Displacement Addressing: Displacement addressing is almost identical to direct addressing except that the instructi is four bytes wide instead of three.

If the operation of the MOV CL, DS:[1234H] instruction is compared to that of the MOV AL, DS:[1234H] instruction both basically perform the same operation except for the destination register (CL versus AL). Another difference is that the MOV AL, DS:[instruction is 3 bytes long and the MOV CL, DS:[1234H] instru is 4 bytes long.

4. Register Indirect Addressing

Register indirect addressing allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI and SI (only those can be used)

Example: If BX = 1000H, DS = 3000H & the µP exec MOV AX, [BX] instruction what physical Addresses that the AX recives its contents?

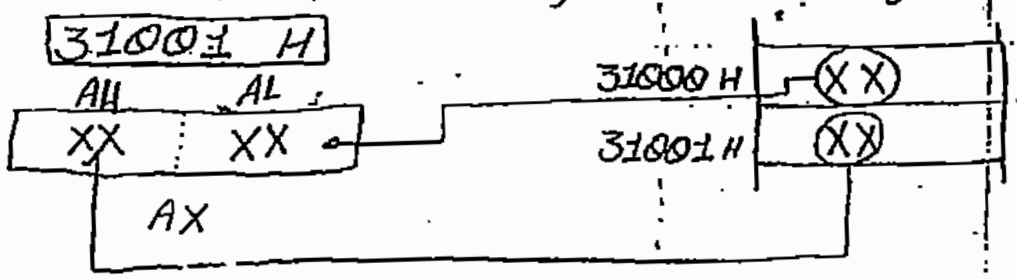
Sol

DS = 3000H

& BX = 1000 = offset

∴ AL will receive a byte from Memory address 30000H + 1000H = 31000H

& AH will receive a byte from memory addres 31001 H



Some typical instructions using indirect addressing appear in table 2.4

Assembly	Size	Operation
MOV CX, [BX]	16-bit	CL ← [DS:[BX]] CH ← [DS:[BX]+1]
** MOV [BP], DL	8-bit	[SS:[BP]] ← DL
MOV [DI], BH	8-bit	[DS:[DI]] ← BH
MOV [DI], [BX]	we can not use this instruction because Memory to Memory moves are Not allowed..	

** Note! Data addressed by BP and by default located in the stack segment SS, while all other indirect addressing modes use the Data Segment DS by default.

Note: We can directs the instruction to a specific segment as shown in the following instructions

MOV CH, ES:[BX] ... CH ← [ES:[BX]]

MOV DS:[DI], AL ... [DS:[DI]] ← AL

Note: In some cases, indirect addressing requires specifying the size of the data with the special assembler directives BYTE PTR or WORD PTR. These directives indicate the size of the memory data addressed by the memory pointer (PTR).
For example, the MOV AL, [DI] instruction is clearly a byte-sized move instruction, but the MOV [DI], 10H instruction is ambiguous.

The instruction MOV BYTE PTR [DI], 10H clearly designates the location addressed by DI as a byte-sized memory location. The BYTE PTR and WORD PTR directives are used only with instructions that address a memory location through a pointer or index register with immediate data.

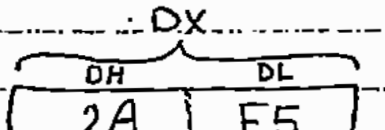
5. Base-Plus-Index Addressing.

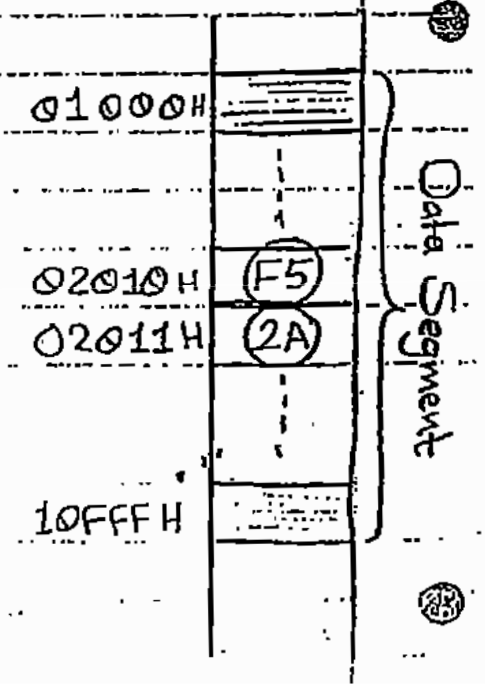
Base-plus-Index addressing is similar to indirect addressing because it indirectly addresses memory data. This type of addressing uses one base register (BP or BX), and one index register (DI or SI) to indirectly address memory. The base register often holds the beginning location of a memory array while the index register holds the relative position of an element in the array.

