



寄存器分配

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

2025年11月17日



程序如何在计算机上执行的？



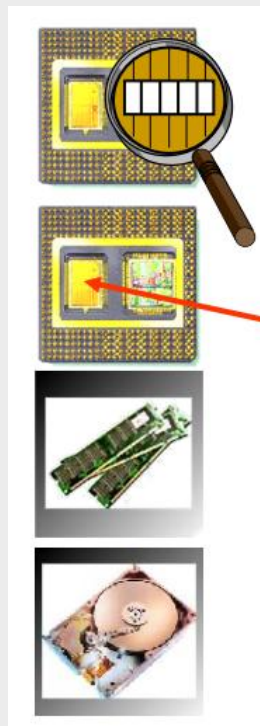
计算单元
(龙芯CPU)

寄存器

缓存

内存

磁盘



存储单元
(长鑫内存、国科微固态硬盘)

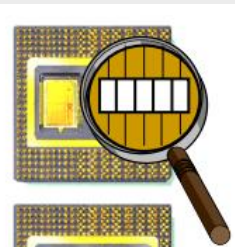


程序如何在计算机上执行的?

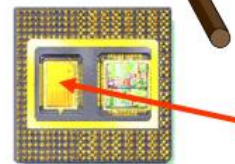


计算单元
(龙芯CPU)

寄存器



缓存



内存



磁盘



读取速度
(指令周期)

1

3

20

5M

容量
(字节)

8K

40M

512G

10T

存储单元
(长鑫内存、国科微固态硬盘)

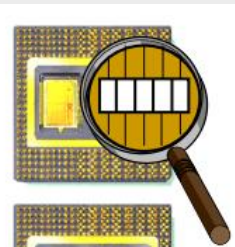


程序如何在计算机上执行的?

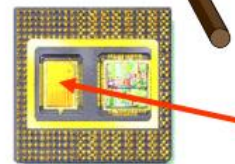


计算单元
(龙芯CPU)

寄存器



缓存



内存



磁盘



代码 数据

读取速度
(指令周期)

1

3

20

5M

容量
(字节)

8K

40M

512G

10T

存储单元
(长鑫内存、国科微固态硬盘)



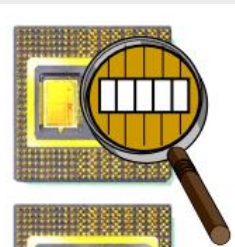
程序如何在计算机上执行的?



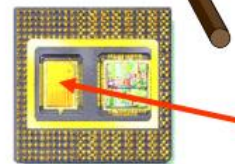
代码

计算单元
(龙芯CPU)

寄存器



缓存



内存



磁盘



代码

代码

代码 数据

读取速度
(指令周期)

1

3

20

5M

容量
(字节)

8K

40M

512G

10T

存储单元
(长鑫内存、国科微固态硬盘)



程序如何在计算机上执行的？



代码

计算单元
(龙芯CPU)

			读取速度 (指令周期)	容量 (字节)
寄存器		数据	1	8K
缓存		代码 数据	3	40M
内存		代码 数据	20	512G
磁盘		代码 数据	5M	10T

存储单元
(长鑫内存、国科微固态硬盘)



寄存器资源管理十分重要



- 寄存器容量和个数十分有限
 - 受限于电源功耗等因素
- 为了编程简单，高级语言假设可以使用无限多个寄存器

架构	32位
ARM	15
Intel x86	8
MIPS	32
RISC-V	16/32
LoongArch	32+32

龙芯

程序片段:

t1 = 0

t2 = t1 - 5

t3 = t1 + t2

t4 = t2 * t3

t5 = t3 - t4

... ..

t1

t2

.

.

.

t00

竞争
寄存器的使用

R1

R2

.

.

.

R32



寄存器分配



- **任务目标：**

- 在不改变程序行为的前提下，将同一个寄存器分配给多个变量

- **约束条件：**

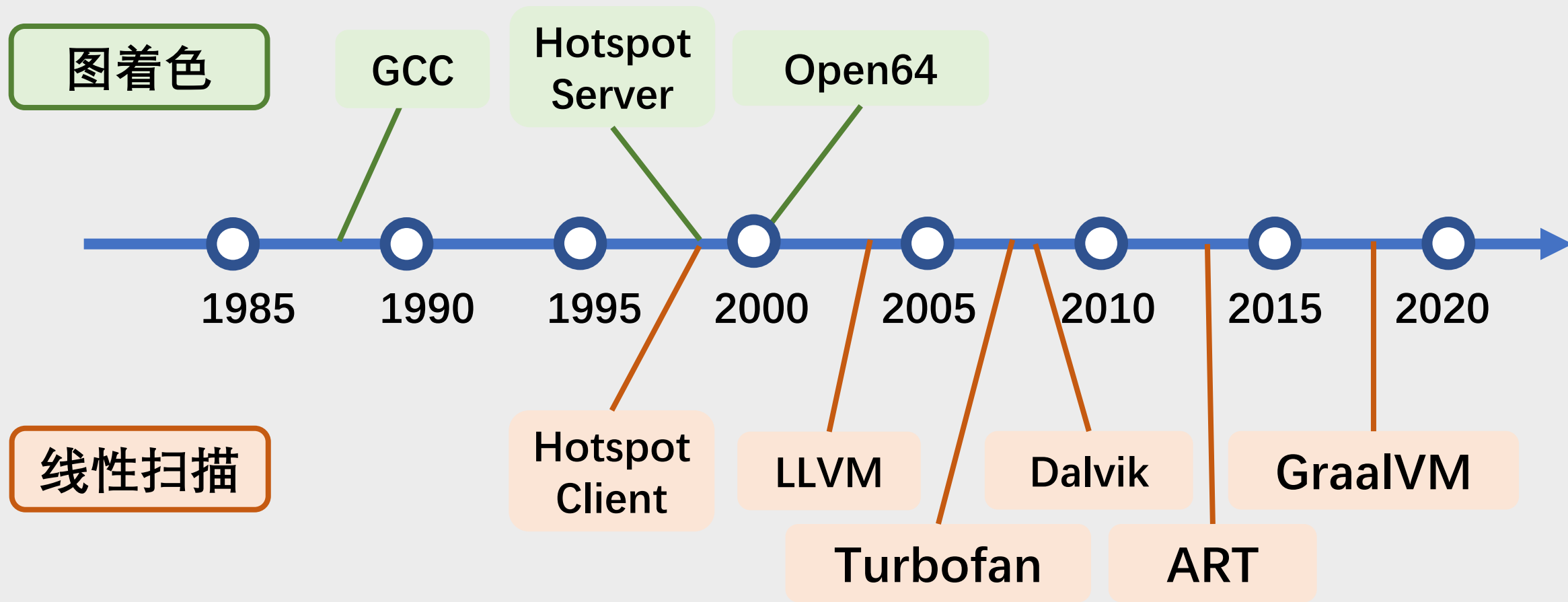
- 同一时刻，一个寄存器只能被一个变量占用
- 寄存器占满后，新的使用申请将选择一个寄存器，移出其所存储的变量，放回内存（成为**Spill**，产生较大的开销）
- 换入换出寄存器的开销尽可能小



