LENGUAJES DE PROGRAMACIÓN
Y
PROCESADORES DE LENGUAJES

Construcción de un compilador

`MenosC`

Parte-III: Generación de Código Intermedio

Material de prácticas

- `Makefile`. Una nueva versión.
- `principal.c`. Una nueva versión en el directorio **src**.
- `libgci` Librería para facilitar la tarea de generación de código intermedio.

  `libgci.h`, el fichero de cabecera, en el directorio **include**;

  `libgci.a`, la librería, en el directorio **lib**.
- `mvm`.- Máquina virtual que permite ejecutar el código intermedio `Malpas`

  Está disponible en el directorio `bin`.
- *Programas de prueba*.

Malpas: inventario de instrucciones

Operaciones aritméticas

OP	arg1	arg2	res	Significado
ESUM	I/P	I/P	P	Suma
EDIF	I/P	I/P	P	Resta
EMULT	I/P	I/P	P	Multiplicación
EDIVI	I/P	I/P	P	División entera
RESTO	I/P	I/P	P	Resto división entera
ESIG	I/P		P	Cambio de signo
EASIG	I/P		P	Asignación

Operaciones de salto

OP	arg1	arg2	res	Significado
GOTOS			E	Salto incondicional a E
EIGUAL	I/P	I/P	E	si arg1=arg2 salto a E
EDIST	I/P	I/P	E	si arg1<>arg2 salto a E
EMEN	I/P	I/P	E	si arg1<arg2 salto a E
EMAY	I/P	I/P	E	si arg1>arg2 salto a E
EMENEQ	I/P	I/P	E	si arg1<=arg2 salto a E
EMAYEQ	I/P	I/P	E	si arg1>=arg2 salto a E
FIN				Fin del programa

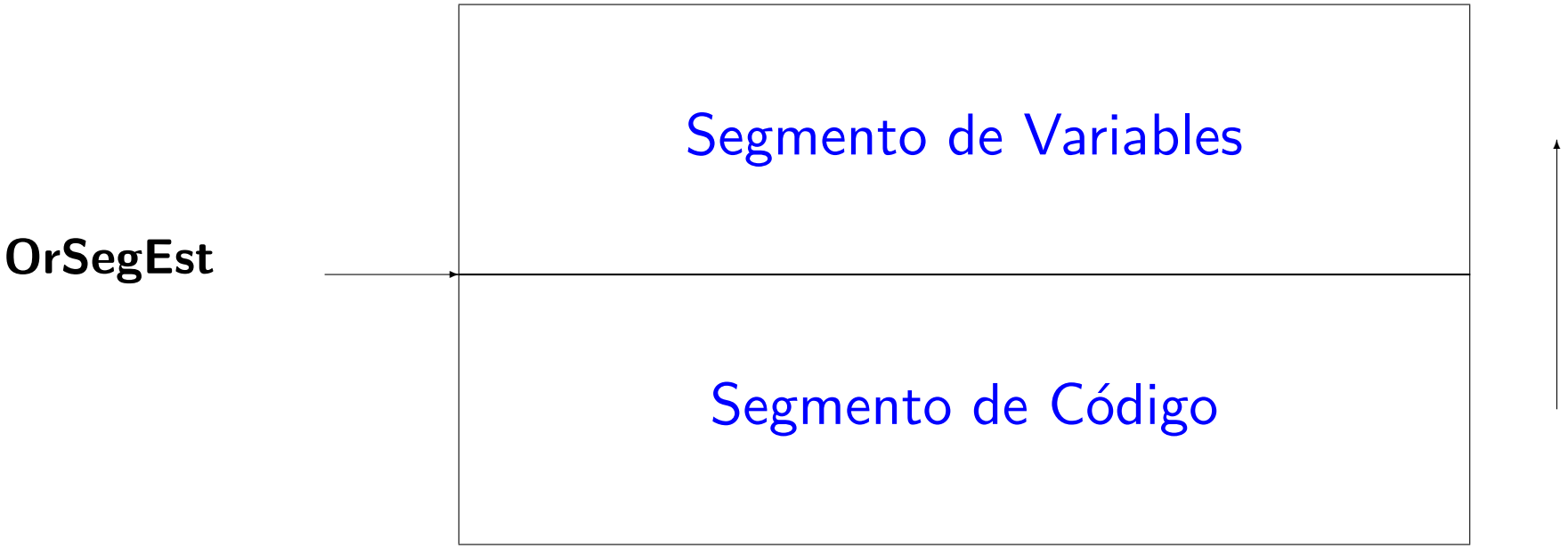
Operaciones de entrada/salida

OP	arg1	arg2	arg3	Significado
EREAD			P	Lectura
EWRITE			I/P	Escritura

Operaciones con direccionamiento relativo (vectores)

