# Ch 3 : Addressing Modes

**Introduction**

- MOV instruction transfers bytes or words of data between two registers or between registers and memory in 8086 through 80286.
- Double words as well in 80386 and above
- CALL and JUMP modify flow of program

## 3.1 Data Addressing Modes

- Source to right, destination to left (Operands)
- Register to register
- For memory to memory transfer, MOVS is used
- Data is copied, not moved.
- Flag register remains unchanged
- **Register Addressing:**
    - Byte or word
    - E.g. MOV CX, DX
    - Segment to segment register MOV instruction is not allowed
    - Moving to CS i.e. changing CS is not allowed to avoid unpredictability of next instruction. Because IP and CS should have same address of next instruction. Changing only CS and not IP will cause error
- **Immediate Addressing:**
    - Byte, word, doubleword or quadword
    - Data to register
    - E.g. MOV AL, 22H (22H is stored in AL)
    - In 64-bit operation of the Pentium 4 or Core2, only a MOV immediate instruction allows access to any location in the memory using a 64-bit linear address.
    - Immediate implies that the data immediately follow the hexadecimal opcode in the memory.
    - Immediate data is constant data, whereas the data transferred from a register or memory location is variable data
    - In some assemblers, # is used before immediate data
    - H is for hexadecimal. Begin with 0 is letter is first digit
        - In some as, #'h1234
    - Nothing for decimal: MOV AL, 100
    - ASCII in ' ' : MOV BH, 'A'
    - Append B or Y for binary
- **Direct Data Addressing:**
    - Byte or word or double word
    - Memory location to register
    - e.g. MOV CX, LIST copies from location named LIST to CX
    - Segment registers can be stored or loaded from memory
    - Address is formed by adding the displacement to the default data segment address or alternate segment address
    - Direct Addressing:
        - Applies to MOV between a memory location and AL, AX or EAX.
        - MOV is 3 byte long but may exceed in 80386 and above
            - Because they appear often, to reduce the length of program, their size was made special (3 byte)
        - E.g. MOV AL, DATA loads AL from data segment memory location DATA
            - DATA is symbolic memory location while 1234H is actual hexadecimal location.
        - Sometimes written as MOV AL, [1234H], where [1234H] is an absolute memory location that is not allowed by all assembler programs.
        - This may need to be formed as MOV AL, DS:[1234H] with some assemblers, to show that the address is in the data segment.
        - Effective address formed by adding 1234H to DS value (0 appended)
        - e.g. in the above case if DS = 1000H, E.A. = 11234H
    - Displacement Addressing:
        - Applies to almost any instruction in the instruction set
        - 4 or more bytes
            - upto 7 bytes in 80386 through Pentium4 if 32 bit register and displacement is specified
        - Very flexible because most often used
        - E.g. MOV CL, DS:[1234H]
- **Register Indirect Addressing:**
    - byte or word or doubleword (in 80386 and above)

