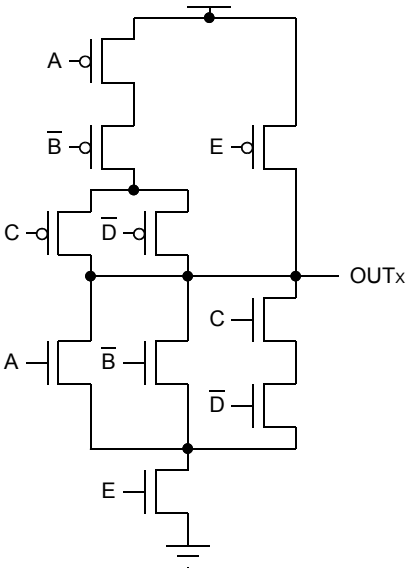
Problem 1 (4 parts, 32 points) Implementation Bonanza

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

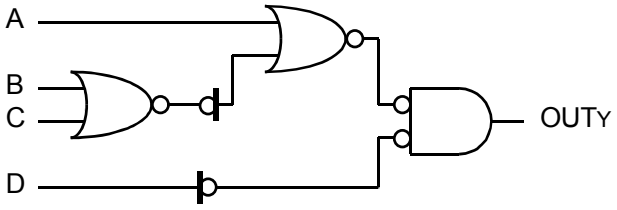
Part A (8 points) Implement the expression below using N and P type switches.

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

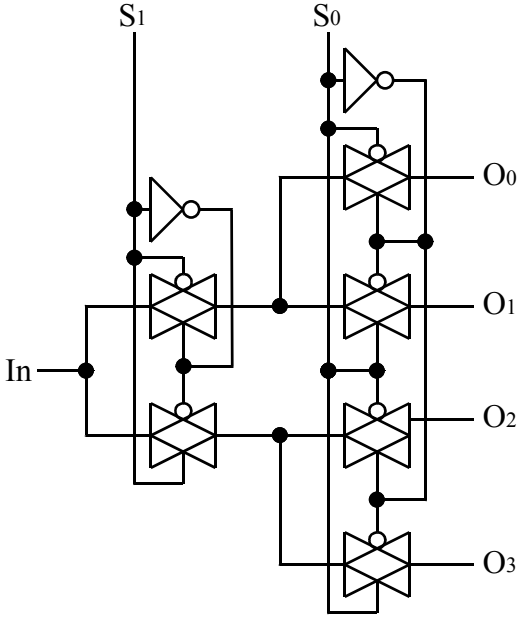


Part B (8 points) Implement the expression in mixed logic notation using NOR gates.

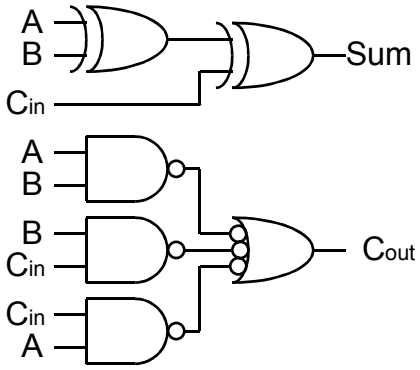
$$OUT_Y = \left(A + \overline{B + C}\right) \cdot \overline{D}$$



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:	and $4, $0, $0	# clear running sum
	and $5, $0, $0	# clear # of even values
	sll $2, $2, 2	# adjust # elem for byte address
	add $2, $1, $2	# point to addr after last elem
Loop:	lw $3, 0($1)	# load next array element
	andi $6, $3, 1	# test if even or odd
	bne $6, $0, Skip	# if odd, skip
	add $4, $4, $3	# if even, add to running sum
	addi $5, $5, 1	# and increment # even values
Skip:	addi $1, $1, 4	# adjust array ptr to next elem
	bne $1, $2, Loop	# if not at end of array, loop
	div $3, $4, $5	# compute even average
	jr $31	# 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
	addi $1, $0, 5000	# load array starting pointer
	addi $2, $0, 100	# load array size (# elements)
	jal EvenAvg	# call even array average
	addi $1, $0, 6000	# load address for result
	sw $3, ($1)	# 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? 128

