ECE 120 Third Midterm Exam
Fall 2017

Tuesday, November 28, 2017

Name: SOLUTIONS

NetID: ______________

Discussion Section and TA name:

11:00 AM	[ ] AD1 Ruby	[ ] AD8 Matt
12:00 PM	[ ] AD2 David	[ ] AD9 Matt
1:00 PM	[ ] AD3 David	[ ] ADA Yu-Hsuan
2:00 PM	[ ] AD4 Wanzheng	[ ] ADB Zhaoheng
3:00 PM	[ ] AD5 Wanzheng	
4:00 PM	[ ] AD6 Spencer	[ ] ADC Ruby
5:00 PM	[ ] AD7 Spencer	[ ] ADD Zhaoheng

- **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	19 points	________
Problem 2	12 points	________
Problem 3	23 points	________
Problem 4	24 points	________
Problem 5	22 points	________
Total	100 points	________

Problem 1 (19 points): FSM Design

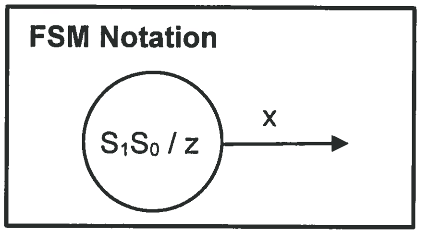
In this problem you will implement an FSM that recognizes a **01** sequence. The circuit has one input ***x***, one output ***z***, and the output is 1 if and only if the pattern **01** has been detected in the input stream.

Example:

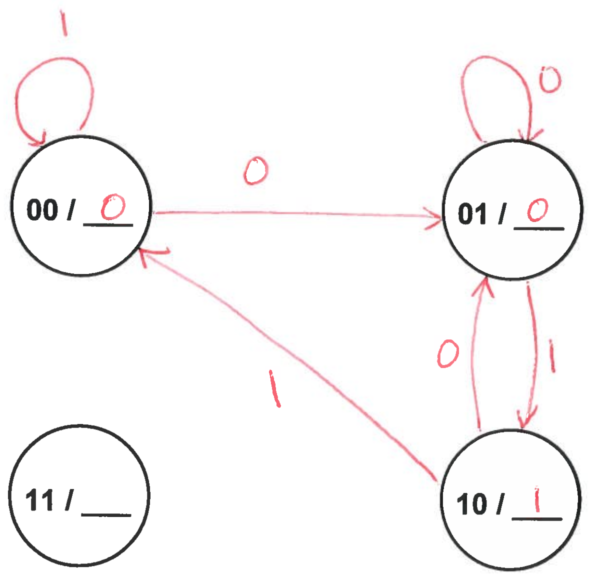
```
Input:   x = 0  0  1  1  1  0  1  0  1  1  1  1  0  1  1  0  1  . .
Output:  z =    0  0  1  0  0  0  1  0  1  0  0  0  0  1  0  0  1  . .
```

Note that the output sequence is delayed by 1 clock cycle compared to the input sequence because the output is a function of the flip-flop outputs.

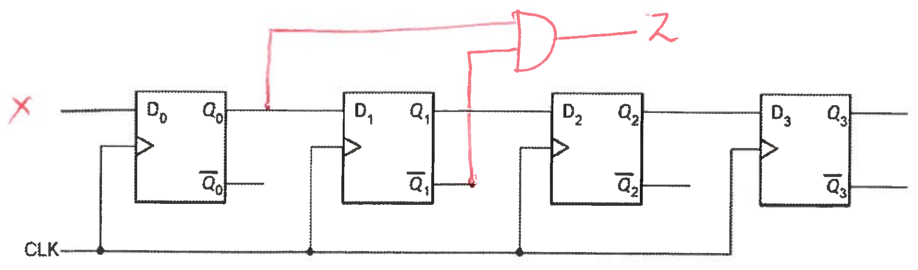
1. **(14 points)** Draw the **Moore** state diagram and label the outputs and inputs. Give the meaning of each state. **To get full credit, use the minimum number of states.**



State	Meaning
"Start" 00	Pattern not yet seen
01	0 of pattern seen
10	01 ~~of~~ seen
11	Not used.



