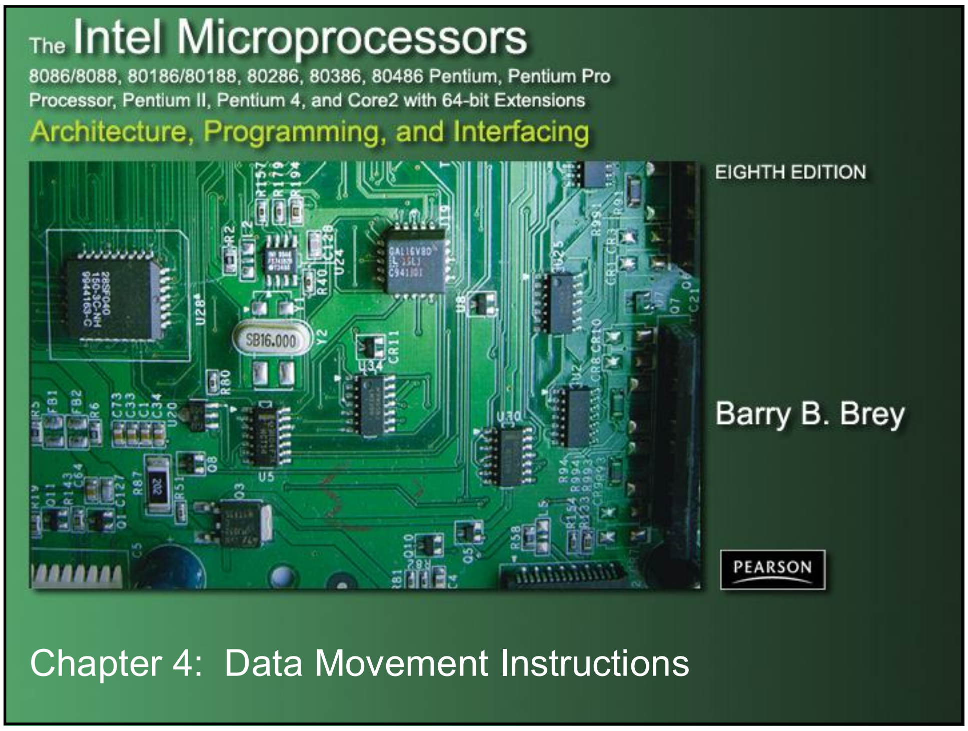


The Intel Microprocessors

8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium 4, and Core2 with 64-bit Extensions

Architecture, Programming, and Interfacing

EIGHTH EDITION

Barry B. Brey

PEARSON

Chapter 4: Data Movement Instructions

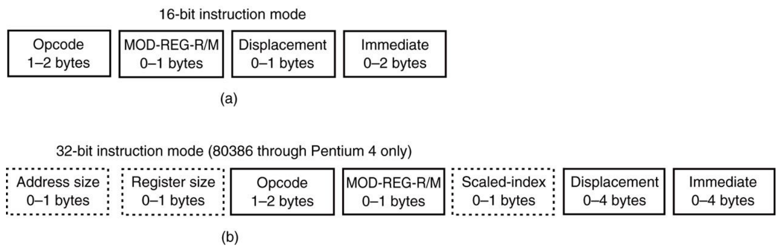
4–1 MOV Revisited

- In this chapter, the MOV instruction introduces machine language instructions available with various addressing modes and instructions.
- It may be necessary to interpret machine language programs generated by an assembler.
- Occasionally, machine language patches are made by using the DEBUG program available with DOS and Visual for Windows.

Machine Language

- Native binary code microprocessor uses as its instructions to control its operation.
  - instructions vary in length from 1 to 13 bytes
- Over 100,000 variations of machine language instructions.
  - there is no complete list of these variations

Figure 4–1 The formats of the 8086–Core2 instructions. (a) The 16-bit form and (b) the 32-bit form.



