

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] = v3[i]*v4[i];  
    v6[i] = v4[i]+v5[i];  
}
```

dove i vettori `v1[]`, `v2[]` (`v2[]` non è mai 0) e `v3[]` contengono numeri Floating Point (FP), sono lunghi 100 e sono stati salvati in precedenza nella memoria. Inoltre sono stati allocati in memoria i vettori vuoti `v4[]`, `v5[]` e `v6[]`.

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 4 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.
- 2) 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, 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 no pipelined (latenza 4)
 - v. unità di divisione FP no pipelined (latenza 10)
 - 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 = 0; i < 100; i++) {
;     v4[i] = v1[i]/v2[i];
;     v5[i] = v3[i]*v4[i];
;     v6[i] = v4[i]+v5[i];
;}
;
```

```
    .data
V1: .double "100 valori"
V2: .double "100 valori"
V3: .double "100 valori"
V4: .double "100 zeri"
V5: .double "100 zeri"
V6: .double "100 zeri"
```

```
    .text
```

```
main: daddui r1,r0,0
      daddui r2,r0,100
loop: l.d    f1,v1(r1)
      l.d    f2,v2(r1)
      div.d  f4,f1,f2
      s.d    f4,v4(r1)
      l.d    f3,v3(r1)
      mul.d  f5,f3,f4
      s.d    f5,v5(r1)
      daddi  r2,r2,-1
      add.d  f6,f4,f5
      s.d    f6,v6(r1)
      daddui r1,r1,8
      bnez   r2,loop
      halt
```

total

Commenti	Colpi di clock
r1? puntatore	
r2? 100	
f1? v1[i]	
f2? v2[i]	
f4 ? v1[i]/v2[i]	
v4[i] ? f3	
f3? v3[i]	
f5 ? v3[i]*v4[i]	
v5[i] ? f5	
r2 ? r2 - 1	
f6 ? v4[i]+v5[i]	
v6[i] ? f6	
r1 ? r1 + 8	