ბათუმის შოთა რუსთაველის სახელმწიფო უნივერსიტეტი

თენგიზ ხინიკაძე

კომპიუტერის არქიტექტურა და ორგანიზება

ბათუმი - 2013

სახელმძღვანელოში განხილულია კომპიუტერის არქიტექტურისა და ორგანიზაციის საფუძვლები; მოტანილია ზოგადი ცნობები კომპიუტერის შესახებ; აღწერილია ბრძანებათა სისტემის არქიტექტურა; განხილულია ფონ ნეიმანისეული გამომთვლელი მანქანის ფუნქციური ორგანიზაცია; დეტალურადაა გარჩეული სალტეთა ორგანიზაცია, მეხსიერების, მართვის მოწყობილობების, ოპერაციული მოწყობილობებისა და შეტანა/გამოტანის სისტემათა აგებულება; აღწერილია ძირითადი მიმართულებები პროცესორების არქიტექტურაში: გამოთვლების კონვეიერიზაცია, არქიტექტურები ბრძანებათა სრული და შეკვეცილი ნაკრებით, სუპერკონვეიერული და სუპერსკალარული პროცესორები. იგი განკუთვნილია კომპიუტინგის სპეციალობის ბაკალავრიატის საფეხურის სტუდენტებისა და კომპიუტერის შესწავლის ყველა მსურველისათვის.

რედაქტორი:
გრიგოლ კახიანი - ბათუმის შოთა რუსთაველის სახელმწიფო უნივერსიტეტის განათლებისა და მეცნიერებათა ფაკულტეტის კომპიუტერულ მეცნიერებათა დეპარტამენტის ასისტენტ პროფესორი

რეცენზენტები:
გ. სურგულაძე - ტექნიკურ მეცნიერებათა დოქტორი, საქართველოს პოლიტექნიკური უნივერსიტეტის ორგანიზაციული მართვის დეპარტამენტის სრული პროფესორი
ი. დიდმანიძე - ფიზიკა-მათემატიკის მეცნიერებათა დოქტორი, ბათუმის შოთა რუსთაველის სახელმწიფო უნივერსიტეტის კომპიუტერულ მეცნიერებათა დეპარტამენტის უფროსი, სრული პროფესორი

1. ზოგადი ცნობები კომპიუტერის შესახებ

კომპიუტერის არქიტექტურის ქვეშ ჩვეულებრივ გულისხმობენ კომპიუტერის ლოგიკურ აგებულებას, ანუ იმას თუ როგორ წარმოუდგენია ის პროგრამისტს. პირველად ტერმინი „კომპიუტერის არქიტექტურა“ (computer arckitecture) გამოყენებული იქნა ფირმა IBM-ის მიერ IBM-360 ოჯახის კომპიუტერების დამუშავების დროს იმ საშუალებათა აღსაწერად, რომლებითაც შეიძლება ისარგებლოს პროგრამისტმა მანქანური ბრძანებების დონეზე პროგრამის შედგენისას. ასეთ განმარტებას „ვიწროს“ ეძახიან. ის მოიცავს ბრძანებების ჩამონათვალსა და ფორმატს, მონაცემების წარმოდგენის ფორმებს, შეტანა/გამოტანის მექანიზმებს, მეხსიერების გადამისამართების ხერხებს და ა. შ. განხილვიდან ამოვარდნილია გამოთვლითი საშუალებების ფიზიკური აგების საკითხები: მოწყობილობათა შემადგენლობა, პროცესორის რეგისტრირების რაოდენობა, მეხსიერების მოცულობა, ნამდვილი რიცხვების დამუშავების სპეციალური ბლოკის არსებობა, ცენტრალური პროცესორის ტაქტური სიხშირე და ა.შ. საკითხთა ეს წრე მიღებულია განისაზღვროს ცნებით „ორგანიზება“ ან „სტრუქტურული ორგანიზება“.

არქიტექტურა (ვიწრო გაგებით) და ორგანიზება ეს კომპიუტერის აღწერის ორი მხარეა. ჩვენ ქვემოთ ვისარგებლეთ ტერმინით „არქიტექტურა“ მხოლოდ „ფართო“ გაგებით, რომელიც მოიცავს როგორც არქიტექტურას ვიწრო გაგებით, ისე კომპიუტერის ორგანიზებას.

1.1. კომპიუტერის შექმნის წინაისტორია

კომპიუტერი - მე-20 საუკუნის უდიდესი გამოგონებაა. გამოთვლითი ტექნიკა იქცა სამეცნიერო-ტექნიკური პროგრესის განვითარების ერთ-ერთ ძირითად საშუალებად. თანამედროვე კომპიუტერების შექმნას მრავალმა გენიალურმა მეცნიერმა ჩაუყარა საფუძველი.

1642 წელს 18 წლის ფრანგმა მათემატიკოსმა და ფიზიკოსმა ბლეზ პასკალმა (1623-1662) შექმნა კომპიუტერის პირველი მოდელი, რომელსაც ოთხი არითმეტიკული ოპერაციის შესრულება შეეძლო. 1645 წელს კომპიუტერის ამ მოდელმა (პასკალინა) დასრულებული სახე მიიღო.

1670 წელს დიდმა გერმანელმა მეცნიერმა გოლტფრიდ ვილჰელმ ლეიბნიცმა (1646-1716) გამოსცადა თავისი პირველი მთვლელი მანქანის აღწერა, რომელიც მექანიკურად ასრულებდა ოთხ არითმეტიკულ ოპერაციას. ამ კომპიუტერის საბოლოო ვარიანტი 1710 დასრულდა. ლეიბნიცმა პირველმა გამოიყენა ორობითი არითმეტიკა - თანამედროვე გამოთვლითი ტექნიკის საფუძველთა საფუძველი. მანვე შემოგვთავაზა ლოგიკის არითმეტიზაცია, მაგრამ ლოგიკის „ალგებრული ეტაპის“ ცენტრალური ფიგურა ინგლისელი მეცნიერი ჯორჯ ბული (1815-1864) იყო. მან ალგებრულ სახეში ჩამოაყალიბა აზროვნების კანონები და აჩვენა, რომ მისი ალგებრა (იწოდება ბულის ალგებრად) მსჯელობისათვის ოპერირებს მხოლოდ ორი ცნებით: ჭეშმარიტი და მცდარი. თვლის ორობითი სისტემისათვის ესაა 1 და 0, რაც პრაქტიკაში ადვილად რეალიზებადია რელეს (ელექტრო-მექანიკური მოწყობილობა) დახმარებით. შესაძლებელია რომ სწორედ ბულის შრომებს უნდა ვუმადლოდეთ რელეებზე აგებული კომპიუტერების შექმნას. რელეების დახმარებით ადვილად რეალიზებადია ძირითადი ლოგიკური სქემები: „და“, „ან“ და „არა“

გ. ლეიბნიცისა და დ. ბულის შრომებით საფუძველი ჩაეყარა თეორიულ ბაზას მაღალმწარმოებლური გამოთვლითი მოწყობილობების პრაქტიკული რეალიზაციისათვის.

პირველად ავტომატური გამომთვლელი მანქანის ფუნქციურ საშუალებათა შემადგენლობა და დანიშნულება განსაზღვრა 1834 წელს ინგლისელმა მათემატიკოსმა და ეკონომისტმა ჩარლზ ბებიჯმა (1792-1871) ანალიტიკური მანქანის თავის განუხორციებელ პროექტში. პროექტის მიხედვით ანალიტიკური მანქანა ოთხი შემადგენელი ნაწილისაგან შედგებოდა - რიცხვების დამმახსოვრებელი მოწყობილობისაგან

(ბებიჯის განმარტებით „საწყობი“); რიცხვებზე არითმეტიკული მოქმედებების შემსრულებელი მოწყობილობისაგან (ბებიჯის განმარტებით „ფაბრიკა“); საჭირო თანმიმდევრობით მანქანის ოპერაციების მმართველი მოწყობილობისაგან (მათ რიცხვში, რიცხვების ერთი ადგილიდან მეორეში გადატანისათვის); რიცხვების შემყვანი და გამომყვანი მოწყობილობისაგან. „ანალიტიკური მანქანისათვის“ პირველი პროგრამები (ორუცნობიან განტოლებათა სისტემის ამოხსნისა და ბერნულის რიცხვების გამოთვლისათვის) შეადგინა ჯონ ბაირონის ქალიშვილმა ადა ლავლეისმა.

ავტომატური კომპიუტერის პირველ შემქმნელად ითვლება გერმანელი მეცნიერი კარლ ცუზე. 1938 წელს მან დაამზადა Z1 მანქანის მოდელი, მომდევნო წელს-Z2-ის, ხოლო კიდევ ორი წლის შემდეგ ააგო პირველი მომქმედიკომპიუტერი პროგრამული მართვით (Z3). ეს იყო რელეებზე აგებული ორობითი კომპიუტერი. კ. ცუზემ 1945 წელს შექმნა ალგორითმული ენების პირველი ანალოგი - ენა Plankaikul („გეგმების გათვლა“).

1944 წელს ამერიკელმა მათემატიკოსმა ჰორვარდ აიკენმა ჰარვარდის უნივერსიტეტში შექმნა ავტომატური კომპიუტერი „მარკ-1“ პროგრამული მართვით რელეურ და მექანიკურ ელემენტებზე.

1945 წელს ამერიკელმა მეცნიერმა ჯონ ფონ ნეიმანმა დაამუშავა ელექტრონულ-გამომთვლელი მანქანის კონცეფცია EDVAC (Electronic Discrete Variable Computer) მეხსიერებაში პროგრამებისა და რიცხვების შეყვანით. თავად მანქანის დამზადება 1950 წელს დასრულდა.

1946 წელს ამერიკელმა ინჟინერმა დ. პ. ეკერტმა და ფიზიკოსმა დ. უ. მოუჩლიმ პელსილვანიის უნივერსიტეტში შექმნეს პირველი ელექტრონულ-გამომთვლელი მანქანა „ENIAC“ (Electronic Numerical integrator and Computer). ის შედგებოდა თითქმის 20000 ელექტრონული მილაკისა და 1500 რელესაგან, ერთი წამის განმავლობაში ასრულებდა 5000 შეკრების ოპერაციას, მოიხმარდა 150 კვტ სიმძლავრის ელექტროენერგიას, ეჭირა 9x15 $მ^2$ ფართი და იწონიდა 30 ტონას.

1949 წელს ინგლისში, კემბრიჯის უნივერსიტეტში, პროფესორ მორის უილკსის ხელმძღვანელობით შეიქმნა მსოფლიოში პირველი კომპიუტერი, რომელსაც შეეძლო პროგრამის დამახსოვრება.

1950 წელს აკადემიკოს ს. ა. ლებედევის ხელმძღვანელობით საბჭოთა კავშირში შეიქმნა კომპიუტერი МЭСМ (Малая электронная счетная машина). ეს მანქანა ექსპლუატაციაში 1951 წელს შევიდა.

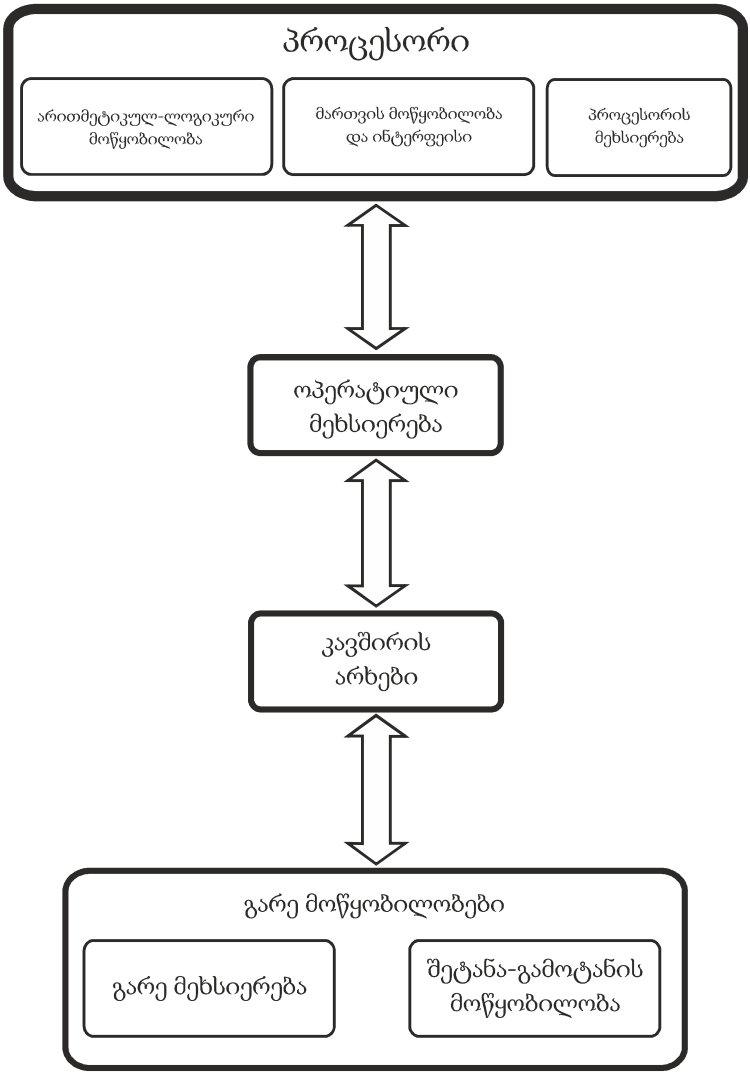
ჯონ ფონ ნეიმანის კონცეფციის მიხედვით კომპიუტერში განხორციელებულია დამახსოვრებული პროგრამისა და გამოთვლების პარალელური ორგანიზაციის პრინციპი. ის აგებულია შემდეგი ოთხი მოწყობილობის ბაზაზე:

- არითმეტიკულ-ლოგიკური მოწყობილობა;
- მართვის მოწყობილობა;
- დამმახსოვრებელი მოწყობილობა;
- შეყვანა-გამოყვანის მოწყობილობა.

კომპიუტერის მეხსიერება უნდა შედგებოდეს რამდენიმე გადანომრილი უჯრედისაგან. თითოეულ მათგანში შეიძლება იყოს დასამუშავებელი მონაცემები, ან პროგრამების ინსტრუქციები. ყველა უჯრედი თანაბრად მისაღწევია კომპიუტერის სხვა მოწყობილობებისათვის.

ზოგადი სახით კომპიუტერის მუშაობა შეიძლება ასე აღვწეროთ. თავდაპირველად, შეყვანა-გამოყვანის მოწყობილობის საშუალებით კომპიუტერის მეხსიერებაში შეიყვანება პროგრამა. მართვის მოწყობილობა მეხსიერების უჯრედებიდან კითხულობს პროგრამის პირველ ინსტრუქციებს (ბრძანებებს) და ორგანიზებას უწევს მის შესრულებას. როგორც წესი, მართვის მოწყობილობა ერთი ბრძანების შესრულების შემდეგ იწყებს იმ ბრძანების შესრულებას, რომელიც მეხსიერების იმ უჯრედშია მოთავსებული, რომელიც მდებარეობს შესრულებული ბრძანების შემდეგ. მაგრამ ასეთი წესი შეიძლება შეწყდეს მართვის გადაცემის ბრძანებით. ეს კი საშუალებას იძლევა მეხსიერების უჯრედებში შენახული ბრძანებების თანმიმდევრობა რამდენჯერმე გამოვიყენოთ პროგრამაში (ე. ი. მეხსიერების უჯრედებში ერთჯერ შეყვანილი ბრძა-

ნებების თანმიმდევრობა, ამოცანის პირობის მიხედვით, შეიძლება რამდენჯერმე გამოვიყენოთ (ციკლის ორგანიზებით)).



ნახ. 1.1. კომპიუტერის გამარტივებული სტრუქტურული სქემა

ამრიგად, მართვის მოწყობილობა ასრულებს პროგრამის ინსტრუქციებს ავტომატურად, ანუ ადამიანის ჩარევის გარეშე. მას შეუძლია გაცვალოს ინფორმაცია კომპიუტერის ოპერატიულ მეხსიერებასთან და გარე მოწყობილობასთან. ვინაიდან როგორც წესი, გარე მოწყობილობები მუშაობს გაცილებით ნელა, ვიდრე კომპიუტერის სხვა შემადგენლები, ამიტომ მართვის მოწყობილობას, შეტანა-გამოტანის ოპერაციის დამთავრებამდე, შეუძლია შეწყვიტოს პროგრამების შესრულების მსვლელობა. პროგრამის მსვლელობის ყველა შედეგი უნდა გამოვიტანოთ შეტანა-გამოტანის მოწყობილობით გარე მოწყობილობაზე. ამის შემდეგ კომპიუტერი გადადის მოლოდინის რეჟიმში.

აქვე შევნიშნავთ, რომ თანამედროვე კომპიუტერების აგებულება განსხვავდება ზემოთ აღწერილი სქემისაგან. კერძოდ, არითმეტიკულ-ლოგიკური მოწყობილობა და მართვის მოწყობილობა, როგორც წესი, გაერთიანებულია ერთ მოწყობილობაში - ცენტრალურ პროცესორში. გარდა ამისა, პროგრამის შესრულების პროცესი შეიძლება შეწყდეს გადაუდებელი მოქმედების გამო, რომელიც დაკავშირებულია კომპიუტერის გარე მოწყობილობიდან წყვეტის სიგნალის მიღებაზე. ბევრი სრაფმოქმედი კომპიუტერი პარალელურად ახორციელებს რამდენიმე პროცესორიდან მონაცემების დამუშავებას. მიუხედავად ამისა, თანამედროვე კომპიუტერების აგებულება ძირითადად შეესაბამება ჯონ ფონ ნეიმანის კონცეფციას.

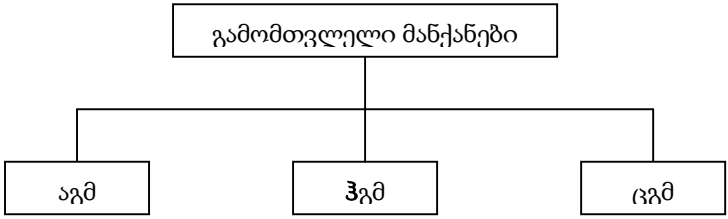
1.2. კომპიუტერების კლასიფიკაცია

კომპიუტერი, ელექტრონული გამომთვლელი მანქანა (ეგმ) - ეს ტექნიკურ საშუალებათა კომპლექსია, რომლის დანიშნულებაა გამოთვლითი და ინფორმაციული ამოცანების გადაწყვეტის პროცესში ინფორმაციის ავტომატური დამუშავება. მათი კლასიფიკაცია შეიძლება

სხვადასხვა ნიშნის მიხედვით, კერძოდ: მუშაობის პრინციპის, დანიშნულების, გამოთვლითი პროცესის ორგანიზაციის ხერხების, ზომების და გამოთვლითი სიმძლავრის, ფუნქციური შესაძლებლობების და სხვა.

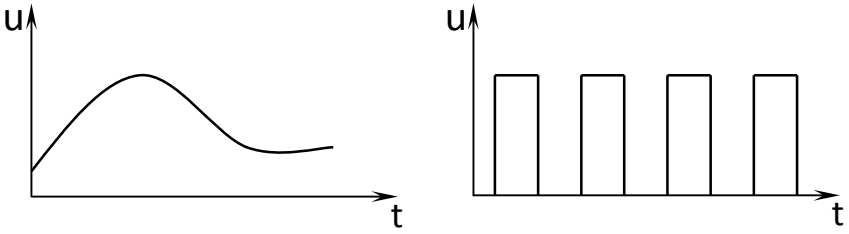
ეგმ-ის კლასიფიკაცია მოქმედების პრინციპის მიხედვით

მოქმედების პრინციპის მიხედვით გამომთვლელი მანქანები სამ დიდ კლასად იყოფა (ნახ. 1.2): ანალოგური (აგმ), ციფრული (ცგმ) და ჰიბრიდული (ჰგმ).



ნახ 1.2. გამომთვლელი მანქანების კლასიფიკაცია მოქმედების პრინციპის მიხედვით

გამომთვლელი მანქანების ამ სამ კლასად დაყოფის კრიტერიუმს წარმოადგენს იმ ინფორმაციის ფორმა, რომელთანაც ისინი მუშაობენ (ნახ. 1.3.).



ა) ბ)

ნახ. 1.3. გამომთვლელ მანქანებში ინფორმაციის წარმოდგენის ორი ფორმა: ა) - ანალოგური; ბ) - ციფრული, იმპულსური.

ციფრული გამომთვლელი მანქანები (ცგმ) - დისკრეტული მოქმედების გამომთვლელი მანქანები, მუშაობენ დისკრეტულ, უფრო ზუსტად, ციფრულ ფორმაში წარმოდგენილ ინფორმაციასთან.

ანალოგური გამომთვლელი მანქანები (აგმ) - უწყვეტი მოქმედების გამომთვლელი მანქანები, მუშაობენ უწყვეტი (ანალოგური) ფორმით წარმოდგენილ ინფორმაციასთან, ე. ი. რაიმე ფიზიკური სიდიდის (ყველაზე ხშირად ელექტრული ძაბვის) მნიშვნელობების უწყვეტ რიგთან.

ჰიბრიდული გამომთვლელი მანქანები (ჰგმ) - კომბინირებული მოქმედების გამომთვლელი მანქანები, მუშაობემ ციფრული და ანალოგური ფორმით წარმოდგენილ ინფორმაციასთან; მას გააჩნია როგორც აგმ-ის, ისე ცგმ-ის უპირატესობები. ჰგმ-ის გამოყენება მიზანშეწონილია რთული, მაღალი სისწრაფით მომუშავე ტექნიკური კომპლექსების მართვის ამოცანების გადასაწყვეტად.

ანალოგური გამომთვლელი მანქანები ექსპლუატაციაში მარტივი და მოხერხებულებია; ამოცანათა დაპროგრამება მათზე, როგორც წესი, არაშრომატევადია; ამოცანათა ამოხსნის სისწრაფე რაგინდ მაღალია (მეტია ვიდრე ცგმ-ზე), მაგრამ ამოხსნის სიზუსტე დაბალია (ფარდობითი ცდომილება 2-5%-ს შეადგენს). აგმ-ზე ყველაზე ეფექტურია ისეთი მათემატიკური ამოცანების ამოხსნა, რომლებიც დიფერენციალურ განტოლებებს შეიცავენ და არ თხოულობენ რთულ ლოგიკას.

ყველაზე ფართო გავრცელება მოიპოვეს ცგმ-ა ინფორმაციის დისკრეტული წარმოდგენით - ელექტრონულმა ციფრულმა გამომთვლელმა მანქანებმა. მათ, ჩვეულებრივ, ელექტრონულ გამომთვლელ მანქანებს, ან უბრალოდ, კომპიუტერებს უწოდებენ.

ეგმ-ის კლასიფიკაცია შექმნის ეტაპების მიხედვით

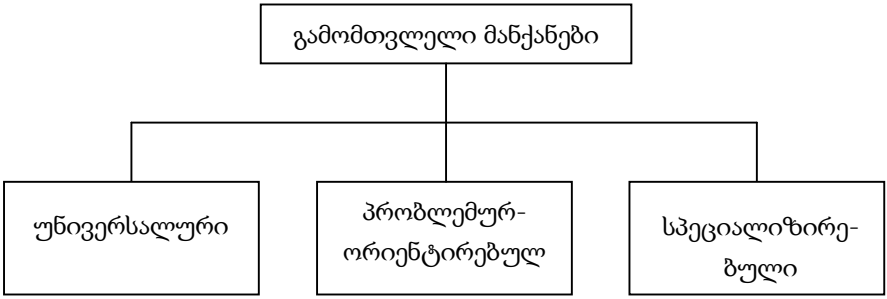
შექმნის ეტაპებისა და ელემენტთა გამოყენებული ბაზის მიხედვით ეგმ-ები პირობითად შემდეგ თაობებად იყოფიან:

- I თაობა, 50-იანი წლები: ელექტრონულ-ვაკუუმურ მილაკებზე აგებული ეგმ-ები;

- II თაობა, 60-იანი წლები: დისკრეტულ ნახევარგამტარულ ელემენტებზე (ტრანზისტორებზე) აგებული ეგმ-ები;
- III თაობა, 70-იანი წლები: მცირე და საშუალო დონის ინტეგრაციის ნახევარგამტარულ ინტეგრალურ სქემებზე აგებული ეგმ-ები;
- IV თაობა, 80-იანი წლები: დიდ ინტეგრალურ სქემებზე - მიკროპროცესორებზე აგებული ეგმ-ები;
- V თაობა, 90-იანი წლებიდან დღემდე: პარალელურად მომუშავე ასობით მიკროპროცესორებზე აგებული ეგმ-ები, რომლებიც საშუალებას იძლევიან აიგოს ცოდნის ბაზების ეფექტური სისტემები; აგებული ეგმ-ები ზერთულ მიკროპროცესორებზე პარალელურ-ვექტორული სტრუქტურით, რომლებიც ერთდროულად ასრულებენ პროგრამის ათობით მიმდევრობით ბრძანებას.

ეგმ-ის კლასიფიკაცია დანიშნულების მიხედვით

დანიშნულების მიხედვით ეგმ-ები იყოფიან სამ ჯგუფად -უნივერსალურ (საერთო დანიშნულების), პრობლემურ-ორიენტირებულ და სპეციალიზირებულებად (ნახ. 1.4.).



ნახ. 1.4. ეგმ-ის კლასიფიკაცია დანიშნულების მიხედვით

უნივერსალური ეგმ-ების დანიშნულებაა სხვადასხვა სახის საინჟინრო-ტექნიკური ამოცანების (ეკონომიკური, მათემატიკური, ინფორმაციული და სხვა) ამოხსნა. ისინი ფართოდ გამოიყენება კოლექტიური მოხმარების გამოთვლით ცენტრებში და სხვა მძლავრ გამოთვლით კომპლექსებში.

უნივერსალური ეგმ-ების დამახასიათებელ ნიშნებს წარმოადგენს:

- მაღალი მწარმოებლურობა (წარმადობა);
- დასამუშავებელი მონაცემების ფორმების მრავალფეროვნება: (ორობითი, ათობითი, სიმბოლური) მათი ცვლადების დიდი დიაპაზონისა და წარმოდგენის დიდი სიზუსტის დროს;
- შესასრულებელ ოპერაციათა (არითმეტიკულ, ლოგიკურ, სპეციალურ) დიდი ნუსხა;
- ოპერატიული მეხსიერების დიდი მოცულობა;
- ინფორმაციის შეყვანა-გამოყვანის სისტემის განვითარებული ორგანიზაცია, რომელიც უზრუნველყოფს მრავალფეროვანი გარე მოწყობილობის მიერთებას.

პრობლემურ-ორიენტირებული ეგმ-ები ემსახურებიან უფრო შეზღუდული წრის ამოცანების ამოხსნას, როგორც წესი, ესაა ტექნოლოგიური პროცესების მართვა; შედარებით მცირე მოცულობის მონაცემების რეგისტრაცია, დაგროვება და დამუშავება; შედარებით არართული ალგორითმებით გათვლების შესრულება; უნივერსალურ ეგმ-ებთან შედარებით მათ შეზღუდული აპარატურული და პროგრამული რესურსები აქვთ.

პრობლემურ-ორიენტირებული ეგმ-ებს მიეკუთვნება მრავალფეროვანი მმართველი გამომთვლელი კომპლექსები.

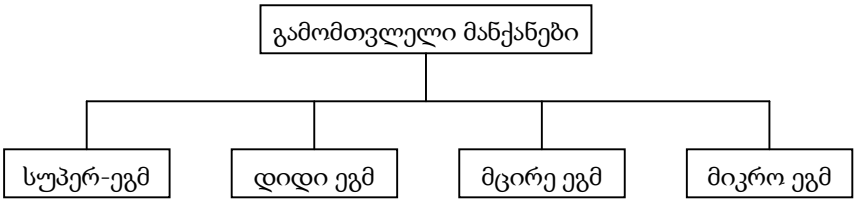
სპეციალიზირებული ეგმ-ები ემსახურებიან ვიწრო წრის ამოცანების ამოხსნას, ან მკაცრად განსაზღვრული ფუნქციების ჯგუფის რეალიზაციისას. ეგმ-ის ასეთი ვიწრო ორიენტაცია საშუალებას იძლევა სპეციალიზირებულ იქნას მისი სტრუქტურა, მნიშვნელოვნად შემცირ-

დეს მათი სირთულე და ღირებულება მუშაობის მაღალი მწარმოებლურობისა და საიმედობის პირობებში.

სპეციალიზირებულ ეგმ-ებს შეიძლება, მაგალითად, მივაკუთვნოთ სპეციალური დანიშნულების პროგრამირებადი მიკროპროცესორები, ადაპტორები, რომლებიც ასრულებენ მართვის ლოგიკურ ფუნქციებს ცალკეული არართული ტექნიკური მოწყობილობებისათვის, აგრეგატებისა და პროცესებისათვის; გამოთვლითი სისტემების კვანძების მუშაობის შეთანხმების მოწყობილობები.

ეგმ-ის კლასიფიკაცია ზომებისა და ფუნქციური შესაძლებლობების მიხედვით

ზომებისა და ფუნქციური შესაძლებლობების მიხედვით ეგმ-ები შეიძლება დაიყოს ზედიდ (სუპერ-ეგმ), დიდ, მცირე, ზემცირე (მიკრო-ეგმ) მანქანებად (ნახ. 1.5.).



ნახ. 1.5. ეგმ-ის კლასიფიკაცია ზომებისა და გამოთვლითი სიმძლავრის მიხედვით

ეგმ-ის ფუნქციურ შესაძლებლობებს განაპირობებს უმნიშვნელოვანესი ტექნიკურ-ექსპლოატაციური მახასიათებლები:

- სწრაფქმედება, მანქანის მიერ დროის ერთეულში შესრულებულ ოპერაციათა გასაშუალებული სიდიდე;
- რიცხვის წარმოდგენის ფორმა და თანრიგთა რაოდენობა;
- მეხსიერების ყველა მოწყობილობის ტიპი, მოცულობა და სწრაფქმედება;
- ეგმ-ის კვანძების შეერთებისა და კავშირის მოწყობილობათა ტიპები და გამტარუნარიანობა;

- ეგმ-ის შესაძლებლობა ერთდროულად იმუშაოს რამდენიმე მომხმარებელთან და ერთდროულად შეასრულოს რამდენიმე პროგრამა;
- მანქანაში გამოყენებულ ოპერაციულ სისტემათა ტიპები და ტექნიკურ-ექსპლოატაციური მახასიათებლები;
- პროგრამული უზრუნველყოფის არსებობა და ფუნქციური შესაძლებლობები;
- სხვა ტიპის ეგმ-ებისათვის დაწერილ პროგრამათა შესრულების შესაძლებლობა (სხვა ტიპის ეგმ-ებთან პროგრამული თავსებადობა - მხარდაჭერა);
- მანქანის ბრძანებათა სისტემა და სტრუქტურა;
- კავშირის ხაზებთან და გამომთვლელ ქსელებთან მიერთების შესაძლებლობა;
- ეგმ-ის ექსპლოატაციური საიმედობა;
- ეგმ-ის დროში სასარგებლო გამოყენების კოეფიციენტი, რომელიც განისაზღვრება სასარგებლო მუშაობის დროის პროფილაქტიკის დროსთან ფარდობით.

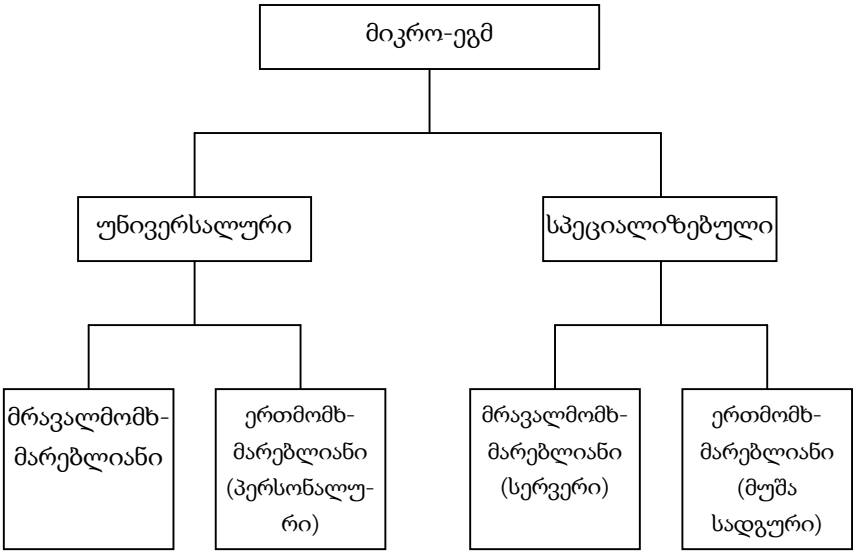
ისტორიულად ჯერ დიდი ეგმ-ები შეიქმნა, რომელთა ელემენტთა ბაზამ გაიარა გზა ელექტრონულ-ვაკუუმური მილაკებიდან ზემაღალი ხარისხის ინტეგრაციის ინტეგრალურ სქემებამდე.

დიდი ეგმ-ების წარმადობა არასაკმარისი აღმოჩნდა რიგი ამოცანებისათვის: მეტეომდგომარეობის პროგნოზირება, რთული სამხედრო კომპლექსების მართვა, ეკოლოგიური სისტემის მოდელირება და სხვა. ეს გახდა წინაპირობა სუპერ-ეგმ-ის - ყველაზე მძლავრი გამოთვლითი სისტემის დასამუშავებლად. ეს სისტემები დღესაც ინტენსიურად ვითარდება.

70-იან წლებში მცირე ეგმ-ის შექმნა განპირობებული იყო, ერთის მხრივ, პროგრესით ელექტრონული ელემენტების ბაზის სფეროში, ხოლო მეორეს მხრივ - დიდი ეგმ-ის რესურსების სიჭარბით რიგი დამატებებისათვის. მცირე ეგმ-ებს ყველაზე ხშირად იყენებენ ტექნოლოგიური პროცესების მართვისათვის. ისინი მნიშვნელოვნად იაფიანებია.

ელემენტთა ბაზის წარმოებაში და არქიტექტურულ გადაწყვეტებში შემდგომმა წარმატებებმა მიგვიყვანა სუპერმინი ეგმ-ის შექმნასთან. ეს გამომთვლელი მანქანა არქიტექტურით, ზომებითა და ღირებულებით მიეკუთვნება მცირე ეგმ-თა კლასს, ხოლო წარმადობით ედარება დიდ ეგმ-ს.

1969 წელს მიკროპროცესორის გამოგონებამ გამოიწვია 70-იან წლებში მიკრო ეგმ-ის (ნახ.1.6.) შექმნა. თავდაპირველად მიკროპროცესორის არსებობა ითვლებოდა მიკრო ეგმ-ის განმსაზღვრელ ნიშნად. ახლა მიკროპროცესორები გამოიყენება ყველა კლასის კომპიუტერებში.



ნახ. 1.6. მიკრო-ეგმ-ის კლასიფიკაცია

მრავალმომხმარებლიანი მიკრო-ეგმ - ეს მძლავრი მიკრო-ეგმ-ია, რომელიც აღჭურვილია რამდენიმე ვიდეოტერმინალით და ფუნქციონირებს დროის გაყოფის რეჟიმში, რაც იძლევა მასზე ერთდროულად რამოდენიმე მომხმარებლის ეფექტური მუშაობის საშუალებას.

პერსონალური კომპიუტერი (პკ) - ერთმომხმარებლიანი მიკრო-ეგმ-ია, რომელიც აკმაყოფილებს საერთო მიღწევის მოთხოვნას და გამოყენების უნივერსალურობას.

მუშა სადგურები (Work Station) წარმოადგენს ერთმომხმარებლიან მძლავრ მიკრო-ეგმ-ს, რომელიც სპეციალიზირებულია განსაზღვრული სახის სამუშაოს (გრაფიკული, საინჟინრო, საგამომცემლო და ა. შ.) შესასრულებლად.

სერვერები (Server) - ქსელებში მომუშავე მრავალმომხმარებლიანი მძლავრი მიკრო-ეგმ-ია, გამოყოფილი ქსელის სადგურიდან მიღებული მოთხოვნების დასამუშავებლად.

ზემოთ მოყვანილი კლასიფიკაცია პირობითია, ვინაიდან მძლავრი თანამედროვე პკ, აღჭურვილი პრობლემურ-ორიენტირებული პროგრამებითა და აპარატურული უზრუნველყოფით, შეიძლება გამოვიყენოთ როგორც სრულფასოვანი მუშა სადგური, ისე მრავალმომხმარებლიანი მიკრო-ეგმ-ი, აგრეთვე როგორც კარგი სერვერი, რომელიც თავისი მახასიათებლებით თითქმის არ ჩამოუვარდება მცირე ეგმ-ს.

1.3. მეხსიერებაში შენახული პროგრამის მქონე გამომთვლელი მანქანის კონცეფცია

გამომთვლელი მანქანა, რომელშიც პროგრამის განსაზღვრული სახით კოდირებული ბრძანებები ინახებიან მეხსიერებაში, ცნობილია სახელწოდებით „გამომთვლელი მანქანა მეხსიერებაში შენახული პროგრამით“. იდეა ეკუთვნით ENIAC -ის შემქმნელებს - ეკერტს, მოუჩლისა და ფონ ნეიმანს. ჯერ კიდევ ENIAC-ზე სამუშაოების დასრულებამდე მათ დაიწყეს მუშაობა ახალ პროექტზე - EDVAC, რომლის მთავარი თა-

ვისებურება გახდა მეხსიერებაში შენახული პროგრამის კონცეფცია. ამ კონცეფციამ მრავალი წლით განსაზღვრა გამომთვლელი მანქანების მომდევნო თაობების აგების საბაზისო პრინციპები. ავტორობის შესახებ რამდენიმე ვერსია არსებობს, მაგრამ ვინაიდან დასრულებული სახით ის პირველად 1945 წელს ფონ ნეიმანის სტატიაში იყო აღწერილი, ამიტომაც ფონ ნეიმანის სახელი ფიგურირებს მსგავსი არქიტექტურების მქონე მანქანათა აღნიშვნებში. ეს მანქანები დღეისათვის გამომთვლელი მანქანებისა და სისტემების უდიდეს უმრავლესობას წარმოადგენენ.

გამომთვლელი მანქანების აგების ფონ ნეიმანისეული კონცეფციის არსი შეიძლება დაყვანილ იქნეს ოთხ პრინციპამდე:

- ორობითი კოდირება;
- პროგრამული მართვა;
- მეხსიერების ერთგვაროვნება;
- მისამართების გამოყენება.

განვიხილოთ თითოეული ამ პრინციპთაგანი უფრო დაწვრილებით.

ორობითი კოდირების პრინციპი

ამ პრინციპის თანახმად, მთელი ინფორმაცია, როგორც მონაცემები, ისე ბრძანებებიც, კოდირება ორობითი ციფრებით 0 და 1. ინფორმაციის თითოეული ტიპი წარმოიდგინება ორობითი თანმიმდევრობით და აქვს თავისი ფორმატი. ფორმატში ბიტების თანმიმდევრობა, რომელსაც განსაზღვრული აზრი აქვს, იწოდება ველად. რიცხვით ინფორმაციაში ჩვეულებრივ ყოფენ ნიშნის ველს და ნიშნად თანრიგთა ველს. ბრძანების ფორმატში შეიძლება გამოვყოთ ორი ველი (ნახ. 1.7.): ოპერაციის ველი და მისამართების ველი (სამისამართო ნაწილი).

ნახ. 1.7. ბრძანების სტრუქტურა

ოპერაციის ველი წარმოადგენს მითითებას იმაზე, თუ რომელი ოპერაცია უნდა შესრულდეს და მოიცემა r - თანრიგა ოროობითი კომბინაციით.

სამისამართო ნაწილი და მისი მისამართების შემქმნელი რიცხვი დამოკიდებულია ბრძანების ტიპზე: მონაცემთა გარდაქმნის ბრძანებებში სამისამართო ნაწილი შეიცავს დასამუშავებელი ობიექტების (ოპერანდების) და შედეგების მისამართებს; გამოთვლების რიგის ცვლილების ბრძანებებში - შეტანა/გამოტანის მოწყობილობის ნომერს. სამისამართო ნაწილი აგრეთვე წარმოდგენილია ოროობითი თანმიმდევრობით, რომლის სიგრძეც p - თი ავღნიშნოთ, ამრიგად, ბრძანებას გამომთვლელ მანქანაში აქვს (r+p) - თანრიგა ოროობითი კომბინაციის სახე.

პროგრამული მართვის პრინციპი

ამოცანის ამოხსნის ალგორითმით გათვალისწინებული ყველა გამოთვლა უნდა იყოს წარმოდგენილი პროგრამის სახით, რომელიც შედგება მმართველ სიტყვათა - ბ რ ძ ა ნ ე ბ ე ბ ი ს თანმიმდევრობისაგან. თითოეული ბრძანება ახორციელებს მიმართვის გამომთვლელ მანქანაში რეალიზებული ოპერაციების ნაკრებიდან ერთ-ერთზე. პროგრამის ბრძანებები ინახებიან გამომთვლელი მანქანის მეხსიერების მიმდევრობით უჯრედებში და სრულდებიან ბ უ ნ ე ბ რ ი ვ ი თ ა ნ მ ი მ დ ე ვ რ ო ბ ით, ანუ პროგრამაში მათი მდებარეობის რიგით. საჭიროების შემთხვევაში, სპეციალური ბრძანებით შეიძლება ამ თანმიმდევრობის შეცვლა. პროგრამის ბრძანებების შესრულების რიგის ცვლილების შესახებ გადაწყვეტილება მიიღება ჩატარებული გამოთვლების შედეგების ანალიზის საფუძველზე ან უპირობოდ.

მეხსიერების ერთგვაროვნების პრინციპი

ბრძანებები და მონაცემები ინახებიან ერთი და იგივე მეხსიერებაში და გარეგნულად მეხსიერებაში არაფრით განსხვავდებიან. მათი გარჩევა შეიძლება მხოლოდ გამოყენების ხერხით. ეს საშუალებას იძლევა

შევასრულოთ ბრძანებებზე იგივე ოპერაციები, რაც რიცხვებზე და გამომდინარე, გვიხსნის რიგ შესაძლებლობას. ასე, ბრძანების სამისამართო ნაწილის ციკლური ცვლილებით შეიძლება ვუზრუნველყოთ მონაცემთა მასივის მიმდევრობით ელემენტებზე მიმართვა. ასეთ ხერხს ბრძანების მოდიფიკაცია ჰქვია და თანამედროვე პროგრამირების პოზიციებიდან დაწუნებულია. უფრო სასარგებლოა ერთგვაროვნების პრინციპის მეორე შედეგი, როცა ერთი პროგრამის ბრძანებები შეიძლება მიღებულ იქნენ როგორც სხვა პროგრამის შესრულების შედეგი. ეს პრინციპი საფუძვლად უდევს ტ რ ა ნ ს ლ ი ა ც ი ა ს - პროგრამის ტექსტის მაღალი დონის ენიდან კონკრეტული გამომთვლელი მანქანის ენაზე გადაყვანას.

ფონ ნეიმანის სტატიაში აღწერილი გამომთვლელი მანქანის კონცეფცია გულისხმობს ბრძანებებისა და მონაცემების შენახვისათვის ერთიანი მეხსიერების გამოყენებას. ასეთი მიდგომა საფუძვლად დაედო პრინსტონის უნივერსიტეტში შექმნილ გამომთვლელ მანქანებს, რის გამოც მან მიიღო „პ რ ი ნ ს ტ ო ნ ი ს ა რ ქ ი ტ ე ქ ტ უ რ ი ს“ სახელი. პრაქტიკულად იმავდროულად ჰარვარდის უნივერსიტეტში შეთავაზებულ იქნა მეორე მოდელი, რომელშიც გამომთვლელ მანქანას გააჩნდა ცალკე მეხსიერება ბრძანებებისათვის და ცალკე მეხსიერება მონაცემებისათვის. არქიტექტურის ამ სახეს „ჰ ა რ ვ ა რ დ ი ს ა რ ქ ი ტ ე ქ ტ უ რ ა ს“ - ეძახიან. დიდი ხნის განმავლობაში უპირატესობა ენიჭებოდა და ახლაც ენიჭება პრინსტონის არქიტექტურას, თუმცა ის იწვევს „პროცესორ - მეხსიერების“ ტრაქტის გამტარუნარიანობის პრობლემებს (ტიპიური ფონ ნეიმანისეული მონაცემების ტრაქტი შედგება არითმეტიკულ-ლოგიკური მოწყობილობის, ჩვეულებრივ, 8-დან 32-მდე რეგისტრისა და რამდენიმე კომუნიკაციური სალტისაგან. პროცესორის ტრაქტის სტრუქტურა, არქიტექტურის თავისებურებები დამოკიდებულია ბრძანებათა სისტემის სტრუქტურისაგან). ბოლო დროს კეშ-მეხსიერების ფართო გამოყენებასთან დაკავშირებით გამომთვლელი მანქანის დამმუშავებლები სულ უფრო და უფრო ხშირად მიმართავენ ჰარვარდის არქიტექტურას.

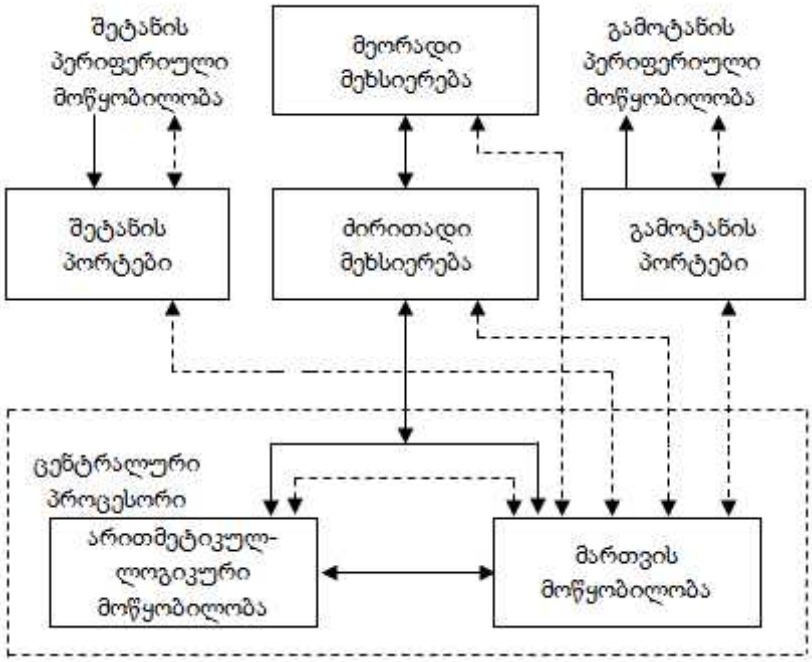
დამისამართების პრინციპი

სტრუქტურულად ძირითადი მეხსიერება შედგება გადანომრილი უჯრედებისაგან, ამასთან პროცესორისათვის ნებისმიერ მომენტში მისაწვდომია ნებისმიერი უჯრედი. ბრძანებებისა და მონაცემების ორობითი კოდები იყოფიან ინფორმაციის ერთეულებად, რომლებსაც სიტყვები ჰქვიათ და ინახებიან მეხსიერების უჯრედებში, ხოლო მათთან მისაღწევად გამოიყენებიან შესაბამისი უჯრედების ნომრები - მისამართები.

ფონ ნეიმანისეული არქიტექტურა

ფონ ნეიმანის სტატიაში განსაზღვრულია გამომთვლელი მანქანების ძირითადი მოწყობილობები, რომელთა დახმარებითაც განხორციელებულ უნდა იქნან ზემოთ ჩამოთვლილი პრინციპები. თანამედროვე გამომთვლელ მანქანათა უმრავლესობა თავისი სტრუქტურით პასუხობს პროგრამული მართვის პრინციპს. ტიპიური ფონ ნეიმანისეული გამომთვლელი მანქანა (ნახ. 1.8.) შეიცავს: მეხსიერებას, არითმეტიკულ-ლოგიკურ მოწყობილობას, მართვის მოწყობილობას და შეტანა /გამოტანის მოწყობილობას.

ნებისმიერ გამომთვლელ მანქანაში არის პროგრამების და ამ პროგრამებისათვის მონაცემების შეტანის საშუალება. გამომთვლელ მანქანას ინფორმაცია ეწოდება მასზე მიერთებული შეტანის პერიფერიული მოწყობილობებიდან. გამოთვლის შედეგები გამოდიან გამოტანის პერიფერიული მოწყობილობებიდან. გამომთვლელი მანქანისა და პერიფერიული მოწყობილობების კავშირსა და ურთიერთქმედებას უზრუნველყოფენ შეტანის პორტები და გამოტანის პორტები. ტერმინით პორტი აღნიშნავენ გამომთვლელ მანქანასთან პერიფერიული მოწყობილობის მიერთებისა და მართვის აპარატურას. შეტანისა და გამოტანის პორტების ერთობლიობას უწოდებენ შეტანა/გამოტანის მოწყობილობას ან გამომთვლელი მანქანის შეტანა/გამოტანის მოდულს.



ნახ. 1.8. ფონ ნეიმანისეული გამომთვლელი მანქანის სტრუქტურა

შეტანილი ბრძანებები და მონაცემები ჯერ ინახება ძირითად მეხსიერებაში, ხოლო საჭიროების შემთხვევაში გადადის მეორად მეხსიერებაში, ხანგრძლივი შენახვის მიზნით. პროგრამა რომ სრულდებოდეს, საჭიროა ბრძანებები და მონაცემები მოთავსებული იყვნენ ისეთნაირად ორგანიზებულ ძირითად მეხსიერებაში, რომ თითოეული ორობითი სიტყვა ინახებოდეს მისამართით ინდენტიფიცირებულ დამოუკიდებელ უჯრედში, ამასთან მეხსიერების მეზობელ უჯრედებს უნდა ჰქონდეს რიგით მომდევნო მისამართები. ძირითადი მეხსიერების დამახსოვრების მოწყობილობის ნებისმიერ უჯრედთან მიღწევა შეიძლება ხორციელდებოდეს ნებისმიერი თანმიმდევრობით. ასეთი მეხსიერება ცნობილია, როგორც მეხსიერება ნებისმიერი წვდომით. თანამედროვე გამომთვლელი მანქანების ძირითადი მეხ-

სიერება ძირითადად შედგება ნახევარგამტარული ოპერატიული დამახსოვრების მოწყობილობებისაგან, რომლებიც უზრუნველყოფენ ინფორმაციის როგორც წაკითხვას, ისე ჩაწერას. ასეთი მეხსიერების მოწყობილობები ენერგოდამოკიდებულნი არიან - ელექტრული კვების გათიშვის შემთხვევაში მათში ჩაწერილი ინფორმაცია იკარგება. თუ საჭიროა, რომ ძირითადი მეხსიერების ნაწილი ენერგოდამოუკიდებელი იყოს, მაშინ ძირითად მეხსიერებაში რთავენ მუდმივი მეხსიერების მოწყობილობებს, რომლებიც აგრეთვე უზრუნველყოფენ ნებისმიერ მიღწევას. მუდმივი მეხსიერების მოწყობილობაში შენახული ინფორმაცია შეიძლება მხოლოდ წავიკითხოთ.

ძირითადი მეხსიერების უჯრედის ზომა ჩვეულებრივ რვა ორობითი თანრიგის - ბაიტის ტოლი აიღება. დიდი რიცხვების შესანახად იყენებენ 2, 4 და 8 ბაიტს, განაწილებულებს თანმიმდევრობითი მისამართების მქონე უჯრედებში. ასეთ შემთხვევაში რიცხვის მისამართად ხშირად იღებენ მისი უმცროსი ბაიტის მისამართს. ასე 32- თანრიგიანი რიცხვის შენახვისას 200, 201, 202, 203 მისამართების მქონე უჯრედებში რიცხვის მისამართი იქნება 200. ასეთ წესს უწოდებენ დამისამართებას უმცროსი ბაიტით ან „მახვილბოლოვან“ მეთოდს. შესაძლებელია საწინააღმდეგო მიდგომაც - უმცირეს მისამართზე თავსდება უფროსი ბაიტი. ეს ხერხი ცნობილია როგორც დამისამართება უფროსი ბაიტით ან „ბლაგვბოლოვანი“ მეთოდი. პრინციპში ბაიტთა ჩაწერის რიგის არჩევა მნიშვნელოვანია მხოლოდ მონაცემების გადაგზავნისას გამომთვლელ მანქანებს შორის მათი მისამართების სხვადასხვა ფორმების დროს ან რიცხვის ცალკეულ ბაიტებთან მანიპულირებისას. გამომთვლელი მანქანების უმრავლესობაში გათვალისწინებულია სპეციალური ინსტრუქციები დამისამართების ერთი ხერხიდან მეორეში გადასასვლელად.

დიდი პროგრამებისა და მონაცემთა მასივების ხანგრძლივი შენახვისათვის გამომთვლელ მანქანაში ჩვეულებრივ არის დამატებითი მეხსიერება, ე. წ. მეორადი მეხსიერება. მეორადი მეხსიერება ენერგოდამოუკიდებელია და უფრო ხშირად ინახება სპეციალური პროგრამულად მხარდაჭერილი ობიექტების - ფაილების სახით.

მ ა რ თ ვ ი ს მ ო წ ყ ო ბ ი ლ ო ბ ა - გამომთვლელი მანქანის უმნიშვნელოვანესი ნაწილია. ის უზრუნველყოფს პროგრამების ავტომატურ შესრულებას (მართვის ფუნქციათა რეალიზაციის გზით) და უზრუნველყოფს გამომთვლელი მანქანის, როგორც ერთიანი სისტემის, ფუნქციონირებას. მართვის მოწყობილობის ფუნქციების გასამართავად საჭიროა გამომთვლელი მანქანა განვიხილოთ როგორც იმ ელემენტთა ერთობლიობა, რომელთა შორისაც ხდება ინფორმაციის გადაგზავნა და შესაძლებელია განხორციელდეს ამ ინფორმაციის განსაზღვრული სახის დამუშავება. ინფორმაციის გადაგზავნა გამომთვლელი მანქანის ნებისმიერ ელემენტს შორის ინიცირდება თავისი მ ა რ თ ვ ი ს ს ი გ ნ ა ლ ი თ, ანუ გამოთვლითი პროცესის მართვა დადის საჭირო მართვის სიგნალების ნაკრების გაცემით საჭირო დროით თანმიმდევრობაში. მართვის სიგნალთა კავშირები ნაჩვენებია ნახ.1.8-ზე წყვეტილი ხაზებით. მართვის მოწყობილობის ძირითად ფუნქციას წარმოადგენს იმ მართვის სიგნალთა ფორმირება, რომლებიც პასუხს აგებენ მეხსიერებიდან პროგრამით მითითებული რიგით ბრძანებათა ამოღებაზე და ამ ბრძანებების შემდგომი გამოყენება. გარდა ამისა, მართვის მოწყობილობა ახორციელებს მართვის სიგნალებს ფორმირებას გამოთვლითი მანქანის შიგა და გარე მოწყობილობათა სინქრონიზაციისა და კოორდინაციისათვის.

გამომთვლელი მანქანის კიდევ ერთ განუყოფელ ნაწილს წარმოადგენს ა რ ი თ მ ე ტ ი კ უ ლ - ლ ო გ ი კ უ რ ი მ ო წ ყ ო ბ ი ლ ო ბ ა. ის უზრუნველყოფს ორი შეტანილი ცვლადის არითმეტიკულ და ლოგიკურ დამუშავებას, რის შედეგადაც ფორმირდება გამოსასვლელი ცვლადი. არითმეტიკულ-ლოგიკური მოწყობილობის ფუნქციები ჩვეულებრივ დაიყვანებიან მარტივ არითმეტიკულ და ლოგიკურ ოპერაციებამდე. ოპერაციის შედეგის გარდა არითმეტიკულ-ლოგიკური მოწყობილობა ახდენს შ ე დ ე გ ი ს ნ ი შ ა ნ თ ა რ ი გ ი ს (ალმების) ფორმირებას, რომლებიც ახასიათებენ მიღებულ შედეგსა და მისი მიღების პროცესში მომხდარ შემთხვევებს (წულთან ტოლობა, ნიშანი, ლუწობა, გადატანა, გადავსება და ა.შ) ალმები შეიძლება ანალიზდებოდნენ მარ-

თვის მოწყობილობაში პროგრამის ბრძანებების შესრულების შემდგომი თანმიმდევრობის შესახებ გადაწყვეტილების მიღების მიზნით.

მართვის მოწყობილობა და არითმეტიკულ-ლოგიკური მოწყობილობა მჭიდრო ურთიერთკავშირშია და მათ ჩვეულებრივ განიხილავენ როგორც ერთ მოწყობილობას, ცნობილს ცენტრალური პროცესორის სახელით ან უბრალოდ პროცესორად. მართვის მოწყობილობისა და არითმეტიკულ-ლოგიკური მოწყობილობის გარდა პროცესორში აგრეთვე შედის საერთო დანიშნულების რეგისტრთა ნაკრები, რომლებიც ემსახურებიან ინფორმაციის შუალედურ შენახვას პროცესორში მისი დამუშავების დროს.

