NICTA

seL4 API: Theory

14 August 2015

Adrian Danis

Australian Government

Australian Capital Territory Government · NSW Government Trade & Investment · State Government Victoria · Queensland Government

Australian National University · Deakin University · Griffith University · La Trobe University · Macquarie University · Monash University · QUT · RMIT University · Swinburne University of Technology · The University of Adelaide · The University of Melbourne · UNSW · The University of Newcastle · The University Of Queensland · University of South Australia · The University of Sydney · University of Technology Sydney · University of Canberra · UTAS · University of Western Sydney

seL4 API: Key Concepts

- Kernel Object
  - in-kernel datastruct, only directly accessible by kernel
- Capability
  - reference to a kernel object
  - allows holder to invoke functions on the objects
    - i.e. ask kernel to do something with the object
    - holder: thread invoking the cap
- Low-level interface for key activities
  - create kernel objects
  - create and manage caps in a CSpace
  - create and manage VSpace
  - create and manage threads
  - communicate between threads

Capability

- Kernel-maintained
  - user-level cannot directly access or manipulate a capability
  - capability is stored in CSpace
  - pass CSpace address of cap in system calls
- Datatypes
  - seL4_CPtr
    - index into current thread's cspace root (cnode)
    - this can be tricky….

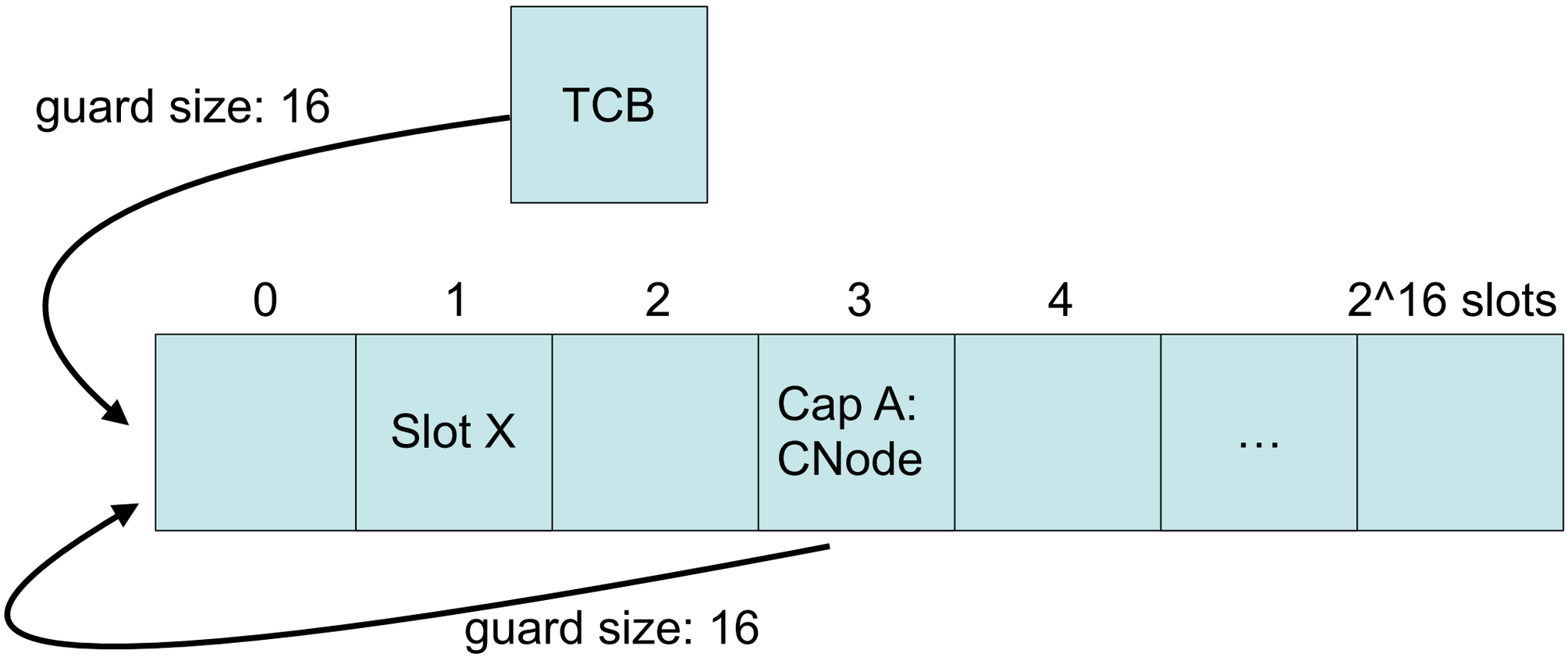
CSpace: CNode

- CNode Object
  - consists of slots in which capabilities are stored
  - can also store CNode caps in slots, creates hierarchical CSpace structure
- API
  - insert cap:
    - indirectly: through untyped retype
    - seL4_CNode_Copy
    - seL4_CNode_Mint
    - seL4_CNode_Move
  - remove cap
    - seL4_CNode_Revoke
    - seL4_CNode_Recycle
    - seL4_CNode_Delete

CSpace: Addressing

- CSpace structure
  - hierarchy of CNodes
  - CNode has: guard, size (radix)
- CSpace address:
  - refers to a specific slot in the CSpace
- Multi-part address
    - 1. root cnode (seL4_CPtr, relative to TCB cnode, 32-bit)
    - 2. index (into cnode we've derived)
    - 3. depth (bits, amount of index to use)
    - keep resolving index, following CNodes, until
      - no more CNode to follow
      - no more index to resolve
    - 4. slot (if the above refers to a CNode, then a slot in that CNode)

