

第六章作业参考答案

第一次作业

练习 6.1.2

3) $a + a + (a + a + a + (a + a + a + a))$



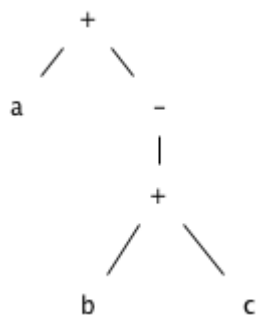
子表达式的值编码如下：

1	id	a	
2	+	1	1
3	+	2	1
4	+	3	1
5	+	3	4
6	+	2	5

练习 6.2.1

$a + - (b + c)$

1) 抽象语法树



2) 四元式序列

	op	arg1	arg2	result
0	+	b	c	t1
1	uminus	t1		t2
2	+	a	t2	t3

3) 三元式序列

	op	arg1	arg2
0	+	b	c
1	uminus	(0)	
2	+	a	(1)

4) 间接三元式序列

	statment
100	(0)
101	(1)
102	(2)

	op	arg1	arg2
0	+	b	c
1	uminus	(0)	
2	+	a	(1)

第二次作业

练习 6.3.1

SDT

```

S ->                                {top = new Evn(); offset = 0;}
    D
D -> T id;                          {top.put(id.lexeme, T.type, offset);
                                     offset += T.width}
    D1
D -> ε
T -> int                            {T.type = interget; T.width = 4;}
T -> float                          {T.type = float; T.width = 8;}
T -> record '{'
                                     {Evn.push(top), top = new Evn();
                                     Stack.push(offset), offset = 0;}
    D '}'                          {T.type = record(top); T.width = offset;

```

```
top = Evn.top(); offset = Stack.pop();}
```

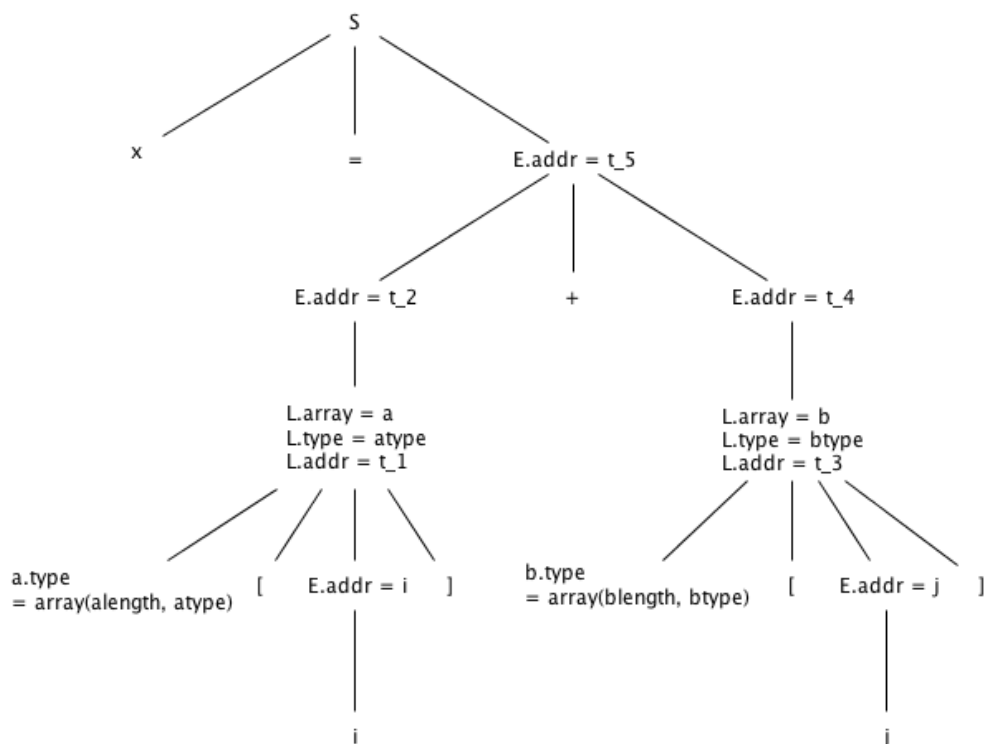
标识符类型和相对地址

line id	type	offset	Evn
1) x	float	0	1
2) x	float	0	2
2) y	float	8	2
2) p	record()	8	1
3) tag	int	0	3
3) x	float	4	3
3) y	float	12	3
3) q	record()	24	1

