Problem 1 (3 parts, 30 points) **Instruction Formats, Etc.**

Part A (9 points) Consider the instruction set architecture below with fields containing zeros.

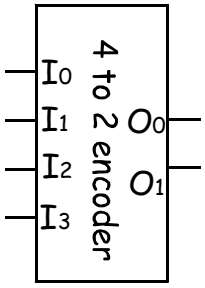
000 0000	00 0000	00 0000	0 0000 0000 0000
opcode	dest. reg.	source 1 reg.	immediate value

What is the maximum number of opcodes? $2^7 = 128$

What is the number of registers? $2^6 = 64$

What is the range of the signed immediate value? $2^{13} = \pm 4K$

Part B (9 points) Suppose the circuit below has the following input priority: $I_1 > I_3 > I_0 > I_2$. Complete the truth table by filling in the input values that would produce the given outputs and derive a simplified expression for O_1.



I_3	I_2	I_1	I_0	V	O_1	O_0
0	0	0	0	0	X	X
0	X	0	1	1	0	0
X	X	1	X	1	0	1
0	1	0	0	1	1	0
1	X	0	X	1	1	1

$O_1 = \overline{I}_1(I_3 + \overline{I}_0)$

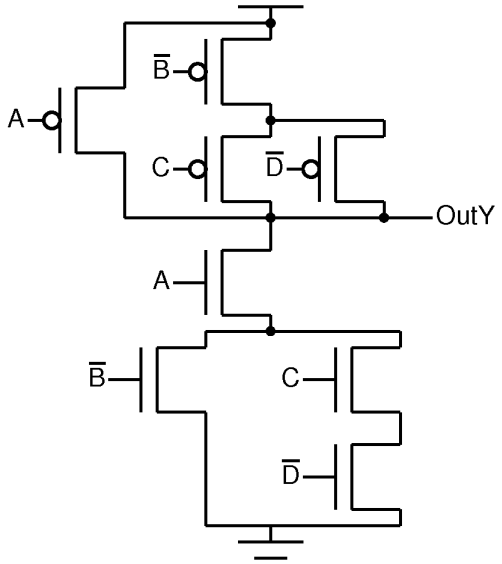
Part C (12 points) For each problem below, compute the operations using the rules of arithmetic, and indicate whether an overflow occurs assuming all numbers are expressed using a **six bit unsigned** and **six bit two's complement** representations.

	101101 + 10101	11010 + 10111	10001 - 111010	110101 - 11110
result	000010	110001	010111	010111
unsigned error?	yes	no	yes	no
signed error?	no	yes	no	yes

Problem 2 (4 parts, 32 points) **Dueling Designs**

For each part implement the specified device. **Label all inputs and outputs.**

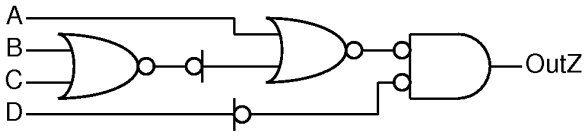
Part A (8 points) Complete the incomplete circuit below using N and P type switches and write the expression for Out_Y.



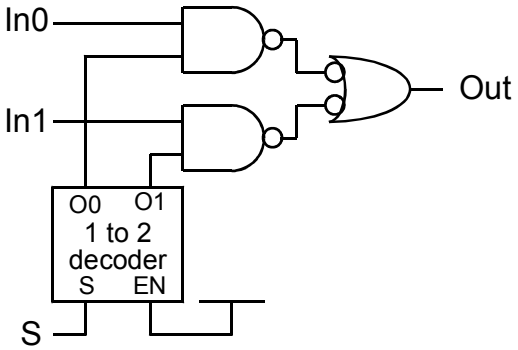
OutY = $\overline{A}$+B($\overline{C}$+D)

Part B (8 points) Implement the expression in mixed logic notation using NOR gates.

