Esercizio Pipeline MIPS

Soluzione

Considerando la pipeline MIPS vista a lezione, si consideri il seguente frammento di codice:

loop:	LW	$1, 0($2)	**R1← mem[0+[R2]]**
	ADDI	$1,$1, 1	**R1← [R1] + 1**
	SW	$1, 0($2)	**mem[0+[R2]] ← [R1]**
	ADDI	$2, $2, 4	**R2← [R2] + 4**
	SUB	$4, $3, $2	**R4← [R3] - [R2]**
	BENZ	$4, loop	**if([R4] != 0) PC ← indirizzo(loop)**

assumendo che il valore iniziale di R3 sia R2+396.

a) si individuino e discutano le dipendenze dovute ai dati

Soluzione:

DIPENDENZE	**[dipendenza dati (senza considerare limiti architettura MIPS)]** **[dipendenza dati considerando i limiti della architettura MIPS]**
R1 in ADDI $1,**$1**, 1	**[input EXE_{ADDI} ha bisogno di output da MEM_{LW}]**

dipende da LW $1, 0($2)	**[ID_{ADDI} deve legge R1 aggiornato da WB_{LW} (stesso ciclo clock)]**
R1 in SW $1,0($2) dipende da LW $1, 0($2)	**[input MEM_{SW} ha bisogno di output da MEM_{LW}]** **[ID_{SW} deve legge R1 aggiornato da WB_{LW} (stesso ciclo clock)]**
R1 in SW $1,0($2) dipende da ADDI $1,$1, 1	**[input MEM_{SW} ha bisogno di output da EXE_{ADDI}]** **[ID_{SW} deve legge R1 aggiornato da WB_{ADDI} (stesso ciclo clock)]**
R2 in SUB $4, $3, $2 dipende da ADDI $2,$2, 4	**[input EXE_{SUB} ha bisogno di output da EXE_{ADDI}]** **[ID_{SUB} deve legge R2 aggiornato da WB_{ADDI} (stesso ciclo clock)]**
R4 in BENZ $4, loop dipende da SUB $4, $3, $2	**[input EXE_{BENZ} ha bisogno di output da EXE_{SUB}]** **[ID_{BENZ} deve legge R4 aggiornato da WB_{SUB} (stesso ciclo clock)]**

b) mostrare come evolve la pipeline durante l'esecuzione del codice per le prime 6 istruzioni eseguite, assumendo:

- possibilità di forwarding, così come visto a lezione per la pipeline MIPS;
- che il salto condizionale (BENZ) sia trattato con stallo della pipeline fino al calcolo dell'indirizzo target.

Si calcoli inoltre il numero totale di cicli di clock necessari per portare a termine l'esecuzione completa del codice.

Soluzione:

Evoluzione pipeline per le prime 6 istruzioni eseguite

			1	2	3	4	5	6	7	8	9	10	11	12	13
loop:	LW	$1,0($2)	IF	ID	EX	MEM	WB								
	ADDI	$1,$1, 1		IF	ID	ID	EX	MEM	WB						
	SW	$1, 0($2)			IF	IF	ID	ID	ID	EX	MEM	WB			
	ADDI	$2,$2, 4					IF	IF	IF	ID	EX	MEM	WB		
	SUB	$4,$3,$2								IF	ID	EX	MEM	WB	
	BENZ	$4, loop									IF	ID	EX	MEM	WB
		non preso/preso										IF	IF	IF **(preso: LW R1,0(R2)**	ID

Notare che la SW deve aspettare che $1 sia scritto perché non è previsto un circuito di bypass in grado di catturare l'uscita della ALU della istruzione ADDI precedente (fase EX) e di portare il dato in ingresso alla memoria durante la fase MEM di SW.

Il numero totale di cicli è calcolato come segue:

numero di iterazioni del ciclo = (396/4) = 99

numero cicli di clock per eseguire il codice = 98 * 11 (si sovrappone MEM_{BENZ} della iterazione *i* con IF_{LW} della iterazione *i+1*) + 1*13 = **1091**

Se si considera che lo stadio WB di BENZ in effetti non fa nulla, i cicli sono 1090.