CSC 3210

Computer Organization and Programming

Lab 5

Answer Sheet

Student Name: **Rafid Shaon**

Section: 06 / CRN: 18119 – Bikram

**Lab 5(a) – Problem 1**

1. **Write and run a program with following instructions:**
   * **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**
2. **Generate listing file for the program.**
   * **Take a screenshot of the listing file content**

**|**

**|**

**|**

**|**

**|**

**|**

**|**

**|**

**|**

**|**

**|**

**\/**

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

**Lab 5(b) – Problem 2**

Graphical user interface, application

Description automatically generated

1. **Write a program to implement the following expression in assembly:**
   * + - * **EAX = Xval - (Yval + Zval)**

**Xval is signed 32-bit integer variable**

**Yval and Zval are unsigned 32-bit integer variable**

**Assign Xval 26, Yval 30, and Zval 40.**

1. **Debug the code until the ‘invoke ExitProcess, 0’. Attach screenshot showing the content of AX register at the end along with the code.**

A screenshot of a computer

Description automatically generated

**Lab 5(c) – Problem 3**

Graphical user interface, application

Description automatically generated

Debug through each line of code, also take screenshots that include the code and register window.

Then fill out the following information.

1. **What is the total size of the myWord array? – 40 bytes**
2. **Debug the code until the ‘invoke ExitProcess, 0’. Attach screenshot showing the content of AX register at the end along with the code. – Down Below (Line Number 23 Screen Cap)**

**|**

**|**

**|**

**|**

**|**

**|**

**|**

**|**

**\/**

Graphical user interface

Description automatically generated

**Success Build – Screen Cap**

Graphical user interface

Description automatically generated

**Start Debug – Screen Cap**

1. *mov 0 to EAX register*

* Line Number: **18**
* Instruction: **mov eax, 0**
* Register Values: EAX = **00000000**
* Screenshot: A screenshot of a computer

  Description automatically generated with medium confidence

1. *mov myWord+0 to AX register*

* Line Number: **19**
* Instruction: **mov ax, myWord+0**
* Register Values: EAX = 0000**0001**
* Screenshot: A screenshot of a computer

  Description automatically generated with medium confidence

1. *add myWord+2 to AX register*

* Line Number: **20**
* Instruction: **add ax, myWord+2**
* Register Values: EAX = 0000**0003**
* Screenshot: A screenshot of a computer

  Description automatically generated

1. *add myWord+4 to AX register*

* Line Number: **21**
* Instruction: **add ax, myWord+4**
* Register Values: EAX = 0000**0006**
* Screenshot: Graphical user interface

  Description automatically generated

1. *add myWord+6 to AX register*

* Line Number: **22**
* Instruction: **add ax, myWord+6**
* Register Values: EAX = 0000**000A**
* Screenshot: Graphical user interface

  Description automatically generated

1. *add myWord+8 to AX register*

* Line Number: **23**
* Instruction: **add ax, myWord+8**
* Register Values: EAX = 0000**000F**
* Screenshot: Graphical user interface

  Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**End Debug – Screen Cap**

**Lab 5(d) – Problem 4**

Graphical user interface, text, application, email

Description automatically generated

Debug through each line of code, also take screenshots that include the code and register window.

Then fill out the following information.

1. **What is the difference between symbolic constant and variables? – Down Below (On the last 2 Pages)**
2. **Debug the code until ‘invoke ExitProcess, 0’. Attach the screenshot showing the content of al register at the end along with the code. – Down Below (Line Number 20 Screen Cap)**

**|**

**|**

**|**

**|**

**|**

**|**

**\/**

Graphical user interface

Description automatically generated

**Success Build – Screen Cap**

Graphical user interface

Description automatically generated

**Start Debug – Screen Cap**

1. *mov 0 to EAX register*

* Line Number: **19**
* Instruction: **mov eax, 0**
* Register Values: EAX = **00000000**
* Screenshot: Graphical user interface

  Description automatically generated

1. *mov myString\_length to AL register*

* Line Number: **20**
* Instruction: **mov al, myString\_length**
* Register Values: EAX = 000000**68**
* Screenshot: Graphical user interface

  Description automatically generated

Graphical user interface

Description automatically generated

**End Debug – Screen Cap**

A symbol is something that the compiler deals with. The compiler treats a const pretty much the way it treats a variable. On the other hand, a #define is something the compiler is not even aware of, because the precompiler transforms it into its value. It's like search-and-replace. If you do

#define A 5

and then

b += A;

The precompiler translates it into

b += 5;

and all the compiler sees is the number 5.

(Borrowing from earlier posts) A literal constant is a value typed directly into your program wherever it is needed. For example

int breakpoint = 10;

The variable breakpoint is an integer (int); 10 is a literal constant. You can't assign a value to 10, and its value can't be changed. Unlike a variable, a constant can't be changed after it is assigned a value (initialized).

A symbol is something that the compiler deals with. In this example, TEN is a symbolic constant created using the #define function. A #define is something the compiler is not even aware of, because the precompiler transforms it into its assigned (defined) value. The precompiler searches out and replaces every symbol constant inside your program with a value.

#define TEN 10 /\* These two lines of code become one... \*/

breakpoint += TEN; /\* after running through the precompiler \*/

The precompiler translates it into

Breakpoint += 10;

The compiler never sees TEN but only its assigned value, 10. Rather than looking through the entire program and changing every variable definition to the new value that was set using a literal constant, 10, change the definition of a single symbol constant... TEN to 11 and let the precompiler do the changes for you.

**Variable**:

The **variable** is a type of container that is used to hold a particular type of value and this type is defined by its data type. A variable value can be modified. The compiler allocates the memory to the variable according to the data type.

**Symbolic Constant:**

The **symbolic constant** is a constant value, and this value can't be changed during the program.

The differences between variable and symbolic constant are given below:

|  |  |
| --- | --- |
| **Variable** | **Symbolic Constant** |
| The variable is a type of container that is used to hold a particular type of value. | The symbolic constant is a constant value that is known by a name. |
| The value can be changed. | The value can't be changed. |
| It can be initialized and modified at any time. | It must be initialized at declaration time. |