

CSC 3210

Computer Organization and Programming

Lab 7

Answer Sheet

Student Name: Aparna Mandapaka

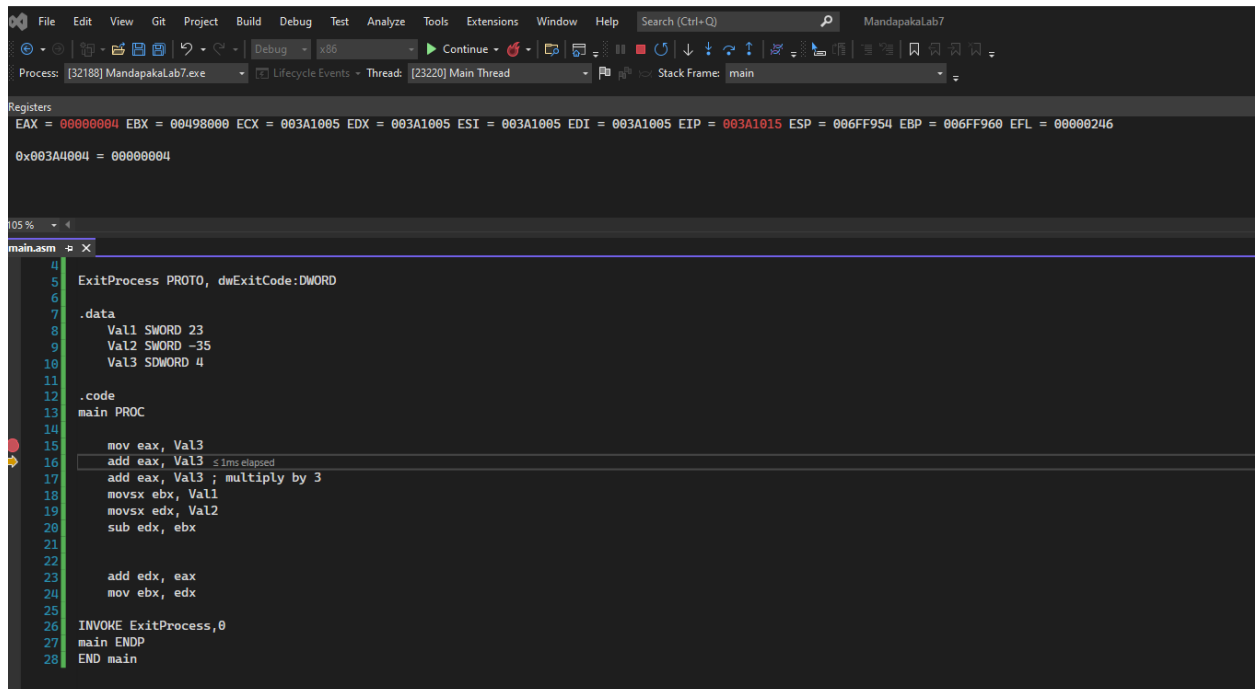
Section:

Debug through each line of code and explain the register content.

