ARCHITETTURE AVANZATE DEI CALCOLATORI, A.A. 2007/08
Esercizi sul pipelining

Esercizio 1)
Si considerino le seguenti sequenze di istruzioni che presentano delle dipendenze sui dati.

```
a) lw $3, 0($4)
   lw $5, 0($3)

b) lw $2, 0($3)
   beq $4, $2, L1

c) add $3, $4, $4
   or $3, $3, $4

d) lw $4, 0($3)
   or $2, $5, $4

e) lw $3, 0($4)
   sw $3, 0($5)

f) sub $4, $3, $2
   or $5, $6, $4

g) add $2, $2, $3
   sw $2, 0($4)

h) and $5, $5, $4
   beq $5, $6, L1
```

Si indichi se le dipendenze determinano criticità sui dati e di quale tipo (RAW di tipo define/use, RAW di tipo load/use). Per ciascuna criticità individuata, si descriva come viene risolta dal processore MIPS con pipeline ottimizzata (lettura e scrittura del banco dei registri nello stesso ciclo di clock, propagazione dei dati ed anticipazione del salto nello stadio ID). Se necessario, è possibile introdurre ulteriori percorsi di propagazione non considerati a lezione. Si motivino le risposte tramite l'uso di diagrammi a cicli multipli della pipeline.

Esercizio 2)
Si considerino le seguenti sequenze di istruzioni che presentano delle dipendenze sui dati.

```
lw $3, 0($4)
addi $5, $5, 1
sw $3, 0($5)

lw $2, 0($3)
sw $4, 0($2)
```

Si determini se le dipendenze determinano criticità sui dati e di quale tipo (RAW di tipo define/use, RAW di tipo load/use). Per ciascuna criticità individuata, si descriva se e come viene risolta dal processore MIPS considerato a lezione con pipeline ottimizzata (lettura e scrittura del banco dei registri nello stesso ciclo di clock, forwarding dei dati ed anticipazione del salto nello stadio ID). Se la criticità viene risolta, si mostri come sono utilizzate dal processore MIPS le condizioni per l'individuazione e gestione delle criticità esaminate a lezione. Se la criticità non viene risolta, si scrivano le opportune condizioni per la sua individuazione e gestione e si descriva l'eventuale hardware da aggiungere (mux per il forwarding) ed il percorso di forwarding usato.

Esercizio 3)
Si consideri la sequenza di istruzioni in assembler MIPS

```
sw $2, 20($4)
lw $5, 20($4)
```

E' necessario utilizzare la tecnica della propagazione per massimizzare le prestazioni?
In caso affermativo, si descriva come modificare l'unità di elaborazione e si scrivano le condizioni che permettono di identificare la criticità; altrimenti, si motivi la risposta.

Esercizio 4)
Si consideri un processore con predizione dinamica dei salti. La Branch History Table (BHT) del processore ha 8 righe e ciascuna riga contiene un bit (1-bit BHT). Si consideri il frammento codice in assembler MIPS (la colonna a sinistra è l'indirizzo dell'istruzione):

```
36          addi $3, $0, 5
40    L2:   addi $1, $0, 5
44    L1:   add $2, $2, $2
48          addi $1, $1, -1
52          bne $1, $0, L1
56          addi $3, $3, -1
60          bne $3, $0, L2
64          add $9, $7, $8
```

Si assuma che il PC sia inizializzato a 36 e che il contenuto iniziale della 1-bit BHT sia il seguente:

Entry	Bit di predizione
0	1
1	0
2	1
3	0
4	1
5	1
6	0
7	1

Per l'accesso alla BHT, si usino i 3 bit meno significativi dell'indirizzo dell'istruzione.

a) Quante istruzioni di salto vengono eseguite?
b) Quanti salti sono predetti correttamente?
c) Quale è l'accuratezza della predizione?

Esercizio 5)
Si consideri il seguente frammento di codice in linguaggio C:

```
sum1 = 0;
sum2 = 0;
for (i=1; i<3; i++) {
      sum1 = sum1 + a[i] + b[i];
      sum2 = sum2 + a[i]; }
```

Il sorgente sia stato compilato nel seguente codice in assembler MIPS. Si assuma che i registri `$s0` e `$s1` siano stati inizializzati rispettivamente con l'indirizzo di `a[1]` e `b[1]` e che `$t2` contenga l'indirizzo successivo all'ultimo elemento di `a`.

```
Loop: lw $t0, 0($s0)
      lw $t1, 0($s1)
      add $s2, $s2, $t0
      add $s2, $s2, $t1
      add $s3, $s3, $t0
      addi $s0, $s0, 4
      addi $s1, $s1, 4
      bne $s0, $t2, Loop
```

a) Effettuare il loop unrolling ed il riordinamento del codice con l'obiettivo di massimizzare il grado di parallelismo sulla pipeline ottimizzata del processore MIPS. Calcolare il valore del CPI ottenuto prima e dopo il loop unrolling ed il riordinamento.

b) Schedulare il ciclo srotolato sul processore MIPS con pipeline statica two-issue. Calcolare il valore del CPI ottenuto.

