第九章 中间代码优化

1. 给出从多元式划分基本块的方法。

(答案)

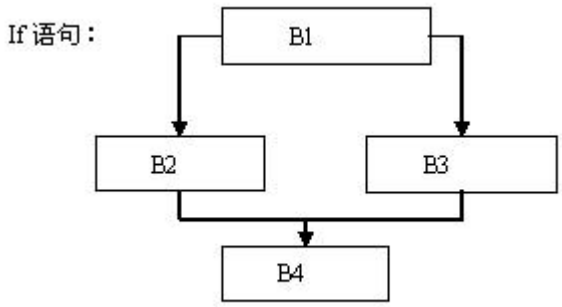
基本块开始标志的四元式有: then, else, while, label, call, proc, func, prog, body.
其中 prog 四元式是程序的第一个基本块的开始. ifend, whileend, do, goto，procend, funcend 都是基本块的结束。式(=:, A, -, X) 其中，如果 X 是一个引用型变量的话，则该四元式为一个基本块的结束标志。

(关闭)

2．给出流程图(以基本块为结点)的一种表示方法。

(答案)

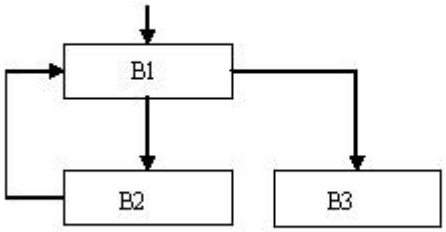
(1)条件语句:if E then S1 else S2 对应的程序流图:
B1:(E 的中间代码)(t 为结果)
(JUMP0,t,L1)
B2:(S1 的中间代码)
(JUMP,L2)
B3:(LABLE,L1)
(S1 的中间代码)
B4:(LABLE,L2)
(后续语句的目标代码)



(2)while 语句:while E do S 对应的程序流图:
B1:(LABLE,L1)
(E 的中间代码)(t 为结果)
(JUMP0,t,L2)
B2:(S 的语句部分)
(JUMP,L1)

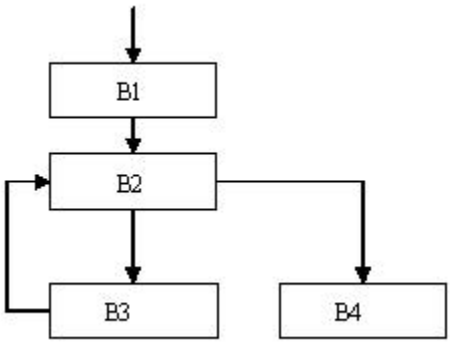
B3:(LABLE,L2)
(后续语句的目标代码)

while 语句：



(3)for 语句:for i:=E1 to E2 do S
B1:(E1 的中间代码)(t 为结果)
(:=,t,i)
B2:(LABLE,L1)
(E2 的中间代码)(j 为结果)
(LE,i,j,t)
(JUMP0,t,L2)
B3:(S 的中间代码)
(+,i,1,t1)
(:=,t1,i)
(JUMP,L1)
B4:(LABLE,L2)
(后续语句的目标代码)

for 语句：



(关闭)

3. 设有语句列

i: = 2

j: = i*(i+1);

k: = 2*(i+j)

试写出优化前和优化后的多元式代码。其中变量均为一般整形变量。

(答案)

(=:, 2, –, i)		(=:, 2, –, i)
(i+, i, 1, T_1)		
(i*, i, T_1, T_2)		
(=:,T_2, –, j)	优化后➔	(=:, 6, –, j)
(i+, i, j, T_3)		
(i*, T_3, 2, T_4)		
(=:,T_4, –, k)		(=:,16, –, k)

(关闭)

4. 设有语句列

a[i][j+1]:=a[i][j]+a[i][a[i][i]]

a[i][j]:=a[i][j+1]

其中变量均为一般变量，且有如下变量说明：

var i,j: integer;

a:array [0..5] of array[0..10] fo integer;

写出优化前和优化后的多元式代码。

(答案)

优化前代码		优化后代码
(1)(*,i,11,t1)		(1)(*,i,11,t1)
(2)([],a,t1,t2)		(2)([],a,t1,t2)
(3)([],t2,j,t3)	a[i,j]	(3)([],t2,j,t3)
(4)(*,i,11,t4)		
(5)([],a,t4,t5)		
(6)[],t5,j,t6)	a[i,j]	
(7)(*,i,11,t7)		

(8)([],a,t7,t8)		
(9)([],t8,t6,t9)	a[i,a[i,j]]	(4)([],t2,t3,t9)
(10)(*,i,11,t10)		
(11)([],a,t10,t11)		
(12)(+,j,1,t12)		(5)(+,j,1,t12)
(13)([],t11,t12,t13)		(6)([],t2,t12,t13)
(14)(+,t3,t9,t14)	a[i,j+1]	(7)(+,t3,t9,t14)
(15)(:=,t14,t13)		(8)(:=,t14,t13)
(16)(*,i,11,t15)		
(17)([],a,t15,t16)		
(18)(+,j,1,t17)		
(19)([],t16,t17,t18)	a[i,j+1]	
(20)(*,i,11,t19)		
(21)([],a,t19,t20)		
(22)([],t20,j,t21)	a[i,j]	
(23)(:=,t18,t21)		(9)(:=,t13,t3)

(关闭)

5. 设有下面语句列，写出优化前和优化后的多元式代码。其中变量为 3 题中的变量。

a[i][j]: = a[i][j]+1；

i: = j；

a[i][j]: = a[i][j]+ a[i][j]；

(答案)

(i-, i, 1, T_1)		
(i*, T_1, 10, T_2)		
([], a, T_2, T_3)		
(i-, j, 1, T_4)		
(i*, T_4, 1, T_5)		
([], T_3, T_5, T_6)		
(i-, i, 10, T_7)		
(i*, T_7, 1, T_8)		([], a, 10, T_1)
([], a, T_8, T_9)		([], T_1, 5, T_2)
(i-, j, 1, T_{10})		(i+, T_2, 1, T_3)
(i*, T_{10}, 1, T_{11})	优化后➔	(=:, T_3, -, T_2)
([], T_9, T_{11}, T_{12})		(=:, 6, -, i)
(i+, T_{12}, 1, T_{13})		([], a, 50, T_3)
(=:, T_{13}, -, T_6)		([], T_3, 5, T_4)
(=:, j, -, i)		(i+, T_4, T_4, T_5)
(i-, i, 10, T_{14})		(=:, T_5, -, T_4)
(i*, T_{14}, 1, T_{15})		
([], a, T_{15}, T_{16})		

(i-, j, 1, T_{17})		
(i*, T_{17}, 1, T_{18})		
([], T_{16}, T_{18}, T_{19})		
(i-, i, 1, T_{20})		
(i*, T_{20}, 10, T_{21})		
([], a, T_{21}, T_{22})		
(i-, j, 1, T_{23})		
(i*, T_{23}, 1, T_{24})		
([], T_{22}, T_{24}, T_{25})		
(i-, i, 1, T_{26})		
(i*, T_{26}, 10, T_{27})		
([], a, T_{27}, T_{28})		
(i-, j, 1, T_{29})		
(i*, T_{29}, 1, T_{30})		
([], T_{28}, T_{30}, T_{31})		
(i+, T_{25}, T_{31}, T_{32})		
(=:, T_{32}, -, T_{19})		

(关闭)

6．对 5 题写出用值编码的优化过程。

(答案)

	中间代码	a	i	j	11	t1	t2a	t2v	t3a	t3v	1	映像码	UE	优化后代码
0	(*,i,11,t1)		1		2	3						(1,2,3)	0	(*,i,11,t1)
1	([],a,t1,t2*)	4					5	5				(4,3,5)	0,1	([],a,t1,t2*)
2	([],t2*,j,t3*)			6					7	7		(5,6,7)	0,1,2	([],t2*,j,t3*)
3	(*,i,11,t4)											(1,2,3)	0,1,2	R1
4	([],a,t4,t5*)											(4,3,5)	0,1,2	R2
5	([],t5*,j,t6*)											(5,6,7)	0,1,2	R3
6	(+,t6*,1,t7)										8	(7,8,9)	0,1,2,6	(+,t3*,1,t7)
7	(:=,t7,t3*)									9				(:=,t7,t3*)
8	(:=,j,i)		2											(:=,j,i)
9	(*,i,11,t8)											(2,2,10)		(*,i,11,t8)
10	([],a,t8,t9*)											(4,10,11)		([],a,t8,t9*)
11	([],t9*,j,t10*)											(11,2,12)		([],t9*,j,t10*)
12	(*,i,11,t11)											(2,2,10)		R8
13	([],a,t11,t12*)											(4,10,11)		R9
14	([],t12*,j,t13*)											(11,2,12)		R10
15	(*,i,11,t14)											(2,2,10)		R8
16	([],a,t14,t15*)											(4,10,11)		R9
17	([],t15*,j,t16*)											(11,2,12)		R10
18	(*,i,11,t17)											(2,2,10)		R8
19	([],a,t17,t18*)											(4,10,11)		R9
20	([],t18*,j,t19*)											(11,2,12)		R10
21	(+,t16*,t19*,t20)											(12,12,13)		(+,t10*,t10*,
22	(:=,t20,t13*)													(:=,t20,t10*)

(关闭)

7． 设有语句列：

u: = X * u + X * u;

w: = X * u + X * u * w;

且其中 X 为整形引用形参，其他为一般整形变量。写出优化前和优化后的多元式代码。

(答案)

(i*, X, u, T_1)		
(i*, X, u, T_2)		(i*, X, u, T_1)
(i+, T_1, T_2, T_3)		(i+, T_1, T_1, T_2)
(=:, T_3, -, u)		(=:, T_2, -, u)
(i*, X, u, T_4)	优化后➔	(i*, X, u, T_3)
(i*, X, u, T_5)		(i*, T_3, w, T_4)
(i*, T_5, w, T_6)		(i*, T_3, T_4, T_5)
(i*, T_4, T_6, T_7)		(=:, T_5, -, w)
(=:, T_7, -, w)		

(关闭)

8. 设有下列循环语句，其中变量均为一般变量。写出外提后的多元式代码。

```
for i:=1 to n do
 begin
  u:=x*y;
  m:=u*u;
  s:=s+m*m
 end
```

(答案)

```
(*,x,y,t1)
(*,t1,t1,t2)
(*,t2,t2,t3)
(+,s,t3,t4)
(:=,1,i)
(LABEL,L1)
(LE,i,n,t5)
(JUMP0,t5,L2)
(:=,t1,u)
(:=,t2,m)
(:+,t4,s)
(+,i,1,t6)
(:=,t6,i)
(JUMP,L1)
(LABLE,L2)
```

(关闭)

9. 设有下面数组相乘循环，写出子表达式节省和循环优化后的多元式代码。

```
for i: = 1 to 10 do
    for j: = 1 to 10 do
    begin
        A[i][j] : = 0;
        for k : = 1 to 10 do
            A[i][j] : = A[i][j] + B[i][k] * C[k][j]
    end
```

(答案)

(=:, 1, −, i)	
(while, −, −, −)	
(<=, i, 10, T_1)	
(DO, T_1, −, −)	
(j-, i, 1, T_2)	
(j*, T_2, 10, T_3)	
([], A, T_3, T_4)	//A [i]
([], B, T_3, T_5)	//B[i]
(=:,1, −, j)	
(while, −, −, −)	
(<=, j, 10, T_6)	
(DO, T_6, −, −)	
(j-, j, 1, T_7)	
(j*, T_7, 1, T_8)	
([], T_4, T_8, T_9)	//A[i][j]
(=:,0, −, T_9)	
(=:,1, −, k)	
(while, −, −, −)	
(<=, k, 10, T_{10})	
(DO, T_{10}, −, −)	
(j-, k, 1, T_{11})	
(j*, T_{11}, 1, T_{12})	
([],T_5, T_{12}, T_{13})	//B[i][k]
(j*, T_{11}, 10, T_{14})	
([], C, T_{14}, T_{15})	//C[k]
([],T_{15}, T_9, T_{16})	//C [k] [j]
(j*, T_{13}, T_{16}, T_{17})	
(j+, T_9, T_{17}, T_{18})	
(=:,T_{18}, −, T_9)	
(j+, k, 1, T_{10})	
(while, −, −, −)	
(j+, j, 1, T_6)	
(while, −, −, −)	
(j+, i, 1, T_1)	
(while, −, −, −)	

(关闭)

10．实现一个具体的常表达式节省算法。

(答案)

```
ConsNode=RECORD
            id:string;
            val:INTEGER;
            next:↑ConsNode
         END
ConsList=RECORD

            length:INTEGER;
            Head:↑ConsNode
         END
Tuple =RECORD
            case kind:char of
            ASSIG:(val,id1);
            OP:     (oper1,oper2,result:int)
       END
curList:↑ConsList ;
curList=NULL;
while(基本块未结束)
{
    read(tuple);
    newtuple=translate(tuple);
    if (newtuple.kind=OP)
    {   if (IsConst(newtuple.oper1)&&IsConst(newtuple.oper2))
          {num=compute(OP,newtuple.oper1,newtuple.oper2);
          insert(curList,result,num); }
       else{output(newtuple);}
    }
    else
    { if (IsConst(newtuple.val)
        {insert(curList,newtuple,id1,newtuple.val);
         }
      else{delete(curList,newtuple.id1);output(newtuple);}
    }
```

