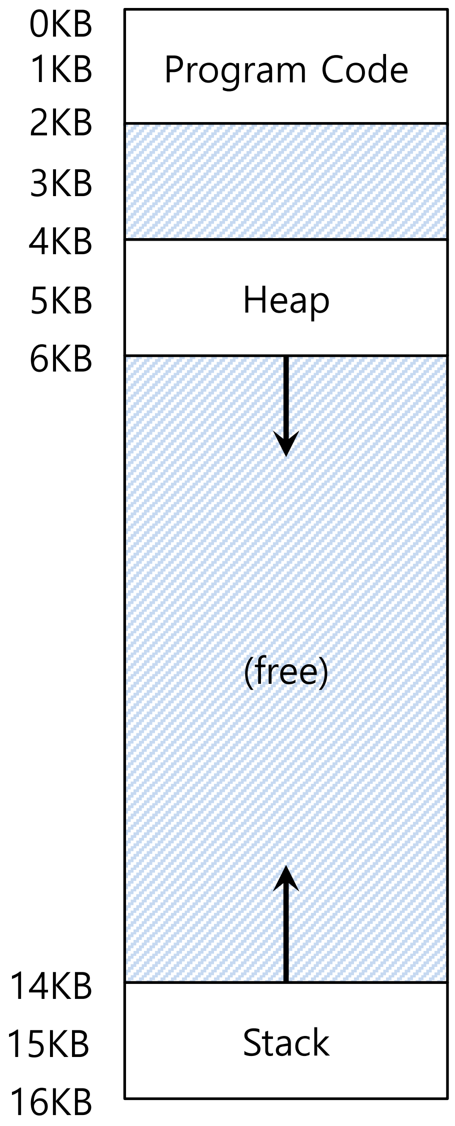
16. Segmentation

Operating System: Three Easy Pieces

Inefficiency of the Base and Bound Approach



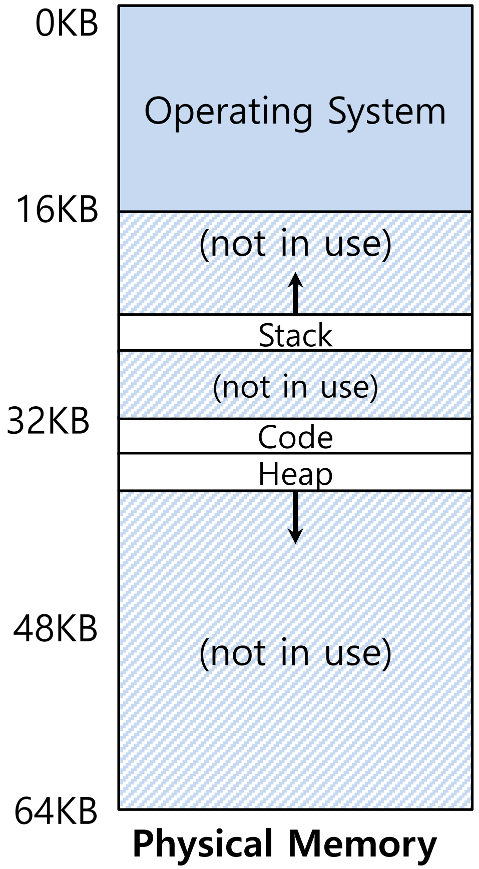
- **Big chunk of "free"** space
- "free" space **takes up** physical memory.
- **Hard** to run when an address space **does not fit** into physical memory

