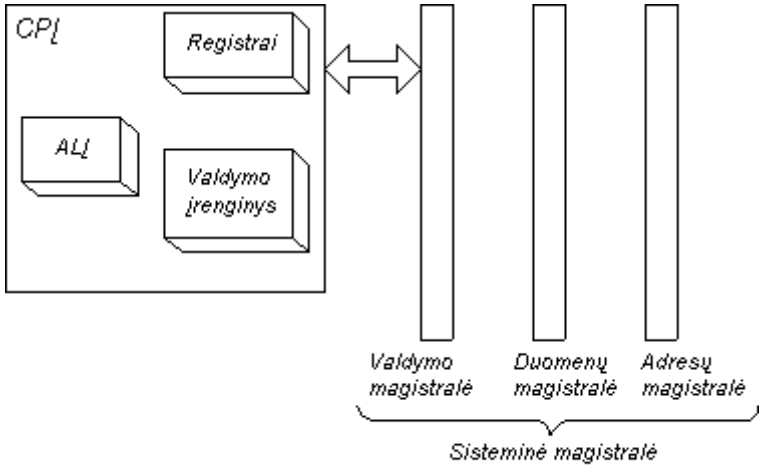
7. CENTRINIS PROCESORINIS ĮRENGINYS

Šiame skyriuje labai apibendrintai nagrinėsime kompiuterio širdį – centrinį procesorinį įrenginį. Pradžioje išsiaiškinsime procesoriaus sandarą. Vėliau išanalizuosime procesoriaus vidinės atminties registrus. Pakartosime instrukcijos ciklą ir skyrių užbaigsime instrukcijų srauto apdorojimo metodo analize.

7.1. Procesoriaus sandara

7.1 pav. parodyta supaprastinta CPĮ sandaros schema ir jo ryšiai per sisteminę magistralę su likusiaja kompiuterizuotos sistemos dalimi. Pagrindinės CPĮ komponentės yra *aritmetikos ir logikos įrenginys* (ALĮ) bei *valdymo įrenginys* (VĮ). ALĮ vykdo visus būtinus apskaičiavimus ir apdoroja duomenis. Valdymo įrenginys kontroliuoja duomenų ir instrukcijų judėjimą į CPĮ ir iš jo, taip pat valdo ALĮ veikimą. 7.1 pav. parodyta maža vidinė atmintie, kurią sudaro saugojimo ląstelės vadinamos *registrais*.



7.1 pav. CPĮ ir kompiuterio sisteminė magistralė

7.2. Registrų sandara

Trečiame skyriuje buvo parodyta, kad kompiuterizuota sistema remiasi atminties hierarchija. Aukštesniuosiuose hierarchijos lygmenyse išdėstyta pati sparčiausia, mažiausios talpos ir brangiausia vieno bito atžvilgiu atmintis. CPĮ viduje yra registrų rinkinys, kuris funkcionuoja kaip atminties hierarchijos lygmuo, esantis aukščiau už pagrindinę ir spartinančiąją atmintis. CPĮ registrai skirti dviem funkcijoms vykdyti:

• *Vartotojo pasiekiami registrai.* Šie registrai leidžia programuotojui, taikant asemblerio programavimo kalbą arba tiesiog per mašininius kodus, sumažinti kreipčių į pagrindinę atmintį skaičių ir optimizuoti šių registrų vartojimą.

• *Valdymo ir būklės registrai.* Jie naudojami valdymo įrenginio CPĮ darbui kontroliuoti, o taip pat valdyti operacinių sistemų programų (utilitų) taikomųjų programų vykdymą.

7.2.1. Vartotojo pasiekiami registrai

Vartotojo pasiekiami registrai yra vieni iš tų, į kuriuos galima kreiptis taikant mašinines komandas (t. y. tas, kurias vykdo CPĮ). Faktiškai visuose dabartiniuose CPĮ yra ne tik pavienis akumuliatorius, bet ir keli vartotojo pasiekiami registrai. Pastaruosius galima sugrupuoti į tokias kategorijas:

• *bendrosios paskirties;*
• *duomenų;*
• *adresų* ir
• *sąlygų kodų.*

Bendrosios paskirties registrai programuotojo gali būti pritaikyti įvairioms funkcijoms.

