KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Nerijus Kislauskas

Šiuolaikinių procesorių architektūrų tyrimas, našumo lyginamoji analizė

Magistro darbas

Vadovas
doc. S. Maciulevičius

KAUNAS, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

TVIRTINU
Katedros vedėjas
doc. E. Kazanavičius
2005-05-23

Šiuolaikinių procesorių architektūrų tyrimas, našumo lyginamoji analizė

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių k. katedros lekt.
dr. J. Mikelionienė
2005-05-19

Recenzentas
dr. R. Marcinkevičius
2005-05-20

Vadovas
doc. S. Maciulevičius
2005-05-20

Atliko
IFM 9/1 gr. stud.
N Kislauskas
2005-05-23

KAUNAS, 2005

KVALIFIKACINĖ KOMISIJA

Pirmininkas:	Laimutis Telksnys, akademikas
Sekretorius:	Stasys Maciulevičius, docentas
Nariai:	Rimantas Barauskas, profesorius
	Raimundas Jasinevičius, profesorius
	Jonas Kazimieras Matickas, docentas
	Jonas Mockus, akademikas
	Rimantas Plėštys, docentas
	Henrikas Pranevičius, profesorius

Turinys

SUMMARY

Investigation on architectures of processors and comparative analysis of their efficiency

The work deals with main aspects of computer efficiency increase. Object of investigation is a system consisting of processor, memory, other components and connecting links called buses. Rather different systems are used in modern world of information, so an interest arises to compare architectures of several producers. Comparison of systems is quite possible as the same architectural features bind them together: processor‘s clock speed, cache, memory, dual channel technology and others. To perform a comparative analysis, software has been used enabling to reveal increase of efficiency of separate computer components. Systems chosen for study are rather new from the point of view of technology: Intel Pentium 4, AMD Athlon XP and AMD Sempron. Experiments having been fulfilled, it came out that efficiency of a system for the most part depends on processor capacity: increase of its clock speed results in 9 – 13%, L1 cache has an effect even up to 1350% (theoretically), L2 cache – 30 – 38%. HyperThreading has been observed to mostly result in operations with floating point numbers (even up to 68%), and branch prediction would have theoretically to increase efficiency up to 47%. Estimating indicator of efficiency of the whole system, the results show that the main role belongs to processor. Influence of other components of the system is less noticeable. Working peculiarities of memory type determine rate of data selection and transmission from memory. The study has shown that memory timings increase work speed only from 0,2 to 0,5%, whereas dual channel memory allows data transmission to be increased up to 65%. It is rather difficult to investigate influence of buses on the system, therefore only theoretical evaluation of this component is presented – 10%. On estimating results of the research, a conclusion may be drawn that comparison of components of different manufacturers is possible only partly - when equal criteria of comparison are agreed about. As all computer components have influence on efficiency of the system, so comparing different architectures each component must be taken into account. Analysis presented in this work will be useful for people investigating various aspects of computer performance, studying characteristics of systems efficiency and their estimation.

Lentelių sąrašas

Paveikslų sąrašas

1 Įvadas

Galima drąsiai teigti, kad kompiuterinės ir informacinės technologijos vis labiau skverbiasi į mūsų gyvenimą, kaip jokia kita. Kompiuteriai greitėja neregėtais tempais. Nors informacijos mokslui dar tik apie 60 metų, bet jis radikaliai keičiasi apimdamas vis naujas sritis. Prieš 50 metų niekas neįsivaizdavo, kad XXI amžiuje gyvensime apsupti elektronikos, skaičiavimo įrenginių, kompiuterių. Dabar daugelis darbo vietų neįsivaizduojami be kompiuterio ant darbo stalo: sekretorės, vadybininko, projektų vadovo, direktoriaus ar net pardavėjos. Nereikia žvelgi toli kad įsitikinti, jog gyvename elektroniniame amžiuje. Informacijos mokslas mokyklose privalomas jau nuo 5 klasės, 9 klasėje mokinys turi mokėti laisvai naudotis kompiuteriu kaip darbo ar komunikavimo priemone, 12 klasėje jau netgi geba sukurti programinį produktą. Vis daugiau darbdavių reikalauja darbo kompiuteriu įgūdžių. Versle kompiuteriai atlieka dar platesnį spektrą darbų: panaudojami nuo skaičiavimo mašinėlės iki tarnybinės stoties grupinių darbo sistemų valdymui, tyrimams ar skaičiavimams atlikti. Sekretorėmis jis dokumentų rengimo ir blankų ruošimo priemonė, direktoriui – elektroninis darbo planavimo kalendorius, buhalterei – pagalbinė priemonė finansinei apskaitai vesti. Didelėse įmonėse tarnybinės stotys naudojamos duomenims, dokumentams ar jų kopijoms saugoti, darbuotojų darbo aplinkoms palaikyti. Laboratorijose kompiuteriais apdorojami mokslinių duomenų rezultatai, atliekami sudėtingi skaičiavimai, rezultatų analizės, sprendžiami realaus pasaulio uždaviniai (oro ir vėjo temperatūros stebėjimai, tiriami žemės drebėjimai) ir kt.

Istoriškai taip susiklostė, kad kompiuterių gamintojai susiskirstė į keletą grupių ir varžosi tarpusavyje skirtingose kategorijose. Vaizdo plokštės gamintojai varžosi, kuris pagamins ir rinkai pasiūlys plokštę su geresniu 3D grafikos spartintuvu, atminties gamintojai varžosi dėl didesnės atminties talpos, procesorių – kas pagamins greitesnį procesorių ir pan. Šių lenktynių rezultatas – skirtingos platformos, skirtingos technologijos, skirtingos architektūros. Kiekvienas gamintojas stengiasi savo produkte realizuoti architektūrines naujoves ir išradimus, kad būtent jo prekė greičiau atsidurtų parduotuvių prekystaliuose ir galbūt taptų nauju standartu ateities technologijoms. Toks skubėjimas naudingas tiek gamintojui marketingo prasme (technologija parduodama už aukštesnę kainą), tiek vartotojui (įsigyjama pažangesnė technologija, nei turima dabar). Marketingo, kainos ir technologijos kokybės santykis į šo darbo temą neįeina, todėl apie jį nekalbėsime. Galima tik pasakyti, kad technologijų vystymąsi didžiąja dalimi lemia rinkos paklausa, gamintojų mokslinių technologijų bazė bei žinoma kainos.

Temos aktualumas, mokslinis naujumas, teorinė ir praktinė nauda. Šis darbas nėra pirmas KTU universitete, tyrinėjantis kompiuterių sandarą, skirtingų platformų architektrūras ir jų skirtumus. Šią kryptį plėtoja darbo vadovas, parašė keletą straipsnių. Pats kompiuterių palyginimas yra senas reiškinys, nes tobulėjant technologijoms atsiranda poreikis lyginti vis naujesnes sistemas, tyrinėti jų vidinę architektūrą ir elementų, įtakojančių kompiuterių spartą, visumą. Užsienio mokslininkai taip pat dirba šioje srityje, todėl dažniausia bus remtasi jų teoriniais ir praktiniais rezultatais, įtakojančiais kompiuterių našumo rodiklį. Šia tema rašo straipsnius specialistai, dirbantys šių dienų kompanijose – lyderėse technologijų rinkoje, o jų parengti dokumentai dažnai būna vertingi ieškant atsakymų į iškilusius klausimus. Šiame darbe bus detaliai aptarti tie kompiuterio komponentai, kurie tiesiogiai įtakoja darbo spartą, pateiktos istorinės technologijos vystymosi detalės ir teoriniai našumo apibendrinimai. Magistro darbo temos analizė bus naudinga toliau plėtojant šios krypties tematiką. Atliktų eksperimentų pagalba bus siekiama įrodyti kai kurių visuomenės požiūrių į tam tikras sistemas pagrįstumą, kodėl vieno ar kito gamintojo produktai yra našesni. Bandymas ir tyrimas turėtų išryškinti kai kurių sistemos dalių vidinės architektūros ypatybes, jų specifiką. Darbe taipogi bus aptariamos našumo matavimo ir spartos didinimo priemonės, kurios naudojamos informacinėje visuomenėje, atskleisti jų privalumai ir trūkumai.

Tyrimo objektas. Šio magistrinio darbo tyrimo objektas – 32 ir 64 bitų kompiuterių sistemos, dažniausiai sutinkamos šiandieninėje informacinių technologijų rinkoje, naudojamos moksliniuose bandymuose, tyrimuose, skaičiavimuose ir t.t.

Šiuolaikinės architektūros pasižymi įvairove. Daug gamintojų, daug to pačio gamintojo produktų. Skirtingo laikmečio architektūros priklauso skirtingoms kartoms, o ir to paties laikmečio – skirtingoms šeimoms. Svarbiausias kompiuterio komponentas – procesorius. Jo architektūrinės savybės didžiausia dalimi lemia viso kompiuterio spartą, todėl šiame darbe daugiausia dėmesio bus skirta procesoriaus darbo spartos nagrinėjimui, skirtingų gamintojų skirtingų šeimų realizacijų panašumus ir skirtumus bus stengtasi išnagrinėti kaip galima detaliau įvertinant spartą lemiančius niuansus. Kaip minėta anksčiau, sistema sudaryta iš komponentų. Nereikia pamiršti ir atminties, kurios organizacijos ypatumai smarkiai įtakoja visos sistemos darbą, nes skiriasi jos tipai bei realizacijos būdai. Duomenys perduodami kanalais, vadinamais magistralėmis. Magistralių tipų yra daug, bet šiame darbe stengtasi išanalizuoti tik tas, kurios turi tiesioginę įtaką kompiuterio spartai. Tyrimo objekto vienetu laikoma sistema, turinti vieną ar kelis iš ankščiau minėtų komponentų ar jų realizacijų.

Tyrimo tikslai ir uždaviniai. Būtų naivu lyginti kompiuterių sistemą, pagamintą šiandien, su sistema, pagaminta prieš 10 metų. Dar labiau tokį lyginimą apsunkina skirtingos

šių sistemų architektūrų ypatybės. Pagrindinis šio darbo tikslas – palyginti 32 ir 64 bitų sistemas siekiant nustatyti lyginimo kriterijus bei suvienodinti šių procesorinių sistemų našumo įvertį lyginimo prasme, bei kiek leidžia magistro darbo apimtis, išryškinti galimas lyginimo problemas. Siekiant užsibrėžto tikslo šiame darbe išsikelti tokie uždaviniai:

1) apžvelgti technologinių raidos etapų specifines kompiuterio darbo našumo kėlimo priemones;
2) apžvelgti ir išanalizuoti procesoriaus, atminties, magistralių architektūrines realizacijas;
3) įvertinti skirtingų platformų realizacijų darbo našumo rodiklius;
4) pasirinkus priemones ištestuoti šiuolaikinių sistemų architektūrų komponentus bei pateikti objektyvų vertinimą

Tyrimo metodika. Tiriant sistemų architektūrų našumą buvo pasirinkta keletas pilnai sukomplektuotų sistemų skirtingose gamintojų platformose. Kadangi šiuo metu procesorių rinkoje dominuoja dvi konkurencingos firmos (Intel ir AMD), darbo tyrimas bus atliekamas būtent šių firmų produktais. Tiriant 32 bitų architektūras, iš Intel firmos buvo pasirinktas Pentium 4 procesorius kaip šviežiausias produktas su naujausiomis technologijomis. Iš AMD firmos procesorių pasirinkome AMD Athlon XP bei AMD Sempron procesorius ir jiems tinkančias platformas. Tyrimai apims procesorių, atmintį bei magistralę.

Procesorių tyrimui atlikti bus pasitelktos tam skirtos priemonės, vykdomos operacijos tiek sveikųjų skaičių, tiek trupmeninių skaičių įtaisuose. Didinamas taktinis dažnis turės atspindėti procesoriaus našumo galimybes. Toliau bus testuojama spartinančiosios atmintinės (tiek L1, tiek L2) įtaka procesoriui dirbant su ja, ir be jos. Atminties tyrimas bus atliktas kaitaliojant jos dažnį. Taipogi darbo našumui turi įtakos duomenų iš atminties išrinkimo laikas, todėl vienas iš tyrimo uždavinių bus atminties kreipinių įtaka bendram sistemos našumui. Magistralių testavimui naudosime programas, kurios pamatuos jų pralaidumą esant normaliam ir padidintam dažniui. 64 bitų architektūrų gauti bei ištestuoti nepavyko.

Darbo struktūra. Šį darbą sudaro keturios dalys: įvadas, teorinė dalis, praktiniai eksperimentai bei išvados. Kiekviena darbo dalis suskirstyta į skyrius, kuriuose analizuojamas sistemos komponentas. Skyriai skaidomi į poskyrius, kuriuose nagrinėjamas paties komponento skirtingos darbingumo specifikos. Magistro darbo struktūrai turėjo įtakos analizuojamų komponentų savybės ir realizacijos.

2 Teoriniai kompiuterių našumo aspektai

Šiuolaikiniame informacijos ir kompiuterių pasaulyje egzistuoja daug įvairių skaičiavimo sistemų, architektūrų, mašinų. Laisvos rinkos sąlygomis gamintojai varžosi tarpusavyje, kad tiek vartotojui, tiek verslui pasiūlyti kuo geresnį produktą už kuo priimtinesnę kainą. Kompiuterių rinkoje geresnis produktas įvertinamas didesniu našumo rodikliu. Kiekvienas pirkėjas nori gauti geresnį kompiuterį už sumokėtą kainą. Dažnai tenka spręsti klausimą, kuri sistema geresnė, dėl to pradėta naudoti našumo sąvoka ir informaciniame pasaulyje: našesnė sistema, našesnis kompiuteris, našesnis procesorius. Dabartinės lietuvių kalbos žodynas pateikia tokią šio žodžio aiškinamąją reikšmę:

našlus, ~i (4)

1. gerai derantis, duodantis daug vaisių (apie augalus): Našūs rugiai. Našios obelys.
2. derlingas (apie žemę): Našūs laukai.
3. sukuriantis daug vertybių: N. darbas. Našios staklės.
4. greitas, vikrus, mitrus: N. žmogus, arklys.
5. lengvas naudoti, patogus: N. grėblys, kirvis, botagas. Našios šlepetės. N. sijonas.

~iai prv.

~umas (2) [1]

Informacinėje visuomenėje našumo terminas suprantamas kaip svarbus veiksnys, nusakantis sistemos produktyvumą atliekant specialias užduotis per tam tikrą laiko vienetą. Sistema laikykime į visumą sujungtus kompiuterio komponentus (atmintis, procesorius, magistralės, valdikliai, disko kaupikliai ir t.t.). Sistemos našumą galima įvertinti tik tiriant jos komponentų darbą. Procesoriaus našumas dažniausiai nusakomas dviem parametrais: a) *CPI* – taktų skaičiumi vienai komandai įvykdyti; b) taktų dažniu *F*.

CPI randamas taip:

$$CPI = \sum_{i=1}^{n} CPI_i * F_i \qquad (1)$$

kur $F_i = \frac{kiekis_i}{kiekis_{visu}}$ – i-jo tipo komandų dažnis uždaviniuose,

$CPI_i = i - tosios\ komandos\ vykdymo\ laikas\ taktais.$ (2)

Taktų dažnis: $F = \frac{1}{\tau}$, (3)

kur τ – takto periodas.

Atminties ir magistralės našumas dažniausiai atspindimas jų darbo dažniu. Kartais magistralės našumui nusakyti naudojamas ir informacijos pralaidumo rodiklis. Taigi, tiriant sistemos darbo spartą būtina apžvelgti visus ją įtakojančius komponentus ir jų charakteristikas.

2.1 Procesorius

Pačią didžiausią įtaką kompiuterio našumui turi procesorius. Procesorius – tai elektroninis įrenginys, galintis vykdyti komandas bei duoti komandas kitiems įrenginiams. Jo architektūrinės ypatybės didžiąja dalimi lemia kompiuterio spartą. Procesoriaus vidaus architektūros ir darbo principai neįeina į šio darbo tyrimą, todėl panagrinėsime tik tuos aspektus, kurie glaudžiai susiję su darbo spartos (našumo) padidėjimu.

2.1.1 Spartinančioji atmintinė. Spartinančiosios atmintinės organizacija procesoriuje

Spartinančioji atmintinė – tai labai greita mažo ar vidutinio dydžio atmintis, kuri jungiama tarp procesoriaus ir pagrindinės atminties ir skirta atminties sistemos našumui padidinti [2]. Kadangi beveik visi šiuolaikiniai procesoriai yra SIMD (vienas komandų srautas, daug duomenų srautų – M. Flyno klasifikacija) tipo, tai vykdant komandą reikalingi keli operandai. Operandų išrinkimas iš atminties yra lėtas, todėl nuspręsta į patį procesorių įdiegti mažą, bet sparčią atmintį. Tokia atmintis dažniausiai būna dviejų lygių:

- pirmo lygio (dar vadinama L1 lygio) – tai sparčiausia atmintis, esanti procesoriaus viduje. Jos vėlinimas 1 – 2 taktai;
- antro lygio (dar vadinama L2 lygio) – šiek tiek lėtesnė atmintis, taip pat esanti procesoriaus viduje ir užimanti iki 50% viso procesoriaus ploto. Jos vėlinimas iki 20 – 30 taktų

Dar būna trečio lygio (L3 lygio) spartinančioji atmintinė, montuojama ant pagrindinės plokštės, todėl yra lėčiausia (iki 40 taktų). Pačios darbinės atminties vėlinimas siekia 200 – 300 taktų.

Spartinančiosios atminties panaudojimas remiasi lokališkumo principu:

- laiko atžvilgiu – jei dabar reikalingas koks nors elementas, tai labai tikėtina, jog netrukus ir vėl jo prireiks;
- vietos atžvilgiu – jei dabar reikalingas koks nors elementas, tai labai tikėtina, jog netrukus bus reikalingas ir jam gretimas elementas.

L1 spartinančioji atmintinė dažnai sudaroma iš atskirų komandų ir duomenų atminčių. L2 spartinančioji atmintinė skirta tiek komandoms tiek ir duomenims kartu saugoti. L3 spartinančioji atmintinė pasiekiama tik per sisteminės magistralės sąsają, o L1 ir L2 – tiesiai iš procesoriaus.

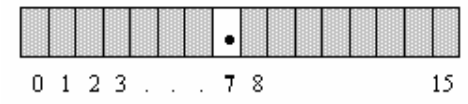
Panagrinėkime spartinančiosios atmintinės struktūrą. Nesvarbu kokio lygio spartinančioji atmintis bebūtų, ji sudaryta pagal žemiau esančią schemą:

Eil.Nr.	Žyma	Duomenys
0		
1		
…		
N-1		

1 pav. Bendra spartinančiosios atmintinės schema (1 krypties)

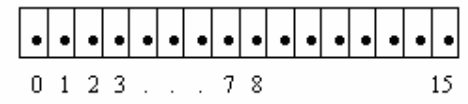
Žymos lauke saugoma informacija padeda nustatyti, kuris iš galimų pagrindinės atminties blokų yra saugomas šioje eilutėje. Be to į žymą įeina ir kita informacija, kuri priklauso nuo spartinančiosios atmintinės organizacijos ir darbo ypatybių. Duomenų lauko ilgis paprastai būna 4 – 128 baitai. Pagal organizaciją spartinančioji atmintis gali būti skirstoma į 3 tipus:

- tiesioginio atvaizdavimo – kiekvienas iš pagrindinės atminties paimtas eilutės dydžio blokas turi vieną apibrėžtą vietą



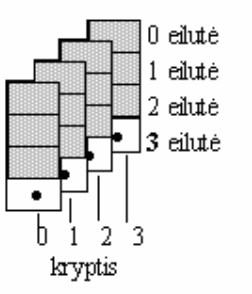
2 pav. Tiesioginio atvaizdavimo spartinančioji atmintinė

- visiškai asociatyvi – iš pagrindinės atminties paimtas eilutės dydžio blokas gali būti bet kurioje vietoje



3 pav. Visiškai asociatyvi spartinančioji atmintinė

- iš dalies asociatyvi – iš pagrindinės atminties paimtas eilutės dydžio blokas gali būti vienoje iš k vietų; k vadinamas asociatyvumo laipsniu arba krypčių skaičiumi



4 pav. Iš dalies asociatyvi spartinančioji atmintis

Lengviausia realizuoti tiesioginio atvaizdavimo atmintinę, bet tokio tipo atmintinė yra labiausiai apribota. Pagrindinės atminties atitinkamos vietos gali būti įkeltos į vieną ir tik į vieną spartinančiosios atmintinės vietą, todėl būtina pagrindinę atmintį skaidyti į atitinkamo dydžio puslapius. Jei turime 64 KB dydžio spartinančiąją atmintinę, tada pagrindinė atmintis turės būti logiškai suskaidoma į 64 KB dydžio puslapius. Tokio tipo atmintis turi greičio privalumą, kai duomenų reikia ieškoti tik vienoje vietoje. Jos našumas krenta tada, kai procesoriaus kreipiniai nuolat kartojasi į du (ar daugiau) skirtingus atminties puslapius. Visiškai asociatyvi spartinančioji atmintis yra praktiškai nenaudojama dėl sudėtingos realizacijos. Tokiai spartinančiosios atmintinės organizacijai komparatorių reikia tiek pat, kiek yra eilučių, sudėtingas būtų ir multiplekseris. Plačiausiai naudojamas iš dalies asociatyvios spartinančiosios atmintinės organizacijos tipas, kur krypčių skaičius būna 2, 4 arba 8, todėl toliau tyrinėsime tik tokio tipo spartinančiąją atmintinę.

Informacijos paieška iš dalies asociatyvioje atmintyje vykdoma taip:

1) pagal kreipinio adresą nustatomas eilučių, kurioje gali būti ieškoma informacija, indeksas;

2) išrenkamos eilutės ir patikrinama, ar yra eilučių, kurių bitas V=1; jei ne, - šios spartinančiosios atmintinės eilutės yra tuščios, todėl fiksuojama, kad į spartinančiąją atmintinę nepataikyta;

3) jei kurios nors eilutės bitas V=1, ši spartinančiosios atmintinės eilutė užpildyta, todėl tikrinamos tos eilutės aukščiausios adreso skiltys. Aišku, su aukščiausiomis adreso skiltimis sutapti gali tik vienos eilutės žyma. Jei nesutampa nė vienos eilutės žyma su aukščiausia adreso skiltimi, fiksuojama, kad į spartinančiąją atmintinę nepataikyta;

4) jei kurios nors eilutės žyma sutampa su aukščiausiomis adreso skiltimis, fiksuojama, kad į spartinančiąją atmintinę pataikyta; tuomet pagal trečią ir antrą adreso skiltis išrenkamas eilutėje saugomas žodis.

