



中国科学技术大学
University of Science and Technology of China

引 论

《编译原理和技术》

张昱

0551-63603804, yuzhang@ustc.edu.cn

中国科学技术大学
计算机科学与技术学院



1

编程语言及设计

2

编译器及形式

3

编译器的阶段

4

示例：程序的表示

5

基础实验的考虑



1

编程语言及设计

2

编译器及形式

3

编译器的阶段

4

示例：程序的表示

5

基础实验的考虑



□ 什么是编程语言

- **A programming language** is a notation for describing computations to people and to machines.

□ 每种编程语言有自己的计算模型

- 过程型(Procedural): **C, C++, C#, Java**
- 声明型(Declarative): **SQL, ...**
- 逻辑型(Logic): **Prolog, ...**
- 函数式(Functional): **Lisp/Scheme, Haskell, ML, ...**
- 脚本型(Scripting): **AWK, Perl, Python, PHP, Ruby, ...**



求最大公约数 gcd

```
int gcd(int a, int b) {                                     // C
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}
```

```
let rec gcd a b =                                         (* OCaml *)
    if a = b then a
    else if a > b then gcd b (a - b)
    else gcd a (b - a)
```

```
gcd(A,B,G) :- A = B, G = A.                               % Prolog
gcd(A,B,G) :- A > B, C is A-B, gcd(C,B,G).
gcd(A,B,G) :- B > A, C is B-A, gcd(C,A,G).
```



编程语言百花齐放

<https://octoverse.github.com/>

□ 众多：GitHub --开源项目涉及337种编程语言 (2017.9)

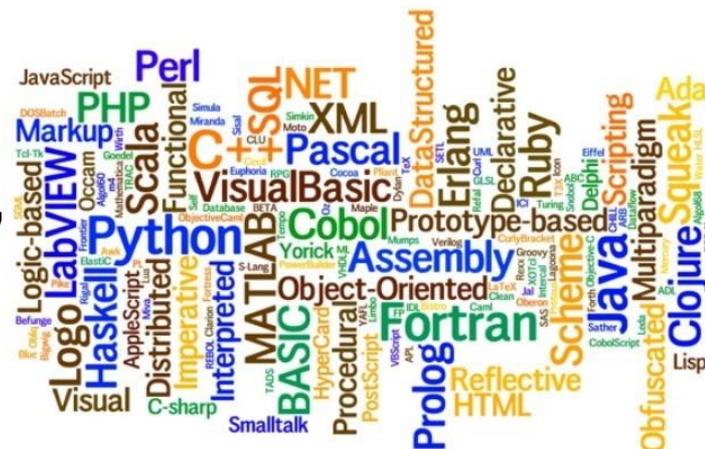
■ <http://www.99-bottles-of-beer.net/> 用1500种语言编写

□ 语言自身在不断发展

■ C - C90, C99, C11;

■ C++ - 1998,..., 2011, 2014, 2017

□ 最流行的语言(动态变化)



Tiobe <https://www.tiobe.com/tiobe-index/>

■ 2017.8: **1**Java, **2**C, **3**C++, **4**C#, **5**Python, **6**VB .NET

■ 2010年: **1**Java, **2**C, **3**PHP, **4**C++, **5**VB, **6**C#, **7**Python

■ 1970年: **1**Fortran, **2**Lisp, **3**Cobol, **4**Algol60, **5**APL, **6**Snobol4



99 Bottles of Beer

**99 bottles of beer on the wall, 99 bottles of beer.
Take one down and pass it around, 98 bottles of beer on the wall.**

**98 bottles of beer on the wall, 98 bottles of beer.
Take one down and pass it around, 97 bottles of beer on the wall.**

·
·
·

**2 bottles of beer on the wall, 2 bottles of beer.
Take one down and pass it around, 1 bottle of beer on the wall.**

**1 bottle of beer on the wall, 1 bottle of beer.
Take one down and pass it around, no more bottles of beer on the wall.**

**No more bottles of beer on the wall, no more bottles of beer.
Go to the store and buy some more, 99 bottles of beer on the wall.**

[Traditional]



C: 99 Bottles of Beer

```
#define MAXBEER (99)
void chug(int beers);
main() {
    register beers;
    for(beers = MAXBEER; beers; chug(beers--))
        puts("");

    puts("\nTime to buy more beer!\n");
    exit(0);
}
void chug(register beers) {
    char howmany[8], *s;
    s = beers != 1 ? "s" : "";
    printf("%d bottle%s of beer on the wall,\n", beers, s);
    printf("%d bottle%s of beeeer . . . ,\n", beers, s);
    printf("Take one down, pass it around,\n");

    if(--beers) sprintf(howmany, "%d", beers); else strcpy(howmany, "No more");
    s = beers != 1 ? "s" : "";
    printf("%s bottle%s of beer on the wall.\n", howmany, s);
}
```

[Bill Wein]



Java: 99 Bottles of Beer

```
class bottles {  
  
    public static void main(String args[]) {  
        String s = "s";  
  
        for (int beers=99; beers>-1;) {  
            System.out.print(beers + " bottle" + s + " of beer on the wall, ");  
            System.out.println(beers + " bottle" + s + " of beer, ");  
  
            if (beers==0) {  
                System.out.print("Go to the store, buy some more, ");  
                System.out.println("99 bottles of beer on the wall.\n");  
                System.exit(0);  
            } else  
                System.out.print("Take one down, pass it around, ");  
  
            s = (--beers == 1)?"": "s";  
            System.out.println(beers + " bottle" + s + " of beer on the wall.\n");  
        }  
    }  
}
```



AWK: 99 Bottles of Beer

```
BEGIN {
  for(i = 99; i >= 0; i--) {
    print ubottle(i), "on the wall,", lbottle(i) "."
    print action(i), lbottle(inext(i)), "on the wall."
    print
  }
}
function ubottle(n) {
  return sprintf("%s bottle%s of beer", n ? n : "No more", n - 1 ? "s" : "")
}
function lbottle(n) {
  return sprintf("%s bottle%s of beer", n ? n : "no more", n - 1 ? "s" : "")
}
function action(n) {
  return sprintf("%s", n ? "Take one down and pass it around," : \
    "Go to the store and buy some more,")
}
function inext(n) {
  return n ? n - 1 : 99
}
```

