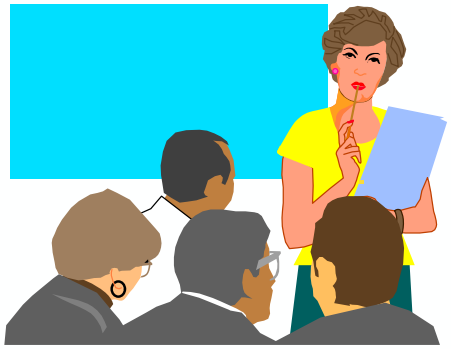
编译技术



张 莉 教授
杨海燕 讲师

2022.9-2023.1

- **Hi,** 各位同学，大家好！
- 欢迎来到编译技术课，很高兴和大家一起学习、探讨编译技术。

编译，这个词大家一定不陌生。大家用过编译器吗？

- 你能告诉我什么是编译吗？
- 所有的程序都需要通过编译才能运行吗？
- 你认为这门课会讲什么？
- 你希望有什么收获？

编译和系统安全

- 操作系统安全
- 编译安全
- 程序安全
- 软件系统安全

- 航空安全

- 卡脖子问题：

一个简单C程序的运行时结构

- #include <stdio.h>
- #include <string.h>

```
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,“bbbb”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

运行结果？

```
#include <stdio.h>
#include < string.h >
void fun1( )
{
 int m=10;
 char  num[4];
  strcpy(num,“bbbbbbbbbbbb\x0F\x10\x40\x00”);
}
void fun2( )
{
 printf (“You were attacked!!!\n”);
}
int main()
{
  fun1();
 return 0;
}
```

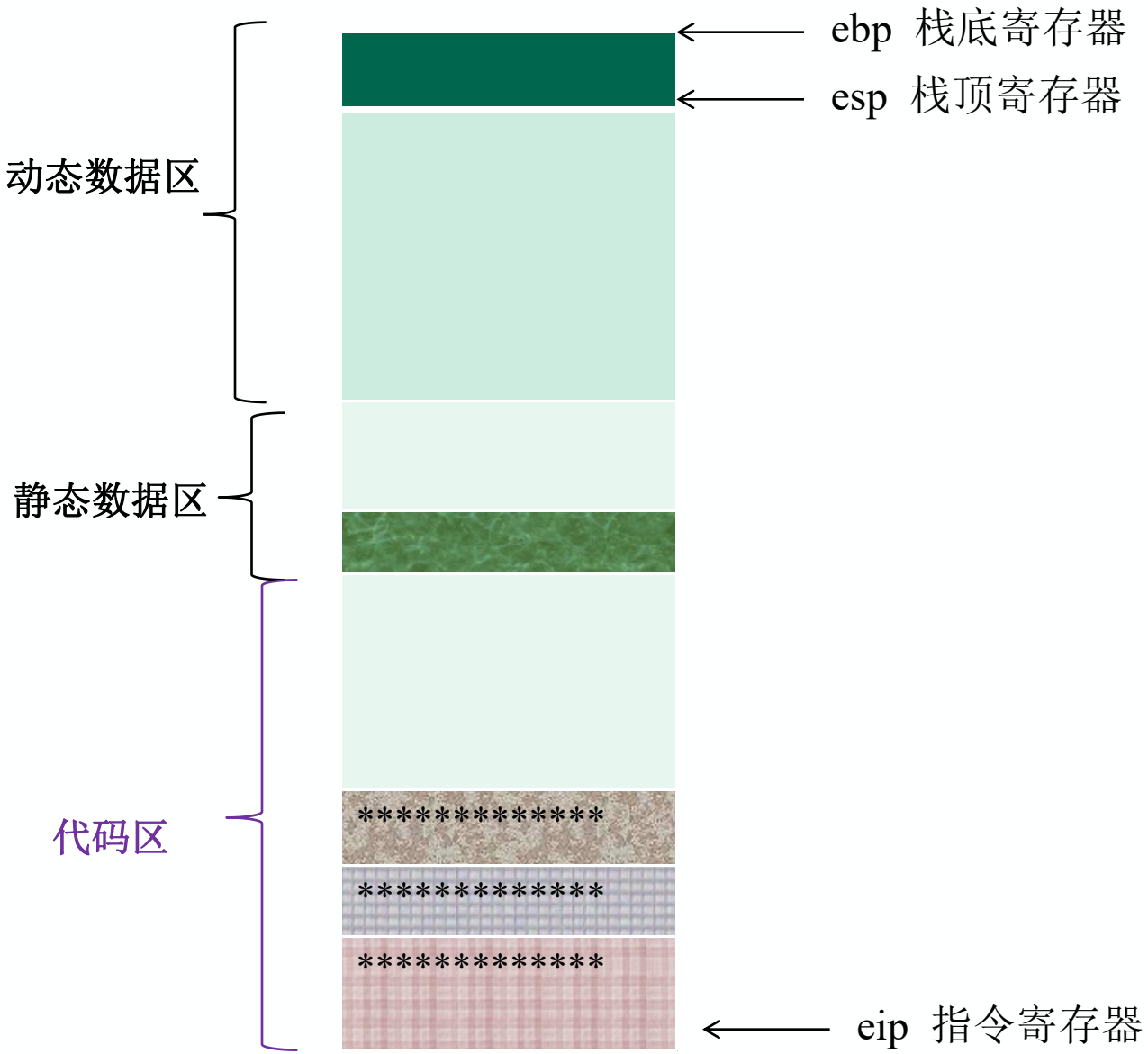
运行结果？

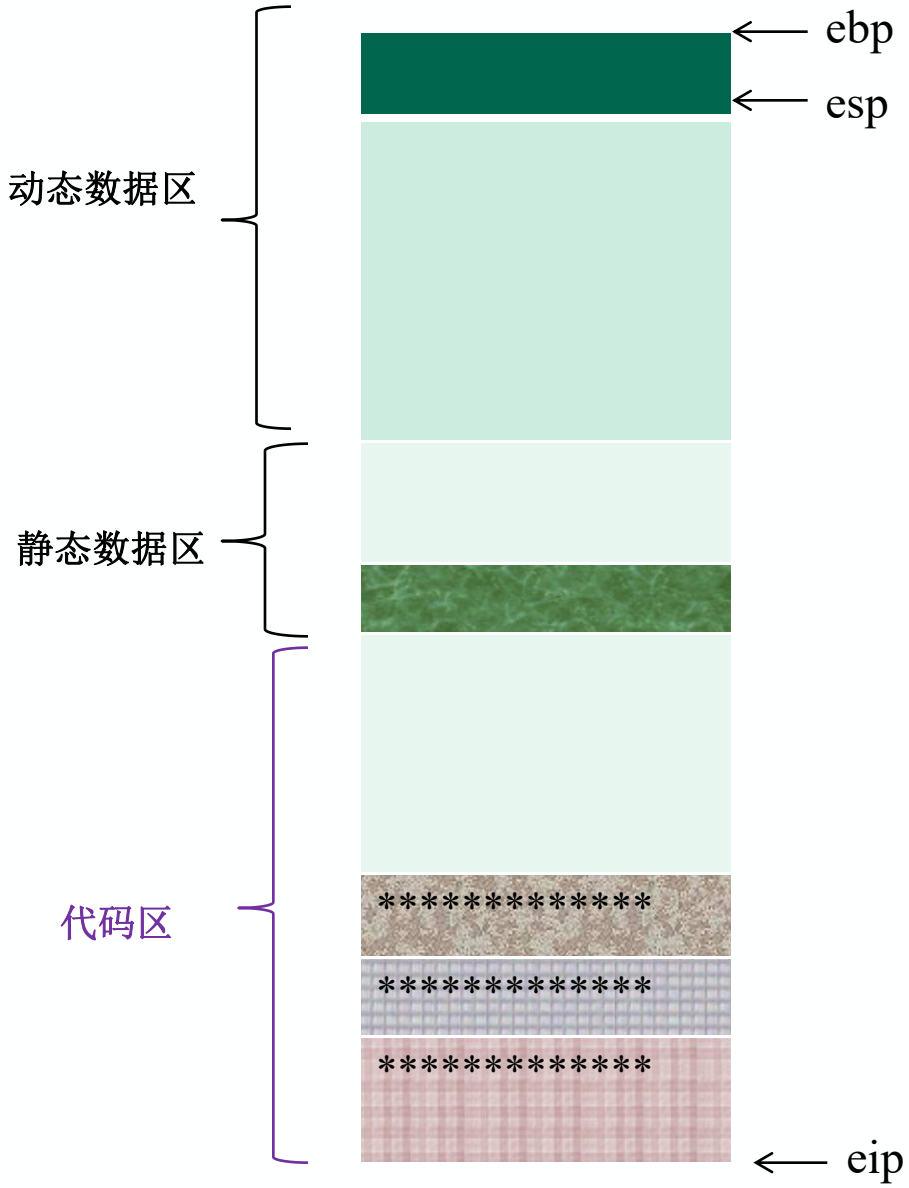
```
#include <stdio.h>
#include < string.h >
void fun1( )
{
 int m=10;
 char  num[4];
  strcpy(num,“bbbb”);
}
void fun2( )
{
 printf (“You were attacked!!!\n”);
}
int main()
{
  fun1();
 return 0;
}
```

```
#include <stdio.h>
#include < string.h >
void fun1( )
{
 int m=10;
 char  num[4];
  strcpy(num,“bbbbbbbbbbbb\x0F\x10\x40\x00”);
  printf(“m=%d\n”, m);
}
void fun2( )
{
 printf (“You were attacked!!!\n”);
}
int main()
{
  fun1();
 return 0;
}
```

运行结果？

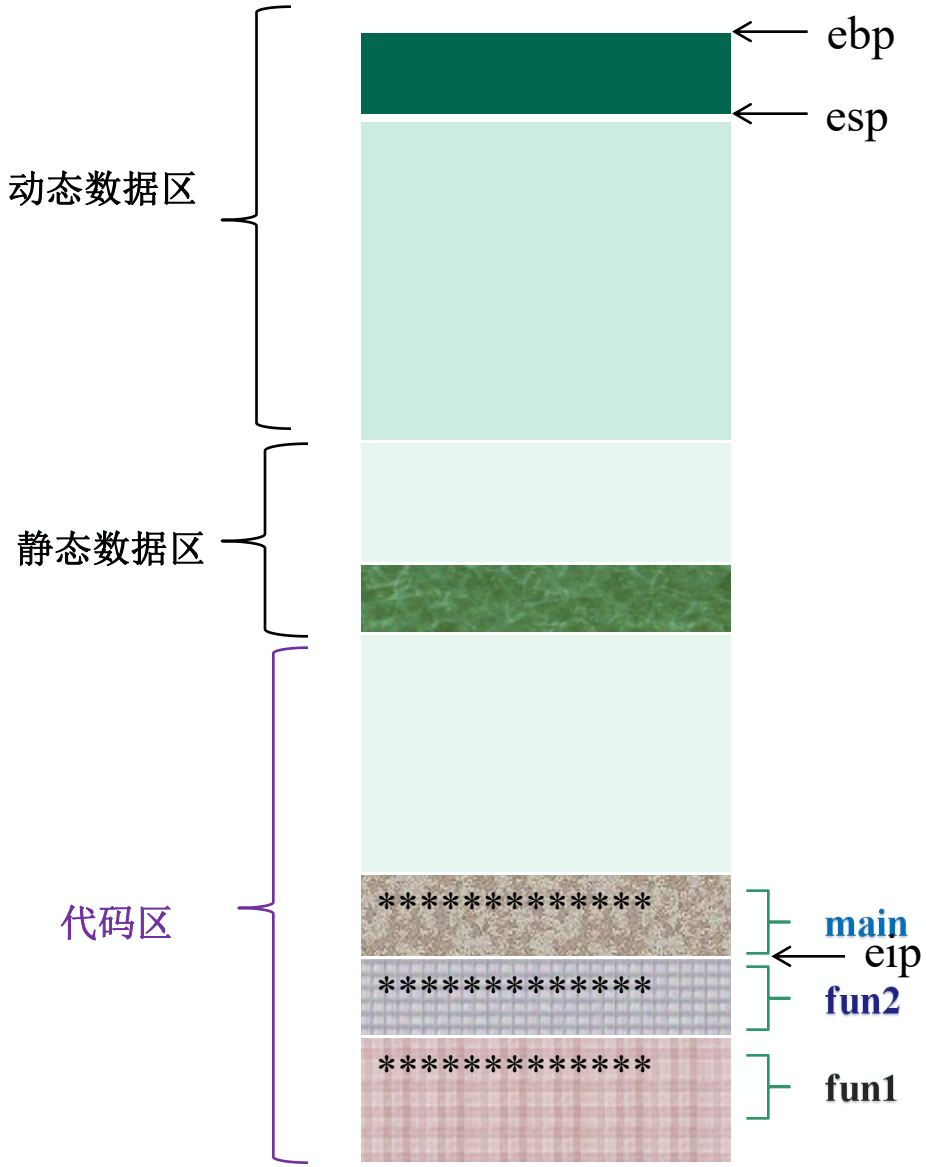
```
#include <stdio.h>
#include < string.h >
void fun1( )
{
 int m=10;
 char  num[4];
  strcpy(num,“bbbb”);
  printf(“m=%d\n”, m);
}
void fun2( )
{
 printf (“You were attacked!!!\n”);
}
int main()
{
  fun1();
 return 0;
}
```


ebp 栈底寄存器
esp 栈顶寄存器
动态数据区
静态数据区
代码区
****
****
****
eip 指令寄存器




```
#include <stdio.h>
#include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,“bbbb”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

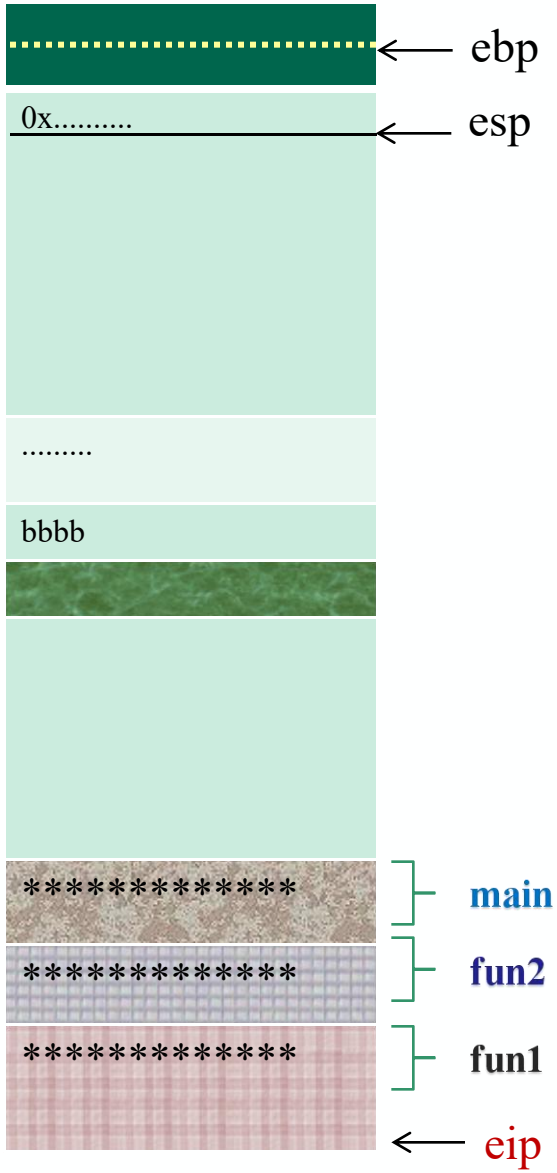
初始化：程序加载




```
• #include <stdio.h>
• #include <strinf.h>
void fun1( )
{
 int m=10;
 char  num[4];
  strcpy(num,“bbbb”);
}
void fun2( )
{
 printf (“You were attacked!!!\n”);
}
int main()
{
  fun1();
 return 0;
}
```

