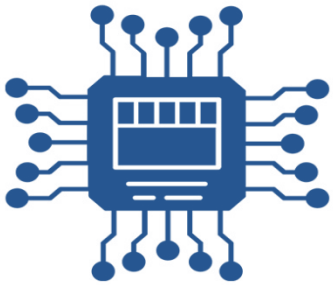

قـســـــــــم علوم الذكاء الاصطناعي
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

SUBJECT:

Microprocessor

CLASS:

SECOND

LECTURER:

Dr. Abdulkadhem A. Abdulkadhem

LECTURE: (3)

Microprocessor Programming

1. Introduction

Microprocessor programming is the process of writing instructions that control the operation of the microprocessor to perform specific tasks. In the case of the Intel 8086 microprocessor, programming is carried out using **Assembly Language**, which provides a low-level interface between software and hardware. Understanding assembly language programming is essential because it allows the programmer to directly manipulate registers, memory, and I/O devices, thereby achieving efficient and precise control over the system.

2. Machine Language vs Assembly Language

2.1 Machine Language

Machine language consists of binary-coded instructions (0s and 1s) that are directly understood and executed by the microprocessor. Each instruction corresponds to a specific opcode.

Characteristics:

- Executed directly by the CPU
- Very fast execution
- Difficult for humans to read, write, and debug
- Error-prone

Example (Machine Code):

```
10110000 00000101
```

2.2 Assembly Language

Assembly language is a symbolic representation of machine language. It uses **mnemonics** (رموز محتصرة) to represent machine instructions, making programs easier to write, read, and maintain.

Example:

```
MOV AL, 05H
```

Advantages:

- Easier to understand than machine language
- Efficient use of hardware resources
- Direct access to registers and memory

Feature	Machine Language	Assembly Language
Readability	Completely unreadable (Binary code)	Uses mnemonics, easier to read
Translation Required	No (Direct execution)	Yes (Requires an assembler)
Error Debugging	Extremely difficult	Easier due to symbolic representation
Performance	Fastest execution	Slightly slower due to the translation step
Portability	Not portable (hardware-specific)	Architecture-dependent
Ease of Use	Very hard for humans to program	Easier but still technical
Common Usage	Directly embedded in firmware, microcontrollers	Used for performance optimization, drivers, and embedded systems

 Exam Note: Differences between machine language and assembly language are frequently asked as short-answer or MCQ questions.

3. Translator Programs: Assembler

An **assembler** is a system program that translates assembly language instructions into equivalent machine language instructions.

Types of Assemblers:

1. Single-pass assembler
2. Two-pass assembler (commonly used)

The assembler also checks syntax errors and generates object code.

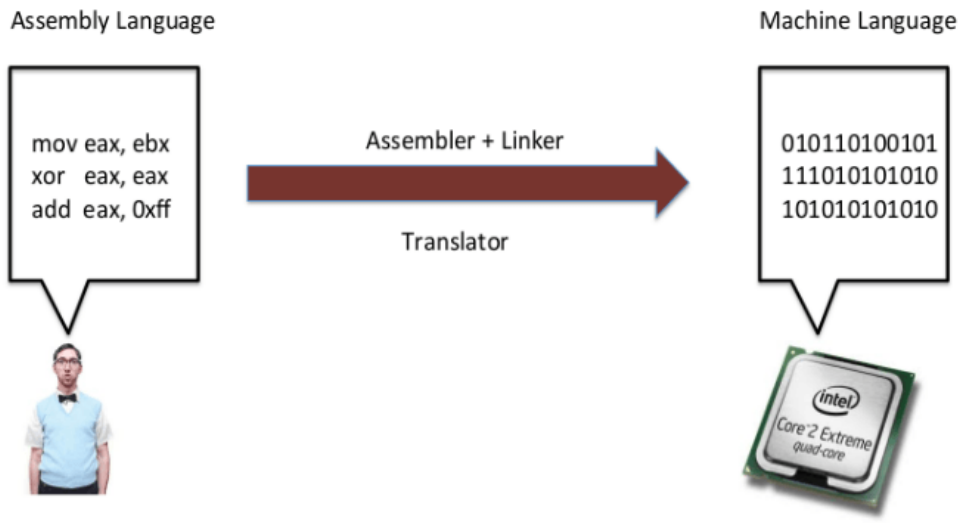

Figure 3.1: Role of Assembler in Program Translation

4. Structure of an 8086 Assembly Language Program

An 8086 assembly program follows a **logical and organized structure** that ensures correct assembly, loading, and execution. This structure reflects the internal organization of the processor and enforces a clear separation between instructions, data, and program flow.

