1/2/17

S. JAGADEESAN

UNIT - II

Data Movement Instructions.

→ These types of instructions are used to transfer data from the source operand to the destination operand.

→ This instructions are used to transfer data blw registers, registers and memory, register and immediate data or memory and immediate data.

Mov Destination, Source.

→ Copy data from source to destination.

Destination	Source.
Register	Immediate data
memory	Immediate data,
Register	Register
Register	memory
memory	Register
Memory	Segment
Segment	Memory
Segment	Register
Register	Segment

Mov Register, Immediate Data.

→ This instruction moves immediate 8-bit / 16-bit data to the specified register.

→ Object code is either 2 or 3 bytes.

W = 0 for 8-bit data.

1011	W	rrr	8-bit Data

W = 1 for 16-bit data

1011	W	rrr	Lower 8-bit Data	Higher 8-bit Data

for Ex:-

MOV AL, FFH

MOV AX, FFFFH

MOV Mem/Reg, Data

→ This instruction is executed immediate 8-bit or 16-bit data moves to a specified register or a memory location

→ not used to transfer immediate data to register.

W=0 for 8-bit data

1100 011W	Mod 000 R/M	8-bit Data

W=1 for 16-bit data

1100 011W	Mod 000 R/M	Lower 8-bit Data	Higher 8-bit Data

for Ex:- MOV [0345], 23H ; opcode = C6, 06, 23,

MOV [03H5], 2345H ; opcode = C7, 06, H5, 23

another Ex:- MOV [BX], H5H ; opcode = C6, 07, H5

MOV Acc, Memory

→ The 8-bit or 16-bit data will be moved to accumulator register AL or AX

1010 000W	16-bit offset address

W=0, Object code = A0, offset address

W=1, Object code = A1, offset address

for Ex:- MOV AL, [23H0] ~~object~~ opcode = A0, H0, 23

MOV memory, Acc

→ The content of the accumulator will be stored into the memory. The object code is

1010 001W	16-bit offset address

For W=1, Object code = A3, offset address

W=0, Object code = A2, offset address

For Ex:- MOV [4000], AL ; object code = A2,00,40

MOV [4000], AX ; object code = A3,00,40

AX will be stored in two consecutive memory location

MOV Memory|Register, Memory|Register

→ This instruction is used to move 8-bit or 16-bit data from one register to another register, memory to register and register to memory.

→ This instruction cannot used for data transfer from memory to memory.

1010 10dW	Mod reg R/M

direction flag d = 0 or 1

d=0 → specified register is source operand.

Mod, R/M used for first operand (M/R-1)

reg represents the second operand (M/R-2)

d=1 → register works as the destination operand

Mod, R/M used for second operand (M/R-2)

reg defines first operand (M/R-1)

for Ex:- MOV BX, CX

data transfer from CX to BX register.

d = 0 register specifies register which is source operand

mod & R/M are used for the first operand. ie BX

reg defines the second operand ie CX

1010 10dW	mod reg R/M

= 1000 1001

11	001	011	= 89, CB
mod	reg	R/M	
for	for	for	
BX	CX	BX	

when d = 1., reg specifies a register used as destination operand.

mod & R/M used for second operand ie CX

reg stands for first operand ie BX as destination

& W = 1. The object code is

1000 1011, 1101 1001 = 89, D9

11	011	001
mod	reg	R/M
for	for	for
CX	BX	CX

89, CB & 89, D9 both code valid for MOV BX, CX

PUSH|POP

→ These instructions are used to manipulate stack-related operations.

S. JAGADEESAN

PUSH Source

→ Push source register onto stack.

SP = SP-2 ; SS:[SP] ← Source register.

Flag affected : None

After Execution, the content of a specified register is pushed onto the stack.

Stack pointer (SP) is decremented by 2

Stores the two-byte contents of the operand

Initially the higher byte is pushed & then the lower byte is pushed.

The object code is

1111 1111	mod 110 R/M

For register | 01010 reg |

For segment register | 000 reg 110 |

Ex:- PUSH AX ; 50H PUSH CX ; 51H PUSH [H000]

PUSH BX ; 53H PUSH DS ; 1E Code: FF 36 00 40

PUSHF

→ Push flags word onto stack.

SP = SP-2 ; SS:[SP] ← flags, SP = SP-2 ; SS:[SP] ← Source

Flags affected : None

The object code of PUSHF is 100 111 00 = 9C

S. JAGADEESAN

POP Destination.

→ Pop word at top of stack to destination.

Destination ← SS:[SP] ; SP = SP+2 flag affected : None

After Execution, it loads the specified register/memory location with the contents of the memory location which the address is formed using the current stack Segment (SS) & the stack pointer (SP).

→ The stack pointer is incremented by 2

The object code of POP is

For register/memory | 1000 1111 | mod 000 R/M |

For register | 0101 1 reg |

For segment register | 000 reg 111 |

Ex:- POP AX ; 58H POP CX ; 59H POP [4000]

POP BX ; 5BH POP DS ; 1F Code: 8F 06 00 40

POPF

→ Pop word at top of stack to flag register.

