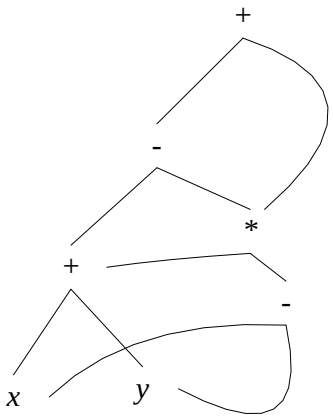
编译原理课后习题解答 第 6 章

6.1 节 语法树的变体

练习 6.1.1：为下面的表达式构造 DAG

$$((x+y)-((x+y)*(x-y)))+((x+y)*(x-y))$$



练习 6.1.2：为下列表达式构造 DAG，假定 + 是左结合的。

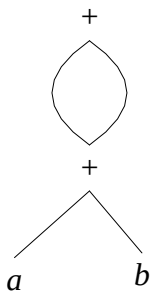
1) $a+b+(a+b)$

2) $a+b+a+b$

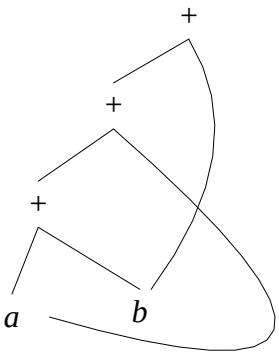
3) $a+a+(a+a+a+(a+a+a+a))$

解：

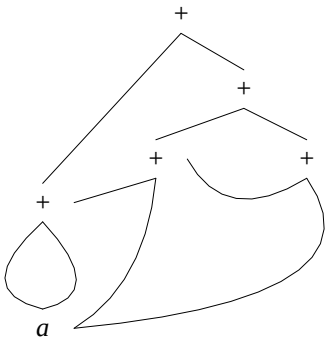
1）



2）



3）



练习 6.2.1：将算术表达式 a + -(b + c)翻译成

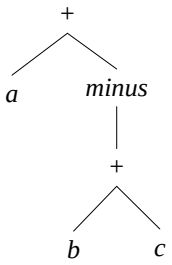
1）抽象语法树

2）四元式序列

3）三元式序列

4）间接三元式序列

解：1）



2）

	op	*arg1*	*arg2*	*restult*
0	+	b	c	t1
1	minus	t1		t2
2	+	a	t2	t3

3）

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

4）

instruction
(0)
(1)
(2)

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

练习 6.2.2：对下列赋值语句重复练习 6.2.1。

1) `a = b[i] + c[j]`

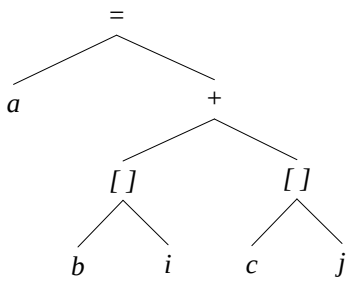
2) `a[i] = b*c - b*d`

3) `x = f(y+1) + 2`

4) `x = *p + &y`

解：1）

```
a = b[i] + c[j]
```



	op	arg1	arg2	restult
0	[]	b	i	t1
1	[]	c	j	t2
2	+	t1	t2	t3
3	=	t3		a

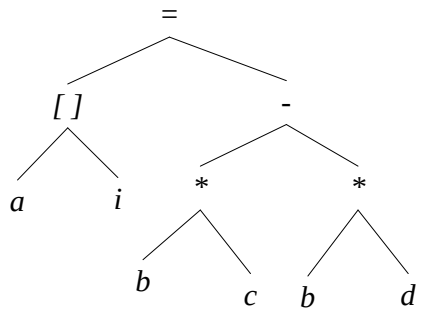
	op	arg1	arg2
0	[]	b	i
1	[]	c	j
2	+	(0)	(1)
3	=	a	(2)

instruction
(0)
(1)
(2)
(3)

	op	arg1	arg2
0	[]	b	i
1	[]	c	j
2	+	(0)	(1)
3	=	a	(2)

2)

```
a[i] = b*c - b*d
```



	op	arg1	arg2	restult
0	*	b	c	t1
1	*	b	d	t2
2	-	t1	t2	t3
3	[]=	a	i	t4
4	=	t3		t4

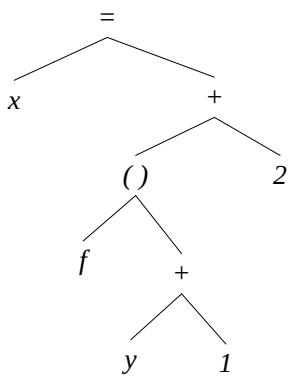
	op	arg1	arg2
0	*	b	c
1	*	b	d
2	-	(0)	(1)
3	[]=	a	i
4	=	(3)	(2)

instruction
(0)
(1)
(2)
(3)
(4)

	op	arg1	arg2
0	*	b	c
1	*	b	d
2	-	(0)	(1)
3	[]=	a	i
4	=	(3)	(2)

