



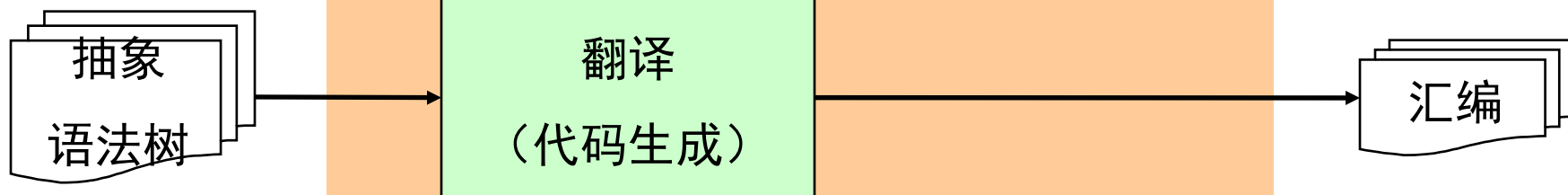
代码生成： 栈式计算机的代码生成

编译原理

华保健

bjhua@ustc.edu.cn

最简单的结构





递归下降代码生成算法： 从C--到Stack

```
P -> D S
D -> T id; D
    |
T -> int
    | bool
S -> id = E
    | printi (E)
    | printb (E)
E -> n
    | id
    | true
    | false
    | E + E
    | E && E
```

```
// 要写如下几个
// 递归函数：
Gen_P(D S);
Gen_D(T id; D);
Gen_T(T);
Gen_S(S);
Gen_E(E);
```

```
// 指令的语法
s -> push NUM
    | load x
    | store x
    | add
    | sub
    | times
    | div
```

递归下降代码生成算法： 表达式的代码生成

// 不变式：表达式的值总在栈顶

```
Gen_E(E e)
    switch (e)
        case n: emit ("push n"); break;
        case id: emit ("load id"); break;
        case true: emit ("push 1"); break;
        case false: emit ("push 0"); break;
        case e1+e2: Gen_E (e1);
                    Gen_E (e2);
                    emit ("add");
                    break;
        case ...: ... // similar
```

```
P -> D S
D -> T id; D
    |
T -> int
    | bool
S -> id = E
    | printi (E)
    | printb (E)
E -> n
    | id
    | true
    | false
    | E + E
    | E && E
```

递归下降代码生成算法： 语句的代码生成

// 不变式：栈的规模不变

```
Gen_S(S s)
    switch (s)
        case id=e: Gen_E(e);
                    emit("store id");
                    break;
        case printi(e): Gen_E(e);
                        emit ("printi");
                        break;
        case printb(e): Gen_E(e);
                        emit ("printb");
                        break;
```

```
P -> D S
D -> T id; D
    |
T -> int
    | bool
S -> id = E
    | printi (E)
    | printb (E)
E -> n
    | id
    | true
    | false
    | E + E
    | E && E
```

递归下降代码生成算法： 类型的代码生成

// 不变式：只生成.int类型

```
Gen_T(T t)
    switch (t)
        case int: emit (".int");
                  break;
        case bool: emit (".int");
                  break;
```

```
P -> D S
D -> T id; D
    |
T -> int
    | bool
S -> id = E
    | printi (E)
    | printb (E)
E -> n
    | id
    | true
    | false
    | E + E
    | E && E
```

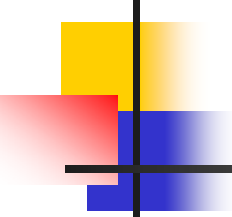
递归下降代码生成算法： 变量声明的代码生成

// 不变式：只生成.int类型

```
Gen_D(T id; D)
    Gen_T (T);
    emit (" id");
    Gen_D (D);
```

```
P -> D S
D -> T id; D
    |
T -> int
    | bool
S -> id = E
    | printi (E)
    | printb (E)
E -> n
    | id
    | true
    | false
    | E + E
    | E && E
```

递归下降代码生成算法： 程序的代码生成



```
Gen_P(D S)
  Gen_D (D);
  Gen_S (S);
```

```
P -> D S
D -> T id; D
    |
T -> int
    | bool
S -> id = E
    | printi (E)
    | printb (E)
E -> n
    | id
    | true
    | false
    | E + E
    | E && E
```




示例

```
int x;  
int y;  
int z;
```

```
x = 10;
```

```
y = 5;
```

```
z = x + y;
```

```
y = z * x;
```

```
.int x  
.int y  
.int z
```

```
1:  push 10
```

```
3:  store x
```

```
4:  push 5
```

```
5:  store y
```

```
6:  load x
```

```
7:  load y
```

```
8:  add
```

```
9:  store z
```

```
10: load z
```

```
11: load x
```

```
12: times
```

```
13: store y
```



如何运行生成的代码？

- 找一台真实的物理机
- 写一个虚拟机（解释器）
 - 类似于JVM
- 在非栈式计算机上进行模拟：
 - 例如，可以在x86上模拟栈式计算机的行为
 - 用x86的调用栈模拟栈