცხოვრებო ტექნიკის დისტანციური მართვის პულტის რეალიზაციის დროს საჭიროა შეფასდეს კლავიატურის გამოკითხვისას გამოყენებული ლოგიკური ფუნქციების შესრულებისა და მართვის მიმდევრობითი

კოდური გზავნილის გენერაციის ხანგრძლივობა. ამიტომ მაღალი სწრაფმოქმედების მომთხოვნ კრიტიკულ სიტუაციებში მწარმოებლურობის განმსაზღვრელ ფაქტორებში დომინირებს იმ ოპერაციათა სიმრავლის შესრულების სისწრაფე,

მესამეც, აუცილებელია იმის გათვალისწინებაც, რომ მიკროკონტროლის სინქრონიზაციის სიხშირის შესახებ ცნობარებში მოყვანილი მონაცემები ჩვეულებრივ შეესაბამება მისაერთებელი კვარცული რეზონატორის სიხშირეს, მაშინ როდესაც ცენტრალური პროცესორის ციკლის ხანგრძლივობა შინაგან კონტროლერული მაგისტრალით ინფორმაციის გაცვლის სიხშირით განისაზღვრება. ზემოთ მოყვანილ ორ სიხშირეს შორის თანაფარდობა თითოეული მიკროკონტროლერისათვის ინდივიდუალურია და იგი უნდა იქნეს გათვალისწინებული სხვადასხვა მოდელის კონტროლერების მწარმოებლურობის შედარებისას.

ბრძანების ამოღებისა და შესრულების პროცესების ორგანიზაციის თვალსაზრისით თანამედროვე ***8***-თანრიგიან მიკროკონტროლერებში გამოიყენება უკვე ნახსენები ფონნეიმანისეული (პრისტონული) ან ჰარვარდული არქიტექტურა.მეხსიერების არე



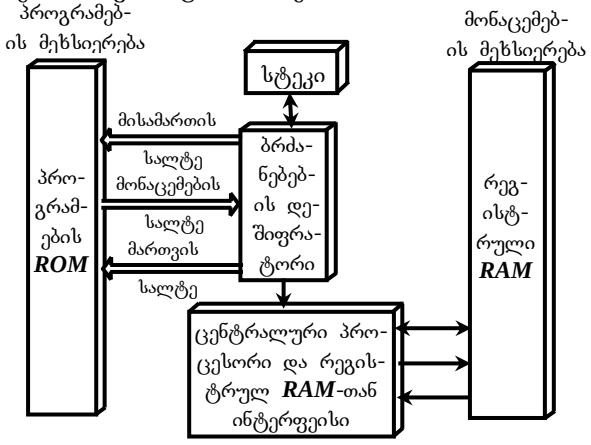
ნახ. 11.2. *ფონნეიმანისეული (პრისტონული) არქიტექტურის მიკროპროცესორული სისტემა*

▲ ***ფონნეიმანისეული არქიტექტურის*** ძირითადი თავისებურებაა პროგრამებისა და მონაცემების შესანახად საერთო მეხსიერების გამოყენება (ნახ. ***11.2***). ერთი მეხსიერების არსებობა ამარტივებს მიკროპროცესორული სისტემის სტრუქტურას. იგი მონაცემების სფეროებს შორის რესურსების ოპერატიული გადანაწილების საშუალებასაც იძლევა, რაც მნიშვნელოვნად ამაღლებს მიკროპროცესორული სისტემის მოქნილობას. საერთო მეხსიერებაში სტეკის განთავსებამ გააიოლა მის შიგთავსთან შეღწევა. ყოველივე ზემოთ ჩამოთვლილის გამო ფონნეიმანისეული არქიტექტურა უნივერსალური, მათ

შორის პერსონალური კომპიტერებისათვის ძირითად არქიტექტურად გადაიქცა.

▲ ***ჰარვარდული არქიტექტურის*** ძირითადი თავისებურებაა ბრძანებებისა და მონაცემების შესანახად სხვადასხვა სამისამართო სივრცის გამოყენება (***ნახ. 11.3***). წინა საუკუნის ***70***-იან წლებამდე, ე. ი. მანამადე, სანამ მიკროკონტროლერების მწარმოებლებისათვის ნათელი არ გახდა, რომ აღნიშნულ არქიტექტურას შეეძლო ავტონომიური მართვის სისტემათა შემქმნელებისათვის გარკვეულ უპირატესობის მოტანა, ამ არქიტექტურას არ გამოიყენებდნენ.

სხვადასხვა ობიექტების მართვაში მიკროპროცესორული სისტემების გამოყენებამ გვიჩვენა, რომ მართვის ალგორითმების უმრავლესობის რეალიზებისათვის დიდი მნიშვნელობა არა აქვს ფონ ნეიმანისეული არქიტექტურის ისეთ ღირსებებს, როგორებიცაა მოქნილობა და უნივერსალობა. ანალიზით დადგინდა, რომ მართვის პროგრამების შესრულებისას საშუალედო შედეგების დამახსოვრებისათვის საჭირო მეხსიერების მოცულობა პროგრამების შესანახად საჭირო მეხსიერების მოცულობის მესამედზე ნაკლებია. ასეთ პირობებში მეხსიერების ერთიანი სამისამართო სივრცის გამოყენება ოპერანდების დამისამართებისათვის საჭირო თანრიგების რაოდენობის გაზრდის ხარჯზე ადიდებს ბრძანებების ფორმატს. მონაცემების შესანახად მცირე მოცულობის მეხსიერების გამოყენება ამოკლებს ბრძანებათა სიგრძეებს და აჩქარებს მონაცემების მეხსიერებაში ინფორმაციის მოძიებას.



ნახ. 11.3. *ჰარვარდული არქიტექტურის მიკროპროცესორული სისტემა*

გარდა ამისა, ფონნეიმანისეულ არქიტექტურასთან შედარებით ჰარვარდულ არქიტექტურას პარალელური ოპერაციების რეალიზაციის ხარჯზე

აქვს პროგრამების უფრო სწრაფად შესრულების პოტენციალი. მოცემული ბრძანების შესრულების პროცესშივე შეიძლება ამოღებული იქნეს მომდევნო ბრძანება, ე. ი. ბრძანების ძიების განმავლობაში პროცესორის მოცდენა გამოირიცხება. ამ მეთოდის შემწეობით სხვადასხვა ბრძანებები შეიძლება ერთნაირი რაოდენობის ტაქტებში შესრულდეს, რაც ციკლების შესრულების დროისა და პროგრამის კრიტიკული უბნების უფრო მარტივად განსაზღვრის საშუალებას გვაძლევს.

თანამედროვე 8-თანრიგიანი მიკროკონტროლერების უმრავლესობაში ჰარვარდული არქიტექტურაა გამოყენებული. მისი ნაკლია ზოგიერთი პროგრამული პროცედურების რეალიზებისათვის არასაკმარისი მოქნილობა. ამიტომ სხვადასხვა არქიტექტურის გამოყენებით რეალიზებული მიკროკონტროლერები ერთმანეთს გამოყენების კონკრეტული სფეროს გათვალისწინებით უნდა შეუდარდეს.

<u>მიკროკონტროლერის პროცესორის ბრძანებათა სისტემა.</u> ნებისმიერი მიკროპროცესორული სისტემის მსგავსად მიკროკონტროლერის პროცესორის ბრძანებათა სიმრავლე შემდეგი ოთხი სახის ბრძანებებს მოიცავს: ***1)*** მონაცემთა გადაგზავნის ბრძანებებს, ***2)*** არითმეტიკულ ბრძანებებს, ***3)*** ლოგიკურ ბრძანებებს, ***4)*** გადასვლათა ბრძანებებს.

პორტების (რეგისტრების) თანრიგების დამოუკიდებლად მართვისათვის თანამედროვე მიკროკონტროლერთა უმრავლესობაში ბიტური მართვის ბრძანებათა ჯგუფიც (ბულის ანუ ბიტური პროცესორიც) გამოიყენება. თანამედროვე მიკროკონტროლერებში გამოიყენება *ბიტური ბრძანებების ჯგუფიც* (ბულის ან ***ბიტური პროცესორი***). ბიტური პროცესორის ბრძანებათა არსებობა მნიშვნელოვნად ამცირებს მმართველი პროგრამების მოცულობასა და მათი შესრულების დროს.

მიკროკონტროლერში გამოიყოფა იმ კონტროლერის რესურსების მმართველი ბრძანებები, რომელიც განკუთვნილია შეტანა/გამოტანის პორტების მუშაობის რეჟიმების აწყობისათვის, ტაიმერის მართვისათვის და ა.შ. თანამედროვე მიკროკონტროლერების უმრავლესობაში კონტროლერის შინაგანი რესურსები მონაცემების მეხსიერებაში აისახება, ამიტომ რესურსების მართვისათვის გამოიყენება გადაგზავნის ბრძანებები.

უნივერსალური მიკროპროცესორის ბრძანებათა სისტემისაგან განსხვავებით მიკროკონტროლერების ბრძანებათა სისტემაში ნაკლებადაა განვითარებული არითმეტიკული და ლოგიკური ბრძანებების ჯგუფი, მაგრამ სამაგიეროდ მათ აქვთ მონაცემების გადაგზავნისა და ბრძანებათა მძლავრი ჯგუფები. ამას განაპირობებს მიკროკონტროლერის გამოყენების თავისებურება, რომლის თანახმადაც მან მუდმივად უნდა აკონტროლოს ირგვლივ არსებული მდგო-

მარეობა და მასზე დამოკიდებულებით წარმოქმნას მმართველი ზემოქმედებები.

მიკროკონტროლერის სინქრონიზაციის სქემა წარმოქმნის ცენტრალური პროცესორის საკომანდო ციკლების შესრულებისა და შინაგან მაგისტრალზე ინფორმაციის გაცვლისათვის აუცილებელ სინქრონიზაციის სიგნალებს. ცენტრალური პროცესორის სტრუქტურაზე დამოკიდებულებით საკომანდო ციკლმა შეიძლება მოიცვას სინქრონიზაციის ერთიდან რამდენიმე (***4-6***) ტაქტი. სინქრონიზაციის სქემა მიკროკონტროლერის ტაიმერების მუშაობისათვის საჭირო დროის მონიშვნის ფუნქციასაც ასრულებს. სინქრონიზაციის სქემის შემადგენლობაში შედის სინქროსიგნალების საჭირო მიმდევრობის მაფორმირებელი სიხშირის გამყოფები.

11.4. მიკროკონტროლერების რეზიდენტული მეხსიერება

მიკროკონტროლერის ***რეზიდენტული*** (ლათ. *residens* – ადგილზე დარჩენა) ***მეხსიერება*** ეწოდება პროგრამებისა და მონაცემების შენახვა-გამოყენებისათვის საჭირო მეხსიერებას, რომელიც მუდმივადაა (რეზიდენტულადაა) განთავსებული მიკროკონტროლერის სტრუქტურაში. თანამედროვე ***8***-თანრიგიანი მიკროკონტროლრეების დახურული არქიტექტურის რეალიზება მიკროსქემის კრისტალში მეხსიერების ორი ტიპის მოდულის ინტეგრაციით (მუდმივად განთავსებით) გახდა შესაძლებელი. პირველია გამოყენებითი პროგრამების კოდების შესანახი ენერგოდამოუკიდებელი (***RAM***) მეხსიერება, ხოლო მეორეა გამოთვლათა საშუალედო შედეგების შესანახი ოპერატიული დამხსომებელი (***RAM***) მეხსიერება. მიკროკონტროლერების გამოჩენის მომენტიდან დღამდე მრავალი ცვლილება განიცადა ენერგოდამოუკიდებელმა მეხსიერებებმა, რის შემწეობითაც არა მარტო გაიზარდა მათი ტევადობა, სწრაფმოქმედება და მათში ინფორმაციის შენახვის საიმედოობა, არამედ წარმოიქმნა მიკროკონტროლერის რეზიდენტული მესხიერების დაპროგრამების პრინციპულად ახალი ტექნოლოგიები

მომხმარებელთა თვალსაზრისით მიკროკონტროლერის რეზიდენტული მეხსიერება იყოფა პროგრამების მეხსიერებად, მონაცემების მეხსიერებად და რეგისტრულ მეხსიერებად. მოკლედ განვიხილოთ თითოეული მათგანი.

პროგრამების მეხსიერება. პროგრამების მეხსიერებისათვის დამახასიათებელი ძირითადი თვისებაა მისი ენერგოდამოუკიდებლობა, რაც კვების არარსებობისას პროგრამების შენახვის შესაძლებლობას ნიშნავს. მიკროკონტროლერის მომხმარებელთა თვალსაწიერიდან შეიძლება ერთმანეთისაგან

განვასხვაოთ პროგრამების ენერგოდამოუკიდებელი სახის შემდეგი მეხსიერებები:

▲ ***ნიღბური ტიპის მუდმივი მეხსიერება***, ანუ, როგორც მას ხშირად ***mask-ROM***-ს უწოდებენ. ამ ტიპის მუდმივი მეხსიერების უჯრედებში შიგთავსები ნიღბების დახმარებით შეიტანება და შემდეგ შეუძლებელი ხდება მისი შეცვლა. ამიტომ პროგრამების ასეთი ტიპის მეხსიერებაში საჭიროა პრაქტიკაში საკმაოდ კარგად შემოწმებული პროგრამები შევიტანოთ, რადგან შეტანის შემდეგ მათი შეცვლა შეუძლებელი გახდება. მოცემული მეხსიერების ძირითადი ნაკლია მაღალი ღირებულება, რაც განპირობებულია ფოტოშაბლონების ახალი კომპლექსების შექმნასა და წარმოებაში დანერგვასთან დაკავშირებულ დიდ ხარჯებთან. ჩვეულებრივ, ამ პროცესს ***2-3*** თვე სჭირდება და ეკონომიკურად მაშინაა გამართლებული, როდესაც რამდენიმე ათეული ათასი ხელსაწყოს გამოშვებაა ნავარაუდევი. ნიღბური ტიპის მუდმივი დამხსომებელი მეხსიერების ***ღირსებაა*** ქარხნული პირობებში დაპროგრამებისას ინფორმაციის შენახვის მაღალი საიმედოობა.

▲ ***ულტრაიისფერი წაშლის უნარიანი გადაპროგრამებადი მუდმივი დამხსომებელი მოწყობილობა EPROM*** (***Erasable Programmable ROM***). მოცემული ტიპის მუდმივი მეხსიერება ელექტრული სიგნალებით დაპროგრამდება და ულტრაიისფერი დასხივებით წაიშლება ***EPROM*** მეხსიერების უჯრედები წარმოადგენს «მცურავი» საკეტის მქონე «ლითონი-ჟანგეული-ნახევარგამტარი» ტიპის ***ლშნ***-ნახევარგამტარს (***MOS***-ნახევარგამტარს), რომელშიც მუხტი შესაბამისი ელექტრული სიგნალების მოწოდების დროს მმართველი საკეტიდან გადაიცემა. უჯრედის შიგთავსის წასაშლელად იგი დასხივდება ულტრაიისფერი შუქით, რომელიც მცურავ საკეტს გადასცემს პოტენციური ბარიერის გადალახვისა და ფუძეშრეზე ჩამოდინებისათვის საკმარის ენერგიას. ეს პროცესი შეიძლება რამდენიმე წამიდან რამდენიმე წუთამდე გაგრძელდეს. ***EPROM***-მეხსიერებიანი მიკროკონტროლერი შეიძლება მრავალჯერადად დაპროგრამდეს. იგი მოთავსებულია ულტრაიისფერი სხივის გასატარებლად საჭირო კვარცული სარკმლიან კერამიკული კორპუსში. ეს კორპუსი საკმაოდ ძვირია, რაც მნიშვნელოვნაა აძვირებს მიკროკონტროლერს. ფასის შესამცირებლად გამოდის უსარკმლო კორპუსში ჩასმული ***EPROM***-მეხსიერებიანი მიკროკონტროლერი, რომელსაც ერთჯერადად დაპროგრამებადი ***EPROM*** ვერსია ეწოდება.

▲ მომხმარებლის მიერ ერთჯერადად დაპროგრამებადი მუდმივი დამხსომებელი მოწყობილობა – ***OTPROM*** (***One-Time Programmable ROM***). იგი წარმოადგენს მიკროკონტოლერის გასაიაფებლად უსარკმლო კერამიკულ კორპუსში ჩასმული ***EPROM*** მეხსიერების ვერსიას. ასეთი კორპუსების გამოყენებით გამოწვეული გაიაფება იმდენად მნიშვნელოვანია, რომ ამ ვერსიის

მეხსიერებას ხშირად ნიღბური მუდმივი დამხსომებელი მოწყობილობის ნაცვლადაც გამოიყენებენ.

▲ ელექტრული წაშლის უნარის მქონე და მომხმარებლის მიერ დაპროგრამებადი მუდმივი დამხსომებელი მოწყობილობა – ***EEPROM*** (***E****lectrically* ***E****rasable* ***P****rogrammable* ***ROM***). იგი წარმოადგენს ***EPROM*** მეხსიერების ახალ თაობას, რომელშიც გვირაბული მექანიზმების გამოყენების ხარჯზე შესაძლებელია მეხსიერების უჯრები ელექტრული სიგნალებითვე წავშალოთ. ***EEPROM***-ის გამოყენებისას შეგვიძლია მიკროკონტროლერი დაფიდან მოუხსნელად დავაპროგრამოთ. ამ ხერხის წყალობით შეიძლება გავმართოთ და შევცვალოთ პროგრამული უზრუნველყოფა. ეს საშუალებას გვაძლევს მნიშვნელოვანი მოგება მივიღოთ მიკროკოტროლერული სისტემების დამუშავების საწყის სტადიაზე ან მათი შესწავლის პროცესში, როდესაც სისტემის მუშაობის უუნარობის მიზეზების პოვნასა და პროგრამების მეხსიერების წაშლა-დაპროგრამების ციკლების შესრულებაზე დიდი დრო გვეკარგება. ფასის მიხედვით ***EEPROM*** იკავებს საშუალო მდგომარეობას ***OTPROM***-სა და ***EPROM***-ს შორის. ***EEPROM***-მეხსიერების დაპროგრამების ტექნოლოგიის დროს შესაძლებელია უჯრედები ბაიტობით წაიშალოს და/ან დაპროგრამდეს. ***EEPROM***-ის თვალნათელი უპირატესობის მიუხედავად, პროგრამების შესანახად ასეთი მეხსიერება მიკროკონტროლერებში იშვიათად გამოიყენება. ამას განაპირობებს ის რომ, ჯერ ერთი, ***EEPROM***-მეხსიერებას გააჩნია შეზღუდული მოცულობა, და მეორეც, ის, რომ გამოჩნდა გაცილებით იაფი ***Flesh*** მეხსიერება, რომლის სამომხმარებლო მახასიათებლები დაემთხვა ***EEPROM***–ის ანალოგურ მახასიათებლებს.

▲ ელექტრული წაშლის უნარის მქონე ***Flash*** ტიპის მუდმივი დამხსომებელი მოწყობილობა – ***Flash-ROM***. ფუნქციურად ***Flash-ROM*** და ***EEPROM*** მეხსიერებები ერთმანეთისაგან ძირითადად ჩაწერილი ინფორმაციის წაშლის ხერხის მიხედვით განსხვავდება. ***EEPROM***-მეხსიერებაში ინფორმაცია ბაიტებად შეიძლება წაიშალოს, ***Flash***-მეხსიერებაში კი – ბაიტების შემცველ ბლოკებად. ***Flash***-მეხსიერების ერთადერთი უჯრედის შიგთავსის შესაცვლელად მთელი ბლოკის გადაპროგრამებაა საჭირო. ***EEPROM***-თან შედარებით მადეკოდირებელი სქემების გამარტივებამ ***Flash***-მეხსიერებიანი მიკროკონტროლერი არა მარტო ერთჯერადად დაპროგრამებადი მუდმივი მეხსიერების მქონე, არამედ ნიღბური მუდმივი მეხსიერების მქონე მიკროკონტროლერზე უფრო კონკურენტუნარიანი გახადა.

<u>მონაცემების მეხსიერება.</u> მიკროკონტროლერებში მონაცემების მეხსიერება, როგორც წესი, ***სტატიკური ოპერატიული დამხსომებელი მოწყობილობის*** საფუძველზეა აგებული. ტერმინი «სტატიკური» ნიშნავს, რომ ენერგომოხმარების შემცირების მიზნით ოპერატიული დამხსომებელი მოწყო-

ბილობის უჯრედების შიგთავსი მიკროკონტროლერის ტაქტური სიხშირის რაგინდ მცირე მნიშვნელობამდე შემცირების დროსაც შენარჩუნდება. მიკროკონტროლერების უმრავლესობას აქვს ინფორმაციის შენახვისათვის აუცილებელი $U_{STANDBY}$ ***ძაბვის პარამეტრი.*** კვების ძაბვის ნომინალურად დასაშვებ U_{DDMIN} დონეზე დაბალ, მაგრამ $U_{STANDBY}$-ზე მაღალ მნიშვნელობამდე შემცირებისას მიკროკონტროლერი პროგრამას ვერ შეარულებს, მაგრამ შეინარჩუნებს ოპერატიულ დამხსომებელ მოწყობილობაში შენახულ ინფორმაციას. კვების ძაბვის აღდგენის შემდეგ მიკროპროცესორი შეიძლება «ჩამოიყაროს» და მონაცემების დაუკარგავად გაგრძელდეს პროგრამის შესრულება. მონაცემების შესანარჩუნებლად საჭიროა $U_{STANDBY}$ ძაბვა დაახლოებით ***1 V***-ს ტოლი იყოს. რაც საშუალებას იძლევა საჭიროების შემთხვევაში მიკროკონტროლერი გადაყვანილი იქნეს ავტონომიური წყაროდან კვებაზე და ამ რეჟიმში შენარჩუნდეს მონაცემები ოპერატიულ დამხსომებელ მოწყობილობაში.

მიკროკონტროლერში მონაცემების მეხსიერების მოცულობა, როგორც წესი, მცირეა და ჩვეულებრივ ათეულ და ასეულ ბაიტს შეადგენს. ეს უნდა გავითვალისწინოთ მიკროკონტროლერის დაპროგრამებისას. მაგალითად, მიკროკონტროლერის დაპროგრამებისას შესაძლებლობის არსებობის შემთხვევაში კონსტანტები ცვლადების სახით კი არ შეინახება, არამედ პროგრამების მუდმივ დამხსომებელ მოწყობილობაში შეიტანება. მაქსიმალურად უნდა გამოვიყენოთ მიკროკონტროლერის აპარატურული შესაძლებლობები, კერძოდ, ***ტაიმერები***. გამოყენებითი პროგრამების დროს უნდა ვეცადოთ თავი ავარიდოთ მონაცემთა დიდი მასივებს.

<u>მიკროკონტროლერთა რეგისტრული მეხსიერება.</u> ნებისმიერი მიკროპროცესორული სისტემების მსგავსად მიკროკონტროლერებშიც არსებობს მართვისათვის საკუთარი რესურსების გამომყენებელი რეგისტრების ნაკრები. მასში ჩვეულებრივ შედის ***პროცესორის რეგისტრები*** (აკუმულატორი, მდგომარეობისა და ინდექსური რეგისტრები), ***მართვის რეგისტრები*** (შეწყვეტებით, ტაიმერით მართვის რეგისტრები), ***მონაცემების შემტანი და გამომტანი რეგისტრები*** (პორტების მონაცემთა რეგისტრები, პარალელური, მიმდევრობითი ან ანალოგური შეტანის მართვის რეგისტრები). ამ რეგისტრებს სხვადასხვაგვარად უნდა მივმართოთ.

RISC-პროცესორიან მიკროკონტროლერში ყველა რეგისტრი (ხშირად, აკუმულატორიც) ცხადად მისამართდება. ეს მუშაობის დროს მაღალ მოქნილობას უზრუნველყოფს.

მიკროკონტროლერის სამისამართო სივრცეში რეგისტრების განთავსება ერთ-ერთი უმნიშვნელოვანესი საკითხია. ზოგიერთ მიკროკონტროლერში ყველა რეგისტრი და მონაცემთა მეხსიერება ერთსა და იმავე სამისამართო

სივრცეში განთავსდება. ეს ნიშნავს, რომ მონაცემების მეხსიერება რეგისტრებთანაა შეთავსებული. ამ მოვლენას «***მეხსიერებაში მიკროკონტროლერის რესურსების ასახვა***» ეწოდება.

სხვა მიკროკონტროლერებში შეტანა/გამოტანის მოწყობილობების სამისამართო სივრცე გამოყოფილია მეხსიერების საერთო სივრცისაგან. შეტანა/გამოტანის განცალკევებული სივრცე ჰარვარდული არქიტექტურიან პროცესორს გარკვეულ უპირატესობას ანიჭებს, კერძოდ, შეტანა/გამოტანის რეგისტრთან მიმართვის პარალელურად მათ ბრძანების წაკითხვაც შეუძლია.

<u>მიკროკონტროლერის სტეკი.</u> მიკროკონტროლერებში ოპერატიულ დამხსომებელ მოწყობილობას ქვეპროგრამების გამოსაძახებლად და შეწყვეტების დასამუშავებლადაც იყენებენ. ამ ოპერაციების დროს საჭიროა შენახული იქნეს პროგრამული მთვლელისა და ძირითადი რეგისტრების (აკუმულატორის, მდგომარეობის რეგისტრისა და სხვების) შიგთავსები, ხოლო ძირითად პროგრამაზე გადასვლისას ისინი ხელახლა აღდგეს.

ფონ ნეიმანისეულ არქიტექტურაში მეხსიერების ერთიანი არე სტეკის რეალიზებისთვისაც გამოიყენება. ეს ამცირებს მოწყობილობის მწარმოებლურობას, რადგან სხვადასხვა სახის მეხსიერებასთან ერთდროულად მიმართვა შეუძლებელია. კერძოდ, ქვეპროგრამის გამოძახების ბრძანების შესრულების შემდეგი ბრძანების ამოღება მხოლოდ სტეკში პროგრამული მთვლელის შიგთავსის მოთავსების შემდეგაა შესაძლებელი.

ჰარვარდულ არქიტექტურაში სტეკური ოპერაციები ამ მიზნით სპეციალურ მეხსიერებაში შეიძლება შესრულდეს. ამის გამო ქვეპროგრამის გამოძახების ბრძანების შესრულებისის პერიოდშივეა შესაძლებელი რამდენიმე სხვა ბრძანების ერთდროული შესრულება.

აუცილებელია, გვახსოვდეს, რომ ორივე არქიტექტურიან მიკროკონტროლერში მონაცემების მცირე მოცულობის მეხსიერებაა გამოყენებული. პროცესორში განცალკევებული სტეკის არსებობისას მასში ჩაწერილი მონაცემების მოცულობა თუ გადააჭარბებს სტეკის ტევადობას, მაშინ ციკლურად შეიცვლება სტეკის მაჩვენებელი და იგი დაიწყებს სტეკის ადრე ჩაწერილ უჯრედებზე მითითებას. ეს ნიშნავს, რომ ძალიან ბევრი ქვეპროგრამის გამოძახებისას სტეკში დაბრუნების არასწორი მისამართი აღმოჩნდება. სტეკში მონაცემების ჩასაწერად მიკროკონტროლერი თუ მეხსიერების საერთო არეს გამოიყენებს, მაშინ არსებობს იმის საშიშროება, რომ სტეკის გადავსებისას მასში ჩასატვირთი მონაცემები ჩაიწეროს მონაცემების სფეროში ან მუდმივ დამხსომებელ მოწყობილობაში.

გარე მეხსიერება. მიკროკონტროლერების დახურულ არქიტექტურაზე გადასვლის დამკვიდრებული ტენდენციის მიუხედავად მაინც წარმოიშობა როგორც პროგრამების, ისე მონაცემების მეხსიერებასთან გარე მეხ-

სიერების დამატებით მიერთების საჭიროება. მისი დაკმაყოფილების შემდე-გი ორი ხერხი არსებობს: ***1)*** მიკროკონტროლერში გარე მეხსიერებასთან მიერთების სპეციალური აპარატურის გათვალისწინება; ***2)*** გარე მეხსიერება-სთან მისაერთებლად შეტანა/გამოტანის მოწყობილობების გამოყენება.

პირველი ხერხი უზრუნველყოფს მაღალ სწრაფმოქმედებას, მაგრამ ზრდის აპარატურულ სიჭარბეს; მეორე ხერხი არ ზრდის სტრუქტურულ სიჭარბეს, მაგრამ ამცირებს სისტემის სწრაფმოქმედებას.

11.5. შეტანა/გამოტანის პორტები

თითოეულ მიკროკონტროლერს აქვს გარკვეული რაოდენობის შეტანა-გა-მოტანის ხაზი. ისინი გაერთიანებულია შეტანა/გამოტანის ***8***-თანრიგიან პა-რალელურ ***PTx*** პორტებად, სადაც «***x***» არაბული ციფრების ან ლათინუ-რი ალფაბეტის ასოებით გამოხატული პორტის სახელია; პირველ შემთხვე-ვაში ვიღებთ ***PT1, PT2, PT2*** და ა. შ., ხოლო მეორე შემთხვევაში - ***PTA, PTB, PTC*** და ა. შ გამოსახულებებს. მიკროკონტროლერის მეხსიერების ბარათში შეტანა/გამოტანის თითოეული პორტი წარმოდგენილია ამ პორტის მონა-ცემების ***DPTx*** რეგისტრის სახით.

შეტანის რეჟიმში PTx პორტის რეგისტრში ჩაიწერება პორტის ხაზებზე ფორმირებული ორობითი კოდი. ***გამოტანის რეჟიმში*** პროგრამის მართვით ***DPTx*** რეგისტრში ჩაწერილი მონაცემები გადაეცემა მიკროკონტროლერის იმ გამომყვანებზე, რომლებიც ***PTx*** პორტის ხაზებადაა გამოყენებული. მონა-ცემების ***DPTx რეგისტრთან მიმართვისათვის*** გამოიყენება ის ბრძანებები, რომლებიც ოპერატიული რეზიდენტული მეხსიერების უჯრედების მიმართვი-სათვისაა განკუთვნილი. გარდა ამისა, მრავალ მიკროკონტროლერში პორტე-ბის ცალკეული თანრიგი შეიძლება ***ბიტური პროცესორის*** ბრძანებებით იქ-ნეს გამოკითხული.

ფუნქციონალური თვალსაზრისით არსებობს: ***1. ცალმხრივმიმართული პორტები,*** რომლებიც მიკროკონტროლერის სპეციფიკაციის შესაბამისად გა-მოიყენება ინფორმაციის მარტო შესატანად ან გამოსატანად; ***2. ორმხრივმი-მართული პორტები,*** რომლებშიც გადაცემის მიმართულება (შეტანა თუ გა-მოტანა) სისტემის ინიციალიზაციის პროცესში განისაზღვრება; ***3. ალტერ-ნატიული ფუნქციის პორტები.*** ასეთი პორტების ცალკეული ხაზები დაკა-ვშირებულია მიკროკონტროლერში ჩაშენებულ ისეთ პერიფერიულ მოწყობი-ლობებთან, როგორებიცაა ტაიმერი, ანალოგურციფრული გარდამქმნელი, მი-მდევრობითი მიმღებ-გადამცემი კონტროლერები. მიკროკონტროლერის შესა-ბამისი პერიფერიული მოდულის არარსებობისას მისი გამომყვანები შეიძლე-ბა შეტანა/გამოტანის ჩვეულებრივ ხაზებად ამოქმედდეს. პირიქით, თუ მო-

დული გააქტივებულია, მაშინ შეტანა/გამოტანის მისი კუთვნილი ხაზები მოდულში მისთვის დაკისრებული ფუნქციონალური დანიშნულების შესაბამისად ავტომატურად ფუნქციონირებს და შეუძლებელია შეტანა/გამოტანის ხაზებად მათი გამოყენება. როგორც ცალმხრივმიმართული, ისე ორმხრივმიმართული პორტების ხაზებს შეიძლება ჰქონდეს ალტერნატიული ფუნქციები.

განვიხილოთ მიკროკონტროლერის შეტანა/გამოტანის ხაზების ბუფერთა სქემოტექნიკური თავისებურებები. სპეციალურ ლიტერატურაში შეტანა/გამოტანის ხაზების გამოსასვლელ კასკადებს ხშირად ***დრაივერებს*** უწოდებენ. ანალოგური სახელწოდება აქვს პერსონალურ კონტროლერის პროგრამულ საშუალებებსაც, მაგრამ მათგან განსხვავებით ჩვენ მიერ ზემოთ აღნიშნული დრაივერები აპარატურული და არა პროგრამული საშუალებებია! თანამედროვე მიკროკონტროლერებში ორმხრივმიმართული პორტების უმრავლესობას შეუძლია დამოუკიდებლად დასახოს თითოეულ ხაზში გადაცემის მიმართულება, ე. ი. პორტად გაერთიანებული ხაზების ჯგუფს შეიძლება ისე მიმართოს, როგორც მეხსიერების უჯრედებს, რაც მოსახერხებელია პარალელური ფორნატით გაცვლის ორგანიზებისას. საჭიროებისამებრ თითოეული ხაზი ინდივიდუალურადაც შეიძლება იქნეს კონფიგურირებული და მას იმავე პორტის სხვა ხაზებისაგან დამოუკიდებლად მოემსახუროს ბიტური პროცესორის ბრძანებები.

განასხვავებენ შეტანა/გამოტანის შემდეგი სახის დრაივერებს: 1) ორმხრივმიმართული ხაზები, რომლებიც შეტანაზე ან გამოტანაზე აეწყობა გადაცემის მიმართულების ***DDPTx*** რეგისტრში ბიტის დაპროგრამებით; შეტანის რეჟიმში მუშაობის დროს ხაზს აქვს მაღალი შესასვლელი წინაღობა; ***2)*** ორმხრივმიმართული ხაზები, რომლებიც წინასწარ ინიციალიზაციას არ საჭიროებს; ასეთ ხაზებს წაკითხვისას აქვს განსაკუთრებული უნარი; შეტანის რეჟიმში მუშაობის დროს ამ ხაზსაც მაღალი შესასვლელი წინაღობა აქვს. ***3)*** კვაზი ორმხრივმიმართული ხაზი; არ საჭიროებს წინასწარ ინიციალიზაციას; შეტანის რეჟიმში დრაივერი ავტომატურად ახდენს კვების წყაროს ძაბვის მნიშვნელობის ლოგიკურ ***1***-მდე «ამქაჩავი» რეზისტორს მიერთებას. ***4)*** «ამქაჩავი» რეზისტორების პროგრამულად მიერთების უნარის მქონე ორმხრივმიმართული ხაზები.

პორტებისა და დრაივერების სქემების განხილვა ცდება ჩვენი წიგნის ფარგლებს.

11.6. რეალურ დროში მიკროკონტროლერების ფუნქციონირების საკითხისათვის

მიკროკონტროლერულმა სისტემამ მართვის ამოცანების უმრავლესობა ***რეალურ დროში*** უნდა გადაწყვიტოს. ეს იმას ნიშნავს, რომ მან სამართავი ობიექტის არსებული მდგომარეობის სასურველ მდგომარეობად შესაცვლელად საჭირო დროის განმავლობაში უნდა მოასწროს ყველა ოპერაციის შესრულება (ობიექტის შესახებ საკონტროლო ინფორმაციის მიღება, დამუშავება და მმართველი ზემოქმედების ფორმირება).

რეალურ დროში ფუნქციონირებისათვის აუცილებელი ფუნქციების პროცესორისათვის დაკისრების შემთხვევაში პროცესორი იძულებული გახდება გამოთვლითი პროცედურებისათვის აუცილებელი საკუთარი ძვირადღირებული რესურსები ამ მიზნით დახარჯოს, რაც ეკონომიკურად გაუმართლებელია. ამიტომ თანამედროვე მიკროკონტროლერების უმრავლესობა აღჭურვილია ***რეალურ დროში ფუნქციონირებისათვის აუცილებელი აპარატურული უზრუნველყოფით***, რომელშიც ძირითად ფუნქციებს ტაიმერი (ტაიმერები) ასრულებს.

ტაიმერების მოდული განკუთვნილია: ***1)*** გარკვეული მოვლენის დაწყების ფიქსირებისათვის (ამ ინფორმაციას მას გარე სენსორები აწვდის); ***2)*** მმართველი ზემოქმედების პროცესის დროში განაწილებისათვის (ნებისმიერი ზემოქმედება ზუსტად მაშინ უნდი იქნეს ფორმირებული, როდესაც ეს აუცილებელია მართვის პროცესის ნორმალური მიმდინარეობისათვის).

8-თანრიგიანი მიკროკონტროლერის ტაიმერის მოდული წარმოადგენს მართვის სქემით აღჭურვილ ***8***- ან ***16***-თანრიგიან მთვლელს. მიკროკონტროლერის სქემოტექნიკით ჩვეულებრივ ტაიმერი გარე მოვლენების მთვლელის რეჟიმში მუშაობისათვის არის გათვალისწინებული, ამიტომ მას ჩვეულებრივ ***ტაიმერ/მთვლელს*** უწოდებენ.

მიკროკონტროლერის შემადგენლობაში გამოყენებული ტიპური ტაიმერ/მთვლელების სქემების განხილვა ცდება ჩვენი წიგნის ფარგლებს.

11.7. 8-თანრიგიანი მიკროკონტროლერების პოპულარული ოჯახები

სარკინიგზო ავტომატიკისა და ტელემექანიკის მიკროპროცესორული მოწყობილობების ასაგებად, როგორც ზემოთ აღვნიშნეთ, ფართოდ გამოიყენება ***8***-თანრიგიანი მიკროკონტროლერები. ამის გამო რამდენადმე დაწვრილებით განვიხილავთ აღნიშნულ მიკროკონტროლერებს.

მსოფლიო ბაზარზე წარმოდგენილი ***8***-თანრიგიანი მიკროკონტროლერების მოდიფიკაციათა რაოდენობა იმდენად დიდია, რომ მხოლოდ მათი სახელწოდებათა ჩამონათვალი რამდენიმე ათეულ გვერდს დაიკავებს. ამიტომ ჩვენ აღნიშნული მიკროკონტროლერების მხოლოდ იმ ოჯახებს დავახასიათებთ მოკლედ, რომლებმაც ბოლო წლებში მოიპოვა დიდი პოპულარობა.

ამ ოჯახებს შორის, უპირველეს ყოვლისა, უნდა ვახსენოთ ***MCS-51*** ბირთვის მქონე მიკროკონტროლერები. ***MCS-51*** ბირთვის მქონე მძლავრ კლანს საფუძველი ჩაუყარა ფირმა ***Intel***-მა, რომელმაც ***1980*** წელს გამოუშვა ***8051AH*** სახელწოდების მიკროკონტროლერი. თავისი დროისთვის იგი საკმაოდ რთულ ნაკეთობად ითვლებოდა, რომლის კრისტალში ***128000*** ტრანზისტორი იყო განთავსებული. ეს მიკროკონტროლერი შეიცავდა მიკროპროცესორულ ***MCS-51*** ბირთვს, ***4*** კილობაიტის მოცულობის რეზიდენტულ მუდმივ დამხსომებელ (***ROM***) მოწყობილობას, ***128*** ბაიტიან ოპერატიულ დამხსომებელ (***RAM***) მოწყობილობას, შეტანა/გამოტანის ***4*** პორტს, ***2*** ტაიმერსა და ასინქრონულ პორტს. მასში არსებული ცენტრალური ***MCS-51*** პროცესორის სწრაფმოქმედება თანამედროვე გადასახედიდან არ იყო დიდი - შიგა სალტის სიხშირე შეადგენდა ***1*** მეგაჰერცს. სამაგიეროდ თავად ***MCS-51*** ბირთვი იმდენად წარმატებული აღმოჩნდა, რომ მან რამდენიმე ათეული წლის განმავლობაში ***8***-თანრიგიანი მიკროკონტროლერების ოჯახში ფაქტობრივი სტანდარტის სახელი დაიმკვიდრა.

ფირმა ***Intel***-ი პერმანენტულად სრულყოფდა ***MCS-51*** არქიტექტურის მქონე მიკროკონტროლერს, რის შედეგადაც მის ბოლო მოდელებში შიგა სალტის სიხშირე ***3*** მეგაჰეცამდე გაიზარდა, გამოჩნდა მოდელები, რომლებშიც პროგრამებისათვის განკუთვნილი მეხსიერების მოცულობები იყო ***8, 16*** და ***32*** კილობაიტი (ნაცვლად პირვანდელი ***2*** კილობაიტისა); მიკროკონტროლერის შემადგენლობაში გაჩნდა ახალი პერიფერიული მოდულები (ანალოგურ-ციფრული გარდამქმნელი, მასივების დაპროგრამებადი მთვლელები, მოდარაჯე ტაიმერი).

ამავდროულად მთელმა რიგმა სხვა ფირმებმა დაამუშავეს პროგრამებისა და მონაცემების თანამედროვე (***Flash*** და ***EEP-ROM***) ტიპის მეხსიერებებისა და პერიფერიული მოდულების გაფართოებული ნაკრების მქონე მიკროკონტროლერები, რომლებიც კვებისათვის იყენებდა ფართო დიაპაზონში ცვლად ძაბვასა და პროგრამულად ***MCS-51*** მიკროკონტროლერთან იყო თავსებადი. ფირმა ***Intel***-მა თანდათან შეწყვიტა ***8***-თანრიგიანი მიკროპროცესორების წარმოება. ამის შედეგად ***51***-ე ოჯახის ძირითად მწარმოებლებად გახდა ფირმები «***Phillips***», «***Infineon***», «***Atmel***», «***Dallas Semiconductor***», «***Temic***». ***1999*** წელს ფირმა «***Analog Devices***»-მა ***51***-ე ბირთვის საფუძველზე წარმოადგინა სრულიად ახალი მიკროკონტროლერი ***Adu812.*** მას ჰქონდა ისეთი

შესანიშნავი ტექნიკური მახასიათებლების მქონე ჩაშენებული ***ЦАВ*** და ***ЦАЗ***, რომ ***Adu812***-ის ოჯახს ეწოდა ***ინტელექტუალური გარდამქმნელები*** ანუ ***მიკროკონვერტორები.***

MCS-51 ოჯახის შექმნის პარალელურად ფირმამ «***Motorola***» დაამუშავა მიკროკონტროლერების დღემდე პოპულარული ***HCO5*** ოჯახი. ამ ოჯახის ფარგლებში ფირმა «***Motorola***»-ამ საჯაროდ დასახა «შეკვეთილი» მიკროკოკონტროლერების სტრატეგია, რომელსაც დღემდე წარმატებით ახორციელებს. ამ ოჯახის მრავალი მოდელი იმის წყალობით იქმნება, რომ მსხვილმა მომხმარებლებმა ფირმები «***Motorola***»-ას კონკრეტული პროდუქციისათვის გარკვეული კონფიგურაციის მიკროკონტროლერებს უკვეთავს. დღეს ***HCO5*** ოჯახში ***DIP16*** კორპუსში მოთავსებული უმარტივესი ***68HC05KJ1*** მიკროკონტროლერიდან დაწყებული და უკორპუსო ***128***-გამომყვანიანი ძალიან რთული ***68HC05L10*** მიკროკონტროლერით დამთავრებული (რომელსაც აქვს თხევადკრისტალური ინდიკატორის ***960*** სეგმენტის მმართველი ჩაშენებული კონტროლერი) სხვადასხვა სახის დაახლოებით ***180*** მიკროკონტროლერია გაერთიანებული მთელი. თავისი არსებობის განმავლობაში ***HCO5*** ოჯახი ***MCS-51***-ის ძლიერი და წარმატებული ოპო-ნენტია.

MCS-51 ოჯახის მიკროკონტროლერების საპირწონედ, რომლებიც აგებულია ***ჰარვარდული არქიტექტურის*** გამოყენებით, ***HCO5*** ოჯახის მიკროკონტროლერებში ***პრისტონული (ფონ ნეიმანისეული) არქიტექტურაა*** რეალიზებული. ამ უკანასკნელ ოჯახში მართვის თუნდაც ძალიან მარტივი ამოცანების გადასაწყვეტად გამოყენებულია შესაძლო ტექნიკური გადაწყვეტების მრავალფეროვანი სპექტრი, რისი მეშვეობითაც მომხმარებელის ტექნიკური მოთხოვნების სრულად დაკმაყოფილება მიიღწევა ყოველგვარი არქიტექტურული და მწარმოებლური სიჭარბის გარეშე. ***HCO5*** ოჯახის მიკროკონტროლერების ასეთი ხანგრლივ წარმატებას განსაზღვრავს არც ზესწრაფმოქმედება (მოდელების უმრავლესობის შიგა სალტის სიხშირე ***2*** მეგაჰერცის ტოლია) და არც ბრძანებათა უნიკალური ნაკრები. წარმატებას განაპირობებს მასობრივი მოხმარების ბაზრის სხვადასხვა სექტორზე ძალიან ზუსტ ორიენტაცია. უმარტივესი ***HCO5*** მოდულის უცვლელად შენარჩუნების ფონზე გამოიყენება უაღრესად მრავალფეროვანი პერიფერიული მოდულები. ეს საშუალებას იძლევა, კონსტრუქტორს თითოეული ამოცანისათვის შეიქმნას არქიტექტურის პრაქტიკულად არაჭარბი რესურსი, რაც ნაკეთობიდ მინიმალურ ღირებულებას განაპირობებს.

იაფ «შეკვეთილ» ***HCO5*** ოჯახთან ერთად «***Motorol***»-მ ***1980*** წელსავე შემოგვთავაზა უნივერსალური და უფრო მწარმოებლური მიკროკონტროლერების ***HCO11*** ოჯახი. დღეისათვის ამ ოჯახში ***40***-მდე მოდელია გერთიანებული ***HCO5*** ოჯახისაგან განსხვავებებით ***HCO11*** ოჯახის პროცესორული

ბირთვისთვის შესაძლებელია ოპერაციების შესრულება ***16***-თანრიგიან ოპერანდებზე, იგი იყენებს დამისამართების დამატებით მეთოდებს და მას აქვს შინაგანი სალტეების ამაღლებული (***4*** მეგაჰრცამდე) სიხშირე. ***HCO11*** ოჯახის მიკროკონტროლერებს აქვს ***სამი ტიპის მეხსიერება***, კერძოდ, პროგრამებისათვის განკუთვნილი ერთჯერადად დაპროგრამებადი მუდმივი მეხსიერება, მონაცემებისათვის განკუთვნილი ოპერატიული დამხსომებელი მოწყობილობა და ელექტრულად როგორც დაპროგრამებადი, ისე წაშლადი მეხსიერება მონაცემებისათვის.

წინა საუკუნის ***90***-იან წლებში ფირმა «***Motorola***»-მ მიკონტროლერთა ***HCO5*** სახელწოდების მქონე ოჯახის მომავალში შეცვლისა და ახალი «სამრეწველო სტანდარტის» შექმნის ფორმირების ამბიციური გეგმის რეალიზების მიზნით მომხმარებელთა სამსჯავროზე გამოიტანა ***8***-თანრიგიან მიკროკონტროლერთა ***HCO8*** სახელწოდების მქონე ახალი ოჯახი. ამ ოჯახის დამახასიათებელი ნიშნებია:

▲ მაღალმწარმოებლური ***8***-თანრიგიანი ***ალმ.*** მისი მწარმოებლურობა გაზრდილია: ***ა)*** შიგა სალტეებით ინფორმაციის გაცვლის სიხშირის ***8,0*** მეგაჰერცამდე ამაღლების; ***ბ)*** მოცემული ბრძანების შესრულებისა და მომდევნო ბრძანების ამორჩევის ციკლების ურთიერთშეთავსების; ***გ)*** ცხრილების დათვალიერების სპეციალური ბრძანებების შემოტანისა და შესაბამისი ციკლების ორგანიზების; ***დ)*** ოპერანდების დამისამართების ხერხების რაოდენობის გაზრდის გზით.

