**Question #1**

Let consider the mechanism of exceptions. You are requested to

1. List the events that can trigger an exception, grouping them in categories
2. Define what we mean by *precise exception*
3. Describe how we can guarantee precise exception management in pipelined processors
4. Describe how we can guarantee precise exception management in superscalar processors with dynamic scheduling and speculation.

Exceptions can be triggered during the execution of an instruction. During fetch or mem there should be an exception due to the memory, like page fault exception, during execution stage an exception can be triggered by arithmetic fault, like divide by 0, and in the decode by an illegal code.

Exceptions can be grouped in categories: Syncronous and asynchronous, user requested and coerced, maskarable and unmaskarable, within and between instruction and resumable and terminate.

We handle a pricese expection when we don’t execute the handler of exception until all the other previous instruction of instruction that raised the exception aren’t completed, and no other new instruction can be completed.

Guarantee precise exception in pipelined processors is more difficult, when a exception is raised, we set a flag to that relative instruction, and the exception is not handled until to all previous instruction aren’t completed.

To guarantee precise exception in superscalar processor with dynamic scheduling and speculation, the exception when is raised is put in ROB and can be executed when the instruction that triggered the exception is committed. If the instruction that triggered the exception is flushed the exception is discarded.

**Question #2**

Let consider a MIPS64 architecture including the following functional units (for each unit the number of clock periods to complete one instruction is reported):

* Integer ALU: 1 clock period
* Data memory: 1 clock period
* FP arithmetic unit: 2 clock periods (pipelined)
* FP multiplier unit: 6 clock periods (pipelined)
* FP divider unit: 10 clock periods (unpipelined)

You should also assume that

* The branch delay slot corresponds to 1 clock cycle, and the branch delay slot is not enabled
* Data forwarding is enabled
* The EXE phase can be completed out-of-order.

You should consider the following code fragment and, using the table in the following page (where each column corresponds to a clock period), and determine the pipeline behavior in each clock period, as well as the total number of clock periods required to execute the fragment, reporting the result in the right column in the table below.

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MIPS64 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; for (i = 0; i < 20; i++) {

; v5[i] = v1[i]\*v2[i] - v3[i]\*v4[i];

; }

|  |  |  |
| --- | --- | --- |
| .data | comments | Clock cycles |
| V1: .double “20 values” |  |  |
| V2: .double “20 values” |  |  |
| V3: .double “20 values”  …  V5: .double “20 zeros” |  |  |
|  |  |
|  |  |
|  |  |
| .text |  |  |
| main: daddui r1,r0,0 | r1← pointer |  |
| daddui r2,r0,10 | r2 <= 20 |  |
| loop: l.d f1,v1(r1) | f1 <= v1[i] |  |
| l.d f2,v2(r1) | f2 <= v2[i] |  |
| mul.d f5,f1,f2 | f5 <= v1[i]/v2[i] |  |
| l.d f3,v3(r1) | f3 <= v3[i] |  |
| l.d f4,v4(r1) | f4 <= v4[i] |  |
| mul.d f6, f3, f4 | f6 <= v3[i]\*v4[i] |  |
| sub.d f5,f5,f6 | f5 <= f5-f6 |  |
| s.d f5,v5(r1) | v5[i] <= f5 |  |
| daddui r1,r1,8 | r1 <= r1 + 8 |  |
| daddi r2,r2,-1 | r2 <= r2 - 1 |  |
| bnez r2,loop |  |  |
| Halt |  |  |
| total |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| daddui r1,r0,0 | F | D | X | M | W | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddui r2,r0,20 |  | F | D | X | M | W | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f1,v1(r1) |  |  | F | D | X | M | W | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f2,v2(r1) |  |  |  | F | D | X | M | W | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul.d f5,f1,f2 |  |  |  |  | F | D | R | X | X | X | X | X | X | M | W | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f3,v3(r1) |  |  |  |  |  | F |  | D | X | M | W |  |  |  |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f4,v4(r1) |  |  |  |  |  |  |  | F | D | X | M | W |  |  |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul.d f6, f3, f4 |  |  |  |  |  |  |  |  | F | D | R | X | X | X | X | X | X | M | W | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| sub.d f5,f5,f6 |  |  |  |  |  |  |  |  |  | F |  | D | R | R | R | R | R | X | X | M | W | 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| s.d f5,v5(r1) |  |  |  |  |  |  |  |  |  |  |  | F |  |  |  |  |  | D | X | S | M | W | 1 |  |  |  |  |  |  |  |  |  |  |  |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | D |  | X | M | W | 1 |  |  |  |  |  |  |  |  |  |  |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F |  | D | X | M | W | 1 |  |  |  |  |  |  |  |  |  |
| bnez r2,loop |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F | R | D | X | M | W | 2 |  |  |  |  |  |  |  |
| Halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | F |  |  |  | 1 |  |  |  |  |  |  |  |

6+21\*20=216