School of Engineering and Computer Science

NWEN 242 **Computer Organization**

Lab 2

The objective of this lab is to test your understanding of pipeline datapath operation and hazards. It is worth 10% of your final grade. The lab is marked out of 100.

The lab is due: midnight Wednesday, 11th October 2017.

Contents:

1. Introduction

In this lab you will use a simulator called **MPSIM**. A screenshot of MPSIM is given in Figure 1. MPSIM provides three settings: insert bubbles, forwarding, and branch assumption. By using the three settings properly, MPSIM can simulate five different MIPS processor pipeline modes:

- Basic Mode (turn off all settings),
- Bubbles Mode (turn on only "insert bubbles"),
- Bubbles with Branch Assumption (turn on "insert bubbles" and "branch assumption"),
- Forwarding (turn on "forwarding"), and
- Forwarding with Branch Assumption (turn on "forwarding" and "branch assumption").

MPSIM is written completely in Java. You can download MPSIM.jar from the course web pages. After downloading the jar file, you need to change your working directory to the folder that contains the jar file. Then on the command line, type the following to start the simulator program.

```
% java -jar mpsim.jar
```

You can also run MPSIM.jar at your home computer. Make sure that your computer has Java runtime installed.

You will discover how hazards arise and you will experiment with different ways of optimizing the hazards away.

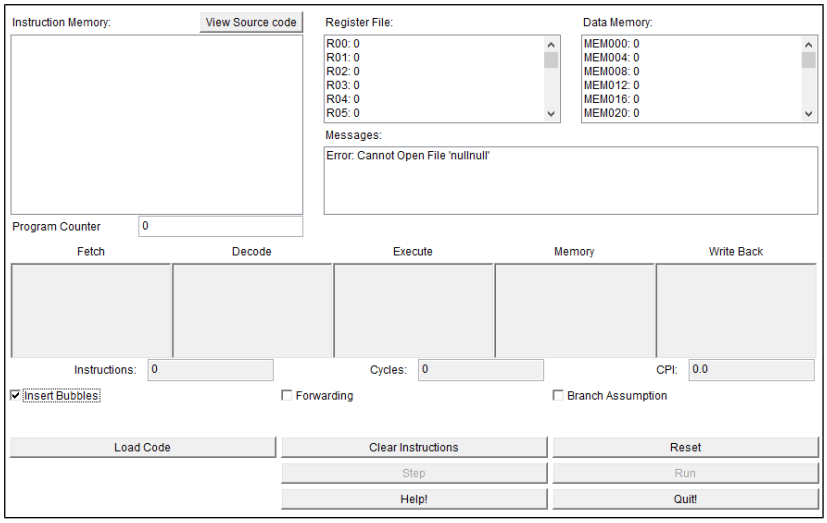

Figure 1: Screenshot of MPSIM.

2. How to prepare for the lab

1. Attend the lectures on pipelining.
2. Read the relevant chapter in the textbook (Chapter 4).
3. Read carefully the **MPSIM Technical Reference**.
4. Answer the preparatory questions.

3. Preparatory questions and answers

1. Which types of hazards did we consider in lectures? List them and briefly describe. Give an example for each type of a hazard. Suppose there is no hardware detection and prevention of hazards.

2. List and briefly describe hazard detection and prevention techniques.

3. Suppose there is neither hardware hazard detection nor prevention. How many nops are needed between any two subsequent instructions being involved in a hazard?

4. Suppose there is hardware detection of hazards. What feature has the special Register File, and what is the net effect of this feature?

5. Describe the way the Forwarding Unit detects a hazard between an R-type instruction and a lw instruction.

6. Suppose a pipelined datapath contains a forwarding unit. How many nops are needed between two consecutive R-type instructions that are at risk of a hazard?

7. Suppose a pipelined datapath contains a forwarding unit. How many nops are needed between an R-type instruction and a consecutive memory instruction that is at risk of a hazard?

8. Suppose a pipelined datapath contains a forwarding unit. How many nops are needed between a lw instruction and a consecutive R-type or a memory instruction that is at risk of a hazard?

9. Suppose a pipelined datapath makes a decision whether to branch or not in the second stage, computes the target address in the second stage and, if the branch is taken, updates the program counter at the end of the second stage. How many nops should be inserted between an R-type instruction and a following branch instruction if the branch instruction needs the result of the computation performed by the R-type instruction?

10. Suppose a pipelined datapath makes a decision whether to branch or not in the second stage, computes the target address in the second stage and, if the branch is taken, updates the program counter at the end of the second stage. How many nops should be inserted between an R-type instruction being in the MEM stage and a following branch instruction being in the ID stage, if the branch instruction needs the result of the computation performed by the R-type instruction?

11. Suppose a pipelined datapath makes a decision whether to branch or not in the second stage, computes the target address in the second stage and, if the branch taken, updates the program counter at the end of the second stage. How many nops should be inserted between a lw instruction and a following branch instruction if the branch instruction needs result of reading the memory done by the lw instruction?

