

完成时间: 10 月 31 日, 星期二

代码生成需要将所有的 Decaf 程序转换为正确的 x86-64 汇编代码。接下来的两个项目作业会包含代码优化。现在, 我们还不关注学生的代码是否高效。

在代码生成项目结束之前, 学生应该已经编制完成一个能完全运行的 Decaf 编译器。学生应该可以编写、编译和运行真实程序!

项目作业

在代码生成部分, 学生的编译器需要可以将指令数翻译为 x86-64 汇编语言代码, 代码要能够在 Amd 64 位计算机上 Linux 环境下运行。对于一个指定的包含 Decaf 程序的输入文件, 学生的编译器必须要能生成一个汇编语言清单 ((文件名) .s)。

学生的代码需要包含能够执行分发资料 6 所包含的运行时间检查功能。附加的检查功能如整数溢出则不需要。学生必须实现对 IR 生成的运行时间检查。如果不行的话, 则需要在代码生成部分结束前完成。

之后的两个部分, 数据流优化和指令优化, 将专注于提高编译器生成的目标代码执行效率。在这个作业中, 学生所有的算术和逻辑操作使用的是"two-register"form。这就意味着需要将梭鱼的变量存储在栈中, 寄存器仅仅存储临时数据。

关于如何产生最终编译代码清单, 学生可以自由选择。但是, 我们建议采用讲座中提到的一般方法。

在代码优化项目中, 学生会有机会做一些创造性设计工作。在这第一个作业中, 学会应该将其创造能量集中于 run-time structure 的机器码表达和 (有可能的话) 讲座中提到的程序调用/返回序列。不要试图制作改良的寄存器 allocation scheme; 稍后会有这些问题需要处理。

系统使用

学生需要在 Athena 或者个人电脑上编写编译器, 不过可以将你的小组作为一个共享的知识库。学生所在小组同时拥有一个在 x86 64 位测试机上的帐号。

一旦生成了汇编输出文件, 比如说是 output.s, 学生需要将其拷贝至.....上的小组空间上。使用 gcc 编译和连接, 并运行之。

```
athena% scp output.s <test machine>: athena% ssh <test machine> <test machine>'s
password: bash3.00$
ls output.s bash3.00$
gcc output.s <library directory>.s o
output bash3.00$
./output Hello, world! bash3.00$
logout Connection to <test machine> closed. athena%
```

需要提交的内容

按惯例，在项目的 **hardcopy documentation**，遵守分发资料 3 中的说明。提交的电子部分也是按常规：在小组 **locker** 上提供一个文件名为 **leNN-codegen.tar.gz** 的 **gzipped tar** 文件，**NN** 为小组代码。这个文件需要包括所有相关的源代码和 **make** 文件。另外提供一个名为 **leNN-codegen.jar** 的 **Java** 档案文件。

解压 **tar** 文件和运行 **make** 文件都可以生成同样的 **Java** 档案文件。正确设置 **CLSAAPATH**，应该可以从下诉命令行中任一条开始运行编译器。

```
Java Compiler<文件名>
```

```
Java Compiler<文件名> -target assembly
```

学生的编译器应该可以将一段 **x86-64** 汇编清单写到扩展名为 **.s** 的同名文件中。

在没有调试标志的情况下，对于同步和语义正确的程序，不应该写出任何的标准输出和标准错误。如果调试标志存在，则编译器需要继续运行同时要生成同样的汇编结果清单。

测试

我们会用课程服务器上公开的测试案例来运行学生的编译器，另外还会有一系列隐藏的测试。

相关分发资料

资料“**x86-64 Architecture Guide**”为本资料的补充材料，从项目的角度，描述了我们对 **x86-64** 的一些观点。在编写代码之前，请务必阅读这份资料，这些代码后来会形成中间表达和 **x86-64** 指令