CSpace Addressing Example

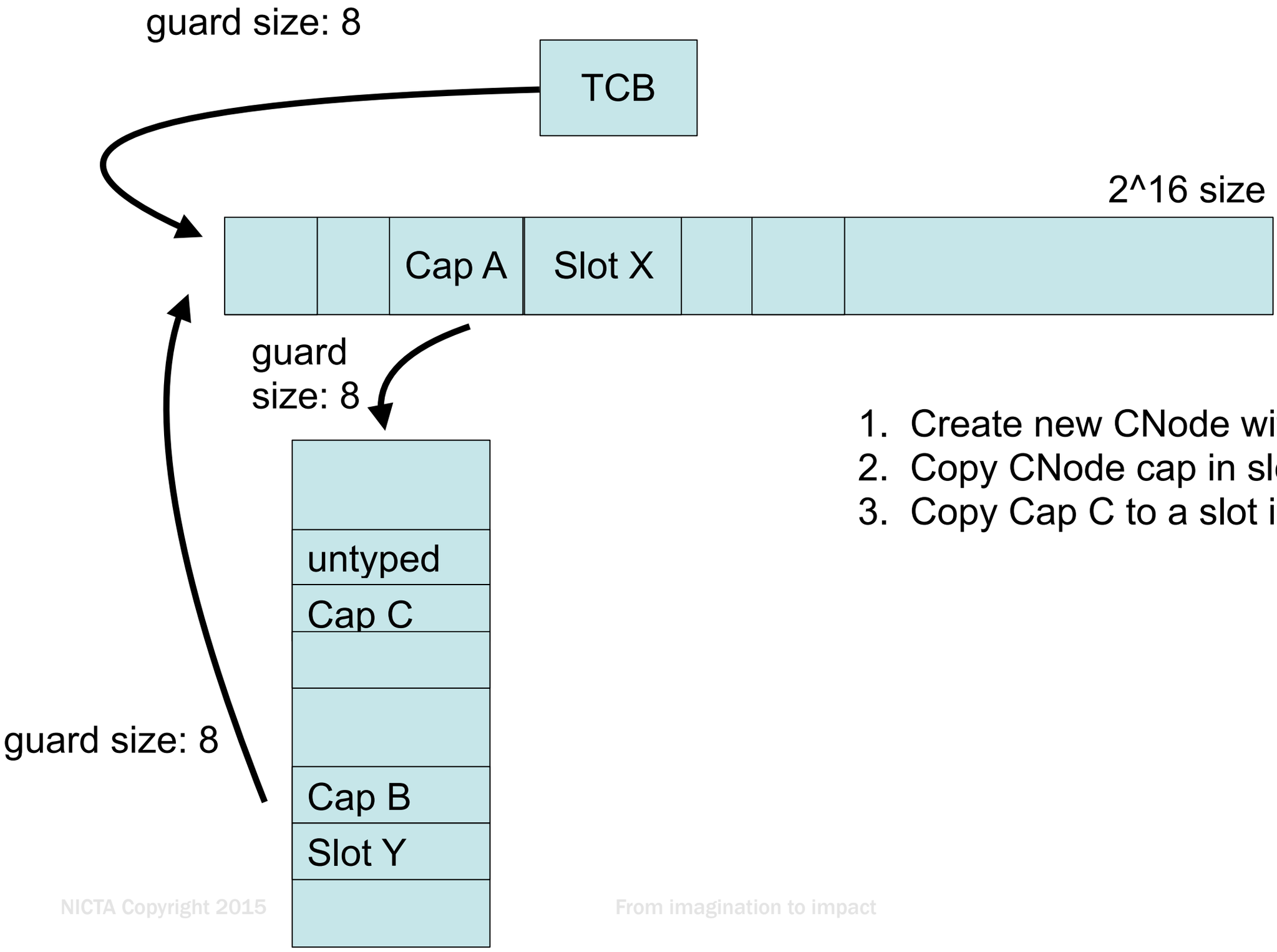


Slot X = root: 0x00000003; index 1; depth 32
0x00000003 - guard = 0x0003 = Cap A CNode
index 0x00000001 at depth 32 = Slot X

Untyped

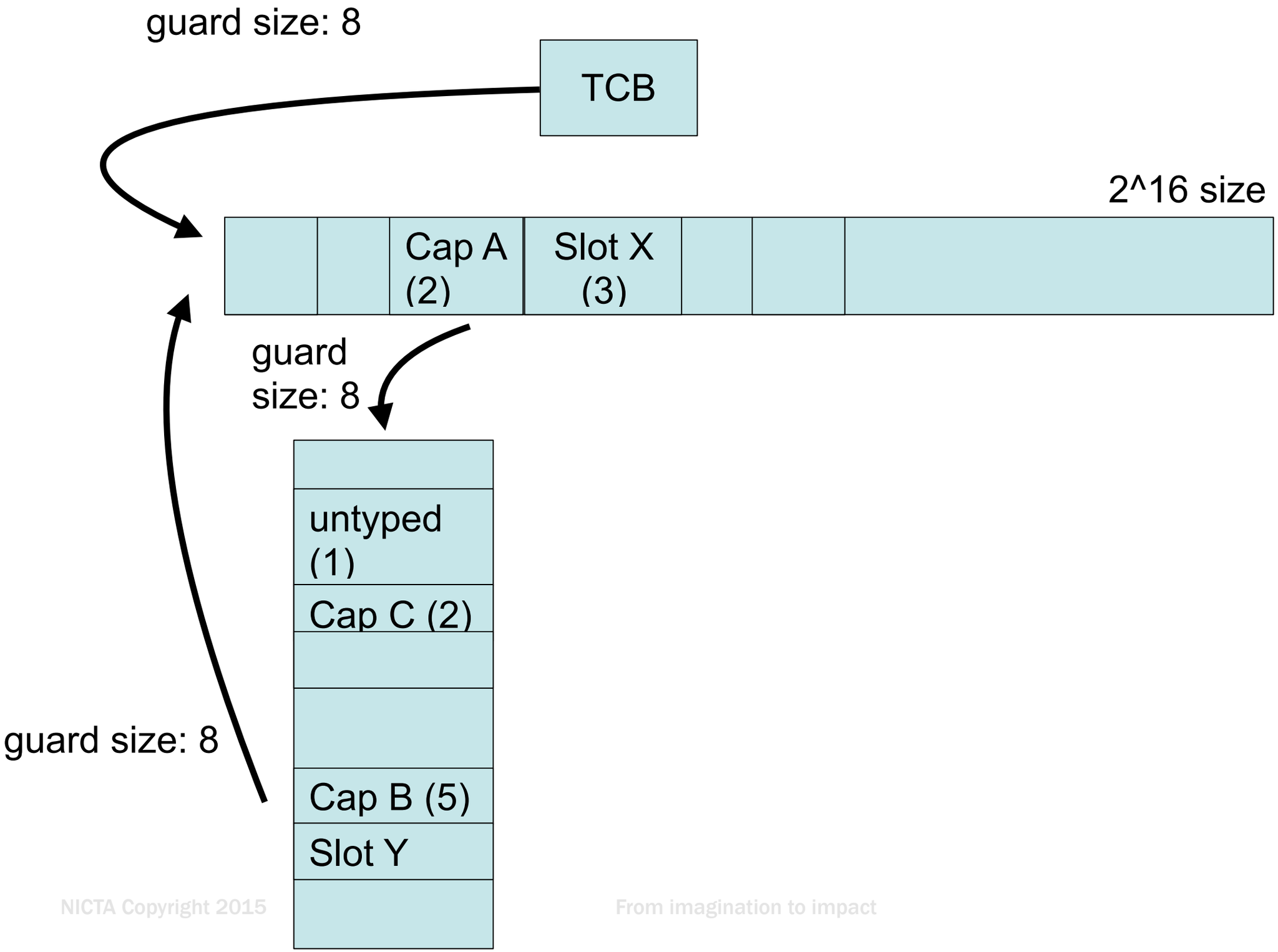
- Untyped Memory Object
  - region of (RAM) memory
  - must be retyped to another object to use it
    - results in a nested tree from a root untyped to other objects:
- Retyping
  - kernel uses part of untyped's memory region to store a new kernel object
    - can only create an object if you have a cap to a big enough untyped object
  - retype provides user with cap to the new object
- seL4_Untyped_Retype
  - seL4_Untyped_Retype(seL4_Untyped service, int type, int size_bits, seL4_CNode root, int node_index, int node_depth, int node_offset, int num_objects)

