Department of Electrical and Computer Engineering
The University of Texas at Austin

EE 306, Fall 2013
Yale Patt, Instructor
Ben Lin, Mochamad Asri, Ameya Chaudhari, Nikhil Garg, Lauren Guckert,
Jack Koenig, Saijel Mokashi, Sruti Nuthalapathi, Sparsh Singhai, Jiajun Wang
Exam 1, October 9, 2013

Name:_____

Problem 1 (20 points):_____

Problem 2 (20 points):_____

Problem 3 (20 points):_____

Problem 4 (20 points):_____

Problem 5 (20 points):_____

Total (100 points):_____

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

I will not cheat on this exam.

_____
Signature

GOOD LUCK!

Name:_____

Problem 1. (20 points):
Part a. (4 points):

R0 contains the ASCII code of a capital letter in the English alphabet. If the instruction

0001000000000001

is executed, we wish to end up with the lower case version of that letter in R0. What must be true of the values in all the other registers before this instruction executes for this to happen?

Part b. (5 points):

Suppose we changed the LC-3 to have only four registers instead of 8. Fewer registers is in general a bad idea since it means loading from memory and storing to memory more often.

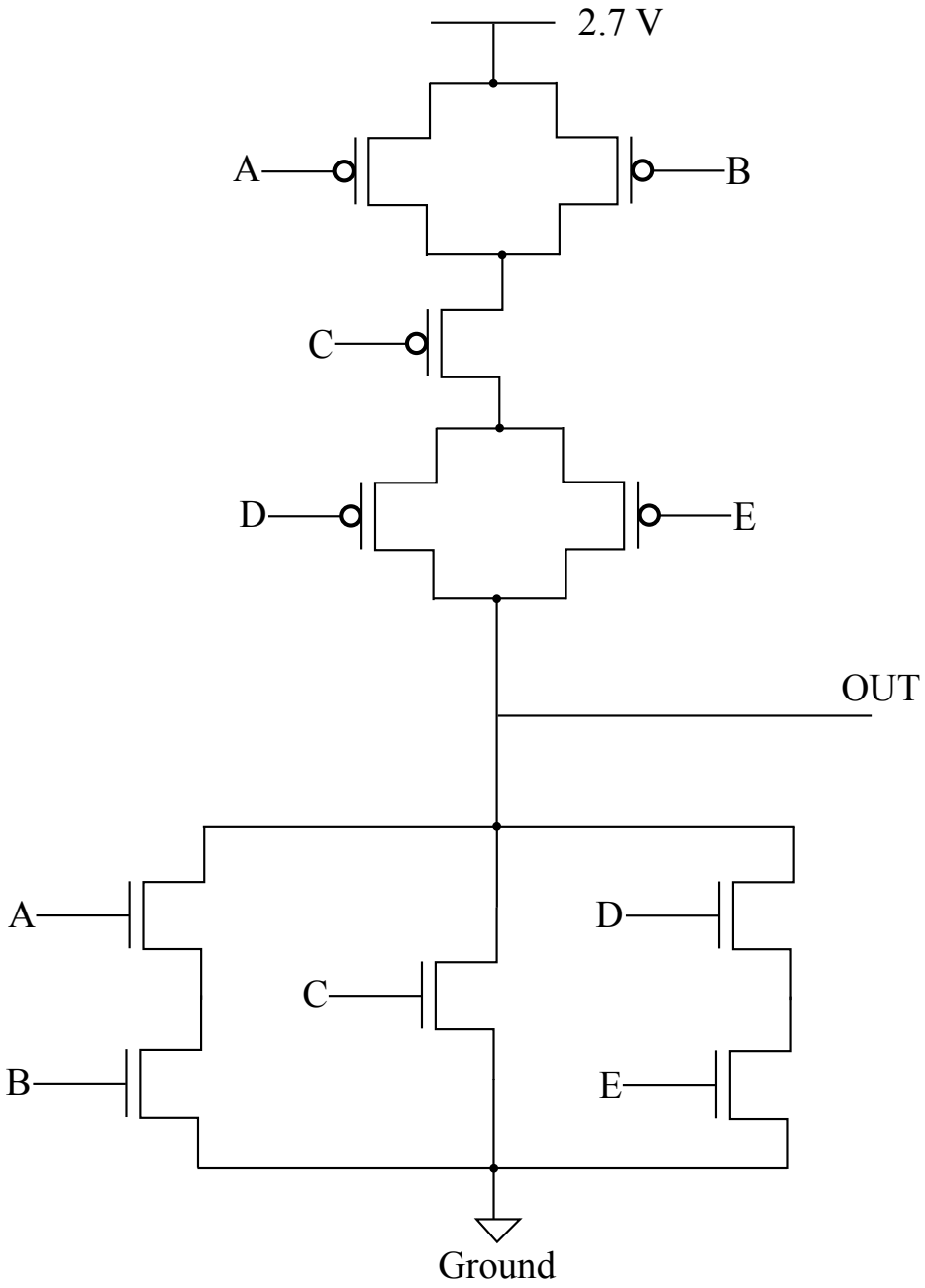
If we keep the basic format of all instructions as they currently are (and keep each instruction 16 bits), is there any benefit that could be had for operate (0001, 0101, 1001) instructions, if we reduce the number of registers to 4?

