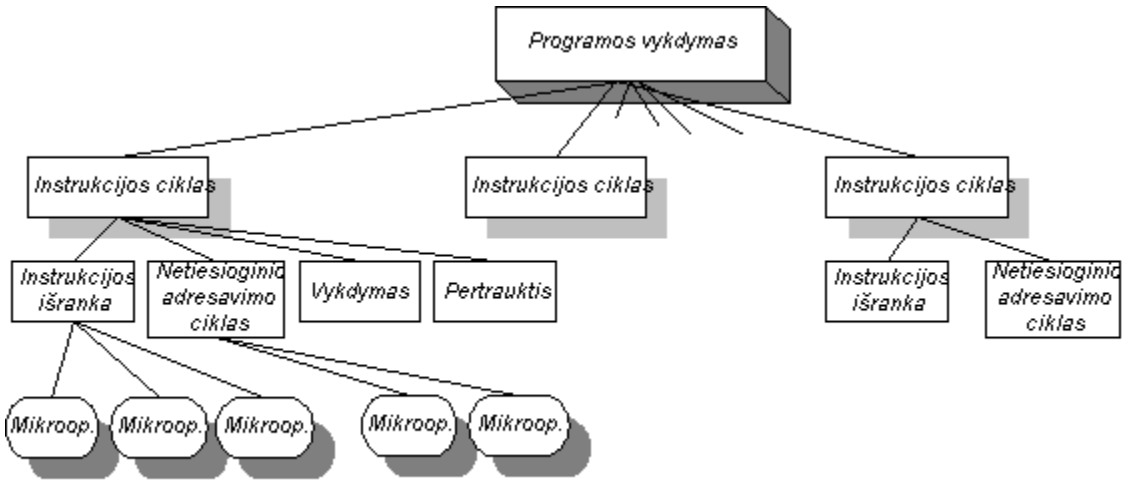
8.VALDYMO ĮRENGINYS

Mašininės instrukcijos jų vykdymo metu nueina ilgą ir sudėtingą kelią, kurį joms nurodo CPĮ. Jeigu mašininių instrukcijų rinkinys yra žinomas, jose aiškus kiekvieno operacijos kodo efektas, suprantamas adresavimo mechanizmas ir žinomi visi vartotojo valdomi registrai, tai galima gana tiksliai nusakyti kokias funkcijas vykdys CPĮ. Tačiau tai bus tik labai paviršutiniškos žinios apie CPĮ darbą. Žinių pagilinimui reikia dar išsiaiškinti CPĮ išorinį interfeisą ir pertraukčių valdymo mechanizmą. Apytikslis įtaisų ir procesų apsprendžiančių CPĮ vykdomas funkcijas sąrašas yra toks:

1. operacijos (operacijų kodai);
2. adresavimo būdai;
3. registrai;
4. įvesties/išvesties modulio interfeisas;
5. atminties modulio interfeisas;
6. pertraukčių apdorojimo struktūra.



8.1. pav. Programos vykdymo sudėtiniai elementai (čia mikroop. – mikrooperacija)

Pirmieji trys sąraše paminėti procesai ir įtaisai turi įtakos sudarant CPĮ instrukcijų rinkinį. Ketvirtojo ir penktojo įtaisų įtaka procesoriaus vykdomoms funkcijoms priklauso nuo kompiuterio sisteminės magistralės. Paskutiniojo šeštojo – dalinai nuo kompiuterio sisteminės magistralės ir dalinai nuo galimybių suteiktų CPĮ operacinei sistemai.

Taigi sąraše išvardinti įtaisai ir procesai apsprendžia ką ir kaip turi daryti CPĮ, t. y. apsprendžia funkcinius reikalavimus CPĮ. Šiame skyriuje analizuosime CPĮ funkcionavimą – aiškinsimės kaip yra valdomi elementai CPĮ, jam vykdant tam tikras funkcijas. Nagrinėsime CPĮ veikimą kontroliuojantį valdymo įrenginį.

8.1. Mikrooperacijos

Pagrindinė kompiuterio funkcija – programų vykdymas. Kompiuteris, vykdydamas programą, turi įvykyti instrukcijų ciklų seką, vykdant kiekviename cikle po vieną mašininę instrukciją. Akivaizdu, kad vykdomų instrukcijų seka gali neatkartoti programoje parašytą instrukcijų seką, nes joje gali būti šakojimosi instrukcijos. Šiame poskyryje išnagrinėsime instrukcijų vykdymą.