▲ ***HCO5*** ოჯახის პროცესორულ ბირთვთან «თავიდან ბოლომდე» პროგრამული შეთავსებადობა როგორც საწყისი ტექსტის, ასევე ობიექტური კოდების დონეზე;

▲ მომხმარებლისა პროგრამებისათვის განკუთვნილი მუდმივი დამხსომებელი (***ROM***) მოწყობილობისათვის ***Flesh***-ტექნოლოგიაზე გადასვლა. უმრავლესი ტიპის მიკროკონტროლერებისათვის პროექტდება «კორპუსიდან კორპუსში» შეცვლის შესაძლებლობის მქონე ორი მოდელი. ეს მოდელები ფუნქციური შემადგენლობის მიხედვით ერთმანეთის სრული იდენტურებია და ერთმანეთისაგან ***ROM***-ში ინფორმაციის შეტანის ტექნოლოგიით განსხვავდება (ერთში გამოიყენება ***maskROM*** ხოლო მეორეში - ***Flesh*** ტექნოლოგია);

▲ პერიფერიული მოდულების ბიბლიოთეკა შეიცავს ინფორმაციის მიმდევრობით გამცვლელი კონტროლერების გაფართოებულ ნაკრებს; ფირმა «***Motorola***»-ს მიკროკონტროლერებისათვის სტანდარტული ასინქრონული (***SCI***) და სინქრონული (***SPI***) გაცვლის პორტების გარდა დამუშავებულია სამრეწველო ქსელებში ***CAN*** პროტოკოლით, აგრეთვე გამოთვლითი ტექნი-

კაში დღეს გავრცელებული ***USB*** სალტით მუშაობისათვის გამიზნული კონტროლერები;

▲ არსებითადაა გაუმჯობესებული მიკროკონტროლერის გამართვის შესაძლებლობა. ჩაშენებული მონიტორი და სპეციალური პორტი შესაძლებელს ხდის მართვის გამოყენებითი პროგრამები უშუალოდ დასამზადებელი პროდუქციის (საბოლოო ნაკეთობის) დაფაზე გავმართოთ ძვირადღირებული სქემური ემულატორების გამოუყენებლად;

▲ პროგრამებისათვის განკუთვნილი ***Flesh*** ტიპის მქონე მიკროკონტროლერში შეიძლება რეალიზებული იქნეს ***სისტემაში დაპროგრამების რეჟიმი***, რომლის დროსაც პროგრამა ნაკეთობის დაფაზე სტაციონარულად განთავსებულ მიკროკონტროლის მეხსიერებაში შეიტანება. პროგრამის კოდები პერსონალური კომპიუტერიდან მიმდევრობითი ინტერფეისით გადაიცემა;

▲ სპეციალური სქემოტექნიკური გადაწყვეტების გამოყენების მეშვეობით ამაღლებულია ელექტრომაგნიტური დაბრკოლებების პირობებსა და არახელსაყრელ გარემოში მომუშავე მიკროკონტროლერების ფუნქციონირების საიმედოობა.

1980 წლის ბოლოში ფირმა «***Motorola***»-მ დაამზადა ***PIC16C5x*** ტიპის მიკროკონტროლერი, რომელმაც დასაბამი მისცა ***PIC16*** ოჯახის ფართო გავრცელებას. მაღალი მწარმოებლურობის, მცირე მოხმარებული სიმძლავრისა და დაბალი ღირებულების გამო ***RISC***-არქიტექტურიანმა ამ ოჯახმა სერიოზული კონკურენცია გაუწია იმ პერიოდში გამოშვებად ***CISC***-არქიტექტურიან მიკროკონტროლერებს და ***სათავე დაუდო მარტივ ერთსიტყვიან ბრძანებათა სისტემის მქონე RISC-არქიტექტურას***. საბაზისო ***PIC16C5x*** ოჯახი მხოლოდ ***33*** ბრძანებას შეიცავს. ყველა ბრძანება, გარდა გადასვლის ბრძანებებისა, ერთ სამანქანო ციკლში სრულდება. ***20*** მეგაჰერცი სიხშირით ტაქტირების დროს ***PIC16C5x***–ს მწარმოებლურობაა ***5 MIPS.*** დღეისათვის ფირმა უშვებს ***RISC***-არქიტექტურიანი მიკროკონტროლერების შემდეგ ხუთ ოჯახს:

1) PIC15C5x ოჯახს, რომელშიც შედის მინიმალუროი რაოდენობის პერიფერიული მოწყობილობების მქონე იაფი მიკროკონტროლერები;

2) PIC12Cxxx ოჯახს, რომელშიც შედის მინიატურულ ***8***-გამომყვანიან კორპუსში მოთავსებულ მიკროკონტროლერებს, რომლებშიც ჩაშენებულია ტაქტური გენერატორი; აღნიშნული «მინიატურულობა» ამ ოჯახის ზოგიერთ წევრს ხელს არ უშლის გააჩნდეს ჩაშენებული ანალოგურ-ციფრული გარდამქმნელის ***8***-თანრიგიანი მოდული;

3) PIC16x/7x/8x/9x ოჯახს, რომელშიც შედის განვითარებული პერიფერიანი მიკროკონტროლერები. პერიფერიული მოდულების შემადგენლობაში შედის წატაცება/შედარების ოპციებიანი ტაიმერ-მთვლელები, განედურ-იმპუ-

ლსური მოდულატორები, ანალოგური კომპარატორები, ***ᲐᲪᲛ***, სხვადასხვა (მიმდევრობითი და პარალელური) მიმდევრობის ინტერფეისული კონტროლერები;

4) PIC17C4 x/5xx ოჯახს, რომელშიც შედის ბრძანებათა გაფართოებული სისტემისა და ვრცელი პერიფერიის მქონე მაღალმწარმოებლური მიკროკონტოლერები; ამ ოჯახის მიკროკონტროლერებს გააჩნია ჩაშენებული აპარატურული ***8x8*** მამრავლებელს, რომელიც გამრავლების ოპერაციას ერთ სამანქანო ციკლში ასრულებს;

5) PIC18Cxxx ოჯახს, რომელშიც შედის ***C***-კომპილატორის გამოყენებით ოპტიმიზებული ***RISC***-ბირთვის მქონე მიკროკონტროლერებს, რომელთა შიგა სალტის სიხშირე ***10*** მეგაჰერცის ტოლია.

1997 წელს ფირმა «***Atmel***»-მა წარმოადგინა ***AVR*** ოჯახის პირველი მიკროკონტროლერები. ***AVRAT90S*** ოჯახი აერთიანებს ჰარვარდულ მძლავრ ***RISC***-პროცესორებს, რომლებშიც პროგრამებისა და მონაცემების მეხსიერეერებასთან განცალკევებულად ხდება დაშვება, გამოიყენება საერთო დანიშნულების ***32***-თანრიგიანი რეგისტრები და ბრძანებათა განვითარებული სისტემა. ***AVR*** ოჯახის მომდევნო ვერსიათა ***ᲐᲚᲛ***-ის შემადგენლობაში შეიტანეს აპარატურული მამრავლებელი. ***AVR***-ის ბრძანებათა ბაზური ნაკრები შეიცავს ***120*** ინსტრუქციას. ბრძანებების უმრავლესობა სრულდება ერთ სამანქანო ციკლში და მთელი რიგი წევრების მწარმოებლურობა ***20 MIPS***–ის ტოლია. ***AVR***-ის პერიფერიაში შედის პარალელური პორტები, ტაიმერ-მთვლელები, სხვადასხვა მიმდევრობითი ინტერფეისები, ***ᲐᲪᲛ***, ანალოგური კომპარატორები. განასხვავებენ ***AVR***–ის შემდეგ სამ სერიას:

1) tiny AVR სერია, რომელიც მოიცავს ***8***-გამომყვანიან იაფ კორპუსში მოთავსებულ მიკროკონტროლერებს;

2) classic AVR სერია, რომელიც მთავარი სერაა. მასში გაერთიანებულია ***16 MIPS***-მდე მწარმოებლურობის მიკროკონტროლერები, რომლებშიც ***8*** კილობაიტამდე მოცულობის პროგრამები შეინახება ***Flesh*** მეხსიერებაში, ხოლო და ***128*** ბაიტიდან ***512*** ბაიტამდე მონაცემებისთვის რეალიზებულია სტატიკური ***RAM***;

3) mega AVR სერია. მასში გაერთიანებულია მიკროკონტროლერები, რომლებსაც აქვს დიდი (***128*** კილბაიტამდე) მოცულობის ***ფლეშ ტიპის*** მუდმივი დამხსომებელი (***Flesh ROM***) მოწყობილობა, ***4*** კილობაიტამდე მოცულობის ოპერატიული (***RAM***) მეხსიერება და რომელთა მწარმოებლურობა ***16 MIPS***-მდეა. აღნიშნული მიკროკონტროლერები გამოიყენება რთული ამოცანების გადასაწყვეტად.

XII თავი
ტრანსპორტზე მიკროპრო-ცესორული ტექნიკის გამოყენება

12.1. ზოგადი ცნობები

ასზე მეტი წლის წინათ ადამიანის მიერ აგებული ***სიგნალიზაციისა და ბლოკირების (სარკინიგზო ავტომატიკისა და ტელემექანიკის) რელეური მოწყობილობები*** საშუალებას გვაძლევდა არა მარტო აგვემაღლებინა მატარებელთა მოძრაობის სიჩქარე, არამედ თავიდან აგვეცილებინა ავარიები. მოძრაობის დაბალი ინტენსივობის დროს უსაფრთხოების საკმარისი დონის დაცვით მატარებელთა მოძრაობის მართვა მეისრეებსაც შეეძლო. მოძრაობის სიჩქარისა და ინტენსივობის ამაღლების შემდეგ მეისრის შესაძლებლობები არ აღმოჩნდა საკმარისი სწორი მოქმედებების დროულად განხორციელებისათვის. შექმნილი სიტუაციიდან გამოსვლის მიზნით დამუშავდა ზემოთ აღნიშნული მოწყობილობები.

დღეს მოწმენი ვართ იმისა, რომ მსგავსი მოწყობილობები გამოიყენება ყველა სხვა სახის ტრანსპორტის მართვისათვის.

საავიაციო ტრანსპორტის მოძრაობის მართვისათვის ზემოთ აღნიშნული რელეური მოწყობილოების ანალოგური მოწყობილობები გამოიყენება როგორც სააეროდომო ზონაში, ისე ტრასებზეც. ამ მოწყობილობათა თავისებურებაა ის, რომ საჰაერო ტრანსპორტი სამ განზომილებაშია განფენილი. ადამიანებს აუცილებლად სჭირდებათ ჰქონდეთ საკუთარი «მესამე განზომილება», რომელიც მათ გაუძლიერებს გადაწყვეტილების სწრაფი მიღების უნარს. რელეური მოწყობილობები ადამიანებს სწორედ ამ «მესამე განზომილების» შეძენაში ეხმარება. სარკინიგზო სიგნალიზაციისა და ბლოკირების მოწყობილობების გამოგონებიდან საუკუნეზე მეტი ხნის გასვლის შემდეგაც ამ მიმართულებით ადამიანის ბუნებრივი უნარი უმნიშვნელოდ გაიზარდა, ხოლო ტრანსპორტის ფუნქციონირების ალგორითმები მნიშვნელოვნად გართულდა. ამის გამო გადაწყვეტილების მიღების ამოცანა ასი წლის გავლის შემდეგ კიდევ უფრო აქტუალური გახდა. საბედნიეროდ თანამედროვე მიკროპროცესორული სისტემების სახის მქონე რელეური მოწყობილობების უდიდესი განვითარების წყალობით ადამიანს დღეს ამ ამოცანის გადასაწყვეტად გაცილებით მეტი შესაძლებლობები აქვს.

საავიაციო ხაზებზე მოძრაობის მართვის პრობლემის გადაწყვეტა გარდაუვლად უნდა ეფუძნებოდეს დამხმარე საშუალებებს, რომლებიც კრიტიკულ

პირობებში ამაღლებს ადამიანის უნარს, სწრაფად მიიღოს სწორი გადაწყვეტილება. გადაწყვეტილებათა შედეგების თანაბარი მნიშვნელობის დროს ზემოთ აღნიშნული საკმარისი არ არის. რომელიმე გადაწყვეტილების უშეცდომობაში დარწმუნების შემთხვევაშიც კი ადამიანი მაინც არ წყვეტს ყველა შესაძლო გადაწყვეტილებებიდან «საუკეთესოს» ძებნის პროცესს. სხვა სიტყვებით რომ ვთქვათ, ადამიანს მანამ არ დააკმაყოფილებს უსაფრთხო გადაწყვეტილებების კონკრეტული ნაკრები, სანამ არ დარწმუნდება, რომ იგი სხვა ნაკრებებზე უფრო ოპტიმალურია.

მიკროპროცესორული ტექნიკა ფართოდ გამოიყენება ***საავტომობილო ტრანსპორტ ზეც.*** განვიხილოთ აღნიშნული გამოყენების ზოგიერთი მაგალითი.

ავტომობილის ერთ-ერთ მთავარი ნაწილია ბენზინზე მომუშავე შიგაწვის ძრავა. მისი მუშაობისათვის საჭიროა ჰაერისა და საწვავი აირების გარკვეული პროპორციის ნარევის მომზადება. აღნიშნული ნარევის მომზადების პროცესს ***კარბურაცია***, ხოლო ამ პროცესის მარეალიზებელ მოწყობილობას – ***კარბურატორი*** ეწოდება. დამყარებული სიჩქარით მანქანის მოძრაობისას საწვავ აირთან ჰაერი ***15:1*** პროპორციით უნდა იყოს შერეული; გაცივებული ძრავას ამუშავებისათვის მოითხოვება უფრო მაღალი, კერძოდ. ***10:1***–ის ტოლი პროპორცია. თანაბარი მოძრაობისა და ძრავას ეფექტური მუშაობის უზრუნველყოფისათვის აუცილებელია ეფექტური კარბურაცია.

კარბურატორი, სამწუხაროს, ვერ უზრუნველყოფს სამუშაო რეჟიმების მთელ დიაპაზონში ოპტიმალური შემადგენლობის ნარევის მიწოდებას. ამას განაპირობებს კარბურატორისათვის დამახასიათებელი ისეთი საყოველთაოდ ცნობილი ნაკლი, როგორიცაა უქმი სვლისა და მცირე დატვირთვების დროს ძრავას მუშაობის დროს ნარევის შემადგენლობის უზრუნველყოფის დაბალი სიზუსტე. ამის შედეგად გვიხდება ნარევი ზედმეტად გავამდიდროთ, რადგან საწინააღმდეგო შემთხვევაში დაბალი ბრუნვების დროს ძრავა არამდგრადად იმუშავებს. კარბურატორში საწვავის მიწოდებას განსაზღვრავს ტივტივურ კამერაში მისი დონე, რომელზეც გავლენას ახდენს ბენზინის სიმკვრივე (სხვადასხვა ხარისხის ბენზინებისათვის იგი იცვლება ***0,7-0,78*** $\frac{გ}{სმ^3}$ ფარგლებში). ბენზინის სიმკვრივე თავის მხრივ მნიშვნელოვნადაა დამოკიდებული ტემპერატურაზე. ამიტომ კარბურატორის მუშაობა უნდა იცვლებოდეს ბენზინის ხარისხისა და ტემპერატურის ცვლილების კვალობაზე, რაც ჯერჯერობით შეუძლებელია. კამერაში ბენზინის დონე იცვლება მანქანის აჩქარებულად, მოსახვევებში, დაღმართზე, რყევებით მოძრაობისას, რაც ცვლის ცილინდრებში მის მიწოდებას; გარდა ამისა, ტემპერატურაზეა დამოკიდებული ჰაერის სიმკვრივეც, ე. ი. მასში არსებული ჟანგბადის რაოდენობაც.

მრავალი ათეული წელია ცდილობენ კარბურაცია შეცვალონ საწვავის მართულად შეფრქვევით. ***1939*** წელს იტალიაში ***Moto Guzzi***–ს მიერ შემოთავაზებული იქნა ***ელექტრული მართვით*** საწვავის შეფრქვევის სისტემა; ***1957*** წელს ფირმა ***Chrysler***–მა დაამუშავა ***ვაკუუმური მილაკებით*** აგებული საწვავის შეფრქვევის მმართველი სისტემა. საწვავის შეფრქვევის პროცესის მმართველი ***ტრანზისტორული*** სისტემები ***1967*** წელს დაამუშავა ***Volkswagen***–მა, ხოლო ***1971*** წელს ***Nissan***–მა.

ბენზინის ძრავებისათვის საწვავის შეფრქვევის მმართველი სისტემა, რომელიც უზრუნველყოფს ყველაზე მკაცრ მოთხოვნებს, მოხერხდა მიკროპროცესორული ტექნიკის გამოჩენის შემდეგ. დღეს ისინი გამოიყენება ყველა კლასის ავტომობილებში. ყველაზე მეტად გავრცელდა საწვავის განაწილებულად შემფრქვევი იმპულსური სისტემები, რომლებშიც ფრქვევანები თითოეული ცილინდრის მილყელებში სარქველების მახლობლად ყენდება.

ძრავას მართვის გარდა მიკროპროცესორები გამოიყენება ანტიბლოკირებისა და ანტიწაბუქსავების სისტემებში.

ანტიბლოკირების სისტემები მუდმივად ზომავს თვლების ბრუნვის კუთხური სიჩქარეს და დამუხრუჭებისას უზრუნველყოფს ყველა თვლის ერთნაირ კუთხური სიჩქარით ბრუნვას; აღნიშნული მაქსიმალურად ამცირებს დამუხრუჭების მანძილს და ექსტრენული დამუხრუჭების დროს ინარჩუნებს მანქანის მართვადობას.

ზემოთ აღნიშნული ფუნქციის მსგავს ფუნქციას ასრულებს ანტიწაბუქსავების სისტემები, რომლებიც წამყვან თვლებს არ აძლევს წაბუქსავების შესაძლებლობას. ეს განსაკუთრებით მნიშვნელოვანია წინაამძრავიან ავტომობილებში, რადგან ისინი უკანაამძრავიან მანქანებზე უფრო ადვილად კარგავს გზასთან შეჭიდულობას.

მიკროპროცესორული სისტემები გამოიყენება აგრეთვე: ***1)*** ფარებისა და ნათების მმართველ სისტემაში; ***2)*** შეჯახების მაფრთხილებელ სისტემაში; ***3)*** უსაფრთხოების ბალიშების მართვისათვის; ***4)*** იმობბილაიზერის (მანქანის გატაცების საწინააღმდეგო ელექტრონული მოწყობილობის ნაირსახეობის) ფუნქციონირებისათვის; ***5)*** საპროექციო დისპლეისათვის; ***6)*** მინის საწმენდების მართვისათვის; ***7)*** აუდოსისტემისათვის; ***8)*** ელექტრონული კომპასისათვის; ***9)*** ადაპტერული საკიდარის სიხისტის რეგულირებისათვის; ***10)*** დასაჯდომების მართვისათვის; ***11)*** საბურავებში წნევის მონიტორინგისა და ავტოდატუმბვის ორგანიზებისათვის; ***12)*** ტრანსმისიის მართვისათვის და ა.შ.

საინტერესოა იმ ფაქტის აღნიშვნა, რომ ავტომობილ «***Peugeot 206***»-ს ბორტზე ფუნქციონირებს ***27*** მიკროკონტროლერი, ხოლო ისეთი უმაღლესი კლასის ავტომობილში, როგორიცაა მეშვიდე სერიის «***BMV***», გამოყენებულია ***60***-ზე მეტი მიკროკონტროლერი.

რკინიგზებსა და საავიაციო ტრანსპორტზე მოძრაობის მართვისათვის რელეური მოწყობილობების გამოყენების შესახებ ზემოთ გამოთქმული მოსაზრებით ნათლად ჩანს, რომ რელეური მოწყობილობების ძირითადი დანიშნულებაა ადამიანს გადაწყვეტილების სწრაფად მიღებაში დაეხმაროს. მმართველი რელეური მოწყობილობების თითოეული გამოსასვლელი სიგნალი კონკრეტული მმართველობითი გადაწყვეტილების ინიცირებას იწვევს. მოწყობილობების შესასვლელებს მიეწოდება სიგნალების როგორც ცალკეული კომბინაციები, ისე კომბინაციათა გარკვეული მიმდევრობები. ცალკეული კომბინაცია განსაზღვრავს დროის მოცემულ მომენტში ფორმირებულ ხდომილობას, ხოლო მიმდევრობები – ამ ხდომილობის წარმოშობის ისტორიას. სხვადასხვა ისტორიული გზით ფორმირებულ ერთი და იმავე შესასვლელ სიგნალებს საჭიროების შემთხვევებში უნდა შეეძლოს სხვადასხვა გამოსასვლელი სიგნალების ფორმირება. სწორედ ეს გარემოება უნდა გაითვალისწინოს კონსტრუქტორმა მმართველი რელეური მოწყობილობათა აგების დროს. ეს იმას ნიშნავს, რომ ხშირად მმართველ რელეურ მოწყობილობას გარკვეული შესასვლელი კომბინაციის ფორმირების ისტორიის დამხსომებელი «მეხსიერებაც» უნდა გააჩნდეს.

მართვის მიკროპროცესორული სისტემების სინთეზისა და ანალიზის ზოგადი საკითხები მესამე თავში გვაქვს გადმოცემული. კონკრეტიზაციის მიზნით ამ თავში შედარებით დაწვრილებით განვიხილავთ ავტომატიკისა და ტელემექანიკის სასადგურე სისტემებში მიკრორროცესორული ტექნიკის გამოყენების ძირითად პრინციპებს.

12.2. სარკინიგზო ავტომატიკისა და ტელემექანიკის მიკროპროცესორულ სისტემებზე გადასვლის საფუძვლები

სარკინიგზო ავტომატიკის ტელემექანიკის სისტემების (***სატს***-ის) განვითარების ეტაპები იცვლებოდა მათ ასაგებად გამოყენებული საელემენტო ბაზისა და ამის საფუძველზე თავად სისტემების ასაგებად გამოყენებული საიმედოობისა და უსაფრთხოების მეთოდების ცვლილების კვალობაზე. განვითარების საუკუნეზე მეტი ხნის განმავლობაში გამოიყოფა აღნიშნული სისტემების განვითარების ხუთი ეტაპი: ***1) I ეტაპი*** (***1860-1900***) – მექანიკური სისტემების ეტაპი; ***2) II ეტაპი*** (***1900-1930***)- ელექტრომექანიკური სისტემების ეტაპი; ***3) III ეტაპი*** (***1930-1960***) - რელეური სისტემების ეტაპი; ***4) IV ეტაპი*** (***1960-1980***) - ნახევარგამტარული სისტემების ეტაპი, რომელიც მომდევნო ეტაპის ქვეეტაპად შეიძლება ჩავთვალოთ; ***5)V ეტაპი*** (***1980*** წლიდან დღემდე) - მიკროპროცესორული სისტემების ეტაპი.

სატს-ების აგებისათვის გამოყენებული რელეური საელემენტო ბაზიდან მიკროპროცესორულ საელემენტო ბაზაზე გადასვლის აუცილებლობას განაპირობებს აღნიშნული სისტემების განვითარების ოთხი ტენდენცია:

1. სისტემათა ფუნქციური სირთულის ამაღლება. რკინიგზების გამტარობისა და გადაზიდვების უნარის მუდმივი გაზრდისა და მატარებელთა მოძრაობის სიჩქარის ამაღლების აუცილებლობა ***სატს***-ებისაგან მოითხოვს ახალი ფუნქციების შესრულებას, მათი საექსპლუატაციო და ტექნიკური მახასიათებლების გაუმჯობესებას. ეს ფუნქციები გაცილებით რთულია, ვიდრე ის ფუნქციები, რომლებიც ადრე წყდებოდა რელეური სისტემების მიერ. მაგალითად, დღეისათვის უკვე საკმარისი არ არის მოძრაობის უსაფრთხოების პირობების დაცვით მარშრუტის გამზადება და შუქნიშნის გაღება, რომლებსაც წარმატებით ასრულებს ავტომატიკისა და ტელემექანიკის სასადგურე რელეური სისტემები. დღის წესრიგში დგება სხვადასხვა კრიტერიუმების გათვალისწინებით ისეთი ოპტიმალური მარშრუტების ამორჩევის ამოცანა, რომლის ფორმირების პროცესში გათვალისწინებული იქნება მატარებლის წონა და სიგრძე, სადგურში მატარებლის შემოსვლის დროის და ამ მატარებლის ნომრის ფიქსირება, მეზობელი სადგურებისათვის მატარებლის მახასიათებლების მიწოდება და ა. შ. სხვა სიტყვებით რომ ვთქვათ, ***სატს***-ების მიერ გადასაწყვეტი ***ლოგიკური ამოცანები*** თანდათანობით გარდაიქმნება ***გამოთვლით ამოცანებად***. უნივერსალური კომპიუტერებითა ან სქემური ხერხებით მათი გადაწყვეტა ვერ მოგვცემს საჭირო ეფექტს; ამას განაპირობებს ის გარემოება, რომ, ***ჯერ ერთი***, ძნელია როგორც აუცილებელი პროგრამების შექმნა, ასევე ***სცპ***-ს პირობებთან უნივერსალური კომპიუტერის შეგუების ამოცანის გადაწყვეტა, ***და მეორეც***, ასეთი მიდგომის შედეგად მიიღება ვეებერთელა და ძალიან ძვირი მოწყობილობები.

აღნიშნულს ადასტურებს ელექტრონული ცენტრალიზაციის შესაქმნელად დახარჯული ოცწლიანი პერიოდის განმავლობაში მიღებული გამოცდილება, რომლის დროსაც ვერ მოხერხდა პრაქტიკულად მისაღები სახის სისტეტემის აგება; წარუმატებლობის მთავარი მიზეზი იყო სისტემის ასაგებად აუცილებელი საელემენტო ბაზის არარსებობა. მიკროპროცესორების გამოყენებამ შესაძლებელი გახადა ამ ამოცანის პრაქტიკულ დონეზე გადაწყვეტა.

დღეისათვის, როდესაც მსოფლიოს მოწინავე ქვეყნებში უკვე ფუნქციონირებს მიკროპროცესორული ტექნიკის გამოყენებით აგებული სასადგურე და საგადასარბენო, ავტომატური სალოკომოტივო სიგნალიზაციის, მატარებელთა ავტომატური ტარების სისტემები და მოწყობილობები, გორაკის ავტომატიკის კომპლექსები, სრულიად ნათელი გახდა, რომ სწორედ ***მიკროპროცესორული ტექნიკა განსაზღვრავს სატს-ების შემდგომ განვითარებას.*** ასევე გამოიკვეთა, რომ მიკროპროცესორების გამოყენება მოითხოვს სარკინიგზო

ავტომატიკისა და ტელემექანიკის სისტემების აგებისათვის დღემდე არსებული მიდგომების გადაფასებას. აუცილებელია ამოცანების ძალიან ზუსტი დასმა, მათი გადაწყვეტების გზების ყველაზე ეფექტური გზების საქმის ცოდნით დასაბუთება და სათანადო რეალიზაცია.

2. სისტემების სტრუქტურული გართულება. სატს-ებზე დაკისრებული ფუნქციების რაოდენობის გაზრდა ართულებს მათ სტრუქტურას. სისტემის შემადგენელი ბლოკების რაოდენობა რამდენიმე ათეულჯერ იზრდება. სწორედ ასეთი პირობების დროს გვაძლევს მოგებას ზედიდი ინტეგრალური სქემების გამოყენება. ამას თუ დავუმატებთ იმას, რომ გამოვიყენებთ ფართო ფუნქციური შესაძლებლობების მქონე უნივერსალურ ელემენტებს, დარწმუნებულები უნდა ვიყოთ, რომ მივიღებთ განსაცვიფრებელ შედეგებს. სწორედ ასეთ ელემენტებს წარმოადგენს მიკროპროცესორები. მცირე გაბარიტებისა და დაბალი ღირებულების გამო სწორედ ***მიკროპროცესორებს შეუძლია წარმატებით მოახდინოს სატს-ების ფუნქციათა რეალიზაცია.***

3. საიმედობისადმი წაყენებული მოთხოვნების ამაღლება. ადამიანიდან ავტომატიკის საშუალებებისათვის გადაცემული ფუნქციების რაოდენობის გაზრდა ზრდის ***მტყუნების ფასს.*** მხედველობიდან არ უნდა გამოგვრჩეს ***სატს***-ების მომსახურების პროცესების შრომატევადობაც. მტყუნება, რომელიც არ წარმოქმნის საშიშ სიტუაციას, ხშირად იწვევს მატარებლების ხანგრძლივ დაყოვნებას. ბუნებრივია, რომ ამ სიტუაციაში სისტემის საიმედოობა უმნიშვნელოვანესი და აუცილებელი პირობაა. საიმედოობის ამაღლების ძირითადი გზაა – სისტემის სხვადასხვა დონეზე სიჭარბის შეტანა, რაც ხშირად აორმაგებს და ასამმაგებს აპარატურის მოცულობას. სრუფასოვნად სწორედ აქ ვლინდება მიკროპროცესორული ტექნიკის შესაძლებლობები.

4. კონტროლვარგისობისადმი წაყენებული მოთხოვნების ამაღლება. სისტემების გართულება და ახალი ინტეგრალური საელემენტო ბაზის გამოყენება ძალიან ართულებს მომსახურებასა და რემონტს. ამის გამო ახალი სისტემის დამუშავების ეტაპზევე უნდა მივიღოთ ზომები მისი მომსახურების გასამარტივებლად, მტყუნებების მოძებნისა და შეკეთების გასაადვილებლად, ე.ი. სისტემა უნდა გავხადოთ ***კონტროლვარგისი.*** ამ თვისებით სისტემის აღჭურვისთვისაც საჭირო ხდება მისი სიჭარბის გაზრდა. ზოგჯერ სულ რაღაც რამდენიმე ელემენტის დამატებაც კი მნიშვნელოვნად აადვილებს უწესივრობის მოძებნას, რაც აადვილებს მომსახურე პერსონალის საქმიანობასა და აუმჯობესებს სისტემის საექსპლუატაციო პირობებს.

სატს-ების განვითარების ზემოთ ჩამოთვლილი ტენდენციებისადმი მიკროპროცესორული სისტემების შესატყვისობას. უპირველეს ყოვლისა, განაპირობებს ***მიკროპროცესორების უნივერსალობა***, ე.ი. მათი უნარი სისტემის სტრუქტურის შეუცვლელად, მხოლოდ პროგრამული მეთოდების გამოყენე-

ბით გადაწყვიტოს მართვის პროცესში წარმოშობილი გამოთვლითი და ლოგიკური ამოცანები. თეორიული თვალსაზრისით უნივერსალობა გამომდინარეობს ფუნდამენტური პრინციპიდან: ***აპარატურული საშუალებებით რეალიზებადი ნებისმიერი პროცესი პროგრამულადაც შეიძლება იქნეს რეალიზებული და პირიქით.***

მიკროპროცესორული სისტემების მეორე უმნიშვნელოვანესი თვისებაა ***მოქნილობა***, ე. ი. ერთი ამოცანიდან მეორეზე პროგრამულად გადაწყობის შესაძლებლობა. სწორედ ამ თვისებაზეა დაფუძნებული ***მოქნილი ავტომატიზებული წარმოების*** განვითარება.

მაშასადამე, მიკროპროცესორების უნივერსალობა და მოქნილობა საშუალებას გვაძლევს, მნიშვნელოვანი დროითი და მატერიალური დანაკარგების გარეშე გადავწყვიტოთ სისტემის სტრუქტურული სირთულის ამაღლებით წარმოქმნილი პრობლემები. საიმედოობისა და კონტროლვარგისობის ამაღლებასთან დაკავშირებული პრობლემების ადვილად დაძლევის საშუალებასაც გვაძლევს მიკროპროცესორებისათვის დამახასიათებელი მაღალი საკუთარი საიმედოობა, მცირე გაბარიტები და ღირებულება. ეს მახასიათებლები მნიშვნელოვანი სიჭარბის მქონე მიკროპროცესორული სისტემების რეალურად აგების საშუალებას გვაძლევს, რაც შეესაბამება ***სპტს***-ებისადმი წაყენებულ ახალ მოთხოვნებს.

უნდა აღვნიშნოთ მიკროპროცესორული სისტემების კიდევ ერთი ღირსება – ***დამზადების მაღალი ტექნოლოგიურობა***. ამას განაპირობებს ის გარემოება, რომ სისტემის ასაგებად გამოყენებული სტანდარტული მოდულები, რომლებიც მაკომპლექტებელი ნაკეთობებია, თავის მხრივ ათეულობით ათასი ფუნქციური ელემენტებისაგან შედგება.

სხვადასხვა მოწყობილობებში მიკროპროცესორების დანერგვის ინტენსიურობა მათი ხელმისაწვდომობითაა განპირობებული. ისედაც იაფი სერიული მიკროპროცესორების ღირებულებას შემცირების ტენდენცია აქვს. თავისი განვითარების ***20*** წლის განმავლობაში წარმოქმნილია ასეთი ელემენტების ***5*** თაობა, რომლებშიც გაერთიანებული მიკროპროცესორების ინტეგრაციის ხარისხი და სწრაფმოქმედება პერმანენტულად იზრდება. აღვნიშნავთ, რომ კრისტალზე განთავსებულმა ელემენტების რაოდენობამ უკვე ***2*** მილიარდს გადააჭარბა, ხოლო წამში შესრულებული ოპერაციების რაოდენობა გადაცდა ***1012***-ს.

დღეისათვის მიკროპროცესორული ტექნიკის განვითარების ტემპები წინ უსწრებს დისკრეტული სისტემების განვითარებისა და ***სპტს***-ებისადმი ახალ მოთხოვნების წაყენების ტემპებს. სწორედ ეს წარმოადგენს მიკროპროცესორული საელემენტო ბაზის პერსპექტულობის მთავარ ნიშანს.

მიკროპროცესორულ სისტემათა ღირსებებზე ლაპარაკის დროს მხედველობიდან არ უნდა გამოგვრჩეს მათი შექმნის დროს წარმოშობილი სიძნელეებიც. უპირველეს ყოვლისა, უნდა აღვნიშნოთ, რომ აპარატურულ დისკრეტულ მოწყობილობებთან შედარებით მიკროპროცესორულ სისტემებში ***მცირდება გამოთვლების სიჩქარე***, რაც ოპერაციების პროგრამულად რეალიზაციითაა გამოწვეული. აპარატურული რეალიზაციის დროს არ არსებობს უნივერსალობა და ყოველი ფუნქციის შესასრულებლად სპეციალური ლოგიკური სქემა აიგება. პროგრამული რეალიზაციის დროს გამოთვლების პროცესი იყოფა მთელ რიგ მიკრონოპერაციებად და ისინი დროში მიმდევრობით რეალიზდება. გარდა ამისა, ერთი და იგივე ამოცანა შეიძლება სხვადასხვა პროგრამით იქნეს გადაწყვეტილი.

სწრაფმოქმედების შემცირებამ დროის რეალურ მასშტაბში მომუშავე მართვის სისტემებს მაშინ შეიძლება შეუქმნას სიძნელე, როდესაც მართვის სიგნალების ფორმირებაზე უფრო სწრაფად თავად მართვის პროცესი მიმდინარეობს. ვინაიდან სარკინიგზო ტრანსპორტზე მიმდინარე ტექნოლოგიური პროცესები საკმაოდ ნელია, ამიტომ, ასეთი შეზღუდვა არ წარმოადგენს განმსაზღვრელს. იგი შეიძლება დავძლიოთ პარალელური გამოთვლების ორგანიზებითა და ზოგიერთი ფუნქციების აპარატურულად შესრულებით (ასეთი ტენდენცია დამახასიათებელია თანამედროვე კომპიუტერული სისტემებისათვისაც).

მიკროპროცესორულ ტექნიკაზე გადასვლისას წარმოშობილი მეორე პრობლემაა რეალურ პირობებში მომუშავე ***მიკროპროცესორული სისტემების დაბრკოლებამდგრადობის უზრუნველყოფა.*** ეს პრობლემა არ არის ახალი და ნებისმიერი მიკროელექტრონული სისტემისთვისაა დამახასიათებელი. იგი განსაკუთრებით გამწვავდა მიკროელექტრონული და რელეურ-კონტაქტური მოწყობილობების ერთობლივი გამოყენების დროს. ამ მიმართულებით ხანგრძლივი მუშაობის პერიოდში დამუშავებული იქნა რთულ ელექტრომაგნიტურ გარემოში ფუნქციონირებადი მიკროპროცესორული სისტემების დაბრკოლებამდგრადობის უზრუნველყოფის მრავალი ხერხი და საშუალება.

კიდევ ერთ სიძნელეს წარმოადგენს მიკროპროცესორული სისტემების მომსახურება, მათში წარმოშობილი უწესივრობების პოვნა და შეკეთება. ამის გადაწყვეტისა გზაა კონტროლებავარგისი სისტემების შექმნა, ბლოკური გაფორმება და მტყუნებათა ინდიკაცია.

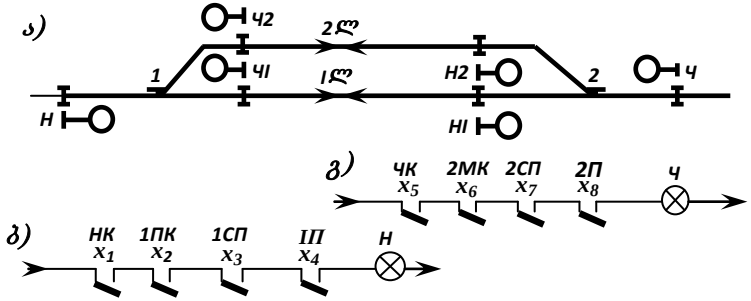
მიკროპროცესორული სისტემების განმასხვავებელი ნიშანია ის, რომ პროგრამული უზრუნველყოფა დიდ ზეგავლენას ახდენს სისტემათა სირთულესა და საიმედოობაზე. ძალიან რთულია დიდ სისტემათა პროგრამები, რაც მოითხოვს, რომ ამ სისტემათა შემქმნელებს დაპროექტების, გამართვისა და ტესტირების სფეროში ჰქონდეთ საკმაოდ მაღალი გამოცდილება.

12.3. ელექტრული ცენტრალიზაციის აგების საკითხები

ელექტრული ცენტრალიზაცია სპეციფიკური ლოგიკური (ორობითი) ფუნქციების მარეალიზებელი რთული დისკრეტული სისტემაა. აღნიშნული ფუნქციები მათემატიკურად ჩაიწერება ლოგიკური გამოსახულებების მეშვეობით.

ლოგიკური გამოსახულებები ლოგიკის ალგებრის ფორმულებია, რომელთა რეალიზება შესაძლებელია დისკრეტული ელემენტებით (რელეებით, ტრანზისტორებით და ა.შ.) აგებული სქემების ან ბინარული პროგრამების საშუალებით. პირველ შემთხვევაში საქმე გვაქვს ლოგიკური გამოსახულებების აპარატურულ, ხოლო მეორე შემთხვევაში – პროგრამულ რეალიზაციასთან.

რელეური ელექტრული ცენტრალიზაციის შემთხვევაში გამოიყენება ლოგიკური გამოსახულებების ***აპარატურული რეალიზაცია***, ხოლო მიკროპროცესორული ცენტრალიზაციის შემთხვევაში – ***პროგრამული რეალიზაცია***.



ნახ. 12.1. *შესასვლელი შუქნიშნების გაღების ალგორითმის აპარატურულად რეალიზაციის მაილუსტრირებელი სქემა*

განვიხილოთ ***12.1,ა*** ნახაზზე მოცემული პირობითი სადგურის შესასვლელი შუქნიშნების ***Н*** და ***Ч*** შუქნიშნის მართვის გამარტივებული სქემები. ნორმალურად აღნიშნული შუქნიშნები დახურულია. დავუშვათ, რომ აუცილებელია გამზადდეს ***Н*** შუქნიშნით ***1П***, ხოლო ***Ч*** შუქნიშნით - ***2П*** ლიანდაგზე მიღების მარშრუტი. პირველ შემთხვევაში ისარი ***1*** უნდა გადავიყვანოთ პლუსოვან, ხოლო მეორე შემთხვევაში ისარი ***2*** - მინუსოვან მდებარეობაში და შემდეგ შესაბამისად გავაღოთ ***Н*** და ***Ч*** შუქნიშანი. აღნიშნული შუქნიშნების გასაღებად საჭიროა შემოწმდეს პირობები, რომლებიც ***12.1*** ცხრილშია მოყვანილი. თუ დავუშვებთ, რომ $x_i,\ i = 1,2,\ldots,8$ პირობის შეუსრულებლობის

შემთხვევაში $x_i = 0$, ხოლო მისი შესრულების შემთხვევაში - $x_i = 1$, მაშინ x_i აღნიშვნები *ლოგიკურ ცვლადებად* გადაიქცევა, ე. ი. $x_i \in (0,1)$

შესასვლელი **Н** შუქნიშნის გასაღებად ერთდროულად უნდა შესრულდეს 12.1 ცხრილში მოყვანილი ყველა $x_1 - x_4$ პირობა; აქედან გამომდინარე, შესასვლელი **Н** შუქნიშნის გაღების ლოგიკური (ორობითი) $F_Н$ ფუნქცია წარმოადგენს შესაბამისი ლოგიკური x_i, $i = 1,2,3,4$ ცვლადების კონიუნქციას (ლოგიკურ ნამრავლს):

$$F_Н = x_1 x_2 x_3 x_4 \qquad x_i \in (0,1), \quad i = 1,2,3,4; \qquad (12.1)$$

ანალოგიურად, შესასვლელი **Ч** შუქნიშნის გაღების ლოგიკურ $F_Ч$ ფუნქციას ექნება შემდეგი სახე:

$$F_Ч = x_5 x_6 x_7 x_8, \qquad x_i \in (0,1), \quad i = 5,6,7,8. \qquad (12.2)$$

$F_Н$ და $F_Ч$ ფუნქციები ლოგიკური ფუნქციებია, ე.ი. $F_Н, F_Ч \in (0,1)$. კერძოდ, $F_Н = 0$-ის შემთხვევაში **Н** შუქნიშანი დახურულია, ხოლო $F_Н = 1$-ის შემთხვევაში - ღიაა. ანალოგიურად, თუ **Ч** შუქნიშნი დახურულია, მაშინ $F_Ч = 0$, ხოლო თუ ღიაა - $F_Ч = 1$.

ცხრ.12. 1. **Н** და **Ч** შუქნიშნების გასაღებად შესამოწმებელი პირობები

აღნიშვნა	აღნიშვნის შინაარსი	აღნიშვნა	აღნიშვნის შინაარსი
x_1	გაცემულია **Н** შუქნიშნის გაღების ბრძანება	x_5	გაცემულია **Ч** შუქნიშნის გაღების ბრძანება
x_2	ისარი 1 პლუსოვან მდებარეობაშია	x_6	ისარი 2 მინუსოვან მდებარეობაშია
x_3	**1СП** უბანი თავისუფალია	x_7	**2СП** უბანი თავისუფალია
x_4	**1Л** ლიანდაგი თავისუფალია	x_8	**2Л** ლიანდაგი თავისუფალია

ლოგიკური $F_Н$ და $F_Ч$ ფუნქციების შეიძლება რეალიზდეს აპარატურულად ან პროგრამულად; ამასთანავე მათი აპარატურული რეალიზება ხდება რელეურ, ხოლო პროგრამული რეალიზება – მიკროპროცესორულ ცენტრალიზაციებში. ცალ-ცალკე განვიხილოთ თითოეული შემთხვევა

აპარატურული რეალიზაცია რელეური ელექტრული ცენტრალიზაციის შემთხვევაში. რელეურ ელექტრულ ცენტრალიზაციაში (*12.1*) და (*12.2*) გამოსახულებებში შემავალ x_i ცვლადებს შეესაბამება რელეთა ფრონტული კონტაქტები, რომელთა დასახელებები *12.2* ცხრილშია მოყვანილი.

ლოგიკური $F_Н$ ფუნქციის აპარატურულად მარეალიზებელი სქემა მიიღება $x_1 \ldots x_4$ ცვლადების შესაბამისი კონტაქტების მიმდევრობით შეერთებით (ნახ.*12,ბ*), ხოლო $F_Ч$ ფუნქციის აპარატურულად მარეალიზებელი სქემა - $x_5 \ldots x_8$ ცვლადების შესაბამისი კონტაქტების მიმდევრობით შეერთებით (ნახ.*12.1,გ*). საბოლოოდ შეიძლება დავასკვნათ, რომ (*12.1*) და (*12.2*) ფუნ-

ქციების აპარატურული რეალიზების შედეგები შესაბამისად ***12.1,ბ*** და ***12.1,გ*** ნახაზებზეა მოყვანილი.

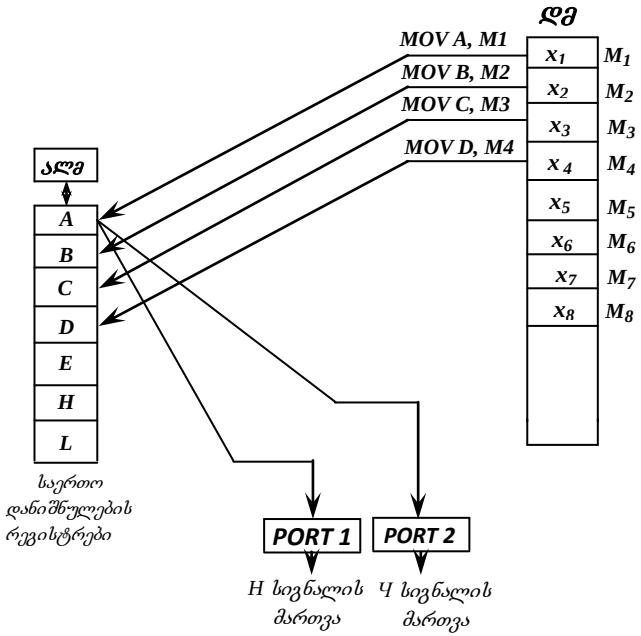
ცხრ. 12.2. *ლოგიკური* x_i, $i=1,\ldots,8$ *ცვლადების კონტაქტებით რეალიზება*

ლოგიკური ცვლადები	ლოგიკური ცვლადების მარეალიზებელი კონტაქტები
x_1	***Н*** შუქნიშნის ღილაკური ***НК*** რელეს ფრონტული კონტაქტი
x_2	საკონტროლო ***1ПК*** რელეს ფრონტული კონტაქტი
x_3	სალიანდაგო ***1СП*** რელეს ფრონტული კონტაქტი
x_4	სალიანდაგო ***1П*** რელეს ფრონტული კონტაქტი
x_5	***Ч*** შუქნიშნის ღილაკური ***ЧК*** რელეს ფრონტული კონტაქტი
x_6	საკონტროლო ***2МК*** რელეს ფრონტული კონტაქტი
x_7	სალიანდაგო ***2СП*** რელეს ფრონტული კონტაქტი
x_8	სალიანდაგო ***2П*** რელეს ფრონტული კონტაქტი

პროგრამული რეალიზაცია მიკროპროცესორული ელექტრული ცენტრალიზაციის შემთხვევაში. მიკროპროცესორულ სისტემებში გამოიყენება ელექტრული ცენტრალიზაციის ალგორითმების პროგრამული რეალიზაციის დროს გამოიყენება ლოგიკური ფუნქციების უშუალოდ გამოთვლისა და ბინარული დაპროგრამების მეთოდი. განვიხილოთ ორივე მათგანი.

1. ლოგიკური ფუნქციების უშუალოდ გამოთვლის მეთოდი. მიკროპროცესორად გამოვიყენოთ გამოვიყენოთ ფირმა ***Intel***-ის მიერ დამუშავებულ ***8080/8085*** ოჯახის მიკროპროცესორი, რომლებიც კლასიკური სახის მიკროპროცესორადაა მიჩნეული. იგი შედგება შემდეგი ***ოთხი ბლოკისაგან***: ***1)*** დამხსომებელი მოწყობილობა; მასში ინახება შესასვლელი მონაცემები და გამოთვლათა შედეგები; ***2)*** საერთო დანიშნულების რვათანრიგიანი ***A,B,C, E,H,L*** რეგისტრები; მათში დროებით ინახება ის მონაცემები (ოპერანდები), რომლებზეც ხორციელდება არითმეტიკული და ლოგიკური ოპერაციები; აღნიშნული რეგისტრებიდან ***A*** წარმოადგენს ერთმაგ რეგისტრს (შეუძლებელია რომელიმე სხვა რეგისტრთან მისი გაერთიანება) და მას ***აკუმულატორი*** ეწოდება; ორ ოპერანდზე არითმეტიკული ან ლოგიკური ოპერაციის ჩატარებისას ერთ-ერთი ოპერანდი და აღნიშნული ოპერაციის შედეგი აუცილებლად ***A*** აკუმულატორშია ჩაწერილი. დანარჩენი რეგისტრებიდან შეწყვილებადია (***B,C***), (***D,E***) და (***H,L***) რეგისტრები, რის შედეგადაც შესაძლებელია ***BC, DE*** და ***HL*** სახელწოდების თექვსმეტთანრიგიანი რეგისტრების წარმოქმნა ***3)*** არითმეტიკულ-ლოგიკური მოწყობილობა (***АЛМ***), რომელშიც სრულდება არითმეტიკული და ლოგიკური ოპერაციები; ***4)*** ინფორმაციის შეტანისა და გამოტანისათვის განკუთვნილი რეგისტრები, რომლებსაც პორ-

ტები ეწოდება და პირობითად აღინიშნება როგორც ***PORT i, i=1,2,…; 8080/8085*** მიკროპროცესორის ბრძანებათა სისტემა [3]-შია მოყვანილი.



