University of Technology- Computer Science

Material: Microprocessors

Stage: Second Class

Year: 2024-2025

Semester :2nd course

Assist lecturer Mohammed Thamer

Lect1 & 2: Load & Move Instructions with Practical Examples

1. Introduction to Load & Move Instructions

The **MOV** instruction is the primary instruction used for data transfer in **8086**. It moves data between registers, memory, or I/O ports without altering the value.

MOV Instruction Format:

```
MOV destination, source
```

- Transfers **source operand** to **destination operand**.
- Does **not** modify any **flag registers**.

Rules for MOV Instruction:

- **Both operands cannot be memory locations**.
- **Operands must be of the same size** (8-bit or 16-bit).
- **Immediate values cannot be moved directly to segment registers** (e.g., `MOV DS, 1234H` is invalid).

2. Register Addressing and Examples

Example 1: Moving Data Between Registers

```
MOV AX, BX   ; Copy the value from BX to AX
MOV CL, DL   ; Copy the value from DL to CL
MOV DX, 1234H ; Load immediate value into DX
```

Example 2: Moving Data from Memory to Register

```
MOV AX, [2000H]  ; Move data from memory location 2000H to AX
MOV [3000H], BX  ; Store BX value into memory location 3000H
```

Example 3: Moving Immediate Value to Register

```
MOV CX, 0F5H  ; Load CX with immediate value F5H
MOV AX, 45H   ; Load AX with immediate value 45H
```

4. Direct vs Indirect Addressing

- **Direct Addressing:** Uses a **specific memory address** in the instruction.

- **Indirect Addressing:** Uses a **register** to hold the address pointing to memory.

Example 4: Direct Addressing

```
MOV AL,[5000H] ;Load value from memory address 5000H into AL
```

Example 5: Indirect Addressing

```
MOV SI, 5000H  ; Load address into SI
MOV AL, [SI]   ; Load value from memory location pointed by SI
```

5. Extended Examples and Practical Applications

Example 6: Copying Data Between Registers

```
MOV AX, 1234H
MOV BX, AX   ; Copy AX to BX
```

Example 7: Using Immediate Value for Data Transfer

```
MOV DX, 0A1H   ; Load DX with immediate value
MOV CX, 0B2H   ; Load CX with immediate value
```

Example 8: Moving Data Between Memory Locations Using Indirect Addressing

```
MOV SI, 2000H
MOV DI, 3000H
MOV AX, [SI]  ; Load value from memory location 2000H
MOV [DI], AX  ; Store the loaded value at memory location 3000H
```

6. Exercises and Solutions

Question 1: Moving Data Between Registers

Q: Write a program to move data from **AX to BX** and from **CX to DX**.

Solution:

```
MOV BX, AX  ; Move data from AX to BX
MOV DX, CX  ; Move data from CX to DX
```

Question 2: Using Immediate Values

Q: Load **0A1H** into DX and **0B2H** into CX.

Solution:

```
MOV DX, 0A1H   ; Load DX with immediate value
MOV CX, 0B2H   ; Load CX with immediate value
```

Question 3: Direct vs Indirect Addressing

Q: Move the value stored at memory address **5000H** into **AL** using **direct addressing**, and then use **indirect addressing** to load the same value.

Solution:

```
MOV AL, [5000H]  ; Direct addressing
MOV SI, 5000H    ; Load address into SI
MOV AL, [SI]     ; Indirect addressing
```

Question 4: Moving Data Between Memory Locations

Q: Write a program to copy a block of data from one memory location to another.

Solution:

```
MOV SI, 2000H  ; Source memory location
MOV DI, 3000H  ; Destination memory location
MOV AX, [SI]   ; Load data from source
MOV [DI], AX   ; Store data at destination
```

Lect3: Arithmetic Instructions (ADD, SUB, MUL, DIV)

1. Introduction to Arithmetic Instructions

Arithmetic instructions allow the processor to perform addition, subtraction, multiplication, and division.

