ECE 120: Introduction to Computing

Fall 2023 – Final Exam

Friday, December 8th, 2023

Answer KEY

- Ensure that your exam booklet has 19 pages.
- **Write your NAME and UIN** clearly in the boxes below.
- Do not tear the exam booklet apart. You can only detach the last page for scratch work if needed.
- This is a closed book/notes exam. You may use a calculator.
- You are allowed **one handwritten sheet** of notes (both sides). Write your name on the cheat sheet. The cheat sheet will be collected at the end of your exam.
- Absolutely no interaction between students is allowed.
- Indicate any assumptions that you make.
- **The questions are not weighted equally**. Budget your time accordingly.
- Show your work and write legibly. Solutions in i**llegible handwriting will be graded as incorrect.**
- Write your **UIN (9-digit #) in the provided space on each page.**

NAME _____	UIN _____

Problem 1	10 points	_____
Problem 2	15 points	_____
Problem 3	15 points	_____
Problem 4	15 points	_____
Problem 5	26 points	_____
Problem 6	08 points	_____
Problem 7	11 points	_____
Total	100 points	_____

Problem 1 (10 points): Binary Representation and Operations

1. **(3 points) a)** What is the minimum number of bits required to encode a character set consisting of 129 symbols?

Answer: _____

b) How many additional symbols, if any, can be encoded with your answer to part (a)?

Answer: _____

2. **(4 points)** The **7-bit** pattern x74 represents what value in each of the following representations?

ASCII: _____

Unsigned (write answer in decimal): _____

2's complement (write answer in decimal): _____

3. **(3 points)** Prove that the set of Boolean functions {OR, NOT} is logically complete.

UIN_____

1. **(3 points) a)** What is the minimum number of bits required to encode a character set consisting of 129 symbols?

(+1) Answer: 8 bits

b) How many additional symbols, if any, can be encoded with your answer to part (a)?

(+2) Answer: 127 symbols

2. **(4 points)** The **7-bit** pattern x74 represents what value in each of the following representations?

(+1) ASCII: t (lower case)

(+1) Unsigned (write answer in decimal): 116

(+2) 2's complement (write answer in decimal): -12

3. **(3 points)** Prove that the set of Boolean functions {OR, NOT} is logically complete.

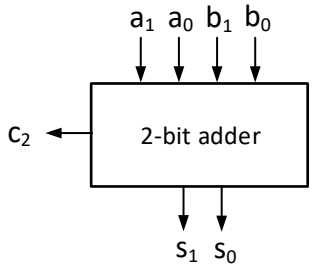
(+1) ① An n-bit NOR gate can be built with an n-bit OR gate followed by a NOT gate.

(+2) ② Since {NOR} is logically complete, {OR, NOT} is also logically complete.

Note: If statement in ② is not complete, (-1)

Problem 2 (15 points): Combinational Logic

1. Shown on the right is a combinational circuit that adds two 2-bit numbers, a_1a_0 and b_1b_0, and outputs the sum bits s_1s_0 and the carry bit c_2.



a. (4 points) Fill in the K-map for c_2 below.

c_2

a_1a_0 \ b_1b_0	00	01	11	10
00				
01				
11				
10				

s_0

a_1a_0 \ b_1b_0	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

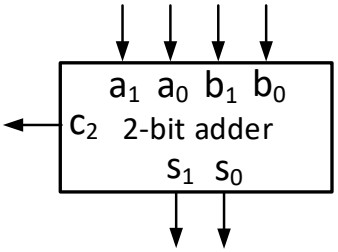
b. (3 points) Based on the K-map to the right above, write a Boolean expression for s_0 in ***POS*** form. **Show** corresponding loops on the K-map.

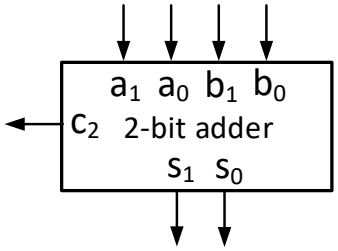
s_0 = _____

c. (3 points) Implement the circuit to calculate s_0 as a **two-level network** using only one type of gate. You can assume that inverted inputs are available.