Duomenų registrai gali būti taikomi tik duomenims saugoti ir niekada netaikomi apskaičiuojant operandų adresus. *Adresų registrai* patys gali būti realizuojami iš bendrosios paskirties registrų, o gali būti skirti specialiems adresavimo metodams.

Paskutinė registrų, į kuriuos gali kreiptis vartotojas, grupė saugo *sąlygų kodus*. Juos taip pat vadina *veiksmaženkliais* {flags}. Sąlygų kodus sudaro tam tikri bitai, kuriais gali operuoti CPĮ techninė įranga, priklausomai nuo vykdomos operacijos rezultato. Pavyzdžiui, paprastos aritmetinės operacijos rezultatas gali būti teigiamas, neigiamas, nulinis arba perteklinis. Tokiu būdu, operacijos rezultatas saugomas atmintyje, arba registruose, tačiau kartu su juo nustatoma tam tikra sąlygos kodo reikšmė, kuri tikrinama šakojimosi operacijoje.

Sąlygų kodo bitai apjungiami į vieną arba kelis registrus. Paprastai jie sudaro valdymo registro dalį. Apskritai, šiuos mašininės instrukcijos bitus galima tik skaityti, programuotojas jų keisti negali.

7.2.2. Valdymo ir būklės registrai

CPĮ darbui valdyti naudojama aibė įvairių registrų. Dauguma jų vartotojams nėra pasiekiami. Kai kuriuos iš jų galima pasiekti vykdant mašinines instrukcijas valdymo arba operacinės sistemos režimuose.

Natūralu, kad skirtinguose procesoriuose skiriasi registrų sandara. Taip pat naudojami skirtingi terminai. Šiame skyrelyje sudarysime beveik visus procesorius apimantį pilną registrų sąrašą su trumpais jų aprašais.

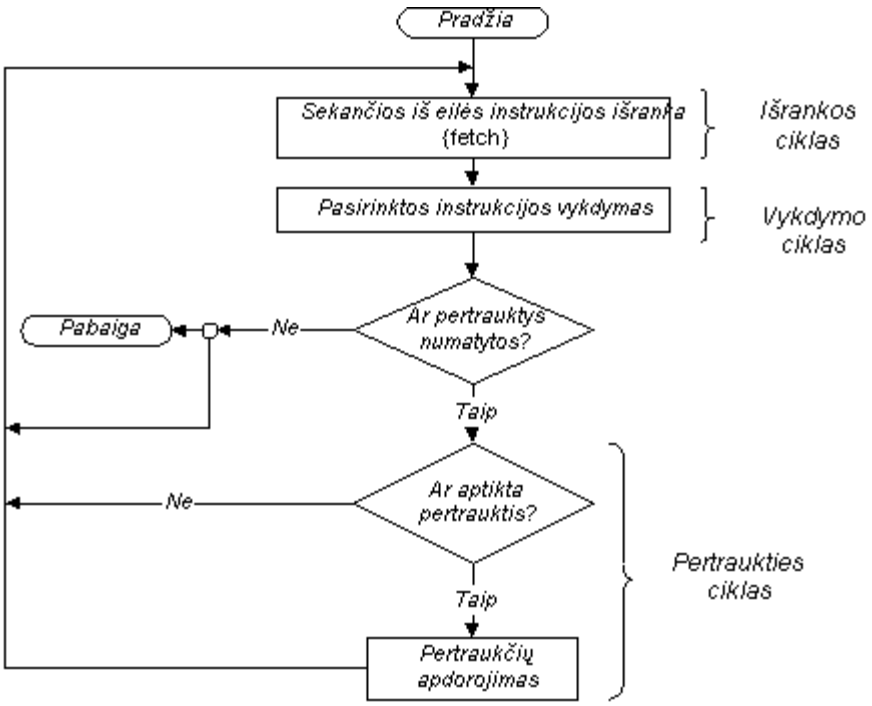
Instrukcijų vykdymui svarbiausi yra šie keturi registrai:

• *Programos skaitiklis (PS).* Saugo sekančios išrenkamos instrukcijos adresą.
• *Instrukcijos registras (IR).* Saugo naujausiąją išrinktą instrukciją.
• *Atminties adreso registras (AAR).* Saugo atminties ląstelės, iš kurios ką tik buvo perskaityta informacija arba į kurią vyks duomenų rašymas, adresą.
• *Atminties buferio registras (ABR).* Saugomas duomenų žodis, kuris bus įrašytas į atmintį arba iš atminties ką tik nuskaitytas.

7.3. Instrukcijos ciklas

2.2 poskyryje mes jau nagrinėjome CPĮ instrukcijos ciklą. Čia pateiktas 7.2 paveikslas atkartoja ten esantį 2.9 paveikslą. Prisiminkime tokius mums labai svarbius instrukcijos ciklą sudarančius subciklus:

• *Išranka* {fetch}. Iš atminties į CPĮ nuskaitoma sekanti instrukcija.
• *Vykdymas* {execute}. Interpretuojamas operacijos kodas ir vykdoma jį atitinkanti operacija.
• *Pertrauktis* {interrupt}. Atradus programos vykdymo metu pertrauktį, kompiuterizuotoje sistemoje su numatytoms pertrauktimis išsaugoma vykdomojo proceso būklės informacija ir apdorojama pertrauktis.



7.2 pav. Instrukcijos su pertrauktimis vykdymo algoritmas

Prisiminę šiuos subciklus, detaliau išnagrinėkime visą instrukcijos ciklą. Iš pradžių įveskime papildomą subciklą. Jis dar vadinamas netiesioginio adresavimo ciklu.

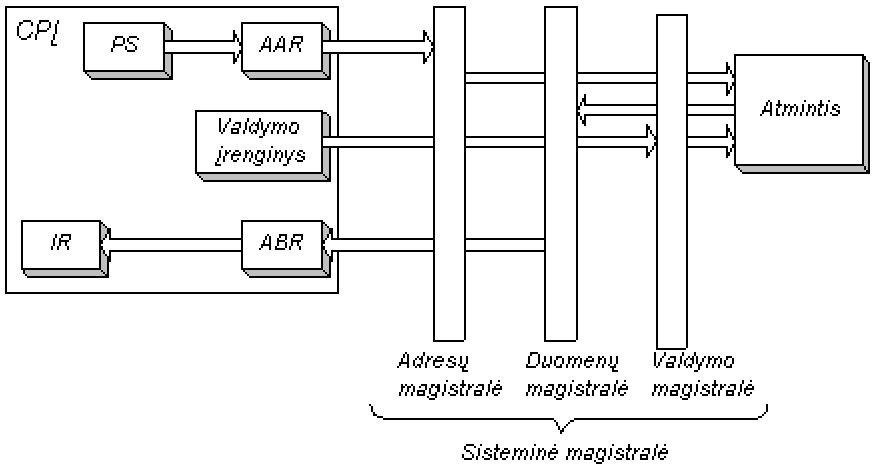
7.3.1. Netiesioginio adresavimo ciklas

Instrukcijos vykdymo metu tenka išsikviesti iš atminties vieną ar kelis operandus. Kiekvieno operando nuskaitymui reikia atskiros kreipties į atmintį. Be to, esant jo netiesioginiam adresavimui reikalingos dar papildomos kreiptys į atmintį.

Netiesioginio adresavimo operandą galima interpretuoti kaip papildomą instrukcijos subciklą. Todėl po instrukcijos išrankos tikrinama ar joje nėra netiesioginio adresavimo. Jeigu aptinkamas netiesioginis adresavimas, operandas nuskaitomas taikant netiesioginį adresavimą.

