

Nome, Cognome, Matricola

Domanda 10 (8 punti)

Si consideri il seguente frammento di codice C:

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

dove i vettori `v1[]`, `v2[]`, `v3[]`, `v4[]`, `v5[]` e `v6[]` contengono numeri Floating Point (FP), sono lunghi 100 e sono stati salvati in precedenza nella memoria; `v3[]` e `v6[]` non sono mai 0. Inoltre è stato allocato in memoria il vettore vuoto `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'unico blocco con una latenza pari 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 8 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++) {
;     v7[i] = (v1[i]+v2[i])/v3[i] + (v4[i]*v5[i])/v6[i];
; }
;

```

| | Commenti | Colpi di clock |
|--------------------------|----------------------------|----------------|
| .data | | |
| V1: .double "100 valori" | | |
| V2: .double "100 valori" | | |
| ... | | |
| V6: .double "100 valori" | | |
| V7: .double "100 zeri" | | |
| .text | | |
| | | |
| main: daddui r1,r0,0 | r1 <= puntatore | |
| daddui r2,r0,100 | r2 <= 100 | |
| loop: l.d f1,v1(r1) | f1 <= v1[i] | |
| l.d f2,v2(r1) | f2 <= v2[i] | |
| add.d f7,f1,f2 | f7 <= v1[i]+v2[i] | |
| l.d f3,v3(r1) | f3 <= v3[i] | |
| div.d f8,f7,f3 | f8 <= (v1[i]+v2[i])/v3[i] | |
| l.d f4,v4(r1) | f4 <= v4[i] | |
| l.d f5,v5(r1) | f5 <= v5[i] | |
| mul.d f9,f4,f5 | f9 <= v4[i]*v5[i] | |
| l.d f6,v6(r1) | f6 <= v6[i] | |
| div.d f10,f9,f6 | f10 <= (v4[i]*v5[i])/v6[i] | |
| add.d f11,f8,f10 | f11 <= f8+f10 | |
| s.d f11,v7(r1) | | |
| daddui r1,r1,8 | r1 <= r1 + 8 | |
| daddi r2,r2,-1 | r2 <= r2 - 1 | |
| bnez r2,loop | | |
| halt | | |
| total | | |

- 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
- si 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 no pipelined (latenza 8)
 - v. 1 unità di divisione FP no pipelined (latenza 8)
 - 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 | l.d f2,v2(r1) | | | | | |
| 1 | add.d f7,f1,f2 | | | | | |
| 1 | l.d f3,v3(r1) | | | | | |
| 1 | div.d f8,f7,f3 | | | | | |
| 1 | l.d f4,v4(r1) | | | | | |
| 1 | l.d f5,v5(r1) | | | | | |
| 1 | mul.d f9,f4,f5 | | | | | |
| 1 | l.d f6,v6(r1) | | | | | |
| 1 | div.d f10,f9,f6 | | | | | |
| 1 | add.d f11,f8,f10 | | | | | |
| 1 | s.d f11,v7(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 | add.d f7,f1,f2 | | | | | |
| 2 | l.d f3,v3(r1) | | | | | |
| 2 | div.d f8,f7,f3 | | | | | |
| 2 | l.d f4,v4(r1) | | | | | |
| 2 | l.d f5,v5(r1) | | | | | |
| 2 | mul.d f9,f4,f5 | | | | | |
| 2 | l.d f6,v6(r1) | | | | | |
| 2 | div.d f10,f9,f6 | | | | | |
| 2 | add.d f11,f8,f10 | | | | | |
| 2 | s.d f11,v7(r1) | | | | | |
| 2 | daddui r1,r1,8 | | | | | |
| 2 | daddi r2,r2,-1 | | | | | |
| 2 | bnez r2,loop | | | | | |

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