Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Archana Chidanandan and J.P. Mellor

Homework 2 - Solutions
(Instruction representation, RTL and Datapath)
Maximum points: 75 points

Directions
This assignment is due Thursday, Jan. 20th. 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

- Determine addressing in branches and jumps.
- Interpret instruction formats and fields.
- Describe the implementation of machine language instructions using Register Transfer Language.
- Identify the components required to implement machine language instructions, including their input, output, and control signals, as well as their high-level behavior.
- Design datapaths to support machine language instruction sets by specifying the interconnections between components and the behavior of the associated control units.
- Determine how to handle exceptions, including interrupts.

Problems
1. [9 pts, 3 each] For the following MIPS assembly language instructions, list the machine language fields and their binary and hexadecimal values:

a) ori $2, $0, 10
opcode = 13 = 001101_2 = 0xD
r_s = $0 = 0 = 00000_2 = 0x0
r_d = $2 = 2 =$00010_2$ = 0x2
16-bit immediate = 10 = 0000 0000 0000 1010_2 = 0xA

b) jal main # main is at address 0x00400020 and the current instruction is in
# the same page as "main"
opcode = 3 = 000011_2 = 0x3
Label = 0000 0100 0000 0000 0000 0010 00_2 = 0x0100008

c) add $t7, $t2, $0
opcode = 0 = 000000_2 = 0x0 function code = 32 = 10 0000_2 = 0x20
r_s = $t2 = $10 = 10 = 01010_2 = 0xA
r_t = $0 = $0 = 0 = 00000_2 =0x0
r_d = $t7 = $15 = 15 = 01111_2 = 0xF

2. [4 pts] Assume that the MIPS instruction beq $t0, $s0, Label is located at address 0x0400 5678, and that Label is located at address 0x0400 1234. What will the binary value of the address field be? *Hint*: Remember that the offset is relative to the

instruction following the branch, and that all branch targets must be word aligned.

PC-relative addressing implies that the branch target address is a signed number of instructions from the PC.
Current instruction is at 0x400 5678. Therefore, PC = 0x0400 5678 + 4 = 0x0400 567C
The number of addresses between the PC and the target address = 0x0400 567C – 0x0400 1234 = 0x4448
The number of instructions = 0x4448/4 = 0x1114
Therefore, the 16-bit signed immediate value = 0001 0001 0001 0100

3. [4 pts] Assume that the MIPS instruction j Label is located at address 0x0400 5678, and that Label is located at address 0x0400 1234. What will the binary value of the target address field be? *Hint*: Remember that all branch targets must be word aligned.
Both the addresses (0x0400 567C i.e. the PC and the Label) are on the same page. This is determined by observing the MSB 4 bits of the address. Since, they are on the same page, we can use the Pseudo-direct addressing scheme to determine the 26-bit immediate value.

a. Drop the MSB 4 bits i.e. the page number. So, we now have 0x400 1234.
b. Divide by 4 or drop the last 2 bits to obtain the instruction count. So, we get 0x100 048D.

The binary value is 01 0000 0000 0000 0100 1000 1101

4. [3 pts] What sequence of MIPS instructions starting at address 0x0400 5678 could be used to branch to address 0x4400 1234?
This is an unconditional branch. The two addresses are not on the same page, therefore, a single "j" instruction will not suffice.
The following sequence is one possible solution:

```
lui $at, 0x4400
ori $at, $at, 0x1234
jr  $at
```

5. Figure 5.28 of Patterson & Hennessy, shows a complete datapath for a multicycle implementation of most R-Type MIPS assembly language instructions, as well as `lw`, `sw`, `beq`, and `j`. For this question, you are required to modify the datapath to include the multicycle implementation of the `bne` instruction. Specifically, you must:

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

b. [5 points] List all new and modified components required for the implementation of the `bne` instruction. Also, list the input, output, and control signals for each of those components. Indicate the number of bits in each signal.

c. [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).

