**Name:** **Id#**

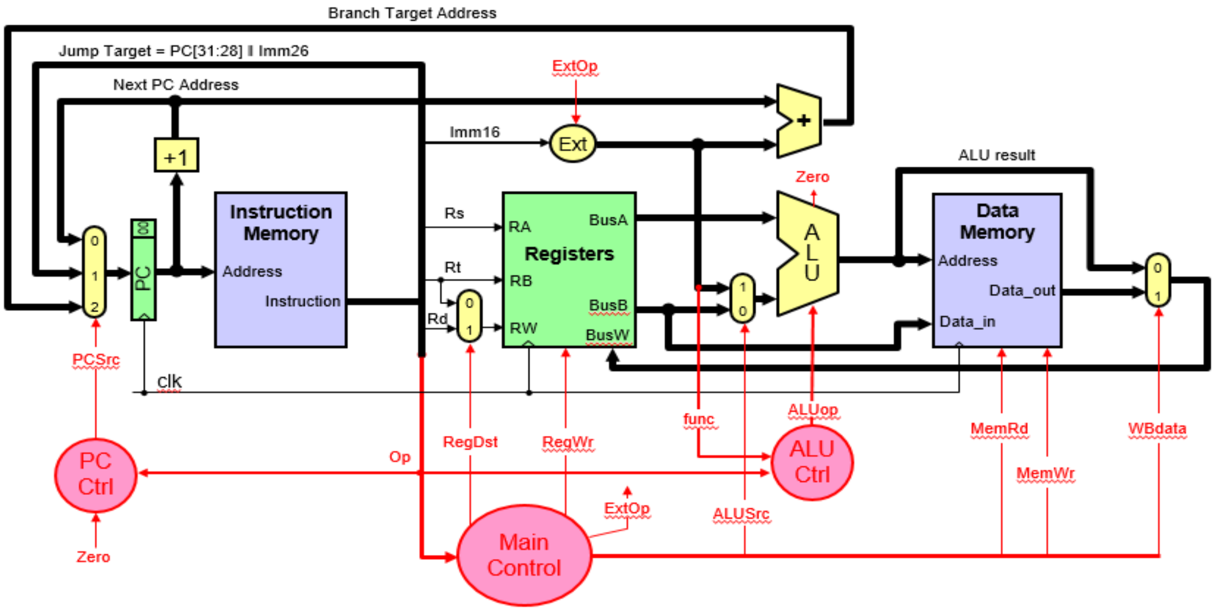
# COE 301/ICS 233, Term 172

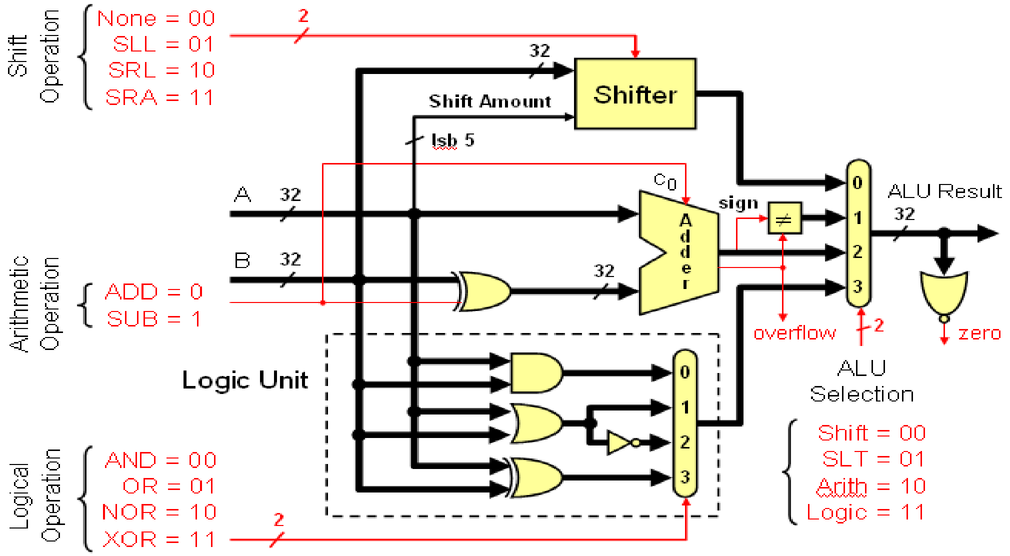
## Computer Architecture & Assembly Language

