CSE-3103: Microprocessor and Microcontroller

Dept. of Computer Science and Engineering
University of Dhaka

Prof. Sazzad M.S. Imran, PhD
Dept. of Electrical and Electronic Engineering
sazzadmsi.webnode.com

Instruction Set of 8086

8086 → 117 basic instructions.

Groups of instruction set →

(i) Data transfer instructions

(ii) Arithmetic instructions

(iii) Logic instructions

(iv) Shift instructions

(v) Rotate instructions

(vi) Flag control instructions

(vii) Compare instructions

(viii) Jump instructions

(ix) Subroutines and subroutine handling instructions

(x) Loop and loop handling instructions

(xi) Strings and string handling instructions.

Data Transfer Instructions

Different types of data transfer instructions →

(i) Move byte or word (MOV)

(ii) Exchange byte or word (XCHG)

(iii) Translate byte (XLAT)

(iv) Load effective address (LEA)

(v) Load data segment (LDS)

(vi) Load extra segment (LES).

Data Transfer Instructions

MOV →

Copies [2nd operand] into [1st operand].

Source operand →

register, memory, constant value.

Destination operand →

register or memory.

Direct memory-to-memory moves are not allowed.

Example →

MOV CX, AX

CX ← AX

If AX = 1234H, then

CX = 1234H, CH = 12H, CL = 34H.

MOV AX, [ALPHA]

Let, DS = 0300H, ALPHA = 1234H.

Then PA = 03000H + 1234H = 04234H

AL ← [04234H], AH ← [04235H].

Data Transfer Instructions

XCHG →

Swaps source-operand with destination-operand.

It’s like doing 3 move operations →

i) from destination to temporary register, then
ii) from source to destination, then
iii) from temporary register to source.

No register needs to be reserved for temporary storage.

Source and destination operands →

register or memory.
only one operand can be in memory,
other must be register.

No FLAGS are modified by this instruction.

Example →

XCHG BX, CX

Interchanges contents of BX and CX,
BX ← CX and CX ← BX

Data Transfer Instructions

XLAT →

Performs table lookup.
Takes no operands.

BX ← offset or starting address of table.
AL ← position of byte in table.
invoke XLAT instruction.
AL ← byte at specified position in table.

In other words,
AL ← [AL+BX]

```
DATA   SEGMENT
       HA_TABLE     DB     '0123456789ABCDEF'
       H_DIGIT      DB     7
       ASC_DIGIT    DB     ?
DATA   ENDS

MAIN   PROC   FAR
       MOV    BX, OFFSET  HA_TABLE
       MOV    AL, H_DIGIT
       XLAT
       MOV    ASC_DIGIT, AL
       RET
MAIN   ENDP
```

Data Transfer Instructions

LEA, LDS and LES →

LEA dest, src →

LEA = Load Effective Address.
[destination operand] ← effective address of source operand.
Source operand = offset part of memory address.
Destination operand = general-purpose register.
LEA loads pointer to addressing item,
MOV loads actual value at that address.

```
mov dx, offset msg
lea dx, msg
```

LDS/LES →

copies word from two memory locations into specified register.
It then copies word from next two memory locations into DS/ES register.

LDS dest, src →

source operand is memory address of 1st word and destination operand is register.

LDS BX, [4326]

BL ← [DS×10H + 4326H], and BH ← [DS×10H + 4327H].
DS ← [DS×10H + 4328H] : [DS×10H + 4329H].

Arithmetic Instructions

INC/DEC →

Used for incrementing/decrementing operand by 1.
It works on single operand.
Operand can be either in register or in memory.

Syntax →

```
INC/DEC      destination
```

Destination could be 8-bit or 16-bit.

Example →

```
INC/DEC      CX
INC/DEC      DL
INC/DEC      [count]
```

Arithmetic Instructions

ADD/SUB →

Performs addition/subtraction of binary data.
Adding or subtracting 8-bit or 16-bit operands.

Syntax →

```
ADD/SUB         destination, source
```

ADD/SUB instruction can take place between →

Register to register,
Memory to register,
Register to memory,
Register to constant data,
Memory to constant data

Memory-to-memory operations are not allowed.
ADD or SUB operation sets or clears OF and CF.

Example →

store two digits in AX and BX registers, and add the values.

```
mov     ax, [num1]
mov     bx, [num2]
add     ax, bx
```

Arithmetic Instructions

MUL/IMUL →

Multiply binary data.
MUL = Multiply → handles unsigned data.
IMUL = Integer Multiply → handles signed data.
Both instructions affect CF and OF.

Syntax →

```
MUL/IMUL    multiplier
```

AL/AX ← Multiplicand.
Register/Memory ← Multiplier.
AX/DX:AX ← Generated product.

Example →

```
MOV     AL, 10
MOV     DL, 25
MUL     DL

; signed multiplication
MOV     DL, 0FFH        [dl = -1]
MOV     AL, 0BEH        [al = -66]
IMUL    DL
```

Two different cases →

(i) 2 bytes are multiplied:
AL ← multiplicand
Memory/register ← multiplier (byte)
AX ← product (word)

(ii) 2 words are multiplied:
AX ← multiplicand
Memory/register ← multiplier (word)
DX:AX ← product (double-word)