Lab 3: Multiplication and Exponent

Submission timestamps will be checked and enforced strictly by the CourseWeb; **late submissions will not be accepted**. Check the due date of this lab on the CourseWeb. Remember that, per the course syllabus, if you are not marked by your recitation instructor as having attended a recitation, your score will be cut in half.

In this lab, you are going to learn how to use bit-wise operations to achieve faster multiplication.

Multiplication

For this lab, you are not allowed to use any multiplication instruction. In other words, you must implement a multiplication algorithm. A simple way to calculate $x \times y$ is by using addition. Since $x \times y$ is equal to $x + x + \ldots + x$ (y times), we can use a simple loop to calculate $x \times y$. However, if y is a large number, this algorithm requires a large number of iterations.
As we discussed in class, since binary is a base-number system, we can use long multiplication method of decimal to calculate binary multiplication. Consider 9×11 in 4-bit binary (unsigned):

```
   1001
   1011 x
   ----
   1001
  1001
 0000   +
1001
-------
1100011
=======
```

Recall that 1100011_2 in unsigned binary is 99 in decimal. Note that instead of 11 addition operations, the above long multiplication only requires 4 addition operations. The number of addition operations required by this method does not depend on the value of a multiplier. It only depends on the number of bits of multiplicand and multiplier. For a 16-bit unsigned numbers x and y, to calculate $x \times y$ only requires 16 addition operations.
This multiplication algorithm can be easily implemented using shift, bit-wise, comparison, and addition operations. Now, consider 9×11 in 4-bit binary (unsigned) one more time:

```
   1001
   1011 x
   ----
   1001     <-- (1001 << 0) * 1 where 1 is the LSB of the multiplier
  1001      <-- (1001 << 1) * 1
 0000   +   <-- (1001 << 2) * 0
1001        <-- (1001 << 3) * 1 where 1 is the MSB of the multiplier
-------
1100011
=======
```

As you may notice, we accumulate the result (starting from 0) using a series of addition operations. The first addition operation depends on the multiplier shifted left by 0, the second addition operation depends on the multiplier shifted left by 1, and so on. Whether too add 0 or non-zero to the result at a step, it depends on a certain bit of the multiplier. The first addition depends on the LSB (bit 0) of the multiplier, the second addition depends on bit 1 of the multiplier, and so on. To check whether a bit of a number is 0 or 1 can be easily done using shift and/or bit-wise AND operations.

Exponent

For this lab, you just have to implement a simple exponent algorithm. Simply use the fact that x^y is equal to $x \times x \times x \times \ldots \times x$ (y times). This algorithm is not efficient, however, we will not test your algorithm with a large number since the result may not fit in a 32-bit register. **Note** that your are not allowed to use any multiplication instruction. You MUST use your implementation of multiplication as described in previous section.

What to do?

Write an assembly program named `lab03.asm` that asks a user to enter a positive integer for x and another positive integer for y. If a user enter a negative number, ask the user to enter a new number. Once a user enter both number, simply display the result of $x \times y$ and x^y as shown below:

```
Please enter a positive integer: -123
A negative integer is not allowed.
Please enter a positive integer: 12
Please enter another positive integer: -3
A negative integer is not allowed.
Please enter another positive integer: 3
12 * 3 = 36
12^3 = 1728
```

Please make sure that output of your program is similar to the above example. **Note** that we will enter a couple of large numbers to check your multiplication algorithm but we will ignore the result of exponent since it may be larger than a 32-bit value. When we test your exponent algorithm, we will use a couple of small numbers.

Submission

Submit your `lab03.asm` file via CourseWeb before the due date stated on the CourseWeb.