编译原理

软件学院　2016

课程概览

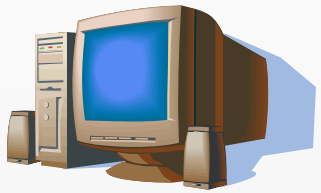
- 第1章 编译引论
- 第2章 词法分析
- 第3章 形式语言*与语法分析
- 第4章 自顶向下语法分析
- 第5章 自底向上语法分析
- 第6章 语义分析
- 第7章 中间代码生成
- 第8章 中间代码优化*
- 第9章 运行时存储空间管理
- 第10章 目标代码生成*

第1章 编译引论

- 1.1 什么是编译程序
  - 1.1.1 程序设计语言
  - 1.1.2 程序设计语言的实现方式
- 1.2 编译程序的构成
- 1.3 开发编译程序的途径
- 1.4 编译程序的伙伴程序

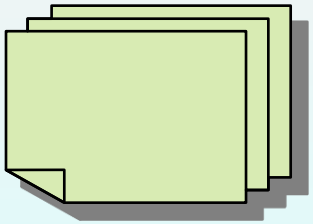
1.1 什么是编译程序

```
#include “stdio.h”
void main()
{
    printf(“%s\n”, “hello!”);
}
```



高级语言程序

机器语言程序



翻译程序

1.1.1 程序设计语言

- 历史：机器语言---汇编语言---高级语言
- 机器指令：

  1011 1000 0011 0100 0001 0010

  (B83412)
- 汇编指令：MOV AX，1234H
- 高级语言：x = 10;

1.1.1 程序设计语言

- 程序设计语言的分类
- 用途分：科学计算(Fortran)、文字排版(Latex)、绘图(R)、商业数据处理(COBOL)、符号计算(Lisp)、脚本语言(Jscript)、通用语言(C、Java、Python)等
- 抽象程度分：低级(机器语言、汇编语言)、高级(C、ML、Ada、Prolog等)
- 编程机制分：命令式(C、Pascal)、函数式(Haskell)、逻辑式(Prolog)、对象式(C++、Java、C#等)

过程式 vs. 函数式

```
#define TYPE int
void square(TYPE x[], int n)
{
 for(int i=0; i<n; i++)
    *(x+i)*=*(x+i);
}
```

```
fun    square([])=[]
   | square(a::x)=a*a ::
square(x)
```

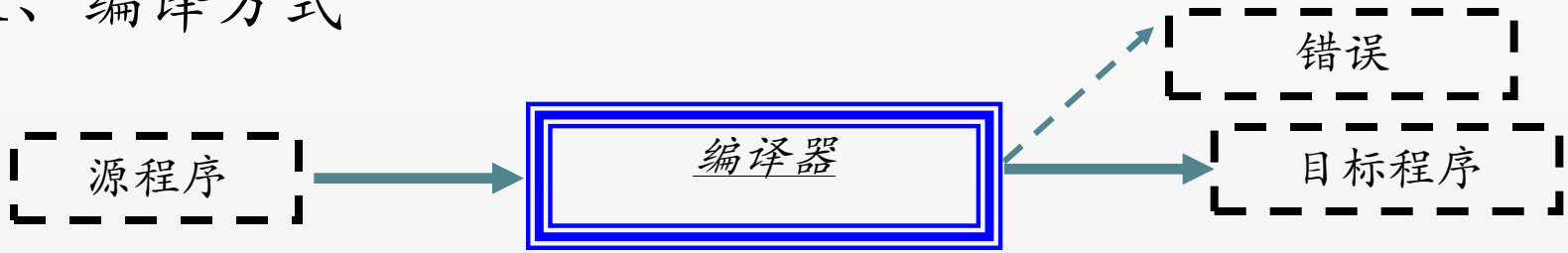
逻辑式

```
domains
   person, another = symbol
predicates
   likes(person, another).
clauses
   likes(jack, sussan).
   likes(john, marry).
   likes(tom,  cathy).
   likes(mark, ellen).
   likes(bob,  tom).
   likes(richard, ellen).
   likes(tom,  ellen).
   likes(jack, X) if likes(tom, X).
```

likes(jack, ellen)?

1.1.2 程序设计语言的实现方式

- 三种：编译方式、解释方式、转换方式
- 1、编译方式



- 编译器(Compiler):是将某种语言(源语言)编写的程序翻译成语义等价的另一种语言(目标语言)编写的程序.
- 特点:(1)目标程序若是机器语言程序，则可直接执行;目标程序若是汇编语言程序，则须经汇编器汇编后才可执行.(2)源程序中若有错误(词法、语法、语义错误)，编译器会报错，错误修正后需重新编译.报错及错误处理是编译程序的重要任务之一.