- 80386 and above assume all instructions are 16-bit mode instructions when the machine is operated in the *real mode* (*DOS*).
- in *protected mode* (*Windows*), the upper byte of the descriptor contains the D-bit that selects either the 16- or 32-bit instruction mode

The Opcode

- Selects the operation (addition, subtraction, etc.,) performed by the microprocessor.
  - either 1 or 2 bytes long for most instructions
- Figure 4–2 illustrates the general form of the first opcode byte of many instructions.
  - first 6 bits of the first byte are the binary opcode
  - remaining 2 bits indicate the **direction** (D) of the data flow, and indicate whether the data are a byte or a word (W)

Figure 4–2 Byte 1 of many machine language instructions, showing the position of the D- and W-bits.

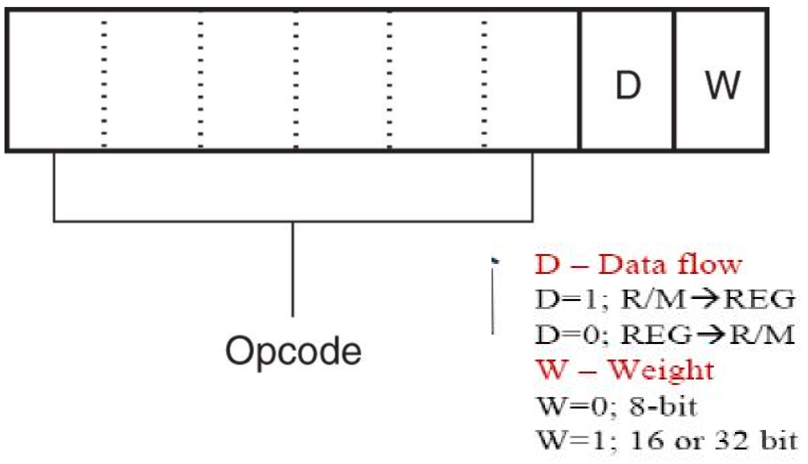
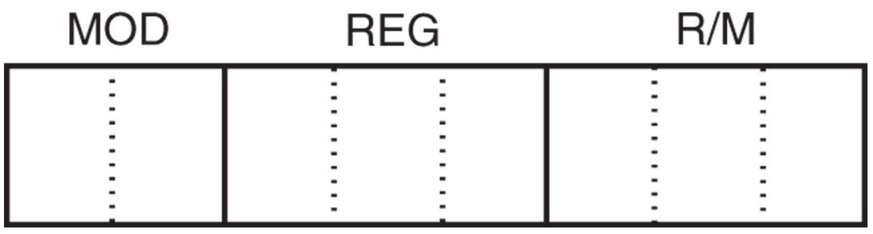


Figure 4–3 Byte 2 of many machine language instructions, showing the position of the MOD, REG, and R/M fields.



MOD Field

- Specifies addressing mode (MOD) and whether a displacement is present with the selected type.

TABLE 4–1 MOD field for the 16-bit instruction mode.

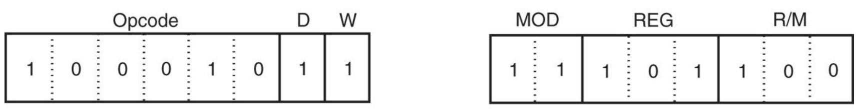
MOD	*Function*
00	No displacement
01	8-bit sign-extended displacement
10	16-bit signed displacement
11	R/M is a register

Register Assignments

TABLE 4–3 REG and R/M (when) MOD = 11 assignments.

Code	W = 0 *(Byte)*	W = 1 *(Word)*	W = 1 *(Doubleword)*
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBX
100	AH	SP	ESP
101	CH	BP	EBP
110	DH	SI	ESI
111	BH	DI	EDI

Figure 4–4 The 8BEC instruction placed into bytes 1 and 2 formats from Figures 4–2 and 4–3. This instruction is a MOV BP,SP.



Opcode = MOV
D = Transfer to register (REG)
W = Word
MOD = R/M is a register
REG = BP
R/M = SP

- the opcode is 100010, a MOV instruction
- D and W bits are a logic 1, so a word moves into the destination register specified in the REG field
- REG field contains 101, indicating register BP, so the MOV instruction moves data into register BP

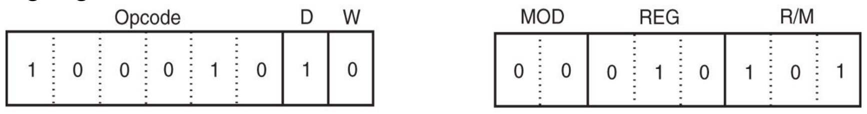
R/M Memory Addressing

- Table 4–4 lists the memory-addressing modes for the R/M field when MOD is a 00, 01, or 10 for the 16-bit instruction mode.

