UNIVERSITY OF CALIFORNIA, SANTA CRUZ
BOARD OF STUDIES IN COMPUTER ENGINEERING

CMPE12/L: COMPUTER SYSTEMS AND ASSEMBLY LANGUAGE

Lab 5: Vigenère cipher
Worth 100 points (90 lab + 10 report)

Lab Objectives:

In this lab, you will build on your MIPS skills, and apply them to a more complex program.

This lab focuses on the design and use of functions. These are modular pieces of code that can run independently from other elements of your code. As coding projects become larger and more complex, good function design becomes essential to the design process.

In this lab, you will create a Vigenère Cipher, which encrypts and decrypts a user-supplied string. Your code must read the key and clear text, encrypt it, decrypt it, and then display the encrypted and plaintext versions of the string. Look at the sample output for exact requirements.

Lab Overview:

You will be implementing the Vigenère cipher (see https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher) to encrypt and decrypt strings provided as the program arguments.

There are two required functions for this lab as shown below:

- **encode**(address of cipher key, address of clear text, address of cipher text): Takes in the addresses of 2 properly terminated strings and the address of memory for output. It will encode the clear text using the given key and store it in the cipher text memory. Be sure to properly terminate the output string.
- **decode**(address of cipher key, address of cipher, text address of clear text): Takes in the addresses of 2 properly terminated strings and the address of memory for output. It will decode the cipher text using the given key and store it in the clear text memory. Be sure to properly terminate the output string.

Neither of these functions perform input or output, this will be the responsibility of your main code.

Instead of providing a flow chart for the code, we are instead providing the source code in C. The code contained in cipher.c is fully functional except that it does not use program arguments for brevity. Match its output format. Do not use a C compiler to write the code.

Program Flow:

Outside of the functions your code has a few requirements.

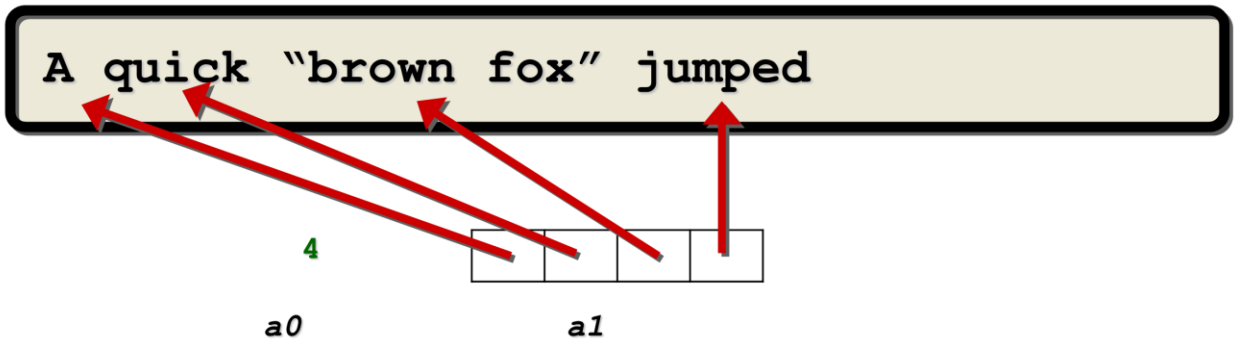
- Use the program arguments to determine both the key and the clear text. The key will be the first string while the second will be the clear text.
- Print out the given clear text and the key.
- Encode using the encode function and print out the encoded version.
- Decode using the decode function and print out the decoded version. They should match and you may not print out the input text again.

For ease of use we are limiting the length of the clear text to 100 characters. Part of your data segment should include declarations for both the encoded and decoded versions of your string. Assume sane input and do not attempt to correct invalid input.

Program Arguments:

As in lab 3 we are using program arguments as input to your program. Unlike lab 3 we will not be providing you the code that handled them for you.

Arguments to your program are stored as strings, you saw this when you had to convert the decimal number to binary. Strings are separated by spaces unless enclosed by double quotes. Look at the example below:



In this case we have four program arguments: *A*, *quick*, *brown fox* and *jumped*. We use *a0* and *a1* to interface with the program arguments in the following manner:

- a0 (argc): This is an integer value telling us how many program arguments have been received. This value can be used to iterate over all the submitted strings.
- a1 (argv): This is an array of memory addresses to strings. Loading a1[0] for example would return the address where "A" is stored while a1[1] would return the address to "quick". Each string is null terminated.

For the purposes of this program you will need to use the first two elements of the array.

Sample Output:

For full credit you must match the output format. The output below is from three separate runs of the program.

```
The given key is: Latin
The given text is: repetitio est mater studiorum
The encrypted text is: >FdNb5U]X1Th    [-UY[?UiMW;SiV
The decrypted text is: repetitio est mater studiorum


The given key is: ABC
The given text is: abcdefghijklmnopqrstuvwxyz
The encrypted text is: "$&%')(*,+-/.0213546879;:<
The decrypted text is: abcdefghijklmnopqrstuvwxyz


The given key is: 01234
The given text is: We are not amused
The encrypted text is:  R & Q "(P ('
The decrypted text is: We are not amused
```

Note that some encrypted text will not be readable directly but the decrypted text should always match the input text.

Lab Requirements

2 files required in lab5 folder in repository with commit id submitted to the google form:

- Lab5.asm
- README.txt

Check lab1 if you do not recall our expectations for lab write-ups. In addition:

- Discuss the encode/decode algorithms. What similarities are there between them?
- Discuss the functions you made and what you were required to save.