

6.035

2005年秋季

讲座1 绪论

计算机语言工程绪论

概要

- 课程署信息
- 计算机语言工程绪论
 - 什么是编译器?
 - 为什么要学习编译器?
 - 编译器的解析原理

课程署

- 师资
- 教材
- 课程概要
- 项目
- 项目组
- 评分规则

师资

讲师

- Saman Amarasinghe 教授
- Martin Rinard 教授

可选教材

龙
书

- Modern Compiler Implementation in Java
A.W.Apple著
牛津出版社，1998

虎
书

- Advanced Compiler design and Implementation
Steven Muchnick著
摩根考夫曼出版社，1997

鲸
书

- Compilers—Principles, Techniques and Tools
Aho, Sethi和Ullman合著
阿狄森—韦斯利出版社，1998

五个阶段

- ①词义及语法分析
- ②语义分析
- ③代码生成
- ④数据流优化
- ⑤指令优化

每阶段内容

- 阶段开始
 - 项目描述
- 讲座
 - 2到4个讲座
- 设计项目时间
- （项目检查点）
- 设计项目时间
- 项目完成时间
 - 项目描述

项目组

- 每个项目组由**3到4**个学生组成
- 评分规则

所有组员（在大多数情况下）所得分数相同

小组与助教的周会

- 助教将安排一个30分钟的周会
- 小组可以在会议中：
 - 询问项目的相关问题
 - 讨论设计决策
 - 描述个人所做工作
 - 回答助教关于项目的问题
- 助教通过这个来衡量个人对项目的贡献 → 就在这里！

评分规则

- 编译器项目 58%
- 论文讨论 12% (每篇3%)
- 课堂测试 30% (每次10%)

项目的评分规则

• 一扫描器/Parser	10%
• 一语义检测	10%
• 一代码生成	14%
• 一数据流优化	12%
• 指令优化	12%
	<hr/>
	58%

测试

- 共三个测试
- 课堂测试
 - 时间为**50**分钟（请准时！）
 - 可查阅书和笔记

论文讨论

- 将分发三篇论文
- 阅读论文，思考.....
 - 你喜欢这项研究吗？
 - 该项研究的研究背景是什么？
 - 这篇论文有出乎你预料之处吗？
 - 你认为其实验结果是否可验证所运用的研究方法？
 - 作者是否在更广泛意义上理解他所做的研究？
 - 在论文发表后，该领域产生了怎样的变化？
 - 在未来，这项研究会有怎样的重要地位？
- 写一篇150到200字的论文总结。
- 就这篇论文，将安排与教授或助教进行一对一的会议（15或20分钟）。

课程提纲

- 课程署信息
- 计算机语言工程绪论
 - 什么是编译器？
 - 为什么要学习编译器？
 - 编译器的解析

什么是计算机语言工程

- 如果对一台计算机输入指令
- 如果让计算机高效的执行指令

编程语言的力量

- 可用来描述任何行为
 - 不受“上下文”约束
- 可用多种方式描述同一行为
 - 灵活多变

怎样操控计算机

- 可使用自然语言么？
 - 英语??
 - “亨利，打开分离舱门。”
 - “对不起，戴夫，恐怕我不能那么做。”
 - 我们还没有做到这一步!!

- 自然语言：
 - 同一表达描述了多种可能行为
 - 有歧异

怎样操控计算机

- 可以用自然语言么？
 - 英语??
 - “亨利，打开分离舱门。”
 - “对不起，戴夫，恐怕我不能那么做。”
 - 我们还没做到这一步！！
- 使用编程语言：
 - 例如：
 - Java, C, C++, Pascal, BASIC, Scheme

编程语言

- 特性
 - 需要精确
 - 需要简练
 - 需要表述能力强
 - 需要是高级语言（具有很高的抽象性）

从高级抽象描述到低级细节实现

总统



我的选票率很低，侵略一个小国家

将军



越过河流，占据防守位置

中士



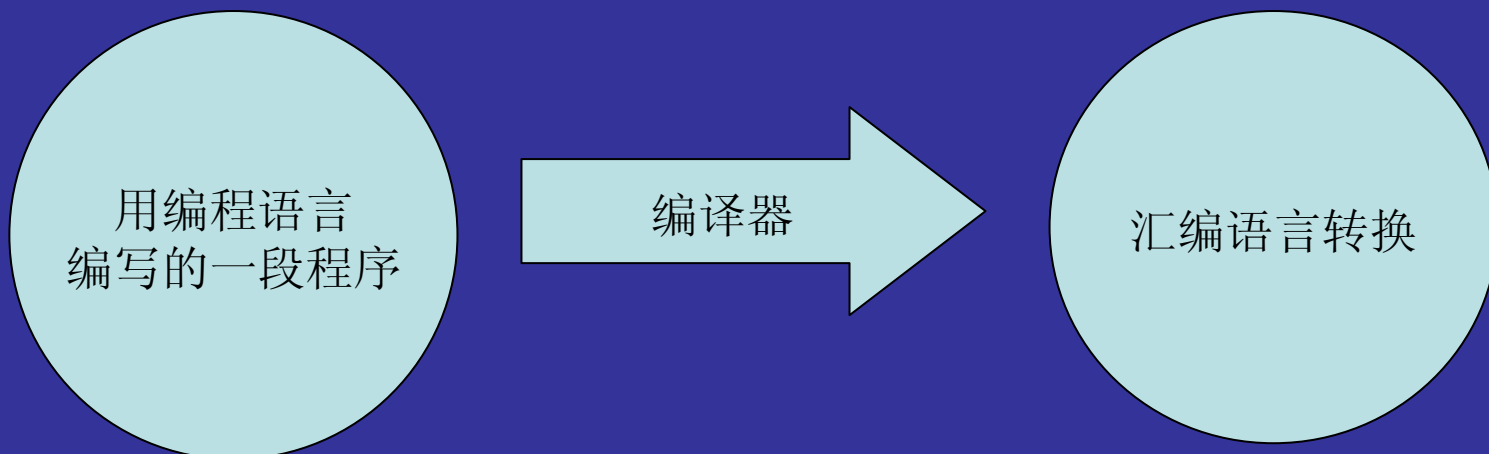
前进，向左转，立定！，开枪

步兵



1. 怎样操控计算机

- 用一种编程语言编写一段程序
 - 高级抽象描述
- 微处理器只能识别汇编语言
 - 低级细节执行



1. 怎样操控计算机

- 输入：高级编程语言
- 输出：低级汇编指令

- 编译器必须做的是：
 - 读取并理解程序
 - 精确判断程序表达的命令
 - 解决如何执行这些命令
 - 操控计算机执行命令

实例（输入程序）

```
int expr(int n)
{
    int d;
    d=4*n*n*(n+1)*(n+1);
    return d;
}
```