Is there any benefit that could be had for load (0010) and store (0011) instructions, if we reduce the number of registers to 4?

Is there any benefit that could be had for conditional branch (0000) instructions, if we reduce the number of registers to 4?

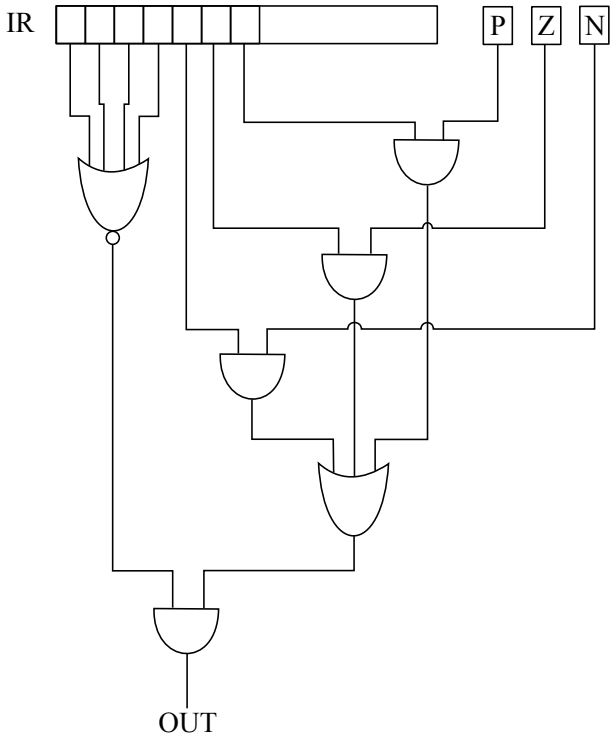
Part c. (4 points):

A properly working transistor circuit for implementing a logic function is one in which the output is always 1 or 0. The circuit below is an example of such that implements a 5 input logic function. Indicate on the truth table on the next page the input combinations for which the output is 1 by putting a 1 in the corresponding row of the output column. All other input combinations produce an output 0. It is not necessary to put the 0s in the output column.



A	B	C	D	E	OUT
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	
0	0	1	0	0	
0	0	1	0	1	
0	0	1	1	0	
0	0	1	1	1	
0	1	0	0	0	
0	1	0	0	1	
0	1	0	1	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	0	1	
0	1	1	1	0	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	0	1	1	
1	0	1	0	0	
1	0	1	0	1	
1	0	1	1	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	0	1	
1	1	0	1	0	
1	1	0	1	1	
1	1	1	0	0	
1	1	1	0	1	
1	1	1	1	0	
1	1	1	1	1	

Part d. (7 points): The logic diagram below produces the logical value OUT.

