|  |
| --- |
| 1. Kompiuterio architektūra – kelių lygių požiūris 1 |
| **Skaitmeninis lygis:**   * Tai ventilių ir loginių schemų lygis. * Ventiliai formuoja paprastas logines funkcijas. * Realizuojamos sudėtingesnė loginės funkcijos, atminties žodžiai ir veiksmai su jais.   **Mikroprogramų lygis:**   * Mikroprograminiame lygyje naudojami registrai bei vidinė registrinė atmintis, realizuojamas aritmetinis – loginis įrenginys. * Formuojami duomenų traktai, kuriais perduodami duomenys bei skaičiavimų rezultatai.   **Komandu lygis:**   * Kiekvieno kompiuterio funkcionavimas aprašomas jo komandų sistema, kuri interpretuojama mikroprogramomis. * Kompiuteris gali interpretuoti kelias komandų sistemas.   **Operacines sitemos lygis:**   * Tai hibridinis lygis – naudojamos kompiuterio komandos, kurias papildo naujas komandų rinkinys, realizuojama skirtinga atminties organizacija. * Naujos trečio lygio priemonės yra interpretuojamos antro lygio komandomis.   **Asamblerio lygis:**   * Tai žemo lygio simbolinė programavimo kalba. * Šiame lygyje jau galima pateikti vartotojui priimtinas programas. * Tokios programos transliuojamos į žemesnį lygį ir po to interpretuojamos virtualaus ar realaus kompiuterio komandomis.   **Auksto lygio kalbos:**   * Taikomosios programos kuriamos aukšto lygio kalbomis. * Programos aukšto lygio kalboje toliau transliuojamos arba interpretuojamos. * Kompiuteris realiai dirba žemiausiame lygyje – iš aukštesniųjų lygių iki realaus programos vykdymo būtinai bus vartotojui neretai nepastebimi parengiamieji transliavimo ar interpretavimo procesai. |
| 1. Komandų sistema. Pagrindiniai komandų sistemų tipai 2 |
| Skaičiavimams reikalingą operacijų rinkinį kompiuteryje realizuoja **komandų sistema**, kurioje vieną operaciją gali atitikti ir kelios komandos, besiskiriančios operandų tipais, adresacijos būdais ir pan. Komandų grupės:   * Aritmetinės ir loginės (sveikųjų skaičių: +, –, \*, /; loginės: IR, ARBA) * Duomenų persiuntimo (registrų įkrovimas, įrašymas į atmintį) * Valdymo (perėjimai, procedūrų iškvietimas ir grįžimas, ciklai) * Slankaus kablelio (slankaus kablelio skaičių: +, –, \*, / ) * Dešimtainės (dešimtainių skaičių: +, –, \*, / ) * Eilučių apdorojimo (eilučių persiuntimas, palyginimas, paieška) * Sisteminės (OS iškvietimas, virtualios atminties valdymas)   Komandų sistemų tipai: 1) stekas, 2) akumuliatorius, 3) registrų grupė. |
| 1. Akumuliatoriaus tipo komandų sistema 1 |
| **Akumuliatorius**: vienas iš operandų saugomas akumuliatoriuje, rezultatas – taip pat, operandų išrinkimas ir įrašymas – load, store.  **Akumuliatorius**: specialus registras (kaupiantysis registras), į kurį informacija  įrašoma prieš operaciją; iš jo rezultatai grąžinami į atmintį. Pavyzdžiai: PDP-8,Motorola 6809 |
| 1. Steko tipo komandų sistema 1 |
| **Stekas –** specialiai suprojektuota atmintis, į kurią informacija įrašoma nuosekliai, iš jos informacija perskaitoma taip pat tik nuosekliai. Informacijos vietą steke rodo **steko rodyklė (*stack pointer*)**.  **Stekas**: operandai saugomi steke, rezultatas – taip pat, operandų išrinkimas ir  įrašymas – procedūros **push**, **pop**. B5500 |
| 1. Bendros paskirties registrų tipo komandų sistema 1 |
| **Registrų grupė**: operandai saugomi registruose, rezultatas – taip pat, operandų išrinkimas ir įrašymas – load, store. Pavyzdžiai: IBM 370, PowerPC. **Registrų failas**: 8-16-32-… registrų grupė procesoriuje, į kuriuos vienas iš  operandų įrašomas prieš operaciją;  Pastaruoju metu vyrauja **bendrosios paskirties registrų tipo architektūra**.Tai paaiškinama tuo,kad:  1) registrai, kaip ir kiti CPU mazgai, greitesni, nei atmintis;  2) kompiliatoriai juos panaudoja kodo generavimui efektyviau, nei kitas vidines CPU atminties formas (pvz., steką);  3) saugant operandus ir tarpinius rezultatus registruose, sumažėja kreipinių į atmintį skaičius, tuo atlaisvinant ją kitiems tikslams (I/O);  4) generuojamas kompaktiškesnis kodas, nes registrams nurodyti reikia mažiau vietos. |
| 1. Komandų formatai. Operandų nuorodos 2 |
