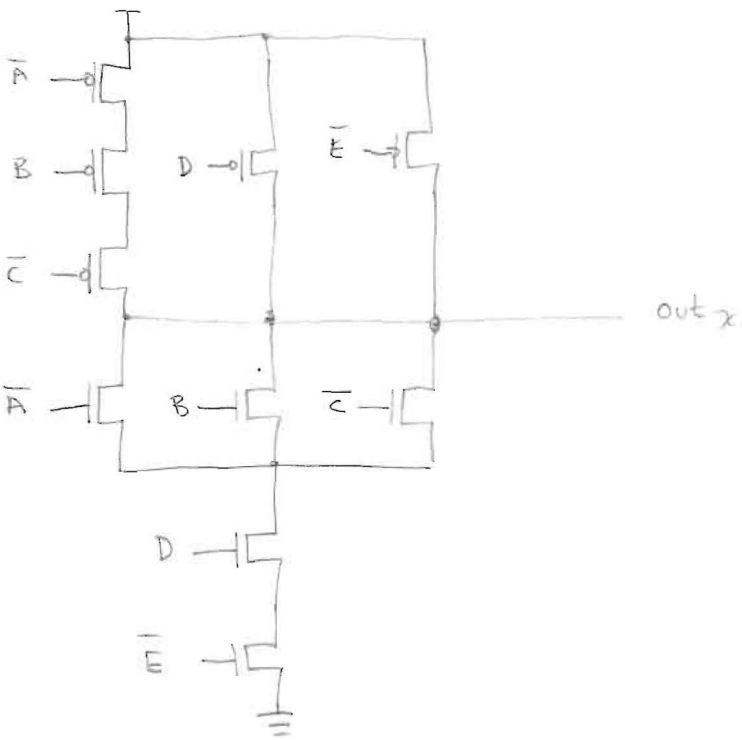
ECE2020 Final Exam Summer 2014 GTL
July 30, 2014 Name: KEY

Show your work for any possible partial credit. 100 possible points, 8 exam pages plus two supplemental pages.

High Level Language		
Assembly Language		
Instruction Set		
Memory	Data Path	Controller
Storage	Functional Units	State Machines
Building Blocks		
Gate		
Switches and Wires		

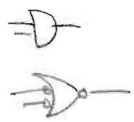
1) (10 points) For the expression below, create a switch level implementation using N and P type switches. Assume both inputs and their complements are available. Your design should contain no shorts or floats. Implement the equation exactly as is (no simplifying).

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

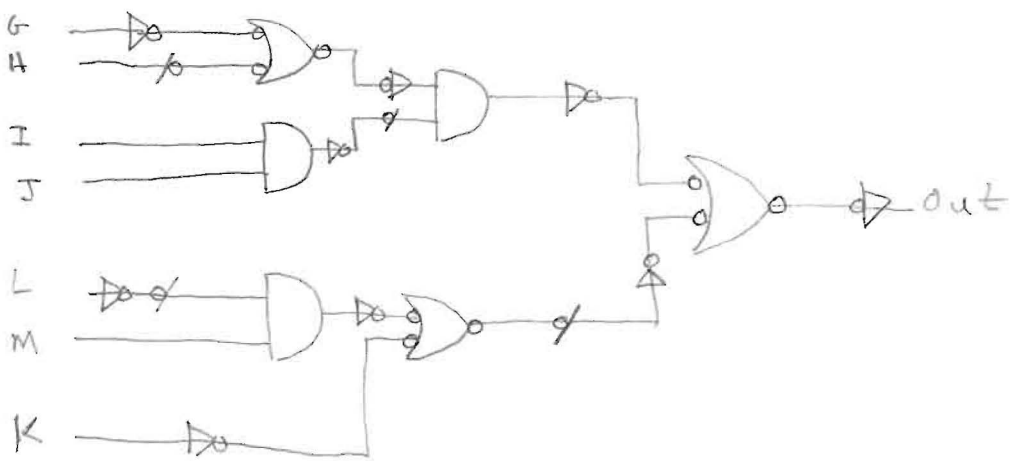


High Level Language		
Assembly Language		
Instruction Set		
Memory	Data Path	Controller
Storage	Functional Units	State Machines
Building Blocks		
Gate		
Switches and Wires		

2) (10 points) Implement the following expression using only AND gates and inverters. Then determine the number of switches required. Use proper mixed logic notation. Do not modify the expression. Do not assume compliments of inputs are available. You may use 3 input AND gates if needed.



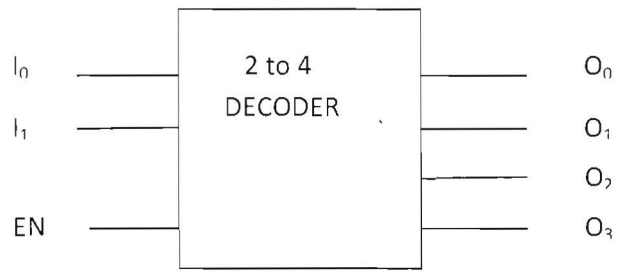
Out = $(G + \overline{H})\overline{(I \cdot J)} + \overline{K + \overline{L} \cdot M}$



6 AND x 6 + 9 inverters x 2

Number of switches 54

High Level Language		
Assembly Language		
Instruction Set		
Memory	Data Path	Controller
Storage	Functional Units	State Machines
Building Blocks		
Gate		
Switches and Wires		



3) (10 points) Implement a 2 to 4 decoder using AND gates and inverters, (you may use 3 input AND Gates). You must show a truth table, boolen equations for each of the four outputs, and then the implementation of the 2 to 4 decoder.

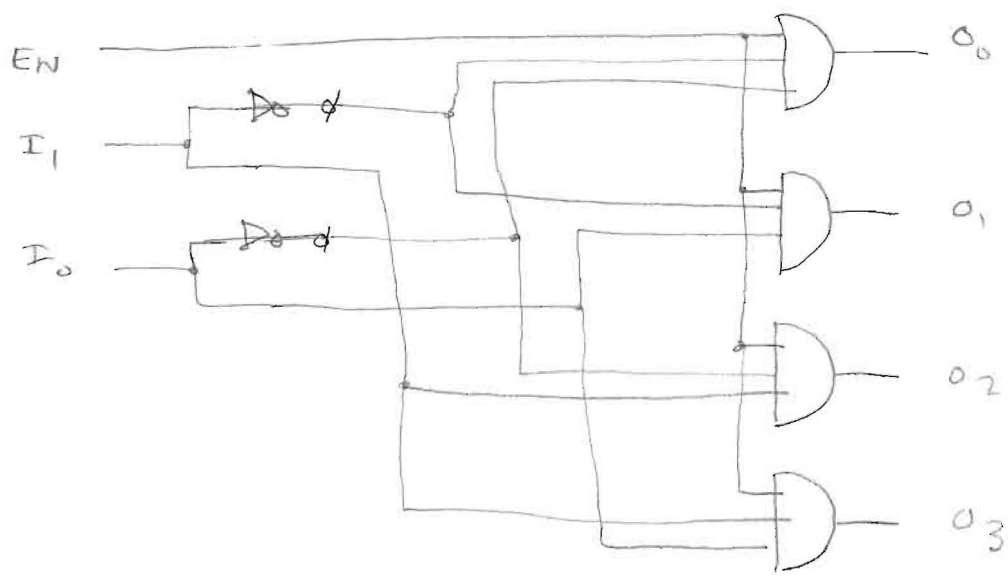
EN	I_1	I_0	O_0	O_1	O_2	O_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$O_0 = EN \cdot \overline{I_1} \cdot \overline{I_0}$

$O_1 = EN \cdot \overline{I_1} \cdot I_0$

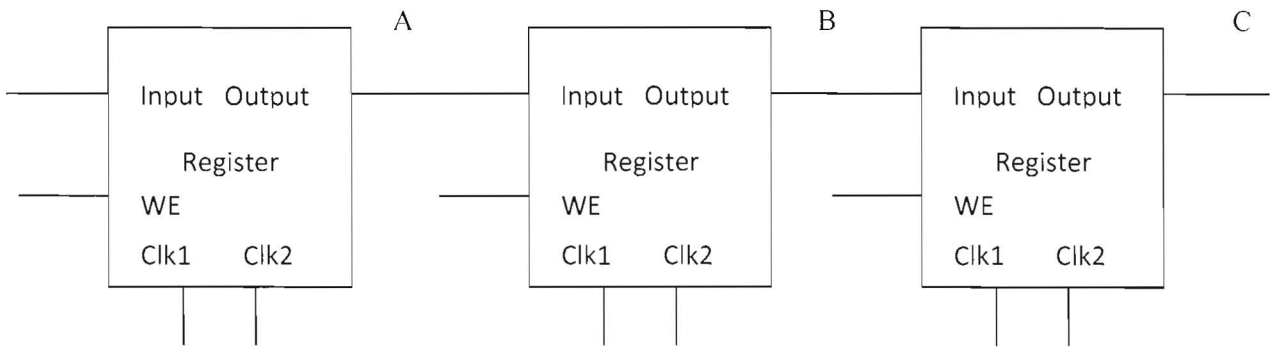
$O_2 = EN \cdot I_1 \cdot \overline{I_0}$

$O_3 = EN \cdot I_1 \cdot I_0$

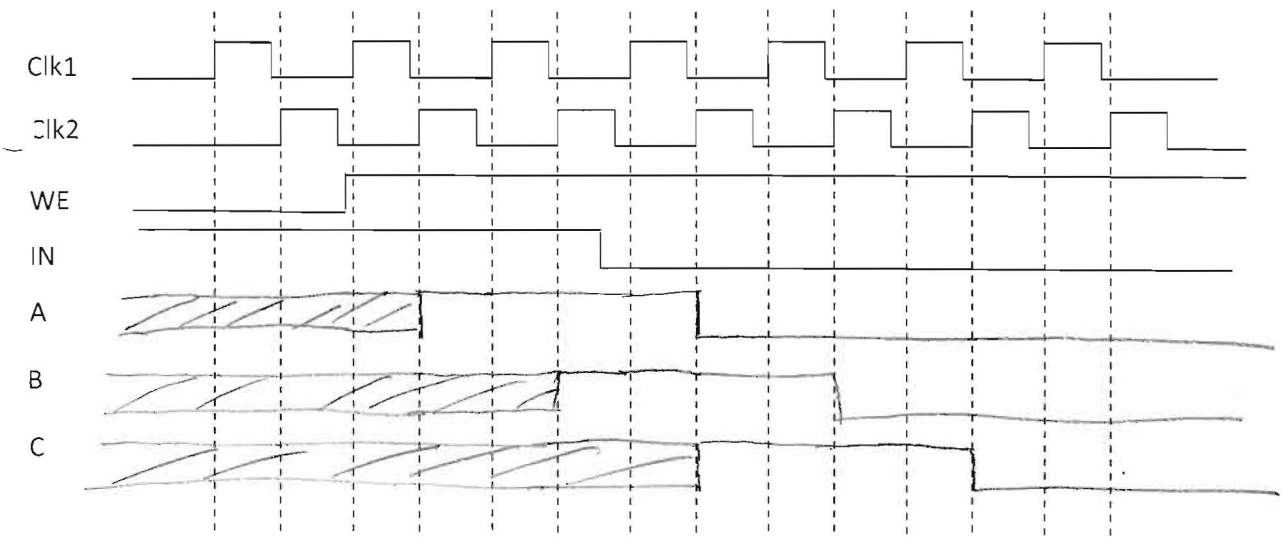


High Level Language		
Assembly Language		
Instruction Set		
Memory	Data Path	Controller
Storage	Functional Units	State Machines
Building Blocks		
Gate		
Switches and Wires		

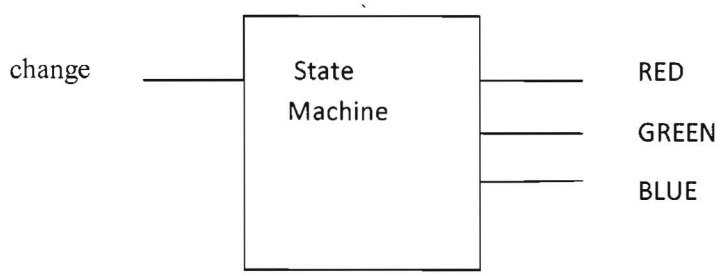
4) (10 points) Consider the register implementation below.



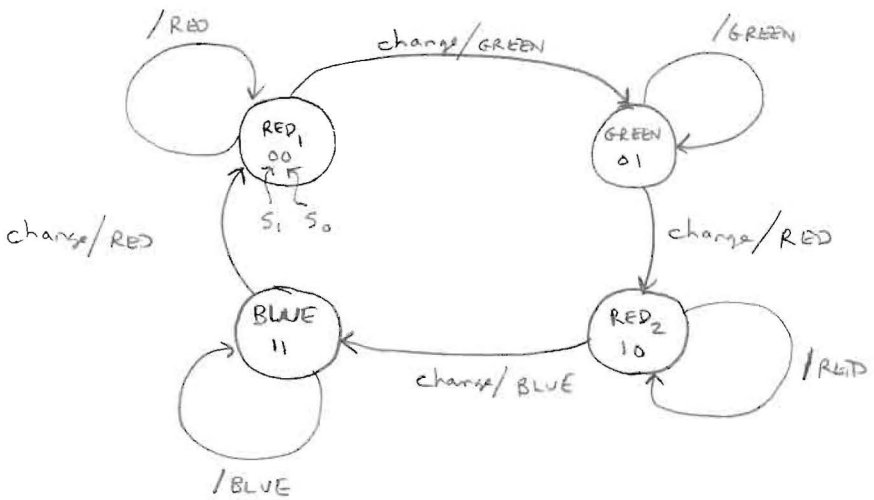
Assume the following signals are applied to the registers above. Draw the signal at point A (output of the first register), point B (output of the second register) and point C (output of the third register). Assume all stored signals are UNKNOWNS at start.



High Level Language		
Assembly Language		
Instruction Set		
Memory	Data Path	Controller
Storage	Functional Units	**State Machines**
Building Blocks		
Gate		
Switches and Wires		

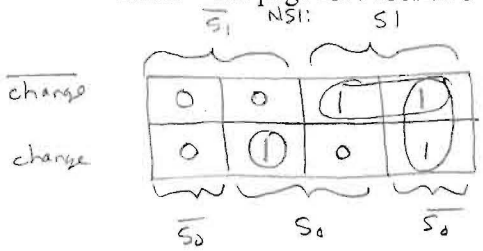


5) (20 points total) Implement a state machine that outputs three signals names RED, GREEN, and BLUE and with the single input change. As long as change input is a high, the outputs go high one at a time in this specific order RED, GREEN, RED, BLUE, RED, GREEN ,RED, BLUE, ...however when the change input is a low, the output that is high at present stays high until the change input is set to a high. Draw a state diagram, show a state table, and show the complete and entire state machine design. You may use any type of logic gate you want.

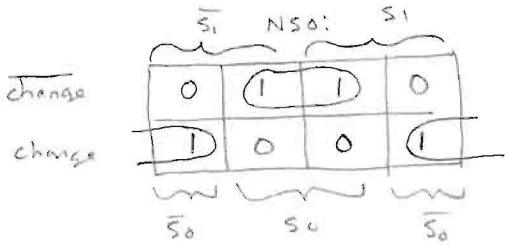


CHANGE	S_1	S_0	NS_1	NS_0	RED	GREEN	BLUE
0	0	0	0	0	1	0	0
1	0	0	0	1	0	1	0
0	0	1	0	1	0	1	0
1	0	1	1	0	1	0	0
0	1	0	1	0	1	0	0
1	1	0	1	1	0	0	1
0	1	1	1	1	0	0	1
1	1	1	0	0	1	0	0

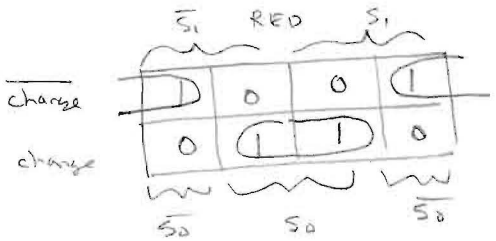
Blank work page for Problem 5



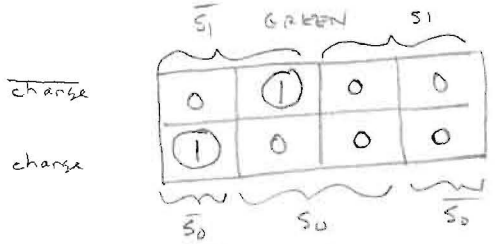
$NS1 = change\ \overline{S1}\ S0 + \overline{change}\ S1 + S1\overline{S0}$



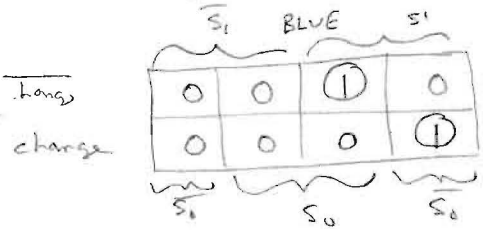
$NS0 = \overline{change}\ S0 + change\ \overline{S_0}$



$RED = change\ S_0 + \overline{change}\ \overline{S_0}$

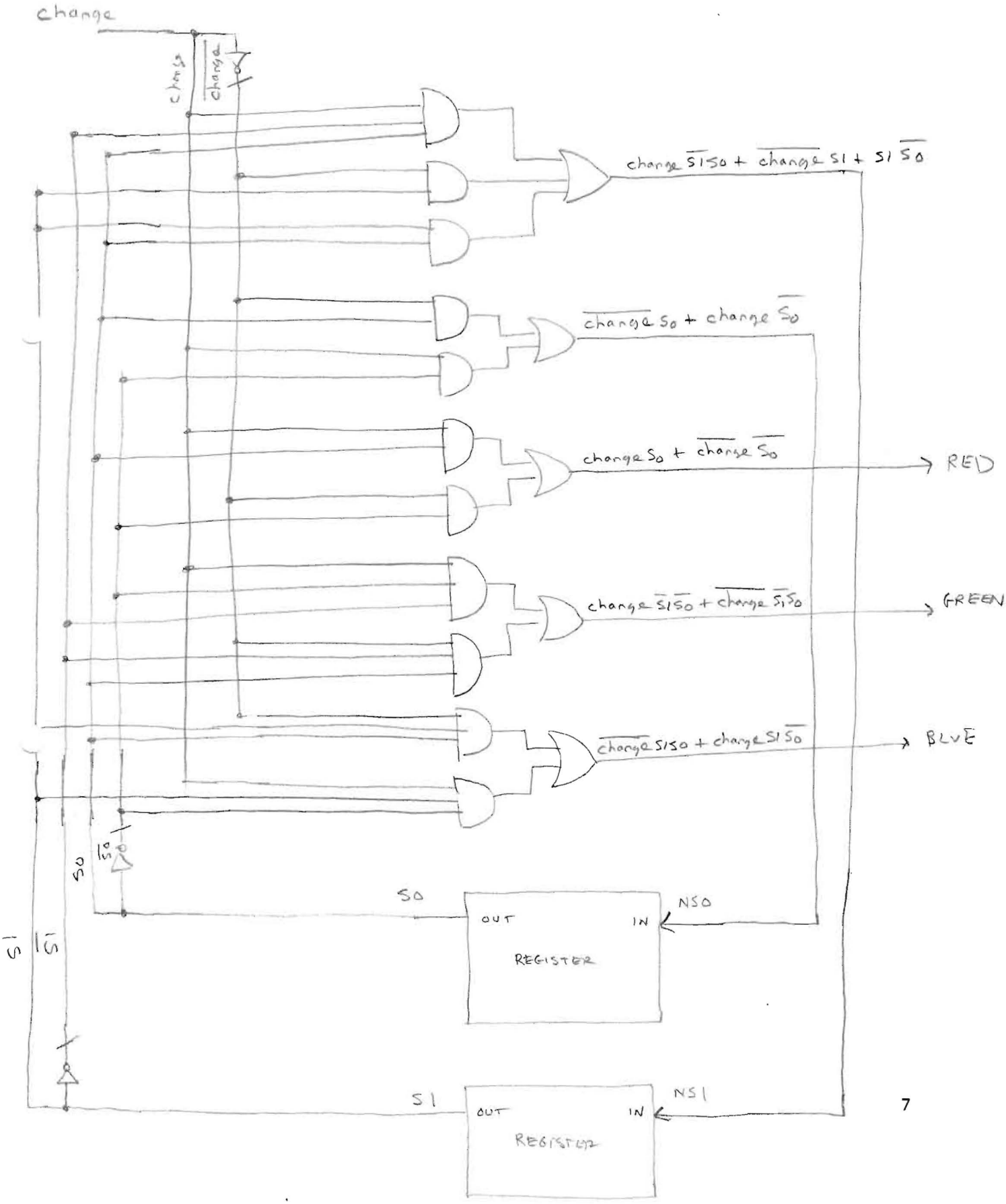


$GREEN = change\ \overline{S_1}\ \overline{S_0} + \overline{change}\ \overline{S_1}\ S_0$



$BLUE = \overline{change}\ S1S_0 + change\ S1\ \overline{S_0}$

Blank work page for Problem 5



High Level Language		
Assembly Language		
Instruction Set		
Memory	Data Path	Controller
Storage	Functional Units	State Machines
Building Blocks		
Gate		
Switches and Wires		

6) (10 points) Using the attached single cycle data path we have discussed extensively in class (see last pages of exam for part of it), using as few registers as possible, implement the microcode for the following operation

M[R5] = (R3 + 8 R2 + 5M[R4])/4

#	X	Y	Z	rwe	imen	im va	auen	-a/s	luen	lf	suen	st	lden	sten	r/-w	msel	description
1	2	X	2	1	1	-3	0	X	0	XXXX	1	00	0	0	X	0	R2=BR2 LOGICAL SHIFT -3 left
2	2	3	3	1	0	X	1	0	0	XXXX	0	XX	0	0	X	0	R3=R3+R2
3	4	X	2	1	0	X	0	X	0	XXXX	0	XX	1	0	1	1	R2=M[R4]
4	2	3	3	1	0	X	1	0	0	XXXX	0	XX	0	0	X	0	R3=R3+R2
5	2	X	2	1	1	-2	0	X	0	XXXX	1	00	0	0	X	0	R2=R2 LOGICAL SHIFT -2
6	2	3	3	1	0	X	1	0	0	XXXX	0	XX	0	0	X	0	R3=R3+R2
7	3	X	3	1	1	+2	0	X	0	XXXX	1	01	0	0	X	0	R3=R3 ARITHMETIC SHIFT +2
8	5	3	X	0	0	X	0	X	0	XXXX	0	XX	0	1	0	1	M[R5]=R3
9																	

High Level Language		
Assembly Language		
Instruction Set		
Memory	Data Path	Controller
Storage	Functional Units	State Machines
Building Blocks		
Gate		
Switches and Wires		

7) (10 points) Instruction Format. Suppose you were designing a new computer with a 64 bit instruction word and a 64 bit addressed memory, 2048 different instructions, and 1024 registers. What are the number of bits required in each of the different required fields in the Register Format Instructions? (You may have more or less lines in the table below than you need).

Field Name	Minimum Number of bits required
OPCODE	11 BITS
Z	10 BITS
X	10 BITS
Y	10 BITS
TO BE DEFINED BITS	64 − 41 = 23

$2048 \Rightarrow 2^{11}$

$1024 \Rightarrow 2^{10}$

8) (20 points) MIPS Assembly language programming. Write a MIPS procedure named search that will take memory value starting at the location contained in register $4 and search sequential memory locations until it finds the value negative one. Once it finds the value negative one in a memory location, put the address of the memory location that contains the negative one into register $5 and return back to the program that called your procedure. You may need more or fewer rows in the tables below.

Label:	Instruction	Comment
SEARCH	addi $6, $0, -1	STORE -1 FOR COMPARISONS
repeat	lw $7 ($4)	PUT FIRST MEMORY VALUE INTO $7
	beq $7, $6, DONE	IF FOUND VALUE DONE
	addi $4, $4, 4	INCREMENT $4 BY 4 BYTES
	jmp repeat	KEEP DOING
DONE	add $5, $0, $4	PUT CURRENT MEMORY ADDRESS IN $5
	jr $31	RETURN TO PC = $31

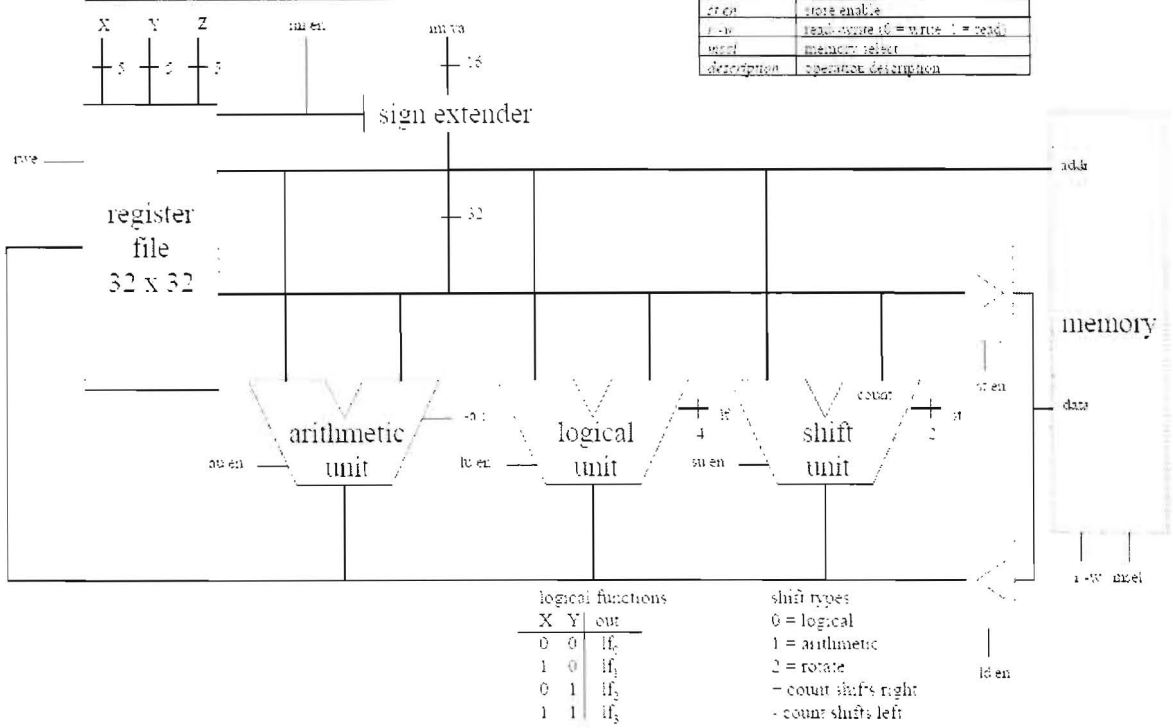
Explain the use of ALL of the registers used in your code:

Register	Was used for the purposes of
$4	Starting location of memory
$5	Contains memory location where -1 was found
$6	STORE -1 TO COMPARE TO
$7	STORE EACH MEMORY VALUE TESTING

You may tear off the following 2 pages from the exam and you do not need to turn in them in.

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
au f	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)
mem	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

instruction	example	meaning
add	`add $1,$2,$3`	`$1 = $2 + $3`
subtract	`sub $1,$2,$3`	`$1 = $2 - $3`
add immediate	`addi $1,$2,-30`	`$1 = $2 + -30`
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,-30`	`$1 = $2 & 226`
or immediate	`ori $1,$2,110`	`$1 = $2 | 110`
xor immediate	`xori $1,$2,-30`	`$1 = $2 xor -30`
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, (40)`	`$1 = memory [40]`
store word	`sw $1, (40)`	`memory [$1] = $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, -11`	`if ($2 < -11), $1 = 1 else $1 = 0`
jump	`j 10000`	`PC = 10000*4`
jump register	`jr $31`	`PC = $31`
jump and link	`jal 10000`	`$31 = PC + 4; PC = 10000*4`

K MAPS: