并行编译与优化
Parallel Compiler and Optimization

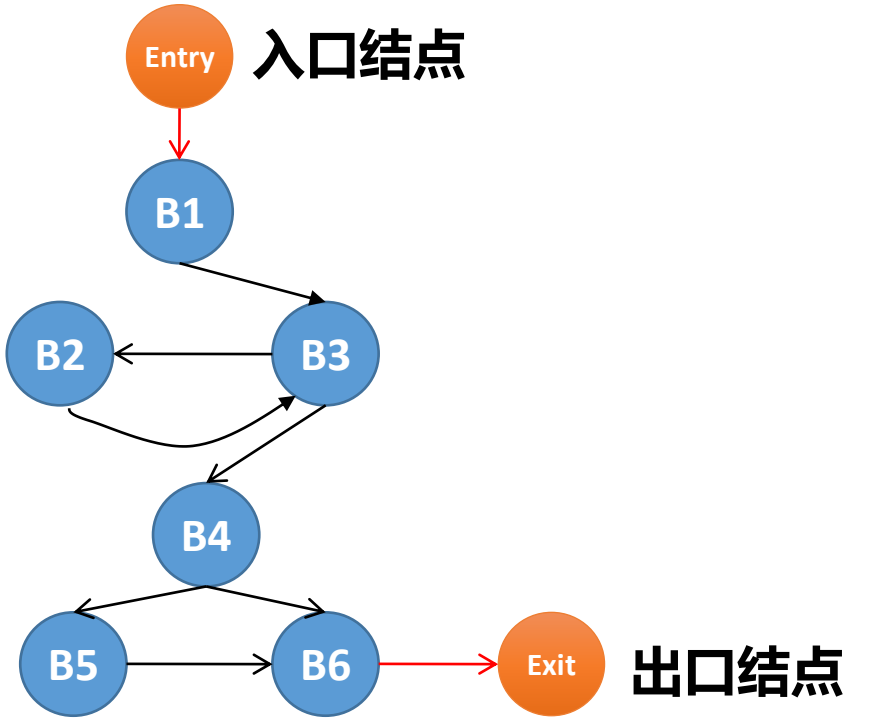
计算机研究所编译系统室

Lecture Six: Data Flow Analysis

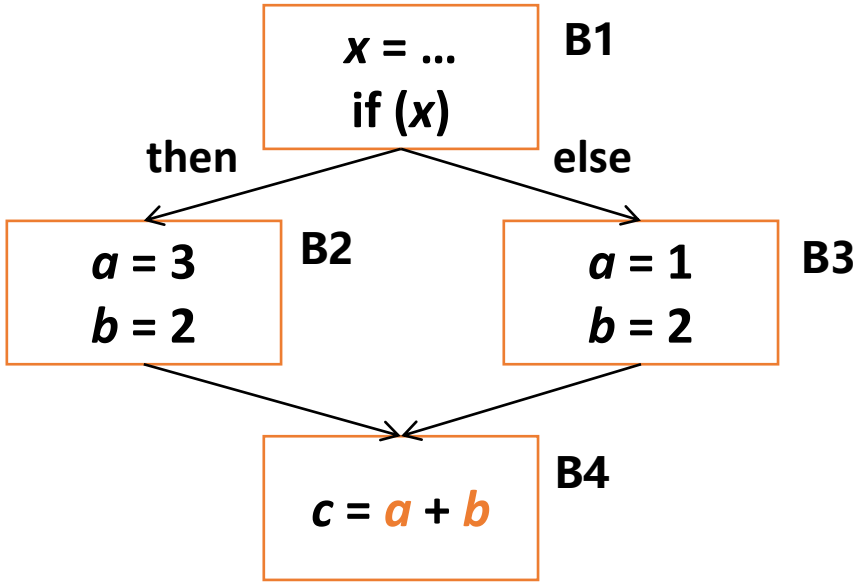
第六课：数据流分析

回顾

- 控制流分析(Control-Flow Analysis，CFA)
  - 分析程序的控制结构，构建程序控制流图
  - 基于控制流图的程序分析: 计算**必经结点**集合，找出**循环**
- 控制流图(Control-Flow Graph，CFG)
  - **结点**表示基本块
    - 基本块是只能从它的开始处进入结束处离开的顺序代码序列
  - **边**表示程序的控制流向
    - 表示程序**可能**走这条路径



变量c是常数吗?

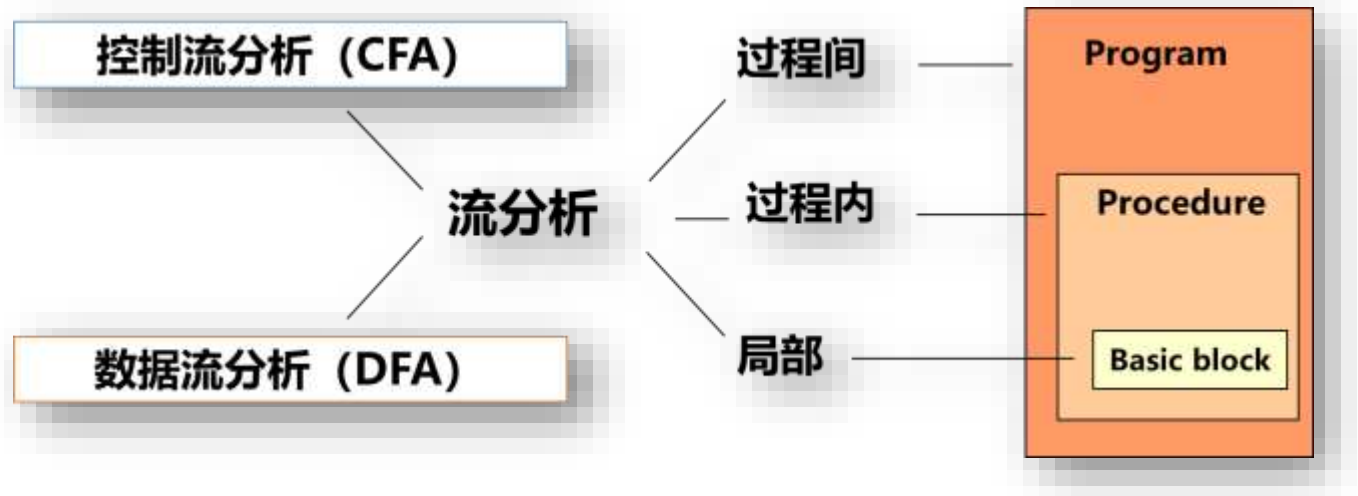


- 变量c是否是常数?
- c=a+b处，a、b的哪些定值能够到达?

- 控制流图中没有包含程序数据信息
- 需要获取数据沿程序执行路径流动的相关信息—数据流分析
- 优化编译器都需要做数据流分析

数据流分析

- 数据流分析(Data Flow Analysis，DFA)
  - 和控制流分析一样，是后续程序分析和编译优化的基础
- DFA是控制流-敏感的分析
  - 数据流动信息和程序的控制转移相关
- 基于数据流分析，完成
  - 中端：冗余优化
    - 公用子表达式删除
    - 死代码删除 ……
  - 后端：寄存器分配



学习内容

COMP
CCRG Open MP

6.1 点和路径

6.2 到达定值分析

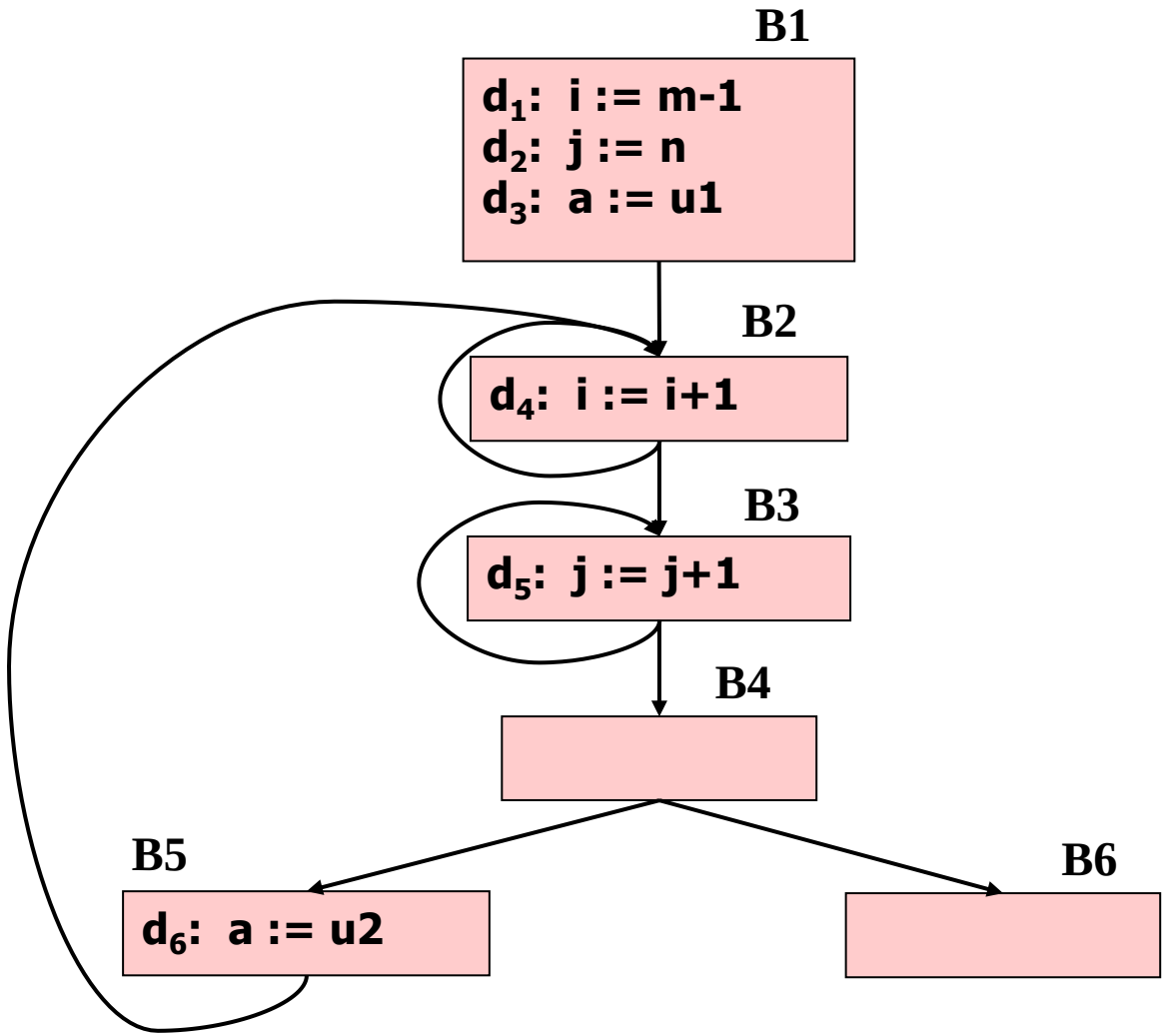
6.3 活跃变量分析

6.4 数据流分析框架

学习目标

- 掌握到达定值分析方法和活跃变量分析方法
- 熟悉数据流分析基本方法和迭代的数据流分析框架

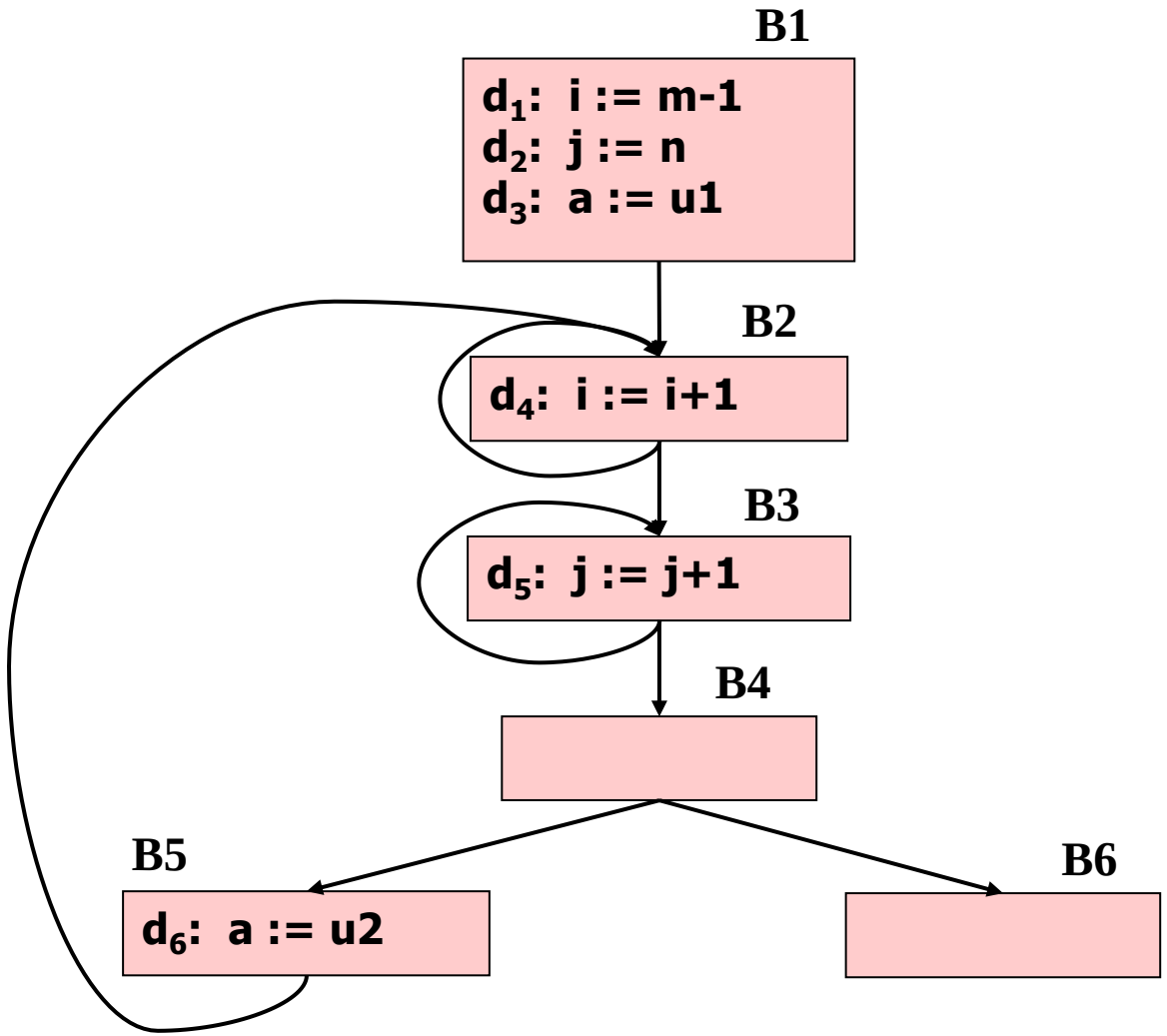
6.1.1 点（Point）



一个基本块内的点：
- 语句之间
- 第一条语句之前
- 最后一条语句之后

第一条语句之前点：基本块入口点
最后一条语句之后：基本块出口点

6.1.1 点（Point）

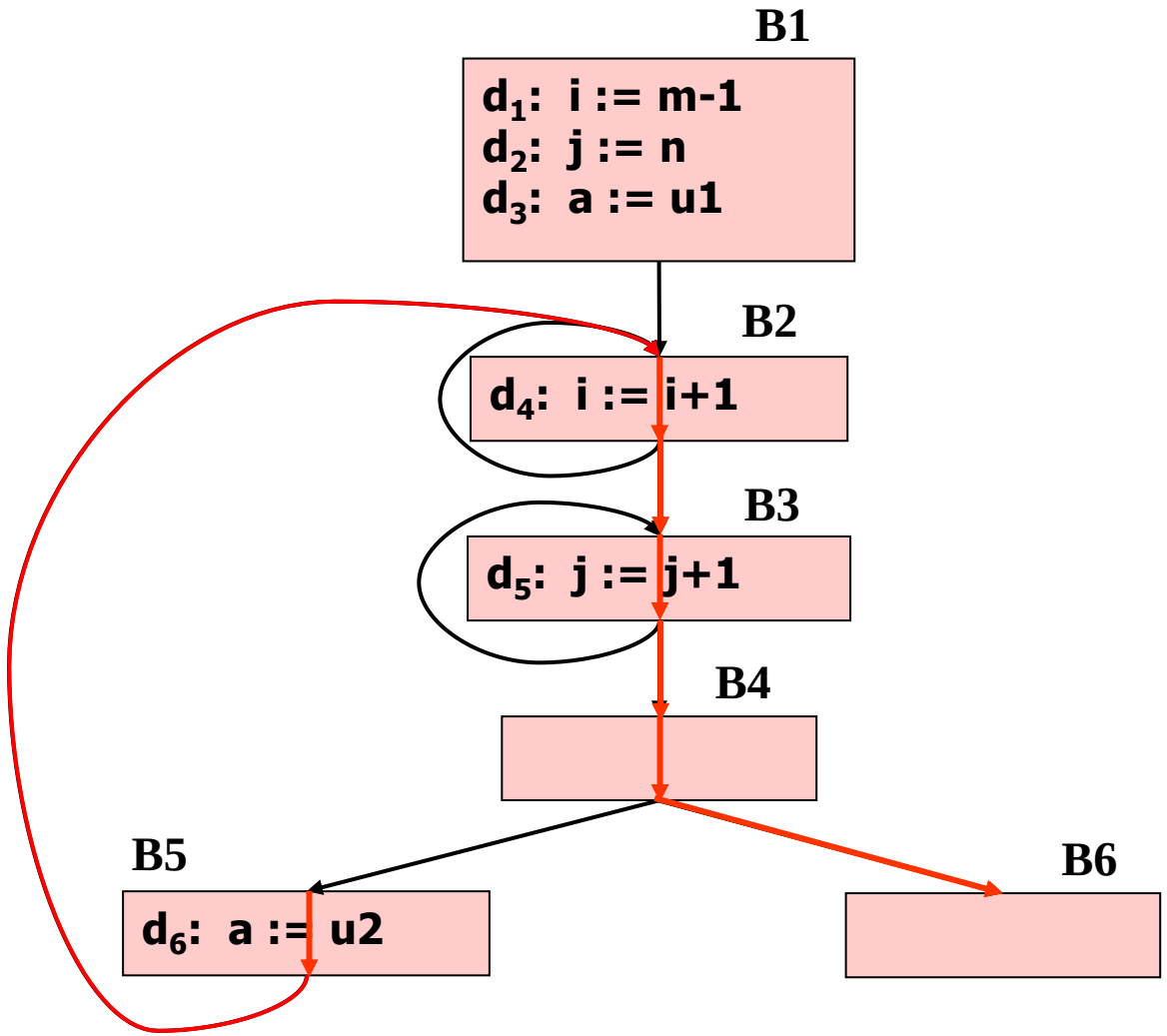


一个基本块内的点：
- 语句之间
- 第一条语句之前
- 最后一条语句之后

基本块 B1, B2, B3和B5 分别有多少个点?

B1 有4个点,
B2, B3, B5 各2个点

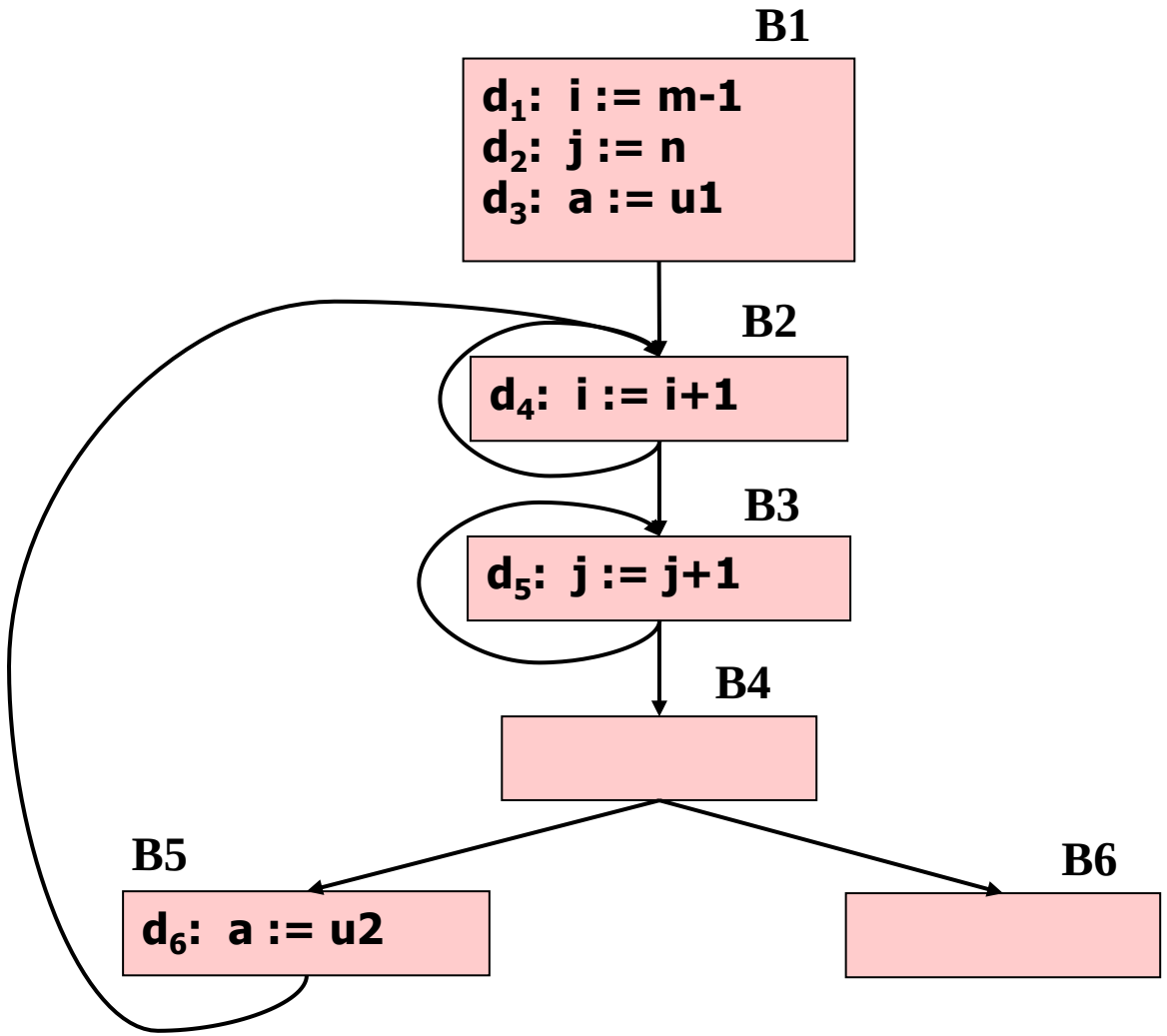
6.1.2 路径（Path）



B5的入口点到B6的入口点是否存在一条路径?

存在。从B5入口点开始，经过B5所有点，到B5的出口点，到B2, B3和B4所有点，到达B6入口点

6.1.2 路径（Path）

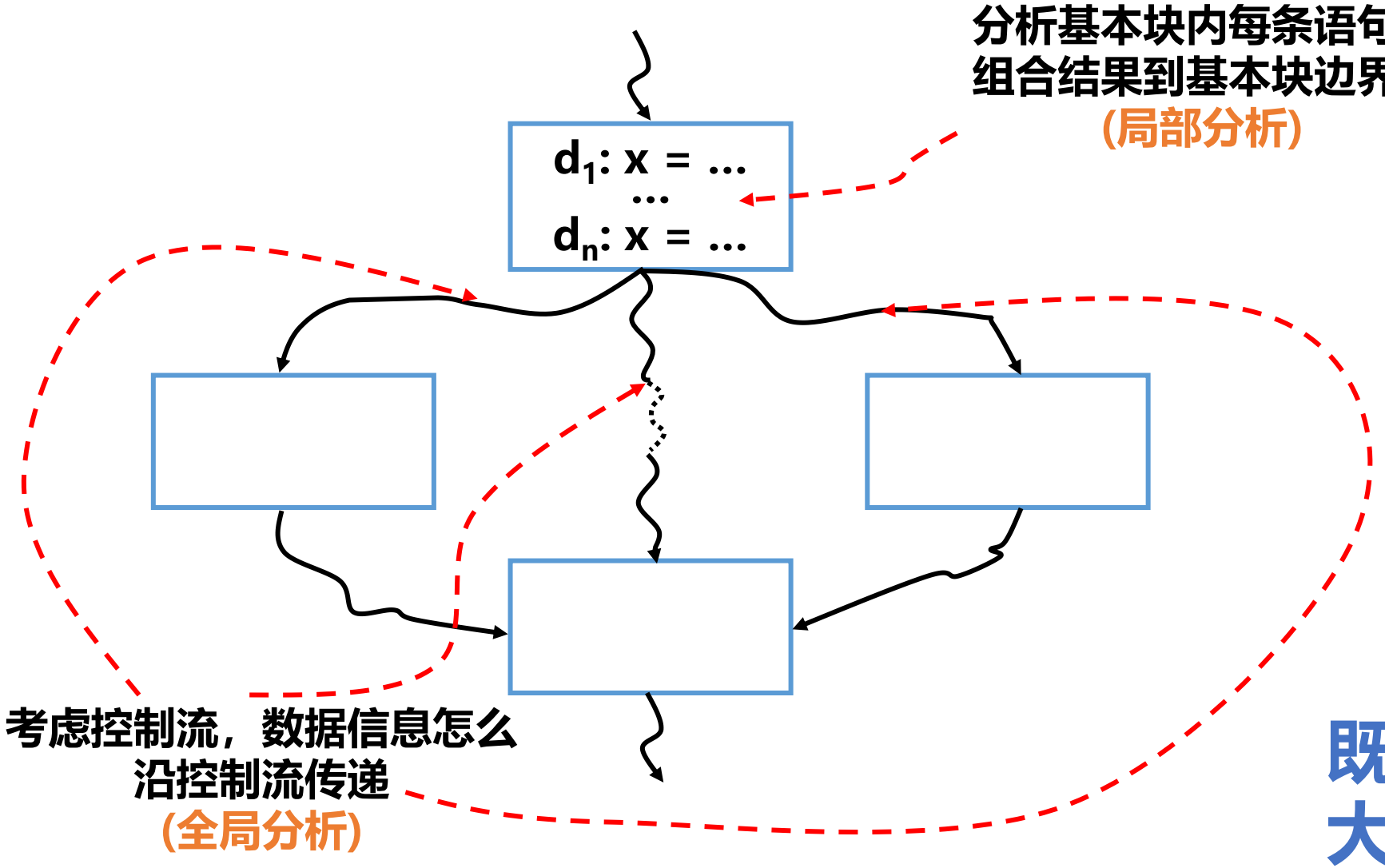


路径是点p_1, p_2, …, p_n 组成的序列，满足：

(i) 如果p_i 紧邻在语句S之前，p_{i+1} 紧跟在S之后

(ii) 或者p_i 是一个基本块的出口点，p_{i+1} 是其后继基本块的入口点

数据流分析基本方法



全局和局部
既见树木，又见森林
大处着眼，小处着手

学习内容

6.1 点和路径

6.2 到达定值分析

6.3 活跃变量分析

6.4 数据流分析框架

6.2.1 到达-定值

- 定值：每一个对变量v的赋值就是v的一次定值
  - S_k 对变量V_1进行定值
  - S_k 使用变量V_2和V_3

$$S_k: V_1 = V_2 + V_3$$

6.2.1 到达-定值

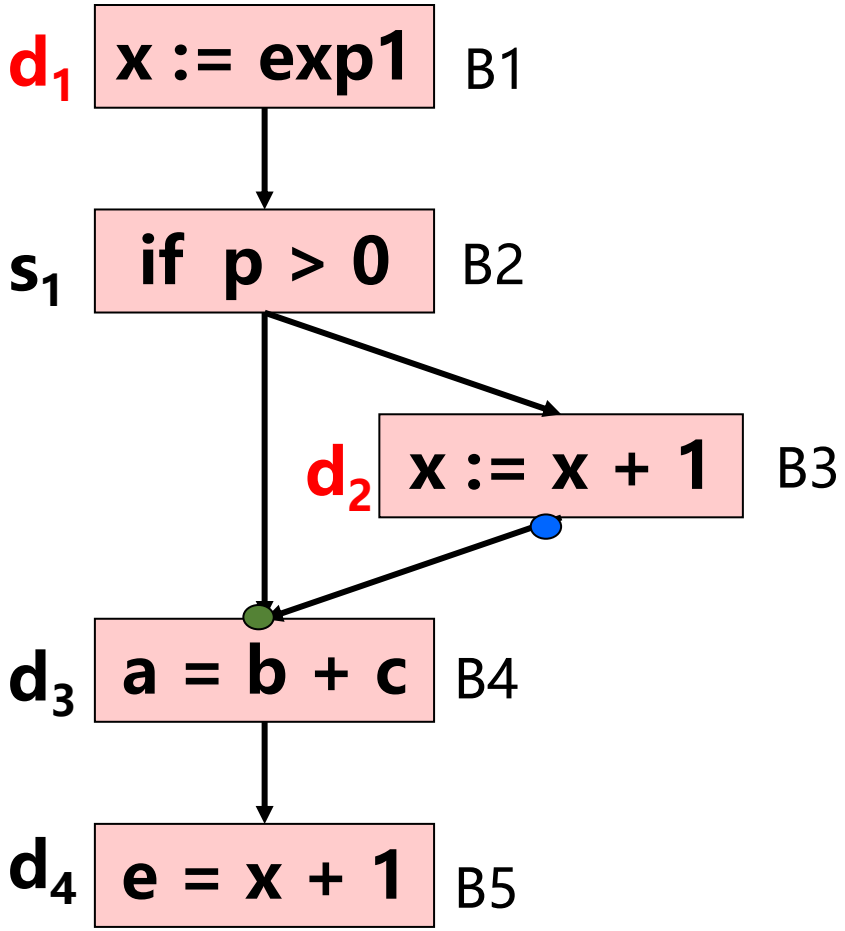
- 到达-定值(Reaching Definitions)
  - 变量v有一次定值d，如果存在一条从该定值点开始的路径到达点p，并且沿着这条路径，定值d没有被其他定值杀死，那么就说定值d达到点p
- 到达-定值分析
  - 对每个基本块，分析求解到达基本块边界(入口点和出口点)的定值集合

6.2.1 到达-定值

示例

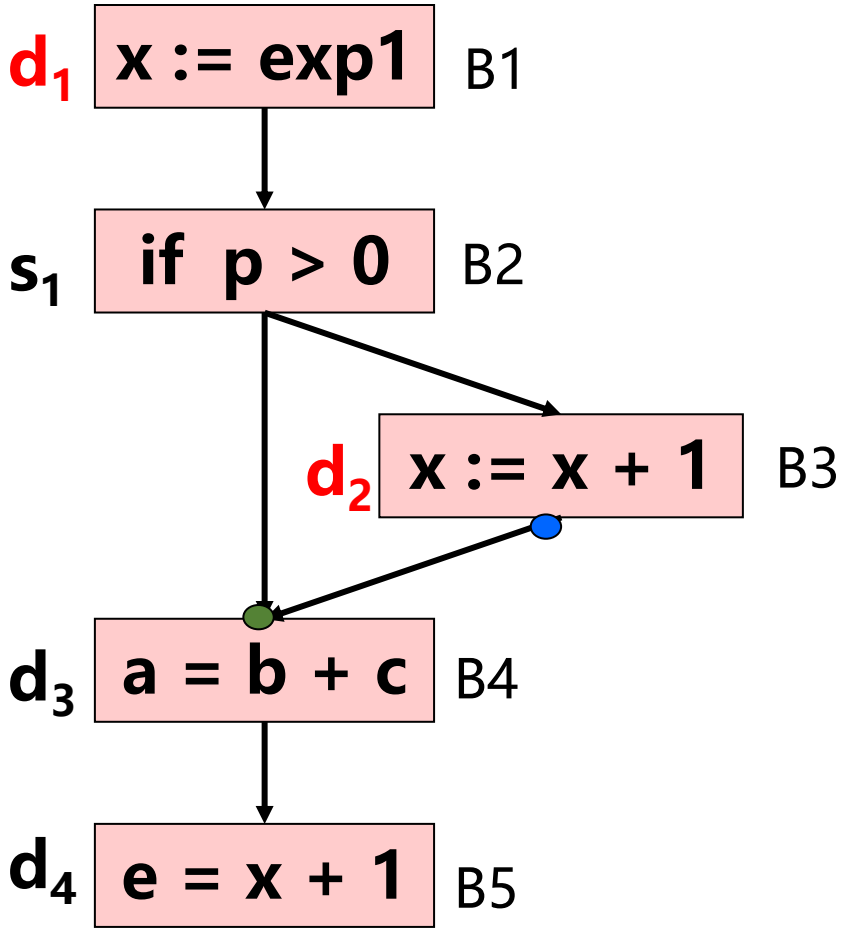
d_1 到达 ●

d_1 不能到达 ●，被定值d_2杀死



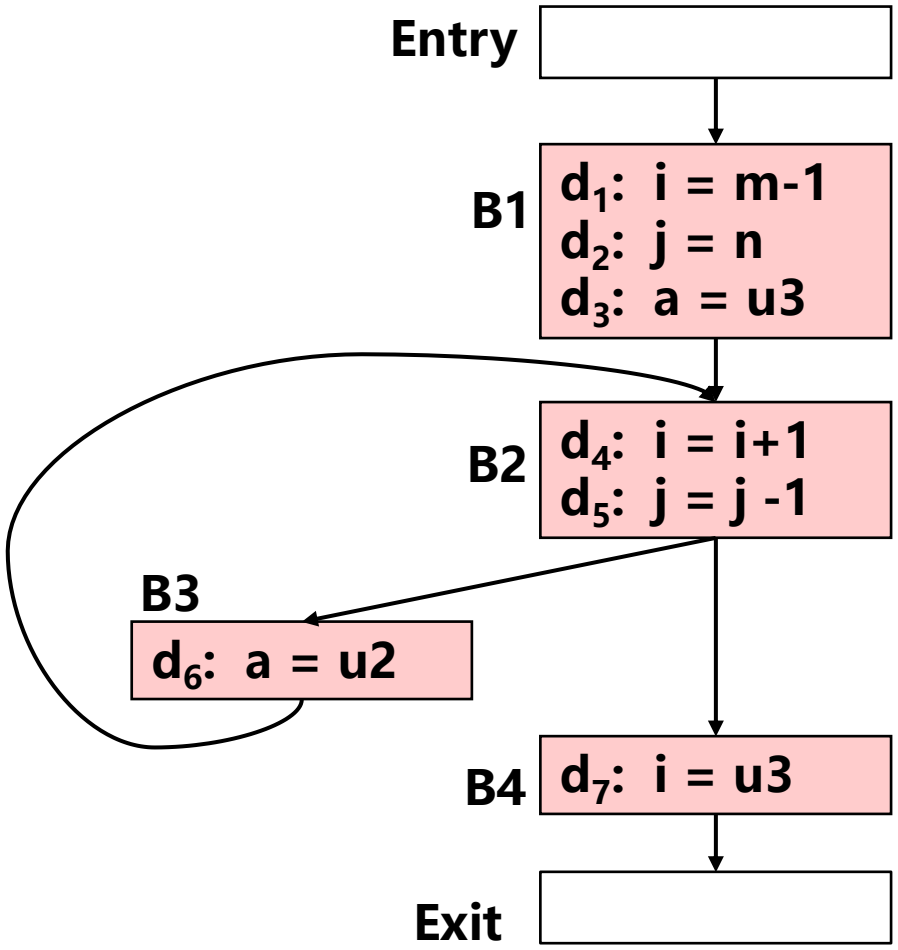
6.2.2 到达-定值分析方法

- **in[B]：到达基本块B入口点的定值集合**
- **out[B]：到达基本块B出口点的定值集合**
- **gen[B]：出现在基本块B内，并且可以到达基本块B出口点的定值集合**
  - gen[B1]={d1}, gen[B4]={d3}
- **kill[B]：在其他基本块内定值，并且被基本块B杀死（即不能达到B出口点）的定值集合**
  - 每一个在B内定值的变量v, 杀死的定值包括所有其他基本块内对v的定值
  - kill[B3]={d1}, kill[B4]= ∅



6.2.2 到达-定值分析方法

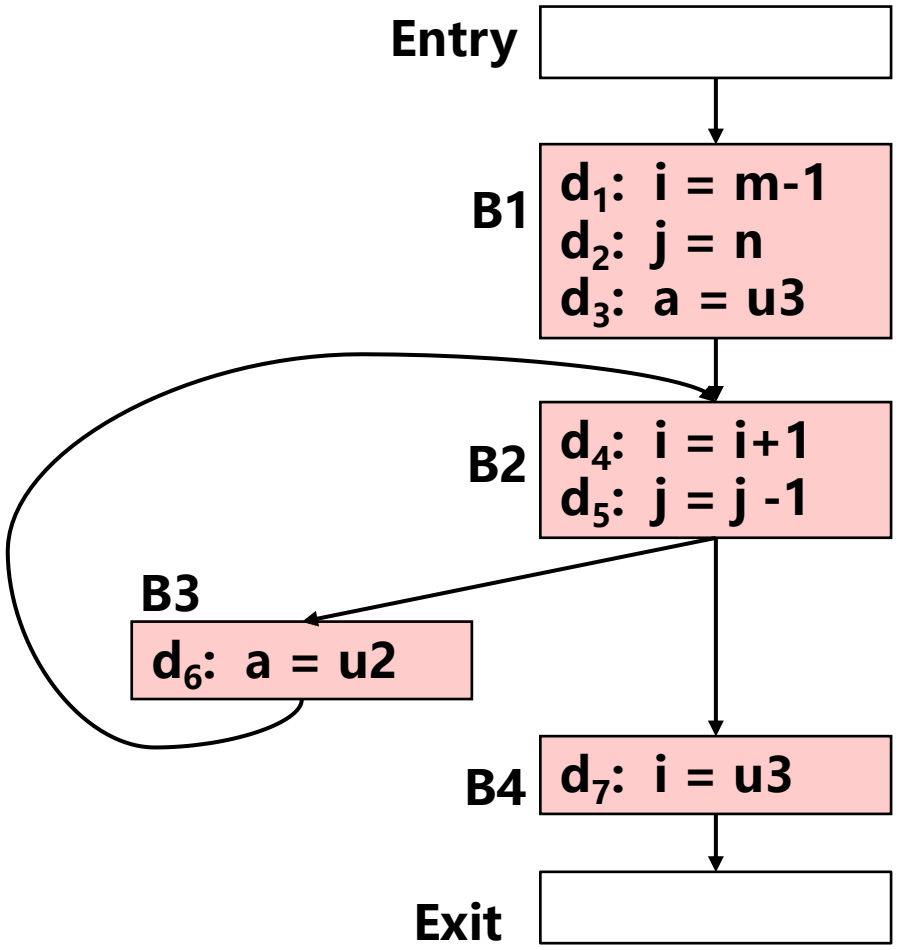
- 第一步：计算各个基本块的gen[B]和kill[B]



	gen[B]	kill[B]
Entry	∅	∅
B1		
B2		
B3		
B4		
Exit	∅	∅

6.2.2 到达-定值分析方法

- 第一步：计算各个基本块的gen[B]和kill[B]

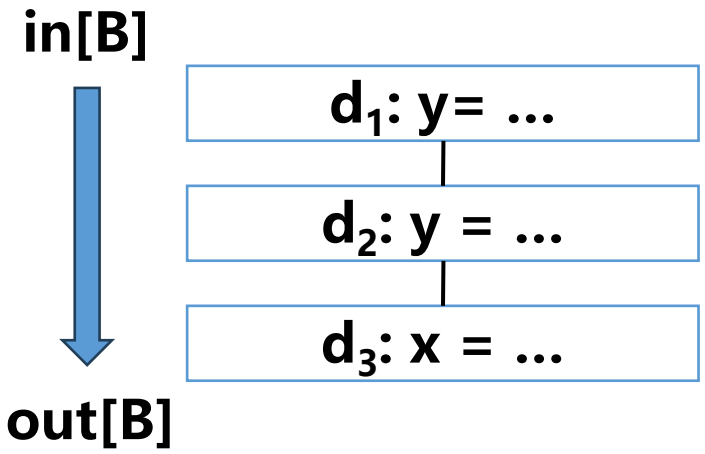


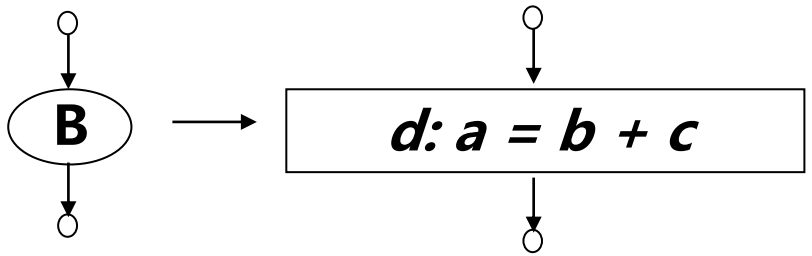
	gen[B]	kill[B]
Entry	∅	∅
B1	$\{d_1,d_2,d_3\}$	$\{d_4, d_5, d_6, d_7\}$
B2	$\{d_4, d_5\}$	$\{d_1, d_2, d_7\}$
B3	$\{d_6\}$	$\{d_3\}$
B4	$\{d_7\}$	$\{d_1, d_4\}$
Exit	∅	∅

6.2.2 到达-定值分析方法

第二步：建立数据流方程

基本块内（局部分析）



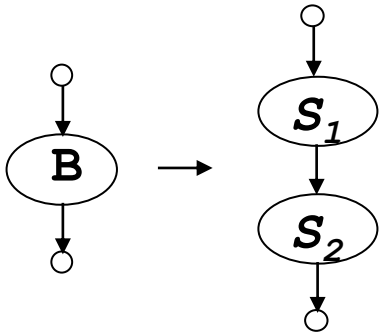


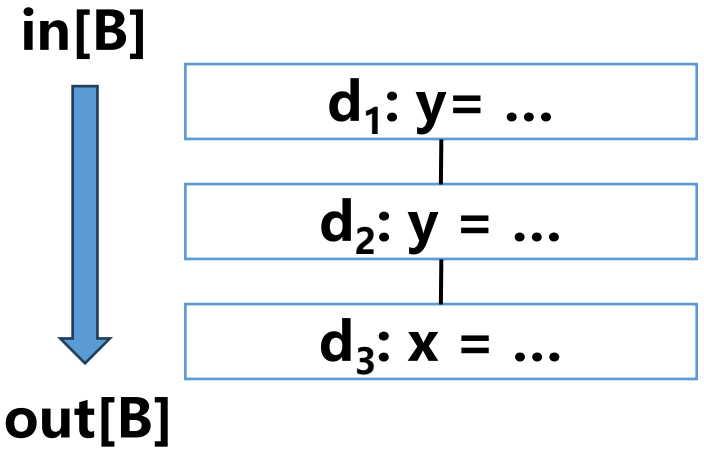
$$out[B] = f_s(in[B])$$
$$= gen[B] \cup (in[B] - kill[B])$$

6.2.2 到达-定值分析方法

第二步：建立数据流方程

基本块内（局部分析）



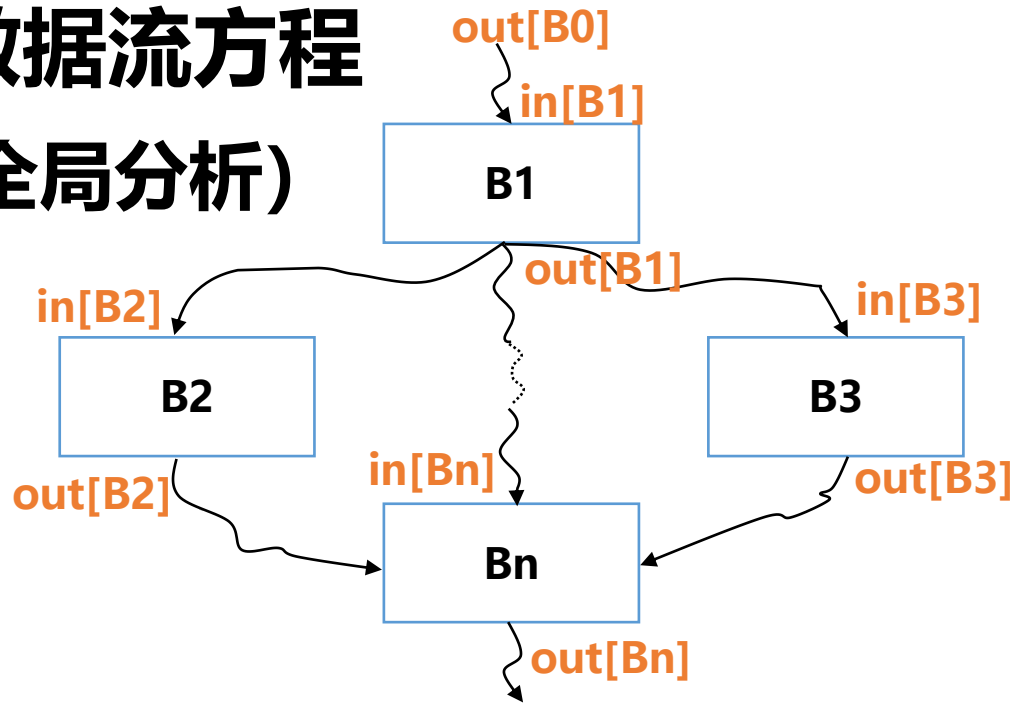


$in[S_1] = in[B]$
$in[S_2] = out[S_1]$
$out[B] = out[S_2]$

$$out[B] = f_B(in[B]) = f_{S2}(f_{S1}(in[B])) = gen[B] \cup (in[B] - kill[B])$$

6.2.2 到达-定值分析方法

- 第二步：建立数据流方程
  - 考虑控制流（全局分析）



in[B2] = out[B1]
out[B2] = gen[B2]∪(in[B2]–kill[B2])

in[Bn] = out[B1]∪out[B2] ∪ … = $\bigcup_{P \in pred[Bn]} out[P]$

out[Bn] = gen[Bn]∪(in[Bn]–kill[Bn])

in[B] = $\bigcup_{P \in pred[B]} out[P]$

out[B] = gen[B]∪(in[B] – kill[B])

6.2.2 到达-定值分析方法

第三步：求解数据流方程

- 控制流图中，有两个特殊的空结点Entry和Exit
- 对于Entry结点，有out[Entry] = ∅
- 对于所有非Entry结点，有

$$in[B] = \bigcup_{P \in pred[B]} out[P]$$

$$out[B] = gen[B] \cup (in[B] - kill[B])$$

- 使用不动点方法求解数据流方程

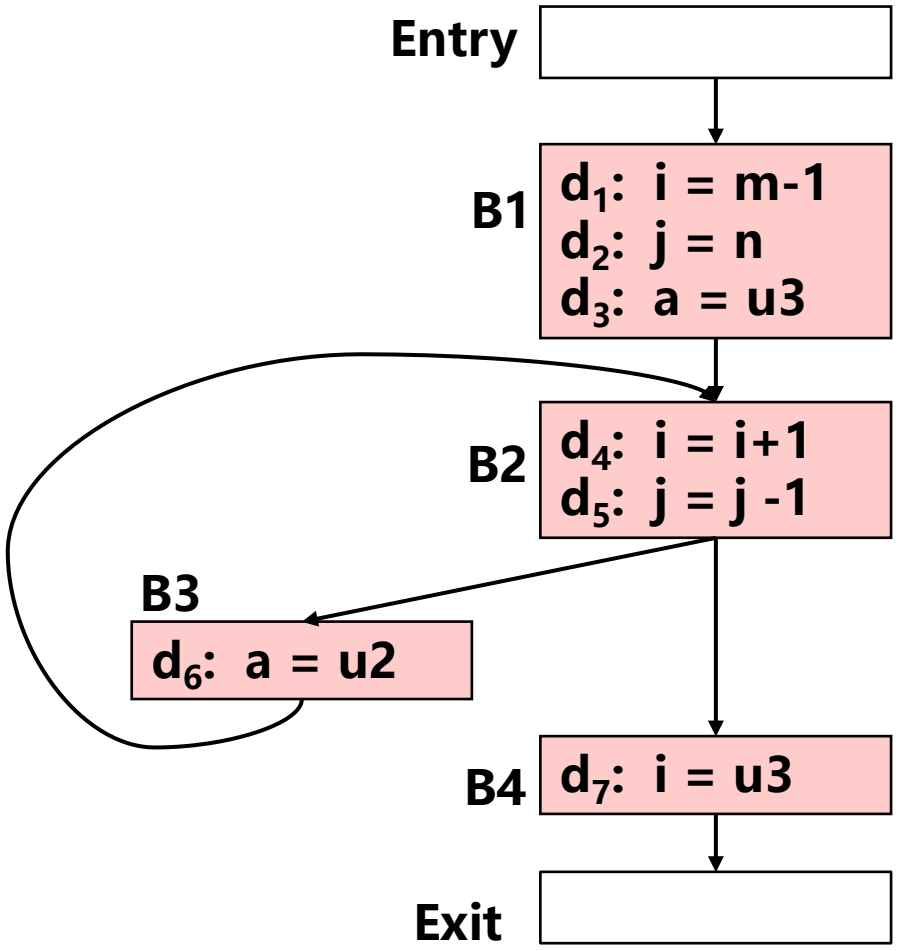
6.2.2 到达-定值分析方法

第三步：求解数据流方程

```
out[Entry]=∅
每个基本块B∈N–{Entry}
        out[B]=∅
change = true
while(changes) {  //循环，直到所有基本块的out集合都不再发生变化
        change = false
        对每个基本块B∈N–{Entry} {
                oldout=out[B]
                in[B]= ⋃_{P∈pred[B]} out[P]
                out[B]= gen[B]∪(in[B]-kill[B])
                if(out[B]≠oldout) change=true;
        }
}
```

求解不动点

6.2.3 到达-定值分析示例



第一步: 计算每个基本块的gen和kill集合

gen[Entry]= ∅
kill[Entry] = ∅

gen[B1]={d_1, d_2, d_3}
kill[B1] ={d_4, d_5, d_6, d_7}

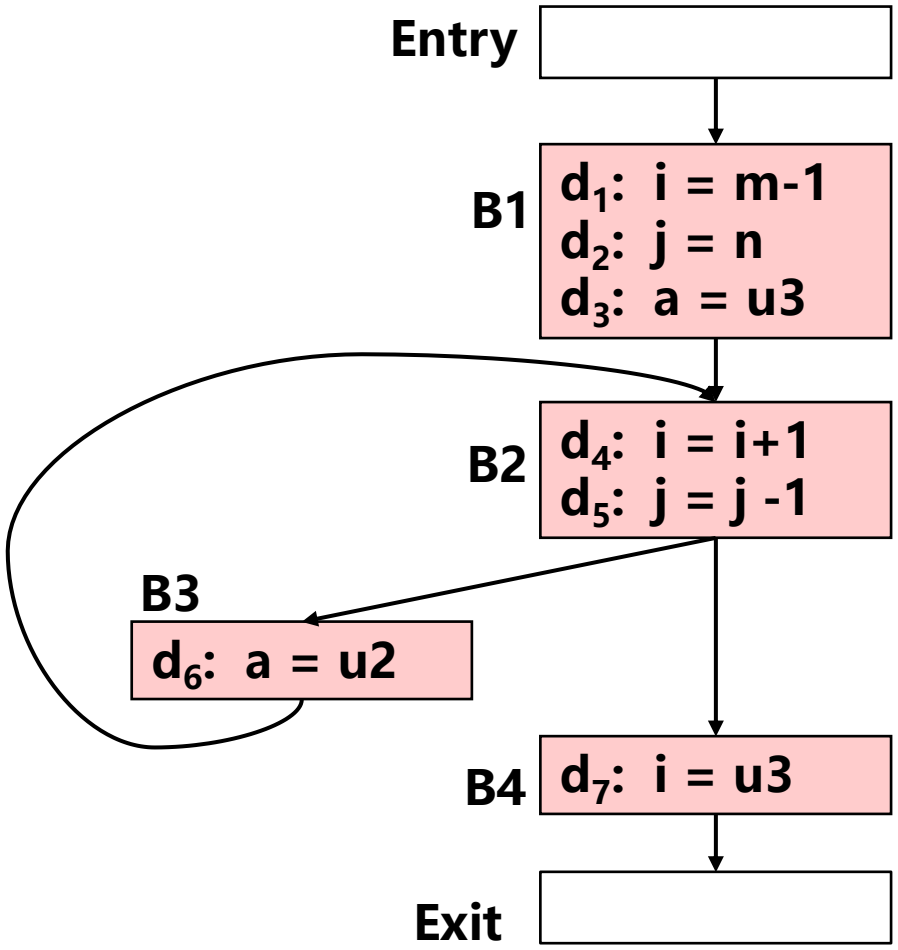
gen[B2]={d_4, d_5}
kill[B2] ={d_1, d_2, d_7}

gen[B3]={d_6}
kill[B3] ={d_3}

gen[B4]={d_7}
kill[B4] ={d_1, d_4}

gen[Exit]= ∅
kill[Exit] = ∅

6.2.3 到达-定值分析示例



第二步: 对每个基本块B，置初值
out[B] = ∅

out[Entry]= ∅

out[B1]=∅

out[B2]=∅

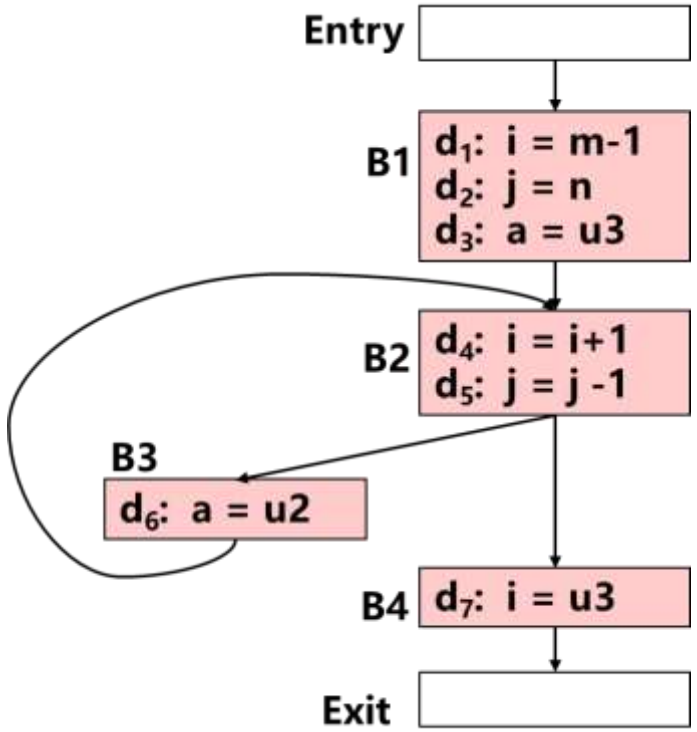
out[B3]=∅

out[B4]=∅

out[Exit]=∅

6.2.3 到达-定值分析示例

	gen[B]	kill[B]
Entry	∅	∅
B1	$\{d_1,d_2,d_3\}$	$\{d_4, d_5, d_6, d_7\}$
B2	$\{d_4, d_5\}$	$\{d_1, d_2, d_7\}$
B3	$\{d_6\}$	$\{d_3\}$
B4	$\{d_7\}$	$\{d_1, d_4\}$
Exit	∅	∅



第三步: 循环，直到找到不动点
in[B]=∪out[P]　P是B的前驱
out[B]=gen[B]∪(in[B]-kill[B])

第一次迭代:
in[B1]= ∅
out[B1]=$\{d_1, d_2, d_3\}$ //变化

in[B2]=$\{d_1, d_2, d_3\}$
out[B2]=$\{d_4, d_5, d_3\}$ //变化

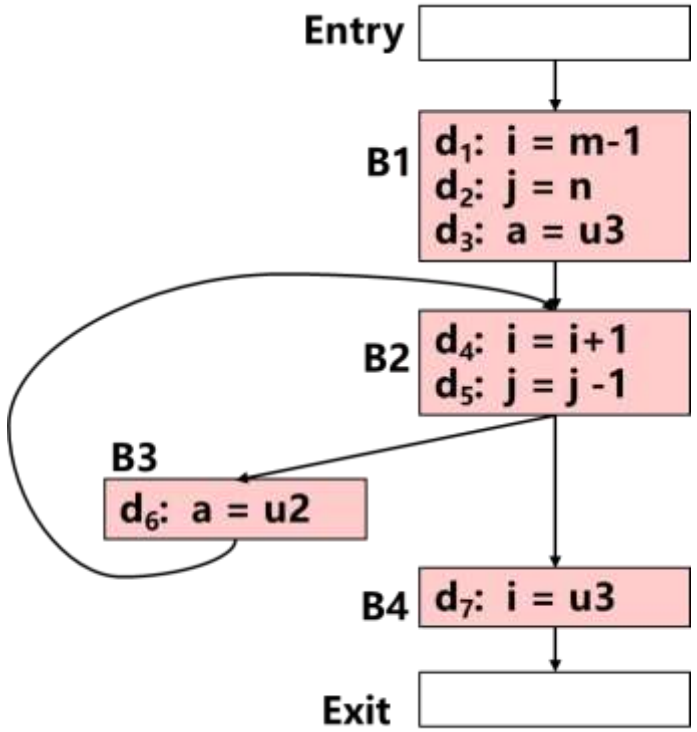
in[B3]=$\{d_4, d_5, d_3\}$
out[B3]=$\{d_4, d_5, d_6\}$ //变化

in[B4]=$\{d_4, d_5, d_3\}$
out[B4]=$\{d_3, d_5, d_7\}$ //变化

in[Exit] = $\{d_3, d_5, d_7\}$
out[Exit] = $\{d_3, d_5, d_7\}$ //变化

6.2.3 到达-定值分析示例

	gen[B]	kill[B]
Entry	∅	∅
B1	$\{d_1,d_2,d_3\}$	$\{d_4, d_5, d_6, d_7\}$
B2	$\{d_4, d_5\}$	$\{d_1, d_2, d_7\}$
B3	$\{d_6\}$	$\{d_3\}$
B4	$\{d_7\}$	$\{d_1, d_4\}$
Exit	∅	∅



第三步: 循环，直到找到不动点
in[B]=∪out[P]　P是B的前驱
out[B]=gen[B]∪(in[B]-kill[B])

第二次迭代:
in[B1]=∅
out[B1]={d_1, d_2, d_3}　　//未变化

in[B2]={d_1, d_2, d_3, d_4, d_5, d_6}
out[B2]={d_3, d_4, d_5, d_6}　//变化

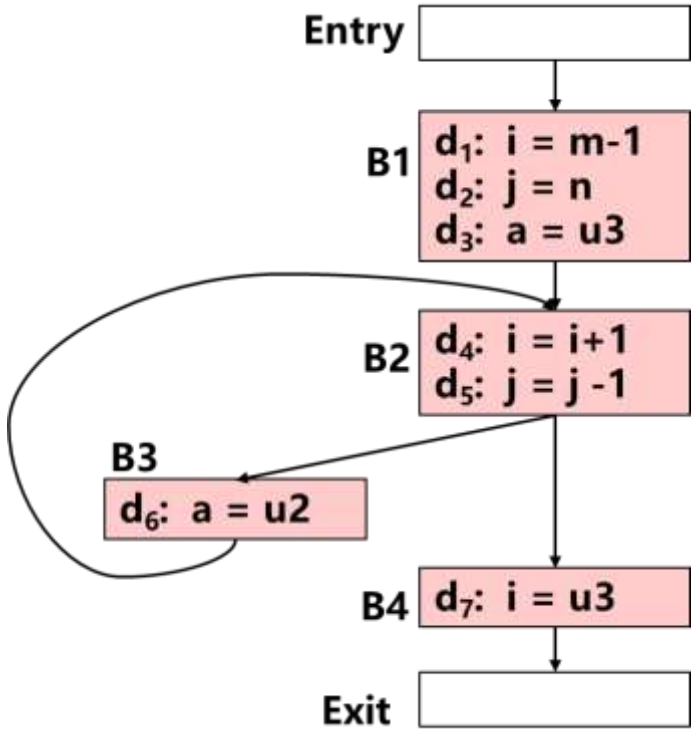
in[B3]={d_3, d_4, d_5, d_6}
out[B3]={d_4, d_5, d_6}　　//未变化

in[B4]={d_3, d_4, d_5, d_6}
out[B4]={d_3, d_5, d_6, d_7}　//变化

in[Exit] = {d_3, d_5, d_6, d_7}
out[Exit] = {d_3, d_5, d_6, d_7} //变化

6.2.3 到达-定值分析示例

	gen[B]	kill[B]
Entry	∅	∅
B1	$\{d_1,d_2,d_3\}$	$\{d_4, d_5, d_6, d_7\}$
B2	$\{d_4, d_5\}$	$\{d_1, d_2, d_7\}$
B3	$\{d_6\}$	$\{d_3\}$
B4	$\{d_7\}$	$\{d_1, d_4\}$
Exit	∅	∅



第三步: 循环，直到找到不动点
in[B]=∪out[P]　P是B的前驱
out[B]=gen[B]∪(in[B]-kill[B])

第三次迭代:
in[B1]=∅
out[B1]={d_1, d_2, d_3}　　//未变化

in[B2]={d_1, d_2, d_3, d_4, d_5, d_6}
out[B2]={d_3, d_4, d_5, d_6}　//未变化

in[B3]={d_3, d_4, d_5, d_6}
out[B3]={d_4, d_5, d_6}　　//未变化

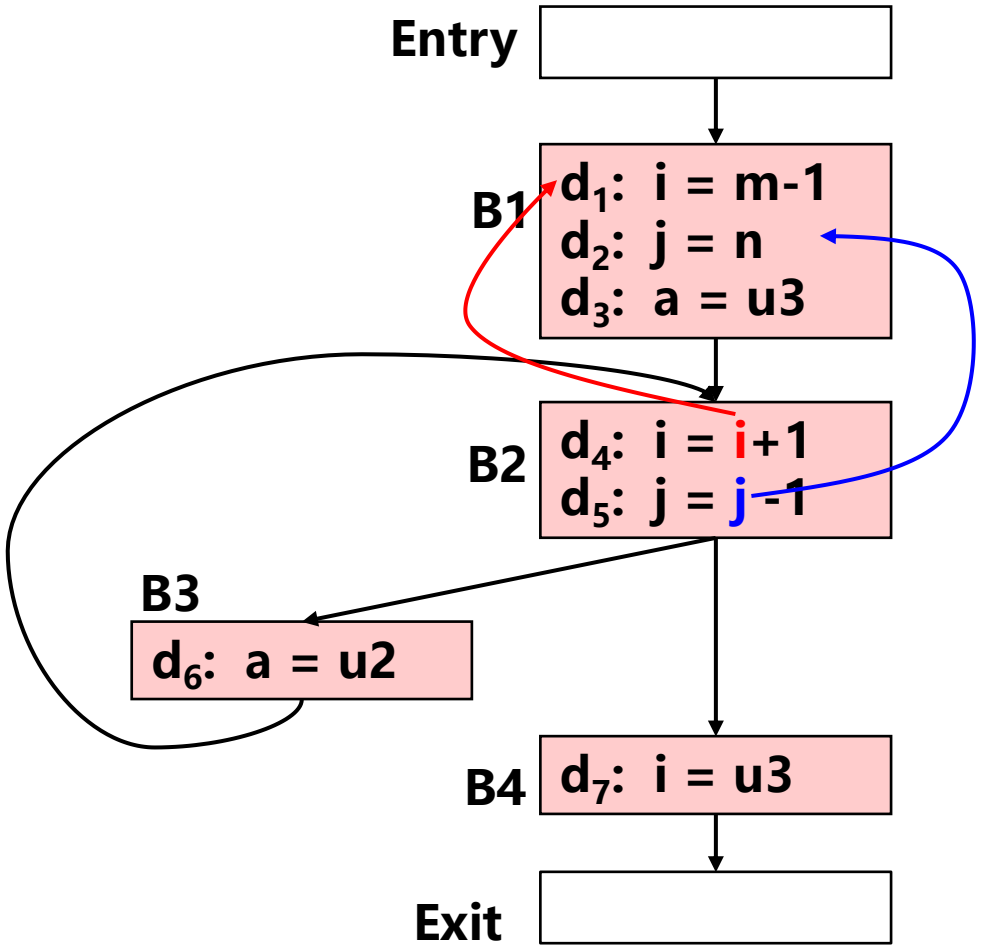
in[B4]={d_3, d_4, d_5, d_6}
out[B4]={d_3, d_5, d_6, d_7}　//未变化

in[Exit] = {d_3, d_5, d_6, d_7}
out[Exit] = {d_3, d_5, d_6, d_7} //未变化

找打不动点
循环终止

6.2.4 到达-定值信息存储

使用-定值链(Use-Definition Chain)



第一次迭代:

in[B1]= ∅

out[B1]={d_1, d_2, d_3} //变化

in[B2]={d_1, d_2, d_3}

out[B2]={d_4, d_5, d_3} //变化

in[B3]={d_4, d_5, d_3}

out[B3]={d_4, d_5, d_6} //变化

in[B4]={d_4, d_5, d_3}

out[B4]={d_3, d_5, d_7} //变化

in[Exit] = {d_3, d_5, d_7}

out[Exit] = {d_3, d_5, d_7} //变化

6.2.4 到达-定值信息存储

位串表示

- 一个比特位表示一次定值
- 位串长度=程序中定值次数

BB	out[B]^0	in[B]^1	out[B]^1
Entry	000 0000	000 0000	000 0000
B1	000 0000	000 0000	111 0000
B2	000 0000	111 0000	001 1100
B3	000 0000	001 1100	000 1110
B4	000 0000	001 1100	001 0101
Exit	000 0000	001 0101	001 0101

第一次迭代:

in[B1]= ∅

out[B1]={d_1, d_2, d_3} //变化

in[B2]={d_1, d_2, d_3}

out[B2]={d_4, d_5, d_3} //变化

in[B3]={d_4, d_5, d_3}

out[B3]={d_4, d_5, d_6} //变化

in[B4]={d_4, d_5, d_3}

out[B4]={d_3, d_5, d_7} //变化

in[Exit] = {d_3, d_5, d_7}

out[Exit] = {d_3, d_5, d_7} //变化

6.2.4 到达-定值信息存储

COMP CCRG Open MP

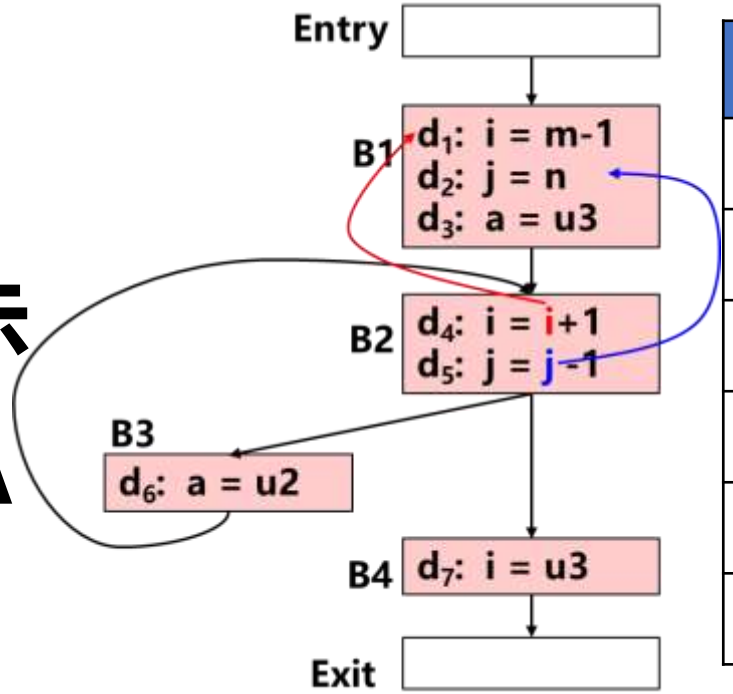
位串表示

- 集合的并运算(∪): 位串的逻辑OR运算
- 集合的差(in-kill): 先求kill位串的补，再与in进行逻辑AND运算

BB	$out[B]^0$	$in[B]^1$	$out[B]^1$	$in[B]^2$	$out[B]^2$	$in[B]^3$	$out[B]^3$
Entry	000 0000	000 0000	000 0000	000 0000	000 0000	000 0000	000 0000
B1	000 0000	000 0000	111 0000	000 0000	111 0000	000 0000	111 0000
B2	000 0000	111 0000	001 1100	111 1110	001 1110	111 1110	001 1110
B3	000 0000	001 1100	000 1110	001 1110	000 1110	001 1110	000 1110
B4	000 0000	001 1100	001 0101	001 1110	001 0111	001 1110	001 0111
Exit	000 0000	001 0101	001 0101	001 0111	001 0111	001 0111	001 0111

讨论

- 到达定值分析：对变量的某个定值可以到达变量的哪个使用
- 到达定值分析可以用来做什么优化?
  - 常数传播
  - 循环不变量外提
- UD链 vs 位串表示
  - 更好的表示? SSA



Block	in[B]³	out[B]³
Entry	000 0000	000 0000
B_1	000 0000	111 0000
B_2	111 1110	001 1110
B_3	001 1110	000 1110
B_4	001 1110	001 0111
Exit	001 0111	001 0111

学习内容

COMP
CCRG Open MP

6.1 点和路径

6.2 到达定值分析

6.3 活跃变量分析

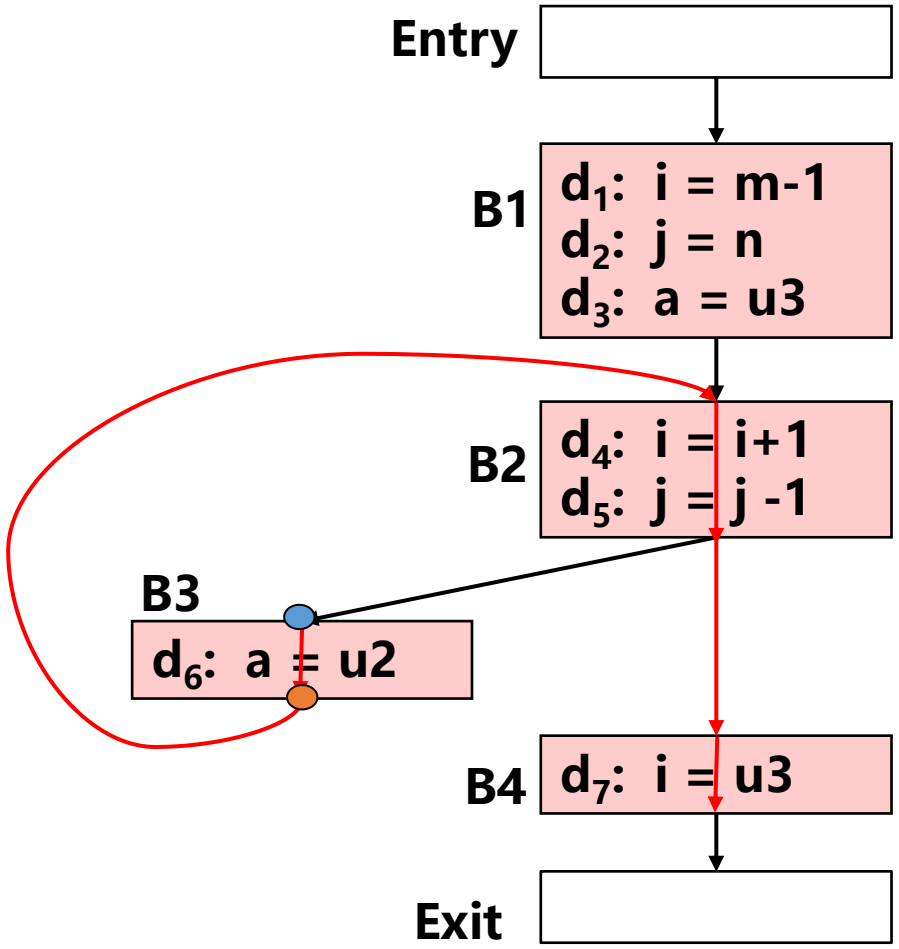
6.4 数据流分析框架

6.3.1 活跃变量

- 如果一个变量v在从点p开始的某条路径上被使用，那么变量v在点p是活跃的(*live*)
- 否则，变量v在点p是死变量

- 活跃变量分析
  - 对每个基本块，分析获取到达基本块边界(入口点和出口点)的活跃变量集合

6.3.1 活跃变量

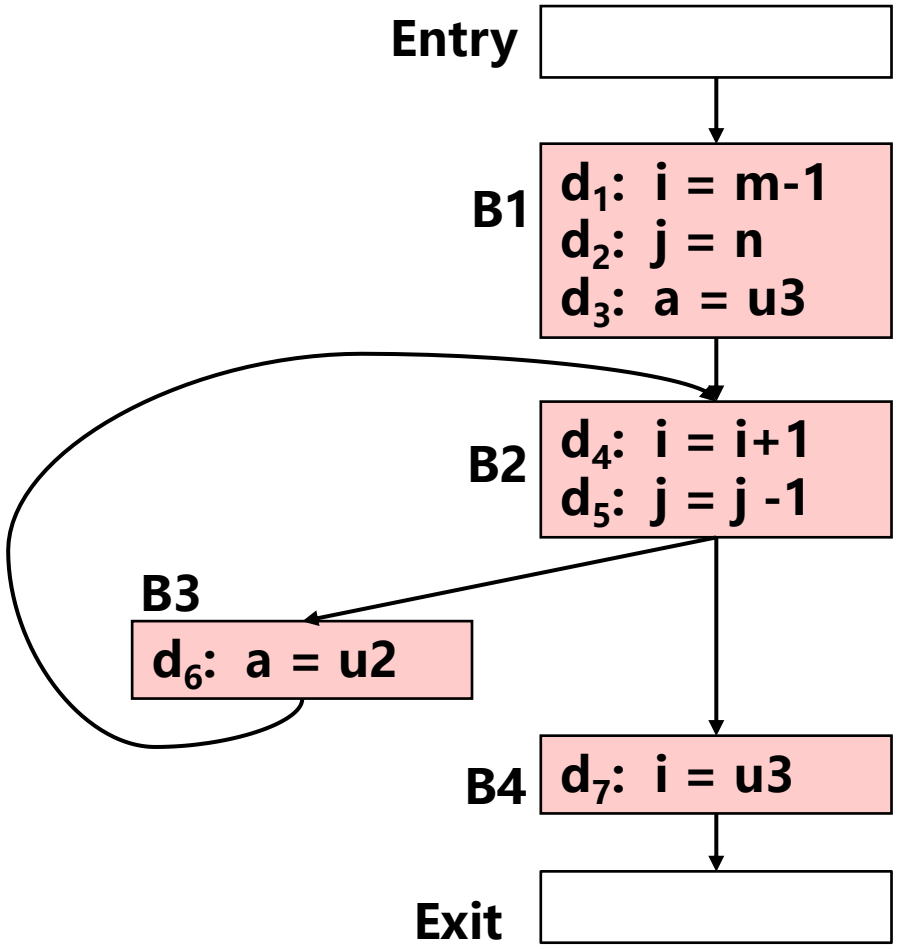


- 在B3入口点，哪些变量活跃?
  - 入口点开始的路径，使用的变量

    {u2, i, j, u3}
- 在B3出口点，哪些变量活跃?
  - 出口点开始的路径，使用的变量

    {u2, i, j, u3}

6.3.2 活跃变量分析方法



- in[B]: 基本块B入口点的活跃变量集合
- out[B]: 基本块B出口点的活跃变量集合
- use[B]: 基本块B中，定值之前使用的变量集合
- def[B]: 被基本块B定值的变量集合

- 第一步: 计算各个基本块的use[B]和def[B]

use[B1] = {m, n, u3}　　def[B1] = {i, j, a}

use[B2] = {i, j}　　def[B2] = {i, j}

use[B3] = {u2}　　def[B3] = {a}

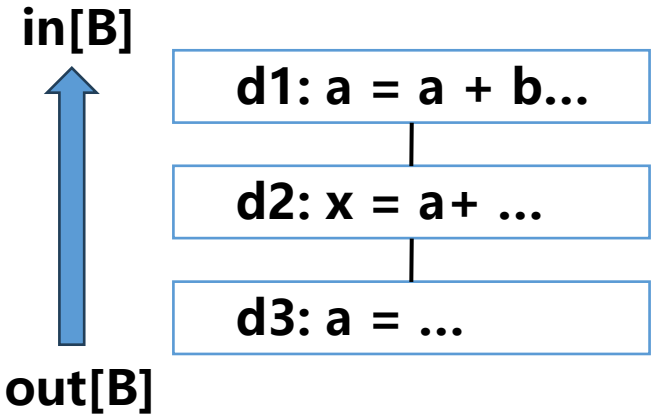
use[B4] = {u3}　　def[B4] = {i}

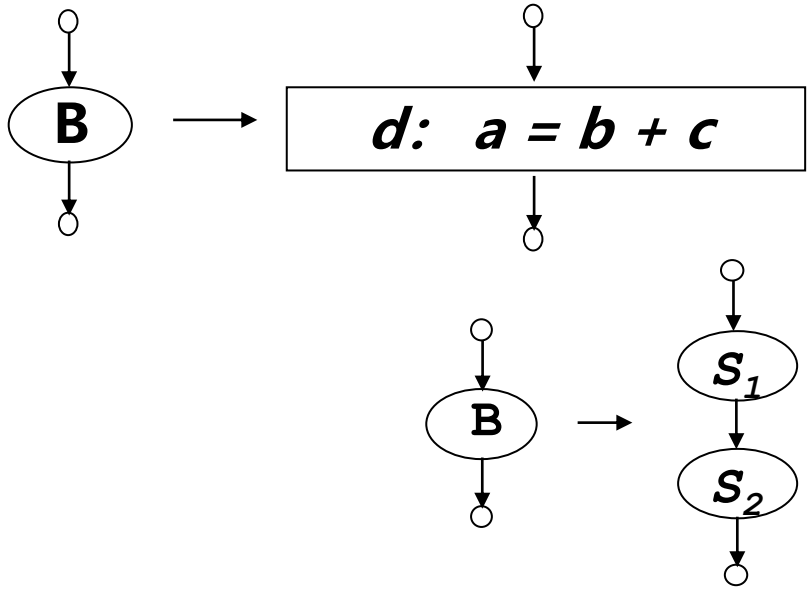
6.3.2 活跃变量分析方法

第二步: 建立数据流方程

- 基本块内（局部分析）

根据使用追踪定值——后向分析





后向分析: in[S] = f_S(out[S])

f_S: out[S]→in[S]

use[S] = {b, c}, def[S]=a

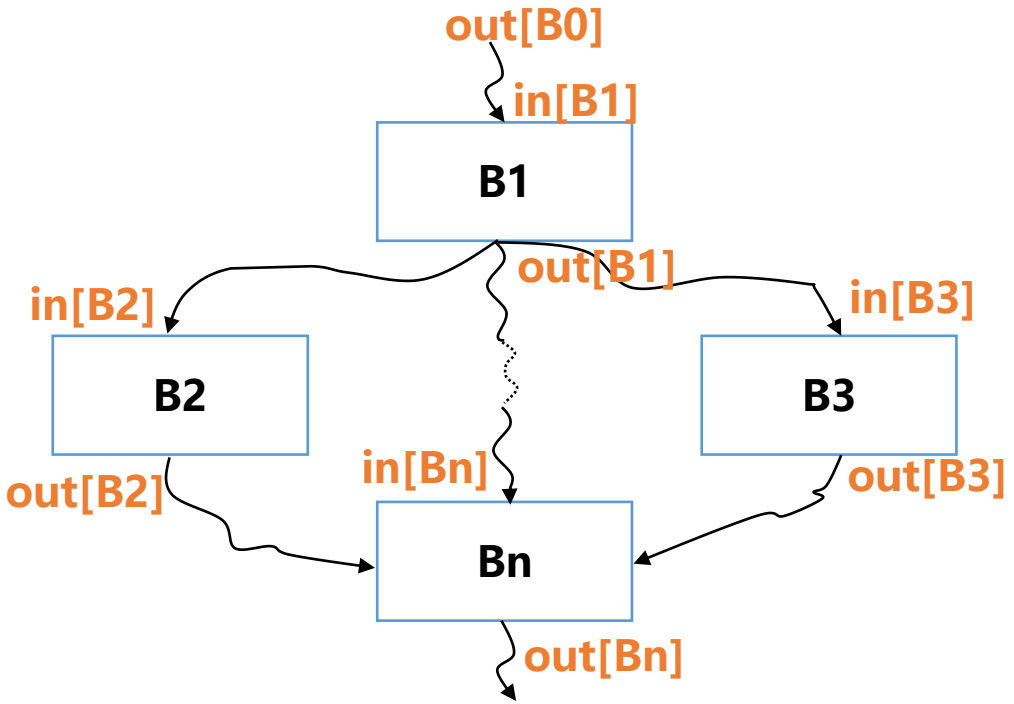
in[S]=use[S]∪(out[S]–def[S])

in[B]=f_B(out[B])=use[B]∪(out[B]–def[B])

6.3.2 活跃变量分析方法

第二步: 建立数据流方程

- 考虑控制流（全局分析）



out[B2]=in[Bn]

in[B2]=use[B2]∪(out[B2]–def[B2])

out[B1]=in[B1]∪in[B2]∪ … = $\bigcup_{S \in succ[B1]} in[S]$

in[B1]=use[B1]∪(out[B1]–def[B1])

out[B] = $\bigcup_{S \in succ[B]} in[S]$

in[B] = use[B]∪(out[B] – def[B])

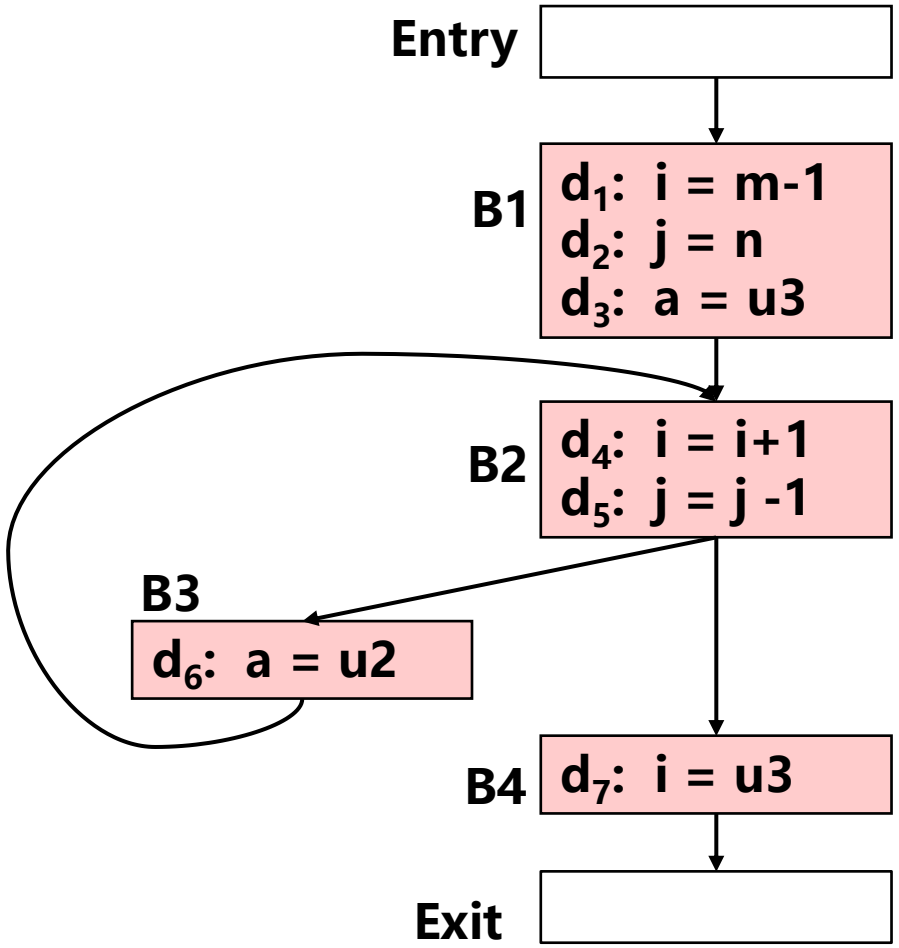
6.3.2 活跃变量分析方法

- 第三步: 使用不动点方法求解数据流方程

```
in[Exit]=∅
对每个基本块B∈N – {Exit}
        in[B]=∅
change=true
while(changes) {  //循环，直到所有基本块的in集合都不再发生变化
        change = false
        对每个基本块B∈N –{Exit}{
                oldin=in[B]
                out[B]= ⋃_{S∈succ[B]} in[S]
                in[B]=use[B]∪(out[B] - def[B])
                if(in[B]≠oldin) change=true;
        }
}
```

求解不动点

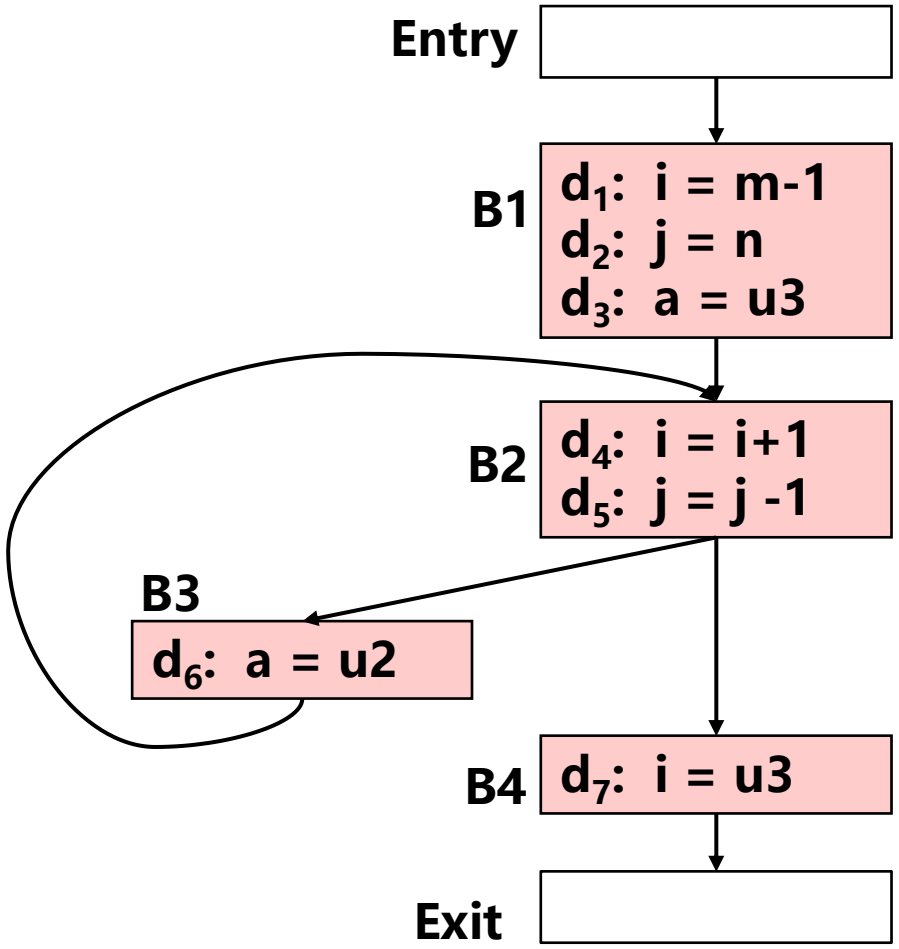
6.3.3 活跃变量分析示例



第一步: 计算每个基本块的use和def集合

BB	use[B]	def[B]	out[B]	in[B]
Entry	∅	∅		
B1	{m, n, u3}	{i, j, a}		
B2	{i, j}	{i, j}		
B3	{u2}	{a}		
B4	{u3}	{i}		
Exit	∅	∅		

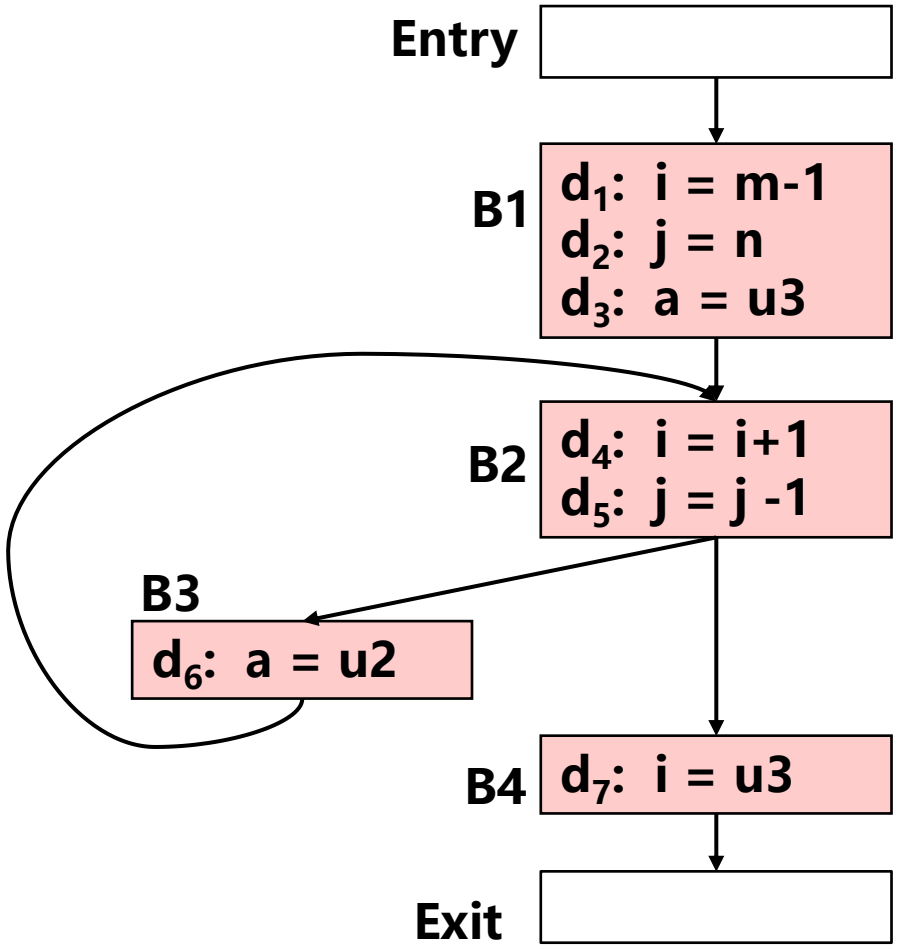
6.3.3 活跃变量分析示例



第二步: 对每个基本块B，置初值in[B] = ∅

BB	use[B]	def[B]	out[B]	in[B]
Entry	∅	∅		∅
B1	{m, n, u3}	{i, j, a}		∅
B2	{i, j}	{i, j}		∅
B3	{u2}	{a}		∅
B4	{u3}	{i}		∅
Exit	∅	∅		∅

