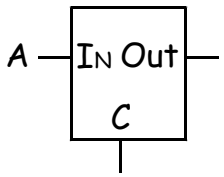
Problem 1 (3 parts, 21 points) "A chip off the old block"

Part A (15 points) Consider the five definitions for the block drawn below. One block input is the logical value **A**. The other input is the control value **C**. The output behavior for each of the five definitions is given in the table. Complete the full truth table and state the *logical (gate) names* for each definition. (hint: the first block one appears to mask **A** when its control input is low.)



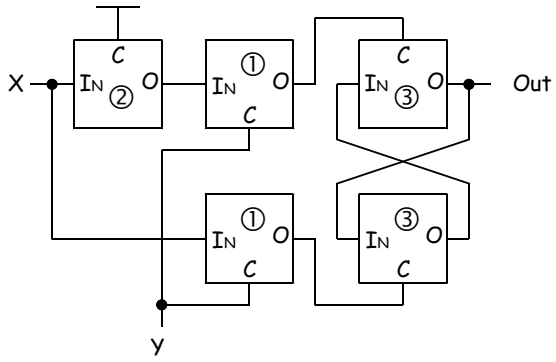
In	C	①	②	③	④	⑤
A	0	0	A	$\overline{A}$	A	Z_o
A	1	A	$\overline{A}$	0	1	A

In	C	①	②	③	④	⑤
0	0	0	0	1	0	Z_o
1	0	0	1	0	1	Z_o
0	1	0	1	0	1	0
1	1	1	0	0	1	1

① AND ② XOR ③ NOR

④ OR ⑤ Pass Gate

Part B (6 points) The circuit below is built using these blocks. Describe its behavior. Also give the circuits common name.



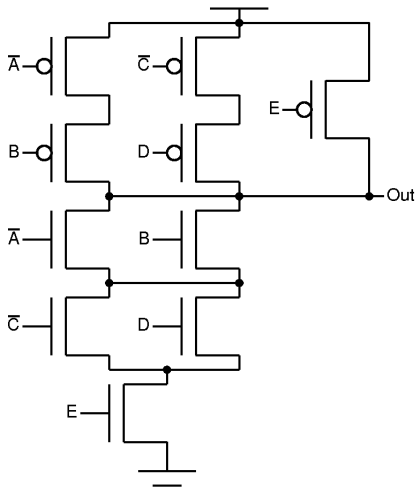
X	Y	Out
0	0	Q_0
1	0	Q_0
0	1	0
1	1	1

It's a transparent latch

Problem 2 (4 parts, 32 points) Design Fun

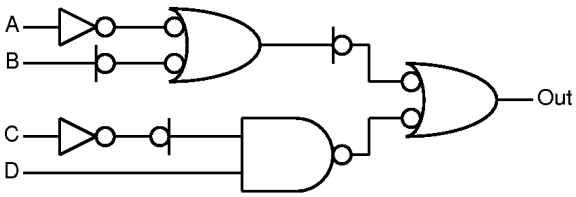
Complete each design below. Be sure to label all signals.

Part A: Complete the following CMOS design. Also express its behavior.



Out = $A \cdot \overline{B} + C \cdot \overline{D} + \overline{E}$

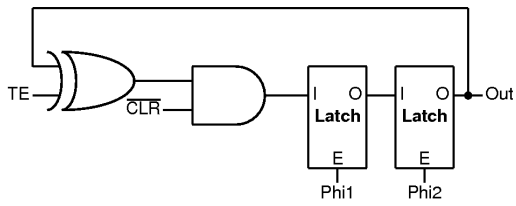
Part B: Derive the proper mixed logic expression for the following design. Determine # of switches needed.



Out = $\overline{\overline{A + \overline{B}} + \overline{C} \cdot D}$

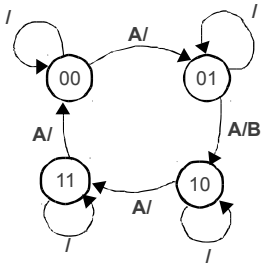
switches = 3 x 4 + 2 x 2 = 12 + 4 = 16T

Part C: Implement a toggle cell using required latches and basics gates (including XORs). Also complete the behavior table.



TE	$\overline{CLR}$	CLK	Out
X	0	↑↓	0
0	1	↑↓	Q_o
1	1	↑↓	$\overline{Q}_o$

Part D: Draw the state table for the following state diagram.



A	S_1	S_0	NS_1	NS_0	B
0	0	0	0	0	0
1	0	0	0	1	0
0	0	1	0	1	0
1	0	1	1	0	1
0	1	0	1	0	0
1	1	0	1	1	0
0	1	1	1	1	0
1	1	1	0	0	0

Problem 3 (3 parts, 34 points) Memory and Maps

Part A (12 points) Consider a gigabit DRAM chip organized as **64 million addresses** of **16 bit words**. Assume both the DRAM cell and the DRAM chip is 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**.