| **Komandų formatai:** LOAD R1, A ADD ADD B  Iš pradžių užrašome komanda, tada operandus su kuriais dirbsime.  Bendruoju atveju darome lentele. OpK – operacijos kodo laukas, Op1, …, Opn – operandų nuorodos. Nuorodų skaičius priklauso nuo konkretaus atvejo ir gali būti lygus 0, 1, 2 ir t.t  **Operandų nuorodos:**  Operando nuorodos pavidalas priklauso nuo to, koks yra operandas, kur jis saugomas.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | OpK | Op1 | Op2 | … | Opn |   Operandu gali būti:   * + kurio nors registro turinys, (registras ir lastele nurodomi adresu)   + atminties ląstelės turinys, \* konstanta, \* adresas, \* sąlyga.   Nuo to priklauso operando nuorodai skirto lauko (laukų) ilgis. |
| 1. Operandų adresavimo būdai 3 |
| Operandų vietai nurodyti kompiuteriuose naudojama daug įvairių adresavimo būdų, kurie tinka vienai ar kitai situacijai. Aprašydami juos, pateiksime ir pavyzdžius, iliustruojančius operando išrinkimą. M[A]žymės atminties ląstelę, kurios adresas A, Reg[n]žymės registrą, kurio numeris n.  1. ***Registrinė*** adresacija – operandas yra registre, kurio numeris nurodytas komandoje, pavyzdžiui:  Add R4, R1 Reg[4] := Reg[4] + Reg[1]  2. ***Netiesioginė registrinė*** – operando adresas yra registre, kurio numeris nurodytas komandoje, pavyzdžiui: Add R4, (R1) R4 := R4 + M[R1]  3. ***Santykinė*** *(****bazinė*** arba ***su poslinkiu***)– operando adresas paskaičiuojamas prie registro turinio pridedant nurodytą poslinkio reikšmę; tinka lokaliniams kintamiesiems pasiekti. Pavyzdžiui:  Add R4, 9(R1) R4 := R4 + M[R1+9]  4. ***Indeksinė*** – operando adresas paskaičiuojamas sumuojant bazės ir indekso registrų turinius; tinka masyvo elementams išrinkti (R1 – masyvo bazinis adresas, R2 – indekso reikšmė). Pavyzdžiui:  Add R4, (R1+R2) R4 := R4 + M[R1 +R2]  4a. ***Indeksinė su koeficientu*** – nuo indeksinės skiriasi tik tuo, kad indekso reikšmė padauginama iš koeficiento (***m***), atitinkančio duomenų elemento ilgį. Pavyzdžiui:  Add R4, 8(R1) [R2] R4 := R4 + M[8+R1 +R2\*m]  5. ***Tiesioginė*** arba ***absoliutinė*** – adreso reikšmė (pilna ar jo dalis) nurodoma komandoje; tinka statiniams duomenims nurodyti. Pavyzdžiui: Add R4, (704) R4 := R4 + M[704]  6. ***Betarpiškoji*** – operandas yra komandos formato dalis. Tai gali būti interpretuojama, kaip adreso nuoroda yra komandų skaitiklyje. Pavyzdžiui: Add R4, #7 R4 := R4 + 7  7a. ***Autoinkrementinė*** – operando adresas yra registre, kurio turinys automatiškai didinamas, pavyzdžiui: Add R4, (R1)+ R4 := R4 + M[R1] R1 := R1 + d  7b. ***Autodekrementinė*** – operando adresas yra registre, kurio turinys automatiškai mazinamas, pavyzdžiui: Add R4, –(R1) R1 := R1 – d R4 := R4 + M[R1]  Patogu apdorojant masyvus. Pradžios adresas įrašomas į lauke R1 nurodytą registrą |
| 1. Komandų sistemos išplėtimai procesoriuose 2 |
| **Išplėtimo priežastys ir prielaidos**   * procesoriai buvo orientuoti į sveikųjų ir slankaus taško skaičių apdorojimą * grafinės ir audio informacijos skaitmeninio apdorojimo plitimas * technologijos vystymasis * procesorių žodžio ilgio didinimas nuo 32 bitų iki 64 bitų * daugeliu atvejų skaitmeninei grafinei ir audio informacijai koduoti pakanka 16 ar net 8 bitų * SIMD ir vektorinio principų panaudojimo galimybės   1996 metais Intel įvedė MMX technologiją - savo procesorių komandų sistemą papildė 57 naujomis komandomis, skirtomis multimedijos programoms optimizuoti  Šios komandos duomenis traktuoja taip, kaip tai priimta SIMD sistemose *(Single Instruction – Multiple Data).* Panašūs komandų sistemų papildymai pasirodė ir kitų firmų procesoriuose  **Reikalavimai išplečiant komandų sistemą**  Siekiant išlaikyti suderinamumą su esančia PĮ ir OS, projektuotojai turėjo atsižvelgti į tokius dalykus:   * MMX komandas naudojančios programos turi dirbti visose turimose daugelio užduočių ir paprastose OS; tai reiškia, kad MMX technologija turi nepridėti naujų architektūriškai matomų būsenų ar įvykių (*exceptions)* * MMX komandų nenaudojančios programos turi dirbti be jokių pakeitimų; tai reiškia, kad MMX technologija turi nekeisti buvusių IA-32 komandų * Turimos taikomosios programos turi galėti panaudoti MMX technologiją be visos programos perdirbimo; tai reiškia, kad MMX technologija gali būti panaudota atskiroje procedūroje, paliekant likusią dalį nepakeistą; tam reikia, kad MMX komandos turi gerai dirbti esamoje procedūrų iškvietimo sistemoje * MMX komandas naudojančios programos turi dirbti ir senesniuose procesoriuose, kurie nepalaiko MMX; tai reiškia, kad turi būti parengtos DLL MMX ir ne-MMX technologijų procesoriams |