3)

```
x = f(y+1) + 2
```



	op	arg1	arg2	restult
0	+	y	1	t1
1	()	f	t1	t2
2	+	t2	2	t3
3	=	t3		x

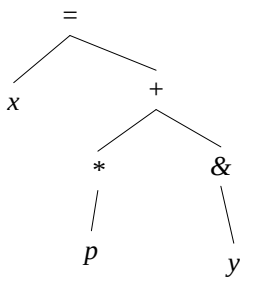
	op	*arg1*	*arg2*
0	+	y	1
1	()	f	(0)
2	+	(1)	2
3	=	x	(2)

instruction
(0)
(1)
(2)
(3)

	op	arg1	arg2
0	+	y	1
1	()	f	(0)
2	+	(1)	2
3	=	x	(2)

4)

```
x = *p + &y
```



	op	*arg1*	*arg2*	*restult*
0	*	p		t1
1	&	y		t2
2	+	t1	t2	t3
3	=	t3		x

	op	*arg1*	*arg2*
0	*	p	
1	&	y	
2	+	(0)	(1)
3	=	x	(2)

instruction
(0)
(1)
(2)
(3)

	op	arg1	arg2
0	*	p	
1	&	y	
2	+	(0)	(1)
3	=	x	(2)

6.3 类型和声明

练习 6.3.1：确定下列声明序列中各个标识符的类型和相对地址。

```
float x;
record { float x; float y; } p;
record { int tag; float x; float y; } q;
```

解：x：float 类型，相对地址 0

p：record 类型；p.x：float 类型，相对地址 0；p.y：float 类型，相对地址 8

q：record 类型；q.tag：integer 类型，相对地址 0； q.x：float 类型，相对地址 4；q.y：float 类型，相对地址 12

6.4 表达式的翻译

练习 6.4.1：向图 6-19 的翻译方案中加入对应于下列产生式的规则：

1）$E \rightarrow E_1 * E_2$

2）$E \rightarrow +E_1$（单目加）

PRODUCTION	SEMANTIC RULES
$S \rightarrow \mathbf{id} = E\ ;$	$S.code = E.code \mid\mid gen(top.get(\mathbf{id}.lexeme)\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \mid\mid E_2.code \mid\mid gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ - E_1$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \mid\mid gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr)$
$\mid\ (\ E_1\ )$	$E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \mathbf{id}$	$E.addr = top.get(\mathbf{id}.lexeme)$ $E.code = ''$

解：

$E \rightarrow E_1 * E_2$ $\quad E.addr = new\ Temp()$

$\quad E.code = E_1.code \mid\mid E_2.code \mid\mid gen(E.addr = E_1.addr'*'E_2.addr)$

$E \rightarrow +E_1$ $\quad E.addr = E_1.addr$

$\quad E.code = E_1.code$

或者作为产生新值的一元运算翻译（类似负号“-”）

$E \rightarrow +E_1$ $\quad E.addr = new\ Temp()$

$\quad E.code = E_1.code \mid\mid gen(E.addr = '+'E_1.addr)$

练习 6.4.2：使用图 6-20 中的增量式翻译方案重复练习 6.4.1。

$S \rightarrow \mathbf{id} = E\ ;$ { $gen(top.get(\mathbf{id}.lexeme)\ '='\ E.addr);$ }

$E \rightarrow E_1 + E_2$ { $E.addr = \mathbf{new}\ Temp();$
$gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);$ }

| $- E_1$ { $E.addr = \mathbf{new}\ Temp();$
$gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr);$ }

| $(\ E_1\ )$ { $E.addr = E_1.addr;$ }

| $\mathbf{id}$ { $E.addr = top.get(\mathbf{id}.lexeme);$ }

解：

$E \rightarrow E_1 * E_2$ { E.addr = new Temp();
gen(E.addr = E_1.addr'*'E_2.addr); }

