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!

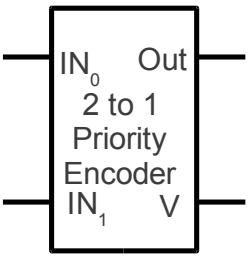
Your Name (***please print***) _____

1	2	3	4	5	total
24	32	30	48	39	173

Problem 1 (3 parts, 24 points) Design This

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

Part A: Define a 2 to 1 priority encoder, where $I_1 > I_0$, by completing the behavior table.



IN_0	IN_1	V	Out
		0	X
		1	0
		1	1

Implement the 2 to 1 encoder using ***one*** basic gate. Only true (non-complemented) inputs are available. Label all inputs (IN0, IN1) and outputs (Out, V).

Part B: Implement a 1 to 2 demux using only pass gates and an inverter. Determine # of switches needed.

switches = _____

Part C: Complete the truth table for even parity. Then write a sum of products (SOP) expression.

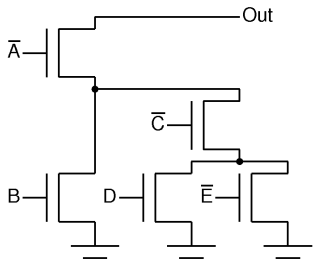
A	B	Out
0	0	
1	0	
0	1	
1	1	

$\overline{A \oplus B} =$ _____

Problem 2 (4 parts, 32 points) Design That

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

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



Out = _____

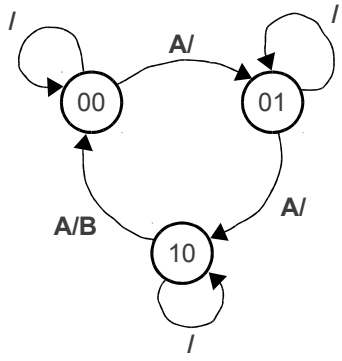
Part B: Implement the following expression using NAND and NOT gates. Use proper mixed logic design. Determine # of switches needed.

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

switches = _____

Part C: Implement a transparent latch using only NOR and NOT gates.

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



A	S_1	S_0	NS_1	NS_0	B

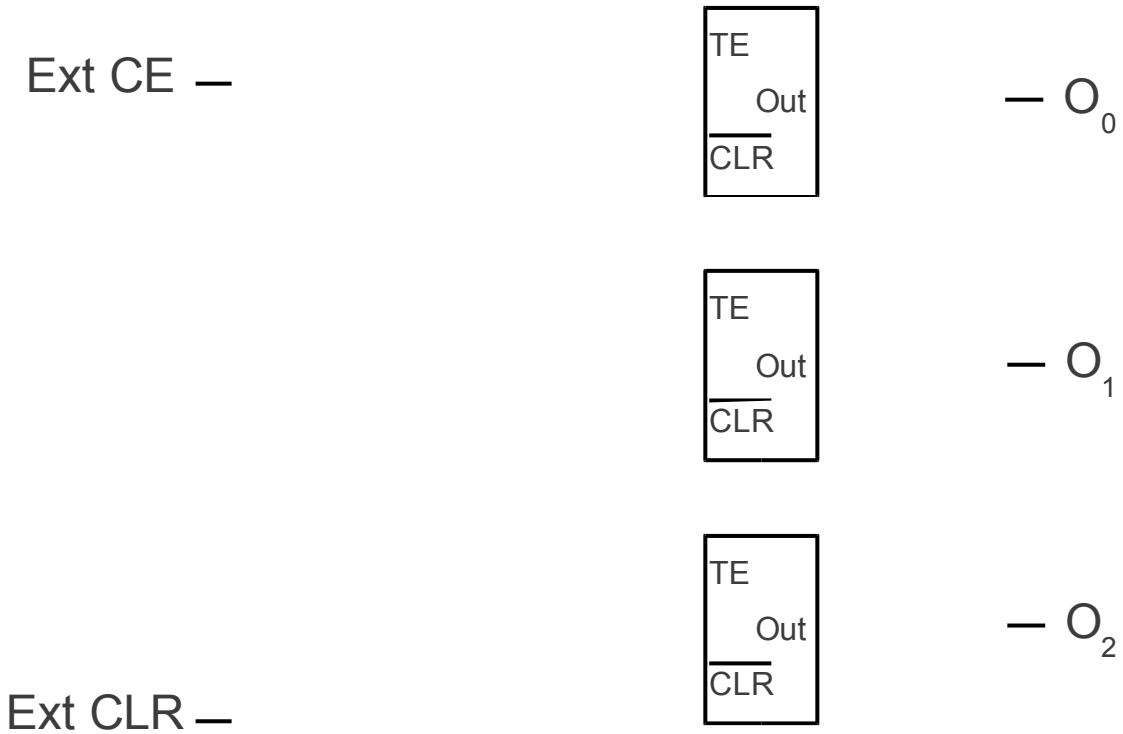
Problem 3 (3 parts, 30 points) Accountable

