



# 机器无关代码优化

## Part2: 数据流与可用表达式分析

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

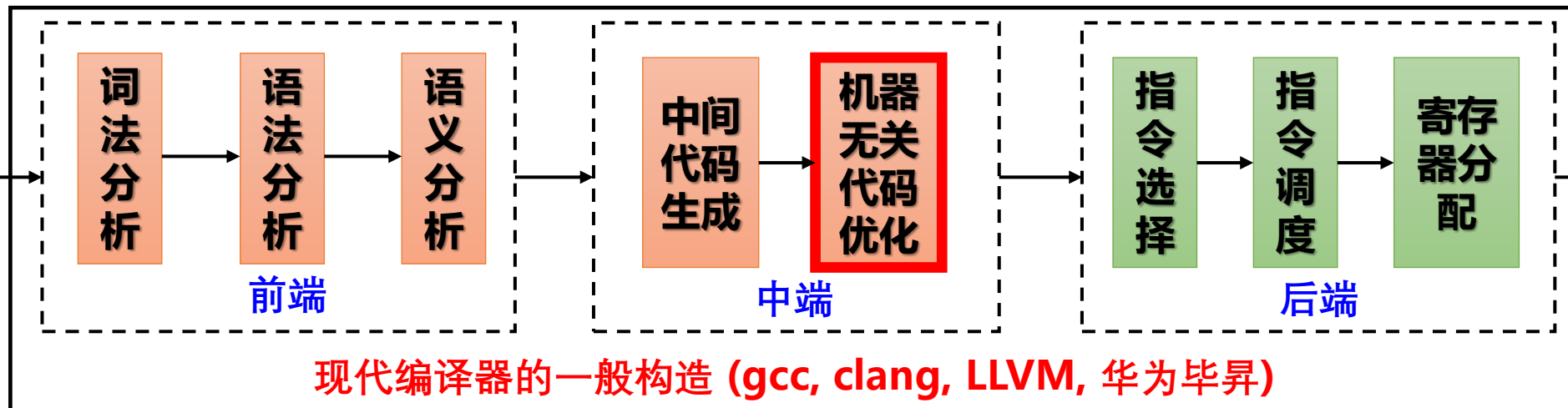
2023年11月15日



# 本节提纲



程序员编  
写的  
源程序



机器硬件上  
运行的  
目标代码



- 可用表达式的定义和简单计算
- 可用表达式分析概述及算法介绍
- 可用表达式分析示例



# 可用表达式



$x = y + z$

•

•

•

$p$

$y + z$  在  $p$  点  
可用

$x = y + z$

•

$y = \dots$

•

$p$

$y + z$  在  $p$  点  
不可用

$x = y + z$

•

$z = \dots$

•

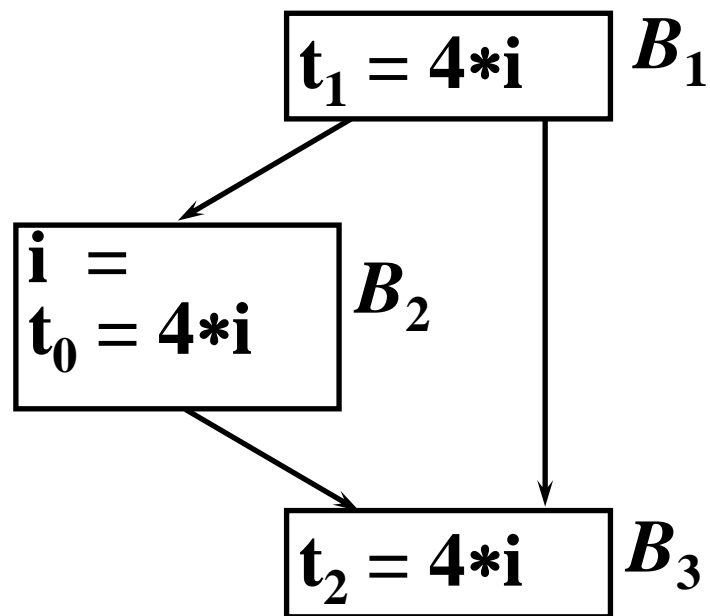
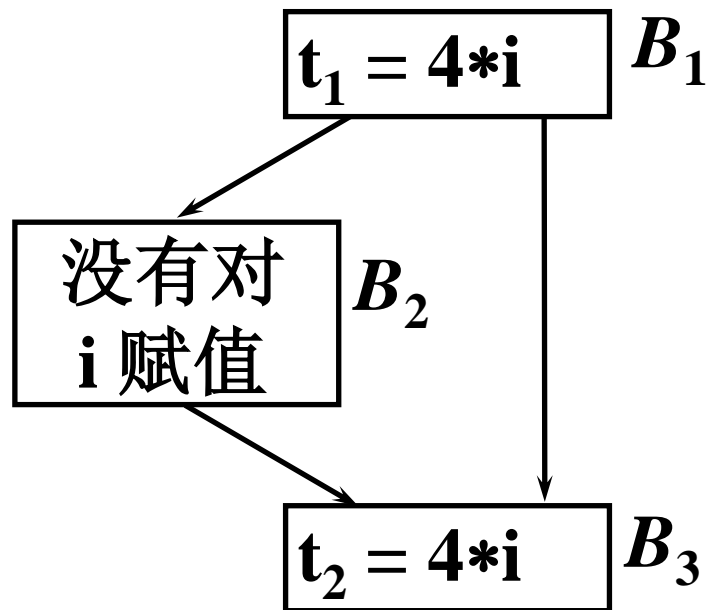
$p$

$y + z$  在  $p$  点  
不可用



## 消除全局公共子表达式

- 例：下面两种情况下， $4*i$ 在 $B_3$ 的入口都可用





- **基本块生成的表达式：**

**基本块中语句d:  $x = y + z$ 的前、后点分别为点p与点q。设在点p处可用表达式集合为S（基本块入口点处S为空集），那么经过语句d之后，在点q处可用表达式集合如下构成：**

**(1)  $S = S \cup \{ y+z \}$**

**(2)  $S = S - \{ S \text{ 中所有涉及变量} x \text{的表达式} \}$**

**注意，步骤(1)和(2)不可颠倒**



- **基本块生成的表达式：**

**基本块中语句d:  $x = y + z$ 的前、后点分别为点p与点q。设在点p处可用表达式集合为S（基本块入口点处S为空集），那么经过语句d之后，在点q处可用表达式集合如下构成：**

**(1)  $S = S \cup \{ y+z \}$**

**(2)  $S = S - \{ S \text{ 中所有涉及变量}x\text{的表达式} \}$**

**注意，步骤(1)和(2)不可颠倒，x可能就是y或z。**

**如此处理完基本块中所有语句后，可以得到基本块生成的可用表达式集合S；**



- 基本块生成的表达式:

基本块中语句 $d: x = y + z$ 的前、后点分别为点 $p$ 与点 $q$ 。设在点 $p$ 处可用表达式集合为 $S$  (基本块入口点处 $S$ 为空集), 那么经过语句 $d$ 之后, 在点 $q$ 处可用表达式集合如下构成:

$$(1) S = S \cup \{ y+z \}$$

$$(2) S = S - \{ S \text{ 中所有涉及变量 } x \text{ 的表达式} \}$$

注意, 步骤(1)和(2)不可颠倒,  $x$ 可能就是 $y$ 或 $z$ 。

如此处理完基本块中所有语句后, 可以得到基本块生成的可用表达式集合 $S$ ;

- 基本块杀死的表达式: 所有其他类似 $y+z$ 的表达式, 基本块中对 $y$ 或 $z$ 定值, 但基本块没有**生成** $y+z$ 。



# 示例：基本块生成的表达式



语句	可用表达式
$a = b + c$	$\emptyset$
$b = a - d$	$\{ b + c \}$
$c = b + c$	$\{ a - d \} // b+c$ 被杀死
$d = a - d$	$\{ a - d \} // b+c$ 被杀死 $\emptyset // a - d$ 被杀死

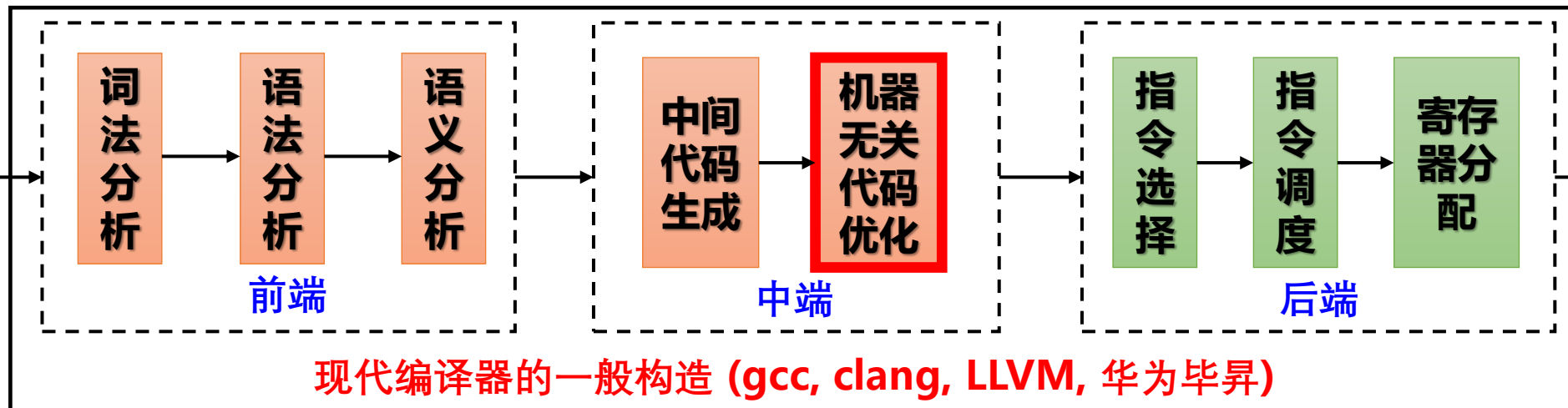




# 本节提纲



程序员编  
写的  
源程序



机器硬件上  
运行的  
目标代码



- 可用表达式的定义和简单计算
- 可用表达式分析概述及算法介绍
- 可用表达式分析示例



## • 定义

- 若到点 $p$ 的每条执行路径都计算 $x \text{ op } y$ , 并且计算后没有对 $x$ 或 $y$ 赋值, 那么称 $x \text{ op } y$ 在点 $p$ 可用
- $e\_gen_B$ : 块 $B$ 产生的可用表达式集合
- $e\_kill_B$ : 块 $B$ 注销的可用表达式集合
- $IN[B]$ : 块 $B$ 入口的可用表达式集合
- $OUT[B]$ : 块 $B$ 出口的可用表达式集合



## • 数据流等式

- $OUT [B] = e\_gen_B \cup (IN [B] - e\_kill_B)$
- $IN [B] = \cap_{P \text{ 是 } B \text{ 的前驱}} OUT [P]$
- $OUT [ENTRY] = \emptyset$ 
  - 在ENTRY的出口处没有可用表达式

## • 同先前的主要区别

- 使用 $\cap$ 而不是 $\cup$ 作为这里数据流等式的汇合算符
- 只有当一个表达式在B的所有前驱的结尾处都可用，那么它才会在B的开头可用
- 求最大解而不是最小解



- 迭代算法:

$U$ 是全体表达式集合

(1)  $OUT[ENTRY] = \emptyset$

(2) for( 除ENTRY之外的每个基本块B)  $OUT[B] = U$

(3) while(某个OUT值发生变化) {

(4)   for(除ENTRY之外的每个基本块B ){

(5)        $IN[B] = \bigcap_{P \text{ 是 } B \text{ 的前驱基本块}} (OUT[P])$

(6)        $OUT[B] = e\_gen_B \cup ( IN[B] - e\_kill_B )$

    } // end-of-for

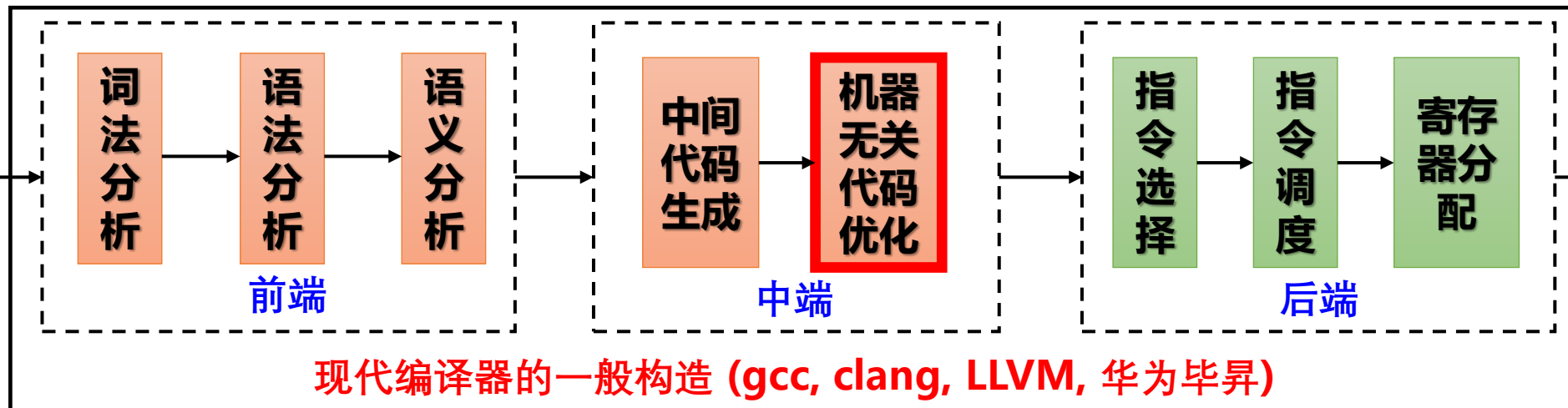
  } // end-of-while



# 本节提纲



程序员编  
写的  
源程序



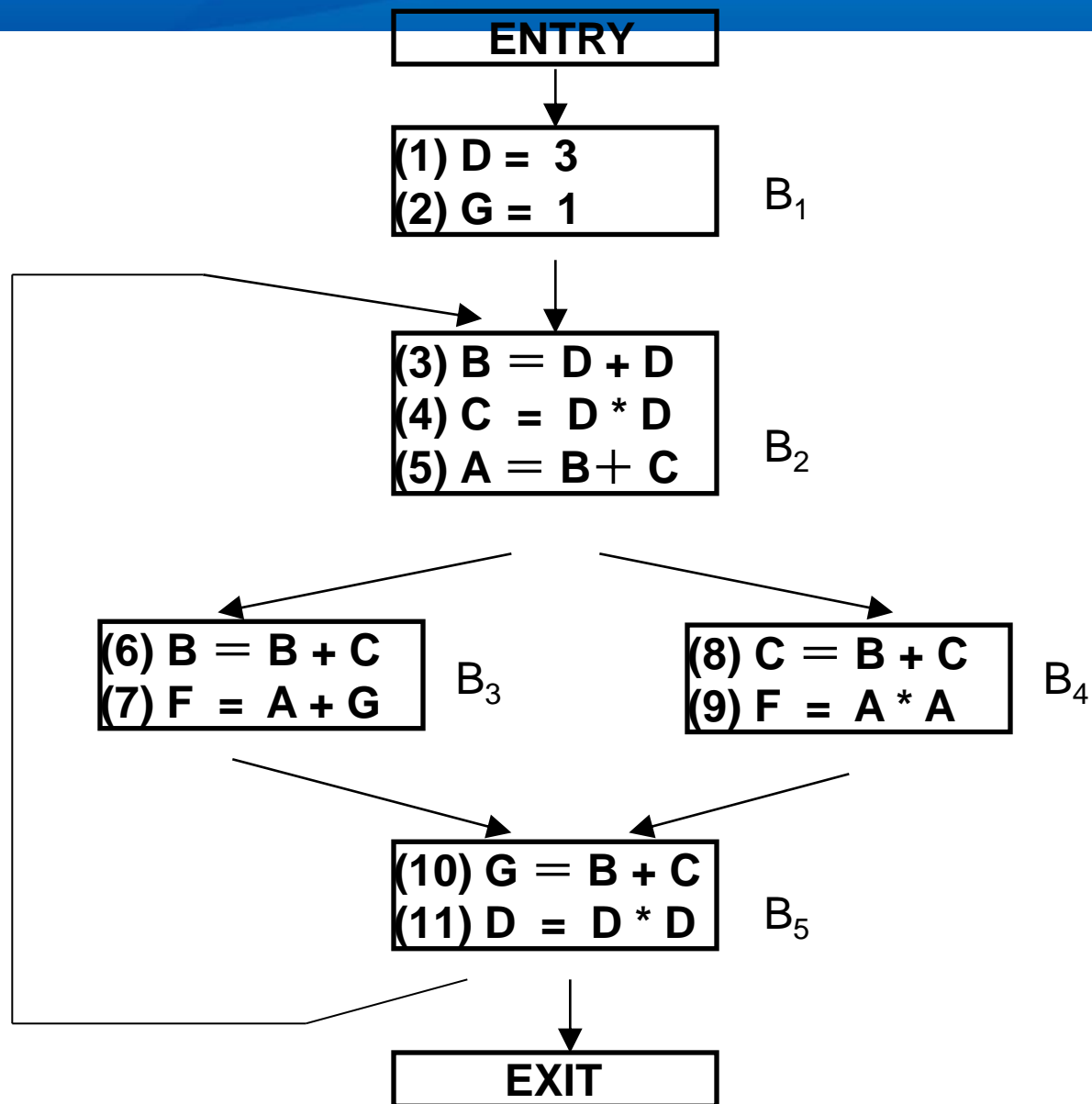
机器硬件上  
运行的  
目标代码



- 可用表达式的定义和简单计算
- 可用表达式分析概述及算法介绍
- 可用表达式分析示例



# 示例：可用表达式分析





# 示例：可用表达式分析



基本块	前驱	后继
ENTRY	—	B <sub>1</sub>
B <sub>1</sub>	ENRITY	B <sub>2</sub>
B <sub>2</sub>	B <sub>1</sub> B <sub>5</sub>	B <sub>3</sub> B <sub>4</sub>