  - between register and memory location addressed by index or base register
    - index and base registers where OA is held: BP, BX, DI, SI
    - in 80386, memory location should be addressed by EAX, EBX, ECX, EDX, EBP, EDI, ESI
  - e.g. MOV AX, [BX]
    - MOV AL, [ECX] loads AL from the DS OA selected by ECX
  - In 64 bit mode, if only 32 bit compatible, 32 bit indirect address is used.
  - [ ] means indirect addressing.
  - In fully 64 bit mode, any register can be used
    - But none in flat model
  - The data segment is used by default with register indirect addressing or any other addressing mode that uses BX, DI, or SI to address memory.
  - If the BP register addresses memory, the stack segment is used by default.
  - In the 64-bit mode, the segment registers are not used in the address calculation because the register contains the actual linear memory address.
  - In some cases, this type of addressing requires specifying type of the data by using special assembler directives like BYTE PTR, WORD PTR, DWORD PTR, or QWORD PTR(64 bit), or even OWORD PTR(octal word - 128bit)
    - They indicate size of memory data accessed by the memory pointer (PTR)
    - E.g. MOV BYTE PTR [DI], 10H
    - used only with instructions that address a memory location through a pointer or index register with immediate data,
  - Allows access to tabular data by referring to starting location.
    - For this, load the starting location of the table into the BX register with a MOV immediate instruction. This location contains a counter in DOS.
    - The count is initialised in CX. Repeat until CX=0 loop is used.
    - After initialising the starting address of the table, use register indirect addressing to store the rest of the table data sequentially .
    - The OFFSET directive tells the assembler to load BX with the offset address of memory location TABLE, not the contents of TABLE.
- **Base-plus-Index Addressing:**
  - byte or word
  - B/w register and memory location addressed by base register BP or BX plus index register DI or SI.
    - Base register:  beginning location of memory array
    - index register: relative position of the element in the array.
    - This combination gives you the OA to add to DS.
  - in 80386 and above, any two from EAX, EBX, ECX, EDX, EBP,EDI, or ESI can be combined.
  - Also indirectly addresses memory
  - With BP, both SS and BP generate the effective address.
  - Written as [BX+DI] or [BX][DI](in Intel ASM assembler).
  - E.g. MOV DX, [BX+DI]  or MOV DX, [BX][DI].
- **Register Relative Addressing:**
  - Byte or word
  - between register and memory location addressed by an index or base register plus a displacement
    - BP(for SS), (for DS):BX, DI or SI
  - e.g. MOV AX, [BX+4] or MOV AX,ARRAY[BX] or MOV AL,DATA[DI+3]
    - loads into AX - DS address formed by BX+4
    - loads into AX - DS memory location in ARRAY + BX
  - In the 8086–80286 microprocessors, the value of the displacement is limited to a 16-bit signed number with a value ranging between +32,767 (7FFFH) and –32,768 (8000H)
  - In the 80386 and above, a 32-bit displacement is allowed with a value ranging between +2,147,483,647 (7FFFFFFFH) and -2,147,483,648(80000000H).
- **Base Register-plus-Index Addressing:**
  - byte or word
  - between a register and memory location addressed by a base and an index register plus a displacement.
  - E.g. MOV AX, ARRAY[BX+DI], MOV AX, [BX+DI+4]
    - loads into AX - DS address formed by BX+DI+4
    - loads into AX - DS memory location in ARRAY + BX +DI
  - Uses 2d array of memory data
  - Least used
- **Scaled-Index Addressing:**
  - Only in 80386 Pentium 4

  - the second register is modified by a scale factor of 1, 2(word), 4(doubleword) or 8(quadword) to generate operand address.
  - E.g. MOV EDX, [EAX, 4*EBX], MOV AL, [2*EBX]
    - loads EDX from the DS memory location addressed by EAX plus four times EBX.
  - Allows access to word, double word, quadword
- **RIP Relative Addressing:**
  - Only for 64 bit extensions on Pentium 4 or Core2
  - allows access to any location in the memory system by adding a 32-bit displacement to the 64-bit contents of the 64-bit IP.
  - The displacement is signed so data located within from the instruction is accessible by this addressing mode.
  - Addresses a linear location in the flat memory model
- .MODEL TINY directs the assembler to assemble the program into a single code segment.
- Model size SMALL allows one data segment and one code segment
  - Often used whenever memory data is required for a program
  - Assembles as an execute program file (.EXE)
- .CODE statement or directive indicates the start of the code segment.
- .STARTUP statement indicates the starting instruction in the program.
  - Also loads the data segment register with the SA.
- .EXIT statement causes the program to exit to DOS.
- END statement indicates the end of the program file
- *Label Opcode Operand Comment*
  - Label: a symbolic name to identify the memory location for storing data
    - Begin with letter or special character: @, $, -, ?
    - 1-35 characters
  - Opcode: to hold the instruction
  - Operand: Contains information used by the opcode
    - 0-3 operands
  - Comment: begins with a semicolon (;)
- The hexadecimal number at left (1000H) is the offset address of the instruction or data. This number is generated by the assembler.
- The number or numbers to the right of the offset address are the machine-coded instructions or data that are also generated by the assembler.
- .386 after .MODEL to select 80386 microprocessor and .486 is 80486, .586 for Pentium, .686 for upper versions of Pentium like (Pentium Pro, Pentium4) and for Core2.
- **Data Structures:**
  - used to specify how information is stored in a memory array.
  - template for data
  - start defined with STRUC directive
  - end defined with ENDS statement
  - Literals are surrounded with apostrophes and the entire field is surrounded with < > symbols when the data structure is used to define data.
  - When data are addressed in a structure, use the structure name and the field name to select a field from the structure.
    - like NAME.STREET
  - *See example 3-12 on page 119 of PDF*

