编译报告之我见

5130309059　　李佳骏

taringlee@sjtu.edu.cn

2015.6.1

简述

题记

"这很可能是一生中唯一一次认真写编译器的日子了"

概览

本次编译原理课程主要是在完成一个简化版的C语言到MIPS指令的翻译器。这是具有一定的挑战性的。首先是相比之于数据结构大作业，实在是小巫见大巫了。而对于笔者这样没有写过这样大工程的初学者，更加有了写畏惧感。根据助教的课程设计以及自身的研究阶段方向，将分为以下四个阶段进行论述:

- Lexer And Parser (AST)
- Semantic
- IR(Intermediate Representation)
- MIPS Generation And Optimization

最后手写代码大概是4000行左右，带有一定的良好注释。但是整体结构较为混乱，这与自己第一次写Java不无关系。

1. Lexer And Parser(AST)

该阶段的最终目的是将一份简化版的C语言代码构建出一棵合适的抽象语法树。

Lexer

工作概述

顾名思义，Lexer就是分词。这里称作词法分析。意思是将这份C语言代码，通过一个正则表达式的分词设计。最后得到若干的有意义“单词”。文件lexer.flex是笔者设计的正则表达式分词控件。可通过使用软件Jflex，将该文件编译为lexer.java。

该文件主要是通过学习和融合Jflex文件夹提供的examples: cup(lcalc.flex), interpreter(scanner.flex)使之完成。

Parser

工作概述

这里所做的操作为语法分析。主要工作为对词法分析后的若干“单词”。通过上下文无关文法(CFG)进行树形划分。构建出一棵语法树。助教提供了有关一棵具体语法树(CST)的CFG文法。而根据该文法，使用JCup作为生成工具，将直接生成一棵CFG。

在这里，需要对CFG做一些优化以使得该CFG更符合人的主观考察态度。主要的方法为：

- 减小树的深度：主要是减小expression的深度。例如primary-expression可以直接成为relational-expression的儿子节点，不需要中间有一个极长的跨链。
- 增加树的宽度：通观CST文法。不难发现诸如：逗号表达式，Compound-statement是有许多儿子的。虽然CFG在遍历时只能建造出左儿子右兄弟的基本形状，但是稍作改变就可以合并为理想的树形了。

文件parser.cup是笔者设计的上下文无关文法，通过Jcup工具进行编译后，可得到笔者最后使用的AST。

在这颗AST上，每一个节点都是一个Node(.java)类型。通过简单地记录son节点链接为树。每一个节点的结构特性记录在NodeType内。

该文件主要是通过学习和融合Jflex文件夹提供的examples: java(java12.cup), interpreter(parser.cup)使之完成。

AST Printer

可以通过在Main函数里运行**TreeRoot.printAst(xxx, 0);**打印AST

2. Semantic

该阶段的最终目的是检查代码是否存在语法错误(Compiler Error)，并通过Phase2的midterm测试。

个人观点是，若该代码是不存在语法错误的，则至少它的AST每一个节点都应该是合法的。

所以，文件Semantic.Java作为主要的节点讨论文件。对于每一个不同类型的节点，都进行一次讨论。为了实现对这一讨论进行的信息记录，对于每一个节点额外附加一个信息标记(即InfoNode.java)该InfoNode在对该语法树进行深度优先遍历时得到完善。信息完善后，对该节点进行检查，主要考察左值和右值，类型转化等。

事实上这样的信息搜集并不是足够的，额外的检查还有：

- 对于C语言的四个命名空间的其中三种进行检查。通过调用(SymbolTable.Java)进行实现。
- 常量检查。了解到sizeof(a)是一个常量的事实后。将额外维护每一个类型的size大小，并进行计算得到结果；同时将一些可以做到的constant-expression计算出来。

