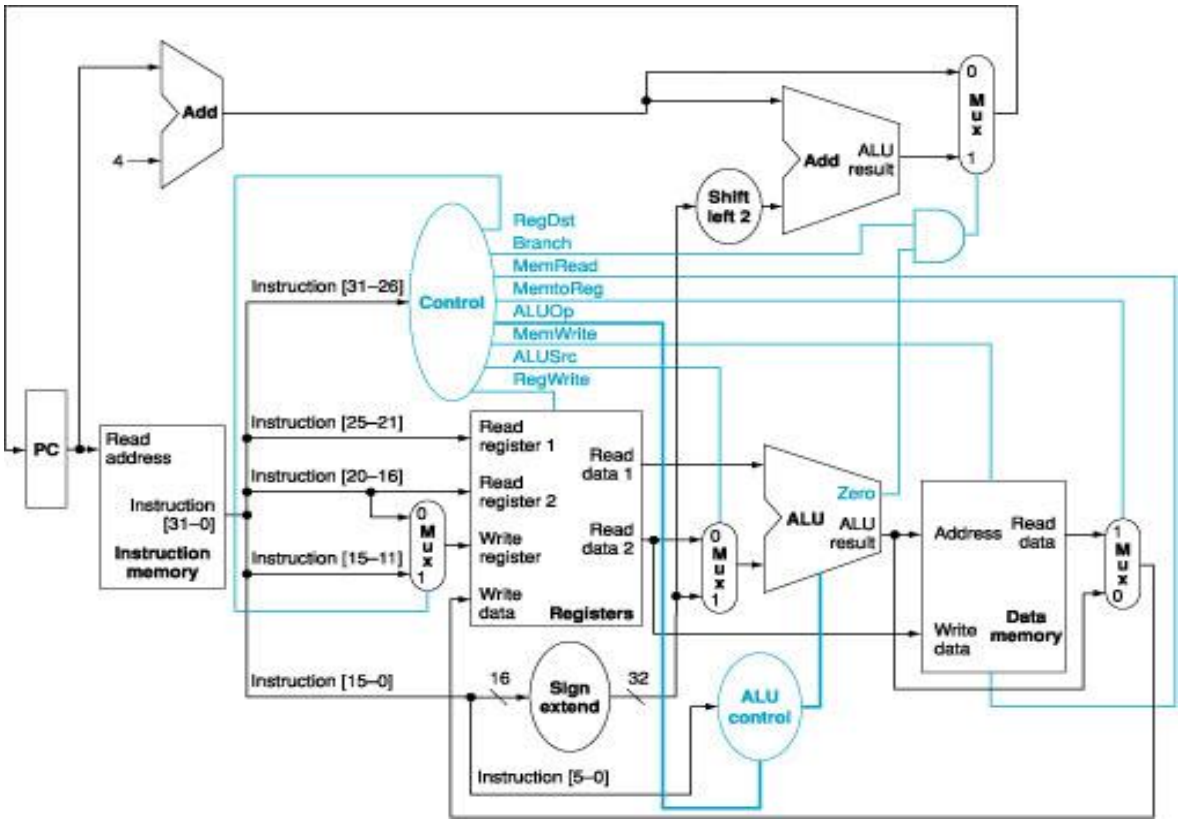
University of Maryland Baltimore County
Department of Computer Science and Electrical Engineering

CMSC 411, Computer Architecture

Assignment #3 **Solutions** Due: Tue 10/17/17 in the class

Question 1: *(30 Points)*



A digital circuit may be hit by stuck-at faults where one or multiple signals stay always 0 or 1 depending on the fault. Describe the effect that a single stuck-at fault would have for the signals shown below, in the single-cycle datapath shown above and discussed in class. Which instructions, if any, **will not work** correctly? Explain why. **6 points each**

Consider each of the following faults separately:

Instruction	RegDst	ALUSrc	Memto-Reg	Reg-Write	Mem-Read	Mem-Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

FIGURE 4.18 The setting of the control lines is completely determined by the opcode fields of the instruction. The first

From the above table, it can be easily observed that for such a control signal (column), if it is stuck-at fault with 0, the instruction(s) of 1 will not work (row). If it is stuck-at fault with 1, the instructions of 0 will not work. The reason for each case must be provided to get full credit.