Line: 15

Instruction: `mov eax, Val3`

Register value: `EAX = 00000004`



The screenshot shows a debugger window for a process named 'MandapakaLab7'. The 'Registers' pane at the top displays the current state of the CPU registers. The 'EAX' register is highlighted in red and shows the value '00000004'. Below the registers, the assembly code for the 'main' function is displayed. Line 15 is highlighted, showing the instruction 'mov eax, Val3'. The code includes a data section with 'Val1' (23), 'Val2' (-35), and 'Val3' (4), and a code section with various arithmetic and control flow instructions. The 'Stack Frame' pane shows the current frame is 'main'.

```
Registers
EAX = 00000004 EBX = 00498000 ECX = 003A1005 EDX = 003A1005 ESI = 003A1005 EDI = 003A1005 EIP = 003A1015 ESP = 006FF954 EBP = 006FF960 EFL = 00000246
0x003A4004 = 00000004

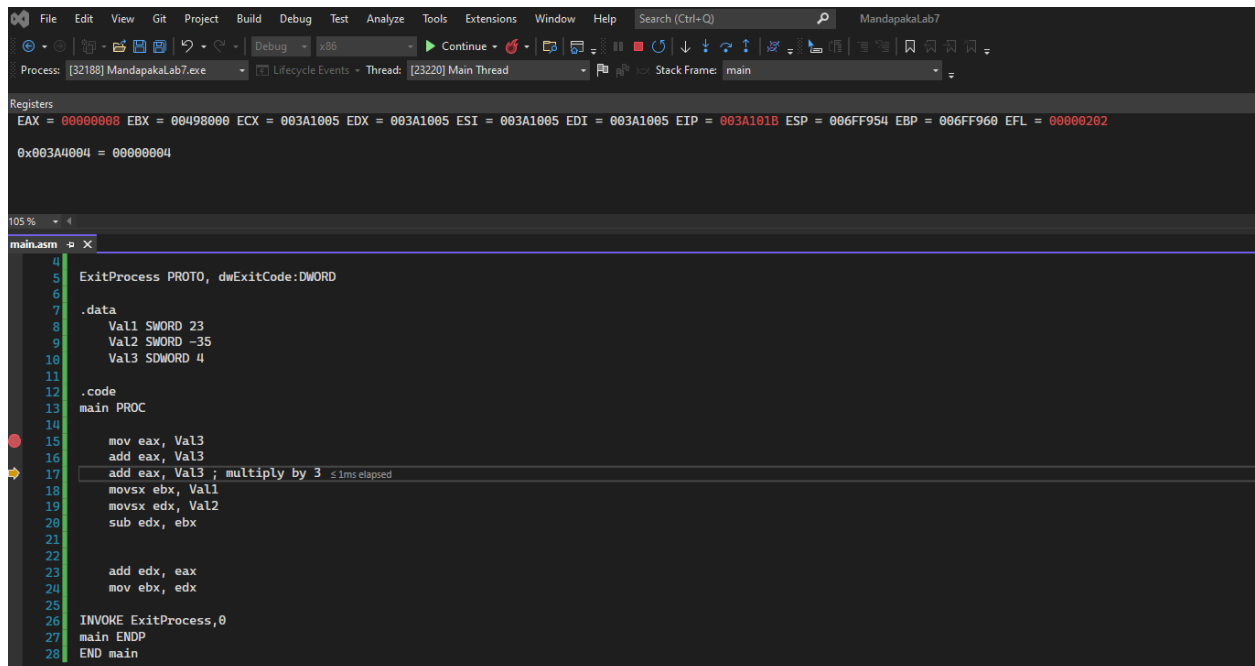
105 %
main.asm
4  ExitProcess PROTO, dwExitCode:DWORD
5
6
7  .data
8      Val1 SWORD 23
9      Val2 SWORD -35
10     Val3 SDWORD 4
11
12  .code
13  main PROC
14
15     mov eax, Val3
16     add eax, Val3 ; 1ms elapsed
17     add eax, Val3 ; multiply by 3
18     movsx ebx, Val1
19     movsx edx, Val2
20     sub edx, ebx
21
22
23     add edx, eax
24     mov ebx, edx
25
26     INVOKE ExitProcess,0
27  main ENDP
28  END main
```

Explanation: In the above-provided instruction we are moving the value of Val3, which is '4' into the eax register, and the eax register displays the number at the end and all the numbers before 4 are 0's and we can treat these 0 values as garbage values, these are here to fill up the space.

Line:16

Instruction: add eax, Val 3

Register value: EAX = 00000008



The screenshot shows a debugger window for a process named 'MandapakaLab7.exe'. The 'Registers' window at the top displays the state of various registers: EAX = 00000008, EBX = 00498000, ECX = 003A1005, EDX = 003A1005, ESI = 003A1005, EDI = 003A1005, EIP = 003A1018, ESP = 006FF954, EBP = 006FF960, and EFL = 00000202. Below the registers, the 'main.asm' window shows the assembly code. Line 16 is highlighted, showing the instruction 'add eax, Val3'. The code includes a data section with Val1 (SWORD 23), Val2 (SWORD -35), and Val3 (SWORD 4), and a code section for the 'main' procedure. The 'main' procedure starts with 'mov eax, Val3' (line 15), followed by 'add eax, Val3' (line 16), and then 'add eax, Val3 ; multiply by 3' (line 17). The instruction at line 17 has a tooltip that says 'multiply by 3' and '≤ 1ms elapsed'. The code continues with 'movsx ebx, Val1', 'movsx edx, Val2', 'sub edx, ebx', 'add edx, eax', 'mov ebx, edx', 'INVOKE ExitProcess,0', 'main ENDP', and 'END main'.

