Operating Systems

Segmentation

Hongliang Liang, BUPT

Sep. 2023

Problems with base-and-bounds virtualization

Internal fragmentation: wasted space between the heap and the stack

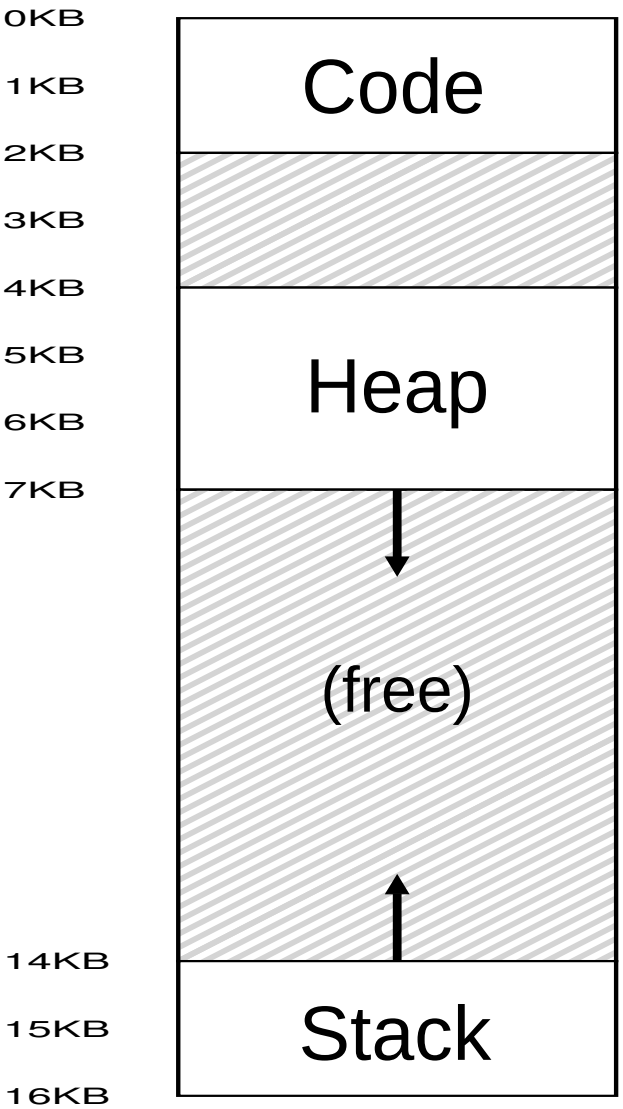
The address space is assumed to fit into the physical memory

Segmentation

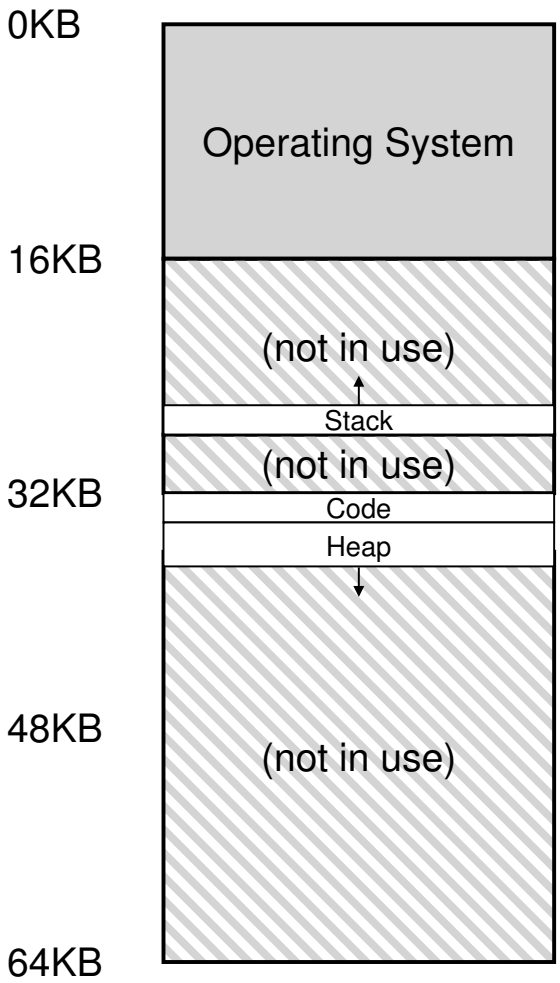
Basic idea: a pair of base and bounds register values for **each logical segment** in the address space

Accommodates **sparse address spaces**, with large amounts of unused address space between the heap and the stack

Example: an address space



Example: placing segments in physical memory



Example: segment register values in the last figure

Segment	Base	Size
Code	32K	2K
Heap	34K	2K
Stack	28K	2K

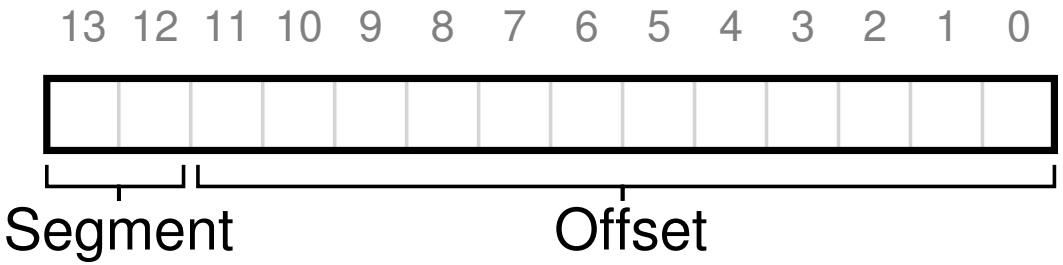
Which segment are we referring to?

The hardware needs to know which segment an address belongs to, in order to compute the offset

Which segment are we referring to?

The hardware needs to know which segment an address belongs to, in order to compute the offset

An **explicit** approach: use the top several bits in the virtual address



Example translation

```
Segment = (Virtual Address & SEG_MASK) >> SEG_SHIFT
Offset = Virtual Address & OFFSET_MASK
if (Offset >= Bounds[Segment])
  RaiseException(SEGMENTATION_FAULT)
else
  PhysicalAddr = Base[Segment] + Offset
```

What about the stack segment?

The stack grows downwards to lower addresses

In addition to base and bounds values, the hardware also needs to know which way the segment grows

Segment	Base	Size	Grows Positive?
Code	32K	2K	1
Heap	34K	2K	1
Stack	28K	2K	0

Support for sharing

To conserve physical memory, sometimes it is useful to share certain memory segments between address spaces

- Example: code sharing

To support sharing, add a few **protection bits**

Segment	Base	Size	GrowsPositive?	Protection
Code	32K	2K	1	Read-Execute
Heap	34K	2K	1	Read-Write
Stack	28K	2K	0	Read-Write

OS Support

On a context switch, save and restore segment registers

What happens when a user process calls `malloc()` to allocate an object on the heap?

- The heap segment may need to grow, by performing a system call — `sbrk()`

How do we manage free space in the physical memory?