A) RegWrite = 0

All R-type instructions with destination register Rd **and all I-type instructions** with destination register Rt will not work because these instructions will not be able to write their results to the register file.

B) ALUSrc = 0

All I-type instructions with a 16-bit immediate constant will not work because these instructions will not be able to use the 16-bit immediate constant encoded inside the instruction.

C) RegDst = 0

All R-type instructions with destination register Rd will not work because these instructions will not be able to write their results to Rd. Instead, they will write their results to the second source register Rt.

D) Branch = 0

The branch instructions will not work because these instructions will not be able to branch (branch is never taken) even when the branch condition is true.

E) MemWrite = 1

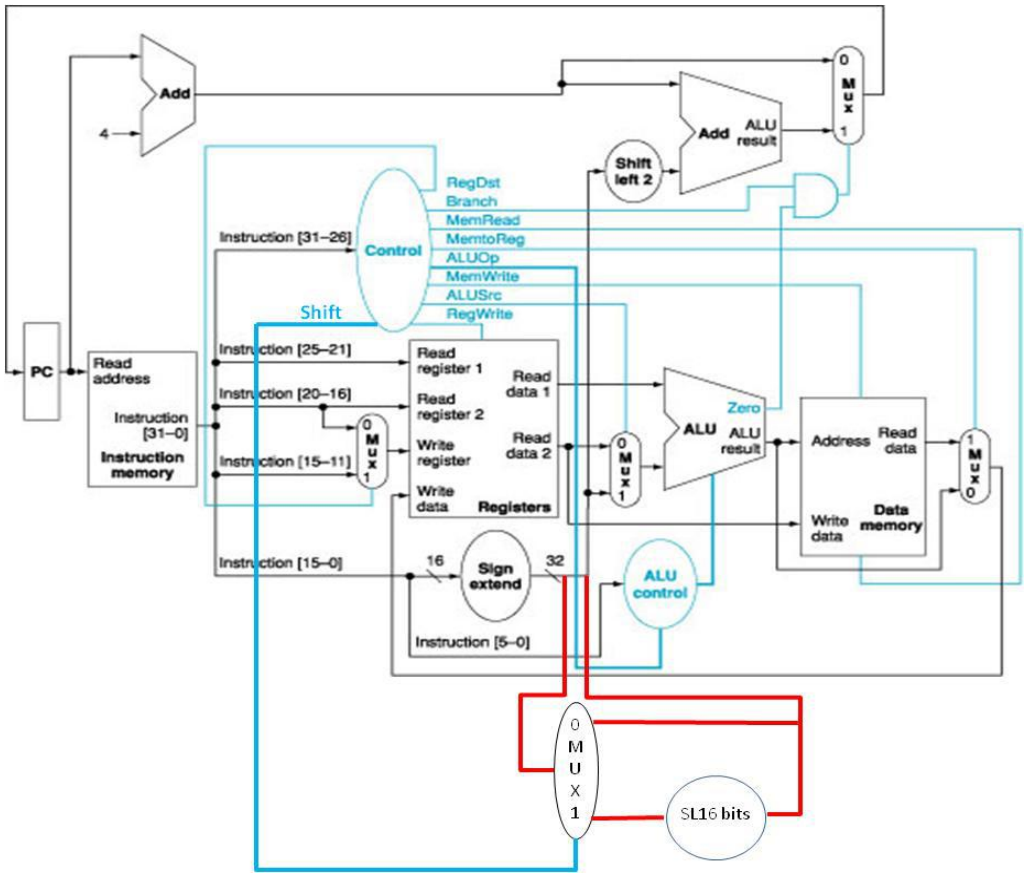
Since only sw will work correctly, thus **the rest of instructions will not work**. To bolster, the rest of instructions will store their results in the data memory, while they should not.

