Architektury systemów komputerowych

Lista zadań nr 14

Na zajęcia 11 i 12 czerwca 2018

UWAGA! W trakcie prezentacji zadań należy być przygotowanym do wytłumaczenia haseł, które zostały oznaczone **wytłuszczoną** czcionką.

W zadaniach 1–4 rozważamy procesor potokowy MIPS, którego schemat widnieje na slajdzie 85 do wykładu. Wiemy, że decyzja o wykonaniu skoku warunkowego obliczana jest pod koniec fazy ID, a załadowanie instrukcji z nowej ścieżki programu wykonuje się na początku fazy EX. Przyjmujemy, że strategia statycznego przewidywania skoków to „zawsze nie wykonuj skoku". Procesor nie implementuje *branch delay slots*.

Zadanie 1. Dla poniższego programu zakreśl wszystkie rejestry między którymi występują zależności danych RAW, oraz narysuj strzałki prowadzące od miejsca użycia wartości do miejsca ich wyprodukowania. Narysuj diagram stanu potoku (jak na slajdzie 89) i oznacz ścieżki przekazywania danych przy pomocy obejść.

```
1 L1:   addi    $3, $5, 4
2       lw      $4, 0($6)
3       beq     $2, $3, L2    # skok wykonał się
4       addi    $2, $2, 4
5 L2:   bne     $2, $4, L1    # skok nie wykonał się
6       sw      $3, 0($2)
7       addi    $3, $2, $3
```

Zadanie 2. Powtórz polecenia z poprzedniego zadania dla poniższego kodu.

```
1       lw      $3, 0($5)
2       srl     $8, $8, 1
3       beq     $2, $3, L1     # skok nie wykonał się
4       lw      $3, 0($6)
5       addi    $3, $3, -16
6 L1:   bne     $2, $3, L2     # skok wykonał się
7       subu    $2, $2, $8
8 L2:   sw      $2, -4($7)
```

Zadanie 3. Ile taktów zegarowych zajmie przetworzenie instrukcji wykonywanych w trakcie pojedynczej iteracji poniższej pętli? Czas przetwarzania instrukcji definiujemy jako różnicę (w taktach zegarowych) między wejściem instrukcji do etapu IF, a opuszczeniem etapu WB.

```
1 Loop:
2       lw      $t0, 0($a0)    # A[i]
3       sll     $t0, $t0, $v0  # (A[i] << a)
4       lw      $t1, 0($a1)    # B[i]
5       slr     $t1, $t1, $v1  # (B[i] >> b)
6       xor     $t2, $t0, $t1  # (A[i] >> a) ^ (B[i] << b)
7       sw      $t2, 0($a2)    # C[i] = (A[i] >> a) + (B[i] << b)
8       addi    $a0, $a0, 4
9       addi    $a1, $a1, 4
10      addi    $a2, $a2, 4
11      bne     $a2, $a3, Loop # $a3 = koniec tablicy C
```

Wykonaj ręcznie optymalizację kodu powyższej pętli. Dopuszczamy dowolną transformację kodu zachowującą semantykę, np. zamiana kolejnością lub zastępowanie instrukcji. Można korzystać z dodatkowych rejestrów `$t3`...`$t7`. Ile cykli zegarowych zajmuje teraz wykonanie jednej iteracji pętli?

UWAGA! W zadaniach 4–6 rozważamy procesor superskalarny ze statycznym zlecaniem instrukcji, który opisano w materiałach dodatkowych do wykładu 14 dostępnych na stronie przedmiotu.

Zadanie 4. Rozważamy kod pętli z zadania 3. Ile czasu trwa zlecanie instrukcji wykonywanych w pojedynczej iteracji tejże pętli? Narysuj tabelkę przypisującą instrukcje do potoków ALU/BRANCH i LOAD/STORE. Wyznacz zależności między instrukcjami, które ograniczają tempo zlecania rozkazów procesora. Ile średnio instrukcji udało się przetworzyć w jednym takcie zegarowym?

Zadanie 5. Dla pętli z zadania 3 wykonaj optymalizację **szeregowania instrukcji** (ang. *instruction scheduling*). Przedstaw na tablicy kod po optymalizacji, po czym powtórz polecenia z zadania 4.

Zadanie 6. Rozwiń dwukrotnie pętlę z zadania 3 i wykonaj szeregowanie instrukcji. Przedstaw na tablicy kod po optymalizacji, po czym powtórz polecenia z zadania 4. Czy dalsze rozwijanie pętli przyniesie zwiększenie wydajności kodu?

Zadanie 7. Rozważamy potokowy procesor MIPS dysponujący dwubitowym dynamicznym predyktorem skoków. Do czego służą **BTB** (ang. *branch target buffer*) i **BHT** (ang. *branch history table*)? W których fazach przetwarzania instrukcji skoków warunkowych i bezwarunkowych procesor odczytuje zawartość BTB i BHT, a kiedy je aktualizuje?

Zadanie 8. W kolejnych iteracjach pętli obliczamy pewne wyrażenie, które używa instrukcji skoku warunkowego. Załóżmy, że skok ten zachowuje się zgodnie z powtarzającym się wzorcem: T, NT, T, T, NT (gdzie T i NT oznaczają odpowiednio „taken” i „not-taken”).

Dla wcześniej nieznanej instrukcji skoku warunkowego stan początkowy dwu-bitowego predyktora ustala się na „strongly not-taken”. Jaka jest skuteczność[1] **statycznego predyktora skoków** „zawsze nie wykonuj skoku” i dwu-bitowego **dynamicznego predyktora skoków** zakładając, że pętla wykonuje się w nieskończoność? Zaprezentuj jak będzie zmieniał się stan predyktora skoków w pierwszych 10 iteracji pętli.

[1] Liczba dobrze przewidzianych skoków w stosunku do wszystkich skoków wyrażona w procentach.