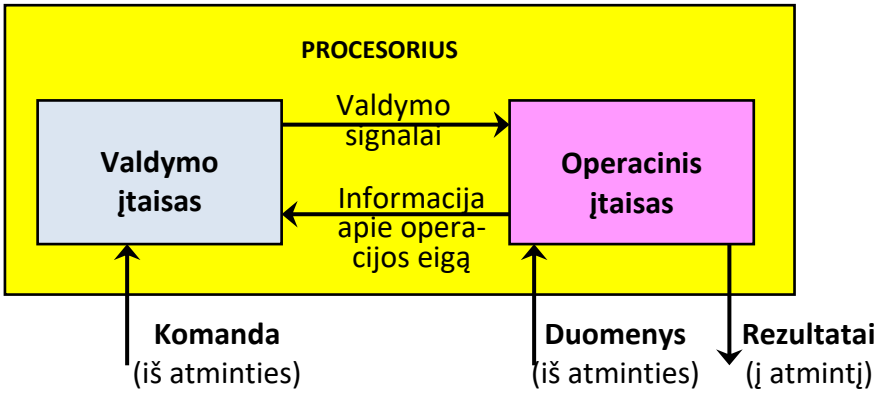
Mikroprogramavimo pagrindai

Pagal fon Neimano architektūros sampratą informaciją apdoroja duomenų procesorius, o apdorojimo procesą valdo komandų procesorius. Šiuos du procesorius dabar priimta laikyti vienu įtaisu – centriniu procesoriumi (CPU), kurį vadiname tiesiog ***procesoriumi***. Tada fon Neimano komandų ir duomenų procesorius atitiks valdymo ir operacinis įtaisai (1 pav.).



1 pav. Valdymo ir operacinis įtaisai

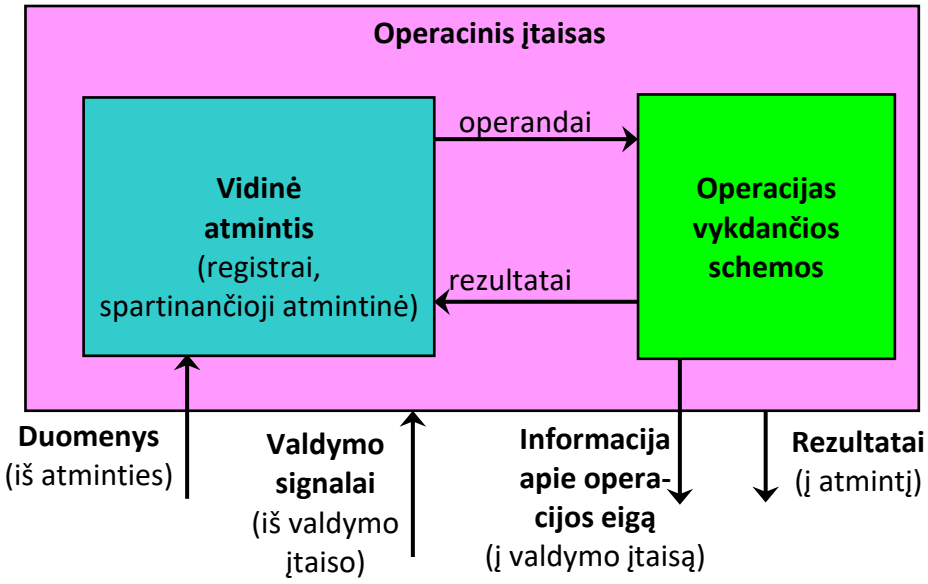
Valdymo įtaisas išrenka iš atminties komandas, jas analizuoja ir valdo operacinio įtaiso darbą (jame vykdomas operacijas, kreipinius į atmintį duomenims išrinkti ar rezultatui įrašyti). Operacinis įtaisas vykdo operacijas, kurias apsprendžia komanda, o operacijų seką nurodo iš valdymo įtaiso gaunami valdymo signalai. Šie du įtaisai dirba kartu. Valdymo įtaisas pagal operacijos kodą formuoja signalus, valdančius operacinio įtaiso darbą. Šis perduoda valdymo įtaisui signalus, informuojančius apie operacijos eigą: nuo susidariusios situacijos, pavyzdžiui, nuo operando ženklo, jo kurio nors bito reikšmės gali priklausyti paskesnių valdymo įtaiso signalų formavimas.

Procesoriaus operaciniame įtaise galima išskirti dviejų tipų schemas (2 pav.):

- vidinę atmintį, kuri reikalinga apdorojamiems duomenims (operandams) laikyti; ją sudaro registrai, atskiri trigeriai, spartinančioji atmintinė, kai kuriuose įtaisuose – dėklas (angl. *stack*);
- operacijas vykdančios schemos, kurios atlieka visus informacijos apdorojimui reikalingus veiksmus – sudėtį, logines operacijas, postūmius ir t. t.