Question 2: *(44 Points)*

A) We wish to add the instruction "*addi*" (Add immediate) and "*lui*" (load upper immediate) to the shown single-cycle simple processor. Add any datapath and control signals and show the value of the control signals while executing the new "*addi*" and "*lui*" instruction. 30 points (15 points for each instruction, *addi* and *lui*)

"*addi*" (Add immediate)

Datapath (7 points): Add shift left 16 bit unit but unused. Since it takes no effect from old datapath, no change on the given datapath is also correct.

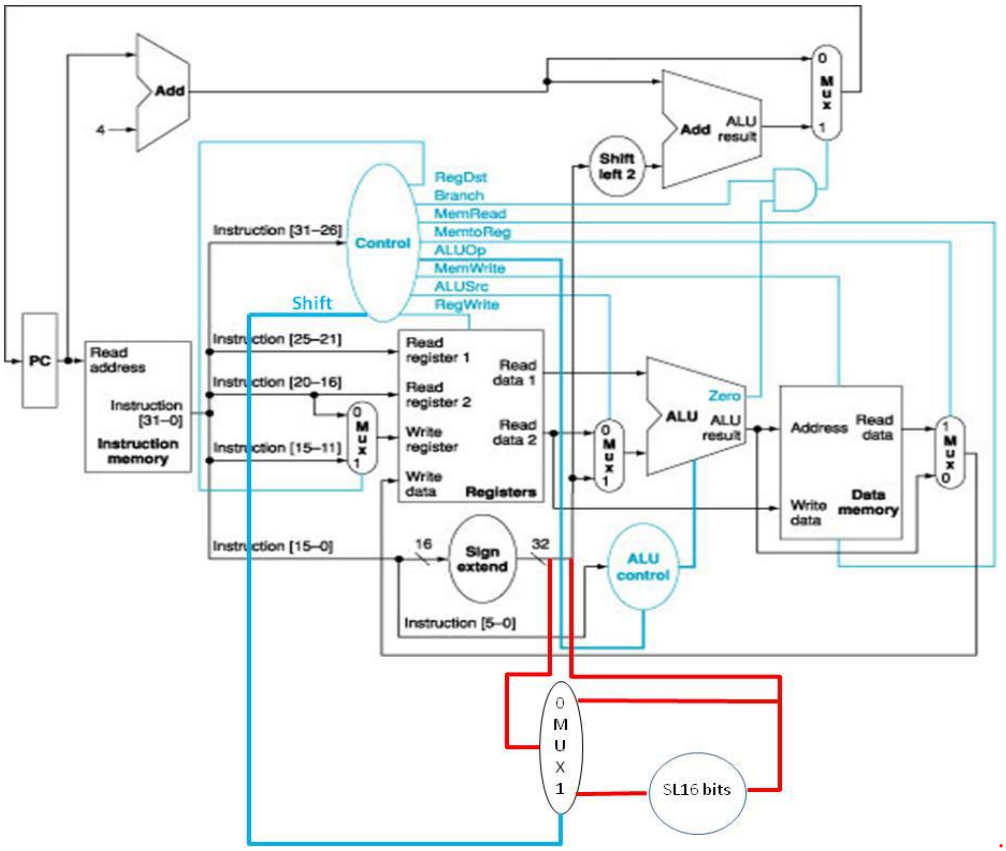


Control signals (8 points):

RegDst = 0	Write back to $rt
Branch = 0	No branch
MemRead = 0	No memory read
MemToReg = 0	Data from ALU
ALUOp = 00	Add shifted immediate value
MemWrite = 0	No memory write
ALUSrc = 1	Feed immediate value into ALU
RegWrite = 1	Write result into register
Shift = 0	**Get immediate value for addi**

***"lui"* (Load upper immediate)**

Datapath (7 points): Same as addi but different in control signal



Control signals (8 points):

