9.1对于图9.32流图：

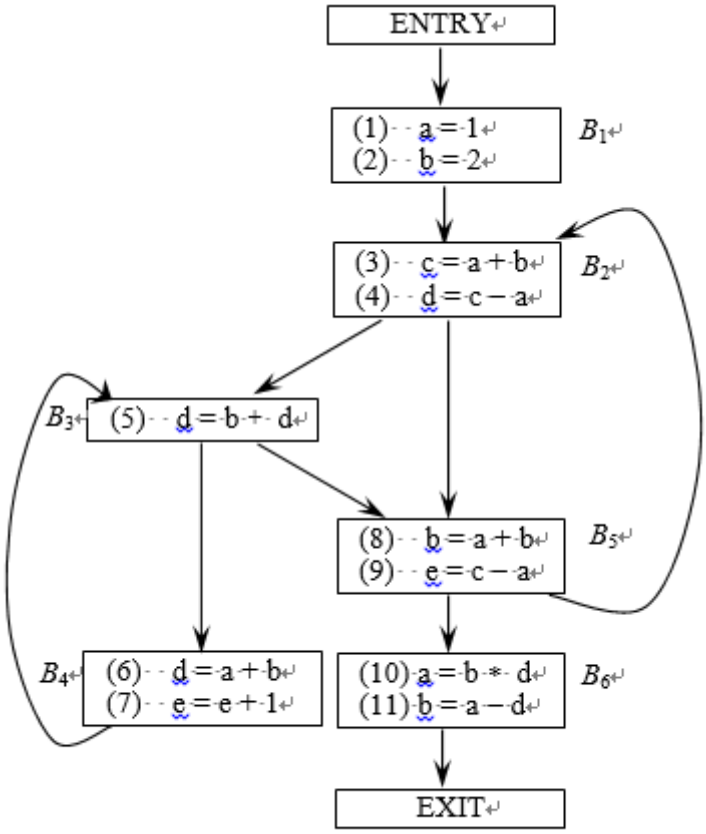

图 9.32 一个流图

(a) 识别该流图的循环。

- $\{B_3, B_4\}$
- $\{B_2, B_3, B_4, B_5\}$

(b) 块B1中的语句(1)和(2)都是复写语句，并且它们给a和b赋的都是常量。可以对a和b的哪些引用实施复写传播并将这些引用替换成对常量的引用?

- 对a复写传播并替换：
  - B_2中$c = a + b$
  - B_2中$d = c - a$
  - B_4中$d = a + b$
  - B_5中$b = a + b$
  - B_5中$e = c - a$
- 对b没有

(c) 识别每个循环的全局公共子表达式。

- 循环$\{B_2, B_3, B_4, B_5\}$的公共子表达式
  - $a + b$, $c - a$

    包括
    - B_2中$c = a + b$
    - B_2中$d = c - a$
    - B_4中$d = a + b$
    - B_5中$b = a + b$

    - B_5中$e = c - a$
- 循环$\{B_3, B_4\}$的公共子表达式

  无

(d) 识别每个循环的归纳变量，不要忘记把(b)的复写传播引入的常量考虑进去。

- 循环$\{B_2, B_3, B_4, B_5\}$的归纳变量为b，c，`b=1+b， c=1+b`
- 循环$\{B_3, B_4\}$的归纳变量为e， `e=e+1`

(e) 识别每个循环的循环不变计算。

- 循环$\{B_3, B_4\}$的循环不变量有B_4中的$d = a + b$
- 循环$\{B_2, B_3, B_4, B_5\}$无

9.3 对图9.32的流图，计算：

（b）为可用表达式分析，计算每个块的e_gen、e_kill、IN和OUT集合。

BB	e_gen	e_kill
$B1$	$1, 2$	$a+b, c-a, b+d, b*d, a-d$
$B2$	$a+b, c-a$	$c-a, b+d, b*d, a-d$
$B3$	NULL	$b+d, b*d, a-d$
$B4$	$a+b$	$b+d, b*d, a-d, e+1$
$B5$	$c-a$	$a+b, b+d, b*d, e+1$
$B6$	$a-d$	$a+b, c-a, a-d, b+d, b*d$

