ECE 2030 12:00pm Computer Engineering Fall 2008
5 problems, 7 pages Final Exam 10 December 2008

Instructions: This is a closed book, closed note exam. Calculators 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
32	34	18	34	34	152

Problem 1 (4 parts, 32 points) Implementation Bonanza

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

Part A (8 points) Implement the expression below using N and P type switches.

$$OUT_X = \overline{A} \cdot B \cdot \left(\overline{C} + D\right) + \overline{E}$$

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

$$OUT_Y = \left(A + \overline{B + C}\right) \cdot \overline{D}$$

Part B (8 points) Implement a 1 to 4 DEMUX using only pass gates and inverters.

Part D (8 points) Implement a full adder using AND, OR, NAND, NOR, NOT, & XOR gates.

Problem 2 (2 parts, 34 points) Even Average

In this problem, you will write a subroutine that determines the average of even numbers in a variable length array in memory. **Comments for each instruction and labels are already provided**. Input parameters to this subroutine include a pointer to the first array element ($1) and the number of elements in the array ($2). The result (the average of even numbers) is returned in $3. As always, the return address for this subroutine arrives in $31.

input parameters				result		working registers					
reg	content	reg	content	reg	content	reg	content	reg	content	reg	content
$1	array pointer	$2	# array elements	$3	input / result	$4	running sum	$5	# even values	$6	predicate

Part A (24 points) Write a subroutine that computes the average of even values in an array.

label	instruction	comment
EvenAvg:		# clear running sum
		# clear # of even values
		# adjust # elem for byte address
		# point to addr after last elem
Loop:		# load next array element
		# test if even or odd
		# if odd, skip
		# if even, add to running sum
		# and increment # even values
Skip:		# adjust array ptr to next elem
		# if not at end of array, loop
		# compute even average
		# return to caller

Part B (10 points) Write a code fragment that calls EvenAvg for a 100 element array starting at address 5000. When the subroutine completes, store the result at memory location 6000.

label	instruction	comment
		# load array starting pointer
		# load array size (# elements)
		# call even array average
		# load address for result
		# store result to memory

Problem 3 (2 parts, 18 points) Instruction Formats

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

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

What is the maximum number of opcodes? ____________________

What is the number of registers? ____________________

What is the range of the signed immediate value? ____________________

Part B (9 points) List three differences between a branch and a jump in the MIPS ISA.

1: ____

2: ____

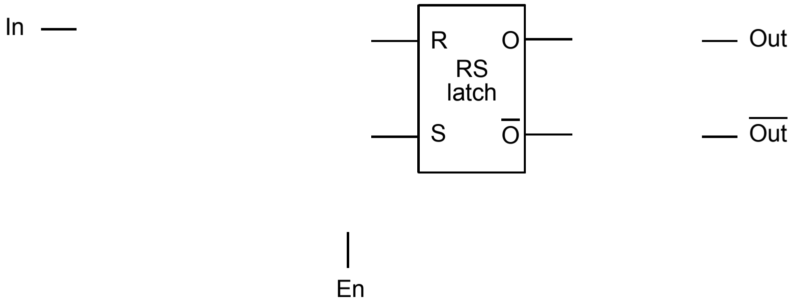
3: ____

Problem 4 (4 parts, 34 points) State of the Union

Part A (7 points) Implement an RS latch with active high inputs, R and S. Use only basic gates (AND, OR, NAND, NOR, and NOT). Label the inputs and output. Also complete the behavior table. Note -Out means $\overline{Out}$.

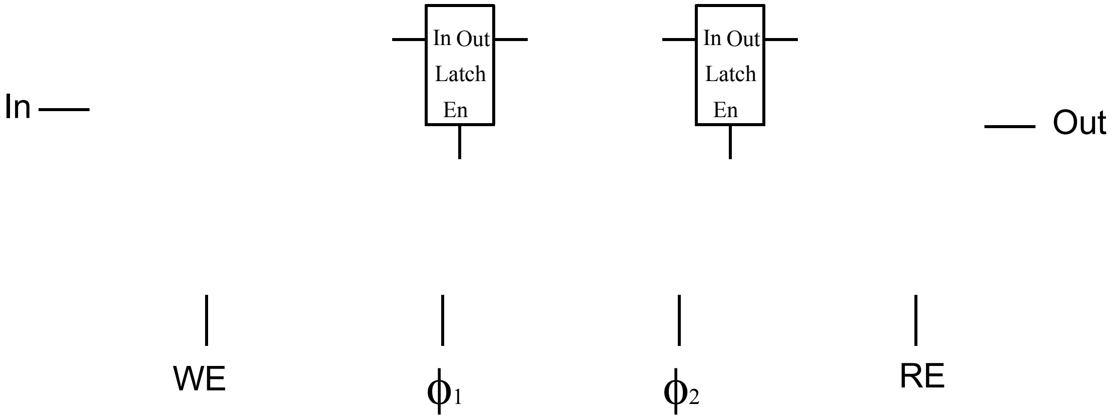
R	S	Out	-Out
0	0		
1	0		
0	1		
1	1		

Part B (7 points) Expand the RS latch to a transparent latch and complete the truth table. Use only basic gates (AND, OR, NAND, NOR, and NOT). Label the inputs and output. Also complete the behavior table.