## 3.2 Program Memory-Addressing Modes

- Used with JMP and CALL instructions
- 3 forms: direct , relative and indirect
- **Direct PMA:**
  - Not often used as compared to rest
  - Stores address with the opcode
  - Intersegment jump: a jump to any memory location within the entire memory system.
  - if a program jumps to memory location 10000H for the next instruction, the address(10000H) is stored following the opcode in the memory.
  - The direct jump is often called a far jump because it can jump to any memory location for the next instruction
    - In real mode, any location within first 1M byte of memory
    - Changes both CS and IP
    - In protected mode, any location in the entire 4G byte address range by accessing a new CS descriptor from its table
  - In 64 bit pentium 4 and Core 2, jump and call can be to any memory location in the system.
    - CS is not used for the address but to contain a pointer to a descriptor that describes the access rights and privilege level of the CS.
  - Name of a memory address, label, refers to the location called or jumped to instead of numeric address.
- **Relative PMA**

    - Means relative to IP
    - if a JMP instruction skips the next 2 bytes of memory, the address in relation to the IP is a 2 that adds to it, developing the address of the next program instruction.
    - JMP is a 1 byte instruction with a 1 or 2 byte displacement
        - 1 byte displacement(+127 to -128 bytes) - short jump
        - 2 byte displacement(+-32K bytes) - near jump
        - Considered as intrasegment jumps
    - Intrasegment jump: a jump anywhere within the current code segment.
    - Relative JMP and CALL instructions contain either an 8-bit or a 16-bit signed displacement that allows a forward memory reference or a reverse memory reference.
    - +-2G bytes in 32 bit displacement
- **Indirect PMA**
    - If a 16-bit register holds the address of a JMP instruction, the jump is near.
    - if the BX register contains 1000H and a JMP BX instruction executes, the microprocessor jumps to OA 1000H in the current CS
    - If a relative register holds the address, the jump is also considered to be an indirect jump.
    - JMP [BX] refers to the memory location within the DS at the OA contained in BX.
        - At this OA is a 16-bit number that is used as the OA in the intrasegment jump. This type of jump is sometimes called an indirect-indirect or double-indirect jump.

## 3.3 Stack Memory-Addressing Modes

- Stack holds data temporarily and stores the return addresses used by procedures
- LIFO memory
- PUSH and POP instructions used, store words of data not bytes
- The CALL instruction also uses the stack to hold the return address for procedures and a RET (return) instruction to remove the return address from the stack.
- Maintained by SP/ESP and SS.
- On PUSH,
    - higher order 8 bits placed into location addressed by SP-1
    - Lower order 8 bits placed into SP-2
    - SP is then decremented by 2 to store next word
- In protected mode operation, the SS register holds a selector that accesses a descriptor for the BA of the SS.
- On POP,
    - low order 8 bits removed from SP
    - high order 8 bits removed from SP+1
    - SP incremented by 2
- Only CS cannot be popped into.
- the 64-bit registers can be pushed or popped from the stack, but they are 8 bytes in length.
- The PUSHA and POPA instructions either push or pop all of the registers, except segment registers, onto the stack.
    - Not available in early 8086/8088
    - Not in 64 bit mode for Pentium4 or Core2