Step 6 Report

刘轩奇 2018011025

2020年10月6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 添加了新的 Token：

* `QUESTION` 问号；
* `COLON` 冒号；
* `IF` 关键词if；
* `ELSE` 关键词else。

2 句法分析 添加了以下新的 AST 节点：

* `conditional` 与指导书给出的相同。
* `if` 从`statement`节点生成`if`节点，这与指导书直接从`statement`生成`if`语句各子节点不同。
* `blockitem` 与指导书给出的相同。

已经实现了 `codeblock` 节点，该节点包括一个左大括号，零个、一个或多个`blockitem`和一个右括号。在 `statement` 生成中若遇到左括号则生成 `codeblock` 节点。

3 产生中间代码 添加了以下中间代码：

* `BNEZ`，`BEQZ`，`BR` 与指导书给定的相同。

`LABEL` 在上一步骤中已经实现。

4 产生汇编代码 添加的中间代码生成对应的汇编代码，与指导书中相同，不再赘述。

2 思考题

1 Rust 和 Go 语言中的 if-else 语法与 C 语言中略有不同，它们都要求两个分支必须用大括号包裹起来，而且条件表达式不需要用括号包裹起来：

```
if CONDITION {
  // execute when CONDITION is true
} else {
  // execute when CONDITION is false
}
```

请问相比 C 的语法，这两种语言的语法有什么优点？

答 一方面，不用小括号包裹条件显得简洁且自然；另一方面，分支必须用大括号包裹，则不再需要区分变量定义与其他语句，它们均可以放在大括号中，程序也显得比较规整。