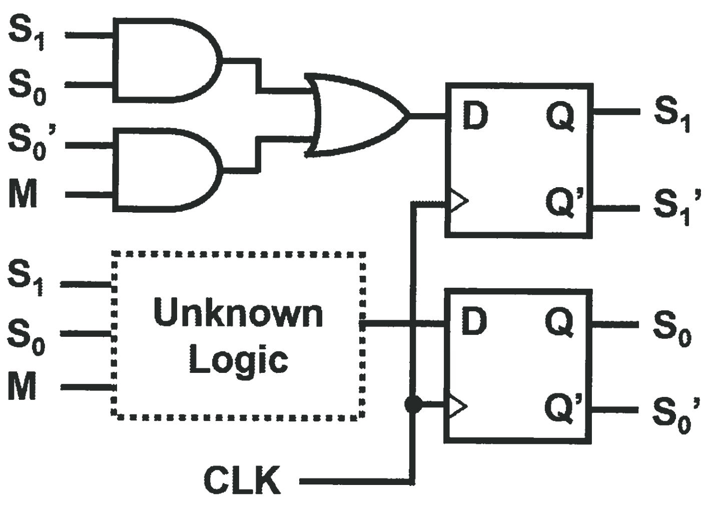
2. **(5 points)** Shown below is a 4-bit **shift register**, constructed with 4 positive-edge-triggered D flip-flops. Use this shift register and **only one gate** (NOT, AND, OR, NAND, NOR, XOR, or XNOR) to implement a circuit, which recognizes **01** *just like the example at the top of the page*. Be sure to label input ***x*** and output ***z***.



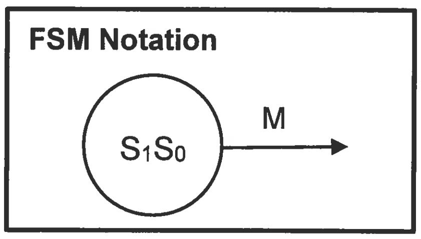
Problem 2 (XX points): FSM Analysis

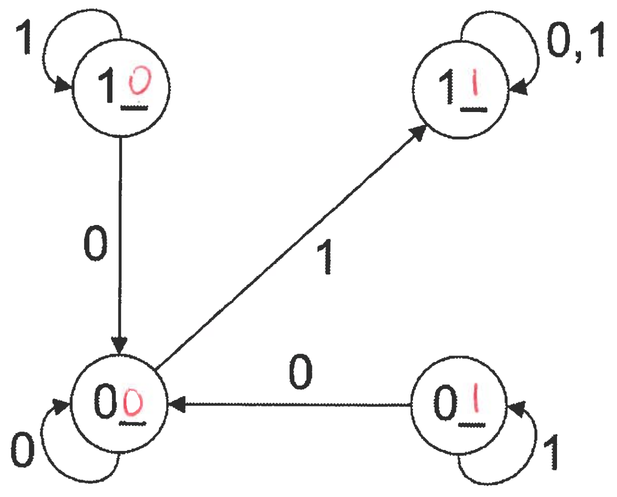
1. **(4 points)** Write the next state equation for S_1^+ from the FSM circuit given below.

$S_1^+ =$ $S_1S_0 + S_0'M$



2. **(8 points)** Complete the state transition diagram (shown below) so that it corresponds to the FSM circuit above.