CSpace Construction Example

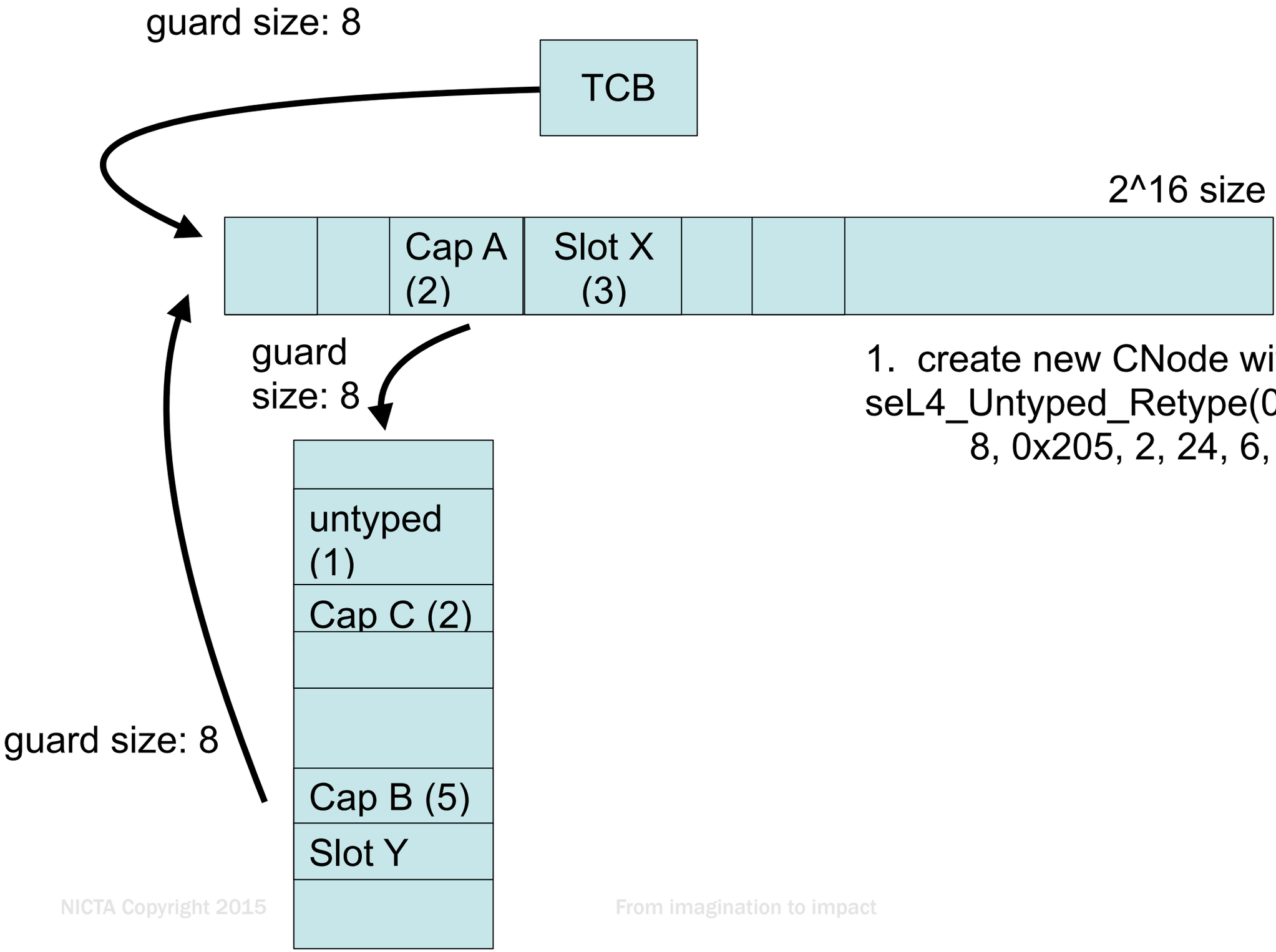


1. Create new CNode with cap in slot Y
2. Copy CNode cap in slot Y to slot X
3. Copy Cap C to a slot in the new CNode

CSpace Construction Example

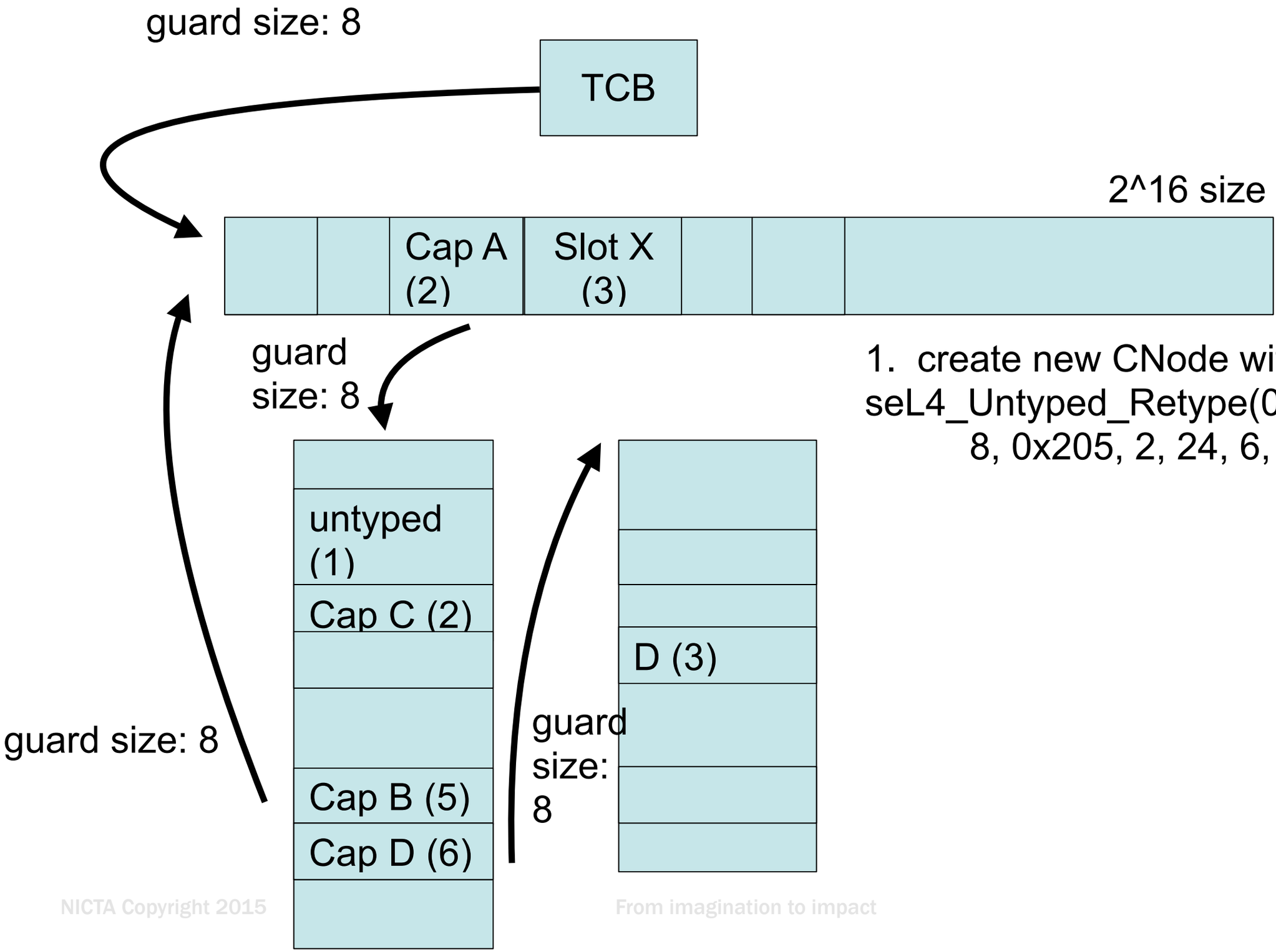


CSpace Construction Example



1. create new CNode with cap in slot Y
seL4_Untyped_Retype(0x201, cnode type,
8, 0x205, 2, 24, 6, 1);

