



# 中间代码生成

## Part4: 标号回填与控制流语句翻译

李 诚

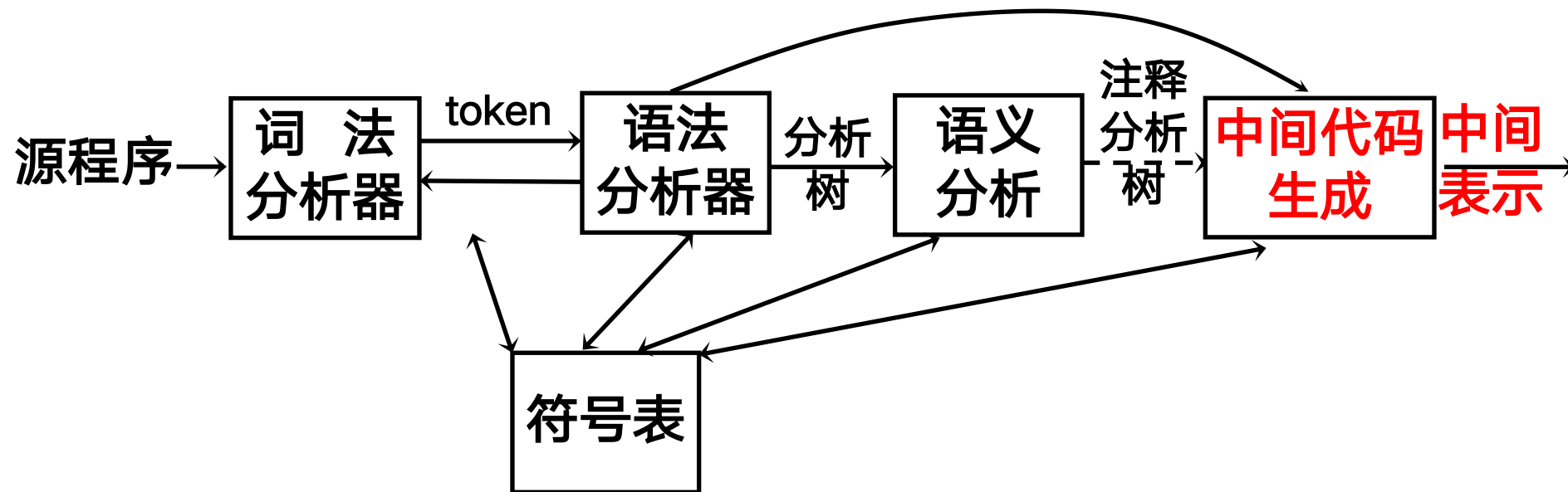
国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2025年04月03日



# 本节提纲



- 基于标号回填的其他控制流语句翻译
- 控制流语句翻译举例



- 对布尔表达式而言，有两个综合属性：
    - B.truelist：代码中所有转向真出口的代码指令链；
    - B.falselist：所有转向假出口的代码指令链；
    - 在生成B的代码时，跳转指令goto是不完整的，目标标号尚未填写，用truelist和falselist来管理
  - 对一般语句而言，有一个综合属性：
    - S.nextlist：代码中所有跳转到紧跟S的代码之后的指令
- 例如：
- $S \rightarrow \{L\}$  //程序块
  - $S \rightarrow A$  //赋值语句
  - $S \rightarrow \text{if } B \text{ then } S_1 \text{ else } S_2$



- 用**S**和**L**分别表示**一条语句**和**语句列表**

$S \rightarrow \text{if } B \text{ then } S_1 // (B) \text{ 的括号此处省略}$

$S \rightarrow \text{if } B \text{ then } S_1 \text{ else } S_2$

$S \rightarrow \text{while } B \text{ do } S_1$

$S \rightarrow A // \text{赋值语句}$

$S \rightarrow \{L\} // \text{大括号的作用是把内部的多个语句绑在一起, 当成一个语句。}$

$L \rightarrow L;S \mid S // \text{语句列表, } L \text{ 和 } S \text{ 之间用}; \text{分割}$



考虑以下语句序列：

**if (  $a < b$  or  $c < d$  and  $e < f$  ) then**

**while (  $a > c$  ) do  $c := c + 1$**

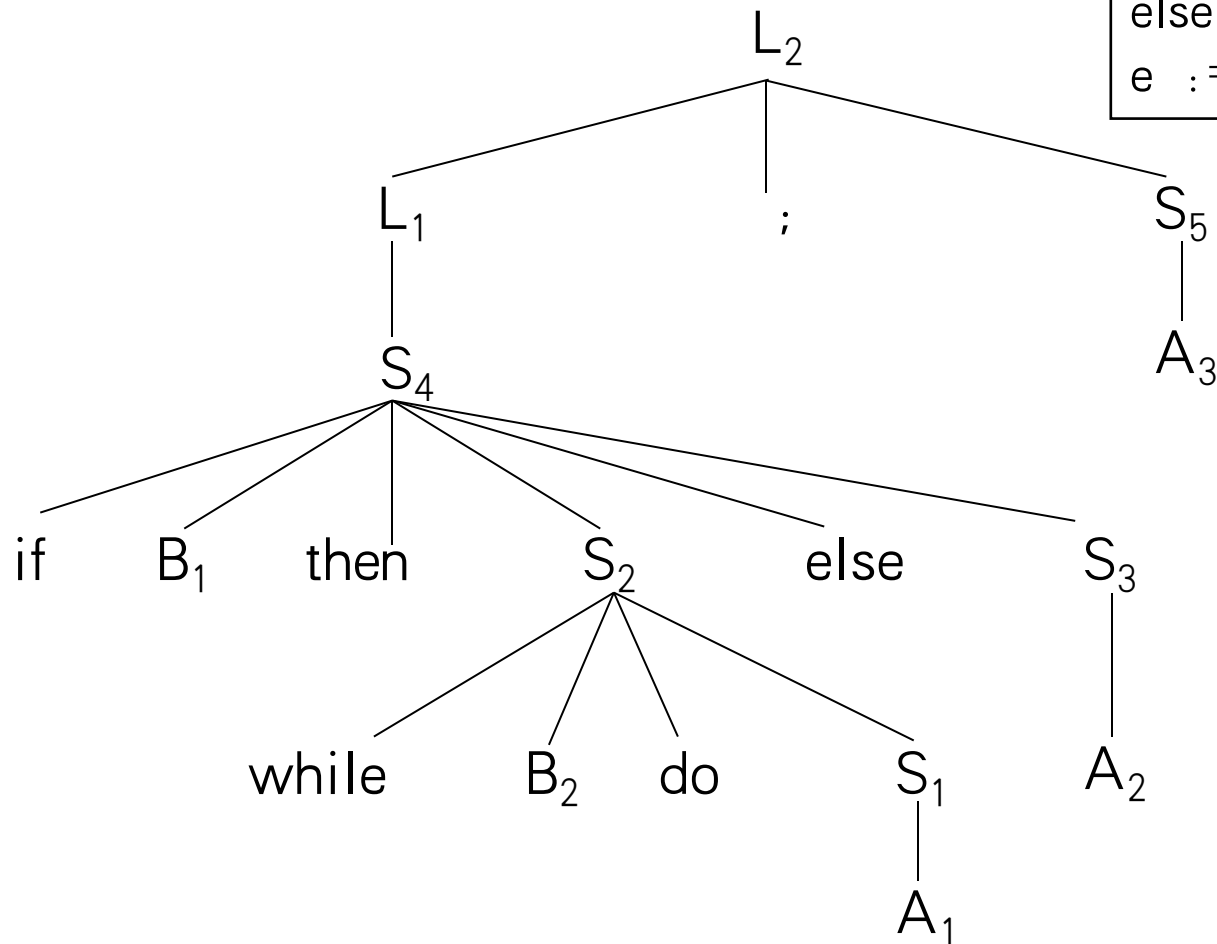
**else  $d := d + 1$ ;**

**$e := e + d$ ;**



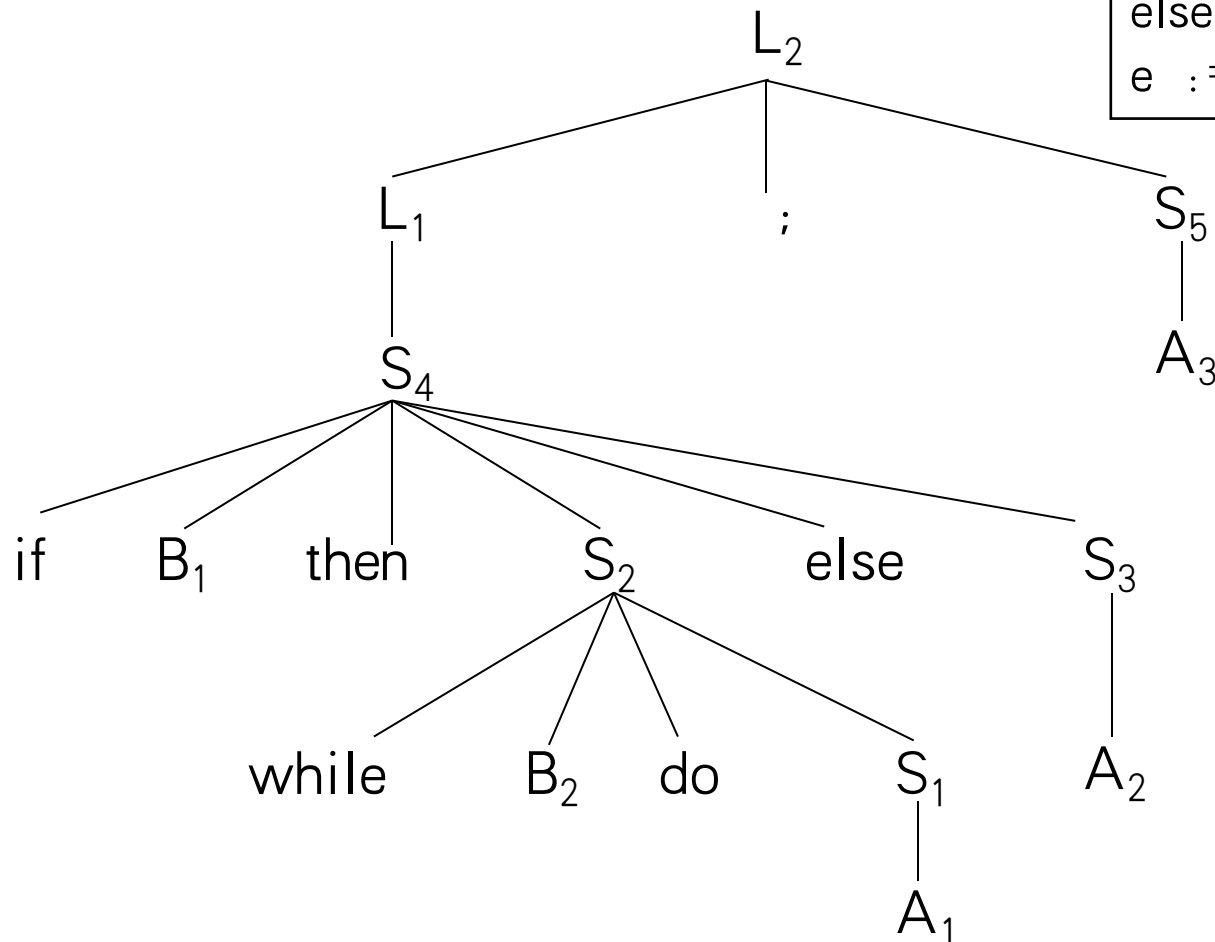
## • 分析树

```
if ( a < b or c < d and e < f ) then  
    while ( a > c ) do c := c + 1  
else d := d + 1;  
e := e + d;
```





## • 分析树



```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c + 1
else d := d + 1;
e := e + d;
```

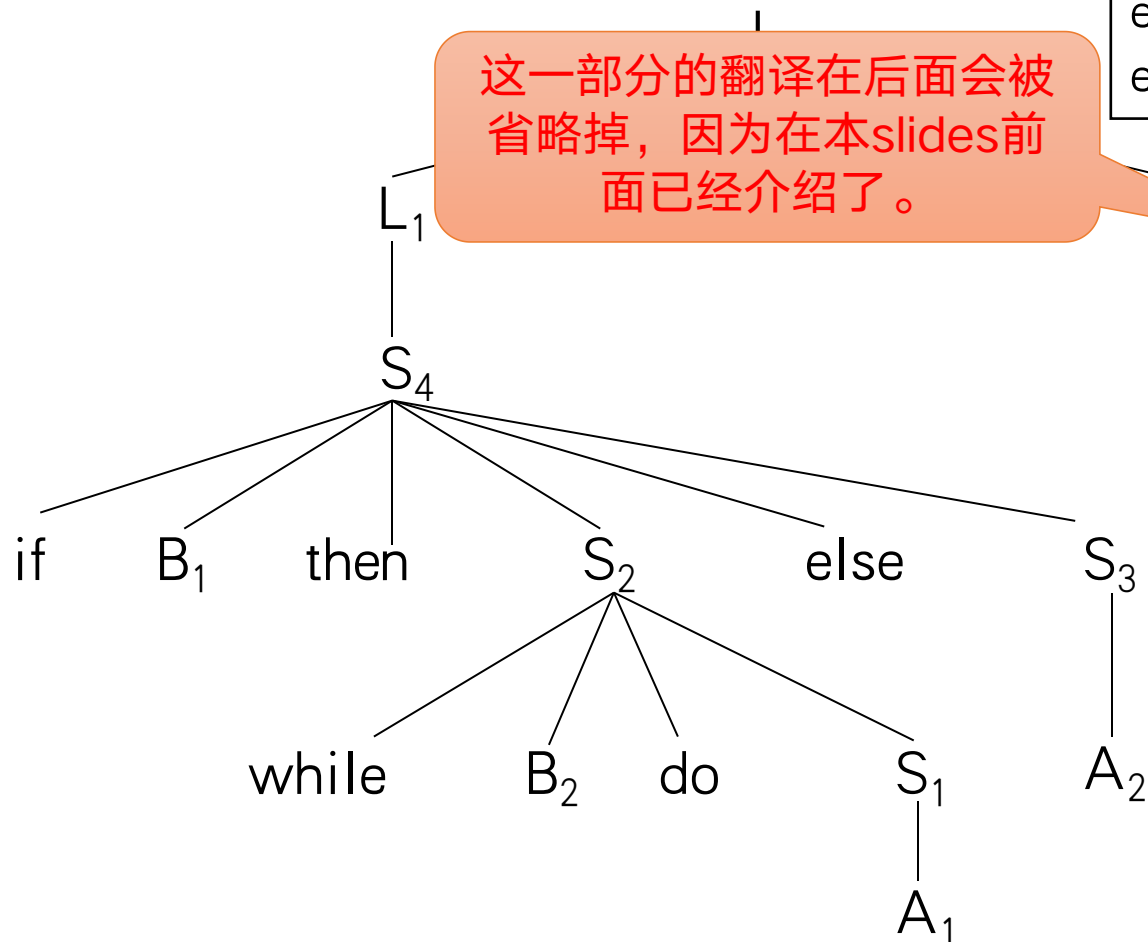
当前的任务:

- 为每一条语句或表达式生成对应的三地址代码
- 为 $B_1$ 和 $B_2$ 创建分别指向真出口和假出口的truelist和 falselist
- 为五个S语句和两个L语句列表创建指向下一指令的 nextlist
- 在特定的归约环节, 将跳转目标指令标号回填对应的列表中的待填指令

按序生成三地址代码



## • 分析树



这一部分的翻译在后面会被省略掉，因为在本slides前面已经介绍了。

```
if ( a < b or c < d and e < f ) then
    while ( a > c ) do c := c + 1
else d := d + 1;
e := e + d;
```

当前的任务:

- 为每一条语句或表达式生成对应的三地址代码
- 为  $B_1$  和  $B_2$  创建分别指向真出口和假出口的 truelist 和 falselist
- 为五个 S 语句和两个 L 语句列表创建指向下一指令的 nextlist
- 在特定的归约环节，将跳转目标指令标号回填对应的列表中的待填指令

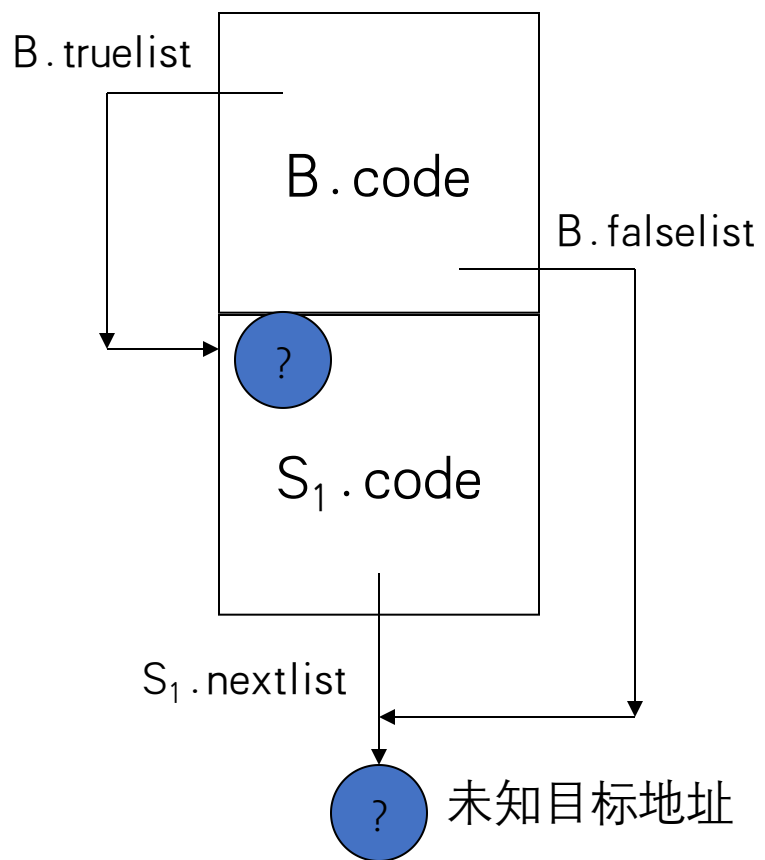




# 条件语句的翻译方案 (1)



if B then  $S_1$  的代码结构



## • 四个跳转目标未知

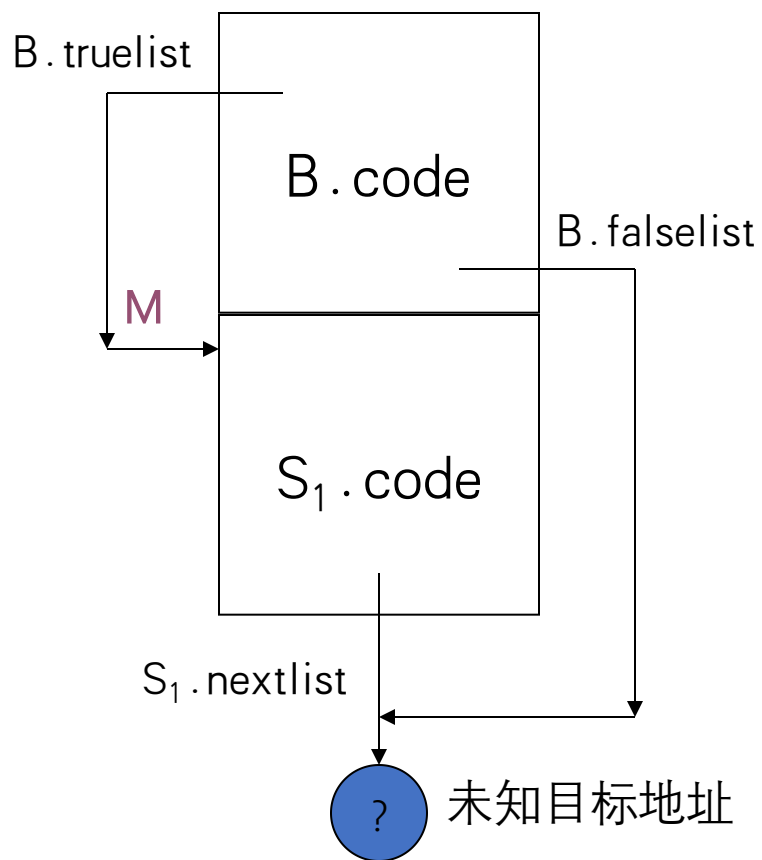
- B的值为真跳转到 $S_1$ 的开始
- B的值为假/ $S_1$ 结束/S结束应该跳转到同一个指令



# 条件语句的翻译方案 (1)



if B then  $S_1$  的代码结构



## • 四个跳转目标未知

- B的值为真跳转到 $S_1$ 的开始
- B的值为假/ $S_1$ 结束/S结束应该跳转到同一个指令

## • 解决方案

- 引入M标记, 记录B.code之后的下一条新的指令标号, 方便回填B.truelist
- 将B.falselist、 $S_1$ .nextlist合并赋给S.nextlist



# 条件语句的翻译方案 (1)



**$S \rightarrow \text{if } B \text{ then } M \ S_1$**

{

**backpatch( B.truelist,  $M.instr$  );**

**$S.nextlist = \text{merge}( B.falselist, S_1.nextlist )$**

}

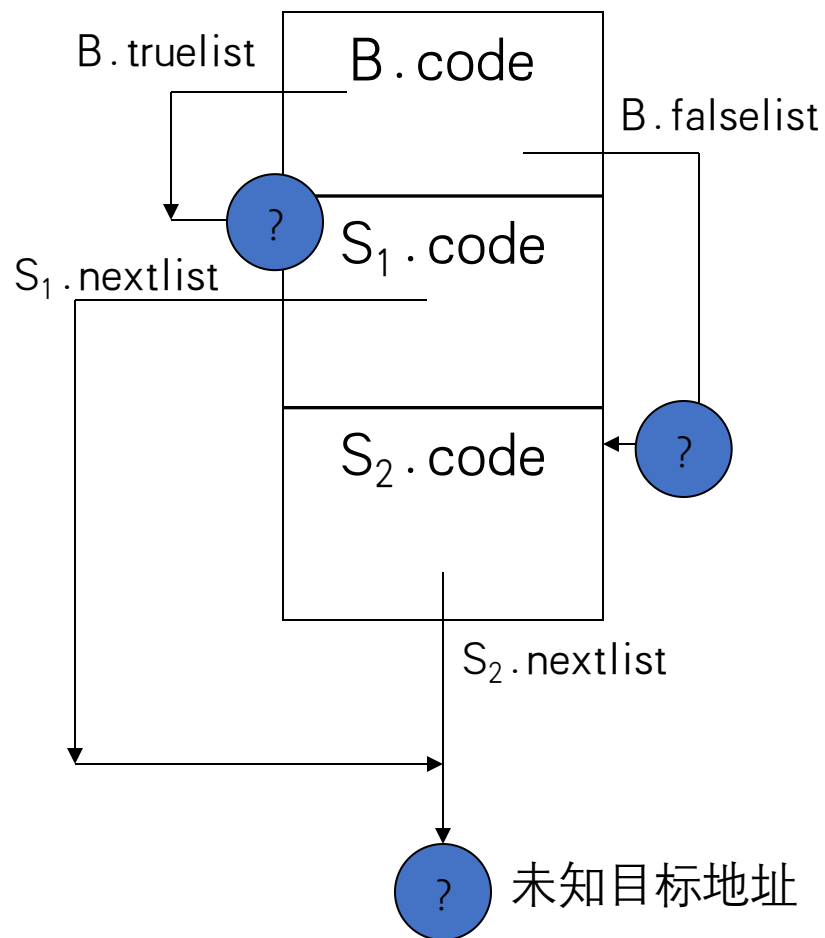
**$M \rightarrow \epsilon \ \{ \ M.instr = nextinstr \}$**



# 条件语句的翻译方案 (2)



if B then  $S_1$  else  $S_2$ 的结构



## • 五个跳转目标未知

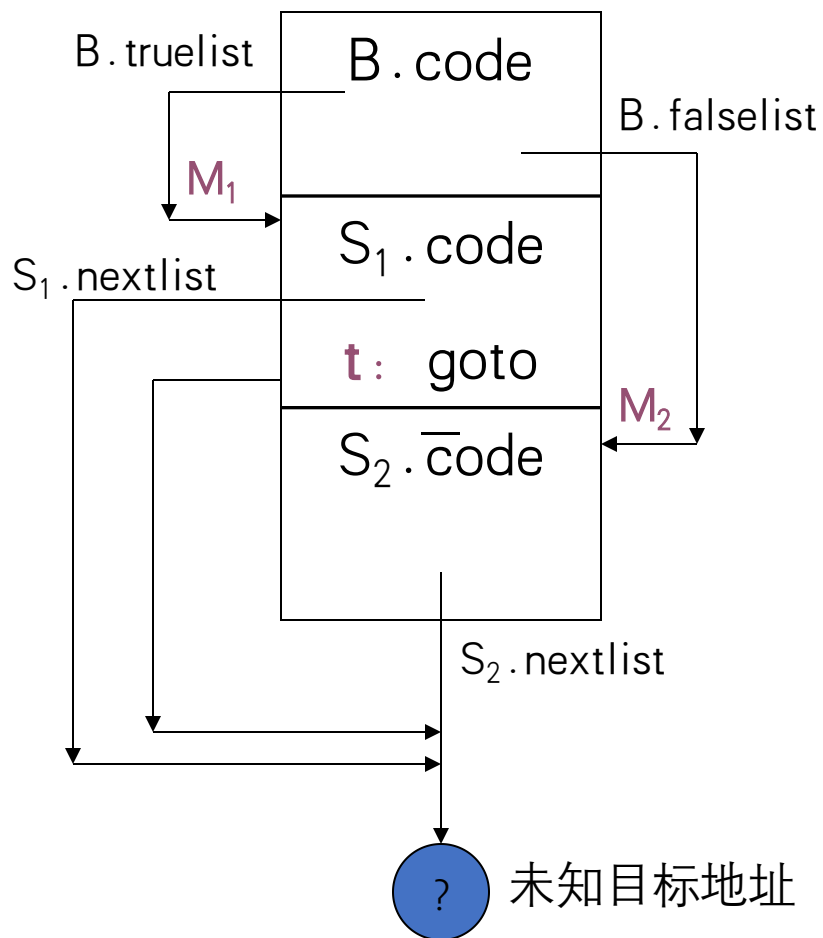
- B的值为真跳转到 $S_1$ 的开始
  - B的值为假跳转到 $S_2$ 的开始
  - $S_1$  /  $S_2$  / S结束应该跳转到同一个指令
- ## • $S_1$ 结束要越过 $S_2$



# 条件语句的翻译方案 (2)



if B then  $S_1$  else  $S_2$ 的结构



## • 五个跳转目标未知

- B的值为真跳转到S<sub>1</sub>的开始
- B的值为假跳转到S<sub>2</sub>的开始
- S<sub>1</sub>/ S<sub>2</sub> /S结束应该跳转到同一个指令

## • S<sub>1</sub>结束要越过S<sub>2</sub>

## • 解决方案

- 引入M<sub>1</sub>和M<sub>2</sub>标记
- 引入N标记, 在S<sub>1</sub>后插入无条件跳转指令
- 将S<sub>1</sub>.nextlist、{t}和S<sub>2</sub>.nextlist合并赋给S.nextlist



## 条件语句的翻译方案 (2)



$S \rightarrow \text{if } B \text{ then } \mathbf{M_1} \ S_1 \ \mathbf{N} \ \text{else } \mathbf{M_2} \ S_2$   
{

}

$N \rightarrow \epsilon$  {  $N.\text{nextlist} = \text{makelist}(\text{nextinstr});$  //标号t  
     $\text{gen}(\text{"goto"} - );$   
}



## 条件语句的翻译方案 (2)



```
S → if B then M1 S1 N else M2 S2  
    { backpatch( B.truelist, M1.instr );  
      backpatch( B.falselist, M2.instr );  
      temp = merge(S1.nextlist, N.nextlist);  
      S.nextlist = merge(temp, S2.nextlist) ;  
    }
```

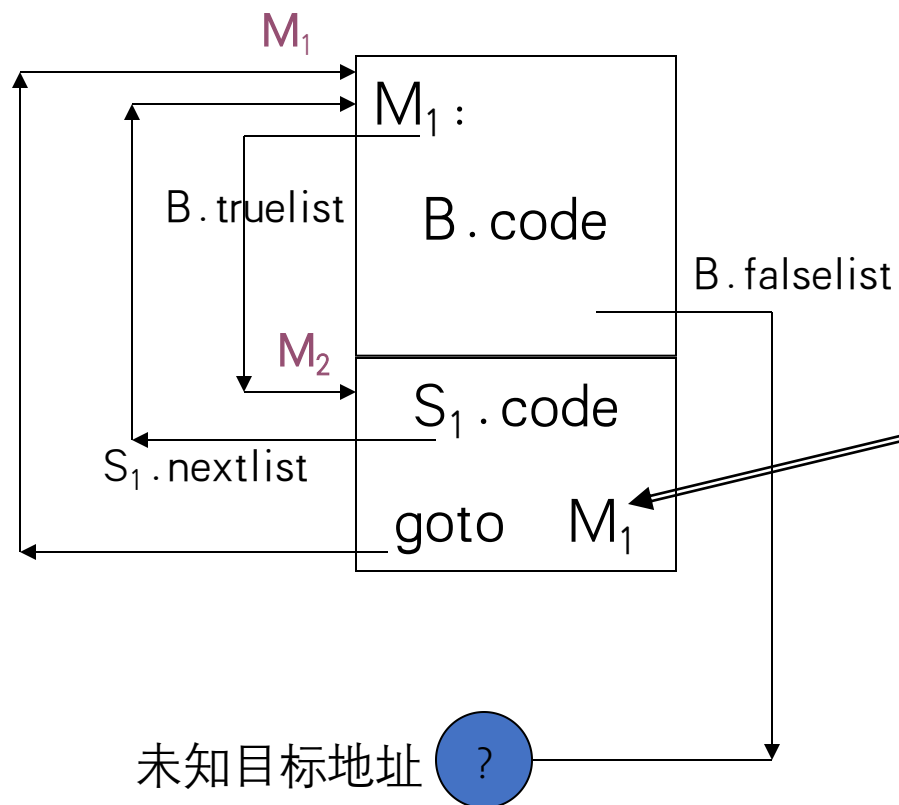
```
N → ε { N.nextlist = makelist(nextinstr); //标号t  
        gen( “goto” - );  
      }
```



# 循环语句的翻译方案



while B do  $S_1$  的代码结构



产生无条件跳转指令

goto  $M_1$  (跳转至循环条件测试代码开始处)

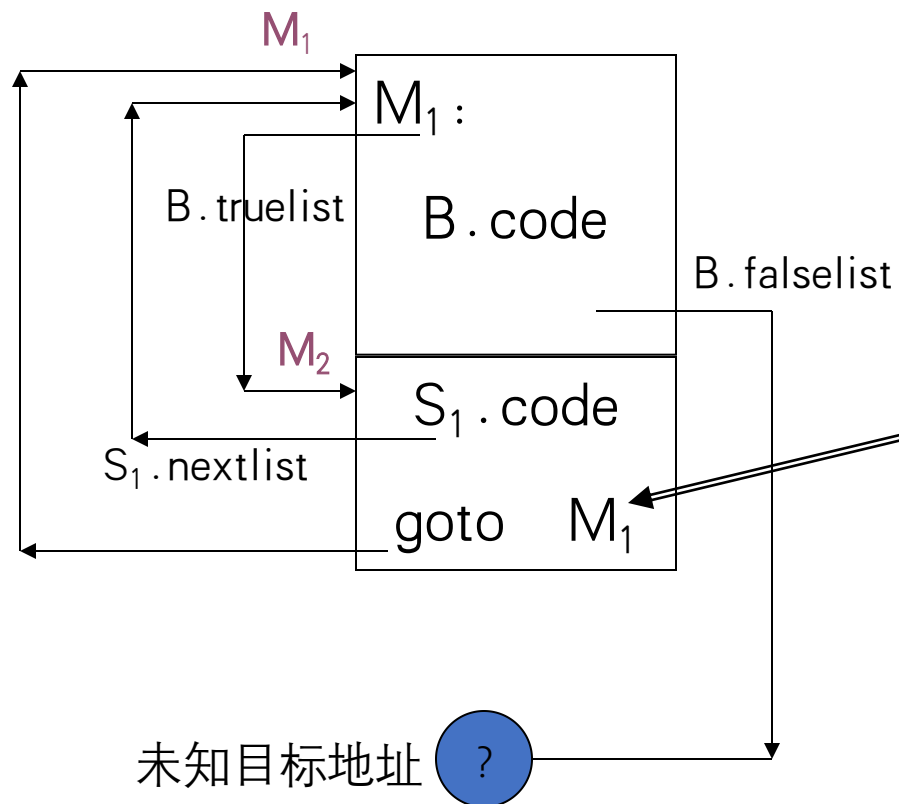




# 循环语句的翻译方案



while B do  $S_1$  的代码结构



与 if B then  $M_1$   $S_1$  N else  $M_2$   $S_2$  不同，此处不用引入N来生成一条无条件跳转指令，因为， $S_1$ 分析后已经看到句柄，因此，该指令的生成可以放到归约里。

产生无条件跳转指令

goto  $M_1$  (跳转至循环条件测试代码开始处)



# 循环语句的翻译方案



```
S → while  $M_1$  B do  $M_2$   $S_1$   
    { backpatch( B.truelist,  $M_2.instr$  );  
      backpatch(  $S_1.nextlist$ ,  $M_1.instr$  );  
       $S.nextlist = B.falselist$ ;  
      gen( “goto”  $M_1.instr$  ); // 已知  
    }
```

```
 $M \rightarrow \epsilon$  {  $M.instr = nextinstr$  }
```



# 简单语句的翻译方案


$$S \rightarrow A \{S.nextlist = \{\};\}$$
$$S \rightarrow \{L\} \{S.nextlist = L.nextlist\}$$
$$L \rightarrow S \{L.nextlist = S.nextlist\}$$



# 语句列表的翻译方案



$L \rightarrow L_1; S$

当分析完 $L_1$ 时，由于不知道 $S$ 是否会出现，因此需要引入 $M$ 标记符，记录 $S$ 的第一条指令；

当 $S$ 分析完后，进行 $L$ 的归约，此时，将 $M$ 记录的指令标号回填 $L_1.nextlist$ ，并将 $S$ 的 $nextlist$ 赋给 $L$ 的 $nextlist$



# 语句列表的翻译方案



$L \rightarrow L_1; S$

当分析完 $L_1$ 时，由于不知道 $S$ 是否会出现，因此需要引入 $M$ 标记符，记录 $S$ 的第一条指令；

当 $S$ 分析完后，进行 $L$ 的归约，此时，将 $M$ 记录的指令标号回填 $L_1.nextlist$ ，并将 $S$ 的 $nextlist$ 赋给 $L$ 的 $nextlist$

$L \rightarrow L_1; M S \{$

$backpatch( L_1.nextlist, M.instr );$

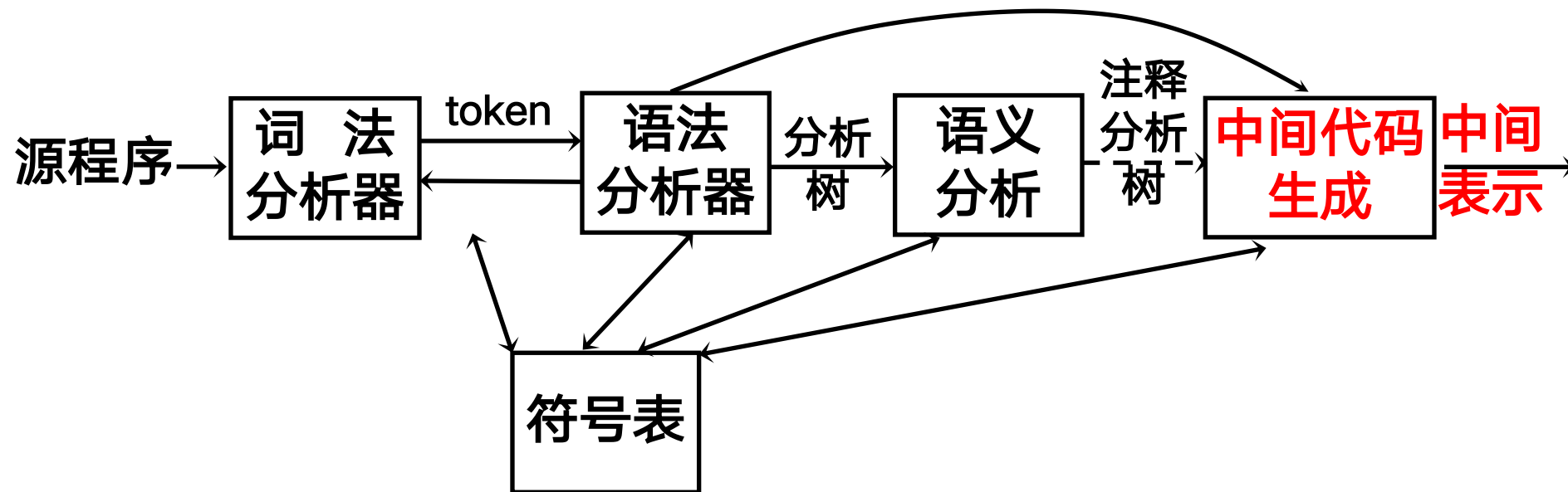
$L.nextlist = S.nextlist;$

$\}$

$M \rightarrow \epsilon \quad \{ M.instr = nextinstr \}$



# 本节提纲



- 基于标号回填的其他控制流语句翻译
- 控制流语句翻译举例



翻译以下语句序列：

**if (  $a < b$  or  $c < d$  and  $e < f$  ) then**

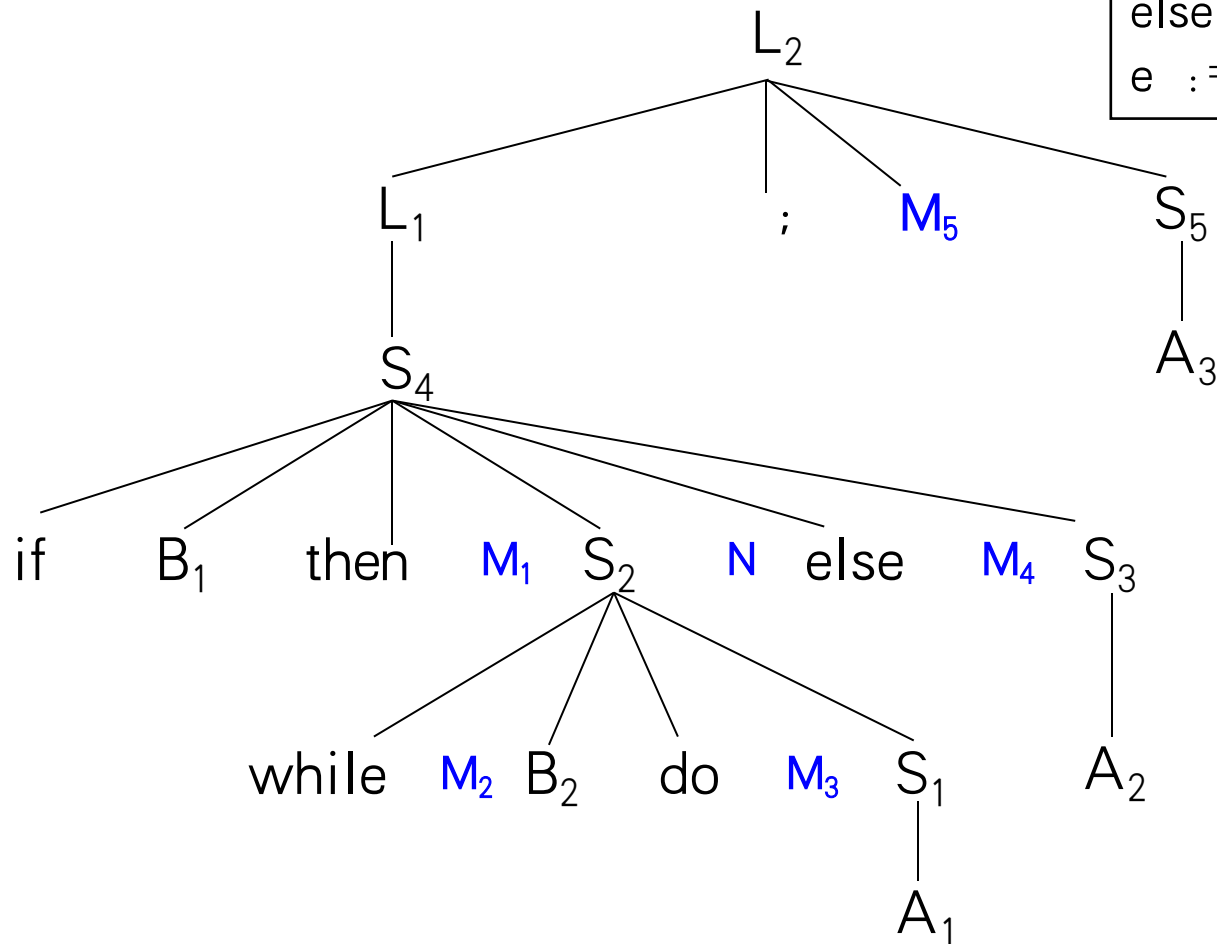
**while (  $a > c$  ) do  $c := c + 1$**

**else  $d := d + 1$ ;**

**$e := e + d$ ;**



## • 分析树

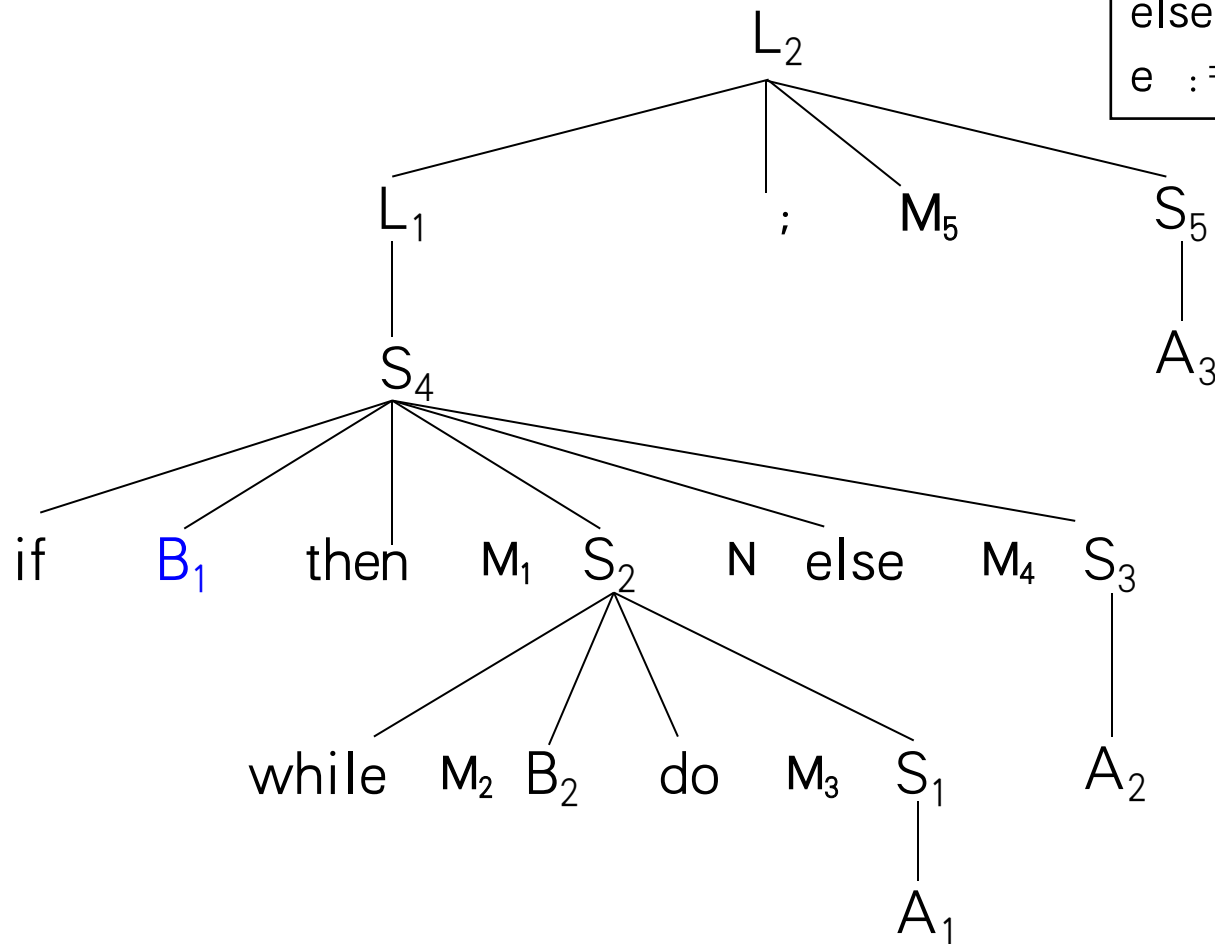


```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c + 1
else d := d + 1;
e := e + d;
```





## • 分析树



```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c + 1
else d := d + 1;
e := e + d;
```

蓝色表示即将翻译  
红色表示需要回填



一、翻译  $B_1$ : (  $a < b$  or  $c < d$  and  $e < f$  )

(100) if  $a < b$  goto -

(101) goto 102

(102) if  $c < d$  goto 104

(103) goto -

(104) if  $e < f$  goto -

(105) goto -

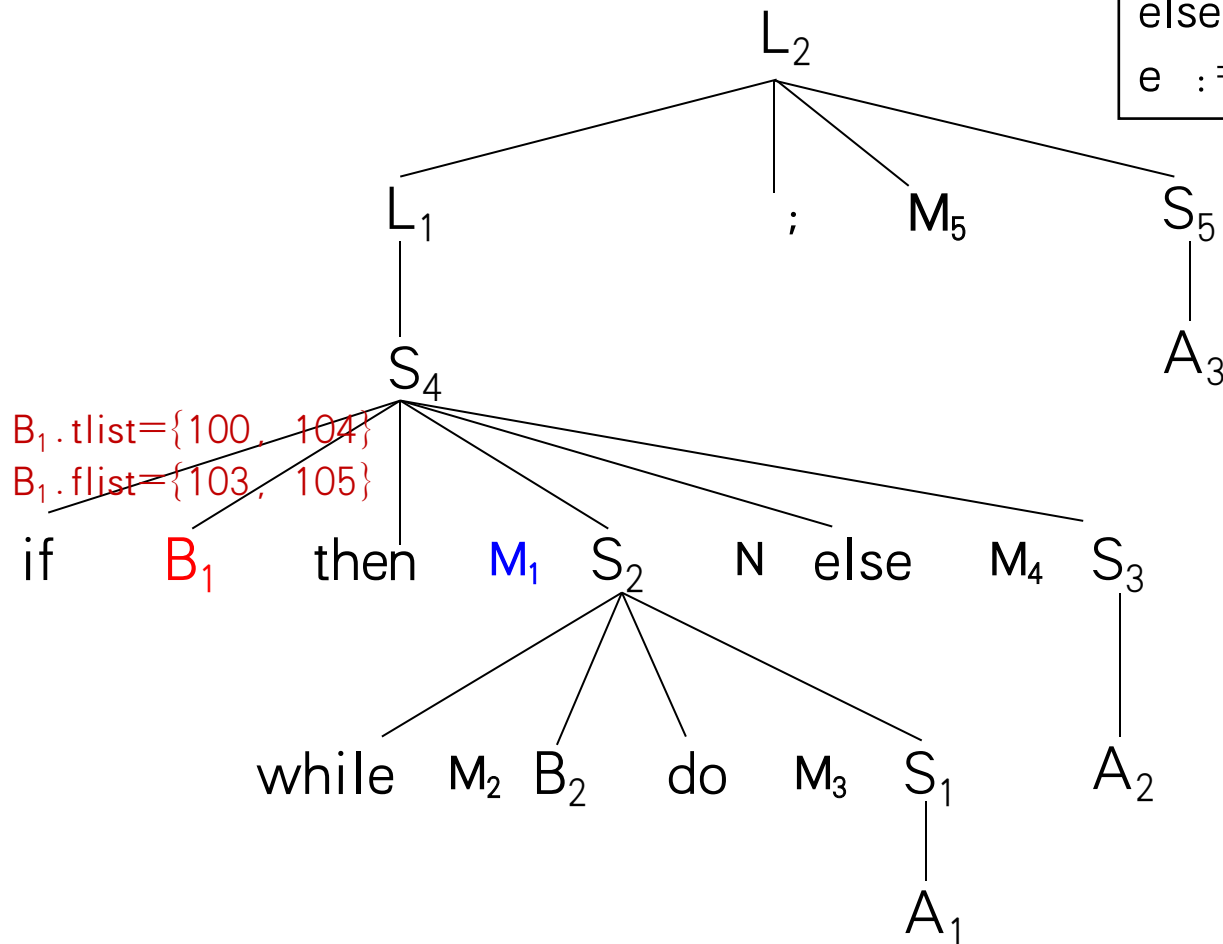
truelist: { 100, 104 } falselist: { 103, 105 }

在Lecture 15的ppt  
中已经进行了翻译，  
此处直接使用结果



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c + 1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填

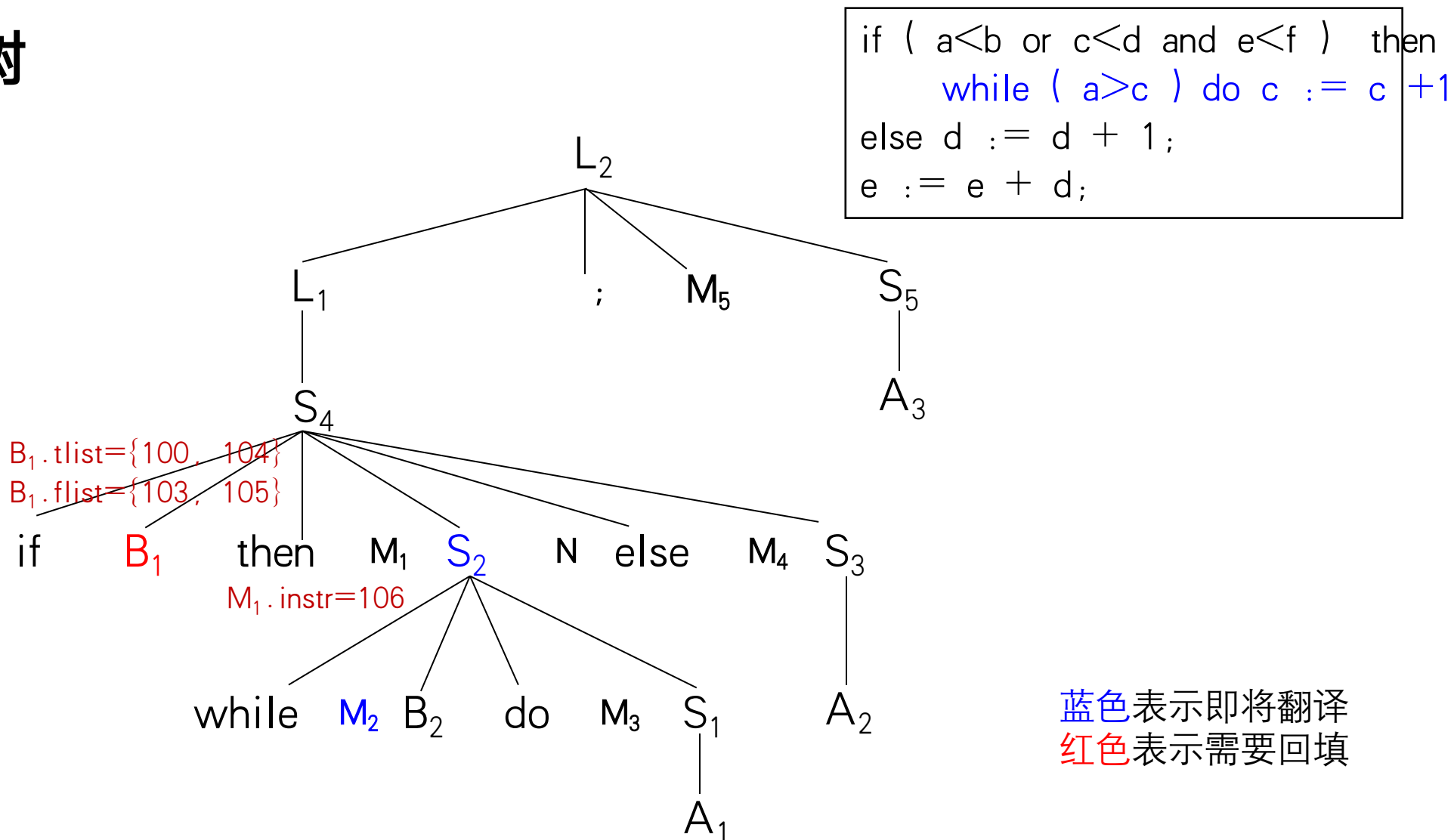


## 二、翻译 $M_1$ : $M_1 \rightarrow \varepsilon$

记录下一个指令标号106, 当if-then-else归约时, 用106回填 $B_1$ 的  
truelist { 100, 104 }



## • 分析树



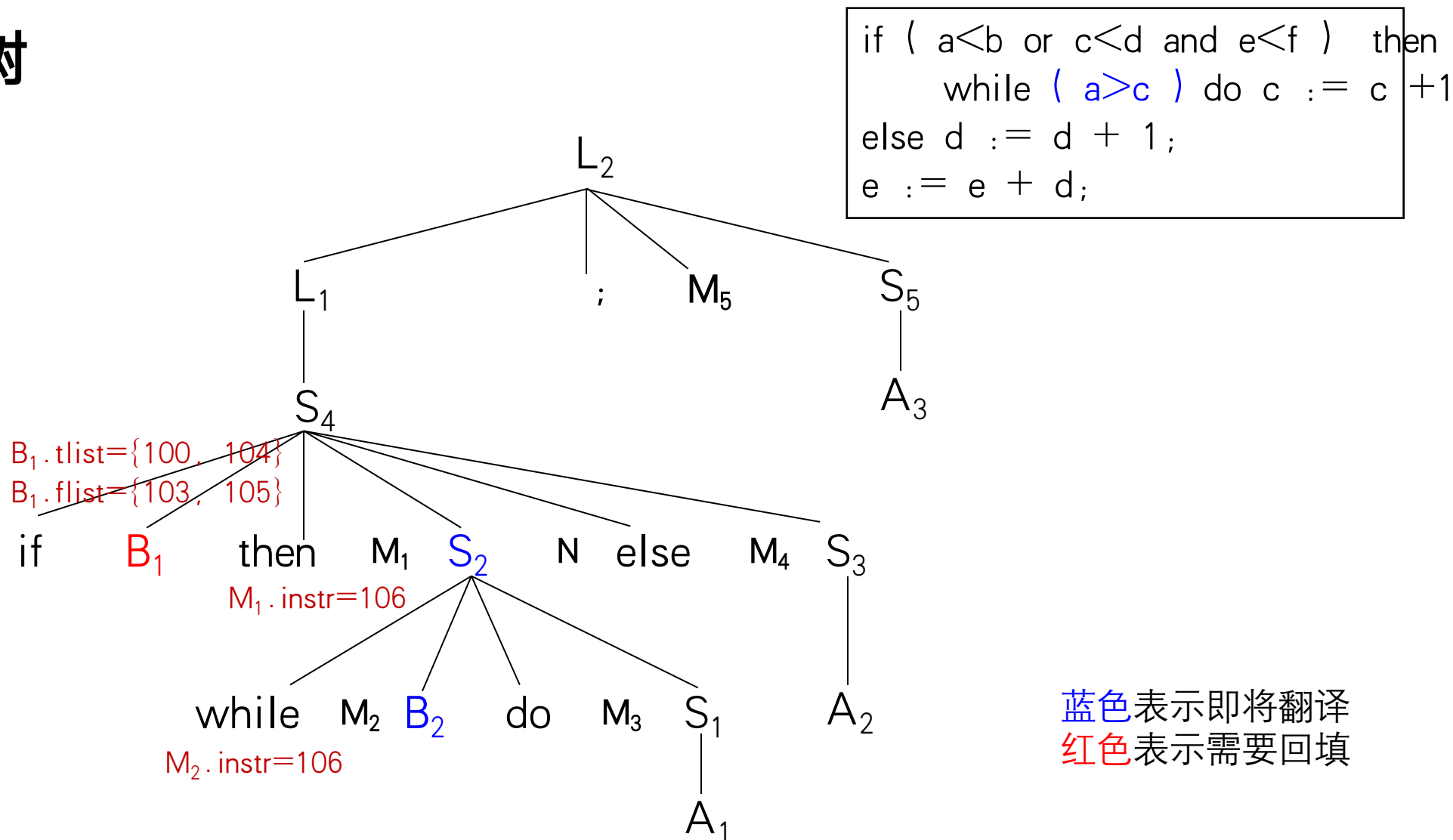


## 三、翻译 $M_2$ : $M_2 \rightarrow \varepsilon$

记录下一个指令标号106, 当while( $B_2$ )  $S_1$ 归约时, 用106回填 $S_1$ 的nextlist, 并生成无条件跳转指令goto  $M_2.instr$



## • 分析树



蓝色表示即将翻译  
红色表示需要回填



四、翻译 $S_2$ 中 $B_2$ : while  $B_2$  do  $S_1$

(106) if  $a > c$  goto -

(107) goto -

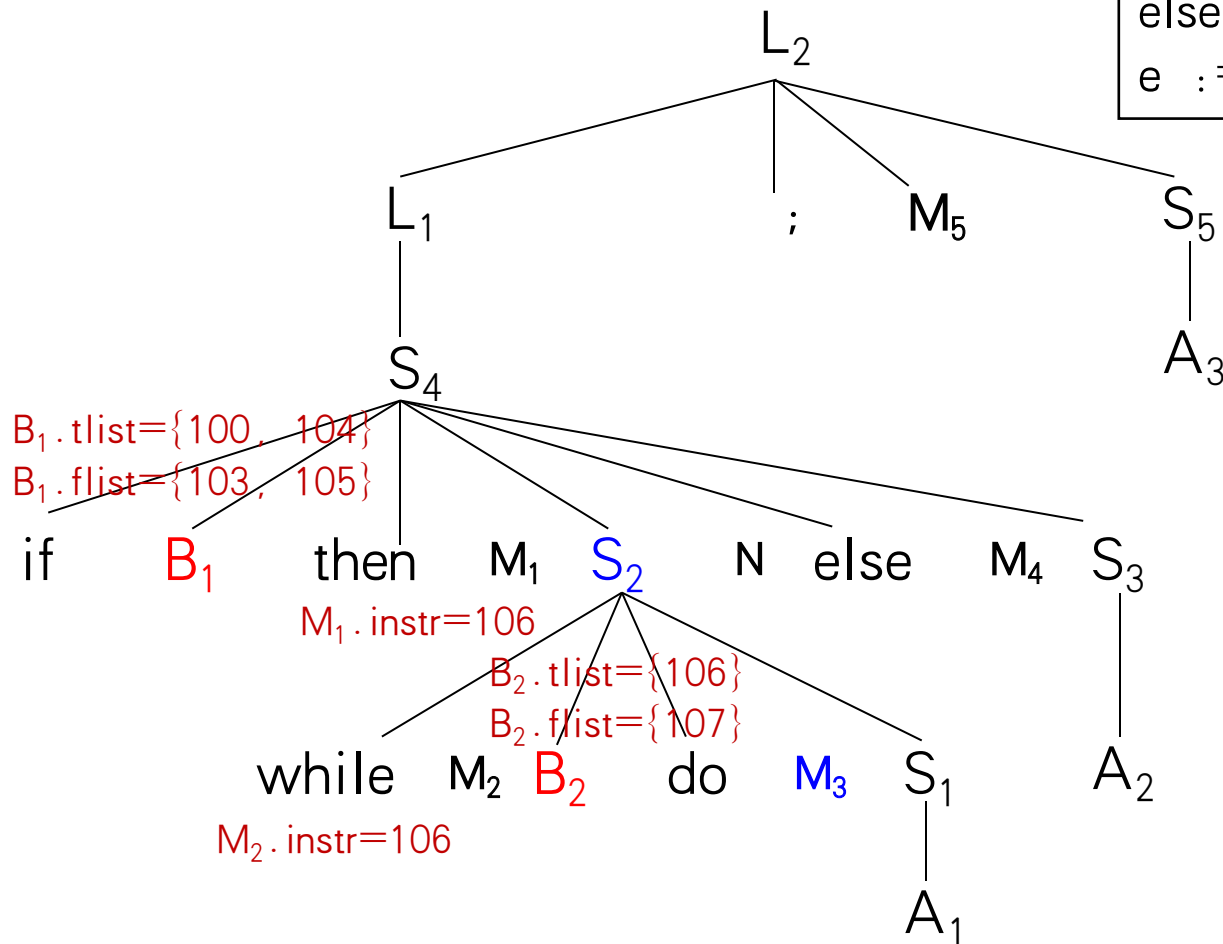
truelist: { 106 } falselist: { 107 }//待回填





## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c + 1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填

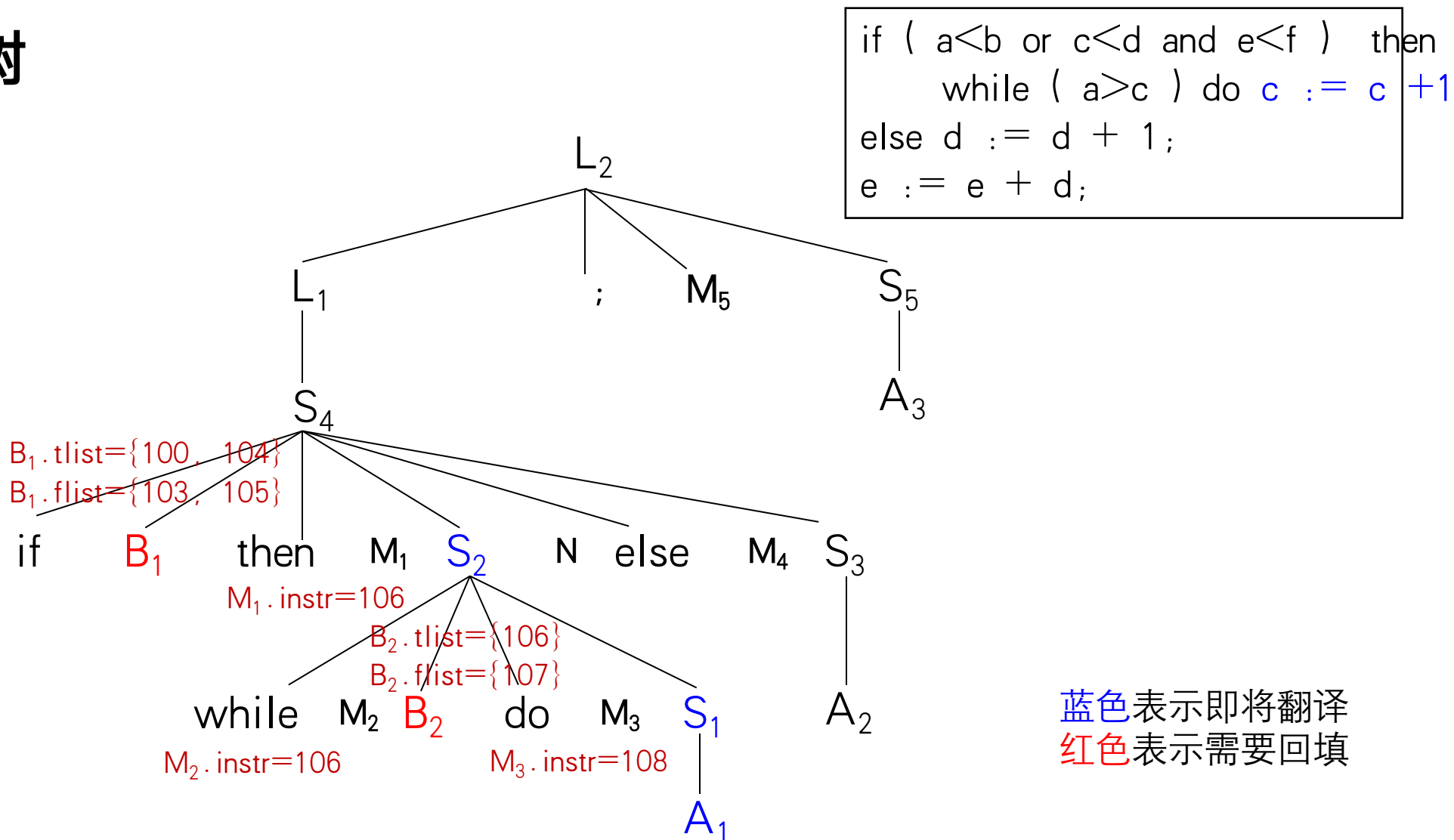


五、翻译  $M_3$ :  $M_3 \rightarrow \varepsilon$

记录下一个指令标号108, 当while( $B_2$ )  $S_1$ 归约时, 用108回填 $B_2$ 的  
truelist



## • 分析树



蓝色表示即将翻译  
红色表示需要回填



六、翻译  $S_1$ : while  $B_2$  do  $S_1$

(106) if  $a > c$  goto -

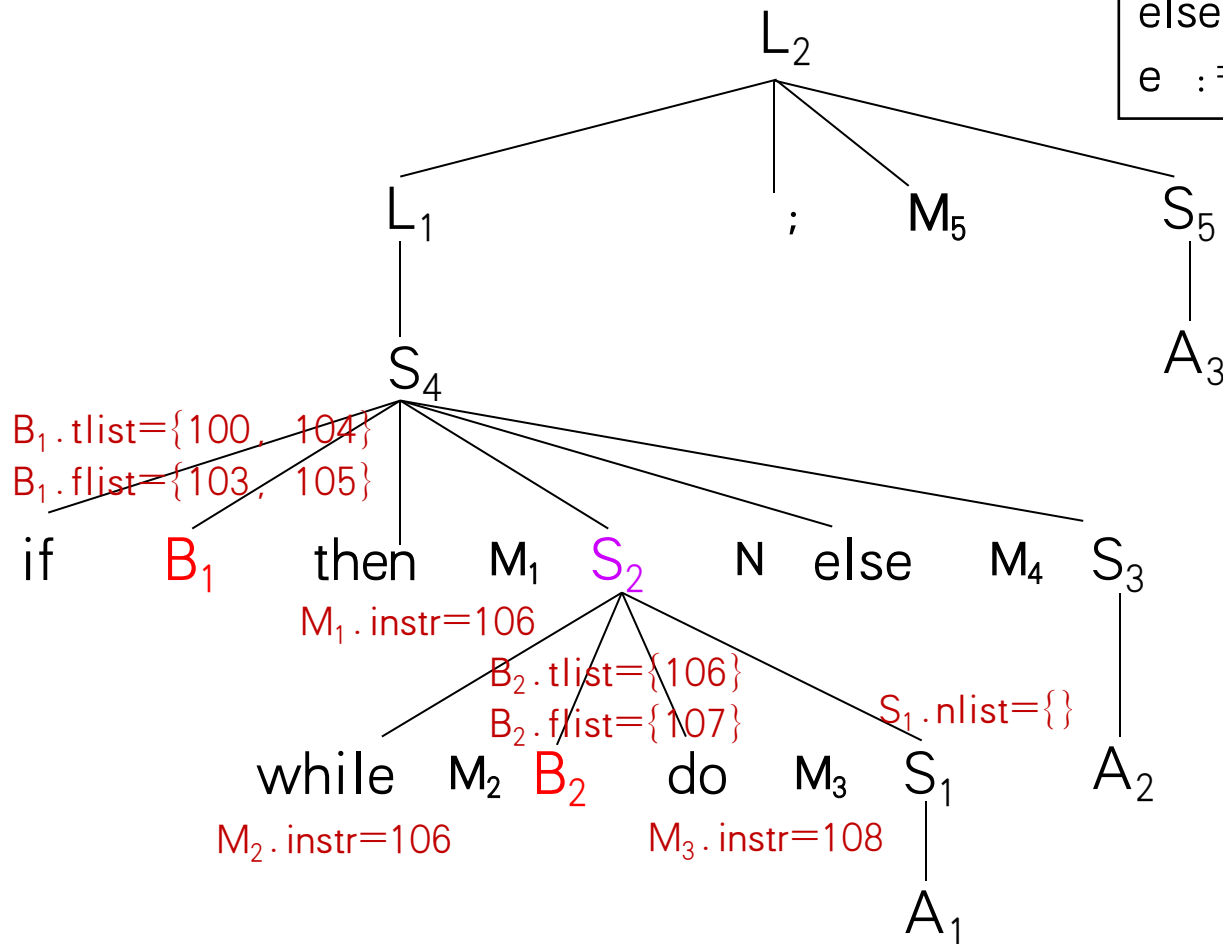
(107) goto -

(108)  $c := c + 1$  //  $S_1 \rightarrow A_1$   $S_1.nextlist = \{\}$



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c + 1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填  
紫色表示即将归约



七、归约  $S_2$ : while  $B_2$  do  $S_1$

(106) if  $a > c$  goto -

(107) goto -

(108)  $c := c + 1$  //  $S_1 \rightarrow A_1$   $S_1.nextlist = \{\}$



七、归约  $S_2$ : while  $B_2$  do  $S_1$

(106) if  $a > c$  goto 108 //用108回填106

(107) goto -

(108)  $c := c + 1$  //  $S_1 \rightarrow A_1$   $S_1.nextlist = \{\}$

(109) goto 106 // 转至循环入口(106)

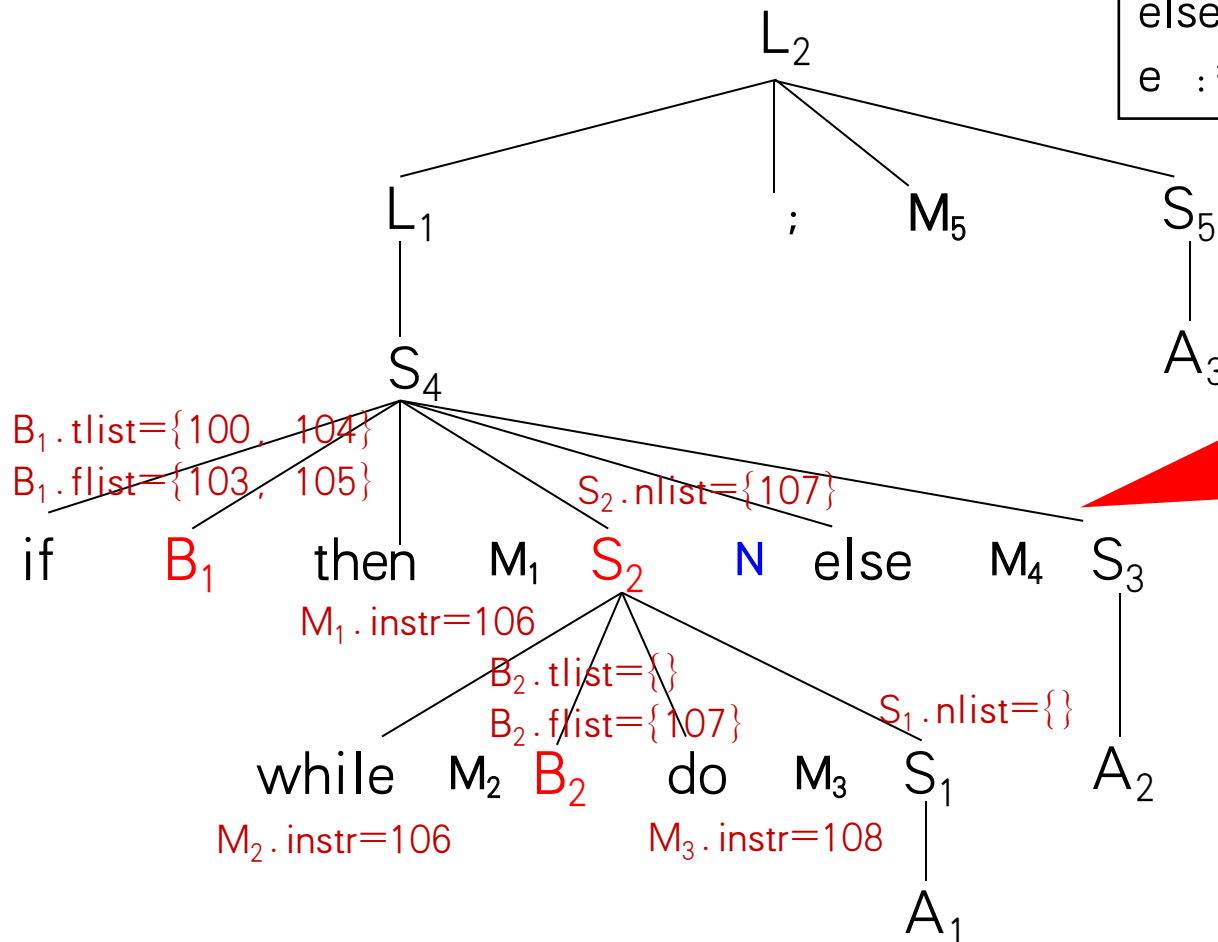
$S_2.nextlist: \{ 107 \}$  //转至循环外部

此处需要用106回填 $S_1.nextlist$ , 但该list为空



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c + 1
else d := d + 1;
e := e + d;
```



虽然while语句归约到S<sub>2</sub>,  
但是S<sub>2</sub>的下一跳指令和  
B<sub>2</sub>的假出口未定待填,  
它们指向相同。

蓝色表示即将翻译  
红色表示需要回填





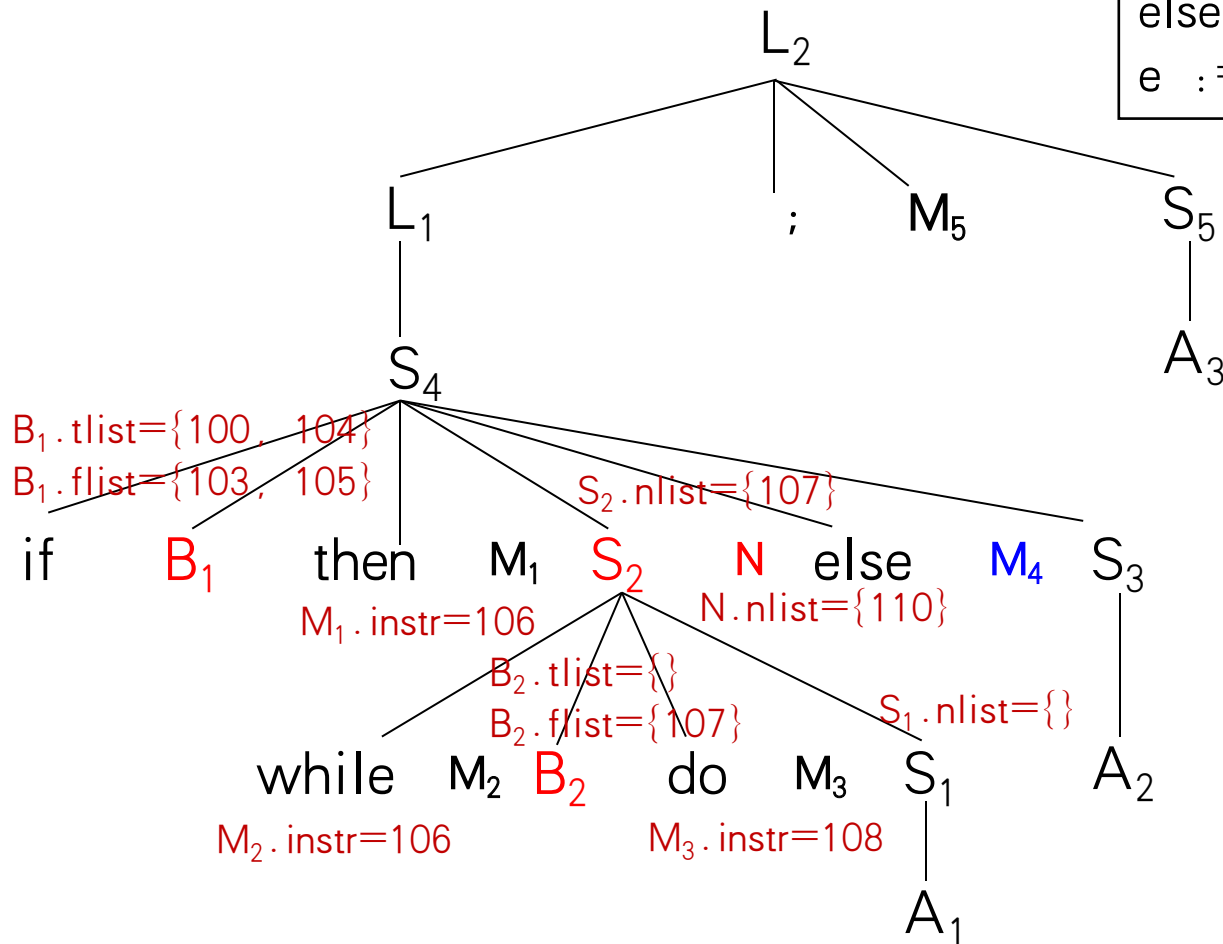
八、翻译 N:  $N \rightarrow \epsilon$

(110) goto - // N.nextlist = {110}



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c +1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填



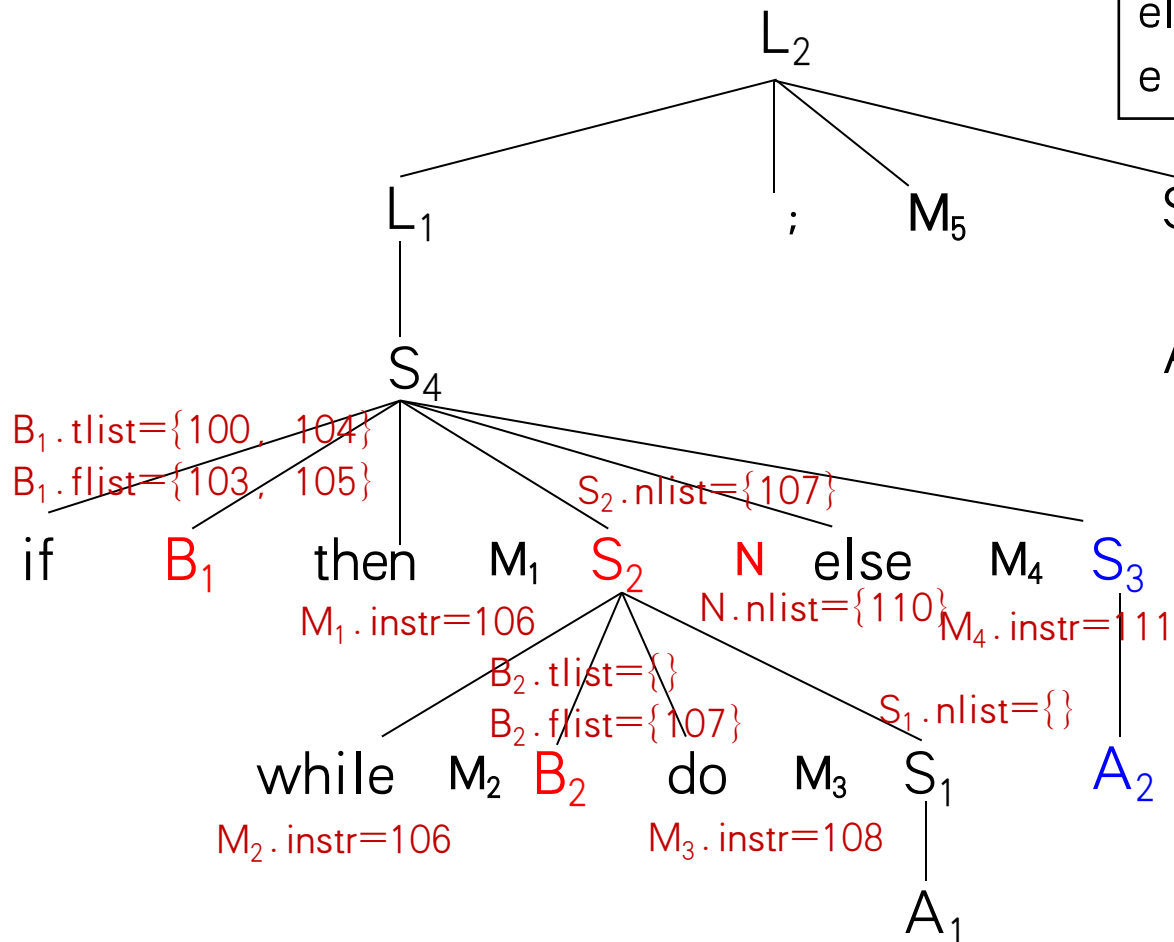
九、翻译  $M_4$ :  $M_4 \rightarrow \varepsilon$

记录下一个指令标号111，当if-then-else归约时，用111回填 $B_1$ 的  
false list{103, 105}



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c +1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填



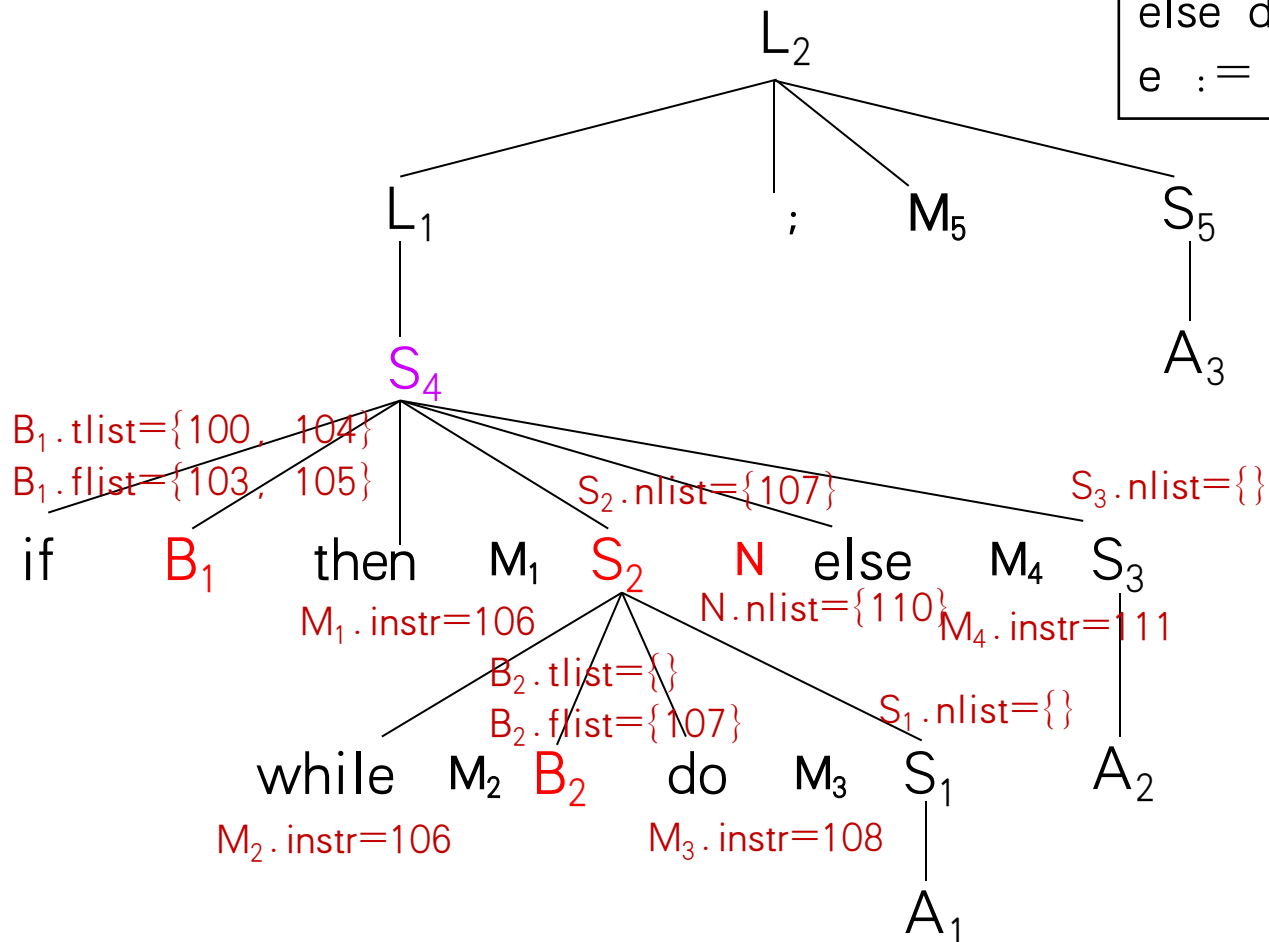
十、翻译  $S_3$  :  $S_3 \rightarrow A_2$

(111)  $d := d + 1$  //  $S_3.nextlist = \{\}$



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c +1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填  
紫色表示即将归约



十一、按if-then-else归约到 $S_4$ ，进行如下操作

- 用 $M_1.instr$ 即106回填布尔表达式 $B_1$ 的truelist{100,104}



**(100) if  $a < b$  goto 106**

**(101) goto 102**

**(102) if  $c < d$  goto 104**

**(103) goto -**

**(104) if  $e < f$  goto 106**

**(105) goto -**

**truelist: { 100, 104 } falselist: { 103, 105 }**





## 十一、按if-then-else归约到 $S_4$ ，进行如下操作

- 用 $M_1.instr$ 即106回填布尔表达式 $B_1$ 的 $truelist\{100,104\}$
- 用 $M_4.instr$ 即111回填布尔表达式 $B_1$ 的 $false\{103,105\}$



**(100) if  $a < b$  goto 106**

**(101) goto 102**

**(102) if  $c < d$  goto 104**

**(103) goto 111**

**(104) if  $e < f$  goto 106**

**(105) goto 111**

truelist: { 100, 104 } falselist: { 103, 105 }

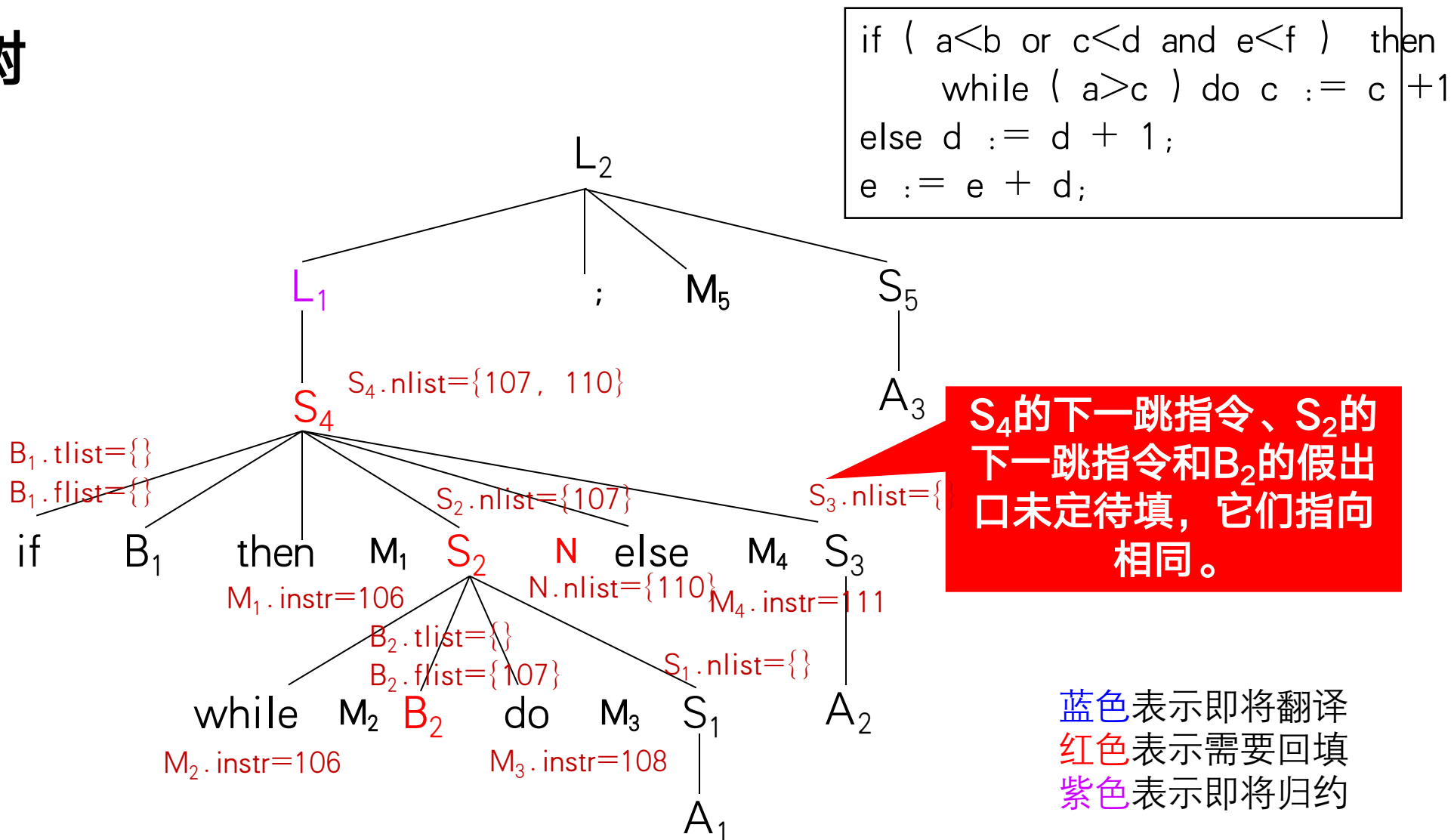


## 十一、按if-then-else归约到 $S_4$ ，进行如下操作

- 用 $M_1.instr$ 即106回填布尔表达式 $B_1$ 的 $truelist\{100,104\}$
- 用 $M_4.instr$ 即111回填布尔表达式 $B_1$ 的 $false\{103,105\}$
- $S_4.nextlist$ 等于 $S_2$ 、 $N$ 和 $S_3$ 的 $nextlist$ 的并集，即为 $\{107, 110\}$



## • 分析树





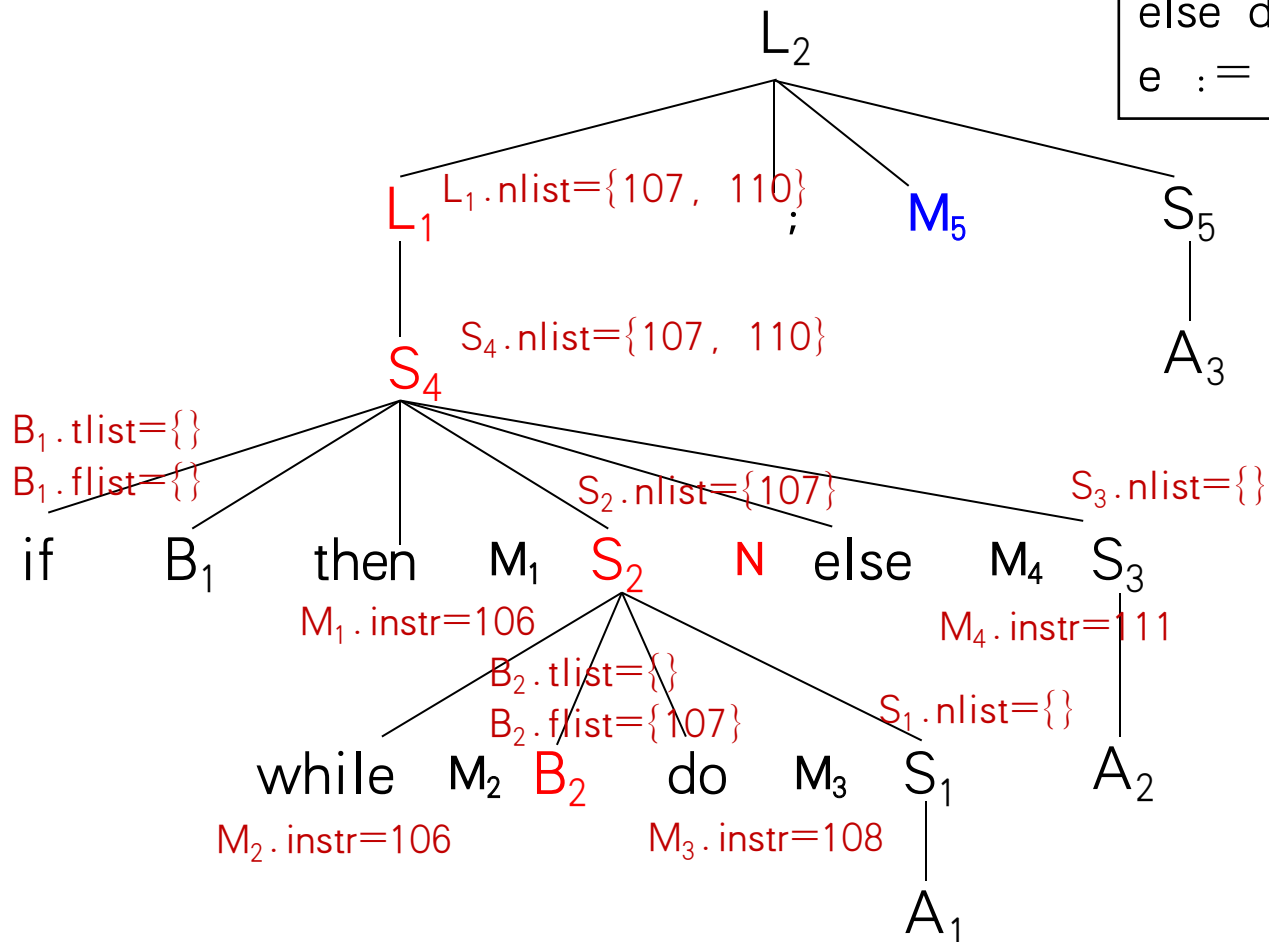
十二、归约 $L_1$ ，将 $S_4.nextlist$ 直接赋给 $L_1.nextlist$

- $L_1.nextlist$ : { 107, 110 }



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c +1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填



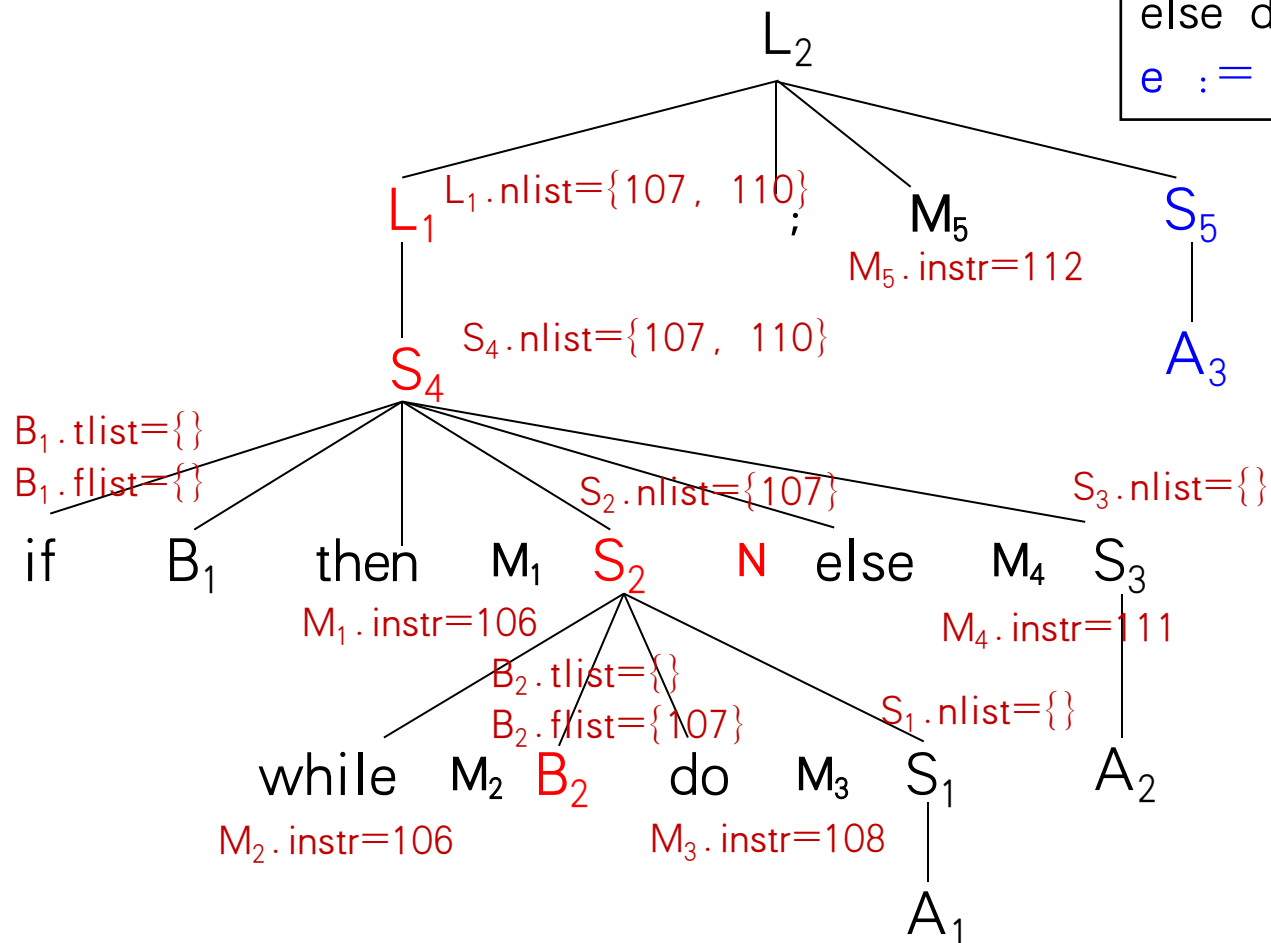
十三、翻译 $M_5$ :  $M_5 \rightarrow \varepsilon$

记录下一个指令标号112, 当 $L_1;S$ 归约时, 用112回填 $L_1$ 的  
`nextlist{107, 110}`



## • 分析树

