Name: ________________________________

1. Convert the following LC3 Assembly instructions to their binary representation (2 points each):
   a. ADD R4, R3, #12
   b. LDR R2, R3, #-5
   c. BRNP #6
   d. JSR #-14
   e. ADD R4, R4, R4

2. Dis-assemble the following LC3 binary instructions. Express offsets as literal constants because no symbol table is provided. (2 points each):
   a. 0x3428
   b. 0x1260
   c. 0x0FF7
   d. 0x6280
   e. 0xC1C0
3. Describe the errors in the following lines of LC3 code (5 points each):

   a. ADD R0,R1,#-23

   b. JSR ZERO_R7
      ...
      ZERO_R7: AND R7,R7,0
      RET

   c. ADD R3,R3,0
      BRNZ ISPOS
      HALT
      .BLKW 512
      ISPOS NOT R3,R3
      ...

   d. ADD R6,R7,R8

   e. LEA R3,MY_DATA
      MY_DATA .FILL xC2222

4. Answer the following questions in one or two sentences... (3 points each)

   a. What is the Condition Code (CC) in LC3 assembler and how is it set to a value?

   b. Why does LC3 use both registers and addressable memory to save data?

   c. What is the difference between Load and Store instructions?

   d. Describe two addressing modes in LC3 assembler.
5. Write the LC3 Assembler code to add all integers from 1 to 100. Do not make any assumptions about the initial value of any registers. (10 points)

6. Write LC3 Assembler code to perform the function in the following pseudo-code: (Note: Assume the value of R1 is unknown at entry to your code, but the value of R2 has been set already. You may use any general purpose register in your code.) (10 points)

```assembly
for(R1=0; R1<10; R1=R1+2) {
    if ( R2 < 50 ) then { R2 = R2 + R1 }
    else { R2 = R2 − R1 }
}
```
7. Write the LC3 assembler code for a function that multiplies the value of R3 by the value of R4, and puts the product in R5. Your function may change the values of R3 and R4. You may assume R3 is positive or zero, but R4 may be negative, zero, or positive. (10 points)

8. Write the LC3 assembler code to compute the cross-product of two vectors of numbers. (10 points) Assume there are five numbers in each vector. You may assume that you have access to the multiply function you wrote in problem 7. The pseudo-code to compute the cross-product is as follows:

   cp=0
   for(i=0; i<vector_count; i=i+1) {
       cp = cp + ( vec1[i] x vec2[i] )
   }

Assume the following values have been defined for you:

   vec1 .fill #10
   .fill #14
   .fill #28
   .fill #36
   .fill #48

   vec2 .fill #18
   .fill #23
   .fill #32
   .fill #41
   .fill #-10

9. Whose birthday is it today (Thursday July 26)? (3 points)
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Opcode</th>
<th>DR</th>
<th>SR1</th>
<th>Type</th>
<th>SR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD+</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>0101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND+</td>
<td>0101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR</td>
<td>0000</td>
<td>n, z, p</td>
<td></td>
<td></td>
<td>PCoffset9</td>
</tr>
<tr>
<td>JMP</td>
<td>1100</td>
<td></td>
<td></td>
<td>BaseR</td>
<td>000000</td>
</tr>
<tr>
<td>JSR</td>
<td>0100</td>
<td></td>
<td>1</td>
<td></td>
<td>PCoffset11</td>
</tr>
<tr>
<td>JSRR</td>
<td>0100</td>
<td>0, 0</td>
<td></td>
<td>BaseR</td>
<td>000000</td>
</tr>
<tr>
<td>LD</td>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDI</td>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDR</td>
<td>0110</td>
<td></td>
<td></td>
<td>BaseR</td>
<td>offset6</td>
</tr>
<tr>
<td>LEA</td>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>1001</td>
<td></td>
<td></td>
<td>SR</td>
<td>111111</td>
</tr>
<tr>
<td>RET</td>
<td>1100</td>
<td></td>
<td>111</td>
<td></td>
<td>000000</td>
</tr>
<tr>
<td>RTI</td>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td>000000000000</td>
</tr>
<tr>
<td>ST</td>
<td>0011</td>
<td></td>
<td></td>
<td>SR</td>
<td>PCoffset9</td>
</tr>
<tr>
<td>STI</td>
<td>1011</td>
<td></td>
<td></td>
<td>SR</td>
<td>PCoffset9</td>
</tr>
<tr>
<td>STR</td>
<td>0111</td>
<td></td>
<td></td>
<td>SR</td>
<td>BaseR</td>
</tr>
<tr>
<td>TRAP</td>
<td>1111</td>
<td>0000</td>
<td></td>
<td></td>
<td>trapvect8</td>
</tr>
<tr>
<td>reserved</td>
<td>1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>