



Xen Memory Layout

Presented By Michael LeMay

Introduction

- Xen runs in ring 0 to protect itself from guest operating systems
- Must also protect its memory regions from modification by guest OSes
- Best solution is full context switch
 - Can be cheap if software TLB used
 - Cheap if TLB tagged with address space (RISC servers)
 - X86 has neither, uses hardware TLB.

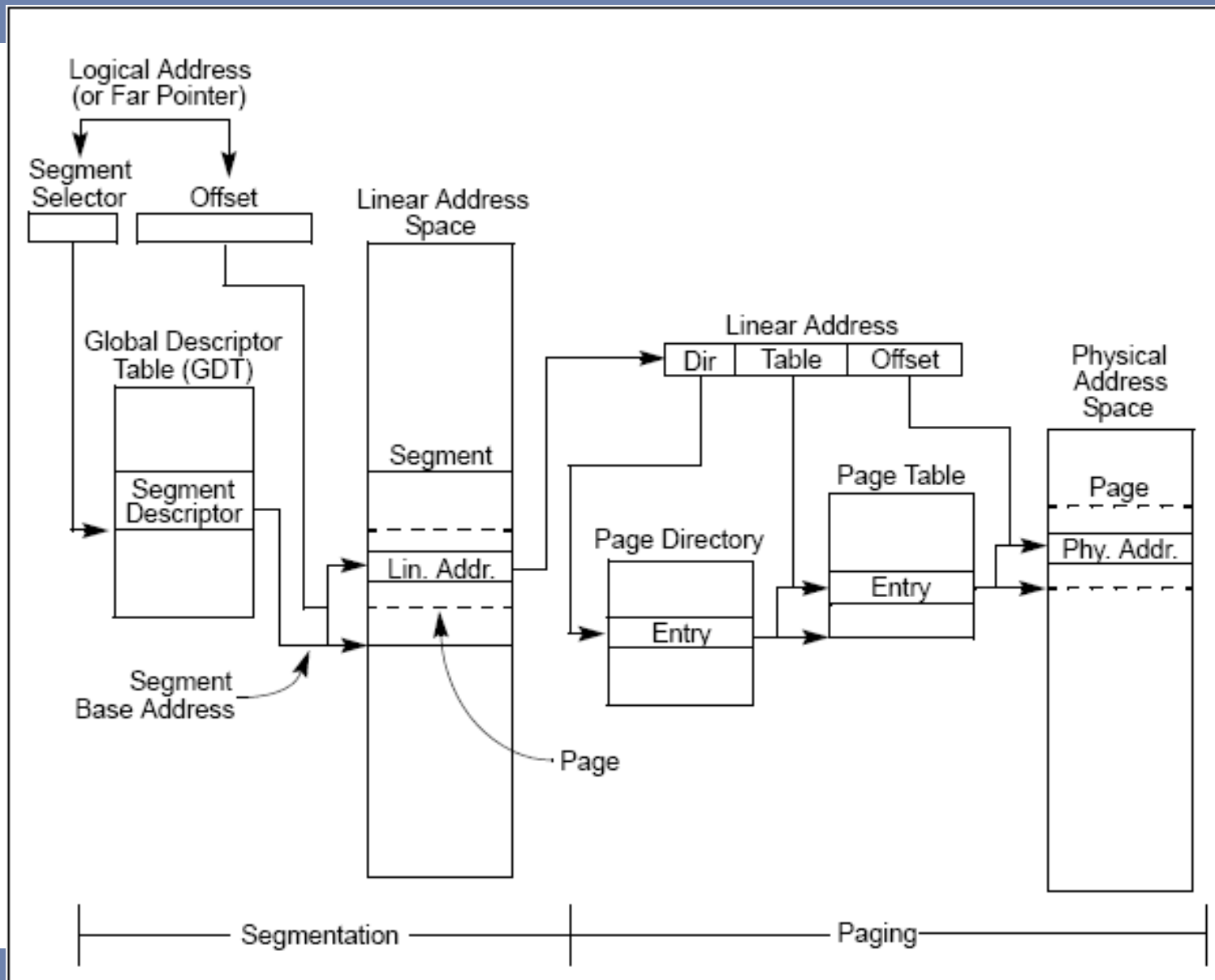
X86 Memory Management Review

- X86 supports both segmentation and paging
- Each process gets a private set of segments and is prevented from modifying other processes' segments
- Each segment described by descriptor in GDT
 - Segment size, base address
 - Access rights, privilege level
- Linux sets all segments to the same 4GB physical location (CS, DS, SS)

Paging Review

- Segments divided into small pages
- Pages can be located in physical RAM or stored on disk
- Typical page size on X86 is 4KB