- Segment sizes vary, rather than assuming the same size for address spaces

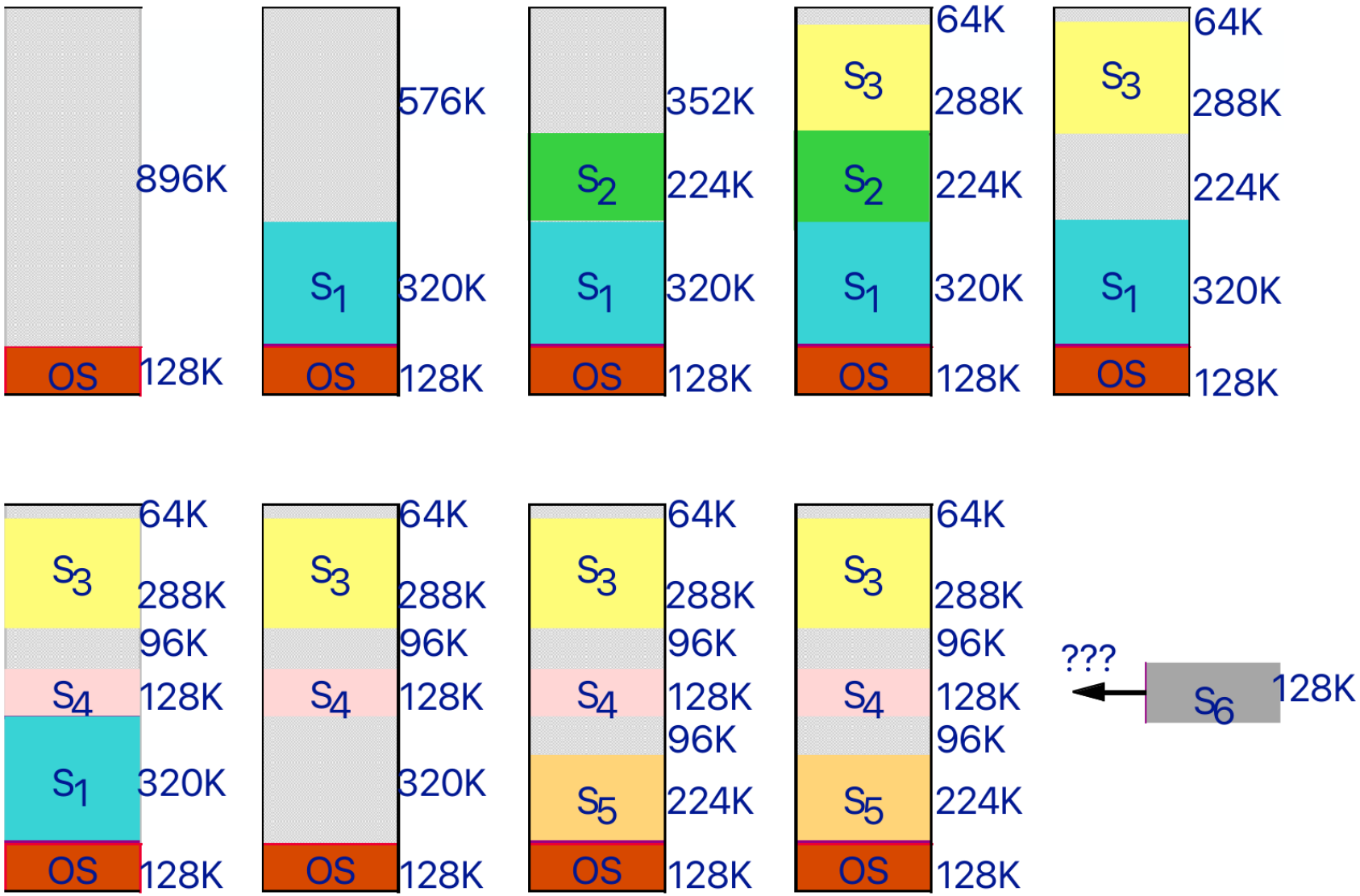
When a new address space is created

The OS needs to find space in the physical memory for its segments

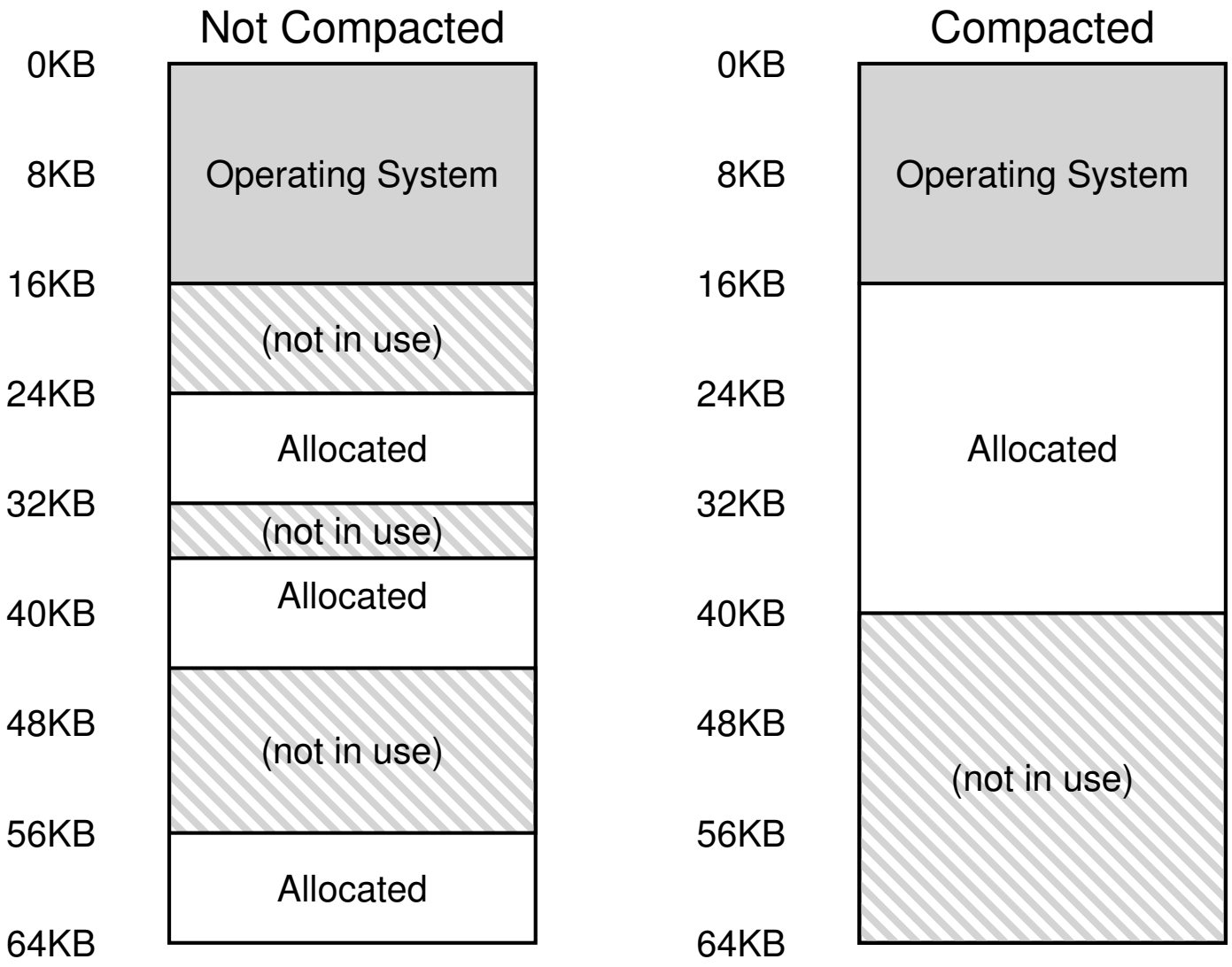
We now have a few segments in each address space, but they may have **different sizes**