寄存器分配算法的演进



分配效果非常**好**、但运行时间**长**、常见于传统编译器。



算法运行时间**很短**，分配效果**接近**图着色、常见于现代编译器。



举例——寄存器线性扫描分配



$e = d + a$

$f = b + c$

$f = f + b$

if $e == 0$ goto _L0

$d = e + f$

goto _L1

_L0: $d = e - f$

_L1: $g = d$

a

b

c

d

e

f

g



仅有4个可用的寄存器



寄存器线性扫描分配——变量存活区间



$e = d + a$

$f = b + c$

$f = f + b$

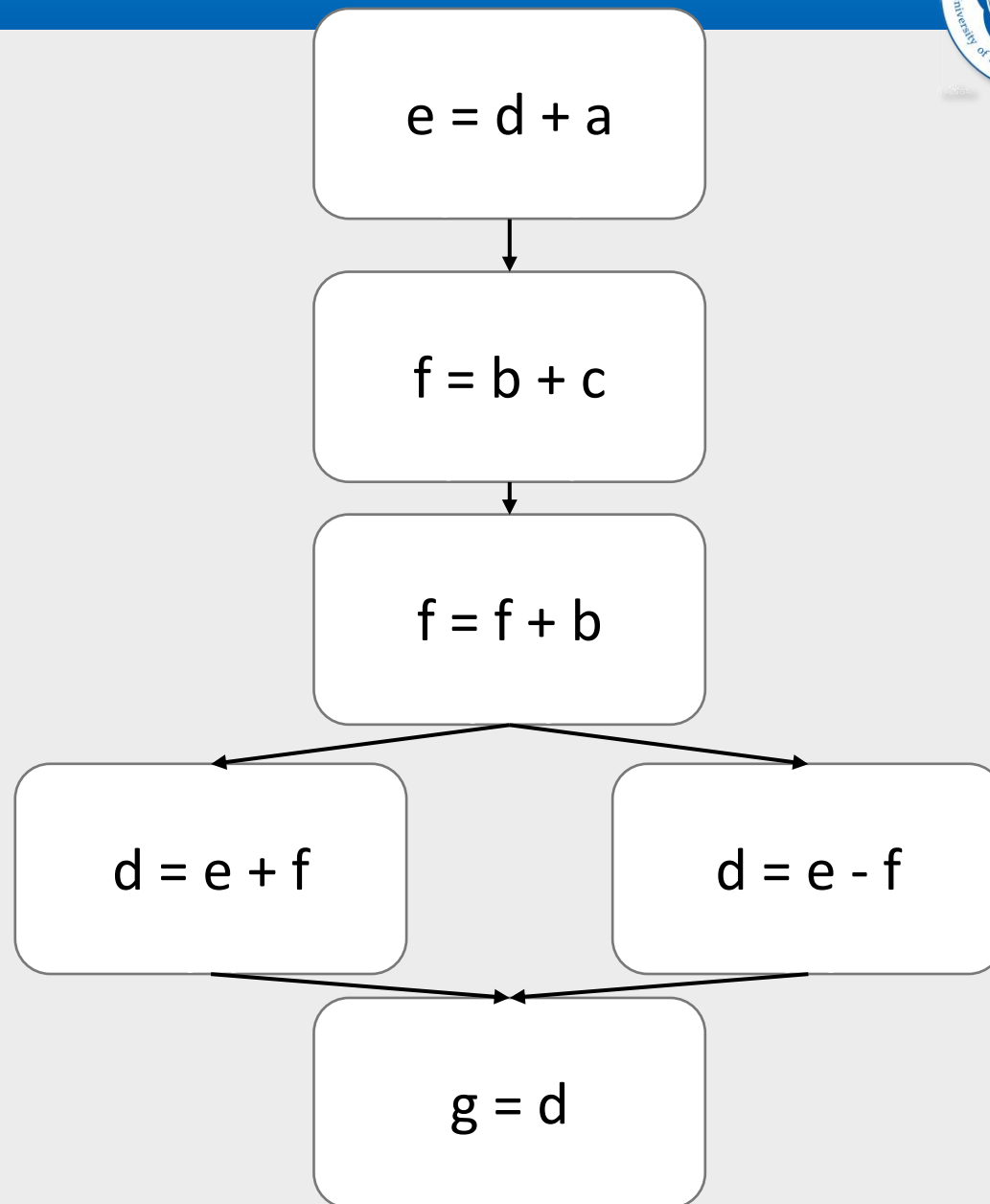
if $e == 0$ goto _L0

$d = e + f$

goto _L1

_L0: $d = e - f$

_L1: $g = d$





寄存器线性扫描分配——变量存活区间



$e = d + a$

$f = b + c$

$f = f + b$

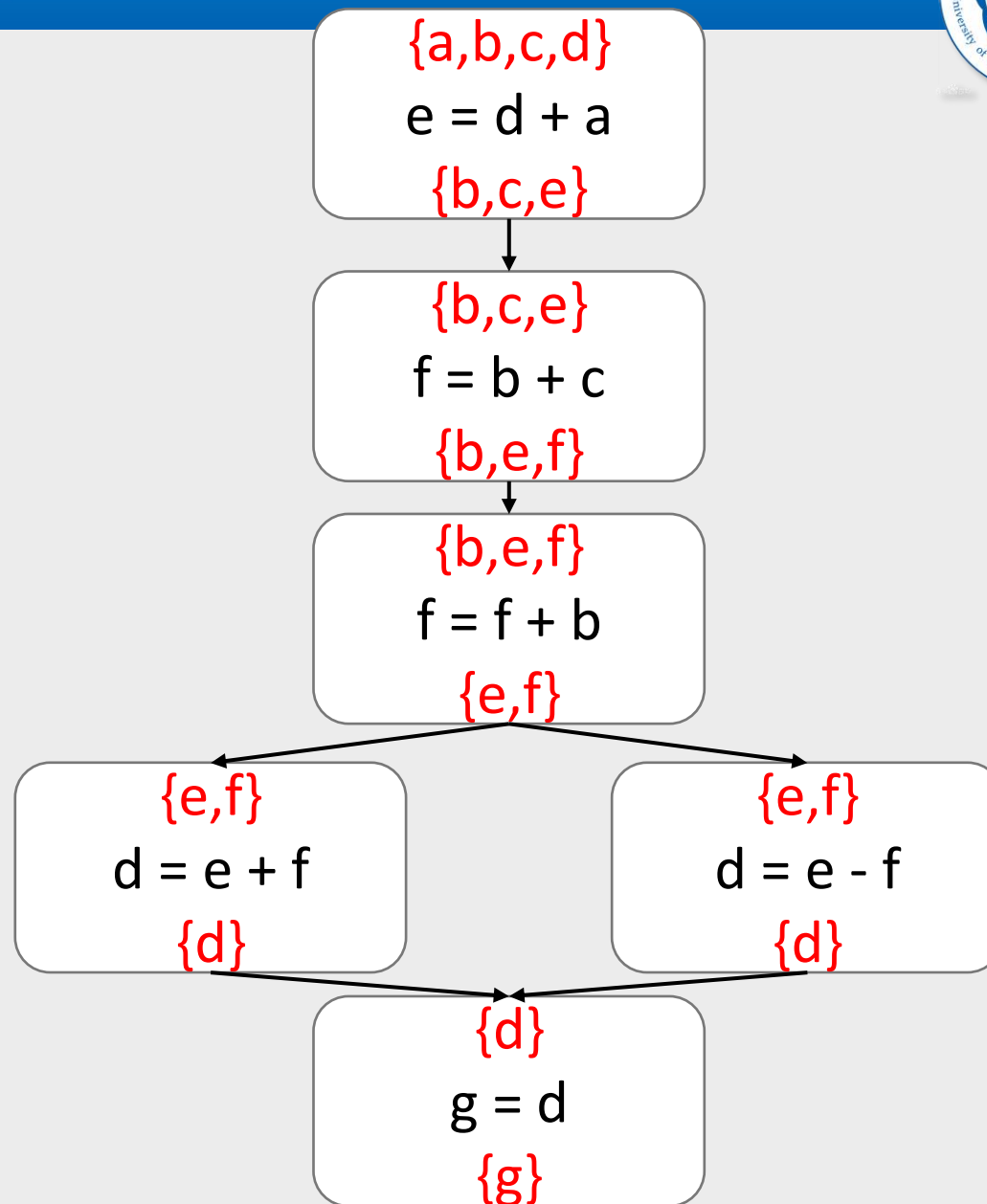
if $e == 0$ goto _L0

$d = e + f$

goto _L1

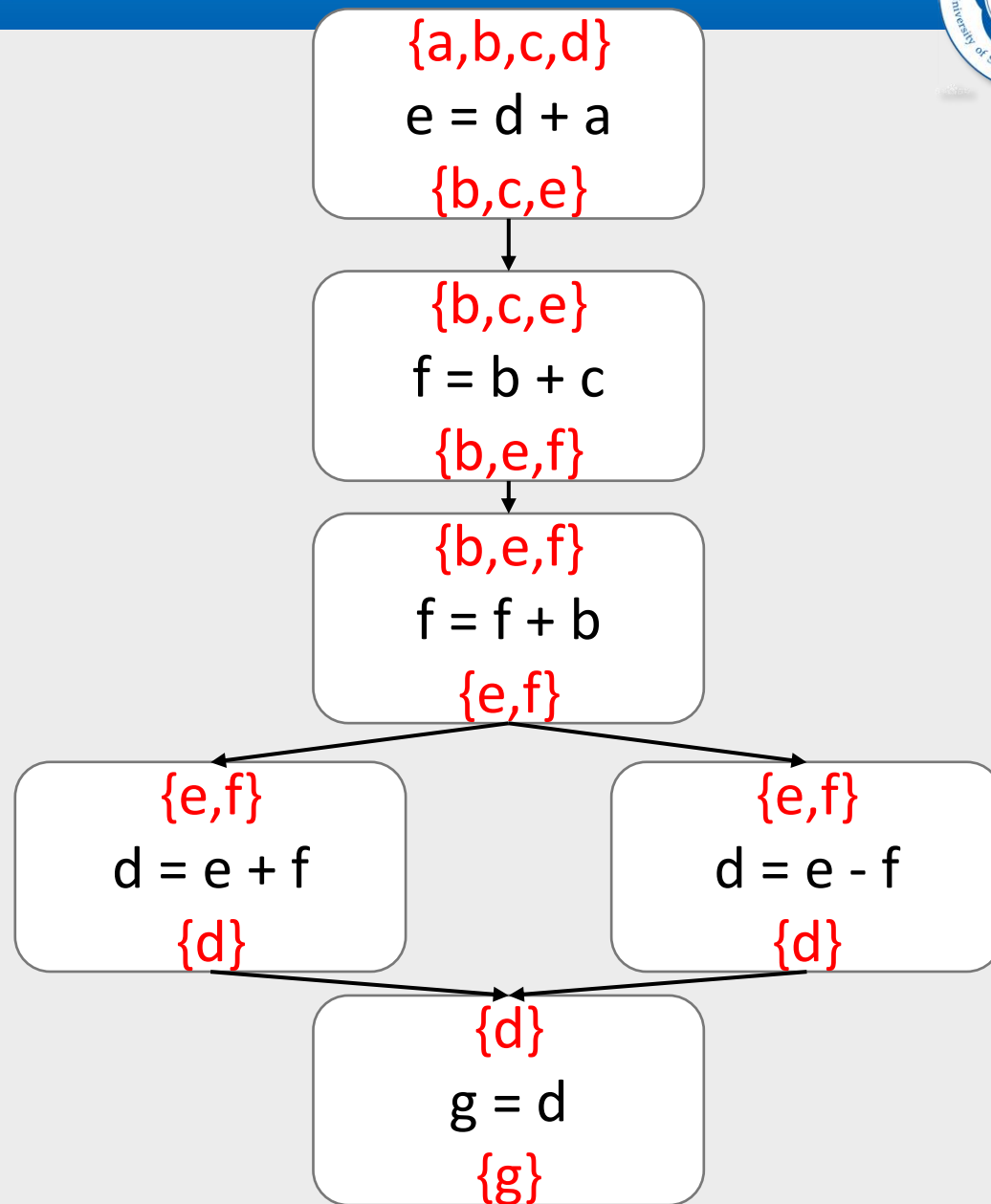
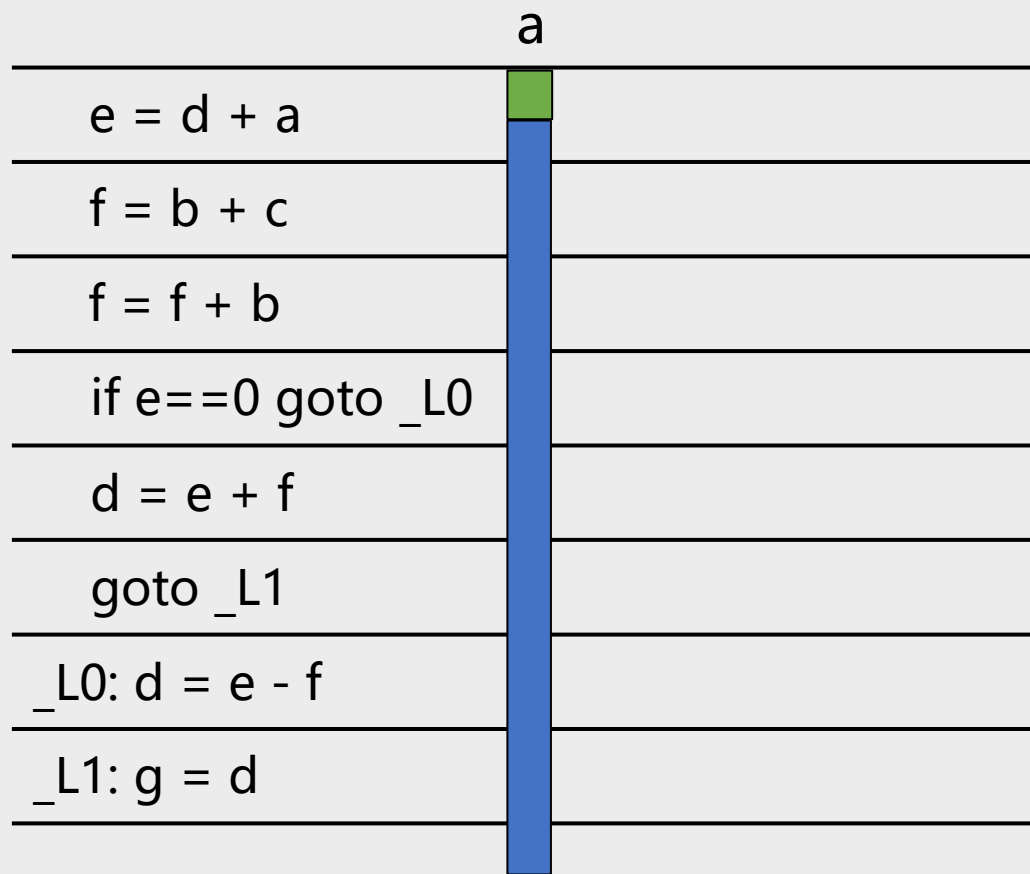
_L0: $d = e - f$

_L1: $g = d$



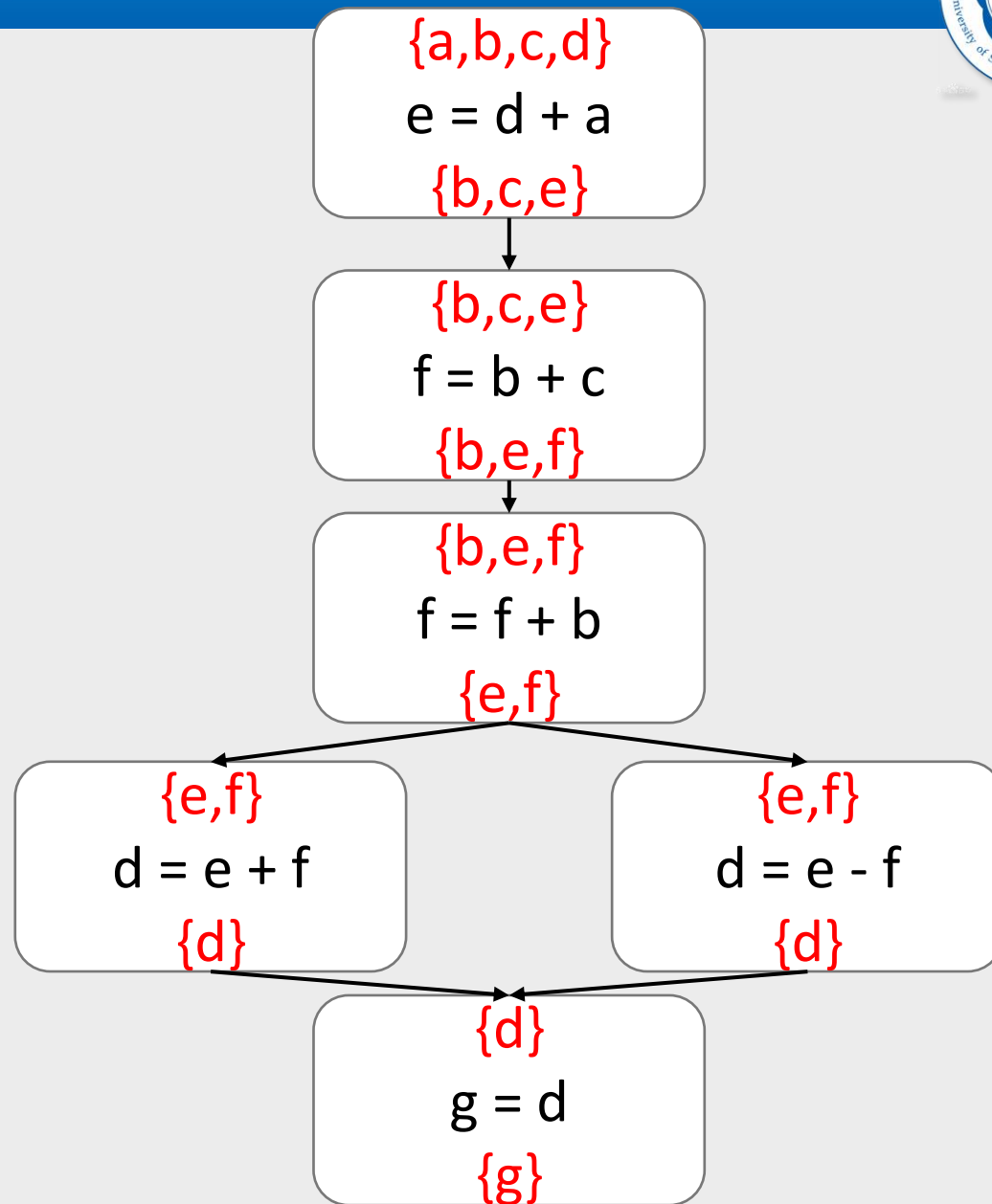
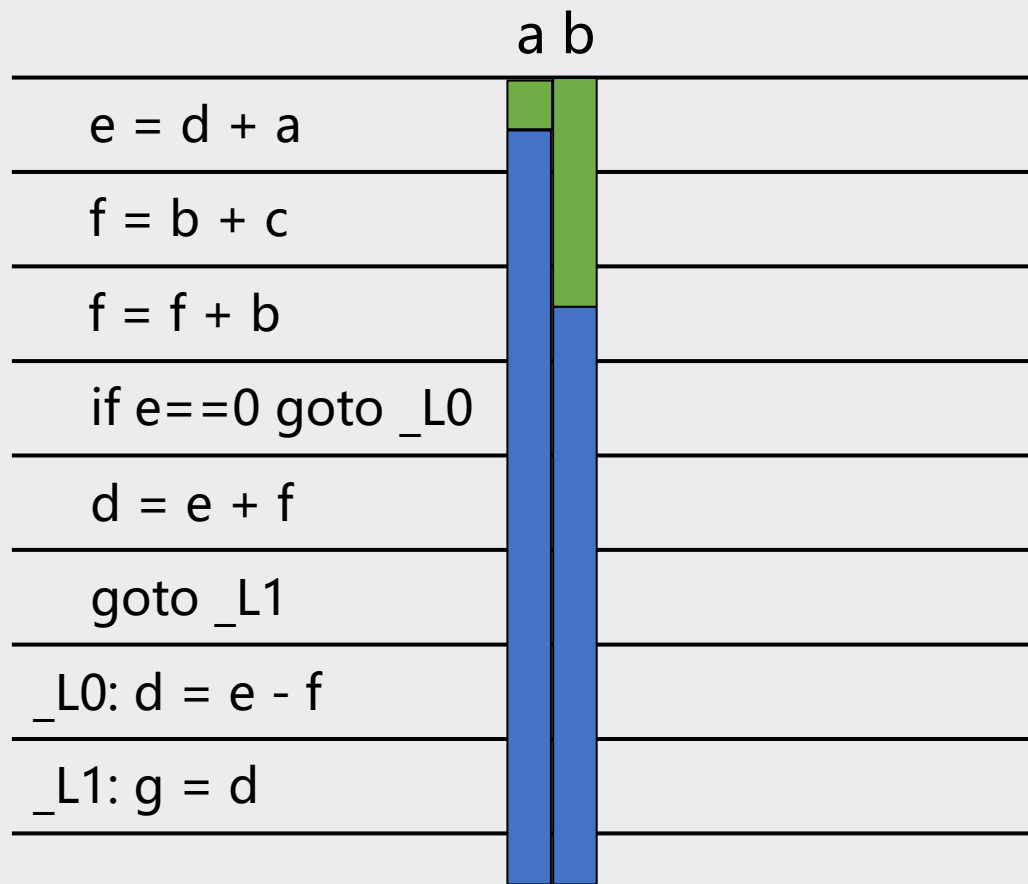