Common Arithmetic Instructions:

Instruction	Description
`ADD dest, src`	Adds `src` to `dest`, result stored in `dest`.
`SUB dest, src`	Subtracts `src` from `dest`, result stored in `dest`.
`MUL src`	Multiplies `AX` by `src` (for 16-bit), or `AL` by `src` (for 8-bit).
`IMUL src`	Signed multiplication.

`DIV src`	Unsigned division. Quotient in `AX`, remainder in `DX` (for 16-bit).
`IDIV src`	Signed division.

2. Arithmetic Operations and Examples

Example 1: Addition of Two Registers

```
MOV AX, 05H  ; Load AX with 5
MOV BX, 03H  ; Load BX with 3
ADD AX, BX   ; AX = AX + BX (AX = 5 + 3 = 8)
```

Example 2: Subtraction of Two Registers

```
MOV AX, 09H  ; Load AX with 9
MOV BX, 04H  ; Load BX with 4
SUB AX, BX   ; AX = AX - BX (AX = 9 - 4 = 5)
```

Example 3: Multiplication of Two Numbers

```
MOV AL, 04H  ; Load AL with 4
MOV BL, 02H  ; Load BL with 2
MUL BL       ; AL = AL * BL (AL = 4 * 2 = 8)
```

Example 4: Division of Two Numbers

```
MOV AX, 10H  ; Load AX with 16
MOV BL, 04H  ; Load BL with 4
DIV BL       ; AX = AX / BL (Quotient in AL, Remainder in AH)
```

4. Exercises and Solutions

Question 1: Perform Addition of Immediate Values

Q: Write a program to add **0AH** and **05H** and store the result in AX.

Solution:

```
MOV AX, 0AH  ; Load AX with A (10 in decimal)
ADD AX, 05H  ; AX = AX + 5
```

Question 2: Subtraction Using Registers

Q: Subtract **07H** from **15H** and store the result in BX.

Solution:

```
MOV BX, 15H  ; Load BX with 15
SUB BX, 07H  ; BX = BX - 7
```

Question 3: Multiply Two Numbers Using MUL

Q: Multiply **03H** and **05H**, store the result in AX.

Solution:

```
MOV AL, 03H  ; Load AL with 3
MOV BL, 05H  ; Load BL with 5
MUL BL       ; AL = AL * BL
```

Question 4: Division of Numbers

Q: Divide **20H** by **04H**, store the quotient in AL and remainder in AH.

Solution:

```
MOV AX, 20H  ; Load AX with 32 (decimal)
MOV BL, 04H  ; Load BL with 4
DIV BL       ; Quotient in AL, Remainder in AH
```

Lect4: Examples of Arithmetic Instructions & Jump Operations

1. Introduction to Jump Instructions

Jump instructions allow control flow changes in assembly programming. These instructions alter the execution sequence based on specific conditions.

Types of Jump Instructions:

Instruction	Description
`JMP label`	Unconditional jump to a specified label.
`JZ label`	Jump if the Zero flag (ZF) is set.
`JNZ label`	Jump if the Zero flag (ZF) is not set.
`JE label`	Jump if Equal (same as JZ).
`JNE label`	Jump if Not Equal (same as JNZ).
`JL label`	Jump if Less (signed comparison).
`JG label`	Jump if Greater (signed comparison).

2. Arithmetic Operations and Jumps with Examples

Example 1: Addition and Conditional Jump

```
MOV AX, 5      ; Load AX with 5
MOV BX, 3      ; Load BX with 3
ADD AX, BX     ; AX = AX + BX
CMP AX, 10     ; Compare AX with 10
JNZ NOT_EQUAL ; Jump if AX is not equal to 10
```

Example 2: Subtraction and Jump if Zero

```
MOV AX, 10     ; Load AX with 10
MOV BX, 10     ; Load BX with 10
SUB AX, BX     ; AX = AX - BX (AX = 10 - 10 = 0)
JZ EQUAL       ; Jump to EQUAL if AX is zero
```

Example 3: Looping Using Jumps

```
MOV CX, 5      ; Counter
LOOP_START:
DEC CX         ; Decrease CX by 1
JNZ LOOP_START ; Repeat loop if CX is not zero
```

---

3. Exercises and Solutions

Question 1: Compare Two Numbers and Jump

Q: Write a program to compare two values and jump to a label if they are equal.