Jei į spartinančiąją atmintinę nepataikyta, reikia kreiptis į pagrindinę atmintį. Perskaityta informacija bus perduota į procesorių, taip pat ir įrašyta į spartinančiąją atmintinę. Įrašant ją į

spartinančiąją atmintinę, kartu turi būti įrašomi ir gretimi žodžiai, priklausantys tam pačiam atminties blokui. Nauja eilutė gali būti užpildyta taip:

1) pagal kreipinio adresą išrenkamas pirmasis žodis ir įrašomas į atitinkamą vietą eilutėje (pagal trečią ir antrą adreso skiltis);

2) dinaminės atminties valdiklis, užfiksavęs kreipinio adresą, generuoja dar tris kreipinius į pagrindinę atmintį, kad būtų išrinkti kiti trys eilutės žodžiai;

3) į žymos lauką įrašomos 6 aukščiausiosios adreso skiltys, o į V lauką - 1.

Spartinančiosios atmintinės eilutės pakeitimas. Prieš pakeičiant spartinančiosios atmintinės eilutę reikia užtikrinti, kad joje esanti informacija atitiktų saugomą pagrindinėje atmintyje. Tai priklauso nuo to, kaip informacija įrašoma į spartinančiąją atmintinę ir pagrindinę atmintį vykdant rašymo operaciją. Pagal tai skiriami du spartinančiosios atmintinės tipai:

- ištisinio įrašymo (*write through*) spartinančiojoje atmintinėje įrašymo metu informacija rašoma ir į atmintinę, ir į pagrindinę atmintį;
- atidėto įrašymo (*write back*) spartinančiojoje atmintinėje informacija rašoma tik į atmintinę; spartinančiosios atmintinės eilutės turinys perkeliamas į pagrindinę atmintį eilutės pakeitimo metu, jei bent vienas šios eilutės bitas buvo pakeistas. Šiam faktui pažymėti eilutėje būna M arba D bitas (*dirty bit*), į kurį įrašomas 1, kai vykdoma rašymo į šią eilutę operacija.

Iš dalies asociatyvioje spartinančiojoje atmintinėje nepataikymo atveju turi būti parinkta viena iš k eilučių (kryptis), į kurią bus įkelta informacija iš pagrindinės atminties. Ji parenkama tokiu būdu:

1) jei yra bent viena neužpildyta eilutė (tokios eilutės bitas V=0), į ją bus įrašytas blokas iš pagrindinės atminties; 2) jei visos eilutės užimtos (visų eilučių bitai V=1), reikia parinkti vieną iš jų. Pakeičiamai spartinančiosios atmintinės eilutei parinkti dažniausiai naudojami du metodai:

1) LRU (*Least Recently Used*) – seniausiai naudota,

2) atsitiktinis parinkimas.

Naudojant pirmą metodą pakeičiama ta eilutė, į kurią buvo kreiptasi seniausiai, nes tikėtina, kad į ją nebus kreipiamasi artimiausiu metu. Šio metodo realizacija sudėtinga, nes reikia fiksuoti paskutinio kreipinio laiką. Kadangi tam reikėtų daug papildomų bitų, praktiškai naudojamas pseudo LRU metodas: eilutėje būna tik vienas LRU bitas, kuris periodiškai išvalomas ir į kurį įrašomas 1 kreipinio į tą eilutę metu. Jei parenkant pakeičiamą eilutę kurioje nors iš kandidačių randamas 0, laikoma, kad ši eilutė ir yra "seniausiai naudota".

Atsitiktinio parinkimo metodas jokia papildoma informacija nesinaudoja, ir spartinančiosios atmintinės eilutę parenka atsitiktiniu būdu. Abu metodai turi savų privalumų ir trūkumų:

- ištisinio įrašymo (*write through*) atveju
  a) nereikia į atmintį grąžinti bloko eilutės pakeitimo atveju;
  b) spartinančiosios atmintinės turinys visuomet atitinka pagrindinės atminties turinį, kas ypač svarbu multiprocesorinėse sistemose;
- atidėto įrašymo (*write back*) atveju
  a) įrašymas trunka mažiau laiko - kiek yra būtina rašant į spartinančiąją atmintinę;
  b) mažiau apkraunamas atminties interfeisas ir magistralė, nes sumažėja bendras rašymo į pagrindinę atmintį operacijų skaičius; tai svarbu multiprocesorinėse sistemose.

Tarp kreipinių į spartinančiąją atmintinę dominuoja skaitymas, todėl šiai operacijai ir skiriama daugiausia dėmesio optimizuojant atmintinę, tačiau rašymo operacijos taip pat negalima užmiršti. Dabar įvertinkime spartinančiosios atmintinės našumą. Apsibrėžkime tokius dydžius:

t_{h1} – išrinkimo laikas pataikius į L1 spartinančiąją atmintinę;

t_{m1} – išrinkimo laikas nepataikius;

f_{h1} – pataikymo į L1 spartinančiąją atmintinę dažnis;

f_{m1} – nepataikymo į L1 spartinančiąją atmintinę dažnis ($f_{m1} = 1 - f_{h1}$).

Tuomet vidutinis duomenų išrinkimo laikas iš spartinančiosios atmintinės bus lygus:

$$t_{ah} = t_{h1} + f_{m1} * t_{m1} \qquad (4)$$

t_{m1} parodo, kiek laiko užtruktume duomenims išrinkti, jei į L1 lygio atmintinę būtų nepataikyta. Šiuolaikiniai procesoriai ieško duomenų ir L2 lygio atmintinėse, todėl 1 formulė turi būti išplėsta įvertinant duomenų išrinkimą L2 lygio spartinančiojoje atmintinėje:

$$t_{ah} = t_{h1} + f_{m1} * (t_{h2} + f_{m2} * t_{m2}) \qquad (5)$$

kur

$$t_{m1} = t_{h2} + f_{m2} * t_{m2} \qquad (6)$$

t_{h2} – išrinkimo laikas pataikius į L2 atmintinę;

t_{m2} – išrinkimo laikas nepataikius į L2 atmintinę (išrinkimo iš pagrindinės atminties laikas);

f_{h2} – pataikymo į L2 atmintinę dažnis;

f_{m2} – nepataikymo į L2 atmintinę dažnis ($f_{m2} = 1 - f_{h2}$).

5 formulėje įvertinome tiek pataikymo laiką į L1 lygio spartinančiąją atmintinę, tiek išrinkimo laiką į jį nepataikius, o pataikius į L2 atmintinę ar į pagrindinę atmintį. Iškyla klausimas, kaip atmintinės našumą įvertinti skaičiuojant viso kompiuterio darbo našumą. Tebūnie T – sprendimo laikas:

$$T = N*(CPI + t_{ah})*\tau \qquad (7)$$

Čia

N – komandų skaičius;

CPI – taktų skaičius, reikalingas įvykdyti komandai;

t_{ah} – vidutinis duomenų išrinkimo laikas;

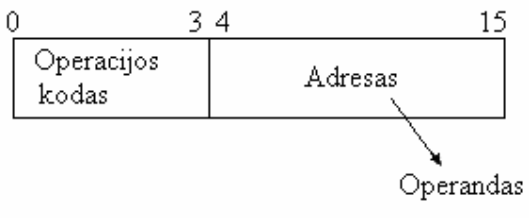
τ – takto periodas.

Užduoties atlikimo laikas T priklauso nuo keleto jį įtakojančių veiksnių. Pirmiausia užduoties atlikimo laikas priklauso nuo vykdomų komandų skaičiaus. Aišku, kad kuo daugiau komandų vykdo procesorius, tuo daugiau laiko jis sugaišta užduočiai atlikti. Jau žinoma, kad skirtingoms komandomis atlikti reikia skirtingų procesoriaus taktų skaičiaus. CPI reikšmė kaip tik įvertina tokią komandos vykdymo trukmę (žr. 1 formulę). Taipogi reikia atsižvelgti į tai, kad pačių komandų operandų (duomenų) išrinkimas vykdomas per keletą atminčių lygių ir išeikvoja papildomų procesoriaus taktų. t_{ah} rodiklis kaip tik įvertina duomenų išrinkimo trukmę. Būtų nelogiška užduočių atlikimo laiką sulyginti procesoriaus taktais, nes net ir skirtingų architektūrų procesoriai tai pačiai užduočiai atlikti gali užtrukti tą patį taktų skaičių. Tada toks palyginimas būtų neprasmingas. Kadangi skirtingų procesorių vieno takto periodas yra skirtingas, įvertindami kiekvieno procesoriaus takto periodą τ garantuojame, kad rezultatas, kurį pateiks 7 formulė, yra prasmingas ir objektyvus laiko atžvilgiu.

2.1.2 RISC ir CISC architektūros

Šiuolaikiniuose procesoriuose naudojamos tiek RISC, tiek CISC architektūros. Vienareikšmiškai atsakyti, kuri sistema yra našesnė, neįmanoma. Tam, kad įvertintume šių architektūrų privalumus ir trūkumus, panagrinėkime procesoriaus komandų sistemą.

Komandos pavyzdys:



5 pav. Komandos schema

Operacijos kodas gali būti viena iš šių komandų:

0001 = Įkrauk iš atminties į laikiną atmintinę
0010 = Išsaugok laikinos atmintinės turinį į atmintį
0101 = Sudėk laikinosios atmintinės turinį su pagrindinės atminties turiniu
.....

Adreso laukas yra pakankamai mažas, todėl, norint operuoti su didelėmis atminties adresų reikšmėmis, reikia aptarti keletą adresavimo būdų:

- Skubus adresavimas – operandas yra pačiame adresavimo lauke; dažniausiai naudojamas konstantoms aprašyti; nereikia kreipinių į atmintį; skaitinės reikšmės dydis yra apribotas lauko ilgiu;
- Tiesioginis adresavimas – adreso laukas laiko operando adresą;
- Netiesioginis adresavimas – adreso laukas laiko operando adreso adresą; jei adreso ilgis lygus j, tai tiesioginio adresavimo sritis bus 2^j; operando adreso ilgis bus k, o k>j; operandų skaičius tada gali būti 2^k;
- Tiesioginis registrų adresavimas – operandas yra registruose;
- Netiesioginis registrų adresavimas – operando adresas yra registruose;
- Adresavimas pakeitimu – tiesioginės adresacijos ir netiesioginio registrų adresavimo kombinacija.

Jeigu operando kodui saugoti naudojama k bitų, tai viso galimų operacijų kodų bus 2^k. Komandų sistema – tai rinkinys visų operacijų, kurias gali įvykdyti procesorius. Nėra vienareikšmiško atsakymo, kiek skirtingų komandų turėtų palaikyti kiekvienas procesorius. Pagal tai, kiek skirtingų operacijų palaiko procesorius, jie skirstomi į 2 tipus: RISC – kompiuteriai su supaprastinta komandų sistema ir CISC – kompiuteriai su sudėtinga komandų sistema.

2.1.2.1 RISC

RISC (*Reduced Instruction Set Computer*) – tai procesoriaus architektūros tipas, kurioje realizuoti optimizuoti maži komandų rinkiniai. Šioje architektūroje siekiama padidinti procesoriaus darbo spartą supaprastinant komandas taip, kad jos būtų atliekamos per vieną taktą, o sudėtingesnės komandos būtų išreiškiamos per paprastesniųjų rinkinius. RISC privalumai:

- Supaprastintos komandos vykdomos pakankamai greitai;
- Komandos yra vienodo ilgio;
- Komandos naudoja vieną ar kelis formatus, todėl jas lengviau dekoduoti;

- Komandos naudoja vieną ar kelis adresavimo būdus norint supaprastinti jų dekodavimą;
- „LOAD“ ir „STORE“ architektūra – tai reiškia, kad tik šios dvi komandos pasiekia atmintį – visos kitos operacijos (pvz., aritmetinės) atliekamos registruose;
- Išplėstas registrų bankas nereikalauja papildomų kreipinių į atmintį;
- Pati architektūra palaiko du (galbūt dar kelis) duomenų tipus – dažniausiai sveikųjų ir trupmeninių skaičių [3].

RISC architektūros labai palengvina programuotojų, kurie rašo kompiliatorius, darbą, nes dėl komandų paprastumo tenka mažiau gaišti laiko analizuojant mašininį kodą. RISC architektūrų pavyzdžiai: PowerPC, SPARC, ALPHA, MIPS.

2.1.2.2 CISC

CISC (*Complex Instruction Set Computer*) – tai tokios procesoriaus architektūros, kuriose realizuotas didelis kiekis skirtingų ir sudėtingų komandų. CISC architektūros filosofija yra ta, kad techninė įranga yra visada greitesnė už programinę įrangą, todėl programuotojui reikia pasiūlyti galingų komandų rinkinį tam, kad ir su mažomis programomis būtų kuo geriau išnaudojama techninė įranga. CISC lustai yra lėtesni lyginant su RISC procesoriais, bet naudoja mažiau komandų tai pačiai užduočiai atlikti. CISC charakteristikos:

- Mažai bendro naudojimo registrų;
- Palaiko daug adresavimo būdų;
- Palaiko daug duomenų tipų;
- Kintamas komandų ilgis;
- Komandų realizacija aukšto lygio kalboms [4].

Nepaisant to CISC architektūra turi ir apribojimų:

- Sudėtingos komandos panaudojimas nėra jau toks dažnas;
- Kreipiniai į atmintį, „LOAD“ ir „STORE“ yra lėti, todėl tai turi reikšmę kitų komandų vykdymui;
- Kreipiniai į procedūras ir funkcijas gali sudaryti dideles kliūtis perduodant argumentus, išsaugojant ir pasiimant reikšmes iš registrų.

Kaip paprastą pavyzdį paimkime CISC procesorių, kuris gali užkrauti duomenis iš atminties, įrašyti atgal į atmintį, sudėti du skaičius ir padauginti vieną skaičių iš kito. Sakykime, kad „LOAD“ komanda užtruks 2 ciklus, „STORE“ komanda taipogi užtruks 2 ciklus, „ADD“ irgi 2 ciklus, o „MULTIPLY“ – 5 ciklus. Paprastų dviejų skaičių sudėčiai

CISC išnaudos 8 ciklus: 2 užkrauti pirmą skaičių, 2 užkrauti antrą skaičių, 2 juos sudėti ir 2 ciklus sudėties rezultatui išsaugoti. Palyginimui paimkime RISC procesorių. Jo „LOAD“ ir „STORE“ užtrunka po 1 ciklą kiekvienai komandai (jis šioms komandoms ir optimizuotas), „ADD“ komanda užtrunka 3 ciklus ir neturi „MULTIPLY“ komandos. RISC procesorius dviejų skaičių sudėčiai užtrunka 6 ciklus: 1 užkrauti pirmam skaičiui, 1 – antram, per 3 ciklus juos sudeda ir 1 ciklas rezultato išsaugojimui. Naudojant paprastą sudėtį atrodytų, jog RISC procesorius 25% greitesnis, bet naudojant daugybą viskas yra šiek tiek sudėtingiau. Sakykim reikia sudauginti skaičius „3“ ir „4“. CISC procesorius užtruks 2 ciklus užkraunant tris, 2 ciklus užkraunant keturis, per 5 ciklus jis šiuos skaičius sudaugina ir per 2 išsaugo reikšmę. Viso 11 ciklų. RISC procesorius neturi daugybos komandos, todėl reikės viską daryti kartojant sudėtis. 1 ciklu užkraunama keturi, 1 ciklu – kiti keturi, 3 ciklai sudėti dviems skaičiams. Toliau procesorius naudoja 1 ciklą užkrauti rezultatą iš naujo, 1 ciklas vėl užkrauti keturis, 3 ciklus išnaudoja sudėčiai ir galiausiai 1 ciklą galutiniam rezultatui išsaugoti. Taigi RISC išnaudojo 12 ciklų.

2.1.2.3 RISC ir CISC palyginimas

Nors CISC atsirado anksčiau, RISC šiuo metu įsigalėjo kaip greitesnių ir paprastesnių procesorių architektūra. Šiandien RISC yra populiaresni, nes laikui bėgant pagrindinės atminties greitis bei jos vartojimas išaugo ženkliai, o asemblerinį kodą pakeitė aukšto lygio programavimo kalbos, todėl kompiuterinių sistemų projektuotojams teko ieškoti būdų, kaip padidinti kompiuterio našumą ir optimizuoti techninę įrangą. Šioms idėjoms įgyvendinti buvo pasirinkta RISC architektūra, nes paprastų komandų sistemos seka duodavo tą patį rezultatą, kaip ir sudėtingų komandų sistemos seka, bet su RISC buvo galima realizuoti daug paprasčiau (ir greičiau).

1 lentelė. CISC ir RISC skirtumai

CISC	RISC
Techninės įrangos išryškinimas	Programinės įrangos išryškinimas
Sudėtingos komandos, vykdomos per keletą taktų	Tiktai supaprastintos komandos, kurios vykdomos per vieną taktą
Atmintis – atminčiai	Registras – registrui
„LOAD“ ir „STORE“ yra kitų komandų dalis	„LOAD“ ir „STORE“ yra kaip atskiros komandos
Tranzistoriai naudojami realizuoti sudėtingoms komandoms	Daugiau tranzistorių išnaudojama atminties registrams

Šiaip ar taip, RISC filosofija įneša svarbių pranašumų. Kadangi komandai įvykdyti užtenka vieno takto, tai visa programa bus įvykdyta per tą patį laiko tarpą, kaip ir pvz. „MULTIPLY“ komanda CISC architektūroje, kaip kad anksčiau buvo parodyta. Šios supaprastintos komandos naudoja mažiau tranzistorių palikdamos daugiau vietos bendros paskirties registrams. Dar daugiau, jeigu bet kokia komanda įvykdoma per tą patį laiką, galimas išlygiagretinimo algoritmo panaudojimas.

Šiuolaikiniuose procesoriuose skirtumai tarp RISC ir CISC architektūrų pastebimai mažėja. Dauguma šiandieninių RISC procesorių palaiko netgi daugiau komandų nei senesni CISC procesoriai. Dažnai netgi tame pačiame procesoriuje pastebimi abiejų architektūrų požymiai: RISC paremtas PowerPC procesorius turi ir kai kurių CISC architektūros galimybių, o šiuolaikiniai procesoriai, pradedant Intel Pentium Pro ir AMD K6-2 branduoliais, nors ir pagaminti kaip RISC, išoriškai jie laikomi kaip CISC architektūros.

2.1.3 Konvejerizacija ir superkonvejerizacija

Tebūnie I_1, I_2, I_3, ... , I_n – komandų seka. Kiekviena komanda turi pereiti kelias logines fazes. Jos gali būti tokios:

F – komandos išrinkimas,

D – komandos dekodavimas,

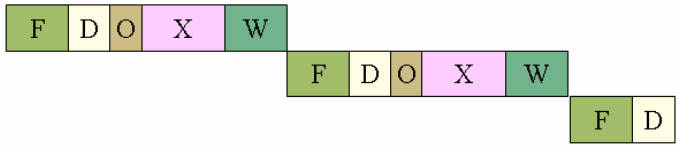
O – operandų išrinkimas,

X – komandų vykdymas,

W – rezultato įrašymas.

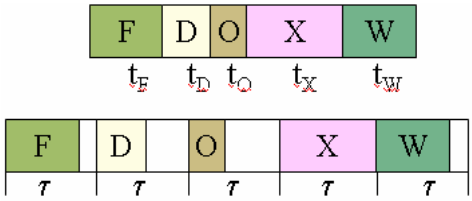
Esant nuosekliam komandų vykdymui, *(i+1) - oji* komanda pradedama vykdyti tik po to, kai baigiama vykdyti *(i) - oji* komanda:

Grafiškai tai būtų galima pavaizduoti taip:



6 pav. Nuoseklus komandų vykdymas

Konvejerizuotam procesoriaus komandų vykdymui reikia, kad pats konvejeris dirbtų ritmingai, t.y. loginių komandų fazių vykdymo laikas būtų vienodas:



7 pav. Komandų fazių vykdymo laikas

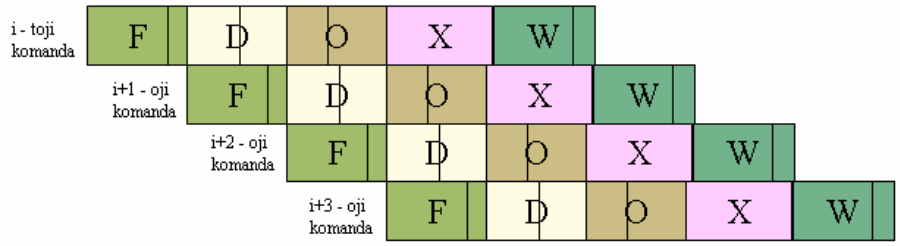
Reikia parinkti vienos fazės vykdymo laiką. Iš 7 paveikslėlio aišku, kad kiekvienos fazės vykdymo laikas skirtingas (t_F, t_D, t_O, t_X, t_W). Tokiu atveju yra paimamas ilgiausiai trunkančios fazės vykdymo laikas ir jis prilyginamas konvejerio etapo fazei, t.y.:

$$\tau = \max(t_F, t_D, t_O, t_X, t_W) \qquad (8)$$

kur

τ – laikas, reikalingas vienai fazei įvykdyti (takto trukmė).

