Chapter 16: Segmentation

Adam Disney

THE UNIVERSITY OF TENNESSEE
KNOXVILLE

Crux: How To Support A Large Address Space

THE CRUX: HOW TO SUPPORT A LARGE ADDRESS SPACE

How do we support a large address space with (potentially) a lot of free space between the stack and the heap? Note that in our examples, with tiny (pretend) address spaces, the waste doesn't seem too bad. Imagine, however, a 32-bit address space (4 GB in size); a typical program will only use megabytes of memory, but still would demand that the entire address space be resident in memory.

Segmentation: Generalized Base/Bounds

- Instead of a single base/bounds pair, why not have a pair per logical segment of the address space?
- Let's split the 3 parts of the address space into segments and we can place them individually into physical memory.
- This avoids wasting the space in-between.

Segmentation: Generalized Base/Bounds

Segment	Base	Size
Code	32K	2K
Heap	34K	3K
Stack	28K	2K

Figure 16.3: **Segment Register Values**

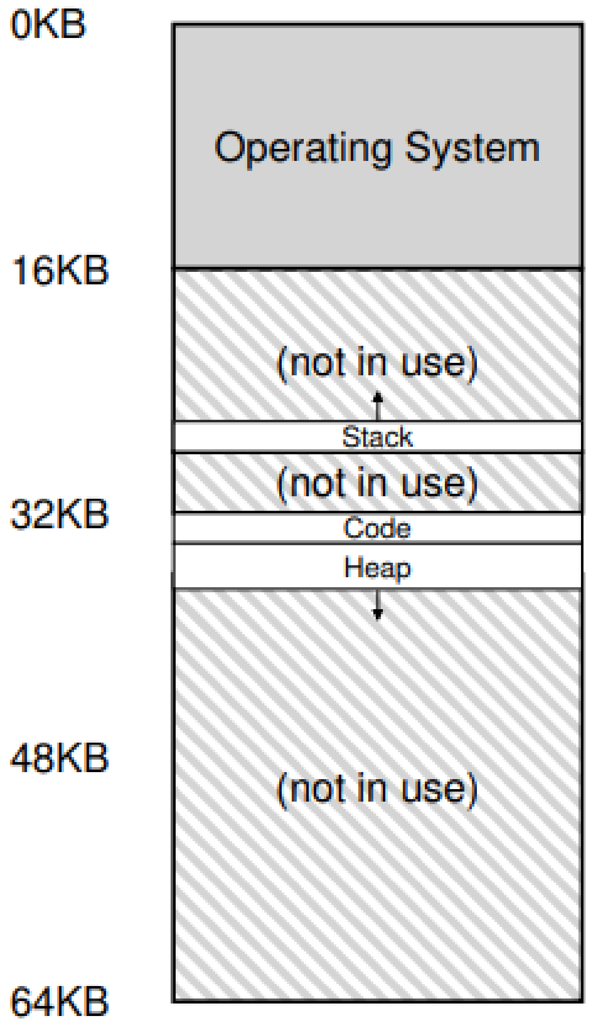


Figure 16.2: **Placing Segments In Physical Memory**

Address Translation

Segment	Base	Size
Code	32K	2K
Heap	34K	3K
Stack	28K	2K

Figure 16.3: **Segment Register Values**

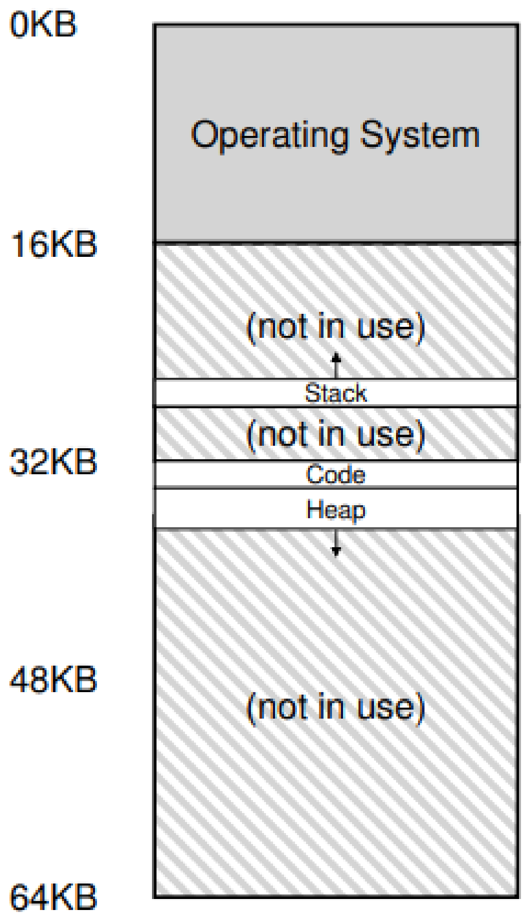


Figure 16.2: **Placing Segments In Physical Mer**

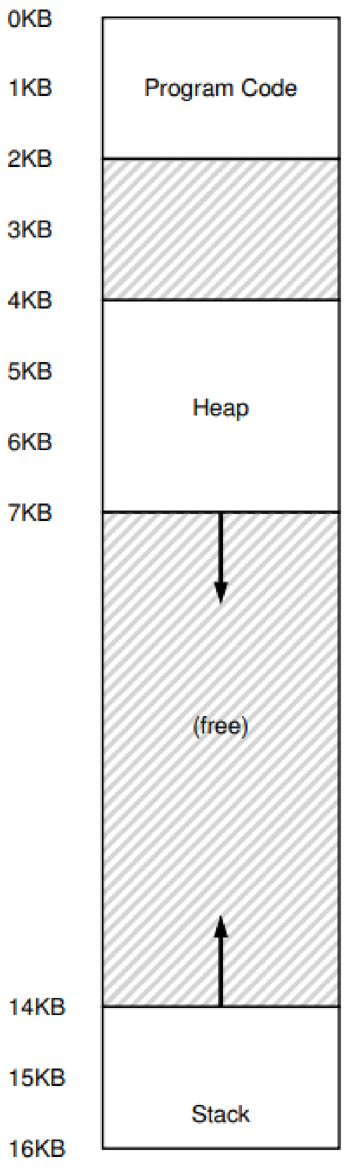
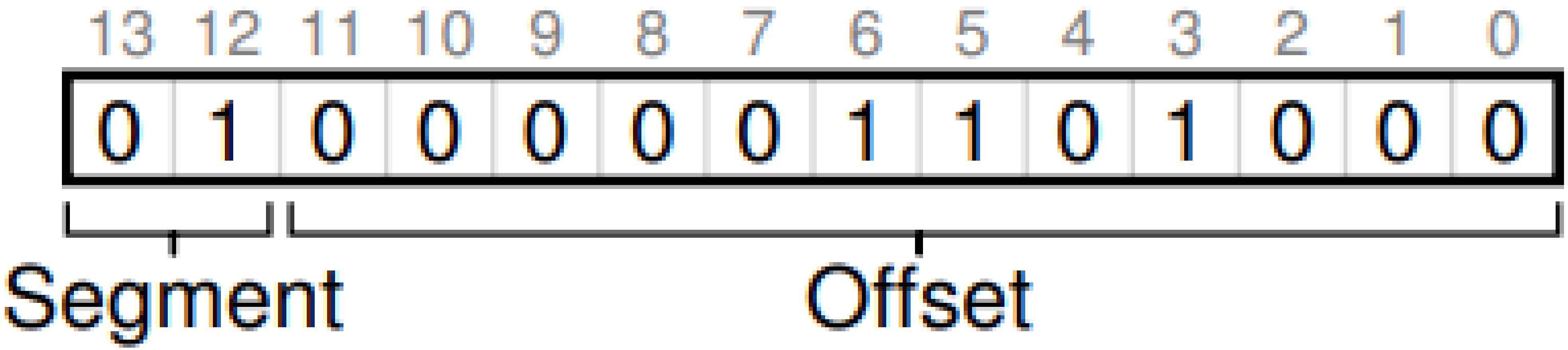