Solution:

```
MOV AX, 10     ; Load AX with 10
MOV BX, 10     ; Load BX with 10
CMP AX, BX     ; Compare AX with BX
JE EQUAL       ; Jump if AX == BX
```

Question 2: Loop Execution Until Zero

Q: Write a program to decrement a counter and loop until it reaches zero.

Solution:

```
MOV CX, 5      ; Initialize loop counter
```

```
LOOP_LABEL:
DEC CX          ; Decrement CX
JNZ LOOP_LABEL ; Repeat if CX is not zero
```

Question 3: Conditional Jump Based on Addition Result

Q: Add **05H** and **07H** and jump if the result is greater than **0AH**.

Solution:

```
MOV AL, 05H
ADD AL, 07H
CMP AL, 0AH
JG GREATER
```

Lect5: Logical Instructions, Shift & Rotate Operations

1. Logical Instructions Overview

Logical operations allow manipulation of individual bits within a byte or word.

Common Logical Instructions:

Instruction	Description
`AND dest, src`	Performs bitwise AND between `dest` and `src`.
`OR dest, src`	Performs bitwise OR between `dest` and `src`.
`XOR dest, src`	Performs bitwise XOR (exclusive OR) between `dest` and `src`.
`NOT dest`	Inverts all bits in `dest`.
`SHL dest, n`	Shifts bits of `dest` left by `n` places.
`SHR dest, n`	Shifts bits of `dest` right by `n` places.
`ROL dest, n`	Rotates bits of `dest` left by `n` places.
`ROR dest, n`	Rotates bits of `dest` right by `n` places.

---

3. Logical Operations and Examples

Example 1: Bitwise AND Operation

```
MOV AL, 0F0H  ; Load AL with 11110000B
AND AL, 0F5H  ; AL = AL AND 11110101B
```

Example 2: Bitwise OR Operation

```
MOV AL, 0A0H  ; Load AL with 10100000B
OR AL, 05H    ; AL = AL OR 00000101B
```

Example 3: Logical Shift Left

```
MOV AL, 05H   ; Load AL with 00000101B
SHL AL, 1     ; AL = AL shifted left by 1 (00001010B)
```

Lect6: Examples of Logical Instructions

1. Practical Applications and Examples

Example 4: Bitwise XOR Operation

```
MOV AL, 0FFH  ; Load AL with 11111111B
XOR AL, 0F0H  ; AL = AL XOR 11110000B
```

Example 5: Logical NOT Operation

```
MOV AL, 0F0H  ; Load AL with 11110000B
NOT AL        ; AL = NOT AL (00001111B)
```

Example 6: Rotate Left Operation

```
MOV AL, 85H   ; Load AL with 10000101B
ROL AL, 1     ; Rotate left by 1 (00001011B)
```

2. Exercises and Solutions

Question 1: Bitwise AND Operation

Q: Perform a bitwise AND operation between `0AAH` and `0F0H`.

Solution:

```
MOV AL, 0AAH  ; Load AL with 10101010B
AND AL, 0F0H  ; AL = AL AND 11110000B
```

Question 2: Shift Left Operation

Q: Shift the value `3CH` left by 2 bits and store the result in AL.

Solution:

```
MOV AL, 3CH    ; Load AL with 00111100B
SHL AL, 2      ; Shift AL left by 2 (11110000B)
```

Question 3: Rotate Right Operation

Q: Rotate the value `0C3H` right by 1 bit.

Solution:

```
MOV AL, 0C3H   ; Load AL with 11000011B
ROR AL, 1      ; Rotate right by 1 (01100001B)
```

Lect7: The Addressing Mode in 8-bit Register

1. Addressing Modes Overview

Addressing modes determine how an instruction accesses operands in memory or registers.

