Lab Manual for Computer Organization and Assembly Language

Lab-1

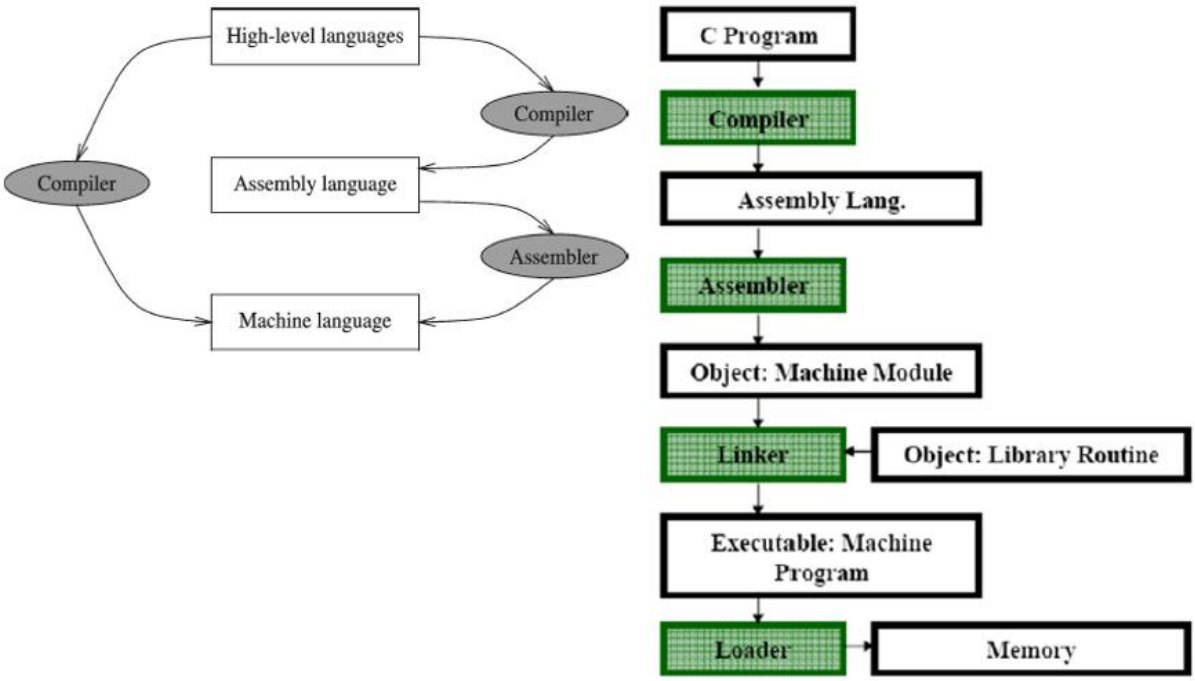
Introduction to Assembly Language

Table of Contents

1. Introduction

Machine language (computer's native language) is a system of impartible instructions executed directly by a computer's central processing unit (CPU).

- Instructions consist of binary code: 1s and 0s



The difference between compiling and interpreting is as follows. Compiling translates the high-level code into a target language code as a single unit. Interpreting translates the individual steps in a high-level program one at a time rather than the whole program as a single unit. Each step is executed immediately after it is translated.

C, C++ code is executed faster than Java code, because they transferred to assembly language before machine language.

2. Objective

- To get basic understanding of Assembly Language.
- To write simple assembly program and learn how to debug and run it
- To get an understanding of identifying basic errors.
- To understand the emu8086 emulator and their purpose.
- To be able to write basic assembly codes.

3. Concept Map

This section provides you the overview of the concepts that will be discussed and implemented in this lab.

3.1 Assembly Language

Assembly Language is a programming language that is very similar to machine language, but uses symbols instead of binary numbers. It is converted by the assembler (e.g. Tasm and Masm) into executable machine-language programs.

Assembly language is machine-dependent; an assembly program can only be executed on a particular machine.

