For an instruction there are 3 ways to express a required operand:

* *register mode*, if the required operand is a register; mov ax, bx
* *immediate mode*, when we use directly the operand’s value (not its address and neither a register holding it); mov eax,2
* *memory addressing mode*, if the operand is located somewhere in memory. In this case, its offset is computed using the following formula:

***offset\_address* = [ base] + [ index × scale ] + [ constant]**

So *offset\_address* is obtained from the following (maximum) four elements:

* the content of one of the registers EAX, EBX, ECX, EDX, EBP, ESI, EDI or ESP as base;
* the content of one of the registers EAX, EBX, ECX, EDX, EBP, ESI or EDI as index;
* scale to multiply the value of the index register with 1, 2, 4 or 8;
* the value of a numeric constant, on a byte or on a doubleword.

From here results the following modes to address the memory:

* *direct addressing*, when only the *constant* is present;
* *based addressing*, if in the computing one of the base registers is present;
* *scale-indexed addressing*, if in the computing one of the index registers is present.

These three mode of addressing could be combined. For example, it can be present direct based addressing, based addressing and scaled-indexed etc.

A NOT direct addressing mosde is named INDIRECT addressing (based and/or indexed). So, an indirect addressing is that for which we have at least one register specified in square brackets ([]).

In the addressing system operations with pointers are performed. Which are the ARITHMETIC operations allowed with pointers in COMPUTER SCIENCE ?...

**Answer**: Any operation that makes sense... meaning any operation that expresses as a result a correct location in memory useful as an information for the programmer.

Pointer arithmetic...?

Adress – adress = ok (q-p = subtraction of 2 pointers = sizeof(array))

Adress + numerical constant (identification of an element by indexing – a[7]) , q+9

Adress - numerical constant - a[-4] , p-7

* subtraction of 2 pointers
* adding a constant to a pointer
* subtracting a constant from a pointer

POINTER ARITHMETIC OPERATIONS

* subtraction of 2 addresess – ok, is allowed, q-p = the number of bytes between those 2 addresses... !!!!
* adding a CONSTANT (INTEGER) to an address – a[7] = \*(a+7)
* subtracting a CONSTANT (INTEGER) from an address – a[-4] =

= \*(a-4) - useful for reffering array elements

p+q = ???? (allowed in NASM...sometimes...) – but it doesn’t mean in the end as we shall see a pointer addition

Mov op\_size\_dest, op\_SAME\_size b,b w,w dw,dw

Mov ax, ebx - syntax error ! “invalid combination of opcode and operands”

Mov ebx, ch - syntax error ! “invalid combination of opcode and operands”

Mov eax, ebx - eax ß the contents of EBX

Mov eax, [ebx] = mov eax, [ds:ebx] ; eax = the doubleword value from memory starting at the address DS:EBX

Mov ax, [ebx] = mov ax, [ds:ebx] ; ax = the word value from memory starting at the address DS:EBX

Mov edx, [eax+ebx] – EDX := the doubleword value from memory starting at the address [DS:EAX+EBX]

Mov edx, eax+ebx ; SYNTAX ERROR !! – see the diff. between the OPERATOR + and THE INSTRUCTION ADD !!!

Mov edx, [ebx+eax] – EDX := the doubleword value from memory starting at the address [DS:EAX+EBX]

Mov edx, [esp+ecx] ; EDX := the doubleword value from memory starting at the address [SS:ESP+ECX]

Mov edx, [ecx+esp]; - same effect as above ESP – BASE register