Explanation: In this instruction we are adding the value that is stored in eax with the initial value in Val3, so from the previous step, we have value 4 in the eax register and now we are adding the value 4 from the variable Val3 to the eax register again, which gives us the value of 8.

Line:17

Instruction: add eax, Val3

Register value: EAX = 0000000C

```
4 ExitProcess PROTO, dwExitCode:DWORD
5
6
7 .data
8 Val1 SWORD 23
9 Val2 SWORD -35
10 Val3 SDWORD 4
11
12 .code
13 main PROC
14
15 mov eax, Val3
16 add eax, Val3
17 add eax, Val3 ; multiply by 3
18 movsx ebx, Val1 ; 1ms elapsed
19 movsx edx, Val2
20 sub edx, ebx
21
22
23 add edx, eax
24 mov ebx, edx
25
26 INVOKE ExitProcess,0
27 main ENDP
28 END main
```

Explanation: Here in the eax register we have the value 8 stored in it from previous steps, so now we are adding the value in Val3 to the current value in the eax register, which is 8+4, which gives us 12, in the eax register it displays the letter 'C', which is basically number 12, but it is C in hexadecimal value. So we can see that the eax register displays the added value which is C also 12.

Line: 18

Instruction: movsx ebx, val1

Register value: EBX = 00000017

```
4 ExitProcess PROTO, dwExitCode:DWORD
5
6
7 .data
8     Val1 SWORD 23
9     Val2 SWORD -35
10    Val3 SDWORD 4
11
12 .code
13 main PROC
14
15     mov eax, Val3
16     add eax, Val3
17     add eax, Val3 ; multiply by 3
18     movsx ebx, Val1
19     movsx edx, Val2
20     sub edx, ebx
21
22
23     add edx, eax
24     mov ebx, edx
25
26     INVOKE ExitProcess,0
27 main ENDP
28 END main
```

Explanation: The value in the val1 has been moved to the ebx register.

