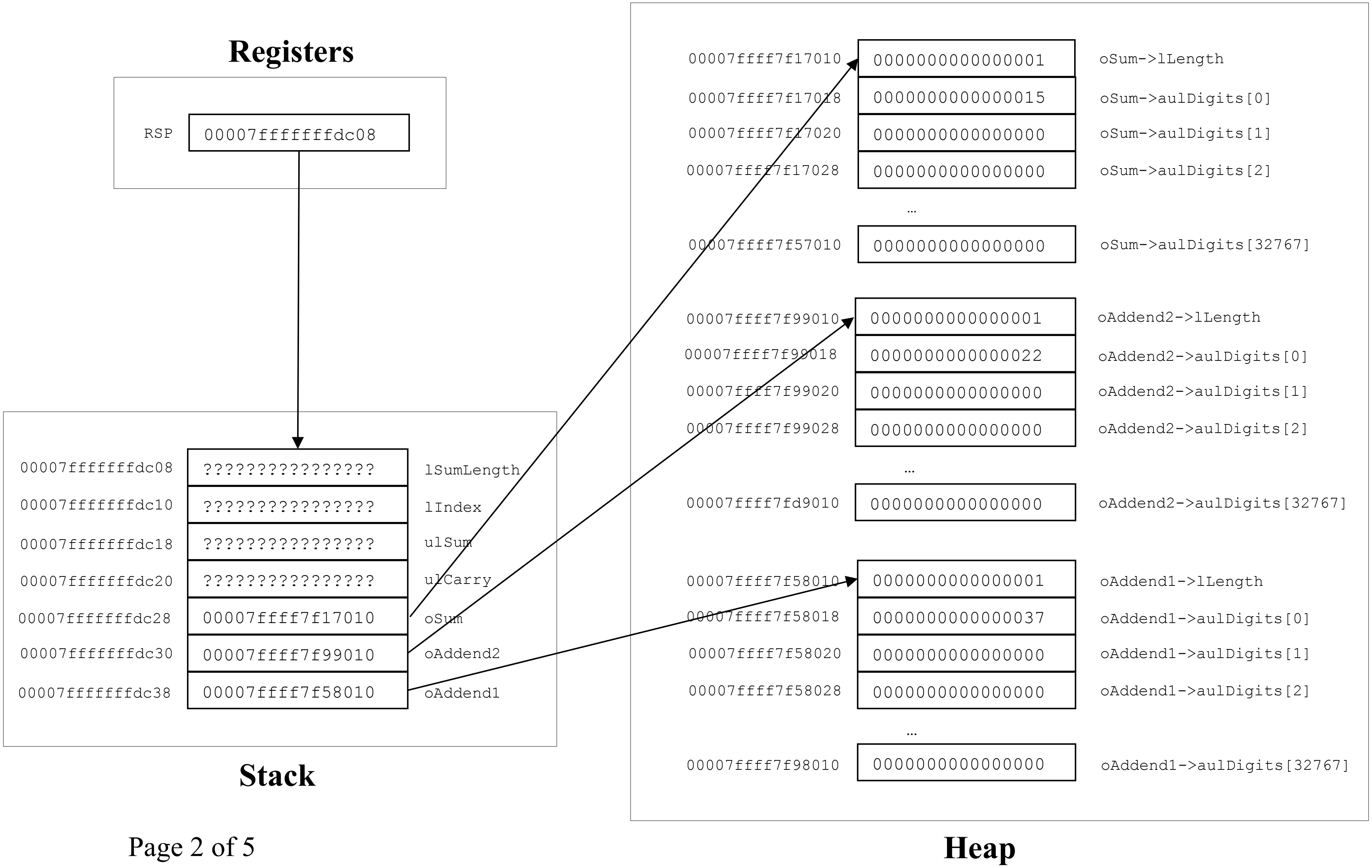
Princeton University
COS 217: Introduction to Programming Systems
The BigInt_add Function

```
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: Normal Pattern

Your addresses may differ



Registers

RSP	00007fffffffdc08

Stack

00007fffffffdc08	????????????????	lSumLength
00007fffffffdc10	????????????????	lIndex
00007fffffffdc18	????????????????	ulSum
00007fffffffdc20	????????????????	ulCarry
00007fffffffdc28	00007ffff7f17010	oSum
00007fffffffdc30	00007ffff7f99010	oAddend2
00007fffffffdc38	00007ffff7f58010	oAddend1

Heap

00007ffff7f17010	0000000000000001	oSum->lLength
00007ffff7f17018	0000000000000015	oSum->aulDigits[0]
00007ffff7f17020	0000000000000000	oSum->aulDigits[1]
00007ffff7f17028	0000000000000000	oSum->aulDigits[2]
	…	
00007ffff7f57010	0000000000000000	oSum->aulDigits[32767]
00007ffff7f99010	0000000000000001	oAddend2->lLength
00007ffff7f99018	0000000000000022	oAddend2->aulDigits[0]
00007ffff7f99020	0000000000000000	oAddend2->aulDigits[1]
00007ffff7f99028	0000000000000000	oAddend2->aulDigits[2]
	…	
00007ffff7fd9010	0000000000000000	oAddend2->aulDigits[32767]
00007ffff7f58010	0000000000000001	oAddend1->lLength
00007ffff7f58018	0000000000000037	oAddend1->aulDigits[0]
00007ffff7f58020	0000000000000000	oAddend1->aulDigits[1]
00007ffff7f58028	0000000000000000	oAddend1->aulDigits[2]
	…	
00007ffff7f98010	0000000000000000	oAddend1->aulDigits[32767]

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

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

Using indirect addressing:

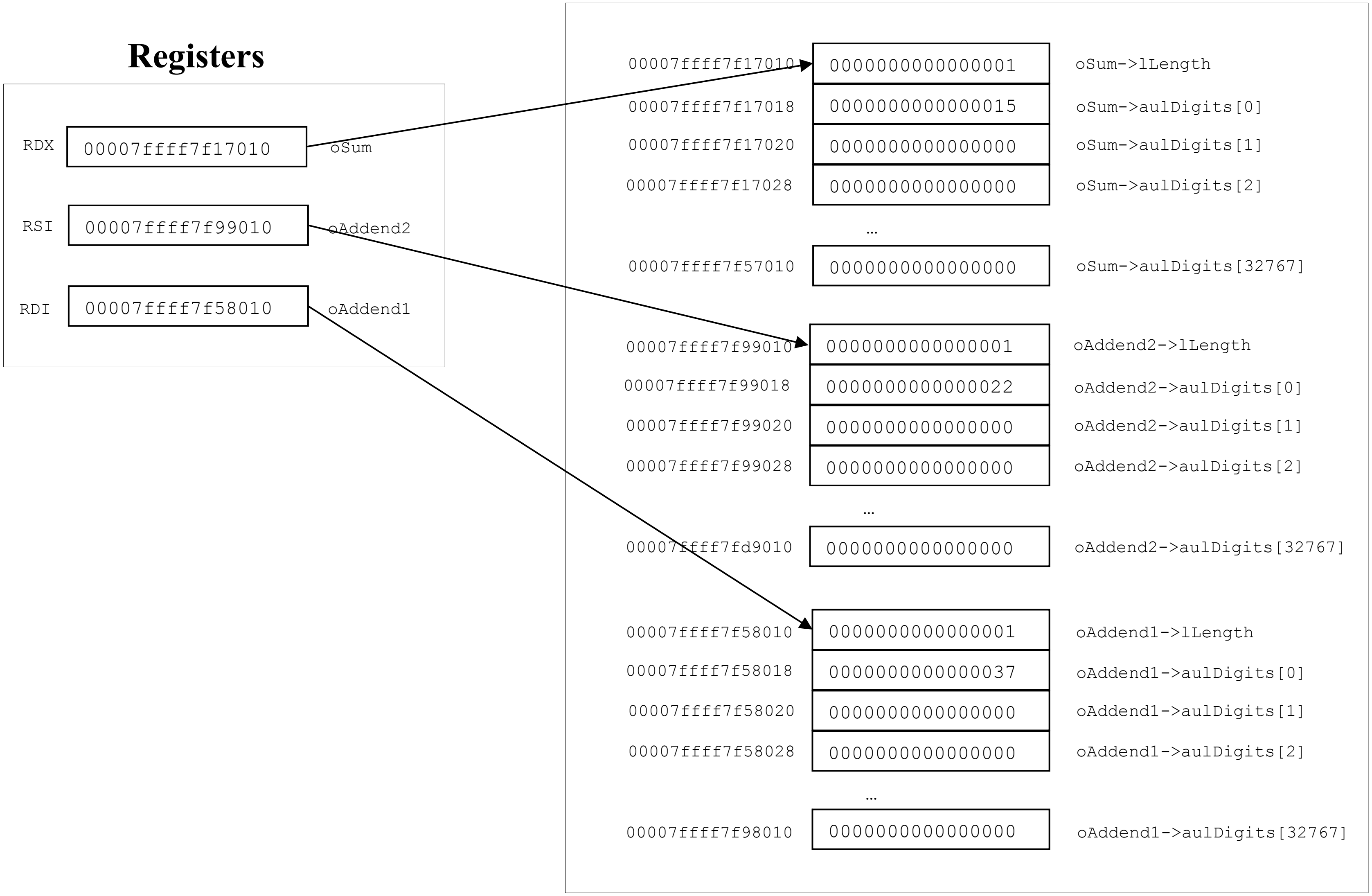
```
movq %rsp, %rax    # RAX contains 00007fffffffdc08 (hex)
                   # RAX contains the addr of the top of stack