B <sub>3</sub>	B <sub>2</sub>	B <sub>5</sub>
B <sub>4</sub>	B <sub>2</sub>	B <sub>5</sub>
B <sub>5</sub>	B <sub>3</sub> B <sub>4</sub>	B <sub>2</sub> EXIT
EXIT	B <sub>5</sub>	—



# 示例：可用表达式分析



基本块	e_gen	e_kill
ENTRY	$\emptyset$	$\emptyset$
B <sub>1</sub>	{3, 1}	{ D+D, D*D, A+G }
B <sub>2</sub>	{ D+D, D*D, B+C }	{ A*A, A+G }
B <sub>3</sub>	{ A+G }	{ B+C }
B <sub>4</sub>	{ A * A }	{ B+C }
B <sub>5</sub>	{ B+C }	{ A+G, D*D, D+D }
EXIT	$\emptyset$	$\emptyset$
全部表达式 $U=\{ 3, 1, D+D, D*D, B+C, A+G, A*A \}$		





# 示例：可用表达式分析



基本块		e_kill
ENTRY		$\emptyset$
B <sub>1</sub>		{ D+D, D*D, A+G }
B <sub>2</sub>		{ A*A, A+G }
B <sub>3</sub>	{ A+G }	{ B+C }
B <sub>4</sub>	{ A * A }	{ B+C }
B <sub>5</sub>	{ B+C }	{ A+G, D*D, D+D }
EXIT	$\emptyset$	$\emptyset$
全部表达式 $U = \{ 3, 1, D+D, D*D, B+C, A+G, A*A \}$		

- B2块的e\_kill集合不包含B+C，因为虽然B和C的赋值改变了B+C的值，但是最后一个语句再次计算了B+C，这样B+C又成为可用表达式。生命力顽强，没有被kill掉。
- 从另一个视角来看，即便是e\_kill中包含了B+C，OUT集合计算的时候也会被e\_gen中的B+C覆盖掉。

