

## **Homework 3 Solutions** **(Control unit design and Recursive procedures)** **Maximum points: 60**

### **Directions**

This assignment is due by 5:00PM on Wednesday, Oct. 27<sup>th</sup> for Sections 1, 2 and 3.

### **Learning Objectives**

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

- Design control units to support machine language instruction sets by applying combinational and sequential digital logic design principles.
- Use stacks to implement procedure calls in assembly language.
- Implement recursive procedures in an assembly language.

### **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.

### **Problems**

1. [25 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. `lwr`
- b. `mfc0`
- c. `mtc0`
- d. `jal`
- e. `lwr` - The description is provided below.

```
lwr rd, rs, rt
```

The instruction must read a value from memory and store it in register `rd`. The address, of the value in memory, is determined by adding the contents of registers `rs` and `rt`.

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

The micro-code fragment for the above instructions if provided below:

Label	ALU control	Src1	Src2	Register control and B input	Co-Processor Register control	Memory control	PCWrite control	Sequencing
Bne-1	Subt	A	B				ALU-Out-not Equal	Fetch
Mfc0-1						Read CoP		Seq
				Write D				Fetch
Mtc0-1					WriteCoP-B			
Jal-1				Write to Reg31			Jump Address	Fetch
Lwr-1	Funct code	A	B					LW2

- a. Read CoP – Read the register content from the Co-Processor using bits 15-11 of the instruction as the source register address and write the value to register D.
  - b. Write to Reg31 – Write to register 31 of the register file, the value of the PC.
  - c. Write to CoP B – Write to the Co-Processor using bits 15-11 of the instruction as the address field, the value from register B.
  - d. Write D – Write the value in register D to the register file, using bits 20-16 of the instruction as the address of the register to write to.
2. [25 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
  - e. lwr - The description is provided with problem 1.

The FSM state diagrams are posted on the website.

3. [10 points] The incomplete MIPS procedure below computes the recursive function

$$foo(a,b) = \begin{cases} b, & \text{if } a = 0 \\ foo(a-1,b-a) * b, & \text{otherwise} \end{cases}$$

Finish the procedure by adding the code for the procedure entrance, the recursive procedure call, and the procedure exit. Do not modify any of the given code, and be sure to follow the MIPS convention for register usage.

Hint: \$t0 and \$s0 hold the local values of *a* and *b*, respectively.

You may write the answer to this question in the boxes provided and attach the sheets with the rest of your solutions.

foo:

```

# Procedure entrance
addi $sp, $sp, -4 # Update stack pointer
sw   $ra, 0($sp) # Save $ra as "foo" calls a
                    # procedure

move $t0, $a0     # $t0 = $a0 = a
addi $sp, $sp, -4 # Update stack pointer
sw   $s0, 0($sp) # Save $s0, as it will be over-
                    # written with the new "b".
                    # The called procedure is
                    # responsible for ensuring that
                    # the saved registers values are
                    # preserved across a procedure
                    # call.
move $s0, $a1     # $s0 = $a1 = b
    
```

```

# if (a == 0) return b
bne $t0, $zero, Else
move $t1, $s0
j   Exit

# otherwise, return foo(a-1,b-a)*b
Else: sub $t2, $t0, 1
      sub $t3, $s0, $t0
      # Recursive procedure call
      move $a0, $t2 # $a0 = a-1 (The new "a").
      move $a1, $t1 # $a1 = b-a (The new "b").

      jal foo # Call the procedure foo with the
                # input arguments a-1 and b-a

      move $t1, $v0 # Move the returned value to $t1, as
                    # required by the "mul" instruction.

      # The color does not signify anything, just Word being
      # impossible. Please ignore.
    
```

```
mul $t1, $t1, $s0
```

```
Exit: move $v0, $t1  
      #Procedure exit
```

```
      lw $s0, 0($sp)           # Restore the values of $ra and $s0  
      addi $sp, $sp, 4  
  
      lw $ra, 0($sp)  
      addi $sp, $sp, 4  
  
      jr $ra
```