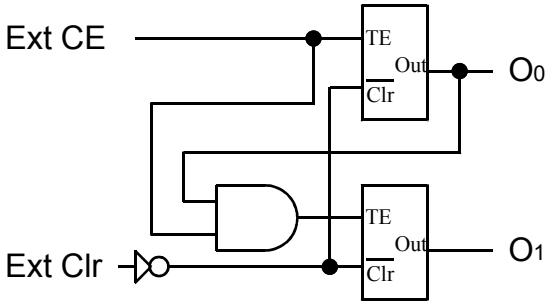
$$OUT_Z = \overline{(A + \overline{B + C}) \cdot \overline{D}}$$



Part C (8 points) Implement a 2 to 1 MUX using a 1 to 2 decoder and basic gates (AND, OR, NAND, NOR, NOT, & XOR).



Part D (8 points) Implement a divide by four counter using toggle cells and minimum additional basic gates.



Problem 3 (3 parts, 26 points) **Microcode**

Using the supplied datapath, write microcode fragments to accomplish the following procedures. Express all values, except memory addresses, in hexadecimal notation. Use 'X' when a value is don't cared. For maximum credit, complete the description field. **In each part, modify only registers 7 & 8.**

Part A (5 points) $R_7 = (R_8 - 15)/512$

#	X	Y	Z	rwe	im en	im va	au en	s/$\overline{a}$	lu en	lf	su en	st	ld en	st en	r/$\overline{w}$	msel	description
1	8	x	7	1	1	F	1	1	0	x	0	x	0	0	x	0	R7 = R8 - 15
2	7	x	7	1	1	9	0	x	0	x	1	1	0	0	x	0	R7 = R7 >> 9
3																	

Part B (15 points) Compute mem[4000] ⊕ R_3 and store the result in mem[4004]. ⊕ **means bitwise logical XOR.**

#	X	Y	Z	rwe	im en	im va	au en	s/$\overline{a}$	lu en	lf	su en	st	ld en	st en	r/$\overline{w}$	msel	description
1	x	x	7	1	1	4000	0	x	1	C	0	x	0	0	x	0	R7 = 4000
2	7	x	8	1	0	x	0	x	0	x	0	x	1	0	1	1	R8 = Mem(R7)
3	8	3	8	1	0	x	0	x	1	6	0	x	0	0	x	0	R8 = R8 ⊕ R3
4	7	x	7	1	1	4	1	0	0	x	0	x	0	0	x	0	R7 = R7 + 4
5	7	8	x	0	0	x	0	x	0	x	0	x	0	1	0	1	Mem(R7) = R8
6																	

Part C (6 points) $R_7 = 18 \cdot R_8$ (multiply R_8 by 18)

#	X	Y	Z	rwe	im en	im va	au en	s/$\overline{a}$	lu en	lf	su en	st	ld en	st en	r/$\overline{w}$	msel	description
1	8	x	7	1	1	FFFC	0	x	0	x	1	1	0	0	x	0	R7 = R8 << 4
2	8	x	8	1	1	FFFF	0	x	0	x	1	1	0	0	x	0	R8 = R8 << 1
3	7	8	7	1	0	x	1	0	0	x	0	x	0	0	x	0	R7 = R7 + R8
4																	

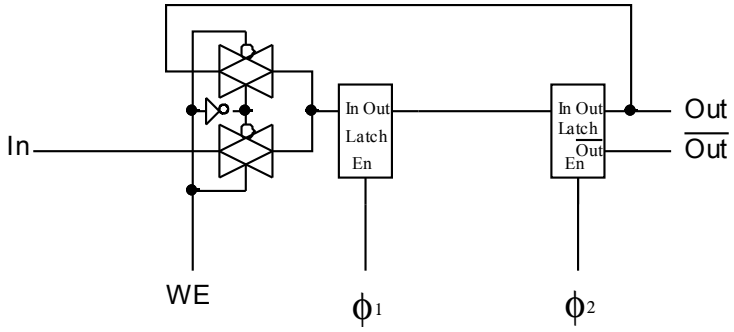
Problem 4 (3 parts, 24 points) **Storage**

Part A (12 points) Consider a **4 Gbit** DRAM chip organized as **64 million addresses** of **64-bit words**. Assume both the DRAM cell and the DRAM chip are square. The column number and offset concatenate to form the memory address. Using the organization approach discussed in class, answer the following questions about the chip. *Express all answers in decimal (not powers of two)*.