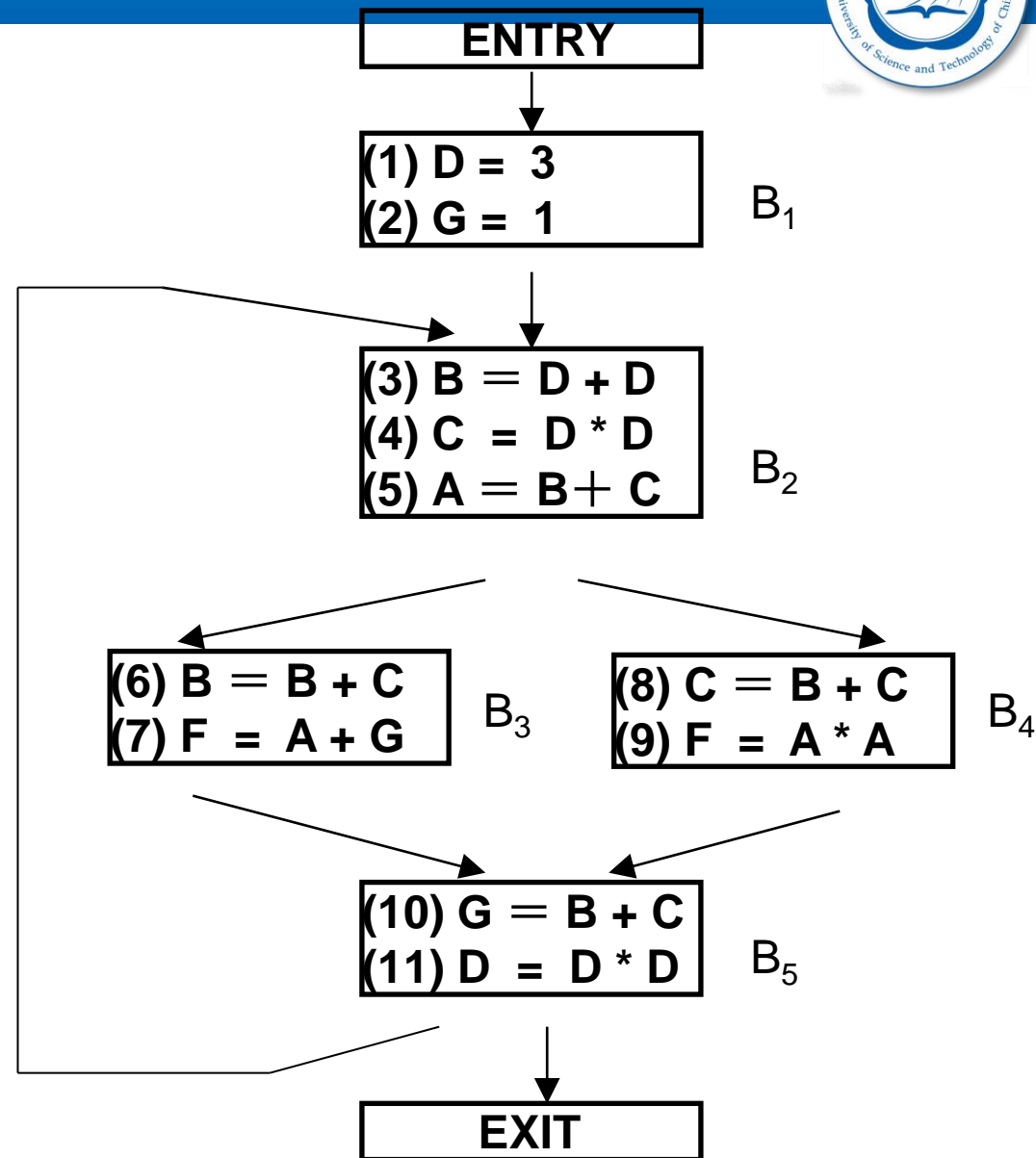


# 示例：可用表达式分析



## • 可用表达式的迭代计算

- 深度优先序，即  $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4 \rightarrow B_5 \rightarrow \text{EXIT}$
- 边界值： $\text{OUT}[\text{ENTRY}] = \emptyset$ ;
- 初始化：for all NON-ENTRY B:  
 $\text{OUT}[B] = U$ ;





