第二次课后作业(Sec 5-6)

1. 扩展右图中的SDD，使它可以像左图所示的那样处理表达式：

产生式	语义规则
1) $L \to E\ \mathbf{n}$	$L.val = E.val$
2) $E \to E_1 + T$	$E.val = E_1.val + T.val$
3) $E \to T$	$E.val = T.val$
4) $T \to T_1 * F$	$T.val = T_1.val \times F.val$
5) $T \to F$	$T.val = F.val$
6) $F \to (E)$	$F.val = E.val$
7) $F \to \mathbf{digit}$	$F.val = \mathbf{digit}.\text{lexval}$

产生式	语义规则
1) $T \to F\ T'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \to * F\ T'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3) $T' \to \epsilon$	$T'.syn = T'.inh$
4) $F \to \mathbf{digit}$	$F.val = \mathbf{digit}.lexval$

2. 对于图中的SDD，给出$int\ x, y, z$对应的注释语法分析树:

产生式	语义规则
1) $D \to T\ L$	$L.inh = T.type$
2) $T \to \mathbf{int}$	$T.type = \text{integer}$
3) $T \to \mathbf{float}$	$T.type = \text{float}$
4) $L \to L_1 , \mathbf{id}$	$L_1.inh = L.inh$ $addType(\mathbf{id}.entry, L.inh)$
5) $L \to \mathbf{id}$	$addType(\mathbf{id}.entry, L.inh)$

3. 图中的SDT计算了一个由0和1组成的串的值，它把输入的符号串当做按照正二进制数来解释。改写这个SDT，使得基础文法不再是左递归的，但仍然可以计算出整个输入串的相同的B.val的值：

$$
\begin{array}{lll}
B & \to & B_1\ 0\ \{B.val = 2 \times B_1.val\} \\
 & | & B_1\ 1\ \{B.val = 2 \times B_1.val + 1\} \\
 & | & 1\ \{B.val = 1\}
\end{array}
$$

4. 为下面的表达式构造DAG：

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

5. 将下列赋值语句翻译为四元式序列，三元式序列，间接三元式序列：

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

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

6. 使用下图所示的翻译方案来翻译赋值语句$x = a[i][j] + b[i][j]$:

$S \rightarrow \mathbf{id} = E\ ;$	$\{\ gen(\ top.get(\mathbf{id}.lexeme)\ '='\ E.addr);\ \}$
$\quad \mid\ L = E\ ;$	$\{\ gen(L.array.base\ '['\ L.addr\ ']'\ '='\ E.addr);\ \}$
$E \rightarrow E_1 + E_2$	$\{\ E.addr = \mathbf{new}\ Temp();$ $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);\ \}$
$\quad \mid\ \mathbf{id}$	$\{\ E.addr = top.get(\mathbf{id}.lexeme);\ \}$
$\quad \mid\ L$	$\{\ E.addr = \mathbf{new}\ Temp();$ $gen(E.addr\ '='\ L.array.base\ '['\ L.addr\ ']');\ \}$
$L \rightarrow \mathbf{id}\ [\ E\ ]$	$\{\ L.array = top.get(\mathbf{id}.lexeme);$ $L.type = L.array.type.elem;$ $L.addr = \mathbf{new}\ Temp();$ $gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \}$
$\quad \mid\ L_1\ [\ E\ ]$	$\{\ L.array = L_1.array;$ $L.type = L_1.type.elem;$ $t = \mathbf{new}\ Temp();$ $L.addr = \mathbf{new}\ Temp();$ $gen(t\ '='\ E.addr\ '*'\ L.type.width);$ $gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}$

图 6-22　处理数组引用的语义动作

7. 一个按行存放的实数型数组 A[i, j, k] 的下标 i 的范围为 1~4，下标 j 的范围为 0~4，且下标 k 的范围为 5~10。每个实数占 8 个字节。假设数组 A 从 0 字节开始存放，计算下列元素的位置:

(1) A[3, 4, 5]

(2) A[1, 2, 7]

(3) A[4, 3, 9]

8. 使用下图中的翻译方案翻译表达式a==b && (c==d||e==f)，并给出每个子表达式的真值列表与假值列表，你可以假设第一条被生成的指令的地址是100:

1)	$B \rightarrow B_1 \mid\mid M\ B_2$	$\{\ backpatch(B_1.falselist, M.instr);$ $B.truelist = merge(B_1.truelist, B_2.truelist);$ $B.falselist = B_2.falselist;\ \}$
2)	$B \rightarrow B_1\ \&\&\ M\ B_2$	$\{\ backpatch(B_1.truelist, M.instr);$ $B.truelist = B_2.truelist;$ $B.falselist = merge(B_1.falselist, B_2.falselist);\ \}$
3)	$B \rightarrow\ !\ B_1$	$\{\ B.truelist = B_1.falselist;$ $B.falselist = B_1.truelist;\ \}$
4)	$B \rightarrow (\ B_1\ )$	$\{\ B.truelist = B_1.truelist;$ $B.falselist = B_1.falselist;\ \}$
5)	$B \rightarrow E_1\ \mathbf{rel}\ E_2$	$\{\ B.truelist = makelist(nextinstr);$ $B.falselist = makelist(nextinstr + 1);$ $gen('\texttt{if}'\ E_1.addr\ \mathbf{rel}.op\ E_2.addr\ '\texttt{goto \_}');$ $gen('\texttt{goto \_}');\ \}$
6)	$B \rightarrow \mathbf{true}$	$\{\ B.truelist = makelist(nextinstr);$ $gen('\texttt{goto \_}');\ \}$
7)	$B \rightarrow \mathbf{false}$	$\{\ B.falselist = makelist(nextinstr);$ $gen('\texttt{goto \_}');\ \}$
8)	$M \rightarrow \epsilon$	$\{\ M.instr = nextinstr;\ \}$

图 6-43　布尔表达式的翻译方案