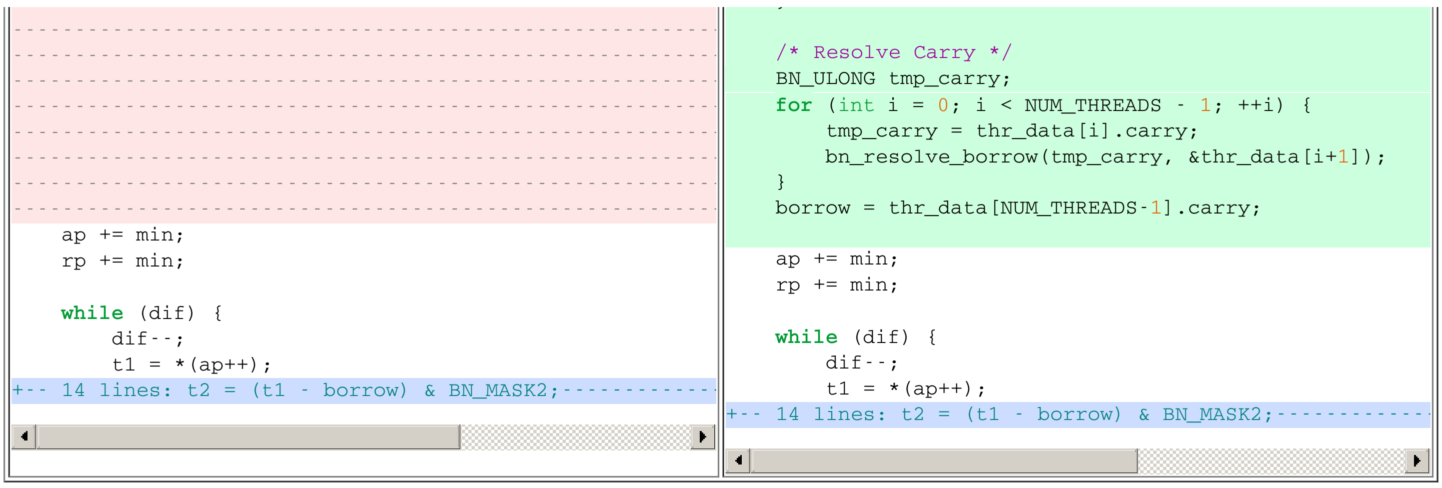
base_bn_add.c	mod_bn_add.c

```
+--  5 lines: Copyright 1995-2018 The OpenSSL Project Aut
 * in the file LICENSE in the source distribution or at
 * https://www.openssl.org/source/license.html
 */

#include "internal/cryptlib.h"
#include "bn_local.h"
----------------------------------------------------------

----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
/* signed add of b to a. */
int BN_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b)
{
    int ret, r_neg, cmp_res;

    bn_check_top(a);
+-- 38 lines: bn_check_top(b);---------------------------
        if (cmp_res > 0) {
            r_neg = a->neg;
            ret = BN_usub(r, a, b);
        } else if (cmp_res < 0) {
            r_neg = !b->neg;
            ret = BN_usub(r, b, a);
        } else {
            r_neg = 0;
            BN_zero(r);
            ret = 1;
        }
    }

    r->neg = r_neg;
    bn_check_top(r);
    return ret;
+--  6 lines: }------------------------------------------
    const BN_ULONG *ap, *bp;
    BN_ULONG *rp, carry, t1, t2;

    bn_check_top(a);
    bn_check_top(b);

----------------------------------------------------------
    if (a->top < b->top) {
        const BIGNUM *tmp;

        tmp = a;
        a = b;
        b = tmp;
+--  8 lines: }------------------------------------------
    r->top = max;

    ap = a->d;
    bp = b->d;
    rp = r->d;

    carry = bn_add_words(rp, ap, bp, min);
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
```

```
+--  5 lines: Copyright 1995-2018 The OpenSSL Project Aut
 * in the file LICENSE in the source distribution or at
 * https://www.openssl.org/source/license.html
 */

#include "internal/cryptlib.h"
#include "bn_local.h"
#include "bn_par.h"

void *bn_add_sub_words_thread(void *ptr) {
    BN_ULONG c;
    add_sub_args *args = (add_sub_args *) ptr;

    const BN_ULONG* ap = args->a;
    const BN_ULONG* bp = args->b;
    BN_ULONG* rp = args->r;
    BN_ULONG min = args->n;

    if (args->type == '+')
        c = bn_add_words(rp, ap, bp, min);
    else if (args->type == '-')
        c = bn_sub_words(rp, ap, bp, min);

    args->carry = c;
    pthread_exit(NULL);
}
/* signed add of b to a. */
int BN_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b)
{
    int ret, r_neg, cmp_res;

    bn_check_top(a);
+-- 38 lines: bn_check_top(b);---------------------------
        if (cmp_res > 0) {
            r_neg = a->neg;
            ret = BN_usub(r, a, b);
        } else if (cmp_res < 0) {
            r_neg = !b->neg;
            ret = BN_usub(r, b, a);
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
----------------------------------------------------------
        }
    }

    r->neg = r_neg;
    bn_check_top(r);
    return ret;
+--  6 lines: }------------------------------------------
    const BN_ULONG *ap, *bp;
    BN_ULONG *rp, carry, t1, t2;

    bn_check_top(a);
    bn_check_top(b);

    // a must be longer than b, if otherwise, swap
    if (a->top < b->top) {
        const BIGNUM *tmp;

        tmp = a;
        a = b;
        b = tmp;
+--  8 lines: }------------------------------------------
    r->top = max;

    ap = a->d;
    bp = b->d;
    rp = r->d;

    // thread init
    pthread_t thr[NUM_THREADS];
    int rc;

    /* create a thread_data_t argument array */
    add_sub_args thr_data[NUM_THREADS];

    /* create threads, divide array */
    int new_n = min/NUM_THREADS;
    rp += min;
    ap += min;

    while (dif) {
        dif--;
        t1 = *(ap++);
+-- 32 lines: t2 = (t1 + carry) & BN_MASK2;--------------
        return 0;

    ap = a->d;
    bp = b->d;
    rp = r->d;

    borrow = bn_sub_words(rp, ap, bp, min);
```

```
    int l_idx = 0;

    for (int i = 0; i < NUM_THREADS; ++i) {
        l_idx = new_n * i;
        // printf("l_idx %d, h_idx %d\n", l_idx, l_idx +
        thr_data[i].a = &ap[l_idx];
        thr_data[i].b = &bp[l_idx];
        thr_data[i].r = &rp[l_idx];
        thr_data[i].type = '+';

        if (i == (NUM_THREADS - 1))
            thr_data[i].n = new_n + min % NUM_THREADS;
        else
            thr_data[i].n = new_n;

        if ((rc = pthread_create(&thr[i], NULL, bn_add_su
          fprintf(stderr, "error: pthread_create, rc: %d
          return EXIT_FAILURE;
        }
    }
    /* block until all threads complete */
    for (int i = 0; i < NUM_THREADS; ++i) {
        pthread_join(thr[i], NULL);
        // printf("t%d %d\n", i, thr_data[i].carry);
    }

    /* Resolve Carry */
    BN_ULONG tmp_carry;
    for (int i = 0; i < NUM_THREADS - 1; ++i) {
        tmp_carry = thr_data[i].carry;
        bn_resolve_carry(tmp_carry, &thr_data[i+1]);
    }
    carry = thr_data[NUM_THREADS-1].carry;

    rp += min;
    ap += min;

    while (dif) {
        dif--;
        t1 = *(ap++);
+-- 32 lines: t2 = (t1 + carry) & BN_MASK2;--------------
        return 0;

    ap = a->d;
    bp = b->d;
    rp = r->d;

    // create threads
    pthread_t thr[NUM_THREADS];
    int rc;

    /* create a thread_data_t argument array */
    add_sub_args thr_data[NUM_THREADS];

    /* create threads, divide array */
    int new_n = min/NUM_THREADS;
    int l_idx = 0;

    for (int i = 0; i < NUM_THREADS; ++i) {
        l_idx = new_n * i;
        // printf("l_idx %d, h_idx %d\n", l_idx, l_idx +
        thr_data[i].a = &ap[l_idx];
        thr_data[i].b = &bp[l_idx];
        thr_data[i].r = &rp[l_idx];
        thr_data[i].type = '-';

        if (i == (NUM_THREADS - 1))
            thr_data[i].n = new_n + min % NUM_THREADS;
        else
            thr_data[i].n = new_n;

        if ((rc = pthread_create(&thr[i], NULL, bn_add_su
          fprintf(stderr, "error: pthread_create, rc: %d
          return EXIT_FAILURE;
        }
    }
    /* block until all threads complete */
    for (int i = 0; i < NUM_THREADS; ++i) {
        pthread_join(thr[i], NULL);
        // printf("t%d %d\n", i, thr_data[i].carry);
    }
```



```
--------------------------------------------------------------
--------------------------------------------------------------
--------------------------------------------------------------
--------------------------------------------------------------
--------------------------------------------------------------
--------------------------------------------------------------
--------------------------------------------------------------
--------------------------------------------------------------
    ap += min;
    rp += min;

    while (dif) {
        dif--;
        t1 = *(ap++);
+-- 14 lines: t2 = (t1 - borrow) & BN_MASK2;------------
```

```
    }

    /* Resolve Carry */
    BN_ULONG tmp_carry;
    for (int i = 0; i < NUM_THREADS - 1; ++i) {
        tmp_carry = thr_data[i].carry;
        bn_resolve_borrow(tmp_carry, &thr_data[i+1]);
    }
    borrow = thr_data[NUM_THREADS-1].carry;

    ap += min;
    rp += min;

    while (dif) {
        dif--;
        t1 = *(ap++);
+-- 14 lines: t2 = (t1 - borrow) & BN_MASK2;------------
```