A general program structure is shown below:

```
ORG 100H

START:
    ; Program instructions

    INT 20H
```

Each element of this structure plays a specific role in program execution, as explained below.

4.1 ORG Directive

The ORG (Origin) directive informs the assembler of the **starting address offset** for the program. It ensures that instructions and data are assembled at the correct memory locations.

This directive is essential for correct address calculation and program execution.

4.2 Labels

Labels are symbolic names assigned to memory locations or instruction addresses. They improve program readability and allow for logical branching and looping.

Example:

```
START:
```

Labels do not generate machine code; they serve as reference points during assembly.

4.3 Instructions

Instructions are executable statements that perform operations such as data transfer, arithmetic, logic, and control flow.

Example:

```
MOV AX, BX
```

Each instruction typically consists of:

- An **opcode** (operation)
- One or more operands (data or addresses)

4.4 Comments

Comments are explanatory notes added to the program to clarify functionality. They are ignored by the assembler and have no effect on execution.

Example:

```
; This instruction copies BX into AX
```

 Exam Tip: Comments improve code clarity but are not translated into machine code.

4.5. Program Termination

Programs must include a proper termination instruction to return control to the operating system.

Common termination methods include:

```
INT 20H

or

MOV AH, 4CH
```

```
INT 21H
```

Failure to terminate correctly may result in ***unpredictable behavior***.

5. Keywords and Mnemonics in Assembly Language

Assembly language uses predefined **keywords and mnemonics** that are recognized by the assembler.

Examples include:

- **Data transfer**: `MOV`, `XCHG`
- **Arithmetic**: `ADD`, `SUB`, `INC`, `DEC`
- **Control**: `JMP`, `CALL`, `RET`

Understanding these keywords is essential for writing correct assembly programs.

6. Simple Example Program

```
ORG 100H

START:
    MOV AX, 0005H
    MOV BX, AX
    INT 20H
```

Explanation:

- The first instruction loads the value 0005H into register AX
- The second instruction copies AX into BX
- The final instruction terminates the program

7. Execution Flow of an Assembly Program

1. Program is loaded into memory
2. Instruction pointer points to the first instruction
3. Instructions are **fetched**, **decoded**, and **executed** sequentially
4. Control flow instructions modify execution order if needed
5. Program terminates and control returns to the operating system