练习 6.4.3

1) $x = a[i] + b[j]$

语法分析树如下：



三地址代码如下：

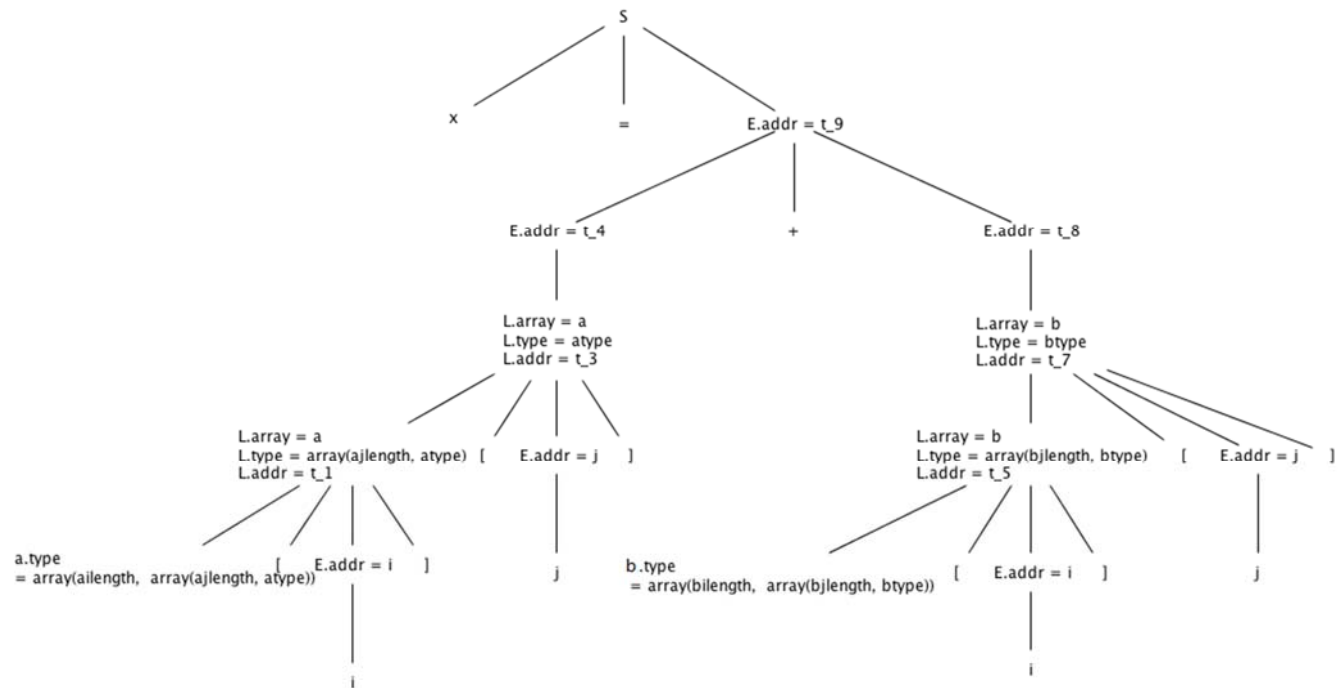
```
t_1 = i * awidth
```

```
t_2 = a[t_1]
```

```
t_3 = j * bwidth
```

$t_4 = b[t_3]$
 $t_5 = t_2 + t_4$
 $x = t_5$

2) $x = a[i][j] + b[i][j]$
 语法分析树如下：



三地址代码如下：

$t_1 = i * ai_width$
 $t_2 = j * aj_width$
 $t_3 = t_1 + t_2$
 $t_4 = a[t_3]$
 $t_5 = i * bi_width$
 $t_6 = j * bj_width$
 $t_7 = t_5 + t_6$
 $t_8 = b[t_7]$
 $t_9 = t_4 + t_8$
 $x = t_9$

练习 6.5.1

1) $x = s + c$
 $t1 = (int) s$
 $t2 = (int) c$
 $t3 = t1 + t2$
 $x = (float) t3$

```

2) i = s + c
t1 = (int) s
t2 = (int) c
i = t1 + t2

```

```

3) x = ( s + c ) * ( t + d )
t1 = (int) s
t2 = (int) c
t3 = t1 + t2
t4 = (int) t
t5 = (int) d
t6 = t4 + t5
t7 = t3 * t6
x = (float) t7

