CSci Spring 2020 Section 010 Problem Set 3

Due in plain text or PDF format on Canvas at the beginning of lecture (3:35pm) on Friday, May 1st. We recommend that you type your solutions with a text editor or word processor and then convert them to PDF. Please label your assignment with your name, UMN email address, and the time of your lab section (12:10 pm, 1:25 pm, or 2:30 pm).

Problem 1

Below are the C code and the corresponding assembly code for a function that reads a value from a three-dimensional array with bounds checking. However, we've left the dimensions of the array a mystery for you to figure out.

```
long multi[ A ][ B ][ C ];

long access_multi(long i, long j, long k) {
    long *ptr = &multi[i][j][k];
    if (ptr - &multi[0][0][0] >= sizeof(multi)/sizeof(long)) {
        return -1;
    } else {
        return *ptr;
    }
}
```

```
access_multi:
        imulq   $3360, %rdi, %rax
        imulq   $280, %rsi, %rcx
        addq    %rax, %rcx
        leaq    (%rcx,%rdx,8), %rsi
        sarq    $3, %rsi
        movq    $-1, %rax
        cmpq    $8399, %rsi
        ja      .LBB0_2
        movq    multi(%rcx,%rdx,8), %rax
.LBB0_2:
        retq
```

What values of A, B, and C correspond to the assembly language code shown? In understanding the assembly code, you may find it helpful to express each new value stored in a register in terms of the parameters i, j, and k. (The three-operand form of the `imul` instruction multiplies together its first two arguments and puts the product in the last operand.)

Problem 2

The median of an odd number of values is the one that would be in the middle position if they were ordered: for instance the median of 6, 1, and 2 is 2, and the median of 1, 1, and 2 is 1.

For this question, write a Y86-64 function that takes three long integers in `%rdi`, `%rsi`, and `%rdx`, and returns their median according to signed comparison. However, you may not use conditional jump instructions in your solution, only conditional move instructions.

Problem 3

Suppose that A and B are HCL word values representing 64-bit signed integers, and S is an HCL word value representing the low 64 bits of their sum, again interpreted as signed. Write an HCL expression in terms of A, B, and S, that evaluates to true (1) when signed overflow occurred in the addition, and to false (0) if there was no overflow.

Problem 4

This question concerns the cache of a small system. The following table shows the contents of the cache. All addresses, tags, and values inside of the table are represented in hexadecimal. The cache used for this question has the following properties:

- Addresses are 12 bits wide.
- Memory is byte addressable.
- Memory accesses are to individual bytes.
- The cache is two-way set associative (E=2), with 4-byte block size (B=4) and four sets (S=4).

Set Index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	00	0	-	-	-	-
	A6	1	20	68	A0	83
1	00	1	83	4B	EF	49
	7A	0	-	-	-	-
2	01	1	1E	AF	42	B1
	5A	0	-	-	-	-
3	DA	1	8A	71	AF	CA
	64	1	C3	B1	46	89

A. The diagram below shows the 12 bits of an address, most-significant on the left. Draw a copy of the diagram in which each bit is labeled according to how it is used by the cache. Use the following letters to label the diagram:

O - Cache block byte offset
I - Cache set index
T - Cache tag

11	10	9	8	7	6	5	4	3	2	1	0

B. For the following memory addresses, simulate the behavior of the cache for a byte load from that address. State whether the access is a hit or a miss, and if the load is a hit, say what value it would return. (Consider each load as starting with the same cache starting state shown above; you don't need to consider what replacement would be performed for a miss.)

- Load from 0x5AA
- Load from 0xA62
- Load from 0x7A7
- Load from 0xDAC