Using segmentation to build a capability-based single address space operating system

Wuyang Chung

wy-chung@outlook.com

Outline

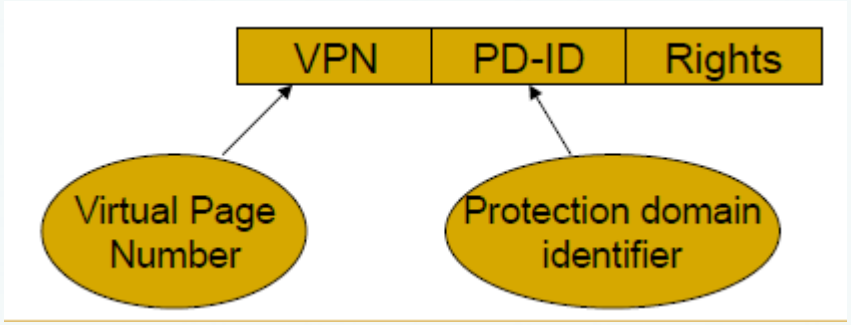
- Single Address Space Operating System
- Capability-based Addressing
- Segmentation
- Benefits of Segmentation
- Compatibility with Old Programs
- Conclusion

Single Address Space OS

- Problems with multiple address space OS
  - Cannot use pointers in shared buffer
  - On platforms without ASID
    - Need to flush TLB during address space switch
  - On platforms with ASID
    - Duplication of translation information in TLB
- Advantages of single address space OS
  - Encourage of sharing memory between protection domains
  - Low context switch overhead
- Challenge of single address space OS
  - Protection

Multiple Protection Domains on a Single Address Space

- Domain page model
  - Protection lookaside buffer

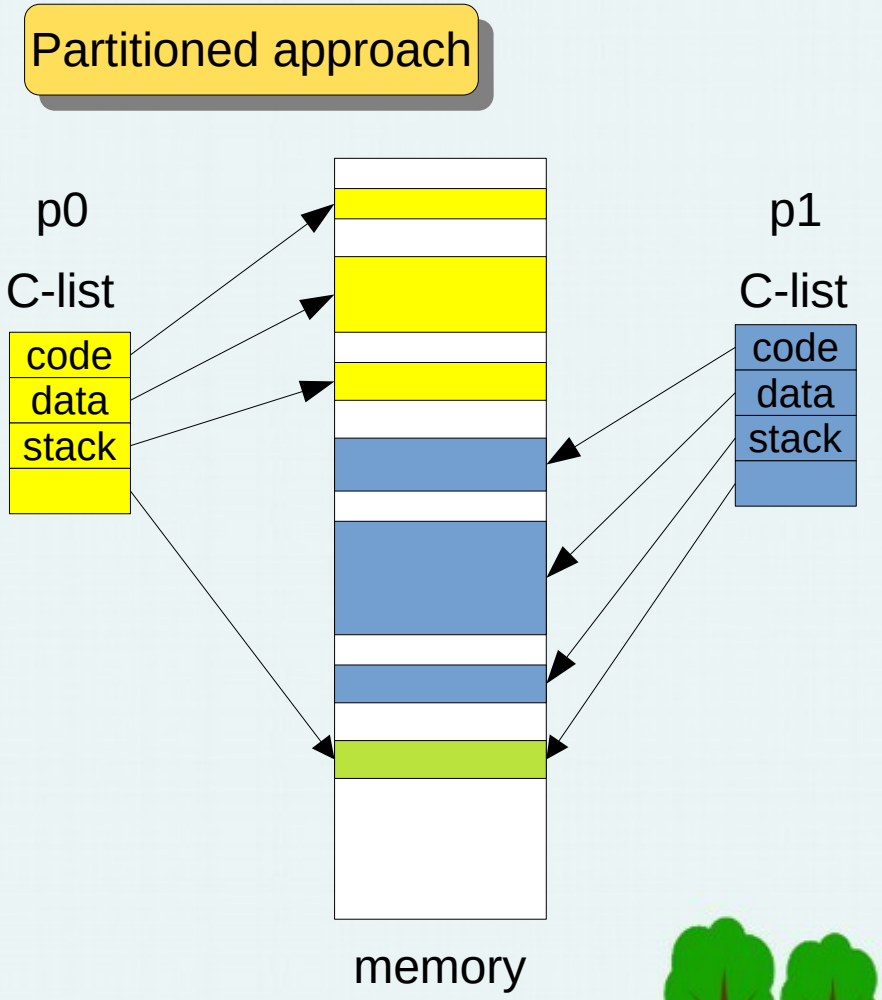


- Page group model
  - There is an AID (access identifier) field in page table entry
  - The CPU has several PID (protection identifier) registers
  - A process can access a page if any of the PIDs is equal to the page AID
- Capability-based addressing

