CSCI-2525

Delivery Method

Implement the following requirement in one MASM program and upload your program file and output file to Canvas prior to the due date and time. The program filename will be **HW_2_*lastname*.asm** and output file name will be ***HW_2_lastname.pdf.*** You may do a screen capture of the output command window to create output PDF file.

Point Value

This assignment is worth 20 points. The rubric is listed on Canvas.

Program Objectives

The objective of this assignment is to build an ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired need. (ABET-c)

Problem Description

Using the template provided below, write an assembly program that accomplishes the following requirements in 4 parts.

Note that you may use "***call DumpReg***" to capture the registers information and "***call WaitMsg***" to pause the execution of your program. Both procedures are from Irvine32 library.

1) Part 1:

1) Write a single statement that computes the product $\prod_{i=2}^{7} i$ where $i \in \mathbb{Z}$. You are calculating the product from 2 – 7

   Hint: To accomplish the task, you need to calculate the product of i between 2 and 7 inclusively. This product can be expressed mathematically as: $2 \times 3 \times 4 \times 5 \times 6 \times 7$

2) **Specifications:**

   a) Place the results in EAX, AX, or AL. You must choose the correct (the smallest possible) size of the register, remember to consider efficient use of available resources.

   b) To compute the product, you ***must use one or more constant expressions*** that computes the value. This will be accomplished in one instruction. No loops or use of anything beyond Chapter 4 in the book may be used. No use of MUL, MULI, … instructions

   c) Note that the register must be zeroed out before the result is stored.

2) Part 2:

For this part of problem, feel free to pick any constant numbers of your choice.

1) Write a short block of statements that cause the EBX register to set the carry flag. Note you are **NOT** allowed to use the ***STC*** instruction.

2) Write a short block of statements that causes the ECX register to set the overflow flag.

3) Specifications:

   a) Make sure no other computations affect the outcome of the registers.
   b) You may use immediate values in your statements.
   c) The registers must be zeroed out before use.
   d) You may use the instructions ADD, SUB, MOV. For more information on these instructions, use appendix B.
   e) **Do not use LOOP instructions.**

3) Part 3:

1) Using directives for creating symbolics, write a **single statement** that computes the number of seconds in a day. Note that you must do the computation in the data segment.

2) **Specifications:**

   a) Place the result in the EDX register.
   HINT: Think about how memory is stored and how you can address memory. FYI there is a slide in the notes that explicitly explains how to do this.

   b) The symbolic constant name may be **SECONDS_IN_DAY**.

   c) The EDX register should be zeroed out before it is used.

   d) Make sure the statement uses the symbolics to the fullest extent; that is, SECONDS_IN_DAY will be the only expression on the instruction line. No other mnemonics, or operands.

4) Part 4:

1) Calculate `A = (A - B) + (C - D)`. Where A=`+781379d (signed)`, `B=010101101011b (signed), C= FAC8h(unsigned) and D=-57d (signed)`. The variables must be correctly initialized in the data segment.

2) **Specifications:**

   a) You must ensure your registers are cleared before starting this part.
   b) Ensure that you are using registers (or parts of registers) of the appropriate size.
   c) You will calculate the value of A in the following sequence.
      - `A – B`
      - `C – D`
      - `(A – B) + (C – D)`
   d) You must declare 4 separate variables (A, B, C, D) of the appropriate size to hold the values given above.
   e) There is no memory-to-memory operation. You must use registers only to perform necessary operations.
   f) Store the final result in EAX, then move the value to variable A.
   g) Then write '***call WriteInt***' to display the result in EAX. This will require to use Irvine32 library

Additional information:

A program template is provided below.

Ensure that the text is entered in the ASCII text mode without any control chracters.

```
TITLE pa2.asm
;// Header comment block as shown in lecture notes
INCLUDE Irvine32.inc
INCLUDE Irvine32.lib
.data
;//{ your variables are be defined here}

.code main PROC ;//{executable code here} call DumpRegs

;// you may use this line of code as necessary to show the contents of the
registers and flags.


exit
main ENDP ;// end of main procedure

END main ;// end of source code
```