6.3.3 活跃变量分析示例

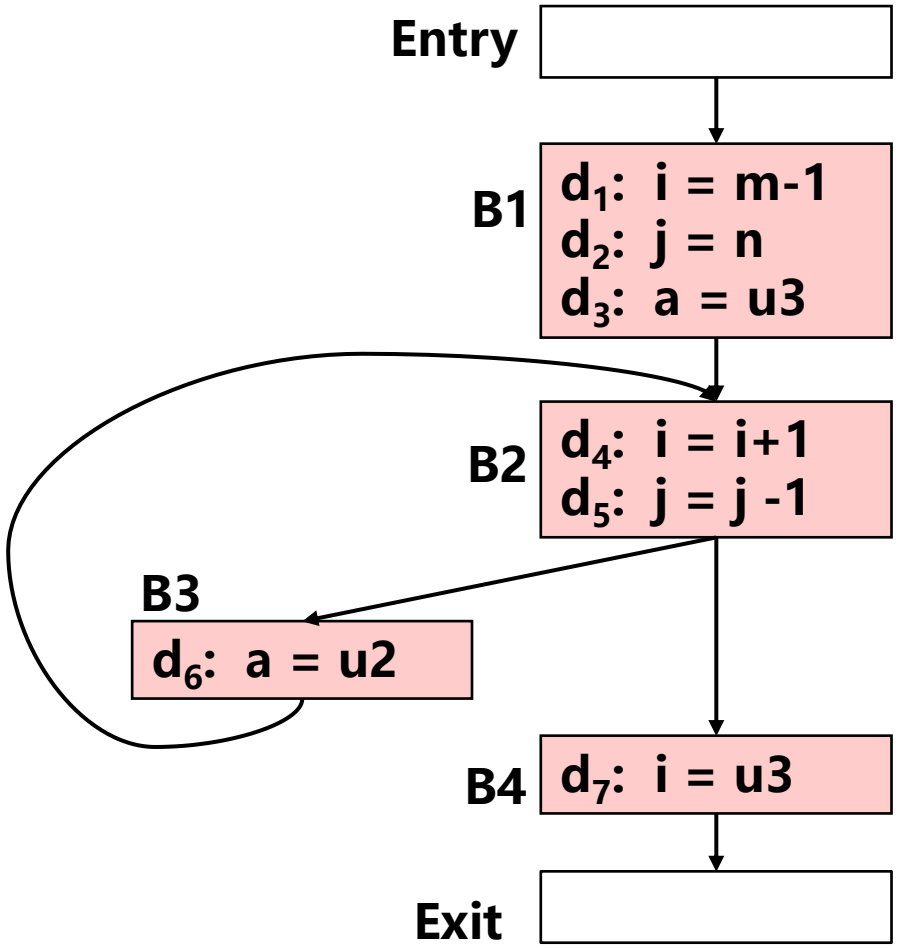


第三步: 循环，直到找到不动点
out[B]=∪in[S]　S是B的后继
in[B]=use[B]∪(out[B]-def[B])

第一次迭代

BB	use[B]	def[B]	out[B]	in[B]	
Entry	∅	∅	{m, n, u2, u3}	{m, n, u2, u3}	变化
B1	{m, n, u3}	{i, j, a}	{i, j, u2, u3}	{m, n, u2, u3}	变化
B2	{i, j}	{i, j}	{u2, u3}	{i, j, u2, u3}	变化
B3	{u2}	{a}	∅	{u2}	变化
B4	{u3}	{i}	∅	{u3}	变化
Exit	∅	∅	∅	∅	未变化

6.3.3 活跃变量分析示例

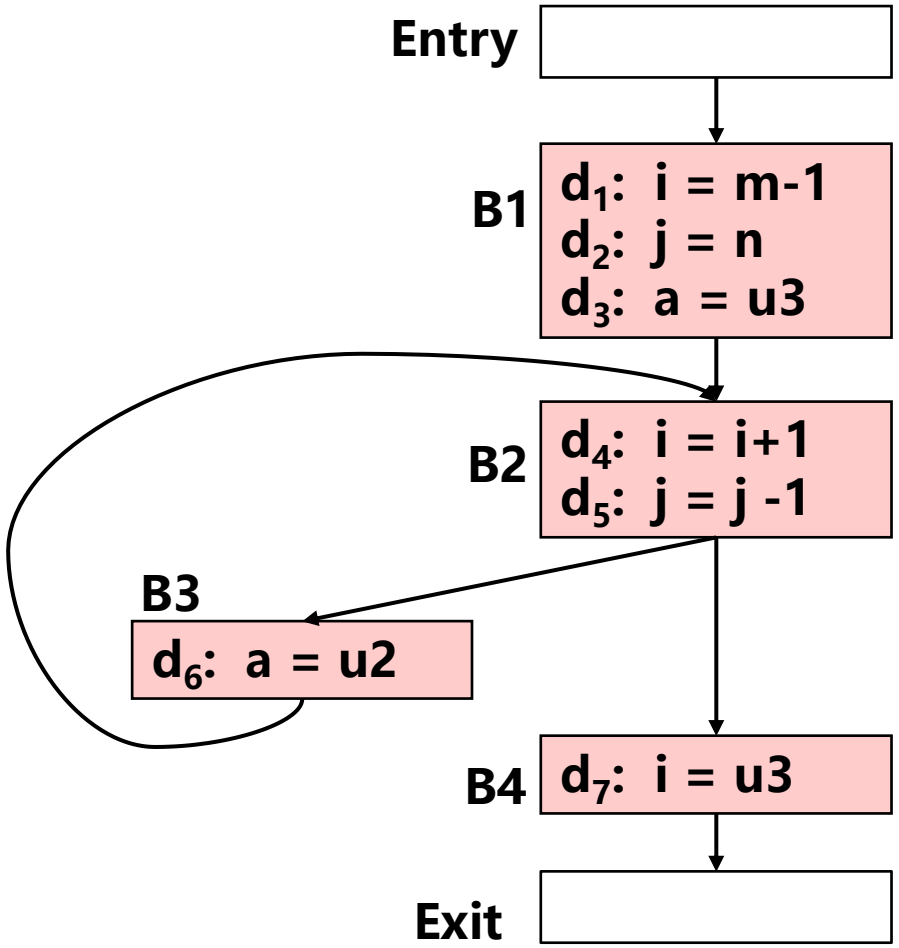


第三步: 循环，直到找到不动点
out[B]=∪in[S]　S是B的后继
in[B]=use[B]∪(out[B]-def[B])

第二次迭代

BB	use[B]	def[B]	out[B]	in[B]	
Entry	∅	∅	{m, n, u2, u3}	{m, n, u2, u3}	未变化
B1	{m, n, u3}	{i, j, a}	{i, j, u2, u3}	{m, n, u2, u3}	未变化
B2	{i, j}	{i, j}	{i, j, u2, u3}	{i, j, u2, u3}	未变化
B3	{u2}	{a}	{i, j, u2, u3}	{i, j, u2, u3}	变化
B4	{u3}	{i}	∅	{u3}	未变化
Exit	∅	∅	∅	∅	未变化

6.3.3 活跃变量分析示例



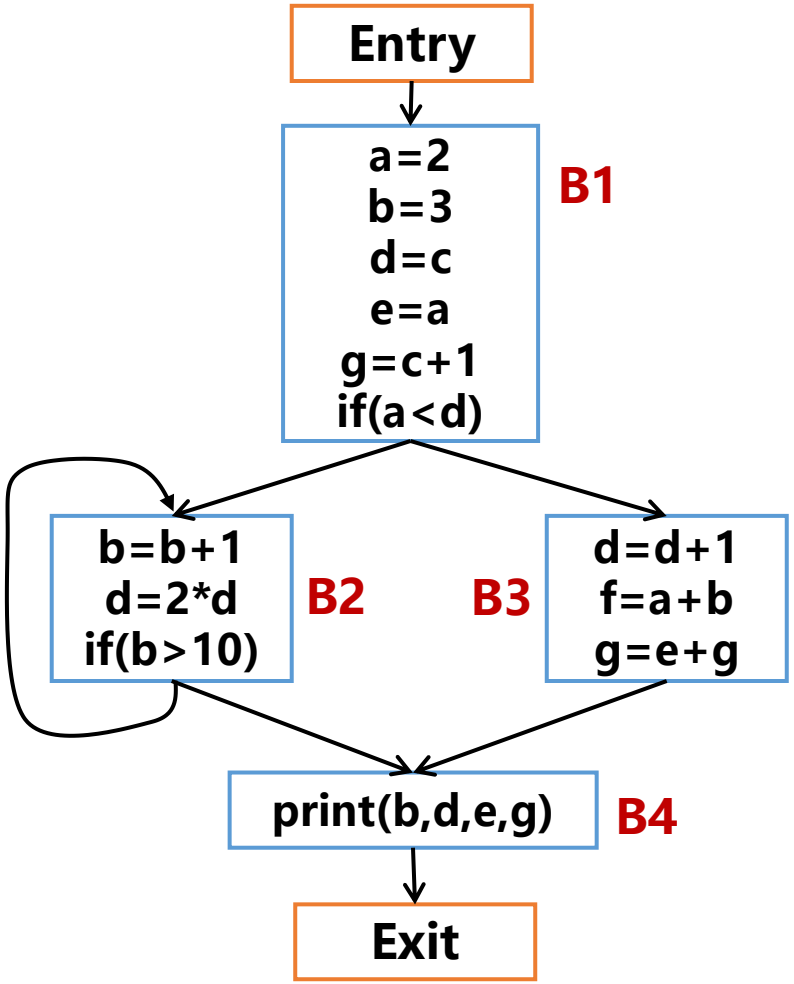
第三步: 循环，直到找到不动点
out[B]=∪in[S]　S是B的后继
in[B]=use[B]∪(out[B]-def[B])

第三次迭代：找到不动点循环终止

BB	use[B]	def[B]	out[B]	in[B]	
Entry	∅	∅	{m, n, u2, u3}	{m, n, u2, u3}	未变化
B1	{m, n, u3}	{i, j, a}	{i, j, u2, u3}	{m, n, u2, u3}	未变化
B2	{i, j}	{i, j}	{i, j, u2, u3}	{i, j, u2, u3}	未变化
B3	{u2}	{a}	{i, j, u2, u3}	{i, j, u2, u3}	未变化
B4	{u3}	{i}	∅	{u3}	未变化
Exit	∅	∅	∅	∅	未变化

练习

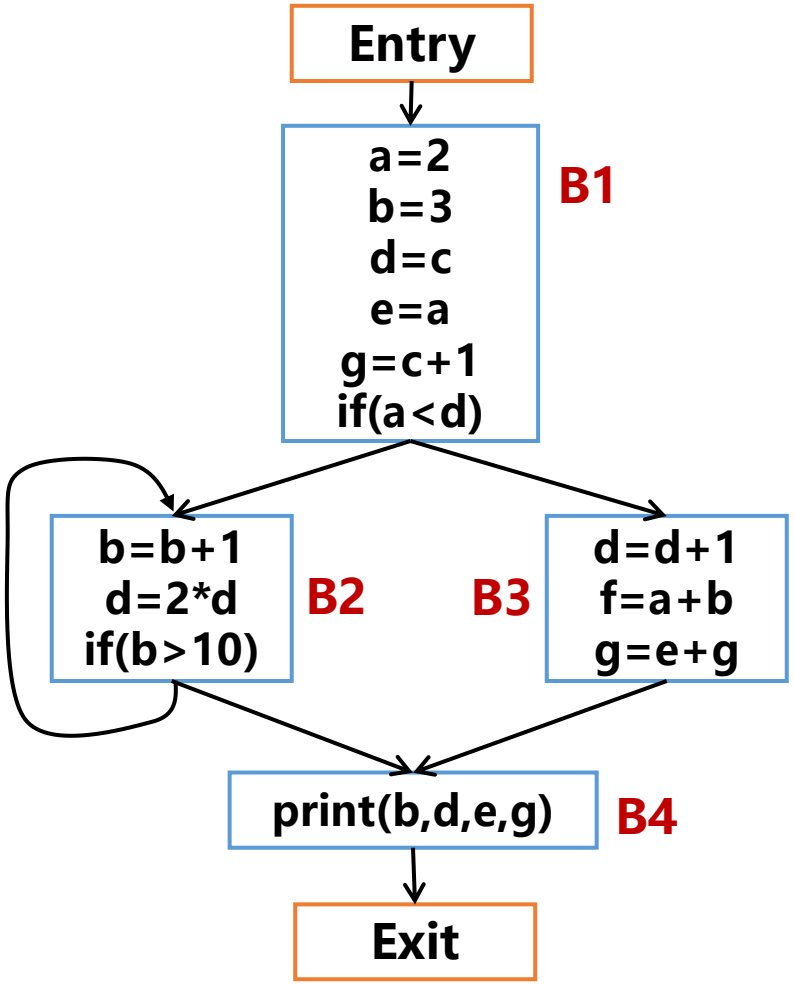
- 根据如下控制流图，分析各基本块边界处(入口处和出口处)的活跃变量情况，给出具体分析过程和最终结果



BB	use[B]	def[B]	out[B]	in[B]
Entry				
B1				
B2				
B3				
B4				
Exit				

练习

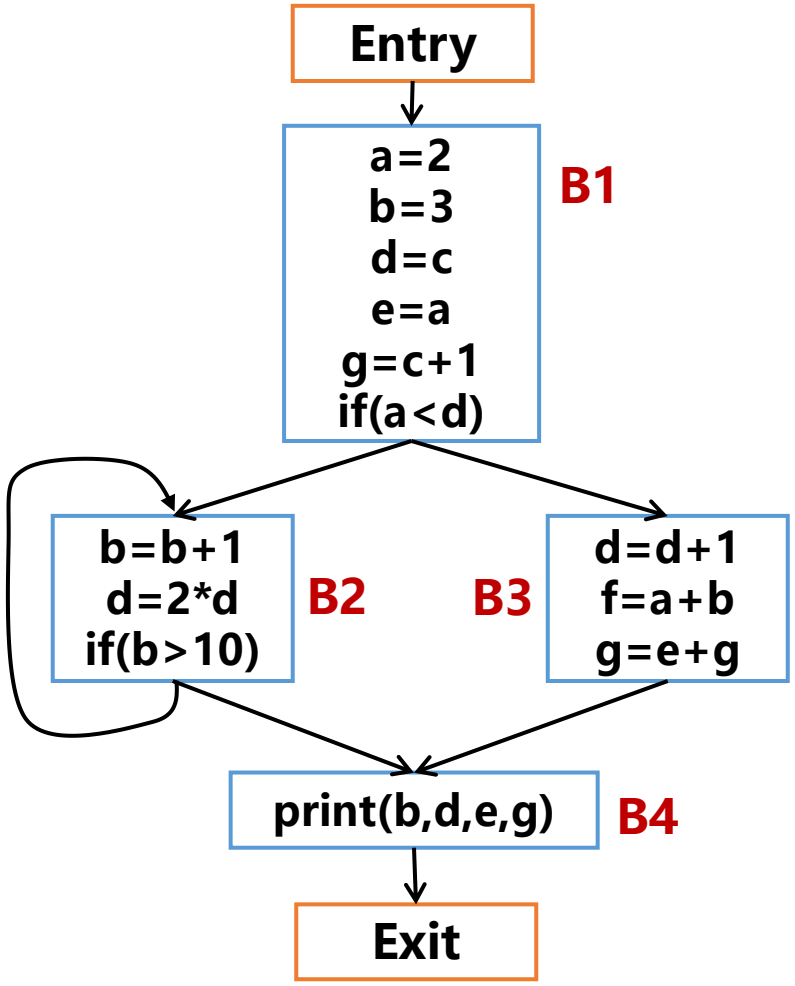
- 根据如下控制流图，分析各基本块边界处(入口处和出口处)的活跃变量情况，给出具体分析过程和最终结果