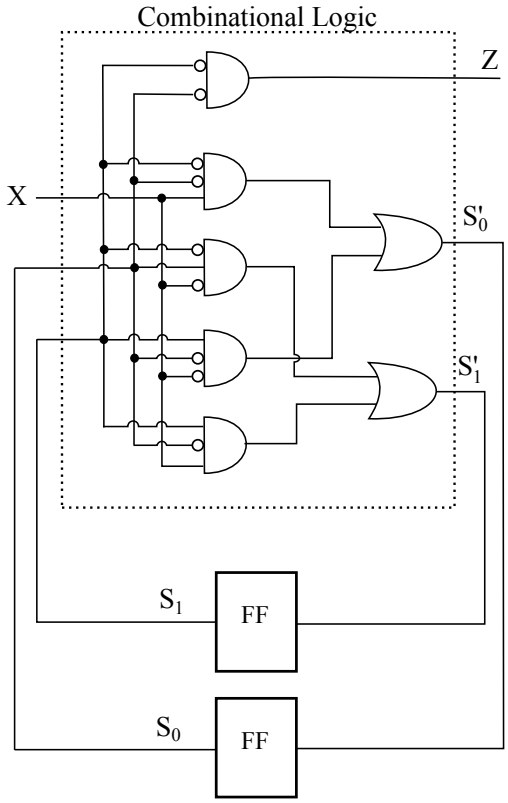


What do the values 0 or 1 for OUT signify?

0 signifies: | 1 signifies:

Name:_____

Problem 2. (20 points):
The logic diagram shown below is a finite state machine.

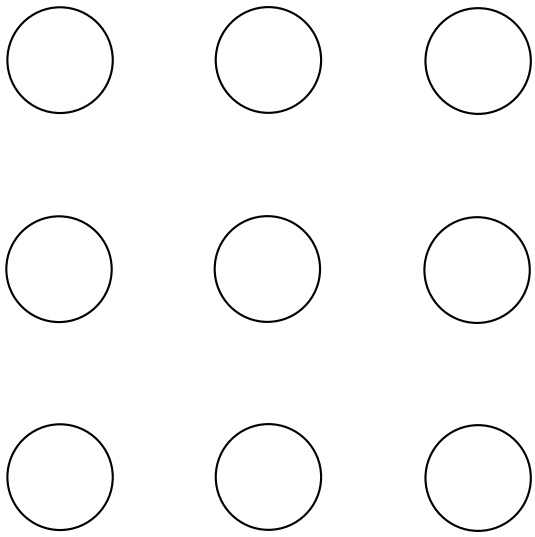


Your job:

a. Construct the truth table for the combinational logic:

S1	S0	X	Z	S1’	S0’
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

b. Complete the state machine (We have provided nine states. You will not need all of them. Use only as many as you need):



Name:_____

Problem 3. (20 points): Floating point

Express the two floating point numbers $2\frac{13}{16}$ and $\frac{1}{16}$ in binary normalized form:

$2\frac{13}{16}$ $\frac{1}{16}$

We wish to design a nine-bit binary floating point data type, using the format of the IEEE standard. We want to be able to represent both of these two values **exactly** in **normalized**(i.e. not subnormal) form.

Your job:

a. Decide how many bits for exponent, how many bits for fraction.

No. of exponent bits: No. of fraction bits:

b. Decide what the BIAS (also known as EXCESS) will have to be in order to be able to represent both values in normalized (i.e. not subnormal) form.

Hint: the BIAS $011\ldots1$ will not work.

BIAS:

c. Show the binary representations of the two values $2\frac{13}{16}$ and $\frac{1}{16}$ in our new nine-bit binary floating point data type

$2\frac{13}{16}$

$\frac{1}{16}$

Name:_____

Problem 4. (20 points):
We wish to invent a two-person game, which we will call XandY that can be played on the computer. Your job in this problem is contribute a piece of the solution.

The game is played with the computer and a deck of cards. Each card has on it one of four values (X, Y, Z, and N). Each player in turn gets five attempts to accumulate points. We call each attempt a round. After player A finishes his five rounds, it is player B's turn. Play continues until one of the players accumulates 100 points.
Your job today is to ONLY design a finite state machine to keep track of the STATE of the **current** round. Each round starts in the intial state, where X=0 and Y=0. Cards from the deck are turned over one by one. Each card transitions the round from its current state to its next state, until the round terminates, at which point we'll start a new round in the initial state.

The transistions are as follows:

X: The number of X's is incremented, producing a new state for the round.

Y: The number of Y's is incremented, producing a new state for the round.

Z: If the number of X's is less than 2, the number of X's is incremented, producing a new state for the round. If the number of X's is 2, the state of the current round does not change.

N: Other information on the card gives the number of points accumulated. N also terminates the current round.

