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* En el directorio **tmp**

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: inventario de instrucciones

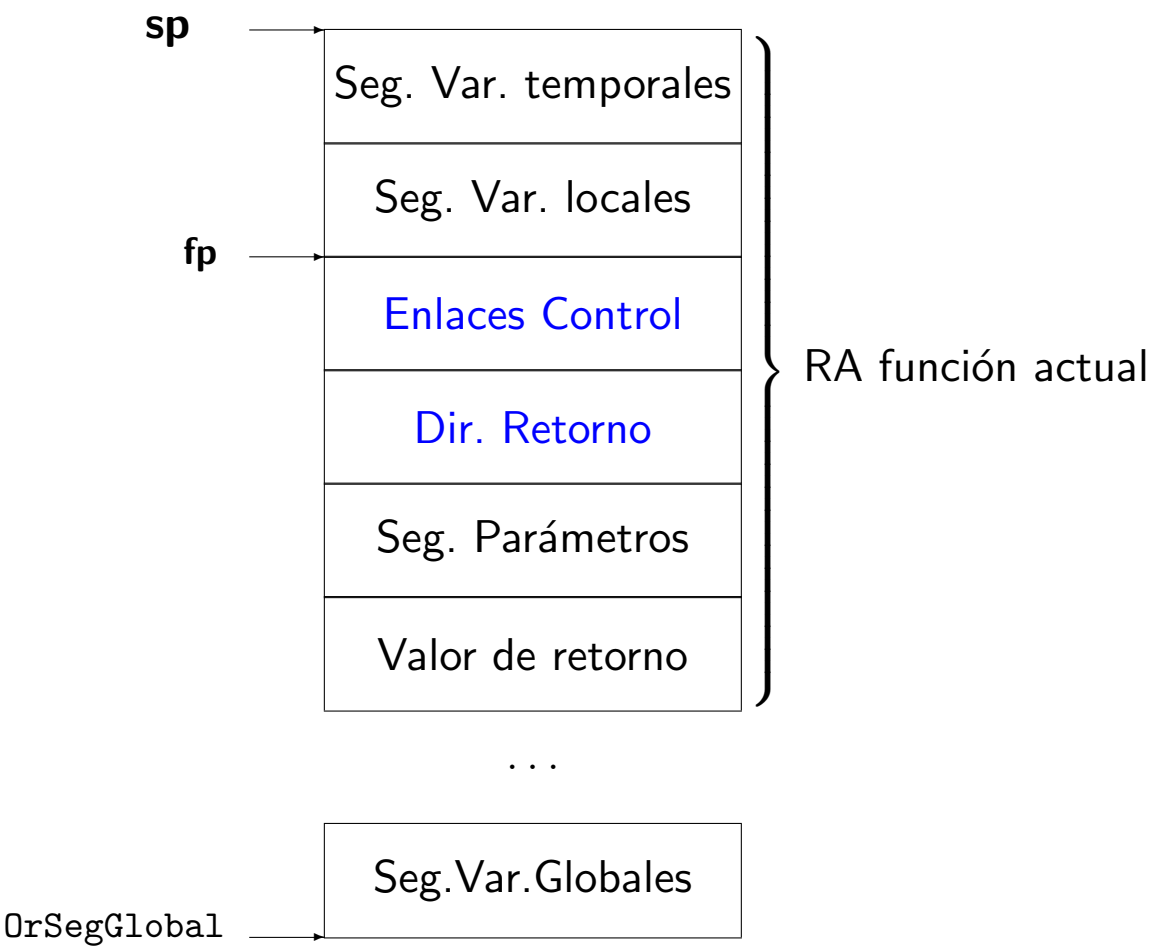
Operaciones de llamada

OP	arg1	arg2	res	Significado
FIN				Fin del programa
RET				Desapila la dirección de retorno y transfiere el control a dicha dirección
CALL			E	Apila la dirección de retorno y transfiere el control a res

