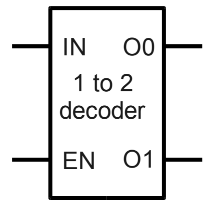
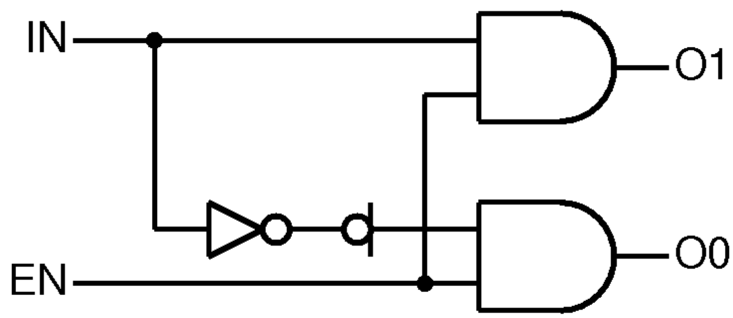
Problem 1 (3 parts, 24 points) Decoding Decoders

Part A (6 points) Define a 1 to 2 decoder by completing the behavior table.

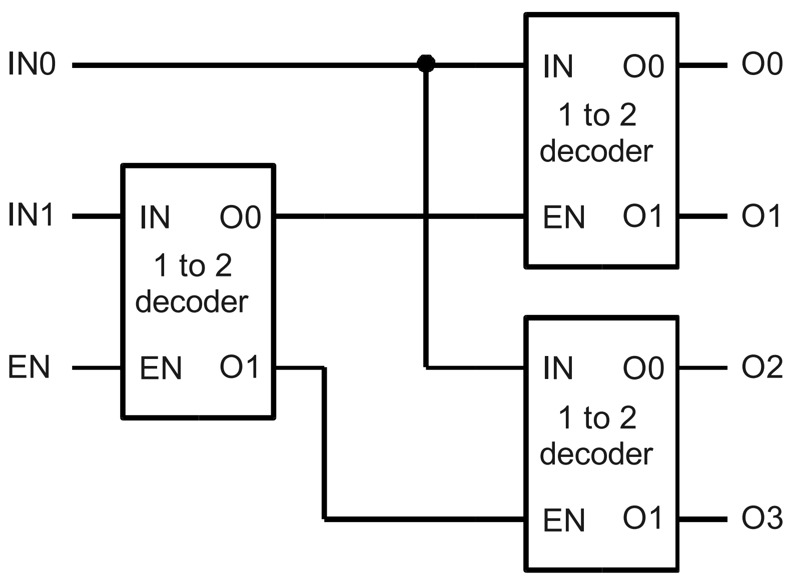
IN	EN	O0	O1
X	0	0	0
0	1	1	0
1	1	0	1



Part B (8 points) Implement a 1 to 2 decoder using basic gates. Assume only true (non-complemented) inputs are available. Label all inputs and outputs.



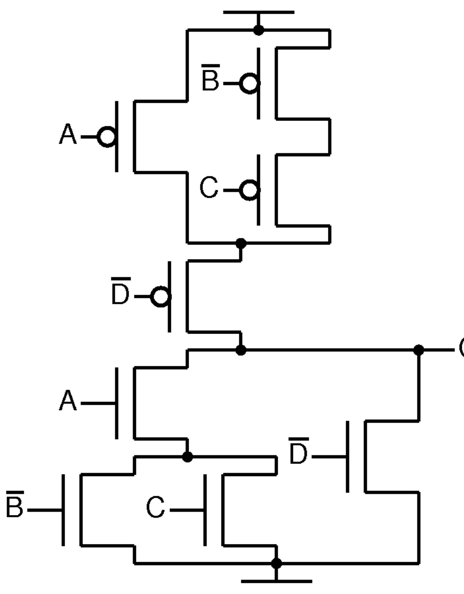
Part C (10 points) Using *only* the three 1 to 2 decoders shown below, implement a 2 to 4 decoder with an enable. Label the decoder inputs (IN_1, IN_0, EN) and outputs (O0, O1, O2, O3).



Problem 2 (4 parts, 30 points) Design Fiesta

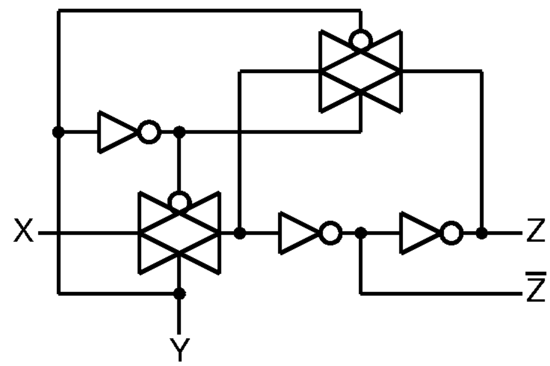
Complete each design below. Be sure to label all signals.

Part A: Implement the following expression using N and P type switches. $Out_X = (\overline{A} + B\cdot\overline{C})\cdot D$

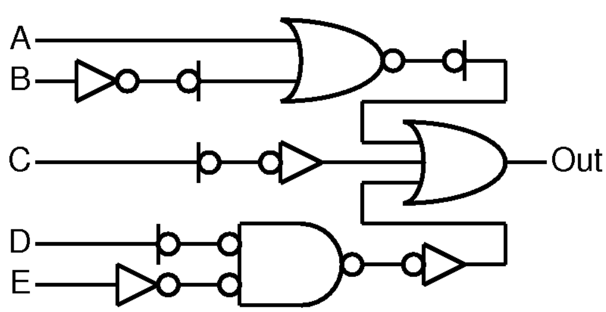


Part B: Implement the following behavior using only pass gates and inverters.

