Lecture 18: Segmentation II (Apr 7, 2005) Yap

Apr 7, 2005

1 ADMIN

- We continue discussion of segmentation.

2 Review

- Q: If you are allowed only 2 segments, how would you use your 2 segments and set their permissions?
  A: One segment for (program) text, which is executable but read-only, another segment for data, which is read and write, but not executable.
- Q: Why do we need TLB’s?
  A: To speed up the memory reference in the presence of paging and segmentation.
- Q: Name 4 advantages of segmentation
  A: 1. Facilitates sharing of library and data across processes
      2. Simply the handling of data structures that grow/shrink independently
      3. Supports the management of different parts of the data/code with different permission properties.
      4. Supports the separate compilation and linking of modules.

3 Review of Segmentation

- Segmentation and paging are similar and yet different.
  - The 2 techniques are normally used simultaneously.
  - Many variations of both techniques exist
- Here is a comparison (cf. Figure 4-37 of Tanenbaum):
  For the purposes of this comparison, we assume ”pure segmentation” which is not combined with paging.
<table>
<thead>
<tr>
<th>Property</th>
<th>Paging</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer aware of technique?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>How many linear address spaces?</td>
<td>1</td>
<td>Many</td>
</tr>
<tr>
<td>Can address space exceed physical memory?</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Separate protection of procedure and/or data?</td>
<td>No(Maybe)</td>
<td>Yes</td>
</tr>
<tr>
<td>Sharing of procedures/data across processes?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can table sizes that changes unpredictably be handled?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Avoids external fragmentation?</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

4 Segmentation in Intel Pentiums

- 16K segments
- Each segment up to 1 Gigawords (32 bits)
- 2 tables: LDT (Local Descriptor Table) and GDT (Global Descriptor Table)
- Each process has a LDT, but there is only one GDT
- LDT: describes segments for each process
- GDT: describes segments for the OS

- **Segment Selectors**: 16-bit word, (Index, GDT-or-LDT-bit, 4-Privilege-bits)
  - The index gives the entry into GDT or LDT
- Pentium has 6 ”segment registers”
- To access a segment, first load a selector for that segment into one of segment registers
- This will fetch the corresponding **segment descriptor** from LDT/GDT into a 64-bit ”microprogram register”.
- Using this segment descriptor, we can check validity of the offset and whether the segment is loaded.
- If so, we form the **linear address** from the segment descriptor and offset.
  - Linear address = (PageDir#, Page#, Offset)
- To speed up, we keep a small TLB to map most recent (PageDir#, Page#) into page frame#.
• PROTECTION: Levels 0 to 3 (kernel, system call, shared library, user progs).
  – Thus 2 bits in the PSW stores this info
  – Each segment also has this protection level info
  – A program trying to access a segment at DIFFERENT level will cause a trap
  – But controlled way to access different levels is possible (use selectors)