Referencias no locales

Referencias no locales

Cada subprograma constituye un bloque que determina un ámbito de visibilidad. En cada ámbito sólo están accesibles un subconjunto de todos los identificadores (constantes, tipos, variables o subprogramas) definidos en el programa. Es decir, las reglas de ámbito de un lenguaje definen el alcance que tienen los identificadores de un programa dentro de cada punto del mismo. En este sentido, en PL1UnedES distinguimos 3 posibilidades:

- **Referencias globales.** Estos identificadores globales son los que se declaran directamente dentro del módulo del programa. Se dice que pertenecen al ámbito global del mismo y, por tanto, están accesibles desde cualquier otro ámbito, ya sea éste el programa principal o cualquiera de los subprogramas definidos.

- **Referencias locales**. Los identificadores locales a un ámbito cuando son solamente accesibles desde dentro de dicho ámbito. Por ejemplo, todas las variables, constantes y tipos definidos dentro de un procedimiento son locales al mismo así como la colección de sus parámetros.

- **Referencias no locales.** Las referencias no locales son referencias a variables que no son globales y tampoco son locales, estando declaradas en algún punto dentro de la jerarquía de anidamiento de subprogramas.

Registro de Activación

El registro índice **IX** lo usaremos como puntero de marco, y la pila de la memoria empezara en las posiciones superiores e ira decrementándose.

#0[.IX]	Valor de Retorno
#-1[.IX]	Enlace de Control
#-2[.IX]	Estado Máquina
#-3[.IX]	**Enlace de Acceso**
#-4[.IX]	Parámetro Actuales
#-4 - nP[.IX]	Dirección de Retorno
#-5 - nP[.IX]	Variables locales
#-5 - np - nV[.IX]	Temporales

nP = número de parámetros
nV = número de variables locales

Ejemplo referencia no local en modulUned

```
MODULE test;

    VAR  suma : INTEGER;
         a1 : INTEGER;
         b1 : INTEGER;

    PROCEDURE Sumar (VAR a: INTEGER; VAR b: INTEGER);
         VAR s: INTEGER;

         PROCEDURE Imprimir();
         BEGIN
              WRITESTRING("Suma:");
              WRITEINT(s);  (* referencia no local a la variable s *)
         END Imprimir;

    BEGIN
         s := a+b;
         Imprimir();
    END Sumar;

BEGIN
   a1 := 2;
   b1 := 3;
   Sumar(a1,b1);
END test;
```

Referencias no locales

A la hora de referenciar variables o parámetros no locales podemos tener las siguientes posibilidades:

- **Encadenamientos de accesos**, mediante el nivel del ámbito de las variables
    - Al crear la variable en el código intermedio, además del nombre de la variable, y el ámbito en que se encuentra, añadimos el salto entre el ámbito que nos encontramos y el ámbito en el que se encuentra la variable declarada.

      ScopeIF scope = scopeManager.getCurrentScope();

      SymbolIF simbolo = scopeManager.searchSymbol(id.getLexema());