Procesoriaus valdymo įtaisas gali būti projektuojamas taikant du pagrindinius principus:

- mikroprogramavimą;
- „kietąją logiką“.



2 pav. Procesoriaus operacinis įtaisas

Mikroprograma – komandos vykdymo aprašas, kai operuojama atskirų procesoriaus komponenčių valdymo signalais (mikrooperacijomis) ir loginėmis sąlygomis, informuojančiomis apie operacijos eigą. Mikroprograma – tai tarsi žemesnio lygmens programa, išsamiai aprašanti komandos realizaciją procesoriuje. Kai kurių sudėtingesnių komandų mikroprogramos buvo vykdomos net per kelis šimtus ciklų.

Mikroprograminiai procesoriai buvo populiarūs prieš keturis dešimtmečius. Mikroprogramavimą kaip procesoriaus projektavimo pagrindą pasiūlė M. Wilkesas[1] dar 1951 metais, tačiau šis metodas buvo pritaikytas kur kas vėliau.

Kuo ypatinga tokia procesorių realizacija? Reikia paminėti šiuos mikroprogramavimo ypatumus:

- paprastesnis projektavimo procesas;
- lengva įgyvendinti bet kokio sudėtingumo operacijas;
- patogu atlikti derinimo ir modifikavimo darbus;
- galima keisti komandų sistemą, pritaikant procesorių specialiems tikslams;
- pertrauktys gali būti apdorojamos tiek komandų, tiek ir mikrokomandų lygiu;
- komanda įvykdoma dideliu žingsnių skaičiumi.

Mikroprograminiams procesoriams realizuoti buvo sukurti modulinių ir sekcijinių mikroprocesorių rinkiniai. Moduliniuose rinkiniuose atskiros didelės integracijos schemos atitinka funkciniu požiūriu išbaigtus elementus (aritmetinį-loginį įtaisą, operacijų valdymo schemą, mikrokomandų atmintį ir pan.). Sekcijiniuose rinkiniuose viena mikroprocesoriaus sekcija apdoroja nedidelio skaičiaus skilčių (2, 4 ir pan.) žodžius. Didesnio skiltiškumo skaičiams apdoroti sujungiamas atitinkamas skaičius sekcijų. Valdymo įtaisas – atskiras sekcijinio rinkinio elementas.

[1] Wilkes, M. V., Stringer, J. B. Microprogramming and the Design of the Control Circuits of an Electronic Digital Computer. Proc. Camb. Phil. Soc. vol. 49, p. 230 (1953).

„***Kietoji logika***“ – valdymo įtaisų konstravimo būdas, kai valdymo signalai formuojami elektroninėmis schemomis, sudarytomis iš loginių ir atminties elementų (anksčiau realizuotų vidutinės integracijos elementais, dabar – kaip mikroprocesoriaus sudėtinis komponentas). Žodis „kietoji“ pabrėžia, kad sukonstravus valdymo įtaisą jo modifikacija praktiškai negalima.

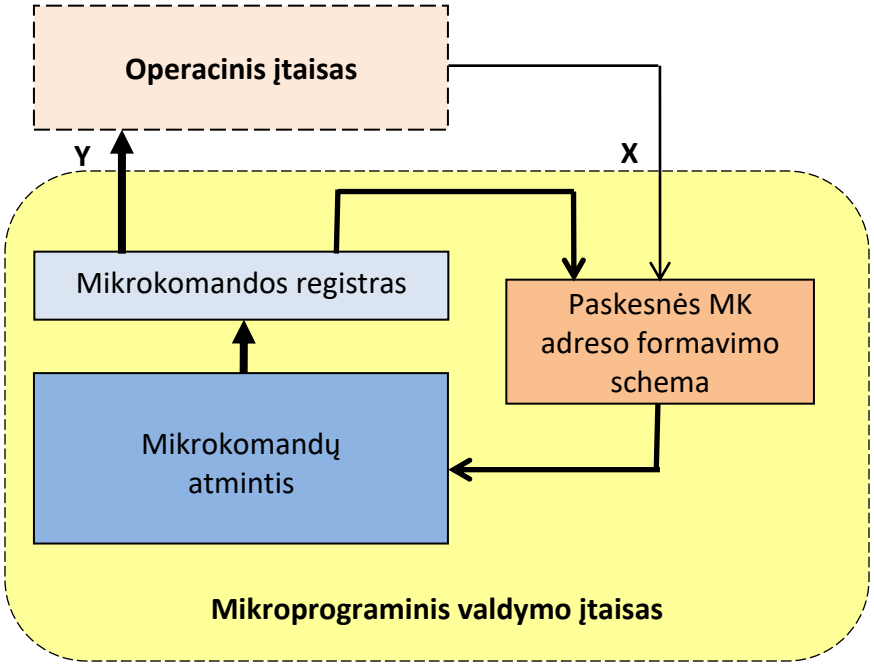
Teoriniu „kietosios logikos“ projektavimo pagrindu yra baigtinių automatų teorija. Šis valdymo įtaisų kūrimo būdas naudojamas dabar, kai komandoms realizuoti pakanka kelių žingsnių.

