

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

EE 306, Fall 2017

Yale Patt, Instructor

Stephen Pruet, Siavash Zangeneh, Aniket Deshmukh, Zachary Susskind, Meiling Tang, Jiahan Liu

Exam 2, November 15, 2015

Name: Solution

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: Solution

Problem 1. (20 points):

Part a. (5 points): Construct the symbol table for the following LC-3 assembly language program:

Symbol Table:

Symbol	Address
AGAIN	x4502
BOBO	x4505
SAM	x4521
JOE	x4528

```

.ORIG x4500
LD R2, BOBO
LD R3, SAM
AGAIN ADD R3, R3, R2
ADD R2, R2, #-1
BRnzp SAM
BOBO .STRINGZ "Why are you asking me this?"
SAM BRnp AGAIN
TRAP x25
.BLKW 5
JOE .FILL x7777
.END
  
```

Part b. (5 points): A stack machine executes the following 6 instructions:

```

Push 5
Push 4
ADD
Push 6
MUL
POP
  
```

What value is popped by the last instruction?

54

Assume the stack is empty, with R6 = xFE00. Before the stack machine executes "Push 5", the contents of memory locations xFDFA to xFDFF are shown. Show the contents of memory and R6 after the six operations above are executed.

Address	Before	After
xFDFA	x0000	x0000
xFDFB	x0000	x0000
xFDFC	x0000	x0000
xFDFD	x0000	x0000
xFDFE	x0000	x0006
xFDFE	x0000	x0036

R6: xFE00

Name: Solution

Part c. (5 points): We want to move a number from A to B. List all LC-3 opcodes that can be used to accomplish this in one instruction when A,B are as specified at the top of each column. We have provided four slots for each column. Use as many as you need.

A is memory location B is a register R0-R7	A is a register R0-R7 B is a register R0-R7	A is a register R0-R7 B is memory location	A is memory location B is memory location
LD	ADD	ST	
LDR	AND	STR	
LDI		STI	

Part d. (5 points): What is wrong with the following program fragment?

```

...
...
LD R0, A
SPIN LDI R1, KBSR
BRzp SPIN
STI R0, KBDR
...
...
RET
KBSR .FILL xFE00
KBDR .FILL xFE02
A .FILL x0041
...
...

```

It attempts to write to keyboard which is an input-only device

Name: Solution

Problem 2. (20 points): Since ASCII codes consist of 8 bits each, we can store two ASCII codes in one word of LC-3 memory. If a user types $2n$ characters on the keyboard, followed by the <ENTER>key, the subroutine PACK on the next page will store the corresponding ASCII codes into n sequential memory locations, two per memory location, starting at location A.

You may assume that a user never enters an odd number of characters.

Part a. (15 points): Your job: Fill in the blanks in the program.

Part b. (5 points): If a user types the string **Please help!** followed by the <ENTER> key, what does the program do?

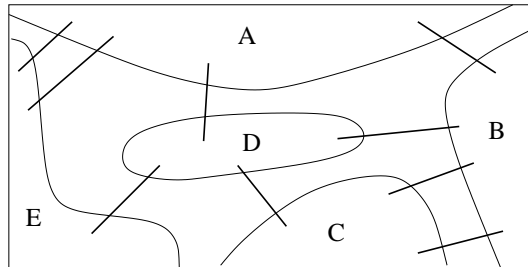
SAVER7 gets overwritten and the subroutine returns to x7021 and executes PACK again.

Name: Solution

```
.ORIG x7020
PACK    ST R7, SAVER7
        ST R6, SAVER6
        ST R4, SAVER4
        ST R3, SAVER3
        LEA R6, A      ; R6 is the pointer
        AND R4, R4, #0
        ADD R4, R4, #8 ; R4 is our counter
        AND R3, R3, #0
        LEA R0, PROMPT
        TRAP x22
POLL    LDI R7, KBSR
        BRzp POLL
        LDI R7, KBDR
        LD R0, NEG_LF
        ADD R0, R7, R0
        BRz DONE
        ADD R4, R4, #0
        BRz NOSHIFT
SHIFT   ADD R7, R7, R7
        ADD R4, R4, #-1
        BRp SHIFT
        ADD R3, R7, #0
        BRnzp POLL
NOSHIFT ADD R3, R3, R7
        STR R3, R6, #0
        ADD R6, R6, #1
        ADD R4, R4, #8
        BRnzp POLL
DONE    LD R7, SAVER7
        LD R6, SAVER6
        LD R4, SAVER4
        LD R3, SAVER3
        LEA R0, A      ; Returns a pointer to the characters
        RET
KBSR    .FILL xFE00
KBDR    .FILL xFE02
NEG_LF  .FILL xFFF6
PROMPT  .STINGZ "Please enter a string: "
A       .BLKW #5
SAVER7  .BLKW #1
SAVER6  .BLKW #1
SAVER4  .BLKW #1
SAVER3  .BLKW #1
        .END
```

