COA

| Machine code – Assembly Language

Prof. Dr. Mostafa Elhosseini

https://youtube.com/drmelhosseini

AGENDA

- Machine language
- Assembly language
- Compiler & Interpreter
- Program Execution Process
- Programmer’s View of a Computer System

Machine language

- Native to a processor; executed directly by hardware.
- Instructions consist of binary code: 1s and 0s
- Difficult to read and understand for humans.
- Provides a one-to-one correspondence with hardware instructions.
- No need for a translator or compiler
- Directly executable by the processor

Assembly language

- A programming language that uses symbolic names to represent operations, registers, and memory locations.
- Slightly higher-level language than machine language
- Readability of instructions is better than machine language.
- Provides a one-to-one correspondence with machine language instructions.
- Translated to machine code by an assembler.

Assembly language

- Can be used to write programs that are more readable and easier to modify than machine language.
- Typically requires less effort to write and debug than machine language.
- Cannot be executed directly by the processor and must be translated to machine code before execution.

Assembly language - Applications

- Assembly language provides a greater level of control over the hardware than higher-level programming languages.
- Operating system development: Assembly language is used to write low-level components of operating systems, such as device drivers, interrupt handlers, and system calls.
- Embedded systems: Assembly language is used to write firmware for embedded systems, such as microcontrollers and sensors.

Assembly language - Applications

- Real-time systems: Assembly language is used to write software for real-time systems, such as industrial control systems and robotics.
- Reverse Engineering: Assembly language is used to analyze and understand the behavior of software, such as malware and viruses.

Assembly language - Applications

- Video game programming: Assembly language is used to write high-performance video game engines and real-time graphics effects.
- Cryptography: Assembly language is used to write efficient cryptographic algorithms and protocols.
- System-level programming: Assembly language is used to write low-level system software, such as bootloaders, BIOS, and device drivers

Compiler

- A compiler is a software tool that translates an entire program written in a high-level programming language into machine code in one go.
- The output of a compiler is typically an executable file that can be run directly by the operating system

Compiler

- Performance: Compiled programs are generally faster and more efficient than interpreted programs, as they are already translated into machine code and do not need to be translated again during execution.
- Debugging: Interpreted programs are generally easier to debug than compiled programs, as the interpreter can provide real-time feedback on the execution of each line of code.

Interpreter

- An interpreter, on the other hand, is a software tool that executes a program written in a high-level programming language line-by-line or statement-by-statement.
- The interpreter does not produce an executable file, but rather reads and executes each line of code on the fly.

Machine Language and Instruction

- A program consists of a set of commands that are executed by a computer.
- Each command is called an instruction and it tells the computer what to do.
- Since computers only deal with binary data, instructions must be expressed in binary format using only 0s and 1s.
- The complete set of instructions in binary form is referred to as the machine language or instruction set of the computer.
- The instruction set specifies all of the operations that the computer can perform and how they should be executed.
- The instruction set is unique to each computer architecture and processor.

Machine Language and Instruction

- Programs written in higher-level programming languages are translated into the corresponding machine language instructions by a compiler or interpreter before being executed by the computer.

```
Address     Machine Code      Assembly Code
-----------------------------------------
0000        B8 00 10          MOV AX, 1000H
0003        B9 05 00          MOV CX, 5
0006        2B C8             SUB CX, AX
0008        8B C1             MOV AX, CX
000A        C3                RET
```

Executable (.exe) ↓ | ↓ | Assembler code ↓

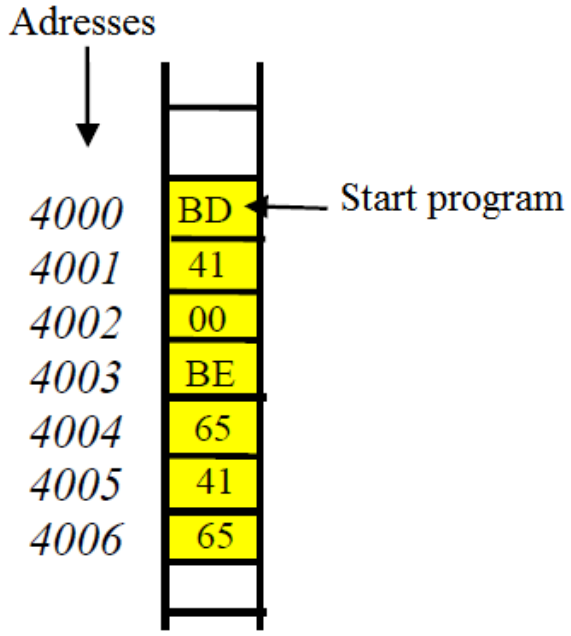
Machine codes as they are submitted to the processor	Hexadecimal Machine codes as one took the habit to represent them on paper	Writing more readable or called Mnemonic Assembler
10111101 01000001 00000000	BD 41 00	MOV BP,41h
10111110 01100101 01000001	BE 65 41	MOV SI,4165h
00000001 11011000	01 D8	ADD AX,BX
00111000 01000111 00000011	38 47 03	CMP [BX+3],AL

Program Execution Process

- When a program is executed, it is loaded into RAM by the operating system.
- The operating system allocates memory to the program and copies the program's instructions and data into that memory space.
- The start address of the program is communicated to the processor so that it knows where to begin executing the program.
- The processor fetches the first instruction of the program from the memory location specified by the start address and begins executing it.

Program Execution Process

- As each instruction is executed, the processor fetches the next instruction from memory and continues the process until the program is complete.
- After the program has finished executing, the operating system may release the memory space that was allocated to the program so that it can be used for other programs.



Instruction Fields

- The two main fields in machine language instructions are the opcode and the operands.
- The opcode specifies the unique operation that the instruction is to perform.
- The operands field specifies the source and destination of the data used in the operation.

Opcode	Operands
1001 0011	0011 1110

- The source/destination of operands can be a constant, the memory, or one of the general-purpose registers.

Programmer's View of a Computer System