程序执行，调用fun1

fun函数执行后的返回地址



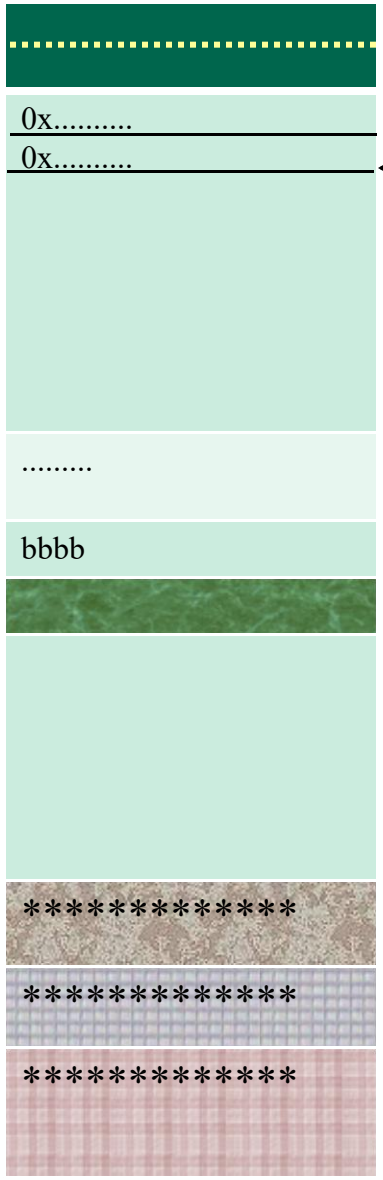
- #include <stdio.h>
- #include <strinf.h>

```
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,“bbbb”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

进入fun1

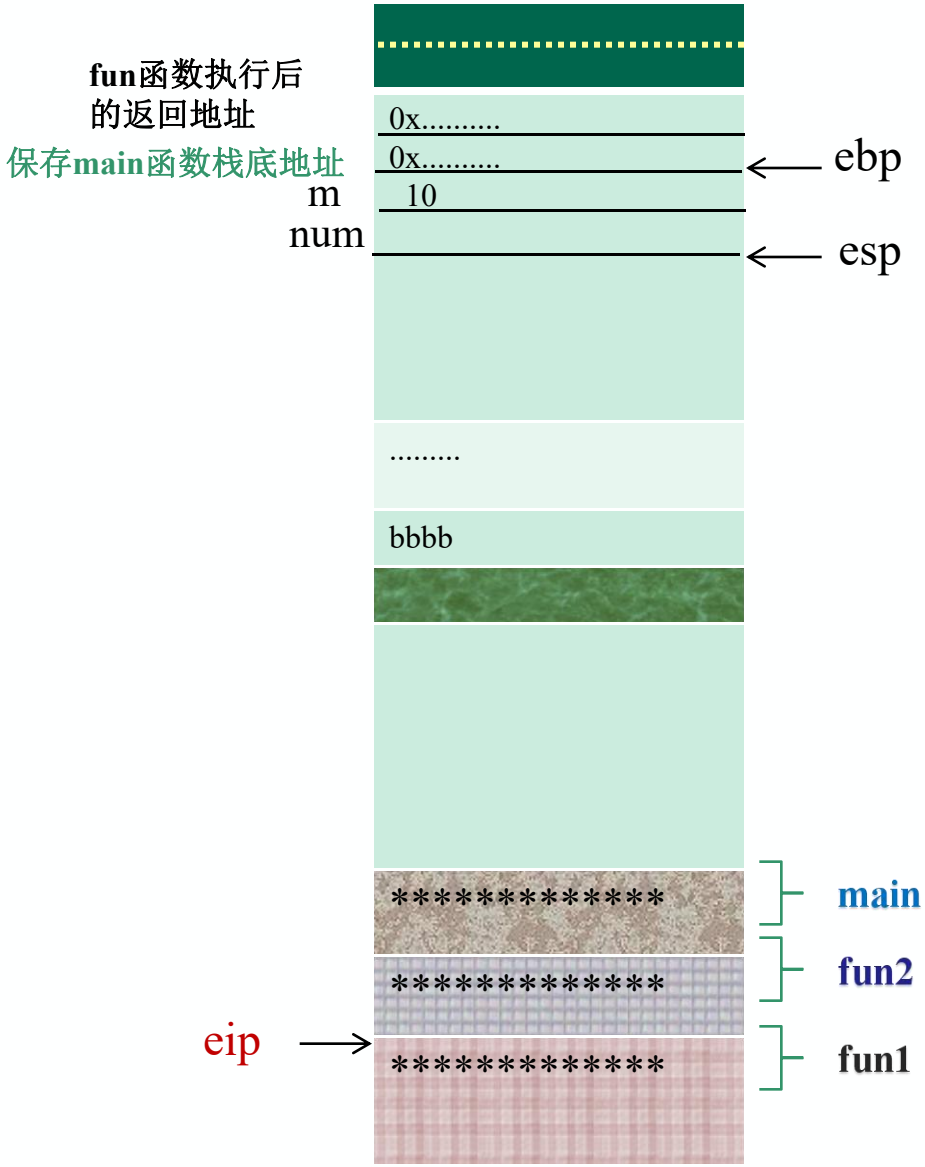
fun函数执行后
的返回地址

保存main函数栈底地址



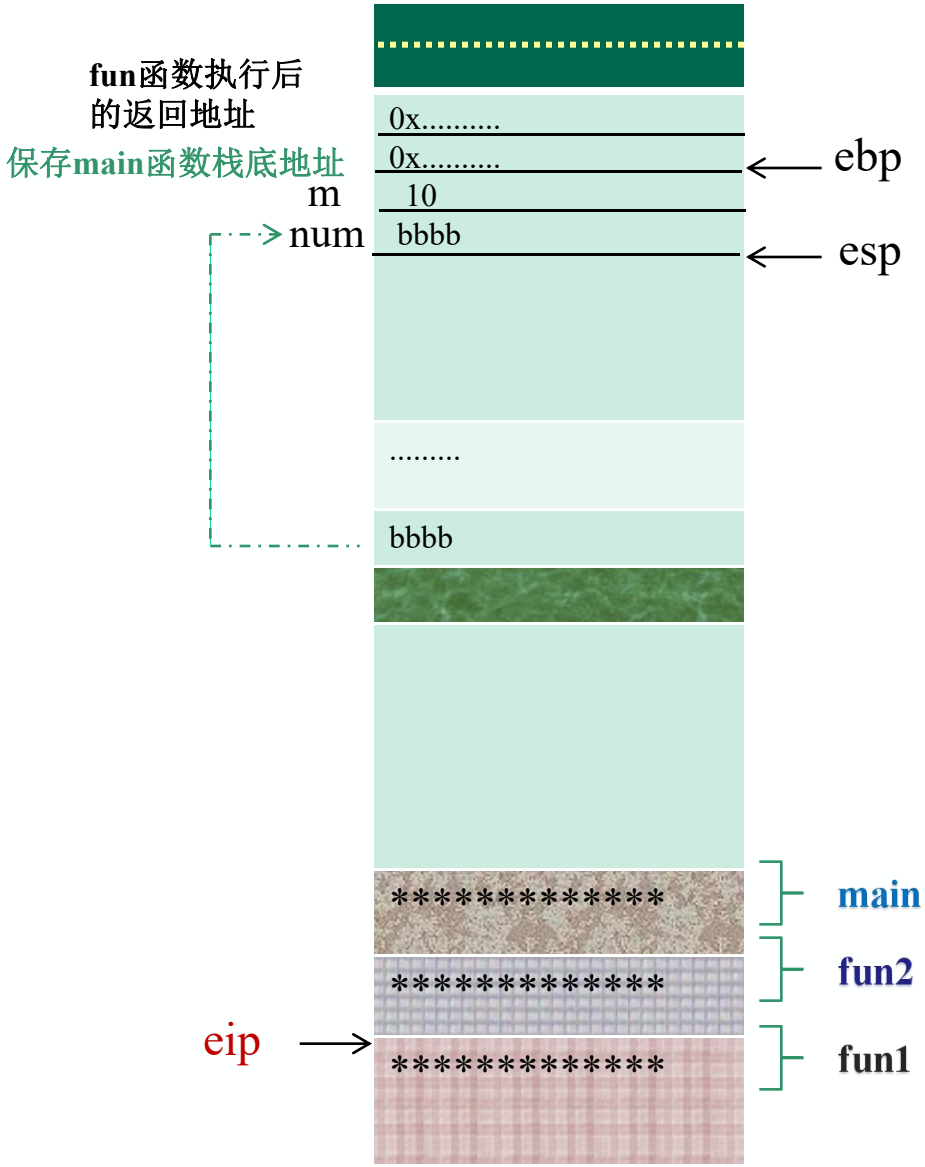

```
• #include <stdio.h>
• #include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,“bbbb”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

执行fun1中指令



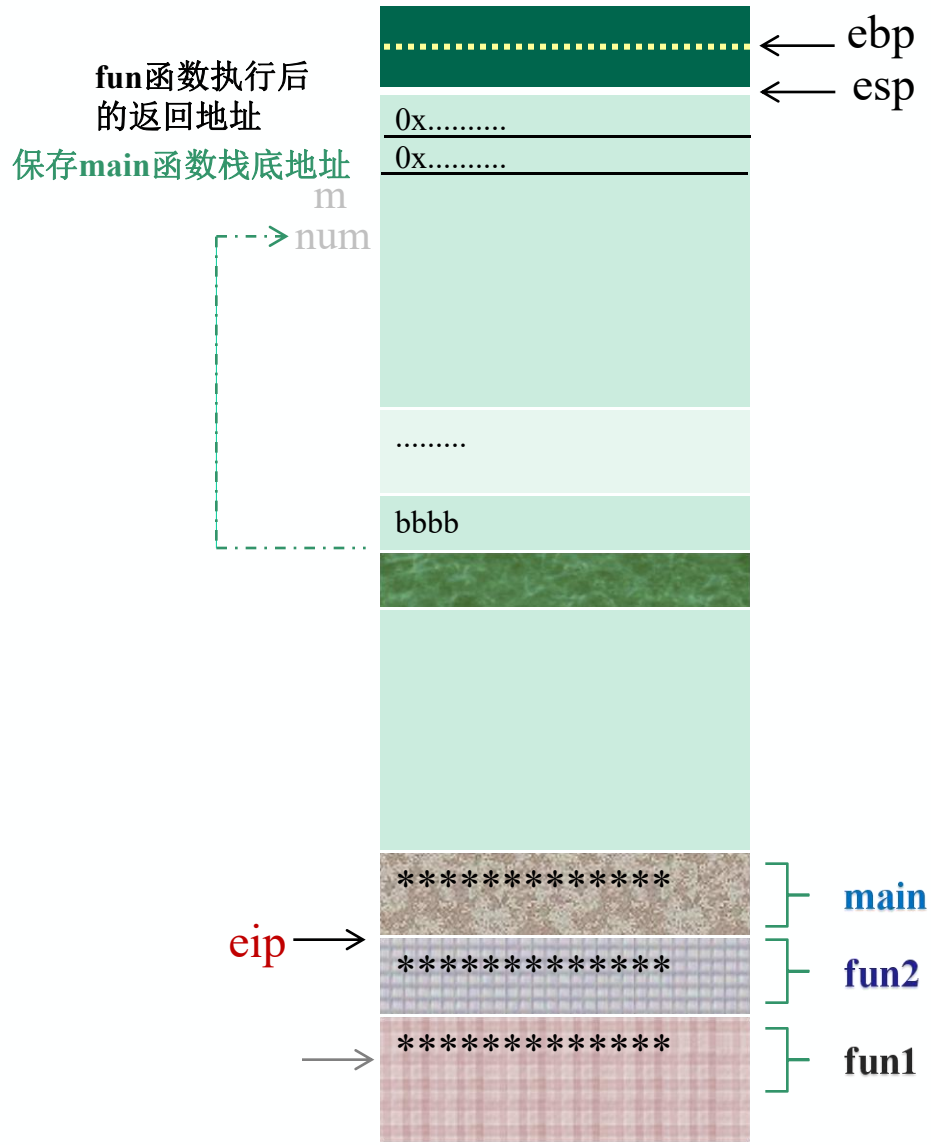

```
• #include <stdio.h>
• #include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,“bbbb”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

执行fun1中指令



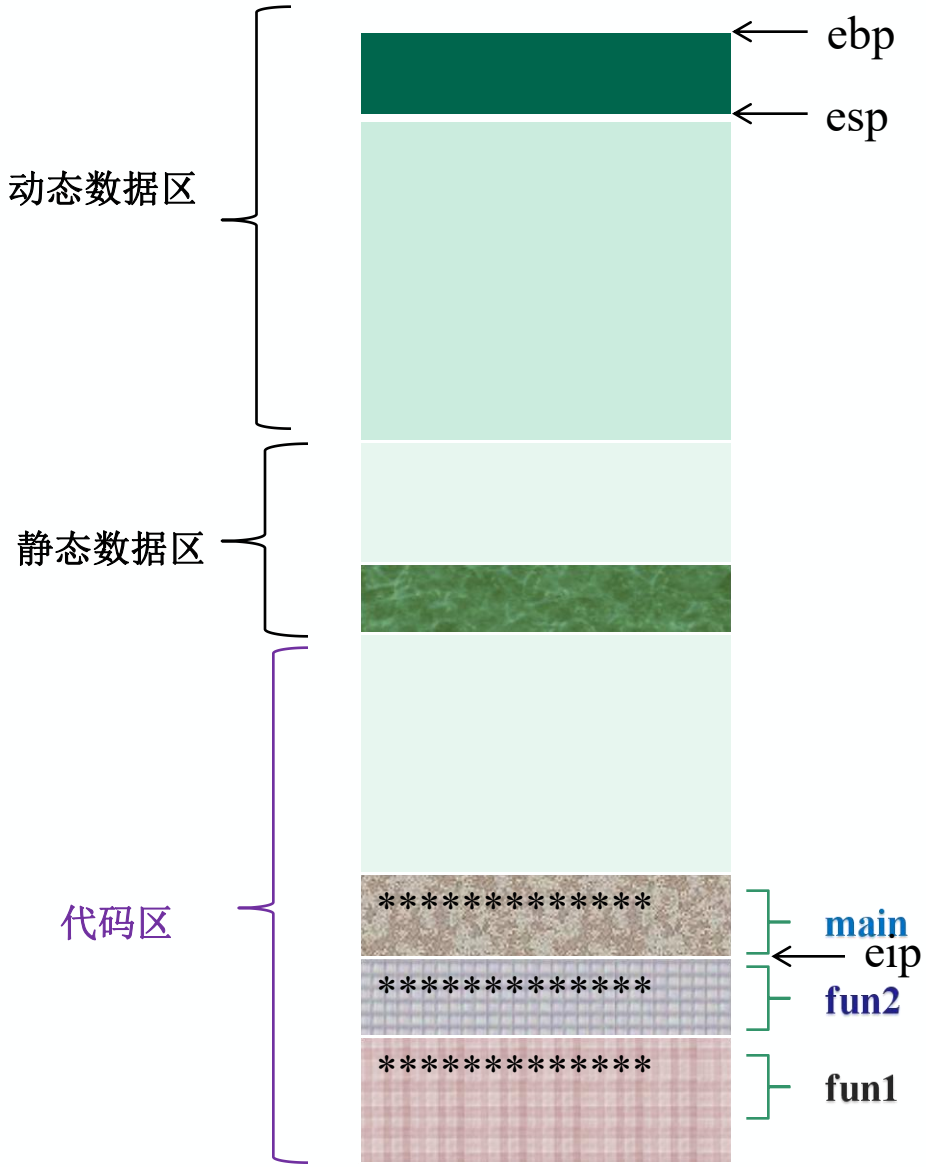