[Osamu Aoki, <http://people.debian.org/~osamu>]



□ 为什么那么多语言？

- 单个语言不能适用所有应用
- 程序员对语言的好坏、如何编程有自己的观点和看法
- 没有评价语言好坏的普遍接受的标准

□ 语言进化之驱动力

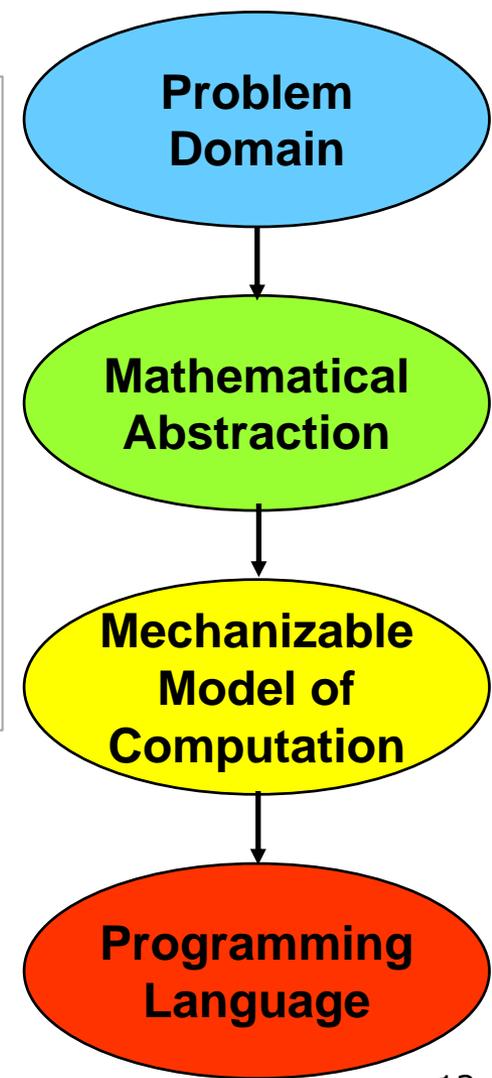
- 应用的多样性
- 提高软件开发生产力(producity)
- 改善软件的安全性、可靠性和可维护性
- 支持并行(parallelism)与并发(concurrency)
- 移动和分发，模块化，多范型

□ 计算思维 (Computational Thinking)



Computational thinking is a **fundamental skill for everyone**, not just for computer scientists. To reading, writing, and arithmetic, **we should add computational thinking to every child's analytical ability**. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.

