Assembly Programming HW3

Assembly Programming (CSE3030)

(Spring 2020)

4st June, 2020

Instructor: Prof. Youngjae Kim

Task I

Make a program that sorts number pairs (X, Y). The program should sort all pairs based on X values and if two pairs' X values are same, then sort it based on Y values. For sorting, you should differentiate the order on odd index pairs and even index pairs. The odd index pairs should be sorted in ascending order for X values and in descending order for Y values in case of same X values between pairs. The even index pairs should be sorted in descending order for X values and in ascending order for Y values in case of same X values.

Assume that pairs are given like (0x51, 0x13), (0x1, 0x1), (0x2, 0x3), (0x1, 0x5), (0x51, 0x21), (0x2, 0x1), (0x1, 0x2).

1st index : (0x51, 0x13)
2nd index : (0x1, 0x1)
3rd index : (0x2, 0x3)
4th index : (0x1, 0x5)
5th index : (0x51, 0x21)
6th index : (0x2, 0x1)
7th index : (0x1, 0x2)

The odd index pairs are (0x51, 0x13), (0x2, 0x3), (0x51, 0x21), (0x1, 0x2). They will be sorted to (0x1, 0x2), (0x2, 0x3), (0x51, 0x21), (0x51, 0x13). The even index pairs are (0x1, 0x1), (0x1, 0x5), (0x2, 0x1). They will be sorted to (0x2, 0x1), (0x1, 0x1), (0x1, 0x5). Therefore, the pairs will be sorted like below at last.

(0x1, 0x2)
(0x2, 0x1)
(0x2, 0x3)
(0x1, 0x1)
(0x51, 0x21)
(0x1, 0x5)
(0x51, 0x13)

Make a file “hw3.inc” and declare following data in hw3.inc. (The values of data will be changed when TAs test your code)

```
LenData DWORD 7
ArrData DWORD 510013h, 10001h, 20003h, 10005h, 510021h, 20001h, 10002h
```

LenData is the number of pairs and ArrData is the given pairs. All pairs are given as double word data type, and their MSB 16 bits are X values and LSB 16 bits are Y values in pair (X, Y). In above, for example, the data 510013h’s X value is 51h, and Y value is 13h (00510013h). In your program, you should print all pairs in the given sequence. After then, you should also print all pairs in sorted sequence like below.

```
C:₩>s171234↲
Before sort : 00510013, 00010001, 00020003, 00010005, 00510021, 00020001, 00010002
After sort : 00010002, 00020001, 00020003, 00010001, 00510021, 00010005, 00510013
Bye!
C:₩>
```

Note:

1. All X and Y of pairs are given in 16bit unsigned hexadecimal integer. That is, X and Y are given together in one double word data type variable.
2. All X are larger than 0 (X > 0).
3. Sort data about X values prior to Y values.
4. Print pairs with WriteHex function in Irvine library.
5. Follow the printing format. (e.g. strings to print, comma, etc)

Task II

Write a procedure named **Get_frequencies** that constructs a character frequency table. Input to the procedure should be a pointer to a string and a pointer to an array of 256 doublewords initialized to all zeros. Each array position is indexed by its corresponding ASCII code. When the procedure returns, each entry in the array contains a count of how many times the corresponding character occurred in the string. For example,

```
.data
target BYTE "AAEBDCFBBC",0
freqTable DWORD 256 DUP(0)
.code
INVOKE Get_frequencies, ADDR target, ADDR freqTable
```

String	A	A	E	B	D	C	F	B	B	C	0
ASCII	41	41	45	42	44	43	46	42	42	43	0

FreqTable	2	3	2	1	1	1	0	0	0	0	0
Index	41	42	43	44	45	46	47	48	49	4A	4B

Note:

1. String will only contain number and. Alphabet(a~z,A~Z)
2. you can use this code when you print array.

```
array DWORD 128 DUP(?)
mov   esi, OFFSET array
mov   ecx, arraySize

L1:
mov   eax, [esi]
call WriteInt
call Crlf
add   esi, 4
loop L1
```

Task III

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
```

Submission:

- Submission Due Date: 6/15 (Tue) 11:59 PM (Late : -10% per day)
- You need to submit compressed **two 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. **201234_1.asm, 201234_2.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 **Assignment3** 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.