```
#include <stdio.h>
#include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,"bbbb");
}
void fun2( )
{
  printf ("You were attacked!!!\n");
}
int main()
{
   fun1();
  return 0;
}
```

fun1结束，返回main函数




```
#include <stdio.h>
#include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,“bbbb”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

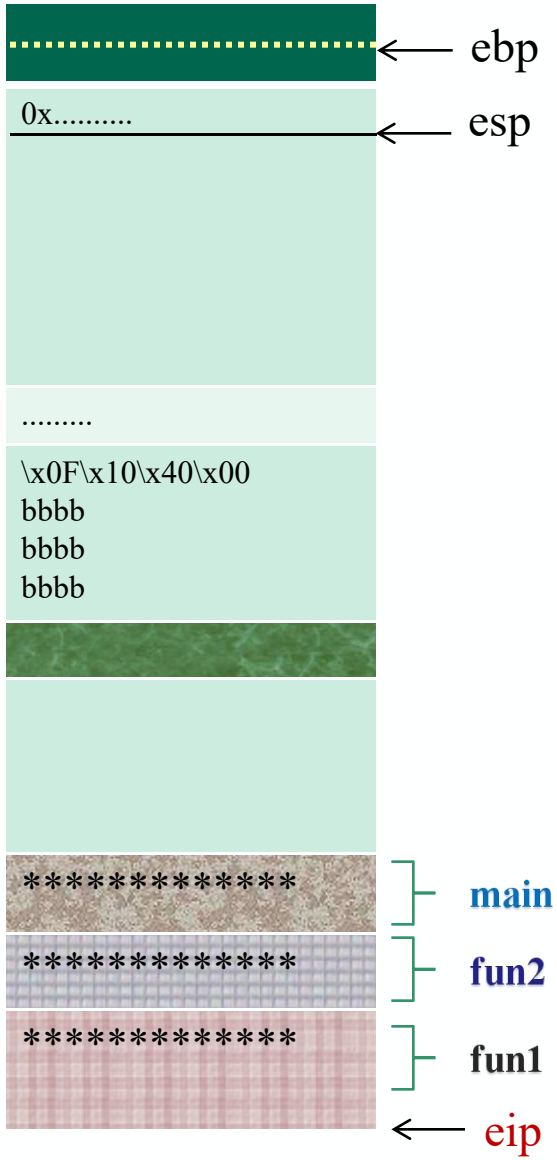
第二个程序-初始化：程序加载




```
• #include <stdio.h>
• #include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
strcpy(num,“bbbbbbbbbbbb\x0F\x10\x40\x00”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

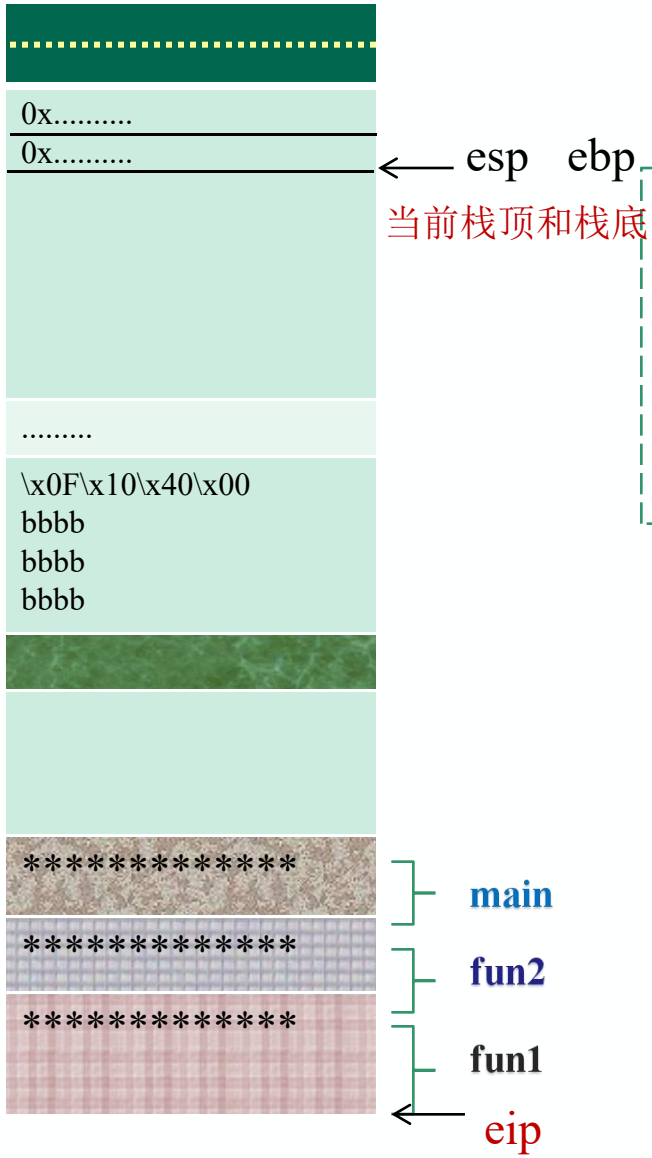
程序执行，调用fun1

fun函数执行后的返回地址



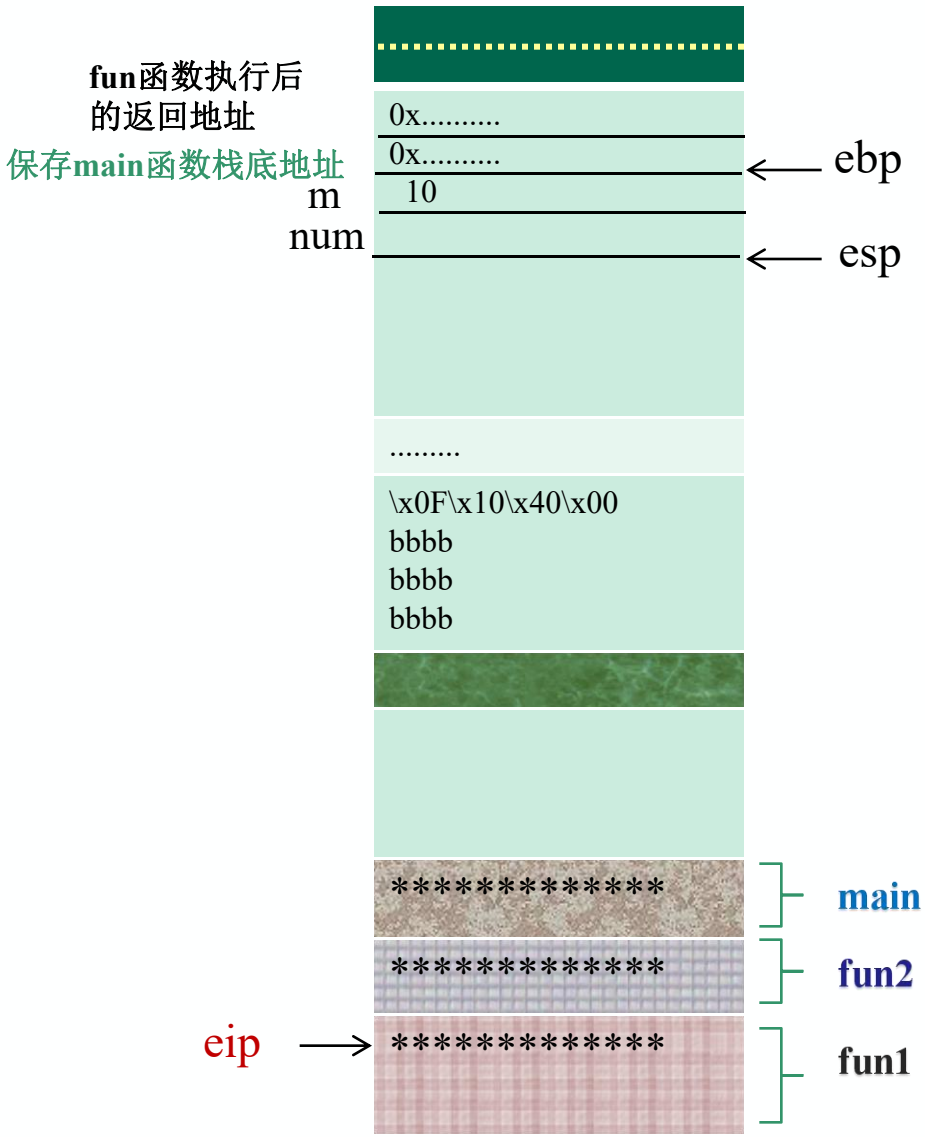

```
• #include <stdio.h>
• #include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
strcpy(num,“bbbbbbbbbbbb\x0F\x10\x40\x00”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

第二个程序：进入fun1




```
• #include <stdio.h>
• #include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
strcpy(num,“bbbbbbbbbbbbb\x0F\x10\x40\x00”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

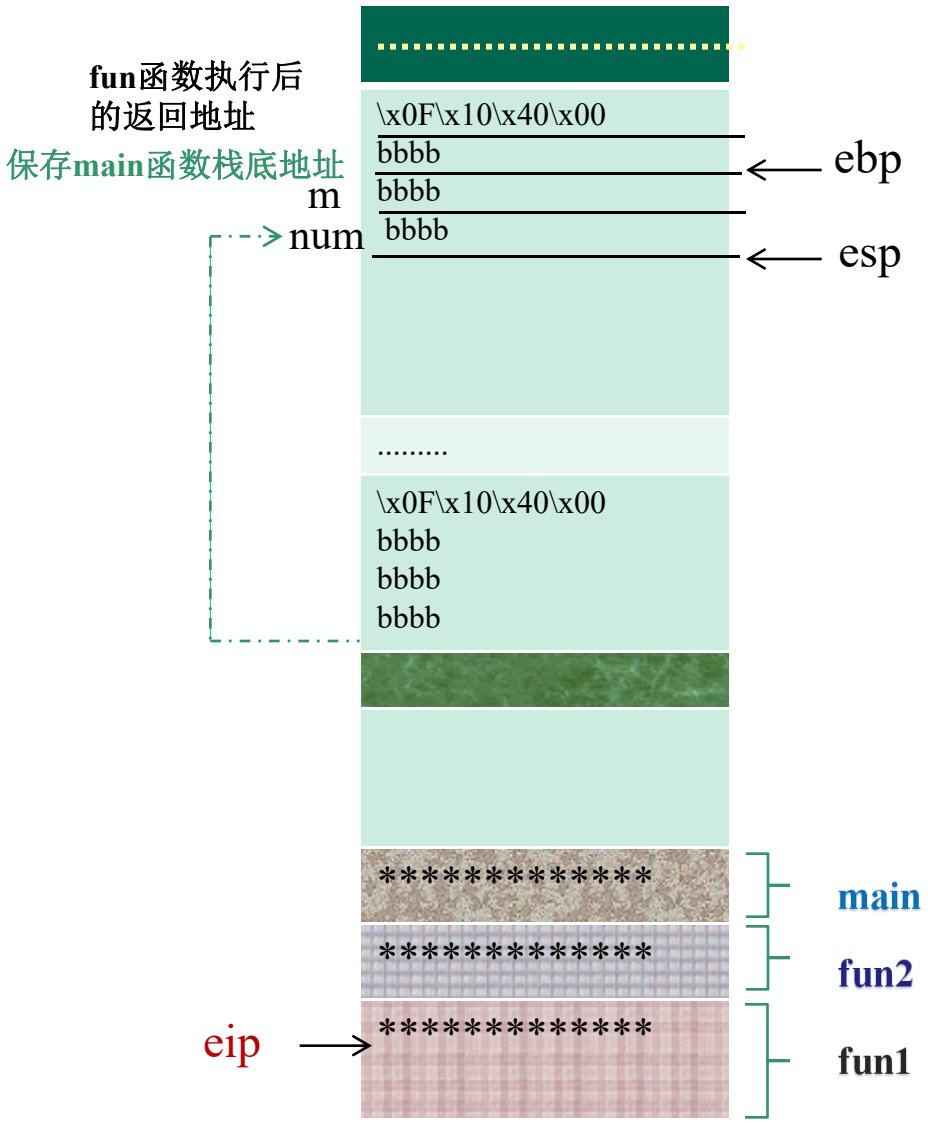
第二个程序：执行fun1中指令



- #include <stdio.h>
- #include <strinf.h>

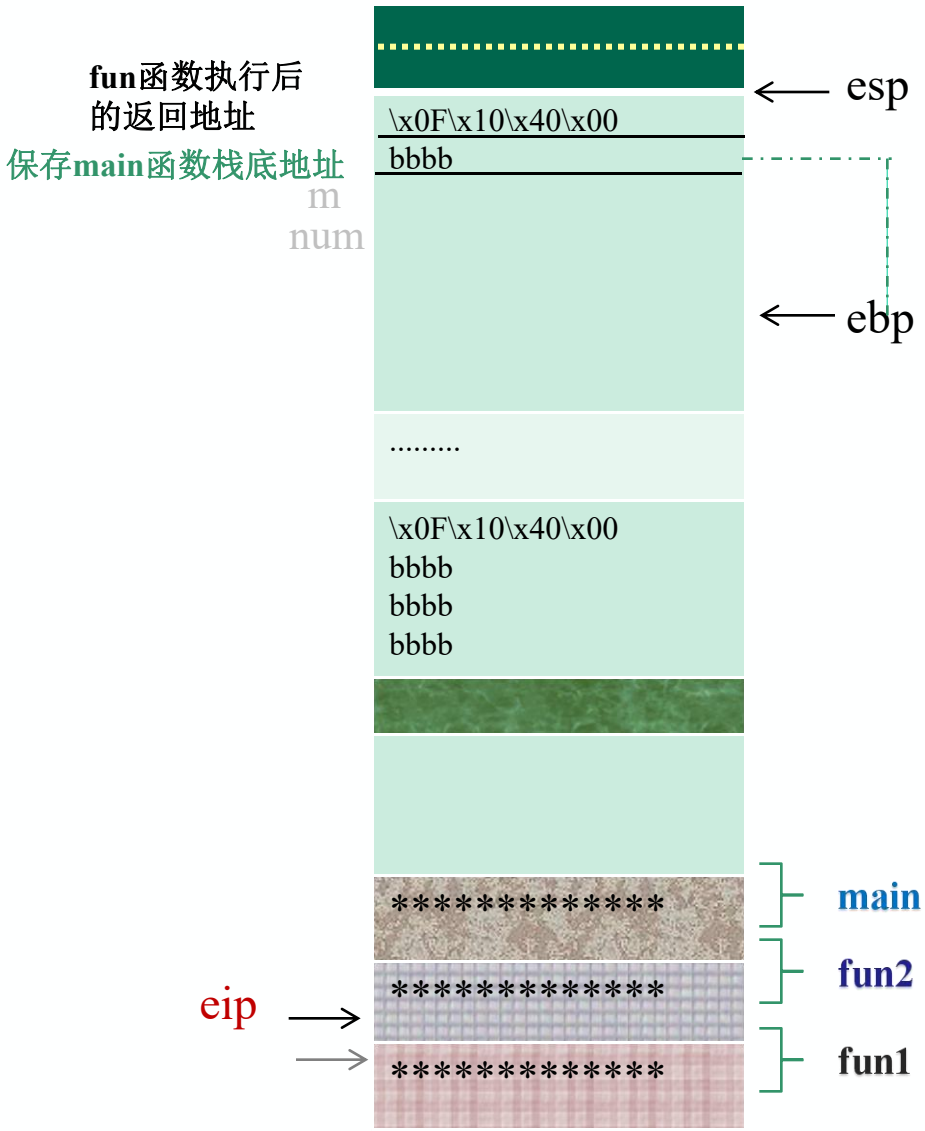
```
void fun1( )
{
  int m=10;
  char  num[4];
strcpy(num,“bbbbbbbbbbbb\x0F\x10\x40\x00”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

