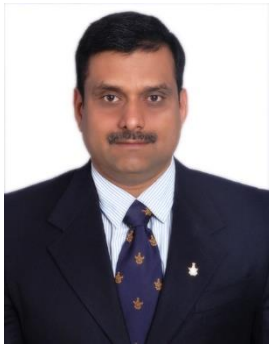
Microprocessors and Interfaces: 2021-22
Lecture 8
8086 Instructions Set : Part-3

By Dr. Sanjay Vidhyadharan



Data Transfer Instructions

Flag Register Data transfer

- LAHF : Load AH register from flags
- SAHF : Store AH register in flags
- PUSHF : Push flags onto stack
- POPF : Pops flags off stack

LAHF

- LAHF instruction transfers the rightmost 8 bits of the flag register into the AH register.
- Copies SF, ZF, AF, PF and CF into bits 7 ,6 ,4 ,2 and 0, respectively of AH.
- Contents of 5 ,3 ,1 are undefined.
- Can be used to observe the status of all conditional flags except the overflow flag.

Flag Register

LAHF

1 0 0 1 1 1 1 1

Machine code format

Control Flags

Control Flags – The control flags enable or disable certain operations of the microprocessor. There are 3 control flags in 8086 microprocessor and these are:

1.Directional Flag (D) – This flag is specifically used in string instructions.
If directional flag is set (1), then access the string data from higher memory location towards lower memory location. **(STD/CLD)**
If directional flag is reset (0), then access the string data from lower memory location towards higher memory location.

2.Interrupt Flag (I) – This flag is for interrupts.
If interrupt flag is set (1), the microprocessor will recognize interrupt requests from the peripherals. **(STI/CLI)**
If interrupt flag is reset (0), the microprocessor will not recognize any interrupt requests and will ignore them.

3.Trap Flag (T) – This flag is used for on-chip debugging. Setting trap flag puts the microprocessor into single step mode for debugging. In single stepping, the microprocessor executes a instruction and enters into single step ISR. **(POP)**
If trap flag is set (1), the CPU automatically generates an internal interrupt after each instruction, allowing a program to be inspected as it executes instruction by instruction.
If trap flag is reset (0), no function is performed.

SAHF

- SAHF instruction transfers the AH register into the rightmost 8 bits of the flag register.
- Transfers bits 7 ,6 ,4 ,2 and 0 of AH register to SF, ZF, AF, PF and CF of FLAG register respectively.
- OF, DF, IF and TF are not affected.

Flag Register

SAHF

10011110

Machine code format

FLAGS

MOV AL,7Fh (127_{10} 01111111_2)

ADD AL,00h **07h** (127_{10} 01111111_2) CF=0 SF=0 OF=0

MOV AL,7Fh
ADD AL,01h **80h** (-128_{10} 10000000_2) CF=0 SF=1 OF=1

MOV AL,0FFh (-1_{10} 11111111_2)
ADD AL,02h **01h** (1_{10} 00000001_2) CF=1 SF=0 OF=0

MOV AL,0FEh (-2_{10} 11111110_2)
ADD AL,01h **FFh** (-1_{10} 11111111_2) CF=0 SF=1 OF=0

MOV AL,0FEh (-2_{10} 11111110_2)
ADD AL,0FFH **FDh** CF=1 SF=1 OF=0

Additional Data Transfer Instructions (X386 onwards)

- MOVSX DST, SRC

  Ex: MOVSX CX, BL

- MOVZX DST, SRC

  Ex: MOVZX CX, BL

- BSWAP REG 32

  Ex: BSWAP EAX

Additional Data Transfer Instructions (X386 onwards)

- **MOVSX DST, SRC**

  Ex: MOVSX CX, BL

- SX– Sign extension
- Destination size > Source size

Example: MOVSX CX, BL

Assume BL= 80H

After execution of MOVSX instruction

BL=80H

CX= CH CL

CL=80H = 1000 0000

CH= 1111 1111= FFH

Thus **CX= FF80H**

Additional Data Transfer Instructions (X386 onwards)

- **MOVZX DST, SRC**

  Ex: MOVZX CX, BL

- ZX– Zero extension
- Destination size > Source size

Example: MOVZX CX, BL

Assume BL= 80H

After execution of MOVZX instruction

BL=80H

CX= CH CL

CL=80H = 1000 0000

CH= 0000 00000=00H

Thus **CX= 0080H**

Additional Data Transfer Instructions (X386 onwards)

- **BSWAP REG 32**

  Ex: BSWAP ECX
- CONVERT LITTLE ENDIAN FORMAT TO BIG ENDIAN FORMAT
- Only 32 bit registers

Example: BSWAP ECX

Assume ECX= **24 56 89 A0H**

After execution of BSWAP ECX instruction

ECX= A0 89 56 24H

STRING DATA TRANSFERS

- 80x86 is equipped with special instructions to handle string operations
- String: A series of data words (or bytes) that reside in consecutive memory locations

- Each allows data transfers as a single byte, word, or double word.