X	Y	Z	$\overline{Z}$
A	0	Q_o	$\overline{Q_o}$
A	1	A	$\overline{A}$



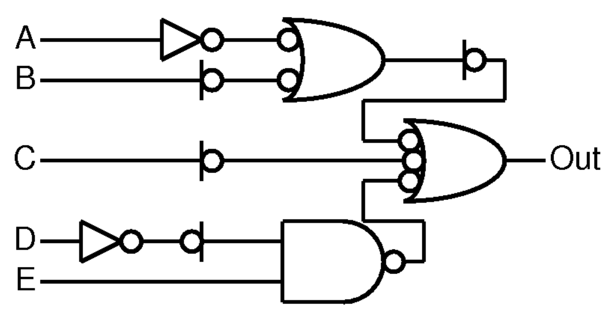
Part C: Determine the appropriate expression for this mixed logic design. How many transistors are required?



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

transistors = 8 + 6 + 4 + 4 x 2 = 26T

Part D: Reimplement the design in Part C using *only* NAND and NOT gates. How many transistors are required?



transistors = 6 + 2 x 4 + 2 x 2 = 18T

Problem 3 (1 part, 25 points) Assembly Programming

Part A (25 points) Complete this subroutine that searches an array of 100 integers beginning at memory address 5000 and returns its minimum ($4) and maximum ($5) values. Use the following registers: $1= array pointer, $2= end address, $3= current value, $6= branch predicate.

label	*instruction*	*comment*
MinMax:	**addi $1, $0, 5000**	**# init array ptr**
	addi $2, $1, 400	**# set end address**
	lw $4, ($1)	**# init min**
	add $5, $3, $0	**# init max**
Loop:	**lw $3, ($1)**	**# load current element**
	slt $6, $3, $4	**# if current >= min**
	beq $6, $0, Skip1	**# then skip update**
	add $4, $3, $0	**# update min**
Skip1:	**slt $6, $5, $3**	**# if current <= max**
	beq $6, $0, Skip2	**# then skip update**
	add $5, $3, $0	**# update max**
Skip2:	**addi $1, $1, 4**	**# point to next element**
	bne $1, $2, Loop	**# if not done, loop**
	jr $31	**# return to caller**

MIPS Instruction Set

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

Problem 4 (4 parts, 36 points) "Math is fun"

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

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

What is the maximum number of opcodes? 256

What is the number of registers? 64

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

Part B (9 points) For the eight bit 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
unsigned integer (8 bits) . (0 bits)	255	1
signed fixed-point (6 bits) . (2 bits)	31	1/4
unsigned fixed-point (0 bits) . (8 bits)	255/256	1/256

Part C (6 points) A 48 bit floating point representation has a 37 bit mantissa field, a 10 bit exponent field, and one sign bit.

What is the largest value that can be represented (closest to infinity)? 2^{511}

What is the smallest value that can be represented (closest to zero)? 2^{-512}

How many decimal significant figures are supported? 11

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.

	0.111 + 0.111	.0011 + 1.1101	1010.0 - 1001.1	0.00 - 100.00
result	1.110	0.0000	0.1	100.00
unsigned error?	■ no □ yes	□ no ■ yes	■ no □ yes	□ no ■ yes
signed error?	■ no □ yes	■ no □ yes	■ no □ yes	□ no ■ yes

Problem 5 (5 parts, 30 points) Microcode in Reverse

The microcode fragment below comes from a color scanner control program that runs on the datapath discussed in class. Unfortunately, don't care values (X) have been converted to zeros. Assume register zero is a normal register (not hardwired to the value zero).

#	*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*
1	0	0	3	1	1	4000	0	0	1	C	0	0	0	0	0	0
2	3	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1
3	0	0	2	1	1	FF	0	0	1	8	0	0	0	0	0	0
4	0	0	0	1	1	8	0	0	0	0	1	0	0	0	0	0
5	0	0	1	1	1	FF	0	0	1	8	0	0	0	0	0	0
6	1	2	2	1	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	1	1	8	0	0	0	0	1	0	0	0	0	0
8	0	0	1	1	1	FF	0	0	1	8	0	0	0	0	0	0
9	1	2	2	1	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	1	1	8	0	0	0	0	1	0	0	0	0	0
11	0	2	2	1	0	0	1	0	0	0	0	0	0	0	0	0
12	2	0	2	1	1	2	0	0	0	0	1	1	0	0	0	0
13	3	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Part A (5 points) Describe the operation that occurs during cycle 2. Be specific.

$0 <- mem[0x4000]

For the remaining parts, assume $0 = 0x44022118 at the end of cycle 2.

Part B (5 points) What is the value of register 0 at the completion of cycle 7 (in hexadecimal).

0x4402

Part C (5 points) What is the value of register 2 at the completion of cycle 9 (in hexadecimal).

0x3B

Part D (5 points) What is the value of register 2 at the completion of cycle 12 (in hexadecimal).

0x1F

Part E (10 points) Describe the operation of this microcode fragment. Be specific.

Four packed eight-bit unsigned integers are loaded from memory at 0x4000, unpacked. The average of the four values is computed and stored back to memory at 0x4000.