




编译原理

信息科学与技术学院
马新娜



联系方式

E-mail: maxinnamxn@163.com

手机: 13331393416

办公室: 信息科学与技术学院办公楼201

资料:



课程简介

- 重要的计算机专业必修基础课程，理论性强、算法多、难度大
- 硕士研究生和博士研究生入学的考试科目
- 课程性质：必修课，4学分
- 学时：52上课+12实验=64学时
- 参考教材：
 - 编译原理(第3版)，清华大学出版社，王生原等编著，2015年6月出版

课程目的和基本要求

- **课程性质**
 - 技术基础
- **基础知识要求**
 - 高级程序设计语言，数据结构与算法，形式语言与自动机
- **主要特点**
 - 既有理论，又有实践
 - 面向系统设计
 - 涉及程序的自动生成技术

课程目的和基本要求

- **本专业人员4种基本的专业能力**
 1. 计算思维能力
 2. 算法的设计与分析能力
 3. 程序设计和实现能力
 4. 计算机软硬件系统的认知、分析、设计与应用能力
- **计算思维能力**
 1. 逻辑思维能力和抽象思维能力
 2. 构造模型对问题进行形式化描述
 3. 理解和处理形式模型

课程目的和基本要求

- **知识**

——掌握编译程序的总体结构、编译程序各个组成部分的任务、编译过程各个阶段的工作原理、编译过程各个阶段所要解决的问题及其采用的方法和技术

- **能力**

1. 掌握程序变换基本概念、问题描述和处理方法
2. 增强理论结合实际能力
3. 培养“问题形式化描述、计算机化”的问题求解过程
4. 使学生在系统级上认识算法和系统的设计，培养系统能力

课程要求

- 课前预习
- 上课认真听讲，不允许聊天、睡觉、玩游戏、看与本课程无关的书籍等
- 无故不得旷课，有事要请假，三次以上无故旷课取消期末考试资格
- 课后认真复习，按时独立地完成作业
- 实验课之前要认真准备，认真、独立做实验
- 考试前认真复习

成绩构成

- 考勤
 - 书面作业
 - 网络课程参与情况
 - 实验
 - 考试
-



第一章 引 论

主要内容

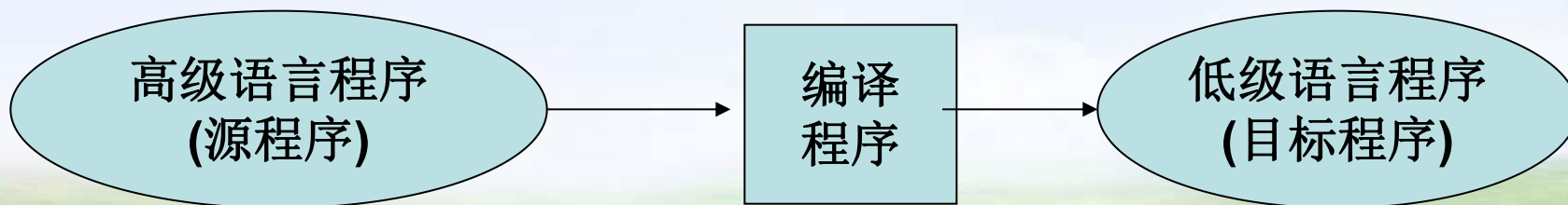
- 什么是编译程序
- 编译过程概述
- 编译程序的结构
- 编译阶段的组合
- 解释程序
- 一些软件工具
- 程序设计语言范型

编程语言的发展历程

- **第一代语言：机器语言**
 - 特点：面向机器
- **第二代语言：汇编语言**
 - 特点：用针对指令的符号代替二进制码
- **第三代语言：高级语言**
 - 包括面向过程的语言和面向对象的语言。
 - 例：Pascal, C, C++, C#, Java等
- **第四代语言：面向问题的语言**
 - 例：SQL
- **编程语言的发展体现了编译技术的发展**

