Addressing Modes of 80386

Introduction

- Addressing modes indicate a way of locating data or operands.
- Described the type of operands and the way they are accessed for executing an instruction.
- The method by which address of source data and address of destination of result is given in the instruction is called as “ Addressing Modes”.
- The 80386 Microprocessor Provide 11 addressing modes.

Addressing Modes

- *Register addressing Mode*
- *Immediate addressing Mode*
- *Direct addressing Mode*
- *Register Indirect addressing Mode*
- *Based addressing Mode*
- *Index addressing Mode*
- *Scaled Index addressing Mode*
- *Based Index addressing Mode*
- *Based Scaled Index addressing Mode*
- *Based Index addressing Mode with Displacement*
- *Based Scaled Index addressing Mode with Displacement*

Register addressing Mode

- The data is stored in a register and it is referred using a particular register.
- All register accept IP used in this addressing mode.
- The 8/16/32 bit data required to execute an instruction is present in 8/16/32 bit register is given along with the instruction is called “Register addressing mode. “
- Example : ADD EAX,EBX

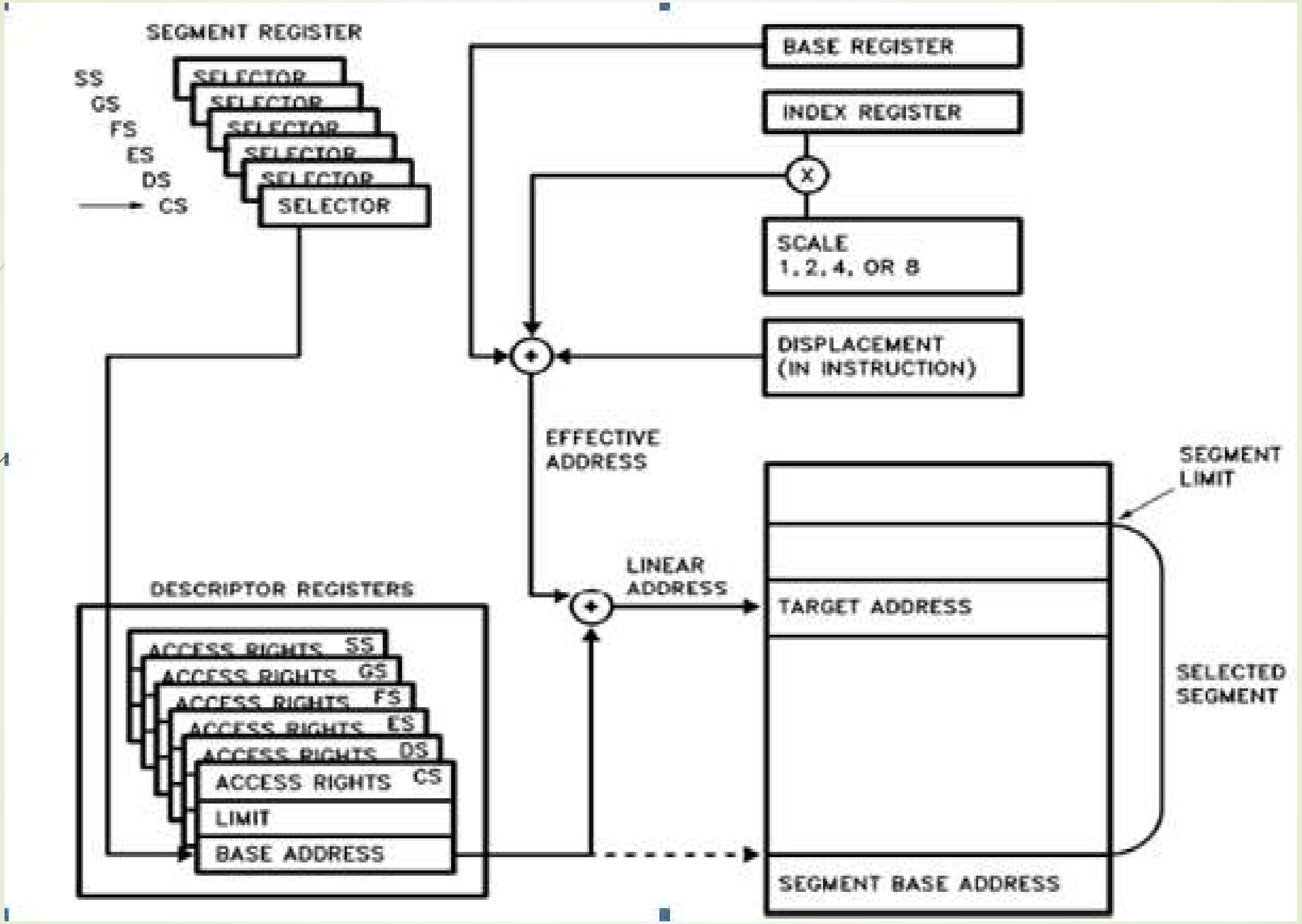
Immediate addressing Mode

- In this addressing mode, immediate data is part of instruction.
- The 8/16/32 bit data required to execute an instruction is given directly along with the instruction is called “Immediate addressing mode”.
