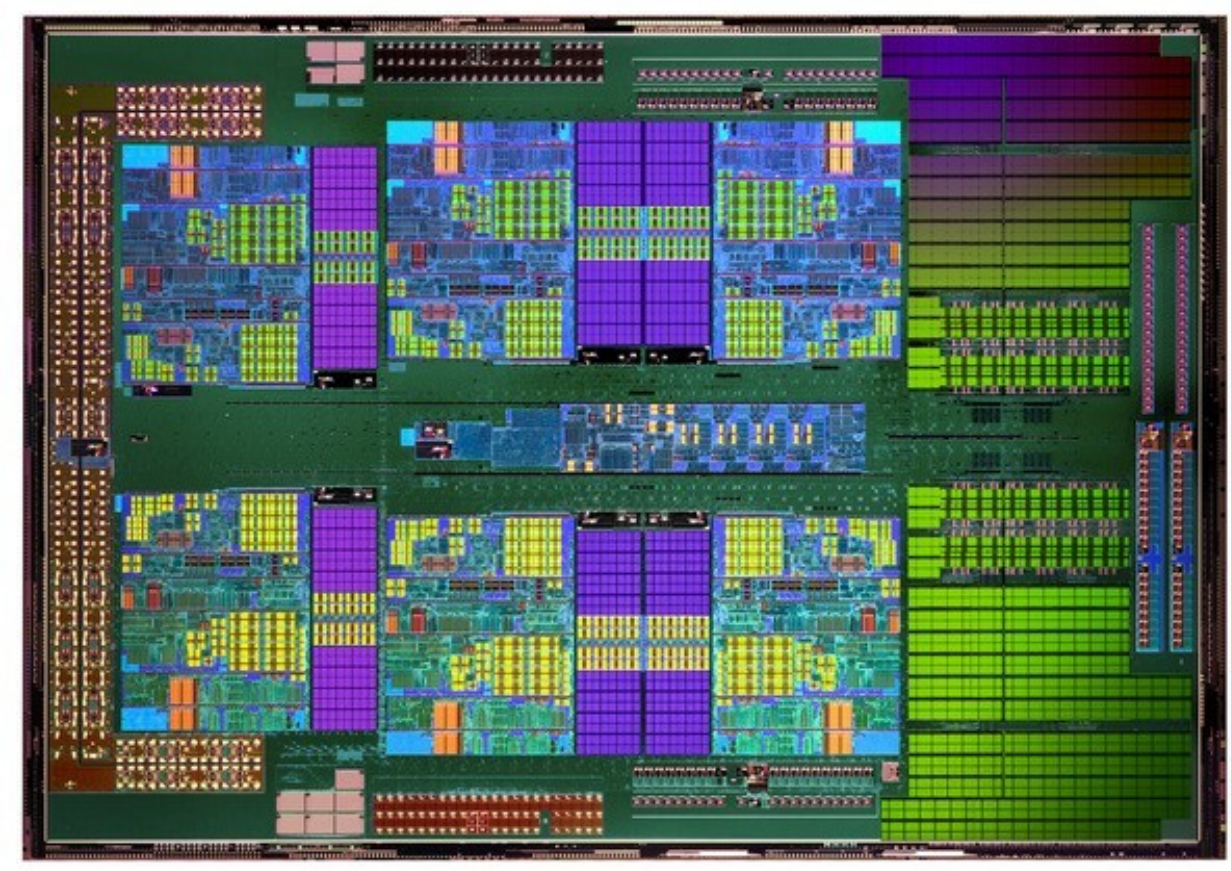
ECE 2030 A 11:00pm	Computer Engineering	Spring 2011
5 problems, 7 pages	Final Exam	6 May 2011

Instructions: This is a closed book, closed note exam. Calculators and other electronics are not permitted. If you have a question, raise your hand and I will come to you. Please work the exam in pencil and do not separate the pages of the exam. For maximum credit, show your work. *Good Luck!*

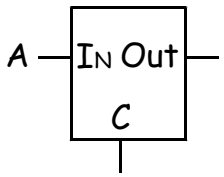
Your Name (***please print***) _____

1	2	3	4	5	total
21	32	34	38	30	155



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.)