Types of Addressing Modes in 8-bit Registers:

Addressing Mode	Description
Immediate	Operand is specified directly in the instruction.
Register	Operand is in a register.
Direct	Address of the operand is specified in the instruction.
Indirect	Register holds the address of the operand.
Indexed	Uses an index register (SI or DI) to determine the operand's address.

2. Addressing Mode Examples

Example 1: Immediate Addressing Mode

```
MOV AL, 0AH    ; Load AL with immediate value 0AH
```

Example 2: Register Addressing Mode

```
MOV BL, AL    ; Copy the value from AL to BL
```

Example 3: Direct Addressing Mode

```
MOV AL, [2000H]  ; Load AL from memory location 2000H
```

Example 4: Indirect Addressing Mode

```
MOV SI, 3000H  ; Load address 3000H into SI
MOV AL, [SI]   ; Load AL with value from memory location 3000H
```

Example 5: Indexed Addressing Mode

```
MOV SI, 2000H  ; Base address
MOV AL, [SI+5] ; Load AL with value at address 2005H
```

4. Exercises and Solutions

Question 1: Immediate and Register Addressing

Q: Load **05H** into AL and copy it to BL.

Solution:

```
MOV AL, 05H   ; Load AL with 05H
MOV BL, AL    ; Copy AL to BL
```

Question 2: Direct Addressing Mode

Q: Store the value **0AH** at memory address **3000H**.

Solution:

```
MOV AL, 0AH
MOV [3000H], AL ; Store AL value at memory 3000H
```

Question 3: Indirect Addressing Mode

Q: Load the value from memory location **4000H** into AL using SI.

Solution:

```
MOV SI, 4000H
MOV AL, [SI]
```

Lect8: Examples of Direct and Immediate Register Addressing

1. Addressing Mode Examples

Example 1: Immediate Addressing Mode

```
MOV AL, 0AH   ; Load AL with immediate value 0AH
MOV BX, 1234H ; Load BX with immediate value 1234H
```

Example 2: Direct Addressing Mode

```
MOV AL, [2000H]  ; Load AL from memory location 2000H
MOV [3000H], AL  ; Store AL value at memory location 3000H
```

Example 3: Register Addressing Mode

```
MOV DL, AL   ; Copy AL to DL
MOV CX, DX   ; Copy DX to CX
```

2. Exercises and Solutions

Question 1: Direct and Immediate Addressing

Q: Load **0FH** into AL and store it in memory location **4000H**.

Solution:

```
MOV AL, 0FH   ; Load AL with immediate value
MOV [4000H], AL ; Store AL value at memory 4000H
```

Question 2: Register to Register Transfer

Q: Move the value in **AX** to **BX**, then move it to **DX**.

Solution:

```
MOV BX, AX    ; Copy AX to BX
MOV DX, BX    ; Copy BX to DX
```

Question 3: Memory to Register Transfer

Q: Load a value from memory **5000H** into AL, then move it to CL.

Solution:

```
MOV AL, [5000H]  ; Load value from memory 5000H into AL
MOV CL, AL       ; Move AL value to CL
```

Lect9: The Addressing Mode in 16-bit Register

1. Addressing Modes Overview

Addressing modes determine how an instruction accesses operands in memory or registers.

Types of Addressing Modes in 16-bit Registers:

Addressing Mode	Description
Immediate	Operand is specified directly in the instruction.
Register	Operand is in a register.
Direct	Address of the operand is specified in the instruction.
Indirect	Register holds the address of the operand.
Base	Uses a base register (BX or BP) to access memory.
Index	Uses an index register (SI or DI) to determine the operand's address.
Base-Index	Combines base and index registers for memory addressing.

