ECE 120 Third Midterm Exam
Spring 2017

Tuesday, April 18, 2017

Name: SOLUTIONS

NetID: ______________

Discussion Section and TA name:

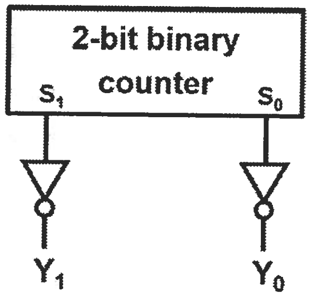
9:00 AM	[ ] AB1 Rui	
10:00 AM	[ ] AB2 Rui	
11:00 AM	[ ] AB3 Matt	
12:00 PM	[ ] AB4 Pawel	
1:00 PM	[ ] AB5 Pawel	
2:00 PM	[ ] AB6 Gowthami	[ ] ABA Huiren
3:00 PM	[ ] AB7 Gowthami	[ ] ABB Huiren
4:00 PM	[ ] AB8 Yu-Hsuan	[ ] ABC Sifan
5:00 PM	[ ] AB9 Yu-Hsuan	[ ] ABD Surya

- **Be sure that your exam booklet has 9 pages.**
- **Write your name, netid and check discussion section on the title page.**
- **Do not tear the exam booklet apart, except for the last two pages.**
- **Use backs of pages for scratch work if needed.**
- **This is a closed book exam. You may not use a calculator.**
- **You are allowed one handwritten 8.5 x 11" sheet of notes (both sides).**
- **Absolutely no interaction between students is allowed.**
- **Clearly indicate any assumptions that you make.**
- **The questions are not weighted equally. Budget your time accordingly.**
- **Show your work.**

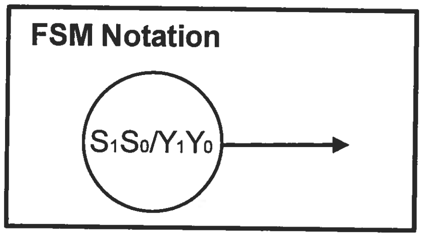
Problem 1	16 points	________
Problem 2	16 points	________
Problem 3	22 points	________
Problem 4	27 points	________
Problem 5	19 points	________
Total	100 points	________

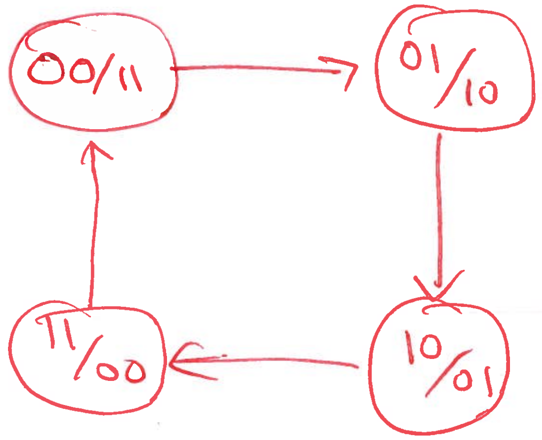
Problem 1 (16 points): Counter Design

The FSM drawn below consists of a 2-bit binary counter (S_1S_0 counts 0, 1, 2, 3, then returns to 0) and two inverters.



1. **(8 points)** Draw a state transition diagram for the FSM. Label each node with the state identifier S_1S_0 and the output Y_1Y_0. Do not cross transition arcs.





2. **(5 points)** In **TEN OR FEWER** words, what does the FSM do? *Hint: Look at the output sequence.*

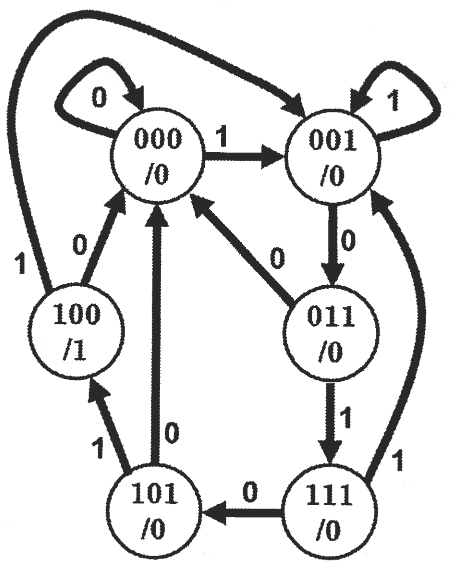
Counts downward: 3, 2, 1, 0, repeat.