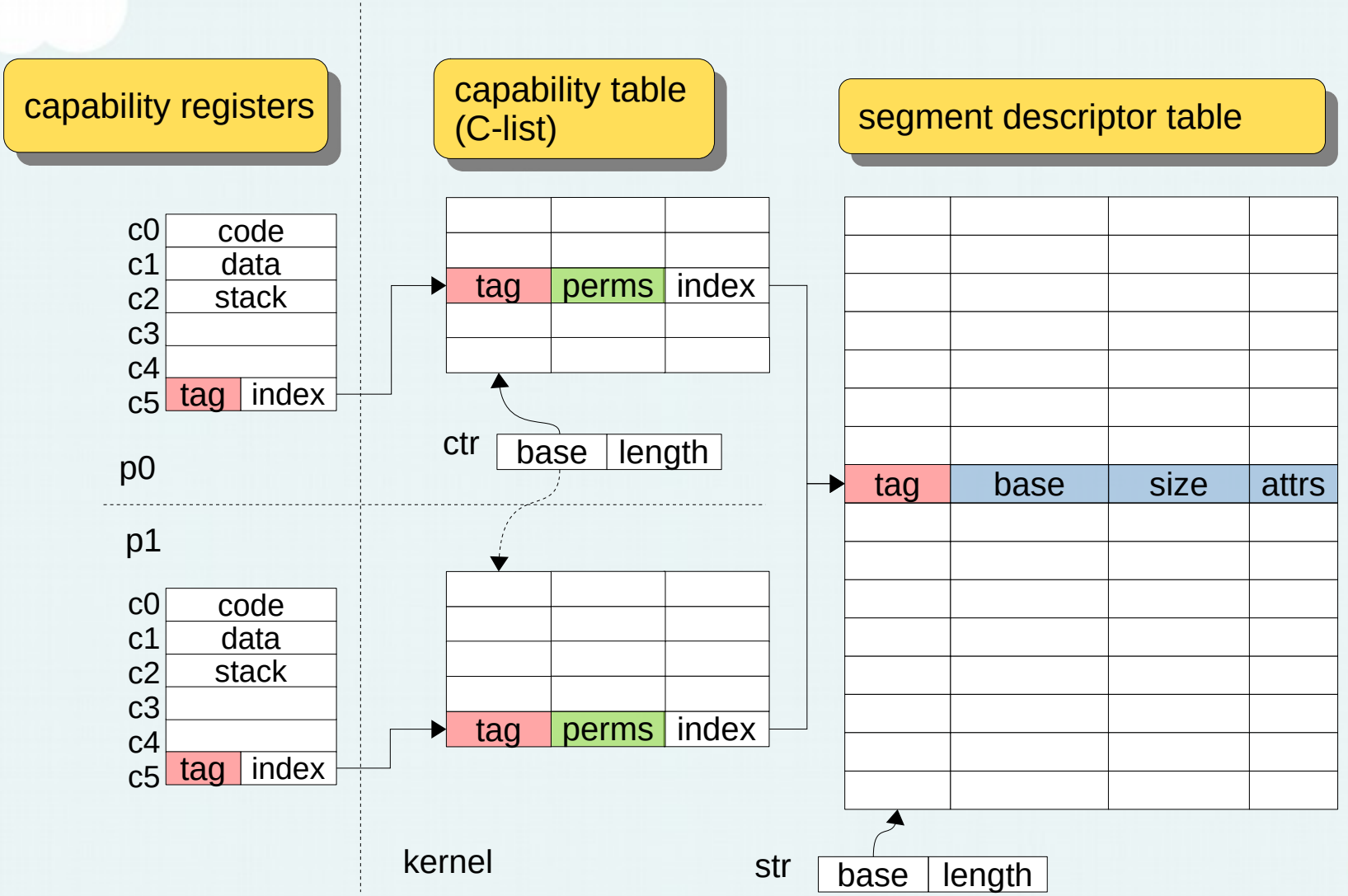
Capability-based Addressing

capability | permission | ~~unique object ID~~ segment ID

- The most important thing in a capability system is to prevent capability from being forged
- Two approaches
  - Tagged approach
    - e.g. CHERI
  - Partitioned approach
    - C-list

