

Tema 2. Instruccions i tipus de dades bàsics

Estructura de Computadors (EC)

Rubèn Tous

rtous@ac.upc.edu
Computer Architecture Department
Universitat Politècnica de Catalunya



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índex

1 2.11 Punters

- 2.11.1 Declaració
- 2.11.2 Inicialització
- 2.11.3 Operació desreferència (indirection)
- 2.11.4 Operació 'aritmètica de punters'
- 2.11.4 Relació entre punters i vectors

Punters

Punter

Un punter és una **variable** que conté una adreça de memòria. Per tant, en MIPS ocupa 32 bits. Si el punter p conté l'adreça de la variable v diem també que *p apunta a v* .

Punters

Com tota variable un punter pot ser variable **global** o **local**:

```
1 int *p1; // global
2
3 void main()
4 {
5     int *p2; // local
6     ...
7 }
```

Índex

1

2.11 Punters

- 2.11.1 Declaració
- 2.11.2 Inicialització
- 2.11.3 Operació desreferència (indirection)
- 2.11.4 Operació 'aritmètica de punters'
- 2.11.4 Relació entre punters i vectors

Declaració

Declaració (si global):

```
1  int *p1, *p2;  
2  char *p3;
```

```
1      .data  
2  p1 :.word 0  
3  p2 :.word 0  
4  p3 :.word 0
```

Si local no cal reservar espai, només decidir a quin registre anirà.

Índex

1

2.11 Punters

- 2.11.1 Declaració
- **2.11.2 Inicialització**
- 2.11.3 Operació desreferència (indirection)
- 2.11.4 Operació 'aritmètica de punters'
- 2.11.4 Relació entre punters i vectors

Inicialització

Operador adreça-de (&)

En C l'operador & davant una variable retorna l'**adreça de la** variable.

Inicialització

Inicialització dins la declaració (si global):

```
1 int v[100];  
2 char a='E', b='K';  
3 char *pglob = &a;
```

```
1 v:      .space 400  
2 a:      .byte 'E'  
3 b:      .byte 'K'  
4 pglob:.word a    # char *pglob = &a
```

Inicialització

Inicialització dins una sentència (si punter global):

```
1 void f ()  
2 {  
3     pglob = &b;  
4 }
```

```
1 f :  
2     la $t1, b    # &b  
3     la $t2, pglob # &pglob  
4     sw $t1, 0($t2) # pglob = &b  
5     ...
```

Inicialització

Inicialització dins una sentència (si punter local):

```
1 void f ()
2 {
3     int *ploc; /* en $t0 */
4     ploc = &b;
5 }
```

```
1 f:
2     la $t0, b # ploc = &b
3     ...
```

Índex

1

2.11 Punters

- 2.11.1 Declaració
- 2.11.2 Inicialització
- **2.11.3 Operació desreferència (indirection)**
- 2.11.4 Operació 'aritmètica de punters'
- 2.11.4 Relació entre punters i vectors

Operació desreferència (indirection)

Operador adreça-de (&)

En C l'operador * davant un punter retorna l'**el valor de la variable a la que apunta (l'adreça continguda en)** el punter.

Operació desreferència (indirection)

Exemple desreferència (si punter global):

```
1 char *pglob;  
2  
3 void g()  
4 {  
5     char tmp; // a $t0  
6     tmp = *pglob;  
7     ...  
8 }
```

```
1 g: ...  
2 la $t2, pglob # &pglob  
3 lw $t3, 0($t2) # pglob  
4 lb $t0, 0($t3) # *pglob  
5 ...
```

Operació desreferència (indirection)

(escriptura)

```
1 char *pglob;  
2  
3 void g()  
4 {  
5     *pglob = 3;  
6     ...  
7 }
```

```
1 g: ...  
2   la $t2, pglob    # &pglob  
3   lw $t3, 0($t2)  # pglob  
4   li $t4, 3  
5   sb $t4, 0($t3)  # *pglob = 3  
6   ...
```

Operació desreferència (indirection)

Exemple desreferència (si punter local):

```
1 void g()  
2 {  
3     char tmp;    //$t0  
4     char *ploc; //$t1  
5     ...  
6     tmp = *ploc;  
7     ...  
8 }
```

```
1 g:  
2     ...  
3     lb $t0, 0($t1)  
4     ...
```


Operació desreferència (indirection)

(escriptura)

```
1 void g()  
2 {  
3     char *ploc; // $t1  
4     ...  
5     *ploc = 3;  
6     ...  
7 }
```

```
1 g:  
2     ...  
3     li $t0, 3  
4     sb $t0, 0($t1)  
5     ...
```

2.11.3 Operació desreferència (indirection)

Operador de desreferència vs. declarador

No s'ha de confondre l'operador de desreferència `*` amb el declarador de punters explicat (mateix símbol).

```
1  int *p1;  
2  void g(int *p2)  
3  {  
4      int *p3;  
5      ...  
6      *p1 = *p2;  
7      ...  
8  }
```

Índex

1

2.11 Punters

- 2.11.1 Declaració
- 2.11.2 Inicialització
- 2.11.3 Operació desreferència (indirection)
- **2.11.4 Operació 'aritmètica de punters'**
- 2.11.4 Relació entre punters i vectors

Aritmètica de punters

Aritmètica de punters

Sumar un enter N a un punter p , que apunta a un tipus de T bytes, dóna com a resultat un altre punter $q = p + N * T$.

```
1 char *p1;  
2 int *p2;  
3 long long *p3;  
4 ...  
5 p1 = p1 + 3;  
6 p2 = p2 + 3;  
7 p3 = p3 + 3;
```

Aritmètica de punters

```
1 char *p1;           // $t1
2 int *p2;            // $t2
3 long long *p3;      // $t3
4 ...
5 p1 = p1 + 3;
6 p2 = p2 + 3;
7 p3 = p3 + 3;
```

```
1 addiu $t1, $t1, 3
2 addiu $t2, $t2, 12
3 addiu $t3, $t3, 24
```

Índex

1

2.11 Punters

- 2.11.1 Declaració
- 2.11.2 Inicialització
- 2.11.3 Operació desreferència (indirection)
- 2.11.4 Operació 'aritmètica de punters'
- 2.11.4 Relació entre punters i vectors

Relació entre punters i vectors

Aritmètica de punters

En C un vector és en realitat un punter que apunta al primer element.

```
1 int vec[100];  
2 int *p;
```

Relació entre punters i vectors

Aritmètica de punters

En C un vector és en realitat un punter que apunta al primer element.

```
1  int vec[100];
2  int *p;
3
4  main()
5  {
6      ...
7      p = vec; //OK
8      ...
9  }
```


Relació entre punters i vectors

L'expressió

```
1 *(p + i) = 0;
```

És equivalent a:

```
1 p[i] = 0;
```

Relació entre punters i vectors

```
1  int vec[100];
2  int *p;
3
4  main()
5  {
6      *v = 3;    /* v[1] = 3;
7      ...
8      p = vec;
9      p[8] = 10; /* element 8 del vector */
10     ...
11 }
```