TP Assembly 8086

MOV (Move)

- **Syntax:**

```
MOV destination, source
```

- **Operands:**
  - destination: Register or memory location
  - source: Register, immediate value, or memory location
- **Example usage:**

```
MOV AX, BX          ; AX = BX
MOV AX, 10          ; AX = 10
MOV AX, 0x0A        ; AX = 10 in Hex
MOV AX, 00001010b   ; AX = 10 in Binary
```

XCHG (Exchange)

- **Syntax:**

```
XCHG operand1, operand2
```

- **Operands:**
  - operand1: Register or memory location
  - operand2: Register or memory location
- **Example usage:**

```
XCHG AX, BX     ; AX , BX = BX , AX
```

ADD (Add)

- **Syntax:**

```
    ADD destination, source
```

- **Operands:**
  - destination: Register or memory location
  - source: Register, immediate value, or memory location
- **Example usage:**

```
    ADD AX, 5          ; AX = AX + 5
```

SUB (Subtract)

- **Syntax:**

```
    SUB destination, source
```

- **Operands:**
  - destination: Register or memory location
  - source: Register, immediate value, or memory location
- **Example usage:**

```
    SUB AX, 2          ; AX = AX - 2
```

INC (Increment)

- **Syntax:**

```
    INC operand
```

- **Operands:**
  - operand: Register or memory location
- **Example usage:**

```
    INC AX             ; AX = AX + 1
```

DEC (Decrement)

- **Syntax:**

```
    DEC operand
```

- **Operands:**
  - operand: Register or memory location
- **Example usage:**

```
    DEC BX          ; BX = BX - 1
```

AND (Bitwise AND)

- **Syntax:**

```
    AND destination, source
```

- **Operands:**
  - destination: Register or memory location
  - source: Register, immediate value, or memory location
- **Example usage:**

```
    AND AX, 0x0F
```

OR (Bitwise OR)

- **Syntax:**

```
    OR destination, source
```

- **Operands:**
  - destination: Register or memory location
  - source: Register, immediate value, or memory location
- **Example usage:**

```
    OR AX, 0xF0
```

XOR (Bitwise XOR)

- **Syntax:**

```
    XOR destination, source
```

- **Operands:**
  - destination: Register or memory location
  - source: Register, immediate value, or memory location
- **Example usage:**

```
    XOR AX, AX      ; Clear Register AX
```

NOT (Bitwise NOT)

- **Syntax:**

```
    NOT operand
```

- **Operands:**
  - operand: Register or memory location
- **Example usage:**

```
    NOT AX          ; Invert all bits in AX
```

CMP (Compare)

- **Syntax:**

```
    CMP operand1, operand2
```

- **Operands:**
  - operand1: Register or memory location
  - operand2: Register, immediate value, or memory location
- **Example usage:**

```
    CMP AX, BX      ; Compare AX and BX
```

MUL (Multiply Unsigned)

- **Syntax:**

```
MUL operand
```

- **Operands:**
  - operand: Register or memory location (8-bit or 16-bit)
- **Description:** Multiplies the accumulator (‘AL‘ or ‘AX‘) by the specified operand. The result is stored in ‘AX‘ for 8-bit operands, or in ‘DX:AX‘ for 16-bit operands.
- **Example usage:**

```
MOV AL, 5
MOV BL, 10
MUL BL          ; AX = AL * BL

MOV AX, 200
MOV BX, 10
MUL BX          ; DX:AX = AX * BX
```

IMUL (Multiply Signed)

- **Syntax:**

```
IMUL operand
```

- **Operands:**
  - operand: Register or memory location (8-bit or 16-bit)
- **Description:** Performs signed multiplication between the accumulator (‘AL‘ or ‘AX‘) and the specified operand. The result is stored in ‘AX‘ for 8-bit operands, or in ‘DX:AX‘ for 16-bit operands.
- **Example usage:**

```
MOV AL, -5
MOV BL, 10
IMUL BL         ; AX = AL * BL

MOV AX, -200
MOV BX, 10
IMUL BX         ; DX:AX = AX * BX
```

DIV (Divide Unsigned)

- **Syntax:**

```
    DIV operand
```

- **Operands:**
  - operand: Register or memory location (8-bit or 16-bit)
- **Description:** Divides the accumulator (‘AX‘ or ‘DX:AX‘) by the specified operand. For 8-bit division, the quotient is stored in ‘AL‘ and the remainder in ‘AH‘. For 16-bit division, the quotient is in ‘AX‘ and the remainder in ‘DX‘.
- **Example usage:**

```
    MOV AX, 100
    MOV BL, 10
    DIV BL          ; AL = 10, AH = 0

    MOV DX, 0
    MOV AX, 1000
    MOV BX, 50
    DIV BX          ; AX = 20, DX = 0
```

IDIV (Divide Signed)

- **Syntax:**

```
    IDIV operand
```

- **Operands:**
  - operand: Register or memory location (8-bit or 16-bit)
- **Description:** Performs signed division of the accumulator (‘AX‘ or ‘DX:AX‘) by the specified operand. For 8-bit division, the quotient is stored in ‘AL‘ and the remainder in ‘AH‘. For 16-bit division, the quotient is in ‘AX‘ and the remainder in ‘DX‘.
- **Example usage:**

```
    MOV AX, -100
    MOV BL, 10
    IDIV BL         ; AL = -10, AH = 0

    MOV DX, 0
```

```
MOV AX, -1000
MOV BX, 50
IDIV BX          ; AX = -20, DX = 0
```

SHL (Shift Left)

- **Syntax:**

```
SHL destination, count
```

- **Operands:**
  - destination: Register or memory location
  - count: Number of bit positions to shift (can be an immediate value or in the 'CL' register)
- **Description:** Shifts the bits of the destination operand left by the specified count. Zeros are shifted in from the right.
- **Example usage:**

```
MOV AL, 3
SHL AL, 1        ; AL = 6 (3 * 2^1)
```

SHR (Shift Right)

- **Syntax:**

```
SHR destination, count
```

- **Operands:**
  - destination: Register or memory location
  - count: Number of bit positions to shift (can be an immediate value or in the 'CL' register)
- **Description:** Shifts the bits of the destination operand right by the specified count. Zeros are shifted in from the left.
- **Example usage:**

```
MOV AL, 8
SHR AL, 1        ; AL = 4 (8 / 2^1)
```

JMP (Unconditional Jump)

- **Syntax:**

```
    JMP label
```

- **Operands:**
  - label: The label where execution will continue
- **Example usage:**

```
    JMP START        ; Jump to the label "START"

    ; some code here

    START:           ; Label definition
    MOV AX, 5        ; Execution resumes here
```