4. Preliminary experiments

4.1 Preliminary experiment 1

Run the following program on MPSIM with basic mode, bubbles mode and forwarding mode. Modify your program to assign distinct values to $5, $1, and $3 and also to ensure correct operation of your program. Check your program by single stepping through the instructions. [**10 marks**]

```
start:
add $2, $5, $1
add $4, $2, $3
nop
nop
nop
END
```

Answer the following questions.

a) For each mode, after how many clock cycles will the destination register of the first add instruction, $2, receive the correct result value? Explain your answer.
b) For each mode, after how many clock cycles is the value of $2 needed in the second instruction? Explain your answer.
c) What is the problem here? What is this kind of hazard called?

4.2 Preliminary experiment 2

Run the following program on MPSIM with basic mode, bubbles mode, bubbles with branch assumption, forwarding, and forwarding with branch assumption. Check your program by single stepping through the instructions. [**10 marks**]

```
start:
nop
nop
beq $2, $1, start
addi $2, $2, 1
nop
nop
nop
END
```

Answer the following questions.

a) For each mode, how many cycles does it take until the branch instruction is ready to jump? Explain your answer.
b) What has happened with the following addi instruction while the branch is calculated? Explain your answer.
c) What is the problem here? What is this kind of hazard called?
d) What are the possible solutions to this problem?

4.3 Preliminary experiment 3

Run the following program on MPSIM with basic mode, bubbles mode, and forwarding mode. Modify your program when necessary to ensure correct operation of your program. Check your program by single stepping through the instructions. [**10 marks**]

```
start:
addi $2, $0, 3
sw   $2, 0($0)
nop
nop
nop
lw   $3, 0($0)
addi  $4, $3, 2
nop
nop
nop
END
```

Answer the following questions.

a) For each mode, after how many clock cycles will the destination register of the load instruction, $3, receive the correct result value? Explain your answer.
b) For each mode, after how many clock cycles is the value of $3 needed in the add instruction? Explain your answer.
c) What is the problem here? What is this kind of hazard called?
d) What are the possible solutions to this problem?

5. Modify and optimize a program

Copy the following assembly code into a file and name it **lab2program.asm**.

```
addi $2, $0, 3
sw   $2, 0($0)
addi $3, $0, 2
addi $9, $0, 12
sw   $9, 12($0)
uncon:
add  $1, $2, $3
lw   $5, 12($0)
sw   $1, 0($0)
addi $4, $0, 100
slt  $6, $1, $4
beq  $6, $0, fin
add  $2, $2, $2
add  $3, $3, $3
lw   $8, 0($0)
beq  $8, $0, fin
lw   $7, 0($5)
add  $7, $3, $7
beq  $0, $0, uncon
fin:
END
```

5.1 Make appropriate changes to the program lab2program.asm to run it in: [**30 marks**]

e) Basic Mode (all mode option buttons off),
f) Bubbles Mode (**Bubbles** button on),
g) Bubbles with Branch Assumption (**Bubbles**, **Branch Assumption** buttons on),
h) Forwarding (**Forwarding** button on), and
i) Forwarding with Branch Assumption (**Forwarding**, **Branch Assumption** buttons on).

To make changes, do not optimize your program by reordering or even omitting particular instructions, only insert an appropriate number of **NOPs** where needed.

In the program, the register **$7** accumulates result. Use the content of the register $7 to test whether your program runs correctly, or not. When the immediate of the first program instruction is 3, register **$7 = 76**. When it is set to 0, **$7 = 140**, use this to test your program.

Note: you will need to make five different tests for 5.1. For most of the tests, you will need to make a slightly different version of the lab2program.asm. You need to keep all versions of your program and demonstrate their correctness in front of the tutor.

When your programs produce correct results, enter appropriate data into the table below (assume that the immediate of the first program instruction is 3):

Mode Option	Instructions	Cycles	CPI
Basic			
Bubbles			
Bubbles with Branch Assumption			
Forwarding			
Forwarding with Branch Assumption			

5.2 When the same program is executed in different pipeline structures, the number of cycles is a measure of the performance of a pipeline structure. Analyze the content of your Cycles column and answer: [**10 marks**]

a) Why is the number of cycles and not number of instructions, or CPI (cycles per instruction) a measure of the performance (efficiency) of a pipeline structures?

b) Different versions of the same program are executed on different pipeline datapaths. Why do they exhibit different performance figures?

5.3 Optimize your Forwarding and Forwarding with Branch Assumption programs by reordering instructions (no instructions should be deleted except nop). Test whether they perform correctly by setting immediate in the first program instruction to **3** and then to **0**. Set the immediate to **0** and record the appropriate data into the following table: [**30 marks**]

Mode Option	Instructions	Cycles	CPI
Forwarding Optimized			
Forwarding with Branch Assumption Optimized			

You need to keep your optimized program and demonstrate its correctness and effectiveness in front of the tutor. Compare the measures of performance of your unoptimized and optimized programs.

Meanwhile, analyze the data you recorded and answer what makes the difference in the numbers of cycles used to run your optimized programs.

• How and what you should submit?

- Submit through the course submissions page before the due date and time.
- Submit your answers to questions 4.1, 4.2, 4.3, 5.1, 5.2, and 5.3. Include all your .asm programs.