- Example :
- Mov EAX,12345678H

Cont.....

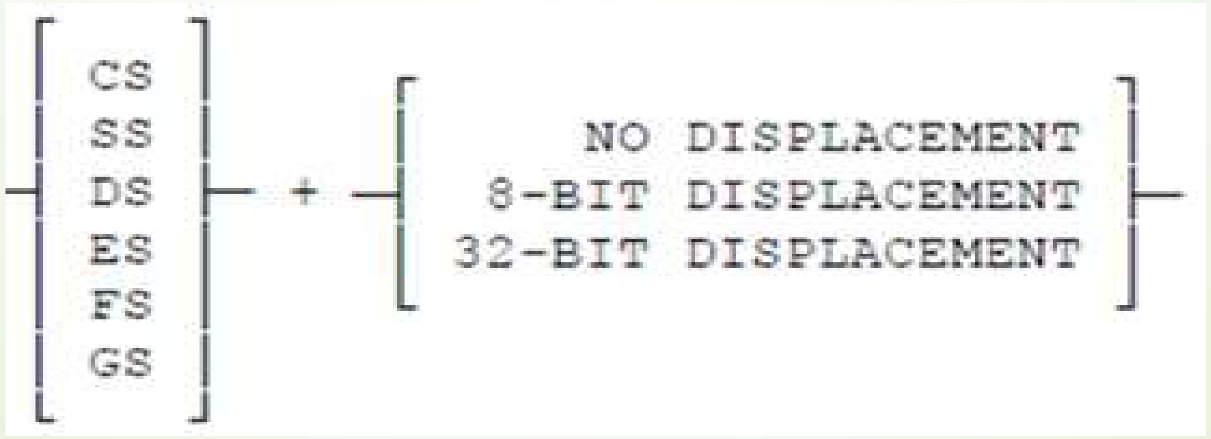
- The linear address consists of two components:
  - The segment base address and
  - An effective address.
- The effective address is calculated by using four address elements:
  - **DISPLACEMENT**: An 8-, or 32-bit immediate value
  - **BASE**: The contents of any general purpose register. It is generally used by compilers to point to the start of **the local variable area.**
  - **INDEX**: The contents of any general purpose register except for ESP. The index registers are used **to access the elements of an array, or a string of characters.**
  - **SCALE:** The index register's value can be multiplied by a scale factor, either 1, 2, 4 or 8. Scaled index mode is especially useful for accessing **arrays or structures.**

- The effective address (EA) of an operand is calculated according to the following formula:
- EA = Base Register + (Index Register * Scaling) + Displacement.


SEGMENT REGISTER
SS
GS
FS
ES
DS
CS
SELECTOR
SELECTOR
SELECTOR
SELECTOR
SELECTOR
SELECTOR
BASE REGISTER
INDEX REGISTER
X
SCALE
1, 2, 4, OR 8
DISPLACEMENT
(IN INSTRUCTION)
+
EFFECTIVE ADDRESS
SEGMENT LIMIT
LINEAR ADDRESS
+
TARGET ADDRESS
DESCRIPTOR REGISTERS
ACCESS RIGHTS SS
ACCESS RIGHTS GS
ACCESS RIGHTS FS
ACCESS RIGHTS ES
ACCESS RIGHTS DS
ACCESS RIGHTS CS
LIMIT
BASE ADDRESS
SELECTED SEGMENT
SEGMENT BASE ADDRESS