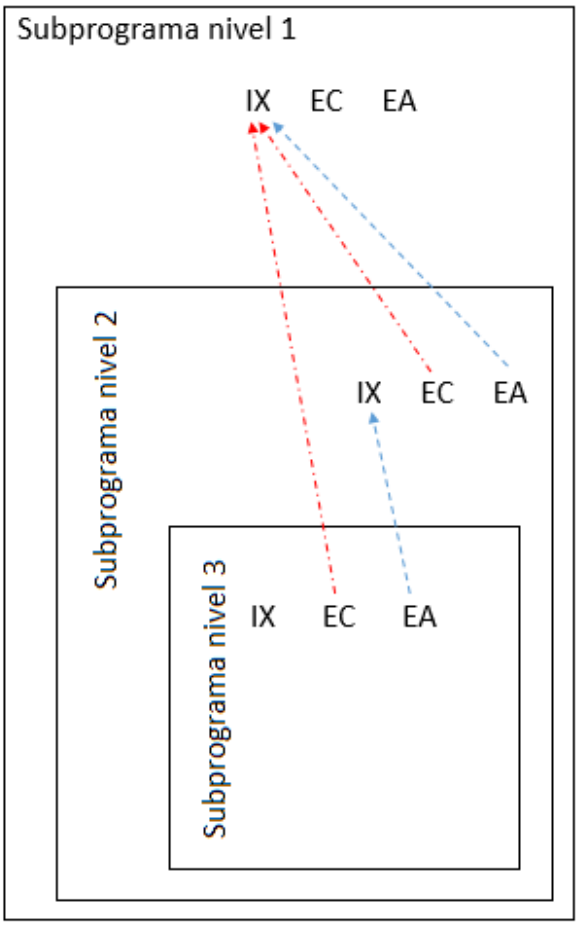
      int salto = scope.getLevel() - simbolo.getScope().getLevel();

      Variable var = new Variable(id.getLexema(), simbolo.getScope(), salto);

      cb.addQuadruple("MVA", temp, var);

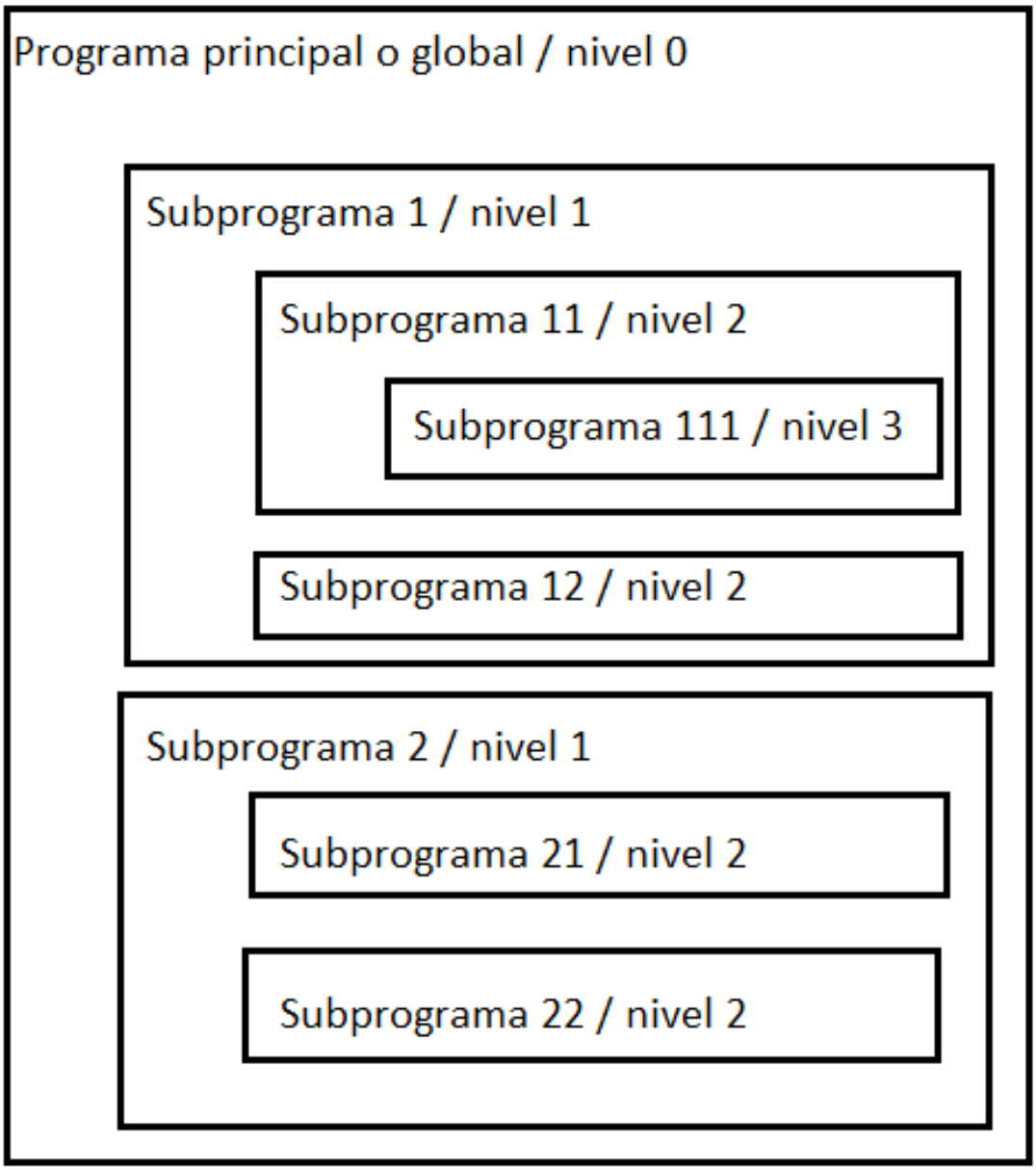
    - Al traducir el código intermedio a código final, recuperamos el salto de la variable y apoyándonos en el EA (enlace de acceso del R.A.) nos posicionamos en el índice del R.A. correspondiente a la variable.

```
if(quadruple.getOperation().equals("MVP")) {    // Quadruple - [MVP T_4, z, null],
        StringBuffer b = new StringBuffer();
        Variable v= (Variable)quadruple.getFirstOperand();
        if(v.isGlobal() || v.getSalto()==0){
                String o1 = operacion(quadruple.getFirstOperand());
                String r = operacion(quadruple.getResult());
                b.append("MOVE " + o1 + ", " + r);
        } else{
//Usaremos el registro índice IY para calcular la posición del registro índice del R.A. de la variable.
                b.append("MOVE #-3[.IX], .IY" + "\n");
                for(int i=1; i<v.getSalto();i++) {
                        b.append("MOVE #-3[.IY], .IY" + "\n");
                }
                String o1 = operacionIY(quadruple.getFirstOperand());
                String r = operacion(quadruple.getResult());
                b.append("MOVE " + o1 + ", " + r);
        }
        return b.toString();
}
Si nos encontramos en un bloque de nivel 3 y la variable esta declarada en el nivel 1
MOVE #-3[.IX], .IY
MOVE #-3[.IY], .IY
MOVE #-5[.IY], #-6[.IX]
```