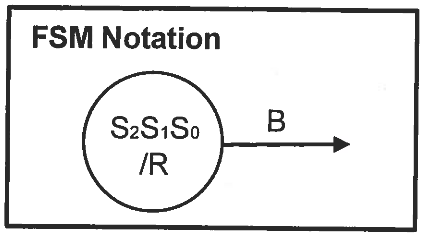
3. **(3 points)** In the design above, the counter is treated as a black box—in other words, only the outputs S_1 and S_0 are available. If instead you had access to all transistors in the 2-bit binary counter design (that produces S_1S_0), what is the minimum number of additional transistors needed to implement the FSM that produces Y_1Y_0?

Minimum number of additional transistors = 0 ($\overline{Q}$ outputs of flip-flops)

Problem 2 (16 points): Sequence Recognizer

Consider the state transition diagram shown below. Nodes are labeled with the state $S_2S_1S_0$ and the output R. Transition arcs are labeled with input B. **Assume that the FSM starts in state $S_2S_1S_0$=000.**





1. **(8 points)** Fill in the K-map to the right for the next state bit S_0^+ as a function of the current state $S_2S_1S_0$ and the input B.

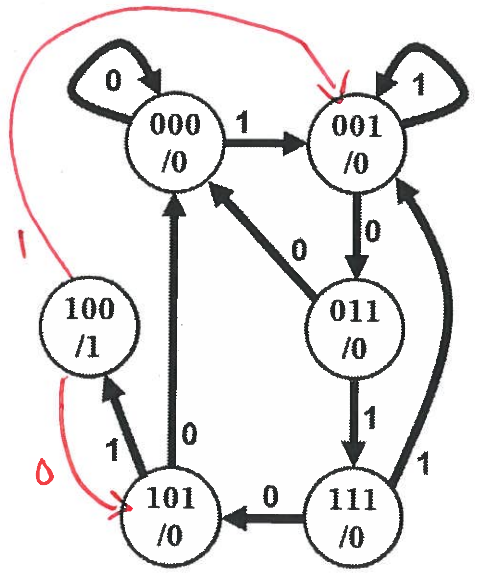
S_0^+

S_2S_1 \ S_0B	00	01	11	10
00	0	1	1	1
01	X	X	1	0
11	X	X	1	1
10	0	1	0	0

2. **(8 points)** The above FSM implements a sequence recognizer for non-overlapping instances of the sequence 10101. For example, if the FSM receives the input sequence 1010101, it produces the output sequence 0000100.

The figure to the right is the same as the one above, except that the two transitions coming from state 100 have been removed.

Draw and label the transitions from state 100 so that the resulting FSM recognizes overlapping instances of the sequence 10101. That is, if the FSM receives the input sequence 1010101, it produces the output sequence 0000101.



Problem 3 (22 points): Memory

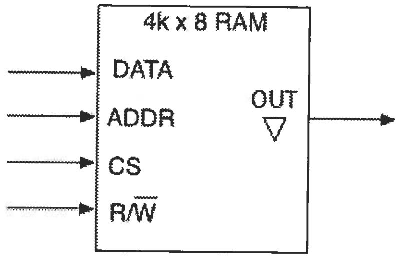
1. **(10 points)** The figure below shows the block diagram of a 4k x 8-bit RAM chip. Recall that 1k = 1024. Provide inputs to the RAM in order to store the **decimal value -6 in location (decimal) 30**. Answer in **binary** with the **correct number of bits**.

