Computer Architecture Project Report

**Rahul Shah**

**Peter Dunham**

**Brandon Jennings**

**November 2, 2013**

Table of Contents

Running the Executable 3

System Requirements 3

Compiling 3

Input 3

Output 3

Test Benchmarks 4

No Dependencies 4

Dependencies 5

Forwarding 6

Structure Hazards 8

Loop 11

Responsibilities 12

Appendix 13

Appendix A 13

Appendix B 14

Appendix C 15

Appendix D 17

# Running the Executable

## System Requirements

* Windows
* Visual Studios

## Compiling

This code can be run in the Visual Studios environment, making compiling as simple as running the code.

## Input

The input file should be a text file structured similar to:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 2 | 1 |  | 1 |
| FP adder | 3 | 3 |  | 1 |
| FP multiplier | 2 | 20 |  | 1 |
| Load/store unit | 3 | 1 | 4 | 1 |

ROB entries = 128

R1=10, R2=20, F2=30.1

Mem[4]=1, Mem[8]=2, Mem[12]=3.4

Add.d F1, F2, F3

Ld F4, 8(R1)

Bne R2, R3, -3

Explicit examples can be viewed in the benchmark section. Line 431 of the code can be modified to reference the particular input file to be used.

## Output

The output of the program will be displayed on the console. There are two ways to view the output. The final output of the instructions can be viewed or the output of each clock cycle can be viewed. Lines 499 and 500 can be commented to show the final output and uncommented to show the output every clock.

# 

# Test Benchmarks

## No Dependencies – All Instructions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 2 | 1 |  | 1 |
| FP adder | 3 | 3 |  | 1 |
| FP multiplier | 2 | 20 |  | 1 |
| Load/store unit | 3 | 1 | 4 | 1 |

ROB entries = 128

R1=10, R2=20, R4=5, R5=15, R7=100, R9=21, R10=11

F2=30.1, F4=2.5, F5=4, F7=23.78, F8=11.28, F10=5, F11=5

Mem[20]=1

ld f1, 10(r1)

sd f2, 10(r2)

add r3, r4, r5

add.d f3, f4, f5

addi r6, r7, 100

sub r8, r9, r10

sub.d f6, f7, f8

mult.d f9, f10, f11

This instruction file does not contain any dependencies.

The output should look like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ISSUE | EX | MEM | WB | COMMIT |
| ld | 1 | 2 | 3-6 | 7 | 8 |
| sd | 2 | 3 | 4-7 | 8 | 9 |
| add | 3 | 4 |  | 9 | 10 |
| add.d | 4 | 5-7 |  |  |  |
| addi | 5 | 6 |  |  |  |
| sub | 10 | 11 |  |  |  |
| sub.d | 11 | 12-14 |  |  |  |
| mult.d | 12 | 13-32 |  |  |  |

|  |  |
| --- | --- |
| R[1] | 10 |
| R[2] | 20 |
| R[3] | 20 |
| R[4] | 5 |
| R[5] | 15 |
| R[6] | 200 |
| R[7] | 100 |
| R[8] | 10 |
| R[9] | 21 |
| R[10] | 11 |

|  |  |
| --- | --- |
| F[1] | 1 |
| F[2] | 30.1 |
| F[3] | 6.5 |
| F[4] | 2.5 |
| F[5] | 4 |
| F[6] | 12.5 |
| F[7] | 23.78 |
| F[8] | 11.28 |
| F[9] | 25 |
| F[10] | 5 |
| F[11] | 5 |

|  |  |
| --- | --- |
| Mem[20] | 1 |
| Mem[30] | 30.1 |

*See Appendix A for output.*

## 

## Dependencies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 1 | 1 |  | 1 |
| FP adder | 3 | 2 |  | 1 |
| FP multiplier | 2 | 10 |  | 1 |
| Load/store unit | 2 | 1 | 1 | 1 |