Tuomet *(i+1) – oji* komanda pradedama vykdyti vienu etapu vėliau, nei *(i) – toji* komanda:



8 pav. Komandų vykdymas konvejeriu

Iš tikrųjų konvejerio darbas nėra toks idealus, kaip buvo vaizduota anksčiau. Konvejeris turi ir kliūčių: a) struktūrinės kliūtys – atsiranda todėl, kad resursai gali gali būti nepakankami komandų etapų reikmėms patenkinti; b) duomenų kliūtys – vienos komandos vykdymo rezultatas gali būti naudojamas kaip kitos komandos operandas; c) valdymo kliūtys – būtinumas išrinkti komandas iš kitos vietos, nei tai daroma jas vykdant nuosekliai. Šios kliūtys yra sprendžiamos konvejerio vykdymo metu, pvz., duomenų pirklausomumas išsprendžiamas įterpiant NOOP komandą (*no operation* – jokios operacijos, laukimas), komandų sekos pakeitimas, apylankų pridėjimas ir pan. Konvejerio valdymo kliūtys pašalinamos blokuojant tam tikrų komandų atitinkamų etapų darbą, dėl to atsiranda laiko nuostoliai. Norint sumažinti laiko nuostolius prognozuojami perėjimai (žr. 2.1.4 skyrių), keičiama komandų tvarka ir t.t.

Konvejerizuotoje sistemoje būtina įvertinti ir tarpinių fiksatorių vėlinimą. Tarpinių fiksatorių vėlinimą pažymėkime *d*. Tuomet konvejerio našumą galime įvertinti šitaip:

$$\tau = \max(t_F, t_D, t_O, t_X, t_W) + d = \tau_m + d \qquad 1 \le i \le k \qquad (9)$$

kur

τ_m – maksimalus fazės laikas (fazė, kuri turi didžiausią vykdymo laiką),

k – konvejerio pakopų skaičius,

d – laikas, reikalingas signalams ir duomenims perduoti iš vienos fazės į kitą [5].

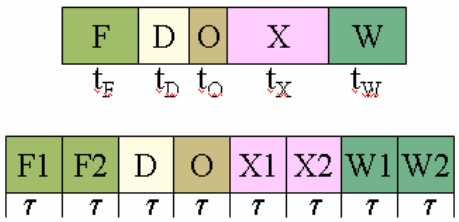
Žinoma, kad $\tau_m >> d$. Sakykime, kad vykdome n komandų nuoseklia tvarka. Laikas T_k, kuris bus reikalingas įvykdyti n komandų bus lygus:

$$T_k = [k + (n-1)]\tau \qquad (10)$$

Iš 8 paveikslėlio matyti, kad vienai komandai įvykdyti reikia *k* taktų, o likusios *n-1* komandų reikalauja *n-1* taktų. Iš čia gauname, kad šioms komandoms įvykdyti reikia 8 taktų : 5+(4-1) = 8. Palyginus komandų vykdymo konvejeriu našumą su nekonvejerizuotu komandų vykdymu gauname:

$$S_k = \frac{T_1}{T_k} = \frac{nk\tau}{[k + (n-1)]\tau} = \frac{nk}{[k + (n-1)]} \qquad (11)$$

Superkonvejerio darbo principas niekuo nesiskiria nuo konvejerio. Skirtumas yra tas, kad superkonvejeryje komandos išrinkimo, vykdymo ir įrašymo fazės yra skaidomos į dar trumpesnes fazes, kas leidžia padidinti teorinį našumą net iki 33% [6].

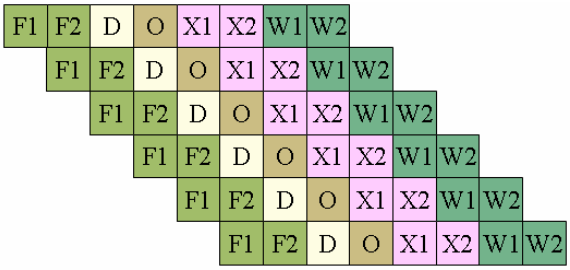


9 pav. Superkonvejerio fazių vykdymo laikas

Superkonvejerio fazės laikas parenkamas taip:

$$\tau = \max(t_X / 2, t_D) \qquad (12)$$

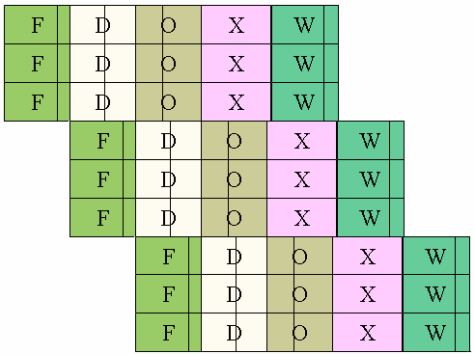
Kaip ir konvejerio atveju, superkonvejeryje neišvengiamai susiduriama su kliūtimis (valdymo, duomenų ir kt.). Kadangi iš karto vykdoma dvi ir daugiau komandų, vienos komandos uždelsimas (kai nepataikoma į spartinančią atmintinę, duomenų srautų valdymo vėlavimas) mažai turi įtakos kitų komandų fazių vykdymui. Takto užlaikymas didina bendrą ciklo atlikimo laiką, todėl ilgiau užtrunkama atliekant kiekvieną komandą. 10 paveikslėlyje pateiktame pavyzdyje fazių padaugėja beveik du kartus, bet praktinis pagerėjimas yra žymiai mažesnis negu 33%.



10 pav. Komandų vykdymas superkonvejeryje

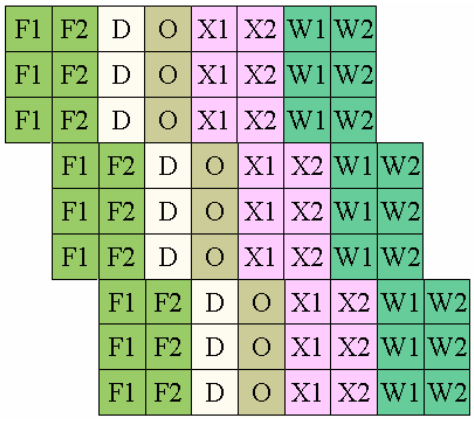
2.1.3.1 Superskaliariškumas ir VLIW procesoriai

Prieš tai buvusiame skyriuje aptarėme konvejerizuotus ir superkonvejerizuotus procesorius. Konvejerizuoti procesoriai daug našesni už procesorius, kuriuose komandos vykdomos nuosekliai, be to juose realizuoti konvejeriniai komandų fazių vykdymo mechanizmai. Superskaliariniuose procesoriuose į blokus jungiamos ir vykdomos iškart kelių komandų fazės.



11 pav. Komandų vykdymas superskaliariniame procesoriuje

Kaip matome iš 11 paveikslėlio, gali būti atliekamos iškart 3 komandos, pvz vieną su sveikaisiais skaičiais, vieną su trupmeniniais skaičiais ir vieną atminties operaciją. Pridėjus papildomų funkcinių įtaisų galima pasiekti, kad procesorius vykdytų dvi sveikojo arba dvi trupmeninio skaičiaus operacijas. Pirmasis procesorius, pagamintas pagal superskaliariškumo principą, buvo IBM POWER1. Netrukus visos RISC architektūros procesorius gaminančios įmonės perėjo prie superskaliariškumo realizavimo savo produktuose (SuperSPARC, ALPHA 21064). Taipogi buvo pagalvota ir apie superkonvejerį bei keleto komandų vykdymo realizavimą, kad procesorius būtų tuo pačiu metu ir superkonvejerizuotas, ir superskaliarinis:

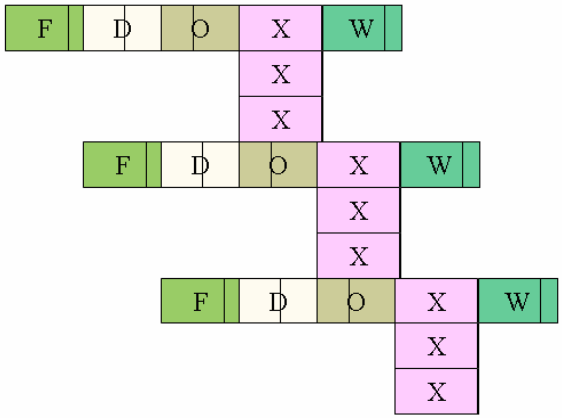


12 pav. Superkonvejerizuoto – superskaliarinio procesoriaus komandų vykdymas

Šiandien beveik visi procesoriai yra superkonvejerizuoti – superskaliariniai, tik dėl trumpesnio pavadinimo vadinami tiesiog superskaliariniais. Dabartinių procesorių konvejerio

plotis gali būti nuo dviejų (MIPS R5000) iki trijų (PowerPC G3/G4, Pentium Pro/ II / III) ar keturių (UltraSPARC, MIPS R10000, Alpha 21164) komandų.

VLIW (*Very Long Instruction Word*) procesoriai pasižymi tuo, kad jų komandų rinkiniai išskaidomi į grupes tam, kad tas grupes būtų galima vykdyti lygiagrečiai. Skaidymas į grupes išsprendžia duomenų pirklausomybių problemą jas inicijuojant, tokius procesorius lengviau projektuoti bei jie teoriškai turėtų geriau išnaudoti procesoriaus laiką. Iš tikrųjų šio procesoriaus komandos yra mažų komandų grupės, bet jos yra labai ilgos (dažnai esti 128 bitų ir ilgesnės), be to kiekviena komanda laiko savyje informaciją lygiagretiems skaičiavimams.



13 pav. VLIW procesoriaus komandų vykdymas

VLIW procesoriai turi ir ryškių trūkumų, pavyzdžiui nepataikymo į spartinančiąją atmintinę metu konvejeris yra stabdomas, nes procesoriuje nėra komandų priklausomybės tikrinimo. Tokiu atveju kompiliatorius turi įterpti pakankamą kiekį komandų prieš priklausomybę, o jei nėra ką įterpti, tuščius procesoriaus taktus užpildyti NOOP operacija. Komercinio pasisekimo šio tipo procesoriai neturi, nebent galima paminėti Intel IA64 architektūrą. Ji yra pavadinta EPIC (*Explicitly Parallel Instruction Computing* – aiškiai išreikšti lygiagretieji skaičiavimai), bet tai nieko daugiau kaip VLIW procesorius su gudriu komandų grupavimu ir predikacija (žr. 2.1.8 skyrių) [7].

2.1.4 Perėjimų prognozė

Konvejerizuotuose procesoriuose būtina užtikrinti efektyvų konvejerio darbą, t.y., visi konvejerio etapai turi būti užpildyti, tačiau ne visada tai pavyksta padaryti. Problema atsiranda tada, kai vykdomi nuoseklumo reikalaujantys sakiniai: *if*, *loop*, *procedure*. Šių sakinių vykdymo metu konvejeris užlaikomas tiek, kiek laiko reikia sužinoti, kokias komandas reikės vykdyti. Perėjimų prognozę iliustruokime pavyzdžiu:

```
if (x > 0) {
    a=0;
```

```
    b=1;
    c=2;
}
d=3;
```

2 lentelė. Nuoseklus komandų vykdymas žinant šaką

Taktai	Išrinkti	Dekoduoti	Vykdyti	Išsaugoti
1	if (x>0)			
2	a=0	if (x>0)		
3	b=1	a=0	if (x>0)	
4	c=2	b=1	a=0	if (x>0)
5		c=2	b=1	a=0
6			c=2	b=1
7				c=2

Jei x daugiau už 0, konvejeryje užkrautos komandos turi būti vykdomos, ir *if* sakinys bus pilnai įvykdomas.

3 lentelė. Komandų vykdymas be perėjimų prognozavimo

Taktai	Išrinkti	Dekoduoti	Vykdyti	Išsaugoti
1	if (x>0)			
2	a=0	if (x>0)		
3	b=1	a=0	if (x>0)	
4	d=3	išstumti b=1	išstumti a=0	if (x>0)
5		d=3	išstumti b=1	išstumti a=0
6			d=3	išstumti b=1
7				d=3

Jei x mažiau arba lygu 0, tada *if* sakinys nebus vykdomas. Šiuo atveju sekanti komanda turėtų būti *d=3*, o komandų, susijusių su *a*, *b* ir *c* iš vis neturėtų būti konvejeryje. Tos komandos turi būti atšauktos ir pašalintos iš konvejerio. Sakykime, kad taip pavyko padaryti, tada komanda *d=3* bus užkrauta 4 cikle, kai bus žinomas x > 0 lyginimo rezultatas. Šiuo atveju konvejeris nėra efektyvus, nes kai kurie etapai nebus išnaudojami teisingų komandų vykdymui. Prireiks 7 ciklų norint užbaigti šias dvi komandas.

4 lentelė Komandų vykdymas prognozuojant

Taktai	Išrinkti	Dekoduoti	Vykdyti	Išsaugoti
1	if (x>0)			
2	d=3	if (x>0)		
3		d=3	if (x>0)	
4			d=3	if (x>0)
5				d=3

Jei tik pavyktų pažvelgti į ateitį ir sužinoti, koks bus lyginimo rezultatas, būtų įmanoma konvejerį laikyti pastoviai užpildytą. Pavyzdžiui, jei būtų žinoma, kad lyginimo rezultatas bus neteisingas, kompiuteris galėtų iškart užkrauti komandą *d=3*, vietoj komandos *a=0*. Jei būtų taip, šios dvi komandos užimtų tik 5 ciklus [8].

Perėjimų prognozės idėja yra nuspėti į priekį, pagal kokią šaką bus vykdomos komandos. Jei spėjimas teisingas, konvejeris išlieka užpildytas. Jei spėjimas neteisingas, tenka stabdyti konvejerį ir jį užpildyti komandomis iš tikrosios krypties, kas mažina konvejerio efektyvumą. Pasinaudojant perėjimų prognoze įmanoma pasiekti iki 90% teisingų spėjimų. Galimi spėjimo tipai: a) statiniai spėjimai; b) dinaminiai spėjimai. Statiniai spėjimai gali būti paremti:

- prognozėmis, kad perėjimo niekada nebus;
- prognozėmis, kad perėjimas bus visada;
- atsižvelgiant į operacijos kodą.

Yra žinoma, kad apie 60% iš valdymo perdavimo pirmyn komandų iš tiesų reikalauja pereiti pirmyn, o net 85% grįžimo atgal komandų iš tiesų reikalauja grįžti. Tokia informacija apie perėjimų pobūdį gali būti panaudota perėjimams prognozuoti.

Dinaminiuose spėjimuose panaudojami keli spėjimo būdai: jungiklis „perėjimas bus/ perėjimo nebus“; perėjimų istorijų lentelė, perėjimų komandų spartinančioji atmintis ar grįžimo adresų stekas. Perėjimų istorijos lentelėje kiekvienam perėjimui skiriama po 2 bitus. Prognozuojama kryptis pakeičiama į priešingą, jei du kartus prognozės nepasitvirtino. Teigiama, kad naudojant 4096 įrašų lentelę prognozės efektyvumas siekia nuo 82% iki 99%. Kai kuriose sistemose naudojama perėjimų komandų spartinančioji atmintinė. Kai sutinkama perėjimo komanda, procesorius patikrina atmintinę ir pradeda komandų išrinkimą pagal atmintinėje esančią informaciją. Prognozės komandų atmintinė dirba maždaug taip:

- kiekvieno perėjimo atveju kreipiamasi į atmintinę;
- jei įrašo atmintinėje nėra, komanda išrenkama iš eilės;
- jei įrašas atmintinėje yra, perėjimo adresas nustatomas pagal prognozę, remiantis būsenos bitais;
- būsenos bitai modifikuojami atsižvelgiant į prognozės rezultatą;
- jei prognozė nepasitvirtino, išrinkimo logika pataiso adresą kitam išrinkimui;
- jei sąlyginio perėjimo komandai įrašo atmintinėje nėra, suformuojamas naujas įrašas, išmetant vieną iš buvusių.

Perėjimų komandų spartinančioji atmintinė turi du privalumus: a) komanda išrenkama iš atmintinės ir nesikreipiama į atmintį; b) jei numatomos šakos komanos jau yra atmintinėje, pati šakojimosi komanda keičiama tomis komandomis neprarandant laiko nuostolių konvejeryje.

Dinaminiams spėjimams dažniausiai panaudojama jau programos vykdymo metu sukaupta informacija.

2.1.5 Registrų pakeitimas

Praeitame skyriuje aptarėme vieną iš daugelio priežasčių, dėl kurių gali būti stabdomas procesoriaus konvejerio darbas. Tokių priežasčių yra ir daugiau. Viena iš jų – programinė duomenų ir rezultatų priklausomybė (duomenų konfliktas). Ji atsiranda tada, kai komandų negalima vykdyti lygiagrečiai, nes vienos komandos rezultatas yra kitos operandas. Programiškai duomenų priklausomybės išspręsti negalima, bet vardų priklausomybė gali būti pašalinta. Vienas iš galimų sprendimo būdų yra parūpinti tiek registrų, kiek tik galima. Problema ta, kad naujų registrų pridėjimas reiškia, kad reikės perprojektuoti visą architektūrą. Kita problema – kuo daugiau registrų, tuo daugiau darbo juos apdorojantiems procesams (išsaugojimas ir atkūrimas). Architektūros, naudojančios tokią strategiją, turi keletą kartų daugiau registrų nei įprastos sistemos. Šie papildomi registrai nėra susieti su architektūros registrais, bet jei tik jų prireikia, išskiriami dinamiškai. Kitas sprendimo būdas – registrų pakeitimo buferio panaudojimas. Situaciją iliustruokime tokiu pavyzdžiu:

```
1.  MUL R2,R2,R3   ; R2 = R2 * R3
2.  ADD R4,R2,1    ; R4 = R2 + 1
3.  ADD R2,R3,1    ; R2 = R3 + 1
4.  DIV R5,R2,R4   ; R5 = R4 * R4
```

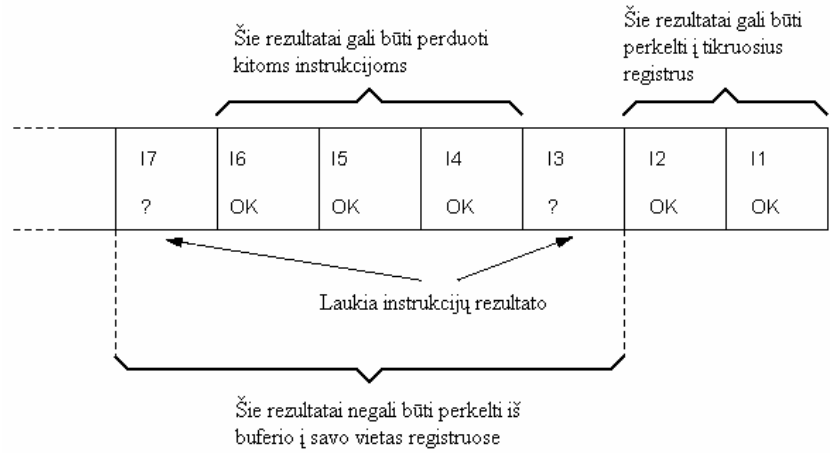
Šis kodo vykdymas parodo pačią problemą: 3 komanda negali būti vykdoma tol, kol nebus pilnai įvykdoma 1 komanda ir neprasidės 2 komandos vykdymas. Egzistuoja išvesties priklausomybė tarp 1 ir 3 komandos – abi rašo į R2, be to 3 komanda perrašo 2 komandos operandą. Pabandykime perrašyti kodą pakeisdami registrų pavadinimus:

```
1.  MUL R2_b,R2_a,R3_a
2.  ADD R4_b,R2_b,1
3.  ADD R2_c,R3_a,1
4.  DIV R5_b,R2_c,R4_b
```

Dabar 3 komanda gali būti vykdoma iškart, nes naudoja „kitą“ R2, skirtingai nuo 1 ir 2 komandų. Pateikiant rezultatą atsižvelgiama į registrų istoriją – R2_c naujausia R2 versija, prieš ją sekė R2_b, o dar anksčiau – R2_a. Kaip minėta anksčiau, yra du galimi būdai problemai spręsti. Pirmiausia apžvelkime registro pakeitimo buferį, po to pereisime prie alternatyvaus registrų pakeitimo.

Registrų pakeitimo buferis naudoja fizinį registrų failą, kuris yra tokio paties dydžio, kaip ir architektūrinių registrų failas, o duomenys yra statomi į eilę. Sakykime, kad mes vykdome *i*, *i+1*, *i+2* ir *i+3* komandas. Jos patalpinamos į pakeitimų buferį. Kai komanda

pasiekia buferio pabaigą, ji išimama ir įrašoma į architektūrinį registrą. Jei komandos rezultatas dar nežinomas, laukiama. Kadangi komandų vykdymo laikas skirtingas, o ir vykdomos jos ne taip, kaip suprogramuota, gali atsitikti, kad aukščiausios buferio komandos reikšmės vis dar nežinome, o tuo tarpu visos kitos komandos, einančios po jos, jau įvykdytos. Šiuo atveju visos komandos lieka buferyje, kol nebaigiama aukščiausioji komanda. Imkime tokį pavyzdį: turime 6 komandas – nuo I1 iki I6. Sakykime, kad I1, I2, I4, I5, I6 jau užbaigtos, o I3 dar neužbaigta. I1 ir I2 komandų rezultatus galime perkelti į architektūrinius registrus. I4, I5 ir I6 komandos turi laukti I3 komandos užbaigimo.



14 pav. Pakeitimų buferis

Taigi kai kurių komandų negalime perkelti į savo registrų vietas, bet jas vis dar galima naudoti skaičiavimuose. Tebūnie vykdykime komandą I7, kuri naudotų R2 kaip operandą. Sakykime, kad R2 operando reikšmę laikysime I5, ir tai yra reikšmė, kurios reikalauja I7 komanda. Kadangi I5 komandos reikšmės nebuvo perkelta į R2, ją galima naudoti I7 komandos skaičiavimuose. Seniausia registro versija saugoma architektūriniame reigstre, vidurinioji – kažkur buferio pabaigoje, jauniausioji – buferio pradžioje. Taip pakeitimo buferis efektyviai užtikrina registrų pakeitimą. Praktikoje nėra viskas taip paprasta. Registrų buferyje reikia saugoti papildomą informaciją – komandą, jos vietą eilėje, rezultato korektiškumą, be to labai svarbu šią informaciją pasiekti kaip galima greičiau. Norint išvengti konvejerio sustojimo, reikia kaip galima greičiau nustatyti, kada rezultatas, naudojamas kitoje komandoje kaip operandas, tampa prieinamas, ir kuo greičiau jį išrinkti.

