

# Compito del Corso di Architettura degli Elaboratori 1

Anno Accademico 2004/2005

Appello del 22 Marzo 2005 - Soluzione esercizi pratici

## Istruzioni

- Scrivere *Nome, Cognome e Matricola* su **ogni** foglio.
- Scrivere la risposta nello spazio bianco al di sotto della domanda; Non è possibile allegare fogli aggiuntivi, quindi cercate di essere chiari e non prolissi.
- In caso di errori indicate chiaramente quale parte della risposta deve essere considerata; annullate le parti non pertinenti.
- Assicuratevi che non manchi alcun foglio al momento della consegna.

## Esercizi pratici

### es9

Sia data la seguente sequenza di istruzioni assembler, dove i dati immediati sono espressi in esadecimale ed il registro R0 contiene il valore 0:

```

LB   R3, 100(R0)    ! load byte da mem[100+[R0]]
ADD  R2, R0, R0     ! R2 = R0 + R0
LB   R1, 108(R2)    ! load byte da mem[108+[R2]]
ADDI R2, R2, 2      ! R2 = R2 + 2
SUB  R4, R3, R2     ! R4 = R3 - R2
ADDI R1, R1, 3      ! R1 = R1 + 3
SB   R1, 108(R2)    ! store byte in mem[108+[R2]]
BGTZ R4, -6         ! PC = PC - 6 se [R4] > 0
                        ! cioe' salta alla istruzione LB   R1, 108(R2)

```

Si assuma la presenza di due cache, una dati ed una istruzioni. La cache dati, in particolare, è di ampiezza 16B, con dimensione di blocco 4B, inizialmente vuota, ed associazione 2-way (con politica di rimpiazzo LRU e politica di scrittura write-through). Si assuma che la memoria abbia il contenuto esadecimale mostrato di seguito (si esprimano gli indirizzi su 12 bit):

Indirizzo	byte	byte	byte	byte
100	08	00	07	02
104	00	00	00	00
108	AE	13	A1	23
10C	A1	42	90	75
110	B9	16	00	00
114	0A	07	03	71

Si mostri come sia il contenuto della cache dati che il contenuto della memoria cambia a causa della esecuzione del codice assembler.

## Soluzione

Poiché un blocco è costituito da 4B, e la cache è di 16B si avranno  $16/4 = 4$  linee. Essendo l'associatività a due linee (2-way), ci saranno due insiemi (insieme 0 e insieme 1) ognuno di 2 linee. Quindi i 12 bit di indirizzo saranno suddivisi nel seguente modo: i 2 bit meno significativi individueranno il byte all'interno del blocco, il terzo bit da destra individuerà l'insieme (0 o 1), ed i restanti bit costituiranno il tag. Mostriamo di seguito l'evoluzione del contenuto dei registri, dei riferimenti a memoria, della cache dati (solo quando cambia) e della memoria (solo quando cambia). Per la cache dati, nel caso in cui tutte e due le linee di un insieme (set) siano libere, si sceglie la linea con indirizzo minore per la allocazione (scelta arbitraria: si poteva usare un criterio diverso).

codice eseguito	[R1]	[R2]	[R3]	[R4]	ind. rif. memoria hex binario	cache dati		modifica memoria mem[ind.] = cont. mem[ind.] = cont.
	hex	hex	hex	hex		set 0 [ linea 0 ] t: tag r: rif. [ linea 1 ] t: tag r: rif.	set 1 [ linea 2 ] t: tag r: rif. [ linea 3 ] t: tag r: rif.	
LB R3, 100(R0)	?	?	8	?	100 000100000000	[08 00 07 02] t:000100000 r:miss [ ] t: r:		
ADD R2, R0, R0	?	0	8	?				
LB R1, 108(R2)	AE	0	8	?	108 000100001000	[08 00 07 02] t:000100000 r: [AE 13 A1 23] t:000100001 r: miss		
ADDI R2, R2, 2	AE	2	8	?				
SUB R4, R3, R2	AE	2	8	6				
ADDI R1, R1, 3	B1	2	8	6				
SB R1, 108(R2)	B1	2	8	6	10A 000100001010	[08 00 07 02] t:000100000 r: [AE 13 A1 23] t:000100001 r: hit		
						[08 00 07 02] t:000100000 r: [AE 13 B1 23] t:000100001 r:write-th.		mem[10A] = [R1] = B1
BGTZ R4, -6	B1	2	8	6				
LB R1, 108(R2)	B1	2	8	6	10A 000100001010	[08 00 07 02] t:000100000 r: [AE 13 B1 23] t:000100001 r: hit		
ADDI R2, R2, 2	B1	4	8	6				
SUB R4, R3, R2	B1	4	8	4				
ADDI R1, R1, 3	B4	4	8	4				
SB R1, 108(R2)	B4	4	8	4	10C 000100001100	[08 00 07 02] t:000100000 r: [A1 42 90 75] t:000100001 r:miss		

