实验报告--step1

2018011340 计 83 郭峥岩

一、 实验内容

1. 整体框架

本次实验中 step1 采用了：源代码 -> IR(中间代码) -> riscv(汇编代码)的三级语法结构。

具体实现的语言和工具：使用 python + antlr4。其中规定好语法分析(CommonLex.g4)和词法分析(MiniDecaf.g4)的语言格式，通过 antlr4 工具生成 Lexer 和 Parser。通过 python 对生成的分析树进行分析，先进行中间代码的生成，随后将中间代码转为汇编代码。

文件具体结构及其作用如下：

```
minidecaf
    -- __main__.py 程序入口
    -- __init__.py 初始化为包
    -- main.py 生成代码的主程序
    -- utils.py 工具类
    -- CommonLexer.g4 词法
    -- Minidecaf.g4 语法
    .ir 生成中间代码
        -- __init__.py 初始化为包
        --irgen.py 生成中间代码的接口
        --irset.py 中间代码的存储与产生类
        --irstr.py 定义不同的指令对应的类
    .asm 生成汇编代码
        -- __init__.py 初始化为包
        -- asmgen.py  将输入的中间代码转化为汇编代码
        -- asmoutput.py 用于将产生的汇编代码输出到标准输出/文件
        -- asmstr.py 定义了几种 riscv 输出时的语法格式与结构
```

.generator 由 antlr4 工具根据给定的语法和词法得到的各种代码

2. Step1 的功能实现

(1)语法与词法的规范的定义：

由于 step1 中只有一种结构，即合法的语法分析树只有一种，因此定义的语法与词法分析比较简单。

(2)AST -> IR：

由于 step1 中牵扯到的语句比较简单,仅仅为 return Integer。处理的时候根据语法树中的节点，遇到 return 后的 Integer 节点则产生对应的 IR 中的 push Integer 的代码。同时 return 语句对应生成 IR 的 ret 语句进行返回。（其中 IR 的语句及其含义与语法书中的给出的定义相同）

(3)IR -> Asm：

此部分根据上面生成的 IR 列表进行生成相对应的汇编代码，具体过程比较简单：先根据上面生成的 IR 列表读取相对应的 IR 的指令的种类，在本次实现中我通过参考学长的框架，利用不同的种类代表不同的种类来进行实现。

其中每条 IR 指令对应的一系列的汇编代码都是已经定义好的（参见实现指导书中的对应关系),重点是读取传入进来的 IR 指令中包含的 return Integer 的 Integer 的值，之后根据要求将其写入文件或者标准输出即可。

(4)对于错误情况的判断：

Step1 实现了以下几种错误的辨别：return 后紧跟着的返回值为小于 0 或超过 int 的最大正整数值、没有 main 函数、含有冗余的程序结构等。其中判断返回值的大小通过生成 ir 阶段即可分析。判断

是否存在 main 函数以及是否有其他的垃圾信息的判断方式参考 stack overflow 上的实现。

首先在.g4 语法分析文件中合适的地方加上 EOF 文件结束符。之后添加 minidecaflexer 和 minidecafparser 的 addErrorListener 添加错误监听事件，出现异常则抛出错误即可。

二、 思考题

1. minilexer 报错的例子：

答：输入如下：

```
int main(){
        return -123;
}
```

原因:minilexer 中会对 return 后的整数进行判断,如果输入为负数，则会进行报错，因此上述例子 minilexer 会进行报错

2. minilexer 不会出错但是 miniparser 会报错的例子：

答：输入如下：

```
int main(){
        int a;
        return 123;
}
```

原因：上述程序能够被 minilexer 正确的进行词法分析，但是 miniparser 中设定的语法分析无法解析上述程序，因此在 minilexer 中不会报错，但是在 miniparser 中会报错。

3. riscv 中，寄存器 a0 用来保存返回值。

三、 参考资料

1. 助教所写的代码：md-dzy branch

参考上述代码，设定了编译器的基本框架。即参考了语法分析树，以及从源代码->ir->asm 的基本实现框架。

同时通读学长代码，学习了一些类的设计思想和结构。

2. Stackoverflow 中 Antlr4 中对于错误的处理实现细节：

具体网址如下：

https://stackoverflow.com/questions/18132078/handling-errors-in-antlr4

四、 总结

本次编译原理的实验 step1 步骤我认为最难的地方在于：

1. 了解 Antlr4 的特性以及其使用方法。
2. 构建合适的产生汇编代码的框架。

对于前者的学习，我通过遇到需求的时候在网上搜索相应的需求来进行，但是往往搜索不到想要的结果。于是通过助教代码先理解一些简单的用法，之后再结合具体情况进行分析。

本次实验收获最大的地方：了解了编译器产生汇编码的大致框架，对其结构有了更深的了解。