7.3.2. Duomenų srautas

Tiksli įvykių seka instrukcijos ciklo vykdymo metu priklauso nuo CPĮ sandaros. Tačiau galima labai apibendrintais bruožais apibūdinti ten vykstančius procesus. Tarkime kad turime CPĮ, kuris turi atminties adreso registrą (AAR), atminties buferinį registrą (ABR), programos skaitiklį (PS) ir instrukcijos registrą (IR).



7.3 pav. Duomenų srautai išrankos cikle

Instrukcija *ciklo išrankos* metu skaitoma iš atminties. 7.3 pav. parodyti šio ciklo metu sudaromi duomenų srautai. PS registre saugomas sekančios išrenkamos instrukcijos adresas. Šis adresas siunčiamas į AAR ir patalpinamas adresų magistralėje. Valdymo įrenginys organizuoja skaitymą iš atminties. Nuskaitytas rezultatas patalpinamas duomenų magistralėje, kopijuojamas į ABR ir toliau siunčiamas į IR. Tuo tarpu PS turinys didinamas vienetu ir taip gaunamas sekančios instrukcijos adresas.

Pasibaigus išrankos ciklui, valdymo įrenginys patikrina IR turinį ir nustato operando specifikatoriaus adresavimo būdą. Jeigu aptinkamas netiesioginis specifikatoriaus adresavimas, vykdomas *netiesioginis ciklas*.

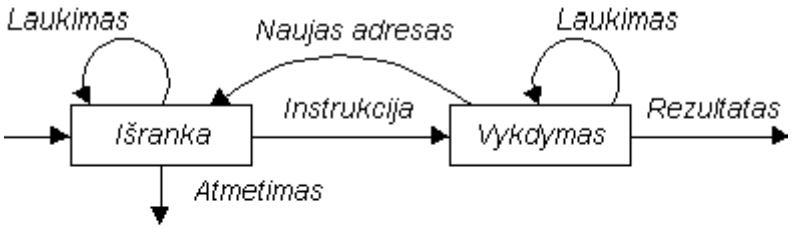
Išrankos ir netiesioginiai ciklai nesudėtingi ir lengvai iš anksto nusakomi. Tuo tarpu *instrukcijos ciklas* priklausomai nuo to, kokiame kompiuteryje jis vykdomas, gali būti įvairios formos. Šį ciklą gali sudaryti keitimosi duomenimis tarp registrų operacijos, skaitymo/rašymo atmintyje arba įvesties/išvesties įrenginiuose operacijos bei kreiptys į ALĮ.

Pertraukties ciklas yra toks pats paprastas ir lengvai iš anksto nusakomas, kaip išrankos bei netiesioginio adresavimo ciklai. Pirmiausia išsaugomas PS esantis vykdomos instrukcijos turinys. Tai daroma tam, kad po pertraukties apdorojimo CPĮ galėtų sugrįžti prie nutrauktos instrukcijos vykdymo. Įrašant PS turinį į atmintį, iš pradžių jis siunčiamas į ABR. PS turinio įrašui atmintyje užrezervuojama speciali sritis, kurios adresą valdymo įrenginys įkrauna į AAR. Tokia sritimi gali būti, pvz., „steko“ rodyklė. Toliau į PS įkeliamas pertraukties pirmos instrukcijos adresas ir t. t.