ROB entries = 128

r7=50

f1=1, f2=5, f3=4, f4=2.5, f5=15.12, f6=3, f7=6, f11=5

Mem[150]=0.5

mult.d f1, f2, f3

mult.d f2, f1, f6

add.d f6, f11, f7

sub.d f8, f5, f2

add.d f2, f4, f11

This instruction file contains dependencies. There is a read-after-write dependency between the first and second *mult.d* instructions and also between the second *mult.d* and the *sub.d* instructions. There is a write-after-read dependency between the second *mult.d* and the first *add.d* instructions. There is a write-after-write between the second *mult.d* and the second *add.d* instructions.

The output should look like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ISSUE | EX | MEM | WB | COMMIT |
| mult.d | 1 | 2-11 |  | 12 | 13 |
| mult.d | 2 | 13-22 |  | 23 | 24 |
| add.d | 3 | 4-5 |  | 6 | 25 |
| sub.d | 4 | 24-25 |  | 26 | 27 |
| add.d | 5 | 6-7 |  | 8 | 28 |

|  |  |
| --- | --- |
| R[7] | 50 |

|  |  |
| --- | --- |
| F[1] | 20 |
| F[2] | 7.5 |
| F[3] | 4 |
| F[4] | 2.5 |
| F[5] | 15.12 |
| F[6] | 11 |
| F[7] | 6 |
| F[8] | -44.88 |
| F[11] | 5 |

|  |  |
| --- | --- |
| Mem[150] | 0.5 |

*See Appendix B for output.*

## Forwarding

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 2 | 10 |  | 1 |
| FP adder | 3 | 2 |  | 1 |
| FP multiplier | 3 | 10 |  | 1 |
| Load/store unit | 3 | 1 | 4 | 1 |

ROB entries = 128

r1=50

r2=2, r3=4, r5=15, r6=3, f4=6, f1=0.5

Mem[60]=0.5, Mem[61]=1.5, Mem[14]=2.5

add r1, r2, r3

add r4, r5, r6

sd f1, 10(r1)

sd f4, 11(r4)

ld f2, 12(r2)

This instruction file contains two *sd* instructions before a *ld* instruction. The *ld* instruction would normally be able to load into *f2* early because its destination is not dependent on a previous instruction. However, until the destination of the *sd* instructions (before the *ld*) are known, the *ld* instruction does not know where the *sd* instructions are storing, which could be the same place it is loading.

The output should look like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ISSUE | EX | MEM | WB | COMMIT |
| add.d | 1 | 2-11 |  | 12 | 13 |
| add.d | 2 | 3-12 |  | 13 | 14 |
| sd | 3 | 13 | 14-17 | 18 | 19 |
| sd | 4 | 14 | 15-18 | 19 | 20 |
| ld | 5 | 15 | 16-19 | 20 | 21 |

|  |  |
| --- | --- |
| R[1] | 6 |
| R[2] | 2 |
| R[3] | 4 |
| R[4] | 18 |
| R[5] | 15 |
| R[6] | 3 |

|  |  |
| --- | --- |
| F[1] | 0.5 |
| F[2] | 2.5 |
| F[4] | 6 |

|  |  |
| --- | --- |
| Mem[14] | 2.5 |
| Mem[16] | 0.5 |
| Mem[29] | 6 |
| Mem[60] | 0.5 |
| Mem[61] | 1.5 |

*See Appendix C.1 for output.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 2 | 5 |  | 1 |
| FP adder | 3 | 2 |  | 1 |
| FP multiplier | 3 | 10 |  | 1 |
| Load/store unit | 3 | 1 | 4 | 1 |

ROB entries = 128

r1=50, r2=8, r3=4, r5=15, r6=3, r7=10

f4=6, f1=0.5

Mem[60]=0.5, Mem[61]=1.5, Mem[14]=2.5

add r1, r2, r3

add r4, r5, r6

sd f1, 10(r7)

ld f2, 12(r2)

This instruction file contains forwarding between the *ld* and *sd* instructions. The *sd* instruction stored to the location the *ld* instruction was loading from. As soon as it finished, the data from *sd* was forwarded to *ld*, hence *ld* instruction can begin after the *sd* instruction executes.

