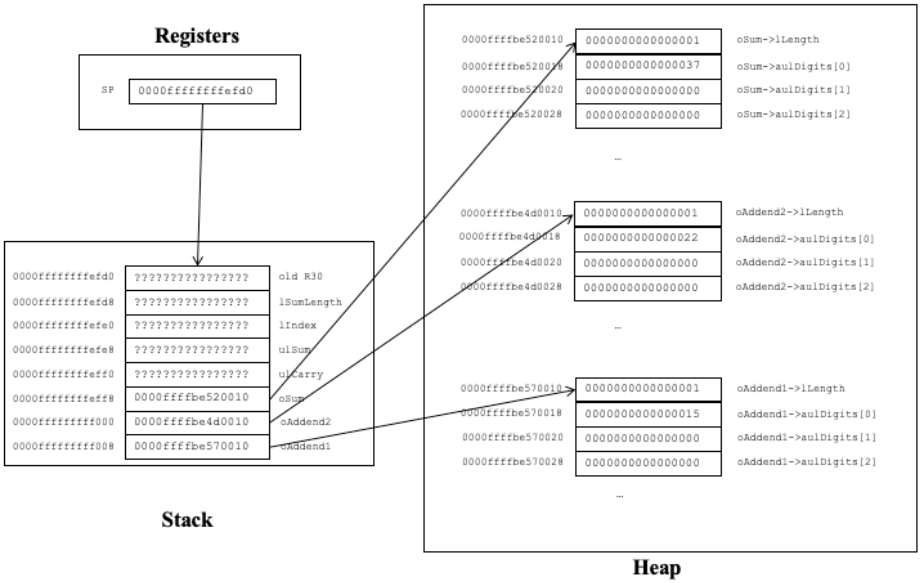
Princeton University
COS 217: Introduction to Programming Systems
The BigInt_add Function

```
enum {MAX_DIGITS = 32768};  /* Arbitrary */

…

struct BigInt
{
   long lLength;
   unsigned long aulDigits[MAX_DIGITS];
};

…

int BigInt_add(BigInt_T oAddend1, BigInt_T oAddend2, BigInt_T oSum)
{
   unsigned long ulCarry;
   unsigned long ulSum;
   long lIndex;
   long lSumLength;
   …
}
```

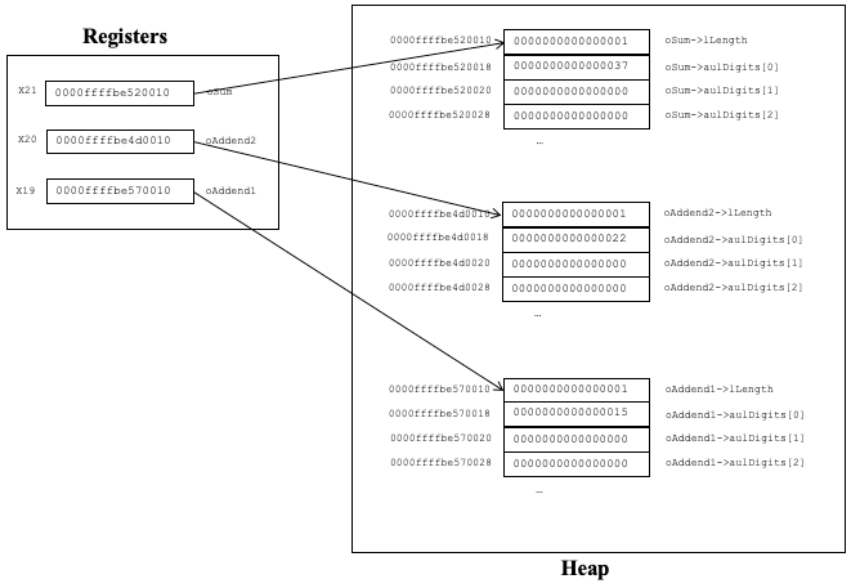
Princeton University
COS 217: Introduction to Programming Systems
The BigInt_add Function: Memory Map

Your addresses may differ

Normal Pattern



Optimized Pattern



Princeton University
COS 217: Introduction to Programming Systems
The BigInt_add Function: Code

Example Code: Access `oAddend2->aulDigits[2]`

Normal Pattern

Using register addressing:

```
mov x0, sp          // X0 contains 0000fffffffefd0 (hex)
                    // X0 contains the addr of the top of stack
add x0, x0, 48      // X0 contains 0000fffffffff000
                    // X0 contains &oAddend2
ldr x0, [x0]        // X0 contains 0000ffffbe4d0010 (hex)
                    // X0 contains oAddend2
add x0, x0, 8       // X0 contains 0000ffffbe4d0018(hex)
                    // X0 contains oAddend2->aulDigits
mov x1, 2           // X1 contains 0000000000000002(hex)
                    // X1 contains the index
lsl x1, x1, 3       // X1 contains 0000000000000010(hex)
                    // X1 contains a byte offset
add x0, x0, x1      // X0 contains 0000ffffbe4d0028(hex)
                    // X0 contains oAddend2->aulDigits + 2
ldr x0, [x0]        // X0 contains 0000000000000000(hex)
                    // X0 contains *(oAddend2->aulDigits + 2)
                    // X0 contains oAddend2->aulDigits[2]
```

Using scaled register offset addressing:

```
ldr x0, [sp, 48]          // X0 contains 0000ffffbe4d0010(hex)
                          // X0 contains oAddend2
add x0, x0, 8             // X0 contains 0000ffffbe4d0018(hex)
                          // X0 contains oAddend2->aulDigits
mov x1, 2                 // x1 contains 0000000000000002(hex)
                          // x1 contains the index
ldr x0, [x0, x1, lsl 3]   // X0 contains 0000000000000000(hex)
                          // X0 contains oAddend2->auiDigits[2]
```

Optimized Pattern

Using register addressing:

```
mov x0, x20         // X0 contains 0000ffffbe4d0010 (hex)
                    // X0 contains oAddend2
add x0, x0, 8       // X0 contains 0000ffffbe4d0018(hex)
                    // X0 contains oAddend2->aulDigits
mov x1, 2           // X1 contains 0000000000000002(hex)
                    // X1 contains the index
lsl x1, x1, 3       // X1 contains 0000000000000010(hex)
                    // X1 contains a byte offset
add x0, x0, x1      // X0 contains 0000ffffbe4d0028(hex)
                    // X0 contains oAddend2->aulDigits + 2
ldr x0, [x0]        // X0 contains 0000000000000000(hex)
                    // X0 contains *(oAddend2->aulDigits + 2)
                    // X0 contains oAddend2->aulDigits[2]
```

Using scaled register offset addressing:

```
mov x0, x20               // X0 contains 0000ffffbe4d0010 (hex)
                          // X0 contains oAddend2
add x0, x0, 8             // X0 contains 0000ffffbe4d0018(hex)
                          // X0 contains oAddend2->aulDigits
mov x1, 2                 // X1 contains 0000000000000002(hex)
                          // X1 contains the index
ldr x0, [x0, x1, lsl 3]   // X0 contains 0000000000000000(hex)
                          // X0 contains *(oAddend2->aulDigits + 2)
                          // X0 contains oAddend2->aulDigits[2]
```