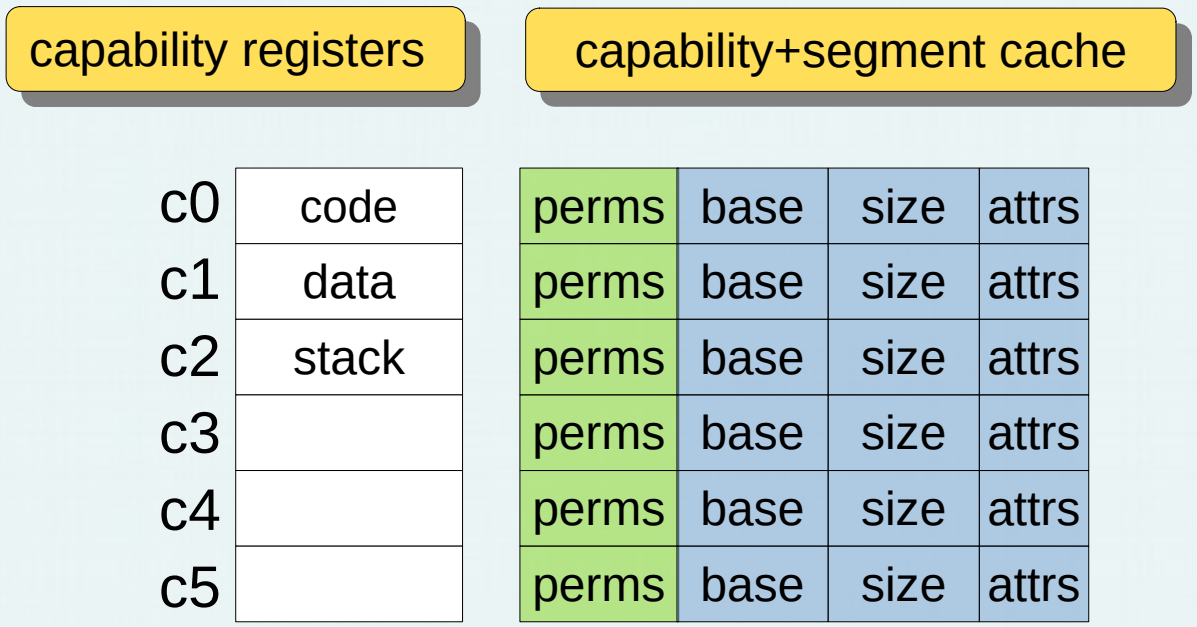


Capability Using Segmentation



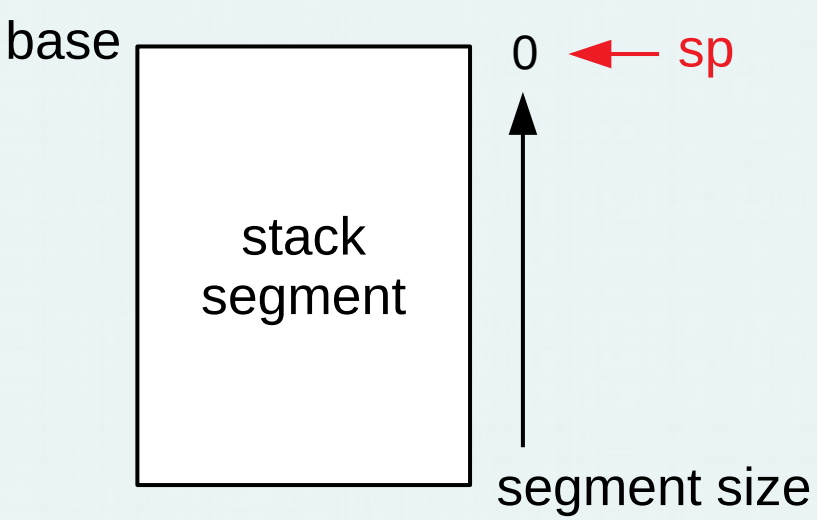
Segmentation Hardware

- Segment TLB
  - Cache segment descriptors
- Each capability handle register has a capability+segment cache