Kiekvienas instrukcijos ciklas sudarytas iš smulkesnių etapų tokių, kaip instrukcijos išranka, netiesioginio adresavimo ciklas, vykdymas ir pertrauktis. Iš paminėtų etapų instrukcijoje visuomet būna išrankos ir vykdymo ciklai.

Projektuojant valdymo įrenginį reikia gilintis į instrukcijų vykdymo procesą dar labiau ir instrukcijos ciklą smulkinti į dar mažesnius žingsnius. Septintame skyriuje nagrinėjant konvejeriavimą buvo parodyta, kad instrukcijų dekompozicija įmanoma. Taigi kiekvieną mažą instrukcijos ciklą galima suskaidyti į žingsnių, kuriuose visada naudojami CPĮ registrai, seką. Šiuos žingsnius vadinsime *mikrooperacijomis*. Priešdėlis *mikro*- reiškia, kad žingsniai yra elementarūs ir labai greitai įvykdomi. 8.1 paveiksle parodyta mikrooperacijų vieta programos vykdymo procese.

Mikrooperacijos – tai pačios elementariausios CPĮ vykdomos operacijos. Šiame poskyryje išsiaiškinsime, kaip instrukcijos

ciklą galima aprašyti mikrooperacijų seka.

8.1.1. Išrankos ciklas

Kiekvienas instrukcijos ciklas prasideda nuo išrankos ciklo, kurio metu instrukcija išrenkama iš atminties. Paprastai kompiuteryje yra keturi registrai:

• *Atminties adreso registras (AAR)*. Jis sisteminėje magistralėje prijungtas prie adresų linijų ir nurodo vykdomos skaitymo arba rašymo operacijos adresą atmintyje.
• *Atminties buferio registras (ABR)*. Jis prijungtas sisteminėje magistralėje prie duomenų linijų ir jame yra patalpinami saugojimui į atmintį arba paskutinieji nuskaityti iš atminties duomenys.
• *Programos skaitiklis (PS)*. Jame yra sekančios išrenkamos instrukcijos adresas.
• *Instrukcijų registras (IR)*. Jame patalpinama paskutinė išrinkta instrukcija.

Išsiaiškinkime instrukcijos išrankos ciklo įvykių sekos poveikį CPĮ registrams. Ciklo pradžioje programos skaitiklyje (PS) yra sekančios vykdomos instrukcijos adresas. Pirmu žingsniu šis adresas perkeliamas į atminties adreso registrą (AAR), nes AAR sisteminėje magistralėje prijungtas tik prie adresų linijų. Kitu žingsniu turi būti nurodytu adresu surasta instrukcija. Nurodytas adresas iš AAR išdėstomas adresų magistralėje, valdymo įrenginys valdymo linijomis siunčia skaitymo READ komandą. Tuomet rezultatas atsiranda duomenų magistralėje ir kopijuojamas į atminties buferio registrą (ABR). Po to, vienetu padidinamas programos skaitiklio turinys (PS) ir taip pasiruošiama sekančiai instrukcijai.

Kadangi du pastarieji veiksmai – žodžio iš atminties skaitymas ir PS didinimas vienetu, tarpusavyje nesąveikauja, juos galima vykdyti vienu metu ir sutaupyti laiko. Trečiu žingsniu ABR turinys perkeliamas į instrukcijų registrą (IR). Po šio žingsnio ABR atsilaisvina ir galime jį panaudoti netiesioginio adresavimo cikle.

Taigi elementarų instrukcijos išrankos ciklą, realiai sudaro trys žingsniai ir keturios mikrooperacijos. Kiekvienos mikrooperacijos metu siunčiami duomenys į registrus arba iš jų. Jeigu duomenų siuntimai tarpusavyje nesąveikauja, taupant laiką keli iš jų gali būti vykdomi tame pačiame žingsnyje vienu metu. Simboliškai analizuotoji įvykių seka gali būti aprašyta taip:

t_1: AAR <—PS (pirma mikrooperacija)
t_2: ABR <— Atmintis (antra mikrooperacija)
PS <— PS + 1 (trečia mikrooperacija)
t_3: IR <— ABR (ketvirta mikrooperacija)

Čia antroji ir trečioji mikrooperacijos vykdomos antrame laiko intervale vienu metu. Trečioji mikrooperacija gali būti sujungta ne su antrąja, o su ketvirtąja mikrooperacija be neigiamų pasekmių visam instrukcijos išrankos ciklui:

t_1 : AAR <— PS (pirma mikrooperacija)
t_2 : ABR <— Atmintis (antra mikrooperacija)
t_3 : PS <— PS+1 (trečia mikrooperacija)
IR <— ABR (ketvirta mikrooperacija)

Mikrooperacijas grupuojant būtina laikytis dviejų paprastų taisyklių:

1. Būtina atsižvelgti į įvykių nuoseklumą. Taigi AAR <— PS mikrooperacija turi įvykti prieš ABR <— Atmintis mikrooperaciją, kadangi vykdant pastarąją adresas jau turi būti AAR.

2. Vengti konfliktų. Negalima viename laiko intervale leisti skaityti ir rašyti tame pačiame registre – rezultatas bus nenuspėjamas. Pavyzdžiui, ABR <— Atmintis ir IR <— ABR mikrooperacijos negali vykti tame pačiame laiko intervale.

8.1.2. Netiesioginio adresavimo ciklas

Išrinkus instrukciją, išrenkami išeities operandai. Tęsdami mūsų pradėto pavyzdžio nagrinėjimą, priimkime, kad yra tik vienas adresavimo formatas, kuriame adresas nurodomas tiesiogiai arba netiesioginiai. Jei instrukcijoje adresas nurodytas netiesiogiai, prieš vykdymo ciklą turi būti įterptas netiesioginio adresavimo ciklas. Jam valdyti naudojamos tokios mikrooperacijos:

t_1 : AAR <— IR (adresas)

t_2 : ABR <— Atmintis
t_3 : IR(adresas) <— ABR (adresas)

Instrukcijos adreso laukas perkeliamas į AAR. Toliau pagal šį adresą išrenkamas operando adresas. Galiausiai, adreso laukas IR iš ABR atnaujinamas ir dabar IR jau turi tiesioginį adresą.

Dabar IR yra būsenoje, kurioje netaikomas netiesioginis adresavimas, ir yra paruoštas vykdymo ciklui.

8.1.3. Pertraukties ciklas

Pertraukties ciklo prigimtis skirtingose kompiuterių šeimose labai įvairi. Aptarkime labai paprastą įvykių seką. Šiuo atveju duomenų srauto valdymui reikia tokių mikrooperacijų:

t_1 : ABR <— PS
t_2 : AAR <— Adresas, kur bus saugoma PS reikšmė
PS <— Pertrauktį apdorojančios procedūros pradinis adresas
t_3 : atmintis <— ABR (senoji PS reikšmė)

Pirmuoju žingsniu PS turinys perkeliamas į ABR tam, kad vėliau galima būtų sugrįžti po pertraukties į nutrauktą programos tašką. Tuomet į AAR įkeliamas vietos kur bus saugoma PS reikšmė adresas, o į PS įkeliamas procedūros apdorojančios pertrauktį pradžios adresas. Šie du veiksmai gali būti atskiromis mikrooperacijomis. Bet kuriuo atveju paskutiniu žingsniu saugojimui į atmintį perkeliamas ABR turinys, kuriame buvo įsiminta senoji PS reikšmė. Dabar CPĮ pasirengęs vykdyti kitą instrukcijos subciklą.

8.1.4. Vykdymo ciklas

Išrankos, netiesioginio adresavimo ir pertraukties ciklai yra paprasti ir lengvai nusakomi. Kiekvieną iš jų sudaro nedaug, fiksuoto dydžio mikrooperacijų ir kiekvienu atveju kai kurios mikrooperacijos kartojasi.

Vykdymo cikle viskas yra kitaip. Procesorius instrukcijų rinkinį gali sudaryti *N* operacijos kodų ir galima gauti *N* skirtingų mikrooperacijų sekų. Išnagrinėkime kelis hipotetinius pavyzdžius.