第二个程序：执行fun1中指令




```
• #include <stdio.h>
• #include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
strcpy(num,“bbbbbbbbbbbb\x0F\x10\x40\x00”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

fun1结束，返回main函数




```
#include <stdio.h>
#include <strinf.h>
void fun1( )
{
  int m=10;
  char  num[4];
   strcpy(num,“bbbb”);
}
void fun2( )
{
  printf (“You were attacked!!!\n”);
}
int main()
{
   fun1();
  return 0;
}
```

- 在C语言中，栈的方向是从高地址向低地址延伸，而数组中数据在栈中的存储方向正好相反。
- 字符串拷贝等数组操作是不对数据长度做审核的。如果实际的数据长度超过了栈中预留的空间，就会将栈中其他数据覆盖。---“栈溢出”。
  - 可能导致不可预测的错误
  - 也可能导致一个精心策划的执行流程发生改变
- 因此，是否能够对自己所写程序的运行时状态做到心中有数，是能否写出高质量、安全代码的前提保证。进一步再考虑尽量不要给黑客留漏洞。

- 因此，编译课：
  - 帮助学生了解程序运行的机理
  - 帮助学生了解程序运行时的存储管理、运行时状态
  - 有利于学生编写高质量的程序
  - 有利于学生使用基于编译的调试工具

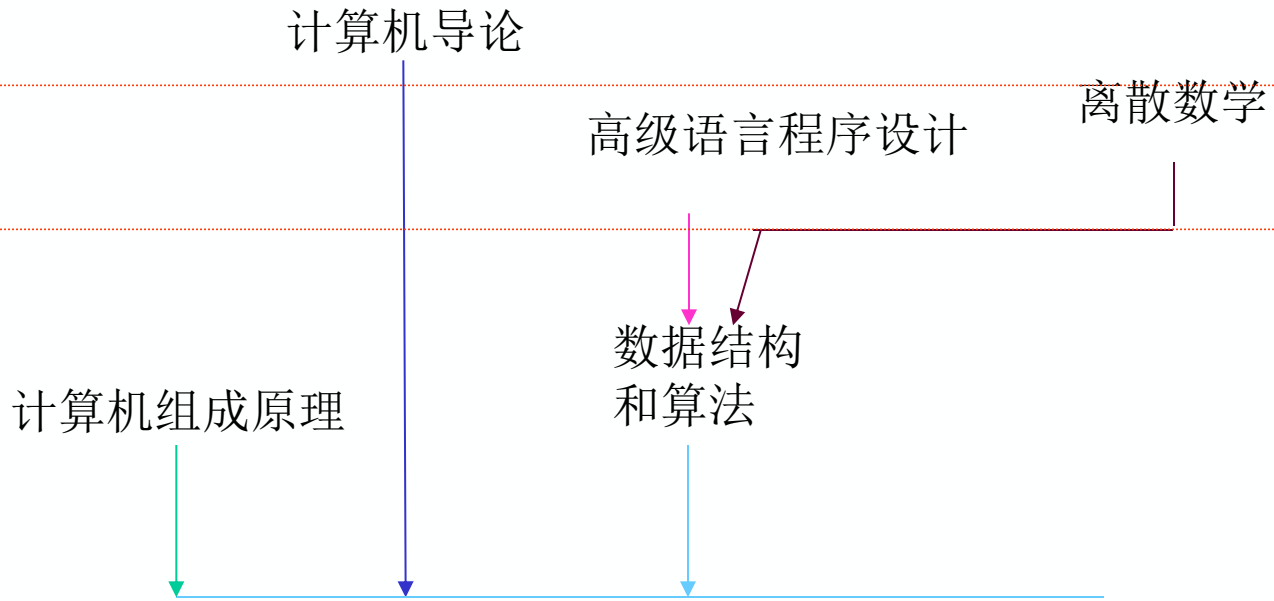
  - 计算机专业、软件工程专业的学生都应该学习编译

课程目的

目的:

- 掌握编译的基本理论、常用的编译技术，了解编译过程及编译系统的构造（结构和机理）。
- 掌握形式语言基础知识，了解高级程序设计语言运行激励
- 能运用所学技术解决实际问题，能独立编写一个小型编译系统。（课程设计大作业，可选不同难易系数）

前序课程

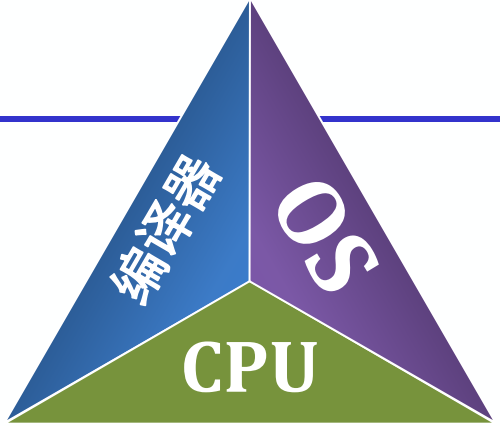


编译技术

你的收获

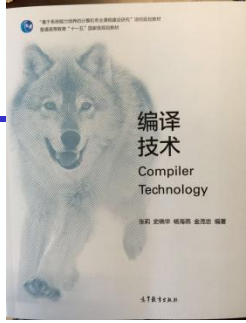
- 掌握编译系统的功能、原理和构造方法（不局限于特定的程序设计语言和目标机）；理解高级程序设计语言的运行机理。
- 具备完整小型编译系统的设计、实现和测试的能力；理解将一种程序设计语言转换/翻译为另一种的程序等价转化的理论和方法。
- 大部分同学具备针对特定微处理器架构和指令集架构进行编译优化的能力；理解程序的安全问题。
- 掌握形式语言以及自动机理论在编译器前端自动生成中的应用；理解不同方法和工具的作用和局限性。
- 了解编译技术的前沿和新发展动向

计算机专业的3问



- 3大核心技术
  - CPU、操作系统和编译器
- 3大必修课程
  - 计算机组成、操作系统、编译技术
- 3个基本问题
  - 能够开发1个CPU吗？
  - 能够开发1个操作系统核心吗？
  - 能够开发1个编译器吗？

来自小鹏院长的ppt：结合学院本科培养目标



- 教材和参考书
  - 张莉，杨海燕，史晓华、金茂忠，《编译技术》，高等教育出版社，2016,9
  - 张莉，杨海燕，史晓华、金茂忠、高仲仪，《编译原理及编译程序构造》，清华出版社，2011,6。
  - Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers—Principles, Techniques, and Tools. 机械出版社（翻译版），2003
  - A. W. Apple, J. Palsberg 著，《Modern Compiler Implementation in Java》
  - Kenneth C. Louden 著，《编译原理及实践》，机械工业出版社，2000,3。
  - 陈火旺，刘春林，谭庆平等 编著，《程序设计语言编译原理》，国防工业出版社出版，2002.1 (第3版）。

特点

- 经典课程：
  - 国内外大部分学校都开设
  - 历史悠久
  - 有经典教材
  - 重视实践教学，尤其是国内外重点大学
  - 强调教学过程

编译技术

- 国外大学：名称不一样：
  - **Compiler Design**： Carnegie Mellon
  - **Programming Languages and Compilers**: University of Illinois–Urbana-Champaign, University of California–Berkeley,
  - **Compilers and Interpreters** :Yale
  - **Compiler Design and Implementation** :Harvard
  - **Compiling Techniques**: Princeton
  - **Compilers**: Stanford
  - Compilers and Translation Systems：Purdue
  - **在MIT这门课被称为“Computer language Engineering”**
- 国内：编译原理、编译技术
- 内容差异不大。

Purdure

- ECE 468 Introduction to Compilers and Translation Engineering
- ECE 573 Compilers and Translation Systems
- ECE 495S Introduction to Compilers and Translation Engineering and
- ECE 663 Advance Optimized Compilers
- EE 573 Compilers & Translator Writing Systems

在MIT这门课被称为“Computer language Engineering”

6.035: Computer Language Engineering — Fall 2018

Home
Overview
General Administrivia
Schedule
Reference Materials
CyberPortal 1.1

Links to notes, labs, etc. on future days are copies of materials from the previous year to give you an idea what the future will bring. We will update the notes as the course progresses.

We anticipate that we will be dynamically updating this schedule as appropriate during the course of the semester.

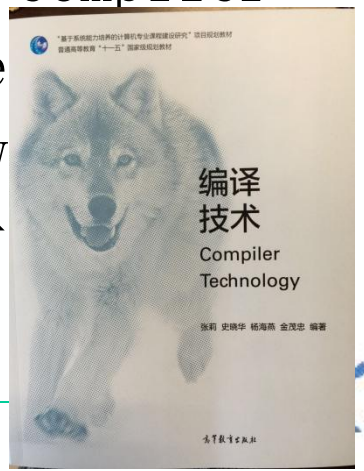
Monday	Tuesday	Wednesday	Thursday	Friday
sep 3	sep 4 Registration Day	sep 5 **L1:** Course Administration and Overview (old slides: S16, OCW) Miniquiz Handouts: Projects overview	sep 6 **L2:** Regular Expressions and Formal Grammars (old slides: S16, OCW) Miniquiz	sep 7 **L3:** Regular Expressions and Formal Grammars (old slides: S16, OCW) Miniquiz **Assignment**: P1, Scanner / Parser Project Scanner / Parser Project
sep 10 Project 1 Information Session **L4:** Bottom-up Parsing (old slides: S16, OCW) Miniquiz Handouts: Decaf language specification, course tools guide, Project 1 info session slides.	sep 11 **L5:** Bottom-up Parsing (old slides: S16, OCW) Miniquiz	sep 12 **L6:** Bottom-up Parsing (old slides: S16, OCW) Miniquiz	sep 13 **L7:** Top-down Parsing (old slides: S16, OCW) Miniquiz	sep 14
sep 17	sep 18	sep 19 DUE: Project 1	sep 20 Project 2 Information Session **L8:** Intermediate Representations (old slides: S16, OCW) Miniquiz Semantic Checker Project , Project 2 info session slides.	*sep 21* *Career Fair*
sep 24 **L9:** Intermediate Representations (old slides: S16, OCW) Miniquiz	sep 25 **L10:** Semantic Analysis (old slides: S16, OCW) Miniquiz DUE: Teams must be finalized	sep 26 **L11:** Semantic Analysis (old slides: S16, OCW) Miniquiz	sep 27 **L12:** Unoptimized Code Generation (old slides: S16, OCW) Miniquiz	sep 28 **L13:** Unoptimized Code Generation (old slides: S16, OCW) Miniquiz
oct 1 **L14:** Unoptimized Code Generation (old slides: S16, OCW) Miniquiz	oct 2	oct 3	oct 4 Office hours for project 2 Project 3 info session slides. **Assignment**: P3, Code Generator Project	oct 5 ADD DATE DUE: Project 2