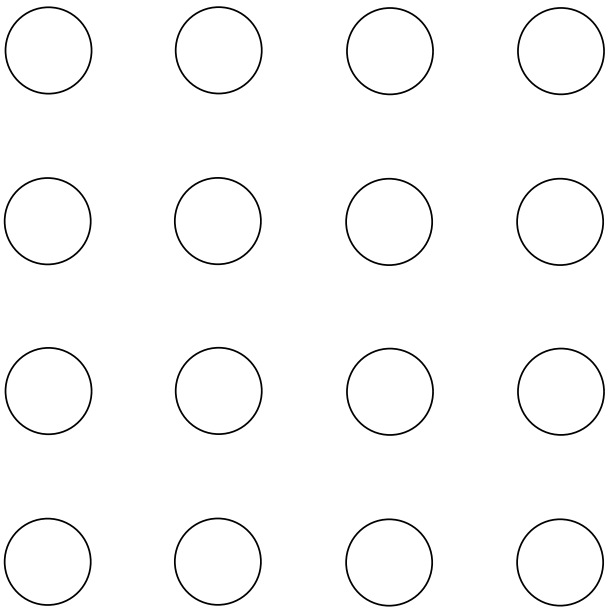
Important rule: If the number of X's or Y's reaches a count of 3, the current round is terminated and another round is started. When a round starts, its state is X=0, Y=0.

Hint: Since the number of X's and Y's specify the state of the current round, how many possible states are needed to describe the state of the current round.

Hint: A state can not have X=3, because then the round would be finished, and we would have started a *new* current round.

On the diagram below, label each state. For each state draw an arrow showing the transition to the next state that would occur for each of the four inputs. (We have provided sixteen states. You will not need all of them. Use only as many as you need)

Note, we did not specify outputs for these states. Therefore, your state machine will not include outputs. It will only include states and transistions represented by inputs.



Name:_____

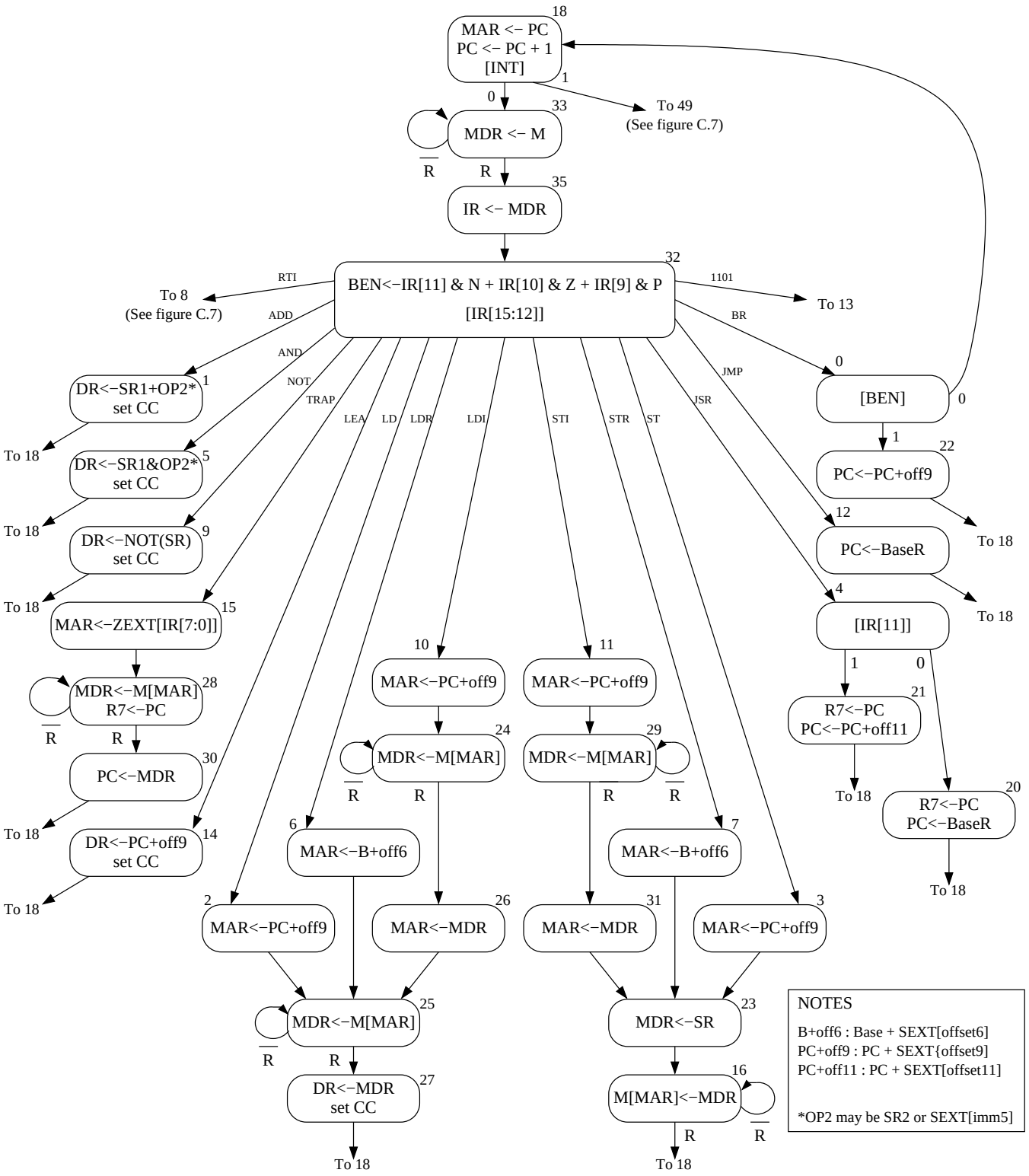
Problem 5. (20 points):

