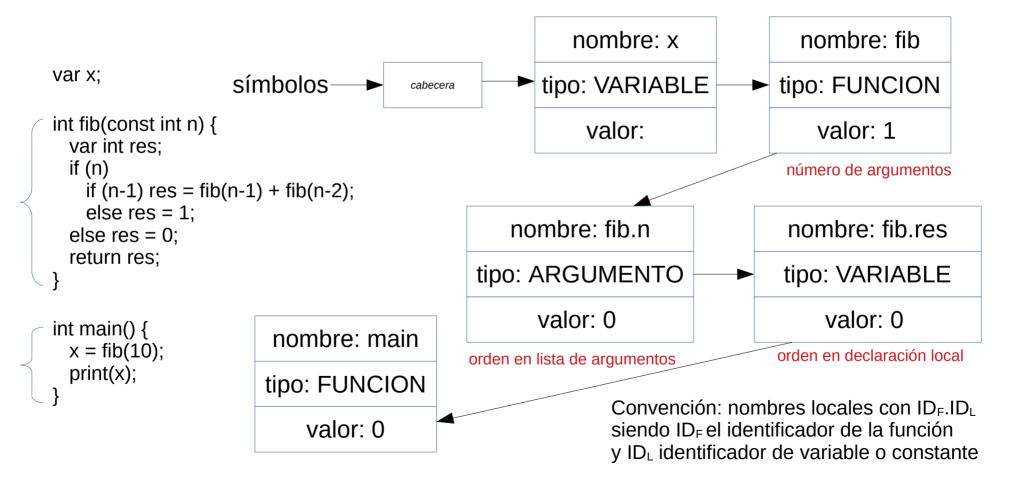
Generación de código Opción: declaración de funciones y llamadas a funciones

Ejemplo y tabla de símbolos



Cambios en la gramática

```
program -> declarations functions
functions -> functions function
          lλ
function -> tipo ID "(" arguments ")" "{" declarations statement list "}"
arguments -> CONS tipo arg list
         I %empty
arg list -> ID
          arg list "," ID
                                                                        Una variable global puede guardar el IDF
                                                                        de la función que se está traduciendo.
statement -> ID "(" call arguments ")"
expression -> ID "(" call arguments ")"
                                                                        En el análisis semántico se puede buscar
                                                                        primero:
call arguments -> expressions
          | %empty
                                                                        ID_{F}.ID_{I}
expressions -> expression
         expressions "," expression
                                                                        Si no se encuentra, se busca ID<sub>L</sub>
```

Estructura de la traducción

```
var x:
                                           .data
int fib(const int n) {
                                           $str0: .asciiz "\n"
 var int res;
                                           x: .word 0
 if (n)
   if (n-1) res = fib(n-1) + fib(n-2);
                                           else res = 1:
                                           .text
 else res = 0;
                                           .globl main
 return res;
                                           _fib:
                                                # Salvar registros en pila
int main() {
 x = fib(10);
                                                # Recuperar registros de pila
 print(x);
                                                jr $ra
                                           main:
```

Generación de código funciones (1)

- ¿Cómo se pueden referenciar los argumentos?
 - En registros \$aX, 0 <= X <= 3, siendo X el valor del argumento.
- Implicación importante:
 - El uso de identificadores que son argumentos se convierte en \$aX en lugar de la etiqueta del identificador.
- ¿Cómo se devuelve un valor desde la función?
 - En el registro \$v0

Generación de código funciones (2)

- ¿Cómo se referencian las variables y constantes locales?
 - Hay que usar la pila para permitir llamadas recursivas.
- Implicación importante:
 - El uso de identificadores que son variables y constantes locales se convierte en X(\$sp) donde:
 - X = 4*simbolo.valor
- Conviene usar una función char * get_ref_id(char *id) que devuelva la cadena para hacer referencia a un identificador.

Traducción de una función

```
_fib:
                                                                                                                   $12:
                                                             addiu $sp,$sp.-20
                                                                                             addiu $sp,$sp,-4
int fib(const int n) {
                                                                                                                         lw $t0,0($sp)
                                                                                             sw $a0,0($sp)
                                                             sw $t0,16($sp)
  var int res;
                                                                                                                         move $v0,$t0
                                                                                             li $t2.2
                                                             sw $t1,12($sp)
                                                 salvar
  if (n)
                                                                                                                         addiu $sp.$sp.4
                                                                                             sub $t3, $a0, $t2
                                                             sw $t2,8($sp)
                                                 a pila
    if (n-1) res = fib(n-1) + fib(n-2);
                                                                                                                         lw $ra,0($sp)
                                                                                             move $a0,$t3
                                                             sw $t3,4($sp)
                                                                                                                         lw $t3,4($sp)
                                                             sw $ra,0($sp)
                                                                                            jal fib
    else res = 1:
                                                                                                                         lw $t2,8($sp)
                                                                                             move $t2,$v0
                                                                                                              recup.
                                                             move $v0.$zero
  else res = 0:
                                                                                                                         lw $t1,12($sp)
                                                                                             lw $a0,0($sp)
                                                    res-
                                                             addiu $sp,$sp,-4
                                                                                                              de pila
  return res;
                                                                                                                         lw $t0,16($sp)
                                                             begz $a0,$13
                                                                                             addiu $sp,$sp,4
                                                                                                                         addiu $sp.$sp.20
                                                             li $t0.1
                                                                                             add $t3,$t0,$t2
                                                                                                                         ir $ra
                                                             sub $t1, $a0, $t0
                                                                                             sw $t3,0($sp)
                                                             begz $t1,$11
                                                                                             j $10
                                                             addiu $sp,$sp,-4
                                                                                      $11:
                                                             sw $a0,0($sp)
                                                                                            li $t0,1
                                                             li $t0.1
                                                                                             sw $t0,0($sp)
                                                             sub $t2,<mark>$a0</mark>,$t0
                                                                                      $10:
                                                             move $a0.$t2
                                        llamada a fib
                                                                                             i $12
                                                             ial fib
                                                                                      $13:
                                                             move $t0.$v0
                                                                                             li $t0.0
                                                             lw $a0,0($sp)
                                                                                             sw $t0,0($sp)
                                                             addiu $sp,$sp,4
```

print y read (1)

Peligro con print y read

```
int y(const int a) {
    print(2);
    ...
}
int main() {
    print(y(1));
}
```

```
_y:
    addiu $sp,$sp,-8
    sw $t0,4($sp)
    sw $ra,0($sp)
    move $v0,$zero
    li $t0,2
    li $v0,1
    move $a0,$t0
    syscall
...
```

Antes del syscall es necesario salvar el contenido de los registros \$a0 y \$v0 si estamos en una función

print y read (2)

```
_y:
                                          addiu $sp,$sp,-8
                                          sw $t0,4($sp)
                                          sw $ra, 0 ($sp)
int y(const int a) {
                                          move $v0,$zero
  print(2);
                                          li $t0,2
                                         move $s0,$a0
                                         move $s1,$v0
int main() {
                                          li $v0,1
  print(y(1));
                                          move $a0,$t0
                                          syscall
                                          move $a0,$s0
                                         move $v0,$s1
                                          lw $ra,0($sp)
                                          lw $t0,4($sp)
                                          addiu $sp,$sp,8
                                          jr $ra
```

Por simplificar, se pueden salvar en los registros \$s0 y \$s1