机器无关代码优化
Part2：数据流与到达定值分析

李 诚

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2024年11月13日

本节提纲

程序员编写的源程序

前端：词法分析 → 语法分析 → 语义分析

→ 中端：中间代码生成 → 机器无关代码优化

→ 后端：指令选择 → 指令调度 → 寄存器分配

→ 机器硬件上运行的目标代码

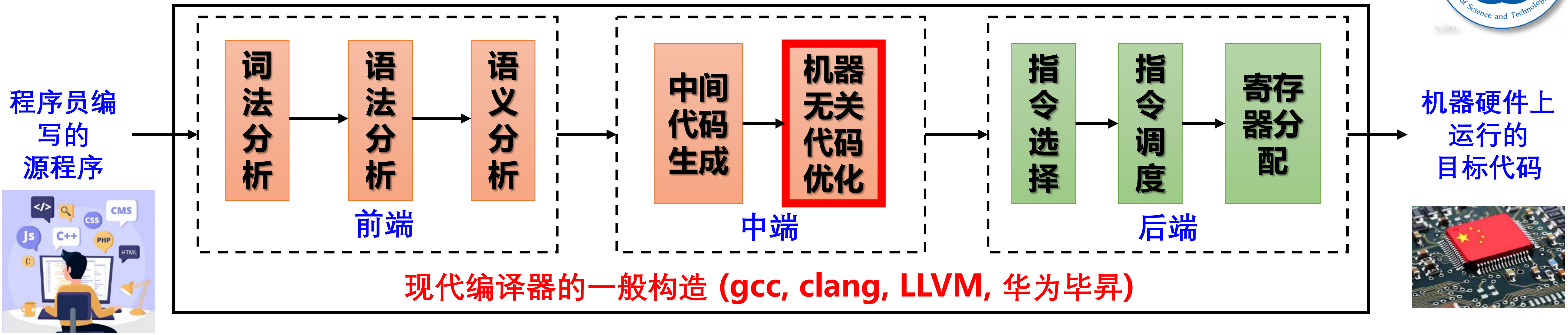
现代编译器的一般构造 (gcc, clang, LLVM, 华为毕昇)

- 代码优化的主要实现方式
- 数据流分析概述
- 数据流分析理论框架
- 到达-定值分析算法

优化的实现方式

- **全局视角-跨基本块的优化**
  - 数据流分析
- **局部视角-基本块的优化**
  - DAG表示

本节提纲



- 代码优化的主要实现方式
- **数据流分析概述**
- **数据流分析理论框架**
- **到达-定值分析算法**

数据流分析

- **Data-flow analysis**
  - 一组用来获取程序执行路径上的数据流信息的技术
- **数据流分析应用**
  - 到达-定值分析(Reaching-Definition Analysis)
  - 活跃变量分析(Live-Variable Analysis)
  - 可用表达式分析(Available-Expression Analysis)
- **在每一种数据流分析应用中，都会把每个程序点和一个数据流值关联起来**

数据流抽象

- **流图上的点(程序点)**
  - 基本块中，两个相邻的语句之间为程序的一个点
  - 基本块的开始点和结束点

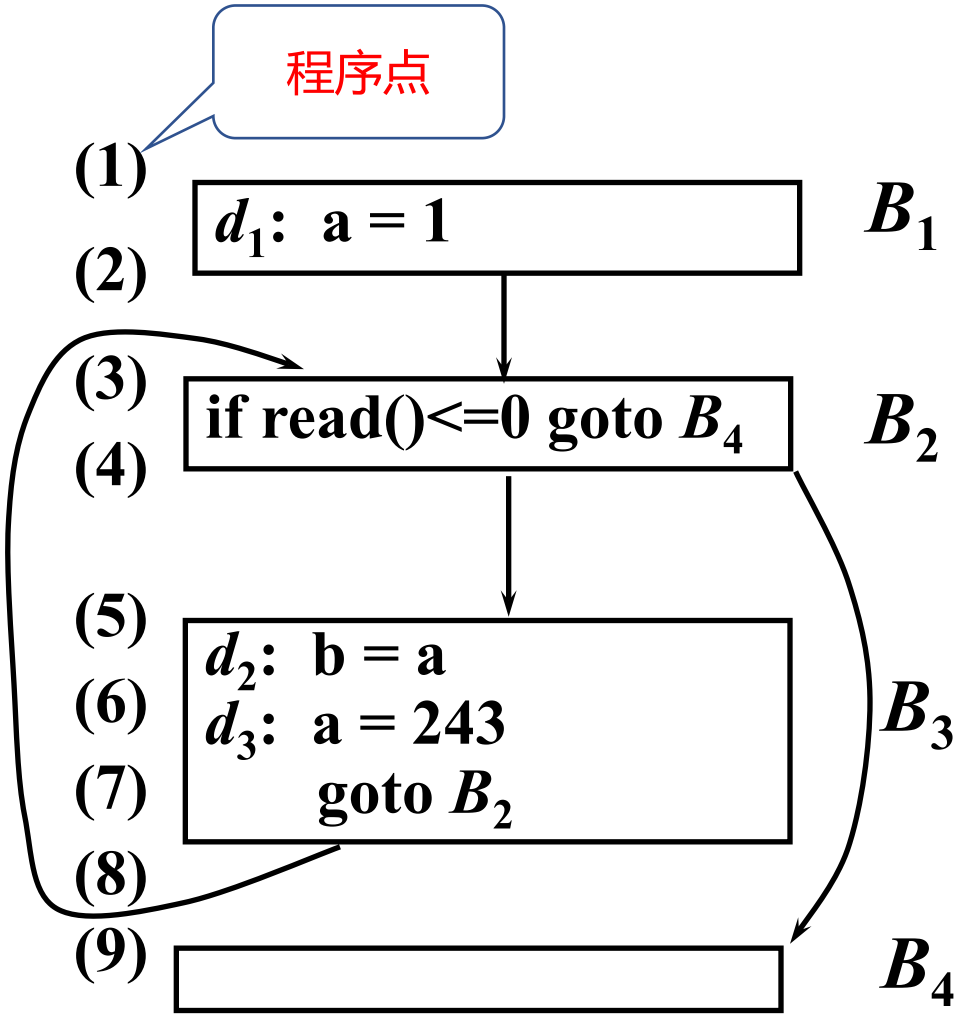
- **流图上的路径**
  - 点序列$p_1, p_2, \ldots, p_n$，对1和n - 1间的每个i，满足

(1) p_i是先于一个语句的点，p_{i+1}是同一块中位于该语句后的点，或者

(2) p_i是某块的结束点，p_{i+1}是后继块的开始点

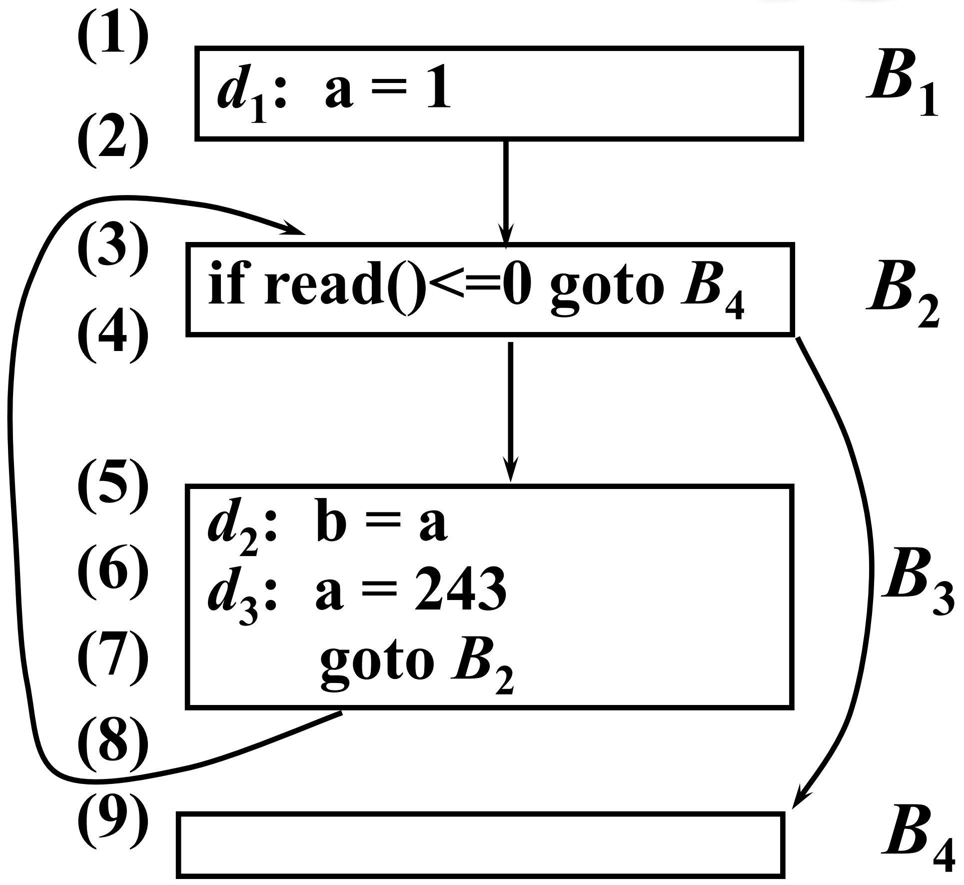
数据流抽象

- 流图上路径实例
  - (1, 2, 3, 4, 9)
  - (1, 2, 3, 4, 5, 6, 7, 8, 3, 4, 9)
  - (1, 2, 3, 4, 5, 6, 7, 8,
    3, 4, 5, 6, 7, 8, 3, 4, 9)
  - (1, 2, 3, 4, 5, 6, 7, 8,
    3, 4, 5, 6, 7, 8,
    3, 4, 5, 6, 7, 8, …)
  - … …



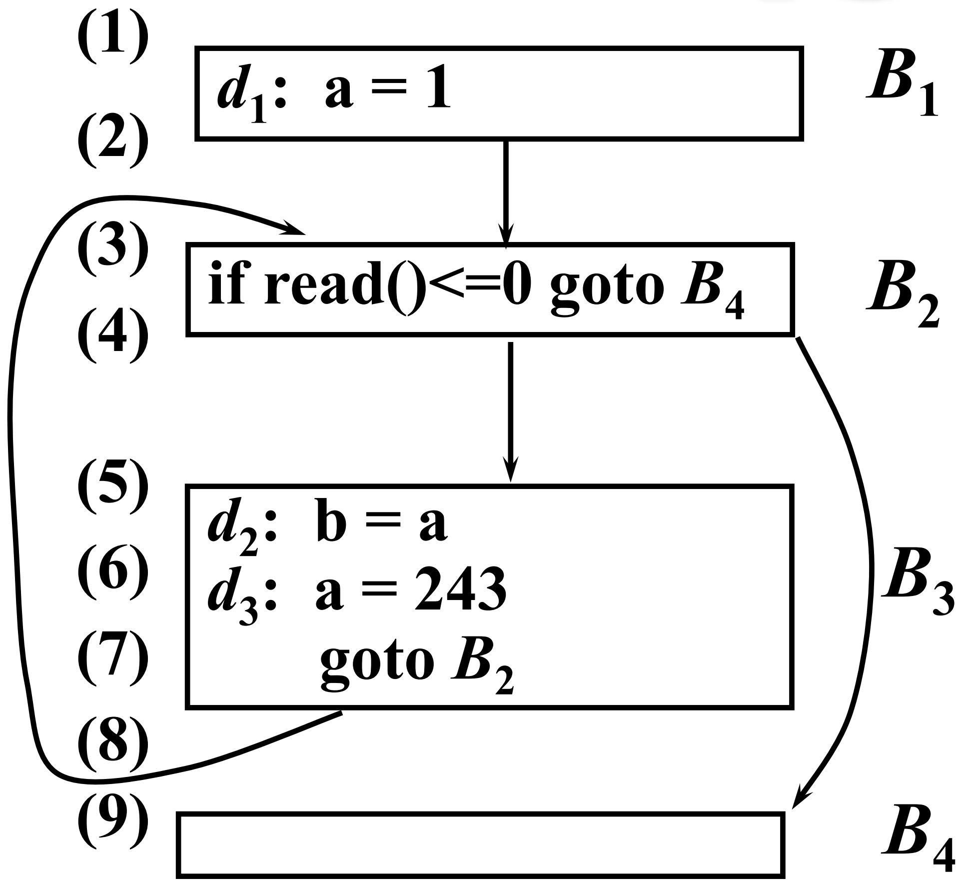
数据流分析介绍

- **分析程序的行为时，必须在其流图上考虑所有的执行路径（在调用或返回语句被执行时，还需要考虑执行路径在多个流图之间的跳转）**
  - 通常，从流图得到的程序执行路径数无限，且执行路径长度没有有限的上界

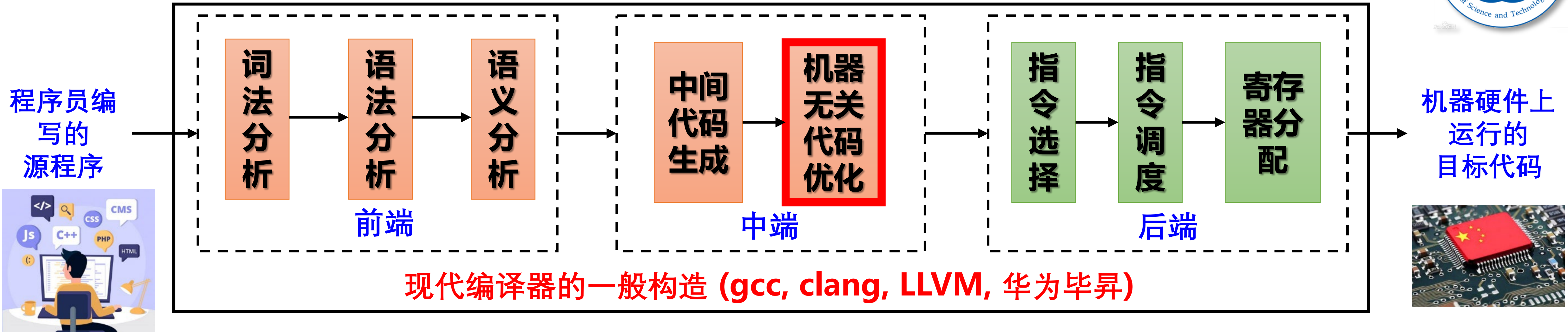


数据流分析介绍

- **分析程序的行为时，必须在其流图上考虑所有的执行路径（在调用或返回语句被执行时，还需要考虑执行路径在多个流图之间的跳转）**
  - 通常，从流图得到的程序执行路径数无限，且执行路径长度没有有限的上界
  - 每个程序点的不同状态数也可能无限 (程序状态：存储单元到值的映射)



本节提纲



- 代码优化的主要实现方式
- 数据流分析概述
- **数据流分析理论框架**
- **到达-定值分析算法**

数据流分析模式

- **数据流值代表在任一程序点能观测到的所有可能程序状态集合的一个抽象**
- **对于一个语句*s***
  - *s*之前的程序点对应的数据流值用**IN[*s*]**表示
  - s之后的程序点对应的数据流值用**OUT[*s*]**表示
- **对于一个基本块呢?**

数据流分析模式

- **传递函数(transfer function)** f
  - 语句前后两点的数据流值受该语句的语义约束
  - 若沿执行路径正向传播，则$\text{OUT}[s] = f_s(\text{IN}[s])$
  - 若沿执行路径逆向传播，则$\text{IN}[s] = f_s(\text{OUT}[s])$

若基本块B由语句$s_1, s_2, \ldots, s_n$依次组成，则

- $\text{IN}[s_{i+1}] = \text{OUT}[s_i],\ i = 1, 2, \ldots, n-1$

考虑的是在语句执行后输入输出之间的变化关系

基本块上的数据流模式

- IN[B]：紧靠基本块B之前的数据流值
  - $\mathrm{IN}[B] = \mathrm{IN}[s_1]$
- OUT[B]：紧靠基本块B之后的数据流值
  - $\mathrm{OUT}[B] = \mathrm{OUT}[s_n]$
- f_B :基本块B的传递函数
  - 前向数据流： $\mathrm{OUT}[B] = f_B(\mathrm{IN}[B])$
    - $f_B = f_n \circ \ldots \circ f_2 \circ f_1$
  - 逆向数据流： $\mathrm{IN}[B] = f_B(\mathrm{OUT}[B])$
    - $f_B = f_1 \circ \ldots \circ f_{n-1} \circ f_n$

基本块间的数据流分析模式

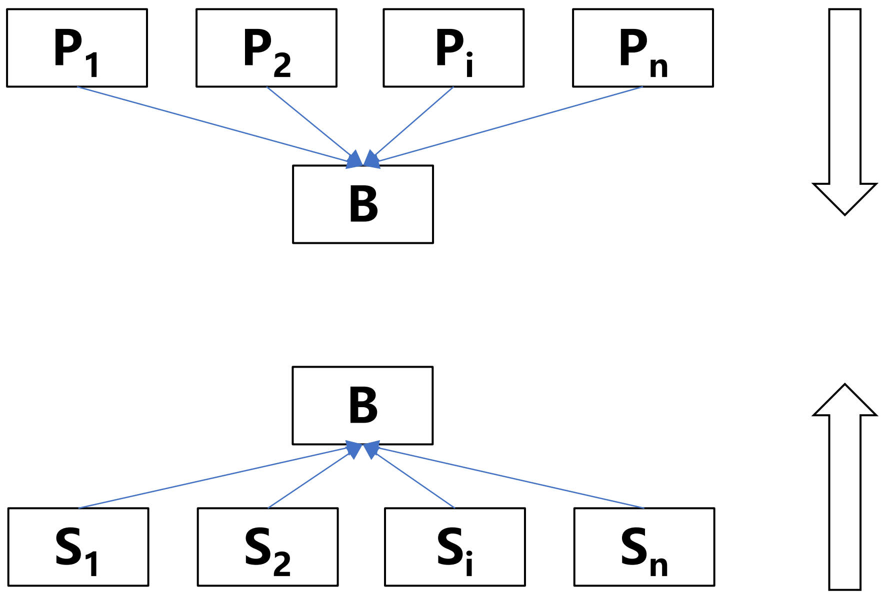
- **控制流约束**
  - 正向传播

U 是汇合的意思，并不一定代表并集，也可能是交集等运算

$$\mathrm{IN}[B] = \cup_{P\text{是}B\text{的前驱}} \mathrm{OUT}[P]$$

  - 逆向传播

$$\mathrm{OUT}[B] = \cup_{S\text{是}B\text{的后继}} \mathrm{IN}[S]$$



- **约束方程组的解通常不是唯一的**
  - 求解的目标是要找到满足这两组约束（控制流约束和迁移约束）的最“精确”解

考虑的是在其他语句或块对于输入的影响和本次执行的输出对其他语句和块的影响

三种有用的分析

- **到达-定值**
  - 应用场景：检测未定义的变量
- **可用表达式**
  - 应用场景：检测全局公共子表达式
- **活跃变量**
  - 应用场景：可用于寄存器分配

本节提纲

程序员编写的源程序

词法分析 → 语法分析 → 语义分析

前端

中间代码生成 → 机器无关代码优化

中端

指令选择 → 指令调度 → 寄存器分配

后端

现代编译器的一般构造 (gcc, clang, LLVM, 华为毕昇)

机器硬件上运行的目标代码

- 代码优化的主要实现方式
- 数据流分析概述
- 数据流分析理论框架
- **到达-定值分析算法**

到达-定值分析

- **到达一个程序点的所有定值(gen)**
- **定值的注销(kill)**
  - 在一条执行路径上，对x的赋值注销先前对x的所有赋值
- **别名给到达-定值的计算带来困难，因此，本章其余部分仅考虑变量无别名的情况**

到达-定值分析

- **定值与引用**

d：　x := y + z　// 语句d 是变量x的一个定值点

ud链，即引用一定值链。
流图中有路径d---->u，
且该路径上x在d处的定
值没有被杀死。

u：　w :=　x + v // 语句u 是变量x的一个引用点

- **变量x在d点的定值到达u点**

到达-定值分析的用途

- **循环不变计算的检测**
  - 如果循环中含有赋值x=y+z，而y和z所有可能的定值都在循环外，那么y+z就是循环不变计算
- **常量合并**
  - 如果对变量x的某次使用只有一个定值到达，且该定值把一个常量赋给x，则可以用该常量替换x
- **错误检测**
  - 判定变量x在p点上是否未经定值就被引用

到达–定值分析

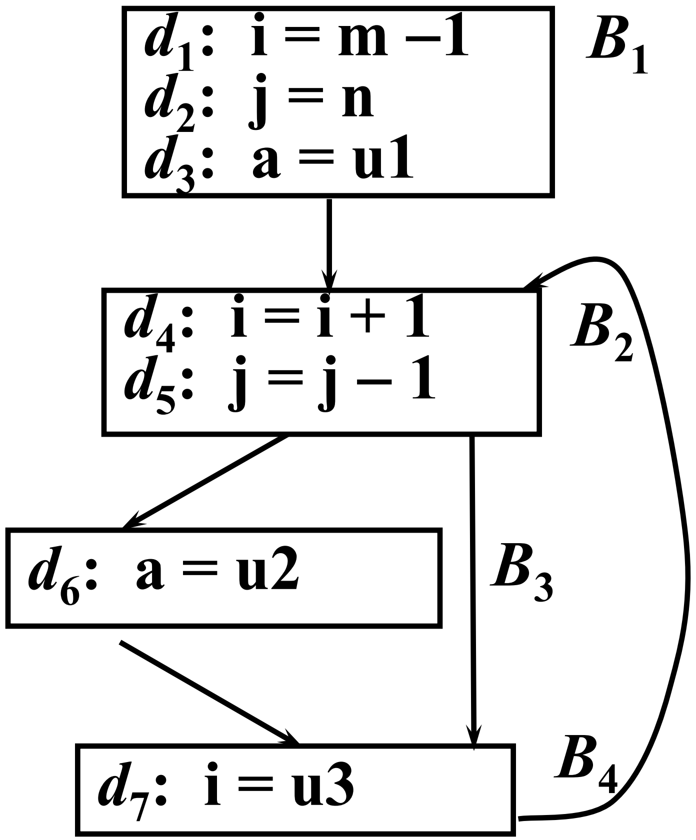
- *gen*和*kill*分别表示一个基本块生成和注销的定值

$$gen\ [B_1] = \{d_1, d_2, d_3\}$$
$$kill\ [B_1] = \{d_4, d_5, d_6, d_7\}$$

$$gen\ [B_2] = \{d_4, d_5\}$$
$$kill\ [B_2] = \{d_1, d_2, d_7\}$$

$$gen\ [B_3] = \{d_6\}$$
$$kill\ [B_3] = \{d_3\}$$

$$gen\ [B_4] = \{d_7\}$$
$$kill\ [B_4] = \{d_1, d_4\}$$



到达-定值分析

- **基本块的*gen*和*kill*是怎样计算的**
  - 对三地址指令 d: u = v + w，它的状态传递函数是
    $$f_d(x) = gen_d \cup (x - kill_d)$$
  - 若：$f_1(x) = gen_1 \cup (x - kill_1)$, $f_2(x) = gen_2 \cup (x - kill_2)$
    则：$f_2(f_1(x)) = gen_2 \cup (gen_1 \cup (x - kill_1) - kill_2)$
    $= (gen_2 \cup (gen_1 - kill_2)) \cup (x - (kill_1 \cup kill_2))$
  - 若基本块B有n条三地址指令
    $$f_B(x) = gen_B \cup (x - kill_B)$$
    $$kill_B = kill_1 \cup kill_2 \cup \ldots \cup kill_n$$
    $$gen_B = gen_n \cup (gen_{n-1} - kill_n) \cup (gen_{n-2} - kill_{n-1} - kill_n) \cup \ldots \cup (gen_1 - kill_2 - kill_3 - \ldots - kill_n)$$

到达-定值分析

- **到达-定值的数据流等式**
  - gen_B：B中能到达B的结束点的定值语句
  - $kill_B$：整个程序中决不会到达B结束点的定值
  - $\mathrm{IN}[B]$：能到达B的开始点的定值集合
  - $\mathrm{OUT}[B]$：能到达B的结束点的定值集合

两组等式（根据*gen*和*kill*定义IN和OUT）

  - $\mathrm{IN}[B] = \cup_{P是B的前驱} \mathrm{OUT}[P]$
  - $\mathrm{OUT}[B] = gen_B \cup (\mathrm{IN}[B] - kill_B)$
  - $\mathrm{OUT}[\mathrm{ENTRY}] = \varnothing$
- **到达-定值方程组的迭代求解，最终到达不动点**

到达-定值的迭代计算算法

// **正向数据流分析**

引入两个虚拟块：ENTRY、EXIT

(1) OUT[ENTRY] = Ø;

(2) for (除了ENTRY以外的每个块B) OUT[B] = Ø;

(3) while (任何一个OUT出现变化){

(4)   for (除了ENTRY以外的每个块B) {

(5) $IN[B] = \cup_{P是B的前驱} OUT[P]$;

(6) $OUT[B] = gen_B \cup (IN[B] - kill_B)$;

(7) }}

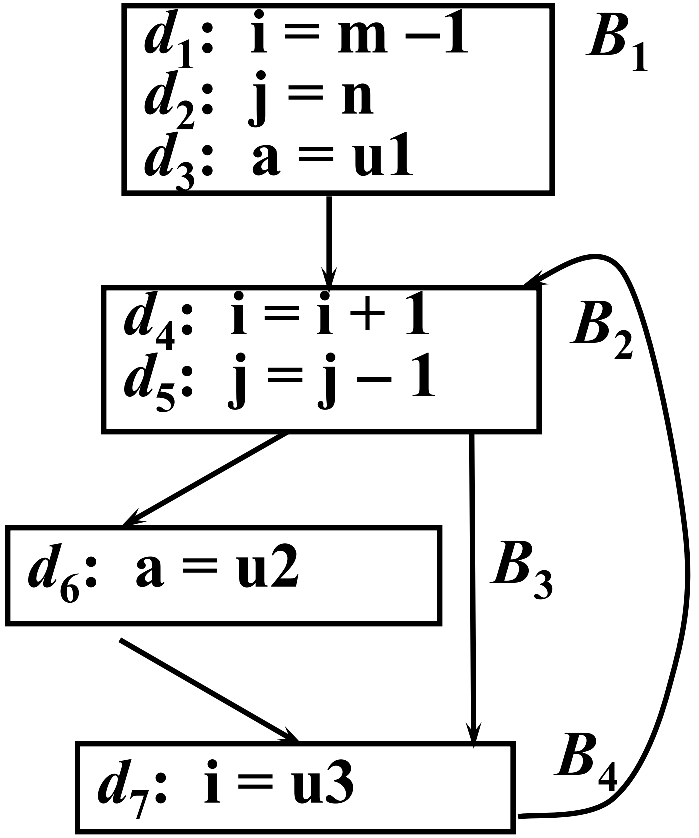
向量求解：集合并操作使用逻辑或，集合相减使用后者求补再逻辑与

到达-定值分析

$IN[B] = \cup_{P是B的前驱} OUT[P]$

$OUT[B] = gen_B \cup (IN[B] - kill_B)$

	IN [B]	OUT [B]
B_1		000 0000
B_2		000 0000
B_3		000 0000
B_4		000 0000



$gen\ [B_1] = \{d_1, d_2, d_3\}$

$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$

$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$

$kill\ [B_3] = \{d_3\}$

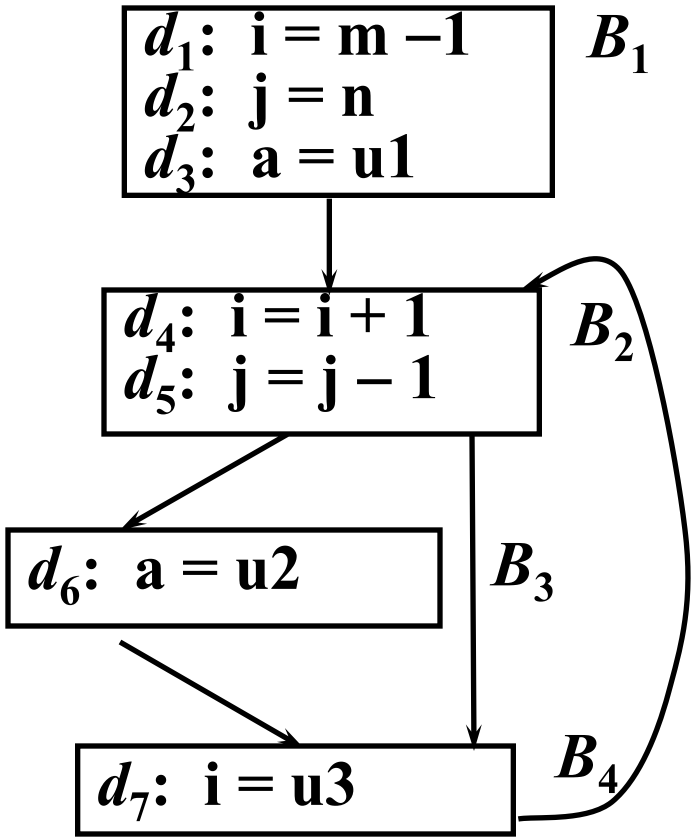
$gen\ [B_4] = \{d_7\}$

$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$$IN[B] = \cup_{P\text{是}B\text{的前驱}} OUT[P]$$
$$OUT[B] = gen_B \cup (IN[B] - kill_B)$$

	IN [B]	OUT [B]
B_1	000 0000	000 0000
B_2		000 0000
B_3		000 0000
B_4		000 0000



$gen\ [B_1] = \{d_1, d_2, d_3\}$
$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$
$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$
$kill\ [B_3] = \{d_3\}$

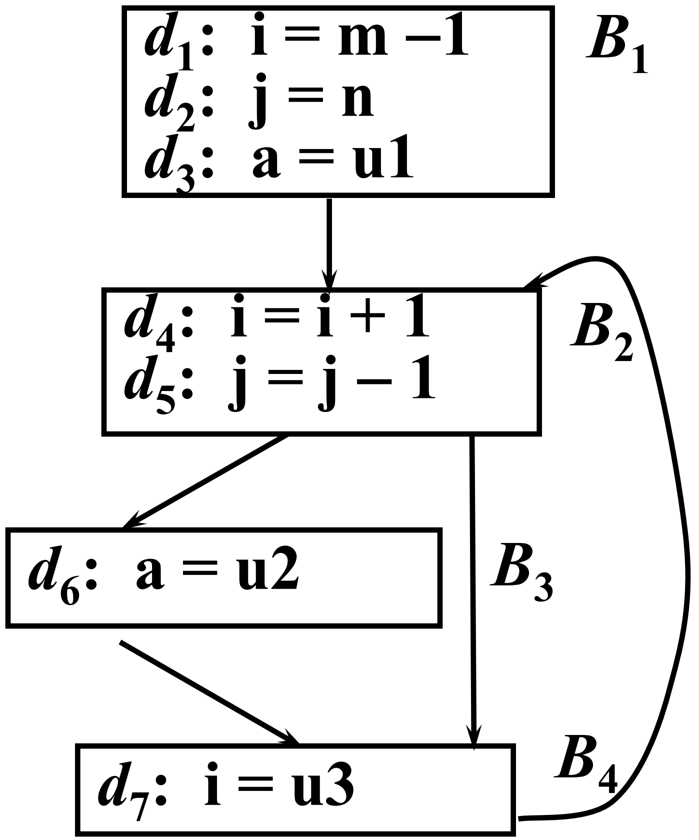
$gen\ [B_4] = \{d_7\}$
$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$\text{IN}[B] = \cup_{P\text{是}B\text{的前驱}} \text{OUT}[P]$

$\text{OUT}[B] = gen_B \cup (\text{IN}[B] - kill_B)$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2		000 0000
B_3		000 0000
B_4		000 0000



$gen\ [B_1] = \{d_1, d_2, d_3\}$

$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$

$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$

$kill\ [B_3] = \{d_3\}$

$gen\ [B_4] = \{d_7\}$

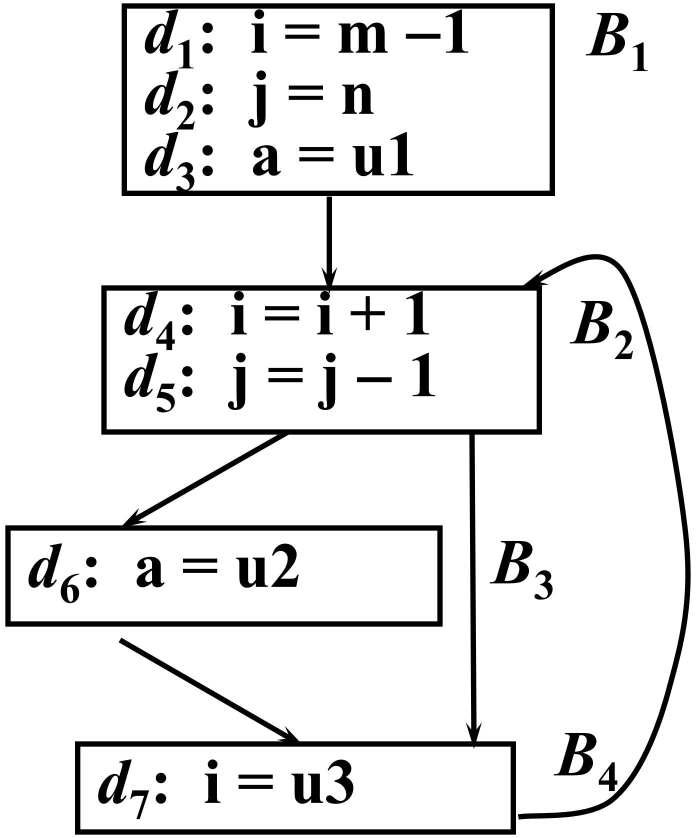
$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$IN[B] = \cup_{P是B的前驱} OUT[P]$

$OUT[B] = gen_B \cup (IN[B] - kill_B)$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0000	000 0000
B_3		000 0000
B_4		000 0000



$gen\ [B_1] = \{d_1, d_2, d_3\}$

$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$

$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$

$kill\ [B_3] = \{d_3\}$

$gen\ [B_4] = \{d_7\}$

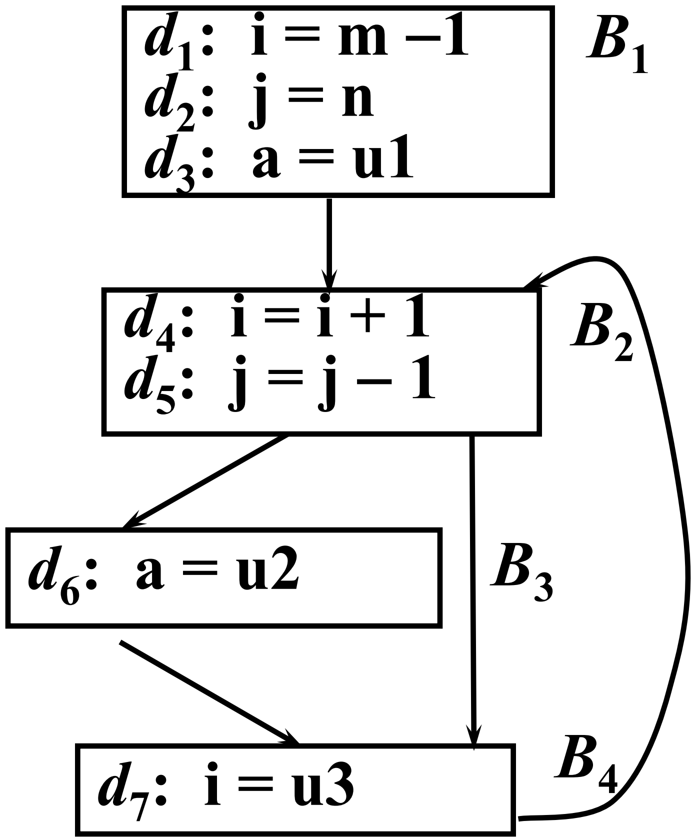
$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$IN[B] = \cup_{P是B的前驱} OUT[P]$

$OUT[B] = gen_B \cup (IN[B] - kill_B)$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0000	001 1100
B_3		000 0000
B_4		000 0000



$gen\ [B_1] = \{d_1, d_2, d_3\}$

$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$

$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$

$kill\ [B_3] = \{d_3\}$

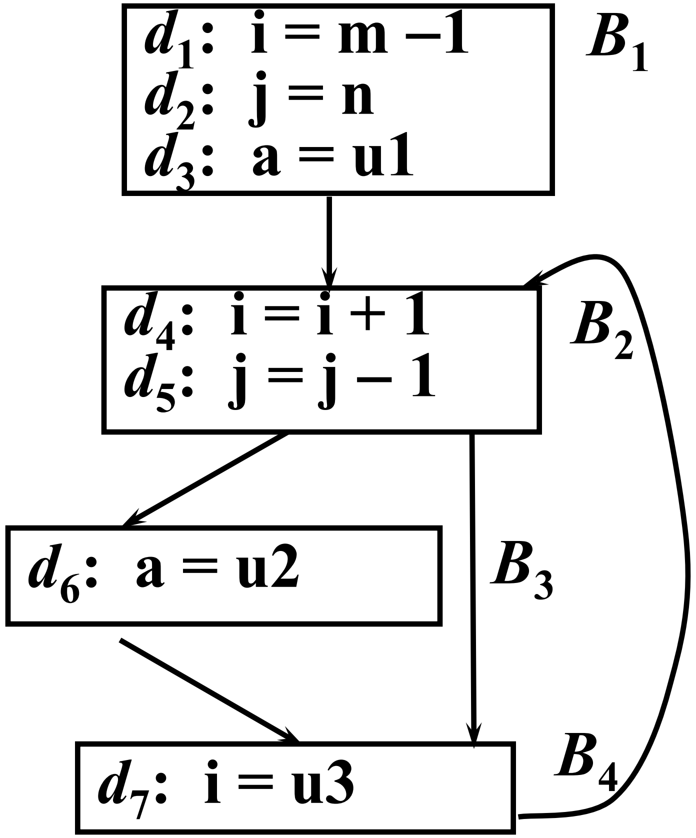
$gen\ [B_4] = \{d_7\}$

$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$$IN[B] = \cup_{P是B的前驱} OUT[P]$$
$$OUT[B] = gen_B \cup (IN[B] - kill_B)$$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0000	001 1100
B_3	001 1100	000 0000
B_4		000 0000



$gen[B_1] = \{d_1, d_2, d_3\}$
$kill[B_1]=\{d_4, d_5, d_6, d_7\}$

$gen[B_2] = \{d_4, d_5\}$
$kill[B_2] = \{d_1, d_2, d_7\}$

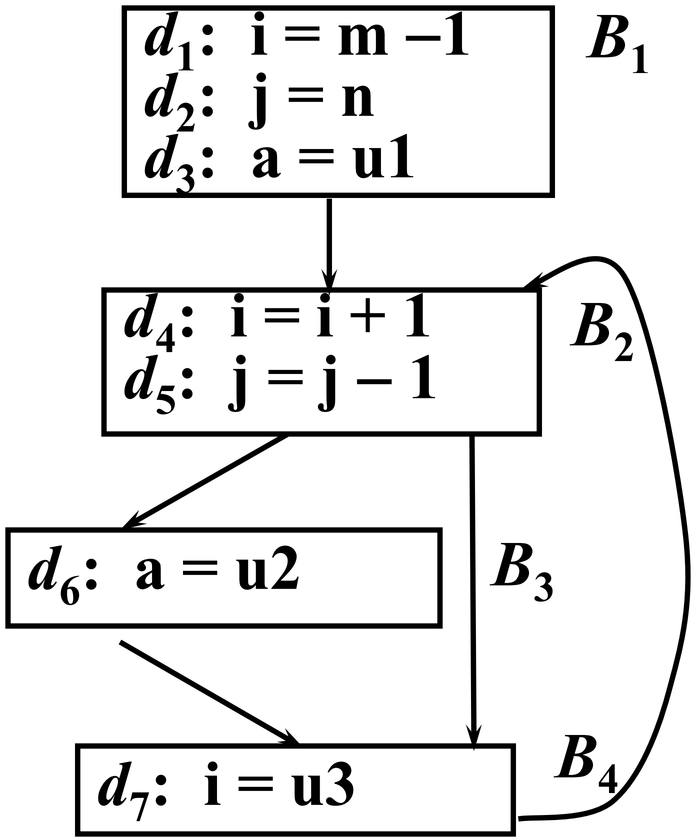
$gen[B_3] = \{d_6\}$
$kill[B_3] = \{d_3\}$

$gen[B_4] = \{d_7\}$
$kill[B_4] = \{d_1, d_4\}$

到达–定值分析

$$IN[B] = \cup_{P\text{是}B\text{的前驱}} OUT[P]$$
$$OUT[B] = gen_B \cup (IN[B] - kill_B)$$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0000	001 1100
B_3	001 1100	000 1110
B_4		000 0000



$gen\ [B_1] = \{d_1, d_2, d_3\}$
$kill\ [B_1] = \{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$
$kill\ [B_2] = \{d_1, d_2, d_7\}$

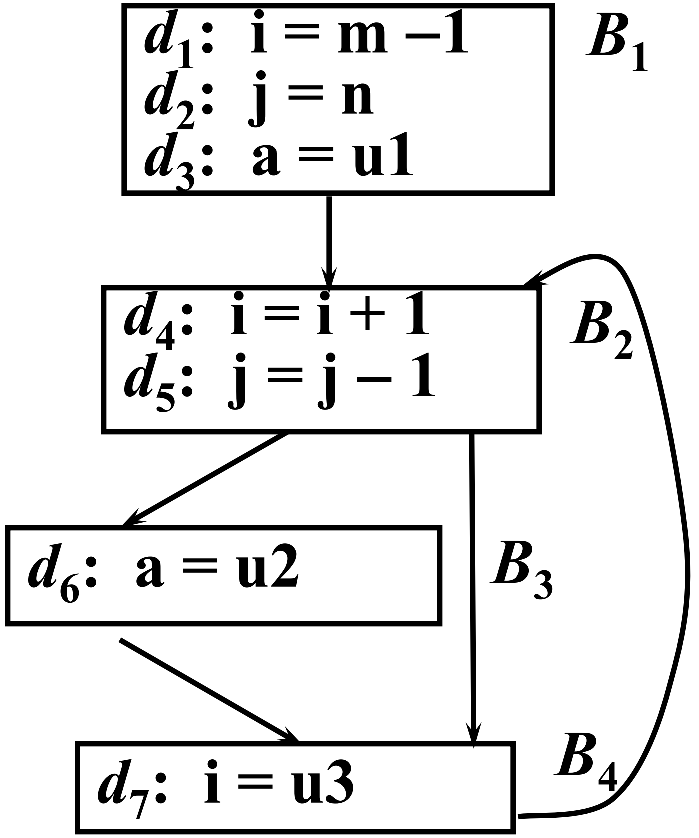
$gen\ [B_3] = \{d_6\}$
$kill\ [B_3] = \{d_3\}$

$gen\ [B_4] = \{d_7\}$
$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$$IN[B] = \cup_{P\text{是}B\text{的前驱}} OUT[P]$$
$$OUT[B] = gen_B \cup (IN[B] - kill_B)$$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0000	001 1100
B_3	001 1100	000 1110
B_4	001 1110	000 0000



$gen\ [B_1] = \{d_1, d_2, d_3\}$
$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$
$kill\ [B_2] = \{d_1, d_2, d_7\}$

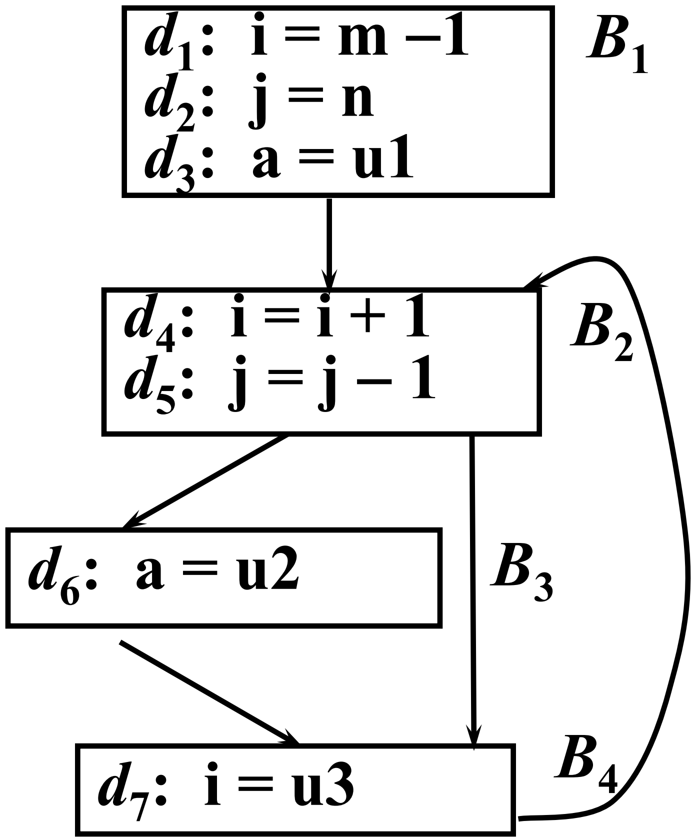
$gen\ [B_3] = \{d_6\}$
$kill\ [B_3] = \{d_3\}$

$gen\ [B_4] = \{d_7\}$
$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$$IN[B] = \cup_{P是B的前驱} OUT[P]$$
$$OUT[B] = gen_B \cup (IN[B] - kill_B)$$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0000	001 1100
B_3	001 1100	000 1110
B_4	001 1110	001 0111



$gen\ [B_1] = \{d_1, d_2, d_3\}$
$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$
$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$
$kill\ [B_3] = \{d_3\}$

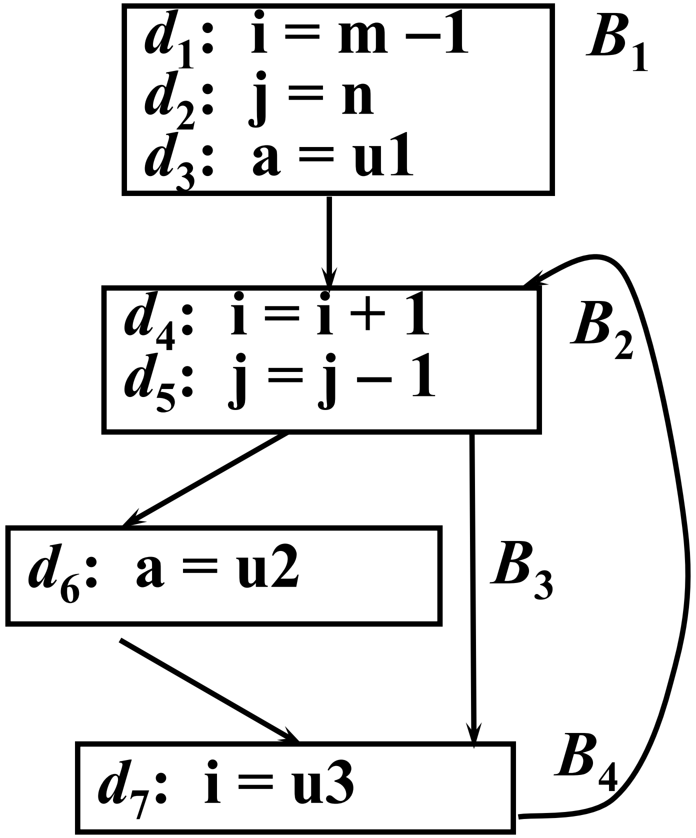
$gen\ [B_4] = \{d_7\}$
$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析

$IN[B] = \cup_{P\text{是}B\text{的前驱}} OUT[P]$

$OUT[B] = gen_B \cup (IN[B] - kill_B)$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0111	001 1100
B_3	001 1100	000 1110
B_4	001 1110	001 0111



$gen\ [B_1] = \{d_1, d_2, d_3\}$

$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$

$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$

$kill\ [B_3] = \{d_3\}$

$gen\ [B_4] = \{d_7\}$

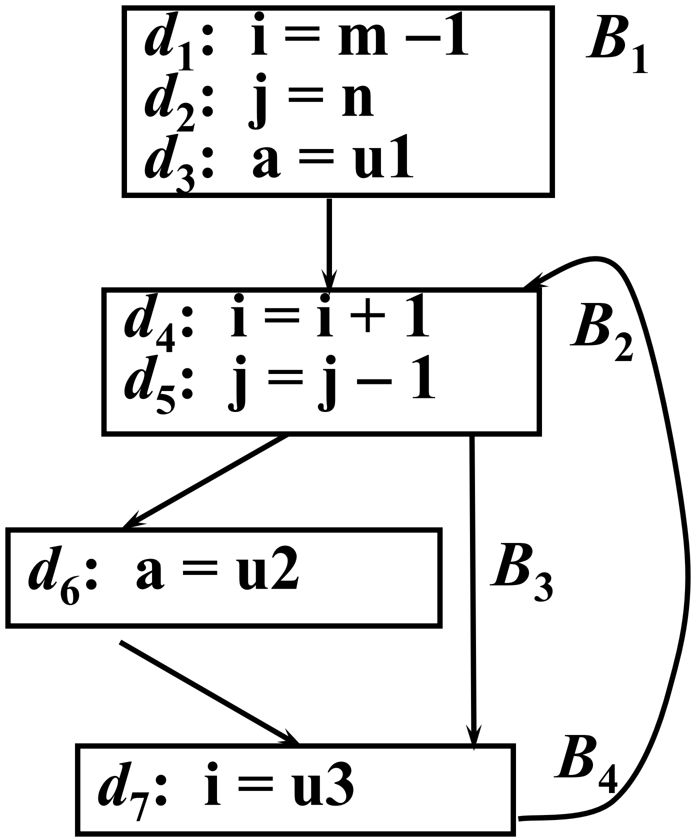
$kill\ [B_4] = \{d_1, d_4\}$

到达–定值分析

$$\mathrm{IN}[B] = \cup_{P\text{是}B\text{的前驱}} \mathrm{OUT}[P]$$
$$\mathrm{OUT}[B] = gen_B \cup (\mathrm{IN}[B] - kill_B)$$

	IN [B]	OUT [B]
B_1	000 0000	111 0000
B_2	111 0111	001 1110
B_3	001 1100	000 1110
B_4	001 1110	001 0111

不再继续演示迭代计算



$gen\ [B_1] = \{d_1, d_2, d_3\}$
$kill\ [B_1]=\{d_4, d_5, d_6, d_7\}$

$gen\ [B_2] = \{d_4, d_5\}$
$kill\ [B_2] = \{d_1, d_2, d_7\}$

$gen\ [B_3] = \{d_6\}$
$kill\ [B_3] = \{d_3\}$

$gen\ [B_4] = \{d_7\}$
$kill\ [B_4] = \{d_1, d_4\}$

到达-定值分析非向量计算方法

- 迭代计算
  - 计算次序，深度优先序，即 B1 -> B2 -> B3 -> B4
  - 初始值：for all B: IN[B] = ∅; OUT[B] = GEN[B]
  - 第一次迭代：

IN[B1] = ∅; // B1 无前驱结点

OUT[B1] = GEN[B1] ∪ (IN[B1]-KILL[B1]) = GEN[B1] = { d1, d2, d3 }

IN[B2] = OUT[B1] ∪ OUT[B4] = {d1, d2, d3 } ∪{ d7 } = { d1, d2, d3, d7 }

OUT[B2] = GEN[B2] ∪ (IN[B2]-KILL[B2]) = { d4, d5 } ∪ { d3 } = { d3, d4, d5 }

IN[B3] = OUT[B2] = { d3, d4, d5 }

OUT[B3] = { d6 } ∪ ({ d3, d4, d5 } – { d3 }) = { d4, d5, d6 }

IN[B4] = OUT[B3] ∪ OUT[B2] = { d3, d4, d5, d6 }

OUT[B4] = { d7 } ∪ ({ d3, d4, d5, d6 } – { d1, d4 }) = { d3, d5, d6, d7 }

到达-定值分析非向量计算方法

一第二次迭代

IN[B1] = ∅; // B1 无前驱结点

OUT[B1] = GEN[B1] ∪(IN[B1]-KILL[B1]) = GEN[B1] = { d1, d2, d3 }

IN[B2] = OUT[B1] ∪ OUT[B4] = { d1,d2,d3 } ∪ { d3, d5, d6, d7 } = {d1, d2, d3, d5, d6, d7}

OUT[B2] = GEN[B2] ∪(IN[B2]-KILL[B2]) = { d4, d5 } ∪{ d3, d5, d6 } = { d3, d4, d5, d6 }

IN[B3] = OUT[B2] = { d3, d4, d5, d6 }

OUT[B3] = { d6 } ∪ ({ d3, d4, d5, d6 } – { d3 }) = { d4, d5, d6 }

IN[B4] = OUT[B3] ∪ OUT[B2] = { d3, d4, d5, d6 }

OUT[B4] = { d7 } ∪ ({ d3, d4, d5, d6 } – { d1, d4 }) = { d3, d5, d6, d7 }

经过三次迭代后，IN[B]和OUT[B] 不再变化。

到达-定值分析

- **到达-定值数据流等式是正向的方程**

$$\text{OUT}\ [B] = gen\ [B] \cup (\text{IN}\ [B] - kill\ [B])$$

$$\text{IN}\ [B] \quad = \cup_{\ P是B的前驱}\ \text{OUT}\ [P]$$

某些数据流等式是反向的

- **到达-定值数据流等式的合流运算是求并集**

$$\text{IN}\ [B] = \cup_{\ P是B的前驱}\ \text{OUT}\ [P]$$

某些数据流等式的合流运算是求交集

- **对到达-定值数据流方程，迭代求它的最小解**

某些数据流方程可能需要求最大解

一起努力
打造国产基础软硬件体系！

李 诚
国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心
计算机科学与技术学院

2024年11月13日