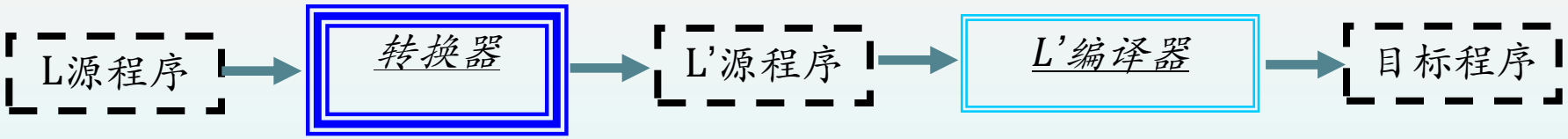
1.1.2 程序设计语言的实现方式

- 三种：编译方式、解释方式、转换方式
- 2、解释方式

- 解释器(Interpreter)：是对某种语言的程序一边翻译、一边执行的程序.
- 特点:(1)逐条指令翻译;(2)解释过程中发现错误会立即停止翻译，返回修正后要从头开始翻译;(3)翻译结束，即可得到源程序的执行结果.

1.1.2 程序设计语言的实现方式

- 三种：编译方式、解释方式、转换方式
- 3、转换方式
- 编译器的开发代价是非常昂贵的，在可能的情况下，可以将一种语言的程序转换成另一种语言的程序，利用另一种语言的编译器进行编译



- 适用：L和L'语言之间比较近似，或者L'是L的扩展，例如C++的编译

1.1.2 程序设计语言的实现方式

- 编译方式与解释方式的比较
- 相同点：使用同样的实现技术
- 不同点：

  (1) 实现机制:翻译(程序 to 程序) vs. 解释(指令 to 指令序列);

  (2) 翻译的效率：编译的效率高，解释的效率低;

  (3) 存储代价：编译方式占用存储少，解释方式占用存储多;

  (4) 错误定位：编译方式定位准确率低，解释方式定位准确率高;
- 结论：解释方式适合交互性较强语言的翻译;