寄存器线性扫描分配——变量存活区间



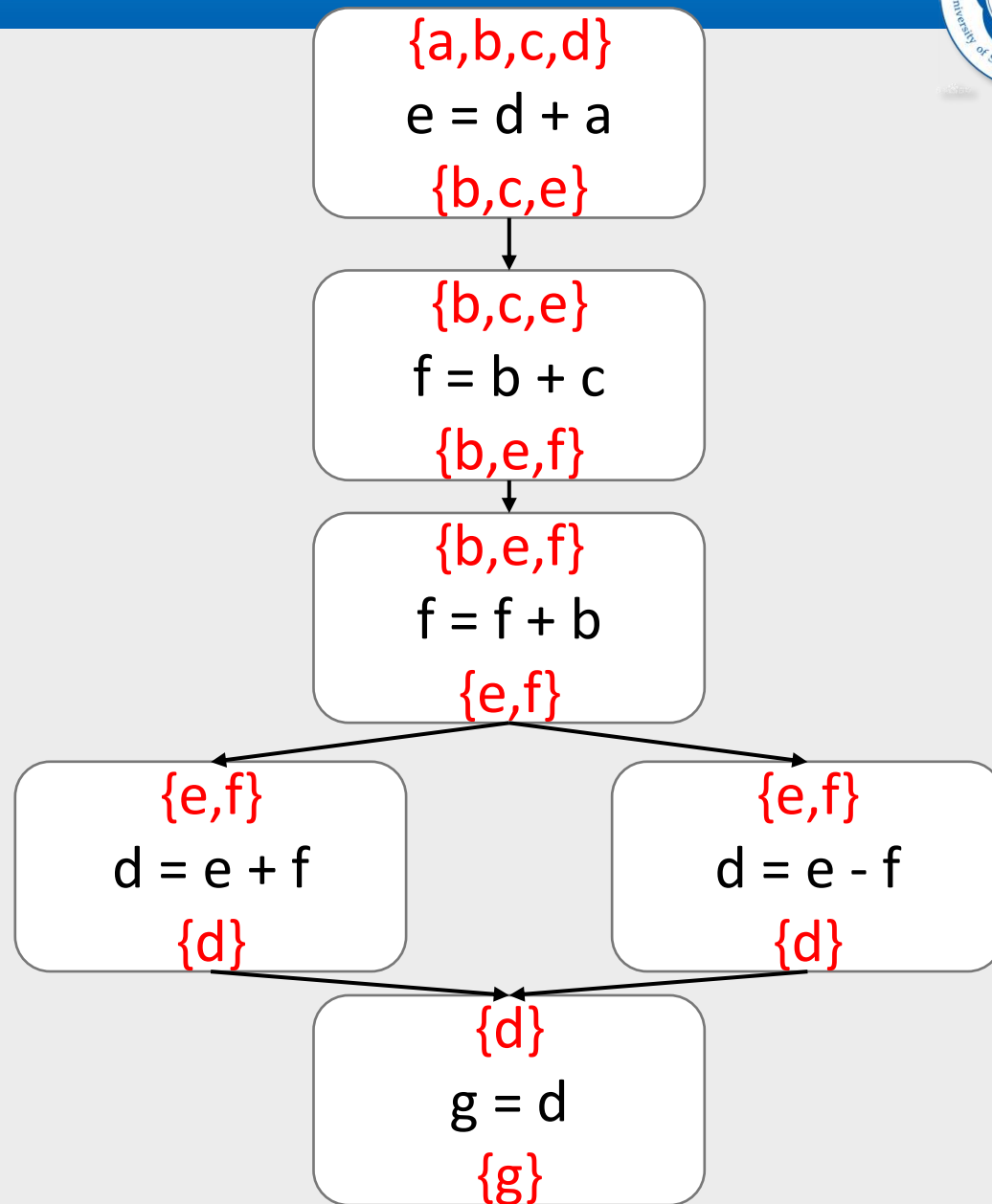
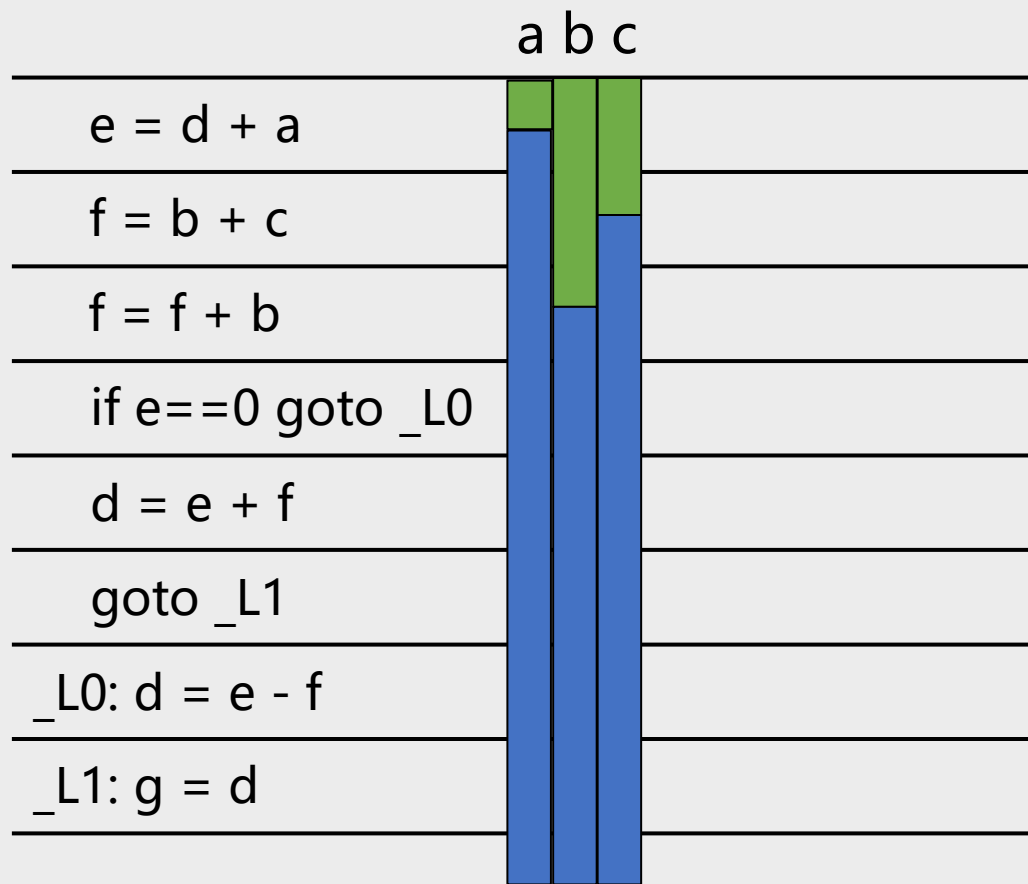


寄存器线性扫描分配——变量存活区间



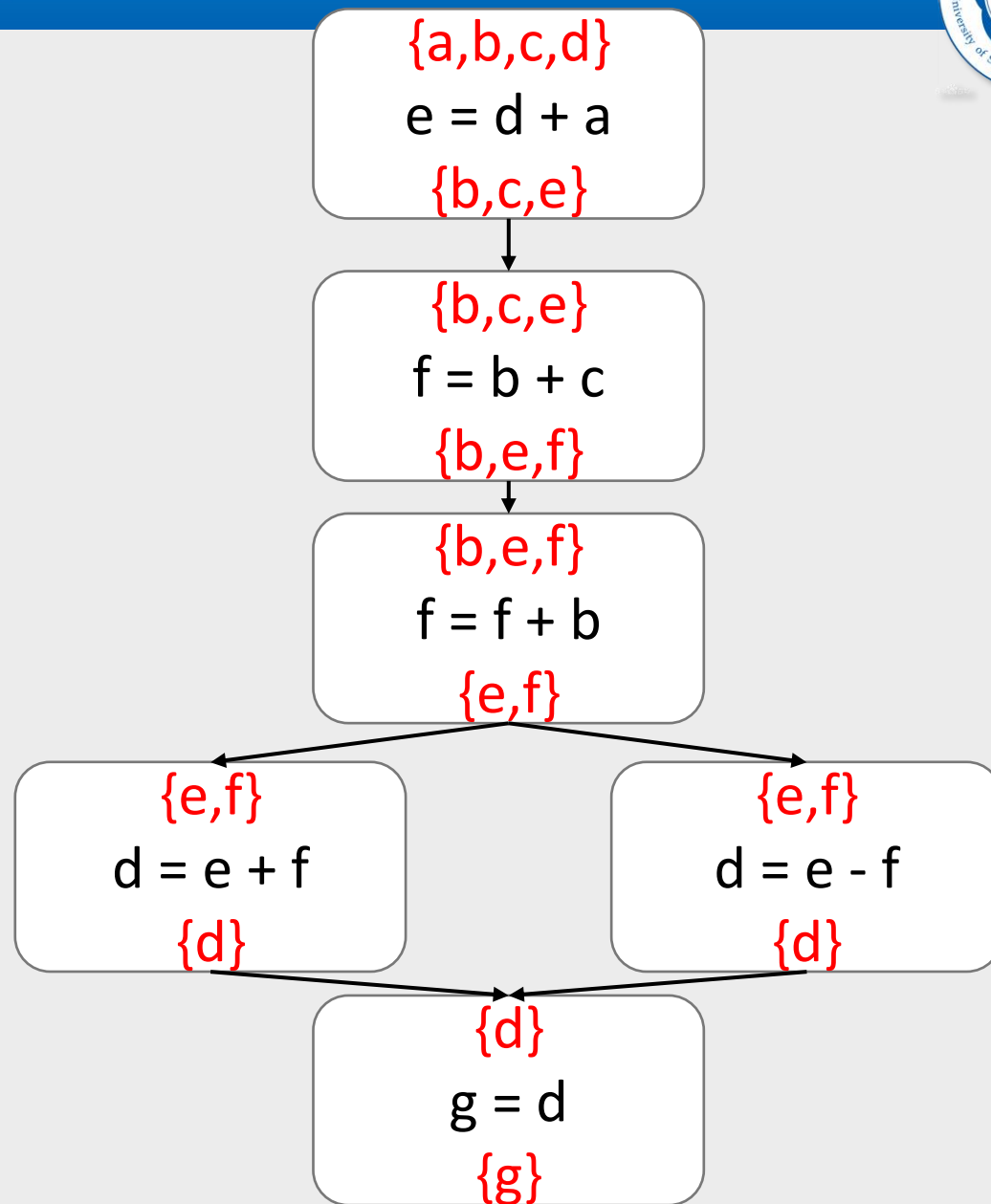
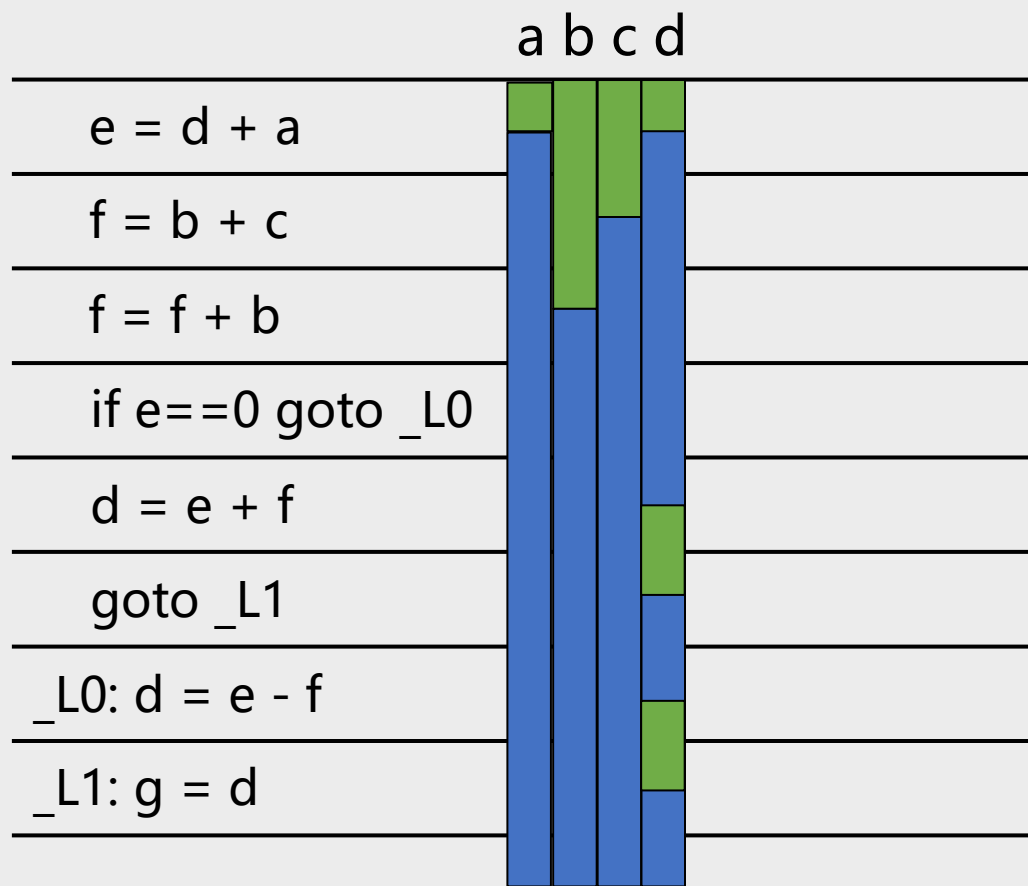


寄存器线性扫描分配——变量存活区间



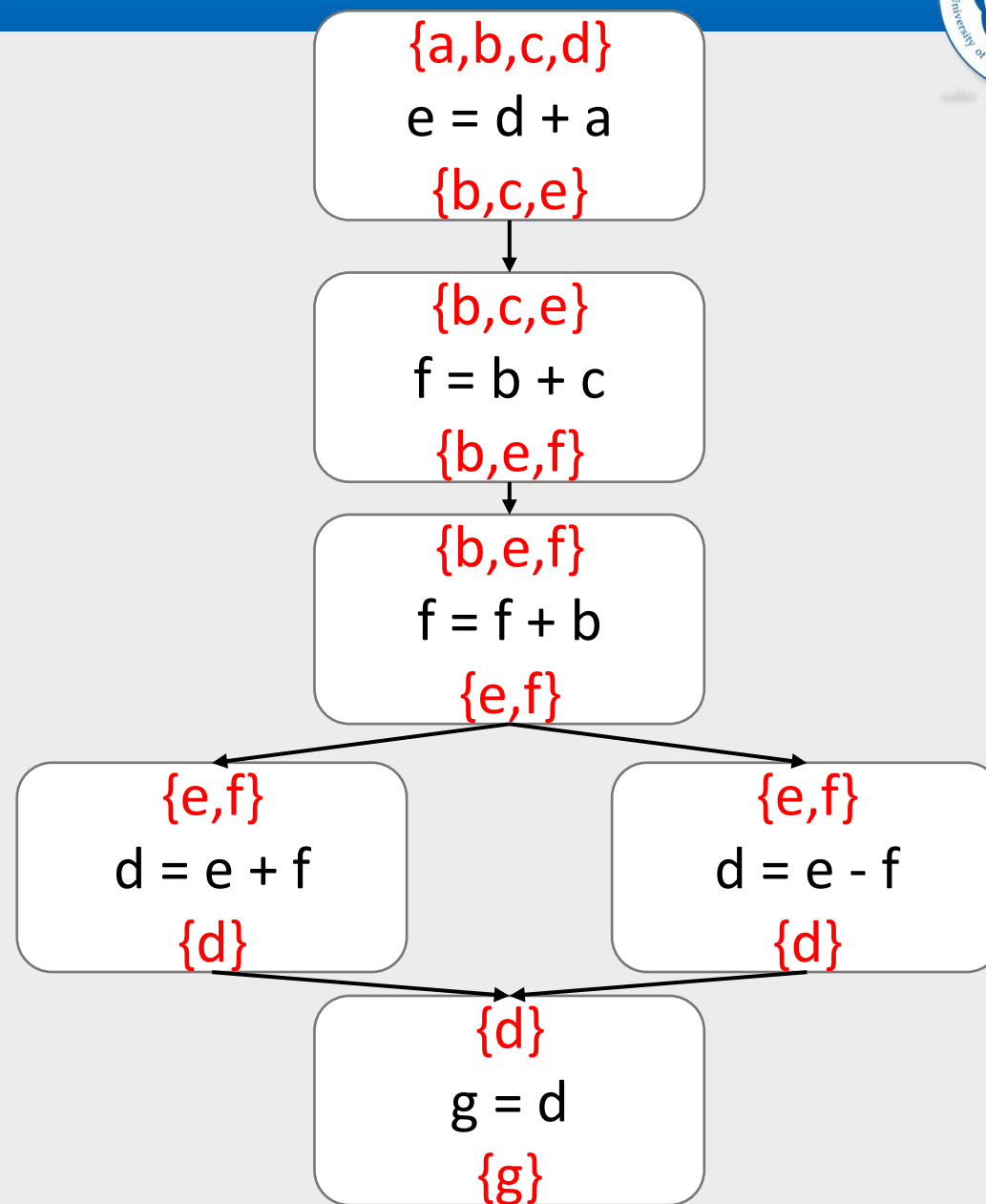
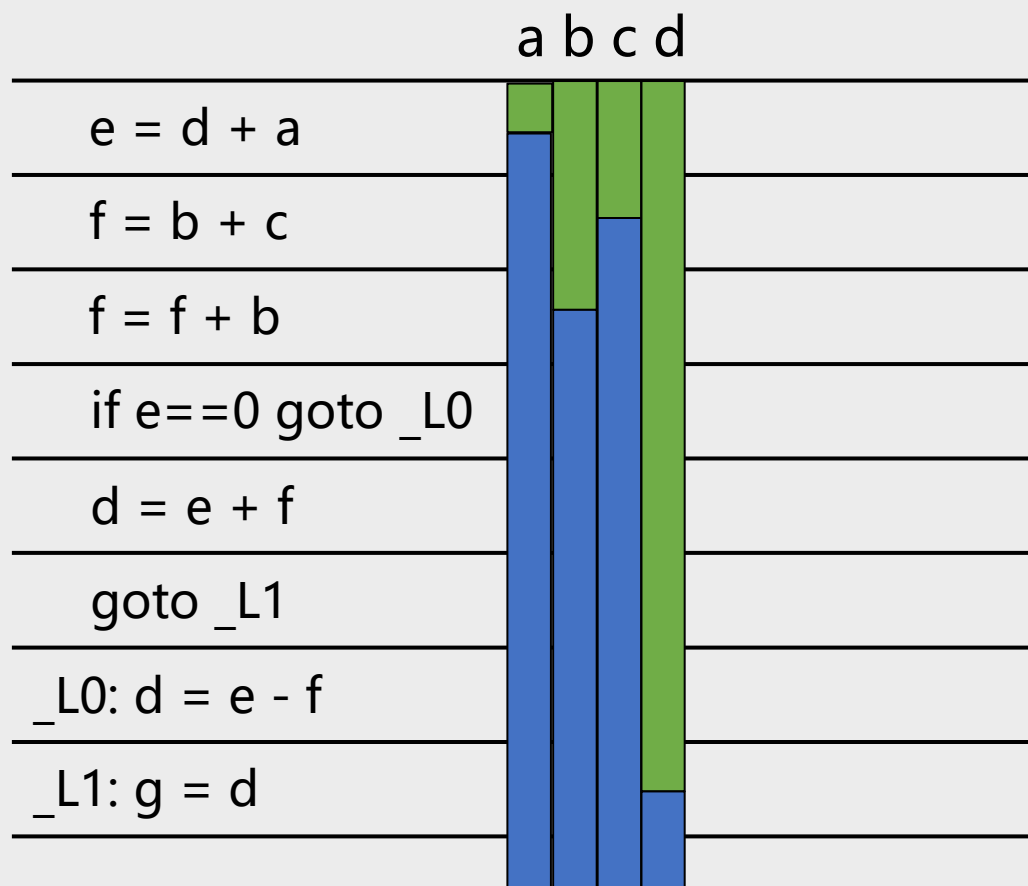


