佳凯，你有时间吗？你有时间的话，请你试着用LLVM编译linux-5.4版本的内核，获取内核的CFG，然后确认以下几个问题：

1、CFG的基本单位是什么？基本块还是语句？

2、给定内核源码中的一个任意语句（非预处理，变量定义等语句），我们能在CFG中找到语句所处的位置吗？

3、给定一个变量定义语句（如 int a = 10;）我们能在CFG中找到它的位置吗？如果不能，我们能不能找到所有应用到这个变量（或其内部成员）的语句在CFG中的位置？

4、给定一个预处理语句(如#define, #include)，我们能在CFG中找到它的位置吗？

以badblocks\_check函数为例

int badblocks\_check(struct badblocks \*bb, sector\_t s, int sectors,

sector\_t \*first\_bad, int \*bad\_sectors)

{

int hi;

int lo;

u64 \*p = bb->page;

int rv;

sector\_t target = s + sectors;

unsigned seq;

if (bb->shift > 0)

{

/\* round the start down, and the end up \*/

s >>= bb->shift;

target += (1 << bb->shift) - 1;

target >>= bb->shift;

sectors = target - s;

}

/\* 'target' is now the first block after the bad range \*/

retry:

seq = read\_seqbegin(&bb->lock);

lo = 0;

rv = 0;

hi = bb->count;

/\* Binary search between lo and hi for 'target'

\* i.e. for the last range that starts before 'target'

\*/

/\* INVARIANT: ranges before 'lo' and at-or-after 'hi'

\* are known not to be the last range before target.

\* VARIANT: hi-lo is the number of possible

\* ranges, and decreases until it reaches 1

\*/

while (hi - lo > 1)

{

int mid = (lo + hi) / 2;

sector\_t a = BB\_OFFSET(p[mid]);

if (a < target)

/\* This could still be the one, earlier ranges

\* could not.

\*/

lo = mid;

else

/\* This and later ranges are definitely out. \*/

hi = mid;

}

/\* 'lo' might be the last that started before target, but 'hi' isn't \*/

if (hi > lo)

{

/\* need to check all range that end after 's' to see if

\* any are unacknowledged.

\*/

while (lo >= 0 && BB\_OFFSET(p[lo]) + BB\_LEN(p[lo]) > s)

{

if (BB\_OFFSET(p[lo]) < target)

{

/\* starts before the end, and finishes after

\* the start, so they must overlap

\*/

if (rv != -1 && BB\_ACK(p[lo]))

rv = 1;

else

rv = -1;

\*first\_bad = BB\_OFFSET(p[lo]);

\*bad\_sectors = BB\_LEN(p[lo]);

}

lo--;

}

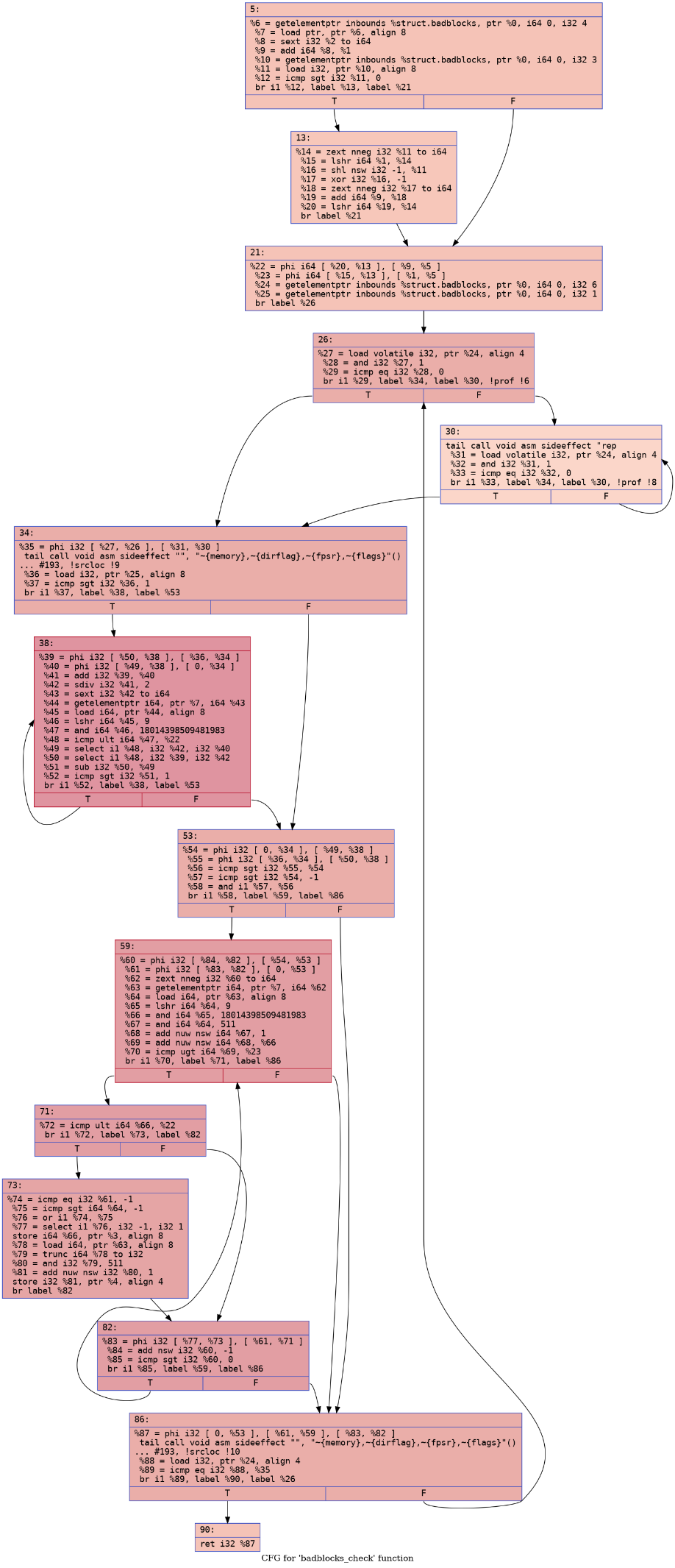
}

if (read\_seqretry(&bb->lock, seq))

goto retry;

return rv;

}



# 表达式语句(a=b+c)

sector\_t target = s + sectors;

%8 = sext i32 %2 to i64

%9 = add i64 %8, %1

# 选择语句(if, if-else, switch)

if (bb->shift > 0)

%12 = icmp sgt i32 %11, 0

br i1 %12, label %13, label %21

# 循环语句(for, while, do-while)

while (hi - lo > 1){}

38号块

# 跳转语句(break, continue, return, goto)

goto retry;

br label %retry

# 声明语句(int a, void function)

int hi;

这个声明在 LLVM IR 中没有直接的对应

# 预处理指令(#define #include)

#define #include为预处理指令，是在编译前被处理

生成cfg是编译之后的，所以找不到对应的部分