The output should look like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ISSUE | EX | MEM | WB | COMMIT |
| add.d | 1 | 2-6 |  | 7 | 8 |
| add.d | 2 | 3-7 |  | 8 | 9 |
| sd | 4 | 4 | 5-8 | 9 | 10 |
| ld | 4 | 5 | 6-10 | 11 | 12 |

|  |  |
| --- | --- |
| R[1] | 12 |
| R[2] | 8 |
| R[3] | 4 |
| R[4] | 18 |
| R[5] | 15 |
| R[6] | 3 |
| R[7] | 10 |

|  |  |
| --- | --- |
| F[1] | 0.5 |
| F[2] | 0.5 |
| F[4] | 6 |

|  |  |
| --- | --- |
| Mem[14] | 2.5 |
| Mem[16] | 0.5 |
| Mem[60] | 0.5 |
| Mem[61] | 1.5 |

*See Appendix C.2 for output.*

## 

## Structure Hazards

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 1 | 1 |  | 1 |
| FP adder | 1 | 3 |  | 1 |
| FP multiplier | 1 | 10 |  | 1 |
| Load/store unit | 1 | 1 | 4 | 1 |

ROB entries = 128

R1=10, R2=20, R4=5, R5=15, R7=100, R9=21, R10=11

F2=30.1, F4=2.5, F5=4, F7=23.78, F8=11.28, F10=5, F11=5

Mem[20]=1

ld f1, 10(r1)

sd f2, 10(r2)

add r3, r4, r5

add.d f3, f4, f5

addi r6, r7, 100

sub r8, r9, r10

sub.d f6, f7, f8

mult.d f9, f10, f11

This instruction file contains a structural hazard in the reservation station. All of the reservation stations for the types of functions only have space for one operation.

The output should look like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ISSUE | EX | MEM | WB | COMMIT |
| ld | 1 | 2 | 3-6 | 7 | 8 |
| sd | 8 | 9 | 10-13 | 14 | 15 |
| add | 9 | 10 |  | 15 | 16 |
| add.d | 10 | 11-13 |  | 16 | 17 |
| addi | 16 | 17 |  | 18 | 19 |
| sub | 19 | 20 |  | 21 | 22 |
| sub.d | 20 | 21-23 |  | 24 | 25 |
| mult.d | 21 | 22-31 |  | 32 | 33 |

|  |  |
| --- | --- |
| R[1] | 10 |
| R[2] | 20 |
| R[3] | 20 |
| R[4] | 5 |
| R[5] | 15 |
| R[6] | 200 |
| R[7] | 100 |
| R[8] | 10 |
| R[9] | 21 |
| R[10] | 11 |

|  |  |
| --- | --- |
| F[1] | 1 |
| F[2] | 30.1 |
| F[3] | 6.5 |
| F[4] | 2.5 |
| F[5] | 4 |
| F[6] | 12.5 |
| F[7] | 23.78 |
| F[8] | 11.28 |
| F[9] | 25 |
| F[10] | 5 |
| F[11] | 5 |

|  |  |
| --- | --- |
| Mem[20] | 1 |
| Mem[30] | 30.1 |

*See Appendix D.1 for output*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 2 | 5 |  | 1 |
| FP adder | 3 | 5 |  | 1 |
| FP multiplier | 2 | 10 |  | 1 |
| Load/store unit | 3 | 1 | 4 | 1 |

ROB entries = 2

R1=10, R2=20, R4=5, R5=15, R7=100, R9=21, R10=11

F2=30.1, F4=2.5, F5=4, F7=23.78, F8=11.28, F10=5, F11=5

Mem[20]=1

ld f1, 10(r1)

sd f2, 10(r2)

add r3, r4, r5

add.d f3, f4, f5

addi r6, r7, 100

sub r8, r9, r10

sub.d f6, f7, f8

mult.d f9, f10, f11

This instruction file contains a structural hazard in the reorder buffer. There are only two entries.

