UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Escuela Técnica Superior de Ingeniería Informática

Procesadores de Lenguajes

Tema 7

El entorno de Ejecución
Compilación de subprogramas

Javier Vélez Reyes
jvelez@lsi.uned.es

Objetivos del Tema

- Estudiar el entorno de ejecución
  - ¿Qué es el entorno de ejecución?
  - ¿Qué estructura tiene?
  - ¿Para qué se utiliza?
- Aprender a gestionarlo para subprogramas
  - Funciones no recursivas
  - Funciones recursivas directas e indirectas
  - Funciones locales

Javier Vélez Reyes jvelez@lsi.uned.es

Índice General

- Introducción
  - El entorno de ejecución
  - Registro de activación
  - Secuencia de llamada y retorno
  - Comprobaciones semánticas
  - Implementación con m2r
  - Gestión de tabla de tipos
  - Gestión de la tabla de símbolos
- Funciones sin recursividad
  - Compilación del cuerpo de la función
  - Compilación de la llamada a la función

Javier Vélez Reyes jvelez@lsi.uned.es

Índice General

- Funciones con recursividad
  - Compilación del cuerpo de la función
  - Compilación de instrucciones
  - Compilación de la llamada a la función
- Funciones locales
  - Encadenamiento de accesos
  - Display
- Paso de parámetros

Javier Vélez Reyes jvelez@lsi.uned.es

Introducción

- El entorno de ejecución

> El entorno de ejecución viene dado por la estructura de memoria incluidos los registros de la unidad central de proceso y la forma de gestionarla que permite desarrollar adecuadamente el proceso de ejecución de un programa

- Condiciona
  - La gestión de memoria dinámica
  - La implementación de ámbitos
  - La gestión de funciones y procedimientos

Javier Vélez Reyes jvelez@lsi.uned.es

El entorno de ejecución

- 2 zonas del entorno de ejecución
  - Zona de código
    - Contiene el código del programa
    - Contiene variables globales y estáticas
    - Permanece inalterada durante la ejecución
    - Es posible direccionar cada instrucción en compilación
  - Zona de datos
    - Contiene variables locales
    - Mantiene información para gestionar funciones recursivas
    - Contiene espacio para datos dinámicos (malloc)
    - No hay relación biunívoca entre variables y direcciones

Javier Vélez Reyes jvelez@lsi.uned.es

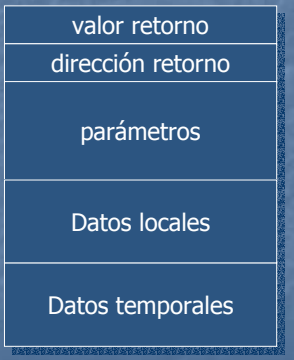
El entorno de ejecución

- Tipos de entornos de ejecución
  - Completamente estáticos
    - La dirección de cada dato es conocida en compilación
    - No existe recursividad, variables locales, memoria dinámica
    - Fortran 77
  - Basados en pila
    - Los datos se almacenan en una pila
    - Permite recursividad, variables locales, memoria dinámica
    - El código es conocido en tiempo de compilación
    - C/C++, Pascal, Ada
  - Completamente dinámicos
    - Código y datos pueden modificarse en ejecución
    - Permite generar y modificar funciones
    - Lisp, Prolog

Javier Vélez Reyes jvelez@lsi.uned.es

Registro de activación

- Registro de activación RA contiene datos de función
  - Parámetros de la función
  - Variables locales y temporales
  - El valor devuelto
  - Dirección de retorno
  - …
- Se almacena en
  - Área estática en Fortran 77
  - Pila en C/C++, Pascal o Ada
  - Montículo en Lisp



Javier Vélez Reyes jvelez@lsi.uned.es

Secuencias de llamadas y retorno

- Llamar una función implica
  - Una secuencia de llamada
  - Ejecutar la función
  - Una secuencia de retorno
- Secuencia de llamada (Call sequence)
  - Reserva de espacio para el RA
  - Almacenamiento de la dirección de retorno
  - Cálculo y almacenamiento de los parámetros actuales
  - Salvaguarda del estado de ejecución (registros)
- Secuencia de retorno (return sequence)
  - Almacenar el valor devuelto por la función
  - Devolver el control al programa llamante

