

Tema 3. Traducció de Programes

Estructura de Computadors (EC)

Rubèn Tous

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



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índex

1 3.7 Subrutines

3.7 Subrutines

- Subprograma amb determinats paràmetres podent retornar un resultat, i permet ser invocada des de múltiples punts del programa.
- Acció/procediment, funció (C), mètode (Java).
- Disseny descendent, reutilització de codi, programació modular, etc.

3.7 Subrutines

- Subprograma amb determinats paràmetres podent retornar un resultat, i permet ser invocada des de múltiples punts del programa.
- Acció/procediment, funció (C), mètode (Java).
- Disseny descendent, reutilització de codi, programació modular, etc.

```
1  int main() {  
2      int x;  
3      x = suma(2, 3);  
4      x = x + 1;  
5  }  
6  int suma(int a, int b) {  
7      return a + b;  
8  }
```

3.7.1 Crida i retorn

Ho podem resoldre així?

```
main:
...
    b suma
ret:
...
suma:
...
    b ret
```

3.7.1 Crida i retorn

- Jump and link: `jal suma # $ra = ret, PC=suma`
- Jump register: `jr $ra # PC = $ra`

```
main:
...
    jal suma
ret:
...
suma:
    ...
    jr $ra
```

Application Binary Interface (ABI)

- Interfície comuna per a interactuar amb el sistema operatiu i amb biblioteques de funcions (libraries).
- Conjunt de REGLES.
- Per a cada ISA, SO i llenguatge d'alt nivell.

3.7.2 Paràmetres i resultats

REGLA 1: Pas de paràmetres i resultats

- PARÀMETRES: \$a0, \$a1, \$a2, \$a3 (floats a \$f12 i \$f14).
- En ordre.
- RESULTATS: \$v0 (\$f0 els floats).
- NOTA 1: Extendrem signe o zeros si tipus < 32 bits.
- NOTA 2: No permetrem altres casos (més pàrams, longs, doubles, structs, etc.)

3.7.2 Paràmetres i resultats

Exemple de pas de paràmetres i resultats:

```
1 int main() {  
2     int x;  
3     x = suma(2, 3);  
4     x = x + 1;  
5 }  
6 int suma(int a, int b) {  
7     return a + b;  
8 }
```

3.7.2 Paràmetres i resultats

Exemple de pas de paràmetres i resultats:

```
1 main :  
2     ...  
3     li      $a0, 2  
4     li      $a1, 3  
5     jal     suma  
6     #fem quelcom amb $v0  
7     ...  
8  
9 suma :  
10    addu    $v0, $a0, $a1  
11    jr      $ra
```

3.7.2 Paràmetres i resultats

Paràmetres de tipus vector o matriu:

En C:

```
f(int vec[10]) = f(int vec[]) = f(int *vec)
```

```
f(int mat[2][3]) = f(int mat[][3])  
(puc no passar numcols però no podré indexar)
```

3.7.2 Paràmetres i resultats

A quina declaració correspon cada crida?

```
short V1[10];  
char V2[20];  
int a;  
int *p;
```

- | | |
|---------------------|-----------------------|
| (1) a = f(V1); | (a) int f(int i); |
| (2) a = f(p); | (b) int f(char c); |
| (3) a = f(&V2[10]); | (c) int f(short *pi); |
| (4) a = f(V2[10]); | (d) int f(char vc[]); |
| (5) a = f(*p); | (e) int f(int vi[]); |

3.7.2 Paràmetres i resultats

Pas de paràmetres per valor i per referència:

- En llenguatge C sols existeix el pas de paràmetres per valor.
- Es pot recórrer als punters per aconseguir un resultat semblant al dels paràmetres per referència

3.7.2 Paràmetres i resultats

Exemple de pas de paràmetres amb vector:

```
1  int v1[10];  
2  int main() {  
3      init(v1);  
4  }  
5  int init(int v2[]) {  
6      int i;  
7      for(i = 0; i < 10; i++) {  
8          init[i] = 0;  
9      }  
10 }
```

3.7.2 Paràmetres i resultats

Exemple de pas de paràmetres amb vector:

```
1 main :  
2     ...  
3     la $a0, v1  
4     jal init  
5     ...  
6 init :  
7     li $t0, 0  
8     li $t1, 10  
9 for :  
10    bge $t0, $t1, ffor  
11    sll $t2, $t0, 2  
12    addu $t2, $t2, $a0  
13    sw $zero, 0($t2)  
14    addiu $t0, $t0, 1  
15    b for  
16 ffor :  
17    jr $ra
```

3.7.3 Les variables locals

REGLA 2: Variables locals

- Lliure elecció: \$t0..\$t9, \$s0..\$s7, \$v0..\$v1.
- Floats simple precissió a \$f0..\$f31.
- Excepcions (aniran a la pila):
 - Vectors i matrius.
 - Si la variable v apareix amb &v.
 - Si ens quedem sense registres.

3.7.3 Les variables locals

Exemple variables locals

```

1 void func(int p) {
2     int l = p;
3     ...
4     l++;
5     ...
6 }

```

```

1 func:
2     ...
3     addiu $a0, $a0, 1
4     ...
5     jr $ra

```

Puc fer servir \$a0 per emmagatzemar la variable local.

3.7.3 Les variables locals

Exemple variables locals

```
1  int func() {  
2      int a;  
3      a = 1;  
4      return a;  
5  }
```

```
1  func:  
2      li $v0, 1  
3      jr $ra
```

Puc fer servir \$v0 per emmagatzemar la variable local.