实例（输出汇编代码）

```
    lda $30, -32($30)
    stq $26, 0($30)
    stq $15, 8($30)
    bis $30, $30, $15
    bis $16, $16, $1
    stl $1, 16($15)
    lds $f1, 16($15)
    sts $f1, 24($15)
    ldl $5, 24($15)
    bis $5, $5, $2
    s4addq $2, 0, $3
    ldl $4, 16($15)
    mull $4, $3, $2
    ldl $3, 16($15)
    addq $3, 1, $4
    mull $2, $4, $2
    ldl $3, 16($15)
    addq $3, 1, $4
    mull $2, $4, $2
    stl $2, 20($15)
    ldl $0, 20($15)
    stl $2, 20($15)
    br $31, $33
$33:
    bis $15, $15, $30
    ldq $26, 0($30)
    ldq $15, 8($30)
    addq $30, $32, $30
    ret $31, ($26), 1
```

命令的高效执行

将军



越过河流，占据防守位置

中士



步兵



命令的高效执行

将军



越过河流，占据防守位置

中士



在哪越过河流？使用上游的桥还是越过下游偷袭敌人？
如何减少伤亡？

步兵



从高级抽象描述到低级细节实现

总统



我的选票率很低，侵略一个小国家

将军



俄罗斯还是百慕大？或者按兵不动等待他的选票率上去？

命令的高效执行

- 从高到底的映射
 - 从程序到汇编语言的简单映射会造成代码执行不高效
 - 抽象程度越高 \longrightarrow 越不高效
- 如若不高效
 - 这些高级抽象则毫无意义
- 需要：
 - 提供高级抽象
 - 具有发出低级指令的性能

实例

```
int sumcalc(int a, int b, int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*y;
    }
    return x;
}
```

```

test:
    subu    $fp, 16
    sw     zero, 0($fp)        # x=0
    sw     zero, 4($fp)       # y=0
    sw     zero, 8($fp)       # i=0
labl:
    # for(i=0;i<N; i++)
    mul    $t0, $a0, 4        # a*4
    div    $t1, $t0, $a1      # a*4/b
    lw     $t2, 8($fp)        # i
    mul    $t3, $t1, $t2      # a*4/b*i
    lw     $t4, 8($fp)        # i
    addui  $t4, $t4, 1        # i+1
    lw     $t5, 8($fp)        # i
    addui  $t5, $t5, 1        # i+1
    mul    $t6, $t4, $t5      # (i+1)* (i+1)
    addu   $t7, $t3, $t6      # a*4/b*i+ (i+1)* (i+1)
    lw     $t8, 0($fp)        # x
    add    $t8, $t7, $t8      # x=x+a*4/b*i+ (i+1)* (i+1)
    sw     $t8, 0($fp)
    lw     $t0, 4($fp)        # y
    mul    $t1, $t0, a1       # b*y
    lw     $t2, 0($fp)        # x
    add    $t2, $t2, $t1      # x=x+b*y
    sw     $t2, 0($fp)
    lw     $t0, 8($fp)        # i
    addui  $t0, $t0, 1        # i+1
    sw     $t0, 8($fp)
    ble    $t0, $a3, labl

    lw     $v0, 0($fp)
    addu   $fp, 16
    $ra

```

优化

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*y;
    }
    return x;
}
```

常量复制

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*y;
    }
    return x;
}
```

常量复制

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*y;
    }
    return x;
}
```

常量复制

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*y;
    }
    return x;
}
```

常量复制

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*y;
    }
    return x;
}
```

常量复制

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*0;
    }
    return x;
}
```

代数简化

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*0;
    }
    return x;
}
```

代数简化

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+b*0;
    }
    return x;
}
```

代数简化

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+0;
    }
    return x;
}
```

代数简化

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+0;
    }
    return x;
}
```

代数简化

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x+0;
    }
    return x;
}
```

代数简化

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x;
    }
    return x;
}
```

复写传播

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);
        x=x;
    }
    return x;
}
```

复写传播

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);

    }
    return x;
}
```

公共子表达式消除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);

    }
    return x;
}
```

公共子表达式消除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+(4*a/b)*i+(i+1)*(i+1);

    }
    return x;
}
```

公共子表达式消除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y,t;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+(i+1)*(i+1);

    }
    return x;
}
```

公共子表达式消除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y,t;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+t*t;

    }
    return x;
}
```

无用代码删除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y,t;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+t*t;

    }
    return x;
}
```

无用代码删除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y,t;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+t*t;

    }
    return x;
}
```

无用代码删除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y,t;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+t*t;

    }
    return x;
}
```

无用代码删除

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y,t;
    x=0;

    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+t*t;
    }
    return x;
}
```

循环不变量移出

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t;
    x=0;

    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+t*t;
    }
    return x;
}
```

循环不变量移出

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t;
    x=0;

    for(i=0;i<=N;i++){
        t=i+1;
        x=x+(4*a/b)*i+t*t;
    }
    return x;
}
```

循环不变量移出

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u;
    x=0;
    u= (4*a/b);

    for(i=0;i<=N;i++){
        t=i+1;
        x=x+u*i+t*t;
    }
    return x;
}
```

降低强度

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u;
    x=0;
    u= (4*a/b);
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+u*i+t*t;
    }
    return x;
}
```

降低强度

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u;
    x=0;
    u= (4*a/b);
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+u*i+t*t;
    }
    return x;
}
```

u*0,	v=0,
u*1,	v=v+u,
u*2,	v=v+u,
u*3,	v=v+u,
u*4,	v=v+u,
...	...

降低强度

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u,v;
    x=0;
    u= (4*a/b);
    v=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+u*i+t*t;
        v=v+u;
    }
    return x;
}
```

降低强度

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u,v;
    x=0;
    u= (4*a/b);
    v=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+v+t*t;
        v=v+u;
    }
    return x;
}
```

降低强度

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u,v;
    x=0;
    u= (4*a/b);
    v=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+v+t*t;
        v=v+u;
    }
    return x;
}
```

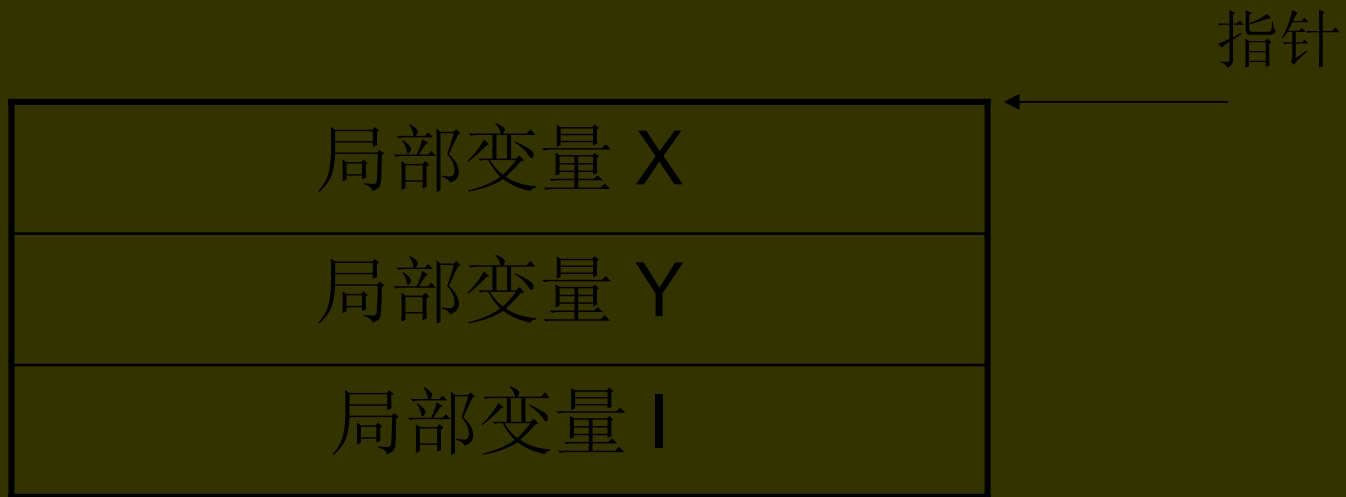
降低强度

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u,v;
    x=0;
    u= (4*a/b);
    v=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+v+t*t;
        v=v+u;
    }
    return x;
}
```

降低强度

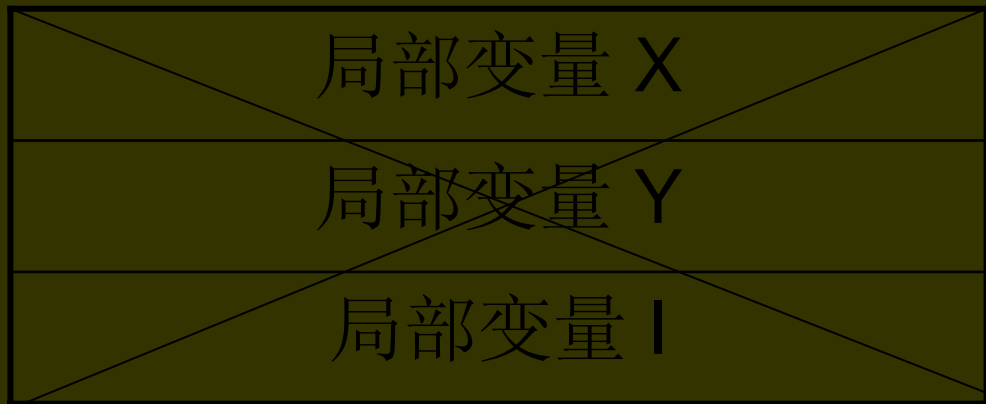
```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u,v;
    x=0;
    u= ((a<<2)/b);
    v=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+v+t*t;
        v=v+u;
    }
    return x;
}
```

寄存器分配



寄存器分配

指针



\$t9 = x

\$t8 = t

\$t7 = u

\$t6 = v

\$t5 = i

降低强度

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u,v;
    x=0;
    u= ((a<<2)/b);
    v=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+v+t*t;
        v=v+u;
    }
    return x;
}
```

test:

```
subu $fp, 16
add $t9, zero, zero      # x=0
sll $t0, $a0, 2          # a<<2
div $t7, $t0, $a1        # u=(a<<2)/b
add $t6, zero, zero     # v=0
add $t5, zero, zero     # i=0
```

labl: # for(i=0,i<N; i++)

```
addui $t8, $t5, 1        # t=i+1
mul $t0, $t8, $t8        # t*t
addu $t1, $t0, $t6       # v+t*t
addu $t9, $t9, $t1       # x=x+v+t*t

addu $6, $6, $7          # v=v+u

addui $t5, $t5, 1        # i=i+1
ble $t5, $a3, labl
```

```
addu $v0, $t9, zero
addu $fp, 16
```