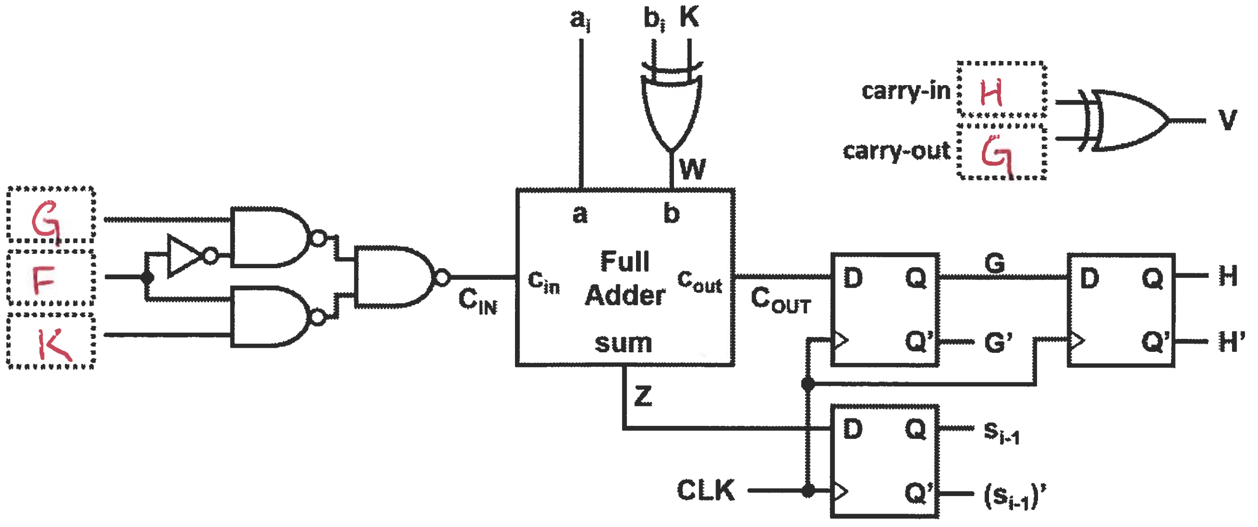




Problem 3 (23 points): Serialized Design

The circuit below will be a serialized **n-bit 2's complement adder/subtractor circuit.** The inputs $a_{n-1}a_{n-2}\ldots a_1a_0$ and $b_{n-1}b_{n-2}\ldots b_1b_0$ are fed in serially (from LSB to MSB) and the output $s_{n-1}s_{n-2}\ldots s_1s_0$ comes out serially (also from LSB to MSB). The control bit **K** determines whether addition or subtraction happens. Also available is a signal **F** that is 1 when a_0 and b_0 are input, and 0 otherwise. (**F** is not shown in the diagram, but you can use it.)

The output **V** will indicate that overflow has occurred, and is usually calculated as the XOR of the carry-in and the carry-out of the (a_{n-1}, b_{n-1}) calculation in the Full Adder. However, in this serialized design **V must depend only on the current state of the FSM.**



1. **(12 points)** Complete the circuit above by writing the missing signals in the 5 boxes using **0, 1, F or any other signal labeled in the diagram**. Complemented signals are **not** available, unless they are already shown in the diagram. You may **not** use additional wires or gates. Remember, the output **V must depend only on the current state of the FSM.**

2. **(4 points)** When does **V** indicate whether overflow has occurred **compared to when s_{n-1} comes out of its flip-flop?** Circle the correct answer.

   2 cycles earlier   1 cycle earlier   [✓ same cycle]   1 cycle later   2 cycles later

3. **(7 points)** Setting n=10, calculate the costs of the following circuits using the component prices shown on the right.

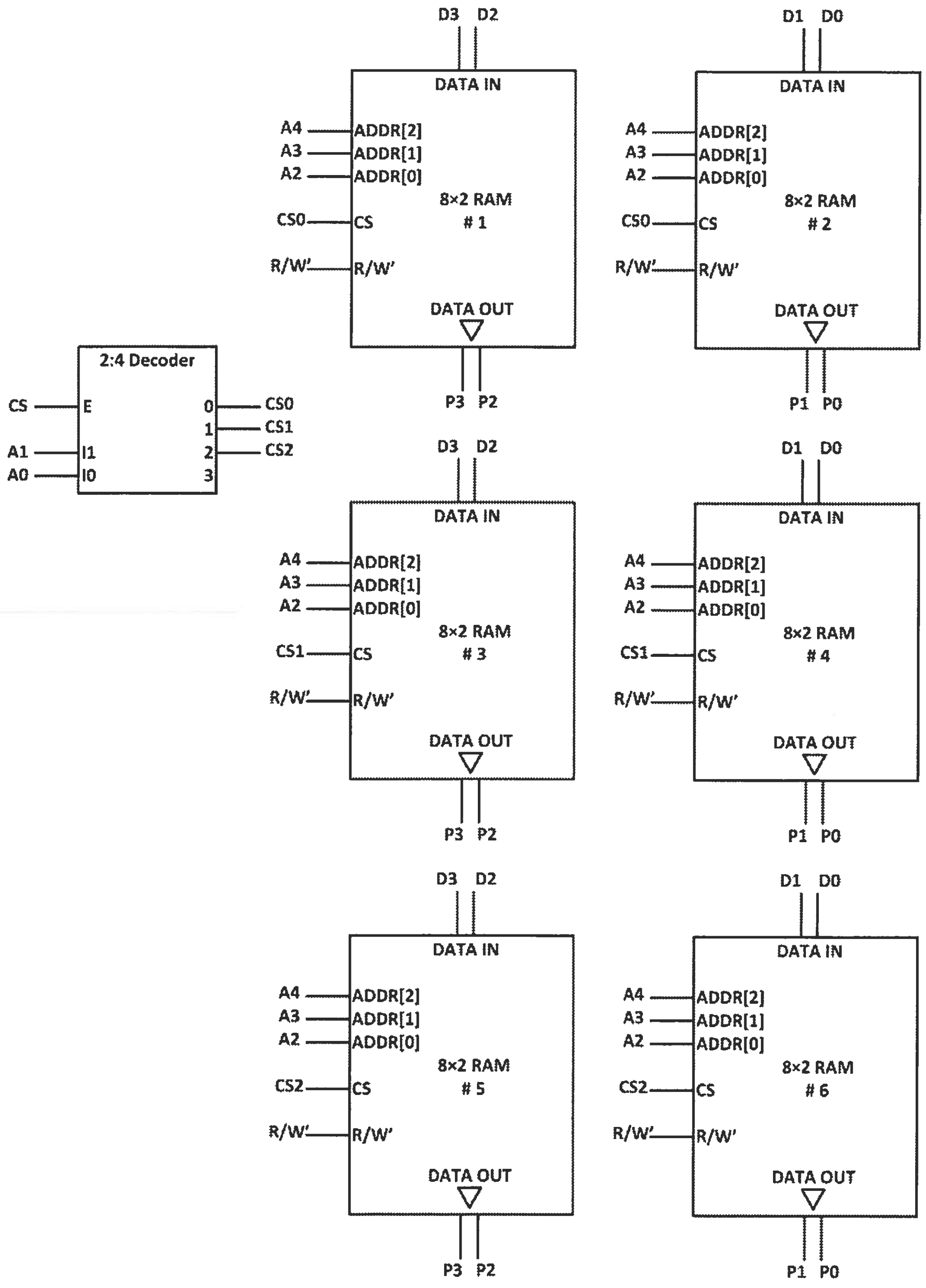
   Cost of the **serialized** 10-bit 2's complement adder/subtractor circuit with overflow detection (shown above): 81 cents

