第一组（本组）听课笔记

——编译史

本组汇报选题是第一题：编译发展历史，包括编译原理/理论的发展史、编译工程/技术的发展史以及编译史上的名人。汇报主要从编译原理的发展、编译技术的发展、编译史上的名人三个方面入手 。

一、编译原理的发展

这部分介绍了 Chomsky 架构、上下文无关文法、分析问题、有限状态自动机、正则表达式等概念。

上世纪 50 年代，IBM 的 John Backus 带领一个研究小组对 FORTRAN 语言及其编译器进行开发。但由于当时人们对编译理论了解不多，开发工作变得既复杂又艰苦。与此同时，Noam Chomsky 开始了他对自然语言结构的研究。他的发现最终使得编译器的结构异常简单，甚至还带有了一些自动化。Chomsky 的研究导致了根据语言文法的难易程度以及识别它们所需要的算法来对语言分类。正如现在所称的 Chomsky 架构（Chomsky Hierarchy），它包括了文法的四个层次：0 型文法、1 型文法、2 型文法和 3 型文法，且其中的每一个都是其前者的特殊情况。2 型文法（或上下文无关文法）被证明是程序设计语言中最有用的，而且今天它已代表着程序设计语言结构的标准方式。分析问题（parsing problem，用于上下文无关文法识别的有效算法）的研究是在 60 年代和 70 年代，它相当完善的解决了这个问题。现在它已是编译原理中的一个标准部分。有限状态自动机（Finite Automaton）和正则表达式（Regular Expression）同上下文无关文法紧密相关，它们与 Chomsky 的 3 型文法相对应。对它们的研究与 Chomsky 的研究几乎同时开始，并且引出了表示程序设计语言的单词的符号方式。人们接着又深化了生成有效目标代码的方法，这就是最初的编译器，它们被一直使用至今。

二、编译技术的发展

这部分介绍了什么是编译器：从本质上看是一段代码，而从功能上看可以将一段代码等价地转化为另一段代码。如果我们用 A(S -> T)表示一个编译器，这个编译器本身是由 A 语言写成的，它的功能是将 S 语言代码等价地转换为 T 语言代码。当 A 语言是机器语言的时候，这个编译器就是可以使用的。

还介绍了通过自展生成编译器，交叉编译程序，编译器反馈，并行编译技术，VHDL 编译技术等。

三、编译史上的名人

这部分主要从国内外两方面介绍在编译发展史上的关键人物。

国外名人有设计了世界上第一个编译器的 Grace Murray Hopper、提出了 RISC 概念（中心思想就是简化硬件设计）的 John Cocke。

60 年代初，董韫美院士和杨芙清院士分别在中科院和北大领导研究组开发编译器，那时面向的高级语言是 ALGOL 和 FORTRAN，目标机是国产机。

70 年代中科院计算所张兆庆教授研究组（ACTGroup）开始在国产机上研制 FORTRAN 语言编译器，先后参与了众多的院级和国家级科研攻关项目，主持开发了 013，757，KJ8920 等国产大型机系统中的 FORTRAN 语言编译器，所研制的编译器支持了数百万行应用软件的运行。

90 年代以来 ACTGroup 承担科学院重大项目，国家攻关项目，863 项目，以及国际合作项目，先后开发了共享内存多处理机的并行识别器，分布式内存多处理机的并行识别器， SIMD 芯片和 VLIW 芯片的并行优化 C 编译器。将编译技术与图形学结合，ACTGroup 还推出了集成化、可视化的并行编程环境。

2019 年，华为首次宣布了华为方舟编译器技术并在之后开源。安卓系统使用 Java 作为编程语言，易于开发，但是不会将代码直接编译成机器语言，程序运行时有相当一部分代码还需要通过手机上的虚拟机临时同步编译，影响程序执行的效率。华为方舟编译器改变了系统及应用的编译和运行机制,直接将高级语言编译成机器码,消除了虚拟机动态编译的额外开销,显著提升了手机运行效率。

正是张教授、李院士等众多前人的高瞻远瞩、坚持不懈，深耕编译领域，才有了我国在该领域的迅速发展。在为编译领域取得进步而欣喜的同时，我们也应该认识到我国在该领域仍与英美等国家存在一定差距。当下的我们，应该认真学习《编译原理》这门课程，追寻前辈的步伐，为解决卡脖子技术难题，占领科技高地，做出应有的贡献。

第二组听课笔记

——离散数学在编译原理中的应用

1. **等价原理**：编译原理中**形式文法**等价的定义以及**有穷自动机**关于两个自动机等价的原理都还原了离散数学中的等价原理。
2. **演绎与归纳**：编译原理中表示语言的形式文法(描述某个字母表上一些有限长字串的集合的方法)而言， 需要通过推导过程得到语言的句型或句子； 文法的推导过程实际上就是离散数学的逻辑推理过程。
3. **图论**：一部分，**有穷自动机**看上去就像是一个有向图，其中状态是图的节点，而状态转换则是图的边。此外这些状态中还必须有一个初始状态和至少一个接受状态；另一部分，编译原理中体现形式化的最重要的内容就是使用**语法树**来对应文法的句型或句子的推导过程，这也同样与图论紧紧关联。
4. **代数系统**：在编译原理中，语言被定义为句子的集合，假设将语言指定为集合$\sum*$，则$\sum*$ 与定义在其上的计算则可构成代数系统。 假如考虑语言（符号串集合）$\sum*$ 上的连接运算，显然连接运算“·”在语言$\sum*$ 上是封闭的，因而， $<\sum*,\cdot>$为代数系统。