ნახ.12.2. *მონაცემების შენახვისა და გადაგზავნის სქემა*

განსახილველი მეთოდის არსი ლოგიკური (***12.1***) და (***12.2***) ფუნქციების გამოთვლის მაგალითზე განვიხილოთ. ***12.2*** ნახაზზე წარმოდგენილია მიკროპროცესორულ სისტემაში მონაცემების შენახვისა და გადაგზავნის სქემა. მასზე ნაჩვენებია გამოთვლის პროცესში მონაწილე ზემოთ აღნიშნული ოთხივე ძირითადი ბლოკი: დამხსომებელი ***დმ*** მოწყობილობა, ***ალმ***, საერთო დანიშნულების რეგისტრები და პორტები.

12.3. ცხრილში მოყვანილია F_H და $F_Ч$ ფუნქციების გამოთვლის პროგრამა. ***H*** და ***Ч*** შუქნიშნების გაღების პირობების განმსაზღვრელი შესასვლელი $x_1,\ldots,x_8$ ცვლადები შეინახება ***დმ***-ს უჯრედებში, რომელთა მისამართებია $M_1,\ldots, M_8$. პირველი რვა ბრძანება გამოითვლის F_H ფუნქციას. № ***1,...,4*** ბრძანებით $x_1,\ldots,x_4$ ცვლადები ***დმ***-დან შეიტანება საერთო დანიშნულების რეგისტრებში. №***5,6,7*** ბრძანებები ასრულებენ აღნიშნული ცვლადების ლოგიკურ გამრავლებას, რის შედეგადაც ფორმირდება $F_H = x_1x_2x_3x_4$ ფუნქცია. №***8*** ბრძანება F_H ფუნქციის მნიშვნელობას გადააგზავნის გამოსასვლელ რე-

გისტრ (პორტ) ***PORT 1***-ში. ამ რეგისტრის მდგომარეობაზე (F_H =*0* ან *1*) დამოკიდებულებით იმართება ***H*** შუქნიშანი. №***9,...,16*** ბრძანებებით ანალოგურად გამოითვლება $F_ч$ ფუნქცია.

ცხრ. 12.3. F_H და $F_ч$ *ფუნქციების გამოთვლის პროგრამა*

ბრძანების №	ბ რ ძ ა ნ ე ბ ე ბ ი	ბრძანების მნემონიკა
1	შიგთავსი გადაიგზავნოს ***M1*** უჯრედიდან ***A*** რეგისტრში	***MOV A, M1***
2	იგივე ***M2*** უჯრედიდან ***B*** რეგისტრში	***MOV B, M2***
3	იგივე ***M3*** უჯრედიდან ***C*** რეგისტრში	***MOV C, M3***
4	იგივე ***M4*** უჯრედიდან ***D*** რეგისტრში	***MOV D, M4***
5	გადამრავლდეს ***A*** და ***B*** რეგისტრების შიგთავსები (x_1x_2)	***ANA B***
6	იმავე ***A*** და ***C*** რეგისტრების შიგთავსები ($x_1x_2x_3$)	***ANA C***
7	იმავე ***A*** და ***D*** რეგისტრების შიგთავსები ($f_N=(x_1x_2x_3x_4)$)	***ANA D***
8	f_N-ის მნიშვნელობა გატანილი იქნას ***.PORT1*** -ში	***OUT PORT1***
9	***M5*** უჯრედიდან შიგთავსი გადაიგზავნოს ***A*** რეგისტრში	***MOV A, M5***
10	***M6*** უჯრედიდან შიგთავსი გადაიგზავნოს ***B*** რეგისტრში	***MOV B, M6***
11	***M7*** უჯრედიდან შიგთავსი გადაიგზავნოს ***C*** რეგისტრში	***MOV C, M7***
12	***M8*** უჯრედიდან შიგთავსი გადაიგზავნოს ***D*** რეგისტრში	***MOV D, M8***
13	გადამრავლდეს ***A*** და ***B*** რეგისტრების შიგთავსები (x_5x_6)	***ANA B***
14	იმავე ***A*** და ***C*** რეგისტრების შიგთავსები ($x_5x_6x_7$)	***ANA C***
15	იმავე ***A*** და ***D*** რეგისტრების შიგთავსები ($f_ч=(x_5x_6x_7x_8)$)	***ANA D***
16	f_N-ის მნიშვნელობა გატანილი იქნას ***PORT2***-ში	***OUT PORT2***

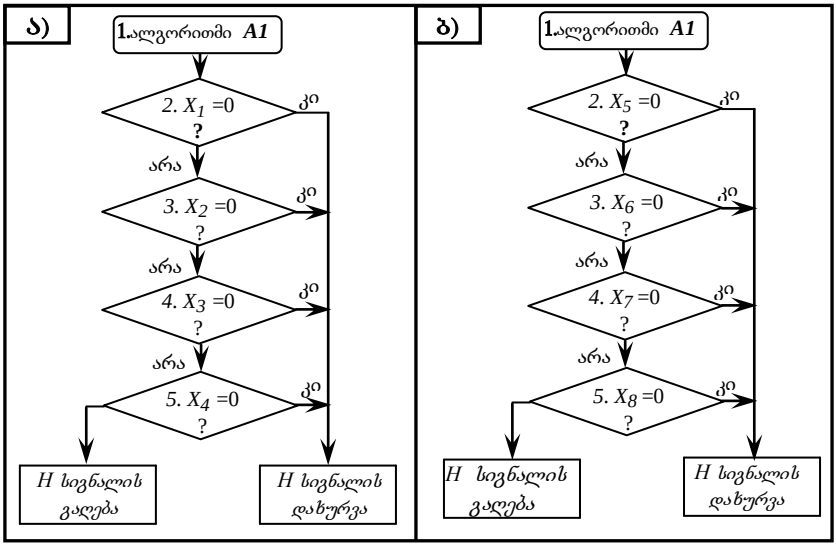
2. ბინარული დაპროგრამების მეთოდი. ბინარული დაპროგრამების მეთოდი დაფუძნებულია იმაზე, რომ ბულის ფუნქციის გამოთვლა დაიყვანება შემდეგი პირობითი გადასვლის ბრძანების მიმდევრობით შესრულებაზე:

i:თუ A, მაშინ j, საწინააღმდეგო შემთხვევაში i+1, სადაც i ბრძანების ნომერია.

12.2 ნახაზზე მოყვანილი (***12.1***) და (***12.2***) ფუნქციების გამომთვლელი ბინარული პროგრამების ბლოკ-სქემები.

უშუალო გამოთვლების მეთოდი გამოირჩევა სიმარტივით და თვალსანოებით. პროგრამების მოცულობა და მათი დამუშავების ხანგრძლივობა კონკრეტული ლოგიკური გამოსახულების სირთულით განისაზღვრება. ***ბინარული პროგრამების მეთოდის შესახებ*** შედარებით დაწვრილებით ვილაპარაკებთ მომდევნო პარაგრაფში.

ზოგადად, მიკროპროცესორული ცენტრალიზაციის პროგრამული უზრუნველყოფა წარმოადგენს ამოცანების მთელი კომპლექსის გადამწყვეტ რთულ პროდუქტს. ეს ამოცანები განისაზღვრება სადგურებზე მატარებლების მოძრაობის რთული ტექნოლოგიური პროცესის თავისებურებებით; აღნიშნული



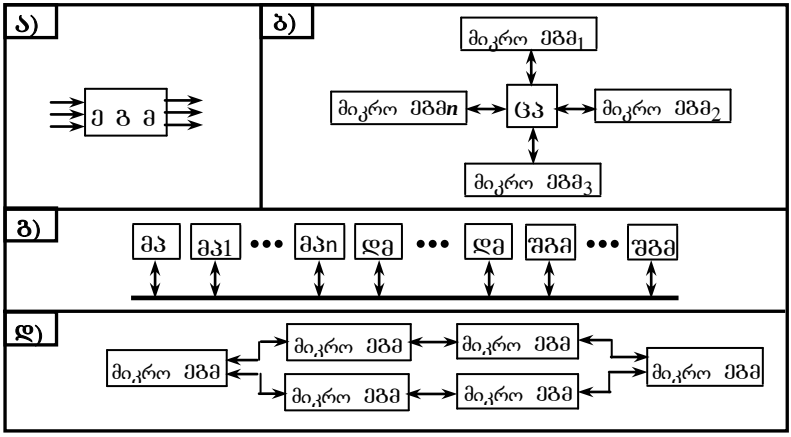
ნახ.12.2. ***H*** *და* ***Ч*** *შუქნიშნების მართვის ბინარულ პროგრამათა ბლოკ-სქემები*

პროცესი საპასუხისმგებლო ასინქრონულ პარალელურ პროცესს წარმოადგენს. სადგურებზე სამატარებლო ერთეულები ერთმანეთისაგან დამოუკიდებლად დროსა და სივრცეში პარალელურად გადაადგილდება (ე.ი. გადადაგილებათა სინქრონიზება არ ხდება). აუცილებელია ერთდროულად დამუშავდეს ინფორმაციები რამდენიმე მარშრუტის შესახებ; ამიტომ შეიძლება გამოიყოს მიკროპროცესორული ცენტრალიზაციის პროგრამული უზრუნველყოფის შემდეგი სამი ამოცანა: ***1)*** ელექტრული ცენტრალიზაციის ტექნოლოგიური ალგორითმების რეალიზაცია; ***2)*** პარალელური გამოთვლების ორგანიზება; ***3)*** პარალელური გამოთვლების ორგანიზება.

პარალელური გამოთვლების ორგანიზებისათვის მმართველ გამოთვლით კომპლექსში გამოიყენება ინფორმაციის მიმდევრობითი, ფუნქციონალური, კონვეიერული და მულტიპროცესორული დამუშავება.

მიმდევრობითი დამუშავებისას კომპლექსში ერთი პროცესორია, რომელიც პროცესებს ფაქტობრივად დროში მიმდევრობით (რიგის მიხედვით) ამუშავებს; ეს შესაძლებელია, თუ გამოთვლების სიჩქარე თავად ტექნოლოგიური

პროცესის (მაგალითად, მატარებლების მოძრაობის) მონაცემების ცვლილების სიჩქარეს მნიშვნელოვნად აღემატება. ასეთ შემთხვევაში იქმნება პარალელური გამოთვლების იღუზია. მაგალითად, ამგვარადაა აგებული სადგურზე შესასვლელი შუქნიშნების გაღების პირობების განსაზღვრის ზემოთ მოყვანილი პროგრამა (იხ. ცხრ*12.3*); პირველი რვა ბრძანებით გამოითვლება f_N ფუნქცია, ხოლო ბოლო რვა ბრძანებით — $f_Ч$ ფუნქცია.



ნახ. 12.3. მიკროპროცესორული ცენტრალიზაციების სტრუქტურული სქემები

ფუნქციური დამუშავების დროს მმართველ გამოთვლით კომპლექსში არსებობს რამდენიმე ფუნქციის ერთდროულად შემსრულებელი და ინფორმაციის ურთიერთგამცვლელი მოწყობილობები. ასეთი მიდგომაა გამოყენებული, მაგალითად, ***12.2*** ნახაზზე მოყვანილი ალგორითმების ბლოკ-სქემების რეალიზებისას; მოცემულ შემთხვევაში ***A1*** ალგორითმი რეალიზდება პროცესორში, რომელიც მატარებლების მოძრაობას მართავს სადგურის კენტ ყელში (ნახაზი ***12.1***), ხოლო ***A2*** ალგორითმი – სადგურის ლუწი ყელის პროცესორში.

კონვეიერული დამუშავებისას გამოთვლითი პროცესი იყოფა რამდენიმე ეტაპად და თითოეული მათგანი პარალელურ-მიმდევრობითად რეალიზდება სხვადასხვა ***პროცესორში*** (კონვეიერის პრინციპის მიხედვით).

მულტიპროცესორული დამუშავება ხორციელდება საერთო სალტეებისა და საერთო მეხსიერების მქონე მრავალი პროცესორით.

მიკროპროცესორული ცენტრალიზაციის დამუშავების დროს ინფორმაციის დამუშავების მითითებული ხერხების მარეალიზებელი სტრუქტურული სქემები ***12.3*** ნახზზეა მოყვანილი. მიმდევრობითი გამოთვლა ხდება ***ერთპროცესორულ სისტემაში*** (ნახ. ***12.3,ა***). იგი გამოიყენება დიდი კომპიუტერით

აღჭურვილ მსხვილ სადგურებში, ან ისეთ მცირე სადგურებში, სადაც ერთი მიკროკომპიუტერის გამოყენებაც საკმარისია. ***რადიალური სტრუქტურის სისტემაში*** (ნახ. ***12.3,ბ***) რეალიზდება ფუნქციური დამუშავება. თითოეული მიკროკომპიუტერი მართავს სადგურის გარკვეულ რაიონს. რაიონულ კომპიუტერებს შორის კავშირი მმართველი ცენტრალური ***ცპ*** პროცერსორის მეშვეობით მყარდება. ***მაგისტრალური სტრუქტურის სისტემაში*** (ნახ.***12.3,გ***) ინფორმაცია მულტიპლექსურად მუშავდება. სისტემის ელემენტები (***მპ*** მიკროპროცესორები, დამხსომებელი ***დმ*** მოწყობილობები, შეტანა-გამოტანის ***შგმ*** მოწყობილობები) საერთო მაგისტრალთანაა (სალტესთანაა) მიერთებული. ყველა ელემენტის მუშაობის რეგლამენტირებას ახდენს მმართველი ***მპ*** პროცესორი. ***ქსელური სტრუქტურის სისტემაში*** (ნახ.***12.3,დ***) რაიონული მიკროკომპიუტერები ინფორმაციას ერთმანეთს კონვეიერის პრინციპის მიხედვით უცვლის. მიკროკომპიუტერები სადგურზე გეოგრაფიული პრინციპითაა განთავსებული.

თითოეულ განხილულ სისტემას აქვს საკუთარი ღირსებებიცა და ნაკლოვანებებიც. მათი შეფასება შეიძლება პროგრამული უზრუნველყოვის სირთულის, საიმედოობისა და სწრაფმოქმედების ნიშნების მიხედვით. ყველაზე მარტივი პროგრამული უზრუნველყოფა აქვს ***ერთპროცესორულ და ქსელურ სისტემებს***. პირველ შემთხვევაში საჭირო არ არის სხვადასხვა მიკროკომპიუტერებს შორის ურთიერთზემოქმედების პრობლემის გადაწყვეტა, ხოლო მეორე შემთხვევაში ეს ურთიერთზემოქმედებები მარტივია – თითოეულ მათგანს მეზობელ მიკროკომპიუტერისათვის ინფორმაციის გადაცემა მოეთხოვება. საიმედოობის თვალსაზრისით საუკეთესო თვისებებისაა ***ქსელური სტრუქტურა.*** მასში არსებული ერთი რაიონული მიკროკომპიუტერის მტყუნება გამორიცხავს ყველა დანარჩენ რაიონში მარშრუტების დაყენების შესაძლებლობას. ***რადიალურ და მაგისტრალურ სტრუქტურებში*** სისტემის მუშაობა ირღვევა მმართველი პროცესორის მტყუნებების ან მაგისტრალის დაზიანების დროს. ყველაზე სწრაფმოქმედია ***ქსელური სისტემა***, რადგან მასში რეალიზებულია ინფორმაციის დამუშავების არა მარტო კონვეიერული, არამედ ფუნქციური პრინციპიც. სადგურის სხვადასხვა რაიონში მარშრუტებს ერთდროულად ამუშავებს სხვადასხვა მიკროკომპიუტერი. ყველაზე ნელმომქმედია ***ერთპროცესორული*** (მასში მარშრუტები დროში მიმდევრობით მუშავდება) და ***მაგისტრალური სისტემები*** (მაგისტრალის შეზღუდული გამტარობის უნარის გამო).

12.4. ბინარული პროგრამის ბლოკ-სქემის აგების ფორმალიზებული მეთოდი [6]

(დამუშავებულია სახელმძღვანელოს ავტორის მიერ 2005 წელს)

ბინარული დაპროგრამების მეთოდი წარმატებით გამოიყენება ლოგიკური ფუნქციების პროგრამული რეალიზებისათვის. იგი გამოირჩევა მაღალი სწრაფმოქმედებით, რადგან მისი გამოყენების დროს საშუალედო მონაცემების შენახვა საჭირო არ არის. ეს ხელს უწყობს ოპერატიული მეხსიერების განტვირთვას, რაც მეტად მნიშვნელოვანია არაერთ სპეციალიზებულ მიკროპროცესორულ სისტემებში ამ მეხსიერებათა მოკრძალებული მოცულობების გამო. მოცემული მეთოდის ერთ-ერთ ნაკლად უნდა ჩაითვალოს ბინარული პროგრამის ბლოკ-სქემის აგების ფორმალური მეთოდის არარსებობა, რაც ართულებს აღნიშნული ბლოკ-სქემის აგების პროცესს.

სარკინიგზო ავტომატიკისა და ტელემექანიკის მიკროპროცესორული სისტემების მიერ რეალიზებული ფუნქციების აბსოლუტური უმრავლესობა ლოგიკური ფუნქციებია. აღნიშნულიდან გამომდინარე მათი რეალიზებისათვის სწორედ დაპროგრამების ბინარული მეთოდის გამოყენებაა მიზანშეწონილი. ამ მეთოდის ზემოთ აღნიშნული ნაკლისა აღმოფხვრის მიზნით ჩვენ მიერ დამუშავებული და სანკტ-პეტერბურგის მიმოსვლის გზათა სახელმწიფო უნივერსიტეტის სამეცნიერო შრომების კრებულში ***2005*** წელს გამოქვეყნებული იქნა ***ბინარული პროგრამის ბლოკ-სქემის აგების ფორმალური მეთოდი***. აღნიშნული მეთოდის არსი ასეთია: ***თავდაპირველად აიგება განსაზღვრული ლოგიკური ფუნქციის ბინარული დაშლის გრაფი, რომლიდანაც ფორმალური მეთოდის გამოყენებით მიიღება აღიშნული ფუნქციის ბინარული პროგრამის ბლოკ-სქემა***.

ვიმედოვნებთ, რომ ჩვენ მიერ შემოთავაზებული მეთოდის გამოყენება მნიშვნელოვნად გაუადვილებს მუშაობას სარკინიგზო ავტომატიკისა და ტელემექანიკის მიკროპროცესორული მოწყობილობების დამპროექტებლებს. აღნიშნულიდან გამომდინარე, მიზანშეწონილად ჩავთვალოთ, მოცემულ სახელმძღვანელოში ქართულ ენაზე მოგვეყანა მისი რამდენადმე მოდერნიზებული ვარიანტი. მოდერნიზების ძირითადი მიზანი იყო შემოთავაზებული მეთოდი პრაქტიკული გამოყენებისათვის მოსახერხებელი ფორმით წარმოგვედგინა.

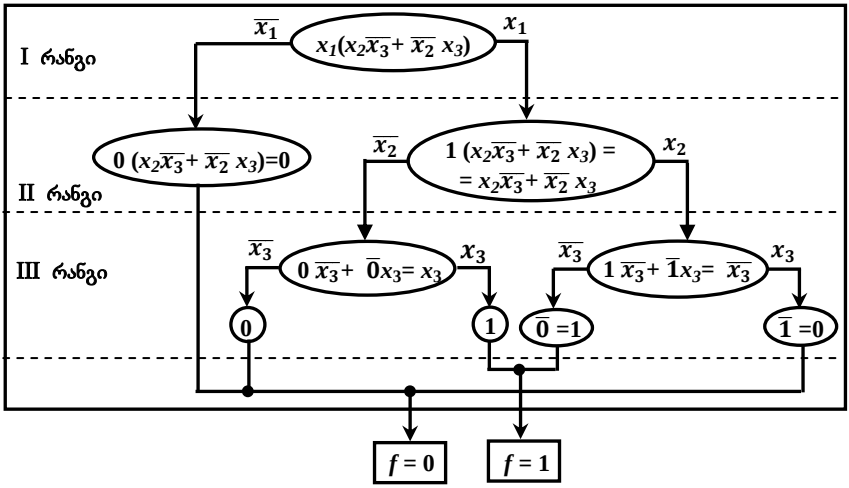
თვალსაჩინოებისათვის აღნიშნული მეთოდი განვიხილოთ კონკრეტული ლოგიკური ფუნქციის მაგალითზე. დავუშვათ, რომ საჭიროა ავაგოთ შემდეგი ლოგიკური ფუნქციის ბინარული პროგრამის ბლოკ-სქემა:

$$f(x_1, x_2, x_3) = x_1 (x_2 \overline{x_3} + \overline{x_2} x_3). \qquad (12.3)$$

დასმული ამოცანის გადაწყვეტას ვიწყებთ მოცემული ფუნქციის ***ბინარული დაშლის გრაფის*** შედგენით, რისთვისაც ვიყენებთ ლოგიკის ალგებრაში

ცნობილ მეთოდს. კერძოდ, n არგუმენტზე დამოკიდებული ლოგიკური $f(x_1,x_2,\dots,x_n)$ ფუნქცია შეიძლება x_i $(x_i=1,\dots,n)$ ცვლადების მიხედვით თანამიმდევრულად დაიშალოს შემდეგი ფორმულის მიხედვით:

$$f(x_1,\dots,x_{i-1}\,x_i,\,x_{i+1},\dots,x_n) = \overline{x_i}\;f(x_1,\dots,x_{i-1},\,0,\,x_{i+1},\dots,x_n) + x_i\,f(x_1,\dots,x_{i-1},\,1,\,x_{i+1},\dots,x_n). \qquad (12.4)$$



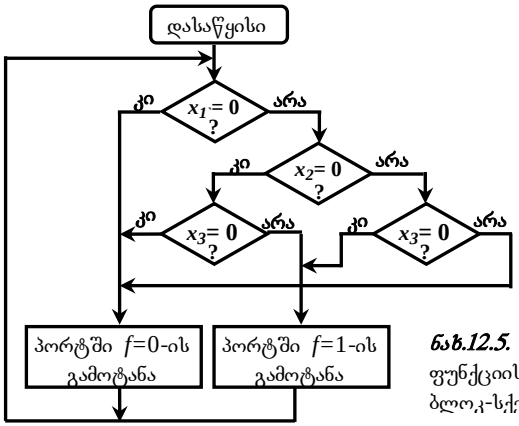
ნახ.12.4. ლოგიკური (***12.1***) ფუნქციის ბინარული დაშლის გრაფი

მოცემული ფუნქციისათვის ბინარული დაშლის გრაფის შედგენას ვიწყებთ ზედა წვეროდან., რომელსაც ***I*** რანგის წვეროს ვუწოდებთ (ნახ.***12.4***). ამ წვეროში ვათავსებთ განსახილველი ფუნქციას და მას (***12.2***) ფორმულის შესაბამისად x_1 ცვლადის მიხედვით ვშლით (***ნახ.12.4).*** განხილული წვეროდან გამოდის ასაგები ბინარული გრაფის ***II*** რანგის წვეროებისაკენ მიმართული შტოები, რომელთაგანაც ერთ-ერთი მონიშნულია $\overline{x_1}$, ხოლო მეორე x_1 ცვლადით (იხ.ნახ. ***12. 4***). $\overline{x_1}$ ცვლადით მონიშნული შტო შედის წვეროში, რომელშიც ჩაწერილია განსახილველი ფუნქციის $\overline{x_1}$ ცვლადის მიხედვით დაშლის შედეგი, ხოლო x_1 ცვლადით მონიშნული შტო შედის იმ წვეროში, რომელშიც ჩაწერილია განსახილველი ფუნქციის x_1 ცვლადის მიხედვით დაშლის შედეგი (იხ. ნახ. ***12.4***). როგორც ნახაზიდან ჩანს, წვეროდან გამოსული $\overline{x_1}$ შტო შედის ***II*** რანგის წვეროში, რომელშიც დგას კონსტანტა ***0,*** რომელსაც ***კონსტანტური წვერო*** ვუწოდოთ, ხოლო x_1 შტო წვეროში, რომელშიც მოთავსებულია ლოგიკური $x_2\overline{x_3}+\overline{x_2}x_3$ ფუნქცია, რომელსაც ***ლოგიკური წვერო*** ვუწოდოთ. კონსტანტურ წვეროსთან ფუნქციის დაშლის პროცესს ვწყვეტთ, ხოლო ლოგიკურ წვეროში მოთავსებულ ფუნ-

ქციას ვშლით, ოღონდ უკვე $\overline{x_2}$ ცვლადის მიხედვით. ვიღებთ ***III*** რანგის წვეროებს. როგორც ***12.4*** ნახაზიდან ჩანს, ***III*** რანგის ყველა წვერო კონსტანტურია, ე. ი. ცვლადების მიხედვით ლოგიკური ფუნქციის დაშლის პროცესი მთავრდება.

ზოგადად, ***n*** არგუმენტზე დამოკიდებული ლოგიკური ფუნქციის ცვლადებად დაშლის პროცესი ***n***-ურ რანგზე წყდება. ჩვენს მაგალითში იგი შეწყდა ***III*** რანგზე, რადგან განვიხილავდით ***n=3*** არგუმენტზე დამოკიდებულ (***12.3***) ლოგიკურ ფუნქციას.

ლოგიკური ფუნქციის დაშლის შედეგად ***0***-ის ტოლი კონსტანტური წვეროებს ვუერთებთ ***f=0*** გამოსასვლელს, ხოლო ***1***-ის ტოლ კონსტანტური წვეროებს - ***f=1*** გამოსასვლელს, როგორც ეს ***12.4*** ნახაზზეა ნაჩვენები. მასზე ნაჩვენებ გრაფს ეწოდება განსახილველი ***ლოგიკური ბინარული დაშლის გრაფი.***



ნახ.12.5. ლოგიკური (***12.1***) ფუნქციის ბინარული პროგრამის ბლოკ-სქემა.

12.4 ნახაზზე მიღებული (***12.1***) ფუნქციის ბინარული დაშლის გრაფიდან უშუალოდ აიგება აღნიშნული ფუნქციის ბინარული ***პროგრამის ბლოკ-სქემა*** (ნახ.***12.5***) შემდეგი ალგორითმის დახმარებით:

1. გრაფის წვეროები რომლებიდანაც გამოდის x_i, $\overline{x_i}$ ცვლადებით მონიშნული შტოები შეიცვალოს ლოგიკური ბლოკებით, რომლებშიც $x_i=0$, $i\in(1,2,\ldots,n)$.

2. ამოიშალოს გრაფის წვეროები, რომლებიდანაც არ გამოდის შტოები;

3. შტოებზე არსებული x_i ნიშნულები შეიცვალოს სიტყვებით «***კი***», ხოლო, $\overline{x_i}$ ნიშნულები სიტყვებით «***არა***» ($i\in(1,2,\ldots,n)$);

4. კონსტანტა ***0***-ის შემცველი წვეროების გაერთიანება მიუერთდეს ბლოკს «პორტ ***f=0***-ში გამოტანა», ხოლო კონსტანტა ***1***-ის შემცველი წვეროების გაერთიანება – ბლოკს «პორტ ***f=0***-ში გამოტანა».

12.5. სარკინიგზო ავტომატიკისა და ტელემექანიკის მიკროპროცესორულ სისტემათა ხარისხობრივი მახასიათებლები

მიკროპროცესორულ ***სატს***-ებს უნდა ჰქონდეთ გარკვეული ხარისხობრივი მაჩვენებლები, რომლებიც მათი დამუშავების პროცესში უნდა იყოს უზრუნველყოფილი. ამდენად მიკროპროცესორული ***სატს***-ების დამმუშავებლებისათვის მათი ცოდნა აუცილებელია. ამის გამო მოკლედ განვიხილოთ აღნიშნული მაჩვენებლები და მათი უზრუნველყოფის გზები.

საიმედოობა ეწოდება ***სატს***-ის უნარს მოცემულ რეჟიმებსა და გამოყენების პირობებში უზრუნველყოს მატარებელთა უწყვეტი და უსაფრთხო მუშაობა, ტექნიკური მომსახურება და რემონტი. მისი ამაღლება შეიძლება შემდეგი სამი ხერხით – მაღალი საიმედოობის მქონე ელემენტების გამოყენებით, სიჭარბის შეტანითა და ტექნიკური მომსახურების ეფექტური მეთოდების გამოყენებით. მიკროპროცესორულ სისატემათა საიმედოობის მაჩვენებლების პროგნოზირებადი მაჩვენებლები დამოკიდებულია სისტემის ტიპზე და განისაზღვრება ექსპერიმენტულად ან სისტემის მათემატიკური მოდელირების შედეგების მიხედვით. აღნიშნული მაჩვენებლები არ უნდა ჩამოუვარდებოდეს მოცემული სისტემის დღეს არსებული რელეურ-კონტაქტური ანალოგების მაჩვენებლებს.

უსაფრთხოება წარმოადგენს სისტემების თვისება მუშაობის პროცესში არ დაუშვას სახიფათო მტყუნებები. ***სატს***-ების კლასის სიტემების აგების თეორიისათვის ფუნდამენტურია ***სახიფათო მტყუნების*** ცნება. პოპულარულად განვმარტოთ იგი. დავუშვათ, რომ წესივრულ მდგომარეობაში ყოფნის დროს გარკვეული სისტემა ახდენს ალგორითმის რეალიზებას, რომელიც ზოგადი სახით შეიძლება ფორმალური ენის საშუალებით ჩავწეროთ. მტყუნებების წარმოშობისას იგი გადაიქცევა ახალი ალგორითმის მარეალიზებელ უწესივრო სისტემად. სისტემა თუ ასრულებს მატარებლების მოძრაობის უსაფრთხოების უზრუნველყოფასთან დაკავშირებულ საპასუხისმგებლო ფუნქციებს, მაშინ მისი მუშაობის ალგორითმის ფორმალურ გამოსახულებას ემატება "სახიფათო" მოვლენის ჩანაწერი, რომელიც იმ პირობებს განსაზღვრავს, რომელთა დროსაც ირღვევა მატარებლების მოძრაობის უსაფრთხოება.

ასეთი ფორმალიზაცია აუცილებელია სახიფათო მტყუნებების გამომრიცხავი სქემების სინთეზის დროს. აღნიშნული პრობლემა თეორიულად უკვე

გადაწყვეტილია. მთავარია, სწორედ ამ თეორიის საფუძველზე აიგოს უსაფრთხო სქემები.

უსაფრთხოების მახასიათებლებია უსაფრთხო მუშაობის ალბათობა და სახიფათო მტყუნებების ინტენსივობა. ***სატს***-ების ასაგებად გამოყენებული ელემენტებისათვის სახიფათო მტყუნებების ინტენსივობა არ უნდა აჭარბებდეს $(10^{-12}...10^{-14})$სთ$^{-1}$-ს.

გამძლეობა წარმოადგენს სისტემის თვისებას ელემენტთა მტყუნებების წარმოშობისას ფუნქციონირების უნარისა და მართვის ავტომატიზაციის დონის შემცირების ხარჯზე შეინარჩუნოს მუშაობის უნარი. ზოგადად, სისტემები შეიძლება იყოს: ***1)*** წესივრული, ***2)*** მუშაობის უნარის მქონე, ***3)*** მუშაობის უუნარო, ***4)*** სახიფათო ან ***5)*** ზღვრულ მდგომარეობაში. ბოლო ოთხ მდგომარეობაში სისტემა მხოლოდ მტყუნებების წარმოშობის შემდეგ შეიძლება გადავიდეს. სათანადოდ აგებისას მან შეიძლება შეინარჩუნოს მუშაობის უნარი, ან გადავიდეს მუშაობის უუნარო ან ზღვრულ მდგომარეობაში. უკანასკნელ შემთხვევაში სისტემის გამოყენება მიზანშეუწონელია. სახიფათო მტყუნებას სისტემა გადაყავს სახიფათო მდგომარეობაში და ამ მტყუნების - სისტემის ექსპლუატაცია დაუშვებელია.

მაშასადამე, გამძლე სისტემამ მტყუნების წარმოშობის შემდეგ უნდა შეინარჩუნოს მუშაობის უნარი. მიკროპროცესორული მომსახურებადი ***სატს***-ები ისე უნდა ავაგოთ, რომ მაღალი ალბათობის რაც შეიძლება დიდი რაოდენობის მტყუნებებმა ვერ დაარღვიოს მათი მუშაობის უნარი. მაგალითად, მიკროპროცესორულმა ელექტრულმა ცენტრალიზაციებმა მტყუნებების წარმოშობის შემდეგ უნდა შეინარჩუნოს ისრების ინდივიდუალურად გადაყვანისა და მომწვევი სიგნალით მატარებლების მიღების უნარი.

მტყუნებამდგრადობა – ეწოდება სისტემის უნარს ელემენტების მტყუნებების დროს ფუნქციონირების ეფექტურობის შეუმცირებლად შეინარჩუნოს მუშაობის უნარი. ასეთი სისტემები მანამ განაგრძობს ფუნქციონირებას, სანამ მათში მომხდარი მტყუნებების რაოდენობა გარკვეულ ***d*** მნიშვნელობას არ გადააჭარბებს. აღნიშნულის გამო სისტემას ***d-მტყუნებამდგრად სისტემას*** უწოდებენ. ისინი განაგრძობს ეფექტურად ფუნქციონირებას მანამ, სანამ წარმოშობილი მტყუნებების რაოდენობა ***d***-ს ტოლი ან მასზე ნაკლებია.

არსებობს მტყუნებამდგრაობის უზრუნველყოფის შემდეგი ორი ძირითადი ხერხი. ***პირველია*** მტყუნებების «შენიღბვა», ე. ი. სისტემის რომელიმე აპარატურული ან პროგრამული მოდულის შიგნით მომხდარი უწესივრობა იმალება და არ მჟღავნდება მოდულის გამოსასვლელებზე. ***მეორე ხერხი*** დაფუძნებულია სისტემის თვითტესტირებაზე: მტყუნება, პირიქით, უსწრაფესად მჟღავნდება და უწესივრო ბლოკი მუშაობის შეფერხებამდე წესივრულით ავტომატურად იცვლება.

მტყუნებამდგრადობის უზრუნველყოფა მოითხოვს აპარატურული და პროგრამული საშუალებების დიდ სიჭარბეს. ამ დროს აპარატურის რაოდენობა სამჯერ და უფრო მეტად იზრდება. ამის გამო ამ უნარით ***რელეური სატს***-ების აღჭურვა პრაქტიკულად შეუძლებელია: არ ***არსებობს მტყუნებამდგრადი რელეური სატს***-ები. მიკროპროცესორული ტექნიკის გამოყენებამ მტყუნებამდგრადი ***სატს***-ების აგება რეალური გახადა.

საკონტროლოდ ვარგისობა – სისტემის თვისებაა გაამარტივოს ტექნიკური მომსახურების, უწესივრო ელემენტების აღმოჩენისა და რემონტის პროცესები. ეს თვისება სისტემაში მისი შექმნის პროცესში შეიძლება ჩაიდოს და ისიც საჭიროებს სიჭარბის შეტანას. არსებულ რელეურ სისტემებს აქვს საკმაოდ მაღალი «ბუნებრივი» კონტროლვარგისობა, რადგან შესაძლებელია რელეთა მუშაობის ვიზუალურად გაკონტროლება. ინტეგრალური სქემებით, და, მაშასადამე, მიკროპროცესორებით აგებულ სისტემებს ასეთი უნარი არ გააჩნია. იგი შეიძლება სისტემის დამუშავების დროს კომპლექსური ტექნიკური ზომების მიღებით იქნეს უზრუნველყოფილი.

დ ა ნ ა რ თ ე ბ ი

დანართი 1
მიკროპროცესორების ტექნიკაში გამოყენებული ძირითადი ინგლისური აბრევიატურები და ტერმინები

- 2S (2-States Output) - ორი აქტიური (***0*** და ***1***) მდგომარეობიანი გამოსასვლელი; ***ტტლ***-ის (***TTL***-ის) სტანდარტული გამოსასვლელი.

- 3S (3-States Output) - სამი (ორი აქტიური – ***0,1*** და მესამე პასიური, გამორთული) მდგომარეობიანი გამოსასვლელი; გამოსასვლელის ორი აქტიური მდგომარეობისაგან განსხვავებული მესამე მდგომარეობა.

- AD - მისამართი/მონაცემები

- Adder - სუმატორი, ამჯამავი.

- AND - ლოგიკური ***და*** ფუნქცია.

- AC (Alternating Current)- ცვლადი დენი.

- ALU (Arithmetic and Logic Unit) - ალმ, არითმეტიკულ-ლოგიკური მოწყობილობა.

- ADC (Analog-to-Digital Converter)- ანალოგურ-ციფრული გარდამქმნელი.

- ASCII (American Standard Cfor ode Information Interchange) სიმბოლური ინფორმაციის გაცვლის სტანდარტული ამერიკული კოდი.

- BCD (Binary-Coded Decimal) – ორობით-ათობითი კოდი.

- BiCMOS (Bipolar Complementary Metal-Oxide-Semiconductor) – მიკროსქემების დამზადების ტექნოლოგია, რომელიც ერთ კრისტალში ერთმანეთს უთავსებს ბიპოლარულ და ***კლშნ*** (კომპლემენტარულ ლითონჟან-გეულ-ნახევარგამტარულ) სტრუქტურებს.

- Buffer - ბუფერი.

- Bus - სალტე, მაგისტრალი.

- CAS (Column-Address Select) – სვეტის მისამართის ამომრჩევი სიგნალი (დინამიკური მეხსიერების მიკროსქემებში).

- Chip - მიკროსქემა, ჩიპი.

- Chipset – კომპიუტერის ორგანიზებისათვის განკუთვნილი კონტროლერთა მიკროსქემების ნაკრები.

- CISC (***Complete Instructrion Set Computer***) – ბრძანებების სრული ნაკრების მქონე კომპიუტერი (ან პროცესორი).

- Clear - გაწმენდა, ნულზე ჩამოყრა.

- CLC, Clock – ტაქტური, მატაქტირებელი სიგნალი.

- CMOS (Complementary Metal-Oxide-Semiconductor) – კომპლემენტარული ლითონ-ჟანგეულ-ნახევარგამტარული (***კლშნ***) ტექნოლოგია.

- *Coder* - შიფრატორი, კოდერი.

- *Comparator* - კომპარატორი.

- *Converter* - გარდამქმნელი.

- *Core Speed* – პროცესორის შიგა სიხშირე, რომელზედაც მუშაობს მისი გამომთვლელი ბირთვი.

- *CPU* (***C**entral **P**rocessor **U**nit*) – ცენტრალური პროცესორი.

- *CRC (Cyclic **R**edundancy **Check)* –** ციკლური საკონტროლო ჯამი, აგრეთვე ასეთი ჯამის გამომყენებელი თეორია.

-*Counter* – მთვლელი.

- *DAC (Digital-to-**A**nalog **C**onverter)* -** ციფრულ-ანალოგური გარდამქმნელი.

- *DC (Direct **C**urrent)* -** მუდმივი დენი.

- *Decoder* - დეშიფრატორი, დეკოდერი.

- *Delay* - დაყოვნება.

- *Desktop* – სამაგიდო პერსონალური კომპიუტერი.

- *DIC* (***D**ual **I**n-line **C**eramic package*) **–** *DIL* ტიპის კერამიკული კორპუსი მიკროსქემისათვის.

- *DIL* (***D**ual **I**n-**L**ine package)* **-** მიკროსქემის კორპუსი, რომელშიც გამომყვანები ორ მწკრივში ვერტიკალურადაა განთავსებული.

- *DIMM* (***D**ual **I**n-Line **M**emory **M**odule*) **-** ორმხრივ განთავსებული გამომყვანების მქონე მეხსიერების მოდული.

- *DIP* (***D**ual **I**n-line **P**lastic package*) **-** *DIL* ტიპის პლასტმასური კორპუსი მიკროსქემისათვის.

- *DIP Switches* - *DIP* ტიპის კორპუსში ჩამონტაჟებული ტიპის მცირეგაბარიტული ამომრთველი.

- *DMA* – (***D**irect **M**emori **A**cces)* – მეხსიერებასთან პირდაპირი დაშვება; ინფორმაციის უშუალო გაცვლა.

- *DOS* (***D**isk **O**perating **S**ystem*) – დისკური ოპერატიული სისტემა.

- *DRAM (Dynamic **RAM)* -** დინამიკური ოპერატიული დამხსომებელი მოწყობილობა ***(DRAM)***.

- *Driver* - გამოსასვლელი ბუფერი, დრაივერი.

- *DSP* – (***D**igital **S**ignal **P**rocessor*) –სასიგნალო პროცესორი.

- *EEPROM (Electrically **E**rasable **P**rogrammable **ROM)* -** ელექტრული წაშლისა და დაპროგრამების უნარიანი მუდმივი დამხსომებელი მოწყობილობა.

-*EISA* (***E**nhanced **ISA***) – სისტემური ***ISA*** სალტის ციფრული გაფართოებული (***32***-თანრიგიანი) ვარიანტი, რომელიც ***ISA***-თან სრულად თავსებადია.

- EPROM (Erasable Programmable ROM) მუდმივი მეხსიერება, რომელშიც ჩაწერილი ინფორმაცია შეიძლება ულტრაიისფერი სხივებით წაიშალოს და მის ნაცვლად ახალი პროგრამა ჩაიწეროს.

- Fault –მტყუნება; შეწყვეტის ტიპი.

- FDD (***F****loppy* ***D****isk* ***D****rive*) – მოქნილ დისკზე ინფორმაციის გარე დამგროვებელი.

- Female - გასართი-როზეტი, ბუდე.

- FIFO (First In, First Out) – «პირველი შევიდა – პირველი გამოვიდა», მიმდევრობითი შეღწევიანი ოპერატიული დამხსომებელი მოწყობილობის ორგანიზაციის ერთ-ერთი ხერხი.

- Firmware – მიკროპროცესორული სისტემის ენერგოდამოუკიდებელ მეხსიერებაში შენახული პროგრამები.

- Flash memory – EEPROM მეხსიერების ნაირსახეობა, რომელიც ხასიათდება მაღალი ტევადობითა და მცირე წინაღობით, მოიხმარს მცირე ელექტრულ ენერგიას და აქვს ხელახლა ჩაწერის დიდი რაოდენობის ციკლები; ფლეშმეხსიერება.

- Flip-Fop - ტრიგერი.

- FLOPS (***F****loaring point* ***O****perations* ***P****er* ***S****econd*) – მცურავწერტილიან რიცხვებზე წამში შესრულებული ოპერაციების რაოდენობა, პროცესორის მწარმოებლურობის გაზომვის ერთეული.

- Gate - ლოგიკური ელემენტი, ვენტილი.

- GND (GrouND) – სქემის საერთო გამოსასვლელი.

- H (High) - სიგნალის მაღალი დონე; დადებითი ლოგიკის დროს იგი ლოგიკური ***1***-ის, ხოლო უარყოფითი ლოგიკის დროს – ლოგიკური ***0***-ის დონეს შეესაბამება.

- H – რიცხვის თექვსმეტობით სისტემით ჩაწერის ნიშანი, მაგ., ***3DE7h.***

- Handshake – ინფორმაციის გაცვლის ასინქრონული რეჟიმი, რომლის დროსაც გამოიყენება შემსრულებლის მზადყოფნის მადასტურებელი სიგნალი.

- Hardware – მიკროპროცესორული სისტემის აპარატურული (ელექტრონული) საშუალებები.

- HDD (***H****ard* ***D****isk* ***D****rive*) – ხისტ დისკზე ინფორმაციის გარე დამგროვებელი.

- Hex – თვლის თექვსმეტობითი სისტემა.

- HMA – (***H****igh* ***M****emory* ***A****rea*) – «მაღალი მეხსიერების» არე პერსონალურ კომპიუტერში.

- IC (Integrated Circuit) - ინტეგრალური მიკროსქემა.

- Inverter – ინვერტორი.

-Interrupts – შეწყვეტა;

- iCOMP (***I****ntel* ***C****omparrative* ***M****icroprocessor* ***P****erformance*) – ფირმა ***Intel***–ის მიკროპროცესორების მწარმოებლურობის შეფასების ინდექსი.

- IDE – (***I****ntegrated* ***D****rive* ***E****lectronics*) კომპიუტერთან დისკოსატარების მისაერთებელი ინტერფეისი.

-Idle – უქმი სვლის რეჟიმი.

-IDT – (***I****nterrupt* ***D****escriptor* ***T****able*) – შეწყვეტათა დესკრიპტორების ცხრილი.

-Instruction – ინსტრუქცია, ბრძანება.

- Instruction Set - პროცესორის ბრძანებათა სისტემა.

- INT – (***INT****errupt*) – შეწყვეტა, შეწყვეტის ვექტორი.

-IO, I/O (***I****nput/****O****utput*) – შეტანა/გამოტანა, შესასვლელი/გამოსასვლელი.

- IOPL (*Input/****O****utput* ***P****rivilege* ***L****evel*) – შეტანა/გამოტანის ოპერაციის პრივილეგიების დონე.

- IPC (***I****nstruction* ***P****er* ***C****ycle*) – ერთ ტაქტში პროცესორის მიერ შესრულებული ოპერაციების რაოდენობა.

- IRQ (***I****nterrupt* ***R****e****Q****uest) – შეწყვეტის მოთხოვნა.

- ISA (***I****ndustrial* ***S****tandard* ***A****rchiteccture*) – სინქრონული არამულტიპლექსირებული სალტე.

- Jumper – დაფაზე მანჭვალური კონტაქტების შემაერთებელი სახსნელი ზღუდარი, ჯემპერი.

- L (***L****ow*) - სიგნალის დაბალი დონე; დადებითი ლოგიკის დროს იგი ლოგიკური ***0***-ის, ხოლო უარყოფითი ლოგიკის დროს – ლოგიკური ***1***-ის დონეს შეესაბამება.

- L1 Cash და ***L2 Cash*** – ***1***-ლი დონის (შინაგანი) და მე-***2*** დონის (გარეგანი) კეშმეხსიერება.

- Latch – «სასხლეტის» ტიპის ტრიგერი (რეგისტრი).

- LCD (***L****iquid* ***C****rystal* ***D****isplay)* – თხევადკრისტალკური დისპლეი, ინდიკატორი.

- LIFO (***L****ast* ***I****n,* ***F****irst* ***O****ut*) – ოპერატიული მეხსიერების სახე, რომელიც მუშაობს პრინციპით: «ბოლოს შევიდა, პირველი გამოვიდა».

- Line driver – ხაზის დრაივერი, ბუფერი.

- LPT (***L****ine* ***P****rin****T****er*) – ინტერფეის ***Centronics***-ის მიხედვით მომუშავე პრინტერის მისაერთებელი პორტი.

- LSB (***L****east* ***S****ignificant* ***B****it*) – სიტყვის ან ბაიტის უმცროსი ბიტი.

- LSI (***L****arge* ***S****cale* ***I****ntegration)* – დიდი ინტეგრალური სქემა.

- LVT (L*ow-****V****oltage* ***T****echnology*) - მიკროსქემების დაბალვოლტური ტექნოლოგია (***3,3*** ვოლტის ტოლი კვების ძაბვა).

- ***Male*** – გასართი ჩანგალი, შტეკერი.

- ***Master*** – ინფორმაციის გაცვლაში მონაწილე წამყვანი, მთავარი მოწყობილობა; მავალებელი.

- ***MFLOPS*** (***Mega FLOPS***) - მცურავწერტილიან რიცხვებზე წამში შესრულებული მილიონი ოპერაციის რაოდენობა, პროცესორის მწარმოებლურობის გაზომვის ერთეული.

- ***MIPS*** (***M***ega ***I****nstructions* ***P****er* ***S****econd*) – წამში შესრულებული მილიონი ოპერაცია, პროცესორის მწარმოებლურობის გაზომვის ერთეული.

- ***MCP*** (***M****ath* ***C****o****P****rocessor*) – მათემატიკური თანაპროცესორი.

- ***MCU*** (***M****icroprogram* ***C****ontrol* ***U****nit*) – მიკროპროგრამული მართვის ბლოკი.

- ***MMU*** (***Memory Management Unit***) – მეხსიერების მართვის ბლოკი.

- ***Monostable multivibrator*** – მომლოდინე მულტივიბრატორი; ერთვიბრატორი.

- ***MSB*** (***M****ost* ***S****ignificant* ***B****it*) – სიტყვაში ან ბაიტში უფროსი ბიტი.

- ***MSW*** (***M****achine* ***S****tate* ***W****ord*) - მანქანის მდგომარეობის სიტყვა.

- ***NIC*** (***N****etwork* ***I****nterface* ***C****ard*) – ქსელური ბარათი, ლოკალური ქსელის ადაპტერი.

- ***NMI*** (***N****on* ***M****asked* ***I****nterrupt*) – შეუნიღბავი შეწყვეტა.