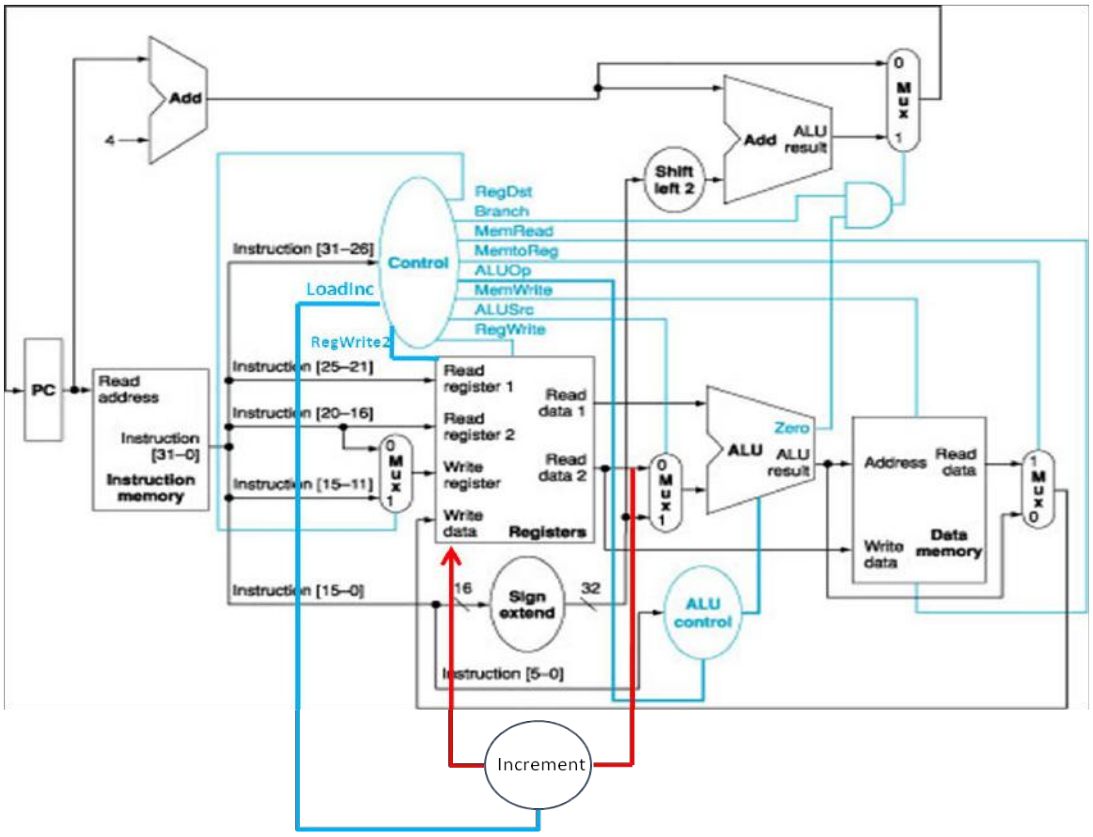
RegDst = 0	Write back to $rt
Branch = 0	No branch
MemRead = 0	No memory read
MemToReg = 0	Data from ALU
ALUOp = 00	Add shifted immediate value
MemWrite = 0	No memory write
ALUSrc = 1	Feed immediate value into ALU
RegWrite = 1	Write result into register
Shift = 1	**Shift left 16 bit for lui**

B) This question is similar to part "A" except that we wish to add a variant of the "*lw*" (load word) instruction, which increments the index register after loading word from memory. This instruction ("*l_inc*") corresponds to the following two instructions: 14 points

```
lw      $rt, L($rs)
addi    $rs, $rs, 1
```

Again add any datapath and control signals and show the value of the control signals while executing the new instruction.

Datapath (7 points):



Control signals (7 points):