3. IR(Intermediate Representation)

该阶段的最终目的是生成中间代码，以方便优化和MIPS代码生成。

以下是中间代码表：

op	dest	src1	src2	meaning
"lw"	a	b	c	load word from c(b) to a
"lb"	a	b	c	load byte from c(b) to a
"sw"	a	b	c	store word from a to c(b)
"sb"	a	b	c	store byte from a to c(b)
"li"	a	c		a = c(constant)
"move"	a	b		a = b
"pos"	a	b		a = +b
"neg"	a	b		a = −b
"not"	a	b		a = b
"lnot"	a	b		a = b
"addi"	a	b	c	a = b + c(constant)
"subi"	a	b	c	a = b − c(constant)
"muli"	a	b	c	a = b ∗ c(constant)
"divi"	a	b	c	a = b / c(constant)
"andi"	a	b	c	a = b & c(constant)
"add"	a	b	c	a = b + c
"sub"	a	b	c	a = b − c(constant)
"mul"	a	b	c	a = b ∗ c(constant)
"div"	a	b	c	a = b / c(constant)
"mod"	a	b	c	a = b % c(constant)
"shl"	a	b	c	a = b << c
"shr"	a	b	c	a = b >> c(constant)
"and"	a	b	c	a = b & c(constant)
"xor"	a	b	c	a = b *xor* c(constant)
"or"	a	b	c	a = b | c(constant)
"goto"	l			goto the label:l
"label"	l			the label:l
"func"	name			the function:name
"call"	name			goto function:name
"return"				return

在分别经历由于Java风格不适应致使的IR framework看不懂和第二次重构代码对于MIPS指令的“地址”和“值”的理解不足后。进行了第三次代码重构并得到了当下的中间代码。每一个函数的中间代码储存在(Function.java)内，全局变量储存在(*start*和*global*中)(Quadruple.java)表示每一条中间代码指令，而(Address.java)则是每一条指令的位置信息。

类似上一节(IR.java)是作为主要功能实现的java文件。有如下环境变量：

- LoopStack 记录当下continue和break的跳转位置
- TargetMap 记录每一个非临时变量的地址
- SymbolTable 继续使用，主要是维护命名空间以及利用语义检查进行过的空间信息。
- StringTable 对于文件中出现的字符串，进行记录以减小一定全局空间。
- InfiniteRegister 假设有一个无限大的寄存器，所有的操作都在寄存器上进行，且一个信息在寄存器上的位置和它在内存上的位置是相同的。

IR printer

可以通过在Main函数里运行**itr.IRprint();**打印IR。

值得注意的是，由于实现了局部寄存器分配。实际现实时也会显示该内存所分配的真实寄存器。

4. MIPS Generation And Optimization

该阶段首要任务是生成MIPS指令，次要任务是生成的MIPS指令在运行时越少越好。

首先是考虑三寄存器版本，由寄存器(8,9,10)进行生成；随后加入寄存器分配。分配的结果将影响整个MIPS指令生成。IR.java的后半部分是MIPS指令生成源码。值得注意的是，由于时间受迫，该寄存器分配有一定的问题，无法通过所有数据。所以对于五个个别数据，并没有使用寄存器分配优化。

小型的优化主要是内联优化，窥孔优化和常量传递。其中内联优化直接作用于IR内部，主要核心代码为$(inline_function_definition(p,q,ret))$。

常量传递和窥孔优化在(Function.java)下进行作用。

最后的测试结果并不理想，只能通过三个程序的Bonus门限。如下图：

Program	Threshold	Sat.	Now.
basicopt1	480000	1544685	558878(-78878)
Bulgarian	1100000	2666209	1181016(-81016)
expr	30000	85661	24681(Excellent)
hanoi2	500000	792456	604061(-104061)
hanoi	180000	302449	233872(-53872)
hashmap	350000	870487	15439662[Error. 4]
heapsort	13000000	35993775	13036699[Error. 7]
horse2	9000000	24239767	10951707(-1951707)
horse3	13000000	24388595	12418568(Excellent)
horse	11000000	21033699	11076989(-76989)
magic	1350000	3588908	1545093(-195093)
maxflow	13000000	25486505	9640351(Excellent)
prime	3500000	8662190	3669111(-169111)
qsort	2500000	7713088	3077831(-577831)
queens	500000	2118461	867360(-307360)
spill2	23000	38745	30782(-7782)
superloop	2100000	6143391	2164762(-64762)
tak	1800000	3880728	2061717(-261717)
treap	6000000	13689046	6269086(-269086)
twinprime	1000000	3312902	1212383(-212383)

Bonus Test

其中Excellent为通过点，Error为异常抛出点。

5. 感谢

最后在此，感谢孙锴，蒋舜宁，廖超，刘爽，许文四位助教的教导和帮助。尤其感谢我的负责人廖超助教，在我遭遇困难时提供了许多宝贵的成熟的意见。

同时，感谢同班同学陈迪，马豪君亲力指导，如果没有他们，我的优化将无法进行。