ECE2020 A Fall 2018 Test 4

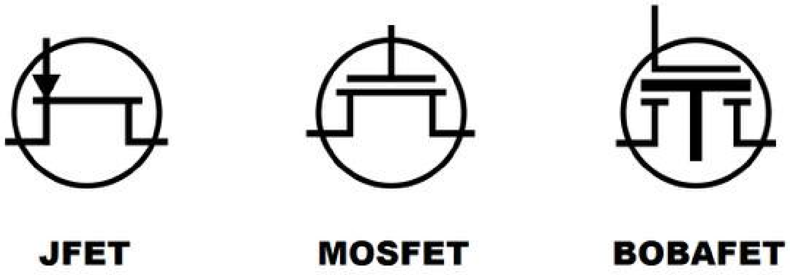
Name: ____

- Only a writing implement may be used on this exam (i.e. no books, notes, or any electronics).
- If the meaning of any question is not clear, please ask for clarification.
- Partial credit can only be awarded for work shown.

Honor pledge:

On my honor, I pledge that I will neither receive nor provide improper assistance in the completion of this test. I understand and accept my responsibility as a member of the Georgia Tech Community to uphold the Honor Code at all times, and I know that I have options for reporting honor violations at osi.gatech.edu.

GTID: ____________________________ Signature: ______________________________________



Boolean Identities

Identity	$A + 0 = A$	$A \cdot 1 = A$
Dominance	$A + 1 = 1$	$A \cdot 0 = 0$
Idempotence	$A + A = A$	$A \cdot A = A$
Inverse	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
Commutative	$A + B = B + A$	$A \cdot B = B \cdot A$
Associative	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Distributive	$A \cdot (B + C) = A \cdot B + A \cdot C$	$A + B \cdot C = (A + B) \cdot (A + C)$
Absorption	$A \cdot (A + B) = A$	$A + A \cdot B = A$
DeMorgan’s	$\overline{(A + B)} = \overline{A} \cdot \overline{B}$	$\overline{(A \cdot B)} = \overline{A} + \overline{B}$
Double Complement	$\overline{\overline{A}} = A$	
FOIL	$(A + B) \cdot (C + D) = A \cdot C + A \cdot D + B \cdot C + B \cdot D$	
Disappearing opposite	$A + \overline{A} \cdot B = A + B$	

Decimal	Binary	Hex
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

2^{0}	1
2^{1}	2
2^{2}	4
2^{3}	8
2^{4}	16
2^{5}	32
2^{6}	64
2^{7}	128
2^{8}	256
2^{9}	512
2^{10}	1024
2^{11}	2048
2^{12}	4096
2^{13}	8192
2^{14}	16,384
2^{15}	32,768
2^{20}	1,048,576

8-channel Multiplexer (inputs A_{7-0}, output Q

S2	S1	S0	Q
0	0	0	A_0
0	0	1	A_1
0	1	0	A_2
0	1	1	A_3
1	0	0	A_4
1	0	1	A_5
1	1	0	A_6
1	1	1	A_7

3-to-8 Line Decoder with Enable

A2	A1	A0	EN	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0
X	X	X	0	0	0	0	0	0	0	0	0

8-channel Demultiplexer (input A, outputs Q_{7-0})

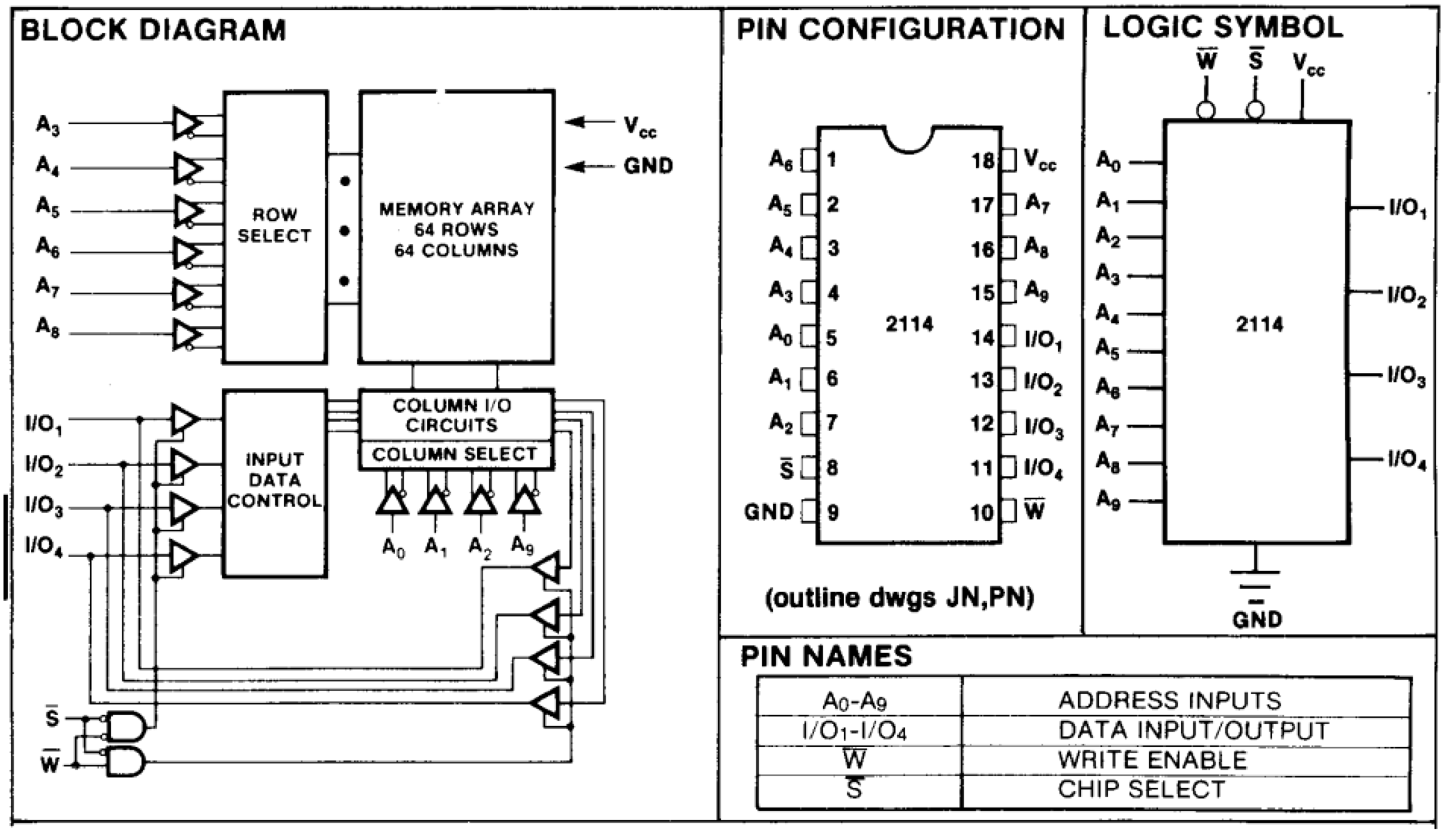
S2	S1	S0	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0	0	0	0	0	0	0	0	0	0	A
0	0	1	0	0	0	0	0	0	A	0
0	1	0	0	0	0	0	0	A	0	0
0	1	1	0	0	0	0	A	0	0	0
1	0	0	0	0	0	A	0	0	0	0
1	0	1	0	0	A	0	0	0	0	0
1	1	0	0	A	0	0	0	0	0	0
1	1	1	A	0	0	0	0	0	0	0

8-to-3 Priority Encoder (Priority A7->A0)

A7	A6	A5	A4	A3	A2	A1	A0	E2	E1	E0	Ac
1	X	X	X	X	X	X	X	1	1	1	1
0	1	X	X	X	X	X	X	1	1	0	1
0	0	1	X	X	X	X	X	1	0	1	1
0	0	0	1	X	X	X	X	1	0	0	1
0	0	0	0	1	X	X	X	0	1	1	1
0	0	0	0	0	1	X	X	0	1	0	1
0	0	0	0	0	0	1	X	0	0	1	1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0

Problem 1 (25 points)

Here is some information about the 2114 RAM chip, which was used for video color RAM in the Commodore 64.



A0-A9	ADDRESS INPUTS
I/O1-I/O4	DATA INPUT/OUTPUT
W̄	WRITE ENABLE
S̄	CHIP SELECT

a) How many words of memory does this chip contain?

b) How many bits are in each word?

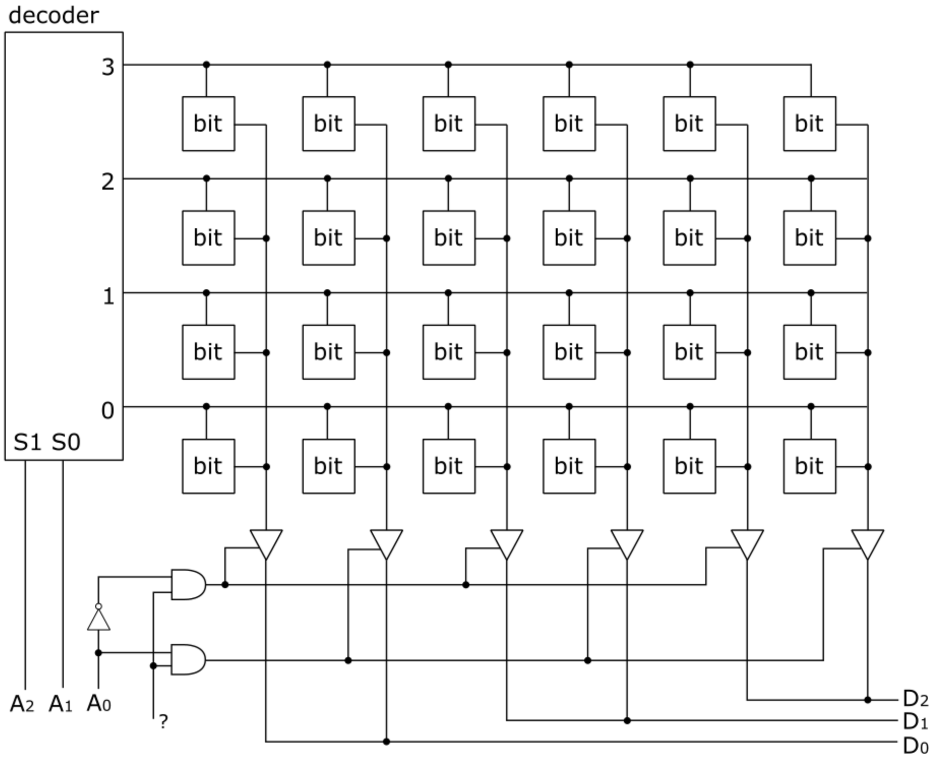
c) In the block diagram, the memory array is described as 64 rows and 64 columns. Is that referring to the words or to the bit cells?

d) How would the memory array be described if it was the opposite of your answer from #3? I.e. if you said it was referring to the bit cells, what would the description be if it was referring to the words, and vice versa.

e) If a different chip contained four times as many words, how many <u>additional</u> address pins would be needed?

f) Based on the block diagram, what combination(s) of S̄ and W̄ will tri-state the I/O pins? Assume that the tri-state drivers are active when their control signal is 1.

Problem 2 (10 points)



a) Based on the memory devices we've seen in this class, what is an appropriate name for the signal marked with a question mark in the bottom left of the diagram?

b) Color in the cells in the diagram above that contain the data at address 6.

The datapath reference is on the last page. You can tear it off if you want.

Problem 3 (15 points)

In the MIPS datapath, all of the following effectively implement `R2 = R1`:

- `R2 = R1 OR 0`
- `R2 = R1 OR R0`
- `R2 = R1 + 0`
- `R2 = R1 + R0`
- Setting st_en and ld_en to 1, disabling the arithmetic, logic, and shift units and memory, and using the Y bus to transfer R1 to R2.

Describe two other unique ways to accomplish `R2 = R1` in the MIPS datapath. You can use existing instructions, or you can describe what the datapath components should do or what the control signals should be.

1)

2)

The datapath diagram is on the last page. You can tear it off if you want.

Problem 4 (25 points)

4a) Use the description column to fill in the datapath signals to implement that operation.

				im		au	-a	lu		su		st	ld	-r		
X	**Y**	**Z**	**rwe**	**en**	**im_va**	**en**	**/s**	**en**	**lf$_{3-0}$**	**en**	**st**	**en**	**en**	**/w**	**msel**	**description**
																MEM[R3] = R7
																R7 = R2 SRL 3

4b) From the provided microcode, write a description of the operation, either as an assembly code instruction or as a plain-English description. Simplify the description of the operation if possible – for example, like in problem 3, if the datapath performs `R2 = R1 OR 0`, it would be better to write `R2 = R1` or "R2 gets the value of R1". Note that everything not bolded is the same in all rows. Remember that the immediate is sign-extended.

				im		au	-a	lu		su		st	ld	-r		
X	**Y**	**Z**	**rwe**	**en**	**im_va**	**en**	**/s**	**en**	**Lf$_{3-0}$**	**en**	**st**	**en**	**en**	**/w**	**msel**	**description**
3	2	1	1	**1**	**FFFF$_{16}$**	**0**	0	**1**	**1110**	0	0	0	0	0	0	
3	2	1	1	**1**	**0**	**0**	0	**1**	**1111**	0	0	0	0	0	0	
3	2	1	1	**1**	**0**	**1**	0	**0**	**0000**	0	0	0	0	0	0	
3	2	1	1	**0**	**FFFF$_{16}$**	**1**	0	**0**	**0000**	0	0	0	0	0	0	

Problem 5 (25 points)

- Assume that R1 contains a value between 0 and 31, and consider that the "input" to the code.
- The "output" is in R5 after the code completes.

```
R2 = 32                 #
R2 = R2 - R1            #
R5 = -1                 #
R5 = R5 SRL R2          # shift by amount stored in R2
```

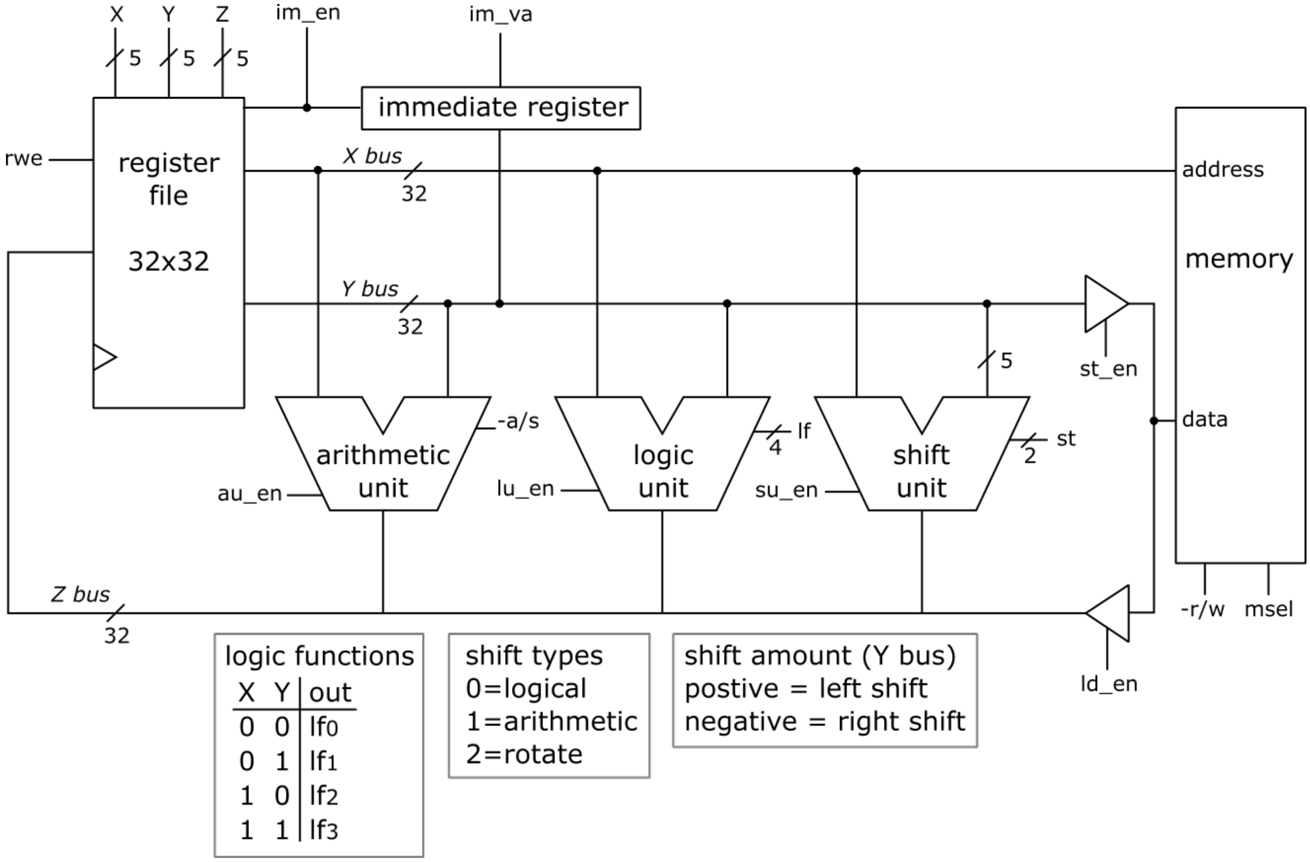
a) Describe the result of the code in terms of the input and output. Hint: the operation is mathematical in nature, although it can be also described in terms of the bits if you don't see the mathematical connection.

