Two-Level Scheme for 32-bit Address

Figure 8.4 A Two-Level Hierarchical Page Table
Page Tables

- The entire page table may take up too much main memory
- Page tables are also stored in virtual memory
- When a process is running, part of its page table is in main memory
Inverted Page Table

- Alternative to (multi-level) page table
  - Used on PowerPC, UltraSPARC, and IA-64 architecture
  - One entry in table for each physical memory frame
  - Page number portion of a virtual address is mapped to a hash value
  - Fixed proportion of real memory is required for the tables regardless of the number of processes
Inverted Page Table

- Page table entries:
  - Page number
  - Process identifier
  - Control bits
  - Chain pointer
Note that $n > m$

Figure 8.6  Inverted Page Table Structure
Translation Lookaside Buffer

• Each virtual memory reference can cause two physical memory accesses
  – One to fetch the page table
  – One to fetch the data

• To overcome this problem a high-speed cache is set up for page table entries
  – Called Translation Lookaside Buffer (TLB)
Translation Lookaside Buffer

- Contains page table entries that have been most recently used
Translation Lookaside Buffer

• Given a virtual address, processor examines the TLB
• If page table entry is present (TLB hit), the frame number is retrieved and the real address is formed
• If page table entry is not found in the TLB (TLB miss), the page number is used to index the process page table
Translation Lookaside Buffer

• First checks if page is already in main memory
  – If not in main memory a page fault is issued
• The TLB is updated to include the new page entry
Figure 8.7 Use of a Translation Lookaside Buffer
Figure 8.8  Operation of Paging and Translation Lookaside Buffer (TLB) [FURH87]
Figure 8.9  Direct Versus Associative Lookup for Page Table Entries
Figure 8.10 Translation Lookaside Buffer and Cache Operation
Page Size - A Tradeoff Space

- Smaller page size, less amount of internal fragmentation
- Smaller page size, more pages required per process
- More pages per process means larger page tables
- Larger page tables means large portion of page tables in virtual memory
- Secondary memory is designed to efficiently transfer large blocks of data so a large page size is better
Page Size

• Small page size, large number of pages will be found in main memory
• As time goes on during execution, the pages in memory will all contain portions of the process near recent references. Page faults low.
• Increased page size causes pages to contain locations further from any recent reference.
Figure 8.11 Typical Paging Behavior of a Program

$P$ = size of entire process
$W$ = working set size
$N$ = total number of pages in process
# Example Page Sizes

## Table 8.2 Example Page Sizes

<table>
<thead>
<tr>
<th>Computer</th>
<th>Page Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas</td>
<td>512 48-bit words</td>
</tr>
<tr>
<td>Honeywell-Multics</td>
<td>1024 36-bit word</td>
</tr>
<tr>
<td>IBM 370/XA and 370/ESA</td>
<td>4 Kbytes</td>
</tr>
<tr>
<td>VAX family</td>
<td>512 bytes</td>
</tr>
<tr>
<td>IBM AS/400</td>
<td>512 bytes</td>
</tr>
<tr>
<td>DEC Alpha</td>
<td>8 Kbytes</td>
</tr>
<tr>
<td>MIPS</td>
<td>4 kbytes to 16 Mbytes</td>
</tr>
<tr>
<td>UltraSPARC</td>
<td>8 Kbytes to 4 Mbytes</td>
</tr>
<tr>
<td>Pentium</td>
<td>4 Kbytes or 4 Mbytes</td>
</tr>
<tr>
<td>PowerPc</td>
<td>4 Kbytes</td>
</tr>
<tr>
<td>Itanium</td>
<td>4 Kbytes to 256 Mbytes</td>
</tr>
</tbody>
</table>