Mikroprogramininis valdymo įtaisas

Pagrindiniai mikroprogravimo elementai yra:

- **mikrooperacijos** – valdymo signalai, inicijuojantys elementarias operacijas jas vykdančiose schemose arba kitaip valdantys šių schemų darbą; mikrooperacijos pavyzdžiu gali būti informacijos įrašymas į registrą, skaitiklio turinio padidinimas ar sumažinimas, aritmetinio-loginio įtaiso vykdomos operacijos nurodymas ir pan.;
- **mikrokomandos** – dvejetainiai žodžiai, užduodantys operacinio įtaiso darbo cikle vykdomų mikrooperacijų rinkinį ir/ar informaciją, reikalingą paskesnės mikrokomandos išrinkimui;
- **mikrokomandų atmintis**, kurioje saugomos ir pagal suformuotą paskesnės mikrokomandos adresą išrenkamos mikrokomandos;
- **mikroprograma** – mikrokomandų masyvas, aprašantis komandų realizaciją procesoriuje;
- **loginės sąlygos** – signalai, informuojantys apie operacijų eigą operacijas vykdančiose schemose.

Jei valdymo įtaisas kuriamas mikroprogramavimo principų pagrindu, jo struktūra gali būti pavaizduota taip:



3 pav. Mikroprogramininio valdymo įtaiso struktūra

Mikroprogramininio valdymo įtaiso struktūroje išskirsime tokius komponentus:

- ***mikrokomandų atmintį***, iš kurios pagal suformuotą paskesnės mikrokomandos adresą išrenkama mikrokomanda;
- ***mikrokomandos registrą***, kuriame saugoma išrinkta mikrokomanda jos vykdymo metu;
- ***paskesnės mikrokomandos adreso formavimo schemą***, kuri pagal mikrokomandoje esančią informaciją ir (jei to reikia) tikrinamos loginės sąlygos reikšmę formuoja paskesnės mikrokomandos adresą.

Mikroprogramininis valdymo įtaisas funkcionuoja taip:

1. Įtaisui pradedant darbą, nurodomas pirmosios mikrokomandos adresas, ir iš mikrokomandų atminties išrenkama pirmoji mikrokomanda.
2. Iš jos išskiriamos mikrooperacijos **Y**, kurios perduodamos į operacijas vykdančias schemas.
3. Formuojamas paskesnės mikrokomandos adresas, panaudojant mikrokomandoje esančią informaciją ir, jei reikia, patikrinus nurodytos loginės sąlygos **X** reikšmę.
4. Iš mikrokomandų atminties išrenkama nauja mikrokomanda, ir veiksmai kartojami iki bus baigtas mikroprogramos vykdymas.

Mikrokomandos formatas priklauso nuo naudojamo mikrokomandų adresacijos būdo ir mikrooperacijų kodavimo principo.

Mikrokomandų adresacija ir formatai

Žinomi du pagrindiniai mikrokomandų adresacijos būdai:

- ***priverstinė adresacija***, kai kiekvienoje mikrokomandoje aiškiai nurodomas paskesnės mikrokomandos adresas **A** (sąlyginio perėjimo (išsišakojimo) atveju – vienos iš paskesnių mikrokomandų adresas ar abiejų mikrokomandų adresai); lauke **X** nurodomas tikrinamos loginės sąlygos kodas (numeris; besąlyginio perėjimo atveju paprastai **X** = 0);
- ***natūrali adresacija***, kai įvertinama tai, kad dažnai sutinkamos mikrokomandų grandinėlės, kuriose viena mikrokomanda seka po kitos; čia naudojami skirtingi mikrokomandų formatai (formato variantą nurodo bito **P** reikšmė):
  - operacinėje mikrokomandoje nurodomos tik inicijuojamos mikrooperacijos (**Y_i**);
  - adresinėje mikrokomandoje aiškiai nurodomas paskesnės mikrokomandos adresas **A** besąlyginio perėjimo atveju, o sąlyginio perėjimo (išsišakojimo) atveju – tikrinamos loginės sąlygos kodas (numeris) **X** ir paskesnės mikrokomandos, nesančios vykdomų mikrokomandų grandinėlėje, adresas **A**.

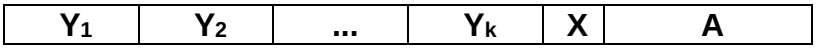
Tokie mikrokomandų formatai parodyti 4 paveiksle.