TABLE 4–4 16-bit R/M memory-addressing modes.

R/M Code	*Addressing Mode*
000	DS:[BX+SI]
001	DS:[BX+DI]
010	SS:[BP+SI]
011	SS:[BP+DI]
100	DS:[SI]
101	DS:[DI]
110	SS:[BP]
111	DS:[BX]

Figure 4–5 A MOV DL,[DI] instruction converted to its machine language form.



Opcode = MOV
D = Transfer to register (REG)
W = Byte
MOD = No displacement
REG = DL
R/M = DS:[DI]

- This instruction is 2 bytes long and has an opcode 100010, D=1 (to REG from R/M), W=0 (byte), MOD=00 (no displacement), REG=010 (DL), and R/M=101 ([DI]).
- If the instruction changes to MOV DL, [DI+1], the MOD field changes to 01 for 8-bit displacement

4–3 LOAD EFFECTIVE ADDRESS

- LEA instruction loads any 16-bit register with the offset address
  - determined by the addressing mode selected
- LDS and LES load a 16-bit register with offset address retrieved from a memory location
  - then load either DS or ES with a segment address retrieved from memory

Load-effective address instructions

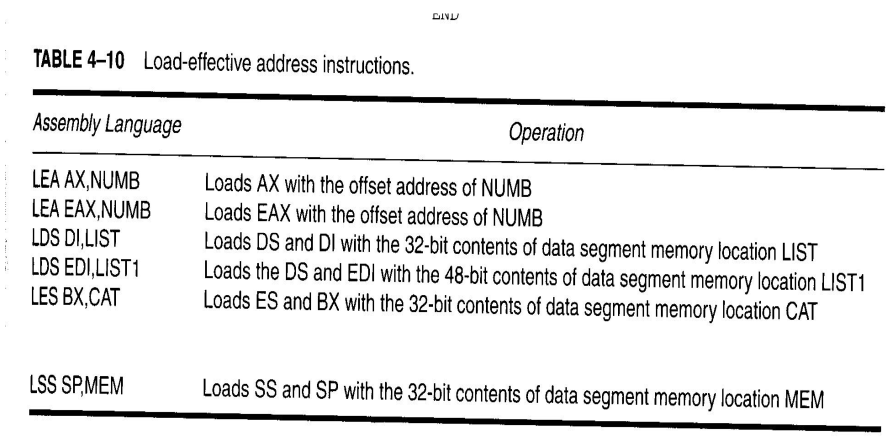


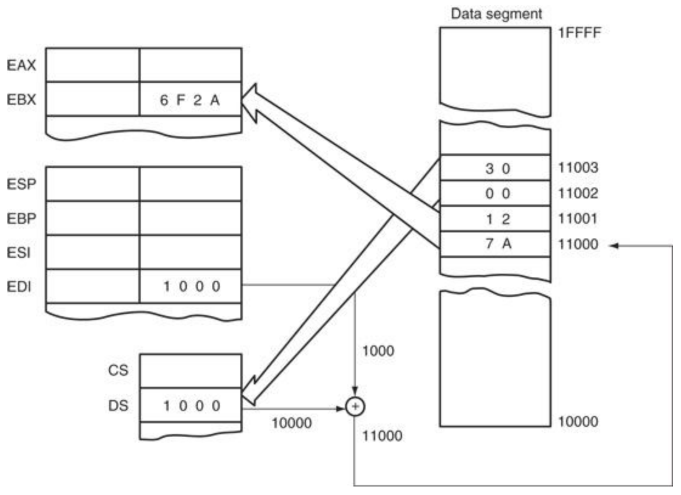
TABLE 4–10 Load-effective address instructions.

