

总结感想

author:21373405 周靖宇

完成了除法优化	origin & MIPS-PLUS	JingYuChou	5 分钟 之前
可能是最终版本了		JingYuChou	今天 上午1:48
完成了消除phi但是没有做寄存器分配		JingYuChou	昨天 下午6:49
finish GVN mem2reg DeadCodeRemove FuncInline		JingYuChou	1/12/2023 下午10:24
完成了函数内联, a@GVN做了一半		JingYuChou	1/12/2023 下午5:07
函数内联有Bug...先与GVN&GCM		JingYuChou	30/11/2023 下午3:08
完成了死代码删除		JingYuChou	29/11/2023 下午5:13
完成了mem2reg		JingYuChou	29/11/2023 下午1:59
完成了mem2reg		JingYuChou	29/11/2023 下午1:59
想清楚了SSA的思路但是没有做完, 也没有真正运行		JingYuChou	27/11/2023 下午5:56
完成了基本的功能, 竞速过慢导致最后一个点tle	origin & MIPS	JingYuChou	26/11/2023 下午5:25
通过了代码生成2但是竞速三个点挂掉了		JingYuChou	22/11/2023 下午1:53
完成了alloc,binary,push and pop		JingYuChou	20/11/2023 下午5:53
重构成功,修改了llvm结构		JingYuChou	19/11/2023 上午11:26
重构了所有instr的架构	origin & llvm-ir-2	JingYuChou	14/11/2023 下午11:18
finish llvm gen code 2 qwq 好累啊		JingYuChou	10/11/2023 上午1:39
Merge pull request #4 from JingYuChou/llvm-ir	master	ZhouJY*	5/11/2023 下午5:51
finish code generate for llvm part1	llvm-ir	JingYuChou	5/11/2023 下午5:45
Merge pull request #3 from JingYuChou/reBuildStructure		ZhouJY*	21/10/2023 下午5:39
重构了项目结构, 增加了Config软件包, 更好的分离选项		JingYuChou	21/10/2023 下午5:38
2023-10-16		JingYuChou	16/10/2023 下午10:03
Merge pull request #2 from JingYuChou/ErrorManage		ZhouJY*	16/10/2023 下午9:46
finish ErrorHandle	origin & ErrorManage	JingYuChou	16/10/2023 下午9:45
10.15 update		JingYuChou	15/10/2023 下午5:20
重构了结点结构		JingYuChou	14/10/2023 下午4:58
update Node		JingYuChou	12/10/2023 下午4:12
Merge pull request #1 from JingYuChou/Parser		ZhouJY*	10/10/2023 下午5:25
Merge branch 'Parser' of github.com:JingYuChou,	origin & Parser	JingYuChou	10/10/2023 下午5:24
finish hw3 Parser		JingYuChou	10/10/2023 下午5:22
Update .gitignore	origin/Lexer	ZhouJY*	25/9/2023 下午11:31
finish hw_1 & 2		JingYuChou	25/9/2023 下午11:07
Finsh hw1 & hw2		JingYuChou	25/9/2023 下午11:05
Initial commit		ZhouJY*	11/9/2023 下午4:53

写在前面

先放上两张图, 来记录编译这三个月来的点点滴滴。写的随心所欲, 请谅解了。

距离竞速结束还有两个周, 距离实验考试还有三个周, 看起来还有很多的时间去继续做优化, 但是我想就先到这里吧, 下来还有编译原理的理论复习, 还有其他一些课程的考试, 可能真的没有时间去做一些更加细致的优化了, 优化陆陆续续做了一周, 中端优化做了一大部分, 后端优化自己写了个寄存器分配, 至少是每个部分都有所体验了, 也算是勉强可以接受。

编译这门课和os一样总有一种后知后觉的感觉, 可能是因为没有提前去预习编译, 也可能是因为我对理论的掌握并不好, 很多地方写的和理论课理想的编译器总是有一种相反的感觉。

词法分析

先是词法分析，词法分析我完全做成了字符串游戏，完全没有体现状态转移的思想，和课程组讲的 `DFA`，`NFA` 八竿子打不着，即使在期中考试，我添加一个词法也废了大半天的劲，至今对这个部分还是不算太满意。设计的思维太浅显了。

语法分析

然后是语法分析，我们的文法是上下文无关文法，但是可以说是非常不优雅的文法了（勿喷），后来学了 `LL(1)` 和 `LR(1)` 还有算符优先，感觉这玩意还是不是好语法，同时有左递归和 `FIRST` 集合的冲突，用常见的递归子程序法无法无脑解决（当然这是可以理解的，锻炼我们的能力），对于消除左递归和 `FIRST` 冲突我也有了一定的体会，但是我自己的实现真的算不上很满意，老师在课上讲过，应该使用超前扫描或者其他方法去避免回溯！！，但是我的代码到处都是回溯，可预见性超级差，幸亏这只是一个玩具编译器，并且不计算前端消耗的时间。。。毫无正规的编译器设计之风。

错误处理

错误处理这个东西的位置很有趣，你可以说他非常好，因为他引入了符号表技术，这在之后的中间代码设计有着举足轻重的作用。但是这玩意没那么好，因为错误处理起来是真的烦人，我自己也在上面受了不少苦难。这一部分我参考了<https://github.com/Hygge/Petrichor> 学长的代码，这几周的我一直在生病，编译理论缺了好多节课，对语法树这玩意完全没什么想法，看了学长的代码之后发现语法树的重要功能，也为我之后的代码生成奠定了基础。

