Architektury Systemów Komputerowych

Ćwiczenia 4: "Mikroarchitektura"

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Harris & Harris: 7.1 – 7.5, 7.7 (drugie wydanie)
- Hennesy & Patterson: 4.1 – 4.10 (piąte wydanie)

Należy być przygotowanym do wytłumaczenia **wytłuszczonych** haseł.

Uwaga: Jeśli zadanie odwołuje się do diagramów mikroarchitektury procesora, to należy bazować na książce autorów Harris & Harris.

Zadanie 1

Rozważmy jednocyklową implementację procesora MIPS i następujące instrukcje:

```
add   Rd,Rs,Rt
beq   Rs,Rt,label
lw    Rt,offset(Rs)
```

a. Jakie są wartości poszczególnych sygnałów generowanych przez **jednostkę kontrolną**?
b. Z których bloków procesora (np. ALU, rejestry, itd.) korzystają w/w instrukcje?
c. Które bloki procesora obliczają wartości, z których dana instrukcja nie korzysta?

Zadanie 2

Rozważmy następujące instrukcje:

```
bezi  (Rs),Label       # Mem[Rs] = 0 → PC = PC + Offset
swi   Rd,Rs(Rt)        # Mem[Rs+Rt] := Rd
```

a. Co należy zmienić w potoku procesora MIPS, aby wspierać te instrukcje?
b. Jakie połączenia należy dodać do schematu procesora podanego w książce (7.58)?
c. Czy instrukcje wprowadzają jakieś nowe hazardy? Czy **przestoje** (ang. *stall*) związane z przetwarzaniem innych instrukcji wydłużyły się?

Zadanie 3

Rozważmy dwa ciągi instrukcji podane poniżej:

```
lw    $1,40($2)          add   $1,$2,$3
add   $2,$3,$3           sw    $2,0($1)
add   $1,$1,$2           lw    $1,4($2)
sw    $1,20($2)          add   $2,$2,$1
```

a. Znajdź wszystkie **zależności danych**.
b. Znajdź wszystkie **hazardy** w powyższych ciągach instrukcji, które wystepują w pięcioetapowym potoku MIPS – z **obejściami** i bez (ang. *bypassing*).
c. Narysuj diagram (jak w podrozdziale 7.8) wykonania ciągów instrukcji w potoku MIPS.

Zadanie 4

Rozważmy następujące dwie instrukcje procesora MIPS:

```
lw    $1,40($6)
add   $5,$5,$5
```

a. Co jest przechowywane w rejestrach między dwoma etapami potoku w trakcie wykonania tych instrukcji?
b. Które z rejestrów muszą być odczytane, a które faktycznie odczytuje procesor?
c. Co dokładnie robią te instrukcje w etapach *Execute* i *Memory* – podaj wartości sygnałów.

Zadanie 5

Poniższe ciągi instrukcji wykonujemy na procesorze MIPS z pięcioetapowym potokiem.

```
lw    $1,40($6)          add   $1,$5,$3
add   $2,$3,$1           sw    $1,0($2)
add   $1,$6,$4           lw    $1,4($2)
sw    $2,20($4)          add   $5,$5,$1
and   $1,$1,$4           sw    $1,0($2)
```

a. Procesor nie ma zaimplementowanych obejść ani wykrywania hazardów – wstaw instrukcje `nop`, aby zapewnić poprawność wykonania ciągów instrukcji.
b. Powtórz poprzedni punkt, ale nie używaj instrukcji `nop` jeśli się da. Możesz zastępować instrukcje innymi lub zmieniać kolejność instrukcji. Rejestr `$7` może być używany do przechowywania tymczasowych wartości.
c. Procesor implementuje obejścia, ale nie ma wykrywania hazardów. Co się stanie jeśli wykonamy powyższe ciągi instrukcji?