DATA 1111 1010

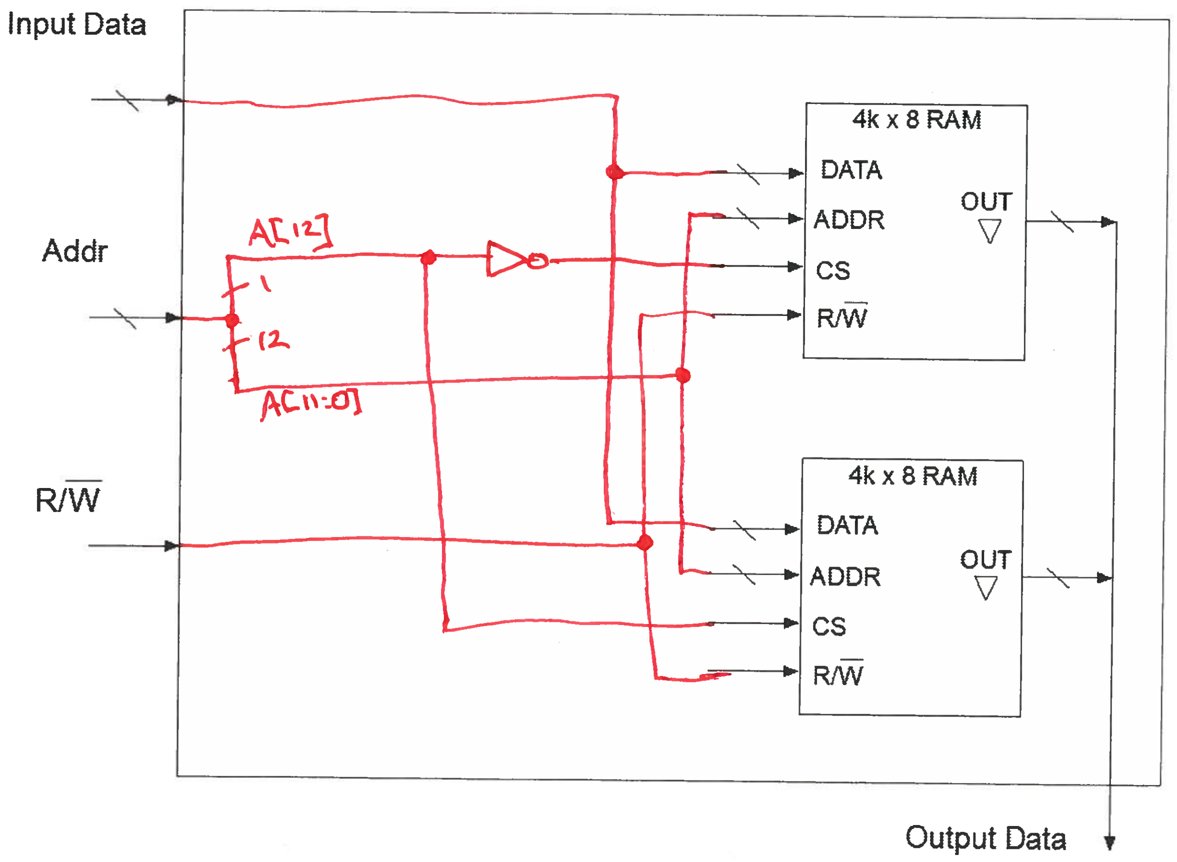
ADDR 0000 0001 1110

CS 1

R / $\overline{\text{W}}$ 0

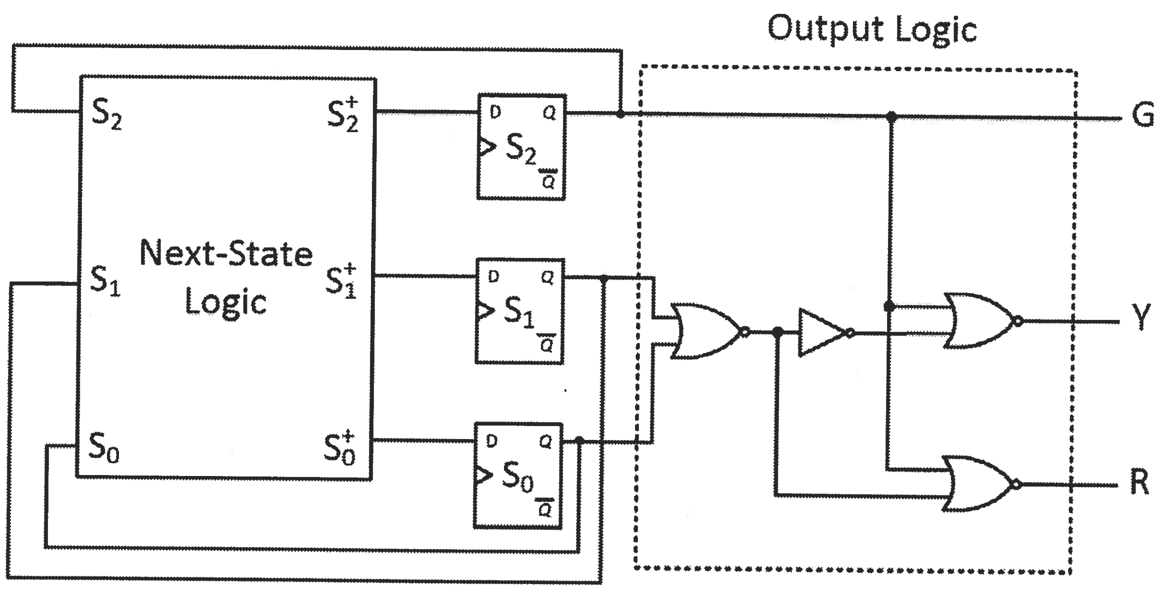


2. **(12 points)** Use two of these 4k x 8-bit RAM chips (with CS) to build an 8k x 8-bit RAM (without CS). **Draw wires** to complete the logic below. The Output Data lines have been drawn for you. You may use **at most one additional gate** (AND, OR, or NOT). Label wires as needed (for example, Addr[3:0]) and write the number of wires wherever a "slash" is drawn.



Problem 4 (27 points): FSM Design

The city of Urbana has the following traffic light controller:



The next-state logic guarantees that the **FSM is a 3-bit binary up-counter** ($S_2S_1S_0$ counts the sequence 0, 1, 2, ... , 7, 0, 1, 2, ...), while the output logic translates the current state $S_2S_1S_0$ into outputs for the three lights: G (green), Y (yellow), and R (red).

The mayor of Urbana has requested the help of the ECE Department to extend the design to cope with emergency vehicles (fire engines, ambulances, police, and so forth.) Since the ECE 120 instructors are *very* busy, and the timing for midterm 3 was just perfect, your task is to extend the FSM as follows:

1. When an emergency vehicle nears the traffic light, your extended FSM receives a new input **E=1** (otherwise, input E=0).
   a) If the **red light is lit (R=1)**, the FSM should **hold its current state** until E returns to 0 (after the emergency vehicle has passed).
   b) If the **yellow light is lit (Y=1)**, the FSM should **continue to advance** in the same way as in the original design.
   c) If the **green light is lit (G=1)**, the extended FSM must **light the yellow light (Y=1)** in the next cycle. (Safety is of utmost concern to the mayor. We don't want vehicles to suddenly stop and crash!)
2. If no emergency vehicle is present (**E=0**), the extended design should **continue to advance** in the same way as the original design behaved.