- El EC (enlace de Control) de un subprograma apuntan al registro índice IX del subprograma llamante.

- El EA (enlace de Acceso) de un subprograma apuntan al registro índice IX del subprograma padre o anidante.


Programa principal o global / nivel 0
Subprograma 1 / nivel 1
Subprograma 11 / nivel 2
Subprograma 111 / nivel 3
Subprograma 12 / nivel 2
Subprograma 2 / nivel 1
Subprograma 21 / nivel 2
Subprograma 22 / nivel 2

IX Sub22	65200	EA = 65300	nivel = 2
IX Sub21	65250	EA = 65300	nivel = 2
IX Sub2	65300	EA = 65535	nivel = 1
IX Sub12	65350	EA = 65500	nivel = 2
IX Sub111	65400	EA = 65450	nivel = 3
IX Sub11	65450	EA = 65500	nivel = 2
IX Sub1	65500	EA = 65535	nivel = 1
IX Global	65535		nivel = 0

Código intermedio:

Quadruple - [STARTGLOBAL null, null, null]
Quadruple - [VARGLOBAL A1, 0, null]
Quadruple - [VARGLOBAL SUMA, 0, null]
Quadruple - [VARGLOBAL B1, 0, null]
Quadruple - [PUNTEROGLOBAL T_6, 12, null]
Quadruple - [MV T_0, 2, null]
Quadruple - [MVA T_1, A1, null]
Quadruple - [STP T_1, T_0, null]
Quadruple - [MV T_2, 3, null]
Quadruple - [MVA T_3, B1, null]
Quadruple - [STP T_3, T_2, null]
Quadruple - [STARTSUBPROGRAMA null, null, null]
Quadruple - [MVA T_4, B1, null]
Quadruple - [PARAM T_4, null, null]
Quadruple - [MVA T_5, A1, null]
Quadruple - [PARAM T_5, null, null]
Quadruple - [CALL L_SUMAR, null, null]
Quadruple - [HALT null, null, null]
Quadruple - [ETIQUETA L_SUMAR, null, null]
Quadruple - [VARSUBPROGRAMA S, 0, null]
Quadruple - [PUNTEROSUBPROGRAMA T_5, 15, null]
Quadruple - [MVP T_0, A, null]
Quadruple - [MVP T_1, B, null]
Quadruple - [ADD T_2, T_0, T_1]
Quadruple - [MVA T_3, S, null]
Quadruple - [STP T_3, T_2, null]
Quadruple - [STARTSUBPROGRAMA null, null, null]
Quadruple - [CALL L_IMPRIMIR, null, null]
Quadruple - [FINSUBPROGRAMA L_FIN_SUMAR, 7, null]
Quadruple - [ETIQUETA L_IMPRIMIR, null, null]
Quadruple - [PUNTEROSUBPROGRAMA T_2, 8, null]
Quadruple - [WRITESTRING T_0, L_0, null]
Quadruple - [MVP T_1, S, null]
Quadruple - [WRITEINT T_1, null, null]
Quadruple - [FINSUBPROGRAMA L_FIN_IMPRIMIR, 5, null]
Quadruple - [CADENA "SUMA:", L_0, null]

```
;Quadruple - [STARTGLOBAL null, null, null]
MOVE .SP, .IX
PUSH #-1
PUSH .IX
PUSH .SR
PUSH .IX
;Quadruple - [VARGLOBAL A1, 0, null]
PUSH #0
;Quadruple - [VARGLOBAL SUMA, 0, null]
PUSH #0
;Quadruple - [VARGLOBAL B1, 0, null]
