编译原理第5次作业

作业要求：

1. 使用作业本，手写
2. 提交时间：11月28日(周四)编译原理课间收

pp.232 练习 6.1.1(厚书)：
pp.220 练习 6.1.1(薄书)：

练习 6.1.1：为下面的表达式构造 DAG

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

pp.237 练习 6.2.2(厚书)：
pp.225 练习 6.2.2（薄书）：

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`

在题目中添加条件：每个数组元素占 8 个存储单元。

(注: 只要求翻译成四元式序列和三元式序列)

pp.247 练习 6.4.3(厚书)：
pp.235 练习 6.4.3(薄书)：

练习 6.4.3：使用图 6-22 所示的翻译方案来翻译下列赋值语句：

2) `x = a[i][j] + b[i][j]`

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

图 6-22　处理数组引用的语义动作

题目添加条件说明：

1） 假设一个整数的宽度为 4 个字节；

2） 假设 a 表示一个 x*3 的整型数组，即 a.type.elem.width=12，则 a[i]的类型宽度为 12；

3） 假设 b 表示一个 y*3 的整型数组，即 b.type.elem.width=12，则 b[i]的类型宽度为 12。

pp.248 练习 6.4.8(厚书)：

pp.236 练习 6.4.8(薄书)：

练习 6.4.8：一个实数型数组 $A[i, j, k]$ 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) $A[4, 3, 9]$

pp.263 练习 6.6.1(厚书):

pp.251 练习 6.6.1(薄书):

产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow$ **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	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$
$S \rightarrow S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

图 6-36　控制流语句的语法制导定义

练习 6. 6. 1：在图 6-36 的语法制导定义中添加处理下列控制流构造的规则：

1）一个 repeat 语句，**repeat** S **while** B。

pp.268 练习 6.7.1(厚书):

pp.256 练习 6.7.1(薄书):

练习 6. 7. 1：使用图 6-43 中的翻译方案翻译下列表达式。给出每个子表达式的真假值列表。你可以假设第一条被生成的指令的地址是 100。

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

1)	$B \rightarrow B_1$ || 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)	$B \rightarrow E_1$ **rel** E_2	{ $B.truelist = makelist(nextinstr)$; $B.falselist = makelist(nextinstr + 1)$; $gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}\ \_')$; $gen('\texttt{goto}\ \_')$; }
6)	$B \rightarrow$ **true**	{ $B.truelist = makelist(nextinstr)$; $gen('\texttt{goto}\ \_')$; }
7)	$B \rightarrow$ **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

pp.283 练习**7.2.4(**厚书**)**：

pp.266 练习**7.2.4(**薄书**)**：

练习 7.2.4：下面是两个 C 语言函数 f 和 g 的概述：

```
int f(int x) { int i; ··· return i+1; ··· }
int g(int y) { int j; ··· f(j+1) ··· }
```

也就是说，函数 g 调用 f。画出在 g 调用 f 而 f 即将返回时，运行时刻栈中从 g 的活动记录开始的顶端部分。你可以只考虑返回值、参数、控制链以及存放局部数据的空间。你不用考虑存放的机器状态，也不用考虑没有在代码中显示的局部值和临时值。但是你应该指出：

1）哪个函数在栈中为各个元素创建了所使用的空间？

2）哪个函数写入了各个元素的值？

3）这些元素属于哪个活动记录？