Aritmetica de pointeri

In cadrul sistemului de adresare se efectueaza operatii cu adrese (pointeri). Care sunt operatiile ARITMETICE cu pointeri permise IN INFORMATICA ?...

Raspuns: Orice operaţie care are sens... aceasta însemnând orice operaţie ce exprimă ca rezultat o localizare în memorie corectă şi utilă ca informaţie pt programator.

Aritmetica de pointeri/adrese (pointer arithmetic – operaţii ARITMETICE cu pointeri) = utilizarea de expresii aritmetice, care au ca operanzi adrese !

- adunări şi scăderi de adrese ?
  - Adunare de adrese = ??? CE reprezinta ? NIMIC !!!!!
  - Scădere de adrese = ??? CE reprezinta ? q-p = nr octeţi dintre cele două adrese de memorie (niciodată nu depăşim dim memoriei ; valoarea obţinută este o CONSTANTA NUMERICA !!!!)

- adunări şi scăderi de constante la o/dintr-o adresă = necesare şi utile pt accesarea elementelor dintr-un array

- înmulţirea a două adrese ?- nepermisă (in majoritatea cazurilor valoarea obtinuta este dincolo de limita maxima a memoriei posibile a fi accesata).
- Inmultirea cu o constanta (in majoritatea cazurilor valoarea obtinuta este dincolo de limita maxima a memoriei posibile a fi accesata). In plus, CE reprezinta valoarea obtinuta ?... Nimic util !!
- Impărţire ?... No way !

  - Singura exceptie de la regulile aritmeticii de pointeri o constituie formula de calcul a offsetului unui operand unde sunt permise adunări de valori de regiştri (NU adunări de pointeri!!)... In rest nu exista excepţii

a[7] = *(a+7) = *(7+a) = 7[a] - atât în C cât şi în asamblare !

Pointer arithmetic...? DOAR 3 operaţii sunt permise cu POINTERI:

1). Scăderea a doua adrese
Adresa – adresa = ok (q-p = scadere de 2 pointeri = sizeof(array) sau nr de elemente (in C)/octeti (asamblare) dintre doua adrese de memorie)

2) Adunarea unei constante numerice la o adresă
Adresa + constanta numerica (identificarea unui element prin indexare – a[7]) , q+9

3). Scăderea unei constante numerice dintr-o adresă
Adresa – constanta numerica - a[-4] , p-7

- scăderea a 2 pointeri – valoare SCALARA !!! (valoare numerică constantă imediată)
- adunarea unei constante la un pointer → POINTER !
- scăderea unei constante dintr-un pointer → POINTER !

(ultimele două sunt utile pt referirea de elemente dintr-un array/zonă de memorie)

ADUNAREA A DOI POINTERI NU ESTE PERMISA !!!!!!

p+q = ???? (allowed in NASM...sometimes...!!!!!!!) – dar nu inseamna ADUNARE DE POINTERI in cele din urma asa cum vom vedea...

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

adc ecx, [EBX+ECX*2 + a+b-7] – Syntax error din cauza “a+b”; invalid effective address – impossible segment base multiplier

sub [EBX+ECX*2 + a-b-7], eax – ok, pt că a-b este o operaţie corectă cu pointeri !!!

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

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

mov eax, [EBX+ECX*2+(-7)] – ok.

L-value; R-value. Valoare stangă vs. valoare dreaptă a unei atribuiri.

Atribuire: i:=i+1 LHS vs. RHS

(Adresa lui I ← valoarea lui I + 1)

LHS(i) = adresa lui I := RHS(i) = (continutul de la adresa I) + 1
LHS (valoare stanga a unei atribuiri este o L-value = adresa) := RHS (valoarea dreapta a unei atribuiri = R-Value = CONTINUT !!)

Sintaxele majorităţii limbajelor de programare prevăd că:
Symbol := expression_value, adică Identificator := expresie

In fapt, sunt limbaje (C++, ASAMBLARE !!!) care permit mai general sintaxa:
Expresie_calcul_de_adresa := expresie
(mov [ebx+2*EDX+v-7], a+2)

Dereferenţierea (extragerea valorii de la o adresa) este implicita in 99% din limbaje. Exemplu exceptie – limbajul BLISS – unde dereferentierea trebuie mentionata explicit intotdeauna; i←*i+1
(+ unele situatii in Algol 68)

Symbol := expression_value (99% of the cases…)

Address_computation_Expression := expression_value
In C++ f(a+3, b-2, 2) = x+y+z

Variabilele “referinţă C++” (C++ reference variables) au 3 utilizări:

1) Int& j = i; // j devine ALIAS pt i
2) Transmiterea de variabile prin referinţă la apelul de subprograme
   float f(int&x, y);…..
3) Returnarea de L-valori prin intermediul funcţiilor

Int& f(x,i) {….return v[i];} – Funcţia f returnează o LHS (valoare stangă)
F(a,7) = 79; înseamnă că v[7]=79 !!!

De asemenea, separat de acestea se permite şi utilizarea operatorului condiţional ternar pe post de valoare stângă:

(a+2?b:c) = x+y+z ; - correct
(a+2?1:c) = x+y+z; - syntax error !!! 1:=n !!!!