Alternatyvus sprendimas pakeitimo buferiui – dinaminių registrų naudojimas. Bet kuriuo laiko momentu nustatoma, kurie dinaminiai registrai atstovaus architektūrinius registrus. Sakykime, sistemoje realizuota 16 architektūrinių registrų ir 64 dinaminiai registrai. Bet kuriuo laiko momentu tik 16 iš jų bus susieti su architektūriniais ir keisis programos

vykdymo metu. Čia lyg ir išsprendžiamos problemos, aptartos anksčiau, bet šis registrų pakeitimo metodas turi ir savų trūkumų: reikia papildomų taktų išsiaiškinti, kurie registrai vis dar reikalingi skaičiavimams; taipogi reikia nustatyti, kurie registrai atstovauja architektūrinius; atlikus komandas dinaminiuose registruose reikia rezultatus išsaugoti į tikruosius registrus, o kuriuos – vėl reikia papildomų taktų. Visa tai reikalauja papildomų konvejerio taktų, bet čia nėra kritinė metodo vieta, nes šiam darbui gali būti išnaudojamas laisvas konvejerio laikas [9].

2.1.6 Vykdymas dinamiškai parenkant paruoštas

Instukcijų eilės vykdymas konvejeriuose yra nuoseklus, o tai reiškia, kad jei vienos komandos vykdymas yra sustabdomas, tai sustoja ir kitų komandų, esančių eilėje, vykdymas. Taip neturėtų būti, nes jei po laukiančios komandos einančiosios neturi duomenų ar rezultatų priklausomybės, tai jos galėtų būti baigiamos vykdyti. Procesoriuose, kuriuose realizuotas dinamiškas komandų vykdymas, ši problema išspręsta taip:

1. Išrenkama komanda;
2. Ji nusiunčiama į komandų eilę (dar vadinamą komandų buferiu, komandų langu ar rezervavimo stotimi). Komandų buferis yra tarp F ir X konvejerio pakopų.
3. Komanda eilėje laukia tol, kol bus paruošti jos operandai. Tuomet jai leidžiama palikti eilę anksčiau tų komandų, kurios yra senesnės;
4. Komanda perduodama į vykdymo įrenginį ir įvykdoma;
5. Rezultatas statomas į eilę;
6. Tik tada, kai senesnių komandų rezultatai bus įrašyti į registrus, tada bus įrašytas ir šios komandos rezultatas.

Taigi procesoriaus konvejeris laisvą savo laiką užpildo tomis komandomis, kurioms jau yra paruošti rezultatai. Proceso gale rezultatai taipogi perrūšiuojami, todėl atrodo, kad komandos buvo vykdytos normalia tvarka. Šio metodo efektyvumas didėja didėjant konvejerio pločiui. Vykdymo dinamiškai parenkant paruoštas technikai realizuoti naudojamas registrų pakeitimas (žr. 2.1.5 skyrių). Rezultatų buferis yra būtinas norint išspręsti šakų vykdymo ar išimčių (spąstų) problemas. Ankstesniuose procesoriuose kaip rezultatų buferį naudota registro pakeitimo buferį, bet vėlesniuose procesoriuose pereita prie registrų žurnalų. Dabar visi šiuolaikiniai procesoriai naudoja dinaminį paruoštų vykdymą.

2.1.7 Sąlyginis komandų vykdymas

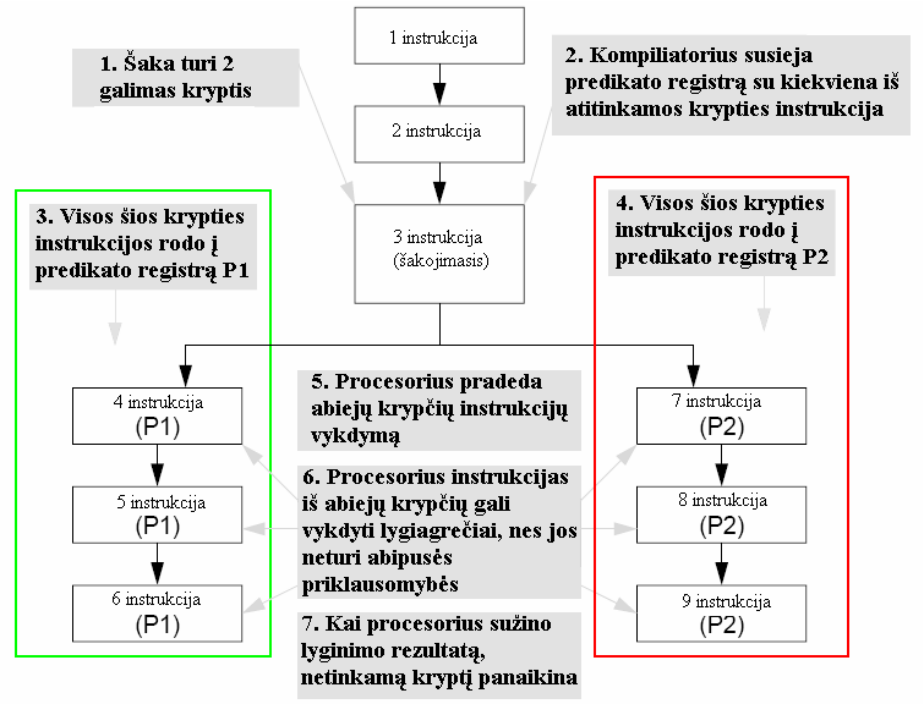
Sąlyginis komandų vykdymas – prognozuojamos sąlygos sakinio šakos vykdymas laukiant, kol bus atliktas loginis patikrinimas ir nurodyta vykdymo kryptis. Sąlyginį vykdymą pailiustruokime tokiu pavyzdžiu:

1. if (*sąlyga*) then
2. R3 = R1 + 1
3. else
4. R3 = R1 – 1

Šiame pavyzdyje matoma duomenų priklausomybė. Tai reiškia, kad procesorius pirma turės rasti *sąlygos* reikšmę, o tik po to paaiškės, ar R3 turės būti padidintas, ar sumažintas. Tuo metu konvejeris bus stabdomas, nes nežinoma, kokią šaką reikia vykdyti. Procesoriuje, kuriame realizuotas sąlyginis komandų vykdymas, konvejerio stabdyti nereikės, nes spėjimo būdu bus parinkta kuri nors viena šaka ir ji įvykdyta. Taigi sąlyginis vykdymas yra komandos vykdymas prieš nustatant, ar komanda bus reikalinga, ar nebus. Šakos spėjimas įtraukia perėjimo prognozavimą ir komandų iš pasirinktos šakos vykdymą. Imkime pavyzdį, kai dažnai vykdoma 2 komanda. Tuomet perėjimų prognozės mechanizmas nustato, kad reikia vykdyti 2 komandą dar neįvykdžius 1-os. 2-os komandos rezultatas patalpinamas komandų buferyje (komandų lange). Sąlyginis komandų vykdymas didina konvejerio našumą, nes 2 komanda vykdoma lygiagrečiai su komandomis tame pačiame bloke, esančiame dar prieš 1 komandą. Jei spėjimas buvo neteisingas, tokiu atveju komanda ir jos rezultatas yra pašalinami iš konvejerio. Sekančiame konvejerio etape užkraunama komanda su jau teisingais duomenimis. Jei dažnai pasikartoja blogos prognozės, nereikalingų komandų pašalinimas iš konvejerio užtrunka, o tai tik sumažina paties konvejerio našumą [10].

2.1.8 Predikacija

Skirtingai nei perėjimų prognozė, predikacija remiasi tuo, jog vykdymos abi sąlygos sakinio šakos vietoj to, kad pasirinkti ir vykdyti tik vieną. Šiuolaikiniai procesoriai bando spėti, kurią šaką vykdyti, o nepataikius sugaišta daug taktų teisingų komandų užkrovimui. Predikacija visiškai pakeičia perėjimų prognozę. Jos metu lygiagrečiai vykdomos abi sąlyginio sakinio šakos, o paaiškėjus lyginimo rezultatui nereikalingą šaką atmeta, todėl konvejerio papildomai stabdyti nebereikia. Patį predikacijos principą iliustruoja 15 paveikslėlis:



15 pav. Procesoriaus predikacijos schema

VLIW procesoriuose (tokiuose kaip Intel Itanium), kur vykdymo žodis yra pakankamai ilgas (128 bitai), galima sutalpinti kelias komandas ir rasti tokią vykdomų komandų kombinaciją, kad jos būtų pilnai išlygiagretintos:

128 bitų ilgio komandų žodžiai	1 Komanda	2 Komanda	3 Komanda (šakojimasis)
	4 Komanda (P1)	7 Komanda (P2)	5 Komanda (P1)
	8 Komanda (P2)	6 Komanda (P1)	9 Komanda (P2)

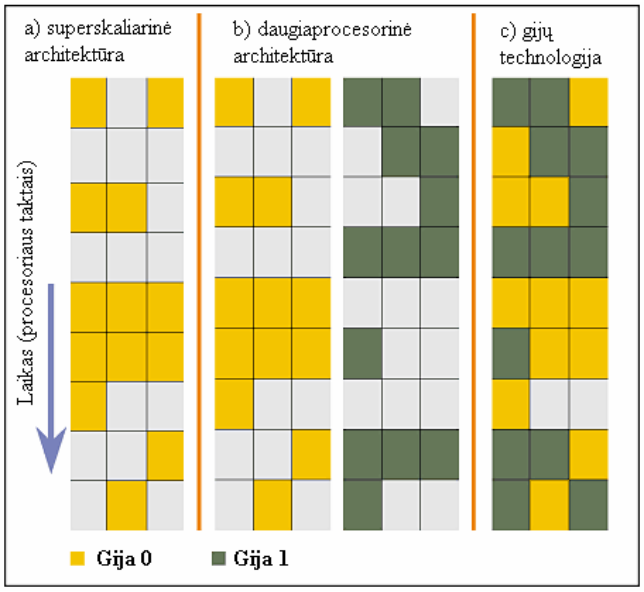
16 pav. Komandų išlygiagretinimas su predikacija

Kompiliatorius analizuoja programinį kodą ir aptikęs sąlygą jame nustato, ar šiam kodo segmentui galima naudoti predikaciją. Jei taip, kiekvienai krypčiai priskiria unikalų predikatą. Toliau analizuojama, ar komandos gali būti atliekamos lygiagrečiai, t.y., suporuojant komandas iš kiekvienos krypties. Abiejų šakų rezultatai neišsaugojami tol, kol nepaaiškėja, kurios šakos rezultatų reikia. Nereikalingos šakos rezultatai anuliuojami [11], [12].

2.1.9 Gijų technologija

Gijų technologija – tai procesoriaus duomenų srautų sinchronizavimo technologija, kuri palaiko komandų vykdymą per keletą duomenų sraugų viename fiziniame procesoriuje. Ši technologija pirmiausiai pritaikyta Intel Xeon MP procesoriuose. Gijų technologija yra paremta principu, kad kiekvienu laiko momentu komandų vykdymui naudojama tik dalis

procesoriaus resursų. Likę nepanaudoti resursai taipogi gali būti išnaudoti lygiagrečiam kitos programos (ar tiesiog tos pačios programos kitos dalies) komandų vykdymui. Toks resursų panaudojimas realizuotas procesoriuose, palaikančiuose gijų technologiją. Paprastai vienas fizinis procesorius suformuoja du loginius procesorius, kurie pasidalija fizinio procesoriaus resursus. Operacinė sistema ir programos „mato“ du procesorius ir atitinkamai paskirsto darbą tarp jų.



17 pav. Gijų technologija

Vienas iš gijų technologijos tikslų yra tas, kad procesoriuje būtų galima naudoti ir vieną duomenų srautą komandoms vykdyti. Procesoriui dirbant vieno srauto režimu loginis įrenginys išnaudoja visus fizinius resursus, kai tuo tarpu kitas loginis įrenginys stabdomas HALT komandomis. Procesoriui dirbant daugiau nei vieno srauto režimu resursai yra dalijami. Kiekvienas loginis įrenginys turi savo bendros paskirties, valdymo bei aptarnavimo registrus. Didžioji komandų dalis yra imama iš trasų spartinančiosios atmintinės, kurioje jos yra jau dekoduotos ir paruoštos vykdyti, o du loginiai įrenginiai jas pasiima pakaitomis, kas 1 taktą. Likusios komandos paruošiamos komandų dekodavimo įtaise, kuris yra bendras abiems loginiams įrenginiams, ir paduodamos taipogi pakaitomis kas 1 taktą. Visų lygių spartinančioji atmintinė (L1 ir L2) irgi yra bendra. Šioje technologijoje yra ir savų problemų: jei vienas loginis įtaisas laukia komandų, jis gali sunaudoti tam visus fizinio procesoriaus resursus ir kliudyti antrojo loginio įtaiso komandų vykdymui. Iškilus šitokiai situacijai procesoriaus našumas su įjungtu gijų palaikymu gali kristi iki 20% [13]. Norint šito išvengti, Intel rekomenduoja vietoj tuščių procesoriaus taktų naudoti PAUSE komandą (atsirado tik Pentium 4 procesoriuose).

2.2 Atmintis

Ne mažesnę svarbą kompiuterio našumui turi pagrindinė atmintis (RAM – *Random Access Memory*). Nuo atminties vidaus architektūros priklauso, kaip greitai bus išrinkti reikiami duomenys, o nuo atminties magistralės – kaip greit jie bus perduoti procesoriui. Yra daug atminčių tipų: FPM, EDO, BEDO, SRAM, SDRAM, DDR, DDR2, RDRAM ir kt. Visos jos pasižymi savita architektūra. Visos atmintys skirstomis į sinchronines ir asinchronines. Asnichroninės atmintys (FPM, EDO, BEDO) šiandien jau nebenaudojamos, todėl mes jų nebenagrinėsime. Tolesniuose skyriuose apžvelgsime tik sinchroninių atminčių rūšis: SDRAM, DDR SDRAM, DDR2 SDRAM bei RDRAM atmintis, t.y., tas, kurių darbo ritmas yra susietas su magistralės taktais.

2.2.1 Atminties kreipinių laikas

Prieš pradėdami kalbėti apie skirtingus atminčių tipus, šiek tiek apžvelkime bendrus principus, kaip yra organizuojamas sinchroninių atminčių (pvz. SDRAM) darbas. SDRAM ląstelės išdėstytos eilutėmis ir stulpeliais. Sakykime, kad turime tokius parametrus:

RAS – signalas, kuris atminčiai parodo, kad duotą adresą reikia priimti kaip eilutės adresą;

CAS – signalas, kuris atminčiai parodo, kad duotą adresą reikia priimti kaip stulpelio adresą;

t_{RCD} (*RAS-to-CAS delay*) – laikas tarp RAS ir CAS išrinkimo;

t_{RP} (*RAS Precharge*) – laikas, reikalingas pasiruošti kitos eilutės aktyvavimui;

t_{AC} – laikas, reikalingas pasiruošti sekančiam duomenų išvedimui;

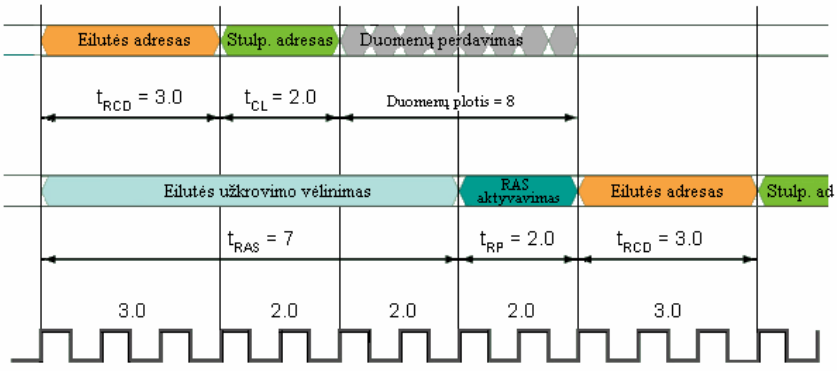
t_{CAC} – stulpelio išrinkimo laikas;

t_{CL} – CAS vėlinimas;

t_{CLK} – takto trukmė.

Kreipiantis į atmintį tam tikrų duomenų išrinkimui, pirmiausia perduodamas eilutės adresas (kartu su RAS). Specialiais stiprintuvais sužadinami visi eilutės kondensatoriai. Po tam tikro laiko (t_{RCD}) pasiunčiamas stulpelio adresas (kartu su CAS). Atmintis sužadina tą eilutės stulpelį, kuris buvo nurodytas. Po CAS vėlinimo (t_{CL}), duomenys yra perduodami į atminties išvesties įrenginį. Po duomenų nuskaitymo stulpelius ir eilutes reikia grąžinti į jų pradinę būseną. Atmintis reikalauja papildomų taktų (t_{RP}) tolesniam duomenų nuskaitymo pasiruošimui. Aktyvavus eilutę galima išrinkti keletą stulpelių adresų ir nusiųsti jų turinius į išvesties įrenginį, kol vis dar aktyvuota eilutė. Tai yra greitesnis būdas nei kiekvienai vietai

naudoti pilną eilutės-stulpelio išrinkimo procedūrą. 18 paveikslėlyje pavaizduota atminties vėlinimų schema.



18 pav.Vėlinimai atmintyje

Sakykime, kad turime 100 MHz dažniu veikiančią SDRAM atmintį, kurios vieno ciklo vėlinimas – 10ns (nanosekundžių). Atminties vėlinimas nusakomas trimis skaičiais: pirmas – CAS vėlinimu, antras – pasiruošimu kitos eilutės aktyvavimui (t_{RP}), trečias – laiku tarp RAS ir CAS išrinkimo (t_{RCD}).

5 lentelė. Kreipinių laiko skaičiavimas 100 MHz dažniui

t_{CAC} = 25 ns	25 / 10 = 2.5 – apvaliname iki 3	**3 – 2 – 2**
t_{RP} = 20 ns	20 / 10 = 2	
t_{RCD} = 20 ns	20 / 10 = 2	

Paskaičiuokime tuos pačius vėlinimus atminčiai, veikiančiai 133 MHz dažniu. Vienas taktas – 7,5ns.

6 lentelė. Kreipinių laiko skaičiavimas 133 MHz dažniui

t_{CAC} = 25 ns	25 / 7,5 = 3,33 – apvaliname iki 4	**4 – 3 – 3**
t_{RP} = 20 ns	20 / 7,5 = 2,67 – apvaliname iki 3	
t_{RCD} = 20 ns	20 / 7,5 = 2,67 – apvaliname iki 3	

Iš pavyzdžio aišku, kad didėjant atminties dažniui didėja jos vėlinimai, o tuo pačiu didėja ir laiko sąnaudos duomenų išrinkimui iš pagrindinės atminties [14].

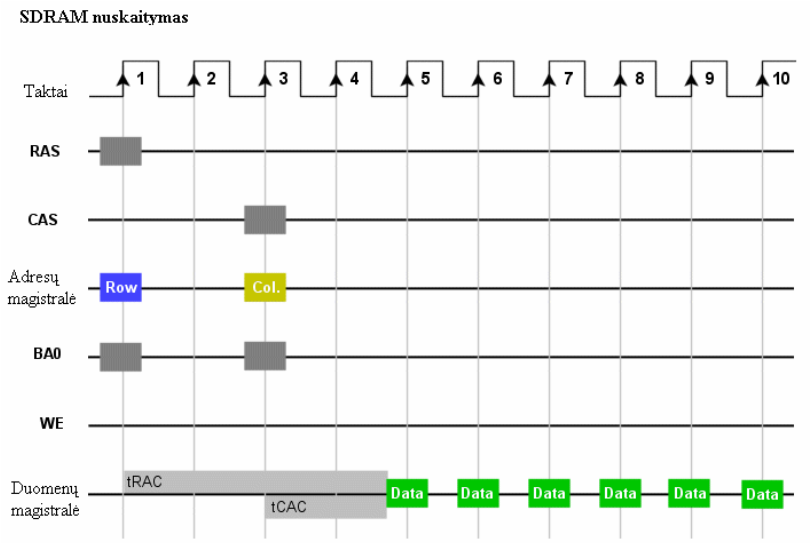
2.2.2 SDRAM atmintis

SDRAM – sinchroninio DRAM tipo atmintis. Tai pirmoji atmintis, kurios darbo taktai yra susieti su magistralės (kartu ir su procesoriaus) sinchronizacija. DRAM tipo atmintis yra organizuota taip, kad kiekvienas bankas užpildytų duomenimis magistralę. Skirtingai nei SIMM, DIMM tipo atmintis turi daugiau kontaktų, todėl užtenka vieno banko visai magistralei užpildyti. Be to, vienoje DIMM plokštėje gali būti realizuota keletas bankų.

Įsivaizduokime, kad turime SDRAM modulį su dviem bankais. Kol vienas bankas naudojamas, kitas bankas gali būti ruošiamas duomenų nuskaitymui. Kai norima nuskaityti duomenis iš kito banko, jis jau pasiruošęs ir papildomai laukti nebereikia. Dažnai dviejų bankų SDRAM moduliuose naudojamas papildomas kontaktas signalui, kurį banką naudoti. Jei signalo lygis žemas, naudojamas bankas0, jei aukštas – bankas1. Modulio skirstymas į bankus nėra vienintelis SDRAM privalumas. Kadangi atmintis yra susieta su magistralės taktais, laisvų taktų metu ji gali priimti valdymo komandas. Tokios komandos yra:

- CS (*Chip Select*) – SDRAM DIMM moduliai turi po keletą atminties bankų. Gaudamas šį signalą valdiklis žino, kad ateinančios komandos skirtos būtent jam
- RAS (*Row Address Strobe*) – eilutės išrinkimas
- CAS (*Column Address Strobe*) – stulpelio išrinkimas
- WE (*Write Enable*) – signalas įrašymui sinchronizuoti
- ir kt.

Faktas yra tas, kad procesoriai spartėja daug greičiau, nei atmintis. Dėl tos priežasties procesoriai turi užpildyti savo tuščius ciklus NOOP komandomis, kol gaus reikalingus duomenis. Kad sumažinti NOOP komandų skaičių, sistemų projektuotojai SDRAM atmintyje realizavo sistemą, kai vienos eilutės išrinkimo metu aktyvuojama keletas stulpelių, o duomenys perduodami paketais, kaip pavaizduota 19 paveikslėlyje:



19 pav. Duomenų nuskaitymas iš SDRAM atminties

Toks duomenų išrinkimas vadinamas paketiniu. Dar įdomu ir tai, kad bet kokios komandos SDRAM atmintyje atliekamos pagal kylantį taktinio impulso frontą (paveikslėlyje pažymėta rodyklėmis).