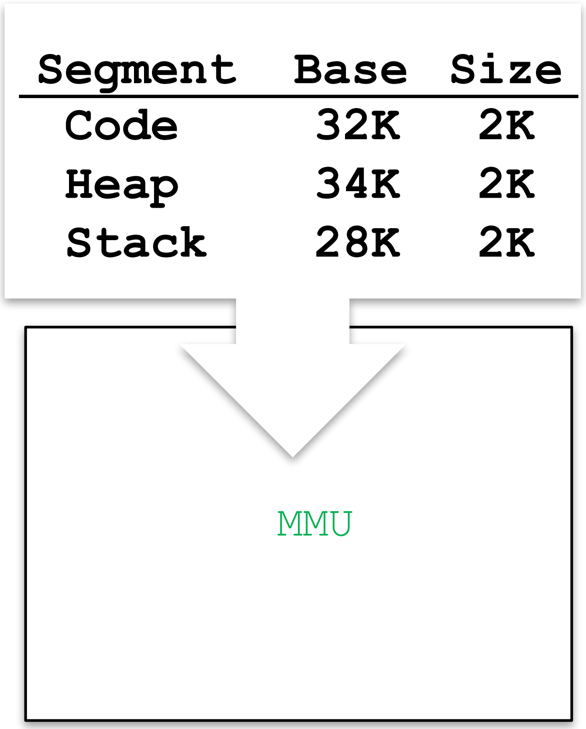
Segmentation

- Segment is just **a contiguous portion** of the address space of a particular length.
  - Logically-different segment: code, stack, heap

- Each segment can be **placed** in **different part of physical memory**.
  - **Base** and **bounds** exist **per each segment**.

Placing Segment In Physical Memory





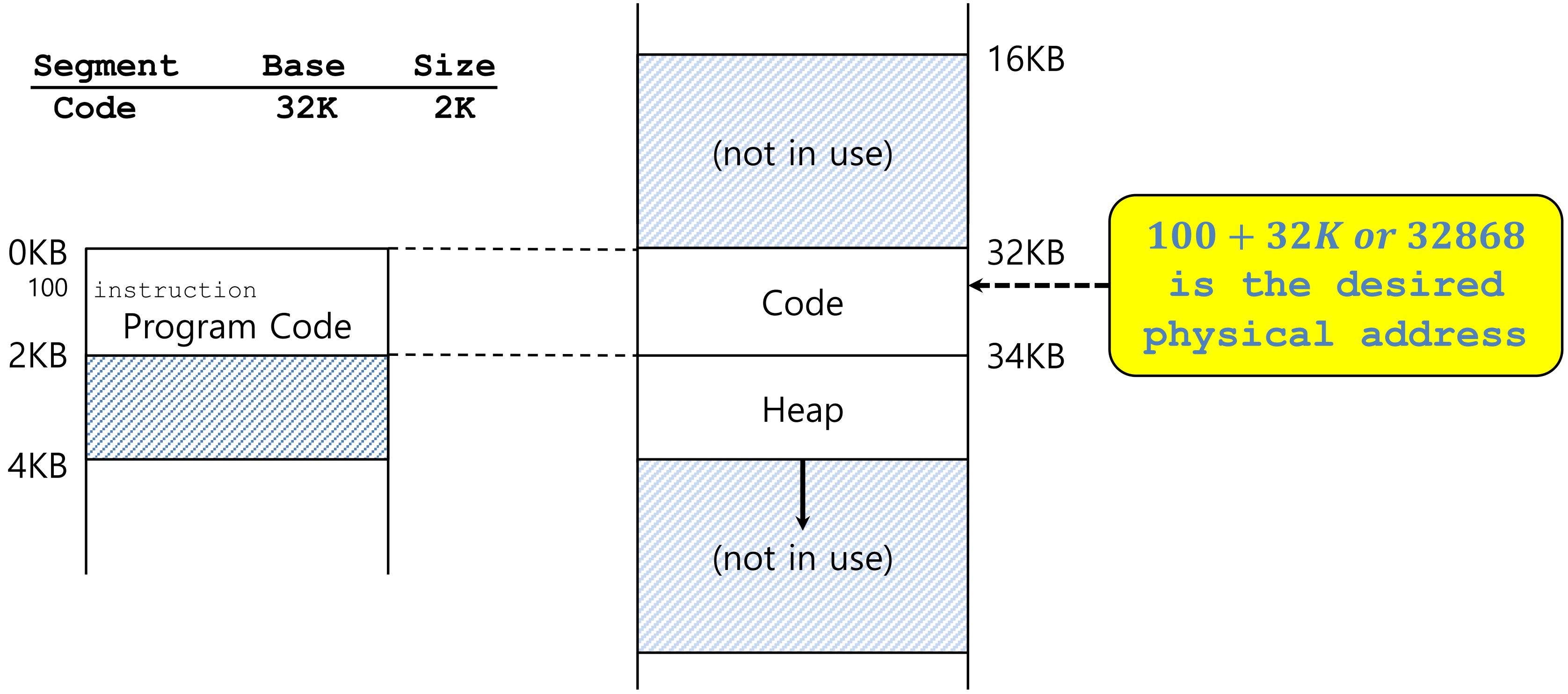
Segment	Base	Size
Code	32K	2K
Heap	34K	2K
Stack	28K	2K

