BCA Part I

Paper-VII

Topic: Addressing Schemes

Prepared by: Dr. Kiran Pandey

Email-id: kiranpandey.nou@gmail.com

INTRODUCTION

Addressing scheme is defined as a method of specifying the effective address of the operands in memory. Whenever the microprocessor executes instructions, it performs a very specific function on data. These data terms often referred to as operands, may be part of the instruction, may be residing in one of the internal registers of the microprocessor or may be stored at a specific address in memory. To access these different types of operands, the microprocessor is provided with various addressing mode. The addressing mode increases the flexibility of the programming language and is useful in implementing the constructs and data structures of the powerful high level programming languages.

Types of Addressing Schemes

We will describe the different addressing schemes with reference to 8086 architecture. Most of the machines employ a set of addressing modes. We will describe some very common addressing modes used in most of the machines. A tree of common addressing mode is given in figure below:

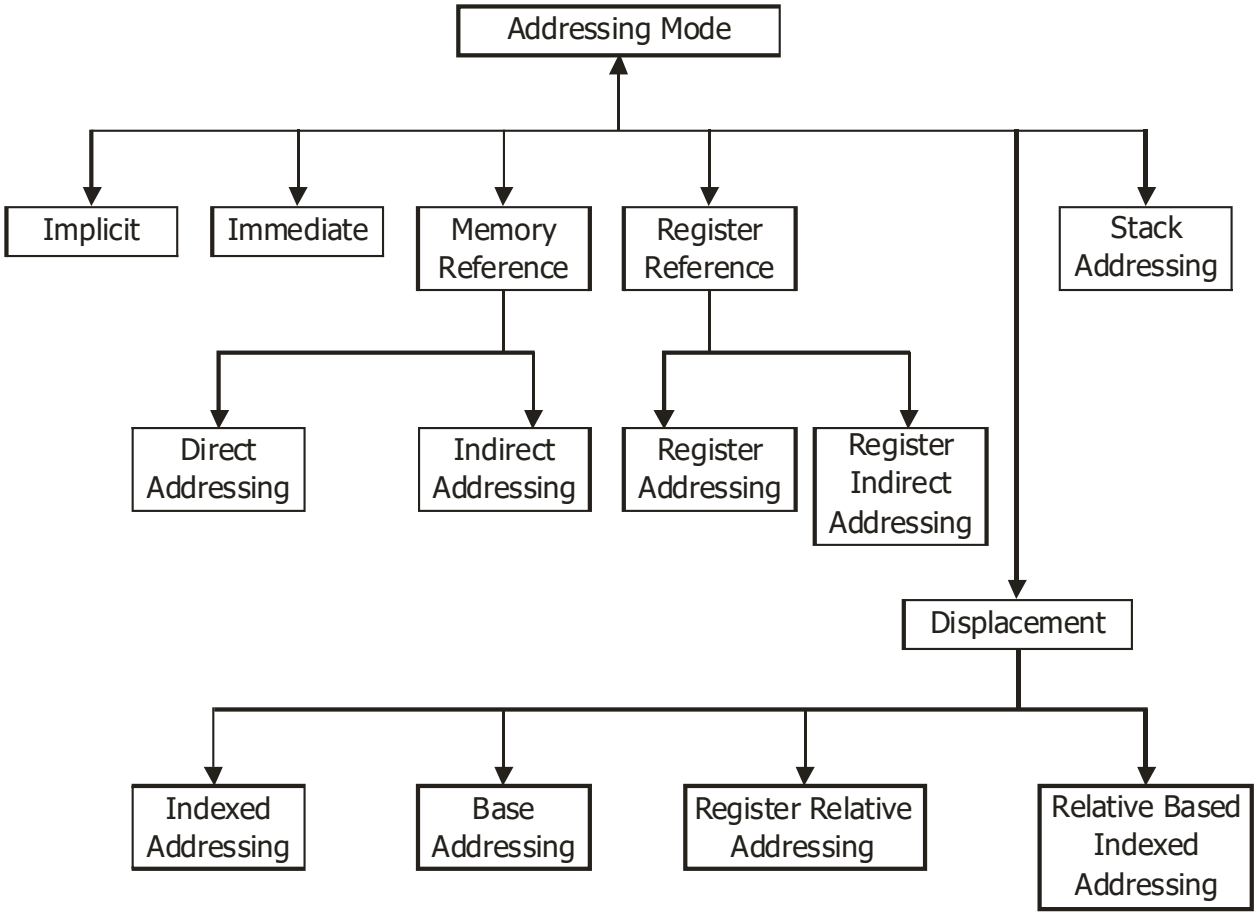

Figure 1 : Addressing Modes

The three components used in accessing the memory operands are:

(i) Base registers: BX and BP

(ii) Index Register : SI and DI

(iii) Displacement : 8 bit sign extended or 16 bit value.

