Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Кафедра «Информатики и защиты информации»

Курсовая работа на тему:

Разработка компилятора подмножества процедурного языка в ассемблер

Специальность: 10.05.04 – Информационно-аналитические системы безопасности

ЕМЕЛЬЯНОВ Алексей Сергеевич,
ст. гр. ИСБ-118

г. Владимир 2021

Описание компилятора

Компилятор реализован на языке Java с использованием библиотек ANTLR и ASM. Трансляция производится в байт-код JVM.

ANTLR - мощный генератор парсеров. На основе грамматики ANTLR генерирует синтаксический анализатор, который может строить и обходить синтаксические деревья.

ASM - это универсальная среда обработки и анализа байт-кода Java. Её можно использовать для динамического создания классов непосредственно в двоичной форме. ASM предоставляет несколько распространенных преобразований байт-кода. ASM предлагает те же функции, что и другие фреймворки байт-кода Java, но ориентирован на производительность.

Грамматика

```
compileUnit
    : function*? main EOF
    ;

main
    : MAIN block
    ;

block
    : '{' (statement)* '}'
    ;
blockFunc
    :'{' (statement)* returnFunc ';' '}'
    ;


function
    : DEF types funcName '(' argsList ')' blockFunc
    ;

returnFunc
    : RETURN varName
    ;
```

Грамматика строится на основе вложенных конструкций – мы объединяем инструкции в блоки и помещаем эти блоки внутри конструкций, которые могут содержать в себе эти инструкции. Из таких блоков выстраивается шаблон, при совпадении с которым входящий текст заполняет собой дерево, генерируемое ANTLR на основе этой грамматики.

Грамматика

```
def int func (int a, float b)
{
    return a;
}

__main__()
{
    int out = func(1,2.2);
}
```

Представление входного потока в виде дерева. Вывод производится с помощью встроенного функционала ANTLR.

