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

Wuyang Chung

wy-chung@outlook.com

Agenda

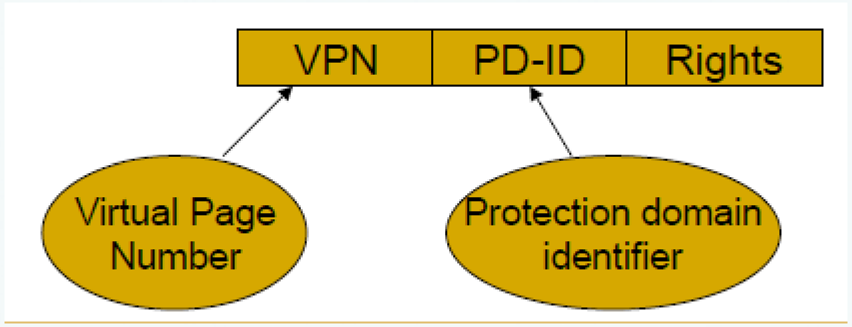
- Single Address Space Operating System
- Capability-based Addressing
- Segmentation
- Misc.
- Conclusion

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

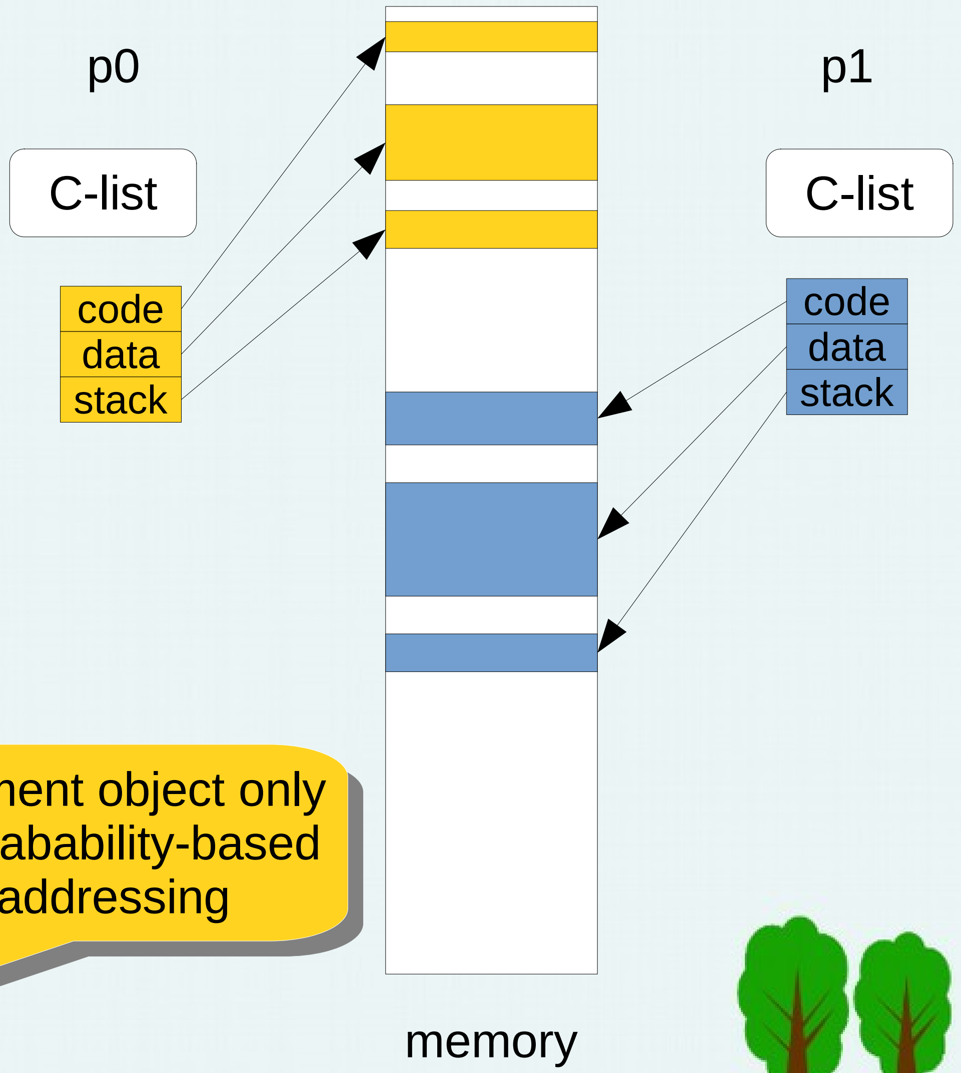
- Domain page model
  - Protection lookaside buffer