The base, index and displacement can be combined as,

BX SI

or + or + Displacement (disp) = memory address of the operand.

BP DI

(Base)

We will use MOV instruction to explain the different addressing modes. The format of MOV is :

MOV dest, source. {It move the content of the source to destination}

Implicit Addressing Mode :

In this mode, the data is assumed to be are defined in the instruction. For example, in the statements CMC, STD or CLC, etc. the data is in the flag register. Such instructions are called zero address instruction.

Immediate Addressing Mode

In this mode data is the part of the instruction. It does not involve computation of address.

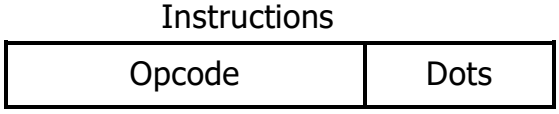

Figure 2 : Immediate addressing

Example :

MOV AX, 100 ; AX $\leftarrow$ 100

The data 100 is the part of the instructions. The value 100 is moved into the register AX.

Some important points of immediate addressing :

(i) Used for initialing the value of a variable.

(ii) No additional memory access required for executing the instructions.

(iii) The size of the instruction and operand field limited.

Direct Addressing Mode

In this mode the address of the data is a part of the instructions.

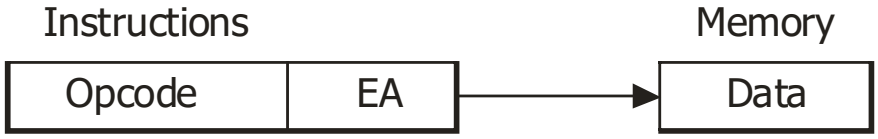

Figure 3 : Direct Addressing

For example :

MOV AX, [1050H]

The data is in the memory location pointed by the address 1050H. Specified constant value gives the offset.

Important point to remember :

(i) This addressing scheme facilitates global variable with fixed offset values to be addressed directly.

(ii) Provides a limited space for address because if the address field has in bits then memory space would be 2^n memory location.

(iii) Only one memory reference is required to fetch the operand.

Indirect Addressing

In this mode, the address of the operand is in the register. The data on which the instruction operates is pointed to by the contents of the register which is a part of the instruction. It requires two memory reference.

For example :

MOV AX, [BX]

The content of register BX is copied to AX.

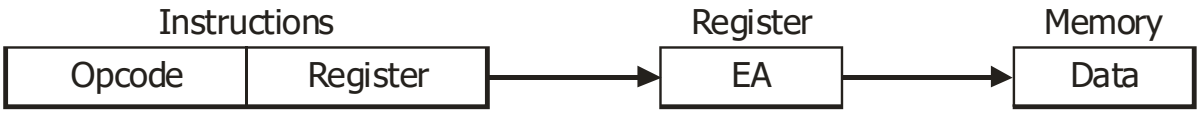

Figure 4 : Indirect mode/Register indirect

Register Addressing Mode

In this mode the operands on which instructions operate are stored in the specified registers. For a 16-bit operand, a register may be AX, BX, CX, DX, SI, DI, SP and BP and for 8-bit operand, a register may be AH, AL, BH, BL, CH, CL, and DH, DL.

Examples :

```
MOV   AX,   DX    ;     AX ← DX
MOV   BX,   AX    ;     BX ← AX
SUB   BP,   CX    ;     BP ← BP — CX
```

It is similar to direct addressing except that the register name or number is substituted for memory address.

Points to remember :

(i) Register access is faster than memory access results in faster instructions execution.

(ii) The size of register is smaller than memory address. This reduces the instruction size.

