实验报告—step2

2018011340 计 83 郭峥岩

一、 实验内容

1. 整体框架

本次实验中 step2 依然延用了 step1 的整体框架，即采用：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。具体的操作过程同 step1，各个工具的具体作用及其解析可参考 step1.pdf。

本次增加的三个一元操作数的功能实现主要在于 MiniDecaf.g4 中的程序分析语法规则的改动，以及新增加的 IR 和 RISCV 的语句的对应的码。

2. Step2 的功能实现

（1） 实现程序的语法规范的改动：

之前的语法规范 return 之后一个整数，现在则是变成了一个表达式，由于只是增加了一元表达式，因此产生式可以写成 unary->Integer | ('-'|'!'|'~') unary

（2） IR 类的更改和产生：

只需要增加一种 IR 类：一元操作符类，同时在类中存储操作符即可。

（3） 汇编代码的产生：

结构同 step1，需要增加具体的一元操作符对应的汇编代码，根据上一步生成的 IR 类的顺序来完成汇编代码的最终产生。

二、 思考题

设计一个表达式，只使用-~!这三个单目运算符和 $[0, 2^{31} - 1]$ 范围内的非负整数，使得运算过程中发生越界。

答：设计的表达式如下所示：

$-\sim(2^{31}-1)$能够产生越界错误。

原因：$2^{31}-1$的二进制表示为$011111\ldots\ldots111$。即首位为 0，剩余位为 1，首先进行按位取反操作，则上述会变为$1000\ldots\ldots0000$，即-2^{31}，此时再进行取负操作，则会变为$2^{31} > 2^{31}-1$，即产生了越界错误。

三、 参考资料

助教所写的代码：md-dzy branch

四、 总结

本次编译原理的实验 step2 并没有什么难点，在 step1 中搭建好基本的框架之后实现起来比较简单，重点在于：

1. 运用上学期学习的自动机的相关知识进行设计，通过右递归的形式设计出合适的语法结构
2. 在遍历生成的语法分析树生成中间代码的时候，要判断节点类的操作类型，从而产生不同的 IR 代码。同时注意遍历语法分析树时的顺序即可。