Javier Vélez Reyes jvelez@lsi.uned.es

Comprobaciones semánticas

- Al trabajar con funciones el ETDS debe comprobar
  - No se usan paréntesis con identificadores de variable
  - No se usan identificadores de funciones como variables
  - En una llamada debe coincidir
    - El número, tipo y orden de los parámetros formales
    - El número, tipo y orden de los parámetros actuales
  - Realizar conversiones implícitas entre tipos compatibles

Javier Vélez Reyes jvelez@lsi.uned.es

Implementación con m2r

- Ejemplo
  - Máquina virtual m2r
  - Entorno de ejecución estático
  - Todos los datos ocupan una posición de memoria

```
int a;
int g1 () {
    return 10;
}
int g2 (int a, int b) {
    int x, y, z;
    return a*b;
}
main () {
     a = g2(3,5);
}
```

```
        jmp L3
L1:     Código de g1
L2:     Código de g2
L3:     Código de main
```

100	Valor retorno
101	Dirección retorno
102	Parámetro a
103	Parámetro b
104	Variable local x
105	Variable local y
106	Variable local z
107	Temporales locales

Javier Vélez Reyes jvelez@lsi.uned.es

Implementación con m2r

- Ejemplo
  - Código en el llamante

```
L4:  mvetq L4 101
     jmp L2
     mov 100 d(a)
```

  - Código de g2

```
mov 102 107
mov 103 108
mov 107 A
muli 108
mov A 109
```

  - Código de secuencia de retorno

```
mov 109 100
mov 101 A
jmp @A
```

Javier Vélez Reyes jvelez@lsi.uned.es

Gestión de la tabla de tipos

- Tipo de una función caracterizado por
  - Tipo de cada argumento
  - Tipo del valor de retorno
- Ejemplo

```
Float foo (int i, float f, char c)
```

Entrada	Tipo	Tipo Base / Argumento	Argumento / Retorno
0	Entero		
1	Real		
2	Carácter		
3	Producto cartesiano	0	1
4	Producto cartesiano	3	2
5	Función	4	1

Javier Vélez Reyes jvelez@lsi.uned.es

Gestión de la tabla de símbolos

- El cuerpo de la función se trata como un ámbito
  - Se guardan en la tabla de símbolos
    - Variables locales
    - Variables temporales
    - Argumentos
  - Se borran al final de la compilación de la función
  - El símbolo de la función no se borra nunca
    - Se puede llamar desde cualquier punto del programa

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones sin recursividad

- No existen funciones recursivas
  - Se reserva una zona para los datos globales
  - Se reserva una zona para el RA de cada función
  - El entorno puede ser completamente estático
- ¿Cómo se genera el código…?
  - Para cuerpo de la función
  - Para la llamada de la función

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones sin recursividad

- Compilación del cuerpo
  - Gestión de la TS y TT
    - Insertar las entradas para la función en la TT
    - Insertar las entradas para la función en la TS
    - Guardar en TS posición de declaraciones locales
    - Insertar en la TS los argumentos de la función
    - Insertar en la TS las variables conforme se declaren
    - Al terminar borrar entradas de la TS

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones sin recursividad

- Compilación del cuerpo
  - Almacenar el valor de retorno
    - Generar código para copiar el valor devuelto en el RA
    - Comprobar coincidencia de tipos con el definido en la función
    - Realizar conversiones de tipos oportunas
  - Devolver control al llamante
    - Generar código para devolver control al llamador
    - Utilizar la información de retorno del RA
  - Comprobar
    - Existe valor de retorno
    - Asignar uno por defecto

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones sin recursividad

- Compilación de una llamada
  - Comprobaciones semánticas
    - Numero, tipo y orden de los argumentos correcto
    - Generar, si procede, conversiones de tipos apropiadas
    - El tipo devuelto es adecuado para la expresión llamante
    - Generar, si procede, conversión de tipo para el valor devuelto
  - Evaluación de parámetros
  - Almacenar la dirección o etiqueta de retorno en el RA
  - Generar código para saltar al comienzo de la función
  - Recuperar el valor devuelto por la función

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones sin recursividad

- Ejemplo

```
int a;
int f1 (int b) {
    int c;
    c = b*b;
    return c;
}
int b;
int f2 (int a, int b) {
    int  c;
    c = f1(a) + f1(b);
    return c;
}
int c;
main () {
     a = 7;
     a = f2(a,3);
}
```

0	Variable global a	
1	Etiqueta de retorno	
2	Valor devuelto	
3	Parámetro b	
4	Variable local c	RA de f1
5	Temporal t1	
6	Temporal t2	
7	Temporal t3	
8	Variable global b	
9	Etiqueta de retorno	
10	Valor devuelto	
11	Parámetro a	
12	Parámetro b	
13	Variable local c	RA de f2
14	Temporal t1	
15	Temporal t2	
16	Temporal t3	
17	Temporal t4	
18	Temporal t5	
19	Variable global c	

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones sin recursividad

- Ejemplo. Tabla de símbolos

Mientras se procesa f1

Identificador	Tipo	Dirección	Etiqueta
a	entero	0	
f1	t (f1)	1	L2
b	entero	3	
c	entero	4	

Mientras se procesa f2

Identificador	Tipo	Dirección	Etiqueta
a	entero	0	
f1	t (f1)	1	L2
b	entero	8	
f2	t (f2)	9	L3
a	entero	11	
b	entero	12	
c	entero	13	

Mientras se procesa main

Identificador	Tipo	Dirección	Etiqueta
a	entero	0	
f1	t (f1)	1	L2
b	entero	8	
f2	t (f2)	9	L3
c	entero	19	

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones sin recursividad

- Ejemplo. Código m2r

```
    jmp L1
    ; Código de f1:
L2  mov 3 5         ; t1 = b
    mov 3 6         ; t2 = b
    mov 5 A
    muli 6          ; b*b
    mov A 7         ; t3 = b*b
    mov 7 4         ; c = t3
    mov 4 5         ; t1 = c
    mov 5 2         ; valor devuelto = t1
    mov 1 A
    jmp @A          ; return c

    ; Código de f2:
L3  mov 11 14       ; t1 = a
    mov 14 3        ; parámetro b de f1 = t1
    mvetq L4 1      ; Guarda retorno
    Jmp L2          ; salta a f1
L4  mov 2 15        ; t2 = valor retorno de f1
    mov 12 16       ; t3 = b
    mov 16 3        ; parámetro b de f1 = t3
    mvetq L5 1      ; Guarda retorno
    Jmp L2          ; salta a f1
L5  mov 2 17        ; t4 = valor retorno f1
    mov 15 A
    addi 17         ; f1(a) + f1(b)
    mov A 18        ; t5 = f1(a) + f1(b)
    mov 18 13       ; c = t5
    mov 13 14       ; t1 = c
    mov 14 10       ; valor devuelto = t1
    mov 9 A
    Jmp @A          ; return c

    ; Código de main:
L1  mov #7 15000    ; t1 = 7
    mov 15000 0     ; a = 7
    mov 0 15000     ; t1 = a
    mov 15000 11    ; parámetro a de f2 = t1
    mov #3 15001    ; t2 = 3
    mov 15001 12    ; parámetro b de f2 = t2
    mvetq L6 9      ; Guarda retorno
    Jmp L3          ; salta a f2
L6  mov 10 15002    ; t3 = valor retorno de f2
    mov 15002 0     ; a = f2(a,3)
    halt
```

Javier Vélez Reyes jvelez@lsi.uned.es

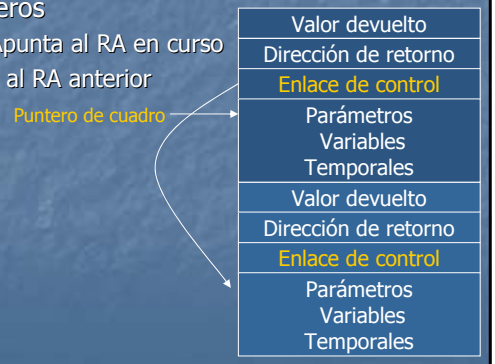
Funciones con recursividad

- Existen funciones recursivas
  - El espacio de los RA no puede reservarse estáticamente
    - Se llama a una función dentro del código de sí misma
    - Es necesario un RA para cada nueva llamada
    - No se puede conocer el número de llamadas recursivas
    - No se puede conocer en compilación el espacio necesario
  - Si el entorno fuese completamente estático
    - Cada llamada recursiva competiría por acceder al RA
    - No podría existir recursividad

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Existen funciones recursivas
  - Solución
    - Utilizar una pila de ejecución de RA
    - Cada llamada a una función apila un RA
    - Al finalizar una función se desapila el RA asociado de la cima
    - Deben mantenerse 2 punteros
      - Puntero de cuadro (FP). Apunta al RA en curso
      - Enlace de control. Apunta al RA anterior



Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Compilación del cuerpo de una función
  - Guardar la función en TS { a }
    - Guardar el símbolo de la función sin tipo en TS
    - Abrir un nuevo ámbito en TS
    - Poner a 0 la dirección del primer argumento (relativa a FP)
  - Poblar TT y TS con los argumentos { b }
    - Almacenar los argumentos en la tabla de símbolos
    - Almacenar el tipo de la función en TT
  - Espacio de temporales { c }
    - Añadir temporales a continuación de la última variable local
  - Cerrar el ámbito { d }
    - Eliminar los símbolos locales de TS
    - Generar una secuencia de retorno por defecto

Func	::= Tipo id ({ a } Args) { b } Bloque { d }
Bloque	::= begin declaraciones { c } sentencias end

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Compilación de instrucciones
  - Deben distinguirse
    - Variables globales. Direccionamiento absoluto
    - Variables locales. Direccionamiento relativo a FP
  - Las variables temporales
    - Del programa principal. Direccionamiento absoluto
    - De funciones. Direccionamiento relativo a FP
  - La instrucción de retorno
    - Almacena el valor a devolver en el RA
    - Salta a la dirección de retorno marcada en RA

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Compilación de una llamada
  - Reservar el espacio necesario para el RA
  - Gestionar los parámetros
    - Comprobar tipo de cada parámetro
    - Realizar las conversiones de tipo adecuadas
    - Generar código para situar cada parámetro en su posición
  - Preparar salto
    - Generar código para almacenar enlace de control actual
    - Actualizar el valor de FP
    - Saltar a la función
  - Restaurar tras llamada
    - Restaurar FP al valor en enlace de control
    - (Puede hacerse en la secuencia de retorno)

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Ejemplo
  - Traducción de la función factorial a m2r
  - B contiene FP

```
int a;
int fact (

int n) {

if (n <= 1) {
```

```
    mov 10000 B              ; temporales del main
    jmp L6                   ; salta a main

Reservar 0 para a
Guarda fact en TS
L1 marca el comienzo del código
Marca el comienzo de la TS de fact
Guardar n en TS de fact con dirección relativa 0
Guardar el tipo de la función en la TT

L1  mov @B+0 @B+1            ; Guarda n en temporal @B+1
    mov #1 @B+2              ; Guarda 1 en temporal @B+2
    mov @B+1 A
    leqi @B+2                ; n<=1?
    mov A @B+3               ; resultado en temporal @B+3
    mov @B+3 A
    jz L3                    ; _Si n > 1 salta a 3
```

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Ejemplo
  - Traducción de la función factorial a m2r
  - B contiene FP

```
return 1;




else return fact (

            n-1
```

```
    mov #1 @B+4              ; Guarda 1 en temporal @B+4
    mov @B+4 @B-3            ; Copia 1 en el valor de retorno
    mov @B-2 A               ; Carga  la dirección de retorno
    jmp @A                   ; salto de retorno
    jmp L4                   ; saltar al final del if

Reservar sitio para otro RA (3 posiciones fijas + 1 argumento)
Reserva desde @B+5 hasta @B+8

L3  mov @B+0 @B+9            ; Guarda n en temporal @B+9
    mov #1 @B+10             ; Guarda 1 en temporal @B+10
    mov @B+9 A
    subi @B+10               ; n-1
    mov A @B+11              ; resultado en temporal @B+11
    mov @B+11 @B+8           ; parámetro de fact = n-1
```

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Ejemplo
  - Traducción de la función factorial a m2r
  - B contiene FP

```
)
```

Llamada a fact

```
    mov  B @B+7          ; guarda B anterior
    mov B A
    addi #8              ; 8 = 4 (temporales usadas) +
                         ;     3 (espacio del RA) +1
    mov A B              ; B apunta al nuevo RA
    mvetq L2 @B-2        ; pone etiqueta de retorno
    jmp L1               ; salto a la función fact
L2  mov @B-1 B           ; restaurar el valor de B
                         ;    el valor devuelto en @B+5
                         ;    el resto de temporales se
                         ;    reciclan
```

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Ejemplo
  - Traducción de la función factorial a m2r
  - B contiene FP

```
        * n
}
```

```
    mov @B+0 @B+6        ; guarda n en temporal @B+6
    mov @B+5 A           ; A = fact (n-1)
    muli @B+6            ; n * valor devuelto
    mov A @B+7           ; sobrescribe el antiguo valor de retorno
    mov @B+7 @B-3        ; devuelve el resultado del producto
    mov @B-2 A
    jmp A                ; return fact (n-1) * n

L4  mov @B-2 A           ; fin de if
    jmp @A               ; secuencia de retorno por defecto
                         ; por si se alcanza el final de la función
                         ; sin hacer return
```

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Ejemplo
  - Traducción de la función factorial a m2r
  - B contiene FP

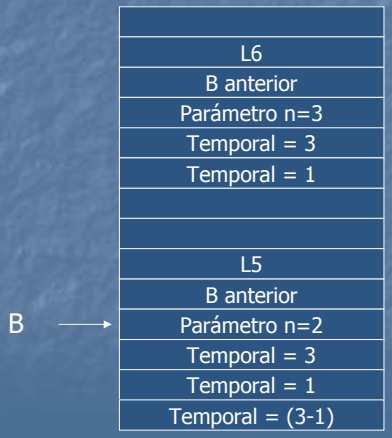
```
int main ( ) {

  a = fact(
            3);




}
```

```
Llamada a fact
Reservar 3+1 posiciones para el RA
Reserva desde @B+0 a @B+3

L6 mov  #3 @B+4          ; guarda 3 en temporal @B+4
   mov @B+4 @B+3         ; parámetro de fact = 3
   mov B @B+2            ; guarda B anterior
   mov @B A
   addi #3
   mov A B               ; pone la nueva B
   mvetq L5 @B-2         ; pone etiqueta de retorno
   jmp L1                ; salta a la función
L5 mov @B-1 B            ; deja la B como estaba y
                         ; termina la llamada
   mov @B+0 0            ; asignación a la variable a
                         ; del valor que devuelve la función

   halt
```

Javier Vélez Reyes jvelez@lsi.uned.es

Funciones con recursividad

- Ejemplo
  - Traducción de la función factorial a m2r
  - B contiene FP



Javier Vélez Reyes jvelez@lsi.uned.es

Funciones locales

- Hay lenguajes permiten definir funciones locales
  - Su ámbito se reduce a la función que la define
  - Son funciones anidadas dentro de otras
  - Pascal, Ada
- Ejemplo
  - *Vabuelo* es Nonlocal reference
    - No es ni global
    - No es local
  - Pertenece al ámbito de *abuelo*

```
int abuelo () {
    int Vabuelo;
    int padre () {
        int hijo () {
            Vabuelo = 7;
        }
        int hermano () {
            hijo ();
        }
        hermano ();
    }
    int tio () {
        padre ();
    }
    tio ();
}
```