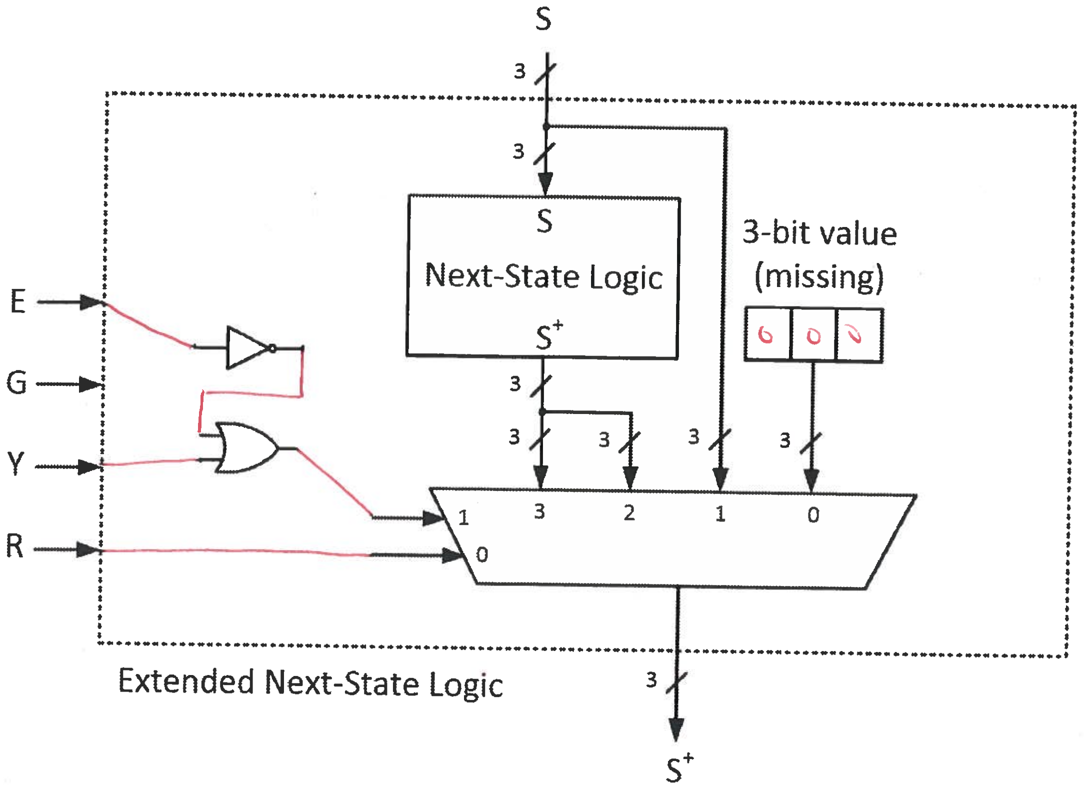
(The questions you need to answer are on the next page.)

Problem 4 (27 points): FSM Design, continued

1. **(18 points)** Write the state transition table for the extended design.

Current State			Input	Next State			Outputs		
S_2	S_1	S_0	E	S_2^+	S_1^+	S_0^+	G	Y	R
0	0	0	0	0	0	1	0	1	0
0	0	1	0	0	1	0	0	0	1
0	1	0	0	0	1	1	0	0	1
0	1	1	0	1	0	0	0	0	1
1	0	0	0	1	0	1	1	0	0
1	0	1	0	1	1	0	1	0	0
1	1	0	0	1	1	1	1	0	0
1	1	1	0	0	0	0	1	0	0
0	0	0	1	0	0	1	0	1	0
0	0	1	1	0	0	1	0	0	1
0	1	0	1	0	1	0	0	0	1
0	1	1	1	0	1	1	0	0	1
1	0	0	1	0	0	0	1	0	0
1	0	1	1	0	0	0	1	0	0
1	1	0	1	0	0	0	1	0	0
1	1	1	1	0	0	0	1	0	0

2. **(9 points)** The circuit below extends the next-state logic as specified in the previous page to cope with the new input E, but it has not been finished yet.
   - **a.** Write the missing 3-bit value in the provided boxes.
   - **b.** Finish the design by drawing the missing wires. **You are NOT allowed to add any other component to the design, only wires.** Inverted inputs are not available.



Problem 5 (19 points): LC-3 Interpretation and Assembly

The registers of an LC-3 processor currently have the values shown in the table to the right.

R0	x0000	R4	x4444	PC	x4401
R1	x1111	R5	x5555	IR	x11E1
R2	x2222	R6	x6666	MAR	x4400
R3	x3333	R7	xFFFF	MDR	x11E1

The table to the right shows some of the contents of the LC-3 processor's memory.

When the bits represent instructions, an interpretation has been provided for you in RTL.

address	contents	RTL interpretation
x4400	0001 000 111 1 00001	R0 ← R7+1
x4401	1001 101 111 111111	R5 ← NOT(R7)
x4402	0110 110 100 000001	R6 ← M[R4+1]
x4403	1011 111 001000000	M[M[PC+x040]]←R7
	. . .	. . .
x4443	0100 0100 0100 0101	(data: x4445)
x4444	0100 0100 0100 0110	(data: x4446)
x4445	1111 1110 1110 1101	(data: xFEED)
x4446	1110 1100 1110 1011	(data: xECEB)

Here's the question:

The LC-3 FSM **PROCESSES THREE INSTRUCTIONS**, starting with the fetch phase.

1. **(7 points)** Write a complete list of the sequence of values taken by the MAR register as the LC-3 processes these instructions. Use only as many lines as are necessary.

