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

EE 306, Fall, 2008
Yale Patt, Instructor
TAs: Jeffrey Allan, Arvind Chandrababu, Eiman Ebrahimi, Aravind Jakkani, Khubaib, Allison Korczynski, Pratyusha Nidamaluri, Zrinka Puljiz, Che-Chun Su, Christopher Wiley.
Exam 1, October 08, 2008

Name: SOLUTION SHEET

Problem 1 (20 points):________

Problem 2 (20 points):________

Problem 3 (15 points):________

Problem 4 (15 points):________

Problem 5 (15 points):________

Problem 6 (15 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 written legibly on each sheet of the exam.

I will not cheat on this exam.

____
Signature

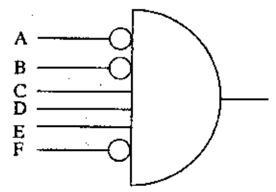
GOOD LUCK!

Name:________________________________

Problem 1 (20 points)

Part a (4 points): For what values of A,B,C,D,E, and F will the output of the 6-input AND gate be 1.

A = B = F = ∅ , C = D = E = 1



Part b (4 points): Add the two numbers. They are in base 7. Your result should also be in base 7.

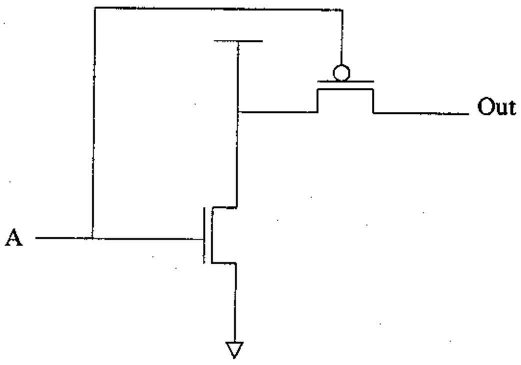
$$\begin{array}{r} 56042 \\ 03561 \\ \hline \boxed{62633} \end{array}$$

Part c (4 points): The value -5 can be represented by strings of 0s and 1s according to the following data types. Please show them below.

1 1 1 1 1 0 1 1	8 bit 2's complement integer
0010 1101 0011 0101	in ASCII
1 10000001 01000000000····0	32 bit floating point

Name:____

Part d (4 points): An Aggie knew that an inverter contained one P-type transistor and one N-type transistor, but he wired them up wrong, as shown below.



What is the value of Out when A=0: 1

What is the value of Out when A=1: Out floats

Part e (4 points): When a computer executes an instruction, the state of the computer is changed as a result of that execution. Is there any difference in the state of the LC-3 computer as a result of executing instruction 1 below vs executing instruction 2 below? Explain. We can assume the state of the LC-3 computer before execution is the same in both cases.

```
instruction 1: 0001 000 000 1 00000   register 0 <-- register 0 + #0

instruction 2: 0000 111 000000000     branch to PC' + #0 if any of N,Z,or P is set
```

Put your answer (no more than 20 words) in the box below.

INSTRUCTION 1 SETS CONDITION CODES
INSTRUCTION 2 DOES NOT

Name:____

Problem 2 (20 points)

Part a (7 points): A program wishes to load a value from memory into register 1, and on the basis of the value loaded, execute code starting at x3040 if the value loaded is positive, execute code starting at x3080 if the value loaded is negative, or execute code starting at location x3003 if the value loaded is zero. The first instruction of this program (load a value into register R1) is shown in x3000.

Your job: write the instructions for locations x3001 and x3002.

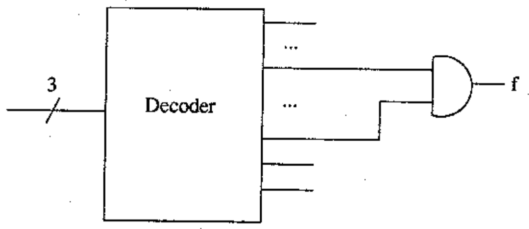
x3000:	0010 001 011111111
x3001:	0000 001 000111110
x3002:	0000 100 001111101

Part b (7 points): The program segment below starts execution at x3000. When the program halts, what is contained in register 0?

```
x3000: 0101 000 000 1 00000    ;register 0 <-- 0
x3001: 0001 000 000 1 00001    ;register 0 <-- register 0 + 1
x3002: 0000 001 111111110      ;branch p -2
x3003: 1111 0000 0010 0101     ;TRAP x25
```

R0: 1000000000000000

Part c (6 points): Two of the outputs of a 3 to 8 decoder are used as inputs to an AND gate as shown below.



Do you have enough information to say for certain what the output of f is?
If yes, give the value of f and explain why you have enough information. If no, explain what further information you need.

SINCE THE EIGHT OUTPUTS OF THE DECODER CAN ONLY HAVE AT ONE TIME EXACTLY ONE OF THEM = 1, ONE OF TWO INPUTS TO THE AND GATE MUST BE Ø. ∴ f=Ø

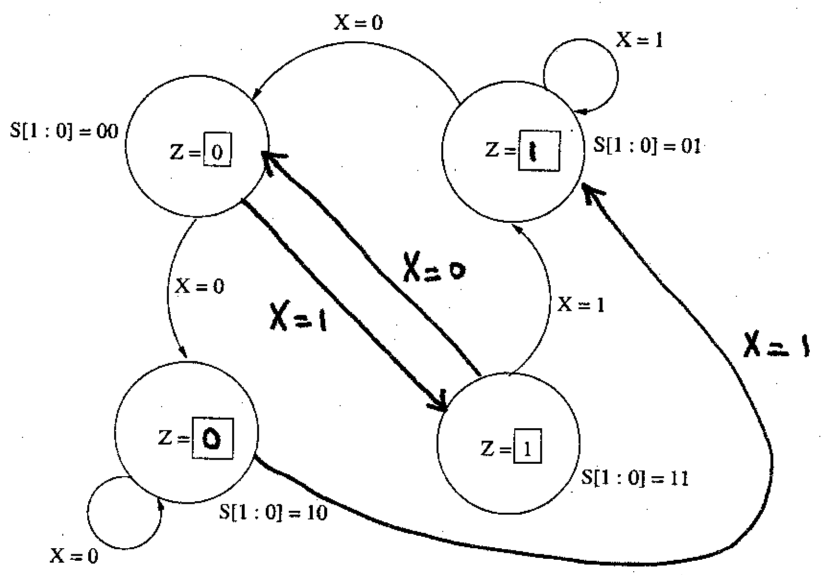
Name:____

Problem 3 (15 points)

Shown below is a state diagram for a 4 state machine, and the truth table showing the behavior of this state machine. Some of the entries in both are missing.
Note that the states are labeled 00, 01, 10, and 11 and the output of each state Z (0 or 1) is shown in each state. The input is shown as X.

Your job, complete both the truth table and the state machine.

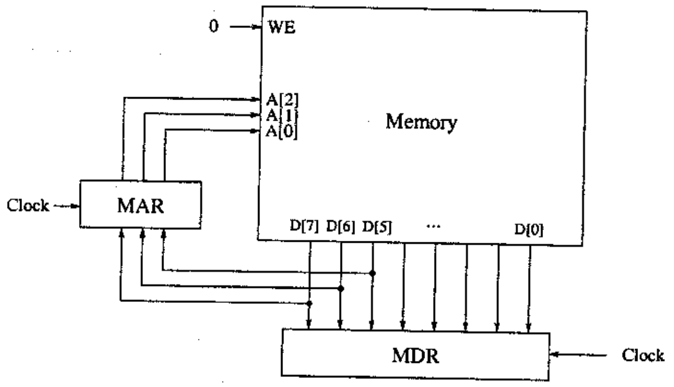
S[1]	S[0]	X	S'[1]	S'[0]	Z
0	0	0	1	0	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	0	1	1



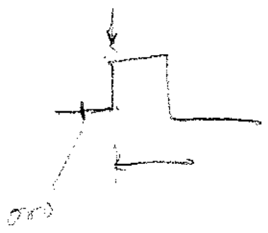
Name:____________________________________

Problem 4 (15 points)

Shown below is a byte-addressible memory consisting of 8 locations, and its associated MAR and MDR. Both MAR and MDR consist of flip flops which are latched at the start of each clock cycle based on the values on their corresponding input lines. A memory read is initiated every cycle and the data is available by the end of that cycle.



Just before the start of cycle 1, MAR contains 000, MDR contains 00010101, and the contents of each memory location is as shown.



Memory Location	Value
x0	01010000
x1	11110001
x2	10000011
x3	00010101
x4	11000110
x5	10101011
x6	00111001
x7	01100010

(a) What do MAR and MDR contain just before the end of cycle 1.

MAR: 010 MDR: 01010000

(b) What does MDR contain just before the end of cycle 4.

MDR: 00111001

Please show your work.

	MAR	MDR
Before end of cycle 1 :	010	01010000
Before end of cycle 2 :	100	10000011
Before end of cycle 3 :	110	11000110
Before end of cycle 4 :	001	00111001

Name:________________________________

Problem 5 (15 points)

An operate instruction present in most ISAs is MOD (example: MOD C,A,B). The result is the remainder one gets when the dividend A is divided by the divisor B. It is sometimes written: A mod B. That is, 10 mod 3 = 1. 20 mod 6 = 2.

Since the LC-3 does not have a MOD instruction, we wish to write a program to produce the result: A mod B. It will require dividing A by B. Since the LC-3 does not have a Divide instruction either, we will extend the idea we used in class to multiply two numbers together by successive additions. That is, we can perform Division by successively subtracting the divisor from the dividend the proper number of times.

The program below will produce A mod B, the remainder when we divide A by B, and store the remainder in location x3100 and the divisor (B) is initially in location x3101.

Your job: fill in the missing instructions so the program works correctly.

```
0011 0000 0000 0000                  ; Starts at x3000

0010 001 011111111                   ; Loads A from x3100 into R1
0010 010 011111111                   ; Loads B from x3101 into R2

1001 010 010 111111                  ; NOT R2 → R2

0001 010 010 1 00001                 ; Adds 1 to R2 and stores in R2
0001 001 001 000 010                 ; Adds R1 and R2 and stores result in R1
0000 011 111111110                   ; If zero or positive, branches to previous instruction

0010 010 011111010                   ; Loads value from x3101 into R2

0001 001 001 000 010                 ; ADD R2,R1 → R1

0011 001 011111001                   ; Stores value in R1 into x3102

1111 0000 00100101                   ; Halt
```

Name:______________________________

Problem 6 (15 points)

One of the opcodes we have not considered yet in class is LDI (1010), which stands for Load Indirect. It is almost identical to LD (0010), which you already know. The difference is as follows: Both instructions use PC' + offset9 to compute a memory address. In the case of LD, the contents of the corresponding memory location is the value that gets loaded into the specified register. In the case of LDI, the contents of that memory location is the ADDRESS of the memory location that contains the value to be loaded into the specified register.

For example, if the instruction 0010 011 011111111 is in location x3000, and location x3100 contains x5000, you know that the computer will load x5000 into register 3. If instead, the instruction 1010 011 011111111 is in location x3000, and x3100 contains x5000, and location x5000 contains x0008, the computer would load the value x0008 into register 3.

All instructions are processed, clock cycle by clock cycle, as the computer goes through the FETCH, DECODE, etc. phases of the instruction cycle.

Your job: Fill in the table below, showing the contents of each of the registers at the end of each clock cycle. Use as many clock cycles as you need. It is not necessary to use all that are provided.

At the start of clock cycle 1:

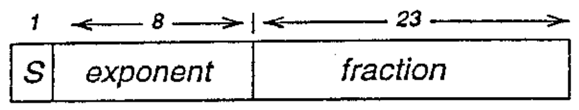
Memory Location	Value
x3030	xA882
x30B3	x6000
x6000	x0002

Note: Recall that during a clock cycle, combinational logic is carried out, based on the values in the registers at the start of that clock cycle. The results of that combinational logic are latched into the registers at the end of the clock cycle (and ONLY at the end of the clock cycle), so as to be available to the logic in the next (subsequent) clock cycle.

Note: Assume a memory access takes one clock cycle. That is, the memory read uses the contents available in MAR at the start of the clock cycle to determine the location to be read, and latches the data from that location into MDR at the end of the clock cycle.

Note: We have provided the data sheet for LDI from Appendix A, and the state machine of the LC-3 from Appendix C, although neither may be necessary.

	PC	IR	MAR	MDR	Reg 4	
Before Execution	x3030	x1540	x302F	x1540	x0005	
At end of cycle 1	x3031		x3030			
At end of cycle 2				xA882		
At end of cycle 3		xA882				
At end of cycle 4						DECODE
At end of cycle 5			x30B3			
At end of cycle 6				x6000		
At end of cycle 7			x6000			
At end of cycle 8				x0002		
At end of cycle 9					x0002	
At end of cycle 10						
At end of cycle 11						
At end of cycle 12						



$$N = (-1)^S \times 1.fraction \times 2^{exponent - 127}, 1 \leq exponent \leq 254$$

Figure 2.2 The floating point data type

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

[illegible]

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD+	0001				DR			SR1			0	00		SR2		
ADD+	0001				DR			SR1			1	imm5				
AND+	0101				DR			SR1			0	00		SR2		
AND+	0101				DR			SR1			1	imm5				
BR	0000				n	z	p	PCoffset9								
JMP	1100				000			BaseR			000000					
JSR	0100				1	PCoffset11										
JSRR	0100				0	00		BaseR			000000					
LD+	0010				DR			PCoffset9								
LDI+	1010				DR			PCoffset9								
LDR+	0110				DR			BaseR			offset6					
LEA+	1110				DR			PCoffset9								
NOT+	1001				DR			SR			111111					
RET	1100				000			111			000000					
RTI	1000				000000000000											
ST	0011				SR			PCoffset9								
STI	1011				SR			PCoffset9								
STR	0111				SR			BaseR			offset6					
TRAP	1111				0000				trapvect8							
reserved	1101															

Formats of the entire LC-3 instruction set. NOTE: + indicates instructions that modify condition codes

LDI Load Indirect

Assembler Format

LDI DR, LABEL

Encoding

15 12	11 9	8 0
1010	DR	PCoffset9

Operation

```
DR = mem[mem[PC† + SEXT(PCoffset9)]];
setcc();
```

Description

An address is computed by sign-extending bits [8:0] to 16 bits and adding this value to the incremented PC. What is stored in memory at this address is the address of the data to be loaded into DR. The condition codes are set, based on whether the value loaded is negative, zero, or positive.

Example

LDI R4, ONEMORE ; R4 $\leftarrow$ mem[mem[ONEMORE]]

† This is the incremented PC.