What is the number of registers? 512

What is the range of the signed immediate value? ±128K

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

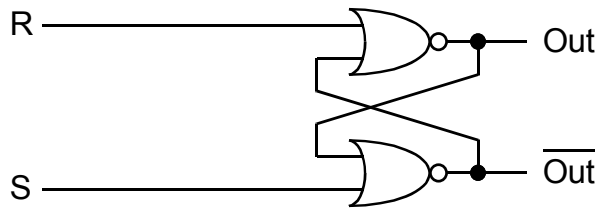
1: Branches are conditional; jumps are unconditional.

2: Branch offsets are relative, jump targets are absolute.

3: Branch range ±32K I, ±128Kbytes; jump range 0-64M I, 0-256Mbytes

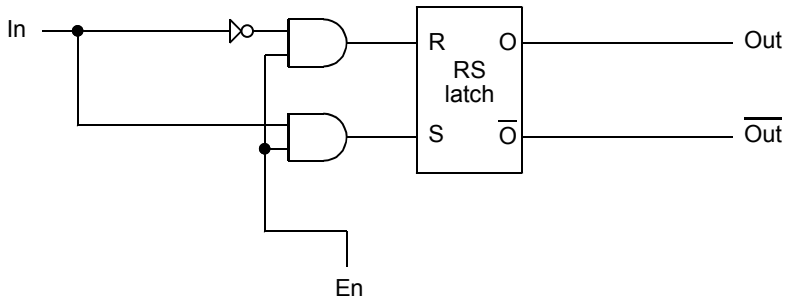
Problem 4 (4 parts, 34 points) State of the Union

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 $\overline{Out}$.



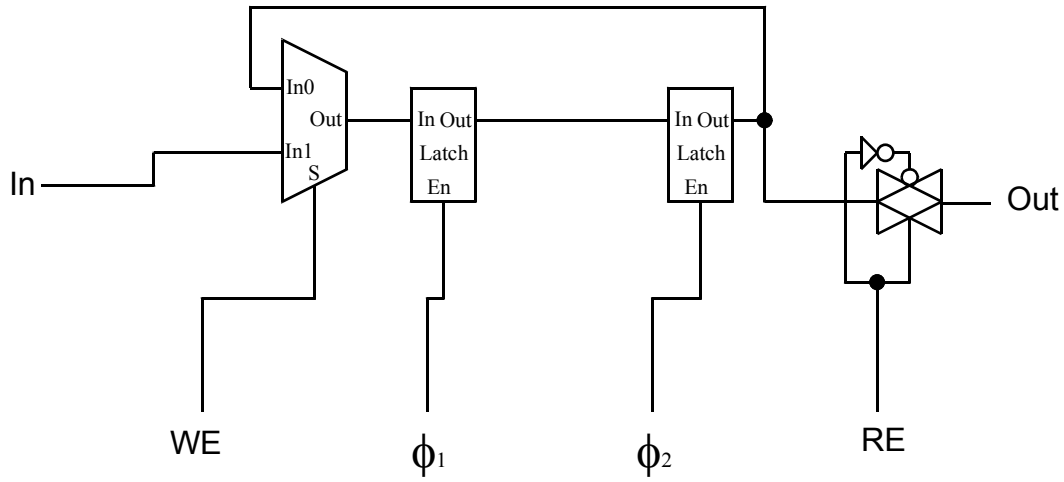
R	S	Out	-Out
0	0	Q_o	$-Q_0$
1	0	0	1
0	1	1	0
1	1	0	0

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.



