

Question 1: Performance (30 points)

a) (5 points) The three principle components of runtime are Instruction Count, CPI, and Clock Period. Briefly define them and describe how they combine to yield runtime.

Suppose that you have been running an important program on your company's 300MHz Acme II processor. By running a detailed simulator, you were able to collect the following instruction mix and breakdown of costs for each instruction type:

Instruction Class	Frequency (%)	Cycles
Integer arithmetic and logical	40	1
Load	20	1
Store	10	2
Branches	20	3
Floating Point	10	5

b) (5 points) What is the CPI and MIPS rating of the Acme II for this program?

c) (5 points) Suppose that you turn on the optimizer and it eliminates 30% of the arithmetic/logic instructions (i.e. 12% of the *total* instructions), 30% of load instructions, and 20% of the floating-point instructions. None of the other instructions are affected. What is the speedup of the optimized program? (Be sure to state the formula that you are using for speedup and show your work)

d) (5 points) What is the CPI and MIPS rating with the optimized version of the program? Compare your result to that of (1c) and explain the difference:

e) (5 points) Now, suppose that the Acme III has just been introduced with a faster clock rate (450 MHz). However, in order to make the clock rate faster, the Acme engineers had to increase the CPI for arithmetic, logical, and load instructions to 2 cycles and floating point instructions to 6 cycles. What is the speedup of the Acme III over the Acme II on the *unoptimized* program? Show your work.

f) (5 points) The engineers for Acme Inc are currently working on the Acme IV. Instead of increasing the clock rate again, they are working on reducing the time for the floating-point instructions. Use Amdahl's law to show the *maximum* speedup that you could expect between the Acme III and Acme IV on the *unoptimized* program (if the clock rates are both 450 MHz)?

Question 2: Instruction Set Architecture (20 points)

Imagine a modified MIPS instruction set with a register file consisting of 64 general purpose registers rather than the usual 32. Assume that we still want to use a uniform instruction length of four bytes (32 bits) and that the total number of opcodes must remain unchanged. Also assume that you can expand and contract fields in an instruction but that you cannot omit them. For reference, here are the formats for the three MIPS instruction types (assuming 32 32 bit register file).

R-type

Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Rd (5 bits)	Shift amount (5 bits)	Function (6 bits)
--------------------	----------------	----------------	----------------	--------------------------	----------------------

I-type

Opcode (6 bits)	Rs (5 bits)	Rt (5 bits)	Immediate (16 bits)
--------------------	----------------	----------------	------------------------

J-type

Opcode (6 bits)	Target address (26 bits)
--------------------	-----------------------------

a) (5 points) How would the format of R-type (arithmetic and logical instructions) change? Label all the fields with their name and bit length. What is the consequence of this change?

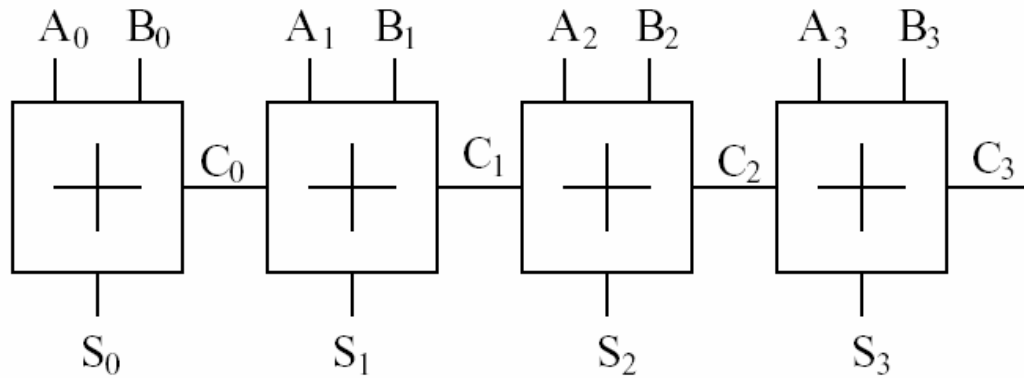
b) (5 points) How does this change the I-type instructions? What is the consequence of this change?

c) (5 points) How does this change the J-type instructions? What is the consequence of this change?

