Arrays in Compilers

int A[5] = { 3, 5, 8, 4, 2 }

	0	1	2	3	4
A	3	5	8	4	2
	200/1	2/3	204/5	6/7	208/9

int x = 10;

x: 10 — 100/101

~~Compiler x diye bir değişken~~

Compiler x ismini makine koduna çevirmez x'in bellekte bulunduğu adresi çevirir. Bu tüm kompiler için geçerlidir.

Örnek olarak 3. indexteki ifadeyi değiştirmek isteyelim ve 6

	0	1	2	3	4
A	3	5	8	4	2
	200/1 Lo	2/3	4/5	6/7	8/9

A[3] = 10;

Add(A[3]) = 200 + 3x2 = 206

Add(A[3]) = Lo + 3*2 = 206

$\boxed{Add(A[3]) = Lo + i \times w}$

Lo = dizinin başlangıç adresi

i = index numarası

w = data size (int=2, float=4, char=1)

eğer index 1 ile başlarsa (arka sayfa)

	1	2	3	4	5
A	3	5	8	4	2
	200/1	2/3	4/5	6/7	8/9

10

$A[3] = 10;$

$Add(A[3]) = 200 + (3-1)*2 = 204$

$Add(A[i]) = \underbrace{L_0 + (i-1)*w}$

Burada 1 farkından ifade bu da program dilde yararlılık kater.

Row Major Formula for 2D Arrays

int A[3][4];

m n

A

a_{00}	a_{01}	a_{02}	a_{03}	a_{10}	a_{11}	a_{12}	a_{13}	a_{20}	a_{21}	a_{22}	a_{23}
200/1	2/3	4	6	8	10	12	14	16	18	20	22/23

Lo

row$_0$ row$_1$ row$_2$

A	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}

Bu görsel ifade gerçekte bellekte böyle yer tutmaz

$$Add(A[1][2]) = 200 + [4+2]*2 = 212$$

$$Add(A[2][3]) = 200 + [2*4+3]*2 = 222$$

$$Add(A[i][j]) = Lo + [i*n+j]*w \rightarrow 5$$

Index 1'den başlarsa

$$Add(A[i][j]) = Lo + [(i-1)*n+(j-1)]*w \rightarrow 7$$

Fazladan 2 tane aritmetik ifade var bu da programın daha yavaş çalışmasına neden olur.

Column Major formula for 2D Arrays

int A[3][4]

m n

A	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}

0	1	2	3	4	5	6	7	8	9	10	11
a_{00}	a_{10}	a_{20}	a_{01}	a_{11}	a_{21}	a_{02}	a_{12}	a_{22}	a_{03}	a_{13}	a_{23}
200/1	2	4	6	8	10	12	14	16	18	20	222/23
L_0											
col 0			col 1			col 2			col 3		

$$Add(A[1][2]) = 200 + [2*3+1]*2 = 214$$

$$Add(A[1][3]) = 200 + [3*3+1]*2 = 220$$

$$Add(A[i][j]) = L_0 + [j*m+i]*w \rightarrow \text{column}$$

~~soldan~~ sağdan sola

$$Add(A[i][j]) = L_0 + [i*n+j]*w \rightarrow \text{row}$$

Soldan sağa

her ikisinde aynı hızda çalışır

bu yüzden her ikisinde seçebiliriz

formulas for nD Arrays

PERA

Type $A[d_1][d_2][d_3][d_4]$;

row-majority

$$Addr(A[i_1][i_2][i_3][i_4]) = L_0 + [i_1 * d_2 * d_3 * d_4 + i_2 * d_3 * d_4 + i_3 * d_4 + i_4] * w$$

genel formül = $L_0 + \sum_{p=1}^{n} \left[i_p * \prod_{q=p+1}^{n} d_q \right] * w$

column-majority

$$Addr(A[i_1][i_2][i_3][i_4]) = L_0 + [i_4 * d_3 * d_2 * d_1 + i_3 * d_2 * d_1 + i_2 * d_1 + i_1] * w$$

bunu detaylı inceleyelim

$$l_0 + [\underbrace{i_4 * d_3 * d_2 * d_1}_{3} + i_3 * \underbrace{d_2 * d_1}_{2} + i_2 * \underbrace{d_1}_{1} + i_1] * w$$

$4D \rightarrow 3+2+1$
$5D \rightarrow 4+3+2+1$
$nD \rightarrow n-1+n-2+ \dots +3+2+1 = \frac{n \cdot (n-2)}{2}$

$$\frac{n(n-1)}{2} = O(n^2)$$

↳ bu da çok zaman kaybı eder

arka sayfadan devam

raw-major

Type $A[d_1][d_2][d_3][d_4]$

$$Addr(A[i_1][i_2][i_3][i_4]) = L_0 + [i_1 \cdot d_2 \cdot d_3 \cdot d_4 + i_2 \cdot d_3 \cdot d_4 + i_3 \cdot d_4 + i_4] \cdot w$$

$$i_1 \cdot d_2 \cdot d_3 \cdot d_4 + i_2 \cdot d_3 \cdot d_4 + i_3 \cdot d_4 + i_4$$

tersten yazdık

$$i_4 + i_3 \cdot d_4 + i_2 \cdot d_3 \cdot d_4 + i_1 \cdot d_2 \cdot d_3 \cdot d_4$$

$$i_4 + d_4 * [i_3 + i_2 d_3 + i_1 \cdot d_2 \cdot d_3]$$

$$i_4 + d_4 * [i_3 + d_3 * [i_2 + i_1 * d_2]]$$

1 2 3

Horner's Rule

4D → 3
5D → 4
nD → n-1

O(n)
?

Bu yüzden row majority daha efektif

formulas for 3D Arrays

int A[l][m][n];

raw - major

Addr (A[i][j][k]) = Lo + [i.m.n + j.n + k] . w

column - major

Addr (A[i][j][k]) = lo + [k.l.m + j.l + i] . w

Scanned by TapScanner