联系方式

- 手机：18340833376
- Email：eng.lzx@pku.edu.cn
- 微信号: lizhengxian19950804

个人信息

- 李拯先/男/1995
- 研究生/北京大学软件与微电子学院
- 本科/大连理工大学/机械工程
- 技术博客：https://cookieli.github.io/blog/
- Github：https://github.com/cookieli

使用语言

自己项目中主要使用过的语言有c,java, python, go.用c++ 实现过一些算法和数据结构.

学习经历

由于自己本科并非计算机出身，所以以下课程都是自己通过网上名校的课程自学并完成其课程项目。

编译器

自学mit6.035 并完成如下项目

实现了一个c的子集的编译器
https://github.com/cookieli/compiler_for_tiny_c

该编译器支持int 型和bool型的变量和数组，支持函数，可以通过import调用c中的函数.具体各部分实现如下：

scanner 和parser

我利用antlr实现其scanner 和parser 部分。

semantics checker:

在semantics checker部分：我设计了中间表示节点(IrNode),并利用其构建抽象解释树(AST_tree),同时设计了symbolTable用来存储变量和方法的相关信息，symbolTable有层次结构，可以方便在不同的scope内进行查询.
执行过程中，将parser生成的符号转化为ASTtree和相关的symbol table,实现以下功能:

- 变量在同一个scope内不可以被定义两次，且必须先定义后使用
- 数组的长度大于零

- 方法调用必须和定义一致，如果方法出现在表达式中则必须返回类型和该表达式一致
- 在各类语句以及相关表达书中要进行类型检测
- break 和continue必须在循环中
- 所有的整形数据必须在−9223372036854775808 ≤ x ≤ 9223372036854775807(64 bits)

code generator

在该部分我设计了数据流图(cfg)和三地址码，利用stack based machine 为基础，将AST不断转化为接近机器码的中间表达式，最终转化为X86_64可以运行的汇编(assembly).并完成了以下运行时检查:

- 数组越界检查
- 有返回值的函数退出时一定会返回相应的值

dataflow analysis

在该部分我完成了俩个工作，一是对每个CFGNode进行基本的优化，包含以下部分:

- Common Subexpression Elimination
- Copy propagation
- Dead Code Elimination

二是全局的的优化，此处利用数据流分析完成以下部分

- Global Common Subexpression Elimination
- Global Copy propagation
- Global Dead Code Elimination
- Global Constant Propagation and Folding

optimizer

在该部分我实现了基于图着色法的寄存器分配算法，此处使用并查集和图的一些算法进行实现。

计算机网络

自学cmu 15-441完成以下俩个项目

实现 一个http server(C)
https://github.com/cookieli/http-server

在该项目中我利用C和e Berkeley Sockets API ,利用select实现了一个web Server, 并基于RFC2616实现了HTTP1.1的一个子集，具体功能如下:

- GET
- HEAD
- POST
  同时还实现了基于TLS的HTTPS 和 服务器端的CGI.

实现一个利用tcp/ip协议进行文件传输的系统
https://github.com/cookieli/Congestion_Control_with_Bit_torrent

在该项目中实现了一个 文件传输的应用，该应用运行在UDP上，我设计一个类似于TCP的可靠传输和拥塞克制的的协议进行文件的传输.我完成以下部分:

- 设计实现相关的packer header
- 实现100%可靠传输和滑动窗口机制
- 拥塞控制，包含Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery

数据库

自学mit6.830并阅读完课程中所涉及到所有paper,完成其课程项目:

实现了一个基本的数据库系统.
https://github.com/cookieli/mit_db_project

在数据库部分我做了以下的工作

- 设计，实现了field 和tuple, catalog和bufferPool
- 设计,实现了 HeapFile access methods
- 设计,实现了基于iterator的operator, SeqScan, filter, join, insertion 和deletion,aggregation,
- 实现了基于统计的对join进行优化的optimizer.
- 实现了lock, transaction,buffer的no steal机制还有deadlock的检测和退出.

分布式系统

完成了部分mit6.828,并阅读过课程所cover的论文
同时完成了相关一些论文的阅读， 笔记如下:
https://cookieli.github.io/blog/

完成了map reduce 和raft算法
https://github.com/cookieli/tiny_distributed_system

此部分利用go语言实现
根据mapreduce论文实现了mapreduce.
根据raft论文实现了raft算法。

其他

学习过cmu的15-213,伯克利的cs61A, cs61b，毕设是利用深度自编码器进行轴承的故障检测，并在毕设期间自学cs231n.

Contact Information

- phone:18340833376
- Email：eng.lzx@pku.edu.cn
- wechat: lizhengxian19950804

Personal Information

- Zhengxian Li/male/1995
- Master/Peking Unversity
- blog：https://cookieli.github.io/blog/
- Github：https://github.com/cookieli

Personal Project

compiler

I wrote a tiny c compiler
github:
https://github.com/cookieli/compiler_for_tiny_c

It supports bool and int variable ,array and Function. It can use import to support c's library invocation.I implement following part.

parser and scanner

I use antlr to generate parser ans scanner.

semantics checker

I design a high-level intermediate representation node and use it to construct a AST tree. I also design a symbol table to store variable and method. Symbol table has parents so it can represent scope.
It checks following constrains:

- No identifier is declared twice in the same scope and used before declared.
- Array's length must be greater than zero.
- Method invocation must be same as definition . If method appears in a expression, it must have a return value.
- Every expresion and statement must check its types.
- Break and Continue must appear in loops
- All integer literals must be in the range $-9223372036854775808 \le x \le 9223372036854775807$(64 bits.)

code generator

I design 3-address-code and control flow graph.Using stack based machine, I transform AST into machine code representative of run time structure. And I finally generate assembly which target is X86_64. And it also implements a run time checking:

- The subscript of an array must be in bounds
- Control must not fall off the end of a method that is declared to return a result.

Dataflow analysis

In this part I implement following two parts. First is basic CFG node optimization which inculdes:

- Common Subexpression Elimination
- Copy propagation
- Dead Code Elimination
  Second is global dataflow analysis
- Global Common Subexpression Elimination
- Global Copy propagation
- Global Dead Code Elimination
- Global Constant Propagation and Folding

optimizer

I implement Registor allocation and spilling via graph coloring.

computer network

I wrote a http server.
https://github.com/cookieli/http-server

In this project I use the Berkeley Sockets API to write a web server using a subset of the HTTP 1.1 Request as described in RFC2616. which inculdes following:

- GET
- HEAD
- POST

I also implement Https via TLS and server-side CGI.

I wrote a Bittorrent with Congestion Control.
https://github.com/cookieli/Congestion_Control_with_Bit_torrent

I implement a file transfer application which run on top of UDP. I implement a reliability and congestion control protocol (similar to TCP) for the application. I complish following parts:

- Design, and implement relevent packet headers.
- 100% Reliability & Sliding Window
- Congestion control which includes Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery.

database

I teached myself mit6.830 and implement a basic database.
https://github.com/cookieli/mit_db_project

I do following jobs using java:

- design, and implement classeds of tuple, tupleschema field, catalog and bufferPool.
- HeapFile access methods
- Desgin ,and implement operators which includes Join , Filter, and Aggregate whos base is iterator.
- optimizer for join based on statictics
- lock. transaction, No steal policy and Deadlock detection.

distributed system

I teached myself mit6.828 and implement mapreduce and raft algorithm
https://github.com/cookieli/tiny_distributed_system

I learned golang myself implement following algorithm:

- mapreduce
- raft

I also read some papers of this topic and have some notes on my blog:
https://cookieli.github.io/blog/