   Cost of the corresponding **bit-sliced** 10-bit 2's complement adder/subtractor circuit with overflow detection: 104 cents

Component	Price (cents)
NOT	1
2-input NAND	2
2-input XOR	4
Full adder	6
Flip-flop	20

Problem 4 (24 points): Memory

The ECE 120 instructors invited Doctor Strange to assist with the exam, and he obliged with the memory design presented below. Unfortunately for students, the instructors realized too late that he did not learn ECE 120 in this dimension, but in some other alternate dimension. Doctor Strange designed a **24×4 RAM**. His design, even though it works fine, is a bit... *strange*!



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

Problem 4 (24 points): Memory (continued)

1. **(12 points)** A student attempts to write (CS=1 and R/W'=0) into the 24x4 RAM with data $D_3D_2D_1D_0$ = 1011 at address $A_4A_3A_2A_1A_0$ = 01110. Indicate in the table below which of the 8x2 RAMs are accessed by **circling YES or NO for each RAM**. If an 8x2 RAM is accessed, also write down the 2-bit Data-In and the 3-bit address ADDR[2:0] for that 8x2 RAM.

RAM #1 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________	RAM #2 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________
RAM #3 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________	RAM #4 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________
RAM #5 Accessed? [✓ YES] / NO Data-In= 10 ADDR[2:0]= 011	RAM #6 Accessed? [✓ YES] / NO Data-In= 11 ADDR[2:0]= 011

2. **(12 points)** Another student now attempts to write (CS=1 and R/W'=0) into the 24x4 RAM with data $D_3D_2D_1D_0$ = 0100 at address $A_4A_3A_2A_1A_0$ = 01011. Indicate in the table below which of the 8x2 RAMs are accessed by **circling YES or NO for each RAM**. If an 8x2 RAM is accessed, also write down the 2-bit Data-In and the 3-bit address ADDR[2:0] for that 8x2 RAM.