- ***NPU*** (***N****umeric* ***P****rocessor* ***U****nit*) – მათემატიკური თანაპროცესორი.

- ***NVRAM*** (***N****on-****V****olatile* ***RAM***) – კვების შეწყვეტისას ინფორმაციის შემნახველი ენერგოდამოუკიდებელი ოპერატიული მეხსიერება.

- ***OC*** (***O****pen-****C****ollector* ***O****utput*) – მიკროსქემის ღია კოლექტორიანი გამოსასვლელი.

- ***Oct*** – თვლის რვაობითი სისტემა.

- ***OTPROM*** (***O****ne-****T****ime* ***P****rogrammable* ***ROM***) – მომხმარებლის მიერ ერთჯერადად დაპრგრამებადი მუდმივი დამხსომებელი მოწყობილობა.

- ***OR*** – ლოგიკური ***ანა*** ფუნქცია.

- ***PAL*** ***(P****rogrammable* ***A****rray* ***L****ogic)* - დაპროგრამებადი ლოგიკური მატრიცა.

- ***Parity*** - *ლუწობა*, პარიტეტი.

- ***PC*** (***P****rogram* ***C****ounter*) – ბრძანებების მთვლელი.

- ***PCI*** (***P****rogrammable* ***I****nterruption* ***C****ontroller*) – შეწყვეტის დაპროგრამებადი კონტროლერი.

- ***PIO*** (***P****rogramming* ***I****nput/Output*) – პროგრამულად მართვადი შესასვლელი/გამისასვლელი.

-***PLD*** (***P****rogrammable* ***L****ogic* ***D****evice*) – დაპროგრამებადი ლოგიკური სქემა.

- ***Plug*** – ჩანგლის ტიპის გასართი.

-PnP, P&P (***Plug-and-Play***) – «ჩასვი და მართე» კომპიუტერის კონფიგურაციის ავტომატური აწყობა.

-Pointer – მაჩვენებელი.

- Polling – ალმის (ბიტის მდგომარეობის) პროგრამულად გამოკითხვა.

-POP – სტეკიდან ამოღება.

- POST (***P****ower* ***O****n* ***S****elf* ***T****est*) – საწყისი ჩართვის ტესტი.

- POST (***P****roicedure* ***O****f* ***S****elf-****T****esting*) – თვითტესტირების პროცედურა.

- Power down – შემცირებული ენერგომოხმარების რეჟიმი.

-PPI (***P****rogrammable* ***P****eripheral* ***I****nterface*) – პერიფერიული მოწყობილობების დაპროგრამებადი ინტერფეისი.

- Preset – წინასწარი დაყენება.

- PROM (P*rogrammable* ***ROM)*** – დაპროგრამებადი მუდმივი დამხსომებელი მოწყობილობა; გადაპროგრამებადი მუდმივი დამხსომებელი მოწყობილობა.

- PSW (***P****rocessor* ***S****tatus* ***W****ord*) - პროცესორის მდგომარეობის სიტყვა, შიგა რეგისტრში პროცესორის მდგომარეობის კოდი.

-Multiplexer - მულტიპლექსორი.

- Push– სტეკში შენახვა.

- Pull-up Resistor - მიკროსქემის გამოსასვლელსა და კვების სალტეს შორის ჩასართავი სადატვირთო წინაღობა.

-Multivibrator – მულტიპლექსორი.

- NAND (***not and***) – ლოგიკური ***ან-არა*** ფუნქცია.

- Noninverter – მამეორებელი.

- NOR(***not or***) –ლოგიკური ***და-არა*** ფუნქცია.

- NVRAM (Non-V*olatile* ***RAM)*** - ენერგოდამოუკიდებელი ოპერატიული დამხსომებელი მეხსიერება, რომელიც ინფორმაციას ინახავს კვების გამორთვის შემდეგ.

- Q-Bus (ცნობილია ***LSI-11 Bus*** სახელითაც) – მულტიპლექსირებული სალტე

- RAM (R*andom* ***A****ccess* ***M****emory)*** - ოპერატიული დამხსომებელი მოწყობილობა.

- RAS (R*ow-****A****ddress* ***S****elect)*** - სტრიქონის მისამართის ამომრჩევი სიგნალი (დინამიკური მეხსიერების მიკროსქემებში).

- Receiver – მიმღები, შესასვლელი ბუფერი.

- Refresh – რეგენერაცია (დინამიკურ მეხსიერებაში).

- Register – რეგისტრი.

- Reset - ნულოვან ან საწყის მდგომარეობაში მყოფი მიკროსქემის დაყენების სიგნალი.

- ***RISC*** (***R**educed **I**nstruction **S**et **C**omputer*) – ბრძანებების შემოკლებული ნაკრებიანი კომპიუტერი (პროცესორი).

- ***ROM (R**ead-**O**nly **M**emory)* – მუდმივი დამხსომებელი მოწყობილობა.

- ***RS-232C*** (***R**eference **S**tandard*) - მონაცემების მიმდევრობითი გადაცემის სტანდარტულიი ინტერფეისი.

- ***RxC (R**eceived **C**lock)* - მისაღები სიგნალი.

- ***RxD (R**eceived **D**ata)* – მისაღები მონაცემები.

- ***RxC*** (***R**eceived **C**lock*) – მისაღები სინქრონსიგნალი.

- ***RxD*** (***R**eceived **D**ata*) – მისაღები მონაცემები.

- ***Schmitt trigger*** -შმიდტის ტრიგერი.

- ***SCSI*** (***S**mall **C**ompute**r** **S**ystem **I**nterface*) – კომპიუტერთან გარე მოწყობილობების, მათ შორის დისკოსატარების, მიერთების ინტერფეისი.

- ***SDRAM (S**ynchronous **D**ynamic **RAM)*** - სინქრონული დინამიკური ოპერატიული დამხსომებელი მოწყობილობა.

- ***Set*** – ლოგიკური ***1***-ის ტოლ მდგომარეობაში გამოსასვლელის დაყენების სიგნალი.

- ***SIMM*** (***S**ingle **I**n-Line **M**emory **M**odule*) - ერთ მწკრივში განლაგებული გამომყვანებიანი მეხსიერების მოდული.

- ***SIP*** (***S**ingle **I**n-line **P**ackage*) - მიკროსქემის ერთ მწკრივში განლაგებული გამომყვანებიანი კორპუსი.

- ***Slave*** – ინფორმაციის გაცვლაში მონაწილე ამყოლი, პასიური მოწყობილობა.

- ***Slot*** –ნაბეჭდი გამტარების სახის მქონე გასართის მქონე ნაბეჭდი დაფების მისაერთებელი ხვრელური გასართი, სლოტი.

- ***Socket*** – დაფაზე მიკროსქემების დასაყენებელი კონტაქტირებადი მოწყობილობა, სოკეტი.

- ***Softwsare*** – მიკროპროცესორული სისტემის პროგრამული საშუალებები (პროგრამები).

- ***SP*** (***S**tack **P**ointer*) – სტეკის მაჩვენებელი.

- ***SRAM*** (***S**tatic **RAM***) – სტატიკური ოპერატიული დამხსომებელი მოწყობილობა.

- ***Stack*** – სტეკის მაჩვენებელი.

- ***Strobe*** - მასტრობირებელი სიგნალი, სტრობი (იმ ოპერაციის დაწყების სიგნალი, რომელიც დროის განსაზღვრულ პერიოდში უნდა შესრულდეს).

- ***Terminator*** – კავშირის ხაზში არსებული ბოლო მათანხმებელი მოწყობილობა (ჩვეულებრივ – რეზისტორი).

- ***Timer*** – ტაიმერი, ტაიმერული ხელსაწყო.

- TI, Til (***T****exas* ***I****nstruments Inc.*) **-** მცირე და საშუალო ინტეგრაციის მქონე ციფრული მიკროსქემების მწარმოებელი წამყვანი ამერიკული ფირმა.

- TR (T*erminate* ***R****esistor)* **–** კავშირის ხაზის სადატვირთო რეზისტორი.

- Transceiver **–** მიმღებ-გადამცემი, ტრანსივერი, ორმხრივმიმართული ბუფერი.

- Transmitter **–** გადამცემი, გამოსასვლელი ბუფერი.

- Trap – მახე; შეწყვეტის ტიპი.

- Trigger **–** ტრიგერი.

- TTL (***T****ransistor-****T****ransistor* ***L****ogic)* **–** ტრანზისტორ-ტრანზისტორული ლოგიკა, ***ტტლ***.

- TTLS (***T****ransistor-****T****ransistor* ***L****ogic* ***S****chottky*) **–** შოტკის ტრანზისტორ-ტრანზისტორული ლოგიკა.

- Turbo **-** ამაღლებული სწრაფმოქმედების რეჟიმი

- TxC (***T****ransmitted* ***C****lock*) **-** გადასაცემი სინქრონსიგნალი.

- TxD (T*ransmitted* ***D****ata)* **–** გადასაცემი მონაცემები.

- UPI (***U****niversal* ***P****eripheral* ***I****nterface*) – უნივერსალური პერიფერიული ინტერფეისი.

- V **-** ძაბვა (***V****oltage*), ვოლტი (***V****olt*).

VLB (***VESTA Local Bus***) – პერსონალური კომპიუტერის ლოკალური სალტე.

- VLSI (***V****ery* ***L****arge* ***S****cale* ***I****ntegration*) **–** ზედიდი ინტეგრალური სქემა.

- Watchdog – მოდარაჯე ტაიმერი. მიკროპროცესორული სისტემა გამოყავს «დაკიდებული» მდგომარეობიდან.

- Waveform **-** სიგნალის ფორმა.

XOR- გამომრიცხველი ***ან***; ***2***-ის მოდულით შემკრება.

-Z (Z-state) **-** მიკროსქემის გამოსასვლელის მესამე (მაღალიმპენდასური) მდგომარეობა.

- ZIF (***Z****ero* ***I****nsertion* ***F****orce*) **–** ნულოვანი ძალვით ჩასადგმელი გასართი ან სოკეტი.

დანართი 2
ნახევარგამტარული ტექნიკის საფუძვლები

ტერმინი «ტრანზისტორი» წარმოქმნილია სიტყვების «ტრანსფერისა» (გარდამქმნელისა) და «რეზისტორის» ურთიერთშერწყმით. საბოლოოდ სიტყვა «***ტრანზისტორი***». «წინაღობის გარდამქმნელს» ნიშნავს. ტრანზისტორი გამოიყენება როგორც ანალოგურ, ასევე ციფრულ ტექნიკაში. ანალოგურ ტექნკაში იგი ასრულებს სუსტი სიგნალების მაძლიერებლის, ხოლო ციფრულ ტექნიკაში – გასაღების ფუნქციას. ორივე აღნიშნულ პროცესში გასარკვევად აუცილებელია იცოდეთ ტრანზისტორის დასამზადებლად გამოყენებულ ნახევარგამტარტში მიმდინარე პროცესები.

დ.2.1. ელექტროგამტარობა და ატომის აგებულება

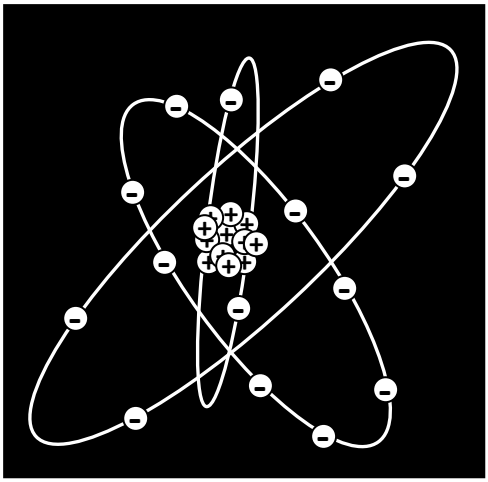
ელექტრული დენი გამტარში ელექტრონების მოძრაობით წარმოიქმნება. იმის წარმოსადენად, თუ როგორ ხდება ეს, საჭიროა განვიხილოთ ატომის აგებულება. ამას მაქსიმალურად გამარტივებულად, შეიძლება ითქვას, რომ, პრიმიტულადაც კი გავაკეთებთ, მაგრამ ეს განხილვა პროცესის არსში იმდენად გაგვარკვევს, რამდენადაც ეს ნახევარგამტარის მუშაობის გასაგებადაა საკმარისი.

1973 წელს დანიელმა ფიზიკოსმა ***ნილს ბორმა*** ატომის პლანეტარული მოდელი შემოგვთავაზა (ნახ.***დ2.1***).

აღნიშნული თეორიის თანახმად ატომი შედგება ბირთვისა და მის ირგვლივ გარკვეულ ორბიტებზე მბრუნავი უარყოფითად დამუხტული ელექტრონებისაგან. ბირთვი შედგება დადებითად დამუხტული პროტონებისა და ნეიტრალური (დაუმუხტავი) ნეიტრონებისაგან. აღნიშნულის შედეგად ბირთვი დადებითადაა დამუხტული, რომლის სიდიდე პროტონების ჯამური მუხტის ტოლია. ელექტრონების ჯამური უარყოფითი მუხტის სიდიდე უდრის ბირთვის დადებითი მუხტის სიდიდეს. ეს მუხტები ერთმანეთს ანეიტრალებს, რის გამოც ატომი ნორმალურად ნეიტრალურია.

ელექტრონის დაკარგვისას ჯამური ელექტრული მუხტი ხდება დადებითი, ხოლო თავად ატომი გადაიქცევა ***დადებით იონად***. უცხო ელექტრონის მიერთებისას ატომი გადაიქცევა ***უარყოფით იონად***.

დ2.2 ნახაზზე მოყვანილია მენდელეევის პერიოდული ცხრილის ფრაგმენტი. ყურადღება მივაქციოთ უჯრას, რომელშიც მოთავსებულია სილიციუმი ანუ კაჟბადი (***Si***).



ნახ.Დ2.1. *ატომის ნილს ბორისეული პლანეტარული მოდელი*

მარჯვენა ქვედა კუთხეში სვეტად დაჯგუფებული ციფრები ატომის ორბიტებზე ელექტრონების განაწილებას გვიჩვენებს. მის თანახმად სილიციუმის ბირთვის ირგვლივ სამი ორბიტაა. პირველ (ბირთვთან უახლოეს) ორბიტაზე ბრუნავს ***2***, მეორე ორბიტაზე – ***8,*** ხოლო მესამე ორბიტაზე – **4** ელექტრონი. სილიციუმის სტრუქტურა, რომელზეც გამოსახულია ბირთვი და მის ირგვლივ მბრუნავი ელექტრონები, ***Დ2.3,ა*** ნახაზზეა ნაჩვენები.

ერთ ორბიტაზე მოძრავი ელექტრონები წარმოქმნის თავისებურ ***გარსს***. ელექტრონების რაოდენობაზე დამოკიდებულებით ატომში შეიძლება სხვადასხვა რაოდენობის გარსი არსებობდეს, მაგრამ მათი რაოდენობა ***7***-ს არ აჭარბებს. ეს გარსები შესაბამისად აღნიშნულია ლათინური ***K, L, M, N, O, P*** და ***Q*** ასოებით. თითოეულ გარსში შემავალი ელექტრონების მაქსიმალური რაოდენობები ასეთია: ***K***=**2**, ***L***=**8**, ***M*** =**18**, ***N***=**32**, ***O*** =***50***, ***P***=***72***, ***Q***=***98***. მაგალითად, ბოლო ***Q*** გარსში შეიძლება შედიოდეს ***98*** ან უფრო ნაკლები რაოდენობის ელექტრონები. თითოეულ გარსში არსებული ელექტრონების რაოდენობები ჩვენთვის საინტერესო არ არის. ჩვენ გვაინტერესებს მხოლოდ ***ატომის გარე გარსში*** განთავსებული ელექტრონები.

ყველა ელექტრონი ერთსა და იგივე სიბრტყეზე არ ბრუნავს. ***K*** სახელწოდების გარსში შემავალი ორი ელექტრონიც ერთმანეთთან ახლო განთავსებულ ორ სფერულ ორბიტებზე ბრუნავს, ხოლო სხვა გარსებში შემავალი ელექტრონების ორბიტების ფორმები იმდენად მრავალფეროვანია, რომ მათი

აღწერა საკმაოდ დიდ დროს წაგვართმევს და ამიტომ მას თავს ავარიდებთ. მსჯელობის გასამარტივებლად შეიძლება ჩავთვალოთ, რომ ყველაფერი ისე ხდება, როგორც ეს ნახაზ ***ლ2.3,ა***-ზეა ნაჩვენები.

III		IV		V	
ა	ბ	ა	ბ	ა	ბ
B 5 ბორი 10,811	3 2	**C** 6 ნახშირბადი 12,011	4 2	**N** 7 აზოტი 14,007	5 2
Al 13 ალუმინი 26,092	3 8 2	**Si** 14 სილიციუმი 26,092	4 8 2	**P** 15 ფოსფორი 30,974	5 8 2
21 **Sc** სკანდიუმი 44,956	2 9 8 2	**22** **Ti** ტიტანი 47,956	2 10 8 2	**23** **V** ვანდანიუმი 50,941	2 11 8 2
Ga 31 გალიუმი 69,72	3 18 8 2	**Ge** 32 გერმანიუმი 72,59	4 18 8 2	**As** 33 დარიშხანი 74,822	5 18 8 2

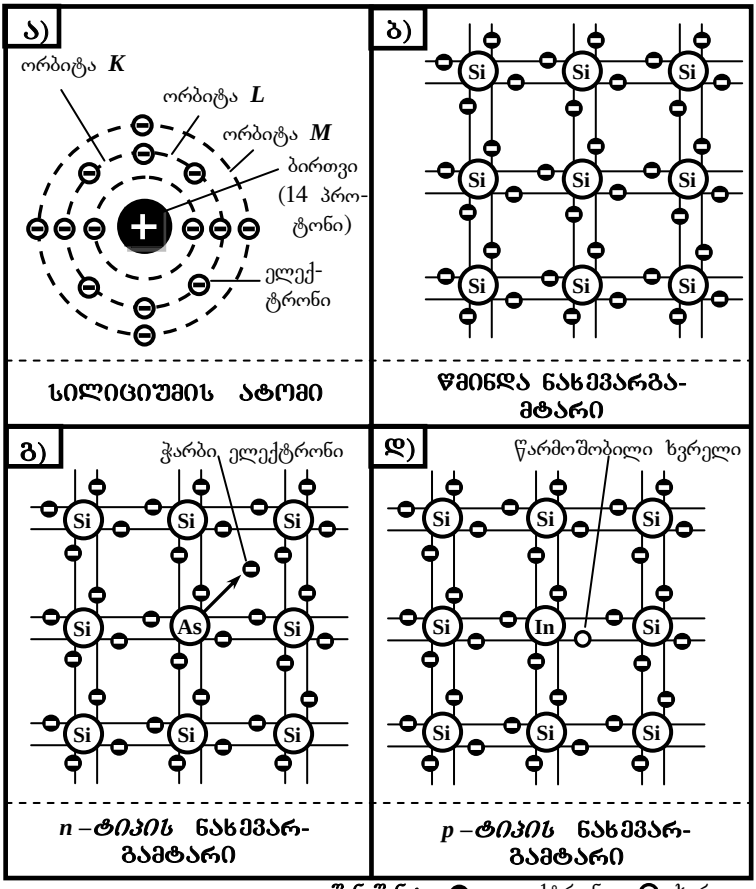
ნახ. ლ3.2. *მენდელეევის პერიოდული ცხრილის ფრაგმენტი*

ასეთი გამარტივების შემთხვევაში სილიციუმის მეტად რთული კონფიგურაციის კრისტალური მესერიც შეიძლება ბრტყელი სახით წარმოვადგინოთ (ნახ ***ლ2.3,ბ***), რაც გაგვიადვილებს მასალის აღქმას.

გარე გარსის ელექტრონებს (იხ. ***ლ2.3,ა***) ***ეწოდება სავალენტო ელექტრონები***. ***ლ2.3,ბ*** ნახაზზე სწორედ სავალენტო ელექტრონებია ნაჩვენები (დანარჩენ ელექტრონებს მნიშვნელობა არა აქვს ჩვენთვის საინტერესი საკითხის განხილვისათვის). სწორედ ისინი მონაწილეობს მოლეკულებად ატომების შეერთებასა და ნივთიერების თვისებების წარმოქმნაში.

გარე გარსში შემავალ ელექტრონებს შეუძლია მოწყდეს საკუთარ ატომს, თავისუფლად იხეტიალოს ნივთიერებაში, ხოლო გარკვეული პირობების არსებობის შემთხვევაში – წარმოქმნას ელექტრული დენი. გარდა ამისა,

სწორედ გარე გარსში მიმდინარეობს ნახევარგამტარული მაძლიერებელი ხელსაწყოების - ***ტრანზისტორების*** წარმომქმნელი პროცესები.



ნახ Დ2.3 *სილიციუმის ანუ ივიე კაჟბადის ატომი და მისგან წარმოშობილი ნახევარგამტარები*

Დ2.3,ბ ნახაზზე ნაჩვენებია სილიციუმის ანუ კაჟბადის (***Si***) ატომების (ნახ. ***Დ2.3,ა***) გაერთიანებით წარმოქმნილი წმინდა სილიციუმური (კაჟბადური) ნახევარგამტარი. მასში აბსოლუტური ნულის ტემპერატურაზე თავისუფალი ელექტრონები არ არსებობს.

სიტუაცია შეიცვლება, თუ სილიციუმში შევიტანთ დარიშხანის (***As***) ატომებს (ნახ.***Დ2.3,გ***), რომელთა გარე ორბიტაზე მოძრაობს ხუთი ელექტრო-

ნი. ამ ზუთი ელექტრონიდან ოთხი კავშირს დაამყარებს სილიციუმის (კაჟბადის) ატომებთან, ხოლო მეხუთე ელექტრონი გადაიქცევა თავისუფალ ელექტრონად. ასეთ ნახევარგამტარს ეწოდება ***n ტიპის ნახევარგამტარი.***

სილიციუმის წმინდა ნახევარგამტარში სამვალენტიანი ინდიუმის (***In***) ატომების შეტანისას სამი ელექტრონი კავშირს დაამყარებს სილიციუმის მეზობელ სამ ატომთან, ხოლო მეოთხე ელექტრონის ადგილზე წარმოიშობა სიცარიელე, რომელსაც ხვრელი ეწოდება (ნახ. ***Დ2.3,დ***). ასეთ ნახევარგამტარს ***p ტიპის ნახევარგამტარი*** ეწოდება.

Დ.2.2. გამტარები, იზოლატორები და ნახევარგამტარები

ელექტროტექნიკაში სახვადასხვა მასალა გამოიყენება. ნივთიერების ელექტრონულ თვისებებს განსაზღვრავს გარე სავალენტო ორბიტაზე ელექტრონების რაოდენობა. რაც უფრო ნაკლები რაოდენობის ელექტრონებია გარე სავალენტო ზონაში, მით უფრო ნაკლებადაა ისინი დაკავშირებული ბირთვთან და მით უფრო ადვილად შეუძლია მათ წავიდეს «სამოგზაუროდ».

ტემპერატურული რყევების ზემოქმედებით ელექტრონები შეიძლება მოწყდეს ატომს და დაიწყოს მოძრაობა ატომთაშორის სივრცეში. ***მათ თავისუფალი ელექტრონები*** ეწოდება და სწორედ ისინი წარმოქმნის დენს. დაისმის კითხვა, თუ რამდენად დიდია ატომთაშორისი სივრცე და აქვს თუ არა ნივთიერების შიგნით თავისუფალ ელექტრონებს მოძრაობის შესაძლებლობა?

მყარი სხეულებისა და სითხეების სტრუქტურა ძაფის გორგალივით უწყვეტი და მჭიდრო გვეგონოს. სინამდვილეში კი მყარი სხეული უფრო სათევზაო ან ფრენბურთის ბადეს უფრო ჰგავს. საყოფაცხოვრებო დონეზე ძნელია ამის შემჩნევა, მაგრამ ზუსტი სამეცნიერო კვლევები გვიჩვენებს, რომ ელექტრონებსა და ბირთვს შორის არსებული მანძილები თავად ბირთვისა და ელექტრონების ზომებზე გაცილებით დიდია.

ატომის ბირთვის ზომას ფეხბურთის ბურთის ზომამდე თუ გავზრდით, მაშინ ელექტრონები მუხუდოს მარცვლების ზომას მიიღებს, ხოლო თითოეული ასეთი მარცვალი «ბირთვისაგან» რამდენიმე ასეული და შეიძლება ათასეული მეტრით აღმოჩნდება დაშორებული. ბირთვსა და ელექტრონს შორის კი სიცარიელეა - უბრალოდ არაფერი არ არის! ასეთ მასშტაბში თუ წარმოვიდგენთ მანძილებს ნივთიერების ატომებს შორის, მაშინ მივიღებთ ნამდვილად ფანტასტიკურ ზომებს – ისინი ათეული და ასეული კილომეტრის ტოლი აღმოჩნდება!

კარგი ***გამტარებია*** ლითონები. მაგალითად, ***ოქროსა*** და ***ვერცხლის*** ატომების გარე ორბიტებზე მხოლოდ თითო-თითო ელექტრონია, ამიტომ ისინი

საკეთესო გამტარებია. ელექტრულ დენს, ოღონდ რამდენადმე ცუდად, ატარებს რკინაც. კიდევ უფრო ცუდად ატარებს დენს მაღალი წინაღობიანი შენადნობები - ნიქრომი, მანგანინი, კონსტანტანი, ფექრალი (რკინა-ქრომალუმინი) და სხვები.

მასალის მიერ ელექტრული დენის გატარების უნარის შესფასებლად შემოტანილი იქნა ***"კუთრი ელექტროგამტარობის"*** ცნება, რომლის შებრუნებული სიდიდეა ***"კუთრი წინაღობა".*** მექანიკაში ამ ცნებებს შეესაბამება **კუთრი წონა.**

გამტარებისაგან განსხვავებით ***იზოლატორები*** ცდილობს არ დაკარგოს ელექტრონები. მათში ელექტრონები ძალიან მტკიცედაა დაკავშირებული ბირთვთან და თავისუფალი ელექტრონების რაოდენობა უმნიშვნელოა; რაც უფრო ნაკლებია ეს რაოდენობა, მით უკეთესია იზოლატორი და პირიქით.

იზოლატორებში თავისუფალ ელექტრონებს წარმოშობს ელექტრონთა თბური რხევები: მაღალი ტემპერატურის ზემოქმედებისას ზოგიერთი ელექტრონი მაინც ახერხებს ბირთვისაგან მოწყვეტას, რაც აუარესებს მაიზოლირებელ თვისებებს.

იდეალური გამტარის კუთრი წინაღობა ნულის ტოლი უნდა იყოს. საბედნიეროდ ასეთი გამტარი არ არსებობს: ასეთი გამტარის არსებობისას უაზრობად გადაიქცეოდა ომის ***I=U/R*** კანონი (ნულზე გაყოფა არ შეიძლება!).

მხოლოდ აბსოლუტური ნულის ***(-273,2C^0)*** ტოლ ტემპერატურაზე სრულიად ქრება თბური რყევები და ამ დროს ყველაზე ცუდი იზოლატორიც საკმაოდ კარგ იზოლატორად გარდაიქმნება. სიცუდისა და სიკარგის რაოდენობრივად შეფასებისათვის გამოიყენება კუთრი წინაღობის ცნება.

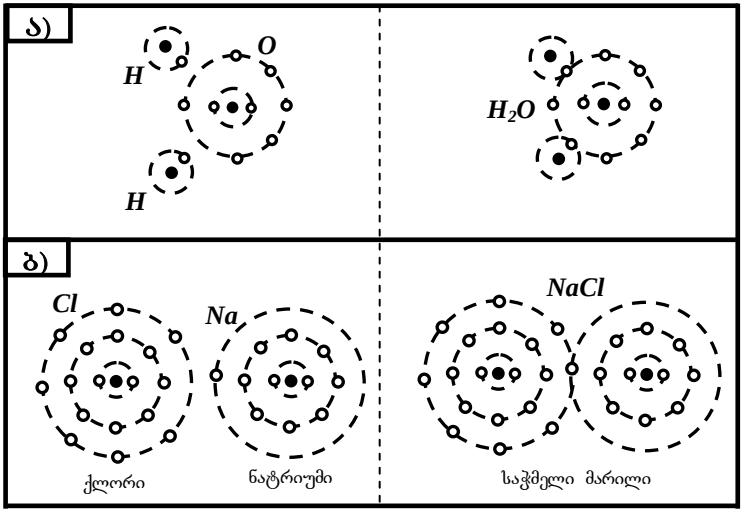
მასალის ***კუთრი წინაღობის*** განზომილებაა ომი/სანტიმეტრი. იგი წარმოადგენს ამ მასალისაგან დამზადებული ისეთი კუბის წინაღობას, რომლის წიბოს სიგრძე ***1*** სანტიმეტრია. კუთრი წინაღობის შებრუნებულ სიდიდეს ეწოდება კუთრი გამტარობა. რომლის სიდიდე სიმენსებში (***სმ***) იზომება (***1სმ = 1/ომი***)

კარგი გამტარობა ანუ მცირე კუთრი წინაღობა აქვს ვერცხლს, სპილენძსა და ალუმინს, რომელთა კუთრი წინაღობებია: ***1,5•10^{-6}*** , ***1,78•10^{-6}*** და ***2,8•10^{-6}*** ომი/სმ. გაცილებით ცუდი გამტარობა ანუ მაღალი კუთრი წინაღობა აქვს შენადნობებს კონსტანტინსა და ნიქრომს, რომელთა კუთრი წინაღობებია შესაბამისად ***0,5•10^{-4}*** და ***1,1•10^{-4}*** ომი/სმ. მათ შეიძლება ვუწოდოთ ***ცუდი გამტარები.***

ცალკე ჯგუფად შეიძლება გამოვყოთ ნახევარგამტარები: გერმანიუმი, კაჟბადი და სელენი, რომელთა კუთრი წინაღობები შესაბამისად ***60, 5000*** და ***100000*** ომი/სმ-ს ტოლია ამ ჯგუფში შემავალ ელემენტებს ცუდ გამტა-

რებზე მაღალი, მაგრამ იზოლატორებზე ნაკლები კუთრი წინაღობები აქვს, რის გამოც მათ ***ნახევარგამტარებს*** უწოდებენ.

ზემოთ მოცემული მოკლე შესავლის შემდეგ განვიხილოთ როგორ ურთიერთზემოქმედებს ატომები და როგორ წარმოშობს სხვადასხვა ნივთიერების წარმომქმნელ ***მოლეკულებს.***



ნახ.Დ2.4. *წყლისა (ა) და საჭმელი მარილის (ბ) მოლეკულების წარმოქმნა*

ატომის სტაბილური მდგომარეობა ეწოდება ისეთ მდგომარეობას, რომლის დროსაც ატომის გარე ორბიტაზე ***8*** ელექტრონია. სტაბილური ატომი არ ცდილობს მეზობელ ატომებს არც წაართვას და არც დაუთმოს ელექტრონები. ამაში დასარწმუნებლად საკმარისია შევხედოთ პერიოდულ ცხრილში არსებულ ინერტულ აირებს ***ნეონს, არგონს, კრიპტონსა*** და ***ქსენონს***. თითოეული მათგანის ატომის გარე შრეში რვა-რვა ელექტრონია, რის გამოც ეს აირები სხვა ნივთიერებებთან რეაქციაში არ შედის და ახალ ქიმიურ ნივთიერებებს არ წარმოქმნის.

სხვანაირად იქცევა ატომები, რომელთა გარე შრეში ***8***-ზე ნაკლები რაოდენობის ელექტრონებია. ისინი ცდილობს გაუერთიანდეს სხვა ატომებს და მათგან გადმოიბიროს ელექტრონები საკუთარ გარე შრეში არსებული ატომების რაოდენობის რვამდე შესავსებად.

მაგალითად, წყლის H_2O მოლეკულა შედგება წყალბადის ორი და ჟანგბადის ერთი ატომისაგან (ნახ.***Დ2.4,ა***).

ნახ.დ2.4,ა-ს მარცხენა მხარეზე ცალ-ცალკეა გამოსახული წყალბადის ორი და ჟანგბადის ერთი ატომი. ჟანგბადის ატომის გარე ორბიტაზე არსებობს ***6*** ელექტრონი და ამ ატომის მახლობლად იმყოფება წყალბადის ორი ატომის ორი ელექტრონი. ჟანგბადს გარე ორბიტაზე არსებული ელექტრონების რაოდენობის სანუკვარ რვამდე შესავსებად სწორედ ორი ელექტრონი აკლია და იგი ამ მიზანს მიაღწევს წყალბადის ზემოთ აღნიშნული ორი ატომის მიერთებით.

ანალოგურ მდგომარეობაშია წყალბადის ატომები. მათ გარე ორბიტის შესავსებად შვიდ-შვიდი ელექტრონი სჭირდება. წყალბადის პირველ (ზედა) ატომს შეუძლია ***6*** ელექტრონი ჟანგბადის ატომის გარე ორბიტიდან, ხოლო ***1*** ატომი მეორე წყალბადის გარე ორბიტიდან მიიღოს. ამის შედეგად მის გარე ორბიტაზე ელექტრონების რაოდენობა ***8***-ის ტოლი გახდება. ასევე შეუძლია მოიქცეს წყალბად-ის მეორე ატომმა და მივიღებთ ***ნახ.დ2.4,ა*** ნახაზის მარჯვენა ნაწილში გამოსახულ წყლის მოლეკულას.

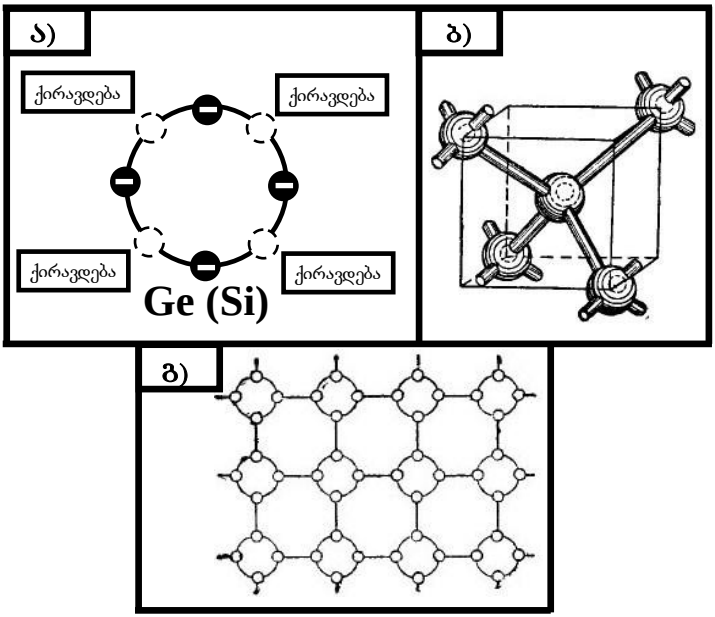
ნახ. ***დ2.4,ბ***-ზე ნაჩვენების ნატრიუმისა და ქლორის ატომების შეერთების პროცესი, რომლის შედეგადაც მიიღება ნატრიუმის ქლორიდი, რომელიც ჩვეულებრივ საჭმელი მარილია.

ამ შემთხვევაშიც რეაქციაში მონაწილე თითოეული ატომი მეორე ატომისაგან იღებს მისთვის საჭირო ელექტრონს: ნატრიუმი საკუთარ ერთადერთ ელექტრონს მიუერთებს ქლორის ***7*** ელექტრონს და იმავდროულად საკუთარი გამგებლობაში გადაიყვანს ქლორის შვიდივე ელექტრონს. ამის შემდეგად ორივე ატომის გარე შრეზე ელექტრონების რაოდენობა რვის ტოლი გახდება, რითაც სრული თანხმობა და კეთილდღეობა მიიღწევა!

დ.2.3. ატომების ვალენტობა

ატომები, რომელთა გარე ორბიტაზე ***6*** ან ***7*** ელექტრონია, ცდილობს მიიერთოს ***1*** ან ***2*** ელექტრონი. ასეთ ატომებზე ამბობენ, რომ ისინი ერთ- ან ორვალენტიანებია. ატომის გარე ორბიტაზე თუ ***1,2*** ან ***3*** ელექტრონია, მაშინ იგი მათ გაცემას ცდილობს და იგი შესაბამისად ერთ-, ორ- ან სამვალენტიანად ითვლება.

ატომის გარე ორბიტაზე ***4*** ელექტრონის არსებობისას იგი ცდილ-ობს ისეთ ატომთან გაერთიანდეს, რომლის გარე შრეზეც ასევე ***4*** ელექტრონია. სწორედ ამიტომ ერთიანდება ტრანზისტორების დასამზადებლად გამოყენებული ***გერმანიუმისა*** და ***სილიციუმის*** ატომები. ამ შემთხვევაში ატომებს ოთხვალენტიანებს უწოდებენ (გერმანიუმისა და სილიციუმის ატომები სხვა, მაგალითად, ჟანგბადისა და წყალბადის ატომებთანაც ერთიანდება, მაგრამ ეს შეერთებები ჩვენთვის არაა საინტერესო).



ნახ. Დ2.5. *გერმანიუმის (სილიციუმის) ატომი (ა), ალმასისებური კრისტალური სტრუქტურა (ბ), გერმანიუმის კრისტალური ბრტყელი სტრუქტურა (გ)*

ნახ.Დ2.5,ა-ზე ნაჩვენებია თავის მსგავს ატომთან გაერთიანების მსურველი გერმანიუმის ან სილიციუმის ატომი. ელექტრონებს შორის პუნქტურული წრეწირები გამოსახავს ადგილებს, რომლებშიც შეიძლება მოხდეს მეზობელი ატომის ოთხი ელექტრონი.

Დ.2.4. ნახევარგამტარების კრისტალური სტრუქტურა

გერმანიუმისა და სილიციუმის ატომები პერიოდულ ცხრილის იმ ჯგუფშია მოთავსებული, რომელშიც შედის ნახშირბადი. ამიტომ ისინი გაერთიანების დროს წარ-მოქმნის ალმასისებურ კრისტალურ სტრუქტურას, რომლის გამარტივებული გამოსახულება ***Დ2.5,ბ*** ნახაზზეა წარმოდგენილი.

გერმანიუმის ერთი ატომი კუბის ცენტრში, ხოლო დანარჩენი ***4*** ატომი – კუბის წვეროებშია განთავსებული. კუბის ცენტრში განთავსებული ატომი საკუთარი სავალენტო ელექტრონებით უახლოეს მეზობლებთანაა დაკავშირებული. კუთხური ატომები საკუთარ სავალენტო ელექტრონებს აძლევს ცე-

ნტრში განთავებულ და მეზობელ ატომებს, რომლებიც ნახაზზე ნაჩვენები არ არის. ამგვარად, გარე ორბიტები რვა-რვა ელექტრონებამდეა შევსებული. კრისტალურ გისოსში, რა თქმა უნდა, არავითარი კუბი არ არსებობს - იგი ნახაზზე ატომების მოცულობითი განლაგების ნათლად გაგებისთვისაა ნაჩვენები.

ნახევარგამტარების აღწერის მაქსიმალურად გამარტივებისათვის კრისტალური მესერი შეიძლება არა სივრცულად (იხ. ნახ. ***დ2.5,ბ***), არამედ ბრტყელი სახითაც გამოვსახოთ (ნახ. ***დ2.5,გ***).

ასეთ კრისტალში ყველა ელექტრონი საკუთარი სავალენტო კავშირებით ატომებთან მჭიდროდაა მიბმული და თავისუფალი ელექტრონები უბრალოდ არ არსებობს. მათი არარსებობის გამო საქმე თითქოს იზოლატორთან უნდა გვქონდეს, მაგრამ სინამდვილეში ეს ასე არ არის.

დ.2.5. გამტარობის სახეები

საქმე ისაა, რომ ტემპერატურის ზემოქმედებით ზოგიერთი ელექტრონი ახერხებს მოწყდეს საკუთარ ატომებს და დროის გარკვეული მონაკვეთის განმავლობაში ბირთვთან კავშირს თავი დააღწიოს. ამიტომ გერმანიუმის კრისტალში ყოველთვის არსებობს მცირე რაოდენობის თავისუფალი ელექტრონები, რომლებითაც შეიძლება წარმოიშვას ელექტრული დენი. რამდენი თავისუფალი ელექტრონი შეიძლება არსებობდდეს გერმანიუმში ნორმალურ პირობებში?

10^{10} (ათ მილიარდ) ატომში შეიძლება არსებობდეს ორზე არაუმეტესი თავისუფალი ელექტრონი, ამიტომ გერმანიუმი ცუდი გამტარი, ანუ, ნახევარგამტარია. ამავე დროს უნდა აღინიშნოს, რომ სულ რაღაც ერთ გრამი გერმანიუმი შეიცავს 10^{22}, ე.ი. $10^3 \cdot 10^{18}$ ანუ ათას კვინტილიონ ატომს, რის გამოც მასში შეიძლება არსებობდეს დაახლოებით ორი ათასი მილიარდი თავისუფალი ელექტრონი. ეს საკმარისი არ არის დიდი დენის წარმოსაქმნელად, რადგან 1 ამპერი დენის დროს წამში გადის 1 კულონი ელექტრული მუხტი, ანუ $6 \cdot 10^{18}$ (ექვსი კვინტილიონი) ელექტრონი. სამაგიეროდ თბური მოძრაობის წყალობით გერმანიუმს მაინც აქვს ***საკუთარ გამტარობად*** წოდებული მცირე გამტარობა.

ტემპერატურის ამაღლებით ელექტრონები იძენს დამატებით ენერგიას და ამის შედეგად ზოგიერთი ელექტრონი ახერხება საკუთარი ატომებისაგან მოწყვეტას. ისინი გადაიქცევა თავისუფალ ელექტრონებად და გარე ელექტრული ველის არარსებობისას კრისტალის თავისუფალ სივრცეში ქაოტურად მოძრაობს.

ელექტრონდაკარგულ ატომებს მოძრაობა არ შეუძლია და თავისი ნორმალური მდგომარეობის მიმართ კრისტალურ მესერში ოდნავ ირხევა. ელექტრონდაკარგულ ასეთ ატომებს ***დადებითი იონები*** ეწოდება. შეიძლება ჩავთვალოთ, რომ საკუთარი ატომებისაგან მოწყვეტილი ელექტრონების ადგილებზე ჩნდება თავისუფალი ადგილები, რომლებსაც ***ხვრელები*** ეწოდება.

ელექტრონებისა და ხვრელების რაოდენობები ერთმანეთის ტოლია და ამიტომ ხვრელს შეიძლია მიიტაცოს მის ახლოს მოხვედრილი ელექტრონი. ამის შედეგად დადებითი იონი ხელახლა ხდება ნეიტრალური. ხვრელებთან ელექტრონების შეერთების პროცესს ***რეკომბინაცია*** ეწოდება.

ასეთივე სიხშირით ხდება ატომებისაგან ელექტრონების მოწყვეტა, ამიტომ კონკრეტულ ნახევარგამტარში ელექტრონებისა და ხვრელების საშუალო რაოდენობა მუდმივი სიდიდეა და გარე პირობებზე, უპირველეს ყოვლისა ტემპერატურაზეა დამოკიდებული.

ნახევარგამტარის კრისტალზე თუ მოვდებთ ძაბვას, მაშინ ელექტრონების მოძრაობა წესრიგდება და კრისტალში გაჩნდება დენი, რომელიც განპირობებული იქნება ელექტრონულ-ხვრელური (ე.ი. ჯამური) გამტარობით. ასეთი ნახევარგამტარი არ შეიძლება გამოვიყენოთ დიოდების, ტრანზისტორებისა და სხვა ელექტრონული ხელსაწყოების ასაგებად, რადგან მასში შეუძლებელია ***p-n*** გადასასვლელის ფორმირება. ამ გადასასვლელის ფორმირებისათვის აუცილებელია გამოვიყენოთ ორი სხვადასხვა ტიპის ნახევარგამტარი. ერთ-ერთ მათგანს უნდა ჰქონდეს მხოლოდ ***p***-გამტარობისა (***p***-ნიშნავს ***positive***–ს, ანუ დადებითს, ხვრელურს), სოლო მეორეს - ***n***–გამტარობის (***n*** ნიშნავს ***nitive***–ს, ანუ უარყოფითს, ელექტრონულს).

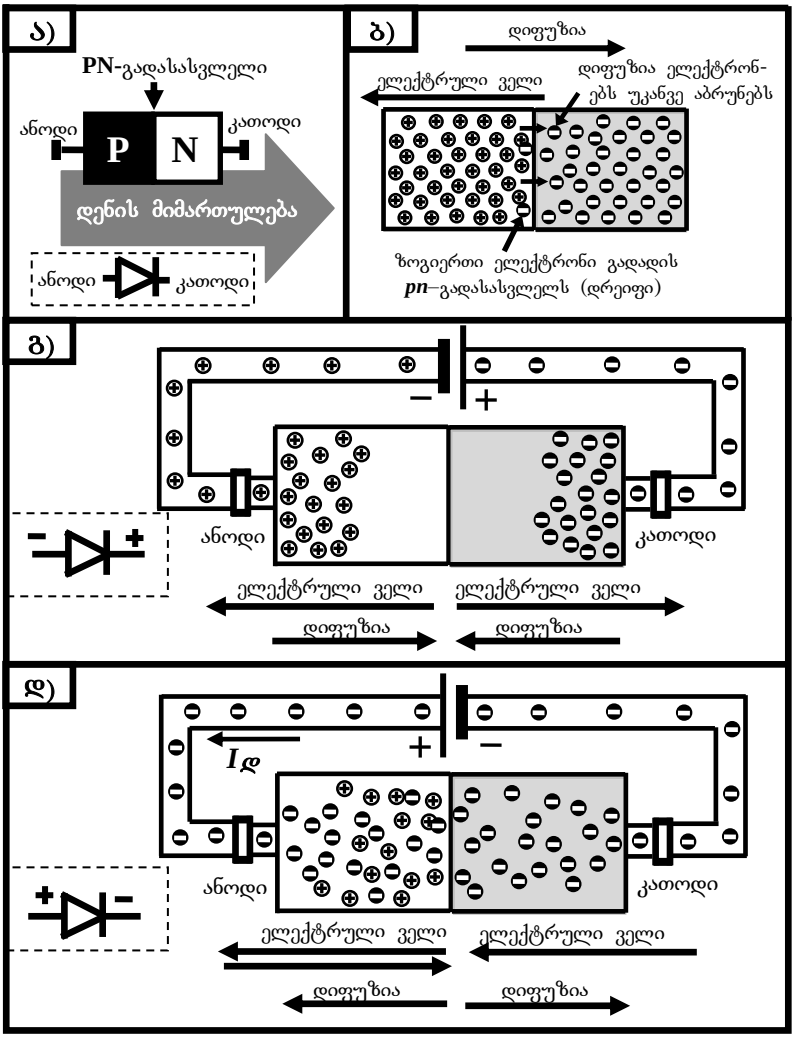
უმნიშვნელო რაოდენობის მინარევების მიუხედავად, მათი არსებობა მნიშვნელოვანწილად ცვლის ნახევარგამტარის თვისებებს და სხვადასხვა გამტარობის მქონე ნახევარგამტარების მიღების საშუალებას იძლევა.

Დ.2.6. ᲓᲘᲝᲓᲘᲡ ᲛᲝᲥᲛᲔᲓᲔᲑᲘᲡ ᲞᲠᲘᲜᲪᲘᲞᲘ ᲓᲐ ᲛᲐᲒᲐᲚᲘᲗᲔᲑᲘ

ნახევარგამტარული მოწყობილობების ოჯახში ***დიოდი*** უმარტივესი ხელსაწყოა. ნახევარგამტარის, მაგალითად, გერმანიუმის ფირფიტის მარცხენა ნაწილში თუ შევიტანთ აქცეპტორულ, ხოლო მარჯვენა ნაწილში – დონორულ მინარევს, მაშინ აღნიშნულ ნაწილებში მივიღებთ შესაბამისად ***p***-ტიპისა და ***n*** ტიპის მახევარგამტარებს. ფირფიტის შუა ნაწილში კი წარმოიშობა ე. წ. ***pn*** გადასასვლელი (ნახ. ***Დ2.6,ა***).

ამ ნახაზის ქვედა ნაწილში პუნქტირითაა შემოხაზული სქემებზე დიოდის პირობითი აღნიშვნა.

ასეთ კრისტალში სხვადასხვა გამტარობის ორი ზონაა, რომლებიდანაც ორი გამომყვანი გამოდის. ამიტომ მიიღო ხელსაწყომ ***დიოდის*** სახელწოდება («***დი***» – ორს ნიშნავს).