Javier Vélez Reyes jvelez@lsi.uned.es

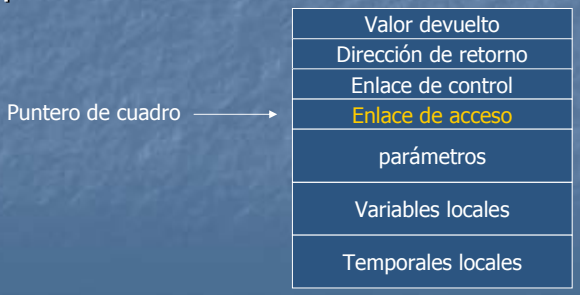
Funciones locales

- ¿Cómo es la pila de ejecución?
- La referencia a Vabuelo en hijo
  - Requiere aplicar encadenamiento
    - No es posible acceder directamente
    - Hay que ir saltando hasta RA del abuelo
    - A través de los enlaces de control
  - Inconveniente
    - No es posible conocer el número de saltos para alcanzar RA abuelo en tiempo de compilación ya que los saltos dependen del punto de llamada de la función
  - Otras técnicas
    - Encadenamiento de accesos
    - Display

Javier Vélez Reyes jvelez@lsi.uned.es

Encadenamiento de accesos

- Encadenamiento de accesos (access chaining)
  - Añadir un campo enlace de acceso (access link) al RA
  - Apunta al RA de la función padre
  - El enlace de control apunta al RA de la función llamante
  - El puntero de cuadro apunta al enlace de acceso del RA
- Esquema del RA



Javier Vélez Reyes jvelez@lsi.uned.es

Encadenamiento de accesos

- Ejemplo
  - Código para acceder a Vabuelo en m2r
  - B mantiene el puntero de cuadro
  - Abuelo no tiene parámetros
  - Vabuelo es la primera variable en abuelo
  - La dirección que guarda la variable es t1

```
mov @B+0 A      ; salto al RA de la función padre
mov @A  A       ; salto al registro de activación de la función abuelo
addi #1         ; desplazamiento para encontrar Vabuelo
mov @A t1       ; guarda la variable en el temporal t1
```

Javier Vélez Reyes jvelez@lsi.uned.es

Encadenamiento de accesos

- Acceso a Variables
  - Es necesario conocer en compilación el número de saltos
    - Se asocia un nivel de anidamiento a cada declaración (0, 1, 2…)
    - Se almacena en la tabla de símbolos
    - Se aplica la fórmula

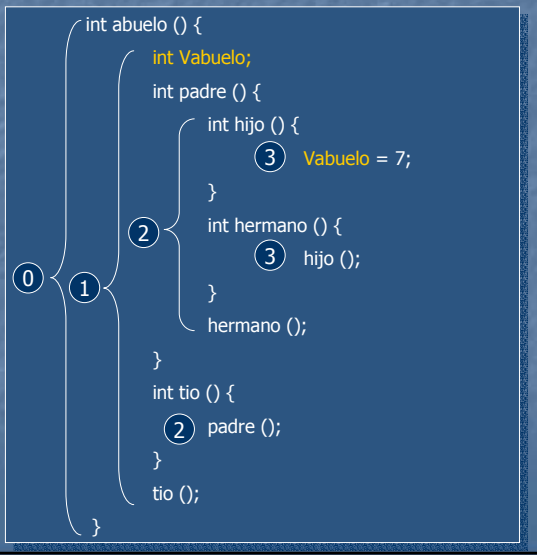
Número Saltos = Nivel de acceso – Nivel de variable accedida

- Ejemplo
  - Niveles
    - Nivel 0 abuelo
    - Nivel 1 padre, tio, variables locales de abuelo
    - Nivel 2 hijo, hermano, variables locales de tio
    - Nivel 3 variables locales de hijo y hermano
  - Número de saltos
    - 3 – 1 = 2

Javier Vélez Reyes jvelez@lsi.uned.es

Encadenamiento de accesos

- Ejemplo (Continuación)
  - Esquema de niveles



Javier Vélez Reyes jvelez@lsi.uned.es

Encadenamiento de accesos

- Llamadas a funciones
  - La secuencia de llamada debe modificarse
    - Almacenar FP del padre en enlace de acceso
    - Acceder al FP es inmediato cuando
      - La función invocada es una función hijo
      - La función invocada es una función hermano
    - En otro caso es mas complicado ya que
      - Debe recorrerse la cadena de enlaces de acceso
      - Parar al encontrar el puntero de cuadro
      - Generar por cada salto una instrucción mov @A A
      - Si el número de saltos es grande esto es ineficiente