Example: Consider the instruction that uses Base plus Index mode:

MOV DX, [BX+DI] with:

BX = 1000H
DI = 0010H
DS = 0100H





Start address of the DS = 01000H.
End address of the DS = 10FFFH

offset = BX + DI = 1010H

Physical address = 01000 + 1010H
= 02010H

Table 2.5 shows an examples of the base-plus-Index addressing mode

Assembly Instruction	Size	Operation
MOV CX, [BX+DI]	16-bit	CL ← [DS:[BX+DI]] CH ← [DS:[BX+DI]+1]
MOV CH, [BP+SI]	8-bit	CH ← [SS:[BP+SI]]
MOV [BX+SI], SP	16-bit	[DS:[BX+SI]] ← SP(L) [DS:[BX+SI]+1] ← SP(H)
MOV [BP+DI], AH	8-bit	[SS:[BP+DI]] ← AH

Important Note: Remember that whenever BP addresses memory data, both the Stack Segment (SS) register and Base Pointer (BP) generate the effective address.

Note: In some assemblers we can write the instruction:

MOV CX, [BX + DI] as

MOV CX, [BX][DI].

6. Register Relative Addressing.

It is similar to base-plus-index addressing and displacement addressing. In register relative addressing, the data in a segment of memory are addressed by adding the displacement to the contents of a base or an index register (BP, BX, DI, or SI).

Example: MOV AX, [BX + 1000H] BX = 0100H
DS = 0200H

(*) So the address generated is the sum of DS * 10H, BX, and the displacement 1000 H = $\boxed{03100H}$.

(*) Remember that BX, DI, or SI addresses the data segment (DS and BP addresses the stack segment.

(*) The displacement can be a number added to the register within the [ ], as in the MOV AL, [DI + 2] instruction, or it can be a displacement subtracted from the register as in MOV AL, [SI-2].

* A displacement also can be an offset address appended to the front of the [ ], as in MOV AL, 1000H[DI]

* Both forms of displacements also can appear simultaneously, as in the MOV AL, 1000H[DI+3] instruction.

* In the 8086/8088 80286 µP's 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)

Table 2-6 lists a few instructions that use register relative addressing.

Assembly Instructions	Size	Operation
MOV AX, [DI + 100H]	16-bit	Copies the word contents of the data segment memory location addressed by DI plus 100H into AX
MOV 3000[SI], BL	8-bit	Copies BL into the data segment memory location addressed by 3000 + SI
MOV 2000[SI+2], CL	8-bit	Copies CL into the data segment memory location addressed by sum of 2000 + SI + 2

7. Base Relative Plus Index Addressing.

The base relative plus index addressing mode is similar to the base plus inde addressing mode, but it adds a displacement, besides using a base register and an index register, to form the memory address. This type of addressing mode often addresses a two-dimensional array of memory date.

Example: Consider the instruction:

MOV AX, [BX+SI+100H] BX = 0020H
SI = 0010H
DS = 1000H

So the effective address for this instruction is:

10000 (DS × 10H)
offset { 0010 (SI)
0020 (DS)
0100 (displacement
10130H effective address

