NUID: Name (Last, First): Key

CSCE 230, Fall 2017
Final Exam

Notes:

- This is a closed notes/book exam. You may use a scientific or graphing calculator.
- **This is an individual exam. Please ensure all work is your own.**
- Enter your name, last name first, on the line above. A point will be lost to those who do so incorrectly. Do not open until told to do so.
- This is a comprehensive exam of all subjects discussed through the semester.

Rubric:

Problem Number	Possible Points	Points Earned
1	10	
2	15	
3	15	
4	10	
5	30	
6	10	
7	20	
8	20	
Total	130	

Problem 1 [10]

Topic: *Number and Data Representations*

Complete the following table, the first row has been completed as an example. Notice that negative is not the negation of but the mathematical negative

(8) Raw Bits	Octal	Sign Extended 2's Comp	Unsigned Decimal	Hex	ASCII	2's Comp Decimal	Negative Decimal	Negative Binary
01000100	104	0000000001000100	68	44	D	68	-68	10111100
11000100	304	1111111111000100	65476	C8	-ve D	-60	60	00111100
00100110	046	0000000000100110	38	26	&	38	-38	11011010
11101100	354	1111111111101100	65516	EC	∞ or ∂	-20	20	00010100

Regular ASCII Chart (character codes 0 - 127)

000	(nul)	016 ►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001 ☺	(soh)	017 ◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002 ☻	(stx)	018 ↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003 ♥	(etx)	019 ‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004 ♦	(eot)	020 ¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005 ♣	(enq)	021 §	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006 ♠	(ack)	022 ▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007 •	(bel)	023 ↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008 ◘	(bs)	024 ↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025 ↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011 ♂	(vt)	027 ←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012 ♀	(np)	028 ∟	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029 ↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014 ♫	(so)	030 ▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015 ☼	(si)	031 ▼	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂

Extended ASCII Chart (character codes 128 - 255)

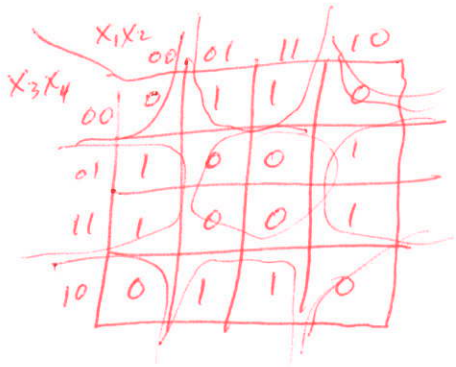
128	Ç	143	Å	158	₧	172	¼	186	║	200	╚	214	╓	228	Σ	242	≥
129	ü	144	É	159	ƒ	173	¡	187	╗	201	╔	215	╫	229	σ	243	≤
130	é	145	æ	160	á	174	«	188	╝	202	╩	216	╪	230	µ	244	⌠
131	â	146	Æ	161	í	175	»	189	╜	203	╦	217	┘	231	τ	245	⌡
132	ä	147	ô	162	ó	176	▒	190	╛	204	╠	218	┌	232	Φ	246	÷
133	à	148	ö	163	ú	177	▓	191	┐	205	═	219	█	233	Θ	247	≈
134	å	149	ò	164	ñ	178	▓	192	└	206	╬	220	▄	234	Ω	248	°
135	ç	150	û	165	Ñ	179	▓	193	┴	207	╧	221	▌	235	δ	249	·
136	ê	151	ù	166	ª	180	▓	194	┬	208	╨	222	▐	236	∞	250	·
137	ë	152	ÿ	167	º	181	▓	195	├	209	╤	223	▀	237	φ	251	√
138	è	153	Ö	168	¿	182	▓	196	─	210	╥	224	α	238	ε	252	ⁿ
139	ï	154	Ü	169	⌐	183	▓	197	┼	211	╙	225	ß	239	∩	253	²
140	î	155	¢	170	¬	184	▓	198	╞	212	╘	226	Γ	240	≡	254	■
141	ì	156	£	171	½	185	▓	199	╟	213	╒	227	π	241	±	255	
142	Ä	157	¥												

Problem 2 [15]

