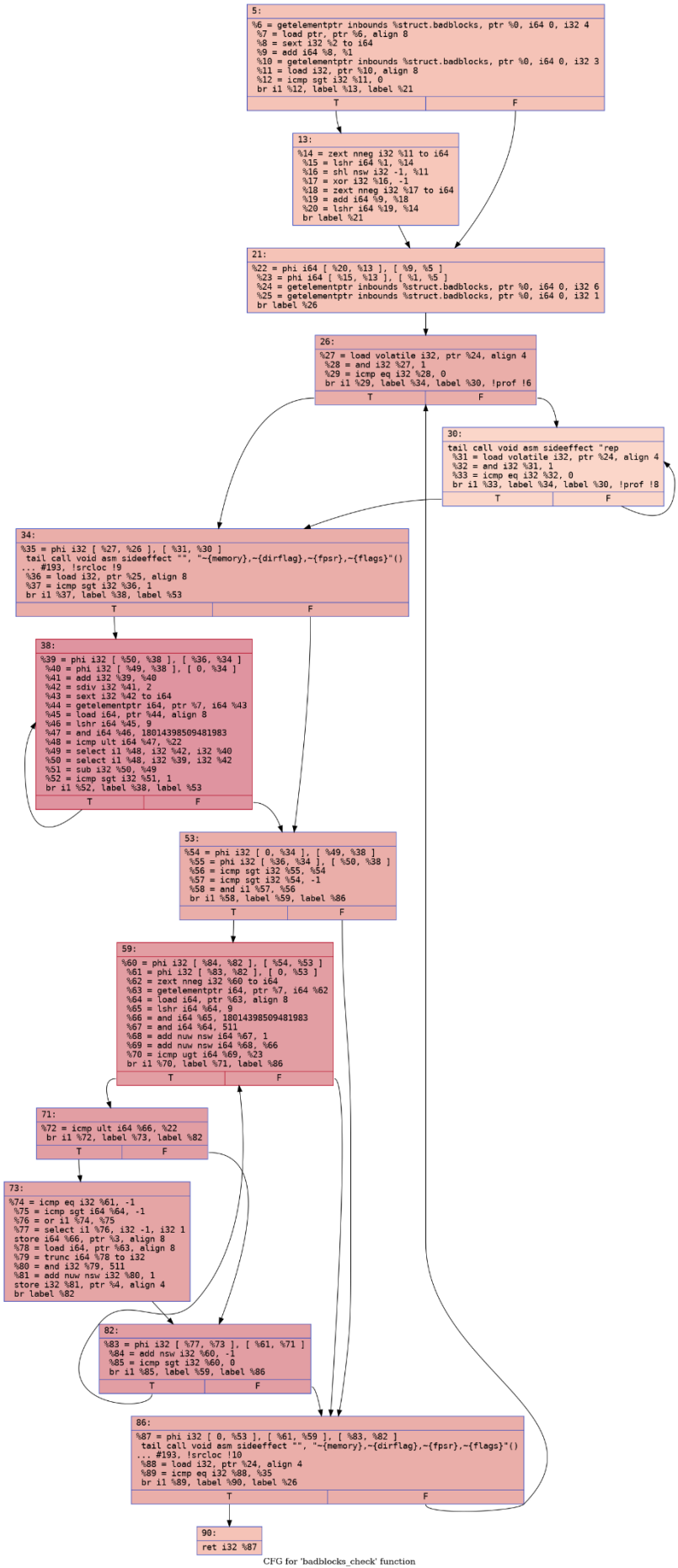
以 badblocks_check 函数为例

```
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;
}
```



CFG for 'badblocks_check' function

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