

Constante de tip string. Reprezentare in memorie si utilizare in cadrul unor instructiuni de transfer.

In cazul initializarii unei zone de memorie cu valori de tip constante string (sizeof > 1) tipul de data utilizat in definire (dw, dd, dq) are rol doar de rezervare a spatiului dorit, **ordinea de "umplere" a zonei de memorie respective fiind ordinea in care apar caracterele (octetii) in cadrul constantei de tip string:**

a6 dd '123', '345','abcd' ; se vor defini 3 dublucuvinte continutul lor fiind
31 32 33 00|33 34 35 00|61 62 63 64|

a6 dd '1234' ; 31 32 33 34

a6 dd '12345' , 'abc'; 31 32 33 34|35 00 00 00| 61 62 63 00|

a7 dw '23','45' | 32 33 | 34 35| - 2 cuvinte = 1 doubleword

a7 dw '2345' - 2 cuvinte - 32 33|34 35|

a7 dw '23456' - 3 cuvinte - 32 33|34 35|36 00|

In C 'x'(1 byte) "x" (2 bytes = 'x' '\0')

In ASAMBLARE **'...' = "..."**

a8 dw '1', '2', '3' - 3 cuvinte - 31 00|32 00|33 00

a9 dw '123' - 2 cuvinte - 31 32|33 00

Urmatoarele definitii produc aceeasi configuratie de memorie

dd 'ninechars' ; constanta string doubleword

dd 'nine','char','s' ; 3 dublucuvinte

db 'ninechars',0,0,0 ; "umplere" zona prin secventa de octeti

Definitia din documentatia oficiala spune:

A character constant with more than one byte will be arranged (WHERE ???) with little-endian order in mind: if you code

```
mov eax, 'abcd' (EAX = 0x64636261)
```

then the constant generated is not 0x61626364, but 0x64636261, so that if you were then to store the value into memory, it would read `abcd` rather than `dcba`. This is also the sense of character constants understood by the Pentium's CPUID instruction.

Esenta acestei definitii este ca VALOAREA asociata unei constante de tip string 'abcd' este de fapt 'dcba' (adica acesta este modul de STOCARE al acestei constante în TABELA DE CONSTANTE)

Mov dword [a], '2345' va aparea in OllyDbg astfel:
mov dword ptr DS:[401000],35343332

iar in memoria rezervata pt a va aparea |32 33 34 35|

....dar daca folosim a data definition like a7 dd '2345' the corresponding memory layout will be NO little-endian representation, but |32 33 34 35|

Astfel, comparativ si in rezumat (constante tip string vs. constante numerice) :

```
a7 dd '2345'      ; |32 33 34 35|
a8 dd 12345678h   ; |78 56 34 12|
```

.....

```
mov eax, '2345' → EAX = '5432' = 35 34 33 32
mov ebx, [a7]   → EBX = '5432' = 35 34 33 32
```

```
mov ecx, 12345678h ; ECX = 12345678h (sizeof <12345678h> = 4 bytes)
mov dword [var1], '12345678' ; var1 = '1234' (31 32 33 34)
      ( sizeof '12345678' = 8 bytes)
```

```
mov edx, [a8] → EDX = 12345678h
```

În cazul în care se folosește DB ca directivă de definire a datelor e normal ca ordinea octetilor data în specificarea constantei să se regasească și în memorie în mod similar, deci acest caz nu comportă analiză și discuții suplimentare.

```
a66 TIMES 4 db '13' ; 31 33 31 33 31 33 31 33 echiv cu ...db '1','3'
```

```
a67 TIMES 4 dw '13' ; 31 33 31 33 31 33 31 33
```

 - cele două moduri de definire diferite
produc același rezultat !!!

```
a68 TIMES 4 dw '1','3' ; 31 00 | 33 00 | 31 00 | 33 00 | 31 00 | 33 00 | 31 00 | 33 00
```

```
a69 TIMES 4 dd '13' ; 31 33 00 00 | 31 33 00 00 | 31 33 00 00 | 31 33 00 00
```

Deci, un șir constant în NASM se comportă ca și cum ar exista o „zonă de memorie” alocată anterior acestor constante (ea există !! și se numește TABELA DE CONSTANTE!!), unde acestea sunt stocate folosind reprezentarea little-endian !!

Dpdv al REPREZENTĂRII valoarea asociată unei constant de tip string este INVERSA SA !!!! (vezi și “definiția oficială” de mai sus!).

Practic, dacă inițializăm o zonă de memorie cu o constantă de tip string (fie prin directive de definire a datelor, fie prin `mov [zona_de_memorie], constanta_string`) ordinea în care caracterele vor fi depuse în memorie este ordinea în care ele apar în scrierea pe hârtie a constantei de tip string !!!

Dacă inițializăm conținutul unui registru cu o constantă de tip string, caracterele se vor depune în ordine inversă apariției lor în scrierea pe hârtie a constantei de tip string !

From a **NUMERIC VALUE** point of view ??? (true both in C and assembler ???)

“abcd” = THE ADDRESS FROM MEMORY OF THIS CONSTANT (in C)

“abcd”[1] = ‘b’

“abcd”[251] = ??