STUDENT NAME and NUMBER: _____

(PLEASE PRINT CLEARLY)

Georgia Institute of Technology
Department of Electrical and Computer Engineering

Test #3

ECE2030: Introduction to Computer Engineering

Marks:		
	Q1	/25
	Q2	/20
	Q3	/15
	Q4	/20
	Q5	/20
Total:		/100

DATE: November 30, 2000 **TIME:** 4:35 p.m. - 5:55 p.m.

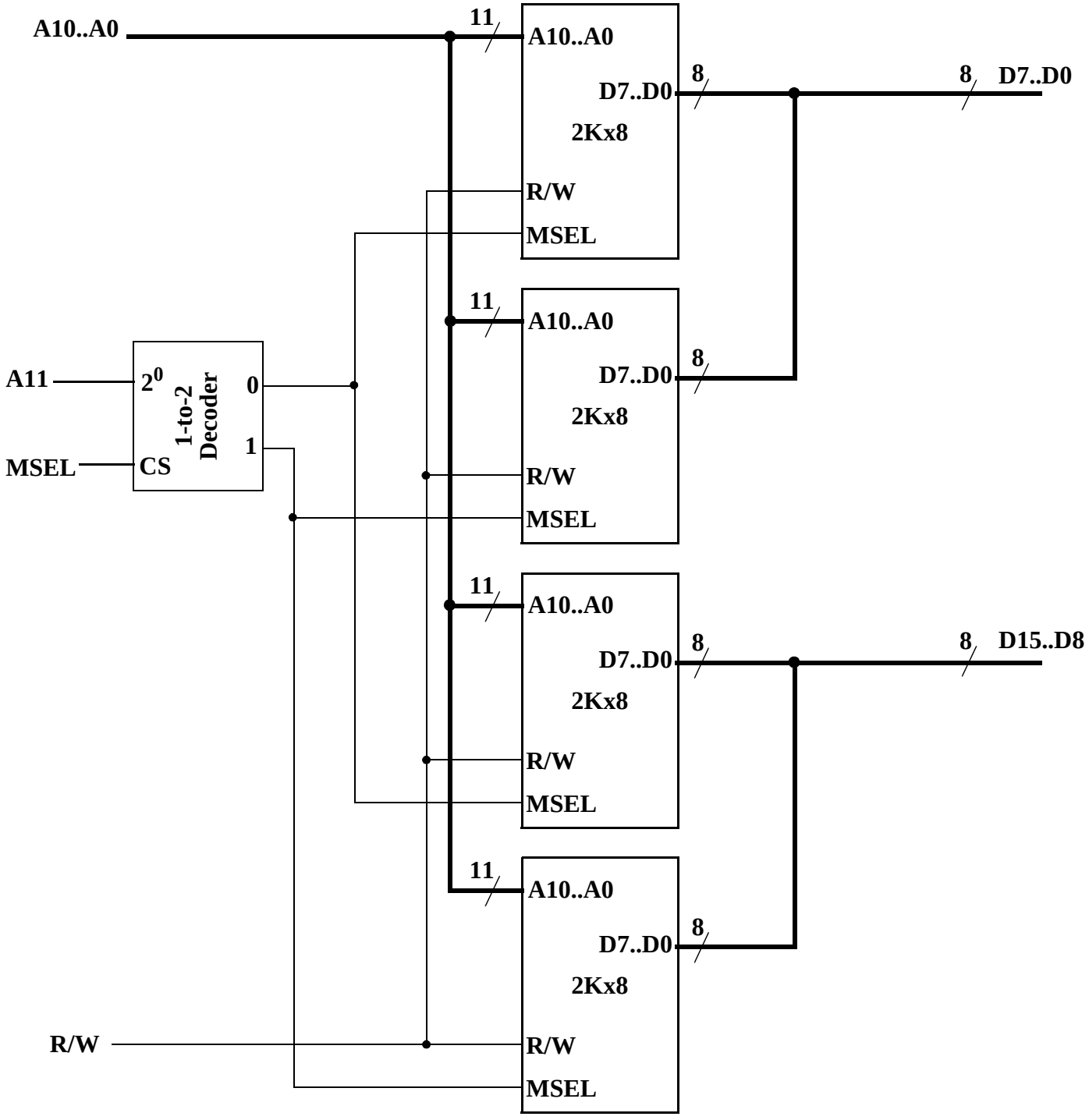
REMARKS: 1. All questions should be answered (in the space provided).
2. Books and notes may NOT be used, with the exception of one 8.5”x11” sheet of paper.

