Sabbir Ahmed
DATE: October 16, 2017
CMSC 411 HW 03

1. **Question** 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.

   Consider each of the following faults separately:

   (a) RegWrite = 0
   (b) ALUSrc = 0
   (c) RegDst = 0
   (d) Branch = 0
   (e) MemWrite = 1

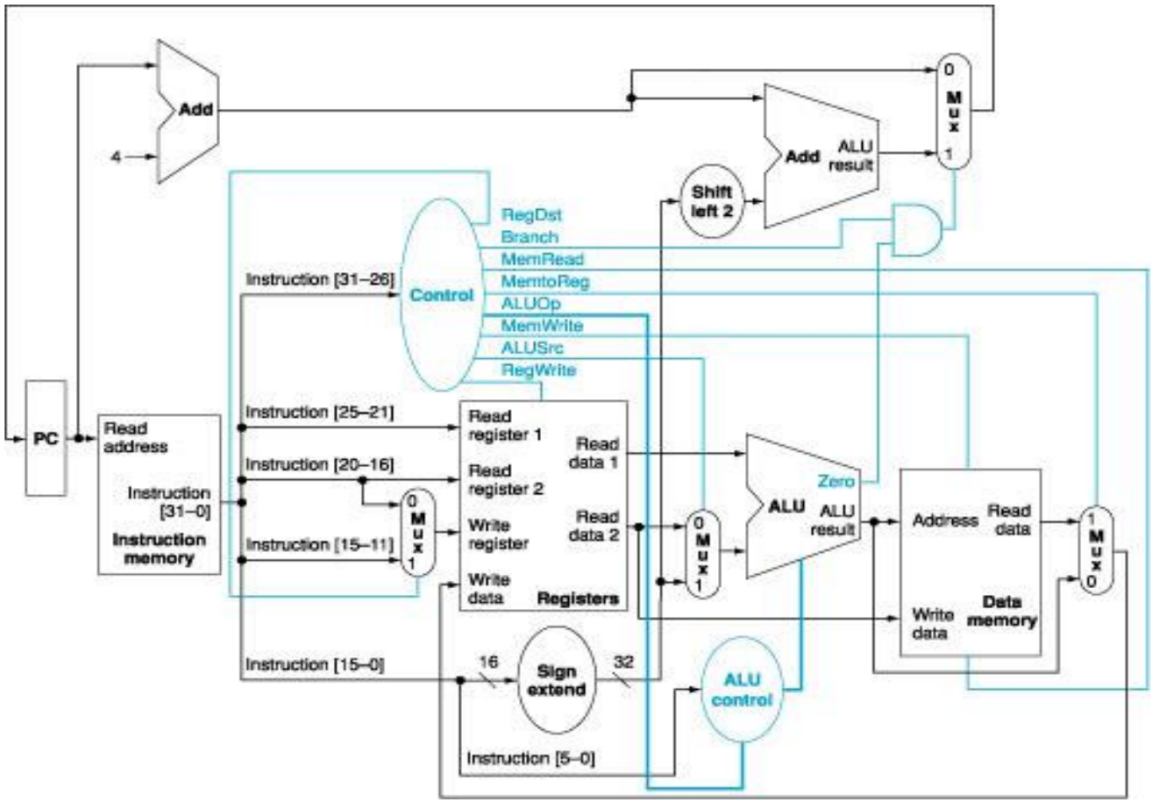

Figure 1: Single-Cycle Datapath of the Instructions Described

Answer

(a) RegWrite = 0 stuck-at fault would affect the ADD/SUB, ORI and LW instructions. The signal enables the destination register to be written to. The ADD/SUB/ORI instructions perform computations between two registers or a register and an immediate value,

and writes the value to the destination register. The LD instruction writes values to destination registers.

(b) ALUSrc = 0 stuck-at fault would affect the ORI, LW and SW instructions. These instructions handle immediate values, and would therefore not receive them if the proceeding ALU is only expecting outputs from two registers.

(c) RegDst = 0 stuck-at fault would affect the ADD and SUB instructions. The signal enables both the rt and rd registers to contribute their values as required by R-type instructions.

(d) Branch = 0 stuck-at fault would only affect the BEQ instruction, since the branch signal would not be enabled to let the program counter know.

(e) MemWrite = 1 stuck-at fault would affect the ADD/SUB, ORI, LW and BEQ instructions. The signal enables writing to memory which is not required by any instructions besides SW.

2. (a) **Question** 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.

Answer The ADDI instruction would not require a separate datapath. The instruction would be identical to ADD, with the only differences being in the following control signals:

- RegDst = 0
- ExtOp = 1
- ALUSrc = 1

These are required to enable the immediate value to pass through to the ALU for addition.