3.1.1 Introduction to Assembly Language Tools

Software tools are used for editing, assembling, linking, and debugging assembly language programming. You will need an assembler, a linker, a debugger, and an editor.

3.1.1.1 Assembler

An assembler is a program that converts source-code programs written in assembly language into object files in machine language. Popular assemblers have emerged over the years for the Intel family of processors. These include MASM (Macro Assembler from Microsoft), TASM (Turbo Assembler from Borland), NASM (Netwide Assembler for both Windows and Linux), and GNU assembler distributed by the free software foundation. We will use MASM 6.15 and TASM. Masm.exe creates an .obj file from an .asm file.

3.1.1.2 Linker

A linker is a program that combines your program's object file created by the assembler with other object files and link libraries, and produces a single executable program. You need a linker utility to produce executable files.

• Link.exe creates an .exe file from an .obj file.
• Use make16.bat to assemble and link a 16-bit format assembly program.
• Use make32.bat to assemble and link a 32-bit format assembly program.

3.1.1.3 Debugger

A debugger is a program that allows you to trace the execution of a program and examine the content of registers and memory.

- For 16-bit programs, MASM supplies a 16-bit debugger named CodeView. CodeView can be used to debug only 16-bit programs and is already provided with the MASM 6.15 distribution.

3.1.1.4 Editor

You need a text editor to create assembly language source files. MASM6.15 has its own editor or you can use for example Notepad++. In this course, we use microprocessor 8086 emulator for assembly language programs. The downloading link is available below.
http://www.emu8086.com/files/emu8086v408r11.zip

4. Sample Program

```
.model small
.data
msg db "Welcome to 1st Assembly Language Lab","$"
```

```
.code
main:
mov ax,@data
mov ds,ax

mov ah,9
mov dx, offset msg
int 21h

mov ah,4ch
int 21h
end main
```

4.1 Analysis of code:

Instruction	Description
. MODEL SMALL	is a directive to tell the assembler to use one data segment and one code segment. All data fits in one 64K segment, all code fits in one 64K segment. Maximum program size is 128K.
.DATA	is a directive to put in data segment.
.CODE	is a directive to put in code segment.
@Data	is a default address of data segment to put it in ax register.
mov ax, @data **mov ds, ax**	As note we can't put data in ds register directly. So we use intermediate register (ax) as in the **mov ds, ax**
mov ah, 9	Put the service 9 in ah.
int 21h	(Interrupt 21 hexa), it has many services like 9,8,2, and each one has special work.
mov ah,4ch **int 21h**	The two statements to terminate the execution of the program.
end main	is a directive to indicate the end of the file

5. Procedure& Tools

In this section, you will study how to setup and MASM Assembler.

5.1 Tools

- Download emu 8086 from (http://www.emu8086.com/files/emu8086v408r11.zip)
- Just extract the emu8086.15.zip on C
- Install emu8086

6. Practice Tasks

This section will provide more practice exercises which you need to finish during the lab. You need to finish the tasks in the required time. When you finish them, put these tasks in the following folder:
\\fs\assignments$\Coal\

6.1 Practice Task 1 [Expected time = 15mins]

Print your university card's detail on console in the proper format.

6.2 Practice Task 2 [Expected time = 15mins]

Print the Triangle on the screen with " * " character.

6.3 Out comes

After completing this lab, student will be able to setup emu8086. He/ She will also be able to compile and run basic Assembly programs.

7. Evaluation Task (Unseen) [Expected time = 30mins for tasks]

The lab instructor will give you unseen task depending upon the progress of the class.

8. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 3: Evaluation of the Lab

Sr. No.	Task No	Description	Marks
1	4	Problem Modeling	20
2	6	Procedures and Tools	10
3	7	Practice tasks and Testing	35
4	8	Evaluation Tasks (Unseen)	20
5		Comments	5
6		Good Programming Practices	10

9. Further Reading

This section provides the references to further polish your skills.

9.1 Slides

The slides and reading material can be accessed from the folder of the class instructor available at \\fs\lectures$\