oct 8 *Columbus Day*	*oct 9* *Columbus Day*	oct 10	oct 11	oct 12 QUIZ #1 2018 Fall (2018 Fall) **Practice Exams (Exam 1):** 2017 Fall (answers) 2016 Fall (answers), 2016 Spring (answers), 2014 (answers), 2013 (answers), 2011 (answers), 2010, on OCW.
oct 15 **L15:** Introduction to Program Analysis (old slides: S16, OCW) Miniquiz	oct 16 **L16:** Introduction to Program Analysis (old slides: S16, OCW) Miniquiz	oct 17 **L17:** Introduction to Data-flow Analysis (old slides: S16, OCW) Miniquiz	oct 18 **L18:** Introduction to Data-flow Analysis (old slides: S16, OCW) Miniquiz	oct 19 **L19:** Introduction to Data-flow Analysis (old slides: S16, OCW) Miniquiz
oct 22	oct 23	oct 24	oct 25	oct 26 Project 4 info session slides. **Assignment**: P4, Data-flow Analysis Project DUE: Project 3 *Graded*
oct 29 **L20:** Loop Optimizations (old slides: S16, OCW) Miniquiz	oct 30 **L21:** Loop Optimizations (old slides: S16, OCW) Miniquiz	oct 31 **L22:** Register Allocation (old slides: S16, OCW) Miniquiz	nov 1 **L23:** Register Allocation Wrap-Up (old slides: S16, OCW) Miniquiz	nov 2

教材

- 1.**龙书(Dragon book)**：书名是Compilers: Principles,Techniques,and Tools；作者是：Alfred V.Aho,Ravi Sethi,Jeffrey D.Ullman ；国内所有的编译原理教材基本都是参考了本书，重点是编译的前端技术。

- 2.**鲸书(Whale book)**：书名是：Advanced Compiler Design and Implementation；作者是：Steven S.Muchnick ；也就是高级编译原理。

- 3.**虎书(Tiger book)**：书名是：Modern Compiler Implementation in Java/C++/ML, Se Edition；作者是：Andrew W.Appel,w Palsberg。这本书是3本书中最薄的一



评分标准分析比较

- **CMU**：作业80%，期中考试 20%
- **UIUC:**
  - 8 道题目，20%
  - 2 次期中考试，各占25%
  - 1 次期末考试，占30%
- **Berkeley:**
  - 编程作业：7个编程作业，30%
  - 考试：两次平时考试和一次期末考试，分别占10%，10%，30%
  - 问题集（Problem sets）：提交“学习问题”，15%
  - 参加讨论：根据出勤、讨论课和协作情况给分，5%
- **Priceton:**
  - 不完成高级项目 (total = 100%)
    - » 标准编程作业 (1-10): 50%
    - » 期中和期末考试 50%
  - 完成高级项目 (total = 100%)
    - 高级项目: N% (0 < N < 50; N 与难度有关.)
    - 标准编程作业(1-5): (50-3N/4)%
    - 期中和期末考试: (50-N/4)%

在MIT这门课被称为“Computer language Engineering”

6.035: Computer Language Engineering — Fall 2018

Home
Overview
General Administrivia
Schedule
Reference Materials
CyberPortal 1.1

Links to notes, labs, etc. on future days are copies of materials from the previous year to give you an idea what the future will bring. We will update the notes as the course progresses.

We anticipate that we will be dynamically updating this schedule as appropriate during the course of the semester.

Monday	Tuesday	Wednesday	Thursday	Friday
sep 3	sep 4 Registration Day	sep 5 **L1:** Course Administration and Overview (old slides: S16, OCW) Miniquiz Handouts: Projects overview	sep 6 **L2:** Regular Expressions and Formal Grammars (old slides: S16, OCW) Miniquiz	sep 7 **L3:** Regular Expressions and Formal Grammars (old slides: S16, OCW) Miniquiz **Assignment**: P1, Scanner / Parser Project Scanner / Parser Project
sep 10 Project 1 Information Session **L4:** Bottom-up Parsing (old slides: S16, OCW) Miniquiz Handouts: Decaf language specification, course tools guide, Project 1 info session slides.	sep 11 **L5:** Bottom-up Parsing (old slides: S16, OCW) Miniquiz	sep 12 **L6:** Bottom-up Parsing (old slides: S16, OCW) Miniquiz	sep 13 **L7:** Top-down Parsing (old slides: S16, OCW) Miniquiz	sep 14
sep 17	sep 18	sep 19 DUE: Project 1	sep 20 Project 2 Information Session **L8:** Intermediate Representations (old slides: S16, OCW) Miniquiz Semantic Checker Project , Project 2 info session slides.	*sep 21* *Career Fair*
sep 24 **L9:** Intermediate Representations (old slides: S16, OCW) Miniquiz	sep 25 **L10:** Semantic Analysis (old slides: S16, OCW) Miniquiz DUE: Teams must be finalized	sep 26 **L11:** Semantic Analysis (old slides: S16, OCW) Miniquiz	sep 27 **L12:** Unoptimized Code Generation (old slides: S16, OCW) Miniquiz	sep 28 **L13:** Unoptimized Code Generation (old slides: S16, OCW) Miniquiz
oct 1 **L14:** Unoptimized Code Generation (old slides: S16, OCW) Miniquiz	oct 2	oct 3	oct 4 Office hours for project 2 Project 3 info session slides. **Assignment**: P3, Code Generator Project	oct 5 ADD DATE DUE: Project 2
oct 8 *Columbus Day*	*oct 9* *Columbus Day*	oct 10	oct 11	oct 12 QUIZ #1 2018 Fall (2018 Fall) **Practice Exams (Exam 1):** 2017 Fall (answers) 2016 Fall (answers), 2016 Spring (answers), 2014 (answers), 2013 (answers), 2011 (answers), 2010, on OCW.
oct 15 **L15:** Introduction to Program Analysis (old slides: S16, OCW) Miniquiz	oct 16 **L16:** Introduction to Program Analysis (old slides: S16, OCW) Miniquiz	oct 17 **L17:** Introduction to Data-flow Analysis (old slides: S16, OCW) Miniquiz	oct 18 **L18:** Introduction to Data-flow Analysis (old slides: S16, OCW) Miniquiz	oct 19 **L19:** Introduction to Data-flow Analysis (old slides: S16, OCW) Miniquiz