未优化代码

```

test:
    subu    $fp, 16
    sw     zero, 0($fp)
    sw     zero, 4($fp)
    sw     zero, 8($fp)
labl:
    mul    $t0, $a0, 4
    div    $t1, $t0, $a1
    lw     $t2, 8($fp)
    mul    $t3, $t1, $t2
    lw     $t4, 8($fp)
    addui  $t4, $t4, 1
    lw     $t5, 8($fp)
    addui  $t5, $t5, 1
    mul    $t6, $t4, $t5
    addu   $t7, $t3, $t6
    lw     $t8, 0($fp)
    add    $t8, $t7, $t8
    sw     $t8, 0($fp)
    lw     $t0, 4($fp)
    mul    $t1, $t0, a1
    lw     $t2, 0($fp)
    add    $t2, $t2, $t1
    sw     $t2, 0($fp)
    lw     $t0, 8($fp)
    addui  $t0, $t0, 1
    sw     $t0, 8($fp)
    ble   $t0, $a3, labl

    lw     $v0, 0($fp)
    addu   $fp, 16
    b     $ra
    
```

已经优化代码

```

test:
    subu $fp, 16
    add $t9, zero, zero
    sll $t0, $a0, 2
    div $t7, $t0, $a1
    add $t6, zero, zero
    add $t5, zero, zero
labl:
    addui $t8, $t5, 1
    mul $t0, $t8, $t8
    addu $t1, $t0, $t6
    addu $t9, $t9, $t1
    addu $t6, $t6, $t7
    addui $t5, $t5, 1
    ble $t5, $a3, labl

    addu $v0, $t9, zero
    addu $fp, 16
    b $ra
    
```