RAM #1 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________	RAM #2 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________
RAM #3 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________	RAM #4 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________
RAM #5 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________	RAM #6 Accessed? YES / [✓ NO] Data-In= _____ ADDR[2:0]= _________

Problem 5 (22 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	x8888	R4	xCCCC	PC	x3710
R1	x9999	R5	xDDDD	IR	x993F
R2	xAAAA	R6	xEEEE	MAR	x370F
R3	xBBBB	R7	xFFFF	MDR	x993F

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
x370F	1001 100 100 111111	R4 ← NOT(R4)
x3710	0101 101 000 000 001	R5 ← AND(R0,R1)
x3711	0000 100 000000001	BRn #1
x3712	0001 110 010 1 00001	R6 ← R2+#1
x3713	1010 111 0 1000 0001	R7←M[M[PC+x0081]]
	...	...
x3793	0011 0111 1001 0100	(data: x3794)
x3794	0011 0111 1001 0110	(data: x3796)
x3795	0011 0111 1001 0011	(data: x3793)
x3796	0011 0111 1001 0101	(data: x3795)

Here is the question:

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

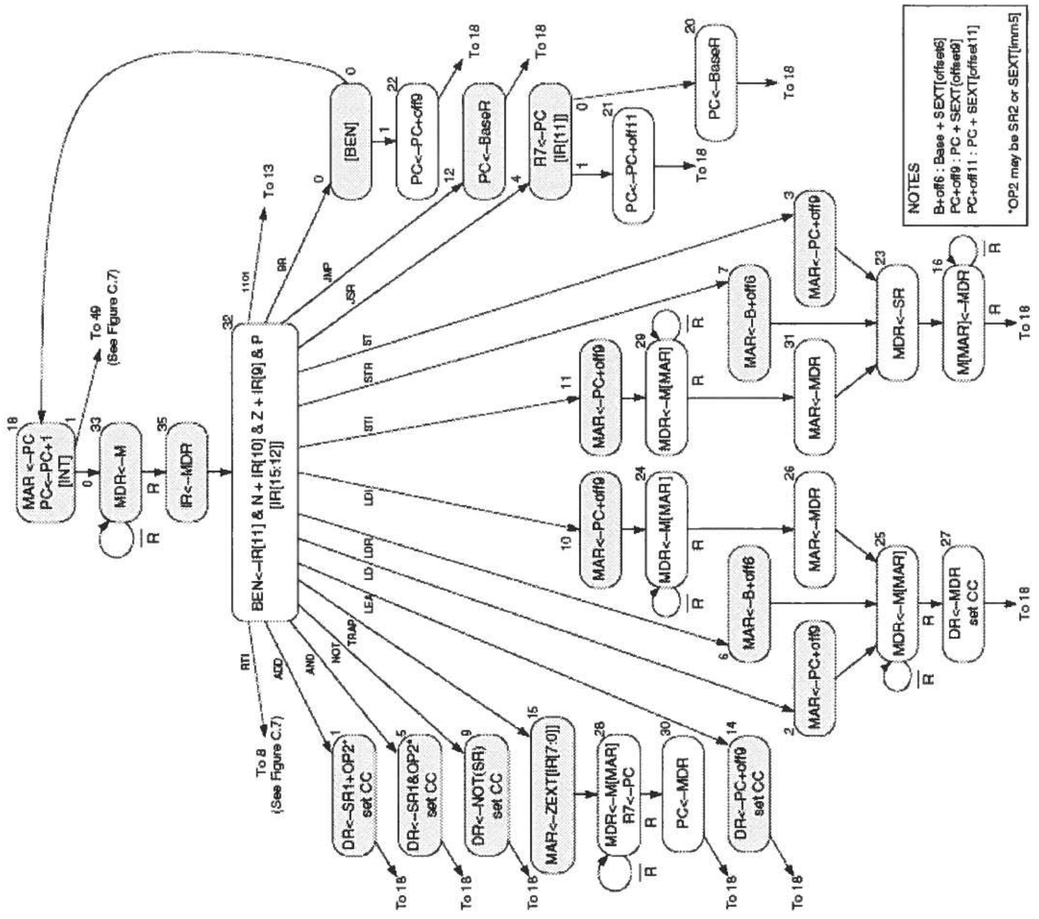
1. **(7 points)** Write a complete list of the sequence of values taken by the MAR register as the LC-3 processes these instructions. Use only as many lines as are necessary.

#1:	**x370F** (initial value)	#5:	x3795
#2:	x3710	#6:	x3793
#3:	x3711	#7:	
#4:	x3713	#8:	

2. **(15 points)** Complete the tables below with the **FINAL** values of each register and memory location (after processing the three instructions). **To receive credit, you must write your answers in hexadecimal**.

