编译原理 H15-2

PB20111686 黄瑞轩

9.3

（a）下面的所有编号都是图 9.32 中的括号内的数字。

	gen	$kill$	IN	OUT
B_1	1, 2	8, 10, 11	$\varnothing$	1, 2
B_2	3, 4	5, 6	1, 2, 3, 4, 5, 8, 9	1, 2, 3, 4, 8, 9
B_3	5	4, 6	1, 2, 3, 4, 6, 7, 8, 9	1, 2, 3, 5, 7, 8, 9
B_4	6, 7	5, 9	1, 2, 3, 5, 7, 8, 9	1, 2, 3, 6, 7, 8
B_5	8, 9	2, 7	1, 2, 3, 4, 5, 7, 8, 9	1, 3, 4, 5, 8, 9
B_6	10, 11	1, 8	1, 3, 4, 5, 8, 9	3, 4, 5, 9, 10, 11

（b）所有的表达式编号如下：

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	2	$a+b$	$c-a$	$b+d$	$e+1$	$b*d$	$a-d$

	e_gen	e_kill	IN	OUT
B_1	1，2	3，4，5，7，8	$\varnothing$	1，2
B_2	3，4	5，7，8	1，2	1，2，3，4
B_3	$\varnothing$	5，7，8	1，2，3，4	1，2，3，4
B_4	3	5，6，7，8	1，2，3，4	1，2，3，4
B_5	4	3，5，6，7	1，2，3，4	1，2，4
B_6	8	3，4，5，7	1，2，4	1，2，8

（c）

	use	def	IN	OUT
B_1	$\varnothing$	a, b	e	a, b, e
B_2	a, b	c, d	a, b, e	a, b, c, d, e
B_3	b, d	d	a, b, c, d, e	a, b, c, d, e
B_4	a, b, e	d, e	a, b, c, e	a, b, c, d, e
B_5	a, b, c	b, e	a, b, c, d	a, b, d, e
B_6	b, d	a, b	b, d	$\varnothing$

9.31

我使用的编译器是 clang 3.7，分别使用

- `clang 1.c -O0 -o 1`
- `clang 2.c -O2 -o 2`

来编译，运行后发现

- 1 程序数秒后就退出了
- 2 程序陷入死循环，不退出

通过

- `clang 1.c -O0 -S -m32`
- `clang 2.c -O2 -S -m32`

生成汇编代码，结果发现是尾递归优化（clang 还用注释标注出来了）。

这个函数本来就是一个不断递归调用的程序，如果不做尾递归优化，第一个程序就会不断开辟新的栈帧直到 stack overflow，但是第二个程序利用尾递归优化，每次开辟新的函数调用时，都会重复利用前一个函数调用的栈帧，栈并不会一直增加，所以不会 stack overflow，只会陷入死循环。