Jeannette M. Wing
Computational Thinking
CACM, vol. 49, no. 3, pp. 33-35, 2006



□ 语言设计中的计算思维



主要内容

1

编程语言及设计

2

编译器及形式

3

编译器的阶段

4

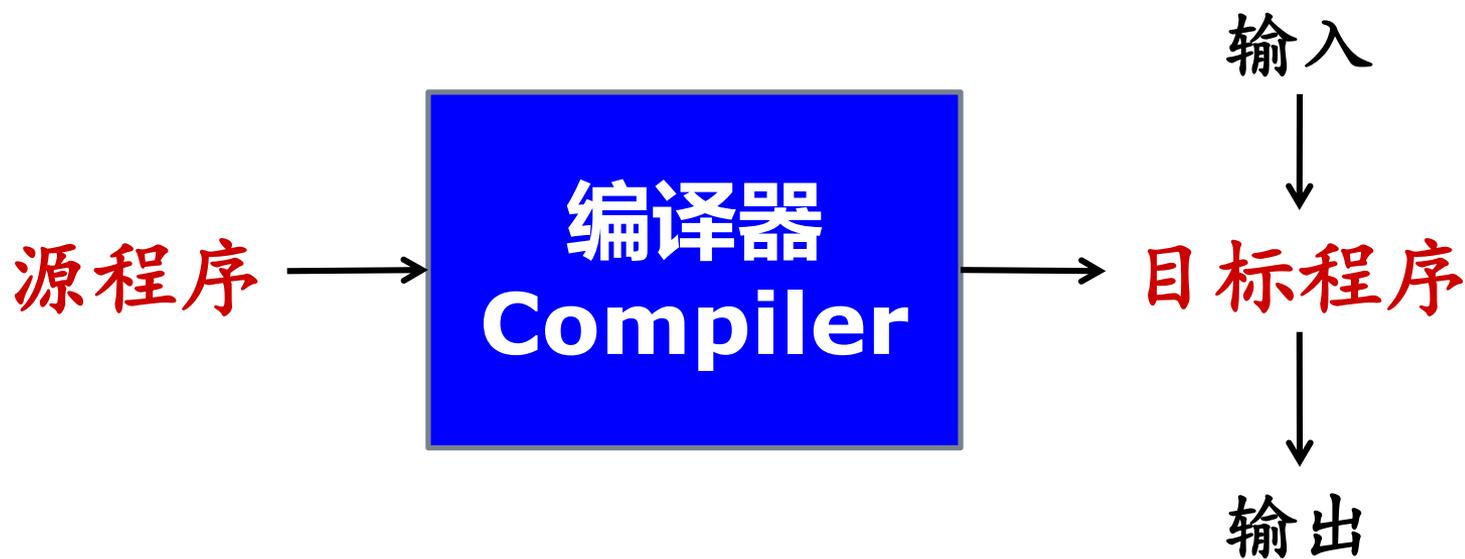
示例：程序的表示

5

基础实验的考虑



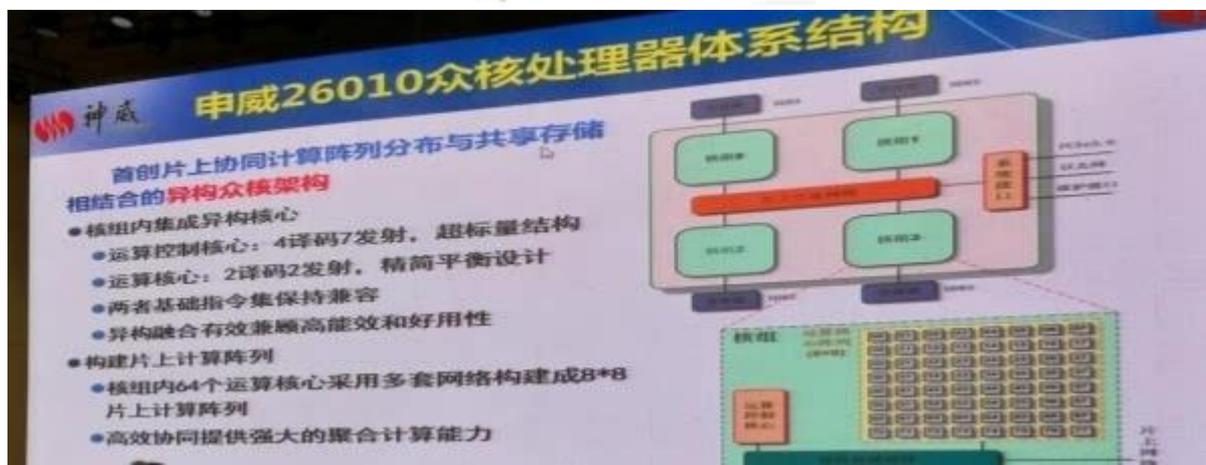
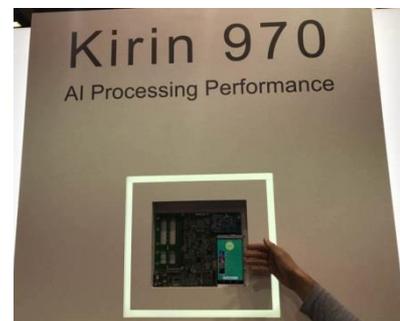
编译器是什么





目标语言

- 另一种编程语言
- CISCs (复杂指令集) : x86、IA64、...
- RISCs (精简指令集) : MIPS、ARM、...
- 多核/众核
- GPUs : CUDA、OpenCL
- FPGAs
- 量子计算机
- TPU, NPU
- ...





直接在输入上执行源程序



源程序

.java



中间表示

Intermediate Representation

.class



输入



输出



编译器的其他形式

□ 交叉编译器 (Cross compiler)

- 在一个平台上生成另一个平台上的代码

PC → **arm-linux-gcc** → ARM

□ 增量编译器 (Incremental compiler)

- 以增量地编译源程序,只编译修改的部分,如 [Freeline](#)

□ 即时编译器 (Just-in-time compiler)

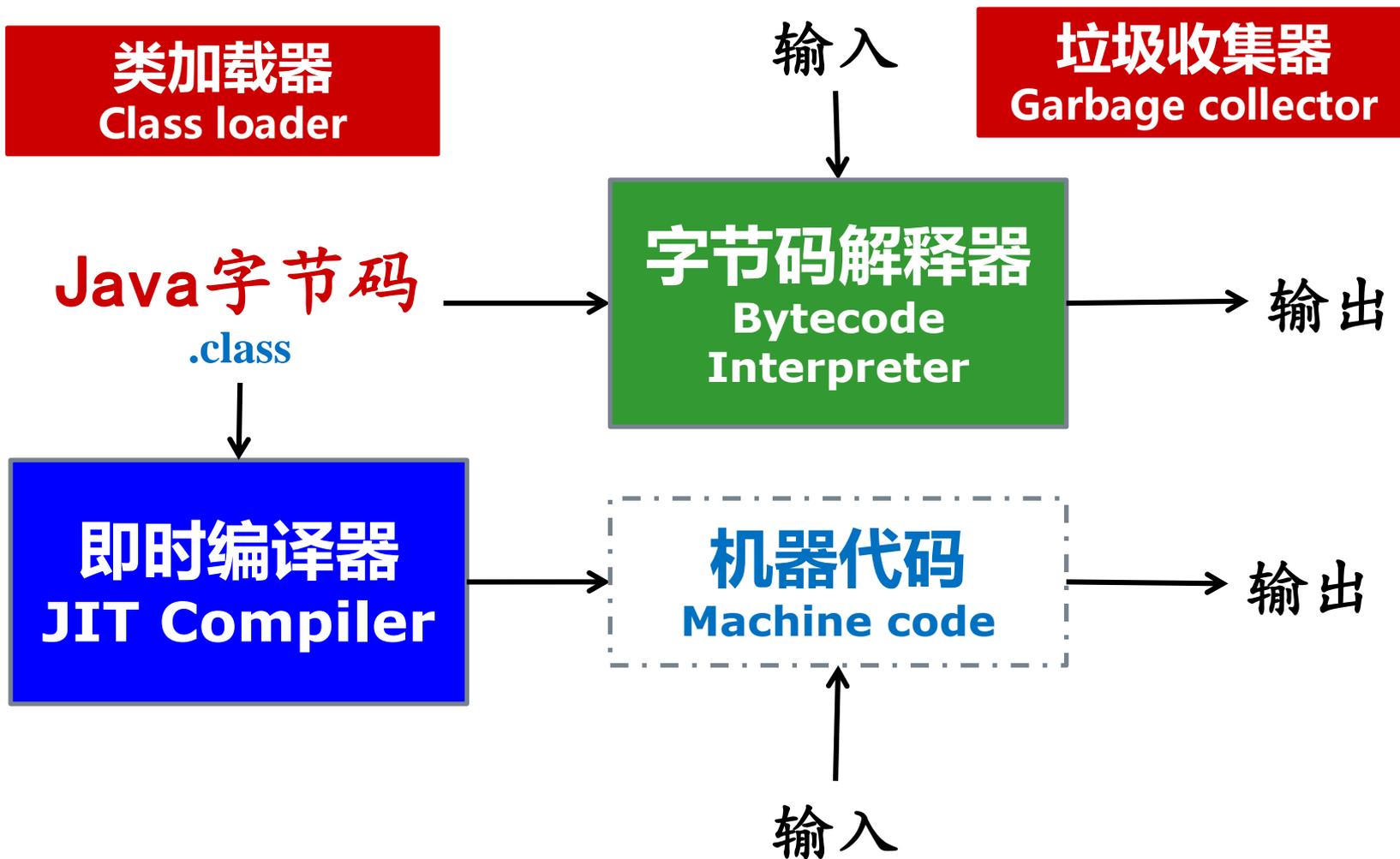
- 在运行时对IR中每个被调用的方法进行编译,得到目标机器的本地代码,如 Java VM 中的即时编译器

