Memory Management Unit & x86 Protected Mode
Table of Contents

• Virtual Memory
• Memory Page Protection
• Translation Lookaside Buffer
• Memory Management Unit
• BIOS
• X86 Protected Mode
Multi tasking kernel

- Multiple processes at the same time
- Each process needs its own memory
- Libraries
  - Static libraries
  - Shared libraries
- Processes may violate access rules
Virtual Memory

- On a Multi-process platform
  - Every process sees the same address space
- Every process has the same entry point
  - So, the application designer need not to be worry about memory addresses and ...
  - Debugging is easy
- There is a mapping between
  - Virtual address of each process
  - Physical memory
- Physical memory
  - Physical RAM (fast limited space)
  - Physical Disk or flash memory (low large space)
Virtual Memory

• Process Thinks
  • It has access to a
    • Large
    • Continuous memory region
  • In real
    • It is not true!
Segmented Virtual Addressing

- Not used in today systems
- Basic idea
  - You can keep separate segment register address for each process
  - To obtain physical memory address
    - Offset value provided by process is combined with segment value
Paged Virtual Addressing

• We divide memory into pages
• We create a mapping
  • Between virtual page number and physical page number
• Each time a process executes
  • A number of memory pages will be allocated to process
  • Memory pages may be added or deleted from process space
### Paged Virtual Memory

**Physical Memory**

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100****</td>
<td></td>
</tr>
<tr>
<td>0x101****</td>
<td></td>
</tr>
<tr>
<td>0x102****</td>
<td></td>
</tr>
<tr>
<td>0x103****</td>
<td></td>
</tr>
</tbody>
</table>

**Page Table**

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100****</td>
<td></td>
</tr>
<tr>
<td>0x101****</td>
<td></td>
</tr>
<tr>
<td>0x102****</td>
<td></td>
</tr>
<tr>
<td>0x103****</td>
<td></td>
</tr>
</tbody>
</table>

**Process 1**

**Process 2**
## Multi-Level Paged Virtual Memory

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100*******</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101*******</td>
<td></td>
</tr>
<tr>
<td>0x102*******</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01*******</td>
<td></td>
</tr>
<tr>
<td>0x02*******</td>
<td></td>
</tr>
<tr>
<td>0x03*******</td>
<td></td>
</tr>
<tr>
<td>0x04*******</td>
<td></td>
</tr>
<tr>
<td>0x05*******</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100*******</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101*******</td>
<td></td>
</tr>
<tr>
<td>0x102*******</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100*******</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101*******</td>
<td></td>
</tr>
<tr>
<td>0x102*******</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100*******</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101*******</td>
<td></td>
</tr>
<tr>
<td>0x102*******</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual Addr</th>
<th>Physical Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x100*******</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101*******</td>
<td></td>
</tr>
<tr>
<td>0x102*******</td>
<td></td>
</tr>
</tbody>
</table>
Paging Supervisor

- Every time a memory page is accessed
  - A time stamp will be updated
- Process with no activity
  - Memory pages are not required to be stored in physical RAM
- A “page file” will be created on disk
  - Holds “not used” memory pages
- When the process activates again
  - Tries to access the specified page
  - Page no longer exist in memory
  - A page fault exception will occur
- Exceptions are similar to interrupts
  - A routine will be executed to serve the exception
Page Permissions

• Each Page has its own set of permissions
  • Some pages are read-only
  • Some pages are specific to one process
  • Some pages can be accessed globally
• Page table also contains permissions for each page
Memory Management Unit

- A computer hardware
  - Responsible for handling accesses to memory requested by CPU
- Tasks
  - Virtual to physical address translation
  - Memory protection
  - Cache Control
  - ...

TLB: Translation Look-aside Buffer
MMU: Memory Management Unit
CPU: Central Processing Unit
Translation Lookaside Buffer

- Page Tables
  - Hold entire mapping between virtual memory and physical memory
  - Page tables are big!
  - Complete page table can be stored in main memory
- Translation Lookaside Buffer:
  - A CPU Cache
  - Used by MMU
  - To improve
    - Virtual address translation speed
- TLB
  - Is usually a CAM (Content Addressable Memory)
  - Virtual address (mainly upper bits)(CAM search key)
  - output: Equivalent physical address (again the upper bits)(CAM search result)
  - TLB hit: found the translation
  - TLB miss: not found! Look at the main “page table” (page walk)
Translation Lookaside Buffer (2)

- TLB can be located
  - Between CPU core and cache
    - Stored cache addresses are physical
  - Between cache and memory
    - Stored cache addresses are virtual
    - ...
- TLB in Harvard Architecture
  - Data side TLB
  - Instruction side TLB
- Cache access can happen in parallel with TLB lookup
- Software loaded TLB
  - Operating system is responsible for updating TLB entries
- Hardware loaded TLB
  - Updates on TLB are hidden from OS (dedicated hardware)
A Typical TLB

- Size: 4096 entries
- Hit time: 1 clock cycle
- Miss penalty: 100 clock cycles
- Miss rate: 1%

- A brief talk on TLB contents while switching context
X86 Protected Mode

- Basically x86 has two modes of operation
  - Real mode: addresses seen by processes are physical ones
  - Protected mode:
    - Memory paging can be activated
    - Virtual addressing will become enabled
    - Memory protection mechanism are activated
    - Larger physical address range
    - Very useful for multi-tasking environments