Parodysime, kaip mikrokomandoms priskiriami adresai abiem adresacijos atvejais. Mikroprogramą vaizduosime mikroprogramos grafo pavidalu; stačiakampio formos viršūnėse nurodysime vykdomas mikrooperacijas, o rombo formos viršūnėse – tikrinamas logines sąlygas.

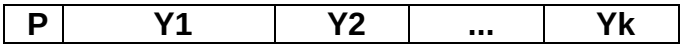
Priverstines adresacijos atveju mikrokomandą gali atitikti:

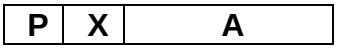
- viena operatorinė viršūnė, kurioje nurodytas vykdomų mikrooperacijų rinkinys (5 pav., a);
- operatorinė viršūnė, kurioje nurodytas vykdomų mikrooperacijų rinkinys, ir po jos esanti salygos tikrinimo viršūnė (5 pav., b);

- tik salygos tikrinimo viršūnė (5 pav., c).



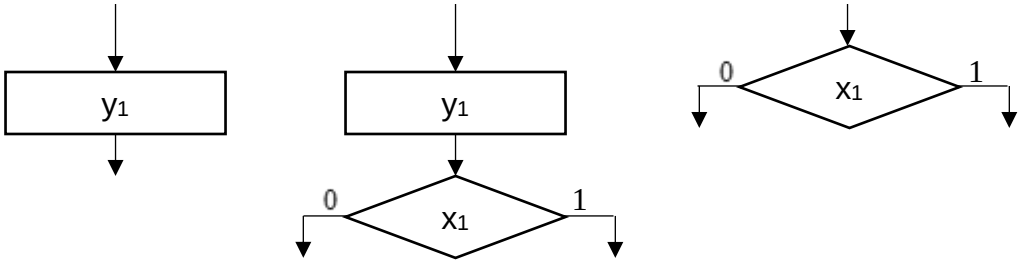
a) Priverstinės adresacijos mikrokomandų formatas





b) Natūralios adresacijos mikrokomandų formatai

4 pav. Mikrokomandų formatai

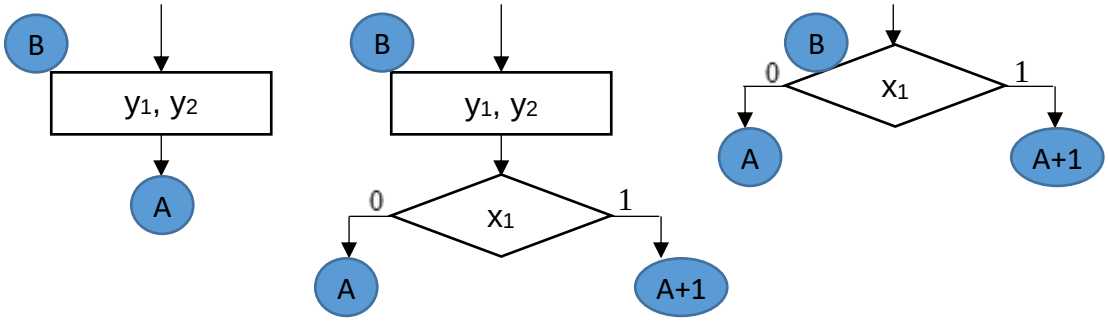


a) b) c)

5 pav. Mikrokomandų variantai priverstinės adresacijos atveju

Priverstinės adresacijos atveju adresai mikrokomandoms priskiriami pagal tokias taisykles:

1) Jei mikrokomandoje loginė sąlyga nėra tikrinama, paskesnei mikrokomandai gali būti priskirtas bet koks adresas (6 pav., a; paveiksle mėlyname apskritime ar ovale įrašyti simboliniai mikrokomandų adresai: B – vykdomos mikrokomandos adresas, A ir A+1 – paskesnės mikrokomandos adresai).
2) Jei mikrokomandoje tikrinama loginė sąlyga, paskesnei mikrokomandai adresas priskiriamas taip: mikrokomandai, kuri vykdoma, kai tikrinamos loginės sąlygos reikšmė lygi 0, priskiriamas adresas A, o mikrokomandai, kuri vykdoma, kai tikrinamos loginės sąlygos reikšmė lygi 1, priskiriamas adresas A+1 (t.y., vienetu didesnis) – žr. 6 pav., b ir c atvejus.

