Pointer arithmetic

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/processor.

- adding a constant value to a pointer a[7] = *(a+7) – useful for going into memory forth and back relative to a starting address
- subtracting ..... a[-4] , a(-4) ...
- multiplying 2 pointers ? – No way ... no practical usage !
- dividing 2 pointers ? - No way ... no practical usage !
- adding/subtracting 2 pointers ?
- ADDING 2 pointers doesn't make sense !! – it is not allowed
- SUBTRACTING 2 pointers !! does makes sense... q-p = nr. of elements (in C) = nr. of bytes between these 2 addresses in assembly (this can be very useful for determine the length of a memory area).

a[7] = *(a+7) = *(7+a) = 7[a]   - both in C and assembly !

POINTER ARITHMETIC OPERATIONS - ***Pointer arithmetic*** represents the set of arithmetic operations allowed to be performed with pointers, this meaning using arithmetic expressions which have addresses as operands.

Pointer arithmetic contains ONLY 3 operations that are possible:

1). Subtracting two addresses
Address – address = ok   (q-p = subtraction of 2 pointers = sizeof(array) in C, the number of bytes between these 2 addresses in assembly)

2). Adding a numerical constant to a pointer
Address + numerical constant (identification of an element by indexing – a[7]) , q+9

3). Subtracting a numerical constant from a pointer
    Adress - numerical constant   - a[-4] , p-7;
    *(a-4)   - useful for reffering array elements

- subtraction of 2 pointers = SCALAR VALUE (constant)
- adding a constant to a pointer → a POINTER !!
- subtracting a constant from a pointer → a POINTER !!

ADDING TWO POINTERS IS NOT ALLOWED !!!

p+q = ???? (allowed in NASM...sometimes...) – but it doesn't mean in the end as we shall see that this is "a pointer addition" !!!

How do we make in NASM the difference between the address of a variable and its contents ?

Var – invoked like that it is an address (offset) ; [var] – is its contents
[] = the dereferencing operator !! (like *p in C)

V db 17
add edx, [EBX+ECX*2 +   v   -7] – OK !!!!

mov ebx, [EBX+ECX*2 - v-7] – Syntax error !!!! invalid effective address – impossible segment base multiplier

mov [EBX+ECX*2 + a+b-7], bx - not allowed ! syntax error ! because of "a+b" invalid effective address – impossible segment base multiplier

sub [EBX+ECX*2 + a-b-7], eax – ok, because a-b is a correct pointers operation !!!

```
[EBX+ECX*2 +   v   -7] – ok
        SIB        depl.  const.
```

```
[EBX+ECX*2 + a-b-7]
        SIB          const.
```

mov eax, [EBX+ECX*2+(-7)] – ok.