Microprocessors and Interfaces
Lecture 8
8086 Instructions Set : Part-2
Data Transfer Instructions

By Dr. Sanjay Vidhyadharan

Data Transfer Instructions

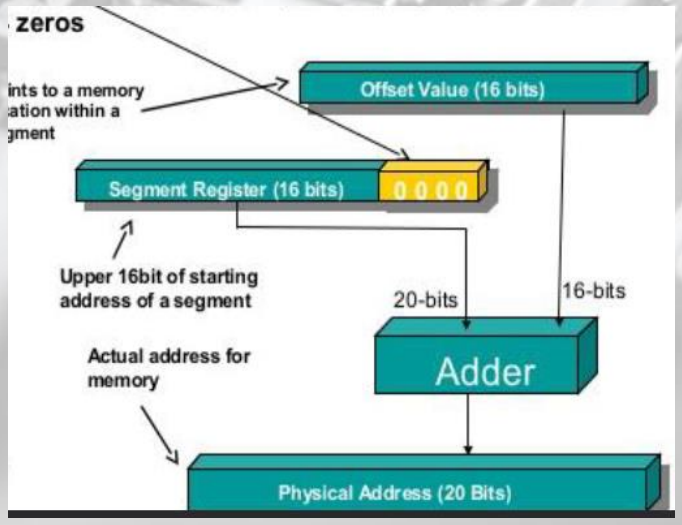
- General Purpose Data Transfer
  (MOV, XCHG, XLAT, PUSH, POP)
- Input / Output Data Transfer
  (IN, OUT)
- Address Object Data Transfer
  (LEA, LDS, LES)
- Flag Transfer Data Transfer
  (LAHF, SAHF, PUSHF, POPF)

Segment Override

- The **Segment Override Prefix** says that if we want to use some other segment register than the default segment for a particular code, then it is possible.
- E.g. MOV AX, SS : [BX]

- Here, in this case, the Stack segment register is used as a prefix for the offset BX. So, instead of DS, which is the default segment register for BX, the SS will be used for finding the effective address location.

- Effective address = SS X 10H + content of BX register



Segment Override

Write the machine language equivalent code : **MOV DS: 2345 [BP], DX**

Solution:

Here we have to specify DX using REG field. The D bit must be 0, indicating that DX is the source register. The REG field must be 010 to indicate DX register. The w bit must be 1 to indicate word operation. 2345 [BP] is specified with MOD=10 and R/M = 110 and displacement = 2345 H. Whenever BP is used to generate the Effective Address (EA), the default segment would be SS. In this example, we want the segment register to be DS, we have to provide the segment override prefix byte (SOP byte) to start with. The SOP byte is 001 SR 110, where SR value is provided as per table shown below.

SR	Segment register
00	ES
01	CS
10	SS
11	DS

To specify DS register, the SOP byte would be 001 11 110 = 3E H. Thus the 5 byte code for this instruction would be 3E 89 96 45 23 H.

SOP	Opcode	D	W	MOD	REG	R/M	LB disp.	HD disp.
3EH	1000 10	0	1	10	010	110	45	23

Suppose we want to code MOV SS: 2345 (BP), DX. This generates only a 4 byte code, without SOP byte as SS is already the default segment register in this case.

MOV = Move	76543210	76543210
Register/Memory to/from Register	100010 dw	mod reg r/m

