CSE240A Project Report

Yan Zhang A53050564

System Implementation

In this project, a 1-way issue R10K Simulator is designed and implemented. It has the same out-of-order strategy as R10K though with some simplification.
The simulator contains 3 instruction queues (integer, float and address), an active list, a free list, a register mapping table, a busy-bit table, a pipelined memory unit and 4 ALUs (2 for integer and 2 for float). Except for the amount of registers, the configuration is exactly the same as R10K.

Designed Test Result

Test Name: Integer instruction queue execution throughput

Test Type: Structural

Rationale: In this test we can verify how many instructions can be issued in one cycle, which will represent the performance of the processor for integer instructions. There are two integer function units, which should be able to execute instructions concurrently. As always, the more instructions the processor can process in one cycle, the more powerful the processor will be.

Trace:

```
S 01 01 00 00000004
L 01 01 00 00000004
I 01 01 02
I 01 01 03
I 01 01 04
```

Results:

```
S 01 01 00: |F|D|I|A|E|C|
L 01 01 00:   |F|D|I|A| |E|C|
I 01 01 02:     |F|D|I| | |E|C|
I 01 01 03:       |F|D|I| |E|C|
I 01 01 04:         |F|D|I| |E|C|
Total cycles: 10
```

Analysis:
The result is exactly the same as expected. Since we have two ALUs, we can execute two integer instructions at the same time. And the dependency between Store and Load, as well as the following Integer instruction, works in the right way.

Class-wide Traces:

1. **S1**

```
M 00 03 00
L 00 00 00 FFFFFFFF
```

Output:

```
M 00 03 00: |F|D|I|E| |F|C|
L 00 00 00:   |F|D|I|A|E|C|
Total cycles: 7
```

Analysis: The result is the same as Sergio's. There is no dependency on register 0, so the second instruction will not wait for the first.

2. **S5**

```
M 01 01 01
M 01 01 01
I 00 00 00
M 01 01 01
M 01 01 01
I 00 00 00
I 00 00 00
M 01 01 01
M 01 01 01
I 00 00 00
I 00 00 00
I 00 00 00
M 01 01 01
M 01 01 01
M 01 01 01
I 00 00 00
I 00 00 00
I 00 00 00
I 00 00 00
```

Output:

```
M 01 01 01: |F|D|I|E| |F|C|
M 01 01 01:   |F|D|I| |E| |F|C|
I 00 00 00:     |F|D|I|E| | |C|
M 01 01 01:       |F|D|I| |E| |F|C|
M 01 01 01:         |F|D|I| | |E| |F|C|
I 00 00 00:           |F|D|I|E| | | |C|
I 00 00 00:             |F|D|I|E| | |C|
```

```
M 01 01 01:                |F|D|I| |E| |F|C|
M 01 01 01:                  |F|D|I| | |E| |F|C|
I 00 00 00:                    |F|D|I|E| | | |C|
I 00 00 00:                      |F|D|I|E| | |C|
I 00 00 00:                        |F|D|I|E| |C|
M 01 01 01:                          |F|D|I|E| |F|C|
M 01 01 01:                            |F|D|I| |E| |F|C|
M 01 01 01:                              |F|D|I| | |E| |F|C|
I 00 00 00:                                |F|D|I|E| | | |C|
I 00 00 00:                                  |F|D|I|E| | |C|
I 00 00 00:                                    |F|D|I|E| |C|
I 00 00 00:                                      |F|D|I|E| |C|
Total cycles: 24
```

Analysis: The result is the same as Sergio's. We can find a lot of 4-way commit in the result. But for the last instruction it could be committed one cycle earlier but due to the 4-way limitation it had to wait for 1 cycle.

3. **B1**

```
M 01 01 01
M 01 01 01
B 01 00 00 0
B 01 00 00 0
```

Output:

```
M 01 01 01: |F|D|I|E| |F|C|
M 01 01 01:   |F|D|I| |E| |F|C|
B 01 00 00:     |F|D|I| | |E|C|
B 01 00 00:       |F|D|I| | |E|C|
Total cycles: 10
```

Analysis: The result is the same as Sergio's. We can see the two branch instructions cannot be executed at the same time due to limited ALU, though they do not have dependency.

