**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 | s | D | s | s | s | / | / | / | / | / | / | / | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7 |
| s.d f4,v4(r1) |  |  |  |  |  |  |  |  | F | 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 | - | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Total |  |  |  |  | 6+21\*100 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 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.

è un HAZARD di tipo DATA e in particolare un RAW HAZARD che si risolve a livello HW tramite gli stalli e a livello SW tramite il rescheduling.

**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 | 2ea | 3 | 4 | 5 |
| 1 | l.d f2,v2(r1) | 1 | 3ea | 4 | 5 | 6 |
| 1 | l.d f1,v3(r1) | 2 | 4ea | 5 | 6 | 7 |
| 1 | add.d f4,f1,f2 | 2 | 7a-10a |  | 11 | 12 |
| 1 | div.d f4,f4,f3 | 3 | 12d-18d |  | 19 | 20 |
| 1 | s.d f4,v4(r1) | 3 | 5ea |  |  | 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 | 7ea | 8 | 9 | 22 |
| 2 | l.d f2,v2(r1) | 6 | 8ea | 9 | 10 | 23 |
| 2 | l.d f1,v3(r1) | 7 | 9ea | 10 | 11 | 23 |
| 2 | add.d f4,f1,f2 | 7 | 12a-15a |  | 16 | 24 |
| 2 | div.d f4,f4,f3 | 8 | 19d-25d |  | 26 | 27 |
| 2 | s.d f4,v4(r1) | 8 | 10ea |  |  | 27 |
| 2 | daddui r1,r1,8 | 9 | 10i |  | 12 | 28 |
| 2 | daddi r2,r2,-1 | 9 | 11i |  | 12 | 28 |
| 2 | bnez r2,loop | 10 | 13j |  |  | 29 |
| 3 | l.d f3,v1(r1) | 10 | 11ea | 12 | 13 | 29 |
| 3 | l.d f2,v2(r1) | 11 | 12ea | 13 | 14 | 30 |
| 3 | l.d f1,v3(r1) | 11 | 13ea | 14 | 15 | 30 |
| 3 | add.d f4,f1,f2 | 12 | 16a-19a |  | 20 | 31 |
| 3 | div.d f4,f4,f3 | 12 | 26d-32d |  | 33 | 34 |
| 3 | s.d f4,v4(r1) | 13 | 14ea |  |  | 34 |
| 3 | daddui r1,r1,8 | 13 | 14i |  | 15 | 35 |
| 3 | daddi r2,r2,-1 | 14 | 15i |  | 16 | 35 |
| 3 | bnez r2,loop | 15 | 17j |  |  | 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.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | daddui r1,r1,8 | 9 | 10i |  | 12 | 28 |

Nel CDB ci sono soltanto due posti per CC e in quest’istruzione si odveva scrivere il dato nel CDB al CC 11, ma era gia riampito dalle istruzioni

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | add.d f4,f1,f2 | 2 | 7a-10a |  | 11 | 12 |

E

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | l.d f1,v3(r1) | 7 | 9ea | 10 | 11 | 23 |