1.4. გამომთვლელი მანქანებისა და სისტემების სტრუქტურათა ტიპები

გამომთვლელი მანქანებისა და სისტემების არქიტექტურის უპირატესობები და ნაკლოვანებები დამოკიდებულია კომპონენტთა შეერთების ხერხზე. ყველაზე უფრო ზოგადი მიდგომისას შეიძლება ვილაპარაკოთ გამომთვლელი მანქანების სტრუქტურათა ორ ძირითად ტიპზე და გამოთვლითი სისტემების სტრუქტურათა ორ ტიპზე.

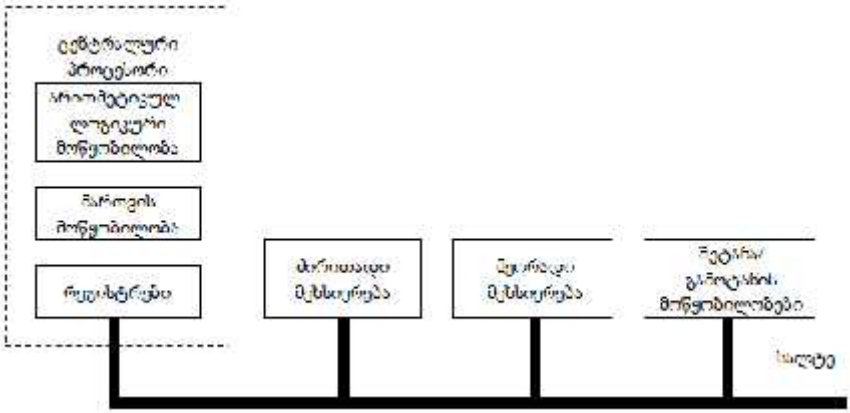
1.4.1. გამომთვლელი მანქანების სტრუქტურები

დღეისათვის დაახლოებით ერთნაირი გავრცელება ჰპოვა გამომთვლელი მანქანის აგების ორმა ხერხმა: უშუალო კავშირებით და სალტის საფუძველზე.

პირველი ხერხის ტიპიურ წარმომადგენლად შეიძლება განვიხილოთ ფონ-ნეიმანისეული გამომთვლელი მანქანა (ნახ. 1.8.). მასში ურთიერთმოქმედ მოწყობილობებს (პროცესორი, მეხსიერება, შეტა-

ნა/გამოტანის მოწყობილობები) შორის არსებობენ უშუალო კავშირები. კავშირთა თავისებურებები (სალტეებში ხაზთა - ფიზიკური გამტარების რიცხვი, გამტარუნარიანობა და ა.შ) განსაზღვრულია ინფორმაციის სახით, გაცვლის ხასიათით და ინტენსივობით. უშუალო კავშირებიანი არქიტექტურის უპირატესობად შეიძლება ჩაითვალოს „ვიწრო ადგილების“ გადაჭრის შესაძლებლობა მხოლოდ განსაზღვრული კავშირების სტრუქტურისა და მახასიათებლების გაუმჯობესების გზით, რაც ეკონომიკური თვალსაზრისით შეიძლება ყველაზე ხელსაყრელი იყოს. ფონ-ნეიმანისეულ გამომთვლელ მანქანებში ასეთ „ვიწრო ადგილს“ წარმოადგენს ცენტრალურ პროცესორსა და მეხსიერებას შორის მონაცემთა გადაგზავნის არხი, და მისი „გადაჭრა“ საკმაოდ რთულია. გარდა ამისა, უშუალო კავშირებიანი გამომთვლელი მანქანები ცუდად ექვემდებარებიან გადაწყობას ანუ რეკონფიგურაციას.

საერთო სალტიან ვარიანტში გამომთვლელი მანქანის ყველა მოწყობილობა მიერთებულია მაგისტრალურ არხზე, რომელიც ერთადერთი ტრაქტია ბრძანებების, მონაცემების და მართვის ნაკადებისათვის (ნახ. 1.9.). საერთო სალტის არსებობა არსებითად ამარტივებს გამომ-



ნახ. 1.9. საერთო სალტიანი გამომთვლელი მანქანის სტრუქტურა

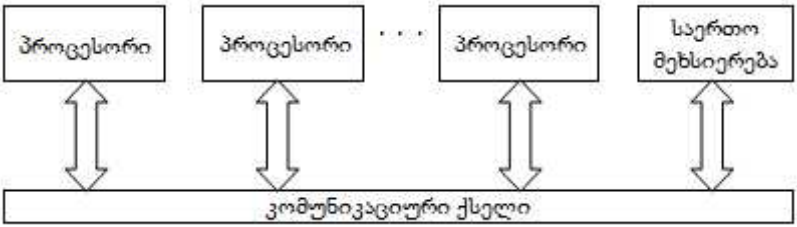
თვლელი მანქანის რეალიზაციას, საშუალებას იძლევა ადვილად შეიცვალოს მანქანის შემადგენლობა და კონფიგურაცია. ამ თვისებათა საშუალებით სალტურმა არქიტექტურამ ფართო გავრცელება ჰპოვა მინი და მიკრო ეგმ-ებში. ამასთან ერთად, სწორედ სალტესთანაა დაკავშირებული არქიტექტურის ძირითადი ნაკლი: დროის თითოეულ მომენტში სალტით ინფორმაციის გაცემა შეუძლია მხოლოდ ერთ მოწყობილობას. სალტეზე ძირითად დატვირთვას ქმნიან გაცვლები პროცესორსა და მეხსიერებას შორის, დაკავშირებულნი მეხსიერებიდან ბრძანებებისა და მონაცემების ამოღებით და გამოთვლის შედეგების მეხსიერებაში ჩაწერით. შეტანა/გამოტანის ოპერაციებზე რჩება სალტის გამტარუნარიანობის შესაძლებლობათა მხოლოდ ნაწილი. პრაქტიკა გვიჩვენებს, რომ საკმაოდ სწრაფი სალტის დროსაც კი დამატებათა 90%-ისათვის ეს დამატებითი რესურსები საკმარისი არაა, განსაკუთრებით მონაცემების დიდი მასივების შეტანის და გამოტანის დროს.

მთლიანობაში უნდა ვაღიაროთ, რომ ბრძანებათა თანმიმდევრობითი ფონ-ნეიმანისეული კონცეფციის შენარჩუნებისას სალტური არქიტექტურა „წმინდა“ სახით არასაკმარისად ეფექტურია. უფრო მეტადაა გავრცელებული არქიტექტურა იერარქიული სალტეებით, სადაც მაგისტრალური სალტის გარდა არის კიდევ რამდენიმე დამატებითი სალტე. მათ შეუძლიათ განახორციელონ უშუალო კავშირები ყველაზე ინტენსიური გაცვლის მქონე მოწყობოლობებს შორის, მაგალითად პროცესორსა და კეშ-მეხსიერებას შორის. დამატებითი სალტეების გამოყენების მეორე ვარიანტია შეტანა/გამოტანის ერთტიპიური მოწყობილობების გაერთიანება შემდგომი გამოსვლით დამატებითი სალტიდან მაგისტრალურზე. ყველა ეს ზომა საშუალებას იძლევა შემცირდეს დატვირთვა საერთო სალტეზე და უფრო ეფექტურად იქნეს გამოყენებული მისი გამტარიანობა.

1.4.2. გამოთვლითი სისტემების სტრუქტურები

ცნება „გამოთვლითი სისტემა“ გულისხმობს პროცესორთა ან დასრულებულ გამომთვლელ მანქანათა სიმრავლეს, რომელთა გაერთიანებისას გამოიყენება ორიდან ერთ-ერთი მიდგომა.

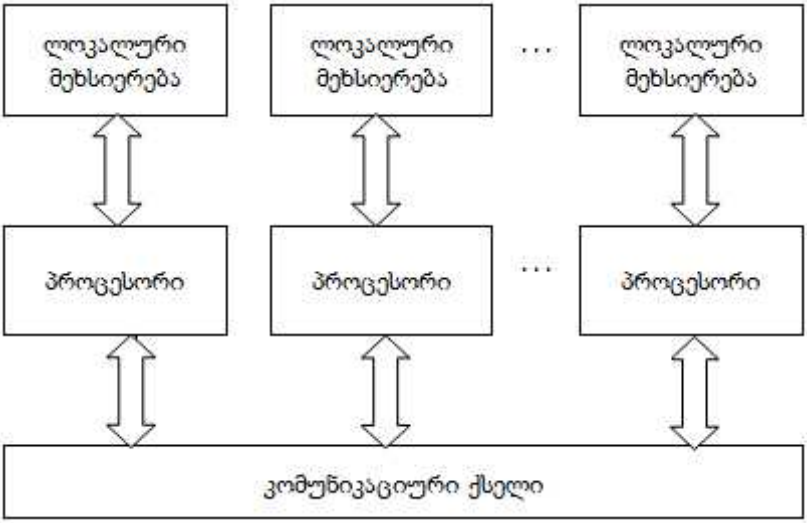
გამოთვლით სისტემებში საერთო მეხსიერებით (ნახ. 1. 10.) საერთოა ძირითადი მეხსიერება, ერთობლივად გამოყენებული სისტემის ყველა პროცესორის მიერ. პროცესორთა კავშირი მეხსიერებასთან უზრუნველყოფილია კომუნიკაციური ქსელის საშუალებით, რომელიც უფრო ხშირად გარდაქმნილია საერთო სალტეში. ამრიგად, საერთო მეხსიერებიანი გამოთვლითი სისტემის სტრუქტურა ანალოგიურია ზემოთ განხილული საერთო სალტიანი არქიტექტურის, რის გამოც მისთვის დამახასიათებელია იგივე ნაკლოვანებები. გამოთვლით სისტემაში გამოყენებისას მოცემულ სქემას გააჩნია დამატებითი უპირატესობები: პროცესორებს შორის ინფორმაციის გაცვლა არაა დაკავშირებული დამატებით ოპერაციებთან და უზრუნველყოფილია მეხსიერების საერთო ზონებთან მიღწევადობით.



ნახ. 1. 10. საერთო მეხსიერების მქონე გამომთვლელი სისტემის სტრუქტურა

ორგანიზაციის ალტერნატიული ვარიანტია განაწილებული სისტემა, სადაც საერთო მეხსიერება არ არის, ხოლო თითოეულ პროცესორს გააჩნია საკუთარი ლოკალური მეხსიერება (ნახ. 1. 11.). ხშირად ასეთი სისტემები აერთიანებენ ცალკეულ (დამოუკიდებელ) გამომთვლელ მანქანებს. სისტემის შემადგენლობას შორის ინფორმაციის

გაცვლა უზრუნველყოფილია კომუნიკაციური ქსელის საშუალებით შეტყობინებათა გაცვლის გზით.



ნახ. 1. 11. განაწილებული გამოთვლითი სისტემის სტრუქტურა

გამოთვლითი სისტემის ამგვარი აგება ხსნის საერთო სალტისათვის დამახასიათებელ შეზღუდვებს, მაგრამ იწვევს დამატებით დანახარჯებს შეტყობინებათა გადაგზავნაზე პროცესორებს ან მანქანებს შორის.

1.5. გამომთვლელი მანქანებისა და გამომთვლელი სისტემების გაუმჯობესების პერსპექტივები

გამომთვლელი მანქანებისა და სისტემების არქიტექტურის გაუმჯობესობა დაიწყო პირველი გამომთვლელი მანქანების გამოჩენის მომენტიდან და არ მთავრდება დღესაც. არქიტექტურაში თითოეული ცვლილება მიმართულია წარმადობის აბსოლუტურ გაზრდაზე ან, უკი-

დურეს შემთხვევაში, განსაზღვრული კლასის ამოცანების უფრო ეფექტურ ამოხსნაზე. არქიტექტურათა ევოლუციას განსაზღვრავენ სრულებით განსხვავებული ფაქტორები - ტექნოლოგია, პარალელიზმი, ღირებულება, ოპერაციული სისტემები, დამატებები და სხვა. თითოეული მათგანის როლის დაუკნინებლად საჭიროა ვაღიაროთ, რომ გამოთვლითი ტექნიკის საშუალებათა დამუშავებისას ყველაზე მკაფიო წარმატებები დაკავშირებულია ტექნოლოგიურ მიღწევებთან.

თითოეული ახალი ტექნოლოგიური წარმატებისას არქიტექტურული იდეებიდან ბევრი გადადის პრაქტიკული რეალიზაციის დონეზე.

ტექნოლოგიური პროგრესი ავსებს მნიშვნელოვან ცვლილებებს გამომთვლელი მანქანების არქიტექტურაში. უპირველეს ყოვლისა, ეს ეხება გამომთვლელი მანქანის პროცესორის შემადგენლობაში ინდექსური რეგისტრების გაჩენას, რამაც გაამარტივა მასივის ელემენტებთან მიღწევა. უპირველესად, მასივის ელემენტთა ციკლური დამუშავებისას, აუცილებელი იყო ბრძანების კოდის მოდიფიცირება, კერძოდ, მასში დამახსოვრებული მასივის ელემენტის მისამართის, როგორც შედეგი, გამოთვლების მიმდინარეობისას ზოგიერთი ბრძანების კოდის მუდმივად იცვლებოდნენ, რაც ართულებდა პროგრამის გამართვას. ინდექსური რეგისტრების გამოყენებით მასივის ელემენტის მისამართი გამოითვლება როგორც ჯამი ბრძანების სამისამართო ნაწილისა და ინდექსური რეგისტრის შემცველობისა. ეს საშუალებას იძლევა შესრულდეს მიმართვა მასივის ნებისმიერ ელემენტზე ბრძანების კოდზე ხელშეუხებლად, მხოლოდ ინდექსური რეგისტრის შემცველობის მოდიფიცირების გზით.

გამომთვლელი მანქანის სტრუქტურაში მეორე პრინციპული ცვლილება გახდა მცურავი მძიმის ფორმატში რიცხვთა დამუშავების აპარატურული ბლოკის დამატება. მანამდე ნამდვილ რიცხვთა დამუშავება ხორციელდებოდა ქვეპროგრამათა მეშვეობით, რომელთაგან თითოეული ახდენდა მცურავი მძიმით რაღაც ოპერაციის (შეკრება, გამ-

რავლება და ა.შ.) შესრულების იმიტაციას, იყენებდა რა ამ მიზნით ჩვეულებრივ მთელრიცხვა არითმეტიკულ-ლოგიკურ მოწყობილობას.

მესამე მნიშვნელოვანი ცვლილება გამომთვლელი მანქანის არქიტექტურაში ესაა გამომთვლელი მანქანის შემადგენლობაში შეტანა/გამოტანის პროცესორების შემოტანა, რომლებმაც გამოათავისუფლეს ცენტრალური პროცესორის შეტანა/გამოტანის მართვის რუტინული ოპერაციებისაგან და უზრუნველყოფს „მეხსიერება - შეტანა/გამოტანის მოწყობილობა“ ტრაქტის უფრო მაღალი გამტარუნარიანობას.

გამომთვლელი მანქანების არქიტექტურაში ერთ-ერთი უმნიშვნელოვანესი მოვლენა გახდა იდეა ბრძანებათა შეკვეცილ ნაკრებიანი გამომთვლელი მანქანების აგებისა (RISC, Redused Instruction Set Computer), რომელიც წამოყენებულ იქნა 1975 წელს და პირველი რეალიზაცია ჰპოვა 1980 წელს. გამარტივებულად RISC კონცეფციის არსი მდგომარეობს გამომთვლელი მანქანის ბრძანებათა ნაკრების დაყვანაზე ყველაზე მეტად გამოყენებულ უმარტივეს ბრძანებებამდე. ეს საშუალებას იძლევა პროცესორის სქემოტექნიკის გამარტივებისა და თითოეული „მარტივი“ ბრძანების შესრულების დროის მკვეთრად შემცირებისა. უფრო რთული ბრძანებები რეალიზებულია როგორც ქვეპროგრამები, შედგენილნი სწრაფი „მარტივი“ ბრძანებებისაგან.

მეხუთე თაობის გამოთვლითი სისტემების არქიტექტურაში შეიქმნა ორი პრინციპულად განსხვავებული მიდგომა: არქიტექტურა ერთობლივად გამოყენებული მეხსიერებით და არქიტექტურა განაწილებული მეხსიერებით.

და ბოლოს, მესამე მიმართულება მეხუთე თაობის გამოთვლითი სისტემების არქიტექტურაში ესაა გამოთვლითი სისტემები, რომლებშიც რამდენიმე ათასი საკმაოდ მარტივი პროცესორი მუშაობს საერთო მართვის მოწყობილობის მართვით და ერთდროულად ასრულებს ერთი და იგივე ოპერაციებს, მაგრამ თითოეული საკუთარ მონაცემებზე.

საერთოდ გამომთვლელი მანქანებისა და გამომთვლელი სისტემების არქიტექტურათა კვლევაში ძირითადი მიმართულებები პირობითად შეიძლება დაიყოს ორ ჯგუფად : ევოლუციურად და რევოლუციუ-

რად. პირველ ჯგუფს მიეკუთვნებიან ის კვლევები, რომელთა მიზეზსაც წარმოადგენს უკვე საკმაოდ ცნობილი იდეების რეალიზაციის მეთოდების სრულყოფა. გამოკვლევები, რომლებსაც პირობითად რევოლუციური ჰქვიათ, მიმართულნი არიან უკვე ტრადიციულებად ქცეულ ფონ-ნეიმანისეული სრულებით ახალი არქიტექტურების შექმნაზე.

ევოლუციური კვლევების უმრავლესობა დაკავშირებულია მიკროპროცესორების არქიტექტურათა სრულყოფაზე. პრინციპში მიკროპროცესორებში კარდინალურად ახალი არქიტექტურული მიდგომების რაოდენობა შედარებით მცირე. ძირითადი იდეები, რომლებიც საფუძვლად უდევთ თანამედროვე მიკროპროცესორებს, წამოყენებული იყვნენ მრავალი წლის წინ, მაგრამ ტექნოლოგიის არასრულყოფილებისა და რეალიზაციის მაღალი ღირებულების გამო გამოყენება ჰპოვეს მხოლოდ დიდ უნივერსალურ გამომთვლელ მანქანებში (მეინფრეიმებში) და სუპერ ეგმ-ებში. მიკროპროცესორთა არქიტექტურაში ყველაზე მეტად მნიშვნელოვანი ცვლილებები დაკავშირებულნი არიან ბრძანებათა დონეზე პარალერულობის დონის ამაღლებასთან (რამდენიმე ბრძანების ერთდროულად შესრულების შესაძლებლობა). აქ პირველ რიგში უნდა ავღნიშნოთ კონვეიერიზაცია, სუპესკალარული დამუშავება და არქიტექტურა ზეგრძელი ბრძანებითი სიტყვებით (VLIW). „დიდი“ სისტემების გლობალური არქიტექტურული მიდგომების მიკროპროცესორებზე წარმატებით გადატანის შემდეგ მკლევართა ძირითადი ძალისხმევა უკვე მიმართულია კერძო არქიტექტურულ ცვლილებებზე. ასეთი ევოლუციური არქიტექტურების მაგალითებს წარმოადგენენ: ბრძანებათა კონვეირებში პროგნოზირების მეთოდების გაუმჯობესება, კეშ-მეხსიერებაზე წარმატებულ მიმართვათა სიხშირის ზრდა ბუფერიზაციის ხერხების გართულების ხარჯზე და ა.შ.

ფართო გამოყენების გამოთვლით საშუალებათა სფეროში განხორციელებული მიღწევები ჯერ-ჯერობით განპირობებულნი არიან სწორედ „ევოლუციური“ კვლევებით. მაგრამ ახლავეა ცხადი, რომ ტრადიციული არქიტექტურის ჩარჩოში დარჩენით, ჩვენ საკმაოდ მალე შევეჯახებით ტექნოლოგიურ შეზღუდვებს. ტექნოლოგიური ბარიერის

დაძლევის ერთ-ერთი გზა არატრადიციულ მიდგომებშია. ამ მიმართულებით ჩატარებული კვლევები, ჩვენი კლასიფიკაციით „რევოლუციურებს“ მიეკუთვნებიან. ასეთი მტკიცებულებების სამართლიანობა დასტურდება არატრადიციული არქიტექტურის გამოთვლით სისტემების პირველი ნიმუშებით.

ვაფასებთ რა გამოთვლითი ტექნიკის ევოლუციური და რევოლუციური პერსპექტივებს, შეიძლება ვამტკიცოთ, რომ ახლო მომავალში დიდ პროგრესს შეძლება ველოდოთ პარალელურობის იდეების გამოყენებისა და დამახსოვრების მოწყობილობათა ეფექტური იერარქიის შექმნის გზაზე.

2. ბრძანებათა სისტემის არქიტექტურა

გამომთვლელი მანქანის ბრძანებათა სისტემა ეწოდება ბრძანებათა სრულ ჩამონათვალს, რომლის შესრულებაც შეუძლია მოცემულ გამომთვლელ მანქანას. თავის მხრივ, ბრძანებათა სისტემის არქიტექტურის ქვეშ მიღებულია განისაზღვროს გამომთვლელი მანქანის ის საშუალებები, რომლებსაც ხედავს პროგრამისტი და ის მისაწვდომია მისთვის. ბრძანებათა სისტემის არქიტექტურა შეიძლება განვიხილოთ, როგორც შეთანხმების ხაზი პროგრამული უზრუნველყოფის დამმუშავებელთა საჭიროებებსა და გამომთვლელი მანქანის აპარატურის შემქმნელთა შესაძლებლობებს შორის.

საბოლოოდ, როგორც პირველთა ისე მეორეთა მიზანია გამოთვლათა რეალიზაცია ყველაზე უფრო ეფექტური სახით, ე. ი. მინიმალურ დროში, და აქ უმნიშვნელოვანეს როლს თამაშობს ბრძანებათა სისტემის არქიტექტურის სწორი შერჩევა.

გამარტივებულად პროგრამის შესრულების დრო ($T_{გამოთ}$) შეიძლება განისაზღვროს პროგრამაში ბრანებათა რიცხვის ($N_{ბრძ}$), ერთ ბრძანებაზე მოსული პროცესორის ტაქტების საშუალო რაოდენობის (CPI) და ტაქტური პერიოდის ხანგრძლივობის ($\tau_{ჯრ}$) საშუალებით:

$$T_{გამოთ} = N_{ბრძ} \times CPI \times \tau_{ჯრ}$$

გამოსახულების თითოეული შემადგენელი დამოკიდებულია ბრძანებათა სისტემის არქიტექტურის ერთ ასპექტზე და თავის მხრივ, მოქმედებს სხვებზე, რაც მეტყველებს ბრძანებათა სისტემის არქიტექტურის არჩევისადმი საგანგებოდ პასუხისმგებლური მიდგომის აუცილებლობაზე.

გამომთვლელი მანქანის ბრძანებათა სისტემის არქიტექტურის ზოგადი დახასიათებისათვის საჭიროა პასუხი გაეცეს შემდეგ შეკითხვებს:

1. რა სახის მონაცემები იქნებიან წარმოდგენილი გამომთვლელ მანქანაში და რა ფორმით?

2. სად შეიძლება შენახულ იქნას ეს მონაცემები გარდა ძირითადი მეხსიერებისა?

3. როგორ განხორციელდება მონაცემებთან მიღწევა?

4. რა ოპერაციები შეიძლება შესრულდეს მონაცემებზე?

5. რამდენი ოპერანდი შეიძლება იყოს ბრძანებაში?

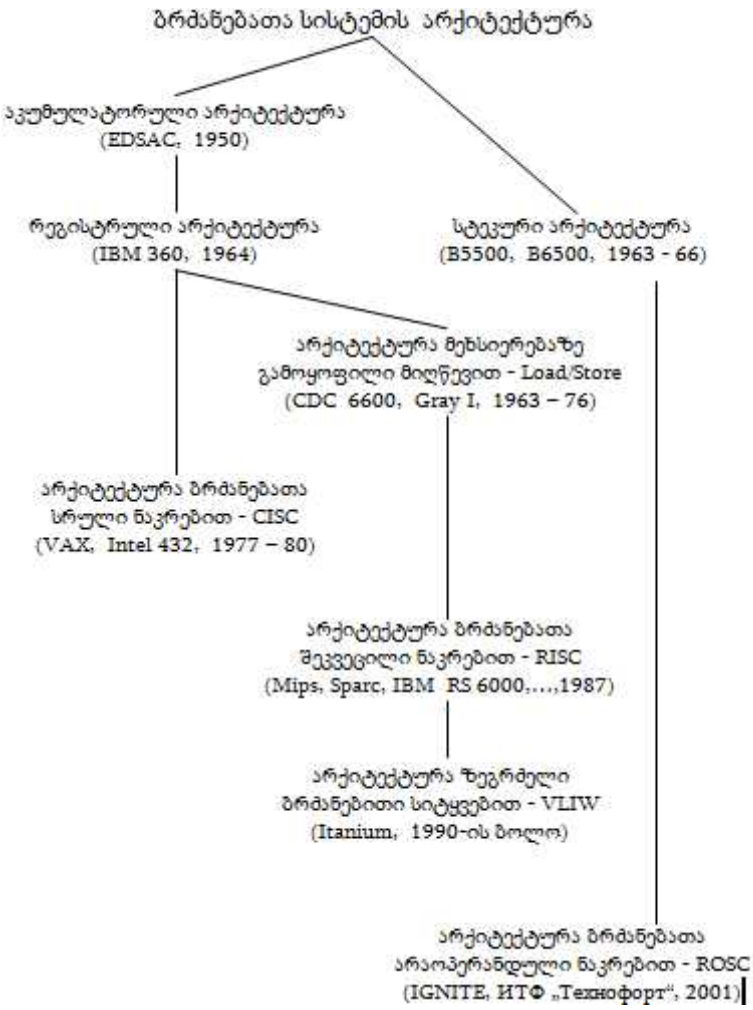
6. როგორ მოხდება მომდევნო ბრძანების მისამართის განსაზღვრა?

7. როგორ იქნებიან ბრძანებები კოდირებული?

2.1. ბრძანებათა სისტემის არქიტექტურის კლასიფიკაცია

დღეისათვის ჩამოყალიბებული ბრძანებათა სისტემის არქიტექტურის კლასიფიკაცია მოყვანილია ნახ.2.1-ზე.

ბრძანებათა სისტემის არქიტექტურის ახალ ტიპზე გადასვლის განმსაზღვრელ მოტივებს შორის შევჩერდებით ორ უფრო არსებითზე. პირველი - გამომთვლელი მანქანის მიერ შესრულებულ ოპერაციათა შემადგენლობა და მათი სირთულე; მეორე - ოპერანდების შენახვის ადგილი, რაც მოქმედებს მონაცემთა დამუშავების ბრძანებების სამისამართო ნაწილში მითითებული მისამართების რაოდენობაზე და სიგრძეზე. სწორედ ეს მონაცემებია აღებული ბრძანებათა სისტემის არქიტექტურის ქვემოთ მოყვანილი კლასიფიკაციის ვარიანტებში კრიტერიუმებად.



ნახ. 2.1. ბრძანებათა სისტემის არქიტექტურათა განვითარების ქრონოლოგია

კლასიფიკაცია ბრძანებათა შემადგენლობასა და სირთულის მიხედვით

პროგრამირების თანამედროვე ტექნოლოგია ორიენტირებულია მაღალი დონის ენებზე, რომელთა მთავარი მიზანია დაპროგრამების პროცესის გამარტივება. ოღონდ, მაღალი დონის ენებზე გადასვლამ წარმოშვა სერიოზული პრობლემა. მაღალი დონის ენებისათვის დამახასიათებელი რთული ოპერატორები არსებითად განსხვავდებიან უმრავლეს გამომთვლელ მანქანებში რეალიზებული მარტივი მანქანური ოპერაციებისაგან. პრობლემამ მიიღო ს ე მ ა ნ ტ ი კ უ რ ი გ ა რ ღ ვ ე ვ ი ს სახელწოდება, ხოლო შედეგად მივიღეთ გამომთვლელ მანქანაზე პროგრამების შესრულების არასაკმარისი ეფექტურობა. სემანტიკური გარღვევის გადასალახავად გამომთვლელი მანქანების დამმუშავებლები ირჩევენ სამიდან ერთ-ერთ მიდგომას და შესაბამისად, ბრძანებათა სისტემის არქიტექტურის სამიდან ერთ-ერთ ტიპს.

- არქიტექტურას ბრძანებათა სრული ნაკრებით:
  CISC (Complex Instruction Set computer);
- არქიტექტურას ბრძანებათა შეკვეცილი ნაკრებით:
  RISC (Redukal Insruktion Set Computer);
- არქიტექტურას ზეგრძელი ბრძანებითი სიტყვებით:
  VLIW (Very long Ins'truction Word).

CISC ტიპის გამომთვლელ მანქანებში სემანტიკური გარღვევის პრობლემა წყდება ბრძანებათა სისტემის გაფართოების ხარჯზე, ბრძანებათა სისტემის არქიტექტურაში ოპერატორების სემანტიკურად ანალოგიური რთული ბრძანებების დამატებით. CISC -არქიტექტურის დამფუძნებლად ითვლება IBM კომპანია, რომელმაც მოცემული მიდგომის გამოყენება დაიწყო IBM 360 ოჯახის მანქანებიდან და აგრძელებს მას თავის მძლავრ თანამედროვე უნივერსალურ მანქანებში, როგორიცაა IBM ES/ 9000. ანალოგიური მიდგომა დამახასიათებელია Intel კომპანიისათვისაც მის 8086 და Pentium სერიის მიკროპროცესორებში. CISC არქიტექტურისათვის დამახასიათებელია:

- პროცესორში საერთო დანიშნულების შედარებით მცირე რიცხვის რეგისტრების არსებობა;

- მანქანური ბრძანებების დიდი რიცხვი, რომელთაგან ზოგიერთი აპარატურულად ახორციელებს ბრძანებათა სისტემის არქიტექტურის რთული ოპერატორების რეალიზაციას;

- ოპერანდების დამისამართების ხერხების მრავალფეროვნება;

- სხვადასხვა თანრიგის ბრძანებათა ფორმატების სიმრავლე;

- იმ ბრძანებათა არსებობა, რომლებშიც დამუშავება შეთავსებულია მეხსიერებაზე მიმართვას.

CISC ტიპს შეიძლება მივაკუთვნოთ 1980-იანი წლების შუამდე გამოშვებული პრაქტიკულად ყველა გამომთვლელი მანქანა და დღეს გამოშვებული მანქანების უმრავლესობა. სემანტიკური გარღვევის პრობლემის გადაჭრის განხილულ ხერხს იმავდროულად მივყავართ გამომთვლელი მანქანის აპარატურის გართულებასთან, განსაკუთრებით მართვის მოწყობილობის, რაც, თავის მხრივ, ნეგატიურად აისახება მთლიანობაში გამომთვლელი მანქანის წარმადობაზე. ამან აიძულა დამმუშავებლები უფრო ყურადღებით გაეანალიზებინათ ბრძანებათა სისტემის არქიტექტურიდან კომპილაციის შედეგად მიღებული პროგრამები.

კომპლექსური კვლევების ჩატარების შემდეგ დადგინდა, რომ ბრძანებათა სისტემის არქიტექტურის ოპერატორების ჩატარების ექვივალენტური დამატებითი ბრძანებების წილი პროგრამათა საერთო მოცულობაში არ აღემატება 10-20%-ს, ხოლო ზოგიერთი ფრიად რთული ბრძანებისა 0,2%-ს. ამასთან, დამატებითი ბრძანებების რეალიზაციისათვის საჭირო აპარატურული საშუალებების მოცულობა მნიშვნელოვნად იზრდება. ასე, მაგალითად, მიკროპროგრამული მეხსიერების ტევადობა რთული ბრძანებების მხარდაჭერის დროს შეიძლება 60% -ით გაიზარდოს.

აღნიშნული კვლევების შედეგების დეტალურმა ანალიზმა მიგვიყვანა ტრადიციული გადაწყვეტილებების სერიოზულ გადაფასებამდე, რისი შედეგიც გახდა RISC - ა რ ქ ი ტ ე ქ ტ უ რ ი ს შექმნა. ტერმი-

ნი RISC პირველად 1980 წელს იქნა გამოყენებული. იდეა მდგომარეობს გამომთვლელი მანქანის ბრძანებათა სიის შეზღუდვაში ყველაზე ხშირად გამოყენებული უმარტივესი ბრძანებებით, რომლებიც ოპერირებენ მხოლოდ პროცესორის რეგისტრებში განლაგებულ მონაცემებზე. მეხსიერებაზე მიმართვა დასაშვებია წაკითხვასა და ჩაწერის მხოლოდ სპეციალური ბრძანებების საშუალებით მკვეთრადაა შემცირებული ბრძანებათა ფორმატები და ოპერანდების მისამართების მითითების ხერხები. ბრძანებების ფორმატების რიცხვის შემცირება და მათი სიმარტივე, დამისამართების ხერხების შეზღუდული რაოდენობის გამოყენება, მონაცემთა დამუშავების ოპერაციების განცალკევება მეხსიერებაზე მიმართვის ოპერაციებიდან, საშუალებას იძლევა მნიშვნელოვნად გამარტივდეს გამომთვლელი მანქანის აპარატურული საშუალებები და გაიზარდოს მათი სწრაფქმედება. RISC არქიტექტურა ისე მუშავდებოდა, რომ შემცირებულიყო $T_{\text{გამოთ}}$ CPI -ის და $\tau_{\text{პრ}}$-ს შემცირების ხარჯზე. როგორც შედეგი, რთული ბრძანებების რეალიზაცია მარტივი, მაგრამ სწრაფი RISC- ბრძანებების თანმიმდევრობით არა ნაკლებ ეფექტურია, ვიდრე რთული ბრძანებების აპარატურული ვარიანტი CISC არქიტექტურაში.

RISC- არქიტექტურის ელემენტები პირველად გამოჩნდნენ კომპანია Cray Resakch-ის CDC 6600 გამომთვლელ მანქანებსა და სუპერ ეგმ-ში. საკმაოდ წარმატებით რეალიზდება RISC -არქიტექტურა თანამედროვე გამომთვლელ მანქანებში.

ავღნიშნოთ, რომ ფირმა Intel-ის და AMD-ს ბოლო მიკროპროცესორებში ფართოდ გამოიყენებიან RISC-არქიტექტურისათვის დამახასიათებელი იდეები; ასე, რომ ბევრი განსხვავება CISC-სა და RISC-ს შორის თანდათან ქრება.

საერთო კლასიფიკაციაში CISC და RISC არქიტექტურების გარდა მოხსენიებული იყო ბრძანებათა სისტემის არქიტექტურის კიდევ ერთი ტიპი - არქიტექტურა ზეგრძელი ბრძანებითი სიტყვებით (VLIW). კონცეფცია VLIW ეფუძნება RISC-არქიტექტურას, სადაც რამდენიმე მარტივი RISC-ბრძანება ერთიანდება ერთ ზეგრძელ ბრძანებაში და პარალელურად სრულდება. ბრძანებების სისტემური არქიტექტურის

თვალსაზრისით VLIW არქიტექტურა შედარებით ნაკლებად განსხვავდება RISC-საგან. გაჩნდა მხოლოდ პარალელურ გამოთვლათა დამატებითი დონე, რის გამოც VLIW არქიტექტურა უფრო ლოგიკურია განვიხილოთ არა გამომთვლელ მანქანებში, არამედ გამოთვლით სისტემებში.

ცხრილი 1 საშულებას იძლევა შეფასდეს ყველაზე მეტად არსებითი განსხვავებები CISC, RISC, და VLIW არქიტექტურებს შორის.

ცხრილი 1.

მახასიათებელი	CISC	RISC	VLIW
ბრძანების სიგრძე	ვარირებს	ერთიანი	ერთიანი
ველების განლაგება ბრძანებაში	ვარირებს	უცვლელია	უცვლელია
რეგისტრების რაოდენობა	რამდენიმე (ხშირად სპეციალიზებული)	საერთო დანიშნულების ბევრი რეგისტრი	საერთო დანიშნულების ბევრი რეგისტრი
მეხსიერებასთან მიღწევა	შეიძლება შესრულდეს როგორც სხვადასხვა ტიპის ბრძანებათა ნაწილი	სრულდება მხოლოდ სპეციალური ბრძანებებით	სრულდება მხოლოდ სპეციალური ბრძანებებით

კლასიფიკაცია ოპერანდების შენახვის ადგილის მიხედვით

ბრძანებათა რაოდენობა და მათი სირთულე, მართლაც მნიშვნელოვანი ფაქტორებია, მაგრამ ბრძანებათა სისტემის არქიტექტურის არჩევისას არანაკლებ როლს თამაშობს პასუხი კითხვაზე, თუ სად უნდა ინახებოდნენ ოპერანდები და რა სახით უნდა განხორციელდეს მათთან მიღწევა. ამ პოზიციებიდან ანსხვავებენ ბრძანებათა სისტემის არქიტექტურის შემდეგ სახეებს:

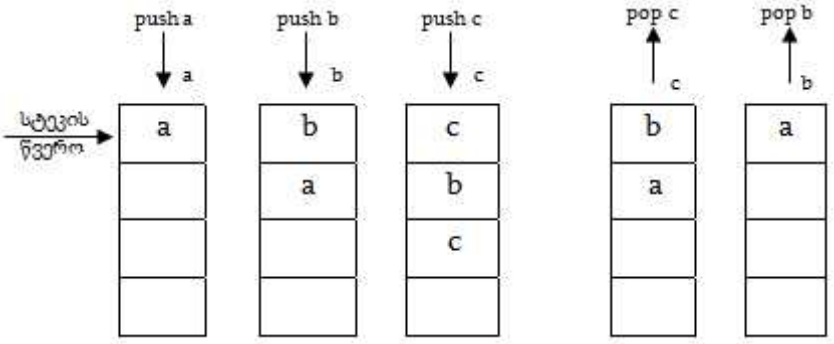
- სტეკური;
- აკუმულატორული;
- რეგისტრული;

- მეხსიერებასთან გამოყოფილი მიღწევით;

ამა თუ იმ არქიტექტურის არჩევა მოქმედებს შემდეგ პრინციპულ მომენტებზე: რამდენი მისამართი იქნება ბრძანების სამისამართო ნაწილში, როგორი იქნება ამ ბრძანებების სიგრძე, რამდენად მარტივად განხორციელდება ოპერანდებთან მიღწევა და, საბოლოოდ, როგორი იქნება ბრძანების საერთო სიგრძე.

სტეკური არქიტექტურა. სტეკი ეწოდება მეხსიერებას, რომელიც თავისი სტრუქტურული ორგანიზაციით განსხვავებულია გამომთვლელი მანქანის ძირითადი მეხსიერებისაგან. სტეკური მეხსიერების აგების პრინციპებს დეტალურად მოგვიანებით განვიხილავთ, აქ კი გამოვყოფთ მხოლოდ იმ ასპექტებს, რომლებიც გვესაჭიროება სტეკის საფუძველზე ბრძანებათა სისტემის არქიტექტურის თავისებურებების განმარტებისათვის.

სტეკი ქმნის ლოგიკურად ურთიერთდაკავშირებულ უჯრედთა სიმრავლეს (ნახ.2.2) რომლებიც ურთიერთქმედებენ პრინციპით „ბოლო შევიდა, პირველი გამოვიდა“ (LIFO, Last In First Out).

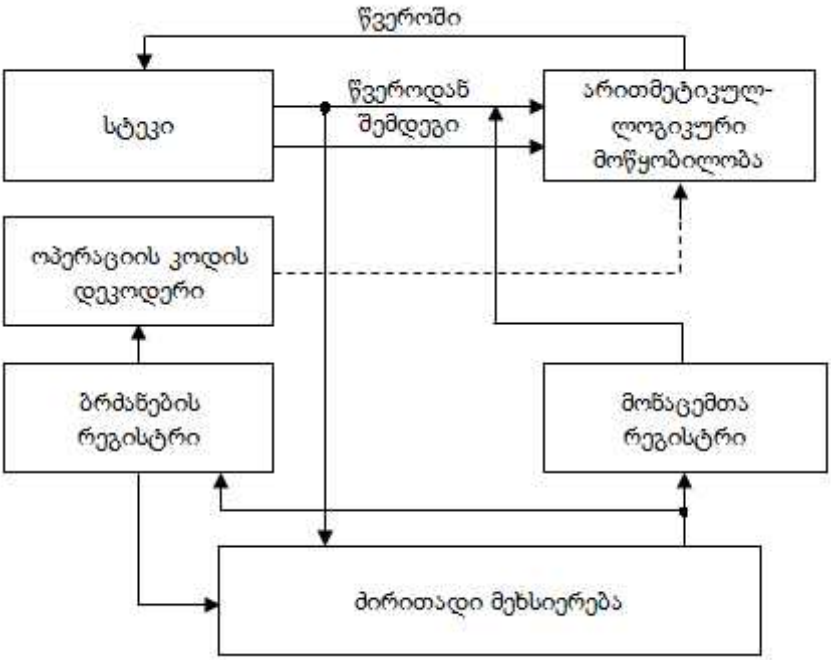


ნახ.2.2. სტეკური მეხსიერების მოქმედების პრინციპი

ზედა უჯრედს **სტეკის წვეროს** უწოდებენ. სტეკთან სამუშაოდ გათვალისწინებულია ორი ოპერაცია: push (მონაცემთა შეტანა სტეკში) და pop (მონაცემთა სტეკიდან გამოტანა). ჩაწერა შეიძლება მხოლოდ სტეკის ზედა უჯრედში, ამასთან სტეკში შენახული მთელი

ინფორმაცია წინასწარ იძვრის ერთი პოზიციით ქვევით. წაკითხვა დასაშვებია აგრეთვე მხოლოდ სტეკის წვეროდან. ამოღებული ინფორმაცია იშლება სტეკიდან, ხოლო მისი დარჩენილი ნაწილი იძვრის ზემოთ გამომთვლელ მანქანებში, რომლებშიც ბრძანებათა სისტემის არქიტექტურა რეალიზებულია სტეკის საფუძველზე (მათ ჩვეულებრივ სტეკურებს ეძახიან), დამუშავების წინ ოპერანდები თავსდებიან სტეკური მეხსიერების ორ ზედა უჯრედში. ოპერაციის შედეგი შეაქვთ სტეკში.

სტეკური ბრძანებათა სისტემის არქიტექტურის საფუძველზე აგებული მანქანის ერთ-ერთი შესაძლო ვარიანტის ძირითადი კვანძები და ინფორმაციული ტრაქტები მოტანილია ნახ. 2.3-ზე.



ნახ. 2.3. სტეკის საფუძველზე აგებული გამომთვლელი მანქანის არქიტექტურა

ინფორმაცია შეიძლება შეტანილი იყოს სტეკის წვეროში მეხსიერებიდან ან არითმეტიკულ-ლოგიკური მოწყობილობიდან. სტეკში x მისამართის მქონე უჯრედის შემცველობის ჩასაწერად სრულდება ბრძანება push x, რომლის მიხედვითაც ინფორმაცია ამოიკითხება მეხსიერების უჯრედიდან, შედის მონაცემთა რეგისტრში, ხოლო შემდეგ ჩაიწერება (იძვრის) სტეკში. გამოთვლის შედეგი არითმეტიკულ-ლოგიკური მოწყობილობიდან ავტომატურად იწერება სტეკის წვეროში.

x მისამართის მეხსიერების უჯრედში სტეკის წვეროს შემცველობის შენახვა ხორციელდება ბრძანებით pop x. ამ ბრძანებით სტეკის ზედა უჯრედის შემცველობა მიეწოდება სალტეს, რომლითაც ხორციელდება ჩაწერა x უჯრედში, რის შემდეგაც სტეკში არსებული მთელი ინფორმაცია იძვრის ერთი პოზიციით ზემოთ.

არითმეტიკული ან ლოგიკური ოპერაციის შესასრულებლად არითმეტიკულ-ლოგიკური მოწყობილობის შესასვლელს მიეწოდება სტეკის ორი ზედა უჯრედიდან წაკითხული ინფორმაცია (ამასთან სტეკის შემცველობა იძვრის ორი პოზიციის ზემოთ, ანუ ხდება სტეკიდან ოპერანდების წაშლა). ოპერაციის შედეგი ჩაიწერება სტეკის წვეროში. შესაძლოა ვარიანტი, როცა შედეგი პირდაპირ გადაიწერება მეხსიერებაში pop x ოპერაციის ავტომატური შესრულების საშუალებით.

სტეკური მეხსიერების ზედა უჯრედები, სადაც ინახებიან ოპერანდები და სადაც იწერება ოპერაციის შედეგი, როგორც წესი, უფრო სწრაფმოქმედნი მზადდებიან და თავსდებიან პროცესორში, მაშინ როცა სტეკის დანარჩენი ნაწილი შეილება მოთავსებული იყოს როგორც ძირითად მეხსიერებაში, ასევე ნაწილობრივ გარე მეხსიერებაში.

სტეკის საფუძველზე აგებული ბრძანებათა სისტემის არქიტექტურის უპირატესობებს უნდა მივაკუთვნოთ ბრძანებების სამისამართო ნაწილის შემცირების (შეკვეცის) შესაძლებლობა. ვინაიდან ყველა ოპერაცია სრულდება სტეკის წვეროს გავლით, ამიტომ ოპერანდებისა და შედეგების მისამართების მითითება ინფორმაციის დამუშავების არითმეტიკულ და ლოგიკურ ბრძანებებში საჭირო არაა. პროგრამის

კოდი კომპაქტური გამოდის. საკმაოდ მარტივად ხორციელდება ბრძანებათა დეკოდირება.

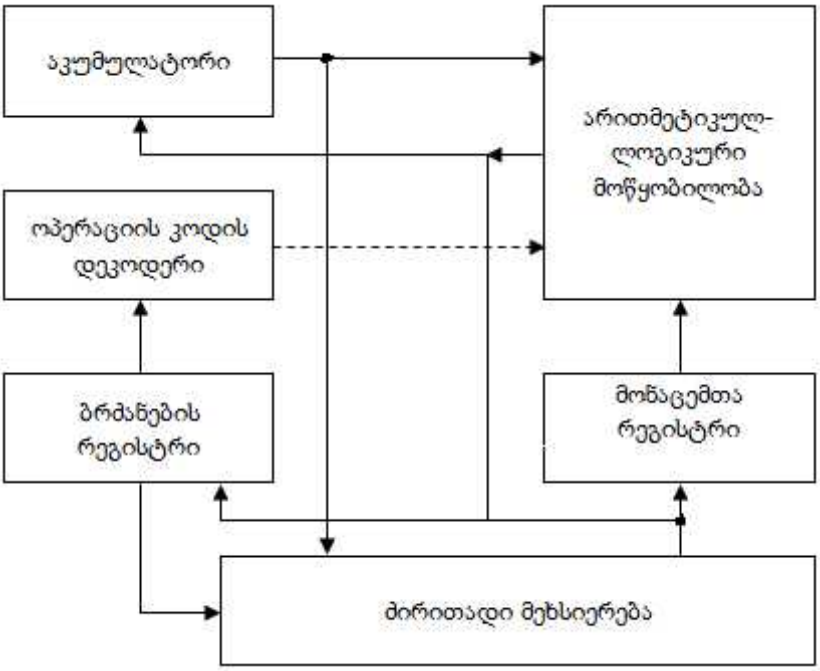
მეორეს მხრივ, განსაზღვრების თანახმად სტეკური ბრძანებათა სისტემის არქიტექტურა არ გულისხმობს მეხსიერებასთან ნებისმიერ მიღწევას, რის გამოც კომპილიატორს ძლიერ უჭირს შექმნას ეფექტური პროგრამული კოდი, თუმცა თავად კომპილიატორთა შექმნა მარტივდება. გარდა ამისა, სტეკი ხდება გამომთვლელი მანქანის „ვიწრო ადგილი“ წარმადობის თვალსაზრისით. ზემოთმოყვანილი მიზეზების გამო, ბრძანებათა სისტემის არქიტექტურის მოცემული სახე დიდი ხნის განმავლობაში ითვლებოდა არაპერსპექტიულად და უმთავრესად გვხვდებოდა 1960-იანი წლების გამომთვლელ მანქანებში, მაგალითად ფირმა Burroughs (B5500, B6500) ან ფირმა Hewlett-Packard (HP2116B, HP3000/70) გამომთვლელ მანქანებში.

გამოთვლითი ტექნიკის სფეროში მომხდარი ბოლო მოვლენები მეტყველებენ გამომთვლელი მანქანების სტეკური არქიტექტურის აღორძინებისადმი ინტერესს. სტეკური ბრძანებათა სისტემის არქიტექტურის მქონე თანამედროვე გამომთვლელ მანქანებს შორის შეიძლება ავღნიშნოთ Ajile Systems კომპანიის JEM 1 და JEM 2 და Chip ფირმის Imsys მანქანები. განსაკუთრებული აღნიშვნის ღირსია კომპანია Patriot Scientist სტეკური მანქანა IGNITE, რომელსაც მისი ავტორები თვლიან ბრძანებათა სისტემის არქიტექტურის ახალი სახის წარმომადგენლად - არქიტექტურა ბრძანებათა არაოპერანდული ნაკრებით. ასეთი გამომთვლელი მანქანის აღნიშნვნისთვის ისინი გვთავაზობენ ROSC (Removed Operand Set Computer) აბრევიატურას. მკაცრად რომ ვიმსჯელოთ, თავისი არსით ROSC ცოტათი განსხვავდება სტეკის საფუძველზე აგებული ტრადიციული არქიტექტურისაგან, და მისი ცალკე გამოყოფა სრულყოფილად დასაბუთებულად არ გვეჩვენება.

ა კ უ მ უ ლ ა ტ ო რ უ ლ ი ა რ ქ ი ტ ე ქ ტ უ რ ა. აკუმულატორის საფუძველზე აგებულ გამომთვლელი მანქანის არქიტექტურა ისტორიულად ერთ-ერთი პირველი შეიქმნა. მასში არითმეტიკული ან ლოგიკური ოპერაციის ერთ-ერთი ოპერანდის შესანახად პროცესორში

არის გამოყოფილი რეგისტრი - ა კ უ მ უ ლ ა ტ ო რ ი. ამავე რეგისტრში შეაქვთ ოპერაციის შედეგიც. ვინაიდან ერთ-ერთი ოპერანდის მისამართი თავიდანვე განსაზღვრულია, ამიტომ დამუშავების ბრძანებებში საკმარისია ცხადად მივუთითოთ მხოლოდ მეორე ოპერანდის ადგილმდებარეობა. თავიდან ორივე ოპერანდი ინახება ძირითად მეხსიერებაში და ოპერაციის შესრულებამდე მათგან ერთი საჭიროა ჩაიტვირთოს აკუმულატორში. დამუშავების ბრძანების შესრულების შემდეგ შედეგი ინახება აკუმულატორში და, თუ ის მომდევნო ბრძანებისათვის ოპერანდი არაა, მაშინ საჭიროა მისი მეხსიერების უჯრედში შენახვა.

აკუმულატორის ბაზაზე აგებული გამომთვლელი მანქანის ტიპიური არქიტექტურა მოტანილია ნახ. 2.4-ზე.



ნახ.2.4. აკუმულატორის ბაზაზე აგებული გამომთვლელი მანქანის არქიტექტურა

აკუმულატორში x უჯრედის შემცველობის ჩასატვირთავად გათვალისწინებულია ჩატვირთვის load x ბრძანება. ამ ბრძანებით ხდება ინფორმაციის ამოკითხვა მეხსიერების x უჯრედიდან, მეხსიერების გამოსასვლელი უერთდება აკუმულატორის შესასვლელებს და ხორციელდება ამოკითხული მონაცემების აკუმულატორში შეტანა.

x უჯრედში აკუმულატორის შემცველობის შეტანა ხორციელდება დამახსოვრების store x ბრძანებით, რომლის შესრულების შემდეგაც აკუმულატორის გამოსასვლელები უერთდებიან სალტეს, რის შემდეგაც ინფორმაცია სალტიდან იწერება მეხსიერებაში.

ოპერაციის შესასრულებლად არითმეტიკულ-ლოგიკურ მოწყობილობაში ხდება მეხსიერებიდან ერთ-ერთი ოპერანდის მონაცემთა რეგისტრში გადმოწერა: მეორე ოპერანდი აკუმულატორში იმყოფება. მონაცემთა რეგისტრისა და აკუმულატორის გამოსასვლელები უერთდებიან არითმეტიკულ-ლოგიკურ მოწყობილობის შესაბამის შესასვლელებს. მიწერილი ოპერაციის დასრულების შემდეგ შედეგი არითმეტიკულ-ლოგიკურ მოწყობილობის გამოსასვლელიდან აკუმულატორში შეიტანება.

ა კ უ მ უ ლ ა ტ ო რ უ ლ ი ბრძანებათა სისტემის არქიტექტურის უპირატესობად შეიძლება ჩაითვალოს მოკლე ბრძანებების დეკოდირების სიმარტივე. ოღონდ მხოლოდ ერთი რეგისტრის არსებობა იწვევს ძირითად მეხსიერებაზე მრავალჯერად მიმართვებს.

ბრძანებათა სისტემის არქიტექტურები აკუმლატორის ბაზაზე პოპულარულები იყვნენ ადრეულ გამომთვლელ მანქანებში, მაგალითად ისეთებში, როგორებიცაა IBM 7090, DEC PDP-8, MOS 6502.

რ ე გ ი ს ტ რ უ ლ ი ა რ ქ ი ტ ე ქ ტ უ რ ა. მოცემული ტიპის მანქანებში პროცესორი შეიცავს რეგისტრების მასივს, ცნობილს საერთო დანიშნულების რეგისტრების სახელწოდებით. ეს რეგისტრები, რაღაც აზრით, შეიძლება განვიხილოთ როგორც ცხადად მართვადი კეში ცოტა ხნის წინ გამოყენებული მონაცემების შესანახად.

რეგისტრების ზომები ჩვეულებრივ ფიქსირებულია და ემთხვევიან მანქანური სიტყვის ზომებს. ნებისმიერ რეგისტრს შეიძლება მივ-

მართოთ მისი ნომრის მითითებით. საერთო დანიშნულების რეგისტრების რაოდენობა CISC ტიპის არქიტექტურაში ჩვეულებრივ დიდი არაა (8-დან 32-მდე) და კონკრეტული რეგისტრის ნომრის წარმოსადგენად აუცილებელია არა უმეტეს ხუთი თანრიგისა, რისი მეშვეობითაც დამუშავების ბრძანებების სამისამართო ნაწილში დასაშვებია ერთდროულად მივუთითოთ ორი, ზოგჯერ კი სამი რეგისტრის ნომერი (ოპერანდთა ორი რეგისტრი და შედეგის რეგისტრი). RISC - არქიტექტურა გულისხმობს საერთო დანიშნულების რეგისტრების მნიშვნელოვნად მეტი რიცხვის გამოყენებას (რამდენიმე ასეულამდე), ოღონდ ასეთი გამომთვლელი მანქანებისათვის ბრძანების ტიპიური სიგრძე (ჩვეულებრივ 32 თანრიგი) საშუალებას გვაძლევს ბრძანებაში განვსაზღვროთ სამ რეგისტრამდე.

რეგისტრული არქიტექტურა უშვებს ოპერანდების განლაგებას ორიდან ერთ-ერთ დამახსოვრების არეში: ძირითად მეხსიერებაში ან რეგისტრებში. ოპერანდების შესაძლო განლაგების გათვალისწინებით რეგისტრული ბრძანებათა სისტემის არქიტექტურის ჩარჩოებში ყოფენ დამუშავების ბრძანებების სამ ქვესახეს:

- რეგისტრი - რეგისტრი;
- რეგისტრი - მეხსიერება;
- მეხსიერება - მეხსიერება.

„რეგისტრი-რეგისტრი“ ვარიანტში ოპერანდები შეიძლება იმყოფებოდნენ მხოლოდ რეგისტრებში. მათშივე იწერება შედეგიც. ქვეტიპი „რეგისტრი-მეხსიერება“, გულისხმობს, რომ ერთ-ერთი ოპერანდი მოთავსებულია რეგისტრში, ხოლო მეორე ძირითად მეხსიერებაში. შედეგი ჩვეულებრივ ცვლის ერთ-ერთ ოპერანდს. „მეხსიერება - მეხსიერება“ ტიპის ბრძანებებში ორივე ოპერანდი ინახება ძირითად მეხსიერებაში. შედეგი შეაქვთ მეხსიერებაში. თითოეულ ამ ვარიანტს გააჩნია თავისი უპირატესობები და ნაკლოვანებები (ცხრილი 2).