The output should look like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ISSUE | EX | MEM | WB | COMMIT |
| ld | 1 | 2 | 3-6 | 7 | 8 |
| sd | 2 | 3 | 4-7 | 8 | 9 |
| add | 9 | 10-14 |  | 15 | 16 |
| add.d | 10 | 11-15 |  | 16 | 17 |
| addi | 17 | 18-22 |  | 23 | 24 |
| sub | 18 | 19-23 |  | 24 | 25 |
| sub.d | 25 | 26-30 |  | 31 | 32 |
| mult.d | 26 | 27-36 |  | 37 | 38 |

|  |  |
| --- | --- |
| R[1] | 10 |
| R[2] | 20 |
| R[3] | 20 |
| R[4] | 5 |
| R[5] | 15 |
| R[6] | 200 |
| R[7] | 100 |
| R[8] | 10 |
| R[9] | 21 |
| R[10] | 11 |

|  |  |
| --- | --- |
| F[1] | 1 |
| F[2] | 30.1 |
| F[3] | 6.5 |
| F[4] | 2.5 |
| F[5] | 4 |
| F[6] | 12.5 |
| F[7] | 23.78 |
| F[8] | 11.28 |
| F[9] | 25 |
| F[10] | 5 |
| F[11] | 5 |

|  |  |
| --- | --- |
| Mem[20] | 1 |
| Mem[30] | 30.1 |

*See Appendix D.2 for output*

## Loop

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of rs | Cycles in EX | Cycles in Mem | # of FUs |
| Integer adder | 3 | 1 |  | 1 |
| FP adder | 3 | 3 |  | 1 |
| FP multiplier | 2 | 20 |  | 1 |
| Load/store unit | 3 | 1 | 4 | 1 |

ROB entries = 128

R1=5, R3=3

add r1, r1, r1

addi r2, r2, 1

bne r2, r3, -12

This instruction file contains a loop using a branch instruction. NOTE: branch instructions have not been implemented yet so there is no valid output from the code.

The output should look like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ISSUE | EX | MEM | WB | COMMIT |
| add | 1 | 2 |  | 3 | 4 |
| addi | 2 | 3 |  | 4 | 5 |
| bne | 3 | 4 |  | 5 | 6 |
| add | 5 | 6 |  | 7 | 8 |
| addi | 6 | 7 |  | 8 | 9 |
| bne | 7 | 8 |  | 9 | 10 |
| add | 9 | 10 |  | 11 | 12 |
| addi | 13 | 14 |  | 15 | 16 |
| bne | 17 | 18 |  | 19 | 20 |

# Responsibilities

Rahul – majority of coding, debugging, test bench cases

Brandon – test bench cases, report writing

Peter – some coding, debugging

# Appendix

## Appendix A

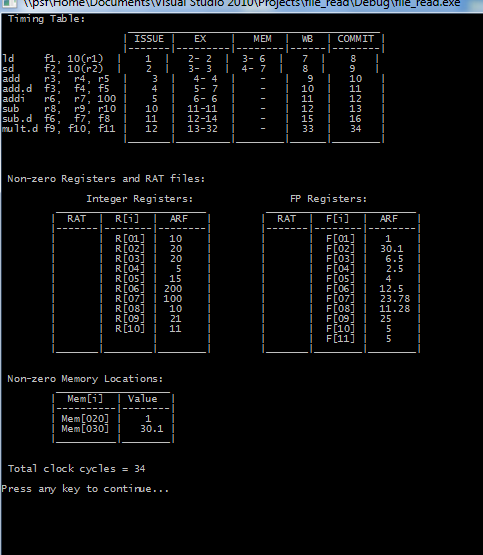


Figure 1: Output from instruction file containing no dependencies.

