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

Soluzione Domanda 3 (solo accuratezza predittore)

Nella soluzione si sono evidenziate in rosso le predizioni sbagliate.

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

T T T T T T T T T T T T

Pertanto l'accuratezza il predittore statico Branch Always Taken è pari a:

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

Usando il predittore dinamico 2-bit BHT inizializzato a weakly predict not taken (01) si ha il seguente risultato di predizione (N: not taken, T:taken)

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

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

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 $t0, $t0, 1`	`Loop: lw $t2, BASEB($s0)`	1
`nop`	`lw $t1, BASEA($s0)`	2
`srl $t2, $t2, 1`	`nop`	3
`sub $t2, $t1, $t2`	`nop`	4
`bne $t0, $s1, Loop`	`sw $t2, BASEA($s0)`	5
`addi $s0, $s0, 4`	`nop`	6

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

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

b)

Loop unrolling

```
Loop: lw $t1, BASEA($s0)
      lw $t2, BASEB($s0)
      srl $t2, $t2, 1
      sub $t2, $t1, $t2
      sw $t2, BASEA($s0)
      lw $t1, (BASEA+4)($s0)
      lw $t2, (BASEB+4)($s0)
      srl $t2, $t2, 1
      sub $t2, $t1, $t2
      sw $t2, (BASEA+4)($s0)
      addi $t0, $t0, 2
      addi $s0, $s0, 8
      bne $t0, $s1, Loop
```

Ridenominazione

```
Loop: lw $t1, BASEA($s0)
      lw $t2, BASEB($s0)
      srl $t2, $t2, 1
      sub $t2, $t1, $t2
      sw $t2, BASEA($s0)
      lw $t3, (BASEA+4)($s0)
      lw $t4, (BASEB+4)($s0)
      srl $t4, $t4, 1
      sub $t4, $t3, $t4
      sw $t4, (BASEA+4)($s0)
      addi $t0, $t0, 2
```

```
        addi $s0, $s0, 8
        bne $t0, $s1, Loop
```

