CMSC 313 — Spring 2017
Project 3 — Print Decimal Subroutine

Assigned	Wednesday, March 8th
Program Due	Tuesday, March 28th by 11:59pm (originally due 3/17)
Updates:	Project due date has been extende from 3/17 to 3/28, due to the snow.

Objectives

The objectives of the programming assignment are 1) to gain further experience writing subroutines in assembly language 2) to learn how to pass parameters on the stack 3) to practice saving values on the stack for reusing registers 4) to gain further experience with loop control

Assignment

For this assignment, you will write an assembly language subroutine that outputs the printable string version of a 32-bit unsigned number to standard output. For example, the number 456123 needs should be converted to the 6-char string "456123", and then printed out. The 32-bit value will be passed to your subroutine ***on the stack***. (It will be the last thing pushed on the stack before the CALL instruction.) Your subroutine must preserve *all* registers (i.e., any registers that you use must be saved on the stack and restored before you return, even EAX). Your subroutine should do nothing else (not even print a linefeed).

To determine the values of the base 10 digits in the decimal representation of the number, you will use the division method (do not use repeated subtraction). Continuously dividing by 10 will give you the next rightmost digit as the remainder each time. The DIV instruction will divide the 64-bit value stored in EDX:EAX by the single 32-bit operand given. After the DIV instruction is executed, the quotient is stored in the EAX register and the remainder is stored in the EDX register. (See further discussion of the DIV instruction in Implementation Notes below.)

Implement your subroutine in a separate file, named `prt_dec.asm`. The entry point of your subroutine must be called `prt_dec`. The "main program" will reside in a separate file. A sample main program is provided for you (`main3.asm`), but you must make your own test program as well, called `mytest3.asm`. **Note:** your test program **must** be in a separate file, otherwise your subroutine will fail to compile with the test program used for grading.

When you assemble your program, you must assemble each source file separately:

```
linux2% nasm -f elf -g main3.asm
```

```
linux2% nasm -f elf -g prt_dec.asm
linux2% ld main3.o prt_dec.o -melf_i386 -o main3.out
```

(We suggest you adapt the Makefile from previous projects. Note that the "OBJS=" line can take multiple .o files, separated by spaces.)

Then, you can run the `main3.out` file produced as usual. A sample run with the `main3.asm` file provided should look like:

```
linux2% ./main3.out
0
4294967295
3413151
17
214123
2223187809
1555544444
2 plus 3 equals 5
7 minus 4 equals 3
```

Note: The graders will use a different main program to test your subroutine. Your test program should fully exercise your subroutine and check for cases that are not tested in `main3.asm`. You can use main3.asm as your template and just modify the test cases.

Implementation Notes/Hints

- The `main3.c` program for Project 3 is also available for copying directly on the GL file system at:

```
/afs/umbc.edu/users/p/a/park/pub/cmsc313/spring17/proj3/main3.asm
```

- Note that the parameter to your subroutine (the number to be converted and printed) will have been the last thing pushed onto the stack by the calling code. However, the CALL instruction will push another item (the EIP register) onto the stack, and any PUSH calls you make at the start of your code will further change EIP. Therefore, to fetch the instruction, you must do appropriate accounting for where the argument will be with respect to the most recently updated stack pointer.
- ***It is a requirement of the project that you must output the entire converted number with a single write syscall.*** That means that if the number has more than one digit, all digits must first be completely prepared into a single buffer in the correct order, before printing with one SYSCALL_WRITE. For example, it is not allowed to push the digits onto the stack, then pop them and print them one at a time. The number of characters that you will print will obviously depend on the value passed in as the argument. You will need to determine or keep count of the length of the decimal number to be printed.

In the division method for converting a number to base 10, you will get the one's place first (the least significant digit), then the 10's place, then the 100's place, ... This is in the opposite order that the decimal number will be printed. The easiest way to handle this situation is to initialize some register to point to the ***end*** of the output buffer and loop backwards, decrementing the pointer as you store each character.

To repeat one more time: ***you are not allowed to print each digit separately with a separate write syscall!***

- If the argument passed in is 0, then your subroutine should print out a single digit 0: no more, no less.
- In order for code in other files to call your subroutine, you must declare the `prt_dec` label to be `global`. (If the label is not global, then only code from the same file can use that label.)
  Thus, you must include the following declaration in your `prt_dec.asm` file:

```
global prt_dec
```

  The `global` declarations are typically made just after the `SECTION .text` declaration.

- In your test main program, you must tell the NASM assembler that the `prt_dec` label is from a different file. Otherwise, it will only look for `prt_dec` in the current file (and not find it). To tell NASM this, include the declaration:

```
extern prt_dec
```

  The `extern` declarations are also typically made just after the `SECTION .text` declaration.

- There are a couple of quirks about the DIV instruction. First, since the dividend is always EDX:EAX, it is not specified in the instruction. For example, to divide EDX:EAX by EBX, we simply say:

```
div    ebx
```

  Another quirk is that the DIV instruction *does not* support immediate operands. So, you cannot divide by 10 by saying:

```
div     10       ; this is WRONG
```

- Since you are converting a 32-bit value into decimal, the dividend always fits in the lower 32 bits, i.e., inside EAX. However, EDX is still part of the computation, so you must zero out the EDX register before *each* DIV instruction.
- We are always dividing by 10 in this program. So, we know that the remainder in EDX will always be less than 10. This value fits in 8 bits, which is very convenient for us. Why?

- Remember that the bytes that you are printing to the screen will be interpreted as ASCII values.

What to Submit

Before you submit your program, record some sample runs of your program using the UNIX script command. You should select sample runs that demonstrate the features supported by your program. Picking good test cases is your responsibility.

Use the UNIX `submit` command on the GL system to turn in your project. You should submit *three* files: (1) your assembly language program, named as `prt_dec.asm`, (2) the assembly language test program, named as `mytest3.asm`, and (3) the typescript file of your sample runs. The UNIX command to do this should look something like:

```
submit cs313_park proj3 prt_dec.asm mytest3.asm typescript
```