1. **Fon Neimano ir Harvardo architektūros**

Fon Neimano: mainai tarp kompiuterio įtaisų vykdomi per vienintelę magistralę. centrinis procesorius (CPU). CPU, atmintis, įvesties – išvesties įtaisas, magistralės. Dvejetainio kodavimo principas. Programinio valdymo principas. Vienalytės atminties principas. Adresavimo principas.

Harvardo: programos saugomos pastoviosios atminties (ROM) įtaisuose, o duomenys – laisvo išrinkimo atminties (RAM) įtaisuose.

1. **Procesoriaus struktūra. Valdantysis ir operacinis įtaisai**

Duomenų procesorius ir komandų procesorius:

***- T*** - laikinojo saugojimo (papildomas) registras,

***- IR*** - komandos registras,

***- PC*** - programos skaitiklis,

***- M*** - atmintis,

***- AR*** - jos adreso registras,

***- DR*** - jos duomenų registras.

Valdymo

įtaisas

Operacinis

įtaisas

PROCESORIUS

Informacija

apie opera-

cijos eigą

Valdymo

signalai

Komanda

(iš atminties)

Duomenys

(iš atminties)

Rezultatai

(į atmintį)

Vidinė

atmintis

(registrai,

spart. atmintis)

Operacijas

vykdančios

schemos

Operacinis įtaisas

rezultatai

operandai

Valdymo

signalai

(iš vald.įtaiso)

Informacija

apie opera-

cijos eigą

(į vald.įtaisą)

1. **Kompiuterių struktūrų raida**

Pirma karta: Procesorius – kartu ir centrinis valdymo įtaisas: jis ne tik interpretuoja programą ir vykdo komandas, bet ir valdo pradinių duomenų įvedimą bei rezultatų išvedimą

Antra karta: Įdėta bendra magistralė

Trečia karta: Duomenų įvedimą bei rezultatų išvedimą (taip pat ir mainus su išorine atmintimi) valdo specialūs įtaisai, vadinami kanalais arba įvesties ir išvesties procesoriais. Procesorius interpretuoja programą ir vykdo komandas bei valdo kanalų darbą.

1. **Komandų sistema. Pagrindiniai komandų sistemų tipai**

Skaičiavimams reikalingą operacijų rinkinį kompiuteryje realizuoja **komandų sistema**, kurioje vieną operaciją gali atitikti ir kelios komandos, besiskiriančios operandų tipais, adresacijos būdais ir pan.

**Aritmetinės ir loginės, Duomenų persiuntimo, Valdymo, Slankaus kablelio, Dešimtainės, Eilučių apdorojimo, Sisteminės**

**Vidinės CPU atminties tipas yra pagrindinis skiriamasis požymis. Čia yra 3 alternatyvos:**

**1) stekas,**

**2) akumuliatorius,**

**3) registrų grupė.**

1. **Akumuliatoriaus tipo komandų sistema**

**Akumuliatorius**: specialus registras (kaupiantysis registras), į kurią informacija įrašoma prieš operaciją; iš jo rezultatai grąžinami į atmintį.

vienas iš operandų saugomas akumuliatoriuje, rezultatas – taip pat, operandų išrinkimas ir įrašymas – load, store.

1. **Steko tipo komandų sistema**

**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*)**

operandai saugomi steke, rezultatas – taip pat, operandų išrinkimas ir įrašymas – procedūros **push**, **pop**

1. **Bendros paskirties registrų tipo komandų sistema**

**Registrų failas**: 8-16-32-… registrų grupė procesoriuje, į kuriuos vienas iš operandų įrašomas prieš operaciją; kitas operandas imamas tiesiai iš atminties. operandai saugomi registruose, rezultatas – taip pat, operandų išrinkimas ir įrašymas – load, store.

1. **Komandų formatai. Operandų nuorodos**

OpK Op1 Op2 ... Opn. bendrosios paskirties registrų tipo architektūra.

OpK – operacijos kodo laukas,

Op1, …, Opn – operandų nuorodos

**IBM/360, Intel x86, Power MPC601,**

Operando nuorodos pavidalas priklauso nuo to, koks yra operandas, kur jis saugomas. Operandu gali būti: kurio nors registro turinys, atminties ląstelės turinys, konstanta, adresas, sąlyga.

1. **Operandų adresavimo būdai**

***Registrinė*** adresacija Add R4, R1 Reg[4] := Reg[4] + Reg[1]

***Netiesioginė registrinė***

***Santykinė*** *(****bazinė*** arba ***su poslinkiu***)

***Indeksinė***

***Indeksinė su koeficientu***

***Tiesioginė*** arba ***absoliutinė***

***Betarpiškoji***

***Autoinkrementinė***

***Autodekrementinė***

1. **Komandų sistemos išplėtimai procesoriuose**

**Išplėtimo priežastys ir prielaidos** (BETARPIŠKAS TECHNOLOGIJŲ TOBULĖJIMAS)

* procesoriai buvo orientuoti į sveikųjų ir slankaus taško skaičių apdorojimą
* grafinės ir audio informacijos skaitmeninio apdorojimo plitimas
* technologijos vystymasis ir normos mažinimas nuo 0,35 μm iki 0,13 μm leido gerokai padidinti tranzistorių skaičių kristale
* RISC architektūros branduolys kompaktiškas - realizuojamas palyginti nedideliu tranzistorių skaičiumi
* 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

MMX(ženklo bitas ir visi eksponentės bitai nustatomi į 1)

SSE, KNI, 3Dnow!, AltiVec, VIS, MAX-2, AVX

1. **Multimedijos informacija ir jos apdorojimo specifika**

Pikselių kodavimas: 8 bitų registras pixeliui, 8 bitų registras pixelio intensyvumui(kai pilka), 12 bitų spalvų pikseliai(RGB), 32 bit spalvų pixeliai(alfa RGB)

Apdorojimas - sudėtis

1. **Konvejerio ir superkonvejerio esmė**

Konvejeris – mašinų komandos seką vykdantis elementas. 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.

Power PC – 3 konvejeriai – dekodavimo, vykdymo ir įrašymo

* Kiekvienai IU komandai dekoduoti pakanka 1 ciklo
* Po dekodavimo – vykdymas sveikųjų skaičių konvejeryje
* **mul** komandai vykdyti reikia 5 ciklų, tad **cmp** negali būti vykdoma 5-jame cikle; todėl ji patenka į IU buferį ir ten pasilieka, iki atsilaisvins vykdymo įtaisas
* Todėl **add** komanda lieka dekodavimo pakopoje

Superkonvejeris:

F1, D2, D, O, X1, X2, W1, W2

1. **Konvejerio kliūčių tipai**

***struktūrinės kliūtys*** *(nepakankami resursai)*

***duomenų kliūtys*** *(komandos vykdymo rezultatai naudojami kitoje komandoje kaip operandai)*

***valdymo kliūtys*** *(būtinumas išrinkti komandas iš kitos vietos)*

1. **Duomenų kliūtys ir jų įveikimas**

Vykstant dvejams procesams, esant dviems kintamiesiems, vieno kintamojo reikšmė gali kisti tuo pat metu, kai kitas procesas turi naudoti šio kintamojo reikšmę

Išeitys: programinės, aparatinės

1. **Valdymo kliūtys ir jų įveikimas**

Laiko nuostolis. Atpažinus branch, konvejeris blokuojamas

* 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**

našumas yra atvirkščiai proporcingas sprendimo laikui, galime rašyti: **Našumas ≡ 1/ TCPU**

**Našumas** ≡ **F / CPI**

našumą galima didinti dviem būdais:

1. didinant darbo dažnį,
   1. gamybos proceso išvystymo laipsnis
   2. procesorių mikroarchitektūros ypatumai.
2. mažinant CPI – tobulinant procesoriaus architektūrą
   1. komandų realizacijos algoritmų išvystymas
   2. didesnės apimties veiksmų atlikimas vieno ciklo metu.

Dašumo didinimo priemonės:

* technologijos normos mažinimas,
* superskaliariškumas,
* superkonvejerizacija,
* specializuoti funkciniai įtaisai (MMX, SSE, …),
* registrų pakeitimas (*register renaming*),
* komandų vykdymo eilės tvarkos pakeitimas,
* kristale esančios atminties talpos didinimas ir integruotos spartinančios atmintys,
* 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*).

1. **Komandų vykdymas ne eilės tvarka**

Load, Store, ALU, Shift, Branch (Superskaliariniam procesoriuje)

Load ir Store užtikrina mainus su atmintim

ALU gali būti keli

* 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.** Jis būna tarp konvejerio F ir X pakopų; jame laukia vykdymo jau išrinktos komandos.

Komandų langas gali būti:

* **centralizuotas** – bendras visiems funkciniams įtaisams
* **paskirstytas** – kiekvienas funkcinis įtaisas turi savo langą

1. **Registrų pakeitimas**

**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**

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.

1. **Daugelio gijų skaičiavimų palaikymas**

**Daugelio gijų skaičiavimai** (*multithreading*) – procesorius perjungiamas nuo vienos gijos prie kitos.

* **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**

kristale integruoti ir kitas kompiuterio sudėtines dalis (pavyzdžiui, duomenų mainų ir periferinių įtaisų valdymo įtaisus arba schemų rinkiniams, angiškai vadinamiems terminu *chipsets*), būdingas funkcijas.

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. **Šiuolaikinių procesorių struktūra**

Dabar komandų ir duomenų procesorius įprasta apjungti į vieną įtaisą – **centrinį procesorių** (CPU).

Valdymo įtaisas. Operacinis įtaisas.

1. **Komandų vykdymas šiuolaikiniuose procesoriuose**

**valdymo įtaisas** išrenka iš atminties komandą, ją analizuoja ir valdo operacinio įtaiso darbą

1. **Koprocesorius ir jo ryšys su procesoriumi**
2. **Intel procesoriai. Pagrindiniai jų bruožai**
3. **64 bitų ir kelių branduolių procesoriai**
4. **Šiuolaikinių procesorių architektūrų ypatumai (Intel ir AMD pavyzdžiu)**
5. **RISC ir CISC procesorių palyginimas**