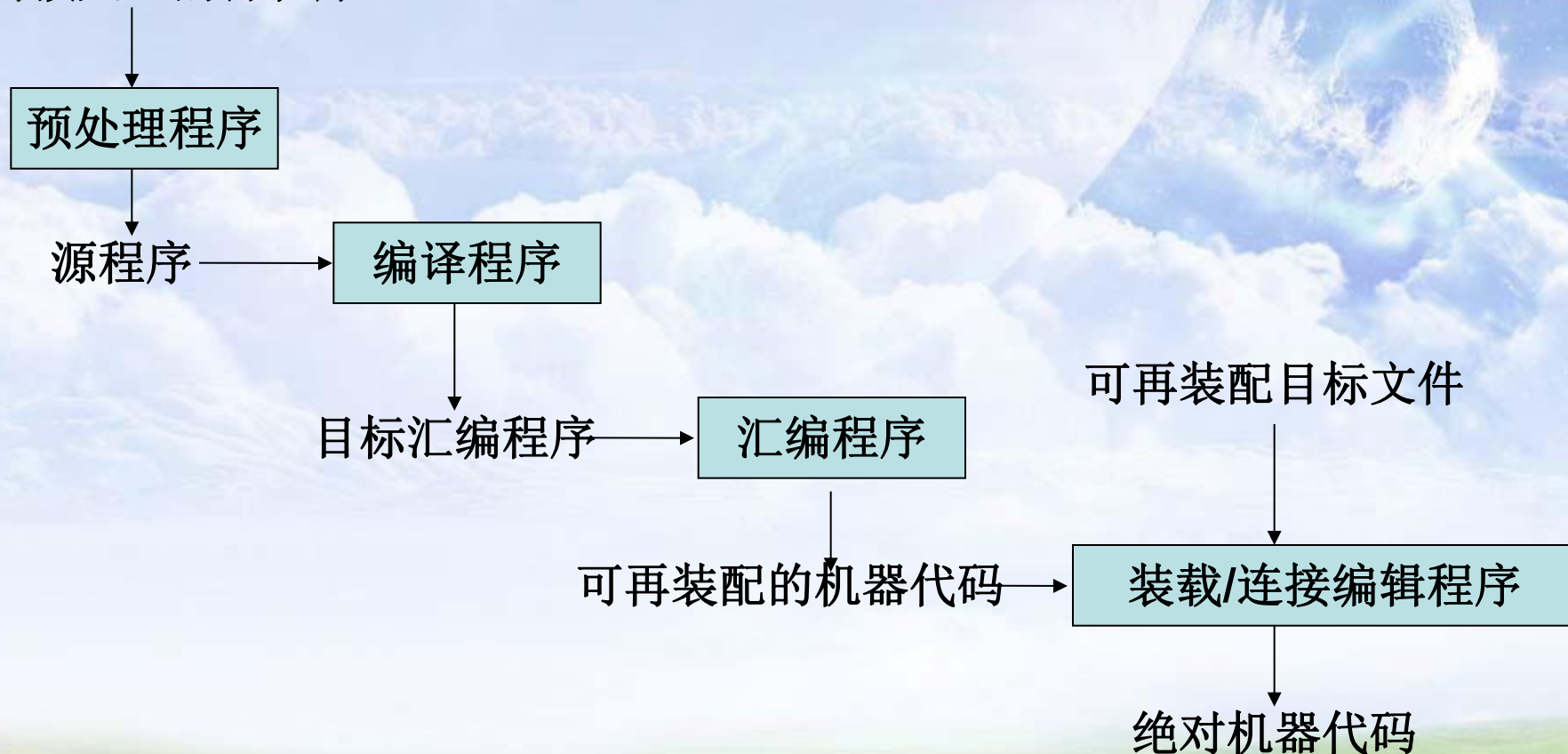
1.1 什么是编译程序

- 语言翻译程序：把一种语言（称作源语言）书写的程序翻译成另一种语言（称作目标语言）的等价的程序
- 如果源语言是高级语言，目标语言是汇编语言或机器语言等低级语言，则这种翻译程序称作编译程序
- 编译程序的功能：



高级语言程序的处理过程

需预处理的源程序



编译器和集成开发环境

- **编译器：**即编译程序，把高级语言经分析翻译为低级语言。
- **集成开发环境：**简称IDE（Integrated Development Environment），是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具。
 - **背景：**早期程序设计的各个阶段都要用不同的软件来进行处理,如先用字处理软件编辑源程序，再用编译程序进行编译，然后用链接程序进行函数、模块连接,开发者必须在几种软件间来回切换操作。
- **人们习惯上经常把IDE称为编译器。**

常见语言及其IDE

语言	IDE
C	TC2.0
C++	C++ Builder, VC++6.0, TC3.0
C#	Visual Studio.NET
Pascal	Turbo Pascal
OOPascal	Delphi
VB	VB6.0 (解释器)
Java	Eclipse, JBuilder

1.2 编译过程概述

- 从概念上来讲，一个编译程序的整个工作过程是划分成阶段进行的，每个阶段将源程序的一种表示形式转换成另一种表示形式，各个阶段进行的操作在逻辑上是紧密连接在一起的。
- 编译程序分成词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成六个阶段。另外两个重要的工作：表格管理和出错处理与上述六个阶段都有联系。

