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

Homework 5 - Solutions
RTL, Datapath and Control Unit(FSM and Micro-programming)
Total points: 45 points + 20 points(extra credit)

This assignment is due Tuesday, February 3rd by 5:00 PM for Sections 1 and 2.

Learning Objectives

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

- Write RTL for assembly language instructions.
- Modify the datapath to implement an instruction,
- Modify the control unit to implement an instruction.
- Design the control unit using microprogramming.

General Instructions

- For the micro-programming problems, you do not have to re-produce the micro-program in Figure 5.46. 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, then clearly specify what each new value denotes.
- Submit your answers on a separate sheet of paper.

Problems

1. [20 points] Figure 5.46 in Hennessy and Patterson, 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. addi
    b. jal
    c. mfc0
    d. mtc0

    Hint: A line of microcode or an instruction in a microprogram is equivalent to a state in a state diagram.

2. [5 points] The effect of the instruction `swap rs, rt` is to move the contents of register `rs` to `rt` and move the contents of register `rt` to `rs`. For example, if
    - register `$1` contains `0x0000 0030` and
    - register `$2` contains `0x0000 0400`

    then the instruction `swap $1, $2` puts `0x0000 0030` in register `$2` and `0x0000 0400` in register `$1`.

The instruction format for the `swap` instruction is shown below:

Opcode	rs	rt	0x0000
31 26	25 21	20 16	15 0

The RTL for the `swap` instruction

1. IR = Memory[PC]
   PC = PC + 4
2. A = Reg [IR[25-21]]
   B = Reg [IR[20-16]]
   ALUOut = PC + (sign-extend (IR[15-0]) << 2)
3. Reg[IR[20:16]] = A
   B = B
4. Reg[IR[25:21]] = B

For the given RTL description, obtain the microinstructions for the `swap` instruction that can be added to the micro-program of Figure 5.46 of Hennessy and Patterson.

Solution to Problems 1 and 2.

Label	ALU Control	Src1	Src2	Register Control and B input	Co-Processor Register Control	Memory Control	PC Write Control	Sequencing
Fetch	Add	PC	4			Read PC	ALU	Seq
	Add	PC	ExtShft	Read	Read CoP			Dispatch1
*ADDI1	Add	A	Extend					Seq
				Write ALU 2				Fetch
*ADDI2				Write ALU 2				Fetch
JAL1				Write to Reg31			Jump Address	Fetch
MFC0				Write D				Fetch
MTC01					Write CoP B			Fetch
SWAP1				Write A				Seq
				Write B				Fetch

*There are two possible solutions for `addi`. If you compare clock cycle 3 for `addi` and `lw/sw`, you will find that they are identical. Therefore, instead of creating a new micro-instruction for cycle 3 of `addi`, one could use `dispatch 1` and go to the `Mem1` micro-instruction and then `dispatch 2` would have entries for `LW2`, `SW2` and `ADDI2`

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 ALU 2 – Write to the register file using bits 20-16 of the instruction as the destination register address, the value from the ALUOut register.
c. Write to Reg31 – Write to register 31 of the register file, the value of the PC.
d. 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.

e. Write A – Write to the register file using bits 20-16 of the instruction as the destination register address, the value from register A. Also, choose the current content of B to be written back toB.
f. Write B - Write to the register file using bits 25-21 of the instruction as the destination register address, the value from register B.

3. [20 points] Problem 5.19 of Hennessy and Patterson (page 429) describes a `wai` instruction.

a. Write the RTL description for the `wai` instruction.

```
1. IR = Mem[PC]; PC = PC + 4
2. A = Reg[IR[25:21]] ; B = Reg[IR[20:16]]
   ALUOut = PC + [SE[IR[15:0]] << 2]
   If (IR[31:26] == waiopcode) then
3. ALUOut = PC – 4      # Obtain the address of the current
                        # instruction
4. Reg[20:16] = ALUOut
```

b. Modify the datapath of Figure 5.33 (H&P) to implement the `wai` instruction. Remember to indicate any changes and additions to the control signals.