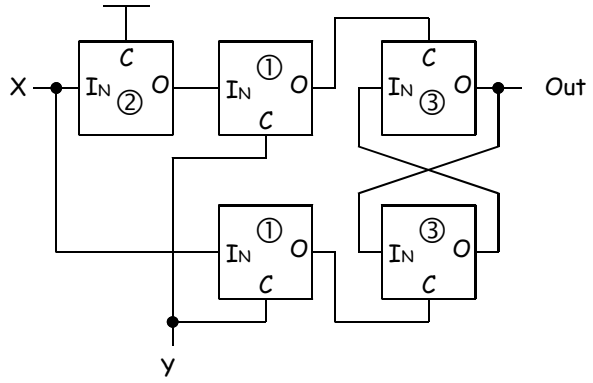
In	C	①	②	③	④	⑤
A	0	0	A	$\overline{A}$	A	Z_o
A	1	A	$\overline{A}$	0	1	A

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

① _____ ② _____ ③ _____

④ _____ ⑤ _____

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



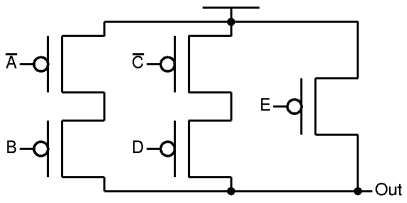
X	Y	Out
0	0	
1	0	
0	1	
1	1	

It's a _____

Problem 2 (4 parts, 32 points) Design Fun

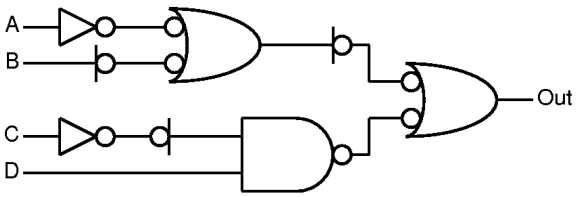
Complete each design below. Be sure to label all signals.

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



Out = _____

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



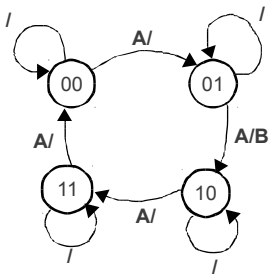
Out = _____

switches = _____

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

TE	$\overline{CLR}$	CLK	Out
		↑↓	
		↑↓	
		↑↓	

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



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

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 _____

column decoder required (*n* to *m*) _____

type of mux required (*n* to *m*) _____

number of muxes required _____

number of address lines in column number _____

number of address lines in column offset _____

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 _____

chips needed in one bank _____

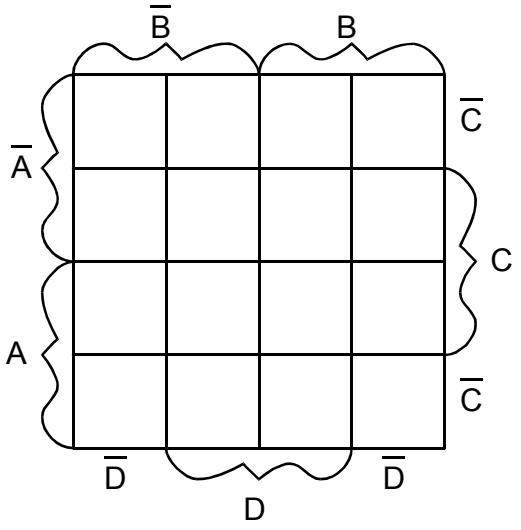
banks for memory system _____

memory decoder required (*n* to *m*) _____

DRAM chips required _____

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
_____	☐	☐
_____	☐	☐
_____	☐	☐
_____	☐	☐
_____	☐	☐
_____	☐	☐
_____	☐	☐
_____	☐	☐

simplified **SOP** expression _____

Problem 4 (4 parts, 38 points) Number Systems

Part A (8 points) Convert the following notations:

binary notation	**decimal notation**
1010 1011.1010	
	41.875
binary notation	**hexadecimal notation**
11 1100 0011.1100 0011 11	
	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)		
unsigned fixed-point (15 bits) . (5 bits)		
signed fixed-point (10 bits) . (10 bits)		