# 示例：可用表达式分析



## • 第一次迭代：(all NON-ENTRY B)

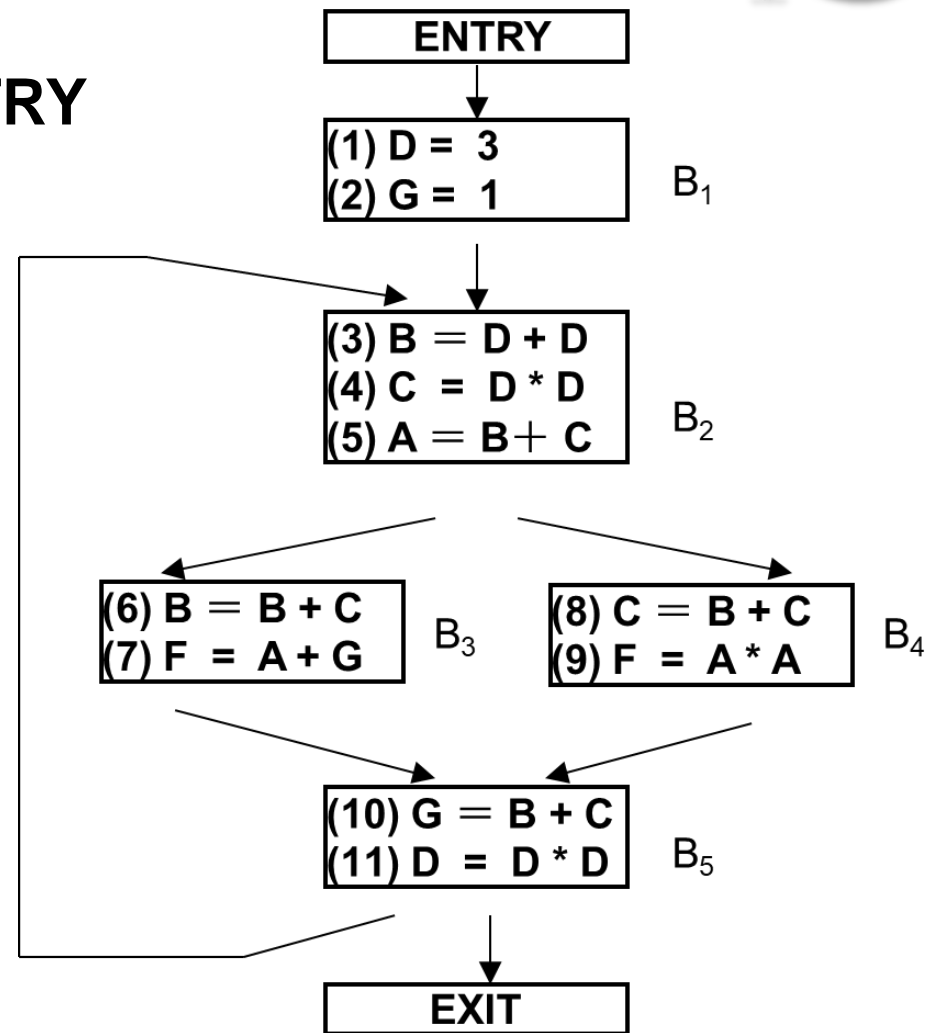
(1)  $IN[B1] = OUT[ENTRY] = \emptyset$ ; // B1 前驱仅为ENTRY

$$\begin{aligned} OUT[B1] &= e\_gen[B1] \cup (IN[B1] - e\_kill[B1]) \\ &= e\_gen[B1] = \{3, 1\} // \text{变化} \end{aligned}$$

(2)  $IN[B2] = OUT[B1] \cap OUT[B5]$

$$= \{3, 1\} \cap U = \{3, 1\}$$

$$\begin{aligned} OUT[B2] &= e\_gen[B2] \cup (IN[B2] - e\_kill[B2]) \\ &= \{D+D, D*D, B+C\} \cup \\ &\quad (\{3, 1\} - \{A*A, A+G\}) \\ &= \{3, 1, D+D, D*D, B+C\} // \text{变化} \end{aligned}$$