ნახ. დ2.6. *ნახევარგამტარული დიოდის სქემები*

p და ***n*** გამტარობებიანი ნახევარგამტარობების შეპირაპირების ადგილზე წარმოშობილი ***pn***-გადასასვლელი ყველა დანარჩენი ნახევარგამტარული ხელსაწყოს საფუძველია. დიოდისაგან განსხვავებით, რომელსაც მხოლოდ ერთი ასეთი გადასასვლელი აქვს, ***ტრანზისტორს*** აქვს ორი, ხოლო ***ტირისტორს*** – სამი ასეთი გადასასვლელი.

დ.2.6.1. *pn* გადასასვლელი სიმშვიდის მდგომარეობაში

განცალკევებულად აღებული ***pn***-გადასასვლელის, ჩვენს შემთხვევაში – ***დიოდის***, შიგნით სიმშვიდის მდგომარეობაშიც მიმდინარეობს საინტერესო პროცესები (***ნახ. დ2.6,ბ***). ასეთი მოძრაობის გამო ***p*** მხარეზე გარკვეულ დონემდე იზრდება ნივთიერების სიმკვრივე. ისევე, როგორც სუნამოს სურნელი მთელ ოთახში ვრცელდება (რასაც დიფუზია ეწოდება), ასევე ნაწილაკებიც ცდილობს მთელ სივრცეში თანაბრად განაწილდეს, ამიტომ, ადრე თუ გვიან, ელექტრონები ხელახლა უბრუნდება ***n*** ზონას.

ელექტროენერგიის მომხმარებელთა უმრავლესობისათვის დენის მიმართულებას მნიშვნელობა არა აქვს - ნათურა ანათებს, უთო ცხელდება და ა.შ. დიოდისათვის დენის მიმართულება ძალიან დიდ როლს თამაშობს. სწორედ ამ თვისებას უზრუნველყოფს ***pn***-გადასასვლელი. განვიხილოთ როგორ იქცევა დიოდი მასთან დენის მიერთების ორი შესაძლო შემთხვევის დროს.

დ.2.6.2. დიოდის უკუმიმართულებით ჩართვა

ნახევარგამტარულ დიოდს თუ კვების წყაროს ისე მივუერთებთ, როგორც ეს ***ნახ დ2.6,გ***-ზეა ნაჩვენები, მაშინ მასში დენი არ გავა. ნახაზის თანახმად კვების წყაროს დადებითი პოლუსი მიერთებულია ***n***, ხოლო უარყოფითი პოლუსი – ***p*** ზონასთან. ამის შედეგად ელექტრონები ***n*** ზონიდან მიისწრაფვის წყაროს დადებითი პოლუსისაკენ. თავის მხრივ ***p*** ზონის დადებით მუხტებს (ხვრელებს) იზიდავს კვების წყაროს უარყოფითი პოლუსი. ამიტომ, როგორც ნახაზიდან ჩანს, გადასასვლელში წარმოიქმნება სიცარიელე და მასში არ რჩება მუხტის მზიდები და უბრალოდ შეუძლებელია დენის წარმოქმნა.

კვების წყაროს ძაბვის გაზრდით იზრდება კვების წყაროს მიერ ელექტრონებისა და ხვრელების მიზიდვის ძალა და გადასასვლელში უფრო მცირდება მუხტის მზიდების რაოდენობა. ამიტომ დიოდის უკუჩართვის დროს დიოდში არ გადის დენი. ასეთი შემთხვევის დროს ამბობენ, რომ ***დიოდი უკუძაბვითაა ჩაკეტილი***. პოლუსების სიახლოვეს ნივთიერების სიმკვრივის გაზრდა წარმოშობს ***დიფუზიას*** – მთელ მოცულობაში ნივთიერების თანაბრ-

ად განაწილებისკენ სწრაფვას, რაც მოხდება დიოდიდან კვების ძაბვის მოხსნის შემდეგ.

ჩაკეტილ დიოდში მაინც გადის უმნიშვნელო სიდიდის დენი, რომელსაც ***უკუდენი*** ეწოდება. ამ დენს წარმოქმნის მუხტის არაძირითადი მზიდები, რომლებიც ძირითადი მზიდებივით, ოღონდ შებრუნებული მიმართულებით მოძრაობს. ბუნებრივია, რომ ასეთი მოძრაობას განაპირობებს უკუძაბვა. უმნიშვნელო რაოდენობის არაძირითადი მზიდები ასევე უმნიშვნელო სიდიდის უკუდენს წარმოქმნის.

კრისტალის ტემპერატურით იზრდება არაძირითადი მზიდების რაოდენობა, რაც ზრდის უკუდენის სიდიდე. ამ დენის სიდიდე შეიძლება იმდენად გაიზარდოს, დაარღვიოს ***pn*** გადასასვლელი. ამიტომ შეზღუდულია ნახევარგამტარული ხელსაწყოების – დიოდების, ტრანზისტორების, მიკროსქემების სამუშაო ტემპერატურის სიდიდე. გადახურებისაგან მძლავრი დიოდებისა და ტრანზისტორების დასაცავად საჭიროა გავითვალისწინოთ სითბოს არინების სპეციალური საშუალებები (ქულერები, რადიატორები და ა. შ.)

დ.2.6.3. დიოდის პირდაპირი მიმართულებით ჩართვა

შევცვალოთ კვების წყაროს ჩართვის პოლარობა (ნახ.6,დ): მინუსი მივუერთოთ ***n*** ზონას (კათოდს), ხოლო მინუსი – ***p*** ზონას (ანოდს). ასეთი ჩართვის დროს ***n*** ზონიდან ელექტრონებს განდევნის ბატარეის მინუსოვანი პოლუსი და ისინი დაიწყებს მოძრაობას ***pn*** გადასასვლელისაკენ. ზონის დადებითად დამუხტულ ხვრელებს ***p*** ზონიდან განდევნის ბატარეის პლუსოვანი პოლუსი. მაშასადამე, ელექტრონები და ხვრელები ურთიერთშემხვედრი მიმართულებით ამოძრავდება.

pn გადასასვლელთან თავს მოიყრის სხვადასვა პოლარობის ნაწილაკი და მათ შორის წარმოიქმნება ელექტრული ველი. ამიტომ ელექტრონები გადალახავს ***pn*** გადასასვლელს და გზას ***p*** ზონის გავლით გააგრძელებს. ამ დროს მათი მცირე ნაწილი რეკომბინირდება ხვრელებთან, მაგრამ დიდი ნაწილი გაეშურება ბატარეის პლუსოვანი პოლუსისაკენ, რის შედეგადაც დიოდში გავა ***Iდ*** დენი (იხ .ნახ. ***დ2.6,დ***)

ამ დენს ეწოდება ***პირდაპირი დენი.*** მის მაქსიმალურ სიდიდეს განსაზღვრავს დიოდის ტექნიკური მონაცემები. პირდაპირი დენის სიდიდე თუ გადააჭარბებს ამ მაქსიმალურ მნიშვნელობას, მაშინ წამოიჭრება დიოდის მწყობრიდან გამოსვლის საშიშროება. ნახაზზე პირდაპირი დენის მიმართულება ემთხვევა საერთოდ მიღებულ მიმართულებას, რომელიც ელექტრონების მოძრაობის საწინააღმდეგოა.

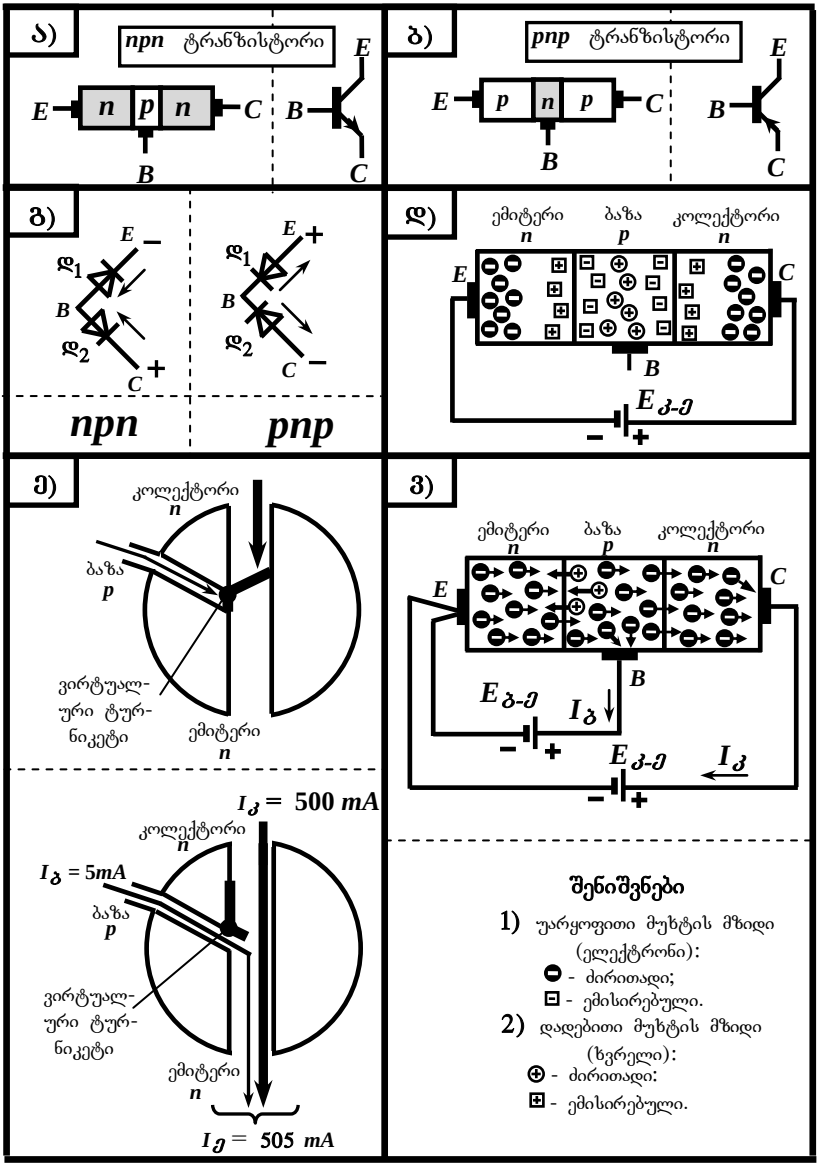
პირდაპირი მიმართულებით დიოდის ჩართვისას მცირეა დიოდის წინაღობა. ეს წინაღობა მრავალჯერ იზრდება დიოდის უკუმიმართულებით ჩართვისას და მასში იმდენად უმნიშვნელო სიდიდის დენი გადის, რომ იგი შეიძლება უგულვებელვყოთ. საბოლოოდ შეიძლება დავასკვნათ, რომ დიოდი იქცევა მექანიკური ვენტილივით, რომელიც წყალს ერთი მიმართულებით ატარებს, ხოლო მეორე მიმართულებით კი არა. ამ თვისების გამო დიოდს ***ნახევარგამტარული ვენტილი*** ეწოდება.

დ.2.7. ბიპოლარული ტრანზისტორის მოწყობილობა და მუშაობა

ზემოთ განხილული ნახევარგამტარული დიოდი თუ ერთი ***pn*** გადასასვლელისაგან შედგებოდა, ტრანზისტორში ამ გადასავლელების რაოდენობა ორის ტოლია. ამიტომ ნახევარგამტარული დიოდი შეიძლება განვიხილოთ ან როგორც ტრანზისტორის წინაპარი, ან როგორც მისი ნახევარი ნაწილი. ბიპოლარული ტრანზისტორის აგებულება ძალიან ჰგავს პურის ორი ნაჭრისაგან დამზადებული სენდვიჩის აგებულებას. პურის ნაჭრების ფუნქციას ტრანზისტორში ასრულებს გარკვეული (***n*** ან ***p***) ნახევარგამტარული მასალისაგან დამზადებული ფირფიტები, ხოლო შიგთავსის ფუნქციას – საწინააღმდეგო ტიპის ნახევარგამტარული მასალისაგან დამზადებული ფირფიტა. მაგალითად, პურის ნაჭრების შესაბამისი ფირფიტები თუ ***n*** ტიპის ნახევრისაგანაა დამზადედული, მაშინ შიგთავსის დასამზადებლად გამოყენებული იქნება ***p*** ტიპის ნახევარგამტარისაგან დამზადებული ფირფიტა და ასეთ ტრანზისტორს მისი შემადგენელი შრეების განთავსების თანამიმდევრობის შესაბამისად ***npn*** ტიპის ტრანზისტორი ეწოდება. ზემოთ აღნიშნული ტიპის ნახევარგამტარებს თუ შევცვლით საწინააღმდეგო ტიპის ნახევარგამტარებით მივიღებთ ***pnp*** ტიპის ტრანზისტორს.

npn ტიპის ტრანზისტორის აგებულება ***დ2.7,ა*** ნახაზზე, ხოლო ***pnp*** ტიპის ტრანზისტორი – ***დ2.7,ბ*** ნახაზზეა გამოხაზული. ამ ნახაზების მარჯვენა ნაწილებში ნაჩვენებია შესაბამისი ტრანზისტორების პირობითი აღნიშვნები. აღნიშნული ნახაზების თანახმად, შიგთავსს ეწოდება ***ბაზა***, ხოლო ამ შიგთავსის შემონაფენებს – ემიტერი და კოლექტორი. აქვე ხაზგასმით უნდა აღვნიშნოთ, რომ ***ტრანზისტორის ნორმალურად მუშაობისათვის ბაზის სისქე გაცილებით ნაკლები უნდა ემიტერისა და კოლექტორის სისქეზე.***

ადრე ნახევარგამტარულ მასალად გამოიყენებოდა გერმანიუმი, მაგრამ უკეთესი ტექნიკური მახასიათებლების გამო დღეს უმეტესწილად სილიციუმისაგან დამზადებული ტრანზისტორები გამოიყენება. გერმანიუმის გამოყენებისას თუ უფრო ტექნოლოგიური იყო ***pnp*** ტიპის ტრანზისტორებ-



ნახ. დ2..7. ტრანზისტორის აგებულებისა და მუშაობის პრინციპის მაილუსტრირებელი ნახაზები

ის წარმოება, სილიციუმის გამოყენების დროს უფრო ტექნოლოგიური გახდა ***npn*** ტიპის ტრანზისტორების წარმოება; ამიტომ დღეისათვის ყველაზე გავრცელებულია ***npn*** ტიპის კაჟბადური ტრანზისტორები და ჩვენ სწორედ მას განვიხილავთ. ყველაფერი, რასაც ვიტყვით ***npn*** ტრანზისტორებზე, სამართლიანი იქნება ***pnp*** ტრანზისტორებისთვისაც, ოღონდ საჭიროა შევცვალოთ ტრანზისტორისა კვების პოლარობა.

ტრანზისტორის მუშაობაში გარკვევისათვის გავიხსენოთ ელექტრონების, ხვრელების, დონორებისა და აქცეპტორების რაობა.

ტრანზისტორების აგებულება და პირობითი გამოსახულებები ***ლ2.7,ა,ბ*** ნახაზებზეა ნაჩვენები; ***ლ2.7,ვ*** ნახაზზე მათში არსებული ***pn***–გადასასვლელები დიოდების სახითაა წარმოდგენილი. იგი დაგვეხმარება ტრანზისტორში მიმდინარე პროცესების ნათლად გააზრებისათვის. დაწვრილებით გავეცნოთ ***npn*** ტიპის ტრანზისტორის შინაგან სამყაროს, რომელიც ***ლ2.7,დ*** ნახაზზეა წარმოდგენილი. მასზე კვების წყაროს პოლუსები ისეა მიერთებული, როგორც ეს ხდება რეალურ სქემებში არსებული ტრანზისტორების დროს. ამ შემთხვევაში ბაზა-ემიტერის გადასასვლელი (ნახაზ ***ლ2.7,ბ*** ნაჩვენები $\mathbf{Д_1}$ დიოდი) გაღებულია და მასში გავა მარცხნიდან მარჯვნივ მოძრავი ელექტრონები. ბაზის გავლის შემდეგ ეს ელექტრონები დაეჯახება ბაზა-ემიტერის დაკეტილ გადასასვლელს (***7,ბ*** ნახაზზე ნაჩვენებ დაკეტილ $\mathbf{Д_2}$ დიოდს), რომელიც შეაჩერებს ამ მოძრაობას: ელექტრონების მოძრაობის გზა დაკეტილია.

კვების წყაროს პოლარობის შეცვლისას დაიკეტება ბაზა-ემიტერის გადასასვლელი (***ლ2.7,ბ*** ნახაზის $\mathbf{Д_1}$ დიოდი), ხოლო გაიღება – ბაზა-კოლექტორის გადასასვლელი (***ლ2.7,ბ*** ნახზის $\mathbf{Д_2}$ დიოდი) და მდგომარეობა არ შეიცვლება: დაკეტილი იქნება ელექტრონების მოძრაობის გზა.

საბოლოოდ შეიძლება დავასკვნათ, რომ ტრანზისტორში დენი არ გავა ემიტერსა და კოლექტორზე ნებისმიერი პოლარობის ძაბვის მოდების დროს.

მიუხედავად იმისა, რომ დაკეტილია ელექტრონების სამოძრაო გზა, მაინც მოიძებნება ტემპერატურის ზემოქმედებით «ძალმოცემული» ზოგიერთი მარდი ელექტრონი, რომელიც დაძლევს პოტენციურ ბარიერსა და იმოძრავებს დაკეტილ გზაზე. ამიტომ ტრანზისტორის ასეთი ჩართვის დროსაც ბაზიდან კოლექტორის მიმართულებით (ე. ი.***ლ7,ბ*** ნახაზზე ნაჩვენები დაკეტილი $\mathbf{Д_2}$ დიოდის გავლით) მაინც იარსებებს ***უმნიშვნელო დენი*** (ემიტერიდან წამოსული დენი ბაზაში მნიშვნელოვნად შესუსტდება და კოლექტორის გავლით გააგრძელებს გზას). იმის გამო, რომ ამ დენის წარმოქმნაში მონაწილეობს ბაზა-კოლექტორზე წარმოშობილი პოტენციური ბარიერის გადამლახავი ***ყველა თავისუფალი ელექტრონი***, ამიტომ მას ***საწყისი გაჯერების დენი*** ეწოდება. იგი უმართავია, არსებობს ყველა ტრანზისტორში და მცი-

რედაა დამოკიდებული გარე ძაბვაზე: დასაშვებ ფარგლებში გარე ძაბვის მაქსიმალურად გაზრდისას იგი უმნიშვნელოდ იზრდება. სამაგიროდ აღნიშნული დენის სიდიდეზე მნიშვნელოვან როლს ასრულებს ტემპერატურა.

ტემპერატურის შემდგომი ამაღლება გაზრდის გაჯერების საწყის დენს, რომელიც დამატებით გააცხელებს ბაზა-კოლექტორის გადასასვლელს ($\mathbf{Д}_2$ დიოდს). ასეთ ტემპერატურულ არასტაბილურობას შეუძლია გამოიწვიოს სითბური გარღვევა და ტრანზისტორი მწყობრიდან გამოიყვანოს. ამიტომ საჭიროა მივიღოთ ტრანზისტორის გაგრილებისათვის საჭირო ზომები და მაღალი ტემპერატურის დროს ტრანზისტორზე არ მოვდოთ ზღვრული ძაბვები.

ზემოთ განხილულ სქემებში ბაზაზე არ იყო მოდებული ძაბვა. ტრანზისტორის ასეთი ჩართვა პრაქტიკულად არ გამოიყენება. აუცილებელია ბაზაზე მოდებული იყოს ემიტერის ძაბვაზე მცირე ისეთი პოლარობის ძაბვა, რომლის მეშვეობითაც პირდაპირ იქნება წანაცვლებული ბაზა-ემიტერის გადასასვლელი (ნახ. ***Д2.7,ვ*** ნახაზზე შესაბამისი $\mathbf{Д}_1$ დიოდი). ტრანზისტორის ასეთი სწორი ჩართვა ***Д2.7,ე*** ნახაზზეა მოყვანილი.

დიოდის შემთხვევაში თუ ყველაფერი ძალიან მარტივადაა მოწყობილი – გაღების შემდეგ მასში გადის დენი, ხოლო ტრანზისტორში სხვა მოვლენებიც ხდება. ემიტერულ დენს $\boldsymbol{n}$ გამტარობის მქონე $\boldsymbol{E}$ ემიტერს ელექტრონები შეაქვს $\boldsymbol{p}$ გამტარობის $\boldsymbol{B}$ ბაზაში (იხ. ნახ. ***Д2.7,ე***). ელექტრონების ნაწილი შეავსებს ბაზაში არსებულ ხვრელებს (მოხდება მათი რეკომბინაცია) და გაჩნდება ბაზის უმნიშვნელო $\boldsymbol{I_ბ}$ დენი (იხ. ნახ. ***Д2.7,ე***). ვინაიდან ***ბაზა თხელია***, ამიტომ ***მცირეა მასში არსებული ხვრელების რაოდენობაც*** და $\boldsymbol{E}$ ემიტერიდან ბაზაში შემოსული ელექტრონების მხოლოდ მცირე ნაწილის რეკომბინაცია მოხდება. დანარჩენი დიდი რაოდენობის ელექტრონები ბაზიდან ამოიღება.

ამგვარად, ბაზა-ემიტერის გადასასვლელზე (იხ. ნახ.***დ.3.7,ე***), ანუ ამ გადასავლელით წარმოქმნილ $\mathbf{Д}_1$ დიოდზე (იხ. ***ნახდ.3.7,ვ***) მოდებული ძაბვის მცირე $\boldsymbol{E_{ბ-ე}}$ წყარო ეხმარება ელექტრონებს დაძლიოს უკუწანაცვლებული (დაკეტილი) ბაზა-კოლექტორის ($\mathbf{Д}_2$ დიოდის) პოტენციური ბარიერი და $\boldsymbol{I_კ}$ დენის სახით დაბრუნდეს $\boldsymbol{E}$ ემიტერში. ამას ეწოდება ***ტრანზისტორული ეფექტი,*** რომლის აღმოჩენისათვის ***უ. შოკლიმ***, (***ჯ. ბარდინთან*** და ***უ. ბრაიტენთან*** ერთად) ***1956*** წელს მიიღო ნობელის პრემია. ეს ეფექტი სამეცნიერო ლიტერატურაში ***შოკლის ტრანზისტორული ეფექტის*** სახელით შევიდა.

შოკლის ტრანზისტორულ ეფეტის არსში გასარკვევად განვიხილოთ ***Д2.7,ვ*** ნახაზი. ამ ნახაზის ზედა ნაწილში გამოხატული სქემა ახდენს ***Д327,დ*** ნახაზზე გამოსახული ტრანზისტორის შინაგანი არხების მდგომარეობის ილუსტრირებას: ვირტუალური ტურნიკეტის ფრთებით თავისუფალ

ელექტრონებს გადაკეტილი აქვს სამოძრაო გზები და ტრანზისტორში დენი არ გადის.

დ.3.7,ე ნახაზის ქვემო ნაწილზე გამოსახული სქემა ახდენს ***დ.3.7,ვ*** ნახაზზე გამოსახული ტრანზისტორის შიგნით მიმდინარე პროცესების ილუსტრირებას. პირდაპირ წანაცვლებულ ***ბაზა-ემიტერში*** გამავალი სუსტი $I_{ბ}$ დენი ვირტუალური ტურნიკეტის მოკლე ფრთაზე ზემოქმედებით თითქოს «შეატრიალებს ტურნიკეტს» და გზას გაუხსნის $I_{ბ}$ და $I_{კ}$ დენებს. სწორედ ამ ვირტუალური ტურნიკეტის მოქმედება ახდენს ***შოკლის ტრანზისტორული ეფექტის ილუსტრირებას***: სუსტი დენი გზის გახსნაში ეხმარება მძლავრ დენს! ამის შემდეგ ყველაფერი კანონზომიერად გრძელდება: ერთმანეთს შეერწყმება $I_{ბ}$ და $I_{კ}$ დენები და ტრანზისტორში გაივლის მძლავრი ემიტერული $I_{ე}\approx I_{კ}+I_{ბ}$ დენი. მიახლოების ნიშანი აქ იმის გამო დგას, რომ ბაზაში მცირე დანაკარგების გამო $I_{ე}$ დენი იმდენად უმნიშვნელოდაა ნაკლები $I_{კ}+I_{ბ}$ დენზე, რომ ისინი შეიძლება თითქმის ერთმანეთის ტოლად ჩავთვალოთ.

დ.2.8. ტრანზისტორის მუშაობა საბასალებო რეჟიმში

ტრანზისტორზე ძაბვას თუ არ მივაწოდებთ, ან ბაზისა და ემიტერის გამომყვანებს ერთმანეთთან შევაერთებთ, მაშინ $U_{ბ-ე}$ ძაბვა ნულის ტოლი გახდება და, აქედან გამომდინარე, არ იქნება ბაზის $I_{ბ}$ დენი. ტრანზისტორი დაიკეტება, კოლექტოლური (სწორედ ის საწყისი) დენი იმდენად უმნიშვნელო იქნება, რომ შეიძლება უგულვებელვყოთ. ამ შემთხვევისას ამბობენ, რომ ტრანზისტორი იმყოფება ***წაკვეთის*** მდგომარეობაში, ე .ი. დაკეტილია.

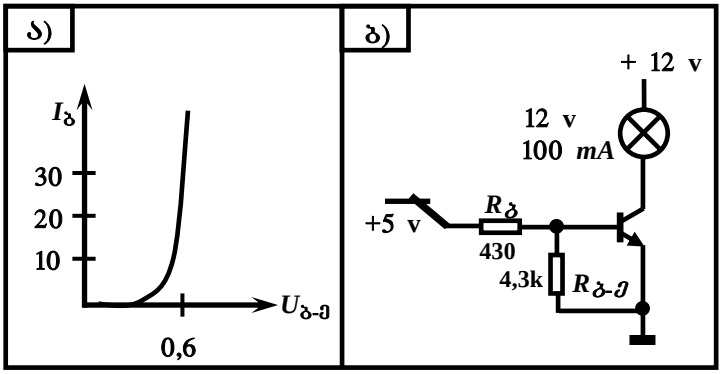
წაკვეთის საწინააღმდეგო მდგომარეობას ***გაჯერების*** მდგომარეობა ეწოდება. ეს ხდება ტრანზისტორის მთლიანად გაღებისას, როდესაც უფრო მეტად გაღება შეუძლებელია. ასეთი გაღების დროს კოლექტორ-ემიტერის უბნის წინაღობა უმნიშვნელოა და კოლექტორულ წრედში რეზისტორის გაუთვალისწინებლად ტრანზისტორის ჩართვისას იგი მყისიერად დაიწვება.

წაკვეთისა და გაჯერების მდგომარეობების არსში გასარკვევად შეიძლება გამოვიყენოთ ***დ2.7,ე*** ნახაზის ქვედა ნაწილზე გამოსახული სქემა. აღნიშნულ მდგომარეობებებს შეესაბამება წარმოსახვითი ტურნიკეტის განაპირა მდგომარეობები. კერძოდ, ***ტრანზისტორი იქნება წაკვეთის მდგომარეობაში***, თუ ტურნიკეტის ორივე (პატარა და დიდი) ფრთა სრულად ჩაკეტავს კონტროლირებად არხებს და პირიქით, ***ტრანზისტორი იქნება გაჯერების მდგომარეობრეობაში***, თუ ტურნიკეტის ორივე ფრთა სრულად გააღებს აღნიშნულ არხებს. პირველ შემთხვევაში აღნიშნული არხების წინაღობები მაქსიმალურია (თითქმის უსასრულობის ტოლია), ხოლო მეორე შემთხვევაში

- მინიმალური (თითქმის ნულის ტოლი). დანარჩენ შემთხვევებში ტრანზისტორი იქნება ნაწილობრივ ღია (არც წაკვეთისა და არც გაჯერების) მდგომარეობაში.

ამ მდგომარეობამდე ტრანზისტორის მისაყვანად საჭიროა არსებობდეს საკმაოდ მაღალი ბაზისური $I_ბ$ დენი, რისთვისაც ბაზის-ემიტერის გადასასვლელზე მოდებული $U_{ბ-ე}$ ძაბვა ***0,6...0,7*** ვოლტის რანგის უნდა იყოს. შემზღუდავი რეზისტორის არარსებობისას ბაზი-ემიტერის გადასასვლელისათვის ასეთი სიდიდის ძაბვა ძალიან დიდია. მართლაც, ტრანზისტორის შესასვლელი მახასიათებელი (***ნახ.Დ2.8,ა***) ძალიან ჰგავს დიოდის მახასიათებლის პირდაპირ შტოს.

ეს ორი (წაკვეთისა და გაჯერების) მდგომარეობა გამოიყენება ტრანზისტორის ***საგასაღებო მდგომარეობაში (ციფრულ ტექნიკაში) მუშაობისას***, როდესაც ტრანზისტორი ასრულებს ჩვეულებრივი რელეს კონტაქტის როლს. ***ანალოგურ ტექნიკაში*** ტრანზისტორის გამოყენებისას, პირიქით, იღებენ სპეციალურ ზომებს გაჯერების მდგო-მარეობაში ტრანზისტორის გადასვლის გამოსარიცხავად.



ნახ. Დ2.8. *ტრანზისტორის შესასვლელი მახასიათებელი (ა), საგასაღებო კასკადის სქემა (ბ)*

გასაღების მდგომარეობაში მუშაობის ძირითადი არსი ისაა, რომ ბაზის მცირე დენი მართავს კოლექტორის მასზე რამდენიმე ათეულჯერ მეტ დიდ დენს. მიუხედავად იმისა, რომ კოლექტორის დიდი დენი ენერგიის გარე წყაროდან მიიღება, ***დენის მიხედვით გაძლიერების*** ფაქტი მაინც გვაქვს. მარტივი მაგალითი: მცირე მიკროსქემას შეუძლია დიდი ნათურის ან ძრავის ჩართვა.

საგასაღებო რეჟიმში ტრანზისტორის ასეთი გაძლიერების სიდიდის განსაზღვრისათვის გამოიყენება «***დიდი დენის რეჟიმში დენის მიხედვით გაძლი-***

ერების β კოეფიციენტი». საგასაღებო რეჟიმში მუშაობისას ეს კოეფიციენტი არავითარ შემთხვევაში არ არის ***10...20-ზე*** ნაკლები.

β განისაზღვრება, როგორც კოლექტორის შესაძლო მაქსიმალური დენის ფარდობა ბაზის შესაძლო მინიმალურ დენთან. სიდიდე უგანზომილებოა და გვიჩვენებს «თუ რამდენჯერაა დიდი»:

$$\beta \geq \frac{I_{კmax}}{I_{ბ\square\square\square min}} \quad . \qquad (1)$$

დიდი უბედურება არ იქნება, დენის სიდიდე მოთხოვნილზე მეტი რომ იყოს: ტრანზისტორი გავჯერებულ მდგომარეობაშია (ნახ ***დ2.7,ე***-ზე ვირტუალური ტურნიკეტი ბოლომდეა შეტრიალებული, ე.ი. მთლიანად აქვს გაღებული არხები) და ამიტომ იგი უფრო მეტად ვერ გაიღება. ჩვეულებრივი ტრანზისტორების გარდა საგასაღებო რეჟიმში მუშაობისათვის გამოიყენება «დარლინგტონური» ანუ შედგენილი ტრანზისტორები, რომელთა «***სუპერ ბეტა***» შეიძლება ***1000***-ის ტოლი და უფრო მეტიც იყოს.

საგასაღებო კასკადის განხილვა უსაბუთო რომ არ გამოგვივიდეს, განვიხილოთ ***დ2.8,ბ*** ნახაზზე ნაჩვენები საგასაღებო კასკადის მუშაობა.

კასკადი დანიშნულებაა ჩართოს და ამორთოს ***ნათურა***. ცხადია, დატვირთვად ნათურის ნაცვლად შეიძლება გამოვიყენოთ ნებისმიერი ელექტრული ხელსაწყო: რელე, ელექტროძრავა, ელექტროქურა და ა. შ. ამოცანის თავისებურებაა ის, რომ ნათურის სრული ვარვარებისათვის საჭიროა სათანადოდ შევირჩიოთ ბაზის წრედში ჩართულ რეზისტორის წინაღობის სიდიდე.

ასეთი ნათურები გამოიყენება, მაგალითად, ავტომობილში ხელსაწყოთა დაფის განათებლად. ***1,5*** ამპერიანი ***KT815*** ტრანზისტორი ჩვენი ხელსაწყოსათვის სრულიად საკმარისია.

ყველაზე საინტერესოა ის ფაქტი, რომ გამოთვლებში ძაბვას ყურადღება არ ექცევა და მხოლოდ პირობა ***(1)***-ის დაცვა მოწმდება. ამიტომ ნათურა შეიძლება იკვებებოდეს ***200*** ვოლტი ძაბვით, ხოლო ტრანზისტორის ბაზის წრედი იმართებოდეს ***5*** ვოლტი კვების მქონე მიკროსქემით. ჩვენს მაგალითში მიკროსქემა არ არსებობს და ბაზის წრედი იმართება კონტაქტით, რომელზედაც მოდებულია ***5*** ვოლტი ძაბვა. ნათურის კვების ძაბვაა ***12*** ვოლტი და იგი მოიხმარს ***100*** მილიამპერ დენს. ივარაუდება, რომ ***β=10***. ბაზა-ემიტერის გადასასვლელზე ძაბვის ვარდნა $U_{ბ-ე}=0,6$ ვოლტს (იხ.ნახ. ***დ2.8,ა***). ასეთი მონაცემების დროს:

- ბაზის დენის სიდიდე უნდა იყოს $I_ბ = I_კ/\beta = 100/10 = 10$ მილიამპერი.
- ბაზის $R_ბ$ რეზისტორზე მოდებული ძაბვა შეადგენს:
  $$5V - U_{ბ-ე} = 5V - 0,6V = 4,4V.$$
- ომის კანონის თანახმად:

$R = U / I = 4,4$ ვოლტი / $0,01$ ამპერი = **440** ომს.

სტანდარტული ტრანზისტორების ჩამონათვლიდან ვირჩევთ ***430*** ომი წინაღობიან რეზისტორს და ამით მთავრდება გამოთვლის პროცესი. ამ პროცესის დროს არ გამოგვითვლია ბაზასა და ემიტერს შორის ჩართული $R_{ბ-ე}$ რეზისტორის წინაღობა, რომლის დანიშნულება არის განრთული ღილაკის დროს საიმედოდ დაკეტოს ტრანზისტორი. ეს რეზისტორი რომ არ არსებობდეს ნებისმიერი დაბრკოლება ტრანზისტორზე აუცილებლად მოახდენდა ზეგავლენას. ამ მხრივ იგი შეიძლება დეტექტორული რადიომიმღების ანტენას შევადაროთ.

ტრანზისტორის საიმედოდ დასაკეტად, წაკვეთის მდგომარეობაში მის გადასაყვანად აუცილებელია ურთიერთტოლი იყოს ემიტერისა და ბაზის პოტენციალები. ჩვენი «სასწავლო სქემისათვის» უფრო მოსახერხებელი იქნებოდა ისეთი გადამრთველი კონტაქტი გამოგვეყენებინა, რომლითაც ნათურას ანთებისათვის მივუერთებდით ***+5*** ვოლტ ძაბვას, ხოლო ჩაქრობისათვის – მთელ კასკადს გადავრთავდით «მიწაზე».

ყველგან და ყოველთვის ამის ფუფუნება არ გვექნება. ამიტომ უფრო ადვილია $R_{ბ-ე}$ რეზისტორის დახმარებით ერთმანეთს გავუტოლოთ ბაზისა და ემიტერის პოტენციალები. ამ რეზისტორის ნომინალის გამოთვლა საჭირო არ არის. მისი წინაღობა ***10***-ჯერ მეტი უნდა იყოს $R_{ბ}$ რეზისტორის წინაღობაზე. პრაქტიკული მონაცემების თანახმად მისი სიდიდე ***5...10*** კილოომის ტოლი უნდა იყოს.

განხილული სქემა წარმოადგენს საერთო ემიტერიანი სქემის ნაირსახეობას. აქ უნდა აღინიშნოს შემდეგი ორი თავისებურება. ***ჯერ ერთი,*** ესაა მმართველ ძაბვად ***+5*** ვოლტი ძაბვის გამოყენება: სწორედ ასეთი ძაბვა გამოიყენება მაშინ, როდესაც საგასაღები კასკადი უერთდება ციფრულ მიკროსქემებს ანუ, უფრო ხშირად, მიკროკონტროლერებს. ***მეორეც,*** კოლექტორზე არსებული სიგნალი ბაზაზე არსებულ სიგნალის მიმართ ინვერტირებულია (ე.ი. შესასვლელი სიგნალი ინვერტირდება). კონტაქტით ბაზაზე ***+5*** ბაზაზე ძაბვის მიწოდებისას კოლექტორზე მოდებული ძაბვა თითქმის ნულამდე ვარდება. ამ დროს ვიზუალურად ნათურა არ ინვერტირდება: ბაზაზე თუ სიგნალი არსებობს, მაშინ ნათურა ანთია. შევნიშნავთ, რომ შესასვლელი შესასვლელი სიგნალი ინვერტირდება ტრანზისტორის ანალოგურ სქემებში გამოყენების დროსაც.

დანართი 3. ციფრული ელექტრონული სქემები

დ.3.1. ველის ტრანზისტორების აგებისა და ფუნქციონირების საკითხები

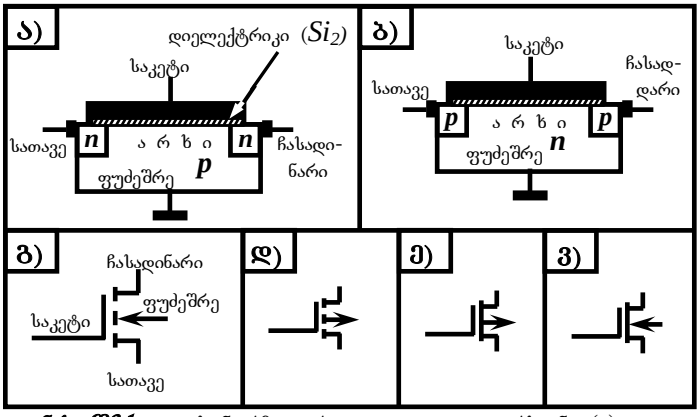
ციფრული ელექტრონული სქემების ასაგებად გამოყენებულ მასობრივად გამოიყენება ***pn***-დიოდები, ბიპოლარული და ველის ტრანზისტორები. აღნიშნული ელემენტებიდან პირველი ორი მათგანის აგებისა და ფუნქციონირების პრინციპები მეტ-ნაკლებად სრულად განვიხილეთ დანართ ***2***-ში. ახლა შევეხებით ველის ტრანზისტორეების აგებისა და ფუნქციონირების პრინციპებს.

<u>ველის ტრანზისტორი</u> ნახევარგამტარული ხელსაწყოა, რომლის ორ გამომყვანს შორის არსებული უბნის წინაღობა იცვლება მესამე გამომყვანზე მოდებული პოტენციალით. ***დ3.1,ა*** ნახაზზე მოყვანილია ველის ტრანზისტორის ერთ-ერთი ნაირსახეობა, რომელსაც ***<u>ლითონ-ჟანგეულ-ნახევარგამტარის</u>***, ანუ შემოკლებით - ***ლჟნ*** ტიპის ტრანზისტორი ეწოდება. მის ასაგებად გამოყენებულია ორი ნახევარგამტარული უბნის მქონე კრისტალი, რომელთაგანაც ერთ-ერთი უბანი ***n***, ხოლო მეორე – ***p*** ტიპისაა. ამ კრისტალს ***ფუძეშრე*** ეწოდება.

ფუძეშრის ერთ-ერთ უბანს ეწოდება ***სათავე***, ხოლო მეორეს – ***ჩასადინარი***. სათავე ფუძეშრისა ის უბანია, რომლიდანაც სათავეს იღებს მუხტის მზიდების ნაკადი, ხოლო ჩასადინარი – ის უბანი, რომელშიც აღნიშნული ნაკადი ჩაედინება. სათავესა და ჩასადინარს შორის არსებულ სივრცეს ***არხი*** ეწოდება. არხის ზემოთ განთავსებულია ***საკეტად*** წოდებული ლითონის ელექტროდი. იგი მთლიანად ფარავს არხს, ოღონდ მისგან დიელექტრიკის (ჩვეულებრივ კაჟბადის ორჟანგის) თხელი შრითაა იზოლირებული. ვინაიდან ხელსაწყო შეიცავს ***ლითონის*** საკეტს, მაიზოლირებელი ***ჟანგეულის*** შრესა და ***ნახევარგამტარის*** ფუძეშრეს, ამიტომ მან მიიღო ლითონ-ჟანგეულ-ნახევარგამტარის ტიპის ხელსაწყოს სახელწოდება.

სამუშაო რეჟიმში დენი სათავიდან არხის მეშვეობით მიედინება ჩასადინარისაკენ. ეს დენი სათავისა და ჩასადინარისათვის მუხტის ძირითადი მზიდებისაგანაა წარმოქმნილი. მაგალითად, ***დ3.1,ა*** ნახაზზე მოყვანილი ტრანზისტორში სათავედ და ჩასადინრად გამოყენებულია ***n*** ტიპის ნახევარგამტარი, რომელშიც მუხტის ძირითადი მზიდები ელექტრონებია და ამიტომ ამ ნახევარგამტარში ზემოთ აღნიშნულ დენი ელექტრონების მოწესრიგებული მოძრაობის შედეგად უნდა წარმოიქმნას. ნორმალურ პირობებში ***p***-ტიპის ნახევარგამტარისაგან წარმოქმნილ არხში მცირე რაოდენობითაა თავისუფალი

(მოძრავი) ელექტრონები და ამიტომ მაღალია მისი წინაღობა. მდგომარეობა შეიცვლება საკეტზე ჩამიწებული ფუძეშრის მიმართ დადებით პოტენციალის მოდებით. ამ პოტენციალის წყალობით საკეტზე გაჩნდება დადებითი მუხტები, რომლის წყალობით ელექტრონები ფუძეშრის ფართო სივრციდან გადაინაცვლებს არხში და შეამცირებს მის წინაღობას.



ნახ. 3.1. ველის ნორმალურად დაკეტილი ***n***–არხიანი (ა) და ***p***–არხიანი (ბ) ტიპის ***ლჟნ***-ტრანზისტორი, ***ლჟნ***- ტრანზისტორების სტანდარტული აღნიშვნები **(ბ, გ, დ, ე, ვ)**

საკეტზე მოდებული ძაბვისათვის არსებობს გარკვეული ზღურბლური მნიშვნელობა, რომლის დროსაც მკვეთრად მცირდება არხის წინაღობა. საკეტზე მოდებული ძაბვის სიდიდე თუ ამ ზღურბლურ მნიშვნელობაზე ნაკლებია, მაშინ ძალიან დიდია არხის წინაღობა, რაც მნიშვნელოვნად ამცირებს მასში გამავალი დენის სიდიდეს და ამიტომ თვლიან, რომ ***ტრანზისტორი დაკეტილია.*** საკეტზე მოდებული ძაბვის სიდიდე თუ აღემატება ზღურბლურ მნიშნელობას, მაშინ არხის წინაღობა უმნიშვნელო ხდება და ითვლება, რომ ***ტრანზისტორი გაღებულია***.

ჩვენ მიერ აღწერილი ტრანზისტორზე ამბობენ, რომ იგი მუშაობს ***გამდიდრების რეჟიმში***, რამდენადაც საკეტზე მოდებული ძაბვა ზრდის არხში ელექტრონების რაოდენობას, ე. ი. არხს ***ელექტრონებით ამდიდრებს.*** ნორმალურად ამ ტრანზისტორის საკეტზე ძაბვა მოდებული არ არის, რის გამოც იგი დაკეტილია. ამიტომ მას ველის ***ნორმალურად დაკეტილ ტრანზისტორსაც*** უწოდებენ. იგი შეიძლება შევადაროთ რელეს კონტაქტს, რომელიც რელეს გრაგნილზე ძაბვის არარსებობისას წყვეტს წრედს.

არსებობს ველის ნორმალურად ღია ტრანზისტორებიც, რომლის საკეტზე ძაბვის არარსებობის (ანუ, ნულოვანი ძაბვის არსებობის) დროს არხს შედარებით დაბალი წინაღობა აქვს. ასეთ ტრანზისტორში, როგორც სათავისა და ჩასადინარის, ისე ფუძეშრის ასაგებად ერთი და იგივე ტიპის ნახევარგამტარული მასალა გამოიყენება. ***Დ3.1,ა*** ნახაზზე ნაჩვენებ ტრანზისტორში ფუძეშრისათვის გამოიყენება ***p***, ხოლო სათავისა და ჩასადინარისათვის - ***n*** ტიპის ნახევარგამტარული მასალა. სამივეს შესაქმნელად თუ ***n*** ტიპის ნახევარგამტარული მასალას გამოვიყენებთ, მაშინ მივიღებთ გაღარიბების რეჟიმში მომუშავე ველის ტრანზისტორს. ნორმალურად, ე. ი. საკეტზე ***0***-ის ტოლი ძაბვის არსებობისას მისი არხის წინაღობა უმნიშვნელოა და მასში დენი უმნიშვნელო დანაკარგით გაივლის ანუ ***ტრანზისტორი ღიაა.*** საკეტზე ზღურბლურ სიდიდეზე მეტი ძაბვის მოდებისას არხიდან განიდევნება ელექტრონები, ე.ი. არხი ელექტრონებისაგან გაღარიბდება, მნიშვნელოვნად გაიზრდება მისი წინაღობა ე. ი. ***ტრანზისტორი დაიკეტება.*** ველის მაშასადამე გაღარიბების რეჟიმში მომუშავე ტრანზისტორი ***ნორმალურად*** (საკეტზე ძაბვის არარსებობისას) ***დაკეტილია.***

Დ3.1,ა ნახაზზე ნაჩვენებ ტრანზისტორის კონფიგურაციის მქონე როგორც დაკეტილ, ისე ღია ველის ტრანზისტორებში დენს ელექტრონები წარმოქმნის და ამიტომ მათ ***n–არხიანი ხელსაწყოები*** ეწოდება. ამ ტრანზისტორების მუშაობის ჩვეულებრივი რეჟიმის დროს სადინართან შეფარდებით სათავეს პოტენციალი დადებითია (მაღალია). არსებობს ***ველის p–არხიანი ტრანზისტორებიც***. მათში დენი ხვრელური გამტარობის ხარჯზე წარმოიქმნება. ისინი ***n–არხიანი ხელსაწყოებივით*** მუშაობს, ოღონდ იმ განსხვავებით, რომ ყველა ძაბვას საწიონააღმდეგო პოლარობა უნდა ჰქონდეს.

Დ3.1,ბ ნახაზზე მოყვანილია ველის ნორმალურად დაკეტილი ***n–არხიანი ლჟნ***- ტრანზისტორის აგებულება.

არსებობს ორი (***n*** და ***p***) ტიპის არხი და ნორმალურ პირობებში შესაძლო ორი (ნორმალურად დაკეტილი და ნორმალურად ღია) მდგომარეობა. მათი კომბინაციით შეიძლება წარმოიქმნეს შემდეგი ოთხი ტიპის ***ლჟნ*** ტრანზისტორი: ***1. n***–არხიანი ნორმალურად დაკეტილი (გამდიდრების რეჟიმში მომუშავე) ***ლჟნ*** ტრანზისტორი (ნახ. ***Დ3.1,გ***); ***2. p***–არხიანი ნორმალურად დაკეტილი (გამდიდრების რეჟიმში მომუშავე) ***ლჟნ*** ტრანზისტორი (ნახ. ***Დ3.1,დ***); ***3. n***–არხიანი ნორმალურად ღია (გაღარიბების რეჟიმში მომუშავე) ***ლჟნ*** ტრანზისტორი (ნახ. ***Დ3.1,ე***); ***4.*** . ***p***–არხიანი ნორმალურად ღია (გაღარიბების რეჟიმში მომუშავე) ***ლჟნ***- ტრანზისტორი (ნახ. ***Დ3.1,ვ***).

ველის ***ლჟნ*** ტრანზისტორებით აგებული სქემების სწრაფმოქმედებას ძირითადად განსაზღვრავს საკეტისა და სხვა ელექტროდებს შორის წარმოქმნილ ტევადობებათა სიდიდეები. ხელსაწყოს ერთი მდგომარეობიდან მე-

ორეში გადასვლამდე ამ ტევადობებმა უნდა მოასწროს დამუხტვა და განმუხტვა. სიტუაციას ისიც ართულებს, რომ ბიპოლარულ ტრანზისტორებთან შედარებით ველის ტრანზისტორებს გამტარ მდგომარეობებში უფრო მაღალი წინაღობები აქვს. ველის ტრანზისტორი, როგორც წესი, მისი ჩართვის წრედში არსებული ველის მეორე ტრანზისტორით იმართება, რაც მნიშვნელოვნად ზრდის მისი ტევადობის დამუხტვის დროს. ამ მიზეზის გამო ***ლშნ*** ტრანზისტორებით აგებული სქემები ზოგადად ბიპოლარული ტრანზისტორებითა და დიოდებით აგებულ სქემებზე უფრო ნელმომქედია.

