**Domanda 1**

Considerando il processore MIPS64 e l’architettura descritta in seguito:

|  |  |  |
| --- | --- | --- |
| * + Integer ALU: 1 clock cycle   + Data memory: 1 clock cycle   + FP multiplier unit: pipelined 6 stages | * + FP divider unit: not pipelined unit that requires 7 clock cycles   + FP arithmetic unit: pipelined 4 stages   + branch delay slot: 1 clock cycle, and the branch delay slot disabled | * + forwarding enabled   + it is possible to complete instruction EXE stage in an out-of-order fashion. |

Usando il frammento di codice riportato, si calcoli il tempo di esecuzione dell’intero programma in colpi di clock e si completi la seguente tabella.

; for (i = 0; i < 100; i++) {

; v4[i] = (v1[i]+v2[i])/v3[i];

;}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| .data |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Clock  cycles |
| V1: .double “100 values” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| V2: .double “100 values” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| V3: .double “100 values”  …  V5: .double “100 zeros” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| V4: .double “100 values” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| V5: .double “100 values” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| .text |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| main: daddui r1,r0,0 | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |
| daddui r2,r0,100 |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| loop: l.d f3,v1(r1) |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f2,v2(r1) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f1,v3(r1) |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| add.d f4,f1,f2 |  |  |  |  |  | F | D | S | + | + | + | + | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5 |
| div.d f4,f4,f3 |  |  |  |  |  |  | F | D | s | s | s | s | / | / | / | / | / | / | / | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7 |
| s.d f4,v4(r1) |  |  |  |  |  |  |  | F | s | s | s | s | D | E | s | s | s | s | s | S | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | s | s | s | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | s | s | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 2 |
| Halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | N | N | N | N |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Total |  |  |  |  | 6+(100\*21) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2106 |

**Domanda 2**

Considerando il programma precedente, e in particolare la coppia di istruzioni:

add.d f4,f1,f2

div.d f4,f4,f3

che tipo di hazard crea l’utilizzo di f4 e come viene risolto? motivare la risposta.

la div vuole leggere f4 , ma deve aspettare che la add aggiorni il valore! Si crea quindi uno stallo di tipo RAW, read after write e lo si risolve facendo stallare la div finchè la add non finisce l’esecuzione. A questo punto entra in azione il meccanismo di forwarding quindi la div può iniziare senza dover aspettare la fase di MEM della add.d

**Domanda 3**

Considerando il programma precedente e l’architettura del processore superscalare descritto in seguito; completare la tabella relativa alle prime 3 iterazioni.

Processor architecture:

* + Issue 2 instructions per clock cycle
  + jump instructions require 1 issue
  + handle 2 instructions commit per clock cycle
  + timing facts for the following separate functional units:
    1. 1 Memory address 1 clock cycle
    2. 1 Integer ALU 1 clock cycle
    3. 1 Jump unit 1 clock cycle
    4. 1 FP multiplier unit, which is pipelined: 6 stages
    5. 1 FP divider unit, which is not pipelined: 7 clock cycles
    6. 1 FP Arithmetic unit, which is pipelined: 4 stages
  + Branch prediction is always correct
  + There are no cache misses
  + There are 2 CDB (Common Data Bus).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # iteration |  | Issue | EXE | MEM | CDB x2 | COMMIT x2 |
| 1 | l.d f3,v1(r1) | 1 | 2m | 3 | 4 | 5 |
| 1 | l.d f2,v2(r1) | 1 | 3m | 4 | 5 | 6 |
| 1 | l.d f1,v3(r1) | 2 | 4m | 5 | 6 | 7 |
| 1 | add.d f4,f1,f2 | 2 | 7a |  | 11 | 12 |
| 1 | div.d f4,f4,f3 | 3 | 12d |  | 19 | 20 |
| 1 | s.d f4,v4(r1) | 3 | 5m |  |  | 20 |
| 1 | daddui r1,r1,8 | 4 | 5i |  | 6 | 21 |
| 1 | daddi r2,r2,-1 | 4 | 6i |  | 7 | 21 |
| 1 | bnez r2,loop | 5 | 8j |  |  | 22 |
| 2 | l.d f3,v1(r1) | 6 | 7m | 8 | 9 | 22 |
| 2 | l.d f2,v2(r1) | 6 | 8m | 9 | 10 | 23 |
| 2 | l.d f1,v3(r1) | 7 | 9m | 10 | 11 | 23 |
| 2 | add.d f4,f1,f2 | 7 | 12a |  | 16 | 24 |
| 2 | div.d f4,f4,f3 | 8 | 19d |  | 26 | 27 |
| 2 | s.d f4,v4(r1) | 8 | 10m |  |  | 27 |
| 2 | daddui r1,r1,8 | 9 | 10i |  | 12 | 28 |
| 2 | daddi r2,r2,-1 | 9 | 12i |  | 13 | 28 |
| 2 | bnez r2,loop | 10 | 13j |  |  | 29 |
| 3 | l.d f3,v1(r1) | 11 | 12m | 13 | 14 | 29 |
| 3 | l.d f2,v2(r1) | 11 | 13m | 14 | 15 | 30 |
| 3 | l.d f1,v3(r1) | 12 | 14m | 15 | 16 | 30 |
| 3 | add.d f4,f1,f2 | 12 | 17i |  | 21 | 31 |
| 3 | div.d f4,f4,f3 | 13 | 26d |  | 33 | 34 |
| 3 | s.d f4,v4(r1) | 13 | 15m |  |  | 34 |
| 3 | daddui r1,r1,8 | 14 | 15i |  | 17 | 35 |
| 3 | daddi r2,r2,-1 | 14 | 17i |  | 18 | 35 |
| 3 | bnez r2,loop | 15 | 18j |  |  | 36 |

**Domanda 4**

Considerando il segmento di codice presentato nella tabella precedente, se assumiamo che ci sia un unico Common Data Bus, qual è la prima istruzione che dovrebbe stallare durante l’esecuzione del programma? motivare la risposta.

Il common data bus serve come bus centrale nell’architettura di tomasulo per portare i vari dati nelle reservation stations. In questo esercizio c’erano due CDB, quindi due dati contemporaneamente potevano essere disponibili sul CDB. Se ne eliminiamo uno, allora la prima istruzione che stallerà sarà daddui r1,r1,8, visto che essa proverà a mettere i suoi dati sul CDB al colpo di clock numero 8, ma esso sarà già occupato dalla load di f1.