PUSH #0
;Quadruple - [PUNTEROGLOBAL T_6, 12, null]
SUB .IX, #12
MOVE .A, .SP
;Quadruple - [MV T_0, 2, null]
MOVE #2, #-7[.IX]
;Quadruple - [MVA T_1, A1, null]
SUB .IX, #4
MOVE .A, #-8[.IX]
;Quadruple - [STP T_1, T_0, null]
MOVE #-8[.IX], .R1
MOVE #-7[.IX], [.R1]
;Quadruple - [MV T_2, 3, null]
MOVE #3, #-9[.IX]
;Quadruple - [MVA T_3, B1, null]
SUB .IX, #6
MOVE .A, #-10[.IX]
;Quadruple - [STP T_3, T_2, null]
MOVE #-10[.IX], .R1
MOVE #-9[.IX], [.R1]
;Quadruple - [STARTSUBPROGRAMA null, null, null]
MOVE .SP, .R0
PUSH #-1
PUSH .R0
PUSH .SR
PUSH .IX
;Quadruple - [MVA T_4, B1, null]
SUB .IX, #6
MOVE .A, #-11[.IX]
;Quadruple - [PARAM T_4, null, null]
PUSH #-11[.IX]
;Quadruple - [MVA T_5, A1, null]
SUB .IX, #4
MOVE .A, #-12[.IX]
;Quadruple - [PARAM T_5, null, null]
PUSH #-12[.IX]
```

```
;Quadruple - [CALL L_SUMAR, null, null]
MOVE .R0, .IX
CALL /L_SUMAR
MOVE .IX, .SP
MOVE #-3[.IX], .R0
MOVE .R0, .IX
;Quadruple - [HALT null, null, null]
HALT
;Quadruple - [ETIQUETA L_SUMAR, null, null]
L_SUMAR : NOP
;Quadruple - [VARSUBPROGRAMA S, 0, null]
PUSH #0
;Quadruple - [PUNTEROSUBPROGRAMA T_5, 15, null]
SUB .IX, #15
MOVE .A, .SP
;Quadruple - [MVP T_0, A, null]
MOVE #-4[.IX], .R1
MOVE [.R1], #-8[.IX]
;Quadruple - [MVP T_1, B, null]
MOVE #-5[.IX], .R1
MOVE [.R1], #-9[.IX]
;Quadruple - [ADD T_2, T_0, T_1]
ADD #-8[.IX], #-9[.IX]
MOVE .A , #-10[.IX]
;Quadruple - [MVA T_3, S, null]
SUB .IX, #7
MOVE .A, #-11[.IX]
;Quadruple - [STP T_3, T_2, null]
MOVE #-11[.IX], .R1
MOVE #-10[.IX], [.R1]
;Quadruple - [STARTSUBPROGRAMA null, null, null]
MOVE .SP, .R0
PUSH #-1
PUSH .R0
PUSH .SR
PUSH .IX
;Quadruple - [CALL L_IMPRIMIR, null, null]
MOVE .R0, .IX
CALL /L_IMPRIMIR
MOVE .IX, .SP
MOVE #-3[.IX], .R0
MOVE .R0, .IX
;Quadruple - [FINSUBPROGRAMA L_FIN_SUMAR, 7, null]
L_FIN_SUMAR : NOP
SUB .IX, #7
MOVE .A, .SP
RET
```

```
;Quadruple - [ETIQUETA L_IMPRIMIR, null, null]
L_IMPRIMIR : NOP
;Quadruple - [PUNTEROSUBPROGRAMA T_2, 8, null]
SUB .IX, #8
MOVE .A, .SP
;Quadruple - [WRITESTRING T_0, L_0, null]
WRSTR /L_0
;Quadruple - [MVP T_1, S, null]
MOVE #-3[.IX], .IY
MOVE #-7[.IY], #-6[.IX]
;Quadruple - [WRITEINT T_1, null, null]
WRINT #-6[.IX]
;Quadruple - [FINSUBPROGRAMA L_FIN_IMPRIMIR, 5, null]
L_FIN_IMPRIMIR : NOP
SUB .IX, #5
MOVE .A, .SP
RET
;Quadruple - [CADENA "SUMA:", L_0, null]
L_0 : DATA "SUMA:"
```

