Architektury systemów komputerowych

Lista 10

$x_{10} = 6$ (minimum na bdb)

Rysunki na tej liście zaczerpnięte są z książki Hennesy'ego i Pattersona *Computer Organization and Design*.

1. Które z rozkazów rozważanych na wykładzie 10 będą działały poprawnie na procesorze jednocyklowym z Rysunku 1 (także notatki z wykładu 10, slajd 14), jeśli pojawi się błąd ustawiający na stałe wartość jednego z sygnałów sterujących. Rozważamy następujące sytuacje:

   (a) `RegWrite=1` (`RegWrite` to sygnał zezwolenia na zapis rejestru)

   (b) `Branch=1` (`Branch` to informacja o tym, czy rozkaz jest skokiem warunkowym)

   (c) `MemRead=1` (`MemRead` to informacja o tym, że pamięć będzie odczytywana)

   (d) `MemWrite=1` (`MemWrite` to sygnał zezwolenia na zapis do pamięci)

   (e) `RegWrite=0`

   (f) `Branch=0`

   (g) `MemRead=0`

   (h) `MemWrite=0`

2. Rozważmy dodanie rozkazu `jr` (skocz pod adres zapisany w rejestrze o numerze podanym w argumencie rozkazu) do implemntacji jednocyklowej (Rysunek 1). Opisz konieczne modyfikacje i rozszerzenia układu, dorysuj na rysunku potrzebne dodaktowe połączenia i układy.

3. Odpowiedz na pytanie z zadania 1, ale tym razem dla implementacji wielocyklowej z Rysunku 2 (wykład 10, slajd 24) i następujących wartości sygnałów:

   (a) `RegWrite=1`

   (b) `MemRead=1`

   (c) `MemWrite=1`

   (d) `IRWrite=1` (zezwolenie na zapis do rejestru rozkazów)

   (e) `PCWrite=1` (zezwolenie na zapis do licznika rozkazów)

   (f) `PCWriteCond=1` (j.w., uaktywniane dla skoków warunkowych)

   (g) `RegWrite=0`

   (h) `MemRead=0`

   (i) `MemWrite=0`

   (j) `IrWrite=0`

   (k) `PCWrite=0`

   (l) `PCWriteCond=0`

4. Rozszerz procesor wielocyklowy z Rysunku 2 o rozkaz `lui` (załaduj stałą 16-bitową do dwóch górnych bajtów wskazanego rejestru). Jak w zadaniu 2, opisz konieczne modyfikacje i rozszerzenia oraz zaznacz je na rysunku.

5. Rozważmy modyfikację procesora wielocyklowego z Rysunku 2, w której układ rejestrów (na rysunku oznaczony jako `Registers`) ma tylko jedno wyjście (`Read Data 1`). Opisz konieczne zmiany, po których procesor nadal będzie poprawnie realizował naszą listę rozkazów. Jak zmieni się automat opisujący działanie jednostki sterującej (slajd 31, wykład 10)?

6. Załóżmy, że MIPS ma instrukcję `bcmp`, która porównuje dwa bloki pamięci. Zakładamy, że adres pierwszego bloku podany jest w rejestrze `$t1`, drugiego w `$t2`, a długość bloków (w słowach, liczba naturalna) w `$t3`. Rozkaz zapiuje wynik w `$t1` – wynikiem jest adres pierwszej znalezionej różnicy lub zero, jeśli bloki są identyczne. Zakładamy również, że podczas wykonania rozkazu mogą być modyfikowane rejestry `$t1-$t5`.

   Napisz program w asemblerze MIPS emulujący opisany powyżej rozkaz. Ile cykli potrwa jego wykonanie dla bloków długości 100 na procesorze wielocyklowym?

