Arhitectura calculatoarelor si sisteme de operare
Laborator 11

Vitel Silviu-Constantin

Table of Contents

Accesarea tablourilor bidimensionale alocate static

- consideram o declaratie de tipul **int m[10][10];**
- elementele acestei matrice sunt plasate liniar in memorie, in ordinea precizata de linii si coloane
  - mai intai elementele liniei 0, apoi elementele liniei 1, etc.

m[0][0]	m[0][1]	...	m[0][9]	m[1][0]	m[1][1]	...	m[9][9]

- in acest caz, adresa unui element **m[i][j]** poate fi calculata prin **&m[0][0] + (i * 10 + j) * 4**
  - &m[0][0] este adresa primului element al matricei
  - 10 este numarul de coloane
  - 4 este dimensiunea unui element din matrice (matricea are elemente de tip int)
- formula generala este **&m[0][0] + (i * nr_coloane_matrice + j) * sizeof(tipul de date al elementelor)**

Accesarea tablourilor bidimensionale alocate static

- la fel ca in cazul tablourilor unidimensionale, adresa la care se afla elementele matricei poate fi incarcata intr-un registru prin intermediul instructiunii **lea**
- consideram ca indicele liniei (i in exemplul de mai sus) este retinut in registrul EBX, iar indicele coloanei (j in exemplul de mai sus) se afla in registrul ECX
- copierea valorii m[i][j] in registrul EDX se face in felul urmator:

```
lea edi, m                      //edi = &m[0][0]
mov eax, 10                     //eax = 10
mul ebx                         //eax = eax * ebx = 10 * i
add eax, ecx                    //eax = eax + ecx = (10 * i + j)
mov edx, [edi + eax * 4]        //edx = [edi + eax * 4] = m[i][j]
```

Accesare tablou bidimensional declarat static dat ca parametru unei functii

- consideram o functie **void f(int m[][10]);** care primeste ca parametru o matrice declarata prin **int m[10][10];**
- parametrul nu reprezinta intreaga matrice, ci este doar un pointer catre inceputul structurii
- structura matricei nu se schimba
- consideram ca indicele liniei (i in exemplul de mai sus) este retinut in registrul EBX, iar indicele coloanei (j in exemplul de mai sus) se afla in registrul ECX
- copierea valorii m[i][j] in registrul EDX se face in felul urmator:

```
mov edi, [ebp + 8]          //edi = &m[0][0]
mov eax, 10                 //eax = 10
mul ebx                     //eax = eax * ebx = 10 * i
add eax, ecx                //eax = eax + ecx = (10 * i + j)
mov edx, [edi + eax * 4]    //edx = [edi + eax * 4] = m[i][j]
```

Accesare tablou bidimensional declarat dinamic

- consideram o declaratie de tipul:

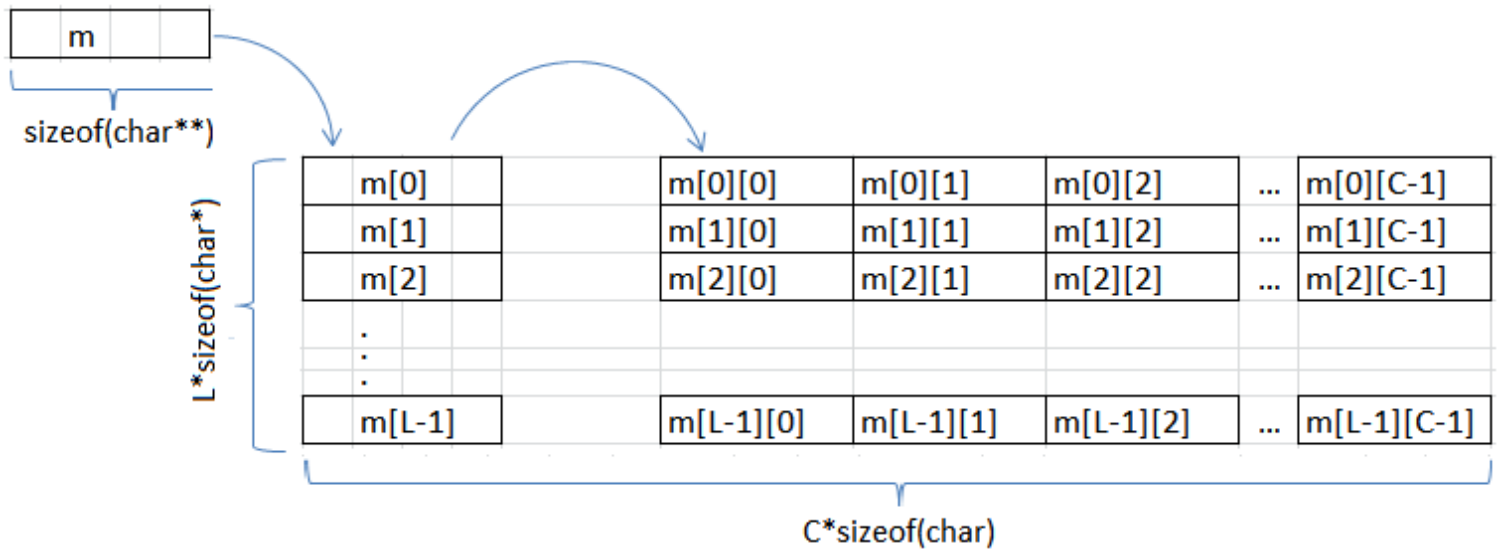
```
char **m;
char nrLinii = 10, nrColoane = 10;

//echivalent cu m = new char*[10];
m = (char**)malloc(nrLinii * sizeof(char*));

for (int i = 0; i < 10; i++)
  //echivalent cu m[i] = new char[10];
  m[i] = (char*)malloc(nrColoane * sizeof(char));
```