```

第三次作业

练习 6.6.1

产生式

$S \rightarrow \text{repeat } S1 \text{ while } B$

B.code

$S \rightarrow \text{for } (S1; B; S2) S3$

|| S3.code

语义规则

$S1.next = \text{newlabel}()$

$B.true = \text{newlabel}()$

$B.false = S.next$

$S.code = \text{label}(B.true) \ || \ S1.code \ || \ \text{label}(S1.next) \ ||$

$S1.next = \text{newlabel}()$

$B.true = \text{newlabel}()$

$B.false = S.next$

$S2.next = S1.next$

$S3.next = \text{newlabel}()$

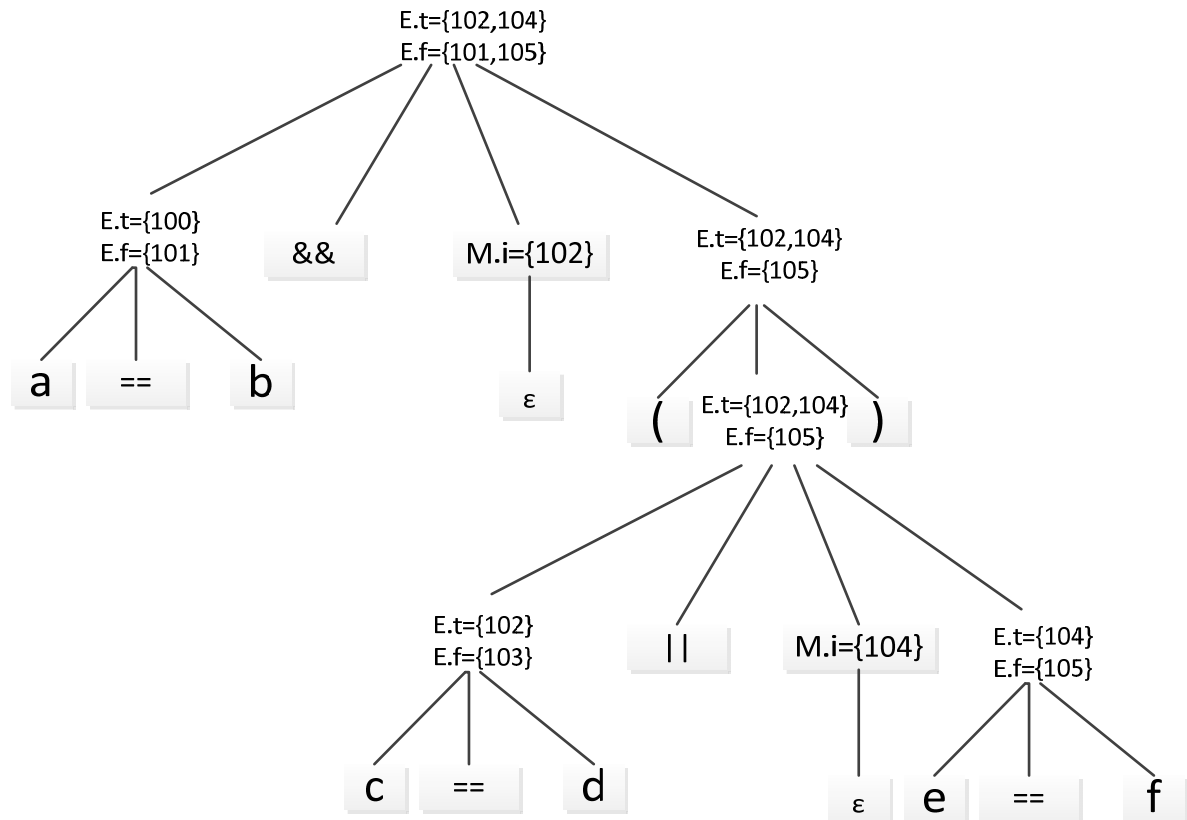
$S.code = S1.code \ || \ \text{lable}(S1.next) \ || \ B.code \ || \ \text{lable}(B.true)$

$|| \ \text{label}(S3.next) \ || \ S2.code \ || \ \text{gen}('goto', S1.next)$

练习 6.7.1

1) $a == b \ \&\& \ (c == d \ || \ e == f)$

各个子表达式的 truelist 和 falselist 如下图所示:



具体分析过程如下：

首先，对于 $a==b$ 按照 $E \rightarrow id1\ relop\ id2$ 的语义动作进行规约，产生如下指令：

100: if $a==b$ goto -

101: goto -

对于 $E \rightarrow E1\ and\ M\ E2$ 中的 M 记录了 $E2$ 的入口，所以 $M.i=\{102\}$ ；

对于 $c==d$ 按照 $E \rightarrow id1\ relop\ id2$ 的语义动作进行规约，产生如下指令：

102: if $c==d$ goto -

103: goto -

对于 $E \rightarrow E1\ or\ M\ E2$ 中的 M 记录了 $E2$ 的入口，所以 $M.i=\{104\}$

对于 $e==f$ 按照 $E \rightarrow id1\ relop\ id2$ 的语义动作进行规约，产生如下指令：

104: if $e==f$ goto -

105: goto -

接着，用产生式 $E \rightarrow E1\ or\ M\ E2$ 进行规约，执行以下动作：

```

{
    backpatch( E1.falselist, M.quad );
    E.truelist := merge( E1.truelist, E2.truelist );
    E.falselist := E2.falselist
}
  
```

先执行 $backpatch(E1.falselist, M.quad)$ ，即 $backpatch(\{103\}, 104)$ ，所以有

100: if $a==b$ goto -

101: goto -

102: if $c==d$ goto -

103: goto 104

104: if e==f goto -

105: goto -

接着执行 $E.truelist := merge(E1.truelist, E2.truelist)$ ，所以 $E.t=\{102,104\}$ ，

接着执行 $E.falselist := E2.falselist$ ，所以 $E.f=\{105\}$

然后用产生式 $E \rightarrow (E1)$ 进行规约，执行以下动作：

```
{  
    E.truelist := E1.truelist;  
    E.falselist := E1.falselist  
}
```

直接把各自的 truelist 和 falselist 拷贝过去，所以有 $E.t=\{102,104\}$ ， $E.f=\{105\}$

然后用产生式 $E \rightarrow E1 \text{ and } M \ E2$ 进行规约，执行以下动作：

```
{  
    backpatch( E1.truelist, M.quad );  
    E.truelist := E2.truelist;  
    E.falselist := merge( E1.falselist, E2.falselist )  
}
```