2.2.3 DDR SDRAM atmintis

DDR DRAM atmintis savo architektūrinėmis savybėmis niekuo ypatingai nesiskiria nuo paprastos SDRAM atminties, o tiesiog yra patobulintas jos variantas. Dabar duomenys yra siunčiami tiek kylančio, tiek ir krintančio taktinio impulso fronto metu. Taigi DDR gali perduoti du žodžius per vieną taktą, todėl jos nuskaitymas ar įrašymas į ją yra optimizuotas ir dvigubai greitesnis. Neveltui DDR yra pažymėta kaip dvigubas duomenų dažnis (*Double Data Rate)*. DIMM skaidymas į bankus turi savų privalumų. Atminkime, kad viename banke gali būti aktyvi tik viena eilutė. Sakykime, kad reikia duomenų iš kitos eilutės, tada 1) reikia paruošti naują eilutę; 2) uždaryti aktyvią eilutę; 3) atidaryti skaitymui jau naują eilutę. Visa tai reikalauja daug laiko (taktų), todėl stengiamasi kiekvieną atidarytą eilutę išlaikyti kiek galima ilgiau. Tuo atveju bus sutaupoma laiko kreipiantis į skirtingų adresų stulpelius, nes eilutės adreso kartoti jau nebereikės. Taigi kuo daugiau aktyvių eilučių, tuo greičiau bus išrinkti duomenys iš atminties ir perduoti procesoriui. Ir jei tam tikru momentu banke gali būti tik viena aktyvi, tai reiškia, kad norint turėti daugiau aktyvių eilučių reikia turėti daugiau bankų.

2.2.4 DDR2 SDRAM atmintis

DDR2 nuo DDR skiriasi keletu dalykų. Visų pirmiausia tai kontaktų skaičiumi. Paprastas DDR turi 184 kontaktus, o DDR2 – 240. DDR2 tikslas yra ne padidinti duomenų pralaidumą, bet tobulinant signalų technologiją pratęsti vidinio dažnio kėlimą. Esminiai DDR2 patobulinimai:

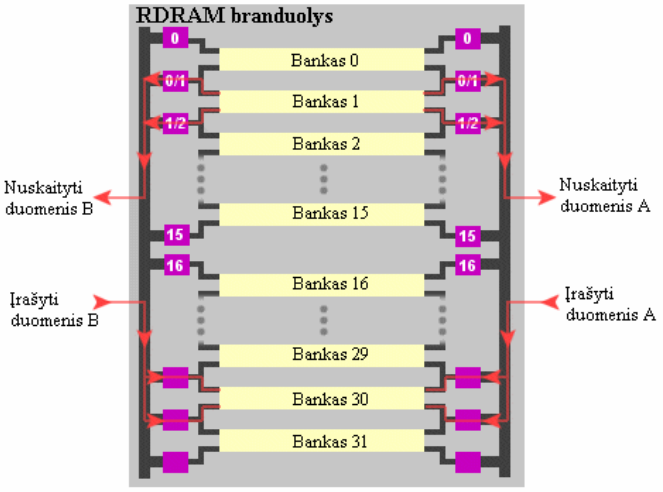
- duomenų išrinkimo dažnio padidėjimas,
- duomenų bankų skaičiaus didėjimas,
- vėlinimų keitimas,
- BGA (*Ball Grid Array*) pakavimo technologijos pritaikymas,
- kt.

Pasiekus 200 MHz dažnį, DDR atminties konvejeriai yra nebeužpildomi duomenimis. Išeitis buvo vėl grįžti prie 100 MHz, bet šį kartą, siekiant geriau išnaudoti duomenų srautus, nuspręsta padvigubinti aptarnaujančių įvesties ir išvesties įrenginių darbo dažnį. Atmintys veikia 100 MHz vidiniu dažniu, bet ląstelės paduodamos 4 taktais į išvesties ir įvesties įrenginius, kurie jau dirba 200 MHz dažniu. Taigi duomenų išrinkimas jau nebe 2 bitai, o 4 bitai, nes bankų skaičius DDR2 atmintyje padidėja iki 4. Galiausiai į magistralę duomenys patenka kylančių ir krintančių taktinio impulso frontų metu, bet jų dažnis yra dvigubai didesnis.

DDR ir DDR2 atmintys skiriasi ne tik šiais patobulinimais. Paskutinio modulio bloke yra integruotas reikalingas terminatorius signalų slopinimui, kad išvengti signalų atspindžio reiškinio. DDR2 atmintyje toks terminatorius integruotas į patį modulį, o DDR atveju – pagrindinėje plokštėje. Taip pat komandų pakavimui naudojamos kitos charakteristikos, bet minėti šios atminties patobulinimai tiesiogiai neįtakoja našumo, todėl plačiau jie nebus nagrinėjami.

2.2.5 RDRAM atmintis

RDRAM – RAMBUS DRAM atmintis, sudaryta lygiai iš tokių pačių atminties ląstelių, kaip ir DRAM atmintis, bet požiūris į duomenų nuskaitymą ir įrašymą yra visiškai skirtingas. Pati idėja yra ta, kad kiekvienas RDRAM valdiklis gali aptarnauti iki 32 atminties bankų, o patys bankai yra atskiri ir savarankiški. Kiekvienas bankas yra sujungtas su kaimynystėje esančiu banku. Dažniausiai RDRAM yra 16 bankų. Atrodytų, kad vienu metu gali būti atviri visi 16 atminties bankai, bet iš tikrųjų atvirų gali būti tik pusė – 8.



20 pav. RDRAM atminties organizacija

Kai nuskaitomas 16 bitų blokas, pusė duomenų keliauja kairiąja magistrale, pusė – dešiniąja. Architektūriškai šios abi magistralės apibrėžiamos kaip viena 16 bitų. Iš paveikslėlio galime matyti, kad duomenų perdavimas iš 1 banko vyksta per kaimyninių bankų linijas, todėl pastarieji tuo pačiu momentu duomenų perduoti negali. Išimtis yra tik 0, 15, 16, ir 31 bankams, nes jie turi tik po 1 kaimyną, todėl jiems reikia laukti iki šie baigs duomenų siuntimą. Įsivaizduokime RIMM atminties modulį su keturiais valdikliais, kuris kiekvienas valdo 32 atminties bankus. Vienu metu iš 128 bankų atvirų gali būti tik pusė, kas mažina RDRAM našumą.

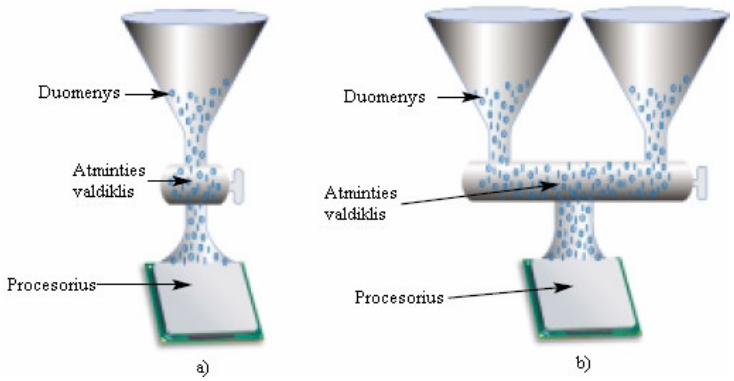
Duomenų išrinkimui RDRAM naudoja panašią schemą, kaip ir DRAM. Skirtumas tas, kad DRAM tipo atmintyje tiek eilučių ir stulpelių adresai, tiek duomenys paduodami per tą

pačią magistralę. RDRAM atveju adresai keliauja tam skirtais kontaktais ir visiškai netrukdo duomenų perdavimui.

RDRAM architektūroje skiriasi duomenų perdavimo magistralėje principas. DRAM architektūrose moduliai sujungti viena magistrale, o duomenys perduodami iš kiekvieno modulio pakaitomis. RDRAM tai būtų didelis greičio praradimas. 16 bitų valdiklio magistralė yra dalis didesnio, paskirstyto RAMBUS kanalo, kuriame yra valdymo, adresavimo ir duomenų linijos (viso 30 impulsais suderintų linijų). Kanale duomenys keliauja per kiekvieną RDRAM modulį kol pasiekia atminties valdiklį, o kanalo gale nutraukiami specialiu rezistoriumi. Magistralė, dirbdama 400 MHz dažniu, naudoja minimaliai signalų informacijai perduoti, bet ji irgi turi trūkumų. Kanalo ilgis ribotas (iki 12 cm), o linijų skaičius atbaido gamintojus plėtoti našumo didinimą iki keleto kanalų, nes tam reikia papildomos vietos plokštėje ir finansinių sąnaudų linijoms [15].

2.2.6 Dviejų kanalų technologija

Dviejų kanalų (*Dual Channel*) technologija pradėta taikyti DDR atminčiai dar palyginti visai neseniai. Terminas „du kanalai“ taikomas tik tiems DDR tipo pagrindinių plokščių valdikliams, kurie suprojektuoti dirbti su dviem atminties kanalais. Dviejų kanalų palaikymas geriau išnaudoja atmintį ir sumažina visos sistemos vėlinimą. Sakykime, kad vienas valdiklis skaito ir rašo duomenis, o tuo metu kitas pasiruošia sekančiam kreipiniui taip pašalindamas pauzes, kurios atsiranda, kol tas pats vienas modulis vėl paruošiamas duomenų skaitymui ar rašymui. Valdiklio vieno kanalo pralaidumas – 64 bitai. Dviejų kanalų technologija duomenų kanalą praplatina dvigubai – iki 128 bitų, o tai reiškia, kad procesorius gauna dvigubai duomenų per tą patį laiko vienetą. Lygiai tas pats procesas galioja ir atvirkšiniam duomenų perdavimui iš procesoriaus į pagrindinę atmintį [16].



21 pav. Atminties valdiklio technologija:
a) viengubo kanalo; b) dvigubo kanalo

2.3 Magistralės

Magistralė – duomenų perdavimo linijos nuo vieno įrenginio iki kito. Kompiuterio komponentai į vieną sistemą susieti magistralėmis duomenų apsikeitimui realizuoti. Pagrindinės magistralės kompiuteryje yra trys:

- PCI (*Peripheral Component Interconnect*) - magistralės, reikalingos bendrauti su papildomais įrenginiais kaip garso, tinklo plokštėmis, modemais, video imtuvais ir pan.;
- AGP (*Accelerated Graphics Port*) – šią magistralę naudoja tik video plokštės, todėl kompiuteriuose ši magistralė yra tik viena. Turi tiesioginį priėjimą prie sistemos atminties;
- FSB (*Front Side Bus*) – sisteminė magistralė, jungianti procesorių su pagrindinės atminties valdikliu ir turinti daugiausia įtakos kompiuterio našumui.

Kiekviena magistralė gali būti nusakyta sekančiais parametrais:

- magistralės pločiu – duomenų linijų skaičiumi;
- magistralės dažniu – maksimalia duomenų perdavimo sparta.;
- magistralės pralaidumu – maksimaliu teoriniu perduodamų duomenų kiekiu per vieną sekundę.

Toliau savo darbe aptarsiu tik tas magistrales, kurios įtakoja kompiuterio našumą.

2.3.1 FSB

„FSB magistralė“ terminas dažniausiai suprantamas kaip procesoriaus duomenų magistralė. Būtent per šią magistralę duomenys iš procesoriaus siunčiami į kitus kompiuterio įrenginius. Magistralėje duomenys perduodami signalais. Paprasčiausias būdas yra signalą perduoti per vieną taktą. SDRAM atmintis naudoja kaip tik šitokį duomenų perdavimo būdą. DDR atmintis yra du kartus greitesnė, todėl magistrale keliauja dvigubai daugiau duomenų. Tokiu atveju duomenų skaitymas ar įrašymas atliekamas pagal kylantį ir krintantį taktinio impulso frontą. Lygiai taip pat dirba ir RDRAM atmintis. DDR2 technologija naudoja jau keturių signalų perdavimą per taktą. Sakykime, kad DDR atmintyje perduodami 2 signalai per taktą, o vidinis dažnis 200 MHz. Gaunamas 400 MHz magistralės dažnis – 400 milijonų signalų per sekundę. Teorinis tokios magistralės pralaidumas – 3,2 GB/s (400 MHz * 8 bitų = 3200 MB/s = 3,2 GB/s). Jei turime du DDR kanalus, tada bendras teorinis magistralės pralaidumas – 6,4 GB/s (400 MHz * 8 bitų * 2 kanalai = 6400 MB/s = 6,4 GB/s). Esant dideliems dažniams, magistralėse atsiranda signalų triukšmai, kurie neigiamai veikia sistemos

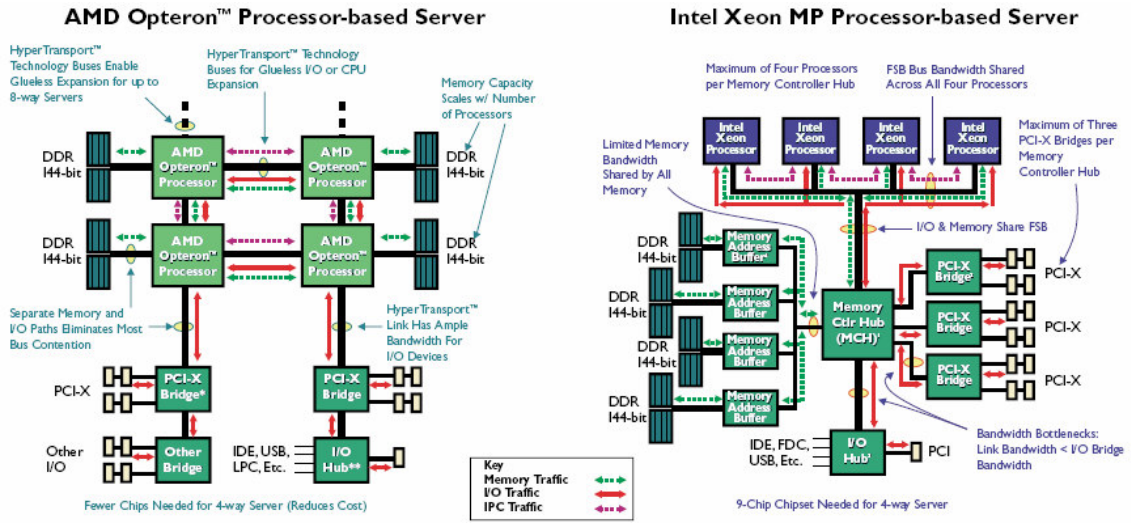
našumą. Dabar naudojamų 32 bitų sistemų FSB magistralės plotis – 64 bitai, o 64 bitų sistemų – 128 bitai.

2.3.2 HyperTransport

HyperTransport yra dar pakankamai nauja technologija. Kūrėja – AMD, įsteigusi HyperTransport konsorciumą, kuris prižiūri šios technologijos plėtojimą. Tai dvikryptė, nuosekli/lygiagreti, didelio pralaidumo, mažo vėlinimo kompiuterių magistralė. HyperTransport gali veikti 200 – 1400 MHz dažniu bei gali būti naudojama DDR atminčių magistralėms, kurios veikia dviem impulso frontais per taktą. Tokiu atveju maksimalus teorinis magistralės pralaidumas gali siekti nuo 100 MB/s iki 22,4 GB/s (daug greitesnis nei dabar naudojami standartai). HyperTransport magistralės plotis – nuo 2 iki 32 bitų.

Ši magistralė yra paremta duomenų perdavimu paketais (žodžiui – 32 bitai). Pirmas žodis pakete – visada komanda. Jei paketas sudarytas iš adreso, paskutiniai 8 bitai susiejami su sekančio žodžio 32 bitais ir taip sudaromas 40 bitų blokas. Pastaruosiuose 32 bituose perduodami duomenys. Dėl šių savybių HyperTransport magistralę galima naudoti ne tik tarp procesoriaus ir atminties, bet ir kitose vietose.

HyperTransport yra lankstesnė ir greitesnė magistralė. Jos tikslas – pakeisti FSB. NVIDIA nForce valdikliai HyperTransport tecnologiją jau šiandien naudoja šiaurinio ir pietinio tiltų sujungimui. Palyginkime sistemas, paremtas FSB ir HyperTransport technologijomis:



22 pav. HyperTranspor™t ir FSB skirtumai [17]

Intel Xeon MP architektūros sistemose bendravimas tarp procesorių ir kitų įrenginių realizuotas per FSB magistralę. Kaip matome iš 22 paveikslėlio, naudojant FSB magistralę (dešinėje) gali susidaryti duomenų kamščiai ten, kur jų reikėtų vengti. Visa tai mažina

našumą, nes procesoriams reikia laukti, kol atsilaisvins duomenų kanalas. Sistemose su AMD Opteron procesoriais ir HyperTransport technologija šių problemų yra išvengiama, nes kiekvienas procesorius turi savo atminties modulius, o duomenų komunikavimui tarp procesorių naudojama sparti magistralė. Galima drąsiai teigti, kad HyperTransport – ateities magistralė.

2.4 Našumo įvertinimas

Ankstesniuose skyriuose buvo aptarti kompiuterio komponentai, kurie turi įtakos visos sistemos našumui. Norint įvertinti skirtingų gamintojų, skirtingų architektūrų realizacijos ypatybes, dažniausiai naudojamasi testais, kurie išryškina tyrinėjamų sistemų komponentų privalumus. Pagal kilmę testai gali būti skirstomi taip:

- Gamintojų testai – sukurti pačių gamintojų ir pernelyg orientuoti į konkrečią architektūrą, tik į to gamintojo produktų našumo įvertinimą.
- Standartiniai testai – organizacijos, kuriančios testavimo platformas ir jas naudojančios kaip pagrindą našumui įvertinti. Šiuos testus kuria nepriklausomi specialistai, turintys autoritetą kompiuterių pasaulyje. Tokios testavimo platformos gali būti: SiSoft Sandra, 3D Mark, WinBench, SPEC, LINPACK ir kt.
- Vartotojų testai – juos kuria garsios kompiuterių firmos, besispecializuojančios įvairių kompiuterinių technologijų diegimo srityje, arba panašios krypties vartotojų grupės, todėl šie testai orientuoti į kompiuterių ir programinės įrangos, geriausiai tinkančių konkretiems taikymo uždaviniams spręsti, parinkimą.

Testai dažniausiai būna skirtingiems įrenginiams arba skirtingiems kompiuterio posistemiams testuoti. Šiame darbe nagrinėjami bus tik tie testai, kuriais galima įvertinti procesoriaus, atminties ar FSB magistralės našumą. Vaizdo posistemio, tinklo ar periferinių įrenginių testai nebus aptariami.

Procesorių našumui testuoti naudojami įvairių tipų testai:

a) dirbtiniai (sintetiniai) testai – operacijos su skaičiais ir eilutėmis. Šio tipo testuose operacijos su sveikaisiais ir turpmeniniais skaičiais įvertinamos atskirai, kad nustatyti, kaip sparčiai dirba ALU ir FPU įrenginiai. Tam naudojami Dhrystone – sveikųjų skaičių, ir Whetstone – trupmeninių skaičių algoritmai.

   Dhrystone algoritmas, R.P. Wecker pasiūlytas 1984 metais, dar ir dabar naudojamas kaip vienas iš labiausiai paplitusių algoritmų sveikųjų skaičių įrenginių našumui testuoti. Dėl savo kodo paprastumo buvo perrašytas ir panaudotas kitose nei x86

architektūrose. Šis testas pirmiausia pasirodė ADA kalba, vėliau buvo perrašytas C kalba.

Whetstone algoritmas, parašytas H. Curnow. Šio algoritmo tikslas – nustatyti FPU įrenginio greičio įvertinimą trupmeninių skaičių skaičiavimuose. Pirmoji versija – ALGOL programavimo kalba 1972 metais. FORTRAN versija išėjo po metų – 1973 [18].

Šie abu algoritmai naudoja paprastą būdą – aritmetinius veiksmus, bet turi ir tam tikrų trūkumų. Pirmiausia, jie neįvertina spartinančiosios atmintinės našumo. Šiandieninėse sistemose, kai spartinančioji atmintinė yra procesoriaus dalis, ji turi labai didelę reikšmę kompiuterio spartai. Antra, kompiliatoriai gali būti parašyti gamintojų taip, kad optimizuotų testų rezultatus siekiant užsitikrinti savo gaminiais didesnę rinkos dalį. Trečia, šiuose testuose neįvertinami jokie kiti kompiuterio komponentai kaip atminties greitis, magistralių pralaidumas, kurie gali turėti svarios įtakos kompiuterio našumui. Na, ir aišku, šie testai jau pakankamai pasenę, todėl dabar naudojamos jau modifikuotos jų versijos. Programiniai paketai: SiSoft Sandra, WinBench, PCMark ir pan.

b) Programiniai testai – realaus pasaulio uždavinių sprendimas. Šio tipo testuose taipogi dominuoja uždaviniai tiek sveikiesiems, tiek trupmeniniams procesoriaus įrenginiams. Kaip šių tipų testų kūrimo atstovą paminėsime SPEC organizaciją. CPU2000 išleistame pakete yra du posistemiai: CINT2000 – sveikųjų skaičių testai; CFP2000 – testai trupmeniniams skaičiams. CINT2000 rinkinį sudaro 11 programų C kalba ir 1 programa C++ kalba. Šio rinkinio uždaviniai: šachmatų figūros ėjimo apskaičiavimas, duomenų suspaudimas, atliekamas tik atmintyje, teksto apdorojimas, darbas su specifinėmis duomenų bazėmis ir kiti uždaviniai. CFP2000 rinkinyje iš viso 14 programų: 6 parašytos FORTRAN-77, 4 – FORTRAN-90 ir 4 – C programos. CFP2000 rinkinyje sprendžiami uždaviniai, kuriems reikia daug trupmeninių skaičiavimų: vandens paviršiaus modeliavimas, branduolinio greitintuvo projektavimas, oro temperatūros bei taršos sklidimo uždavinys, vaizdų apdorojimas ir kiti panašaus pobūdžio uždaviniai [19]. CPU2000 pakete gauti rezultatai lyginami su etaloniniu vienetu – darbo stotimi Sun Ultra 5/10 (300 MHz UltraSPARC II procesorius). Žinoma, tokio pobūdžio testų yra ir daugiau.

c) Matematiniai testai – matematinių lygčių sistemų sprendimas. Šio tipo testuose dažniausiai sprendžiamos tiesinės lygčių sistemos kol randamas sistemos sprendinys. Įvertinamas sugaištas laikas lygčių sistemos sprendinui rasti. Tokio tipo

testus naudoja LINPACK paketas, kurį pasiūlė J. Dongarra. Nuo 1980 metų jis naudojamas kaip populiariausias mokslinių programų rinkinys superkompiuterių našumui nusakyti. Pati pirmoji versija parašyta FORTRAN, bet vėliau perrašyta C kalba. Standartinė C kalbos versija sprendžia 100x100 tiesinių lygčių sistemą dvigubu tikslumu. Kompiuterių našumui įvertinti laiko atžvilgiu naudojamos metrikos:

- MIPS (milijonai komandų per sekundę)

$$MIPS = \frac{N}{T * 10^6} = \frac{N}{N * CPI * \tau * 10^6} = \frac{N * dažnis}{N * CPI * 10^6} = \frac{dažnis}{CPI * 10^6} \qquad (13)$$

Čia

N – įvykdytų komandų skaičius;

T – sugaištas laikas;

CPI – ciklų skaičius, reikalingas komandai įvykdyti;

τ – takto periodas.

