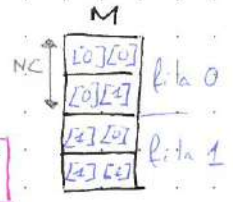
MATRIUS (T4)

```
mult rs, rt   //$hi:lo ← rs*rt
mflo rd   // rd ← $lo
mfhi rd   // rd ← $hi
```

```
      .align 2
mat:  .space  NF*NC*4
```

$$\boxed{@mat[i][j] = mat + (i*NC + j)*T}$$



Es pot fer directament "la $t3, mat + 5*4" Si sempre has d'accedir a mat[i][5].

// Has de desplegar l'eq. per veure quines operacions pots fer | mat + i*NC*T + j*T

Aquí si que és <u>important</u> <u>l'eficiència</u>.

K = mat[3][5] ≡ "la $t3, mat + 3*NC*4 + 5*4"; "lw $t0, $t3"

<u>STRIDE</u>: Quantitat constant que augmenta l'index. (Significa Pas) +4 +4 +4 STRIDE = 4.

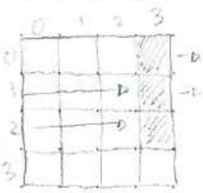
$$\begin{array}{lclcl} & @mi\_next & = \&array[i+1] & = & array + i + 1 \\ \ominus & @mi & = \ominus\&array[i] & \ominus & array + i \\ \hline & \underline{STRIDE} = & & & 1 \text{ element} \\ & & & & 1*T \Rightarrow \underline{4\ Bytes} \end{array}$$

Podem fer servir
$t3 = &array[last_value]
per controlar while (p ≤ $t3)
i no fer servir "$t0<10"

- Accés aleatori: for (int i=0; i<10; i++) V[i] = 0;
- Accés seqüencial: for (p = v; p < &v[num_elements]; p++) *p = 0;

Per exemple el STRIDE de mat[i][3] ⇒

$$\begin{array}{rl} & \cancel{mat} + i*\cancel{NC} + NC + \cancel{3} \\ \ominus & \cancel{mat} + i*\cancel{NC} \quad + \cancel{3} \\ \hline STRIDE: & NC \text{ elements} \\ & NC*T \text{ bytes} \end{array}$$



Pau Bru Ribes.

22

EC Examen de Problemes

Exercici 1 (Ex. Parcial 2013-2014 Q1)

Tradueix a llenguatge assemblador MIPS la subrutina func1:

```
short func2(short a, char *b, short *c);

short func1 (int x, short *y) {
  char V1[7];
  short V2[7];
  short res;
  res=func2(*y,&V1[x],V2);
  if (x>0)
    res++;
  return res+(*y);
}
```

s0 a0

s1 a1

$va

7B | V1[7] | 0(sp)
1B | vuit |
14 B | V2[7] | 8(sp)
2B | /// |
4B | s0 old 124 ($sp)
4B | s1 old 128(sp)
4B | ra 132 (sp)

36B

Exercici 2

Donades les següents declaracions en C (on N és una constant):

```
#define N 10

int A[N][N];

void func(int B[][N]) {
  int i, j; // a $t1 i $t2 respectivament.
    ... /* aquí va la sentència de cada apartat */
}
```

Tradueix a MIPS les següents sentències, suposant que pertanyen a la funció func:

a) `B[i][3] = 0;`

b) `B[i][j] = 0;`

c)
```
for (i=0; i<N; i++) //utilitza accés seqüencial
      B[3][i] = 0;
```

d)
```
for (i=0; i<N; i++) //utilitza accés seqüencial
       B[i][i] = 0;
```

e)
```
for (i=0; i<N; i++) //utilitza accés seqüencial
      B[i][N-1-i] = 0;
```

Exercici 3 (Ex. Final 2011-2012 Q2)

Considera el següent programa

```
int v[20],m[20][20];
main() {
     int i;
     for (i=19; i>=0; i--)
          v[i] = m[19-i][i];
}
```

Tradueix el programa principal a llenguatge assemblador MIPS. Només superaran aquesta pregunta aquelles solucions en què cada iteració del bucle tingui 7 o menys línies de codi.