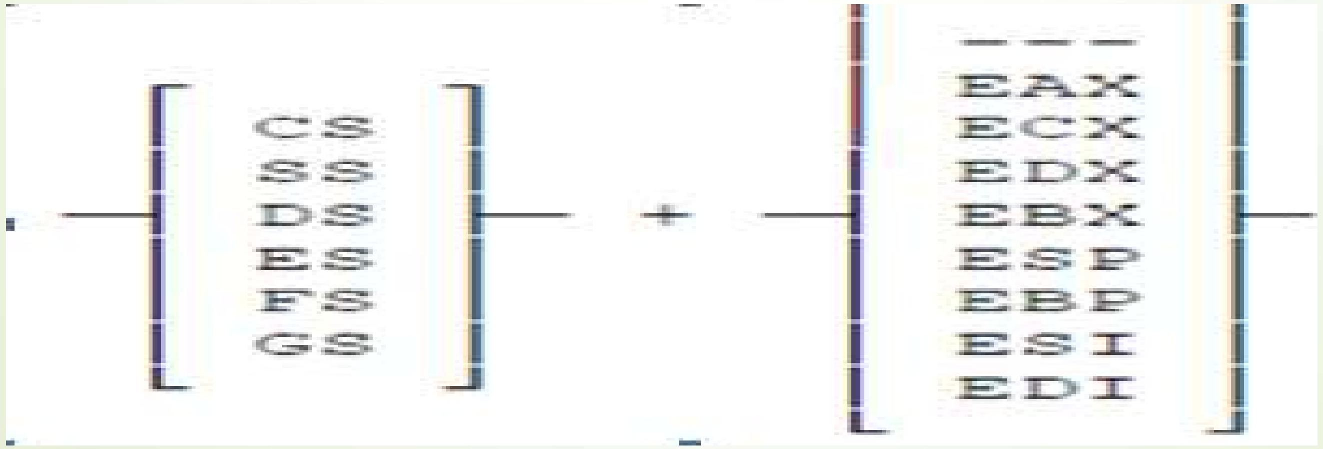
Direct addressing Mode

- The 8/16/32 bit data required to execute an instruction is present in memory location and effective address of this memory location is given directly along with the instruction then it is called “Direct addressing mode”.
- Example : Mov AX,[5000H]



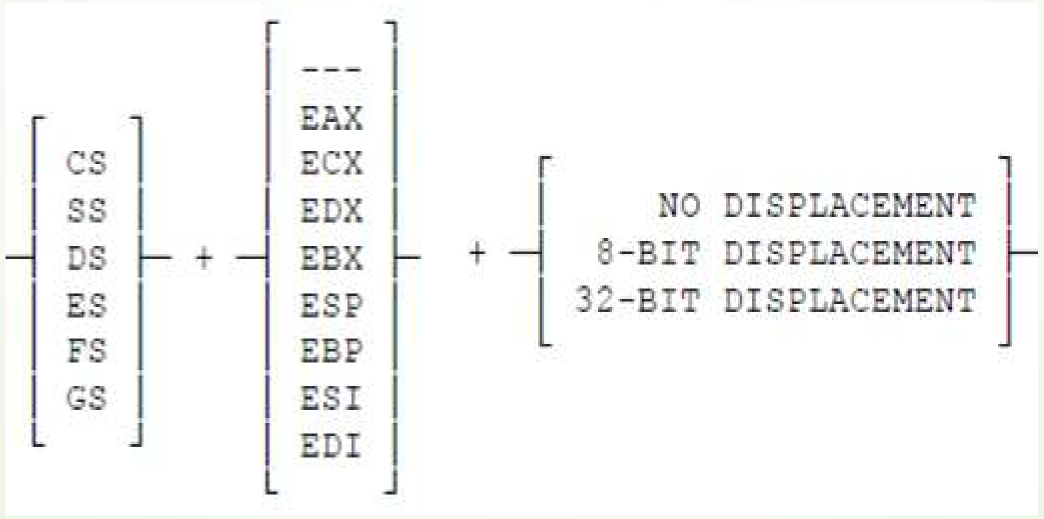
Register Indirect addressing Mode

- A base register will contain the address of operand
- Example: MOV [ECX], EDX
- The 8/16/32 bit data required to execute an instruction is present in memory location and effective address of that memory location is present in a 32 bit register and the name of this register is given along with the instruction then it is called “Register Indirect addressing mode”.