The LUI instruction would utilize the same signals as LW, but would require an extra shift register. The shift register would shift-right the immediate value by 16 bits before inputting into the ALU. The updated datapath is as follows:

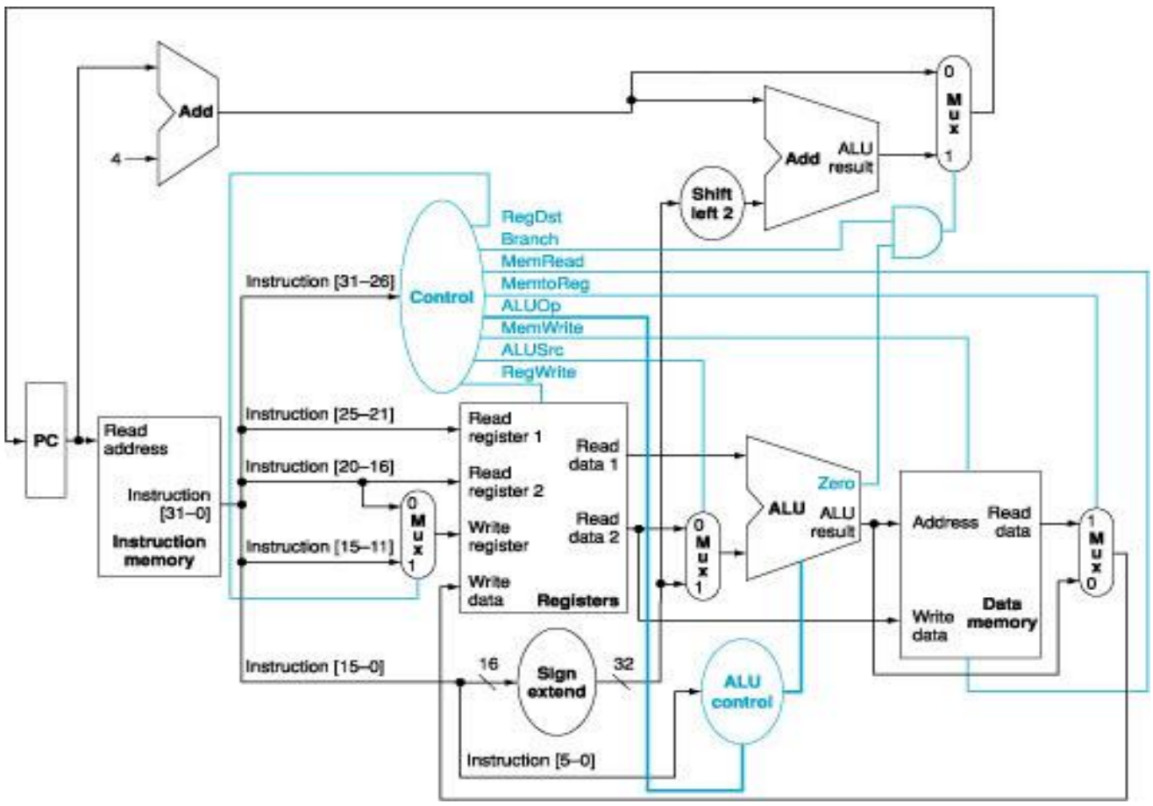

Figure 2: Adding the Datapath for LUI For Question 2A

(b) **Question** 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:

LW $rs, L($rt)
ADDI $rt, $rt, 1

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

Answer The only difference in the control signal values between LW and ADDI is MemToReg. Therefore, the MemToReg signal could be set to 1, and another 1-bit adder enabled by the L_INC instruction may be added before the input goes into the multiplexer. The updated datapath is as follows:

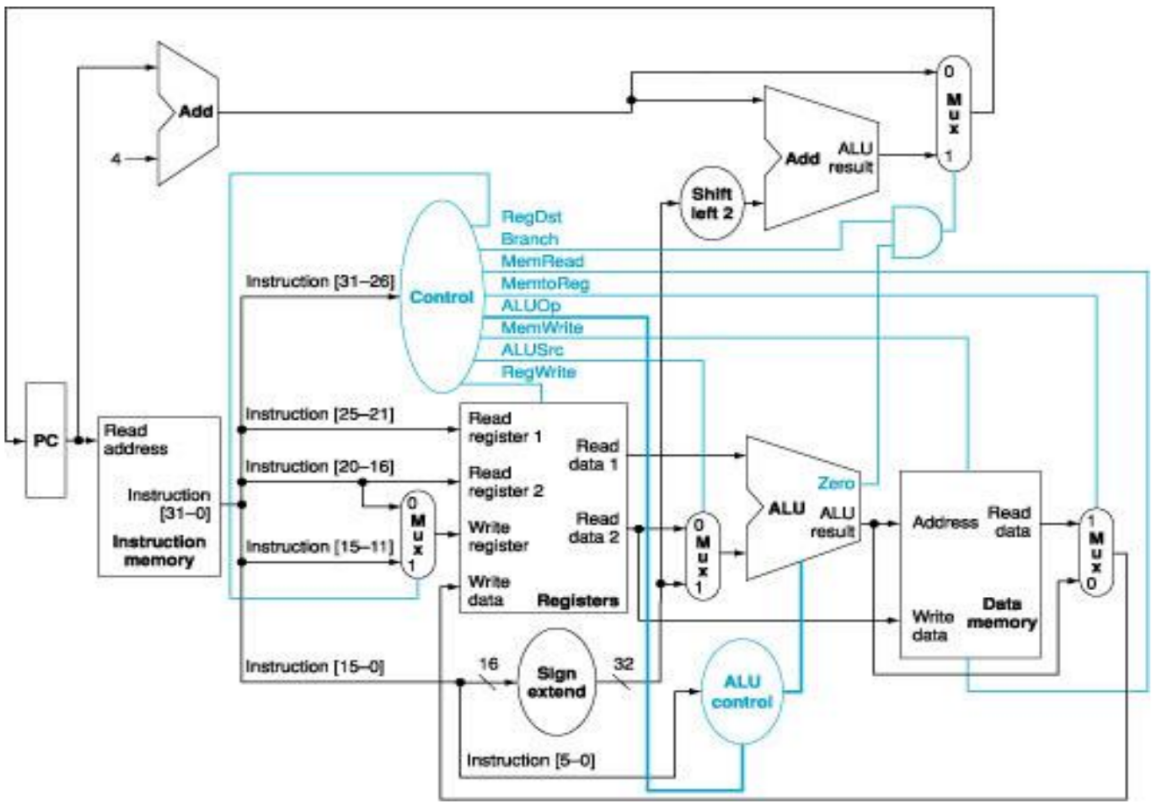

Figure 3: Adding the Datapath for L_INC For Question 2B

3. **Question** 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 $\leftarrow$ ($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.

   Answer The JR instruction would require to load the value from the register, and then increment the program counter with it. This instruction may be implemented by directing the output of the LW instruction directly to the Instruction Fetch Unit. This would increment the program counter similarly to the BEQ instruction; where instead of the PC being equal to PC + 4 + SignExt(imm16)*4 it would equal PC + 4 + ($r1)*4. The JR control signal would feed into a separate multiplexer proceeding the one following the data memory bank, and would direct the value loaded to the Instruction Fetch Unit. The updated datapath is as follows:

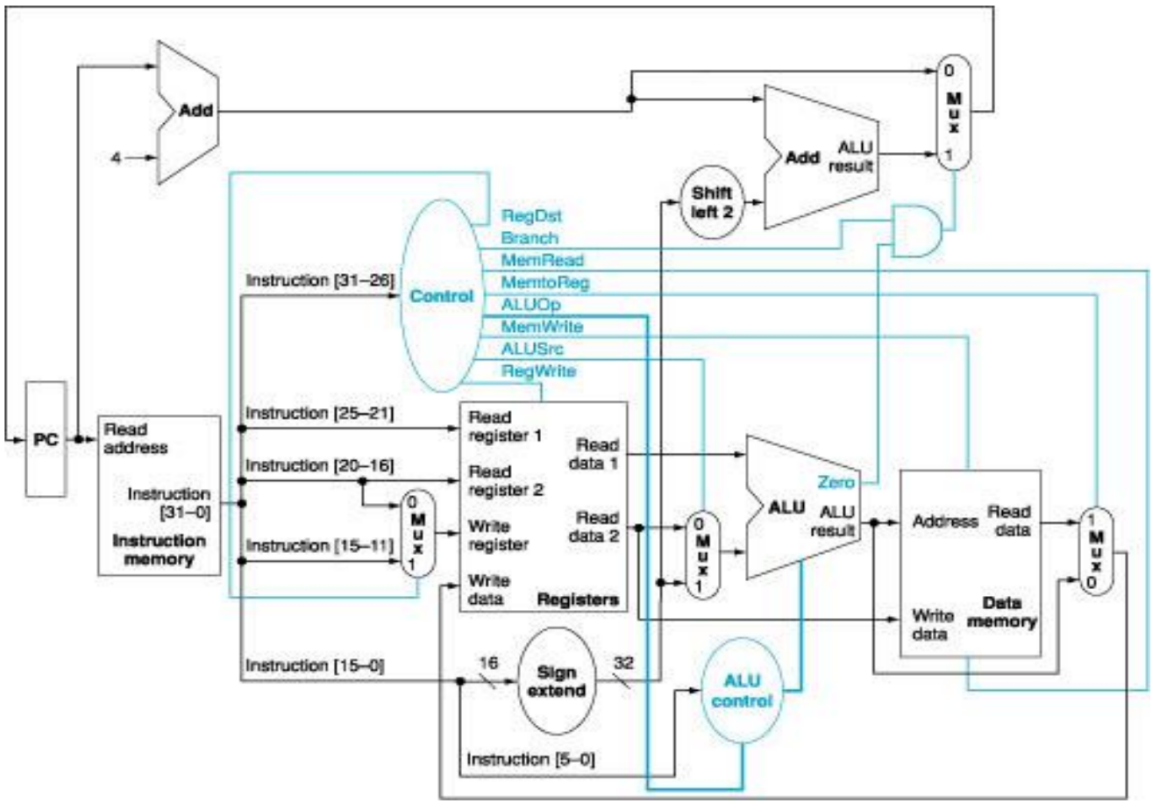

Figure 4: Adding the Datapath for JR For Question 3