Segmentation

- Paging
  - mitigates various memory allocation complexities (e.g., fragmentation)
  - view an address space as a linear array of bytes
  - divide it into pages of equal size (e.g., 4KB)
  - use a page table to map virtual pages to physical page frames
    - page (logical) => page frame (physical)
- Segmentation
  - partition an address space into logical units
    - stack, code, heap, subroutines, ...
  - a virtual address is <segment #, offset>

What’s the point?

- More “logical”
  - a linker takes a bunch of independent modules that call each other and linearizes them
  - they are independent; treat them as such
- Facilitates sharing and reuse
  - a segment is a natural unit of sharing – a subroutine or function
- A natural extension of variable-sized partitions
  - variable-sized partition = 1 segment/process
  - segmentation = many segments/process

Hardware support

- Segment table
  - multiple base/limit pairs, one per segment
  - segments named by segment #, used as index into table
    - a virtual address is <segment #, offset>
  - offset of virtual address added to base address of segment to yield physical address

Segment lookups

Pros and cons

- Yes, it’s “logical” and it facilitates sharing and reuse
- But it has all the horror of a variable partition system
  - except that linking is simpler, and the “chunks” that must be allocated are smaller than a “typical” linear address space
- What to do?
Combining segmentation and paging

- Can combine these techniques
  - x86 architecture supports both segments and paging
- Use segments to manage logical units
  - segments vary in size, but are typically large (multiple pages)
- Use pages to partition segments into fixed-size chunks
  - each segment has its own page table
    - there is a page table per segment, rather than per user address space
    - memory allocation becomes easy once again
      - no contiguous allocation, no external fragmentation

<table>
<thead>
<tr>
<th>Segment #</th>
<th>Page #</th>
<th>Offset within page</th>
<th>Offset within segment</th>
</tr>
</thead>
</table>

- Linux:
  - 1 kernel code segment, 1 kernel data segment
  - 1 user code segment, 1 user data segment
  - N task state segments (stores registers on context switch)
  - 1 "local descriptor table" segment (not really used)
  - all of these segments are paged
- Note: this is a very limited/boring use of segments!