Name: Solution

Problem 3. (20 points): Many cities, like New York City, Stockholm, Konigsberg, etc. consist of several areas, connected by bridges. The figure below shows a map of FiveParts, a city made up of five areas A,B,C,D,E, with the areas connected by 9 bridges as shown.



The following program prompts the user to enter two areas, and then stores the number of bridges from the first area to the second in location x4500. Your job: On the next page, design the data structure for the city of FiveParts that the program below will use to count the number of bridges between two areas.

```
.ORIG x3000
LEA R0, FROM
TRAP x22
TRAP x20      ; Inputs a char without banner
NOT R1, R0
ADD R1, R1, #1
LEA R0, TO
TRAP x22
TRAP x20
NOT R0, R0
ADD R0, R0, #1
AND R5, R5, #0
LDI R2, HEAD
SEARCH      BRz DONE
            LDR R3, R2, #0
            ADD R7, R1, R3
            BRz FOUND_FROM
            LDR R2, R2, #1
            BRnzp SEARCH
FOUND_FROM  ADD R2, R2, #2
NEXT_BRIDGE LDR R3, R2, #0
            BRz DONE
            LDR R4, R3, #0
            ADD R7, R0, R4
            BRnp SKIP
            ADD R5, R5, #1 ; Increment Counter
SKIP        ADD R2, R2, #1
            BRnzp NEXT_BRIDGE
DONE        STI R5, ANSWER
            HALT
HEAD        .FILL x3050
ANSWER      .FILL x4500
FROM        .STRINGZ "FROM: "
TO          .STRINGZ "TO: "
            .END
```

Name: Solution

Your job is to provide the contents of the memory locations that are needed to specify the data structure for the city of FiveParts, which is needed by the program on the previous page. We have given you the HEAD pointer for the data structure and in addition, five memory locations and the contents of those five locations. We have also supplied more than enough sequential memory locations after each of the five to enable you to finish the job. Use as many of these memory locations as you need.

Node for A

x4000	x0041
x4001	x A243
x4002	x A243
x4003	x BBBB
x4004	x 3100
x4005	x 3100
x4006	x 0000

HEAD pointer

x3050	x 4000
-------	--------

Node for B

xA243	x0042
xA244	x4100
xA245	x4000
xA246	x 4100
xA247	x 4100
xA248	xBBBB
xA249	x0000

Node for C

x4100	x0043
x4101	xBBBB
x4102	x A243
x4103	x A243
x4104	xBBBB
x4105	x0000
x4106	

Node for D

xB BBBB	x0044
xB BBBC	x 3100
xB BBBD	x 4000
xB BBBE	x A243
xB BBBF	x 4100
xB BBC0	x 3100
xB BBC1	x 0000