Javier Vélez Reyes jvelez@lsi.uned.es

Display

- Display
  - Alternativa al encadenamiento de accesos
  - Mantiene en tiempo de ejecución los enlaces de accesos
    - Utiliza un vector en memoria
    - Se indexa por el nivel de anidamiento
    - Este vector se llama Display
  - El acceso es directo
  - Se evita la secuencia de saltos

Javier Vélez Reyes jvelez@lsi.uned.es

Display

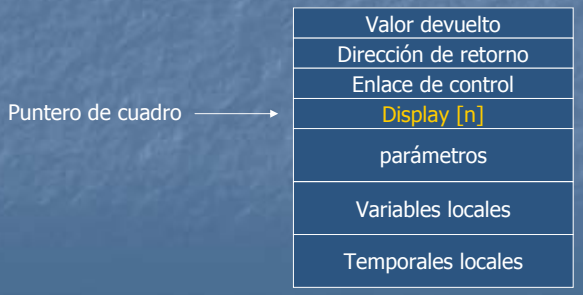
- Acceso a variables
  - TS indica el nivel de anidamiento n donde se declara
  - El puntero de cuadro estará en Display [n]
- Ejemplo
  - Resolver acceso a Vabuelo
  - Display empieza en posición 100
  - Nivel de abuelo es 0

```
mov 100 A
addi #1
mov @A t1
```

Javier Vélez Reyes jvelez@lsi.uned.es

Display

- Llamadas a funciones
  - Cuando se invoca a una función de nivel n
    - En display [n] se guarda la dirección de su RA (FP)
    - Se salva el antiguo valor de display [n] en el RA
    - Al terminar display [n] se restaura con el valor en RA
  - El RA pasa a ser

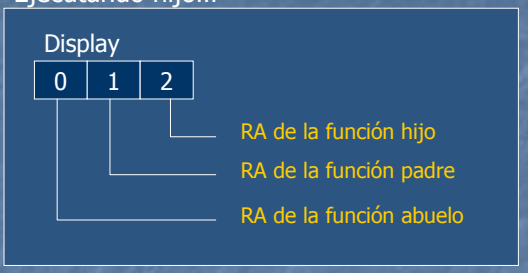


Javier Vélez Reyes jvelez@lsi.uned.es

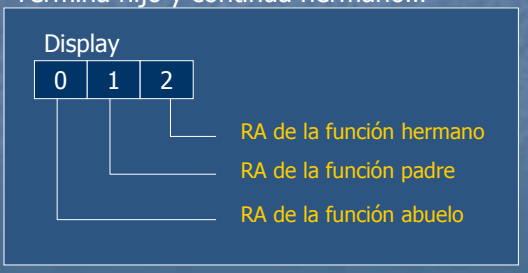
Display

- Ejemplo

Ejecutando hijo...



Termina hijo y continua hermano...



Javier Vélez Reyes jvelez@lsi.uned.es

Paso de parámetros

- Varias técnicas
  - Paso por valor
    - El valor del parámetro se copia en el RA de la función
    - El tratamiento es similar al de una variable local
    - Cualquier modificación desde la función no afecta al valor fuera
  - Paso por referencia
    - La dirección del parámetro se copia en el RA
    - Los accesos desde la función requieren resolver la indirección
    - Los cambios se ven afectados desde el exterior
    - Son parámetros de entrada / salida
  - Paso por valor-copia
    - Solución intermedia de las dos anteriores
    - El valor del parámetro se copia al RA de activación de la función
    - Al final el valor del parámetro se copia a la variable utilizada
    - Son parámetros de entrada /salida

Javier Vélez Reyes jvelez@lsi.uned.es

Bibliografía

[AJO] AHO, SETHI, ULLMAN: *Compiladores: Principios, técnicas y herramientas*,: addison-Wesley Iberoamericana, 1990

[GARRIDO] A. Garrido, J. Iñesta, F. Moreno y J. Pérez. 2002. *Diseño de compiladores*. Universidad de Alicante.