- Page group model
  - There is a lock in page table entry
  - Each process has several keys
  - A process can access a page if any one of its keys can unlock the page
- Capability-based addressing

Capability-based Addressing

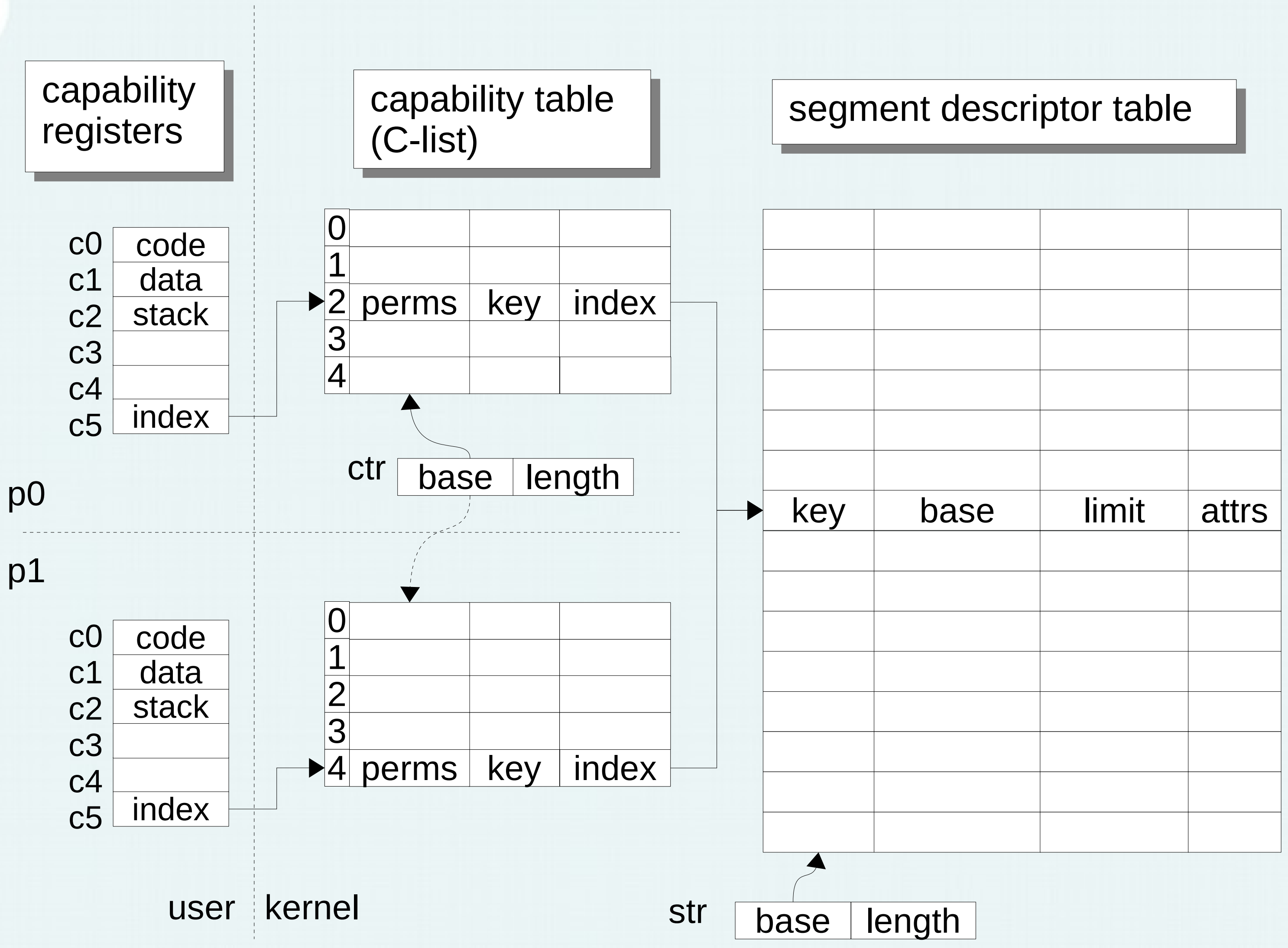
- Capability can create multiple protection domains on a single address space
- Two approaches
  - Tagged approach
    - e.g. CHERI
  - Partitioned approach
    - C-list