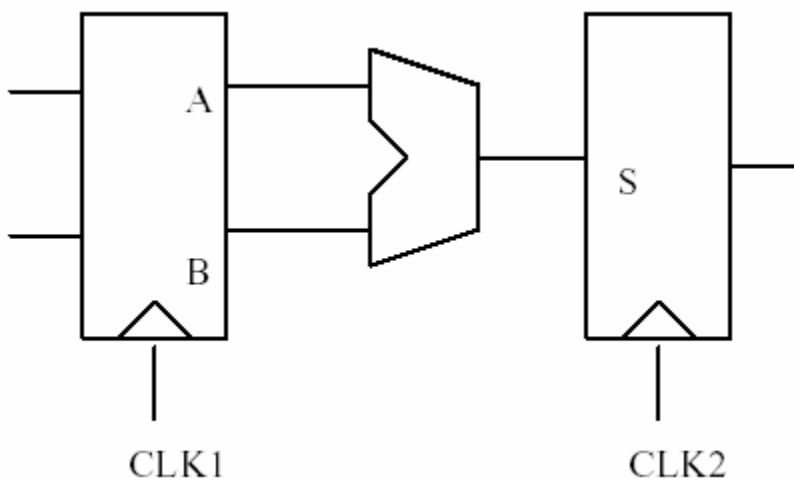
d) (5 points) Imagine we are translating machine code to use the larger register set? Give an example of an instruction that used to fit into the old format but is impossible to translate directly into a single instruction in the new format. Write a short sequence of instructions that could replace it.

Question 3: Arithmetic (25 points)

Suppose we have the following 4 bit adder



with the following setup:



The one-bit full adder blocks have the following delay properties:

$A_i \rightarrow S_i = B_i \rightarrow S_i = 6\text{ns}$, $A_i \rightarrow C_i = B_i \rightarrow C_i = 3\text{ns}$, $C_{i-1} \rightarrow S_i = 3\text{ns}$, $C_{i-1} \rightarrow C_i = 2\text{ns}$

The registers have the following properties: $t_{\text{clk-to-q}}=3\text{ns}$, $t_{\text{setup}} = 2\text{ns}$, $t_{\text{hold}}=1\text{ns}$

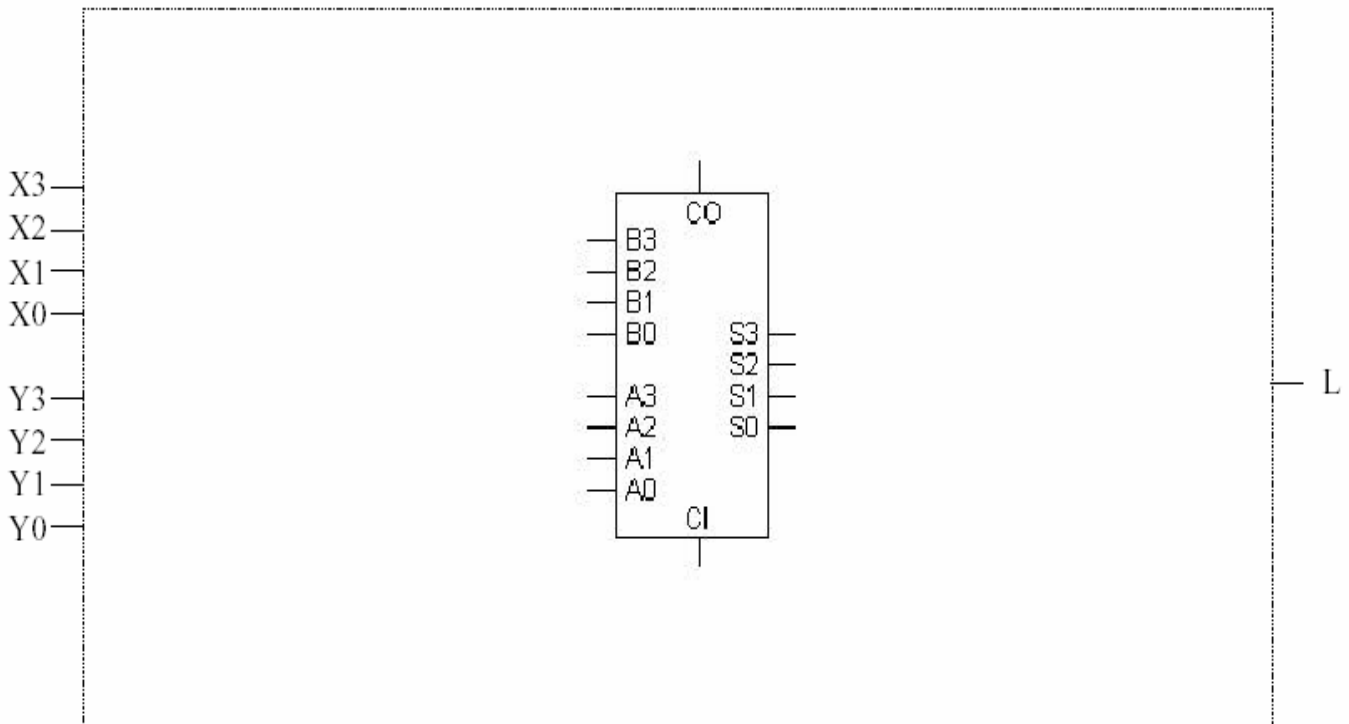
(a) (5 points) What is the maximum clock frequency we can run this circuit at, assuming $\text{CLK1}=\text{CLK2}$ (no clock skew?)

b) (5 points) If there is a clock skew of 3 ns with CLK2 later from CLK1, what is the maximum clock frequency?