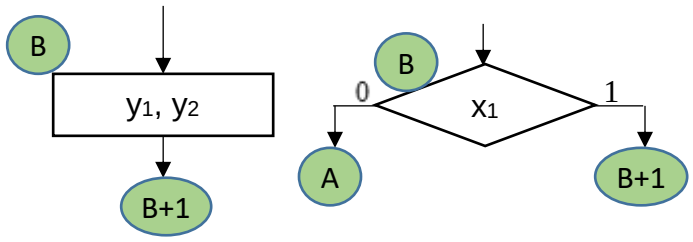


a) b) c)

6 pav. Adresų priskyrimas priverstinės adresacijos atveju

Natūralios adresacijos atveju mikrokomandą gali atitikti:

- operatorinė viršunė, kurioje nurodytas vykdomų mikrooperacijų rinkinys (7 pav., a) arba
- salygos tikrinimo viršūnė, kurioje nurodytas tikrinamos loginės sąlygos kodas (numeris) (7 pav., b).



a) b)

7 pav. Adresų priskyrimas natūralios adresacijos atveju

Natūralios adresacijos atveju adresai mikrokomandoms priskiriami pagal tokias taisykles:

1) Jei mikrokomandoje nurodytos vykdomos mikrooperacijos (tokią mikrokomandą vadinsime operacine), paskesnei mikrokomandai priskiriamas adresas, vienetu didesnis už vykdomos mikrokomandos adresą (7 pav., a).
2) Jei mikrokomandoje tikrinama loginė sąlyga (tokią mikrokomandą vadinsime perėjimo mikrokomanda), paskesnei mikrokomandai adresas priskiriamas taip: mikrokomandai, kuri vykdoma, kai tikrinamos loginės sąlygos reikšmė lygi 0, priskiriamas adresas A, o mikrokomandai, kuri vykdoma, kai tikrinamos loginės sąlygos reikšmė lygi 1, – adresas B+1 (t.y., vienetu didesnis nei vykdomos mikrokomandos adresas; žr. 7 pav., b).
3) Besąlyginio perėjimo atveju X = 0, o paskesnei mikrokomandai priskiriamas adresas A.

Mikrokomandų adresavimo tvarka ir ypatumai

Didžiausią laisvę turime priverstinės adresacijos atveju – vienoje grandinėlėje esančių mikrokomandų adresai gali būti priskirti bet kokia tvarka. Tačiau reikia atidžiai adresuoti mikrokomandas mikroprogramos išsišakojimo atveju – po išsišakojimo (sąlygos tikrinimo) esančioms mikrokomandoms privalo būti priskirti gretimi adresai (žr. 6 pav., b ir c). Tai galima užrašyti tokia formule:

$$\mathbf{A}_{t+1} = \mathbf{A} + \mathbf{x}_X;$$

čia $\mathbf{A}_{t+1}$ – paskesnės mikrokomandos adresas, **A** – mikrokomandos lauko A turinys, $\mathbf{x}_X$ – loginės sąlygos, kuri nurodyta lauke X, reikšmė (0 arba 1).

Todėl pirmiausia reikia atkreipti dėmesį į mikroprogramos išsišakojimo vietas ir joms teikti pirmenybę priskiriant adresus.

Tarkime, turime mikroprogramos grafą, kuris parodytas žemiau. Priverstines adresacijos atveju mikrokomandų adresai gali būti priskirti taip, kaip pavaizduota 8 paveiksle. Kiekviena mikrokomanda įrėminta žalios spalvos rėmeliu, o jai priskirtas adresas užrašytas šalia.

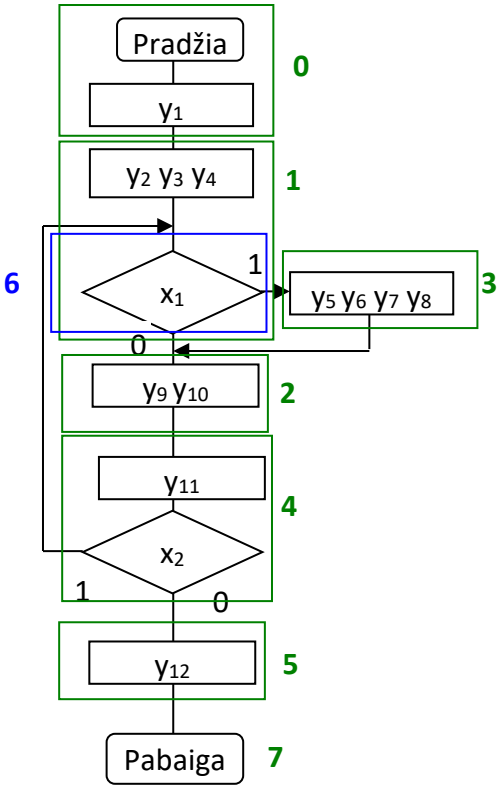
Atkreipkite dėmesį į tai, kad esant x_2=1 vykdomas ciklas, grįžtant į 1 mikrokomandos „vidurį“. Todėl sąlygos x_1 tikrinimas išskirtas ir į atskirą mikrokomandą, kuriai priskirtas adresas 6 (to reikia, nes

mikrooperacijos y_2, y_3, y_4 cikle neturi būti vykdomos). Tad ši sąlyga pirmąjį kartą tikrinama vykdant mikrokomandą 1, o vėliau – vykdant mikrokomandą 6.

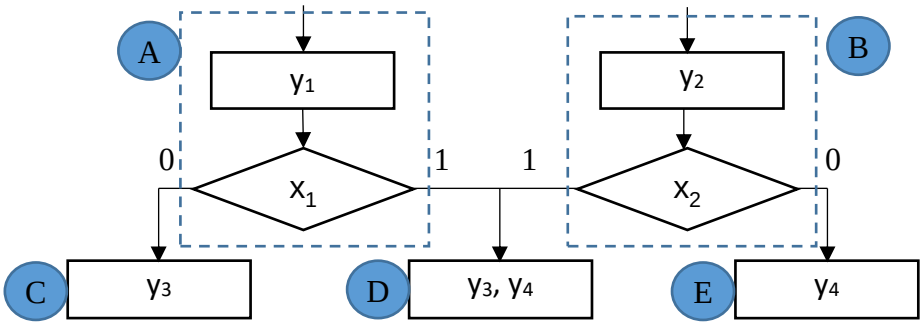
Tačiau galimos ir probleminės situacijos, kurios reikalauja šiek tiek modifikuoti mikroprogramą.

Tarkime, turime mikrokomandoms grafą, kurio fragmentas parodytas 9 paveiksle. Pradėkime nuo perėjimo iš mikrokomandos A. Mikrokomandų adresai C ir D turi būti gretimi, be to, D vienetu didesnis nei C. Tarkime, C bus lygus 6, tuomet D bus 7.

Dabar nagrinėkime perėjimą iš mikrokomandos B. Mikrokomandų adresai D ir E taip pat turi būti gretimi, be to, E vienetu mažesnis nei D. Kadangi D jau priskirtas ir lygus 7, tai E turi būti vienetu mažesnis nei D ir lygus 6. Tačiau anksčiau nustatėme, kad C lygus 6.



8 pav. Mikrokomandų adresavimas priverstinės adresacijos atveju

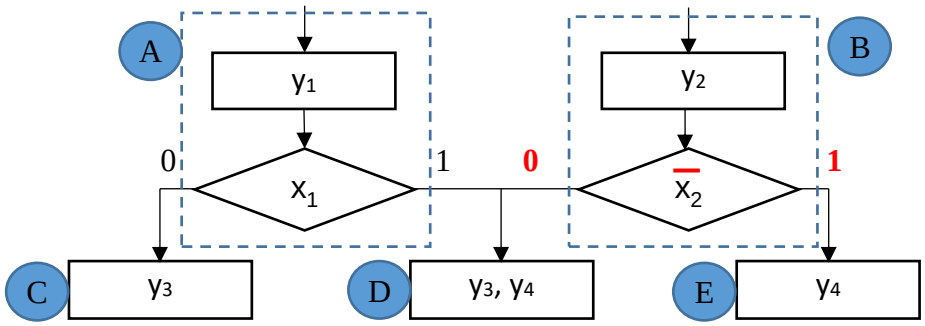


9 pav. Mikroprogramos fragmentas

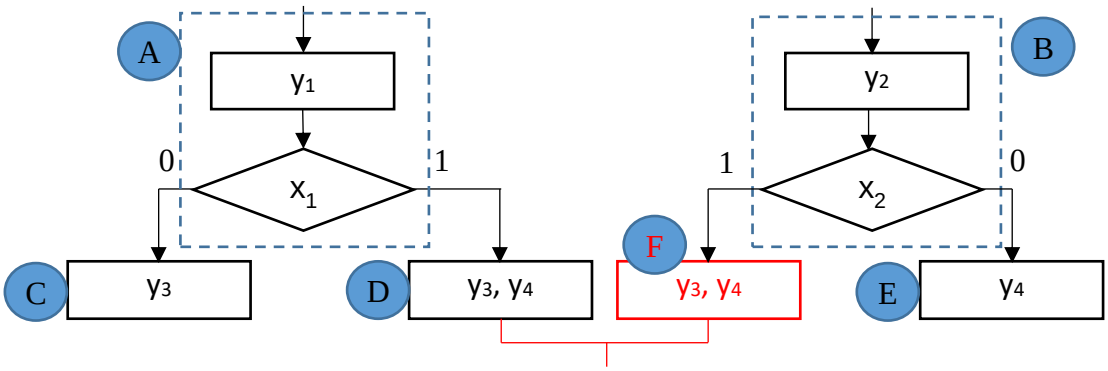
Ši situacija reikalauja rasti kitą sprendimą. Adresavimo konfliktas gali būti išsprestas dviem būdais.