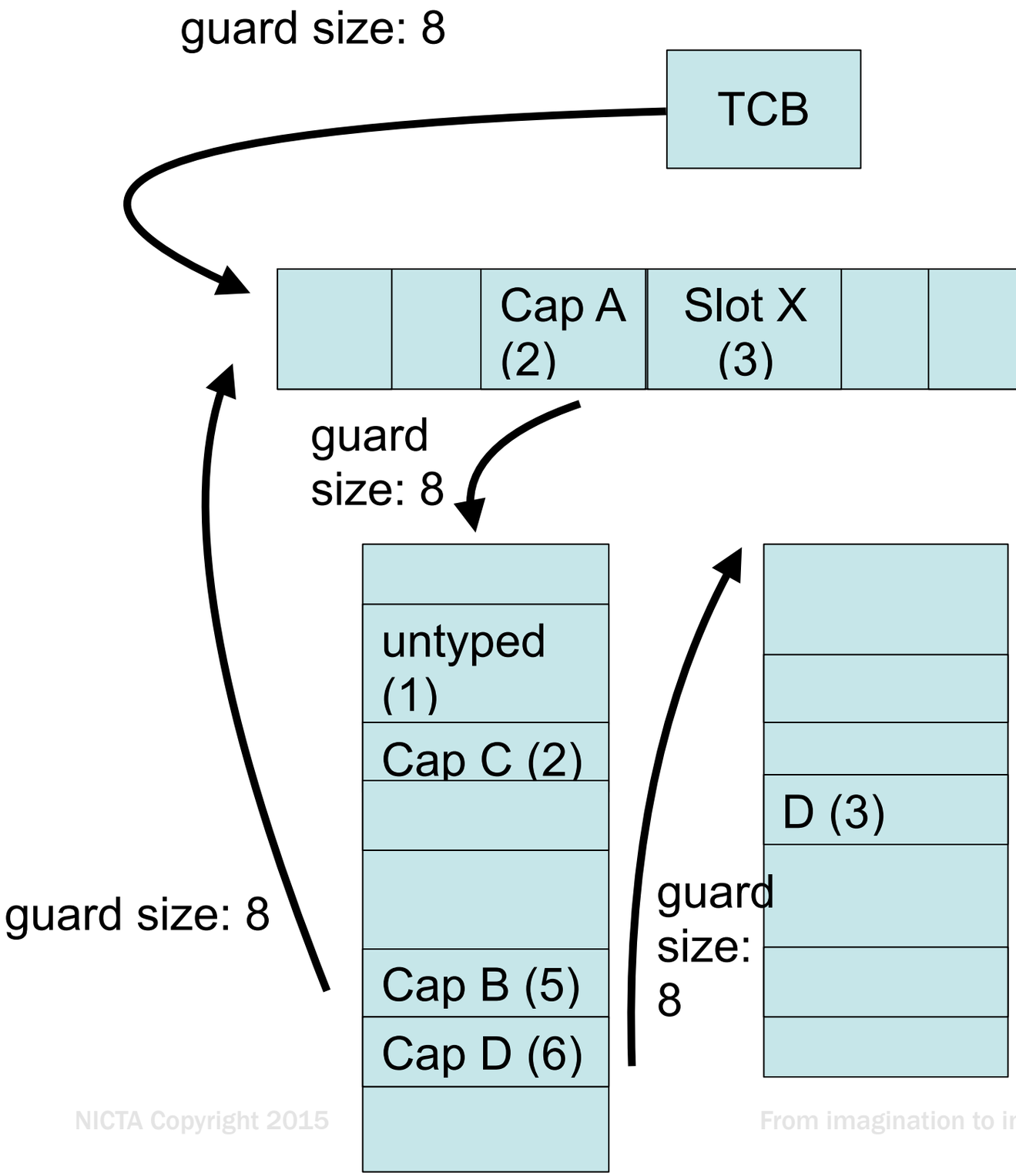
CSpace Construction Example



1. create new CNode with cap in slot Y

seL4_Untyped_Retype(0x201, cnode type, 8, 0x205, 2, 24, 6, 1);

CSpace Construction Example



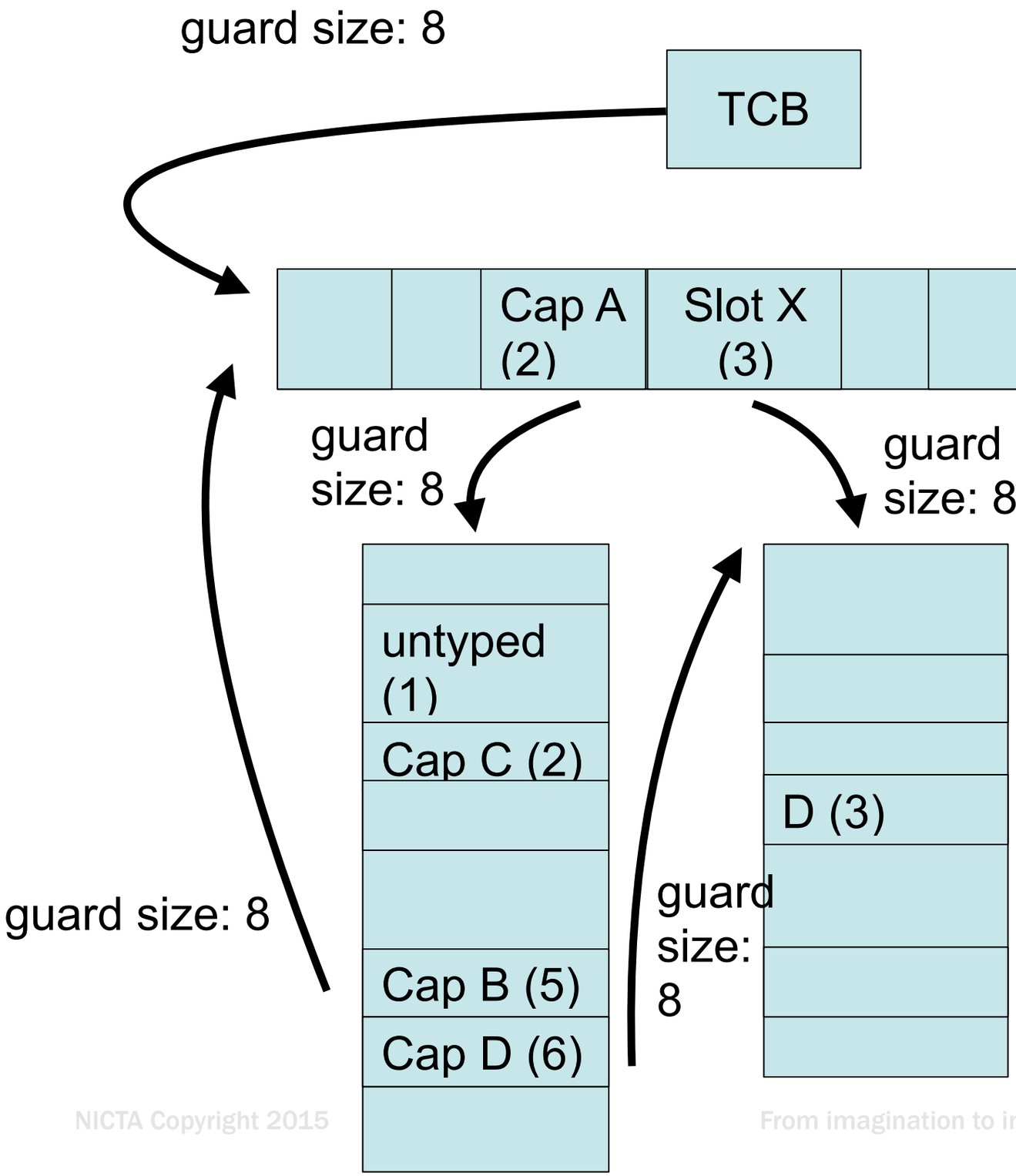
1. create new CNode with cap in slot Y

```
seL4_Untyped_Retype(0x201, cnode type,
        8, 0x205, 2, 24, 6, 1);
```