ცხრილის პირველ სვეტში მოტანილ (m,n) სახის გამოსახულებებში m აღნიშნავს ძირითად მეხსიერებაში შენახული ოპერანდების რაო-

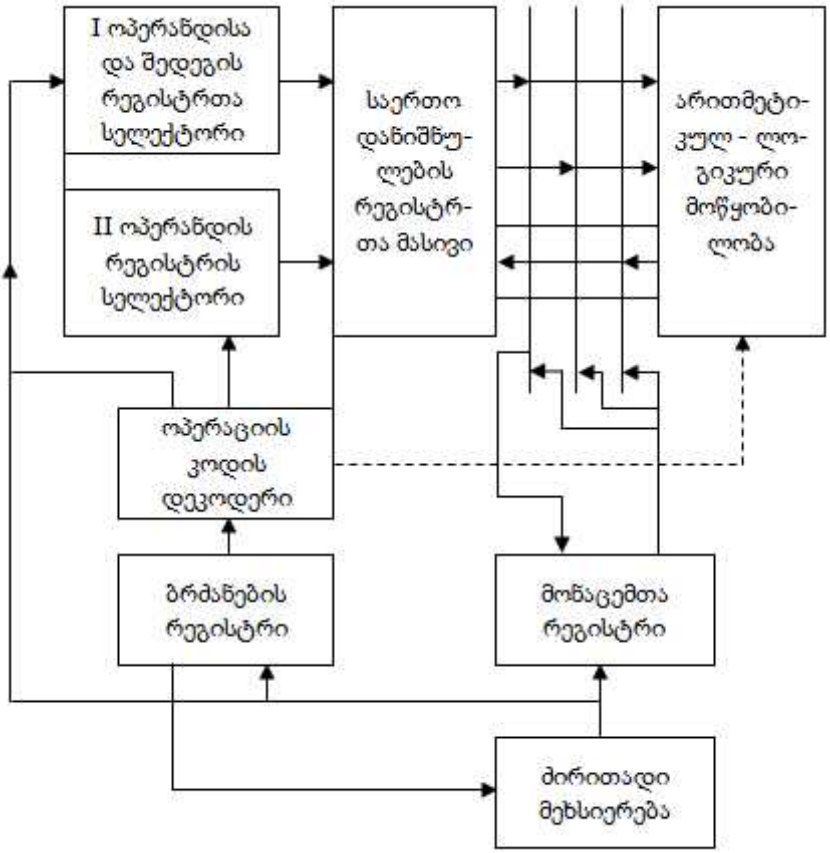
დენობას, ხოლო n - არითმეტიკული ან ლოგიკური დამუშავების ბრძანებაში ოპერანდების საერთო რიცხვს.

„რეგისტრი-რეგისრი“ ვარიანტი ძირითადია RISC ტიპის გამომთვლელ მანქანებში. „რეგისრი-მეხსიერება“ ტიპის ბრძანებები დამახასიათებელია CISC მანქანებისათვის. და ბოლოს, „მეხსიერება-მეხსიერება“ ვარიანტი არაეფექტურად ითვლება, თუმცა კი რჩება CISC კლასის მანქანების ყველაზე რთულ მოდელებში.

ცხრილი 2

ვარიანტი	უპირატესობები	ნაკლოვანებები
რეგისტრი - რეგისტრი (0, 3)	რეალიზაციის სიმარტივე, ბრძანებათა ფიქსირებული სიგრძე, პროგრამების კომპილაციის დროს ობიექტური კოდის ფორმირების მარტივი მოდელი, ყველა ბრძანების ტაქტების ერთნაირი რაოდენობის განმავლობაში შესრულების შესაძლებლობა.	ობიექტური კოდის დიდი სიგრძე, მოკლე ბრძანებების დროს თანრიგთა ნაწილი არ გამოიყენება ბრძანებათა ფიქსირებული სიგრძის გამო.
რეგისტრი - მეხსიერება (1, 2)	მონაცემები შეიძლება მიღწევადები იყვნენ პროცესორის რეგისტრებში ჩატვირთვის გარეშე, ბრძანებების კოდირების სიმარტივე, მიიღება საკმაოდ კომპაქტური ობიექტური კოდი.	შედეგის ჩაწერისას ერთ- ერთი ოპერანდის დაკარგვა, მეხსიერების მისამართის გრძელი ველი ბრძანების კოდში ამცირებს ადგილს რეგისტრის ნომრისათვის, რაც ზღუდავს სდრ-ს საერთო რაოდენობას. CPI დამოკიდებულია ოპერანდის მოთავსების ადგილზე.
მეხსიერება-მეხსიერება (3, 3)	ობიექტური კოდის კომპაქტურობა, საშუალედო მონაცემების დამახსოვრებისათვის მცირე მოთხოვნა რეგისტრებზე.	ბრძანების ფორმატებისა და მათი შესრულების დროის მრავალფეროვნება, მცირე სწრაფქმედება მეხსიერებაზე მიმართვის გამო.

ბრძანებათა სისტემის რეგისტრული არქიტექტურის გამომთვლელი მანქანის შესაძლო სტრუქტურა და ინფორმაციული ტრაქტები მოყვანილია ნახ. 2.5-ზე.



ნახ. 2.5. საერთო დანიშნულების რეგისტრების საფუძველზე აგებული გამომთვლელი მანქანის არქიტექტურა

მეხსიერებიდან რეგისტრების ჩატვირთვის ოპერაციები და მეხსიერებაში რეგისტრთა შემცველობის დამახსოვრება იდენტურია აკუმულატორთან იგივე ოპერაციებისა. განსხვავება მდგომარეობს საჭირო

რეგისტრის ამორჩევის ეტაპში, რომელიც უზრუნველყოფილია შესაბამისი სელექტორებით.

არითმეტიკულ-ლოგიკურ მოწყობილობაში ოპერაციათა შესრულება ითვალისწინებს:

- პირველი ოპერანდის რეგისტრის ამორჩევას;

- მეორე ოპერანდის განლაგების განსაზღვრას (მეხსიერება თუ რეგისტრი);

- არითმეტიკულ-ლოგიკურ მოწყობილობის შესასვლელზე ოპერანდების მიწოდებისა და ოპერაციის შესრულებას;

- შედეგის რეგისტრის ამორჩევას და მასში არითმეტიკულ-ლოგიკურ მოწყობილობიდან ოპერაციის შედეგის შეტანას.

მივაქციოთ ყურადღება იმას, რომ არითმეტიკულ-ლოგიკურ მოწყობილობასა და რეგისტრულ მასივს შორის სამი სალტე მაინც უნდა იყოს.

რეგისტრული ბრძანებათა სისტემის არქიტექტურის უპირატესობებს უნდა მივაკუთვნოთ: მიღებული კოდის კომპაქტურობა, გამოთვლათა მაღალი სისწრაფე, მიღწეული ძირითად მეხსიერებაზე მიმართვათა შეცვლით სწრაფ რეგისტრებზე მიმართვებით. მეორეს მხრივ, მოცემული არქიტექტურა მოითხოვს უფრო გრძელ ინსტრუქციებს აკუმულატორულ არქიტექტურასთან შედარებით.

საერთო დანიშნულების რეგისტრების საფუძველზე აგებული მანქანების ნიმუშებს მიეკუთვნებიან CDC 6600, IBM 360/370, PDP -11, ყველა თანამედროვე პერსონალური კომპიუტერი დღეისათვის ბრძანებათა სისტემის არქიტექტურის ეს სახე ყველაზე ფართოდაა გავრცელებული.

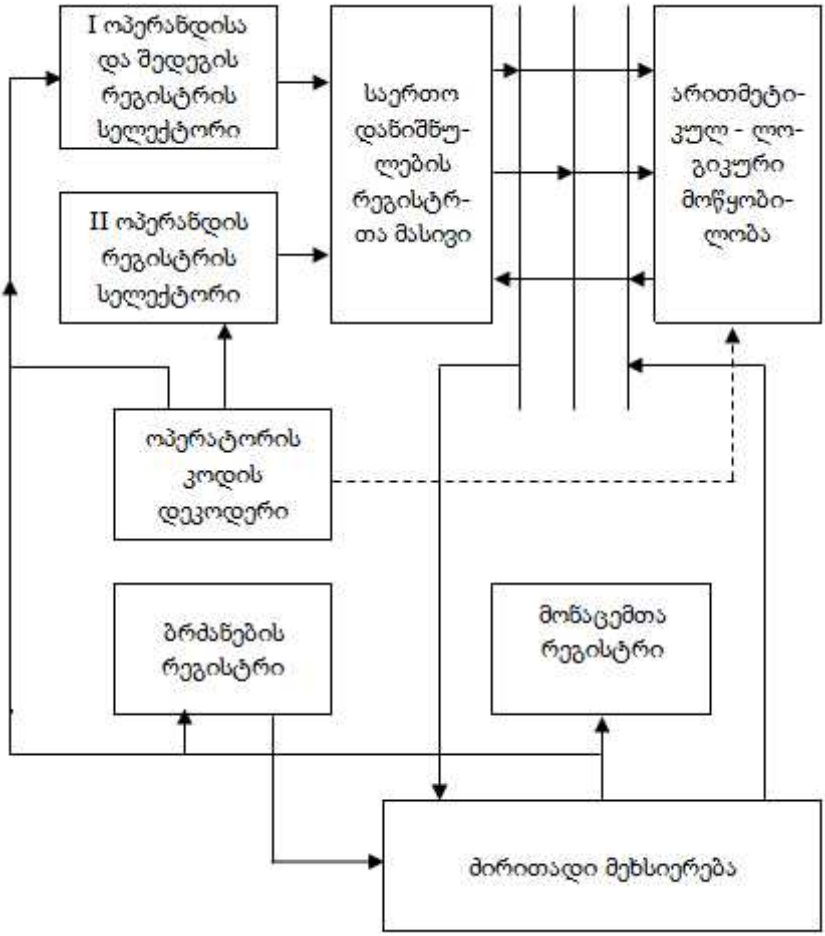
არქიტექტურა მეხსიერებასთან გამოყოფილი მიღწევით. არქიტექტურაში მეხსიერებასთან გამოყოფილი მიღწევით მიმართვა ძირითად მეხსიერებებზე შესაძლებელია მხოლოდ ორი სპეციალური ბრძანებით: load და store. ინგლისური ტრანსკრიფციით მოცემულ არქიტექტურას ეწოდება Load/Store architecture. ბრძანება Load (ჩატვირთვა) უზრუნველყოფს ძირითადი მეხსიერებიდან მნიშვნელო-

ბის ამოკითხვას და მის შეტანას პროცესორის რეგისტრში (ბრძანებაში ჩვეულებრივ ეთითება მეხსიერების უჯრედის მისამართი და რეგისტრის ნომერი). საწინააღმდეგო მიმართულებით ინფორმაციის გადაგზავნა ხორციელდება ბრძანებით store (დამახსოვრება). ოპერანდები ინფორმაციის დამუშავების ყველა ბრძანებაში შეიძლება იმყოფებოდნენ მხოლოდ პროცესორის რეგისტრებში (ყველაზე ხშირად საერთო დანიშნულების რეგისტრებში) ოპერაციის შედეგიც ასევე შეაქვთ რეგისტრში. არქიტექტურაში არ არის დამუშავების ისეთი ბრძანებები, რომლებიც უშვებენ ძირითად მეხსიერებაზე პირდაპირ მიმართვას. ბრძანებათა სისტემის არქიტექტურაში დასაშვებია ისეთ ბრძანებათა შეზღუდული რიცხვის არსებობა, რომლებშიც ოპერანდი ბრძანების კოდის ნაწილს წარმოადგენს.

მეხსიერებაზე გამოყოფილი მიღწევის მქონე გამომთვლელი მანქანის შემადგენლობა და ინფორმაციული ტრაქტები მოყვანილია ნახ. 2.6-ზე. საერთო დანიშნულების რეგისტრების მასივსა და არითმეტიკულ-ლოგიკურ მოწყობილობას შორის მოთავსებული სამიდან ორი სალტე უზრუნველყოფს საერთო დანიშნულების ორ რეგისტრში შენახული ოპერანდების გადაცემის არითმეტიკულ-ლოგიკურ მოწყობილობაში. მესამე სალტე ემსახურება შედეგების შეტანას ამისათვის გამოყოფილ რეგისტრში. ამ სალტეების მეშვეობით ხდება ძირითადი მეხსიერების უჯრედთა შემცველობის ჩატვირთვა რეგისტრებში და საერთო დანიშნულების რეგისტრების არსებული ინფორმაციის ძირითად მეხსიერებაში დამახსოვრება.

ბრძანებათა სისტემის არქიტექტურა მეხსიერებაზე გამოყოფილი მიღწევით დამახასიათებელია RISC - არქიტექტურის ყველა გამომთვლელი მანქანისათვის. ასეთ გამომთვლელ მანქანებში ბრძანებებს, როგორც წესი, 32 ბიტის ტოლი სიგრძე და სამმისამართიანი ფორმატი აქვთ. მეხსიერებაზე გამოყოფილი მიღწევის მქონე გამომთვლელი მანქანის მაგალითებად შეგვიძლია მოვიყვანოთ HP PA-RISC, IBM RS/6000, Sun SPARC, MIPS R4000, DEC Alpha და ა.შ. მეხსიერებებზე გამოყოფილი მიღწევის ბრძანებათა სისტემის არქიტექტურის უპირატესობებს შეიძ-

ლება მივაკუთვნოთ დეკოდირებისა და ბრძანებების შესრულების სიმარტივე.



ნახ. 2.6. მეხსიერებასთან გამოყოფილი მიღწევის მქონე გამომთვლელი მანქანის არქიტექტურა

2.2. ოპერანდთა ტიპები და ფორმატები

მანქანური ბრძანებები მონაცემებზე ოპერირებენ, რომლებსაც ამ შემთხვევაში მიღებულია ო პ ე რ ა ნ დ ე ბ ი ვუწოდოთ. ოპერანდების ყველაზე უფრო ზოგად (საბაზისო) ტიპებს შეიძლება მივაკუთვნოთ: მისამართები, რიცხვები, სიმბოლოები და ლოგიკური მონაცემები. მათ გარდა გამომთვლელი მანქანები უზრუნველყოფენ უფრო რთული ინფორმაციული ერთეულების დამუშავებას: გრაფიკული გამოსახულებების, აუდიო, ვიდეო და ანიმაციური ინფორმაციას, ასეთი ინფორმაცია წარმოებულია მონაცემთა საბაზისო ტიპებიდან და ფაილების სახით ინახება გარე დამახსოვრების მოწყობილობებზე. მანქანაში გათვალისწინებულია განსაზღვრული ფორმატები.

რიცხვითი ინფორმაცია

ციფრულ მონაცემებში შეიძლება გამოვყოთ ორი ჯგუფი:

- მთელი ტიპები, რომლებიც გამოიყენებიან მთელი რიცხვების წარმოდგენისათვის;

- ნამდვილი ტიპები რაციონალური რიცხვების წარმოსადგენად.

პირველი ჯგუფის ფარგლებში გვაქვს ციფრული ინფორმაციის წარმოდგენის რამდენიმე ფორმატი, დამოკიდებული მათ მახასიათებლებზე. აქ უპირველესად უნდა აღვნიშნოთ ფიქსირებული მძიმით წარმოდგენის ფორმა. აგრეთვე ათობითი რიცხვების ზონურ ფორმატში და შეფუთულ ფორმატში წარმოდგენის ფორმა. ნამდვილი რიცხვების წარმოდგენისათვის გამოიყენება მცურავი მძიმის ფორმა.

სიმბოლური ინფორმაცია

გამოთვლითი მოქმედების საერთო მოცულობაში სულ უფრო მეტი წილი მოდის სიმბოლური ინფორმაციის დამუშავებაზე, რომელიც შეიცავს ასოებს, ციფრებს, სასვენ ნიშნებს, მათემატიკურ და სხვა

სიმბოლოებს. თითოეულ სიმბოლოს აკუთვნებენ შესაბამის განსაზღვრულ ორობით კომბინაციას. შესაძლო სიმბოლოთა და მათზე მიკუთვნებულ ორობით კოდთა ერთობლიობა ქმნიან კ ო დ ი რ ე ბ ი ს ცხრილს. დღეისათვის გამოიყენება კოდირების მრავალი სხვადასხვა ცხრილი. მათ აერთიანებს წონათა პრინციპი, რომლის დროსაც ციფრთა წონები იზრდებიან ციფრის ზრდის მიხედვით, ხოლო სიმბოლოთა წონები ალფავიტური რიგით იზრდებიან. ასე მაგალითად, „B“ ასოს წონა ერთით მეტია „A“ ასოს წონაზე. ეს ხელს უწყობს გამომთვლელ მანქანაში დამუშავების გამარტივებას.

დღეისათვის ფართოდაა გავრცელებული ASCII და Unicode სტანდარტები. Unicode სტანდარტი უკვე შესატყისობაშია ASCII კოდირებასთან. პირველად Unicode გამოყენებულ იქნა ოპერაციულ სისტემა Windows NT-ში.

ლოგიკური მონაცემები

ლოგიკური მონაცემების ელემენტს წარმოადგენს ლოგიკური (ბულის) ცვლადი, რომელმაც შეიძლება მიიღოს მხოლოდ ორი მნიშვნელობა: ჭეშმარიტი“ ან „მცდარი“. ლოგიკური მნიშვნელობის კოდირება მიღებულია განხორციელდეს ინფორმაციის ბიტით: ერთიანით ახდენენ ჭეშმარიტი მნიშვნელობის კოდირებას, ხოლო ნულით - მცდარის, როგორც წესი, გამომთვლელ მანქანაში ოპერირებენ მანქანური სიტყვის სიგრძის ლოგიკური ცვლადების ნაკრებებით. ასეთი სიტყვები მუშავდებიან ლოგიკურ ოპერაციათა (და, ან, არა და ა.შ.) ბრძანებების დახმარებით, ამასთან ყველა ბიტი ერთნაირად მუშავდება, მაგრამ ერთმანეთისაგან დამოუკიდებლად, ე. ი. თანრიგებს შორის არავითარი გადატანები არ ხდება.

სტრიქონები

ს ტ რ ი ქ ო ნ ე ბ ი ბიტების, ბაიტების, სიტყვებისა და ორმაგი სიტყვების უწყვეტი მიმდევრობაა. ბ ი ტ უ რ ი ს ტ რ ი ქ ო ნ ი შეიძ-

ლება იწყებოდეს ბაიტის ნებისმიერი პოზიციიდან და შეიცავდეს 2^{32} - მდე ბიტს. ბ ი ტ უ რ ი ს ი ტ ყ ვ ა შეიძლება შედგებოდეს ბაიტების, სიტყვებისა ან ორმაგი სიტყვებისაგან. ასეთი სტრიქონის სიგრძე იცვლება ნულიდან 2^{32-1} ბაიტამდე (4 გბაიტი). მოყვანილი ციფრები დღეისათვის დამახასიათებელია 32-თარიგიანი გამომთვლელი მანქანებისათვის.

თუ ბიტური სტრიქონის ბაიტები წარმოადგენენ სიმბოლოთა კოდებს, მაშინ საუბრობენ ტ ე ქ ს ტ უ რ ს ტ რ ი ქ ო ნ ზ ე. ვინაიდან ტექსტური სტრიქონის სიგრძე შეიძლება იცვლებოდეს ძალიან ფართო საზღვრებში, ამიტომ სტრიქონის დასასრულსის მისათითებლად ბოლო ბაიტში შეაქვთ კოდის შემზღუდველი - ჩვეულებრივ ეს ნულებია ბაიტის ყველა თანრიგში.

ინფორმაციის სხვა სახეები

გამომთვლელ მანქანაში წარმოდგენილი ინფორმაცია შეიძლება სტატიკური ან დინამიური იყოს. რიცხვითი, სიმბოლური და ლოგიკური ინფორმაცია სტატიკურია - მისი მნიშვნელობა არაა დამოკიდებული დროზე. პირიქით, აუდიოინფორმაციას დინამიური ხასიათი აქვს - არსებობს მხოლოდ დროის რეალურ რეჟიმში და არ შეიძლება შეჩერებულ იქნას უფრო დაწვრილებითი შესწავლისათვის. თუ დროის მასშტაბს შევცვლით, მაშინ აუდიოინფორმაცია მახინჯდება, რაც გამოიყენება, მაგალითად, ხმოვანი ეფექტების შესაქმნელად.

ვ ი დ ე ო ი ნ ფ ო რ მ ა ც ი ა. ვიდეოინფორმაცია არის როგორც სტატიკური, ისე დინამიური. სტატიკური ინფორმაცია შეიცავს ტექსტს, ნახატებს, გრაფიკებს, ნახაზებს და სხვა.

დინამიური ვიდეოინფორმაცია - ესაა ვიდეო -, მულტ - და სლაიდ - ფილმები. მათ საფუძვლად უდევთ ეკრანზე დროის რეალურ მომენტში სცენარის შესაბამისად ცალკეულ კადრების მიმდევრობითი ექსპონირება. დინამიური ინფორმაცია გამოიყენება ან მოძრავი გამოსა-

ხულების გადმოსაცემად (ანიმაცია), ანდა ცალკეული კადრების მიმდევრობითი დემონსტრაციისათვის (სლაიდ - ფილმები).

გამოთვლით ტექნიკაში არსებობს გრაფიკული გამოსახულებების წარმოდგენის ორი ხერხი: მ ა ტ რ ი ც უ ლ ი (რ ა ს ტ რ უ ლ ი) და ვ ე ქ ტ ო რ უ ლ ი. მატრიცული (bitmap) ფორმატები კარგად ესადაგებიან ისეთი გამოსახულებების წარმოდგენას, რომლებსაც ახასიათებს ფერთა გამების ელფერებისა და ფორმათა სირთულე, ისეთები როგორებიცაა ფოტოგრაფიები, ნახატები, გადასკანირებული მონაცემები. ვექტორული ფორმატები უფრო მორგებულია ნახაზებისა და მარტივ ფორმებიან ჩრდილებით და შეფერადებით შესრულებული გამოსახულებების გამოსასახავად.

მატრიცულ ფორმატებში გამოსახულება წარმოიდგინება წერტილთა - პ ი ქ ს ე ლ თ ა (picture element) მართკუთხა მატრიცით. მატრიცაში პიკსელთა მდებარეობა შეესაბამება ეკრანზე წერტილთა კოორდინატებს.

ვექტრული წარმოდგენა, მატრიცული გრაფიკისაგან განსხვავებით, განსაზღვრავს გამოსახულების აღწერას არა პიქსელებით, არამედ მრუდებით - ს პ ლ ა ი ნ ე ბ ი თ. სპლაინი გლუვი მრუდია, რომელიც გადის ორ ან მეტ საყრდენ წერტილზე, რომლებიც მართავენ სპლაინის ფორმას. ვექტრულ გრაფიკაში ყველაზე მეტადაა გავრცელებული ბეზიეს სპლაინები.

ვექტრული გრაფიკის ძირითადი უპირატესობაა ობიექტის მარტივი აღწერა და მეხსიერებაში მცირე მოცულობის დაკავება. გარდა ამისა, ვექტრულ გრაფიკას მატრიცულთან შედარებით შემდეგი უპირატესობები გააჩნია:

- გამოსახულების მასშტაბირების სიმარტივე მისი ხარისხის გაუარესების გარეშე;

- გამოსახულების შენახვისათვის საჭირო მეხსიერების მოცულობის ამორჩეული ფერადი მოდელისაგან დამოუკიდებლობა.

ვექტრული გამოსახულების ნაკლოვანებას წარმოადგენს მისი რაღაც ხელოვნურობა, რაც იმაში მდგომარეობს, რომ ნებისმიერი გამოსახულება საჭიროა დაიყოს მის შემადგენელ სასრულო პრიმიტივებად.

მატრიცული და ვექტრული გრაფიკა ერთმანეთისაგან განცალკევებულად არ არსებობენ. ასე, მაგალითად, ვექტრული ნახატები შეიძლება შეიცავდნენ მატრიცულ გამოსახულებებსაც. გარდა ამისა, ვექტორული და მატრიცული გამოსახულებები შეიძლება ერთმანეთში გარდაიქმნან. ფორმატები, რომლებიც საშუალებას იძლევიან შევათავსოთ გამოსახულების მატრიცული და ვექტრული აღწერა, იწოდებიან მ ე ტ ა ფ ა ი ლ ე ბ ა დ . მეტაფაილები უზრუნველყოფენ ფაილების საკმარის კომპაქტურობას გამოსახულებას მაღალი ხარისხის შენარჩუნებით.

ა უ დ ი ო ი ნ ფ ო რ მ ა ც ი ა. ცნება ა უ დ ი ო დაკავშირებულია ბგერასთან, რომელიც შეძლება აღითქვას ადამიანის ყურმა (15ჰც-20კჰც). გამომთვლელ მანქანაში წარმოდგენამდე ა უ დ ი ო ი ნ ფ ო რ მ ა ც ი ა უნდა გარდაიქმნას ციფრულ ფორმაში. ეს ხორციელდება ანალოგიურ-ციფრული გარდამქმნელით. საწინააღმდეგო ქმედება კი ციფრულ-ანალოგიური გარდამქმნელით. რაც უფრო ხშირად ხდება სიგნალთა ამოკრეფა (უწყვეტი სიგნალის დაყოფა) მით უფრო მაღალია საწყისი სიგნალის გადმოცემის სიზუსტე, მით უფრო მაღალია საწყისი სიგნალის გადმოცემის სიზუსტე, მაგრამ მეხსიერების მით მეტი ტევადობაა საჭირო ციფრულ სახეში გარდაქმნილი ბგერების შესანახად.

აუდიო სიგნალების ციფრული ექვივალენტი ჩვეულებრივ ინახება ფაილების სახით, ამასთან ფართოდ გამოიყენება ასეთი ინფორმაციის შეკუმშვის სხვადასხვა საშუალება.

2.3. ბრძანებათა ტიპები

მიუხედავად სხვადასხვა გამომთვლელ მანქანებში ბრძანებათა სისტემის განსხვავებისა, ოპერაციათა ზოგიერთი ძირითადი ტიპი შეიძლება ნაპოვნი იყოს ნებისმიერ მათგანში. ამ ტიპებისათვის მიღებულია შემდეგი კლასიფიკაცია:

- მონაცემთა გადაგზავნის ბრძანებები;
- არითმეტიკული და ლოგიკური დამუშავების ბრძანებები;
- სტრიქონებთან მუშაობის ბრძანებები;
- SIMD ბრძანებები;
- შეტანა/გამოტანის ბრძანებები;
- ბრძანებათა ნაკადის მართვის ბრძანებები.

2.4. ბრძანებათა ფორმატები

საერთო შემთხვევაში ტიპიური ბრძანება უნდა უთითებდეს:

- შესასრულებელ ბრძანებას;
- საწყისი მონაცემების (ოპერანდების) მისამართებს, რომლებზეც უნდა შესრულდნენ ოპერაციები;
- მისამართს, რომელზეც უნდა მოთავსდეს ოპერაციის შედეგი.

ამის შესაბამისად ბრძანება შესდგება ორი ნაწილისაგან: ოპერაციულისა და სამისამართოსაგან (ნახ. 2.7):

ოპერაციული ნაწილი	სამისამართო ნაწილი

ნახ. 2.7. ბრძანების სტრუქტურა

ბრძანების ფორმატი განსაზღვრავს მის სტრუქტურას, ანუ მთელ ბრძანებაზე გამოყოფილი ორობითი თანრიგების რაოდენობას, აგრეთვე ბრძანების ცალკეული ველების რაოდენობასა და განლაგებას. **ველი** ეწოდება ორობით თანრიგთა რაოდენობას, რომლებიც

ახორციელებენ ბრძანების შედგენილი ნაწილის კოდირებას. შესაძლო ფორმატების შეფასებებისას საჭიროა გავითვალისწინოთ შემდეგი ფაქტორები:

- განსხვავებულ ბრძანებათა საერთო რიცხვი;
- ბრძანების საერთო სიგრძე;
- ბრძანების ველთა ტიპები (ფიქსირებული ან ცვლადი სიგრძის) და მათი სიგრძე;
- დეკოდირების სიმარტივე;
- გადამისამართება და მისი ხერხები;
- დეკოდირებისა და ბრძანებების შესრულებისათვის საჭირო მოწყობილობათა ღირებულება.

ერთი გამომთვლელი მანქანის ბრძანებათა სისტემაში შეიძლება გამოიყენებოდნენ ბრძანებათა სხვადასხვა ფორმატები. ჩვეულებრივ ეს დაკავშირებულია გადამისამართების სხვადასხვა ხერხის გამოყენებასთან. ამ შემთხვევაში ბრძანების კოდის შემადგენლობაში შეჰყავთ ველი გადამისამართების ხერხის მისათითებლად და ბრძანების განზოგადებული ფორმატი იღებს ნახ. 2.8-ზე მოცემულ სახეს.

ბრძანების ოპერაციის კოდი	გადამისამართების ხერხი	სამისამართო ნაწილი

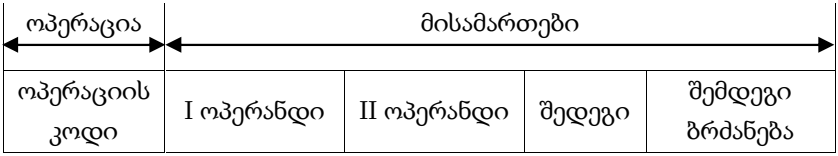
ნახ. 2.8. ბრძანების განზოგადებული ფორმატი

გამომთვლელი მანქანის უმრავლესობაში ერთდროულად თავსდება ბრძანებათა რამდენიმე განსხვავებული ფორმატი. უნდა აღინიშნოს, რომ ბრძანების სიგრძე უმნიშვნელოვანესი მახასიათებელია, რომელიც ზემოქმედებს მეხსიერების ორგანიზაციასა და მოცულობაზე, სალტეთა სტრუქტურაზე, ცენტრალური პროცესორის სირთულესა და სწრაფქმედებაზე.

ბრძანებაში მისამართების რაოდენობა

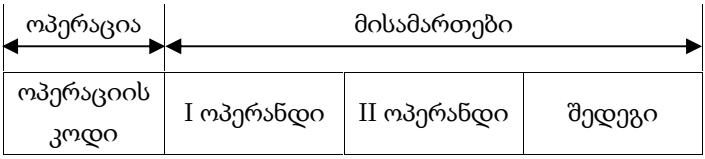
სამისამართო ნაწილში ჩართული მისამართების რაოდენობის განსაზღვრისათვის გამოვიყენებთ ტერმინს მ ი ს ა მ ა რ თ ი ა ნ ო ბ ა.

„მაქსიმალურ ვარიანტში“ აუცილებელია სამი კომპონენტის მითითება: პირველი ოპერანდის მისამათის, მეორე ოპერანდის მისამართის და იმ უჯრედის მისამართის, სადაც ჩაიწერება ოპერაციის შედეგი. პრინციპში შეიძლება დამატებული იყოს კიდევ ერთი მისამართი, რომელიც უთითებს მომდევნო ინსტრუქციის შენახვის ადგილს. შედეგად ადგილი აქვს ბრძანების ოთხმისამართიან ფორმატს (ნახ. 2.9). ასეთი ფორმატი გამოიყენება გასული საუკუნის 40-იან წლებში დამუშავებულ გამომთვლელ მანქანაში EDVAC.



ნახ. 2.9. ბრძანების ოთხმისამართიანი ფორმატი

ფონ-ნეიმანისეულ გამომთვლელ მანქანაში მეოთხე მისამართი აუცილებელი აღარაა, ვინაიდან მეხსიერებაში ბრძანებები განლაგებულია მისი შესრულების რიგის მიხედვით, და მომდევნო ბრძანების მისამართი შეიძლება მიღებულ იქნას ბრძანებათა მთვლელში მიმდინარე ბრძანების მისამართის უბრალო გაზრდის ხარჯზე (ნახ. 2.10). მხოლოდ საჭიროა გამომთვლელი მანქანის ბრძანებათა სისტემაში დაემატოს ბრძანებები, რომლებსაც შეუძლიათ შეცვალონ გამოთვლების რიგი (თანმიმდევრობა).

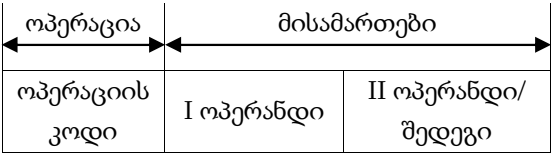


ნახ. 2.10. ბრძანების სამმისამართიანი ფორმატი

სამწუხაროდ, სამმისამართიან ფორმატშიც ბრძანების სიგრძე შეიძლება აღმოჩნდეს ძლიერ დიდი. ასე მაგალითად, თუ ძირითადი მეხ-

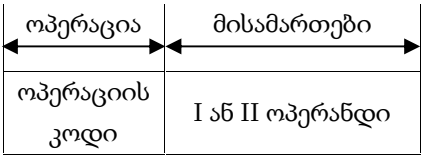
სიერების უჯრედის მისამართს აქვს 32 ბიტის ტოლი სიგრძე, ხოლო ოპერაციის კოდი 8 ბაიტია, მაშინ ბრძანების სიგრძე 104 ბიტი (13 ბაიტი) იქნება.

თუ მითითების გარეშე შედეგის მისამართად ავიღებთ ერთ-ერთი ოპერანდის მისამართის (ჩვეულებრივ მეორის), მაშინ შეიძლება მესამე მისამართის გარეშეც ვიმუშაოთ და შედეგად მივიღებთ ბრძანების ორმისამართიან ფორმატს (ნახ. 2.11). ბუნებრივია, რომ ამ შემთხვევაში შესაბამისი ოპერანდი ოპერაციის შესრულების შემდეგ იკარგება.



ნახ. 2.11. ბრძანების ორმისამართიანი ფორმატი

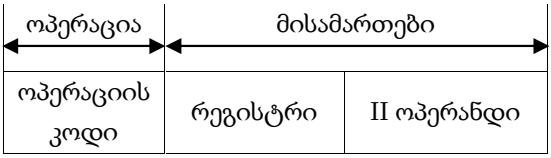
ბრძანება შეიძლება კიდევ უფრო შევკვეცოთ, თუ გადავალთ ერთმისამართიან ფორმატზე (ნახ. 2.12), რაც შესაძლებელია პირველი ოპერანდისა და შედეგის შესანახად განსაზღვრული სტანდარტული ადგილის გამოყოფით. ჩვეულებრივ ამ მიზნისათვის გამოიყენება ცენტრალური პროცესორის სპეციალური რეგისტრი, ცნობილი ა კ უ მ უ ლ ა ტ ო რ ი ს სახელით, ვინაიდან აქ აკუმულირდება შედეგი.



ნახ. 2.12. ბრძანების ერთმისამართიანი ფორმატი

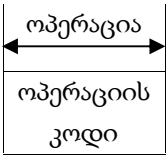
ერთ-ერთი ოპერანდისა და შედეგის შესანახად ერთადერთი რეგისტრის გამოყენება წარმოადგენს შემზღუდავ ფაქტორს, ამიტომ აკუმულატორის გარდა ხშირად იყენებენ ცენტრალური პროცესორის სხვა რეგისტრებსაც. ვინაიდან ცენტრალურ პროცესორში რეგისტრების რი-

ცხვი დიდი არაა, ამიტომ ერთ-ერთი მათგანის მისათითებლად საკმარისია ბრძანებაში გვქონდეს შედარებით მოკლე სამისამართო ველი. შესაბამის ფორმატს ერთნახევარმისამართიანი ან რეგისტრული ფორმატი ჰქვია (ნახ. 2.13):



ნახ. 2.13. ბრძანების ერთნახევარმისამართიანი ფორმატი

და ბოლოს, თუ ორივე ოპერანდისათვის მივუთითებთ მკაცრად განსაზღვრულ ადგილს, აგრეთვე ისეთი ბრძანებებისათვის, რომელთაც ოპერანდი არ სჭირდებათ, შეიძლება მივიღოთ ბრძანების ნულმისამართიანი ფორმატი (ნახ. 2.14). ასეთ ვარიანტში ბრძანების სამისამართო ნაწილი საერთოდ არ არის ან არ მოქმედებს.



ნახ. 2.14. ბრძანების ნულმისამართიანი ფორმატი.

ბრძანებების დამისამართების არჩევა

ბრძანების სამისამართო ნაწილში მისამართების რაოდენობის არჩევისას ჩვეულებრივ ხელმძღვანელობენ შემდეგი კრიტერიუმებით:

- პროგრამის შესანახად საჭიროა მეხსიერების მოწყობილობის ტევადობით;

- პროგრამის შესრულების დროით;

-პროგრამის შენახვისას მეხსიერების უჯრედების ეფექტური გამოყენებით.

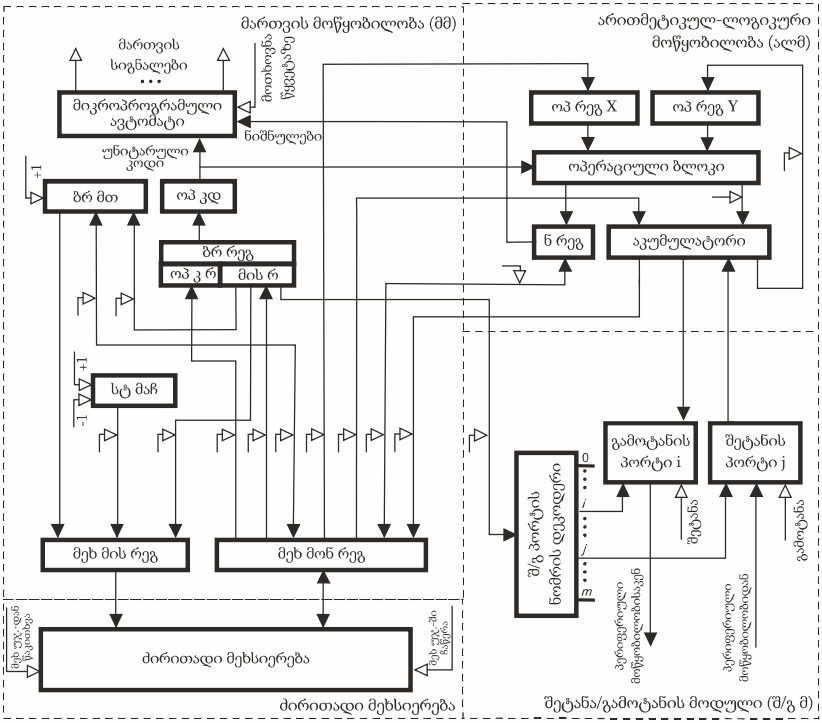
ოპერაციათა სისტემა

ოპერაციათა სისტემა ჰქვია გამომთვლელი მანქანის ტექნიკური საშუალებებით უშუალოდ შესრულებულ ოპერაციათა სიას. გამომთვლელი მანქანის ოპერაციათა სისტემა განისაზღვრება მისი გამოყენების ზონით, მოთხოვნებით ღირებულებაზე, წარმადობითა და გამოთვლების სიზუსტით.

3. ფონ ნეიმანისეული გამომთვლელი მანქანის ფუნქციური ორგანიზაცია

3.1. ფონ ნეიმანისეული გამომთვლელი მანქანის ფუნქციური სქემა

გამომთვლელი მაქანის სტრუქტურისა და მოწყობილობების ფუნქციების შესახებ დეტალური წარმოდგენის მიღების მიზნით განვიხილოთ აკუმულატორული არქიტექტურის ჰიპოთეტური მანქანის სქემა (ნახ. 3.1). გადმოცემის გასამარტივებად მიღებულია მანქანის შემდეგი მახასიათებლელი:



ნახ. 3.1. ფონ ნეიმანისეული გამომთვლელი მანქანის ფუნქციური სქემა

- ე რ თ მ ი ს ა მ ა რ თ ი ა ნ ი ბ რ ძ ა ნ ე ბ ე ბ ი. ბრძანებების სამისამართო ნაწილი შეიცავს მხოლოდ ერთ მისამართს. ორ ოპერანდთან ოპერაციების შესრულებისას იგულისხმება, რომ მეორე ოპერანდი იმყოფება არითმეტიკულ-ლოგიკური მოწყობილობის სპეციალურ რეგისტრში აკუმულატორში, ხოლო შედეგიც ასევე აკუმულატორში რჩება.

- ფ ო რ მ ა ტ ე ბ ი ს ე რ თ ი ა ნ ო ბ ა. ბრძანებებისა და მონაცემების სიგრძე ემთხვევა მეხსიერების უჯრედთა მისამართებს, ანუ ნებისმიერი ბრძანება ან ოპერანდი იჭერს მეხსიერების მხოლოდ ერთ უჯრედს. ამრიგად, მეხსიერებაში ბრძანების მომდევნო მისამართი შეიძლება მიღებულ იქნას მიმდინარე ბრძანების მისამართზე ერთიანის მიმატების გზით, ხოლო მეხსიერებიდან ნებისმიერი ბრძანების ან ოპერანდის მისაღებად საკმარისია მისამართზე ერთი მიმართვა.

ფუნქციურ სქემაზე (ნახ. 3.1) ნაჩვენებია გამომთვლელი მანქანის ნებისმიერი ძირითადი მოწყობილობის ტიპიური კვანძები, აგრეთვე სიგნალები, რომლებიც ახდენენ მანქანის ფუნქციონირებისათვის აუცილებელი ინფორმაციის გადაგზავნასა და მისი დამუშავების ცალკეული ოპერაციების შესრულების ინიცირებას.

3.1.1. მართვის მოწყობილობა

როგორც ადრე იყო აღნიშნული, მართვის მოწყობილობა გამომთვლელი მანქანის ნაწილია, რომელიც ახორციელებს პროგრამების ავტომატურ შესრულებას და გამომთვლელი მანქანის, როგორც ერთიანი სისტემის, ფუნქციონირებას. ეხლა შევჩერდეთ მართვის მოწყობილობის კვანძების აღწერაზე.

ბრძანებათა მთვლელი

ბ რ ძ ა ნ ე ბ ა თ ა მ თ ვ ლ ე ლ ი (ბრ მთ) ფონ ნეიმანის პროგრამული მართვის პრინციპის შესაბამისად აგებული ნებისმიერი გამომთვლელი მანქანის მართვის მოწყობილობის განუყოფელი ელემენტია. ამ პრინციპის შესაბამისად, პროგრამის მეზობელი ბრძანებები მოთავსებულნი არიან მისამართების თანმიმდევრული რიგით მეხსიერების უჯრედში და უპირატესად სრულდებიან იგივე რიგითობით, როგორითაც არიან მოთავსებულნი გამომთვლელი მანქანის მეხსიერებაში. ამრიგად, მომდევნო ბრძანების მისამართი შეიძლება მიღებული იყოს იმ უჯრედის მისამართის გაზრდით, რომლიდანაც იყო ამოკითხული მიმდინარე ბრძანება. ასეთი რეჟიმის რეალიზაციის უზრუნველყოფა წარმოადგენს ბრძანებითი მთვლელის მიზანს. ბრძანებათა მთვლელი ოროზითი მთვლელია, რომელშიც ინახება და მოდიფიცირდება პროგრამის მომდევნო ბრძანების მისამართი. გამოთვლების დაწყების წინ ბრძანებათა მთვლელში შეიტანება ძირითადი მეხსიერების უჯრედის ის მისამართი, რომელშიც ინახება ბრძანება, რომელიც პირველი უნდა შესრულდეს. თითოეული ბრძანების შესრულების პროცესში ბრძანებათა მთვლელის მნიშვნელობის გაზრდით მთვლელში ფორმირდება მომდევნო შესასრულებელი ბრძანების მისამართი. განსახილველ გამომთვლელ მანქანაში ნებისმიერი ბრძანება იჭერს ერთ უჯრედს, ამიტომ ბრძანებათა მთვლელის შემცველობა იზრდება ერთი ერთეულით, რაც უზრუნველყოფილია ბრძანებათა მთვლელზე მართვის +1 სიგნალის მიწოდებით. მიმდინარე ბრძანების დასრულების შემდეგ პროგრამის მომდევნო ბრძანების მისამართი ყოველთვის იღება ბრძანებათა მთვლელიდან. გამოთვლების ბუნებრივი რიგის შესაცვლელად (პროგრამის სხვა წერტილში გადასვლა) საკმარისია ბრძანებათა მთვლელში გადასვლის წერტილის მისამართის შეტანა.

თუმცა ტერმინი „ბრძანებათა მთვლელი“ მიღებულია, ის არ შეიძლება ჩაითვალოს მისაღებად იმის გამო, რომ ქმნის არასწორ წარმოდგენას მოცემული კვანძის ამოცანების შესახებ. ამ მიზეზის გამო გამომ-

თვლელი მანქანის დამმუშავებლები იყენებენ სხვა დასახელებებს, კერძოდ პროგრამულ მთვლელს (PC, Program counter) ან ბრძანების მიმთითებელს (IP, Instruction Pointer). ბოლო განსაზღვრება, ჩვენი აზრით, საუკეთესოდ ასახვავს მართვის მოწყობილობის განსახილველი კვანძის დანიშნულებას.

ბოლოს აღვნიშნოთ, რომ რიგ გამომთვლელ მანქანებში ბრძანებათა მთვლელი განხორციელებულია ჩვეულებრივი რეგისტრის სახით, ხოლო მისი შემცველობის გაზრდა ხორციელდება გარე სქემით (ინკრიმენტ/დეკრემენტის სქემით).

ბრძანების რეგისტრი

ბრძანებების მთვლელი განსაზღვრავს მეხსიერებაში ბრძანების მხოლოდ ადგილმდებარეობას, მაგრამ არ შეიცავს ინფორმაციას თავად ბრძანების შესახებ. ბრძანების შესრულების დაწყებისათვის აუცილებელია მისი მეხსიერებიდან ამოღება და ბრძანების რეგისტრში მოთავსება. ამ ეტაპს ბრძანების ამორჩევა ჰქვია. მხოლოდ ბრძანების ბრძანების რეგისტრში ჩატვირთვის მომენტიდან ხდება ის „ხილული“ პროცესორისათვის. ბრძანების რეგისტრში ბრძანება ინახება მისი შესრულების მთელი დროის განმავლობაში. როგორც ადრე იყო აღნიშნული, ნებისმიერი ბრძანება შეიცავს ორ ველს: ოპერაციის კოდის ველს და სამისამართო ნაწილის ველს. ამ მდგომარეობის გათვალისწინებით ბრძანების რეგისტრს ზოგჯერ იხილავენ როგორც ორი რეგისტრის ერთობლიობას - ოპერაციის კოდის რეგისტრის და მისამართის რე- გისტრის, რომლებშიც ინახებიან ბრძანების შესაბამისი შემადგენლები.

თუ ბრძანებას უჭირავს რამდენიმე მიმდევრობითი უჯრედი, მაშინ ოპერაციის კოდი ყოველთვის იმყოფება ბრძანების იმ სიტყვაში, რომელიც მეხსიერებიდან პირველი ამოიღება. ეს საშუალებას იძლევა ოპერაციის კოდით განისაზღვროს ესაჭიროება თუ არა ბრძანების და-

ნარჩენ სიტყვებს მეხსიერებიდან ამოკითხვა და ბრძანების რეგისტრში ჩატვირთვა. თავად ბრძანების შესრულება იწყება ბრძანების რეგისტრში მისი სრული კოდის შეტანის შემდეგ.

სტეკის მაჩვენებელი

სტეკის მაჩვენებელი (სტ მაჩ) - ეს რეგისტრია, სადაც ინახება სტეკის წვეროს მისამართი. რეალურ გამომთვლელ მანქანებში სტეკი განხორციელებულია ძირითადი მეხსიერების ნაწილის სახით, რომელიც ჩვეულებრივ განლაგებულია უდიდესი მისამართების ზონაში. სტეკის შევსება ხდება მისამართების შემცირების მხარეს, ამასთან სტეკის წვერო ეს უჯრედია, რომელშიც მოხდა დროში ბოლო ჩაწერა. ასეთი მისამართის შესანახადაა სწორედ გათვალისწინებული სტეკის მაჩვენებელი. ოპერაცია push - ის (სტეკში შეტანა) შესრულებისას სტეკის მაჩვენებლის შემცველობა -1 სიგნალის დახმარებით მცირდება ერთით, რის შემდეგაც გამოიყენება მისამართად, რომლის მიხედვითაც წარმოებს ჩაწერა. შესაბამისი უჯრედი ხდება სტეკის ახალი წვერო. სტეკიდან ამოკითხვა (ოპერაცია pop) წარმოებს იმ უჯრედიდან, რომელზეც მიუთითებს მიმდინარე მისამართი სტეკის მაჩვენებელში, რის შემდეგაც სტეკის მაჩვენებლის შემცველობა +1 სიგნალით იზრდება ერთი ერთეულით. ამრიგად, სტეკის წვერო ეშვება, ხოლო ამოკითხული სიტყვა ითვლება სტეკიდან წაშლილად. თუმცა ფიზიკურად წაკითხული სიტყვა დარჩა მეხსიერების უჯრედში, მაგრამ სტეკში ჩაწერისას ის ახალი ინფორმაციით შეიცვლება.

მეხსიერების მისამართის რეგისტრი

მეხსიერების მისამართის რეგისტრის დანიშნულებაა ძირითადი მეხსიერების უჯრედის მისამართის შენახვა ამ უჯრედთან ოპერაციის დასრულებამდე (წაკითხვის ან ჩაწერის). მეხსი-

ერების მისამართის რეგისტრის არსებობა საშუალებას იძლევა მოხდეს ძირითადი მეხსიერებებისა და მანქანის სხვა მოწყობილობების სწრაფქმედებაში განსხვავების კონპენსაცია.

მეხსიერების მონაცემთა რეგისტრი

მეხსიერების მონაცემთა რეგისტრის დანიშნულებაა სწრაფქმედებაში განსხვავების კონპენსირება მეხსიერების მოწყობილობებსა და იმ მოწყობილობებს შორის, რომლებიც გამოდიან შენახული ინფორმაციის წყაროსა და მომხმარებელთა როლში. წაკითხვის პროცესში მეხსიერების მონაცემთა რეგისტრში ხდება ძირითადი მეხსიერების უჯრედში შემცველობის შეტანა, ჩაწერის დროს კი - თავსდება ძირითადი მეხსიერების უჯრედში შესანახი ინფორმაცია. თავად უჯრედში ჩაწერისა და წაკითხვის მომენტი განისაზღვრება შესაბამისად ჩაწერისა და წაკითხვის სიგნალებით.

ოპერაციის კოდის დეკოდერი

ოპერაციის კოდის დეკოდერი გარდაქმნის ოპერაციების კოდს მიკროპროგრამულ ავტომატთან სამუშაოდ საჭირო ფორმაში. დეკოდირების შემდეგ ინფორმაცია განსაზღვრავს მიკროპროგრამული ავტომატის შემდგომ მოქმედებებს, ხოლო მისი სახე დამოკიდებულია მიკროპროგრამული ავტომატის ორგანიზაციისაგან. განხილულ გამომთვლელ მანქანაში - ეს უნიტარული კოდია. ხშირად ოპერაციის კოდი გარდაიქმნება მიკროპროგრამის პირველი ბრძანების მისამართში, რომელიც ახორციელებს ბრძანებაში მითითებულ ოპერაციას. ამ პოზიციებიდან უფრო სწორი იქნებოდა ოპერაციის კოდის დეკოდერისათვის გვეწოდებინა არა დეკოდერი, არამედ კოდების გარდამქმნელი.

მიკროპროგრამული ავტომატი

მართებულია მიკროპროგრამული ავტომატი ჩავთვალოთ მართვის მოწყობილობის ცენტრალურ კვანძად. სწორედ მიკროპროგრამული ავტომატი ახდენს მართვის სიგნალების თანმიმდევრობის ფორმირებას, რომელთა შესაბამისადაც ხორციელდება მეხსიერებიდან ამორჩევისა და ბრძანებების შასასრულებლად აუცილებელი ყველა მოქმედება. მიკროპროგრამული ავტომატისათვის საწყის ინფორმაციას წარმოადგენს: ოპერაციის დეკოდირებული კოდი, წინა გამოთვლების შედეგის დამახასიათებელ ნიშანთა (ალმების) მდგომარეობა, აგრეთვე გარე მოთხოვნები მიმდინარე პროგრამის შეწყვეტაზე და გადასვლა წყვეტების მომსახურების პროგრამაზე.

არითმეტიკულ-ლოგიკური მოწყობილობა

როგორც დასახელებიდან ჩანს, ამ მოწყობილობის დანიშნულებაა მონაცემების არითმეტიკული და ლოგიკური დამუშავება. ნახ. 3.1-ზე წარმოდგენილ მანქანაში ის შეიცავს შემდეგ კვანძებს: ოპერაციულ ბლოკს, ოპერანდების რეგისტრებს, ნიშნულთა რეგისტრებს და აკუმულატორს. მოკლედ განვიხილოთ თითოეული მათგანი.

ოპერაციული ბლოკი

ოპერაციული ბლოკი არითმეტიკულ-ლოგიკური მოწყობილობის იმ ნაწილს წარმოადგენს, რომელიც ასრულებს შესასვლელზე მოწოდებულ ოპერანდებზე არითმეტიკულ და ლოგიკურ ოპერაციებს. მოცემული ოპერაციული ბლოკისათვის ოპერაციათა შესაძლო სიიდან კონკრეტული ოპერაციის ამორჩევა განისაზღვრება ბრძანების ოპერაციის კოდით. ჩვენს გამომთვლელ მანქანაში ოპერა-

ციის კოდის მოწოდება ხდება უშუალოდ ბრძანებების რეგისტრიდან. რეალურ მანქანებში ოპერაციის კოდი ხშირად მიკროპროგრამულ ავტომატში გარდაიქმნება სხვა ფორმაში და უკვე მიკროპროგრამული ავტომატიდან ეწოდება არითმეტიკულ-ლოგიკურ მოწყობილობას. თანამედროვე არითმეტიკულ-ლოგიკურ მოწყობილობების ოპერაციული ბლოკები იგებიან როგორც კომბინაციური სქემები, ანუ მათ არ გააჩნიათ შიგა მეხსიერება და შედეგის დამახსოვრების მომენტამდე ოპერანდები უნდა იმყოფებოდნენ ბლოკის შესასვლელზე.

ოპერანდთა რეგისტრები

„ოპ რეგ X“ და „ოპ რეგ У“ რეგისტრები უზრუნველყოფენ ოპერანდების დამახსოვრებას ოპერაციული ბლოკის შესასვლელზე ოპერაციის შედეგის მიღებამდე და მის ჩაწერამდე (ჩვენს შემთხვევაში აკუმულატორში).

ნიშნულთა რეგისტრი

ნიშნულთა რეგისტრის დანიშნულებაა ბოლოს შესრულებული არითმეტიკული ან ლოგიკური ოპერაციის შედეგის დამახასიათებელ ნიშნულია (ალმების) ფიქსირება და შენახვა. ასეთმა ნიშნულებმა შეიძლება მოგვაწოდონ ინფორმაცია შედეგის ნულთან ტოლობაზე, შედეგის ნიშანზე, ზედა თანრიგიდან გადატანის გაჩენაზე, თანრიგობრივი ბადის გადავსებაზე და ა. შ. ნიშნულთა რეგისტრის შემცველობა ჩვეულებრივ გამოიყენება მართვის მოწყობილობის მიერ არითმეტიკულ-ლოგიკური მოწყობილობის ოპერაციების შედეგების მიხედვით პირობითი გადასვლების განხორციელებისათვის. თითოეული შესაძლო ნიშნულისათვის გამოიყოფა ნიშნულთა რეგისრის ერთი თანრიგი.

ნიშნულთა ფორმირება ხორციელდება ნიშნულთა რეგისტრის მდგომარეობის ფორმირების ბლოკით, რომელიც შეიძლება შედიოდეს ოპერაციული ბლოკის შემადგენლობაში, ან რეალიზებული იყოს ოპერაციულ ბლოკსა და ნიშნულთა რეგისტრს შორის მოთავსებული გარე სქემის სახით.

აკუმულატორი

ა კ უ მ უ ლ ა ტ ო რ ი - რეგისტრია, რომელსაც ევალება ყველაზე განსხვავებული ფუნქციები. ასე მაგალითად, მასში წინასწარ იტვირთება არითმეტიკულ ან ლოგიკურ ოპერაციაში მონაწილე ერთ-ერთ ოპერანდი. აკუმულატორში შეიძლება ინახებოდეს წინა ბრძანების შედეგი და მასში ხდება მომდევნო ოპერაციის შედეგის შეტანა. ხშირად აკუმულატორის გავლით ხორციელდება შეტანისა და გამოტანის ოპერაციები.

მკაცრი მსჯელობით, აკუმულატორი თანაბარი ზომით შეიძლება მივანიჭოთ როგორც არითმეტიკულ-ლოგიკურ მოწყობილობას, ისე მართვის მოწყობილობას, ხოლო გამომთვლელ მანქანებში რეგისტრული არქიტექტურით ის შეიძლება განვიხილოთ როგორც ერთ-ერთი რეგისტრი.