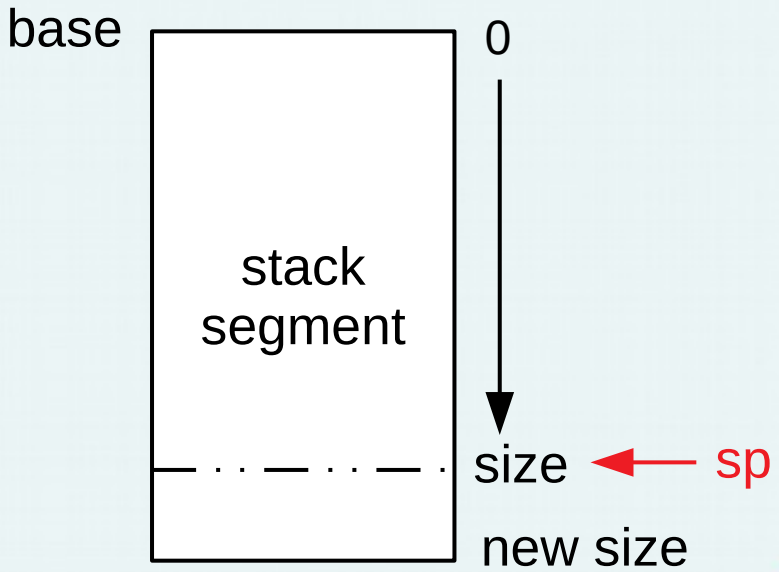


Stack Growth Direction

- Stack should grow towards ∞ instead of 0
  - Can expand the stack segment when it reaches its maximum size



towards 0



towards ∞

Segment Types

- Segmentation translates logical address to linear address
- Segment types
  - Virtual segment
    - Linear address is virtual address
  - Physical segment
    - Linear address is physical address
  - I/O segment
    - Linear address is I/O address

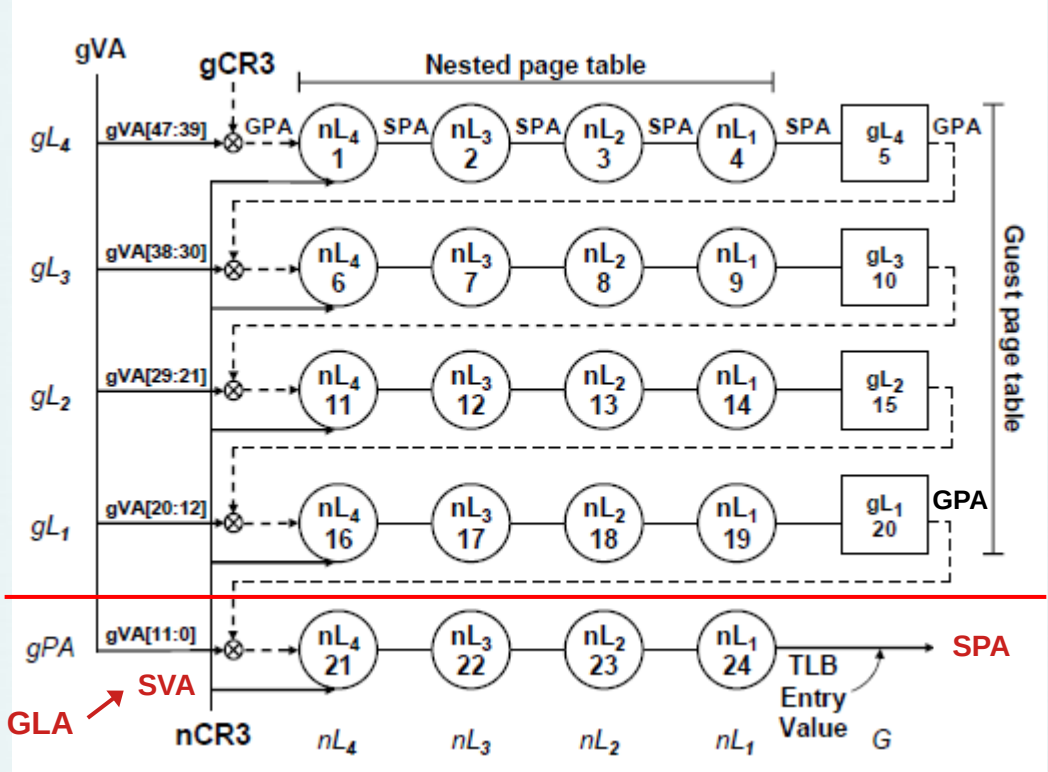
logical address is
{capability register, offset within the segment}

