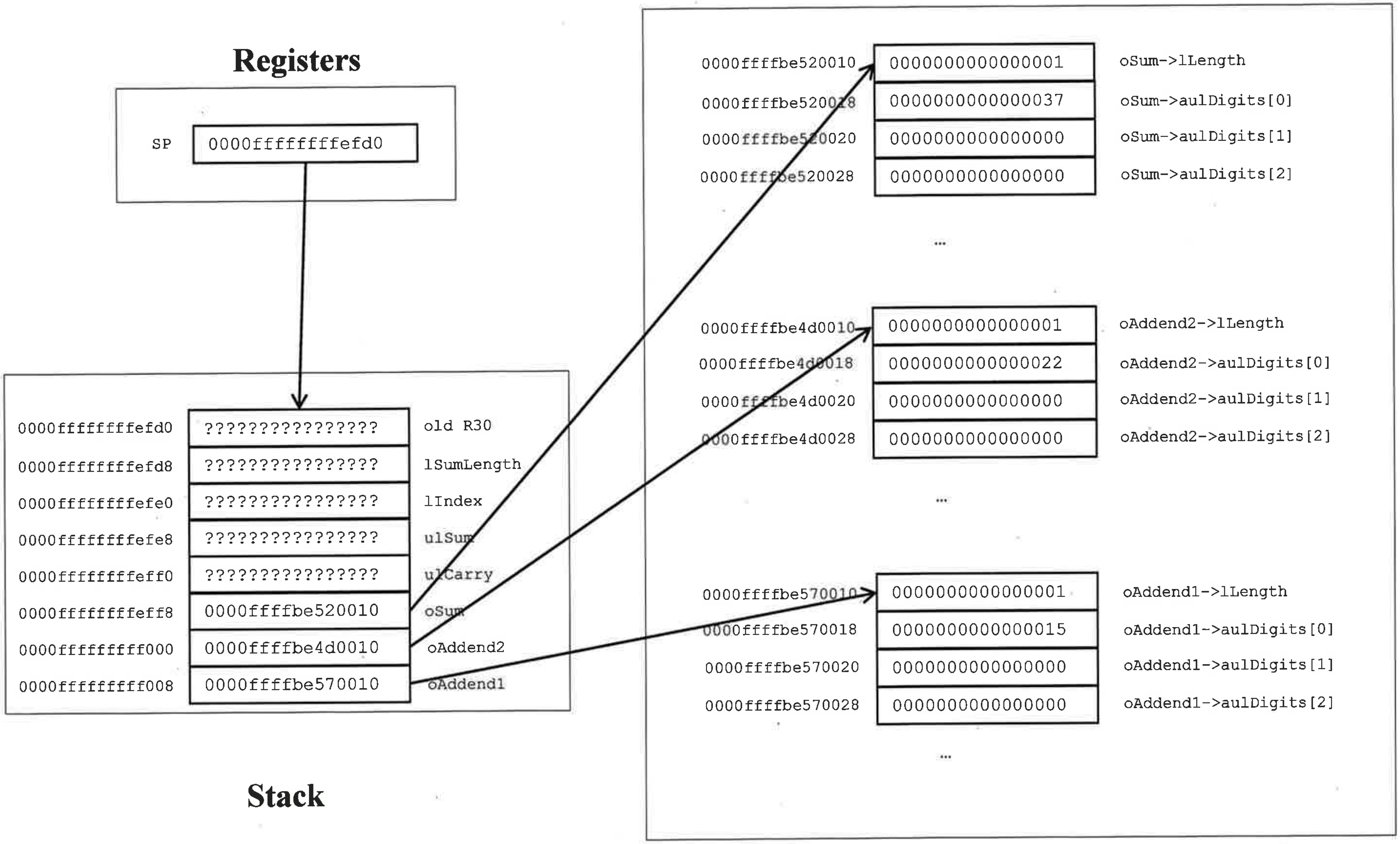
Princeton University
COS 217: Introduction to Programming Systems
The BigInt_add Function

```
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;
   ...
}
```