## Appendix B

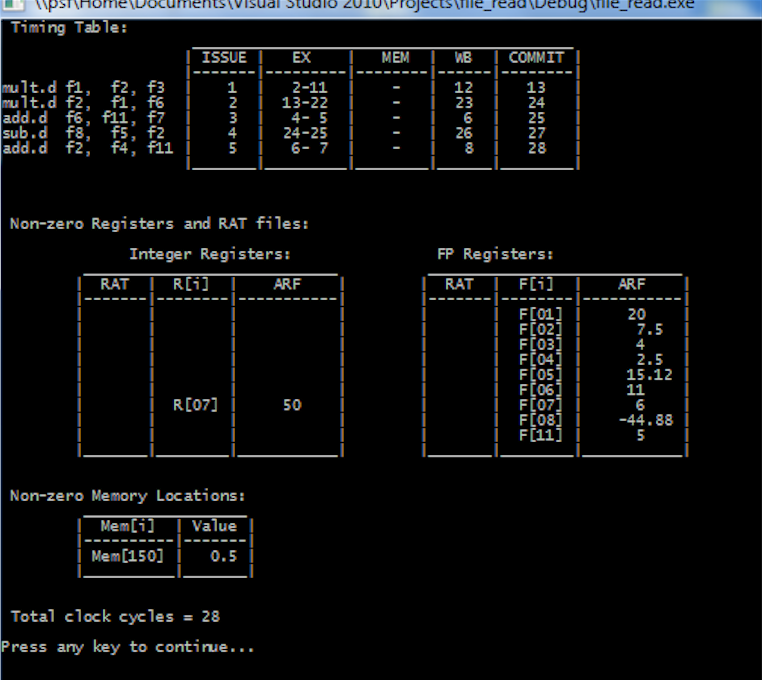


Figure 2: Output from instruction file containing dependencies.

## Appendix C

#### Appendix C.1

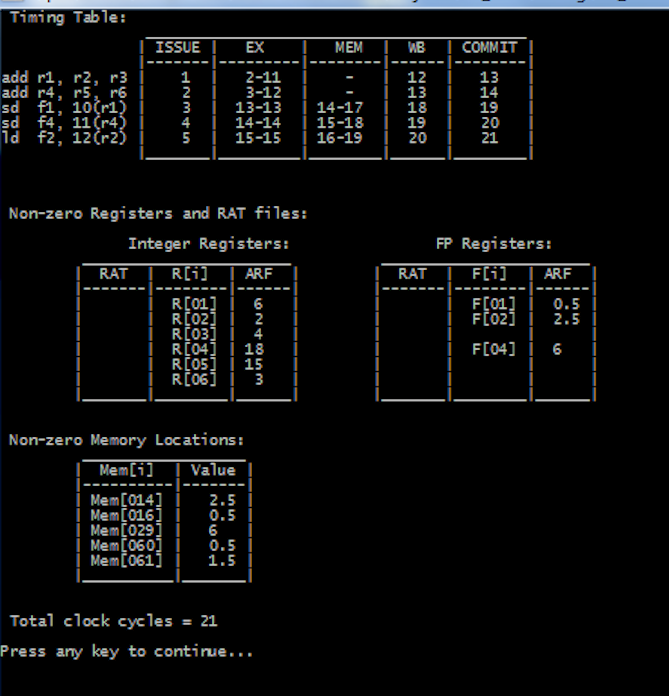


Figure 3: Output from instruction file containing a load instruction waiting for store instructions.

