Architektury systemów komputerowych

Lista 11

$x_{11} = 6$ (minimum na bdb)

1. W tym zadaniu rozważamy procesor, w którym poszczególne fazy wykonywania rozkazów trwają odpowiednio: IF 300ps, ID 400ps, EX 250ps, MEM 500ps, WB 100ps. Jaka będzie długość cyklu zegarowego w przypadku procesora jednocyklowego, a jaka w przypadku procesora z przetwarzaniem potokowym? Załóżmy, że możemy rodzielić jedną z faz na dwie mniejsze (każdą o czasie działania równym połowie wyjściowej fazy). Którą fazę najlepiej wybrać. Jak takie rozdzielenie wpłynie na długość cyklu? To samo zadanie rozwiąż przy założeniu, że fazy trwają odpowiednio: IF 200ps, ID 150ps, EX 120ps, MEM 190ps, WB 140ps.

2. Wzorując się na rysunku ze slajdu 8, wykład 11, pokaż ścieżki forwardowania danych dla następującego kodu:

```
add $3, $4, $6
sub $5, $3, $2
lw  $7, 100($5)
add $8, $7, $2
```

3. Znajdź wszystkie zależności danych w poniższym kodzie. Które będą hazardami (nieusuwalnymi), a które zostaną rozwiązane przez forwarding?

```
add $3, $4, $2
sub $5, $3, $1
lw  $5, 100($3)
add $7, $3, $6
```

4. Rozważmy następujący ciąg rozkazów:

```
lw $1, 40($6)
add $6, $2, $2
sw $6, 50, ($1)
```

   Znajdź wszystkie zależności danych. Jaki będzie czas wykonania tego ciągu rozkazów na procesorze z przetwarzaniem potokowym, jeśli założymy, że:

   (a) nie ma forwardingu, a cykl zegarowy ma 200 ps

   (b) jest forwarding ALU-ALU, ale nie ma forwardingu MEM-ALU, a cykl zegarowy ma 220ps

   (c) jest pałny forwarding, a cykl zegarowy ma 250 ps

5. Rozważmy ciąg instrukcji `lw, add, lw, add, ...` długości 1000, w którym każda instrukcja zależy dokładnie od poprzedniej instrukcji (`add` ma dodać wartość z rejestru ładowanego przez poprzedni rozkaz `lw`, a `lw` ma bazę adresu w rejestrze, który jest wyliczny przez poprzedni rozkaz `add`. Ile cykli zabierze wykonanie tego ciągu procesorem ze sladju 23, wykład 11? Ile zajęłoby, gdybyśmy nie używali forwardingu?

6. Rozważmy dwie nowe instrukcje: `bezi (Rs), Label` oraz `swi Rd, Rs(Rt)`. Pierwsza z nich ma działać następująco: jeśli MEM[Rs]=0, to PC:=PC+Offs. Druga: Mem[Rs+Rt]:=Rd.

   (a) Opisz zmiany w procesorze potokowym, które pozwolą rozszerzyć listę rozkazów o `bezi` i `swi` (jakie nowe układy i sygnały sterujące są potrzebne? jak zmodyfikować stare?)

   (b) Czy nowe rozkazy mogą powodować hazardy?

(c) Podaj przykład kodu, w którym użycie nowych rozkazów jest użyteczne.

(d) Przetłumacz nowe rozkazy na ciągi starych.

7. Rozważmy następujące dwa fragmenty kodu:

```
a)                              b)
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)
```

Dla każdego z nich:

(a) Załóżmy, że procesor nie ma układów forwardowania danych ani wykrywania hazardów. Dopisz dodatkowe instrukcje `nop` (nic nie rób, *no operation*), które zapewnią poprawne wykananie.

(b) Spróbuj usunąć hazardy za pomocą przestawiania/modyfikacji rozkazów, zmniejszając tym samym liczbę koniecznych rozkazów `nop`. Ewentualne modyfikacje rozkazów mogą polegać na zmianie numerów rejestrów. Możesz założyć, że rejestr $7 nie jest używany przez program.

(c) Załóżmy, że procesor ma zaimplemetowane układy forwardowania, ale nie ma układu wykrywania hazardów (*hazard detection*). Co stanie się podczas wykanywania podanego kodu?

(d) Rozważmy teraz model procesora ze slajdu 24, wykład 11. Jakie sygnały sterujące wyprodukuje *hazard detection unit* podczas pierwszych pięciu cykli wykonywania kodu?

8. Rozważmy procesor z przetwarzaniem potokowym, ale bez forwardingu. Opisz konieczne modyfikacje układu wykrywania hazardów. Przeanalizuj działanie nowego układu na przykładach kodu z poprzedniego zadania.