```
        MOV   CX,05        ;CX holds the loop count
        MOV   BX,0200H     ;BX holds the offset data address
        MOV   AL,00        ;initialize AL
ADD_LP: ADD   AL,[BX]      ;add the next byte to AL
        INC   BX           ;increment the data pointer
        DEC   CX           ;decrement the loop counter
        JNZ   ADD_LP       ;jump to next iteration if counter not zero
```

1.6 80x86 Addressing Modes

The CPU can access operands (data) in various ways, called addressing modes. The number of addressing modes is determined when the microprocessor is designed and cannot be changed. The 80x86 provides a total of seven distinct addressing modes:

A. Register addressing mode

The register addressing mode involves the use of registers to hold the data to be manipulated. Memory is not accessed when this addressing mode is executed; therefore, it is relatively fast. **Examples** of register addressing mode follow:

```
MOV BX, DX  ;copy contents of DX into BX
MOV ES, AX   ;copy contents of AX into  ES
ADD AL, BH   ;add the contents of BH to the contents of AL and store in AL
```

It should be noted that the source and destination registers must match in size. In other words coding "MOV CL,AX" will give an error, since the source is a 16-bit register and the destination is an 8-bit register.

B. Immediate addressing mode

In the immediate addressing mode, the source operand is a constant. In immediate addressing mode, as the name implies, when the instruction is assembled, the operand comes immediately after the opcode. For this reason, this addressing mode executes quickly. However, in programming it has limited use. Immediate addressing mode can be used to load information into any of the registers except the segment registers and flag registers. **Examples:**

```
MOV AX 3F50H ;move 3F50H into AX
MOV CX, 425 ;load decimal value 425 into CX
MOV BL, 40H ;load 40H into BL
```

To move information to the segment registers, the data must first be moved to a general-purpose registers and then to the segment register. **Example:**

```
MOV AX,2550H
MOV DS,AX
```

In other words, the following would produce an error:

```
MOV DS,0123H ;illegal!!
```

In the first two addressing modes, the operands are either inside the microprocessor or tagged along with the instruction. In most programs, the data to be processed is often in some memory location outside the CPU. There are many ways of accessing the data in the data segment. The following describes those different methods.

C. Direct addressing mode

In the direct addressing mode the data is in some memory location(s) and the address of the data in memory comes immediately after the instruction. Note that in immediate addressing, the operand itself is provided with the instruction, whereas in direct addressing mode, the address of the operand is provided with the instruction. This address is the offset address and one can calculate the physical address by shifting left the DS register and adding it to the offset as follows:

```
MOV DL,[2400] ;move contents of DS:2400H into DL
```

In this case the physical address is calculated by combining the contents of offset location 2400 with DS, the data segment register. Notice the bracket around the address. In the absence of this bracket it will give an error since it is interpreted to move the value 2400 (l6-bit data) into register DL, an 8-bit register. **Example 1-15** gives another example of direct addressing.

Example 1-15

Find the physical address of the memory location and its contents after the execution of the following, assuming that DS = 1512H.

```
MOV     AL,99H
MOV     [3518],AL
```

Solution:

First AL is initialized to 99H, then in line two, the contents of AL are moved to logical address DS:3518 which is 1512:3518. Shifting DS left and adding it to the offset gives the physical address of 18638H (15120H + 3518H = 18638H). That means after the execution of the second instruction, the memory location with address 18638H will contain the value 99H.

D. Register indirect addressing

In the register indirect addressing mode, the address of the memory location where the operand resides is held by a register. The registers used for this purpose are SI, DI, and BX. If these three registers are used as pointers, that is, if they hold the offset of the memory location, they must be combined with DS in order to generate the 20-bit physical address. **For example:**

```
MOV    AL,[BX]          ;moves into AL the contents of the memory location
                        ;pointed to by DS:BX.
```

Notice that BX is in brackets. In the absence of brackets, it is interpreted as an instruction moving the contents of register BX to AL (which gives an error because source and destination do not match) instead of the contents of the memory location whose offset address is in BX. The physical address is calculated by shifting OS left one hex position and adding BX to it. The same rules apply when using register SI or Or DI

```
MOV    CL,[SI]          ;move contents of DS:SI into CL
MOV    [DI],AH          ;move contents of AH into DS:DI
```

In the examples above, the data moved is byte sized. Example 1-16 shows 16-bit operands.

Example 1-16

Assume that DS = 1120, SI = 2498, and AX = 17FE. Show the contents of memory locations after the execution of

```
MOV [SI],AX
```

Solution:

The contents of AX are moved into memory locations with logical address DS:SI and DS:SI + 1; therefore, the physical address starts at DS (shifted left) + SI = 13698. According to the little endian convention, low address 13698H contains FE, the low byte, and high address 13699H will contain 17, the high byte.