

麻省理工学院

电气工程与计算机科学

6.035, 2005 年秋

分发资料 3——项目概述

9 月 7 日, 星期三

这里介绍课程项目的概况和评分规则。学生不需要理解所有的技术术语, 因为我们也没有在课程中全面涵盖这些术语。我们分发这份资料的目的是让学生能够对我们布置的项目类型有所理解, 同时明确不同的截止日期。分发资料 5 会给出项目的技术细节。

全班会被分为三人一组的若干小组。学生可以尽量自己选择队友。每小组要位一个简单编程语言用 Java 编写一个编译器。希望个小组能成功的完成个阶段项目。课程的起步是快节奏的: 不要落后!

重要的项目日程

9 月 7 日, 星期三	布置词法分析, 语法分析项目
9 月 19 日, 星期一	词法分析, 语法分析项目完成时间 布置语义检测器项目
9 月 26 日, 星期一	语义检测器完成时间 布置代码生成项目
10 月 11 日, 星期二	代码生成项目完成时间 布置数据优化项目
11 月 7 日, 星期一	数据优化项目完成时间 布置指令优化功能
12 月 5 日, 星期一	指令优化项目完成
12 月 7 日, 星期三	编译器比赛

项目阶段

对编译器六部分的描述要和学生创建他们的顺序一致。

词法分析

这部分需要扫描输入流(程序), 并且用一种对后续的编译器项目适合的形式对其编码。你需要确定所想要的符号集并且为词法分析生成器创制创制一般表达式。实际上, 这部分的一些惯例已近标准了。

我们会提供一个词法分析生成器(JLex), 会包含一个词法分析器程序, 详细描述了一系列字符数组类型并且输出一个 Java 程序。这个详细描述使用常规定义描述了词法上的字符数组, 如字符序列, 如何对应生成字符数组类型的。通过解释与常规定义相对的确有限自动机(DFA), 生成的词法分析器对输入源代码进行处理。

语法分析器

语法分析器对词法分析器生成的字符数组流进行语法上的正确性检查, 并且造出在代码

生成模块中使用的程序中间表达。学生还需要构建一个符号表，因为没有这个表的话是不可能创建代码生成器的。

我们会提供一个语法分析生成器（Java CUP）。其中包含一个原始的表驱动语法分析器和一段该词法分析器使用的可将 LALR(1)文法转化为词法分析表的程序。学生必须要将相关文法转化为 LALR(1)文法。

语意检测器

这个部分检查大量的非上下文无关文法限制，如类型兼容，是否在被观察状态。我们会提供完整的观察清单，同时也会创建一张符号表来保存每个标志符的类型和地址。从以往的经验看来，很多小组都会低估完成静态语意监测器的时间，所以学生需要加倍注意这个部分的时限。

创建符号表是非常重要的，因为如果没有符号表就不可能创建代码生成器。然而，这些检测的完成却不会对项目的后续阶段产生很大的影响。在这部分项目结束的时候，编译器的前端就已经完成了，学生需要开始设计编译器剩余阶段需要使用的中间表达（IR）。

代码生成

这个项目是非常耗时的。举例来说，在之前，这个项目是分为两部分来完成的。所以，**学生需要尽早开始！**

在这个部分，学生需要把之前生成的中间表达，通过生成未被优化的 x86 汇编代码，来创建一个可运行的编译器。由于此部分时间相对较少，所以学生需要专注于正确性，而不要考虑任何的优化，无论这些优化有多么的简单。

代码生成的步骤为：首先，Decaf 语言的丰富语义要分解为简单的中间表达。如，循环和条件结构扩展到代码段就是简单的 goto 或 jump 结构。其次，中间表达和应用二进制接口相匹配，如调用惯例和寄存器使用。再次，相应的 x86 机器码就生成了。最后，代码、数据结构和存储都为汇编形式。我们会提供目标语言和 Java 接口的描述。使用这个接口的代码会在一个测试机上运行（更多的会很快在测试机上测试）。

这部分作业是有时间检查点的。在检查点，小组必须提交已编写的代码。设置这个检查点就是为了督促学生尽早开始项目。如果最终编译器能够正常工作，那么这个检查点就完全无效。但是，如果小组未能完成项目，那么在检查点所提交就对学生的分数有关键影响。如果我们判定小组在检查点之前并没有实质性的工作，我们会严厉处罚的。

代码生成检查点在网站上链接的日程上有标注。

数据流优化

这部分作业主要是编译器生成代码的优化。在稍后分发的资料中，我们会提供必须要完成优化功能的详细描述。当然，如果学生愿意，也可以实现其他的数据流优化。和之前一样，这部分生成的目标代码会在 SPIM 模拟器上进行模拟。

这部分也有检查点。请检查时间表！

指令优化

在这部分，学生需要完成一系列指令级优化功能，如寄存器分配和指令调度。这些低级

优化对在现代微处理器上的运行效果有至关重要的影响。

这部分也有检查点。请检查时间表！

评分

请一定要明确理解这部分的内容，不要被一些小疏忽而被扣分。整个项目在课程 6.035 成绩中占 70%。另外三次测试占剩余的 30%，每次测试占 10%。项目的评分规则如下：