7.4. Instrukcijų konvejeriavimas

Naujųjų technologijų dėka kompiuterizuotos sistemos tobulėja, didėja jų našumas. Tai leidžia padaryti atsirandantys nauji spartesni schemotechniniai sprendimai. Kompiuterizuotų sistemų našumas didinamas ir tobulinant CPĮ sandarą. Pavyzdžiui, vienintelį akumuliatorių pakeitus registrų rinkiniu, įvedus spartinančiąją atmintį. Bendresnio pobūdžio CPĮ sandaros tobulinimas yra instrukcijų konvejeriavimas.

a) Supaprastinta schema



b) Išsamesnė schema

7.4 pav. Dviejų etapų instrukcijų konvejeriavimas

7.4.1. Konvejeriavimo esmė

Instrukcijų konvejeriavimas kažkuo panašus į surinkimo linijos taikymą produkcijos gamyboje. Surinkimo linijos privalumas yra tame, kad surinkimo procesas surinkimo linijoje išskaidomas į etapus ir tame, kad tam tikros gaminio surinkimo procedūros gali būti vykdomos vienu metu. Toks procesas dar vadinamas *konvejeriu* arba *vamzdynu* {pipeline}, kadangi konvejeryje nauja operacija pradedama viename gale nesulaukus kol kita operacija kitame gale bus užbaigta.

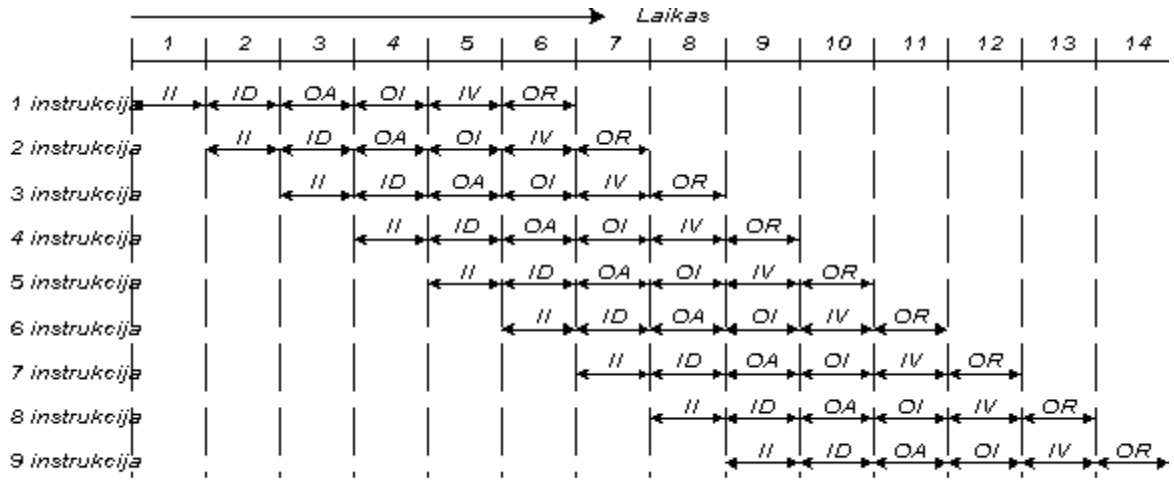
Taikant tokią koncepciją instrukcijos vykdymui, būtina suvokti tą faktą, kad instrukcijoje yra tam tikras etapų skaičius. Natūralu, kad konvejeriavimui organizuoti reikia specialių priemonių.

Paprasčiausios instrukcijos vykdymą galima padalinti į du etapus: instrukcijos išranką ir instrukcijos įvykdymą. Vykdant kiekvieną instrukciją, tam tikrą laiko tarpą į pagrindinę atmintį nesikreipiama. Šį laiko tarpą galima panaudoti lygiagrečiai su jau vykdoma instrukcija kitos instrukcijos išrankai.

Akivaizdu, kad konvejeriavimo procesas pagreitina instrukcijų vykdymą. Jeigu išrankos ir vykdymo etapai būtų vienodos trukmės, instrukcijos ciklo vykdymo laikas būtų padalinamas į dvi lygias dalis.

Išnagrinėkime instrukcijos vykdymo proceso dekompanavimą.