| 1. Multimedijos informacija ir jos formatai 1 |
| * Videoinformacija būna statinė ir dinaminė. * Naudojami du pagrindiniai grafinės informacijos vaizdavimo būdai – matricinis ir vektorinis. * Audiosignalai yra analoginiai, jų dažnis nuo 15 Hz iki 20 KHz. * Kompiut atvaizduojama audioinformacija turi būti pervesta į skaitmeninį pavidalą naudojant analoginius-skaitmeninius keitiklius. * Analoginis audiosignalas atstatomas per skaitmeninius-analoginius keitiklius. |
| 1. Multimedijos informacijos apdorojimo specifika 2 |
| Vaizdų apdorojimas, sudėtinga kompiuterinė grafika – būtini šiuolaikinių automatizuoto projektavimo sistemų ir kompiuterinių žaidimų elementai. Tam reikalingos specialios operacijos, kurioms vykdyti ir kuriami **specializuoti funkciniai įtaisai**, kartais vadinami multimedijos funkciniais įtaisais.  Pirmieji tokie įtaisai pasirodė jau senokai - Intel 80860 (tiesa, asmeniniuose kompiuteriuose šis mikroprocesorius nebuvo naudojamas).  Vėliau jie imti naudoti **Pentium MMX** procesoriuose (MMX vadinamas Intel firmoje sukurtas komandų rinkinys), AMD procesoriuose (pradedant K6-2, komandų rinkinys vadinamas **3D Now!),** Motorolos (PowerPC; komandų rinkinys **AltiVec)** ir kituose procesoriuose.  Dabartiniu metu visi asmeniniams kompiuteriams skirti procesoriai (ir ne tik jie) turi kokį nors multimedijos komandų rinkinį ir atitinkamą funkcinį įtaisą (ar įtaisus).  Pagrindinis multimedijos funkcinių įtaisų bruožas – lygiagretus apdorojimas, kai į vieną ilgą žodį sujungiami keli (2-8) trumpi. |
| 1. Konvejerio ir superkonvejerio esmė 2 |
| Kompiuteryje vykdomi procesai - tai mašinos komandų sekos I1, I2, I3, ... , In.  Kiekvieną komandą Ii  logiškai sudaro kelios fazės, kurias, pavyzdžiui, galima nusakyti taip:  F - komandos išrinkimas, D - komandos dekodavimas, O - operandų išrinkimas, X - operacijos vykdymas,  W - rezultato įrašymas. Toks komandos suskirstymas į fazes nėra vienintelis  Esant nuosekliam kom vykdymui, (i+1)-ji kom pradedama vykdyti tik po to, kai baigiama vykdyti i-ji komanda  Konvejerizuotam komandų vykdymui reikia, kad konvejeris dirbtų ritmingai:  **Konvejerizacija** reiškia, kad sudėtingi veiksmai (šiuo atveju – komandos vykdymas) suskaidomi į etapus, kurių trukmė yra maža. Kiekvienas etapas tai atskirose schemose (blokuose) atliekami veiksmai. Jų trukmė atitinka CPU taktą. Siekiant mažinti takto trukmę tenka didinti etapų skaičių. Čia gali būti skiriami du konvejeriai:  \*Procesoriaus (komandos) konvejeris \*Funkcinio įtaiso konvejeris  Pradžioje dauguma procesorių turėjo dviejų – penkių pakopų konvejerius.  Siekiant sumažinti takto trukmę, pakopų skaičius buvo didinamas ir dabar viršija 10 (juo daugiau pakopų, tuo kiekviena iš jų paprastesnė, todėl gali būti įvykdyta per trumpesnį laiką). Tokie procesoriai vadinami **superkonvejerizuotais**.  Pentium konvejeris turi tik 5 pakopas, AMD **Athlon** sveikųjų skaičių įtaiso konvejeris turi **10** pakopų, FPU - **15** pakopų, o **Pentium 4** konvejeris – net **20.** Čia naudojamas terminas – **hiperkonvejeris**.  1 ir 2 pakopose pagal PC ir BTB turinius į trasavimo kešą TC įkraunamos mikrooperacijos (MO).  3 ir 4 pakopose iš TC atrenkamos MO.  5 pakopoje MO perduodamos į resursų paskirstymo įtaisą.  6 pakopoje paskirstomi resursai, reikalingi MO vykdyti. |
| 1. Konvejerio kliūčių tipai 2 |
| * ***struktūrinės kliūtys***, atsirandančios dėl to, kad resursai gali būti nepakankami įvairių pakopų ir komandų etapų reikmėms tenkinti * ***duomenų kliūtys***, atsirandančios dėl to, kad vienos komandos vykdymo rezultatai naudojami kitoje komandoje kaip operandai * ***valdymo kliūtys***, atsirandančios dėl būtinumo išrinkti komandas iš kitos vietos, nei tai aliekama jas vykdant nuosekliai |