ძირითადი მეხსიერება

გამოყენებული მიკროსქემების ტიპისაგან დამოუკიდებლად, ძირითადი მეხსიერება წარმოადგენს დამამახსოვრებელ ელემენტთა მასივს, რომლებიც ორგანზომილებიანებია უჯრედთა სახით და შეუძლიათ შეინახონ რაღაც ერთეული ინფორმაცია, ჩვეულებრივ ერთი ბაიტი. თითოეულ უჯრედს აქვს უნიკალური მისამართი. ძირითადი მეხ-

სიერების უჯრედები ორგანიზებულია მატრიცის სახით, ხოლო უჯრედის ამორჩევა ხორციელდება ამ მატრიცის შესაბამის სტრიქონსა და სვეტზე ნების დართვის სიგნალების მიწოდების გზით. ამის უზრუნველყოფა ხდება მეხსიერების მისამართის დეკოდერით, რომელიც გარდაქმნის მეხსიერების მისამართის რეგისტრიდან მიღებულ უჯრედის მისამართის რეგისტრიდან მიღებულ უჯრედის მისამართის ნების დართვის (ნებართვის) სიგნალებში, რომლებიც მიეწოდებიან იმ ჰორიზონტალურ და ვერტიკალურ ხაზებს, რომელთა გადაკვეთაზეცაა მოთავსებული სამისამართო (საჭირო) უჯრედი. ძირითადი მეხსიერების თანამედროვე მოცულობისას მოცემული სიგნალების რეალიზაციისთვის იყენებენ დამახსოვრების მოწყობილობების რამდენიმე მიკროსქემას. ასეთ პირობებში უჯრედისადმი მიმართვის პროცესი შედგება საჭირო მიკროსქემის არჩევისა (მისამართის უფროსი თანრიგების საფუძველზე) და მიკროსქემის შიგნით უჯრედის ამორჩევაში (განისაზღვრება მისამართის უმცროსი თანრიგებით). პროცედურის პირველი ნაწილი ხორციელდება გარე სქემებით, ხოლო მეორე - დამამახსოვრებელი მოწყობილობის მიკროსქემათა შიგნით.

შეტანა/გამოტანის მოდული

ნახ. 3.1-ზე მოყვანილი შ ე ტ ა ნ ა / გ ა მ ო ტ ა ნ ი ს მ ო დ უ -ლ ი ს სტრუქტურა უზრუნველყოფს მხოლოდ გამომთვლელი მანქანის მუშაობის ლოგიკის ახსნას. რეალურ გამომთვლელ მანქანებში მანქანის ამ მოწყობილობის რეალიზაცია შეიძლება მნიშვნელოვნად განსხვავდებოდეს განხილულისაგან. შეტანა/გამოტანის მოდულის ამოცანას წარმოადგენს გამომთვლელ მანქანაზე სხვადასხვა პერიფერიული მოწყობილობების მიერთების უზრუნველყოფა და მათთან ინფორმაციის გაცვლა. განხილულ ვარიანტში შეტანა/გამოტანის მოდული შედგება შეტანა/გამოტანის პორტის ნომრის დეკოდერის, შეტანის პორტების სიმრავლისა და გამოტანის პორტების სიმრავლისაგან.

შეტანის პორტები და გამოტანის პორტები

პორტი ჰქვია სქემას, რომელიც ახორციელებს ინფორმაციის გადაცემას შეტანის პერიფერიული მოწყობილობიდან არითმეტიკულ-ლოგიკური მოწყობილობის აკუმულატორში (შეტანის პორტი) ან აკუმულატორიდან გამოტანის პერიფერიულ მოწყობილობაზე (გამოტანის პორტი). სქემა უზრუნველყოფს გამომთვლელი მანქანის ელექტრულ და ლოგიკურ დაკავშირებას მასზე მიერთებულ პერიფერიულ მოწყობილობასთან.

შეტანა/გამოტანის პორტის ნომრის დეშიფრატორი

განსახილველი გამომთვლელი მანქანის შეტანა/გამოტანის მოდულში იგულისხმება, რომ თითოეული პერიფერიული მოწყობილობა უერთდება თავის პორტს. თითოეულ პორტს გააჩნია უნიკალური ნომერი, რომელიც ეთითება შეტანა/გამოტანის ბრძანების სამისამართო ნაწილში. შეტანა/გამოტანის პორტის ნომრის დეკოდერი უზრუნველყოფს პორტის ნომრის გარდაქმნას შესაბამის პორტზე შეტანის ან გამოტანის ოპერაციაზე ნებართვის სიგნალში. უშუალოდ შეტანა (გამოტანა) ხდება მიკროპროგრამული ავტომატიდან მიღებული „შეტანა (გამოტანა)“ სიგნალით.

4. სალტეთა ორგანიზაცია

ტრაქტების ერთობლიობა, რომელიც აერთიანებს გამომთვლელი მანქანის ძირითად მოწყობილობებს (ცენტრალურ პროცესორს, მეხსიერებას და შეტანა/გამოტანის მოდულებს), ქმნის გამომთვლელი მანქანის ურთიერთკავშირთა სტრუქტურას. ურთიერთკავშირთა სტრუქტურამ უნდა უზრუნველყოს ინფორმაციის გაცვლა:

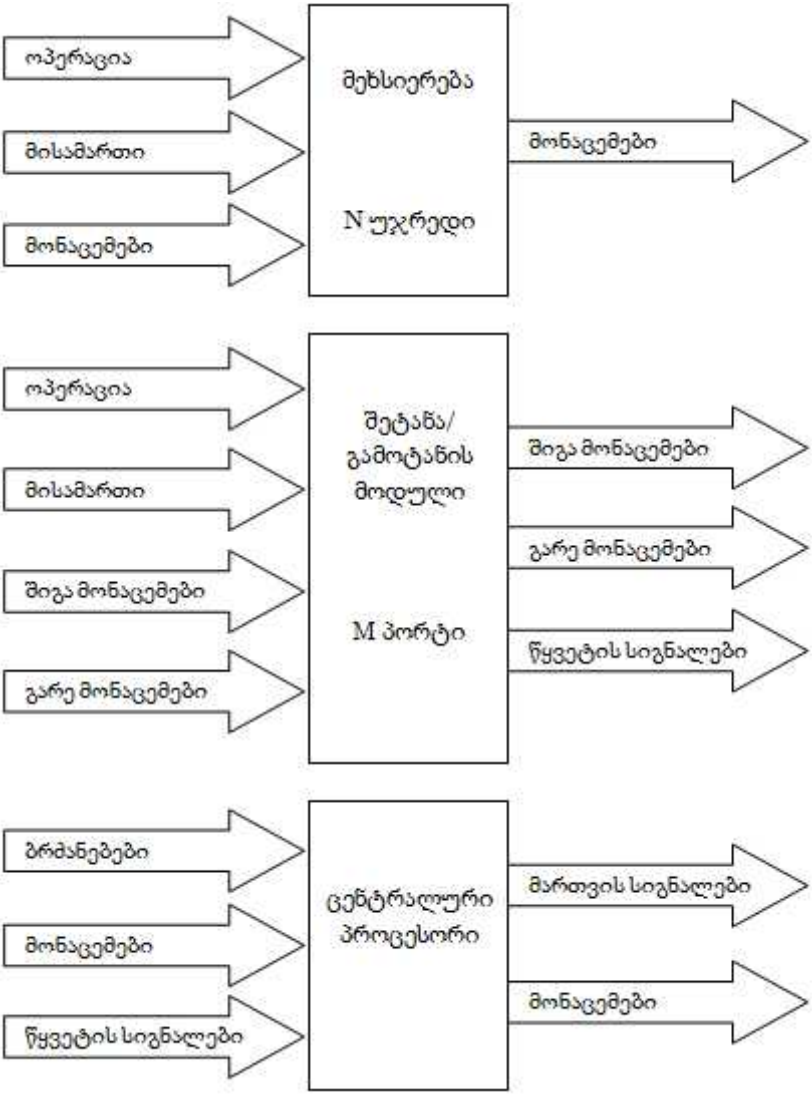
- ცენტრალურ პროცესორსა და მეხსიერებას შორის;

- ცენტრალურ პროცესორსა და შეტანა/გამოტანის მოდულებს შორის;

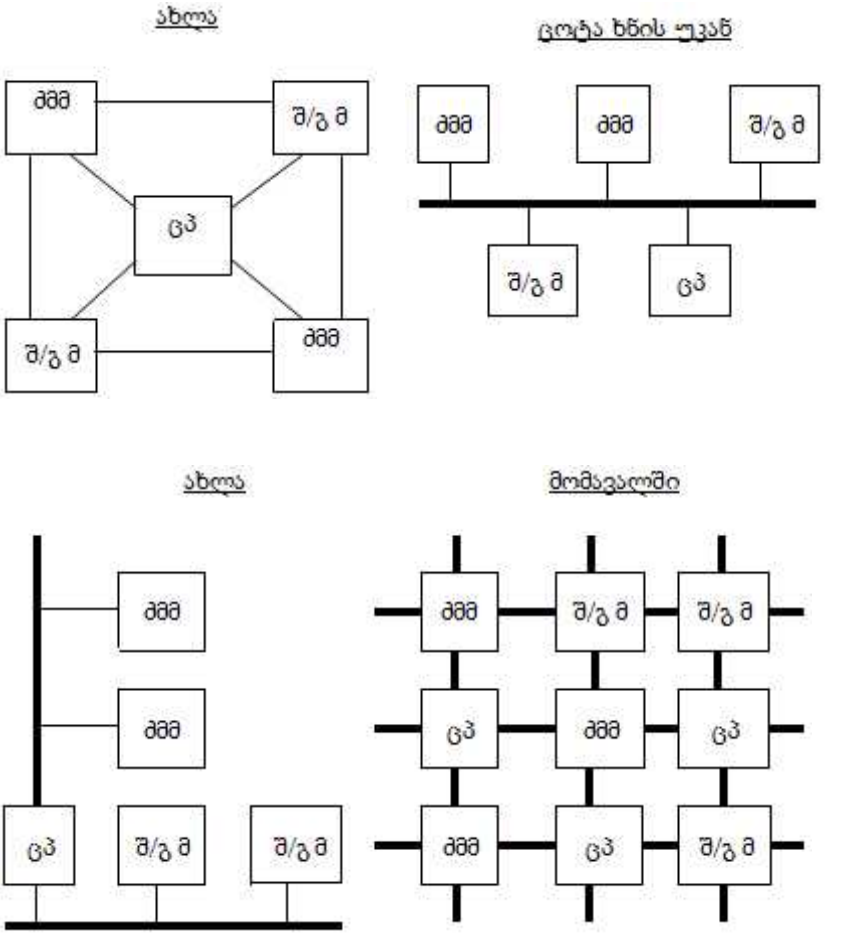
- მეხსიერებასა და შეტანა/გამოტანის მოდულებს შორის.

გამომთვლელი მანქანის ძირითადი მოწყობილობებისათვის დამახასიათებელი ინფორმაციული ნაკადები მოტანილია ნახ. 4.1-ზე.

გამოთვლითი ტექნიკის განვითარებასთან ერთად იცვლებოდა გამომთვლელი მოწყობილობების ურთიერთკავშირთა სტრუქტურაც (ნახ. 4.2). საწყის ეტაპზე ჭარბობდნენ უშუალო კავშირები გამომთვლელი მანქანის ურთიერთმოქმედ მოწყობილობებს შორის. მინი ეგმ-ების, და განსაკუთრებით, პირველი მიკრო-ეგმ-ების გამოჩენასთან ერთად სულ უფრო პოპულარული ხდება სქემა ერთი საერთო სალტით, ამიტომ გამომთვლელი მანქანის პრაქტიკულად ყველა მოწყობილობის წარმადობის სწრაფმა ზრდამ მიგვიყვანა ერთადერთი სალტის უუნაროზასთან გამკლავებოდა გაზრდილ ტრაფიკს (ტრაფიკი - შეტყობინებათა ნაკადი მონაცემთა გადაცემის ქსელში, კავშირის ხაზის საშუალო დატვირთვა), და მას ენაცვლებიან რამდენიმე სალტის საფუძველზე აგებული ურთიერთკავშირთა სტრუქტურები. გამოთვლების წარმადობის ზრდის შემდგომი პერსპექტივები დაკავშირებულნი არიან არა იმდენად ერთპროცესორულ მანქანასთან, არამედ მრავალპროცესორულ გამოთვლით სისტემებთან.



ნახ. 4.1. ინფორმაციული ნაკადები გამომთვლელ მანქანაში



ნახ. 4.2. ურთიერთკავშირის სტრუქტურათა ევოლუცია (ცპ- ცენტრალური პროცესორი, ძმმ - ძირითადი მეხსიერების მოდული, შ/გ მ- შეტანა/გამოტანის მოდული)

გამომთვლელი მანქანის ნაწილების ურთიერთკავშირი და მისი „ურთიერთობა“ გარე სამყაროსთან უზრუნველყოფილია სალტეთა სისტემით. მანქანათა უმრავლესობა შეიცავს რამდენიმე განსხვავებულ სალტეს, რომელთაგან თითოეული ოპტიმიზებულია კომუნიკაციათა განსაზღვრული სახის მიმართ. სალტეთა ნაწილი დამალულია ინტეგრალური მიკროსქემების შიგნით ან მისაღწევია მხოლოდ ბეჭდური დაფის საზღვრებში. ზოგიერთ სალტეს აქვს გარედან მისაწვდომი წერტილები, რომ მათზე ადვილად განხორციელდეს დამატებით მოწყობილობათა მიერთება. ამასთან, ასეთ სალტეთა უმრავლესობა არა მხოლოდ მისაწვდომია, არამედ პასუხობს გარკვეულ სტანდარტებს, რაც საშუალებას იძლევა სალტეზე მიერთებულ იქნას სხვადასხვა წარმადობის მოწყობილობები.

კონკრეტული სალტის დასახასიათებლად საჭიროა აღვწეროთ:

- სასიგნალო ხაზების ერთობლიობა;

- სალტის ფიზიკური, მექანიკური და ელექტრული მახასიათებლები;

- არბიტრაჟის, მდგომარეობის, მართვისა და სინქრონიზაციის გამოყენებული სიგნალები;

- სალტესთან მიერთებული მოწყობილობების ურთიერთქმედების წესები (სალტის პროტოკოლი).

სალტეს ქმნის კომუნიკაციურ ხაზთა ნაკრები, რომელთაგან თითოეულს შეუძლია გადასცეს ორობითი ციფრებით 1 და 0 წარმოდგენილი სიგნალები. ხაზით შეიძლება გადაიგზავნოს ასეთი სიგნალების დროში გაშლილი თანმიმდევრობა. რამდენიმე ხაზის ერთობლივი გამოყენების შემთხვევაში შესაძლებელია ორობითი რიცხვების ერთდროული (პარალელური) გადაცემა. ფიზიკურად სალტის ხაზები რეალიზებულია ცალკეული გამტარების სახით, როგორც გამტარი მასალის ზოლები სამონტაჟო დაფაზე ან როგორც ალუმინის ან სპილენძის გამტარი ბილიკები მიკრო სქემის კრისტალზე.

სალტეზე ოპერაციებს ტ რ ა ნ ზ ა ქ ც ი ე ბ ს უწოდებენ. ტრანზაქციათა ძირითადი სახეებია - წ ა კ ი თ ხ ვ ი ს ტ რ ა ნ ზ ა ქ ც ი ა და

ჩ ა წ ე რ ი ს ტ რ ა ნ ზ ა ქ ც ი ა. თუ გაცვლაში მონაწილეობს შეტანა /გამოტანის მოწყობილობა, შეიძლება ვისაუბროთ შ ე ტ ა ნ ი ს ა და გ ა მ ო ტ ა ნ ი ს ტ რ ა ნ ზ ა ქ ც ი ე ბ ი ს შესახებ, რომლებიც არსებითად ექვივალენტურებია შესაბამისად ჩაწერისა და წაკითხვის ტრანზაქციების. სალტური ტრანზაქცია შედგება ორი ნაწილისაგან: მისამართის გაგზავნისა და მონაცემთა მიღებისაგან (ან გადაგზავნისაგან).

როცა ორი მოწყობილობა ახორციელებს ინფორმაციის გაცვლას სალტის მეშვეობით, მაშინ ერთ-ერთმა მათგანმა უნდა მოახდინოს გაცვლის ინიცირება და მისი მართვა. ასეთ მოწყობილობას წ ა მ ყ ვ ა ნ ი (bus master) ეწოდება. კომპიუტერულ ტერმინოლოგიაში „წამყვანი“ - ნებისმიერი მოწყობილობაა, რომელსაც შეუძლია თავის თავზე აიღოს სალტის ფლობა და მართოს მონაცემთა გადაგზავნა. აუცილებელი არაა, რომ წამყვანმა თავად გამოიყენოს მონაცემები. მან, მაგალითად, შეიძლება მიიტაცოს სალტის მართვა სხვა მოწყობილობის ინტერესებიდან გამომდინარე. მოწყობილობებს, რომლებსაც არ გააჩნიათ ტრანზაქციათა ინიცირების შესაძლებლობა, მ ი მ ყ ო ლ ს (bus slave) ეძახიან. პრინციპში სალტეზე შეიძლება მიერთებული იყოს რამდენიმე პოტენციური წამყვანი, მაგრამ დროის ნებისმიერ მომენტში აქტიურია მხოლოდ ერთი მათგანი: თუ რამდენიმე მოწყობილობა ერთდროულად გადასცემს ინფორმაციას, მაშინ მათი სიგნალები ფარავენ ერთმანეთს და მახინჯდებიან. რამდენიმე წამყვანის ერთდროული გააქტიურების თავიდან ასაცილებლად ნებისმიერ სალტეში გათვალისწინებულია სალტის მართვასთან ერთ-ერთი პრეტენდენტის დაშვების პროცედურა (არბიტრაჟი). ამასთან ზოგიერთი სალტე უშვებს ჩაწერის მრავლის მაუწყებელ (მრავლის აღმთქმელ) რეჟიმს, როცა ინფორმაცია ერთი წამყვანიდან პირდაპირ გადაეცემა რამდენიმე მიმყოლს (აქ არბიტრაჟი საჭირო არაა). ერთი მოწყობილობით გაცემული სიგნალი მისაღწევია სალტეზე მიერთებული ყველა დანარჩენი მოწყობილობისათვის.

4.1. სალტეთა ტიპები

მნიშვნელოვან კრიტერიუმად, რომელიც განსაზღვრავს სალტის მახასიათებლებს, შეიძლება აღებულ იქნას მისი მიზნობრივი დანიშნულება. ამ კრიტერიუმით შეიძლება გამოვყოთ:

- სალტე „პროცესორი-მეხსიერება“;
- შეტანა/გამოტანის სალტე;
- სისტემური სალტე.

მოკლედ განვიხილოთ თითოეული მათგანი.

სალტე „პროცესორი - მეხსიერება“

სალტე „პროცესორი-მეხსიერება“ უზრუნველყოფს უშუალო კავშირს გამომთვლელი მანქანის ცენტრალურ პროცესორსა და ძირითად მეხსიერებას შორის. თანამედროვე მიკრო პროცესორებში ასეთ სალტეს ხშირად წინა მხარის სალტეს უწოდებენ და აღნიშნავენ FSB (Front-Side Bus)-ით. პროცესორსა და მეხსიერებას შორის ინტენსიური ტრაფიკი მოითხოვს, რომ სალტის გამტარუნარიანობა, ანუ დროის ერთეულში სალტეში გატარებული ინფორმაციის რაოდენობა, იყოს უდიდესი. ამ სალტის როლს ხშირად სისტემური სალტე ასრულებს (იხ. ქვემოთ), მაგრამ ეფექტურობის თვალსაზრისით მნიშვნელოვნად მომგებიანია თუ გაცვლა ცენტრალურ პროცესორსა და ძირითად მეხსიერებას შორის ხორციელდება ცალკე სალტით. განსახილველ სახეობას შეიძლება მივაკუთვნოთ აგრეთვე სალტე, რომელიც აკავშირებს პროცესორს მეორე დონის კეშ-მეხსიერებასთან. ის ცნობილია, როგორც უკანა მხარის სალტე - BSB (Back-Side Bus). BSB საშუალებას იძლევა გაცვლა წარმოებდეს უფრო დიდი სისწრაფით, ვიდრე FSB-ს დროს, და სრულად იქნას გამოყენებული უფრო სწრაფი კეშ - მეხსიერების შესაძლებლობები.

ვინაიდან ფონ-ნეიმანისეულ მანქანებში სწორედ პროცესორსა და მეხსიერებას შორის გაცვლა ბევრად განაპირობებს გამომთვლელი მანქანის სწრაფქმედებას, ამიტომ დამმუშავებლები ცენტრალური პროცესორის მეხსიერებასთან კავშირს განსაკუთრებულ ყურადღებას აქცევენ. მაქსიმალური გამტარუნარიანობის უზრუნველსაყოფად სალტე „პროცესორი - მეხსიერება“ ყოველთვის პროექტირდება მეხსიერების სისტემის ორგანიზაციის თავისებურებათა გათვალისწინებით, ხოლო სალტის სიგრძეს შესაძლებლად მინიმალურს აკეთებენ.

შეტანა/გამოტანის სალტე

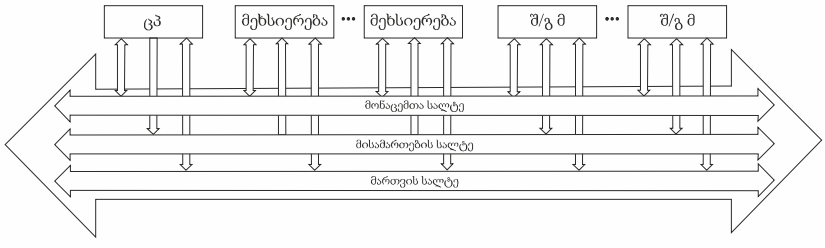
შეტანა/გამოტანის სალტე ემსახურება პროცესორის (მეხსიერების) შეტანა/გამოტანის მოწყობილობებთან დაკავშირებას. ასეთ მოწყობილობათა მრავალფეროვნების გათვალისწინებით ახდენენ შეტანა/გამოტანის სალტეთა უნიფიცირებასა და სტანდარტიზაციას. უმრავლეს შეტანა/გამოტანის მოწყობილობებთან კავშირი (მაგრამ არა ვიდეო სისტემებთან) არ მოითხოვენ სალტის მაღალ გამტარუნარიანობას. შეტანა/გამოტანის სალტეთა პროექტირებისას ითვალისწინებენ კონსტრუქტივისა და შემაერთებელი გასართების ღირებულებას. ასეთი სალტეები „პროცესორ-მეხსიერება“ სალტეებთან შედარებით შეიცავენ ნაკლებ ხაზებს, მაგრამ ხაზთა სიგრძე შეიძლება საკმაოდ დიდი იყოს. მსგავსი სალტეების ტიპიური წარმომადგენლებია PCI და SCSI სალტეები.

სისტემური სალტე

ფასდაკლების მიზნით ზოგიერთ გამომთვლელ მანქანას აქვს საერთო სალტე მეხსიერებისა და შეტანა/გამოტანის მოწყობილობებისათვის. ასეთ სალტეს ხშირად სისტემურს უწოდებენ. ს ი ს ტ ე მ უ რ ი

ს ა ლ ტ ე ემსახურება გამომთვლელი მანქანის ყველა მოწყობილობის ფიზიკურ და ლოგიკურ გაერთიანებას. ვინაიდან მანქანის ძირითადი მოწყობილობები, როგორც წესი, მოთავსებულნი არიან საერთო სამონტაჟო დაფაზე, ამიტომ სისტემურ სალტეს ხშირად გამაერთიანებელ სალტეს (backplane bus) ეძახიან, თუმცა ეს ტერმინები არ შეიძლება მკაცრად ექვივალენტურებად ჩავთვალოთ.

სისტემურ სალტეში შეიძლება გაერთიანდეს რამდენიმე ასეული ხაზი. სალტის ხაზთა ერთობლიობა შეიძლება დაიყოს სამ ფუნქციურ ჯგუფად (ნახ. 4.3): მონაცემთა სალტედ, მისამართის სალტედ და მართვის სალტედ. ამ უკანასკნელს ჩვეულებრივ აკუთვნებენ სისტემურ სალტეზე მიერთებული მოდულებისათვის კვების ძაბვის მომწოდებელ სადენებს.



ნახ. 4.3. სისტემური სალტის სტრუქტურა

სისტემური სალტის ფუნქციონირება შემდეგნაირად შეიძლება აღვწეროთ. თუ ერთ-ერთ მოდულს სურს გადასცეს მონაცემები მეორეს, მაშინ მან უნდა შეასრულოს ორი მოქმედება: თავის განკარგულებაში მიიღოს სალტე და მასზე გადასცეს მონაცემები. თუ რომელიმე მოდულს სურს მიიღოს მონაცემები მეორე მოდულიდან, მან უნდა მიიღოს დაშვება სალტესთან და მართვისა და მისამართის შესაბამისი ხაზებით გადასცეს მეორე მოდულს მოთხოვნა. შემდეგ ის უნდა დაელოდოს, როცა მოთხოვნის მიმღები მოდული გამოგზავნის მონაცემებს.

ფიზიკურად სისტემური სალტე წარმოადგენს ელექტრული გამტარების პარალელურ ერთობლიობას. ამ გამტარებად გამოიყენებიან მეტალის ზოლები ბეჭდურ დაფაზე. სალტე მიიყვანება ყველა მოდულთან და თითოეული მათგანი უერთდება ყველა დანარჩენს ან ზოგიერთ მის ხაზს. თუ გამომთვლელი მანქანა კონსტრუქციულად რამდენიმე დაფაზეა აგებული, მაშინ სალტის ყველა ხაზი გამოყვანილია გასართებზე, რომლებიც შემდეგ ერთიანდებიან სადენებით საერთო შასიზე.

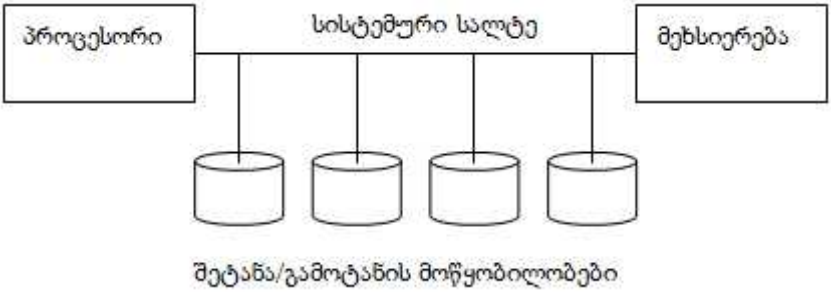
უნივერსალური გამომთვლელი მანქანის სტანდარტიზებულ სისტემურ სალტეებს შორის ყველაზე მეტად ცნობილია Unibus, Fastbus, Futurebus, VME, NuBus, Multibus-II სალტეები. როგორც წესი, პერსონალურ კომპიუტერებს აგებენ ISA, ან ETSA ან MCA სტანდარტის სისტემური სალტეების საფუძველზე.

4.2. სალტეთა იერარქია

თუ სალტეზე მიერთებულია მოწყობილობათა დიდი რაოდენობა, მაშინ მისი გამტარუნარიანობა ეცემა, ვინაიდან სალტის მიერ მართვის უფლებების ძლიერ ხშირი გადაცემა ერთი მოწყობილობიდან მეორეზე იწვევს საგრძნობ დაყოვნებებს. ამ მიზეზის გამო ბევრ გამომთვლელ მანქანაში უპირატესობა ენიჭება რამდენიმე სალტის გამოყენებას, რომლებიც ქმნიან განსაზღვრულ იერარქიას. ჯერ განვიხილოთ ერთსალტიანი გამომთვლელი მანქანა.

ერთსალტიანი გამომთვლელი მანქანა

ერთი სალტით ურთიერთკავშირთა სტრუქტურებში არის ერთი სისტემური სალტე, რომელიც უზრუნველყოფს ინფორმაციის გაცვლას პროცესორსა და მეხსიერებას შორის, აგრეთვე შეტანა/გამოტანის მოწყობილობასა და პროცესორს ან მეხსიერებას შორის (ნახ. 4.4).



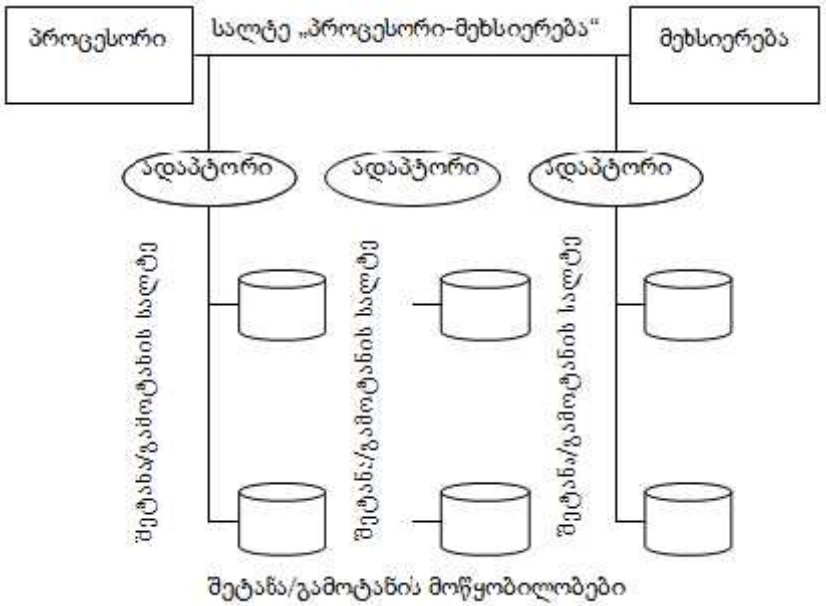
ნახ. 4.4. ურთიერთკავშირთა სტრუქტურა ერთი სალტის დროს

ასეთი მიდგომისათვის დამახასიათებელია სიმარტივე და დაბალი ღირებულება. ოღონდ ერთსალტიან ორგანიზაციას არ შეუძლია განახორციელოს ტრანზაქციათა მაღალი ინტენსივობა და სისწრაფე, ამასთან კუმპიუტერის „ვიწრო ადგილად“ იქცევა სწორედ სალტე.

გამომთვლელი მანქანა ორი სახის სალტით

თუმცა შეტანა/გამოტანის მოწყობილობის კონტროლერები შეიძლება უშუალოდ იყვნენ მიერთებული სალტესთან, მაგრამ დიდი ეფექტი მიიღწევა მხოლოდ შეტანა/გამოტანის ერთი ან რამდენიმე სალტის გამოყენებით (ნახ. 4.5). შეტანა/გამოტანის მოწყობილობები უერეთდებიან შეტანა/გამოტანის სალტეებს, რომლებიც თავის თავზე იღებენ ძირითად ტრაფიკს, რომელიც არაა დაკავშირებული პროცესორზე ან მეხსიერებაზე გასვლაზე. ს ა ლ ტ ე თ ა ა დ ა პ ტ ო რ ე ბ ი უზრუნველ-

ყოფენ მონაცემეთა ბუფერიზაციას მათი გადაგზავნისას სისტემურ სალტესა და შეტანა/გამოტანის მოწყობილობას შორის. ეს საშუალებას აძლევს გამომთვლელ მანქანას მხარი დაუჭიროს შეტანა/გამოტანის მრავალ მოწყობილობასთან მუშაობას და იმავე დროს განახორციელოს ინფორმაციის გაცვლის პროცესი პროცესორი-მეხსიერება ტრაქტით და ინფორმაციის გაცვლა შეტანა/გამოტანის მოწყობილობებთან.



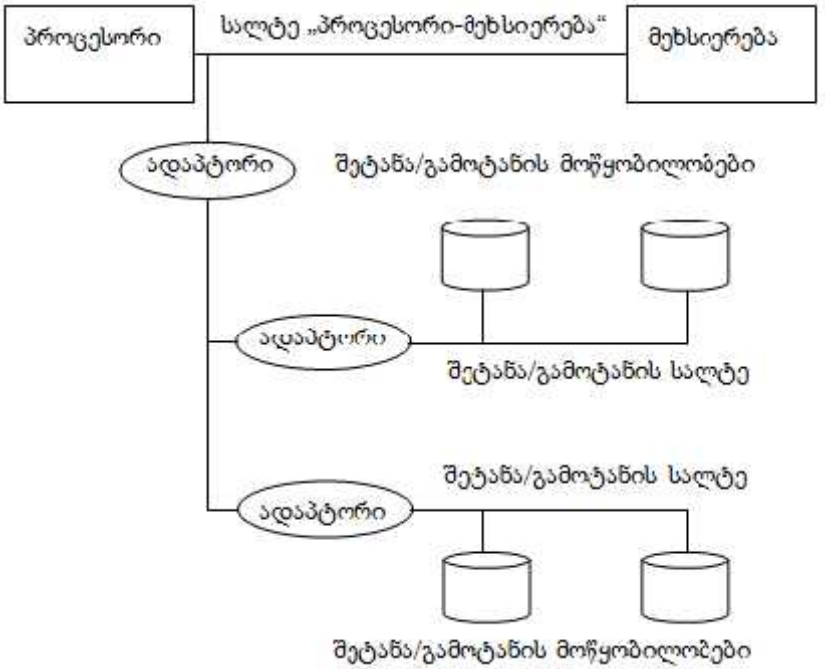
ნახ. 4.5. ურთიერთკავშირთა სტრუქტურა ორი სახის სალტით

მსგავსი სქემა მნიშვნელოვნად ამცირებს დატვირთვას სწრაფქმედ სალტეზე „პროცესორი-მეხსიერება“ და ხელს უწყობს გამომთვლელი მანქანის საერთო წარმადობის ზრდას. მაგალითისათვის შეგვიძლია მოვიყვანოთ გამომთვლელი მანქანა Apple Macintosh II, სადაც „პროცესორი-მეხსიერება“ სალტის როლს ასრულებს NuBus სალტე. პროცესორისა და მეხსიერების გარდა მასზე მიერთებულია ზოგიერთი

შეტანა/გამოტანის მოწყობილობა. შეტანა/გამოტანის სხვა მოწყობილობები მიერთებულია SCSI Bus სალტეზე.

გამომთვლელი მანქანა სამი სახის სალტით

სწრაფქმედი პერიფერიული მოწყობილობების მისაერთებლად სალტეთა სისტემას შეიძლება დაემატოს გაფართოების სწრაფქმედი სალტე (ნახ. 4.6).



ნახ. 4.6. ურთიერთკავშირთა სტრუქტურა სამი სახის სალტით

შეტანა/გამოტანა სალტეები უერთდებიან გაფართოების სალტეს, ხოლო უკვე მისგან ადაპტორის გავლით სალტეს „პროცესორი-მეხსიე-

რება“. ეს სქემა კიდევ უფრო ამცირებს დატვირთვას სალტეზე „პროცესორი-მეხსიერება“. სალტეთა ასეთ ორგანიზაციას „მინაშენიან“ არქიტექტურას (mezzanine architecture) უწოდებენ.

4.3. სალტეთა არბიტრაჟი

რეალურ სისტემებში წამყვანის როლზე უფლება აქვს ერთდროულად პრეტენზია ჰქონდეს სალტეზე მიერთებულ მოწყობილობებიდან რამდენიმეს, მაგრამ დროის თითოეულ მომენტში სალტის მართვა მხოლოდ ერთს შეუძლია. კონფლიქტების თავიდან ასაცილებლად, სალტე უნდა ითვალისწინებდეს მოთხოვნების არბიტრაჟის განსაზღვრულ მექანიზმებს და მომთხოვნი მოწყობილობებიდან ერთ-ერთისათვის სალტის მიკუთვნების წესს. გადაწყვეტილება ჩვეულებრივ მიიღება პრეტენდენტების პრიორიტეტების საფუძველზე.

პრიორიტეტების სქემები

თითოეულ პოტენციურ წამყვანს ენიჭება პრიორიტეტის განსაზღვრული დონე, რომელიც შეიძლება უცვლელი რჩებოდეს (სტატიკური ანუ ფიქსირებული პრიორიტეტი) ანდა იცვლებოდეს რაიმე ალგორითმით (დინამიური პრიორიტეტი).

სტატიკური პრიორიტეტების ძირითადი ნაკლოვანება იმაში მდგომარეობს, რომ მაღალი პრიორიტეტის მქონე მოწყობილობას შეუძლია მოახდინოს დაბალი დონის პრიორიტეტის მქონე მოწყობილობების სალტეზე მიღწევის სრული ბლოკირება.

დინამიურ პრიორიტეტიანი სისტემები თითოეულ მომთხოვნ მოწყობილობას აძლევენ შანს ადრე თუ გვიან მიიღოს სალტის მართვის უფლება, ანუ ასეთ სისტემებში რეალიზებულია თანაბარი მიღწევადობის პრინციპი.

ყველაზე ფართო გავრცელება მოიპოვეს პრიორიტეტის დინამიური ცვლილების შემდეგმა ალგორითმებმა:

- პრიორიტეტების მარტივი ციკლური ცვლა;
- პრიორიტეტების ციკლური ცვლა ბოლო მოთხოვნის გათვალისწინებით;
- პრიორიტეტების ცვლა შემთხვევითი კანონით;
- თანაბარი პრიორიტეტების სქემა;
- ყველაზე ძველი გამოყენების ალგორითმი.

პრიორიტეტების მარტივი ციკლური ცვლის ალგორითმში არბიტრაჟის თითოეული ციკლის შემდეგ ყველა პრიორიტეტი დაბლდება ერთი დონით, ამასთან მოწყობილობა, რომელსაც ადრე პრიორიტეტის უდაბლესი დონე ჰქონდა, უმაღლეს პრიორიტეტს ღებულობს.

ბოლო მოთხოვნის გათვალისწინებით პრიორიტეტების ცვლის სქემაში ყველა შესაძლო მოთხოვნა წესრიგდება ციკლური სიის შესაბამისად. მომდევნო მოთხოვნის დამუშავების შემდეგ მომსახურებულ წამყვანს ენიჭება უდაბლესი დონის პრიორიტეტი. სიაში შემდეგი მოწყობილობა იღებს უმაღლეს პრიორიტეტს, ხოლო დანარჩენ მოწყობილობებს პრიორიტეტი ენიჭებათ დაღმავალი რიგით (კლებადობით), ციკლურ სიაში მათი მიდევნების თანახმად.

პრიორიტეტების ციკლური ცვლის ორივე სქემაში თითოეული წამყვანი უზრუნველყოფილია შანსით მიიღოს სალტე თავის განკარგულებაში, თუმცა უფრო ფართო გავრცელება მოიპოვა მეორე ალგორითმმა.

შემთხვევითი კანონით პრიორიტეტების ცვლისას არბიტრაჟის მომდევნო ციკლის შემდეგ ფსევდო შემთხ-

ვევითი რიცხვების გენერატორის საშუალებით თითოეულ წამყვანს ენიჭება პრიორიტეტის დონის შემთხვევითი მნიშვნელობა.

თანაბარი პრიორიტეტების სქემაში არბიტრთან რამდენიმე მოთხოვნის მიწოდებისას თითოეულ მათგანს მომსახურების თანაბარი შანსი გააჩნია. შესაძლო კონფლიქტი არბიტრის მიერ წყდება. ასეთი სქემა მიღებულია ასინქრონულ სისტემებში.

ყველაზე ადრეული (ძველი) გამოყენების ალგორითმში (LRU, Least Recently Used) არბიტრაჟის ყოველი ციკლის შემდეგ უმაღლესი პრიორიტეტი ენიჭება იმ წამყვანს, რომელსაც სხვებზე დიდი ხნის განმავლობაში არ გამოუყენებია სალტე.

განხილულებს გარდა არსებობს პრიორიტეტების ცვლის რამდენიმე ალგორითმი, რომლებიც წმინდად დინამიურები არ არიან, ვინაიდან პრიორიტეტების ცვლა ხორციელდება არბიტრაჟის არა ყოველი ციკლის შემდეგ. ასეთ ალგორითმებს მიეკუთვნებიან:

- რიგის ალგორითმი (პირველი მივიდა-პირველს მოემსახურნენ);
- ალგორითმი დროის ფიქსირებული ქვანტით.

რიგის ალგორითმში მოთხოვნების მომსახურება ხორციელდება არბიტრაჟის ციკლის დაწყების მომენტისათვის შექმნილი რიგის მიხედვით. ჯერ ხდება რიგის პირველი მოთხოვნის მომსახურება, ანუ სხვებზე ადრე მოსული მოთხოვნის. ალგორითმის აპარატურული რეალიზაცია დაკავშირებულია რიგ სირთულეებთან, ამიტომ ის იშვიათად გამოიყენება.

დროის ფიქსირებული ქვანტის ალგორითმში წამყვანს სალტის მიტაცებისათვის არბიტრაჟის ციკლის განმავლობაში გამოყოფილი აქვს დროის განსაზღვრული ქვანტი. თუ ამ მომენტში წამყვანს სალტე არ ესაჭიროება, მაშინ მასზე გამოყოფილი ქვანტი გამოუყენებელი რჩება. ასეთი მეთოდი ყველაზე მეტად გამოყენებადია სინქრონული პროტოკოლის მქონე სალტეებისათვის.

5. მეხსიერება

ნებისმიერ გამომთვლელ მანქანაში, მისი არქიტექტურისაგან დამოუკიდებლად, პროგრამები და მონაცემები მეხსიერებაში ინახებიან. მეხსიერების ფუნქციები უზრუნველყოფილია მეხსიერების მოწყობილობებით, რომელთა დანიშნულებაა გამომთვლელი მანქანის მუშაობის პროცესში ინფორმაციის ფიქსირება, შენახვა და გაცემა. მეხსიერების მოწყობილობაში ინფორმაციის ფიქსირების პროცესს ჩაწერა ჰქვია, ინფორმაციის გაცემის პროცესს - წაკითხვა ან ამოკითხვა, ხოლო ერთად მათ განსაზღვრავენ როგორც მეხსიერების მოწყობილობასთან მიმართვის პროცესებს.

5.1. მეხსიერების სისტემის მახასიათებლები

ძირითადი მახასიათებლების ჩამონათვალი, რომელიც აუცილებელია გათვალისწინებულ იქნას კონკრეტული სახის მეხსიერების მოწყობილობის განხილვისას, შემდეგია:

- განთავსების ადგილი;
- ტევადობა;
- გადაგზავნის ერთეული;
- შეღწევის მეთოდი;
- სწრაფქმედება;
- ფიზიკური ტიპი;
- ფიზიკური თავისებურებები;
- ღირებულება.

განთავსების ადგილის მიხედვით გვაქვს პროცესორული, შიგა და გარე მეხსიერების მოწყობილობები. მეხსიერების ყველაზე სწრაფი სახეობები (რეგისტრები, პირველი დონის კეშ-მეხსიერება) ჩვეულებრივ მოთავსებულნი არიან საერთო კრისტალზე პროცესორთან ერთად, ხოლო საერთო დანიშნულების რეგისტრები საერთოდ ითვლებიან ცენტრალური პროცესორის ნაწილად. მეორე ჯგუფს (შიგა მეხსიერებას) ქმნიან მეხსიერების ის მოწყობილობები, რომლებიც მოთავსებულნი არიან სისტემურ პლატაზე. შიგა მეხსიერებას აკუთვნებენ ძირითად მეხსიერებას, აგრეთვე მეორე და მომდევნო დონეების კეშ-მეხსიერებას (მეორე დონის კეშ-მეხსიერება ასევე შეიძლება მოთავსებული იყოს პროცესორის კრისტალზე). დიდი ტევადობის ნელი მეხსიერების მოწყობილობებს (მაგნიტური და ოპტიკური დისკები, მაგნიტური ლენტები) ეძახიან გარე მეხსიერებას, ვინაიდან გამომთვლელი მანქანის ბირთვს ისინი ერთვებიან შეტანა/გამოტანის მოწყობილობების ანალოგიურად.

მეხსიერების მოწყობილობის ტევადობას ახასიათებენ ბიტების ან ბაიტების იმ რიცხვით, რომელიც შეიძლება ინახებოდეს მასში. პრაქტიკაში გამოიყენება უფრო დიდი ერთეულები, ხოლო მათი აღნიშვნისათვის „ბიტ“ ან „ბაიტ“ სიტყვებს წინსართებს უმატებენ: კილო, მეგა, გიგა, ტერა, პეტა, ეკზა. სტანდარტულად ეს წინსართები ნიშნავენ ძირითადი გაზომვის ერთეულის გამრავლებას შესაბამისად 10^3, 10^6, 10^9, 10^{12}, 10^{15} და 10^{18}-ზე.

მეხსიერების მოწყობილობის მნიშვნელოვან მახასიათებელს წარმოადგენს გადაგზავნის ერთეული. ძირითადი მეხსიერებისათვის გადაგზავნის ერთეული განისაზღვრება მონაცემთა სალტის სიფართით (სიგანით), ანუ სალტის ხაზებით პარალელურად გადაცემული ბიტების რაოდენობით. ჩვეულებრივ გადაცემის ერთეული ტოლია სიტყვის სიგრძის, მაგრამ არა აუცილებლად. გარე მეხსიერებისათვის მონაცემები ხშირად გადაიცემიან სიტყვის ზომაზე მეტი ერთეულებით და ასეთ ერთეულებს ბლოკებს ეძახიან.

სწრაფქმედების შეფასებისას აუცილებლადაა გასათვალისწინებელი მეხსიერების მოწყობილობის მოცემულ ტიპში გამოყენებულ მონაცემებთან წვდომის მეთოდი. ანსხვავებენ წვდომის ოთხ ძირითად მეთოდს:

- მიმდევრობითი წვდომა. მეხსიერების მოწყობილობა მიმდევრობითი წვდომით ორიენტირებულია ინფორმაციის შენახვაზე მონაცემთა ბლოკების თანმიმდევრობის სახით. მათ ჩანაწერებს ეძახიან. საჭიროა ელემენტთან მიღწევისათვის (სიტყვასთან ან ბაიტთან) აუცილებელია წაკითხულ იქნას მის წინ გასული ყველა მონაცემი. წვდომის დრო დამოკიდებულია საჭირო ჩანაწერის მდებარეობით ინფორმაციის მატარებელზე ჩანაწერთა თანმიმდევრობაში და მოცემული ჩანაწერის შიგნით ელემენტის პოზიციაზე. ამის მაგალითია მეხსიერების მოწყობილობა მაგნიტურ ლენტზე.

- პირდაპირი წვდომა. ყოველ ჩანაწერს გააჩნია უნიკალური მისამართი, რომელიც ასახავს მის ფიზიკურ მდებარეობას ინფორმაციის მატარებელზე. მიმართვა ხორციელდება როგორც სამისამართო წვდომა ჩანაწერის დასაწყისთან, შემდგომში მიმდევრობითი მიღწევით ჩანაწერის შიგნით ინფორმაციის განსაზღვრულ ერთეულთან. შედეგად გარკვეულ პოზიციასთან წვდომის დრო ცვლად სიდიდეს წარმოადგენს. ასეთი რეჟიმი მაგნიტური დისკებისთვისაა დამახსოვრებული.

- ნებისმიერი წვდომა. მეხსიერების ყოველ უჯრედს გააჩნია უნიკალური ფიზიკური მისამართი. ნებისმიერ უჯრედზე მიმართვას ესაჭიროება ერთი და იგივე დრო და შეიძლება სრულდებოდეს ნებისმიერი რიგით. ამის მაგალითია ძირითადი მეხსიერების მოწყობილობა.

- ასოციატიური წვდომა. მიღწევის ეს სახე საშუალებას იძლევა შესრულდეს უჯრედთა ძიება, რომლებიც შეიცავენ ისეთ ინფორმაციას, რომელშიც ცალკეული ბიტების მნიშვნელობა ემთხვევა მოცემულ ნიმუშში იგივე სახელის მქონე ბიტების მდგომარეობას. შედარება ხორციელდება პარალელურად მეხსიერების ყველა უჯრედი-

სათვის, დამოუკიდებლად მისი ტევადობისა. ასეთი პრინციპითაა აგებული კეშ-მეხსიერების ზოგიერთი ბლოკი.

მეხსიერების მოწყობილობის **სწრაფქმედება** არის მისი ერთ-ერთი უმნიშვნელოვანესი მახასიათებელი. სწრაფქმედების რაოდენობრივი შეფასებისათვის ჩვეულებრივ სამ პარამეტრს იყენებენ.

- **წვდომის დრო** ($T_წ$). მეხსიერებისათვის ნებისმიერი წვდომით ის შეესაბამება დროის ინტერვალს მისამართის მოსვლის მომენტიდან იმ მომენტამდე, როცა მონაცემები შეაქვთ მეხსიერებაში ან მისაღწევნი ხდებიან. მეხსიერების მოწყობილობაში ინფორმაციის მოძრავი მატარებლით - ეს საჭირო პოზიციაში ჩაწერა/ წაკითხვის თავაკის (ან მატარებლის) დაყენების დროა.

- **მეხსიერების ციკლის ხანგრძლივობა ან მიმართვის პერიოდი** ($T_ც$). ცნება გამოიყენება მეხსიერებისათვის ნებისმიერი წვდომით, რომლისათვისაც ის ნიშნავს მინიმალურ დროს მეხსიერებისადმი ორ მიმდევრობით მიმართვას შორის. მიმართვის პერიოდი შედგება წვდომის დროისა და კიდევ რაღაც დამატებითი დროისაგან. დამატებითი დრო შეიძლება საჭირო იყოს ხაზებში სიგნალების მილევისათვის, ხოლო ზოგიერთი ტიპის მეხსიერების მოწყობილობებში, რომლებშიც ინფორმაციის ამოკითხვა მის განადგურებას იწვევს, - ამოკითხული ინფორმაციის აღდგენისათვის.

- **გადაცემის სისწრაფე.** ეს ის სისწრაფეა, რომლითაც მონაცემები შეიძლება გადაეცემოდნენ მეხსიერებას ან პირიქით. მეხსიერებისათვის ნებისმიერი შეღწევით ის $\frac{1}{T_ც}$ - ს ტოლია სხვა სახის მეხსიერებისათვის გადაცემის სისწრაფე განისაზღვრება გამოსახულებით:

$$T_N = T_A + \frac{N}{R},$$

სადაც T_N – N ბიტის წაკითხვის ან ჩაწერის საშუალო დროა;

T_A - შეღწევის საშუალო დრო;

R - გადაგზავნის სიჩქარე (ბიტებისა წამში).

ვსაუბრობთ რა მეხსიერების მოწყობილობის ფ ი ზ ი კ უ რ ტ ი პ ზ ე, აუცილებელია მოვიხსენიოთ მეხსიერების მოწყობილობის სამი ყველაზე მეტად გავრცელებული ტექნოლოგია - ესაა მეხსიერება ნახევარგამტარებზე, მეხსიერება ინფორმაციის მაგნიტურ მატარებელზე, გამოყენებული მაგნიტურ დისკებსა და ლენტებზე და მეხსიერება ოპტიკურ მატარებელზე - ოპტიკური დისკები.

გამოყენებული ტექნოლოგიებისაგან დამოკიდებულებით საჭიროა გავითვალისწინოთ მეხსიერების მოწყობილობების რიგი ფ ი ზ ი კ უ რ ი თ ა ვ ი ს ე ბ უ რ ე ბ ე ბ ი, მაგალითად ენერგოდამოკიდებულება. ენერგოდამოკიდებულ მეხსიერებაში ინფორმაცია შეიძლება დამახინჯდეს ან დაიკარგოს კვების წყაროს გათიშვისას. ენერგოდამოუკიდებელ მეხსიერების მოწყობილობებში ჩაწერილი ინფორმაცია არ იშლება კვების წყაროს გათიშვის დროსაც კი. მაგნიტური და ოპტიკური მეხსიერებები - ენერგოდამოუკიდებლებია. ნახევარგამტარული მეხსიერება შეიძლება როგორც ენერგოდამოკიდებული, ისე ენერგოდამოუკიდებელი იყოს მისი ტიპისაგან დამოკიდებულებით. ენერგოდამოკიდებულების გარდა გათვალისწინებული უნდა იყოს ის გარემოება, რომ ინფორმაციის ამოკითხვა იწვევს თუ არა მის განადგურებას.

მეხსიერების მოწყობილობის ღ ი რ ე ბ უ ლ ე ბ ა მიღებულია შეფასდეს მეხსიერების მოწყობილობის საერთო ღირებულების ფარდობით მას ტევადობასთან ბიტებში, ანუ ერთი ბიტი ინფორმაციის შენახვის ღირებულება.

5.2. მეხსიერების მოწყობილობათა იერარქია

მეხსიერებას ფონ-ნეიმანისეულ გამომთვლელ მანქანაში ხშირად „ვიწრო ადგილს“ ეძახიან პროცესორთან შედარებით სწრაფქმედების

სერიოზული ჩამორჩენის გამო. ამასთან ეს გარღვევა განუწყვეტლივ იზრდება.

თუ გავაანალიზებთ დღეისათვის გამოყენებული მეხსიერების მოწყობილობათა ტიპებს, შევამჩნევთ შემდეგ კანონზომიერებას:

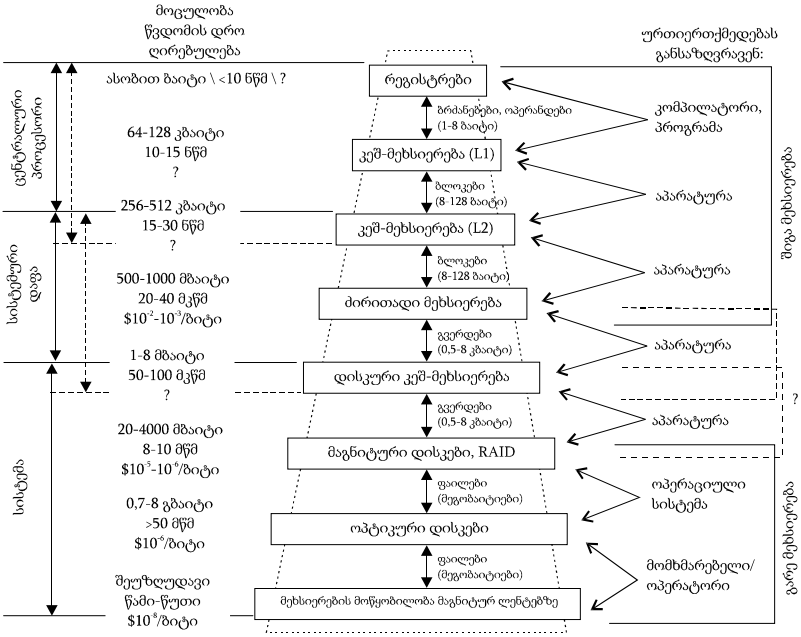
- რაც უფრო ნაკლებია წვდომის დრო, მით უფრო მაღალია ბიტის შენახვის ღირებულება;

- რაც მეტია ტევადობა, მით ნაკლებია ბიტის შენახვის ღირებულება, მაგრამ მეტია წვდომის დრო.

მეხსიერების სისტემის შექმნისას ყოველთვის ხდება მისაღები ფასით მოთხოვნილი ტევადობისა და მაღალი სწრაფქმედების უზრუნველყოფის ამოცანის გადაწყვეტა. აქ ყველაზე მეტად გავრცელებულ მიდგომას წარმოადგენს გამომთვლელი მანქანის მეხსიერების სისტემის იერარქიული პრინციპით აგება. იერარქიული მეხსიერება შედგება სხვადასხვა ტიპის მეხსიერების მოწყობილობებისაგან (ნახ. 5.1), რომლებსაც, მახასიათებლებისაგან დამოკიდებულებით, აკუთვნებენ იერარქიის განსაზღვრულ დონეს. ყველაზე მაღალი დონე ნაკლები ტევადობისაა, უფრო სწრაფია და გააჩნია მაღალი ღირებულება ბიტზე გადაანგარიშებით, ვიდრე დაბალ დონეს. იერარქიის დონეები ურთიერთდაკავშირებულია: ერთ დონეზე ყველა მონაცემი ასევე შეიძლება მოძიებულ იქნან უფრო დაბალ დონეზე, და ყველა მონაცემი ამ უფრო დაბალ დონეზე შეიძლება ნაპოვნი იყოს შემდეგ უფრო დაბალ დონეზე და ა. შ.

იერარქიის ზედა ოთხი დონე ქმნის გამომთვლელი მანქანის შიგა მეხსიერებას, ხოლო ქვედა დონე - გარე ანუ მეორადი მეხსიერებაა. იერარქიულ სტრუქტურაში ქვემოთ მოძრაობის მიხედვით:

1. მცირდება ფარდობა „ღირებულება/ბიტი“;
2. იზრდება ტევადობა;
3. იზრდება წვდომის დრო;
4. მცირდება ცენტრალური პროცესორის მხრიდან მეხსიერებაზე მიმართვის სიხშირე.



ნახ. 5.1. დამახსოვრების მოწყობილობათა იერარქია

თუ მეხსიერება ორგანიზებულია 1-3 პუნქტების შესაბამისად, ხოლო მონაცემებისა და ბრძანებების მასში განლაგების ხასიათი აკმაყოფილებს მე-4 პუნქტს, მაშინ იერარქიულ ორგანიზაციას მივყავართ საერთო ღირებულების შემცირებისაკენ წარმადობის მოცემული დონის პირობებში.

