Math 230 Project 3: Encryption/Decryption Using MIPS

B. Smith

October 20, 2017

1 Encryption/Decryption

The ASCII character set represents each character with an integer. It is therefore possible to perform some sort of arithmetic operations on these integer values to get a value that represents a different integer. The process of *encryption* takes a string of characters, `String1`, and performs some operations on the characters so that you end up with a different string, `String2`, which hides the original message. The process of encryption should be reversible - that is, if you perform the inverse operation on `String2`, you should be able to get the original message `String1`. This revers process is known as *decryption*. Note: The arithmetic operations take two operands - one will be the character you are encrypting, and the second operand is a user-selected "key" to determine how the encryptions takes place (e.g., which bits to invert).

2 Task Description

In this assignment you will input a string of characters from the keyboard, encrypt or decrypt it and print it back to the screen. Encryption and decryption are to use 2 keys. Details as follows:

2.1 Encryption

The Encryption process should have two operations:

1. Addition of a number (the addition key) to the character's ASCII value. See Caesar Cipher for additional interesting background information.
2. Flipping a single bit in the result of part 1 - the second key specifies which bit to toggle.

2.2 Decryption

The decryption process will involve the exact inverse, that is

1. Flipping a bit specified by the second key
2. Subtraction of the addition key.

2.3 Specifications

You are required to use the input/output and string storage features (syscall, .asciiz) to print out prompts asking the user for the necessary data and then read in the data. The user should input:

1. A single letter, either ‘e’ or ‘d’ depending on whether you wish to either encrypt or decrypt the text you enter. You may assume that these are always entered in lower case, though you can try to make it case insensitive as an option.
2. The addition key, a single integer between 0 and 9
3. The toggle key, a single integer between 0 and 7 (since ASCII codes are 8-bit values). The bits are numbered using the standard convention. Bit 0 is the LSB and bit 7 is the MSB.
4. Your code must generate an **Exception** if the addition key or the toggle key is invalid.
5. The text to encrypt or decrypt as a stream of characters. You should stop taking in character input when you hit the Enter key.

2.4 Example display on console

(your prompts should be exactly as follows)

```
Enter ‘e’ or ‘d’ to select encrypt or decrypt:     e
Enter addition key:         4
Enter bit toggle key:       1
Enter text to encrypt:      Hello World!
Encrypted text:             Nkrrq&Yqtrj’
```

3 Example encryption and decryption

3.1 Encryption:

Consider the letter ‘H.’ This is how it gets encrypted to ‘N.’

```
ASCII - 0x48      0100 1000  #This is the 8-bit code for H
Add key 4         0100 1100  #0x48+4=0x4C.  The sum is shown here.
Toggle bit 1      0100 1110  #Bits numbered using standard convention (7:0)

Encrypted value for 0x48 is 0x4E    #ASCII code for the letter N
```

3.2 Decryption:

Consider the letter ‘N’. This example shows the steps to decrypt it back to ‘H.’

```
ASCII - 0x4E     0100 1110
Toggle bit 1     0100 1100
Subtract key 4   0100 1000  #0x4E - 4 = 0x48

Decrypted value for 0x4E is 0x48  #ASCII code for the letter H
```

Store the resulting string (after encryption or decryption) and print it out to the console.

4 Hints for Toggling/Inverting a Bit

Toggling a single bit can be done in 2 ways

4.1 Toggle with XOR

1. The exclusive-or function: `xor $t2,$t2,$s3`

XOR-ing a bit with a 1 flips the bit. To toggle a given bit in a number (a register), XOR it with a number (a register) that has 0s in all but the bit position you wish to toggle.

e.g.

```
To toggle bit 4 in   01100110
XOR with             00010000
to get               01110110
```

4.2 Toggle with AND

2. Masking

Another way to bit toggle is to use "masking" using the AND operation.

First determine whether the bit to toggle is 1 or 0. If it is a 1, make it a 0 by ANDing with the appropriate mask. If it is a 0, make it a 1 by ORing with the appropriate mask.
e.g.

```
To toggle bit 4 in    01100110
AND with              00010000
to get                00000000
```

The result is all zeros, so bit 4 must have been a zero. In the original, change bit 4 to a 1, using OR, to get 01110110. If the result is nonzero, bit 4 must have been a 1.

4.3 Directives

.asciiz - Store the string in the Data segment and add null terminator
.byte - Store the listed value(s) as 8 bit bytes

4.4 SYSCALL

A number of system services, mainly for input and output, are available for use by your MIPS program. They are described in the table found under the Syscalls tab of the MARS help page. MIPS register contents are not affected by a system call, except for result registers as specified in the help page.
How to use SYSCALL system services:

1. Load the service number in register \$v0.
2. Load argument values, if any, in \$a0, \$a1, \$a2, or \$f12 as specified.
3. Issue the SYSCALL instruction.
4. Retrieve return values, if any, from result registers as specified.

Example: display the value stored in \$t0 on the console

```
li  $v0, 1           # service 1 is print integer
add $a0, $t0, $zero  # load desired value into argument register $a0
syscall
```

4.5 Sample Code

```
.data
#pre-made prompt messages
Prompt_encDec:           .asciiz "type ‘e’ to encrypt or ‘d’ to decrypt: "
Prompt_ChooseBitToggle:  .asciiz "choose a bit to toggle (0-7): "
Prompt_AddKey:           .asciiz "choose an addition key: "
Prompt_AskForText:       .asciiz "enter text: "

#inputs
INBUFFER:        .byte 0xa:100  #filling memory with this to make it easy to see

#outputs
OUTBUFFER:       .byte 0xb:100 #filling memory with this to make it easy to see

.text

#prompt user to input ’e’ or ’d’
li $v0,4
la $a0,Prompt_encDec
syscall

#read encrypt or decrypt choice
li $v0,12          #set syscall to read char
syscall
```

4.6 Rubric

```
stu:
Proj 3  Encrypt and Decrypt Simple Strings
==========================================

Rubric

1. Prompts user for input             (1 pts)
2. Accepts all user input commands       (1 pts)
3. Encrypts  (3 pts)
4. Decrypts  (2 pts)
5. Modular: effective use of functions:  (2 pts)
6. Generates an Exception if entry invalid  (1 pt)
7. comments, format/ease of reading/understanding code, white
   space, consistent indentation          (1 pts)

Programs that don't assemble are likely
to not get any credit.
*********************************************



Points for above:  (max 11)
-----------------
1. Prompts /1
2. Accepts input commands /1
3. Encrypts /3
4. Decrypts /2
5. Modular /2
6. Exception      /1
7. comments, format /1

total:  xx/11



Instructor Comments:
=================
```