Topic: *Boolean Algebra and K-maps*

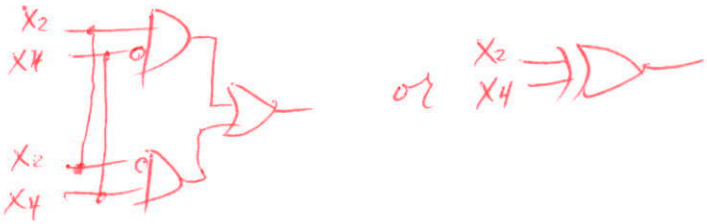
Simplify the following, using any method, into BOTH Sum of Products and Product of Sums. Then draw the combinational circuit, drawing the circuit in the minimum number of gates will grant 2 bonus points.

$$x_2\bar{x_4} + x_1\bar{x_2}x_3x_4 + \bar{x_2}\bar{x_3}x_4 + \bar{x_1}x_2\bar{x_4} + \bar{x_1}\bar{x_2}x_3x_4$$



SOP $X_2\bar{X_4} + \bar{X_2}X_4$

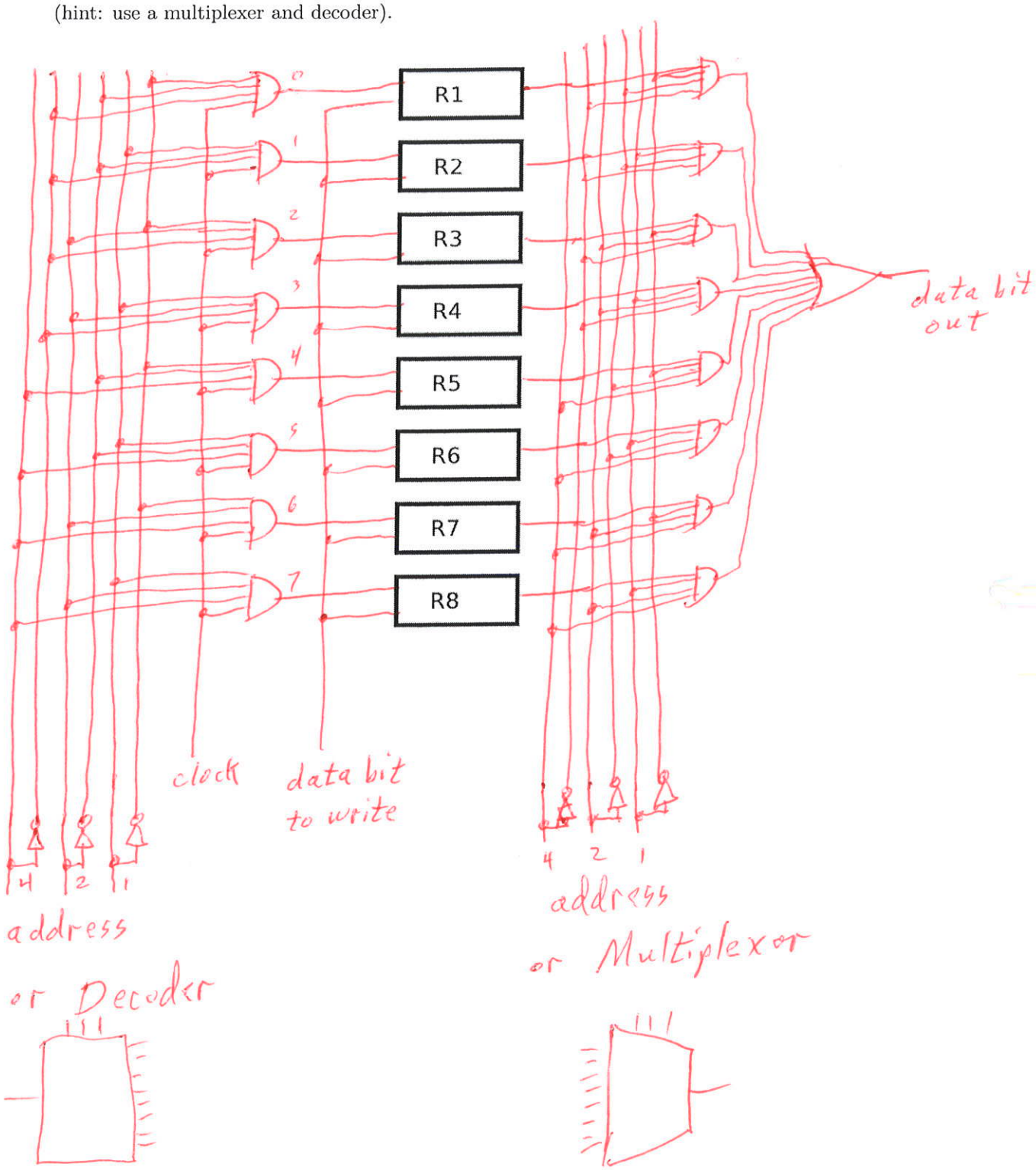
POS $(X_2+X_4)(\bar{X_2}+\bar{X_4})$



Problem 3 [15]