Zadanie 6

Wykonujemy poniższy kod na procesorze potokowym MIPS ze statycznym przewidywaniem skoków typu *zawsze skocz*.

```
            lw    $1,40($6)
label1:     beq   $2,$3,label2     # wykonany
            add   $1,$6,$4
label2:     beq   $1,$2,label1     # niewykonany
            sw    $2,20($4)
            and   $1,$1,$4
```

a. Narysuj diagram wykonania ciągu instrukcji zakładając, że skoki wykonują się w etapie *Execute* i procesor nie implementuje **delay slots**.
b. j.w ale procesor implementuje delay slots. Czemu w oryginalnym procesorze MIPS wybrano takie rozwiązanie?

Zadanie 7

Predyktor skoków jest realizowany przy pomocy małej i szybkiej pamięci RAM (np. 2^{12} słów) – tzw. *branch cache*. Załóżmy, że możemy równocześnie robić dostępy do pamięci instrukcji jak i pamięci podręcznej skoków. W którym etapie przetwarzania instrukcji w procesorze potokowym MIPS należałoby dodać logikę odpowiedzialną za przewidywanie skoków? Jakie dane należałoby składować w *branch cache*, żeby przy 100% skuteczności przewidywania procesor nie doświadczał przestojów związanych z wykonaniem skoków warunkowych?

Zadanie 8

Predyktor skoków ma istotny wpływ na **przepustowość** przetwarzania instrukcji. Załóżmy, że nasz procesor potokowy wykonuje nastepujący ciąg instrukcji – `ALU` : 50%, `beq` : 15%, `jmp` : 10%, `lw` : 15%, `sw` : 10% – a poszczególne predyktory skoków charakteryzują się następującą skutecznością prawidłowego przewidywania – *zawsze skocz* : 60%, *nigdy nie skacz* : 40%, *nasycający licznik 2-bitowy* : 90%.

a. Oblicz **CPI** (ang. *cycles per instruction*) dla powyższych danych. Załóż, że dopiero w fazie *Execute* wiadomo, czy skok zostanie wykonany. Dla uproszczenia przyjmujemy, że nie występują hazardy danych.
b. Niektóre skoki warunkowe można łatwiej klasyfikować niż pozostałe. Przyjmijmy, że 80% wszystkich skoków wykonanych w programie to łatwie do przewidzenia skoki wstecz używane w pętlach typu `do-while` i są zawsze poprawnie przewidywane. Jaka jest skuteczność predyktora z licznikiem 2-bitowym dla pozostałych 20% skoków.

Zadanie 9 [2pkt, Kiero]

W tym ćwiczeniu porównamy efektywność procosora potokowego (1-issue) i procesora superskalarnego (2-issue). Rozważmy nastepujący program w języku C:

```
for (i=0; i != j; i += 2)
      b[i] = a[i] - a[i+1];
```

a. Przetłumacz ten program do asemblera MIPS w jak najbardziej naturalny sposób. Załóż, że `i` jest przechowywane w rejestrze `$5`, `j` w `$6`, adres `a` w `$1`, a adres `b` w `$2`. Dodatkowo możesz użyć pomocniczych rejestrów `$10`, `$11`, `$12`.
b. Przedstaw diagram wykonania uzyskanego kodu na procesorze potokowym.
c. Przedstaw diagram wykonania dwóch pierwszych pętli uzyskanego kodu na procesorze superskalarnym. Załóż, że predykcja skoków jest bezbłędna. Dla uproszczenia zakładamy, że procesor potrafi pobrać dwie dowolne instrukcje (niekoniecznie kolejne) w tym samym cyklu. Nie ma zatem hazardów strukturalnych, ale mogą być hazardy danych, które muszą być rozwiązane poprawnie.
d. Przeorganizuj tłumaczenie danego programu do asemblera MIPS w taki sposób, aby uzyskać krótszy czas wykonania na procesorze superskalarnym. Przedstaw diagram wykonania dwóch pierwszych pętli uzyskanego kodu.
e. Ile zajmie wykonanie 1 000 000 pętli kodu z podpunktu (a) na procesorze potokowym, a ile na superskalarnym? Ile zajmie wykonanie kodu z podpunktu (d) na procesorze superskalarnym?

Krystian Bacławski