编译过程比喻

编译过程	翻译文章
词法分析	识别一个一个单词
语法分析	分析文章每句话的语法
语义分析	分析文章每句话的含义
中间代码生成	初步给出翻译结果
代码优化	对初步的翻译结果进行修改
目标代码生成	最终翻译结果

1.2.1 词法分析

- 词法分析阶段是编译过程的第一个阶段。
- 这个阶段的任务是从左到右一个字符一个字符地读入源程序，对构成源程序的字符流进行扫描和分解，从而识别出一个个单词。
- 单词：逻辑上紧密相连的一组字符，这些字符具有集体意义。

词法分析举例(源程序)

```
begin  
    var sum, first, count: real;  
    sum := first + count * 10  
end.
```

词法分析举例(分析结果)

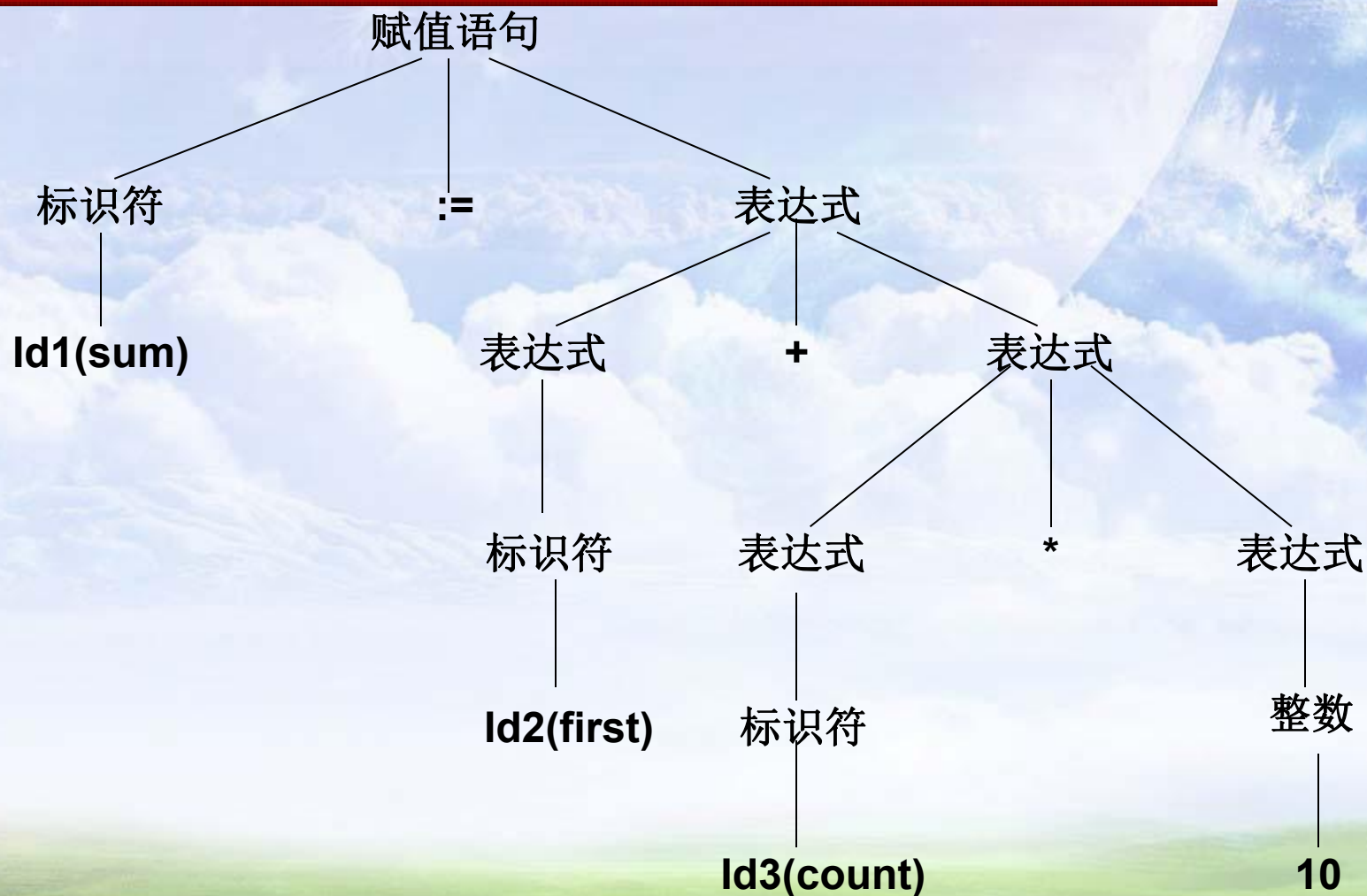
- | | |
|---------------|-------------|
| 1. 保留字 begin | 2. 保留字 var |
| 3. 标识符 sum | 4. 逗号 , |
| 5. 标识符 first | 6. 逗号 , |
| 7. 标识符 count | 8. 冒号: |
| 9. 保留字 real | 10. 分号 ; |
| 11. 标识符 sum | 12. 赋值号 := |
| 13. 标识符 first | 14. 加号 + |
| 15. 标识符 count | 16. 乘号 * |
| 17. 整数 10 | 18. 保留字 end |
| 19. 界符 . | |

```
begin  
var sum, first, count: real;  
sum := first + count * 10  
end.
```