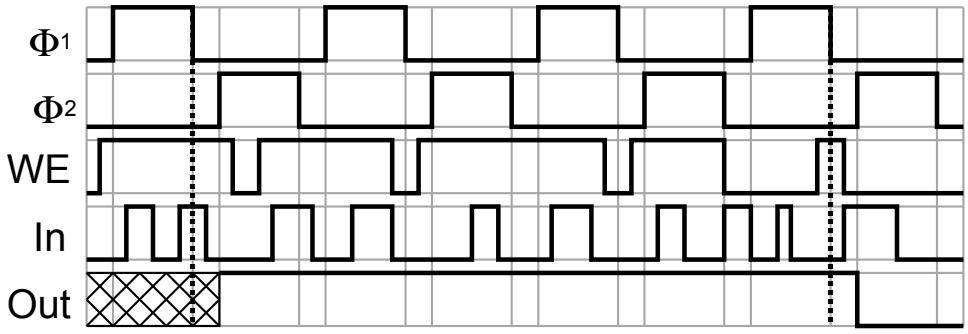
total number of bits in address	log2(64M) = 26
number of columns	sqrt(4G) = sqrt(2^{32}) = 2^{16} = 64K
column decoder required (*n* to *m*)	16 to 64K
number of words per column	64K / 64 = 1K
type of mux required (*n* to *m*)	1K to 1
number of address lines in column offset	$\log_2$(1K) = 10

Part B (6 points) Implement a register below using *only* latches, pass gates, and inverters (all in icon form). Complete the behavior table at right. Recall that the CLK signal indicates a full Φ_1 Φ_2 cycle; so the output should be the value at the end of a cycle (for the given inputs).

In	WE	Clk	Out	$\overline{\text{Out}}$
A	0	↑↓	Q_o	$\overline{Q}_o$
A	1	↑↓	*A*	$\overline{A}$



Part C (6 points) Assume the following signals are applied to a register with write enable. Draw the output signal **Out**. Draw a vertical line where **In** is sampled. *Assume **Out** is initially zero.*



Problem 5 (2 parts, 23 points) **Assembly Language Programming**

Part A (15 points) Write a MIPS subroutine `CountX` that reads in a vector of integers and counts the number of elements that have the value X (given in register $1), placing the total count in register $2. Assume the length of the vector (# of integer elements) is stored in register $4 and the base address of the vector is in register $5. *The size of the vector may be 0.* Follow the steps outlined in the comments in the rightmost column below. **You may modify only registers $2 through $5**.

label	instruction	comment
CountX:	**addi $2, $0, 0**	# clear match count ($2 = 0)
	sll $4, $4, 2	# compute end addr: scale vector # length by 4 and add to base address # (2 instructions)
	add $4, $4, $5	
Loop:	**beq $5, $4, Exit**	# if current elem addr = end address, # then exit loop
	lw $3, ($5)	# load current vector elem
	bne $3, $1, Skip	# if current elem != X then Skip
	addi $2, $2, 1	# else increment match count
Skip:	**addi $5, $5, 4**	# inc vector ptr to next elem
	j Loop	# loop back
Exit:	**jr $31**	# return to caller

Part B (8 points) Consider the following code fragment.

address	label	instruction
1000		addi $10, $0, 0
1004	Loop2:	lw $1, ($12)
1008		jal CountX
1012		slt $9, $2, $10
1016		bne $9, $0, Continue
1020		add $10, $2, $0
1024	Continue:	addi $12, $12, 4
1028		...

1. What is the branch offset (in bytes) for the `bne` instruction at 1016? 4

If $10 holds M and $2 holds the result R of subroutine CountX, what simple mathematical expression do instructions at addresses 1012-1020 compute (express your answer in terms of M and R, not registers)?

2. Instructions 1012-1020 compute max R returned from CountX (maintained in M)

What is the value of $31 after the `jal` instruction at 1008 is executed?

3. The value of $31 = 1012