Problem 2, continued

2. (5 points) Using two copies of the combinational circuit from Part 1 as building blocks and as few additional gates as possible, design a combinational circuit that subtracts two 2-bit 2's complement numbers, p_1p_0 and q_1q_0. The circuit should output the difference bits $d_1d_0 = p_1p_0 - q_1q_0$





a. (4 points) Fill in the K-map for c_2 below.

c_2

a_1a_0 \ b_1b_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	1

(-1) For each incorrect row

s_0

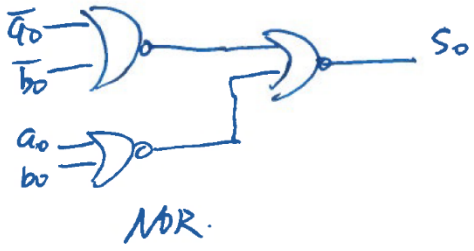
a_1a_0 \ b_1b_0	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

s_0

a_1a_0 \ b_1b_0	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

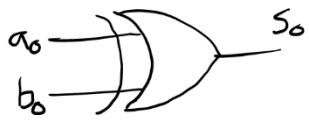
(+2) For expression (+1) For k-map.

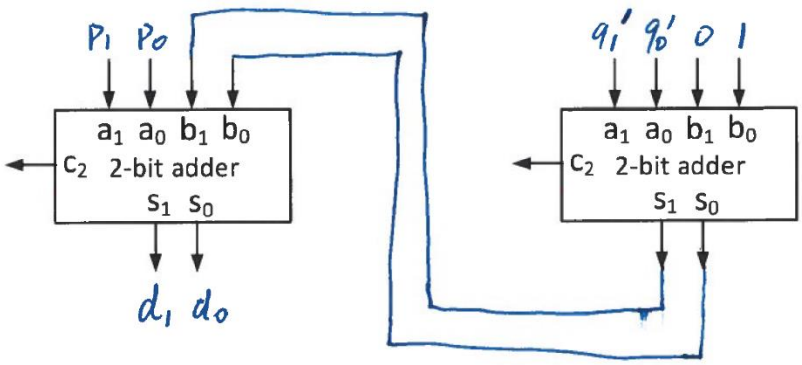
$s_0 = (\bar{a}_0 + \bar{b}_0) \cdot (a_0 + b_0)$



(+1) for NOR.
(+1) for 2-level
(+1) for labels

XOR Implementation is also correct.





(You should not need to write below this line.)

Rubric:
(+2) For $-(q_1 q_0)$
(+2) For $(P_1 P_0) + (-q_1 q_0)$
(+1) For labels (d_1, d_0, etc)

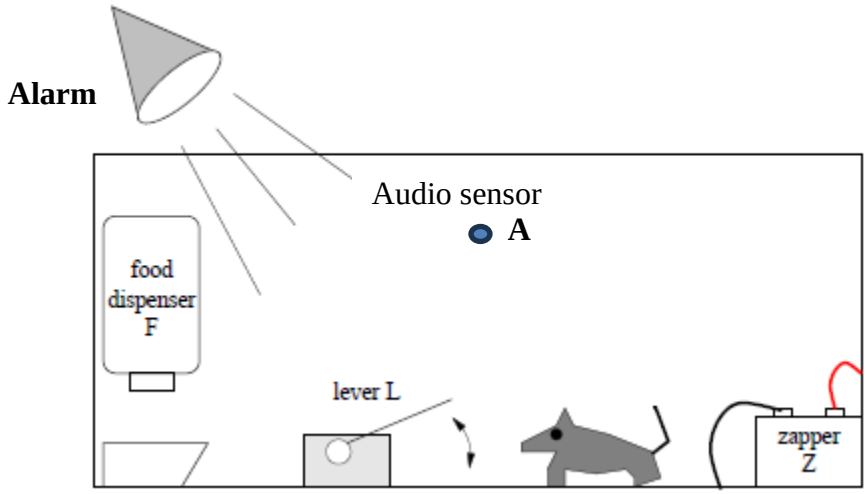
NOTE: $P_1 P_0$ and $q_1' q_0'$ could be switched but $q_1' q_0'$ has to go with -1

Problem 3: FSM Design Problem (15 points)

Your part-time employer Professor Zapper from Psychology asks you to design the logic for an experiment to study the memory of rats. The experimental setup is shown below. In each "cycle," the experimenter turns the alarm on or off. Your system receives a signal from the audio sensor A: when the alarm is on, A = 1. In the same "cycle" (long cycles), the rat responds by either depressing the lever L (in which case L = 1) or not depressing the lever (L = 0).

Professor Zapper wants the rats to follow a protocol. In the first and second cycles, the rat should not match the lever with the alarm. In other words, the rat should press the lever when the alarm is off, and not press the lever when the alarm is on. In the third cycle, the rat should match the alarm: press the lever when the alarm is on, and don't press the lever when the alarm is off.

If the rat succeeds in this endeavor, it should receive food (set F = 1 for a cycle). If it fails, it should immediately be zapped for a cycle (set Z = 1 for a cycle). After the penalty/reward cycle, start over.