STRING DATA TRANSFERS

- Five string data transfer instructions:

  MOVS, LODS, STOS, INS, and OUTS.

- Before the string instructions are presented, the operation of the D flag-bit (direction), DI, and SI must be understood as they apply to the string instructions.

The Direction Flag

- The direction flag (D, located in the flag register) selects the auto-increment or the auto-decrement operation for the DI and SI registers during string operations.
  - used only with the string instructions

- The **CLD** instruction clears the D flag (D flag =0 or reset) and the **STD** instruction sets it (D flag =1 or set) .

  - **CLD** instruction selects the auto-increment mode and **STD** selects the auto-decrement mode

DI and SI

- During execution of string instruction, memory accesses occur through DI and SI registers.
  - DI offset address accesses data in the extra segment for all string instructions that use it
  - SI offset address accesses data by default in the data segment
  - Operating in 32-bit mode EDI and ESI registers are used in place of DI and SI.

MOVS/MOVSB/MOVSW/MOVSD

- Copies a byte or word or double-word from a location in the data segment to a location in the extra segment
- Source –DS:SI
- Destination –ES:DI
- No Flags Affected
- For multiple-byte or multiple-word moves, the count to be in CX register
- Byte transfer, SI or DI increment or decrement by 1
- Word transfer, SI or DI increment or decrement by 2
- Double word transfer SI or DI increment or decrement by 4

MOVS/MOVSB/MOVSW/MOVSD

- 2nd method: Declaring the source and destination strings as DB → for byte transfer
- Declaring both as DW → for word transfer

MOVS with a REP

- The repeat prefix (REP) is added to any string data transfer instruction except LODS.
  - REP prefix causes CX to decrement by 1 each time the string instruction executes; after CX decrements, the string instruction repeats

- If CX reaches a value of 0, the instruction terminates and the program continues.

EX: If CX is loaded with 100 and a REP MOVSB instruction executes, the microprocessor automatically repeats the MOVSB 100 times.

COPY A BLOCK OF DATA FROM ONE MEMORY AREA TO ANOTHER MEMORY AREA-50 DATA

```
.data
Array1  db0ah,bch,deh,0f5h,11h, 56h,78h,0ffh,0ffh ,23h4ah, …
Array2   db   50 dup(0)
.code
startup
MOV     CX, 32H
LEA      SI, array1
LEA     DI, array2
CLD
REP   MOVSB
.EXIT
END
```

LODS/LODSB/LODSW

- Loads AL, AX with data at segment offset address indexed by the SI register.
- 1 is added to or subtracted from SI for a byte-sized LODS (LODSB)
- 2 is added or subtracted for a word-sized LODS (LODSW).

- 4 is added or subtracted for a doubleword-sized LODS.

