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 arithmetic unit: pipelined 2 stages   + FP divider unit: not pipelined unit that requires 9 clock cycles   + 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f2,v2(r1) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f3,v3(r1) |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| mul.d f4,f1,f2 |  |  |  |  |  | F | D | \* | \* | \* | \* | \* | \* | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |
| div.d f4,f4,f3 |  |  |  |  |  |  | F | D | s | s | s | s | s | / | / | / | / | / | / | / | / | / | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 9 |
| s.d f4,v4(r1) |  |  |  |  |  |  |  | F | s | s | s | s | s | D | E | s | s | s | s | s | s | s | S | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | s | s | s | s | S | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | s | s | s | S | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| Halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | - | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Total |  |  |  |  | 6+23\*100 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2306 |

**Domanda 2**

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

mul.d f4,f1,f2

div.d f4,f4,f3

come viene attivato e qual è il cammino di forwarding che partecipa alla loro esecuzione? Motivare la risposta

**Viene attivato prendendo i risultati dell’operazione direttamente dai registri e non dalla memoria, in questo caso la div inizia la propria fase di execute quando quella della mul è terminata (RAW HAZARD) così da risparmiare dei CC e non aspettare la fase di write back per ottenere il dato in f4.**

**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: 9 clock cycles
    6. 1 FP Arithmetic unit, which is pipelined: 2 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 f1,v1(r1) | 1 | 2ea | 3 | 4 | 5 |
| 1 | l.d f2,v2(r1) | 1 | 3ea | 4 | 5 | 6 |
| 1 | l.d f3,v3(r1) | 2 | 4ea | 5 | 6 | 7 |
| 1 | mul.d f4,f1,f2 | 2 | 6m-11m |  | 12 | 13 |
| 1 | div.d f4,f4,f3 | 3 | 13d-21d |  | 22 | 23 |
| 1 | s.d f4,v4(r1) | 3 | 5ea |  |  | 23 |
| 1 | daddi r2,r2,-1 | 4 | 5i |  | 6 | 24 |
| 1 | daddui r1,r1,8 | 4 | 6i |  | 7 | 24 |
| 1 | bnez r2,loop | 5 | 7j |  |  | 25 |
| 2 | l.d f1,v1(r1) | 6 | 7ea | 8 | 9 | 25 |
| 2 | l.d f2,v2(r1) | 6 | 8ea | 9 | 10 | 26 |
| 2 | l.d f3,v3(r1) | 7 | 9ea | 10 | 11 | 26 |
| 2 | mul.d f4,f1,f2 | 7 | 11m-16m |  | 17 | 27 |
| 2 | div.d f4,f4,f3 | 8 | 22d-30d |  | 31 | 32 |
| 2 | s.d f4,v4(r1) | 8 | 10ea |  |  | 32 |
| 2 | daddi r2,r2,-1 | 9 | 10i |  | 11 | 33 |
| 2 | daddui r1,r1,8 | 9 | 11i |  | 12 | 33 |
| 2 | bnez r2,loop | 10 | 12j |  |  | 34 |
| 3 | l.d f1,v1(r1) | 11 | 12ea | 13 | 14 | 34 |
| 3 | l.d f2,v2(r1) | 11 | 13ea | 14 | 15 | 35 |
| 3 | l.d f3,v3(r1) | 12 | 14ea | 15 | 16 | 35 |
| 3 | mul.d f4,f1,f2 | 12 | 16m-21m |  | 22 | 36 |
| 3 | div.d f4,f4,f3 | 13 | 31d-39d |  | 40 | 41 |
| 3 | s.d f4,v4(r1) | 13 | 15ea |  |  | 41 |
| 3 | daddi r2,r2,-1 | 14 | 15i |  | 16 | 42 |
| 3 | daddui r1,r1,8 | 14 | 16i |  | 17 | 42 |
| 3 | bnez r2,loop | 15 | 17j |  |  | 43 |

**Domanda 4**

Considerando il segmento di codice presentato nella tabella precedente, se assumiamo che il ROB ha una dimensione di 8 elementi, qual è la prima istruzione che dovrebbe stallare durante la esecuzione del programma? motivare la risposta.

La nona istruzione fa la issue al CC 5 e a quel punto il rob ha liberato la prima istruzione, quindi non stalla. Lo stesso vale per la decima istruzione in quanto il rob libera un’altra posizione quando essa fa la issue (CC 6), quindi a stallare è la 11esima che fa issue al CC 6, ma la rob libera un altro posto al CC 7:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | l.d f2,v2(r1) | 6 | 8ea | 9 | 10 | 26 |