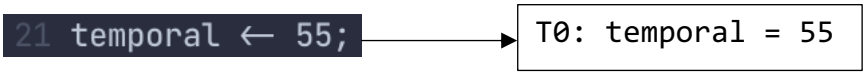
Marco Fuentes – 18188
Andy Castillo - 18040

Diseño de representación intermedia

Instrucciones

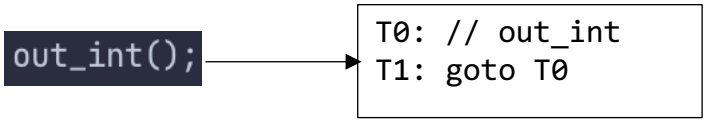
Las formas de las instrucciones que se aceptarán son:

1. Instrucciones de copia de la forma $x = y$.
   En donde:
   - x, y son direcciones de memoria.
   - y puede ser una **CONSTANTE.**
   - El valor de y será asignado a la dirección de x.
   - Una expresión de copia en YAPL se traduciría a la siguiente representación intermedia:



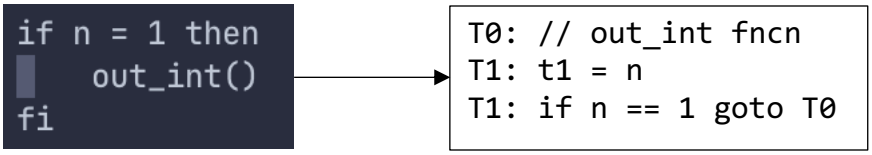
2. Instrucciones de asignación de la forma $x = y\ op\ z$.
   En donde
   - x, y, z son direcciones de memoria
   - op es un operador aritmético o lógico
   - y puede no estar presente, en casos de operaciones unarias tales como la negación de un elemento booleano o un número negativo.
   - Una expresión de asignación en YAPL se traduciría a la siguiente en representación intermedia:

3. Saltos incondicionales del tipo $goto\ L$
   En donde:
   - L representa una instrucción previamente definida en la representación intermedia que será ejecutada.
   - Una expresión de este tipo en YAPL se traduce a la siguiente representación intermedia



4. Saltos condicionales del tipo $if\ x\ comp\ y\ goto\ L$
   En donde:
   - x, y representan direcciones de memoria
   - L representa una instrucción previamente definida en la representación intermedia que será ejecutada condicionalmente.
   - $comp$ representa un elemento comparativo, como $\leq, \geq, ==, != =$

- Una expresión de este tipo en YAPL se traduce a la siguiente representación intermedia



5. Paso de parámetros del tipo $param\ x_1$
   En donde:
   - x_1 es un parámetro de una función que será llamada en el futuro
   - Una expresión de este tipo en YAPL se traduce a la siguiente representación intermedia

6. Llamadas a funciones previamente definidas del tipo $call\ p, n$
   En donde:
   - p es el nombre de una función
   - n es el número de parámetros que le pasaremos a dicha función
   - Una expresión de este tipo en YAPL se traduce a la siguiente representación intermedia

Tipos de datos básicos

Los tipos de datos que se aceptarán son lo siguientes:

1. `Int`: 8B
2. `Bool`: 1b
3. `String`: 8b por carácter escrito.

Estructura de datos

En vista de optimizar el espacio en memoria, se usará una estructura de tripletas para almacenar las representaciones intermedias de las instrucciones de código escrito en el programa fuente. Si bien es cierto que estructuras como la cuarteta ofrecen un mejor rendimiento al pasarle menos trabajo al procesador, hemos encontrado que los procesadores actuales son suficientemente robustos y rápidos como para soportar esta carga.