□ 预先编译器 (Ahead-of-time compiler)

- 在程序执行之前将IR翻译成本地码,如 ART中的AOT

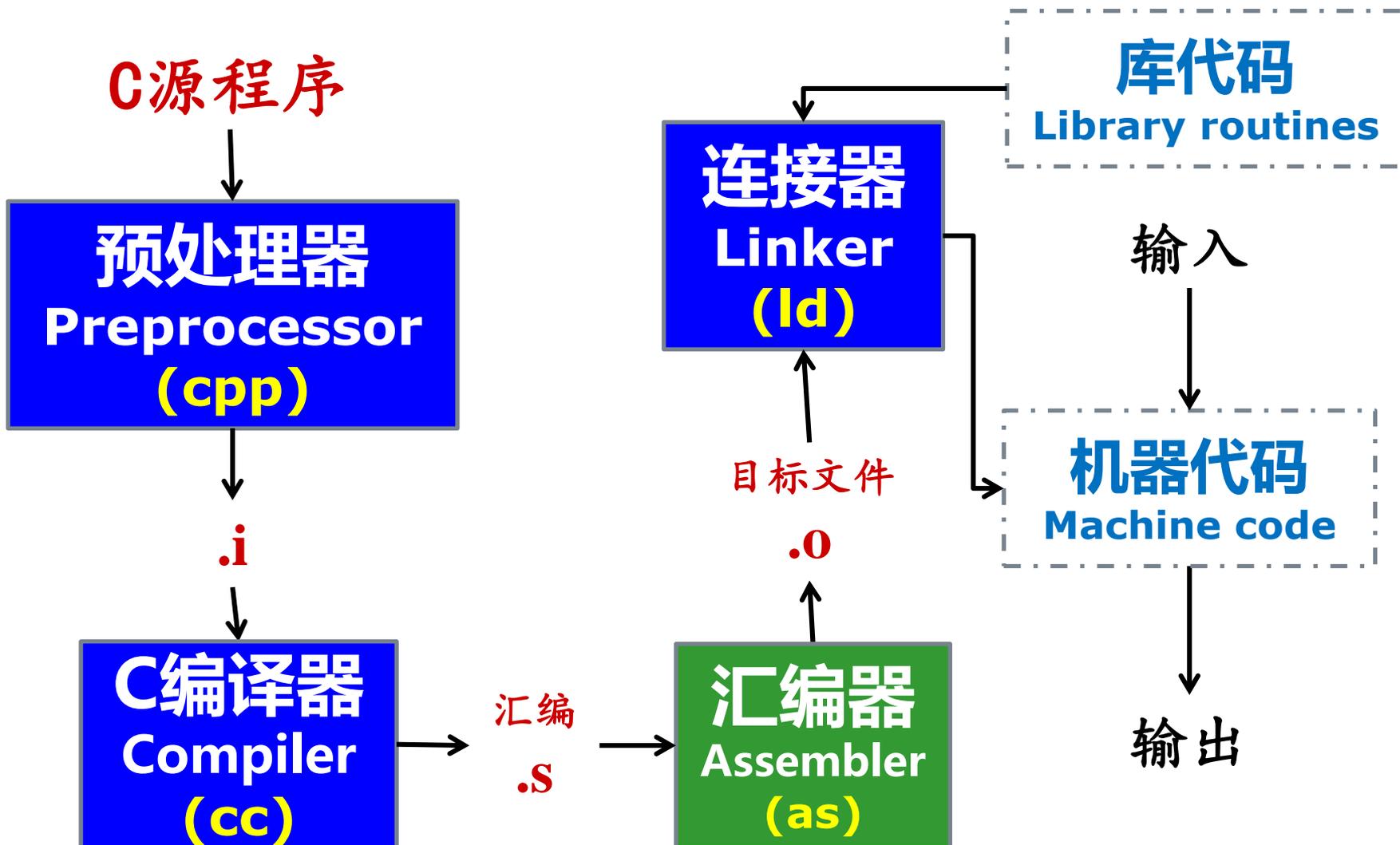


Java虚拟机





C 编译器





主要内容

1

编程语言及设计

2

编译器及形式

3

编译器的阶段

