Name: ____________________________

Problem 1 (20 points): 20
Problem 2 (15 points): 15
Problem 3 (15 points): 15
Problem 4 (15 points): 15
Problem 5 (15 points): 15
Problem 6 (15 points): 15
Problem 7 (15 points): 15
Problem 8 (15 points): 15
Total (125 points): 125

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 written legibly on each sheet of the exam.

I will not cheat on this exam.

Signature

GOOD LUCK!
Problem 1 (20 points):

Part a (5 points): The following program is assembled. Complete the symbol table entry for THIS.

Note: Descriptions for pseudo-ops such as .ORIG, .END, .FILL, .BLK, and .STRINGZ can be found in the auxiliary packet.

```
.ORIG x3000
LD R0,START
TRAP x21
HALT
START .STRINGZ "Let's go!"
THIS .STRINGZ "and THAT"
.END
```

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>x3003</td>
</tr>
<tr>
<td>THIS</td>
<td>x3000</td>
</tr>
</tbody>
</table>

Part b (5 points): The finite state machine shown below has one external input and two external outputs. The output for each state is shown inside the state. The machine is initially in state A and receives an input string of n 1's followed by a single 0. What does a final output of 00 signify about n?

![Finite State Machine Diagram]

**Answer:**

n is even.

Part c (5 points): During the execution of the following program, how many times does the instruction labeled AA get executed?

```
.ORIG x3000
AND R1, R1, #0
ADD R3, R1, #2
AA
ADD R3, R3, R3
B3AP AA
HALT
.END
```

**Answer:** 15
Part d (5 points): What does the Assembler produce when presented with the assembly language instruction LDR R3, R2, #–35. (An instruction layout is provided for your use, if you need it.)

It produces an error message saying that #–35 cannot be represented in 5 bits.
Name: Solution

Problem 2 (15 points):

In this problem you are asked to design a logic circuit that computes \( n^m \), for 2-bit unsigned integers \( n, m \).

\[
\begin{array}{c}
\overset{2}{n} & \rightarrow & k \quad m \rightarrow \overset{2}{m} \\
\text{(expressed as } r_{11} r_{10} \ldots r_0 \text{)}
\end{array}
\]

Part a: What is the maximum value of \( n? \) \( 3 \) of \( m? \) \( 3 \)

Part b: What is the maximum possible value for \( n^m \)? \( 27 \)

Part c: In the block diagram above, what is the value of \( k? \) \( 5 \)

Part d: Construct the truth table for the logic function that computes \( n^m \). (Assume \( 0^0 = 1 \))

<table>
<thead>
<tr>
<th>( r_{11} )</th>
<th>( r_{10} )</th>
<th>( r_0 )</th>
<th>( r_1 )</th>
<th>( r_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Problem 2 (continued):

**Part e:** Construct the logic circuit for the output $r_3$. 

![Logic Circuit Diagram]
Problem 3 (15 points):

Figure 3.22 in the textbook shows a $2^2$ by 3 memory. In the figure below, we have replaced the one-bit WE signal with a 2-bit signal we call $X$ plus the additional logic that is necessary to perform the functions required by the four values of $X$. This figure is also included in your auxiliary packet.
Name: **Solution**

Problem 3 continued:

**Part a:** In 15 words or fewer each, what function is performed for each of the four values of X.

<table>
<thead>
<tr>
<th>X=00</th>
<th>X=01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enables writing to bit 0</td>
<td>Enables writing to bit 2</td>
</tr>
<tr>
<td>of the value stored at A.</td>
<td>of the value stored at A.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X=10</th>
<th>X=11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enables writing to bit 2</td>
<td>Enables writing to all bits</td>
</tr>
<tr>
<td>of the value stored at A.</td>
<td>of the value stored at A.</td>
</tr>
</tbody>
</table>

**Part b:** What important situation can no longer occur if we replace WE by the 2-bit signal X? (In 15 words or fewer, please.)

**Answer:**

You can no longer do a pure read from memory.
Solution

Problem 4 (15 points):

What does the following program do? Explain in the box provided below.

```
.CRIG x3000
AND R1, R1, #0
AND R2, R2, #0
LDI R0, IN_ADDR
L0 BRz END
BRp L1
ADD R1, R1, #1
L1 ADD R0, R0, R0
BRnez LO
END ADD R1, R1, #8
BRnz L2
ADD R2, R2, #1
L2 ST R2, OUTPUT
HALT
IN_ADDR .FILL x4000
OUTPUT .BLKW x1
. END
```

Answer:

If the value at location x4000 has a majority (> 8) of its bits set, a 1 will be stored at OUTPUT, otherwise a 0 will be stored at OUTPUT.
Name: Solution

Problem 5 (15 points):

Shown below is a snapshot of the 8 registers and the PC of the LC-3 at two instances of time: (1) before the instruction at location x4000 is fetched, and (2) after the instruction at location x4001 has completed. Note that one piece of data is missing.

<table>
<thead>
<tr>
<th></th>
<th>Before Instruction at x4000</th>
<th>After Instruction at x4001</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>x4000</td>
<td>x4010</td>
</tr>
<tr>
<td>R0</td>
<td>x0000</td>
<td>x0000</td>
</tr>
<tr>
<td>R1</td>
<td>x1000</td>
<td>x1000</td>
</tr>
<tr>
<td>R2</td>
<td>x2000</td>
<td>x2000</td>
</tr>
<tr>
<td>R3</td>
<td>x3000</td>
<td>x3000</td>
</tr>
<tr>
<td>R4</td>
<td>x4000</td>
<td>x020E</td>
</tr>
<tr>
<td>R5</td>
<td>x5000</td>
<td>x5000</td>
</tr>
<tr>
<td>R6</td>
<td>x6000</td>
<td>x6000</td>
</tr>
<tr>
<td>R7</td>
<td>x7000</td>
<td>x7000</td>
</tr>
</tbody>
</table>

Part a: Complete the following two entries:

Instruction at x4000:

```
15
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
```

Instruction at x4001:

```
15
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
```

Part b: Supply the missing data element in the table (shown above).

Note: More than one correct set of answers is possible for this problem. Any correct set of answers will do.

Note: There are 3 other solutions since the N and Z bits in the branch instruction can be either 0 or 1.
Name: 

Solution

Problem 6 (15 points):

It has been suggested that we use the unimplemented LC-3 opcode to add push and pop to the LC-3 ISA, as follows:

**PUSH:** 1110 SR 000 0 00000

**POP:** 1110 DR 000 1 00000

The assembly language syntax is:

**PUSH SR**

**POP DR**

Pushing and popping works as discussed in the textbook, using R6 as the stack pointer. The table shown below shows the contents of several memory locations before and after the following assembly language program is executed.

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>x3FFB</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>x3FFC</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>x3FFF</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>x3FFE</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>x3FFF</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

LD R6, STACK; 
AND R0, R0 #0; 
PUSH R0

**Pop R1**
ADD R1, R1, #3
PUSH R1
ADD R2, R1, #1
PUSH R2 ; *
PUSH R2
ADD R3, R2, R2 ; *
PUSH R3
PUSH R3

**Pop R2**

**Pop R1**
ADD R3, R2, R1
PUSH R3
HALT

STACK .FILL x4000
.END

Note that three of the instructions are missing from the assembly language program, and that four instructions have an * in the comment field.

Your job:

**Part a**: Fill in the three missing instructions in the program, one per box, in the boxes provided.

**Part b**: For each of the four instructions with the * comment, provide the corresponding machine language instruction in the box to the right of it.

**Part c**: What is the contents of R6 after this program halts? **x3FFC**
Problem 7 (15 points):

What does the following program do? Explain in not more than 15 words in the box provided below.

```
.ORIG x3000
AND R1, R1, #0
ADD R2, R1, #15
ADD R3, R1, #1
LDI R0, IN_ADDR
L0
  AND R4, R0, R3
  BRz L1
ADD R1, R1, #1
L1
  ADD R1, R1, R1
  ADD R3, R3, R3
  ADD R2, R2, #-1
  BRp L0
  AND R4, R0, R3
  BRz L2
ADD R1, R1, #1
L2
  ST R1, OUTPUT
HALT
IN_ADDR .FILL x4000
OUTPUT .ELKW x1
.END
```

Answer: Reverses the bit string of the value at x4000, and stores the result at OUTPUT.

If you are not sure what this program does, answer the following questions for partial credit. If you know what it does, you do not need to answer these questions.

Part a (1 points): Where in memory is the input to the program:

Answer: x4000

Part b (1 points): Where in memory is the output of the program:

Answer: x3011

Part c (2 points): At the end of the program what are the contents of the following registers (for any input):

R2: 0
R3: x8000

Part d (1 points): What input will cause the output to be 0:

Answer: 0

Part e (2 points): What is the output for the following binary inputs:

0000 0000 0000 0101: 1010 0000 0000 0000 1100 0000 0000 0000: 0000 0000 0000 0011
Name: Solution

Problem 8 (15 points):

What is the output of the following program? Show your work. Be very careful.

```
.ORIG x3000
AND R1, R1, #0
LDI R0, START
ST R1, #0
LEA R0, Message1
Trap x22
HALT
START .FILL x3002
Message1 .STRINGZ "I HOPE"
.BLKW x01F0
Message2 .STRINGZ "I PASSED"
.END
```

Answer:

PASSED