Topic: *Registers*

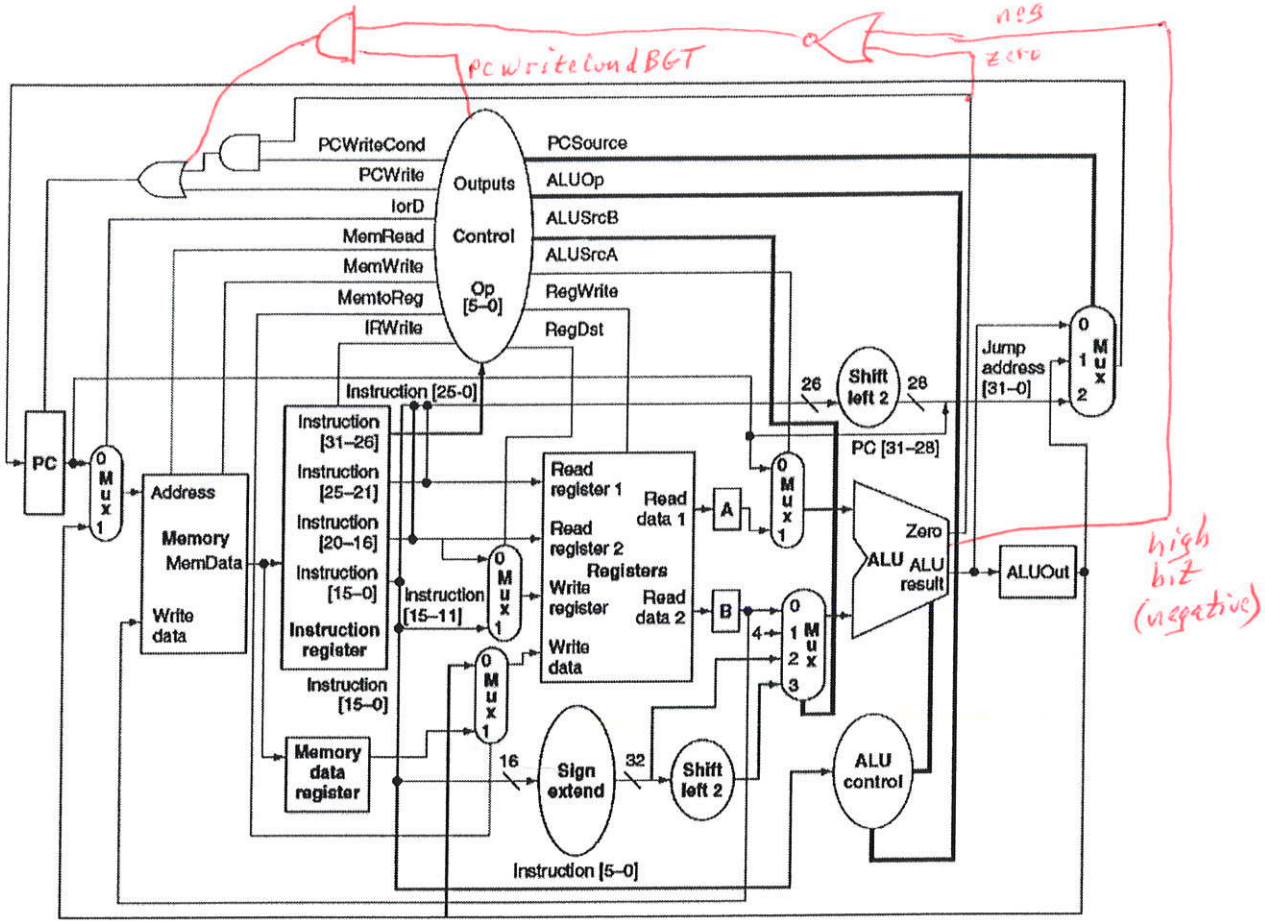
Given the one bit registers R1-R8 draw the necessary components and connections to read and write bits to registers. Clearly label all connections and components.
(hint: use a multiplexer and decoder).

