ZKOUŠKA Z ARCHITEKTUM POČÍTAČŮ 2021

Filip Kaslr

1) SEKVENČNÍ ÚKOL PŘIDÁVÍM PROCESORŮ ~~JÁDER~~ NEZRYCHLÍME. PROTO RESPONSE TIME ZŮSTANE STEJNÝ. THROUGHPUT SE VŠAK ZVĚTŠÍ, PROTOŽE S VÍCE JÁDY PROCESORU MŮŽEME VYKONAT VÍCE TĚCHTO ÚKOLŮ NAJEDNOU.

2) $t_1 = \frac{1{,}5\,CPI \cdot n}{2\,GH_z}$ (n: počet instrukcí)

$t_2 = 80\% \cdot t_1$

$t_2 = \frac{120\% \cdot 1{,}5\,CPI \cdot n}{?\,GH_z}$

$\frac{0{,}8 \cdot 1{,}5 \cdot n}{2} = \frac{1{,}2 \cdot 1{,}5 \cdot n}{x}$

$\frac{0{,}8}{2} = \frac{1{,}2}{x}$

$x = \frac{2 \cdot 1{,}2}{0{,}8} = \frac{2{,}4}{0{,}8} = 3$

PROCESOR MUSÍ PRACOVAT NA FREKVENCI 3 GH_z

3) POUŽIJEME AMDAHLŮV ZÁKON

ZRYCHLENÍ OPROTI 1 UZLU $S_1 = \frac{1}{0{,}1 + 0{,}9 \cdot \frac{1}{16}} = \frac{1}{\frac{16}{160} + \frac{9}{160}} = \frac{160}{25} = \frac{32}{5} = 6\frac{2}{5}$

$S_2 = \frac{1}{0{,}1 + 0{,}9 \cdot \frac{1}{32}} = \frac{1}{\frac{32}{320} + \frac{9}{320}} = \frac{320}{41} \overset{\text{zhruba}}{\approx} 8$

TAKŽE ZRYCHLENÍ 32 CUSTENŮ OPROTI 16 JE: $\frac{S_2}{S_1} = \frac{8}{6\frac{2}{5}}$

4) UDĚLÁME SI PRAVDIVOSTNÍ TABULKU KOMB. OBVODU. POTOM JI ZAPÍŠEME DO ROM TAK, ŽE ADRESA BUDOU VSTUPY Z ITY KOMB. OBVODU A POD TU ADRESOU BUDOU ULOŽENY VÝSTUPY KOMB. OBVODU JAKO CONTROL WORD.

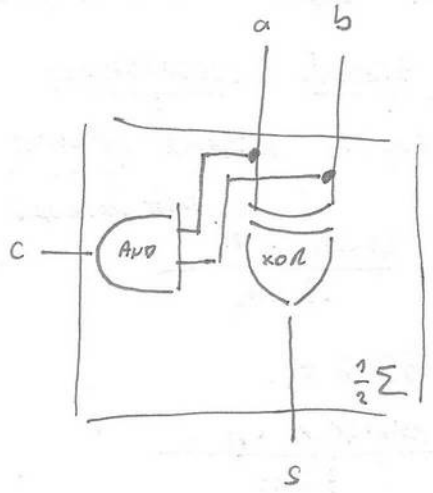
PŘÍSTUP JE MOŽNÝ, PROTOŽE DATOVÁ CESTA JE JEDNOCYKLOVÁ A TEDY PRO NÍ EXISTUJE JEDNOZNAČNÉ MAPOVÁNÍ OP CODE → CONTROL WORD. PROTO JE TO TAKÉ KOMBINAČNÍ A NE SEKVENČNÍ OBVOD.

TENTO PŘÍSTUP JE VHODNÝ TŘEBA KDYŽ VÍME, ŽE BUDEME CHTÍT ŘADIC POZDĚJI UPDATOVAT – POTOM POUŽIJEME (E)EPROM.

5)

a	b	c_i	c_{out}	S
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

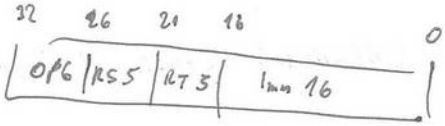
- ÚPLNÁ SČÍTAČKA MÁ VSTUP NA carry



6)

lw Rt, Offset (Rs)

Reg [Rt] = Mem [Reg [Rs] + SignExt (Offset)]



JUMP = 0
REG_DST = 0
REG_WRITE = 1
ALU_SRC = 0
ALU_OP = ADD
MEM_TO_REG = 1
BRANCH = 0
MEM_WRITE = 0

- DATOVÁ CESTA VYBERE V REGISTROVÉM POLI, DO KTERÉHO REGISTRU SE BUDE ZAPISOVAT. (RT, PROTO REG_DST=0)
- DO PC VRÁTÍ PC+4. PROTO BRANCH A JUMP=0.
- SEČTE IMMEDIATE ČÁST INSTRUKCE S RS (PROTO ALU_SRC=0 A ALU_OP=ADD.
- VYBERE V PAMĚTI ADRESU, KTERÁ JE VÝSLEDKEM ↑
- TU PŘEČTE, NE ZAPÍŠE (PROTO MEM_WRITE=0)
  (A PROTO MEM_TO_REG=1)
  (A PROTO REG_WRITE=1)