#1:	**x4400** (initial value)	#5:	x4403
#2:	x4401	#6:	x4444
#3:	x4402	#7:	x4446
#4:	x4445	#8:	

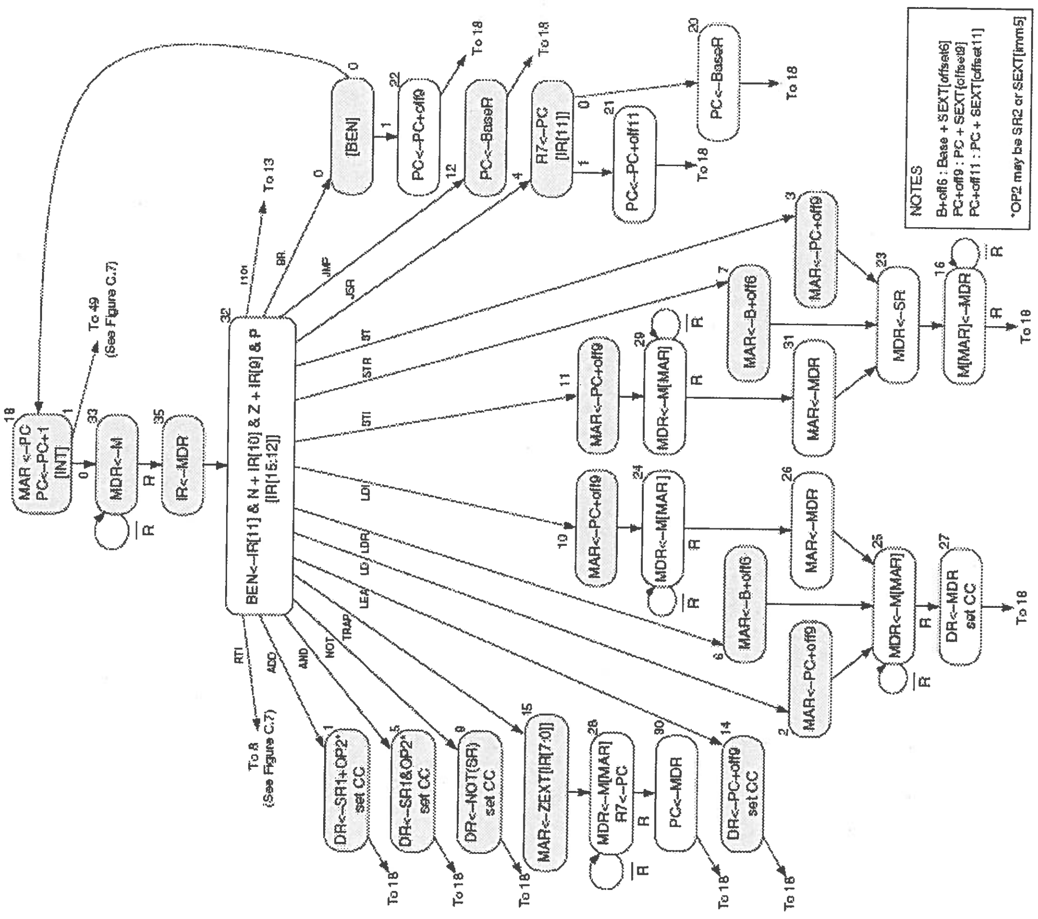
2. **(12 points)** Complete the tables below with the **FINAL** values (after processing of three instructions) of each register and memory location. **To receive credit, you must write your answers in hexadecimal.**