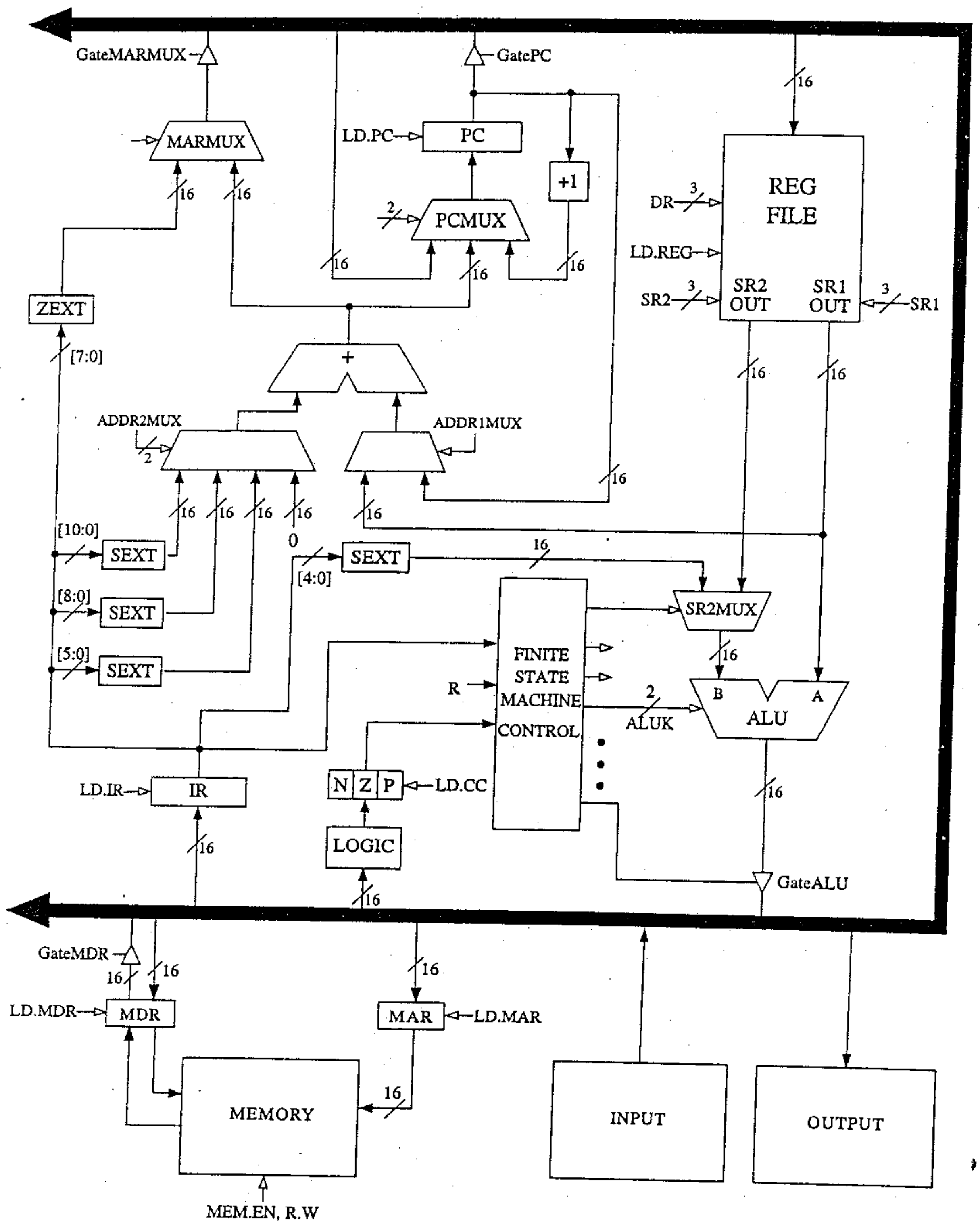


Figure 5.18 The data path of the LC-3

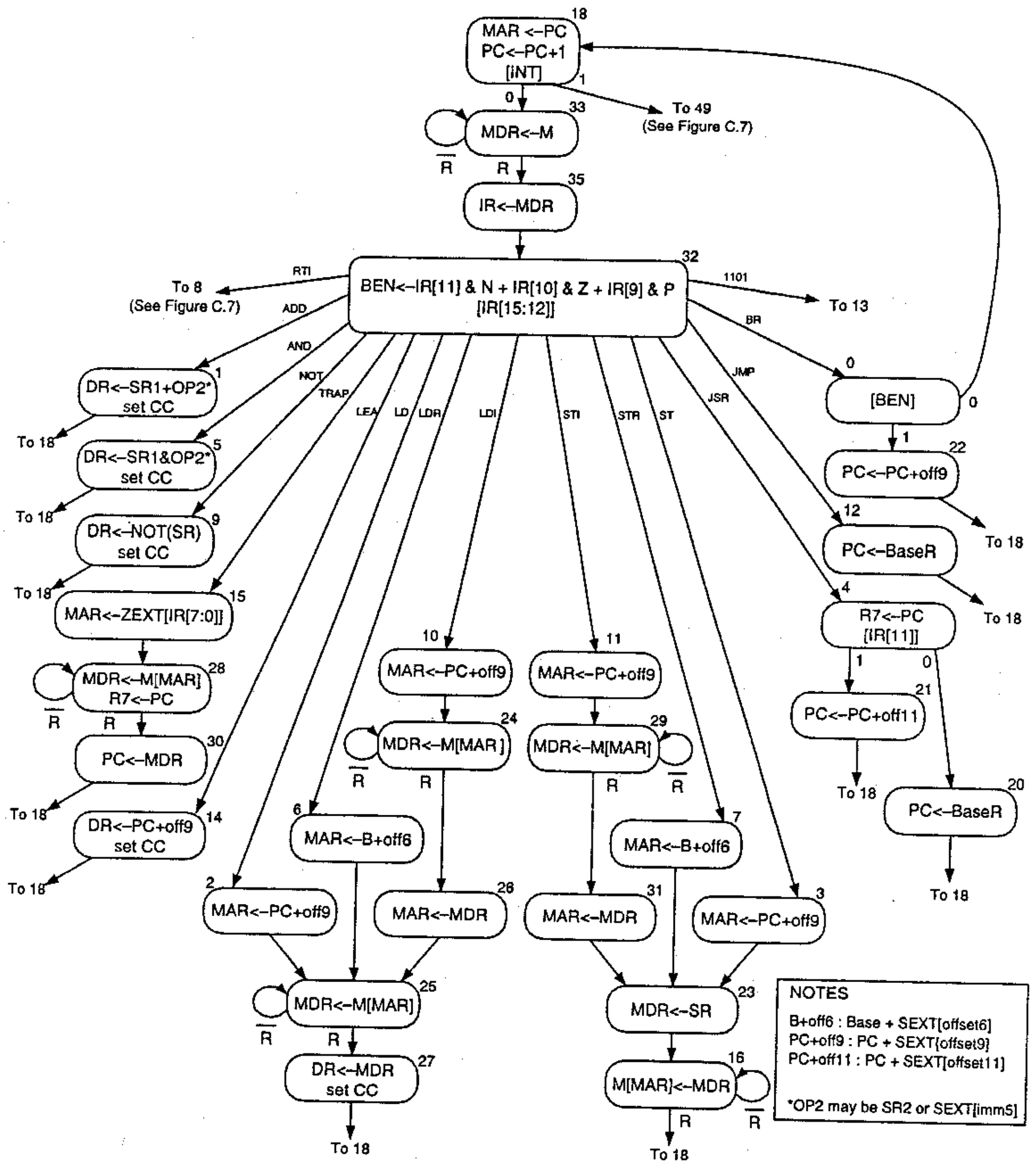


Figure C.2 A state machine for the LC-3