SPM → L.1

Alexandru Ghiondea → alexandra.ghiondea@gmail.com
F. imp. → curățenia în lab.!!!
Temele → în ROMÂNĂ!

4 teme: individuale
- net research
- simulare (interfață grafică)

Putem face și .NET, la. vrem, sau numai Linux. (stabilim săpt. viitoare)
Prezența nu e obligatorie, dar cei care vin primesc 1 pct. (din 10)

Pipeline → instrcț. e împărțită în 5 stadii ⇒ după ce pr. instrcț. a trecut de stadiul 1, a 2-a instrcțiune poate intra și ea în execuție.

Fiecare instrcț. care trece prin pipeline ~~este cel mai mult în pipeline~~ are cel mai mare timp de execuție (în rap. cu celelalte, care intrarea în pipeline pe rând după ce pr. terminase stadiul 1 al execuției ⇒ după ce iese prima instr. din pipeline, celelalte vor ieși la intervale ~~egale~~ mult mai mici, pt. că fiecare mai are de exec. doar 1 stagiu față de cea anterioară).

2 tendințe:
- super-pipeline (instr. e fragmentată ⇒ f. multe)
- super scalar (nr. mic de stadii ale instr., dar se multiplică unitățile funcționale)

uneori stagii dif. ale pipeline au nevoie de aceeași unit. funcționale → de. e o sg. unitate funcț. ⇒ apare pb. priorităților ⇒ de. sunt mai multe u. funcț. nu mai apare —.

Procesor multinucleu ⇒ 2 µP (sau mai multe) pe același chip ⇒ poate exec. 2 programe simultan.

De. sunt 2 pastile de Si, comunic. între ele Nu. se trec prin placa de bază (la sist. multiprocesor). De. sunt pe a. pastilă de Si,

comunic. se face direct.

ms.dnaa.microsoft-lab.pub.ro → produs microsoft gratis de luat
Windows, Office, Visual Studio etc

Hiperthreading → 2 pipeline, 1 unit. de exec. care comută între ele. Ptr. că avem o comutare foarte-rapidă se creează impresia că se execută 2 operații în același timp.

∃ 2 tipuri de sisteme care folosesc proc. multiple < multiprocesor / multicomputer

Sist. multiprocesor
- SMP (Simetric MultiProcesor) → proc. în număr de 2^n; memorie comună; MP sunt conectate la fels sunt fol. în sist. comerciale; MP au același rol în sistem.
- MPP (Masive Parallell Procesor) → mașini științifice construite special ptr. a rezolva o anumită ptb; au în jur de 1000 MP.

Sist. multicalculator → sunt o conexiune între mai multe calculatoare de același fel → fiecare calc. are memoria lui (nu mai e necesară partajarea de mem.). Calc. sunt unit. indep. → se pot extinde pe dist. geografice f. mari. Aplicație: gridcomputing (?) → prj. Monalisa, N. Țăpuș.

Concursuri internaționale:
- CSIDC (.com) ↳ design | Microsoft
- Imagine Cup ↑ Windows Embedded Student Challenge

Metode de evaluare a procesoarelor: nu se poate crea un progr. care să meargă pe toate platformele și să dea rez. concludente ⇒ ∃ algoritmi universali, bazați pe legea lui Amdahl.

↓

câștigul de performanță al unui sistem după o anumită îmbunătățire e proporțional cu timpul de utilizare a îmbunătățirii respective.

FA = factor de accelerare = $\frac{\text{timpul de execuție vechi}}{\text{timpul de execuție nou}}$ → det. performanța sist. nou față de cel vechi.

FÎmb = fracțiunea de îmbunătățire

FAÎmb = factor de accelerare a îmbunătățirii

$$FA = \frac{1}{(1 - F\hat{I}mb) + \frac{F\hat{I}mb}{FA\hat{I}mb}}$$

TEx = timp execuție al unui task de referință

TEx = durata unui ciclu mașină * nr. de cicli mașină necesari unei instrucțiuni * nr. (de) instrucțiuni

↓

util la pipeline.

Pb.: Avem o arhitectură în care 20% din timpul de rulare se efect. instr. de extragere a radicalului în virgulă mobilă, iar 50% din timpul de exec. se ef. alte instr. în virgulă mobilă. E mai bine să accelerez de 10 ori op. de √ sau să acc. de 2 ori op. a 2-a? Id. FÎmb, FA, FAÎmb.

$FA_1 = \frac{1}{0{,}82}$; $FA_2 = \frac{1}{0{,}75}$

↓

a 2-a e mai bună.

$F\hat{I}mb_1 = 0{,}2$; $F\hat{I}mb_2 = 0{,}5$

$FA\hat{I}mb_1 = 10$; $FA\hat{I}mb_2 = 2$

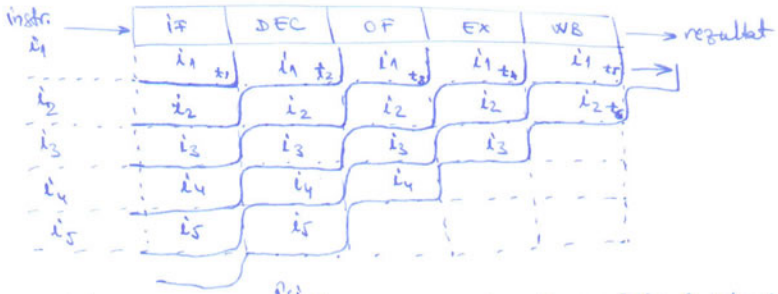
↓

de câte ori vreau să acc.