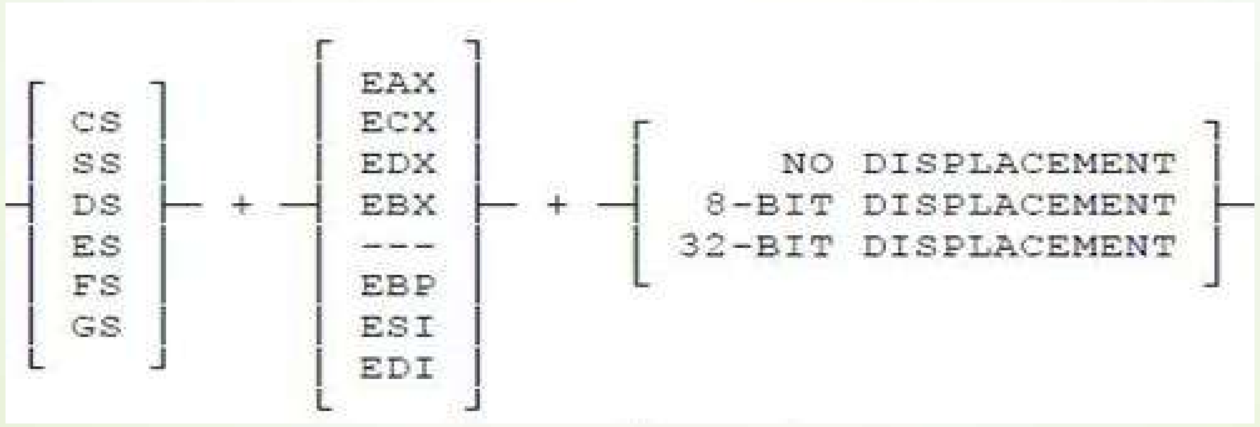
Based addressing Mode

- A BASE register's contents is added to a DISPLACEMENT to form the operands offset.
- Example: MOV ECX, [EAX+24]



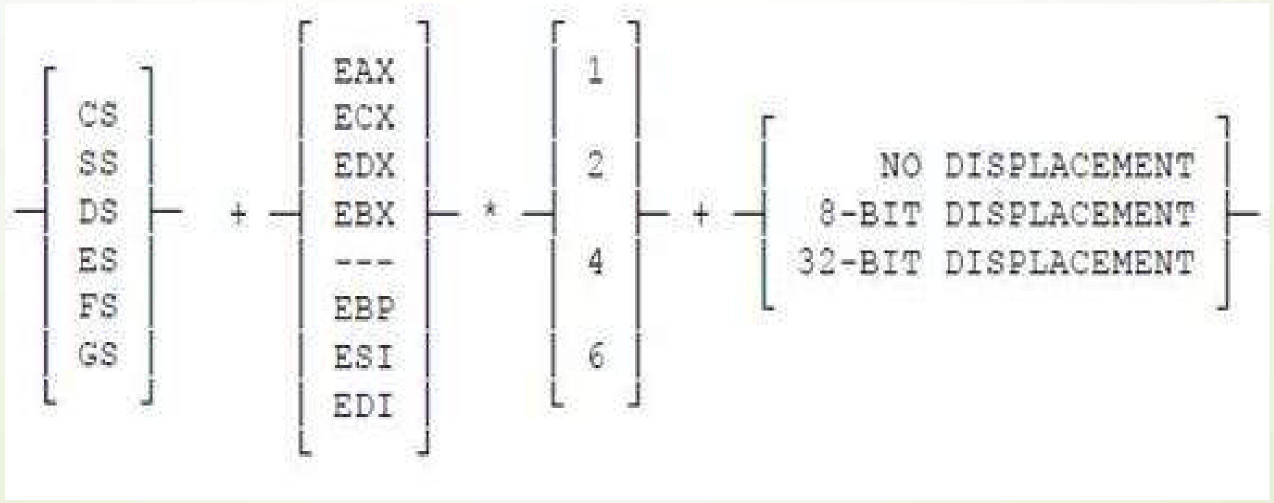
Index addressing Mode

- An INDEX register's contents is added to a DISPLACEMENT to form the operands offset. EXAMPLE: ADD EAX, TABLE[ESI]
- The 8/16/32 bit data required to execute an instruction is present in memory location and effective address of that memory location is obtained by adding three contents : 1. The 32 bit content of base register 2. The 32 bit content of Index Register 3. Displacement.



