ICS 233, Term 072

Computer Architecture & Assembly Language

HW# 6 Solution

Q.1. Describe the effect that a single stuck-at-0 fault (i.e., the signal is always 0 regardless of what it should be) would have for the signals shown below, in the singlecycle Datapath. Which instructions, if any, will not work correctly? Explain why. Consider each of the following faults separately:

(i) RegWrite = 0

(ii) RegDst = 0

(iii) ALUSrc = 0

(iv) MemtoReg = 0

In order to find the effect of these faults, we need to examine the control table given below:

Op	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	Beq	Bne	J	MemRead	MemWrite	MemtoReg
R-type	1 = Rd	1	x	0=BusB	R-type	0	0	0	0	0	0
Addi	0 = Rt	1	1=sign	1=Imm	ADD	0	0	0	0	0	0
Slti	0 = Rt	1	1=sign	1=Imm	SLT	0	0	0	0	0	0
Andi	0 = Rt	1	0=zero	1=Imm	AND	0	0	0	0	0	0
Ori	0 = Rt	1	0=zero	1=Imm	OR	0	0	0	0	0	0
Xori	0 = Rt	1	0=zero	1=Imm	XOR	0	0	0	0	0	0
Lw	0 = Rt	1	1=sign	1=Imm	ADD	0	0	0	1	0	1
Sw	x	0	1=sign	1=Imm	ADD	0	0	0	0	1	x
Beq	x	0	x	0=BusB	SUB	1	0	0	0	0	x
Bne	x	0	x	0=BusB	SUB	0	1	0	0	0	x
J	x	0	x	x	x	0	0	1	0	0	x

(i) RegWrite = 0

As can be seen from the control table above, all the instructions that will require this signal to be one will not function correctly. This includes the following instructions: R-type instructions, Addi, Slti, Andi, Ori, Xori, Lw.

(ii) RegDst = 0

For this signal, the only instructions affected are the R-type instructions.

(iii) ALUSrc = 0

For this signal, the following instructions are affected: Addi, Slti, Andi, Ori, Xori, Lw, Sw.

(iv) MemtoReg = 0

For this signal, the only instruction affected is the Lw instruction.

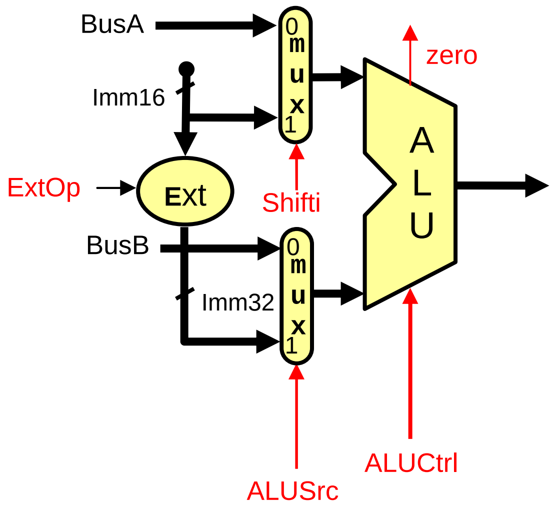
Q.2. Consider the single-cycle datapath. A friend is proposing to modify this single-cycle datapath by eliminating the control signal MemtoReg. The multiplexor that has MemtoReg as an input will instead use either the ALUSrc or the MemRead control signal. Will your friend's modification work? Can one of the two signal (MemRead and ALUSrc) substitute for the other? Explain.

Looking at the control table above, it is clear that the ALUSrc signal cannot be used to replace the signal MemtoReg. However, the signal MemRead can be used to replace the signal MemtoReg as the don't care values can be set to 0's. ALUSrc and MemRead signals cannot substitute for each other since they have different values for a number of instructions including Addi, Slti, Andi, Ori, Xori, Sw.

Q.3. We wish to add the following instructions to the single-cycle datapath. Add any necessary datapath and control signals needed for the implementation of these instructions. Show only the modified and added components to the datapath. Show the values of the control signals to control the execution of each instruction.

(i) Sll

For the Sll instruction, examining the ALU one can see that the shift amont 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. The modified part in the datapath is shown below:



The values of the control signals to control the execution of this instruction are given below:

Op	Shifti	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	Beq	Bne	J	MemRead	MemWrite	MemtoReg
Sll	1	1 = Rd	1	x	0=BusB	SLL	0	0	0	0	0	0