Pau Bru Ribes

(Ex 1). Tradueix a MIPS.

```
addiu $sp, $sp, -36
sw    $s0, 24 ($sp)
sw    $s1, 28 ($sp)
sw    $ra, 32 ($sp)
move  $s0, $a0
move  $s1, $a1
lh    $a0, 0($s1)     # $a0 = *y
addu  $a1, $sp, $s0   # $a1 = &V1[x]
addiu $a1, $sp, 8     # $a2 = V2[0]
jal   func2
```

```
    ble $v0, $zero, fi   x ≤ 0 goto fi
    addiu $v0, $v0, 1
fi:
    addu $v0, $v0, $a0   Això ho faig pq. func2 no m' hauria tocat a0 (hauria fet sX).

    lw $s0, 24($sp)
    lw $s1, 28($sp)
    lw $ra, 32($sp)
    addiu $sp, $sp, 36. ☑
```

(Ex 2). Tradueix a MIPS. //A[][] i B[][] són de int.

a) B[i][3] = 0;

```
#B + i*NC*4 + 3*4

li $t0, B + 3*4
sll $t1, $t1, 2 # i*4
li $t2, N
mult $t1, $t2 # i*4*10
mflo $t2 # Agafar resultat
addu $t0, $t0, $t2
sw $zero, 0($t0) B[i][3]=0;
```

b) B[i][j] = 0; # &B + i*NC*T + j*T

```
la $t0, B
sll $t1, $t1, 2
li $t3, N
mult $t1, $t3
mflo $t1
addu $t0, $t0, $t1
sll $t2, $t2, 2
mult $t2, $t3
mflo $t2
addu $t0, $t0, $t2
sw $zero, 0($t0).
```

c) for (i=0; i<N; i++) B[3][i]=0;

```
li $t0, B+3*N*4 &B[3][0]
addiu $t2, $t0, N*4 last_val
While :
sw $zero, 0($t0)
addiu $t0, $t0, 4
blt $t0, $t2, while (p < &B[3][last]) goto "while"
```

(iN + N) + (i + 1)
(iN) + (i)

N + 1

44 B de STRIDE

d) for (i=0; i<N; i++) B[i][i] = 0;

```
la $t0, B &B[0][0]
li $t2, 44*N last_val
addiu $t2, $t0, $t2 &B[9][9]
While:
sw $zero, 0($t0)
addiu $t0, $t0, 44
blt $t0, $t2, while
```

□ Podria haver estalviat el "li"

□ Faig 44*N pq. hi ha 10 elements en la matriu principal

e) for (i=0; i<N; i++) B[i][N-1-i] = 0

```
la $t0, B
addiu $t3, $t0, 36 &B[0][9]
addiu $t2, $t0, (10*10-10)*4 B[9][0]
While:
sw $zero, 0($t3)
addiu $t3, $t3, 36
blt $t3, $t2, while
```

(iN - j)*T on i = N-1 pq és l'últ.

((N-1)N - j)*T = ((N*N - N) - j)*T

((10*10 - 10) - 0)*4

(90)*4 = 360

(Ex 3). Tradueix amb bucle menys 7 línies.

```
la $t1, v
la $t2, m
addiu $t0, $t2, 76 &m[0][19]
addiu $t1, $t1, 76 &v[19]
addiu $t3, $t2, 1520 &m[6][6]
While:
lw $t2, 0($t0) $t2 = m[0][19]
sw $t2, 0($t1) v[19] = $t2
addiu $t0, $t0, 76 m[i+][i-]
addiu $t1, $t1, -4; v[i-].
ble $t0, $t3, while &m[a][i] ≤ m[last][last] goto while
```

(iN + N) + (j)
Θ(iN) + (j + j)

N - j ⇒ 19 elements ≡ 76 B

Ec-4-E-1