11. 实现一个具体的子表达式节省算法。

(答案)

引入编码表 TCL(Temporary Coding List)

TCL 单元为 (临时变量　编码)

引入等价表 PAIR

(临时变量　编码)

PAIR 单元为 若<Ti,Tj>在 PAIR 表中,则所有 Ti 的出现可以用 Tj 替换.

可用四元式表 UsabelQD

在本基本块中已经扫描过并且没有节省的可用四元式

步骤:

Ⅰ. 进入基本块, 清空 PAIR, UsabelQD, TCL.

Ⅱ. 扫描每一个四元式.

Ⅲ. 设当前四元式为 Dj(ω, A, B, Tj)

①替换 A, B, 设替换后为(ω, A1, B1, Tj)

②若 A1,B1 是分量的第一次出现, 将(A1,NewCode), (B1,NewCode)填入 TCL; 若不是第一次出现, 用偏码替设四元式为 (ω, Code(A1),Code(B1), Tj).到 UsabelQD 中找是否有相似的.

③若找到 Di (ω, Code(A11),Code(B11), Ti)相似, Tj 填到 TCL 表, 其偏号 Code(Tj)和 Code(Ti)填入 PAIR 表. 若没找到 Di, Tj 偏码填入 TCL, 当前四元式送入 UsabelQD.

④(=:, A, -, B) B 是非间接变量, 将 B 偏码改成 A 的偏码, 删除可用四元式表中所有与 B 相关的四元式.

(关闭)

12. 实现一个具体的循环不变式的循环外提算法。

(答案)

(无)

(关闭)

13. 试说明形如 X: = Expr 的语句对于各种优化的影响，其中 X 是引用型形参变量。实际对过程调用也有一样的
(1) 对常表达式优化的影响？
(2) 对公共表达式优化的影响？
(3) 循环不变表达式外提优化的影响？
(5) 对归纳表达式优化的影响？

(答案)

(1) 什么也不做,结束当前基本块.
(2) 若有下面的语言, A 为全局变量, X 是引用型形参变量, 且初值为 A.
B:=2*A;
X:=Expr;
C:=2*A;
则不可以对表达式 2*A 进行外提, 因为 X 初始值为 A.则 B=2A, 而当 X:=Expr 时, A 也变为 Expr,而不是 A
C:=2*Expr, 故 B, C 不相等, 即不能进行公共表达式外提.
(3). 考虑下面情况, A 为全局变量, X 是引用型形参变量, 且初值为 A
对于下面语句序列
i:=0,
while (i<A) do
Begin X:=o;
i:=i+1;
end
则 X:=o;语句改变了 A 的值为 0, 因此, X:=o 不能外提; 因为若 A>0 则循环可以进入一次, 而若外提了, 则不
循环了.
(4). 考虑下面情况, A 为全局变量, X 是引用型形参变量, 且初值为 A
对于下面语句序列
FOR i:=1 TO 10 DO
BEGIN S:=A*i + B;
X:=2A;
END;
则不能进行归纳表达式优化.

(关闭)

14. 假设不把上述语句作为基本块的结束，而且不进行别名分析，那么基本块为单位的常表达式和公共表达式优
进行(即应做哪些修改)。

(答案)

常表达式优化：每遇见上述语句时把常量登记表的非临时变量写成赋值代码，并清除，临时变量保存；
公共表达式优化也类似，遇见上述语句时，清除可用表达式中含非临时变量为分量的表达式.

(关闭)

15. 假设在循环内有对引用变量的赋值或过程调用，而且没有进行别名分析，那么最差的一种方案是对这种循环表达式外提优化；但我们可以利用 FOR 循环变量在内循环中不改变值的事实对内层循环进行外提优化。

(答案)

若引用型变量有一个仅关于循环变量的赋值，则可能进行外提;设置一个表 LINK,若只受制于循环变量就将这个 LINK 中，以后若引用型变量的赋值右部都为常量或出自 LINK 则可以外提。

(关闭)

<<上一章◎ ◎回首页◎ ◎下一章>>