Pirmiausia, išnagrinėkime sudėties ADD instrukciją:

ADD R1, X

Pagal šią instrukciją atminties ląstelės X turinys sumuojamas su registro R1 turiniu. Sudėčiai atlikti reikalinga tokia mikrooperacijų seka:

t_1 : AAR<— IR (adresas)
t_2 : ABR <— Atmintis
t_3 : R1 <— R1+ ABR

Sudėtis pradedama nuo IR registro, kuriame saugoma ADD instrukcija. Pirmuoju žingsniu instrukcijos adreso dalis įkeliama į AAR. Tuomet skaitoma nurodytu adresu atminties ląstelė. Po to ALĮ sudeda R1 ir ABR turinius ir sudėties vykdymas baigiamas. Išnagrinėtas sudėties pavyzdys yra labai supaprastintas.

Išnagrinėkime šiek tiek sudėtingesnį pavyzdį – bendrojo pobūdžio instrukciją vadinamą *padidinti ir praleisti jeigu nulis* – Increment and Skip if Zero – ISZ X. Ši instrukcija atminties ląstelės turinį X padidina vienetu ir, jei dėl to gaunamas 0, sekančią instrukciją praleidžia. Šiai instrukcijai atlikti reikalinga tokia mikrooperacijų seka:

t_1 : AAR <— IR (X ląstelės adresas)
t_2 : ABR <— Atmintis
t_3 : ABR <— ABR + 1
t_4 : Atmintis <— ABR
IF (ABR = 0) then (PS = PS + 1)

Šioje mikrooperacijų sekoje yra naujovė – sąlygos tikrinimo veiksmas. PS didinamas vienetu, jei ABR turinys lygus 0 (ABR

= 0). Čia tikrinimas ir veiksmas vykdomi kaip viena mikrooperacija. Būtina atkreipti dėmesį į tai, kad ši mikrooperacija vykdoma tuo pačiu metu, kai atnaujinta ABR reikšmė rašoma atgal į atmintį (t_4 laiko intervale).

8.1.5. Instrukcijos ciklas

Mes jau žinome, kad kiekviena instrukcijos ciklo fazė gali būti dekomponuota į mikrooperacijų seką. Išnagrinėtuose pavyzdžiuose išrankos, netiesioginio adresavimo ir pertraukties ciklus vykdydavo tam tikra mikrooperacijų seka. Tuo tarpu vykdymo cikluose kiekvienos operacijos kodą atitindavo unikali mikrooperacijų seka.

8.2. CPĮ valdymas

8.1 poskyryje aiškinantis CPĮ veikimą jo vykdomos funkcijos buvo dekomponuotos iki elementarių operacijų, vadinamų mikrooperacijomis. Tai mums leido išsiaiškinti valdymo įrenginio veikimo esmę. CPĮ vykdomas funkcijas susmulkinant iki fundamentaliausio lygmens galima tiksliai nustatyti valdymo įrenginys veiksmus vienoje ar kitoje situacijoje. Taip galima suformuluoti valdymo įrenginiui *funkcinius reikalavimus*, t. y. išvardinti funkcijas kurias turi vykdyti valdymo įrenginys. Funkcinių reikalavimų nustatymas yra valdymo įrenginio projektavimo ir įgyvendinimo darbų pagrindas.

Prieš kiekvieno valdymo įrenginio apibūdinimą būtina remiantis apriorinė informacija atlikti tokią į tris etapus skirstytą jo analizę:

1. Nustatyti pagrindinius CPĮ elementus.
2. Apibūdinti CPĮ vykdomas mikrooperacijas.
3. Nurodyti valdymo įrenginio atliekamas funkcijas, vykdant mikrooperacijas.

Pirmąjį ir antrąjį analizės etapus esame jau atlikę. Dabar padarykime tik išvadas.

CPĮ pagrindiniai funkciniai elementai:

• ALĮ;
• registrai;
• vidiniai duomenų keliai;
• valdymo įrenginys.

Programos vykdymas susideda iš atskirų operacijų vykdymo, įtraukiant anksčiau išvardintus CPĮ elementus. Kiekviena operacija sudaroma iš mikrooperacijų sekų. Apžvelgus 8.1 poskyrį išplaukia, kad visas mikrooperacijas galima suskirstyti į tokias apibendrintas grupes:

• keitimasis duomenimis tarp registrų;
• duomenų siuntimas iš registrų į išorinį interfeisą, t. y. į sisteminę magistralę;
• duomenų siuntimas iš išorinio interfeiso į registrus;
• aritmetinių ir loginių operacijų vykdymas, taikant registrus įvesčiai ir išvesčiai.

