**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: not pipelined unit that requires 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f2,v2(r1) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| mul.d f4,f1,f2 |  |  |  |  | F | D | s | \* | \* | \* | \* | \* | \* | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7 |
| l.d f3,v3(r1) |  |  |  |  |  | F | s | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 |
| mul.d f4,f4,f3 |  |  |  |  |  |  |  | F | D | s | s | s | s | \* | \* | \* | \* | \* | \* | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |
| s.d f4,v4(r1) |  |  |  |  |  |  |  |  | F | s | s | s | s | D | E | s | s | s | s | S | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | s | S | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | S | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| Halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | - | - | - | - |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Total |  |  |  |  | 6+20\*100 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2006 |

**Domanda 2**

Considerando il programma precedente, e in particolare la istruzione:

mul.d f4,f4,f3

quali potrebbero essere i cammini di forwarding che potrebbero partecipare alla sua esecuzione? Perché dovrebbero essere attivati? motivare la risposta.

Il cammino che si attiva con la load precedente su f2, il quale ci risparmia un CC, in quanto leggiamo il valore dell’operando prima che termini la fase di write back.

Inoltre è presente un’altra mul successiva che usa il dato calcolato in f4, in questo caso si attiva il cammino di forwardin che permette alla seconda mul di ottenere il dato dopo la fase di EXECUTE della prima mul, senza aspettare write back, così da risolvere il RAW HAZARD.

Il cammino che prende il risultato memorizzato nel registro f4 dall’istruzione mul.d f4,f1,f2 e quello che prende il risultato memorizzato nel registro f3 dall’istruzione l.d f3,v3(r1).

Essi sono fondamentali perché ci permettono di usare il risultato di un’istruzione accedendo al registro destinazione prima della fase di WRITE BACK, in questo caso infatti, l’esecuzione (fase E) di mul.d f4,f4,f3 comincia prima della fase di W di mul.d f4,f1,f2 e questo ci fa guadagnare dei colpi di clock.

**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 not pipelined: 6 stages
    5. 1 FP divider unit, which is not pipelined: 6 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 f1,v1(r1) | 1 | 2ea | 3 | 4 | 5 |
| 1 | l.d f2,v2(r1) | 1 | 3ea | 4 | 5 | 6 |
| 1 | mul.d f4,f1,f2 | 2 | 6m-11m |  | 12 | 13 |
| 1 | l.d f3,v3(r1) | 2 | 4ea | 5 | 6 | 13 |
| 1 | mul.d f4,f4,f3 | 3 | 13m-18m |  | 19 | 20 |
| 1 | s.d f4,v4(r1) | 3 | 5ea |  |  | 20 |
| 1 | daddi r2,r2,-1 | 4 | 5i |  | 6 | 21 |
| 1 | daddui r1,r1,8 | 4 | 6i |  | 7 | 21 |
| 1 | bnez r2,loop | 5 | 7j |  |  | 22 |
| 2 | l.d f1,v1(r1) | 6 | 7ea | 8 | 9 | 22 |
| 2 | l.d f2,v2(r1) | 6 | 8ea | 9 | 10 | 23 |
| 2 | mul.d f4,f1,f2 | 7 | 19m-24m |  | 25 | 26 |
| 2 | l.d f3,v3(r1) | 7 | 9ea | 10 | 11 | 26 |
| 2 | mul.d f4,f4,f3 | 8 | 26m-31m |  | 32 | 33 |
| 2 | s.d f4,v4(r1) | 8 | 10ea |  |  | 33 |
| 2 | daddi r2,r2,-1 | 9 | 10i |  | 11 | 34 |
| 2 | daddui r1,r1,8 | 9 | 11i |  | 12 | 34 |
| 2 | bnez r2,loop | 10 | 12j |  |  | 35 |
| 3 | l.d f1,v1(r1) | 11 | 12ea | 13 | 14 | 35 |
| 3 | l.d f2,v2(r1) | 11 | 13ea | 14 | 15 | 36 |
| 3 | mul.d f4,f1,f2 | 12 | 32m-37m |  | 38 | 39 |
| 3 | l.d f3,v3(r1) | 12 | 14ea | 15 | 16 | 39 |
| 3 | mul.d f4,f4,f3 | 13 | 39m-44m |  | 45 | 46 |
| 3 | s.d f4,v4(r1) | 13 | 15ea |  |  | 46 |
| 3 | daddi r2,r2,-1 | 14 | 15i |  | 16 | 47 |
| 3 | daddui r1,r1,8 | 14 | 16i |  | 17 | 47 |
| 3 | bnez r2,loop | 15 | 17j |  |  | 48 |

**Domanda 4**

Considerando il segmento di codice presentato nella tabella precedente, se assumessimo che il ROB abbia una dimensione di 16 elementi, quale sarebbe la prima istruzione che dovrebbe stallare durante la esecuzione del programma? motivare la risposta.

Le prime 16 entrano nel rob, siamo arrivati alla 17esima che fa issue al CC 9, nel frattempo le prime 2 istruzioni nel rob hanno fatto commit, quindi entrano nel rob sia la 17esima che la 18esima, siamo arrivati alla 19esima che fa issue al CC 9, mentre la prossima istruzione a fare commit nel rob lo fa al CC 13 dunque la 19esima istruzione stallerà, ed è la seguente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| l.d f1,v1(r1) | 11 | 12ea | 13 | 14 | 35 |

Le prime 16 istruzioni entrano nel ROB, la 17esima è:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | daddui r1,r1,8 | 9 | 11i |  | 13 | 34 |

La cui issue è al CC 9, a questo punto però le prime due istruzioni del ROB hanno fatto commit e liberano 2 posti, perciò entrano la 17esima e la 18esima istruzione senza stallare. A questo punto siamo al CC 11 per la issue della 19esima istruzione, ma la prossima posizione del ROB si libera al CC 13, perciò la prima istruzione a stallare è proprio la 19esima, ovvero:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | l.d f1,v1(r1) | 11 | 14ea | 15 | 16 | 35 |