第8章复习笔记

基本块和流图

中间代码划分成为基本块(basic block)

控制流只能从第一个指令进入，除基本块的最后一个指令外，控制流不会跳转/停机

结点：基本块 边：指明了基本块的执行顺序

流图可以作为优化的基础，指出了基本块之间的控制流，可以根据流图了解到一个值是否会被使用等信息

划分基本块的算法

确定基本块的首指令：

1）第一个三地址指令

2）任意一个条件或无条件转移指令的目标指令

3）紧跟在一个条件/无条件转移指令之后的指令

确定基本块：每个首指令对应于一个基本块：从首指令开始到下一 个首指令

确定基本块中的活跃性、后续使用

变量值的使用（use）与活跃（live）

假设三地址语句 i 给 x 赋了一个值，如果在后续的另一语句 j 的一个运算分量为 x，且从 i 开始有一条没有对 x 进行赋值的路径能够到达 j， 那么 j 就使用了 i 处计算得到的x的值。我们可以说称 x 在语句 i 后的程序点上活跃 （live）

即在程序执行完语句i的时刻，x 中存放的值将被后面的语句使用

不活跃是指变量中存放的值不会被使用，而不是变量不会被使用

- 输入：基本块B，开始时B的所有非临时变量都是活跃的
- 输出：各语句i上的变量的活跃性、后续使用信息
- 方法：
  - 从B的最后一个语句开始反向扫描
  - 对于每个语句i：x=y+z
    1. 令语句i和x、y、z的当前活跃性信息/使用信息关联
    2. 设置x为不活跃、无后续使用
    3. 设置y和z为活跃，并指明它们的下一次使用为语句i
  - 注意：2和3的顺序

流图中的循环

循环的定义：循环L是一个结点集合

存在一个循环入口（loop entry）结点，是唯一的、前驱可以在循环L之外的结点，到达其余结点的路径必然先经过这个入口结点。其余结点都存在到达入口结点的非空路径， 且路径都在L中

基本块的DAG表示

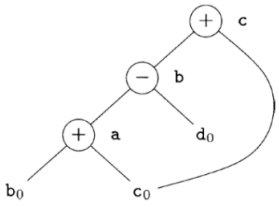
- 基本块可以用DAG表示
  - 每个变量有对应的DAG的结点，代表初始值
  - 每个语句s有一个相关的结点N，代表计算得到的值
    - N的子结点对应于（其运算分量当前值的）其它语句
    - 结点N的标号是s的运算符
    - N和一组变量关联，表示s是在此基本块内最晚对它们定值的语句
- 输出结点：结点对应的变量在基本块出口处活跃
- 从DAG，我们可以知道各个变量最后的值和初始值的关系

构造过程

为基本块中出现的每个变量建立结点（表示初始值），各变量和相应结点关联

顺序扫描各个三地址指令，进行如下处理

- 如果指令为x=y op z
  - 为这个指令建立结点N，标号为op
  - N的子结点为y、z当前关联的结点
  - 令x和N关联
- 如果指令为x=y
  - 不建立新结点
  - 设y关联到N，那么x现在也关联到N



扫描结束后，对于所有在出口处活跃的变量x，将x所关联的结点设置为输出结点

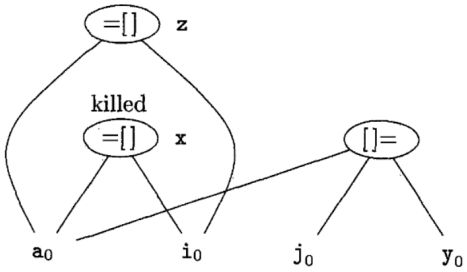
数组引用

- 数组引用实例
  - x=a[i]
  - a[j]=y
  - z=a[i]

  } a[i]是公共子表达式吗？

- 注意：a[j]可能改变a[i]的值，因此不能和普通的运算符一样构造相应的结点
- 数组引用的DAG表示
  - 从数组取值的运算x=a[i]对应于=[]的结点
    - x作为这个结点的标号之一
    - 该节点的左右子节点分别代表数组初始值和下标
  - 对数组赋值（例如a[j]=y）的运算对应于[]=的结点，没有关联的变量、且杀死所有依赖于a的变量
    - 三个子节点分别表示a_0、j和y
  - 数组赋值隐含的意思：将数组作为整体考虑，对数组元素赋值即改变整个数组

- 基本块
  - x = a[i]
  - a[j] = y
  - z = a[i]



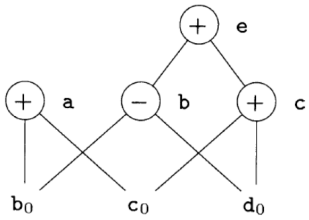
DAG的作用

消除局部公共子表达式

消除死代码

- 在DAG上消除没有附加活跃变量的根结点（没有父结点的结点），即消除死代码
- 如果图中c、e不是活跃变量，则可以删除标号为e、c的结点

死代码=？不可达代码



对语句重新排序

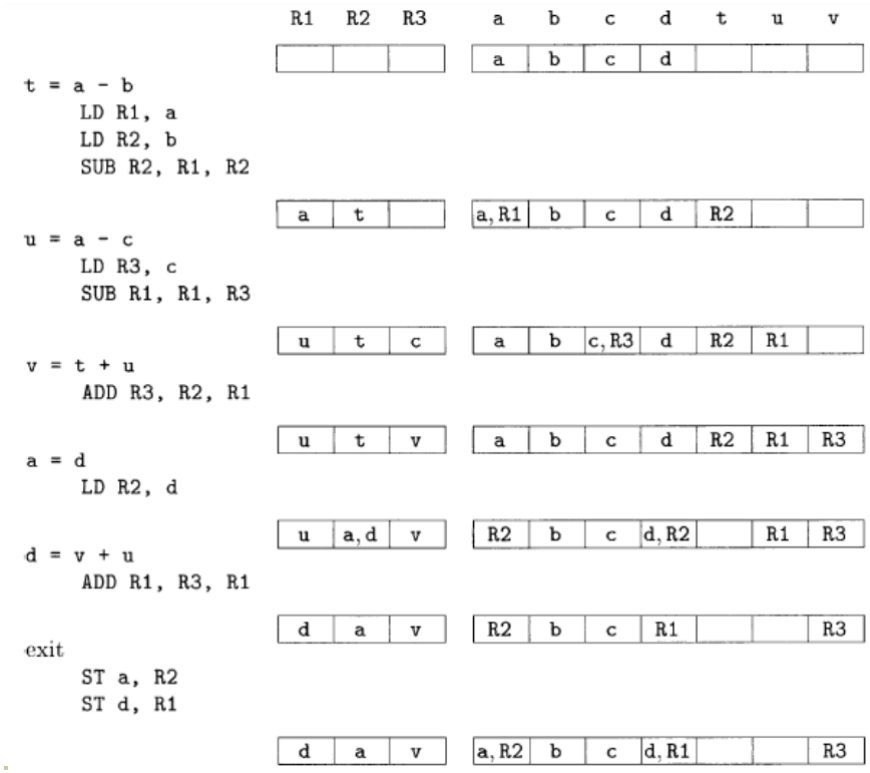
对运算分量的顺序进行重排

重构的方法：每个结点构造一个三地址语句，计算对应的值；结果应该尽量赋给一个活跃的变量；如果结点有多个关联的变量，则需要用复制语句进行赋值

- 重组时应该注意求值的顺序
  - 指令的顺序必须遵守DAG中结点的顺序
  - 对数组的赋值必须跟在所有原来在它之前的赋值/求值操作之后
  - 对数组元素的求值必须跟在所有原来在它之前的赋值指令之后
  - 对变量的使用必须跟在所有原来在它之前的过程调用和指针间接赋值之后
  - 任何过程调用或者指针间接赋值必须跟在原来在它之前的变量求值之后
- 总的来说，我们必须保证：如果两个指令之间可能相互影响，那么他们的顺序就不应该改变

寄存器指派

寄存器描述符和地址描述符


R1 R2 R3
a b c d t u v
t = a - b
LD R1, a
LD R2, b
SUB R2, R1, R2
a t
a, R1 b c d R2
u = a - c
LD R3, c
SUB R1, R1, R3
u t c
a b c, R3 d R2 R1
v = t + u
ADD R3, R2, R1
u t v
a b c d R2 R1 R3
a = d
LD R2, d
u a, d v
R2 b c d, R2 R1 R3
d = v + u
ADD R1, R3, R1
d a v
R2 b c R1 R3
exit
ST a, R2
ST d, R1
d a v
a, R2 b c d, R1 R3