R5	x0000
R6	xFEED
R7	xFFFF

PC	x4404
IR	xBE40
MDR	xFFFF

memory address	contents
x4443	x4445
x4444	x4446
x4445	xFEED
x4446	xFFFF

LC-3 FSM



NOTES: RTL corresponds to execution (after fetch!); JSRR not shown

LC-3 Instructions

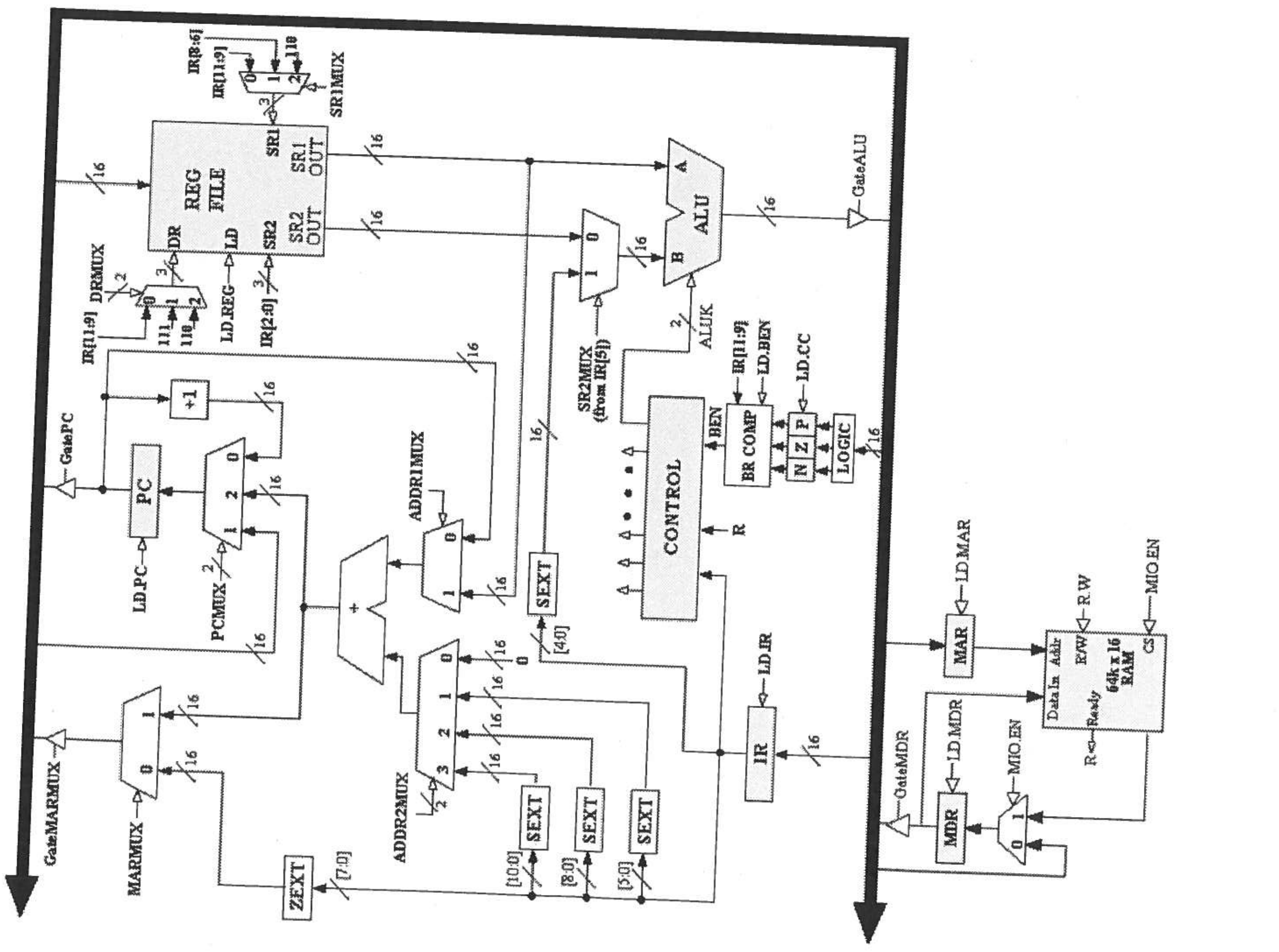
ADD	0001 | DR | SR1 | 0 | 00 | SR2	ADD DR, SR1, SR2	DR ← SR1 + SR2, Setcc
ADD	0001 | DR | SR1 | 1 | imm5	ADD DR, SR1, *imm5*	DR ← SR1 + SEXT(imm5), Setcc
AND	0101 | DR | SR1 | 0 | 00 | SR2	AND DR, SR1, SR2	DR ← SR1 AND SR2, Setcc
AND	0101 | DR | SR1 | 1 | imm5	AND DR, SR1, *imm5*	DR ← SR1 AND SEXT(imm5), Setcc
BR	0000 | n | z | p | PCoffset9	BR{nzp} *PCoffset9*	((n AND N) OR (z AND Z) OR (p AND P)): PC ← PC + SEXT(PCoffset9)
JMP	1100 | 000 | BaseR | 000000	JMP BaseR	PC ← BaseR
JSR	0100 | 1 | PCoffset11	JSR *PCoffset11*	R7 ← PC, PC ← PC + SEXT(PCoffset11)
TRAP	1111 | 0000 | trapvect8	TRAP *trapvect8*	R7 ← PC, PC ← M[ZEXT(trapvect8)]
LD	0010 | DR | PCoffset9	LD DR, *PCoffset9*	DR ← M[PC + SEXT(PCoffset9)], Setcc
LDI	1010 | DR | PCoffset9	LDI DR, *PCoffset9*	DR ← M[M[PC + SEXT(PCoffset9)]], Setcc
LDR	0110 | DR | BaseR | offset6	LDR DR, BaseR, *offset6*	DR ← M[BaseR + SEXT(offset6)], Setcc
LEA	1110 | DR | PCoffset9	LEA DR, *PCoffset9*	DR ← PC + SEXT(PCoffset9), Setcc
NOT	1001 | DR | SR | 111111	NOT DR, SR	DR ← NOT SR, Setcc
ST	0011 | SR | PCoffset9	ST SR, *PCoffset9*	M[PC + SEXT(PCoffset9)] ← SR
STI	1011 | SR | PCoffset9	STI SR, *PCoffset9*	M[M[PC + SEXT(PCoffset9)]] ← SR
STR	0111 | SR | BaseR | offset6	STR SR, BaseR, *offset6*	M[BaseR + SEXT(offset6)] ← SR

