Considering the MIPS64 architecture presented in the following:

|  |  |  |
| --- | --- | --- |
| * + Integer ALU: 1 clock cycle   + Data memory: 1 clock cycle   + FP multiplier unit: pipelined 8 stages | * + FP arithmetic unit: pipelined 4 stages   + FP divider unit: not pipelined unit that requires 10 clock cycles   + branch delay slot: 1 clock cycle, and the branch delay slot is not enable | * + forwarding is enabled   + it is possible to complete instruction EXE stage in an out-of-order fashion. |

* and 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++) {

v3[i] = v4[i]/v3[i];

v6[i] = (v1[i]\*v2[i])+(v3[i]\*v4[i]);

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| .data |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Clock  cycles |
| V1: .double “100 values” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| V2: .double “100 values” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| …  V5: .double “100 zeros” |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 |
| div.d f3,f4,f3 |  |  |  |  |  |  | F | D | s | : | : | : | : | : | : | : | : | : | ; | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 11 |
| s.d f3,v3(r1) |  |  |  |  |  |  |  | F | s | D | E | s | s | s | s | s | s | s | s | s | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| mul.d f7,f1,f2 |  |  |  |  |  |  |  |  |  | F | D | x | x | x | x | x | x | x | x | s | s | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| mul.d f6,f3,f4 |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | s | s | s | s | x | x | x | x | x | x | x | x | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 6 |
| add.d f1,f7,f6 |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | s | s | s | s | D | s | s | s | s | s | s | s | + | + | + | + | M | W |  |  |  |  |  |  |  |  |  |  | 4 |
| s.d f1,v6(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | **s** | s | s | s | s | s | s | D | E | s | s | s | M | W |  |  |  |  |  |  |  |  |  | 1 |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | s | s | s | E | M | W |  |  |  |  |  |  |  |  | 1 |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | s | s | D | E | M | W |  |  |  |  |  |  |  | 1 |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | s | D | E | M | W |  |  |  |  |  | 2 |
| halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  | 1 |
| total |  |  |  |  | 6+33\*10=3306 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |

Considering the same loop-based program, and assuming the following processor architecture for a superscalar MIPS64 processor implemented with multiple-issue and speculation:

* + 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: 12 stages
    5. 1 FP Arithmetic unit, which is pipelined: 6 stages
    6. 1 FP divider unit, which is not pipelined: 14 clock cycles
  + Branch prediction is always correct
  + There are no cache misses
  + There are 2 CDB (Common Data Bus).
* Complete the table reported below showing the processor behavior for the 2 initial iterations.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # iteration |  | Issue | EXE | MEM | CDB x2 | COMMIT x2 |
| 1 | l.d f1,v1(r1) |  |  |  |  |  |
| 1 | l.d f2,v2(r1) |  |  |  |  |  |
| 1 | l.d f3,v3(r1) |  |  |  |  |  |
| 1 | l.d f4,v4(r1) |  |  |  |  |  |
| 1 | div.d f3,f4,f3 |  |  |  |  |  |
| 1 | s.d f3,v3(r1) |  |  |  |  |  |
| 1 | mul.d f7,f1,f2 |  |  |  |  |  |
| 1 | mul.d f6,f3,f4 |  |  |  |  |  |
| 1 | add.d f1,f7,f6 |  |  |  |  |  |
| 1 | s.d f1,v6(r1) |  |  |  |  |  |
| 1 | daddui r1,r1,8 |  |  |  |  |  |
| 1 | daddi r2,r2,-1 |  |  |  |  |  |
| 1 | bnez r2,loop |  |  |  |  |  |
| 2 | l.d f1,v1(r1) |  |  |  |  |  |
| 2 | l.d f2,v2(r1) |  |  |  |  |  |
| 2 | l.d f3,v3(r1) |  |  |  |  |  |
| 2 | l.d f4,v4(r1) |  |  |  |  |  |
| 2 | div.d f3,f4,f3 |  |  |  |  |  |
| 2 | s.d f3,v3(r1) |  |  |  |  |  |
| 2 | mul.d f7,f1,f2 |  |  |  |  |  |
| 2 | mul.d f6,f3,f4 |  |  |  |  |  |
| 2 | add.d f1,f7,f6 |  |  |  |  |  |
| 2 | s.d f1,v6(r1) |  |  |  |  |  |
| 2 | daddui r1,r1,8 |  |  |  |  |  |
| 2 | daddi r2,r2,-1 |  |  |  |  |  |
| 2 | bnez r2,loop |  |  |  |  |  |