(ii) Sllv

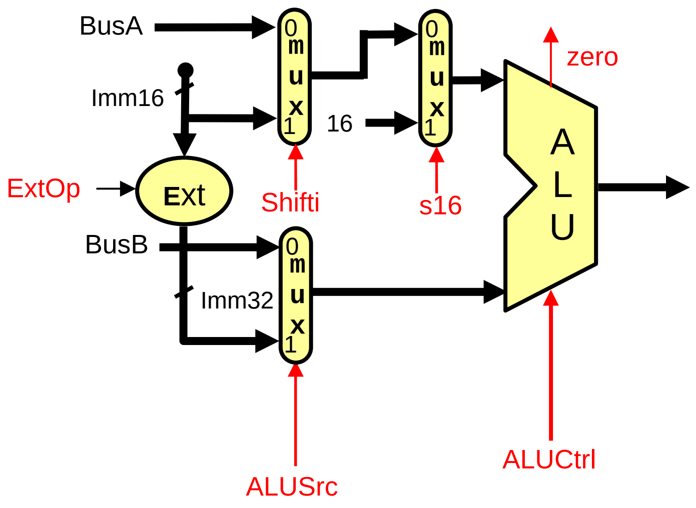
For this instruction, the shift amount is specified by a register determined by the RS field which comes in BusA. Since BusA is already connected to the A-input of the ALU which controls the shift amount, no additional changes are needed in the datapath for the execution of this instruction.

The values of the control signals to control the execution of this instruction are given below:

Op	Shifti	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	Beq	Bne	J	MemRead	MemWrite	MemtoReg
Sllv	0	1 = Rd	1	x	0=BusB	SLL	0	0	0	0	0	0

(iii) 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:

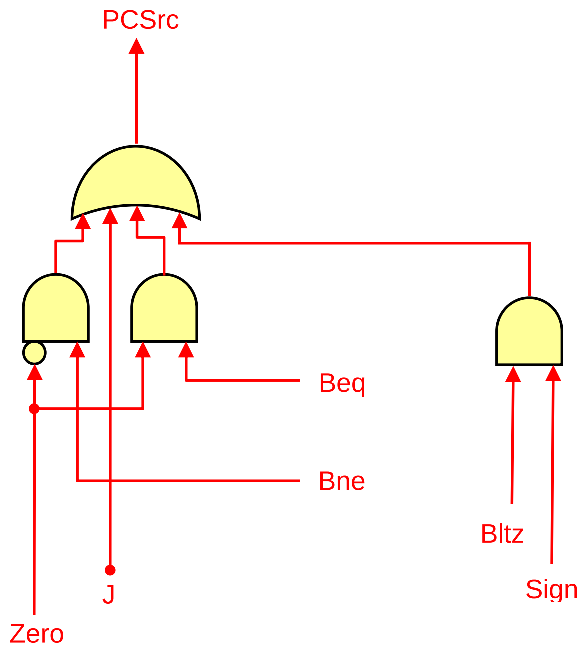


The values of the control signals to control the execution of this instruction are given below:

Op	S16	Shifti	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	Beq	Bne	J	MemRead	MemWrite	MemtoReg
lui	1	x	1 = Rd	1	x	1=Imm	SLL	0	0	0	0	0	0

(iv) Bltz

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. If the sign bit is 1, then it is less than zero. Thus, we only need to check the sign bit. The changes needed to be done are in the NextPC block as shown below:

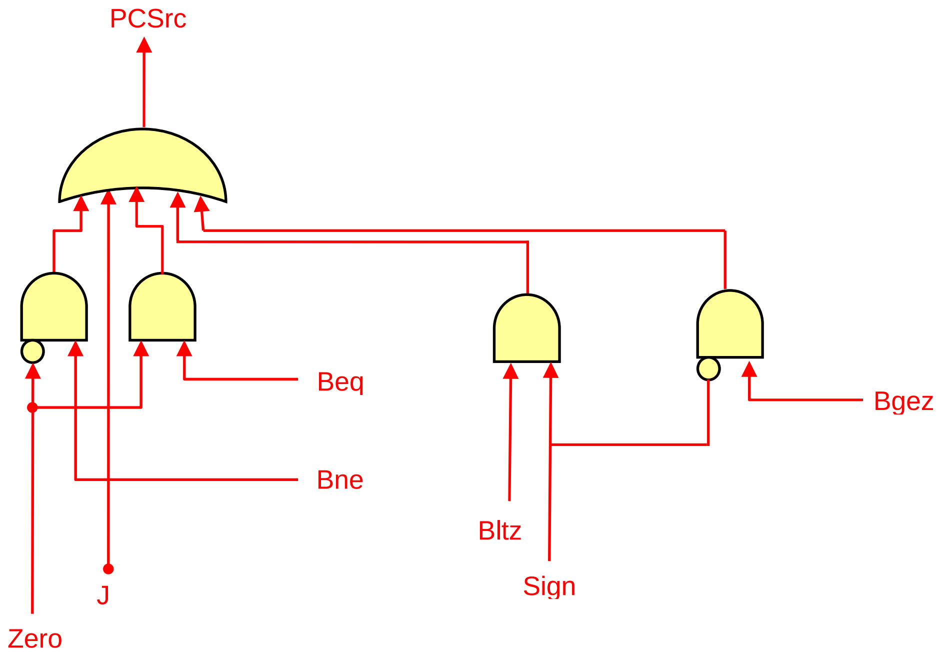


The values of the control signals to control the execution of this instruction are given below:

Op	S16	Shifti	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	Bltz	Beq	Bne	J	MemRead	MemWrite	MemtoReg
Bltz	0	0	x	0	x	0= BusB	ADD	1	0	0	0	0	0	x

(v) Bgez

This instruction is similar to the previous instruction and we also need to check the sign bit. If the sign bit is zero this means that the operand is greater than or equal zero. The required changes to the datapath are also in the NextPC block as shown below:



The values of the control signals to control the execution of this instruction are given below:

Op	S16	Shifti	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	Bgez	Bltz	Beq	Bne	J	MemRead	MemWrite	MemtoReg
Bgez	0	0	x	0	x	0= BusB	ADD	1	0	0	0	0	0	0	x

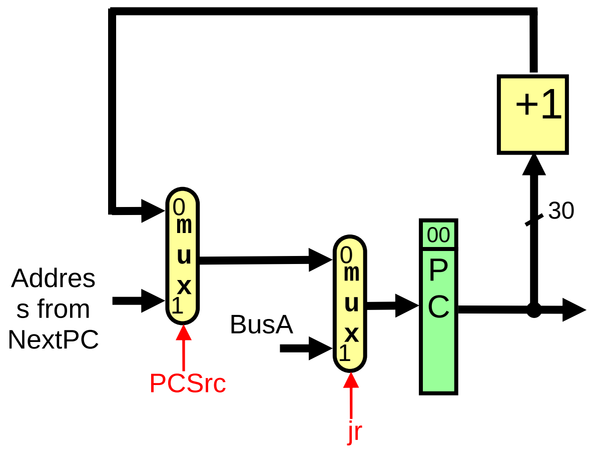
(vi) Slti

This instruction is already supported by the datapath and no changes are needed. The control signals for this instruction are:

Op	S16	Shifti	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	Bgez	Bltz	Beq	Bne	J	MemRead	MemWrite	MemtoReg
Slti	0	0	0 = Rt	1	1=sign	1=Imm	SLT	0	0	0	0	0	0	0	x

(vii) Jr

For this instruction, the changes required in the datapath to implement it is to load the PC from BusA which is driven by the RS field. Thus we need to add a MUX to select the target address to be loaded in the PC either from the output of the MUX choosing between the address from NextPC block and incremented PC or from BusA. The required changes are shown below:



The control signals for this instruction are:

Op	S16	Shifti	RegDst	RegWrite	ExtOp	ALUSrc	ALUOp	jr	Bgez	Bltz	Beq	Bne	J	MemRead	MemWrite	MemtoReg
jr	x	x	x	0	x	x	x	1	0	0	0	0	0	0	0	x

Q.4. We want to compare the performance of a **single-cycle CPU design** with a **multicycle CPU**. Suppose we add the multiply and divide instructions. The operation times are as follows:

Instruction memory access time = 190 ps, Data memory access time = 190 ps
Register file read access time = 150 ps, Register file write access = 150 ps
ALU delay for basic instructions = 190 ps, Delay for multiply or divide = 550 ps

Ignore the other delays in the multiplexers, control unit, sign-extension, etc.

Assume the following instruction mix: 30% ALU, 15% multiply & divide, 30% load & store, 15% branch, and 10% jump.

(i) What is the total delay for each instruction class and the clock cycle for the single-cycle CPU design?

Instruction Class	Instruction Memory	Register Read/Decoding	ALU Operation	Data Memory	Register Write	Total
ALU	190	150	190		150	680 ps
Load	190	150	190	190	150	870 ps
Store	190	150	190	190		720 ps
Branch	190	150	190			530 ps
Jump	190	150				340 ps
Mul/div	190	150	550		150	1040 ps

Clock cycle = 1040 ps determined by the longest delay.

(ii) Assume we fix the clock cycle to 200 ps for a multi-cycle CPU, what is the CPI for each instruction class and the speedup over a fixed-length clock cycle? Note that this implies that multiply and divide operations will be performed in multiple cycles.

Instruction Class	CPI
ALU	4
Load	5
Store	4
Branch	3
Jump	2
Mul/div	6

Average CPI= 4*0.3 + 5*0.15 + 4*0.15 + 3*0.15 + 2*0.1 + 6*0.15=4.1
Note that we assumed that load and store instructions have equal percentage.

Speedup = 1040 ps / (4.1*200 ps) = 1.268