# 示例：可用表达式分析



## • 第一次迭代：(all NON-ENTRY B)

(3)  $IN[B3] = OUT[B2]$

$= \{3, 1, D+D, D*D, B+C\}$

$OUT[B3] = e\_gen[B3] \cup (IN[B3] - e\_kill[B3])$

$= \{A+G\} \cup (\{3, 1, D+D, D*D, B+C\} - \{B+C\})$

$= \{3, 1, D+D, D*D, A+G\}$  //变化

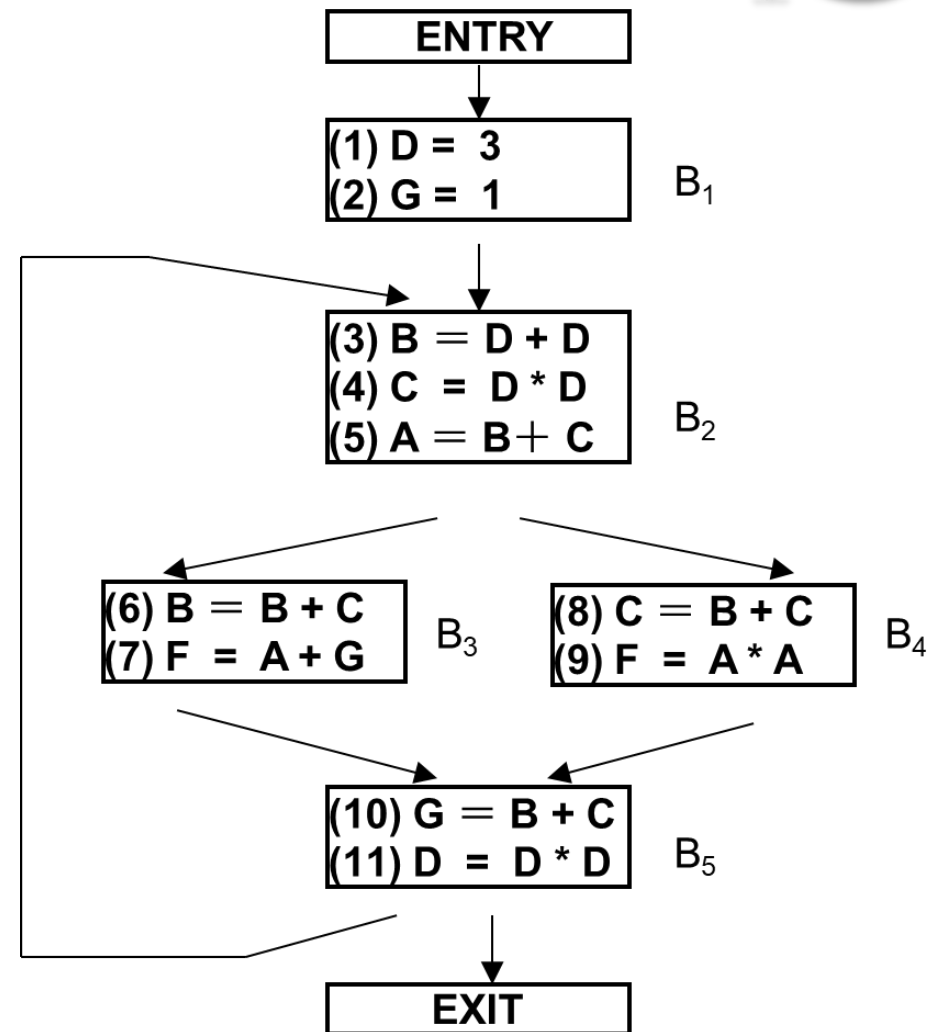
(4)  $IN[B4] = OUT[B2]$

$= \{3, 1, D+D, D*D, B+C\}$

$OUT[B4] = e\_gen[B4] \cup (IN[B4] - e\_kill[B4])$

$= \{A * A\} \cup (\{3, 1, D+D, D*D, B+C\} - \{B+C\})$

$= \{3, 1, D+D, D*D, A * A\}$  //变化





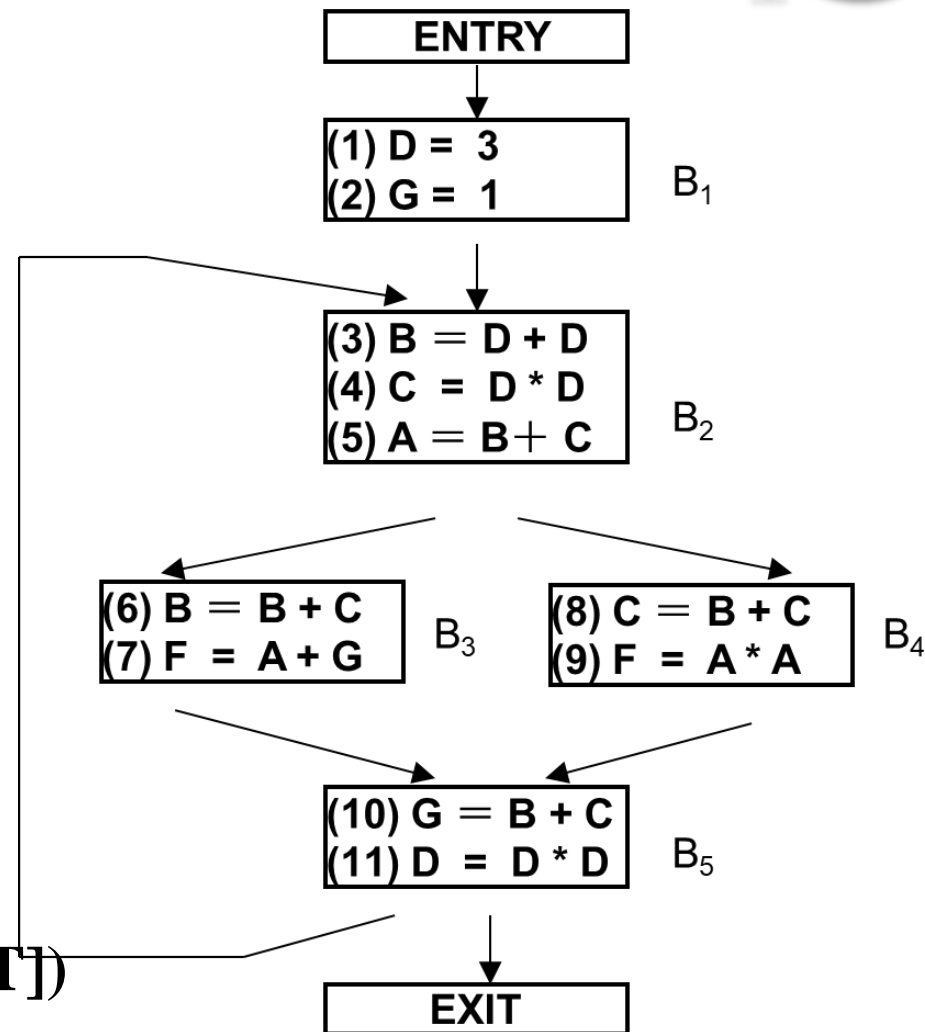
# 示例：可用表达式分析



## • 第一次迭代：(all NON-ENTRY B)

$$\begin{aligned}
 (5) \text{ IN}[B5] &= \text{OUT}[B3] \cap \text{OUT}[B4] \\
 &= \{ 3, 1, D+D, D*D, A+G \} \cap \\
 &\quad \{ 3, 1, D+D, D*D, A * A \} \\
 &= \{ 3, 1, D+D, D*D \} \\
 \text{OUT}[B5] &= \text{e\_gen}[B5] \cup ( \text{IN}[B5] - \text{e\_kill}[B5] ) \\
 &= \{ B+C \} \cup ( \{ 3, 1, D+D, D*D \} - \\
 &\quad \{ A+G, D*D, D+D \} ) \\
 &= \{ 3, 1, B+C \} \text{ //变化}
 \end{aligned}$$

$$\begin{aligned}
 (6) \text{ IN}[\text{EXIT}] &= \text{OUT}[B5] = \{ 3, 1, B+C \} \\
 \text{OUT}[\text{EXIT}] &= \text{e\_gen}[\text{EXIT}] \cup \\
 &\quad ( \text{IN}[\text{EXIT}] - \text{e\_kill} [\text{EXIT}] ) \\
 &= \emptyset \cup ( \{ 3, 1, B+C \} - \emptyset ) \\
 &= \{ 3, 1, B+C \} \text{ //变化}
 \end{aligned}$$





# 示例：可用表达式分析



## • 第二次迭代：(all NON-ENTRY B)

$$(1) \text{IN}[B1] = \text{OUT}[\text{ENTRY}] = \emptyset;$$

$$\begin{aligned} \text{OUT}[B1] &= e\_gen[B1] \cup (\text{IN}[B1] - e\_kill[B1]) \\ &= e\_gen[B1] = \{ 3, 1 \} // \text{不变} \end{aligned}$$

$$(2) \text{IN}[B2] = \text{OUT}[B1] \cap \text{OUT}[B5]$$

$$= \{ 3, 1 \} \cap \{ 3, 1, B+C \} = \{ 3, 1 \} // \text{不变}$$

