Generación de Código y Ejecución en Máquina Virtual (Etapa 3)

Fabián Fernández Kevin Jiménez Justin Martínez

Compiladores e Intérpretes
Tecnológico de Costa Rica
Profesor: Oscar Mario Víquez Acuña
II Semestre 2024

Contents

1 Introducción

En esta tercera etapa del proyecto, se aborda la generación de código intermedio en formato de bytecode para su ejecución en una máquina virtual desarrollada en C#. Este proceso incluye la definición de las instrucciones necesarias, la implementación de plantillas para generar código a partir del árbol de sintaxis abstracta (AST), y el diseño de una máquina virtual para interpretar y ejecutar dicho código.

A pesar de las limitaciones en el desarrollo, esta etapa sigue las directrices establecidas para proporcionar una solución básica que ilustre los principios fundamentales de la generación de código y su ejecución.

2 Generación de Código

2.1 Requerimientos

La generación de código se basa en recorrer el AST utilizando el patrón `Visitor`. Cada nodo relevante tiene una plantilla asociada que define cómo traducir las construcciones del lenguaje Mini-Python en bytecode comprensible para la máquina virtual.

Las instrucciones implementadas incluyen:

- `LOAD_CONST`: Coloca una constante en el tope de la pila.
- `LOAD_FAST`: Carga el valor de una variable local.
- `STORE_FAST`: Guarda el valor del tope de la pila en una variable.
- `CALL_FUNCTION`: Llama a una función.
- `RETURN_VALUE`: Retorna un valor desde una función.

2.2 Plantillas de Generación

Se desarrollaron plantillas básicas para las siguientes construcciones:

- Declaraciones de variables.
- Operaciones aritméticas.
- Llamadas a funciones.

Por ejemplo, la generación de código para una asignación se realiza mediante la plantilla:

```
assignStatement: ID ASSIGN expression
    -> STORE_FAST <ID>
```

3 Máquina Virtual

3.1 Componentes

La máquina virtual está implementada en C# y se compone de los siguientes módulos:

- **Pila**: Almacena los valores temporales necesarios para las operaciones.
- **Almacén**: Maneja variables globales y locales.
- **Intérprete**: Procesa las instrucciones de bytecode y ejecuta el programa.

3.2 Ejecución de Bytecode

El bytecode generado se ejecuta mediante un programa principal que recibe el archivo de instrucciones y utiliza la máquina virtual para interpretarlo. Un ejemplo de ejecución básica incluye:

```
LOAD_CONST 5
STORE_FAST x
LOAD_FAST x
CALL_FUNCTION print
```

4 Resultados Obtenidos

A pesar de las limitaciones en el tiempo de desarrollo, se lograron los siguientes avances:

Funcionalidad	**Estado**
Generación de Bytecode	Parcialmente Implementado
Máquina Virtual	Funcional Básica
Ejecución de Programas Simples	Implementado
Manejo de Errores en Ejecución	No Implementado

5 Conclusiones

Esta etapa representa un acercamiento inicial a la generación y ejecución de código en el proyecto del compilador Mini-Python. Aunque no se logró una implementación completa, las bases desarrolladas son un punto de partida para futuras mejoras. La experiencia obtenida resalta la importancia de planificar adecuadamente las etapas y gestionar el tiempo disponible.

6 Bibliografía

References

[1] Terrence Parr. *The Definitive ANTLR 4 Reference*. 2013.

[2] MSDN. *C# Programming Guide*. `https://learn.microsoft.com/`.