OP	arg1	arg2	res	Significado
EAV	P	I/P	P	Asigna un elemento de un vector a una variable: $res := arg1[arg2]$
EVA	P	I/P	P	Asigna una variable a un elemento de un vector: $arg1[arg2] := res$

Malpas: arquitectura



➢ Estructura de la librería `libgci`

Constantes, variables globales y estructuras básicas (ver Sección 9.1 del Enunciado)

➢ Funciones de para la GCI

```
Funciones generales
void emite (int cop, TIPO_ARG arg1, TIPO_ARG arg2, TIPO_ARG res);
int creaVarTemp ();
void vuelcaCodigo (char *nom);

Funciones para crear los argumentos de las instrucciones
TIPO_ARG crArgNul ();
TIPO_ARG crArgEnt (int valor);
TIPO_ARG crArgEtq (int valor);
TIPO_ARG crArgPos (int valor);

Funciones para la manipulación de las LANS
int creaLans (int d);
int fusionaLans (int x, int y);
void completaLans (int x, TIPO_ARG arg);
```

EJEMPLO DE GENERACIÓN DE CÓDIGO

```
operadorAditivo
  : MAS_          { $$ = ESUM; }
  | MENOS_        { $$ = EDIF; } ;
expresionAditiva
  : expresionMultiplicativa          { $$ = $1; }
  | expresionAditiva operadorAditivo expresionMultiplicativa
    {
      $$.tipo = T_ERROR;
      if ($1.tipo == $3.tipo == T_ENTERO) $$.tipo = T_ENTERO;
      else yyerror("Error de tipos en la 'expresión aditiva'");

      $$.pos = creaVarTemp();
      /*************** Expresión a partir de un operador aritmético */
      emite($2, crArgPos($1.pos), crArgPos($3.pos), crArgPos($$.pos));
    } ;
```

➢ Ejemplo de programa de código intermedio

```
// Calcula el factorial de un número > 0 y < 13
{ int n; int fac; int i; bool f;

  f = true; fac = 1;
  while ( f ) {
    read(n);
    if ((n > 0) && (n < 13)) {
      i = 2;
      while (i <= n) { fac = fac * i; i++; }
      print(fac); f = false;
    }
    else {}
  }
}
```

EJEMPLO DE GCI

0	EASIG	i: 1	,		, p: 4
1	EASIG	p: 4	,		, p: 3
2	EASIG	p: 3	,		, p: 5
3	EASIG	i: 1	,		, p: 6
4	EASIG	p: 6	,		, p: 1
5	EASIG	p: 1	,		, p: 7
6	EASIG	p: 3	,		, p: 8
7	EIGUAL	p: 8	, i: 0	, e: 45
8	EREAD		,		, p: 0
9	EASIG	p: 0	,		, p: 9
10	EASIG	i: 0	,		, p: 10
11	EASIG	i: 1	,		, p: 11
12	EMAY	p: 9	, p: 10	, e: 14
13	EASIG	i: 0	,		, p: 11
14	EASIG	p: 0	,		, p: 12
15	EASIG	i: 13	,		, p: 13
16	EASIG	i: 1	,		, p: 14
17	EMEN	p: 12	, p: 13	, e: 19
18	EASIG	i: 0	,		, p: 14
19	EMULT	p: 11	, p: 14	, p: 15
20	EIGUAL	p: 15	, i: 0	, e: 44
21	EASIG	i: 2	,		, p: 16
22	EASIG	p: 16	,		, p: 2

23	EASIG	p: 2	,		, p: 17
24	EASIG	p: 2	,		, p: 18
25	EASIG	p: 0	,		, p: 19
26	EASIG	i: 1	,		, p: 20
27	EMENEQ	p: 18	, p: 19	, e: 29
28	EASIG	i: 0	,		, p: 20
29	EIGUAL	p: 20	, i: 0	, e: 38
30	EASIG	p: 1	,		, p: 21
31	EASIG	p: 2	,		, p: 22
32	EMULT	p: 21	, p: 22	, p: 23
33	EASIG	p: 23	,		, p: 1
34	EASIG	p: 1	,		, p: 24
35	EASIG	p: 2	,		, p: 25
36	ESUM	p: 2	, i: 1	, p: 2
37	GOTOS		,		, e: 24
38	EASIG	p: 1	,		, p: 26
39	EWRITE		,		, p: 26
40	EASIG	i: 0	,		, p: 27
41	EASIG	p: 27	,		, p: 3
42	EASIG	p: 3	,		, p: 28
43	GOTOS		,		, e: 44
44	GOTOS		,		, e: 6
45	FIN		,		,