LODS/LODSB/LODSW /LODSD

Loads AL or AX or EAX with the data stored at the data segment

•Offset address indexed by SI register

•After loading contents of SI INC if D = 0 & DEC if D = 1

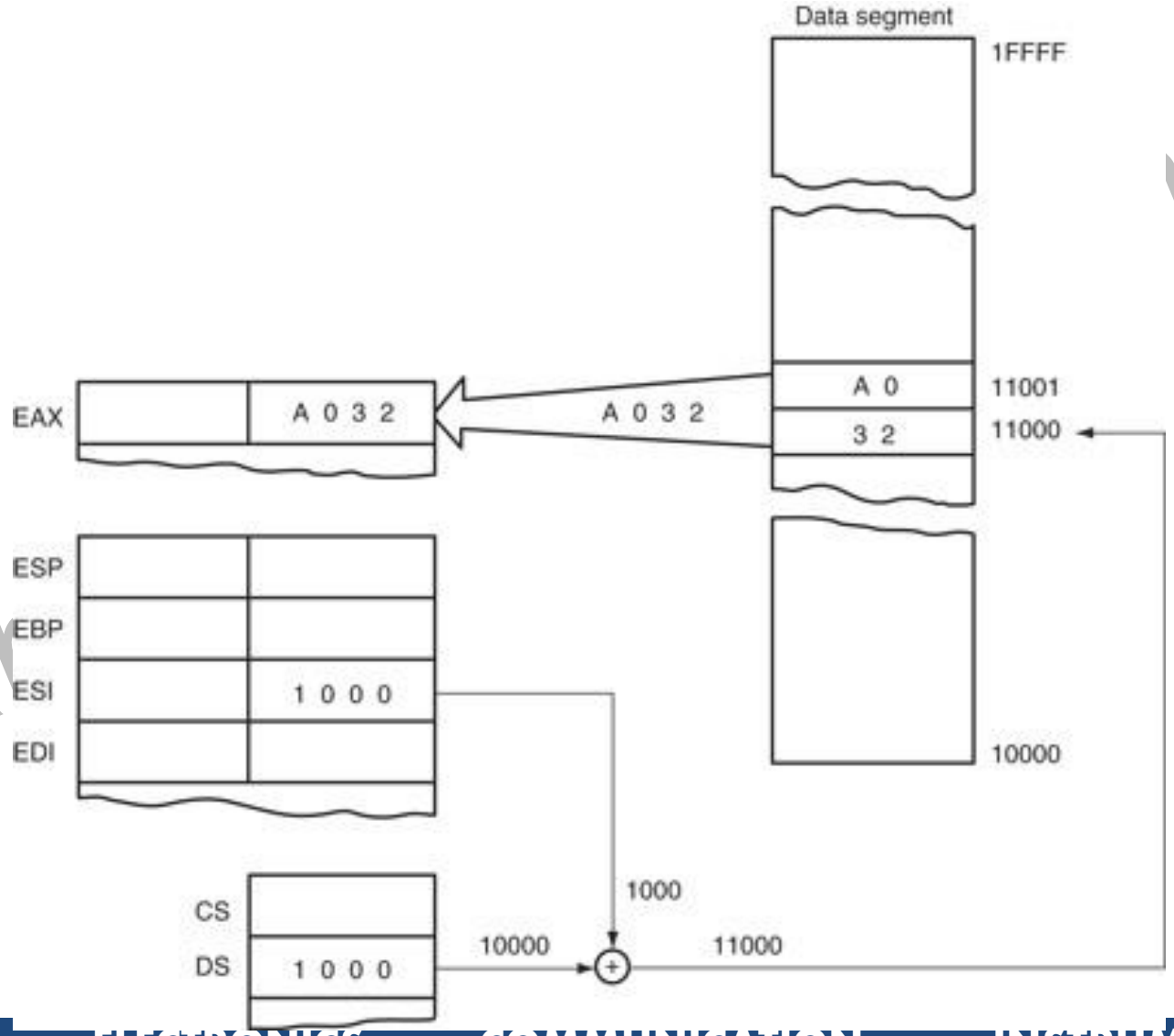
LODSB ; AL = DS:[SI]; SI = SI ±1

LODSW ; AX = DS:[SI]; SI = SI ±2

LODSD; EAX = DS:[SI]; SI = SI±4

•**LODS** affects no FLAGs

The operation of the LODSW instruction if DS=1000H, D=0,11000H,=32 11001H = A0. This instruction is shown after AX is loaded from memory, but before SI increments by 2.



Ex:

```
CLD                 ; Clears the direction flag
                    ; so SI is Automaticallyincremented.

MOV SI,OFFSET SOURCE-STRING
                          ; point SI at start of string

LODS SOURCE-STRING  ; Copy byte or word from string
                          to   AL or AX.
```

STOS /STOSB/STOSW

- Stores AL, AX into the Extra segment memory location addressed by the DI register.
- STOSB (**stores a byte**) stores the byte in AL at the extra segment memory location addressed by DI.
- STOSW (**stores a word**) stores AX in the memory location addressed by DI.
- After the byte (AL), word (AX), or doubleword (EAX) is stored, contents of DI increment or decrement.

STOS /STOSB/STOSW

Stores AL or AX or EAX into the Extra segment ES memory at Offset address indexed by DI register

•After storing contents in DI, INC if D = 0 & DEC if D = 1

STOSB ; ES:[DI]=AL; DI = DI ±1

STOSW ; ES:[DI]=AX; DI = DI ±2

STOSD; ES:[DI]=EAX; DI = DI ±4

STOS affects no FLAGs

Write an ALP to fill a set of 100 memory locations starting at displacement ‘DIS1’ with the value F6H

```
 .DATA
 DAT1          DB     100 DUP(?)
.CODE
.STARTUP
MOV  DI,  OFFSET DAT1
MOV AL, 0F6H
MOV CX, 64H
CLD
REP   STOSB
.EXIT
END
```

INS

- Transfers a byte or word of data from an I/O device into the extra segment memory location addressed by the DI register.
  - I/O address is contained in the DX register
- Useful for inputting a block of data from an external I/O device directly into the memory.

- Ex : One application transfers data from a disk drive to memory.
  - disk drives are often considered and interfaced as I/O devices in a computer system

THREE basic forms of the INS.

- INSB inputs data from an 8-bit I/O device and stores it in a memory location indexed by SI.
- INSW instruction inputs 16-bit I/O data and stores it in a word-sized memory location.
- INSD instruction inputs 32-bit I/O data and stores it in a word-sized memory location.
- These instructions can be repeated using the REP prefix
  - allows an entire block of input data to be stored in the memory from an I/O device

OUTS

- Transfers a byte or word data from the data segment memory location address indexed by SI to an I/O device.
  - I/O device addressed by the DX register as with the INS instruction

Thank you