

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

ECE 306 Fall 2025  
Instructor: Yale N. Patt  
TAs: Luke Mason, Evan Lai, Madeleine Dreher  
Exam 1  
October 8, 2025

Name: \_\_\_\_\_

Problem 1 (20 points): \_\_\_\_\_

Problem 2 (15 points): \_\_\_\_\_

Problem 3 (15 points): \_\_\_\_\_

Problem 4 (25 points): \_\_\_\_\_

Problem 5 (25 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.

Please read the following sentence, and if you agree, sign where requested:  
I have not given nor received any unauthorized help on this exam.

Signature: \_\_\_\_\_

**GOOD LUCK!**

Name: \_\_\_\_\_

**Question 1 (20 pts):**

Answer the following questions. If you leave a question unanswered, you will receive 1 point out of the 5.

**Part A (5 pts):** Before the code below executes, memory location x3050 contains the 2's complement integer x. What does memory location x3050 contain after the following code executes?

```
x3000: 0010 011 001001111
x3001: 0000 011 000000011
x3002: 1001 011 011 111111
x3003: 0001 011 011 1 00001
x3004: 0011 011 001001011
x3005: 1111 0000 0010 0101
```

**Part B (5 pts):** The LC-3 has 3 load instructions (opcodes 0010, 0110, and 1010), each with a different addressing mode for determining the location of the data to be loaded into the register. What benefit does the instruction with opcode 0110 provide over the instruction with opcode 0010.

**Part C (5 pts):** The LC-3 has an instruction (opcode 1111) that allows the user program to invoke the operating system to carry out some work that is beyond the knowledge of the user programmer. Most ISAs have that feature. It is usually called **System Call** because the operating system is being called to execute code on behalf of the user program. What is the maximum number of system calls we could provide if we wanted to for the LC-3?

**THE PROBLEM CONTINUES ON THE NEXT PAGE**

Name: \_\_\_\_\_

**Part D (5 pts):** We know that a function  $f$  is logically complete if we can implement any truth table, regardless how many input variables it has, if we have a sufficient number of  $f$  gates. We know that if we have a sufficient number of AND gates, OR gates, and NOT gates, we can implement any function represented by its truth table. Prove that a NAND gate is by itself logically complete.

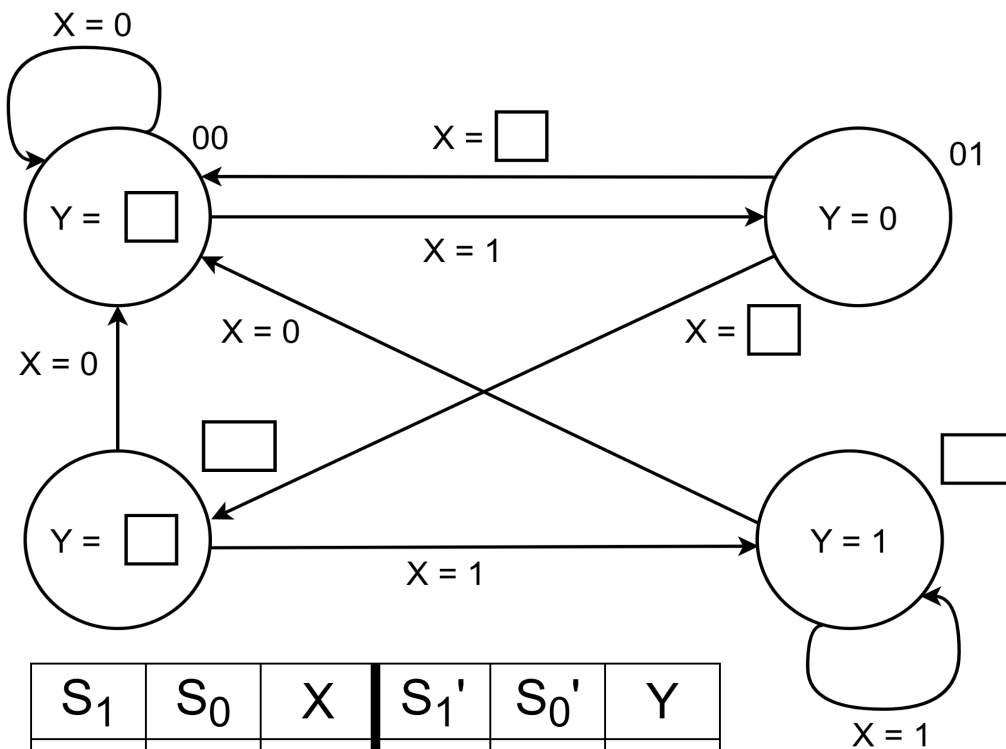
NOTE: Your proof must be explicit, that is, each step in your proof must be specific and not rely on the reader to figure out how to do anything. No points if your proof contains statements that rely on the reader to know how to do something.

Name: \_\_\_\_\_

### Question 2 (15 pts):

You have been given an incomplete Moore state machine. Remember, the output(s) of a Moore machine are determined by only the current state. The state numbers are encoded as 2-bit values represented by  $S_1S_0$ . For instance,  $S_1=0$  and  $S_0=1$  correspond to the top right state, state 01.

**Your job:** Complete the partial state machine and truth table below by filling the blanks.

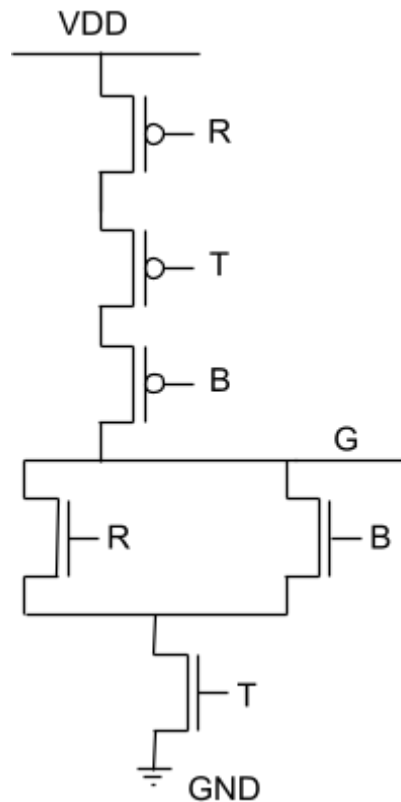


| $S_1$ | $S_0$ | $X$ | $S_1'$ | $S_0'$ | $Y$ |
|-------|-------|-----|--------|--------|-----|
| 0     | 0     | 0   |        |        | 0   |
| 0     | 0     | 1   | 0      | 1      |     |
| 0     | 1     | 0   | 0      | 0      | 0   |
| 0     | 1     | 1   | 1      | 0      |     |
| 1     | 0     | 0   |        |        | 0   |
| 1     | 0     | 1   | 1      | 1      | 0   |
| 1     | 1     | 0   | 0      | 0      |     |
| 1     | 1     | 1   |        |        |     |

Name: \_\_\_\_\_

**Question 3 (15 pts):**

Aggie has attempted to construct a CMOS transistor circuit. His goal was to determine if the garden sprinkler should turn on ( $G = 1$ ). If it rained today ( $R = 1$ ) or the water bill was high last month ( $B = 1$ ), the sprinkler should **not** turn on. Otherwise, it should turn on if it's 9pm ( $T = 1$ ). Unfortunately, Aggie doesn't know how to design a circuit and created what you see below:



**Part A (3 pts):** Write the intended logic equation for the output  $G$  in terms of  $R$ ,  $B$ , and  $T$

|  |
|--|
|  |
|--|

**Part B (6 pts):** Write down two specific reasons the design does not work. Explain briefly.

**Reason 1:**

**Reason 2:**

**THE PROBLEM CONTINUES ON THE NEXT PAGE**

Name: \_\_\_\_\_

**Part C (6 pts):** Help Aggie and sketch the correct transistor level circuit:

Name: \_\_\_\_\_

**Problem 4 (25 points):**

Aggie has returned to you for help debugging a new program. The program is trying to perform an integer divide, with the value at x300A being the dividend, the value at x300B being the divisor, and R2 being the quotient. In other words,  $R2 = \frac{M[x300A]}{M[x300B]}$ .

Note that in an integer divide, the remainder is thrown away (i.e  $10/3 = 3$ ).

A snapshot of the PC and some memory locations before the program is executed is shown below. **There is exactly one bug.** The Comments column will not be graded.

|    |       |
|----|-------|
| PC | x3000 |
|----|-------|

| Address      | Contents                               | Comments                            |
|--------------|--|-------------------------------------|
| x3000        | 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1        |                                     |
| x3001        | 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1        |                                     |
| x3002        | 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0        |                                     |
| x3003        | 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1        |                                     |
| x3004        | 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1        |                                     |
| x3005        | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1        |                                     |
| x3006        | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1        |                                     |
| x3007        | 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 1        |                                     |
| x3008        | 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0        |                                     |
| x3009        | 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1        |                                     |
| x300A        | 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 1        | Contains dividend x3025             |
| x300B        | 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0        | Contains divisor x1000              |
| x300C<br>... | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0<br>... | The rest of the locations are x0000 |

**Part A (5 points):** After 1 instruction is executed, the PC is x3001 and R0 has been loaded with M[x300A]. What is the PC after 10 instructions are executed?

PC:

**THE PROBLEM CONTINUES ON THE NEXT PAGE**

Name: \_\_\_\_\_

**Part B (10 points):** As stated above, there is exactly one bug in the program. Describe the bug (in 15 words or fewer).

**Part C (10 points):** Despite the bug, the program shown still completes execution correctly. Explain why.



Name: \_\_\_\_\_

### Question 5 (25 pts):

Write a program to implement  $f(x) = 7 * x$ . The location x3050 contains x, and you will store the result,  $(7 * x)$ , into location x3051. To accomplish this, you are given a list of allowed instructions. **Not every instruction will be used, and each instruction can only be used once.**

**Part A (20 pts):** Fill in the program on the right, using instructions from the instruction list on the left. PC = x3000 to start.

*Hint:  $7 * x$  can be accomplished by calculating  $1 * x + 2 * x + 4 * x$ .*

| Allowed Instruction List |
|--------------------------|
| 0010 000 001001110       |
| 1111 0000 00100101       |
| 0001 001 000 0 00 000    |
| 0001 001 001 0 00 000    |
| 0001 001 001 0 00 001    |
| 0001 010 010 1 11111     |
| 0000 100 111111100       |
| 0000 101 111111100       |
| 0011 001 001001000       |
| 0101 001 001 1 00000     |
| 0101 010 010 1 00000     |

| Program |                       |
|---------|-----------------------|
| x3000   |                       |
| x3001   |                       |
| x3002   | 0001 010 010 1 00011  |
| x3003   |                       |
| x3004   |                       |
| x3005   | 0001 000 000 0 00 000 |
| x3006   |                       |
| x3007   |                       |
| x3008   |                       |
| x3009   |                       |

**Part B (5 pts):** Assume any state in the LC-3 FSM that accesses memory takes 5 clock cycles. After 30 clock cycles of this program's execution, what are the contents of the PC? (Refer to the LC-3 FSM at the end of this exam)

Name: \_\_\_\_\_

**This page is left blank intentionally. Feel free to use it for scratch work.  
You may tear the page off if you wish.  
Nothing on this page will be considered for grading.**