Memory Management - Segmentation

CS350
Kartik Gopalan

Chapter 3 Tanenbaum’s book
Separate Instruction and Data Spaces

- One address space
- Separate I and D spaces
Example: Compiler program without segmentation

- One-dimensional address space with growing tables
Example: Compiler program with segmentation

- One process has multiple address spaces.
- Each address space grows or shrinks, independently
## Comparison of paging and segmentation

<table>
<thead>
<tr>
<th>Consideration</th>
<th>Paging</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need the programmer be aware that this technique is being used?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>How many linear address spaces are there?</td>
<td>1</td>
<td>Many</td>
</tr>
<tr>
<td>Can the total address space exceed the size of physical memory?</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can procedures and data be distinguished and separately protected?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can tables whose size fluctuates be accommodated easily?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Is sharing of procedures between users facilitated?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Why was this technique invented?</td>
<td>To get a large linear address space without having to buy more physical memory</td>
<td>To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection</td>
</tr>
</tbody>
</table>
Implementation of Pure Segmentation

(a)-(d) Development of checkerboarding (external fragmentation)
(e) Removal of the checkerboarding by compaction
MULTICS — Paged Segmentation

- Every process can have multiple virtual address spaces (or segments)

- Each Segment has its own page table

- Advantage
  - Each segment can have the full virtual address space allowed by number of address bits

- Disadvantage
  - Switching from one segment to another has a high context switch penalty, even within the same process.
Translation of a Multics Virtual Address
Segmentation with Paging: MULTICS

- Segment descriptor table has one descriptor for each segment
- Each segment descriptor points to a page table
Pentium — Paged Segmentation

• Each process can have multiple segments
• Multiple segments map to one linear address space
• Linear address space has one page table
Translation of a Pentium Virtual Address

Step 1: Convert (selector, offset) pair to a linear address using segment descriptor

Step 2: Convert linear address onto a physical address using page table entry
Segmentation with Paging: Pentium

Similar to MULTICS, but addresses a number of different design goals

A Pentium selector
(equivalent to Segment Number in Multics)
GDT = Global Descriptor Table
LDT = Local Descriptor Table

Four privilege levels in x86

- Code segment descriptor
- Data segment descriptors are slightly different
References

- Chapter 3: Modern Operating Systems, Andrew S. Tanenbaum

- Segmentation
  - http://en.wikipedia.org/wiki/Memory_segment

- x86

- Intel Memory model