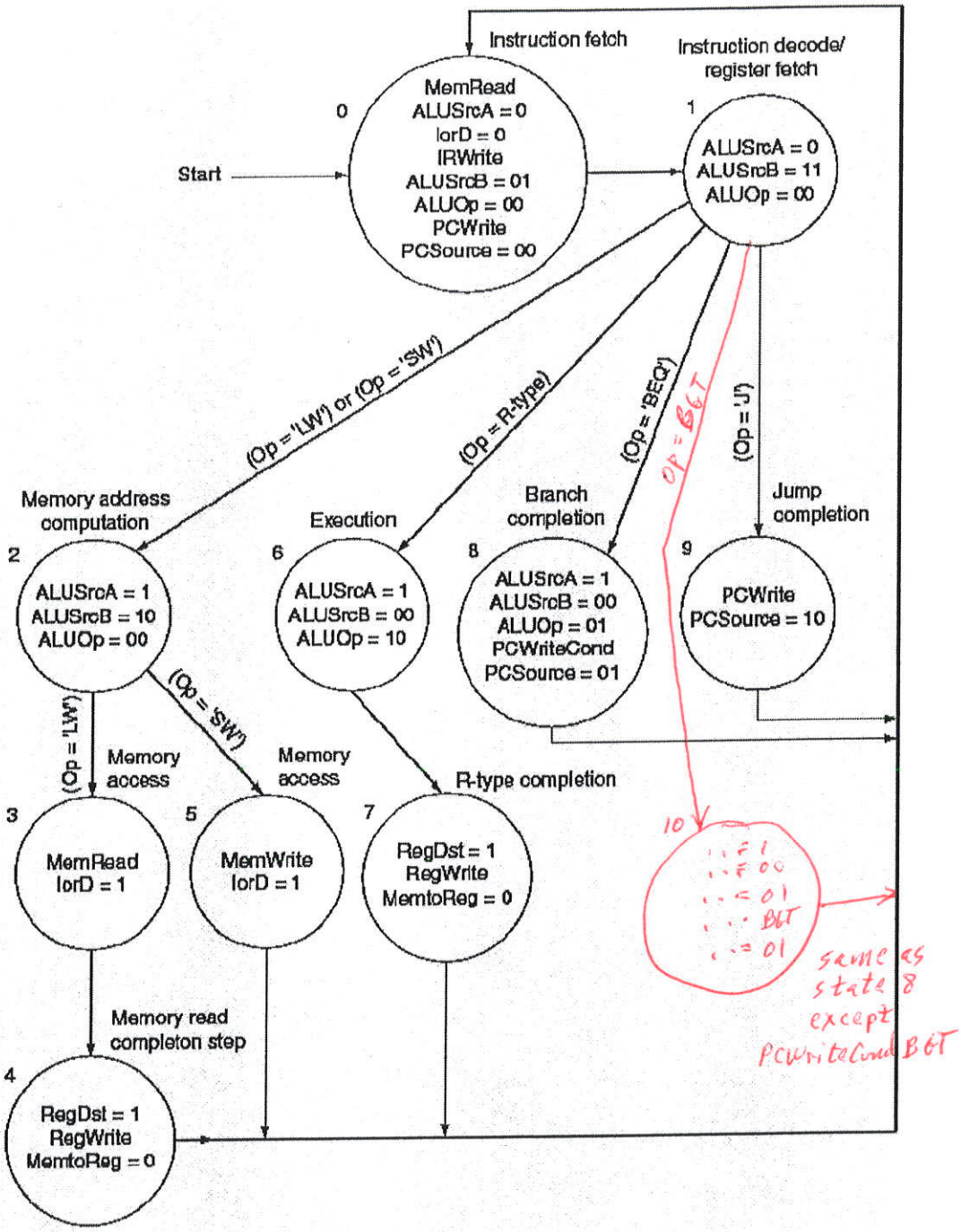


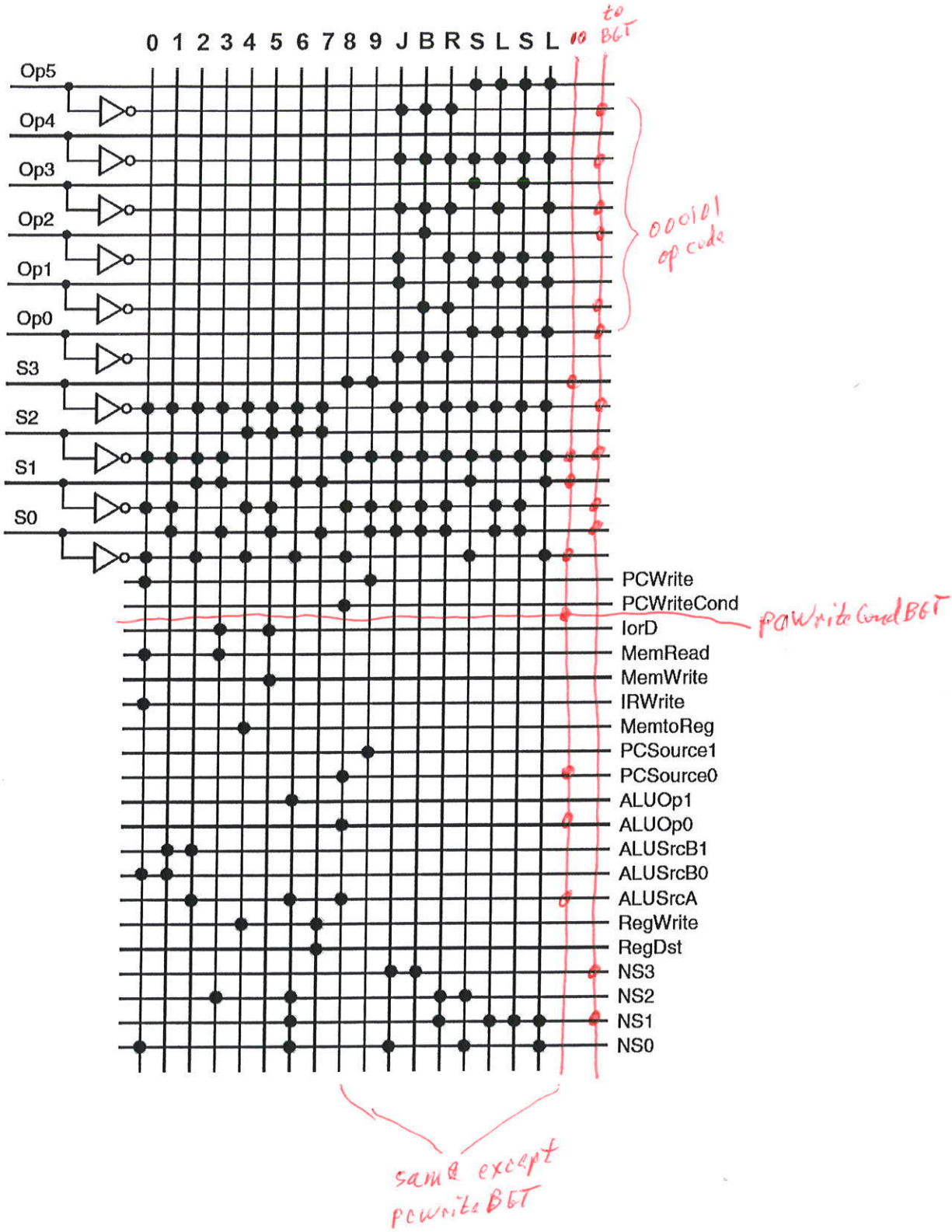
Problem 4 [10]

Topic: *Control Unit*

Using the diagrams on the following pages explain how to add the command "BGT" or branch if greater than. Feel free to draw on any of the diagrams and reference the drawings for a complete answer. Use the opcode 000101.