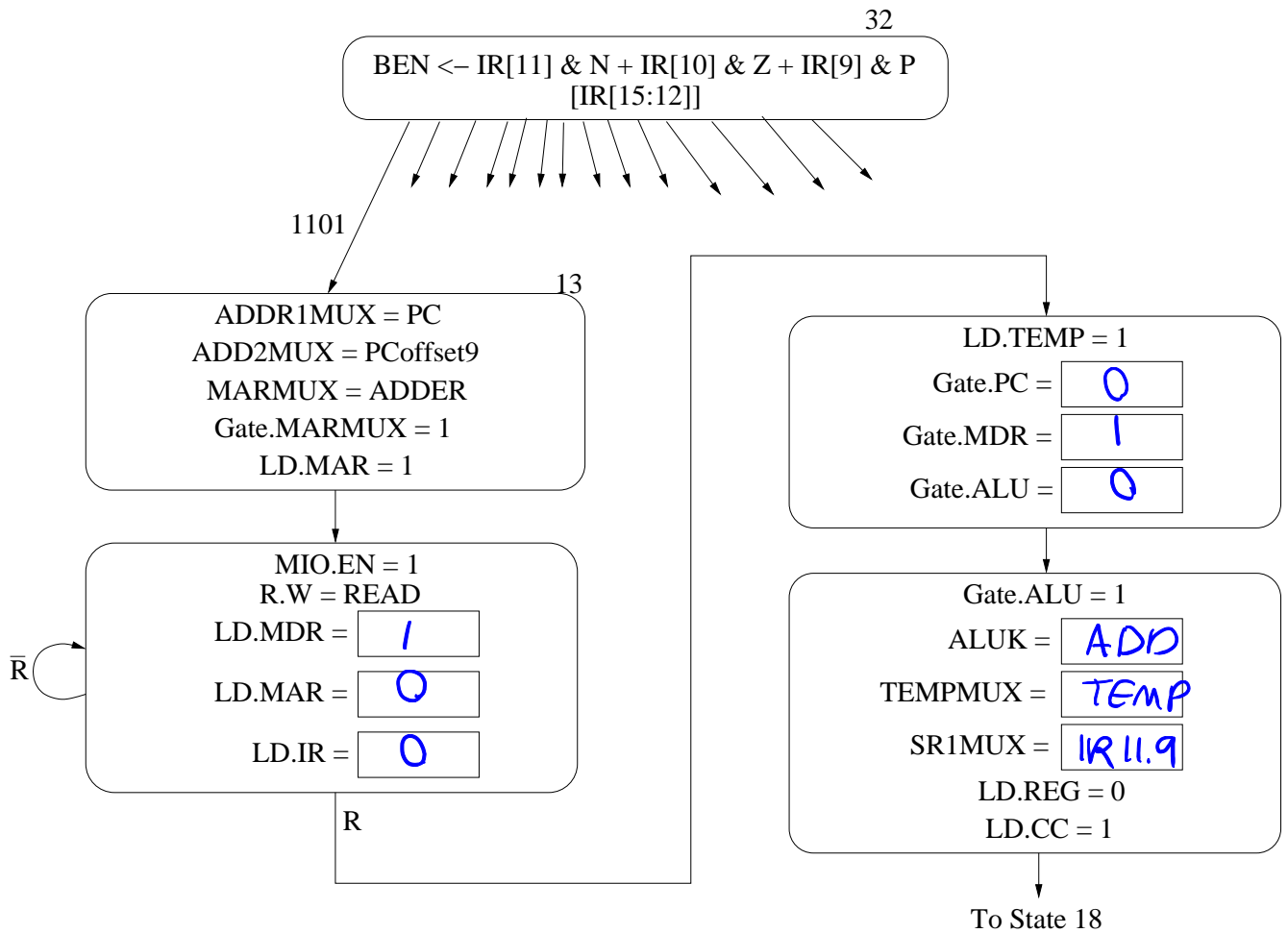
Node for E

x3100	x0045
x3101	x0000
x3102	x4000
x3103	x 4000
x3104	xBBBB
x3105	x0000
x3106	

NOTE: the order of islands in the linked list and the order of bridges per island do not matter

Name: Solution

Problem 4. (20 points): We wish to use the unused opcode 1101 to add a new instruction to the LC-3 ISA. This requires four new states in the state machine (shown below) and additions to the data path (shown on the next page).

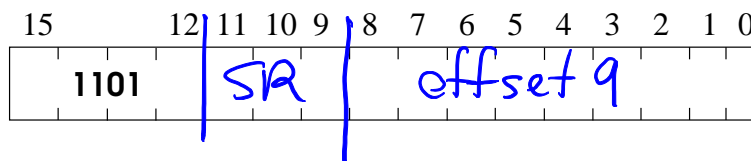


Part a. (5 points): Fill in the missing information in the four states. You can assume all control signals not shown are 0. A table of relevant control signals is included in your exam packet.