Line: 19

Instruction: movsx edx, val2

Register value:EDX = FFFFFFFD

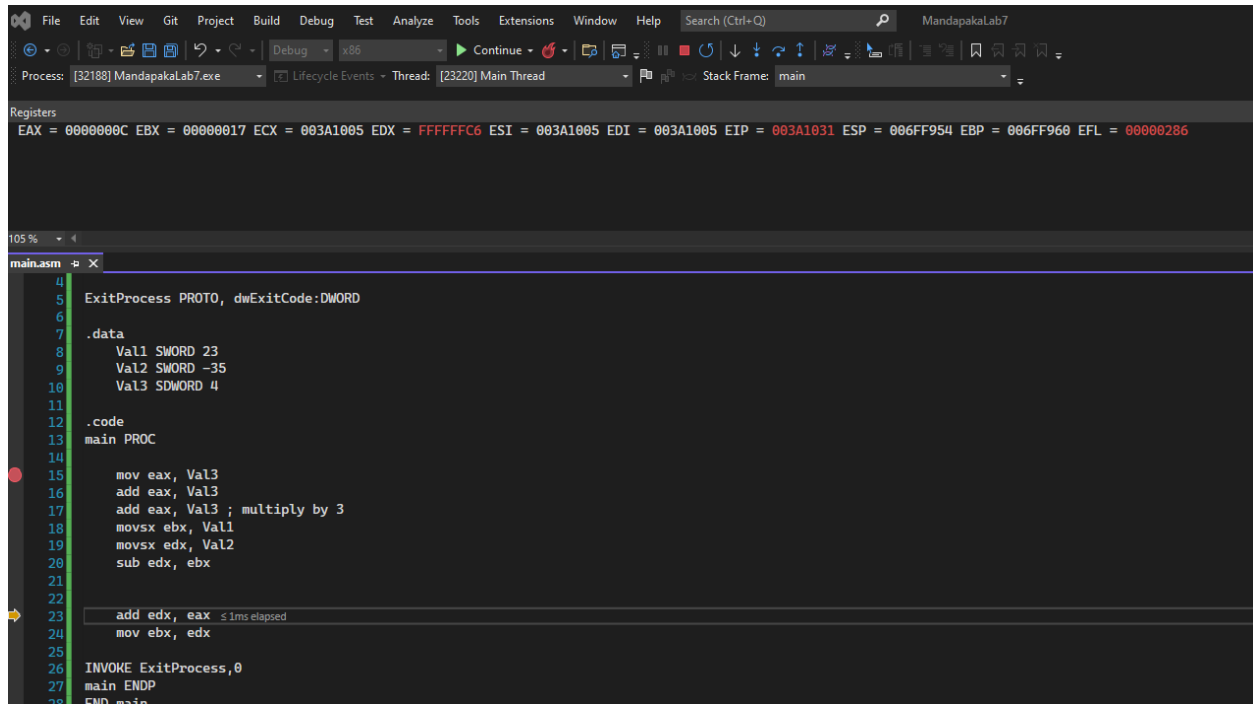
```
4 ExitProcess PROTO, dwExitCode:DWORD
5
6
7 .data
8     Val1 SWORD 23
9     Val2 SWORD -35
10    Val3 SDWORD 4
11
12 .code
13 main PROC
14
15     mov eax, Val3
16     add eax, Val3
17     add eax, Val3 ; multiply by 3
18     movsx ebx, Val1
19     movsx edx, Val2
20     sub edx, ebx
21
22
23     add edx, eax
24     mov ebx, edx
25
26     INVOKE ExitProcess,0
27 main ENDP
28 END main
```

Explanation: here the value of the val2, which is -35 in decimal has been moved to the edx register and we know that all registers work and give hexadecimal values so the value of val2 is -35, which is FFDD in hexadecimal, so the value FFDD has been stored in the edx register.

Line: 20

Instruction: sub edx, ebx

Register value: EDX = FFFFFFFC6



The screenshot shows a debugger window with the following components:

- Registers:** EAX = 0000000C, EBX = 00000017, ECX = 003A1005, EDX = FFFFFFFC6, ESI = 003A1005, EDI = 003A1005, EIP = 003A1031, ESP = 006FF954, EBP = 006FF960, EFL = 00000286.
- Assembly Code (main.asm):**