დ.3.2. ლოგიკური ელემენტები

ლოგიკური ელემენტები ანუ ***ვენტილები*** წარმოადგენს ელემენტალურ ლოგიკურ ფუნქციათა მარეალიზებელ დისკრეტულ მოწყობილობებს, რომლებიც რთული დისკრეტული მოწყობილობების ასაგებად საჭირო საელემენტო ბაზის შესაქმნელად გამოიყენება. მინიმალური რაოდენობის ლოგიკური ელემენტების ერთობლიობას, რომლის გამოყენებითაც ნებისმიერი სირთულის დისკრეტული მოწყობილობის სინთეზია შესაძლებელი, ***ლოგიკურ ელემენტთა ფუნქციონალურად სრული სისტემა*** ეწოდება.

ნებისმიერი სირთულის დისკრეტული მოწყობილობის სინთეზის თეორიული პრობლემების გადასაწყვეტად გამოიყენება ლოგიკურ ელემენტების ფუნქციურად სრული სისტემა, რომელიც შედგება კონიუნქტორის (***და***-ელემენტის), დიზიუნქტორისა (***ან***-ელემენტისა) და ინვერტორისაგან (***არა***-ელემენტისაგან), ე. ი. იგი ***სამი ელემენტისაგან შემდგარი სისტემაა.*** რაც შეეხება რთული დისკრეტული მოწყობილობის პრაქტიკულ რეალიზებას, ამ მიზნისათვის გამოიყენება ***ერთი ლოგიკური ელემენტისაგან წარმოქმნილი სრული სისტემა*** და ეს ელემენტი შეიძლება იყოს ***და-არა*** ელემენ-ტი ან ***ან-არა*** ელემენტი.

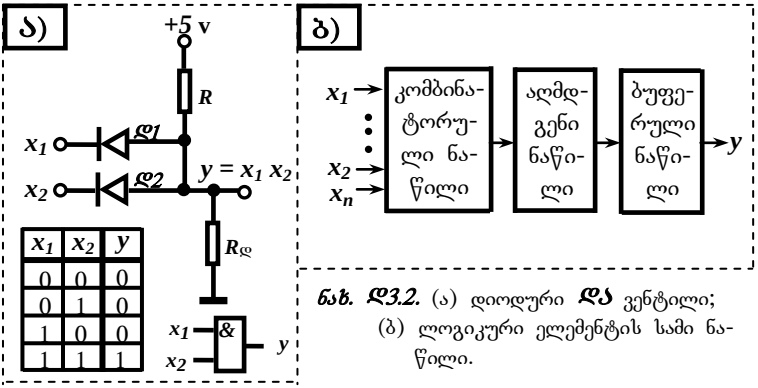
ზემოთ აღნიშნულიდან გამომდინარე, რთული ლოგიკური ფუნქციების მარეალიზებელი დისკრეტული მოწყობილობები აიგება ლოგიკური ელემენტებით. თავად ლოგიკური ელემენტები შეიძლება ავაგოთ ელექტრომაგნიტური ელემენტების კონტაქტებით ან უკონტაქტო ელექტრონული ელემენტებით – დიოდებით ან ტრანზისტორებით. ***დ4.2,ა*** ნახაზზე მოყვანილია დიოდებით აგებული ***და***-ვენტილის სქემა. მის ორივე x_1 და x_2 შესასვლელზე მაღალი (***5*** ვოლტის ტოლი) პოტენციალის, ე.ი. ლოგიკური სიგნალ ***1***-ის არსებობისას ***y*** გამოსასვლელზედაც იარსებებს ლოგიკური ***1***. თუნდაც ერთ შესასვლელზე ნულოვანი პოტენციალის (ლოგიკური ***0***-ის) არსებობისას შესაბამისი დიოდი პირდაპირ წანაცვლდება, ე.ი. მისი წინაღობა თითქმის ***0***-ის ტოლი გახდება და გამოსასვლელზე დამყარდება ***0***-თან ახლომდებარე

პოტენციალი (ლოგიკური ***0***). ამ სქემის მოქმედებას გამოხატავს ***დ3.2,ა*** ნახაზზე მოყვანილი ჭეშმარიტობის ცხრილი, რომელიც ***და***-ფუნქციას შეესაბამება.

დიოდური ვენტილის გამოყენება შეიძლება ზოგჯერ სასარგებლო იყოს, მაგრამ ლოგიკურ ელემენტად მის ფართოდ გამოყენებას ხელს უშლის მისთვის დამახასიათებელი შემდეგი ორი ნაკლი.

1. მის გამოსასვლელზე ფორმირებული პოტენციალი მგრძნობიარეა შესასვლელზე მოდებული პოტენციალების დონეთა უმნიშვნელო ცვლილებების მიმართ. ეს ნაკლი განსაკუთრებით მწვავედ მაშინ იჩენს თავს, როდესაც ერთ-ერთ შესასვლელზე გვაქვს ლოგიკური ***0***. ამ შემთხვევაში შესასვლელზე იდეალური ნულოვანი პოტენციალისაგან ნებისმიერი გადახრა ზუსტად ასეთივე გადახრას წარმოშობს მის გამოსასვლელზე. გარდა ამისა, გამოსასვლელზე ნულოვანი ძაბვისაგან დამატებით გადახრას განაპირობებს პირდაპირ წანაცვლებულ დიოდზე არსებული ***არანულოვანი ძაბვის ვარდნაც***.

დიოდური ვენტილების მიმდევრობით ჩართვისას ერთი ვენტილიდან მეორე ვენტილში შესასვლელი სიგნალის გავლისას რხევებისადმი ზემოთ აღწერილი მგრძნობიარობით გამოწვეული გადახრები იკრიბება, რამაც შეიძლება სერიოზული დარღვევები გამოიწვიოს.



ნახ. დ3.2. (ა) დიოდური ***და*** ვენტილი; (ბ) ლოგიკური ელემენტის სამი ნაწილი.

2. ვენტილის მუშაობა დამოკიდებულია მის გამოსასვლელზე ჩართული მიმღების წინაღობაზე. ***მაღალი დონის გამოსასვლელი დენის დროს*** დენი ვენტილის გამოსასვლელიდან მიმღებისაკენ ვენტილის რეზისტორის გავლით მიედინება; რეზისტორზე ძაბვის ვარდნა გამოსასვლელზე პოტენციალს ამცირებს. ეს შემცირება მნიშვნელოვანია, რადგან შესასვლელზე დაბალი პოტენციალების დროს შესასვლელი წრედების გადატვირთვის თავიდან ასაცილებლად აღნიშნული რეზისტორის წინაღობა მნიშვნელოვანი უნდა იყოს. ***მცირე***

დონის გამოსასვლელი დენის დროს დენი მიმღებიდან ვენტილისაკენ არის მიმართული. ამ დენმა დაბალი პოტენციალის შესასვლელებში უნდა გაიაროს. ამან შეიძლება წარმოქმნას სიტუაცია, რომლის დროსაც ერთ შესასვლელს მოუხდება ბევრ სხვა ვენტილიდან მოსული დენის გატარება, რაც გარკვეულ სირთულესთან არის დაკავშირებული.

ზოგადად, დიოდური სქემებისადმი დამახასიათებელ ნაკლს განაპირობებს ის, რომ დიოდი ***პასიური ხელსაწყოა***. ტრნზისტორები პირიქით ***აქტიური ხელსაწყოებია*** იმ თვალსაზრისით, რომ მათში დიდ სიგნალებს სუსტი სიგნალები მართავს.

დიოდური ვენტილების ზემოთ მოყვანილი ნაკლოვანებები შეგვიძლია ***სპეციალური ტრანზიტორული სქემებით*** აღმოვფხვრათ. კერძოდ, ***პირველს ნაკლს*** თავიდან ავიცილებთ ტრანზისტორული სქემით, რომელიც ვენტილის გამოსასვლელზე ფორმირებულ პოტენციალის დონეს დაუახლოვებს იდეალურ მნიშვნელობას; ***მეორე ნაკლს*** აღმოვფხვრით მიმღებისათვის განკუთვნილი დენისაგან ვენტილის შესასვლელების განმამხოლოებელი დამატებითი წრედის მეშვეობით.

ამ ორ დამატებით სქემას ვენტოლის ***აღმდგენი*** და ***ბუფერული ნაწილები*** უწოდეს. საწყისმა დიოდურმა სქემამ კი მიიღო ვენტილის ***კომბინაციური ნაწილის*** სახელი, რადგან მისი საშუალებით შესასვლელ სიგნალთა სხვადასხვა კომბინაციები წარმოიშობა. დიოდური ვენტილის ამგვარად მოდერნიზებული სქემის ბლოკური გამოსახულება ***ღ3.2,ბ*** ნახაზზეა მოყვანილი. მისი სტრუქტურა შეიცავს როგორც დიოდებისაგან, ისე ტრანზისტორებისაგან აგებულ ნაწილებს. ვენტილების აგების ასეთმა ლოგიკამ ***დიოდურ-ტრანზისტორული ლოგიკის*** (***დტლ***-ის) სახელწოდება მიიღო.

ვენტილის სქემის ***ღ3.2,ბ*** ნახაზზე მოყვანილი პირობითი დაყოფა თითოეული ნაწილის მუშაობის განცალკევებულად განხილვის საშუალებას გვაძლევს. ამა თუ იმ კონკრეტული მიზნის მისაღწევად შეგვიძლია მარტო ბუფერული ან აღმდგენი ნაწილის მოდერნიზება მოვახდინოთ, ხოლო სხვა კომპონენტები უცვლელად დავტოვოთ. არსებობს ისეთი ვენტილებიც, რომელთა ასეთ სამ ნაწილად დაყოფა შეუძლებელია. ასეთია, მაგალითად, ***უშუალო კავშირებიანი სქემების ოჯახში*** შემავალი ვენტილები. მათში სამივე ფუნქციას ასრულებს განუყოფლად დაკავშირებული ტრანზისტორებისაგან შედგენილი სქემა.

ვენტილები გარკვეულ ოჯახებადაა დაყოფილი. თითოეულ ოჯახში გაერთიანებულია საერთო ტექნოლოგიით აგებული, მაგრამ სხვადასხვა ლოგიკური ფუნქციის შემსრულებელი ვენტილების ნაკრები. ამ ლოგიკურ ფუნქციებიდან ერთ-ერთი მათგანი წამყვანია და მასზე მახვილდება ყურადღება. ასეთ წამყვან ფუნქციად ყოველთვის აღებულია ისეთი ფუნქცია, რომელიც

დამოუკიდებლად წარმოქმნის ლოგიკური ფუნქციების ფუნქციურად სრულ სისტემას. ***პოსტ-იაბლონსკის*** თეორემის თანახმად ასეთებია ლოგიკუ-რი ***და-არა*** და ***ან-არა*** ფუნქციები. პირველ მათგანს შეფერის ფუნქცია, ხოლო მეორე მათგანს – პირსის ისარს უწოდებენ. ლოგიკურ ოჯახი შეიძ-ლება დავყოთ ***ბიპოლარული ტრანზისტორებისაგან*** აგებული ვენტილებისა და ***ლშნ ტრანზისტორებისაგან*** აგებული ვენტილების ***კლასებად.*** ვენტილების განხილვისას ოპერირებას ვახდენთ დაბალი და მაღალი დონის პოტენციალზე. ამ დროს საჭიროა შევთანხმდეთ მათ აღნიშვნებზე. დაბალ პოტენციალს თუ აღვნიშნავთ ლოგიკური ***0***-ით, ხოლო მაღალ პოტენციალს – ლოგიკური ***1***-ით, მაშვინ გვექნება ***დადებითი კონვენცია***, ხოლო პირიქით აღნიშვნის დროს – ***უარყოფითი კონვენცია***.

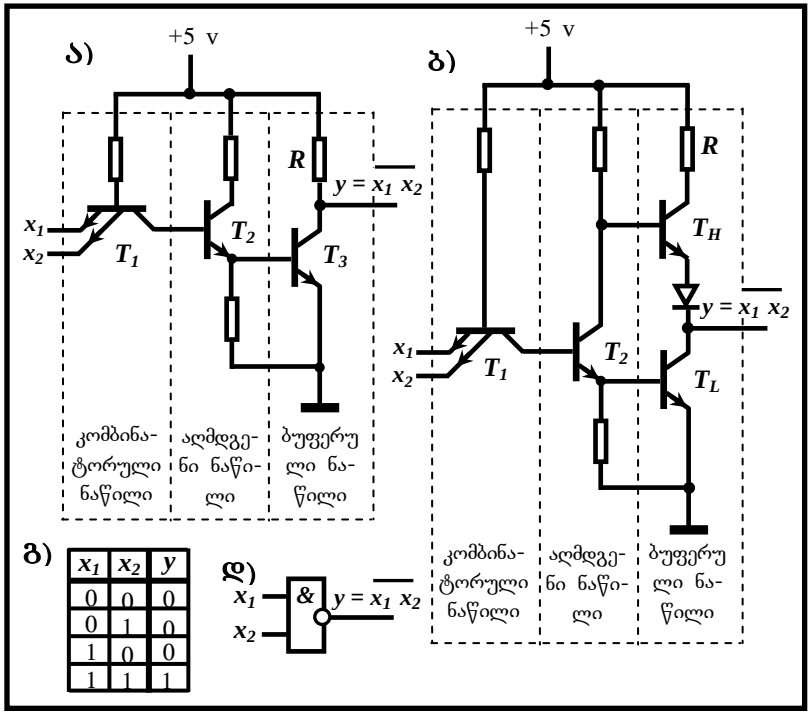
ვენტილების დამახასიათებელი ძირითადი პარამეტრებია: ***1) გამოსასვლე-ლის მიბჯდვით განშტოების კოეფიციენტი***, რომელიც გვიჩვენებს ვენტილის გამოსასვლელთან მისაერთებელი ვენტილების მაქსიმალურ რაოდენობას; იგი განსაზღვრება გამოსასვლელში გამავალი დენის ძალის მნიშვნელობით, რო-მელიც აუცილებელია გამოსასვლელზე მნიშვნელოვანი გადახრის გარეშე ლოგიკური ***0***-ისა და ***1***-ის შესაბამისი პოტენციალების შესანარჩუნებლად; ***2) შესასვლელის მიბჯდვით გაერთიანების კოეფიციენტი***, რომელიც გვიჩვენ-ებს მოცემული ტიპის ვენტილის შესასვლელების რაოდენობას; ***3) დაყოვ-ნების დრო*** ანუ ***სიგნალის გავრცელების დაყოვნება***, რომელიც გვიჩვენებს ვენტილზე სიგნალების კომბინაციის მიწოდებიდან რა დროის გასვლის შემდეგ ფორმირდება შესაბამისი გამოსასვლელი სიგნალი. ***4) დაბრკოლება-მდგრადობა***, რომელიც ახასიათებს ვენტილის უნარს გაუმკლავდეს ორი ლოგიკური მნიშვნელობების შესაბამისი სტანდარტული სიდიდეებიდან სი-გნალის გადახრას; ***5) სიმძლავრის განბნევა***, რომელიც წარმოადგენს მუ-შაობის დროს ვენტილის მიერ მოხმარებულ სიმძლავრეს;

Დ3.3. ტრანზისტორულ-ტრანზისტორული ლოგიკა და მისი მოდიფიკაციები (მონტაჟური და სამსტაბილური ლოგიკები)

ტრანზისტორულ-ტრანზისტორული ლოგიკის (***ტტლ***-ის) სახელწოდების ოჯახში შემავალი ლოგიკური ელემენტები მხოლოდ ბიპო-ლარული ტრანზისტორებითაა აგებული. მასში შემავალი ერთ-ერთი ვენტი-ლი ***Დ3.3,ა*** ნახაზზეა გამოსახული.

ვენტილის კომბინატორული ნაწილი შეიცავს მრავალემიტერულ T_1 ტრანზისტორს. ნახაზზე ნაჩვენებია ორი ემიტერი, თუმცა მათი რაოდენობა შეიძლება უფრო მეტიც იყოს. თითოეული შესასვლელი სიგნალი საკუთარ

ემიტერზეა მიწოდებული. ემიტერები ბაზასთან წარმოქმნის ***pn***–გადასასვლელს. პირდაპირ წანაცვლებული თუნდაც ერთი ემიტერის არსებობისას T_1 ტრანზისტორი გამტარ მდგომარეობაში იქნება, ე. ი. კოლექტორის გამომყვანში შეიძლება გავიდეს დენი. რეზისტორით ბაზაზე დადებითი პოტენციალის მიწოდების გამო ბაზა-ემიტერის გადასასვლელი პირდაპირ წანაცვლდება ყოველთვის, როდესაც შესაბამისი ემიტერის პოტენციალი «ნულოვან», ანუ ჩამიწებული წერტილის პოტენციალთან ახლოს იქნება. ეს იმას ნიშნავს, რომ T_1 ტრანზისტორი გამტარ მდგომარეობაში იქნება მაშინ, როდესაც თუნდაც ერთ-ერთ x_i შესასვლელზე მიწოდებული იქნება ლოგიკური ***0*** (დაბალი პოტენციალი). ამ შემთხვევაში დაბალი პოტენციალის მქონე ემიტერში გავა ძირითადად ბაზური დენით განპირებული შესამჩნევი დენი. ვენტილის ყველა შესასვლელზე ლოგიკური ***1***-ის (მაღალი პოტენციალის) მიწოდებისას T_1 ტრანზისტორი დაიკეტება და ყველა მის ემიტერში ძალიან სუსტი დენი გაივლის.



ნახ. დ3.3. ᲓᲐ-ᲐᲠᲐ- ფუნქციის მარეალიზებელი ***ტტლ*** ვენტილის ნაირსახეობები;

ვენტილის აღმდგენი ნაწილი შედგება T_2 ტრანზისტორისა და ორი რეზისტორისაგან. T_2-ის ბაზა T_1-ის კოლექტორთანაა შეერთეული, ე.ი. T_1-ის კოლექტორის დენი T_2-ის ბაზურ დენს წარმოადგენს. როდესაც T_1 გაღებულია, მაშინ დენი მისი კოლექტორიდან დაბალი პოტენციალის მქონე ემიტერის ან ემიტერებისაკენ არის მიმართული. ეს მიმართულება იმ დენის მიმართულების საწინააღმდეგოა, რომელიც უნდა ჰქონდეს T_2-ის ბაზის დენს ამ ტრანზისტორის ბაზა-ემიტერის გადასასვლელის პირდაპირ წასანაცვლებლად. ე.ი. ***ყოველთვის, როდესაც T_1 ღიაა, დაკეტილია T_2.***

T_1-ის დაკეტილ მდგომარეობაში ყოფნისას მის ბაზაზე მოდებული ***+5*** ვოლტის პოტენციალის გამო ბაზა-კოლექტორის გადასასვლელი პირდაპირაა წანაცვლებული. ამიტომ პირდაპირ წანაცვლებული ***pn***-დიოდის მსგავსად იგი გამტარ მდგომარეობაში უნდა იყოს. ეს თავის მხრივ პირდაპირ წაანაცვლებს T_2-ის ბაზა-ემიტერსაც, ასე რომ T_2 აღმოჩნდება გაჯერებილ (გამტარ) მდგომარეობაში. T_2 ტრანზისტორის ამ ურთიერთსაწინააღმდეგო (წაკვეთისა და გაჯერების) მდგომარეობიდან თითოეული შესასვლელზე პირობების მთელი დიაპაზონისათვის შეიმჩნევა. ამ თვალსაზრისით T_2 ტრანზისტორის წრედი რეალურად ასრულებს სიგნალის აღმდგენის ფუნქციას.

ვენტილის ბუფერული ნაწილი T_3 ტრანზისტორისა ***R*** რეზისტორისაგან შედგება. T_3-ის ბაზის პოტენციალს მართავს აღმდგენი ნაწილის T_2 ტრანზისტორი. ***T_2-ის გამტარ მდგომარეობაში ყოფნისას*** მისი ემიტერი იღებს ***0***-სა და ***+5*** ვოლტს შორის არსებულ რაღაც დადებით მნიშვნელობას (კონკრეტული სიდიდე აღმდგენი ნაწილის ორ რეზისტორზეა დამოკიდებული). ეს თავის მხრივ პირდაპირ წაანაცვლებს ბუფერული ნაწილის T_3 ტრანზისტორს და იგი გაიღება. ასეთ სიტუაციაში ვენტილის ***y*** გამოსასვლელზე გაჩნდება «მიწის» პოტენციალთან ახლომდებარე პოტენციალი.

მეორე შემხვევაში, როდესაც ***T_2 არ ატარებს,*** ნულის ტოლი აღმოჩნდება მის ემიტერზე მოდებული ძაბვა, რის გამოც T_3-ის ბაზა-ემიტერის გადასასვლელი არ იქნება პირდაპირ წანაცვლებული და T_3 დაიკეტება. ამ სიტუაციაში ბუფერის კოლექტორული წრედის რეზისტორის წყალობით ვენტილის გამოსასვლელზე დამყარდება კვების ***+5*** ვოლტთან ახლომდებარე პოტენციალი. გამოსასვლელზე პოტენციალის რეალური სიდიდე დამოკიდებულია ***R*** რეზისტორზე ძაბვის ვარდნით და მას განსაზღვრავს ამ რეზისტორში გამავალი დატვირთვის დენი. განხილული ***ტტლ-*** ვენტილის ფუნქციონირების განმსაზღვრელი ჭეშმარიტობის ცხრილი ***ლ3.3,გ*** ნახაზზეა მოყვანილი - იგი შესაბამება ***და-არა*** ფუნქციის ჭეშმარიტობის ცხრილს. ე. ი. განხილული ვენტილი წარმოადგენს ***და-არა*** ლოგიკურ ელემენტს. ამ ელემენტის პირობითი გამოსახულება ***ლ3.3,დ*** ნახაზზეა ნაჩვენები. მის შესა-

სვლელზე მარტო ერთი ემიტერის არსებობისას მივიღებთ ***არა*** ელემენტს (***ინვერტორს***).

ბუფერს უნდა შეეძლოს მასთან მიერთებული სხვა ვენტილების სწორად მუშაობისათვის საჭირო სიდიდის დენის გატარება. ***ტტლ***- ვენტილის შესასვლელზე მნიშვნელოვანი დენი მასზე დაბალი პოტენციალის არსებობის დროს შეინიშნება. ამ შემთხვევაში დენი მართული ვენტილის შესასვლელიდან მმართველი ვენტილის გამოსასვლელისაკენ არის მიმართული. ჩვენს მიერ განხილული სქემა დააკმაყოფილებს მმართველი ვენტილისადმი წაყენებულ მოთხოვნებს, თუ ბუფერულ T_3 ტრანზისტორს შეეძლება გაატაროს მისი მართული ყველა შესასვლელიდან მოსული ჯამური და საკუთარი კოლექტორული რეზისტორიდან მოსული დენი.

განხილულ ვენტილში დაკეტილი T_3-ის დროს ვენტილის გამოსასვლელზე დაბალი პოტენციალიდან მაღალ პოტენციალზე გადასართველად R რეზისტორი გამოიყენება, რაც ზრდის სიგნალის გავრცელების დაყოვნებას. ვენტილის გამოსასვლელ ხაზსა და «მიწას» შორის ყოველთვის წარმოიშვება საგრძნობი პარაზიტული ტევადობა. ***დაბალი პოტენციალი მაღალი პოტენციალით ვერ შეიცვლება ამ ტევადობის განმუხტვამდე, რაც აყოვნებს ამ პროცესს. მაღალი პოტენციალი დაბალი პოტენციალით გაცილებით სწრაფად იცვლება.***

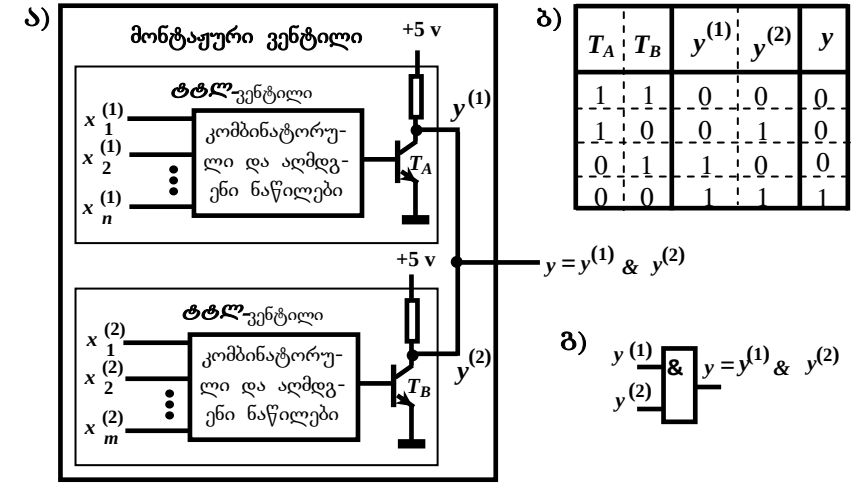
ტტლ-ვენტილების გადართვის სიჩქარის გასაზრდელად ბუფერულ ნაწილში არსებული პასიური R რეზისტორის ნაცვლად ხშირად ტრანზისტორი გამოიყენება. ეს ***ტრანზისტორი გაიღება მაშინ, როდესაც ვენტილის გამოსასვლელზე პოტენციალი მაღალი ხდება*** და წარმოქმნის პარაზიტული ტევადობის სწრაფად დამმუხტველ დაბალომიან წრედს. ***ლ3.3,ბ*** ნახაზზე წარმოდგენილი ამგვარად მოდიფიცირებული ***ტტლ***- ვენტილის სქემის ბუფერულ ნაწილს ეწოდება ***აქტიური დატვირთვიანი გამოსასვლელი კასკადი.*** ამ ბუფერში გვაქვს T_H და T_L ტრანზისტორები. ამათგან T_H ტრანზისტორი განკუთვნილია ვენტილის გამოსასვლელზე მაღალი დონის პოტენციალის, ხოლო T_L ტრანზისტორი, პირიქით – დაბალი დონის პოტენციალის ფორმირებისათვის.

T_2-ის დაკეტილი მდგომარეობის დროს T_L-ის ბაზაზე მოდებულია «მიწის» დაბალი პოტენციალი, ხოლო T_H–ის ბაზაზე მაღალი პოტენციალი; ამიტომ T_H გამტარ მდგომარეობაშია, ხოლო T_L დაკეტილია.

T_2-ის გამტარი მდგომარეობის დროს ორივე T_L და T_H ტრანსისტორების ბაზაზე ისეთი საშუალო სიდიდის პოტენციალია მოდებული, რომ იგი საკმარისია T_L ტრანზისტორის, მაგრამ არაა საკმარისი T_H ტრანზისტორის გასაღებად. T_H-ს მიმდევრობით მიერთებული დიოდი განკუთვნილია T_H-ის პოტენციალის დაახლოებით ***0,7*** ვოლტამდე გასაზრდელად. ეს უზრუნველ-

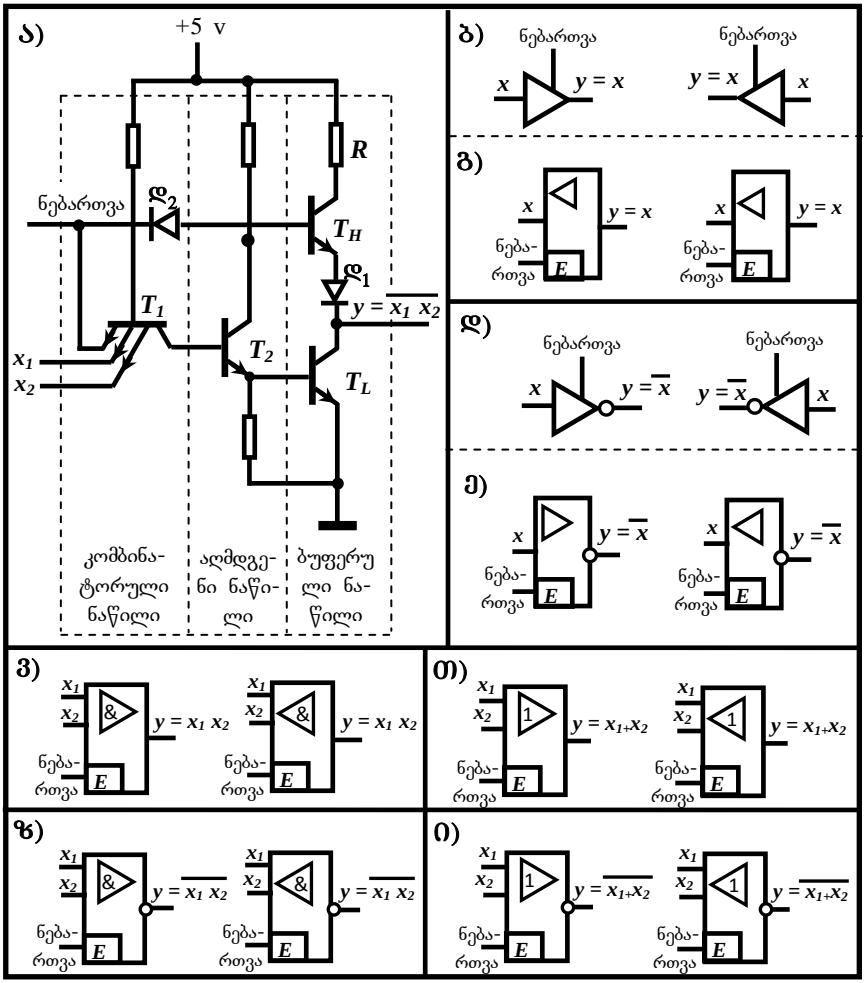
ყოფს T_2 და T_L ტრანზისტორების ღია მდგომარეობაში ყოფნისას გარანტირებულად იყოს დაკეტილი T_H ტრანზისტორი. T_H და T_L ტრანზისტორებს შორის მცირე წინაღობა ჩართულია ერთი მდგომარეობიდან მეორეში გადასვლის დროს გამავალი დენის შეზღუდვისათვის, როდესაც ორივე ტრანზისტორის ნაწილობრივ გამტარ მდგომარეობაში იმყოფება.

გამოსასვლელზე სადატვირთო რეზისტორის მქონე რამდენიმე **ტტლ**-ვენტილის გამოსასვლელების შეერთებით ახალი ლოგიკური ელემენტი მიიღება. ახალი ლოგიკური ელემენტის მიღების ასეთ სამონტაჟო ხერხს ***მონტაჟური ლოგიკა*** ეწოდება. **დ3.4,ა** ნახაზზე გამოსახულია **და-არა** ფუნქციის მარეალიზებელი ორი **ტტლ** ვენტილი, რომელთა გამოსასვლელი ხაზებია შეერთებული. ერთ-ერთი ან ორივე ბუფერული (T_A,T_B) ტრანზისტორის ღია მდგომარეობაში ყოფნის დროს საერთო ხაზზე მყარდება «მიწის» ტოლი დაბალი პოტენციალი, ანუ ლოგიკური **0**. საწინააღმდეგო შემთხვევაში რეზისტორების წყალობით გამოსასვლელზე მყარდება მაღალი პოტენციალი, ანუ ლოგიკური **დ3.4,ბ** ნახაზზე მოყვანილია ამ გაერთიანებით მიღებული სქემის ჭეშმარიტობის ცხრილი, რომლის თანახმადაც მიღებული სქემა ახდენს $y = y^{(1)} \,\&\, y^{(2)}$ კონიუნქციის რეალიზებას, ე. ი. წარმოადგენს **და** ელემენტს.



ნახ. დ4.4. ორი **ტტლ** ვენტილით მონტაჟური **და** ვენტილის აგება.

მონტაჟური ლოგიკის გამოყენებით მიღებულ ვენტილს ექნება მაღალი სწრაფმოქმედება მის ბუფერულ ნაწილში აქტიური დატვირთვის მქონე **ტტლ** ვენტილების (ნახ. **დ3.4,ბ**) გამოყენებისას (ამ უკანასკნელთა მაღალი სწრაფმოქმედების გამო). ამ იდეის რეალიზების საშუალებას არ გვაძლ-



ნახ. დ4.5. (ა)სამსტაბულური ***ტტლ***-სქემები (ბ)-(ი) სამსტაბილური სქემების სახესხვაობების პირობითი აღნიშვნები

ევს შემდეგი ორი გარემოება: ***1)*** ერთ-ერთი ***ტტლ***- ვენტილი მის ბუფერული ნაწილის ზედა ტრანზისტორის ღია მდგომარეობის დროს ***შეიცდებოდა აემაღლებინა პოტენციალი*** საერთო გამოსასვლელზე; იმავე დროს მეორე ***ტტლ***- ვენტილი მის ბუფერულ ნაწილში ქვედა ტრანზისტორის ღია მდგომარეობისას ამ საერთო გამოსასვლელის პოტენციალის პირიქით, ***დასა-***

დაბლებად არ დაიშურებდა ძალისხმევას. ამის შედეგად ღია ტრანზისტორებში გაივლიდა ძალიან დიდი დენი, რომლებსაც შეიძლება ისინი დაეზიანებინა. ***2)*** ხიფათს თავს ან ვერც ამ დაზიანების განუხორციელებისას ავიცილებდით, რადგან საერთო გამოსასვლელზე პოტენციალის დონე არ იქნებოდა არც იმდენად დაბალი, რომ იგი ლოგიკურ ***0***-ად ჩაგვეთვალა და არც იმდენდენად მაღალი, რომ ლოგიკურ ***1***-ად მიგვეჩნია.

ზემოთ აღნიშნული მიუხედავად შეიძლება სამონტაჟო ლოგიკის ღირსებას შევუთავსოთ გამოსასვლელ კასკადში სადატვირთო ტრანზისტორების მქონე ვენტილების სწრაფმოქმედება. კერძოდ, ***ტტლ-*** ვენტილის ქცევა შეიძლება შევცვალოთ, თუ მდგომარეობისათვის, რომლის დროსაც მისი პოტენციალი არც საკმაოდ მაღალია და არც საკმაოდ დაბალი, ***მესამე მდგომარეობის ცნებას შემოვიტანთ***! ამ დროს მიღებულ ვენტილს ***სამი მდგომარეობის მქონე სქემას,*** ანუ ***სამსტაბილურ სქემას (ვენტილს)*** უწოდებენ.

დ3.5.ა ნახაზზე მოყვანილია ***და-არა*** ფუნქციის მარეალიზებელი სამსტაბილური ***ტტლ-*** ვენტილი, რომელშიც T_H და T_L ტრანზისტორების სამართავად შემოტანილია ***ნებართვის ბაზად*** წოდებული დამატებითი ***მმართველი ბაზი.*** ეს ბაზი $Д_2$ დიოდით უერთდება აღმდგენი ნაწილის T_2 ტრანზისტორის ემიტერსა და ბუფერული სქემის ზედა T_H ტრანზისტორის ბაზას.

ნებართვის ბაზზე დაბალი პოტენციალის (ლოგიკური ***0***-ის) მიწოდებისას ორივე გამოსასვლელი T_H და T_L ტრანზისტორი დაკეტილია. ზედა T_H ტრანსისტორს კეტავს $Д_2$ დიოდის პირდაპირი წანაცვლების (გამტარ მდგომარეობაში ყოფნის) გამო მის ბაზაზე დამყარებული დაბალი პოტენციალი. ქვედა T_L ტრანზისტორის ჩაკეტვას კი იწვევს შესასვლელ T_1 ტრანზისტორის ემიტერზე ლოგიკური ***0***-ის არსებობა.

ნებართვის ბაზზე მაღალი პოტენციალის (ლოგიკური ***1***-ის) მიწოდებისას $Д_2$ დიოდი დაიკეტება და ამიტომ ზედა T_H ტრანზისტორის ქცევა დაემთხვევა ***ტტლ-*** ვენტილში მის ნორმალურ ქცევას. ამიტომ შესასვლელი T_1 ტრანზისტორის რომელიმე ემიტერზე მიწოდებულ ლოგიკურ ***1***-ს ეს ემიტერი თითქოს თამაშიდან გაყავს. ამგვარად, ნებართვის ბაზზე მაღალი პოტენციალის მიწოდებისას ვენტილის მდგომარეობას მთლიანად განსაზღვრავს სხვა შესასვლელები და იგი ***და-არა*** ფუნქციის მარეალიზებელი ჩვეულებრივი ***ტტლ-*** ვენტილივით იმუშავებს.

არსებობს სამი შესასვლელის მქონე შემდეგი სახის სამსტაბილური ვენტილი:

1. სამსტაბილური მაფორმირებელი, რომელსაც გამოსასვლელზე გადააქვს მის შესასვლელზე მიწოდებული პოტენციალი. მისი პირობითი გამოსახულება ***ტ3.5,ბ*** ნახაზზე მოყვანილია უცხოურ ლიტერატურაში, ხოლო ***ტ3.5,გ*** ნახაზზე – სამამულო სამეცნიერო-ტექნიკურ ლიტერატურაში მიღე-

ბული მისი პირობითი გამოსახულება; ***2.*** სამსტაბილური ***არა-*** ვენტილი (ნახ. ***დ3.5,დ,ე***); ***3.*** სამსტაბილური ***და-*** ვენტილი (ნახ.***დ4.5,ვ***); ***4.*** სამსტაბილური ***და-არა-*** ვენტილი (ნახ. ***დ4.5,ზ); 3.*** სამსტაბილური ***ან*** ვენტილი (ნახ. ***დ3.5,თ***); ***4.*** სამსტაბილური ***ან-არა-*** ვენტილი (ნახ. ***დ3.5,ი***).

სამსტაბილური ორი ან უფრო მეტი რაოდენობის ვენტილების გამოსასვლელების შეერთებისას ***ნებართვის სიგნალი*** დროის ნებისმიერ მომენტში მხოლოდ ერთ ვენტილს უნდა ეწოდებოდეს, ე. ი. «ნებართვა» მხოლოდ ერთ ვენტილს უნდა ეძლეოდეს. ამის შედეგად მიიღება ***გარკვეულად შეზღუდული სახის სამონტაჟო ლოგიკა.*** მისი ქცევის აღწერა ჩვეულებრივი სამონტაჟო ლოგიკის ქცევის აღწერაზე რამდენადმე რთულია.

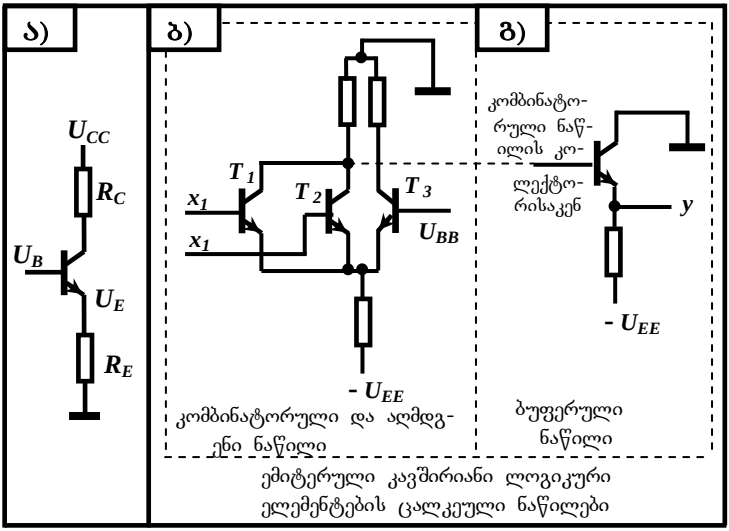
ტტლ ფართოდ გავრცელდა კომპიუტერებში, ელექტრონულ-სამუსიკო ინსტრუმენტებში, საკონტროლო-საზომ აპარატურასა და ავტომატიკაში. ***ტტლ***-ის ფართოდ გავრცელების გამო ელექტრონული მოწყობილობებს ხშირად ***ტტლ***-თან შეთავსებად შესასვლელ და გამოსასვლელ წრედებით აღჭურავენ.

ელექტრონული სისტემების კონსტრუქტორებისათვის ***ტტლ*** პოპულარული ***Texas Instruments*** ფირმის მიერ ***1965*** წელს ***7400*** სერიის შემოთავაზების შემდეგ გახდა. მიკროსქემების ეს სერია სამრეწველო სტანდარტად გადაიქცა. ვინაიდან ***Texas Instruments*** ფირმის ***7400*** სერია ყველაზე მეტად გავრცელდა, ამიტომ მას ფუნქციურად და პარამეტრულად იმეორებს სხვა ***(AMD, Harris, Intel, Intersil, Motorola, National)*** ფირმების მიერ გამოშვებული პროდუქციები. ***ტტლ***-ის მნიშვნელობას განაპირობებს ის, რომ ***ტტლ***-მიკროსქემები მეტად მოსახერხებელია მასობრივი წარმოებისათვის და მისმა პარამეტრებმა მნიშვნელოვნად გააასწრო ადრე წარმოებული (***რლტ-, დრტ-***) მიკროსქემების პარამეტრებს.

დ3.4. ემიტერული კავშირიანი ლოგიკური ელემენტების აგების საფუძვლები

ბიპოლარული ტრანზისტორებით აგებული ლოგიკური ვენტილების სქემების სწრაფმოქმედების გასაზრდელად საჭიროა გამოვრიცხოთ მათში არსებული ტრანზისტორების გაჯერების მდგომარეობაში გადასვლა. ***გაჯერება*** ეწოდება ტრანზისტორის მდგომარეობას, რომლის დროსაც მისი ორივე გადასასვლელი პირდაპირ არის წანაცვლებული. გაჯერების მდგომარეობაში მყოფ ტრანზისტორში მუხტის არაძირითადი მზიდავები ორივე გადასასვლელის გავლით მიგრირებს და ბაზის ზონაში გროვდება ჭარბი რაოდენობის არაძირითადი მზიდავები, რომელიც უნდა გაიფანტოს ტრანზისტორის დაკეტილ მდგომარეობაში გადასვლამდე. ლოგიკური ელემენტების ***<u>ემიტერული კავშირებიანი ლოგიკის</u>*** (***ეკლ*** ლოგიკის) სახელწოდების ოჯახში გაჯერების რეჟიმს კოლექტორულ დენს საჭირო ზღვრებში შენარჩუნების გზით იცილებენ თავიდან.

ЭЖЛ-ვენტილის ტრანზისტორში კოლექტორული დენის დონეს მართავს ემიტერთან მიერთებული რეზისტორი (ნახ. ***Დ3.6,ა***). ამ რეზისტორის არარსებობის (ე. ი. ემიტერის პირდაპირ ჩამიწების) დროს გაღებული ტრანზისტორის გაჯერების თავიდან აცილება გაგვიჭირდებოდა, რადგან მისი ამ მდგომარეობაში ყოფნისას ბაზა-ემიტერის გადასასვლელზე მოდებულ ძაბვის სიდიდე მცირედ (***0,2-0,3*** ვოლტით) განსხვავდება გაჯერების მდგომარეობის დროს იმავე გადასასვლელზე მოდებული ძაბვის სიდიდისაგან. ამის გამო ტრანზისტორის გასაღებად ბაზაზე ძაბვის მიწოდებისას სიგნალის დონის ოდნავმა დარღვევამ, ან ტრანზისტორის პარამეტრების მცირეოდენმა გადახრამ შეიძლება ტრანზისტორი გაღებულ მდგომარეობაში გადაიყვანოს.



ნახ.Დ3.6. *კოლექტორული დენის შემზღუდველი ემიტერიანი რეზისტორის მქონე ტრანზისტორის სქემა (ა);* ***ЭЖЛ-*** *ვენტილის კომბინატორული (ბ) და ბუფერული (გ) ნაწილების სქემა*

დავუშვათ, რომ ტრანზისტორის ბაზაზე მიწასთან შეფარდებით მიწოდებულია დადებითი U_B პოტენციალი (იხ. ნახ. ***Დ3.6,ა***). ემიტერული დენით R_E რეზისტორზე გამოწვეული ძაბვის ვარდნა ემიტერზე წარმოშობს დადებით U_E პოტენციალს. ამის გამო ბაზა-ემიტერზე მოდებული $U_{B\text{-}E}$ ძაბვა ბაზაზე მოდებულ U_B ძაბვაზე U_E სიდიდით იქნება ნაკლები. უფრო მეტიც, U_B ძაბვის ზრდის კვალობაზე გაიზრდება ემიტერში გამავალი დენი, რაც გაზრდის ემიტერზე მოდებულ U_E ძაბვას. ემიტერზე მოდებული ძაბვის ზრდას გარკვეულწილად აკომპენსირებს ბაზაზე ძაბვის გაზრდა. ამგვარად ბაზა-ემიტერის

გადასასვლელზე პოტენციალთა სხვაობა გაცილებით უფრო ნელა გაიზრდება, ვიდრე მიწის მიმართ ბაზის ძაბვა. სხვა სიტყვებით რომ ვთქვათ, ემიტერის წრედში რეზისტორის არსებობის გამო წარმოიშვება ***უარყოფითი უკუკავშირი***, რომელიც ***ამცირებს ბაზაზე მოდებული ძაბვის ცვლილებისადმი ტრანზისტორის მგრძნობიარობას***. ამ უარყოფითი კავშირის გამო ***მცირდება ტრანზისტორის პარამეტრების ცვლილებისადმი კოლექტორში გამავალი დენის მგრძნობიარობაც.*** ამ ორი გარემოების ერთობლიობა გვაძლევს ტრანზისტორის გაჯერების გამორიცხვის საშუალებას

როგორც დავრწმუნდით, ტრანზისტორის ემიტერულ წრედში რეზისტორის ჩართვით თავიდან ვიცილებთ გაჯერებას, ოღონდ ემიტერის დენის სასურველი ცვლილების უზრუნველყოფისათვის საჭიროა ბაზაზე მოდებული ძაბვა საკმაო ფართო დიაპაზონში იცვლებოდეს. სამწუხაროდ, არაა სასურველი, ორი ლოგიკური მნიშვნელობის შესაბამისი შესასვლელი ძაბვის დონეები ერთმანეთისაგან მნიშვნელოვნად განსხვავდებოდეს. ამიტომ ***ეკლ-*** ვენტილში შესასვლელი სიგნალები კომბინირდება დიფერენციალური (ან დენით გადამრთველი) სქემის მეშვეობით (ნახ. ***დ3.6,ბ***). ასეთი სქემის დროს ორი ლოგიკური მნიშვნელობის შესაბამის შესასვლელ ძაბვებს შორის განსხვავება მცირეა. სქემა შეიცავს ბმული ემიტერებიან რამდენიმე ტრანზისტორს, რომლებსაც აქვს საერთო ემიტერული რეზისტორი. სქემის მარცხენა ნაწილში განთავსებული ტრანზისტორების ბაზები x_1 და x_2 შესასვლელებთანაა დაკავშირებული. ამ შესასვლელი ტრანზისტორების კოლექტორები საერთო კოლექტორულ რეზისტორთანაა მიერთებული. მარჯვნივ განთავსებული ტრანზისტორის ბაზას მიეწოდება სტაბილური საყრდენი U_{BB} ძაბვა, ხოლო კოლექტორულ წრედში ჩართულია განცალკევებული რეზისტორი. კოლექტორული რეზისტორების ზედა ბოლოები ჩამიწებულია და კვების უარყოფითი U_{BB} ძაბვა ემიტერული რეზისტორის ქვედა ბოლოზეა მოდებული.

T_3 ტრანზისტორის ბაზაზე მოდებული U_{BB} ძაბვის სიდიდე შესასვლელი ძაბვებისათვის ზღურბლურ სიდიდეს წარმოადგენს. T_3 ტრანზისტორის წყალობით ყველა ემიტერის შეერთების წერტილზე მოდებულ ძაბვის სიდიდე ($U_{BB} - U_{B\text{-}E}$)-ზე არანაკლებ სიდიდემდე დაიყვანება. ამიტომ, თუ ერთ-ერთ მარცხენა ტრანზისტორზე მოდებული შესასვლელი ძაბვა U_{BB}-ზე ნაკლები იქნება, მაშინ ეს ტრანზისტორი დაიკეტება, ვინაიდან მისი ბაზა-ემიტერის გადასასვლელი საკმაოდ პირდაპირ ვერ წანაცვლდება.

მეორე მხრივ, ერთ-ერთ მარცხენა ტრანზისტორის ბაზაზე U_{BB}-ზე მეტი ძაბვის მოდებისას მის ემიტერზე მოდებული ძაბვა ($U_{BB} - U_{B\text{-}E}$)-ზე მეტი აღმოჩნდება. ამ სიტუაციაში T_3 ტრანზისტორი დაიკეტება, ვინაიდან მისი ბაზა-ემიტერის გადასასვლელი საკმაოდ პირდაპირ ვერ წანაცვლდება. ამგვარად, ვენტილის შესასვლელებზე მოდებული ძაბვის მნიშვნელობაზე დამოკიდებულებ-

ით დენი მხოლოდ მარცხენა ან მარჯვენა ნაწილის ტრანზისტორში (ან ტრანზისტორებში) გაივლის.