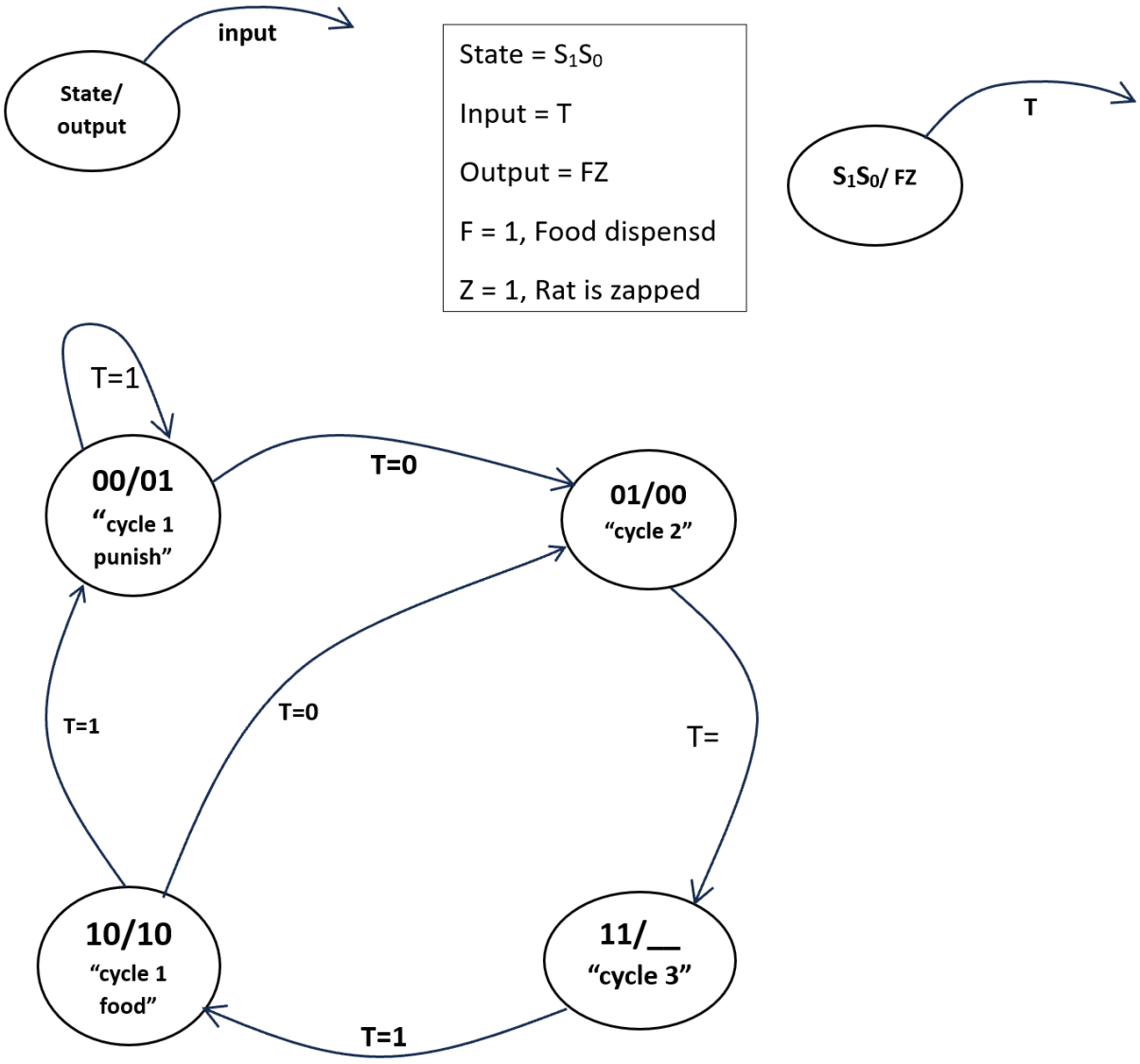
Designing the FSM:

We can make one crucial observation to simplify the design. We can introduce a variable T such that when rat's action is in accordance with the alarm, T=1. This is the case when

A	L	T
0	0	1
0	1	0
1	0	0
1	1	1

Thus, T = NOT (A XOR L)

1. **(4 points)** Based on this description, design a finite state machine to implement the desired experimental protocol. In particular, draw a complete state transition diagram labeled with input, internal state bits, and outputs. For your convenience, the partial state diagram is given to you. Complete the state diagram showing necessary transitions, inputs, and output. Hint: from every state, there will be two possible transitions.



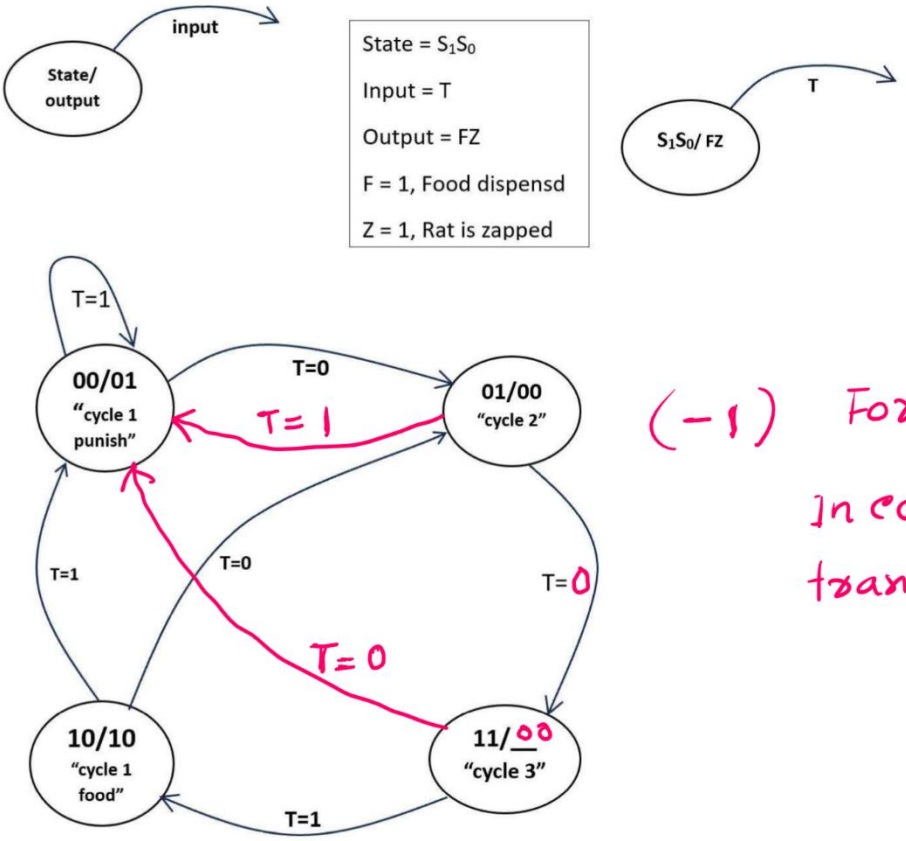
A side effect of the above implementation is that when the experiment starts, the rat is zapped at first

1. (**6 points**) Find simple logical expressions (minimal SOP) for next-state variables as well as functions mapping your internal state values to the outputs F and Z.

Next State Table

S_1	S_0	T	S_1^+	S_0^+	F	Z
0	0	0	**0**	**1**	**0**	**1**
0	1				**0**	**0**
1	1	1	**1**	**0**		

2. (**5 points**) Implement the FSM by drawing a logic schematic circuit diagram for your system.



(-1) For every incorrect transition/label

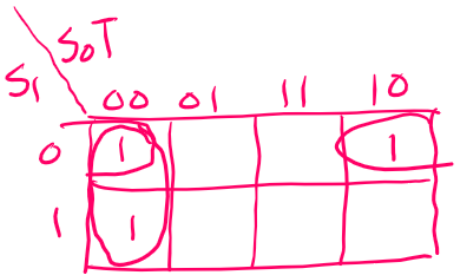
Next State Table

	S_1	S_0	T	S_1^+	S_0^+	F	Z
0	0	0	0	0	1	0	1
1	0	0	1	0	0	0	1
2	0	1	0	1	1	0	0
3	0	1	1	0	0	0	0
4	1	0	0	0	1	1	0
5	1	0	1	0	0	1	0
6	1	1	0	0	0	0	0
7	1	1	1	1	0	0	0

$F = S_1 \bar{S_0}$

$Z = \bar{S_1} \bar{S_0}$

$S_1^+ = \bar{S_1} S_0 \bar{T} + S_1 S_0 T$



$S_0^+ = \bar{S_0} \bar{T} + \bar{S_1} \bar{T}$

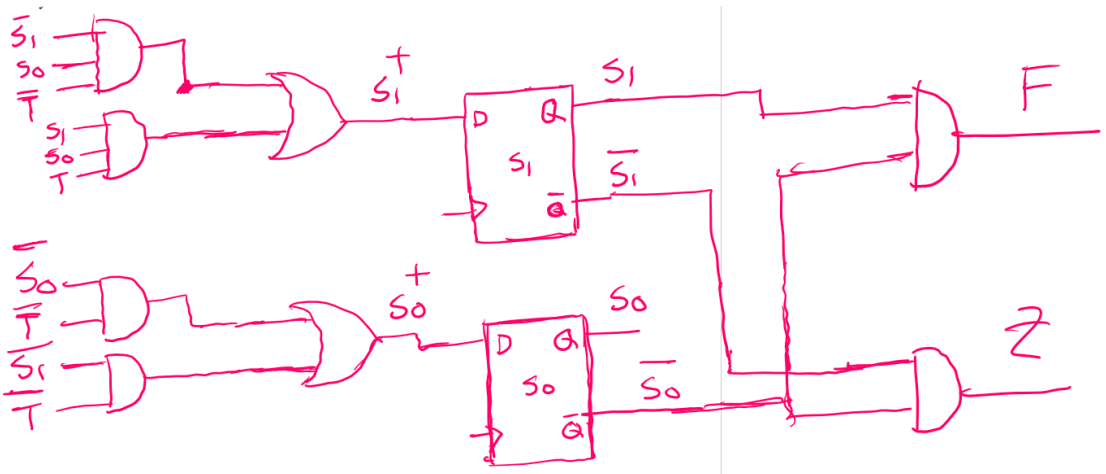
Truth Table : (-0.5) every incorrect ROW

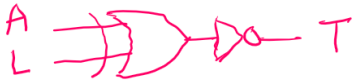
(-1) every incorrect S_1^+, S_0^+

(-0.75) every incorrect F, Z

7

* Note : S_1^+, S_0^+, F, Z should be based on student's table, which could be wrong/incorrect.





$T \equiv$ NOT (A XOR L)

Rubric:

(-1) for every incorrect implementation of S_1^+, S_0^+ F, Z, T

Problem 4 (15 points): LC-3 Interpretation and Assembly

1. The program below consists of LC-3 instructions and data.

```
x3000 0010 0110 0000 1011  LD R3, #11
x3001 0101 1011 0110 0000  _____________________________
x3002 0110 1100 1100 0000  _____________________________
x3003 0000 0110 0000 0011  _____________________________
x3004 1001 1001 1011 1111  _____________________________
x3005 0001 1101 0010 0001  _____________________________
x3006 0111 1100 1100 0000  _____________________________
x3007 0001 0110 1110 0001  _____________________________
x3008 0001 1011 0110 0001  ADD R5, R5, #1
x3009 0001 1101 0111 1000  _____________________________
x300A 0000 1001 1111 0111  _____________________________
x300B 1111 0000 0010 0101  _____________________________
x300C 1111 0000 0000 0000  FILL .xF000
```

a. **(9 points)** Decode the remaining instructions in the program above into LC-3 assembly language using the format shown in those already done for you. Note that all blanks correspond to instructions, not data.

b. **(4 points)** In **twenty words or less**, explain what the program does.

Problem 4, continued:

2. **(2 points)** The following program has exactly one error. State the nature of the error, in which pass the assembler identifies the error (first or second), and suggest how the program can be corrected.

```
    .ORIG x3000
    LEA R0,STRING
FIND
    LDR R1,R0,#0
    BRn DONE
    ADD R0,R0,#1
    BRnzp FIND
STRING .BLKW x1000
EOS    .FILL xFFFF
DONE HALT
    .END
```

Answer (use no more than 20 words)**:**

UIN_____

(-1) For every incorrect decoding
* Max (-9)

1. The program below consists of LC-3 instructions and data.

```
x3000 0010 0110 0000 1011  LD R3, #11
x3001 0101 1011 0110 0000  AND R5, R5, #0
x3002 0110 1100 1100 0000  LDR R6, R3, #0
x3003 0000 0110 0000 0011  BRzp #3
x3004 1001 1001 1011 1111  NOT R4, R6
x3005 0001 1101 0010 0001  ADD R6, R4, #1
x3006 0111 1100 1100 0000  STR R6, R3, #0
x3007 0001 0110 1110 0001  ADD R3, R3, #1
x3008 0001 1011 0110 0001  ADD R5, R5, #1
x3009 0001 1101 0111 1000  ADD R6, R5, #-8
x300A 0000 1001 1111 0111  BRn #-9
x300B 1111 0000 0010 0101  HALT
x300C 1111 0000 0000 0000  FILL .xF000
```

b. **(4 points)** In **twenty words or less**, explain what the program does.

Read an array of eight memory locations from xF000; invert the negative values and store them in their corresponding location

Problem 4, continued:

2. **(2 points)** The following program has exactly one error. State the nature of the error, in which pass the assembler identifies the error (first or second), and suggest how the program can be corrected.

```
    .ORIG x3000
    LEA R0,STRING
FIND
    LDR R1,R0,#0
    BRn DONE
    ADD R0,R0,#1
    BRnzp FIND
STRING .BLKW x1000
EOS    .FILL xFFFF
DONE HALT
    .END
```

Error

"If someone claims that this is the source of error", is also correct

Answer (use no more than 20 words):

Out of range errors. → PCoffset9 out of range

The assembler identifies the error in the second pass

Problem 5 (26 points)

In this problem we introduce a new instruction to the LC3 instruction set, called ADDM: Add to Memory.

