

Domanda 10 (8 punti)

Si consideri il seguente frammento di codice C:

```
for (i = 0; i < 100; i++) {  
    v5[i] = (v1[i]+v2[i]);  
    v6[i] = (v2[i]*v3[i]);  
    v7[i] = (v2[i]*v3[i])/v4[i];  
}
```

dove i vettori `v1[]`, `v2[]`, `v3[]`, `v4[]` contengono numeri Floating Point (FP), sono lunghi 100 e sono stati salvati in precedenza nella memoria; gli elementi di `v4[]` non sono mai 0. Inoltre è stato allocato in memoria lo spazio vuoto per i vettori `v5[]`, `v6[]` e `v7[]`.

Si eseguano le seguenti operazioni:

- 1) Con riferimento al programma riportato nel seguito, scritto per l'architettura del processore MIPS64 (descritta sotto), ed utilizzando gli spazi a ciò appositamente destinati, si calcoli il numero di colpi di clock richiesti per l'esecuzione dell'intero programma. L'architettura da considerare ha le seguenti caratteristiche:

- L'unità di moltiplicazione FP è un'unità pipelined a 8 stadi
- L'unità aritmetica FP è un'unità pipelined a 2 stadi
- L'unità di divisione FP è un'unico blocco con una latenza pari a 10 colpi di clock
- Il branch delay slot è pari ad 1 colpo di clock
- Il delay slot non è abilitato (ossia, la pipeline viene svuotata se il salto viene preso)
- Il data forwarding è abilitato.

Si assuma che le diverse unità funzionali che compongono lo stadio di EX possano lavorare in parallelo su istruzioni diverse: l'architettura considerata può quindi implementare un meccanismo che permette il completamento out-of-order delle istruzioni.

```

1)
; ***** WinMIPS64 *****
;   for (i = 0; i < 100; i++) {
;       v5[i] = (v1[i]+v2[i]);
;       v6[i] = (v2[i]*v3[i]);
;       v7[i] = (v2[i]*v3[i])/v4[i];
;   }
;

```

	Commenti	Colpi di clock
.data		
v1: .double "100 valori"		
v2: .double "100 valori"		
...		
v4: .double "100 valori"		
v5: .double "100 zeri"		
v6: .double "100 zeri"		
v7: .double "100 zeri"		
.text		
main: daddui r1,r0,0	r1 <= puntatore	5
daddui r2,r0,100	r2 <= 100	1
loop: l.d f1,v1(r1)	f1 <= v1[i]	1
l.d f2,v2(r1)	f2 <= v2[i]	1
mul.d f3,f1,f2	f3 <= v1[i]*v2[i]	9
l.d f4,v3(r1)	f4 <= v3[i]	0
mul.d f5,f2,f4	f5 <= v2[i]*v3[i]	3
l.d f6,v4(r1)	f6 <= v4[i]	0
div.d f7,f5,f6	f7 <= (v2[i]*v3[i])/v4[i]	10
s.d f3,v5(r1)		0
s.d f5,v6(r1)		0
s.d f7,v7(r1)		1
daddi r2,r2,-1	r2 <= r2 - 1	1
daddui r1,r1,8	r1 <= r1 + 8	1
bnez r2,loop		1
halt		1
total		

```

div eeeeeMW
s.d eeeeeEMW

div eeeeeMW
s.d eeeeeesMW

```

- 2) Si ottimizzi il programma utilizzando la tecnica nota come *loop unrolling* abilitando il *Branch Delay Slot* in maniera tale che il programma esegua lo stesso calcolo del programma precedente in un tempo minore. Si calcoli il numero di colpi di clock richiesti per l'esecuzione del nuovo programma.

3) Con riferimento all'architettura di un processore MIPS che implementa la strategia multiple-issue con speculazione (descritta sotto), si calcoli il numero di colpi di clock necessari all'esecuzione di 2 cicli del programma proposto.

L'architettura da considerare ha le seguenti caratteristiche:

- può eseguire l'issue di 2 istruzioni per colpo di clock
- in presenza di un'istruzione di salto, viene eseguita una sola issue
- può eseguire il commit di 2 istruzioni per colpo di clock
- sono disponibili le seguenti unità funzionali indipendenti:
 - i. 1 unità Memory address
 - ii. 1 unità per operazioni intere (ALU)
 - iii. 1 unità per il calcolo dei salti
 - iv. 1 unità di moltiplicazione FP pipelined a 8 stadi
 - v. 1 unità di divisione FP no pipelined (latenza 10)
 - vi. 1 unità di somma e sottrazione FP pipelined a 2 stadi
- la previsione sui salti è sempre corretta
- le cache non producono mai situazioni di miss
- sono disponibili due CDB (Common Data Bus).

# iterazione		Issue	EXE	MEM	CDB x2	COMMIT x2
1	l.d f1,v1(r1)	1	2	3	4	5
1	l.d f2,v2(r1)	1	3	4	5	6
1	mul.d f3,f1,f2	2	6		14	15
1	l.d f4,v3(r1)	2	4	5	6	15
1	mul.d f5,f2,f4	3	7		15	16
1	l.d f6,v4(r1)	3	5	6	7	16
1	div.d f7,f5,f6	4	16		26	27
1	s.d f3,v5(r1)	4	6			27
1	s.d f5,v6(r1)	5	7			28
1	s.d f7,v7(r1)	5	8			28
1	daddi r2,r2,-1	6	7		8	29
1	daddui r1,r1,8	6	8		9	29
1	bnez r2,loop	7	9			30
2	l.d f1,v1(r1)	8	9	10	11	30
2	l.d f2,v2(r1)	8	10	11	12	31
2	mul.d f3,f1,f2	9	13		21	32
2	l.d f4,v3(r1)	9	11	12	13	32
2	mul.d f5,f2,f4	10	14		22	33
2	l.d f6,v4(r1)	10	12	13	14	33
2	div.d f7,f5,f6	11	27		37	38
2	s.d f3,v5(r1)	11	13			38
2	s.d f5,v6(r1)	12	14			39
2	s.d f7,v7(r1)	12	15			39
2	daddi r2,r2,-1	13	14		15	40
2	daddui r1,r1,8	13	15		16	40
2	bnez r2,loop	14	16			41

I primi 2 cicli sono eseguiti in _____ colpi di clock.