7)

- LATENCE BUDOU STEJNÉ, A TO $200 + 150 + 120 + 190 + 140 = 800$ ps (350, 470, 660)

- JAK V PŘÍPADĚ JEDNOCYKLOVÉ, TAK V PŘÍPADĚ PIPELINED DATOVÉ CESTY MUSÍ BÝT JEDEN PRŮCHOD DATOVOU CESTOU VŽDY STEJNĚ DLOUHÝ A MUSÍ BÝT DOSTATEČNĚ DLOUHÝ, ABY INSTRUKCE MOHLA PROJÍT VŠEMI ČÁSTMI DATOVÉ CESTY.

8)

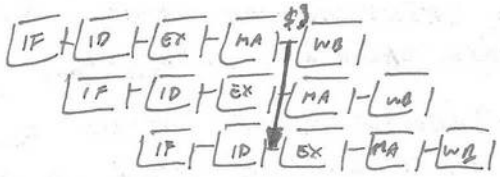
- ZAPISOVAT DO REGISTRU BUDOU ALU A lw OPERACE.
- TEDY PORT BUDE MÍT VYTÍŽENÍ 60%

~~(SLAB MŮŽE?~~

9) - NĚKTERÉ DATOVÉ HAZARDY NASTÁVAJÍ PROTOŽE INSTRUKCE A CHCE PRACOVAT S DATY, KTERÁ INSTRUKCE B VE DOPOČÍTALA, ALE JEŠTĚ JE NEUMÍSTILA ZPĚT DO HLAVNÍCH REGISTRŮ.

- FORWARDING (BYPASSING) JE TECHNIKA, KDE SI PROCESOR UVĚDOMÍ, ŽE K NĚČEMU TAKOVÉMU DOCHÁZÍ, VEZME POTŘEBNÁ DATA Z PIPELINY A DODÁ JE INSTRUKCI A.

```
add $1, $2, $3
or  $4, $5, $6
add $3, $4, $6
```



10) - PO TOM, CO TATO SEKVENCE PROBĚHNE JEDNOU SE USTÁLÍ, ŽE BUDE ZAČÍNAT VŽDY S PREDIKTOREM VE STAVU (1)

- JEDNA TATO SEKVENCE BUDE VYPADAT TAKTO:

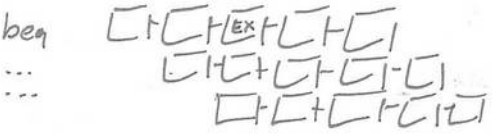
STAV (PŘED SKOKEM)	DOOPRAVDY	CO TUDÍ PREDIKTOR	ÚSPĚCH
1	T	NT	X
2	T	T	✓
3	T	T	✓
3	NT	T	X
2	NT	T	X

TEDY PRO KAŽDOU SEKVENCI BUDE MÍT PREDIKTOR ÚSPĚŠNOST 40%, TUDÍŽ I CELKOVĚ BUDE MÍT PŘESNOST 40%. (IGNORUJME TEN PRVNÍ PRŮCHOD)

11) - POLOVINU INSTRUKCÍ beq NAHRADÍME

- TEDY JE NOM 5% BUDOU beq A Z TOHO 95% BUDE SPRÁVNĚ PREDIKOVÁNO

- KDYBYCHOM NEMUSELI ŘEŠIT beq HAZARDY ... 1t
- KDYBYCHOM JE VŠAK ŘEŠILI STALLOVÁNÍM ... KAŽDÝ beq PŘIDÁ 2 INSTRUKCE, TEDY 10% → 30% TEDY 1,2t



ŠPATNÁ PREDIKCE → 2 nopy
(NEBO JINÁ VĚTVENÍ BEZ PREDIKCE)

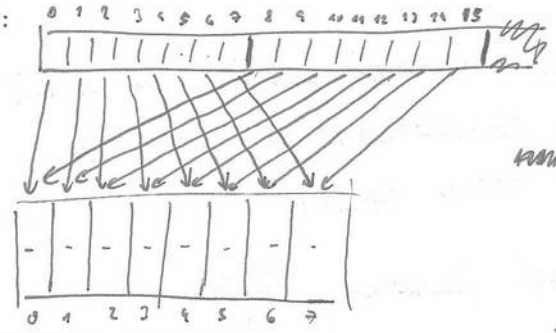
Pokračování na str. 5

- S predikcí a nahrazením bude jenom 5% * (100 – 95 %) = 1/400 instrukcí potřeba řešit pomocí stallů

12) NEMOHOU. DATA MUSEJÍ PROJÍT VYŠŠÍMI VRSTVAMI PAMĚŤOVÉ HIERARCHIE. V TOMTO PŘÍPADĚ TO ZNAMENÁ, ŽE MUSÍ BÝT NAČTENA DO OPERAČNÍ PAMĚTI.

