Procesadores de lenguaje

Tema 7 – Generación de código intermedio

Salvador Sánchez, Daniel Rodríguez
Departamento de Ciencias de la Computación
Universidad de Alcalá

Resumen

- Representaciones intermedias
  - Código de 3 direcciones
  - Cuádruplas
  - Tripletas

Representaciones intermedias

- Una representación intermedia es una estructura de datos que representa al programa fuente durante el proceso de su traducción a código objeto.
- Debería reunir las siguientes características :
  1. Parecerse al código objeto final.
  2. Ser fácil de generar a partir del árbol sintáctico.
  3. Ser independiente del entorno de ejecución.

Código de tres direcciones

- Las instrucciones utilizan tres direcciones de memoria o registros: dos para los operandos y una para el resultado.
- Normalmente las operaciones se representan de la manera siguiente:

```
resultado = operando1 operador operando2
```

- Es la representación intermedia más extendida.

Código de tres direcciones: Ejemplos

`(a + b) / (c + e)`	`z: = x + y + z / y`
↓	↓

`t1 := a + b`
`t2 := c + e`
`t3 := t1 / t2`
	`t1 := x + y`
	`t2 := t1 + z`
	`t3 := t2 / y`
	`z := t3`

Código de tres direcciones

Instrucción	Explicación	Ejemplo
Operaciones binarias	x = y op z, en la que op es un operador binario (aritmético, lógico o relacional).	a = b and c
Operaciones unarias	x = op y, en la que op es un operador unario (negación lógica, menos unario, operadores de desplazamiento o conversión de tipos, etc.).	a = – c
Asignaciones	x = y, operación de copia en la que el valor de y se asigna a x.	a = c
Definir punto Ir al punto	label etiq, define una etiqueta. goto etiq, salto incondicional.	label T2 goto T2
Saltos condicionales	Saltos condicionales como if false x goto etiq.	if x = 0 goto T2
Ir y volver de un bloque	call f, para llamar funciones o acciones, y return, para retornar valores.	call etiqueta
Matrices (arrays) y tablas	Asignaciones con índices de la forma x = y[i], en la que se asigna a x el valor de la posición de memoria situado i unidades más allá de la posición y. También x[i] = y.	x[3] = 5
Registros	Registros Tratamiento de los campos con registro.camp, en los que el campo tiene el valor guardado a una determinada distancia del comienzo del registro.	x = y.dia
Indirecciones	Asignaciones de direcciones y apuntadores de la forma x = &y, x = *y o *x = y.	*x = 6

Código de tres direcciones

```
read x;
if 0 < x then
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1;
  until x = 0;
write fact;
end;
```

```
01 read x
02 t1 = 0 < x
03 if false t1 goto L1
04 fact = 1
05 label L2
06 t2 = fact * x
07 fact = t2
08 t3 = x - 1
09 x = t3
10 t4 = x == 0
11 if false t4 goto L2
12 write fact
13 label L1
14 halt
```


Cuádruplas

- Una cuádrupla es una estructura de tipo registro con cuatro campos: op, result, arg1 y arg2.
  - Por ejemplo, la proposición de tres direcciones $x = y + z$ se representaría como: *(suma, x, y, z)*.
- En general, las instrucciones que no requieren todos los campos dejan vacíos los que no utilizan:
  - Ej: Las proposiciones con operadores unarios no utilizan *arg2*

Tripletas

- Como las cuádruplas pero utilizando sólo 3 campos: (op, arg1, arg2)
- Para evaluar una expresión compleja, hay que dividirla en una secuencia equivalente de expresiones simples.
- Ejemplo: evaluar `x = y + 2` requiere dos instrucciones:
  (suma, y, 2)
  (asigna, x, <resultado_de_la_suma_anterior>)

Tripletas

- Para evitar introducir identificadores temporales en la tabla de símbolos, se hace referencia a un valor temporal según la posición (línea) de la proposición que lo calcula.
- Originalmente, el resultado de una tripleta anterior se asignaría a una variable temporal creada por el compilador:
  - tmp1 = (suma, y, 2)
  - (asigna, x, tmp1)
- Pero es posible evitar identificadores temporales, usando las propias instrucciones para representan el valor del nombre temporal, eliminando el campo "tmp1" en el que se guardaba temporalmente el resultado (por innecesario):
  - **(1)** (suma, y, 2)
  - (2) (asigna, x, **(1)**)

Comparación tripletas y cuádruplas

- La diferencia en la implementación es cuestión de mayor o menor indirección:
  - La de tripletas necesita menos espacio y el compilador no genera identificadores temporales, pero cambiar de sitio una proposición que defina un valor temporal exige modificar todas las referencias a ella.
  - Esto representa un inconveniente a la hora de optimizar el código, ya que en cada paso del proceso de optimización hay que cambiar constantemente las proposiciones de lugar.
- En la práctica se utiliza la notación de cuádruplas como implementación del código de 3 direcciones

Ejemplo - cuádruplas vs. tripletas

```
read x;
if 0 < x then
fact := 1;
repeat
fact := fact * x;
x := x - 1;
until x = 0;
write fact;
end;
```

```
(read,x,-,-)
(isbigger,t1,x,0)
(if_false,t1,L1,-)
(assign,fact,1,-)
(label,L2,-,)
(mult,t2,fact,x)
(assign,fact,t2,-)
(sub,t3,x,1)
(assign,x,t3,-)
(isequal,t4,x,0)
(if_false,t4,L2,-)
(write,fact,-,-)
(label,L1,-,-)
(halt,-,-,-)
```

```
(0) (read,x,-)
(1) (isbigger,x,0)
(2) (if_false,(1),(11))
(3) (assign,fact,1)
(4) (mult,fact,x)
(5) (assign, fact, (4))
(6) (sub,x,1)
(7) (assign, x, (6))
(8) (isequal,x,0)
(9) (if_false,(8),(4))
(10) (write,fact,-)
(11) (halt,-,-)
```


Bibliografía

- *Básica:*
  - *Construcción de compiladores. Principios y práctica.* Kenneth C. Louden. Thomson-Paraninfo. 2004.
  - *Compiladores: principios, técnicas y herramientas*. A.V. Aho, R. Sethi, J.D. Ullman. Addison-Wesley Iberoamerica. 1990.