1. MEMORY SYSTEMS *(MAX MARK: 25)*

a) You are given three different memory chips as listed in the table below. Fill in the remaining table cells with the total bits in each of the memory chips, the number of addressable locations, the number of address lines required to address all these locations, and the number of input/output data lines.

Memory chip	Total # of bits	# of addresses	# address lines	# data lines
1Kx8	8,192	1,024	10	8
4Mx12	50,331,648	4,194,304	22	12
2Gx4	8,589,934,592	2,147,483,648	31	4

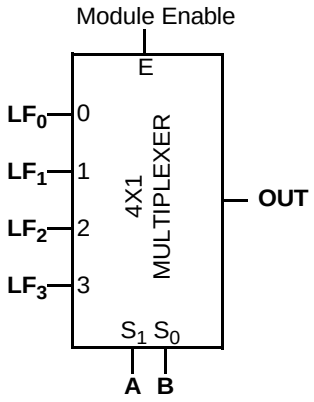
b) The module shown below is a 2Kx8 memory chip. Design a 4Kx16 bank of memory using multiple copies of the 2Kx8 memory chip shown and whatever other combinational building blocks or logic elements you need. The inputs and outputs of your circuit should be the appropriate number of ADDRESS and DATA lines, as well as the MSEL (an enable to select the entire memory) and R/W lines to specify a read (or write) operation. It is OK to use "bus notation" to indicate multi-line datapaths, such as address and data lines. *Label all inputs and outputs to your circuit, and label all pins on chips and building blocks used*. Otherwise points will be deducted.



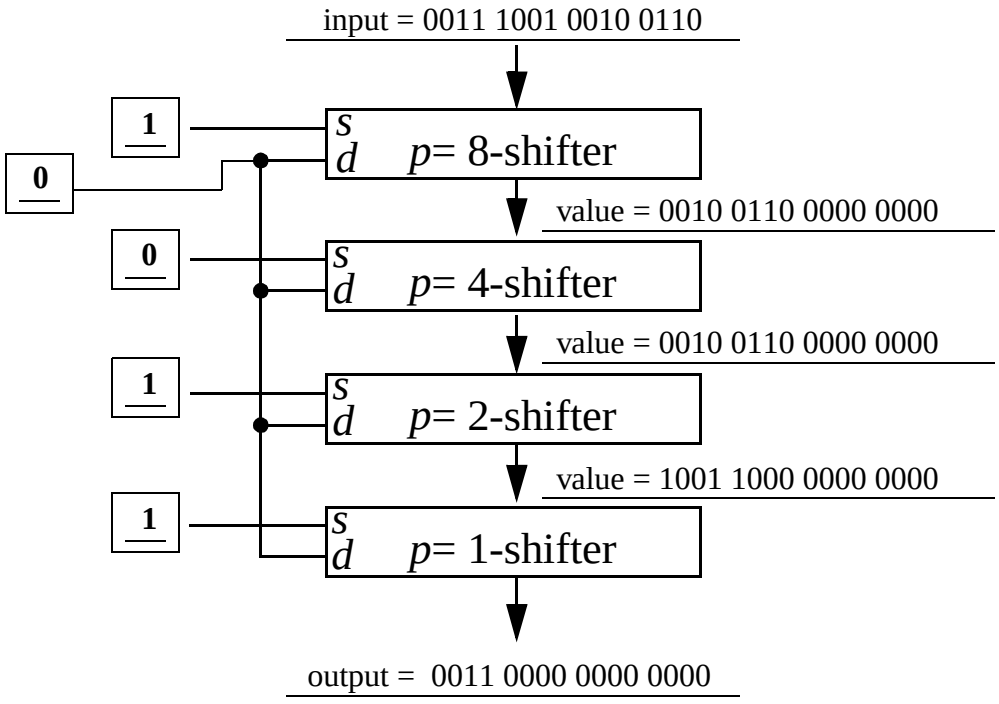
2. DATAPATH ELEMENTS *(MAX MARK: 20)*

a) The below 4x1 MUX can implement any unary and binary logic function on **A** and **B**. In the table below, indicate the values for the logic function **LF**={$\mathbf{LF_3}$, $\mathbf{LF_2}$, $\mathbf{LF_1}$, $\mathbf{LF_0}$} that will produce the logic operation in the first column of the table.