Scaled Index addressing Mode

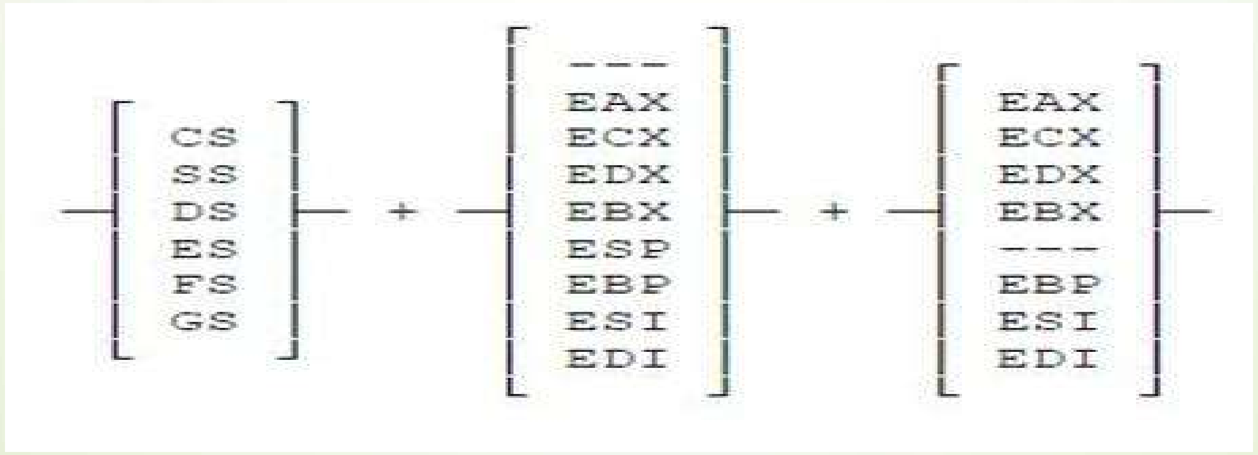
- An INDEX register's contents is multiplied by a scaling factor which is added to a DISPLACEMENT to form the operands offset.
- Example: MOV EBX, LIST[ESI*4]



Based Index addressing Mode

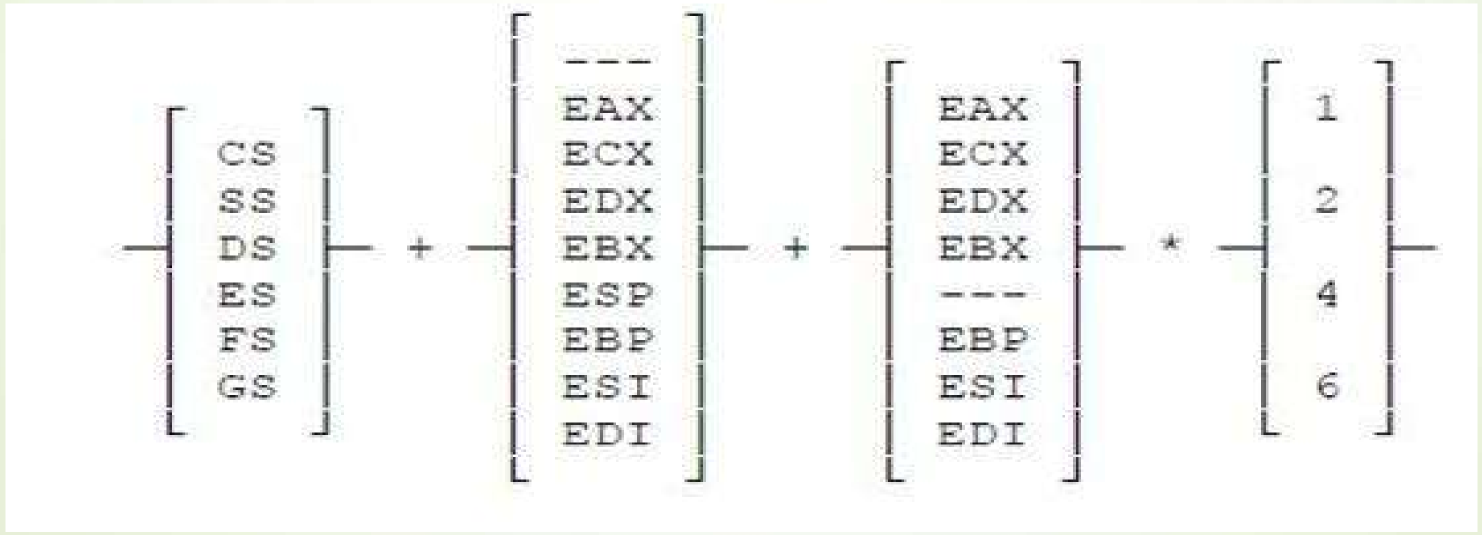
- The contents of a BASE register are added to the contents of an INDEX register to form the effective address of an operand.
- The 8/16/32 bit data required to execute an instruction is present in memory location and effective address of that memory location is obtained by adding two contents : 1. The 32 bit content of base register 2. The 32 bit content of Index Register is called as Based Index addressing Mode.