Assembly Language	*Operation*
LEA AX,NUMB	Loads AX with the offset address of NUMB
LEA EAX,NUMB	Loads EAX with the offset address of NUMB
LDS DI,LIST	Loads DS and DI with the 32-bit contents of data segment memory location LIST
LDS EDI,LIST1	Loads the DS and EDI with the 48-bit contents of data segment memory location LIST1
LES BX,CAT	Loads ES and BX with the 32-bit contents of data segment memory location CAT
LSS SP,MEM	Loads SS and SP with the 32-bit contents of data segment memory location MEM

LDS, LES, and LSS

- Load any 16- or 32-bit register with an offset address, and the DS, ES, or SS segment register with a segment address.
- Figure 4–17 illustrates an example LDS BX,[DI] instruction.

Figure 4–17 The LDS BX,[DI] instruction loads register BX from addresses 11000H and 11001H and register DS from locations 11002H and 11003H. This instruction is shown at the point just before DS changes to 3000H and BX changes to 127AH.



4–4 STRING DATA TRANSFERS

- Five string data transfer instructions: LODS, STOS, MOVS, INS, and OUTS.
- Each allows data transfers as a single byte, word, or double word.
- 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

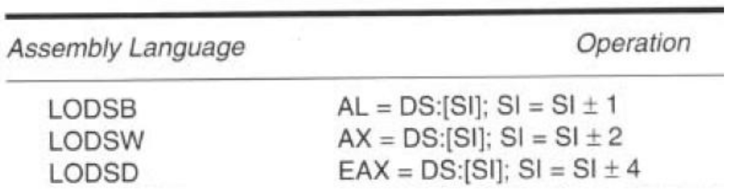
- The direction flag (D, located in the flag register) selects the auto-increment or the auto-decrement operation for the DI and SI registers during string operations.
  - used only with the string instructions
- The CLD instruction clears the D flag and the STD instruction sets it .
  - CLD instruction 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.
  - this allows string using any memory location in the entire 4G-byte protected mode address space

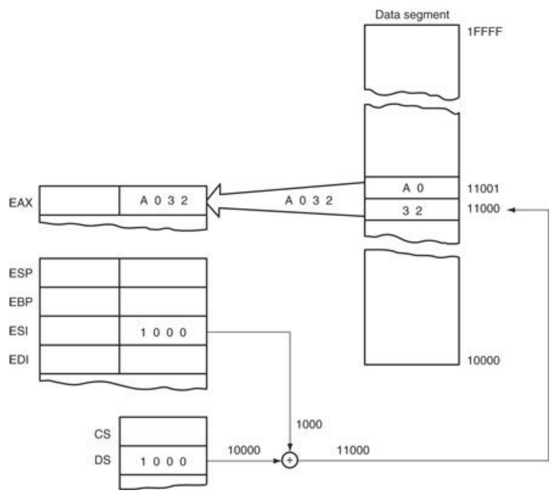
LODS

- Loads AL, AX, or EAX with data at segment offset address indexed by the SI register.



Assembly Language	*Operation*
LODSB	AL = DS:[SI]; SI = SI ± 1
LODSW	AX = DS:[SI]; SI = SI ± 2
LODSD	EAX = DS:[SI]; SI = SI ± 4

Figure 4–18 The operation of the LODSW instruction if DS=1000H, D=0,11000H, 11001H = A0. This instruction is shown after AX is loaded from memory, but before SI increments by 2.



STOS

- Stores AL, AX, or EAX 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.(DI=DI±1)
- STOSW (**stores a word**) stores AX in the memory location addressed by DI.(DI=DI±2)
- After the byte (AL), word (AX), or doubleword (EAX) is stored, contents of DI increment or decrement.

MOVS

- Transfers a byte, word, or doubleword from data segment addressed by SI to extra segment location addressed by DI.
  - pointers are incremented or decremented, as dictated by the direction flag
- Only the source operand (SI), located in the data segment may be overridden so another segment may be used.
- The destination operand (DI) must always be located in the extra segment.

- The only memory-to-memory transfer allowed.
- MOVSB transfers byte from data segment to extra segment.
- MOVSW transfers word from data segment to extra segment.