R4	xCCCC	PC	x3714
R5	x8888	IR	xAE81
R6	xEEEE	MDR	x3794
R7	x3794		

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)]

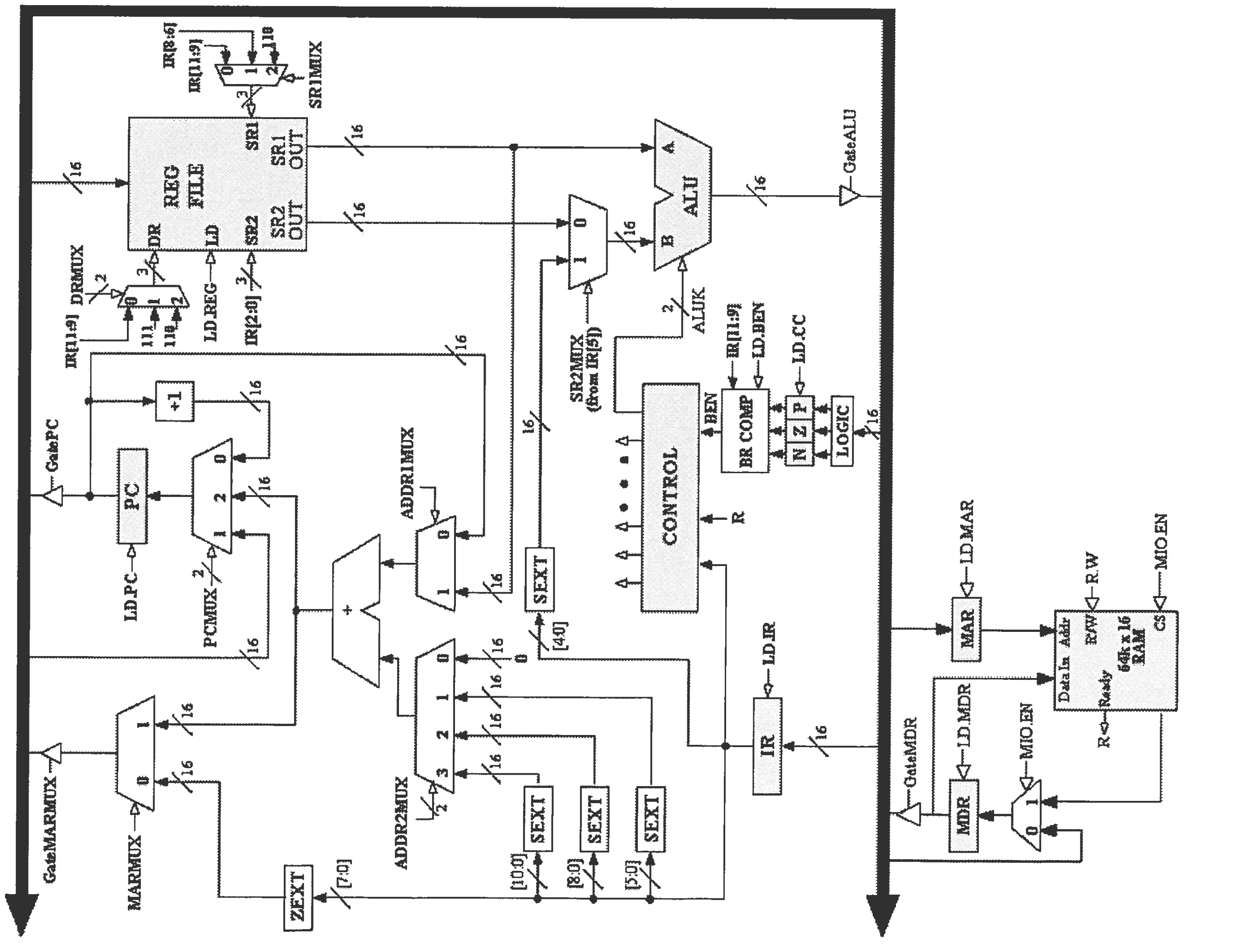
LD	0010 | DR | PCoffset9	LD DR, *PCoffset9*	DR ← M[PC + SEXT(PCoffset9)], Setcc
LDI	1010 | DR | PCoffset9	LDI DR, *PCoffset9*	DR ← M[M[PC + SEXT(PCoffset9)]], Setcc
LDR	0110 | DR | BaseR | offset6	LDR DR, BaseR, *offset6*	DR ← M[BaseR + SEXT(offset6)], Setcc
LEA	1110 | DR | PCoffset9	LEA DR, *PCoffset9*	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



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