

第六章：中间代码优化（1）

代码优化的阶段

- ◆ 欲提高源程序的运行速度，需要经过几个阶段的优化：
 - ❖ 用户对源程序进行优化
 - ❖ 编译器前端对中间代码进行优化
 - ❖ 编译器后端对目标代码进行优化

中间代码优化的分类

- ◆ 从优化的种类来看，中间代码的优化可有如下分类：
- ◆ 局部优化
 - 循环上的优化
 - 循环不变式外提
 - 削减运算强度
 - 基本块的优化
 - 常表达式节省
 - 公共子表达式节省。
- ◆ 全局优化
 - 全局数据流分析，从而使优化的效果更好。

常见的编译优化的种类

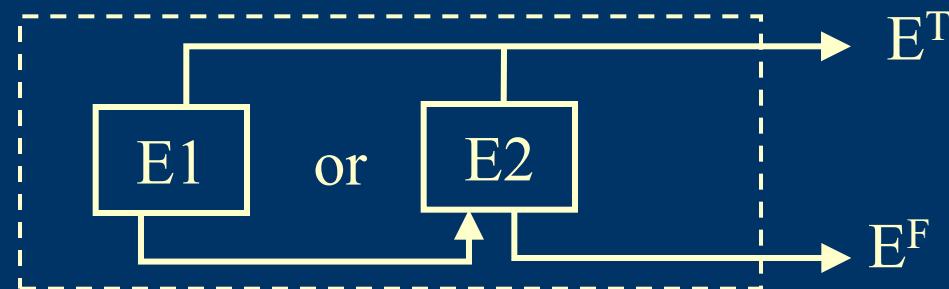
- ◆ 数学上的优化
 - ❖ $(-, a, 0, t1)$ $(*, a, 1, t2)$
 - ❖ $2*a$ 转化成 $a+a$, a^2 转化成 $a*a$

原则上说，可以不计算的则直接删去其四元式，直接写出结果；高运算强度的可以转化成低运算强度的。

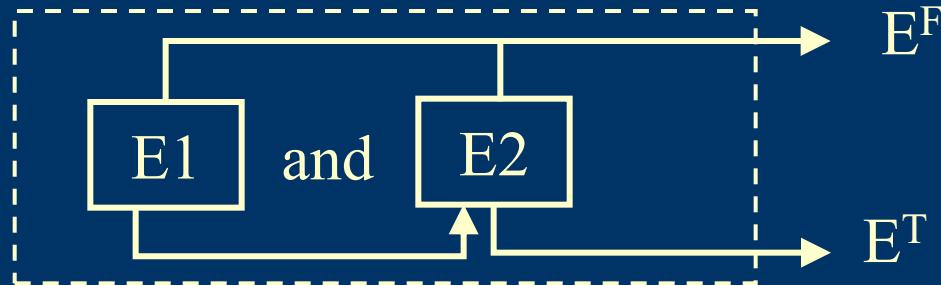
常见的编译优化的种类

- ◆ 表达式短路问题

$E = E1 \text{ or } E2$



$E = E1 \text{ and } E2$



常见的编译优化的种类

- ◆ 常表达式的节省

在我们的计算过程中有一个表达式是 $3*3.14$, 这个实际上是两个常数, 他的结果我是可以计算出来的, 这样我们在编译的过程中把 $3*3.14$ 算出来, 在目标程序的进行中就不用进行计算了。

- ◆ 公共子表达式节省(消除重复操作)

- $t = b*c; e = \underline{b*c} + \underline{b*c}; c = \underline{b*c} + 10; d = b*c + d;$
- $a[i][j] + a[i][k]$

常见的编译优化的种类

- ◆ 循环不变量式提

- `while (k<10) {a[k] = b*c; k=k+1}`
- `t = b*c; while(k<10) {a[k]=t; k=k+1;}`

如果有表达式的值在循环中不会改变，就需要把它提到循环体外面，可以大大提高目标程序的运行效率

常见的编译优化的种类

- ◆ 削减程序的运算强度
- ◆ 是指用强度低的运算代替强度大的运算，通常也是针对循环的。
 - `for j:=1 to 100 do A[j]:=3*j+10;`
 - `m=13; for j:=1 to 100 do { A[j]:=m; m:=m+3; }`

常见的编译优化的种类

- ◆ 消除无用语句、消除冗余代码。

- if (E1恒等于true) S1 else S2
 - S1

- ◆ 寄存器优化

- 涉及到目标程序执行的时候，如何分配目标机的寄存器。

常见的编译优化的种类

- ◆ 中间变量的优化

属于空间上的优化，假如两个临时变量的活动区不相交，则可以共用同一个存储单元。

- ◆ 目标代码优化

通过确定目标代码减少目标程序指令个数来提高执行效率。譬如两个运算分量都在寄存器中可以直接参与计算，不需要将其存入内存后导出计算。

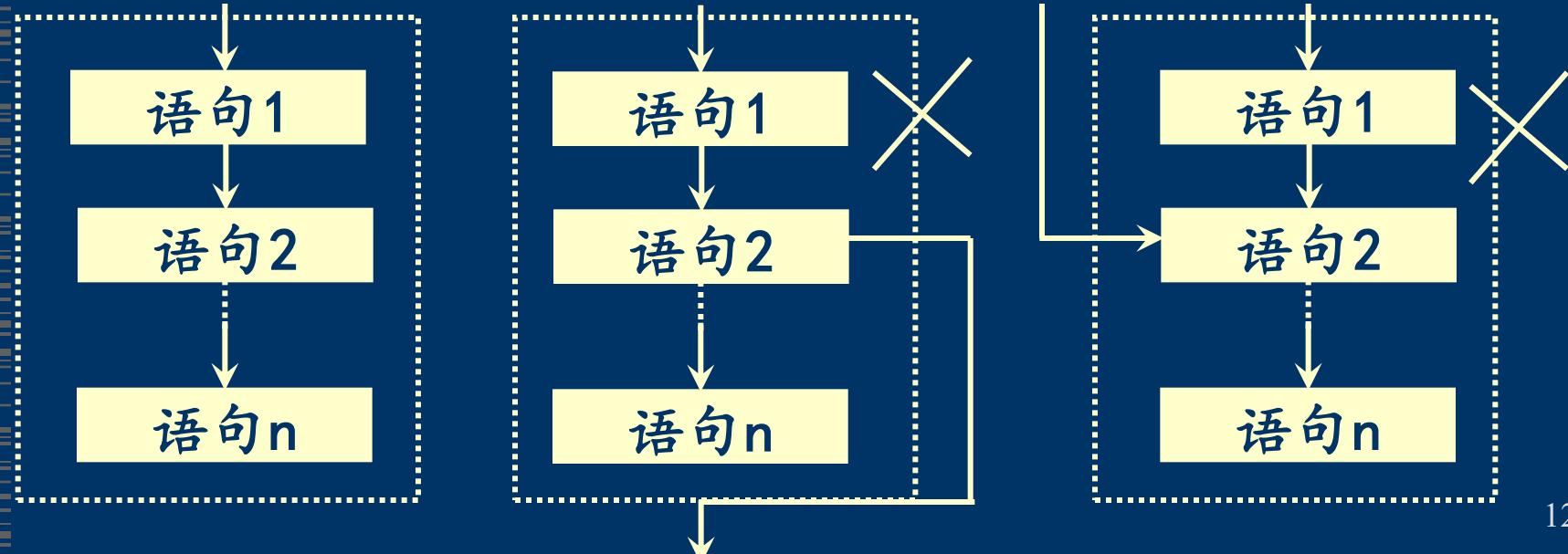
常见的编译优化的种类

- ◆ 全局优化

对程序全局进行数据流分析，优化技术比较复杂，会导致编译代价很大，优化的效果也不是十分明显，只有在特殊需求的情况下才要进行。

基本块的定义

- ◆ 基本块是指程序的一组顺序执行的语句序列，其中只有一个出口和一个入口。
 - 入口：基本块的第一条语句；
 - 出口：基本块的最后一条语句；



- 对于一个基本块而言，执行时只能从它的入口进入，从出口退出；
- 一个基本块内部的语句要么全执行，要么全不执行，不能执行其中的一部分，不能在中间转出，也不能从中间转入。
- 基本块可以基于源代码、中间代码和目标代码。

基本块的划分原则

- ◆ 整个四元式序列的第一个四元式为基本块的入口四元式
- ◆ 遇转移性四元式时，结束当前基本块，并把该四元式作为当前基本块的出口，下一条四元式作为新基本块的入口
- ◆ 遇标号性四元式时结束当前基本块，四元式本身作为新基本块的入口
- ◆ 遇到对地址引用型变量赋值四元式时，结束当前基本块，并作为该块的出口。

基本块的划分原则

- ◆ 转移性四元式是指在生成目标代码时一定产生跳转指令的四元式。例如：

- (**JMP** , — , — , L)
- (**JMP1** , E , — , L)
- (**JMP0** , E , — , L)
- (**ENDPROC** , — , — , —)
- (**ENDFUNC** , — , — , —)
- (**THEN** , E , — , —)
- (**ELSE** , — , — , —)
- (**DO** , E , — , —)
- (**ENDWHILE** , — , — , —)

基本块的划分原则

- ◆ 标号性四元式也称定位性四元式，起到一个定位的作用不产生跳转指令，例如：
 - (**LABEL** , — , — , L)
 - (**ENTRY** , Label , Size , Level)
 - (**WHILE** , — , — , —)
 - (**ENDIF** , — , — , —)

基本块划分的例子

设有源程序如下：

```
y := 1 ;
L: if A and B
    then x := 0
    else y := 0 ;
x := x + 1 ;
y := y - 1 ;
while x + y > 0
    do x := x - 1 ;
z := 0 ;
```

注：x, y为非引用型整数
类型形参。

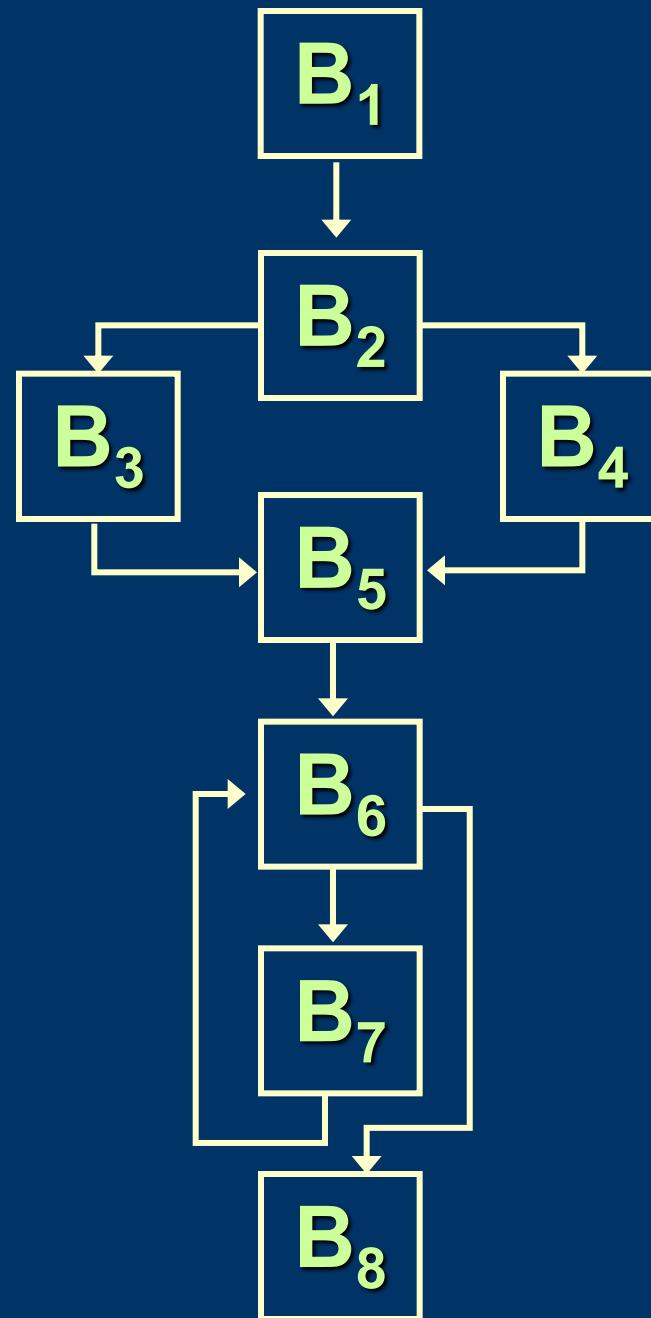
(ASSIGN, 1, _, y)	(WHILE, -, -, -)
(LABEL, _, _, L)	(ADDI, x, y, t ₃)
(AND, A, B, t ₀)	(GT, t ₃ , 0, t ₄)
(THEN, t ₀ , -, -)	(DO, t ₄ , -, -)
(ASSIG, 0, -, x)	(-, x, 1, t ₅)
(ELSE, -, -, -)	(ASSIG, t ₅ , x)
(ASSIG, 0, _, y)	(ENDWHILE, -, -, -)
(ENDIF, -, -, -)	(ASSIGN, 0, _, z)
(+, x, 1, t ₁)	
(ASSIG, t ₁ , _, x)	
(-, y, 1, t ₂)	
(ASSIGN, t ₂ , _, y)	

基本块划分的例子

B ₁	(ASSIG,1,-,y)	B ₆	(SUBI,y,1,t ₂)
B ₂	(LABEL,-,-,L)		(ASSIG,t ₂ ,-,y)
	(AND,A,B,t ₀)	B ₆	(WHILE,-,-,-)
	(THEN,t ₀ ,-, -)		(ADDI,x,y,t ₃)
B ₃	(ASSIG,0,-,x)		(GT,t ₃ ,0,t ₄)
	(ELSE,-,-,-)	B ₇	(DO,t ₄ ,-, -)
B ₄	(ASSIG,0,-,y)		(SUBI,x,1,t ₅)
B ₅	(ENDIF,-,-,-)		(ASSIG,t ₅ ,-,x)
	(ADDI,x,1,t ₁)		(ENDWHILE,-,-,-)
	(ASSIG,t ₁ ,-,x)	B ₈	(ASSIG,0,-,z)

程序流图

- 程序流图是以基本块为节点的有向图。



常表达式节省

- ◆ 常表达式：在编译过程中能够计算出常量值的表达式
- ◆ 处理思想：针对每个基本块，如果一个多元式的两个分量的值已知，则计算其值，并删掉相应的中间代码。

常表达式节省

例：假设有下列语句：

a := m + 10 ;

b := a + m ;

c := a + b - d ;

并假设当执行第一个语句时，m总是取常数10，则上列语句可优化如下：

a := 20 ; b := 30 ; c := 50 - d ;

原理：

常量定值表ConstDef：表元素为二元组 (Var, Val) 。如果在ConstDef中有元素 (V, c) ，表示变量此时一定取常数值 c ，在 V 被更改之前出现的 V 均可替换成 c 。

基本块上常量表达式的局部优化算法：

- ① 基本块入口置ConstDef为空；
- ② 读当前四元式；
- ③ 对当前四元式的分量利用ConstDef表进行值代换得新四元式newtuple；
- ④ 如果新多元式newtuple 形如 (ω , A, B, t)：若A和B是常数，则计算 $A \omega B$ 的值v，并将 (t, v) 填入ConstDef表。删除当前四元式。
- ⑤ 如果新多元式newtuple形如 (ASSIG, A, -, B)：如果A是常数，则把 (B, A) 填入ConstDef表，若已有B项，只需修改其值；否则 (A为非常数) 从ConstDef中删除B的登记项。
- ⑥ 重复②~⑤直到基本块结束。

常表达式局部优化的例子

源程序	中间代码	ConstDef	优化后的代码
a:=1	(ASSIGN, 1, a)	(a, 1)	(ASSIGN, 1, a)
b := a+1	(ADDI, a, 1, t1)	(a, 1) (t1, 2)	()
	(ASSIGN, t1, b)	(a, 1) (t1, 2) (b, 2)	(ASSIGN, 2, b)
a:=x	(ASSIGN, x, a)	(t1, 2) (b, 2)	(ASSIGN, a, x)
c := b+5	(ADDI, b, 5, t2)	(t1, 2) (b, 2) (t2, 7)	()
	(ASSIGN, t2, c)	(t1, 2) (b, 2) (t2, 7) (c, 7)	(ASSIGN, 7, c)

习题

i=1 ;

j=i*(i+1) ;

k=2*(i+j) ;

j=n;

k=k+j;