segment object only for cabability-based addressing

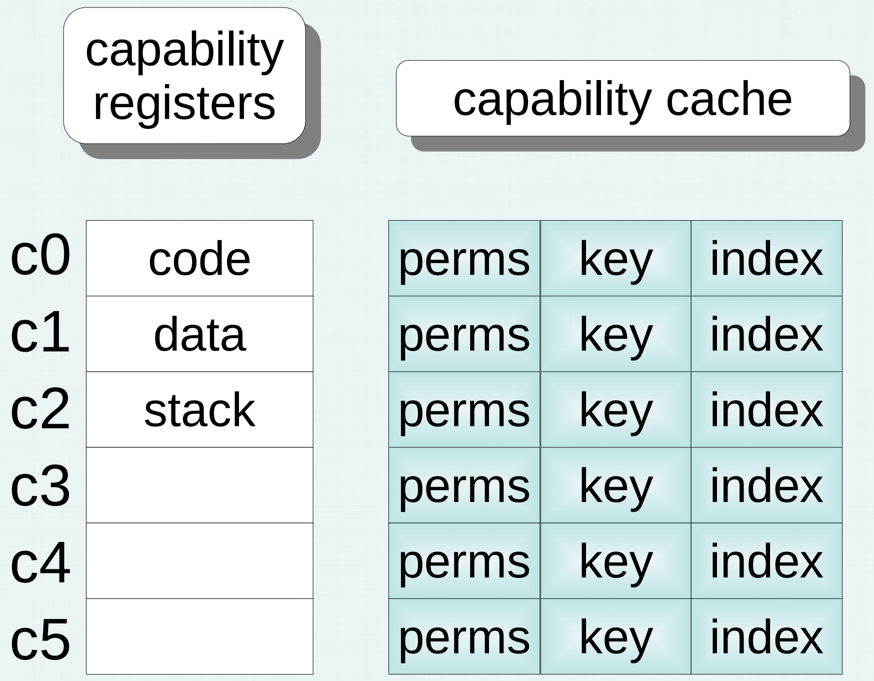
capability	permission	unique object ID

Segmentation for Capability



Segmentation Hardware

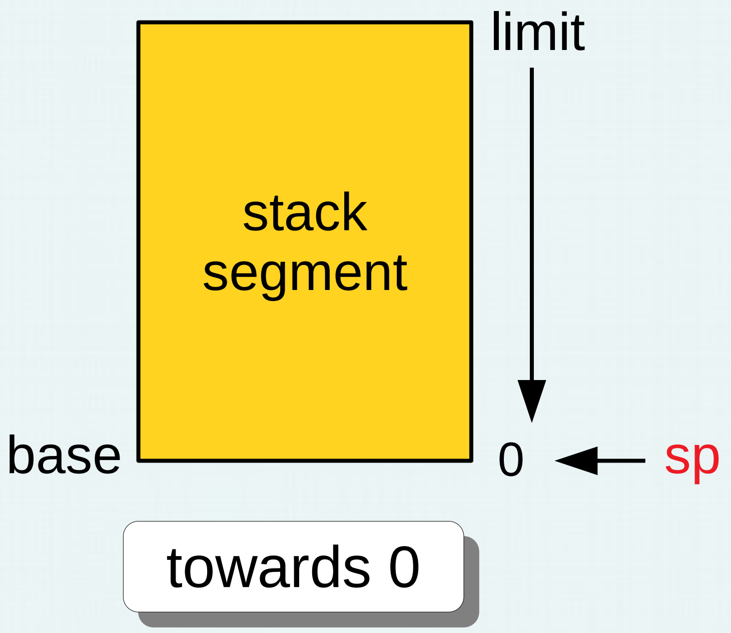
- Segment TLB
  - Cache segment descriptors
- Each capability register has a capability cache