| 1. Duomenų kliūtys ir jų įveikimas 3 |
| * ***duomenų kliūtys***, atsirandančios dėl to, kad vienos komandos vykdymo rezultatai naudojami kitoje komandoje kaip operandai   Galimos išeitys:   * ***programinės:***   – įterpti NOOP  – pakeisti komandų eilės tvarką   * ***aparatinės:***   – blokuoti konvejerio darbą  – pridėti papildomas duomenų linijas – apylankas |
| 1. Valdymo kliūtys ir jų įveikimas 2 |
| ***valdymo kliūtys***, atsirandančios dėl būtinumo išrinkti komandas iš kitos vietos, nei tai aliekama jas vykdant nuosekliai  Ką daryti, kad būtų galima sumažinti galimus laiko nuostolius?  anksčiau išsiaiškinti, ar bus perėjimas; anksčiau paskaičiuoti naują PC reikšmę  Priemonės perėjimo vėlinimui sumažinti:  Prognozuojama, kad perėjimo nebus; Komandų eilės tvarkos pakeitimas.  Kitos priemonės laiko nuostoliams sumažinti:  kelių srautų vykdymas lygiagrečiai; papildomo išrinkimo buferio naudojimas; ciklų buferio naudojimas; perėjimų prognozavimas; komandų eilės tvarkos pakeitimas |
| 1. Našumas, jo įvertinimas. Technologijos išvystymas ir našumas 2 |
| **Našumas** ≡ **F / CPI** Ši formulė rodo, kad našumą galima didinti dviem būdais:   1. didinant darbo dažnį, 2) mažinant CPI – tobulinant procesoriaus architektūrą   **Kompiuterių našumas**. Kas lemia darbo dažnio didinimą?  gamybos proceso išvystymo laipsnis procesorių mikroarchitektūros ypatumai.  Kas lemia CPI mažinimą? - Procesorių mikroarchitektūros ypatumai:  1.komandų realizacijos algoritmų išvystymas 2.didesnės apimties veiksmų atlikimas vieno ciklo metu.  Dar viena aplinkybė, padedanti didinti našumą – SIMD arba vektorinio apdorojimo principų panaudojimas Nasumo didinimas:  technologijos normos mažinimas, superskaliariškumas,  superkonvejerizacija, specializuoti funkciniai įtaisai (MMX, SSE, …),  registrų pakeitimas (*register renaming*), komandų vykdymo eilės tvarkos pakeitimas,  perėjimų spėjimas (*branch prediction*), išankstinis įkrovimas (*speculation*),  kelių branduolių panaudojimas, daugelio gijų skaičiavimų palaikymas (*multithreading*),  sudėtingų sistemų realizacija viename kristale (SOC - *Systems On the Chip*).  **Technologijos norma** apsprendžia kristalo plotą ir faktiškai riboja jame realizuojamo įtaiso sudėtingumą  1.Jei pirmasis asmeniniams kompiuteriams skirtas Intel 16 bitų mikroprocesorius buvo gaminamas naudojant 3 mikronų technologiją, tai dabar naudojama 32 nm (0,032 mikrono) technologija  2.Technologijos normos sumažinimas beveik **100** kartų sudaro galimybę tame pačiame kristalo plote sutalpinti beveik **10000** kartų sudėtinges-nes schemas ir padidinti jų darbo spartą |
| 1. Komandų vykdymas ne eilės tvarka 2 |
| * Superskaliariniuose procesoriuose vienu metu gali būti pradėtos vykdyti kelios komandos. * Kadangi kompiliatoriaus suformuota komandų seka programoje ne visuomet bus palanki kelių funkcinių įtaisų aprūpinimui darbu, programos vykdymo metu šiuolaikiniuose procesoriuose esančios specialios schemos analizuoja didelį vykdymo eilėje laukiančių komandų skaičių ir **dinamiškai** parenka vykdymui tas komandas, kurios konkrečiu momentu galėtų būti vykdomos, nors prieš jas eilėje yra kitos komandos.   Siekiant pasiekti, kad komandos būtų inicijuojamos ne eilės tvarka, naudojamas komandų buferis, vadinamas **komandų langu.** Komandų langas gali būti:   * **centralizuotas** – bendras visiems funkciniams įtaisams * **paskirstytas** – kiekvienas funkcinis įtaisas turi savo langą |
| 1. Registrų pakeitimas 1 |
| Tegul programoje turime tokias dvi komandas:  **k1: add ..., r2, ...;** [ ... ← (**r2**) + (…) ]  **k2: mult r2, ..., …;** [ **r2** ← (...) + (…) ]  Antroji komanda pakeičia registro **r2** turinį nauju. Aišku, kad komandos **k2** negalima vykdyti iki tol, kol **k1** nepanaudojo buvusio jo turinio. Jei procesorius turi “atsarginių” registrų, vieną iš jų (pavyzdžiui, **r33**) panaudokime komandoje **k2**:  **k1: add ..., r2, ...;** [ ... ← (r2) + (…) ]  **k2: mult r33, ..., …;** [ r33 ← (...) + (…) ]  Taip konfliktas bus išspręstas. Aišku, kad ir paskesnės komandos turi kreiptis į **r33** vietoj **r2** (kol registro pakeitimas galioja). |
