Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Prof. Archana Chidanandan

Homework 3
(RTL, Datapath and Control Unit)
Maximum points: 75 points

Directions

This assignment is due Friday, Jan. 28th. Submit your solutions on a separate sheet of paper.

Learning Objectives

In the process of completing this homework assignment, students will develop their abilities to

- Describe the implementation of machine language instructions using Register Transfer Language.
- Design datapaths to support machine language instruction sets by specifying the interconnections between components and the behavior of the associated control units.
- Design control units to support machine language instruction sets by applying combinational and sequential digital logic design principles.
- Design digital logic circuits for computer arithmetic.
- Predict the qualitative effect on clock cycle time of modifications to the design of digital logic circuits

General Instructions

- For the micro-programming problems, you do not have to re-produce the micro-program in Figure 5.7.3. Assume that that part of the micro-program already exists and write down only the new micro-instructions that are required. Also, if you create any new values for the fields or new fields, clearly specify what each new value/field denotes.
- Submit your answers on a separate sheet of paper.
- The state diagram for the FSM representation of the control unit (Figure 5.38) is provided on the class website.
- Figures 5.7.2 and 5.7.3 that deal with micro-programming are provided on the class website.
- If you are unsure of the RTL and datapath modifications that are required for Problems 1 and 2, refer to the class website for solutions to Homework 2.
- Figures for the 32-bit adder are also provided on the class website. Make modifications on these figures for Problem 4.

Problems

1. [20 points] Figure 5.7.3 in Hennessy and Patterson (the CD actually), shows the microprogram for the control unit of the MIPS architecture that can handle `lw, sw, beq, j` and R-type instructions. Add microcode to this program to implement the following MIPS instructions:

   a. `bne`
   b. `mfc0`

c. `mtc0`
d. `jal`

Figure 5.7.2 on the CD lists the control fields and their values that the microprogram uses.

2. [20 points] For each of the problems below, modify the textbook's multi-cycle control (Figure 5.38 on page 345) to implement the indicated MIPS instruction. Use a separate printout for each of the instructions. A copy of Figure 5.38 is provided on the class website.

a. `bne`
b. `mfc0`
c. `mtc0`
d. `jal`

3. Consider the `addmem` instruction - The description is provided below.

```
addmem rd, rs, rt
```

The instruction reads two values from memory, add the values and stores the result back in memory. The addresses of the values to be added are the contents of registers `rs` and `rt`. The address to which the result must be stored is available in register `rd`.

a. [5 points]Write a multicycle RTL description of an implementation of the `addmem` instruction that uses as few cycles as possible without extending the clock cycle of your design.

b. [5 points] Add any necessary datapath components and control signals to the multicycle datapath. A pdf version of Figure 5.28 is available on the class website(Homework).

c. [5 points] Modify the textbook's multi-cycle control (Figure 5.38 on page 345) to implement the indicated instruction. Use a separate printout for the instruction.

d. [5 points]Add microcode to the program in Figure 5.7.3 to implement the `addmem` instruction.

4. [5 points each] For each of the problems below, modify the 32-bit ripple-carry ALU developed in class as indicated. Use only inverters, AND gates, OR gates, and multiplexers. In each case, describe any new or modified control signals. Also, assuming that the ALU is currently on the critical path for your design, determine whether or not your modifications would extend the clock cycle time.

a. Modify the ALU to support the MIPS xor instruction.
b. Modify the ALU so that it detects overflow for both addition and subtraction (i.e. so that it has a new 1-bit output called Overflow which is asserted iff overflow

occurs). Read pages 170-172 of your textbook and Figure 3.3 for more information.

c. Modify the ALU so that it implements slt correctly even when overflow occurs. Assume that you have a combinational logic unit that detects overflow, as you are required to design for the previous problem.