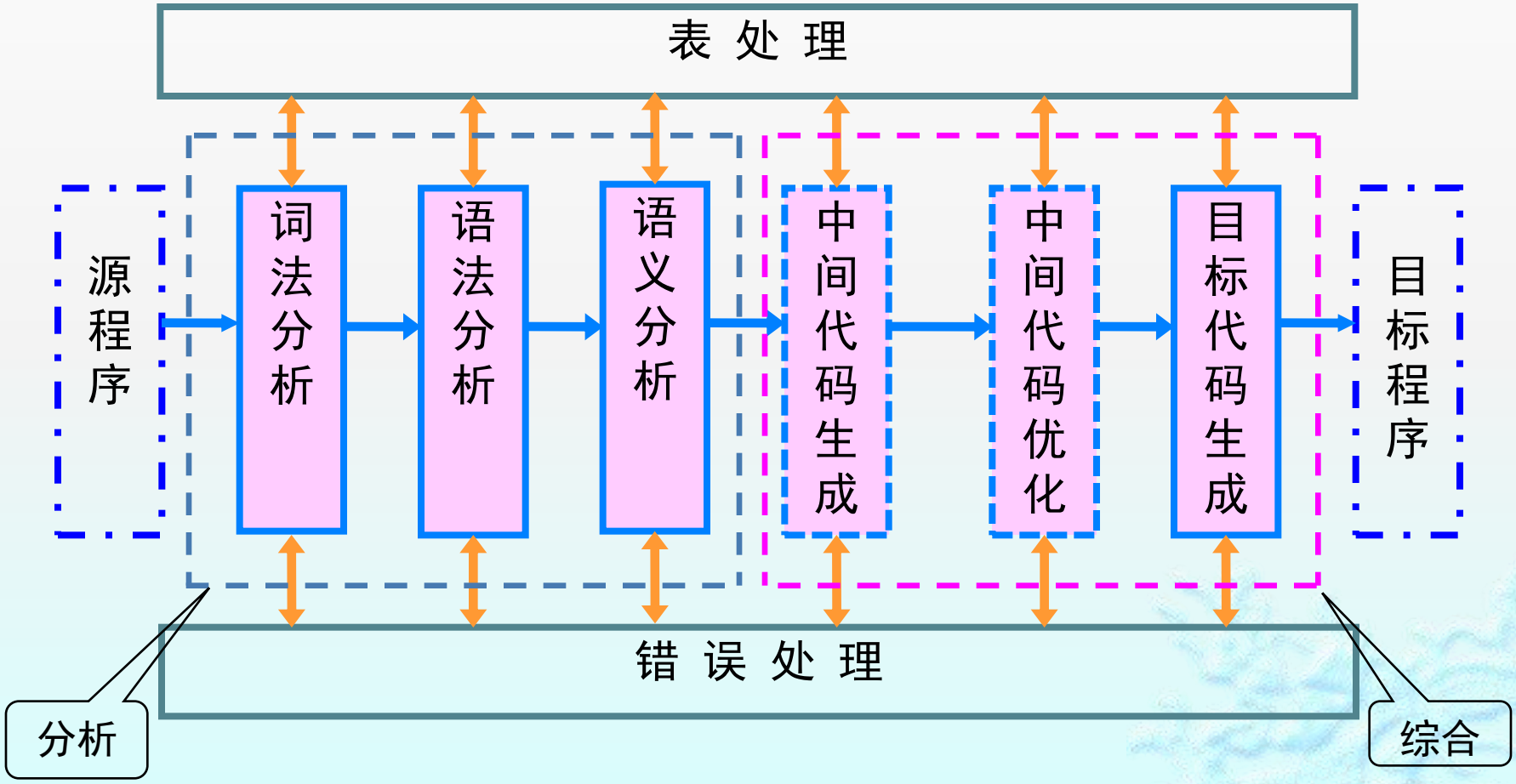
  编译方式适合对目标程序运行速度有要求的情形

1.2 编译程序的构成

- 自然语言的翻译过程：
- 中文：你能够通过自己的努力实现你的梦想！
- 英文：You can put your dreams into reality through your efforts!
- 翻译过程：
  - 识别单词、检查语法、检查语义、翻译(非逐字直译！)