Šiame sąraše yra visos mikrooperacijos būtinos instrukcijos ciklui arba bet kurios instrukcijos iš CPĮ instrukcijų rinkinio vykdymui.

Tikimės, kad dabar pasidarė aiškesnė valdymo įrenginio paskirtis ir veikimas. Valdymo įrenginys atlieka dvi pagrindines užduotis:

• *Nuoseklumo tvarkymas.* Valdymo įrenginys verčia CPĮ žingsniais nuo vykdomos programos priklausančia tvarka vykdyti mikrooperacijas.

• *Vykdymas.* Valdymo įrenginys sąlygoja ir valdo visų mikrooperacijų vykdymą.

Valdymo įrenginys visas savo funkcijas vykdo panaudodamas valdymo signalus.

8.2.1. Valdymo signalai

Buvo nustatyti elementai sudarantys CPĮ (ALĮ, registrai, duomenų keliai) ir jo vykdomos mikrooperacijos. Valdymo įrenginys savo funkcijoms atlikti turi turėti įėjimus, pro kuriuos jis nustato kompiuterizuotos sistemos būklę ir išėjimus sistemos veikimui valdyti. Tokia yra išorinė valdymo įrenginio specifikacija. Viduje valdymo įrenginys turi turėti tam tikrą logiką užtikrinančią minėtas nuoseklumo tvarkymo ir vykdymo funkcijas. Vidinę valdymo įrenginio sandara nagrinėsime 8.3 poskyryje. Likusioje šio skyriaus dalyje aptarsime sąveiką tarp valdymo įrenginio ir kitų CPĮ elementų.

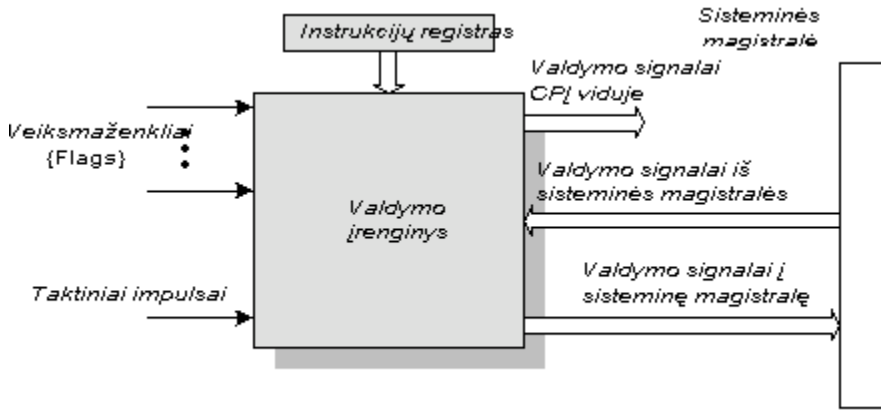
8.2 pav. pateiktas apibendrintas valdymo įrenginio modelis, kuriame parodyti visi jo įėjimai ir išėjimai. Valdymo įrenginyje yra šie įėjimai:

• *Taktavimas* {clock}. Įgalina valdymo įrenginį „laikyti ritmą“. Kiekvieno taktinio impulso metu valdymo įrenginys kontroliuoja vienos mikroopreacijos (arba kelių vienu metu vykdomu mikrooperacijų) vykdymą.