Šia metrika pavojinga naudotis, nes ji turi trūkumų:

I. neįvertinamas vykdomų komandų skaičius N;
II. MIPS metrika negali būti naudojama kompiuterių, turinčių skirtingas komandų sistemas, lyginimui;
III. Kadangi neįvertinamas vykdomų komandų skaičius, MIPS metrika negali būti naudojama skirtingų programų tame pačiame kompiuteryje lyginimui;
IV. MIPS skaičiavimo metrika gali neigiamai veikti našumą.

- MFLOPS (milijonai slankiųjų skaičių operacijų per sekundę)

$$MFLOPS = \frac{F}{T * 10^6} \qquad (14)$$

Čia

F – slankiųjų skaičių operacijų skaičius;

T – sugaištas laikas;

Naudojant šią metriką reikia turėti omenyje, kad:

I. skirtingi kompiuteriai turi skirtingus slankaus kablelio operacijų rinkinius;
II. skirtingos slankaus kablelio operacijos atliekamos per skirtingą laiką.

Norėdami suvienodinti MFLOPS skaičiavimą Livermoro ciklų testų autoriai pasiūlė įvesti tokius parametrus: add = 1; mult =2; div =4; func(sin, cos) = 8 ir t.t. LINPACK testai naudoja būtent MFLOPS metriką.

Taigi, apžvelgėme pačius svarbiausius kompiuterio našumo didinimo aspektus. Verta pastebėti, kad ne visi procesoriai turi tokias našumo didinimo savybes, kaip apžvelgta prieš tai esančiuose skyriuose, pvz. predikaciją turi tik 64 bitų Intel Itanium ir Itanium2 procesoriai. Be to Intel serveriams gaminamuose procesoriuose realizavo ir L3 spartinančiąsias atmintines, kurių dydis gali siekti nuo 1,5 iki 9 MB. Taipogi šiuolaikiniuose procesoriuose darbui paspartinti naudojami ne vienas, bet keli funkciniai įtaisai , pvz. AMD Athlon procesoriuose naudojami trys trupmeninių ir trys sveikųjų skaičių superskaliariniai, pilnai konvejerizuoti funkciniai įtaisai. Jie paremti vykdymu dinamiškai parenkant paruoštas bei jau turi savyje MMX ir 3DNnow! komandų rinkinius. Prie viso šito realizuotas pažangus dinaminės perėjimų prognozės įtaisas dar labiau padidina našumo rodiklius. Intel taip pat neatsilieka sukurdama gijų technologiją ir pritaikydama ją Xeon ir Pentium 4 procesoriuose. Maža to, Prescott serijoje Intel įnešė žymių patobulinimų, pvz. L1 ir L2 atmintinių talpos padidinamos 2 kartus (L1 – 16 KB, L2 – 1 MB), spartinančiųjų atmintinių asociatyvumo laipsnis padidintas iki 8 krypčių. Taip pat konvejeris buvo pailgintas nuo 20 pakopų iki 31, nors neteisingos perėjimo prognozės atveju atsiranda didesni konvejerio perkrovimo laiko nuostoliai. Čia Intel projektuotojai padirbėjo ir pagerino perėjimų prognozės įtaiso tikslumą. Atminčių gamintojai taipogi nesnaudžia ir pasiūlo vis naujus technologinius sprendimus. Perėjimas nuo SDRAM iki DDR, o vėliau ir prie DDR2, leido atminties posistemę paspartinti keletą kartų. Netgi dviejų kanalų technologija šiuolaikinėse sistemose yra ypač dažnai sutinkamas dalykas. Atskirų komponenčių įtaką bendram sistemos našumui įvertinti yra labai keblu, nes tarp jų egzistuoja labai glaudus ryšys. Atsiriboti nuo kurios nors komponentės neįmanoma, nes tai pažeistų Fon Noimano kompiuterio struktūrą, kuria yra paremti visi šiuolaikiniai kompiuteriai. Taigi, belieka džiaugtis, kad kompiuterių mokslas nestovi vietoje, ir pasiūlo vis naujų technologinių sprendimų kompiuterių našumo didinimui.

3 Kompiuterių našumo tyrimas

Apie našumą ir kompiuterio komponentų reikšmę buvo kalbama 2 skyriuje. Kaip jau minėta anksčiau, informaciniame pasaulyje našumas suprantamas kaip sistemos produktyvumo rodiklis per laiko vienetą. Kadangi vienareikšmiškai sistemos našumo nusakyti negalima, pabandysime tirti atskiras sistemos komponentus taip įvertindami kiekvienos komponento įtaką visos sistemos darbui. Tyrimus atliksime su trijomis skirtingomis sistemų architektūromis bandydami sulyginti, kiek teorinėje dalyje aptartos ypatybės gali praktiškai įtakoti bendrą sistemos darbą. Sistemų techninės charakteristikos pateiktos 1 priede. Kadangi testuojamos sistemos nepriklauso labai naših architektūrų grupei (vartotojų kompiuteriai), testams pasirinkome SiSoftware (http://www.sisoftware.co.uk) kompanijos sukurtą Sandra2004 paketą. Šis paketas patogus tuo, kad jo pagalba galima ištestuoti visus nagrinėjamus kompiuterio komponentus ir jų pagrindines savybes: procesoriams – sveikųjų ir trupmeninių skaičių skaičiavimo įtaisus, spartinančiai artmintinei ar atminčiai – pralaidumą, duomenų paketinį perdavimą ir pan. Taip pat pasinaudosime CPU-Z v1.28.6 (http://www.cpuid.org/cpuz.php) programa informacijai apie testuojamus procesorius išgauti. Dėl finansinių sumetimų 64 bitų procesorinių sistemų (Xeon, Itanium, Opteron, Athlon MP) gauti nepavyko, todėl tolimesniuose tyrimuose šios sistemos nebus nagrinėjamos.

3.1 Procesoriaus dažnio ir spartinančiosios atmintinės įtaka našumui

Šių dienų informacinėje visuomenėje taktinis dažnis dar neparodo tikrų procesoriaus pajėgumų. Tikrosios procesoriaus savybės išryškėja atliekant bandymus ir eksperimentus. Šio darbo eksperimentuose dalyvauja trys procesoriai:

- Intel® Pentium® 4 3,00 GHz su gijų technologija,
- AMD Athlon™ XP 2500+,
- AMD Sempron™ 2500+.

Šių procesorių tyrinėjamos charakteristikos pateikiamos 7 lentelėje (detalesnę informaciją rasite 2 priede). Iš lentelės matome, kad tiek Intel, tiek AMD spartinančiosioms atmintinėms realizuoti naudoja iš dalies asociatyvų tipą, tik skiriasi krypčių skaičius. Pasak Intel, 8 krypčių atmintinė pasižymi geresniu pataikymo dažniu vykdant „LOAD“ ir „STORE“ komandas. Kuo AMD firma argumentuoja krypčių skaičiaus spartinančiosiose atmintinėse pasirinkimą nėra žinoma. Remiantis teorinėmis prielaidomis galima spėti, kad kuo didesnė atmintinės talpa, tuo sudėtingesnis išrinkimas, todėl mažesnis krypčių skaičius užtikrintų mažesnį vėlinimą. Krypčių skaičiaus įtaką našumui įvertinti būtų sudėtinga, nes reikėtų

pasirinkti identiškus procesorius (ir jų sistemas), kad skirtųsi tik krypčių skaičius spartinančiojoje atmintinėje. Įvertinant finansinę pusę tai padaryti yra pakankamai keblu, todėl tolimesniuose tyrimuose krypčių skaičiaus įtakos našumui nenagrinėsime. Procesoriaus tyrimuose bus atliekami taktinio dažnio, spartinančiosios atmintinės bei, kurie procesoriai leis, gijų technologijos eksperimentai.

7 lentelė. Pagrindinės tiriamų procesorių charakteristikos

Procesorius	Intel® Pentium® 4	AMD Athlon™ XP 2500+	AMD Sempron™ 2500+
Taktinis dažnis	3 GHz	1,8 GHz	1,75 GHz
L1 spartinančioji atmintinė	16 KB 8 krypčių	64 KB 2 krypčių	64 KB 2 krypčių
L2 spartinančioji atmintinė	1024 KB 8 krypčių	512 KB 16 krypčių	256 KB 16 krypčių
Gijų technologija	Taip	Ne	Ne
FSB magistralės dažnis	200 MHz	166 MHz	166 MHz
Sisteminės magistralės dažnis	800 MHz	333 MHz	333 MHz

3.1.1 Taktinis dažnis

Patyrinėkime minėtų procesorių taktinių dažnių didinimo galimybes. Vertėtų prisiminti, jog taktinis dažnis matuojamas gigahercais (GHz) ir gaunamas FSB magistralės dažnį padauginus iš atitinkamo daugiklio, pvz, minėtų procesorių taktiniai dažniai yra:

Intel Pentium4: 200 MHz * 15 = 3000 MHz = 3 GHz

AMD Athlon XP: 166 MHz * 11 = 1826 MHz = 1,8 GHz

AMD Sempron: 166 MHz * 10,5 = 1754 MHz = 1,75 GHz

Toks yra normalus šių procesorių taktinis dažnis. Atlikime testus sveikųjų (ALU) ir trupmeninių skaičių (FPU) įrenginiams:

8 lentelė. Normalus taktinis procesorių darbo dažnis

Procesorius	**Taktinis dažnis**	**ALU (MIPS)**	**FPU (MFLOPS)**
Intel Pentium4	3 GHz	7741	3635
AMD Athlon XP	1,8 GHz	6813	2810
AMD Sempron	1,75 GHz	6590	2718

Procesorių taktinį dažnį padidinkime apie 10%. Atlikime tuos pačius testus. Po eksperimento buvo gauti tokie rezultatai:

9 lentelė. Procesorių darbo dažnis padidintas apie 10 %

Procesorius	Taktinis dažnis	Spartos padidėjimas	ALU (MIPS)	FPU (MFLOPS)
Intel Pentium4	3,30 GHz	10%	8596	4014
AMD Athlon XP	2,00 GHz	11%	7469	3091
AMD Sempron	1,96 GHz	12%	7463	3082

Įvertinkime taktinį dažnį imdami santykinį šių procesorių našumą:

10 lentelė. Santykinis procesorių našumo padidėjimas

Procesorius	Taktinis dažnis	Spartos padidėjimas	ALU (MIPS) pagerėjimas	FPU (MFLOPS) pagerėjimas
Intel Pentium4	3,30 GHz	10%	11%	10,4%
AMD Athlon XP	2,00 GHz	11%	9,6%	10%
AMD Sempron	1,96 GHz	12%	13,2%	13,4%

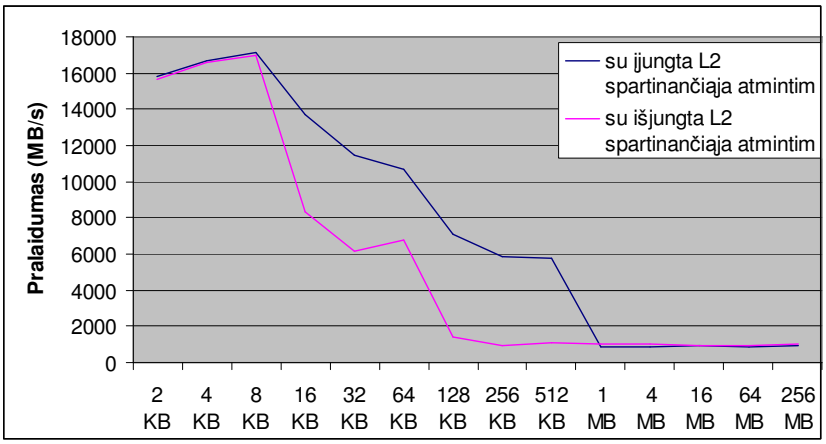
Iš 10 lentelės matome, jog aukščiausią spartos rodiklį parodė AMD Sempron procesorius, kurio darbo našumas su trupmeniniais skaičiais padidėjo net iki 13,4%. Tokį našumo padidėjimą galima sieti su šiek tiek didesniu spartos padidinimu kitų sistemų atžvilgiu.

3.1.2 Spartinančioji atmintinė

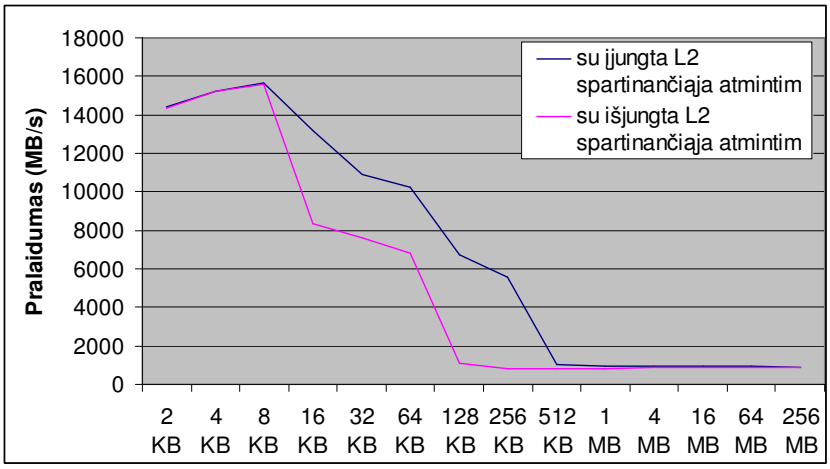
Kadangi kai kurių kompiuterių BIOS nustatymai leidžia išjungti L1 ir L2 spartinančiąsas atmintines, buvo nuspręsta atlikti eksperimentus jas išjungiant ir stebint, kaip tai paveiks sistemos darbą. Intel Pentium 4 procesoriaus sistemoje įvertinti spartinančiosios atmintinės įtakos nepavyko, nes BIOS nustatymuose nebuvo rasta jos įjungimo ar išjungimo galimybė. Toliau pateikti eksperimentų rezultatai tik su AMD Athlon XP ir AMD Sempron procesoriais.

3.1.2.1 L2 spartinančiosios atminties įtaka sistemos našumui

Ištestuokime sistemą išjungę L2 spartinančią atmintinę. BIOS nustatymuose *Advanced BIOS features* pakeiskime *CPU L2 Cache* reikšmę iš *Enabled* į *Disabled*. Gauname štai tokius rezultatus:



23 pav. AMD Athlon XP sistemos atminties pralaidumas



24 pav. AMD Sempron sistemos atminties pralaidumas

Kadangi L2 spartinančiosios atmintinės išjungimas įtakoja kreipinių į pagrindinę atmintį skaičių, dėl to sumažėja duomenų pralaidumas, t.y. per tą patį laiko vienetą perduodama mažiau duomenų dėl pagausėjusių kreipinių. Kitas svarbus faktorius – atsako laikas, nes pagrindinės atminties duomenų išrinkimas yra žymiai lėtesnis negu spartinančiosios atmintinės, todėl laiko faktorius yra labai svarbus matuojant duomenų pralaidumą. 3 priede pateikti L2 atmintinės testavimų rezultatai. Pastebėta, kad su įjungta L2 spartinančiaja atmintine bendras atminties pralaidumas AMD Athlon XP sistemoje padidėja 38%, o AMD Sempron – 30%. Spartinančioji atmintinė testuota naudojant normalų šių procesorių taktinį dažnį (žr. 3.1.1 skyrių).

3.1.2.2 L1 spartinančiosios atminties įtaka sistemos našumui

Nuo sistemos atjunkime L1 spartinančiąją atmintinę. Norint tą padaryti BIOS nustatymuose *Advanced BIOS features* pakeiskime *CPU L1 Cache* reikšmę į *Disabled*. Nei vienos iš tyrinėjamų sistemų nebuvo sulaukta užsikraunant. Tai galime paaiškinti tuo, kad

kreipiantis į L1 spartinančiąją atmintinę visi kreipiniai traktuojami kaip nepataikyti, bei nukreipti į L2 spartinančiąją atmintinę. Kadangi išjungta ir ši atmintinė, kreipiniai ir į ją traktuojami vėl kaip nepataikyti. Tenka kreiptis į pagrindinę atmintį. Paskaičiuokime laiko sąnaudas atsižvelgdami į realių sistemų parametrus. Sakykime, kad vykdome ADD operaciją. Visoms komandoms atlikti reikia 8 procesoriaus taktų (žr. 2.1.2.2 skyrių). Imkime, kad duomenims iš L1 išrinkti reikia 1 procesoriaus takto, iš L2 – 5 taktų, o iš pagrindinės atminties –50 taktų. Tebūnie L1 pataikymas – 95%, o L2 – 80%. Procesoriaus vieno takto periodas – 0,55 ns. Laiko sąnaudoms paskaičiuoti pasinaudokime 7 formule:

$$T_{su\ spart.} = 8*(2+0.95+0.05*(5+0.2*50))*2 = 8*(2+1.7)*0.55 = 16{,}28 \approx 17\,ns$$

$$T_{be\ spart.} = 8*(2+0+1*(0+1*50))*2 = 8*(2+50)*0.55 = 228{,}8 \approx 229\,ns$$

Šiame pavyzdyje nėra įvertinti signalų perdavimo vėlinimai, todėl galima drąsiai teigti, kad šie skaičiai tėra teoriniai. Matome, kad toms pačioms 8 komandoms atlikti spartinančioji atmintinė laiko sąnaudas sumažina 13,5 karto. Įvertinus tai, kad procesoriuje vykdoma milijonai komandų, gauname, kad šis skaičius išauga daug kartų. Į viską pažiūrėkime šiek tiek kitaip. Sakykime, kad kompiuteris su įjungtomis spartinančiosiomis atmintinėmis be papildomų kliūčių užsikrauna per 3 minutes. Tada tas pats kompiuteris išjungus abiejų lygių spartinančiąsias atmintines užsikraus per 41 minutę, o įvertinus signalų vėlinimą šis laikas dar pailgėtų.

3.1.3 Gijų technologijos tyrimas

Kadangi gijų technologiją sukūrė Intel ir panaudojo ją Xeon bei Pentium 4 kartos procesoriuose, plačiau ją patyrinėsime imdami turimą Pentium 4 sistemą. Testuojama sistema turi galimybę per BIOS parametrus gijų technologiją išjungti arba įjungti. Testavimo metodika paprasta – pirmiausia testą atliekame su įjungta gijų technologija, vėliau tą patį testą pakartojame jau išjungę gijų palaikymą. Eksperimentai atliekami naudojant fiksuotą taktinį dažnį, nustatant kitų sistemos komponenčių reikšmęs į numatytąsias ir testo metu jos nekeičiamos.

11 lentelė. Gijų įtaka Pentium 4 procesoriaus našumui

Taktinis dažnis	ALU (MIPS)	FPU (MFLOPS)
3 GHz be gijų	7741	3635
3 GHz su gijomis	8065	6121
3,30 GHz be gijų	8596	4014
3,30 GHz su gijomis	8807	6710

Įvertinkime santykinį šių rodiklių padidėjimo koeficientą:

12 lentelė. Santykinis gijų technologijų našumo padidėjimas

Taktinis dažnis	ALU (MIPS) pagerėjimas	FPU (MFLOPS) pagerėjimas
3 GHz su gijomis	4,2%	68,4%
3,30 GHz su gijomis	2,5%	67,2%

Iš 12 lentelės galima įžvelgti tai, jog gijų technologija labai paspartina darbą atliekant operacijas su trupmeniniais skaičiais (iki 68,4%), o operuojant su sveikaisiais skaičiais tik 4,2%. Vadinasi, gijų technologija naudinga ten, kur atliekamas darbas reikalauja daug aritmetinių operacijų su trupmeniniais skaičiais, t.y. vaizdų apdorojimas, fizikinių reiškinių modeliavimas ir pan. Taip yra todėl, kad trupmeninių skaičių operacijoms atlikti reikia daugiau papildomų komandų ir laiko, pvz. sinuso ir kosinuso operacijos atliekamos pasitelkiant kelių pakopų aritmetines skaičiavimo formules. Tokiu atveju galimas aritmetinių veiksmų išlygiagretinimas gijomis, kai atskiri uždaviniai vykdomi dviejuose loginiuose procesoriuose, o sveikųjų skaičių komandos yra pakankamai trumpos procesoriaus laiko atžvilgiu, todėl naudojama gijų technologija jų skaičiavimams neturi didelės įtakos.