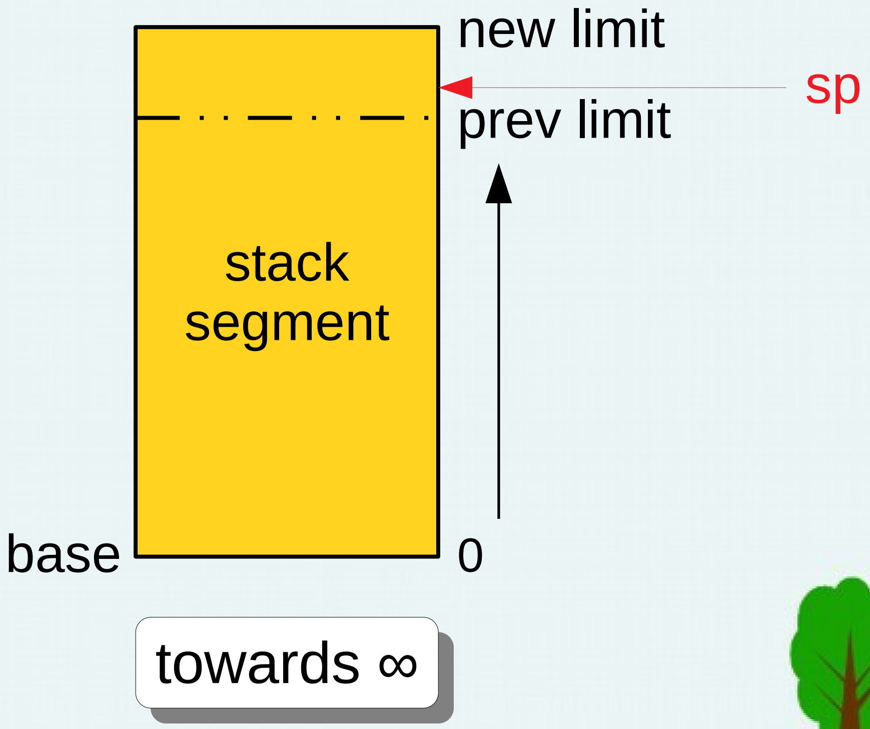


	capability registers	capability cache		
c0	code	perms	key	index
c1	data	perms	key	index
c2	stack	perms	key	index
c3		perms	key	index
c4		perms	key	index
c5		perms	key	index

Stack Growth Direction

- Should grows towards ∞ instead of 0
  - Can expand the stack segment when it reaches its maximum size





Segment Types

- Segmentation translates segment address to linear address
- Four segment types
  - Virtual segment
    - Linear address is virtual address
  - Physical segment
  - I/O segment
  - MSR (Model Specific Register) segment
- One load/store instruction can be used to load from or store to these segments

I/O Segment

- Let memory address space for memory only and move all the other stuff to I/O address space, such as PCI configuration space, video frame buffer, boot ROM, etc.
- Device driver can access its hardware device by using the capability to its I/O segment
- Advantages
  - No memory holes in memory address space
  - Segment is more fine-grained then page
  - Only one TLB entry is needed for a very large I/O segment
  - Device can only be accessed by a driver with the right capability

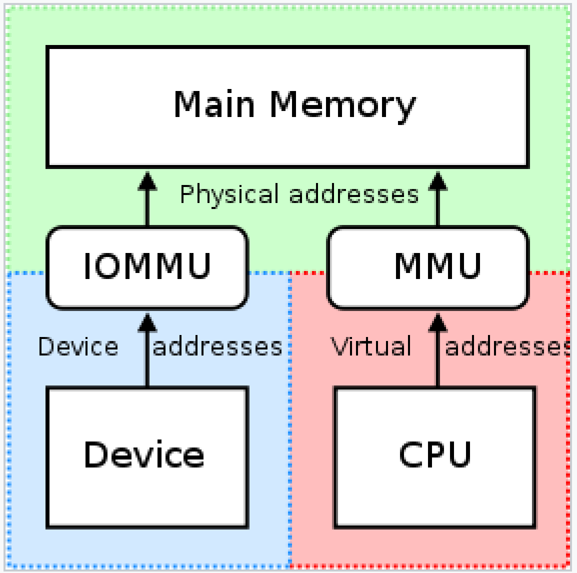
Physical Segment

- 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.
  - Data with very low locality of reference should not be put in virtual segment
- IOMMU simplification
  - Capability to this segment is send to the hardware device
  - IOMMU is just a cache for physical segment descriptor

Simplification of IOMMU with Physical Segment

- Steps for device doing DMA

1. Device sends the capability to IOMMU
2. IOMMU verifies the capability
3. Device reads from or writes to main memory with the offset between 0 to buffer size
4. IOMMU checks that the offset is within segment limit and adds segment base to the offset
5. The calculated address is then used to access main memory



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

From Wikipedia

Simplification of Paging in Guest Machine

- Traditional virtual machines need to do two virtual address translation
  - Guest virtual address to host virtual address
  - Host virtual address to host physical address
- Only one virtual address translation is needed with segmentation
  - Have guest’s capability point to host's segment

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
- Shared libraries can be quasi-statically linked
  - data/function offsets can be statically linked in link time
  - data/function segments are linked in program load time

Cross-domain Call

- It has many names: migrating thread model, protected control transfer, passive object model
- Currently thread is confined in a protection domain
- Thread should be able to travel around domains
  - Encourage modularity and improve security
  - Resource accounting can be made more accurate
- Segmentation can make cross-domain call easier to implement
  - To protect callee from caller
    - Move the stack capability from caller domain to callee domain
  - To protect caller from callee – stack protection base register
    - Callee can only access the stack above the address stored in spb register

Conclusion

- Lots of benefits we can get from single address space, capability and segmentation
- Changes
  - OS
    - Capability-based single address space
  - Compiler support
    - Far data pointer, far function call
    - New shared library implementation
  - Hardware
    - Capability and segmentation architecture

The End

Questions?