Systemy komputerowe

Lista zadań nr 6

Na ćwiczenia 13. i 14. marca 2022

Każde zadanie warte jest 1 punkt.

Zadanie 1. Przedstawiony poniżej fragment kodu wykonujemy na procesorze potokowym bez jednostki wykrywania hazardów. Początkowa wartość rejestru s0 to 11 a s1 to 22.

```
s0 = s1 + 5
s2 = s0 + s1
s3 = s0 + 15
s4 = s0 + s0
```

Jakie będą końcowe wartości rejestrów s2 , s3 i s4, a jakie powinny być, gdyby procesor wykonał program poprawnie? Uzupełnij kod o instrukcje nop by usunąć hazardy danych i pokaż, że program wykona się teraz poprawnie. Omawiając wykonanie kodu posługuj się diagramami wykonania.

Wskazówka: Pamiętaj, że plik rejestrów jest zapisywany w pierwszej połowie cyklu, a odczytywany w drugiej. Tzn. odczyt rejestru w fazie ID zwróci wartość zapisaną tam w fazie WB odbywającej się w tym samym cyklu.

Zadanie 2. Przedstawione poniżej fragmenty kodu wykonujemy na procesorze potokowym z jednostką wykrywania hazardów danych, *forwardingiem* i możliwością wstrzymywania (ang. *stall*) potoku. Przedstaw ich diagramy wykonania i pokaż, w jaki sposób procesor upora się z hazardami danych.

a)
```
s1 = s2 + 5
t0 = t1 - t2
t3 = *(s1 + 15)
*(t0 + 72) = t5
t2 = s4 & s5
```

b)
```
s0 = t0 + t1
s1 = t2 - t3
s2 = s0 & s1
s3 = t4 | t5
s4 = s2 + s3
```

c)
```
t0 = s0 + s1
t0 = t0 - s2
t1 = *(t0 + 60)
t2 = t1 & t0
```

d)
```
t0 = s0 + s1
t1 = *(s2 + 60)
t2 = t0 - t3
t3 = t1 & t0
```

Zadanie 3. Rozważmy następujący program:

```
LOOP: s0 = *s3
      s1 = *(s3 + 8)
      s2 = s0 + s1
      s3 = s3 - 16
      if s2 != 0 goto LOOP
```

Załóżmy, że procesor (z jednostką wykrywania hazardów danych, forwardingiem i możliwością wstrzymywania potoku) używa bezbłędnego predyktora skoków (nie będzie żadnych wstrzymań potoku spowodowanych hazardami sterowania), oraz że decyzja o skoku podejmowana jest w fazie EX.

a) Narysuj diagram wykonania dla pierwszych dwóch iteracji pętli.
b) Zaznacz fazy potoku, w których procesor nie wykonuje użytecznej pracy.

Wskazówka: Predyktor skoków działa w fazie IF instrukcji skoku. To, że decyzja o skoku podjęta zostaje w fazie EX oznacza, że odpowiedni adres instrukcji zapisany jest do licznika rozkazów w fazie MEM.

Zadanie 4. Powtórz zadanie 3. dla procesora ze statycznym predyktorem skoków w wariancie a) *always-taken* oraz b) *always-not-taken*. W punkcie b) możesz przyjąć, że po kodzie pętli następuje dowolny wybrany przez Ciebie ciąg instrukcji.

Zadanie 5. Wyjaśnij, na czym polega wczesne wykonywanie skoków (ang. *early branch execution*), w której fazie potoku następuje decyzja o skoku i aktualizacja licznika rozkazów, oraz jaka motywacja stoi za tym rozwiązaniem. Następnie wykonaj poniższe punkty.

a) Powtórz zadanie 3. dla procesora ze statycznym predyktorem skoków *always-not-taken* oraz wczesnym wykonywaniem skoków.
b) Podaj przykład kodu, wskazujący, że wczesne wykonanie skoku może wygenerować hazard danych.

Zadanie 6. Załóżmy, że rozkład instrukcji w programie wykonywanym na procesorze potokowym jest następujący:

x = y binop z	if x relop y goto L	x = *(y+ imm)	*(x +imm) = y

45%	25%	25%	5%

Dla tego programu prawdopodobieństwa poprawnej predykcji skoków przez różne predyktory widnieją w tabelce:

statyczny always-taken	statyczny always-not-taken	2-bitowy
45%	55%	85%

Przy tych (realistycznych) założeniach porównamy wydajność predyktorów skoków.

a) Jaki jest przyrost wartości CPI wywołany błędnie przewidzianymi skokami dla procesora z predyktorem always-taken? Procesor używa wczesnego wykonywania skoków, a w programie nie ma hazardów danych.
b) Powtórz punkt a) dla procesora z predyktorem always-not taken.
c) Powtórz punkt a) dla procesora z predyktorem 2-bitowym.
d) Załóżmy, że nasz program potrafimy przepisać w taki sposób, że połowę skoków zastąpiliśmy instrukcjami arytmetycznymi. Jakie otrzymamy przyśpieszenie? Załóż, że poprawnie i niepoprawnie przewidziane skoki mają taką samą szansę bycia zastąpionymi.

Zadanie 7. Rozważmy następujący program:

```
*(s3 + 12) = s5
s5 = *(s3 + 8)
s4 = s2 - s1
if s4 == 0 goto Label
s2 = s0 + s1
s2 = s6 -  s1
```

gdzie Label jest pewną etykietą w programie. Zmodyfikowaliśmy procesor potokowy w ten sposób, że występuje w nim tylko jeden układ pamięci roboczej, przeznaczony dla dostępu do danych i instrukcji jednocześnie. Spowoduje to powstanie nowego typu hazardu — hazardu strukturalnego, w każdym cyklu, w którym procesor będzie jednocześnie pobierał nową instrukcję z pamięci i wykonywał dostęp do pamięci na rzecz instrukcji już wykonującej się. W takiej sytuacji należy wstrzymać potok.

a) Narysuj diagram wykonania dla powyższego kodu.

b) Czy można zredukować liczbę cykli w których potok jest wstrzymany zmieniając kolejność instrukcji w programie (z zachowaniem jego semantyki)?
c) Czy kompilator może radzić sobie z tego typu hazardami przez wstawienie do programu instrukcji nop? Czy też problem musi być rozwiązany przez sam procesor?

Zadanie 8. Przy założeniach jak w poprzednim zadaniu, oszacuj liczbę cykli procesora utraconych wskutek wstrzymania potoku dla typowego programu. Przyjmij następujący rozkład instrukcji:

x = *(y+ imm)	* (x +imm) = y	if x relop y goto L	x = y binop z
25%	11%	12%	52%

Zadanie 9. Załóżmy, że wszystkie instrukcje dostępu do pamięci są postaci x = *y lub *x = y. Dzięki temu żadna instrukcja nie używa jednocześnie faz EX i MEM potoku. Fazy te można zatem połączyć uzyskując czterofazowy potok. W jaki sposób ta modyfikacja wpłynie długość cyklu procesora? Czy wpływ na wydajność będzie jednoznacznie pozytywny?