Instituto Tecnológico y de Estudios Superiores de Monterrey

TC3002B.202

Actividad Final - Mini Proyecto Parte 4

Integrantes:

Fernando Doddoli Lankenau - A00827038

Periodo Feb-Jun 2023

Pasos para Correr Programa

1. Abrir directorio donde se encuentra el proyecto en la terminal.
2. Correr: python3 virtual_machine.py
3. Ingresar la prueba deseada

Documentación Técnica

Estructuras de Datos : Lexer y Parser

Estructura de Datos	Descripción
Tabla de Variables	Diccionario con el nombre como llave y dirección como valor.
Tabla de Constantes	Diccionario con la constante como llave y dirección como valor.
Tabla de Operadores	Diccionario con el operador como llave y dirección como valor.
Tabla de Direcciones	Diccionario con el objetivo de mantener un contador para las diferentes variables vistas en el programa. La llave es el tipo: int, float, bool, cte_i, cte_f, y el valor es la dirección. Las dirección de los diferentes tipos se dividen en los siguientes rangos: int: 1000 a 1999 float: 2000 a 2999 bool: 3000 a 3999 cte_i: 4000 a 4999 cte_f: 5000 a 5999
Pila de Operandos	Pila que incluye las direcciones de los operandos.
Pila de Operadores	Pila que incluye las direcciones de los operadores.
Pila de Tipos	Pila que incluye las direcciones de los tipos.
Pila de Jumps	Pila que incluye los números de quadruplos a los que se brincará cuando se creen los quadruplos de GotoF, Goto, GotoT.
Fila de Quadruplos	Lista de quadruplos, donde cada quadruplo es una tupla que tiene cuatro valores: la

	dirección del operador, la dirección del operando izquierdo, la dirección del operando derecho, y la dirección del resultado resultado.

Estructuras de Datos : Virtual Machine

Estructura de Datos	Descripción
Tabla de Variables	Diccionario con la dirección como llave y el valor como nombre.
Tabla de Constantes	Diccionario con la dirección como llave y la constante como valor.

Nota: Se invierten las llaves y los valores en los diccionarios para poder hacer referencia a esas direcciones, ya que en la máquina virtual trabajamos con los quadruplos, que son en direcciones.

Principales Algoritmos Desarrollados.

A continuación se mencionan los algoritmos más complejox del programa, cabe resaltar que no representa la totalidad de las funciones del mismo.

Función fill

La función de fill se utiliza para llenar los quadruplos de Goto, GotoF, y GotoT. recibe como parámetros el numero del cuádruplo al cual pertenece, y el contador de quádruplos actual. Como no podemos editar directamente los valores de una tupla, transformamos la tupla del cuádruplo a una lista, después editamos el valor que queremos utilizando su indice, transofrmamos la lista a una tupla y cambiamos la tupla original por la nueva.

Funciones p_seen_factor y p_seen_termino y p_seen_expresion_bool

Estos algoritmos son muy similares, la única diferencia es que p_seen_factor se llama en la gramática de término, en el punto neurálgico al salir de factor, significando que se hizo una multiplicación y división. Mientras que p_seen_termino se llama en la gramática de exp, en el punto neurálgico al salir de termino, significando que se hizo una suma o resta. Y p_seen_expresion_bool se llama en la gramática de expresion, en el punto neurálgico al terminar de hacer una comparación.

Consequentemente, los tres algoritmos tienen una condicional que sólo permite entrar cuando se haya hecho una multiplicación o división (p_seen_factor); o una suma o resta

(p_seen_termino); o una comparación con una condicional mayor que, menor que, y menor mayor que (p_seen_expresion_bool). Una vez cumplida esta condición, entonces se sacan de las pilas los operandos (derecho y izquierdo) y sus respectivos tipos. Después, se llama a la tabla de consideraciones para saber cual sería el tipo de dato de la operación y el resultado se almacena en result_type. Si el result_type es válido, entonces se obtiene una nueva dirección en base a ese tipo y este valor se guarda en result. Finalmente, generamos el cuádruplo con el operador, el operando izquierdo, el operando derecho, y el resultado.

Función p_seen_const_var_id

Esta función sirve cuando el programa ve una constante de variable. Busca en la tabla de variables su dirección y la ingresa a la pila de operandos y tipos.

Función p_seen_const_var_cte_i y p_seen_const_var_cte_f

Estas funciones sirven para cuando el programa ve una constante entera o flotante, respectivamente. La función primero checa si la constante ya está en la tabla de constantes, si sí está entonces obtiene su dirección y la mete a la pila de operandos.

En caso contrario, para constante entereas, obtenemos la dirección utilizando la tabla_de_direcciones y la agregamos a la tabla_de_constantes–donde la llave es la constante y el valor es la dirección. Adicionalmente, le agregamos la misma dirección a la pila de operandos. Finalmente, ya fuera del if y else, ingresar la dirección a la pila de tipos.