Instruction fetch
Instruction decode/
register fetch
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
Start
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
Memory address
computation
Execution
Branch
completion
Jump
completion
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
9
PCWrite
PCSource = 10
(Op = 'LW')
(Op = 'SW')
Memory
access
Memory
access
R-type completion
3
MemRead
IorD = 1
5
MemWrite
IorD = 1
7
RegDst = 1
RegWrite
MemtoReg = 0
Memory read
completon step
4
RegDst = 1
RegWrite
MemtoReg = 0
10
same as
state 8
except
PCWriteCond BGT


0 1 2 3 4 5 6 7 8 9 J B R S L S L 10
to BGT
Op5
Op4
Op3
Op2
Op1
Op0
S3
S2
S1
S0
000101 op code
PCWrite
PCWriteCond
IorD
MemRead
MemWrite
IRWrite
MemtoReg
PCSource1
PCSource0
ALUOp1
ALUOp0
ALUSrcB1
ALUSrcB0
ALUSrcA
RegWrite
RegDst
NS3
NS2
NS1
NS0
PCWriteCond BGT
same except PCWriteBGT

Problem 5 [30]

Topic: *Assembly*

Given the following pseudo-assembly (on the next page) code explain the purpose of each section of the code. Also give the C-Code that is equivalent for the ISR. Note that the C-Code should not just mirror the functionality but the implementation as well.

ISR:

- A. lines 1-3 Save registers in stack
- B. lines 4-6 Get keyboard char and store in buffer
- C. lines 7-8 Increment pointer to next position in buffer
- D. lines 9-12 Poll for display ready to echo char
- E. lines 13-18 If CR, we are done, set EOF flag + clear int keyboard enable
- F. lines 19-22 Restore registers from stack

Main

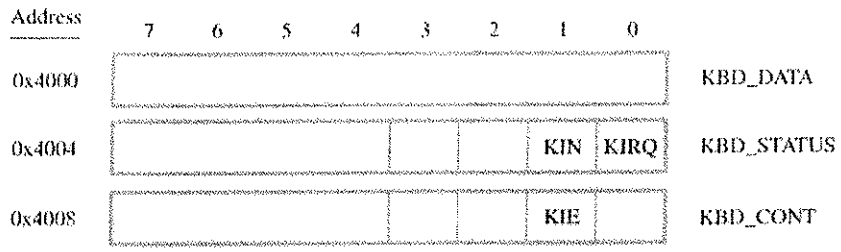
- A. lines 1-2 Initialize buffer pointer for incoming string
- B. lines 3-4 Clear EOL flag
- C. lines 5-6 Set enable keyboard interrupts
- D. lines 7-9 Set processor to recognize keyboard int's
- E. lines 10-12 Set interrupt enable bit

Interrupt-service routine

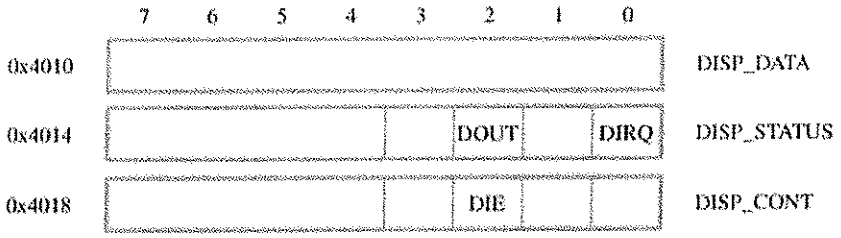
1	ILOC:	Subtract	SP, SP, #8
2		Store	R2, 4(SP)
3		Store	R3, (SP)
4		Load	R2, PNTR
5		LoadByte	R3, KBD_DATA
6		StoreByte	R3, (R2)
7		Add	R2, R2, #1
8		Store	R2, PNTR
9	ECHO:	LoadByte	R2, DISP_STATUS
10		And	R2, R2, #4
11		Branch_if_[R2]=0	ECHO
12		StoreByte	R3, DISP_DATA
13		Move	R2, #CR
14		Branch_if_[R3]≠[R2]	RTRN
15		Move	R2, #1
16		Store	R2, EOL
17		Clear	R2
18		StoreByte	R2, KBD_CONT
19	RTRN:	Load	R3, (SP)
20		Load	R2, 4(SP)
21		Add	SP, SP, #8
22		Return-from-interrupt	