Pakeiskime vienos iš loginių salygų tikrinimą priešingu – vietoj x tikrinkime $\overline{x}$; tuomet pasikeis vietomis ir tos viršūnės išėjimų reikšmės – žr. 10 paveikslą (pakeitimai – raudona spalva). Dabar adresai jau gali būti priskirti taip: C = 6, D = 7, E = 8 – adresavimo konfliktas išspręstas.

Jei toks sąlygos tikrinimo pakeitimas negalimas, dubliuokime mikrokomandą y_3, y_4 (11 pav.); tuomet konfliktą sukėlę perėjimai bus atskirti. Dabar adresai gali būti priskirti taip: C = 6, D = 7, E = 8, F = 9 – adresavimo konfliktas išspręstas.



10 pav. Modifikuotas mikroprogramos fragmentas

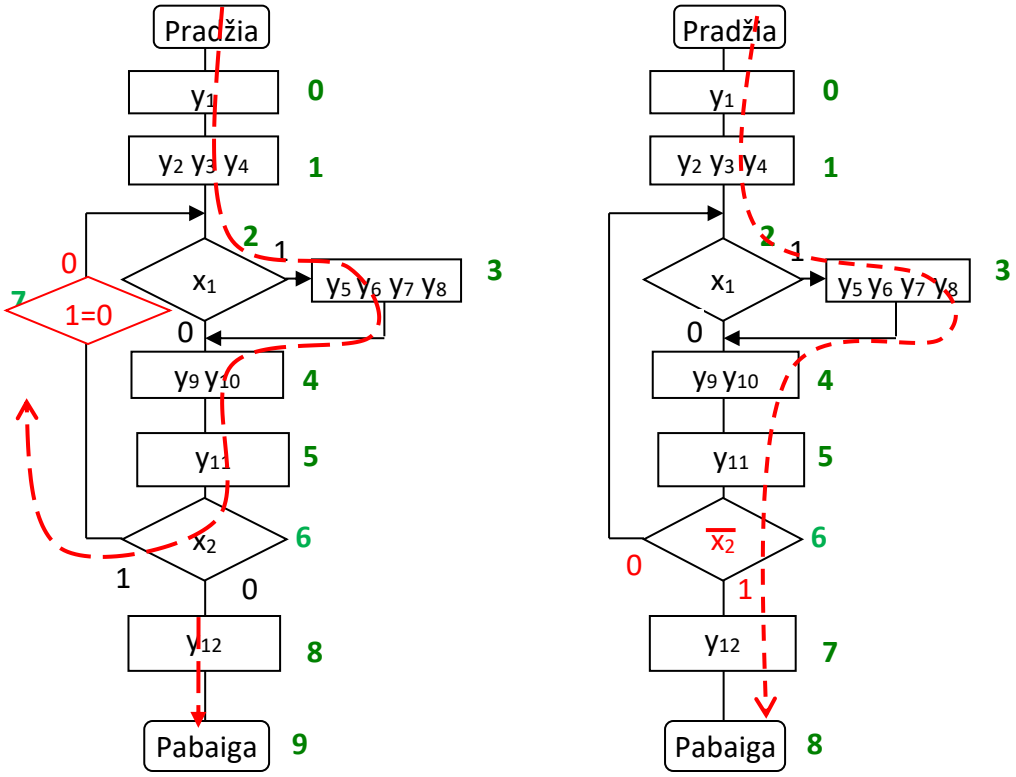


11 pav. Kitu būdu modifikuotas mikroprogramos fragmentas

Dabar pažiūrėkime, kokie adresai bus priskirti mikrokomandoms tame pačiame mikroprogramos grafe, kurį nagrinėjome aukščiau (8 pav.), kai naudojama natūrali adresacija.

Matėme, kad natūralios adresacios atveju viena mikrokomanda atitinka arba operatorinę, arba salygos tikrinimo viršūnę (7 pav.). Tuomet mikrokomandoms adresai bus priskirti tokie, kaip parodyta 12 paveiksle, a. Raudonu punktyru pažymėtas kelias, kuriuo einant mikrokomandų adresai didėja. Matome, kad išsišakojimo atveju kelias pasuka per sąlygos tikrinimo viršūnės išėjimą, pažymėtą 1. Todėl, kai x_2 = 1, kelias pasuka į viršų, link sąlygos x_1 tikrinimo viršūnės. Kadangi po adreso 6 turi sekti adresas 7, o sąlygos x_1 tikrinimo viršūnei jau priskirtas adresas 2, į minėtą kelia būtina pridėti papildomą tokios sąlygos tikrinimo viršūnę, kuri turi vienintelį išėjimą, pažymėtą 0 (tai sąlyga, kuri visada lygi 0, pavyzdžiui, tikrinimas, ar 1=0; faktiškai taip realizuojamas besąlyginis perėjimas reikiamu adresu). Per likusias mikroprogramos grafo viršūnes eina kitas.

