SPM.5

Cap. 2: Procesoare Pipeline

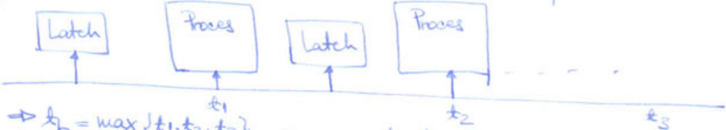
- introducere:

[ppl = pipeline]

→ totul e realizat prin hw. și programare ⇒ programatorul nu tb. să facă nimic;

→ ptr. ca struct. ppl. să fie fiabilă se exploatează paralelismul la niv. instrucțiune → granularitate fină;

→ o struct. ppl. e compusă din trepte (stages) & fiecare treaptă e compusă dintr-un circuit de prelucrare și un latch (dispozit. de memorare);



⇒ $t_b = \max\{t_1, t_2, t_3\}$ → programatorul vrea ca $t_1 \cong t_2 \cong t_3$

Dc. acest lucru nu se poate: • treapta lentă e dublată sau triplată a.î. aici să aibă loc 2/3 execuții în || ⇒ prin dispozitive speciale se face sincronizarea cu celelalte trepte.

- treapta lentă e divizată în trepte mai mici

Treapta cea mai lentă e pct. de strangulare – bottleneck.

pg. 6: P = frecv. / perioada de ceas

În primele m unit. de timp ceasul bate, dar la out. ppl. nu se regăsește nimic ⇒ tb. să treacă m perioade de timp P până iese pr. instr. din ppl. (pr. instr. tb. să parcurgă ppl.). Ptr restul de n-1 taskuri vom avea rezultat la out. ppl. la fiecare tact al ceasului.

$\tau_i = p$ = timpul de execuție a segm. i

O struct. ppl. poate fi: • statică: dc. nu își schimbă configurația pe durata de funcționare. → doar un sg. tip de cmpo de instrucț. se poate exec. în același timp;

- dinamică: își schimbă configurația ptr. a putea exec. op. diferite (de ex. adunarea & înmulțirea în VM.) → într-o struct. ppl. dinamică pot coexista în ppl. 2 instrucț. în execuție.

Drain = golire completă a ppl. static de instrucț. și introducerea unui alt tip de instruct.

∃ sist. în care struct. ppl. e golită în mod forțat → în cz. exec. instr. de salt → Flush (se ia instr. de la adr. de salt).

pg. 11: Struct. ppl. dinamică:

Drain Flush Collision ⇒ în exemplu, când 1 și 2 generează simultan rezultat către 3.

Pipeline de instrucțiuni: 5 trepte:

IF: extragerea cod. instr. de la adr. indicată de IP (Instr. Pointer); după extragere – IP++;

ID: decodif. instr. → poate fi necesară extragerea de operanzi

OF: extragere de operanzi:
- din mem. program cu IP (de abia l-am incrementat) ⇒ și aici se increm. de fiec. dată când tb. să extragă op. a.î. să fie gata;
- din registre ⇒ și reg. sunt indicate în codul instr. → citim reg. și nu avem nevoie de IP;

Ex: execuția

WB: după execuție info. e scrisă în
- registres
- memorie → nu fol. IP (un program nu se automodifică) ⇒ e posibil ca la op. extrasă cu OF să am adr. de mem. unde tb. pus rezultatul

∃ 3 modalități de îmbunătățire a ratei de transfer a unui μP ppl. ptr. instrucțiuni:
- Fetching: tb. să citesc rapid din memorie ⇒ am nevoie de mem. rapidă ⇒ costă ⇒ facem un fel de mem. cache a.î. ppl. să fie alim. cu instr.;
- Bottleneck: unele sgm. din struct. ppl. lucrează mai încet ⇒ vezi sol. pg. 13.
- Issuing: pot ∃ instr. care nu pot fi executate (datorită hazardului).

Hazard
- de dte.
- structural
- de comandă

Hazard structural ⇒ ∃ instr., dar struct. hw. nu dispune de resursele suficiente ~~necesare~~ ai. instr. să se poată executa.

Hazard de dte. ⇒ ∃ instr. de exec., ∃ și hw. care să o execute, dar execuția instr. necesită dte. anterioare care nu ~~au fost~~ sunt disponibile încă.

Hazard de comenzi ⇒ instr. de salt ⇒ tot ce s-a introdus în ppl s-a introdus până la salt ⇒ tb. să facem Fetch.

pg.21: când compilatorul prevede un data dependency dictează introducerea unei întârzieri în i_2 ⇒ între i_1 și i_2 tb. introdusă o întârz. de 2 . . . ⇒ se introduce NOP între i_1 și i_2

pg.27:

WAW
- când avem un buffer la out. în care se pun rezultatele
- când avem mai multe ppline-uri
- când compilatorul transformă programul scris de programator în lbj. de niv. înalt ⇒ se face dynamic data dependency ~~check etc~~ check

~~2 metode~~
- ~~Tomasulo (?): pg.30 → pg.39 → la var. a) au concurență ptr. RS. WAN și sumatorul și multiplicatorul vor să scrie în RS ⇒ b) e~~
- ~~Scoreboard~~

2 metode:
- Tomasulo: (pg. 30) ⇒ pg.39 → la var. a) am concurență ptr. RS. WAW și sumat. și multiplicat. vor să scrie în RS ⇒ b) e mai bună
- Scoreboard: (pg.40) → instantaneu, la un mom. de timp dat
  → se păstrează până când apar modificări, apoi se modifică.

SPM.6

pg.41: Scoreboard:

Met. folosește niște tbl.:
- Function Unit Status
- Destination Register Status

↓

arată reg. în care se scriu informații și unitatea funcțională în care se