Figure 16.1: **An Address Space (Again)**

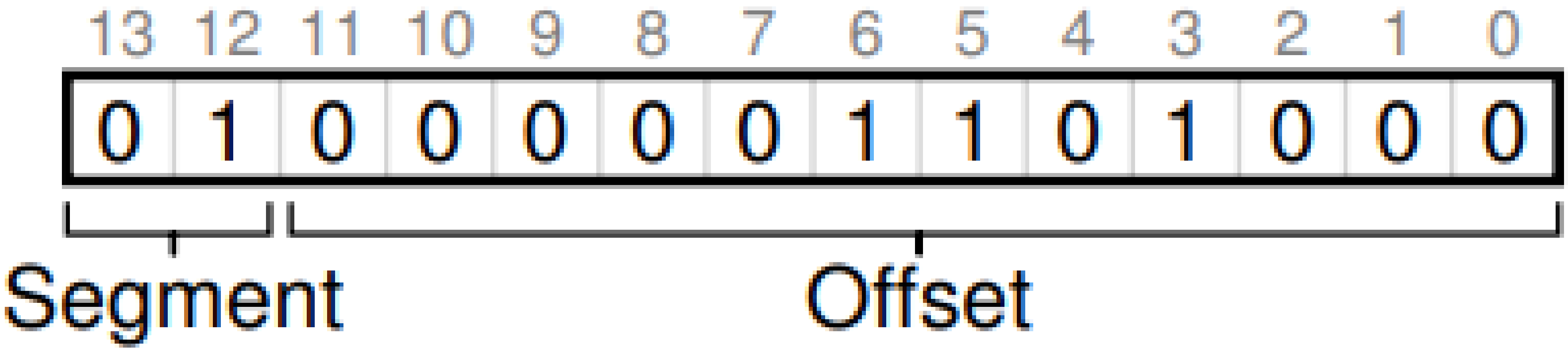
Which Segment Are We Referring To?

- **Explicit** approach
  - Use top few bits of the virtual address to determine which segment we're in.



Which Segment Are We Referring To?

- **Explicit** approach issues
  - With only 3 segments, we're wasting virtual address space with two bits.
  - Alternatively, we could make the code+heap as a single segment and use one bit.
  - Segments are limited in size by the offset field



Which Segment Are We Referring To?

- **Implicit** approach
  - Hardware determines segment by noticing how the address was formed.
  - Program counter address -> code segment
  - Address based off stack pointer -> stack segment
  - Any other -> heap segment

What About The Stack?

- Our current approach doesn't work for stack because it grows backwards!
- Need some extra hardware support to mark segment direction

Segment	Base	Size (max 4K)	Grows Positive?
$Code_{00}$	32K	2K	1
$Heap_{01}$	34K	3K	1
$Stack_{11}$	28K	2K	0

Figure 16.4: **Segment Registers (With Negative-Growth Support)**

Address Translation

Segment	Base	Size (max 4K)	Grows Positive?
$Code_{00}$	32K	2K	1
$Heap_{01}$	34K	3K	1
$Stack_{11}$	28K	2K	0