იერარქიის თითოეულ დონეზე ინფორმაცია იყოფა ბლოკებად, რომლებიც წარმოადგენენ იერარქიის ორ მეზობელ დონეს შორის გადაგზავნილ უმცირეს ინფორმაციულ ერთეულებს. ბლოკების ზომა შეიძლება როგორც ფიქსირებული ისე ცვლადი იყოს. ბლოკის ფიქსირებული ზომის დროს მეხსიერების ტევადობა ჩვეულებრივ მისი ზომის ჯერადია. იერარქიის თითოეულ დონეზე ბლოკთა ზომა უფრო ხშირად განსხვავებულია და იზრდება ზედა დონეებიდან ქვედასაკენ.

ყველაზე სწრაფი, მაგრამ ტევადობით მინიმალური მეხსიერების ტიპია ცენტრალური პროცესორის რეგისტრები, რომლებსაც ზოგჯერ აერთიანებენ ცნებით „ზ ე ო პ ე რ ა ტ ი უ ლ ი მ ე ხ ს ი - ე რ ე ბ ი ს მ ო წ ყ ო ბ ი ლ ო ბ ა“. როგორც წესი, მასში რეგისტრების რაოდენობა მცირეა, თუმცა არქიტექტურებში ბრძანებათა შემცირებული ნაკრებით მათმა რიცხვმა შეიძლება რამდენიმე ასეულს მიაღწიოს. მნიშვნელოვნად დიდი ტევადობის ძირითადი მეხსიერება მოთავსებულია რამდენიმე დონით ქვემოთ. ცენტრალური პროცესორის რეგისტრებსა და ძირითად მეხსიერებას შორის ხშირად ათავსებენ კეშ-მეხსიერებას, რომელიც ტევადობით შესამჩნევად აგებს ძირითად მეხსიერებასთან, მაგრამ მნიშვნელოვნად უკეთესია ამ უკანასკნელზე სწრაფქმედებით. ის იგივე პარამეტრით აგებს ზეოპერატიულ მეხსიერების მოწყობილობასთან. თანამედროვე გამომთვლელი მანქანების უმრავლესობაში არის კეშ-მეხსიერების რამდენიმე დონე, რომლებიც L ასოთი და კეშ-მეხსიერების დონის ნომრით აღინიშნებიან. ნახ. 5.1-ზე მოყვანილია ორი ასეთი დონე. ბოლო დამუშავებებში უფრო ხშირად მოჰყავთ კეშ-მეხსიერების მესამე დონე (L3), ამასთან გამომთვლელი მანქანის დამმუშავებლები საუბრობენ მეოთხე დონის - L4 შემოტანის მიზანშეწონილობის შესახებ. კეშ-მეხსიერების ყოველ მომდევნო დონეს აქვს უფრო დიდი ტევადობა, მაგრამ ერთდროულად ნაკლები სწრაფქმედება წინამორბედთან შედარებით. როგორც არ უნდა იყოს, სწრაფქმედებით კეშ-მეხსიერების ნებისმიერი დონე აჭარბებს ძირითად მეხსიერებას. შიგა მეხსიერების ყველა სახე რეალიზებულია ნახევარგამტარული ტექნოლოგიების საფუძველზე და ძირითადად ენერგოდამოკიდებულები არიან.

ინფორმაციის (პროგრამებისა და მონაცემების) დიდი მოცულობების ხანგრძლივი შენახვა უზრუნველყოფილია გარე მეხსიერების მოწყობილობებით, რომელთა შორის ყველაზე მეტადაა გავრცელებული მეხსიერების მოწყობილობები მაგნიტური და ოპტიკური დისკების საფუძველზე, აგრეთვე მაგნიტოლენტური მეხსიერების მოწყობილობები.

და ბოლოს, იერარქიის კიდევ ერთი დონე შეიძლება იქნეს დამატებული ძირითად მეხსიერებასა და დისკებს შორის. ამ დონეს დისკუ-

რი კეშ-მეხსიერება ჰქვია. ის რეალიზდება დამოუკიდებელი მეხსიერების მოწყობილობის სახით და შედის მაგნიტური დისკის შემადგენლობაში. დისკური კეშ-მეხსიერება მნიშვნელოვნად აუმჯობესებს წარმადობას დისკსა და ძირითად მეხსიერებას შორის ინფორმაციის გაცვლის დროს.

იერარქია შეიძლება შევსებულ იქნას მეხსიერების სხვა სახეებით. ასე მაგალითად, ფირმა IBM- ის გამომთვლელი მანქანის ზოგიერთი მოდელი შეიცავს ე. წ. გაფართოებულ მეხსიერებას (expanded storage), შესრულებულს ნახევარგამტარული ტექნოლოგიის საფუძველზე, ოღონდ ძირითად მეხსიერებასთან შედარებით გააჩნია ნაკლები სწრაფქმედება და ღირებულება. მკაცრად რომ ვიმსჯელოთ, მეხსიერების ეს სახე არ შედის იერარქიაში, არამედ წარმოადგენს მისგან განშტოებას, ვინაიდან მონაცემები შეიძლება გადაცემულ იქნან მხოლოდ გაფართოებულ და ძირითად მეხსიერებას შორის, მაგრამ დაუშვებელია გაცვლა გაფართოებულსა და გარე მეხსიერებას შორის.

5.3. ძირითადი მეხსიერება

ძ ი რ ი თ ა დ ი მ ე ხ ს ი ე რ ე ბ ა მეხსიერების ის ერთადერთი სახეა, რომელთანაც ცენტრალურ პროცესორს შეუძლია უშუალოდ მიმართვა (გამონაკლის წარმოადგენს მხოლოდ ცენტრალური პროცესორის რეგისტრები). გარე მეხსიერების მოწყობილობაში შენახული ინფორმაცია პროცესორისათვის მისაღწევი ხდება მხოლოდ მას შემდეგ, როცა ის ძირითად მეხსიერებაში გადაიწერება.

ძირითად მეხსიერებას ქმნიან მეხსიერების მოწყობილობები ნებისმიერი წვდომით. ასეთი მეხსიერების მოწყობილობა შექმნილია რო-

გორც უჯრედთა მასივი, ხოლო „ნებისმიერი წვდომა“ ნიშნავს, რომ ნებისმიერ უჯრედზე მიმართვას ერთი და იგივე დრო სჭირდება და შეიძლება ნებისმიერი თანმიმდევრობით შესრულდეს. თითოეული უჯრედი შეიცავს დამახსოვრების ელემენტების ფიქსირებულ რიცხვს და უნიკალური მისამართი გააჩნია, რომლის საშუალებითაც ხორციელდება უჯრედის გარჩევა ჩაწერის ან წაკითხვის ოპერაციების შესასრულებლად მათზე მიმართვის დროს.

ნახევრადგამტარული ტექნოლოგიების სფეროში მიღწეული უდიდესი წარმატების შედეგია ძირითადი მეხსიერების ელემენტური ბაზის შეცვლა. ფერომაგნიტური რგოლების საფუძველზე აგებული მეხსიერების მოწყობილობები ყველგან ნახევარგამტარული მიკროსქემებით შეიცვალა.

ძირითადი მეხსიერება შეიძლება შედგებოდეს ორი ტიპის მოწყობილობებისაგან: ოპერატიული მეხსიერების მოწყობილობებისაგან და მუდმივი მეხსიერების მოწყობილობისაგან.

ძირითადი მეხსიერების უდიდეს ნაწილს ოპერატიული მეხსიერების მოწყობილობა წარმოადგენს, რომელსაც ოპერატიული იმიტომ ჰქვია, რომ ის უშვებს ინფორმაციის როგორც ჩაწერას, ისე წაკითხვას. ამასთან ორივე ოპერაცია სრულდება ერთტიპიურად (ერთნაირად), პრაქტიკულად ერთი და იგივეა სიჩქარით და ხორციელდება ელექტრული სიგნალების დახმარებით. ინგლისურ ენოვან ლიტერატურაში ოპერატიულ მეხსიერების მოწყობილობას შეესაბამება აბრევიატურა RAM –Random Access Memory, ანუ „მეხსიერება ნებისმიერი წვდომით“, რაც სულ მთლად კორექტული არაა, ვინაიდან მუდმივი მეხსიერების მოწყობილობაც და პროცესორის რეგისტრებიც წარმოადგენენ მეხსიერებას ნებისმიერი წვდომით. ნახევარგამტარული ოპერატიული მეხსიერების მოწყობილობების უმრავლესობისათვის დამახასიათებელია ენერგოდამოკიდებულება - კვების ხანმოკლე შეწყვეტის დროსაც კი ინფორმაცია იკარგება. ოპერატიული მეხსიერების მოწყობილობის მიკ-

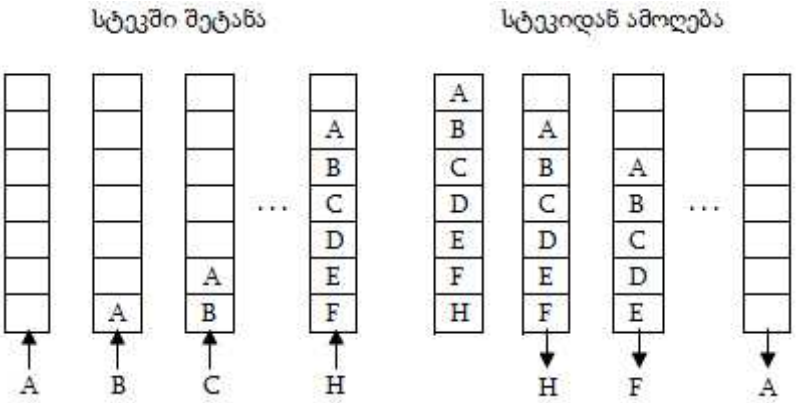
როსქემა მუდმივად უნდა იყოს მიერთებული კვების წყაროსთან და ამიტომ შეიძლება გამოიყენებოდეს, როგორც დროებითი მეხსიერება.

ძირითადი მეხსიერების ნახევარგამტარული მეხსიერების მოწყობილობების მეორე ჯგუფს ქმნიან მუდმივი მეხსიერების მოწყობილობების ენერგოდამოკიდებული მიკროსქემები (ROM – Rend – Only Memory). მუდმივი მეხსიერების მოწყობილობა უზრუნველყოფს ინფორმაციის წაკითხვას, მაგრამ არ უშვებს მის შეცვლას.

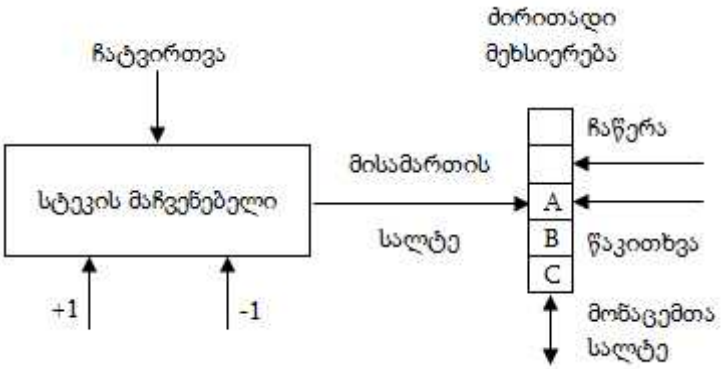
5.4. სტეკური მეხსიერება

სტეკური მეხსიერება უზრუნველყოფს მუშაობის ისეთ რეჟიმს, როცა ინფორმაცია ჩაიწერება და ამოიკითხება პრინციპით „ბოლო ჩაიწერა - პირველი ამოიკითხა (წაიკითხა)“ (LIFO – Last In First Out). ასეთი ორგანიზაციის მეხსიერება ფართოდ გამოიყენება პროცესორის რეგისტრების შემცველობის (კონტექსტის) დამახსოვრებისა და აღდგენისათვის პროგრამებისა და წყვეტების დამუშავების დროს. სტეკური მეხსიერების მუშაობას განმარტავს ნახ. 5.2,ა.

როცა A სიტყვა შეიტანება სტეკში, ის თავსდება პირველ თავისუფალ უჯრედში. ყოველი მომდევნო ჩასაწერი სიტყვა გადააადგილებს სტეკის მთელ შემცველობას ერთი უჯრედით ზემოთ და იკავებს გამოთავისუფლებულ უჯრედს. H -ის მომდევნო კოდის ჩაწერა იწვევს სტეკის გადავსებას და A კოდის დაკარგვას. კოდების სტეკიდან ამოკითხვა ხორციელდება უკუ რიგით, ანუ იწყება H კოდით, რომელიც ბოლოს იყო ჩაწერილი. აღვნიშნოთ, რომ სტეკში ნებისმიერ კოდთან მიღწევა ფორმალურად აკრძალულია ყველა იმ კოდის ამოღებამდე, რომლებიც უფრო გვიან იყვნენ ჩაწერილი.



ა)



ბ)

ნახ. 5.2. სტეკური მეხსიერების ორგანიზაცია : ა) -მუშაობის ლოგიკა; ბ) - აპარატურულ -პროგრამული სტეკი.

დღეისათვის ყველაზე გავრცელებულია გარე ან აპარატურულ პროგრამული სტეკი, რომელშიც ინფორმაციის შესატანად გამოყოფილია ძირითადი მეხსიერების ზონა. ჩვეულებრივ ამ მიზნებისათვის გა-

მოიყოფა ყველაზე დიდ მისამართებიანი მეხსიერების უბანი, ხოლო სტეკი ფართოვდება მისამართების შემცირების მიმართულებით. ვინაიდან ჩვეულებრივ პროგრამა იტვირთება მცირე მისამართებიდან დაწყებული, ამიტომ ასეთი ხერხი ბევრ შემთხვევაში საშუალებას იძლევა აცილებულ იქნას პროგრამისა და სტეკის ზონების გადაფარვა. სტეკის დამისამართება უზრუნველყოფილია სპეციალური რეგისტრით - სტეკის მაჩვენებლით (SP – Stack pointer), რომელშიც წინასწარ თავსდება სტეკისათვის გამოყოფილი ძირითადი მეხსიერების ზონის უდიდესი მისამართი (ნახ. 5.2,ბ).

სტეკში მომდევნო ელემენტის შეტანისას ჯერ ხდება სტეკის მაჩვენებლის შემცველობის ერთი ერთეულით შემცირება, რომელიც შემდეგ გამოიყენება იმ უჯრედზე მისამართავად, სადაც ხდება ჩაწერა. ანუ სტეკის მაჩვენებელი ინახავს იმ უჯრედის მისამართს, რომელზეც იყო განხორციელებული ბოლო მიმართვა.

სტეკიდან სიტყვის ამოკითხვისას ამ სიტყვის მისამართად იღება სტეკის მაჩვენებლის მიმდინარე შემცველობა, ხოლო სიტყვის ამოღების შემდეგ, სტეკის მაჩვენებლის შემცველობა ერთი ერთეულით იზრდება.

5.5. ასოციაციური მეხსიერება

ზემოთ განხილულ მეხსიერების მოწყობილობებში ინფორმაციასთან მიღწევა მოითხოვდა უჯრედის მისამართის მითითებას. ხშირად ბევრად უფრო მოსახერხებელია ინფორმაციის ძიება არა მისამართით, არამედ თავად ინფორმაციაში არსებულ რაიმე დამახასიათებელ ნიშანზე დაყრდნობით. ასეთი პრინციპი უდევს საფუძვლად ასოციაციური მეხსიერების მოწყობილობას. ლიტერატურაში გვხვდება მსგავსი მეხსიერების მოწყობილობების სხვა დასახელებებიც:

შემცველობით დამისამართებული მეხსიერება (content addressable memory); მონაცემებით დამისამართებული მეხსიერება (data addressable memory); მეხსიერება პარალელური ძიებით (parallel search memory); კატალოგიური მეხსიერება (catalog memory); ინფორმაციული მეხსიერების მოწყობილობა (information storage); ტეგირებული მეხსიერება (tag memory). ასოციაციური მეხსიერების მოწყობილობა ისეთი მოწყობილობაა, რომელსაც შეუძლია შეინახოს ინფორმაცია, შეადაროს ის რომელიმე მოცემულ ნიმუშს და მიუთითოს მათი ერთმანეთთან შესატყვისობისა ან განსხვავების შესახებ. ნიშანს, რომლის მიხედვითაც ხდება ინფორმაციის ძებნა, ჰქვია ასოციაციური ნიშანი, ხოლო კოდურ კომბინაციას, რომელიც გამოდის ძიებისათვის ნიმუშის როლში, - ძიების ნიშანი. ასოციაციური ნიშანი შეიძლება საძიებელი ინფორმაციის ნაწილი იყოს ან დამატებით ემატებოდეს მას. უკანასკნელ შემთხვევაში მიღებულია მას ვუწოდოთ ტეგი ან ირლიყი.

ასოციაციური მეხსიერების აგების ერთ-ერთი ვარიანტი მოტანილია ნახ. 5.3-ზე.

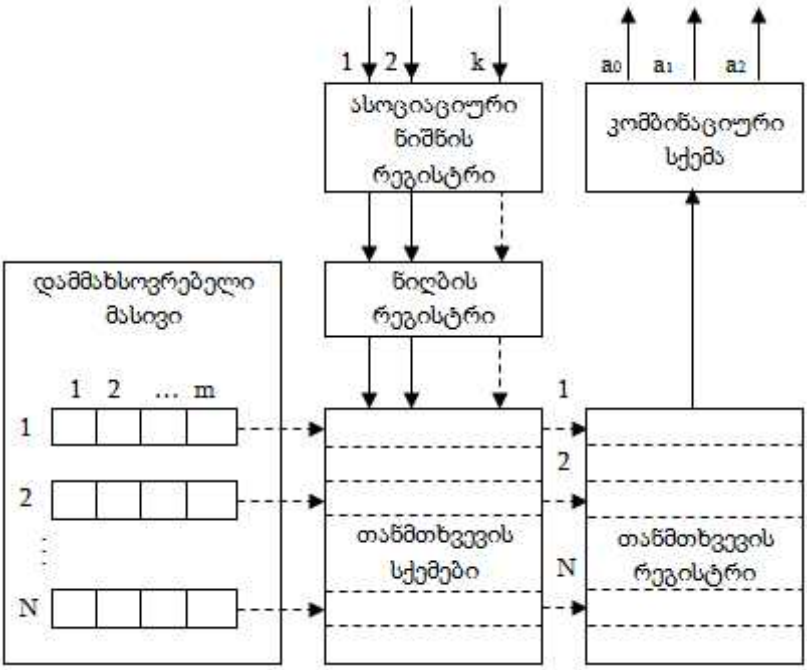
ასოციაციური დამახსოვრების მოწყობილობა შეიცავს:

- დამახსოვრებულ მასივს, რომლის დანიშნულებაა m - თანრიგა N სიტყვის შენახვა, რომელთაგან თითოეულში რამდენიმე უმცროსი თანრიგი უჭირავს სამსახურებლივ ინფორმაციას;

- ასოციაციური ნიშნის რეგისტრს, სადაც თავსდება საძიებელი ინფორმაციის კოდი (ძიების ნიშანი) რეგისტრის K თანრიგობა ჩვეულებრივ ნაკლებია სიტყვის m სიგრძეზე;

- თანხვედრის სქემებს, რომლებიც გამოიყენებიან ყველა შენახული სიტყვის თითოეული ბიტის პარალელური შედარებებისათვის ძიების ნიშნის შესაბამის ბიტთან და თანხვედრის სიგნალების გამომუშავებისათვის;

- თანხვედრის რეგისტრს, სადაც დამახსოვრების მასივის თითოეულ უჯრედს შეესაბამება ერთი თანრიგი, რომელშიც შეაქვთ ერთიანი, თუ შესაბამისი უჯრედის ყველა თანრიგი დაემთხვა ძიების ნიმუშის ერთი და იგივე სახელის მქონე თანრიგებს;



ნახ. 5.3. ასოციაციური დამახსოვრების მეხსიერების სტრუქტურა

- ნიღაბის რეგისტრს, რომლის საშუალებითაც იკრძალება განსაზღვრული ბიტების შედარება;

- კომბინაციურ სქემას, რომელიც თანხვედრის რეგისტრის შემცველობის ანალიზის საფუძველზე აფორმირებს ინფორმაციის ძიების შედეგების მახასიათებელ სიგნალებს.

ასოციაციური დამახსოვრების მოწყობილობაზე მიმართვისას ჯერ ნიღბის რეგისტრში ხდება იმ თანრიგების განულება, რომლებიც არ უნდა იყვნენ გათვალისწინებულნი ინფორმაციის ძიებისას. თანხვედრის რეგისტრის ყველა თანრიგი დგება ერთის მდგომარეობაში. ამის შემდეგ ასოციაციური ნიმუშის რეგისტრში შეიტანება საძიებელი ინფორმაციის კოდი (ძიების ნიშანი) და იწყება მისი ძიება, რომლის პროცესში თანხვედრის სქემები ერთდროულად ადარებენ დამმახსოვ-

რებელი მასივის ყველა უჯრედის პირველ ბიტს ძიების ნიშნის პირველ ბიტს. ის სქემები, რომლებმაც დააფიქსირეს არა თანხვედრა, აფორმირებენ სიგნალს, რომელსაც თანხვედრის რეგისტრის შესაბამისი ბიტი გადაჰყავს ნულის მდგომარეობაში. ასევე სრულდება ძიების პროცესი ძიების ნიშნის სხვა არა შენიღბული ბიტებისათვის. შედეგად ერთიანები რჩებიან თანხვედრის რეგისტრის მხოლოდ იმ თანრიგებში, რომლებიც შეესაბამებიან იმ უჯრედებს, სადაც იმყოფება საძიებელი ინფორმაცია. თანხვედრის რეგისტრში ერთიანების კონფიგურაცია გამოიყენება მისამართებად, რომელთა მიხედვითაც ხდება ამოკითხვა დამახსოვრებული მასივიდან.

იმის გამო, რომ ძიების შედეგები შეიძლება არაერთმნიშვნელოვანნი აღმოჩნდნენ, თანხვედრის რეგისტრის შემცველობა მიეწოდება კომბინაციურ სქემას, სადაც ფორმირდებიან სიგნალები, რომლებიც იტყობინებიან იმის შესახებ, რომ საძიებელი ინფორმაცია:

- $a0$ - არაა ნაპოვნი;
- $a1$ - არის ერთ უჯრედში;
- $a2$ - არის ერთზე მეტ უჯრედში.

თანხვედრის რეგისტრის შემცველობისა და $a0, a1, a2$ სიგნალების ფორმირება ატარებს ასოციაციის კონტროლის ოპერაციის დასახელებას. ის წაკითხვისა და ჩაწერის ოპერაციათა შემადგენელი ნაწილია, თუმცა შეიძლება გააჩნდეს დამოუკიდებელი მნიშვნელობაც.

შეკითხვისას ჯერ ხდება ასოციაციის კონტროლი ძიების არგუმენტის მიხედვით. შემდეგ, თუ $a0 = 1$, მაშინ წაკითხვა უქმდება საძიებელი ინფორმაციის არ არსებობის გამო, თუ $a1 = 1$, მაშინ ხდება სიტყვის წაკითხვა, რომელზეც თანხვედრის რეგისტრში უთითებს ერთიანი, ხოლო თუ $a2 = 1$, მაშინ თანხვედრის რეგისტრში ნულდება ყველაზე უფროსი ერთიანი და ამოიკითხება მისი შესაბამისი სიტყვა. ამ ოპერაციის გამეორებით თანმიმდევრობით შეიძლება ყველა სიტყვის ამოკითხვა.

ასოციაციურ დამახსოვრების მოწყობილობაში ჩაწერა ხორციელდება კონკრეტული მისამართის მითითების გარეშე პირველ თავისუფალ უჯრედში. თავისუფალი უჯრედის მოსაძებნად ხორციელდება ამოკითხვის ოპერაცია, რომელშიც შენიღბული არაა მხოლოდ სამომსახურეო თანრიგები, რომლებიც უჩვენებენ, თუ რამდენად დიდი ხნის უკან ხდებოდა მოცემულ უჯრედზე მიმართვა, და თავისუფლად ითვლება ან ცარიელი უჯრედი, ან ის, რომელიც ყველაზე დიდი ხნის განმავლობაში არ გამოიყენებოდა.

ასოციაციური დამახსოვრების მოწყობილობის მთავარი უპირატესობა იმაში მდგომარეობს, რომ ინფორმაციის ძიების დრო დამოკიდებულია მხოლოდ ძიების ნიშანში თანრიგების რაოდენობასა და თანრიგების გამოკითხვის სიჩქარეზე და არაა დამოკიდებული დამახსოვრებულ მასივში უჯრედის რიცხვზე.

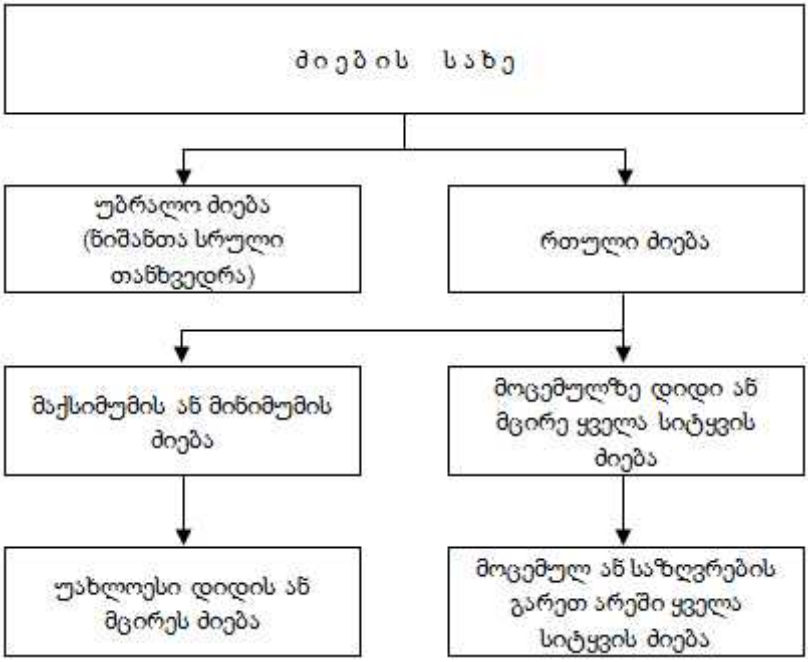
ინფორმაციის ასოციაციური ძიების იდეა სრულებითაც არ გამორიცხავს ასოციაციური დამახსოვრების მოწყობილობის არქიტექტურათა მრავალფეროვნებას. კონკრეტული არქიტექტურა განისაზღვრება ოთხი ფაქტორის შერწყმით: ინფორმაციის ძიების სახით; ნიშნების შედარების ტექნიკით; მრავალჯერადი თანხვედრებისას ინფორმაციის ამოკითხვის ხერხით და ინფორმაციის ჩაწერის ხერხით.

ასოციაციური დამახსოვრების მოწყობილობის ყოველი კონკრეტული გამოყენებისას ინფორმაციის ძიების ამოცანა შეიძლება სხვადასხვაგვარად იყოს ჩამოყალიბებული (ნახ. 5.4).

მარტივი (უბრალო) ძიების დროს მოითხოვება ძიების ნიშნის ყველა თანრიგის სრული თანხვედრა დამახსოვრებულ მასივში შენახული სიტყვების ერთი და იგივე სახელის მქონე თანრიგებთან.

შესაბამისი ორგანიზაციის პირობებში ასოციაციურ დამახსოვრების მოწყობილობას შეუძლია ძიების რთული ვარიანტების განხორციელება, მაგალითად ნაწილობრივი თანხვედრისას. მაგალითად, შეიძლება დაისვას ასოციაციური ნიშნის მაქსიმალური ან მინიმალური მნიშვნელობით სიტყვების ძიების ამოცანის დასმა. ასოციაციური დამახსოვრების მოწყობილობიდან სიტყვის მრავალჯერადი ამორჩევა

ასოციაციური ნიშნის მაქსიმალური ან მინიმალური მნიშვნელობით (შემდგომი ძიებიდან მისი გამორიცხვით) არსებითად წარმოადგენს ინ-ფორმაციის მოწესრიგებულ ამორჩევას. მოწესრიგებული ამორჩევა (ამონაკრები) შეიძლება სხვა ხერხითაც იყოს უზრუნველყოფილი, თუ მოვძებნით ისეთ სიტყვებს, რომელთა ასოციაციური ნიშანი გამოკითხ-ვის ნიშანთან მიმართებაში არის უახლესი დიდი ან მცირე მნიშვნელო-ბა.



ნახ. 5.4. დამახსოვრებულ მასივში ინფორმაციის ძიების სახის მიხედვით ასოციაციური დამახსოვრების მოწყობილობათა კლასიფიკაცია

რთული ძიების კიდევ ერთი ვარიანტი შეიძლება იყოს ისეთი სიტყვების ძიება, რომელთა ნიშნის რიცხობრივი მნიშვნელობა მოცე-მულ მნიშვნელობაზე მეტია ან ნაკლები. მსგავსი მიდგომა საშუალებას

იძლევა სიტყვები ისეთი ნიშნებით, რომლებიც იმყოფებიან მოცემული საზღვრების შიგნით ან გარეთ.

ცხადია, რომ ძიების რთული მეთოდების რეალიზაცია დაკავშირებულია ასოციაციური დამახსოვრების მოწყობილობის არქიტექტურაში შესაბამისი ცვლილებებით, კერძოდ დამახსოვრების მოწყობილობის სქემის გართულებით და მასში დამატებითი ლოგიკის შეტანით.

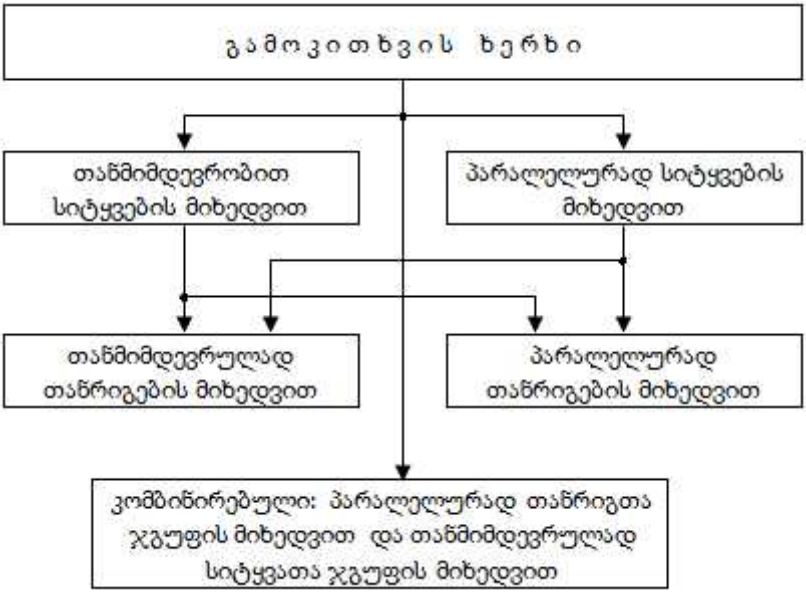
ასოციაციური დამახსოვრების მოწყობილობის აგების დროს არჩევანს აკეთებენ მეხსიერების შემცველობის გამოკითხვის ორგანიზაციის ოთხ ვარიანტს შორის (ნახ. 5.5). ეს ვარიანტები შეიძლება კომბინირდებოდნენ პარალელურად თანრიგთა ჯგუფის მიხედვით და მიმდევრობით ჯგუფების მიხედვით. ძიების დროის მხრივ ყველაზე ეფექტურად შეიძლება ჩაითვალოს პარალელური გამოკითხვა როგორც სიტყვების, ისე თანრიგების მიხედვით, მაგრამ დამამახსოვრებელ ელემენტთა არა ყველა სახე უშვებს ასეთ შესაძლებლობას.

მნიშვნელოვან მომენტს წარმოადგენს ასოციაციური დამახსოვრების მოწყობილობიდან ინფორმაციის ამოკითხვის ორგანიზაცია, როცა ძიების ნიშანს ემთხვევიან რამდენიმე სიტყვის ასოციაციური ნიშნები. ამ შემთხვევაში გამოიყენება ორიდან ერთი მიდგომა (ნახ. 5.6):

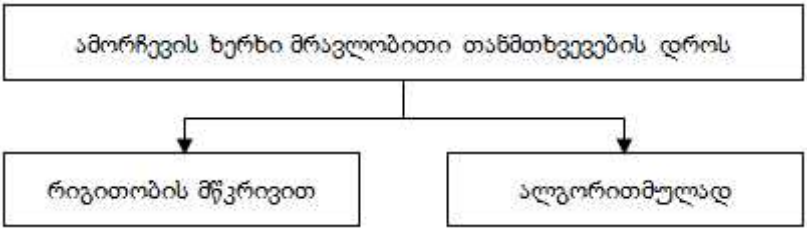
რიგობითი მწკრივი რეალიზებულია საკმაოდ რთული მოწყობილობით, რომელშიც ფიქსირდებიან მრავალმნიშვნელოვანი პასუხის შემქმნელი სიტყვები. რიგითობის მწკრივი (ჯაჭვი) საშუალებას იძლევა მოხდეს სიტყვების ამოკითხვა ასოციაციური დამახსოვრების მოწყობილობის უჯრედის ნომრის ზრდის მიხედვით ასოციაციური ნიშნების სიდიდისაგან დამოუკიდებლად.

მრავალმნიშვნელოვანი ხერხის დროს ამორჩევა ხორციელდება გამოკითხვათა სერიის შედეგებით. გამოკითხვათა სერია შეიძლება ფორმირდებოდეს იმ თანრიგთა მოწესრიგების გზით, რომლებიც შენიღბულები იყვნენ და არ მონაწილეობდნენ ძიების ნიშანში. მეორე ვარიანტში ამ მიზნებისათვის გამოიყოფიან სპეციალური უჯრედები. არსებობს ალგორითმთა მთელი რიგი, რომლებიც ახორციელებენ ასო-

ციაციური დამახსოვრების მოწყობილობიდან მოწესრიგებული ამონაკრეფის ამოღების ორგანიზაციას.



ნახ. 5.5. ასოციაციური დამახსოვრების მოწყობილობათა კლასიფიკაცია დამმახსოვრებელი მასივის შემცველობის გამოკითხვის ხერხის მიხედვით



ნახ. 5.6. ასოციაციური დამახსოვრების მოწყობილობათა კლასიფიკაცია თანმთხვეული სიტყვების ამოკრეფის ხერხის მიხედვით მრავალჯერადი თანმიმდევრობის დროს

ასოციაციური დამახსოვრების მოწყობილობის არქიტექტურაში მნიშვნელოვანი განსხვავებები შეიძლება დაკავშირებულნი იყვნენ ინფორმაციის ჩაწერის ამორჩეულ პრინციპთან. ზემოთ აღწერილი იყო უმცირესი ნომრის მქონე თავისუფალ უჯრედში ჩაწერის ვარიანტი. პრაქტიკაში გამოიყენება სხვა ხერხებიც (ნახ. 5.7), რომელთაგან ყველაზე რთულია - ჩაწერა ასოციაციური დამახსოვრების მოწყობილობის შესასვლელზე ინფორმაციის ნიშნის სიდიდის მიხედვით დახარისხებით. აქ იმ უჯრედის ადგილმდებარეობა, რომელშიც უნდა მოთავსდეს

ნახ. 5.7. ასოციაციური დამახსოვრების მოწყობილობათა კლასიფიკაცია ინფორმაციის ჩაწერის ხერხის მიხედვით

ახალი სიტყვა, დამოკიდებულია ახალი ჩასაწერი სიტყვისა და ასოციაციური დამახსოვრების მოწყობილობაში უკვე შენახული სიტყვების ასოციაციურობის ნიშნების დამოკიდებულებაზე.

შედარებით მაღალი ღირებულების გამო ასოციაციური დამახსოვრების მოწყობილობა იშვიათად გამოიყენება როგორც მეხსიერების დამოუკიდებელი სახე.

5.6. კეშ - მეხსიერება

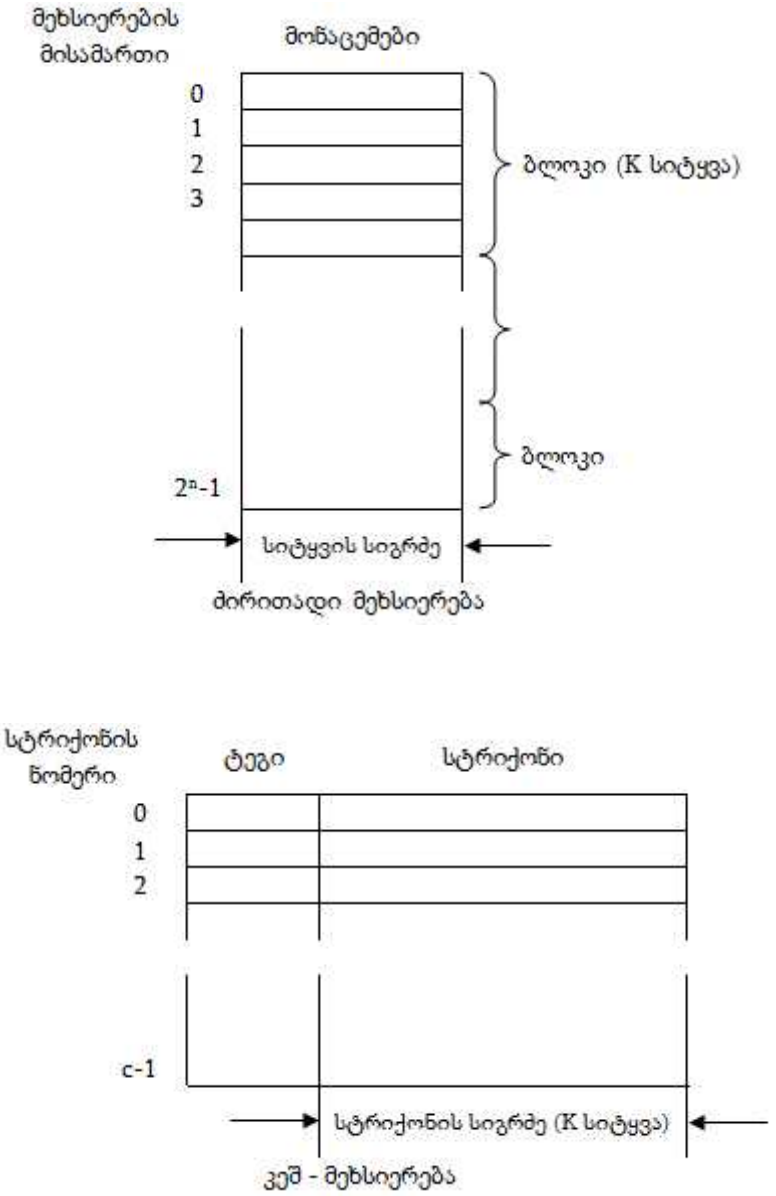
როგორც უკვე ავღნიშნეთ, ძირითადი მეხსიერების ელემენტთა ბაზად უმრავლეს გამომთვლელ მანქანაში გამოიყენებიან დინამიური ოპერატიული დამახსოვრების მოწყობილობების მიკროსქემები, რომლებიც სწრაფქმედებით მთელი რიგით ჩამორჩებიან ცენტრალურ პროცესორს. შედეგად პროცესორი იძულებულია უქმად იყოს რამდენიმე ტაქტური პერიოდის განმავლობაში, სანამ ინფორმაცია ინტეგრალური მიკროსქემიდან დადგება გამომთვლელი მანქანის მონაცემთა სალტეზე. თუ ძირითადი მეხსიერებას ავაგებთ სტატიკური მეხსიერების სწრაფ მიკროსქემებზე, მაშინ გამომთვლელი მანქანის ღირებულება მნიშვნელოვნად გაიზრდება. ამ პრობლემის ეკონომიურად მისაღები გადაწყვეტა შემოთავაზებული იყო მ. უილკსის მიერ 1965 წელს გამომთვლელი მანქანა Atlas-ის დამუშავებისას და მდგომარეობს ორდონიანი მეხსიერების გამოყენებაში, როცა ძირითად მეხსიერებას და პროცესორს შორის თავსდება მცირე, მაგრამ სწრაფი ბუფერული მეხსიერება. ასეთი სისტემის მუშაობის პროცესში ბუფერულ მეხსიერებაში ხდება ძირითადი მეხსიერების იმ უბნების კოპირება, რომლებზეც სრულდება მიმართვა პროცესორის მხრიდან. საერთოდ მიღებული ტერმინოლოგიით - ხდება ძირითადი მეხსიერების უბნების ასახვა ბუფერულ მეხსიერებაში. მოგება მიიღწევა ადრე განხილული ლოკალურობის თვისების ხარჯზე - თუ ძირითადი მეხსიერების უბანს ავსახავთ უფრო სწრაფ ბუფერულ მეხსიერებაში და მასზე გადავამისამართებთ ყველა მიმართვას გადაკოპირებული უბნის ფარგლებში, მაშინ შესაძლებელია მივაღწიოთ გამომთვლელი მანქანის წარმადობის მნიშვნელოვან ამაღლებას.

უილკსი განსახილველ ბუფერულ მეხსიერებას უწოდებდა დაქვემდებარებულს (Slave memory). მოგვიანებით გავრცელება ჰპოვა ტერმინმა კეშ - მეხსიერება (ინგლისური სიტყვიდან - cache - თავ-

შესაფარი, სამალავი), ვინაიდან ასეთი მეხსიერება ჩვეულებრივ დაფარულია პროგრამისტისაგან იმ აზრით, რომ მას არ შეუძლია მისი დამისამართება და შეიძლება საერთოდ არ იცოდეს მისი არსებობის შესახებ. პირველად კეშ - სისტემები გამოჩნდნენ IBM 360 ოჯახის 85 მოდელის მანქანებში.

საერთო სახით კეშ-მეხსიერების გამოყენება განვმარტოთ შემდეგი სახით. როცა ცენტრალური პროცესორი ცდილობს ძირითადი მეხსიერებიდან სიტყვის წაკითხვას, ჯერ ხორციელდება ამ სიტყვის ასლის მოძიება კეშში. თუ ასეთი ასლი არსებობს, მაშინ ძირითად მეხსიერებაზე მიმართვა არ წარმოებს, ხოლო ცენტრალურ პროცესორში გადაიცემა კეშ-მეხსიერებიდან ამოღებული სიტყვა. მიღებულია მოცემულ სიტუაციას უწოდონ წარმატებული მიმართვა ან მ ო ხ -ვ ე დ რ ა (hit). კეშში სიტყვის არ არსებობისას, ანუ არა წარმატებული მიმართვის დროს - ა ც ი ლ ე ბ ი ს ა ს (miss), - საჭირო სიტყვა ცენტრალურ პროცესორში გადაიცემა ძირითადი მეხსიერებიდან, მაგრამ ერთდროულად ძირითადი მეხსიერებიდან კეშ- მეხსიერებაში გადაიცემა მონაცემთა ბლოკი, რომელიც ამ სიტყვას შეიცავს.

ნახ. 5.8-ზე მოყვანილია სისტემის სტრუქტურა ძირითადი და კეშ-მეხსიერებით. ძირითადი მეხსიერება შესდგება 2^n დამისამართებული სიტყვისაგან, სადაც თითოეულ სიტყვას გააჩნია უნიკალური n - თანრიგიანი მისამართი. კეშთან ურთიერთქმედებისას ეს მეხსიერება განიხილება როგორც ფიქსირებული სიგრძის M ბლოკი თითოეულში K სიტყვით ($M=2^n/K$). კეშ- მეხსიერება შესდგება ანალოგიური ზომის C ბლოკისაგან (კეშ-მეხსიერებაში ბლოკებს მიღებულია ს ტ რ ი ქ ო ნ ე ბ ი ვუწოდოთ), ამასთან მათი რაოდენობა მნიშვნელოვნად ნაკლებია ძირითად მეხსიერებაში ბლოკების რაოდენობაზე ($C << M$). ძირითადი მეხსიერების რომელიმე ბლოკიდან სიტყვის ამოკითხვისას ეს ბლოკი კოპირდება კეშის ერთ-ერთ სტრიქონში. ვინაიდან ძირითადი მეხსიერების ბლოკების რაოდენობა მეტია სტრიქონების რაოდენობაზე, ამიტომ დამოუკიდებელი სტრიქონი არ შეიძლება მუდმივად იყოს გამოყოფილი ძირითადი მეხსიერების ერთი და იგივე ბლოკისათვის. ამ



ნახ. 5.8. სისტემის სტრუქტურა ძირითადი და კეშ-მეხსიერებით

მიზეზით კეშ-მეხსიერების თითოეულ სტრიქონს შეესაბამება ტ ე გ ი (ნიშანი), რომელიც შეიცავს ცნობებს იმის შესახებ, თუ ძირითადი მეხსიერების რომელი ბლოკის ასლი ინახება მოცემულ მომენტში მოცემულ სტრიქონში. ტეგად ჩვეულებრივ გამოიყენება ძირითადი მეხსიერების მისამართის ნაწილი.

მეხსიერების იერარქიულ სისტემაში კეშ-მეხსიერების ეფექტურ გამოყენებაზე მოქმედებს მომენტების მთელი რიგი. მათგან ყველაზე მნიშვნელოვნებს შეიძლება მივაკუთვნოთ:

- კეშ-მეხსიერების მოცულობა;
- სტრიქონის ზომა;
- ძირითადი მეხსიერების კეშ-მეხსიერებაზე ასახვის ხერხი;
- შევსებულ კეშ-მეხსიერებაში ინფორმაციის შენაცვლების ალგორითმი;
- კეშ-მეხსიერების დონეთა რიცხვი.

5.7. ვირტუალური მეხსიერება

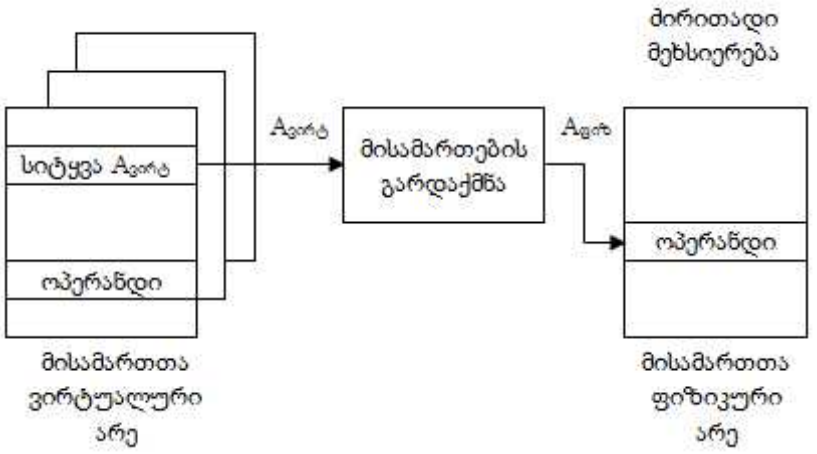
გამომთვლელი მანქანის ტიპიური გამოყენების უმრავლესობისათვის დამახასიათებელია სიტუაცია, როცა მთელი პროგრამის ძირითად მეხსიერებაში მოთავსება შეუძლებელია მისი დიდი ზომის გამო. თუმცა ამაში არც არსებობს პრინციპული აუცილებლობა, ვინაიდან დროის ყოველ მომენტში მანქანის „ყურადღება“ კონცენტრირდება პროგრამის განსაზღვრულ შედარებით მცირე უბნებზე. ამრიგად, ძირითად მეხსიერებაში საკმარისია შევინახოთ მხოლოდ მოცემულ მომენტში გამოყენებული პროგრამის ნაწილები, ხოლო დანარჩენი ნაწილები შეიძლება განლაგდნენ გარე დამახსოვრების მოწყობილობაში.

მსგავსი მიდგომის სირთულე იმაში მდგომარეობს, რომ ძირითად მეხსიერებაზე და გარე დამახსოვრების მოწყობილობაზე მიმართვის პროცესები მნიშვნელოვნად განსხვავდებიან, და ეს ართულებს პროგრამისტის ამოცანას. ასეთი სიტუაციიდან გამოსავალი გახდა 1959 წელს ვ ი რ ტ უ ა ლ უ რ ი მ ე ხ ს ი ე რ ე ბ ი ს იდეის გაჩენა. მის ქვეშ ესმით იერარქიული მეხსიერების ავტომატური მართვის მეთოდი, როდესაც პროგრამისტს ეჩვენება, რომ მას საქმე აქვს დიდი მოცულობისა და მაღალი სრაფქმედების ერთიან მეხსიერებასთან. ამ მეხსიერებას ვირტუალურს (მოჩვენებითს) უწოდებენ. თავისი არსით მეხსიერების გავირტუალება იერარქიული მეხსიერების კონცეფციის აპარატურული და პროგრამული რეალიზაციის ხერხს წარმოადგენს.

ვირტუალური მეხსიერების იდეის ჩარჩოებში ძირითადი მეხსიერება განიხილება როგორც N მისამართის წრფივი არე, წოდებული მეხსიერების ფ ი ზ ი კ უ რ ა რ ე დ. იმ ამოცანებისათვის, რომლებსაც N უჯრედზე მეტი ესაჭიროებათ, გამოიყოფა მისამართების მნიშვნელოვნად უფრო დიდი არე (ჩვეულებრივ ყველა სახის მეხსიერების საერთო მოცულობის ტოლი), წოდებული ვ ი რ ტ უ ა ლ უ რ ა რ ე დ, ზოგად შემთხვევაში არა აუცილებლად წრფივი. ვირტუალური არის მისამართებს ვ ი რ ტ უ ა ლ უ რ ე ბ ს ეძახიან, ხოლო ფიზიკური არის მისამართებს - ფ ი ზ ი კ უ რ ე ბ ს. პროგრამა იწერება ვირტუალურ მისამართებში, ვინაიდან მისი შესრულებისათვის საჭიროა, რომ დასამუშავებელი ბრძანებები და მონაცემები იმყოფებოდნენ ძირითად მეხსიერებაში, ამიტომ მოთხოვნილია, რომ თითოეულ ვირტუალურ მისამართს შეესაბამებოდეს ფიზიკური. ამრიგად, გამოთვლების პროცესში უპირველესად საჭიროა გარე დამახსოვრების მოწყობილობიდან ძირითად მეხსიერებაში ინფორმაციის იმ ნაწილის გადაწერა, რომელზეც მიუთითებს ვირტუალური მისამართი (ვირტუალური არის ფიზიკურად ასახვა), რის შემდეგაც საჭიროა ვირტუალური მისამართის ფიზიკურში გარდაქმნა (ნახ. 5.9).

ვირტუალური მეხსიერების სისტემაში შეიძლება ორი კლასის გამოყოფა: სისტემები ბლოკების ფიქსირებული ზომებით (გვერდებით

ორგანიზაცია) და სისტემები ბლოკების ცვლადი ზომებით (სეგმენტური ორგანიზაცია). ჩვეულებრივ ორივე ვარიანტს ათანხმებენ (სეგმენტურ - გვერდებითი ორგანიზაცია).



ნახ. 5.9. ვირტუალური მისამართის გამოსახვა ფიზიკურში

5.8. გარე მეხსიერება

დამახსოვრების მოწყობილობების იერარქიაში მნიშვნელოვან რგოლს წარმოადგენს გარე, ან მეორადი მეხსიერება, რომელიც რეალიზებულია სხვადასხვა დამახსოვრების მოწყობილობათა საფუძველზე. ასეთი დამახსოვრების მოწყობილობების ყველაზე გავრცელებული ტიპებია მაგნიტური და ოპტიკური დისკები და მაგნიტოლენტური მოწყობილობები.

6. მართვის მოწყობილობები

6.1. ცენტრალური მართვის მოწყობილობების ფუნქციები

გამომთვლელი მანქანის მართვის მოწყობილობა ახორციელებს გამოთვლითი პროცესის მსვლელობის მართვის ფუნქციებს პროგრამის ბრძანებების ავტომატური შესრულების უზრუნველყოფით. გამომთვლელ მანქანაში პროგრამის შესრულების პროცესი წარმოადგენს მანქანური ციკლების თანმიმდევრობას. მოვახდინოთ ტიპიური მანქანური ციკლის მსვლელობის დროს მართვის მოწყობილობის მიერ რეალიზებული ძირითადი ფუნქციების დეტალიზაცია. სიმარტივისათვის მივიღოთ, რომ გამომთვლელი მანქანა უზრუნველყოფს ერთმისამართიან ბრძანებათა სისტემას. ამასთან, კერძოდ ჩავთვალოთ, რომ ორ ოპერანდიანი არითმეტიკული ბრძანების შესრულების დაწყებამდე მეორე ოპერანდი უკვე იმყოფება პროცესორში.

მანქანურ ციკლში პირველ ეტაპს წარმოადგენს მეხსიერებიდან ბ რ ძ ა ნ ე ბ ი ს ა მ ო რ ჩ ე ვ ა.

ბრძანების ამორჩევას მოსდევს მისი ოპერაციული ნაწილის (ოპერაციის კოდის) დეკოდირების ეტაპი. სიმარტივისათვის ჯერ-ჯერობით ეს ეტაპი განვიხილოთ როგორც ბრძანების ამორჩევის ეტაპის შემადგენელი ნაწილი.

მეორე მიზნის ფუნქციაა მ ო მ დ ე ვ ნ ო ბ რ ძ ა ნ ე ბ ი ს მ ი ს ა მ ა რ თ ი ს ფ ო რ მ ი რ ე ბ ა.

შემდეგ მოდის ეტაპი ო პ ე რ ა ნ დ ი ს შ ე მ ს რ უ ლ ე ბ ე ლ ი მ ი ს ა მ ა რ თ ი ს ან გადასვლის მ ი ს ა მ ა რ თ ი ს ფ ო რ მ ი რ ე ბ ა. აქ ფუნქციას იმდენი მოდიფიკაცია აქვს, დამისამართების რამდენი ხერხიცაა გათვალისწინებული გამომთვლელ მანქანაში.

მეოთხე ეტაპზე სრულდება მიზნის ფუნქცია - ო პ ე რ ა ნ დ ი ს ა მ ო რ ჩ ე ვ ა მეხსიერებიდან წინა ეტაპზე ფორმირებული შემსრულებელი მისამართით.

და ბოლოს, მანქანური ციკლის ბოლო ეტაპზე მოქმედებას უთითებს მიზნის ფუნქცია - ო პ ე რ ა ც ი ი ს შ ე ს რ უ ლ ე ბ ა. ცხადია, რომ ამ ფუნქციის მოდიფიკაციათა რაოდენობა ტოლია გამომთვლელი მანქანის ბრძანებათა სისტემაში არსებული ოპერაციების რაოდენობის.

მიზნის ფუნქციათა მიმდევრობის რიგი მთლიანად განსაზღვრავს მართვის მოწყობილობისა და მთლიანად მთელი გამომთვლელი მანქანის მუშაობის დინამიკას.

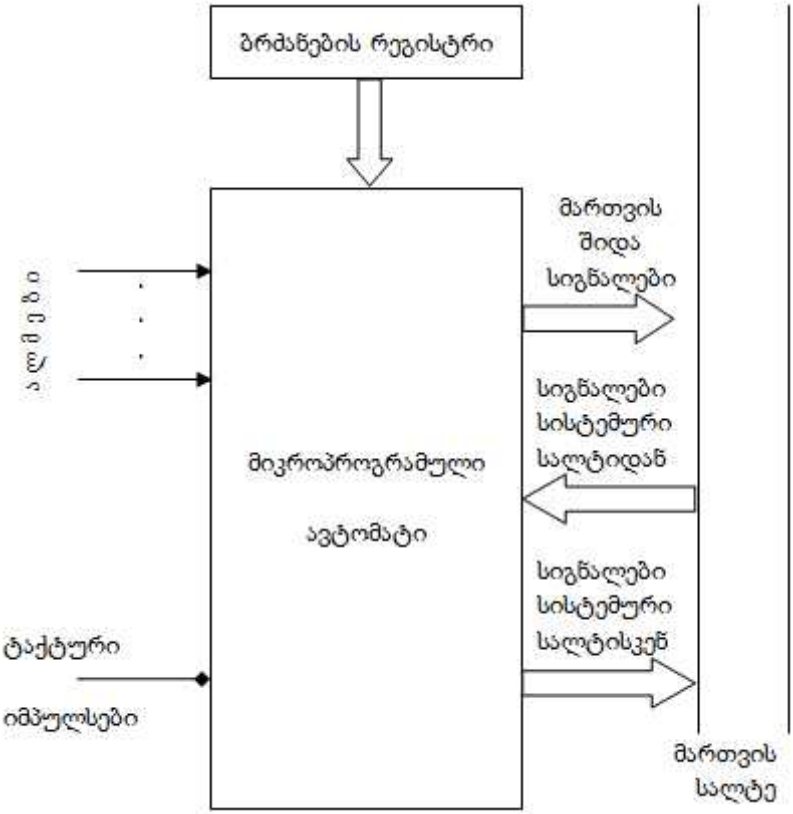
6.2. მართვის მოწყობილობის მოდელი

თავისი ფუნქციების შესასრულებლად მართვის მოწყობილობას უნდა გააჩნდეს თავისი შესასვლელები, რომლებიც საშუალებას მისცემენ მას განსაზღვროს სამართავი სისტემის მდგომარეობა, და გამოსასვლელები, რომლებითაც განხორციელდება სისტემის მოქმედების მართვა. მართვის მოწყობილობის მოდელი მოყვანილია ნახ. 6.1-ზე.