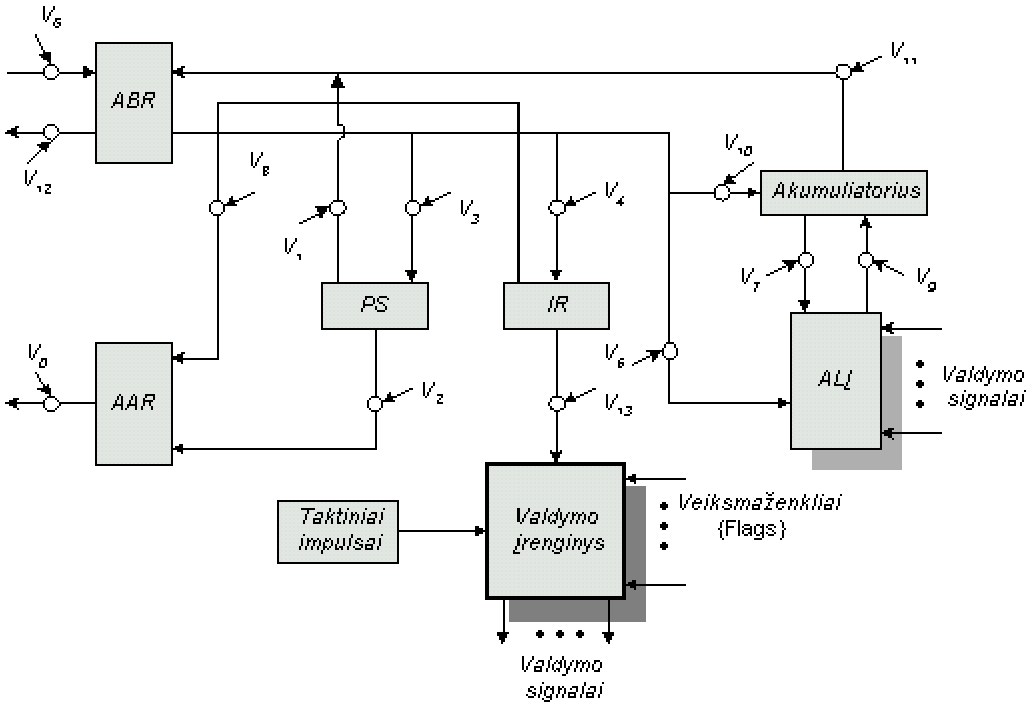
• *Instrukcijų registras.*
• *Veiksmaženkliai* {flags}. Pagal šiuos signalus valdymo įrenginys nustato CPĮ būklę ir ALĮ įvykdytos operacijos rezultatą.
• *Valdymo signalai iš valdymo magistralės.* Valdymo magistralė yra sisteminės magistralės dalis generuojanti signalus į valdymo įrenginį, pvz., pertraukties arba patvirtinimo signalus.

Valdymo įrenginio išėjimai yra šie:

• *Vidiniai CPĮ valdymo signalai.* Šie signalai būna dviejų tipų: valdantys keitimąsi duomenimis tarp registrų ir suaktyvinantys specifines ALĮ funkcijas.
• *Valdymo signalai į valdymo magistralę.* Jie būna taip pat dviejų tipų: valdymo signalai į atmintį ir į įvesties/išvesties modulius.



8.2 pav. Valdymo įrenginio modelis



8.3 pav. Duomenų keliai ir valdymo signalai

Naujas elementas įvestas 8.2 paveiksle – yra valdymo signalai. Yra trijų tipų valdymo signalai: suaktyvinantys ALĮ funkcijas, suaktyvinantys tam tikrus duomenų kelius ir signalai esantys išorinėje, CPĮ atžvilgiu, sisteminėje magistralėje. Visi šitie signalai iš esmės yra dvejetainiai signalai, siunčiami į tam tikrų loginių schemų (raktų) įėjimus.

8.2.2. Valdymo signalų pavyzdžiai

Valdymo įrenginio funkcionavimo analizei pasinaudokime pavyzdžiu pritaikytu iš [4] ir pateiktu 8.3 paveiksle. Šiame paveiksle pavaizduotas pavienis CPĮ su vienu akumuliatoriumi ir duomenų keliai tarp atskirų elementų. Valdymo signalų keliai iš valdymo įrenginio čia neparodyti. Apskritimais paveiksle pažymėti valdymo signalų kelių paskirties taškai, o patys signalai parodyti simboliais V_i. Valdymo įrenginys signalus gauna iš taktinio generatoriaus {clock}, instrukcijų registro ir veiksmaženklių {flags}. Kiekvieno taktinio impulso metu valdymo įrenginys nuskaito savo įėjimus ir generuoja valdymo signalų visumą.

Valdymo signalai siunčiami trimis pagrindinėmis kryptimis:

• *Duomenų keliai.* Valdymo įrenginys valdo vidinius duomenų srautus. Pvz., išrenkant instrukciją, atminties buferinio registro turinys siunčiamas į instrukcijų registrą. Visi duomenų keliai valdomi raktais, 8.5 paveiksle parodytais apskritimais. Valdymo signalai iš valdymo įrenginio trumpam atidaro šiuos raktus ir leidžia tam tikra kryptimi siųsti duomenis.

