Assembly Programming HW1

Assembly Programming (CSE3030)

(Spring 2021)

Instructor: Prof. Youngjae Kim

Task1) Summing the Gaps between Array Values

Write a program with a loop and indexed addressing that calculates the sum of all the gaps between successive array elements. The array elements are doublewords, sequenced in nondecreasing order. So, for example, the array {0, 2, 5, 9, 10} has gaps of 2, 3, 4, and 1, whose sum equals 10. Insert the sum in the **EAX** register, and, at the end of your code, check the value using **DumpRegs**. The size of the array may vary.

Below is an example of how the array would be declared (Other arrays will be used for grading).

Make a file with name "hw1.inc". Declare array1 in hw1.inc.

(The values in the array will be changed by TA when grading your code)

```
array1 DWORD 0, 2, 5, 9, 10
```

<hw1.inc file example>

Include the hw1.inc file in your program as below:

```
. . .

.data
INCLUDE hw1.inc
. . .

.code
. . .
call DumpRegs
exit
. . .
```

(Assume there is no overflow and the array is variable in size)

Task2) Fibonacci Numbers

Write a program that uses a loop to calculate the n^{th} $(n > 2)$ value of the Fibonacci number sequence, described by the following formula:

$$Fib(1) = 1, \qquad Fib(2) = 1, \qquad Fib(n) = Fib(n-1) + Fib(n-2)$$

Insert the n^{th} Fibonacci number in the **EAX** register, and, at the end of your code, check the value using **DumpRegs**.

Below is an example of how the array would be declared (Other values for n will be used for grading).

Make a file with name "hw2.inc". Declare fib in hw2.inc and include the file in your program as shown in **Task1)**.

(Again, the value of fib will be changed by TA for grading purposes)

```
fib DWORD 7
```

<hw2.inc file example>

(Assume there is no overflow and n is larger than 2)

Task3) Exponential Power

Make a file with name "hw3.inc". Declare the following data in hw3.inc (The values of data will be changed when TAs test your code)

```
X   DWORD   8
Y   DWORD   15
```

<hw3.inc file example>

Include the above file in your program as shown in Task1).

Make a program to calculate the value of X^Y(X to the power of Y) and Y^X(Y to the power of X). Store each result into EAX and EBX respectively and then print contents of the registers using DumpRegs (assume there is no overflow).

Task4) Copy a String in Reverse Order

Write a program with a loop that copies a string from **source** to **target**, reversing the character order in the process, and print the **target** string to the console.

Make a file "hw4.inc" and declare the two string variables **source** and **target**.

```
source BYTE "This is the source string", 0
target BYTE SIZEOF source DUP ('#')
```

Contents of the **source** string will be changed when grading your code. Include the hw4.inc file in your program.

The desired output for the string above should look like this:

```
gnirts ecruos eht si sihT
```

You can write a null-terminated string to the console using the **WriteString** function (provided by the Irvine library) by giving the **offset** of the string to the **EDX** register, as shown below:

```
mov edx, OFFSET target
call WriteString
```

Instructions:

- Place the **Irvine library directory at C:₩**. Set the additional library directory as C:₩Irvine.
- Do not declare memory data in the .inc files other than those mentioned above. If you need additional variables for your program, declare them in your .asm source code.
- Try to reduce the number of instructions in the code.

Submission:

- Submission Due Date: 04/16 (Fri) 11:59 PM (Late : -10% per day, up to 3 days)
- Make a separate source code file for each task, and compress them into one.
- Name the four .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, 171234_4.asm**)
- Compress source code files into **zip format** and name it with the last 6-digits of your student id. (e.g. 171234.zip)
- Please submit your assignment in the **Assignment1** post under the Assignment menu of Cyber Campus.

Grading Policy:

- If there is an **assemble error**, you will get 0 point.
- You are only allowed to use instructions covered till now in class. For example, instructions such as SHL, MUL, and IMUL are not allowed.
- You will get minus one (-1) point for each wrong file name.
    - Your program will be tested with different .inc files.