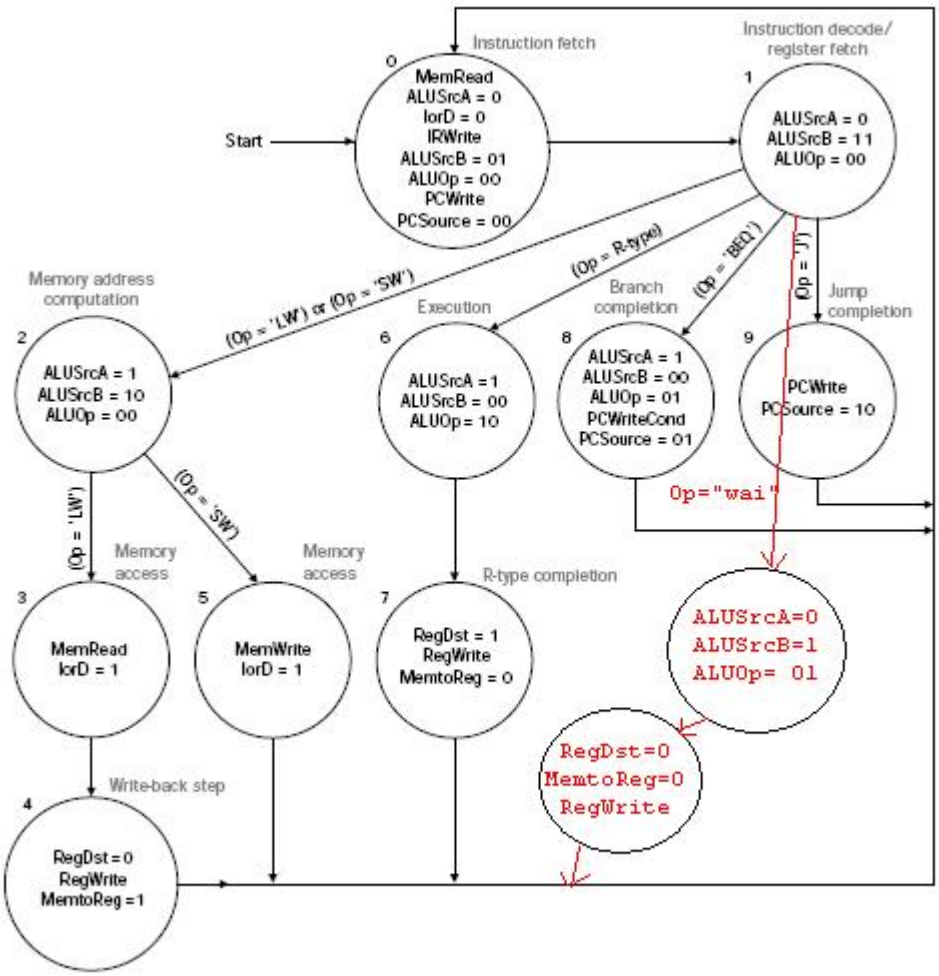
No changes required.

c. Modify the micro-program of Figure 5.46 (H&P) to implement the `wai` instruction.

See next page for solution.

d. Modify the state diagram of Figure 5.42 (H&P) to implement the `wai` instruction.

Label	ALU Control	Src1	Src2	Register Control	Memory Control	PC Write Control	Sequencing
WAI1	Sub	PC	4				Seq
				Write ALU 2			Fetch

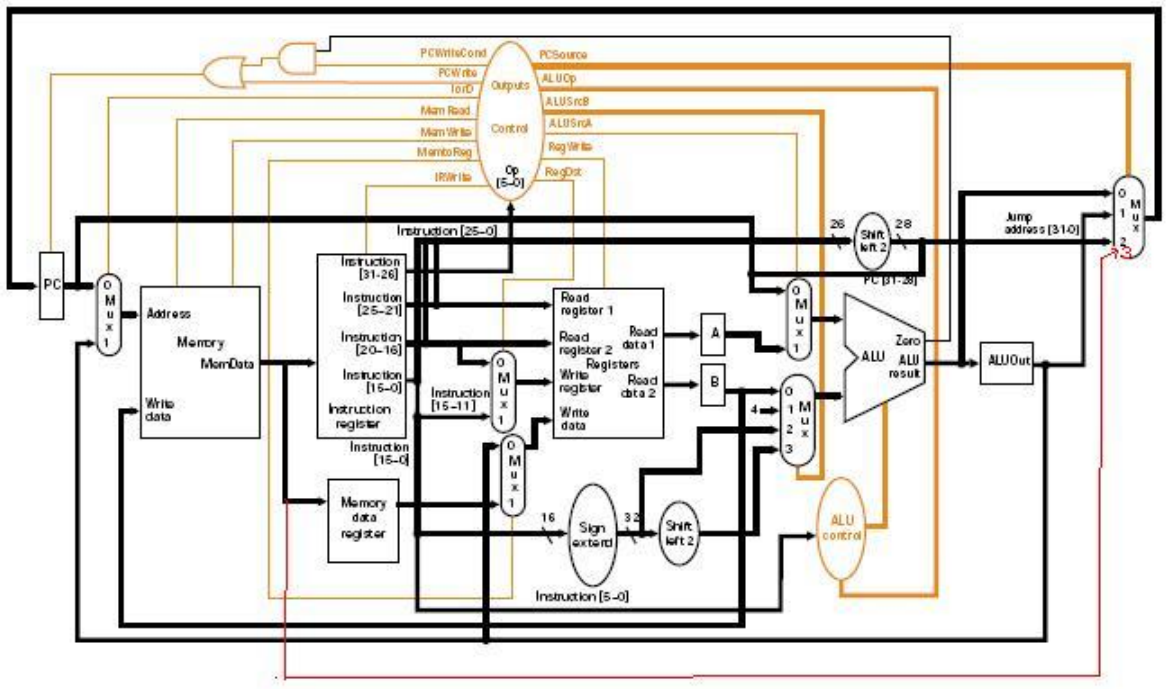


4. [Extra credit: 20 points] Problem 5.20 of Hennessy and Patterson (page 429) describes a `jm` instruction.

a. Write the RTL description for the `jm` instruction.