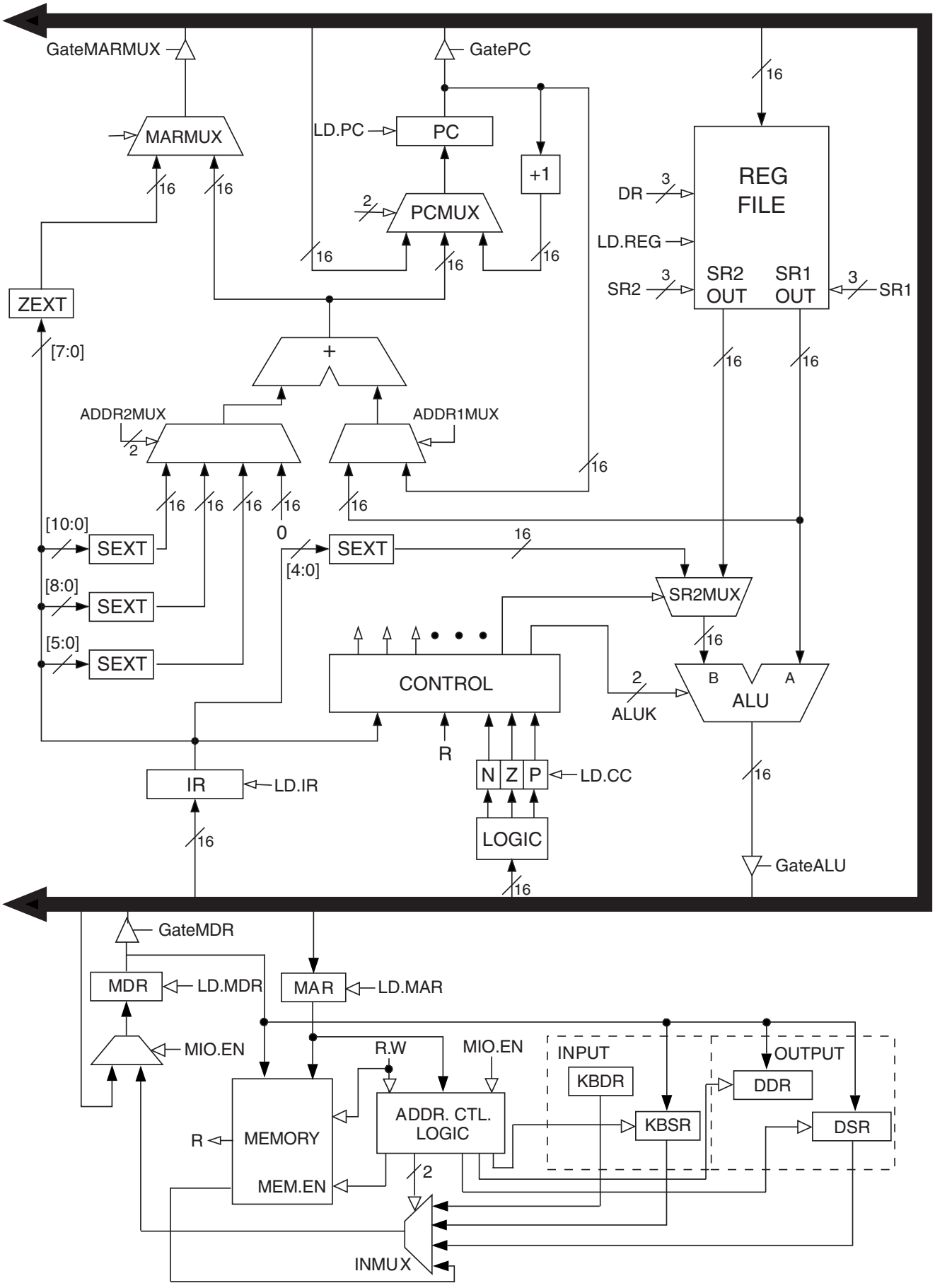
You have been asked to design the volume control system in a stereo. The user controls the volume by using volume up and volume down buttons on the stereo. When the user presses the volume up button, the volume should increase by 1; when the user presses the volume down button, the volume should decrease by 1. The volume level is represented as a 4-bit unsigned value, ranging from 0 to 15. If the user presses volume up when the volume is already at the maximum level of 15, the volume should remain at 15; similarly, if the user presses volume down when the volume is already at the minimum level of 0, the volume should remain at 0. The memory location x3100 has been directly hooked up to the speakers so that reading bits 3 through 0 from that memory location will give the current speaker volume, while writing bits 3 through 0 of that memory location will set the new speaker volume.