寄存器线性扫描分配——变量存活区间



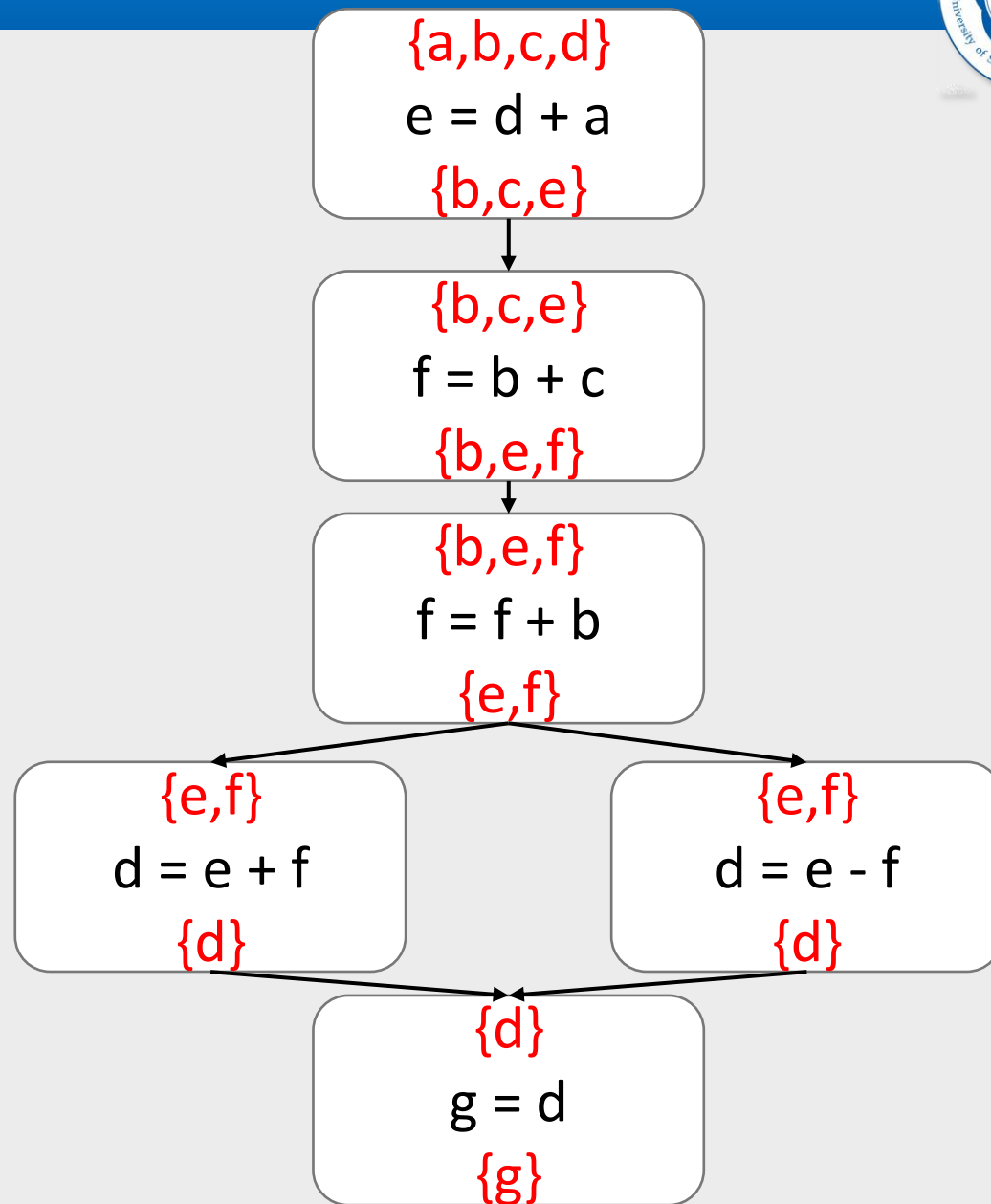
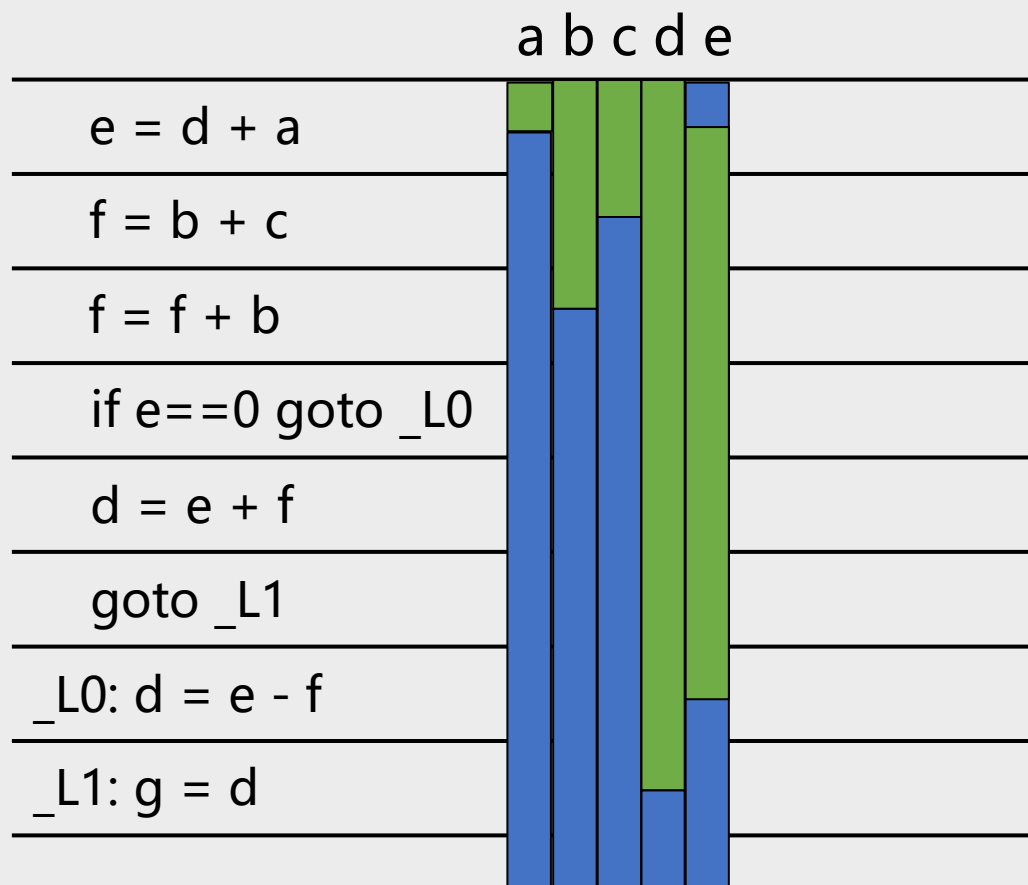


寄存器线性扫描分配——变量存活区间



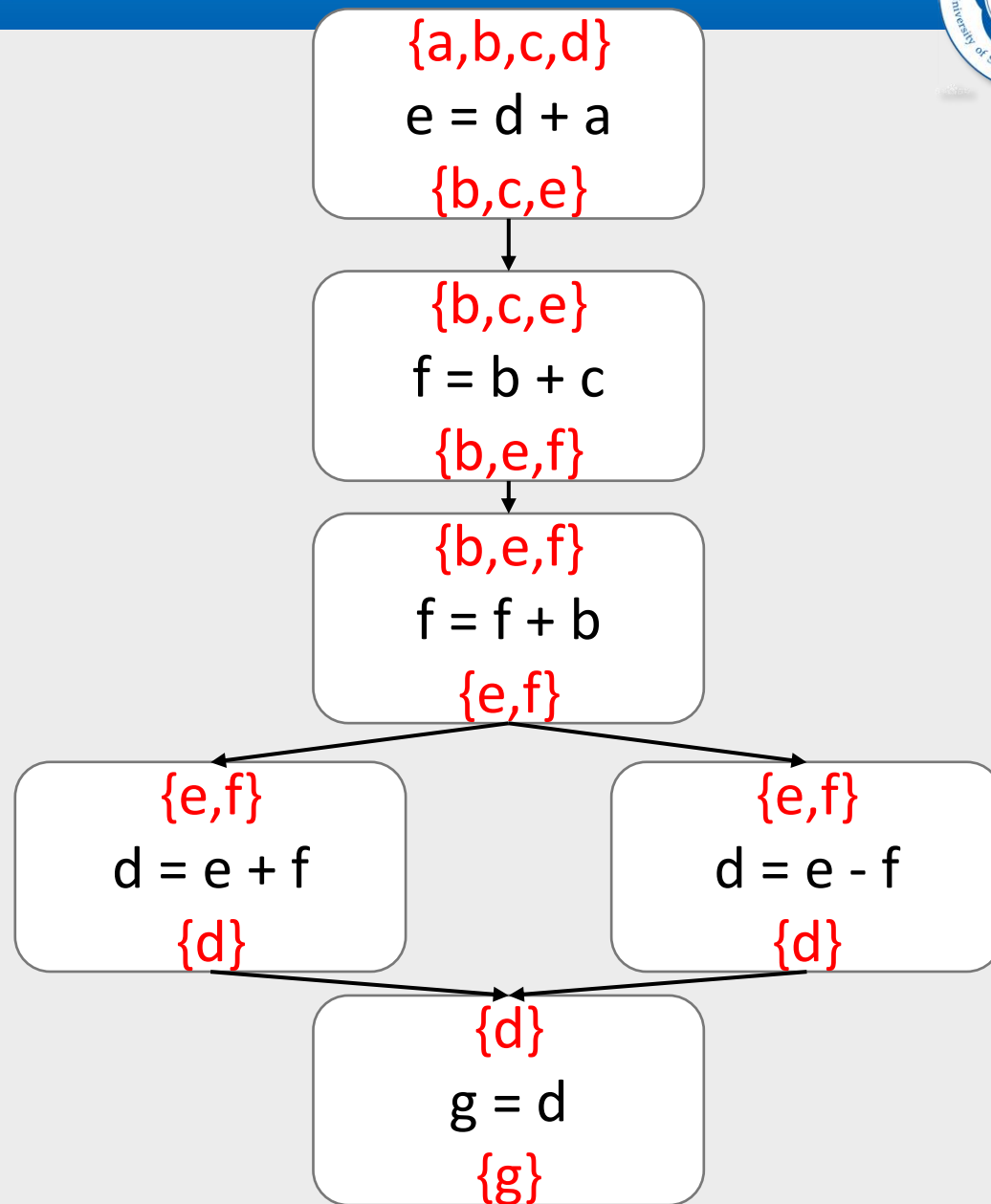
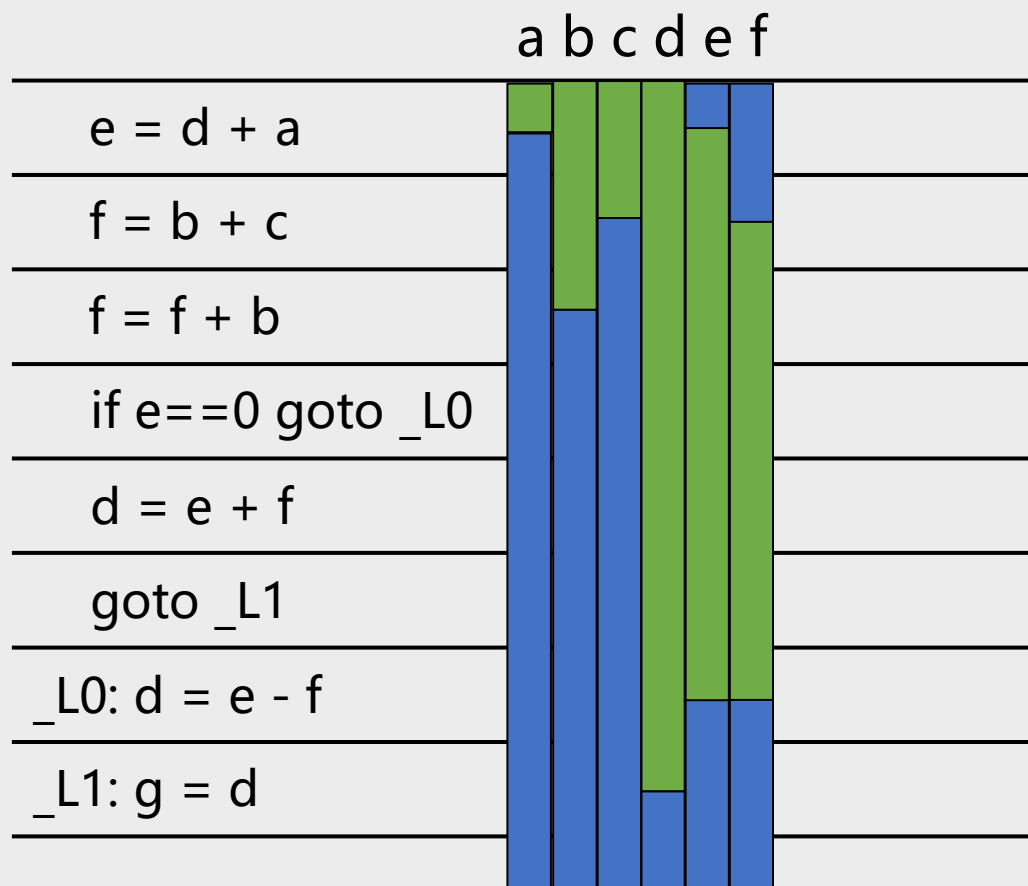