$$4 * 1d / st + 2 * add / sub + br +$$

$$6 * add / sub + sh$$

$$N * (9 * 1d / st + 6 * add / sub + 4 * mul + div + br)$$

$$N * (5 * add / sub$$

$$= 7 + N * 21$$

$$= 9 + N * 7$$

执行时间=43 秒

执行时间=17 秒

编译器优化程序的作用

- 性能/速度
- 代码大小
- 占用资源大小
- 快速/有效编译
- 安全性/可靠性
- 可调试

编译器有助于提高抽象程度

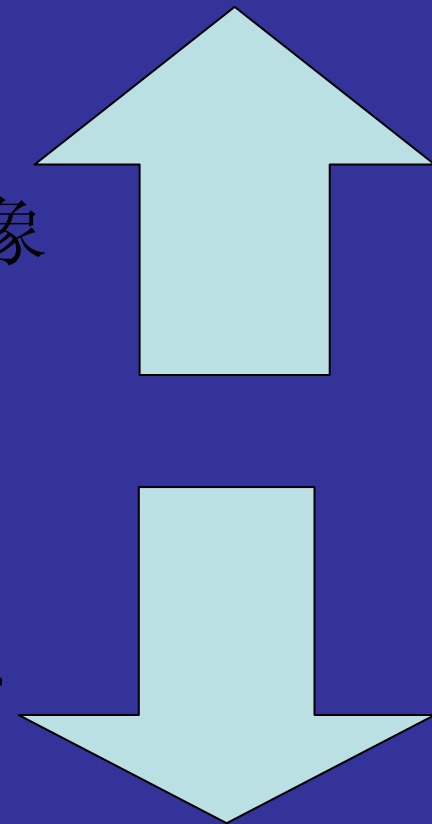
- 编程语言

- 从C语言到具有垃圾搜集器的面向对象 (Object-Oriented) 语言

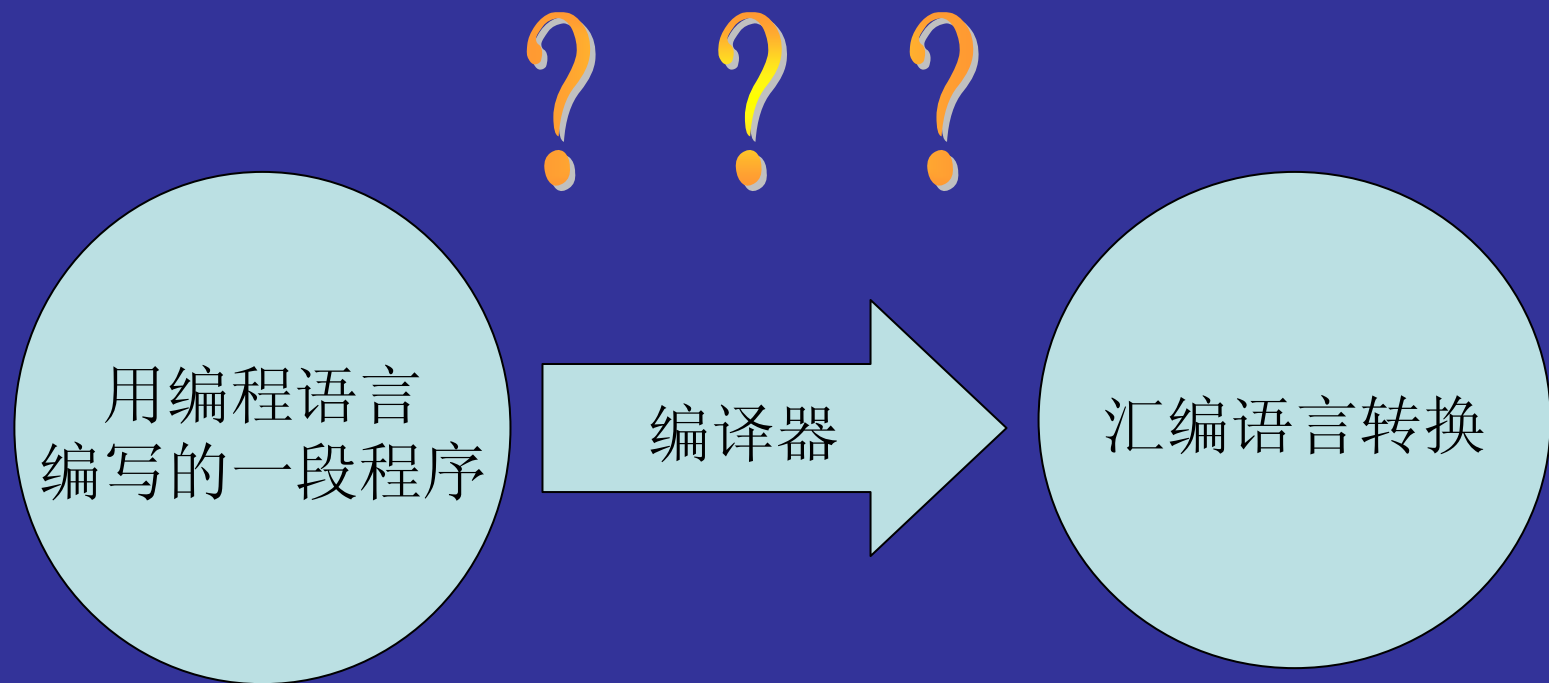
- 更抽象的定义

- 微处理器

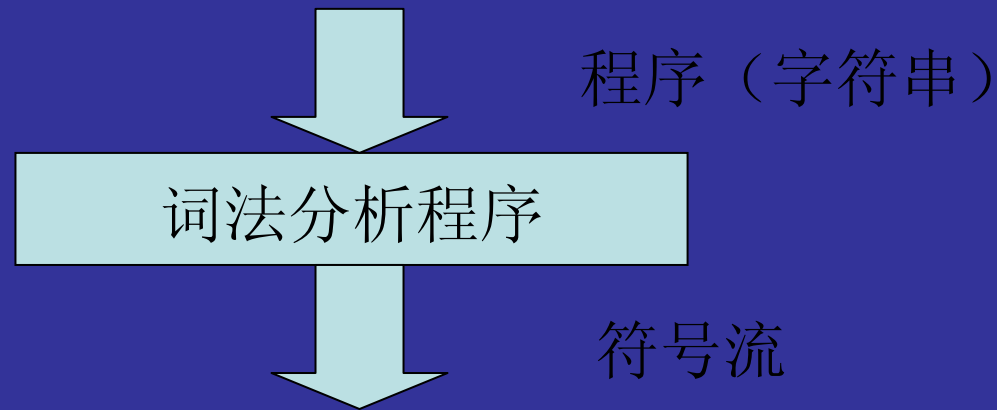
- 从简单的CISC到RISC到VLIW到.....