4

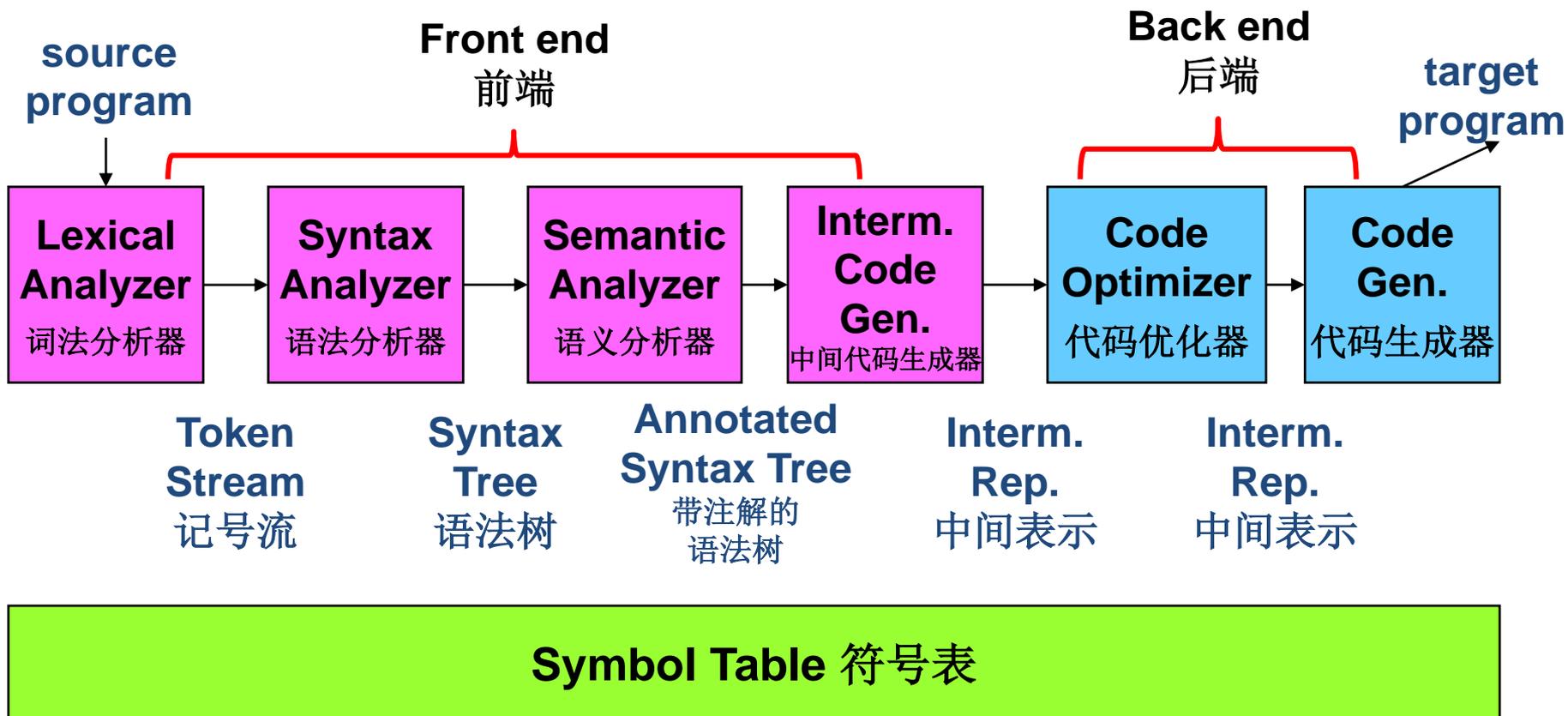
示例：程序的表示

5

基础实验的考虑

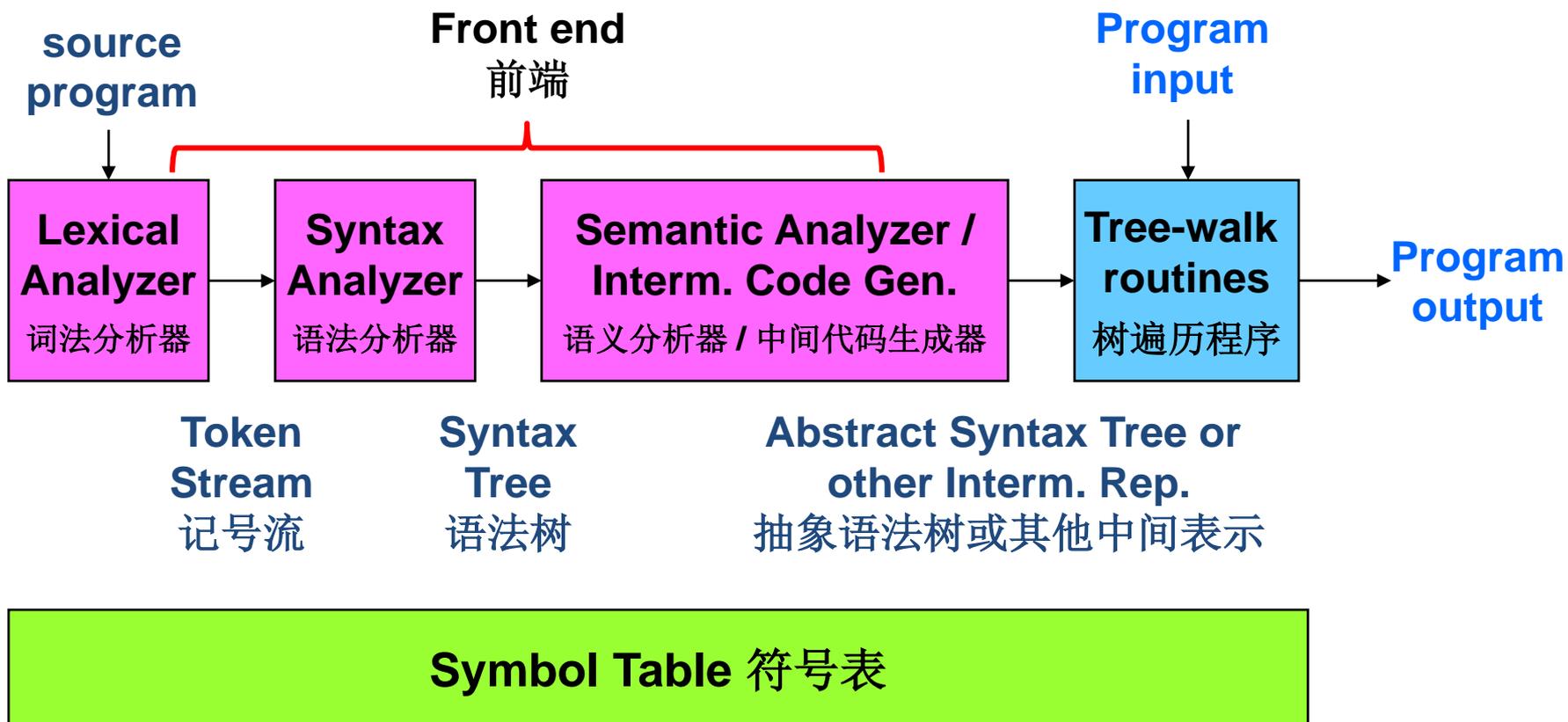


编译器的阶段





编译器的阶段





主要内容

1

编程语言及设计

2

编译器及形式

3

编译器的阶段

