Memory Management Basics
Are We Done?

- Segmentation allows sharing
- ...but allocation/deallocation of arbitrary size segments is complex
  - Yields external fragmentation
- How can we improve memory management?
- Solution: Paged segmentation!
  - Partition segments into fixed-size pages
  - Allocate and deallocate pages
  - Contiguous pages in VAS need not be contiguous in PAS
  - No external fragmentation, but internal fragmentation possible

Implementing Paged Segmentation
Combining segmentation and paging

- Individual segments can be implemented as a paged, virtual address space
  - A segment address is a pair \((s, va)\) or a triple \((s, p, o)\)
    - \(s\) — segment number
    - \(p\) — page number \((p_{\text{max}}\) pages\)
    - \(o\) — page offset \((o_{\text{max}}\) bytes/pages\)

Virtual address = \((s \times p_{\text{max}} + p) \times o_{\text{max}} + o\)

Implementing Paged Segmentation
Sharing in Paged-Segmented Systems

- Add one additional level of indirection to the page table
- If segments are paged then page tables are automatically shared
  - Processes need only agree on a number for the shared segment

Shared Segment Page Table
Physical Memory
Virtual Memory and Address Translation

Virtual Memory Concept

- Key problem: How can one support programs that require more memory than is physically available?
- Hide all physical aspects of memory from users
  - Memory is a logically unbounded virtual address space of \( 2^n \) bytes
  - Only portions of VAS are in physical memory at any one time

Issues

- Placement strategies
  - Where to place programs in physical memory
- Replacement strategies
  - What to do when there exist more processes than can fit in memory
- Load control strategies
  - Determining how many processes can be in memory at one time

Realizing Virtual Memory

Paging

- Physical memory partitioned into equal sized page frames

A memory address is a pair \((f, o)\)

\[ f \text{ — frame number (} f_{\text{max}} \text{ frames)} \]
\[ o \text{ — frame offset (} o_{\text{max}} \text{ bytes/frames)} \]

Physical address = \( o_{\text{max}} \times f + o \)

Physical Address Specifications

Frame/Offset Pair v. An absolute index

Example: A 16-bit address space with \( o_{\text{max}} = 512 \) byte page frames

- Addressing location \((3, 6) = 1,542\)
Realizing Virtual Memory

Paging

A process's virtual address space is partitioned into equal sized pages

- Page = page frame

A virtual address is a pair \((p, o)\)

- \(p\) — page number (\(p_{\text{max}}\) pages)
- \(o\) — page offset (\(o_{\text{max}}\) bytes/pages)

Virtual address = \(o_{\text{max}} \times p + o\)

Paging Mapping virtual addresses to physical addresses

- Pages map to frames
- Pages are contiguous in a VAS...
- But pages are arbitrarily located in physical memory, and
- Not all pages mapped at all times

Virtual Address Translation Details

Page Table structure

- Contents:
  - Flags — dirty bit, resident bit, clock/reference bit
  - Frame number

1 table per process
Part of process's state

- A page table maps virtual pages to physical frames
Virtual Address Translation Details

Example

A system with 16-bit addresses
- 32 KB of physical memory
- 1024 byte pages

CPU

Page Table

Physical Addresses

Virtual Addresses

P

(4, 1023)

(4, 0)

(3, 1023)

Virtual Address Translation

Performance Issues

- Problem — VM reference requires 2 memory references!
  - One access to get the page table entry
  - One access to get the data
- Page table can be very large; a part of the page table can be on disk!
  - For a machine with 64-bit addresses and 1024 byte pages, what is the size of a page table?
- What to do?
  - Most computing problems are solved by some form of...
    - Caching
    - Indirection