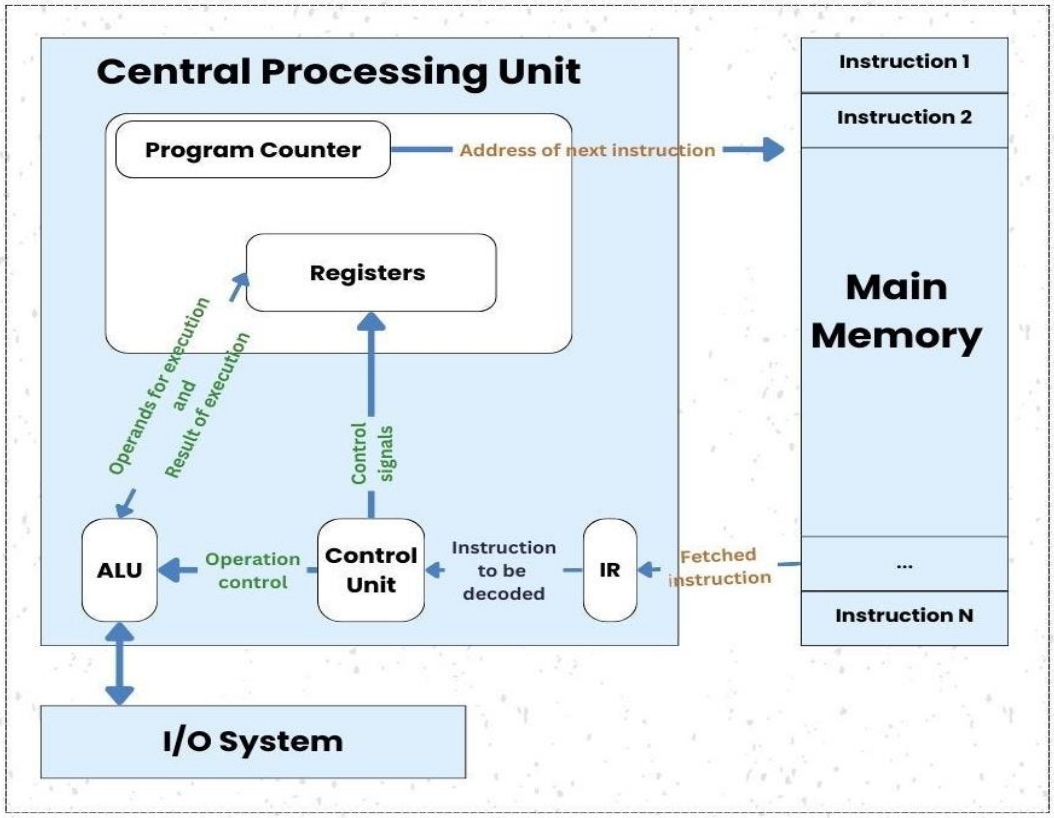

Figure 3.2: Instruction execution cycle in an 8086 assembly language program.

8. Program Segments in 8086

The 8086 uses a **segmented memory architecture**. Therefore, assembly programs are divided into segments:

1. **Code Segment (CS)** – stores executable instructions
2. **Data Segment (DS)** – stores variables and constants
3. **Stack Segment (SS)** – stores return addresses and temporary data
4. **Extra Segment (ES)** – used for additional data storage

 Exam Note: Segment types and their functions are essential theoretical questions.

9. Instruction Format of 8086

An 8086 instruction generally consists of:

- Opcode
- Operand(s)

Example:

```
MOV AX, BX
```

Here:

- `MOV` → Opcode
- `AX, BX` → Operands

10. Data Types in Assembly Language

The 8086 supports various data representations:

- Binary
- Decimal
- Hexadecimal
- Signed numbers
- Unsigned numbers

Example:

```
MOV AL, 25H
```

11. Addressing Modes (Introduction) سوف نتناولها بالتفصيل في المحاضرة ٥و٦

Addressing modes define how the operand of an instruction is accessed.

Common addressing modes in 8086:

1. Immediate addressing
2. Register addressing
3. Direct memory addressing
4. Register indirect addressing

Example:

```
MOV AX, 1234H   ; Immediate addressing
MOV AX, BX      ; Register addressing
```

12. Simple Example Program

Program to move a constant into a register:

```
MOV AX, 0005H
MOV BX, AX
```

Explanation:

- The first instruction loads AX with the value 0005H
- The second instruction copies the value of AX into BX

12. Importance of Assembly Language Programming

- Provides deep understanding of CPU architecture
- Essential for embedded systems and low-level programming
- Forms the basis for understanding compilers and operating systems

 Exam-Oriented Summary: Assembly language acts as a bridge between hardware and high-level languages and is tightly coupled with the internal architecture of the 8086 microprocessor.

Check your understanding

? (Easy – 10 Questions)

Q1. What is microprocessor programming?
A) Designing hardware circuits
B) Writing instructions to control microprocessor operation
C) Compiling high-level languages
D) Debugging operating systems
 Correct Answer: B

Q2. Which language is directly understood by the CPU?
A) Assembly language
B) High-level language
C) Machine language
D) Pseudocode
 Correct Answer: C

Q3. Assembly language instructions are represented using:
A) Binary codes
B) Mnemonics
C) Flowcharts
D) Electrical signals
 Correct Answer: B

Q4. Which of the following is an example of an assembly instruction?
A) 10101010
B) MOV AL, 05H
C) int main()
D) LOAD R1
 Correct Answer: B

Q5. What program translates assembly language into machine language?
A) Compiler
B) Interpreter
C) Assembler
D) Loader
 Correct Answer: C