- **Técnica del display**, mediante un vector con el valor del registro índice de cada subprograma.
  - Creamos un variable para indicar la posición en memoria que tendrá el vector del display.
  - En el vector vamos guardando el valor del registro índice de cada subprograma y bloque.
  - Al traducir el código intermedio a código final, comprobamos si se trata de una variable global, local o no local y si se trata de una variable no local recuperamos recuperaremos del vector su registro índice del R.A. donde ésta declarada la variable.

Creamos una cuadrupla con la etiqueta del subprograma y valor que representa el orden de creación del ámbito del subprograma.

```
ScopeIF scope = scopeManager.getCurrentScope();
LabelFactoryIF lF= new LabelFactory();
LabelIF l1 = lF.create(scope.getName());
cb.addQuadruple("ETIQUETA", l1, new Value(scope.getId()*));
```

scope.getId() indica el orden de creación del ámbito siendo cero el del ámbito del programa principal

```
private static int display=10000;  //posición de memoria del vector del display
```

```
; Quadruple - [ETIQUETA L_Subprograma111, 3, null]
if(quadruple.getOperation().equals("ETIQUETA")) {
        StringBuffer b = new StringBuffer();
        String r = operacion(quadruple.getResult());
        String o1 = operacion(quadruple.getFirstOperand());
        b.append(r + " : NOP ");
        b.append("MOVE " + ".IX" + ", " + o1);          //IX registro índice
        return b.toString();
}

private String operacion (OperandIF o) {
        if(o instanceof Value){
                return "/" + (display + ((Value)o).getValue());
  }
L_Subprograma111 : NOP

MOVE .IX, /10003
```