#### Appendix C.2

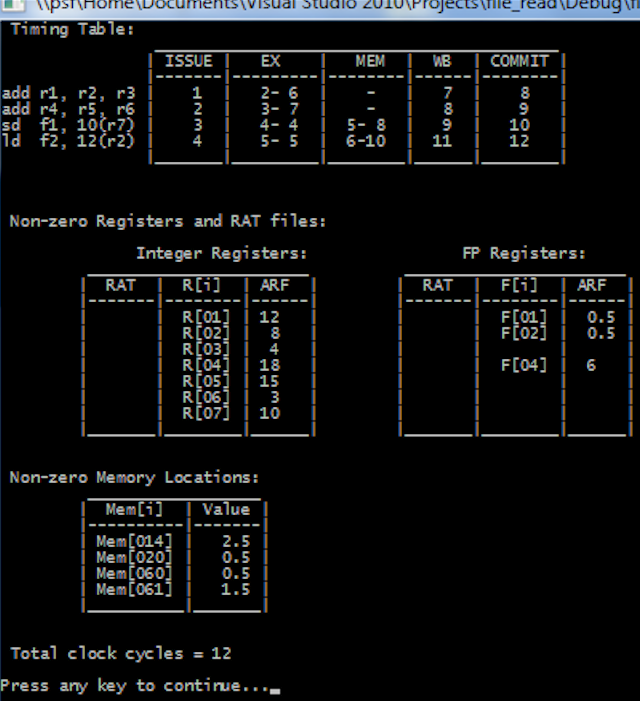


Figure 4: Output from instruction file containing forwarding between load/store.

## Appendix D

#### Appendix D.1

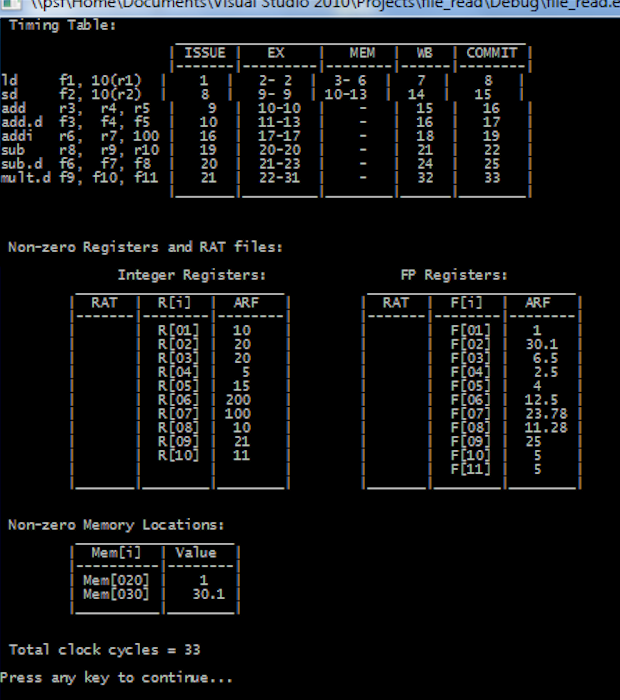


Figure 5: Output from instruction file containing a structural hazard in the reservation station.

#### Appendix D.2

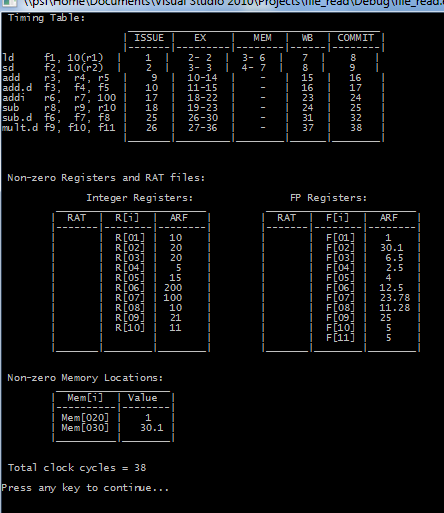


Figure 6: Output from instruction file containing a structural hazard in the reorder buffer