### Quiz# 6 Solution

Date: Tuesday, April 17, 2018

**Q1.** Consider the single-cycle datapath and control given below along with ALU design for the MIPS processor implementing a subset of the instruction set:





**(i)** **(5 points)** Show the control signals generated for the execution of the following instructions by filling the table given below:

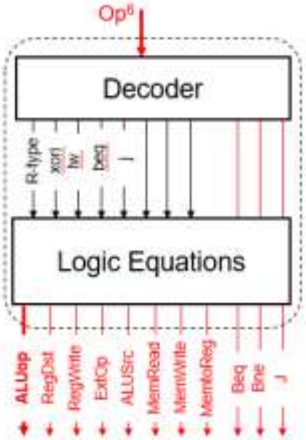
| Op | RegDst | RegWrite | ExtOp | ALUSrc | MemRead | MemWrite | WBdata |
|---|---|---|---|---|---|---|---|
| R-type | 1 = Rd | 1 | x | 0=BusB | 0 | 0 | 0 |
| xori | 0 = Rt | 1 | 0=zero | 1=Imm | 0 | 0 | 0 |
| lw | 0 = Rt | 1 | 1=sign | 1=Imm | 1 | 0 | 1 |
| bne | x | 0 | x | 0=BusB | 0 | 0 | x |
| j | x | 0 | x | x | 0 | 0 | x |

**(ii)** **(4 points)** Show the block diagram for designing the main control unit for this CPU and show the logic gates or equations for the control signals **RegDst**, **RegWrite** and **ExtOp** based on these instructions. Assume that the opcode of these instructions is a 6-bit opcode such that the opcode for R-type instructions is 0, the opcode for xori is 1, the opcode for lw is 2, and so on for the rest of the instructions.

RegDst <= R-type

RegWrite <= $\overline{(bne + j)}$

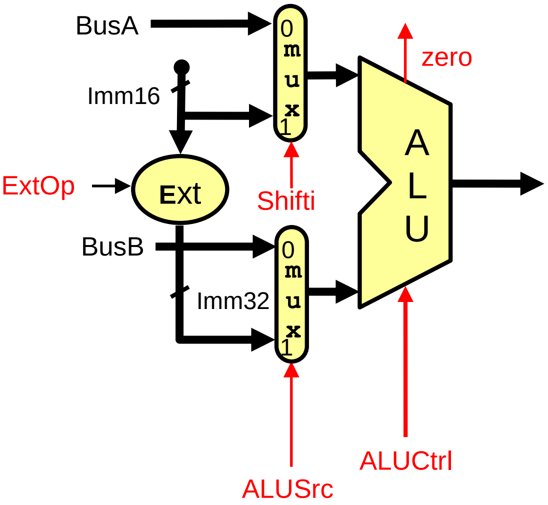
ExtOp <= lw



**(iii)** **(12 points)** We wish to add the following instructions to the MIPS single-cycle datapath. Add any necessary datapath modifications and control signals needed for the implementation of these instructions. Show only the **modified** and **added** components to the datapath.

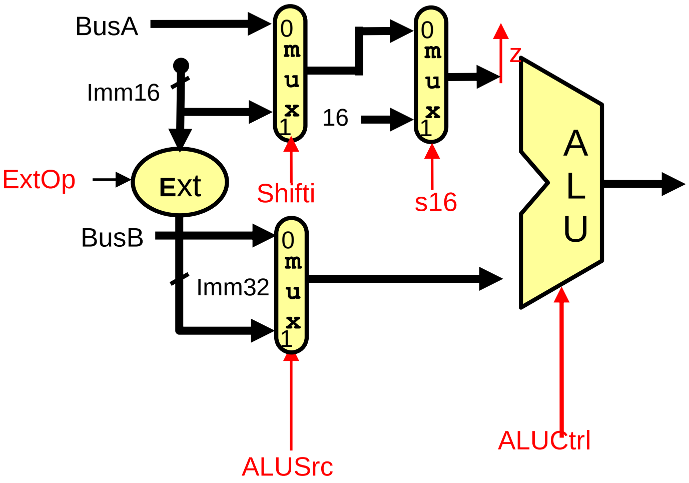
**a.** srl

For the srl instruction, examining the ALU one can see that the shift amount is coming through the A-input of the ALU and the operand to be shifted comes through the B input of the ALU. Thus, we need-to add a MUX on the A-input to select between the output of a register and the immediate values. This MUX needs to select only between the least significant 5 bits of BusA and bits 6 to 10 from Imm16. The modified part in the datapath is shown below:



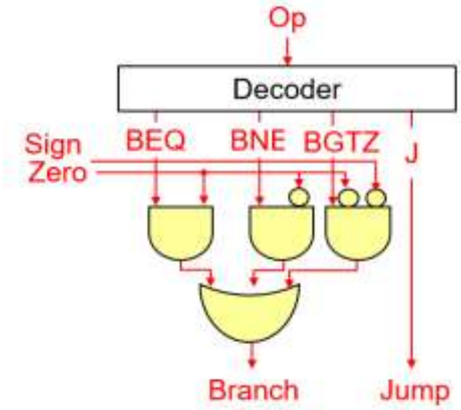
**b.** lui

For this instruction, the shift amount is 16 and the operand to be shifted is the immediate value selected on the B-input of the ALU. Thus, we need to add another MUX to select the shift amount as 16 for this instruction. The modified parts of the datapath to support the execution of this instruction is given below:



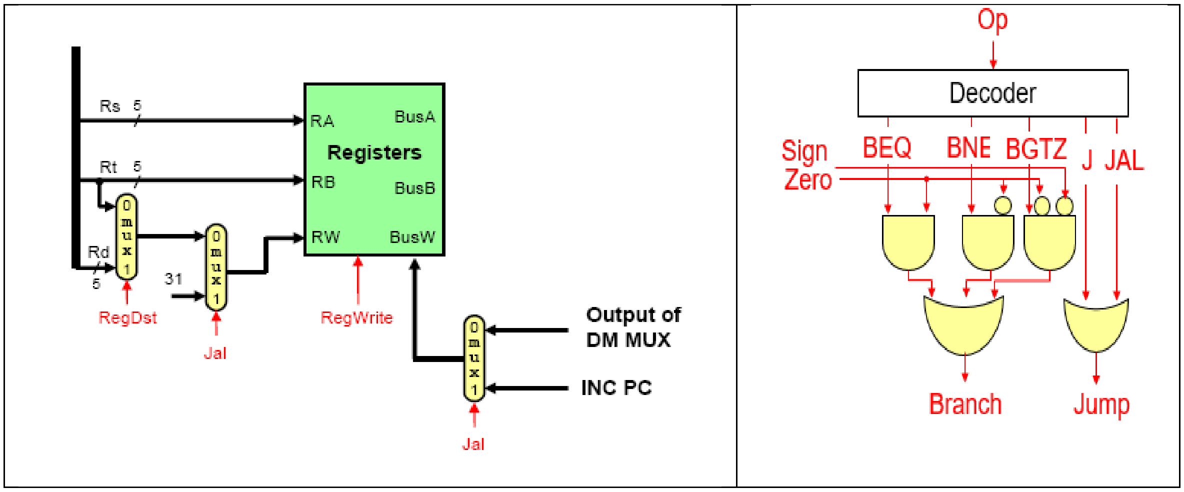
**c.** bgtz

Since the first source operand specified by RS comes on BusA and the second operand which is the Zero register specified by the RT filed comes on BusB, all we need is to get the operand on BusA to appear at the output of the ALU as we just need to check the sign bit (i.e. most significant bit of the result). Performing an addition, subtraction, xoring, oring operations will work. Let us assume that we will do an ALU addition operation. To check that the result is greater than 0, we need to check that the sign bit is 0 and that the result is not equal to zero. Thus, the changes needed to be done in the PC Control Logic are shown below:



**d.** jal

This instruction is similar to the jump instruction (J) with the difference that register $31 should be loaded with the incremented PC value. Thus, we need to add a MUX at the input of RW input to the register file to select the value 31 when executing this instruction. We also need to add a MUX at the input of BusW in the register file to select the incremented PC value to be loaded instead of the value coming from the output of the data memory MUX. In addition, we need to make changes to the PC Control Logic to perform the same operation needed by the J instruction for Jal instruction. These changes are shown below:


Rs 5
Rt 5
Rd
5
0 mux 1
31
0 mux 1
RegDst
Jal
RA
BusA
Registers
RB
BusB
RW
BusW
RegWrite
0 mux 1
Output of DM MUX
INC PC
Jal
Op
Decoder
Sign
Zero
BEQ
BNE
BGTZ
J
JAL
Branch
Jump