When the user presses one of the volume buttons, the stereo hardware will reset the PC of the processor to x3000 and begin execution. If the user had pressed volume up, then memory location x3101 will be set to 1; otherwise, if the user had pressed volume down, then the memory location x3101 will be set to 0.

Below is the program that controls the volume on the stereo. Two of the instructions in the program have been left out. Your job: fill in the missing instructions so that the program controls the volume correctly as specified.

Address	Contents	Description
x3000	0010000011111111	R0 ← M[x3100]
x3001	0010001011111111	R1 ← M[x3101]
x3002	0000010000000100	Branch to x3007 if Z is set
x3003		
x3004	0000010000000101	Branch to x300A if Z is set
x3005	0001000000100001	R0 ← R0 + x0001
x3006	0000111000000011	Branch always to x300A
x3007	0001001000100000	R1 ← R0 + x0000
x3008	0000010000000001	Branch to x300A if Z is set
x3009		
x300A	0011000011110101	M[x3100] ← R0
x300B	1111000000100101	TRAP x25


18
MAR <− PC
PC <− PC + 1
[INT]
1
To 49
(See figure C.7)
0
33
MDR <− M
R
R
35
IR <− MDR
32
BEN<−IR[11] & N + IR[10] & Z + IR[9] & P
[IR[15:12]]
RTI
To 8
(See figure C.7)
1101
To 13
ADD
AND
NOT
TRAP
LEA
LD
LDR
LDI
STI
STR
ST
JSR
JMP
BR
0
[BEN]
0
1
22
PC<−PC+off9
To 18
12
PC<−BaseR
To 18
4
[IR[11]]
1
0
21
R7<−PC
PC<−PC+off11
To 18
20
R7<−PC
PC<−BaseR
To 18
1
DR<−SR1+OP2*
set CC
To 18
5
DR<−SR1&OP2*
set CC
To 18
9
DR<−NOT(SR)
set CC
To 18
15
MAR<−ZEXT[IR[7:0]]
28
MDR<−M[MAR]
R7<−PC
R
R
30
PC<−MDR
To 18
14
DR<−PC+off9
set CC
To 18
10
MAR<−PC+off9
24
MDR<−M[MAR]
R
R
6
MAR<−B+off6
2
MAR<−PC+off9
26
MAR<−MDR
25
MDR<−M[MAR]
R
R
27
DR<−MDR
set CC
To 18
11
MAR<−PC+off9
29
MDR<−M[MAR]
R
R
7
MAR<−B+off6
31
MAR<−MDR
3
MAR<−PC+off9
23
MDR<−SR
16
M[MAR]<−MDR
R
R
To 18
NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT{offset9]
PC+off11 : PC + SEXT[offset11]
*OP2 may be SR2 or SEXT[imm5]


