

CSC 3210 – Assignment #3
Fall 2021
11/03/21 11:59 PM

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

Requirements:

1. (5 points) Write an assembly program to compute the following expressions

- Create a **DWORD array named 'z'** of size 3 using DUP operator. Leave the array 'z' uninitialized. You can denote the items in the array as $[z_0, z_1, z_2]$, where z_0 is the first item, z_1 is the second item, z_2 is the third item
- Update each array item using the following expressions.

$$\begin{aligned}z_0 &= x + 130 \\z_1 &= y + x - z_0 \\z_2 &= r + x - 13\end{aligned}$$

- Where x, y, r are **16-bit integer memory variables**.
- $x = 10, y = 15, r = 4$
- Use mov, movzx, movsx, add, sub instructions only.
- (hint: Do not alter the value of x, y and r during the computation. Transfer them to appropriate registers to do computation)
- At the end, open memory window to see the variable z stored in memory (little endian format).
- If you code correctly, $z_0 = 140$ in decimal, $z_1 = -115$ in decimal, $z_2 = 1$ in decimal
- Use the debugger to verify your answer.
 - o **Submit the following:**
 - Rename the asm file using your last name as Lastname1.asm
 - Screenshot of the code and memory window showing the content of the variable z (little endian format).

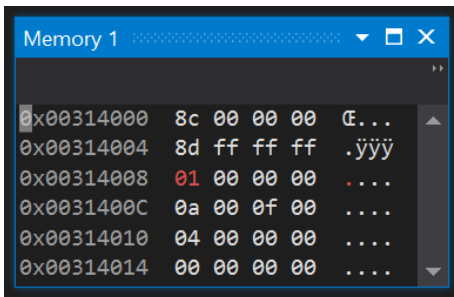
Screenshot(s):

```

Registers
EAX = 0000000A EBX = 0000000F ECX = 00000004 EDX = 0000000C ESI = 00311005 EDI = 00311005 EIP = 0031105F ESP = 0117FA48 EBP = 0117FA54 EFL = 00000202

100 %
Registers Memory 1
Dof.asm
1 ; Vivian Do
2 ; Class: CSC 3210
3 ; Assignment 3 Question 1
4 ; This program will create an uninitialized array of DWORD elements being updated with various values
5
6 .386
7 .model flat, stdcall
8 .stack 4096
9 ExitProcess proto, dwExitCode:dword
10
11 .data
12     z DWORD 3 DUP (?) ; creating given DWORD array
13     x WORD 10 ; initializing x, y, and r with their given values
14     y WORD 15 ; as 16-bit integers
15     r WORD 4
16
17 .code
18 main proc
19     movzx eax, x ; moving these integers in different registers
20     movzx ebx, y
21     movzx ecx, r
22
23     mov [z + 0], eax ; calculating z_0 = x + 130
24     add [z + 0], 130 ; z_0 = 8Ch (140d)
25
26     mov [z + 4], ebx ; calculating z_1 = y + x - z_0
27     add [z + 4], eax
28     mov edx, [z + 0]
29     sub [z + 4], edx ; z_1 = FF8Dh (-115d)
30
31     mov [z + 8], ecx ; calculating z_2 = r + x - 13
32     add [z + 8], eax
33     sub [z + 8], 13 ; z_2 = 1h (1d)
34
35     invoke ExitProcess, 0 ; 1ms elapsed
36
37 main endp
38 end main
39
100 % No issues found Ln: 39 Ch: 1 TABS CRLF

```



2. (5 points) Use a loop instruction with indirect addressing to solve the problem.

- Do not copy the elements to any other array.
 - **Use the LOOP and XCHG instruction.**
 - The input array, *inputStr* contains elements: "A", "B", "C", "D", "E", "F", "G", "H".
 - The array's elements after running the program should look like: "H", "G", "F", "E", "D", "C", "B", "A".
- Submit the following:
- Rename the asm file using your last name as Lastname2.asm
 - Screenshot of the code and memory window showing the content of the variable *inputStr*.

Screenshots:

Screenshot:

The screenshot shows a debugger window with the 'Registers' tab selected. At the top, the register values are displayed: EAX = 00000D17, EBX = 00000307, ECX = 00000408, EDX = 00000102, ESI = 00E7100A, EDI = 00E7100A, EIP = 00E710C0, ESP = 0099F854, EBP = 0099F860, EFL = 00000206. Below this, the 'Memory' tab is active, showing the assembly code for 'Do3.asm'. The code includes comments and instructions for setting up the program, loading data, and calculating the sum of words in the data segment. The instruction at address 00E710C0 is 'invoke ExitProcess, 0', which is highlighted with a red dot. The status bar at the bottom indicates '1ms elapsed'.

```
Registers
EAX = 00000D17 EBX = 00000307 ECX = 00000408 EDX = 00000102 ESI = 00E7100A EDI = 00E7100A EIP = 00E710C0 ESP = 0099F854 EBP = 0099F860 EFL = 00000206

100 %
Memory 1 Registers
Do3.asm
1 ; Vivian Do
2 ; Class: CSC 3210
3 ; Assignment 3 Question 3
4 ; This program will find the sum of the words in the data segment
5
6 .386
7 .model flat, stdcall
8 .stack 4096
9 ExitProcess proto, dwExitCode:dword
10
11 .data
12 ; creating given value as QWORD (64-bit integer)
13 qVal QWORD 0506030704080102h
14
15 .code
16 main proc
17     mov eax, 0 ; putting zeros in the register values
18     mov ebx, 0
19     mov ecx, 0
20     mov edx, 0
21
22     mov ax, WORD PTR [qVal + 6] ; loading 0506h in ax
23     mov bx, WORD PTR [qVal + 4] ; loading 0307h in bx
24     mov cx, WORD PTR [qVal + 2] ; loading 4080h in cx
25     mov dx, WORD PTR [qVal] ; loading 0102h in dx
26
27     add ax, bx ; adding up the values of each register into ax
28     add ax, cx
29     add ax, dx ; EAX = 00000D17h
30
31     invoke ExitProcess, 0 1ms elapsed
32
33 main endp
34 end main
35
```

The screenshot shows a debugger window with the 'Memory' tab selected. The address 0x00E74018 is entered in the address field. The memory dump shows the following values:

Address	Value
0x00E74018	02 01 08 04 07 03 06 05
0x00E74020	00 00 00 00 00 00 00 00
0x00E74028	00 00 00 00 00 00 00 00
0x00E74030	00 00 00 00 00 00 00 00
0x00E74038	00 00 00 00 00 00 00 00
0x00E74040	00 00 00 00 00 00 00 00

The screenshot shows a debugger window with the 'Memory' tab selected. The address 0x00E710C0 is entered in the address field. The memory dump shows the following values:

Address	Value
0x00E710C0	6a 00 e8 0f 00 00 00 cc j.è....i
0x00E710C8	cc cc cc cc cc cc cc cc iiii
0x00E710D0	cc cc cc cc cc cc ff 25 iiii%
0x00E710D8	00 50 e7 00 cc cc cc cc .Pc.iii
0x00E710E0	cc cc cc cc cc cc cc cc iiii
0x00E710E8	cc cc cc cc cc cc cc cc iiii

??

Note:

- **Comment header** for .ASM files:

Student: Full name

Class: CSC3210

Assignment#: 3

Description: This program

- Follow the program standards as presented in your book. Pay more attention to code comments and consistent indentation.
- Create a new project for every question. Do not use one project with multiple .asm files.