• *Instrukcijos išranka (II).* Į buferį nuskaitoma sekanti instrukcija po vykdomos.
• *Instrukcijos dekodavimas (ID).* Nustatomi operacijos kodas ir operando specifikatorius.
• *Operando apskaičiavimas (OA).* Apskaičiuojamas operando efektyvusis (realusis arba fizinis) adresas. Tai gali būti perkėlimo, registrų netiesioginis adresavimo ir kitų formų adresų apskaičiavimai.
• *Operandų išranka (OI).* Operandas arba operandai nuskaitomas iš atminties.
• *Instrukcijos vykdymas (IV).* Vykdoma instrukcijoje nurodyta operacija ir išsaugomas jos rezultatas.
• *Operando rašymas (OR).* Operacijos rezultatas įrašomas į atmintį.



7.5 pav. Instrukcijų konvejeriavimo laiko diagrama

Esant tokiam instrukcijos dekomponavimui, visų jos vykdymo elementarių etapų trukmės apytiksliai lygios. 7.5 paveiksle parodyta kaip šešių etapų konvejeriavimas gali sutrumpinti 9 instrukcijų vykdymą nuo 54 (9 ´ 6 = 54) iki 14 sąlyginių laiko vienetų. Vaizdumo padidinimui čia priimta, kad visų etapų trukmės visiškai lygios.

Išnagrinėkime 7.12 paveiksle esančią diagramą atidžiau. Joje priimta, kad kiekviena instrukcija praeina visas šešias konvejeriavimo stadijas. Realiai taip ne visada būna, pvz., įkėlimo {load} instrukcijoje nėra OR etapo. Šiame paveiksle, konvejeriavimo procesui supaprastinti, taktavimas padarytas taip, kad kiekvienai instrukcijai reikalingi visi 6 etapai. Diagramoje taip pat matyti, kad visi konvejeriavimo etapai vykdomi lygiagrečiai. Iš to išplaukia, kad konfliktų atmintyje kilti neturėtų. Pvz., II, OI ir OR etapuose vykdomos kreiptys į atmintį. Diagramoje numatyta, kad visos šios kreiptys gali vykti vienu metu. Dauguma realių atminties sistemų to neužtikrina. Tačiau reikalinga informacija gali būti patalpinama ne pagrindinėje atmintyje, o spartinančiojoje atmintyje. Be to, OI arba OR etapai gali būti nuliniai. Taigi daugeliu atvejų atminties konfliktai nelėtina konvejeriavimo.

Yra ir kitų konvejeriavimo efektyvumą mažinančių faktorių. Pvz., jeigu skiriasi trukmės minėtų 6 etapų – kai kuriuos etapuose turi būti numatytas laukimo laikotarpis. Kitokių problemų sukuria sąlygiškojo šakojimosi instrukcija. Ji turi neigiamos įtakos kelių instrukcijų išrankoms. Tam tikrą neapibrėžtumą taip pat įneša pertrauktys.

7.4.2. Šakojimosi problemos sprendimas

Viena didžiausių problemų kuriant instrukcijų konvejerį – sukurti stabilų instrukcijų srautą pradinėse konvejerio etapuose. Anksčiau buvo parodyta, kad pirmoji kliūtis stabiliam srautui sukurti yra sąlygiškojo šakojimosi instrukcija. Nes, kol instrukcija nebus įvykdyta, negalima tiksliai žinoti ar įvyks šakojimasis ar ne.

Sąlygiškojo šakojimosi problemai pašalinti taikomi tokie metodai:

• Daugialypiai srautai;
• Šakojimosi krypties išankstinė išranka;
• Ciklinis (kilpinis) {loop} buferis;
• Šakojimosi nuspėjimas;
• Uždelstas šakojimasis;

Daugialypiai srautai

Paprastame konvejeriavime, esant šakojimosi instrukcijai, dažnai atsiranda prastova, kadangi į instrukcijos registrą turi būti įrašyta sekanti viena iš dviejų galimų instrukcijų. O ji gali būti išrinkta neteisingai. Šiuo atveju galima toks sprendimas, pakartoti konvejerio pradinį etapą ir leisti jam abiejų instrukcijų išranką, tokiu būdu sudaroma galimybė toliau dirbti dviem srautais. Tačiau šis sprendimas sukuria kelias problemas:

• Esant daugialypiams srautams, atsiranda papildomi uždelsimai dėl kreipčių į registrus ir į atmintį.
• Dar iki pirminio šakojimosi apdorojimo pabaigos, į konvejerį gali būti įtrauktos kitos šakojimosi instrukcijos.

