x64 assembly programming without C library

Introduction

In this assignment, you will write programs using Intel x64 assembly(nasm syntax) without help of C library i.e. you will interact with OS system call interface directly. 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 x64 assembly: you cannot invoke any other code except the kernel system call interface and code you write.

Question 1: Environment variable query tool

In this question, you will implement a tool that prints the value of all environment variables passed to the program. See sample session for more.

Sample session

```
$ ./get-env.out
...
HOME=/home/blah
USER=blah
....
```

How we will test your program

We will compile your assembly program using nasm and gcc and run the resulting executable. No arguments will be passed to the program.

Question 2: File encryption

In this question, you will implement a simple tool that performs XOR encryption. We will pass two values to the program: a filename F and a key K. The tool will read the contents of F, encrypt them by XORing with K and print the result to stdout.

Sample session

The actual execution of the executable will be similar to the following:

```
$ ./encrypt-file.out FILENAME KEY
```

However, since the keys and file contents can be non-readable, the below sample sessions display all non-ASCII characters in a readable format. To generate this readable format which you can use for debugging, create a file “repr-output.py” with the content shown below.

```
$ cat repr-output.py
print(repr(raw_input()))
```

Now you can copy-paste all the commands(without needing to understand them) and compare the output.

```
$ cat test.txt
abcdefghijklmnopqrstuvwxyz
$ wc -c test.txt          # Command displays number of characters in test.txt
26 test.txt
$ ./encrypt-file.out test.txt -+ | python2 repr-output.py
'LINOHMJCDAFG@EB[\\Y^_X]ZSTQ'

$ cat file.txt | python2 repr-output.py
'\xaa\t\xef\xb0\xe5'
$ wc -c file.txt          # Command displays number of characters in file.txt
5
$ ./encrypt-file.out file.txt `python -c "print '\x13G'"` | python2 repr-output.py
'\xb9N\xfc\xf7\xf6'

$ cat file.txt | python2 repr-output.py
'z/\x94\xc1\x1ar\x10\xb7\x98b'
$ wc -c file.txt          # Command displays number of characters in file.txt
10
$ ./encrypt-file.out file.txt `python -c "print '\xe6Hy\xe7'"`| python2 repr-output.py
'\x9cg\xed&\xfc:iP~*'

$ cat file.txt | python2 repr-output.py
'\xdf\xa9\x84>ezy'
$ wc -c file.txt          # Command displays number of characters in file.txt
7
$ ./encrypt-file.out file.txt `python -c "print '\xa7\x84\x19'"`| python2 repr-output.py
'x-\x9d\x99\xe1c\xde'
```

How we will test your program

We will compile your assembly code using nasm and gcc and run the resulting executable. We will pass the filename and the key as command line arguments, as shown in the sample session.

Question 3: Shellcode

In this question, you will write assembly code to read a file, flag.txt, by invoking the cat command. The code should satisfy the following conditions:

1. The code must contain a globally accessible function named shellcode, which will be invoked by the grader.
2. The shellcode must use the cat command to print contents of flag.txt. Writing assembly code to read flag.txt is not accepted(it will likely not fit within the size limits specified below).
3. There should be no null bytes in the code.
4. The code should be at most 55 bytes in size.
5. .data and .bss sections cannot be used. The code must contain only the .text segment.
6. The code must not make any assumptions about the available of memory locations and data already present in them, contents of all registers and the contents of file “flag.txt”.

You can assume that “flag.txt” will exist before your shellcode runs and that cat command is installed to normal location(/bin).

How will we test your program

We will compile your assembly code, check if above conditions are satisfied and invoke the shellcode function in your code without any arguments, if all are satisfied. If any conditions are not satisfied, it will printed out by the grader.