计算机系统课程设计（编译器）

2022级（2022-2023 夏季学期）

Why Compiler? Why this time?

- 锻炼能力：
  - 代码能力：这是ACM班标准大作业中首个预计代码量超过1万行代码的作业。
  - 工程能力：你需要从头完全实现一个编译器。
  - 学习能力：我们不会开编译原理课详细讲怎么编译，需要自己学习。
  - 同伴帮助能力：互相交流很重要。
- 传承：这作业快有 10 年了。
- 打开系统研究的大门：系统的知识需要大量上手实践的内容。

- 根据2019、2020、2021级的经验，需要一个不受打扰的环境完成该作业。

程序如何运行?

- 贯穿整个系统课程的核心问题
  - 计算机系统（1）【体系结构】：指令在硬件上的运行方式（PPCA）
  - 计算机系统（2）【操作系统】：软件管理资源并且调度执行任务
  - 计算机系统课程设计【编译器】：代码翻译为硬件指令
  - 计算机系统综合课程设计(TBA)【分布式系统与软硬件结合】：多个机器如何协同计算

```
(base) ➜  example_case ./how_program_execute
Hello, World!
(base) ➜  example_case cat how_program_execute.cpp
#include <iostream>

int main(){
        std::cout << "Hello, World!" << std::endl;
        return 0;
}
```

- **系统研究不需要数学，非常依赖编写代码水平**

从源代码到真正执行，有哪些阶段?

```
(base) ➜  example_case clang++ -S -o how_program_execute.S how_program_execute.cpp
(base) ➜  example_case head -n 20 how_program_execute.S
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 13, 0     sdk_version 13, 3
        .globl  _main                           ## -- Begin function main
        .p2align        4, 0x90
_main:                                  ## @main
        .cfi_startproc
## %bb.0:
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register %rbp
        subq    $16, %rsp
        movl    $0, -4(%rbp)
        movq    __ZNSt3__14coutE@GOTPCREL(%rip), %rdi
        leaq    L_.str(%rip), %rsi
        callq   __ZNSt3__1lsINS_11char_traitsIcEEEERNS_13basic_ostreamIcT_EES6_PKc
        movq    %rax, %rdi
        movq    __ZNSt3__14endlIcNS_11char_traitsIcEEEERNS_13basic_ostreamIT_T0_EES7_@GOTPCREL(%rip), %rsi
        callq   __ZNSt3__113basic_ostreamIcNS_11char_traitsIcEEElsB6v15006EPFRS3_S4_E
```

```
(base) ➜  example_case objdump -d how_program_execute

how_program_execute:    file format mach-o 64-bit x86-64

Disassembly of section __TEXT,__text:

0000000100003b30 <_main>:
100003b30: 55                           pushq   %rbp
100003b31: 48 89 e5                     movq    %rsp, %rbp
100003b34: 41 56                        pushq   %r14
100003b36: 53                           pushq   %rbx
100003b37: 48 83 ec 10                  subq    $16, %rsp
100003b3b: 48 8b 3d f6 04 00 00         movq    1270(%rip), %rdi
100003b42: 48 8d 35 2f 04 00 00         leaq    1071(%rip), %rsi
100003b49: ba 0d 00 00 00               movl    $13, %edx
100003b4e: e8 7d 00 00 00               callq   0x100003bd0 <__ZNS
ERNS_13basic_ostreamIT_T0_EES7_PKS4_m>
100003b53: 48 89 c3                     movq    %rax, %rbx
100003b56: 48 8b 00                     movq    (%rax), %rax
```

高级语言代码 → 汇编代码 → 二进制代码 → 调度、加载 → 发射指令

```
(base) ➜  example_case cat how_program_execute.cpp
#include <iostream>

int main(){
        std::cout << "Hello, World!" << std::endl;
        return 0;
}
```

```
00000000: cffa edfe 0700 0001 0300 0000 0200 0000  ................
00000010: 1100 0000 9004 0000 8580 2100 0000 0000  ..........!.....
00000020: 1900 0000 4800 0000 5f5f 5041 4745 5a45  ....H..._PAGEZE
00000030: 524f 0000 0000 0000 0000 0000 0000 0000  RO..............
00000040: 0000 0000 0100 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 0000 0000 1900 0000 d801 0000  ................
00000070: 5f5f 5445 5854 0000 0000 0000 0000 0000  __TEXT..........
```

汇编代码?

- 汇编语言（Assembly language，称ASM）是任何一种用于电子计算机、微处理器、微控制器，或其他可编程器件的低级语言。
- 汇编语言与系统架构（x86、ARM等）相关，不同的架构间不能迁移。

- 代表了程序执行的指令，“真正在CPU上运行的程序”