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

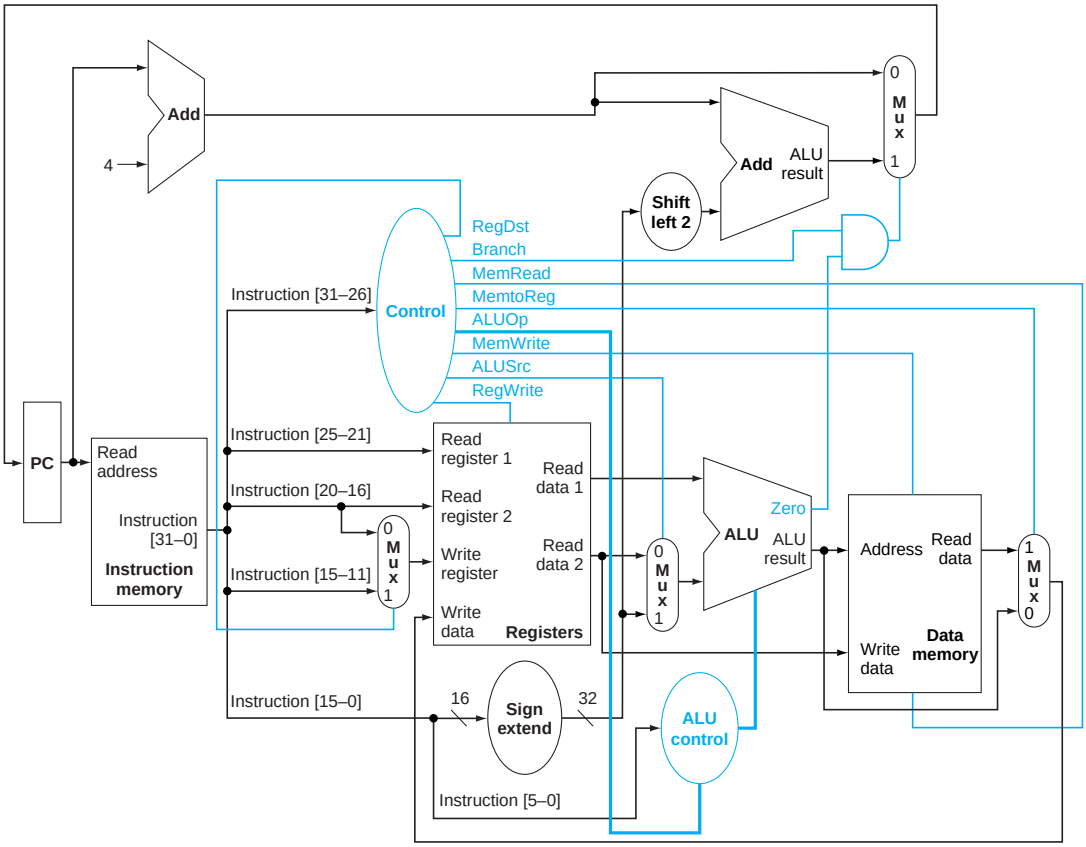
```
add $3, $4, $6
sub $5, $3, $2
lw  $7, 100($5)
add $8, $7, $2
```

8. 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
```

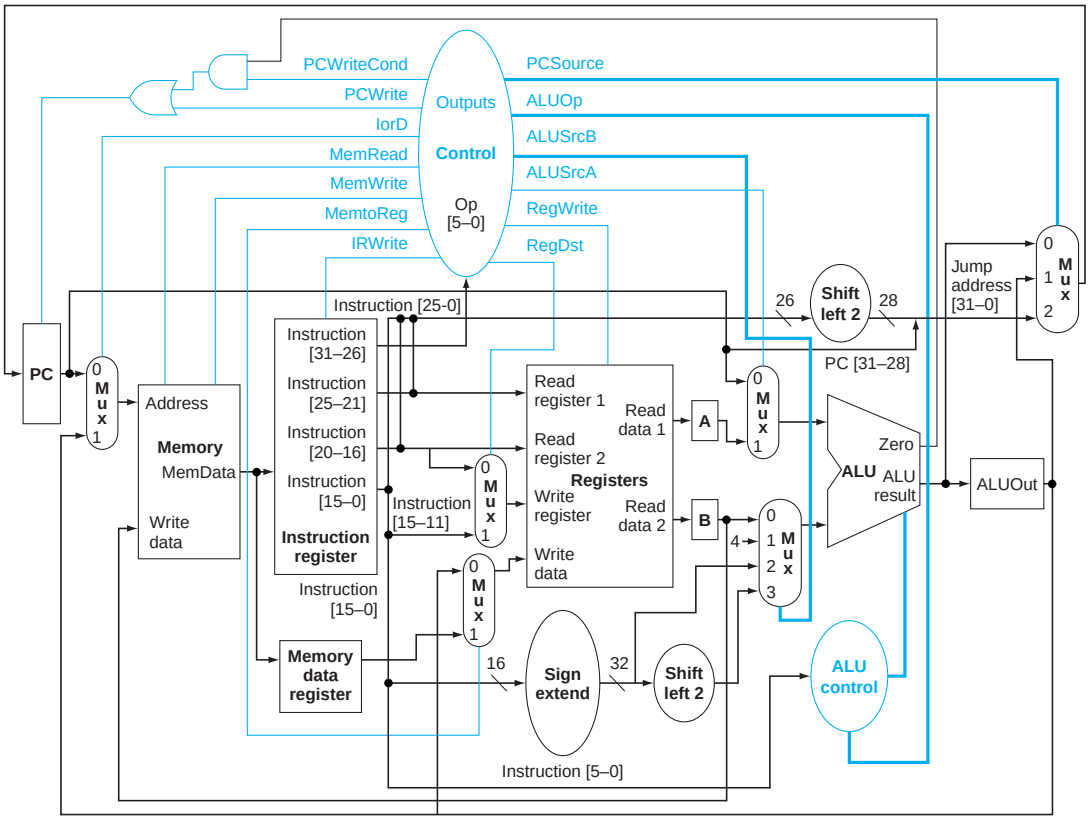
9. Rozważmy ciąg instrukcji `lw, add, lw, add, ...` długości 1000, w którym każda instrukcja zależy dokładnie od poprzedniej instrukcji (`add` ma dodać wartość z rejestru ładowanego przez poprzedni rozkaz `lw`, a `lw` ma bazę adresu w rejestrze, który jest wyliczny przez poprzedni rozkaz `add`. Ile cykli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

Rysunek 1: Implementacja jednocyklowa.



PAT05F17.eps

Rysunek 2: Implementacja wielocyklowa



PAT05F28.eps