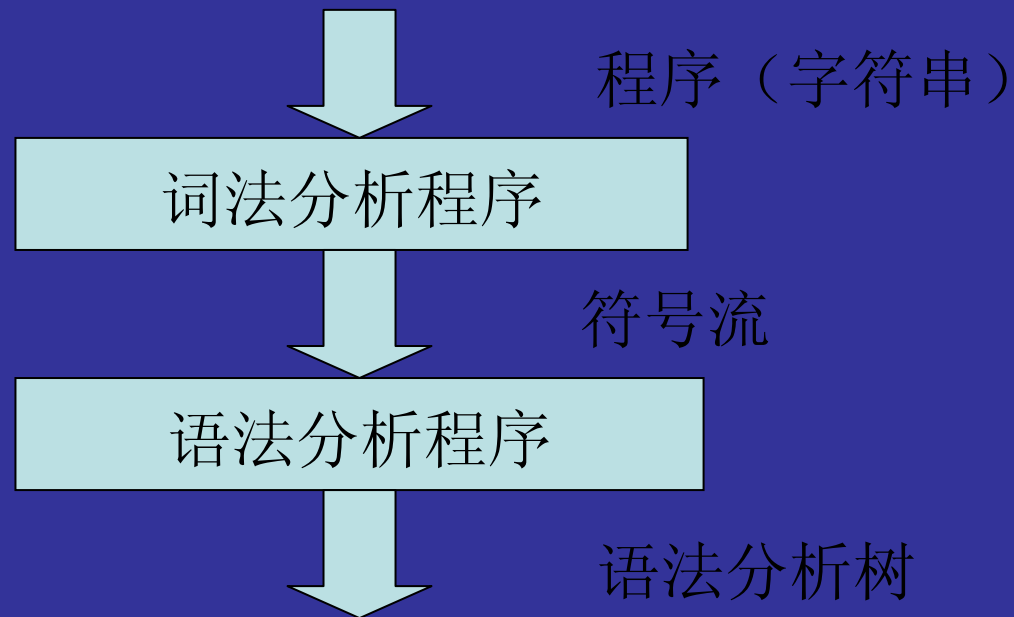
计算机解析原理



计算机解析原理

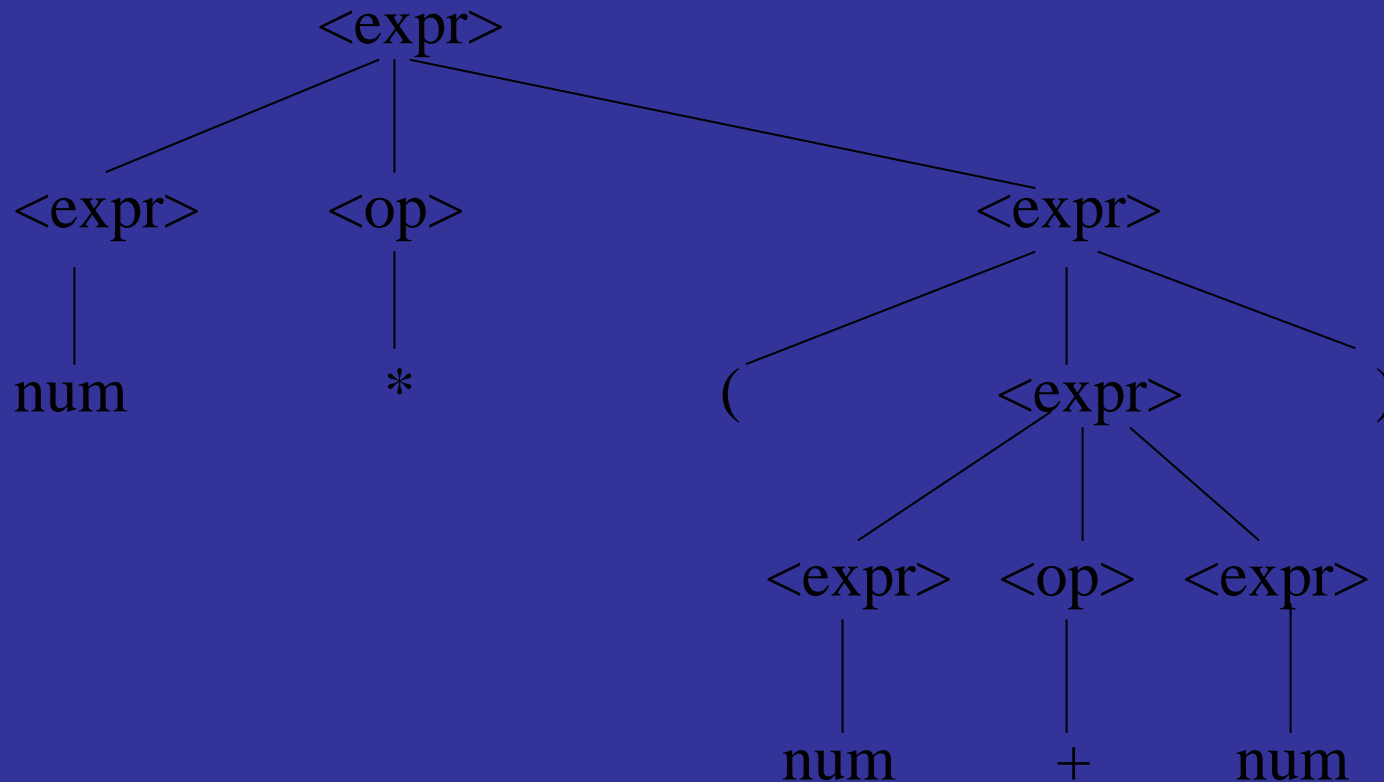


计算机解析原理



语法分析程序

num'*' '(' num '+' num ')'



语法分析程序

```
int * foo(i, j, k,))  
    int l;  
    int j;  
{  
    for(i=0; i j) {  
    fj(j>l)  
    return j;  
}
```