– 0: Data flow from the REG field to the R/M

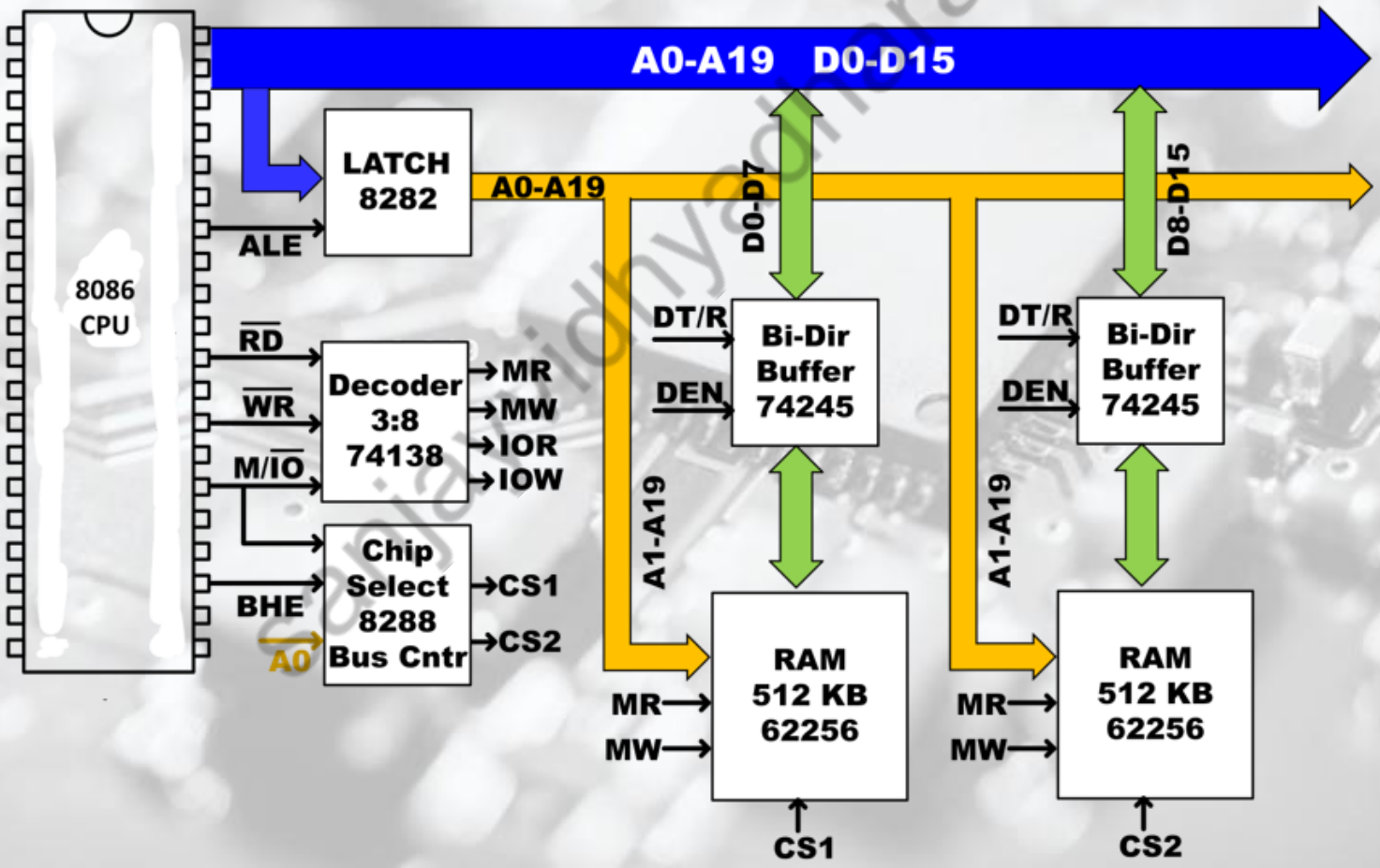
010	DL	DX

Mod 10 memory mode with 16 bit displacement

110	D16	(BP) + D8	(BP) + D16	DH	SI

Input / Output

- IN : Input byte or word
- OUT : Output byte or word



IN and OUT

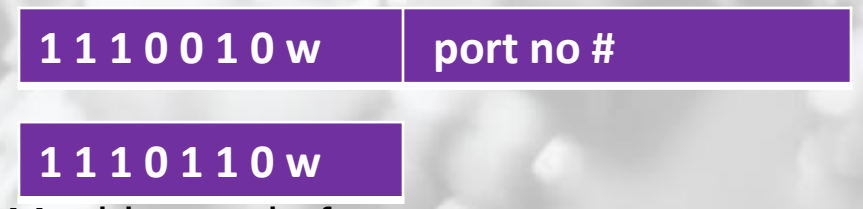
- IN & OUT instructions perform I/O operations.
- Contents of AL, AX, or EAX are transferred only between I/O device and microprocessor.
  - an IN instruction transfers data from an external I/O device into AL, AX, or EAX
  - an OUT transfers data from AL, AX, or EAX to an external I/O device
- Only the 80386 and above contain EAX

IN

- IN transfers a byte or word from an input port to the AL register or AX register.
- IN instruction has two formats:
  - Fixed port: port number is specified directly in the instruction (port no: 0-255).
  - Variable port: port number is loaded into the DX register before IN instruction (port no : 0 – 65535).