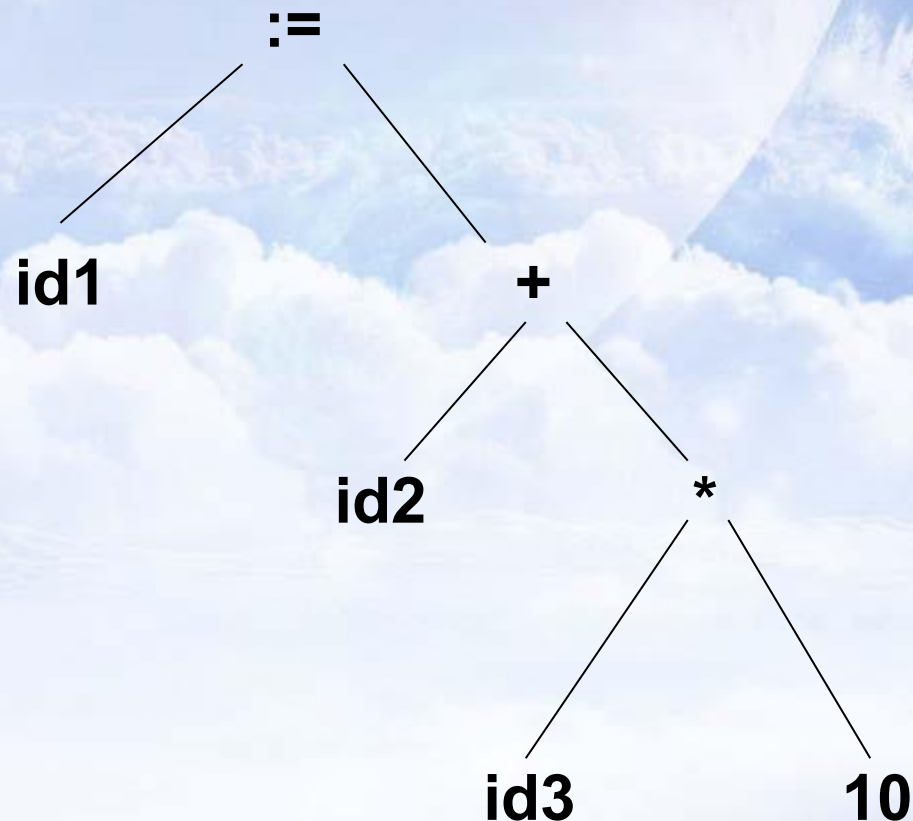

1.2.2 语法分析

- 语法分析是编译过程的第二个阶段。
- 任务：在词法分析的基础上将单词序列分解成各类语法短语，如“程序”，“语句”，“表达式”等。语法短语可表示成**语法树**。
- 依据：**语言的语法规则**，即描述程序结构的规则。通过语法分析确定整个输入串是否构成一个语法上正确的程序。语法规则通常是由递归规则定义的。

语法树举例一 (id1:=id2+id3*10)



语法树举例二 (id1:=id2+id3*10)



语法规则举例一(表达式)

- 任何标识符是表达式
- 任何常数（整常数、实常数）是表达式
- 若表达式1和表达式2都是表达式，那么：
 - 表达式1+表达式2
 - 表达式1*表达式2
 - （表达式）都是表达式