• *ALĮ.* Valdymo įrenginys valdo ALĮ darbą siųsdamas jam valdymo signalus. Šie signalai suaktyvina įvairius loginius elementus ALĮ viduje.

• *Sisteminė magistralė.* Valdymo įrenginys siunčia valdymo signalus išorėn į sisteminės magistralės valdymo linijas (pvz., READ – atminties skaitymo linija).

Valdymo įrenginys turi visada „žinoti“, kuri instrukcijos ciklo stadija duotuoju momentu vykdoma. Tai „žinodamas“ ir nuskaitęs savo įėjimo signalus, valdymo įrenginys generuoja valdymo signalų sekas, kurios valdo mikrooperacijas. Įvykiams sekose sinchronizuoti naudojami taktavimo impulsai, kuriuose numatyti specialūs laiko intervalai būtini loginiams lygiams tarp įvykių nusistovėti.

Štai kaip galima aprašyti iki minimumo supaprastintą valdymo įrenginio esmę. Valdymo įrenginys yra viso kompiuterio varomoji jėga. Savo funkcijas jis atlieka "žinodamas" tik instrukcijas, kurios turi būti vykdomos ir aritmetinių bei loginių operacijų rezultatų esmę (pvz., teigiamas, neigiamas, perpildymas ir t. t.). Jam visai nerūpi kaip atrodo duomenys, kuriuos jis apdoroja ir ar teisingi apdorojimo rezultatai. Valdymo įrenginys sugeba viską valdyti panaudodamas tik kelis valdymo signalus CPĮ viduje ir kelis valdymo signalus sisteminėje magistralėje.

8. 3. Valdymo įrenginio schemotechninis įgyvendinimas

Aptarėme valdymo įrenginio įvesties ir išvesties signalus bei jo vykdomas funkcijas. Dabar atėjo eilė išnagrinėti valdymo įrenginio įgyvendinimo būdus. Jų yra labai daug, tačiau dauguma jų galima suskirstyti į dvi grupes:

• Schemotechninį įgyvendinimą;
• Mikroprograminį įgyvendinimą.

Schemotechninio įgyvendinimo atveju, valdymo įrenginys įsivaizduojamas kaip kombinatorinė schema. Joje įvesties loginiai signalai transformuojami į atitinkamus išvesties loginius signalus, kurie ir tampa valdymo signalais. Šiame poskyryje nagrinėsime tik schemotechninį įgyvendinimo būdą.

Valdymo įrenginio įvesties signalai

8.2 paveiksle pavaizduotas valdymo įrenginys ir jo siunčiami bei gaunami signalai. Čia pagrindiniai įvesties valdymo signalai yra signalai iš instrukcijų registro, taktiniai impulsai, veiksmaženkliai {flags} ir signalai iš valdymo magistralės. Veiksmaženkliuose ir valdymo magistralės signaluose kiekvienas atskiras bitas turi tam tikrą reikšmę (pvz., perteklius). Kiti du įėjimo signalai – taktiniai impulsai ir signalai iš instrukcijų registro – valdymo įrenginyje tiesiogiai nenaudojami.