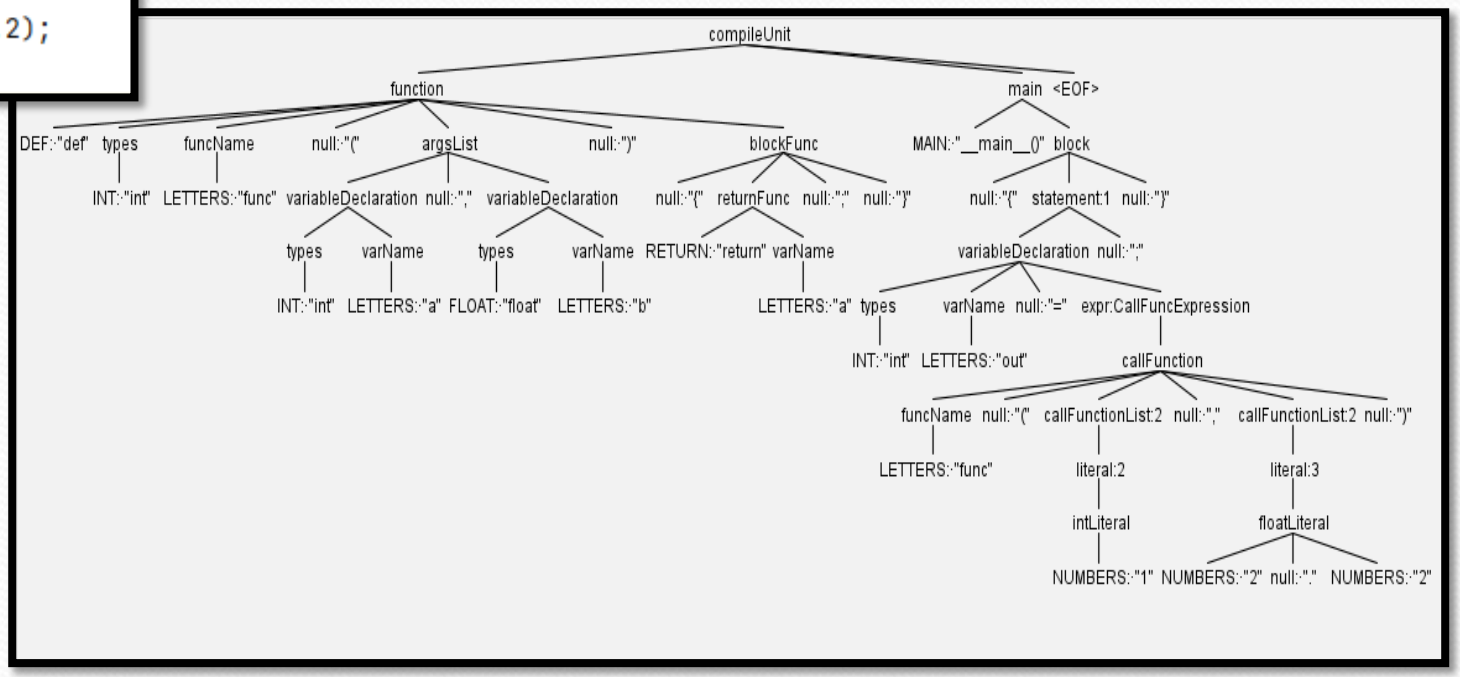

Таблица символов

Таблица символов представляет собой объект класса HashMap, в котором ключом является имя переменной, а значением её тип.

```
@Override
public Node visitVariableDeclaration(LangSiParser.VariableDeclarationContext ctx) {
    String varName = ctx.varName().getText();
    Types p = Types.getType(ctx.types().getText());
    if (p == null) error( s: ctx.start.getLine() + ": " + varName + " null type exception");
    Id id = new Id(varName, p, used++);
    top.put(varName, id);
    if (ctx.expr() != null) {
        Expr x = (Expr) visit(ctx.expr());
        return new Declare(id, x);
    }
    return new Declare(id);
}
```

Трансляция в целевой код

```
public While (Expr x, Stmt s) {expr = x; stmt = s;}

public void genJVM() {
    startCycle = new Label();
    statements = new Label();
    endCycle = new Label();
    VisitImplement.start = startCycle;
    VisitImplement.end = endCycle;
    Compile.mv.visitLabel(startCycle);
    expr.genJVM();
    Compile.mv.visitJumpInsn(Opcodes.IFEQ, endCycle);
    Compile.mv.visitLabel(statements);
    stmt.genJVM();
    Compile.mv.visitJumpInsn(Opcodes.GOTO, startCycle);
    Compile.mv.visitLabel(endCycle);

}
```

Библиотека ASM позволяет удобно транслировать инструкции напрямую в исполняемый файл.
При помощи команд:

mv.visit…(код из таблицы JVM);

мы можем записывать в выходной поток нужные инструкции при обходе дерева.

Пример работы компилятора

sqrt.ls

```
/*
    Программа для нахождения квадратного корня
    Алгоритм: Bakhsali Approximation
*/

def float sqrt (int num)
{
    int i = 0;
    int e = i * i;
    while ( e <= num)
    {
        i = i + 1;
        e = i * i;
    }
    i = i - 1;
    float d = num - i * i;
    float t = i * 2;
    float p = d / t;
    float a = i + p;
    d = p * p;
    t = 2 * a;
    p = a - d / t;
    return p;

}

__main__()
{
    float out = sqrt(23);
    print(out);
}
```

test.class

Decompiled .class file, bytecode version: 52.0 (Java 8)

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

public class test {
    public test() {
    }

    public static float sqrt(int var0) {
        int var1 = 0;

        for(int var2 = var1 * var1; var2 <= var0; var2 = var1 * var1) {
            ++var1;
        }

        --var1;
        float var3 = (float)(var0 - var1 * var1);
        float var4 = (float)(var1 * 2);
        float var5 = var3 / var4;
        float var6 = (float)var1 + var5;
        var3 = var5 * var5;
        var4 = (float)2 * var6;
        var5 = var6 - var3 / var4;
        return var5;
    }

    public static void main(String[] var0) {
        float var7 = sqrt( var0: 23);
        System.out.println(var7);
    }
}
```

```
D:\Users\Alex\IdeaProjects\Compile>java test
4.7964745

D:\Users\Alex\IdeaProjects\Compile>
```

Пример работы компилятора

additionalTest.ls

```
/*
    Программа для вычисления суммы всех чётных чисел
    от 0 до заданного значения.
*/

def int sumEvenNum (int limit)
{
    int i = 0;
    int k = 1;
    int result = 0;
    while(true)
    {
        result = result + i;
        i = i + 2;
        if (k < 5 and k != 4 )
        {
            print('k');
            k = k+1;
        }
        if (i < limit )
        {
            continue;
            i = 10000000;
            print('e');
        }
        if (i >= limit)
        {
            break;
        }
    }
    print('e');
    print('n');
    print('d');

    return result;
}

__main__()
{
    int out = sumEvenNum(90000);
    print(out);
}
```

test.class

Decompiled .class file, bytecode version: 52.0 (Java 8)

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

public class test {
    public test() {
    }

    public static int sumEvenNum(int var0) {
        int var1 = 0;
        int var2 = 1;
        int var3 = 0;

        while(true) {
            var3 += var1;
            var1 += 2;
            if ((var2 >= 5 ? 0 : 1) * (var2 == 4 ? 0 : 1) + (var2 == 4 ? 0 : 1) != 0) {
                System.out.println('k');
                ++var2;
            }

            if (var1 >= var0 && var1 >= var0) {
                break;
            }
        }

        System.out.println('e');
        System.out.println('n');
        System.out.println('d');
        return var3;
    }

    public static void main(String[] var0) {
        int var4 = sumEvenNum( var0: 90000);
        System.out.println(var4);
    }
}
```

```
D:\Users\Alex\IdeaProjects\Compile>java test
k
k
k
e
n
d
2024955000

D:\Users\Alex\IdeaProjects\Compile>
```