Lecture No. 28

Data Transfer Instructions/8086

-Mrs. Jyotsna A. Kulkarni

LEA

LEA REG, memory

- Just stores the resulting address in the destination register, rather than actually fetching the data from the address.
- LEA AX, m ; AX = offset of m
- LEA BX,[DI]
  - BX= offset address (Not the data stored)specified by [DI] (contents of DI) reg.
  - How is this different from MOV BX,[DI]?
- LEA BX,[BX+DI]

Ex.1 : If [BX]= 1000 & [DI] = 2000 then offset =[BX]+[DI]=3000H hence, SI=3000H

- Ex.2

```
MOV BX, 35h
MOV DI, 12h
LEA SI, [BX+DI] ; SI = ?
LEA SI, [BX+DI] ; SI = 35h + 12h = 47h
```

- C Z S O P A Flags: unchanged

LES

LES REG, memory

- Loads any 16 bit reg. with offset address retrieved from memory location.
- And then loads ES with seg address retrieved from memory.
- REG = first word
- ES = second word

- Ex.1: LES BX, MEM1 ; loads ES & BX with 32 bit contents of DS:MEM1

- Ex2. LES AX, m

m -> 1234h– First word

5678h --- second Word

AX is set to 1234h, ES is set to 5678h.

- C Z S O P A Flags: unchanged

LDS

LDS REG, memory

- Loads any 16 bit reg. with offset address retrieved from memory location.
- And then loads DS with seg address retrieved from memory.
- REG = first word
- DS = second word

- Ex.1: LDS BX, MEM1 ; loads DS & BX with 32 bit contents of DS:MEM1

- Ex2. LdS AX, m

m -> 1234h– First word

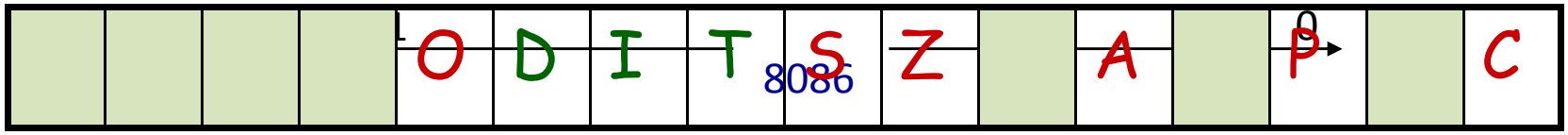
5678h --- second Word

AX is set to 1234h, DS is set to 5678h.

- C Z S O P A Flags: unchanged

STRING Instructions

(D- Flag, SI & DI)



- 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
- D-Flag : selects the auto-increment/decrement operation for the DI and SI reg.

  (used only with the string instructions)
- The CLD instruction clears the D flag and the auto-increment mode
- The STD instruction sets D flag and the auto-decrement mode

LODS

LODS No operands

- Loads AL, AX, with data at segment offset address indexed by the SI register.
- LODSB
  - AL=DS:[SI]; SI=SI+1
- LODSW
  - AX=DS:[SI]; SI=SI+2
  - HB-> [HMA]
  - LB->[LMA]
- If DS=1000H, SI=1000, D=0?
- AX is loaded from memory 11000 & 11001.
- Ex: LEA SI, a1
  LODSB ---?

a1 DB 'H', 'e', 'l', 'l', 'o‘

- C Z S O P A Flags: unchanged

STOS

STOS No operands

- Stores AL, AX, at 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.
  - ES:[DI] =AL; DI=DI+1 {Assumed D=0}

- STOSW (**stores a word**) stores AX in the memory location addressed by DI.
- ES:[DI]= AX; DI=DI+2 {Assumed D=0}
- If ES=1000H, DI=1000, D=0?
- AX is loaded from memory 11000 & 11001.

- C Z S O P A Flags: unchanged

STOS with a REP

- The **repeat prefix** (REP) is added to any string data transfer instruction except LODS.
- 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.
- If CX is loaded with 100 and a REP STOSB instruction executes, the microprocessor automatically repeats the STOSB 100 times.
- EX: LET DF=0

```
LEA DI, a1
MOV AL, 12h
MOV CX, 5
REP STOSB
RET
```

- C Z S O P A Flags: unchanged

MOVS

MOVS No operand

- Transfers a byte or word from a data segment addressed by SI to extra segment location addressed by DI.
- ES:[DI] = DS:[SI]
- if DF = 0 then
  - SI = SI + 1 & DI = DI + 1
- else
  - SI = SI – 1 & DI = DI - 1
- Ex:

```
CLD
LEA SI, a1
 LEA DI, a2
MOV CX, 5
 REP MOVSB
RET
```

- C Z S O P A Flags: unchanged

INS

INS No operand

- Transfers a byte or word of data from an I/O device into the extra segment memory location addressed by the DI register.

