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

|  |  |  |
| --- | --- | --- |
| * + Integer ALU: 1 clock cycle   + Data memory: 1 clock cycle   + FP multiplier unit: pipelined 7 stages | * + FP arithmetic unit: pipelined 3 stages   + FP divider unit: not pipelined unit that requires 7 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++) {

; v5[i] = v1[i]/v2[i];

; v6[i] = (v1[i]\*v2[i])/v3[i] + v4[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” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| V6: .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 |
| div.d f5,f1,f2 |  |  |  |  | F | D | s | / | / | / | / | / | / | / | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 8 |
| s.d f5,v5(r1) |  |  |  |  |  | F | s | D | E | s | s | s | s | s | S | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f3,v3(r1) |  |  |  |  |  |  |  | F | D | s | s | s | s | s | S | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f4,v4(r1) |  |  |  |  |  |  |  |  | F | s | s | s | s | s | S | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| mul.d f6,f1,f2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | \* | \* | \* | \* | \* | \* | \* | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 7 |
| div.d f7,f6,f3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | s | s | s | / | / | / | / | / | / | / | M | W |  |  |  |  |  |  |  |  |  |  | 7 |
| add.d f7,f7,f4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | s | s | D | s | s | s | s | s | s | + | + | + | M | W |  |  |  |  |  |  |  | 3 |
| s.d f7,v6(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | s | s | D | E | s | S | M | W |  |  |  |  |  |  | 1 |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | E | M | W |  |  |  |  |  | 1 |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | D | E | M | W |  |  |  |  | 1 |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  | 1 |
| Halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | - | - | - | - |  |  | 1 |
| Total |  |  |  |  | 6+100\*35 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 3506 |

**Domanda 2**

Si consideri una BHT di 1K elementi con contatori di saturazione a 2 bit. Usando il frammento di codice proposto, si calcoli lo stato finale della BHT al completamento dell’esecuzione del programma proposto. Si calcoli inoltre, la *misprediction rate* complessiva. Lo stato iniziale della BPU è indicato nella tabella.

Assunzioni generali:

* R10 è il registro di controllo del loop ed è inizializzato a 100
* R3 e R7 sono registro di riferimento con valore 10 (R3 = R7 = 10)
* R2 e R6 sono i registri in ingresso
  + R2 riceve un valore sempre maggiore a 10 (R2 > 10)
  + R6 riceve un valore sempre minore a 10 (R6 < 10)
* Gli altri registri sono inizializzati a 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Address** | **Instruction** | | **BHT (2-bit)** | **Prediction** | **misP. counter** |
| **0x0000** | **L0:** | **…** |  |  |  |
| **…** | ; | ***Reading input values*** |  |  |  |
| **0x0010** |  | **SLT R1, R2, R3**  **[Sempre 0, R2 > R3]** | 2 | T |  |
| **0x0014** |  | **BEQZ R1, L1**  [SEMPRE TAKEN] | 2-3-…-3 | T-T-…-T | 0-0…-0 |
| **0x0018** |  | **DADDI R11, R0, 10**  [R11 = 10] | 2 | T |  |
| **0x001C** | **L1:** | **SLT R4, R6, R7**  [SEMPRE 1, R6 < R7] | 2 | T |  |
| **0x0020** |  | **BEQZ R4, L2**  [SEMPRE NOT TAKEN] | 2-1-0…-0 | T-NT-NT-…-NT | 1-1…-1 |
| **0x0024** |  | **DADDI R12, R0, 10**  [R12 = 10] | 2 | T |  |
| **0x0028** | **L2:** | **SLT R5, R2, R6**  [sempre 0, R2 > R6] | 2 | T |  |
| **0x002C** |  | **BEQZ R5, L3**  [SEMPRE TAKEN] | 2-3-…-3 | T-T-…-T | 0-0-…-0 |
| **0x0030** |  | **DADDI R13, R0, 10**  [R13 = 10] | 2 | T |  |
| **0x0038** | **L3:** | **DADD R14, R13, R12**  [R14 = 20] | 2 | T |  |
| **0x003C** |  | **DADD R14, R14, R11**  [R14 = 30] | 2 | T |  |
| **0x0040** |  | **SLT R5, R6, R14**  [sempre 1, R6 < R14] | 2 | T |  |
| **0x0044** |  | **BEQZ R5, L4**  [SEMPRE NOT TAKEN] | 2-1-0…0 | T-NT-…-NT | 1-1-…-1 |
| **0x0048** |  | **DADDI R15, R0, 10** | 2 | T |  |
| **0x004C** | **L4:** | **…** | 2 | T |  |
| **0x0050** |  | **DADDI R10, R10, #-1** | 2 | T |  |
| **0x0054** |  | **BNEZ R10, L0**  **[SEMPRE TAKEN APPARTE L’ULTIMO]** | 2-3…-3-2 | T-T…-NT | 0…-0-1 |
| **0x0058** |  |  |  |  |  |
|  |  |  |  |  |  |

Note:

SLT R1,R2,R3 ;IF (R2 < R3) R1 🡨 1

;ELSE R1 🡨 0

**misP=3/500=0.6%**