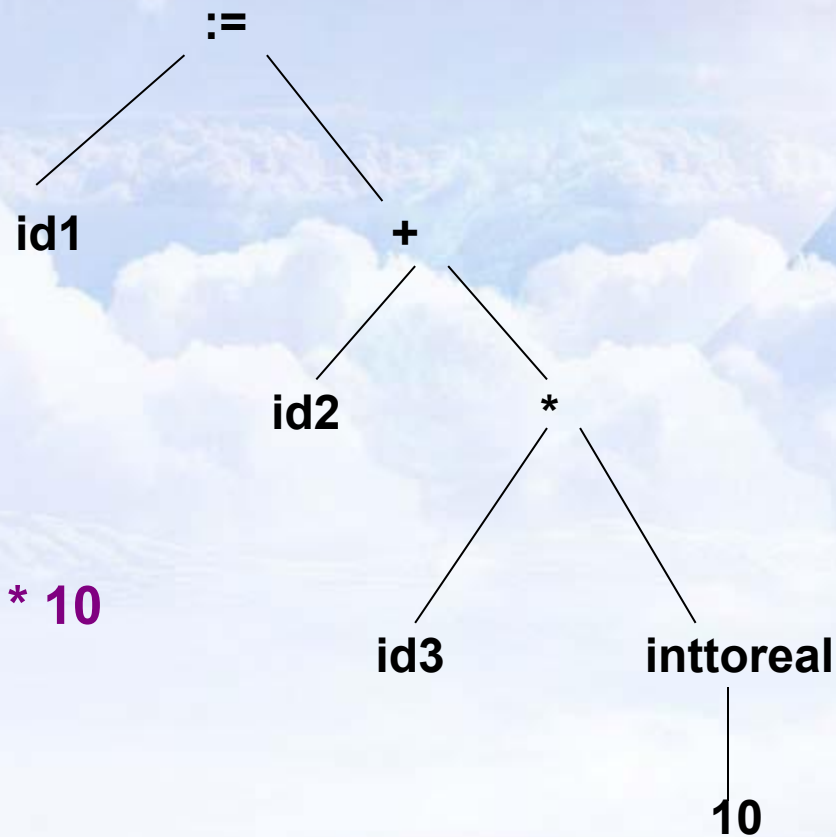
语法规则举例二(语句)

- 标识符:=表达式 是语句
- While (表达式) do 语句
If (表达式) then 语句 else 语句
都是语句。

1.2.3 语义分析

- 任务：审查源程序有无语义错误，为代码生成阶段收集类型信息。
- 例如：语义分析的一个工作是在进行类型检查，审查每个算符是否具有语言规范允许的运算对象，当不符合语言规范时，便报告错误。
- 假如在语句`sum:=first + count * 10`中，*的两个运算对象：`count`是实型，`10`是整型，则语义分析阶段进行语义审查之后，在语法分析得到的分析树上增加一个语义处理结点，表示整型变成实型的一目运算符`inttoreal`。

语义分析举例



sum:=first + count * 10

1.2.4 中间代码生成

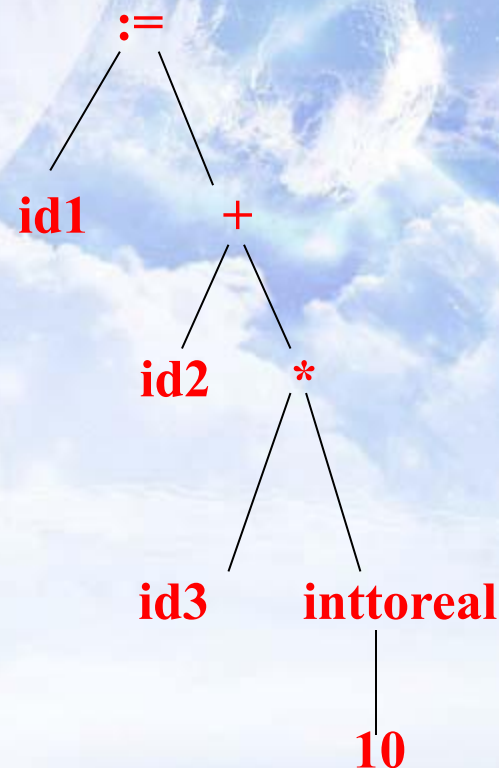
- 在进行语法分析和语义分析阶段的工作之后，有的编译程序将源程序编程一种内部表示形式，内部表示形式叫中间语言或中间代码。
- 中间代码设计的原则：
 - 容易生成
 - 容易将它翻译成目标代码
- 常用的中间代码形式四元式的语法：
(运算符，运算对象1，运算对象2，结果)

中间代码举例

- 源程序: $\text{sum} := \text{first} + \text{count} * 10$

- 生成的四元式:

1.	(inttoreal	10	-	t1)
2.	(*	id3	t1 t2)
3.	(+	id2	t2 t3)
4.	(:=	t3	- id1)



1.2.5 代码优化

- 任务：对前阶段产生的中间代码进行变换或进行改造，目的是使生成的目标代码更为高效，即省时间和省空间。
- 例：上节生成的中间代码

```
1. (inttoreal    10    -    t1)
2. (    *        id3    t1    t2)
3. (    +        id2    t2    t3)
4. (    :=       t3    -    id1)
```

- 优化为：

```
1. (*    id3    10.0    t1)
2. (+    id2    t1    id1)
```

1.2.6 目标代码生成

- 任务：把中间代码变换成特定机器上的绝对指令代码或可重定位的指令代码或汇编指令代码。
- 例：使用两个寄存器，上节的中间代码可能生成如下汇编代码：

1. MOV	id3	R2
2. MUL	#10.0	R2
3. MOV	id2	R1
4. ADD	R1	R2
5. MOV	R1	id1

1.3 编译程序的结构



1.4 编译阶段的组合

- 编译的过程可以分为前端和后端两个部分
- 前端由那些主要依赖于源语言而与目标机无关的阶段组成，通常包括词法分析、语法分析、语义分析和中间代码生成。
- 后端指那些依赖于目标机而一般不依赖源语言，只与中间代码有关的那些阶段，即目标代码生成，以及相关出错处理和符号表操作。
- 前端和后端可以任意组合实现编译程序。

前端和后端的组合

- 某一编译程序的前端加上相应不同的后端可以为不同的机器构成同一个源语言的编译程序
- 不同语言编译的前端生成某一种中间语言，在使用一个共同的后端，可以为同一机器生成几个语言的编译程序