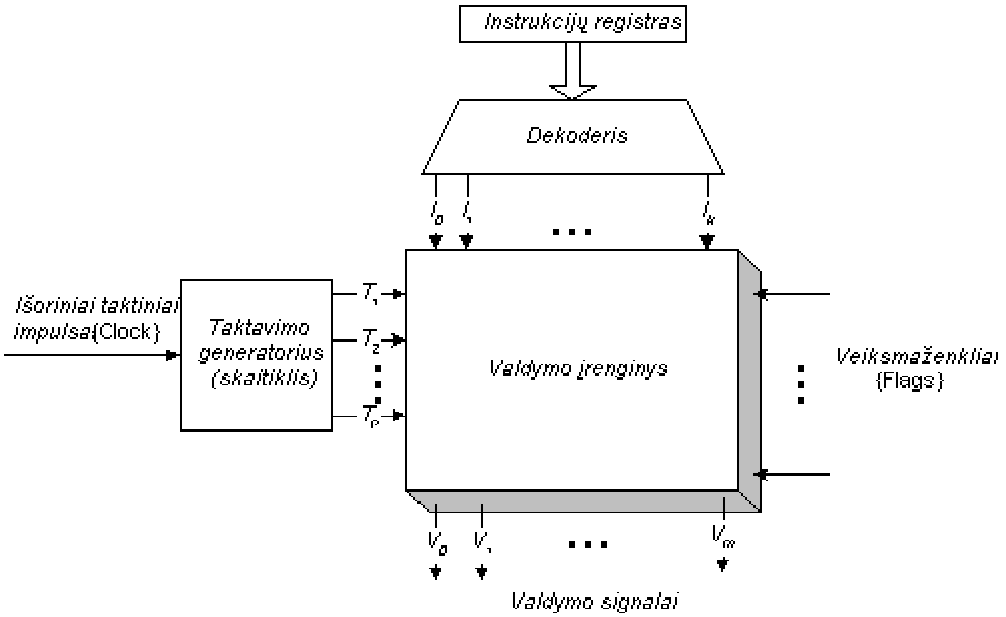
Pirmiausia išsiaiškinkime instrukcijų registrą. Valdymo įrenginys priima operacijos kodą ir pagal jį vykdo tam tikrus veiksnius – tiksliau savo išvestyje generuoja tam tikrą valdymo signalų kombinaciją. Valdymo įrenginio loginei schemai supaprastinti kiekvieną operacijos kodą reikia pakeisti unikalia jo įėjimo signalo logine kombinacija. Šią funkciją vykdo dekoderis. Jis priima siunčiamą kodą ir tik viename tam tikrame savo išėjime generuoja signalą. Apskritai, dekoderis gali priimti *n* dvejetainių signalų ir sugeneruoti 2^n skirtingų dvejetainių išėjimo signalų. Kiekviena iš 2^n skirtingų įvesties kombinacijų suaktyvina vienintelį unikalų išvesties signalą (8.1 lentelė). Realiuose valdymo įrenginiuose dekoderis būna daug sudėtingesnis nei tas, kurį čia aptarėme. Realūs dekoderiai apdoroja skirtingo ilgio operacijos kodus.

8.1 lentelė. Keturių įėjimų dekoderis

I_1	I_2	I_3	I_4	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}	O_{16}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Taktiniai impulsai atrodo kaip periodinė impulsų seka. Pagal taktinius impulsus kontroliuojama mikrooperacijų trukmė. Norint garantuoti signalų sklidimą įvairaus ilgio duomenų keliais ir CPĮ grandinėmis, taktinių impulsų periodas turi būti pakankamai ilgas. Anksčiau buvo parodyta, kad valdymo įrenginys instrukcijos ciklo metu tam tikrais laiko momentais generuoja skirtingus valdymo signalus. Taigi valdymo įrenginio įėjime turėtų būti skaitiklis, generuojantis valdymo įrenginiui T_1, T_2 ir t. t. valdymo signalus. Instrukcijos ciklo pabaigoje valdymo įrenginys skaitiklį turėtų sugrąžinti į pradinę – T_1 būseną.

Atsižvelgus būtinybę turėti dekoderį ir priiminėti taktinius impulsus, valdymo įrenginį galima pavaizduoti taip, kaip tai padaryta 8.4 paveiksle.



8.4 pav. Valdymo įrenginio struktūrinė schema su įvesties signalų dekodavimu

Literatūra papildomam skaitymui apie valdymo įrenginį

Valdymo įrenginio funkcijos nagrinėjamos aibėje knygų apie kompiuterių architektūrą. Tarp jų:

1. **J.Hennessy, J.L.Hennessy, D.Goldenberg, D.A.Patterson.** Computer Architecture : A Quantitative Approach **/** Morgan Kaufmann Publishers; ISBN: 1558603298. 1996, 760 p.
2. **M.M.Mano, C.R.Kime.** Logic and Computer Design Fundamentals **/** Prentice Hall; ISBN: 0130124680. 1999, 652 p.
3. **S.A.Ward, R.H.Halstead.** Computation Structures (MIT Electrical Engineering and Computer Science) **/** MIT Press; ISBN: 0262231395. 1989, 811 p.
4. **W.Stallings.** Computer Organization and Architecture: Designing for Performance **/** Prentice Hall; ISBN: 0130812943. 1999, 768 p.
5. **M.J.Murdocca, V.P.Heuring.** Principles of Computer Architecture **/** Prentice Hall; ISBN: 0201436647. 2000, 553 p.