2. Addressing Mode Examples

Example 1: Immediate Addressing Mode

```
MOV AX, 1234H   ; Load AX with immediate value 1234H
```

Example 2: Register Addressing Mode

```
MOV DX, AX   ; Copy AX to DX
```

Example 3: Direct Addressing Mode

```
MOV AX, [2000H]  ; Load AX from memory location 2000H
```

Example 4: Indirect Addressing Mode

```
MOV BX, 3000H  ; Load address 3000H into BX
MOV AX, [BX]   ; Load AX with value from memory location 3000H
```

Example 5: Base-Index Addressing Mode

```
MOV BX, 2000H  ; Base address
MOV SI, 10H    ; Offset
MOV AX, [BX+SI] ; Load AX with value at address (2000H + 10H)
```

---

3. Exercises and Solutions

Question 1: Immediate and Register Addressing

Q: Load **1234H** into AX and move it to BX.

Solution:

```
MOV AX, 1234H  ; Load AX with immediate value
MOV BX, AX     ; Copy AX to BX
```

Question 2: Direct Addressing Mode

Q: Load a value from memory **4000H** into AX.

Solution:

```
MOV AX, [4000H]  ; Load AX from memory 4000H
```

Question 3: Base-Index Addressing Mode

Q: Load a value from **[3000H + SI]** into AX, where **SI = 10H**.

Solution:

```
MOV BX, 3000H
MOV SI, 10H
MOV AX, [BX+SI]
```

Lect10: Examples of Direct, Indirect, Base, Index, and Base-Index Register Addressing

1. Addressing Modes in Practice

Addressing Mode	Description
Direct	Operand is stored at a specific memory address.
Indirect	Operand is stored at a memory address pointed to by a register.
Base	Uses a base register (BX, BP) for memory access.
Index	Uses an index register (SI, DI) for offset memory access.
Base-Index	Combines base and index registers for complex memory access.

2. Addressing Mode Examples

Example 1: Direct Addressing Mode

```
MOV AX, [2000H]  ; Load AX from memory location 2000H
```

Example 2: Indirect Addressing Mode

```
MOV BX, 3000H  ; Load address 3000H into BX
MOV AX, [BX]   ; Load AX with value from memory location 3000H
```

Example 3: Base-Index Addressing Mode

```
MOV BX, 2000H  ; Base address
MOV SI, 10H    ; Offset
MOV AX, [BX+SI] ; Load AX with value at address (2000H + 10H)
```

---

3. Exercises and Solutions

Question 1: Direct Addressing Mode

Q: Load a value from memory **5000H** into AX.

Solution:

```
MOV AX, [5000H]  ; Load AX from memory 5000H
```

Question 2: Base-Index Addressing Mode

Q: Load a value from **[3000H + SI]** into AX, where **SI = 10H**.

Solution:

```
MOV BX, 3000H
MOV SI, 10H
MOV AX, [BX+SI]
```

Lect11: The Addressing Mode in 32-bit Register

1. Addressing Modes in 32-bit Registers

32-bit addressing modes in the **8086 microprocessor** provide more flexibility and allow efficient memory access. The common addressing modes are:

Addressing Mode	Description

Direct Addressing	The operand is stored at a specific memory address mentioned in the instruction.
Register Addressing	The operand is stored in a 32-bit register.
Indirect Addressing	A register contains the memory address of the operand.
Base Addressing	A base register (EBX, EBP) holds the base memory address.
Index Addressing	An index register (ESI, EDI) is used for offset memory access.
Base-Index Addressing	Combines a base register and an index register for complex memory operations.

---

2. Addressing Mode Examples in 32-bit Registers

Example 1: Direct Addressing Mode

```
MOV EAX, [2000H]  ; Load EAX from memory location 2000H
```

Example 2: Register Addressing Mode

```
MOV ECX, EAX  ; Copy value from EAX to ECX
```

Example 3: Indirect Addressing Mode

```
MOV EBX, 3000H  ; Load address 3000H into EBX
MOV EAX, [EBX]  ; Load EAX with value from memory location 3000H
```

Example 4: Base Addressing Mode

```
MOV EBP, 4000H  ; Load base address into EBP
MOV EAX, [EBP]  ; Load EAX with value from base address
```

Example 5: Base-Index Addressing Mode

```
MOV EBX, 5000H  ; Load base address into EBX
MOV ESI, 20H    ; Load offset into ESI
MOV EAX, [EBX+ESI] ; Load value at address (5000H + 20H) into
EAX
```

Lect12: Examples of Direct, Indirect, Base, Index, and Base-Index Register Addressing in 32-bit Registers

1. Addressing Modes in Practice (32-bit Registers)

Addressing Mode	Description
Direct	Operand is stored at a specific memory address.
Indirect	Operand is stored at a memory address pointed to by a register.
Base	Uses a base register (EBX, EBP) for memory access.
Index	Uses an index register (ESI, EDI) for offset memory access.
Base-Index	Combines base and index registers for complex memory access.