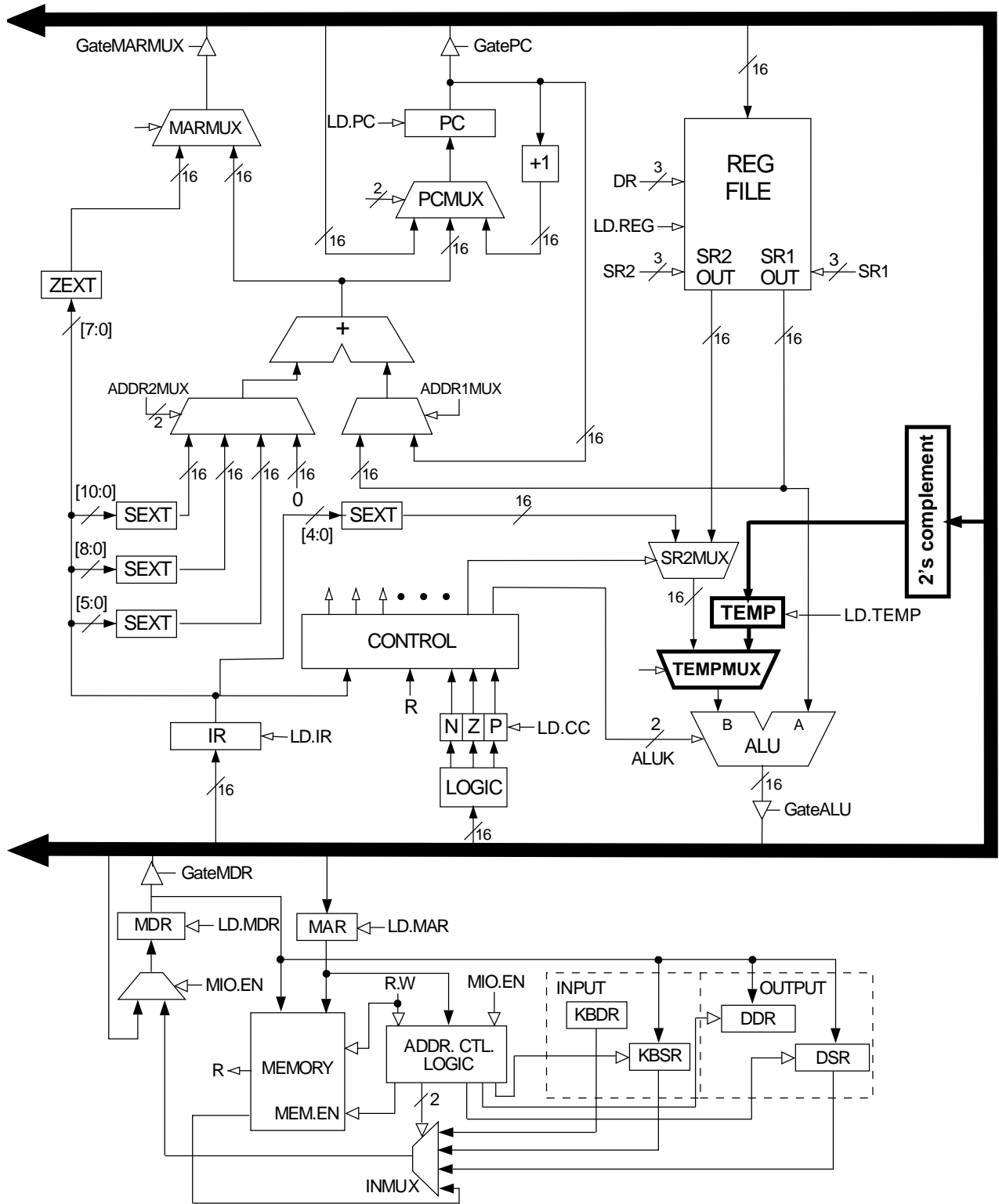
Part b. (10 points): What does the new instruction do (in 15 words or fewer)?

Compare a register with a memory location

Part c. (5 points): Identify the fields of the new instruction. Be sure you indicate clearly the correct bits for each field.

