Programming Exercise 2: Base Conversion

Due: Tuesday February 26, 2002

Objective

The objective of this programming exercise 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 an input string, reads the input string and interprets it as an unsigned number in base 10. Your program must convert the ASCII string representation of this number into a 32-bit unsigned binary number and store it in a memory location. Finally, your program must prepare and store an ASCII string for the hexadecimal representation of the same number and print it out.

Your program should generate an error if the input string contains a character that is not 0 through 9 (except for the linefeed at the end) and if the number does not fit in 32 bits.

Example:

```
linux1% a.out
Enter Base 10 number: 32767
Original: 32767
Convert:  00007FFF
```

Implementation Issues:

1. The MUL instruction always multiplies something with the EAX, AX or AL register. The product is stored in EDX:EAX, DX:AX or AX. So the EDX register might be affected by the MUL instruction. Also, the MUL instruction does not take an immediate operand.
2. The user input has a linefeed character at the end. You do not want to consider this character when you convert from base 10 to binary.
3. The length of the output string is not the same as the length of the input string! The ASCII string printed out always has 8 characters followed by a linefeed (assuming that you print out leading zeroes).
4. You will find the rotate left instruction ROL especially useful when you prepare an ASCII string for the hexadecimal representation of a number.
5. Note that the character ‘A’ does not follow the digit ‘9’ in ASCII. Remember this when you convert the number into a hex string.

Turning in your program

Use the UNIX script command to record some sample runs of your program. You should use inputs that result in a variety of hex string outputs and also inputs that demonstrate the error checking features of your program. For some of the test cases, use the gdb debugger to show that the correct number is stored in memory.

You should submit two files:1) your assembly language program and 2) a typescript file of your sample runs. The class name for submit is ‘cs313’ and the assignment name is ‘prog2’. The UNIX command to do this should look something like:

submit cs313 prog2 convert.asm typescript