Address Translation on Segmentation

$$physical\ address = offset + base$$

- The `offset` of virtual address 100 is `100`.
  - The code segment **starts at virtual address 0** in address space.

Segment	Base	Size
Code	32K	2K

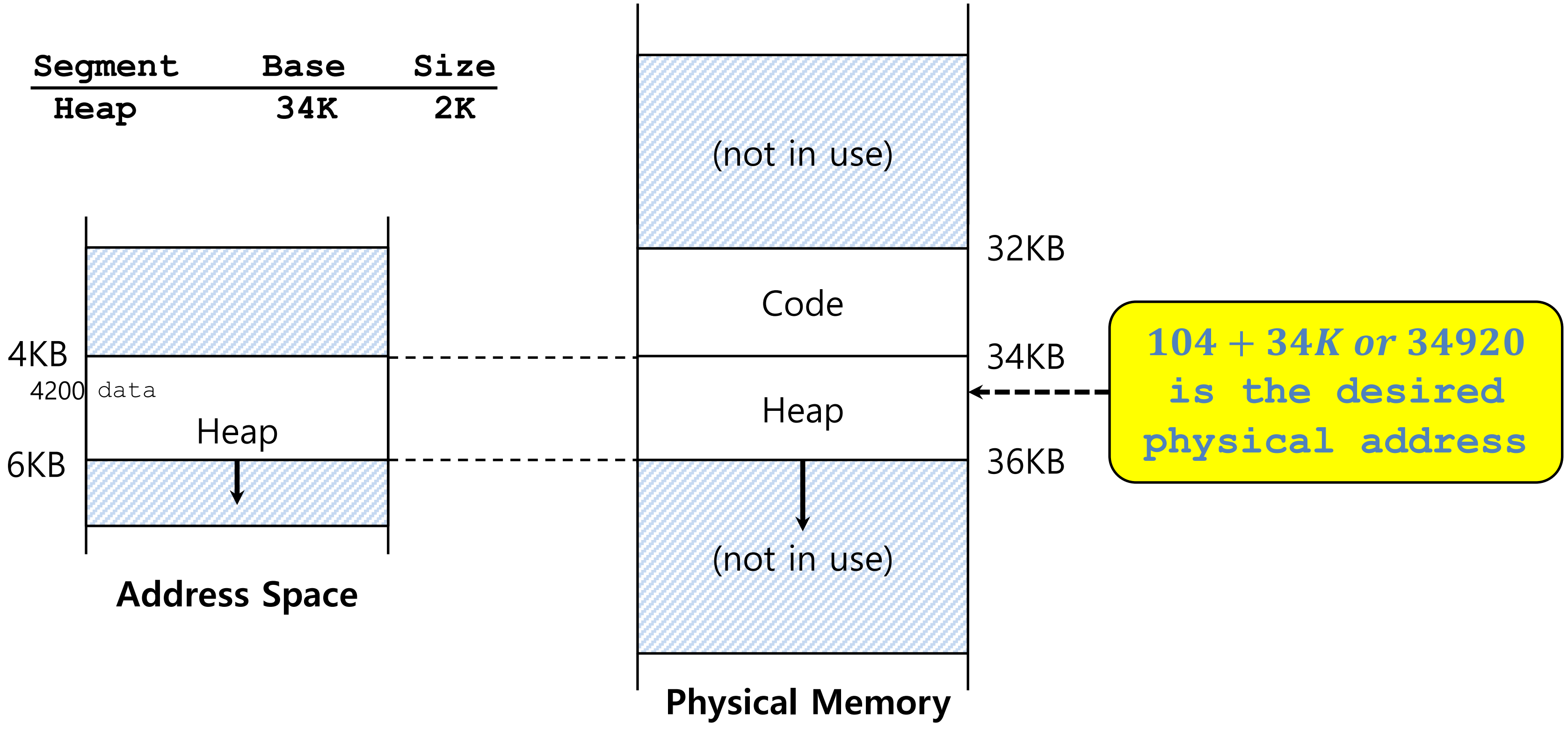


Address Translation on Segmentation(Cont.)

Virtual address + base **is not the correct physical address!**

- The `offset` of virtual address 4200 is `104`.
  - The heap segment **starts at virtual address 4096** in address space.

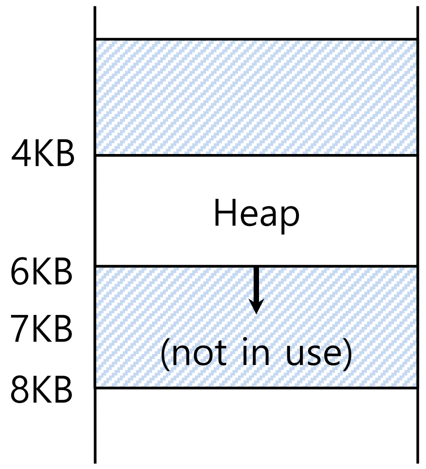