寄存器线性扫描分配——变量存活区间



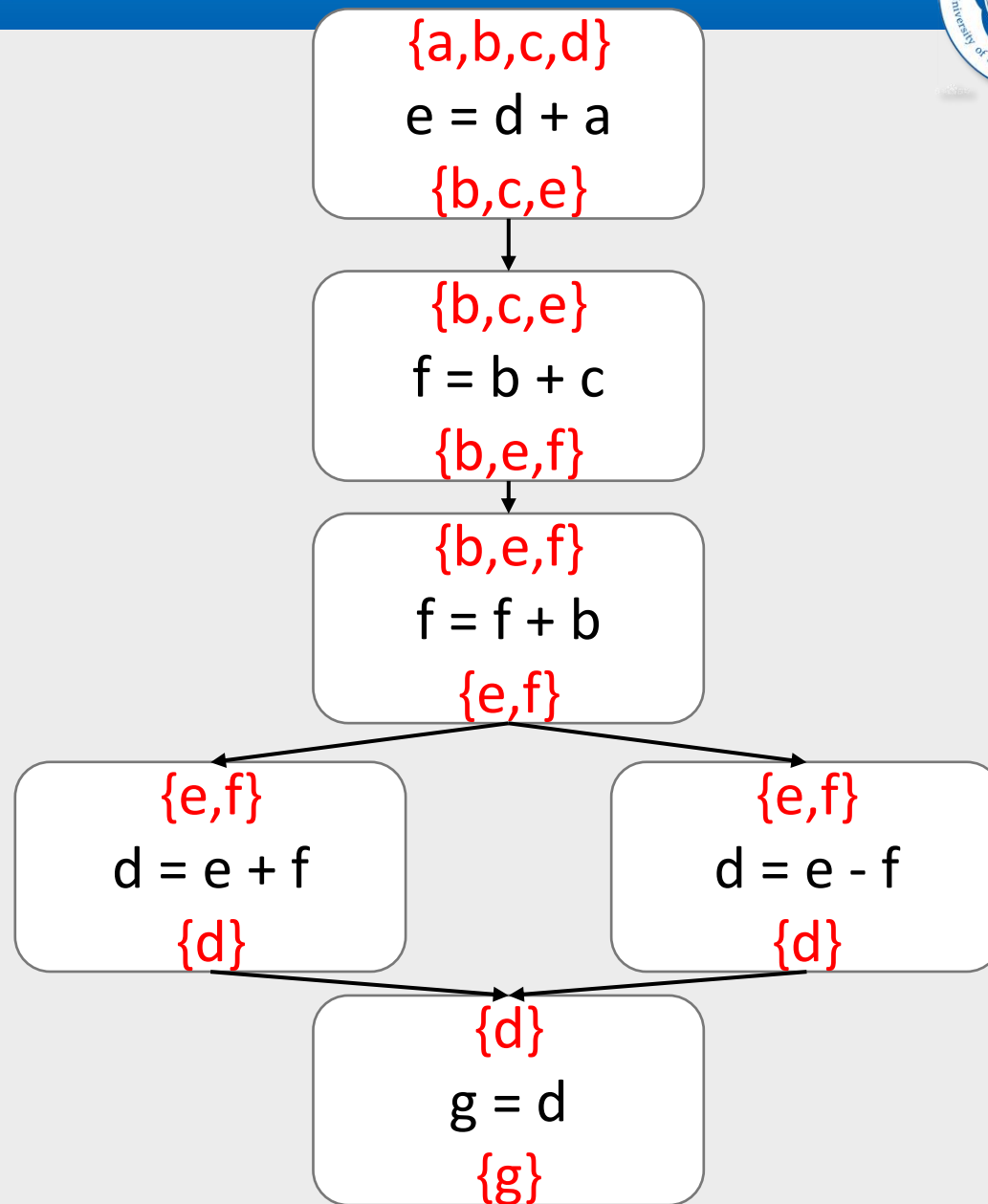
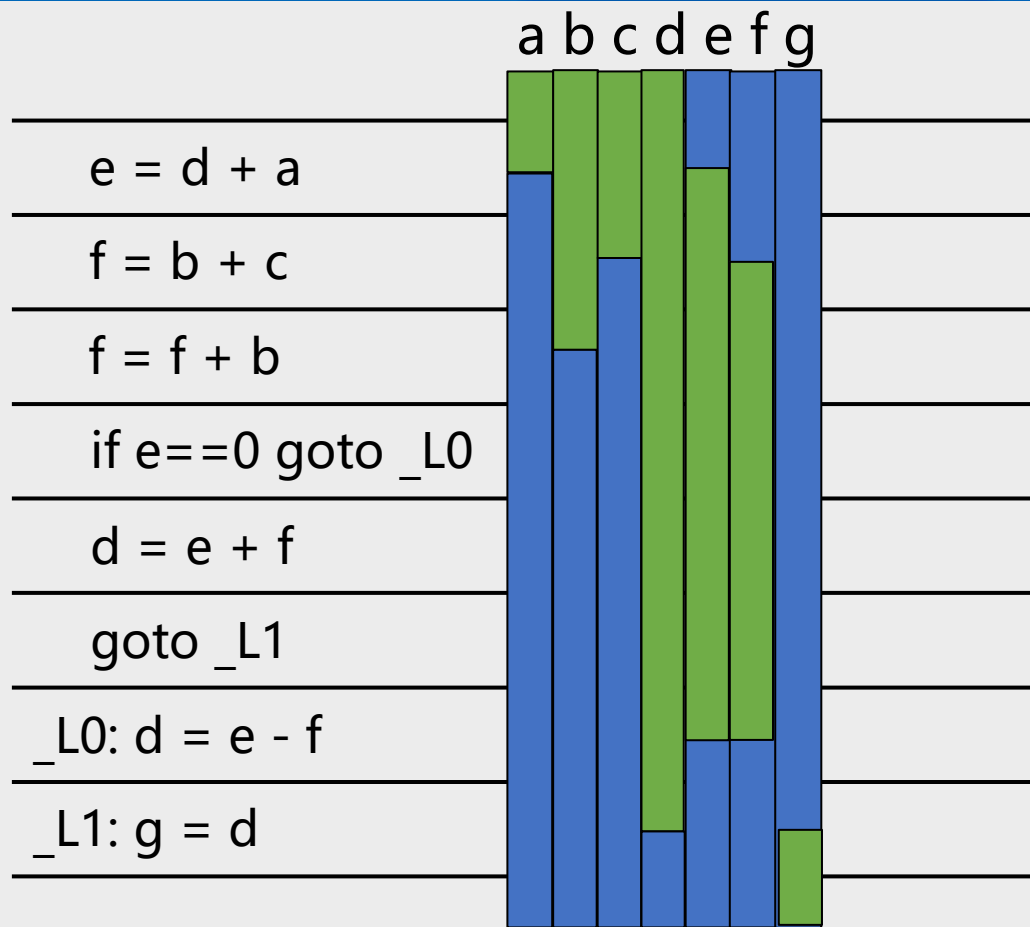