ADDM DR, SR1, SR2

ADDM adds the content of the source register **SR2** and the content of the memory location whose address is in register **SR1**, and puts the result in register **DR**. ADDM has opcode 1101 and the RTL is:

DR ← M[SR1] + SR2, set CC

1. (**4 points**) Give the binary encoding of the instruction **ADDM R3, R4, R5** by filling in the missing bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										0	**0**	**0**			

2. (**2 points**) Statement: "If we set the 5th bit to 1 (IR[5]=1) when encoding the ADDM instruction, it won't work as expected."

   Is this statement True or False? Justify your choice with reference to SR2MUX.

   True, because _____

   False, because _____

3. (**8 points**) Give the sequence of 4 microinstructions that implement the ADDM instruction **after the decode state**. If needed, you may use R6 as a temporary register.

   Answer: _____

   _____

   _____

   _____

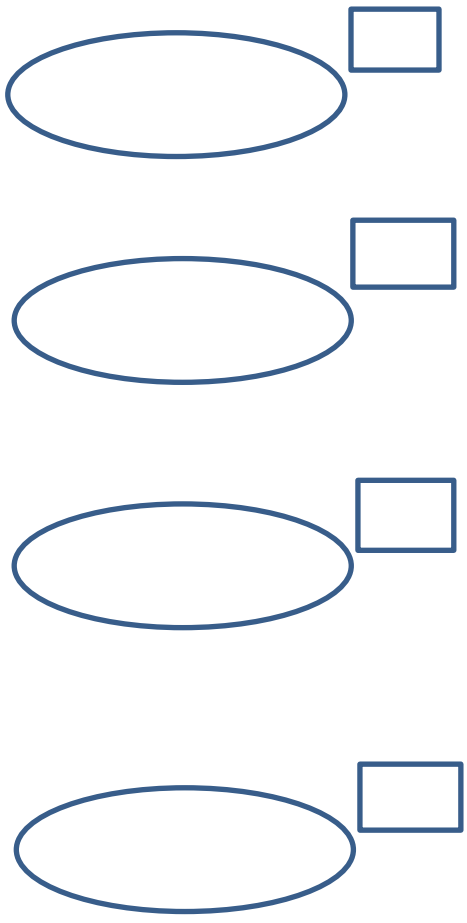
4. **(8 points)** Determine the control ROM microinstructions that implement the RTL statements from part (a). Complete the table below by filling in **0**, **1**, or **x** as appropriate. Specify ROM addresses in **decimal**. Note that your first state number is implied by the decode strategy used in the LC-3 microarchitecture. State numbers 51, 52, 53, and 54 are available as additional states for your use.

UIN_____

ROM address in decimal	IRD	COND(3) , COND0 is LSB	J(6), J0 is the LSB	LD.BEN	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	LC.CC	GateMARMUX	GateMDR	GateALU	GatePC	MARMUX	PCMUX(2)	ADDR1MUX	ADDR2MUX(2)	DRMUX(2)	SR1MUX(2)	ALUK(2)	MIO.EN	R.W
				Do not fill in datapath control word fields for these two microinstructions. But be sure to fill in the remaining two.																				

5. (**4 points**) Draw the state diagram for ADDM **after the decode state with the arc labels as needed, the RTLs mentioned in the states,** and include the state numbers in the boxes provided**.** State numbers 51, 52, 53 and 54 are available as additional states for your use. Hint: your answer should be consistent with the simplified Patt and Patel microsequencer circuit attached to this booklet.

Note: $51_{10} = 110011_2$, $52_{10} = 110100_2$, $53_{10} = 110101_2$, $54_{10} = 110110_2$



missing bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	1	1	1	0	0	0	0	0	1	0	1

opcode → +1
R_3 → +1
R_4 → +1
R_5 → +1

If the operands are correct, but misplaced (−1)

Is this statement True or False? Justify your choice with reference to SR2MUX.

True, because IR[5] must be zero such that SR2 (i.e. R5) is selected by the SR2MUX

False, because _____

3. (**8 points**) Give the sequence of 4 microinstructions that implement the ADDM instruction **after the decode state**. If needed, you may use R6 as a temporary register.

Answer: MAR ← R[IR[8..6]] ; MAR ← R4

MDR ← M[MAR]

R6 ← MDR

R[IR[11..9]] ← R6 + R[IR[2..0]] ; R3 ← R6 + R5

ROM address in decimal	IRD	COND(3), COND0 is LSB	J(6), J0 is the LSB	LD.BEN	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	LC.CC	GateMARMUX	GateMDR	GateALU	GatePC	MARMUX PCMUX(2)	ADDR1MUX	ADDR2MUX(2)	DRMUX(2)	SR1MUX(2)	ALUK(2)	MIO.EN	R.W
13	0	000	110100	0	1	0	0	0	0	0	0	0	1	0	x	xx	x	xx	xx	01	11	0	x
52	0	001	110100	0	0	1	0	0	0	0	0	0	0	0	x	xx	x	xx	xx	xx	xx	1	0
54	0	000	110011	Do not fill in datapath control word fields for these two microinstructions. But be sure to fill in the remaining two.																		
53	51	0	000	010010																		

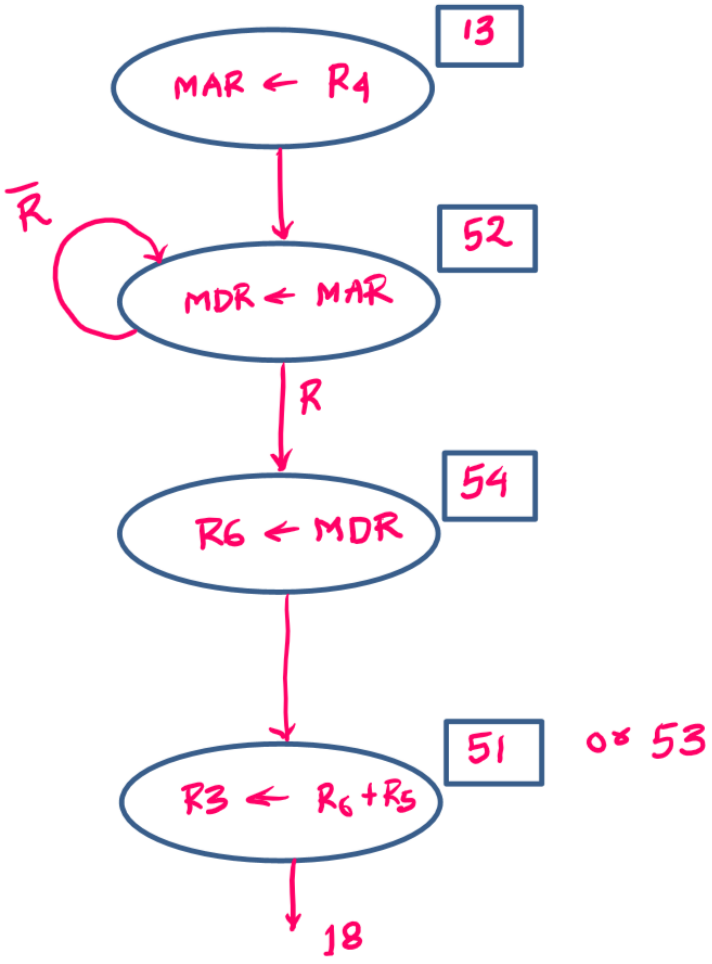
110101 (also correct)

(−1) For every incorrect group (i.e ROM address, IRD, COND(3), J(6), and control signal groups)

ROM address in decimal	IRD	COND(3), COND0 is LSB	J(6), J0 is the LSB	LD.BEN	LD.MAR	LD.MDR	LD.IR	LD.PC	LD.REG	LC.CC	GateMARMUX	GateMDR	GateALU	GatePC	MARMUX PCMUX(2)	ADDR1MUX	ADDR2MUX(2)	DRMUX(2)	SR1MUX(2)	ALUK(2)	MIO.EN	R.W
13	0	000	110100	0	1	0	0	0	0	0	0	0	1	0	0	1	xx	1	00	xx	01	xx	0	x
52	0	001	110100	0	0	1	0	0	0	0	0	0	0	0	x	x	x	xx	xx	xx	xx	xx	1	0
54	0	000	110011	Do not fill in datapath control word fields for these two microinstructions. But be sure to fill in the remaining two.																		
53	51	0	000	010010																		

110101

Also Correct



(-1) point for every incorrect state no.

Problem 6 (8 points): LC-3 Assembly Programming

Shown below is an incomplete LC-3 assembly program that computes the sum of the squares of a list of positive integers, stored in consecutive memory locations starting at address **x5000**. A zero or a negative number indicates the end of the list of integers. The result is saved to memory address **x6000**. **Write the missing lines of code**. You must **write only one instruction per missing line**. To receive credit, **you can only use registers R0, R1, R2, R3, R4, and no other register**.

LINE	PROGRAM
1	; Store program in location x3000
2	_____
3	; R0 stores the address of the list, 0x5000
4	LD R0, NUM_LIST
5	; Initialize the sum in R1 to zero
6	_____
7	SUM_LOOP ; Load current number from the list to R2
8	_____
9	; Exit if the number is negative or zero
10	BR____ END_LOOP
11	; Clear R3 to store the square result
12	AND R3, R3, #0
13	; Copy the current number to R4
14	ADD R4, R2, #0
15	MULT_LOOP ; The loop calculates the square of R4
16	_____
17	ADD R2, R2, #-1
18	; Repeat inner loop to calculate the square
19	BR____ _____
20	; Add the calculated square R3 to the sum
21	ADD R1, R1, R3
22	; Move to the next number in the list
23	ADD R0, R0, #1
24	BR SUM_LOOP
25	END_LOOP ; Store the sum at x6000
26	_____
27	HALT
28	; Memory address of the number list
29	NUM_LIST _____
30	; Memory address to store the result
31	RESULT .FILL x6000

UIN_____

	.END

(-1) for every Incorrect instruction

LINE	PROGRAM
1	; Store program in location x3000
2	**.ORIG x3000**
3	; R0 stores the address of the list, 0x5000
4	LD R0, NUM_LIST
5	; Initialize the sum in R1 to zero
6	**AND R1, R1, #0**
7	SUM_LOOP ; Load current number from the list to R2
8	**LDR R2, R0, #0**
9	; Exit if the number is negative or zero
10	**BRnz** END_LOOP
11	; Clear R3 to store the square result
12	AND R3, R3, #0
13	; Copy the current number to R4
14	ADD R4, R2, #0
15	MULT_LOOP ; The loop calculates the square of R4
16	**ADD R3, R3, R4**
17	ADD R2, R2, #-1
18	; Repeat inner loop to calculate the square
19	**BRp MULT_LOOP**
20	; Add the calculated square to the sum
21	ADD R1, R1, R3
22	; Move to the next number in the list
23	ADD R0, R0, #1
24	BR SUM_LOOP
25	END_LOOP ; Store the sum at x6000
26	**STI R1, RESULT**
27	HALT
28	; Memory address of the number list
29	NUM_LIST **.FILL x5000**
30	; Memory address to store the result
31	RESULT .FILL x6000
	.END

Problem 7 (11 points): LC-3 Assembly Analysis

You are given the following program written in LC-3 assembly language.

```
            .ORIG x3000
                  LEA   R1, INPUT
                  LEA   R2, MASK
                  LDR   R3, R2, #0
                  NOT   R2, R2
                  ADD   R2, R2, #1
                  AND   R4, R4, #0
OUTER_LOOP        LDR   R0, R1, #0
                  AND   R5, R5, #0
                  ADD   R5, R5, #15
INNER_LOOP        AND   R6, R0, R3
                  BRz   SKIP_ADD
                  ADD   R4, R4, #1
SKIP_ADD          ADD   R0, R0, R0
                  ADD   R5, R5, #-1
                  BRzp  INNER_LOOP
                  ADD   R1, R1, #1
                  ADD   R6, R1, R2
                  BRn   OUTER_LOOP
ST                R4, RESULT
                  HALT
INPUT       .FILL x1248
            .FILL x2814
            .FILL x4821
MASK        .FILL x8000
RESULT      .BLKW #1
            .END
```

1. **(5 points)** For each label in the table below, fill in its memory address and the number of times the LC-3 reads from that memory address during the execution of the entire program. **Count only memory reads, not writes.**

Label	Memory Address (4-digit hexadecimal value)	Number of Memory Reads (decimal number)
OUTER_LOOP		
INNER_LOOP		
SKIP_ADD		
INPUT		
MASK		

2. **(6 points)** Write the **4-digit hexadecimal values** held in the following registers/memory when the program halts. Assume that HALT does not modify any registers.

R1 = _____ R2 = _____ R4 = _____

(5 points) For each label in the table below, fill in its memory address and the number of times the LC-3 reads from that memory address during the execution of the entire program. **Count only memory reads, not writes.**

Label	Memory Address (4-digit hexadecimal value)	Number of Memory Reads (decimal number)
OUTER_LOOP	x3006	3
INNER_LOOP	x3009	48
SKIP_ADD	x300c	48
INPUT	x3014	1
MASK	x3017	1

(-0.5) For every incorrect memory address

(-0.5) For every incorrect no. of memory read.

M[x300C] = _____ M[x300E] = _____M[x3018] = _____

(6 points) Write the **4-digit hexadecimal values** held in the following registers/memory when the program halts. Assume that HALT does not modify any registers.

R1 = __**x3017**____ R2 = ___**xCFE9**____ R4 = ___**x000C**_____

M[x300C] = _**x1000**_ M[x300E] = __**x07FA**___ M[x3018]=___**x000C**___

(-1) For every incorrect answer