CSE 2312: Computer Organization & Assembly Language Programming
Summer 2015
Program #2

In this assignment, you will implement an iterative solution for computing the GCD of two positive integers. Your program, at a minimum, will consist of the following procedure call:

> GCD_ITERATIVE: Computes the GCD of integers stored in R1 and R2 iteratively and returns the result in R0. An iterative solution for computing the GCD in C code is given below (you do not have to directly port this example to assembly, but you must iteratively solve the problem):

```
    int gcd_iterative(int x, int y)
    {
        while((x % y) != 0)
        {
            y--;
        }
        return y;
}
```

Your main function will contain a loop that continuously checks for keyboard input in the following pattern:

```
<OPERAND_1><ENTER>
<OPERAND_2><ENTER>
```

Once the 2 lines of input are acquired, the operands should be loaded into the proper registers and the GCD_ITERATIVE procedure should be called. The procedure should return the result in register R0, and the main function should print the value to the console and skip to a new line.

All input test cases will consist of positive numbers only. The value stored in OPERAND_1 will always be greater than or equal to OPERAND_2. Below are some example use cases:

24	100	500	200
12	75	500	150
GCD: 12	GCD: 25	GCD: 500	GCD: 50

You will also need to implement a recursive solution for computing the GCD of two positive integers. Your program, at a minimum, will consist of the following procedure call:

GCD_EUCLID: Computes the GCD of integers stored in R1 and R2 recursively using Euclid's algorithms and returns the result in R0. A C code implementation of Euclid's algorithm is given below (you do not have to directly port this example to assembly, but you must implement Euclid's algorithm to solve the problem):

```
int gcd_euclid(int x, int y)
{
        if (y == 0)
        {
                return x;
        }
        else if (x >= y && y > 0)
        {
                return gcd_euclid(y, (x % y));
        }
}
```

Your main function will contain a loop that continuously checks for keyboard input in the following pattern:

```
<OPERAND_1><ENTER>
<OPERAND_2><ENTER>
```

Once the 2 lines of input are acquired, the operands should be loaded into the proper registers and the GCD_ EUCLID procedure should be called. The procedure should return the result in register R0, and the main function should print the value to the console and skip to a new line.

All input test cases will consist of positive numbers only. The value stored in OPERAND_1 will always be greater than or equal to OPERAND_2. Below are some example use cases:

```
24          100         500         200
12          75          500         150
GCD: 12     GCD: 25     GCD: 500    GCD: 50
```