CS 3843 Computer Organization, Fall 2013 Assignment 5

Assigned on Monday Nov. 4, 2013
Due Monday, Nov. 11, 2013

Problem 1 (35 points) Fill in the following table on the cover sheet
Assume that x and y are of type int which is 32 bits. Enter the value of (y-x) in decimal. This is the value that would be stored in z if

\[ \text{int } z = y - x; \]

Consider the instruction:

```
   cmpl %eax, %ecx
```

Fill in the value of the flags if %eax contains x and %ecx contains y.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>z = y - x</td>
<td>ZF</td>
<td>SF</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>-------------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>42</td>
<td>-15</td>
<td>(1) signed -15-42 = -57</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) unsigned 4294967281-42 = 4294967239</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-15</td>
<td>42</td>
<td>(1) Signed 42-(-15) = 57</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) unsigned 42-4294967281 = -4294967239</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-17</td>
<td>-17</td>
<td>-17-(-17) = 0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0x7fffffffd</td>
<td>67</td>
<td>(1) signed 67-2147483645 = -2147483578</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) unsigned 67-2147483645 = -2147483578</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7fffffffd</td>
<td>-67</td>
<td>(1) signed -67-2147483645 = -2147483712</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) unsigned 4294967229-2147483645 = 2147483584</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>0x7fffffffd</td>
<td>(1) signed 2147483645-67 = 2147483578</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) unsigned 2147483645-67 = 2147483578</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>-0x7fffffffd</td>
<td>(1) signed 2147483645-67=-2147483712</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) unsigned 2147483651-67=2147483584</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sol:

The range of 32-bit signed number: -2147483648 ~ 2147483647
The range of 32-bit unsigned number: 0 ~ 4294967296
2's complement representation:

-15 = \(N^* = 2^{32} - N = 4294967296 - 15 = 4294967281\)

0x7fffffff = \(2^{31} - 3 = 2147483645\)

-67 = \(N^* = 2^{32} - 67 = 4294967296 - 67 = 4294967229\)

-0x7fffffff = \(N^* = 2^{32} - 0x7fffffff = 2^{32} - 2147483645 = 2147483651\)

**Problem 2** (30 points). A function with prototype

\[
\text{int decode2(int } x, \text{ int } y, \text{ int } z)\; \text{;}
\]

is compiled into IA32 assembly code. The body of the code is as follows:

1. movl 12(%ebp), %edx  // y into %edx
2. subl 16(%ebp), %edx  // compute y-z
3. movl %edx, %eax  // y-z into %eax
4. sall $31, %eax  // (y-z)<<31
5. sarl $31, %eax  // (y-z)<<(31)>>31
6. imull 8(%ebp), %edx  // x * (y-z)
7. xorl %edx, %eax  // (x * (y-z)) ^ ((y-z)<<31)>>31

Parameters \(x, y,\) and \(z\) are stored at memory locations with offsets 8, 12, and 16 relative to the address in register %ebp. The code stores the return value in register %eax.

Write C code for \(\text{decode2}\) that will have an effect equivalent to our assembly code.

\[
\text{int decode2(int } x, \text{ int } y, \text{ int } z) \{ \!
\\text{return } (((y-z) << 31) >> 31) ^ (x * (y-z)); \}
\]

**Problem 3** (40 points). Consider the following assembly code:

\[
x \text{ at } %ebp+8, \; n \text{ at } %ebp+12
\]

1. movl 8(%ebp), %esi  // \(x\) into %esi
2. movl 12(%ebp), %ebx  // \(n\) into %ebx
3. movl $-1, %edi  // result = -1
4. movl $1, %edx  // mask = 1
5. .L2:
6. movl %edx, %eax  // %eax = mask
7. andl %esi, %eax  // \(x\) & mask
8. xorl %eax, %edi  // result ^ = (mask & \(x\))
9. movl %ebx, %ecx  // %ecx = \(n\)
The preceding code was generated by compiling C code that the following overall form:

1. int loop (int x, int n)
2. {
3.     int result = ____-1____;
4.     int mask;
5.     for (mask = ____1____; mask_!=0_____; mask =_mask << n_______) {
6.         result ^= _(mask & x)_______;
7.     }
8.     return result;
9. }

Your task is to fill in the missing parts of the C code to get a program equivalent to the generated assembly code. Recall that the result of the function is returned in register %eax. You will find it helpful to examine the assembly code before, during, and after the loop to form a consistent mapping between the registers and the program variables.

A. Which registers hold program values x, n, result, and mask? (4 points)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>%esi</td>
</tr>
<tr>
<td>n</td>
<td>%ebx</td>
</tr>
<tr>
<td>result</td>
<td>%edi</td>
</tr>
<tr>
<td>mask</td>
<td>%edx</td>
</tr>
</tbody>
</table>

B. What are the initial values of result and mask? (4 points)

Ans: result is -1; mask is 1

C. What is the test condition for mask? (4 points)

Ans: mask not zero

D. How does mask get updated? (4 points)

Ans: Register %edx holds the value of mask. So mask gets updated every time %edx is left shifted by n bits in line number 10. testl instruction in line 11 does not change the register value and only changes the flags.

E. How does result get updated? (4 points)

Ans: Bitwise AND is applied on mask and x and the result is XORed with result;

F. Fill in all the missing parts of the C code. (20 points)

Ans: Check code above.