addq $40, %rax     # RAX contains 00007fffffffdc30
                   # RAX contains &oAddend2
movq (%rax), %rax  # RAX contains 00007ffff7f99010 (hex)
                   # RAX contains oAddend2
addq $8, %rax      # RAX contains 00007ffff7f99018(hex)
                   # RAX contains oAddend2->aulDigits
movq $2, %r10      # R10 contains 0000000000000002(hex)
                   # R10 contains the index
salq $3, %r10      # R10 contains 0000000000000010(hex)
                   # R10 contains a byte offset
addq %r10, %rax    # RAX contains 00007ffff7f99028(hex)
                   # RAX contains oAddend2->aulDigits + 2
movq (%rax), %rax  # RAX contains 0000000000000000(hex)
                   # RAX contains *(oAddend2->aulDigits + 2)
                   # RAX contains oAddend2->aulDigits[2]
```

Using scaled-indexed addressing:

```
movq 40(%rsp), %rax         # RAX contains 00007ffff7f99010(hex)
                            # RAX contains oAddend2
movq $2, %r10               # R10 contains 0000000000000002(hex)
                            # R10 contains the index
movq 8(%rax, %r10, 8), %rax # RAX contains 0000000000000000(hex)
                            # RAX contains oAddend2->auiDigits[2]
```

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

Your addresses may differ



Registers

RDX	00007ffff7f17010	oSum
RSI	00007ffff7f99010	oAddend2
RDI	00007ffff7f58010	oAddend1

00007ffff7f17010	0000000000000001	oSum->lLength
00007ffff7f17018	0000000000000015	oSum->aulDigits[0]
00007ffff7f17020	0000000000000000	oSum->aulDigits[1]
00007ffff7f17028	0000000000000000	oSum->aulDigits[2]
	…	
00007ffff7f57010	0000000000000000	oSum->aulDigits[32767]