Segment	Base	Size
Heap	34K	2K



104 + 34K or 34920 is the desired physical address

Segmentation Fault or Violation

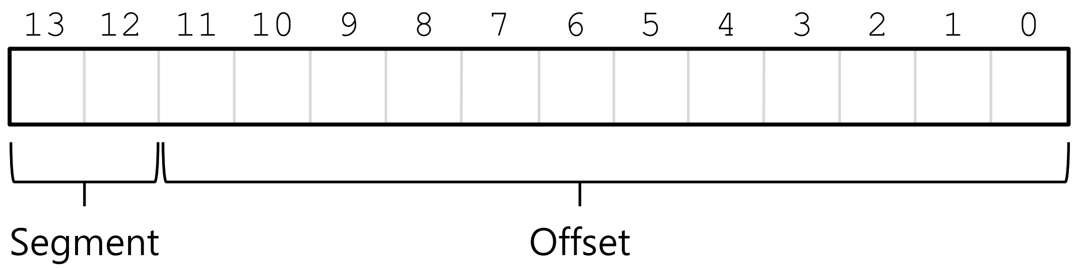
- If an **illegal address** such as 7KB which is beyond the end of heap is referenced, the OS occurs **segmentation fault**.
  - The hardware detects that address is **out of bounds**.



Address Space

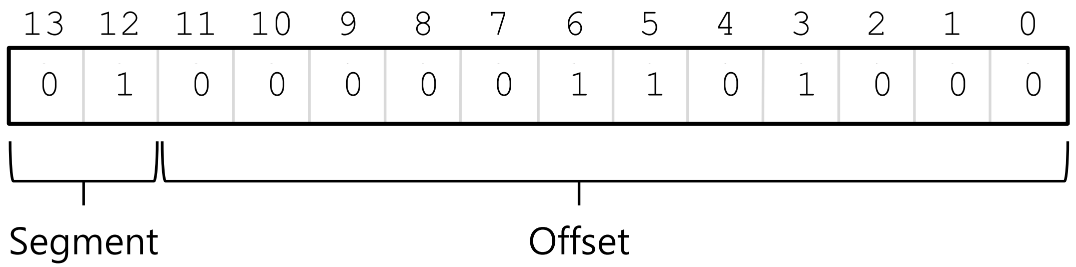
Referring to Segment

- **Explicit approach**
  - Chop up the address space into segments based on the **top few bits** of virtual address.