Figure 16.4: **Segment Registers (With Negative-Growth Support)**

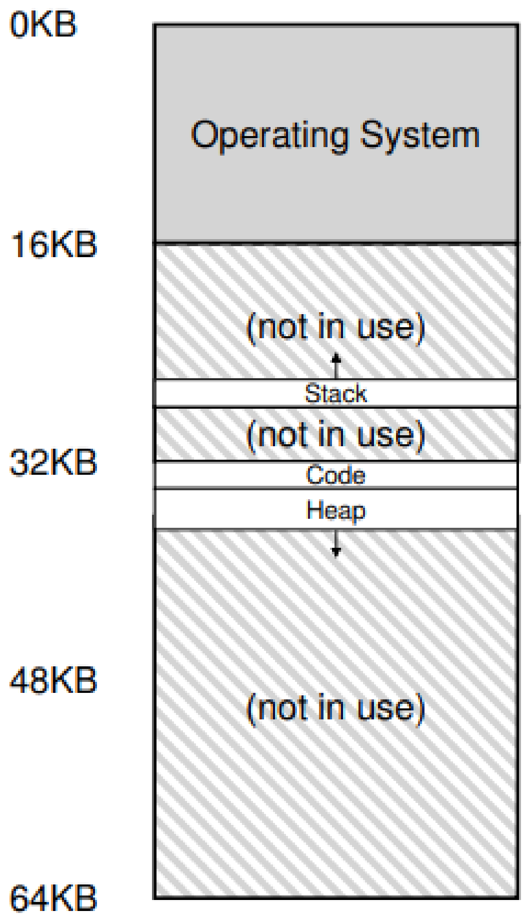


Figure 16.2: **Placing Segments In Physical Memory**

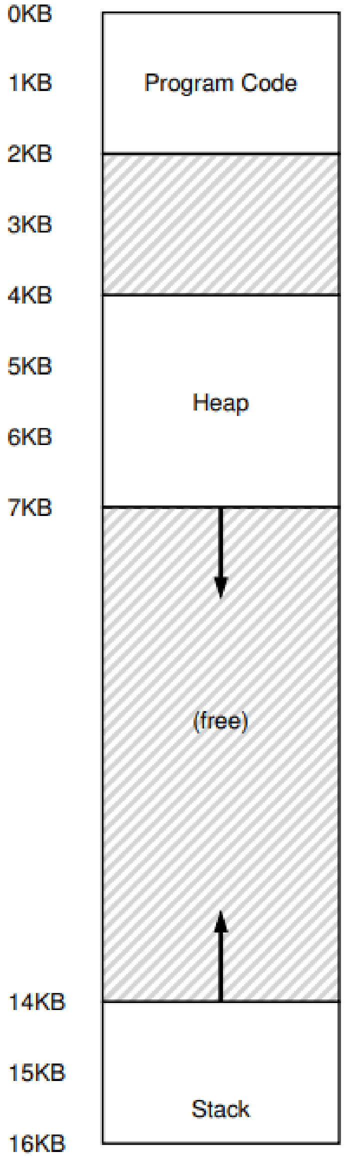


Figure 16.1: **An Address Space (Again)**

?

Support for Sharing

- System designers realized they could save space by sharing segments.
  - We still use this concept today with shared libraries.
- Need additional hardware support to mark segments with RWX privileges.

Segment	Base	Size (max 4K)	Grows Positive?	Protection
$Code_{00}$	32K	2K	1	Read-Execute
$Heap_{01}$	34K	3K	1	Read-Write
$Stack_{11}$	28K	2K	0	Read-Write

Figure 16.5: **Segment Register Values (with Protection)**

Fine-grained vs Coarse-grained Segmentation

- We've been talking about coarse-grained (large) segments.
- Some systems have used fine-grained (small) segments.
  - This requires a segment table in memory.
  - The idea was that the OS could learn which segments are in use and utilize main memory better.

OS Support

- OS must save segment registers on a context switch.
- OS must allow segments to grow or even shrink.
  - Calling malloc() might need to allocate more space in the heap segment.
- OS must manage free space in physical memory.
  - We end up with little holes of free space. Often, they're useless. This is called external fragmentation.