Part A (10 points) Design a toggle cell using **transparent latches**, **2to1 muxes**, and **inverters** (use icons, **labeling inputs & outputs**). Your toggle cell should have an active high toggle enable input **TE**, and an active low clear input $\overline{\textbf{CLR}}$, clock inputs Φ_1 and Φ_2, and an output **Out**. The $\overline{\textbf{CLR}}$ signal has precedence over **TE**. Also complete the behavior table for the toggle cell.

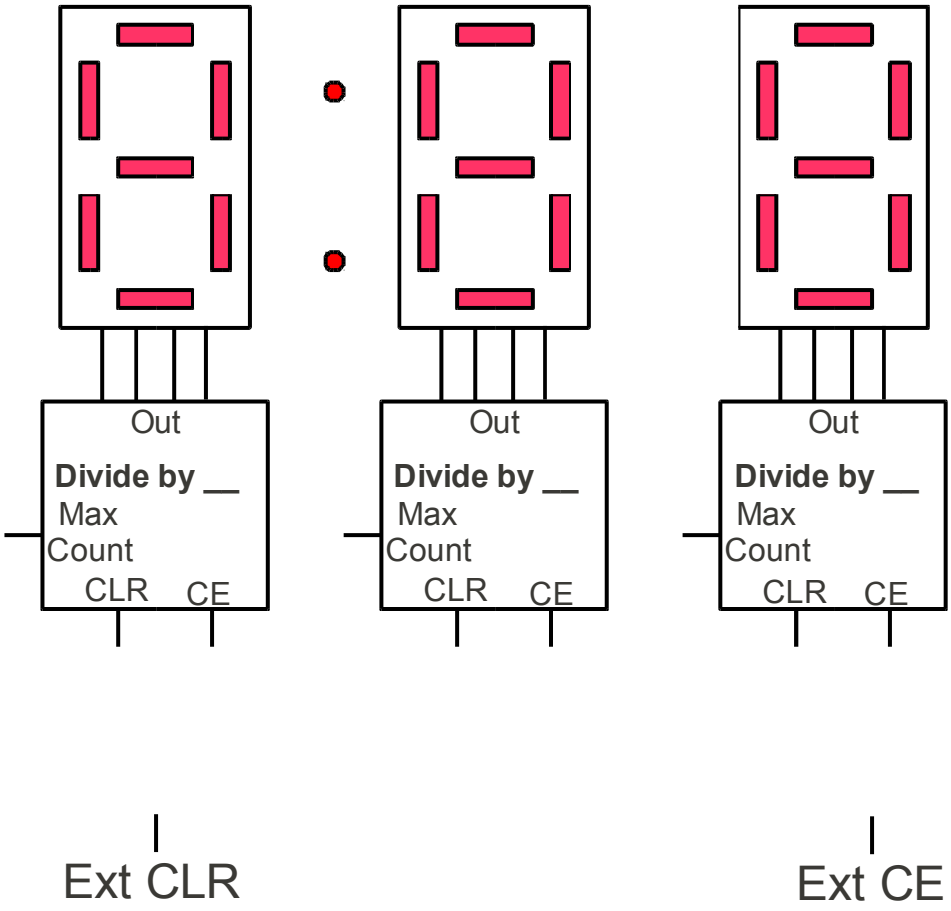
— Out

TE	$\overline{CLR}$	CLK	Out
0	0	↑↓	
1	0	↑↓	
0	1	↑↓	
1	1	↑↓	

Part B (10 points) Now combine these toggle cells to build a **divide by 6** counter. Your counter should have an external clear, external count enable, and three count outputs O_2, O_1, O_0. Use any basic gates (AND, OR, NAND, NOR, & NOT) you require. Assume clock inputs to the toggle cells are already connected. *Your design should support multi-digit systems*.



Part C (10 points) Build a stopwatch that counts seconds and minutes using divide by N counters drawn below. ***Be sure to fill in the needed divider for seconds, tens of seconds, and minutes.*** Use any basic gates you require. Assume a one hertz clock is already connected.



