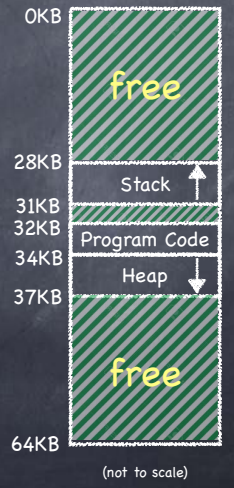
E Pluribus Unum

- An address space comprises multiple segments
  - contiguous sets of virtual addresses, logically connected
    - heap, code, stack, (and also globals, libraries…)
  - each segment can be of a different size

Segmentation: Generalizing Base & Bound

- Base & Bound registers to each segment
  - each segment is independently mapped to a set of contiguous addresses in physical memory
    - no need to map unused virtual addresses

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

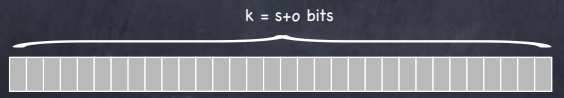


Segmentation

- Goal: Supporting large address spaces (while allowing multiple processes to coexist in memory)
- Needed hardware
  - two registers (Base and Bound) per segment
    - Stored in the PCB
  - a segment table, stored in memory, at an address point to by a Segment Table Register (STBR)
    - STBR stored in the PCB

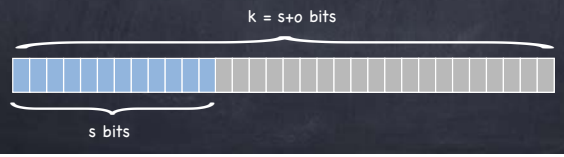
Segmentation: Mapping

- How do we map a virtual address to the appropriate segment?
  - Read VA as having two components
    - s most significant bits identify the segment
      - at most 2^s segments
    - o remaining bits identify offset within segment
      - each segment's size can be at most 2^o bytes



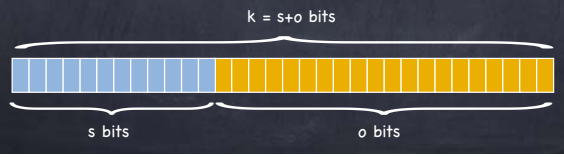
Segmentation: Mapping

- How do we map a virtual address to the appropriate segment?
  - Read VA as having two components
    - s most significant bits identify the segment
      - at most 2^s segments
    - o remaining bits identify offset within segment
      - each segment's size can be at most 2^o bytes



Segmentation: Mapping

- How do we map a virtual address to the appropriate segment?
  - Read VA as having two components
    - s most significant bits identify the segment
      - at most 2^s segments
    - o remaining bits identify offset within segment
      - each segment's size can be at most 2^o bytes



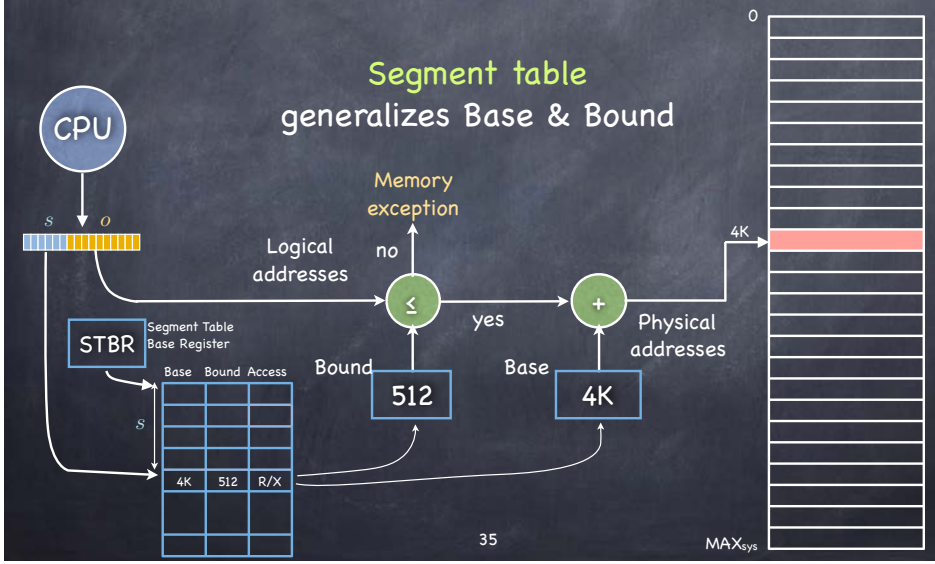
Segment Table

- Use s bits to index to the appropriate row of the segment table

	Base	Bound	Access
Code	32K	2K	Read/Execute
Heap	34K	3K	Read/Write
Stack	28K	3K	Read/Write

- Segments can be shared by different processes
  - use protection bits to determine if shared Read only (maintaining isolation) or Read/Write
    - processes can share code segment while keeping data private

Implementing Segmentation



Segments and Dynamically Allocated Memory

- Memory on heap and stack dynamically allocated
  - memory reallocated to a new process, must be zeroed (or else?)
  - zeroing memory is expensive; size of needed heap is unknown
- Zeroed on reference
  - bound register for heap segment starts at a few KB
  - zero only those first few KB
  - when program expands heap, exception!
    - zero out additional memory

Revisiting fork()

- Copying an entire address space can be costly…
  - especially if you proceed to obliterate it right away with exec()!

Revisiting fork(): Segments to the Rescue

- Instead of copying entire address space, copy just segment table (the VA->PA mapping)

	Base	Bound	Access
Code	32K	2K	RX
Heap	34K	3K	RW
Stack	28K	3K	RW

Parent

	Base	Bound	Access
Code	32K	2K	RX
Heap	34K	3K	RW
Stack	28K	3K	RW

Child

- but change all writeable segments to read only

Revisiting fork(): Segments to the Rescue

- Instead of copying entire address space, copy just segment table (the VA->PA mapping)

	Base	Bound	Access
Code	32K	2K	RX
Heap	34K	3K	R
Stack	28K	3K	R

Parent

	Base	Bound	Access
Code	32K	2K	RX
Heap	34K	3K	R
Stack	28K	3K	R

Child

- but change all writeable segments to read only
- Segments in VA spaces of parent and child point to same locations in physical memory

Copy on Write (COW)

- When trying to modify an address in a read-only segment:
  - exception!
    - exception handler copies just the affected segment, and changes both the old and new segment to writeable
- If exec() is immediately called, only stack segment is copied!

Managing Free space

- Many segments, different processes, different sizes
- OS tracks free memory blocks ("holes")
  - Initially, one big hole
- Many strategies to fit segment into free memory (think "assigning classrooms to courses")
  - First Fit: first big-enough hole
  - Next Fit: Like First Fit, but starting from where you left off
  - Best Fit: smallest big-enough hole
  - Worst Fit: largest big-enough hole

External Fragmentation

- Over time, memory can become full of small holes
  - Hard to fit more segments
  - Hard to expand existing ones
- Compaction
  - Relocate segments to coalesce holes

External Fragmentation

- Over time, memory can become full of small holes
  - Hard to fit more segments
  - Hard to expand existing ones
- Compaction
  - Relocate segments to coalesce holes

External Fragmentation

- Over time, memory can become full of small holes
  - Hard to fit more segments
  - Hard to expand existing ones
- Compaction
  - Relocate segments to coalesce holes
    - Copying eats up a lot of CPU time!
      - if 4 bytes in 10ns, 8 GB in 20s!
- But what if a segment wants to grow?

Eliminating External Fragmentation: Swapping

- Preempt processes and reclaim their memory
- Move images of suspended processes to backing store