2. copy CNode cap in slot Y to slot X

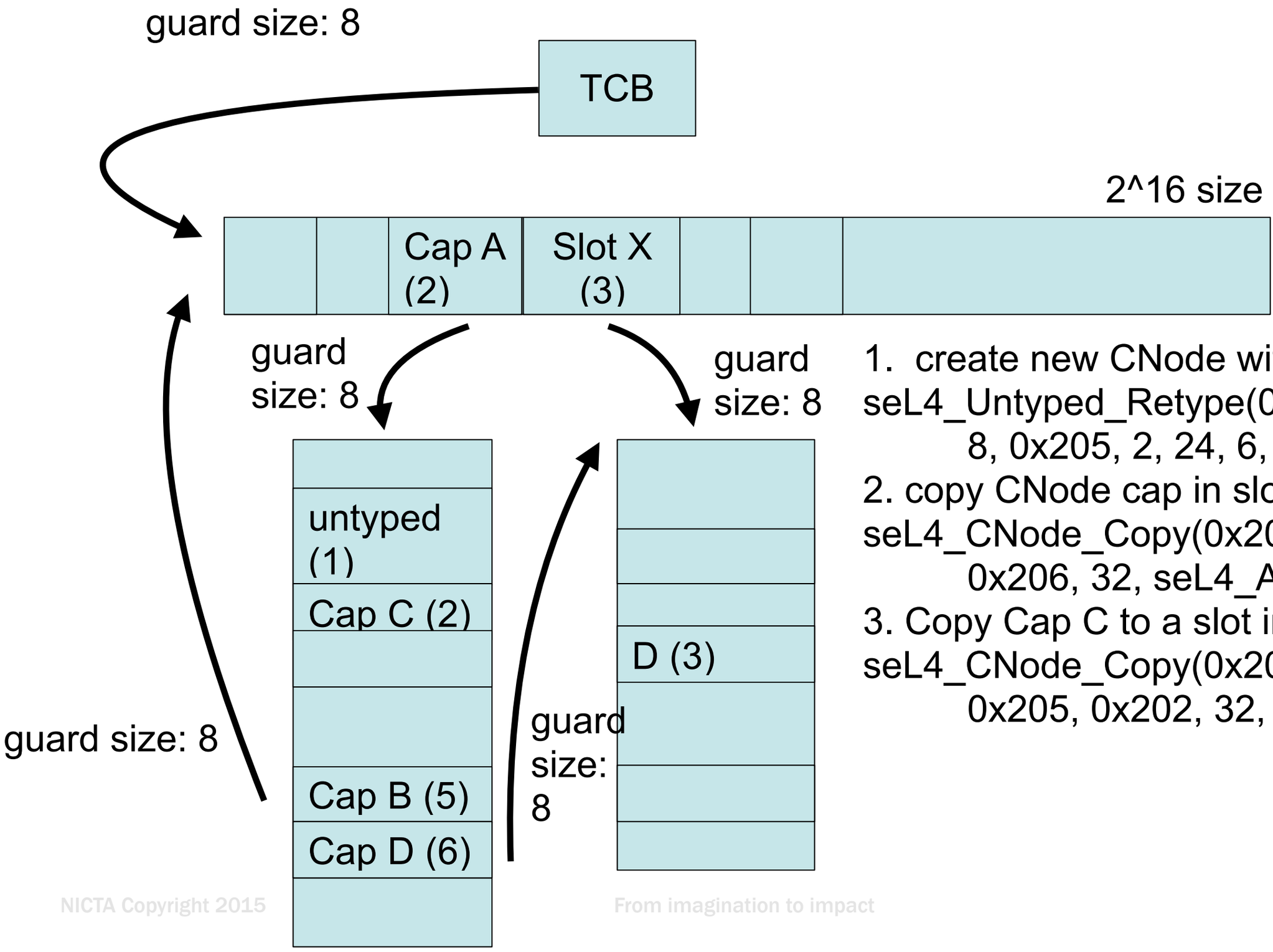
```
seL4_CNode_Copy(0x205, 3, 24, 0x205,
        0x206, 32, seL4_AllRights);
```

CSpace Construction Example



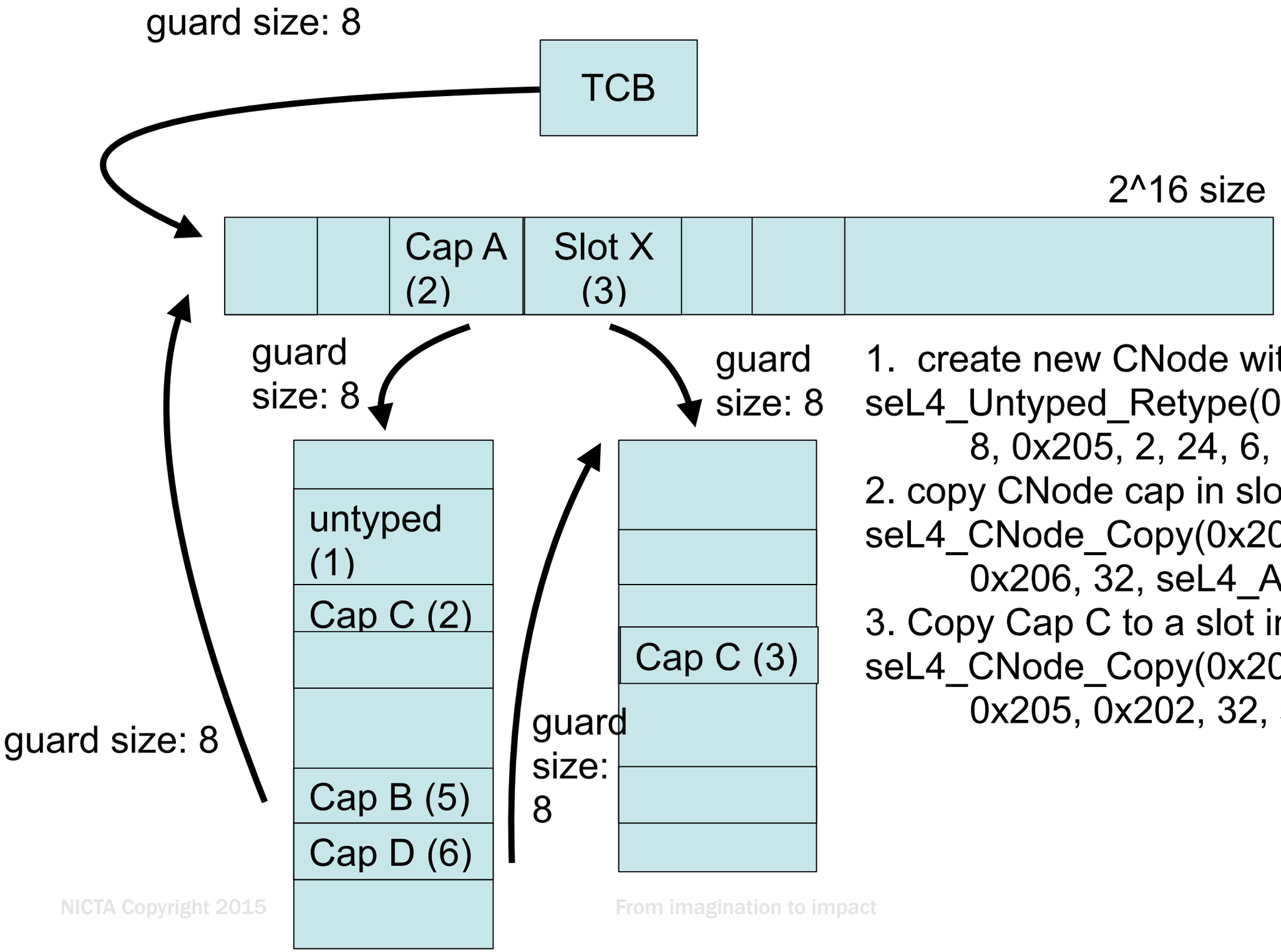
1. create new CNode with cap in slot Y
seL4_Untyped_Retype(0x201, cnode type, 8, 0x205, 2, 24, 6, 1);
2. copy CNode cap in slot Y to slot X
seL4_CNode_Copy(0x205, 3, 24, 0x205, 0x206, 32, seL4_AllRights);

CSpace Construction Example



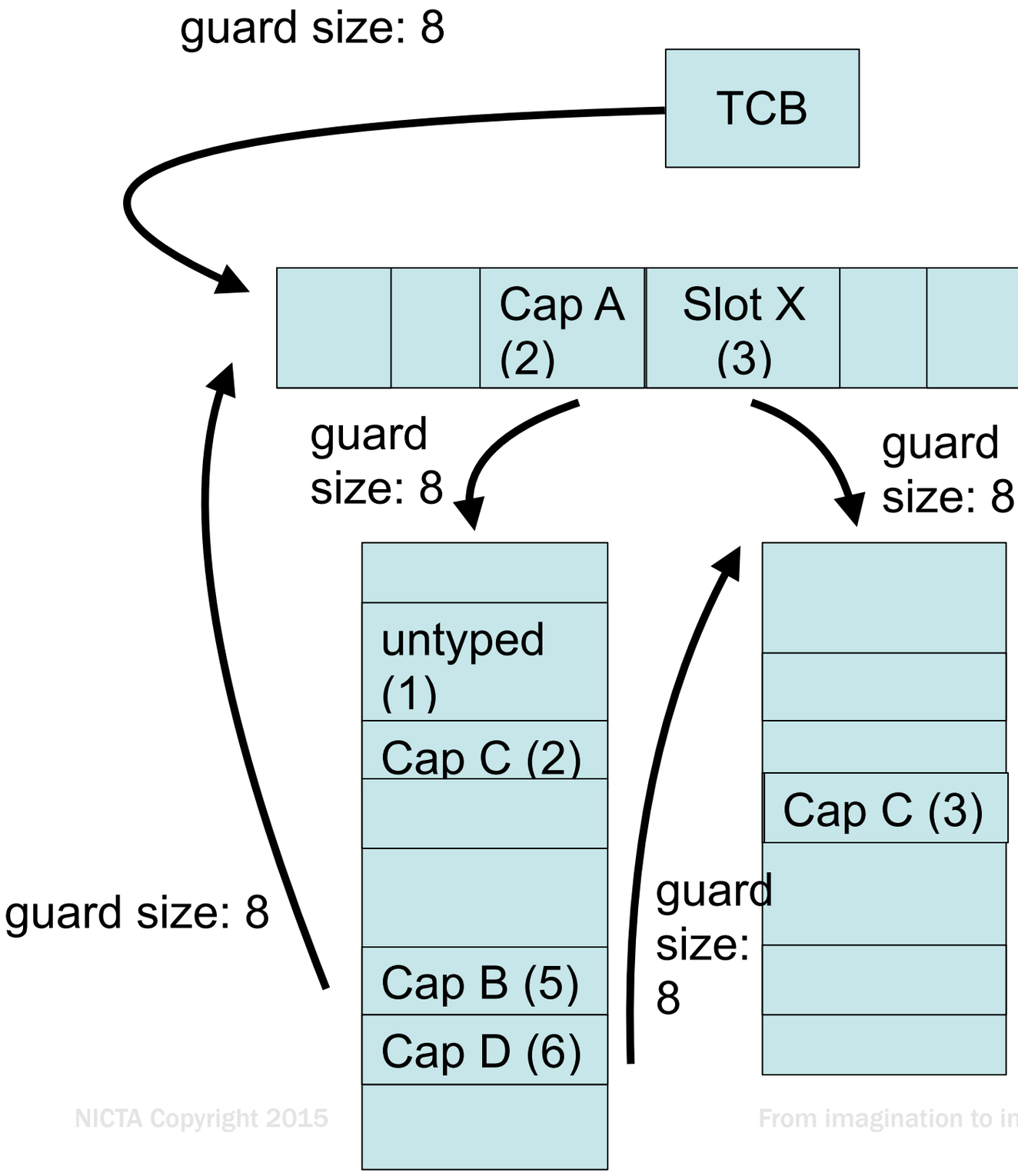
1. create new CNode with cap in slot Y
seL4_Untyped_Retype(0x201, cnode type, 8, 0x205, 2, 24, 6, 1);
2. copy CNode cap in slot Y to slot X
seL4_CNode_Copy(0x205, 3, 24, 0x205, 0x206, 32, seL4_AllRights);
3. Copy Cap C to a slot in the new CNode
seL4_CNode_Copy(0x205, 0x303, 32, 0x205, 0x202, 32, seL4_AllRights);

CSpace Construction Example



1. create new CNode with cap in slot Y
seL4_Untyped_Retype(0x201, cnode type, 8, 0x205, 2, 24, 6, 1);
2. copy CNode cap in slot Y to slot X
seL4_CNode_Copy(0x205, 3, 24, 0x205, 0x206, 32, seL4_AllRights);
3. Copy Cap C to a slot in the new CNode
seL4_CNode_Copy(0x205, 0x303, 32, 0x205, 0x202, 32, seL4_AllRights);

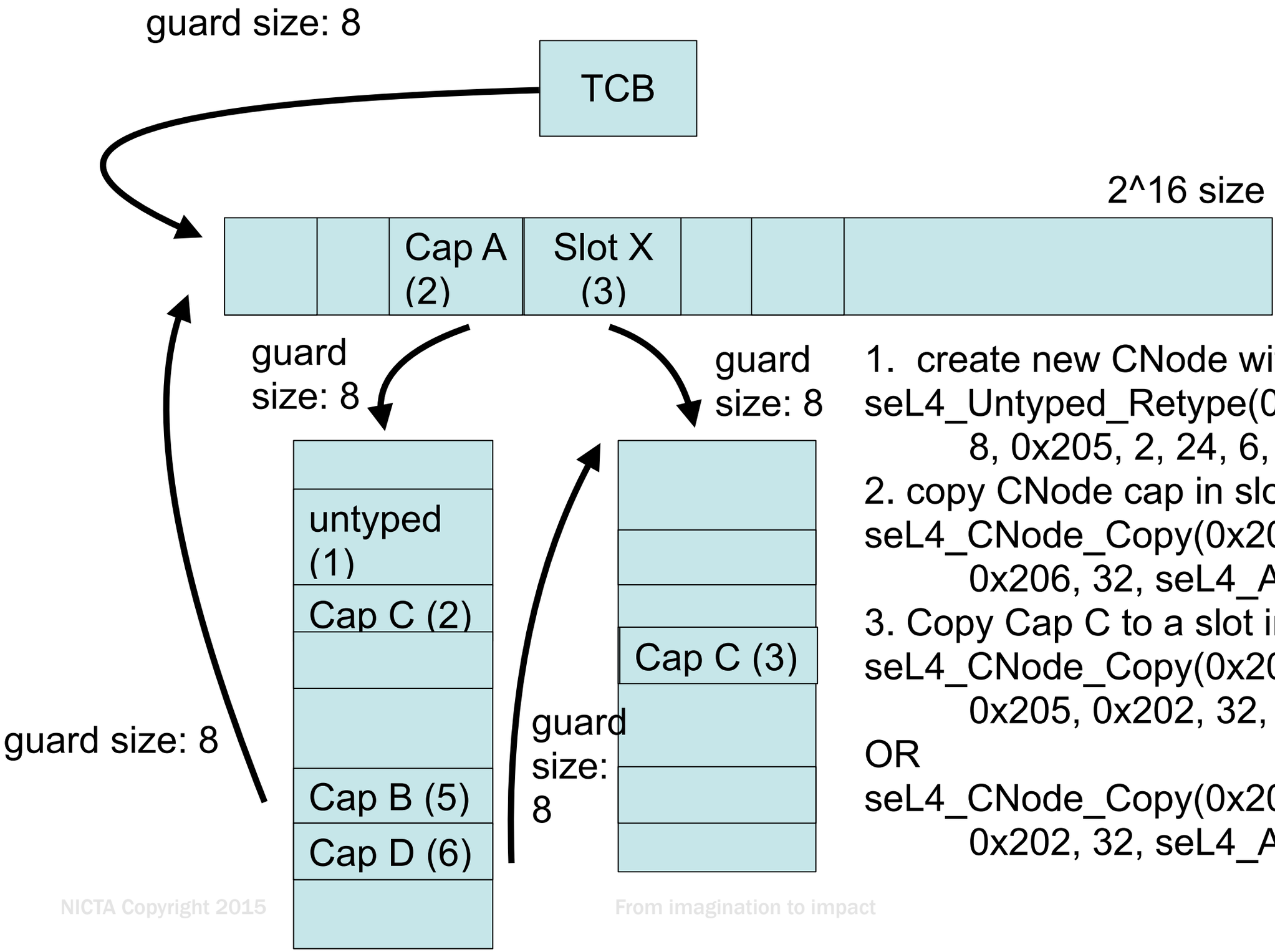
CSpace Construction Example



1. create new CNode with cap in slot Y
seL4_Untyped_Retype(0x201, cnode type, 8, 0x205, 2, 24, 6, 1);
2. copy CNode cap in slot Y to slot X
seL4_CNode_Copy(0x205, 3, 24, 0x205, 0x206, 32, seL4_AllRights);
3. Copy Cap C to a slot in the new CNode
seL4_CNode_Copy(0x205, 0x303, 32, 0x205, 0x202, 32, seL4_AllRights);