სინამდვილეში შესასვლელებზე მოდებულ ძაბვათა დონეები U_{BB}-ს მნიშვნელობასთან ძალიან ახლოა; ასე, რომ ემიტერზე მოდებული პოტენციალის სიდი-დე შედარებით მუდმივია. ამიტომ შედარებით მუდმივია ემიტერულ რეზისტორ-ში გამავალი დენიც, რომელიც ღია ტრანზისტორში გამავალ დენს წარმოადგ-ენს. საყრდენი U_{BB} ძაბვის სიდიდე ისე შეირჩევა, რომ I დენმა არ გამოიწვიოს ტრანზისტორის გაჯერება.

სქემის იმ ნაწილის კოლექტორულ რეზისტორზე, რომელშიც I დენი გადის, მოხდება ძაბვის ვარდნა. ამის შედეგად გამტარი ნაწილის კოლექტორზე მოდებული ძაბვა მიიღებს უარყოფით $-U_{CL}$ მნიშვნელობას და არაგამტარ ნაწილში კოლექტორზე მოდებული ძაბვა ნულის ტოლი გახდება. ერთ-ერთ შესასვლელზე მოდებული ძაბვის მნიშვნელობა თუ U_{BB}-ს მნიშვნელობაზე მეტი (ე.ი. უფრო ნაკლებ უარყოფითი) იქნება, მაშინ მარცხენა ნაწილის ტრანზისტორი გაღებული იქნება, რის შემწეობითაც მარცხენა ნაწილის კოლექტორზე მოდებული ძაბვა მიიღებს $-U_{CL}$ მნიშვნელობას, ხოლო მარჯვენა ნაწილში კოლექტორზე მოდებული ძაბვა ნულის ტოლი გახდება.

მეორე მხრივ, თუ ორივე შესასვლელზე მოდებული ძაბვა თუ U_{BB}-ზე ნაკლები იქნება, მაშინ დაიკეტება მარცხენა ნაწილში არსებული ორივე T_1 და T_2 ტრანზისტორი და, მაშასადამე, მარცხენა ნაწილის კოლექტორზე მოდებული ძაბვა ნულის ტოლი, ხოლო მარჯვენა ნაწილის კოლექტორზე მოდებული ძაბვა $-U_{CL}$-ს ტოლი გახდება. ამგვარად, სქემა ახდენს ***შესასვლელი სიგნალების კომბინირებას.*** იგი ***ალმდგენის ფუნქციასაც ასრულებს***, რადგან კოლექტორზე მოდებული ძაბვები შესასვლელი სიგნალის რყევებისადმი ნაკლებად მგრძნობიარეა, ამ რყევების სიმცირის გამო.

ზემოთ განხილულ სქემის გარდა ***ეჰლ-*** ვენტილს აქვს ბუფერული ნაწილიც (ნახ. ***დ3.6,გ***). ბუფერული ტრანზისტორის ბაზა სქემის კომბინატორული ნაწილის ნებისმიერ კოლექტორულ გამომყვანთან არის მიერთებული. მთლიანად ვენტილის y გამოსასვლელს ფაქტურად ბუფერული ტრანზისტორის ემიტერი წარმოადგენს (იხ.ნახ. ***დ3.6,გ***). ვინაიდან ბუფერული ტრანზისტორის ემიტერულ ნაწილზე ჩართულია რეზისტორი, რომელზეც კვების უარყოფითი - U_{EE} ძაბვაა მოდებული, ამიტომ ამ ტრანზისტორის ბაზა-ემიტერის გადასასვლელი პირდაპირ მიმართულებითაა წანაცვლებული და იგი ყოველთვის გაღებულია. მიუხედავად ამისა, ემიტერული რეზისტორის წყალობით ბუფერული ტრანზისტორის გაჯერება არ ხდება. მის ემიტერზე მოდებული ძაბვის სიდიდე უდრის ბაზაზე მოდებულ ძაბვის სიდიდეს მინუს $U_{B\text{-}E}$ (ე.ი. ემიტერზე მოდებული ძაბვა ბაზაზე მოდებულ ძაბვაზე უფრო უარყოფითია). ბაზის ძაბვის ცვლილებისას მასთან ერთად იცვლება ემიტერის ძაბვაც. ამ თვალსაზრისით ემი-

ტერზე მოდებული ძაბვა იმეორებს ბაზაზე მოდებულ ძაბვას. ამიტომ ასეთ სქემას ჩვეულებრივ ***ემიტერულ მიმეორებელს*** უწოდებენ.

ეკლ- ვენტილში ბუფერული ნაწილი ასრულებს ორ ფუნქციას. ***ჯერ ერთი,*** იგი გამორიცხავს დატვირთვის დენზე (რომელიც ვენტილის გამოსასვლელზე გვაქვს) კომბინაციური ნაწილის კოლექტორზე მოდებული ძაბვის დამოკიდებულებას; ***და მეორეც***, მისი წყალობით განისაზღვრება ვენტილის გამოსასვლელზე ლოგიკური ***0***-ისა და ლოგიკური ***1***-ის შესაბამისი ძაბვების ისეთი მნიშვნელობები, რომელთა დროსაც გამოირიცხება ამ გამოსასვლელთან დაკავკავშირებული სხვა ***ეკლ***- ვენტილის შესასვლელი ტრანზისტორის გაჯერება. კერძოდ, ლოგიკურ ***1***-ს შეესატყვისება $U_{B\text{-}E}$ ძაბვა, ხოლო ლოგიკურ ***0***-ს – $(U_{CL} - U_{B\text{-}E})$ ძაბვა.

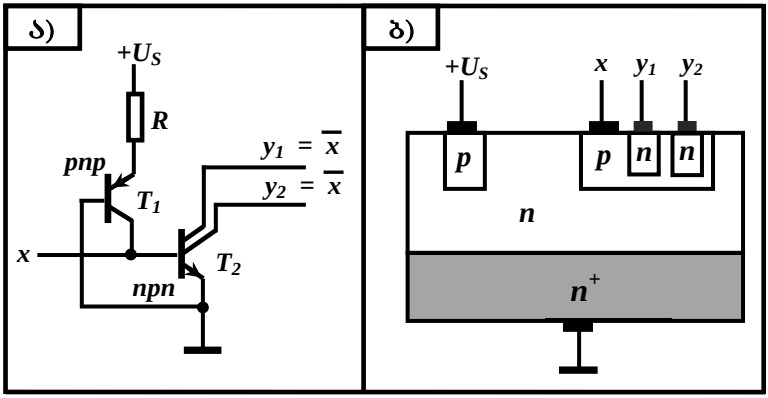
§3.5. ინტეგრალური ინჟექციური ლოგიკის თავისებურებები

მიკროპროცესორებისა და მასთან დაკავშირებული კომპონენტების რეალიზაციისას ერთ-ერთი უმნიშვნელოვანესი პარამეტრია ვენტილების რაოდენობა, რომლებიც შეიძლება ჩალაგდეს ერთ ინტეგრალურ სქემაში. ერთიკრისტალური მიკროპროცესორის ასაგებად საკმაოდ მაღალი უნდა იყოს კრისტალზე ვენტილების განთავსების სიმჭიდროვე. ასე მჭიდროდ შეიძლება განთავსდეს ***<u>ინტეგრალური ინჟექციური ლოგიკის (I^2L) ოჯახის</u>*** შემადგენლობაში შემავალი ლოგიკური სქემების ასაგებად გამოყენებული ვენტილები..

ჩვეულებრივ I^2L ვენტილები აიგება ***ნ3.7,ა*** ნახაზზე მოყვანილი ორტრანზისტორული სქემის საფუძველზე, რომელიც ინვერტორს წარმოადგენს. მასში ***R*** რეზისტორიანი ***pnp*** ტიპის T_1 ტრანზისტორი ასრულებს ***npn*** ტიპის T_2 ტრანზისტორის ბაზაზე დენის მიმწოდებელი კვების წყაროს როლს. T_1 ტრანზისტორის ბაზა ჩამიწებულია და მისი ბაზა-ემიტერის გადასასვლელი უკუწანაცვლებულია. ამიტომ T_1-ის ემიტერის პოტენციალი შედარებით მუდმივი და მცირე (დაახლოებით ***0,7*** ვოლტის ტოლი) რჩება ასევე თითქმის მუდმივია ***R*** რეზისტორზე არსებული ძაბვის ვარდნა და იგი დაახლოებით $(+U_S - 0,7)$ ვოლტის ტოლია. ამგვარად, ***R*** რეზისტორში გადის T_1-ის ემიტერული დენის ტოლი $(U_S\text{-}0,7)/R$ სიდიდის დენი. ეს დენი პრაქტიკულად T_1-ის კოლექტორული დენის ტოლია (მათი სხვაობა ძალიან მცირეა და უდრის ბაზის დენს).

შესასვლელი წრედი თუ არ მოიხმარს დენს (***x*** შესასვლელი გაწყვეტილია), მაშინ T_1-ის კოლექტორული დენი T_2-ის ბაზას მიეწოდება. ამბობენ, რომ ეს დენი «***ინჟექცირდება***» T_2-ის ბაზაში (აქედან წარმოდგება ოჯახის სახელწოდება.) ამ რეჟიმში T_2-ის ბაზა-ემიტერი საკმაოდაა პირდაპირ წანაცვლებული და გაღებულია. მეორე მხრივ, შესასვლელი თუ მიწასთანაა მიერთებული, მაშინ T_1-ში გამავალი დენი გაივლის შესასვლელ წრედში, რის შედეგადაც T_2 დაიკეტება. ამგვარად, როდესაც ვენტილის შესასვლელი «განრთულია», მაშინ მი-

სი გამოსასვლელი «ჩამიწებული», ხოლო თუ "ჩამიწებულია", მაშინ მისი გამოსასლელი «განრთული» აღმოჩნდება. ლოგიკურ სიდიდეებად «განრთვისა» და «ჩამიწების» ინტერპრეტირებისას მივიღებთ ინვერტორს.



ნახ. ლ47. I^2L*-ინვერტორი (ა) და მისი ფიზიკური სტრუქტურა (ბ)*

I^2L ვენტილის სქემაზე უფრო ადვილი მისი დამზადებააა (ნახ.***ლ3.7,ბ***). რამდენადაც ერთმანეთთანა შეერთებული T_1-ის კოლექტორისა და T_2-ის ბაზას დასამზადებლად ***p***-ტიპის ნახევარგამტარი გამოიყენება და ამიტომ ისინი კრისტალზე ერთიანი გარემის სახით რეალიზდება. ანალოგურ გაერთიანებას წარმოქმნის T_1-ის ბაზა და T_2-ის ემიტერი. ამიტომ უწოდებენ I^2L ლოგიკას შეთავსებულ ტრანზისტორულ ლოგიკასაც.

რამდენიმე შესასვლელიანი ვენტილები შეიძლება განხილულ საბჭისო ვენ-ტილების მონტაჟის გზით, ე. ი. ***მონტაჟური ლოგიკის გამოყენებით ავაგოთ***. კერძოდ, ***ან-არა*** ლოგიკური ფუნქციის მარეალიზებელი ორშესასვლელიანი I^2L ვენტილს მივიღებთ თუ ორი ინვერტორის გამოსასვლელებს ერთმანეთთან შევაერთებთ.

I^2L ლოგიკის განსაკუთრებული ღირსებაა ის, რომ მისი განთავსებისათვის კრისტალის ძალიან მცირე ფართობია საკმარისი, რის გამოც იგი ინტეგრირებისათვის მეტად მოსახერხებელია.

T_1 ტრანზისტორი ასრულებს მუდმივი I_0 დენის წყაროს როლს. თუ $x = 1$ მაშინ ეს დენი მთლიანად შედის T_2 ტრანზისტორის ბაზაში, რის შედეგადაც იგი ღიაა. ამ შემთხვევაში $y_1 = 0$ და $y_2 = 0$. თუ $x = 0$, მაშინ იგი გადის T_1 ტრანზისტორის ბაზაში და $y_1 = 1$, $y_2 = 1$.

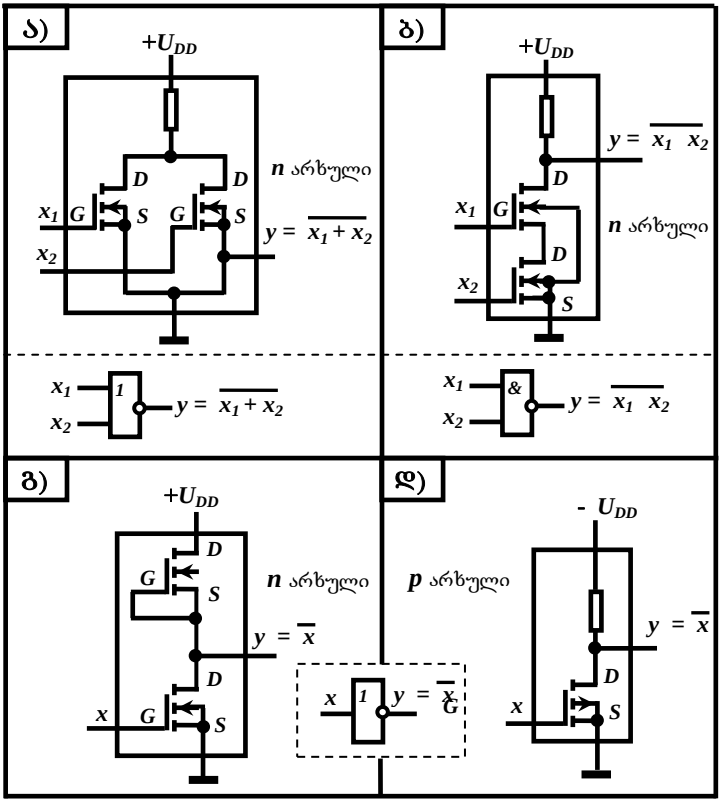
Დ.3.6. ველის ტრანზისტორებით აგებული ლოგიკური ელემენტები

ველის ტრანზისტორებით აგებული ელემენტების რამდენიმე სერია არსებობს. ორი ფართოდ ცნობილი ოჯახი აგებულია ***n-ლშნ-*** და ***p-ლშნ*** ტექნოლოგიით; ისინი შესაბამისად იყენებს ***n***–არხულ და ***p***–არხულ ტრანზისტორებს. მესამე ოჯახის ვენტილებში გამოიყენება როგორც ***n***–არხული, ასევე ***p***–არხული ტრანზისტორები, რომლებიც ერთმანეთს «ავსებს», ერთმანეთის მიმართ «დამატებას» (ინგ. ***Complementary*** – «დამატებითი», «ურთიერთშემავსებადი») წარმოადგენს, ამიტომ მას კომპლემენტარული ლოგიკის ***ლშნ*** ოჯახი ეწოდა და შემოკლებულად აღნიშნავენ, როგორც ***კლშნ***.

n-არხული და ***p***-არხული ***ლშნ-*** ტრანზისტორებით აგებული ლოგიკური ელემენტები სქემები მიკუთვნება ***უშუალო კავშირებიანი სქემების ოჯახს***. ამ ოჯახში შემავალ სქემებში გადამრთველად მომუშავე ტრანზისტორები მიმდევრობით ან პარალელურადაა შეერთებული. მაგალითისათვის, გავაანალიზოთ პარალელურად შეერთებული ნორმალურად დაკეტილი (გამდიდრების რეჟიმში მომუშავე) ***n***-არხული ორი ***ლშნ-*** ტრანზისტორისაგან მიღებული სქემა (ნახ. ***Დ3.8,ა***). x_1 და x_2 შესასვლელიდან ერთ ერთზე (ან ორივეზე) დადებითი პოტენციალის (ლოგიკური ***1***-ის) მოდებისას მასთან დაკავშირებული ტრანზისტორი (ტრანზისტორები) გაიღება, გამოსასვლელ ხაზსა და მიწას შორის წარმოიქმნება დაბალი წინაღობის წრედი და გამოსასვლელზე დამყარდება დაბალი პოტენციალი. მეორე მხრივ, თუ ორივე ტრანზისტორზე მოდებულია მიწის პოტენციალი, ისინი დაიკეტება. ამ სიტუაციაში ***R*** რეზისტორის ხაზით გამოსასვლელზე დამყარდება დაახლოებით U_{DD}–ს ტოლი პოტენციალი, რადგან მაღალია დაკეტილი ტრანზისტორების იმპენდასი. დადებითი კონვენციის გამოყენების შემთხვევაში ***Დ3.8,ა*** ნახაზზე მოყვანილი ვენტილი მოახდენს ***ან-არა-*** ფუნქციის რეალიზებას, რადგან მის გამოსასვლელზე დაბალი პოტენციალი გაჩნდება მაშინ და მხოლოდ მაშინ, როდესაც ერთ-ერთ შესასვლელზე მაინც იქნება მოდებული მაღალი პოტენციალი.

ნორმალურად დაკეტილი ***n***-არხულ ***ლშნ-*** ტრანზისტორების მიმდევრობით შეერთებისას (ნახ. ***Დ3.8,ბ***) მივიღებთ ***და-არა-*** ვენტილს. ამ შემთხვევაში მიმდევრობითად შეერთებული ტრანზისტორები შესასვლელსა და მიწას შორის შედარებით დაბალი წინაღობის მქონე წრედს მხოლოდ მაშინ წარმოქმნის, როდესაც ორივე შესასვლელზე გვექნება მაღალი პოტენციალი. ერთ-ერთ შესასვლელზე მაინც დაბალი პოტენციალის არსებობისას გამოსასვლელზე გვექნება დაახლოებით U_{DD}–ს ტოლი პოტენციალი. რადგან მიმ-

დევრობითი შეერთების დროს წინაღობები იკრიბება, ამიტომ შეზღუდულია მიმდევრობით შეერთებული ტრანზისტორების რაოდენობა.



ნახ. 3.8. *ველის ტრანზისტორებით აგებული ლოგიკური: ა)* ***ან-არა****-ელემენტი; ბ)* ***და-არა*** *ელემენტი; გ)* ***n-ლჟნ*** *ინვერტორი; დ)* ***p-ლჟნ*** *ინვერტორი*

რეზისტორები ინტეგრალურ სქემაში სხვა კომპონენტებთან შედრებით დიდ მოცულობას იკავებს, რის გამოც მოუხერხებელია მათი გამოყენება. ამიტომ ***ლჟნ-*** ტრანზისტორებით აგებულ ვენტილში სადატვირთო რეზისტორადაც ***ლჟნ-*** ტრანზისტორს იყენებენ, რომლებიც არაწრფივი რეზისტორებივით მოქმედებს. ძალიან ხშირად ამ მიზნისათვის იყენებენ გაღარიბების რეჟიმში მომუშავე ***ლჟნ-*** ტრანზისტორებს, რომლებშიც საკეტი სათავესთანაა შეერთებული, ხოლო ფუძეშრესთან შეერთებული კონტაქტი გამოუყენებლადაა დატოვებული, როგორც ეს ***ნ3.8,გ*** ნახაზზე მოყვანილ, ***ინვერტორის***

სქემაზეა ნაჩვენები. ამ ინვერტორის ***x*** შესასვლელზე დაბალი პოტენციალის (ლოგიკური ***0***-ის) მიწოდებისას დაკეტილია მასთან დაკავშირებული ქვედა ტრანზისტორი. ***y*** გამოსასვლელისა და სადატვირთო ზედა ტრანზისტორის შემცველ ხაზში გადის მცირე დენი. ზედა სადატვირთო ტრანზისტორი იღება, რადგან იგი გაღარიბების რეჟიმში მუშაობს; ძაბვის ვარდნა ძირითადად ქვედა ტრანზისტორზე ხდება და გამოსასვლელზე მყარდება მაღალი პოტენციალი (ლოგიკური ***1***).

x შესასვლელზე პირიქით, მაღალი პოტენციალის (ლოგიკური ***1***-ის) არსებობისას ღიაა მასთან დაკავშირებული ქვედა ტრანზისტორი; ზედა სადატვირთო ტრანზისტორის საკეტსა და სათავეზე, აგრეთვე ვენტილის ***y*** გამოსასვლელზე მყარდება დაახლოებით მიწის პოტენციალის ტოლი დაბალი პოტენციალი (ლოგიკური ***0***). ჩასადინართან დაკავშირებულ არხთან შეფარდებით საკეტის პოტენციალი უარყოფითი ხდება; მკვეთრად იზრდება სადატვირთო ტრანზისტორის წინაღობა ე. ი იგი ტრანზისტორი არაწრფივი რეზისტორივით იქცევა და მისი გამოყენება სჯობია არაწრფივი რეზისტორის გამოყენებას.

n-არხულ ***ლშნ***-სქემებში ძაბვა დადებითი უნდა იყოს. კვების ძაბვის სიდიდეს თუ ***+5*** ვოლტის ტოლად ავიღებთ და ***ლშნ***-ტრანზისტორების საკეტებზე უზრუნველვყოფთ ზღურბლური ძაბვის საჭირო U_T მნიშვნელობას, მაშინ მიღებული ***n***-არხულ ***ლშნ***-სქემები ***ტტლ***-სქემებისადმი შეთავსებადები იქნება.

p-***ლშნ***-ვენტილების სქემა ძირითადად ***n***-***ლშნ***-სქემების ანალოგურია, ოღონდ ***p***-არხული ***ლშნ***-ტრანზისტორების მუშაობისას მათ საკეტსა და ჩასადინარზე სათავესთან შეფარდებით უარყოფითი (დაბალი) ძაბვა უნდა იყოს მოდებული, როგორც ეს ***ლ3.8,დ*** ნახაზზე მოყვანილ ***ინვერტორის*** სქემაზეა ნაჩვენები.

ლშნ-ტრანზისტორებს საკეტში გამავალი დენის მიმართ აქვს მაღალი წინაღობა, რის შედეგადაც სტატიკურ მდგომარეობაში ***ლშნ*** ვენტილები მათი ***მმართველი სქემებისაგან დენს პრაქტიკულად არ მოიხმარს***. ამის გამო მათ აქვთ გამოსასვლელის მიმართ განშტოების დიდი კოეფიციენტი. ამასთანავე შეგვიძლია ინტეგრალურ სქემაში ისინი მჭიდროდ ჩავალაგოთ. მეორე მხრივ საკმაოდ დიდია ***ლშნ***-ტრანზისტორის საკეტს, სათავეს, ჩასადინარსა და ფუძეშრეს შორის წარმოქმნილი ტევადობების მნიშვნელობები. ამიტომ ბიპოლარული ტრანზისტორებით აგებულ ვენტილებთან შედარებით დაბალია ***ლშნ***-ვენტილების სწრაფმოქმედება (ამ უკანასკნელებში მდგომარეობის შეცვლამდე აღნიშნული ტევადობები საჭიროა განიმუხტოს). გარდა ამისა, ტევადობის გადამმუხტველი დენი მმართველი ვენტილიდან უნდა მო-

დიოდეს, რაც ხშირი გადართვების დროს მნიშვნელოვნად ზრდის გაბნეულ სიმძლავრეს.

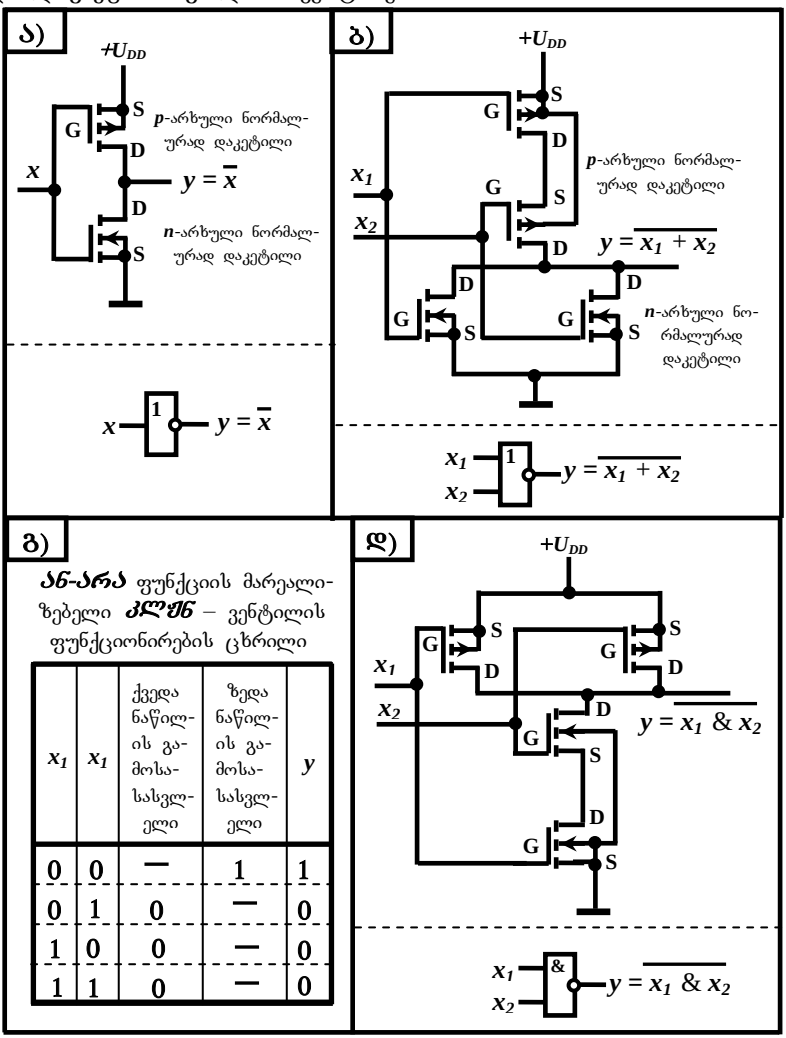
დ.4.7. კლშნ-სტრუქტურებით აგებული ლოგიკური ელემენტები

კომპლემენტარული ლითონ-ჟანგეულ-ნახევარგამტარის ანუ ***კლშნ*** სტრუქტურაში ზემოთ განხილული ***n-ლშნ-*** და ***p-ლშნ-*** სტრუქტურებისაგან განსხვავებით არსებობს ველის როგორც ***n***- ასევე ***p***-არხიანი ტრანზისტორები. ***კლშნ-*** ტექნოლოგია ***1963*** წელს, ხოლო ამ ტექნოლოგიით აგებული ***პირველი მიკროსქემები - 1968*** წელს დამუშავდა.

კლშნ ვენტილი შედგება ორი ნაწილისაგან, რომელთაგანაც ერთ-ერთი ***ადაბლებს*** გამოსასვლელზე, ხოლო გარკვეული პირობების დროს, შესასვლელზეც მოდებულ პოტენციალს; მეორე ნაწილი კი პირიქით - აღნიშნულ პოტენციალებს ***ამაღლებს.*** ორივე ნაწილი აგებულია უშუალო კავშირიანი სქემის სახით. ***ნ.3,ა*** ნახაზის ზედა ნაწილზე მოყვანილია ***კმოპ-***ელემენტებით აგებული ***არა-*** ვენტილის სქემა, ხოლო ქვედა ნაწილზე ამ ვენტილის პირობითი აღნიშვნა. მისი «***დამდაბლებელი ნაწილი***» შედგება ***n***-არხული ნორმალურად დაკეტილი ***ლშნ-*** ტრანზისტორისაგან, რომლის ***ჩასადინარი y*** გამოსასვლელთან, ხოლო ***სათავე*** – მიწასთანაა შეერთებული. «***ამამაღლებელი ნაწილი***» შედგება შედგება ***p***-არხული ნორმალურად დაკეტილი ***ლშნ-*** ტრანზისტორისაგან, რომლის ***ჩასადინარი y*** გამოსასვლელთან, ხოლო ***სათავე*** - კვების დადებით U_{DD} ძაბვასთანაა მიერთებული. ინვერტორის ***x*** შესასვლელი ორივე ტრანზისტორის ***საკეტს*** უერთდება.

n-არხული ქვედა ***ლშნ-*** ტრანზისტორი მაშინ გაიღება, როდესაც ***x*** შესასვლელზე მოდებული ძაბვის სიდიდე ვენტილის ზღურბლურ U_T სიდიდეს გადააჭარბებს. ამის შედეგად ***y*** გამოსასვლელზე დამყარდება დაბალი პოტენციალი. საწინააღმდეგო შემთხვევაში ამ ტრანზისტორს ექნება მაღალი წინაღობა და გავლენას ვერ მოახდენს გამოსასვლელის დონეზე. ***p***-არხული ზედა ტრანზისტორი მუშაობის სათავის პოტენციალზე შედარებით მაღალი ჩასადინარის პოტენციალის დროს. იგი მაშინ გაიღება, როდესაც ***საკეტის*** პოტენციალი სათავის U_{DD}-ს ტოლ პოტენციალზე საკმაოდ ნაკლები იქნება. ამგვარად, ყოველთვის, როდესაც შესასვლელის პოტენციალი U_{DD}-ს ჩამოუვარდება ვენტილის ზღურბლური U_T ძაბვაზე მეტი სიდიდით, ეს ტრანზისტორი გაიღება და გამოსასვლელზე დამყარდება მაღალი პოტენციალი. საწინააღმდეგო შემთხვევაში ***p***-არხულ ტრანსისტორს ექნება მაღალი წინაღობა და გავლენას ვერ მოახდენს გამოსასვლელი სიგნალის დონეზე. ნათელია, რომ დადებითი კონვენციის დროს თუ ***x=0***, მაშინ ***y=1*** და პი-

რიქით, როდეს-აც ***x=1***, მაშინ ***y=0***, ე.ი. სქემა ახდენს შესასვლელზე მოდებული ლოგიკური სიგნალის ინვერტირებას.



ნახ.დ 3.9. კმოპ-*ვენტილები: ა)* ***კმოპ-*** *ინვერტორი; ბ) ორშესასვლელიანი* ***ან-არა****- ვენტილი; გ)* ***ან-არა*** *ვენტილის ფუნქციონირების ცხრილი; დ) ორშესასვლელიანი* ***და-არა-*** *ვენტილი;*

კმოპ-ელემენტებისაგან აგებული ორშესასვლელიანი ***ან-არა***- ვენტილის სქემა ***ლ3.9,ბ*** ნახაზის ზედა, ხოლო მისი პირობითი გამოსახულება – ქვედა ნაწილზეა მოყვანილი. ვენტილის «***დამადაბლებელი ნაწილი***» სქემის ქვე-და ნაწილში მოყვანილი პარალელურად შეერთებული და ჩამიწებული სათავეების მქონე ორი ***n***-არხული ტრანზისტორისაგან შედგება. თითოეული ამ ტრანზისტორის საკეტი ერთ-ერთ (x_1 ან x_2) შესასვლელს უკავშირდება. რომელიმე შესასვლელზე თუ მოდებულია ზღურბლურ U_T ძაბვაზე დიდი ძაბვა (ლოგიკური ***1***), მაშინ მასთან დაკავშირებული ტრანზისტორი გაიღება და გამოსასვლელზე დამყარდება ლოგიკური ***0*** (დაბალი პოტენციალი). საწინააღმდეგო შემთხვევაში ორივე ტრანზისტორი დიდი წინაღობიან რეზისტორებივით იმოქმედებს და გავლენას ვერ მოახდენს გამოსასვლელი ***y*** სიგნალის დონეზე.

ვენტილის «ამამაღლებელი» ნაწილი შედგება მიმდევრობით შეერთებული ორი ***p***-არხიანი ***ლშნ***-ტრანზისტორისაგან. ამ ტრანზისტორების თითოეული საკეტი შეერთებულია ერთ-ერთ გამოსასვლელთან. ორივე შესასვლელზე მოდებული პოტენციალი როდესაც U_{DD}-ზე ნაკლებია ვენტილის გაღებისათვის საჭირო U_T-ზე მეტი სიდიდით, მაშინ ორივე ტრანზისტორი გაიღება და მაღლა ასწევს გამოსასვლელის პოტენციალს. საწინააღმდეგო შემთხვევაში ისინი მაღალი წინაღობიანი რეზისტორების მსგავსად იმოქმედებს.

დადებითი ლოგიკის ჩარჩოში ნებისმიერ შესასვლელზე ლოგიკური ***1***-ის დროს სქემის ქვედა ნაწილი გამოსასვლელზე დონეს ამცირებს ლოგიკურ ***0***-მდე, ხოლო როდესაც ორივე შესასვლელს მიეწოდება ლოგიკური ***0***, მაშინ სქემის ზედა ნაწილი გამოსასვლელზე უზრუნველყოფს ლოგიკურ ***1***-ს. ამგვარად ვენტილი ახდენს ლოგიკურ ***ან-არა*** ოპერაციას. მისი ფუნქციონირების ცხრილიდან (ნახ.***ლ3.9.9,ბ***) ჩანს, რომ სქემის ქვედა ნაწილი ახდენს ლოგიკური ***0***-ის, ხოლო ზედა ნაწილი – ლოგიკური ***1***-ის ფორმირებას. ***და-არა*** ოპერაციის მარეალიზებელი ***კმოპ***- ვენტილები შეიძლება ავაგოთ «დამადაბლებელი» ნაწილის ***n***-არხული ტრანზისტორების მიმდვრობთი და «ამამაღლებელი» ნაწილის ***p***-არხული ტრანზისტორების პარალელური შეერთების გზით (***ლ3.9,დ*** ნახაზი).

კლშნ- ლოგიკის სტრუქტურა წინა პარაგრაფში განხილულ სტრუქტურებზე უფრო რთულია და ამ ტექნოლოგიით დამზადებულ ლოგიკური ელემენტებს მიკროსქემაში მაღალი სიმჭიდროვით ვერ განვათავსებთ. სამაგიეროდ მას აქვს ერთი მეტად მნიშვნელოვანი ღირსება. სტრუქტურის შესასვლელი სიგნალების მუდმივად მიწოდების პერიოდში სტრუქტურა უმნიშვნელო სიმძლავრეს მოიხმარს. ეს ძალიან მნიშვნელოვანია ბატარეით მომუშავე მოწყობილობებისათვის.

დანართი 4.
ციფრული მოწყობილობების აბსტრაქტული სინთეზის ფორმალური ალგორითმი [6,7]
(დამუშავებულია სახელმძღვანელოს ავტორის მიერ 2005-2008 წლებში)

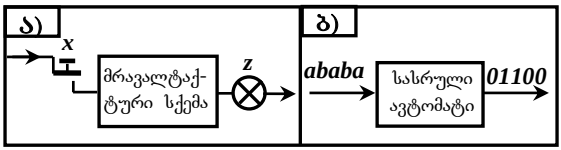
მმართველი მოწყობილობის აგების პირველ სტადიაზე გარკვეული ენის გამოყენებით შედგენილი უნდა იქნეს დავალება, რომელშიც ზუსტად და ლაკონიურად იქნება ფორმულირებული აღნიშნული მოწყობილობის მიერ შესასრულებელი ფუნქციები. ასეთ ენად ბუნებრივი სალაპარაკო ენის გამოყენება არაა სასურველი მისი ბუნებრივი სიჭარბისა და არაერთმნიშვნელიანობის გამო. ამიტომ დამუშავებული იქნა სხვადასხვა არაჭარბი ფორმალური ენები. ერთ-ერთ ასეთ ენას წარმოადგენს ***რეგულარულ გამოსახულებათა ენა***.

რეგულარული გამოსახულების მეშვეობით ზუსტად და ცალსახად ფორმულირდება მმართველი მოწყობილობის აგების დავალება, რომლის მიხედვითაც კონსტრუქტორი ახდენს აღნიშნული გამოსახულების ***პროგრამულ ან აპარატურულ რეალიზებას***. რეგულარული გამოსახულების სახით შედგენილი დავალების მიხედვით მმართველი მოწყობილობის რეალიზების პროცესში გარკვევისათვის, უპირველეს ყოვლისა, უნდა გვესმოდეს თავად რეგულარული გამოსახულების რაობა, რომელიც ავტომატების აბსტრაქტული თეორიაშია განმარტებული.

Დ4.1. ავტომატების აბსტრაქტული თეორიის ელემენტები

აუცილებელ საწყის ცნებებს გავეცნოთ კერძო მაგალითის განხილვის პროცესში.

დავუშვათ, რომ ასაგებია ერთი შესასვლელისა (ღილაკი ***x***) და ერთი გამოსასვლელის (ნათურა ***z***) მქონე დისკრეტული მოწყობილობა, რომელიც ***z*** ნათურას ჩართავს ***x*** ღილაკზე თითის ყოველ კენტი რაოდენობით და ამორთავს - თითის ყოველი ლუწი რაოდენობით დაჭერისას (ნახ. **Დ4.1,ა**).



ნახ. Დ4.1. *აბსტრაქტული ავტომატის განსაზღვრის ილუსტრირება*

ავტომატების აბსტრაქტული თეორია დისკრეტულ მოწყობილობას «***შავი ყუთის***» სახით განიხილავს, ე.ი. მას არ აინტერესებს მისი შინაგანი სტრუ-

ქტურა, ე. ი როგორაა აგებული რეალური მრავალტაქტური სქემა. ამ თეორიის მეთოდები დისკრეტული მოწყობილობის ქცევას განსაზღვრავს სიგნალების შესასვლელი და გამოსასვლელი მიმდევრობების სახით. ეს საშუალებას გვაძლევს, ვიპოვოთ დისკრეტული მოწყობილობის ფუნქციონირების უზოგადესი კანონზომიერებები. «შავი ყუთის» სახით განხილულ დისკრეტულ მოწყობილობას ***აბსტრაქტული ავტომატი*** ანუ უბრალოდ, ***ავტომატი*** ეწოდება.

ავტომატის შესასვლელი ცვლადების ნაკრებს ვუწოდოთ ***შესასვლელი ასო***. *დ4.1,ა* ნახაზზე ნაჩვენები შემთხვევისათვის არსებობს შესასვლელი ცვლადების ორი ნაკრები ***x=0*** და ***x=1***, რომლებიც შესაბამისად ***a*** და ***b*** ასოებით აღვნიშნოთ. შესასვლელი ასოების სიმრავლეს ეწოდება ***შესასვლელი ალფაბეტი*** და ***A*** სიმბოლოთი აღინიშნება, ე.ი. ***A=(a,b)***. ანალოგურად, გამომოსასვლელი ცვლადების ნაკრებს ეწოდება ***გამოსასვლელი ასოები,*** ხოლო მათ სიმრავლეს – ***გამოსასვლელი ალფაბეტი.*** ჩამქრალი ***z*** ნათურის შესაბამის გამოსასვლელ ასოს თუ აღვნიშნავთ ***0***-ით, ჩამქრალი ნათურის შესაბამის ასოს - ***1***-ით, ხოლო გამოსასვლელ ალფაბეტს ***Z***-ით, მაშინ შეგვიძლია დავწეროთ, რომ ***Z=(0,1)***.

ავტომატის თითოეულ შინაგან მდგომარეობას შევუსაბამოთ გარკვეული გამოსასვლელი ასო. განხილულ შემთხვევაში სულ ორი გამოსასვლელი ასო არსებობს, ავტომატის შინაგან მდგომარეობათა რაოდენობაც ორის ტოლი იქნება. ერთ-ერთს, რომლის დროსაც ნათურაში დენი არ გადის, შევუსაბამოთ გამოსასვლელი ასო ***0***, ხოლო მეორეს, რომლის დროსაც ნათურაში დენი გადის – გამოსასვლელი ასო ***1.*** მივიღებთ ავტომატს, რომლის გამოსასვლელი სიგნალი მხოლოდ შინაგან მდგომარეობის ფუნქციას წარმოადგენს. ასეთ ავტომატს ***მურის ავტომატი*** ეწოდება, ე.ი. ჩვენი ავტომატი ***მურის ტიპის სასრულ ავტომატს*** წარმოადგენს (მას სასრული რაოდენობის, კერძოდ ორი შინაგანი მდგომარეობა აქვს). არსებობს ე. წ. ***მილის ავტომატიც***, რომლის გამოსასვლელი სიგნალი ავტომატის შესასვლელი სიგნალისა და შინაგანი მდგომარეობის ფუნქციას წარმოადგენს. მურის ნებისმიერ ავტომატს ცალსახად შეესაბამება ეკვივალენტური მილის ავტომატი, ამიტომ საკმარისია მხოლოდ მურის ავტომატი განვიხილოთ: მიღებული შედეგები სამართლიანი იქნება მილის ავტომატისთვისაც.

სასრული შესასვლელი და გამოსასვლელი ალფაბეტების მქონე ავტომატს, რომელსაც აქვს სასრული რაოდენობის შინაგანი მდგომარეობები, ***სასრული ავტომატი*** ეწოდება. ცხადია, რომ ჩვენს მაგალითში განხილულია მურის ტიპის აბსტრაქტული სასრული ავტომატი.

ავტომატის შინაგან მდგომარეობებს, რომელებსაც ***z=1*** გამოსასვლელი შეესაბამება, ***გამოსასვლელი სიგნალით მონიშნული მდგომარეობა*** ანუ სიმა-

რტივისათვის უბრალოდ ***მონიშნული მდგომარეობა*** ვუწოდოთ. ჩვენ მიერ განხილულ ავტომატში მონიშნულია ყველა ის მდგომარეობა, რომელშიც ავტომატის ყოფნის დროს ანთია ***z*** (იხ.ნახ.***ᲓᲐ1,ა***) ნათურა, ე. ი. როდესაც ***z=1***. ავტომატი მუშაობის დაწყების წინ გარკვეულ ფიქსირებულ მდგომარეობაში უნდა იყოს. ამ მდგომარეობას ***საწყისი მდგომარეობა*** ეწოდება. ყოველი ამუშავების წინ ავტომატი გადაიყვანება საწყის მდგომარეობაში. ამას ***ავტომატის ინიციალიზაცია*** ეწოდება.

განვიხილოთ მათემატიკური აპარატი, რომელიც საშუალებას მოგვცემს შესასვლელ და გამოსასვლელ სიგნალთა მიმდევრობების გამოყენებით აბსტრაქტული სასრული ავტომატის მუშაობა აღვწეროთ. შესასვლელი თუ გამოსასვლელი ასოების ნებისმიერ მიმდევრობას ***სიტყვა*** ვუწოდოთ. შესასვლელი ასოებით წარმოიქმნება ავტომატის ***შესასვლელი სიტყვები***, ხოლო გამოსასვლელი ასოებით – ***გამოსასვლელი სიტყვები***. ივარაუდება, რომ აბსტრაქტული ავტომატი მუშაობს ***დისკრეტულ დროში***. მაშინ რეალური სქემის შესასვლელების ნებისმიერი ცვლილება აბსტრაქტულად (ფიზიკური არსში გარკვევის გარეშე) შეიძლება ავტომატის შესასვლელზე გარკვეული შესასვლელი სიტყვის მოსვლად ინტერპრეტირდეს.

ᲓᲐ1,ბ ნახაზზე ნაჩვენებია, რომ შესასვლელი ***ababa*** სიტყვის მოსვლისას ავტომატი გამოიმუშავებს გამოსასვლელ ***01100*** სიტყვას, რაც ჩაიწერება ასე: ***ababa…→ 01100 …***; ამგვარად, სასრული ავტომატი შეიძლება ***სიტყვების გარდამქმნელად*** განვიხილოთ. იგი ***A={a,b}*** ალფაბეტით შედგენილი სიტყვების სიმრავლეს ცალსახად ასახავს ***Z={0,1}*** ალფაბეტით შედგენილი სიტყვების სიმრავლეში. აქვე შევნიშნავთ, რომ სასრული ალფაბეტის საშუალებით წარმოქმნილი სიტყვების რაოდენობა უსასრულოა.

ჩავთვალოთ, რომ შესასვლელი ***ρ*** სიტყვა ავტომატში ***წარმოდგენილია***, თუ მისი ზემოქმედებით ავტომატი საწყისი მდგომარეობიდან გადადის ერთერთ მონიშნულ მდგომარეობაში. მაგალითად, ***ababa*** სიტყვა ჩვენს ავტომატში არ არის წარმოდგენილი, რადგან მას ავტომატი გადაჰყავს მოუნიშნავ მდგომარეობაში (გამოსასვლელ სიტყვაში უკანასკნელი ასოა ***z=1***). ავტომატში წარმოდგენილი შესასვლელი სიტყვის მაგალითია ***ababab*** სიტყვა, რადგან ***ababab→011001.*** ამ შემთხვევაში ღილაკზე თითი სამჯერ არის დაჭერილი და ნათურა ანთია.

A={a,b} ალფაბეტით აგებული სიტყვების სიმრავლეს ეწოდება ***ხდომილება*** და აღინიშნება ***E*** ასოთი. ხდომილება წარმოდგენილია ავტომატში, თუ ამ ავტომატში წარმოდგენილია ხდომილებაში შემავალი ყველა სიტყვა. მაგალითად, განვიხილოთ ხდომილება ***E_1={ab, aba, abab}***. ჩვენს ავტომატში წარმოდგენილია ამ ხდომილებაში შემავალი ***ab, abab*** სიტყვები და არაა წა-

რმოდგენილი სიტყვა ***aba***, ამიტომ ავტომატში E_1 ხდომილება წარმოდგენილი არ არის.

მტკიცდება, რომ სასრულ ავტომატში წარმოდგენილია ნებისმიერი რეგულარული ხდომილება. ***რეგულარული ხდომილება*** ეწოდება ისეთ ხდომოლებას, რომელიც შეიძლება ერთი ფორმულის სახით ჩავწეროთ მხოლოდ ისეთი სამი ოპერაციის გამოყენებით, როგორიცაა დიზიუნქცია, გამრავლება და იტერაცია. ამ ფორმულას ეწოდება რეგულარული გამოსახულება, ე. ი. ***რეგულარული გამოსახულება*** წარმოადგენს ერთი ფორმულის სახით ჩაწერილ ხდომილებას. ამ ფორმულაში პირველად სრულდება იტერაციის, შემდეგ - გამრავლებისა და ბოლოს იტერაციის ოპერაცია. განვსაზღვროთ ზემოთ აღნიშნული ოპერაციები: ***1)*** E_1 და E_2 ხდომილების $E_1 \vee E_2$ ***დიზიუნქცია*** ეწოდება ამ ხდომილებების სიტყვათა სიმრავლეების გაერთიანებით მიღებულ სიმრავლეს. თუ E_1=(***ab, aba, abab***), ხოლო E_2=(***ba, bab, aba***), მაშინ $E_1 \vee E_2$=(***ab, aba, abab***, ***ba, bab, aba***) ***2)*** E_1 და E_2 ხდომილების E_1E_2 ***გამრავლება*** ეწოდება ხდომილებას, რომელიც შედგება ***pq*** სახის ყველა სიტყვისაგან, სადაც $p \in E_1$ ხოლო $q \in E_2$. ***3)*** E ხდომილების $\{E\}$ იტერაციაეწოდება ხდომილებას: $\{E\} = e \vee E \vee EE \vee EEE \vee \ldots$, სადაც არის ცარიელი სიტყვა, რომელის შეესატყვისება ავტომატის შესასვლელზე შესასვლელი ასოს მოუსვლელობას, მაგ., $\{ab\} = e \vee ab \vee abab \vee ababab \vee \vee abababab \vee \ldots$

ალგებრას, რომლის მზიდი სიმრავლეა შესასვლელი ალფაბეტით შედგენილი ყველა სიტყვის სიმრავლე, ხოლო ოპერაციები - ზემოთ ჩამოთვლილი სამი ოპერაცია, ***ხდომილების ალგებრა*** ეწოდება. მაშასადამე, რეგულარული გამოსახულება წარმოადგენს ხდომილების ალგებრის ფორმულას.

§4.2. რეგულარული გამოსახულების ანალიზი

რეგულარული გამოსახულებაში შეყურსულია სასრული ავტომატების ყველა ***შესასვლელი სიტყვა***. სასრული ავტომატების ***აბსტრაქტული სინთეზის ალგორითმი*** [5] საშუალებას გვაძლევს, თითოეულ შესასვლელი სიტყვის საპასუხოდ ავტომატის მიერ ფორმირებული ***გამოსასვლელი სიტყვა*** რეგულარული გამოსახულებიდან განვსაზღვროთ. რეგულარული გამოსახულებიდან მიმდევრობითი ავტომატის შესასვლელი სიტყვებისა და მათი შესატყვისი გამოსასვლელი სიტყვების განსაზღვრის პრაქტიკულ მეთოდს ***რეგულარული გამოსახულების გაშლა*** ეწოდება.

რეგულარული გამოსახულების გაშლის ოპერაცია საშუალებას გვაძლევს, ავაგოთ ავტომატის ***გადასვლებისა და გამოსასვლელების ცხრილი***, აგრეთვე ***გადასვლების დიაგრამა და მატრიცა***. აღნიშნული ელემენტების ერ-

თობლიობა წარმოქმნის სპეციფიკურ ენას, რომელსაც მოწყობილობის ***დამმუშავებლის ენას*** ვუწოდებთ. ამ ენის გამოყენება აძლევს საშუალებას დამმუშავებელს ფორმალური გზით მოახდინოს მოწყობილობის სტრუქტურის სინთეზირება.