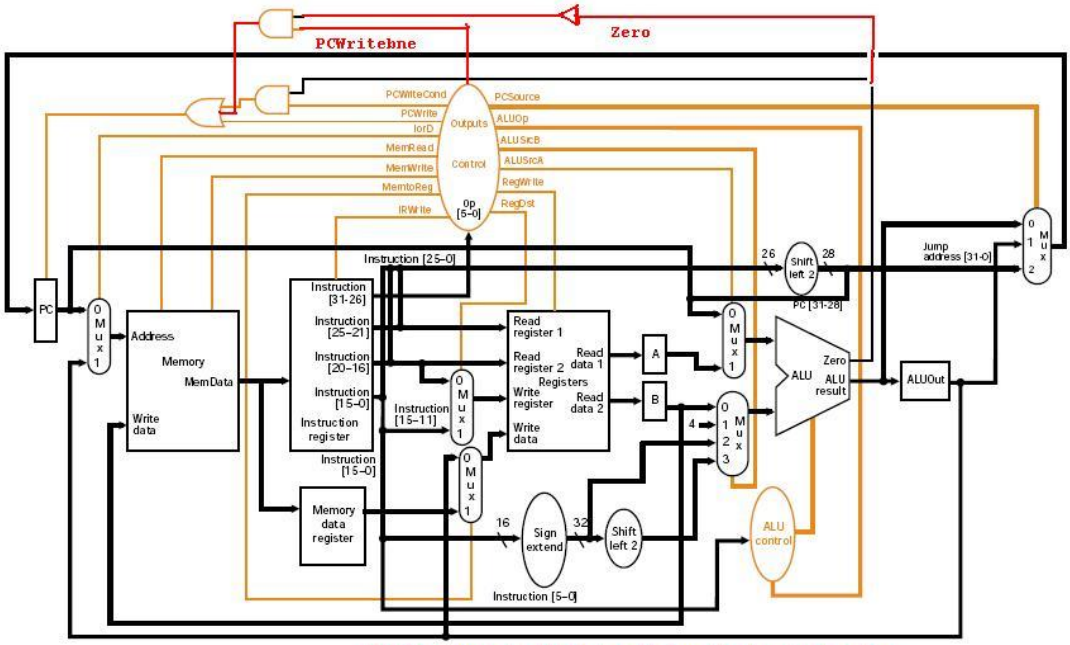
Solution:

a. RTL for the bne instruction

```
1. PC = PC + 4; IR = Mem[PC]
2.  A = Reg[IR[25:21]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If(IR[31:26]==5) then
3. If (A – B != 0) PC = ALUOut
```

b. An AND gate and an inverter are required.

c. The following datapath shows the changes required to implement the MIPS `bne` instruction.



Modifications to the datapath for the bne instruction

6.[15 points] Repeat Steps 1a, 1b and 1c for the `mfc0` and `mtc0` MIPS instructions (combine the component lists and the datapath modifications). *Hint:* Page A71 (on the CD) of Hennessey and Patterson, has a description for these two instructions.

a.RTL description for `mfc0` and `mtc0`

i) `mfc0 rt rd` - Move the contents of co-processor 0's register rd to register rt.

```
(1)     PC = PC + 4; IR = Mem[PC]
(2)     A = Reg[IR[25:21]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If((IR[31:26]==16)and(IR[25:21]==0)) then
```

```
(3)     D = CoP0Reg[IR[15:11]]; //Could do this in clock
                                //cycle 2 and save a clock cycle
(4)     Reg[IR[20:16]] = D;
```

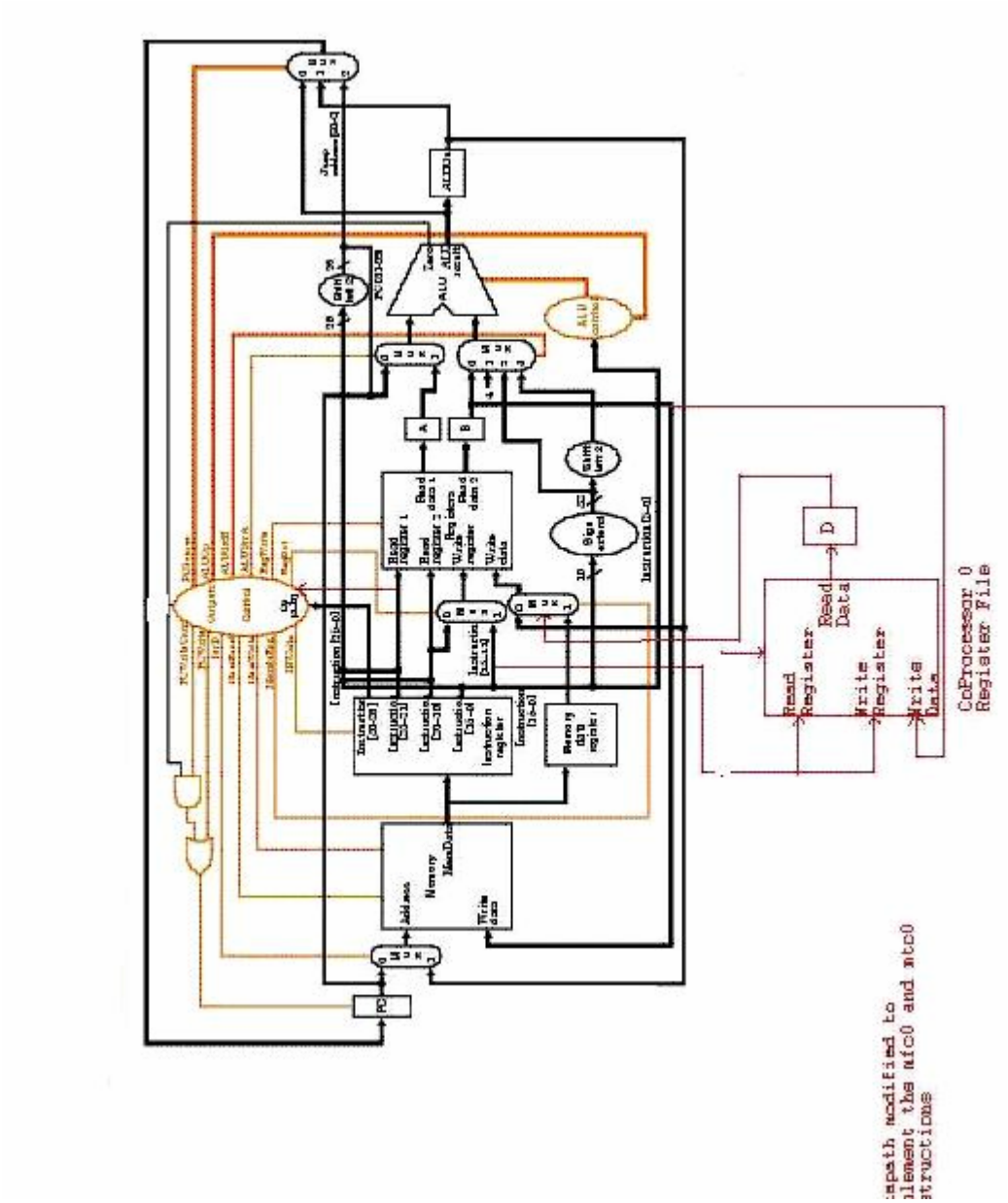
ii) `mtc0 rt rd` – Move the contents of register rd to co-processor 0's register rt.

```
(1)     PC = PC + 4; IR = Mem[PC];
(2)     A = Reg[IR[25:12]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If((IR[31:26]==16)and(IR[25:21]==4))then
(3)     CoP0Reg[IR[15:11]] = B;
```