Flags ← SS:[SP] ; SP = SP+2 , Flag affected : All

The object code of POPF is

100•111 01 = 9D

LEA reg16, addr (Load Effective Address)

→ Loads the effective address or offset of memory variable into reg16.

→ This type of data-transfer operation is important to load a segment or general-purpose register with an address directly from memory.

→ This instruction load a specified register with 16-bit offset address.

The object code is

1000 1101	Mod reg R/M

Ex:- LEA SI ; 16-bit EA loads in the SI register.

LEA BX, ADR ; EA of ADR will be transferred to Bx reg.

The object code of LEA BX, [0245] =

1000 1101 00 000 001 45 02 = 8D, 1E, 45, 02

LDS reg16, memory (Load Data Segment)

reg16 ← [memory16] ; DS ← [memory 16+2]

Flag affected : None.

→ Loads the DS register & reg16 from memory with the segment & offset values.

→ It loads the contents of the memory locations following the specified memory locations into DS reg.

The object code is

1100 0101	Mod reg R/M

= C5, mod reg r/m.

Ex:- LDS AX, [BX]

Loads the content of memory location specified by the content of Bx into AX register.

Mod for [BX] = 00

R/M for [BX] = 111

reg for AX = 000

The object code is C5, 0000 0111 = C5,07

LES reg16, memory (Load Extra segment)

reg16 ← [mem16] ; ES ← [mem16+2] Flag Affected : None

→ Load the ES register & reg16 with the segment & offset values for the variable in memory.

The object code is

1100 0100	Mod reg R/M

Ex:- LES Cx, [4000]

The object code = 1100 0100 00 001 110 00 40 = C4, 06, 00, 40.

Note :- LFS, LGS & LSS are only available in 80386

String Data Transfer. S.JAGADEESAN

LODS (Load String)

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

LODSB (Load String Byte)

AL ← DS:[SI] ; SI = SI ± 1 ; Flag affected : None

→ Moves a string one byte at a time from source memory address DS:SI to AL

→ SI is incremented or decremented by 1, depending on Direction Flag (DF)

→ The object code of LODSB is 1010 110W = AC as W=0

LODSW (Load String Word)

AX ← DS:[SI] ; SI = SI ± 2 ; Flag affected : None.

→ moves a string one word at a time from source memory address ~~DS:[SI]~~ DS:SI to AX

→ SI is incremented or decremented by 2, depending on Direction Flag (DF)

→ The object code of LODSW is 1010 110w = AD as w=1

3/2/17

STOS (Store String)

→ Stores AL, AX or EAX at the extra segment memory location addressed by the DI register

S.JAGADEESAN

STOSB (Store String Byte)

• ES:[DI] ← AL ; DI = DI ± 1 ; Flag affected : None

→ moves a string one byte at a time from AL to destination memory ES:DI

→ DI is incremented or decremented by 1, depending on Direction Flag (DF)

→ The object code of STOSW is 1010 101W = AA as W=0

STOSW (Store String word)

ES:[DI] ← AX ; DI = DI ± 2 ; Flag affected : None

→ moves a string one ~~word byte~~ at a time from AX to destination memory address ES:DI

→ DI is then incremented or decremented by 2, depending on Direction Flag (DF)

→ The object code of STOSW is 1010 101W = AB as W=1

MOVS (Move String)

→ This instruction transfers a byte, word or double word from the data segment location addressed by SI to the extra segment location addressed by SI.

MOVSB (Move String Byte)

ES:[DI] ← DS:[SI] ; DI = DI ± 1 ; SI = SI ± 1 ;

Flag affected : None

→ moves a string a byte at a time from source memory DS:SI to destination memory ES:DI

→ SI & DI are incremented or decremented by 1, depending on Direction Flag (DF).

→ The object code of MOVSB is 1010 010w = A4 as w=0

MOVSW (move string word)

ES:[DI] ← DS:[SI] ; DI = DI ± 2; SI = SI ± 2 ;

Flag affected : none

→ moves a string a word at a time from source memory DS:SI to destination memory ES:DI

→ SI & DI are incremented or decremented by 2, depending on direction flag (DF)

→ The object code of MOVSW is 1010 010w = A5 as w=1

NOTE: INS (IIP STR), OUTS (OIP STR) are not available on 8086/8088 µPs

MISCELLANEOUS DATA TRANSFER INSTRUCTIONS.

Miscellaneous → Don't be fooled by the term.

→ These instructions are used in programs.

XCHG Destination, Source

→ This instruction is used to exchange the contents of the specified source & destination operands, which may be registers or a memory location.

S. JAGADEESAN

→ The exchange of contents of two memory locations is not allowed.

→ Immediate data is not allowed.

→ The data format for register to register and register to memory is

1000 011W	Mod Reg R/M.

data format for register to Accumulator is

1001 0 reg

For Ex1:- XCHG [4000], AX ; Exchange data b/w AX & a memory location represented by offset address 4000H with the content of Data segment.