| 1. Perėjimų spėjimas 2 |
| Konvejerizuotame (ir ypač – superkonvejerizuotame) procesoriuje sutikus perėjimo komandą, tenka pasirinkti, iš kurios krypties turi būti išrenkamos paskesnės komandos konvejeriui užpildyti. Kadangi perėjimo kryptis, priklausanti nuo tikrinamos sąlygos reikšmės, paaiškėja kiek vėliau, specialios schemos “nuspėja” perėjimo kryptį. Jei spėjimas nepasitvirtina, tenka stabdyti konvejerį ir jį užpildyti komandomis iš tikrosios krypties. Juo tikslesnis perėjimų spėjimas, tuo mažiau laiko sugaištama konvejeriui perkrauti. Spėti galima įvairiai:   * prognozuojant, kad perėjimo niekada nebus; * prognozuojant, kad perėjimas visada bus; * atsižvelgiant į perėjimo operacijos kodą; * naudojant jungiklį “perėjimas bus/perėjimo nebus”; * naudojant perėjimų istorijos lentelę.   Pirmieji trys perėjimų spėjimo būdai priskiriami statiniams, o du paskutinieji - dinaminiams.  **Statinė perėjimo krypties prognozė**  Straipsnyje “PC Processor Microarchitecture” teigiama, kad santykiu 4:1 dominuoja valdymo perdavimo pirmyn atžvilgiu grįžimo atgal komandos. Apie 60% iš valdymo perdavimo pirmyn komandų iš tiesų reikalauja pereiti, o net 85% grįžimo atgal komandų iš tiesų reikalauja grįžti (dėl programos ciklų realizacijos ypatumų). Tokia informacija apie perėjimų pobūdį gali būti panaudota perėjimams prognozuoti. **Statinės prognozės įtaisas** turėtų analizuoti “offset” turinį (rodantį “atstumą” esamo PC turinio atžvilgiu) iš karto, kai tik komanda dekoduojama. Kadangi grįžimo atgal komandose dažniausiai ir grįžtama, taip prognozės įtaisas ir spėtų.  Žinoma, prognozės patikimumas priklauso nuo programos ypatumų, taip pat nuo programuotojo kodavimo stiliaus.  **Dinaminė perėjimo krypties prognozė**  **Dinaminės prognozės** atveju naudojami keli perėjimo krypties spėjimo užtikrinimo būdai:   * + - ***Branch Prediction Buffer***  - perėjimo prognozės buferis arba ***Branch History Table***  - perėjimo istorijos lentelė (tai sinonimai) * ***Branch Target Buffer***  - perėjimo komandų buferis * ***Branch Target Cache***  - perėjimo komandų kešas * ***Return Address Stack*** - grįžimo adresų stekas arba ***Return Stack Buffer*** - grįžimo steko buferis   Čia perėjimo krypties spėjimui panaudojama programos vykdymo metu sukaupta informacija.  BHT - **Branch History Table** – paprasčiausia priemonė: žemiausieji PC adreso bitai naudojami kaip 1-o bito reikšmių lentelės indeksas; bitas rodo, ar paskutinįjį kartą buvo pereita (1) ar ne (0). Prognozės efektyvumui padidinti kiekvienam perėjimui skiriami du bitai. Tuomet prognozuojama kryptis pakeičiama priešinga buvusiai, jei du kartus buvo nepataikyta.  **Branch target** Kai kuriose prognozės sistemose naudojamas specialus buferis, kuriame laikomi perėjimo adresai (kartu su prognozės bitais). Procesorius, sutikęs perėjimo komandą, iš karto gali pradėti komanų išrinkimą iš perėjimo vietos pagal šiame buferyje esančią informaciją. |
| 1. Išankstinis įkrovimas 1 |
| * Nemaži laiko nuostoliai siejami su komandomis, kurioms reikalingi duomenys įkraunami iš pagrindinės atminties. * Siekiant sumažinti šiuos nuostolius, kai kuriuose procesoriuose naudojamas **išankstinis** (*speculative)* **įkrovimas**, kai specialios schemos analizuoja vykdymo laukiančių komandų seką ir, aptikę joje įkrovimo komandą, ją pradeda vykdyti nelaukiant eilės, kad pagrindinėje atmintyje esančių duomenų laukimas minimaliai trukdytų juos naudojančių komandų vykdymą. |
| 1. Daugelio gijų skaičiavimų palaikymas 1 |