13)
- SNIŽUJE $\%_{MISS}$, PROTOŽE ZMENŠUJE MNOŽSTVÍ CONFLICT MISSŮ.
  - A TÍM PÁDE ZVYŠUJE VÝKONNOST CACHE, JELIKOŽ LATENCE PŘÍSTUPU DO PAMĚTI PŘES CACHE = $t_{HIT} + \%_{MISS} \cdot t_{MISS}$
- ~~ZÁROVEŇ~~ - ZÁROVEŇ MÍRNĚ ZVYŠUJE t_{HIT}, PROTOŽE BUDEME MUSET ZJIŠŤOVAT, KTERÁ CACHE LINE V MNOŽINĚ MÁ SPRÁVNÝ TAG

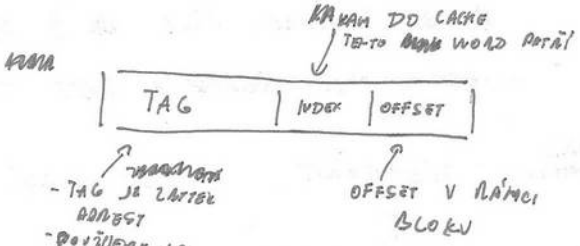
14)



V MOŽNÁ JSEM TO POCHOPIL ZCELA ŠPATNĚ. V TOM PŘÍPADĚ JE NA STR. 95 A OPRAVA.

(PŘEDPOKLÁDÁM, ŽE WORD = 1 B)
- BLOCK 4B → OFFSET ČÁST ADRESY BUDE DLOUHÁ 2 bity
- CACHE MÁ 8 POLOŽEK (CACHE LINES/BLOKŮ) → INDEX ČÁST BUDE 3 bity

TAG	INDEX	OFFSET
11	3	2



- TAG JE ZBYTEK ADRESY
- POUŽÍVÁME KE ROZLIŠENÍ MEZI BLOKY, KTERÉ MAJÍ STEJNÝ INDEX
- HORNÍ BITY ADRESY SLOVA

OFFSET V RÁMCI BLOKU

- CACHE BUDE MUSET JAKO OVERHEAD UKLÁDAT PRO KAŽDOU POLOŽKU A PRO KAŽDOU CESTU 11 bitů TAG A 1 VALID BIT

TEDY OVERHEAD BUDE $(11+1) \cdot 8 \cdot 2 = 192$ bitů

15)

1 P2 WRITE
- P2 POSLAL BusRdX
- P2 MÁ BLOK VE STAVU (M)
- OSTATNÍ ŘADIČE DALY BLOK DO STAVU (I)
  - POPŘ. P2 PŘERUŠIL A FLUSHNUL NĚKDO
  - (TEDY POKUD TEN BLOK MĚLY VE SVÉ CACHI)

2 P0 READ
- P2 PŘERUŠIL P0 A FLUSHNUL
- P0 A P2 TEĎ MAJÍ BLOK V (S)

3 P1 READ
- P0, P1, P2 MAJÍ BLOK V (S)

4 P0 WRITE
- P0 DÁ BLOK DO STAVU (M)
- OSTATNÍ ŘADIČE SI DAJÍ BLOK DO STAVU (I)

5 P2 READ
- P0 PŘERUŠÍ P2 A FLUSHNE
- TEĎ P0 A P1 MAJÍ BLOK VE STAVU (I)
- P2 MÁ BLOK VE STAVU (M)

- Předpokládám, že když je řadič přerušen, hned operaci zopakuje

11 pokr.)

Takže t_0 := čas bez stallování

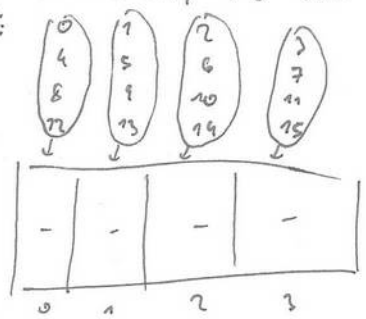
$t_1 = 1{,}2\,t_0$ čas bez predikce, bez nahrazení, se stallováním

$t_2 = 1\frac{2}{400}\,t_0$ čas s predikcí a nahrazením a jenom malým množstvím stallování

Tedy zrychlení $= 1 - \frac{t_1}{t_2} = 1 - \frac{1{,}2}{1\frac{2}{400}} \sim 20\%$ skoro jako kdyžtachom stally úplně eliminovali

14)

- Možná jsem pochopil zadání špatně a 8 položek je v cachi ~~celkem~~ celkem, ne pro každou cestu. Potom:



- potom také overhead (nelíbí) bude $(11+1) \cdot 4 \cdot 2 = 96$ bitů