100 points possible. Open Book, Open Notes.

NAME:______________________________________________________________

1. (5 pts) Which of the following LC-3 instructions move the value in R5 into R6?

i. 0001110101100000
ii. 0101110101111111
iii. 1001110101111111

Circle one: a) Only i  b) Only ii  c) i and ii  d) i and iii  e) i, ii, and iii

2. (8 pts) Show the contents (in HEX) that would be found in the LC-3’s memory, starting at address x3000, if the following assembly language directives were assembled and loaded:

```
.ORIG x3000
Var1 .FILL #25
Var1 .FILL #-10
Var1 .FILL x33
Var1 .FILL x-3A
.END
```

<table>
<thead>
<tr>
<th>Address</th>
<th>HEX Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>x3000</td>
<td></td>
</tr>
<tr>
<td>x3001</td>
<td></td>
</tr>
<tr>
<td>x3002</td>
<td></td>
</tr>
<tr>
<td>x3003</td>
<td></td>
</tr>
</tbody>
</table>

3. (10 pts) How many memory access will occur in processing each of the following LC-3 instructions (not including fetching the instruction itself into the IR):

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Number of memory accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD R1, Var1</td>
<td></td>
</tr>
<tr>
<td>ADD R2, R2, #5</td>
<td></td>
</tr>
<tr>
<td>LEA R0, Var1</td>
<td></td>
</tr>
<tr>
<td>LDI R0, Var1</td>
<td></td>
</tr>
<tr>
<td>LDR R3, R2, #-6</td>
<td></td>
</tr>
</tbody>
</table>

4. (13 pts) Suppose I have a system with a fully associative cache with the following cache and memory parameters:

- Cache line/block size = 8 words
- Total cache size = 512 words
- Total memory = 16 Megawords

   a. (7 pts) Show how memory addresses will be divided into tag and word fields:
b. (4 pts) If a program accesses memory at location x03FB02, what tag will be compared with the tags in the cache? *(Give your answer in binary.)*

c. (2 pts) How many comparisons will need to be made in searching for this tag?

5. (13 pts) Suppose I have a system with a 4-way set associative cache with the same cache and memory parameters as the previous problem:
   Cache line/block size = 8 words
   Total cache size = 512 words
   Total memory = 16 Megawords

   a. (7 pts) Show how memory addresses will be divided into tag, set, and word fields:

   b. (4 pts) If a program accesses memory at location x03FB02, what tag will be compared with the tags in the cache? *(Give your answer in binary.)*

   c. (2 pts) How many comparisons will need to be made in searching for this tag?

6. (6 pts) Suppose an LC-3 CPU receives an interrupt signal and acknowledges it, and then receives the interrupt vector x023. What is the exact address of the next memory location the LC-3 will access? What will it find there?
7. (10 pts) Suppose you are given a memory hierarchy with the following statistics:

<table>
<thead>
<tr>
<th>Location</th>
<th>Average Access Time</th>
<th>Hit Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main memory</td>
<td>200 ns</td>
<td>–</td>
</tr>
<tr>
<td>L1 Cache</td>
<td>50 ns</td>
<td>90%</td>
</tr>
<tr>
<td>L0 Cache</td>
<td>25 ns</td>
<td>80%</td>
</tr>
</tbody>
</table>

What is the average access time for a memory read in this system? *Show your work.*

8. (5 pts) Consider the IEEE 32-bit floating point format below. Which of the following best describes the 32-bit value \( \text{x}815\text{AE}8\text{A9} \), interpreted as a 32-bit IEEE floating-point number (circle one):

- a. A small (near-zero) negative number.
- b. A large negative number.
- c. A small (near-zero) positive number.
- d. A large positive number.
9. (30 pts) Consider the following C++ program fragment:

Note: The parameter y is passed by reference to mysub.

```cpp
int main() {
    int a = 12, b = 29, c = -1;
    c = mysub(a, b);
    ... etc ...
}

int mysub(int x, int &y) {
    int d, e;
    ... more code ...
    d = e + y;
    return d;
}
```

a. (5 pts) Fill in the symbol table, above, that the compiler will generate for the subroutine mysub.

b. (10 pts) To the right is the stack as it would appear during the execution of main. Fill in the rest of the stack as it will appear while executing the region labeled “more code” in mysub. Show the new locations of R5 and R6. Label each stack location. SHOW EXACT HEX VALUES WHEREVER POSSIBLE.

R6 → x6FFD xffffff c
R5 → x6FFF x000c a
R6 → x6FF8
R5 → x6FF9

Note! R5

R6

R5

x6FEF
x6FF0
x6FF1
x6FF2
x6FF3
x6FF4
x6FF5
x6FF6
x6FF7
x6FF8
x6FF9
x6FFA
x6FFB
x6FFC
x6FFF
x7000

Note!

x000d
b

x001d
a

c.

(10 pts) Show the assembly language code that will be generated for the line “d = e + y;” in mysub. Use R0 and R1 for temporary storage, if necessary.

d. (5 pts) Show the assembly language code that will be generated to store the return value and clean up the stack in main, just after returning from mysub.