第六章：中间代码优化（2）

公共表达式节省

- 公共表达式：是指两个表达式中的子表达式他们的运算和运算分量的值都相同。
- 公共表达式节省在局部进行的时候都是针对基本块的。

公共表达式节省

- 设有下列语句：

```
t := b * c ;
e := b * c + b * c ;
c := b * c + 10 ;
d := b * c + d ;
```

- 优化后，可得到下列语句：

```
t := b * c ;   e :=t + t ;
c := t + 10 ;   d := b * c + d ;
```

基于值编码的公共表达式节省

- 值编码优化方法的主要思想:

  对中间代码中出现的每个值确定一个编码，使得具有相同值的运算分量对应的编码相同.

基于值编码的公共表达式节省

- 编码原理：
- 用到一张值编码表，表示分量当前值的编码
- 若当前考察的代码为dk：（ω，u1，u2，u3）：
  - 若值编码表中已有(ui，mi)则令dk:ui的值编码为mi， i=1，2，3
  - 否则为dk:ui创建一个新编码m填入编码表
- 若考察代码为dk：（＝，u1，－，u2）：
  - u1的处理与之前相同
  - 令dk:u2的编码与dk:u1的编码相同

基于值编码的公共表达式节省

- 值编码性质

1. 不同分量上的相同常量一定具有相同值编码
2. 不同分量上的相同变量未必具有相同编码
3. 不同常量的编码值一定不同，对变量来说并非如此
4. 每当一个变量x被赋值，x将得到一个新的编码，使得后面代码中的x分量取编码值n，直至x再被赋值

基于值编码的公共表达式节省

- 在分析的过程中对应每条四元式还要生成一个编码四元式。
- μ（ x ） 表示任意运算分量x的值编码；
- 把 （ ω， μ（ A)，μ（ B ），μ（ t ））叫作（ ω ， A ， B ， t ）的映象码；

基于值编码的公共表达式节省

算法流程：从基本块的第一条四元式开始

- 等价变量表替换
- 对四元式中的分量进行编码
- 值编码表替换，生成编码四元式（当前表达式的映象码）
- 遇到运算型四元式，查当前基本块中已处理过表达式的映象码中是否存在与编码四元式运算符、运算分量都相同的，若存在则说明当前表达式为公共表达式，删去当前四元式，把等价的名字填入等价表
- 遇到赋值(=, a, –, b)，b的编码赋值为a的编码

例： a=b*c+b*c; d=b； e=d*c+b*c

序号	中间代码	映像码	a	b	c	d	e	t1	t2	t3	t4	t5	t6	等价表	优化后
1	*,b,c,t1	*,1,2,3		1	2			3							不变
2	*,b,c,t2	*,1,2,3							3					(t1,t2)	节省
3	+,t1,t2,t3	+,3,3,4								4					+,t1,t1,t3
4	=,t3,-,a		4												不变
5	=,b,-,d					1									不变
6	*,d,c,t4	*,1,2,3									3			(t1,t4)	节省
7	*,b,c,t5	*,1,2,3										3		(t1,t5)	节省
8	+,t4,t5,t6	+,3,3,4											4	(t3,t6)	节省
9	=,t6,-,e						4								=,t3,-,e

循环不变式外提

- 循环不变式:如果一个表达式E在一个循环中不改变其值，则称它为该循环的不变表达式。
- 循环不变式外提：将循环不变式提到循环外面进行.

