Computer Organization and Assembly Language

Week 1 to 3

Dr. Muhammad Nouman Durrani

Introduction

Definition 1:

- Computer organization is concerned with the way hardware components operate and the way they are connected together to form a computer system.

Definition 2:

- Computer organization is concerned with the structure and behavior of a computer system.

Definition 3:

- *Assembly language* consists of statements written with short mnemonics such as ADD, MOV, SUB, and CALL.

Introduction

- The main objective of this course is to:
  - Understand the basic structure of a computer system
  - How to write programs by understanding the hardware structure

Assembly Language Programming

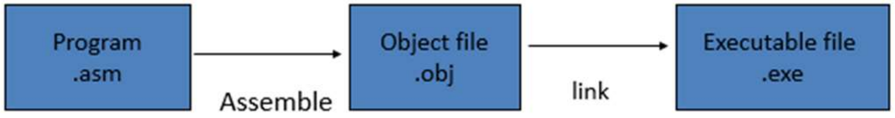
- An assembly language is a low-level programming language designed for a specific type of processor
- Assembly language is used to write the program using alphanumeric symbols (or mnemonic), e.g. ADD, MOV, PUSH etc.
- Assembly Language depends on the machine code instructions
  - Every assembler has its own assembly language
- Assembly language has a one-to-one relationship with machine language:
  - Each assembly language instruction corresponds to a single machine-language instruction.
  - A single machine-language instruction can take up one or more bytes of code

Assembly language programming

- The instruction must specify which operation (opcode) is to be performed and the operands
- E.g. ADD AX, BX
  - ADD is the operation
  - AX is called the destination operand
  - BX is called the source operand
  - The result is AX = AX + BX
- When writing assembly language program, you need to think in the instruction level

Assembly language programming: Example

- MOV AL, 00H
  - The native language of a computer is machine language (using 0,1 to represent the operation)
  - The machine language code for the above instruction is B4 00 (2 bytes)
  - After assembled, and linked into an executable program, the value B400 will be stored in the memory
  - When the program is executed, then the value B400 is read from memory, decoded and carry out the task
    - The executable program could be .com, .exe, or .bin files



Assembly language programming

- Learning assembly language programming will help understanding the operations of a microprocessor
- To learn:
  - Need to know the functions of various registers
  - Need to know how external memory is organized and how it is addressed to obtain instructions and data (different addressing modes)
  - Need to know what operations (or the ***instruction set***) are supported by the CPU

High Level Languages

- A high-level language (HLL) is a programming language that enables a programmer to write programs that are more or less independent of a particular type of computer

- Such languages are considered high-level because they are closer to human languages and further from machine languages.

- Examples are C, C++, Java, FORTRAN, or Pascal

High Level Languages

- High-level languages such as C++ and Java have a *one-to-many* relationship with assembly language and machine language.

- A single statement in C++ expands into multiple assembly language or machine instructions.

- The following C++ code carries out two arithmetic operations and assigns the result to a variable. Assume X and Y are integers:

```
int Y;
int X = (Y + 4) * 3;
```

Example:
int Y;
int X = (Y + 4) * 3;

- Following is the equivalent translation to assembly language.

- The translation requires multiple statements because assembly language works at a detailed level:

```
mov eax,Y ; move Y to the EAX register
add eax,4 ; add 4 to the EAX register
mov ebx,3 ; move 3 to the EBX register
imul ebx ; multiply EAX by EBX
mov X,eax ; move EAX to X
```

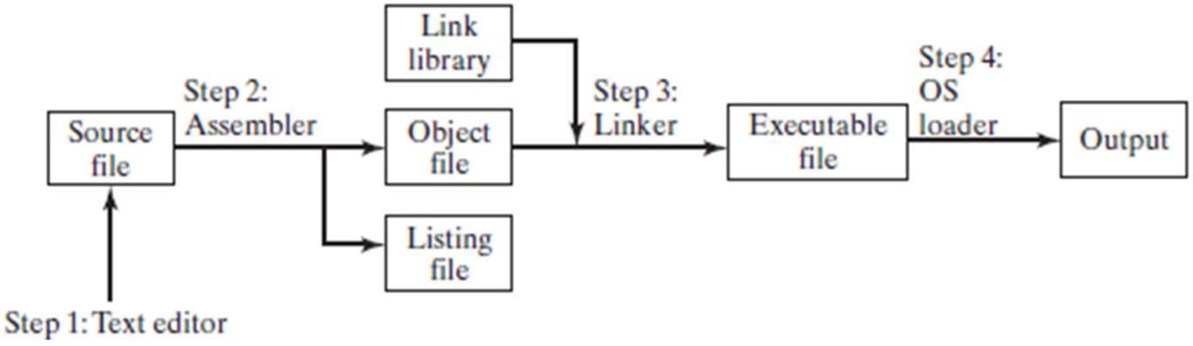
Registers are named storage locations in the CPU that hold intermediate results of operations

What Are Assemblers and Linkers?