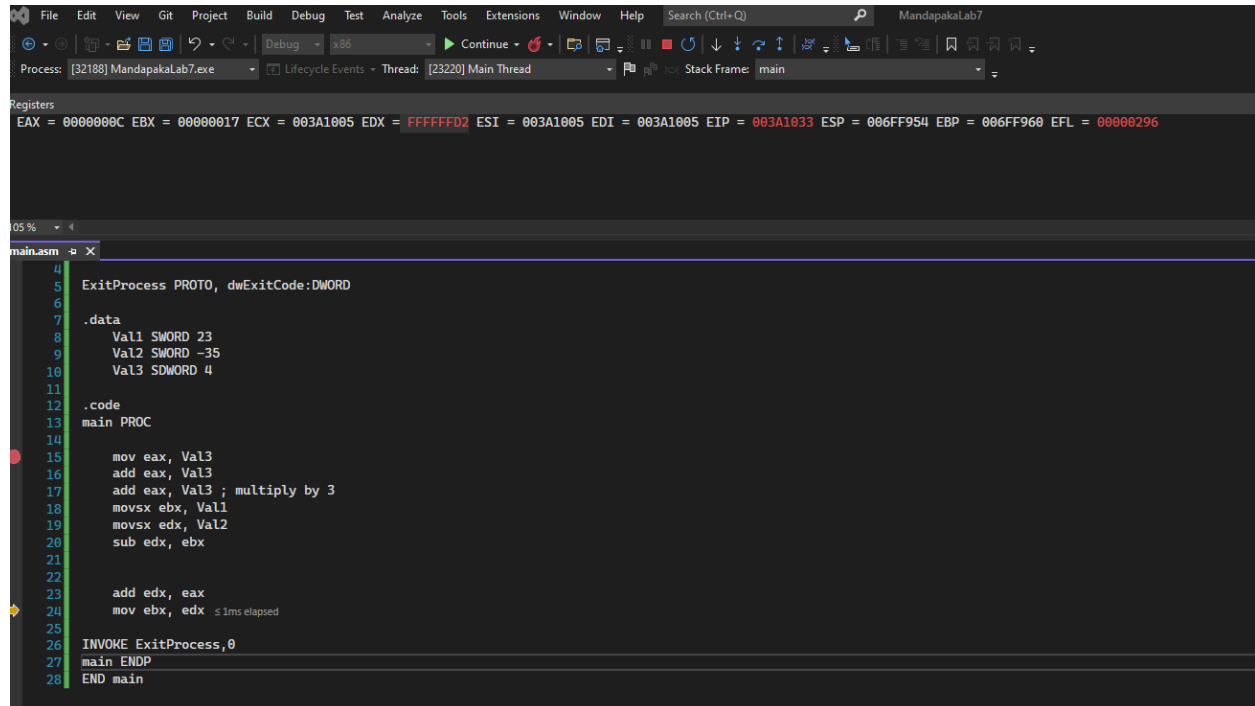
```
4 ExitProcess PROTO, dwExitCode:DWORD
5
6
7 .data
8     Val1 SWORD 23
9     Val2 SWORD -35
10    Val3 SDWORD 4
11
12 .code
13 main PROC
14
15     mov eax, Val3
16     add eax, Val3
17     add eax, Val3 ; multiply by 3
18     movsx ebx, Val1
19     movsx edx, Val2
20     sub edx, ebx
21
22
23     add edx, eax ; 1ms elapsed
24     mov ebx, edx
25
26 INVOKE ExitProcess,0
27 main ENDP
28 END main
```

Explanation: Here I am subtracting the value from the edx register with the ebx register which is subtracting val2 with val1 and the sum or the answer of that subtraction will be stored in the edx register

Line: 23

Instruction: add edx, eax

Register value: EDX = FFFFFFFD2



The screenshot shows a debugger window for a process named 'MandapakaLab7'. The 'Registers' pane at the top displays the following values: EAX = 0000000C, EBX = 00000017, ECX = 003A1005, EDX = FFFFFFFD2, ESI = 003A1005, EDI = 003A1005, EIP = 003A1033, ESP = 006FF954, EBP = 006FF960, and EFL = 00000296. The 'main.asm' file is open in the main pane, showing assembly code. The instruction at line 23 is 'add edx, eax'. The code includes a 'main' procedure that initializes variables Val1, Val2, and Val3, and then performs several arithmetic operations on registers. The instruction at line 23 is highlighted with a yellow cursor.