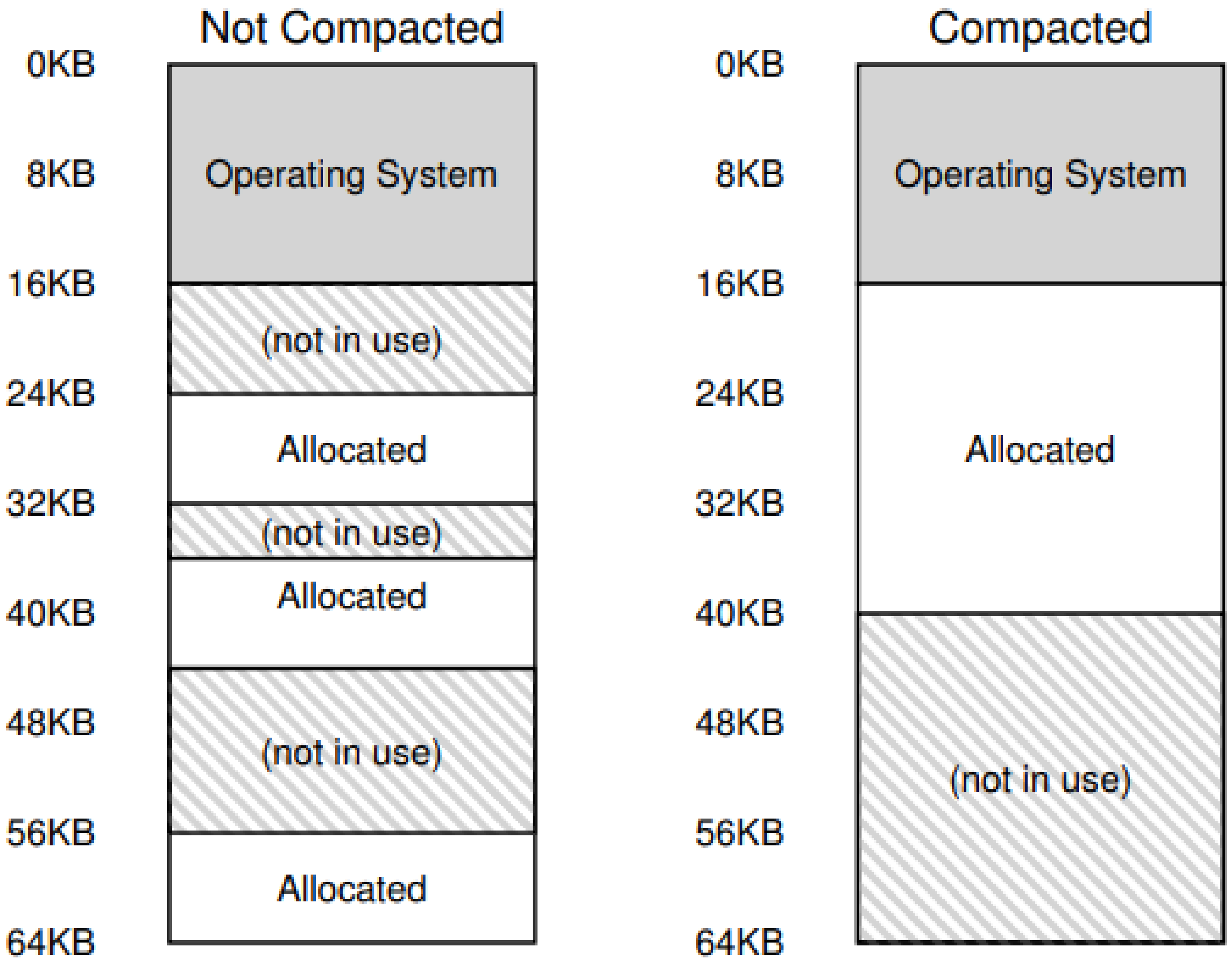


Figure 16.6: **Non-compacted and Compacted Memory**

Compact Physical Memory

- We could compact the physical memory, but this is expensive.
- Also makes requests to grow segments hard to serve.
- We could use various free-list management algorithms:
  - Best-fit
  - Worst-fit
  - First-fit
  - Buddy algorithm
- Unfortunately, we never eliminate external fragmentation.