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

EE 306, Fall 2011
Yale Patt, Instructor
Faruk Guvenilir, Milad Hashemi, Jennifer Davis, Garrett Galow,
Ben Lin, Taylor Morrow, Stephen Pruett, and Jee Ho Ryoo, TAs
Exam 2, November 2, 2011

Name: Solution

Problem 1 (15 points): 15

Problem 2 (10 points): 10

Problem 3 (20 points): 20

Problem 4 (15 points): 15

Problem 5 (15 points): 15

Problem 6 (25 points): 25

Total (100 points): 100

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.

Solution
Signature

GOOD LUCK!

Name: Solution

Problem 1. (15 points):

Part a. (5 points): An LC-3 computer, during execution of a program, encounters during a period of 19 clock cylces the following states of the state machine in sequence:

18, 33, 35, 32, (2), 25, 27, 18, 33, 35, 32, (6), 25, 27, 18, 33, 35, 32, (1)

List the opcodes of the instructions executed in the order executed. Use as many entries as you need:

LD
LDR
ADD
/
/

Part b. (5 points): Construct the symbol table for the block of code on the left:

```
          .ORIG x4000
        0 ADD   R1,R2,R3
DIVIDE  1 AND   R1,R2,R3
        2 JMP   HERE
HELP1     .BLKW 4 3,4,5,6
HELP2     .STRINGZ "HELP" 7,8,9,A,B
HERE    C .FILL xF025
          .END
```

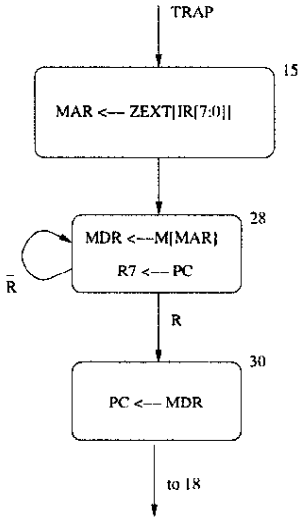
Symbol Table:

Symbol	Address
DIVIDE	x4001
HELP1	x4003
HELP2	x4007
HERE	x400C

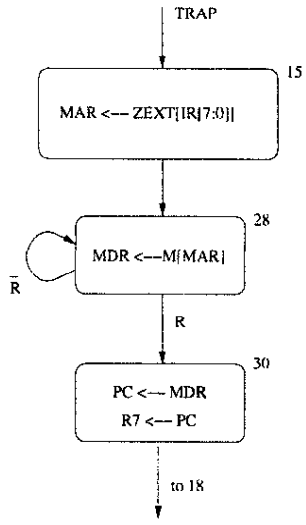
Name: Solution

Part c. (5 points): The LC-3 state machine shows that after Decode (state 32), the TRAP instruction requires three clock cycles to finish execution, represented as states 15, 28, and 30. These three states are reproduced below (on the left). Note that R7 gets loaded with PC many times, depending on how many cycles it takes to read memory. A suggestion: Move R7 <– PC from state 28 to state 30 so it will get done once (as shown on the right).

Before



After



Will the suggested new implemenation work on our existing data path? EXPLAIN in 15 words or fewer.

No, Both MDR to PC and PC to R7 would require using the Bus. Only 1 value can be gated on the Bus at a time (during a single cycle).

Name: Solution

Problem 2. (10 points): We wish to add a new TRAP service routine, which will be called by the instruction `TRAP x9A`. The new trap routine will wait for someone to type a lower case letter, then echo on the screen the corresponding capital letter. Assume the user will not type anything except a lower case letter. The assembly language code for this trap service routine is shown below:

```
        .ORIG  x2055
```

[ST R7, B]

```
        ST R1, SaveR1
        ST R0, SaveR0
        TRAP x20
        LD R1, A
```

[ADD R0, R0, R1]

```
        TRAP x21
```

[LD R7, B]

```
        LD R1, SaveR1
        LD R0, SaveR0
        JMP R7
SaveR1  .BLKW 1
SaveR0  .BLKW 1
A       .FILL [ #-32 ]
[ B ]   .BLKW 1

        .END
```

Part a: In order for `TRAP x9A` to call this service routine, what memory location must contain what value.

Address: [x009A] Value: [x2055]

Part b: Fill in the missing information in the assembly language program. i.e, the three missing instructions, the one missing label, and the operand of the `.FILL` pseudo-op.

Name: Solution

Problem 3. (20 points): The following LC-3 assembly language program:

```
            .ORIG x3000
            AND R2, R2, #0
            AND R6, R6, #0
            ADD R2, R2, #1
TOP         ADD R3, R2, #0
            ADD R4, R1, #0
SEARCH      ADD R3, R3, R3
            ADD R4, R4, #-1
            BRp SEARCH
            AND R5, R3, R0
            BRz NEXT
            ADD R6, R6, R2
NEXT        ADD R2, R2, R2
            BRzp TOP
END         ST R6, RESULT
            HALT
RESULT      .BLKW 1
            .END
```

Must be inputs

What does it do (in twenty words or fewer)? Please be BRIEF but PRECISE.
You can assume that some of the registers will already contain numbers that are relevant to the program.

Shifts the number in R0 right by the number of bit positions specified in R1, and stores the result in RESULT.

What is the function of R0? For what range of input values does the program function as you've described above?

The number to shift. Any Input Values

What is the function of R1? For what range of input values does the program function as you've described above?

How much to Shift By. Positive Input Values

What is the function of R6? For what range of input values does the program function as you've described above?

Keep track of Result/Output. Any Input Values.

Name: Solution

Problem 4. (15 points): The following LC-3 assembly language program executes to completion.

```
            .ORIG   x3000
            AND     R0, R0, #0
            ADD     R1, R0, #1
            ADD     R3, R0, #10
            LEA     R6, RESULTS

LOOP        ADD     R2, R0, R1
            ADD     R0, R1, #0
            ADD     R1, R2, #0

STORE       STR     R1, R6, #0

            LD      R2, STORE
            ADD     R2, R2, #1
            ST      R2, STORE

            ADD     R3, R3, #-1
            BRp     LOOP

            HALT

RESULTS     .BLKW   10
            .END
```

Part a. (9 points): After the program has halted, what values are contained (in decimal) in the ten consecutive memory locations starting at the memory location labeled RESULTS?

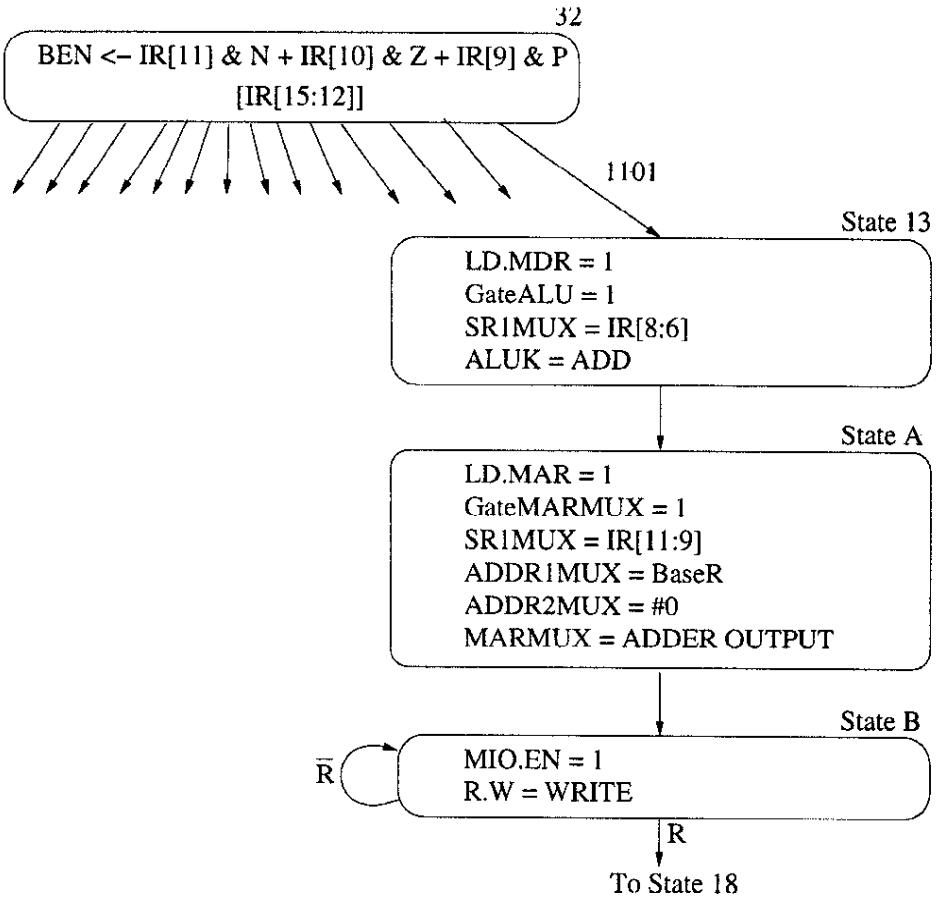
1
2
3
5
8
13
21
34
55
89

Part b. (6 points): The three instructions inside the box above can be replaced by a single instruction while preserving the functionality of the program. What is that single instruction?

ADD R6, R6, #1

Name: Solution

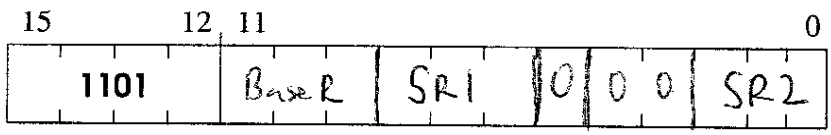
Problem 5. (15 points): Let's use the unused opcode 1101 to specify a new instruction. We will require 3 states after decode (state 32) to complete the job. The control signals required to carry out the work of the new instruction are shown below. All control signals not shown below are 0.



In 20 words or fewer, what does the new instruction do?

Adds the contents of SR1(IR[8:6]) to SR2 (or immediate) and stores result in location specified by BaseR (IR[11:9]).

The instruction may have one or two potential formats. Fill out one or both as you deem appropriate.



* Full Credit for only specifying first format (IR[5]=0).

Name: Solution

Problem 6. (25 points): An LC-3 program is executing on the LC-3 Simulator when a breakpoint is encountered, and the Simulator stops. At that point, **the contents of several registers are as shown in the first row of the table.** After the run button is subsequently pushed, the next four instructions that are executed, none of which are an STI or LDI, produce the values shown in the table, **two rows of the table per instruction executed.** The first row of each pair shows the contents after the fetch phase of the corresponding instruction, and the second row of each pair after that instruction completes.

Note that some values are missing, and are presented by letters A, B, C, D, E, F, G, H, I, and J.

PC	MAR	MDR	IR	R0	R1	R2	R3	R4	R5	R6	R7
x1800	x7FFF	x2211	xBFFE	x31FF	x2233	x5177	x3211	x21FF	x5233	x3177	x2211
A	x1800	B	B	x31FF	x2233	x5177	x3211	x21FF	x5233	x3177	x2211
A	x1800	B	B	x31FF	x2233	x5177	x3211	x21FF	C	x3177	x2211
D	A	E	E	x31FF	x2233	x5177	x3211	x21FF	C	x3177	x2211
D	F	G	E	x31FF	x2233	x5177	x3211	x21FF	C	x3177	x2211
H	D	I	I	x31FF	x2233	x5177	x3211	x21FF	C	x3177	x2211
F	D	I	I	x31FF	x2233	x5177	x3211	x21FF	C	x3177	x2211
A	F	J	J	x31FF	x2233	x5177	x3211	x21FF	C	x3177	x2211
A	F	J	J	x31FF	x2233	x5177	x3211	x223A	C	x3177	x2211

Your job: Determine the values of A, B, C, D, E, F, G, H, I, and J. Note that some of the values may be identical.

A	B	C	D	E
x1801	xEA66	x1867	x1802	x3BFE

F	G	H	I	J
x1800	x1867	x1803	xOFED	x1867

J: ADD R4, R1, #7

0001 | 100 | 001 | 0 0 1 1

x 1 8 6 7

E: STR R5, #-2

0011 | 101 | 1 1 1 1 1 1 1 0

x 3 B F E

B: LEA R5, x66

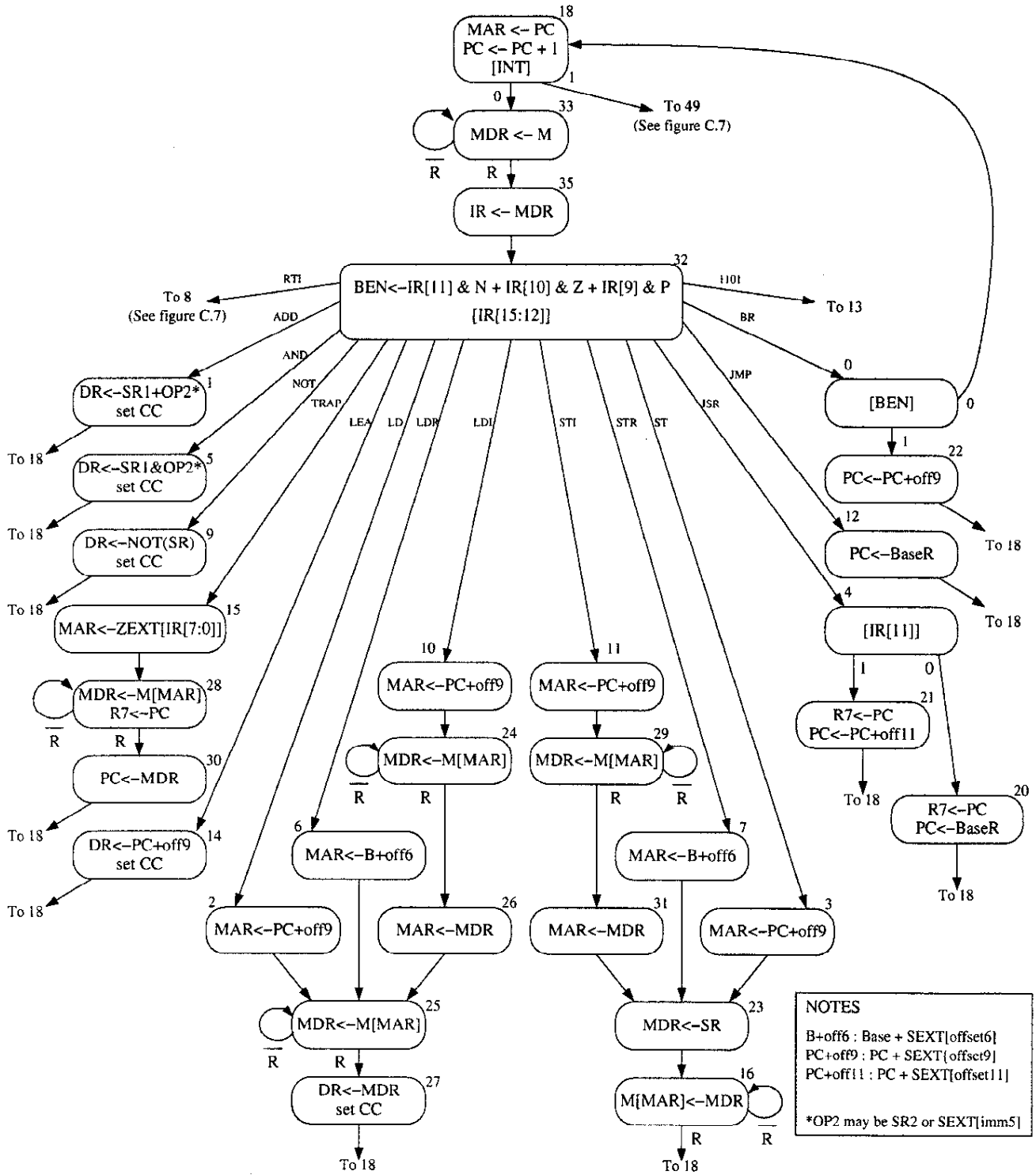
1110 | 101 | 0 0 1 1 0 0 1 1 0

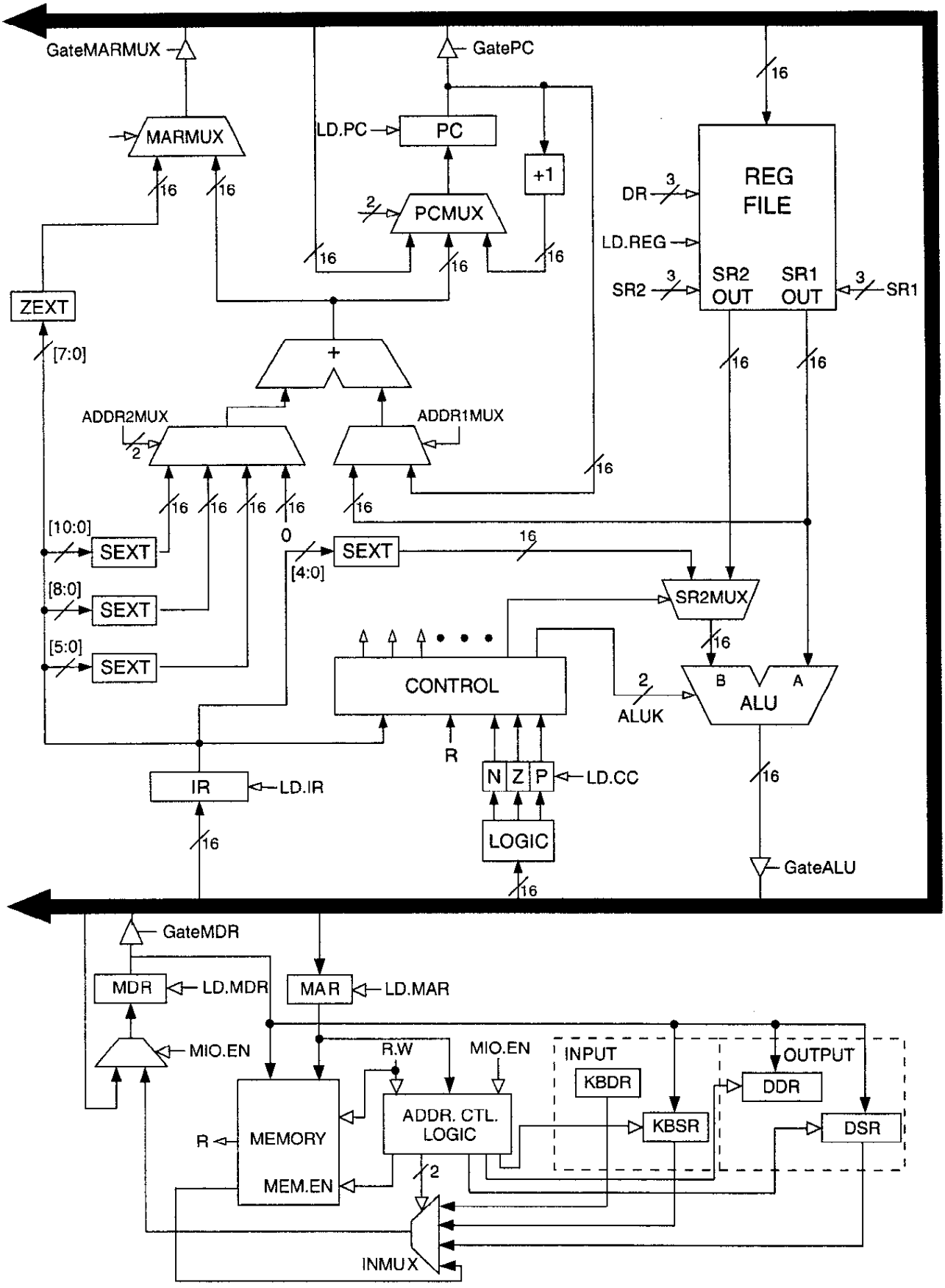
x E A 6 6

I: BRnzp #-3

0000 | 1 1 1 | 1 1 1 1 1 1 0 1

x 0 F F D


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

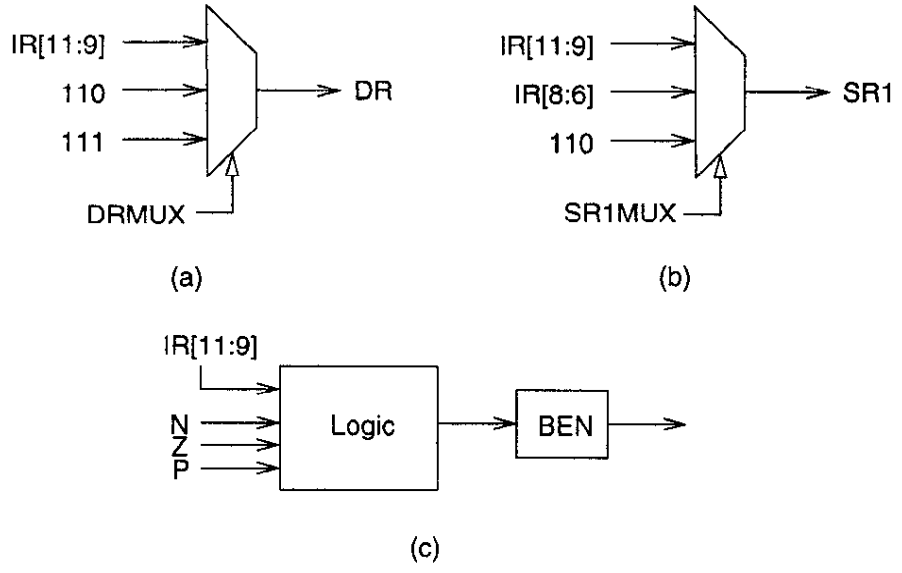

IR[11:9]
110
111
DR
DRMUX
(a)
IR[11:9]
IR[8:6]
110
SR1
SR1MUX
(b)
IR[11:9]
N
Z
P
Logic
BEN
(c)

Table A.2 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 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.

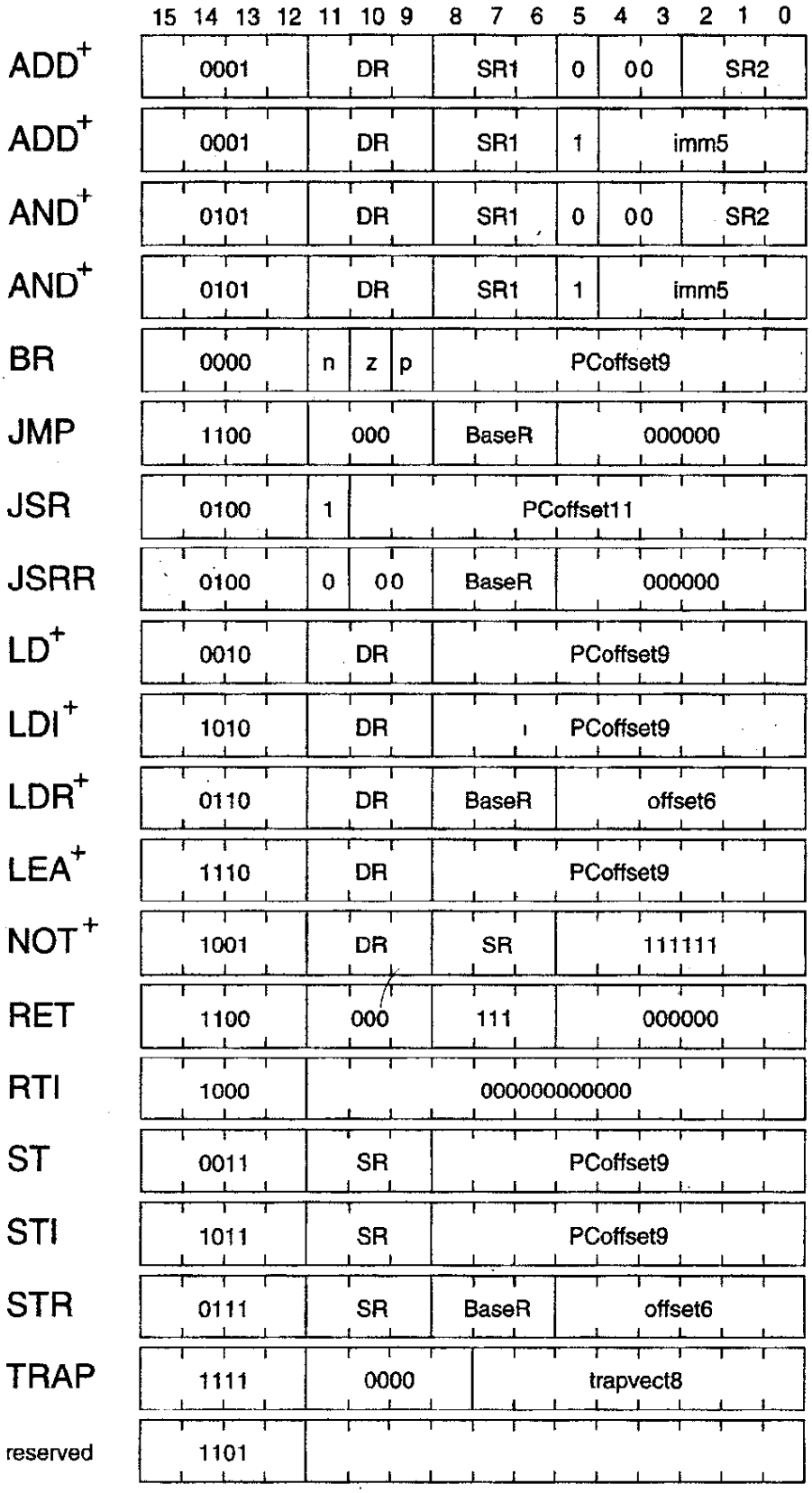


Figure A.2 Format of the entire LC-3 instruction set. **Note:** + indicates instructions that modify condition codes

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