| Nemaži laiko nuostoliai siejami su komandomis, kurioms reikalingi duomenys įkraunami iš pagrindinės atminties. Siekiant padidinti kompiuterio našumą, jame gali būti panaudoti įvairūs lygiagrečiųjų skaičiavimų organizavimo būdai, tame tarpe ir naudojami multiprocesorinėse sistemose:  **\*Daugiaprocesoriškumas** (*multiprocessing*) – kelių procesorių įstatymas į sistemą ir darbo paskirstymas tarp jų. Tai gerokai didina tiek pagrindinės plokštės sudėtingumą, tiek ir visos sistemos kainą.  **\*Multiprocesorius viename kristale** (*CMP – chip multiprocessing*) – du procesoriaus branduoliai fiziškai sutalpinami viename kristale, naudodami tą patį kešą arba atskirus kešus. Tokie kristalai gali būti ir keli vienoje plokštėje.  \***Daugelio gijų skaičiavimai** (*multithreading*) – procesorius perjungiamas nuo vienos gijos prie kitos.  Kadangi šiuolaikinės kristalų gamybos technologijos įgalina viename kristale sutalpinti dešimtis milijonų tranzistorių, atsirado galimybė į kristalo vidų perkelti net ir tokias sudėtingas schemas, kurios reikalingos dinaminei programos kodo analizei, gijų išskyrimui, jų sąlyginei inicializacijai bei kelių gijų palaikymui.  **\*Kvantuojami daugelio gijų skaičiavimai** (*time-slice multithreading*) – procesorius reguliariai perjungiamas nuo vienos gijos prie kitos (fiksuoto dydžio intervalais).  **\*Proginiai daugelio gijų skaičiavimai** (*switch-on-event multithreading*) – procesorius perjungiamas nuo vienos gijos prie kitos, kai vykdytoje gijoje atsiranda priverstinė pauzė (pvz., nepataikius į kešą).  **\*Vienalaikiai daugelio gijų skaičiavimai** (*simultaneous multithreading*) – procesorius vykdo skirtingas gijas “vienu metu”, be perjungimo nuo vienos gijos prie kitos. Resursai skirstomi dinamiškai (”nereikalingas tau – atiduok kitam”). |
| 1. Sudėtingų sistemų realizacija viename kristale 1 |
| Didinti dažnį, siekant vis didesnio našumo, darėsi vis sunkiau ir sunkiau. Vietoj to firmos nukreipė savo jėgas lygiagretumo didinimui - dviejų branduolių procesorių kūrimui, vėliau pereinant prie kelių branduolių viename procesoriuje. Tai yra logiškas daugelio gijų vystymo rezultatas. Šia kryptimi eina Intel, AMD, Motorola, Sun ir kitos firmos. Minėti kristalų gamybos technologijų pasiekimai sudaro galimybes viename kristale integruoti ir kitas kompiuterio sudėtines dalis pavyzdžiui, duomenų mainų ir periferinių įtaisų valdymo įtaisus. Tai ne tik kompiuterio branduolį daro kompaktiškesniu, bet ir žymiai padidina jo darbo spartą, kadangi duomenų mainai viename kristale vykdomi žymiai sparčiau, nei tarp kristalų. Kita vertus, viename kristale galima realizuoti du ar daugiau procesorių, sudarant prielaidas galingoms multiprocesorinėms sistemoms kurti. |
| 1. Procesorius. Jo valdymo ir operacinis įtaisai 2 |
| Dabar komandų ir duomenų procesorius įprasta apjungti į vieną įtaisą – **centrinį procesorių** (CPU).  Todėl komandų ir duomenų procesorius toliau vadinsime **valdymo** ir **operaciniais įtaisais**, nes šie pavadinimai geriau atspindi jų paskirtį ir funkcijas  **valdymo įtaisas** išrenka iš atminties komandą, ją analizuoja ir valdo operacinio įtaiso darbą (jame vykdomas operacijas, kreipinius į atmintį duomenims išrinkti ar rezultatui įrašyti);  **operacinis įtaisas** vykdo operaciją, kurią nurodo komanda;  šie du įtaisai dirba kartu: valdymo įtaisas pagal operacijos kodą formuoja signalus, valdančius operacinio įtaiso darbą; pastarasis perduoda į valdymo įtaisą signalus, informuojančius apie operacijos eigą, nuo kurių gali priklausyti paskesnių valdymo įtaiso signalų formavimas (pavyzdžiui, operando ženklas, jo kurios nors skilties reikšmė ir t.t.).  Jei pažvelgtume į operacinio įtaiso vidų, galėtume jame išskirti dvi schemų grupes:   * **vidinę atmintį**, kuri reikalinga apdorojamiems duomenims (operandams) laikyti; ją sudaro registrai, atskiri trigeriai, spartinančioji atmintis [kešas], kai kuriuose įtaisuose – stekas; * **operacijas vykdančios schemos**, kurios atlieka visus informacijos apdorojimui reikalingus veiksmus – sudėtį, logines operacijas, postūmius ir t.t. |
| 1. Šiuolaikinių procesorių struktūra 3 |