编译过程的扫描遍数

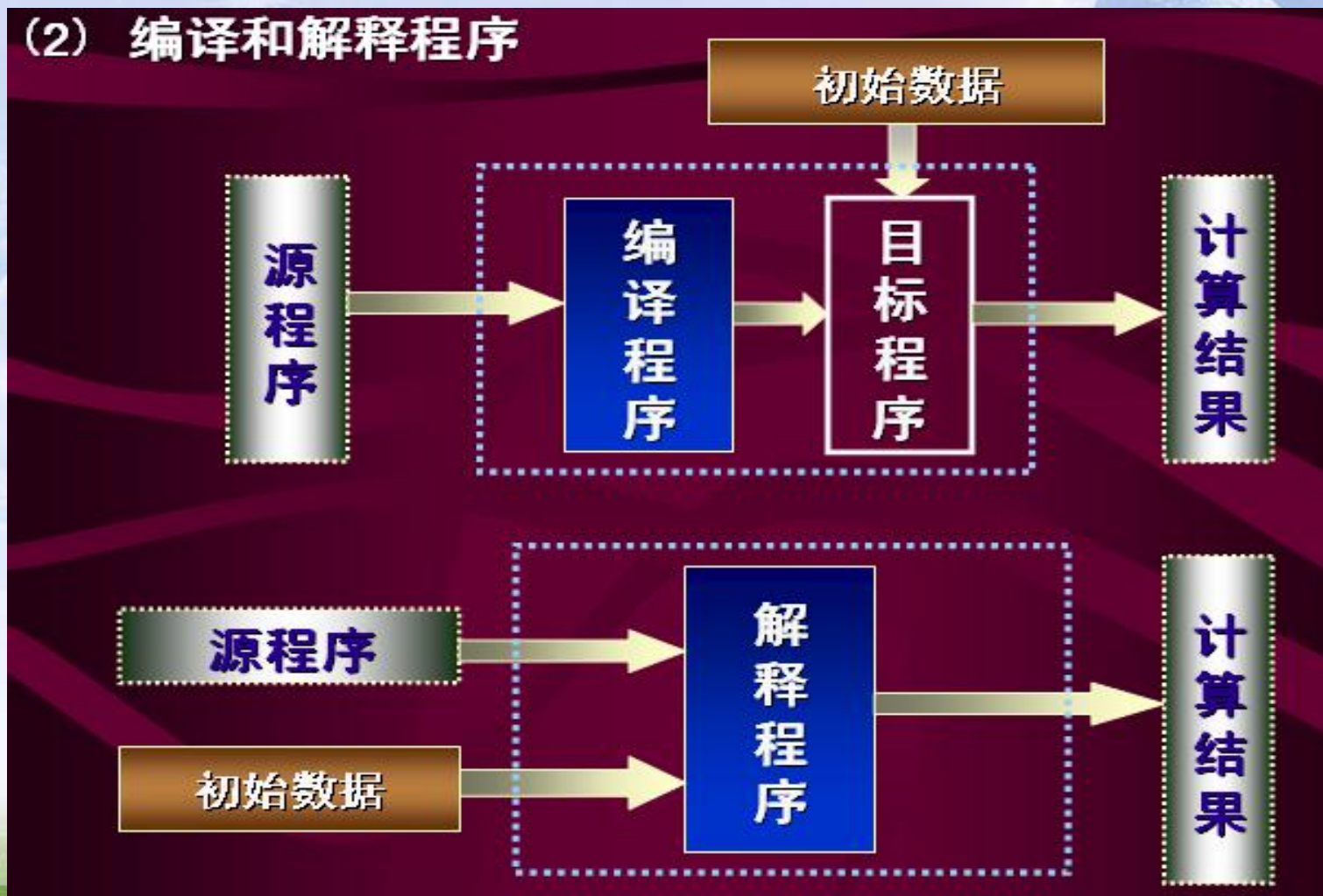
- “遍”，也称作“趟”，是对源程序或等价的中间语言程序从头到尾扫视并完成规定任务的过程。
- 一个编译过程可以由一遍，两遍或者多遍完成。
- 在实际的编译系统的设计中，编译的几个阶段的工作应该怎样组合，即编译程序究竟分成几遍，参考的因素主要是源语言和目标机器的特征。

1.5 解释程序

- 编译程序和解释程序：
 - 编译程序的**编译和运行**是两个独立分开的阶段，它**先把一个高级语言程序翻译成某个机器的汇编或二进制代码程序**，**然后这个二进制代码程序在机器上运行以生成结果**。
 - 解释程序接受某个语言的程序并立即运行这个源程序，它的工作模式是一个个的获取、分析并执行源程序语句，**一旦第一个语句分析结束，源程序便开始运行并且生成结果**。解释程序的输入包括源程序和源程序的输入数据，它不生成目标代码，直接输出结果。

解释程序的功能

(2) 编译和解释程序



编译程序和解释程序

- 编译程序和解释程序在存储组织方面的不同：
 - 编译程序处理时，在源程序被编译阶段，存储区中要为源程序和目标代码开辟空间，要存放编译用的各种各样表格，如符号表。在目标代码运行阶段，存储区中主要是目标代码和数据，编译所用的任何信息都不再需要。
 - 解释程序一般是把源程序一个语句一个语句的进行语法分析，转换为一种内部表示形式，存放在源代码区。在解释程序工作的整个过程中，源程序、符号表等内容始终存放在存储区中，并且存放格式要设计得易于使用和修改。

编译程序的运行区内容



编译时

运行时

解释程序的存储区内容

解释系统
源程序
工作单元及名字表
标号表
缓冲区（输入输出）
栈区

Java平台和虚拟机

• Java平台

- 主要由Java虚拟机（JVM，Java Virtual Machine）和Java应用程序接口（Java API）两部分组成；
- Java虚拟机易于移植到不同硬件的平台上，是Java平台的基础；
- Java应用程序接口提供了丰富的Java资源；