Q6. Which of the following is a characteristic of machine language?
A) Easy to debug
B) Uses mnemonics
C) Human-readable
D) Executed directly by CPU
 Correct Answer: D

Q7. Which symbol is used to write comments in 8086 assembly language?
A) //
B) #
C) ;
D) /* */
 Correct Answer: C

Q8. Which instruction is used to terminate an 8086 program?
A) JMP
B) RET
C) INT 20H
D) CALL
 Correct Answer: C

Q9. What does the opcode represent in an instruction?
A) Data
B) Address
C) Operation
D) Register
 Correct Answer: C

Q10. Which register stores executable instructions?
A) DS

B) SS
C) CS
D) ES
 Correct Answer: C

(Medium – 10 Questions)

Q11. Why is assembly language easier to use than machine language?
A) It executes faster
B) It uses symbolic mnemonics
C) It requires less memory
D) It hides hardware details
 Correct Answer: B

Q12. What is the main role of the ORG directive?
A) End program execution
B) Allocate stack memory
C) Specify starting address offset
D) Define variables
 Correct Answer: C

Q13. Labels in assembly language are mainly used for:
A) Storing data
B) Improving execution speed
C) Referencing instruction addresses
D) Generating machine code
 Correct Answer: C

Q14. Which of the following does NOT generate machine code?
A) MOV AX, BX
B) ADD AX, BX
C) Label
D) JMP START
 Correct Answer: C

Q15. What happens if a program does not terminate properly?
A) Faster execution
B) Assembler error
C) Predictable behavior
D) Unpredictable behavior
 Correct Answer: D

Q16. Which of the following is considered an assembler directive?
A) MOV
B) ADD
C) ORG
D) JMP
 Correct Answer: C

Q17. What is the function of comments in assembly language?
A) Speed up execution
B) Affect instruction flow
C) Improve code readability
D) Generate object code
 Correct Answer: C

Q18. Which of the following is a data transfer instruction?
A) ADD
B) SUB
C) MOV
D) INC
 Correct Answer: C

Q19. In the instruction `MOV AX, BX`, AX and BX are:
A) Opcodes
B) Operands
C) Directives
D) Labels
 Correct Answer: B

Q20. Which segment stores return addresses during program execution?
A) CS
B) DS
C) ES
D) SS
 Correct Answer: D

(Hard – 10 Questions)

Q21. Which statement best describes the relationship between assembly language and hardware?
A) Assembly language is hardware-independent
B) Assembly language hides processor architecture
C) Assembly language directly reflects CPU architecture
D) Assembly language replaces machine language
 Correct Answer: C

Q22. Why is the ORG directive critical for correct program execution?
A) It initializes registers
B) It defines instruction size
C) It ensures correct memory addressing
D) It terminates the program
 Correct Answer: C

Q23. What would happen if labels generated machine code?
A) Faster execution
B) Incorrect memory allocation
C) Improved readability
D) No effect
 Correct Answer: B

Q24. Which component is responsible for fetching and executing instructions sequentially?
A) Assembler
B) Operating system
C) Instruction pointer and CPU
D) Compiler
 Correct Answer: C

Q25. In immediate addressing mode, the operand is:
A) Stored in a register
B) Located in memory
C) Given directly in the instruction
D) Found using an offset
 Correct Answer: C

Q26. Which of the following is NOT a valid data type in 8086 assembly?
A) Hexadecimal
B) Binary
C) Floating-point
D) Signed numbers
 Correct Answer: C

Q27. The instruction `MOV AL, 25H` transfers:
A) Decimal data
B) Binary data
C) Hexadecimal immediate data
D) Memory address
 Correct Answer: C

Q28. Why is assembly language essential for embedded systems?
A) Uses high-level abstraction

B) Provides direct hardware control
C) Requires large memory
D) Is platform-independent
 Correct Answer: B

Q29. Which segment is mandatory for execution of any instruction?
A) DS
B) ES
C) SS
D) CS
 Correct Answer: D

Q30. Assembly language is best described as:
A) Replacement for machine language
B) A bridge between hardware and high-level languages
C) A type of compiler
D) An operating system component
 Correct Answer: B