- **Assembler** is a utility program that converts source code programs from assembly language into an object file, a machine language translation of the program.
  - Optionally a Listing file is also produced.
  - We'll use MASM as our assembler.
- The **linker** reads the object file and checks to see if the program contains any calls to procedures in a link library.
  - The **linker** copies any required procedures from the link library, combines them with the object file, and produces the *executable file*.
  - Microsoft 16-bit linker is LINK.EXE and 32-bit is Linker LINK32.EXE.
- **OS Loader:** A program that loads executable files into memory, and branches the CPU to the program's starting address, (may initialize some registers (e.g. IP)) and the program begins to execute.
- **Debugger** is a utility program, that lets you step through a program while it's running and examine registers and memory

MASM provides CodeView, TASM provides Turbo Debugger and msdev.exe for 32-bit Window console programs.

FIGURE 3–7 Assemble-Link-Execute cycle.



Listing File

- A *listing file* contains:
  - a copy of the program’s source code,
  - with line numbers,
  - the numeric address of each instruction,
  - the machine code bytes of each instruction (in hexadecimal), and
  - a symbol table.

The symbol table contains the names of all program identifiers, segments, and related information.

Figure 3–8 Excerpt from the AddTwo source listing file.

```
; AddTwo.asm - adds two 32-bit integers.
; Chapter 3 example

.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO,dwExitCode:DWORD

 00000000                      .code
 00000000                      main PROC
 00000000  B8 00000005            mov  eax,5
 00000005  83 C0 06               add  eax,6

                                  invoke ExitProcess,0
 00000008  6A 00                  push   +000000000h
 0000000A  E8 00000000 E          call   ExitProcess
 0000000F                      main ENDP
                               END main
```

Assembly Language for x86 Processors

- *Assembly Language for x86 Processors* focuses on programming microprocessors compatible with the Intel IA-32 and AMD x86 processors running under Microsoft Windows.

- Assembly language bears the closest resemblance to native machine language.

Is Assembly Language Portable?

- A language whose source programs can be compiled and run on a wide variety of computer systems is said to be *portable.*

- A C++ program, for example, should compile and run on just about any computer, unless it makes specific references to library functions that exist under a single operating system.

- A major feature of the Java language is that compiled programs run on nearly any computer system.

Is Assembly Language Portable?

- Assembly language is not portable because it is designed for a specific processor family.

- There are a number of different assembly languages widely used today, each based on a processor family.

  - Some well-known processor families are Motorola 68x00, x86, SUN Sparc, Vax, and IBM-370.

- The instructions in assembly language may directly match the computer's architecture or they may be translated during execution by a program inside the processor known as a *microcode interpreter*.

What you'll learn from Assembly Language Programming

You will learn:

- Some basic principles of computer architecture, as applied to the Intel IA-32 processor family.
- How IA-32 processors manage memory, using real mode, protected mode, and virtual mode.
- How high-level language compilers (such as C++) translate statements from their language into assembly language and native machine code.
- How high-level languages implement arithmetic expressions, loops, and logical structures at the machine level.

What you'll learn from Assembly Language Programming

- You will improve your machine-level debugging skills.
  - Even in C++, when your programs have errors due to pointers or memory allocation, you can dive to the machine level and find out what really went wrong.
- High-level languages purposely hide machine-specific details, but sometimes these details are important when tracking down errors.
- Assembly language will help you understanding the interaction between the computer hardware, operating system, and application programs.

Applications of Assembly Language

- Embedded systems programs are written in C, Java, or assembly language, and downloaded into computer chips and installed in dedicated devices.
  - Some examples are automobile fuel and ignition systems, air-conditioning control systems, security systems, flight control systems, hand-held computers, modems, printers, and other intelligent computer peripherals.

- Many dedicated computer game machines have stringent memory restrictions, requiring programs to be highly optimized for both space and runtime speed.
  - Game programmers use assembly language to take full advantage of specific hardware features in a target system.