CSC 3210

Computer Organization and Programming

Lab 5

Answer Sheet

Student Name: **Asrar Syed**

Section: **022 / CRN: 17915**

**Lab 5(a): (3 points)**

**Create a project and generate listing file**

**Screenshot of the listing file content for the program (slide 38):**

* **mov 20 to AL register**
* **mov 100 to AX register**
* **mov 1000 to EAX register**
* **add 2 to the content of EAX register**
* **subtract 20 from al register**

**Done: Attached with File**

**Lab 5(b): (3 Points)**

**Write a program (slide 55) to evaluate an expression with variables**

**Debug the code until the ‘invoke ExitProcess, 0’.**

**Attach screenshot showing the content of AX register at the end along with the code**

**Start Debug**

**A screenshot of a computer

Description automatically generated**

**End Debug**

A screenshot of a computer

Description automatically generated

**Line number: 20**

**Instruction: mov ebx, Yval**

**Register value: EBX = 0000001E & EIP = 00F41016**

**Flags: Don’t need flags this time – according to TA**

**Explanation: The value at the memory address represented by Yval is copied to EBX.**

**Line number: 21**

**Instruction: add ebx, Zval**

**Register value: EBX = 00000046 & EIP = 00F4101C**

**Flags: Don’t need flags this time – according to TA**

**Explanation: The value at the memory address represented by Zval is added to the value in EBX.**

**Line number: 23**

**Instruction: sub Xval, ebx**

**Register value: EBX = 00000046 & EIP = 00F41022**

**& EFL = 00000287**

**Flags: Don’t need flags this time – according to TA**

**Explanation: The value at the memory address represented by Xval is subtracted from the value in EBX.**

**Line number: 24**

**Instruction: mov eax, Xval**

**Register value: EAX = FFFFFFD4 & EIP = 00F41027**

**Flags: Don’t need flags this time – according to TA**

**Explanation: The value at the memory address represented by Xval is copied to Eax.**

**Overall: We have DWORD variables used to do math problem.**

**Lab 5(c): (2 points)**

**Write a program (slide 59) to see the data items in an array**

**Debug the code until the ‘invoke ExitProcess, 0’.**

**Attach screenshot showing the content of AX register at the end along with the code.**

**Start Debug**

**A screenshot of a computer

Description automatically generated**

**End Debug**

A screenshot of a computer

Description automatically generated

**Line number: 17**

**Instruction: mov eax, 0**

**Register value: EAX = 00000000 & EIP = 00F01015**

**Flags: Don’t need flags this time – according to TA**

**Explanation: Sets the accumulator register (eax) to 0.**

**Line number: 18**

**Instruction: mov ax, myWord+0**

**Register value: EAX = 00000001 & EIP = 00F0101B**

**Flags: Don’t need flags this time – according to TA**

**Explanation: myWord is a word array of (2-byte elements) with 4 elements initialized to the values 1, 2, 3, 4, 5.** **Copies the first word from the myWord array (value 1) to the lower half of eax (register ax). This effectively loads the initial value 1 from the array for adding.**

**Line number: 19**

**Instruction: add ax, myWord+2**

**Register value: EAX = 00000003 & EIP = 00F01022**

**& EFL = 00000206**

**Flags: Don’t need flags this time – according to TA**

**Explanation: myWord is a word array of (2-byte elements) with 4 elements initialized to the values 1, 2, 3, 4, 5. Adds the third word from the array (value 3) to the existing value in ax.** **So, the current value in AX becomes 1 + 2 = 3.**

**Line number: 20**

**Instruction: add ax, myWord+4**

**Register value: EAX = 00000006 & EIP = 00F01029**

**Flags: Don’t need flags this time – according to TA**

**Explanation: myWord is a word array of (2-byte elements) with 4 elements initialized to the values 1, 2, 3, 4, 5. Adds the fifth word from the array (value 5) to the accumulated sum.**

**Line number: 21**

**Instruction: add ax, myWord+6**

**Register value: EAX = 0000000A & EIP = 00F01030**

**Flags: Don’t need flags this time – according to TA**

**Explanation: myWord is a word array of (2-byte elements) with 4 elements initialized to the values 1, 2, 3, 4, 5. Adds the value located at myWord+6 (which holds 4) to the AX register. The value in AX becomes 6 + 4 = 10. Hence the A in Hex.**

**Line number: 22**

**Instruction: add ax, myWord+8**

**Register value: EAX = 0000000F & EIP = 00F01037**

**Flags: Don’t need flags this time – according to TA**

**Explanation: myWord is a word array of (2-byte elements) with 4 elements initialized to the values 1, 2, 3, 4, 5. This adds the value located at myWord+8 (which holds 5) to the AX register. The final sum accumulated in AX becomes 10 + 5 = 15. Hence the F in Hex.**