Main program

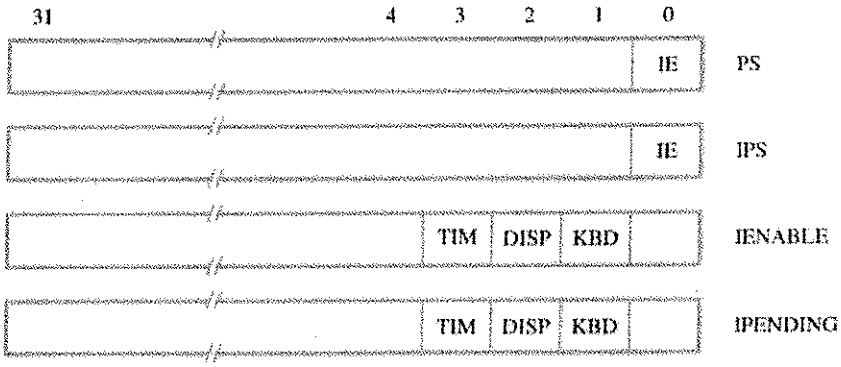
1	START:	Move	R2, #LINE
2		Store	R2, PNTR
3		Clear	R2
4		Store	R2, EOL
5		Move	R2, #2
6		StoreByte	R2, KBD_CONT
7		MoveControl	R2, IENABLE
8		Or	R2, R2, #2
9		MoveControl	IENABLE, R2
10		MoveControl	R2, PS
11		Or	R2, R2, #1
12		MoveControl	PS, R2
13		next instruction	



(a) Keyboard interface



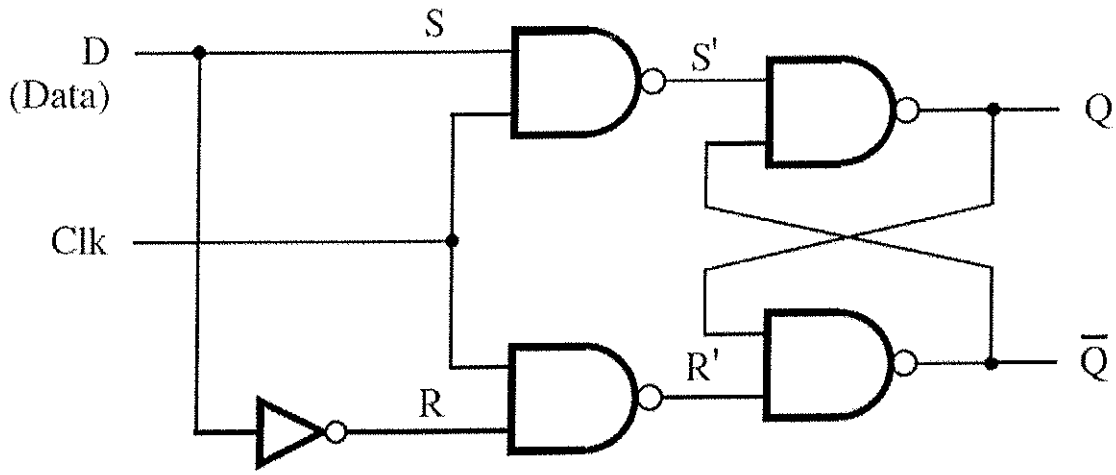
(b) Display interface

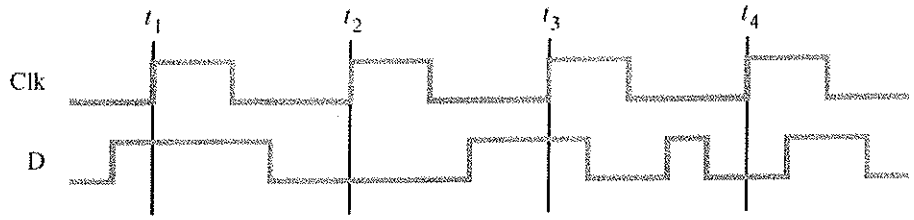


Problem 6 [10]

Topic: *D Flip-Flop*

Given the following positive edge triggered D Flip-Flop circuit fill in the truth table for the D Flip-Flop.