T.1: www.top500.org ⇒ referat despre calc. de pe locul 2 → performante, la ce e utilizat, cum e conectat, etc. cate calculatoare contine, FLOPS, etc. ⇒ peste 2 sapt, 3÷5 pg.

SPM. → L.2

Pipeline → introdus de Ford
→ instr. impartita in 5 etape
- instruction fetch IF
- decode DEC
- operand fetch OF
- execute EX
- write back WB



⇒ instructiunea i_2 se termină execuția după 6 u.t.; dar lucra secvential, ar fi terminat după 10 u.t.!

Secvential: n instr. ⇒ $n \cdot T$ = timp de execuție

Pipeline: n instr. → $T + (n-1)t_p = T_{ex}$
- t_p → durata unui pas în pipeline (se ia cel mai mare ca fiind consum de timp)
- n-1 → nr. de [redacted] instrucțiuni
- T → timpul necesar unei instr. să treacă prin pipeline

Pipeline
- de instrucțiuni
- aritmetice

Pipeline
- liniar
- dinamic → se reconfigurează în fcție. de necesități

$FA = \frac{T_V}{T_N} = \frac{T}{\frac{T}{n}} = n$; T = timp de trecere a unei instr. prin pipeline

Nu se poate asigura un flux continuu de instr. în pipeline (∃ instr. de salt ⇒ se golește pipeline-ul până se det. unde se face saltul) ⇒ ⇒ prin arct. pb. a apărut mec. branch prediction ⇒ la Pentium era eficient în măsură de 96%.

Pb.1: Mașină secvențială cu ciclu de ceas de 10 ns. Ea execută 3 tipuri de operații: → 40% aritmetice, și durează 4 cicli de ceas
→ 20% salt, și durează 4 cicli de ceas
→ 40% op. cu memoria, și durează 5 cicli de ceas

Dc transform a. mașină într-un pipeline ideal cu 5 trepte, dar cu o întârziere de 1 ns datorită transferului între trepte, care e FA între mașina secv. și cea pipeline?

$T_{ex} = T_{ck} \cdot \sum N \cdot NI$
- T_{ex}: timpul de exec.
- T_{ck}: durata unui ciclu mașină
- $\sum N$: nr. de cicli necesari unei instrucțiuni
- NI: nr de instrucțiuni

$T_V = 40 \cdot (10 \cdot 4) + 20 \cdot 10 \cdot 4 + 40 \cdot 10 \cdot 5 = 1600 + 800 + 2000 = 4400$

$T_N = 54 + 99 \cdot 11 = 54 + 1089 = 1143$

$FA = \frac{4400}{1143} = 3,85$

sau $T_V = 10(0,4 \cdot 4 + 0,2 \cdot 4 + 0,4 \cdot 5) \cdot n$

$T_N = (10+1) \cdot 1 \cdot n$ (putem neglija T_{exec} pt instr prin pipeline — e neglij. în rap. cu T_{exec} a instr. —)

Pb.2: Avem o mașină Pipeline cu 5 trepte neech. dpdv. al performanței → I: 10ns, II: 8ns, III: 10ns, IV: 10ns, V: 7ns. ∃ o întârziere de 1 ns între trepte (transfer). De câte ori e mai rapidă a. mașină pipeline decât o mașină secv. echiv.?

$$FA = \frac{T_{secv.}}{T_{pipeline}} = \frac{T_s}{T_p}$$

$$T_p = (10+1) \cdot 1 \cdot n = 11n$$

$$T_s = (\underset{I}{10} + \underset{II}{8} + \underset{III}{10} + \underset{IV}{10} + \underset{V}{7}) \cdot n = 45n$$

$$\Rightarrow FA = \frac{45}{11} \approx 4,1$$

La secvențial nu mai apare întârzierea de la transferul între trepte în pipeline.

Hazard: avem o instr. în WB și una în OF ⇒ ambele se exec. simultan, dar prima tb. să scrie în locația A, iar a 2-a scrie în locația A ⇒ care are loc prima?

Hazard:
- structural ⇒ acces concurent la anumite resurse hw.
- de date ⇒ — " — " — " — aceleași date
- de comandă ⇒ salturi, bucle, decizii → nu știu ce urm. să intre în pipeline.

Interdependențe între instrucțiuni:
- Write after Read
- Write after Write
- Read after Write

Hazard de comandă: 3 aspecte imp.:
- detectarea
- controlul
- evitarea
  - statică (la compilare)
  - dinamică (la rulare)

În cz. salturilor ∃ 2 opțiuni:
- predicție a saltului
  - static
  - dinamic
- procesezi ambele ramuri, apoi renunți la cea de care nu ai nevoie

Hazardul structural se rezolvă prin dublarea/multiplicarea unităților hw la care apare concurența.