GateMARMUX
GatePC
16
MARMUX
LD.PC
PC
+1
REG
FILE
DR
3
16
16
2
PCMUX
LD.REG
16
16
16
SR2
OUT
SR1
OUT
3
SR2
3
SR1
ZEXT
[7:0]
+
16
16
ADDR2MUX
2
ADDR1MUX
16
16
16
16
16
16
[10:0]
SEXT
0
SEXT
16
[4:0]
[8:0]
SEXT
SR2MUX
16
[5:0]
SEXT
CONTROL
2
B
A
ALU
ALUK
R
N
Z
P
LD.CC
IR
LD.IR
16
LOGIC
16
16
GateALU
GateMDR
MDR
LD.MDR
MAR
LD.MAR
MIO.EN
R.W
MIO.EN
INPUT
KBDR
OUTPUT
DDR
ADDR. CTL.
LOGIC
KBSR
DSR
R
MEMORY
MEM.EN
2
INMUX

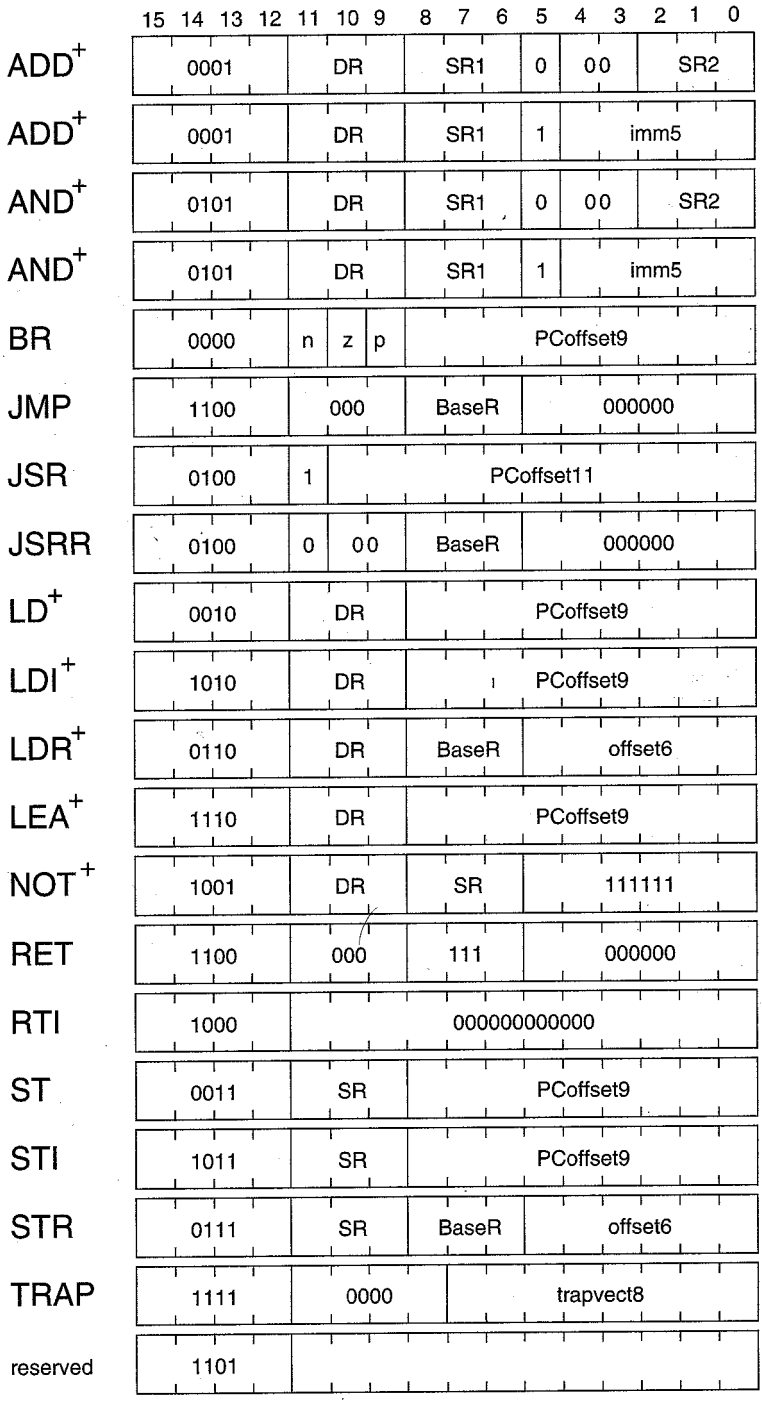


Figure A.2 Format of the entire LC-3 instruction set. **Note:** + indicates instructions that modify condition codes

