

CSC 3210
Computer Organization and Programming
Lab 6
Answer Sheet

Student Name: Hakan Gunerli
#002504797
Section: 014

Lab 6(a)

Debug through each line of instructions.
Take screenshot that includes code and register window.
Record the register content.
and explain the register contents. (4 points)

Line number: 7,8,9

Instruction:

Xval SDWORD 26
Yval DWORD 30
Zval DWORD 40

Register values:

eax, 0000001A
ecx, 0000001E
edx, 00000028

Screenshot:

The screenshot shows a debugger interface with two windows. The top window is the 'Registers' window, displaying the current values of various registers: EAX = 0000001A, ECX = 0000001E, EDX = 00000028, ESI = 00491005, EDI = 00491005, EIP = 00491021, ESP = 00EFF908, EBP = 00EFF9E4, EFL = 00000246. The bottom window is the 'Disassembly' window, showing the assembly code for the current function. The instructions for lines 7, 8, and 9 are highlighted: line 7: Xval SDWORD 26, line 8: Yval DWORD 30, and line 9: Zval DWORD 40. The code also includes a .data section for these variables and a .code section for the main function.

Explanation:

Initializing values for xval yval and zval and moving the values to eax, ecx, and edx registers.

Line number: 15

Instruction:

add ecx, edx

Register values:

ECX = 00000046

Screenshot:

The screenshot shows a debugger window with the 'Registers' tab selected. The register values are: EAX = 0000001A, EBX = 003A1000, ECX = 00000046, EDX = 00000028, ESI = 00491005, EDI = 00491005, EIP = 00491023, ESP = 0059FD20, EBP = 0059FD2C, EFL = 00000212. Below the registers, the 'Disassembly' tab is active, showing the assembly code for 'main.asm'. The code includes directives like .386, .model flat, stdcall, .stack 4096, and .data. The main procedure starts with mov eax, xval, mov ecx, yval, mov edx, zval, and add ecx, edx. The instruction at line 15 is highlighted, showing a tooltip for 'sub eax, ecx' with a duration of 0.1ms elapsed. The status bar at the bottom indicates 'No issues found'.

```
1 .386
2 .model flat, stdcall
3 .stack 4096
4 ExitProcess proto, dwExitCode:dword
5
6 .data
7     Xval SDWORD 26
8     Yval DWORD 30
9     Zval DWORD 40
10
11 .code
12 main proc
13     mov eax, xval
14     mov ecx, yval
15     mov edx, zval
16     add ecx, edx
17     sub eax, ecx 0.1ms elapsed
18
19     invoke ExitProcess, 0
20 main endp
21 end main
```

Explanation:

The equation could be done in many ways. The order in which I felt comfortable with was to start with the parentheses so I did $ecx + edx$, which is $yval$ and $zval$ and store it at ecx , which then we can subtract from $xval$ which is stored at eax .

Line number: 16

Instruction:

sub eax, ecx

Register values:

EAX = FFFFFFFD4

Screenshot:

The screenshot shows a debugger window with the 'Registers' tab selected. The register values are: EAX = FFFFFFFD4, EBX = 00CA5000, ECX = 00000046, EDX = 00000028, ESI = 00491005, EDI = 00491005, EIP = 00491025, ESP = 00EFF9D8, EBP = 00EFF9E4, EFL = 00000287. Below the registers, the 'Disassembly' tab is selected, showing the assembly code for 'main.asm'. The code is as follows:

```
1 .386
2 .model flat, stdcall
3 .stack 4096
4 ExitProcess proto,dwExitCode:dword
5
6 .data
7     Xval SDWORD 26
8     Yval DWORD 30
9     Zval DWORD 40
10
11 .code
12 main proc
13     mov eax, xval
14     mov ecx, yval
15     mov edx, zval
16     add ecx, edx
17     sub eax, ecx
18
19     invoke ExitProcess, 0
20 main endp
21 end main
```

Explanation:

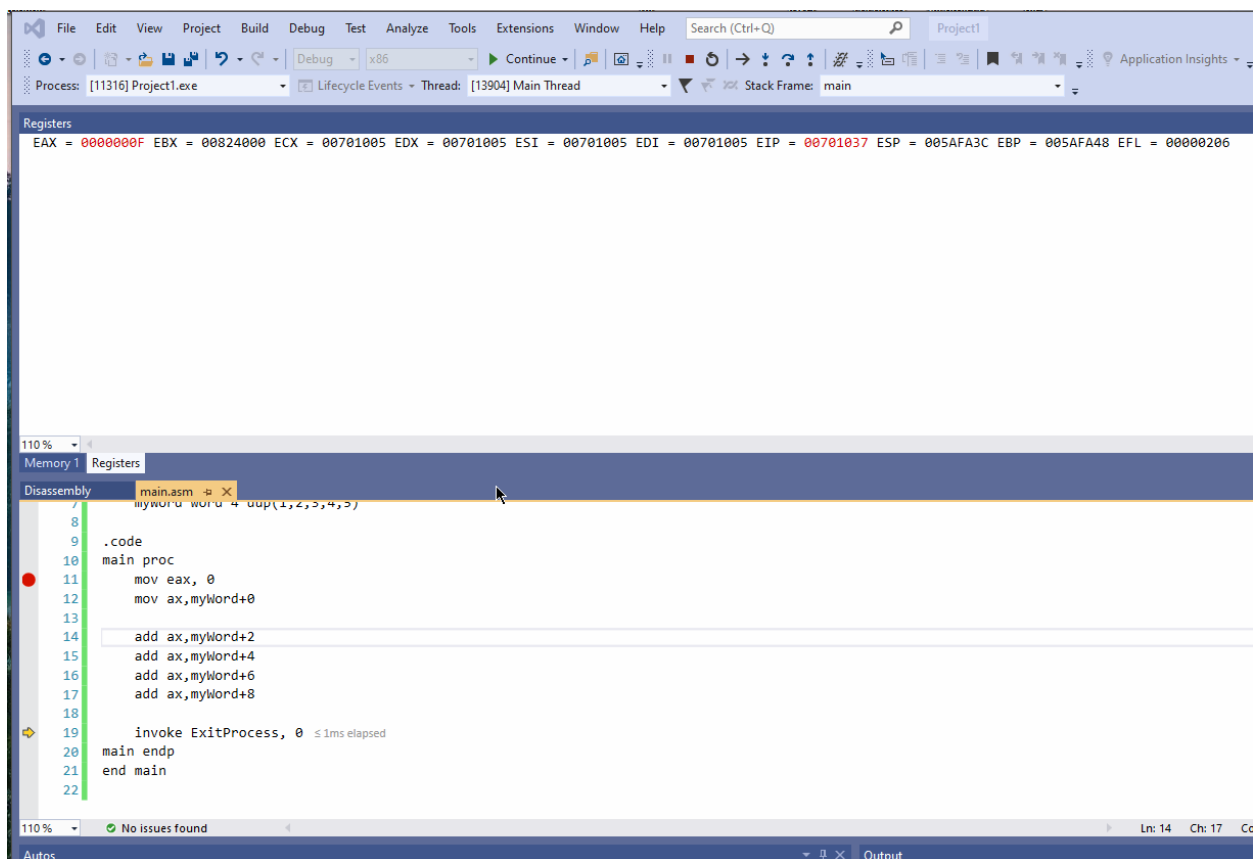
Subtract eax from ecx. Ecx holds the value for yval+zval, and it is being subtracted from xval finally stored at eax.

Lab 6(b)

(1) What is the total size of the myWord array? (1 Point)

The size of the array is 4.

(2) Debug the code until the 'invoke ExitProcess, 0'. Attach screenshot showing the content of AX register. (2 points)



The screenshot shows the Visual Studio IDE with the debugger active. The 'Registers' window at the top displays the state of various registers, including EAX = 0000000F, EBX = 00824000, ECX = 00701005, EDX = 00701005, ESI = 00701005, EDI = 00701005, EIP = 00701037, ESP = 005AFA3C, EBP = 005AFA48, and EFL = 00000206. The 'Disassembly' window shows the assembly code for 'main.asm'. The code includes a declaration of 'myWord' as a 4-element array of 'word' type, followed by a 'main' procedure. The procedure initializes 'eax' to 0, moves 'myWord+0' into 'ax', and then adds 'myWord+2', 'myWord+4', 'myWord+6', and 'myWord+8' to 'ax'. Finally, it invokes 'ExitProcess, 0'. The 'Output' window at the bottom shows 'No issues found'.

```
7 myWord word 4 dup(1,2,3,4,5)
8
9 .code
10 main proc
11     mov eax, 0
12     mov ax, myWord+0
13
14     add ax, myWord+2
15     add ax, myWord+4
16     add ax, myWord+6
17     add ax, myWord+8
18
19     invoke ExitProcess, 0
20 main endp
21 end main
22
```

Lab 6(c):

(1) What is the difference between symbolic constant and variables? (1 point)

Symbolic constant just represents a name, when a constant is initialized however, you cannot change its value.

(2) Debug the code until 'invoke ExitProcess, 0'. Attach the screenshot showing the content of al register. (2 points)

The screenshot shows the Visual Studio Code interface with the following details:

- Registers Window:** Displays the state of various registers. EAX is highlighted in red and contains the value 00000068. Other registers like EBX, ECX, EDX, ESI, EDI, EIP, ESP, EBP, and EFL are also listed with their respective values.
- Disassembly Window:** Shows the assembly code for 'main.asm'. The code includes directives like .386, .model flat, stdcall, .stack 4096, and .data. It defines a string 'myString' and its length. The main procedure starts with 'main proc' and includes instructions like 'mov eax, 0' and 'mov al, myString_length'. A red dot on the left margin indicates the current instruction pointer at line 11.
- Output Window:** Located at the bottom, it shows 'No issues found'.