第三组听课笔记

——编译原理研究现况及主流方向

1. **WasmAndroid**（让 Android 中的应用可以运行在多个硬件平台）：将源代码转为

webassemble，然后在设备上运行。使用 WasmAndroid，开发人员可以编译他们的本机源代码将现有二进制文件编码或传输到 WebAssembly 和 exe，在不同的硬件平台上运行，无需额外的重新配置。

2. **优化并发编程**:

非线程安全的情况下，多线程对同一个地址空间进行写操作容易引起**数据竞争。**

对此的解决方案:

方案一：提前检测，一个内置的数据竞争检测器，我们可以使用它来确定应用程序里潜在的数据竞争条件;

方案二：使用 waitgroup 锁，组织读取访问，直到写入操作完成;

方案三：使用通道阻塞;

方案四：使用 Mutex 互斥锁。

3. 软件可靠性和调试技术：

软件可靠性设计的实质就是减少缺陷和避免暴露，形象点说就是把门关起来，不让错误进来造成缺陷，也不让故障出去导致失效。

调试就是我们让软件在各种可能的输入模式（模拟真实使用环境）下运行，然后分别检查输出是否正确。

4. 内存技术：

垃圾回收：不能被引用的数据被垃圾处理器回收（数据位置、类型）

目标：类型安全、性能度量（总体运行时间、空间、停顿时间、程序局部性）。

目前策略：引用计数垃圾回收器。

碎片整理： 切分窗口过度

目前策略：分在满足请求的最小可用窗口 best-fit；第一个满足请求的可用窗 口 first-fit。

深度学习编译（利用编译手段加速深度学习应用）:

1、编译器需要做一系列优化(比如寄存器分配等)。

2、深度学习图优化和代码优化技术。

第四组听课笔记

——主流编译器简介

1. 介绍了编译器的定义--一种计算机程序，它会将某种编程语言写成的源代码（原始语言）转换成另一种编程语言（目标语言）。以及编译器的结构--（有一个前端接口，有一个后端接口。一个前端解析符合此语言的源代码，并产生一抽象语法树，以及翻译此语法树成该编译器的寄存器转换语言的后端）。

2. 介绍了几种 C++的编译器：GCC、LLVM + Clang、cl.exe。GCC 通常是跨平台软件的编译器首选。GCC 在所有平台上都使用同一个前端处理程序，产生一样的中介码，因此此中介码在各个其他平台上使用 GCC 编译，有很大的机会可得到正确无误的输出程序。Clang 本身性能优异，其生成的 AST 所耗用掉的内存很小。cl.exe 就是 VS 内部的编译器，只能在 windows 上运行。

3. Go 语言编译器：Go 的语法分析器对错误的处理有一个很有特色,它不会遇到一个错误就停止编译，而是要尽可能跳过当前这个出错的地方。Go 语言的编译器完全用 Go 语言本身来实现但是相较于 GCCGO GC 实现的不严谨，对 32 位系统支持很差。

第五组听课笔记

1. 介绍了 Visual Studio 不同时期的发展历程。

2. 介绍了 Visual Studio Code 的软件定位和插件模型。通过与 eclipse 的插件进行对比，说明 Visual Studio Code 插件的稳定性。介绍 Visual Studio Code 的 UI 和 LSP--基于文本的协议。

3. 介绍 Java 的编译器——javac 命令。该部分从其历史和特点入手，阐述了使用 javac 指令可以将.java 源文件翻译成.class 字节码文件，.class 文件可以装载到 JVM。与此同时介绍了 Java

的编辑器——ECJ（**实际上他们在解说时说的是编译器**）。

4. 介绍 c 语言的编译器——GCC。首先用大段文字说明 gcc 的特点，但是 ppt 上字体太小，又没有修改字体颜色，我们看不清楚。最后讲述 gcc 的区别与联系，但是我不明白他们的标题是什么意思，gcc 和谁对比？PPT 上的字体还是太小，看不清。

5. 介绍 PyCharm。PyCharm 是 Python 的专用 IDE，安装后的配置也非常简单。然后罗列了这款编辑器的优点：活跃的社区支持、支持全面的 Python 开发，不论是数据科学还是非数据科学项目、新手和老兵都易于使用。以及其加载可能比较慢的缺点。

6. **总结：总体来说，他们组可能理解错题目的意思了。题目要求讨论的是主流的编译器，而该组花费了大量的篇幅介绍了主流的编辑器 Visual Studio、Visual Studio Code、ECJ、PyCharm。其次，ppt 可能还要调整一下排版，不然我们的视力真的达不到要求。最重要的一点，关于小班讨论课，我们想听到一些比较深入的东西，而是不听一个人在台上念 PPT，如果是这样的话，我们大可以自己看 ppt。**

第六组听课笔记

1. 介绍了 GCC 与 llvm&Clang 的诞生和发展。

2. 介绍 GCC 和 llvm&Clang 各自的特点。其中 GCC 系统支持众多前端编程语言、众多目标机器体系结构，最重要的是 GCC 编译系统生成的代码具有很高的可靠性和运行效率并且支持并行编译。而 LLVM 前端对于不同的语言它都提供了同一种中间表示，后端则可以将中间表示再转化成对应平台的机器码。

3. 两种编译器之间性能和使用的区别以及 GCC 与 Clang 之间的联系：

   GCC 产生的代码联合度高、不好独立、难以整合，促使的 Clang 的诞生。

第七组听课笔记

1. 介绍语法分析的应用，以抄袭检测程序为例，对其进行了具体介绍。

2. 介绍编译原理在语言翻译方面的应用，如百度翻译。

3. 介绍有限状态机在 Invoice 收票自动化系统中的应用，即在结构化过程中，按照一定模型把序列化的字符流还原为结构化的发票信息。这里便运用了编译原理的知识。

4. 在安全方面，以 GS 安全机制为例，对 GS 的工作原理作了详细说明。