例如，假设有程序段：

```
i:=1;// t=x*y;
while i<=1000 do
    begin a[i]:= x*y //t ;i:=i+1 end;
```

则x*y是该循环的不变表达式，可以把它提到循环外边

矩阵乘法：有a，b为10*10数组

```
for(i=1~10)
   for(j=1~10)
      for(k=1~10)
         c[i][j]=c[i][j]+a[i][k]*b[k][j];
```

a[i]地址的计算与j、k循环无关，c[i][j]地址的计算与k循环无关

循环不变式外提

- 解决两个问题
  - 检查循环体中哪些变量的值被改变过
  - 根据这个结果来看哪些表达式是不变的表达式
- 建立变量定值表，将循环体中值被改变的变量都填到表里，若某运算型四元式中两个运算分量都不出现在这个表里，就说明其值不发生改变，可以进行外提。

循环不变式外提

- 对循环体四元式进行第一遍扫描，把有定值的变量填到变量定值表中，若它是一个运算型四元式(ω_1, A_1, B_1, t_1)，则把t1填到表中，若为赋值型四元式(=，a，-，b)则把b填入表中
- 循环不变式外提为第二遍扫描，每遇到一个运算型四元式(ω_1, A_1, B_1, t_1)，若A_1、B_1都不在变量定值表中，则将其提到循环体外，同时在变量定值表中删去t_1

```
i:=1
while i<=100 do
        begin
            z:=i*k*5;
            a:=2*k+2*k*2;
            i:=i+1;
        end
```

优化前四元式序列：

1. (ASSIG, 1, —, i)
2. (WHILE, —, —, —)
3. (LE, i, 100, t1)
4. (DO, t1, —, —)
5. (MULTI, i, k, t2)
6. (MULTI, t2, 5, t3)
7. (ASSIG, t3, —, z)
8. (MULTI, 2, k, t4)
9. (MULTI, 2, k, t5)
10. (MULTI, t4, 2, t6)
11. (ADDI, t4, t6, t7)
12. (ASSIG, t7, —, a)
13. (ADDI, i, 1, t8)
14. (ASSIG, t8, —, i)
15. (ENDWHILE, —, —, —)

LoopDef
={t1, t2, t3, z,
t4, t5, t6, t7, a,t8 ,i}

循环优化前四元式序列：

1. (ASSIG, 1, —, i)
2. (WHILE,-, -, -)
3. (LE, i, 100, t1)
4. (DO, t1, -, -)
5. (MULTI, i, k, t2)
6. (MULTI, t2, 5, t3)
7. (ASSIG, t3,- , z)
8. (MULTI, 2, k, t4)
9. (MULTI, t4, 2, t6)
10. (ADDI, t4, t6, t7)
11. (ASSIG, t7, —, a)
12. (ADDI, i, 1,t8)
13. (ASSIG, t8, —, i)
14. (ENDWHILE,-, -, -)

LoopDef={t1, t2, t3, z, t4, t6, t7, a,t8 ,i}

循环优化前四元式序列：

1. (ASSIG, 1, －, i)
2. (WHILE, -，-，-)
3. (LE, i, 100, t1)
4. (DO, t1, —，—)
5. (MULTI, i, k, t2)
6. (MULTI, t2, 5, t3)
7. (ASSIG, t3, －, z)
8. (MULTI, 2, k, t4)
9. (MULTI, t4, 2, t6)
10. (ADDI, t4, t6, t7)
11. (ASSIG, t7, －, a)
12. (ADDI, i, 1,t8)
13. (ASSIG, t8, －, i)
14. (ENDWHILE, -，-，-)

循环优化后四元式序列：

1. (ASSIG, 1, －, i)
2. (MULTI, 2, k, t4)
3. (MULTI, t4, 2, t6)
4. (ADDI, t4, t6, t7)
5. (WHILE,-，-，-)
6. (LE, i, 100, t1)
7. (DO, t1, —，—)
8. (MULTI, i, k, t2)
9. (MULTI, t2, 5, t3)
10. (ASSIG, t3, －, z)
11. (ASSIG, t7, －, a)
12. (ADDI, i, 1,t8)
13. (ASSIG, t8, －, i)
14. (ENDWHILE,，-，-)

循环不变式外提中注意的问题

1. 多层循环问题中，一个四元式从里层开始可以被外提若干次，里层变量定值表属于外层变量定值表
2. 除法不外提
3. 赋值绝不外提
4. 非良性循环（循环体有函数调用或地址引用型变量的赋值）不做外提优化
5. 对于运算型四元式，如果其中一运算分量是间接访问的，则该四元式不做外提优化

例子

例2:已知:a:array[1..10][1..1000] of integer;设有如下循环语句:

```
j:=1
  while j<=1000 do
     begin a[i][j]:=0;
           j:=j+1;
      end;
```

优化前的四元式序列：

(ASSIG, 1, —, j)
(WHILE, —, —, —)
(LE, j, 1000, t1)
(DO, t1, —, —)
(SUBI, i, 1, t2)
(MULTI, t2, 1000, t3)
(AADD, a, t3, t4)
(SUBI, j, 1, t5)
(MULTI, t5, 1, t6)
(AADD, t4, t6, t7)
(ASSIG, 0, —, t7)
(ADDI, j, 1, t8)
(ASSIG, t8, —, j)
(ENDWHILE, —, —, —)

LoopDef={t1, t2, t3, t4, t5, t6, t7, t8, j}

优化后的四元式序列：

(ASSIG, 1, —, j)
(SUBI, i, 1, t2)
(MULTI, t2, 1000, t3)
(AADD, a, t3, t4)
(WHILE, —, —, —)
(LE, j, 1000, t1)
(DO, t1, —, —)
(SUBI, j, 1, t5)
(MULTI, t5, 1, t6)
(AADD, t4, t6, t7)
(ASSIG, 0, —, t7)
(ADDI, j, 1, t8)
(ASSIG, t8, —, j)
(ENDWHILE, —, —, —)

LoopDef={t1, t5, t6, t7, t8, j}

```
j:=1
while j<=1000 do
        begin a[i][j]:=0;
              j:=j+1;
        end;
```

给出如下程序段的四元式中间代码，并利用常表达节省、公共表达式节省和循环不变式外提三种优化技术对中间代码进行优化。其中A：array[1..100] of integer，程序中的所有变量都为整型变量，占1个存储单元。

```
a:=0;
j:=(a+1)*5;
while j<100 do
begin
        a:=a+A[3];
        if a<10 then  k:=0;
        else    k:=x*y+x*y/5;
         j:=j+1;
end
```

中间代码:

1. (: = , 0 , _ , a)
2. (+, a, 1, t1)
3. (*, t1, 5, t2)
4. (:=, t2,_, j)
5. (WHILE, _, _, _)
6. (<, j, 100, t3)
7. (Do, t3, _, _)
8. (-,3, 1, t4)
9. (*, t4, 1, t5)
10. (AADD, A, t5, t6)
11. (+, a, t6, t7)
12. (:=, t7, _, a)
13. (<, a, 10, t8)
14. (THEN, t8, _, _)
15. (:=, 0, _, k)
16. (ELSE, _, _, _)
17. (*, x, y, t9)
18. (*, x, y, t10)
19. (/, t10, 5, t11)
20. (+, t9, t11, t12)
21. (:=, t12, _, k)
22. (ENDIF, _, _, _)
23. (+, j, 1, t13)
24. (:=, t13, _, j)
25. (ENDWHILE, _, _, _)

优化后的中间代码：

1. (: = , 0 , _ , a)
2. (:=,5,_, j)
3. (AADD, A, 2, t6)
4. (*, x, y, t9)
5. (WHILE, _, _, _)
6. (<, j, 100, t3)
7. (Do, t3, _, _)
8. (+, a, t6, t7)
9. (:=, t7, _, a)
10. (<, a, 10, t8)
11. (THEN, t8, _, _)
12. (:=, 0, _, k)
13. (ELSE, _, _, _)
14. (/, t9, 5, t11)
15. (+, t9, t11, t12)
16. (:=, t12, _, k)
17. (ENDIF, _, _, _)
18. (+, j, 1, t13)
19. (:=, t13, _, j)
20. (ENDWHILE, _, _, _)