Problem 4 (2 parts, 48 points) Microcode in Reverse

Part A (20 points) Translate this undocumented microcode fragment to corresponding MIPS assembly instructions. Also include comments summarizing the instruction.

#	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	5	0	7	1	0	0	0	0	0	0	0	0	1	0	1	1
2	7	0	9	1	1	C	0	0	0	0	1	0	0	0	0	0
3	9	0	9	1	1	FFF	0	0	1	8	0	0	0	0	0	0
4	9	A	A	1	0	0	1	0	0	0	0	0	0	0	0	0
5	8	A	0	0	0	0	0	0	0	0	0	0	0	1	0	1

1		**#**
2		**#**
3		**#**
4		**#**
5		**#**

Part B (28 points) Complete a recursive subroutine that computes the factorial of N. Assume N is received in $1 and N! is returned in $2. $29 is the stack pointer.

label	instruction	comment
Fact:		**# init result to 1**
		# if N < 2
		# you're done
		# allocate stack space
		# push return address
		# push N
		# decrement N
		# call Fact(N-1)
		# pop N
		# pop return address
		# deallocate stack space
		# N * Fact(N-1)
		# place result in $2
		# return to caller

Problem 5 (4 parts, 39 points) "Random Bits"

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

0 0000	0000	0000	0000 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) 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 (15 bits) . (0 bits)		
unsigned fixed-point (10 bits) . (5 bits)		
signed fixed-point (5 bits) . (10 bits)		

Part C (9 points) A 16 bit floating point representation has a 10 bit mantissa field, a 5 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)? _____

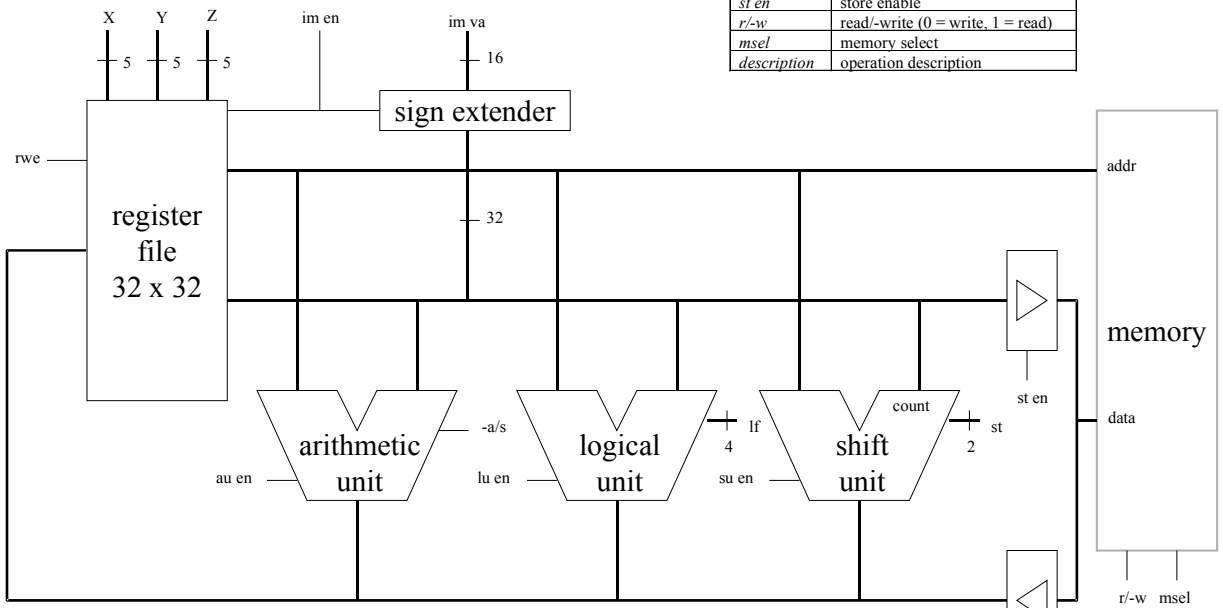
How many decimal significant figures are supported? _____

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.

	11.1 + 111.1	11.011 + 10.101	10101 - 1010	100.00 + .01
result				
unsigned error?	□ no □ yes	□ no □ yes	□ no □ yes	□ no □ yes
signed error?	□ no □ yes	□ no □ yes	□ no □ yes	□ no □ yes

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

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