CS151B/EE116C - Homework #5, Due 5/8/2017 (no late homeworks)

The midterm exam will be held in class, on Wednesday May 10, 2017, during the normal lecture time slot, in 190 Royce Hall.

- During the exam you will be able to use: the course textbook (**5th Edition**), hard copy printout of pp. 318-340 from chapter 5 of the 3rd edition, the class notes posted on the class web page, and a **simple calculator**. **No other material or devices can be used**. It will be **assumed** that you will have this material with you.

  Do not forget to bring the required material!
- You cannot use any material on which you have written your own notes. However, it is fine if all you have done is highlight or underline parts of otherwise permitted material.
- You are ''responsible'' for all the reading assignments, the class notes, as well as for everything covered in lectures and discussion sections.
- The midterm will cover all the material up to, and including, pipelining.

Reading Assignment:

- Section 2.21, pp. 2.21-1 – 2.21-6. This material can be found on the textbook web site. A copy is also available in the ''Supplementary Readings'' part of the class web page.
- ''Instruction Set Styles'' and ''The IBM/Motorola PowerPC'' from the **3rd Edition** of the book. This material is available in the ''Supplementary Readings'' part of the class web page.
- Chapter 6, of the **4th Edition**: pp. 570-572, 575-595 (input-output).
  Note: This is from the **Fourth** Edition of the Patterson & Hennessy book. This chapter is available in the ''Supplementary Readings'' part of the class web site.
- Chapter 5: pp. 381-383 (''Disk Memory'').
- Appendix A: pp. A-33 – A-37 (Exceptions and Interrupts on MIPS).
- We will use some material from chapter 8 of the 2nd edition of the Patterson and Hennessy book. We will refer to this as *Ed2-Chap8*. This material is available on the class web page in the ''Supplementary Readings'' section.
  - Ed2-Chap8: pp. 646-647 (''Magnetic Disks'')
  - Ed2-Chap8: 656-659 (ignore figures 8.7 and 8.8)
  - The examples in Ed2-Chap8: pp. 676-682
  - Ed2-Chap8: pp. 662-673

Problems:

(1) Consider the pipelined MIPS implementation shown in Figure 4.51 (page 304). The value of the `PCSrc` signal is computed in the MEM stage. Could the `PCSrc` signal be computed in the EX stage instead? If so, what would be the advantages of making this change? Also, what would be the disadvantages of making this change?

(2) Consider the pipelined MIPS implementation shown in Figure 4.51 (page 304). Consider each of the following two instructions separately:

a. `add $5,$15,$24`

b. `beq $1,$1,200`

For each of these instructions (separately), what is the value of the `PCSrc` signal when the instruction is in the MEM pipeline stage? Explain your answer.

(3) Consider executing the following code on the pipelined datapath of Figure 4.60 on page 316:

```
add  $2, $3, $1
sub  $4, $3, $5
add  $5, $3, $7
add  $7, $6, $1
add  $8, $2, $6
```

During the fifth cycle, which registers of the register file are read and which register of the register file is written?

(4) Consider the pipelined MIPS implementation shown in Figure 4.51 (page 304).

Figure 4.51 includes support for the `beq` instruction. However, Figures 4.51 ignores the existence of control dependencies. Specifically, Figure 4.51 does not include the required control hazard detection and stall implementation.

Your task is to add control hazard detection and stall implementation to Figure 4.51. Do not use any branch prediction mechanism (such as ''predict not taken''). Do not modify the basic datapath (you cannot use the modified datapath shown in Figure 4.62).

As a starting point, consider the hazard detection and stall implementation shown in Figure 4.60 (page 316), with the hazard detection unit logic described on pages 313-314. Modify this circuit as necessary and add the possibly modified circuit to Figure 4.51. Using the same style as in page 314, specify the logic of the hazard detection unit.

(5) Figure 4.65 (page 325) shows the pipelined MIPS implementation with the details of how stalls due to the `lw` and `beq` instructions are implemented. As discussed in class, the book doesn't fully explain how the stalls for `beq` are implemented. Assume that the implementation is as shown on slide 7.54 in the class notes.

As shown in Figure 4.65, the IF/ID register is a selectively-modified register. Assume that this register is implemented with flip-flops, as shown on the right side of slide 5.19 in the class notes.

A) On a copy of Figure 4.65, show the digital circuit (gate-level implementation) that generates the IF.Flush signal. Be sure that it is clear **exactly** how it is connected to the rest of the implementation. If it is connected to a module that generates multiple outputs, clearly identify the specific output to which your circuit is connected.

B) Given the assumption above regarding how the IF/ID register is implemented, show the digital circuit (gate-level implementation) that uses the IF.Flush signal to implement the desired functionality. Note that this part should be shown **separately** from the copy of Figure 4.65.

(6) Problem 4.10.2 in the book.

(7) Problem 4.10.3 in the book.

(8) An ISA that is very similar to the MIPS ISA supports 95 opcodes and 64 registers. Instructions are 32 bits wide. There is a class of instructions in this ISA that specify two register arguments and one signed (2's complement) immediate. What is the maximum possible range of values of this immediate?

(9) Problem 2.57 in the supplementary readings: ''The IBM/Motorola PowerPC.''

Practice problems: You do not need to hand in a solution to the problems below.

(10) Consider the pipelined MIPS implementation shown in Figure 4.60 (page 316), with the hazard detection logic described on page 314.

Consider only the following subset of the MIPS instruction set:

```
add, addi, sub, subi, and, andi, or, ori, slt, lw, sw
```

Assume that any forwarding that can be implemented, is implemented (don't worry about exactly how it is implemented).

A) Is the specified hazard detection and stall implementation logic (Figure 4.60 plus page 316) sufficient to ensure correct operation? Explain your answer.

B) **Claim:** The specified hazard detection and stall implementation logic (Figure 4.60 plus page 316) is not optimal in terms of performance.

I) Explain why this claim is correct.

II) Fix any deficiencies in the hazard detection and stall implementation logic. Specifically, if Figure 4.60 needs to be modified, show the modified figure. If the logic described on page 372 needs to be modified, provide the corrected description.

III) Are the changes made in part (II) likely to have a significant impact on performance? Explain your answer.

(11) Problem 4.10.1 in the book.

(12) Problem 4.16.1 in the book.

(13) Problem 4.16.2 in the book.

(14) Compare 0-, 1-, 2-, and 3-address ISAs by writing programs to compute

$$X = (A + B \times C)/(D - E \times F)$$

for each of the four ISAs. The instructions available for use are as follows:

0 Address	1 Address	2 Address	3 Address
PUSH M	LOAD M	MOV (X := Y)	MOV (X := Y)
POP M	STORE M	ADD (X := X+Y)	ADD (X := Y+Z)
ADD	ADD M	SUB (X := X−Y)	SUB (X := Y−Z)
SUB	SUB M	MUL (X := X*Y)	MUL (X := Y*Z)
MUL	MUL M	DIV (X := X/Y)	DIV (X := Y/Z)
DIV	DIV M		

M is a 16-bit memory address, and X, Y, and Z are either 16-bit addresses or 4-bit register numbers. The 0-address machine uses a stack, the 1-address machine uses an accumulator, and the other two have 16 registers and instructions operating on all combinations of memory locations and registers. SUB X,Y subtracts Y from X and SUB X,Y,Z subtracts Z from Y and puts the result in X. Assuming 8-bit opcodes and instruction lengths that are multiples of 4 bits, how many bits are needed to store the program for each ISA?