先执行 $backpatch(E1.truelist, M.quad)$ 即 $backpatch(\{100\}, 102)$ ，所以有：

100: if a==b goto 102

101: goto -

102: if c==d goto -

103: goto 104

104: if e==f goto -

105: goto -

接着执行 $E.truelist := E2.truelist$ ，所以 $E.t=\{102,104\}$

接着执行 $E.falselist := merge(E1.falselist, E2.falselist)$ ，所以 $E.f=\{101,105\}$

如果分别用 L1 和 L2 表示整个表达式的真、假两个出口，则最终生成的指令如下：

100: if a==b goto 102

101: goto L2

102: if c==d goto L1

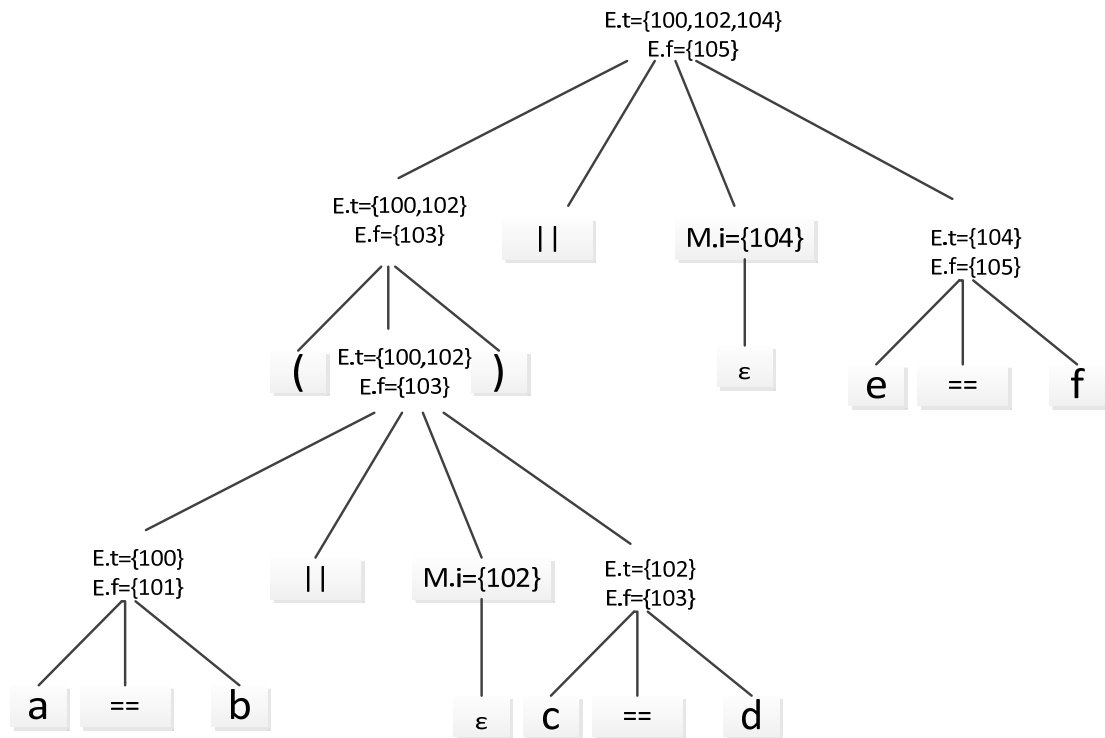
103: goto 104

104: if e==f goto L1

105: goto L2

2) $(a == b || c == d) || e == f$

与 1) 类似，可以得到各个子表达式的 truelist 和 falselist 如下：



如果分别用 L1 和 L2 表示整个表达式的真、假两个出口,则最终生成的指令如下:

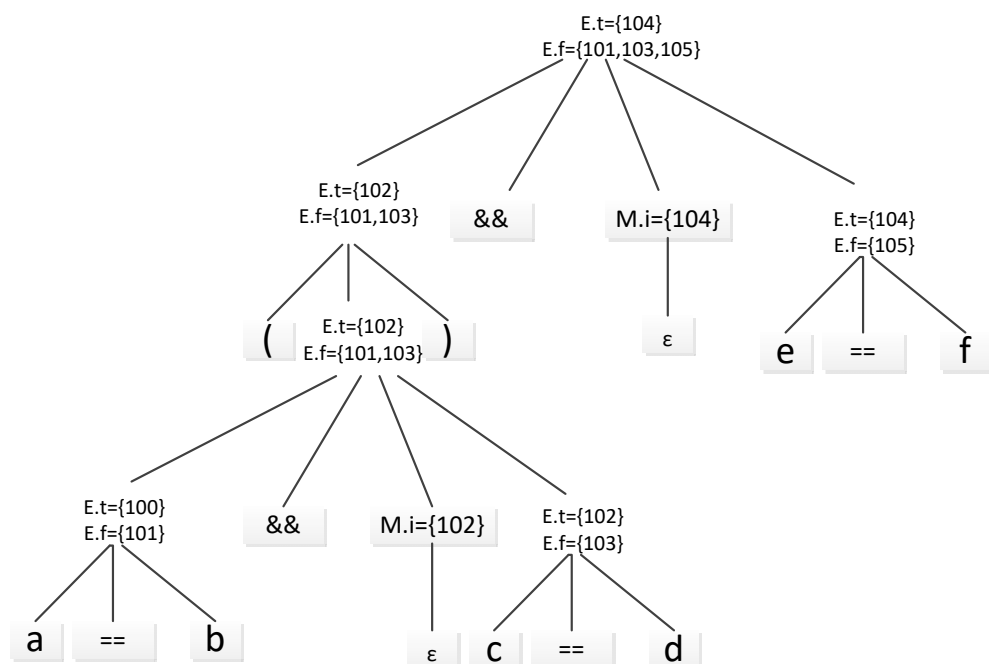
```

100: if a==b goto L1
101: goto 102
102: if c==d goto L1
103: goto 104
104: if e==f goto L1
105: goto L2

```

3) (a == b && c == d) && e == f

与 1)类似, 可以得到各个子表达式的 truelist 和 falselist 如下:



如果分别用 L1 和 L2 表示整个表达式的真、假两个出口,则最终生成的指令如下:

100: if a==b goto 102

101: goto L2

102: if c==d goto 104

103: goto L2

104: if e==f goto L1

105: goto L2