2. Additional Addressing Mode Examples in 32-bit Registers

Example 6: Indexed Addressing Mode

```
MOV EDI, 6000H  ; Load base address into EDI
MOV EAX, [EDI+10H] ; Load value at address (6000H + 10H) into
EAX
```

Example 7: Complex Base-Index Addressing Mode

```
MOV EBX, 7000H  ; Load base address into EBX
MOV ESI, 30H    ; Load offset into ESI
MOV EAX, [EBX+ESI] ; Load value at address (7000H + 30H) into
EAX
```

3. Exercises and Solutions

Question 1: Using Indirect Addressing

Q: Load a value from memory location **8000H** into `EAX` using indirect addressing.

Solution:

```
MOV EBX, 8000H  ; Load memory address into EBX
MOV EAX, [EBX]  ; Load value from memory into EAX
```

Question 2: Using Base-Index Addressing

Q: Load a value from **[9000H + SI]** into `EAX`, where `SI = 50H`.

Solution:

```
MOV EBX, 9000H  ; Load base address into EBX
MOV ESI, 50H    ; Load index offset into ESI
MOV EAX, [EBX+ESI] ; Load value at address (9000H + 50H) into
EAX
```

Lect13: Bit Scan and Bit Test Register

1. Bit Scan and Bit Test Instructions

Bit Scan Instructions:

Instruction	Description
`BSF dest, src`	Finds the first set bit (1) in `src` and stores the position in `dest`.
`BSR dest, src`	Finds the last set bit (1) in `src` and stores the position in `dest`.

Bit Test Instructions:

Instruction	Description
`BT src, bit`	Tests a specific bit in `src` and sets Carry Flag (CF) if it is 1.
`BTC src, bit`	Tests and complements (toggles) a specific bit in `src`.
`BTR src, bit`	Tests and resets (clears) a specific bit in `src`.
`BTS src, bit`	Tests and sets (turns on) a specific bit in `src`.

2. Bit Scan and Bit Test Examples

Example 1: Using BSF to Find First Set Bit

```
MOV EAX, 0B8H  ; Binary: 10111000B
BSF ECX, EAX   ; ECX = Position of first set bit (3)
```

Example 2: Using BT to Test a Bit

```
MOV AX, 5      ; Binary: 00000101B
BT AX, 2       ; Test bit 2 (CF = 1 since bit 2 is set)
```

Example 3: Using BTS to Set a Bit

```
MOV BX, 0H     ; BX = 00000000B
BTS BX, 4      ; Set bit 4 (BX = 00010000B)
```

Example 4: Using BTR to Clear a Bit

```
MOV DX, 0FH    ; DX = 00001111B
BTR DX, 3      ; Clear bit 3 (DX = 00000111B)
```

Lect14: Examples of Bit Scan and Bit Test Instructions

1. Advanced Examples and Practical Applications

Example 5: Using BSR to Find the Highest Set Bit

```
MOV EAX, 0F0H  ; Binary: 11110000B
BSR ECX, EAX   ; ECX = Position of highest set bit (7)
```

Example 6: Toggling a Bit with BTC

```
MOV AX, 4      ; AX = 00000100B
BTC AX, 2      ; Toggle bit 2 (AX = 00000000B)
BTC AX, 2      ; Toggle bit 2 again (AX = 00000100B)
```

Example 7: Using Bit Test for Conditional Execution

```
MOV CX, 2      ; CX = 00000010B
BT CX, 1       ; Test bit 1
JC BIT_IS_SET  ; Jump if bit is set
```

Example 8: Clearing a Bit with BTR and Checking Flag

```
MOV DX, 3      ; DX = 00000011B
BTR DX, 0      ; Clear bit 0 (DX = 00000010B)
JNC BIT_CLEARED ; Jump if bit was already clear
```

2. Exercises and Solutions

Question 1: Using BSF to Find First Set Bit

Q: Given `AX = 01011000B`, find the position of the first set bit.