- Example: MOV EAX, [ESI] [EBX]



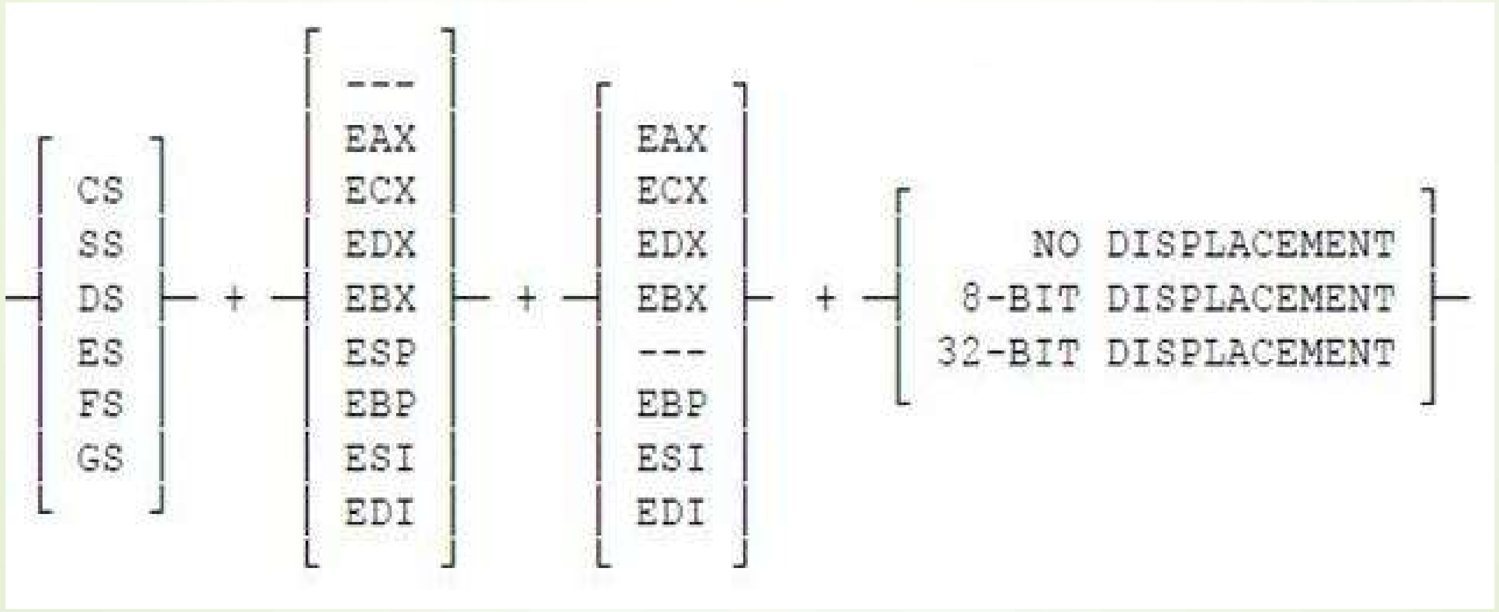
Based Scaled Index addressing Mode

- The contents of an INDEX register is multiplied by a SCALING factor and the result is added to the contents of a BASE register to obtain the operands offset.
- Example: MOV ECX, [EDX*8] [EAX]