  (Note: I/O address is contained in the DX register)
- ES:[DI] = [DX]
- if DF = 0 then DI = DI + 1 else DI = DI – 1 for INSB
- if DF = 0 then DI = DI + 2 else DI = DI – 2for INSW

- Ex:

```
MOV DI,OFFSET LOC1
MOV DX,3ACH
CLD
MOV CX,50
REP INSB
```

- C Z S O P A Flags: unchanged
- Useful for inputting a block of data from an external I/O device directly into the memory. Using REP prefix with INS

OUTS

- OUTS: Transfers a byte or word of data from the data segment memory location address by SI to an I/O device.
- I/O device addressed by the DX register as with the INS instruction
- [DX] = DS:[SI]
- if DF = 0 then SI = SI + 1 else SI = SI – 1 for OUTSB
- if DF = 0 then SI = SI + 2 else SI = SI – 2for OUTSW
- More effectively used with REP prefix.

XCHG,LAHF & SAHF

XCHG REG, memory OR memory, REG OR REG, REG

- Exchanges contents of a register with any other register or memory location.
  - cannot exchange segment registers or memory-to-memory data
- Exchanges are byte- or word- and use any addressing mode except immediate addressing.
- LAHF instruction transfers the rightmost 8 bits of the flag register into the AH register.
- SAHF instruction transfers the AH register into the rightmost 8 bits of the flag register.

XLAT

XLATB NO Operands

- AL = DS:[BX + unsigned AL]
- An XLATB instruction first adds the contents of AL to BX to form a memory address within the data segment.
  - copies the contents of this address into AL
  - only instruction adding an 8-bit to a 16-bit number

```
Let                 dat DB 11h, 22h, 33h, 44h, 55h
 LEA BX, dat
MOV AL, 2
 XLATB ;  AL = 33h
```

- Converts the contents of the AL register into a number stored in a memory table.
- performs the direct table lookup technique often used to convert one code to another

IN and OUT

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

CMOV

- Many variations of the CMOV instruction.
  - these move the data only if the condition is true
- CMOVZ instruction moves data only if the result from some prior instruction was a zero.
  - destination is limited to only a 16- or 32-bit register, but the source can be a 16- or 32-bit register or memory location
- Because this is a new instruction, you cannot use it with the assembler unless the .686 switch is added to the program

SEGMENT OVERRIDE PREFIX

- May be added to almost any instruction in any memory-addressing mode
  - allows the programmer to deviate from the default segment
  - only instructions that cannot be prefixed are jump and call instructions using the code segment register for address generation
  - Additional byte appended to the front of an instruction to select alternate segment register
  - MOV Ax,[DI]
  - MOV AX,ES:[DI]

PUSH & POP Instructions revisited

Ex: Exchange contents of AX & BX

```
ORG 100h
MOV AX, 1212h ; store 1212h in AX.
MOV BX, 3434h ; store 3434h in BX
PUSH AX ; store value of AX in stack.
PUSH BX ; store value of BX in stack.
POP AX ; set AX to original value of BX.
POP BX ; set BX to original value of AX.
RET
END
```

```
MOV AX, 1234h
PUSH AX ; store value of AX in stack.
MOV AX, 5678h ; modify the AX value.
POP AX ; restore the original value of AX.
```

- It is very important to do equal number of **PUSH**s and **POP**s, otherwise the stack maybe corrupted and it will be impossible to return to operating system.

EX 1. Swap the word at memory location 24000_H with 25000_H

MOV AX, 2000_H

MOV DS, AX

MOV SI, 4000_H

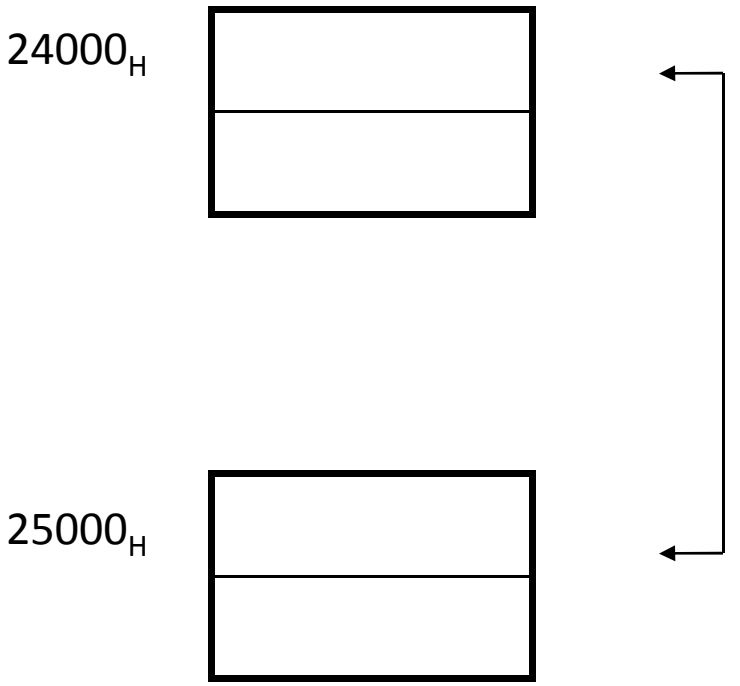
MOV DI, 5000_H

MOV BX, [SI]

MOV DX, [DI]

MOV [SI], DX

MOV [DI], BX



EX 2

MOV BX, 2000_H

MOV DI, 10_H

MOV AL, [BX+DI]

MOV DI, 20_H

MOV [BX+DI], AL

DS: 2020 ← DS: 2010