Overview of X86 Mem Mgmt



X86 Translation Lookaside Buffer

- Hardware cache containing parts of page table
- Translates virtual into real addresses
- TLB must be changed when context switching
- Minimum cost on Pentium 4 to change TLB is **516 cycles (184ns)**
- (see <http://www.mega-tokyo.com/osfaq2/index.php/Context%20Switching>)
- Thus, Xen avoids context switching on system calls for performance reasons

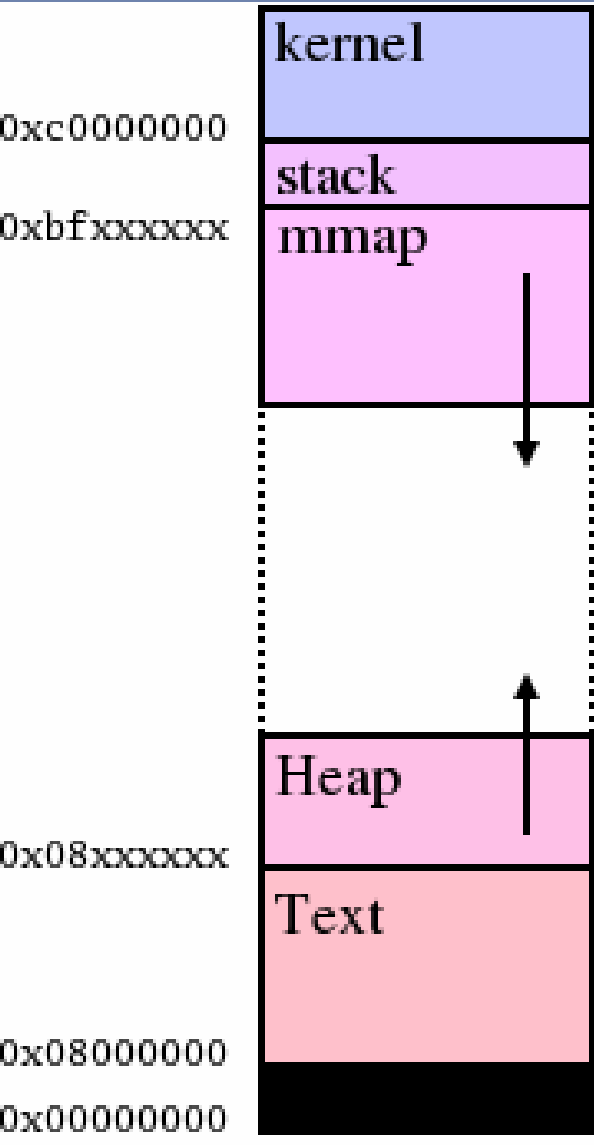
Xen OS Modifications

- OS must manage page table
- In Xen, OS rescinds write access to page table after it is created for new process
- OS requests updates through hypervisor
- Requests often batched
- Segment registers only accessible via hypervisor
 - Must have lower privileges than Xen
 - Must not allow access to Xen memory region

X86 Specialties

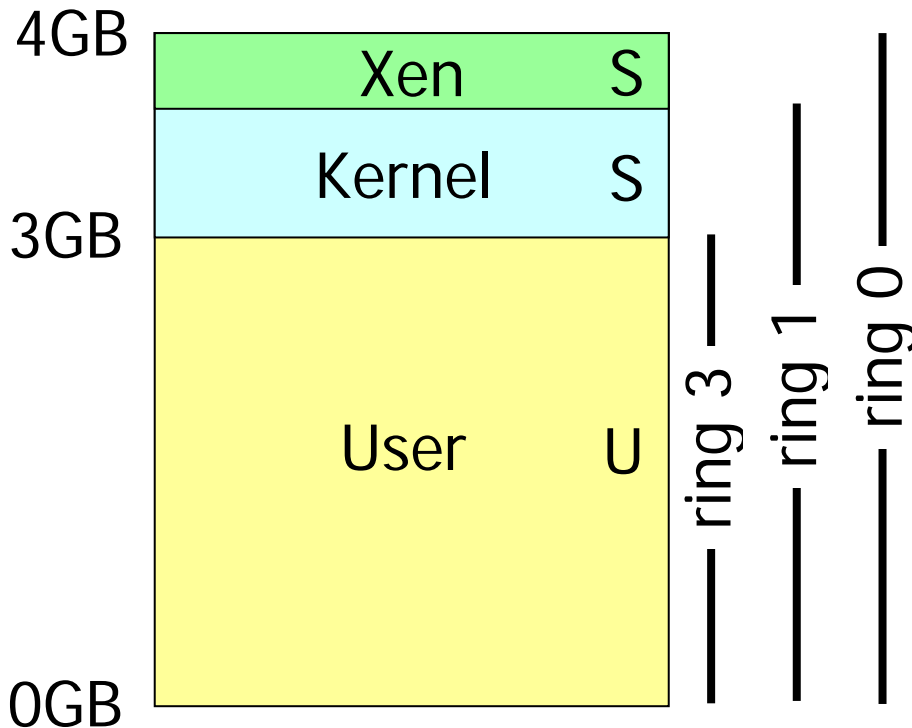
- On ordinary ia32, Xen runs in top 64MB of 4GB mem space
- On Physical Address Extension (PAE) equipped machines, runs in top 168MB of 16GB mem space (36-bit address space supported on Pentium Pro +)
- Larger block used in middle of 4 terabyte x86-64 space (48-bit address space)