- Example: virtual address 4200 (01000001101000)

Segment	bits
Code	00
Heap	01
Stack	10
-	11



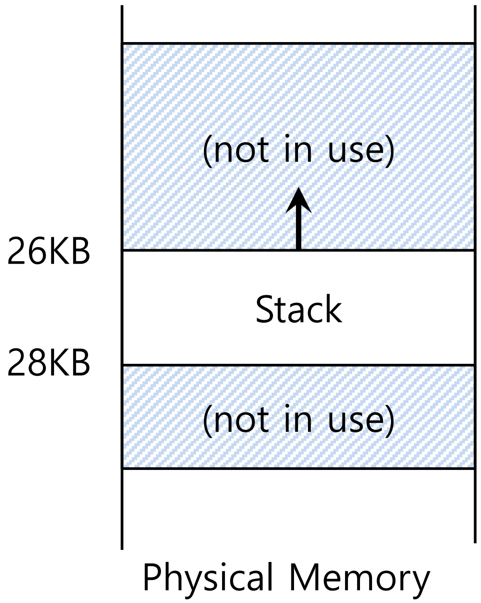
Referring to Segment(Cont.)

```
// get top 2 bits of 14-bit VA
Segment = (VirtualAddress & SEG_MASK) >> SEG_SHIFT
// now get offset
Offset = VirtualAddress & OFFSET_MASK
if (Offset >= Bounds[Segment])
     RaiseException(PROTECTION_FAULT)
else
     PhysAddr = Base[Segment] + Offset
     Register = AccessMemory(PhysAddr)
```

- SEG_MASK = 0x3000(11000000000000)
- SEG_SHIFT = 12
- OFFSET_MASK = 0xFFF (00111111111111)

Referring to Stack Segment

- Stack grows **backward**.
- **Extra hardware support** is need.
  - The hardware checks which way the segment grows.
  - 1: positive direction, 0: negative direction



Segment Register(with Negative-Growth Support)

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

Support for Sharing

- Segment can be **shared between address** space.
  - **Code sharing** is still in use in systems today.
  - by extra hardware support.
- Extra hardware support is need for form of **Protection bits.**
  - **A few more bits** per segment to indicate **permissions** of **read,** write and **execute**.

Segment Register Values(with Protection)

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

Fine-Grained and Coarse-Grained

- **Coarse-Grained** means segmentation in a small number.
  - e.g., code, heap, stack.
- **Fine-Grained** segmentation allows **more flexibility** for address space in some early system.
  - To support many segments, Hardware support with a **segment table** is required.

OS support: Fragmentation

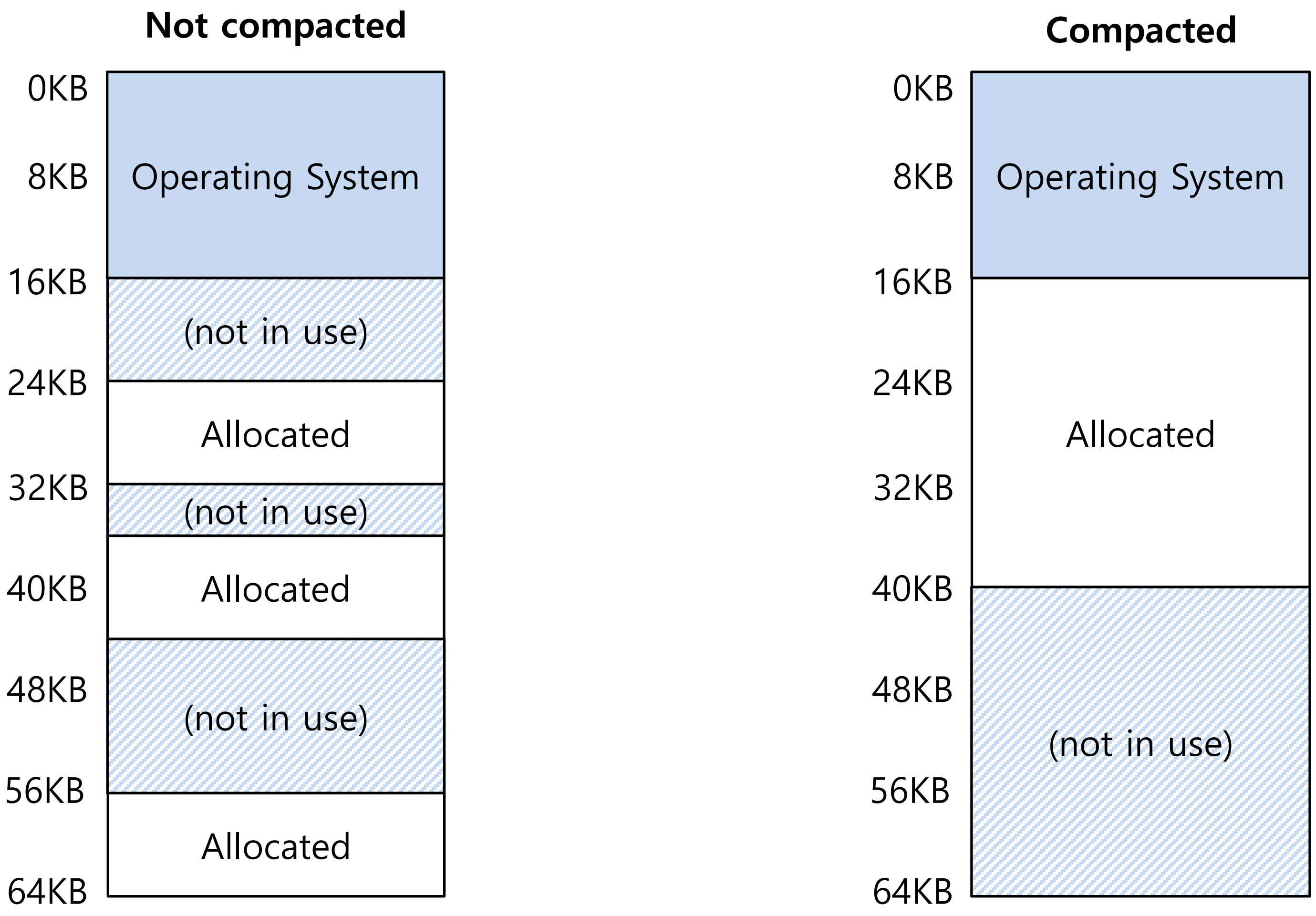
- **External Fragmentation**: little holes of **free space** in physical memory that make difficulty to allocate new segments.
  - There is **24KB free**, but **not in one contiguous** segment.
  - The OS **cannot** satisfy the **20KB request**.

- **Compaction**: **rearranging** the exiting segments in physical memory.
  - Compaction is **costly**.
    - **Stop** running process.
    - **Copy** data to somewhere.
    - **Change** segment register value.

- 1000 ways to solve it
  - None of them are the “best”

- *Added to creating, terminating, and context switches*

Memory Compaction



- Disclaimer: Disclaimer: This lecture slide set is used in AOS course at University of Cantabria. Was initially developed for Operating System course in Computer Science Dept. at Hanyang University. This lecture slide set is for OSTEP book written by Remzi and Andrea Arpaci-Dusseau (at University of Wisconsin)