$E \rightarrow +E_1$ { E.addr = E_1.addr; }

或者作为产生新值第一元运算翻译（类似负号“-”）

$E \rightarrow +E_1$ { E.addr = new Temp();
gen(E.addr = '+'E_1.addr); }

6.6 控制流

练习 6.6.1：在图 6-30 的语法制导定义中添加处理

下列控制流构造的规则：

1） A repeat-statment **repeat** S **while** B

2） A for-loop **for** $(S_1;\ B;\ S_2)\ S_3$

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$
$S \rightarrow$ **if** (B) S_1	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$
$S \rightarrow$ **while** (B) S_1	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$
$S \rightarrow S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

解：1）

$S \rightarrow repeat\ S_1\ while\ B$

$begin = newlabel()$

$B.true = begin$

$S_1.next = newlabel();$

$B.false = S.next;$

$S.code = label(begin) \mid\mid S_1.code \mid\mid label(S_1.next) \mid\mid B.code$

解：2）

$S \rightarrow for(S_1\ ;B;\ S_2\ )\ S_3$

$begin = newlabel()$
$B.true = newlabel()$
$S_3.next = newlabel()$
$B.false = S.next$
$S.code = S1.code \ ||\ label(begin)\ ||\ B.code\ ||\ label(S_3.next)||S_2.code\ ||$
$gen('goto'\ begin)||\ label(B.true)||\ S_3.code||\ gen('goto'\ S_3.next)$

练习 6.6.4：使用避免 goto 语句的翻译方案，翻译下列表达式：

1) `if (a==b && c==d || e==f) x == 1;`

2) `if (a==b || c==d || e==f) x == 1;`

3) `if (a==b && c==d && e==f) x == 1;`

解：1）

初步翻译代码	避免 goto 语句的翻译代码
if a==b goto L1 goto L2 L1: if c==d goto L3 goto L2 L2: if e==f goto L3 goto L4 L3: if x==1 goto L4 goto L4 L4:	if a==b goto L1 goto L2 L1: ifFalse c==d goto L2 L2: ifFalse e==f goto L4 if x==1 goto L4 goto L4 L4:

2）

初步翻译代码	避免 goto 语句的翻译代码
if a==b goto L3 goto L2 L2: if c==d goto L3 goto L1 L1: if e==f goto L3 goto L4 L3: if x==1 goto L4 goto L4 L4:	if a==b goto L1 L2: if c==d goto L1 L1: ifFalse e==f goto L4 L3: if x==1 goto L4 goto L4 L4:

3）

初步翻译代码	避免 goto 语句的翻译代码
if a==b goto L1 goto L4 L1: if c==d goto L2 goto L4 L2: if e==f goto L3 goto L4 L3: if x==1 goto L4 goto L4 L4:	ifFalse a==b goto L4 L1: ifFalse c==d goto L4 L2: ifFalse e==f goto L4 L3: if x==1 goto L4 goto L4 L4:

6.7 回填

练习 6.7.1：使用图 6-37 中的翻译方案翻译下列表达式

```
a==b && (c==d || e==f)

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

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

解：1）

```
    if a==b goto _(L1)
    goto _                  (B.false)
L1: if c==d goto_           (B.true)
    goto_(L2)
L2: if e==f goto_           (B.true)
    goto_                   (B.false)
```

2）

```
    if a==b goto _          (B.true)
    goto _(L1)
L1: if c==d goto_           (B.true)
    goto_(L2)
L2: if e==f goto_           (B.true)
    goto_                   (B.false)
```

3）

```
    if a==b goto _(L1)
    goto _                  (B.false)
L1: if c==d goto_(L2)
    goto_                   (B.false)
L2: if e==f goto_           (B.true)
    goto_                   (B.false)
```

练习 6.7.2：使用图 6-40 中的翻译方案翻译下列代码

```
1)while (a>b && a>c)
     if (a>0) a = a-2;
     else a = a-1;
2)if (a>b) while (a>0) a = a-2;
     else if (a<d) while(!a>0) a = a+1;
```

解：1）

```
L0: if a>b goto_(L1)
    goto_(L6)
```

```
L1: if a>c goto_(L2)
    goto_(L6)
L2: if a>0 goto_(L3)
    goto_(L4)
L3: t1 = a-2
    a = t1
    goto_(L5)
L4: t2 = a-1
    a = t2
L5: goto L0
L6:
```

2）

```
    if a>b goto_(L1)
    goto_(L4)
L1: if a>0 goto_(L2)
    goto_(L3)
L2: t1 = a-2
    a = t1
    goto_(L1)
L3: goto_(L7)
L4: if a<d goto_(L5)
    goto_(L7)
L5: ifFalse a>0 goto_(L6)
    goto_(L7)
L6: t2 = a+1
    a = t2
    goto L5
L7:
```