ზემოთ აღნიშნულის თანახმად, მართვის ციფრული მოწყობილობების კო-ნსტრუირების პროცესში დამმუშავებლის წინაშე ბუნებრივად წამოიჭრება შემკვეთის ენიდან მისთვის ჩვეულ ენაზე გადასვლის საჭიროება. დღეისათვის აღნიშნული გადასვლა არაა ფორმალიზებული, რაც ართულებს მოწყობილობის კონსტრუირების საერთო პროცესს. ჩვენს სწორედ ამ გადასვლის ფორმალური მეთოდი დავამუშავეთ. რადგან დამმუშავებლის ენის ელემენტების დახმარებით მმართველი ციფრული მოწყობილობის სტრუქტურის აგების ამოცანა უკვე ფორმალიზებულია, ჩვენი მეთოდის გამოყენებით მთლიანად ფორმალიზდება ***მიმდევრობითი ავტომატების სინთეზის კლასიკური მეთოდების დახმარებით მმართველი მოწყობილობების კონსტრუირების*** მთელი ციკლი.

რეგულარული გამოსახულებების განხილვისას ადვილად შევამჩნევთ მათში შემავალ ცალკეულ ნაწილებს, რომლებიც მოთავსებულია როგორც იტერაციულ (ფიგურულ), ისე ალგებრულ (მრგვალ) ფრჩხილებში. ფიგურულ ფრჩხილებში მოქცეულ რეგულარული გამოსახულების ცალკეულ ნაწილებს ***იტერაციული რგოლები***, ხოლო მრგვალ ფრჩხილები მოქცეულ ნაწილებს – ***ალგებრული რგოლები*** ვუწოდოთ. სხვა იტერაციული (ალგებრული) რგოლის შემცველ იტერეტიულ (ალგებრულ) რგოლს ***რთული***, ხოლო ასეთი რგოლების არმქონე რგოლი – ***მარტივი რგოლი*** ვუწოდოთ.

შემოვიღოთ ელემენტის ცნება. ნებისმიერი (როგორც იტერატიულ, ისე ალგებრულ) რგოლში დიზიუნქციის ნიშნებით დაკავშირებულ წევრებს ამ რგოლის ***ელემენტები*** ვუწოდოთ. ერთი ელემენტის შემცველ რგოლს – ***ერთელემენტიანი,*** ხოლო რამდენიმე ელემენტის შემცველ რგოლს – ***მრავალელემენტიანი რგოლი*** ვუწოდოთ. განსაზღვრების თანახმად შეიძლება არსებობდეს ***ერთელემენტიანი რთული*** ან ***მრავალელემენტიანი მარტივი*** რგოლი.

რეგულარული გამოსახულების მაგალითად ავიღოთ შემდეგი გამოსახულება [**19**]**:**

$$E = (b\, a \vee \{ a \}\, b\, \{ a \vee \{ b \}\, b \}\,)\, a, \qquad (\text{ფ}4.1)$$

სადაც არსებული ლათინური ასოები წარმოადგენს გარკვეული მიმდევრობითი აბსტრაქტული ავტომატის შესასვლელი ალფაბეტის ასოებს.

მოცემულ რეგულარულ გამოსახულებაში შედის მარტივი ერთელემენტიანი **{*a*}** და **{*b*}**, რთული ორელემენტიანი **{*a*∨{*b*}*b*}** იტერატიული რგოლი,

აგრეთვე მარტივი ორელემენტიანი ალგებრული **(*b*∨{*a*}*b*{*a*∨{*b*}*b*})** ელემენტი.

ჩვენი მიზანია დავამუშავოთ მისი გაშლის ფორმალური ალგორითმი.

რეგულარული გამოსახულების გაშლისათვის შემოღებულია მოცემული გამოსახულების ადგილების ცნება. რეგულარული გამოსახულების ***ადგილები*** ეწოდება გამოსახულების ნებისმიერ ორ სიმბოლოს (ასოს, ფრჩხილის, დიზიუნქციის ნიშნის) შორის, აგრეთვე გამოსახულების მარცხენა და მარჯვენა განაპირა ადგილებზე მოთავსებულ განცალკევების სპეციალურ (პატარა ვერტიკალურ) ხაზებს. რეგულარული გამოსახულება თუ მრავალწევრს წარმოადგენს, იგი უნდა ჩაისვას ფრჩხილებში.

რეგულარული გამოსახულების ადგილებს შორის განასხვავებენ ძირითად და არაძირითად ადგილებს. ***ძირითადი ადგილი*** ეწოდება: ***1)*** ადგილს, რომლის უშუალოდ მარცხნივ დგას შესასვლელი ალფაბეტის ასო და ***2)*** საწყის ასოს (რომლის მარჯვნიდან იწყება გამოსახულება). ყველა დანარჩენი ასო ითვლება ***არაძირითად ასოებად.***

რეგულარული გამოსახულების არაძირითადი ადგილების სიმრავლიდან გამოიყოფა ძირითადისწინა ადგილები***. ძირითადისწინა ადგილი*** ეწოდება რეგულარული გამოსახულების ისეთ არაძირითად ადგილს, რომლის უშუალოდ მარჯვნივ დგას შესასვლელი ალფაბეტას ასო.

რეგულარული გამოსახულების ადგილები სპეციფიკური გეომეტრიული ობიექტებია. ამ ობიექტებით მანიპულირების საშუალება რომ გვქონდეს, ისინი ინომრება ნატურალური რიცხვებით და ამის შემდეგ თითოეული კონკრეტული ადგილის რიგითი ნომერი იგივდება ამ გეომეტრიულ ადგილთან.

გეომეტრიული ადგილების დასანომრად გამოვიყენოთ ჩვენს მიერ დამუშავებული შემდეგი სპეციალური ***P-პროცედურა:***

1. ნატურალური ***1,2,…,(n-1)*** ასოებით, სადაც ***n*** ძირითადი ასოების საერთო რაოდენობაა, დავნომროთ ძირითადი ადგილები; ***2.*** პირველი პუნქტის შესრულების შემდეგ წარმოქმნილი რიცხვითი სტრიქონის ქვემოთ გავავლოთ ჰორიზონტალური ხაზი; ***3.*** უნომროდ დარჩენილი არაძირითადი ადგილების გამომსახავი ვერტიკალური ხაზები გავაგრძელოთ წინა პუნქტში გატარებული ჰორიზონტალური ხაზის გადაკვეთამდე და დავნომროთ ნატურალური (***n+1***),(***n+2***),…,(***m-1***), სადაც ***m*** რეგულარული გამოსახულების ადგილების საერთო რაოდენობაა; ***4.*** გავხაზოთ ძირითადის წინა ადგილების ნომრები; ***5.*** პროცედურის დასასრული.

რეგულარული გამოსახულების ფორმას, რომელსაც იგი მიიღებს მასზე ***P-პროცედურის*** ჩატარების შემდეგ რეგულარული გამოსახულების ***N-ფორმა*** ვუწოდოთ. რეგულარული (**დ4.1**) გამოსახულების ***N***-ფორმა მოყვანილია ***დ4.2,ა*** ნახაზზე.

დ4.3. რეგულარული გამოსახულების დამუშავების საკითხები

შემოვიღოთ შემდეგი განსაზღვრებები:

1) რეგულარული ***გამოსახულების საწყისი და ბოლო ადგილები*** ვუწოდოთ ამ გამოსახულების განაპირა მარცხენა და მარჯვენა ადგილებს. (**დ5.1**) გამოსახულების საწყისი და ბოლო ადგილებია მისი **0** და **8** ადგილები;

2) რეგულარული გამოსახულებაში არსებული იტერატიული და ლოგიკური ***რგოლის საწყისი ადგილი*** ვუწოდოთ მოცემული რგოლის პირველი (გამხსნელი) ფრჩხილის უშუალოდ მარჯვნივ არსებულ ადგილს.

3) რეგულარული გამოსახულებაში არსებული იტერაციული და ლოგიკური ***რგოლის ბოლო ადგილი*** ვუწოდოთ მოცემული რგოლის უკანასკნელი (დამხურავი) ფრჩხილის უშუალოდ მარჯვნივ არსებულ ადგილს. (**დ4.1**) გამოსახულების ცალკეული იტერატიული და ლოგიკური რგოლების საწყისი ადგილები მოყვანილია ***ცხრილ 1-ში*** (ნახ. ***დ4.2,ბ***);

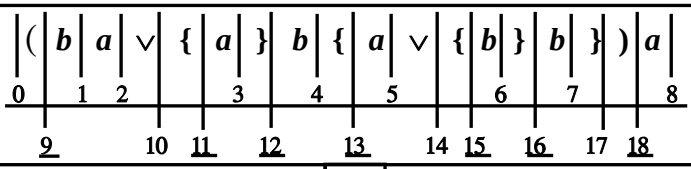
4) რეგულარული გამოსახულებაში არსებული (იტერაციულ და ალგებრულ) რგოლის ***ელემენტის საწყისი ადგილი*** ვუწოდოთ ამ ელემენტის პირველი სიმბოლოს (ასოს ან ფრჩხილის) უშუალოდ მარცხნივ არსებულ ადგილს;

5) რეგულარული გამოსახულებაში არსებული (იტერაციულ და ალგებრულ) რგოლის ***ელემენტის ბოლო ადგილი*** ვუწოდოთ ამ ელემენტის პირველი სიმბოლოს (ასოს ან ფრჩხილის) უშუალოდ მარჯვნივ არსებულ ადგილს;

ცხრილ ***1-ში*** (ნახ. ***დ4.2,ბ***) (**დ4.1**) გამოსახულების რგოლები დაშლილია ცალკეულ ელემენტებად და მითითებულია მათი საწყისი და ბოლო ადგილების ნომრები.

მიმდევრობითი ავტომატის შესასვლელ სიტყვებად რეგულარული გამოსახულების გაშლის პროცესი ინტერპრეტირდება როგორც მოცემული გამოსახულების ერთი ადგილიდან მეორე ადგილზე გადასვლების მიმდევრობა, რომელიც იწყება მოცემული გამოსახულების საწყისი ადგილიდან და მთავრდება ბოლო ადგილზე. ამ გადასვლების ანალიზურად ჩაწერისათვის შემოვიტანოთ დიქოტომიის ცნებაზე დამყარებული ახალი მათემატიკური აპარატი. ***დიქოტომია*** (ბერძ. ***dichotomia*** – ორად გაკვეთა) ვუწოდოთ რეგულარული გამოსახულების ორ ადგილს, რომელთა შორის ხდება გადასვლა; მასში ***აქტიურ ადგილად*** ჩავთვალოთ ადგილი, რომლიდანაც ხდება გადასვლა, ხოლო ***პასიურ ადგილად*** – ადგილი, რომელშიც ხდება გადასვლა.

ა)



ბ)

ცხრილი 1

იტერატიული და (ალგებრული) რგოლები	საწყისი ადგილი	ბოლო ადგილი
{a}	10	12
{b}	14	16
{a∨{b}b}	4	17
(ba∨{a}b{a∨{b}b})	0	18

დ)

ცხრილი 3

წესები	I გვარის დიქოტომიები
I.1 წესი	(1; 2)$_a$
I.2 წესი	(9;1)$_b$; (11;3)$_a$; (12;4)$_b$; (13;5)$_a$; (15;6)$_b$; (16;7)$_b$; (18;8)$_a$.

გ)

ცხრილი 2

რეგულარული გამოსახულების დაშლა:		ელემენტების ადგილები	
რგოლებად	რგოლების ელემენტებად	საწყისი	ბოლო
{a}	a	11	3
{b}	b	15	6
{a∨{b}b}	{b}b	13	5
	a	14	7
(ba∨{a}b{a∨{b}b})	ba	9	2
	{a}b{a∨{b}b}	10	17

ე)

ცხრილი 4

წესები	რეგოლები	II გვარის დიქოტომიები
II.1 წესი	(ba∨{a}b{a∨{b}b})	(0; 9), (0; 10)
II.2 წესი		(18; 2), (18; 11)
II.3 წესი	{a}	(10; 12), (10; 11)
	{a∨{b}b}	(4; 13), (4; 14) (4; 17)
	{b}	(14; 16), (14; 15)
II.4 წესი	{a}	(12; 11), (12; 3)
	{a∨{b}b}	(17; 13), (17; 14), (17; 5),(17;7_)
	{b}	(16; 15), (16; 6)

ნახ. 4.2. *I და II გვარის დიქოტომიების განსაზღვრის საკითხები*

ერთი ადგილიდა მეორეზე გადასვლა ხდება ***უშუალოდ*** ან ***შესასვლელი ალფაბეტას ასოს*** მეშვეობით [*19*]. პირველი სახის გადასვლის დროს შესასვლელი ალფაბეტის ასო არ წარმოიქმნება, ხოლო მეორე შემთხვევაში წარმოიშვება ის ასო, რომლის მეშვეობითაც განხორციელდა მოცემული გადასვლა.

I გვარის დიქოტომია ვუწოდოთ რეგულარული გამოსახულების იმ ორი ადგილის ერთობლიობას, რომელთა შორისაც გადასვლა შესასვლელი ასოს მეშვეობით ხდება, ხოლო ***II გვარის დიქოტომია*** - რეგულარული გამოსახულების იმ ორი ადგილის ერთობლიობას, რომელთა შორისაც ხდება უშუალო გადასვლა;

შემოტანილი განსაზღვრების თანახმად ***I*** გვარის დიქოტომიისა დახმარებით წარმოიქმნება შესასვლელი ალფაბეტას ასო, ხოლო ***II*** გვარის დიქოტომიის საშუალებით მომდვნო ასოს წარმოსაქმნელად საჭირო ახალ სატარტო ადგილზე ხდება გადანაცვლება.

აქტიური δ, პასიური η ადგილებისაგან შემდგარი ***I*** გვარის ისეთი დიქოტომია და რომლის δ-დან η–ზე გადასვლა ხდება $x_i \in A$ ასოს მეშვეობით (სადაც A მიმდევრობითი ავტომატის შესასვლელი ალფაბეტაა), პირობითად აღვნიშნოთ როგორც $(\underline{\delta}, \eta)_{x_i}$, ხოლო აქტიური α და პასიური β ასო-ასოსასაგან შემდგარი ***II*** გვარის დიქოტომია – როგორც $(\underline{\alpha}, \beta)$.

ჩამოვაყალიბოთ ***I*** და ***II*** გვარის დიქოტომიების გასაზღვრის წესები.

1) I რიგის დიქოტომიის განსაზღვრის წესები:

I.1. შესასვლელი ალფაბეტის რომელიმე ასოს უშუალოდ მარცხნივ არსებული ძირითადი ადგილი ამ ასოს უშუალოდ მარჯვნივ არსებულ ადგილთან წარმოქმნის ***I*** გვარის დიქოტომიას, რომელშიც იგი არის აქტიური. ამ დიქოტომიის განხორციელებისას წარმოიქმნება დიქოტომიის ელემენტებს შორის არსებული შესასვლელი ასო.

I.2. შესასვლელი ალფაბეტის რომელიმე ასოს უშუალოდ მარცხნივ არსებული ძირითადისწინა ადგილი ამ ასოს უშუალოდ მარჯვნივ არსებულ ადგილთან წარმოქმნის ***I*** გვარის დიქოტომიას, რომელშიც იგი არის აქტიური. ამ დიქოტომიის განხორციელებისას წარმოიქმნება დიქოტომიის ელემენტებს შორის არსებული შესასვლელი ასო.

ზემოთ ფორმულირებული წესების მეშვეობით რეგულარული (**ᲓᲐ4.1**) გამოსახულებისათვის განსაზღვრული ***I*** რანგის დიქოტომიები მოყვანილია ცხრილ ***3***-ში (***ნახ.ᲓᲐ4.2,ᲓᲐ***).

2) II რიგის დიქოტომიის განსაზღვრის წესები:

II.1. ალგებრული რგოლის საწყისი ადგილი ***II*** გვარის დიქოტომიებს წარმოქმნის ამ რგოლის თითოეული ელემენტის საწყის ადგილებთან და ამ დიქოტომიებში იგი არის აქტიური;

I.2. ალგებრული რგოლის ბოლო ადგილი ***II*** გვარის დიქოტომიებს წარმოქმნის ამ რგოლის თითოეული ელემენტის ბოლო ადგილებთან და ამ დიქოტომიებში აქტიური არის ელემენტთა ბოლო ადგილები;

II.3. იტერაციული რგოლის საწყისი ადგილი ***II*** გვარის დიქოტომიებს წარმოქმნის:

- ამ რგოლის ბოლო ადგილთან;
- ამ რგოლის თითოეული ელემენტის საწყის ადგილებთან და ორივე შემთხვევაში იგი არის აქტიური.

II.4. იტერაციული რგოლის ბოლო ადგილი ***II*** გვარის დიქოტომიებს წარმოქმნის:

- ამ რგოლის თითოეული ელემენტის საწყის ადგილებთან და ამ დიქოტომიებში იგი არის აქტიური;
- ამ რგოლის თითოეული ელემენტის ბოლო ადგილებთან და ამ დიქოტომიებში აქტიურია ელემენტთა ბოლო ადგილები

II.5. ცარიელი სიმბოლოები საწყისი და ბოლო ადგილები ერთმანეთთან წარმოქმნის აქტური ადგილის არმქონე ***II*** გვარის დიქოტომიებს, რომლებიც ცარიელ (გადაგვარებულ) დიქოტომიებს წარმოადგენს

II.6. ***II*** რიგის დიქოტომიების წარმოქმნის ზემოთ ჩამოთვლილი წესებისაგან განსხვავებული სხვა წესები არ არსებობს.

§4.4. რეგულარული გამოსახულების არაძირითადი ადგილების მონიშვნის ფორმალური მეთოდის დამუშავება

რეგულარული გამოსახულებიდან მიმდევრობითი ავტომატის შესასვლელი სიტყვის გაშლის პროცესში აღნიშნული სიტყვის ასოები, როგორც ზემოთ აღვნიშნეთ, წარმოიქმნება ძირითადი ან წინაძირითადი ადგილებიდან გარკვეულ არაძირითად ადგილებზე გადასვლის მეშვეობით. კონკრეტულ არაძირითად ადგილზე გადასვლა შეიძლება არა ნებისმიერი ძირითადი ან ძირითადისწინა ადგილიდან, არამედ მხოლოდ ზოგიერთი მათგანიდან. ეს იმას ნიშნავს, რომ კონკრეტული არაძირითადი ადგილისათვის არსებობს ისეთი ძირითადი ან ძირითადის წინა ადგილები, რომლებიდანაც შესაძლებელია მოხდეს მასზე გადასვლა.

არაძირითადი ადგილის ნომრის ქვემოთ იმ ძირითადი და ძირითადის წინა ადგილთა ნომრების მიწერას, რომლებიდანაც შესაძლებელია მოხდეს აღნიშნული ადგილებიდან მასზე გადასვლა, ამ ***არაძირითადი ადგილის მონიშვნა*** ეწოდება.

რეგულარული გამოსახულების ფორმას, რომელსაც იგი იღებს არაძირითადი ადგილების მონიშვნის შემდეგ, ***F*-ფორმა** ვუწოდოთ.

N*-ფორმით** წარმოდგენილ რეგულარულ გამოსახულების არაძირითადი ადგილების მონიშვნის არსებული წესი [**5,19,25**] მოითხოვს ხელით გადარჩევას, რაც შეუძლებელს ხდის კომპიუტერზე შესაბამისი პროცესის ორგანიზებას. ჩვენი მიზანი აღნიშნული ნაკლის გამოსწორებაა. მისი მიღწევა შეიძლება ზემოთ განსაზღვრული ***II რიგის დიქოტომიების გამოყენებით. ცხრილ ***4*-**ში (ნახ. ***ლ4.2,ე***) მოყვანილია განხილული რეგულარული გამოსახულებაში არსებული რგოლების ელემენტები და თითოეული რგოლისათვის განსაზღვრული ***II*** რიგის დიქოტომებიების ჩამონათვალი.

II რიგის ყველა დიქოტომიის გაერთიანებით მიღებული სიმრავლე აღვნიშნოთ D_{II} სიმბოლოთი და იგი წარმოვიდგინოთ ორი ქვესიმრავლის გაერთიანებით მიღებული სიმრავლის ფორმით:

$$D_{II} = D_{II(1)} \cup D_{II(2)},$$

სადაც $D_{II(1)}$ ქვესიმრავლე შედგება ***II*** რიგის მხოლოდ ისეთი დიქოტომიებისაგან, რომლებიც შეიცავს ერთ ძირითად ადგილს მაინც, ხოლო $D_{II(2)}$ ქვესიმრავლე შედგება მხოლოდ არაძირითადი ადგილების შემცველი დიქოტომიებისაგან. განსახილველი (***ლ4.1***) გამოსახულების D_{II} სიმრავლე საკუთარივე ქვესიმრავლეებით მოყვანილია ცხრილ ***5*-**ში (ნახ.***ლ4.3,ა***)

$D_{II(1)}$ და $D_{II(2)}$ ქვესიმრავლეებისათვის შევადგინოთ სპეციალური ბარათები, რომლებსაც შესაბამისად $D_{II(1)}$ - და $D_{II(2)}$ -ბარათები ვუწოდოთ.

▲ $D_{II(1)}$-ბარათი შედგება ნატურალური ***0,1,…,n*** რიცხვებით დანომრილი (***n+1***)რაოდენობის სვეტისა და ასევე ნატურალური ***n+1, n+1,…,m*** რიცხვებით დანომრილი ***m*-(*n+1*)** რაოდენობის სტრიქონისაგან, სადაც ***n*** არის რეგულარული გამოსახულების ძირითადი ადგილების, ხოლო ***m*** ამავე გამოსახულების ყველა ადგილის რაოდენობა. ***II*** რიგის ყოველ $(\alpha, \beta) \in D_{II(1)}$ ქოქოტომიას, სადაც $\alpha \in (1,2,\ldots,n)$, $\beta \in (n+1, n+1,\ldots,m)$, შეესაბამება ამ ცხრილის α სტრიქონისა და β სვეტის გადაკვეთაზე არსებული უჯრა; ამ უჯრაში უნდა ჩავწეროთ აღნიშნული დიქოტომიის აქტიური წევრის α ნომერი

▲ $D_{II(2)}$-ბარათი შედგება ერთი სვეტისა და ***n+1, n+2,…,m*** რიცხვებით დანომრილი ***m*-(*n+1*)** რაოდენობის მწკრივისაგან. ***II*** რიგის ყოველ დიქოტომია $(\delta, \eta) \in D_{II(2)}$-ს, სადაც $\alpha \in (1,2,\ldots,n)$, $\beta \in (n+1, n+1,\ldots,m)$, შეესაბამება ისარი, რომელიც იწყება მოცემული დიქოტომიის აქტიური წევრის δ ნომრის შესაბამისი სტრიქონიდან და მთავრდება ამავე დიქოტომიის არააქტიური წევრის β ნომრის სტრიქონში; ამასთანავე, ისარი მიმართულია არააქტიური β ნომრის შესაბამისი სტრიქონისაკენ.

$D_{II(1)}$ –ბარათისთვის მარჯვნივ $D_{II(2)}$ - ბარათის მიდგმით მიიღება გაერთიანებული D_{II} –ბარათი, რომელიც ***ლ4.3,ბ*** ნახაზზეა მოყვანილი. მოცემულ ნახაზზე D_{II} –ბარათის მისაღებად შავი ფერის უჯრები საჭიროა ჩავთვლოთ ცარიელ უჯრებად.

შემოვიტანოთ რამდენიმე განსაზღვრება:

1) D_{II} –ბარათის ურთიერთდაკავშირებული სტრიქონები ვუწოდოთ ამ ბარათის სტრიქონთა წყვილს, რომლებშიც $D_{II(2)}$ – ბარათის ისრების ბოლოებია განთავსებული; ამასთანავე სტრიქონს, საიდანაც იწყება ისარი, ვუწოდოთ ***მაღალი დონის სტრიქონი***, ხოლო სტრიქონს, სადაც მთავრდება ისარი – ***დაბალი დონის სტრიქონი.***

2) D_{II} –ბარათის ურთიერთდაკავშირებული სტრიქონების დონეთა გათანაბრების ოპერაცია ვუწოდოთ აქტიურ სტრიქონში არსებული რიცხვების პასიურ სტრიქონში გადაკოპირების პროცესს.

ა)

ცხრილი 5

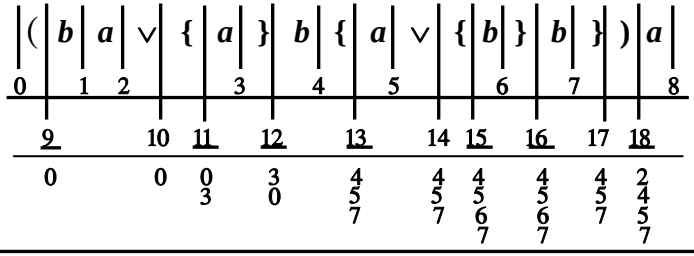
D_{II} სიმრავლე	
$D_{II(1)}$ ქვე-იმრავლე	$D_{II(2)}$ ქვე-იმრავლე
(0; 9), (0; 10), (4; 13), (4; 14), (4; 17), (12; 3), (16; 6), (17; 5), (17; 7), (18; 2).	(10; 11), (10; 12), (12; 11), (14; 15), (14; 16), (16; 15), (17; 13), (17; 14), (18; 17).

ბ)

D_{II} - ბარათი

	$D_{II(1)}$ ბარათი									$D_{II(2)}$ ბარათი
	0	1	2	3	4	5	6	7	8	
9	0									
10	0									
11	0			3						
12	0			3						
13					4	5		7		
14					4	5		7		
15					4	5	6	7		
16					4	5	6	7		
17					4	5		7		
18			2		4	5		7		

გ)



ნახ. ლ4.3. *რეგულარული გამოსახულების წარმოდგენა* ***F-*** *ფორმის სახით*

დავუშვათ, რომ შეერთებული D_{II}- ბარათში ურთიერთდაკავშირებული სტრიქონებია α და β. ამ სტრიქონებიდან მაღალი დონის თუ არის ***α,*** ხო-

ლო დაბალი დონის - β სტრიქონი, მაშინ მათი დონეების გათანაბრების ოპერაცია აღვნიშნოთ როგორც $\alpha\rightarrow\beta$; იგი α სტრიქონში არსებული რიცხვების β სტრიქონში გადაკოპირებას ნიშნავს. D_{II}– ბარათში ამავდროულად თუ β-თან ურთიერთდაკავშირერებული δ სტრიქონიც აღმოჩნდება, მაშინ შესასრულებელი იქნებ დონეთაა გათანაბრების რთული $\alpha\rightarrow\beta\rightarrow\delta$ ოპერაცია. იგი ნიშნავს, რომ რიცხვები ჯერ α სტრიქონიდან გადაკოპირდეს β სტრიქონში და მხოლოდ ამის შემდეგ მოხდეს უკანასკნელ სტრიქონში ფორმირებული ყველა რიცხვი გადაკოპირება δ სტრიქონში.

განხილულ მაგალითში (იხ.ნახ. ***დ4.2,ბ***) გვაქვს ურთიერთდაკავშირებული სტრიქონების დონეთა გათანაბრების შემდეგი რთული ოპერაციები:

$$10\rightarrow12\rightarrow11;\quad 14\rightarrow16\rightarrow15;\quad 18\rightarrow17\rightarrow13;\ 18\rightarrow17\rightarrow14. \quad (\textbf{დ4.2})$$

3) გათანაბრებული D_{II} - ბარათი ვუწოდოთ ისეთ D_{II} - ბარათს, რომელშიც გათანაბრებულია ყველა ურთიერთდაკავშირებულ სტრიქონთა დონეები.

განხილულ მაგალითისთვის ***დ4.3,ბ*** ნახაზზე ნაჩვენებ ბარათში გათანაბრების ყველა პროცესის შედეგად გაჩენილი რიცხვები შავი ფერის უჯრედებშია ნაჩვენები. ყველა (როგორც თეთრ, ისე შავ) უჯრებში არსებული რიცხვების ერთობლიობით წარმოიქმნება ***გათანაბრებული D_{II} - ბარათი***.

გათანაბრებული D_{II} - ბარათი გამოყენებით შესაძლებელია ჩამოვაყალიბოთ რეგულარული გამოსახულების არაძირითადი ***i***-ური ($i\in\{n+1, n+1,\ldots,m\}$) ადგილის მონიშვნის, ე.ი. აღნიშნული გამოსახულების ***F***- ფორმის მიღების პროცესის ფორმალური შესრულების შემდეგი წესი

რეგულარულ გამოსახულების F-ფორმის მისაღებად, ანუ არაძირითადი ***i***-ური, $i\in(n+1, n+2,\ldots, m)$ ადგილის მონიშვნისათვის აუცილებელი და საკმარისია ***გათანაბრებული D_{II} -ბარათის i***-ურ სტრიქონში მდგარი ყველა რიცხვი სვეტის სახით ქვემოთ მივუწეროთ ***N***-ფორმის რეგულარული გამოსახულების ***i***-ური ადგილი ნომერს.

ზემოთ ფორმულირებული წესის გამოყენებით მიღებული განხილული რეგულარული გამოსახულების ***F***-ფორმა ***დ.4.3,ბ*** ნახაზზეა მოყვანილი.

დ4.3. რეგულარული გამოსახულების გაშლის ფორმალური მეთოდი

აბსტრაქტული სასრული ავტომატის შესასვლელი და გამოსასვლელი სიტყვების ჩაწერის ფორმაა აღნიშნული ავტომატის გადასვლებისა და გამოსასვლელების ცხრილი. აღნიშნულიდან გამომდინარე ცალკეულ სიტყვებად რეგულარული გამოსახულების გაშლა აღნიშნული ცხრილის ფორმირების ტოლფასია.

რეგულარული გამოსახულებიდან მურის ავტომატის გადასვლების ცხრილის მიღების არსებული წესიც [5,19,25] არაფორმალიზებულია (მოითხოვს ხელით გადარჩევას) რაც შეუძლებელს ხდის კომპიუტერით ამ პროცესის ავტომატიზებას. აღნიშნული ნაკლის გამოსწორება შეიძლება ზემოთ განსაზღვრული I რიგის დიქოტომიების გამოყენებით.

I რიგის დიქოტომიების D_I სიმრავლე წარმოვადგინოთ შემდეგნაირად:

$$D_I = D_{I(x_1)} \cup D_{I(x_2)} \cup \ldots \cup D_{I(x_m)}, \qquad (\text{დ}4.2)$$

სადაც $D_{I(x_i)}$ წარმოადგენს შესასვლელი x_i ასოს წარმომქმნელი I გვარის დიქოტომიების სიმრავლეს, რომელთა განხორციელებისას წარმოიქმნება შესასვლელი x_j, $j=1,2,\ldots, m$ ასო.

განხილული მაგალითისათვის (იხ. ნახ.*დ.3,გ*) შესასვლელი $A=\{a,b\}$ ალფაბეტი შეიცავს 2 ასოს, ამიტომ დ4.2 გამოსახულება იღებს სახეს:

$$D_I = D_{I(a)} \cup D_{I(b)} = [(\underline{1};2)_a;\ (\underline{11};3)_a;\ (\underline{13};5)_a;\ (\underline{18};8)_a\ ] \cup$$
$$\cup\ [\ (\underline{9};1)_b;\ (\underline{12};4)_b;\ (\underline{15};6)_b;\ (\underline{16};7)_b\ ]. \qquad (\text{დ}4.3)$$

D_I სიმრავლის გამოყენებით ***შევადგინოთ D_I –ბარათი*** (ნახ.*დ.4.3,ა*) შემდეგნაირად:

▲ $D_{I(a)} \subset D_I$ ქვესიმრავლეში შემავალი თითოეული დიქოტომიისათვის გამოვყოთ საკუთარი სვეტი, რომელთა ერთობლიობა წარმოქმნის $D_{I(a)}$-***ველს***. ასევე $D_{I(b)}$ ***ველს*** წარმოქმნის $D_{I(b)} \subset D_I$ ქვესიმრავლის დიქოტომიებისათვის გამოყოფილი სვეტები;

▲ D_I–ბარათში გავითვალისწინოთ $m+1$ სტრიქონი და დავნომროთ ისინი $0,1,\ldots,m$ რიცხვებით (m უდიდესი რიცხვია იმ რიცხვებს შორის, რომლითაც დანომრილია ძირითადი ადგილები).

■ ჩავთვალოთ, რომ D_I–ბარათის ქვესვეტები $i \in (0,1,\ldots,m)$ სტრიქონით ***გადაიფარება,*** თუ ამ ქვესვეტების დიქოტომიის აქტიური წევრები რეგულარული გამოსახულების F-ფორმაში მონიშნულია i რიცხვით (თუ აქტიურია რეგულარული გამოსახულების ძირითადისწინა ადგილი) ან როდესაც აქტიური ადგილის ნომერი ემთხვევა i რიცხვს (თუ აქტიურია რეგულარული გამოსახულების ძირითადი ადგილი).

■ ***D_I–ბარათის უჯრები შევსების წესი.*** D_I–ბარათის უჯრებში, რომლებიც დგას i სტრიქონისა და ან სტრიქონით გადაფარული ქვესვეტების გადაკვეთებზე, ჩაიწერება აღნიშნული ქვესვეტების შესაბამის დიქოტომიებში არსებული პასიური ადგილის ნომრები. D_I–ბარათის დანარჩენი უჯრები რჩება შეუვსებელი.

■ განვიხილოთ D_I–ბარათის რომელიმე, კერძოდ $(\underline{11};3)$, დიქოტომიით აღნიშნული ქვესვეტის უჯრების შევსების მაგალითი. დიქოტომიაში შემავალი ადგილი 11 მონიშნულია რიცხვებით 0 და 3 (იხ. ნახ. *დ.3,გ*), ამიტომ

ეს ქვესვეტი დაიფარება D_I– ბარათის ***0*** და ***3*** სტრიქონებით. აქედან გამომდინარე უჯრებში, რომლებიც განთავსებულია სვეტისა და ***0;1*** სტრიქონების გადაკვეთებზე, ჩაიწერება დიქოტომიური ადგილის ნომერი ***3.*** ანალოგურად შეივსება D_I–ბარათის დანარჩენი ადგილები (იხ. ნახ. ***ლ4.4,ა***).

D_I– ბარათის გამოყენება საშუალებას ფორმალური მეთოდის გამოყენებით ავაგოთ ***მურის ავტომატის პირველადი გადასვლების ცხრილი,*** რომლის მარტივი გარდაქმნით მიიღება ***მურის ავტომატის გადასვლების ცხრილი.***

D_I- ბარათის მეშვეობით მურის ავტომატის გადასვლების პირველადი ცხრილის აგების ალგორითმი:

1. ავტომატის შინაგან მდგომარეობებად აიღება ძირითადი ადგილების M სიმრავლის ქვესიმრავლეები. ამ ქვესიმრავლეებით წარმოიშვება გადასვლების პირველადი ცხრილის სტრიქონები, ამიტომ მათ ამ სტრიქონების წარმომშობი ქვესიმრავლეები ვუწოდოთ და აღვნიშნოთ $L_k, k = 1,2,\ldots$ ასოებით; გადასვლების ცხრილის სტრიქონები აღინიშნება მათი წარმომშობი ქვესიმრავლით;

2. გადასვლების პირველადი ცხრილის (ნახ.***ლ.4.4,ბ***) აგება იწყება ***საწყი-სი მდგომარეობის*** შესაბამისი სტრიქონის შედგენით. მის წარმომშობ ქვესიმრავლედ აპრიორულად აიღება $0 \in M$ ადგილის შემცველი წარმომშობი $L_1 = = (0)$ ქვესიმრავლე.

3. აღნიშნული ცხრილის დანარჩენი მდგომარეობების შესაბამისი სტრიქონების ფორმირდება შედგენილ სტიქონებში მათი წარმომშობი ქვესიმრავლეების გამოჩენის შემდეგ;

4. წარმომშობი $L_k=(\alpha_1, \alpha_2,\ldots, \alpha_M)$ ქვესიმრავლის მიხედვით (სადაც $\alpha_i \in N$, ხოლო N – ნატურალური რიცხვების სიმრავლეა) გადასვლების პირველადი ცხრილის შედგენის წესი: ***ა)*** განვიხილოთ ***გადასვლების პირველადი ცხრილის*** სტრიქონი, რომელსაც შეესაბამება L_k ქვესიმრავლე და სვეტები, რომლებიც შეესაბამება x_{ji} შესასვლელები; მათ გადაკვეთებზე არსებული უჯრები აღვნიშნოთ როგორც (L_k, x_{ji}). მათი რაოდენობა უდრის L_k ქვესიმრავლეში არსებული α_i რიცხვების რაოდენობას; ***ბ)*** განვიხილოთ D_I–ბარათის ის სტრიქონები, რომელთა ნომრები ემთხვევა L_k ქვესიმრავლეში არსებულ რიცხვებს; ***გ)*** D_I–ბარათის წინა პუნქტში განხილულ სტრიქონებისა და ამავე ბარათის $D_{I(ji)}$ –ველის გადაკვეთაზე არსებული რიცხვებისაგან შევადგინოთ სიმრავლე და იგი ჩავწეროთ გადასვლების პირველადი ცხრილის ***ა*** პუნქტში ფორმირებულ (L_k, x_{ji}) უჯრაში; დ) წინა პუნქტში ფორმირებული სიმრავლეები წარმოადგენს წარმომშობ სიმრავლეს. თუ ასეთი სიმრავ-

ლე არ არსებობს უკვე ფორმირებულ წარმომშობ სიმრავლეებს შორის, მას შეესაბამება გადასვლების პირველადი ცხრილის ახალი სტრიქონი.

5. გადასვლების ცხრილის სვეტ «გამოსასვლელის» იმ სტრიქონში, რომელსაც შეესაბამება მონიშნული წარმომშობი ქვესიმრავლე, ჩაიწერება ციფრი ***1***, ხოლო ამ სვეტის დანარჩენ სტრიქონებში ჩაიწერება ციფრები ***0***.

6. გადასვლების პირველადი ცხრილის აგება დამთავრდება ყველა წარმომშობი სიმრავლის შესაბამისი სტრიქონის ფორმირების შემდეგ.

ა)

D_I-ბარათი

	$D_{I(a)}$ ველი				$D_{I(b)}$ ველი			
	(1;2)	(11;3)	(13;5)	(18;8)	(9;1)	(12;4)	(15;6)	(16;17)
0		3			1	4		
1	2							
2				8				
3		3				4		
4			5	8			6	7
5			5	8			6	7
6							6	7
7			5	8			6	7
8								

ბ)

გადასვლების პირველადი ცხრილი

	S მდგომარეობები	x_j შესასვლელები		გამოსასვლელი
№	წარმომშობი ქვესიმრავლეები	$x_{j1}=a$	$x_{j2}=b$	
1	L_1=(0)	(3)	(1;4)	0
2	L_2=(3)	(3)	(4)	0
3	L_3=(1;4)	(2;5;8)	(6;7)	0
4	L_4=(4)	(5;8)	(6;7)	0
5	L_5=(2;5;8)	(5;8)	(6;7)	1
6	L_6=(6;7)	(5;8)	(6;7)	0
7	L_7=(5;8)	(5;8)	(6;7)	1

გ)

S	შესასვლელები		გამოსასვლელი
	a	b	
1	2	3	0
2	2	4	0
3	5	4	0
4	5	4	0
5	5	4	1

ნახ. 4.4. ავტომატის გადასასვლელებისა და გამოსასვლელების ცხრილის აგება

განვიხილოთ ჩვენი მაგალითისათვის გადასვლების პირველადი ცხრილის შედგენის პროცესის ფრაგმენტი.

ცხრილის შედგენას ვიწყებთ საწყისი მდგომარეობის შესაბამისი სტრიქონის აგებით, რომლისთვისაც წარმომშობ ქვესიმრავლედ აპრიორულად ვიღებთ L_1**=(0)** ქვესიმრავლეს. იგი შეიცავს ელემენტ ***0***-ს.

D_I– ბარათის (იხ. ნახ.***ლ.4.4,ა***) სტრიქონ ***0***-ისა და $D_{I(a)}$ ველის გადაკვეთაზე დგას რიცხვი ***3,*** ხოლო ამ სტრიქონისა $D_{I(b)}$ ველის გადაკვეთაზე რიცხვები ***1*** და ***4.*** ამიტომ ჩნდება L_2**=(3)** და L_3**=(1;4)** სიმრავლეები. ამათგან პირველი სიმრავლე L_2**=(3)** ჩაიწერება **(**L_1**;** x_{j1}**=*a*)** უჯრაში, ხოლო მეორე სიმრავლე L_3**=(1;4) -** **(**L_1**;** x_{j2}**=*b*)** უჯრაში.

მიღებული სიმრავლეები წარმოადგენს ახალ წარმომშობ L_2 **= (3)** და L_3 **= (1;4)** ქვესიმრავლეებს გადასვლების პირველად ცხრილში შესაბამება ***2*** და ***3*** სტრიქონები.

▉ განვიხილოთ გადასვლების პირველად ცხრილში სტრიქონ ***3***-ის შევსების პროცესი. მას შეესაბამება L_3**=(1;4)** ქვესიმრავლე.

D_I– ბარათის (იხ. ნახ.***ლ.4.4,ა***) ***1*** და ***4*** სტრიქონებისა $D_{I(a)}$ ველის გადაკვეთაზე დგას რიცხვები ***2,4,8,*** ხოლო ამ სტრიქონებსა $D_{I(b)}$ ველის გადაკვეთაზე რიცხვები ***6*** და ***7.*** ამიტომ ჩნდება L_5**=(2;4;8)** და L_6**=(6;7)** სიმრავლეები. ამათგან პირველი სიმრავლე L_5**=(2;4;8)** ჩაიწერება **(**L_3**;** x_{j1}**=*a*)** უჯრაში, ხოლო მეორე სიმრავლე L_6**=(6;7) -** **(**L_3**;** x_{j2}**=*b*)** უჯრაში.

მიღებული სიმრავლეები წარმოადგენს ახალ წარმომშობ L_4**=(2;4;8)** და L_5**=(6;7)** ქვესიმრავლეებს გადასვლების პირველად ცხრილში შესაბამება ***5*** და ***6*** სტრიქონები. ასე შეივსება გადასვლების პირველადი ცხრილის სხვა სტრიქონებიც.

მონიშნულ წარმომშობ ქვესიმრავლეებს წარმოადგენს L_5**=(2;4;8)** და L_7 **=** **=(*5;8*)** ქვესიმრავლეები (მათში შედის განხილული რეგულარული გამოსახულების ბოლო ადგილის ნომერი ***8***), ამიტომ სვეტში «გამოსასვლელი» ციფრი ***1*** ჩაიწერება ამ სვეტის მე-***5*** და მე-***7*** სტრიქონებში (იხ. ნახ. ***ლ.4.4,ა***).

▉ გადასვლების მიღებულ პირველად ცხრილი შეიცავს ***8*** სტრიქონს, რომელთაგანაც ერთნაირია მე-***5*** და მე-***6*** სტრიქონები. მათი გაერთიანება სტრიქონების რაოდენობას ***7***-მდე ამცირებს. შემდგომში აღნიშნული ცხრილის მინიმიზების [***1,5***] მეშვეობით გვრჩება ხუთი სტრიქონის მქონე გადასვლების მინიმიზებული ცხრილი (ნახ. ***ლ.4,4,ბ***).

მიღებული გადასვლების ცხრილის შესაბამისი აბსტრაქტული ავტომატის ეკვივალენტური მმართველი დისკრეტული მოწყობილობა შეიძლება რეალიზდეს როგორც პროგრამულად, ისე აპარატურულადაც.

ლიტერატურა

1. ***დუნდუა ა.ა., საპოჟნიკოვი ვალ. ვ., საპოჟნიკოვი ვლ. ვ.*** დისკრეტულ მოწყობილობათა საკითხები. - თბ.: ***სტუ, 1990*** წ. – 119 გვ.

2. დუნდუა ა.ა. კომპიუტერული სისტემებისა და საინფორმაციო ტექნოლოგიების თეორიული საფუძვლები. - თბ.: ***სტუ,, 2014*** წ. - 258 გვ.

3. დუნდუა ა.ა. ავტომატიკისა და ტელემექანიკის სასადგურო და საგადასარბენო სისტემები, II ნაწ. - თბ.: ***სტუ,, 2013*** წ. - 478 გვ.

4. Гивоне Д., Россер Р. Микропроцессор и микрокомпьютеры. М.: Мир, ***1983***. – 464 с.

5. Глушков В. М. Синтез цифровых автоматов. – М.:Гос. изд.физико-математической литературы, **1962**. -476 ст.

6. Дундуа А. А. Формализация некоторых этапов программной реализации последовательностных схем на микропроцессорах - Проблемы разработки, внедрения и эксплуатации микроэлектронных систем железнодорожной автоматики и телемеханики: сб. науч. тр. - СПб.: Петербургский государственный университет путей сообщения, ***2005.*** Стр. 88 – 98.

7. Дундуа А. А. Координатный метод перехода от языка заказчика к языку разработчика цифровых систем управления - Автоматика и телемеханика железных дорог России. Техника, технология, сертификация: сб. науч. тр. – СПб.: Петербургский государственный университет путей сообщения, ***2008.*** Стр.51 – 58.

8. Информатика: учебник для бакалавров/под ред. ***Трофимова В.В. –*** *М.*: «Юрайт», ***2013*** -917 с.

9. Кузин А.В., Жаворонков М.А. Микропролцессорная техника. М.: Академия, ***2013*** - 304 с.

10. Колдуэл С. Логический синтез релейных устройств. – М.: Изд. Иностранной литературы, ***1962.*** **-7**37 **с.**

11. **Микропроцессорные системы / Александров Е.К.** и др.**;** Под общ. ред. **Пузанкова Д. В. –** М: СПб.: Политехника, ***2002.*** **–** 935 с.

12. **Микропроцессорные системы централизации /** Сапожников Вл. В. и др.; Под общ. ред. Сапожникова Вл. В. – М.: ГОУ «Учебно-методический центр по образованию на железнодорожном транспорте». ***2008.***- 398 с.

13. Новиков Ю. В., Скоробогатов П. К. Основы микропроцессорной техники. М.: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний. - ***2012.*** 357с.

14. Новожилов О. П. Архитектура ЭВМ и систем . *М.*: «Юрайт», ***2016.*** -527 с.

15. Новожилов О. П. Информатика . М.: «Юрайт», **2012.** -564 с.

16. Прангишвили И. В. Микропроцессоры и микро-ЭВМ. М.: Энергия, ***1979.*** – 232 с.

17. Сапожников В. В., Сапожников Вл. В., Борисенко Л. И. Какими должны быть микропроцессорные системы железнодороной автоматики и телемеханики***,*** «Автоматика, телемеханика и связь», ***1988,*** №5, с. 32-34 .

18. Сапожников В. В., Сапожников Вл. В., Христов Х. А., Гавзов Д.В. Методы построения безопасных микроэлектронных систем железнодорожной автоматики – М.: «Транспорт», ***1995***. стр. 272.

19. Сапожников В. В., Кравцов Ю А., Сапожников Вл.В. Дискретные устройства железнодорожной автоматики, телемеханики и связи. М.: Транспорт, ***1988.*** – 255 с.

20. Системы автоматики и телемеханики на железных дорогах мира/Пер.с англ.;под ред***. Г. Теега,С. Власенко.***-М.:Интекст***, 2010.*** -496 с***.***

21. Уоррен С. Мак-каллок и Вальтер Питтс. Логическое исчисление идей, относящихся к нервной ак-тивности. ***Автоматы.*** Сб.статей под ред. К.Е.Шеннона и Дж. Маккарти. М.: Изд. Иностранной литературы, ***1956***. ст.362-384).

22. Мур Е. Шеннон К. Надежные схемы из ненадежных реле - *Сб. науч. труд* ***К.Шеннона.*** Работы по теории информации и кибернетике. М.: Изд. Иностранной литературы, ***1963***. с.114-153).

23. Шеннон К. Символический анализ релейных и переключательных схем - Работы по теории информации и кибернетике. М.: Изд. Иностранной литературы, ***1963***. с.9-45).

24. Нейман Дж. Вероятностная логика и синтез надежных организмов из ненадежных компононт***. Автоматы.*** Сб. статей под ред. Шеннона К. Е***.*** и Маккарти Дж. М.: Изд. Иностранной литературы, ***1956***. ст.68-140).

25. Фридман А., Менон П. Теория и проектирование переключательных схем. М.:Мир,1978.-580 с.

ს ა რ ჩ ე ვ ი