As address spaces are created and removed, the physical memory quickly becomes full of little holes of free space — called **external fragmentation**

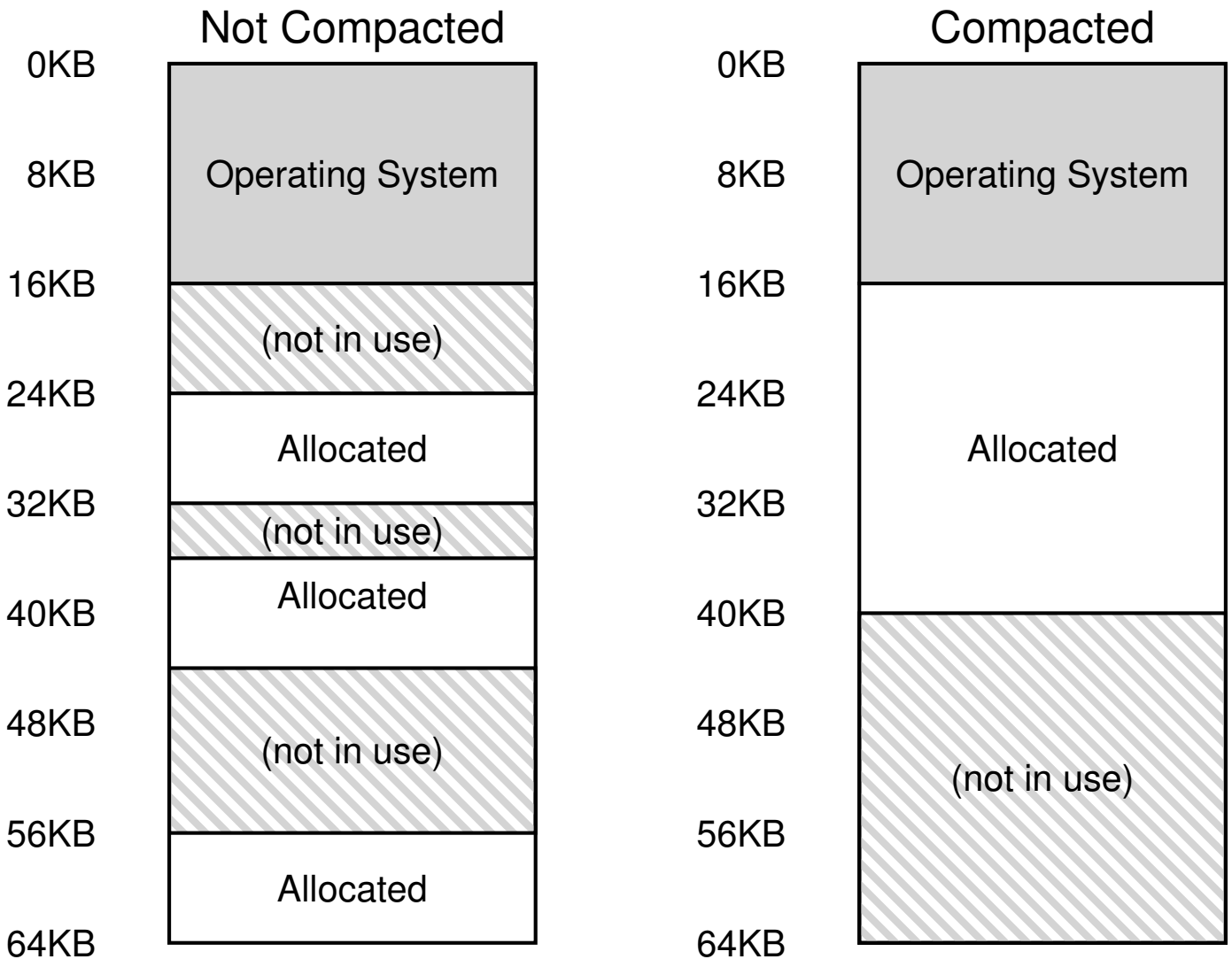
Example: external framentation



Solution: Compaction



Solution: Compaction



But copying segments is memory-intensive — heavy overhead