Posición de memoria	Contenido	Ámbito
10000	65535	Global
10001	65500	Subprograma1
10002	65450	Subprograma11
10003	65400	Subprograma111
10004	65350	Subprograma12
10005	65300	Subprograma2
10006	65250	Subprograma21
10007	65200	Subprograma22

Suponiendo que nos encontramos en el Subprograma111 y referenciamos la variable “x” que se encuentra en el Subprograma11

;Quadruple - [MVA T_2, x, null]

MOVE #10002, .R0

SUB [.R0], #5

MOVE .A, #-7[.IX]

Código intermedio:

Quadruple - [STARTGLOBAL null, null, null]
Quadruple - [ETIQUETA L_TEST, 0, null]
Quadruple - [VARGLOBAL A1, 0, null]
Quadruple - [VARGLOBAL SUMA, 0, null]
Quadruple - [VARGLOBAL B1, 0, null]
Quadruple - [PUNTEROGLOBAL T_6, 12, null]
Quadruple - [MV T_0, 2, null]
Quadruple - [MVA T_1, A1, null]
Quadruple - [STP T_1, T_0, null]
Quadruple - [MV T_2, 3, null]
Quadruple - [MVA T_3, B1, null]
Quadruple - [STP T_3, T_2, null]
Quadruple - [STARTSUBPROGRAMA null, null, null]
Quadruple - [MVA T_4, B1, null]
Quadruple - [PARAM T_4, null, null]
Quadruple - [MVA T_5, A1, null]
Quadruple - [PARAM T_5, null, null]
Quadruple - [CALL L_SUMAR, null, null]
Quadruple - [HALT null, null, null]
Quadruple - [ETIQUETA L_SUMAR, 1, null]
Quadruple - [VARSUBPROGRAMA S, 0, null]
Quadruple - [PUNTEROSUBPROGRAMA T_5, 15, null]
Quadruple - [MVP T_0, A, null]
Quadruple - [MVP T_1, B, null]
Quadruple - [ADD T_2, T_0, T_1]
Quadruple - [MVA T_3, S, null]
Quadruple - [STP T_3, T_2, null]
Quadruple - [STARTSUBPROGRAMA null, null, null]
Quadruple - [CALL L_IMPRIMIR, null, null]
Quadruple - [FINSUBPROGRAMA L_FIN_SUMAR, 7, null]
Quadruple - [ETIQUETA L_IMPRIMIR, 2, null]
Quadruple - [PUNTEROSUBPROGRAMA T_2, 8, null]
Quadruple - [WRITESTRING T_0, L_0, null]
Quadruple - [MVP T_1, S, null]
Quadruple - [WRITEINT T_1, null, null]
Quadruple - [FINSUBPROGRAMA L_FIN_IMPRIMIR, 5, null]
Quadruple - [CADENA "SUMA:", L_0, null]

```
;Quadruple - [STARTGLOBAL null, null, null]
MOVE .SP, .IX
PUSH #-1
PUSH .IX
PUSH .SR
PUSH .IX
;Quadruple - [ETIQUETA L_TEST, 0, null]
L_TEST : NOP
MOVE .IX, /10000
;Quadruple - [VARGLOBAL A1, 0, null]
PUSH #0
;Quadruple - [VARGLOBAL SUMA, 0, null]
PUSH #0
;Quadruple - [VARGLOBAL B1, 0, null]
PUSH #0
;Quadruple - [PUNTEROGLOBAL T_6, 12, null]
SUB .IX, #12
MOVE .A, .SP
