**ARCHITETTURE AVANZATE DEI CALCOLATORI - Prima prova in itinere del 5/11/2005**
**Compito B**
**Soluzione Domanda 3 e Esercizi da 1 a 3**

**Soluzione Domanda 3** (solo accuratezza predittore)

Si evidenziano in rosso le predizioni sbagliate.

Usando il predittore statico Branch Always Not Taken si ha il seguente risultato di predizione (N: not taken, T:taken)

N N N N N N N N N N N N

Pertanto l'accuratezza il predittore statico Branch Always Taken è pari a 6/12 = 50%

Usando il predittore dinamico 2-bit BHT inizializzato a weakly predict taken (10) si ha il seguente risultato di predizione (N: not taken, T:taken)

10 11 11 11 10 01 00 01 10 11 10 01

T T T T T N N N T T T N

dove sulla prima riga si è riportato il valore assunto dal contatore a 2 bit in saturazione.
Pertanto l'accuratezza il predittore dinamico 2-bit BHT è pari a:

#predizioni esatte / # predizioni totali = 6/12 = 50%

**Soluzione Esercizio 1**
**a)**
Scheduling efficiente su processore MIPS con pipeline statica two-issue

| **Istruzione ALU o branch** | **Istruzione load o store** | **Ciclo di clock** |
|---|---|---|
| `addi $t2, $t2, -1` | `Loop: lw $t1, BASEB($s1)` | 1 |
| `Nop` | `lw $t0, BASEA($s1)` | 2 |
| `sll $t1, $t1, 1` | `nop` | 3 |
| `add $t0, $t0, $t1` | `nop` | 4 |
| `bnez $t2, Loop` | `sw $t0, BASEA($s1)` | 5 |
| `addi $s1, $s1, -4` | `nop` | 6 |

$CPI_{2\text{-issue}} = 6/8 = 0{,}75$
$CPI_{\text{ideale 2-issue}} = 4/8 = 0{,}5$

**b)**
Loop unrolling

```
Loop: lw $t0, BASEA($s1)
      lw $t1, BASEB($s1)
      sll $t1, $t1, 1
      add $t0, $t0, $t1
      sw $t0, BASEA($s1)
      lw $t0, BASEA($s1)
      lw $t1, BASEB($s1)
      sll $t1, $t1, 1
      add $t0, $t0, $t1
      sw $t0, BASEA($s1)
      addi $t2, $t2, -2
      addi $s1, $s1, -8
      bnez $t2, Loop
```

Ridenominazione

```
Loop: lw $t0, BASEA($s1)
      lw $t1, BASEB($s1)
      sll $t1, $t1, 1
      add $t0, $t0, $t1
      sw $t0, BASEA($s1)
      lw $t3, (BASEA-4)($s1)
      lw $t4, (BASEB-4)($s1)
      sll $t4, $t4, 1
      add $t3, $t3, $t4
      sw $t3, (BASEA-4)($s1)
      addi $t2, $t2, -2
      addi $s1, $s1, -8
```

```
        bnez $t2, Loop
```

Riordinamento

```
Loop: lw $t0, BASEA($s1)
      lw $t1, BASEB($s1)
      lw $t3, (BASEA-4)($s1)
      lw $t4, (BASEB-4)($s1)
      sll $t1, $t1, 1
      sll $t4, $t4, 1
      add $t0, $t0, $t1
      add $t3, $t3, $t4
      sw $t0, BASEA($s1)
      addi $t2, $t2, -2
      sw $t3, (BASEA-4)($s1)
      bnez $t2, Loop
      addi $s1, $s1, -8
```

Il valore del CPI per una generica iterazione del ciclo originale prima di effettuare il loop unrolling è:

$CPI_{orig} = (8+2)/8 = 10/8 = 1,25$

poiché sulla pipeline ottimizzata è necessario aggiungere 2 stalli:

- 1 stallo tra `lw` e `sll` (criticità RAW di tipo load/use),
- 1 stallo dopo `bne` (criticità di controllo).

Il valore del CPI per una generica iterazione del ciclo modificato dopo aver effettuato il loop unrolling ed il riordinamento è:

$CPI_{fin} = 13/13 = 1$

poiché sulla pipeline ottimizzata non sono necessari stalli.

Il numero di cicli di clock necessari ad eseguire l'intero ciclo prima di effettuare il loop unrolling (trascurando i cicli di clock necessari a riempire la pipeline) è:

$\text{num cicli clock}_{orig} = \text{num iterazioni}_{orig} * \text{num cicli clock per iterazione}_{orig} = 600 * 10 = 6000$

Il numero di cicli di clock necessari ad eseguire l'intero ciclo dopo aver effettuato il loop unrolling ed il riordinamento (trascurando i cicli di clock necessari a riempire la pipeline) è:

$\text{num cicli clock}_{fin} = \text{num iterazioni}_{fin} * \text{num cicli clock per iterazione}_{fin} = 300 * 13 = 3900$

*(facoltativo)*

Ai fini del CPI ottenibile dopo aver effettuato il loop unrolling ed il riordinamento non è necessario srotolare con un fattore di srotolamento maggiore di 2, perché il CPI che si ottiene dopo lo srotolamento pari a 2 ed il riordinamento è già quello ideale.