En caso contrario, para constantes flotantes, hacemos el mismo procedimiento anterior pero envés de buscar por la dirección en la tabla_de_direcciones utilizando cte_i, utilizamos cte_f para decir que lo que buscamos es la dirección de una constante.

Función p_seen_punto_comma_asigna

Esta función se llama en el punto neurálgico al terminar una asignación con el objetivo de generar un nuevo quadruplo. Obtiene el la dirección del operador de la pila de operadores, la dirección que almancerará lo asignado de la pila de operandos, y el valor que se está asignando también de la pila de operandos. Posteriormente genera el quadruplo y actualiza el contador global de quadruplos.

Función p_seen_expresion

Esta función se llama en el punto neurálgico de condición, después de ver una expresión. Primero checa si el resultado de la expresión es booleana, si no es entonces tira un error. En lo contrario procede y genera un quadruplo sacando la dirección del resultado de la pila de operandos y agregando un Goto en falso. Finalmente, agrega a la pila de jumps el valor del contador global que representa el número del quadruplo actual.

Función p_seen_sino

Esta función se llama en el punto neurálgico de condición, después de ver un sino. Genera un quadruplo agregando un Goto en falso y el resto en None. Finalmente, agrega a la pila de jumps el valor del contador global que representa el número del quadruplo actual y llama a la función de fill.

Función p_seen_punto_comma

Esta función se llama en el punto neurálgico de condición, después de ver el punto y comma al final de la condición. Lo que hace es obtener de la pila de jumps el valor del cuádruplo donde agregamos el GotoF o Goto; y posteriormente llamar a la función de fill con el valor anterior y el valor del contador global del numero de quadruplos que hay. El objetivo de esto es poder modificar el cuádruplo generado de GotoF o Goto con el numero del quadruplo al cual tiene que dirigirse dada la condición.

Función p_seen_escritura_exp

Esta función se llama dentro de la gramática de escritura en el punto neururálgico después de ver una expresión. Genera un quadruplo con la dirección de referencia a “print”, y la dirección del operando, que en este caso es el valor resultante de la expresión. Finalmente, actualiza al contador.

Conjunto de Pruebas para Probar el Programa

Asignacion

Prueba	Resultado Esperado
prog id; var int: id2, id3, id4; [id2 = (5 + 3) * 5;] end	tabla_de_memoria_global = {1000: 40}
prog id; var int: id2, id3; [id2 = (5 + 3) / 2;] end	tabla_de_memoria_global = {1000: 4.0}
prog id; var int: id2, id3; [id2 = (5 - 3);] end	tabla_de_memoria_global = {1000: 2}
prog id; var int: id2; [id2 = 5;] end	tabla_de_memoria_global = {1000: 5}
prog id; var int: id1, id2; [id1 = 5; id2 = id1 * 4;] end	tabla_de_memoria_global = {1000: 5, 1001: 20}

Condicion

Prueba	Resultado Esperado

prog id; var int: id2; [si (5 > 3) [id2 = 200;] sino [id2 = 100;];] end	tabla_de_memoria_global = {1000: 200}
prog id; var int: id2; [si (5 < 3) [id2 = 200;];] end	tabla_de_memoria_global = {}
prog id; var int: id2; [si (5 < 3) [id2 = 200;] sino [id2 = 3;];] end	tabla_de_memoria_global = {1000: 3}
prog id; var int: id2; [si (5 <> 3) [id2 = 200;] sino [id2 = 3;];] end	tabla_de_memoria_global = {1000: 200}

Ciclo

Prueba	Resultado Esperado
prog id; var int: id2; [id2 = 0; mientras (id2 < 5) [print(id2); id2 = id2 + 1;];] end	0 1 2 3 4 tabla_de_memoria_global = {1000: 5}
prog id; var int: id2; [id2 = 8; mientras (id2 > 5) [print(id2); id2 = id2 - 1;];] end	8 7 6 tabla_de_memoria_global = {1000: 5}

Escritura

Prueba	Resultado Esperado
prog id; var int: id2; [print("Hello World");] end	"Hello World"
prog id; var int: id2; [print(3 + 5);] end	8
prog id; var int: id2; [print("Hello World", 3 + 5);] end	"Hello World" 8

Fibonacci

Prueba	Resultado Esperado

prog id; var int: id1, id2, id3; [id1 = 1; id2 = 1; id3 = 0; mientras (id3 < 5) [id3 = id1 + id2; id1 = id2; id2 = id3; print(id3);];] end	2 3 5 tabla_de_memoria_global = {1000: 3, 1001: 5, 1002: 5}