$$\text{OUT}[B2] = e\_gen[B2] \cup (\text{IN}[B2] - e\_kill[B2])$$

$$\begin{aligned} &= \{ D+D, D*D, B+C \} \cup (\{ 3, 1 \} - \{ A*A, A+G \}) \\ &= \{ 3, 1, D+D, D*D, B+C \} // \text{不变} \end{aligned}$$



# 示例：可用表达式分析



## • 第二次迭代：(all NON-ENTRY B)

$$\begin{aligned} (3) \text{ IN}[B3] &= \text{OUT}[B2] \\ &= \{3, 1, D+D, D*D, B+C\} \text{ //不变} \\ \text{OUT}[B3] &= \text{e\_gen}[B3] \cup (\text{IN}[B3] - \text{e\_kill}[B3]) \\ &= \{A+G\} \cup (\{3, 1, D+D, D*D, B+C\} - \{B+C\}) \\ &= \{3, 1, D+D, D*D, A+G\} \text{ //不变} \end{aligned}$$
  
$$\begin{aligned} (4) \text{ IN}[B4] &= \text{OUT}[B2] \\ &= \{3, 1, D+D, D*D, B+C\} \text{ //不变} \\ \text{OUT}[B4] &= \text{e\_gen}[B4] \cup (\text{IN}[B4] - \text{e\_kill}[B4]) \\ &= \{A * A\} \cup (\{3, 1, D+D, D*D, B+C\} - \{B+C\}) \\ &= \{3, 1, D+D, D*D, A * A\} \text{ //不变} \end{aligned}$$



# 示例：可用表达式分析



## • 第二次迭代：(all NON-ENTRY B)

$$(5) \text{IN}[B5] = \text{OUT}[B3] \cap \text{OUT}[B4]$$

$$= \{ 3, 1, D+D, D*D, A+G \} \cap \{ 3, 1, D+D, D*D, A * A \}$$

$$= \{ 3, 1, D+D, D*D \} \text{ //不变}$$

$$\text{OUT}[B5] = \text{e\_gen}[B5] \cup ( \text{IN}[B5] - \text{e\_kill}[B5] )$$

$$= \{ B+C \} \cup ( \{ 3, 1, D+D, D*D \} - \{ A+G, D*D, D+D \} )$$

$$= \{ 3, 1, B+C \} \text{ //不变}$$

$$(6) \text{IN}[\text{EXIT}] = \text{OUT}[B5] = \{ 3, 1, B+C \} \text{ //不变}$$

$$\text{OUT}[\text{EXIT}] = \text{e\_gen}[\text{EXIT}] \cup ( \text{IN}[\text{EXIT}] - \text{e\_kill}[\text{EXIT}] )$$

$$= \emptyset \cup ( \{ 3, 1, B+C \} - \emptyset )$$

$$= \{ 3, 1, B+C \} \text{ //不变}$$





# 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

2023年11月15日