4

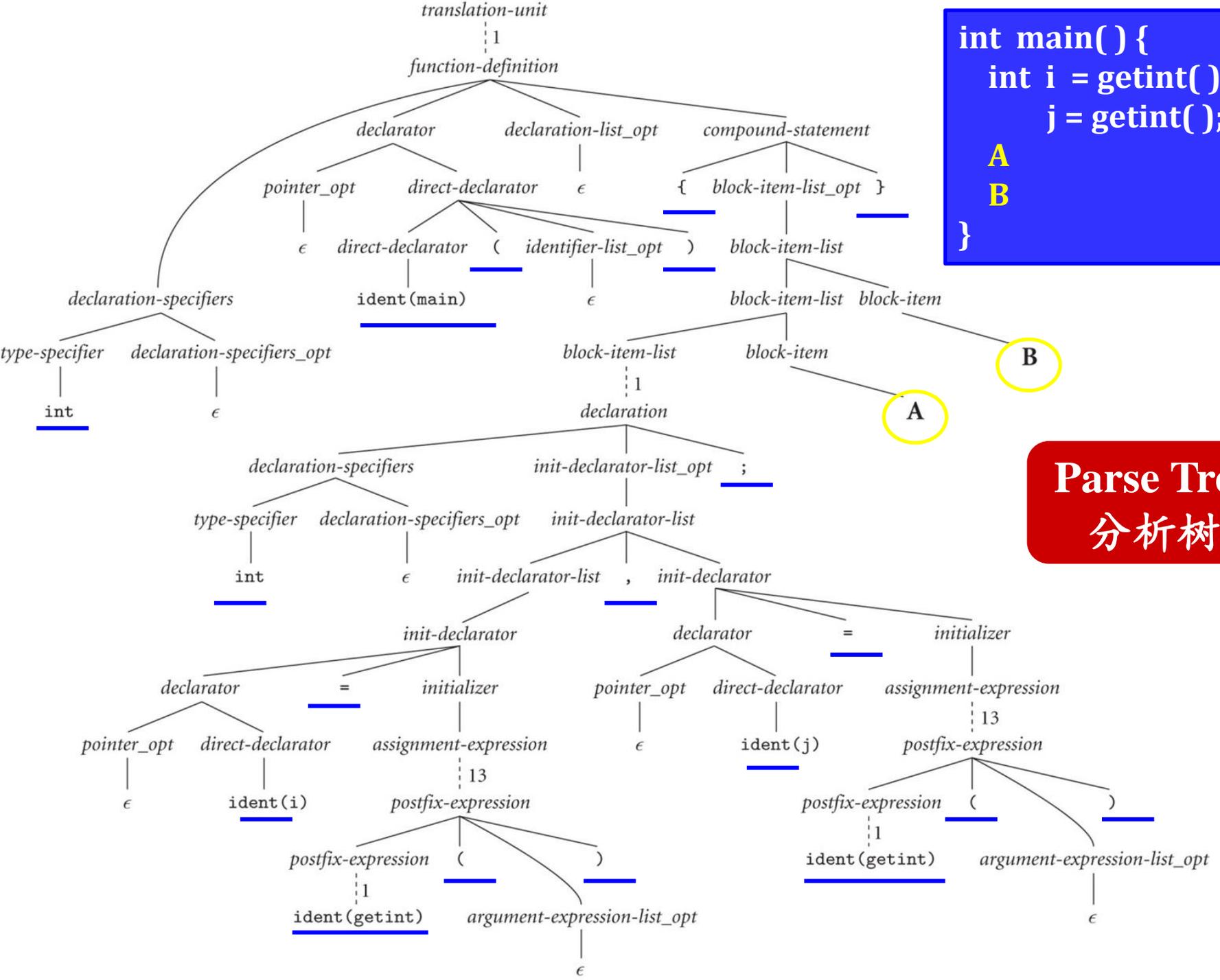
示例：程序的表示

5

基础实验的考虑

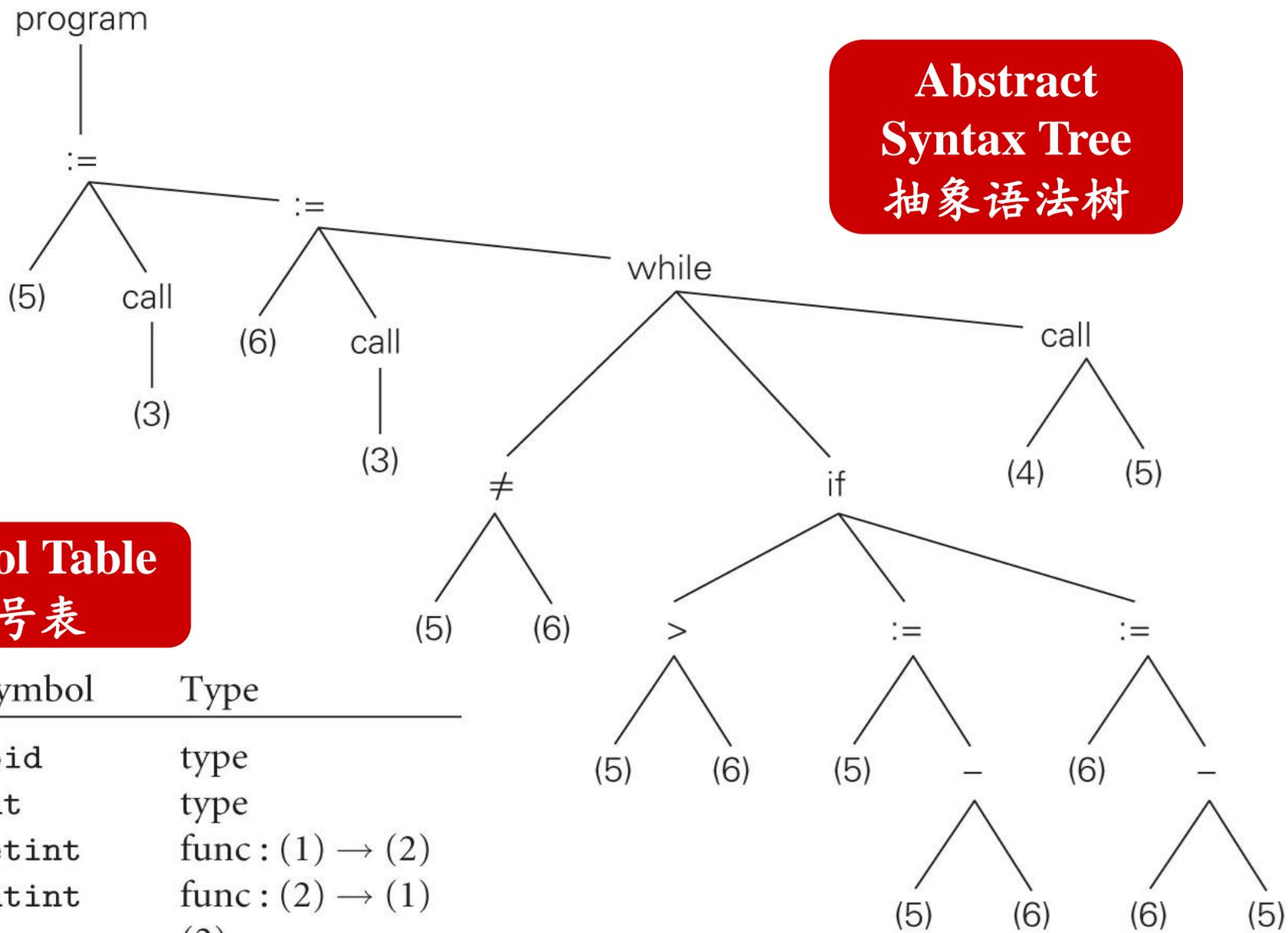
```
int main() {
    int i = getInt(),
        j = getInt();
}