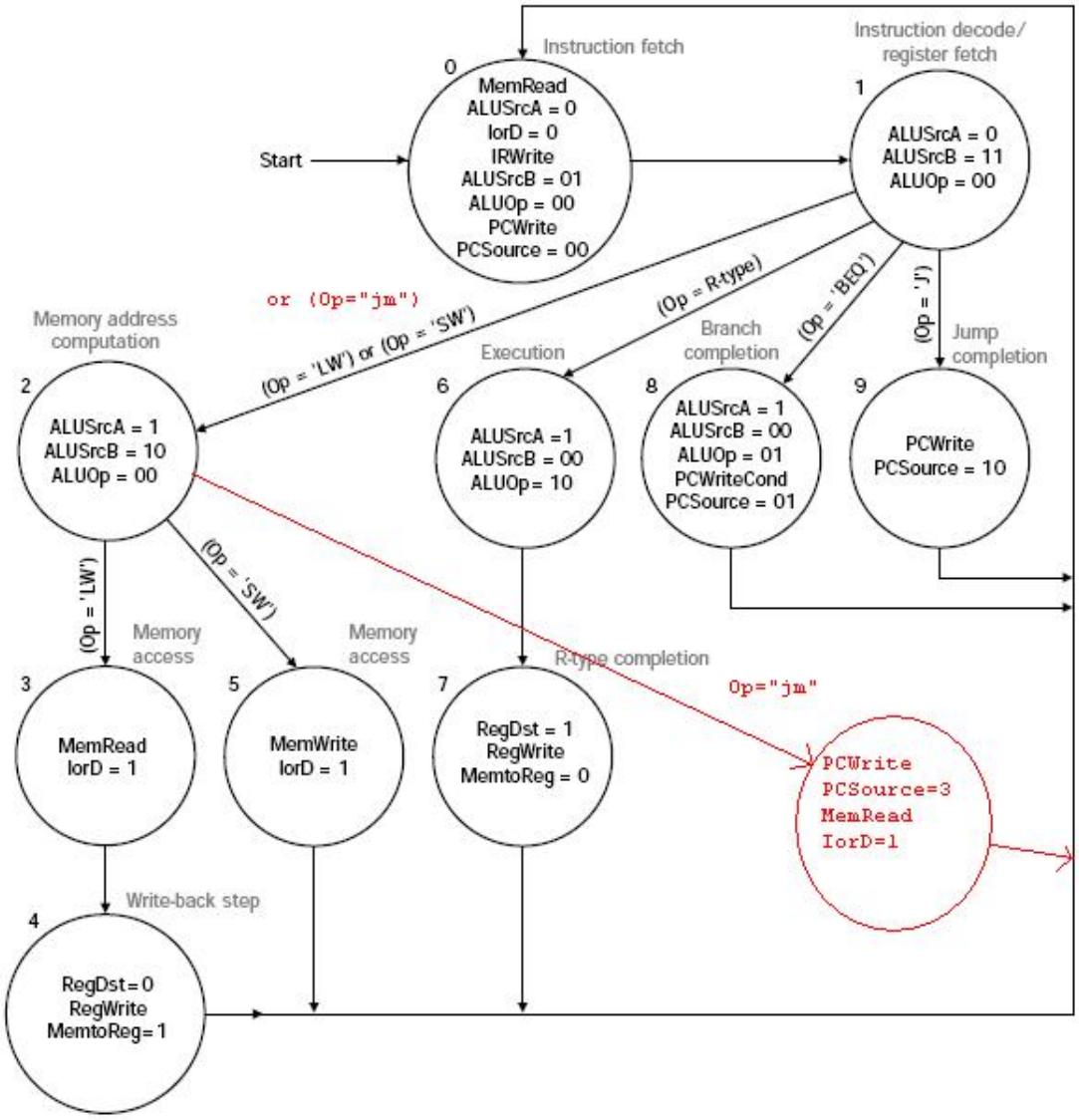
```
1. IR = Mem[PC]; PC = PC + 4
2. A = Reg[IR[25:21]] ; B = Reg[IR[20:16]]
   ALUOut = PC + [SE[IR[15:0]] << 2]
   If (IR[31:26] == jmopcode) then
3. ALUOut = A + SE[IR[15:0]]   # Identical lw/sw/addi
4. PC = Mem[ALUOut]
```

b. Modify the datapath of Figure 5.33 (H&P) to implement the `jm` instruction. Remember to indicate any changes and additions to the control signals.



Changes made to implement the "jm" instruction.

c. Modify the state diagram of Figure 5.42 (H&P) to implement the `jm` instruction.



d. Modify the micro-program of Figure 5.46 (H&P) to implement the `jm` instruction.

Label	ALU Control	Src1	Src2	Register Control	Memory Control	PC Write Control	Sequencing
JM2					Read ALUOut 2	Write Mem	Fetch

a. Read ALUOut2 – Read from memory using the value in ALUOut as the address. Write the output to the PC.
b. Write Mem – Write the output of memory to the PC.