EL 2003 COMP ORG & ASSEMBLY LANGUAGE LAB

LAB 02

ASSEMBLY LANGUAGE FUNDAMENTALS

_____ _____ ___

STUDENT NAME ROLL NO SEC

_____

SIGNATURE & DATE

MARKS AWARDED: _____

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI

Prepared by: Muhammad Nadeem

Version: 1.0
Date:

Lab Session 02 ASSEMBLY LANGUAGE FUNDAMENTAL

Objectives:

- Debugging of programs
- Basic elements of Assembly language

Steps Involved in Creating and Running a Program:

ASSEMBLER:

It converts the assembly language to machine language (Object Code) May contain unresolved references (i.e. file contains some or all of complete program)

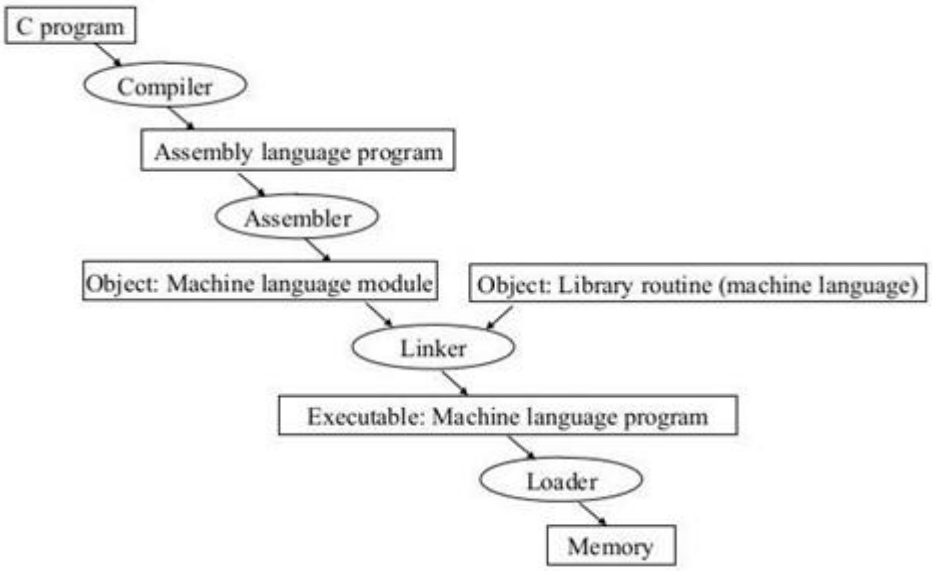
LINKER:

A program that combines object files to create a single "executable" file. Major functional difference is that all references are resolved. (i.e. Program contains all parts needed to run) A program that loads executable files into memory, and may initialize some registers (e.g. IP) and starts it going.

DEBUGGER:

A program that loads but controls the execution of the program. To start/stop execution, to view and modify state variables.

STEPS IN CREATING & RUNNING CODE:



SECTION 1: DEBUGGING OUR PROGRAM

We have seen how to configure Visual Studio 2019 for Assembly Language and tested it with a sample program. The output of our sample program was displayed using a console window but it is usually more desirable to watch the step by step execution of our program with each line of code using breakpoints.

Let us briefly define the keywords relevant to debugging in Visual Studio and then we will cover an example for understanding.

DEBUGGER

The (Visual Studio) debugger helps us observe the run-time behavior of our program and find problems. With the debugger, we can break execution of our program to examine our code, examine and edit variables, view registers, see the instructions created from our source code, and view the memory space used by our application.

BREAKPOINT

A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point. When execution is suspended at a breakpoint, your program is said to be in break mode.

CODE STEPPING

One of the most common debugging procedures is stepping: executing code one line at a time. The Debug menu provides three commands for stepping through code:

- Step Into (By pressing F11)
- Step Over (By pressing F10)
- Step Out (Shift+F11)

SINGLE STEPPING

To see the values of internal registers and memory variables during execution, let us use an example. Copy the following code onto your Test.asm file.k

1. Right-click on line 6 to insert a breakpoint.
2. Click on Debug tab from the toolbar, select Start Debugging OR press F10 to start stepping over the code.

BASIC ELEMENT OF ASSEMBLY LANGUAGE

INTEGER CONSTANTS

Integer constants are made up of an optional leading sign, one or more digits and an optional suffix character.

Format:

[{+ | -}] digits radix

Examples:

26	for decimal
26d	for decimal
10111110b	for binary
42o	for octal
1Ah	for Hexadecimal
0A3h	for Hexadecimal

CHARACTER CONSTANTS

Character constants are made up of a single character enclosed in either single or double quotes.

Example:

'A' "d"

STRING CONSTANTS

A string of characters enclosed in either single or double quotes.

Example:

"Hello World"

IDENTIFIERS

An identifier is a programmer-defined name of a variable, procedure or code label.

Format:

They may contain between 1 and 247 characters. They are not case sensitive. The first character must be a letter (A..Z, a..z), underscore (_), @ , ?, or $. Subsequent characters may also be digits.

An identifier cannot be the same as an assembler reserved word. For example: reserved words are instruction mnemonics, directives, attributes, operators, predefined symbols.

Examples:

myVar
_abc
hello2

DIRECTIVES

A directive is a command embedded in the source code that is recognized and acted upon by the assembler. Directives do not execute at runtime. They can assign names to memory segments. In MASM, directives are case insensitive. For example, it recognizes .data, .DATA and .Data as equivalent.

Let us see what different directives we can use to define segments of our program:

The **.DATA** directive identifies the area of a program containing variables:

Syntax:

.data

The **.CODE** directive identifies the area of a program containing executable instructions:

Syntax:

.code

PROGRAM TEMPLATE:

```
TITLE Program Template (Template.asm)
; Program Description:
; Author:
; Creation Date:
; Revisions:
; Date:
INCLUDE Irvine32.inc
.data
        ; (insert variables here)
.code
        main PROC
                ; (insert executable instructions here)
        exit
        main ENDP
        ; (insert additional procedures here)
END main
```

Comments: Comments are an important way for the writer of a program to communicate information about the program's design to a person reading the source code.

Comments can be specified in two ways:

• **Single-line comments**: beginning with a semicolon character (;). All characters following the semicolon on the same line are ignored by the assembler.

• **Block comments**: beginning with the COMMENT directive and a user-specified symbol. For example,

COMMENT !

This line is a comment.

This line is also a comment.

!

SECTION 3: EXERCISE

1. **Implement all of these equations in assembly language.**

- 47 + 39 + 60 + 85 + 64+54o-0Ah
- 30- 9 + 186 – 150
- 101110 + 50Ah + 6710d + 1010001 + F
- 10001101 – D83h + 385+10 + 1111101 – E+F
- 101b- 9 + 1A4h – 569o

2. **Write a program in assembly language that implements following expression:**

- edx = eax + 1 + ebx– ecx + 0Ah-65o+73d
- eax = 5ADh – ebx + 65o + 65d – 11110111 + 150
- ebx = 5ADh – eax + 65d + 73o – 11100101 + 7Bh
- ecx = 110010101101b + 45h-73o + ebx -ecx + 1

1. **Submit screenshots of each task containing register values i.e. Debugging window. (in single word file).**

2. ***The codes of all task in a text file. Use Notepad.***
3. ***Submissions should be made on Google Classroom***.