Riordinamento

```
Loop: lw $t1, BASEA($s0)
      lw $t2, BASEB($s0)
      lw $t3, (BASEA+4)($s0)
      lw $t4, (BASEB+4)($s0)
      srl $t2, $t2, 1
      srl $t4, $t4, 1
      sub $t2, $t1, $t2
      sub $t4, $t3, $t4
      sw $t2, BASEA($s0)
      addi $t0, $t0, 2
      sw $t4, (BASEA+4)($s0)
      bne $t0, $s1, Loop
      addi $s0, $s0, 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 `srl` (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:

$\text{num cicli clock}_{orig} = \text{num iterazioni}_{orig} * \text{num cicli clock per iterazione}_{orig} = 500 * 10 = 5000$

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:

$\text{num cicli clock}_{fin} = \text{num iterazioni}_{fin} * \text{num cicli clock per iterazione}_{fin} = 250 * 13 = 3250$

(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 pari a 1.

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:

$\text{num cicli clock}_{fin4} = \text{num iterazioni}_{fin4} * \text{num cicli clock per iterazione}_{fin4} = 125 * 23 = 2875$

In questo caso il numero di cicli di clock per iterazione è pari a 23, perché 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 5000/3250 = 1,538; srotolando ancora 2 volte lo speedup è 3250/2875 = 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
`Loop: lw $t2, BASEA($t6)`	IF	ID	EX	MEM	WB										
`sw $t2, BASEB($t6)`		IF	ID	EX	MEM	WB									
`lw $t3, BASEC($t6)`			IF	ID	EX	MEM	WB								
`sw $t3, BASED($t6)`				IF	ID	EX	MEM	WB							
`lw $t4, BASEE($t6)`					IF	ID	EX	MEM	WB						
`add $t4, $t4, $t2`						IF	ID	EX	MEM	WB					
`add $t4, $t4, $t3`							IF	ID	EX	MEM	WB				
`sw $t4, BASEF($t6)`								IF	ID	EX	MEM	WB			
`addi $t6, $t6, 4`									IF	ID	EX	MEM	WB		
`bne $t6, $t7, 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 $t2, BASEA($t6)`		
`sw $t2, BASEB($t6)`	3	D (load/use)
`lw $t3, BASEC($t6)`		
`sw $t3, BASED($t6)`	3	D (load/use)
`lw $t4, BASEE($t6)`		
`add $t4, $t4, $t2`	3	D (load /use)
`add $t4, $t4, $t3`	3	D (define/use)
`sw $t4, BASEF($t6)`	3	D (define/use)
`addi $t6, $t6, 4`		
`bne $t6, $t7, Loop`	3	D (define/use)
	3	C

Numero totale di stalli inseriti = 21

$CPI_{as} = (10 + 21)/10 = 31/10 = 3,1$

Throughput in MIPS = $f_{clock}/ (CPI_{as}*10^6) = 10^9/(3,1*10^6) = 10^3/3,1 = 322,58$

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
`Loop: lw $t2, BASEA($t6)`	IF	ID	EX	MEM	WB										
`sw $t2, BASEB($t6)`		IF	ID	EX	MEM	WB									
`lw $t3, BASEC($t6)`			IF	ID	EX	MEM	WB								
`sw $t3, BASED($t6)`				IF	ID	EX	MEM	WB							
`lw $t4, BASEE($t6)`					IF	ID	EX	MEM	WB						
`add $t4, $t4, $t2`						IF	ID	EX	MEM	WB					
`add $t4, $t4, $t3`							IF	ID	EX	MEM	WB				
`sw $t4, BASEF($t6)`								IF	ID	EX	MEM	WB			
`addi $t6, $t6, 4`									IF	ID	EX	MEM	WB		
`bne $t6, $t7, 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 $t2, BASEA($t6)`		
`sw $t2, BASEB($t6)`		da MEM a EX/MEM *oppure* da MEM/WB a MEM
`lw $t3, BASEC($t6)`		
`sw $t3, BASED($t6)`		da MEM a EX/MEM *oppure* da MEM/WB a MEM
`lw $t4, BASEE($t6)`		
`add $t4, $t4, $t2`	1	da MEM/WB a EX
`add $t4, $t4, $t3`		da EX/MEM a EX
`sw $t4, BASEF($t6)`		da EX/MEM a EX
`addi $t6, $t6, 4`		
`bne $t6, $t7, Loop`	1	da EX/MEM a ID
	1	

Numero totale di stalli inseriti = 3
CPI_{as} = (10 + 3)/10 = 13/10 = 1,3
Throughput in MIPS = f_{clock}/ (CPI_{as}*10^6) = 10^9/(1,3*10^6) = 10^3/1,3 = 769,23

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 `addi` e l'istruzione `bne` ed effettuare la propagazione del risultato dell'addizione 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 `addi` è nello stadio EX e l'istruzione `bne` è nello stadio ID. Le condizioni per rilevare la criticità sono quindi:

```
if ((ID/EX.RegWrite and (ID/EX.MemtoReg = 0)) and //addi in EX?
    IF/ID.Branch and // bne in ID?
    ((ID/EX.RegisterRt = IF/ID.RegisterRs) or
     (ID/EX.RegisterRt = IF/ID.RegisterRt)) and // stesso registro?
    (ID/EX.RegisterRt ≠ 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 `bne`.

Per propagare il risultato dell'istruzione `addi` 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 //addi in MEM?
    IF/ID.Branch and // bne 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 //addi in MEM?
    IF/ID.Branch and // bne in ID?
    (EX/MEM.RegisterRd = IF/ID.RegisterRt) and // stesso registro?
    (EX/MEM.RegisterRd ≠ 0)) // ma diverso da 0?
Cmp2 = 1
```