Solution:

```
MOV AX, 58H   ; Binary: 01011000B
BSF BX, AX    ; BX = Position of first set bit (3)
```

Question 2: Toggling a Bit

Q: Toggle bit 5 of register `CX` using BTC.

Solution:

```
MOV CX, 20H   ; Binary: 00100000B
BTC CX, 5     ; Toggle bit 5 (CX = 00000000B)
```

Question 3: Testing and Setting a Bit

Q: Test bit 7 of `DX`, and if it is 0, set it using `BTS`.

Solution:

```
MOV DX, 10H    ; Binary: 00010000B
BT DX, 7       ; Test bit 7
JNC SET_BIT    ; Jump if bit is clear
BTS DX, 7      ; Set bit 7 (DX = 10010000B)
```

Lect15: General Examples

1. Comprehensive Assembly Programming Examples

Example 1: Arithmetic and Logical Operations Combined

```
MOV AX, 5       ; Load AX with 5
ADD AX, 3       ; AX = AX + 3 (AX = 8)
AND AX, 0FH     ; Apply bitmask (AX = 8 AND 15 = 8)
```

Example 2: Looping with Conditional Execution

```
MOV CX, 5       ; Set loop counter
LOOP_START:
DEC CX          ; Decrease counter
JNZ LOOP_START ; Repeat if CX is not zero
```

Example 3: String Manipulation (Copying Data)

```
MOV SI, 1000H   ; Source address
MOV DI, 2000H   ; Destination address
MOV CX, 10      ; Number of bytes to copy
REP MOVSB       ; Repeat move operation for CX times
```

Example 4: Stack Operations and Subroutines

```
CALL SUB_ROUTINE  ; Call a subroutine
...
SUB_ROUTINE:
PUSH AX           ; Save AX on stack
MOV AX, 10H       ; Modify AX
POP AX            ; Restore AX
RET               ; Return to main program
```

Example 5: Interrupts and I/O Operations

```
MOV AH, 09H       ; DOS interrupt for printing string
MOV DX, OFFSET MESSAGE
INT 21H           ; Call DOS interrupt
...
MESSAGE DB 'Hello, World!$', 0
```

2. Exercises and Solutions

Question 1: Combining Arithmetic and Logical Operations

Q: Write an assembly program that adds **07H** and **09H**, then applies a bitmask `0FH` to the result.

Solution:

```
MOV AL, 07H     ; Load AL with 07H
ADD AL, 09H     ; AL = AL + 09H (AL = 10H)
AND AL, 0FH     ; Apply bitmask (AL = 10H AND 0FH = 0H)
```

Question 2: Implementing a Counter Using Loops

Q: Create an assembly program that decrements CX from 5 to 0 using a loop.

Solution:

```
MOV CX, 5       ; Set loop counter
LOOP_LABEL:
DEC CX          ; Decrease CX by 1
JNZ LOOP_LABEL ; Repeat if CX is not zero
```

Question 3: Using Stack Operations in Subroutines

Q: Implement a subroutine that saves **AX** and modifies it, then restores the original value.

Solution:

```
CALL MY_SUB     ; Call the subroutine
...
MY_SUB:
PUSH AX         ; Save AX on stack
MOV AX, 20H     ; Modify AX
POP AX          ; Restore original AX
```

```
RET
```

Question 4: Printing a String Using an Interrupt

Q: Write a program that prints `HELLO` using an interrupt.

Solution:

```
MOV AH, 09H    ; DOS interrupt for printing string
MOV DX, OFFSET HELLO_MSG
INT 21H        ; Call DOS interrupt
...
HELLO_MSG DB 'HELLO$', 0
```