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

User Area

OS
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

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

Virtual Address

| V | M | R | Prot | Page Frame # |

Operating Systems in Depth
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

<table>
<thead>
<tr>
<th>L1 Page #</th>
<th>L2 Page #</th>
<th>Offset</th>
</tr>
</thead>
</table>

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

CR3

Page directory table

Page table

Page
Quiz 1

Suppose a process on an IA32 has exactly one page residing in real memory. What is the total number of combined pages of page-directory table and page tables required to map this page?

a) 1  
b) 2  
c) 4  
d) 8
x86-64 Virtual Address Format 2

63  47  38  29  20  0

unused

Page map table

Page directory table

Page directory pointer table

2MB 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
Address Space

- OS kernel: $0xffffffffffffffff$, $2^{47}$ bytes
- Illegal: $0xffffffff80000000000000$, $2^{64} - 2^{48}$ bytes
- User: $0x00007fffffffffff$, $2^{47}$ bytes
Linear Page Table

Space

VPN

Offset

Page

Space x Page Table
VAX Linear Page Translation

VA: 00 VPN Offset

VPN: 00 BR

PTEA: 10 VPN Offset

10 BR

10 PT

00 AS

10 AS

00 PT
• 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

00 AS

stack

dynamic

bss

data

text
Quiz 2

Suppose the page size is 512 bytes (2^9) and each page-table entry requires 4 bytes. How many pages of page-table entries are required to map 1 megabyte (2^{20}) of address space?

a) 4  
b) 8  
c) 16  
d) 32
$\

- 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
$ 

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

- Page #
- Offset
- Virtual Address

Hash

Tag
Link
Entry
Tag
Link
Entry
Tag
Link
Entry
Tag
Link
Entry
Tag
Link
Entry

Page Frame
## Clustered Page Tables

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

### Hash Table
- Tag
- Link
- Entry

### Pages
...
Inverted Page Tables

Page # | Offset
---|---

Virtual Address

Hash

Hash Table

Page Frame Table
Translation Lookaside Buffers

<table>
<thead>
<tr>
<th>Tag</th>
<th>Key</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>6</td>
<td>12</td>
</tr>
</tbody>
</table>

- 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()}
Region registers

Region

VRN

Page

Offset

23 0

RID

Intel IA-64
IA-64 Address Translation

• TLB
  – software-managed

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

region reg → TLB → page number → offset

TLB → offset
Translation: TLB Miss

region reg

Hash Function

Virtual Hash Page Table

| page number | offset |
Virtual Machines Meet Virtual Memory

Virtual machine’s page table

Virtual real memory

Virtual virtual memory

Real memory

VMM’s page table

Shadow page table
Paravirtualization to the Rescue

virtual memory

real memory

Direct translation

virtual memory

real memory

0 1
1 2
2 3
3 i

0 1
1 i
2 i
3 1

0 3
1 1
2 1
3 3
Hardware to the Rescue

virtual memory

0 1
1 i
2 i
3 2

virtual real memory

0 i
1 3
2 1
3 2

real memory

Extended Page Tables
x86 Paging with EPT

CR3

Page Directory (pd)

Page Table (pt)

EPTP

Page
Quiz 3

We’d like to virtualize EPT. Assume that setting EPTP causes a VMexit if done on a VMM that’s not running in real ring -1. What does the VMM running at level 0 do when it receives such a VMexit from a VMM running at level 1?

a) nothing: the EPT mechanism is virtualized by the hardware

b) it sets EPTP to point to the composition of the page tables mapping the level 1 VM (on which the level 1 VMM runs) and the page tables pointed to by the value being attempted to be put in EPT

c) something else