Esercizio Pipeline 1: Dipendenze

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]
      ADD $2, $1, $2      ! R2← [R1] + [R2]
      SUB $4, $3, $2      ! R4← [R3] - [R2]
      BENZ $4, LOOP       ! if([R4] != 0) PC ← indirizzo(loop)
```

si individuino le dipendenze **R**ead**A**fter**W**rite (RAW) e **W**rite**A**fter**W**rite (WAW).

Soluzione

#	codice	R1	R2	R3	R4	commento
1	LOOP: LW $1, 0($2)	**W**	**R**			legge R2, scrive R1
2	ADDI $1,$1, 1	**RW**				legge e scrive R1
3	SW $1, 0($2)	**R**	**R**			legge R1 e R2
4	ADD $2, $1, $2	**R**	**RW**			legge R1, legge e scrive R2
5	SUB $4, $3, $2		**R**	**R**	**W**	legge R2 3 R3, scrive R4
6	BENZ $4, LOOP				**R**	legge R4

Linee codice	Spiegazione dipendenza	Tipo
2←1	ADDI legge R1 che è scritto da LW	RAW
2←1	ADDI scrive R1 che è scritto da LW	WAW
3←2, 3←1	SW legge R1 che è scritto da ADDI, e prima da LW	RAW
4←2, 4←1	ADD legge R1 che è scritto da ADDI, e prima da LW	RAW
5←4	SUB legge R2 che è scritto da ADD	RAW
6←5	BENZ legge R4 che è scritto da SUB	RAW

Esercizio 2: valutazione delle prestazioni

- Si considerino le seguenti statistiche:
  - 15% delle istruzioni sono di salto condizionale
  - 1% delle istruzioni sono di salto incondizionale
  - Il 60% delle istruzioni di salto condizionale hanno la condizione soddisfatta (prese)
- …ed una pipeline a 4 stadi (IF, ID, EI, WO) per cui:
  - i salti incondizionati sono risolti (identificazione salto e calcolo indirizzo target) alla fine del secondo stadio (ID)
  - i salti condizionati sono risolti (identificazione salto, calcolo indirizzo target e calcolo condizione) alla fine del terzo stadio (EI)
  - il primo stadio (IF) è indipendente dagli altri
  - ogni stadio impiega 1 ciclo di clock
- inoltre si assuma che non ci siano altre istruzioni che possano mandare in stallo la pipeline e che si predica di non saltare in caso di salto condizionale

Domanda:

calcolare quanto più veloce, a regime, sarebbe la pipeline senza gli stalli introdotti dai salti

Aiuto: fattore di velocizzazione di una pipeline a k stadi, a regime, in funzione del numero di stalli:

$$S_k = \frac{1}{1 + frazione\_cicli\_stallo}\, k$$

Soluzione: valutazione delle prestazioni

- Per rispondere alla domanda bisogna calcolare il rapporto tra le prestazioni di una pipeline a 4 stadi senza stalli con le prestazioni della pipeline con ritardi
- Le prestazioni di una pipeline a 4 stadi senza ritardi si ottengono considerando la formula data con k=4 e 0 cicli di stallo:

$$\frac{1}{1+0}4 = 4$$

- Per calcolare le prestazioni in presenza di stalli bisogna calcolare:
  - la probabilità di eseguire una delle istruzioni di salto

salto incondizionato	→	**0,01** perché 1 su 100 è un salto incondizionato
salto condizionato preso	→	0,15*0,6 = **0,09** perché 15 istr. su 100, e il 60% salta
salto condizionato non preso	→	0,15*0,4 = **0,06** perché 15 istr. su 100, e il 40% non salta

  - la frazione di cicli di stallo per tipo di istruzione di salto

    vedi prossimi lucidi

Soluzione: valutazione delle prestazioni

- Stalli per salto incondizionato (salta all'istruzione con indirizzo *j*)

	cicli clock					
istr. eseguita	1	2	3	4	5	6
jump	IF	ID	EI	WO		
i + 1		~~IF~~	*(qui la pipeline è "svuotata")*			
istr. target			IF	ID	EI	…
j + 1				IF	ID	…
j + 2					IF	…

quindi si ha **1 ciclo** di "stallo" (non è un vero e proprio stallo: la pipeline è svuotata, quindi si esegue un IF inutile)

Soluzione: valutazione delle prestazioni

- Stalli per salto condizionato **preso** (salta all'istruzione con indirizzo *j*)

	cicli clock					
istr. eseguita	1	2	3	4	5	6
branch	IF	ID	EI	WO		
i + 1		IF	~~ID~~	*(qui la pipeline è "svuotata")*		
i + 2			~~IF~~	*(qui la pipeline è "svuotata")*		
istr. target				IF	…	
j + 1					IF	…

quindi si hanno **2 cicli** di "stallo"

- Stalli per salto condizionato **non preso**

	cicli clock					
istr. eseguita	1	2	3	4	5	6
branch	IF	ID	EI	WO		
i + 1		IF	ID	EI	…	
i + 2			IF	ID	…	
i + 3				IF	…	

quindi si hanno **0 cicli** di "stallo"

Soluzione: valutazione delle prestazioni

Rappresentazione alternativa

- Stalli per salto condizionato **preso** (salta all'istruzione con indirizzo *j*)

stadi \ cicli clock	1	2	3	4	5	6
IF	branch	*i+1*	*i+2*	*istr. target*	*j+1*	*j+2*
ID		branch	*i+1*	*bubble*	*istr. target*	*j+1*
EI			branch	*bubble*	*bubble*	*istr. target*
WO				…		

Si noti che ogni stadio "perde" 2 cicli di clock:

- IF carica le istruzioni con indirizzi *i+1* e *i+2* che poi non terminano l'esecuzione;
- ID decodifica l'istruzione con indirizzo *i+1* che non termina l'esecuzione e poi rimane inattiva durante il ciclo di clock 4 (*bubble*);
- EI (e successivamente WO) rimane inattiva durante i cicli di clock 4 e 5.

Soluzione: valutazione delle prestazioni

- la frazione di cicli in cui si ha stallo è:

prob_jump*stalli_jump	[**0,01*1**]
+	+
prob_branch_preso* stalli_branch_preso	[**0,09*2**]
+	+
prob_branch_non_preso* stalli_branch_non_preso	[**0,06*0**]
	=
	0,19

- e quindi le prestazioni della pipeline con stalli è:

$$S_k = \frac{1}{1+0,19} 4 = 3,36$$