მართვის მოწყობილობისათვის შესასვლელ ინფორმაციას წარმოადგენს:

- ტ ა ქ ტ უ რ ი ი მ პ უ ლ ს ე ბ ი - თითოეულ ტაქტურ იმპულსთან ერთად მართვის მოწყობილობა ახდენს ერთი ან რამდენიმე ოპერაციის ინიცირებას;

- ო პ ე რ ა ც ი ი ს კ ო დ ი - მიმდინარე ბრძანების ოპერაციის კოდი მიეწოდება ბრძანების რეგისტრიდან და გამოიყენება იმის გასარ-



ნახ. 6.1. მართვის მოწყობილობის მოდელი

კვევად, თუ რა მიკრო ოპერაციები უნდა შესრულდნენ მანქანური ციკლის დროს;

- ა ლ მ ე ბ ი - მართვის მოწყობილობას ესაჭიროება ცენტრალური პროცესორის მდგომარეობისა და წინა ოპერაციის შედეგის შესაფასებლად, რაც აუცილებელია პირობითი გადასვლის ბრძანებათა შესასრულებლად;

- ს ი გ ნ ა ლ ე ბ ი ს ი ს ტ ე მ უ რ ი ს ა ლ ტ ი დ ა ნ (სისტემური სალტის სიგნალები) - სისტემური სალტის სიგნალების ნაწილი,

რომლებიც უზრუნველყოფენ მართვის მოწყობილობისათვის წყვეტებზე მოთხოვნების, დადასტურებების და ა. შ. გადაცემას.

- თავის მხრივ მართვის მოწყობილობა, უფრო ზუსტად კი მიკროპროგრამული ავტომატი, ქმნის შემდეგ გამოსასვლელ ინფორმაციას:

- მ ა რ თ ვ ი ს შ ი გ ა ს ი გ ნ ა ლ ე ბ ი - ეს სიგნალები ზემოქმედებენ ცენტრალური პროცესორის შიგა სქემებზე და მიეკუთვნებიან ორიდან ერთ-ერთ ტიპს: იმათ, რომლებიც იწვევენ მონაცემების გადაადგილებას რეგისტრიდან რეგისტრში, და იმათ, რომლებიც ახორციელებენ გამომთვლელი მანქანის ოპერაციული მოწყობილობის განსაზღვრული ფუნქციების ი ნ ი ც ი რ ე ბ ა ს ;

- ს ი გ ნ ა ლ ე ბ ი ს ი ს ტ ე მ უ რ ი ს ა ლ ტ ი ს კ ე ნ - აგრეთვე მიეკუთვნებიან ორიდან ერთ-ერთ ტიპს: მმართველი სიგნალები მეხსიერებისაკენ და მმართველი სიგნალები შეტანა/გამოტანის მოდულისკენ.

6.3. მართვის მოწყობილობის სტრუქტურა

როგორც ზემოთ აღვნიშნეთ, გამომთვლელი მანქანის ფუნქციონირების რეჟიმი შედგება მის კვანძებში ელემენტალურ მოქმედებათა თანმიმდევრობისაგან. ინფორმაციის ასეთ ელემენტალურ გარდაქმნებს, შესრულებულებს სინქრონიზაციის სიგნალთა ერთი ტაქტის განმავლობაში, მ ი კ რ ო ო პ ე რ ა ც ი ე ბ ს უწოდებენ. მართვის სიგნალთა ერთობლიობები, რომლებიც იწვევენ ერთდროულად შესასრულებელ მიკროოპერაციებს, ქმნიან მ ი კ რ ო ბ რ ძ ა ნ ე ბ ა ს . თავის მხრივ მიკრობრძანებათა თანმიმდევრობას, რომელიც განსაზღვრავს

მანქანური ციკლის შემადგენლობას და რეალიზაციის რიგს, მიღებულია ვუწოდოთ მ ი კ რ ო პ რ ო გ რ ა მ ა. მართვის სიგნალები გამომუშავდებიან მართვის მოწყობილობით, უფრო ზუსტად კი მისი ერთ-ერთი კვანძით - მ ი კ რ ო პ რ ო გ რ ა მ უ ლ ი ა ვ ტ ო მ ა ტ ი თ. სახელწოდება ასახავს იმას, რომ მიკროპროგრამული ავტომატი განსაზღვრავს მიკროპროგრამას როგორც მიკროოპერაციათა შესრულების თანმიმდევრობას.

ზემოთ ჩამოთვლილი მიზნის ფუნქციების რეალიზაციის მიკროპროგრამები ინიცირდებიან (იქმნებიან) დ ა ვ ა ლ ე ბ ი ს მ ი მ ც ე მ ი (წარმმართველი) მ ო წ ყ ო ბ ი ლ ო ბ ი თ, რომელიც გამოიმუშავებს მართვის სიგნალების მოთხოვნილ თანმიმდევრობას და შედის მართვის მოწყობილობის მმართველი ნაწილის შემადგენლობაში.

მიკროპროგრამები სრულდებიან შ ე მ ს რ უ ლ ე ბ ე ლ ი მ ო წ ყ ო ბ ი ლ ო ბ ი თ, რომელიც შედის ძირითადი მეხსიერების და ოპერაციული მოწყობილობის შემადგენლობაში. მომდევნო ბრძანების მისამართის ფორმირებისა და ოპერაციათა შესრულების მიზნის ფუნქციებისათვის შემსრულებული მოწყობილობის როლს ასრულებს მართვის მოწყობილობის სამისამართო ნაწილი. მართვის მოწყობილობის განზოგადებულ სტრუქტურაში (ნახ. 6.2) შეიძლება გამოვყოთ ორი ნაწილი: მმართველი და სამისამართო.

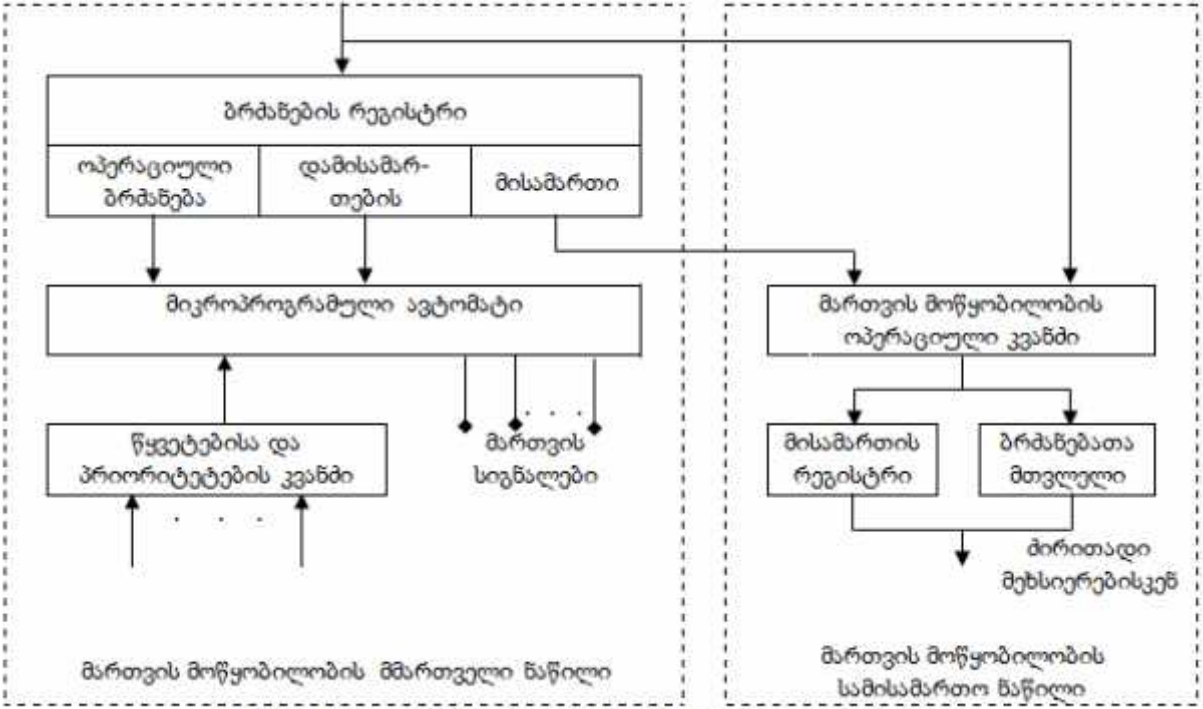
მართვის მოწყობილობის მმართველი ნაწილის დანიშნულებაა გამომთვლელი მანქანის ოპერაციული ბლოკის, მართვის მოწყობილობის სამისამართო ნაწილის, ძირითადი მეხსიერების და გამომთვლელი მანქანის სხვა კვანძების მუშაობის კოორდინაცია.

მართვის მოწყობილობის სამისამართო ნაწილი უზრუნველყოფს ძირითადი მეხსიერების ბრძანების მისამართების და ოპერანდების შემსრულებელი მისამართების ფორმირებას.

მართვის მოწყობილობის მმართველ ნაწილში შედიან:

- ბრძანების რეგისტრი, რომელიც შედგება სამისამართო და ოპერაციული ნაწილებისაგან;

- მიკროპროგრამული ავტომატი;



ნახ. 6.2. მართვის მოწყობილობის განზოგადებული სტრუქტურა.

- წყვეტებისა და პრიორიტეტების კვანძი.

ბრძანების რეგისტრი დანიშნულებაა მეხსიერების მოწყობილობიდან მომდევნო ბრძანების მიღება. მიკროპროგრამული ავტომატი ბრძანების ოპერაციული ნაწილის გაშიფრის შედეგების საფუძველზე გამოიმუშავებს მიკრობრძანებათა განსაზღვრულ თანმიმდევრობას, რომლებიც იწვევენ მართვის მოწყობილობის ყველა მიზნის ფუნქციის შესრულებას.

მიკრობრძანებების ფორმირების ხერხისგან დამოკიდებულებით გვაქვს შემდეგი მიკროპროგრამული ავტომატები:

- ხისტი ანუ აპარატურული ლოგიკით;

- პროგრამირებადი ლოგიკით.

წყვეტებისა და პრიორიტეტების კვანძის საშუალებით ხორციელდება რეაგირება სხვადასხვა სიტუაციებზე, რომლებიც დაკავშირებულნი არიან როგორც მუშა პროგრამების შესრულებასთან, ისე გამომთვლელი მანქანის მდგომარეობასთან.

მართვის მოწყობილობის სამისამართო ნაწილი შედგება:

- მართვის მოწყობილობის ოპერაციული კვანძისაგან;

- მისამართის რეგისტრისაგან;

- ბრძანებების მთვლელისაგან.

მისამართის რეგისტრი გამოიყენება ოპერანდების შემსრულებელი მისამართების შენახვისათვის, ხოლო ბრძანებათა მთვლელი - ბრძანების მისამართების გამომუშავებისა და შენახვისათვის. მისამართის რეგისტრისა და ბრძანებათა მთვლელის შემცველობა იგზავნება ძირითადი მეხსიერების მისამართის რეგისტრში შესაბამისად ოპერანდებისა და ბრძანებების გამოსამუშავებლად.

მართვის მოწყობილობის ოპერაციული კვანძი, რომელსაც სხვაგვარად ინდექსური არითმეტიკის კვანძს ან სამისამართო კვანძს ეძახიან, ამუშავებს ბრძანებების სამისამართო ნაწილებს, აფორმირებს ოპერანდების შემსრულებელ მისამართებს, აგრეთვე ამზადებს მომდევნო ბრძანების მისამართს გადასვლის ბრძანების შესრულების დროს. მართვის მოწყობილობის ოპერაციული კვანძის შემად-

გენლობა შეიძლება ანალოგიური იყოს გამომთვლელი მანქანის ძირითადი ოპერაციული მოწყობილობის შემადგენლობის (ზოგჯერ უმარტივეს გამომთვლელ მანქანებში მოწყობილობებზე დანახარჯების ეკონომიის მიზნით მართვის მოწყობილობის ოპერაციული კვანძი შეთავსებულია ძირითად ოპერაციულ მოწყობილობასთან.)

მართვის მოწყობილობაში შეიძლება აგრეთვე შედიოდნენ დამატებითი კვანძები, კერძოდ მეხსიერებასთან პირდაპირი შეღწევის (მიღწევის) ორგანიზაციის კვანძი. ჩვეულებრივ ეს კვანძი მზადდება დამოუკიდებელი მოწყობილობის - მეხსიერებასთან პირდაპირი შეღწევის კონტროლერის სახით. ის უზრუნველყოფს ოპერაციული მოწყობილობის მუშაობის დროში შეთავსებას ძირითად მეხსიერებასა და გამომთვლელი მანქანის სხვა მოწყობილობებს შორის ინფორმაციის გაცვლის პროცესთან, რითიც ზრდის მანქანის საერთო წარმადობას.

საკმაოდ ხშირად მართვის მოწყობილობის სხვადასხვა კვანძების რეგისტრებს აერთიანებენ მართვის მოწყობილობის მმართველ (სპეციალურ) რეგისტრთა დამოუკიდებელ კვანძად.

7. გამომთვლელი მანქანების ოპერაციული მოწყობილობები

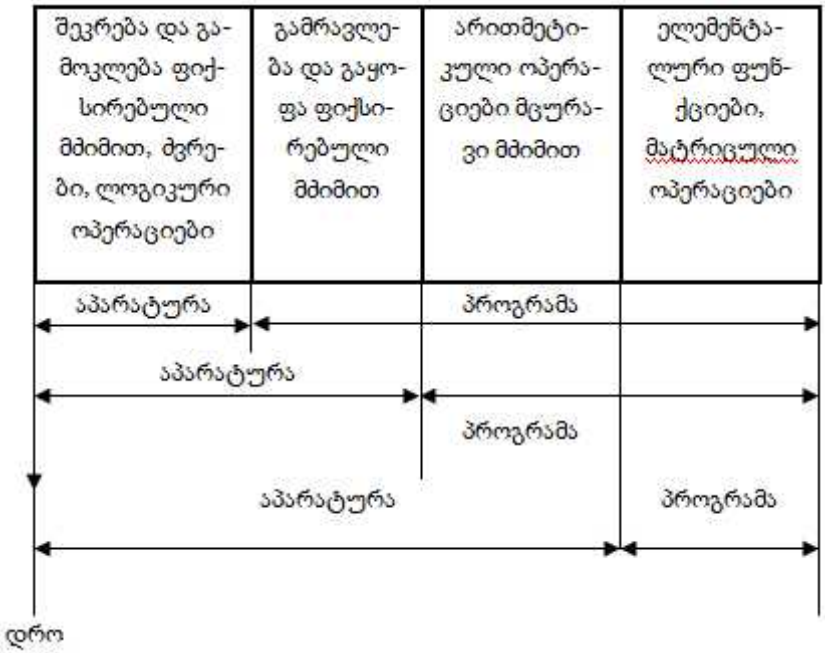
კლასიკურ ფონ-ნეიმანისეულ გამომთვლელ მანქანაში მონაცემთა არითმეტიკული და ლოგიკური დამუშავების ფუნქცია დაკისრებული აქვს არითმეტიკულ-ლოგიკურ მოწყობილობას. შესასრულებელი ოპერაციებისა და დასამუშავებელი მონაცემების ტიპების მრავალფეროვნების გათვალისწინებით შეგვიძლია ვისაუბროთ არა ერთიან მოწყობილობაზე, არამედ სპეციალიზირებულ ოპერაციულ მოწყობილობათა კომპლექსზე, რომელთაგან თითოეული ასრულებს არითმეტიკულ და ლოგიკურ ოპერაციათა განსაზღვრულ ქვესიმრავლეს, გათვალისწინებულს გამომთვლელი მანქანის ბრძანებათა სისტემით. ამ პოზიციებიდან საჭიროა გამოვყოთ:

- მთელრიცხვა არითმეტიკის ოპერაციული მოწყობილობები;
- ოპერაციული მოწყობილობები ლოგიკური ოპერაციების შესასრულებლად;
- ათობითი არითმეტიკის ოპერაციული მოწყობილობები;
- ოპერაციული მოწყობილობები რიცხვებისათვის მცურავი მძიმით.

პრაქტიკაში პირველი ორი ჯგუფი ჩვეულებრივ ერთიანდება ერთი ოპერაციული მოწყობილობის ჩარჩოებში. ათობითი არითმეტიკის სპეციალიზებული ოპერაციული მოწყობილობები თანამედროვე გამომთვლელ მანქანებში საკმაოდ იშვიათად გვხვდება, ვინაიდან ორობით-ათობით ფორმაში წარმოდგენილი რიცხვების დამუშავება საკმაოდ ეფექტურად შეიძლება მთელრიცხვა ორობითი არითმეტიკის საფუძველზე. ამრიგად, ითვლება, რომ არითმეტიკულ-ლოგიკური მოწყობილობა ქმნის ორი სახის ოპერაციულ მოწყობილობებს - მთელ-

რიცხვა ოპერაციული მოწყობილობებს და ოპერაციული მოწყობილობებს რიცხვების დასამუშავებლად მცურავი მძიმის ფორმატში.

მინიმალურ ვარიანტში არითმეტიკულ-ლოგიკური მოწყობილობა უნდა შეიცავდეს აპარატურას, რომლის საშუალებითაც განხორციელდება მხოლოდ ძირითადი ლოგიკური ოპერაციები, ძვრები, აგრეთვე ფიქსირებული მძიმის ფორმაში რიცხვების შეკრება და გამოკლება. ამ ნაკრებზე დაყრდნობით შეიძლება პროგრამული ხერხით ვუზრუნველყოთ დანარჩენი არითმეტიკული და ლოგიკური ოპერაციების შესრულება როგორც რიცხვებისათვის ფიქსირებული მძიმით, ისე ინფორმაციის წარმოდგენის სხვა ფორმებისათვის. ოღონდ საჭიროა გავითვალისწინოთ, რომ მსგავსი ვარიანტი არ იძლევა გამოთვლების მაღალი



ნახ. 7.1. არითმეტიკულ-ლოგიკური მოწყობილობის აპარატურულ და პროგრამულ რეალიზაციებს შორის ფარდობის ცვლილების დინამიკა

სისწრაფის მიღწევის საშუალებას, ამიტომ ტექნოლოგიური შესაძლებლობების გაფართოებისას არითმეტიკულ-ლოგიკურ მოწყობილობის შემადგენლობაში აპარატურული საშუალებების წილი მუდმივად იზრდება.

ნახ. 7.1-ზე მოტანილია გამოთვლითი ტექნიკის ელემენტური ბაზის განვითარებასთან ერთად არითმეტიკულ-ლოგიკური მოწყობილობის ფუნქციების აპარატურულ და პროგრამულ რეალიზაციებს შორის ფარდობის ცვლილების დინამიკა. აქ იგულისხმება, რომ ვერტიკალურ ღერძზე აითვლება კალენდარული დრო.

7.1. ოპერაციული მოწყობილობების სტრუქტურები

ელემენტთა ნაკრებს, რომლის ბაზაზეც იქმნებიან ოპერაციულ მოწყობილობათა სხვადასხვა სტრუქტურები, ეწოდება **სტრუქტურული ბაზისი**. ოპერაციული მოწყობილობის სტრუქტურული ბაზისი შედგება:

- რეგისტრებისაგან, რომლებიც უზრუნველყოფენ მონაცემთა სიტყვების ხანმოკლე შენახვას;

- მართვადი სალტეებისაგან, რომელთა დანიშნულებაა მონაცემთა სიტყვების გადაცემა;

- კომბინაციური სქემებისაგან, რომლებიც ახორციელებენ მიკროპერაციათა ფუნქციებისა და ლოგიკური პირობების შესრულებას მართვის მოწყობილობის მმართველი სიგნალებით.

შესაძლებელია ოპერაციული მოწყობილობის სინთეზირება ე. წ. კანონიკური სტრუქტურით, რომელიც ფუძემდებლურია სხვა სტრუქტურების სინთეზისათვის. ასეთი სტრუქტურა იქმნება სარეალიზაციო

ფუნქციის ყოველი ელემენტის შეცვლით სტრუქტურული ბაზისის შესაბამისი ელემენტით. კანონიკურ სტრუქტურას სხვა ვარიანტებთან შედარებით გააჩნია მაქსიმალური წარმადობა, ოღონდ მასში მოწყობილობათა სიჭარბეა. პრაქტიკული პოზიციებიდან გამომდინარე დიდ ინტერესს იწვევს ოპერაციული მოწყობილობების სტრუქტურათა სხვა ორი სახე: ხისტი და მაგისტრალური.

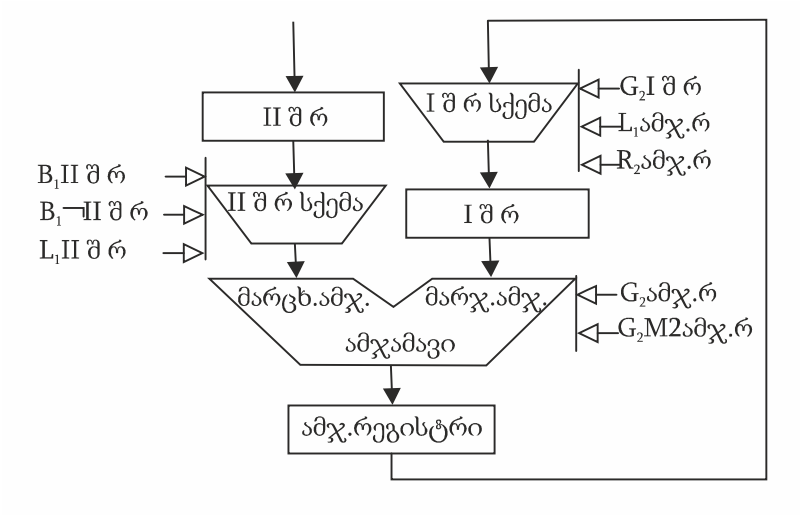
7.1.1. ოპერაციული მოწყობილობები ხისტი სტრუქტურით

ხისტი სტრუქტურის მქონე ოპერაციულ მოწყობილობაში კომბინაციური სქემები ხისტად არიან განაწილებული ყველა რეგისტრს შორის. თითოეულ რეგისტრს მიეკუთვნება კომბინაციური სქემების თავისი ნაკრები, რომელიც საშუალებას იძლევა განხორციელდეს განსაზღვრული მიკრო ოპერაციები. ხისტი სტრუქტურის მქონე ოპერაციული მოწყობილობის მაგალითი, რომელიც ასრულებს „შეკრების“ ტიპის ოპერაციას, მოტანილია ნახ. 7.2-ზე.

ოპერაციული მოწყობილობის შემადგენლობაში შედის სამი რეგისტრი თავისი ლოგიკური სქემებით:

- პირველი შესაკრების რეგისტრი (I შ რ) და სქემა (I შ რ სქემა);
- მეორე შესაკრების რეგისტრი (II შ რ) და სქემა (II შ რ სქემა);
-ამჯამავის რეგისტრი (ამჯ. რ) და კომბინაციური ამჯამავის სქემა (ამჯამავი).

მეორე შესაკრების ლოგიკური სქემა ახორციელებს მეორე შესაკრების გადაცემის მიკროოპერაციებს მეორე შესაკრების რეგისტრიდან ამჯამავის მარცხენა შესასვლელზე:



ნახ. 7.2. ოპერაციული მოწყობილობა ხისტი სტრუქტურით

- პირდაპირი კოდით მარცხ. ამჯ. := II შ რ (მართვის B_1 II შ რ სიგნალით);

- შებრუნებული კოდით მარცხ. ამჯ. := ¬ II შ რ (მართვის B_1 ¬ II შ რ სიგნალით);

- ერთი თანრიგით მარცხნივ ძვრით მარცხ. ამჯ. := L_1(II შ რ • 0) (მართვის L_1 II შ რ სიგნალით).

პირველი შესაკრების ლოგიკური სქემა უზრუნველყოფს შედეგის გადაგზავნას ამჯამავის რეგისტრიდან პირველი შესაკრების რეგისტრში:

- პირდაპირი კოდით I შ რ := მარჯ. ამჯ. (მართვის R_2 ამჯ. რ სიგნალით);

- მარცხნივ ერთი თანრიგით ძვრით I შ რ := L_1 (ამჯ. რ • 0) (მართვის L_1 ამჯ. რ სიგნალით);

- ორი თანრიგით მარჯვნივ ძვრით I შ რ := R_2 (S•S•ამჯ. რ) (მართვის G_2 I შ რ სიგნალით).

კომბინირებული ამჯამავის დანიშნულებაა მის მარცხენა (მარცხ. ამჯ.) და მარჯვენა (მარჯ. ამჯ.) შესასვლელებზე მოსული ოპერანდების აჯამვა (ჩვეულებრივ ან 2-ის მოდულით). აჯამვის შედეგი შედის ამჯამავის რეგისტრში: ამჯ. რ : = მარცხ. ამჯ. + მარჯ. ამჯ. (მართვის G_2 ამჯ. რ სიგნალით) ან ამჯ. რ : = მარცხ. ამჯ.⊕ მარჯ. ამჯ. (მართვის (G_2 M2 ამჯ. რ სიგნალით).

ხისტი სტრუქტურის მქონე ოპერაციული მოწყობილობის მოდელს წარმოადგენს ე. წ. I ავტომატი.

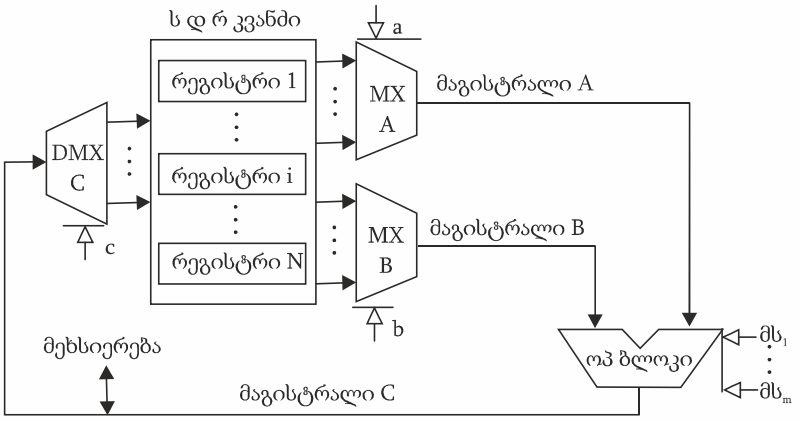
ხისტი სტრუქტურის მქონე ოპერაციული მოწყობილობის უპირატესობას მაღალი სწრაფქმედება წარმოადგენს, ხოლო ნაკლს-სტრუქტურის მცირე რეგულარობა, რაც ართულებს ასეთი ოპერაციული მოწყობილობების დამზადებას დიდი ინტეგრალური სქემების სახით.

7.1.2. ოპერაციული მოწყობილობები მაგისტრალური სტრუქტურით

მაგისტრალურ სტრუქტურიან ოპერაციულ მოწყობილობებში ყველა შიდა რეგისტრი გაერთიანებულია საერთო დანიშნულების რეგისტრთა (ს დ რ) დამოუკიდებელ კვანძში, ხოლო ყველა კომბინაციური სქემა - ოპერაციულ ბლოკში (ოპ ბლოკი), რომელიც ხშირად ასოცირდება ტერმინთან „არითმეტიკულ-ლოგიკური მოწყობილობა“.

ოპერაციული ბლოკი და რეგისტრთა კვანძი ერთმანეთს უკავშირდება მაგისტრალების დახმარებით - სწორედ აქედანაა დასახელება „მაგისტრალური ომ“.

მაგისტრალური ოპერაციული მოწყობილობის მაგალითი მოტანილია ნახ. 7.3-ზე.



ნახ. 7.3. მაგისტრალური ოპერაციული მოწყობილობა

აქ საერთო დანიშნულების რეგისტრთა კვანძის შემადგენლობაში შედის საერთო დანიშნულების N რეგისტრი, შეერთებულები A და B მაგისტრალებთან MX A და MX B მულტიპლექსორებით. თითოეული მულტიპლექსორი წარმოადგენს მართვად კომუტატორს, რომელიც აერთებს ერთ-ერთი საერთო დანიშნულების რეგისტრის გამოსასვლეს შესაბამის მაგისტრალთან. მისაერთებელი რეგისტრის ნომერი განისაზღვრება a -ს და b -ს მისამართით, რომელიც მიეწოდება მულტიპლექსორის სამისამართო შესასვლელებს მართვის მოწყობილობიდან.

A და B მაგისტრალებიდან ოპერანდები ეწოდებიან ოპერაციული ბლოკის შესასვლელებს (მართვის მოწყობილობიდან მოწოდებული მართვის სიგნალით), რომლებსაც უერთდება მოთხოვნილი ოპერაციის განმახორციელებელი კომბინაციური სქემა. ამრიგად, ოპერაციული ბლოკის ნებისმიერი მიკროოპერაცია შეიძლება შესრულდეს ოპერაციული მოწყობილობის ნებისმიერ რეგისტრთა შემცველობაზე. მიკროოპერაციის შედეგი C მაგისტრალით შეიტანება DMX C დემულტიპლექსორის გავლით საერთო დანიშნულების რეგისტრთა კვანძის კონკრე-

ტულ რეგისტრში. დემულტიპლექსორი წარმოადგენს მართვად კომუტატორს, რომელსაც აქვს ერთი ინფორმაციული შესასვლელი და N ინფორმაციული გამოსასვლელი. შესასვლელი უერთდება მოცემული C მისამართის მქონე გამოსასვლელს, რომელიც ეწოდება (მისამართი) DMX C-ს სამისამართო შესასვლელებს მართვის მოწყობილობიდან.

მაგისტრალურ სტრუქტურიანი ოპერაციული მოწყობილობის მოდელს წარმოადგენს M-ავტომატი. M-ავტომატი ეწოდება ოპერაციული მოწყობილობის მოდელს, აგებულს კომბინაციური სქემების გაერთიანების პრინციპის საფუძველზე და რომელიც ყოველ ტაქტზე ახორციელებს მხოლოდ ერთ მიკრო ოპერაციას.

მაგისტრალური ოპერაციული მოწყობილობის ძირითად ღირსებას წარმოადგენს მაღალი უნივერსალურობა და სტრუქტურის რეგულარულობა, რაც ამარტივებს მათ რეალიზაციას ინტეგრალური სქემის კრისტალებზე. საერთოდ, ოპერაციულ მოწყობილობებს მაგისტრალური სტრუქტურით თანამედროვე გამომთვლელ მანქანებში უპირატესი ადგილი უჭირავთ.

7.1.2.1. მაგისტრალურ სტრუქტურიანი ოპერაციული მოწყობილობების კლასიფიკაცია

მაგისტრალური ოპერაციული მოწყობილობების კლასიფიკაციას ახდენენ მაგისტრალთა სახითა და რაოდენობით, საერთო დანიშნულების რეგისტრების კვანძის ორგანიზაციით, ოპერაციული ბლოკის ტიპით.

მაგისტრალური ოპერაციული მოწყობილობები შეიძლება იყვნენ ერთმიმართულებიანი და ორმიმართულებიანი, შესაბამისად მათში ხორციელდება მონაცემთა გადაცემა ერთი ან ორი სხვადასხვა მიმართულებით. მაგისტრალის მუშაობის ტიპიურ რეჟიმს წარმოადგენს დროის დაყოფა, რომლის დროსაც მაგისტრალი გამოიყენება ფუნქციუ-

რად სხვადასხვა ტიპის მონაცემთა გადასაცემად დროის სხვადასხვა მომენტში.

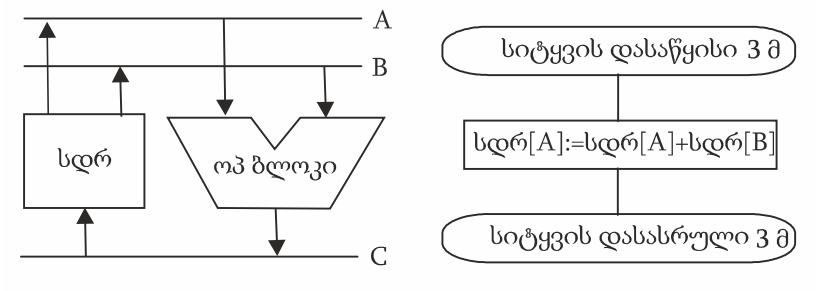
ფუნქციური დანიშნულების მიხედვით გამოყოფენ:

- გარე კავშირთა მაგისტრალებს, რომლებიც ოპერაციულ მოწყობილობებს აკავშირებენ მეხსიერებასთან და გამომთვლელი მანქანის შეტანა/გამოტანის არხებთან;

- ოპერაციულ მოწყობილობებს შიგა მაგისტრალებით, რომლებიც პასუხს აგებენ საერთო დანიშნულების რეგისტრების კვანძსა და ოპერაციულ ბლოკს შორის კავშირზე.

გარე კავშირთა მაგისტრალების რაოდენობა დამოკიდებულია კონკრეტული გამომთვლელი მანქანის არქიტექტურაზე და ჩვეულებრივ გარე კავშირებისათვის არ აღემატება ორს, ხოლო შიგა კავშირებისათვის სამს.

სამმაგისტრალური ოპერაციული მოწყობილობის სტრუქტურა წარმოდგენილია ნახ. 7.4, ა -ზე, ხოლო მისი შესაბამისი „შეკრების“ ტიპის ოპერაციის შესრულების მიკროპროგრამა - ნახ. 7.4, ბ-ზე.



ნახ. 7.4. სამმაგისტრალური ოპერაციული მოწყობილობა: ა - სტრუქტურა; ბ - შეკრების მიკროპროგრამა

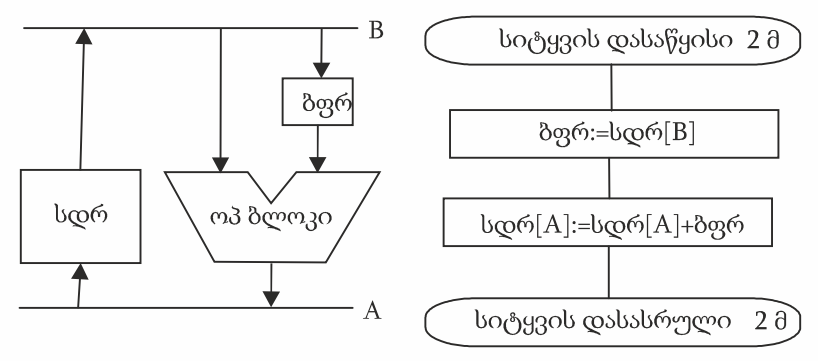
მოცემული ვარიანტი ხასიათდება უდიდესი სწრაფქმედებით: საერთო დანიშნულების რეგისტრიდან ოპერანდის ამორჩევა, აჯამვის მიკროოპერაციის შესრულება და შედეგის საერთო დანიშნულების რეგისტრში ჩაწერა - ყველა ეს მოქმედება სრულდება ერთი ტაქტის გან-

მავლობაში. სამმაგისტრალური ორგანიზაციის ძირითადი ნაკლია დიდი ინტეგრალური სქემის კრისტალზე დაკავებული დიდი ფართობი (კრისტალის ფართობის 0,16-იდან 0,22-მდე ნაწილი).

ორმაგისტრალური ორგანიზაცია მაგისტრალებით დაკავებული ნაკლები ფართობისას (კრისტალის ფართობის 0,06-იდან 0,19-მდე ნაწილი) მოითხოვს, როგორც მინიმუმ, ერთი ბუფერული რეგისტრის (ბფრ) შემოტანას, რომლის დანიშნულებაა ერთ-ერთი ოპერანდის დროებითი შენახვა (ნახ. 7.5, ა), ამასთან შეკრების ოპერაცია უკვე ორ ტაქტში სრულდება (ნახ. 7.5, ბ):

- I ტაქტი: ერთ-ერთი ოპერანდის ბუფერულ რეგისტრში ჩატვირთვა;

-II ტაქტი: ოპერაციულ ბლოკში მიკროოპერაციის შესრულება ბუფერული რეგისტრისა და ერთ-ერთი საერთო დანიშნულების რეგისტრის შემცველობებზე; შედეგის საერთო დანიშნულების რეგისტრში ჩაწერა.



ნახ. 7.5. ორმაგისტრალური ოპერაციული მოწყობილობა: ა - სტრუქტურა; ბ - შეკრების მიკროპროგრამა

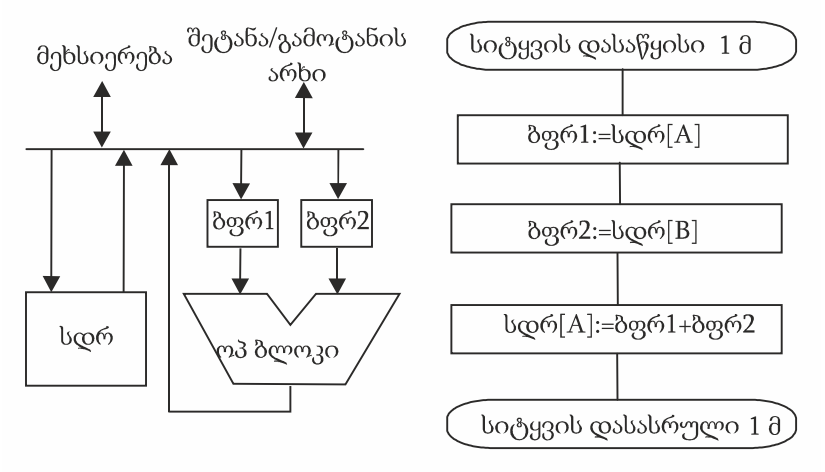
და ბოლოს, ოპერაციული მოწყობილობის ორგანიზაცია მხოლოდ ერთი მაგისტრალის საფუძველზე (ნახ. 7.6, ა) ახდენს ფართობის დანახარჯების მინიმიზაციას (კრისტალის ფართობის 0.03-იდან 0,09-

მდე ნაწილი). ამასთან, ერთამაგისტრალურ ოპერაციულ ბლოკში ჩნდება არანაკლებ ორი ბუფერული რეგისტრის ბფრ1 და ბფრ2-ის შეტანის აუცილებლობა, ხოლო ოპერაციის ხანგრძლივობა იზრდება სამ ტაქტამდე (ნახ. 7.6, ბ):

-I ტაქტი : ერთ-ერთი ოპერანდის ბფრ1-ში ჩატვირთვა;

-II ტაქტი : მეორე ოპერანდის ბფრ2-ში ჩატვირთვა;

-III ტაქტი : ოპერაციულ ბლოკში მიკროოპერაციის შესრულება ბფრ1-სა და ბფრ2 -ის შემცველობაზე; შედეგის ერთ-ერთ საერთო დანიშნულების რეგისტრში ჩაწერა.



ნახ. 7.6. ერთმაგისტრალური ოპერაციული მოწყობილობა: ა - სტრუქტურა; ბ - შეკრების მიკროპროგრამა

7.1.2.2. მაგისტრალური ოპერაციული მოწყობილობის საერთო დანიშნულების რეგისტრის კვანძის ორგანიზაცია

მაგისტრალური ოპერაციული მოწყობილობის საერთო დანიშნულების რეგისტრის კვანძში რეგისტრების რაოდენობა ჩვეულებრივ

აღემატება იმ მინიმუმს, რომელიც აუცილებელია ოპერაციათა უნივერსალური სისტემის რეალიზაციისათვის. რეგისტრების სიჭარბე გამოიყენება:

- მისამართის შემადგენელი ნაწილების შესანახად (ინდექსის, ბაზის);

- ბუფერულ, ზეოპერატიულ მეხსიერებად (გამომთვლელი მანქანის წარმადობის გაზრდისათვის ძირითად მეხსიერებასა და ოპერაციულ მოწყობილობას შორის საჭირო გადაგზავნების შემცირების ხარჯზე).

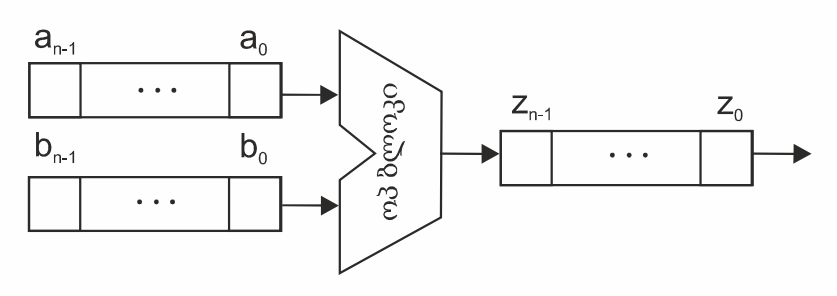
საშუალოდ რეგისტრების რაოდენობა მერყეობს 8-დან 16-მდე, ზოგჯერ შეიძლება მიაღწიოს 32- 64-ს. პროცესორებში ბრძანებათა შემცირებული ნაკრებით საერთო დანიშნულების რეგისტრების რაოდენობა ასეულს აღწევს.

საერთო დანიშნულების რეგისტრის კვანძის ორგანიზაციამ შეიძლება უზრუნველყოს ერთარხიანი ან ორარხიანი შეღწევა, როგორც შესასვლელით (ჩაწერა), ისე გამოსასვლელით (წაკითხვა). პირველ შემთხვევაში კვანძის შესასვლელს უერთდება ერთი დემულტიპლექსორი, ხოლო გამოსასვლელს - ერთი მულტიპლექსორი. მეორე შემთხვევაში შეღწევა ხორციელდება ორი დემულტიპლექსორის და (ან) ორი მულტიპლექსორის საშუალებით. ორარხიანი შეღწევა ზრდის ოპერაციული მოწყობილობის სწრაფქმედებას, ვინაიდან საშუალებას იძლევა განხორციელდეს პარალელური მიმართვა ორ რეგისტრზე.

7.1.2.3. მაგისტრალური ოპერაციული მოწყობილობის ოპერაციული ბლოკის ორგანიზაცია

ოპერაციული ბლოკის ტიპი განისაზღვრება მონაცემთა დამუშავების ხერხით. ანსხვავებენ მიმდევრობითი და პარალელური ტიპის ოპერაციულ ბლოკებს.

მიმდევრობით ოპერაციულ ბლოკში (ნახ. 7.7) ოპერაციები სრულდებიან ბიტებად, თანრიგ-თანრიგ.

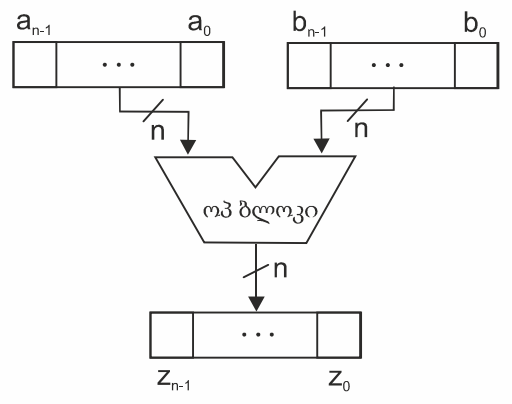


ნახ. 7.7. მიმდევრობითი ოპერაციული ბლოკი

ოპერანდების i-ური თანრიგის დამუშავებისას გაჩენილი გადატანის ბიტი მიეწოდება ოპერაციული ბლოკის შესასვლელს და გაითვალისწინება ოპერანდების (i +1)-იური თანრიგის დამუშავებისას. შედეგი ბიტების მიხედვით შედის გამოსასვლელ რეგისტრში, რომლის წინა შემცველობაც მანამდე იძვრის ერთი თანრიგით. ამრიგად, n - ციკლის შემდეგ გამოსასვლელ რეგისტრში ფორმირდება შედეგის სიტყვა, სადაც თითოეულ თანრიგს უჭირავს მისთვის გათვალისწინებული პოზიცია.

ოპერაციული ბლოკის პარალელური ორგანიზაციის დროს (ნახ. 7.8) ოპერანდების ყველა თანრიგი ერთდროულად მუშავდება. შიდა გადატანები უზრუნველყოფილია ოპერაციული ბლოკის სქემით.

„გრძელი“ სიტყვის ჩარჩოებში გადატანათა ეფექტური სისტემის რეალიზაცია დაკავშირებულია განსაზღვრულ აპარატურულ დანახარჯებთან, ამიტომ პრაქტიკაში ხშირად იყენებენ ოპერაციული ბლოკის პარალელურ - მიმდევრობით სქემას. მასში სიტყვა იშლება 2, 4, და 8 თანრიგიან ჯგუფებად. ჯგუფის შიგნით ყველა თანრიგის დამუშავება პარალელურად ხდება, ხოლო თავად ჯგუფები მიმდევრობით მუშავდებიან.



ნახ. 7.8. პარალელური ოპერაციული ბლოკი

ოპერაციული ბლოკის განზოგადებული სქემა მოტანილია ნახ. 7.9-ზე. მასში შედის მიკრობრძანების დეკოდერი, კოდების მაფორმირებლები, მრავალფუნქციური ამჯამავი, ძვრის რეგისტრი და ლოგიკური პირობების მაფორმირებელი.

მიკრობრძანების დეკოდერი გამოიმუშავებს მართვის შიგა სიგნალებს ოპერაციული ბლოკის ელემენტებისათვის. ის შეტანილია სქემაში მართვის მოწყობილობიდან სიგნალების გადაცემისათვის საჭიროა კავშირების რაოდენობის მინიმიზაციის მიზნით.

კოდების მაფორმირებლები (კოდ.მაფ.1 და კოდ.მაფ.2) ემსახურებიან A და B მაგისტრალებით მიწოდებული ოპერანდების პირდაპირი და შებრუნებული კოდების ფორმირებას. ისინი ახორციელებენ მიკროპერაციათა შემდეგ ნაკრებს:

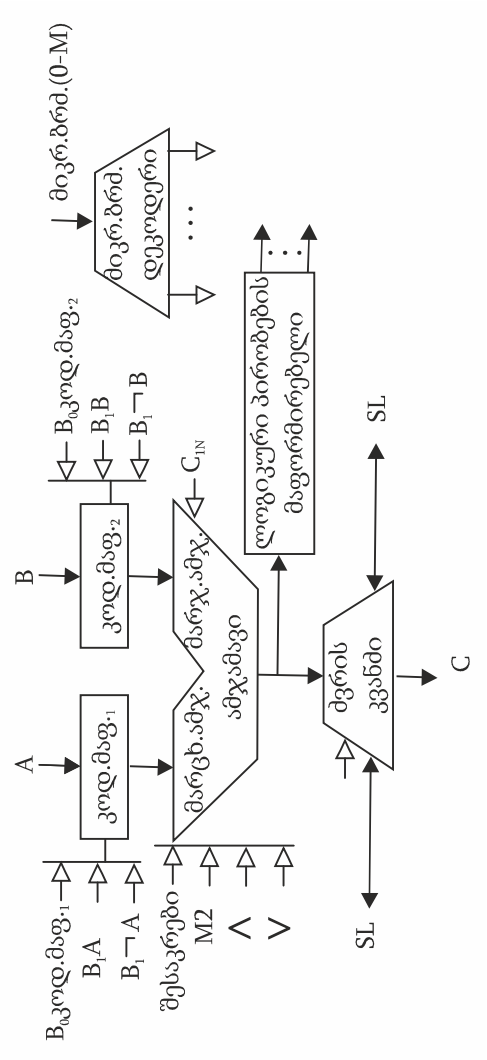
B_0 კოდ.მაფ.1: მარცხ.ამჯ. :=0; B_0 კოდ.მაფ.2: მარჯ.ამჯ. :=0;

B_1 A: მარცხ.ამჯ. > A; B_1B: მარჯ.ამჯ. := B;

B_1 ¬A: მარცხ.ამჯ. := ¬A; B_1¬B: მარჯ.ამჯ. := ¬B.

მრავალფუნქციური ამჯამავი ასრულებს არითმეტიკული შეკრების (გადატანის გათვალისწინებით C_{IN}), ორის მოდულით შეკრების,

მარცხენა და მარჯვენა შესასვლელზე კოდების ლოგიკური შეკრებისა და ლოგიკური გამრავლების მიკროოპერაციებს:



ნახ. 7.9. ოპერაციული ბლოკის განზოგადებული სქემა

შეკრ.: ამჯ. := მარცხ.ამჯ. + მარჯ.ამჯ. + C_{1N};

M2: ამჯ. : = მარცხ.ამჯ. $\oplus$ მარჯ.ამჯ.;

$\wedge$: ამჯ. := მარცხ.ამჯ. $\wedge$ მარჯ.ამჯ.;

$\vee$: ამჯ. := მარცხ.ამჯ. $\vee$ მარჯ.ამჯ. .

ლოგიკური პირობების მაფორმირებელი ამჯამავის გამოსასვლელზე კოდის ანალიზის საფუძველზე გამოიმუშავებს გასაცნობი სიგნალების მნიშვნელობებს, რომლებიც გადაეცემიან მანქანის მართვის მოწყობილობას. შეიძლება იყოს შემდეგი გასაცნობი სიგნალები: ნიშნის ნიშანი S, გადავსების ნიშანი V, შედეგის ნულოვანი მნიშვნელობის ნიშანი Z და ა.შ.

დამძრავი ემსახურება ამჯამავის გამოსასვლელზე კოდის ძვრის მიკროოპერაციის შესრულებას:

T_1: C := ამჯ.;

R_1: C := R1(SL • ამჯ.); SR : =ამჯ.(n);

L_1: C := L1(ამჯ. • SR); SL : =ამჯ.(0).

T_1 მიკროპერაცია უზრუნველყოფს შედეგის გადაცემას C მაგისტრალზე ძვრის გარეშე. R_1 მიკროოპერაციის მსვლელობისას შედეგი იძვრის ერთი თანრიგით მარჯვნივ, ამასთან გამოთავისუფლებულ უფროს თანრიგში შეიტანება მნიშვნელობა გარე SL კონტაქტიდან, ხოლო ამჯამავის „გამოჩოჩებული“ (უმცროსი) თანრიგი იგზავნება გარე SR კონტაქტზე.

L_1 მიკროოპერაციაში შედეგი იძვრის ერთი თანრიგით მაცხნივ. აქ გამოთავისუფლებულ მარცხენა თანრიგში ხდება გარე SR კონტაქტის მნიშვნელობის შეტანა, ხოლო ამჯამავის „გამოჩოჩებული“ (უფროსი) თანრიგი გადაეცემა გარე SL კონტაქტზე.

8. შეტანა / გამოტანის სისტემები

ცენტრალური პროცესორისა და მეხსიერების გარდა, გამომთვლელი მანქანების არქიტექტურის მესამე ძირითად ელემენტს შეტანა /გამოტანის სისტემა (შ/გ ს) წარმოადგენს. შეტანა/გამოტანის სისტემის დანიშნულებაა ინფორმაციის გაცვლა გამომთვლელი მანქანის ბირთვსა და სხვადასხვა გარე მოწყობილობებს შორის. შეტანა/გამოტანის სისტემის ტექნიკური და პროგრამული საშუალებები პასუხს აგებენ გამომთვლელი მანქანის ბირთვსა და გარე მოწყობილობების ფიზიკურ და ლოგიკურ დაკავშირებაზე.

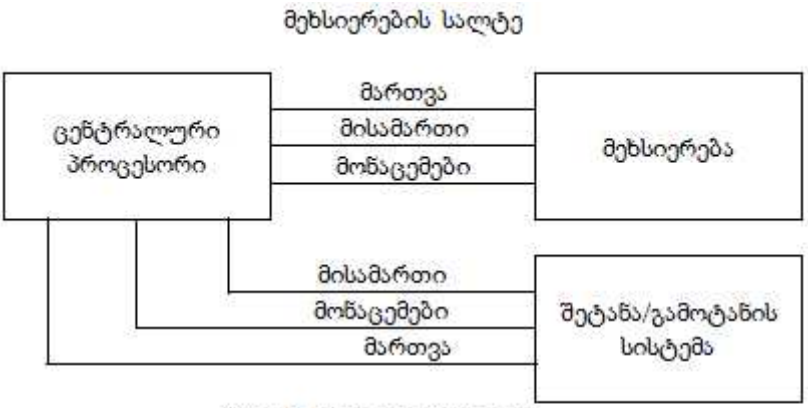
გამომთვლელი მანქანების ევოლუციის პროცესში შეტანა /გამოტანის სისტემებს არქიტექტურის სხვა ელემენტებთან შედარებით ნაკლები ყურადღება ექცეოდა. ამის ირიბი დადასტურებაა ის, რომ წარმადობის კონტროლის ბევრი პროგრამა (ბენჩმარკები) საერთოდ არ ითვალისწინებენ შეტანა/გამოტანის ოპერაციათა გავლენას გამომთვლელი მანქანის ეფექტურობაზე. მსგავსი დამოკიდებულების შედეგად მნიშვნელოვანი გარღვევა მოხდა პროცესორისა და მეხსიერების წარმადობისა, ერთის მხრივ და შეტანა/გამოტანის სიჩქარეს შორის - მეორეს მხრივ.

ტექნიკურად შეტანა/გამოტანის სისტემა გამომთვლელი მანქანის ჩარჩოებში რეალიზებულია შეტანა/გამოტანის მოდულების კომპლექსით. შეტანა/გამოტანის მოდული ახორციელებს გარე მოწყობილობათა დაკავშირებას გამომთვლელი მანქანის ბირთვს შორის და სხვადასხვა კომუნიკაციურ ოპერაციებს მათ შორის. შეტანა/გამოტანის მოდულს შემდეგი ორი ძირითადი ფუნქცია გააჩნია:

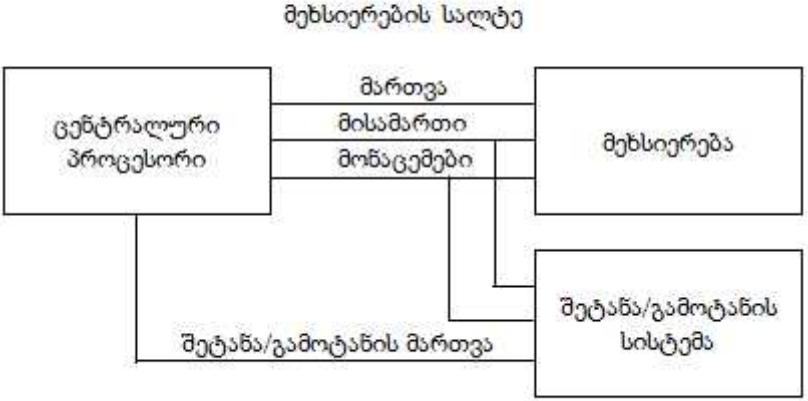
- ცენტრალური პროცესორისა და მეხსიერების ინტერფეისის უზრუნველყოფა („დიდი“ ინტერფეისი);

- ინტერფეისის უზრუნველყოფა ერთ ან რამდენიმე პერიფერიულ მოწყობილობასთან („მცირე“ ინტერფეისი).

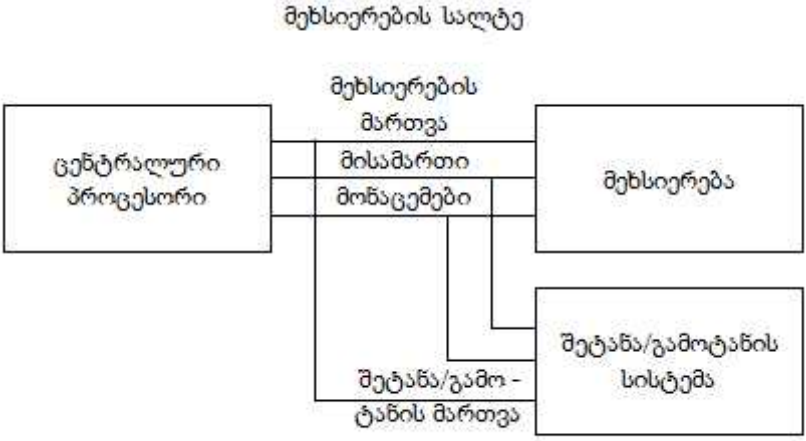
ცნობილი გამომთვლელი მანქანების არქიტექტურების ანალიზის შედეგად შეიძლება გამოვყოთ შეტანა /გამოტანის სისტემის პროცესორთან მიერთების სამი ძირითადი ხერხი (ნახ. 8.1).



ა



ბ



ნახ. 8.1. გამომთვლელი მანქანის არქიტექტურაში შეტანა/გამოტანის სისტემის ადგილი: ა - მეხსიერებისა და შეტანა/გამოტანის განცალკევებული სალტეებით; ბ - მონაცემებისა და ხაზების ერთობლივი გამოყენებით; გ - პროცესორთან და მეხსიერებასთან საერთო უფლებებით მიერთებით.