3.1.4 Perėjimų prognozė ir registrų pakeitimas

Perėjimų prognozavimas šiuolaikiniuose procesoriuose yra pakankamai tikslus - siekia 90% ir daugiau. Deja, nėra priemonių, kurios leistų nustatyti perėjimų prognozavimo tikslumą imant vieną kažkurį procesorių. Tai galima įvertinti nebent teoriškai. Sakykime, procesorius atlieka 10^6 komandų. Imkime idealų atvejį, kai vienai komandai įvykdyti reikia vieno takto. Tyrimais nustatyta, kad konvejeryje vidutiniškai kas šešta komanda yra perėjimas. Vadinasi, perėjimų prognozavimas atliekamas apie 16% visų komandų. Laikykime, kad perėjimų prognozavimo įtaiso tikslumas – 92%. Perėjimų prognozavimo įtaisui atlikus neteisingą spėjimą konvejeris bus stabdomas ir turės laukti teisingų duomenų. Sakykime, kad toks konvejerio perkrovimas užtrunka 10 taktų. Paskaičiuokime laiko sąnaudas:

Viso perėjimų: $10^6 * 0{,}16 = 160\,000$

Taktų skaičius, reikalingas atlikti esant teisingam perėjimų prognozavimui:

$$160\,000 * 0.92 = 147\,200$$

Taktų skaičius, reikalingas konvejeriui perkrauti po neteisingų perėjimų prognozavimų:

$$160\,000 * 0.08 * 10 = 128\,000$$

Visoms komandoms įvykdyti su perėjimų prognozavimu procesoriaus taktų reikės:

$$840\,000 + 147\,200 + 128\,000 = 1115\,200$$

Dabar imkime atvejį, kai procesoriuje nerealizuota perėjimų prognozė. Tuomet konvejeris galėtų vidutiniškai įvykdyti 5 komandas laukdamas perėjimo reikšmės. Vadinasi, laiko nuostoliai būtų 5 taktai kiekvienai perėjimo komandai, o bendras komandų atlikimo laikas taktais:

$$840\,000 + 800\,000 = 1640\,000$$

Įvertinus šių skaičių santykį gauname, kad teorinė perėjimų prognozės įtaka procesoriaus našumui idealiu atveju – 47%. Prisiminkime, kad dauguma komandų atliekamos per skirtingą procesoriaus taktų skaičių, o ir jų vykdymą įtakoja kiti faktoriai, kaip vykdymas dinamiškai parenkant paruoštas, registrų pakeitimas, gijų technologija, praktiškai šio skaičiaus negalėtume laikyti perėjimų prognozės našumo rodikliu.

Registrų pakeitimo technologijos įtakos neįmanoma ištestuoti standartinėmis testavimo priemonėmis. Pačių gamintojų nėra numatyta, kad ši procesorių savybė gali būti išjungta. Tai yra daugiau architektūriniai ypatumai ir jų realizacijos nėra prieinamos vartotojui. Net ir teoriniams spėjimams trūksta duomenų, kaip pvz. kiek registrų iš viso naudojama konkrečioms komandoms atlikti, kiek jų yra pakeičiama, koks registrų pakeitimo įvertis laiko atžvilgiu ir pan.

3.2 Atminties tyrimas

Tiriant kompiuterio atminties įtaką reikia įvertinti tai, kad ji yra sinchronizuota ir didžiąja dalimi jos darbas priklauso nuo procesoriaus. Pačios atminties darbo specifikai turi įtakos jos tipas, eilutės ir stulpelio išrinkimo laikai bei duomenų perdavimas. Šiame skyriuje bus tiriamas kreipinių į atmintį reikšmingumas sistemos darbui. Taip pat bus įvertintas dvigubo kanalo technologijos našumas. Duomenų perdavimo technologijų ypatumai tyrinėjami 3.3 skyriuje.

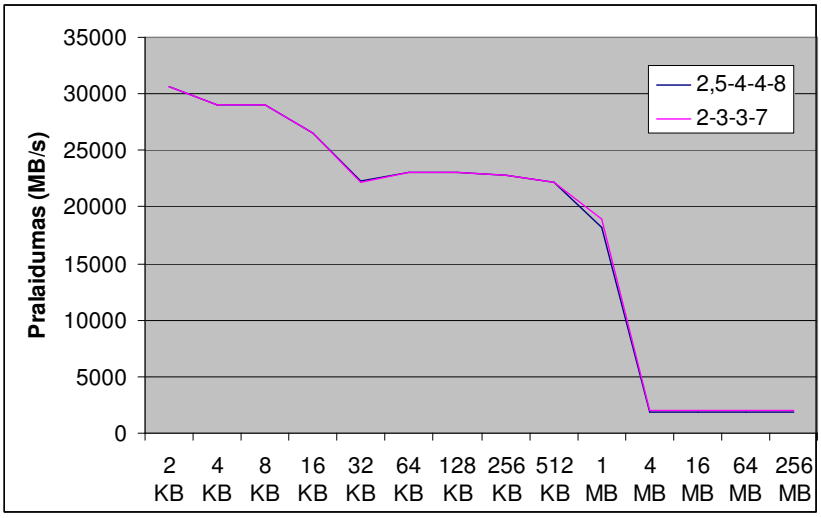
3.2.1 Išrinkimo laiko iš atminties įtaka našumui

Teoriniai kreipinių į atmintį aspektai laiko atžvilgiu buvo aptarti 2.2.1 skyriuje. Dabar patyrinėsime praktinę šių kreipinių įtaką našumui. Kiek leidžia techninės galimybės, pabandysime pakaitalioti jų reikšmes ir stebėti sistemos darbą. Bus tyrinėjamos visos 3 anksčiau minėtos sistemos, o jų techninės charakteristikos pateiktos 2 priede. Vėlgi sistemas tyrinėsime esant normaliam taktiniui dažniui. Toliau kiekvienai sistemai nusistatykime tokias atminties kreipinių reikšmes, ir su jomis atlikime testus:

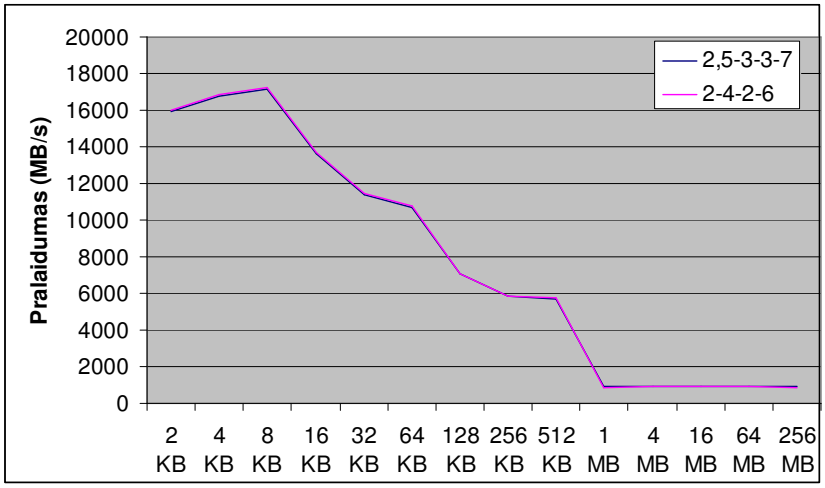
13 lentelė. Atminties kreipinių nuostatos

Procesorius	Taktinis dažnis	Normalus kreipinių laikas (iš SPD)	Sumažintas kreipinių laikas
Intel Pentium4	3 GHz	2,5 – 4 – 4 – 8	2 – 3 – 3 – 7
AMD Athlon XP	1,8 GHz	2,5 – 3 – 3 – 7	2 – 4 – 2 – 6
AMD Sempron	1,75 GHz	2,5 – 4 – 4 – 8	2 – 3 – 3 – 7

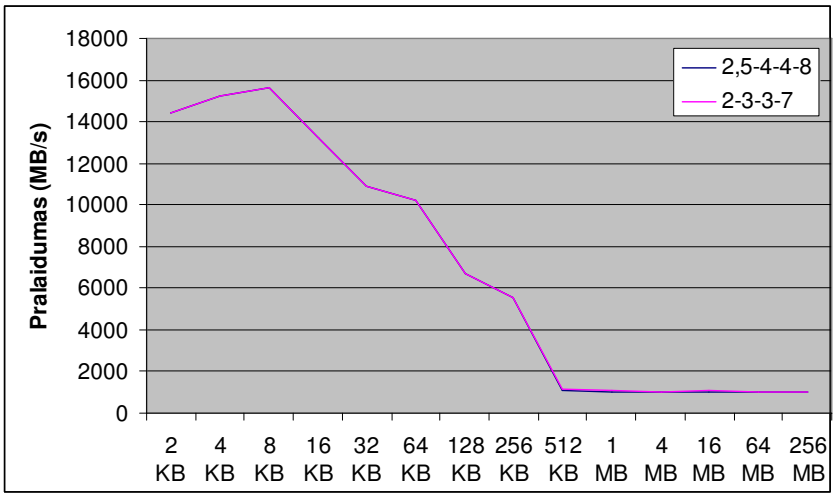
Atliekant testus reikia žinoti, kad atminties kreipinių laiko mažinimas yra ribotas, o atminties modulis turi palaikyti nustatytą kreipinių laiką. Įvertinkime atminties duomenų pralaidumą esant tokioms kreipinių laiko reikšmėms:



25 pav. Atminties kreipinių laiko įtaka Intel Pentium4 sistemos našumui



26 pav. Atminties kreipinių laiko įtaka AMD Athlon XP sistemos našumui



27 pav. Atminties kreipinių laiko įtaka AMD Sempron sistemos našumui

Kaip matome 25, 26 ir 27 paveikslėliuose, kiekvienos iš šių sistemų atminties magistralės pralaidumas po kreipinių laiko sumažinimo pagerėjo nežymiai. Santykinis pagerėjimo rodiklis Intel Pentium 4 – 0,5%, AMD Athlon XP – 0,4%, AMD Sempron – 0,2%. Pagerėjimą galima paaiškinti tuo, kad sunaudojama mažiau procesoriaus taktų duomenų išrinkimui, t.y. sutaupomas laikas, o per tą patį laiko vienetą galima perduodi didesnį duomenų kiekį.

3.2.2 Dviejų kanalų technologijos įtaka našumui

Valdikliai, dirbantys dviejų kanalų technologija, sugeba dirbti ir paprastu vieno kanalo režimu. Pabandykime paeksperimentuoti, kiek dviejų kanalų pralaidumo valdikliai našesni už vieno kanalo. Testuokime Intel Pentium 4 sistemą ir patyrinėkime atminties pralaidumą imdami normalų taktinį dažnį (žr. 3.1.1 skyrių) tiek imant vieną, tiek dviejų kanalų plotį.

14 lentelė. Kanalų technologijos pralaidumas

	ALU	**FPU**
Vieno kanalo DDR (*Single Channel DDR*)	2805 MB/s	2810 MB/s
Dviejų kanalų DDR (*Dual Channel DDR*)	4615 MB/s	4626 MB/s

Iš 14 lentelės matyti, kad tiek sveikųjų, tiek trupmeninių skaičių perdavimo pralaidumas yra beveik vienodas. Teoriškai dvigubo kanalo technologija turėtų duoti dvigubai greitesnį duomenų perdavimą. Atlikus testą paaiškėjo, kad sveikųjų skaičių perdavimas padidėjo 64,5%, o trupmeninių – 64,6%. Taip yra todėl, kad ne visi taktai skiriami duomenų perdavimui. Iš 18 paveikslėlio aišku, kad dalis procesoriaus taktų skiriama informacijai iš atminties išrinkti (eilutės ir stulpelio adresams perduoti). Be to, pridėjus į valdiklį antrą kanalą, dažnai abiejų kanalų plotis nėra išnaudojamas. Jei vienu kanalu siunčiami duomenys,

o tuo metu kitas kanalas laukia, kol bus išrinkta reikiama informacija. Kaip žinoma, duomenys perduodami paketais, todėl gali skirtis į kanalą paduodamų išrinktų duomenų paketų dydžiai, kas sąlygoja mažesnį nei nurodytas teorinis pralaidumas.

3.3 Magistralės tyrimas

Sisteminės magistralės tyrimo idėja būtų tokia, kad pakėlus FSB magistralės dažnį ir šiek tiek sumažinus procesoriaus daugiklį būtų galima gauti tą patį procesoriaus taktinį dažnį esant didesniam magistralės dažniui. Deja, šiuolaikinių procesorių gamintojai nebeleidžia laisvai kaitalioti daugiklio, todėl jis paprasčiausiai vartotojui būna neprieinamas. Iš visų trijų testuojamų sistemų nė vienos procesorius neturi galimybės keisti daugiklio, todėl praktinis FSB magistralės tyrimas yra neįmanomas.

Imkime sistemą ir pabandykime paskaičiuoti teorinę magistralės įtaką. Šiuolaikinėse sistemose dažniausiai FSB ir pagrindinės magistralės santykis būna 1:1, t.y. tiek FSB tiek pagrindinė atmintis dirba tuo pačiu dažniu (sinchronizuotai). Imkime anksčiau aprašytą atvejį, kai teorinis duomenų pralaidumas pasiekiamas 3,2 GB/s esant 200 MHz FSB magistralės dažniui. Sakykime, kad procesoriaus taktinis dažnis nepakito, o FSB magistralės dažnį pakelkime iki 220 MHz. Tuomet iki 220 MHz padidės ir pagrindinės atminties dažnis. Tada idealiu atveju turėsime tokį teorinį duomenų pralaidumą (vertėtų prisiminti, kad signalai perduodami kylančio ir krintančio taktinio impulso fronto metu): 220 MHz * 2 impulsai * 8 bitai = 3520 MB/s = 3,52 GB/s. Matome, kad 10% magistralės pakėlimas idealiu atveju galėtų duoti 10% padidėjusį pralaidumą. Vertinant praktiškai tokio žymaus padidėjimo nesulauktume dėl to, kad pailgėtų kreipinių į atmintį laikas, be to augantis perduodamų signalų skaičius žymiai padidintų signalų vėlinimus, todėl praktinis paspartėjimas būtų mažesnis nei apskaičiuotas teorinis.

3.4 Bendras našumo įvertinimas

Iki šiol tyrėme atskirus kompiuterio komponentus neįvertindami jų bendros sistemos rėmuose. Kadangi sistema yra ją sudarančių komponentų visuma, kiekvieno komponento įtaka sistemos darbui yra labai svarbi. Be to, patys komponentai priklauso vienas nuo kito darbo spartos, pvz. FSB magistralės dažnis priklauso tiek nuo procesoriaus, tiek nuo pagrindinės atminties dažnio. Vienareikšmiškai įvertinti šios priklausomybės negalima, todėl pasakyti, kiek patys komponentai įtakoja vienas kitą yra neįmanoma, nors žinoma, kad įtakoja. Atlikti eksperimentai parodė, kad įmanoma bent apytiksliai pasakyti, kiek komponentai įtakoja pačią sistemą. Sudarykime komponentų įtakojimų lentelę:

15 lentelė. Bendra komponentų įtaka sistemai

Komponentas	**Savybė**	**Pagerėjimas**		
		Pentium4	**AthlonXP**	**Sempron**
Procesorius	Taktinis dažnis	10–11%	9–10%	13%
	L1 spartinančioji atmintinė	1350% (teoriškai)		
	L2 spartinančioji atmintinė	–	38%	30%
	Gijų technologija	4,2% ALU 68,4% FPU	–	–
	Perėjimų prognozė	47% (teoriškai)		
Atmintis	Kreipinių laikas	0,5%	0,4%	0,2%
	Dviejų kanalų technologija	65%	–	–
Magistralė	Dažnis	10% (teoriškai)		

Kaip matyti iš 15 lentelės, daugiausia sistemos darbo našumui įtakos turi procesoriaus architektūra ir jo ypatybės, todėl jo darbo spartinimui skiriamas ypatingas dėmesys, o gamintojai ieško vis naujų technologijų. Kai kurių paminėtų procesoriaus savybių ištirti neįmanoma, galima pateikti tik teorinius pagerėjimo paskaičiavimus. Atminties organizacijoje didžiausią reikšmę turi jos tipas bei naudojamos atminčių technologijos kaip dviejų atminties kanalų valdikliai. Magistralių įtaka nėra tokia žymi, bet galima teigti, kad dirbanti didesniu dažniu, magistralė vistik prisidės prie bendro sistemos našumo didinimo. Atliktuose eksperimentuose neįvertinti papildomi kompiuterio įrenginiai: kietasis diskas, vaizdo posistemis, kiti komponentai, prie sistemos jungiami per PCI magistrales ar išoriniai įrenginiai. Norint objektyviai įvertinti sistemos našumą reikėtų atsižvelgti ir į šiuos paminėtus komponentus.

4 Išvados

Šiame darbe buvo aptarti kompiuterio našumo didinimo aspektai. Atlikta Intel® Pentium® 4, AMD Athlon™ ir AMD Sempron™ procesorių ir kitų šiuolaikinio kompiuterio komponentų analizė našumo didinimo atžvilgiu. Remiantis tyrimo metu atliktų testų rezultatais galima daryti tokias išvadas:

- neįmanoma kiekybiškai įvertinti visų teorinėje dalyje aptartų našumo didinimo priemonių. Nei sąlyginio komandų vykdymo, nei registrų pakeitimo, nei dinamiškai parenkamų paruoštų komandų vykdymo, nei predikacijos nebuvo galima ištirti naudojant programines testavimo aplinkas, o teoriniam įvertinimui taipogi trūksta duomenų;
- procesoriaus taktinio dažnio padidinimo (*overclocking*), spartinančiųjų atmintinių bei gijų technologijos įtaka sistemos darbo spartai buvo įvertinta eksperimentiškai. Nustatyta, kad taktinio dažnio kėlimas sistemos našumą padidina nuo 9 iki 13%, L2 atmintinė – nuo 30 iki 38%, gijų technologija – net iki 68% operacijose su slankaus kablelio skaičiais;
- atlikti testiniai eksperimentai su atmintimi parodė, kad sumažinus kreipinių į atmintį laiką gaunamas našumo padidėjimas yra labai nežymus – nuo 0,2 iki 0,5%, o dviejų kanalų atminties technologijos naudojimas našumą padidina iki 65%;
- L1 spartinančiosios atminties, perėjimo prognozės bei magistralės dažnio parametrų įtaka buvo įvertinta teoriškai. Remiantis teorinėje dalyje išdėstytais šių komponentų darbo principais nustatyta, kad L1 spartinančioji atmintinė visos sistemos našumą padidina iki 1350%, perėjimo prognozė – iki 47%, o magistralės dažnio pakėlimas – iki 10%;
- skirtingų gamintojų komponentų (ar net sistemų) lyginimas yra pakankamai sunkus dėl naudojamų skirtingų technologinių ypatumų, testavimo programinės įrangos įvairovės ir metrikų ar net pačių komponentų darbo specifikos.

Technologijos nestovi vietoje, todėl galimos tyrimo kryptys šia linkme galėtų būti procesorių su dviem branduoliais tyrimas ir jų įtaka sistemų našumui ar naujų ypač aukšto našumo sistemų tyrimas.

5 Literatūros sąrašas

1. Dabartinės lietuvių kalbos žodynas : apie 50000 žodžių lizdų / Lietuvių k. inst. ; redkol.: S. Keinys (vyr. red.) ... [et al.] 3-iasis patais. ir papild. leid. Vilnius : Mokslo ir enciklopedijų l-kla, 1993

2. Maciulevičius S. Spartinančioji atmintinė [Žiūrėta 2005 m. balandžio 18 d.]. Prieiga per internetą: http://ifko.ktu.lt/~stama/Tekstai/KESAS.htm

3. Harman N.A. High Performance Microprocessors, 2005 [Žiūrėta 2005 m. balandžio 19 d.]. Prieiga per internetą: http://www.cs.swan.ac.uk/~csneal/HPM/risc.html

4. Gerritsen A CISC vs RISC, 1999 [Žiūrėta 2005 m. balandžio 20 d.]. Prieiga per internetą: http://www.tomax7.com/aplus/APlusCD/CISC%20vs%20RISC.doc

5. Engelhart K. CPU structure and Function [Žiūrėta 2005 m. balandžio 20 d.]. Prieiga per internetą: http://www.ee.unb.ca/kengleha/courses/CMPE4233/Notes/Chapter11.pdf

6. Ronen R. The Pentium® II/III Processor "Compiler on a Chip" [Žiūrėta 2005 m. balandžio 20 d.]. Prieiga per internetą: http://www.cs.tau.ac.il/~afek/FewB&W.pdf

7. Patterson J. Modern Microprocessors: A 90 Minute Guide!, 2003 [Žiūrėta 2005 m. balandžio 21 d.]. Prieiga per internetą: http://www.pattosoft.com.au/Articles/ModernMicroprocessors/

8. Downey T. Branch Prediction [Žiūrėta 2005 m. balandžio 22d]. Prieiga per internetą: http://www.cs.fiu.edu/~downeyt/cop3402/prediction.html

9. Harman N.A. High Performance Microprocessors [Žiūrėta 2005 m. balandžio 23 d.]. Prieiga per internetą: http://www.cs.swan.ac.uk/~csneal/HPM/reorder.html

10. Smith M. D. Support For Speculative Execution In Highperformance Processors, 1992 [Žiūrėta 2005 m. balandžio 26 d.]. Prieiga per internetą: http://velox.stanford.edu/papers/mds_thesis.pdf

11. Brüning U. Predication [Žiūrėta 2005 m. balandžio 27 d.]. Prieiga per internetą: http://mufasa.informatik.uni-mannheim.de/lsra/lectures/ss02/vl_ra1/vl_2.pdf

12. PC Tech Guide [Žiūrėta 2005 m. Balandžio 27 d.]. Prieiga per internetą: http://www.pctechguide.com/02procs_Itanium.htm

13. Severinovskiy E. Intel Hyper-Threading Technology Review, 2004 [Žiūrėta 2005 m. Balandžio 27 d.]. Prieiga per internetą: http://www.digit-life.com/articles/pentium4xeonhyperthreading/

14. DEW Associates Corporation Memory Speeds, 2000 [Žiūrėta 2005 m. Balandžio 28 d.]. Prieiga per internetą: http://www.dewassoc.com/performance/memory/memory_speeds.htm

15. Stokes J. Ars Technica RAM Guide [Žiūrėta 2005 m. Balandžio 29 d.]. Prieiga per internetą: http://arstechnica.com/paedia/r/ram_guide/ram_guide.part3-3.html

16. Infineon Technologies North America Corporation and Kingston Technology Company, Inc. Intel Dual-Channel DDR Memory Architecture White Paper, 2003 [Žiūrėta 2005 m. Balandžio 30 d.]. Prieiga per internetą: http://www.kingston.com/newtech/MKF_520DDRwhitepaper.pdf

17. AMD Opteron™ 4P Server Comparison Reference [Žiūrėta 2005 m. Balandžio 30 d.]. Prieiga per internetą: http://www.amd.com/us-en/assets/content_type/DownloadableAssets/AM190_briefv3.pdf

18. Longbottom R. Classic Benchmarks [Žiūrėta 2005m. Gegužės 1d.]. Prieiga per internetą: http://homepage.virgin.net/roy.longbottom/classic.htm

19. SPEC CPU2000 V1.2, 2005 [Žiūrėta Gegužės 1d.]. Prieiga per internetą: http://www.spec.org/cpu2000/

Santrumpų ir terminų žodynėlis

ALU (*Arithmetic Logic Unit*) – aritmetinis loginis įtaisas, procesoriaus dalis, atliekanti dvejetainių skaičių matematines ir logines operacijas su sveikaisiais skaičiais.