| Programos kodo “aktyvioji” dalis būna L1 komandų keše. Kai kuriuose CPU čia laikomos komandos pradiniame pavidale, kituose – iš dalies ar visiškai perkoduotos. Jei šiame keše reikalingų komandų nėra, jos paimamos iš L2 kešo ir, jei reikia, perkoduojamos. Šios komandos imamos blokais, naudojant išankstinį išrinkimą. Blokas nukreipiamas į du įtaisus – komandų perkodavimo ir perėjimų prognozės  Perkodavimo įtaisas jas paverčia mikrooperacijomis (MO), o perėjimų prognozės įtaisas nustato, ar taro jų yra perėjimo komandos, ar bus pereinama ir kokiu adresu. Perkoduotos MO patenka į registrų pakeitimo bei resursų išskyrimo įtaisus.  ris1Registrų pakeitimo įtaise MO gali būti suteiktas rezervinis registras jos vykdymo rezultatui saugoti. Vėliau vykdomos MO tai įvertins ir operandą ims iš tokio registro. Informacija apie tokius registrų pakeitimus laikoma specialioje lentelėje. Po perkodavimo ir registrų pakeitimo MO grupė įrašoma į ROB (tvarkos pakeitimo buferio) eilės galą. Šiame buferyje saugomos MO ir pagalbiniai duomenys iki pat užbaigimo momento. Tuo pačiu metu šo MO grupė patenka ir į rezervavimo įtaisus (*Reservation Station* ), kuriuose sekama MO vykdymo galimybė nepriklausomai nuo MO patekimo eilės tvarkos  Kai planuotojas nukreipia MO vykdymui į atitinkamą FĮ, ši MO pašalinama iš planuotojo eilės, o įvykdžius operaciją daroma apie tai atžyma ROB. Kai prieš duotąją MO visos buvusios MO jau įvykdytos ir apie tai pažymėta jas pašalinant iš ROB, ši taip pat iš jo pašalinama. Kiekviena planuotojo eilė turi ryšį su vienu ar keliais specializuotais FĮ. Vieno takto metu ivairiuose procesoriuose nukreipiama vykdymui nuo 5 iki 10 MO. Be ALU, procesoriuose būna ir įkrovimo/įrašymo (*Load/Store*) įtaisai, kurie vado kreipinius į kešus ir pagrindinę atmintį (struktūrose jie dažnai nerodomi |
| 1. Koprocesorius ir jo ryšys su procesoriumi 2 |
| Pirmieji mikroprocesoriai atliko veiksmus tik su sveikaisiais skaičiais, o veiksmai su slankaus kablelio skaičiais buvo emuliuojami, t.y., realizuojami programiniu būdu (naudojant specialią paprogramių biblioteką).  Intel sukurtasis koprocesorius **8087** (dar vadinamas ***matematiniu koprocesoriumi***, ***aritmetiniu koprocesoriumi***) skirtas veiksmams su slankaus kablelio skaičiais ir kai kurioms specialioms operacijoms (sinuso, kvadratinės šaknies ir kt.).  Komandų išrinkimą valdo 8086. Kai koprocesorius pastebi, kad išrenkama slankaus kablelio komanda, jis ją pasiima ir vykdo  Magistralę pasidalija naudodami RQ/GT (*Request/Grant*) liniją |
| 1. Intel procesoriai. Pagrindiniai jų bruožai 2 |
| **Intel 4004, pirmasis mikroprocesorius (1971 lapkr.).**  skirtas kalkuliatoriui; duomenų žodis - 4 bitai, komandų ilgis - 8 bitai, atskiros atmintys: 1 KB - duomenims, 4 KB – programoms, PC ilgis - 12 bitų, 16 registrų (po 4 bitus), komandų skaičius - 46;  **Intel 8086.** programiškai suderinamas su 8080, turi panašų registrų rinkinį, duomenų ilgis išplėstas iki 16 bitų, 6 baitų išankstinio komandų išrinkimo eilė, adresuojama atmintis - 1 MB, Šio procesoriaus komandų sistema tapo daugelio procesorių (ir ne tik Intel) pagrindu. Apibendrintai ji vadinama **x86** komandų sistema.  **Intel 80386** išplėstos adresavimo galimybės, pridedant indekso daugiklį, apsaugos režimai (vadinamieji žiedai), adresuojama atmintis - 4 GB, Superskaliariškumas, Perejimo prognoze  **Pentium 4 (2000)** 7 kartos procesorius |
| 1. 64 bitų procesoriai 2 |
| **Žodžio ilgio faktorius:**   * 32 bitų procesorius gali operuoti su sveikaisiais skaičiais siekiančiais 232 arba apie 4.3 milijardo; 64 bitų procesoriaus galimybės siekia 264 arba apie 18.4 kvintilijono (18,400,000,000,000,000,000); * 32 bitų procesoriai ir operacinės sistemos gali palaikyti iki 4 gigabaitų atminties, iš jų taikomajai programai gali būti skirta tik 2 gigabaitai; CAD/CAM ir moksliniams skaičiavimams to ima nebepakakti; Athlon 64 palaiko 40 bitų atminties registrus, taip padidinant atminties palaikymą iki 164 gigabaitų   Perėjimas prie 64 bitų leidžia išplėsti sveikųjų skaičių diapazoną (264 tai nėra taip svarbu) ir adresavimo galimybes (tai gerokai svarbiau) . Gaila, bet našumas čia visiškai nesusietas su žodžio ilgio padidėjimu. Operacijų atlikimo požiūriu – netgi priešingai: veiksmai su didesnio ilgio operandais trunka ilgiau.Norint pilnai išnaudoti 64 bitų potencialą. Tai reiškia, kad reikalingos 64 bitų operacinės sistemos ir taikomosios programos. Situacija keičiasi ne taip greitai. Atminkime, kad Microsoft prireikė 10 metų perėjimui nuo 16 bitų prie 32 bitų.  64 bitai leidžia naudoti gerokai daugiau virtualios ir fizinės erdvės nei x86, padvigubina sveikųjų skaičių registrų plotį nuo 32 iki 64 bitų, padidina sveikųjų skaičių registrų kiekį ir teikia kitus privalumus |
