Memory Management Part 1
The Address-Space Concept

- Protect processes from one another
- Protect the OS from user processes
- Provide efficient management of available storage
Memory Fence

User Area

OS
Base and Bounds Registers
Swapping
Overlays

Overlay

Resident
Virtual Memory

Process 1

Process 2

Process 3

Memory

Disk
Structuring Virtual Memory

• Paging
  – divide the address space into fixed-size pages

• Segmentation
  – divide the address space into variable-size segments (typically each corresponding to some logical unit of the program, such as a module or subroutine)
Paging

• Map fixed-size pages into memory (into page frames)

• Many hardware mapping techniques
  – page tables
  – translation lookaside buffers
Page Tables

Page # | Offset
------|------

Virtual Address

V | M | R | Prot | Page Frame #
Page-Table Size

• Consider a full $2^{32}$-byte address space
  – assume 4096-byte ($2^{12}$-byte) pages
  – 4 bytes per page table entry
  – the page table would consist of $2^{32}/2^{12} (= 2^{20})$ entries
  – its size would be $2^{22}$ bytes (or 4 megabytes)
Forward-Mapped Page Table

L1 Page #  |  L2 Page #  |  Offset

L1 Page table  |  L2 Page tables  |  Page frame
IA32 Paging

- CR3
- Page directory table
- Page table
- Page
The diagram illustrates the x86-64 Virtual Address Format 1. The format is shown as a 64-bit virtual address, with the most significant bits (63-20) used for page directory and page table entries, and the least significant bits (19-0) used for page table entries. The diagram includes the page map table, page directory pointer table, page directory table, and page table, leading to the 4KB page.
x86-64 Virtual Address
Format 3

Page map table

Page directory
pointer table

1GB page
Why Multiple Page Sizes?

• External fragmentation
  – for region composed of 4KB pages, average external fragmentation is 2KB
  – for region composed of 1GB pages, average external fragmentation is 512MB

• Page-table overhead
  – larger page sizes have fewer page tables
    - less overhead in representing mappings
      • both in memory and in cache
Linear Page Table

S 21 9
Space VPN Offset

Space x Page Table

Page
• VAX architecture introduced in 1978
  – memory cost $40,000/MB
    - 3.8¢/byte
      (.475¢/bit)
Linear Page-Table Management

• 00 and 01 page tables each require contiguous locations in 10 space
  – with 512-byte pages, 8MB each:
    - maximum of 128 such page tables
    - (need room for other things, e.g. OS)
• Reduce size requirements with partial page tables
  – length registers constrain size of each space
Traditional Unix with Linear PTs

- **01 AS**
  - stack
- **00 AS**
  - dynamic
  - bss
  - data
  - text
• Limit size of 00 space to 1 MB
  – requires 16-page 00 page table in 10 virtual memory
    - requires 16 entries in 10 page table
• Same requirements if 01 space limited to 1 MB
• What are real-memory requirements?
  – 10 page table resides in real memory
  – at least one page of real memory must be allocated for each of 00 and 01 page tables
  – minimum real memory is 1152 bytes
    - $43.95 in 1978
Modern Unix

- Stack 1
- Stack 2
- Stack 3
- Mapped File
- Mapped File
- Dynamic
- BSS
- Data
- Text
• Requires sufficient 10 page-table entries to map almost all of 00 and 01 space
  – $2^{14}$ 10 page-table entries for each space
    - requiring 64KB each, 128KB total
    - $5000 in 1978

  - <1¢/process today
    • who cares?
    • increase address space from $2^{32}$ to $2^{64}$
      – 4,294,967,296-fold increase
      – significant …
Hashed Page Tables
Clustered Page Tables

- Page #
- Offset
- Virtual Address

Hash Table

Hash

Tag
Link
Entry
Entry
Entry
Entry
Entry
Entry
Entry
Entry
Entry

Pages

...
### Inverted Page Tables

<table>
<thead>
<tr>
<th>Page #</th>
<th>Offset</th>
<th>Virtual Address</th>
</tr>
</thead>
</table>

- **Hash Table**: Stores page numbers and their corresponding offsets.
- **Page Frame Table**: Contains information about available page frames.

The hash function takes a virtual address and maps it to a hash value, which is then used to look up the corresponding page frame in the page frame table.
Translation Lookaside Buffers

Tag | Key | Offset
---|---|---
14 | 6 | 12

Tag | Page Frame #
---|---
Tag | Page Frame #
Tag | Page Frame #

... | ...

Tag | Page Frame #
TLBs and Multiprocessors

Processor 1

Processor 2

TLB

TLB

Process
TLB Shootdown Algorithm

// shooter code
for all processors i sharing address space
  interrupt(i);
for all processors i sharing address space
  while (noted[i] == 0)
    ;
modify_page_table();
update_or_flush_tlb();
done[me] = 1;

// shootee i interrupt handler
receive_interrupt_from_processor j
  noted[i] = 1
while (done[j] == 0)
  ;
flush_tlb()
IA-64 Address Translation

• TLB
  – software-managed
• Virtual Hash Page Table (VHPT)
  – per-region linear page table
  – single large hashed page table
Translation: TLB
Translation: TLB Miss

region reg

Hash Function

page number

offset

Virtual Hash Page Table
Virtual Machines Meet Virtual Memory

Virtual machine’s page table

Virtual real memory

Virtual virtual memory

VMM’s page table

Real memory

Shadow page table
Paravirtualization to the Rescue
Hardware to the Rescue

virtual memory

<table>
<thead>
<tr>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>i</td>
</tr>
<tr>
<td>2</td>
<td>i</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

virtual memory

<table>
<thead>
<tr>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

real memory

Extended Page Tables

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>i</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
x86 Paging with EPT

- CR3
- Page Directory (pd)
- Page Table (pt)
- EPTP
- Page