BGA (*Ball Grid Array)* – pakavimo sistema duomenų perdavimui. Šis pakavimo metodas sumažina paketo dydį bei perdavimo trasomis trukmę.

BIOS (*Basic Input/Output System*) – sistema, atliekanti techninės įrangos sužadinimą įjungus kompiuterį. Taip pat kontroliuoja užsikrovimo procesą, leidžia keisti kai kurių sistemų komponentų konfigūraciją.

CAS (*Column Address Strobe*) – atminties valdiklio signalas, kuris susieja stulpelio adresą su atitinkama vieta eilutėje.

CISC (*Complex Instruction Set Computer*) – procesorių architektūra, kurioje realizuotas didelis kiekis skirtingų ir sudėtingų komandų.

CPI (*Cycles Per Instruction*) – taktų trukmė, reikalinga vienai komandai įvykdyti.

DDR (*Double Data Rate*) – SDRAM atminties tipas, kuris informaciją perduoda kylančio ir krintančio taktinio impulso fronto metu. Dar kitaip ji žymima DDR SDRAM.

EPIC (*Explicitly Parallel Instruction Computing)* – procesorių architektūros, kuriose kiekviena komandos priklausomybė užkoduojama pačia komanda.

FPU (*Floating Point Unit*) – aritmetinio įtaiso koprocesorius, atliekantis aritmetinius veiksmus su dvejetainiais trupmeniniais skaičiais.

FSB (*Front Side Bus*) – kompiuterio magistralė, jungianti procesorių su pagrindinės atminties valdikliu ir naudojama duomenims perduoti į procesorių arba iš jo.

Funkcinis įtaisas – procesoriaus įtaisas, skirtas vykdyti apibrėžtas funkcijas, pvz. įkrauti duomenis į spartinančią atmintinę ar įvykdytį trupmeninių skaičių sudėtį.

LRU (*Least Recently Used*) – išrinkimo algoritmas, kurio metu sudaromas sąrašas, surikuotas pagal seniausiai naudotus elementus.

MFLOPS (*Million Floating point Operations Per Second*) – metrika, naudojama kompiuterių našumui nusakyti. Matuojama 10^6 trupmeninių skaičių operacijomis per sekundę.

MIPS (*Million Instructions Per Second*) – metrika, naudojama kompiuterių našumui nusakyti. Matuojama 10^6 operacijomis per sekundę.

NOOP (*No Operation*) – asemblerinės kalbos komanda, kuri nieko nevykdo. Šia komanda užpildomos laukiančios konvejerio fazės (etapai).

RAM (*Random Access Memory*) – kompiuterio laikmenos tipas, kurioje laikoma nepastovi informacija, reikalinga laikiniems skaičiavimams atlikti.

RAS (*Row Address Strobe*) – atminties valdiklio signalas, kuris susieja eilutės adresą su atitinkama eilute pagrindinės atminties bankuose.

RDRAM (*Rambus Dynamic Random Access Memory*) – DRAM tipo atmintis. Pasižymi mažu magistralės pločiu, bet didžiausiu dažniu.

Registrai – labai sparčios, mažo dydžio duomenų laikmenos procesoriuje. Naudojamos saugoti laikiniems duomenims, komandoms ar skaičiavimo rezultatams.

RISC (*Reduced Instruction Set Computer*) - procesoriaus architektūros tipas, kurioje realizuoti optimizuoti maži komandų rinkiniai.

SDRAM (*Synchronous Dynamic Random Access Memory*) – DRAM atminties technologija, kurios darbo taktai yra susieti su procesoriaus darbo taktais. Tuomet vykdant komandas ir jų operandų perdavimą sutaupoma laiko, todėl ji dar vadinama sinchronizuota atmintimi.

SIMD (*Single Instruction Multiple Data*) – M. Flyno klasifikacijoje apibrėžtas procesoriaus tipas, kuomet viena komanda atlieką tą patį veiksmą su skirtingomis duomenų porcijomis.

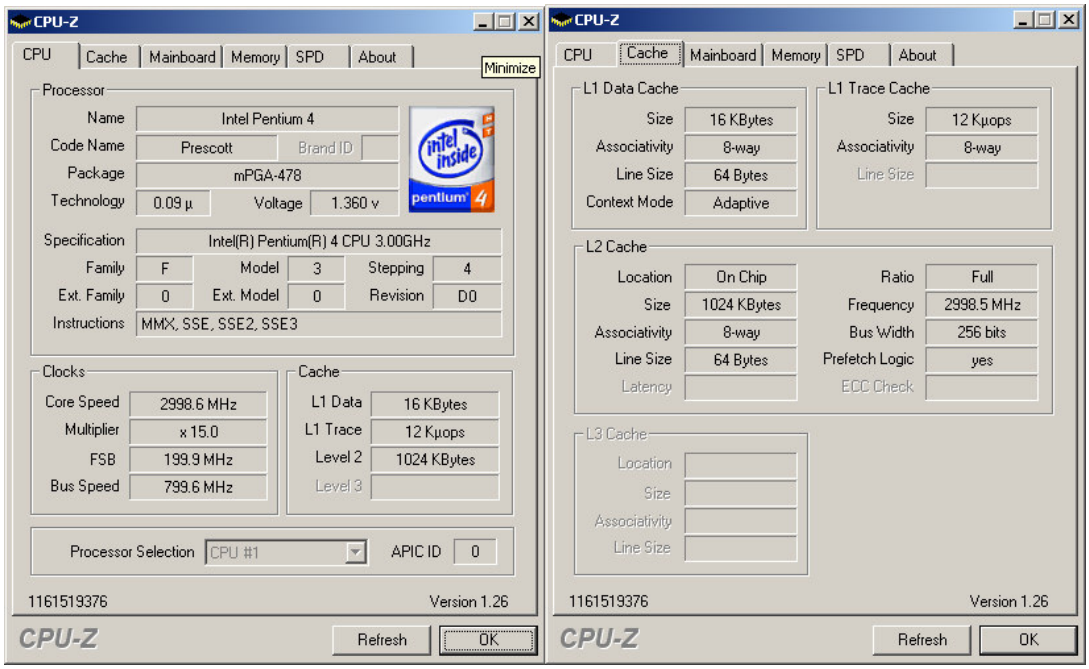
VLIW (*Very Long Instruction Word*) – procesoriaus architektūra, kurios komandų rinkinys yra supakuotas į vieną ilgą komandą. Tokie procesoriai naudojami aukšto našumo sistemose.

1 Priedas. Testuojamų sistemų charakteristikos

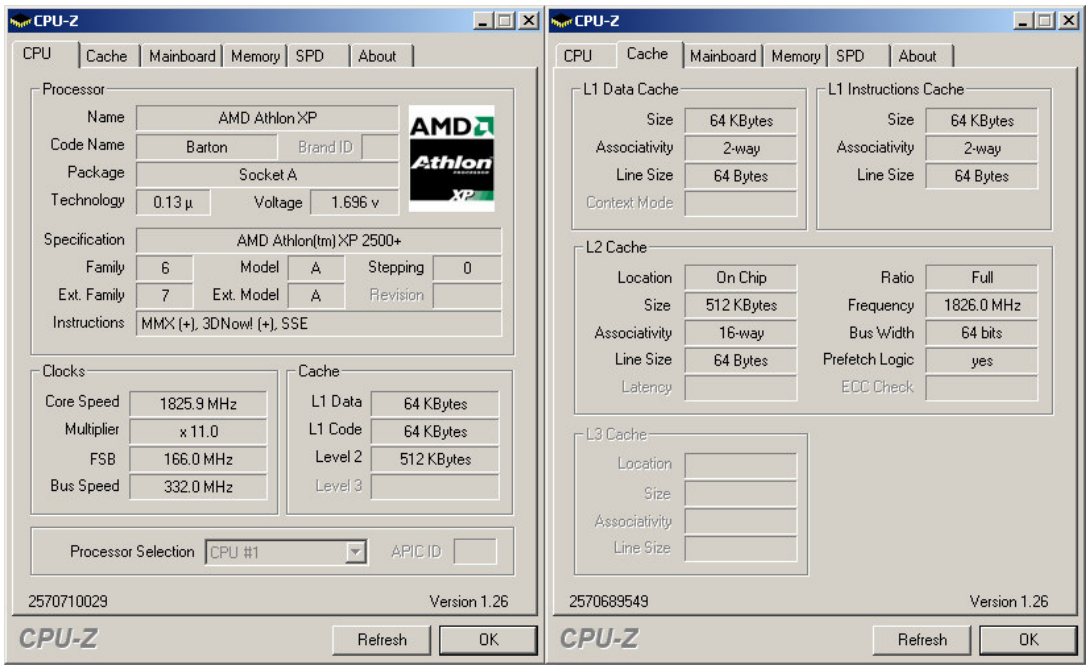
16 lentelė. Sistemų charakteristikos

	Sistema 1	**Sistema 2**	**Sistema 3**
Procesorius	Intel® Pentium® 4 E su gijų technologija 3,06 GHz, 1 MB L2, 0,09 nm, 800 MHz FSB, S478	AMD Athlon™ XP 2500+ 0,13 nm 1,8 GHz 333MHz FSB, 512KB L2, Socket A	AMD Sempron™ 2500+, 0,13 nm 1,75 GHz, 400 MHz FSB, 512 KB L2, Socket A
Pagrindinė atmintis	512 MB ADATA PC3200 Kreipinių laikas: CL 2,5-4-4-8 512 MB PQI PC3200 Kreipinių laikas: CL 3-4-4-8	512 MB PQI PC2700 Kreipinių laikas: CL 2,5-3-3-7	512 MB PQI PC2700 Kreipinių laikas: CL 2,5-4-4-8
Pagrindinė plokštė	Asus P4C800E-Deluxe Valdiklis: Intel 875P BIOS: 2004 03 04	Dfi AD-77 Valdiklis: Via KT400A BIOS: 2004 11 02	Epox 8RDAE Valdiklis: nForce400 BIOS: 2004 08 30
Vaizdo plokštė	Sparkle GeFORCE FX5500 8X AGP, 128MB DDR, 128Bit	Sparkle GeFORCE MX440 8X AGP, 64MB DDR, 64Bit	Daytona GeFORCE MX440 8x AGP, 128 MB DDR
Kietasis diskas	Seagate 160GB 7200rpm su NCQ, 8MB SerialATA	Maxtor 120 GB 7200rpm, 2MB UDMA133	80 GB 7200rpm, 2 MB UDMA100
CD/DVD įrenginys	Lite-On SOHW-1673S	Lite-On LTC-48161H	Lite-On Combo
Operacinė sistema	Microsoft Windows 2003 Enterprise Edition su SP1	Microsoft Windows 2003 Enterprise Edition su SP1	Windows 2000 Professional Edition su SP4

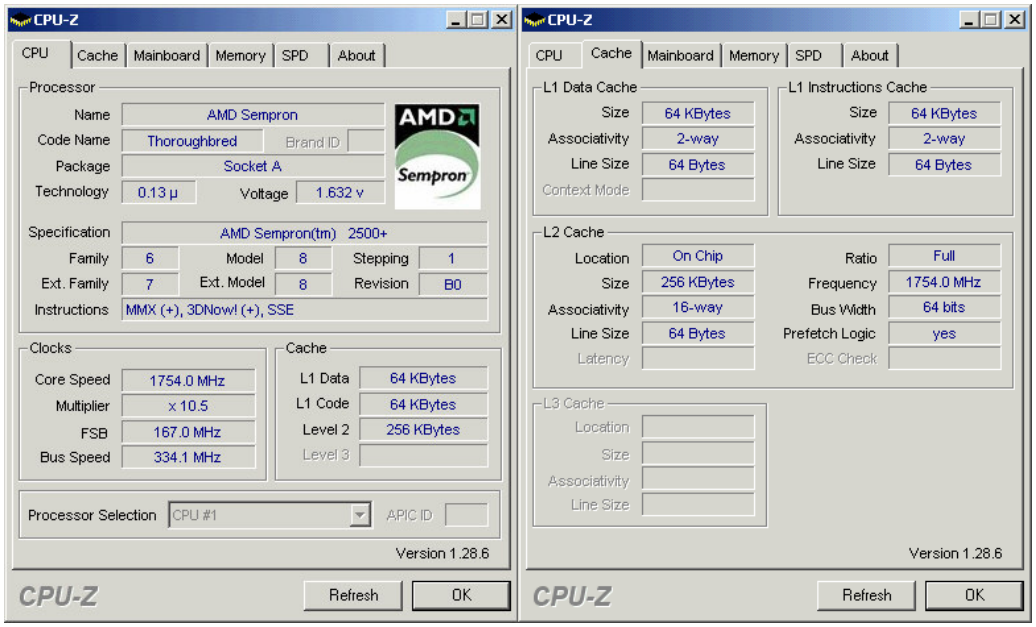
2 Priedas. Procesorių charakteristikos



28 pav. Intel® Pentium® 4 procesoriaus duomenys



29 pav. AMD Athlon™ XP procesoriaus duomenys



30 pav. AMD Sempron™ procesoriaus duomenys

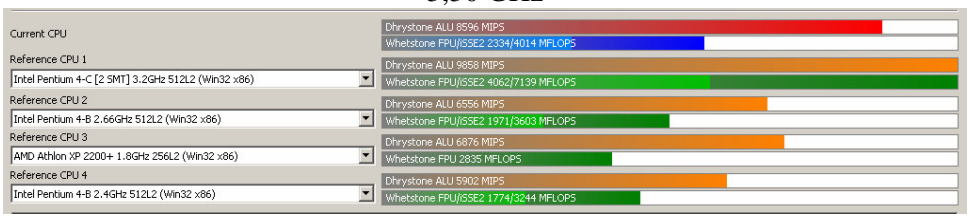
3 Priedas. Testavimų rezultatai

Raudona – sveikųjų skaičių operacijos.
Mėlyna – trupmeninių skaičių operacijos.

Taktinis dažnis

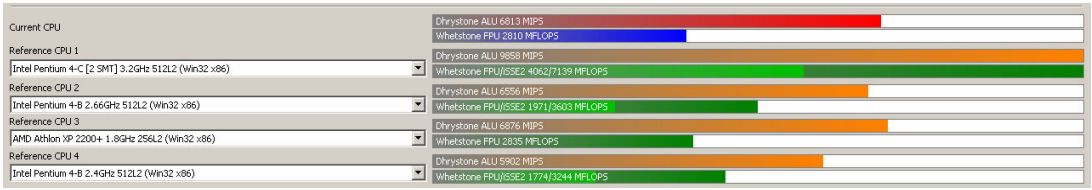
3 GHz

3,30 GHz

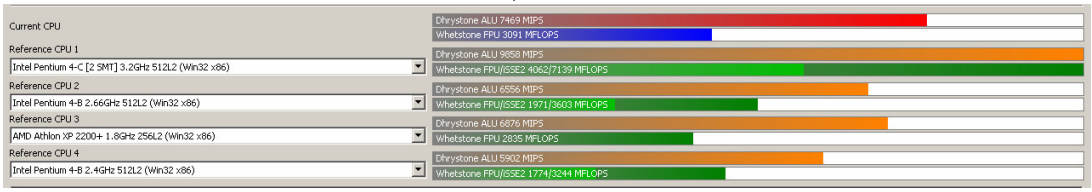


Intel® Pentium® 4

1,8 GHz



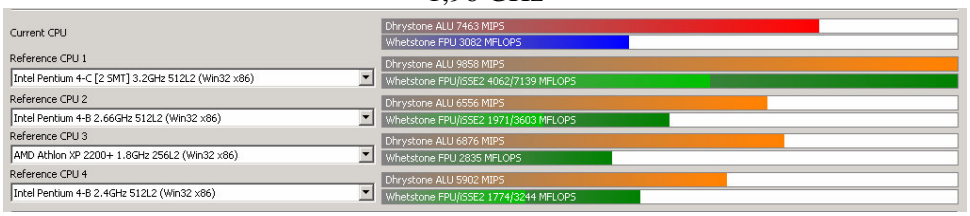
2,0 GHz



AMD Athlon™ XP

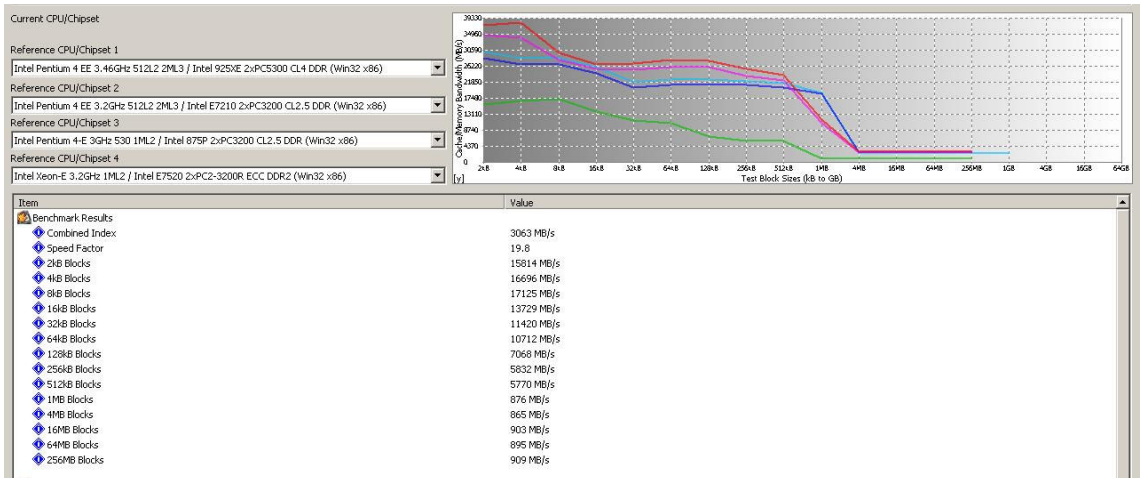
1,75 GHz

1,96 GHz

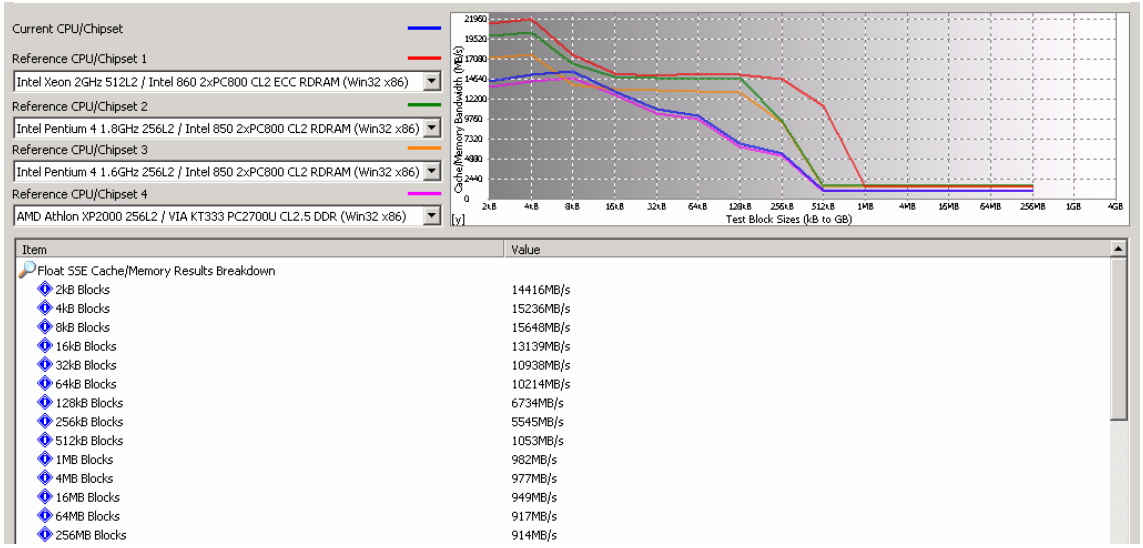


AMD Sempron™

L2 spartinančioji atmintis



AMD Athlon™ XP (žalia)

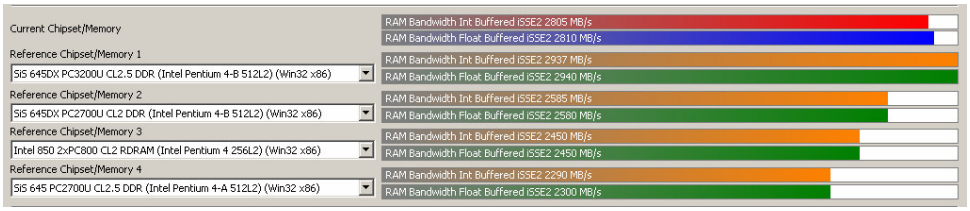


AMD Sempron™ (mėlyna)

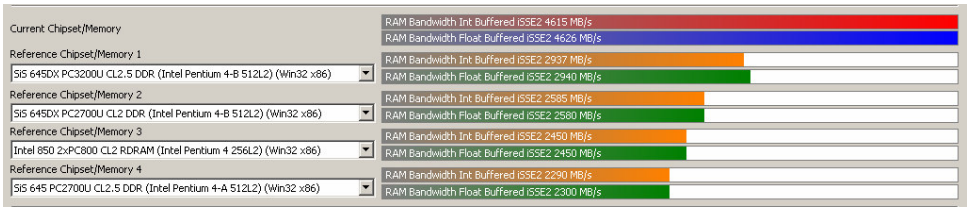
Dviejų kanalų technologija

Raudona – sveikųjų skaičių atminties posistemio pralaidumas.
Mėlyna – trupmeninių skaičių atminties posistemio pralaidumas.

1 atminties kanalas



2 atminties kanalai



Intel® Pentium® 4