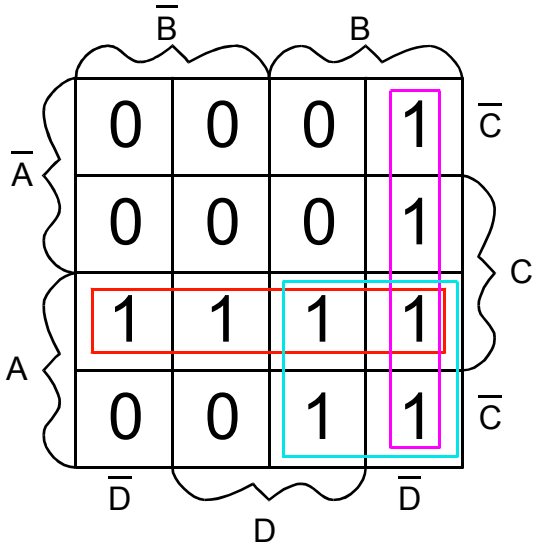
number of columns	sqrt(2^{30}) = 2^{15} = 32K
column decoder required (*n* to *m*)	15 to 32K
type of mux required (*n* to *m*)	32K/16 → 2K to 1
number of muxes required	16
number of address lines in column number	15
number of address lines in column offset	11

Part B (10 points) Consider a memory system with **512 million addresses** of **four byte words** using DRAM chips organized as **64 million** addresses by **16 bit words**

word address lines for memory system	29
chips needed in one bank	4 bytes / 2 bytes = 2 chips/bank
banks for memory system	512M / 64M = 8 banks/system
memory decoder required (*n* to *m*)	3 to 8
DRAM chips required	2 x 8 = 16 chips/system

Part C (12 points) For the follow expression, derive a simplified ***sum of products*** expression using a Karnaugh Map. Circle and list *all* prime implicants, indicating which are essential.

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



prime implicants	essential? yes	no
A B	☒	☐
A C	☒	☐
B $\overline{D}$	☒	☐
	☐	☐
	☐	☐
	☐	☐
	☐	☐
	☐	☐

simplified **SOP** expression $A \cdot B + A \cdot C + B \cdot \overline{D}$

Problem 4 (4 parts, 38 points) Number Systems

Part A (8 points) Convert the following notations:

binary notation	**decimal notation**
1010 1011.1010	171.625
10 1001.1110	41.875
binary notation	**hexadecimal notation**
11 1100 0011.1100 0011 11	3C3.C3C
11 1111.0010 0011	3F.23

Part B (9 points) For the representations below, determine the most positive value and the step size (difference between sequential values). All answers should be expressed in decimal notation. Fractions (e.g., 3/16ths) may be used. Signed representations are two's complement.

representation	most positive value	step size
signed integer (20 bits) . (0 bits)	512K	1
unsigned fixed-point (15 bits) . (5 bits)	32K	1/32
signed fixed-point (10 bits) . (10 bits)	512	1/1K

Part C (9 points) A 20 bit floating point representation has a 13 bit mantissa field, a 6 bit exponent field, and one sign bit. *Express all answers in decimal*.

What is the largest value that can be represented (closest to infinity)? 2^{31} = 2B

What is the smallest value that can be represented (closest to zero)? 2^{-32} = 1/4B

How many decimal significant figures are supported? ~4

Part D (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 **five bit unsigned fixed-point** and **five bit two's complement fixed-point** representations.

	1111.0 + 1.1	10.00 + 10.00	10001 - 11	1.1100 - 0.1011
result	0000.1	100.00	1110	1.0001
unsigned error?	□ no ■ yes	■ no □ yes	■ no □ yes	■ no □ yes
signed error?	■ no □ yes	□ no ■ yes	□ no ■ yes	■ no □ yes

Problem 5 (1 part, 30 points) Assembly Programming

Part A (30 points) Complete an assembly language routine that computes the **average of positives integers** in a **200** element array. The array begins at array 5000. Ignore integers in the array that negative (less than zero). Remember that memory is byte addressed and each word in memory is four bytes long (the first word starts at 5000, the second word starts at 5004, etc.) Each time a positive integer is encountered, the number of positive integers ($5) is incremented. Later $5 is used to compute the average. Use the following register assignments: $1= array pointer, $2= end address, $3= current element, $4= current sum, $5= num positive integers, $6= branch predicate. The result (positive integer average) should be stored in $4.

label	*instruction*	*comment*
PosAvg:	**addi $1, $0, 5000**	**# init array ptr**
	addi $2, $1, 800	**# compute end address**
	addi $4, $0, 0	**# clear current sum**
	addi $5, $0, 0	**# clear number pos ints**
Loop:	**lw $3, 0($1)**	**# load current element**
	slt $6, $3, $0	**# if element < 0**
	bne $6, $0, Skip	**# then skip**
	add $4, $4, $3	**# else add to sum**
	addi $5, $5, 1	**# increment num pos ints**
Skip:	**addi $1, $1, 4**	**# move to next element**
	bne $1, $2, Loop	**# if not done, loop**
	div $4, $5	**# sum / num pos ints**
	mflo $4	**# move avg to $4**
	jr $31	**# return to caller**