Segments make both sharing and protection easier. Why?

Why?

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

- This process thus use virtual addresses that are segments and segment offsets.
- The compiler generates references that identify the segment and the offset in the segment, e.g., a code segment with offset = 399.
- The heap view the program in logical segments, e.g., code, global variables, stack, heap (dynamic data structures), not a single linear array of bytes.
- User views the program and gives it to the OS. 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 onto pages, manages a page table for every process.
- Generates virtual addresses from 0 to Max.
Let’s combine the ease of sharing we get from segments with efficient
memory utilization we get from pages.

Similar memory mapping algorithm 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

Hardware support: multiple base/limit registers.

modeled, etc).

Segment and protection information (can this segment be shared, read, write, etc)
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 packing 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) or

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

- Sized sharing the page table for each 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
How does fragmentation of segmented paging compare with continuous allocation when pure paging?

Compared to continuous allocation, segmented allocation can reduce fragmentation by using smaller pages.

What is the performance of address translation of segmented paging?

Table: How many entries in a segmented paging system?

Pure paging system: (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 protection and valid bits can go either on the segment or the page table.

1. Store segment tables in a small number of associative registers; page tables are in main memory with a TLB.
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).
3. Segment table and page table stored.

Sharing Pages and Segments: Implementation Issues
- 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
- Pages are not a problem.
- Only need to allocate as many page table entries as we need (large virtual address)

Segmentation & Paging

- Pages (especially when virtual address space is large and pages large)
- Page tables can be very large
- Simpler memory allocation since any page can be allocated to any frame

Putting it all together