b. New and modified components list

Component	Inputs	Outputs	Control signals
Co-processor 0 Register file	$CP0WriteData_{31:0}$, $CP0ReadAddr_{4:0}$, $CP0WriteAddr_{4:0}$	$CP0DataOut_{31:0}$	CP0Write
D	$DIn_{31:0}$	$DOut_{31:0}$	-

c. The following datapath shows the changes required to implement the MIPS `mfc0` and `mtc0` instructions.



7.[10 points] Repeat Steps 1a and 1b for an "undefined" instruction in MIPS. Figure 5.39 in your textbook, shows the modifications required for the datapath. *Hint:* Save ~~PC-4~~ PC in EPC, modify the PC, and update the ~~Status Register~~ Cause register. Read through pages 340-343 and pages A33-A35 (on the CD) of Patterson and Hennessy for more information.

a. RTL for the undefined instruction:

```
1. PC = PC + 4; IR = Mem[PC]
2. A = Reg[IR[25:21]];   B = Reg[IR[20:16]];
   ALUOut = PC + (SE[IR[15:0]] << 2);
   If((IR[31:26]==undefined) then
3. Cause[6:2] = value to indicate an undefined
   instruction; PC = 0x8000 0180; EPC = PC – 4;
```

b. New and modified components list

Component	Inputs	Outputs	Control signals
EPC	$EPCIn_{31:0}$,	$EPCOut_{31:0}$	EPCWrite(1 bit)
Cause	$CauseIn_{31:0}$	$CauseOut_{31:0}$	CauseWrite

The MUX that is connected to the input of the PC is modified to be a 4-input MUX

2) Figure 5.24 of Patterson & Hennessy, shows a complete datapath for a single-cycle implementation of most R-Type MIPS assembly language instructions, as well as `lw`, `sw`, `beq`, and `j`. For this question, you are required to modify the datapath to include the single cycle implementation of the `bne` instruction.

a. [5 points] Write a single-cycle RTL description of an implementation of the `bne` instruction.

b. [5 points] List all new and modified components required for the implementation of the `bne` instruction. Also, list the input, output, and control signals for each of those components. Indicate the number of bits in each signal.

c. [5 points] Add any necessary datapath components and control signals to the single-cycle datapath. A pdf version of Figure 5.24 is available on the class website(Homework).

a. RTL for the `bne` instruction

```
PC = PC+4
   if ( (Mem[PC])[31:26] == 5) then
          if ( ((Mem[PC])[25:21] – (Mem[PC])[20:16]) != 0) then
                    PC = PC + SE[(Mem[PC])[15:0]] << 2
```

b. An AND gate and a NOT gate are required.

c. The datapath modifications are shown below: