# Segmentation

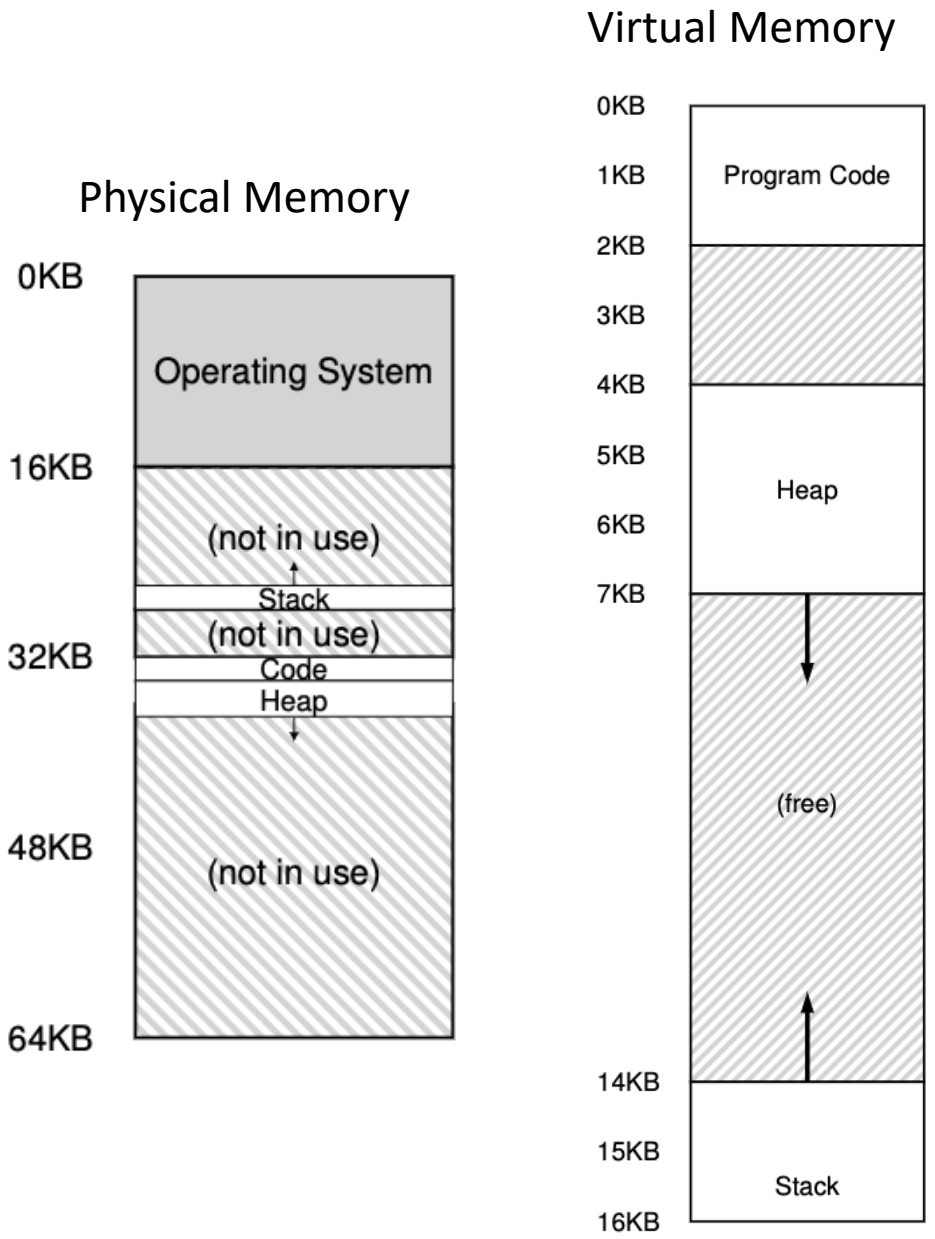
Chapter 16

# Previously in CS212…

- Discussed Address Translation for process relocation
  - Allows Virtual Address Space to be mapped to Physical Address Space
- The role of the OS and Hardware in process relocation
- Wondered if we could avoid internal fragmentation empty free space between stack and heap

# Segmentation

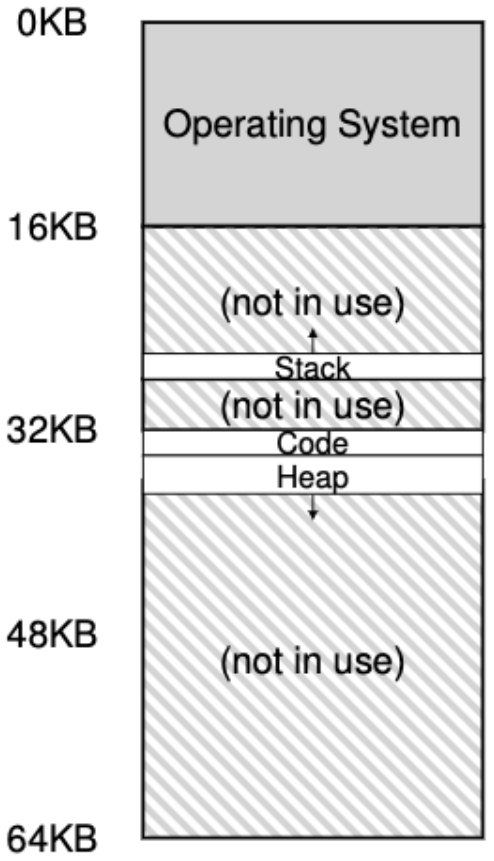
- Generalize the concept of the base and bound register
- Keep track of base and bounds of each segment of a process
  - Program code
  - Stack
  - Heap
- We are now free to place each segment anywhere in physical memory with varying sizes… AWESOME!

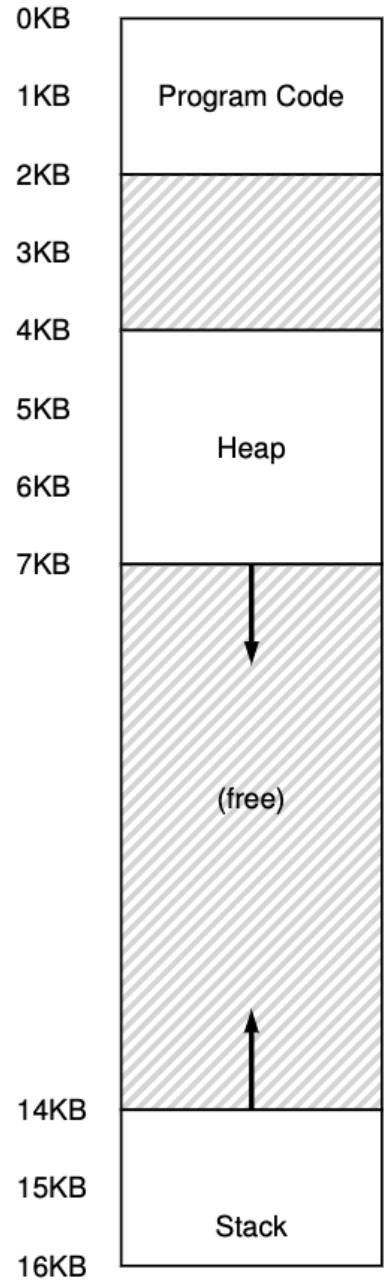


# Address Translation for Segments

- We want virtual address 100 (bytes)
  - Program code starts at 0, so it's 100 bytes away from the base 32K
    - 100 + 32,768 = 32,868 (100 < 2K safe!)
- We want virtual address 4200
  - Heap starts at 4,096 (4KB)
  - 34K + 4200?

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



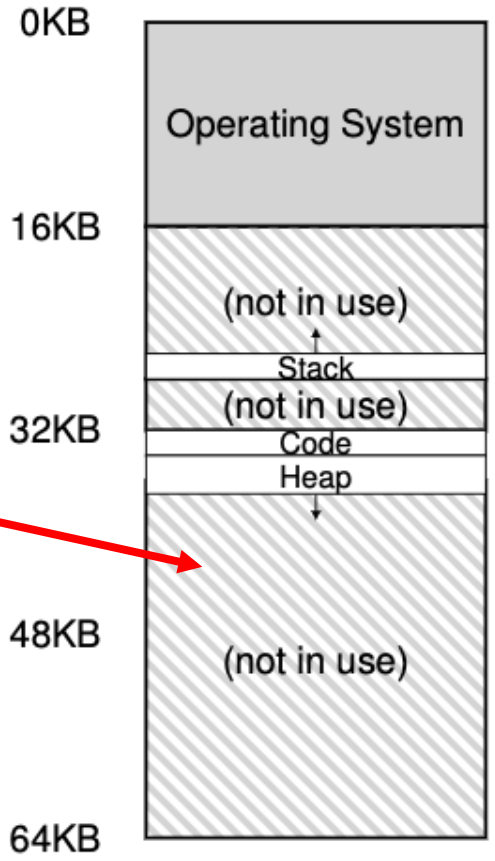
# Address Translation for Segments

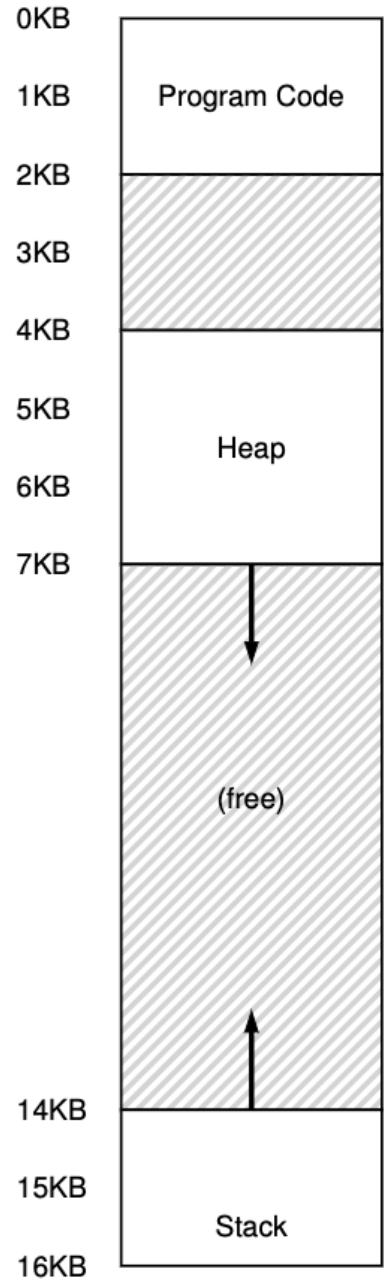
- We want virtual address 100 (bytes)
  - Program code starts at 0, so it's 100 bytes away from the base 32K
    - 100 + 32,768 = 32,868 (100 < 2K safe!)
- We want virtual address 4200
  - Heap starts at 4,096 (4KB)
  - 34K + 4200 = 39,016

**NOPE!**

**SEGMENTATION FAULT**

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

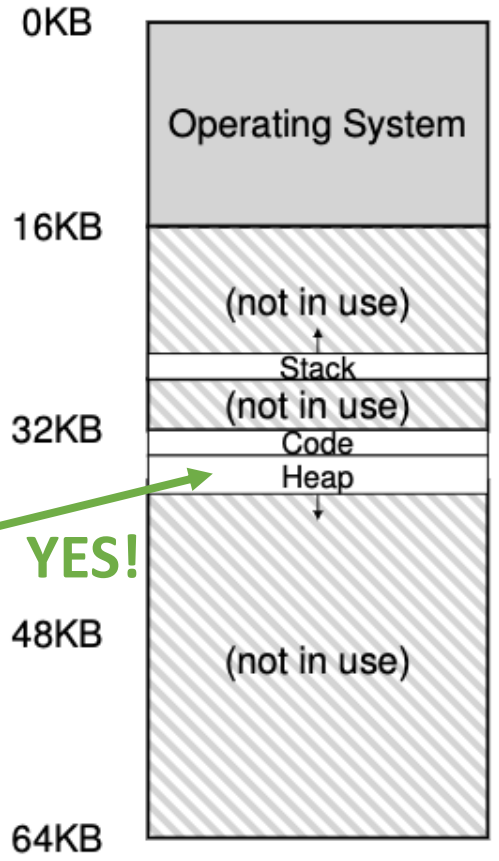


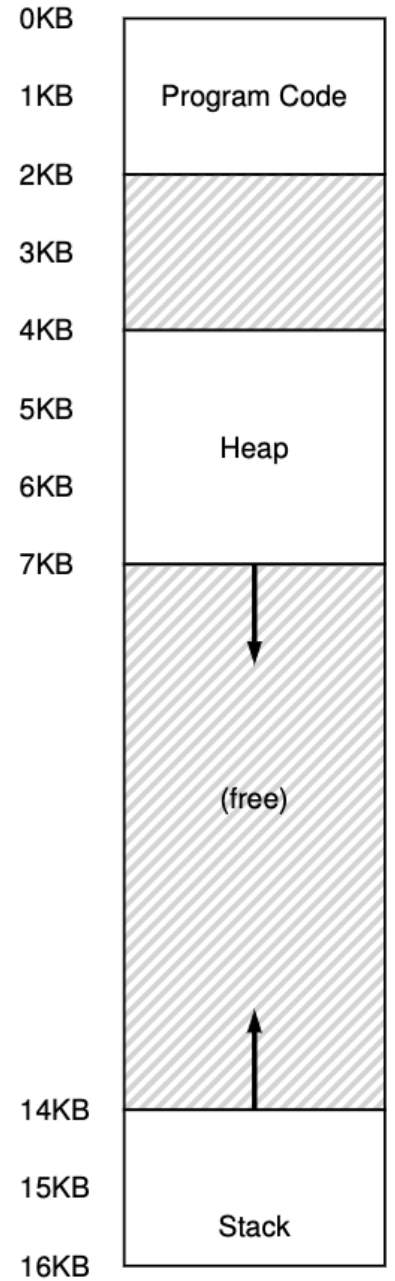



# Address Translation for Segments

- We want virtual address 100 (bytes)
  - Program code starts at 0, so it's 100 bytes away from the base 32K
    - 100 + 32,768 = 32,868 (100 < 2K safe!)
- We want virtual address 4200
  - Heap starts at 4,096 (4KB)
  - Need the virtual offset into the Heap first
    - 4200K – 4096 = 104-byte offset
  - Now use the offset 34,816 + 104 = 34,920

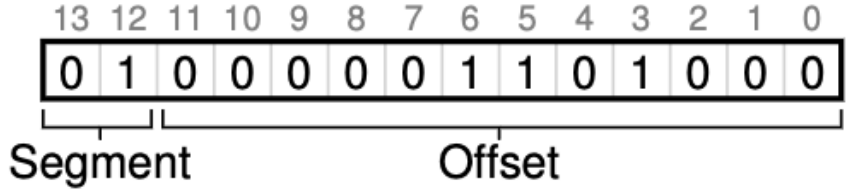
| Segment | Base | Size |
|---|---|---|
| Code | 32K | 2K |
| Heap | 34K | 3K |
| Stack | 28K | 2K |



# Identifying the Correct Segment

- Explicit
  - Supply bits along with the virtual address to determine which segment we are interested in
  - For our three-segment approach we need two bits
    - 00 – Program Code
    - 01 – Heap
    - 11 – Stack



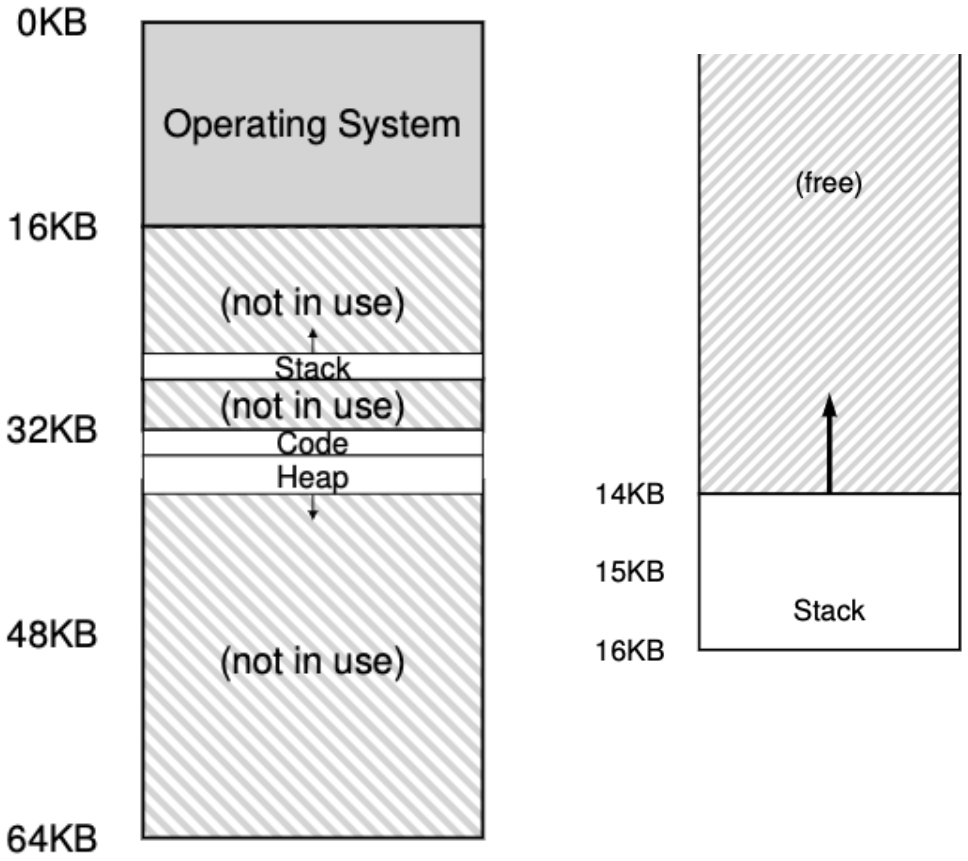

Offset 104 in the Heap segment

- Implicit
  - Hardware checks what data was used to produce the offset and deduces the segment
    - Program counter – code segment
    - Base pointer – stack segment
    - Other – heap segment

# Stack Segment Address Translation

- The stack is an odd case for the address translation as it ”grows” in a negative direction (toward lower addresses)
- Requires more hardware support to indicate positive growth with the offset
- Access stack at virtual address 15K
  - The stack can be a maximum of 4K which means at it’s largest, the stack can reach 12K virtual memory
    - 15K – 12K = 3K the offset
    - 4K – 3K is 1K backwards from the base
    - 28K – 1K = 27K

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



# Sharing Memory

- Wait…you said we didn't share memory between processes
  - True, I did say that, and we still support that abstraction the process may not realize this is happening

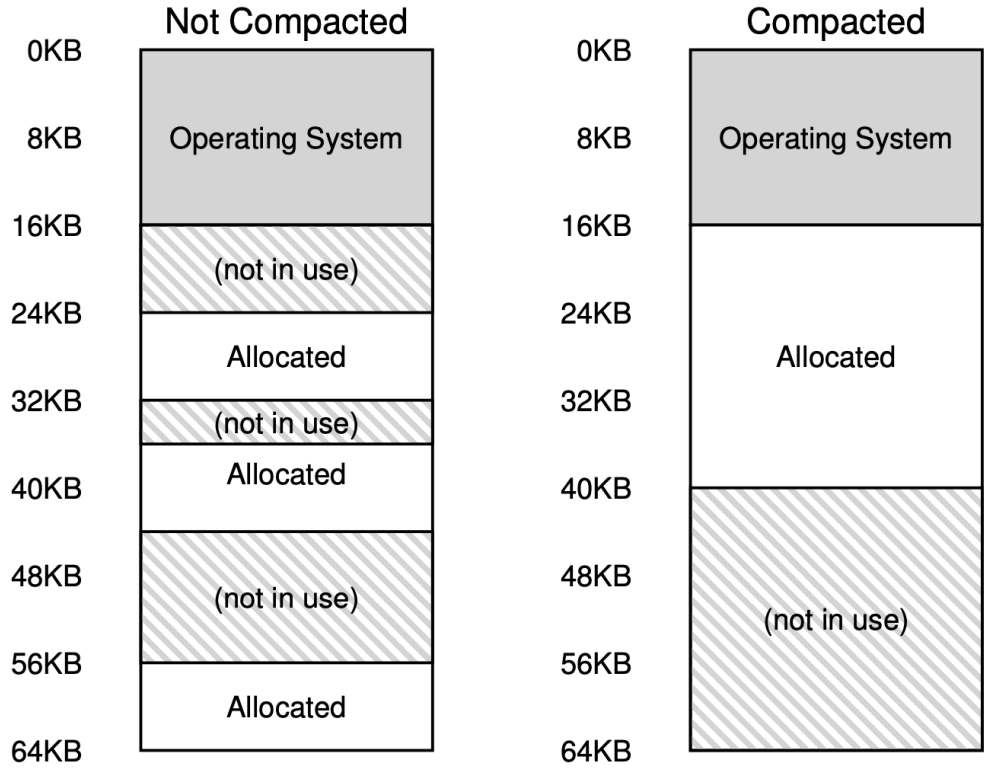
- Code sharing is something that is commonly still done
  - Can you think of a reason why?

- If we add a little extra hardware, we can keep track of permissions per segment

| 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 |

# Challenges

- Segments can be coarse (fewer large chunks) or fine grained (many smaller chunks)
- Segments helped with internal fragmentation, but cause external fragmentation
- Can compact the memory to better layout used memory
  - An expensive operation that can result in more compaction later as memory needs change



# Next Time…

- We solved internal fragmentation (space between stack and heap) but now we cause external fragmentation and waste space in between segments
- Compaction can help us to resolve wasted space, but it’s expensive
- While there isn’t a perfect solution, we need to find a compromise between efficient resource usage and performance