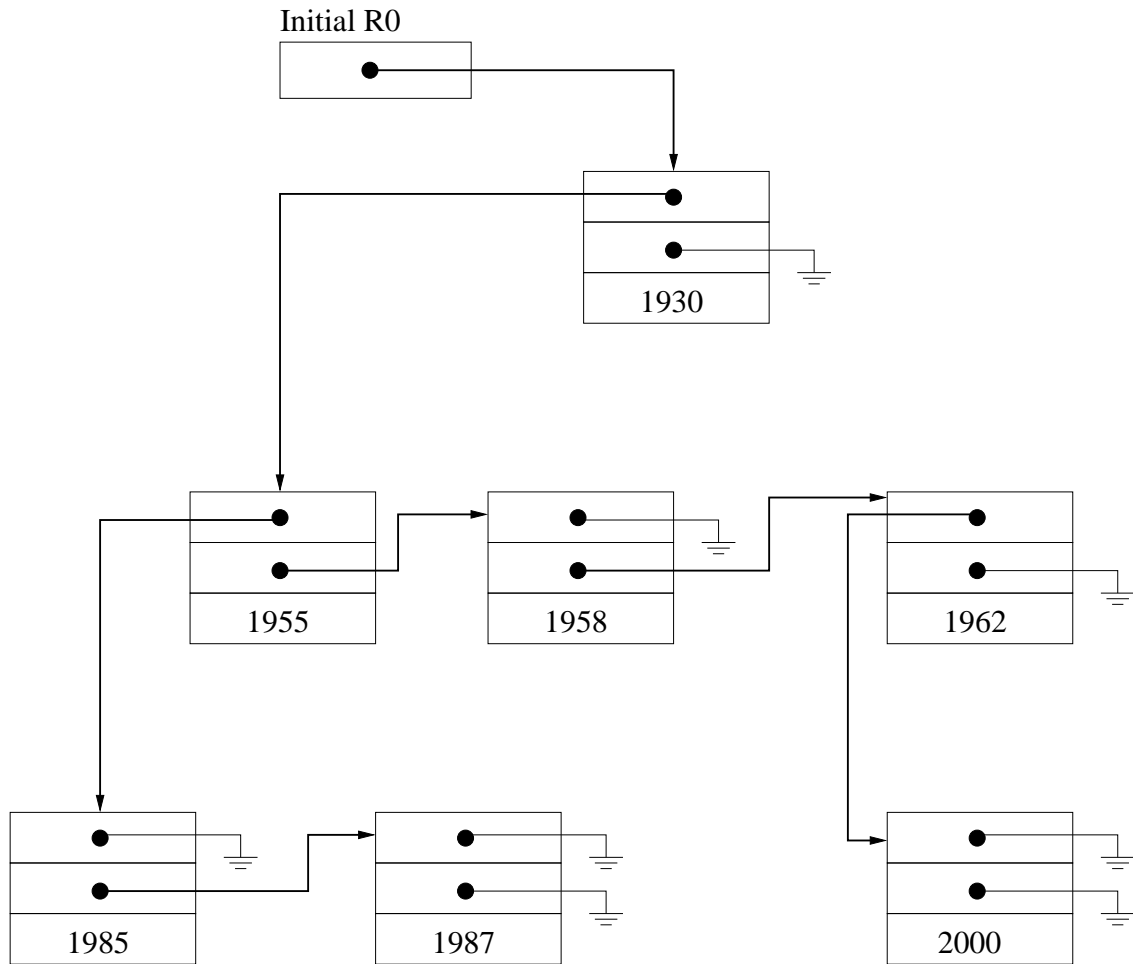


Name: Solution



Name: Solution

Problem 5. (20 points): Information about members of an extended family is stored in a tree. The first two words in each node are pointers to the oldest child and the next younger sibling. That is, the children of a parent are ordered according to age. We use the 3rd word in each node to represent the year the person was born.



Name: Solution

Part a. (15 points): The following recursive subroutine counts the number of family members who are born before 1960. R0 is a pointer to the root of a tree. R1 is the output count. Assume the main program initializes R1 to 0 and R6 to the stack pointer before calling the subroutine. The stack does not overflow during the execution of the subroutine. Fill in the missing instructions.

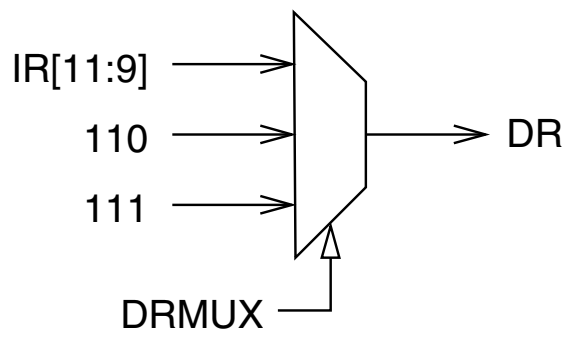
```
.ORIG x4000
COUNT  ADD R6, R6, #-1
        STR R7, R6, #0
        ADD R6, R6, #-1
        STR R2, R6, #0
        ADD R0, R0, #0
        BRz RESTORE
        LDR R2, R0, #2
        LD R7, NEG_VALUE
        ADD R7, R2, R7
        BRzp SKIP
        ADD R1, R1, #1
SKIP     ADD R2, R0, #0
        LDR R0, R2, #0
        JSR COUNT
        LDR R0, R2, #1
        JSR COUNT
        ADD R0, R2, #0
RESTORE  LDR R2, R6, #0
        ADD R6, R6, #1
        LDR R7, R6, #0
        ADD R6, R6, #1
        RET
NEG_VALUE .FILL #-1960
```

Part b. (5 points): Can we speed up the subroutine by eliminating visits to unnecessary nodes in the tree? How (in 20 words or fewer)?