```



Parse Tree
分析树

Abstract Syntax Tree 抽象语法树



Symbol Table 符号表

Index	Symbol	Type
1	void	type
2	int	type
3	getint	func: (1) → (2)
4	putint	func: (2) → (1)
5	i	(2)
6	j	(2)



```
pushl   %ebp                # \  
movl    %esp, %ebp         # ) reserve space for local variables  
subl    $16, %esp          # /  
call    getint             # read  
movl    %eax, -8(%ebp)     # store i  
call    getint             # read  
movl    %eax, -12(%ebp)    # store j
```

```
A:  movl    -8(%ebp), %edi    # load i  
     movl    -12(%ebp), %ebx  # load j  
     cmpl   %ebx, %edi       # compare  
     je     D                # jump if i == j  
     movl    -8(%ebp), %edi    # load i  
     movl    -12(%ebp), %ebx  # load j  
     cmpl   %ebx, %edi       # compare  
     jle   B                # jump if i < j  
     movl    -8(%ebp), %edi    # load i  
     movl    -12(%ebp), %ebx  # load j  
     subl   %ebx, %edi        # i = i - j  
     movl    %edi, -8(%ebp)   # store i  
     jmp    C
```

```
B:  movl    -12(%ebp), %edi    # load j  
     movl    -8(%ebp), %ebx    # load i  
     subl   %ebx, %edi        # j = j - i  
     movl    %edi, -12(%ebp)  # store j
```

```
C:  jmp    A
```

```
D:  movl    -8(%ebp), %ebx    # load i  
     push   %ebx              # push i (pass to putint)  
     call   putint            # write
```

```
     addl   $4, %esp          # pop i  
     leave  # deallocate space for local variables  
     mov    $0, %eax          # exit status for program  
     ret
```

```
int main() {
```

```
    int i = getint(),  
        j = getint();
```

```
    while (i != j) {  
        if (i > j) i = i - j;  
        else j = j - i;
```

```
    }  
    putint(i);  
}
```