Figure 5 : Register Addressing

Register Indirect Mode

The effective address (EA) of the data is in the base register BX or in index register SI and DI that is specified in the instruction.

That is

(BX)

EA = (SI)

(DI)

In the instruction EA is designated as [Register].

Examples :

MOV AX, [BX]; copy BX content to AX.

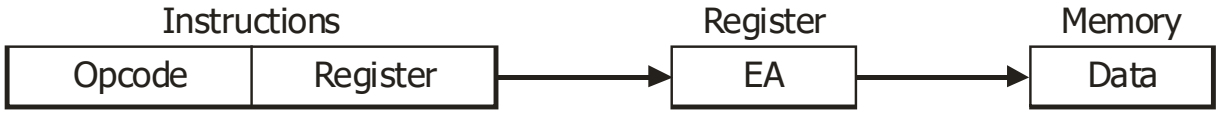

Figure 6 : Register Indirect

Indexed Addressing Mode

In this mode the operand field of the instruction contains an address and an index register, which contains an offset. This addressing is used to address the consecutive location of memory.

For example :

MOV DX, [SI] + 5H

Here the physical address (PA) obtained

as $PA = \begin{Bmatrix} SI \\ DI \end{Bmatrix} + \begin{Bmatrix} 8\text{-bit displacement} \\ 16\text{-bit displacement.} \end{Bmatrix}$

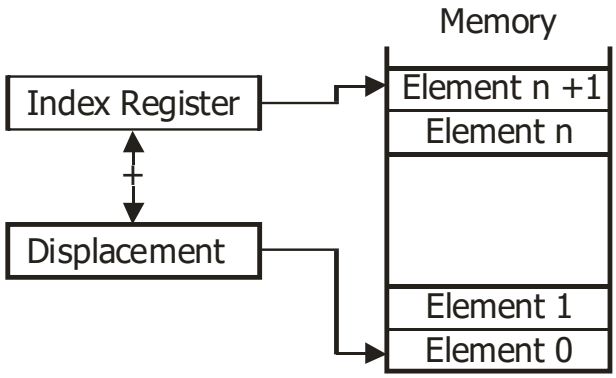

Figure 7 : Indexed Addressing

Base Addressing

In this mode effective address of the operand is obtained by addressing a direct or indirect displacement to the content of either base register BX or base pointer BP. The physical address (PA) can be calculated as :

$$PA = \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} 8\text{ - bit displacement} \\ 16\text{ - bit displacement.} \end{Bmatrix}$$

For example :

```
MOV   AX, [BX]
MOV   CX, [BX] + 10H
MOV   AL, [BP] + 5H
```

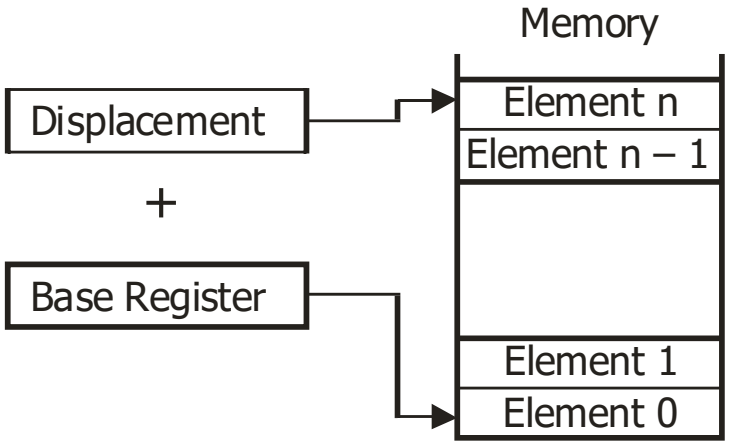

Figure 8 : Base Addressing

Register Relative Mode

In this mode the effective address (EA) is generated by adding the content of the base register or index register to an 8-bit or a 16-bit displacement value that is,

```
EA  =  (BX) + disp
       (BP) + disp
       (SI) + displ
       (DI) + displ
```

Examples

(1) A [BX]

(2) [BX + 20]

The process of register relative mode can be shown as follows :