3.7.3 Les variables locals

La pila (stack) i el bloc d'activació (stack frame)

- Subrutines: Cal guardar a memòria i alliberar la darrera.
- Estructura LIFO (last in first out).
- Stack pointer $\$sp = 0x7FFFFFFFC$.
- No farem PUSH/POP. Reservarem espai al principi i alliberarem al final.

3.7.3 Les variables locals

Exemple ús de la pila:

```

1  int f() {
2      int i;
3      int vec[10];
4      i = vec[2];
5      return i;
6  }
```

```

1  f:
2      addiu $sp, $sp, -40
3      lw    $v0, 8($sp)
4      addiu $sp, $sp, 40
5      jr   $ra
```

3.7.3 Les variables locals

REGLA 3: Bloc d'activació

- Variables en ordre.
- Alineades com si estiguèssin al `.data`.
- Mida total múltiple de 4.

3.7.3 Les variables locals

Exemple variables locals:

```

1  int f1 () {
2      int i;
3      char a;
4      int b[10];
5      char c[5]
6      ...
7      ... &a ...
8      return i;
9  }
```

```

0 ($sp) |  a  | 1b
        |    | 3b
4 ($sp) |  b  | 40b
44 ($sp) |  c  | 5b
        |    | 3b
total = 52 bytes
```

7.4 Subrutines multinivell

Preservar el context del programa:

```
int f1(int a) {  
    int b = 10;  
    ---f2()---  
    return a+b;  
}
```

Perill en a, b i \$ra.

7.4 Subrutines multinivell

REGLA 4: Preservar el context

- Una subrutina ha de preservar els valors originals de \$s0-\$s7.
- Protegirem els valors que calgui en aquests registres (cridadors).
- Com a cridats, guardarem a la pila (després de les locals) els valors originals.
- Tot això ho farem al principi/fin de la subrutina, no on hi hagi la crida (if, bucle, etc.).

7.4 Subrutines multinivell

Exemple preservar context:

```

1 f1 :
2     addiu $sp, $sp, -12
3     sw     $s0, 0($sp)    # VALOR ANTIC $s0
4     sw     $s1, 4($sp)    # VALOR ANTIC $s1
5     sw     $ra, 8($sp)    # VALOR ANTIC $ra
6     move   $s0, $a0       # $s0 = VALOR A PRESERVAR
7     li     $s1, 10        # $s1 = VALOR A PRESERVAR
8     jal    f2
9     addu   $v0, $s0, $s1
10    lw     $s0, 0($sp)
11    lw     $s1, 4($sp)
12    lw     $ra, 8($sp)
13    addiu  $sp, $sp, 12
14    jr     $ra

```

7.4 Subrutines multinivell

Patró:

func:

```

    addiu $sp, $sp, -TAM_BLOC
    sw    $s0, TAM_LOCALS($sp)
    sw    $s1, TAM_LOCALS+4($sp)
    ...
    #      $s0 = VALOR A PRESERVAR
    #      $s1 = VALOR A PRESERVAR

    #      COS SUBROUTINA

    lw    $s0, TAM_LOCALS($sp)
    lw    $s1, TAM_LOCALS+4($sp)
    ...
    addiu $sp, $sp, TAM_BLOC
    jr    $ra

```

7.4 Subrutines multinivell

Situació 1: $x = f()$ vs. $*x = f()$

```

1  int f1 () {
2      int x = 1;
3      x = f2 ();
4      return x;
5  }
```

(no cal preservar x -> bloc a. = 4 bytes)

```

1  void f1 (int *x) {
2      *x = f2 ();
3  }
```

(cal preservar x -> bloc a. = 8 bytes)

7.4 Subrutines multinivell

Situació 2: $f1() + f2()$

```
1  int f1 () {  
2      int ;  
3      x = f2 () + f3 () ;  
4      return x ;  
5  }
```

(cal preservar result de f3 -> bloc a. = 8 bytes)

7.4 Subrutines multinivell

Situació 3: Crides recursives / dins condicionals /bucles

```
1  int f(int a) {  
2      int r;  
3      if (a == 0) {  
4          r = f();  
5      else  
6          r = 1  
7      return r;  
8  }
```

(Les recursives les tractem com qualsevol funció)

(No niuarem la preservació del context)

7.4 Subrutines multinivell

Una altre exemple situació 3 (bucle):

```
1  int f1(int c) {  
2      int s = 0;  
3      for (i = 0; i < c; i++)  
4          s = s + f2();  
5      return s;  
6  }
```

(Cal guardar i, c i s)

7.4 Subrutines multinivell

Exemple complert:

```

1  int f(int m, int *n);
2  int g(char *y, char *z);
3  char exemple(int a, int b, int c) {
4      int d, e, q;
5      char v[18], w[20];
6      d = a + b;
7      e = f(d, &q) + g(v, w);
8      return v[e + q] + w[d + c];
9  }
```

- Calcularem primer f i després g.
- Cal guardar: resultat de f, d, c, \$ra.
- No cal: e, q (a la pila).

7.4 Subrutines multinivell

Exemple complert:

```

0 ($sp) |  q  | 4b
4 ($sp) |  v  | 18b
22 ($sp) |  b  | 20b
        |      | 2b
44 ($sp) | $s0 | 4b
48 ($sp) | $s1 | 4b
52 ($sp) | $s2 | 4b
56 ($sp) | $ra | 4b

```