Domanda 10 (8 punti)

Si consideri il seguente frammento di codice C:

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

dove i vettori v1`[i], v2[i]` e `v3[i]` contengono numeri Floating Point (FP), sono lunghi 100 e sono stati salvati in precedenza nella memoria.

Si consideri un processore MIPS64 con le seguenti caratteristiche:

- L'unità di moltiplicazione FP è un'unità pipelined a 4 stadi
- L'unità aritmetica FP è un'unità pipelined a 2 stadi
- L'unità di divisione FP è un'unica unità con una latenza pari a 4 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.

Con riferimento al programma riportato nel seguito, scritto per l'architettura del processore MIPS64 descritta, ed utilizzando gli spazi a ciò appositamente destinati, si eseguano le seguenti operazioni:

1) Si calcoli il numero di colpi di clock richiesti per l'esecuzione dell'intero programma.

2) Si ottimizzi il programma utilizzando esclusivamente le tecniche di scheduling, register renaming e di delayed branch, in maniera tale che il programma esegua lo stesso calcolo. Si esegua il calcolo dei colpi di clock richiesti dal nuovo programma del punto 2, evidenziando il miglioramento delle prestazioni.

3) Si calcoli il numero di colpi di clock necessari all'esecuzione di 2 cicli del programma non ottimizzato proposto per un processore MIPS che implementa la strategia multiple-issue con scheduling dinamico e speculazione; si supponga che:
    - si possa eseguire l'issue di 2 istruzioni per colpo di clock
    - in presenza di un'istruzione di salto, venga eseguita una sola issue
    - si possa eseguire il commit di 2 istruzioni per colpo di clock
    - siano disponibili le seguenti unità funzionali indipendenti:
        i. unità Memory address
        ii. unità per operazioni intere (ALU)
        iii. unità per il calcolo dei salti
        iv. unità di moltiplicazione FP senza pipeline (latenza 4)
        v. unità di divisione FP senz a pipeline (latenza 4)
        vi. unità di somma e sottrazione FP no pipelined (latenza 2)
    - la previsione sui salti sia sempre corretta
    - le cache non producano mai situazioni di miss
    - siano disponibili due CDB (Common Data Bus).

Punto 1)

```
; ********************* WinMIPS64       ************************
;for (i = 1; i <= 100; i++){
;         v4[i] = v1[i]*v2[i];
;         v5[i] = v1[i]+v2[i]+ (v1[i]*v3[i]);
;}
                  .data
V1:       .double  “100 valori”
V2:       .double “100 valori”
V3:       .double  “100 valori”
V4:       .double “100 zeri”
V5:       .double “100 zeri”

                  .text

main:             daddui r1,r0,0
                  daddui r2,r0,100
loop:             l.d      f1,v1(r1)
                  l.d      f2,v2(r1)
                  l.d      f3,v3(r1)

                  mul.d f4,f1,f2
                  s.d f4,v4(r1)
                  add.d f4,f1,f2
                  mul.d f2,f1,f3
                  add.d f1,f4,f2
                  s.d f1,v5(r1)
                  daddui r1,r1,8
                  daddi r2,r2,-1
                  bnez r2,loop
                  halt
                                  total
```

Commenti	Colpi di clock
r1? puntatore	
r2? 100	
f1? v1[i]	
f2? v2[i]	
f3? v3[i]	
f4 ? v1[i]*v2[i] v4[i] ? f4	
f4 ? v1[i]+v2[i]	
f2 ? v1[i]*v3[i]	
f1 ? v1[i]+v2[i]+v1[i]*v3[i]	
v5[i] ? f1	
r1 ? r1 + 8	
r2 ? r2 - 1	

Punto 2)

Punto 3)

# iterazione		Issue	EXE	MEM	CDB x2	COMMIT x2
1	l.d f1,v1(r1)					
1	l.d f2,v2(r1)					
1	l.d f3,v3(r1)					
1	mul.d f4,f1,f2					
1	s.d f4,v4(r1)					
1	add.d f4,f1,f2					
1	mul.d f2,f1,f3					
1	add.d f1,f4,f2					
1	s.d f1,v5(r1)					
1	daddui r1,r1,8					
1	daddi r2,r2,-1					
1	bnez r2,loop					
2	l.d f1,v1(r1)					
2	l.d f2,v2(r1)					
2	l.d f3,v3(r1)					
2	mul.d f4,f1,f2					
2	s.d f4,v4(r1)					
2	add.d f4,f1,f2					
2	mul.d f2,f1,f3					
2	add.d f1,f4,f2					
2	s.d f1,v5(r1)					
2	daddui r1,r1,8					
2	daddi r2,r2,-1					
2	bnez r2,loop					

	unit	latency
m	Memory address unit	1
x	FP Multiplier	4
a	FP Adder	2
i	integer ALU	1
j	jump unit	1

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