Precept 21
Week 12 Mon/Tue

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

Your addresses may differ



Registers

SP	0000ffffffffefd0

Stack

0000ffffffffefd0	????????????????	old R30
0000ffffffffefd8	????????????????	lSumLength
0000ffffffffefe0	????????????????	lIndex
0000ffffffffefe8	????????????????	ulSum
0000ffffffffeff0	????????????????	ulCarry
0000ffffffffeff8	0000ffffbe520010	oSum
0000fffffffff000	0000ffffbe4d0010	oAddend2
0000fffffffff008	0000ffffbe570010	oAddend1

Heap

0000ffffbe520010	0000000000000001	oSum->lLength
0000ffffbe520018	0000000000000037	oSum->aulDigits[0]
0000ffffbe520020	0000000000000000	oSum->aulDigits[1]
0000ffffbe520028	0000000000000000	oSum->aulDigits[2]
	...	
0000ffffbe4d0010	0000000000000001	oAddend2->lLength
0000ffffbe4d0018	0000000000000022	oAddend2->aulDigits[0]
0000ffffbe4d0020	0000000000000000	oAddend2->aulDigits[1]
0000ffffbe4d0028	0000000000000000	oAddend2->aulDigits[2]
	...	
0000ffffbe570010	0000000000000001	oAddend1->lLength
0000ffffbe570018	0000000000000015	oAddend1->aulDigits[0]
0000ffffbe570020	0000000000000000	oAddend1->aulDigits[1]
0000ffffbe570028	0000000000000000	oAddend1->aulDigits[2]
	...	

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

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

Using register addressing:

```
mov x0, sp          // X0 contains 0000ffffffffefd0 (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]
```

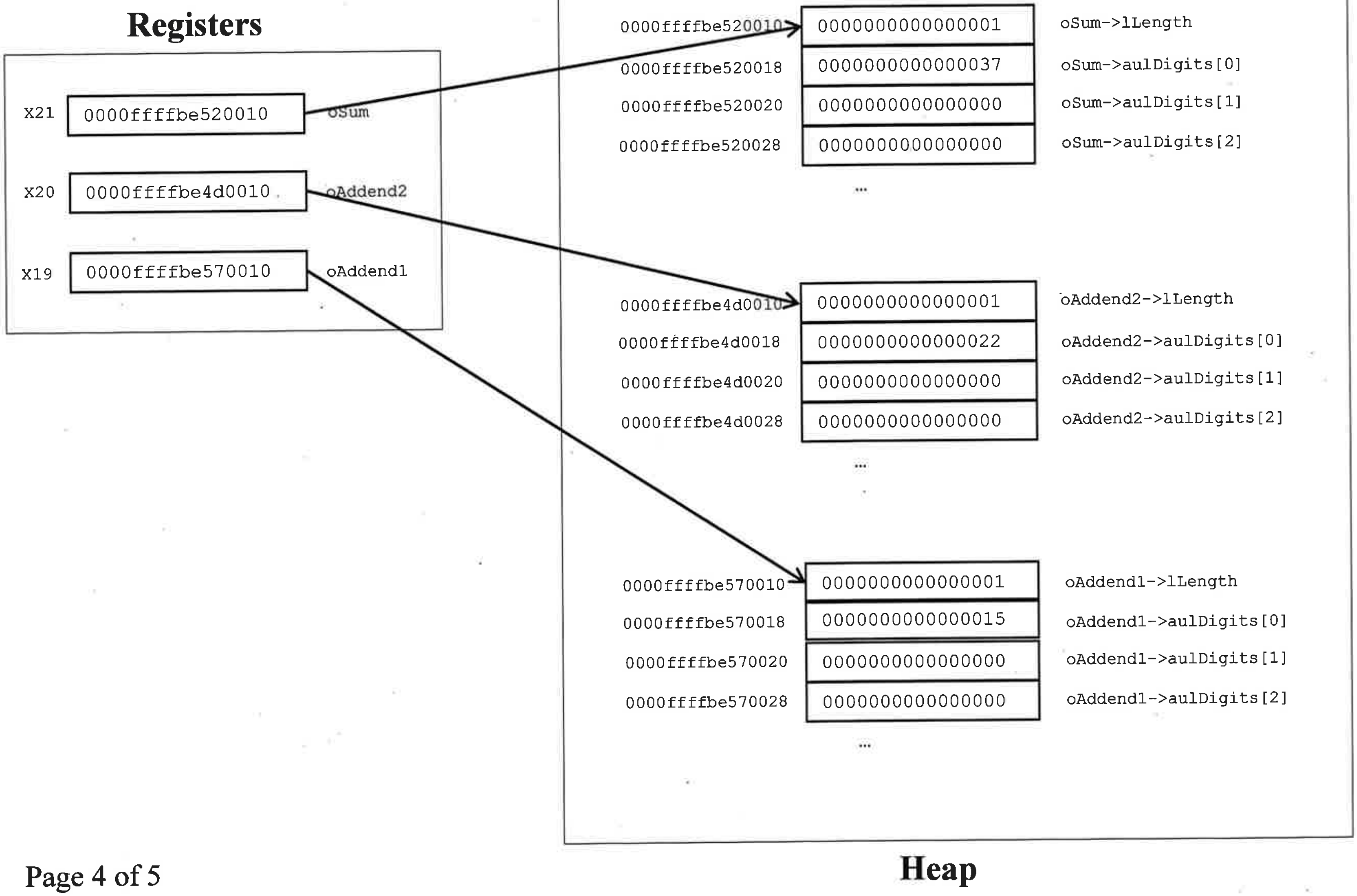
Using scaled register offset addressing:

```
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]
```