Con un maggior fattore di srotolamento (ad esempio pari a 4) diminuisce comunque il numero di cicli di clock necessari ad eseguire l'intero ciclo dopo aver effettuato il loop unrolling ed il riordinamento, in quanto esso diviene pari a:

$\text{num cicli clock}_{fin4} = \text{num iterazioni}_{fin4} * \text{num cicli clock per iterazione}_{fin4} = 150 * 23 = 3450$

Il numero di cicli di clock per iterazione è pari a 23 poiché le istruzioni del corpo del ciclo sono 5, mentre le istruzioni di controllo del ciclo sono 3, quindi: 4*5 + 3 = 23.

Con un fattore di srotolamento pari a 2 si ha uno speedup pari a 6000/3900 = 1,538; srotolando ancora 2 volte lo speedup è 3900/3450 = 1,13. Da notare la diminuzione dello speedup e l'aumento del numero di linee di codice al crescere del fattore di srotolamento.

**Soluzione Esercizio 2**

**a)**

Si individuano le criticità sui dati (in rosso) e sul controllo (in blu) presenti nel codice

| Istruzione | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `Loop: lw $t0, BASEA($t4)` | IF | ID | EX | MEM | WB | | | | | | | | | | | |
| `addi $t0, $t0, 2` | | IF | ID | EX | MEM | WB | | | | | | | | | | |
| `sw $t0, BASEA($t4)` | | | IF | ID | EX | MEM | WB | | | | | | | | | |
| `addi $t1, $t0, 4` | | | | IF | ID | EX | MEM | WB | | | | | | | | |
| `sw $t1, BASEB($t4)` | | | | | IF | ID | EX | MEM | WB | | | | | | | |
| `addi $t2, $t0, 8` | | | | | | IF | ID | EX | MEM | WB | | | | | | |
| `sw $t2, BASEC($t4)` | | | | | | | IF | ID | EX | MEM | WB | | | | | |
| `addi $t3, $t0, 12` | | | | | | | | IF | ID | EX | MEM | WB | | | | |
| `sw $t3, BASED($t4)` | | | | | | | | | IF | ID | EX | MEM | WB | | | |
| `addi $t4, $t4, 4` | | | | | | | | | | IF | ID | EX | MEM | WB | | |
| `bne $t5, $t4, Loop` | | | | | | | | | | | IF | ID | EX | MEM | WB | |
| | | | | | | | | | | | | IF | ID | EX | MEM | WB |

Si indicano nella seconda colonna il numero di stalli da inserire prima di ciascuna istruzione in modo da risolvere le criticità presenti e nella terza colonna il tipo di criticità.

| Istruzione | Numero stalli | Tipo di criticità |
|---|---|---|
| `Loop: lw $t0, BASEA($t4)` | | |
| `addi $t0, $t0, 2` | 3 | D (load/use) |
| `sw $t0, BASEA($t4)` | 3 | D (load/use) |
| `addi $t1, $t0, 4` | | D (define/use) |
| `sw $t1, BASEB($t4)` | 3 | D (define/use) |
| `addi $t2, $t0, 8` | | |
| `sw $t2, BASEC($t4)` | 3 | D (define/use) |
| `addi $t3, $t0, 12` | | |
| `sw $t3, BASED($t4)` | 3 | D (define/use) |
| `addi $t4, $t4, 4` | | |
| `bne $t5, $t4, Loop` | 3 | D (define/use) |
| | 3 | C |

Numero totale di stalli inseriti = 21

$CPI_{as}$ = (11 + 21)/11 = 32/11 = 2,9

Throughput in MIPS = $f_{clock}$/ ($CPI_{as}$*$10^6$) = (0,5*$10^9$)/(2,9*$10^6$ ) = 500/2,9 = 172,41

**b)**
Si individuano le criticità sui dati (in rosso) e sul controllo (in blu) che non possono essere risolte solo mediante forwarding e si indicano (in verde) i percorsi di forwarding usati e (in arancione) la lettura/scrittura del banco dei registri.