მეხსიერებისა და შეტანა/გამოტანის განცალკევებულ სალტეებიან ვარიანტში (ნახ. 8.1,ა) ინფორმაციის გაცვლა ცენტრალურ პროცესორსა და მეხსიერებას შორის ფიზიკურად გამოყოფილია სრულებით დამოუკიდებელი სალტეებით. ეს საშუალებას იძლევა განხორციელდეს მეხსიერებაზე მიმართვა შეტანა/გამოტანის შესრულების ერთდროულად. გარდა ამისა, გამომთვლელი მანქანის მოცემული არქიტექტურული ვარიანტი საშუალებას იძლევა მოვახდინოთ თითოეული სალტის სპეციალიზაცია, გავითვალისწინოთ გადასაგზავნი მონაცემების ფორმატი, გაცვლის სინქრონიზაციის თავისებურებები და ა. შ. კერძოდ, შეტანა/გამოტანის სალტეს, რეალური გარე მოწყობილობებს მახასიათებლების გათვალისწინებით, შეიძლება ნაკლები გამტარუნარიანობა

გააჩნდეს, რაც საშუალებას იძლევა შემცირდეს დანახარჯები მის რეალიზაციაზე. ამ გადაწყვეტილების ნაკლად შეიძლება ჩაითვალოს ცენტრალურ პროცესორთან მიერთების წერტილების დიდი რაოდენობა.

მეორე ვარიანტია მონაცემებისა და მისამართების ხაზების ერთობლივი გამოყენება (ნახ. 8.1, ბ) . მეხსიერებასა და შეტანა /გამოტანის სისტემას აქვს მართვის საერთო მისამართის ხაზები და მონაცემების ხაზები, განცალკევებულები დროში. იმავე დროს მეხსიერებასა და შეტანა /გამოტანის სისტემის მართვა, აგრეთვე მათი პროცესორთან ურთიერთქმედების სინქრონიზაცია ხორციელდება დამოუკიდებლად მართვის განცალკევებული ხაზებით. ეს საშუალებას იძლევა გავითვალისწინოთ მეხსიერებასთან და შეტანა/გამოტანის მოდულებთან მიმართვის პროცედურების თავისებურებები და მივაღწიოთ მეხსიერების უჯრედებთან და გარე მოწყობილობებთან მიღწევის უდიდეს ეფექტურობას.

გამომთვლელი მანქანის არქიტექტურის ბოლო ტიპი ითვალისწინებს შეტანა /გამოტანის სისტემის სისტემურ სალტეზე მიერთებას პროცესორთან და მეხსიერებასთან საერთო უფლებებით (ნახ. 8.1, გ). ასეთი მიდგომის უპირატესობები და ნაკლოვანებები განხილული იყო სალტეების ორგანიზაციის საკითხთა განხილვის დროს (თავი 4). პოტენციურად შესაძლებელია აგრეთვე გარე მოწყობილობების სისტემურ სალტეზე პირდაპირი მიერთების ვარიანტი, შეტანა/გამოტანის მოდულების გამოყენების გარეშე, მაგრამ მის საწინააღმდეგოდ შეგვიძლია წამოვაყენოთ რამდენიმე არგუმენტი. უპირველეს ყოვლისა, ამ შემთხვევაში საჭირო შეიქმნებოდა ცენტრალური პროცესორი აღგვეჭურვა ნებისმიერი გარე მოწყობილობის სამართავი უნივერსალური სქემებით. გარე მოწყობილობათა დიდი მრავალფეროვნების გამო, რომლებსაც ამავე დროს გააჩნიათ განსხვავებული მოქმედების პრინციპები, ასეთი სქემები აღმოჩნდებიან ძალიან რთული და ჭარბი ელემენტებით გადატვირთულები. მეორე, შეტანისა და გამოტანისას მონაცემების გადაგზავნა ხორციელდება მნიშვნელოვნად უფრო ნელა, ვიდრე ცენტრალურ პროცესორსა და მეხსიერებას შორის გაცვლისას, და არახელსაყრელი

იქნებოდა გარე მოწყობილობასთან ინფორმაციის გაცვლისათვის გამოგვეყენებინა მაღალი სიჩქარის მქონე სისტემური სალტე. და ბოლოს, გარე მოწყობილობებში ხშირად გამოიყენებიან მათთან მიერთებული გამომთვლელი მანქანისაგან განსხვავებული მონაცემთა ფორმატები და სიტყვათა სიგრძეები.

8.1. გარე მოწყობილობები

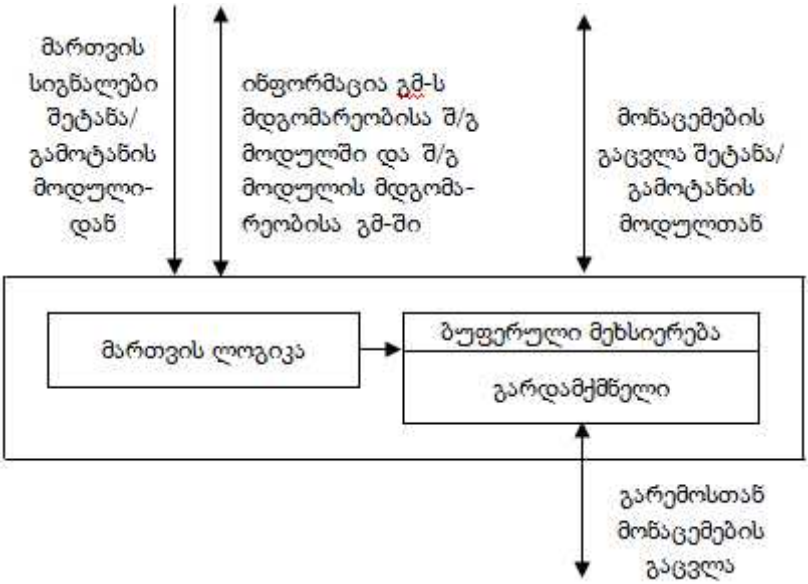
გამომთვლელი მანქანის გარე სამყაროსთან კავშირი ხორციელდება სრულიად განსხვავებული გარე მოწყობილობების დახმარებით. თითოეული გარე მოწყობილობა უერთდება შეტანა/გამოტანის მოდულს ინდივიდუალური სალტით. ინტერფეისს, რომლითაც ხორციელდება შეტანა/გამოტანის მოდულის და გარე მოწყობილობების ასეთი ურთიერთქმედება, ხშირად ეძახიან მ ც ი რ ე ს. ინდივიდუალური სალტე უზრუნველყოფს მონაცემებისა და მართვის სიგნალების, აგრეთვე გაცვლაში მონაწილეთა მდგომარეობის შესახებ ინფორმაციის გაცვლას. შეტანა/გამოტანის მოდულზე მიერთებულ გარე მოწყობილობას, ჩვეულებრივ პ ე რ ი ფ ე რ ი უ ლ მ ო წ ყ ო ბ ი ლ ო ბ ა ს უწოდებენ. პერიფერიულ მოწყობილობათა მთელი სიმრავლე სამ კატეგორიად შეიძლება დავყოთ:

- მომხმარებელთან ურთიერთობისათვის;
- გამომთვლელ მანქანასთან ურთიერთობისათვის;
- შორს მყოფ მოწყობილობასთან კავშირისათვის.

პირველი ჯგუფის მაგალითებია ვიდეო ტერმინალები და პრინტერები. მეორე ჯგუფს მიეკუთვნებიან გარე მეხსიერების მოწყობილო-

ბები (მაგნიტური და ოპტიკური დისკები, მაგნიტური ლენტები და ა. შ.), გადამწოდები და შემსრულებელი მექანიზმები. აღვნიშნოთ გარე მეხსიერების გაორებული როლი, რომელიც ერთის მხრივ, წარმოადგენს გამომთვლელი მანქანის ნაწილს, მეორეს მხრივ - გარე მოწყობილობაა. და ბოლოს, მესამე კატეგორიის მოწყობილობები საშუალებას აძლევენ გამომთვლელ მანქანას გაცვალოს ინფორმაცია შორს მყოფ ობიექტებთან, რომლებიც შეიძლება მიეკუთვნებოდნენ ორ პირველ ჯგუფს. შორს მყოფი ობიექტების როლში შეიძლება იყვნენ აგრეთვე სხვა გამომთვლელი მანქანები.

გარე მოწყობილობების განზოგადებული სტრუქტურა მოტანილია ნახ. 8.2-ზე.



ნახ. 8.2. გარე მოწყობილობის სტრუქტურა

შეტანა/გამოტანის მოდულით ინტერფეისი რეალიზდება მართვის, მდგომარეობისა და მონაცემების სიგნალების სახით. მ ო ნ ა ც ე მ ე ბ ი წარმოდგენილებია ბიტების ერთობლიობით, რომლებიც უნდა გადაეცნენ შეტანა/გამოტანის მოდულში ან მისგან უნდა იქნან მიღებულნი. მ ა რ თ ვ ი ს ს ი გ ნ ა ლ ე ბ ი განსაზღვრავენ გარე მოწყობილობის მიერ შესრულებულ ფუნქციას. ეს შეიძლება იყოს ყველა მოწყობილობისათვის სტანდარტული ფუნქცია - მონაცემთა გადაგზავნა შეტანა/გამოტანის მოდულში ან მისგან მონაცემების მიღება, ანდა მოცემული ტიპის გარე მოწყობილობისათვის სპეციფიკური ფუნქცია; ისეთი, მაგალითად, როგორიცაა მაგნიტური დისკის თავაკის პოზიციონირება ან მაგნიტური ლენტის გადახვევა. მდგომარეობის სიგნალები ახასიათებენ მოწყობილობის მიმდინარე მდგომარეობას, კერძოდ ჩართულია თუ არა გარე მოწყობილობა და მზადაა თუ არა ის მონაცემთა გადასაცემად.

მართვის ლოგიკა - ეს სქემებია, რომლებიც ახორციელებენ გარე მოწყობილობის მუშაობის კოორდინაციას მონაცემთა გადაცემის მიმართულების შესაბამისად. გარდამქმნელის ამოცანას წარმოადგენს სრულებით განსხვავებული ფიზიკური ბუნების ინფორმაციული სიგნალების ტრანსფორმაცია ელექტრულ სიგნალებში და აგრეთვე უკუ გარდაქმნა. ჩვეულებრივ გარდამქმნელთან ერთად გამოიყენება ბუფერული მეხსიერება, რომელიც უზრუნველყოფს შეტანა/გამოტანის მოდულსა და გარე მოწყობილობას შორის გადასაგზავნი მონაცემების დროებით შენახვას.

8.2. შეტანა/გამოტანის მოდულები

8.2.1. მოდულის ფუნქციები

გამომთვლელი მანქანის შემადგენლობაში შეტანა/გამოტანის მოდული პასუხს აგებს ერთი ან რამდენიმე გარე მოწყობილობის მართვაზე და მონაცემების გაცვლაზე ერთის მხრივ ამ მოწყობილობებსა და, მეორეს მხრივ, ძირითადი მეხსიერებისა და ცენტრალური პროცესორის რეგისტრებს შორის. შეტანა/გამოტანის მოდულის ძირითადი ფუნქციები შემდეგნაირად შეიძლება ჩამოვაყალიბოთ:

- მონაცემთა ლოკალიზაცია;
- მართვა და სინქრონიზაცია;
-ინფორმაციის გაცვლა;
- მონაცემების ბუფერიზაცია;
- შეცდომების აღმოჩენა.

მ ო ნ ა ც ე მ თ ა ლ ო კ ა ლ ი ზ ა ც ი ი ს ქვეშ გვესმის მიმართვის შესაძლებლობა ერთ-ერთ გარე მოწყობილობაზე, აგრეთვე მონაცემთა გადამისამართება მათზე.

მართვისა და სინქრონიზაციის ფუნქცია იმაში მდგომარეობს, რომ შეტანა/გამოტანის მოდულმა უნდა მოახდინოს მონაცემების გადაადგილების კოორდინაცია გამომთვლელი მანქანის შიგა რესურსებსა და გარე მოწყობილობებს შორის.

შეტანა/გამოტანის მოდულის ძირითად ფუნქციას წარმოადგენს ინფორმაციის გაცვლის უზრუნველყოფა. „დიდი“ ინტერფეისის მხრიდან ესაა ცენტრალურ პროცესორთან გაცვლა, ხოლო „მცირე“ ინტერფეისის მხრიდან - გარე მოწყობილობებთან გაცვლა.

შეტანა/გამოტანის მოდულის მნიშვნელოვან ფუნქციას წარმოადგენს მონაცემთა ბუფერიზაცია. მიუხედავად იმისა, რომ ყველა გარე მოწყობილობას ინფორმაციის გაცვლის სხვადასხვა სიჩქარე გააჩნია, მაინც ყველა ისინი ამ მხრივ მნიშვნელოვნად ჩამორჩებიან ცენტრალურ პრო-

ცესორსა და მეხსიერებას. ასეთი განსხვავება ბუფერიზაციის ხარჯზე კომპენსირდება. გარე მოწყობილობაზე ინფორმაციის გამოტანისას მონაცემები გადაიგზავნებიან ძირითადი მეხსიერებიდან შეტანა/გამოტანის მოდულში მაღალი სისწრაფით. მოდულში ეს მონაცემები ბუფერიზდებიან, ხოლო შემდეგ მიემართებიან გარე მოწყობილობისაკენ მისთვის დამახასიათებელი სისწრაფით. გარე მოწყობილობიდან შეტანისას მონაცემები ისევ ბუფერიზდებიან, რომ არ აიძულონ იმუშაოს მეხსიერება ნელი გადაცემის რეჟიმში. ამრიგად, შეტანა/გამოტანის მოდულს უნდა გააჩნდეს იმის შესაძლებლობა, რომ იმუშაოს როგორც მეხსიერების სიჩქარით, ისე გარე მოწყობილობის სიჩქარით.

შეტანა/გამოტანის მოდულის კიდევ ერთი უმნიშვნელოვანესი ფუნქციაა შეტანა /გამოტანის პროცესში წარმოქმნილი შეცდომების აღმოჩენა. საჭიროა ცენტრალურ პროცესორს გაეგზავნოს შეტყობინება შეცდომის აღმოჩენის ყოველი შემთხვევის შესახებ. უკანასკნელთა წარმოშობის მიზეზები მრავალგვარი ფაქტორებია, რომლებიც პირველ მიახლოებაში შეიძლება შემდეგ ჯგუფებამდე დავიყვანოთ:

- გარემოს ზემოქმედება;
- ელემენტთა ბაზის დაძველება;
- სისტემური პროგრამული უზრუნველყოფა;
- სამომხმარებლო პროგრამული უზრუნველყოფა.

8.2.2. მოდულის სტრუქტურა

შეტანა/გამოტანის მოდულის სტრუქტურა მნიშვნელოვნადაა დამოკიდებული იმ გარე მოწყობილობათა რაოდენობაზე და სირთულეზე, რომლებსაც ის მართავს, თუმცა განზოგადებული სახით ასეთი მოდული შეგვიძლია წარმოვიდგინოთ ნახ. 8.3-ზე მოტანილი ფორმით.

ნახ. 8.3. მეხსიერების/პერიფერიის მოდულის სტრუქტურა

შეტანა/გამოტანის მოდულის კავშირი გამომთვლელი მანქანის ბირთვთან ხორციელდება სისტემური ან სპეციალიზებული სალტის საშუალებით. ამ მხრიდან შეტანა/გამოტანის მოდულში რეალიზდება ე. წ. „დიდი“ ინტერფეისი. დიდი განსხვავება ბრძანებათა სისტემასა და გამომთვლელი მანქანის სალტეთა არქიტექტურაში იმის მიზეზია, რომ „დიდი“ ინტერფეისის მხრიდან შეტანა/გამოტანის მოდულების უნიფიცირება საკმაოდ რთულია და ხშირად შეტანა/გამოტანის მოდულები, შექმნილნი ერთი ტიპის გამომთვლელი მანქანებისათვის, არ შეიძლება გამოყენებულ იქნან სხვებისათვის. მიუხედავად ამისა, სტრუქტურული სახით ისინი საკმაოდ მსგავსებია.

მისგან მოდულში გადაცემული მონაცემები ბუფერიზდებიან მონაცემთა რეგისტრში. ბუფერიზაციის საშუალებით კომპენსირდება განსხვავება გამომთვლელი მანქანის ბირთვის სწრაფქმედებასა და გარე მოწყობილობებს შორის. როგორც წესი, რეგისტრის თანრიგიანობა ემთხვევა მონაცემთა სალტის სიგანეს „დიდი“ ინტერფეისის მხრიდან (2, 4 ან 8 ბაიტი). თავის მხრივ, გარე მოწყობილობების უმრავლესობა ორიენტირებულია ინფორმაციის ბაიტებად გაცვლაზე. ინფორმაციის ბაიტებად გადაგზავნა „ფართო“ სისტემური სალტით ფრიად ეფექტური გადაწყვეტაა, ამიტომ „მცირე“ ინტერფეისის მხრიდან მონაცემთა რეგისტრს ხშირად ავსებენ შეფუთვა/განფუთვის კვანძით (სქემაზე ნაჩვენები არაა). შეტანისას ეს კვანძი უზრუნველყოფს მონაცემთა რეგისტრის მიმდევრობით ბაიტურ შევსებას (შეფუთვა), ხოლო გამოტანისას - რეგისტრის შემცველობის მიმდევრობით ბაიტურ გაცემას გარე მოწყობილობაზე (განფუთვა). შედეგად „დიდი“ ინტერფეისით მონაცემთა გაცვლისას დაკავებულია მონაცემთა სალტის მთელი სიგანე. შეტანა/გამოტანის მოდულში, რომლებიც გათვალისწინებულია გარე მოწყობილობების დიდ რიცხვთან სამუშაოდ, შეიძლება შედიოდეს მონაცემთა რამდენიმე რეგისტრი, რაც საშუალებას არ იძლევა დამოუკიდებლად შევინახოთ თითოეული გარე მოწყობილობის მიმდინარე მონაცემები.

მონაცემთა რეგისტრის გარდა შეტანა/გამოტანის მოდულის შემადგენლობაში აგრეთვე შედის მართვის რეგისტრი და მდგომარეობის რეგისტრი (ან შერწყმული მართვა/მდგომარეობის რეგისტრი).

მ ა რ თ ვ ი ს რ ე გ ი ს ტ რ შ ი ფიქსირდება ცენტრალური პროცესორიდან მიღებული მოდულის ან მასთან მიერთებული გარე მოწყობილობების მართვის სიგნალები. რეგისტრის ცალკეული თანრიგები შეიძლება წარმოადგენდნენ ისეთ ბრძანებებს, როგორიცაა შეტანა/გამოტანის მოდულის რეგისტრების გასუფთავება, გარე მოწყობილობის ჩამოყრა (განულება), კითხვის დაწყება, ჩაწერის დაწყება და ა. შ. რთულ შეტანა/გამოტანის მოდულში არის მართვის რამდენიმე რეგისტრი, მაგალითად მთლიანად მოდულის მმართველი სიგნალების რეგისტრი და თითოეული გარე მოწყობილობისათვის დამოუკიდებელი მართვის რეგისტრი.

მ დ გ ო მ ა რ ე ო ბ ი ს რ ე გ ი ს ტ რ ი ს დანიშნულებაა შეტანა/გამოტანის მოდულის და მასთან მიერთებული გარე მოწყობილობების მდგომარეობის შენახვა. რეგისტრის განსაზღვრული თანრიგის შემცველობა შეიძლება ახასიათებდეს, მაგალითად შეტანის მოწყობილობის მზადყოფნას მონაცემთა მომდევნო ულუფის მისაღებად, გამოტანის მოწყობილობის დაკავებულობას ან გარე მოწყობილობის ავტონომიურ რეჟიმში ყოფნას (off line). შეტანა/გამოტანის მოდულში გამორიცხული არაა მდგომარეობის ერთზე მეტი რეგისტრის არსებობა.

შეტანა/გამოტანის პროცედურა გულისხმობს შეტანა/გამოტანის მოდულის თითოეულ რეგისტრთან ან ცალ-ცალკე გარე მოწყობილობასთან მუშაობის შესაძლებლობას. ასეთი შესაძლებლობა უზრუნველყოფილია სამისამართო სისტემით. თითოეულ მოდულს შეტანა/ გამოტანის სამისამართო სივრცეში (შეთავსებულში ან განცალებულში) გამოეყოფა მისამართების უნიკალური ნაკრები, რომლებშიც მისამართების რაოდენობა დამოკიდებულია სამისამართო ელემენტების რიცხვისაგან. ცენტრალური პროცესორიდან მიღებული მისამართი მისამართის სელექტორის დახმარებით მოწმდება მოცემული შეტანა/გამოტანის მოდულისათვის გამოყოფილი დიაპაზონის კუთვნილებაზე. დას-

ტურის შემთხვევაში DC დეკოდერი ახდენს მისამართის განკოდირებას, ნებას რთავს მუშაობაზე მოდულის ან გარე მოწყობილობის შესაბამისი რეგისტრით.

შეტანა/გამოტანის მართვის კვანძი თავისი არსით ასრულებს შეტანა/გამოტანის მოდულის მართვის ადგილობრივი მოწყობილობის როლს. მას ორი ამოცანა აკისრია: ცენტრალურ პროცესორთან ურთიერთქმედების უზრუნველყოფა და შეტანა/გამოტანის მოდულის ყველა შემადგენლის მუშაობის კოორდინაცია. ცენტრალურ პროცესორთან კავშირი ხორციელდება მართვის ხაზების საშუალებით, რომლებითაც ცენტრალური პროცესორიდან მოდულს მიეწოდებიან სიგნალები, რომლებიც ემსახურებიან შეტანა/გამოტანის ოპერაციათა სინქრონიზაციას. უკუ მიმართულებით გადაეცემიან ის სიგნალები, რომლებიც ინფორმაციას გასცემენ მოდულში მომხდარი შემთხვევების შესახებ, მაგალითად წყვეტის სიგნალები. მართვის ხაზების ნაწილი შეიძლება მოდულის მიერ გამოყენებულ იქნას არბიტრაჟისათვის. მართვის კვანძის მეორე ფუნქცია რეალიზდება მართვის შიგა სიგნალების დახმარებით.

„მცირე“ ინტერფეისის მხრიდან შეტანა/გამოტანის მოდული უზრუნველყოფს გარე მოწყობილობათა მიერთებას და მათთან ურთიერთქმედებას. შეტანა/გამოტანის მოდულის ეს ნაწილი უფრო უნიფიცირებულია, ვინაიდან გარე მოწყობილობები ყოველთვის ექვემდებარებიან ერთ-ერთ სტანდარტულ პროტოკოლს. თითოეულ გარე მოწყობილობას „ემსახურება“ „მცირე“ ინტერფეისის თავისი კვანძი, რომელიც ახორციელებს მოცემული გარე მოწყობილობისათვის მიღებულ ურთიერთქმედების სტანდარტულ პროტოკოლს.

გარე მოწყობილობათა ფართო სპექტრის მართვისას მოდულმა შესაძლებლობის ფარგლებში უნდა გაათავისუფლოს ცენტრალური პროცესორი კონკრეტული გარე მოწყობილობის დეტალების ცოდნისაგან, რომ ცენტრალურმა პროცესორმა შეძლოს ნებისმიერი მოწყობილობის მართვა წაკითხვისა და ჩაწერის მარტივი ბრძანებების დახმარებით. ამ დროს შეტანა/გამოტანის მოდული თავის თავზე იღებს სინქრონიზაციის, მონაცემთა ფორმატების შეთანხმების და ა. შ. ამოცანებს.

შეტანა/გამოტანის მოდულს, რომელიც თავის თავზე იღებს გარე მოწყობილობის დეტალურ მართვას და ურთიერთობას ახორციელებს ცენტრალურ პროცესორთან მხოლოდ მაღალი დონის ბრძანებებით, ხშირად უწოდებენ შეტანა/გამოტანის არხს ან შეტანა/გამოტანის პროცესორს. ყველაზე პრიმიტიულ შეტანა/გამოტანის მოდულს, რომელიც მოითხოვს ცენტრალური პროცესორიდან დეტალურ მართვას, უწოდებენ შეტანა/გამოტანის კონტროლერს ან მართვის კონტროლერს. როგორც წესი, შეტანა/გამოტანის კონტროლერები ტიპიურებია მიკრო ეგმ-ებისათვის, ხოლო შეტანა/გამოტანის არხები-უნივერსალური გამომთვლელი მანქანებისათვის.

8.3. შეტანა/გამოტანის მართვის მეთოდები

გამომთვლელ მანქანებში გავრცელება ჰპოვა შეტანა/გამოტანის ორგანიზაციის სამმა ხერხმა:

- პროგრამულად მართული შეტანა/გამოტანა;

-შეტანა/გამოტანა წყვეტებით;

- მეხსიერებასთან პირდაპირი შეღწევა.

შეტანა/გამოტანის პროგრამული მართვის დროს მასთან დაკავშირებული ყველა მოქმედება სრულდება ცენტრალური პროცესორის ინიციატივით და მისი სრული კონტროლის ქვეშ. ცენტრალური პროცესორი ასრულებს პროგრამას, რომელიც უზრუნველყოფს შეტანა/გამოტანის პროცესის მართვას, მოწყობილობის მდგომარეობის შემოწმების ჩათვლით, შეტანისა ან გამოტანის ბრძანე-

ბების გაცემით. გასცემს რა შეტანა/გამოტანის მოდულისადმი ბრძანებას, ცენტრალური პროცესორი უნდა დაელოდოს მისი შესრულების დასრულებას. ვინაიდან ცენტრალური პროცესორი უფრო სწრაფად მუშაობს, ვიდრე შეტანა/გამოტანის მოდული, ამიტომ ამას მივყავართ დროით დანაკარგებამდე.

შ ე ტ ა ნ ა / გ ა მ ო ტ ა ნ ა წ ყ ვ ე ტ ე ბ ი თ ბევრი რამით ემთხვევა პროგრამული მართვის მეთოდს. განსხვავება იმაში მდგომარეობს, რომ შეტანა/გამოტანის ბრძანების გაცემის შემდეგ ცენტრალურმა პროცესორმა მოწყობილობის მდგომარეობის გამოსავლენად ციკლურად არ უნდა გამოკითხოს შეტანა/გამოტანის მოდული. ამის ნაცვლად პროცესორმა შეიძლება გააგრძელოს სხვა ბრძანებების შესრულება მანამ, სანამ არ მიიღებს მოთხოვნას წყვეტაზე შეტანა/გამოტანის მოდულიდან, რითაც მას ატყობინებენ ადრე გაცემული შეტანა/გამოტანის ბრძანების დასრულების შესახებ. ისევე როგორც პროგრამულად მართული შეტანა/გამოტანის დროს, ცენტრალური პროცესორი პასუხს აგებს მეხსიერებიდან მონაცემების ამოკრეფაზე (გამოტანაზე) და მეხსიერებაში მონაცემების ჩაწერაზე (შეტანაზე).

როგორც შეტანა/გამოტანის სიჩქარის, ასევე ცენტრალური პროცესორის ეფექტურ გამოყენებას უზრუნველყოფს შეტანა/გამოტანის მესამე ხერხი - მ ე ხ ს ი ე რ ე ბ ა ს თ ა ნ პ ი რ დ ა პ ი რ ი წ ვ დ ო მ ა. ამ რეჟიმში ძირითადი მეხსიერება და შეტანა/ გამოტანის მოდული ინფორმაციის გაცვლას ახორციელებენ პირდაპირ, პროცესორის გვერდის ავლით.

8.4. შეტანა/გამოტანის არხები და პროცესორები

შეტანა/გამოტანის სისტემების განვითარებასთან ერთად მათი ფუნქციები რთულდებიან. ასეთი გართულების მთავარი მიზანი ცენტრალური პროცესორის მაქსიმალური გათავისუფლებაა შეტანა/გამოტანის პროცესების მართვისაგან. ამ ამოცანის გადაწყვეტის რამდენიმე გზა ზემოთ იყო განხილული. ამ პრობლემების გადაწყვეტად არსებობს შემდეგი გზები:

1. შეტანა/გამოტანის მოდულის შესაძლებლობების გაფართოება და მისთვის შეტანა/გამოტანის ოპერაციებზე ორიენტირებულ ბრძანებათა სპეციალიზირებულ ნაკრებიანი პროცესორის გამოყენება. ცენტრალური პროცესორი აძლევს მითითებას ასეთ შეტანა/გამოტანის პროცესორს, რომ მან შეასრულოს გამომთვლელი მანქანის მეხსიერებაში შენახული შეტანა/გამოტანის პროგრამა. შეტანა/გამოტანის პროცესორი იღებს და ასრულებს ამ პროგრამის ბრძანებებს ცენტრალური პროცესორის მონაწილეობის გარეშე და შეტყობინების სიგნალს უგზავნის ცენტრალურ პროცესორს მხოლოდ შეტანა/გამოტანის მთელი პროგრამის დასრულების შემდეგ.

2. პუნქტ 1-ში განხილულ შეტანა/გამოტანის პროცესორს ეძლევა თავისი ლოკალური მეხსიერება. ამ შემთხვევაში შესაძლებელია შეტანა/გამოტანის მრავალი მოწყობილობის მართვა ცენტრალური პროცესორის მინიმალური გამოყენებით.

პირველ შემთხვევაში შეტანა/გამოტანის მოდულს ეძახიან შეტანა/გამოტანის არხებს, ხოლო მეორე შემთხვევაში შეტანა/გამოტანის პროცესორს. პრინციპში განსხვავება შეტანა/გამოტანის არხსა და პროცესორს შორის საკმაოდ პირობითია, ამიტომ მომავალში გამოვიყენებთ ტერმინს „არხი“.

შეტანა/გამოტანის სისტემის კონცეფცია შეტანა/გამოტანის არხებით დამახასიათებელია დიდი უნივერსალური გამომთვლელი მანქა-

ნებისათვის (მეინფრეიმებისათვის), სადაც შეტანა/გამოტანის ეფექტური ორგანიზაციის პრობლემა და ცენტრალური პროცესორის მაქსიმალური გამოთავისუფლება მისი ძირითადი ფუნქციის სასარგებლოდ ძალიან მკაცრად დგას. შეტანა/გამოტანის სისტემა შეტანა/გამოტანის არხით შემოთავაზებულ და რეალიზებულ იქნა IBM 360 ოჯახის გამომთვლელ მანქანებში და შემდგომ განვითარება ჰპოვა IBM 370 და IBM 390 ოჯახის გამომთვლელ მანქანებში.

შეტანა/გამოტანის არხიან გამომთვლელ მანქანებში ცენტრალური პროცესორი უშუალოდ არ მონაწილეობს გარე მოწყობილობათა მართვაში. მას ეს ამოცანა დელეგირებული აქვს შეტანა/გამოტანის არხების შემადგენლობაში მყოფი სპეციალიზებული პროცესორისადმი. ცენტრალური პროცესორის ყველა ფუნქცია დაყვანილია შეტანა/გამოტანის არხებში ოპერაციების გაშვებაზე და გაჩერებაზე, აგრეთვე არხის და მასთან მიერთებული გარე მოწყობილობების მდგომარეობის შემოწმებაზე. ამ მიზნებისათვის ცენტრალური პროცესორი იყენებს შეტანა/გამოტანის მხოლოდ 4-იდან 7-მდე ბრძანებას. მაგალითად IBM 360-ში ასეთი ბრძანება ოთხია:

- „დაიწყოს შეტანა/გამოტანა“;
- „გაჩერდეს შეტანა/გამოტანა“;
- შემოწმდეს შეტანა/გამოტანა“;
- „შემოწმდეს არხი“.

შეტანა/გამოტანის არხი ახორციელებს შეტანა/გამოტანის ოპერაციებს ე. წ. არხის პროგრამის შესრულების გზით. არხის პროგრამები თითოეული გარე მოწყობილობისათვის, რომელთანაც გათვალისწინებულია ინფორმაციის გაცვლა, აღწერენ შეტანა/გამოტანის ოპერაციათა საჭირო თანმიმდევრობას და ინახებიან გამომთვლელი მანქანის ძირითად მეხსიერებაში. არხის პროგრამებში ბრძანებების როლს ასრულებენ არხის მმართველი სიტყვები, რომელთა სტრუქტურაც განსხვავდება ჩვეულებრივი მანქანური ბრძანების სტრუქტურისაგან. არხის მმართველი ტიპიური სიტყვა შეიცავს:

- ო პ ე რ ა ც ი ი ს კ ო დ ს, რომელიც შეტანა/გამოტანის არხსა და გარე მოწყობილობისათვის განსაზღვრავს ოპერაციის ტიპს: „ჩაწერა“ (ძირითად მეხსიერებიდან ინფორმაციის გადატანა გარე მოწყობილობაში), „წაკითხვა“ (გარე მოწყობილობიდან ინფორმაციის ძირითად მეხსიერებაში შეტანა), „მართვა“ (მაგნიტურ დისკებზე დამგროვებლის თავაკების, მაგნიტური ლენტის გადაადგილება და ა. შ.);

- მ ი მ თ ი თ ე ბ ლ ე ბ ი - დამატებითი მიწერილობებია, რომლებიც უთითებენ შეტანა/გამოტანის ოპერაციათა უფრო რთულ მიმდევრობებს, მაგალითად შეტანისას გაშვებულ იქნან ცალკეული ჩაწერები და პირიქით - ერთი ბრძანებით შეტანილ იქნას მთელ ძირითად მეხსიერებაში „გაფანტული“ მასივი, როგორც მთლიანი:

- მ ო ნ ა ც ე მ თ ა მ ი ს ა მ ა რ თ ი, რომელიც უთითებს შეტანა/გამოტანის ოპერაციაში გამოყენებულ მეხსიერების არეს;

- მ ო ნ ა ც ე მ თ ა მ თ ვ ლ ე ლ ი, რომელიც ინახავს მონაცემთა გადასაცემი ბლოკის სიგრძეთა მნიშვნელობებს.

გარდა ამისა, არხის მმართველ სიტყვაში შეიძლება იყოს გარე მოწყობილობის იდენტიფიკატორი და ინფორმაცია მისი პრიორიტეტის დონის შესახებ, მითითებები მოქმედებათა შესახებ, რომლებიც უნდა შესრულდნენ შეცდომების წარმოშობისას და ა.შ.

ცენტრალური პროცესორი ახდენს შეტანა/გამოტანის ინიცირებას არხის ინსტრუქტირებით ძირითად მეხსიერებაში არსებული არხის პროგრამის შესრულების აუცილებლობის შესახებ და გამომთვლელი მანქანის მეხსიერებაში ამ პროგრამის საწყისი მისამართის მითითების გზით. შეტანა/გამოტანის არხი მიჰყვება ამ მითითებებს და მართავს მონაცემთა გადაგზავნას. აღვნიშნოთ, რომ არხის მიერ ინფორმაციის გადაგზავნა ხდება მეხსიერებაში პირდაპირი შეღწევის რეჟიმში. გარე მოწყობილობა ურთიერთმოქმედებს არხთან, ღებულობს მისგან ბრძანებებს. ამრიგად, შეტანა/გამოტანის არხიდან გამომთვლელ მანქანაში შეტანა/ გამოტანის მართვა აგებულია იერარქიული სახით. შეტანა/გამოტანის ოპერაციებში მონაწილეობს მოწყობილობათა სამი ტიპი:

- პროცესორი (მართვის პირველი დონე);

- შეტანა/გამოტანის არხი (მეორე დონე):

- გარე მოწყობილობები (მესამე დონე).

მოწყობილობათა თითოეულ ტიპს შეესაბამება მმართველი ინ-ფორმაციის თავისი სახე:

- პროცესორს - შეტანა/გამოტანის ბრძანებები;

- არხს - არხის მმართველი სიტყვები;

- პერიფერიულ მოწყობილობას - ბრძანებები.

შეტანა/გამოტანის არხული სისტემის მქონე გამომთვლელი მან-ქანის სტრუქტურა მოტანილია ნახ. 8.4-ზე.

შეტანა/გამოტანის არხსა და ძირითად მეხსიერებას შორის ინ-ფორმაციის გაცვლა ხორციელდება გამომთვლელი მანქანის სისტემუ-რი სალტის საშუალებით. გარე მოწყობილობები არხს უერთდებიან არა უშუალოდ, არამედ გარე მოწყობილობების მართვის ბლოკების საშუა-ლებით. გარე მოწყობილობების მართვის ბლოკი არხიდან ღებულობს გარე მოწყობილობათა მართვაზე სიგნალებს (წაკითხვა, ჩაწერა, მატა-რებლის ან მაგნიტური თავაკის გადაადგილება და ა. შ.) და გარდაქმნის მათ მოცემული ტიპის გარე მოწყობილობის შესაბამისი მართვის სიგნა-ლებში. ჩვეულებრივ ერთი გარე მოწყობილობების მართვის ბლოკი შე-იძლება მოემსახუროს რამდენიმე ერთი ტიპის გარე მოწყობილობას, მაგრამ სწრაფი გარე მოწყობილობების მისაერთებლად ხშირად გამოი-ყენებიან მართვის ინდივიდუალური ბლოკები. თავის მხრივ, ზოგიერ-თი გარე მოწყობილობა შეიძლება ერთდროულად მიუერთდეს რამდე-ნიმე გარე მოწყობილობების მართვის ბლოკს. ეს საშუალებას იძლევა ვისარგებლოთ სხვა გარე მოწყობილობების მართვის ბლოკის თავისუ-ფალი ტრაქტით, თუ მოცემული გარე მოწყობილობების მართვის ბლო-კი დაკავებულია მასზე მიერთებული ერთ-ერთი პერიფერიული მო-წყობილობის მომსახურებით. ფიზიკურად გარე მოწყობილობების მარ-თვის ბლოკი შეიძლება იყოს დამოუკიდებელი მოწყობილობა ან ინ-ტეგრირებული გარე მოწყობილობასთან ან არხთან.

ნახ.8.4. გამომავალი მეხსიერება მეხსიერების/გამომავალის არხიანი სისტემით

გარე მოწყობილობების მართვის ბლოკსა და შეტანა/გამოტანის არხს შორის ინფორმაციის გაცვლა უზრუნველყოფილია ე. წ. ა რ ხ უ ლ ი ტ რ ა ქ ტ ე ბ ი თ. ჩვეულებრივ თითოეული გარე მოწყობილობების მართვის ბლოკი დაკავშირებულია ერთ-ერთ არხულ ტრაქტთან, მაგრამ აგრეთვე შესაძლებელია მართვის ბლოკის მიერთება ერთდროულად რამდენიმე ტრაქტთან, რაც იმის საშუალებას იძლევა, რომ აცილებულ იქნას არასასურველი დაყოვნებები ერთ-ერთი მათგანის დაკავებულობის გამო.

შეტანა/გამოტანის არხის საზღვრებში ითვლება, რომ თითოეული გარე მოწყობილობა მიერთებულია თავის ქ ვ ე ა რ ხ თ ა ნ. ქვეარხებს აქვთ თავისი უნიკალური ნომრები, რომელთა საშუალებითაც ხდება არხის პროგრამის გადამისამართება კონკრეტულ გარე მოწყობილობაზე. ფიზიკურად ქვეარხი რეალიზებულია მეხსიერების უბნის სახით, რომელშიც ინახებიან მოცემული გარე მოწყობილობის მიერ შესრულებული შეტანა/გამოტანის ოპერაციის პარამეტრები: მისამართისა და მონაცემთა მთვლელის მიმდინარე მნიშვნელობები, შეტანა/ გამოტანის ოპერაციის კოდი და მიმთითებლები, მომდევნო არხის მმართველი სიტყვის მისამართი და სხვა. ამ პარამეტრების შენახვისათვის ჩვეულებრივ გამოყენებულია არხის ლოკალური მეხსიერება.

როგორც უკვე აღვნიშნეთ, გარე მოწყობილობასა და ძირითად მეხსიერებას შორის ინფორმაციის გაცვლა რეალიზებულია მეხსიერებასთან პირდაპირი შეღწევის (მპშ) რეჟიმში, ამასთან ცენტრალური პროცესორისა და არხის ურთიერთქმედებისათვის ამოქმედებულია სიგნალები „მოთხოვნა მპშ“ და „დასტური მპშ“.

ცენტრალური პროცესორისათვის მიმდინარე არხული პროგრამის დასრულების ან მისი შესრულებისას დაშვებული შეცდომების შესახებ შეტყობინებისათვის შეტანა/გამოტანის არხი ცენტრალურ პროცესორში გასცემს სიგნალს „მოთხოვნა წყვეტაზე“. თავის მხრივ, ცენტრალურმა პროცესორმა შეიძლება არხის ყურადღება მიიპყროს სიგნალით „ყურადღება“.

გარე მოწყობილობის არხთან ურთიერთქმედების ორგანიზაციის ხერხი განისაზღვრება ძირითად მეხსიერებასა და გარე მოწყობილობის სწრაფქმედების შეფარდებით. ამ ნიშნით არსებობს გარე მოწყობილობების ორი ჯგუფი: სწრაფები (დამგროვებლები მაგნიტურ დისკებზე, დამგროვებლები მაგნიტურ ლენტებზე), ინფორმაციის მიღებისა და გადაცემის სიჩქარით 1მბაიტი/წმ - მდე; ნელად მოქმედნი (დისპლეები, საბეჭდი მოწყობილობები და სხვა) 1კბაიტი/წმ და ნაკლები სიჩქარით. ძირითადი მეხსიერების სწრაფქმედება მნიშვნელოვნად მაღალია. გარე მოწყობილობის წარმადობის გათვალისწინებით შეტანა/გამოტანის არხში ხორციელდება მუშაობის ორი რეჟიმი: მ უ ლ ტ ი პ ლ ე ქ ს ურ ი (დროის დაყოფის რეჟიმი) და მ ო ნ ო პ ო ლ უ რ ი.

მ უ ლ ტ ი პ ლ ე ქ ს უ რ რ ე ჟ ი მ შ ი რამდენიმე გარე მოწყობილობა არხს ჰყოფს დროში, ამასთან არხთან პარალელურად მომუშავე თითოეული გარე მოწყობილობა უკავშირდება შეტანა/გამოტანის არხს დროის მოკლე შუალედებში მხოლოდ მას შემდეგ, როცა გარე მოწყობილობა მზად იქნება მიიღოს ან გასცეს ინფორმაციის მომდევნო ულუფა (ბაიტი, ბაიტთა ჯგუფი და ა. შ.). ასეთი სქემაა მიღებული შ ე ტ ა ნ ა / გ ა მ ო ტ ა ნ ი ს მ უ ლ ტ ი პ ლ ე ქ ს უ რ არ ხ შ ი. თუ კავშირის სეანსის განმავლობაში გადაიგზავნება ერთი ან რამდენიმე ბაიტი, რომლებიც ერთ მანქანურ სიტყვას ქმნიან, მაშინ არხს ბ ა ი ტ - მ უ ლ ტ ი - პ ლ ე ქ ს უ რ ს უწოდებენ. არხი, რომელშიც კავშირის სეანსის ფარგლებში მონაცემთა გადაგზავნა ბლოკებად ხორციელდება, ბ ლ ო კ - მ უ ლ ტ ი პ ლ ე ქ ს უ რ ი ჰქვია.

მ ო ნ ო პ ო ლ უ რ რ ე ჟ ი მ შ ი არხსა და გარე მოწყობილობას შორის კავშირის დამყარების შემდეგ გარე მოწყობილობა ახდენს არხის მონოპოლიზაციას მთელი იმ დროის განმავლობაში, რაც ესაჭიროება პროცესორის მიერ ინიცირებული არხული პროცესისა და ამ პროგრამით გათვალისწინებული გარე მოწყობილობას და ძირითად მეხსიერებას შორის მონაცემთა გადაგზავნების დასრულებას. არხული პროგრამის შესრულების მთელი დროის განმავლობაში არხი მიუწვდომელი ხდება სხვა გარე მოწყობილოებისათვის. მოცემულ პროცედურას უზ-

რუნველყოფს შეტანა/გამოტანის სელექტორული არხი. აღვნიშნოთ, რომ, ბლოკ-მულტიპლექსურ არხში კავშირის სეანსის ფარგლებში ბლოკის გადაგზავნა ხორციელდება მონოპოლურ რეჟიმში.

8.4.1. არხთა ქვესისტემა

IBM 360-ის ოჯახის უნივერსალური გამომთვლელი მანქანების მომდევნო ოჯახებში - IBM 370-ში და განსაკუთრებით IBM 390-ში არხების ბაზაზე შეტანა/გამოტანის სისტემის კონცეფციამ კიდევ უფრო მაღალი განვითარება ჰპოვა და აისახა, ე. წ. შეტანა/გამოტანის არხულ ქვესისტემაში. ძირითადი იდეა მდგომარეობს ცალკეული შეტანა/გამოტანის არხების ინტეგრირებაში შეტანა/გამოტანის ერთიან სპეციალიზირებულ პროცესორში არხული ტრაქტებისა და ქვეარხების დიდი რიცხვით. გარე მოწყობილობების მართვის ბლოკები ჩვეულებრივ უერთდებიან რამდენიმე არხულ ტრაქტს, რაც საშუალებას იძლევა დინამიურად ვცვალოთ ინფორმაციის გადაგზავნის გზა მათი მიმდინარე დატვირთვის გათვალისწინებით. გარდა ამისა, სხვადასხვა არხულ ტრაქტებს შეიძლება ახასიათებდეთ სხვადასხვა გამტარუნარიანობა და განსაზღვრული გარე მოწყობილობების მიერთებისას ტრაქტების არჩევის დროს შეიძლება გათვალისწინებულ იქნას მათი სწრაფქმედება.

ყველაზე უფრო სრულყოფილი არხული ქვესისტემა გააჩნია IBM 390 ოჯახის გამომთვლელ მანქანებს. მათში გათვალისწინებულია 65536 - მდე ქვეარხისა და 256-მდე არხული ტრაქტის გამოყენება. რეალიზებულია არხული ტრაქტის ორი ტიპი: პარალელური და მიმდევრობითი.

თავისი შესაძლებლობებითა და მოქმედების პრინციპით პარალელური არხული ტრაქტები ანალოგიურია ზემოთ განხილული მულტიპლექსური და სელექტორული არხებისა, მაგრამ

მათგან განსხვავებით უნივერსალურები არიან, ანუ შეუძლიათ იმუშაონ ბაიტ-მულტიპლექსურ, ბლოკ-მულტიპლექსურ და სელექტორულ რეჟიმებში. შეტანა/გამოტანის არხულ ქვესისტემაში ასეთ არხულ ტრაქტებს პარალელურებს ეძახიან, ვინაიდან ისინი უზრუნველყოფენ ინფორმაციის პარალელური კოდით გადაგზავნას.

შეტანა/გამოტანის არხულ ქვესისტემასთან ბეჭდურ-ოპტიკური ხაზებით მიერთებულ გარე მოწყობილობებთან სამუშაოდ გამოიყენებიან მიმდევრობითი არხული ტრაქტები, რომლებიც ახორციელებენ ESCON (Enterprise Systems Connection Architecture) პროტოკოლს. მიმდევრობითი არხული ტრაქტი გათვალისწინებულია ინფორმაციის მხოლოდ მიმდევრობითი კოდით და მხოლოდ სელექტორულ რეჟიმში გადაცემაზე. გარე მოწყობილობების მართვის ბლოკების ESCON ტრაქტთან მიერთებას ემსახურება სპეციალური მოწყობილობები, რომლებსაც ESCON - დირექტორიებს ეძახიან. თითოეულ ასეთ მოწყობილობას შეუძლია 60 გარე მოწყობილობების მართვის ბლოკამდე მიერთება და მათგან 30-დან ინფორმაციის ერთეული გადაეცემა 10 მბაიტი/წმ - მდე სიჩქარით.

გარდა ამისა, შეტანა/გამოტანის არხულ ქვესისტემაში გათვალისწინებულია სპეციალური კომუნიკაციური არხული ტრაქტები გამომთვლელი მანქანების ქსელებთან, მოდემებთან, სხვა სისტემებთან მისაერთებლად.

პრინციპში შეტანა/გამოტანის არხული ქვესისტემის ძირითადი უპირატესობა - არხული ტრაქტების დინამიური გადანაწილება - რაღაც სახით შეიძლება რეალიზებულ იყოს თითოეული ცალკე არხის ფარგლებში. მაგრამ ყველა არხული რესურსის ერთიან არხულ ქვესისტემაში გაერთიანება საშუალებას იძლევა გამოყენებულ იქნას დინამიური გადანაწილებისა და ამ რესურსების გამოყენების ოპტიმალური სტრატეგიები და ამის მეშვეობით შეტანა/ გამოტანის სისტემის ეფექტურობის ხარისხობრივად ახალ დონეზე აყვანა.

9. ძირითადი მიმართულებები პროცესორთა არქიტექტურაში

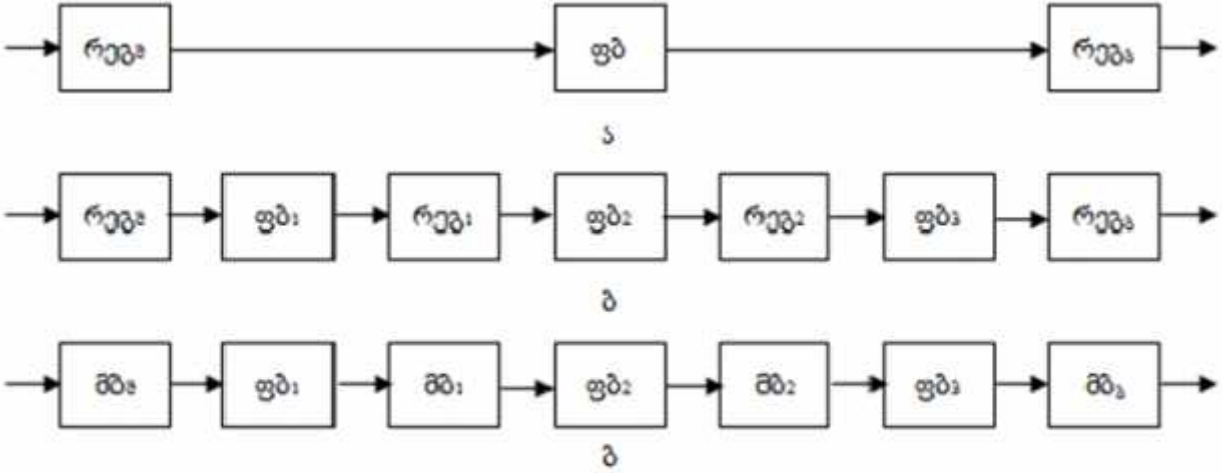
ზემოთ ჩვენ უკვე განვიხილეთ ცენტრალური პროცესორის ძირითადი შემადგენლები. ახლა ყურადღებას გავამახვილებთ პროცესორის როგორც ერთიანი მოწყობილობის არქიტექტურისა და წარმადობის ამაღლების ხერხების საერთო საკითხებზე.

9.1. გამოთვლების კონვეიერიზაცია

ელემენტარული ბაზის სრულყოფა უკვე აღარ იწვევს გამომთვლელი მანქანების წარმადობის კარდინალურ ზრდას. ამ საქმეში უფრო პერსპექტიულად გვესახებიან არქიტექტურული ხერხები, რომელთა შორის ერთ-ერთი ყველაზე მნიშვნელოვანია კონვეიერიზაცია.

კონვეიერის იდეის განსამარტავად ჯერ მივმართოთ ნახ. 9.1, ა -ს, სადაც ნაჩვენებია დამოუკიდებელი ფუნქციური ბლოკი (ფბ). საწყისი მონაცემები თავსდებიან შესასვლელ რეგისტრში (რეგ$_{შ}$), მუშავდებიან ფუნქციურ ბლოკში, ხოლო დამუშავების შედეგი ფიქსირდება გამოსასვლელ რეგისტრში (რეგ$_{გ}$). თუ ფუნქციურ ბლოკში დამუშავების მაქსიმალური დრო T_{max}-ის ტოლია, მაშინ ახალი მონაცემები შესასვლელ რეგისტრში შეიძლება შეტანილ იქნან T_{max} დროის გასვლის შემდეგ.

ახლა გავანაწილოთ ფუნქციურ ბლოკში შესასრულებელი ფუნქციები სამ დამოუკიდებელ ბლოკს - ფბ$_1$, ფბ$_2$ და ფბ$_3$ - შორის (იხ. ნახ. 9.1, ა), თანაც ისე, რომ თითოეულ ფბi-ურ ბლოკში დამუშავების მაქ -



ნახ. 9.1. ინფორმაციის დამუშავება: ა - ერთეულოვან ბლოკში; ბ - რეგისტრებიან კონვეიერში; გ - კონვეიერში ბუფერული მეხსიერებით

სიმალური დრო იყოს ტოლი და უდრიდეს $T_{max/3}$. ბლოკებს შორის მოვათავსოთ ბუფერული რეგi რეგისტრები (ნახ. 9.1, ბ), რომელთა დანიშნულებაა ფბi - ში დამუშავების შედეგის შენახვა, იმ შემთხვევისათვის თუ მისი მომდევნო ფუნქციური ბლოკი ჯერ კიდევ არაა მზად გამოიყენოს ეს შედეგი. განხილულ სქემაში მონაცემები კონვეიერის შესასვლელს შეიძლება მიეწოდებოდნენ $T_{max/3}$ (სამჯერ უფრო ხშირად) ინტერვალით, თუმცა დაყოვნება მონაცემთა პირველი ერთეულის შესასვლელ რეგისტრზე მოწოდების მომენტიდან მისი დამუშავების შედეგის გამოსასვლელი რეგისტრის გამოსასვლელზე გამოჩენის მომენტამდე ისევ T_{max} - შეადგენს, შემდეგი შედეგები გამოსასვლელი რეგისტრის გამოსასვლელზე ჩნდებიან უკვე $T_{max/3}$ ინტერვალით.

პრაქტიკაში იშვიათად შეიძლება მივაღწიოთ იმას, რომ დაყოვნებები თითოეულ ფუნქციურ ბლოკში ერთნაირები იყვნენ. როგორც შედეგი, კონვეიერის წარმადობა ეცემა, ვინაიდან შესასვლელი მონაცემების მიწოდების პერიოდი განისაზღვრება თითოეულ ფუნქციურ ბლოკში მისი დამუშავების მაქსიმალური დროით. ამ ნაკლოვანების გამოსასწორებლად ან, უკიდურეს შემთხვევაში, მისი ნაწილობრივი კონპენსაციის მიზნით თითოეული ბუფერული რეგისტრი (რეგi) საჭიროა შეიცვალოს ბუფერული მეხსიერებით (ბმi), რომელსაც შეუძლია შეინახოს მონაცემთა სიმრავლე და ორგანიზებულია პრინციპით FIFO -„პირველი შევიდა - პირველი გამოვიდა“ (ნახ. 9.1, ბ). დაამუშავებს რა მონაცემთა ელემენტს ფუნქციურ ბლოკს შეაქვს შედეგი i-ურ ბუფერულ მეხსიერებაში, იღებს i-1 ბუფერული მეხსიერებიდან მონაცემთა ახალ ელემენტს და იწყებს დამუშავების ახალ ციკლს. ამასთან ეს თანმიმდევრობა სრულდება თითოეული ფუნქციური ბლოკით სხვა ბლოკებისაგან დამოუკიდებლად. თითოეულ ბლოკში დამუშავება შეიძლება მანამ გრძელდებოდეს, სანამ არ მოხდება წინა რიგის ლიკვიდაცია, ან სანამ არ გადაივსება მომდევნო რიგი. თუ ბუფერული მეხსიერების ტევადობა საკმაოდ დიდია, მაშინ დამუშავების დროის განსხვავებები არ ვლინდებიან წარმადობაზე, მაგრამ მიუხედავად ამისა სასურველია, რომ დამუშავების საშუალო ხანგრძლივობა ყველა ფუნქციურ ბლოკში იყოს ერთნაირი.

გამომთვლელი მანქანების არქიტექტურაში შეიძლება ვიპოვოთ მრავალი ობიექტი, სადაც კონვეიერიზაცია უზრუნველყოფს გამომთვლელი მანქანის წარმადობის საგრძნობ მატებას. ზემოთ უკვე განხილული იყო ორი ასეთი ობიექტი - ოპერაციული მოწყობილობები და მეხსიერება, ოღონდ განსაკუთრებით საგრძნობი ეფექტი მიიღწევა მანქანური ციკლის ეტაპების კონვეიერიზაციის დროს.

საფეხურების მუშაობის სინქრონიზაციის ხერხის მიხედვით კონვეიერები შეძლება იყვნენ სინქრონულები და ასინქრონულები. ტრადიციული გამომთვლელი მანქანებისათვის დამახასიათებელია სინქრონული კონვეიერები. უპირველესად, ეს დაკავშირებულია პროცესორის მუშაობის სინქრონულ ხასიათთან. კონვეიერის საფეხურები პროცესორში ჩვეულებრივ ერთმანეთთან ახლოს არიან განლაგებულნი, რისი მეშვეობითაც სინქრონიზაციის სიგნალების გავრცელების ტრაქტები საკმაოდ მოკლე მიიღება და სიგნალების „გამრუდება“ ხდება არც თუ ისე მნიშვნელოვანი. ასინქრონული კონვეიერები სასარგებლონი ხდებიან, თუ საფეხურებს შორის კავშირები არც თუ ისე ძლიერია, ხოლო სიგნალური ტრაქტების სიგრძეები სხვადასხვა საფეხურებს შორის ძლიერ განსხვავებულია. ასინქრონული კონვეიერების მაგალითს შეიძლება წარმოადგენდნენ სისტოლიკური მასივები.

