**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 6 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 f1,v1(r1) |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1\*100 |
| l.d f2,v2(r1) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | … |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1\*100 |
| div.d f3,f1,f2 |  |  |  |  | F | D | s | d | d | d | d | d | d | M | W |  |  |  |  |  |  |  |  |  |  | F | D | s | … |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7\*100 |
| l.d f1,v3(r1) |  |  |  |  |  | F | s | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | … |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0\*100 |
| mul.d f3,f3,f1 |  |  |  |  |  |  |  | F | D | s | s | s | s | m | m | m | m | m | m | M | W |  |  |  |  |  |  |  | … |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6\*100 |
| s.d f3,v4(r1) |  |  |  |  |  |  |  |  | F | s | s | s | s | D | E | s | s | s | s | s | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1\*100 |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | s | s | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1\*100 |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | s | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1\*100 |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 2\*100 |
| Halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | D | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1\*100 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Total |  |  |  |  |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2106 |

**Domanda 2**

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

div.d f3,f1,f2

l.d f1,v3(r1)

che tipo di hazard crea l’utilizzo di f1 e come viene risolto? motivare la risposta.

Crea un hazard WAR, perchè l.d esegue un Write di un registro dopo (After) che la div.d esegue un Read dello stesso registro.

Potrebbe causare problemi se l.d venisse eseguita prima di div.d, in quanto div avrebbe un valore errato in f1.

Può essere risolto dal compilatore con un register renaming, utilizzando un registro diverso (per esempio, f4) nella l.d e in tutte le istruzioni seguenti che usano f1.

Un’altra soluzione è utilizzare un’architettura che effettui il renaming già a livello hardware, come avviene per esempio nell’architettura di Tomasulo.

**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 -> (ma)
    2. 1 Integer ALU 1 clock cycle -> (ia)
    3. 1 Jump unit 1 clock cycle -> (j)
    4. 1 FP multiplier unit, which is pipelined: 6 stages -> (m)
    5. 1 FP divider unit, which is not pipelined: 6 clock cycles -> (d)
    6. 1 FP Arithmetic unit, which is pipelined: 4 stages -> (a)
  + 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 | #Instr |
| 1 | l.d f1,v1(r1) | 1 | 2 (ma) | 3 | 4 | 5 | 1 |
| 1 | l.d f2,v2(r1) | 1 | 3 (ma) | 4 | 5 | 6 | 2 |
| 1 | div.d f3,f1,f2 | 2 | 6-11 (d) |  | 12 | 13 | 3 |
| 1 | l.d f1,v3(r1) | 2 | 4 (ma) | 5 | 6 | 13 | 4 |
| 1 | mul.d f3,f3,f1 | 3 | 13-18 (m) |  | 19 | 20 | 5 |
| 1 | s.d f3,v4(r1) | 3 | 5 (ma) |  |  | 20 | 6 |
| 1 | daddui r1,r1,8 | 4 | 5 (ia) |  | 6 | 21 | 7 |
| 1 | daddi r2,r2,-1 | 4 | 6 (ia) |  | 7 | 21 | 8 |
| 1 | bnez r2,loop | 5 | 8 (j) |  |  | 22 | 9 |
| 2 | l.d f1,v1(r1) | 6 | 7 (ma) | 8 | 9 | 22 | 10 |
| 2 | l.d f2,v2(r1) | 6 | 8 (ma) | 9 | 10 | 23 | 11 |
| 2 | div.d f3,f1,f2 | 7 | 12-17 (d) |  | 18 | 23 | 12 |
| 2 | l.d f1,v3(r1) | 7 | 9 (ma) | 10 | 11 | 24 | 13 |
| 2 | mul.d f3,f3,f1 | 8 | 19-24 (m) |  | 25 | 26 | 14 |
| 2 | s.d f3,v4(r1) | 8 | 10 (ma) |  |  | 26 | 15 |
| 2 | daddui r1,r1,8 | 9 | 10 (ia) |  | 11 | 27 | 16 |
| 2 | daddi r2,r2,-1 | 9 | 11 (ia) |  | 12 | 27 | 17 |
| 2 | bnez r2,loop | 10 | 13 (j) |  |  | 28 | 18 |
| 3 | l.d f1,v1(r1) | 11 | 12 (ma) | 13 | 14 | 28 | 19 |
| 3 | l.d f2,v2(r1) | 11 | 13 (ma) | 14 | 15 | 29 | 20 |
| 3 | div.d f3,f1,f2 | 12 | 18-23 (d) |  | 24 | 29 | 21 |
| 3 | l.d f1,v3(r1) | 12 | 14 (ma) | 15 | 16 | 30 | 22 |
| 3 | mul.d f3,f3,f1 | 13 | 25-30 (m) |  | 31 | 32 | 23 |
| 3 | s.d f3,v4(r1) | 13 | 15 (ma) |  |  | 32 | 24 |
| 3 | daddui r1,r1,8 | 14 | 15 (ia) |  | 16 | 33 | 25 |
| 3 | daddi r2,r2,-1 | 14 | 16 (ia) |  | 17 | 33 | 26 |
| 3 | bnez r2,loop | 15 | 18 (j) |  |  | 34 | 27 |

**Domanda 4**

Considerando il segmento di codice presentato nella tabella precedente, se assumiamo che il ROB ha una dimensione di 16 elementi e alla partenza del codice in tabella è vuoto, qual è la prima istruzione che dovrebbe stallare durante l’esecuzione del programma? motivare la risposta.

Le prime 16 istruzioni sono inserite nel ROB senza problemi, perché il ROB è vuoto.

Quando la 16° è inserita nel ROB, il sistema è al CC 9: a quel punto, le prime 2 istruzioni sono già committate, perciò hanno lasciato libere due entries del ROB. Si si può quindi procedere con la issue di altre due istruzioni.

Dopo la 18° issue, il sistema è al CC 10: rispetto a prima, non è stato effettuato alcun commit aggiuntivo. Questo significa che il ROB è pieno, perciò non si può fare la issue di altre istruzioni.

Quindi la prima istruzione che stalla è la 19°, ovvero la l.d f1, v1(r1)della terza iterazione