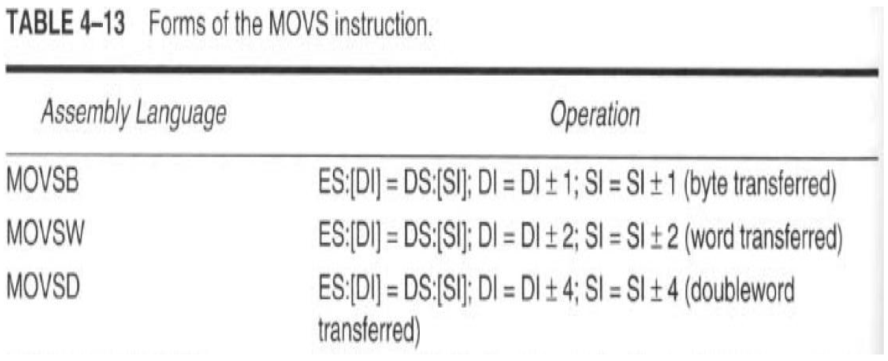


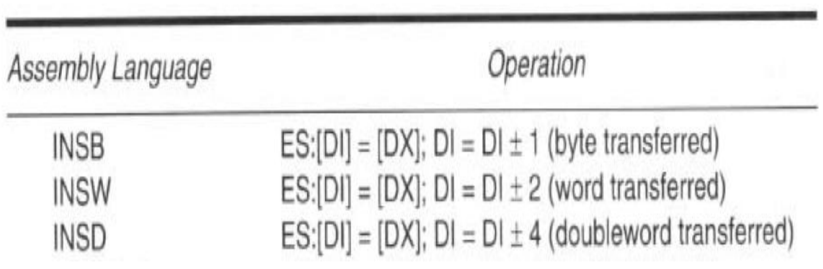
TABLE 4–13 Forms of the MOVS instruction.

Assembly Language	Operation
MOVSB	ES:[DI] = DS:[SI]; DI = DI ± 1; SI = SI ± 1 (byte transferred)
MOVSW	ES:[DI] = DS:[SI]; DI = DI ± 2; SI = SI ± 2 (word transferred)
MOVSD	ES:[DI] = DS:[SI]; DI = DI ± 4; SI = SI ± 4 (doubleword transferred)

INS

- Transfers a byte, word, or doubleword 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.
- One application transfers data from a disk drive to memory.

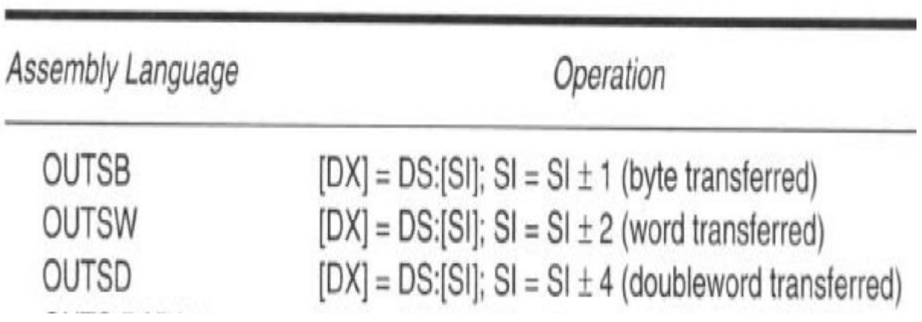
- Three basic forms of the INS.
- INSB inputs data from an 8-bit I/O device and stores it in a memory location indexed by SI.
- INSW instruction inputs 16-bit I/O data and stores it in a word-sized memory location.
- INSD instruction inputs a doubleword.



Assembly Language	Operation
INSB	ES:[DI] = [DX]; DI = DI ± 1 (byte transferred)
INSW	ES:[DI] = [DX]; DI = DI ± 2 (word transferred)
INSD	ES:[DI] = [DX]; DI = DI ± 4 (doubleword transferred)

OUTS

- Transfers a byte, word, or doubleword 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
- OUTSB
- OUTSW
- OUTSD
- INS and OUTS instructions not available on 8086/8088microprocessors.



Assembly Language	Operation
OUTSB	[DX] = DS:[SI]; SI = SI ± 1 (byte transferred)
OUTSW	[DX] = DS:[SI]; SI = SI ± 2 (word transferred)
OUTSD	[DX] = DS:[SI]; SI = SI ± 4 (doubleword transferred)