| Istruzione | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `Loop: lw $t0, BASEA($t4)` | IF | ID | EX | MEM | WB | | | | | | | | | | | |
| `addi $t0, $t0, 2` | | IF | ID | EX | MEM | WB | | | | | | | | | | |
| `sw $t0, BASEA($t4)` | | | IF | ID | EX | MEM | WB | | | | | | | | | |
| `addi $t1, $t0, 4` | | | | IF | ID | EX | MEM | WB | | | | | | | | |
| `sw $t1, BASEB($t4)` | | | | | IF | ID | EX | MEM | WB | | | | | | | |
| `addi $t2, $t0, 8` | | | | | | IF | ID | EX | MEM | WB | | | | | | |
| `sw $t2, BASEC($t4)` | | | | | | | IF | ID | EX | MEM | WB | | | | | |
| `addi $t3, $t0, 12` | | | | | | | | IF | ID | EX | MEM | WB | | | | |
| `sw $t3, BASED($t4)` | | | | | | | | | IF | ID | EX | MEM | WB | | | |
| `addi $t4, $t4, 4` | | | | | | | | | | IF | ID | EX | MEM | WB | | |
| `bne $t5, $t4, Loop` | | | | | | | | | | | IF | ID | EX | MEM | WB | |
| | | | | | | | | | | | | IF | ID | EX | MEM | WB |

Si indicano nella seconda colonna il numero di stalli da inserire prima di ciascuna istruzione in modo da risolvere le criticità presenti e nella terza colonna il percorso di forwarding, specificando il registro di pipeline coinvolto.

| Istruzione | Numero stalli | Percorso di forwarding |
|---|---|---|
| `Loop: lw $t0, BASEA($t4)` | | |
| `addi $t0, $t0, 2` | 1 | da MEM/WB a EX |
| `sw $t0, BASEA($t4)` | | da EX/MEM a EX |
| `addi $t1, $t0, 4` | | da MEM/WB a EX |
| `sw $t1, BASEB($t4)` | | da EX/MEM a EX |
| `addi $t2, $t0, 8` | | |
| `sw $t2, BASEC($t4)` | | da EX/MEM a EX |
| `addi $t3, $t0, 12` | | |
| `sw $t3, BASED($t4)` | | da EX/MEM a EX |
| `addi $t4, $t4, 4` | | |
| `bne $t5, $t4, Loop` | 1 | da EX/MEM a ID |
| | 1 | |

Numero totale di stalli inseriti = 3
$CPI_{as}$ = (11 + 3)/11 = 14/11 = 1,27
Throughput in MIPS = $f_{clock}$/ ($CPI_{as}$*$10^6$) = (0,5*$10^9$)/(1,27*$10^6$ ) = 500/1,27 = 393,7

**Soluzione Esercizio 3**

E' presente una criticità sui dati di tipo define/use, che non viene risolta dal processore MIPS considerato a lezione con pipeline ottimizzata. Per risolverla, occorre infatti inserire uno stallo tra l'istruzione `sub` e l'istruzione `beq` ed effettuare la propagazione del risultato della sottrazione dal registro di pipeline EX/MEM allo stadio ID. E' quindi necessario determinare le opportune condizioni per il rilevamento e la gestione della criticità.

L'unità di rilevamento della criticità posizionata nello stadio ID rileva lo stallo quando l'istruzione `sub` è nello stadio EX e l'istruzione `beq` è nello stadio ID. Le condizioni per rilevare la criticità sono quindi:

```
if ((ID/EX.RegWrite and (EX/MEM.MemtoReg = 0)) and // sub in EX?
    IF/ID.Branch and // beq in ID?
    ((ID/EX.RegisterRd = IF/ID.RegisterRs) or
     (ID/EX.RegisterRd = IF/ID.RegisterRt)) and // stesso registro?
    (ID/EX.RegisterRd ≠ 0)) // ma diverso da 0?
stall the pipeline
```

Il segnale `IF/ID.Branch` non è prelevabile direttamente dal registro di pipeline IF/ID ma occorre generarlo riconoscendo che i 6 bit del codice operativo (Instruction[5-0]) corrispondono all'istruzione `beq`.

Per propagare il risultato dell'istruzione `sub` al comparatore nello stadio ID occorre inserire un percorso di forwarding dal registro di pipeline EX/MEM all'ingresso del comparatore posto a valle del banco dei registri. Occorre quindi aggiungere due multiplexer davanti ai due ingressi del comparatore, che sono controllati dai nuovi segnali di controllo `Cmp1` e `Cmp2`, come mostrato in figura. Questi segnali sono valorizzati dall'unità di propagazione posta nello stadio EX in base al verificarsi delle seguenti condizioni:

```
if ((EX/MEM.RegWrite and (EX/MEM.MemtoReg = 0)) and // sub in MEM?
    IF/ID.Branch and // beq in ID?
    (EX/MEM.RegisterRd = IF/ID.RegisterRs) and // stesso registro?
    (EX/MEM.RegisterRd ≠ 0)) // ma diverso da 0?
Cmp1 = 1

if ((EX/MEM.RegWrite and (EX/MEM.MemtoReg = 0)) and // sub in MEM?
    IF/ID.Branch and // beq in ID?
    (EX/MEM.RegisterRd = IF/ID.RegisterRt) and // stesso registro?
    (EX/MEM.RegisterRd ≠ 0)) // ma diverso da 0?
Cmp2 = 1
```