Step 2 Report

刘轩奇 2018011025

2020年9月28日

1 工作内容

本人选择不使用辅助工具 ANTLR 因此自己实现了 lexer 和 parser。

1.1 文件说明

`montLexer.h/cpp` 词法分析器；

`montParser.h/cpp` 语法分析器；

`montConceiver.h/cpp` 产生中间代码；

`montAssembler.h/cpp` 从中间代码产生汇编代码；

`montLog.h` 记录编译错误信息；

`montCompiler.cpp` MiniDecaf 编译器，包含主函数，编译成功返回 0 否则返回 -1，并将错误信息输出到 `std::cerr`。

1.2 本步骤完成的工作

1 词法分析 添加了三种 Token，类型分别为 `Exclamation` 感叹号， `Tilde` 波浪线符号和 `Minus` 负号。

2 句法分析 添加了 `Unary` 类型节点，生成此节点时，查看 lexer 提供的第一个符号是否是一元运算符，若是则继续递归生成 `Unary` 否则生成 `Value` 类型节点。

3 产生中间代码 按照指导书上的说明，将三种符号对应生成三种中间代码：逻辑非即感叹号对应 `LNOT` 操作，按位取反即波浪线符号对应 `NOT` 操作，负数对应 `NEG` 操作。

4 产生汇编代码 在 `MontAssembler` 中，按照指导书的说明，将对应中间代码转换为对应汇编语句，逻辑非即感叹号对应 `seqz` 操作，按位取反即波浪线符号对应 `not` 操作，负数对应 `neg` 操作。

2 思考题

1 我们在语义规范中规定整数运算越界是未定义行为，运算越界可以简单理解成理论上的运算结果没有办法保存在32位整数的空间中，必须截断高于32位的内容。请设计一个表达式，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 取 `-~2147483647` 即可。