Jeigu pakeistume loginės salygos x_2 tikrinimą salygos $\overline{x_2}$ tikrinimu, kelias apims visas mikroprogramos grafo viršūnes (12 pav., b). Tokiu atveju mikroprograma bus viena mikrokomanda trumpesnė. Tiesa, ve visuomet toks loginės salygos tikrinimo pakeitimas galimas.



a) pradinis mikroprogramos grafas b) modofikuotas mikroprogramos grafas

12 pav. Mikrokomandų adresavimas natūralios adresacijos atveju

Natūralios adresacijos atveju besąlyginis perėjimas būtinas ir situacijose, panašiose į parodytas 13 paveiksle, a. Matome, kad raudona punktyrine linija pažymėtas kelias palieka nuošalyje operatorinę viršūnę (mikrokomanda y_3, y_4). Tikrinamos loginės sąlygos keitimas inversine čia nepadės. Adresavimo problemai išspręsti po šios mikrokomandos taip pat reikia įterpti besąlyginį perėjimą (13 pav., b).

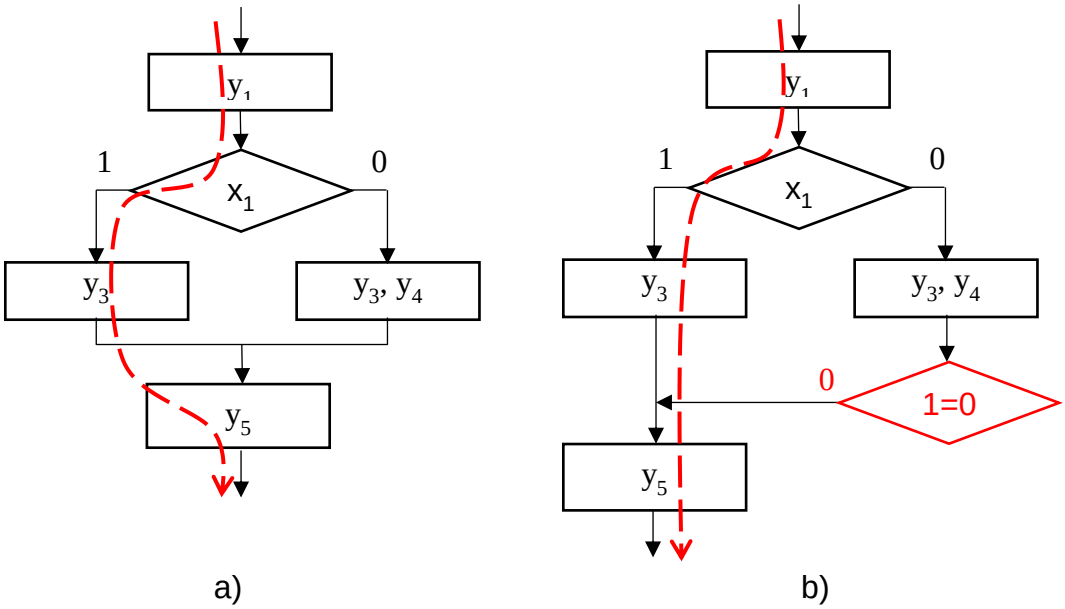
Mikrooperacijų kodavimas

Mikrokomandos laukų $\mathbf{Y_i}$ skaičius priklauso nuo pasirinkto mikrooperacijų kodavimo principo:

- Esant dideliam mikrooperacijų skaičiui jos grupuojamos, į vieną grupę priskiriant mikrooperacijas, kurios kartu nėra niekad vykdomos. Grupei skiriamas atskiras mikrokomandos laukas, kurio plotis priklauso nuo mikrooperacijų skaičiaus grupėje. Toks kodavimo principas dar vadinamas vertikaliuoju kodavimu.
- Esant mažesniam mikrooperacijų skaičiui, kiekvienai mikrooperacijai skiriamas atskiras mikrokomandos bitas. Toks kodavimo principas dar vadinamas horizontaliuoju kodavimu.

Natūralu, kad praktikoje dažniau sutinkamas kombinuotas atvejis, kai dalis mikrooperacijų grupuojamos, o kitoms skiriami atskiri mikrokomandos bitai. Pavyzdžiui, vidutinės integracijos ALU

operacija nurodoma 4 bitų kodu, tad racionalu šį kodą ir įrašyti į atskirą mikrokomandos lauką. Tuo tarpu kelių atminties elementų nustatymui į pradinę būseną (*reset*) tikslinga skirti atskirą mikrokomandos bitą.



a) b)

13 pav. Papildomas perėjimas natūralios adresacijos atveju