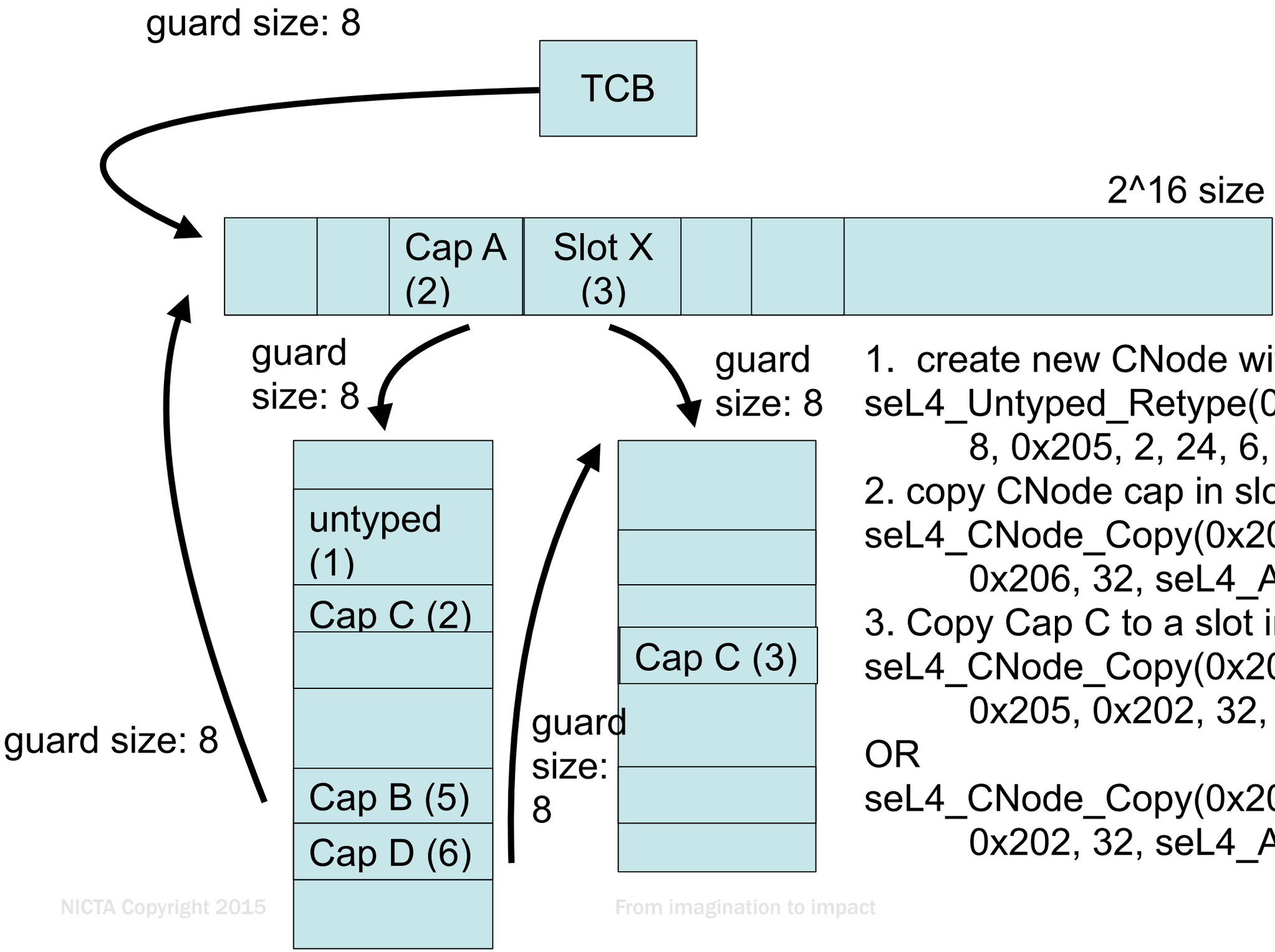
OR

CSpace Construction Example



1. create new CNode with cap in slot Y
seL4_Untyped_Retype(0x201, cnode type, 8, 0x205, 2, 24, 6, 1);
2. copy CNode cap in slot Y to slot X
seL4_CNode_Copy(0x205, 3, 24, 0x205, 0x206, 32, seL4_AllRights);
3. Copy Cap C to a slot in the new CNode
seL4_CNode_Copy(0x205, 0x303, 32, 0x205, 0x202, 32, seL4_AllRights);
OR
seL4_CNode_Copy(0x206, 3, 8, 0x205, 0x202, 32, seL4_AllRights);

CSpace Construction Example



1. create new CNode with cap in slot Y
seL4_Untyped_Retype(0x201, cnode type, 8, 0x205, 2, 24, 6, 1);
2. copy CNode cap in slot Y to slot X
seL4_CNode_Copy(0x205, 3, 24, 0x205, 0x206, 32, seL4_AllRights);
3. Copy Cap C to a slot in the new CNode
seL4_CNode_Copy(0x205, 0x303, 32, 0x205, 0x202, 32, seL4_AllRights);
OR
seL4_CNode_Copy(0x206, 3, 8, 0x205, 0x202, 32, seL4_AllRights);

VSpace: PageDir, PageTable, Frame

NICTA

- VSpace:
  - represents mapping: virtual address → physical address
    - i.e. abstraction of CPU page table
  - VSpace-related objects are platform-specific
- Sizes (ARM)
  - PD: 16KiB, 4 byte slots
  - PT: 1KiB, 4 byte slots
  - Frame: 4KiB, 64KiB, 1MiB, 16MiB
- Size (x86)
  - PD: 4KiB, 4 byte slots
  - PT: 4KiB, 4 byte slots
  - Frame: 4KiB, 4MiB

VSpace API

- PD
  - seL4_ARCH_PageTable_Map
  - seL4_ARCH_PageTable_Unmap
  - seL4_ARCH_Page_Map - for large frames
  - seL4_ARCH_Page_Unmap - for large frames
- PT
  - seL4_ARCH_Page_Map
  - seL4_ARCH_Page_Unmap
- Note:
  - ARCH is either ARM or IA32

TCB: Thread Control Block

- TCB Object:
  - kernel's representation of a thread.
  - contains:
    - Caps: CSpace, VSpace, IPC Buffer Frame
    - Other: IP (instruction pointer), SP (stack pointer), IPC Buffer, Priority
- IPC Buffer
  - buffer used to pass data during IPC
    - 512 byte object, must be wholly in one frame
    - also used for all syscalls
  - passed to TCB as:
    - index to Frame cap in TCB's CSpace
    - address where it is mapped in TCB's VSpace

TCB: API

- Configure
  - seL4_TCB_Configure
- Write Registers
  - seL4_TCB_ReadRegisters
  - seL4_TCB_WriteRegisters
- Resume
  - seL4_TCB_Resume
- Suspend
  - seL4_TCB_Suspend

Inter-Process Communication

- Synchronous Endpoint Object
  - enables synchronous (blocking) communication
  - communicating threads must hold caps to same endpoint
- Endpoint Caps
  - master cap (receiver), derived caps (senders)
  - ***badge***: identifies specific sender cap
  - ***reply cap***: temporary cap allows receiver to reply to sender for two-way communication

Endpoint API

- Sending and Receiving
  - seL4_Send
  - seL4_Wait
  - seL4_Call
  - seL4_ReplyWait
  - seL4_NBSend
- Message Registers
  - seL4_GetMR
  - seL4_SetMR
  - seL4_GetCap
  - seL4_SetCap

Notification

- Asynchronous Endpoint Object
  - allows one thread to send a notification to another
  - notification: asynchronous (non-blocking) message
  - sends limited data (32-bit word)
- AsyncEndpoint Caps
  - master cap (receiver), derived caps (senders)
- API
  - seL4_Notify
  - seL4_Wait

BootInfo: Start-up Information

- On startup, kernel creates:
  - root task CSpace
  - root task VSpace
  - frames for device memory
  - untyped caps for RAM memory
- All startup objects are available to root task
  - kernel places caps to these objects in root task CSpace
  - kernel needs to tell root task
    - what caps it has
    - what the objects are
- Bootinfo
  - info about all the initial objects and the caps to them

Initial Caps

- Some Initial Caps
  - seL4_CapNull = 0, /* null cap */
  - seL4_CapInitThreadTCB = 1, /* initial thread's TCB cap */
  - seL4_CapInitThreadCNode = 2, /* root CNode cap */
  - seL4_CapInitThreadVSpace = 3, /* VSpace cap */
  - seL4_CapBootInfoFrame = 9, /* bootinfo frame cap */
  - seL4_CapInitThreadIPCBuffer = 10, /* initial thread's IPC buffer frame cap */

Bootinfo contents

- seL4_Bootinfo struct
  - seL4_IPCBuffer* ipcBuffer;   /* pointer to initial thread's IPC buffer */
  - seL4_SlotRegion empty;   /* empty slots (null caps) */
  - seL4_SlotRegion userImageFrames; /* userland-image frame caps */
  - seL4_SlotRegion userImagePDs;   /* userland-image PD caps */
  - seL4_SlotRegion userImagePTs;   /* userland-image PT caps */
  - seL4_SlotRegion untyped;   /* untyped-object caps (untyped caps) */
  - seL4_Word untypedPaddrList /* physical address of each untyped cap */
  - seL4_Uint8 untypedSizeBitsList; /* size (2^n) bytes of each untyped cap */
  - seL4_Uint8 initThreadCNodeSizeBits; /* root CNode size (2^n slots) */
  - seL4_Word numDeviceRegions; /* number of device regions */
  - seL4_DeviceRegion deviceRegions /* device regions */
- seL4_GetBootInfo