IN acc , port no#

IN acc , DX



Machine code formats

OUT

- OUT transfers a byte or a word from AL register or AX register respectively, to an output port.
- OUT instruction has two formats:
  - Fixed port: port number is specified directly in the instruction (port no: 0-255).
  - Variable port: port number is loaded into the DX register before OUT instruction (port no : 0 – 65535).

OUT port no# , acc

1 1 1 0 0 1 0 w	port no #

OUT DX, acc

1 1 1 0 1 1 1 w

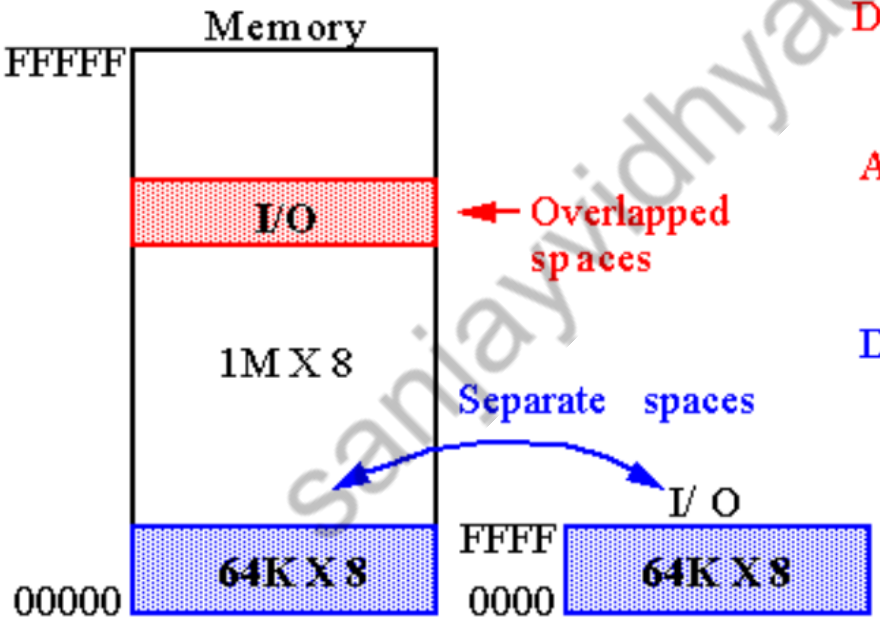
Machine code formats

IN/OUT

IN AL, 19 H
IN AX, 19 H
IN AL, DX
IN AX, DX
OUT 19H, AL
OUT 19H , AX
OUT DX, AL
OUT DX, AX

Isolated versus Memory-Mapped I/O

- In the Isolated scheme, IN, OUT, and IO/M are required.
- In the Memory-mapped scheme, any instruction that references memory can be used.



Disadvantage:
A portion of the memory space is used for I/O devices.

Advantage:
$\overline{IORC}$ and $\overline{IOWC}$ not required.
Any data transfer instruction.

Disadvantage:
Hardware using M/$\overline{IO}$ and W/$\overline{R}$ needed to develop signals $\overline{IORC}$ and $\overline{IOWC}$.
Requires IN, OUT, INS and OUTS

