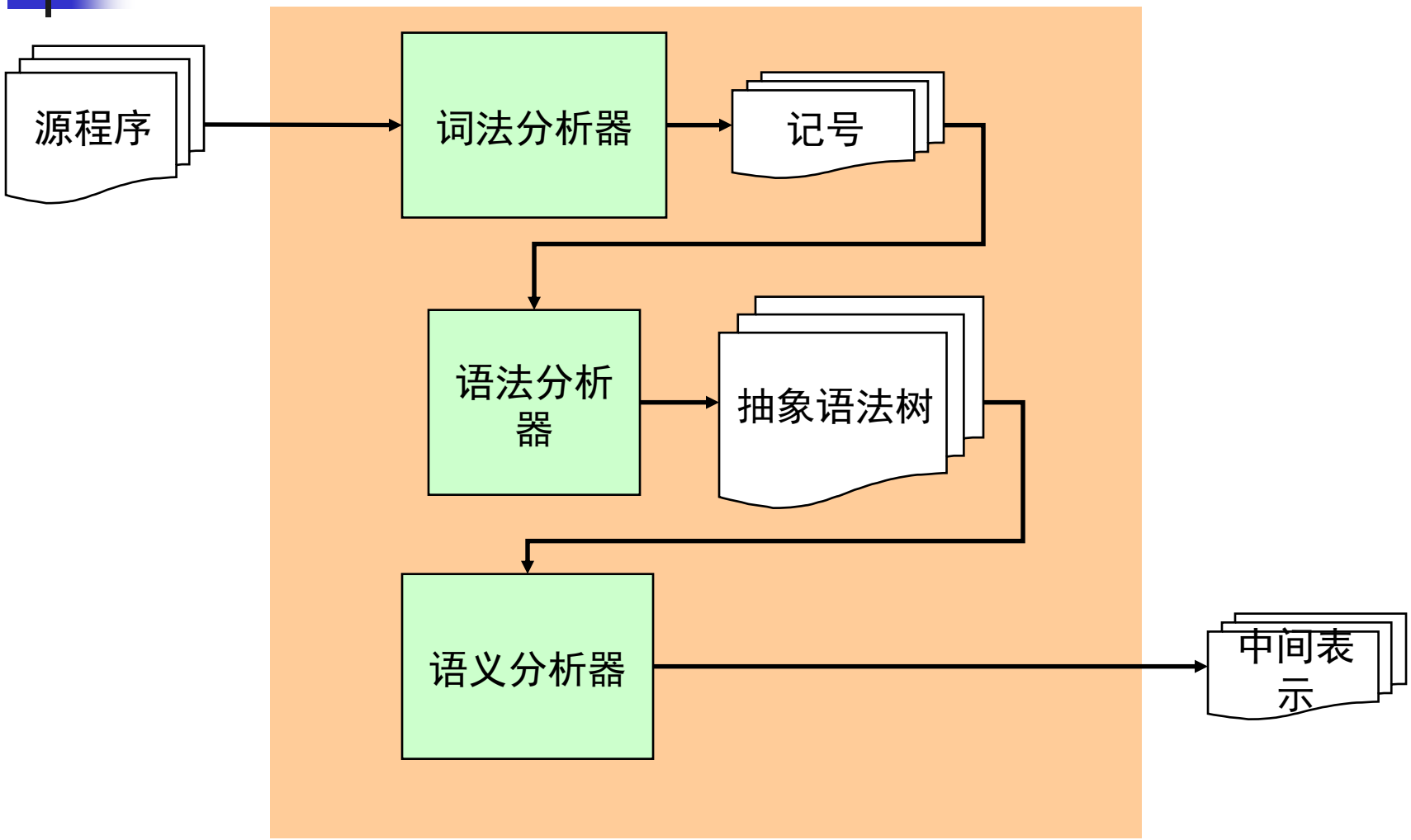
中间表示：三地址码

编译原理

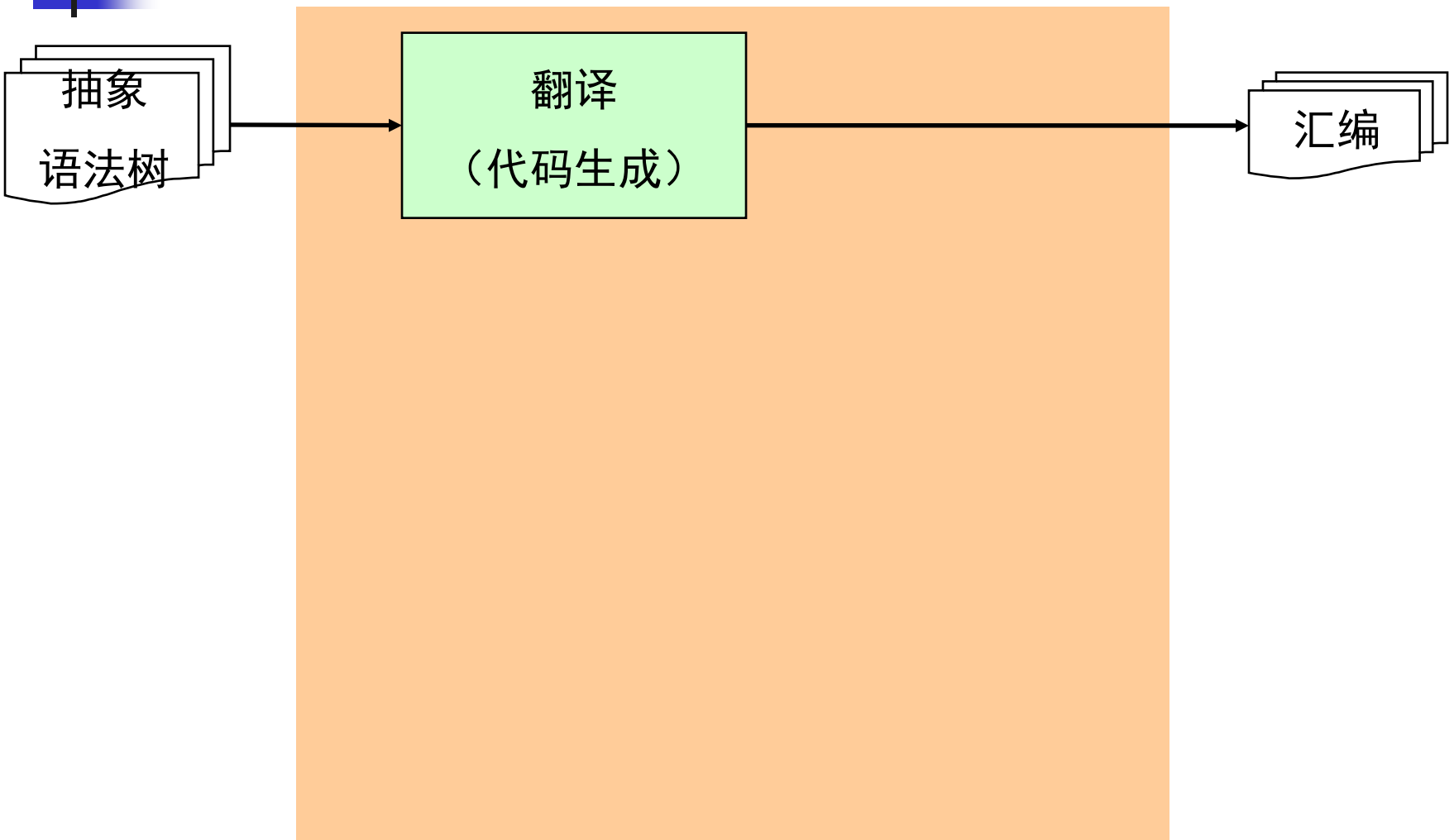
华保健

bjhua@ustc.edu.cn

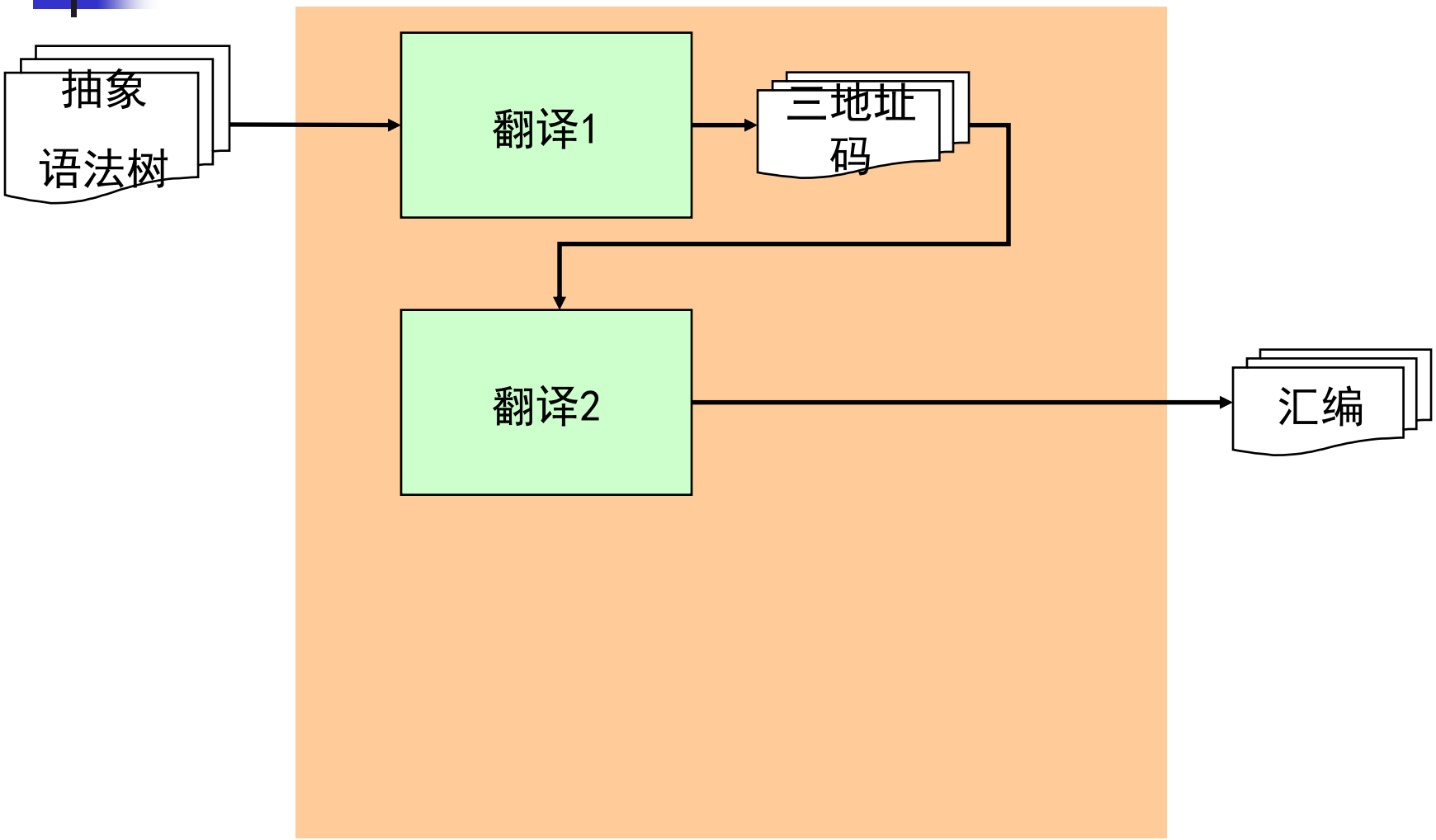
前端

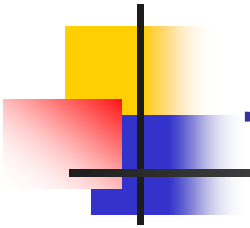


最简单的结构



使用三地址码的编译器结构





三地址码的基本思想

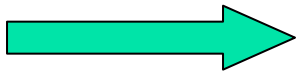
- 给每个中间变量和计算结果命名
  - 没有复合表达式
- 只有最基本的控制流
  - 没有各种控制结构
  - 只有goto，call等
- 所以三地址码可以看成是抽象的指令集
  - 通用的RISC

示例

```
a = 3 + 4 * 5;




if (x < y)
  z = 6;
else
  z = 7;
```



```
x_1 = 4;
x_2 = 5;
x_3 = x_1 * x_2;
x_4 = 3;
x_5 = x_4 + x_3;
a   = x_5;
Cjmp (x<y, L_1, L_2);
L_1:
   z = 6;
   jmp L_3;
L_2:
   z = 7;
   jmp L_3;
L_3:
   …
```

三地址码的定义

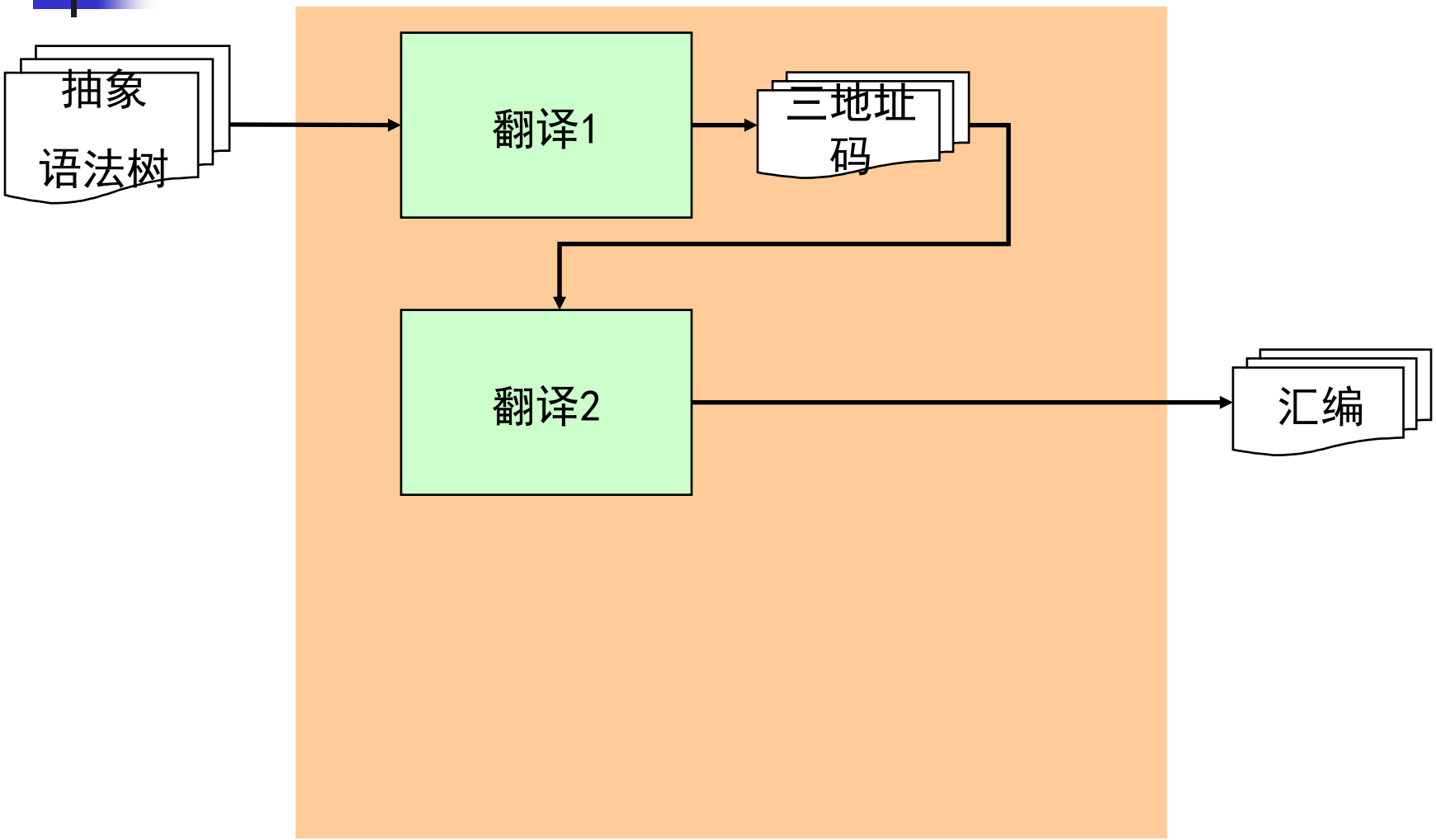
```
s -> x = n                          // 常数赋值
   | x = y ⊕ z                      // 二元运算
   | x = Θ y                        // 一元运算
   | x = y                          // 数据移动
   | x[y] = z                       // 内存写
   | x = y[v]                       // 内存读
   | x = f (x1, …, xn)              // 函数调用
   | Cjmp (x1, L1, L2)              // 条件跳转
   | Jmp L                          // 无条件跳转
   | Label L                        // 标号
   | Return x                       // 函数返回
```

如何定义三地址码数据结构？

```
enum instr_kind {INSTR_CONST, INSTR_MOVE, …};
struct Instr_t {enum instr_kind kind;};
struct Instr_Add {
  enum instr_kind kind;
  char *x;
  char *y;
  char *z;
};
struct Instr_Move {
  …;
};
```

其它的编码留作练习。

如何生成三地址码？



从C--生成三地址码

```
P -> F*
F -> x ((T id,)*) { (T id;)* S*}
T -> int
   | bool
S -> x = E
   | printi (E)
   | printb (E)
   | x (E1, …, En)
   | return E
   | if (E, S*, S*)
   | while (E, S*)
```

// 要写如下几个递归函数：

Gen_P(P); Gen_F(F); Gen_T(T);

Gen_S(S); Gen_E(E);

```
// 续：表达式的语法
E -> n
   | x
   | true
   | false
   | E + E
   | E && E
```

递归下降代码生成算法：语句的代码生成（I）

```
Gen_S(S s)
  switch (s)
    case x=e:
      x1 = Gen_E(e);
      emit(“x = x1”);
      break;
    case printi(e):
      x = Gen_E(e);
      emit (“printi(x)”);
      break;
    case printb(e):
      x = Gen_E(e);
      emit (“printb(x)”);
      break;
```

```
S -> x = E
   | printi (E)
   | printb (E)
   | x (E1, …, En)
   | return E
   | if (E, S*, S*)
   | while (E, S*)
```

递归下降代码生成算法：语句的代码生成（II）

```
case x(e1, …, en):
  x1 = Gen_E(e1);
  …;
  xn = Gen_E(en);
  emit(“x(x1, …, xn)”);
  break;
case return e;
  x = Gen_E(e);
  emit (“return x”);
  break;
```

```
S -> x = E
   | printi (E)
   | printb (E)
   | x (E1, …, En)
   | return E
   | if (E, S*, S*)
   | while (E, S*)
```

递归下降代码生成算法：语句的代码生成（III）

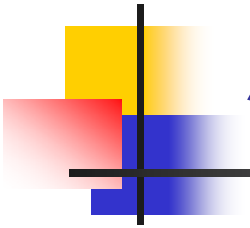
```
case if(e, s1, s2):
  x = Gen_E(e);
  emit (“Cjmp(x, L1, L2)”);
  emit (“Label L1:”);
  Gen_SList(s1);
  emit (“jmp L3”);
  emit (“Label L2:”);
  Gen_SList (s2);
  emit (“jmp L3”);
  emit (“Label L3:”);
  break;
```

```
S -> x = E
   | printi (E)
   | printb (E)
   | x (E1, …, En)
   | return E
   | if (E, S*, S*)
   | while (E, S*)
```

递归下降代码生成算法：语句的代码生成（IIII）

```
case while(e, s):
   emit (“Label L1:”);
   x = Gen_E(e);
   emit (“Cjmp(x, L2, L3)”);
   emit (“Label L2:”);
   Gen_SList(s);
   emit (“jmp L1”);
   emit (“Label L3:”);
   break;
```

```
S -> x = E
   | printi (E)
   | printb (E)
   | x (E1, …, En)
   | return E
   | if (E, S*, S*)
   | while (E, S*)
```



小结

- 三地址码的优点：
  - 所有的操作是原子的
    - 变量！没有复合结构
  - 控制流结构被简化了
    - 只有跳转
  - 是抽象的机器代码
    - 向后做代码生成更容易
- 三地址码的不足：
  - 程序的控制流信息是隐式的
  - 可以做进一步的控制流分析

从三地址码生成机器指令