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)? _____

What is the smallest value that can be represented (closest to zero)? _____

How many decimal significant figures are supported? _____

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				
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:		**# init array ptr**
		# compute end address
		# clear current sum
		# clear number pos ints
		# load current element
		# if element < 0
		# then skip
		# else add to sum
		# increment num pos ints
		# move to next element
		# if not done, loop
		# sum / num pos ints
		# move avg to $4
	jr $31	**# return to caller**

MIPS Instruction Set

instruction	*example*	*meaning*
arithmetic		
add	add $1,$2,$3	$1 = $2 + $3
subtract	sub $1,$2,$3	$1 = $2 - $3
add immediate	addi $1,$2,100	$1 = $2 + 100
add unsigned	addu $1,$2,$3	$1 = $2 + $3
subtract unsigned	subu $1,$2,$3	$1 = $2 - $3
add immediate unsigned	addiu $1,$2,100	$1 = $2 + 100
set if less than	slt $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if less than immediate	slti $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
set if less than unsigned	sltu $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if < immediate unsigned	sltui $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
multiply	mult $2,$3	Hi, Lo = $2 * $3, 64-bit signed product
multiply unsigned	multu $2,$3	Hi, Lo = $2 * $3, 64-bit unsigned product
divide	div $2,$3	Lo = $2 / $3, Hi = $2 mod $3
divide unsigned	divu $2,$3	Lo = $2 / $3, Hi = $2 mod $3, unsigned
transfer		
move from Hi	mfhi $1	$1 = Hi
move from Lo	mflo $1	$1 = Lo
load upper immediate	lui $1,100	$1 = 100 x 2^{16}
logic		
and	and $1,$2,$3	$1 = $2 & $3
or	or $1,$2,$3	$1 = $2 | $3
and immediate	andi $1,$2,100	$1 = $2 & 100
or immediate	ori $1,$2,100	$1 = $2 | 100
nor	nor $1,$2,$3	$1 = not($2 | $3)
xor	xor $1, $2, $3	$1 = $2 ⊕ $3
xor immediate	xori $1, $2, 255	$1 = $2 ⊕ 255
shift		
shift left logical	sll $1,$2,5	$1 = $2 << 5 (logical)
shift left logical variable	sllv $1,$2,$3	$1 = $2 << $3 (logical), variable shift amt
shift right logical	srl $1,$2,5	$1 = $2 >> 5 (logical)
shift right logical variable	srlv $1,$2,$3	$1 = $2 >> $3 (logical), variable shift amt
shift right arithmetic	sra $1,$2,5	$1 = $2 >> 5 (arithmetic)
shift right arithmetic variable	srav $1,$2,$3	$1 = $2 >> $3 (arithmetic), variable shift amt
memory		
load word	lw $1, 1000($2)	$1 = memory [$2+1000]
store word	sw $1, 1000($2)	memory [$2+1000] = $1
load byte	lb $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
load byte unsigned	lbu $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
store byte	sb $1, 1002($2)	memory[$2+1002] = $1 (byte modified only)
branch		
branch if equal	beq $1,$2,100	if ($1 = $2), PC = PC + 4 + (100*4)
branch if not equal	bne $1,$2,100	if ($1 ≠ $2), PC = PC + 4 + (100*4)
jump		
jump	j 10000	PC = 10000*4
jump register	jr $31	PC = $31
jump and link	jal 10000	$31 = PC + 4; PC = 10000*4