oct 22	oct 23	oct 24	oct 25	oct 26 Project 4 info session slides. **Assignment**: P4, Data-flow Analysis Project DUE: Project 3 *Graded*
oct 29 **L20:** Loop Optimizations (old slides: S16, OCW) Miniquiz	oct 30 **L21:** Loop Optimizations (old slides: S16, OCW) Miniquiz	oct 31 **L22:** Register Allocation (old slides: S16, OCW) Miniquiz	nov 1 **L23:** Register Allocation Wrap-Up (old slides: S16, OCW) Miniquiz	nov 2

课程定位的理解

- **系统性**：编译是一个完整的系统，也是学生接触到的第一个系统
- **过程完整性**：编译不仅讲解了编译技术，其实质是讲解了模型从一种语言表达形式到另一种语言表达形式的等价转化方法，应该保证过程的完整性
- **实践性**：理论和实践相结合。无论是研究性大学、非研究性大学都应该注重实践。不同的在于具体要求的不同。

培养目标

- 结合北航“与国家发展和民族振兴紧密相系”的发展观，坚持“把一流学生培养成一流人才”的育人理念，考虑本专业学生基础及专业对计算机系统能力培养的要求，编译技术课程培养目标为：
  - 掌握编译系统的功能、原理和构造方法（不局限于特定的程序设计语言和目标机）；理解高级程序设计语言的运行机理。
  - 具备完整小型编译系统的设计、实现和测试的能力；理解将一种程序设计语言转换/翻译为另一种的程序等价转化的理论和方法。
  - 大部分同学具备针对特定微处理器架构和指令集架构进行编译优化的能力；理解程序的安全问题。
  - 掌握形式语言以及自动机理论在编译器前端自动生成中的应用；理解不同方法和工具的作用和局限性。
  - 了解编译技术的前沿和新发展动向

课程要求

★ 课时：

理论课：48学时（1－13周）

- 周一上午： **1-2节（8:00-9:35）， 1-13周，F227**
- 周四上午： **3-4节（9:50~11:25）， 1-13周，F227**

实践课：48学时（2-17周，每周三小时）

- 周一下午 **8-10节** 机房 （杨老师和我）

★ 考试：两部分分别计分

- 理论基础（60%）:课堂教学，按时交作业。
  - 作业10分；
  - 3-6次随堂考试，共计30分；（不补）
  - 期末闭卷考试，60分
  - 主动回答问题，每次奖励0.5分，5分封顶(考前公布)
- 实践部分(40%)：上机实践(48 机时)
- 两部分如果不及格，单独补考。

要求：

提前预习，上课认真听讲；

课后及时复习，独立认真完成作业。

辅导老师：

课程助教：

王正达

刘桢炜

地点：新主楼G311，312

网址：学校在线课程平台

爱课程：http://www.icourses.cn

答疑时间：每周二、周五晚上交作业

周三晚上8:00－10:00 （暂定）

编译技术

1、理论和实践相结合、基础性和前沿性相结合；2、强调编译过程的完整性，因材施教，要求每一个同学独立完成一个不同难度的完整编译系统。通过对每个学生进行面试答辩，严把质量关。3、依托北航计算机软件与理论的学术优势，科研和教学紧密互动，研制了编译课程教学辅助软件，开发了特色教学网站。

开始学习 参与课堂互动

国家级

课程名称：编译技术
课程学时：48
所属学校：北京航空航天大学
学科门类：工学
负责人：张莉
专业类：计算机类
课程类型：理论课（含实践/实验）
专业：计算机科学与技术
课程属性：专业基础课/技术基础课
适用专业：计算机软件与理论、软件工程、计算机系统结构

收藏课程 站内分享 分享到： 0

课程介绍

课程指导思想及定位 “编译技术”是计算机软件学科的一门核心专业必修课程，是学生了解高级程序设计语言原理和编译相关技术的基础课，也是学生接触到的第一个完整的软件系统，在教学中具有重要地位。根据北航计算机学院的培养定位，我们要求学生既要掌握有关编译的经典基础理论，又要学会运用先进的软件开发技术构造实际编译系统的方法，因此这是一门理论和实践要求很高的课程，也是学生在本科学习阶段培养动手能力的一个非常重要...

查看更多


课程大纲


教学日历


考评方式与标准


学习指南

第一章　概论

（介绍名词术语、了解编译系统的结构和编译过程）

- 编译的起源：程序设计语言的发展
- 基本概念
- 编译过程和编译程序构造
- 编译技术的应用

1.1 程序设计语言的发展

机器语言（机器指令） → 汇编语言 → 面向用户的语言 → 面向问题的语言

C7 06 0000 0002　　MOV x, 2

x = 2

低级语言

高级语言

- 低级语言（Low level Language)
  - 字位码、机器语言、汇编语言
    - 特点：与特定的机器有关，功效高，但使用复杂、繁琐、费时、易出错
- 高级语言
  - -- Fortran、Pascal、C 、Java 语言等
    - 特点：不依赖具体机器，移植性好、对用户要求低、易使用、易维护等。

用高级语言编制的程序，计算机不能立即执行，必须通过一个“翻译程序”加工，转化为与其等价的机器语言程序，机器才能执行。

这种翻译程序，称之为“编译程序”。

1.2 基本概念

- **源程序**

  用汇编语言或高级语言编写的程序称为源程序。

- **目标程序**

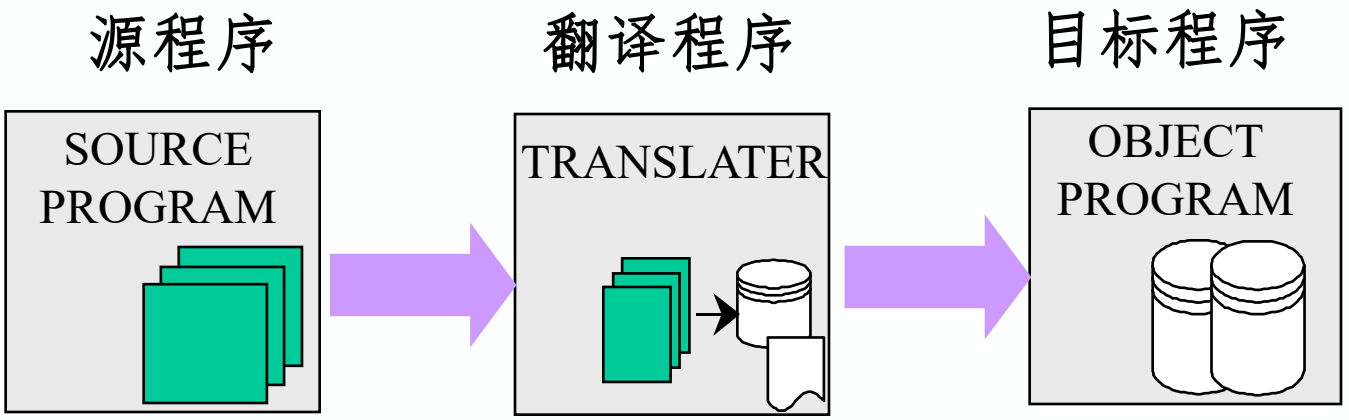
  用目标语言所表示的程序。

  目标语言：可以是介于源语言和机器语言之间的“中间语言”，可以是某种机器的机器语言，也可以是某机器的汇编语言。

- **翻译程序**

  将源程序转换为目标程序的程序称为翻译程序。它是指各种语言的翻译器，包括汇编程序和编译程序，是汇编程序、编译程序以及各种变换程序的总称。

源程序、翻译程序、目标程序 三者关系：



即源程序是翻译程序的输入，目标程序是翻译程序的输出

源程序	翻译程序	目标程序
汇编语言	汇编程序	机器语言
高级语言	编译程序	目标程序

- **汇编程序**

若源程序用汇编语言书写，经过翻译程序得到用机器语言表示的程序，这时的翻译程序就称之为汇编程序，这种翻译过程称为“汇编”（Assemble）

- **编译程序**

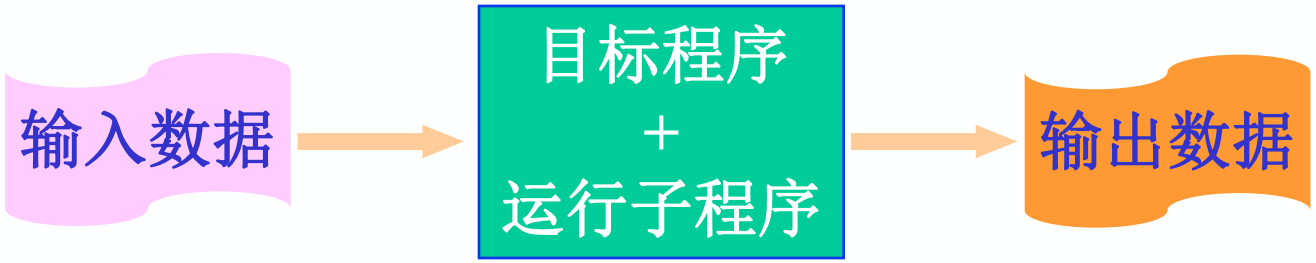
若源程序是用高级语言书写，经加工后得到目标程序，这种翻译过程称“编译”（Compile)

汇编程序与**编译程序**都是**翻译程序**，主要区别是加工对象的不同。由于汇编语言格式简单，常与机器语言之间有一一对应的关系，汇编程序所要做的翻译工作比编译程序简单得多。

源程序的编译和运行

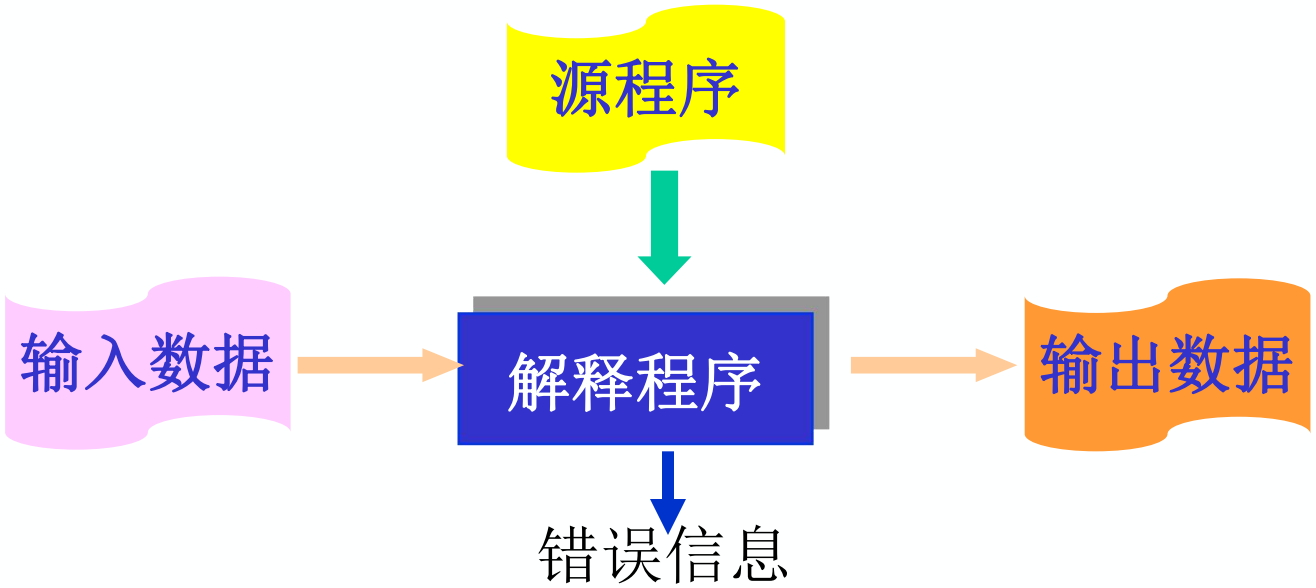
- 编译或汇编阶段

- 运行阶段



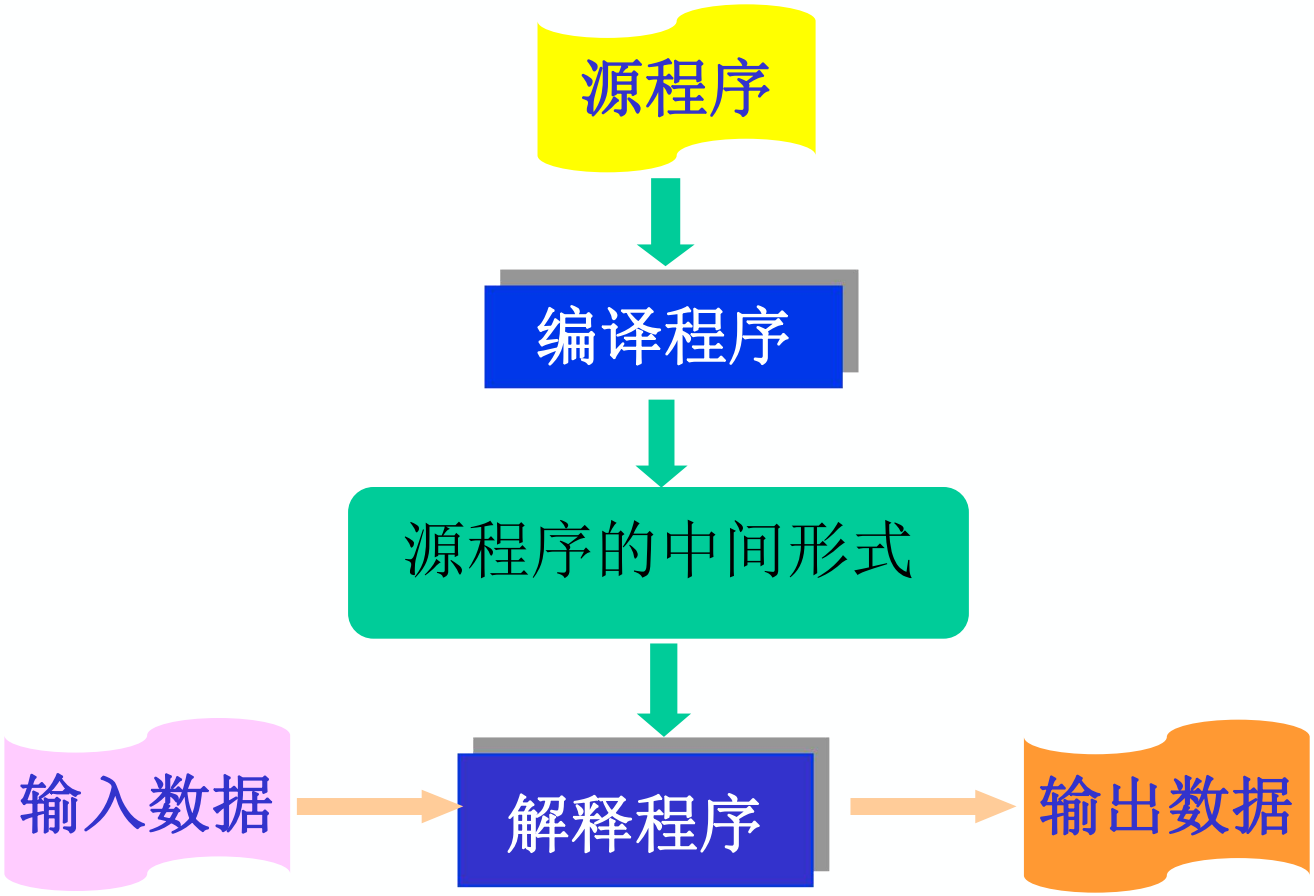
- 解释程序（Interpreter）
  对源程序进行解释执行的程序。

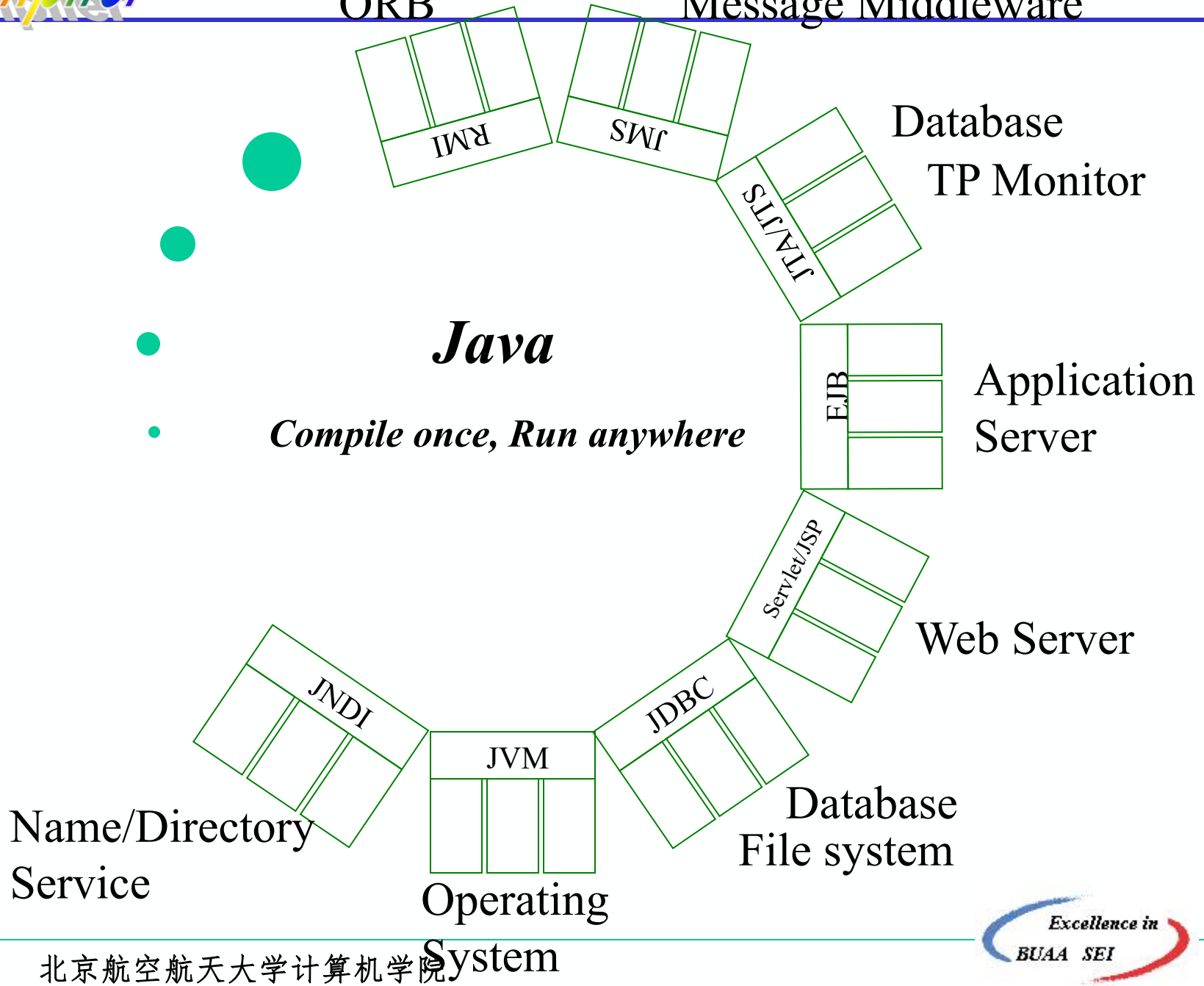
- 工作过程



- 特点、与编译程序比较

“编译-解释执行”系统




ORB
Message Middleware
RMI
JMS
Database
TP Monitor
JTA/JTS
Java
Compile once, Run anywhere
EJB
Application
Server
Servlet/JSP
Web Server
JNDI
JVM
JDBC
Database
File system
Name/Directory
Service
Operating
System

1.3.1 编译过程

编译过程是指将高级语言程序翻译为等价的目标程序的过程。

习惯上是将编译过程划分为5个基本阶段：

一、词法分析

任务：分析和识别单词。

源程序是由字符序列构成的，词法分析扫描源程序（字符串），根据语言的词法规则分析并识别单词，并以某种编码形式输出。

- **单词：是语言的基本语法单位，一般语言有四大类单词**

对于如下的字符串，词法分析程序将分析和识别出9个单词：

X1	:=	(	2.0	+	0.8	)	*	C1
1	2	3	4	5	6	7	8	9

→ 也称为线性分析。

二、语法分析

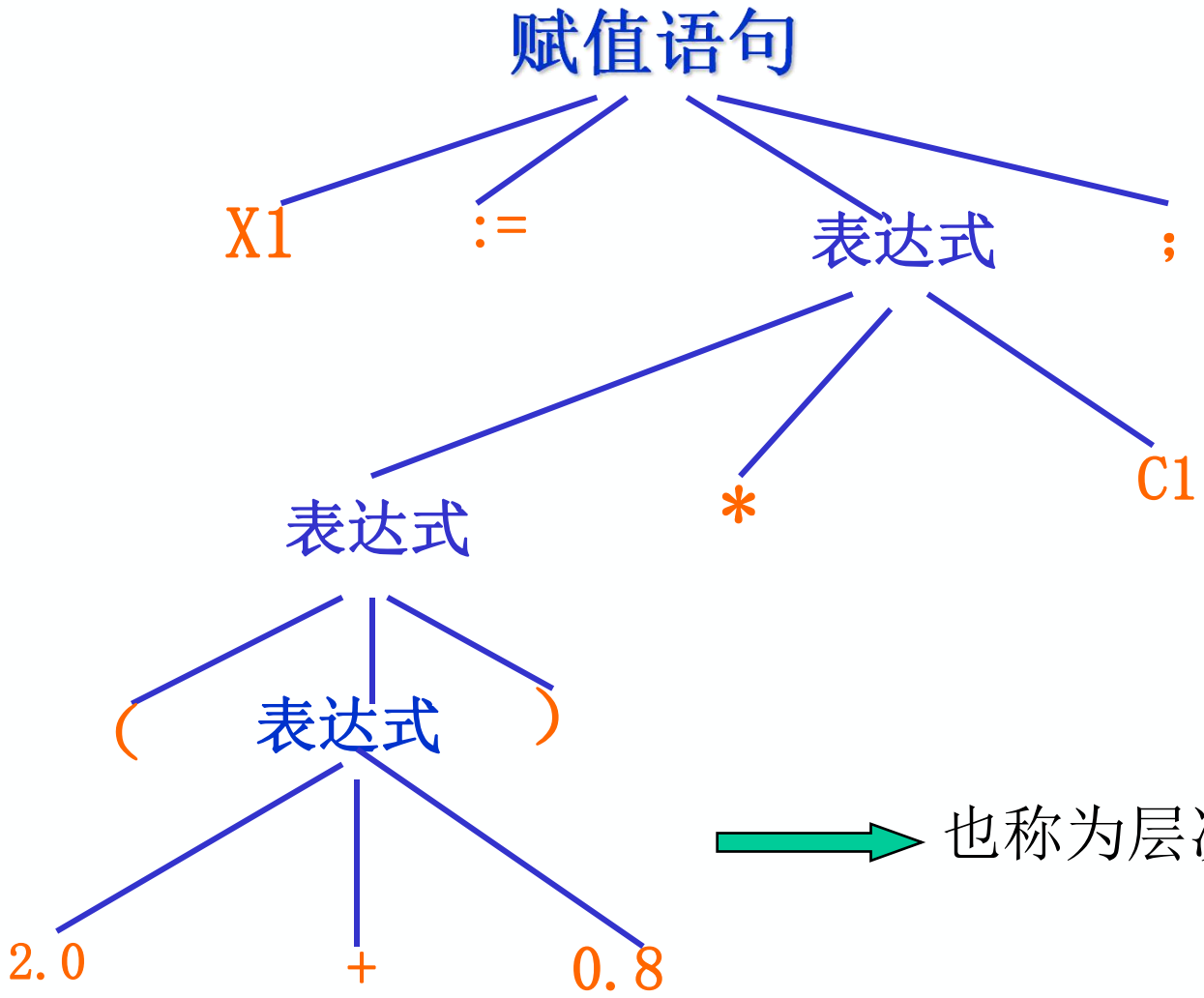
任务：根据语法规则（即语言的文法），分析并识别出各种语法成分，如表达式、各种说明、各种语句、过程、函数等，并进行语法正确性检查。

X1:= (2.0 + 0.8) * C1

赋值语句的文法：

<赋值语句>→<变量><赋值操作符><表达式>
<变量>→<简单标识符>
<赋值操作符>→:=
<表达式>→ .....

X1:= (2.0 + 0.8) * C1;



也称为层次分析。

三、语义分析、生成中间代码

任务：对识别出的各种语法成分进行语义分析，
并产生相应的中间代码。

- 中间代码：一种介于源语言和目标语言之间的中间语言形式
- 生成中间代码的目的：
  〈1〉便于做优化处理；
  〈2〉便于编译程序的移植。
- 中间代码的形式：编译程序设计者可以自己设计，常用的有四元式、三元式、逆波兰表示等。

例： X1:= (2.0 + 0.8) * C1

- 由**语法分析**识别出为赋值语句，语义分析首先要分析语义上的正确性，例如要检查表达式中和赋值号两边的类型是否一致。

- 根据赋值语句的语义，生成中间代码。即用一种语言形式来代替另一种语言形式，这是翻译的关键步骤。（**翻译的实质：**语义的等价性）

★ 四元式（三地址指令）

X1:= (2.0 + 0.8) * C1

	运算符	左运算对象	右运算对象	结果
(1)	+	2.0	0.8	T1
(2)	*	T1	C1	T2
(3)	:=	X1	T2	

其中T1和T2为编译程序引入的工作单元

四元式的语义为：
$$2.0 + 0.8 \rightarrow T1$$
$$T1 * C1 \rightarrow T2$$
$$T2 \rightarrow X1$$

这样所生成的四元式与原来的赋值语句在语言的形式上不同，但语义上等价。

四、代码优化

任务：目的是为了得到高质量的目标程序。

例如：前面的四元式中第一个四元式是计算常量表达式值，该值在编译时就可以算出并存放在工作单元中，不必生成目标指令来计算，这样四元式可优化为：

编译时：　　2.0 + 0.8 → T1

（1）* T1 C1 T2

（2）:= X1 T2

优化前的四元式：

(1)	+	2.0	0.8	T1
(2)	*	T1	C1	T2
(3)	:=	X1	T2	

五、生成目标程序

由中间代码很容易生成目标程序（地址指令序列）。这部分工作与机器关系密切，所以要根据机器进行。在做这部分工作时（要注意充分利用累加器），也可以进行优化处理。 X1:=（2.0 + 0.8）* C1

