Architektury systemów komputerowych

Lista 9

$x_9 = 6$ (minimum na bdb)

W niektórych zadaniach na dzisiejszej liście będziemy odwoływać się do poniższych fragmentów kodu MIPS:

```
Program 1:

sw r16, 12(r6)
lw r16, 8(r6)
beq r5, r4, Label #załóż, że r5 != r4
add r5, r1, r4
slt r5, r15, r4


Program 2:

        lw r2, 0(r1)
label1: beq r2, r0, label2 #za pierwszym razem r2 != r0, potem równe
        lw r3, 0(r2)
        beq r3, r0, label1 #załóż, że r3 = r0
        add r1, r3, r1
label2: sw r1, 0(r2)
```

1. Rozważamy procesor potokowy z bezbłędną predykcją skoków (nie ma żadnych hazardów sterowania). Załóżmy, że mamy tylko jedną jednostkę pamięci (nie ma podziału na pamięć rozkazów oraz danych). Wtedy, za każdym razem gdy chcemy pobrać rozkaz, a inny rozkaz z potoku pobiera dane dochodzi do hazardu strukturalnego. Przyjmijmy, że taki hazard jest zawsze rozwiązywany na korzyść rozkazu pobierającego dane. Ile cykli zajmie wykonanie `Programu 1` (które rozkazy trzeba będzie opóźnić?)? Czy tego typu hazardy strukturalne dałoby się rozwiązać poprzez dopisanie (w czasie kompilacji) rozkazów `nop` w odpowiednich miejscach kodu?

2. Ponownie rozważamy procesor potokowy z bezbłędną predykcją skoków. Rozważmy modyfikację rozkazów load/store taką, że biorą one adres bezpośrednio z rejestru (bez dodawania offsetu). Przy tej modyfikacji rozkazy te nie muszą używać ALU. A wtedy fazy MEM i EX mogą być nałożone na siebie i potok ma tylko cztery fazy. Zmodyfikuj `Program 1` tak, aby działał dla przebudowanej w opisany sposób architektury. Zakładając, że modyfikacja nie wpływa na długość cyklu zegarowego porównaj czas działania zmodyfikowanego pragramu z oryginalnym.

3. Przedstaw diagram wykonania `Programu 2` na procesorze potokowym (tzn. pokaż, w jakich fazach wykonania są rozkazy w kolejnych cyklach zegarowych) zakładając, żę skoki wykonywane są w fazie EX. Następnie powtórz to ćwiczenie zakładając, że tym razem implementacja używa tzw. skoków opóźnionych (deleyed branches). Instrukcja znajdująca się za `beq` wypełnia tzw. *delayed slot* i wykonuje się zawsze (ewentualny skok następuje dopiero po niej).

4. Rozważmy teraz architekturę, w której rozkazami skoków warunkowych są `bez rd, Label` oraz `bnez rd, Label` (odpowiednio: skocz jeśli w rejestrze `rd` jest zero, skocz jeśli w rejestrze `rd` nie ma zera). Przepisz kod `Programu 2` do tak zmodyfikowanej architektury. Możesz użyć rejestru `r8` jako rejestru pomocniczego oraz rozkazu `seq rd, rs, rt` (*set if equal*; rozkaz typu R).

5. Przedstaw diagram wykonania `Programu 2` na procesorze potokowym zakładając, że skoki wykonywane są w fazie ID. Przyjmując, że długość cyklu jest taka sama jak w przypadku procesora wykonującego skoki w fazie EX porównaj czasy wykonania w `Programu 2` w obydwu wariantach.

6. Zbadamy teraz skuteczność różnych wariantów predykcji skoków na pewnym rozkazie skoku (można np. myśleć, że jest to skok zamykający pętlę), który podczas swoich pierwszych pięciu wykonań kolejno: skoczył, nie skoczył, skoczył, skoczył, nie skoczył.

   (a) Jaka jest skuteczność predyktora, który zakłada, że skok zawsze się wykonuje?

   (b) Jaka jest skuteczność predyktora, który zakłada, że skok nigdy się nie wykonuje?

   (c) Jaka jest skuteczność predyktora dwubitowego (patrz *Wykład 9, Slajd 43*) podczas pierwszych skoków, przy założeniu, że startuje on w swoim lewym dolnym stanie?

   (d) Jaka jest skuteczność predyktora dwubitowego, jeśli przez rozważany skok przechodzimy nieskończenie wiele razy (opisana sekwencja wykonań skoku powtarza się cyklicznie).

7. W tym i w następnym zadaniu rozważamy 4-gigabajtową przestrzeń adresową, adres określa pojedynczy bajt pamięci, a więc ma 32 bity. Przyjrzymy się dwóm wariantom bezpośredniego mapowania przestrzeni adresowej w pamięci cache.

   Załóżmy, że mamy do dyspozycji pamięć cache, która może przechowywać 16 kB danych (nie liczymy miejsca przeznaczonego na znaczniki). Każdy bajt pamięci cache jest adresowany osobno, a więc każdy musi posiadać własny znacznik. Ile bitów ma adres bajtu w pamięci cache, a ile bitów musi mieć znacznik? Pod jakim adresem w pamięci cache może być przechowywana zawartość pamięci RAM o adresie 0x1A2B3C4D (podanym szesnastkowo)? Ile bitów (łącznie z miejscem na znaczniki) ma nasza pamięć cache?

8. Załóżmy teraz, że adresowalną jednostką pamięci cache jest wiersz (blok) złożony z czterech 4-bajtowych słów. Ile wierszy ma nasza 16 kB pamięć cache? Gdzie w pamięci cache będzie przechowana zawartość adresu 0x1A2B3C4D (w którym wierszu, w którym słowie tego wiersza, w którym bajcie tego słowa)? Ile bitów (łącznie z miejscem na znaczniki) ma nasza pamięć cache?