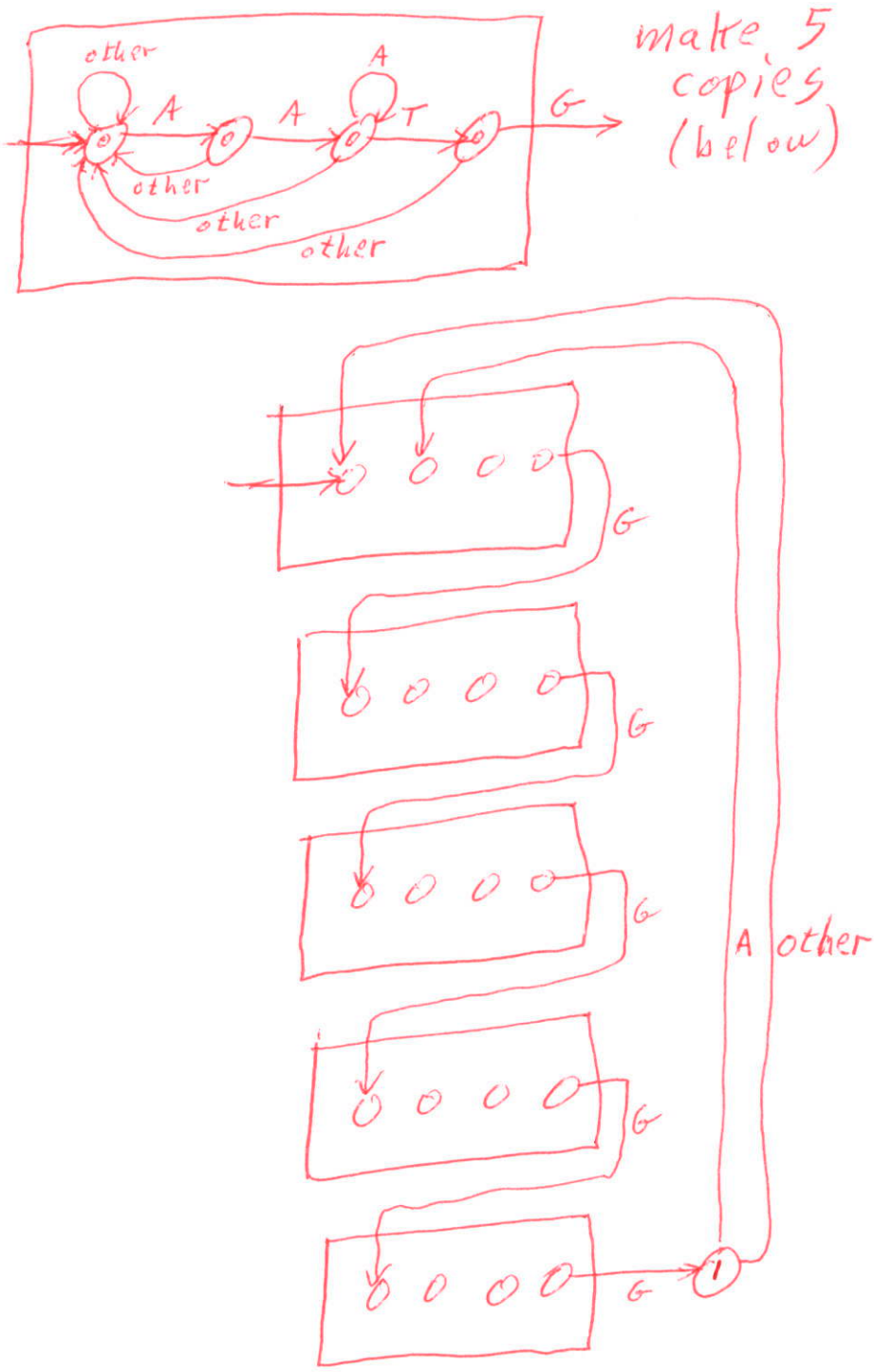
time	t_1	t_2
S	1	0
R	0	1
S'	0	1
R'	1	0
Q	1	0
$\neg Q$	0	1

Problem 7 [20]

Topic: *Finite State Machines*

Create a Finite State Machine that turns on an LED every fifth time the DNA sequence AATG is inputted. The sequences do not have to be contiguous. You do not need to construct a PLA or sequential circuits.

note: AATGCCAATG would result in 2 sequences being found.



Problem 8 [20]

Topic: *Digital Logic*

Build a adder that takes 2 3-bit binary strings using mux's, decoder, and, or or gates. You may make components to reuse so long as they are clearly labeled.
hint: Build a truth table