AEX 21

5.1, 5.2.1 a 5.2.4

(0,0)	(0,1)	(0,2)	...
(1,0)	(1,1)	(1,2)	...
⋮	⋮	⋮	...

5.1)

→ C: elementos da mesma linha são guardados um ao lado do outro na cache

→ word: 32 b

1) Bloco: 16 bytes → $\frac{16}{4}$ = 4 inteiros
int: 4 bytes (32 bits)

2)
```
for (i = 0; i < 8; i++) {
    for (j = 0; j < 8000; j++) {
        A[i][j] = B[i][0] + A[j][i]
```

Localidade temporal: B[i][0], i (8000 vezes) e j ($64 \cdot 10^3$ vezes)

verificando (☆)

i = 0:

A[0][0] = B[0][0] + A[0][0]
A[0][1] = B[0][0] + A[1][0]
A[0][2] = B[0][0] + A[2][0]
⋮ ↓ constante para cada laço de j

→ localidade temporal

3) como a linguagem C guarda elementos da mesma linha contiguamente, haverá localidade espacial. Basta observar (☆). → A[i][j]

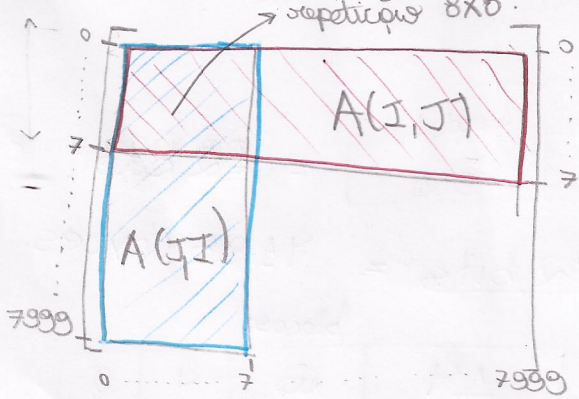
→ hit!

4) Cache 16-bytes por bloco. $A_{8000 \times 8000}$ $B_{8 \times 1}$.

$i \in [0,7]$ e $j \in [0,7999]$

A(I,J) ↘ A(J,I) ↘

A(0,0) até A(7,7999) A(0,0) até A(7999,7)



→ A(J,I) usa 8000 x 8 elementos
→ A(I,J) usa 8 x 8000 elementos
→ repete 8 x 8

$R = 64 \cdot 10^3 + 64 \cdot 10^3 - 64$

$R = 12736$ elementos

→ cada elemento tem 32 bits logo

$$\frac{12736 \cdot 4 \text{ (bytes)}}{16 \frac{\text{bytes}}{\text{bloco}}} = \frac{12736}{4} \text{ blocos} = 3184 \text{ "}$$

5) Matlab guarda desta forma na cache:

(0,0)	(1,0)	(2,0)	...
(0,1)	(0,2)	(0,3)	...
⋮	⋮	⋮	...

temporal: i, j e B(J,0)

verificando (*)

i=0

A(0,0) = B(0,0) + A(0,0)
A(0,1) = B(0,0) + A(1,0)
A(0,2) = B(0,0) + A(2,0)
⋮

6) A(J,I)

5 2) → word adress!

→ endereço é um número inteiro

3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186 e 253

1) · Mapeamento Direto
· 16 blocos · 1 palavra / bloco

Decimal

→ block adress: fica igual ao word adress porque será dividido por um.

→ índice:

3 mod 16 = 3 180 mod 16 = 4 43 mod 16 =

2^{10} bytes

4) • MD • 32 KiB • 1 ciclo • $\frac{2\,w}{bloco} = \frac{8\,bytes}{bloco}$

$$32\,KiB = \frac{2^5 \cdot 2^{10}\,bytes}{8\,\frac{bytes}{bloco}} = 2^{12}\,blocos.$$

→ tag = 17
índice = 12
Woff = 1
Boff = 2

Bloco:

válido	tag	w0	w1
1	17	32	32

$Total_{Cache_1} = 2^{12} \cdot 18 + 64 \cdot 2^{12} = 335872\,bits$

(2^{12} blocos; 18 w: v+tag; $64 \cdot 2^{12}$: 2^{12} blocos, words (bits))

Não tem mais onde mexer

2^a Cache: • MD • 16w/bloco → (8x) (2³) + words → -3 bits da tag

Bloco:

v	tag	w0	...	w15
1	14	32	...	32

tag	índice	word offset	Byte offset
14	log2 x	4	2

$Total_{Cache} = Dados + (tag + válido) \cdot blocos$

→ Dados = número de words · Tamanho (bits) da word · blocos

$Dados = 16 \cdot 32 \cdot x$

$Total_{Cache_2} = 512x + (14+1)x$

$Total_{C_2} = 527x$

$335872 \leq 527x$

$x \geq 637$ → potência de 2 → 2^{10}

2^{10} blocos
blocos.

Comparação:

→ blocos: Quanto menos blocos, mais concorrência por um índice na cache (↑ miss)

→ Tamanho: Caches maiores têm tempo de acesso maior.