Considering the MIPS64 architecture presented in the following:

|  |  |  |
| --- | --- | --- |
| * + Integer ALU: 1 clock cycle   + Data memory: 1 clock cycle   + FP multiplier unit: pipelined 6 stages | * + FP arithmetic unit: pipelined 3 stages   + FP divider unit: not pipelined unit that requires 6 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. |

* Using the following code fragment, show the timing of the presented loop-based program and compute how many cycles does this program take to execute?

for (i = 0; i < 100; i++) {

v7[i] = (v1[i]\*v2[i]+v3[i]/v4[i]+v5[i]/v6[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 |
| l.d f3,v3(r1) |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f4,v4(r1) |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f5,v5(r1) |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| l.d f6,v6(r1) |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| mul.d f7,f1,f2 |  |  |  |  |  |  |  |  | F | D | \* | \* | \* | \* | \* | \* | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |
| div.d f8,f3,f4 |  |  |  |  |  |  |  |  |  | F | D | / | / | / | / | / | / | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| div.d f9,f5,f6 |  |  |  |  |  |  |  |  |  |  | F | D | S | S | S | S | S | / | / | / | / | / | / | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |
| add.d f10,f8,f7 |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | s | D | + | + | + | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0 |
| add.d f10,f9,f10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | s | + | + | + | M | W |  |  |  |  |  |  |  |  |  |  |  | 3 |
| s.d f10,v7(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | 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\*27 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 2706 |

Considering a (2,2) correlating predictor of 1K entries; and assuming that the MIPS-like processor executes the following code fragment, **calculate the misprediction rate** in the presented case.

The BPU initial state is indicated in the table.

General assumptions:

* R10 is the main loop control register and is initialized to 0
* R3 and R7 are the reference values set to 5
* R2 and R6 are input registers
  + R2 input values are 3 in the even iterations (R10 = 0, 2, 4, 6,…), and 7 in the odd ones (R10 = 1,3,5,7,…).
  + R6 input values are always 0 < R6 < 5
* R2 SLT R1,R2,R3      ;IF (R2 < R3) R1  1

                        ;ELSE R1  0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **Instruction** | | **2-bit predictors** | | | | **2-bit shift register** | **misP. counter** |
| **00** | **01** | **10** | **11** |
| **0x0000** | **L0:** | **…** | **0** | **0** | **0** | **0** | **00** |  |
| **…** | ; | ***Reading input values*** | **0** | **0** | **0** | **0** |  |  |
| **0x0010** |  | **SLT R1, R2, R3** | **0** | **0** | **0** | **0** |  |  |
| **0x0014** |  | **BNEZ R1, L1** | **0-1-0-0** | **0-0-0-0** | **0-0-0** | **0-0-0-0** | 01-10-10-10-10-10-10-10-00-00-00-00 | 1-1-1-1-1-1-1-1-1-1-1-1 |
| **0x0018** |  | **DADDI R12, R0, 10** | **0** | **0** | **0** | **0** |  |  |
| **0x001C** | **L1:** | **SLT R4, R6, R7** | **0** | **0** | **0** | **0** |  |  |
| **0x0020** |  | **BNEZ R4, L2** | **0-1-2** | **0-1** | **0-1-2-3** | **0** | 11-01-01-00-00-00-00-00-01-01-00 | 1-2-3-3-3-3-3-3-4-5-5-…- |
| **0x0024** |  | **DADDI R16, R0, 10** | **0** | **0** | **0** | **0** |  |  |
| **0x0028** | **L2:** | **SLT R3, R2, R7** | **0** | **0** | **0** | **0** |  |  |
| **0x002C** |  | **BEQZ R3, L3** | **0-1-0-1-0-1-0** | **0-1-0-0-1** | **0** | **0-0** | 10-11-10-01-00-01-00-01-10-11-01 | 1-1-1-2-2-3-3-4-4-5-6-…-49 |
| **0x0030** |  | **…** | **0** | **0** | **0** | **0** |  |  |
| **0x0038** | **L3:** | **…** | **0** | **0** | **0** | **0** |  |  |
| **0x003c** |  | **DADDI R10, R10, #1** | **0** | **0** | **0** | **0** |  |  |
| **0x0040** |  | **DADDI R11, R10, #-99** | **0** | **0** | **0** | **0** |  |  |
| **0x0044** |  | **BNEZ R11, L0** | **0-1-2** | **0-1-2** | **0-1-2** | **0-1-2** | 01-11-01-11-01-11-01-10-00-10 | 1-2-3-4-5-6-7-7-7-8-…-9 |
| **0x0048** |  | **…** | **0** | **0** | **0** | **0** |  |  |

**MISPREDICTION RATE**

**Number of mispredictions / total number of decisions**

**\_\_\_\_\_\_\_\_\_\_\_\_64\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_99\*4=396\_\_\_\_\_\_\_\_\_\_\_\_\_\_-> 64/396 =0.1616**