Ex2:- XCHG AX, BX ; Exchange data b/w AX and BX

→ The object code of XCHG AX, BX is 1000 011W mod reg R/M

= 1000 0111, 11 000 011 = 87, C3 as W=1, Mod = 11, reg = 000 & R/M = 011.

→ The object code of XCHG AL, [BX] = 1000 0110, 00 000 111

= 86, 07 where w=0, mod=00, reg = 000 & R/M = 111

LAHF & SAHF

→ These instructions are seldom used because they were designed as bridge instructions.

→ This instructions allowed 8085 s/w to be translated into 8086 s/w by a translation program

LAHF (Loads the Lower byte into AH)

AH ← Flags, Flags not affected: O, A, C, Z & P

→ Loads the low byte of the flags word into the AH register.

→ This instruction loads the AH register with the lower byte of the flag register.

→ This instruction followed by a PUSH AX instruction has the same effect of PUSH PSW instruction in 8085

AH = Flag register

AH bit: 7 6 5 4 3 2 1 0

SF ZF [0] AF [0] PF [1] [CF]

Here bit 1, 3 & 5 are reserved.

The object code of LAHF instruction is 9F

1001 1111

SAHF (Saves AH into Lower Flags byte)

Flags ← AH, Flag affected : None

→ Saves the AH register bit pattern into the low byte of the flag register.

→ The lower byte of the 8086 flag register is exactly same as flag register of 8085

→ This instruction replaces the equivalent flag byte of 8085 with a byte from the AH register.

→ POP PSW instruction of 8085 will be translated to POP AX SAHF on 8086 processor.

→ The object code of SAHF instruction is 9E

1001 1110

XLAT (Translate Byte in AL by look-up Table)

AL ← DS : [BX+AL], Flag affected : None

→ This instruction is used to translate the byte in the AL register by adding it to a base value in BX which has been set to locate the look-up table and the byte located is returned in AL

→ The physical address of memory location of look-up table is computed from DS : [BX+AL]

→ The object code of XLAT is 1101 0111 = D7

IN & OUT

→ These instructions perform I/O operations.

→ The content of AL, AX or EAX are transferred only b/w the I/o device & Microprocessor.

S. JAGADEESAN

IN port8 or DX (I/p data from I/o device)

byte : AL ← port

word : AL ← [port] ; AH ← [port+1] or AX ← (DX)

Flag affected : none.

→ used to read data from an i/p port.

→ The address of the i/p port can be specified within the instruction directly or indirectly.

→ AL & Ax registers can be used as destination for 8-bit & 16-bit i/p operands

→ Dx is allowed to carry the port address

→ When port address contain 8-bit · I/p a byte or word from direct I/o ports 00H to FFH (0-255)

→ When the port address consist of 16-bit · I/p a byte or word from Indirect I/o ports 0000H – FFFFH (0-65535)

port address in DX & flags are not affected.

→ The object code is

fixed port | 1110 010w | port | Variable port | 1110 110w |

Ex:-

IN AL,01 ; Load the content of 8-bit port address 01H to AL reg

The object code of IN AL,01 is 1110 010W = E4,01 as w=0

S. JAGADEESAN

IN AX, DX ; Read data from a 16-bit port address specified by DX register & stores it in Ax register.

The object code of IN AX,DX is 1110 110W = ED as w=1

6/2/17

OUT port8 or Dx (O/p data to I/o device)

byte : [port] ← AL word : [port] ← AL [port+1] ← AH or (DX ← AX) Flag affected: none

→ used to write an o/p port.

→ The address of the o/p port may be specified in the instruction directly or implicitly in Dx.

→ The content of AL or AX are transferred to directly or indirectly addressed port after execution.

→ instruction can o/p a byte or word to direct ports 00H to FFH (0 to 255)

→ if the port address is 16-bit in Dx. output a byte or word to indirect I/o ports 0000H to FFFFH. (0 to 65,535) port address in DX & flags are not affected.

Fixed port | 1110 011w | port | variable port | 1110 111w |

Ex1: OUT 02,AL ; Sends the content of AL to a port address 02H

The object code is 1110 0110 = E6 · as w=0

Ex2: OUT DX,AX ; Sends data available in Ax to port address specified by the Dx reg. Object code = 1110 111w = EF as w=1

Note:

1) MOVSx & movzx instructions are in 80386 - Pentium H
2) BSWAP (Byte swap) in 80486 - Pentium H.
3) CMOV (conditional move) in Pentium PRO-core2

Assembler Detail

→ The assembler (MASM) can be used in two ways
1) with models that are unique to a particular assembler.
2) with full-segment definitions that allow complete control over the assembly process & are universal to all assembler.

Directives

→ The directives (Pseudo-operations) that ctrl the assembly process
→ It indicates how an operand of a program is to be processed by the assembler.
→ Some directives generate & store information in the memory but not all.

Storing Data in memory segment.

S.JAGADEESAN