主要内容

1

编程语言及设计

2

编译器及形式

3

编译器的阶段

4

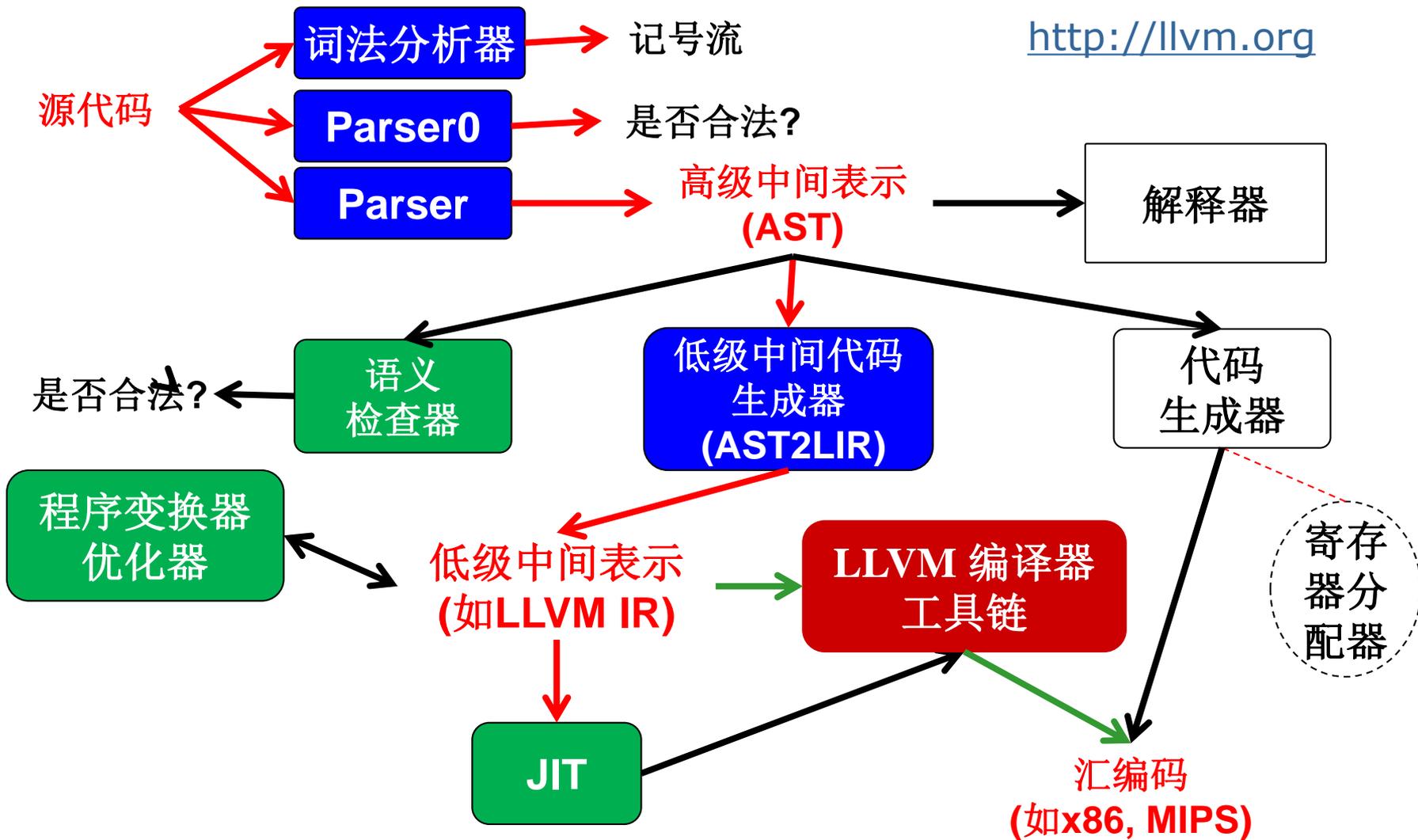
示例：程序的表示

5

基础实验的考虑



基础实验的考虑





□ 生成器

- 生成Lexer: [Flex](#) ([for windows](#))、[Jflex](#)
- 生成Parser
 - LALR: [Bison](#) ([for windows](#))、[Java CUP](#)
 - LL: [JavaCC](#)、[ANTLR](#) – LL(*) [[PLDI2011](#)]

□ 文法对Parser的影响

- LR Parser的优势: 速度快、表达能力强
- LL Parser的优势: 代码结构与文法对应, 易理解, 容易增加错误处理和错误恢复

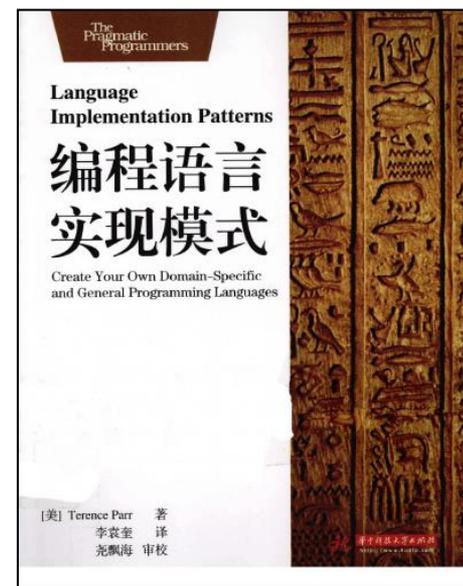
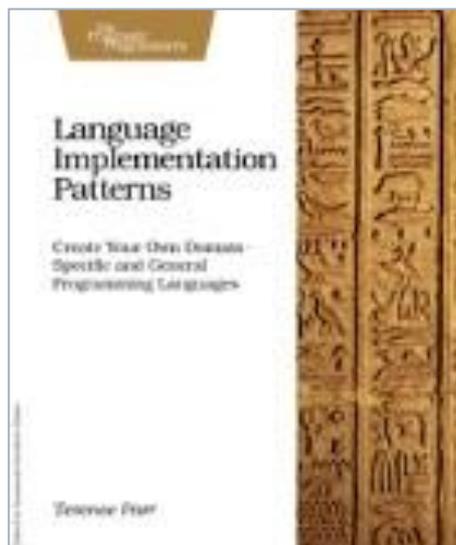
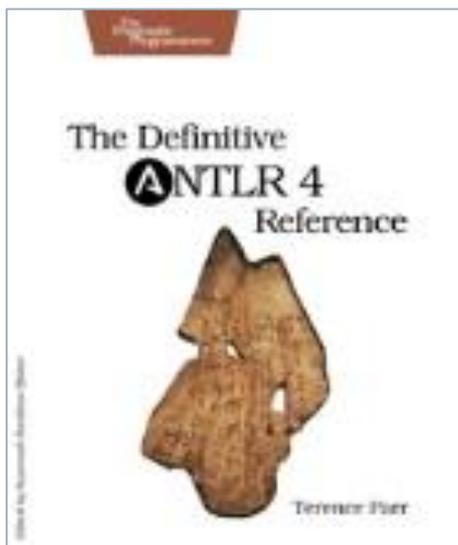
ANTLR(ANother Tool for Language Recognition)

[<http://www.antlr.org/>]

□ Prof. Terence Parr , since 1989

□ 支持多种代码生成目标

Java、C++、C#、Python、Go、JavaScript、Swift





□ 开源：源码阅读，消化、理解生成器基于的原理

□ 各种语言实现：31种模式

1. 从文法到递归下降识别器
2. LL(1)递归下降词法分析器
3. LL(1)递归下降语法解析器
4. LL(k)递归下降语法解析器
5. 回溯解析器
6. 记忆解析器
7. 谓词解析器
8. 解析树（分析树）
9. 同型抽象语法树
10. 规范化异型AST
11. 不规则的异型AST

12. 内嵌遍历器
13. 外部访问者
14. 文法访问者
15. 模式匹配者（子树）
16. 单作用域符号表
17. 嵌套作用域符号表
18. 数据聚集的符号表
19. 类的符号表型

20. 计算表达式的类型
21. 自动类型提升
22. 静态类型检查
23. 动态类型检查
24. 语法制导的解释器
25. 基于树的解释器
26. 字节码汇编器
27. 栈式解释器
28. 寄存器解释器
29. 语法制导的翻译器
30. 基于规则的翻译器
31. 模型驱动转换





我听到的会忘掉，
我看到的能记住，
我做过的才真正明白。