;Quadruple - [MV T_0, 2, null]
MOVE #2, #-7[.IX]
;Quadruple - [MVA T_1, A1, null]
SUB .IX, #4
MOVE .A, #-8[.IX]
;Quadruple - [STP T_1, T_0, null]
MOVE #-8[.IX], .R1
MOVE #-7[.IX], [.R1]
;Quadruple - [MV T_2, 3, null]
MOVE #3, #-9[.IX]
;Quadruple - [MVA T_3, B1, null]
SUB .IX, #6
MOVE .A, #-10[.IX]
;Quadruple - [STP T_3, T_2, null]
MOVE #-10[.IX], .R1
MOVE #-9[.IX], [.R1]
;Quadruple - [STARTSUBPROGRAMA null, null, null]
MOVE .SP, .R0
PUSH #-1
PUSH .R0
PUSH .SR
PUSH .IX
;Quadruple - [MVA T_4, B1, null]
SUB .IX, #6
MOVE .A, #-11[.IX]
;Quadruple - [PARAM T_4, null, null]
PUSH #-11[.IX]
```

```
;Quadruple - [MVA T_5, A1, null]
SUB .IX, #4
MOVE .A, #-12[.IX]
;Quadruple - [PARAM T_5, null, null]
PUSH #-12[.IX]
;Quadruple - [CALL L_SUMAR, null, null]
MOVE .R0, .IX
CALL /L_SUMAR
MOVE .IX, .SP
MOVE #-3[.IX], .R0
MOVE .R0, .IX
;Quadruple - [HALT null, null, null]
HALT
;Quadruple - [ETIQUETA L_SUMAR, 1, null]
L_SUMAR : NOP
MOVE .IX, /10001
;Quadruple - [VARSUBPROGRAMA S, 0, null]
PUSH #0
;Quadruple - [PUNTEROSUBPROGRAMA T_5, 15, null]
SUB .IX, #15
MOVE .A, .SP
;Quadruple - [MVP T_0, A, null]
MOVE #-4[.IX], .R1
MOVE [.R1], #-8[.IX]
;Quadruple - [MVP T_1, B, null]
MOVE #-5[.IX], .R1
MOVE [.R1], #-9[.IX]
;Quadruple - [ADD T_2, T_0, T_1]
ADD #-8[.IX], #-9[.IX]
MOVE .A , #-10[.IX]
;Quadruple - [MVA T_3, S, null]
SUB .IX, #7
MOVE .A, #-11[.IX]
;Quadruple - [STP T_3, T_2, null]
MOVE #-11[.IX], .R1
MOVE #-10[.IX], [.R1]
;Quadruple - [STARTSUBPROGRAMA null, null, null]
MOVE .SP, .R0
PUSH #-1
PUSH .R0
PUSH .SR
PUSH .IX
```

```
;Quadruple - [CALL L_IMPRIMIR, null, null]
MOVE .R0, .IX
CALL /L_IMPRIMIR
MOVE .IX, .SP
MOVE #-3[.IX], .R0
MOVE .R0, .IX
;Quadruple - [FINSUBPROGRAMA L_FIN_SUMAR, 7, null]
L_FIN_SUMAR : NOP
SUB .IX, #7
MOVE .A, .SP
RET
;Quadruple - [ETIQUETA L_IMPRIMIR, 2, null]
L_IMPRIMIR : NOP
MOVE .IX, /10002
;Quadruple - [PUNTEROSUBPROGRAMA T_2, 8, null]
SUB .IX, #8
MOVE .A, .SP
;Quadruple - [WRITESTRING T_0, L_0, null]
WRSTR /L_0
;Quadruple - [MVP T_1, S, null]
MOVE #10001, .R0
SUB [.R0], #7
MOVE [.A], #-6[.IX]
;Quadruple - [WRITEINT T_1, null, null]
WRINT #-6[.IX]
;Quadruple - [FINSUBPROGRAMA L_FIN_IMPRIMIR, 5, null]
L_FIN_IMPRIMIR : NOP
SUB .IX, #5
MOVE .A, .SP
RET
;Quadruple - [CADENA "SUMA:", L_0, null]
L_0 : DATA "SUMA: "
```