```
LOAD   2.0
ADD    0.8
STO    T1
LOAD   T1
MUL    C1
STO    T2
LOAD   T2
STO    X1
```

作利用累加器的优化

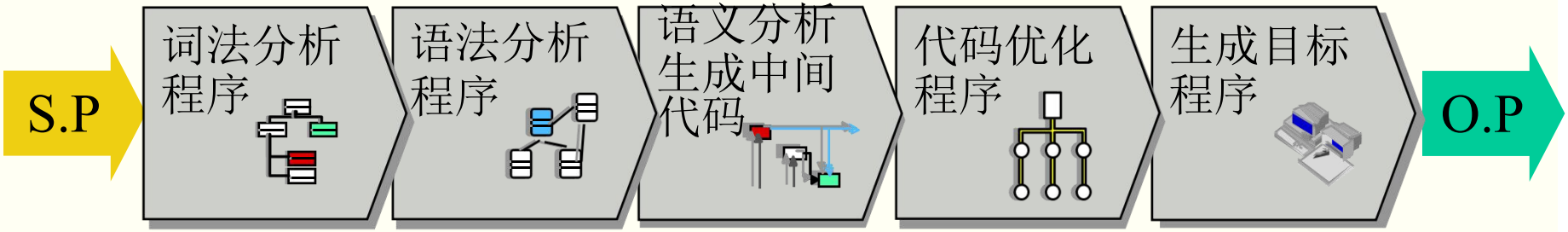
```
LOAD   2.0
ADD    0.8
MUL    C1
STO    X1
```

注意：在翻译成目标程序的过程中，要切记保持语义的等价性。

1.3.2 编译程序构造

一、 编译程序的逻辑结构

按逻辑功能不同，可将编译过程划分为五个基本阶段，与此相对应，我们将实现整个编译过程的编译程序划分为五个逻辑阶段（即五个逻辑子过程）。



在上列五个阶段中都要做两件事：

（1）建表和查表；（2）出错处理；

所以编译程序中都要包括符号表管理和出错处理两部分

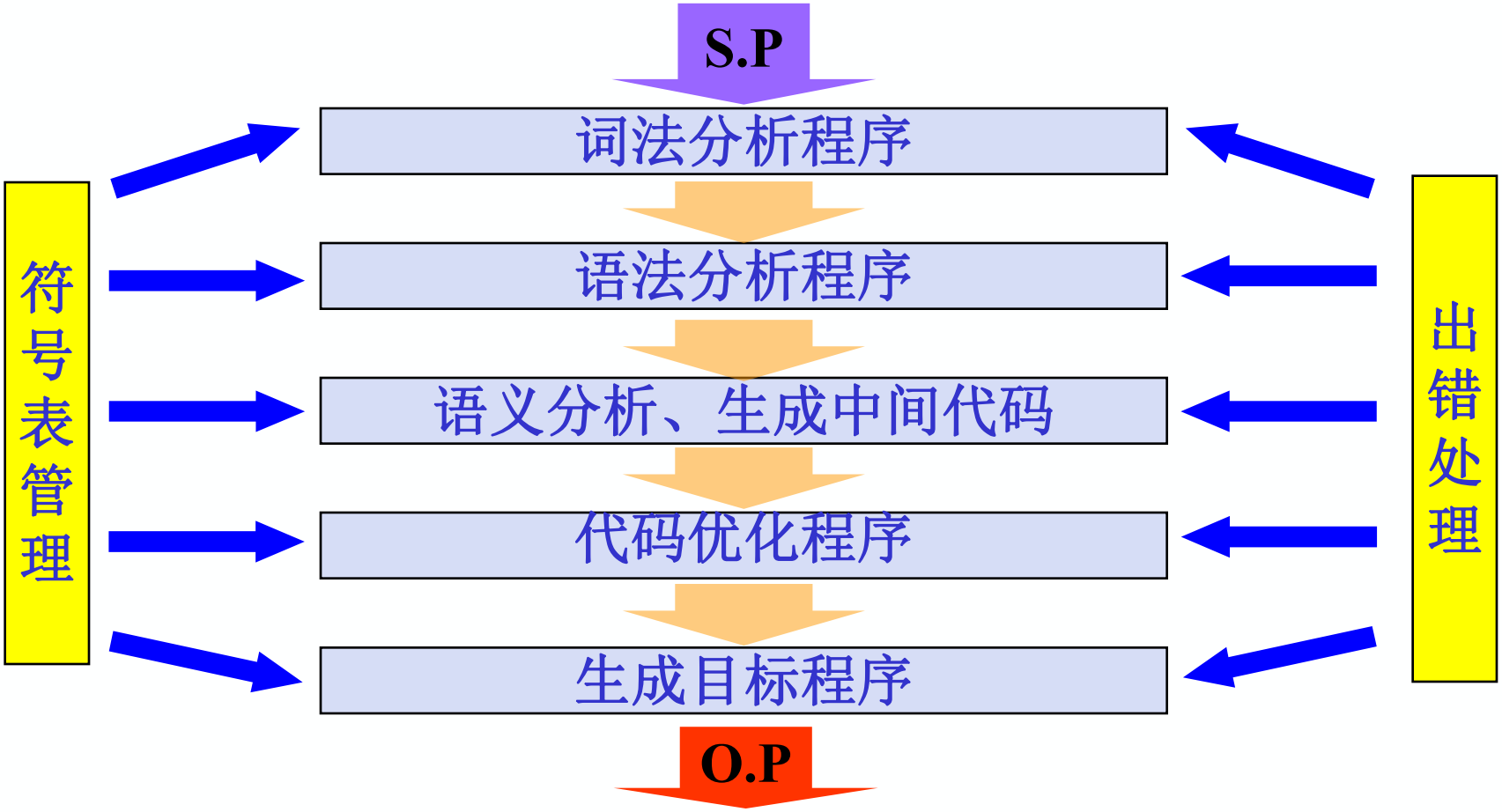
★ 符号表管理

在整个编译过程中始终都要贯穿着建表（填表）和查表的工作。即要及时地把源程序中的信息和编译过程中所产生的信息登记在表格中，而在随后的编译过程中同时又要不断地查找这些表格中的信息。

★ 出错处理

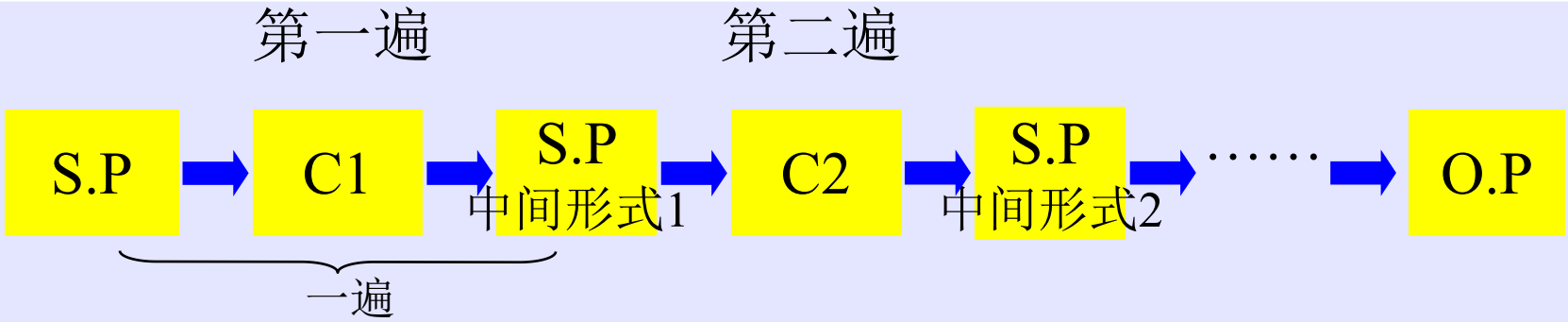
规模较大的源程序难免有多种错误，编译程序必须要有出错处理的功能。即能诊察出错误，并能报告用户错误的性质和位置，以便用户修改源程序。出错处理能力的大小是衡量编译程序质量好坏的一个重要指标。

典型的编译程序具有7个逻辑部分



二、 遍（PASS）

遍：对源程序（包括源程序中间形式）从头到尾扫描一次，并做有关的加工处理 ， 生成新的源程序中间形式或目标程序，通常称之为一遍。



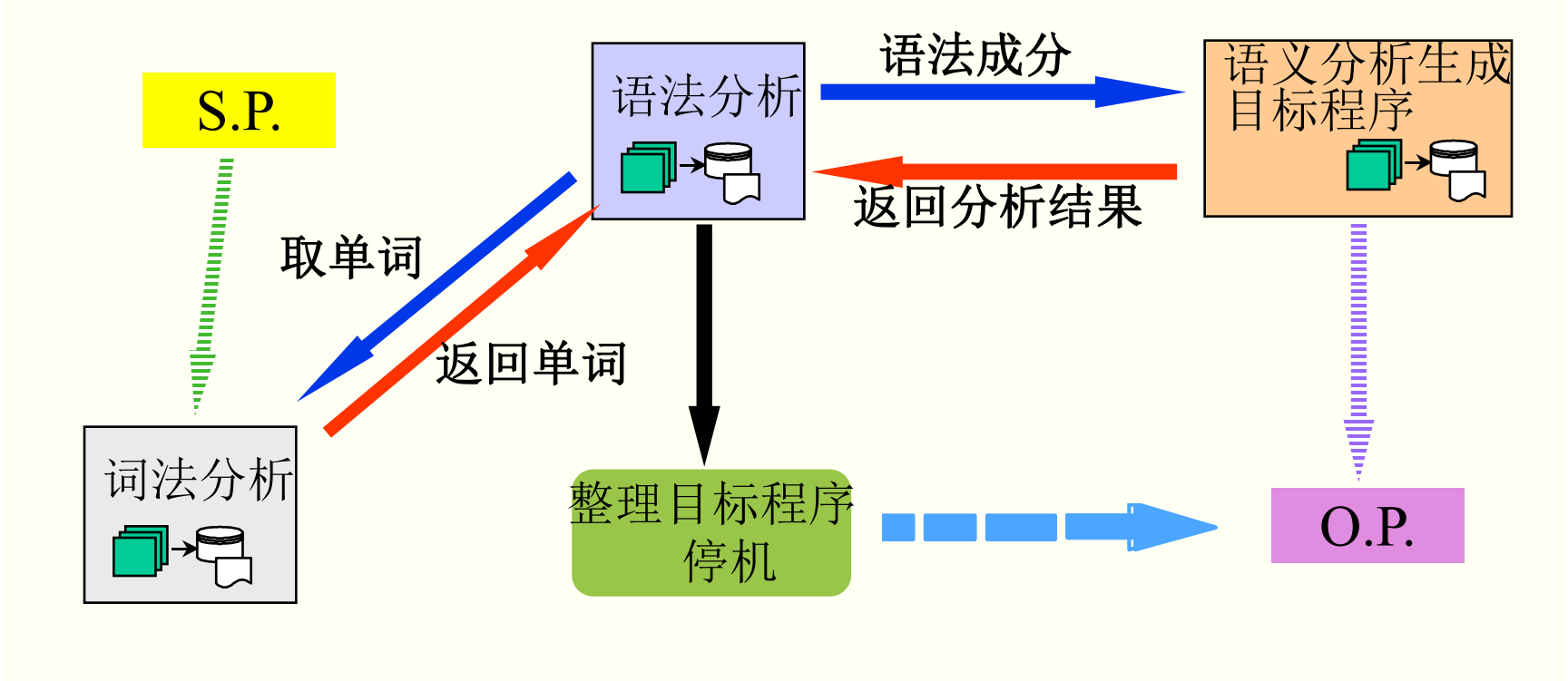
★要注意遍与基本阶段的区别

五个基本阶段：是将源程序翻译为目标程序在逻辑上要完成的工作。

遍：是指完成上述5个基本阶段的工作，要经过几次扫描处理。

一遍扫描即可完成整个编译工作的称为一遍扫描编译程序

其结构为：



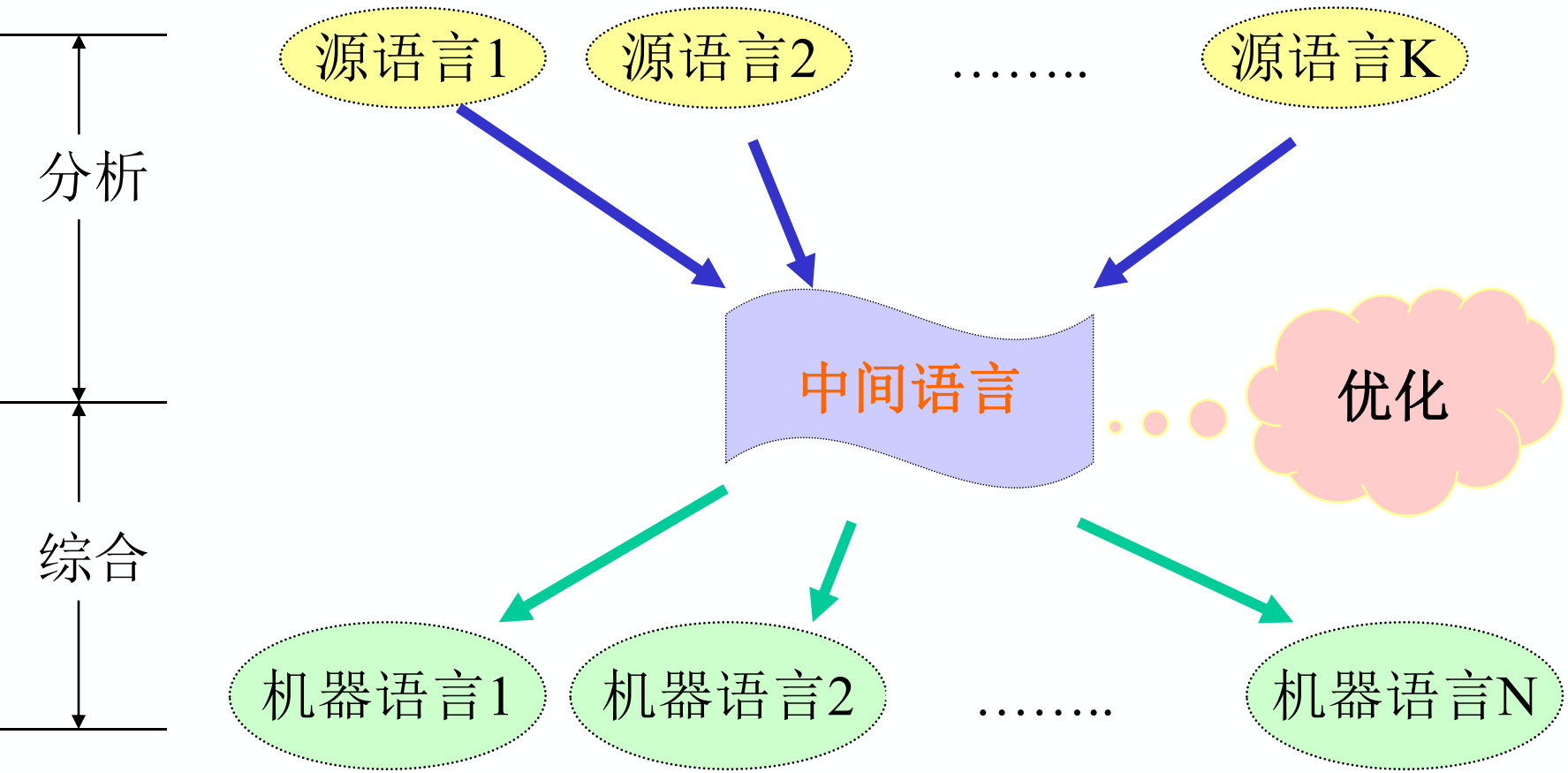
三、前端和后端

根据编译程序各部分功能，将编译程序分成前端和后端。

前端：通常将与源程序有关的编译部分称为前端。
词法分析、语法分析、语义分析、中间代码生成、代码优化 --------分析部分
特点：与源语言有关