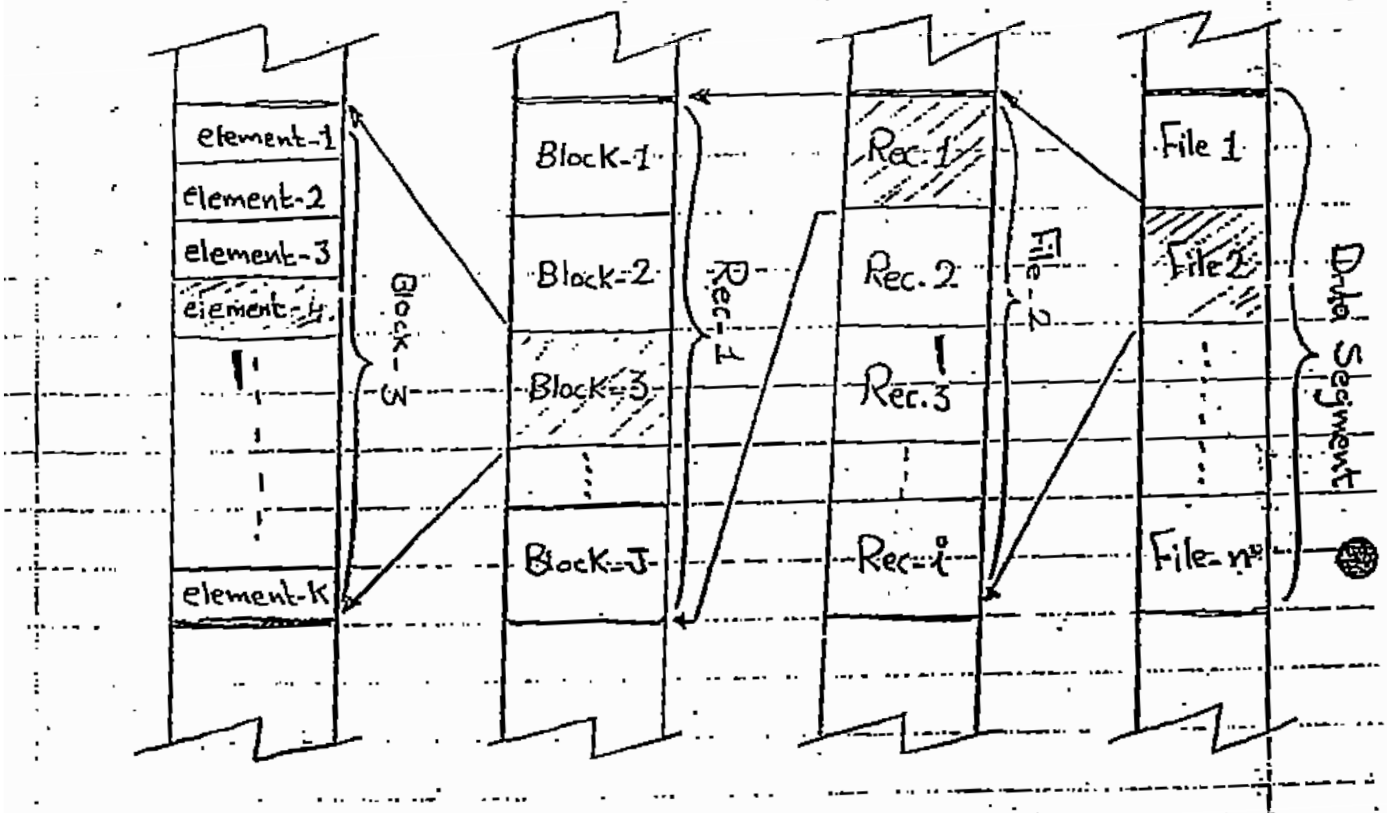
Some typical instructions using base relative plus index addressing appear in Table 2.7.

Assembly Instruction	Size
MOV DH, [BX + DI + 20H]	8-bit
MOV AX, 3000H[BX + DI]	16-bit
MOV 0F00H[BP + DI], CL	8-bit
MOV 0BBBH[BP + SI + 4], DH	8-bit

Base relative-plus-index addressing mode is the least used addressing mode. Suppose that a file of many records, each record consists of many Blocks exists in memory and each Block contains many elements then:

1. The displacement addresses the File.
2. The base register (BX) addresses a Record.
3. The Index register (DI or SI) addresses a Block.
4. 2-nd displacement addresses the Element.

Figure 2.1 illustrates this very complex form of addressing.



ex Suppose that the data segment has n-Files, each file has i-Records each record contains J-Blocks, in each Block there is K-elements. then if we want to read element 4 in Block 3 of Record 1 in File 2, we use the following values in the:

MOV. AL, disp.1[BX+SI+disp.2]

1. Displacement (disp.1) = size of File
2. Base Register (BX) = 0000H
3. Index register (SI) = [(size of Block) * 2]
4. Displacement (disp.2) = 3