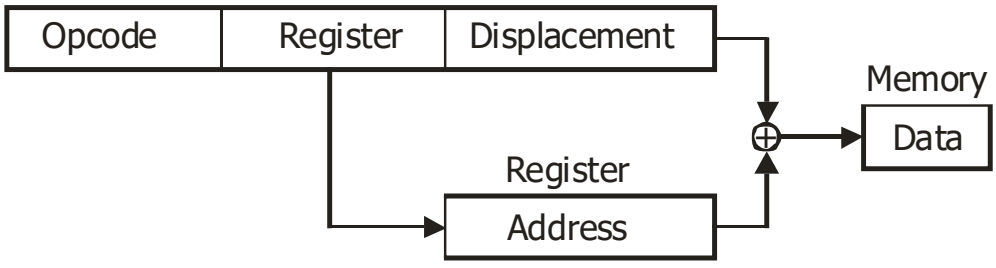

Figure 9 : Register relative mode

Relative Based Indexed Mode

The effective address (EA) is computed by addressing the contents of the base register and index register with an 8-bit or a 16-bit displacement value that is :

EA = (BX) + (SI) + disp
(BX) + (DI) + disp
(BP) + (SI) + disp
(BP) + (DI) + disp

Examples

(1) [BP + 2] [SI + 1]

(2) A [BX] [SI]

(3) X-ARRAY [BP] [SI + 2]

The process of relative based indexed addressing mode is shown below in the figure.

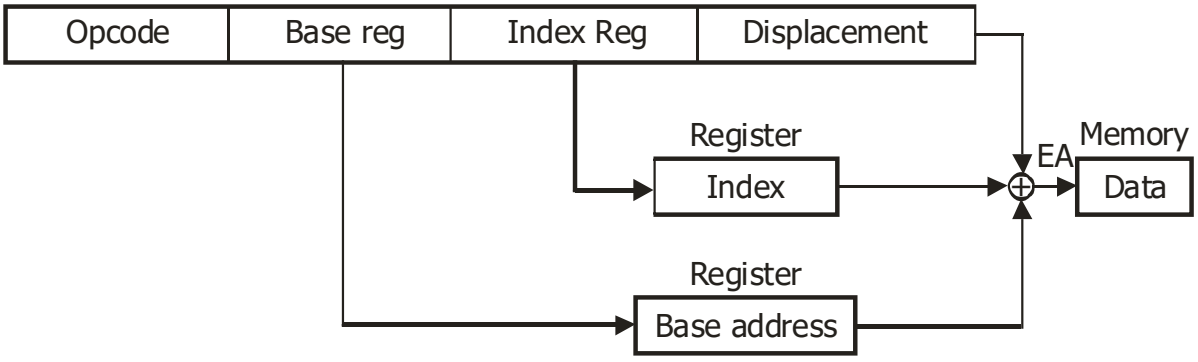

Figure 10 : Relative based index addressing

Stack Addressing

In this addressing scheme, the operand is implied as top of stack. It is not explicit, but implied. It uses a CPU Register called Stack Pointer (SP). The SP points to the top of the stack i.e. to the memory location where the last value was **pushed**. A stack provides a sort-of indirect addressing and indexed addressing. This is not a very common addressing scheme. The operand is found on the top of a stack. In some machines the top two elements of stack and top of stack pointer is kept in the CPU registers, while the rest of the elements may reside in the memory. Figure 14 shows the stack addressing schemes.

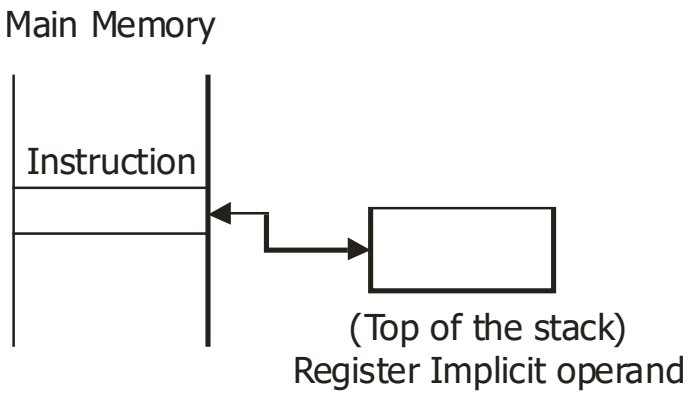

Figure 11 : Stack Addressing