| 1. Kelių branduolių procesoriai 2 |
| **Del ko du branduoliai:**   * Procesorių dažnį didinti darosi vis sunkiau * 2004 metais Intel paskelbė, kad 4 GHz procesoriaus Pentium 4 nebus * Vietoj to firmoms tenka ieškoti kitų priemonių procesorių našumui didinti: didinti kešo talpą, didinti magistralės darbo spartą * Pastaruoju metu procesorių gamintojai pradėjo leisti **dviejų branduolių** procesorius   Dviejų branduoliu panaudojimas nėra visai paprastas dalykas:   * dviejų branduolių procesorius naudos maždaug 2 kartus daugiau energijos nei paprastas; * Jei norime atnaujinti (*upgrade)* sistemą įstatydami procesorių su 2 brand, energijos ir aušinimo balansai gali būti pažeisti; * Jei sistema suprojektuota su atitinkama atsarga – tai galima daryti (regis, taip suprojektuoti kai kurie Sun serveriai); * Jei 20-30% sumažinti dažnį ir optimizuoti struktūrą mažesnėms energijos sąnaudoms, atnaujinimas taptų galimu. |
| 1. Šiuolaikinių procesorių architektūros ypatumai (Intel ir AMD pavyzdžiu) 2 |
| Intel naujas mikroprocesorių architektūras įveda kas 2 metai kaip jų “tick-tock” strategijos dalį:  Tarkime, vienais metais Intel pagerino technologiją, nekeisdama procesoriaus mikroarchitektūros (“*tick*”) ;Kitame etape (“*tock*”) Intel palieka gerai įsisavintą technologiją, bet kuria naują procesoriaus mikroarchitektūrą: **Intel Haswell**  Pagrindiniai bruožai: platus panaudojimo diapazonas; aukštas branduolio našumas; nedidelės energijos sąnaudos  Tolesnis AVX išvystymas: naujos AVX2 komandos su FMA ir 256 bitų sveikųjų skaičių vektoriais  bitų manipuliavimo komandos; TSX (tranzakcinė atmintis) geresniam gijų panaudojimui  Energetinis efektyvumas: mikroarchitektūra: mažiau galios, įvairesni tuščios eigos režimai;  įvairesnės galios ir našumo valdumo galimybės  Nuo Nehalem mikroarchitektūros įvedimo **AMD** ėmė gerokai atsilikti našių procesorių srityje  Nors AMD inžinieriai pirmieji yra sumanę ir panaudoję tokius dalykus, kaip 64 bitų x86 mikroarchitektūra, kelių branduolių procesoriai, atminties kontrolerio integravimas procesoriuje  Manoma, kad AMD išgelbės naujoji idėja – kurti specialius įtaisus skaičiavimams spartinti, kuriuos jie pavadino **APU** (*Accelerated Processing Units*), kai viename kristale apjungiami tradiciniai x86 branduoliai su našiu grafikos branduoliu  Pirmasis žingsnis – **Fusion**, kuriame apjungti centrinis ir grafikos procesoriai  Intel ir AMD, rungtyniaudamos procesorių kūrime, kasmet pateikia vis naujus produktus |
| 1. RISC ir CISC procesorių palyginimas 1 |
| **CISC - *Complex Instruction Set Computing - sudėtinga komandų sistema.***  **RISC - *Reduced Instruction Set Computing - supaprastinta komandų sistema.***  **Prielaidos CISC naudojimui :**  trumpesnė programa (pvz., 8080 - nėra daugybos); lengviau sukurti kompiliatorių.  **Pasekmė - kuriami kompiuteriai, orientuoti aukšto lygmens programavimo kalboms palaikyti.**  **CISC:**   * komandų skaičius – kiek galima daugiau; \*adresacijos būdų skaičius – kiek galima daugiau; * komandų formatų skaičius –kiek galima daugiau; \*komandų trukmė taktais – CPI>1; * į atmintį kreipiasi įvairių tipų komandos; \*yra specialios paskirties registrai; * valdymas – mikroprograminis; \*komandos būna aukštesnio lygio (artimos ALPK).   **CISC po bandymu**   * sunku pasiekti komandų visišką atitikimą ALPK operatoriams; * CISC atveju sunkiau optimizuoti kodą, užtikrinti efektyvų konvejerio darbą; * kompiliatoriaus sugeneruotame kode panaudojama tik dalis komandų; * programos ilgis (komandų skaičius) svyruoja nedaug.   **RISC**   * trumpos komandos (dauguma įvykdomos per 1 ciklą) * registras-registras tipo komandos (*load/store* architektūra) * paprastesni adresavimo būdai * paprastesni komandų formatai * efektyvūs kompiliatoriai \*efektyvus konvejerizavimas |