Project 2: Hamming Distance

Due: Tue 09/23/03, Section 0101 (Chang) & Section 0301 (Macneil)
Wed 09/24/03, Section 0201 (Patel & Bourner)

Objective

The objective of this programming project is to practice designing your own loops and branching code in assembly language and to gain greater familiarity with the i386 instructions set.

Assignment

Write an assembly language program that prompts the user for two input strings and computes the Hamming distance between the two strings. The Hamming distance is the number of bit positions where the two strings differ. For example, the ASCII representations of the strings `"foo"` and `"bar"` in binary are:

```
"foo" = 0110 0110 0110 1111 0110 1111
"bar" = 0110 0010 0110 0001 0111 0010
```

So, the Hamming distance between "foo" and "bar" is 8.

Some details:

- Your program must return the Hamming distance of the two strings as the exit status of the program. This is the value stored in the EBX register just before the system call to exit the program.
- To see the exit status of your program, execute the program using the Unix command:

  ```
  a.out ; echo $?
  ```

- Since the exit status is a value between 0 and 255, you should restrict the user input to 31 characters.
- If the user enters two strings with different lengths, your program should return the Hamming distance up to the length of the shorter string.
- Look up the i386 instructions ADC and XOR and determine how these instructions are relevant to this programming project.
- Record some sample runs of your program using the Unix script command.

Implementation Notes

- The easiest way to examine the contents of a register bit-by-bit is to use successive SHR instruction to shift the least significant bit into the carry flag.
- When you use the gdb debugger to run your program, note that gdb reports the exit status as an octal (base 8) value. The Unix shell reports the exit status in decimal.
- The Hamming distance between the following two strings is 38:

  ```
  this is a test
  of the emergency broadcast
  ```

  You must also make your own test cases.
- Part of this project is for you to decide which registers should hold which values and whether to use 8-bit, 16-bit or 32-bit registers. A logical plan for the use of registers will make your program easier to code and easier to debug — i.e., think about this *before* you start coding.

Turning in your program

Use the UNIX `submit` command on the GL system to turn in your project. You should submit two files: 1) the modified assembly language program and 2) the typescript file of sample runs of your program. The class name for submit is `cs313_0101`, `cs313_0102` or `cs313_0103` for respectively sections 0101 (Chang), 0201 (Patel & Bourner) or 0301 (Macneil). The name of the assignment name is `proj2`. The UNIX command to do this should look something like:

```
submit cs313_0101 proj2 hamming.asm typescript
```