LC-3 Datapath Control Signals

Signal	Description
LD.MAR	= 1, MAR is loaded
LD.MDR	= 1, MDR is loaded
LD.IR	= 1, IR is loaded
LD.PC	= 1, PC is loaded
LD.REG	= 1, register file is loaded
LD.BEN	= 1, updates Branch Enable (BEN) bit
MARMUX	= 0, chooses ZEXT IR[7:0] = 1, chooses address adder output
ADDR1MUX	= 0, chooses PC = 1, chooses reg file SR1OUT
ADDR2MUX	= 00, chooses "0...00" = 01, chooses SEXT IR[5:0] = 10, chooses SEXT IR[8:0] = 11, chooses SEXT IR[10:0]
PCMUX	= 00, chooses PC + 1 = 01, chooses system bus = 10. chooses address adder output
SR1MUX	= 00, chooses IR[11:9] = 01, chooses IR[8:6] = 10, chooses "110"

Signal	Description
LD.CC	= 1, updates status bits from system bus
GateMARMUX	= 1, MARMUX output is put onto system bus
GateMDR	= 1, MDR contents are put onto system bus
GateALU	= 1, ALU output is put onto system bus
GatePC	= 1, PC contents are put onto system bus
MIO.EN	= 1, Enables memory, chooses memory output for MDR input = 0, Disables memory, chooses system bus for MDR input
R.W	= 1, M[MAR]<-MDR when MIO.EN = 1 = 0, MDR<-M[MAR] when MIO.EN = 1
ALUK	= 00, ADD = 01, AND = 10, NOT A = 11, PASS A
DRMUX	= 00, chooses IR[11:9] = 01, chooses "111" = 10, chooses "110"

LC-3 Datapath


REG FILE
SR1
SR2
DR
LD
SR1 OUT
SR2 OUT
SR1MUX
DRMUX
LD.REG
IR[11:9]
IR[8:6]
IR[2:0]
110
111
ALU
A
B
ALUK
GateALU
SR2MUX (from IR[5])
CONTROL
BEN
BR COMP
N Z P
LOGIC
LD.BEN
LD.CC
R
PC
+1
LD.PC
PCMUX
GatePC
ADDR1MUX
ADDR2MUX
MARMUX
GateMARMUX
ZEXT
SEXT
[10:0]
[8:0]
[5:0]
[4:0]
[7:0]
IR
LD.IR
MAR
LD.MAR
MDR
LD.MDR
GateMDR
MIO.EN
R.W
64k x 16 RAM
Addr
Data In
Ready
CS
16