Segments make both sharing and protection easier. Why?

Why?

Segments make it easier for the call stack and heap to grow dynamically.

offsets.

Thus processes thus use virtual addresses that are segments and segment offsets.

The compiler generates references that identify the segment and the offset in the segment.

user views the program in logical segments, e.g., code, global variables, stack, heap (dynamic data structures), not a single linear array of bytes.

Segments take the user's view of the program and gives it to the OS.

Today: Segmentation

Hardware maps from virtual addresses to physical addresses.

OS divides the process onto pages, manages a page table for every process; and manages the pages in memory.

Last Class: Paging
memory utilization we get from pages. Let's combine the ease of sharing we get from segments with efficient

Similar memory mapping algorithm as paging. We need something like

- External fragmentation can be a problem again
- Each segment is allocated a contiguous piece of physical memory.

Segmentation can be combined with a dynamic or static relocation

Segment number.

Compiler needs to generate virtual addresses whose upper order bits are a

Hardware support: multiple base/limit registers.

- modeling, etc.

Segment and protection information (can this segment be shared, read,

Segment table: each entry contains a base address in memory, length of

Implementing Segmentation

Implementing Segmentation
Combining Segments and Paging

Map a logical segment onto multiple page frames by paging the segments.

Segments are typically larger than page frames.

Treat physical memory as a sequence of fixed size page frames.

Treat virtual address space as a collection of segments (logical units) of arbitrary sizes.
Addresses in a Segmented Paging System

- Add the frame and the offset to get the physical address.

  (The rest of this is just like paging.)
  - Use the page number to index the page table. The entry in the frame,
    - Check the remainder of the address (page number and offset) against the
      segment limit of the segment.
    - base address of the page table for that segment.
      - The segment number indexes into the segment table which yields the
        segment, and an offset within the page.
        - A virtual address becomes a segment number, a page within that
          segment, and an offset within the page.

Addresses in a Segmented Paging System
When would segments containing data be shared?

- Need protection bits to specify and enforce read/write permission.

- Sharing the page table for that segment

- Share whole segments by sharing segment table entries, which is the same

- Share individual pages by copying page table entries.

Sharing Pages and Segments

3. How many segment table entries do we need?

2. How many bits is a virtual address?

1. How many bits is a physical address?

8 logical segments

- a page size of 32 words, and

- a page table indexing 8 pages,

- Given a memory size of 256 addressable words,

Addresses in a Segmented Paging System: Example
 allocated? With pure paging?

- How does fragmentation of segmented paging compare with continuous
  paging compared to continuous allocation with relocation? Compared to pure
  segmented paging (virtual address space/page size) entries in page
  table. How many entries in a segmented paging system?
  - Costs: somewhat slower context switches, slower address translation,
  - Benefits: faster process start times, faster process growth, memory
  sharing between processes.

### Segmented Paging: Costs and Benefits

- Segmentation when the tables get too big.
  
  **Note:** Just like recursion, we can do multiple levels of paging and
  segmentation when the tables get too big.

- Protection and valid bits can go either on the segment or the page table

  (slower but no restrictions on the number of segments per program)

  index and page index combined used in the TLB lookup

  - Both the segment tables and page tables can be in main memory with the segment
    (faster but limits the number of segments a program can have)

  - Store segment tables in a small number of associated registers; page tables are in
    main memory with a TLB

  - Where are the segment table and page tables stored?

### Sharing Pages and Segments: Implementation Issues
Segmentation vs. Paging

- Two lookups per memory reference
- Increased internal fragmentation over paging
- Sharing at either the page or segment level
- Easy memory allocation, any frame can be used
  (large virtual address spaces are not a problem)
- Only need to allocate as many page table entries as we need (large virtual address spaces).

Paging:

- Put together

Segmentation:

- Simple, but inflexible
- Reclamation using base and limit registers

Putting it all together