Kiekvienai tokiai naujai šakojimosi instrukcijai reikalingas atskiras srautas.

Nors daugialypiai srautai turi trūkumų, jie gerokai padidina darbo spartą. Pvz., kompiuteriuose IBM370/168 ir IBM3033 galimi du ir daugiau srautų.

Šakojimosi krypties išankstinė išranka

Naudojant šį metodą, aptikus sąlygiškąjį šakojimąsi, išrenkama įprasta šakojimosi instrukcija ir dar papildomai vadinamoji šakojimosi kryptis, t. y. instrukcija, į kurią gali persijungti programa, esant teigiamai šakojimosi sąlygai. Vykdant šakojimosi instrukciją, šakojimosi kryptis visą laiką saugoma. Jeigu išanalizavus sąlygą, šakojimasis privalo įvyksti, tai šiuo atveju jo kryptis jau būna iš anksto išrinkta.

Toks šakojimosi metodas taikomas kompiuteryje IBM 360/91.

Cikliškas buferis

Cikliškas buferis {loop buffer} yra nedidelė labai sparti konvejerio išrankos etape veikianti atmintis. Šioje atmintyje saugoma *n* paskutinių išrinktų instrukcijų. Jeigu šakojimasis turi įvykti, tai techninė įranga visų pirma patikrina ar cikliškame buferyje nėra įsiminta šakojimosi krypties. Buferyje ją aptikus, sekanti instrukcija išrenkama iš cikliško buferio.

Šakojimosi nuspėjimas

Šakojimuisi nuspėti taikomi labai įvairūs metodai. Bendriausieji iš jų yra tokie:

• *Nuspėjimas netaikomas* – Predict Never Taken,
• *Nuspėjimas nuolat taikomas* – Predict Always Taken,
• *Nuspėjimas pagal operacijos kodą* – Predict by Opcode,
• *Nuspėjimas taikomas/netaikomas jungiklis* – Taken/Not Taken Switch,
• *Šakojimosi istorijos lentelė* – Branch History Table.

Pirmieji trys metodai yra statiniai. Jie neįvertina programos vykdymo priešistorijos iki šakojimosi instrukcijos atsiradimo momento. Du paskutiniai būdai priklauso nuo programos vykdymo priešistorijos ir vadinami dinaminiais.

Uždelstas šakojimasis

Konvejeriavimo efektyvumą galima padidinti automatiškai sutvarkant programoje instrukcijas taip, kad visos šakojimosi instrukcijos atsirastų vėliau negu tai buvo iš anksto numatyta.

Literatūra apie CPĮ

1. **R.Lunde.** Empirical evaluation of some features of instruction set processor architectures. Communications of the ACM. March 1977.
2. **F.Willams, G.Steven.** Address and data register separation on the M68000 family // Computer architecture news, June 1990.
3. **B.Peuto.** Architecture of a new microprocessor // Computer, February 1979.
4. **S.Heywood.** The 8086 – an architecture for the future // Byte, June 1983.
5. **E.Stritter, T.Gunter.** A microprocessor architecture for a changing world: The Motorola 68000 // Computer, February 1979.
6. **D.Lilja.** Reducing the branch penalty in pipelined processors // Computer, July 1988.
7. **D.Anderson, T.Shanley.** Pentium processor system architecture (PC system architecture series) / Addison-Wesley Pub Co, 1995
8. **B.D.Shriver.** The anatomy of a high-performance microprocessor: A systems perspective / Bk&Cd-Rom edition, 1998.
9. **J.Silc, B.Robic,T.Ungerer.** Processor Architecture: From Dataflow to Superscalar and Beyond / Springer Verlag, 1999.