Operaciones de manejo de pila de RA

OP	arg1	arg2	res	Significado
EPUSH			I/P	Apila res en la cima de la pila
EPOP			P	Desapila la cima de la pila y la deposita en res
PUSHFP				Apila el FP en la cima de la pila
FPPOP				Desapila la cima y la deposita en el FP
FPTOP				El FP apunta a la misma posición que la cima de la pila
TOPFP				La cima de la pila apunta a la misma posición que el FP
INCTOP			I	Incrementa la cima de la pila en res posiciones
DECTOP			I	Decrementa la cima de la pila en res posiciones

Malpas: arquitectura



➢ Estructura de la librería `libgci`

Constantes, variables globales y estructuras básicas (ver Sección `10.1` 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 n, 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'");

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

➢ Ejemplo de programa de código intermedio

```
// Calcula el factorial de un múmero > 0 y < 13
//---------------------------------------
int factorial (int n)
{ int f;
  if (n <= 1) f=1;
  else f= n * factorial(n-1);
  return f;
}
int main ()
{ int x;
  read(x);
  if (x > 0)
    if (x < 13) print(factorial(x));
    else {}
  else {}
  return 0;
}
```

EJEMPLO DE GCI

0	INCTOP		,		, i: 0
1	GOTOS		,		, e: 30
2	PUSHFP		,		,
3	FPTOP		,		,
4	INCTOP		,		, i: 12
5	EASIG	p: (1, -3)	,		, p: (1, 1)
6	EASIG	i: 1	,		, p: (1, 2)
7	EASIG	i: 1	,		, p: (1, 3)
8	EMENEQ	p: (1, 1)	, p: (1, 2)	, e: 10
9	EASIG	i: 0	,		, p: (1, 3)
10	EIGUAL	p: (1, 3)	, i: 0	, e: 14
11	EASIG	i: 1	,		, p: (1, 4)
12	EASIG	p: (1, 4)	,		, p: (1, 0)
13	GOTOS		,		, e: 25
14	EASIG	p: (1, -3)	,		, p: (1, 5)
15	EPUSH		,		, i: 0
16	EASIG	p: (1, -3)	,		, p: (1, 6)
17	EASIG	i: 1	,		, p: (1, 7)
18	EDIF	p: (1, 6)	, p: (1, 7)	, p: (1, 8)
19	EPUSH		,		, p: (1, 8)
20	CALL		,		, e: 2
21	DECTOP		,		, i: 1
22	EPOP		,		, p: (1, 9)
23	EMULT	p: (1, 5)	, p: (1, 9)	, p: (1, 10)
24	EASIG	p: (1, 10)	,		, p: (1, 0)
25	EASIG	p: (1, 0)	,		, p: (1, 11)
26	EASIG	p: (1, 11)	,		, p: (1, -4)
27	TOPFP		,		,
28	FPPOP		,		,
29	RET		,		,

30	PUSHFP		,		,
31	FPTOP		,		,
32	INCTOP		,		, i: 10
33	EREAD		,		, p: (1, 0)
34	EASIG	p: (1, 0)	,		, p: (1, 1)
35	EASIG	i: 0	,		, p: (1, 2)
36	EASIG	i: 1	,		, p: (1, 3)
37	EMAY	p: (1, 1)	, p: (1, 2)	, e: 39
38	EASIG	i: 0	,		, p: (1, 3)
39	EIGUAL	p: (1, 3)	, i: 0	, e: 55
40	EASIG	p: (1, 0)	,		, p: (1, 4)
41	EASIG	i: 13	,		, p: (1, 5)
42	EASIG	i: 1	,		, p: (1, 6)
43	EMEN	p: (1, 4)	, p: (1, 5)	, e: 45
44	EASIG	i: 0	,		, p: (1, 6)
45	EIGUAL	p: (1, 6)	, i: 0	, e: 54
46	EPUSH		,		, i: 0
47	EASIG	p: (1, 0)	,		, p: (1, 7)
48	EPUSH		,		, p: (1, 7)
49	CALL		,		, e: 2
50	DECTOP		,		, i: 1
51	EPOP		,		, p: (1, 8)
52	EWRITE		,		, p: (1, 8)
53	GOTOS		,		, e: 54
54	GOTOS		,		, e: 55
55	EASIG	i: 0	,		, p: (1, 9)
56	EASIG	p: (1, 9)	,		, p: (1, -3)
57	TOPFP		,		,
58	FPPOP		,		,
59	FIN		,		,