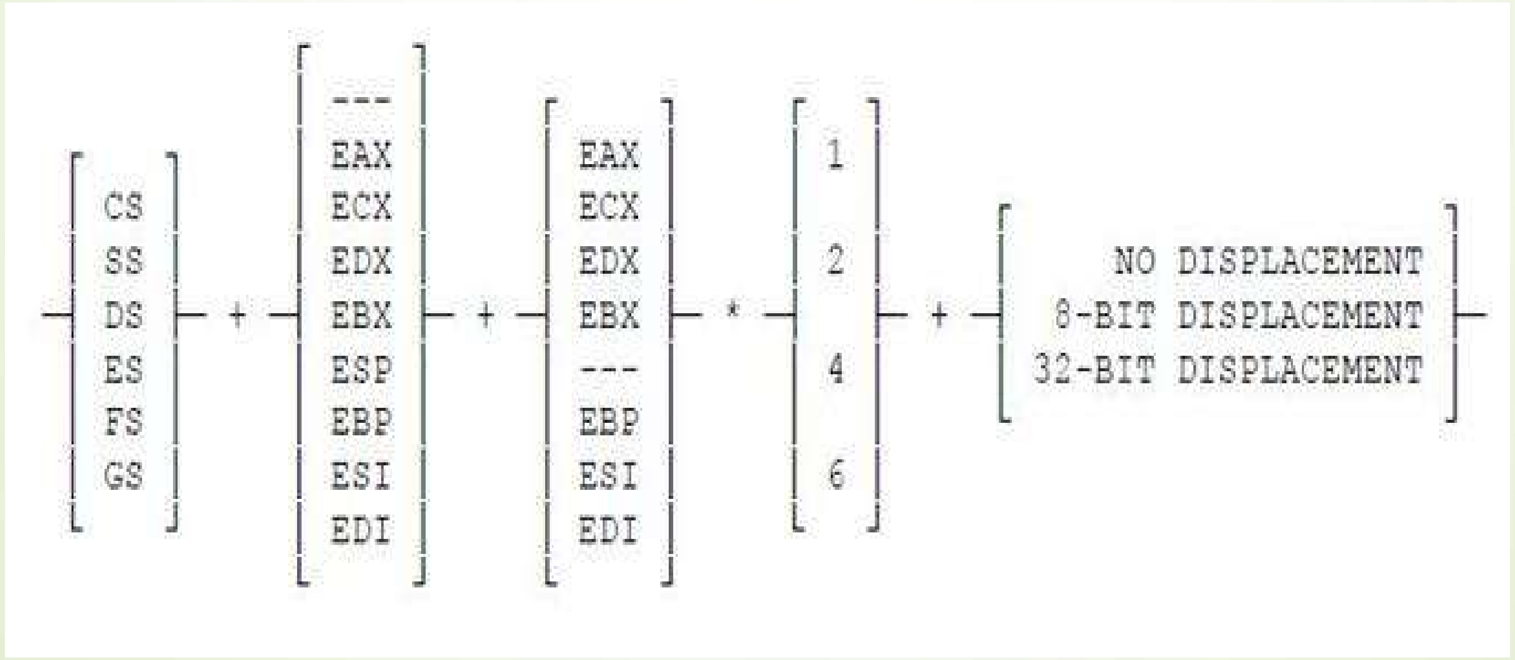
Based Index addressing Mode with Displacement

- The contents of an INDEX Register and a BASE register's contents and a DISPLACEMENT are all summed together to form the operand offset.
- Example: ADD EDX, [ESI] [EBP+00FFFFF0H]



Based Scaled Index addressing Mode with Displacement

- The contents of an INDEX register are multiplied by a SCALING factor, the result is added to the contents of a BASE register and a DISPLACEMENT to form the operand's offset.
- EXAMPLE: MOV EAX, LIST [EDI*4] [EBP+80]


CS
SS
DS
ES
FS
GS
+

EAX
ECX
EDX
EBX
ESP
EBP
ESI
EDI
+
EAX
ECX
EDX
EBX

EBP
ESI
EDI
*
1
2
4
6
+
NO DISPLACEMENT
8-BIT DISPLACEMENT
32-BIT DISPLACEMENT