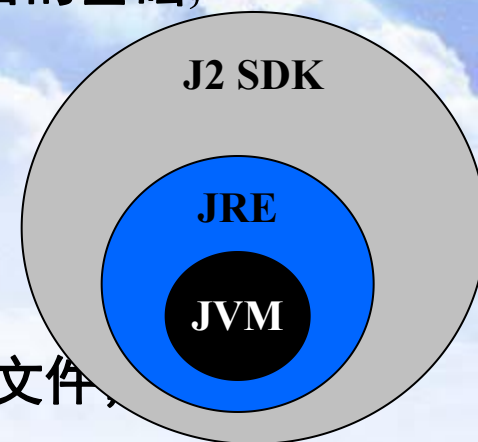
• Java 虚拟机

- 是一种利用软件方法来实现硬件功能的虚拟计算机；
- JVM的实现—Java运行时系统；

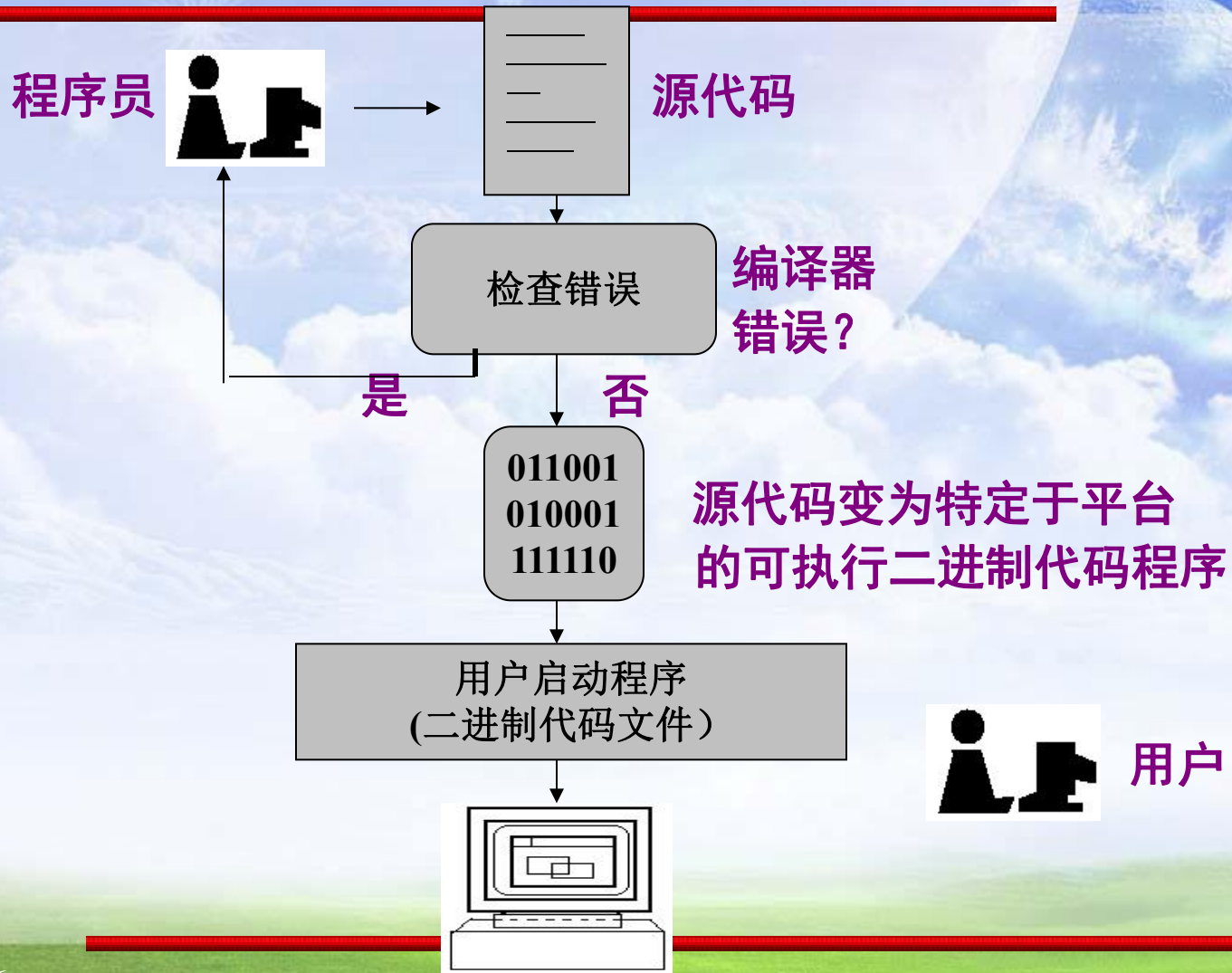
嵌入Java运行时系统的应用程序，可以执行Java字节码文件；

– Java工作原理

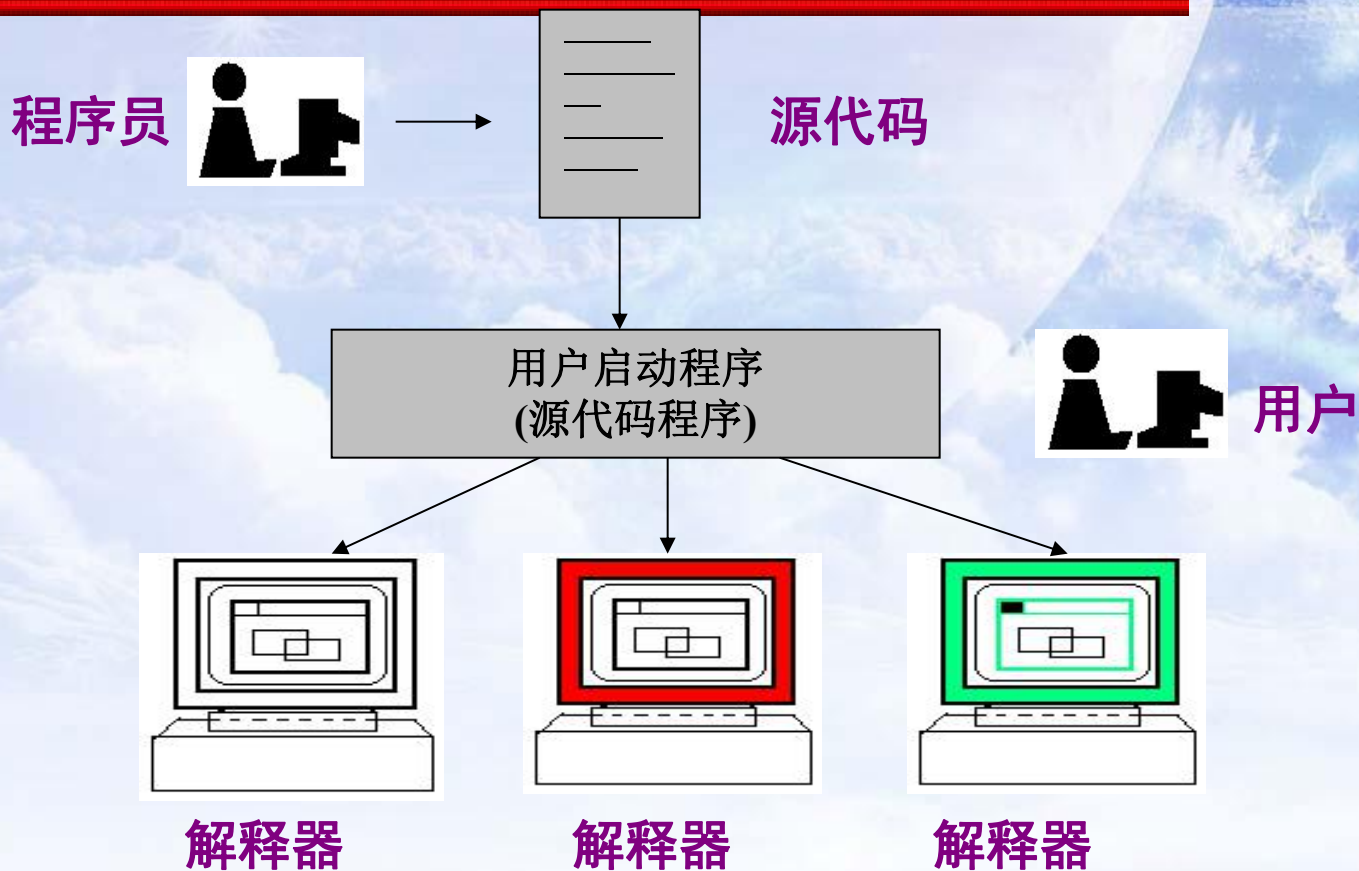
- **编译器**：对源代码进行半编译，生成与平台无关的字节码文件；
- **解释器**：分布在网络中不同的操作系统平台上，用于对字节码文件半解释执行；