```
if ( a<b or c<d and e<f ) then
    while ( a>c ) do c := c +1
else d := d + 1;
e := e + d;
```



蓝色表示即将翻译  
红色表示需要回填



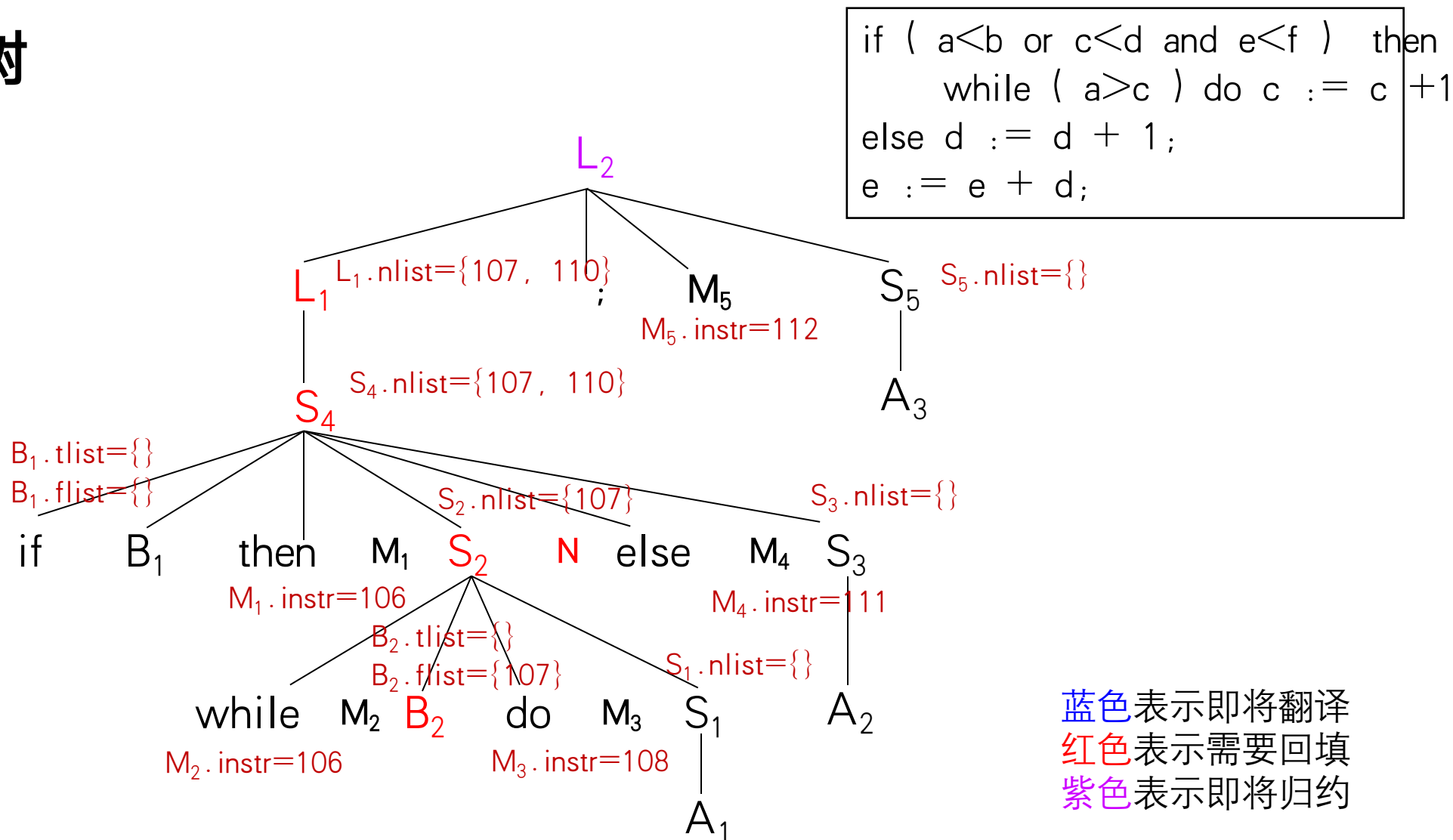


## 十四、翻译 $S_5$

(112)  $e := e + d \quad // \quad S_5 \rightarrow A_3 \quad S_5.nextlist = \{\}$



## • 分析树



蓝色表示即将翻译  
红色表示需要回填  
紫色表示即将归约



## 十五、归约 $L_2$

- 用 $M_5.instr$ 即112回填 $L_1$ 的 $nextlist\{107, 110\}$
- $L_2.nextlist = S_5.nextlist$  , 所以为空



# 控制流语句的翻译-例



(100) if  $a < b$  goto 106

(101) goto 102

(102) if  $c < d$  goto 104

(103) goto 111

(104) if  $e < f$  goto 106

(105) goto 111

(106) if  $a > c$  goto 108

(107) goto -

(108)  $c := c + 1$

(109) goto 106

(110) goto -

(111)  $d := d + 1$

(112)  $e := e + d$



# 控制流语句的翻译-例-终



(100) if  $a < b$  goto 106

(101) goto 102

(102) if  $c < d$  goto 104

(103) goto 111

(104) if  $e < f$  goto 106

(105) goto 111

(106) if  $a > c$  goto 108

(107) goto 112

(108)  $c := c + 1$

(109) goto 106

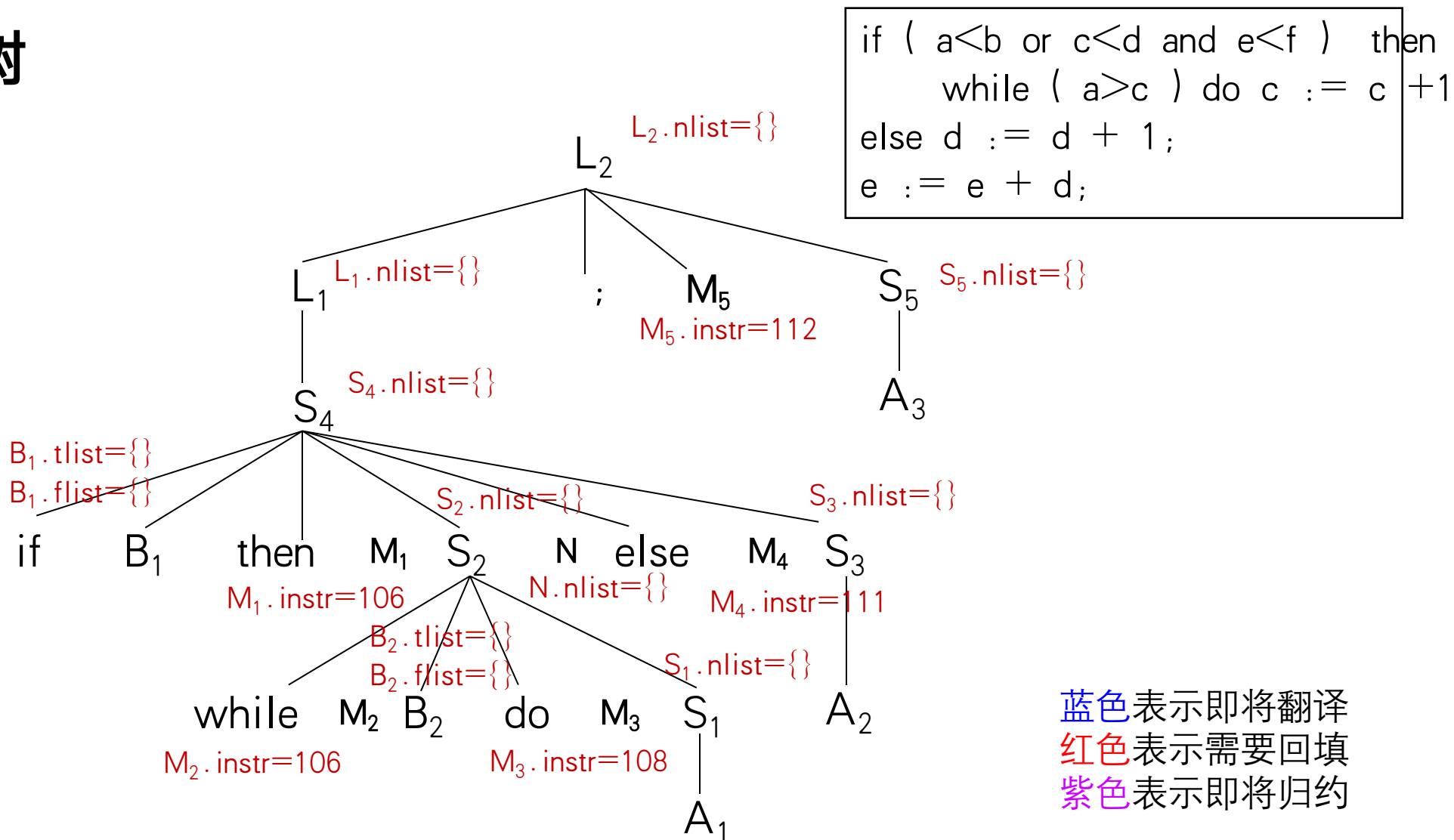
(110) goto 112

(111)  $d := d + 1$

(112)  $e := e + d$



## • 分析树



# 一起努力 打造国产基础软硬件体系！

李 诚

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2025年04月03日