						[AE 13 B1 23]	[		]	
						t:000100001	t:			
						r:	r:			
						[08 00 07 02]	[B4 42 90 75]			
						t:000100000	t:000100001			
						r:	r:write-th.			
						[AE 13 B1 23]	[		]	mem[10C] = [R1]
						t:000100001	t:			= B4
						r:	r:			
BGTZ R4, -6	B4	4	8	4						
LB R1, 108(R2)	B4	4	8	4	10C	[08 00 07 02]	[B4 42 90 75]			
					000100001100	t:000100000	t:000100001			
						r:	r:hit			
						[AE 13 B1 23]	[		]	
						t:000100001	t:			
						r:	r:			
ADDI R2, R2, 2	B4	6	8	4						
SUB R4, R3, R2	B4	6	8	2						
ADDI R1, R1, 3	B7	6	8	2						
SB R1, 108(R2)	B7	6	8	2	10E	[08 00 07 02]	[B4 42 90 75]			
					000100001110	t:000100000	t:000100001			
						r:	r:hit			
						[AE 13 B1 23]	[		]	
						t:000100001	t:			
						r:	r:			
						[08 00 07 02]	[B4 42 B7 75]			mem[10E] = [R1]
						t:000100000	t:000100001			= B7
						r:	r:write-th.			
						[AE 13 B1 23]	[		]	
						t:000100001	t:			
						r:	r:			
BGTZ R4, -6	B7	6	8	2						
LB R1, 108(R2)	B7	6	8	2	10E	[08 00 07 02]	[B4 42 B7 75]			
					000100001110	t:000100000	t:000100001			
						r:	r:hit			
						[AE 13 B1 23]	[		]	
						t:000100001	t:			
						r:	r:			
ADDI R2, R2, 2	B7	8	8	2						
SUB R4, R3, R2	B7	8	8	0						
ADDI R1, R1, 3	BA	8	8	0						
SB R1, 108(R2)	BA	8	8	0	110	[B9 16 00 00]	[B4 42 B7 75]			
					000100010000	t:000100010	t:000100001			
						r:miss (LRU)	r:			
						[AE 13 B1 23]	[		]	
						t:000100001	t:			
						r:	r:			
						[BA 16 00 00]	[B4 42 B7 75]			mem[110] = [R1]
						t:000100010	t:000100001			= BA
						r:write-th.	r:			
						[AE 13 B1 23]	[		]	
						t:000100001	t:			
						r:	r:			
BGTZ R4, -6	BA	8	8	0						

**es10**

Si consideri il codice assembler dell'esercizio precedente (es9) e la pipeline MIPS a 5 stadi vista a lezione, senza possibilità di data-forwarding, ma possibilità di scrittura e successiva lettura dei registri in uno stesso ciclo di clock. Si assuma che ogni operazione di memoria impieghi un solo ciclo di clock e che i salti condizionali siano predetti come "taken" (presi). Si mostri il diagramma degli stadi della pipeline per l'esecuzione del codice fino alla prima occorrenza (inclusa) della istruzione `BGTZ R4, -6`.

Dire inoltre quanti cicli occorrono per completare l'esecuzione del codice.

**Soluzione**

Non essendoci possibilità di data-forwarding, tutte le dipendenze da dati che non si riducono a scrittura e successiva lettura di un registro provocano stallo, come descritto di seguito

istruzione	C I C L I C L O C K																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LB R3, 100(R0)	IF	ID	EX	ME	WB												
ADD R2, R0, R0		IF	ID	EX	ME	WB											
LB R1, 108(R2)			IF	ID	ID	ID	EX	ME	WB								
ADDI R2, R2, 2				IF	IF	IF	ID	EX	ME	WB							
SUB R4, R3, R2							IF	ID	ID	ID	EX	ME	WB				
ADDI R1, R1, 3								IF	IF	IF	ID	EX	ME	WB			
SB R1, 108(R2)											IF	ID	ID	ID	EX	ME	WB
BGTZ R4, -6												IF	IF	IF	ID	EX	ME

Il fetch della istruzione dopo il primo salto (predetto correttamente come preso) avviene al ciclo 11, mentre per il secondo salto avviene al ciclo 19, e per il terzo salto avviene al ciclo 27, ed infine al ciclo 37 si completa l'ultima istruzione di salto (non preso) con la fase MEM.