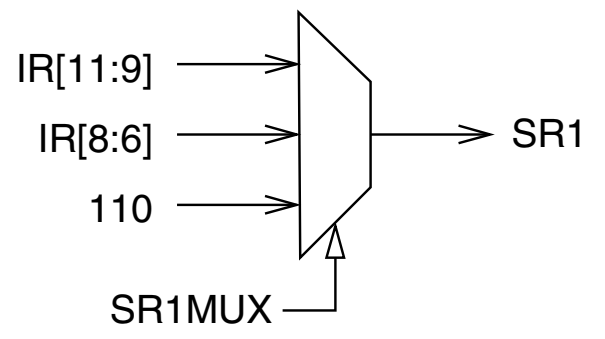
Do not recurse further if the current person is not born before 1960

Data Path Control Signals

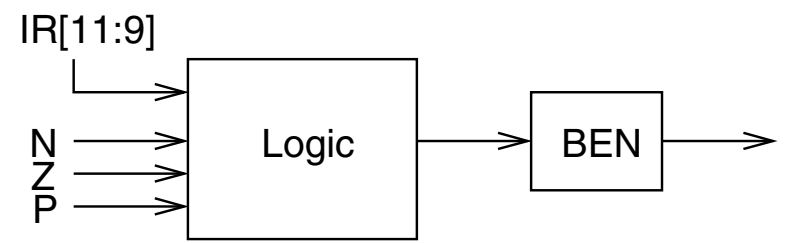
Signal Name	Signal Values
LD.MAR/1:	NO(0), LOAD(1)
LD.MDR/1:	NO(0), LOAD(1)
LD.IR/1:	NO(0), LOAD(1)
LD.REG/1:	NO(0), LOAD(1)
LD.CC/1:	NO(0), LOAD(1)
LD.TEMP/1:	NO(0), LOAD(1)
Gate.MARMUX/1:	NO(0), YES(1)
Gate.MDR/1:	NO(0), YES(1)
Gate.PC/1:	NO(0), YES(1)
Gate.ALU/1:	NO(0), YES(1)
ADDR1MUX/1:	PC(0), BaseR(1)
ADDR2MUX/2:	ZERO(00), offset6(01), PCoffset9(10), PCoffset11(11)
MARMUX/1:	IR7.0(0), ADDER(1)
SR1MUX/2:	11.9(00), 8.6(01), SP(10)
TEMPMUX/1:	OP2(0), TEMP(1)
ALUK/2:	ADD(00), AND(01), NOT(10), PASSA(11)
MIO.EN/1:	NO(0), YES(1)
R.W/1:	RD(0), WR(1)