BB	OUT[B]_0	IN[B]_1	OUT[B]_1
$B1$	$1, 2, a+b, c-a, a-d, b+d, b*d, e+1$	NULL	$1, 2$
$B2$	$1, 2, a+b, c-a, a-d, b+d, b*d, e+1$	$1, 2$	$1, 2, a+b, c-a$
$B3$	$1, 2, a+b, c-a, a-d, b+d, b*d, e+1$	$1, 2, a+b, c-a$	$1, 2, a+b, c-a$
$B4$	$1, 2, a+b, c-a, a-d, b+d, b*d, e+1$	$1, 2, a+b, c-a$	$1, 2, a+b, c-a$
$B5$	$1, 2, a+b, c-a, a-d, b+d, b*d, e+1$	$1, 2, a+b, c-a$	$1, 2, c-a$
$B6$	$1, 2, a+b, c-a, a-d, b+d, b*d, e+1$	$1, 2, c-a$	$1, 2, a-d$

BB	IN[B]_2	OUT[B]_2
$B1$	NULL	$1,2$
$B2$	$1,2$	$a+b,c-a,1,2$
$B3$	$a+b,c-a,1,2$	$a+b,c-a,1,2$
$B4$	$a+b,c-a,1,2$	$a+b,c-a,1,2$
$B5$	$a+b,c-a,1,2$	$c-a,1,2$
$B6$	$c-a,1,2$	$a-d,1,2$

第一次和第二次扫描后结果不变，结束扫描。

（c）为活跃变量分析，计算每个块的def、use、IN和OUT集合。

BB	USE[B]	def[B]
$B1$	NULL	a,b
$B2$	a,b	c,d
$B3$	b,d	NULL
$B4$	a,b,e	d
$B5$	a,b,c	e
$B6$	b,d	a

BB	IN[B]_0	OUT[B]_1	IN[B]_1
$B6$	NULL	NULL	b,d
$B5$	NULL	b,d	a,b,c,d
$B4$	NULL	NULL	a,b,e
$B3$	NULL	a,b,c,d,e	a,b,c,d,e
$B2$	NULL	a,b,c,d,e	a,b,e
$B1$	NULL	a,b,e	e

BB	OUT[B]_2	IN[B]_2
$B6$	NULL	b,d
$B5$	a,b,d,e	a,b,c,d
$B4$	a,b,c,d,e	a,b,c,e
$B3$	a,b,c,d,e	a,b,c,d,e
$B2$	a,b,c,d,e	a,b,e
$B1$	a,b,e	e

BB	OUT[B]_3	IN[B]_3
$B6$	NULL	b, d
$B5$	a, b, d, e	a, b, c, d
$B4$	a, b, c, d, e	a, b, c, e
$B3$	a, b, c, d, e	a, b, c, d, e
$B2$	a, b, c, d, e	a, b, e
$B1$	a, b, e	e

第二次和第三次扫描后结果不变，结束扫描。

9.31 下面的C程序分别经非优化编译和2级以上（含2级）的优化编译后，生成的两个目标程序运行时的表现不同（编译器是GCC: (GNU) 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)）。

请回答它们运行时的表现有何不同，并说明原因。

```
int f(int g( ) ) {
    return g(g);
}
main() {
    f(f);
}
```

- 该程序不会终止，会陷入无穷的递归调用。
- 非优化情况下，无穷的递归调用导致运行栈溢出，引起系统报告segmentation fault.

```
f:
.LFB0:
    .cfi_startproc
    endbr64
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    subq    $16, %rsp
    movq    %rdi, -8(%rbp)
    movq    -8(%rbp), %rax
    movq    -8(%rbp), %rdx
    movq    %rax, %rdi
    movl    $0, %eax
    call    *%rdx
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
```

- 高级全局优化情况下，可知函数f的return语句中函数调用是尾递归调用。函数f被优化为把当前活动记录作为原本要新增的活动记录，并把调用指令改为跳转指令，导致程序陷入死循环但不会出现运行栈的溢出。

```
f:
.LFB0:
    .cfi_startproc
    endbr64
    xorl    %eax, %eax
    jmp *%rdi   ;跳转到f的入口地址
    .cfi_endproc
```