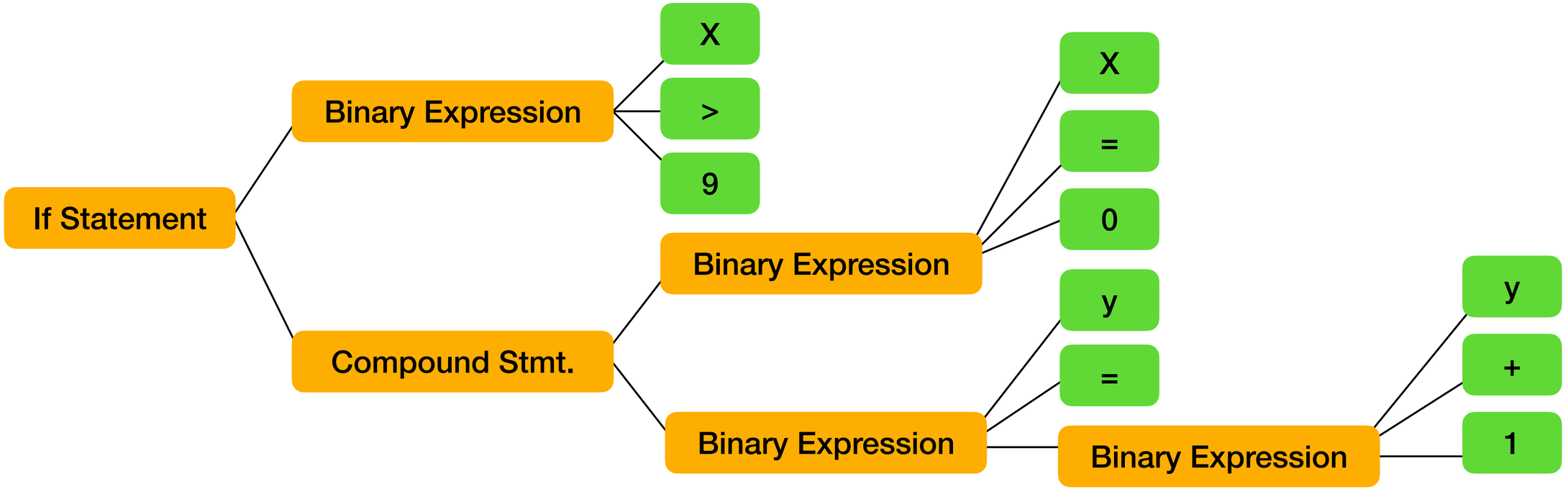
- 汇编执行：顺序执行，遇到跳转则跳转

编译过程

- 源代码 - AST - IR - ASM

```
if(x>9){ x = 0; y = y + 1; }
```

  - AST：抽象语法树、IR：中间表达、ASM：目标汇编



编译过程

- 源代码 - AST - IR - ASM

if(x>9){ x = 0; y = y + 1; }

  - AST：抽象语法树、IR：中间表达、ASM：目标汇编

```
define void @foo(i32 noundef %0, i32 noundef %1) #0 {
  %3 = alloca i32, align 4
  %4 = alloca i32, align 4
  store i32 %0, i32* %3, align 4
  store i32 %1, i32* %4, align 4
  %5 = load i32, i32* %3, align 4
  %6 = icmp sgt i32 %5, 9
  br i1 %6, label %7, label %10
7:                                                ; preds = %2
  store i32 0, i32* %3, align 4
  %8 = load i32, i32* %4, align 4
  %9 = add nsw i32 %8, 1
  store i32 %9, i32* %4, align 4
  br label %10
10:                                               ; preds = %7, %2
  ret void
}
```

Something Old, Something New

- 实现一个从前端到汇编代码的Mx语言编译器。
  - 仓库：https://github.com/ACMClassCourses/Compiler-Design-Implementation
- Language：任意，有特殊环境请联系助教。
- 仅允许使用ANTLR等词法分析库。
- 分为 3 个阶段：Semantic、Codegen、Register Allocation
  - Semantic：检查输入的源代码是否符合语言规范
  - Codegen：由源代码生成汇编代码
  - Register Allocation：分配寄存器
- 小学期第7周 - 第10周：完成到源代码生成汇编代码（即Codegen）
  - 时间紧张，切忌浪费时间
  - 小学期第1周 - 第6周：每一周有一个体验性任务需要完成（不检查），如果体验任务搞不明白的。**立即找TA直到你弄明白为止。**

体验任务

第1周 6月19日-6月25日	1. 配置ANTLR环境； 2. 熟悉并了解ANTLR遍历语言的过程。
第2周 6月26日-7月2日	1. 尝试修改ANTLR的一部分，了解ANTLR中的Tagging使用； 2. 利用ANTLR进行简单的语法检查过程。
第3周 7月3日-7月9日	1. 利用ANTLR进行简单的语法检查。
第4周 7月10日-7月16日	1. 利用ANTLR进行简单的语法检查； 2. 尝试构建AST。
第5周 7月17日-7月23日	1. 尝试构建AST； 2. 了解IR（中间表达）并且熟悉从AST转换为IR的过程。
第6周 7月24日-7月30日	1. 了解IR（中间表达）并且熟悉从AST转换为IR的过程； 2. 学习LLVM IR的基本设计。

评分标准与时间节点

- 体验任务不计入评分，**但是很大程度影响上手。**
  - 我们会了解体验任务的完成情况。
- 80%测评分数 + 20% Code Review
  - 20分：Code Review 并**不会保证满分**。
  - 80分：30分 Semantic，20分 Codegen，30分 Register Allocation
- 迟交扣分准则 累进扣分制：超过x天提交的扣分不超过 $\sum_{i=1}^{x} i$。
- 超过极限迟交日期则该课程以不及格计。
- 特殊情况请联系 TAs。