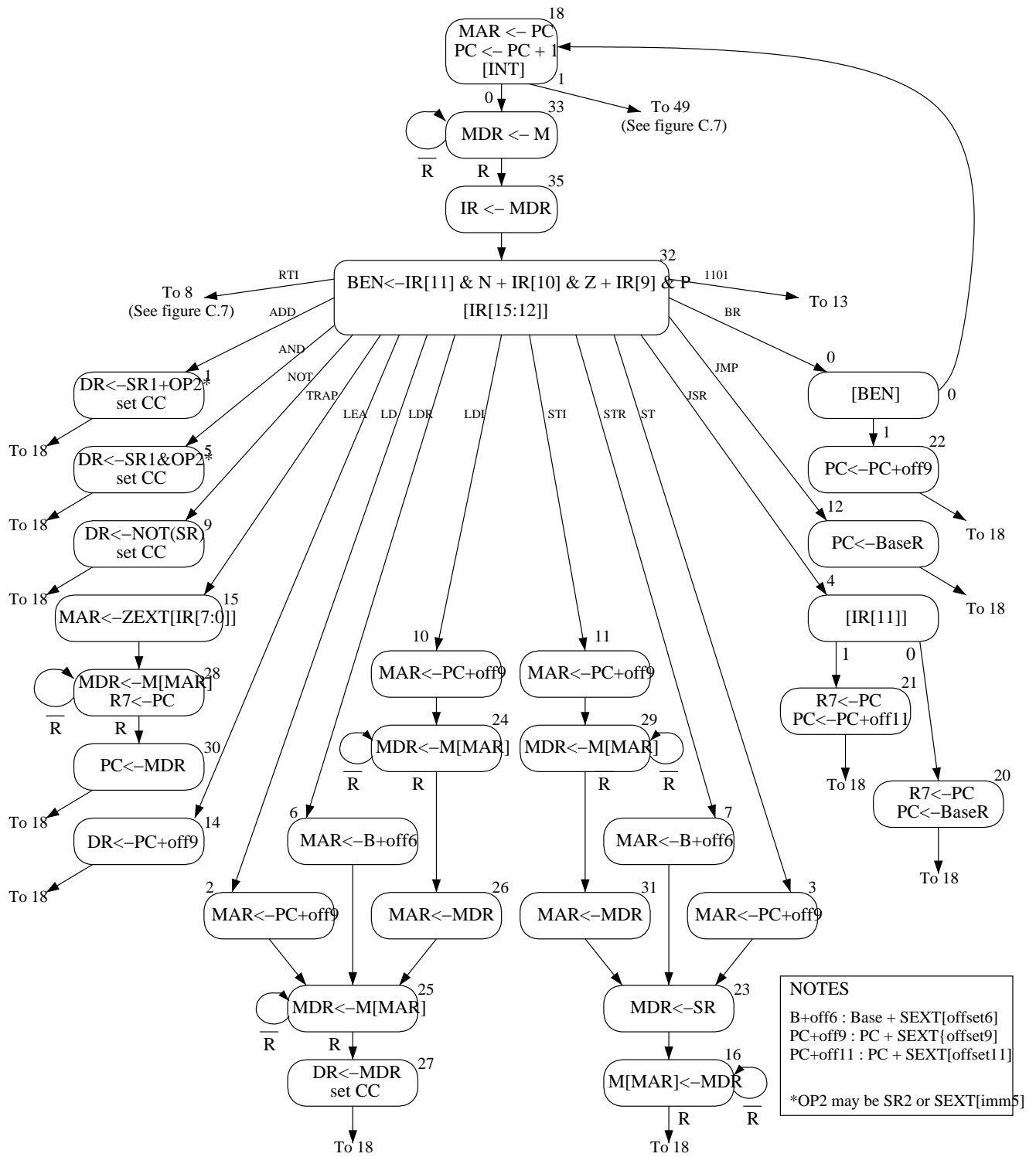
(a)