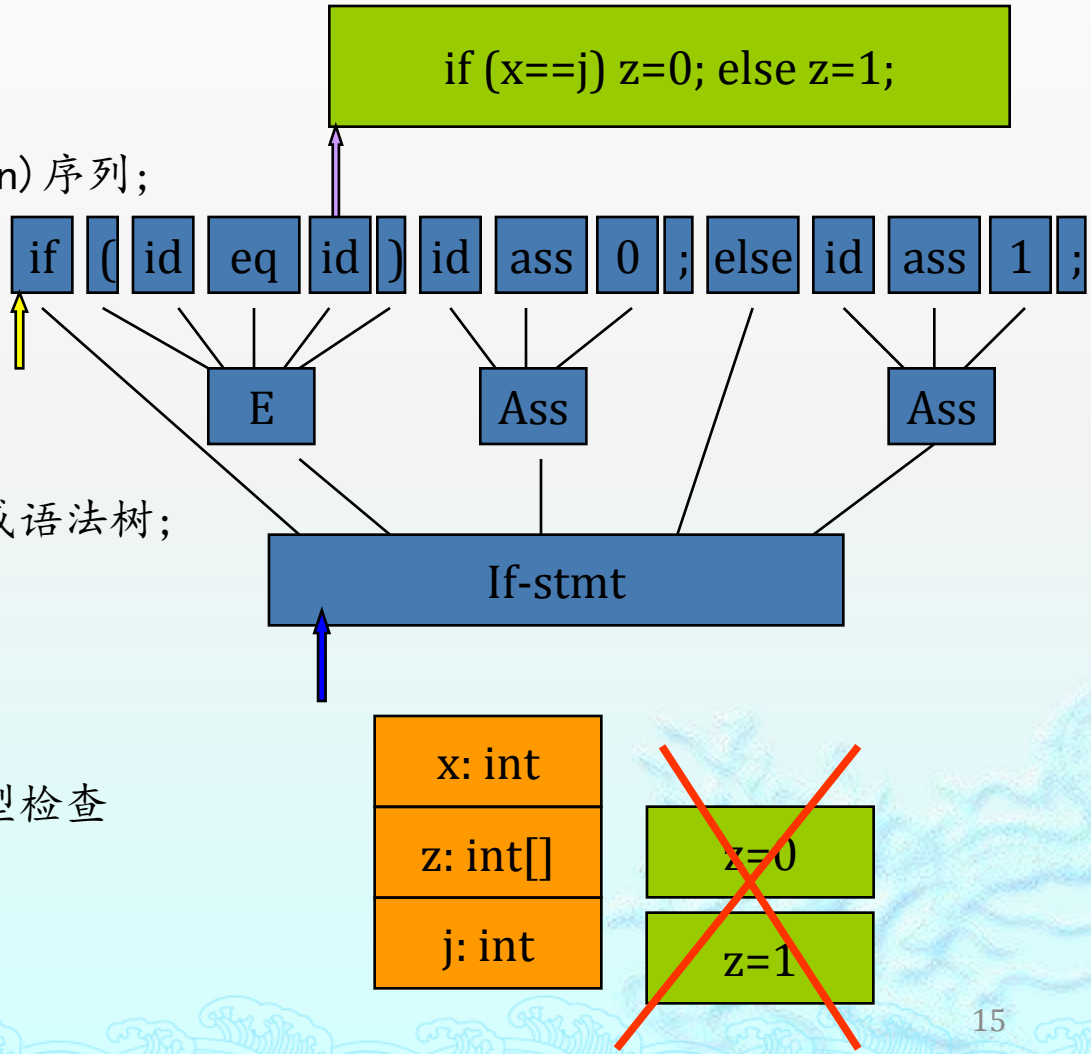
1.2 编译程序的构成

- 程序设计语言程序的翻译也类似：



1.2 编译程序的构成

- 词法分析
  - 扫描源程序的字符流；
  - 整理成有意义的单词(token)序列；
  - 识别并报告词法错误
- 语法分析
  - 扫描token序列；
  - 分析程序的语法结构；
  - 将分析结果表示成分析树或语法树；
  - 识别并报告语法错误
- 语义分析
  - 建立符号表
  - 进行静态语义检查，如类型检查
  - 识别并报告语义错误



1.2 编译程序的构成

- 代码生成
  - 中间代码生成：为优化和移植考虑，不是必要的
  - 目标代码生成：通常生成汇编代码
- 代码优化
  - 目的是提高目标程序的执行效率
    - 中间代码优化
    - 目标代码优化
- 表处理
  - 符号表及优化时建立的各种表
- 错误处理
  - 词法错误、语法错误、语义错误的处理

if (x==j) z=0; else z=1;

(==,x,j,t1)
(JMP0,t1,-,elseL)
(=,0,-,z)
(JMP,-,-,outL)
(Label, -,-,elseL)
(=,1,-,z)
(Label,-,-,outL)

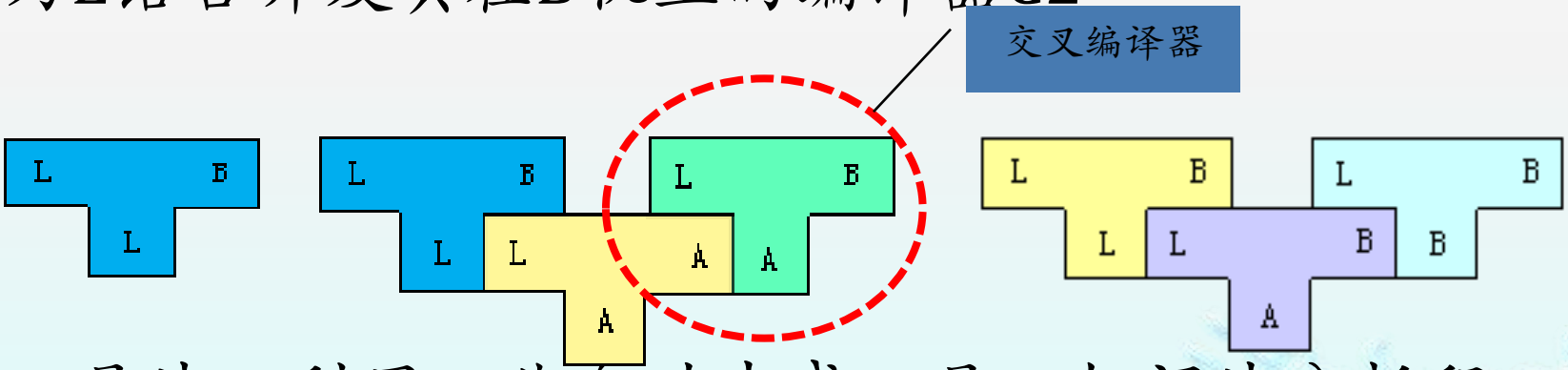
x=y*0 ⟺ x=0

1.2 编译程序的构成

- 几个术语
- 遍/趟(pass)：所谓“遍”就是对源程序或源程序的中间表示形式从头到尾扫描一次，并作加工处理，生成新的中间表示或目标程序。
- 前端：编译程序中与源语言有关，与目标程序无关的部分，称为前端。
- 后端：编译程序中与源语言无关，与目标程序有关的部分，称为后端。

1.3 开发编译程序的途径

- 手工编写：没有任何编译器时
- 自展法：先实现一个语言子集的编译，再利用该子语言的编译器逐步实现完整语言的编译
- 移植法：已有L语言及其在A机上的编译器C1，要为L语言开发其在B机上的编译器C2



- 工具法：利用一些自动生成工具，如词法分析程序自动生成工具FLEX，语法分析程序自动生成工具Yacc, Bison等

1.4 编译程序的伙伴程序

- 语言的集成开发环境(Integrated Development Environment, IDE):