寄存器线性扫描分配——变量存活区间



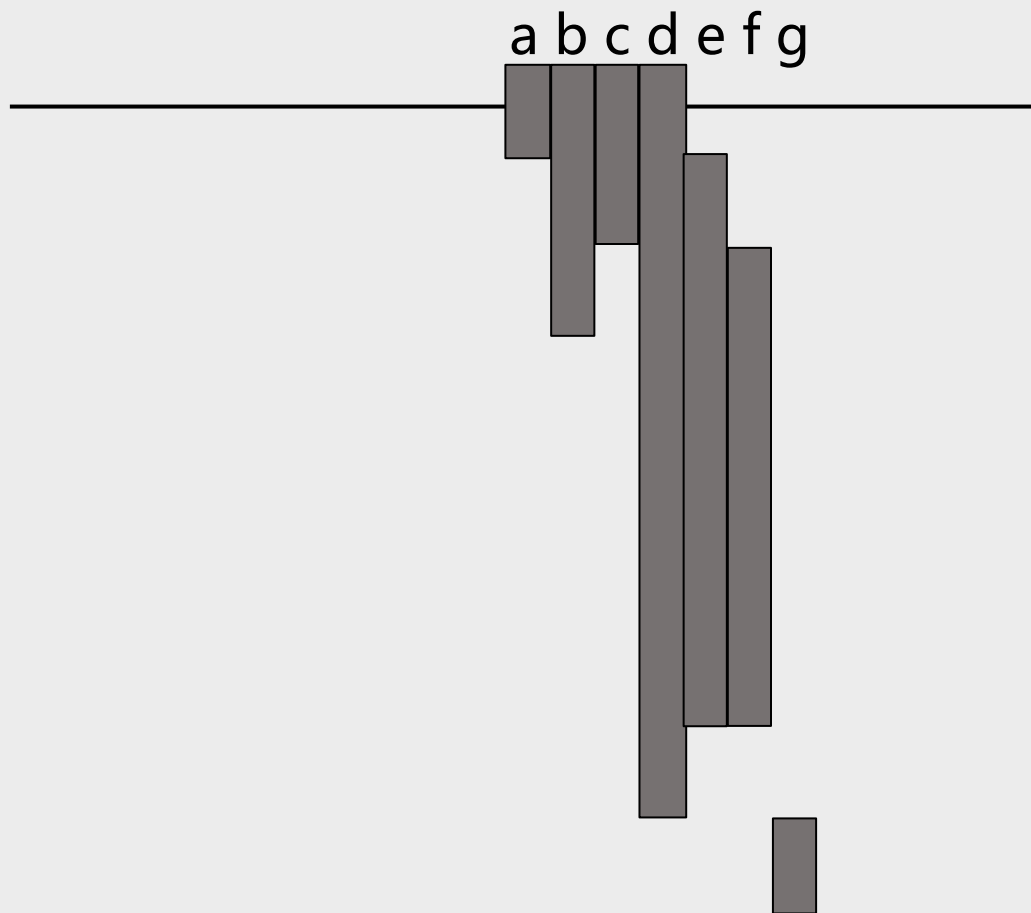


寄存器线性扫描分配——变量存活区间



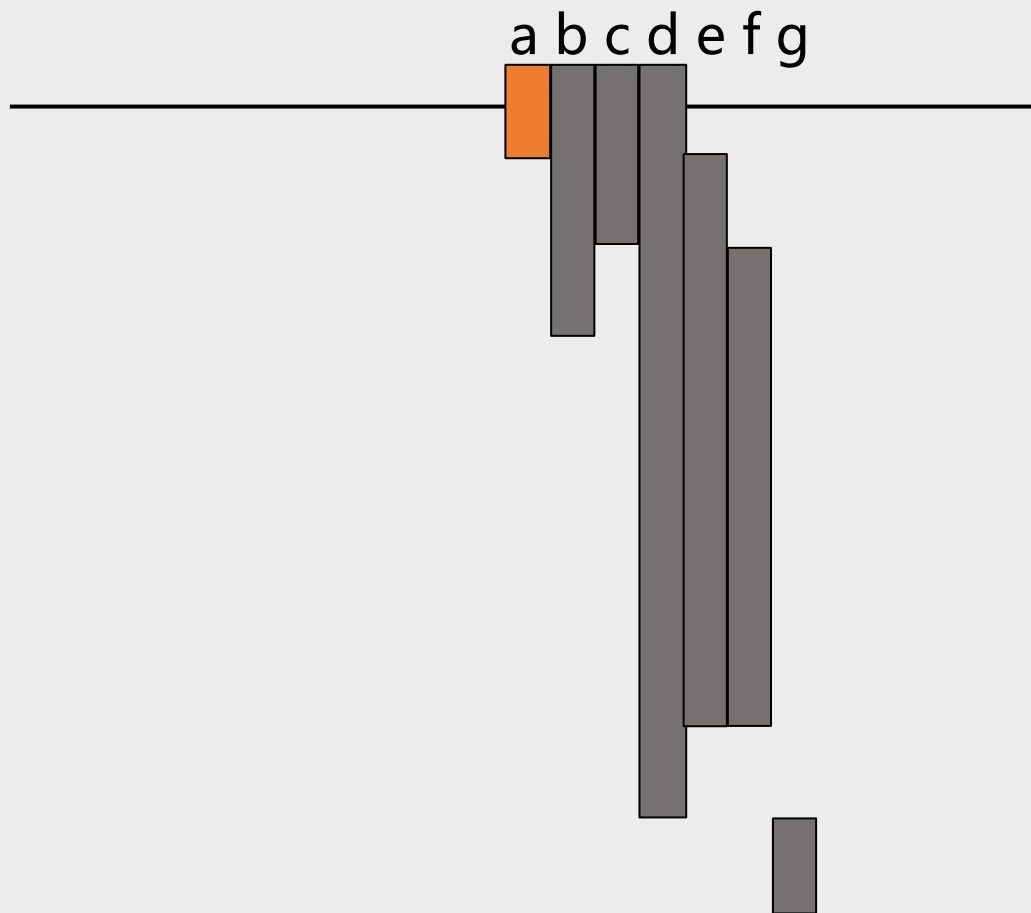


寄存器线性扫描——贪心分配策略

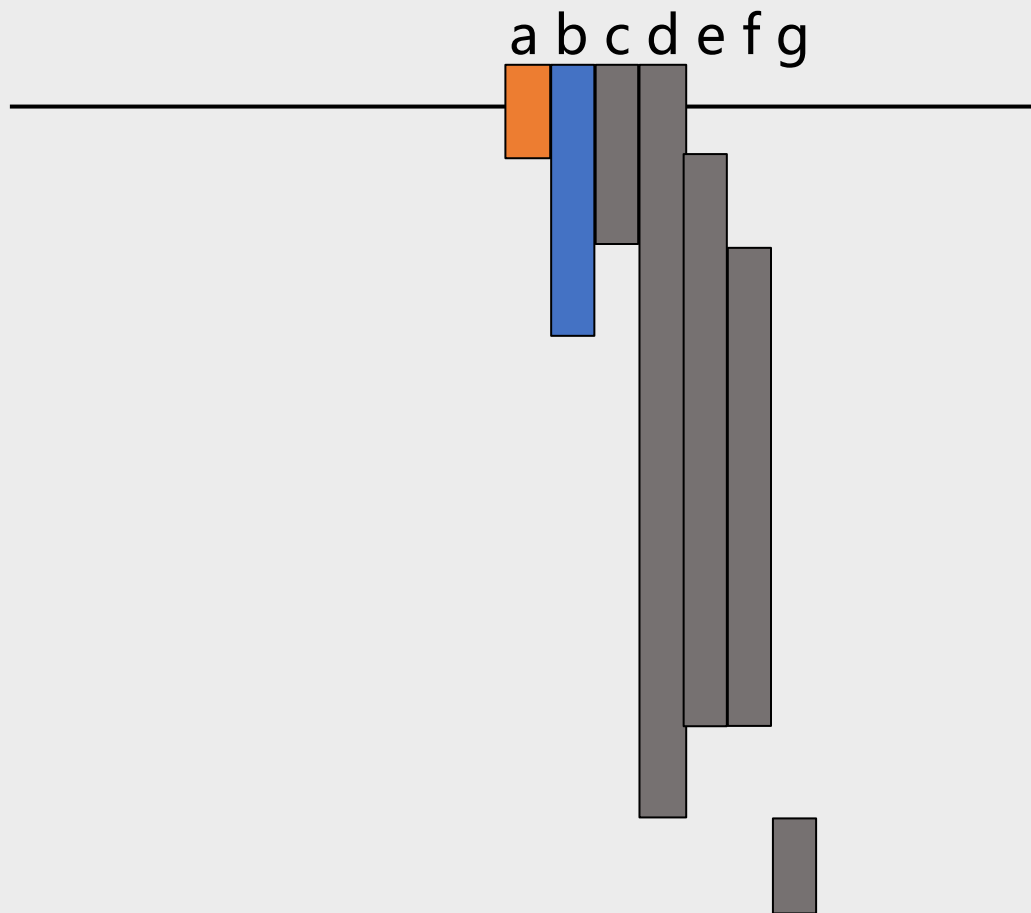




寄存器线性扫描——贪心分配策略

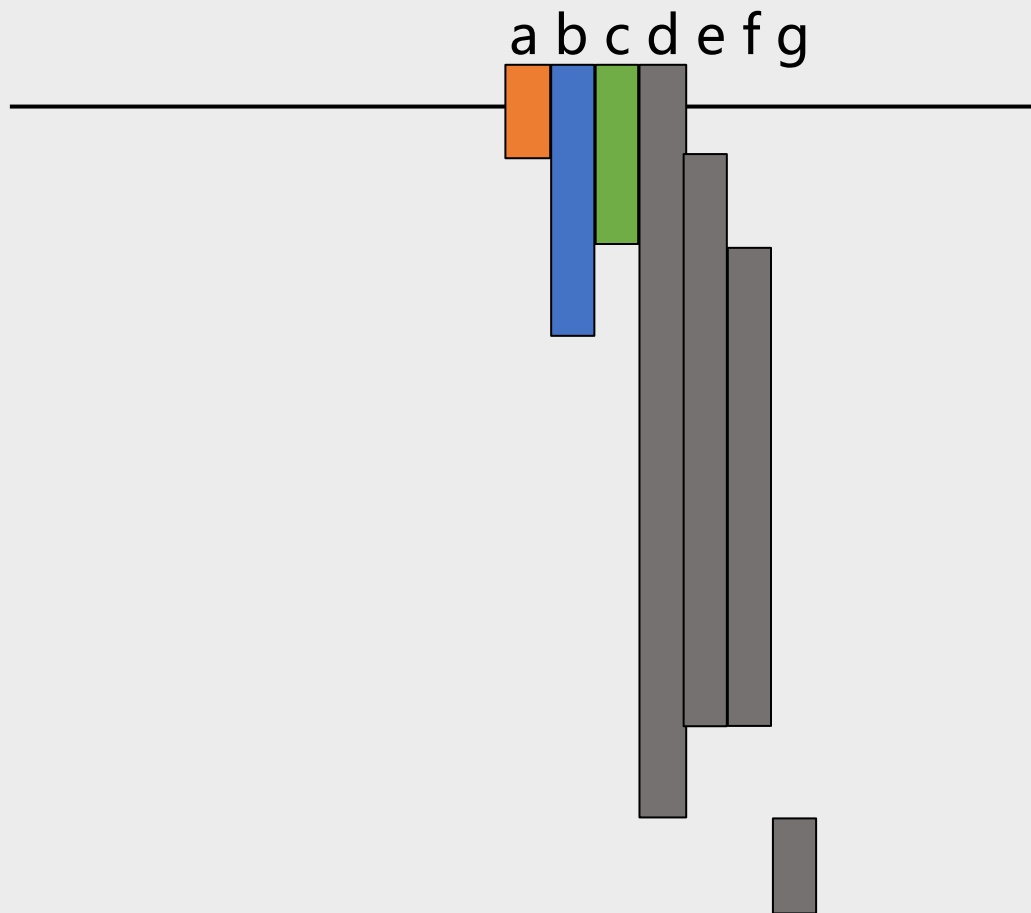


寄存器线性扫描——贪心分配策略

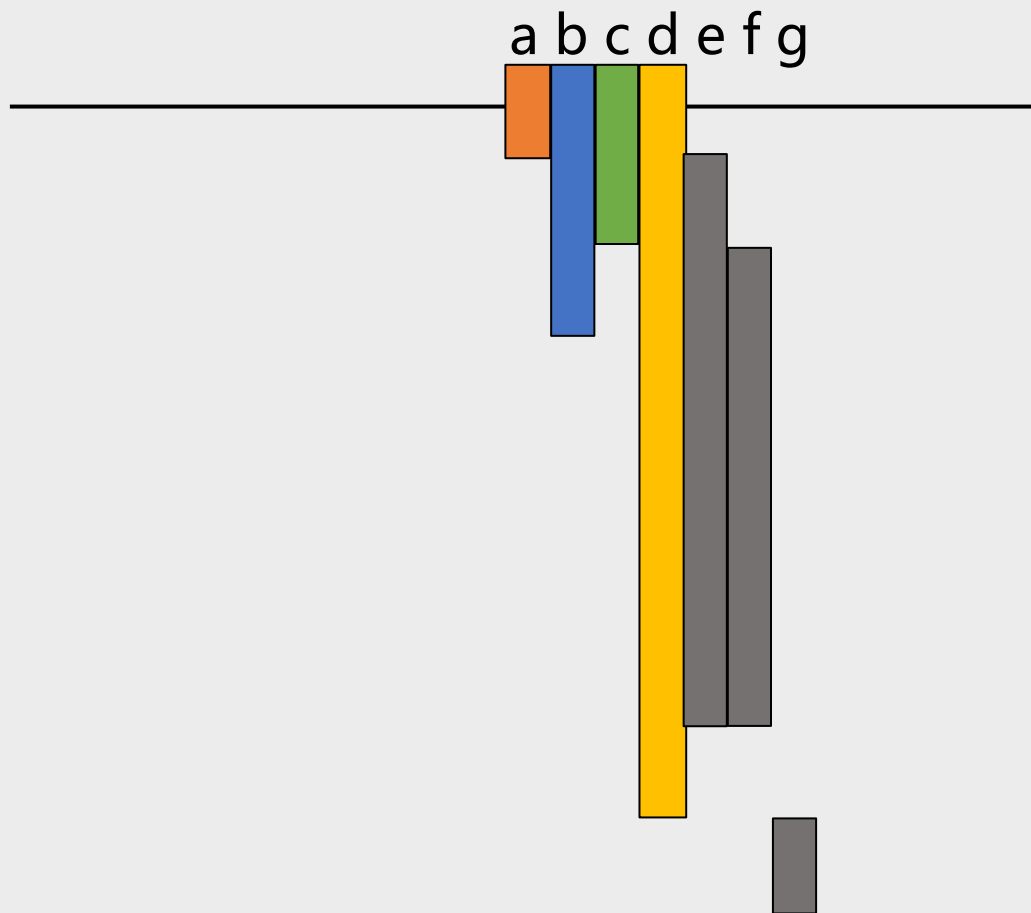




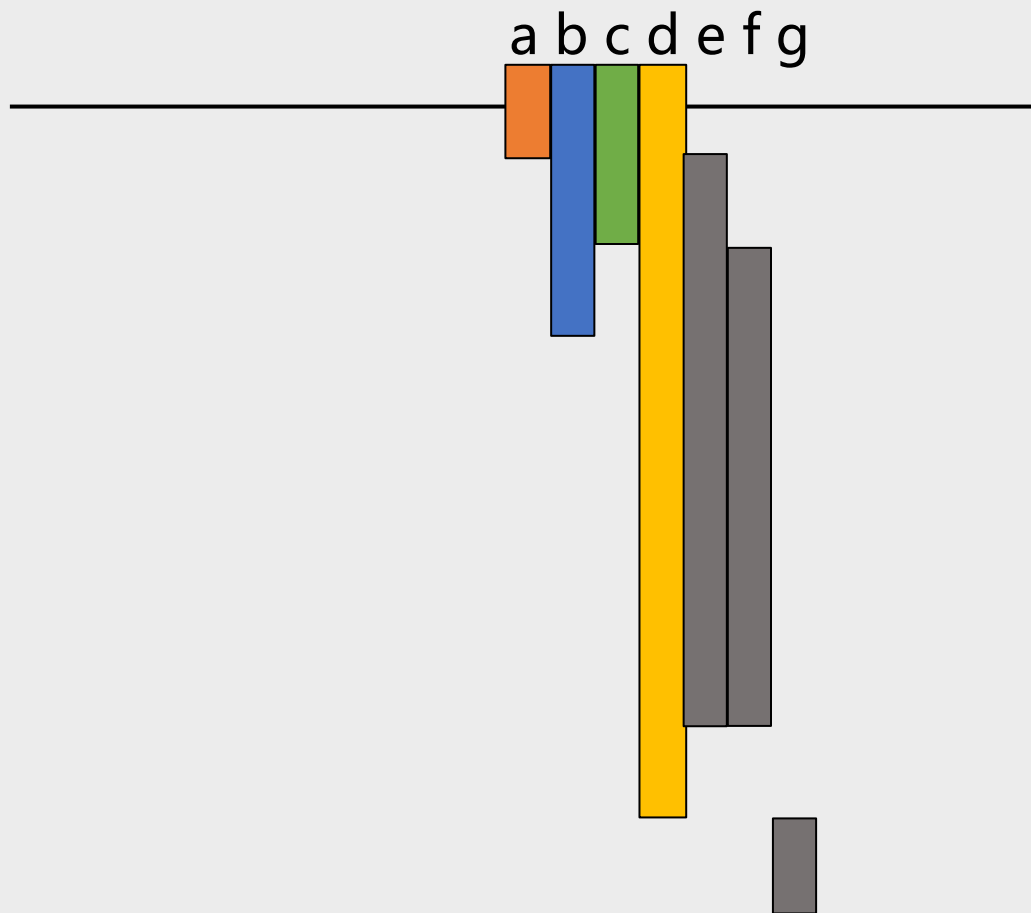
寄存器线性扫描——贪心分配策略



寄存器线性扫描——贪心分配策略

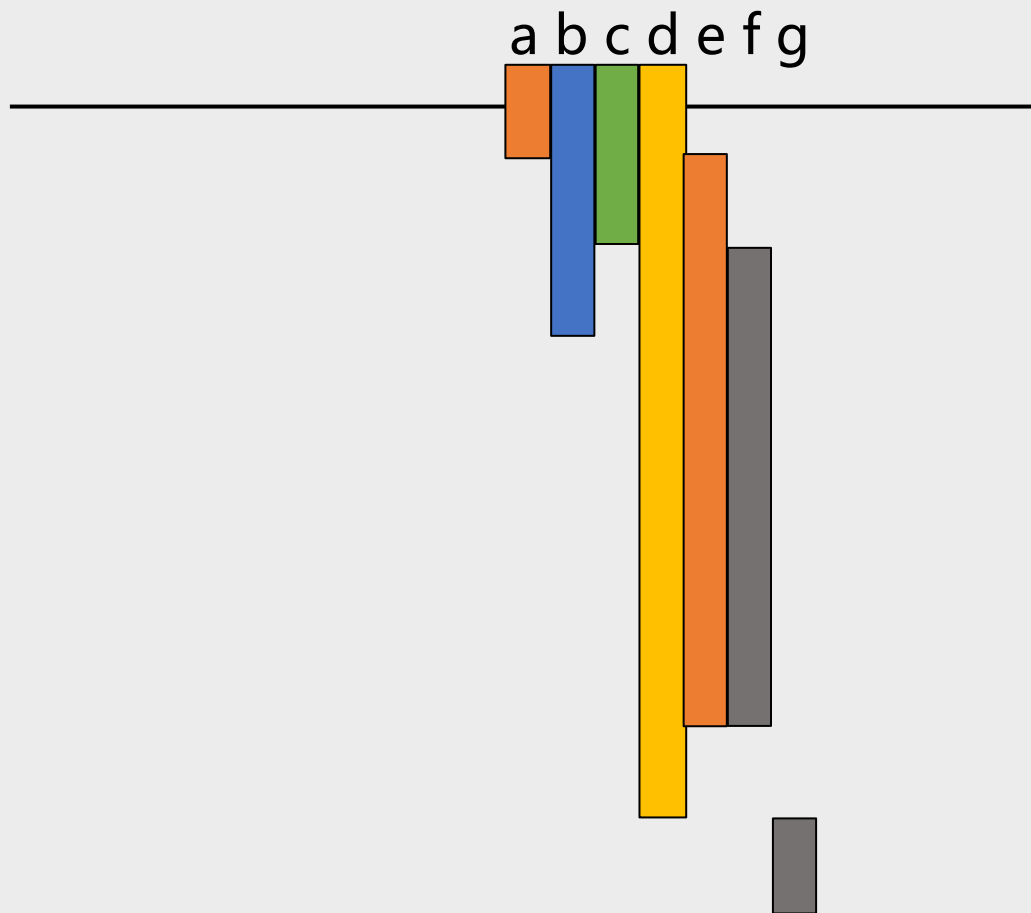


寄存器线性扫描——贪心分配策略



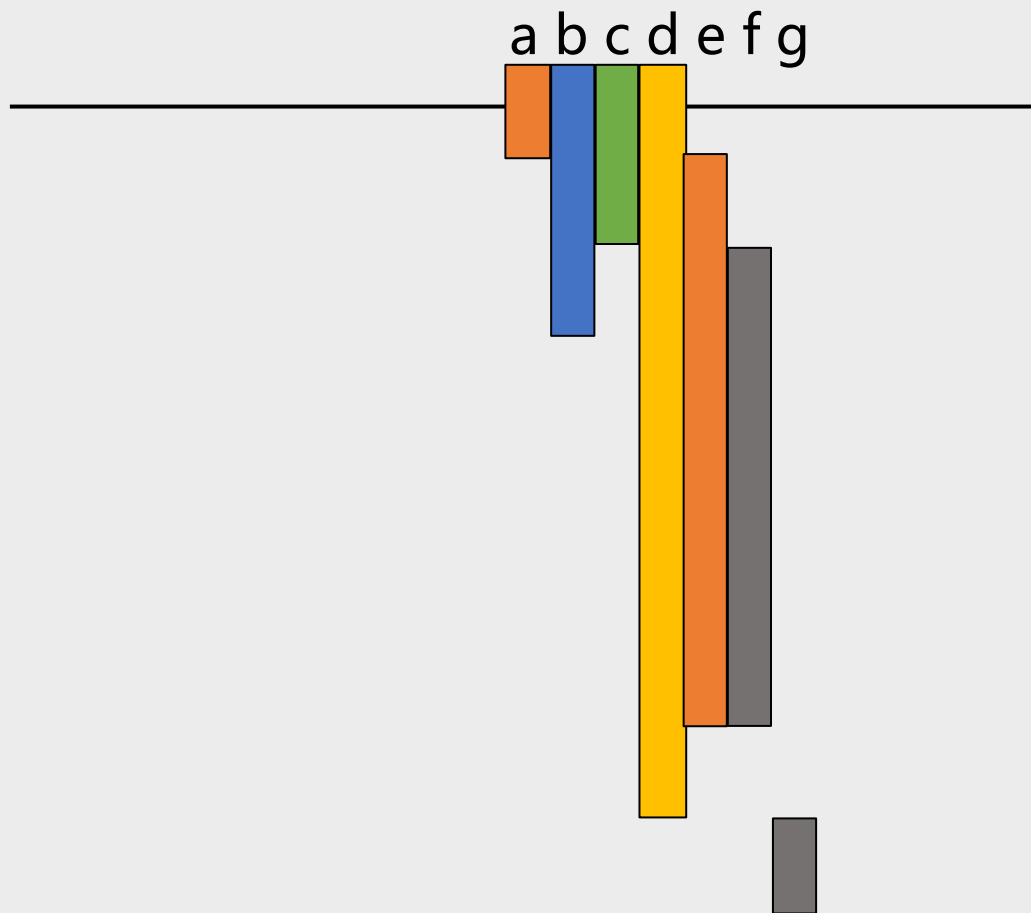


寄存器线性扫描——贪心分配策略



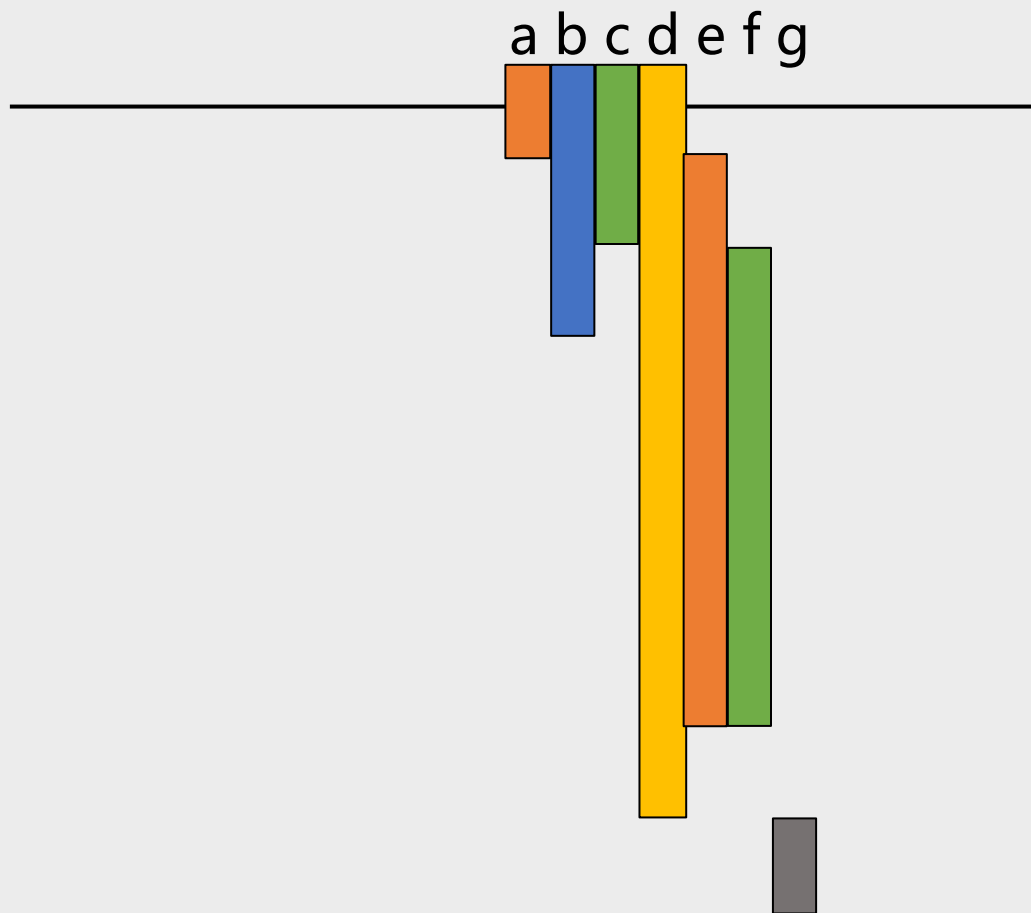