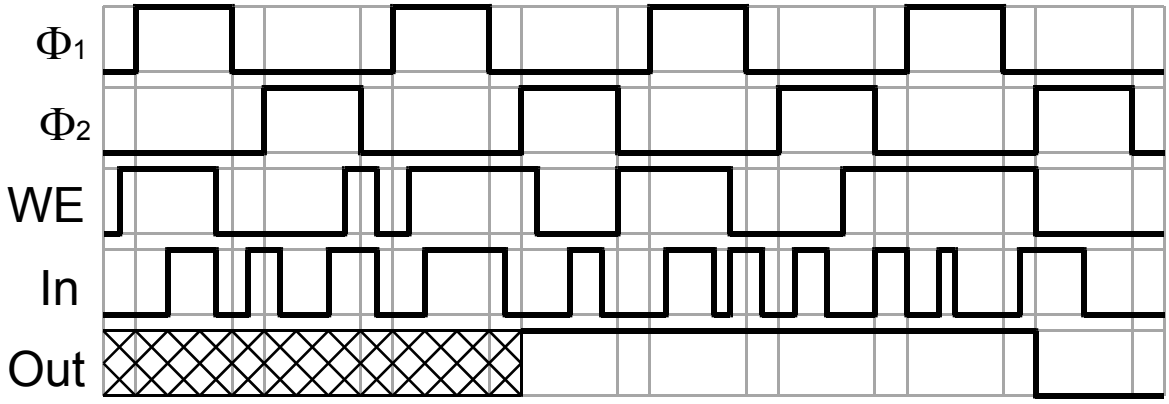
In	En	Out	-Out
A	0	Q_o	$-Q_o$
A	1	A	-A

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).



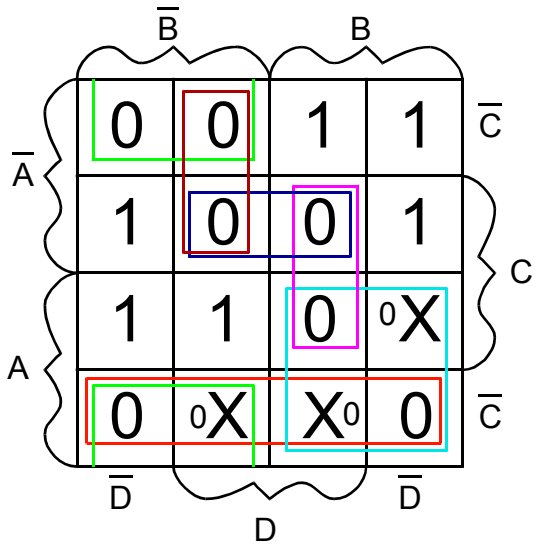
In	WE	RE	Clk	Out
A	0	0	↑↓	Z_o
A	1	0	↑↓	Z_o
A	0	1	↑↓	Q_o
A	1	1	↑↓	A

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
$\bar{A} + \bar{B}$	☒	☐
$B + C$	☒	☐
$\bar{A} + C$	☐	☒
$A + B + \bar{D}$	☐	☒
$A + \bar{C} + \bar{D}$	☐	☒
$\bar{B} + \bar{C} + \bar{D}$	☐	☒
	☐	☐
	☐	☐

simplified **POS** expression $(B+C)\cdot(\bar{A}+\bar{B})\cdot(A+\bar{C}+\bar{D})$

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. ∩ 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	X	X	1	1	1	64	0	X	1	C	0	0	0	0	X	0	R1 <- 100
2	1	X	1	1	0	X	0	X	0	X	0	X	1	0	1	1	R1 <- mem[100]
3	1	X	1	1	1	8	0	X	0	X	1	0	0	0	X	0	R1 <- R1 >> 8
4	1	X	1	1	1	FF	0	X	1	8	0	X	0	0	X	0	R1 <- R1 & 255

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	$\log_2(512M) = 29$ address lines
chips needed in one bank	8byte/16bit = 64/16 = 4 chips
banks for memory system	$512M/64M = 2^{29}/2^{26} = 2^3 = 8$ banks
memory decoder required (*n* to *m*)	3 to 8 decoder
DRAM chips required	4 x 8 = 32 chips