

Step 5 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 添加了对注释的支持。遇到双斜线或斜线加星号时，忽略之后所有的注释。

2 句法分析 添加了以下新的 AST 节点，对应于指导书给出的非终结符：

- * assignment
- * declaration

对 `statement` 的处理与指导书上有所不同，分为五种，即 `return` 语句、变量定义语句、表达式语句、代码块语句（对应 step7）、空语句。

同时在 `parse` 的过程中统计每个节点的子树中所含有的变量声明占用空间的大小，以便后续处理。

3 产生中间代码 添加及修改了五种中间代码：

- * `BUILDFRAME` 在每个函数起始处使用，建立栈帧。带有一个整数参数为除了 `fp` 和 `ra` 以外所需的栈帧大小。
- * `FRAMEADDR`, `POP`, `LOAD`, `STORE` 与指导书上相同。
- * `RET` 作用与指导书上相同，但并不用跳转到一段特殊代码，而是直接弹出栈帧（`fp` 的值给 `sp`）并据此还原原先的 `fp` 和 `ra`（分别位于 `sp` 下 4 位和 8 位）。生成的具体代码详见下一节。。当遍历 AST 遇到函数末尾，而当前最后一条指令并非 `ret` 时，自动生成 `PUSH 0` 和 `RET` 语句。

4 产生汇编代码 新的或修改的中间代码将生成对应的汇编代码：

```
* [BUILDFRAME] sw ra, -4(sp); sw fp, -8(sp); ori fp, sp, 0; addi sp, sp, -8+4*k;
* [FRAMEADDR] addi sp, sp -4; addi t1, fp, -12-4*k; sw t1, 0(sp);
* [LOAD] lw t1, 0(sp); lw t1, 0(t1); sw t1, 0(sp);
* [STORE] lw t1, 4(sp); lw t2, 0(sp); sw t1, 0(t2); addi sp, sp, 4;
* [POP] addi sp, sp, 4;
* [RET] lw a0, 0(sp); addi sp, sp, 4; ori sp, fp, 0; lw ra, -4(sp); lw fp, -8(sp); jr ra;
```

2 思考题

1 描述程序运行过程中函数栈帧的构成，分成哪几个部分？每个部分所用空间最少是多少？

答 程序函数栈帧由保存的寄存器、局部变量和运算栈三部分构成。保存的寄存器至少占用8字节因为至少需要保存fp和ra寄存器；局部变量可以为空，则不需要占用空间；运算栈也可以为空，也不需要占用空间。

2 有些语言允许在同一个作用域中多次定义同名的变量，例如这是一段合法的 Rust 代码（你不需要精确了解它的含义，大致理解即可）：

```
fn main() {
    let a = 0;
    let a = f(a);
    let a = g(a);
}
```

其中f(a)中的a是上一行的let a = 0;定义的，g(a)中的a是上一行的let a = f(a);。如果 MiniDecaf 也允许多次定义同名变量，并规定新的定义会覆盖之前的同名定义，请问在你的实现中，需要对定义变量和查找变量的逻辑做怎样的修改？

答 只需要将重复的定义视为赋值语句即可。