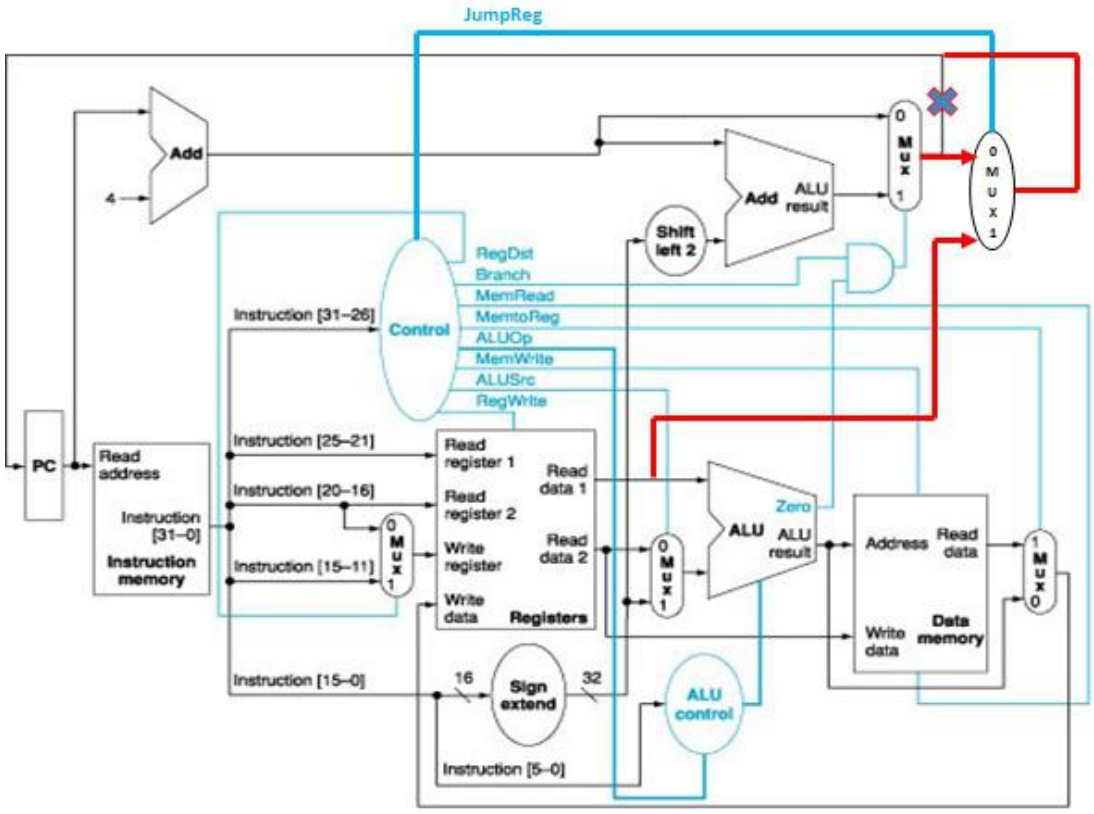
RegDst = 0	Write back to $rt
Branch = 0	No branch
MemRead = 0	No memory read
MemToReg = 0	Data from ALU
ALUOp = 00	Add shifted immediate value
MemWrite = 0	No memory write
ALUSrc = 1	Feed immediate value into ALU
RegWrite = 1	Write result into register
RegWrite2 = 1	**Write incremented $rs**
LoadInc = 1	**Increment $rs**

In order to correctly perform **LoadInc** and **RegWrite2**, we need to extend/modify the register file (the square box in the middle) to have dual write ports for writing two registers (no extra hardware to the datapath, just the additional function of register file, thus you need to mention). To bolster, we need to allow $rs and $rt of *lw* and *addi* being written. It is mandatory.

Question 3: *(26 Points)*

We wish to add the instruction "*jr* (jump register)" to the single-cycle datapath shown in question 1. The instruction loads the program counter with the value stored in the specified register. For example, the effect of "*jr* $r1" can be summarized as PC ← [$r1]. Add any necessary datapaths and control signals and show the value of these control signals when executing all instructions supported by the new datapath. (You can photocopy the figure to make it faster to show the additions.)

Datapath (13 points):



Control signals (13 points):

RegDst = 0	Not writing to register, avoid problems
Branch = 0/X	No branch or no effect when **JumpReg** is 1
MemRead = 0	No read from memory
MemToReg = X	Don't care
ALUOp = X	Don't care
MemWrite = 0	No memory write
ALUSrc = X	Don't care
RegWrite = 0	Not writing to register, if 1 it will overwrite a register
JumpReg = 1	**Get the value of new PC from register**