Lab 8 (All Sections) Prelab: ALU Control and Datapath

Name:

Your Name

Sign the following statement:

On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work

Type Name

1 Objective

In this lab you will implement part of the processor data and control paths. You are expected to be familiar with the MIPS single cycle processor from sections 4.2, 4.3 and 4.4 of the textbook.

2 Introduction

The datapath is the "brawn" of a processor, since it implements the fetch-decode-execute cycle. The general procedure to designing a single datapath is as follows:

1. Determine the instruction classes and formats in the instruction set one wishes to support;
2. Design datapath components and interconnections for each instruction class or format;
3. Integrate the datapath segments designed in Step 2 to yield a composite datapath which supports all desired instructions.

Simple datapath components include Instruction memory (stores the current instruction), program counter or PC (stores the address of current instruction), and ALU (executes current instruction). Implementation of the datapath for I- and J-format instructions requires two more components - a data memory and a sign extender.

In this lab, you will be adding more pieces to the datapath. You have already designed the ALU and sign extender, and next you will design the data memory, ALU control, and PC logic.

3 Questions

1. The data memory stores ALU results and operands, including instructions, and has two enabling inputs (MemWrite and MemRead). Can both these inputs (MemWrite and MemRead) be active at the same time? Explain why or why not?

2. List the functional units (Instruction memory, Register file, ALU, Data memory) that will be used by the following instruction classes:

   (a) R-type

   (b) Load Word

   (c) Store Word

   (d) Branch

   (e) Jump

3. How would you extend the single cycle datapath in Figure 1 to implement `bne` (Branch on not equal) instruction? This is similar to `beq` except that the branch is taken when the values are not equal. Explain your work briefly.

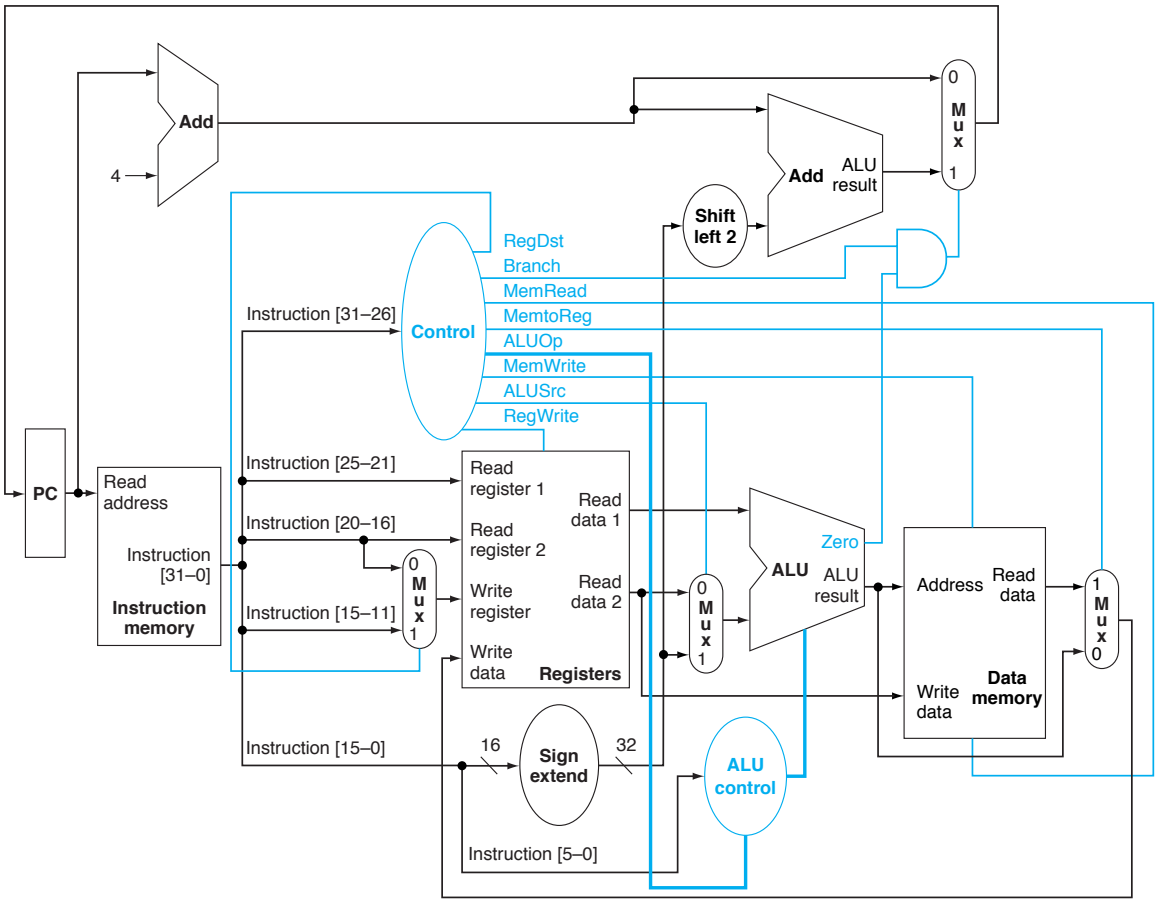

Fig. 1: Control Datapath

4. The processor needs to compute the next PC after each instruction. Write a verilog module which computes the next PC? Include the `j`, and `beq` cases. You may use the output of the "Control" module as shown in Figure 1.

5. How would you extend the single cycle data and control paths in Figure 1 to implement the `jr` (Jump To Register) instruction?

6. Investigate the possibility of adding a new instruction `srjr` that combines two existing instructions. For example, `srjr $ra, $sp, 16` will implement:

```
1        addi $sp, $sp, 16
2        jr $ra
```

   List any additional control or datapaths than need to be added above what is needed in `jr`. What will be the values of `RegDst`, `ALUSrc`, `MemToReg`, `RegWrite`, `MemRead`, `MemWrite`, `Branch`, `JumpToRegister`, and any control signals you added in the previous problem and added for this problem.