00007ffff7f99010	0000000000000001	oAddend2->lLength
00007ffff7f99018	0000000000000022	oAddend2->aulDigits[0]
00007ffff7f99020	0000000000000000	oAddend2->aulDigits[1]
00007ffff7f99028	0000000000000000	oAddend2->aulDigits[2]
	…	
00007ffff7fd9010	0000000000000000	oAddend2->aulDigits[32767]
00007ffff7f58010	0000000000000001	oAddend1->lLength
00007ffff7f58018	0000000000000037	oAddend1->aulDigits[0]
00007ffff7f58020	0000000000000000	oAddend1->aulDigits[1]
00007ffff7f58028	0000000000000000	oAddend1->aulDigits[2]
	…	
00007ffff7f98010	0000000000000000	oAddend1->aulDigits[32767]

Heap

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

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

Using indirect addressing:

```
movq %rsi, %rax    # RAX contains 00007ffff7f99010 (hex)
                   # RAX contains oAddend2
addq $8, %rax      # RAX contains 00007ffff7f99018(hex)
                   # RAX contains oAddend2->aulDigits
movq $2, %r10      # R10 contains 0000000000000002(hex)
                   # R10 contains the index
salq $3, %r10      # R10 contains 0000000000000010(hex)
                   # R10 contains a byte offset
addq %r10, %rax    # RAX contains 00007ffff7f99028(hex)
                   # RAX contains oAddend2->aulDigits + 2
movq (%rax), %rax  # RAX contains 0000000000000000(hex)
                   # RAX contains *(oAddend2->aulDigits + 2)
                   # RAX contains oAddend2->aulDigits[2]
```

Using scaled-indexed addressing:

```
movq $2, %r10                # R10 contains 0000000000000002(hex)
                             # R10 contains the index
movq 8(%rsi, %r10, 8), %rax  # RAX contains 0000000000000000(hex)
                             # RAX contains oAddend2->auiDigits[2}
```