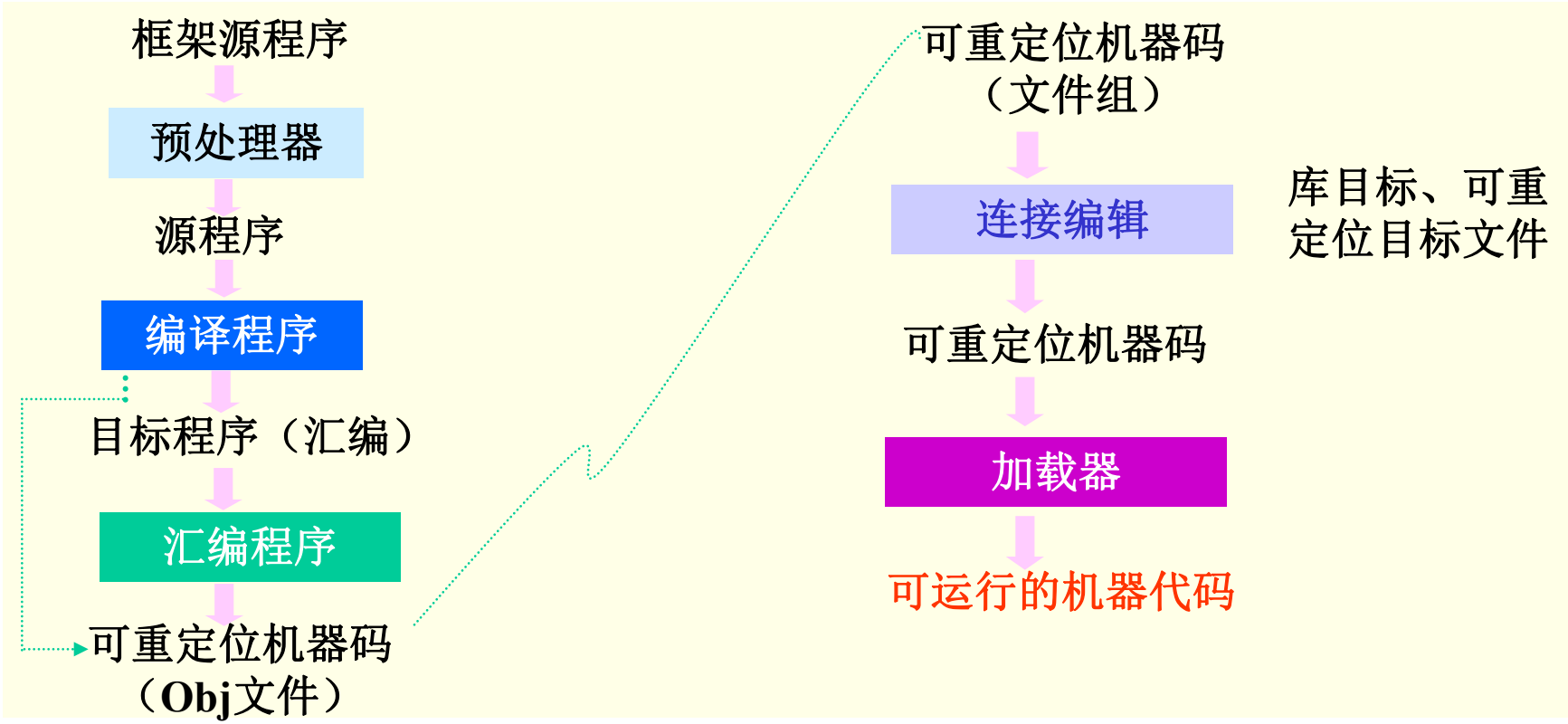
后端：与目标机有关的部分称为后端。
目标程序生成（与目标机有关的优化）
--------综合部分
特点：与目标机有关

中间表示的重要性



四、 编译程序的前后处理器

源程序：多文件、宏定义和宏调用，包含文件
目标程序：一般为汇编程序或可重定位的机器代码



1.4 编译技术的应用

- 语法制导的结构化编辑器
- 程序格式化工具
- 软件测试工具
- 程序理解工具
- 高级语言的翻译工具
- 等等。

作业： P21： 1,2,3,4,5

- 教学安排

过程完整：至少翻译过程完整

习惯上是将编译过程划分为5个基本阶段：

词法分析

语法分析

语义分析、生成中间代码

代码优化

生成目标程序

词法分析程序：状态图，手工编写程序

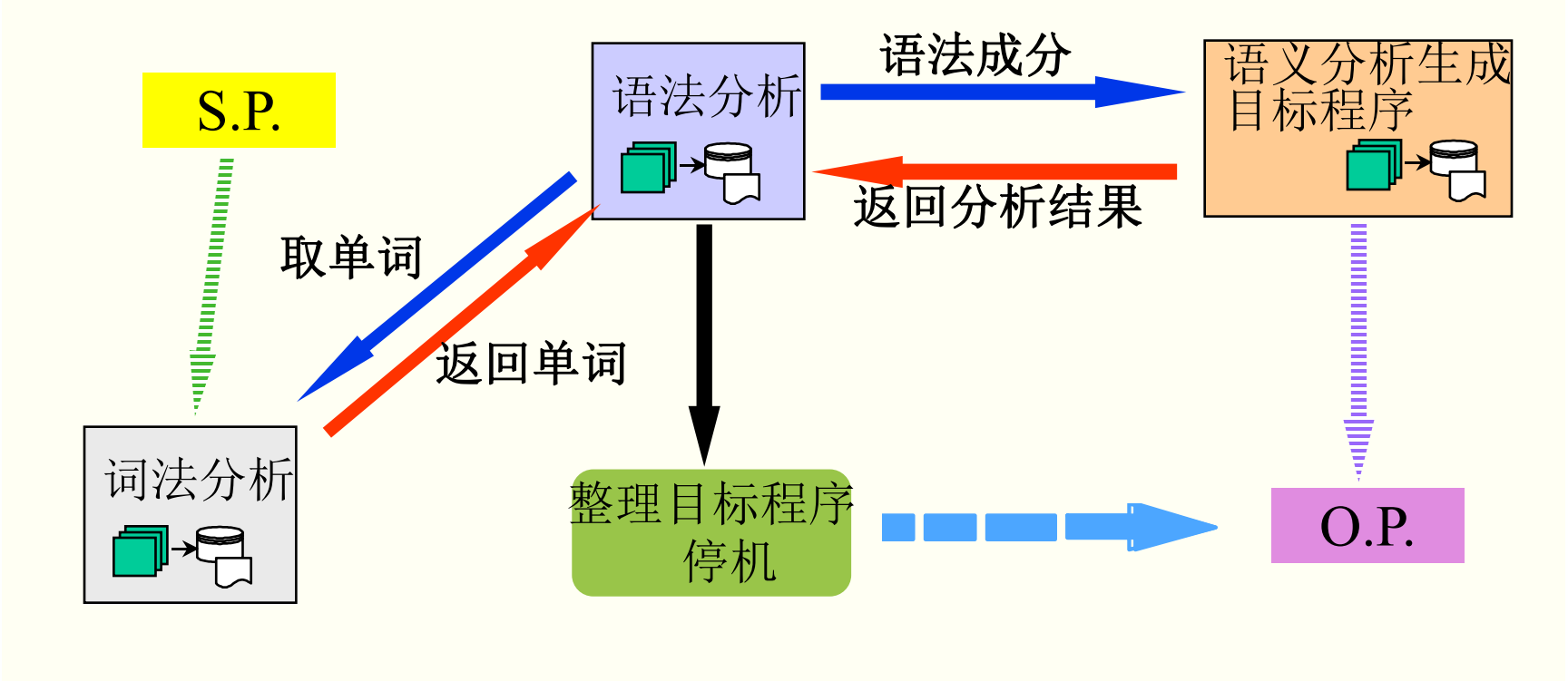
语法分析程序：递归子程序法，编写程序

语义分析和代码生成：语法制导的翻译（属性翻译文法）

完成了一个翻译过程

一遍扫描即可完成整个编译工作的称为一遍扫描编译程序

其结构为：



过程完整基础上：提高

习惯上是将编译过程划分为5个基本阶段：

词法分析

语法分析

语义分析、生成中间代码

代码优化

生成目标程序

词法分析程序：状态图，手工编写程序

词法分析程序的自动生成：有穷自动机，LEX

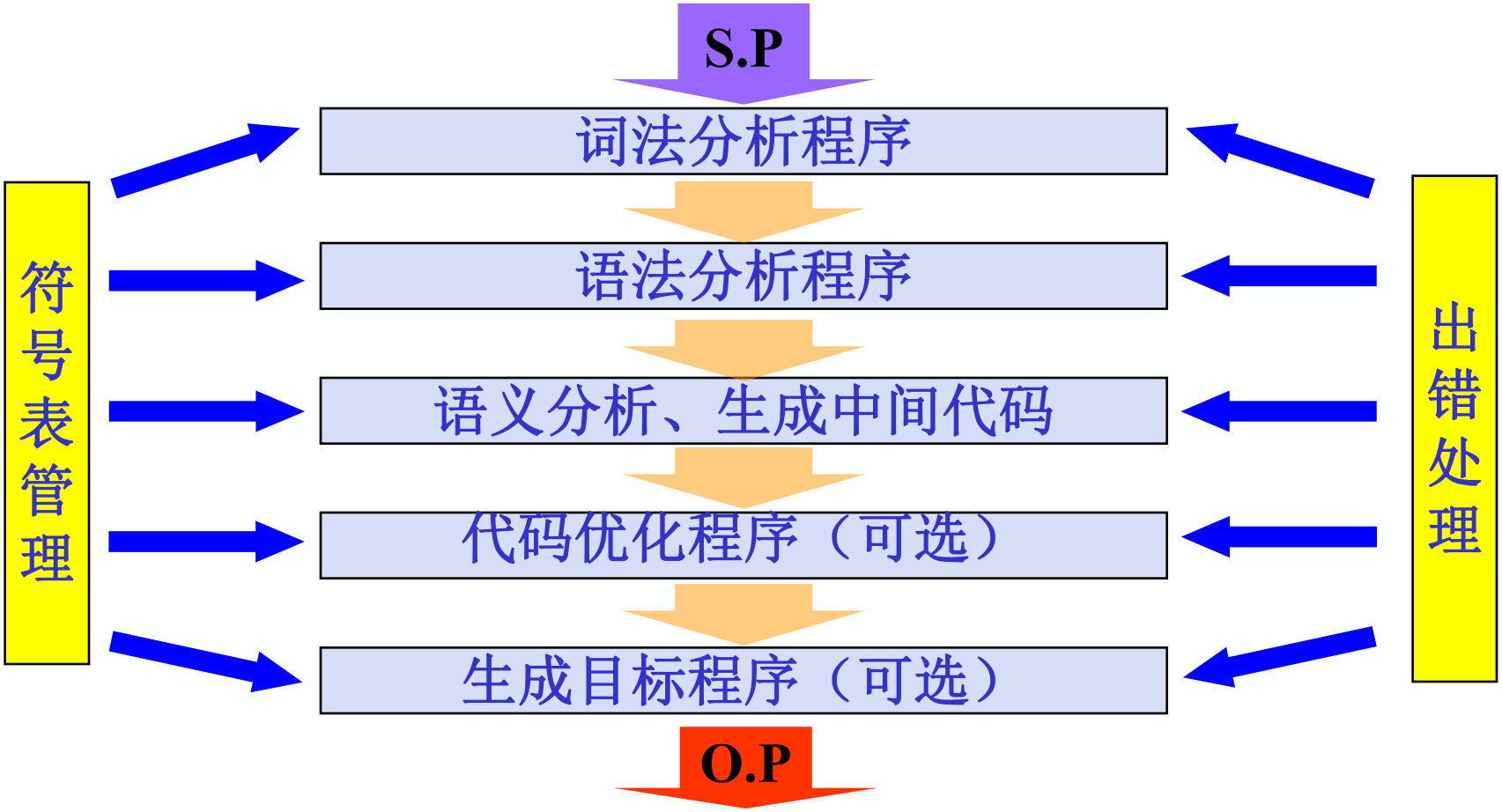
语法分析程序：递归子程序法，编写程序

自顶向下分析法：

自底向上分析法

语义分析和代码生成：语法制导的翻译（属性翻译文法）

系统完整


S.P
词法分析程序
语法分析程序
语义分析、生成中间代码
代码优化程序（可选）
生成目标程序（可选）
O.P
符号表管理
出错处理

教学安排

词法分析程序：状态图，手工编写程序

编写词法分析程序

语法分析程序：递归子程序法，编写程序

编写语法分析程序

语义分析和代码生成：语法制导的翻译（属性翻译文法）

完成了一个翻译过程

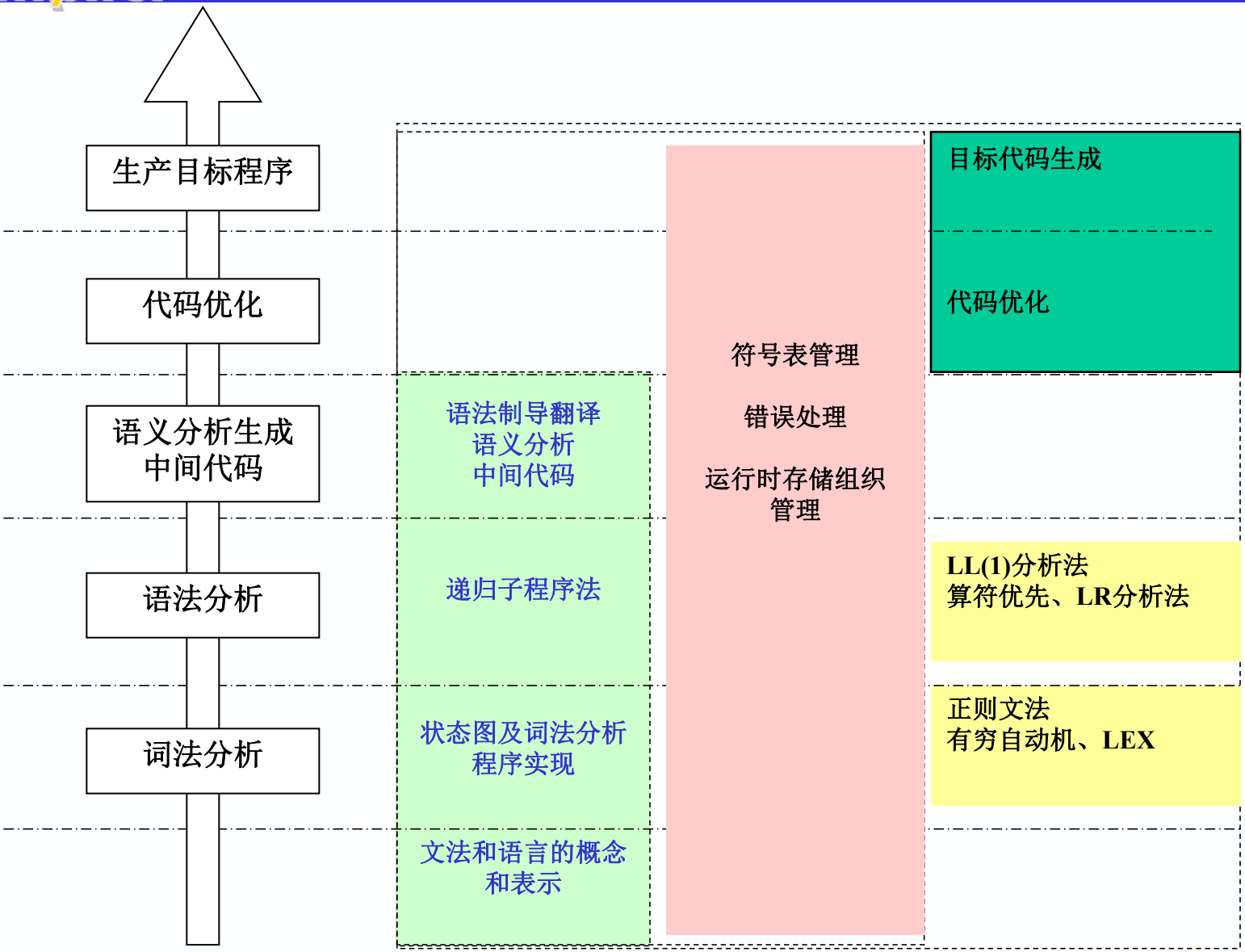


图1 编译过程和知识点组织

回顾：

编译的起源

概念：源程序 目标程序 翻译程序

汇编程序 编译程序

编译过程：5个基本阶段

编译程序：7个逻辑部分

遍 前端 后端 前后处理器

- 教育：被教育者在这个过程中获得自信。
- 如何获得自信：参与教育、证明自己很棒---自信！！
- “痛并快乐着!!!”