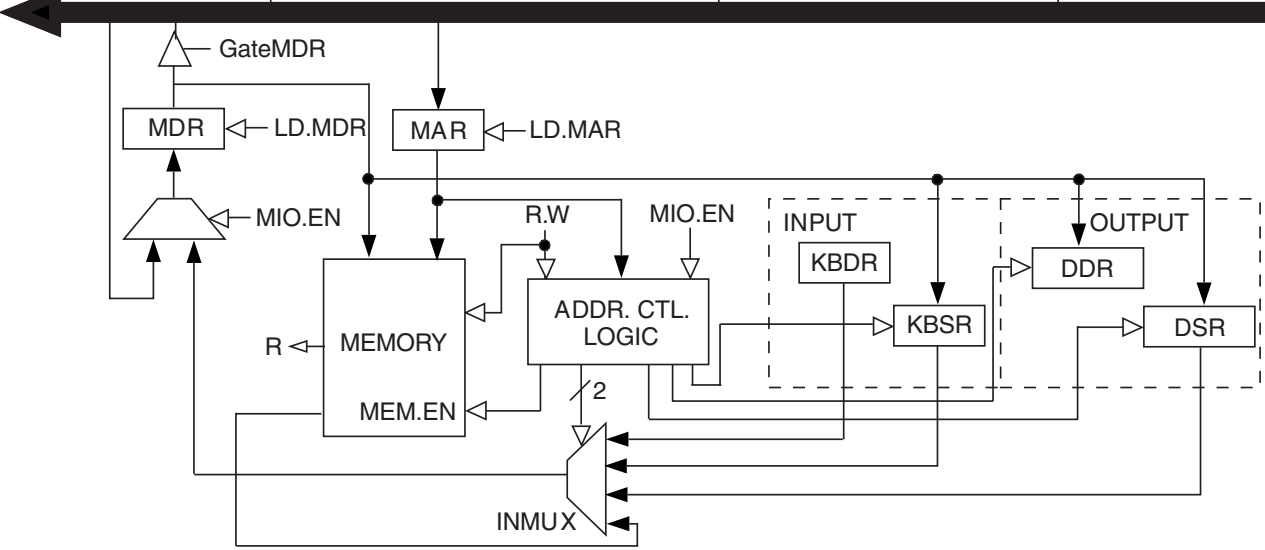
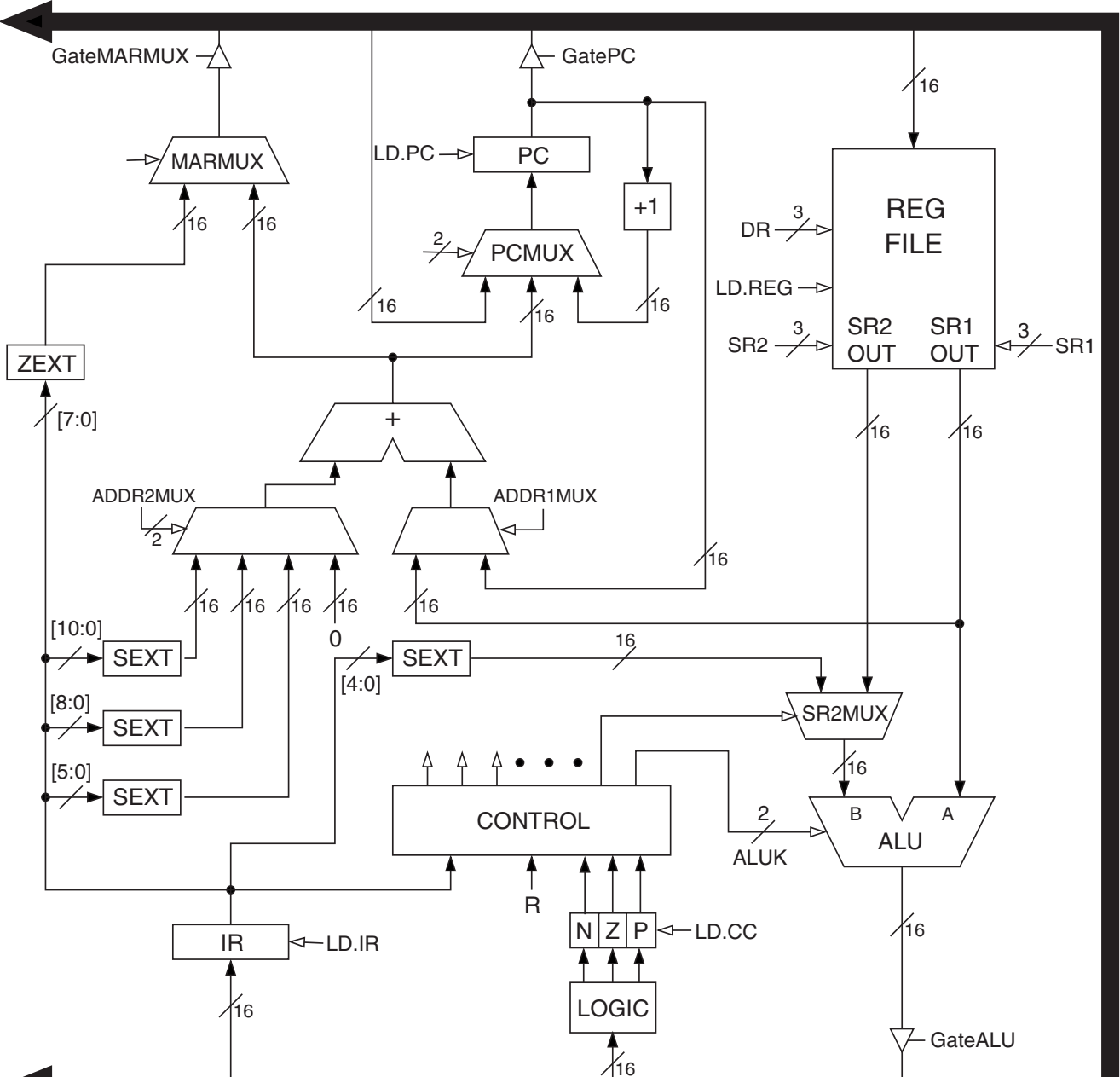


(b)



(c)





	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															

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.