BB	use[B]	def[B]	out[B]	in[B]
Entry	∅	∅		
B1	{c}	{a,b,d,e,g}		
B2	{b,d}	{b,d}		
B3	{a,b,d,e,g}	{d,f,g}		
B4	{b,d,e,g}	∅		
Exit	∅	∅		

练习

- 根据如下控制流图，分析各基本块边界处(入口处和出口处)的活跃变量情况，给出具体分析过程和最终结果



BB	use[B]	def[B]	out[B]	in[B]
Entry	∅	∅	{c}	{c}
B1	{c}	{a,b,d,e,g}	{a,b,d,e,g}	{c}
B2	{b,d}	{b,d}	{b,d,e,g}	{b,d,e,g}
B3	{a,b,d,e,g}	{d,f,g}	{b,d,e,g}	{a,b,d,e,g}
B4	{b,d,e,g}	∅	∅	{b,d,e,g}
Exit	∅	∅	∅	∅

经过2次迭代后到达不动点

到达定值分析和活跃变量分析

- 到达-定值分析(Reaching Definitions)
  - 变量v有一次定值d，如果存在一条从该定值点开始的路径到达点p，并且沿着这条路径，定值d没有被其他定值杀死，那么就说定值d达到点p
  - 应用：常数传播、循环不变量外提
- 活跃变量分析(Live Variables)
  - 如果一个变量v在从点p开始的某条路径上被使用，那么变量v在点p是活跃的(live)
  - 应用：删除无用赋值、寄存器分配

学习内容

COMP
CCRG Open MP

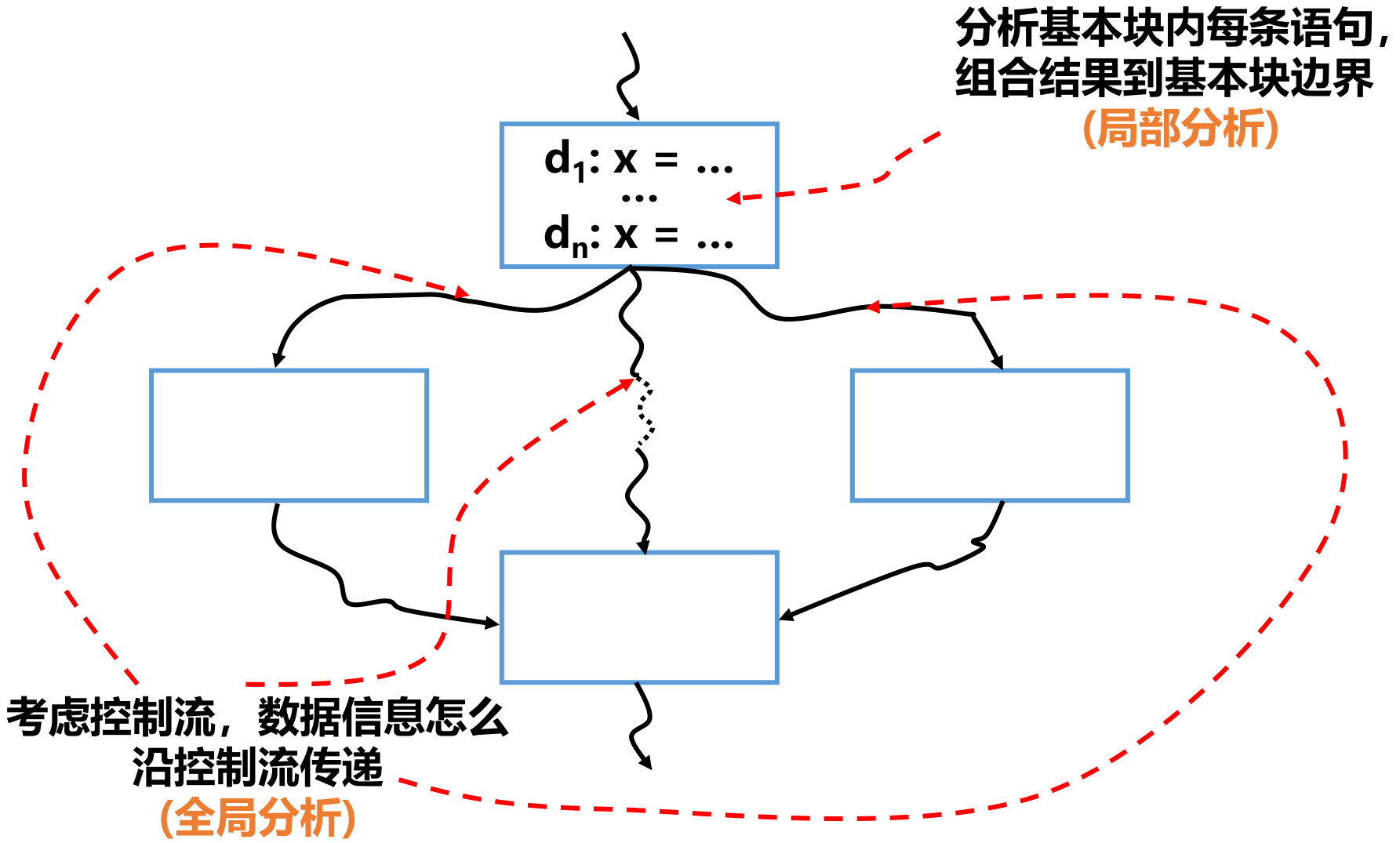
6.1 点和路径

6.2 到达定值分析

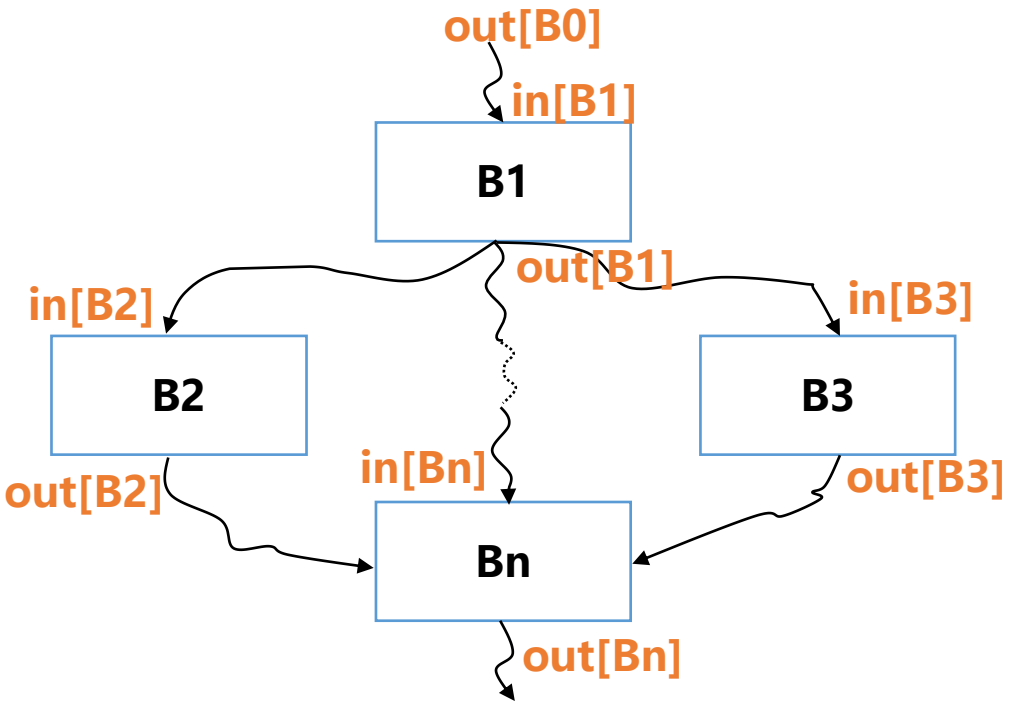
6.3 活跃变量分析

6.4 数据流分析框架

6.4.1 数据流分析基本方法



6.4.1 数据流分析基本方法



数据流方程的建立依赖于程序语句对求解的数据问题的影响

- 建立一组方程，对于每个基本块，求解出口点数据流信息集合out[B]和入口点数据流信息集合in[B]
  - 局部分析：基本块内语句对求解的数据流问题的影响

    转移函数f_B:

    in[B]→out[B]/out[B] → in[B]
  - 全局分析：控制流作用

    如果B1、B2有边连接，确立out[B1]和in[B2]之间的关系
- 寻找方法求解方程组（迭代求不动点）

6.4.1 数据流分析基本方法

- 前向分析: 在点p计算过去的行为产生的数据流信息
  - 计算CFG中基本块的前驱结点
  - 例如: 到达-定值分析
- 后向分析: 在点p计算未来行为的数据流信息
  - 计算CFG中基本块的后继结点
  - 例如: 活跃变量分析

6.4.2 迭代的数据流分析框架

- 输入: 一个由下列部分组成的数据流框架
  - 一个控制流图，包含两个特殊的结点Entry和Exit
  - 数据流方向D (前向分析/后向分析)
  - 一个值集V （定值/变量/表达式）
  - 一个交汇运算∧ （并集∪/交集∩）
  - 一个传递函数的集合F，其中f_B表示基本块B的传递函数
  - V中一个常量值V_{entry}或V_{exit}，分别表示前向和后向数据流分析框架的边界条件
- 输出: 各个基本块B边界处V的值 (in[B]和out[B])

6.4.2 迭代的数据流分析框架

	到达定值分析	活跃变量分析
数据流方向D	前向	后向
值集V	定值的集合	变量的集合
交汇运算∧	∪	∪
传递函数F	$f_B(x) = gen[B] \cup (x - kill[B])$	$f_B(x) = use[B] \cup (x - def[B])$
数据流方程	$in[B] = \cup\ out[pred(B)]$ $out[B] = f_B(in[B])$	$out[B] = \cup\ in[succ(B)]$ $in[B] = f_B(out[B])$
边界条件	$out[entry] = \varnothing$	$in[exit] = \varnothing$
初始值	$out[B] = \varnothing$	$in[B] = \varnothing$

6.4.2 迭代的数据流分析框架

```
out[Entry] = V_entry;
for (each B∈N-{Entry})
   out[B] = T //初值
while(change to any out occur){
   for(each B∈N-Entry){
      in[B] = ∧_{p∈pred[B]} out[P];
      out[B] = f_B(in[B]);
   }
}
```

前向数据流分析的迭代算法

```
in[Exit] = V_exit;
for (each B∈N-{Exit})
   in[B] = T //初值
while(change to any in occur){
   for(each B∈N-Exit){
      out[B] = ∧_{s∈succ[B]} in[S];
      in[B] = f_B(out[B]);
   }
}
```

后向数据流分析的迭代算法

其他数据流分析问题

- 常数传播
- 可用表达式
- 公用子表达式删除
- 值编号
- 循环不变量外提
- 死代码删除，激进的死代码删除

本讲小结

- 点和路径
  - 路径是点p_1, p_2, …, p_n 组成的序列

- 到达定值分析: 求解各基本块边界的定值集合

计算gen[B]和kill[B] → 建立数据流方程
$f_B(x) = gen[B] \cup (x - kill[B])$
$in[B] = \cup\ out[pred(B)]$
$out[B] = f_B(in[B])$ → 使用不动点法求解

- 活跃变量分析: 求解各基本块边界的活跃变量集合

计算use[B]和def[B] → 建立数据流方程
$f_B(x) = use[B] \cup (x - def[B])$
$out[B] = \cup\ in[succ(B)]$
$in[B] = f_B(out[B])$ → 使用不动点法求解

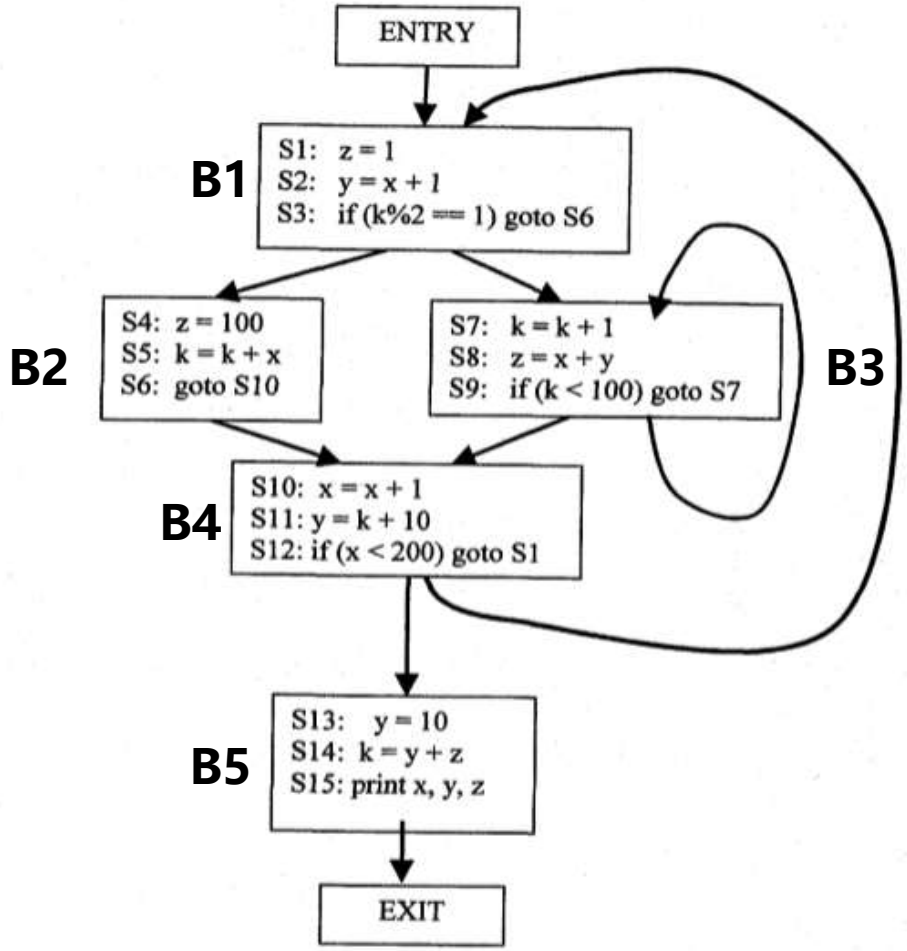
本讲小结

- 数据流分析框架
  - 定义通用的数据流分析框架(D, V, ∧, F)
  - 数据流分析基本方法
    - 局部分析：在基本块内确定转移函数f_B
    - 全局分析：考虑控制流，确定in[B]和out[pred(B)]或out[B]和in[succ(B)]之间的关系

    建立数据流方程
    - 通过迭代算法求解数据流方程，找到不动点，得到in[B]和out[B]

作业

- 对如下控制流图，进行活跃变量分析，给出各基本块边界处(入口处和出口处)的活跃变量情况，给出具体分析过程和最终结果



参考资料

- 《编译原理》(龙书) 第9章
- 《高级编译器设计与实现》(鲸书) 第8章