- (10%) 设计和文档 (主观)。这部分的分数主要基于学生的设计，清晰的设计文档和在设计选择和重大事件上学生讨论时的思想尖锐程度。在项目的一些部分，我们会要求另外的文档。请阅读“提交”部分。
- (90%) 实现 (客观)。通过特定的测试就可以获得分数。每个阶段的项目都包含特殊的说明，以规范程序应该如何执行和输出结果应该是怎样的。如果学生认为有充分理由应该做一些不同的事情，要首先咨询助教。
 - 公开测试 (30%)
 - 隐藏测试 (60%)

小组成员在每部分项目上都会得到同样的分数，除非出现意外情况。出现意外时，请尽快联系助教。

<code>-o<outname></code>	将输出写到<outname>中
<code>-target<stage></code>	<stage>是词法分析、语法分析、中间表达或汇编代码中的一个。汇编应该发生在指定阶段。
<code>-opt[optimization...]</code>	执行列出的优化功能。all 表示所有支持的优化功能。-<optimization>将优化从列表中移出。
<code>-debug</code>	打印调试信息。如果这个选项不可选的话，则在成功编译后没有输出到屏幕的信息。

提交

在项目的每个部分，学生需要提交给我们两部分内容：在线源文件和硬拷贝文档。

在线源文件

在小组文件放置地点顶层用项目名创建一个新的路径 (如，词法分析器—语法分析器)。把所有相关源文件都放在这个新路径下。学生还需要提供一个 `make` 文件，用以编译源文件并把结构 `class` 文件打包成 `java` 文件 (如何使用 `make` 详细见分发资料 4)。jar 文件是助教用来运行大量测试用的。

在上交项目期限时间后 (预计完成日期 5: 00p.m.)，保存在学生子目录下的文件不允许修改。对于不同阶段的要求，请参考另外的项目分发资料。

命令行接口

学生的编译器需要有下面的命令行接口。

`javac Compiler[option|filename...]`

必须实现的命令行实参在表 1 中列出。具体的文件名需要提供，并且不能以“-”开头。文件名不能列在 `-opt` 后面，因为这样会让人认为是一个优化。

默认情况下，需要完成最近项目作业要求的编译功能，并且创建一个文件，文件扩展名主要基于目标文件，如果目标文件未指明的话，就基于“.out”文件。

默认情况下，不需要进行优化。需要进行的优化功能在优化作业中提供。

我们提供了类和 CLI，足以实现接口。同时可以返回其不能解读的自变量矢量，以增添其特点。助教不会因为这些新特点而对学生甲酚。但是，学生可以告诉我们任何这些新特点，可以用在编译器比赛中。学生可以提供“-O”标志打开学生想打开的优化。

硬拷贝文档

文档需要在预计完成日 5: 00p.m.之前提交给课程事务员。文档需要清晰、准确和易于阅读。不需要花哨的格式；无格式的文本是最可接受的。学生当然可以做一些更过度的事情，但是这无助于学生的分数。

学生的文档必须包含以下部分：

1. 大体描述小组如何分工的。这不会影响学生的分数；这在遇到任何突发事件时可以提醒助教。

2. 一个对所布置问题的任何说明、假定和附加行为的列表。项目的说明范围很宽泛，学生有很多自己决策的机会。这是真正软件工程的一个方面。如果学生认为需要一些主要说明，请咨询助教。

3. 一个对设计、学生认为的设计决策分析和关键设计决定的概述。请务必证明学生做的所有设计决定。任何由令人信服的辩论产生附加决定也是可以接受的。如果学生认为在设计的执行过程中有不足的话，可以对这些不足进行讨论和之前你会如何以不同的方法避免这种不足。另外还需要包括学生对之前部分的任何更改和为什么这些更改时必须的。

4. 对一些有意义的编译问题的大体描述。需要包含任何较重要的算法、方法和数据结构。还应该保护学生在项目过程中所发现的任何领悟。

5. 只对词法分析器和语法分析器：与本部分项目相关的带注释代码的清单。在后面的阶段，助教需要的话，可以答应这些源代码。即使已经有了小的改变，也不要项目的其他部分重复提交源代码。

6. 一个项目已知问题的清单，并且尽你所能的描述产生原因。如果学生的项目不能提供测试，同时学生也不能修复问题，就请描述一下对这个问题的理解。如果在学生自己的测试过程中发现了项目的问题，学生也不能修复的话，但是也不能由提供的测试中暴露出来，就请尽量详细的尽量多的描述这个问题的产生原因。如果这些原因使项目不能通过隐藏的测试，学生可能仍然会因为思考了这些问题而得到一定的分数。如果这些问题也没有被隐藏测试暴露的话，学生不会因为这些问题而被扣分。描述项目的问题会有助于凸现学生的优点；当然，最好能够解决这些问题。

学生可以完全自己决定如何测试自己的项目。完全的测试结构主要取决于在隐藏测试的表现。