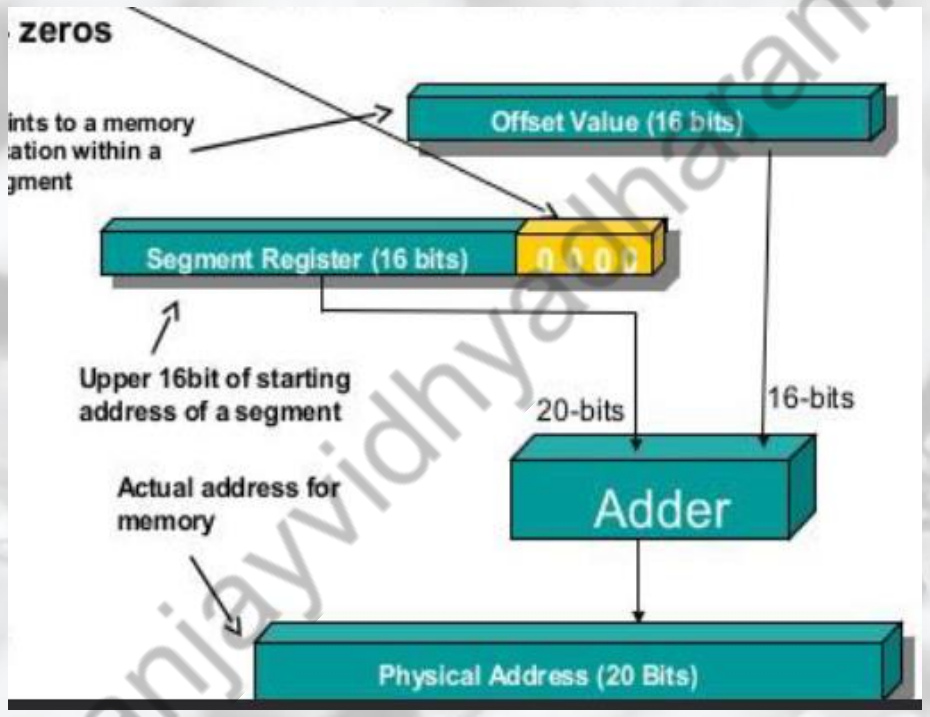
Address Object data transfer

- LEA : Load effective address
- LDS : Load pointer using DS
- LES : Load pointer using ES
- LFS : Load pointer using FS
- LGS : Load pointer using GS
- LSS : Load pointer using SS

Example

LEA BX , [1234H]

MOV BX , [1234H]



LEA AX, [BP+SI+5] ; Compute address of value

MOV AX, [BP+SI+5] ; Load value at that address

LEA(Load Effective Address)

- LEA transfers the offset of the source operand to a destination operand.
- The source operand must be a memory operand.
- The destination operand must be a 16-bit general purpose register.
- Does not effects flags.