寄存器线性扫描——贪心分配策略



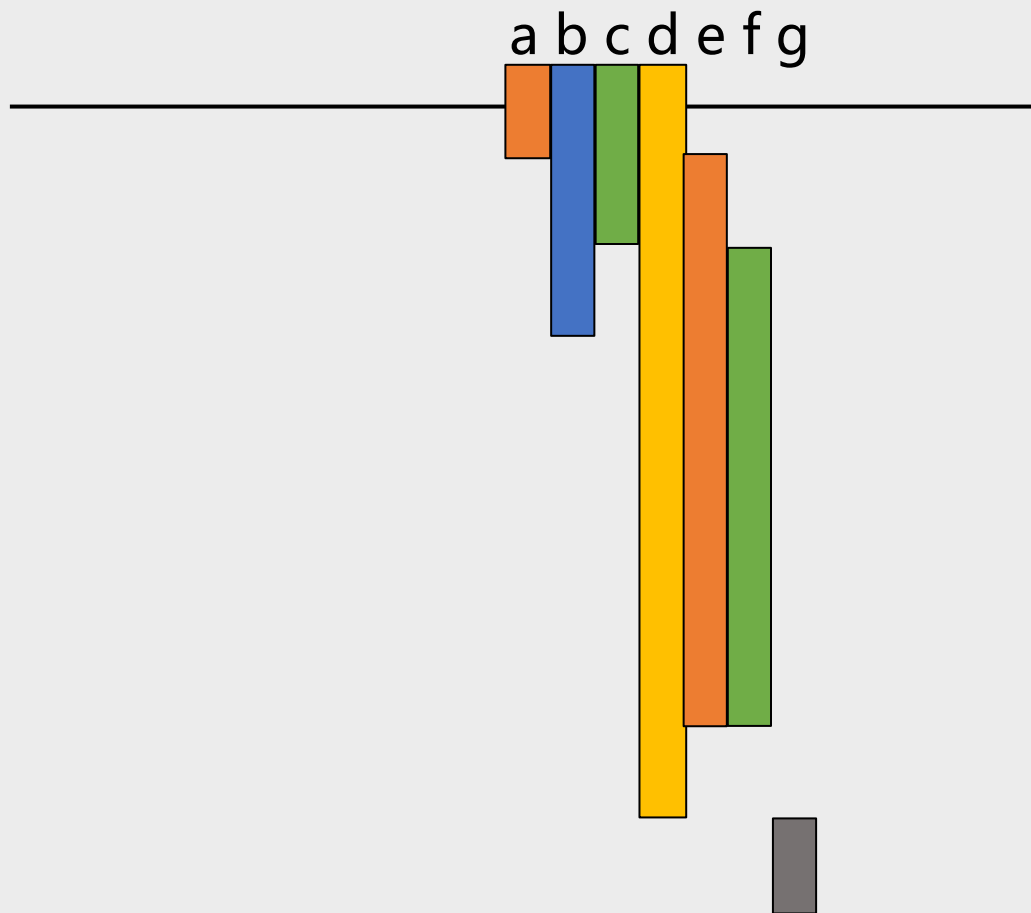


寄存器线性扫描——贪心分配策略



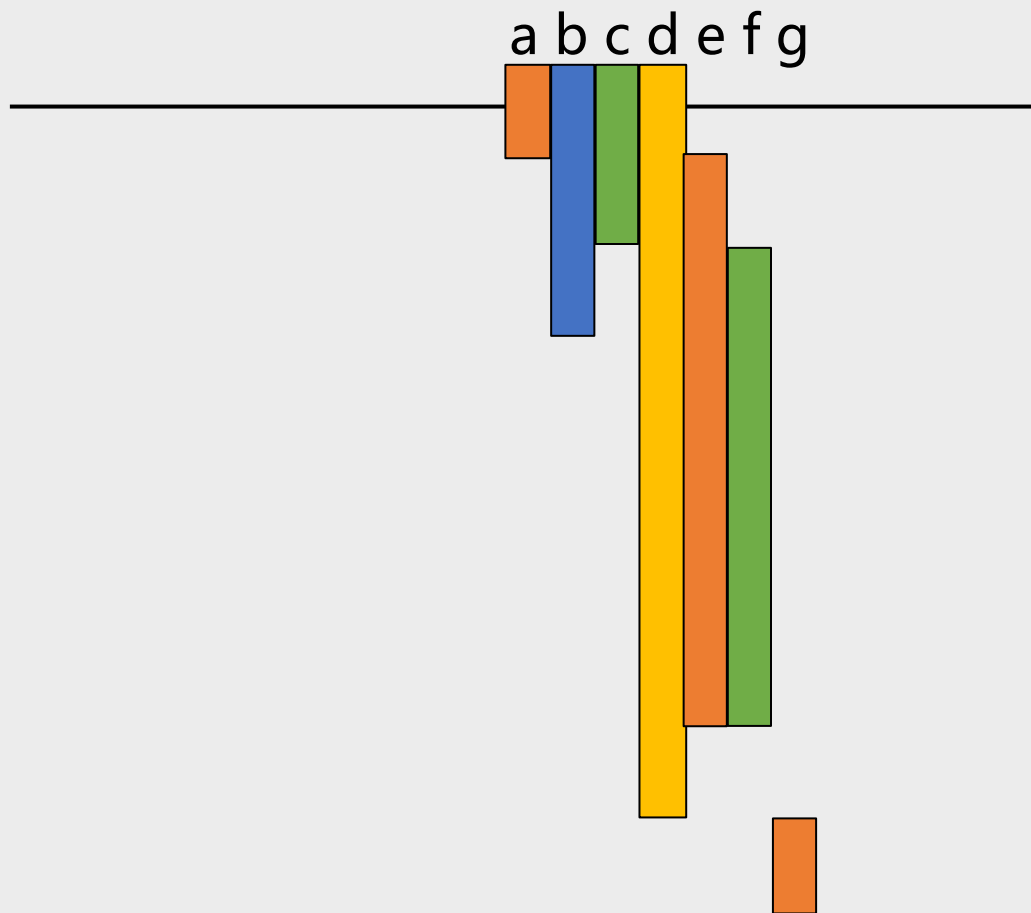


寄存器线性扫描——贪心分配策略





寄存器线性扫描——贪心分配策略





本节小结



- **寄存器是宝贵的计算机资源，需要合理利用和分配**
- **寄存器分配主要有线性扫描和图着色两类算法**
 - 前者比后者性能更好，应用更加广泛
- **线性扫描需要借助于变量存活区间的分析**
 - 需要数据流分析的抽象



- **延伸阅读:**

- 线性扫描算法:

- Linear Scan Register Allocation for the Java HotSpot™ Client Compiler

- 图着色算法:

- Register allocation & spilling via graph coloring



一起努力 打造国产基础系统软件体系！

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

2025年11月17日