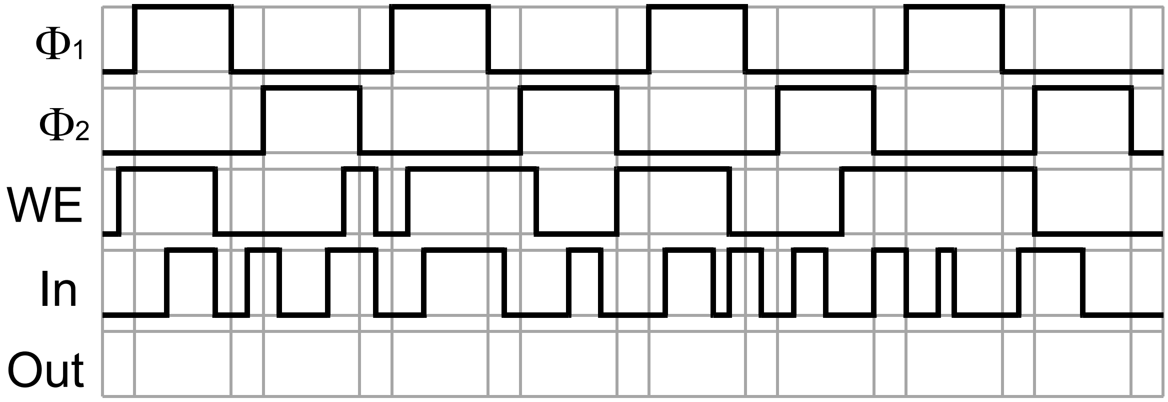
In	En	Out	-Out
A	0		
A	1		

Part C (12 points) Build a register using two transparent latches plus a 2to1 mux (draw the labeled icon), a pass gate, and an inverter. Again, complete the behavior table. 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 (with the given inputs).



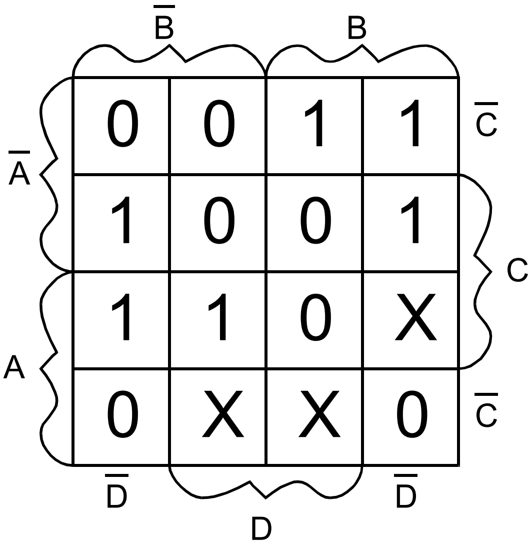
In	WE	RE	Clk	Out
A	0	0	↑↓	
A	1	0	↑↓	
A	0	1	↑↓	
A	1	1	↑↓	

Part D (8 points) Assume the following signals are applied to your register. Draw the output signal **Out**. Draw a vertical line where **In** is sampled. *Draw crosshatch where* ***Out*** *is unknown.*



Problem 5 (3 parts, 34 points) This and That

Part A (12 points) For the following Karnaugh Map, derive a simplified ***product of sums*** expression. Circle and list the prime implicants, indicating which are essential.



prime implicants	essential? yes	no
____________________	□	□
____________________	□	□
____________________	□	□
____________________	□	□
____________________	□	□
____________________	□	□
____________________	□	□
____________________	□	□

simplified **POS** expression ______________________________

Part B (12 points) Using the supplied datapath, write a microcode fragment to accomplish the following expression. Express all values in hexadecimal notation. Use 'X' when a value is don't cared. For maximum credit, complete the description field. $\cap$ means bitwise logical AND.

$$R_1 = \left(\frac{mem[100]}{256} \cap 255 \right)$$

#	X	Y	Z	rwe	im en	im va	au en	-a /s	lu en	lf	su en	st	ld en	st en	r/ -w	msel	description
1																	
2																	
3																	
4																	