中间代码生成

中间代码我选择了 `11vm`，这玩意刚开始写的时候感觉纯是用来恶心人的，规范很多，命名还必须按照顺序（后来才发现只有按照数字顺序命名才有这种情况/(T o T)/~~，这也导致了我在优化的时候效率非常低），写完之后才发现，这玩意感觉还好，至少标准是规范的，（四元式如果没有可靠的测试工具的话，按照我自己写代码的水准，我是不敢保证我写的代码能过的。。。），到了做优化之后我才发现，这东西简直是神中神，甚至可以检测 `phi` 语句，一个标准的中间代码确实是有大用处的。

在错误处理中我对符号表使用了单例模式，我感觉这玩意去做一个控制器实在是太好用了（为什么我当时在电梯的时候就没有学过这东西呢？？）我在中间代码生成也用了这东西，感觉他更好用了

(\doge)。中间代码这东西对应的其实是编译理论单元中的语义翻译和**翻译文法这一大堆东西，在这个时候我第一次感觉到了实验和理论的脱节，我看了往年的一些学长的设计，有些学长的理解真的很好，可以按照属性翻译的思想写出来 `11vm` 中间代码的生成，但是我对理论的理解一直很差，如果可以的话，希望课程组（理论也好，实验也好），可以讲讲语义翻译生成中间代码的思想。

目标代码生成-MIPS

我其实是先做的目标代码生成再做的优化，就先从代码生成开始吧，其实 `MIPS` 的生成我是没有遇到太大的问题的，因为 `11vm` 这玩意已经和 `MIPS` 特别接近了，只需要对着指令做翻译，做好内存管理就ok，但是在其中我还是遇到了一些困难，比如说我在类型转换漏掉了一条语句，我怎么都找不出来。特别感谢两位助教真的非常非常有耐心，非常非常棒。。给我大概提出了问题，终于完成了代码生成2。

`MIPS` 的生成和 `11vm` 的生成在我看来非常类似，所以就不多赘述了。

优化

优化我花费了大约一整周的精力，不知道算少还是算多，但是也算是体会了大多数常见的优化。。。

这些优化在我看来最让我印象深刻的还是 Mem2Reg 也就是生成 SSA，SSA 这玩意就是属于越琢磨越好使，而且越学理论的知识越觉得这玩意我 SSA 直接无脑搞定，而中端我做的比如死代码删除，函数内联什么的都是属于比较自然的优化。

中端优化的另一个杀器就是 GVN，我只做了个半成品，没有上 GCM，他大大减少了冗余的计算，配合 SSA 使用起来简直是神中神。很可惜没有做 GCM 或者是其他循环相关的优化，其实我个人还是觉得这东西是编译优化相当重要的一部分，有些代码看着就是能移出去，但是就没法做到，还是有些遗憾的。

后端我做了消除 phi，整合基本块和寄存器分配。

消除 phi 应该算不上优化，他只是个必要的过程，毕竟 phi 是不能直接转化为 MIPS 的。

整合基本块也只能说是非常简单的优化，无需多言。

寄存器分配我个人感觉是生成目标代码的核心优化了，寄存器毕竟是非常珍贵的资源，如果不能好好利用，就不能算一个好的编译器，我的寄存器分配的指导思想是参考了19届郭学长的思路，在上面增多了一些关于 spill 的判断，但是我还是觉得 spill 应该更加科学一些，所以我加入了 usetime 这种属性，但是想来还是太稚嫩，很可惜没有去写真正的图着色，没有消掉我的一大坨 move。。。

说在后面

其实扯皮了这么多没用的，我在编译中感受到了一个非常重要的理念就是**设计大于实现**，我无数次感叹于自己的框架之冗余，尤其是我阅读了<https://github.com/Thysrael/Pansy>这份编译器代码，我更是感觉到我自己写的编译器真的是玩具中的玩具。

写编译的过程中，我也不记得我重构了几次了，语法分析至少重构了一次，写完错误处理重构了一次文件结构，写完 llvm 整个重构了我的 llvmValue，在此之前，我的框架充满了特判，乱飞的字符串让我觉得头大，在搞定这个框架之后我再也没有重构过了，也顺利完成了 MIPS 以及优化等任务。至今我还是觉得我的框架实在是弊病太多，冗余极多，到处都是多余的代码，如果有机会在写一次编译器，我相信我是肯定不会再犯类似的错误了。

写编译器的路上我也不记得我忘记吃了多少次午餐晚餐，有时候写着写着就到半夜了，仿佛置身世外，我的舍友也很包容，很感谢他们。也感谢每一位为我答疑的助教，cm，szc助教给我解答了 MIPS 代码生成2的问题，感谢答疑的时候某位不知道名字的助教给我手把手画支配树，虽然当时的我还是没听懂，但是这也告诉了我程序的流图也可以转化为树的形式。感谢课程组提供了辅助测试库，这让我不用花太多的时间在生成数据和测试上，也感谢编译给我带来的时间管理能力的提升。。。总之，编译无疑是一门好课