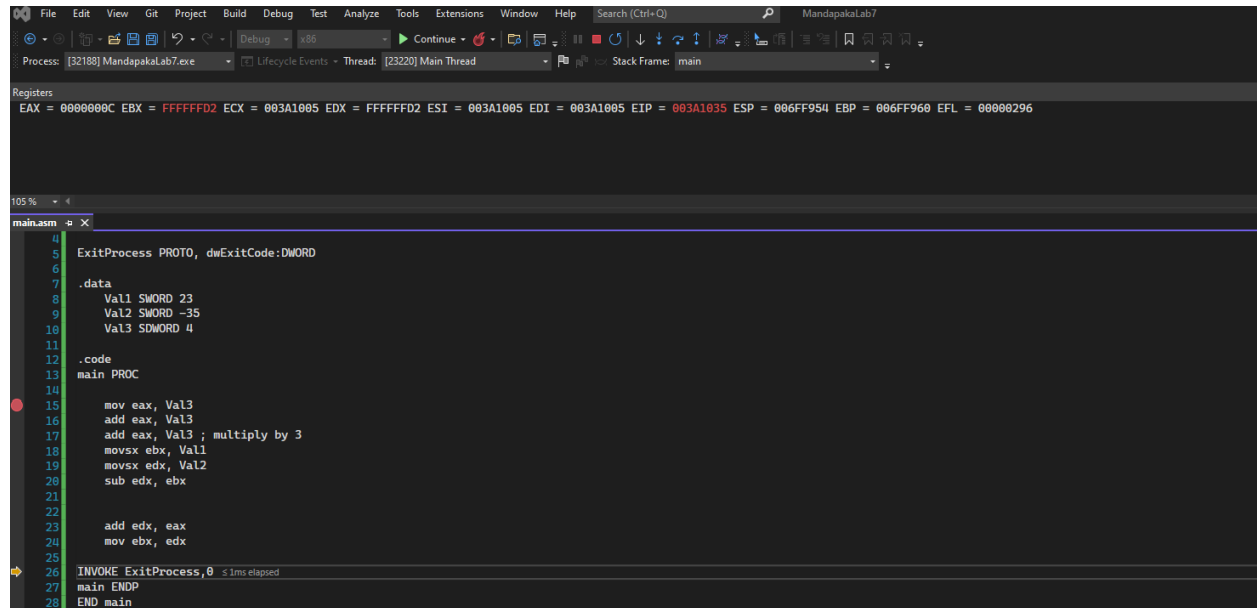
```
4 ExitProcess PROTO, dwExitCode:DWORD
5
6
7 .data
8     Val1 SWORD 23
9     Val2 SWORD -35
10    Val3 SDWORD 4
11
12 .code
13 main PROC
14
15     mov eax, Val3
16     add eax, Val3
17     add eax, Val3 ; multiply by 3
18     movsx ebx, Val1
19     movsx edx, Val2
20     sub edx, ebx
21
22
23     add edx, eax
24     mov ebx, edx ; 1ms elapsed
25
26 INVOKE ExitProcess,0
27 main ENDP
28 END main
```

Explanation: in this instruction we are adding the value of edx, which is -58 with the value in ebx which is 12 which gives us -48 and the hexadecimal value for -48 is D2.

Line: 24

Instruction: mov ebx, edx

Register value: EBX = FFFFFFFD2



The screenshot shows a debugger window with the following components:

- Registers:** EAX = 0000000C, EBX = FFFFFFFD2, ECX = 003A1005, EDX = FFFFFFFD2, ESI = 003A1005, EDI = 003A1005, EIP = 003A1035, ESP = 006FF954, EBP = 006FF960, EFL = 00000296.
- Assembly View:** The main.asm file is open, showing assembly code for the main function. The code includes data definitions (Val1, Val2, Val3) and a series of instructions: mov eax, Val3; add eax, Val3; add eax, Val3; movsx ebx, Val1; movsx edx, Val2; sub edx, ebx; add edx, eax; mov ebx, edx; invoke ExitProcess, 0.
- Disassembly View:** The bottom pane shows the disassembly of the main function, including the instruction: invoke ExitProcess, 0.

Explanation: Here I have performed the operation in the edx register and I did mov operation because I want to move the sum value of the evaluated expression into the ebx register.

Complete for the all the lines inside main

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) MandapakaLab7

Process: [32188] MandapakaLab7.exe Thread: [23220] Main Thread Stack Frame: main

Registers
EAX = 0000000C EBX = FFFFFFFD2 ECX = 003A1005 EDX = FFFFFFFD2 ESI = 003A1005 EDI = 003A1005 EIP = 003A1035 ESP = 006FF954 EBP = 006FF960 EFL = 00000296

105 %

main.asm

```
4 ExitProcess PROTO, dwExitCode:DWORD
5
6
7 .data
8     Val1 SWORD 23
9     Val2 SWORD -35
10    Val3 SDWORD 4
11
12 .code
13 main PROC
14
15     mov eax, Val3
16     add eax, Val3
17     add eax, Val3 ; multiply by 3
18     movsx ebx, Val1
19     movsx edx, Val2
20     sub edx, ebx
21
22
23     add edx, eax
24     mov ebx, edx
25
26 INVOKE ExitProcess,0 ; 1ms elapsed
27 main ENDP
28 END main
```