**Assembly Programming HW4**

Assembly Programming (CSE3030)

(Spring 2020)

7th June, 2020

Instructor: Prof. Youngjae Kim

**Total point for HW4 is 60 points**

**TASK1 [15 points]** Make a program that gets a plain text and a key as input. Use the key to encrypt and decrypt the plain text by rotating each character of the key against a corresponding character in the message. Write a procedure that performs encryption.

**\*rotate : bitwise rotate instruction**

**\*character : fits in 8-bit**

NOTE:

1. The length of plain text does not exceed 40 characters.
2. ‘Enter’ character (<ent>) is not counted in string length.
3. The size of each key element does not exceed 1 byte and the length of key does not exceed 10 integers.
4. The negative value of key indicates a rotation to left and a positive value indicates a rotation to right.
5. If the length of key is less than plain text, repeat the key as many times as necessary until all plain text is translated.

Ex. Plain text : “Assembly”, Key : -2, 4, 1, 0, -3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | s | s | e | m | b | l | y |
| 2 Rleft | 4 Rright | 1 Rright | 0 Rotate | 3 Rleft | 2 Rleft | 4 Rright | 1 Rright |

1. Print out the original text, encrypted text, and decrypted text once the user enters a text and a key.
2. Input ‘Enter’ character (<ent>) to finish the program.

|  |
| --- |
| C:\>s171234↲  Enter a plain text : This is the secret message↲  Enter a key : -2 4 1 0 -3↲  Original Text : This is the secret message  Encrypted Text : (print message encrypted by given key)  Decrypted Text : This is the secret message  Enter a plain text : I am your father↲  Enter a key : 1 -2 3 -4 5 -6 7 -8↲  Original Text : I am your father  Encrypted Text : (print message encrypted by given key)  Decrypted Text : I am your father  Enter a plain text : ↲  Bye!  C:\> |

**TASK2 [15 points]** Make a program that functions as a simple Boolean calculator for 32-bit integers. It should display a menu that asks the user to make a selection from the following list:

1. x AND y
2. x OR y
3. NOT x
4. X XOR y
5. Exit program

When the user makes a choice, call a procedure that displays the name of the operation about to be performed. You must implement this procedure using the *Table-Driven Selection* technique, shown in Section 6.5.4 in textbook.

For the Boolean calculator, students have to implement the following procedures:

* AND\_op: Prompt the user for two hexadecimal integers. AND them together and display the result in hexadecimal.
* OR\_op: Prompt the user for two hexadecimal integers. OR them together and display the result in hexadecimal.
* NOT\_op: Prompt the user for a hexadecimal integer. NOT the integer and display the result in hexadecimal.
* XOR\_op: Prompt the user for two hexadecimal integers. Exclusive-OR them together and display the result in hexadecimal.

(The Irvine32 library is required for this solution problem.)

NOTE)

1. The program exits only when getting 5 as an input in the operation menu prompt.

2. If a user enters improper values(other than 1~5) in the menu prompt, get the value again.

3. If a user enters improper input as a value of an x(or y), then get the value again.

4. If a user chooses N for the **question “Do you want to change the mode(Y/N)?”**, Keep in the current calculation mode.

|  |
| --- |
| C:\>s171234  1. x AND y  2. x OR y  3. NOT x  4. X XOR y  5. Exit program  Choose Calculation Mode : 1  Enter x : 4A  Enter y : 51  Result of x AND y :  Do you want to change the mode(Y/N)? : Y  1. x AND y  2. x OR y  3. NOT x  4. X XOR y  5. Exit program  Choose Calculation Mode : 2  Enter x : 4A  Enter y : 51  Result of x OR y :  Do you want to change the mode(Y/N)? : Y  1. x AND y  2. x OR y  3. NOT x  4. X XOR y  5. Exit program  Choose Calculation Mode : 7  Choose Calculation Mode : 4  Enter x : 4A  Enter y : 51  Result of x XOR y :  Do you want to change the mode(Y/N)? : N  Enter x : 4A  Enter y : 51  Result of x XOR y :  Choose Calculation Mode : 5  Bye!  C:\>s171234 |

**TASK3 [30 points]** Write a procedure named BitwiseMultiply that multipies any unsigned 32-bit integer by EAX, using only shifting and addition. Pass the integer to the procedure in the EBX register, and return the produce in the EAX register.

Write a short program that calls the procedure and displays the product. (Assume that the produce is never larger than 32 bits.) This is a fairly challenging program to write. One possible approach is to use a loop to shift the multiplier to the right, keeping track of the number of shifts that occur before the Carry flag is set. The resulting shift count can then be applied to the SHL instruction, using the multiplicand as the destination operand. Then, the same process must be repeated until you find the last 1 bit in the multiplier.

NOTE

1. Get User input as a hexadecimal.
2. Print the result as a hexadecimal.
3. below example shows the 21 \* 500.
4. Input ‘Enter’ character (<ent>) to finish the program.

|  |
| --- |
| C:\>s171234↲  Enter a Multiplier : 15  Enter a Multiplicand : 1F4  Product : 00002904  Enter a Multiplier : ↲  Bye!  C:\> |

***If you use arithmetical instructions other than shift and addition, you will get 0 point.***

**Submission:**

* Submission Due Date: 6/14 (Sunday) 11:59 PM (Late : -10% per day)
* You need to submit compressed **three code files**, each corresponds to each task.
* Name the .asm files with the last 6-digits of your student id and an underscore (‘\_’) and the number of the task. (e.g. **171234\_1.asm, 171234\_2.asm, 171234\_3.asm**)
* Compress source code files into **zip format** and name it with your last 6-digits. (e.g. 171234.zip)
* Please submit your assignment under **Assignment4** in Assignment menu in Cyber Campus.

**Grading Policy:**

* If there is an assemble error, you will get 0 point.
* You will get minus one (-1) point for each wrong file name.
* Your program will be tested with different keys and texts and pairs.
* You can get higher score if the code + data size is shorter.
* You will get minus one(-1) point per task if output format is different from example(including line alignment, spaces)
* **Don’t use screen clear function such as Clrscr, (if use, -50%)**