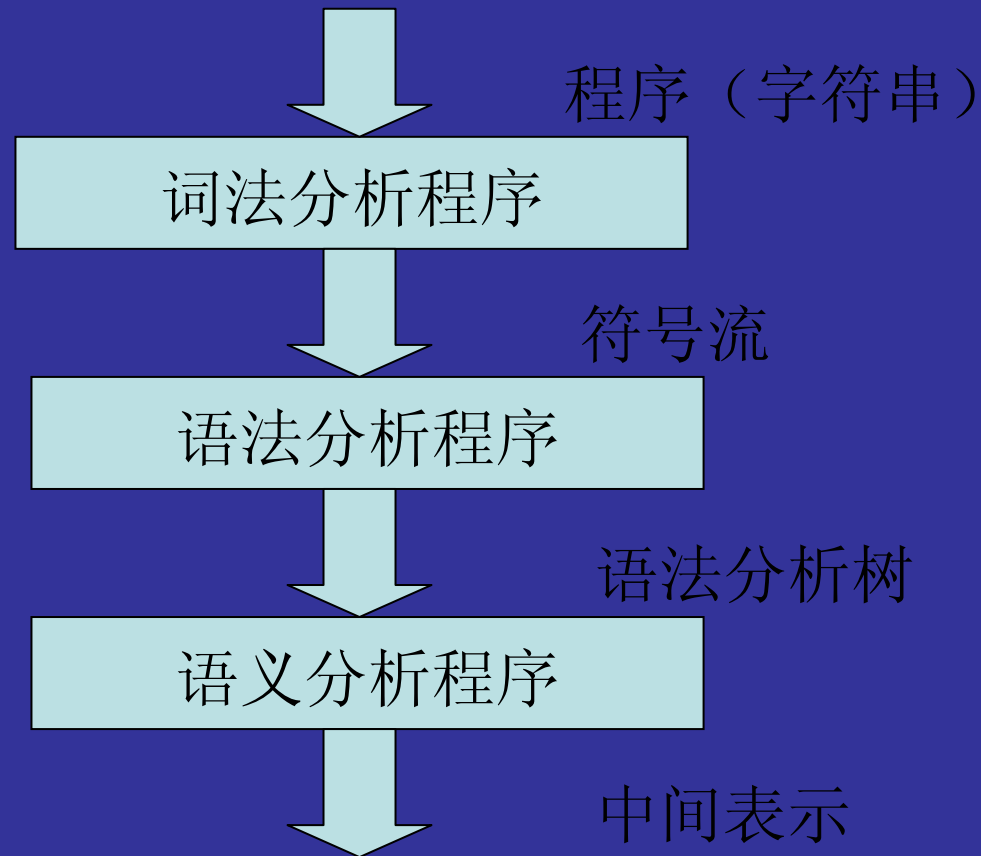
多余的括号

缺少增量

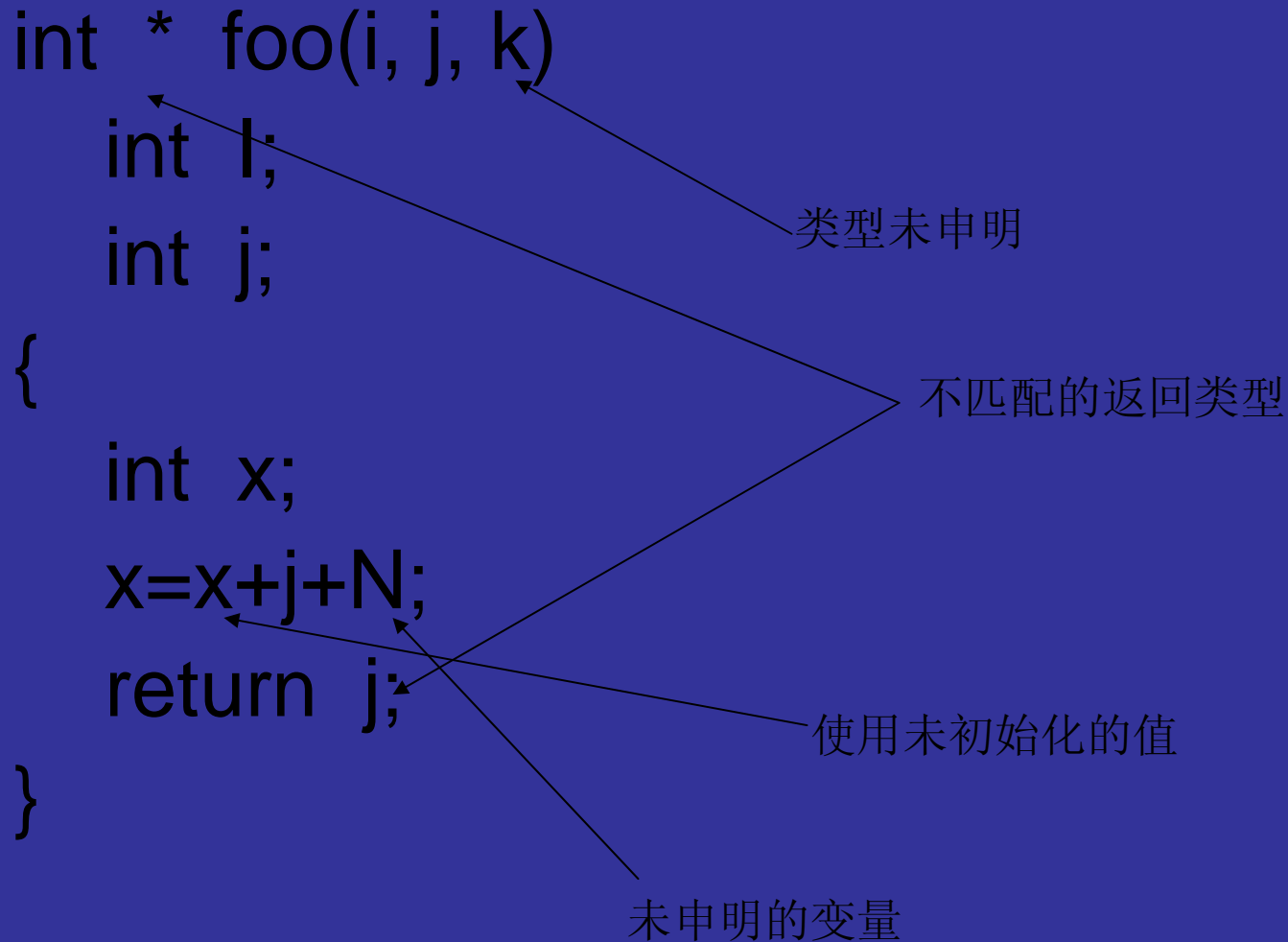
不是一个表达式

不是关键字

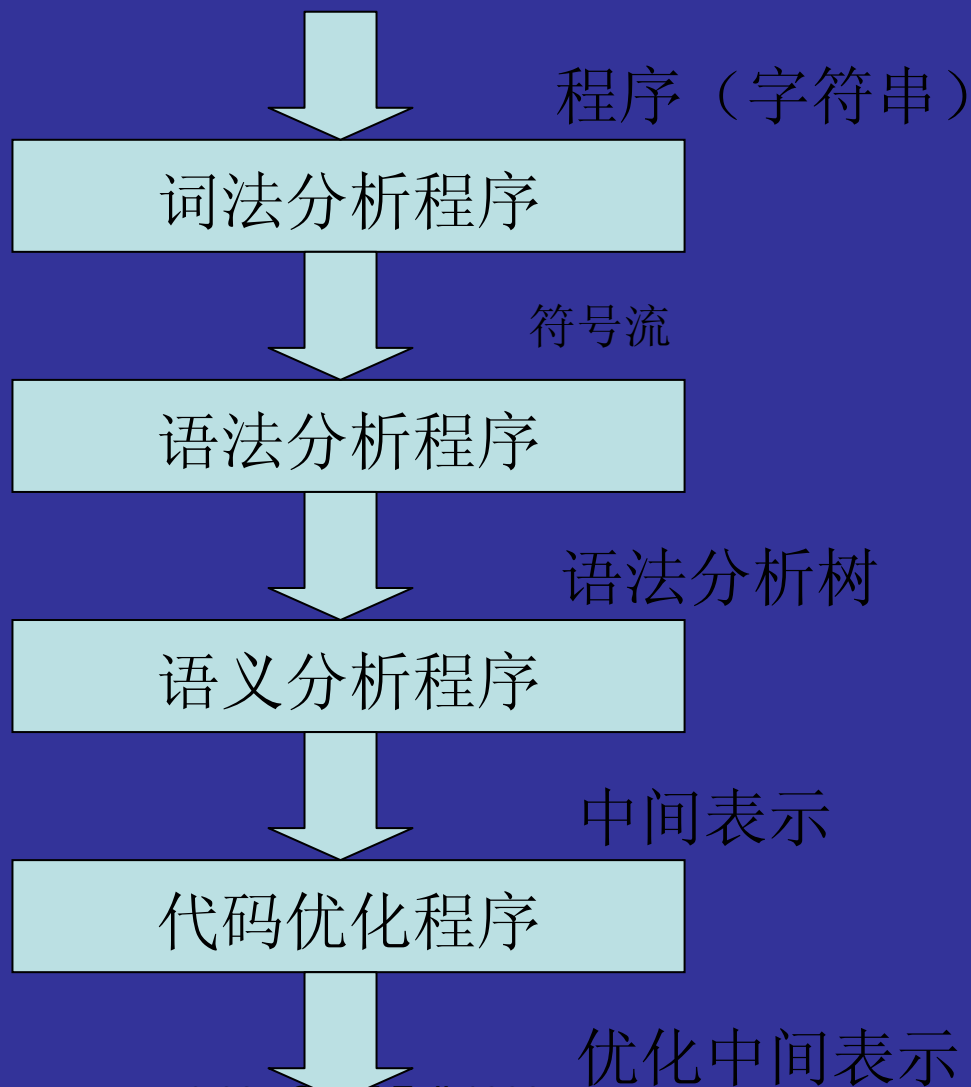
计算机解析原理



语义分析程序

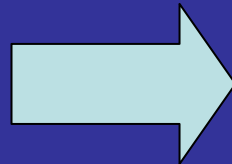


计算机解析原理



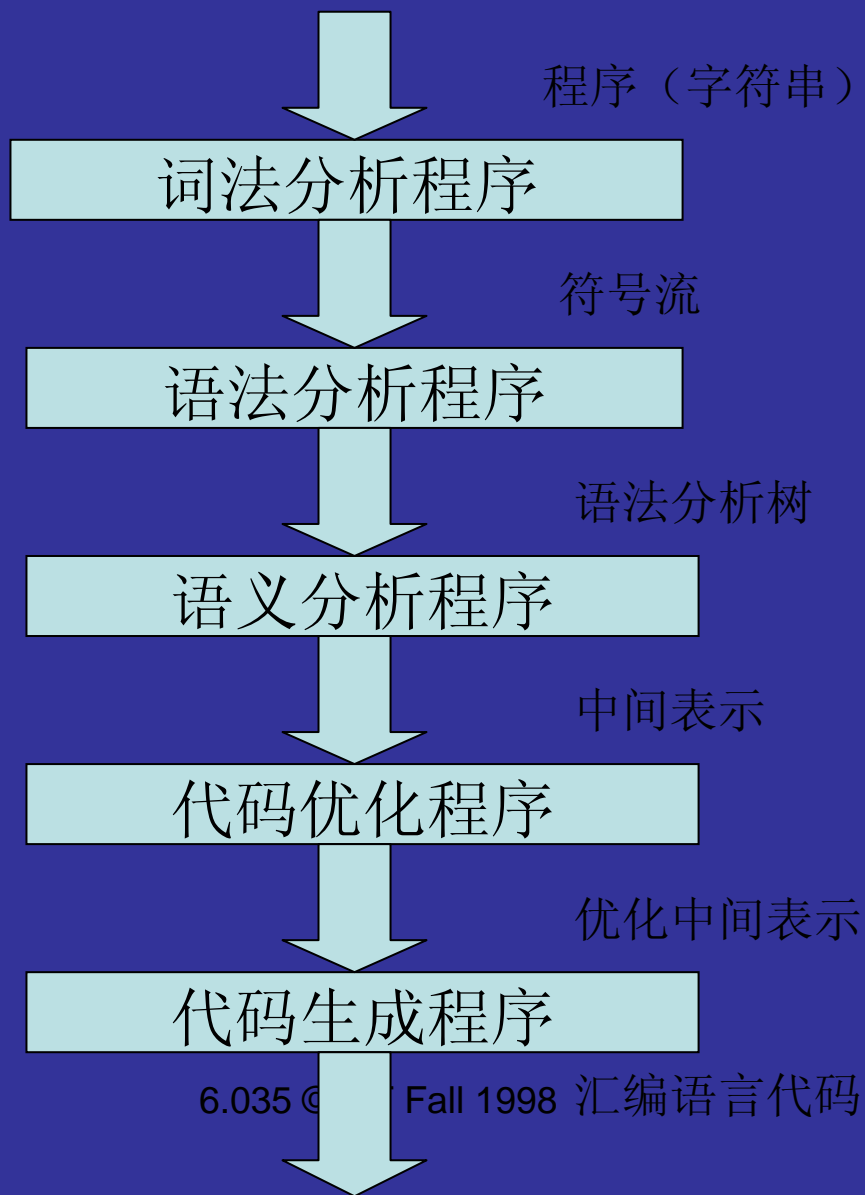
优化程序

```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,y;
    x=0;
    y=0;
    for(i=0;i<=N;i++){
        x=x+4*a/b*i+(i+1)*(i+1);
        x=x+b*y;
    }
    return x;
}
```



```
int sumcalc(int a,int b,int N)
{
    int i;
    int x,t,u,v;
    x=0;
    u= ((a<<2)/b);
    v=0;
    for(i=0;i<=N;i++){
        t=i+1;
        x=x+v+t*t;
        v=v+u;
    }
    return x;
}
```

计算机解析原理



代码生成器

```
int sumcalc(int a,int b,int N)
```

```
{  
    int i;  
    int x,t,u,v;  
    x=0;  
    u= ((a<<2)/b);  
    v=0;  
    for(i=0;i<=N;i++){  
        t=i+1;  
        x=x+v+t*t;  
        v=v+u;  
    }  
    return x;  
}
```



test:

```
subu $fp, 16  
add $t9, zero, zero  
sll $t0, $a0, 2  
div $t7, $t0, $a1  
add $t6, zero, zero  
add $t5, zero, zero
```

lab1:

```
addui $t8, $t5, 1  
mul $t0, $t8, $t8  
addu $t1, $t0, $t6  
addu $t9, $t9, $t1  
addu $6, $6, $7  
addui $t5, $t5, 1  
ble $t5, $a3, lab1  
  
addu $v0, $t9, zero  
addu $fp, 16  
b $ra
```