4. **B6**

```
A 01 01 01
A 01 01 01
A 01 01 01
A 01 01 01
B 00 00 00 1
M 01 01 01
```

```
M 01 01 01
```

Output:

```
A 01 01 01: |F|D|I|E| |F|C|
A 01 01 01:   |F|D|I| |E| |F| |C|
A 01 01 01:     |F|D|I| | |E| | |F|C|
A 01 01 01:       |F|D|I| | | | |E| |F|C|
B 00 00 00:         |F|D|I|E| | | | | |C|
M 01 01 01: XXX aborted.
M 01 01 01: XXX aborted.
B 00 00 00:                   |F|D|I|E|C|
M 01 01 01:                     |F|D|I|E| |F|C|
M 01 01 01:                       |F|D|I| |E| |F|C|
Total cycles: 19
```

Analysis: The result is the same as Sergio's. The format of aborted instructions is kind of different since I didn't have enough time to print them in a right format. We can see the mispredicted branch and aborted instructions were re-executed after two cycles, and previous instructions were not cancelled.

5. **LS1**

```
M 01 01 01
M 01 01 01
S 01 01 01 00000004
S 00 00 00 00000000
```

Output:

```
M 01 01 01: |F|D|I|E| |F|C|
M 01 01 01:   |F|D|I| |E| |F|C|
S 01 01 01:     |F|D|I| | |A|E|C|
S 00 00 00:       |F|D|I|A| | |E|C|
Total cycles: 11
```

Analysis: The result is the same as Sergio's. The second store instruction waited until the first one finished 2 cycles even though they do not have dependency, since store instructions must be executed in-order for memory access.

6. **LS4**

```
M 01 01 01
M 01 01 01
M 01 01 01
L 01 01 01 FF
```

```
S 00 00 00 FF
L 00 00 00 FF
```

Output:

```
M 01 01 01: |F|D|I|E| |F|C|
M 01 01 01:   |F|D|I| |E| |F|C|
M 01 01 01:     |F|D|I| | |E| |F|C|
L 01 01 01:       |F|D|I| | | |A|E|C|
S 00 00 00:         |F|D|I|A| | | |E|C|
L 00 00 00:           |F|D|I|A| | | | |E|C|
Total cycles: 15
```

Analysis: The result is the same as Sergio's. Loads must be executed for the second cycle after all previous Stores have been committed. This memory consistency is guaranteed by the dependency matrix, and it works fine in the simulator.

7. **T1**

```
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
```

```
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
B 00 00 00 1
```

Output:

```
B 00 00 00: |F|D|I|E| |C|
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00:           |F|D|I|E|C|
B 00 00 00:             |F|D|I|E| |C|
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00:                       |F|D|I|E|C|
B 00 00 00:                         |F|D|I|E| |C|
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00:                                   |F|D|I|E|C|
B 00 00 00:                                     |F|D|I|E| |C|
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00:
|F|D|I|E|C|
B 00 00 00:
|F|D|I|E| |C|
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00:
|F|D|I|E|C|
B 00 00 00:
|F|D|I|E| |C|
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
B 00 00 00: XXX aborted.
```

```
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
```

```
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
```

```
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
```

```
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
```

```
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00: XXX aborted.
     B 00 00 00:
|F|D|I|E|C|
     B 00 00 00:
|F|D|I|E| |C|
     B 00 00 00:
|F|D|I|E|C|
     Total cycles: 202
```

Analysis: The result is the same as Sergio's. All the branches were mispredicted but re-executed rightly. This is implemented with a branch stack.

8. **T5**

   I didn't survive this test case. There should be some problems in my program causing infinite loop…

Arch Trade-offs

Since I am using 1-way issue I cannot modify the maximum number of dispatches per cycle, but I set a parameter for maximum number of commits per cycle. I also adjusted the size of active list.

1. By changing the 4-way commits into 1-way, according to the tests, the performance decreases for about 30%-40%, but this design is cheaper in hardware.
2. By changing the size of active list. It didn't bring much benefit since I was not able to run T5.