x86 assembly programming

Introduction

In this assignment, you will write programs using Intel x86 assembly(nasm syntax) with help of C library to solve some simple questions. We will be using the netwide assembler(nasm) and gcc to compile the assembly code into an executable for testing. You have to write the entire solution in x86 assembly: you cannot invoke any other code except the standard C library and code you write. All these are doable with help of the C library. Be sure to read the submission guidelines for the assignment and remember the rules applicable to all assignments.

Question 1: Fibonacci calculator

In this question, you will implement a program that computes the Nth fibonacci number and prints its value to stdout. If no fibonacci number exists, print -1. N will be passed as a command line argument to your tool. See sample session for more.

Sample session

```
$ ./compute-fib.out 1 2 3 4 5
0
1
1
2
3
$ ./compute-fib.out 10
34
$ ./compute-fib.out -12
-1
$ ./compute-fib.out 48
2971215073
```

How we will test your program

We will compile your assembly program using nasm and gcc as shown below and run the resulting executable. We will pass at least one N to your program as a command line argument, as shown in the sample session.

```
$ nasm -f elf compute-fib.asm
$ gcc compute-fib.o -o compute-fib.out
$ ./compute-fib.out <list of Ns>
```

Question 2: Run length encoding

In this question, you will implement run length encoding. Run length encoding is a simple data compression technique where consecutively repeated characters are stored just once along with their count. You can assume that input will consist only of uppercase and lowercase English alphabets. See sample session for more.
This question is from the run-length-encoding exercise on exercism.io.

Sample session

```
$ ./run-length-encoder.out ccccccccUUUUUpppppppTTTTTTTTTTfffffffffgggaaaaSSSSSSSSS
  9c5U8p10T9f3g4a9S
$ ./run-length-encoder.out mmmmmmmmmmjjjjjjOOOOttttttkkkkkkkkkbbbbIIIIIIIITTTCFFF
  10m6j4O6t9k4b8I3TC3F
$ ./run-length-encoder.out UUUUUUUUUUyyyyAAAAAooooooooooWWWWd \
  aaaaaaaMQQQQQQDDgggggggjjjjjjjjXXXXXX \
  TPPPPPPPPgggggxxxxxxxxxQQQQQQQQQQVVVVVjjj
  10U4y5A10o4Wd
  7aM6Q2D7g8j6X
  T8P5g9x10Q5V3j
```

How we will test your program

We will compile your assembly program using nasm and gcc as shown below and run the resulting executable. We will pass at least one string to your program as a command line argument, as shown in the sample session.

```
$ nasm -f elf run-length-encoder.asm
$ gcc run-length-encoder.o -o run-length-encoder.out
$ ./run-length-encoder.out <list of strings>
```

Question 3: SHA256 calculator

In this question, you will implement a program that accepts a filename as argument and prints the SHA256 checksum of the contents of the file. Use the OpenSSL library functions to compute the hash of the file's contents. You can assume the file, whose name is passed as argument, will exist before your program executes. Display the SHA256 checksum in hexadecimal digits. See sample session for more.

Sample session

```
$ cat file1.txt
abcdef
$ wc -c file1.txt
6
$ ./calc-sha256.out file1.txt
bef57ec7f53a6d40beb640a780a639c83bc29ac8a9816f1fc6c5c6dcd93c4721
$ sha256sum file1.txt
bef57ec7f53a6d40beb640a780a639c83bc29ac8a9816f1fc6c5c6dcd93c4721 file1.txt
$ cat file2.txt
abcdefghijklmnopqrstuvwxyz
$ wc -c file2.txt
26
$ ./calc-sha256.out file2.txt
71c480df93d6ae2f1efad1447c66c9525e316218cf51fc8d9ed832f2daf18b73
$ sha256sum file2.txt
71c480df93d6ae2f1efad1447c66c9525e316218cf51fc8d9ed832f2daf18b73 file2.txt
```

How we will test your program

We will compile your assembly program using nasm and gcc as shown below and run the resulting executable. We will pass exactly 1 filename to your program as a command line argument, as shown in the sample session. You will need to install the libssl-dev package if gcc command below fails for you. You can use the sha256sum command to verify if you answers are correct or not.

```
$ nasm -f elf calc-sha256.asm
$ gcc calc-sha256.o -o calc-sha256.out -lcrypto
$ ./calc-sha256.out <filename>
```