9.1.1. ბრძანებათა კონვეიერი

ბრძანებათა კონვეიერის იდეა 1956 წელს აკადემიკოს ს. ა. ლებედევის მიერ იქნა შემოთავაზებული. როგორც ცნობილია, ბრძანებების ციკლი წარმოადგენს ეტაპების თანმიმდევრობას. თუ თითოეული მათგანის რეალიზაციას დავაკისრებთ დამოუკიდებელ მოწყობილობას და თანმიმდევრობით გავაერთიანებთ ასეთ მოწყობილობებს, მაშინ მივიღებთ ბრძანებათა კონვეიერის სქემას. ილუსტრაციისათვის ვისარგებ-

ლოთ კონკრეტული მაგალითით. გამოვყოთ ბრძანების ციკლში ექვსი ეტაპი:

1. ბ რ ძ ა ნ ე ბ ი ს ა მ ო რ ჩ ე ვ ა. მეხსიერებიდან მომდევნო ბრძანების ამოკითხვა და შეტანა ბრძანების რეგისტრში.

2. ბ რ ძ ა ნ ე ბ ი ს დ ე კ ო დ ი რ ე ბ ა. ოპერაციის კოდისა და ოპერანდების დამისამართების ხერხების განსაზღვრა.

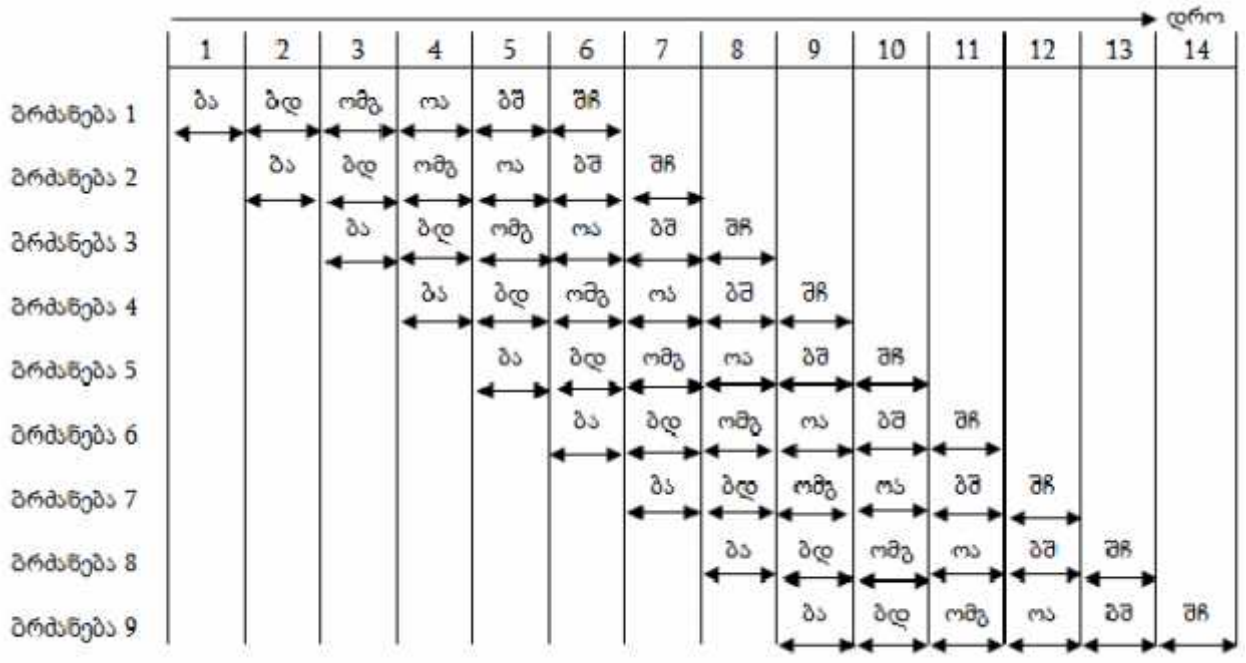
3. ო პ ე რ ა ნ დ ე ბ ი ს მ ი ს ა მ ა რ თ ე ბ ი ს გ ა მ ო თ ვ ლ ა. თითოეული ოპერანდის საშემსრულებლო მისამართის გამოთვლა მათი დამისამართების ხერხის ბრძანებაში მითითების შესაბამისად.

4. ო პ ე რ ა ნ დ ე ბ ი ს ა მ ო რ ჩ ე ვ ა. მეხსიერებიდან ოპერანდების ამოღება. ეს ოპერაცია საჭიროა არაა რეგისტრებში მოთავსებული ოპერანდებისათვის.

5. ბ რ ძ ა ნ ე ბ ი ს შ ე ს რ უ ლ ე ბ ა. მითითებული ოპერაციის შესრულება.

6. შ ე დ ე გ ი ს ჩ ა წ ე რ ა. მეხსიერებაში შედეგის შეტანა.

ნახ. 9.2-ზე ნაჩვენებია კონვეიერი ექვსი საფეხურით, რომლებიც შეესაბამებიან ბრძანების ციკლის ექვს ეტაპს. დიაგრამაში იგულისხმება, რომ თითოეული ბრძანება აუცილებლად გადის ყველა ექვს საფეხურს, თუმცა ეს შემთხვევა სულ მთლად ტიპიური არაა. ასე, მაგალითად, რეგისტრის ჩატვირთვის ბრძანება არ მოითხოვს შედეგის ჩაწერის ეტაპს. გარდა ამისა, აქ მიღებულია, რომ ყველა ეტაპი შეიძლება ერთდროულად სრულდებოდეს. კონვეიერიზაციის გარეშე ცხრა ბრძანების შესრულებას დასჭირდებოდა 9 x 6 = 54 დროის ერთეული. კონვეიერის გამოყენება საშუალებას იძლევა შემცირდეს დამუშავების დრო 14 ერთეულამდე.



ნახ. 9.2. პრძანებათა კონვეიერის მუშაობის ლოგიკა

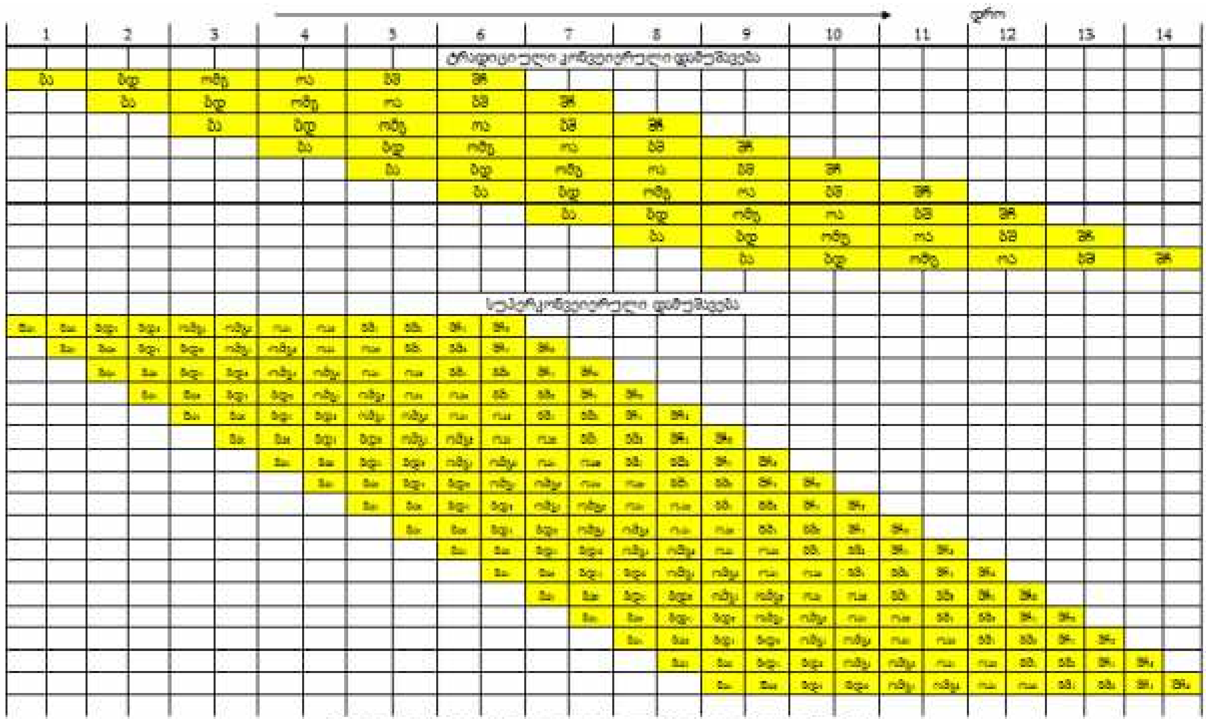
9.1.2. სუპერკონვეიერული პროცესორები

კონვეიერის ეფექტურობა პირდაპირაა დაკავშირებული იმასთან, თუ მის შესასვლელზე რა სიხშირით ხდება დასამუშავებელი ობიექტების მიწოდება. კონვეიერის მუშაობის ტემპის n- ჯერადი ამაღლების მიღწევა ორი გზით შეიძლება:

- კონვეიერის თითოეული საფეხურის n „ქვესაფეხურად“ დაყოფით კონვეიერის შიგნით ტაქტური სიხშირის n-ჯერ გაზრდის პირობებში;

- პროცესორის შემადგენლობაში გადაფარვით მომუშავე n კონვეიერის ჩართვით.

აქ ჩვენ განვიხილავთ პირველ მათგანს, ცნობილს ს უ პ ე რ კ ო ნ ვ ე ი ე რ ი ზ ა ც ი ი ს სახელით (ტერმინი პირველად 1988 წელს იქნა გამოყენებული). სუპერკონვეიერიზაციის ეფექტის საილუსტრაციოდ შეიძლება ნახ. 9.3-ზე მოყვანილი დიაგრამის გამოყენება, რომელზეც განხილულია ზემოთ მოყვანილი მაგალითი (იხ. ნახ. 9.2). სტანდარტული კონვეიერის ექვსი საფეხურიდან თითოეული დაყოფილია ორ უფრო მარტივ ქვესაფეხურად, რომლებიც აღნიშნულია ინდექსებით 1 და 2. ქვესაფეხურებში ოპერაციათა შესრულებას ტაქტური პერიოდის ნახევარი ესაჭიროება. კონვეიერის შიგნით ოპერაციათა ტაქტირება სრულდება სიხშირით, რომელიც ორჯერ აღემატება კონვეიერის „გარე“ ტაქტირების სიხშირეს. რის გამოც კონვეიერის თითოეულ საფეხურზე „გარე“ ტაქტური პერიოდის შუალედში შეიძლება ორი ბრძანების შესრულება.



ნახ. 9.3. ბრძანებათა ტრადიციული და სუპერკონვეიერული დამუშავება

არსებითად სუპერკონვეიერიზაცია დადის კონვეიერის საფეხურების რაოდენობის ზრდაზე როგორც ახალი საფეხურების დამატების ხარჯზე, ასევე არსებული საფეხურების დანაწევრებით რამდენიმე მარტივ ქვესაფეხურად. მთავარი მოთხოვნაა თითოეულ ქვესაფეხურში ოპერაციათა რეალიზაციის შესაძლებლობა უმარტივესი ტექნიკური საშუალებებით, ანუ დროის მინიმალური დანახარჯებით. მეორე, არა ნაკლებ მნიშვნელოვან პირობას წარმოადგენს ყველა ქვესაფეხურზე დაყოვნებების ტოლობა.

პროცესორის სუპერკონვეიერულებზე მიკუთვნების კრიტერიუმს წარმოადგენს ბრძანებათა კონვეიერში საფეხურთა რაოდენობა. სუპერკონვეიერულებს აკუთვნებენ პროცესორებს, რომლებშიც ასეთი საფეხურები ექვსზე მეტია. პირველ სერიულ სუპერკონვეიერულ პროცესორად ითვლება MIPS R4000, რომლის ბრძანებათა კონვეიერი შედგება რვა ქვესაფეხურისაგან. აქ სუპერკონვეიერიზაცია გახდა ბრძანების ამორჩევისა და ოპერანდის ამორჩევის ეტაპების დაყოვნების შედეგი, აგრეთვე კონვეიერში ტეგის შემოწმების ეტაპის შეტანა, რომლის წარმოშობაც განპირობებულია მანქანის ბრძანებათა სისტემის არქიტექტურით.

სამწუხაროდ, სუპერკონვეიერიზაციის ხარჯზე მიღწეული მოგება პრაქტიკაში შეიძლება აღმოჩნდეს მხოლოდ გონებაჭვრეტითი. კონვეიერის გაგრძელება იწვევს არა მხოლოდ ნებისმიერი კონვეიერისათვის დამახასიათებელი პრობლემების გაღრმავებას, არამედ დამატებითი სირთულეების წარმოშობასაც. გრძელ კონვეიერში იზრდება კონფლიქტების ალბათობა. უფრო ძვირი ხდება გადასვლის პროგნოზის შეცდომა - საჭირო ხდება კონვეიერის საფეხურების მეტი რაოდენობის გასუფთავება, რასაც მეტი დრო სჭირდება. რთულდება კონვეიერის საფეხურების ურთიერთქმედების ლოგიკა. მიუხედავად ამისა გამომთვლელი მანქანების შემქმნელები ახერხებენ წარმატებით დაძლიონ ზემოთჩამოთვლილი პრობლემების უმეტესობა, რაზეც მიუთითებს თანამედროვე პროცესორებში ბრძანებათა კონვეიერში საფეხურთა რიცხვის ზრდა (ცხრილი 3).

ცხრილი 3

ბრძანებათა კონვეიერის სიგრძე პოპულარულ მიკროპროცესორებში

მიკროპროცესორის ტიპი	ბრძანებათა კონვეიერში საფეხურების რაოდენობა
MIPS R4400	8
UltraSPARC I	9
Pentium III	10
Itanium	10
UltraSPARC III	14
Pentium 4	20

9.2. არქიტექტურები ბრძანებათა სრული და შეკვეცილი ნაკრებით

პროგრამირების თანამედროვე ტექნოლოგია ორიენტირებულია მაღალი დონის ენებზე, რომელთა მთავარი ამოცანაა პროგრამის დაწერის პროცესის გამარტივება. პროგრამირების მთელი პროცესის 90%-ზე მეტი ხორციელდება მაღალი დონის ენებზე. სამწუხაროდ მაღალი დონის ენებისათვის დამახასიათებელი ოპერაციები განსხვავდებიან მანქანური ბრძანებებით რეალიზებული ოპერაციებისაგან. ამ პრობლემამ

მიიღო სემანტიკური გარღვევის სახელი და მას მივყავართ პროგრამების არასაკმაოდ ეფექტურ შესრულებასთან. ცდილობენ რა გადალახონ სემანტიკური გარღვევა, გამომთვლელი მანქანის დამმუშავებლები აფართოებენ ბრძანებათა სისტემას, ავსებენ მათ ბრძანებებით, რომლებიც ახორციელებენ მაღალი დონის ენებზე რთულ ოპერაციებს აპარატურულ დონეზე, შემოაქვთ გადამისამართებას დამატებითი სახეები და ა. შ. გამომთვლელ მანქანებს, რომლებშიც განხორციელებულია ეს საშუალებები, ეწოდებათ კომპიუტერები ბრძანებათა სრული ნაკრებით (CISC –Complex Instruction Set Computer). CISC-ის ტიპს შეიძლება მივაკუთვნოთ 80-იანი წლების შუამდე გამოშვებული ყველა გამომთვლელი მანქანა და დღეს გამოშვებულ მანქანათა მნიშვნელოვანი ნაწილია.

სემანტიკური გარღვევის პრობლემის გადაწყვეტის CISC-ისათვის დამახასიათებელ ხერხებს, ამასთან ერთად, მივყავართ გამომთვლელი მანქანის არქიტექტურის გართულებასთან, ძირითადად მართვის მოწყობილობის, რაც თავის მხრივ, ნეგატიურად აისახება მთლიანად წარმადობაზე. გარდა ამისა, CISC-ში ძლიერ რთულია ბრძანებების ეფექტური კონვეიერის შექმნა, რომელიც როგორც უკვე აღვნიშნეთ, არის გამომთვლელი მანქანის წარმადობის ზრდის ერთ-ერთ ყველაზე უფრო პერსპექტიული გზა. ყველაფერმა ამან აიძულა დამმუშავებლები გაეანალიზებინათ მაღალი დონის ენების კომპილაციის შემდეგ მიღებული პროგრამები. ჩატარებულ იქნა კვლევების კომპლექსი, რის შედეგადაც გამოვლინდნენ საინტერესო კანონზომიერებანი:

- მაღალი დონის ენების ოპერატორების ექვივალენტური რთული ბრძანებების რეალიზაცია, მოითხოვს მიკროპროგრამულ მართვის მოწყობილობაში მმართველი მეხსიერების ტევადობის გაზრდას. რთული ბრძანებების მიკროპროგრამებმა შეიძლება დაიკავონ მმართველი მეხსიერების 60/%-მდე, იმ დროს როცა პროგრამების მთელ მოცულობაში მათი წილი ხშირად არ აღემატება 0,2%-ს;

- კომპილირებულ პროგრამაში მაღალი დონის ენების ოპერატორები რეალიზდებიან პროცედურების (ქვეპროგრამების) სახით, ამიტომ

პროცედურის გამოძახებისა და მისგან გამოსვლის ოპერაციებზე მოდის გამოთვლითი დატვირთვის 15-იდან 45%-მდე.

- პროცედურის გამოძახებისას გამომძახებელი პროგრამა გადასცემს ამ პროცედურას არგუმენტების რაღაც რაოდენობას. შემთხვევების 98%-ში გადაცემული არგუმენტების რიცხვი არ აღემატება ექვს. დაახლოებით ასეთივე მდგომარეობაა იმ პარამეტრებთანაც, რომლებსაც პროცედურა უბრუნებს გამომძახებელ პროგრამას. პროგრამის მიერ გამოყენებული ცვლადების 80%-ზე მეტი ლოკალურებია, ანუ იქმნებიან პროცედურაში შესვლისას და ნადგურდებიან მისგან გამოსვლისას. დამოუკიდებელი პროცედურით შექმნილი ლოგიკური ცვლადების რაოდენობა შემთხვევათა 92%-ში არ აღემატება ექვს.

- გამოთვლების მსვლელობისას ოპერაციათა თითქმის ნახევარს შეადგენს მინიჭების ოპერაცია, რომელიც დაიყვანება მონაცემთა გადაგზავნაზე რეგისტრებს, მეხსიერების უჯრედებს ან რეგისტრებსა და მეხსიერებას შორის.

კვლევის შედეგების დეტალურმა ანალიზმა დამმუშავებლები მიიყვანა ტრადიციული არქიტექტურული გადაწყვეტილებების სერიოზულ გადასინჯვასთან, რისი შედეგიც გახდა არქიტექტურების ბრძანებათა შეკვეცილი ნაკრებით (RISC- Reduced Instruction Set Computer) გაჩენა. ტერმინი RISC პირველად გამოყენებულ იქნა პატერსონისა და დიტცელის მიერ 1980 წელს.

9.2.1. RISC- არქიტექტურის ძირითადი თვისებები

RISC არქიტექტურაში მთავარი ძალისხმევა მიმართულია მაქსიმალურად ეფექტური ბრძანებების კონვეიერის შექმნაზე, ანუ ისეთის, სადაც ყველა ბრძანება იღება მეხსიერებიდან და მიეწოდება ცენტრალურ პროცესორს დასამუშავებლად თანაბარი ნაკადის სახით, ამასთან არც ერთი ბრძანება არ უნდა იმყოფებოდეს მოლოდინის რეჟიმში, ხო-

ლო ცენტრალური პროცესორი დატვირთული უნდა რჩებოდეს მთელი დროის განმავლობაში. გარდა ამისა, იდეალური იქნება ის ვარიანტი, როცა ბრძანების ციკლის ნებისმიერი ეტაპი სრულდება ერთი ტაქტური პერიოდის განმავლობაში.

ბოლო პირობა შედარებით მარტივად შეიძლება განვახორციელოთ ამორჩევის ეტაპისათვის. მხოლოდ საჭიროა, რომ ყველა ბრძანებას ჰქონდეს ცენტრალური პროცესორისა და მეხსიერების შემაერთებელი მონაცემთა სალტის სიგანის ტოლი სტანდარტული სიგრძე. სხვადასხვა ბრძანებისათვის შესრულების დროის უნიფიკაცია- მნიშვნელოვნად უფრო რთული ამოცანაა, ვინაიდან რეგისტრულებთან ერთად არსებობენ აგრეთვე მეხსიერებაზე მიმართვის ბრძანებები.

ბრძანებების ერთნაირი სიგრძის გარდა, საჭიროა გვქონდეს დეკოდირებისა და მართვის შედარებით მარტივი ქვესისტემა: მართვის რთული მოწყობილობა შეიტანს დამატებით დაყოვნებებს მართვის სიგნალის ფორმირებაში. მართვის მოწყობილობის მნიშვნელოვანი გამარტივების ცხადი გზა - ესაა შესასრულებელი ბრძანებების რიცხვის, ბრძანებებისა და მონაცემების ფორმატების, აგრეთვე გადამისამართების სახეების შემცირება.

ზედმეტია შეგახსენოთ, რომ ბრძანებების შეკვეცილ სიაში უნდა დარჩნენ ისინი, რომელებიც ყველაზე ხშირად გამოიყენებიან. კვლევებმა აჩვენა, რომ ტიპიური პროგრამების შესრულების 80-90% მოდის ბრძანებების შედარებით მცირე ნაწილზე (10-20%). ყველაზე ხშირად მოთხოვნილ მოქმედებებს მიეკუთვნებიან მონაცემთა გადაგზავნა, არითმეტიკული და ლოგიკური ოპერაციები. ძირითადი მიზეზი, რომელიც ხელს უშლის ბრძანების ციკლის ყველა ეტაპის დაყვანას ერთ ტაქტურ პერიოდამდე, ესაა ოპერანდების ამორჩევისათვის და/ან შედეგების ჩაწერისათვის მეხსიერებასთან მიღწევის პოტენციური აუცილებლობა. საჭიროა მაქსიმალურად შემცირდეს იმ ბრძანებათა რიცხვი, რომლებსაც აქვთ მეხსიერებასთან შეღწევის საშუალება. ეს მოსაზრება RISC-ის ზემოთ მოხსენებულ პრინციპებს უმატებს კიდევ ორს:

- შესრულების დროს მეხსიერებასთან წვდომა ხორციელდება მხოლოდ „წაკითხვა“ და ჩაწერა“ ბრძანებებით;

- ყველა ოპერაციას, გარდა „წაკითხვა“ და „ჩაწერა“, აქვთ ტიპი „რეგისტრი - რეგისტრი“.

უმრავლესი ბრძანებების შესრულების გასამარტივებლად და მათ დასაყვანად „რეგისტრი-რეგისტრი“ ტიპთან საჭიროა ცენტრალური პროცესორი აღვჭურვოთ საერთო დანიშნულების რეგისტრების მნიშვნელოვანი რიცხვით. ცენტრალური პროცესორის რეგისტრთა ფაილში რეგისტრების დიდი რიცხვი საშუალებას იძლევა ვუზრუნველვყოთ მომდევნო ოპერაციებში ოპერანდების სახით გამოყენებული შუალედური შედეგების დროებითი შენახვა და მეხსიერებაზე მიმართვების რიცხვის შემცირება, ოპერაციათა შესრულების დაჩქარება. 32-ის ტოლი რეგისტრების მინიმალური რიცხვი მიღებულია როგორც დე-ფაქტო სტანდარტი RISC- კომპიუტერების უმრავლესი მწარმოებლისათვის.

9.2.2. RISC-ის უპირატესობები და ნაკლოვანებები

CISC-ისა და RISC-ის უპირატესობებისა და ნაკლოვანებების შედარებისას შეუძლებელია ცალსახად გავაკეთოთ დასკვნა ერთი არქიტექტურის მეორეზე უდავო უპირატესობის შესახებ. გამომთვლელი მანქანების ცალკეულ სფეროებში გამოყენებისას უკეთესია ერთი ან მეორე არქიტექტურა. მიუხედავად ამისა, ქვემოთ მოყვანილია ძირითადი არგუმენტაცია RISC-არქიტექტურის სასარგებლოდ და საწინააღმდეგოდ.

RISC ტექნოლოგიებისათვის დამახასიათებელია მართვის მოწყობილობის შედარებით მარტივი სტრუქტურა. მართვის მოწყობილობის რეალიზაციისათვის მიკროსქემის კრისტალზე გამოყოფილი ფართობი მნიშვნელოვნად მცირეა. ასე მაგალითად RISC I-ში ის შეადგენს 6%-ს, ხოლო RISC II-ში 10%-ს. როგორც შედეგი, ჩნდება შესაძლებლობა კრი-

სტალზე მოთავსდეს ცენტრალური პროცესორის რეგისტრების დიდი რაოდენობა (138 RISC II-ში). გარდა ამისა, რჩება მეტი ადგილი ცენტრალური პროცესორის სხვა კვანძებისათვის და დამატებითი მოწყობილობებისათვის: კეშ-მეხსიერების, მცურავი მძიმის არითმეტიკის ბლოკის, ძირითადი მეხსიერების ნაწილის, მეხსიერების მართვის ბლოკის, შეტანა/გამოტანის პორტებისათვის.

ბრძანებების ნაკრების უნიფიკაცია, ორიენტაცია ნაკადურ კონვეიერულ დამუშავებაზე, ბრძანებათა ზომებისა და მათი შესრულების ხანგრძლივობის უნიფიკაცია, კონვეიერზე ლოდინის პერიოდების აღმოფხვრა - ყველა ეს ფაქტორი დადებითად აისახება საერთო სწრაფქმედების ამაღლებაზე. მარტივ მართვის მოწყობილობას აქვს ცოტა ვენტილები და, გამომდინარე, კავშირის მოკლე ხაზები მართვის სიგნალების გასატარებლად. ბრძანებების, ფორმატებისა და რეჟიმების მცირე რიცხვი იწვევს დეკოდირების სქემის გამარტივებას და ის უფრო სწრაფად სრულდება. RISC -ში გამოყენებული მართვის მოწყობილობა „ხისტი“ ლოგიკით მიკროპროგრამულზე სწრაფია. მაღალ წარმადობას ხელს უწყობს პროცედურებს შორის პარამეტრების გადაცემის გამარტივებაც. ამრიგად, RISC -ის გამოყენება იწვევს პროგრამის მეხსიერების დროის შემცირებას ან ბრძანებაზე ციკლთა რიცხვის შემცირების ხარჯზე სიჩქარის გაზრდას.

მართვის მოწყობილობის სიმარტივე, ღირებულების შემცირებისა და საიმედობის ზრდასთან ერთად, RISC -ის სასარგებლოდ მეტყველებს. მართვის მოწყობილობის დამუშავება ნაკლებ დროს მოითხოვს. მარტივ მართვის მოწყობილობაში იქნება ნაკლები კონსტრუქციული შეცდომა და ამიტომ ის უფრო საიმედოა.

ბევრ თანამედროვე CISC - მანქანას, ისეთებს როგორებიცაა VAX 11/780, VAX -8600, გააჩნია ბევრი საშუალება მაღალი დონის ენების იმ ფუნქციების პირდაპირი მხარდაჭერისათვის, რომლებიც ყველაზე ხშირად გამოიყენებიან ამ ენებში (პროცედურების მართვა, ოპერაციები მასივებთან, მასივების ინდექსების შემოწმება, ინფორმაციის დაცვა, მეხსიერების მართვა და ა. შ.). RISC - საც გააჩნია რიგი საშუალებებისა

მაღალი დონის ენების უშუალო მხარდაჭერისა და მაღალი დონის ენების კომპილატორების დამუშავების გამარტივებისათვის, რისი მეშვეობითაც ეს არქიტექტურა მაღალი დონის ენების მხარდაჭერის მხრივ არაფრით არ ჩამოუვარდება CISC - ს.

RISC - ის ნაკლოვანებები პირდაპირ არიან დაკავშირებული ამ არქიტექტურის ზოგიერთ უპირატესობასთან. პრინციპული ნაკლოვანებაა ბრძანებების რიცხვის შემცირება: რიგი ფუნქციების შესასრულებლად საჭიროა რამდენიმე ბრძანების გამოყენება ნაცვლად ერთისა CISC-ში. ეს აგრძელებს პროგრამის კოდს, ზრდის მეხსიერების დატვირთვას და ბრძანებათა ტრაფიკს მეხსიერებასა და პროცესორს შორის. უკანასკნელმა გამოკვლევებმა აჩვენეს, რომ RISC -პროგრამა საშუალოდ 30% -ით გრძელია ფუნქციის შემსრულებელ CISC - პროგრამაზე.

თუმცა რეგისტრების დიდი რაოდენობა მნიშვნელოვან უპირატესობებს იძლევა, მაგრამ თავის თავად ის ართულებს რეგისტრის კოდის დეკოდირების სქემას, რის გამოც იზრდება რეგისტრებთან მიღწევის დრო.

უმრავლეს RISC - სისტემებში „ხისტი“ ლოგიკით რეალიზებული მართვის მოწყობილობა ნაკლებ მოქნილია, შეცდომებისადმი უფრო მეტი მიდრეკილება აქვს, ართულებს შეცდომების მოძიებასა და გასწორებას, აგებს რთული ბრძანებების შესრულებისას.

ერთსიტყვიანი ბრძანება გამორიცხავს პირდაპირ გადამისამართებას სრული 32 -ბიტიანი მისამართისათვის. ამიტომ ზოგიერთი მწარმოებელი უშვებს ორმაგი სიგრძის ბრძანებების მცირე ნაწილს, მაგალითად Intel 80960 - ში.

9.3. სუპერსკალარული პროცესორები

ვინაიდან ელემენტთა ბაზის გაუმჯობესების შესაძლებლობა პრაქტიკულად ამოწურულია, ამიტომ გამომთვლელი მანქანების მწარმოებლურობის შემდგომი ზრდა მოთავსებულია არქიტექტურულ გადაწყვეტათა სიბრტყეში. როგორც უკვე ავღნიშნეთ, ამ მხრივ ერთ-ერთი ყველაზე ეფექტური მიდგომა მდგომარეობს გამოთვლით პროცესში პარალელურობის სხვადასხვა დონის შეტანაში. ადრე განხილული ბრძანებათა კონვეიერი ასეთი მიდგომის ტიპიური მაგალითია. იგივე მიზნებს ემსახურებიან არითმეტიკული კონვეიერები, სადაც კონვეიერიზაციაში ჰყვება არითმეტიკული ოპერაციების შესრულების პროცესი. პარალელურობის დამატებითი დონე რეალიზდება ვექტრულ და მატრიცულ პროცესორებში, მაგრამ მხოლოდ ვექტორებისა და მასივების ტიპის მრავალკომპონენტიანი ოპერანდების დამუშავების დროს. აქ მაღალი სწრაფქმედება მიიღწევა ვექტორის ან მასივის ყველა კომპონენტის ერთდროული დამუშავების ხარჯზე, ოღონდ მსგავსი ოპერანდები დამახასიათებელია დასამუშავებელი ამოცანების მხოლოდ საკმაოდ ვიწრო წრისათვის. ჩვეულებრივ გამოთვლითი დატვირთვის ძირითადი მოცულობა მოდის სკალარულ გამოთვლებზე, ანუ ერთეული (ცალკეული) ოპერანდების დამუშავებაზე, მაგალითად ისეთებისა, როგორებიცაა მთელი რიცხვები. მსგავსი გამოთვლებისათვის დამატებითი პარალელიზმი მნიშვნელოვნად რთულად რეალიზდება, მაგრამ, მიუხედავად ამისა, შესაძლებელია და ამის მაგალითია სუპერსკალარული პროცესორები.

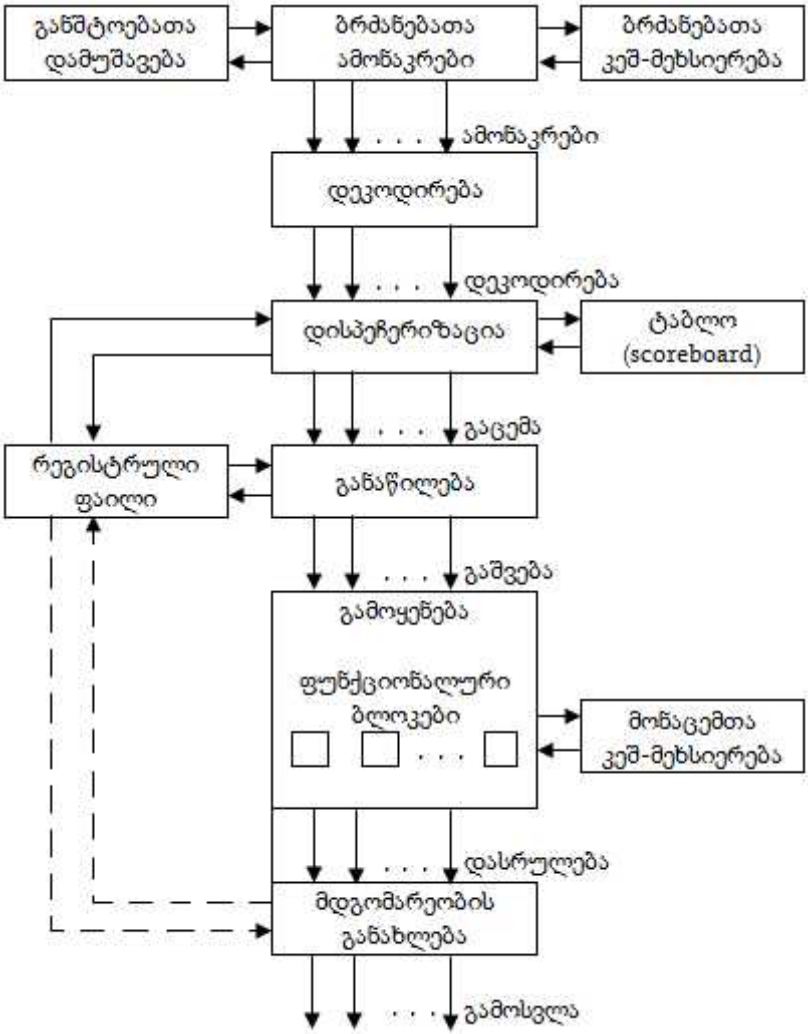
ს უ პ ე რ ს კ ა ლ ა რ უ ლ ი (ეს ტერმინი პირველად 1987 წელს იქნა გამოყენებული) ჰქვია ცენტრალურ პროცესორს, რომელიც ერთდროულად ასრულებს ერთზე მეტ სკალარულ ბრძანებას. ეს მიიღწევა ცენტრალური პროცესორის შემადგენლობაში რამდენიმე დამოუკიდებელი ფუნქციური (შემსრულებელი) ბლოკის, რომელთაგან თითოეული პასუხს აგებს ოპერაციათა თავის კლასზე და შეიძლება პროცესორ-

ში რამდენიმე ეგზემპლიარად იყოს, ჩართვის ხარჯზე. ასე, მაგალითად, Pentium III მიკროპროცესორში მთელრიცხვა არითმეტიკისა და მცურავი მძიმის ოპერაციათა ბლოკები დუბლირებულებია, ხოლო Pentium 4 და Athlon მიკროპროცესორებში - გასამებულებია. ტიპიური სუპერსკალარული პროცესორის სტრუქტურა მოტანილია ნახ. 9.4-ზე. პროცესორი შეიცავს ექვს ბლოკს: ბრძანებების ამორჩევის, ბრძანებების დეკოდირების, ბრძანებების დისპეჩერიზაციის, ბრძანებების ფუნქციურ ბლოკებად განაწილების, შემსრულებელ ბლოკს და მდგომარეობის განახლების ბლოკს.

ბრძანებების ამორჩევის ბლოკი ძირითადი მეხსიერებიდან იღებს ბრძანებებს ბრძანებების კეშ-მეხსიერების გავლით. ეს ბლოკი ინახავს ბრძანებათა მთვლელის რამდენიმე მნიშვნელობას და ამუშავებს პირობითი გადასვლის ბრძანებებს.

დეკოდირების ბლოკი ახდენს კეშ-მეხსიერებიდან ამოღებულ ბრძანებებში არსებული ოპერაციის კოდის გაშიფრას. ზოგიერთ სუპერსკალარულ პროცესორში, მაგალითად Intel ფირმის მიკროპროცესორებში, ამორჩევისა და დეკოდირების ბლოკები შეთავსებულია (შერწყმულია).

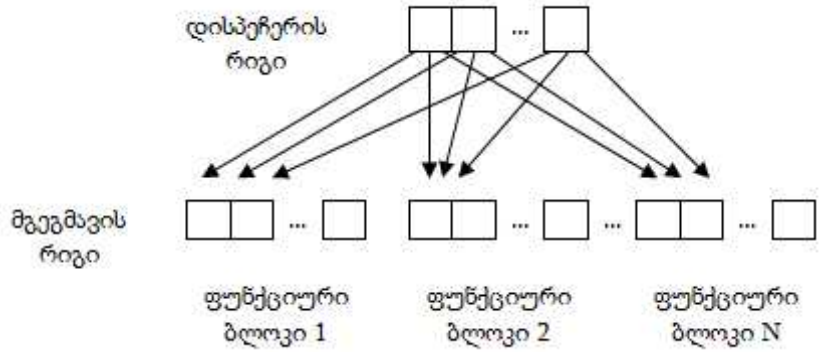
დისპერჩერიზაციისა და განაწილების ბლოკები ურთიერთქმედებენ ერთმანეთს შორის და ერთობლიობაში სუპერსკალარულ პროცესორში ასრულებენ ტრაფიკის კონტროლერის როლს. ორივე ბლოკი ინახავს დეკოდირებული ბრძანებების რიგს. განაწილების ბლოკის რიგი ხშირად ნაწილდება რამდენიმე დამოუკიდე -



ნახ. 9.4. სუპერსკალარული პროცესორის არქიტექტურა

ბელ ბუფერს (ბრძანებათა დამგროვებლების ან რეზერვირების სქემების (reservation station)) შორის, რომელთა დანიშნულებაა დეკოდირებული, მაგრამ ჯერ კიდევ შეუსრულებელი ბრძანებების შენახვა. ბრძანებათა

თითოეული დამგროვებელი დაკავშირებულია თავის ფუნქციურ ბლოკთან, ამიტომ დამგროვებელთა რიცხვი ჩვეულებრივ ფუნქციურ ბლოკთა რიცხვის ტოლია, მაგრამ თუ პროცესორში გამოყენებულია რამდენიმე ერთნაირი ფუნქციური ბლოკი, მაშინ მათ უმატებენ საერთო დამგროვებელს. დისპეჩერიზაციის ბლოკისადმი დამოკიდებულებაში ბრძანებათა დამგროვებლები გამოდიან ვირტუალურ ფუნქციურ მოწყობილობათა როლში. რიგის ორივე სახე ნაჩვენებია ნახ. 9.5-ზე. ზოგიერთ სუპერსკალარულ პროცესორში ისინი გაერთიანებულია ერთ რიგში.



ნახ. 9.5. დისპეჩერიზაციისა და განაწილების რიგები

გარდა ამისა, დისპეჩერიზაციის ბლოკი აგრეთვე ინახავს ტაბლოდ (scoreboard) წოდებულ თავისუფალ ფუნქციურ ბლოკთა სიას. ტაბლო გამოიყენება განაწილების რიგის მდგომარეობისათვის თვალის მისადევნებლად. ციკლის განმავლობაში ერთხელ დისპეჩერიზაციის ბლოკი იღებს ბრძანებებს თავისი რიგიდან, კითხულობს მეხსიერებიდან ან რეგისტრებიდან ამ ბრძანებათა ოპერანდებს, რის შემდეგაც ტაბლოს მდგომარეობისგან დამოკიდებულებით, ათავსებს ბრძანებებისა და ოპერანდების მნიშვნელობებს განაწილების რიგში. ამ ოპერაციას ბრძანებათა გაცემა ჰქვია. განაწილების ბლოკი თითოეულ

ციკლში ამოწმებს თავის რიგში თითოეულ ბრძანებას მისი შესრულებისათვის ყველა აუცილებელი ოპერანდის არსებობაზე და დადებითი პასუხის შემთხვევაში იწყებს ასეთი ბრძანებების შესრულებას შესაბამის ფუნქციურ ბლოკში.

შ ე ს რ უ ლ ე ბ ი ს ბ ლ ო კ ი შეიცავს ფუნქციურ ბლოკთა ნაკრებს. ფუნქციურ ბლოკთა მაგალითებად შეიძლება მოვიტანოთ მთელრიცხვა ოპერაციული ბლოკები, მცურავი მძიმით შეკრებისა და გამრავლების ბლოკები, მეხსიერებასთან მიღწევის ბლოკი. როცა მთავრდება ბრძანების შესრულება, მაშინ მისი შედეგი იწერება და ანალიზდება მ დ გ ო მ ა რ ე ო ბ ი ს გ ა ნ ა ხ ლ ე ბ ი ს ბ ლ ო კ ი თ, რომელიც უზრუნველყოფს განაწილების რიგში იმ ბრძანებებით მიღებული შედეგის აღრიცხვას, სადაც ეს შედეგი გამოდის ერთ-ერთი ოპერანდის სახით.

როგორც ზემოთ იყო აღნიშნული, სუპერსკალარულობა გულისხმობს შემსრულებელი ბლოკების მაქსიმალური რაოდენობის პარალელურ მუშაობას, რაც შესაძლებელია მხოლოდ რამდენიმე სკალარული ნამრავლის ერთდროული შესრულებისას. ბოლო პირობა კარგად ეხამება კონვეიერულ დამუშავებას, ამასთან სასურველია, რომ სუპერსკალარულ პროცესორში რამდენიმე კონვეიერი იყოს, მაგალითად ორი ან სამი.

მსგავსი მიდგომა რეალიზებულია Intel Pentium მიკროპროცესორებში, სადაც ორი კონვეიერია, თითოეული თავისი არითმეტიკულლოგიკური მოწყობილობით (ნახ. 9.6). აღვნიშნოთ, რომ აქ სტანდარტული კონვეიერისაგან განსხვავებით, თითოეულ ციკლში აუცილებელია შევასრულოთ ერთზე მეტი ბრძანების ამორჩევა. შესაბამისად გამომთვლელი მანქანის მეხსიერება უნდა უშვებდეს რამდენიმე ბრძანებისა და ოპერანდის ერთდროულ ამოკითხვას, რაც ყველაზე ხშირად უზრუნველყოფილია მისი მოდულური აგებულების ხარჯზე.

ნახ.9.6. ორ კომპლექსიანი სუპერსკალარული პროცესორი

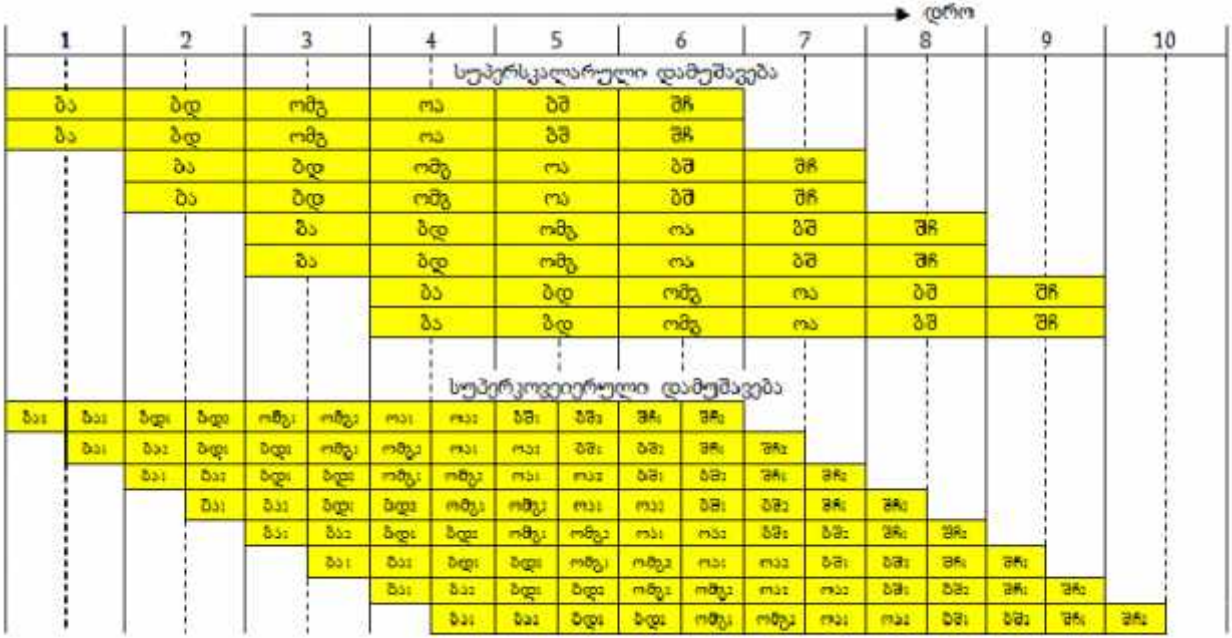
სუპერსკალარული კონვეიერის უფრო ინტეგრირებული მიდგომა ნაჩვენებია ნახ. 9.7-ზე. აქ ამორჩევის ბლოკი იღებს მეხსიერებიდან ერთზე მეტ ბრძანებას და გადასცემს მათ ბრძანებათა დეკოდირების და ოპერანდების მისამართების გამოთვლის საფეხურების გავლით ოპერანდების ამორჩევის ბლოკში. როცა ოპერანდები მისაღწევნი ხდებიან, მაშინ ბრძანებები ნაწილდებიან შესაბამისი შემსრულებელი ბლოკების მიხედვით. მივაქციოთ ყურადღება იმას, რომ ოპერაციები „წაკითხვა“, „ჩაწერა“ და „გადასვლა“ რეალიზდებიან დამოუკიდებელი შემსრულებელი ბლოკებით. სუპერსკალარული პროცესორის მსგავსი სახე გამოიყენება ფირმა Intel-ის Pentium II და Pentium III მიკროპროცესორებში, ხოლო ფორმა სამი კონვეიერით - ფირმა AMD -ს Athlon მიკროპროცესორში.

სხვადასხვა შეფასებით, სუპერსკალარული მიდგომის გამოყენება იწვევს გამომთვლელი მანქანის წარმადობის 1,8-დან 8-მდე ზრდას.

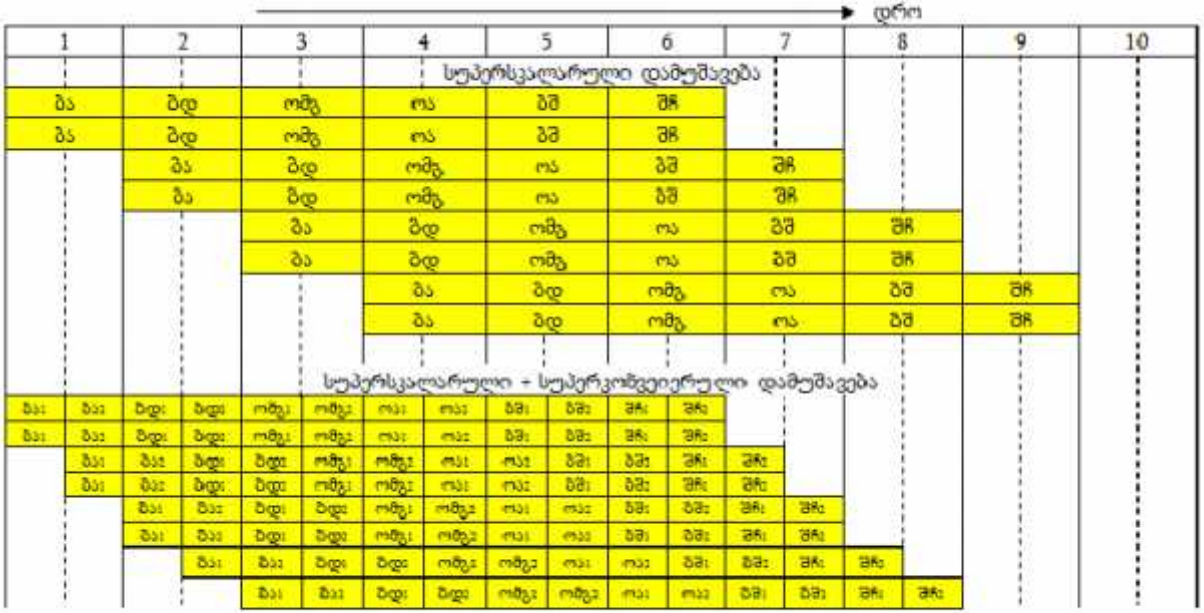
სუპერსკალარული და სუპერკონვეიერული რეჟიმების ეფექტურობის შეფასებისათვის ნახ. 9.8-ზე ნაჩვენებია რვა მიმდევრობითი სკალარული ბრძანების შესრულების პროცესი. ზედა დიაგრამა ასახავს სუპერსკალარულ კონვეიერს, რომელიც უზრუნველყოფს თითოეულ ტაქტურ პერიოდში ორი ბრძანების ერთდროულ შესრულებას. აღვნიშნოთ, რომ შესაძლებელია ისეთი სუპერსკალარული კონვეიერების აგება, რომლებშიც ერთდროულად მუშავდება ბრძანებათა დიდი რაოდენობა.

ზოგიერთი გამომთვლელი მანქანის პროცესორებში რეალიზებულია როგორც სუპერსკალარულობა, ისე სუპერკონვეირულობაც (ნახ. 9.9). ასეთ შეთავსებას ადგილი აქვს ფირმა AMD-ს Athlon და Duron მიკროპროცესორებში, ამასთან ეს მოიცავს არა მარტო ბრძანებათა კონვეიერს, არამედ მცურავი მძიმის ფორმაში რიცხვების დამუშავების ბლოკსაც.

ნახ. 9.7. სუპერკომპიუტერი კომპლექსი სპეციალიზირებული მეხსიერებით მოდულებით



ნახ. 9.8. სუპერსკალარული და სუპერკონვეიერული მიდგომების შედარება



ნახ. 9.9. სტანდარტული სუპერსკალარული და სუპერსკალარული შეთავსებული სქემების ეფექტურობის შედარება

ლიტერატურა

1. Циклер Б. Я., Орлов С. А. Организация ЭВМ и систем: Учебник для вузов.-СПб.: Питер,2004.
2. Таненбаум Э. Архитектура компьютера. 5-е изд.-СПб.: Питер,2007.
3. Степанов А. Н. Архитектура вычислительных систем и компьютерных сетей.-СПб.: Питер, 2007.
4. Костров Б. В., Ручкин В. Н. Архитектура микропроцессорных систем.-М.: Издательство Диалог-МИФИ, 2007.
5. Королев Л. Н. Архитектура электронных вычислительных машин.-М.: Научный мир, 2005.
6. Новожилов О. П. Архитектура ЭВМ и систем.Учебное пособие для бакалавров.-М.: Издательство Юрайт, 2012.
7. Мартиросян С. Т. Организация ЭВМ и систем.-М.: Издательство МГИЭМ, 2007.
8. Бройдо В. Л., Ильина О. П. Архитектура ЭВМ и систем.-Учебник для вузов.-СПб.: Питер,2006.
9. Жмакин А. П. Архитектура ЭВМ.-СПб.: БХВ-Петербург, 2006.
10. Баула В. Г. Введение в архитектуру ЭВМ и системы программирования.-М.: Издательство МГУ, 2003.
11. Александриди Т. М., Котович И. С., Матюхина Е. Н. Организация ЭВМ и систем. Часть 4. Микропроцессорные устройства: Учебное пособие / МАДИ (ГТУ).-М. 2008.
12. Горбачев Д. В. Организация ЭВМ.-Оренбург: Издательство ОГУ, 2009.
13. Ершов С. С. Архитектура и организация ЭВМ: Учебное пособие.-Челябинск: Изд-во ЮурГУ, 2008.
14. Деева Н. В. Архитектура ЭВМ и систем: Учебное пособие.-Новосибирск, СГГА, 2006.
15. Семенихин И. Н. Архитектура ЭВМ.-Шахты: Издательство ЮРГУЭС, 2006.

ს ა რ ჩ ე ვ ი

1. ზოგადი ცნობები კომპიუტერის შესახებ 3
   1.1. კომპიუტერის შექმნის წინაისტორია 3
   1.2. კომპიუტერების კლასიფიკაცია 8
   1.3. მეხსიერებაში შენახული პროგრამის მქონე გამომთვლელი მანქანის კონცეფცია 16
   1.4. გამომთვლელი მანქანებისა და სისტემების სტრუქტურათა ტიპები 24
      1.4.1.გამომთვლელი მანქანების სტრუქტურები 24
      1.4.2. გამოთვლითი სისტემების სტრუქტურები 27
   1.5. გამომთვლელი მანქანებისა და გამომთვლელი სისტემების გაუმჯობესების პერსპექტივები 28
2. ბრძანებათა სისტემის არქიტექტურა 33
   2.1. ბრძანებათა სისტემის არქიტექტურის კლასიფიკაცია 34
   2.2. ოპერანდთა ტიპები და ფორმატები 52
   2.3. ბრძანებათა ტიპები 57
   2.4. რძანებათა ფორმატები 57
3. ფონ ნეიმანისეული გამომთვლელი მანქანის ფუნქციური ორგანიზაცია 63
   3.1. ფონ ნეიმანისეული გამომთვლელი მანქანის ფუნქციური სქემა 63
      3.1.1. მართვის მოწყობილობა 64
4. სალტეთა ორგანიზაცია 74
   4.1. სალტეთა ტიპები 79
   4.2. სალტეთა იერარქია 82
   4.3. სალტეთა არბიტრაჟი 86
5. მეხსიერება 89
   5.1. მეხსიერების სისტემის მახასიათებლები 89
   5.2. მეხსიერების მოწყობილობათა იერარქია 93
   5.3. ძირითადი მეხსიერება 97

5.4. სტეკური მეხსიერება 99
5.5. ასოციაციური მეხსიერება 101
5.6. კეშ - მეხსიერება 110
5.7. ვირტუალური მეხსიერება 113
5.8. გარე მეხსიერება 115
6. მართვის მოწყობილობები 116
6.1. ცენტრალური მართვის მოწყობილობების ფუნქციები . . 116
6.2. მართვის მოწყობილობის მოდელი 117
6.3. მართვის მოწყობილობის სტრუქტურა 119
7. გამომთვლელი მანქანების ოპერაციული მოწყობილობები 124
7.1. ოპერაციული მოწყობილობების სტრუქტურები 126
7.1.1. ოპერაციული მოწყობილობები ხისტი სტრუქტურით 127
7.1.2. ოპერაციული მოწყობილობები მაგისტრალური სტრუქტურით 129
7.1.2.1. მაგისტრალურ სტრუქტურიანი ოპერაციული მოწყობილობების კლასიფიკაცია 131
7.1.2.2. მაგისტრალური ოპერაციული მოწყობილობის საერთო დანიშნულების რეგისტრის კვანძის ორგანიზაცია 134
7.1.2.3. მაგისტრალური ოპერაციული მოწყობილობის ოპერაციული ბლოკის ორგანიზაცია 135
8. შეტანა / გამოტანის სისტემები 140
8.1. გარე მოწყობილობები 144
8.2. შეტანა/გამოტანის მოდულები 147
8.2.1. მოდულის ფუნქციები 147
8.2.2. მოდულის სტრუქტურა 148
8.3. შეტანა/გამოტანის მართვის მეთოდები 153
8.4. შეტანა/გამოტანის არხები და პროცესორები 155
8.4.1. არხთა ქვესისტემა 162
9. ძირითადი მიმართულებები პროცესორთა არქიტექტურაში 164

9.1. გამოთვლების კონვეიერიზაცია 164
9.1.1. ბრძანებათა კონვეიერი 167
9.1.2. სუპერკონვეიერული პროცესორები 170
9.2. არქიტექტურები ბრძანებათა სრული და შეკვეცილი ნაკრებით 173
9.2.1. RISC- არქიტექტურის ძირითადი თვისებები 175
9.2.2. RISC-ის უპირატესობები და ნაკლოვანებები 177
9.3. სუპერსკალარული პროცესორები 180

ლ ი ტ ე რ ა ტ უ რ ა 190

კომპიუტერის არქიტექტურა და ორგანიზება

ავტორი:

თენგიზ ხინიკაძე - ტექნიკურ მეცნიერებათა კანდიდატი, ბათუმის შოთა რუსთაველის სახელმწიფო უნივერსიტეტის განათლებისა და მეცნიერებათა ფაკულტეტის კომპიუტერულ მეცნიერებათა დეპარტამენტის ასოცირებული პროფესორი