JAVA编译执行的程序

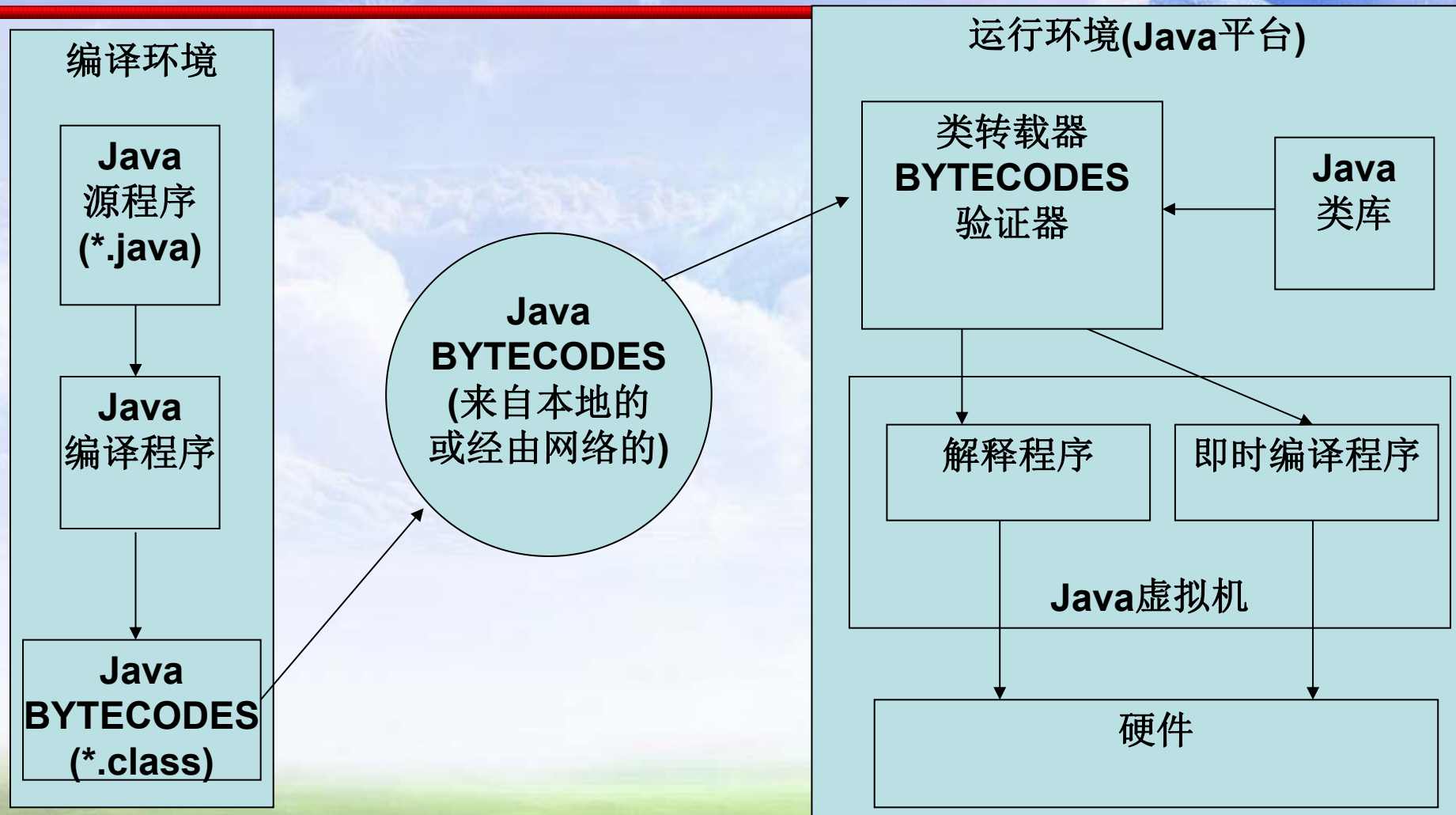


解释执行程序



**解释器检查是否有错误和安全问题，
在当前的平台上解释并运行此程序**

Java语言环境

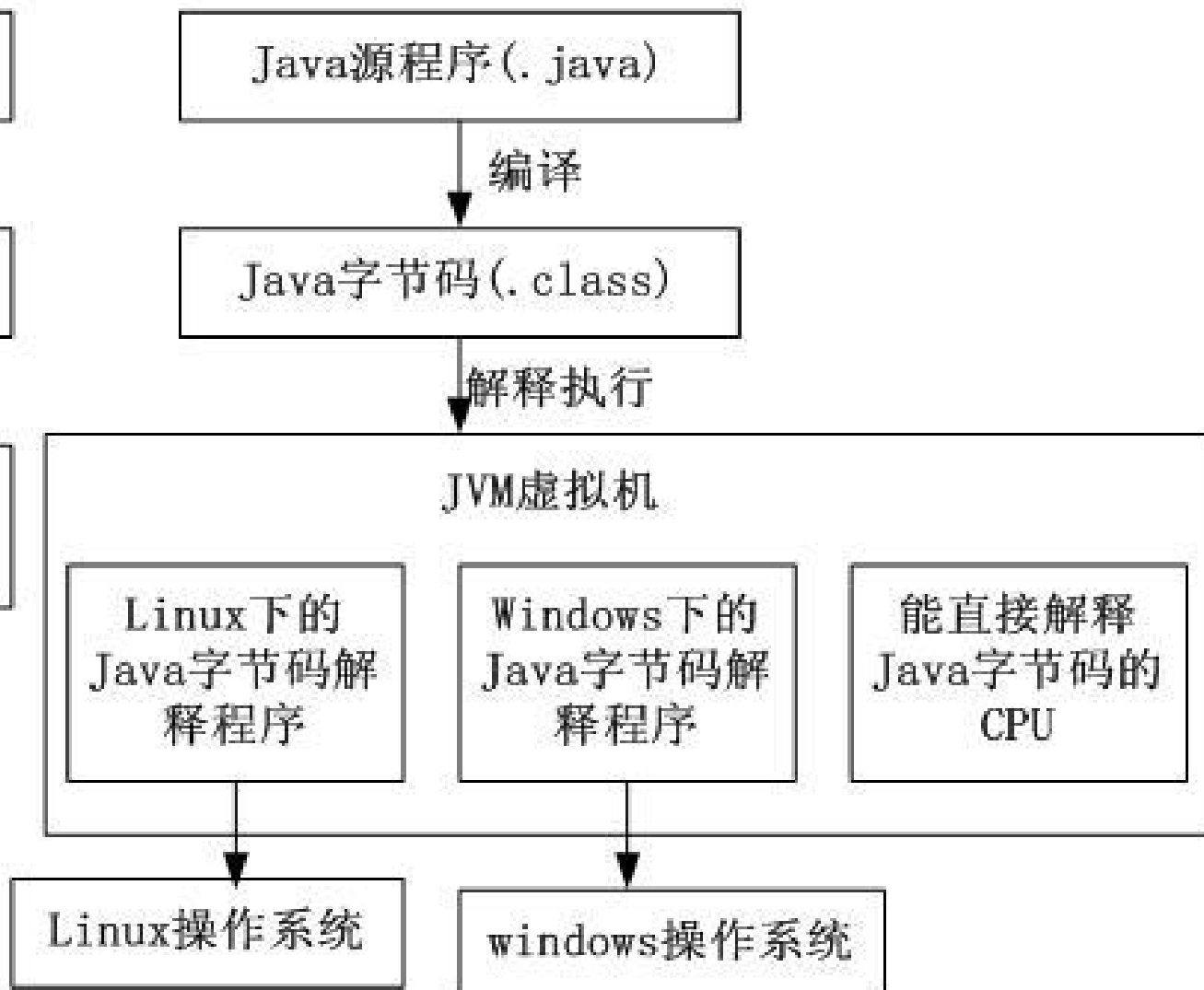


Java程序可移植性原理

windows下C语言编程过程



Java语言编程过程



1.6 处理源程序的软件工具

- 语言的结构化编辑器
- 语言程序的调试工具
- 程序格式化工具
- 语言程序测试工具
 - 静态分析器
 - 动态测试器
- 程序理解工具
- 高级语言之间的转换工具

1.7 程序设计语言范型

- **强制（命令）式语言(imperative language)**
 - 也称过程式语言。
 - 代表语言：C, FORTRAN, Pascal
 - 这种语言是面向动作的，即一个计算过程看作是一系列动作。其动作是命令驱动，以语句形式表示。
 - 一个强制式语言程序由一系列的语句组成，每个语句的执行引起若干存储单元中的值的改变。
 - 强制式语言执行的解释与冯诺依曼机的体系结构对应：顺序执行指令，通过机器状态（内存，各种寄存器和外存的内容）的改变理解指令执行的含义。

程序设计语言范型

- 函数式语言(functional language)
 - 也成为应用式语言。
 - 代表语言：ML, LISP
 - 函数式语言更注重程序所表示的功能，而不是一个语句接一个语句地执行。
 - 程序的开发过程是从前面已有的函数出发构造出更复杂的函数，对初始数据集进行操作，直至最后形成的函数可以用于从初始数据计算出最终的结果。
 - 程序的执行过程是：执行一个个函数施用在数据上的变换最终得到结果。

程序设计语言范型

- 基于规则（逻辑）的语言
 - 这类语言的语法形式通常为：
条件1->动作1
条件2->动作2
条件3->动作3
 - 程序的执行过程是：检查一定的使能条件，当它满足时，则执行适当的动作。
 - 最常见的这类语言是PROLOG，也成为逻辑程序设计语言。

程序设计语言范型

- 面向对象语言

- 主要特点是提供抽象数据类型，支持封装性、继承性和多态性。
- 主要语言：Ada, C++, Java

总结

什么是编译程序

编译过程（六个阶段）

编译程序的结构（六个模块、表格、出错处理）

编译阶段的组合（前端、后端）

解释程序（编译和解释的区别）

一些软件工具

程序设计语言范型

作业

- 课本第12页 第4题。
-