Vivian Do
CSC 3210 | CRN 90913
Oct 14 2021

CSC 3210 – Assignment #2
Fall 2021
Due date: 10/14/21 11:59 PM

Objective: Learn memory organization/layout, data transfer concepts and instructions, direct memory access, memory allocation.

Requirements:

1. **(5 points) Implement the following expression in assembly language:**

`AX = (val3 + 7) - (val2 + val1) + (5/3)*7` - Assume that `val1`, `val2`, and `val3` are 16-bit integer variables

- You need to implement the expression the way it is provided, you cannot do any reduction on the expression while implementing it.
- Initialize `val1` with 12 (decimal), `val2` with 9 (decimal), and `val3` with 2 (decimal) - You are ONLY allowed to use **16-bit registers**.
- Use ONLY **`mov, add, sub`** instructions whenever needed.
- Use the debugger to verify your answer.

- **Submit the following:**
  - Save your source code using your last name, `Lastname1.asm` and upload the `Lastname1.asm`
  - Screenshot (showing the code and register window) of `AX` register contains the correct result.

```
Registers
EAX = 0000FFFB EBX = 00000015 ECX = 00F81005 EDX = 00F81005 ESI = 00F81005 EDI = 00F81005 EIP = 00F81039 ESP = 00E2FB18 EBP = 00E2FB24 EFL = 00000282

OV = 0 UP = 0 EI = 1 PL = 1 ZR = 0 AC = 0 PE = 0 CY = 0
```

Do1.asm

```
1  ; Vivian Do
2  ; Class: CSC 3210
3  ; Assignment 2 Question 1
4  ; This program will compute (val3 + 7) - (val2 + val1) + (5 / 3) * 7 for the AX register
5
6  .386
7  .model flat, stdcall
8  .stack 4096
9  ExitProcess proto, dwExitCode:dword
10
11 .data
12     val1 SWORD 12d              ; initializing varialbes as signed 16-bit integers
13     val2 SWORD 9d
14     val3 SWORD 2d
15
16 .code
17 main proc
18     mov eax, 0h                 ; putting zeros in the register value
19     mov ebx, 0h
20
21     mov ax, val3                ; computing val3 + 7 in ax register
22     add ax, 7d
23
24     mov bx, val2                ; computing val2 + val1 in bx register and then subtracting it to ax register
25     add bx, val1
26     sub ax, bx
27
28     add ax, ((5d / 3d) * 7d)    ; computing (5 / 3) * 7 and then adding it to ax register
29
30     invoke ExitProcess, 0  ≤1ms elapsed
31
32 main endp
33 end main
```

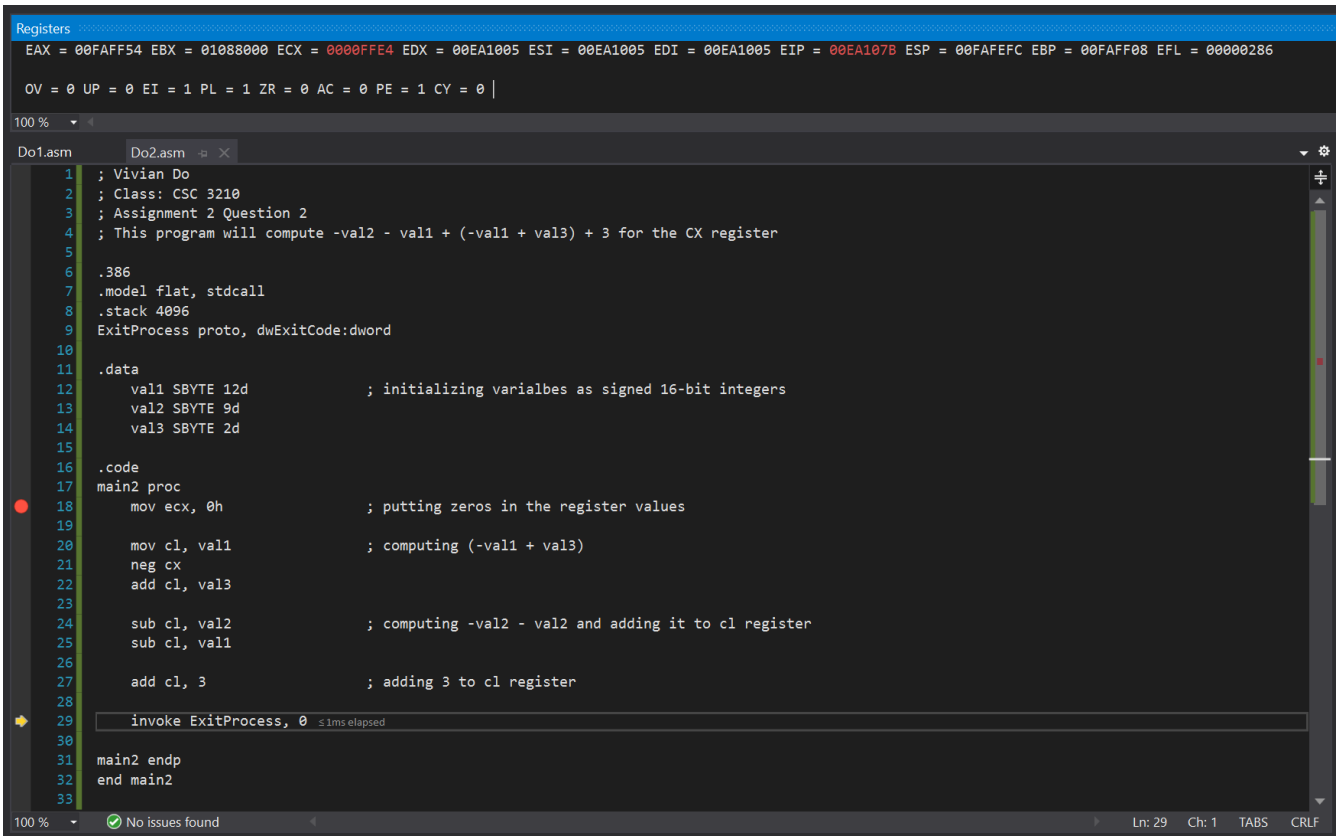
100 % No issues found Ln: 30 Ch: 1 TABS CRLF

