UNIVERSITY OF CALIFORNIA, SAN DIEGO

CSE 240A - Graduate Computer Architecture

Project Phase II - Traces

Test name - Fortunate Instruction Mix

Costas Zarifis

December 15, 2014

1 TEST DESCRIPTION - EXPECTED OUTCOME

After executing a series of FP instructions with read after write dependencies, the floating point queue gets filled up. During the next cycle, the Fetch stage feeds the decode stage with 4 integer instructions. If the active list is empty and the integer queue is also empty (which will be the case in this test) the 4 instructions are going to get pushed to the integer list and get issued during the next stage even if the FP queue is full. As a result the integer instructions get issued and executed before the floating points instructions.

2 TRACE FILE - RESULTS

The trace file which can be found on the submitted HTML page, on this URL and on the page bellow contains a series of floating point operations followed by integer operations. Notice that all floating point add instructions use the same physical register. After the registers get renamed the only kind of dependency that will still appear is RAW. Since the next instruction needs to wait until the second stage of the previous floating point operation, the floating point queue gets filled up. The reason why we have to wait until the end of the second cycle of the previous instruction and not until the commit stage is because there exists a forwarding path from the second stage of the execution to the beginning of the first one. Also notice that the integer operations don't have any dependencies.

Listing 1: Fortunate Instruction Mix - Trace File

```
1  A 01 01 01
2  A 01 01 01
3  A 01 01 01
4  A 01 01 01
5  A 01 01 01
6  A 01 01 01
7  A 01 01 01
8  A 01 01 01
9  A 01 01 01
10 A 01 01 01
11 A 01 01 01
12 A 01 01 01
13 A 01 01 01
14 A 01 01 01
15 A 01 01 01
16 A 01 01 01
17 A 01 01 01
18 A 01 01 01
19 A 01 01 01
20 A 01 01 01
21 I 02 02 01
22 I 02 02 01
23 I 02 02 01
24 I 02 02 01
```

That brings us to line 19. This line includes a floating point operation, however since the FP queue is full at that point, the decoded instruction has to wait before entering the queue until the 7th stage, because at that point a previous instruction leaves the queue. Notice that the active list has the new instruction on the first appearance of the decode stage but the instruction actually enters the queue on the 7th cycle (both the active list and the FP list are available on the submitted HTML file). The same procedure is followed by instruction 20 but it has to wait even more until an instruction leaves the queue.

Notice however that the following four instractions are integer operations. Since there are no dependencies these instructions are issued without any delays and are executed two at a time by the corresponding ALU units. There are only two ALU units so these instructions obviously cannot get executed all at the same time.