X86 Linux Memory Layout



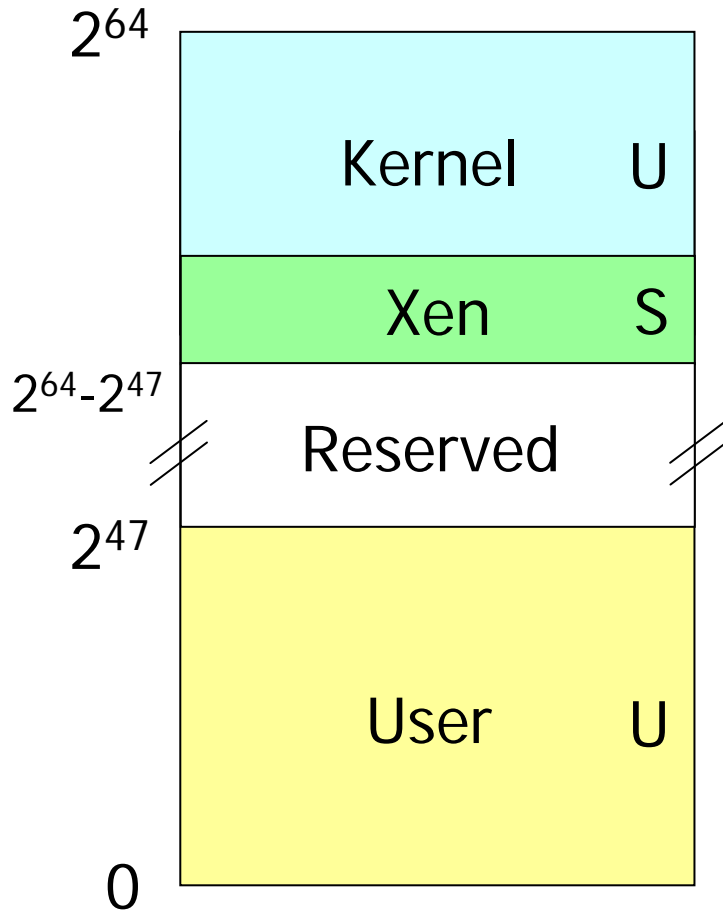
Xen occupies unused memory part above kernel, causes no impact on user programs

x86_32



- Xen reserves top of VA space
- Segmentation protects Xen from kernel
- System call speed unchanged
- Xen 3 now supports PAE for >4GB mem

x86_64



- Large VA space makes life a lot easier, but:
- No segment limit support
- ➔ Need to use page-level protection to protect hypervisor

X86 Memory Vulnerabilities

- Hardware devices can perform DMA to protected segment
- Counteracted by integrating IOMMU into chipset, which is expected soon

ARM MMU

- Three-level MMU, unlike x86
 - 1KB, 4KB, 64KB pages
 - Access controls over 1KB and 16KB page quarters
 - 1MB segments (sections in ARM lit.)
 - 4096 top-level descriptors (4GB total virtual space)
- Two-part hardware TLB:
 - Main TLB
 - Two-way set associative cache with 64 entries total
 - Lockdown TLB
 - Eight-entry fully-associative cache to prevent page walk penalties for eight selected regions

MMU Protections

- Domain Access Control Register: determines whether access permissions will be checked in each of 16 domains
- Top-level descriptors specify domain
- Lower-level sections and pages are marked with AP bits
- AP bits can selectively allow R/W access to supervisor- and/or user-mode code

Fast Context Switch Extension

- ARM 1026 has single register that directs replacement of top 7 bits of virtual addresses depending on current process ID (128 processes supported)
- Contents of caches and TLBs are not invalidated
- Thus, Xen on ARM could feasibly use a context switch for every system call

References

- Wikipedia TLB:
http://en.wikipedia.org/wiki/Translation_Lookaside_Buffer
- Xen Overview Presentation:
<http://www.cl.cam.ac.uk/netos/papers/2005-xen-ols.ppt>
- Page Table Management:
<http://www.informit.com/articles/article.asp?p=336868>
- Explore the Linux memory model: <http://www-128.ibm.com/developerworks/linux/library/l-memmod/>
- Pentium 4 system programmer's manual:
<http://download.intel.com/design/Pentium4/manuals/25366818.pdf>
- ARM 1026EJ-S manual:
http://www.arm.com/pdfs/DDI0244C_1026ejs_r0p2_trm.pdf