2. **(5 points) Implement the following expression in assembly language:**

`CX = -val2 - val1 + (-val1 + val3) + 3` - Assume that `val1, val2,` and `val3` are **8-bit integer variables**

- You need to implement the expression the way it is provided, you cannot do any reduction on the expression while implementing it.
- Initialize `val1` with 12 (decimal), `val2` with 9 (decimal), and `val3` with 2 (decimal) - You are NOT allowed to **update the values** stored in `val1, val2,` and `val3` - You are only allowed to use **16-bit registers** to hold intermediate results, whenever needed. - Use `mov, add, sub, movzx, movzx,` or `neg` instructions whenever needed.
- Use the debugger to verify your answer.

- **Submit the following:**
  - Save your source code using your last name, `Lastname2.asm` and upload the `Lastname2.asm`
  - Screenshot (showing the code and register window) of `CX` register contains the correct result.



3. (3 points) True/False

(2.1) The instruction, `var BYTE '?'`
The above instruction declares a variable named `var` and keeps in uninitialized.
Flase - the var variable in byte size does not stay uninitialized.

(2.2) The instruction, `var DWORD "ABCD"`
stores the string 'ABCD' in to variable named `var`
True - DWORD, or doubleword, will directly store a complete string,

(2.3) The instruction, `var BYTE "ABCD"`

stores the characters ‘A’,’B’,’C’,’D’ in an array of characters named `var`

True - all four characters are stored in (4) byte sizes

4. (2 points) Declare a variable:

```
Var1 DWORD 2 DUP (6 DUP ( 3 DUP (?) ) )
```

What is the total size of the array `Var1` ? Explain your answer.

The total size of the Var1 array is 144 bytes.

Let 1 DWORD = 4 Bytes

3 DUP (?)			
	?	?	?

= 3 DWORD = 3 * 4= 12 bytes

6 DUP (3 Dup (?))			
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?

= (6 * 3) DWORD = (6*3) * 4 = 72 bytes

2 DUP (6 DUP (3 DUP (?)))			
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?
	?	?	?

= (2 * 6 * 3) DWORD = (2*6*3) * 4 = 144 bytes

The first DUP *(3 DUp (?))* has an array of 3 x 1 ?’s, and since each ? is 4 bytes, the array is 12 bytes. The second DUP *(6 DUP (3 DUP (?))* has an array of 3 x 6 ?’s, and since each ? is 4 bytes, the array is 72 bytes. The last DUP *(2 DUP (6 DUP (3 DUP (?)))* has an array of 3 x 12 ?’s, and since each ? is 4 bytes, the array is 144 bytes.

Therefore, the total size of Var1 is 144 bytes.

Note:

- **Submit** your source code by **only** uploading **.ASM file** using **iCollege** in the respective assignment dropbox: ▪ Lastname1.ASM, Lastname2.ASM
- **Put the following information as Comment header** for .ASM files:

  Student: Full name

  Class: CSC3210

  Assignment#: 2

  Description: This program …….
- Follow the program standards as presented in your book. Pay more attention to code comments and consistent indentation.