Function	$\mathbf{LF_3}$	$\mathbf{LF_2}$	$\mathbf{LF_1}$	$\mathbf{LF_0}$
$\mathbf{A + \bar{B}}$	**1**	**1**	**0**	**1**
$\mathbf{\bar{A}B}$	**0**	**0**	**1**	**0**
$\mathbf{\bar{B}}$	**0**	**1**	**0**	**1**



b) Below is a barrel shifter that can shift a vector by up to 15 bits either to the left (*d* = 0) or to the right (*d* = 1). First, label each of the *p*-shifters with the appropriate value of *p*. Next, apply the appropriate inputs for *s* and *d* to perform a left shift by eleven and show the input vector **0011 1001 0010 0110** as it passes through each stage of the barrel shifter.



3. ASSEMBLY DIRECTIVES *(MAX MARK: 15)*

Show the values in the data segment for the following assembler directives. Assume that the machine is *little endian*, that the data segment starts at address 0x00100000, and each location can hold one byte. The ASCII for 'A' in hexadecimal is 0x41 and for '+' is 0x2B.

```
.data
.space 1
.asciiz "A+"
.byte 5_10
.align 2
.word 14_10
```

0x00100000	
0x00100001	0x41
0x00100002	0x2B
0x00100003	0x00
0x00100004	0x05
0x00100005	
0x00100006	
0x00100007	
0x00100008	0x0E
0x00100009	0x00
0x0010000A	0x00
0x0010000B	0x00

4. SINGLE CYCLE DPU *(MAX MARK: 20)*

Write a microcode fragment in the table below that will compute the following register transfer equation. Use the single cycle datapath unit diagram attached at the back of the exam for reference on the control signals. In the computation **use only R1, R2, R3, and R4**. Only **R1** and **R3** may be modified after their initial values have been used. For maximum credit, include a description of each microcode operation as well as mark the appropriate "don't care" inputs with an X.

M[R4] = ((R1 * 16) & R3) + M[R2]

Clock Cycle	X	Y	Z	rwe	imm en	imm va	a en	asl	a/s	LF (3210)	ST	st en	ld en	r/w	msel	Description
1	1	X	1	1	1	-4	1	01	X	X	00	0	0	X	0	R1 = R1 asf -4
2	1	3	1	1	0	X	1	10	X	1000	X	0	0	X	0	R1 = R1 & R3
3	2	X	3	1	0	X	0	X	X	X	X	0	1	0	1	R3 = M[R2]
4	1	3	1	1	0	X	1	00	0	X	X	0	0	X	0	R1 = R1 + R3
5	4	1	X	0	0	X	0	X	X	X	X	1	X	1	1	M[R4] = R1
6																
7																
8																