b) The author of the code forgot that the shift amount for the MIPS shifter is 5 bits (refer to the datapath reference). It comes from the least-significant 5 bits of the Y bus. How does that affect the behavior of the code for the intended input range of 0-31?

(note that you can ignore any related incorrect behavior in your answer to part a)

Bonus (1 possible bonus point)

If you've finished early, fix the code so that it works for all input values 0-31. You can write actual code, or you can describe the fix as long as it's clear how it would be translated to assembly code.



logic functions

X	Y	out
0	0	lf0
0	1	lf1
1	0	lf2
1	1	lf3

shift types
0=logical
1=arithmetic
2=rotate

shift amount (Y bus)
postive = left shift
negative = right shift

Signal	Description	Signal	Description
X, Y	Read register addresses	lf	Logic function (see table below diagram))
Z	Write register address	su_en	Shift unit enable
rwe	Register write enable	st	Shift type (see table below diagram)
im_en	Immediate enable	st_en	Store enable
im_va	Immediate value	ld_en	Load enable
au_en	Arithmetic unit enable	-r/w	Read/write memory (0=read, 1=write)
-a/s	Add/subtract (0=add, 1=subtract)	msel	Memory select (memory output enable)
lu_en	Logic unit enable		

instruction	RTL description
add	$d = $s + $t;
subtract	$d = $s - $t;
add immediate	$t = $s + imm;
and	$d = $s AND $t;
or	$d = $s OR $t;
xor	$d = $s XOR $t;
and immediate	$t = $s AND imm;
or immediate	$t = $s OR imm;
xor immediate	$t = $s XOR imm;
shift left logical	$d = $t SLL a;
shift right logical	$d = $t SRL a;
shift left arithmetic	$d = $t SLA a;
shift right arithmetic	$d = $t SRA a;
load word	$t = MEM[$s];
store word	MEM[$s] = $t;
load immediate	$t = imm;
branch if equal	IF $s = $t GOTO label;
branch if not equal	IF $s != $t GOTO label;
set if less than	$d = $s < $t; (if $s < $t $d = 1; else $d = 0;)
set if less than immediate	$d = $s < imm; (if $s < imm $t = 1; else $t = 0;)
jump	GOTO label;