w

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

Your addresses may differ



Registers

X21	0000ffffbe520010	oSum
X20	0000ffffbe4d0010	oAddend2
X19	0000ffffbe570010	oAddend1

Heap

0000ffffbe520010	0000000000000001	oSum->lLength
0000ffffbe520018	0000000000000037	oSum->aulDigits[0]
0000ffffbe520020	0000000000000000	oSum->aulDigits[1]
0000ffffbe520028	0000000000000000	oSum->aulDigits[2]
	...	
0000ffffbe4d0010	0000000000000001	oAddend2->lLength
0000ffffbe4d0018	0000000000000022	oAddend2->aulDigits[0]
0000ffffbe4d0020	0000000000000000	oAddend2->aulDigits[1]
0000ffffbe4d0028	0000000000000000	oAddend2->aulDigits[2]
	...	
0000ffffbe570010	0000000000000001	oAddend1->lLength
0000ffffbe570018	0000000000000015	oAddend1->aulDigits[0]
0000ffffbe570020	0000000000000000	oAddend1->aulDigits[1]
0000ffffbe570028	0000000000000000	oAddend1->aulDigits[2]
	...	

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

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

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]
```

Precept Activity Instructions:

Manually identify (not using gdb) and fix bugs in the following instructions (normal pattern, use the memory map page 2 for reference):

1) load oAddend1->lLength into x0:

```
ldr x0, [sp, 56]
```

2) Load oAddend1->aulDigits[3] into x0:

```
ldr x0, [sp, 56]
mov x1, 3
ldr x0, [x0, x1, lsl 3]
```

3) Store ulSum into oSum->aulDigits[5]:

```
ldr x0, [sp, 24]
ldr x1, [sp, 40]
add x1, 8
mov x2, 5
ldr x3, [x1, x2, lsl 3]
str x0, [x3]
```

Precept Activity Answers:

1) load oAddend1->lLength into x0

```
ldr x0, [sp, 56]        // put oAddend1 pointer into x0
ldr x0, [x0]            // store oAddend1->lLength in x0
```

2) Load oAddend1->aulDigits[3] into x0

```
ldr x0, [sp, 56]        // put oAddend1 pointer into x0
add x0, x0, 8           // add array offset
mov x1, 3               // put array index into x1
ldr x0, [x0, x1, lsl 3] // oAddend1>aulDigits[3] into x0
```

3) Store ulSum into oSum->aulDigits[5]

```
ldr x0, [sp, 24]        // put ulSum into x0
ldr x1, [sp, 40]        // put oSum pointer into x1
add x1, x1, 8           // add array offset correctly
mov x2, 5               // put array index into x2
str x0, [x1, x2, lsl 3] // store ulSum in oSum->aulDigits[5]
```