Matricas un steka izmantošana

Divdimensiju masīvs *C++* valodā:

```
const int N = 2, M = 3;
int Matrix[N][M] = { {1, 2, 3}, {4, 5, 6} };
```

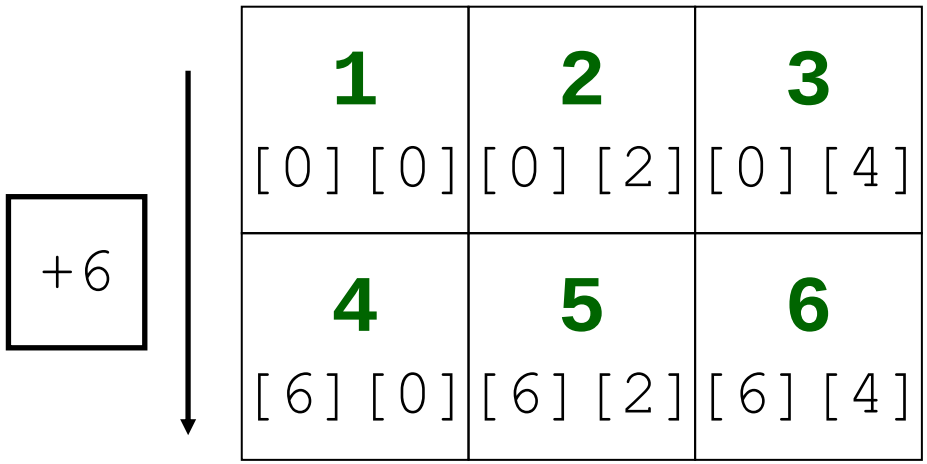
Matricas elementu adresēšana:

1 [0][0]	**2** [0][1]	**3** [0][2]
4 [1][0]	**5** [1][1]	**6** [1][2]

"Divdimensiju" masīvs *Assembler* valodā:

```
N           Equ     2
M           Equ     3
Matrix      DW      1, 2, 3
            DW      4, 5, 6
S           Equ     Type Matrix
```

Matricas elementu adresēšana:




```
6 = M*S = 3*2
```

C++ valodā:

`Matrix[i][j]` un `Matrix[j][i]` **nav** viens un tāds pats elements.

Assembler valodā:

`Matrix[i][j]` un `Matrix[j][i]` **ir** viens un tāds pats elements.

Assembler valodā:

```
Matrix[0][2] = Matrix[2][0] = Matrix[1][1]
```

Iespējama adresēšana (reģistru kārtību var izmainīt):

```
Matrix[Bx][Si]   Matrix[Bx][Di]
Matrix[Bp][Si]   Matrix[Bp][Di]
```

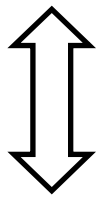
Kļūdaina adresēšana (reģistru kārtību var izmainīt):

```
Matrix[Bx][Bp]   Matrix[Si][Di]
```

Informācijas saglabāšana stekā:

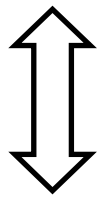
```
Mov   Cx, N
Push  Cx              ; SP = SP-2
...
;Darbs ar Cx reģistru
...
Pop   Cx              ; SP = SP+2
```

```
Push Ax Bx Cx
```



```
Push Ax
Push Bx
Push Cx
```

```
Pop Cx Bx Ax
```



```
Pop Cx
Pop Bx
Pop Ax
```

Ieliktie cikli:

```
        Mov   Cx, N
Rows:
        Push  Cx
        Mov   Cx, M
Cols:
        ...
        Loop  Cols
        ...
        Pop   Cx
        Loop  Rows
```

Alternatīvais risinājums – saglabāt skaitītāju
a. Citā *reģistrā*.
b. *Atmiņas* šūnā.

Elementu izmēra *neskaidrība*:

```
Min   Equ   2
...
Cmp   [Bx][Si], Min ;nav ieteicams!
```

Rezultāts: kompilatora brīdinājums

```
*Warning* l4_com_2.ASM(25)
Argument needs type override
```

Pareizs risinājums:

```
Min   Equ   2
...
Cmp   [Bx][Si], Word Ptr Min
```

Lai masīvā ir *baiti*, nevis *vārdi*:

```
Cmp       [Bx][Si], Byte Ptr Min
```