Esercizio 6)
Sia dato il seguente ciclo in linguaggio C

```
for (i=0; i<n; i++)
      C[i] = A[i] + B[i] + INC1 + INC2
```

Il sorgente sia stato compilato nel seguente codice in assembler MIPS. Si supponga che i registri `$4` e `$7` siano stati inizializzati rispettivamente ai valori `0` e `4n`. I simboli `BASEA`, `BASEB`, `BASEC`, `INC1` e `INC2` sono costanti a 16 bit, prefissate.

```
L1:   lw $2, BASEA($4)
      addi $2, $2, INC1
      lw $3, BASEB($4)
      addi $3, $3, INC2
      add $5, $2, $3
      sw $5, BASEC($4)
      addi $4, $4, 4
      bne $4, $7, L1
```

Si supponga che il ciclo venga iterato N volte (con N grande a piacere) tramite una pipeline MIPS a 5 stadi. Si consideri una generica iterazione del ciclo.

a) Si supponga che la pipeline sia priva di ottimizzazioni.
   - Disegnare il diagramma a ciclo multiplo della pipeline ed individuare le criticità sui dati di tipo RAW e le criticità sul controllo.
   - Inserire gli stalli necessari a risolvere le criticità individuate.
   - Calcolare il numero totale di stalli inseriti ed il CPI asintotico ottenuto.

b) Si supponga che nella pipeline siano state introdotte le seguenti ottimizzazioni:
   - nel banco dei registri è possibile la lettura e la scrittura nello stesso ciclo di clock;
   - forwarding dei dati;
   - anticipazione del salto nello stadio ID.
   - Disegnare il diagramma a ciclo multiplo della pipeline ed individuare le criticità sui dati di tipo RAW e le criticità sul controllo rimaste; indicare i percorsi di forwarding utilizzati.
   - Inserire gli stalli necessari a risolvere le criticità rimaste.
   - Calcolare il numero totale di stalli inseriti ed il CPI asintotico ottenuto.

Esercizio 7)

Si consideri il codice in assembler MIPS dell'Esercizio 6.
Si supponga che il ciclo venga iterato N volte (con N grande a piacere) tramite una pipeline MIPS a 5 stadi. Si consideri una generica iterazione del ciclo.

a) Si supponga che nella pipeline siano state introdotte le ottimizzazioni indicate al punto b) dell'Esercizio 5 più la seguente:
   - predizione statica dei salti all'indietro di tipo branch always taken
   - Disegnare il diagramma a ciclo multiplo della pipeline ed individuare le criticità sui dati di tipo RAW e le criticità sul controllo rimaste; indicare i percorsi di forwarding utilizzati.
   - Inserire gli stalli necessari a risolvere le criticità rimaste.
   - Calcolare il numero totale di stalli inseriti ed il CPI asintotico ottenuto.

b) Se possibile, si riordini il codice in modo tale da ridurre ulteriormente il valore del CPI ottenuto al punto a).
   - Disegnare il diagramma a ciclo multiplo della pipeline ed individuare le criticità sui dati di tipo RAW e le criticità sul controllo rimaste dopo lo scheduling; indicare i percorsi di forwarding utilizzati.
   - Inserire gli stalli necessari a risolvere le criticità rimaste.
   - Calcolare il numero totale di stalli inseriti ed il CPI asintotico ottenuto.

Esercizio 8)

Sia dato il seguente frammento di codice in assembler MIPS.

```
      addi $s1, $0, 2
L1:   lw $t0, 0($s0)
      lw $t1, 4($s0)
      add $t2, $t0, $t1
      sw $t1, 0($s0)
      sw $t2, 4($s0)
      addi $s1, $s1, -1
      bne $s1, $0, L1
```

a) Si supponga che la pipeline sia priva di ottimizzazioni.
   - Disegnare il diagramma a ciclo multiplo della pipeline ed individuare le criticità sui dati di tipo RAW e le criticità sul controllo.

- Inserire gli stalli necessari a risolvere le criticità individuate.
- Calcolare il numero totale di stalli inseriti ed il numero totale di cicli di clock necessari ad eseguire l'intero ciclo (fino al caricamento dell'istruzione successiva all'ultima `bne`).

b) Si supponga che nella pipeline siano state introdotte le seguenti ottimizzazioni:
   - nel banco dei registri è possibile la lettura e la scrittura nello stesso ciclo di clock;
   - forwarding dei dati;
   - anticipazione del salto nello stadio ID.
   - Disegnare il diagramma a ciclo multiplo della pipeline ed individuare le criticità sui dati di tipo RAW e le criticità sul controllo rimaste; indicare i percorsi di forwarding utilizzati.
   - Inserire gli stalli necessari a risolvere le criticità eventualmente rimaste.
   - Calcolare il numero totale di stalli inseriti ed il numero totale di cicli di clock necessari ad eseguire l'intero ciclo.

c) Se possibile, si riordini il codice in modo tale da eliminare gli stalli rimasti al punto b).
   - Disegnare il diagramma a ciclo multiplo della pipeline ed individuare le criticità sui dati di tipo RAW e le criticità sul controllo rimaste dopo il riordino; indicare i percorsi di forwarding utilizzati.
   - Calcolare il numero totale di cicli di clock necessari ad eseguire l'intero ciclo.