Segments make both sharing and protection easier. Why?

Why?

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

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

Thus, in the segment, e.g., a code segment with offset = 399 offsets in the segment, e.g., the compiler generates references that identify the segment and the stack, heap (dynamic data structures), not a single linear array of bytes.

User views the program in logical segments, e.g., code, global variables, 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.

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

Process generates virtual addresses from 0 to Max.

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

memory utilization we get from pages.

Similar memory mapping algorithms as paging. We need something like:

- External segmentation 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

Implementing Segmentation

Hardware support: multiple base/limit registers.

modifed, etc).

Segment, and protection information (can this segment be shared, read, or
Segment table: each entry contains a base address in memory, length of

Implementing Segmentation
Combining Segments and Paging

- Map a logical segment onto multiple page frames by pagining 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.

Combining Segments and Paging
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 is the frame.

- Check the remainder of the address (page number and offset) against the limit of the segment.

- Get the 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.
- When would segments containing data be shared?
- When would segments containing code 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 2^36 addressable words,

Addresses in a Segmented Paging System: Example
How does fragmentation of segmented paging compare with contiguous

Page？

compared to contiguous allocation with relocation？ Compared to pure

table. How many entries in a segmented paging system？

Pure paging system (virtual address space/pagetable entries in page

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

Note: Just like recursion, we can do multiple levels of paging and

Table entries

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

2. 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)

main memory with a TLB

1. Store segment tables in a small number of associative registers: page tables are in

Where are the segment table and page tables stored?

Sharing Pages and Segments: Implementation Issues
Segmentation & Paging

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

Put It All Together

Segmentation

- Compaction is needed to resolve external fragmentation
- Memory allocation is expensive and complicated (first fit, worst fit, best fit)
- Segment tables tend to be small
- Compiler’s view presented to OS

Relocation using Base and Limit Registers

Put It All Together