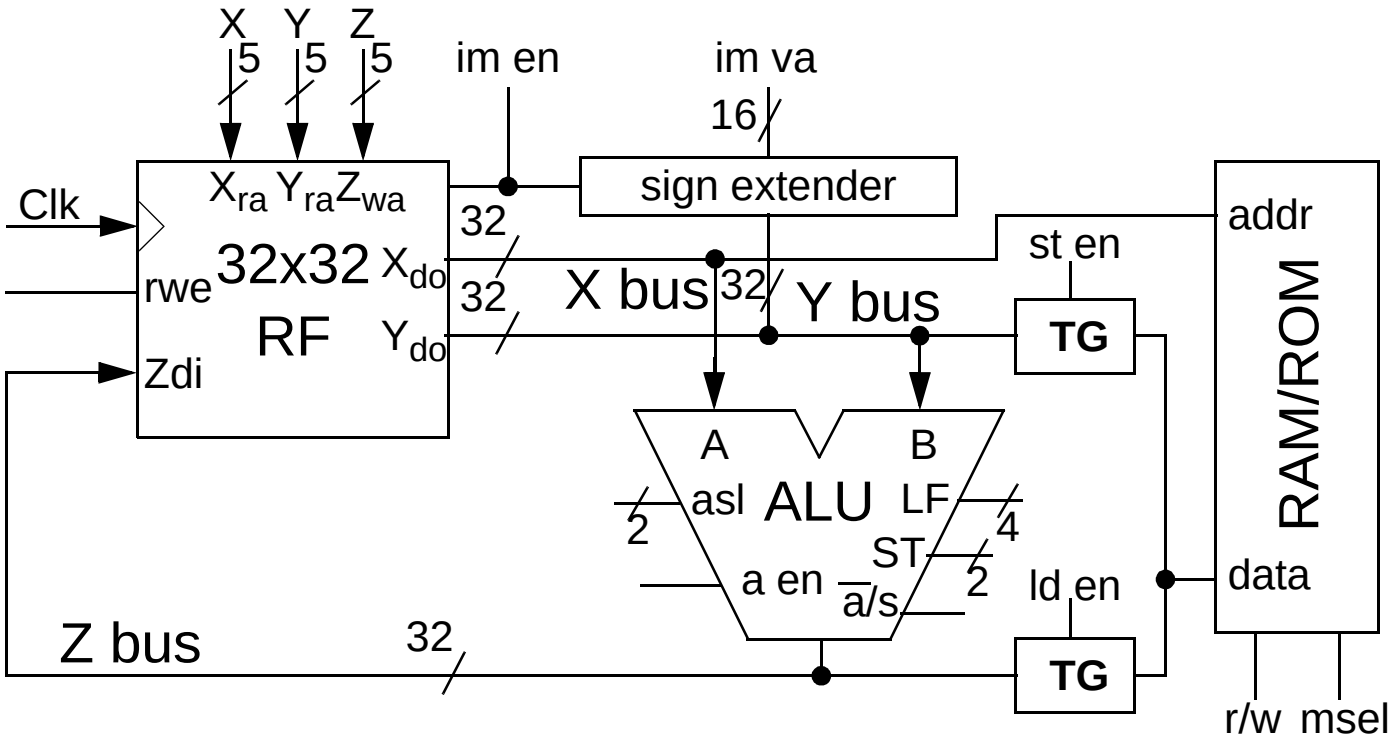
5. INSTRUCTIONS *(MAX MARK: 20)*

Execute the following assembly language program (for reference, a short table of assembly language instructions is given at the bottom of the page). For each clock cycle in the execution, fill in the register values in the table below with the computed values at the end of the clock cycle. Assume that all registers start initially with an unknown value so are marked with an "X" for unknown (i.e. at clock cycle 0).

```
andi $5, $5, 0x0000
addi $3, $5, 0x0017
andi $1, $1, 0x0000
add $2, $3, $3
ori $4, $1, 0x0400
or $1, $4, $3
sub $5, $1, $3
```

Clock Cycle	R1	R2	R3	R4	R5
0	X	X	X	X	X
1	X	X	X	X	0x00000000
2	X	X	0x00000017	X	0x00000000
3	0x00000000	X	0x00000017	X	0x00000000
4	0x00000000	0x0000002E	0x00000017	X	0x00000000
5	0x00000000	0x0000002E	0x00000017	0x00000400	0x00000000
6	0x00000417	0x0000002E	0x00000017	0x00000400	0x00000000
7	0x00000417	0x0000002E	0x00000017	0x00000400	0x00000400

Instruction	Example	Meaning
add	**add $1,$2,$3**	**$1 = $2 + $3**
add immediate	**addi $1,$2,100**	**$1 = $2 + 100**
subtract	**sub $1,$2,$3**	**$1 = $2 - $3**
subtract immediate	**subi $1,$2,100**	**$1 = $2 - 100**
and	**and $1,$2,$3**	**$1 = $2 & $3**
and immediate	**andi $1,$2,100**	**$1 = $2 & 100**
or	**or $1,$2,$3**	**$1 = $2 | $3**
or immediate	**ori $1,$2,100**	**$1 = $2 | 100**



For all enables, 0=disable and 1=enable

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 register enable
im va - immediate value
st en - store enable
ld en - load enable
r/w - memory read/write (read=0,write=1)
msel - memory select

a en - ALU enable
LF - 4 bit logical function
ST - 2 bit shift type
a/s - add or subtract (add=0, sub=1)
asl - arithmetic unit, shift unit, logical unit select

asl
00 - select arithmetic unit
01 - select shift unit
10 - select logical unit

Shift Types
00 = Logical
01 = Arithmetic
10 = Rotate

Shift Direction (B on ALU)
positive B = shift right
negative B = shift left

Logical Function

A	B	LF
0	0	LF_0
0	1	LF_1
1	0	LF_2
1	1	LF_3