Part C (10 points) Consider a **4 Gbyte** memory system with **512 million addresses** of **8 byte words** using **1 Gbit** DRAM chips organized as **64 million** addresses by **16 bit words.**

word address lines for memory system ______________________

chips needed in one bank ______________________

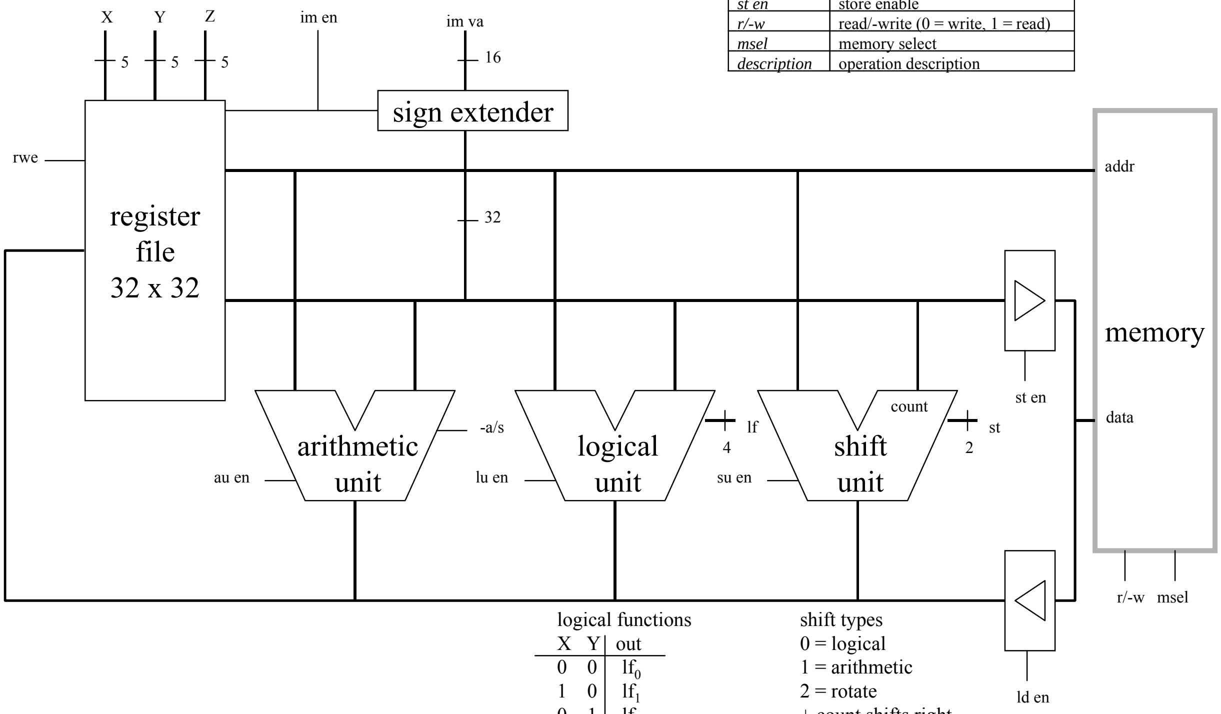
banks for memory system ______________________

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

DRAM chips required ______________________

cycle	cycle number
X	register driven onto X bus
Y	register driven onto Y bus
Z	register written from Z bus
rwe	register write enable
im en	immediate enable on Y bus
im va	immediate value

au en	arithmetic unit enable
-a/s	-add / sub (0 = add, 1 = subtract)
lu en	logical unit enable
lf	logical function
su en	shift unit enable
st	shift type
ld en	load enable
st en	store enable
r/-w	read/-write (0 = write, 1 = read)
msel	memory select
description	operation description



logical functions

X	Y	out
0	0	lf_0
1	0	lf_1
0	1	lf_2
1	1	lf_3

shift types
0 = logical
1 = arithmetic
2 = rotate
+ count shifts right
- count shifts left

instruction	example	meaning
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`
multiply	`mul $1,$2,$3`	`$1 = $2 * $3`
divide	`div $1,$2,$3`	`$1 = $2 / $3`
and	`and $1,$2,$3`	`$1 = $2 & $3`
or	`or $1,$2,$3`	`$1 = $2 | $3`
xor	`xor $1,$2,$3`	`$1 = $2 xor $3`
and immediate	`andi $1,$2,100`	`$1 = $2 & 100`
or immediate	`ori $1,$2,100`	`$1 = $2 | 100`
xor immediate	`xori $1,$2,100`	`$1 = $2 xor 100`
shift left logical	`sll $1,$2,5`	`$1 = $2 << 5 (logical)`
shift right logical	`srl $1,$2,5`	`$1 = $2 >> 5 (logical)`
shift left arithmetic	`sla $1,$2,5`	`$1 = $2 << 5 (arithmetic)`
shift right arithmetic	`sra $1,$2,5`	`$1 = $2 >> 5 (arithmetic)`
load word	`lw $1, ($2)`	`$1 = memory [$2]`
store word	`sw $1, ($2)`	`memory [$2] = $1`
load upper immediate	`lui $1,100`	$1 = 100 x 2^{16}
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)`
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`
jump	`j 10000`	`PC = 10000`
jump register	`jr $31`	`PC = $31`
jump and link	`jal 10000`	`$31 = PC + 4; PC = 10000`