Accesare tablou bidimensional declarat dinamic

- in acest caz, structura matricei in memorie este diferita de cea prezentata anterior
- in memorie se va regasi un tablou (vector) de pointeri cu 10 elemente (a carui adresa este retinuta in variabila m) si 10 tablouri de numere intregi cu cate 10 elemente (adresele tablourilor sunt retinute in variabilele m[0], m[1], etc.)



Accesare tablou bidimensional declarat dinamic

- pentru a accesa elementul m[i][j] se procedeaza astfel
  - folosind adresa m, se obtine m[i] - adresa de inceput a tabloului corespunzator liniei i
  - cu ajutorul adresei m[i] se acceseaza m[i][j]
- consideram ca indicele liniei (i) este retinut in registrul EBX, iar indicele coloanei (j) se afla in registrul ECX
- copierea valorii m[i][j] in registrul AL se face in felul urmator:

```
mov edi, m
mov eax, [edi + ebx * 4]      //eax = m[i];
mov al, [eax + ecx]          //al = m[i][j];
```

- Observatii:
  - pentru ca pointerii sunt valori pe 4 octeti, inmultim ebx cu 4
  - valorile de tip char sunt retinute pe 1 octet, deci nu mai inmultim ecx cu 4

Accesare tablou bidimensional declarat dinamic dat ca parametru unei functii

- consideram o functie **void f(char** m);**
- consideram ca indicele liniei (i in exemplul de mai sus) este retinut in registrul EBX, iar indicele coloanei (j in exemplul de mai sus) se afla in registrul ECX
- copierea valorii m[i][j] in registrul AL se face in felul urmator:

```
mov edi, [ebp + 8]
mov eax, [edi + ebx * 4]
mov al, [eax + ecx]
```

Table of Contents

- o structura este o zona de memorie ce contine tipuri diferite de date
- pentru a accesa un anumit membru al structurii, sunt necesare doua elemente:
  - adresa de inceput a structurii (o putem accesa cu instructiunea **lea**)
  - deplasamentul membrului in cadrul structurii
- **un membru se regaseste in structura la o adresa de memorie ce este multiplul dimensiunii tipului sau de date**
  - consecinta: dimensiunea unei structuri este multiplu de cel mai mare camp din structura

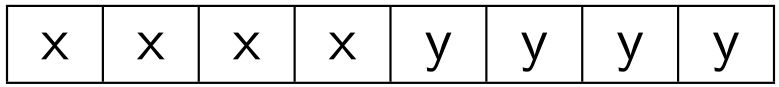
- consideram urmatoarea structura:

```
struct Point
{
  int x, y;
};
Point p;
```

- elementele acestei structuri se acceseaza in urmatorul mod:

```
lea edi, p
mov eax, [edi]          //eax = p.x
mov ebx, [edi + 4]      //ebx = p.y
```

- dimensiunea structurii este de 8 octeti
- alinierea membrilor in structura este urmatoarea:



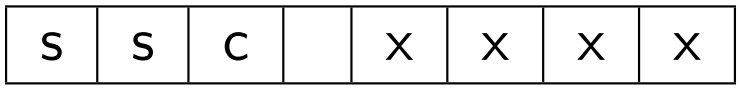
- consideram urmatoarea structura:

```
struct Info
{
   short s;
   char c;
   int x;
};
Info inf;
```

- elementele acestei structuri se acceseaza in urmatorul mod:

```
lea edi, inf
mov ax, [edi]          //ax = inf.s
mov bl, [edi + 2]      //bl = inf.c
mov edx, [edi + 4]     //edx = inf.x
```

- dimensiunea structurii este de 8 octeti
- alinierea membrilor in structura este urmatoarea:



- consideram urmatoarele structuri:

```
struct Info2
{
   int a;
   short b;
};

struct Info
{
   short s;
   Info2 inf2;
   char c;
   int x;
};

Info inf;
```

- elementele structurii inf se acceseaza in urmatorul mod:

```
lea edi, inf
mov ax, [edi]              //ax = inf.s
mov eax, [edi + 4]         //eax = inf.inf2.a
mov bx, [edi + 8]          //bx = inf.inf2.b
mov bl, [edi + 12]         //bl = inf.c
mov ecx, [edi + 16]        //ecx = inf.x
```

- dimensiunea structurii inf este de 20 octeti, dimensiunea structurii inf2 este de 8 octeti
- alinierea membrilor in structura este urmatoarea:

s	s			a	a	a	a	b	b			c				x	x	x	x