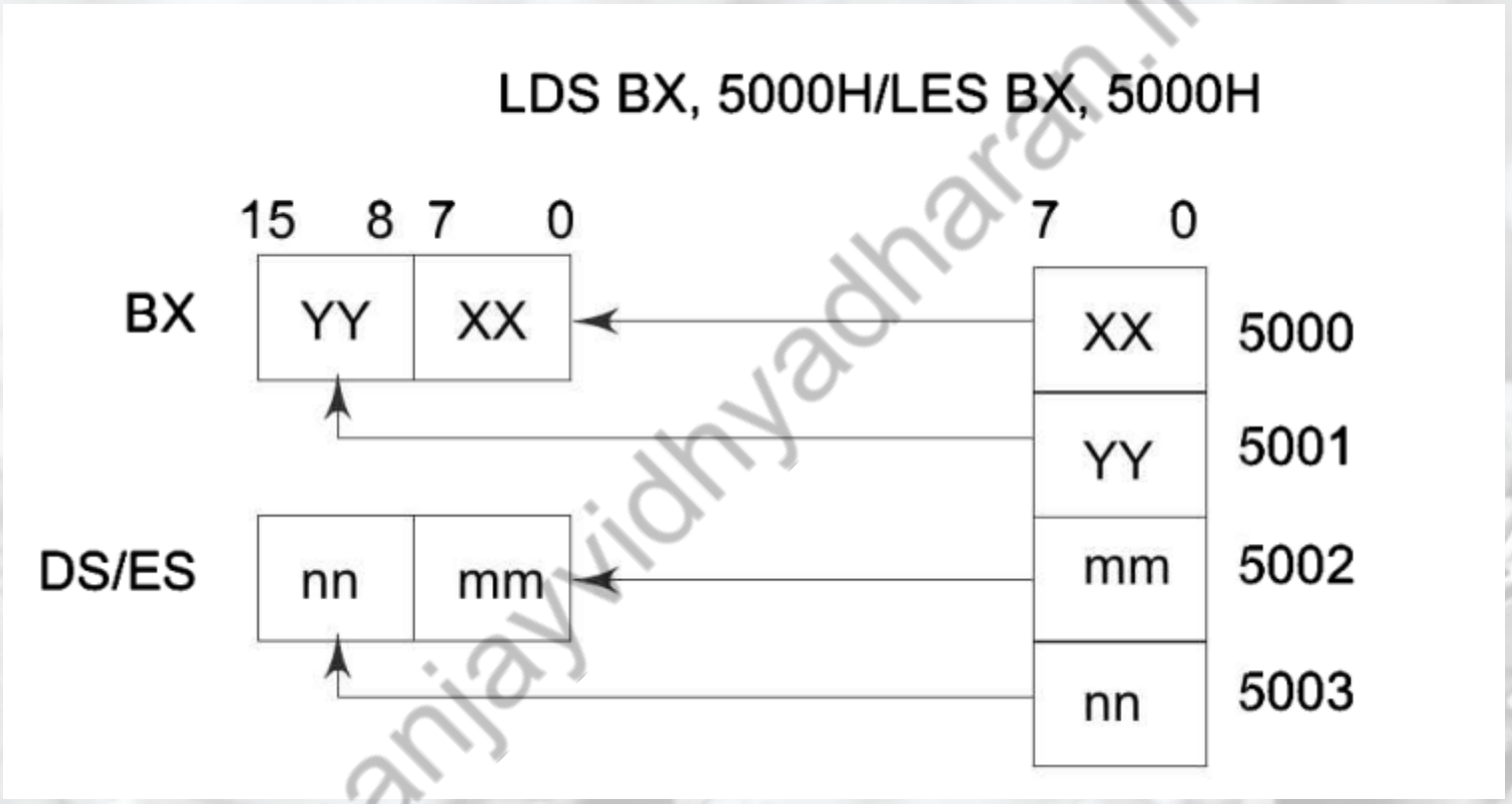
LEA reg , mem

Machine code format

LDS (load pointer using DS)

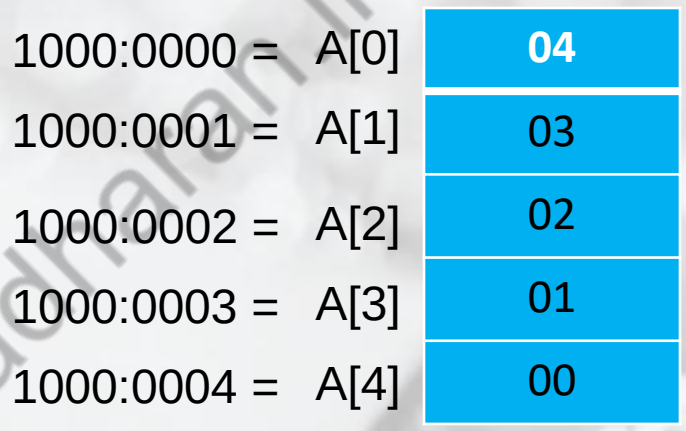
- LDS transfers 32-bit pointer variable from source operand to destination operand and DS register.
- The source operand must be a memory operand.
- The destination operand may be any 16-bit general purpose register.
- The first word of the pointer variable is transferred into 16-bit general purpose register.
- The second word of the pointer variable transferred into DS.

LDS (load pointer using DS)



Accessing array in data segment

```
A       DW      0000, 1000
```




```
MOV     SI , WORD PTR A
MOV     AX , WORD PTR A+2
MOV     DS , AX

LDS     SI , A
```

LES (load pointer using ES)

- LES transfers 32-bit pointer variable from source operand to destination operand and ES.
- The source operand must be a memory operand.
- The destination operand may be any 16-bit general purpose register.
- The first word of the pointer variable is transferred into 16-bit general purpose register.
- The second word of the pointer variable transferred into ES.

Accessing array in extra segment

```
B       DW      0000, 2000
```

2000:0000 = B[0]	04
2000:0001 = B[1]	03
2000:0002 = B[2]	02
2000:0003 = B[3]	01
2000:0004 = B[4]	00

```
MOV     DI , WORD PTR B
MOV     AX , WORD PTR B + 2
MOV     ES , AX

LES     DI , B
```

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)**

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

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

Thank you