Virtual Segment

- Simplification of segment move
  - Don't need to copy the segment data, only need to copy the virtual to physical mapping
  - Don't need to fix any pointers within the segment

Simplification of TLB miss walk in Guest Virtual Machine

- Hardware needs to do two virtual address translations for TLB miss on a virtual machine
  - Guest virtual address to guest physical address
  - System virtual address to system physical address
- Only one virtual address translation is needed with segmentation
  - Guest logical address can be translated directly to system virtual address



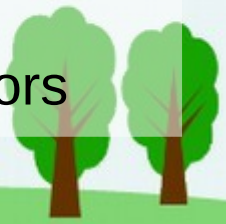
Two-Dimensional Page Table Walk

Accelerating two-dimensional page walks for virtualized systems, acm ASPLOS, 2008

logical address is {capability register, offset within the segment}

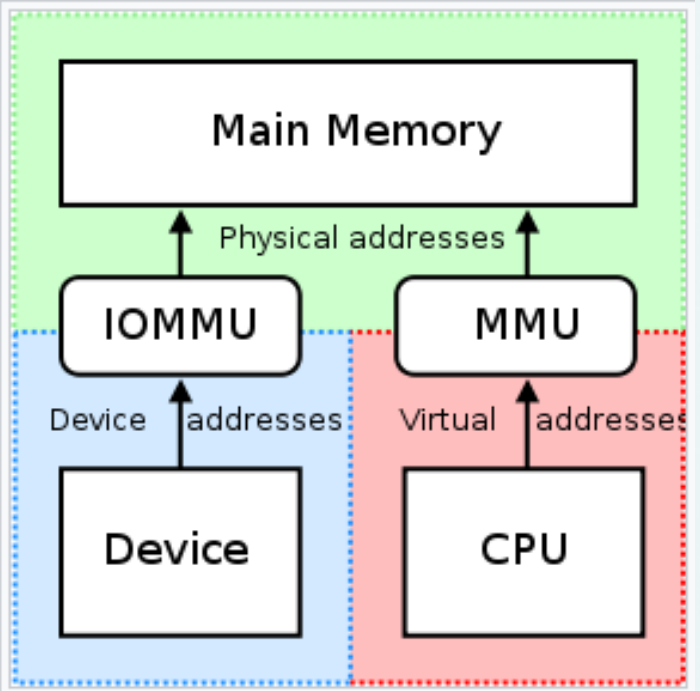
Physical Segment

- Benefits
  - Software-manged TLB
    - Can guarantee no page TLB misses during page TLB miss handling
  - Performance improvement
    - TLB misses consume up to 51% of execution cycles for big memory servers
      - Arkaprava Basu, Jayneel Gandhi, Jichuan Chang, Mark D. Hill, Michael M. Swift. Efficient virtual memory for big memory servers. In Proc. ISCA, 2013.
  - IOMMU simplification
    - No page translation is needed
    - Simple authorization
    - IOMMU is just a segment TLB for physical segment descriptors



Simplification of IOMMU

- Steps for device doing DMA via IOMMU
  1. Device sends the capability, offset and data length to IOMMU
  2. IOMMU verifies that the capability is valid and it points to a physical segment
  3. IOMMU then checks that the offset and length is within segment size
  4. IOMMU adds segment base address to the offset
  5. The calculated physical address is used to access system main memory

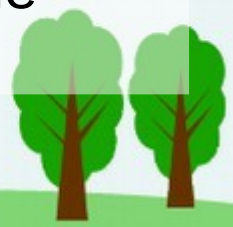


Comparison of the I/O memory management unit (IOMMU) to the memory management unit (MMU).

From Wikipedia

I/O Segment

- Purpose
  - Let memory address space for memory only and move all the other stuff to I/O address space, such as CPU's CSRs, PCI configuration space, video frame buffer, boot ROM, etc.
- Advantages
  - No memory holes in memory address space
  - Segment is more fine-grained then page
  - Only one segment TLB entry is needed for a large I/O segment
  - Same load/store instruction can be used to load from or store to either I/O or memory address space
  - User level device driver can access its hardware if it has the capabilities to its I/O segments



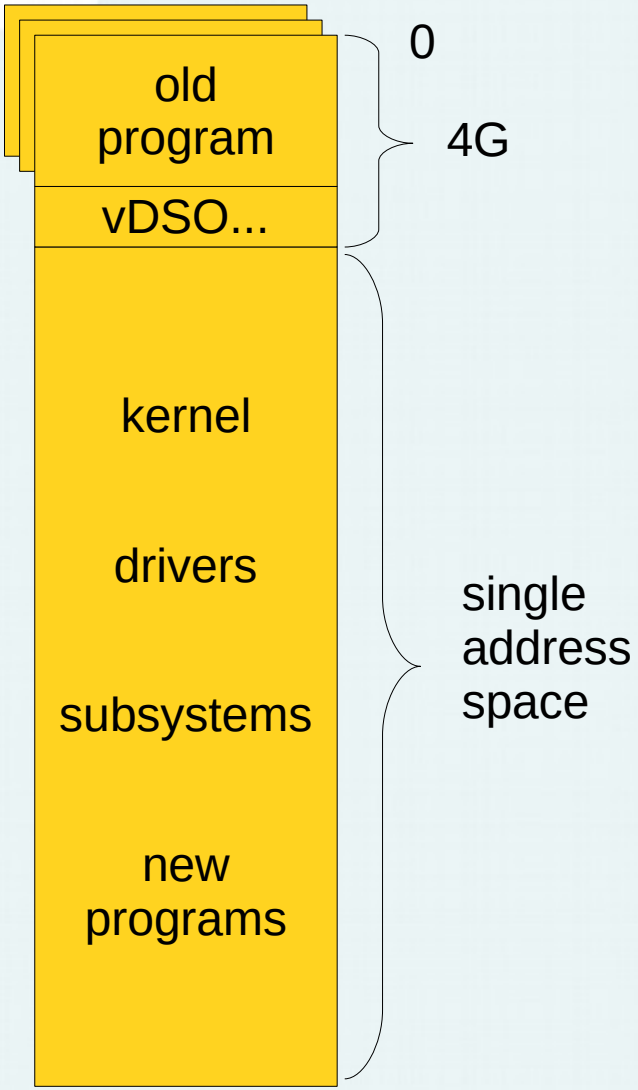
Simplification of Shared Library

- No need for PIC (Position Independent Code), GOT (Global Offset Table) and PLT (Procedure Linkage Table)
  - Shared library has its own code and data segment and segment always starts with offset 0
- Shared libraries can be partially linked
  - The offset part of the library's global variables and functions can be statically linked at link time
  - The segment part of the library's global variables and functions are linked at program load time

Simplification of Cross-domain Call

- Cross-domain call has many names: migrating thread model, protected control transfer, passive object model
- Currently thread is confined in two protection domains
- Thread should be able to travel around domains
  - Encourage modularity and improve security
  - Resource accounting can be made more accurate
  - Preferable for real-time systems
- Segmentation can make cross-domain call easier to implement
  - The same stack segment is used in server domain
  - To protect server from client
    - Move the stack capability from client domain to server domain
  - To protect client from server – stack domain boundary (sdb) register
    - Server can only access part of the stack with the address above the spb register

Compatibility with Old Programs



- Ideal platform
  - 32-bit program on 64-bit virtual address space
- The first 4G is reserved for old programs
- The reset of the virtual address space is for kernel, drivers, subsystems and new programs

Conclusion

- Lots of benefits we can get from single address space, capability and segmentation
- Changes
  - OS
    - Segmentation, Capability-based single address space
    - new system calls: sfork, sexec, etc.
  - Compiler support
    - Far pointer, far function call
    - New shared library implementation
  - Hardware
    - Capability and segmentation architecture

The End

Questions?