We want to build a comparison circuit for four-bit signed two's complement numbers. The circuit inputs will be X and Y. The output is a single bit L, which should be 1 if $X < Y$ (strictly less) and 0 otherwise.

c) (15 points) Show how to build this comparator around an unsigned four-bit adder. For full credit, your solution must account for the possibility of overflow. Your library elements include AND, OR, and inverter gates.

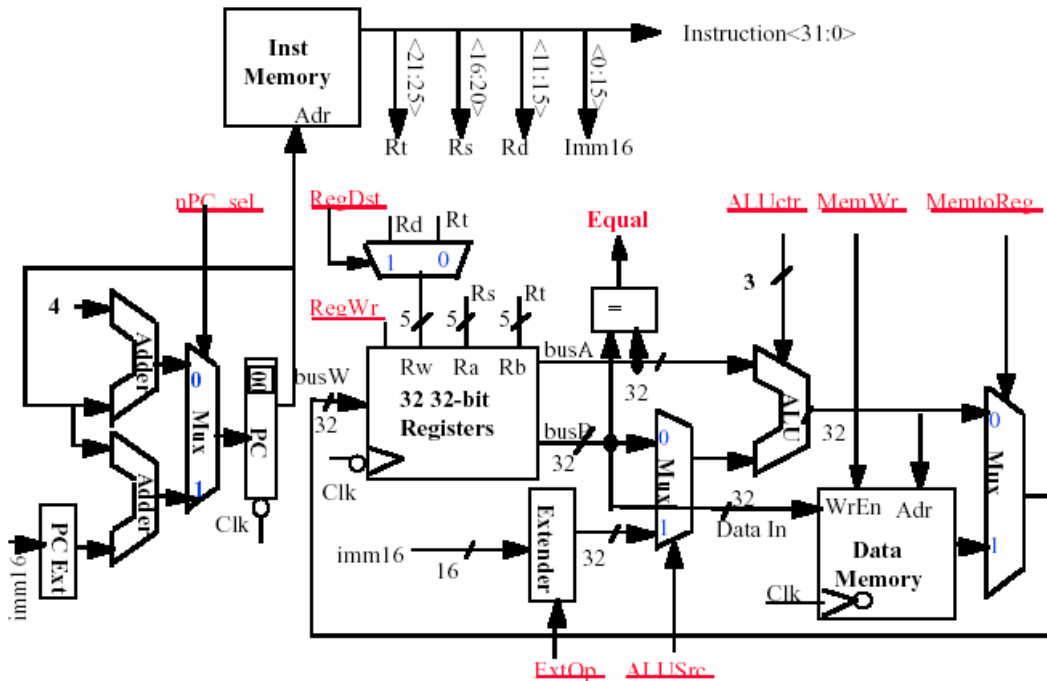
Remember to show your work (i.e. describe your reasoning for how you design the circuit)!



Question 4: Single Cycle (25 points)

The following single cycle datapath is capable of implementing six instructions:

Instruction	Register Transfers
ADDU	$R[rd] \leftarrow R[rs] + R[rt]; PC \leftarrow PC + 4$
SUBU	$R[rd] \leftarrow R[rs] - R[rt]; PC \leftarrow PC + 4$
ORI	$R[rt] \leftarrow R[rs] + \text{zero_ext}(\text{Imm16}); PC \leftarrow PC + 4$
LOAD	$R[rt] \leftarrow \text{MEM}[R[rs] + \text{sign_ext}(\text{Imm16})]; PC \leftarrow PC + 4$
STORE	$\text{MEM}[R[rs] + \text{sign_ext}(\text{Imm16})] \leftarrow R[rs]; PC \leftarrow PC + 4$
BEQ	$\text{if } (R[rs] == R[rt]) \text{ then } PC \leftarrow PC + \text{sign_ext}(\text{Imm16}) \parallel 00 \text{ else } PC \leftarrow PC + 4$



Possible control signals:

ALUctr = "ADD", "SUB", "OR", "AND"

ExtOp = "SIGN", "ZERO"

nPC_sel, RegDst, ALUSrc, MemtoReg = "0", "1"

MemWr, RegWr = "1" for write enable, "0" for write disable

We wish to add the following instructions to our subset:

ADDIU – add immediate unsigned

OR

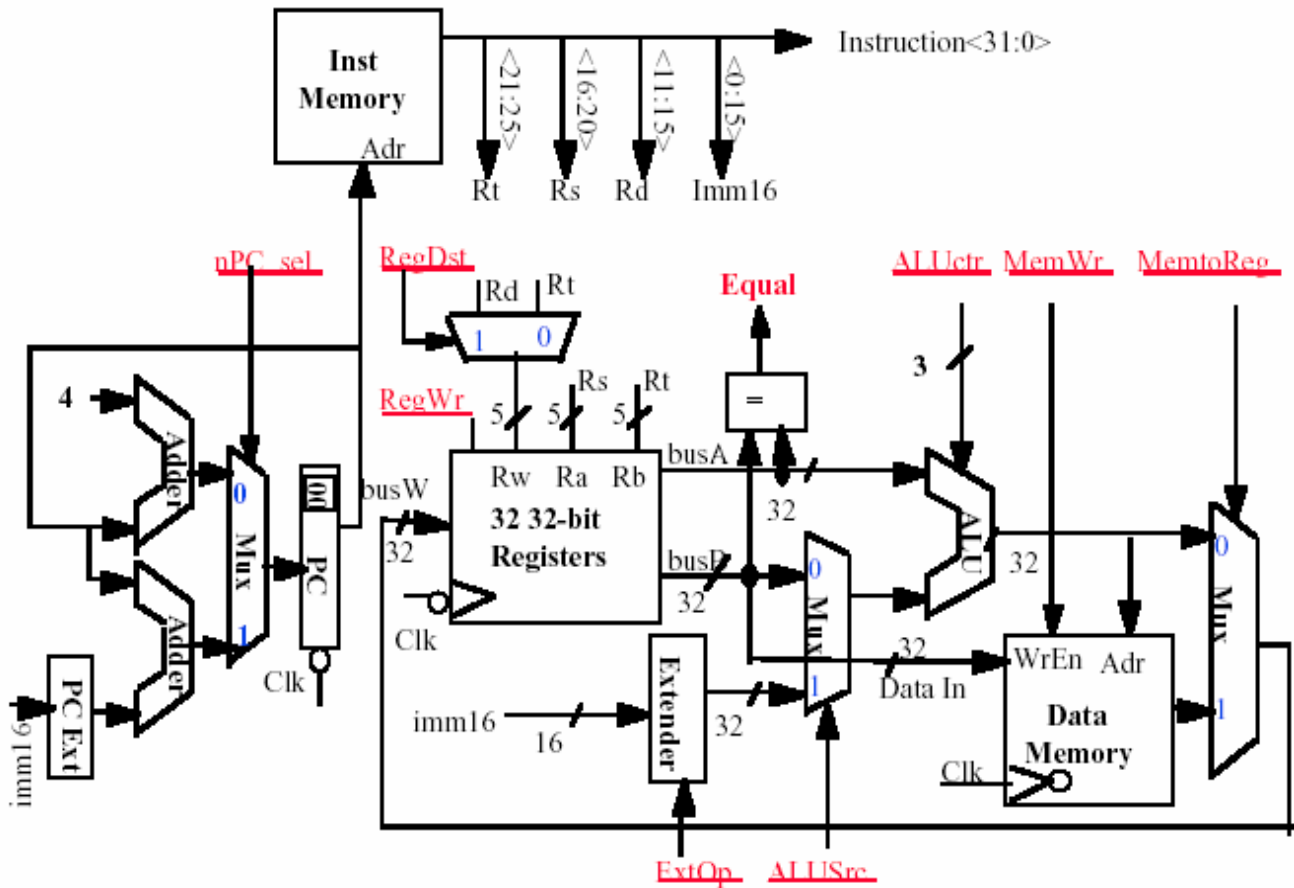
AND

BLTZAL (Branch on less than zero and link) – conditionally branch the number of instructions specified by the offset if register Rs is less than 0. Save the address of the next instruction in register 31.

a) (5 points) Write the register transfers for each of the new instructions:

Instruction	Register Transfers
ADDIU	
OR	
AND	
BLTZAL	

b) (10 points) Sketch the modifications to the datapath that are necessary to implement the new instructions. For full credit, you must explain each modification. Additionally, describe the function of any additional control signals that you add.



c) (10 points) Specify the control signals for the new instructions. For full credit, you must specify any don't care conditions (using an "X").

	nPC_Sel	RegDst	RegWr	ExtOp	ALUSrc	ALUctr	MemWr	MemtoReg
ADDIU								
OR								
AND								
BLTZAL								

Additional control signals (that you added to the datapath):

ADDIU				
OR				
AND				
BLTZAL				