The Standard ASCII Table

ASCII			ASCII			ASCII			ASCII		
Character	Dec	Hex	Character	Dec	Hex	Character	Dec	Hex	Character	Dec	Hex
nul	0	00	sp	32	20	@	64	40	`	96	60
soh	1	01	!	33	21	A	65	41	a	97	61
stx	2	02	"	34	22	B	66	42	b	98	62
etx	3	03	#	35	23	C	67	43	c	99	63
eot	4	04	$	36	24	D	68	44	d	100	64
enq	5	05	%	37	25	E	69	45	e	101	65
ack	6	06	&	38	26	F	70	46	f	102	66
bel	7	07	'	39	27	G	71	47	g	103	67
bs	8	08	(	40	28	H	72	48	h	104	68
ht	9	09	)	41	29	I	73	49	i	105	69
lf	10	0A	*	42	2A	J	74	4A	j	106	6A
vt	11	0B	+	43	2B	K	75	4B	k	107	6B
ff	12	0C	,	44	2C	L	76	4C	l	108	6C
cr	13	0D	-	45	2D	M	77	4D	m	109	6D
so	14	0E	.	46	2E	N	78	4E	n	110	6E
si	15	0F	/	47	2F	O	79	4F	o	111	6F
dle	16	10	0	48	30	P	80	50	p	112	70
dc1	17	11	1	49	31	Q	81	51	q	113	71
dc2	18	12	2	50	32	R	82	52	r	114	72
dc3	19	13	3	51	33	S	83	53	s	115	73
dc4	20	14	4	52	34	T	84	54	t	116	74
nak	21	15	5	53	35	U	85	55	u	117	75
syn	22	16	6	54	36	V	86	56	v	118	76
etb	23	17	7	55	37	W	87	57	w	119	77
can	24	18	8	56	38	X	88	58	x	120	78
em	25	19	9	57	39	Y	89	59	y	121	79
sub	26	1A	:	58	3A	Z	90	5A	z	122	7A
esc	27	1B	;	59	3B	[	91	5B	{	123	7B
fs	28	1C	<	60	3C	\	92	5C	|	124	7C
gs	29	1D	=	61	3D	]	93	5D	}	125	7D
rs	30	1E	>	62	3E	^	94	5E	~	126	7E
us	31	1F	?	63	3F	_	95	5F	del	127	7F

Table A.2 Trap Service Routines

Trap Vector	Assembler Name	Description
x20	GETC	Read a single character from the keyboard. The character is not echoed onto the console. Its ASCII code is copied into R0. The high eight bits of R0 are cleared.
x21	OUT	Write a character in R0[7:0] to the console display.
x22	PUTS	Write a string of ASCII characters to the console display. The characters are contained in consecutive memory locations, one character per memory location, starting with the address specified in R0. Writing terminates with the occurrence of x0000 in a memory location.
x23	IN	Print a prompt on the screen and read a single character from the keyboard. The character is echoed onto the console monitor, and its ASCII code is copied into R0. The high eight bits of R0 are cleared.
x24	PUTSP	Write a string of ASCII characters to the console. The characters are contained in consecutive memory locations, two characters per memory location, starting with the address specified in R0. The ASCII code contained in bits [7:0] of a memory location is written to the console first. Then the ASCII code contained in bits [15:8] of that memory location is written to the console. (A character string consisting of an odd number of characters to be written will have x00 in bits [15:8] of the memory location containing the last character to be written.) Writing terminates with the occurrence of x0000 in a memory location.
x25	HALT	Halt execution and print a message on the console.

Table A.3 Device Register Assignments

Address	I/O Register Name	I/O Register Function
xFE00	Keyboard status register	Also known as KBSR. The ready bit (bit [15]) indicates if the keyboard has received a new character.
xFE02	Keyboard data register	Also known as KBDR. Bits [7:0] contain the last character typed on the keyboard.
xFE04	Display status register	Also known as DSR. The ready bit (bit [15]) indicates if the display device is ready to receive another character to print on the screen.
xFE06	Display data register	Also known as DDR. A character written in the low byte of this register will be displayed on the screen.
xFFFE	Machine control register	Also known as MCR. Bit [15] is the clock enable bit. When cleared, instruction processing stops.