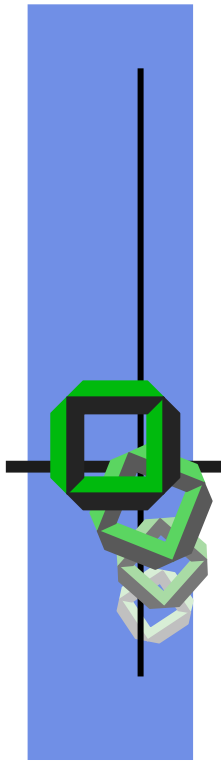
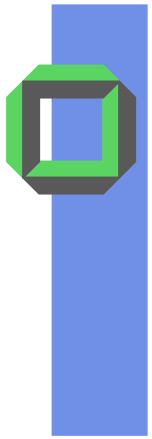


µ-Kernel Construction (12)

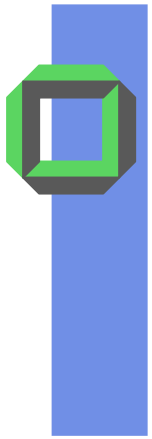
Review



Threading

- Thread state must be saved/restored on thread switch
- We need a Thread Control Block (TCB) per thread
- TCBs must be kernel objects
  - TCBs implement threads
- We need to find
  - Any thread's TCB using its global ID
  - The currently executing thread's TCB (per processor)

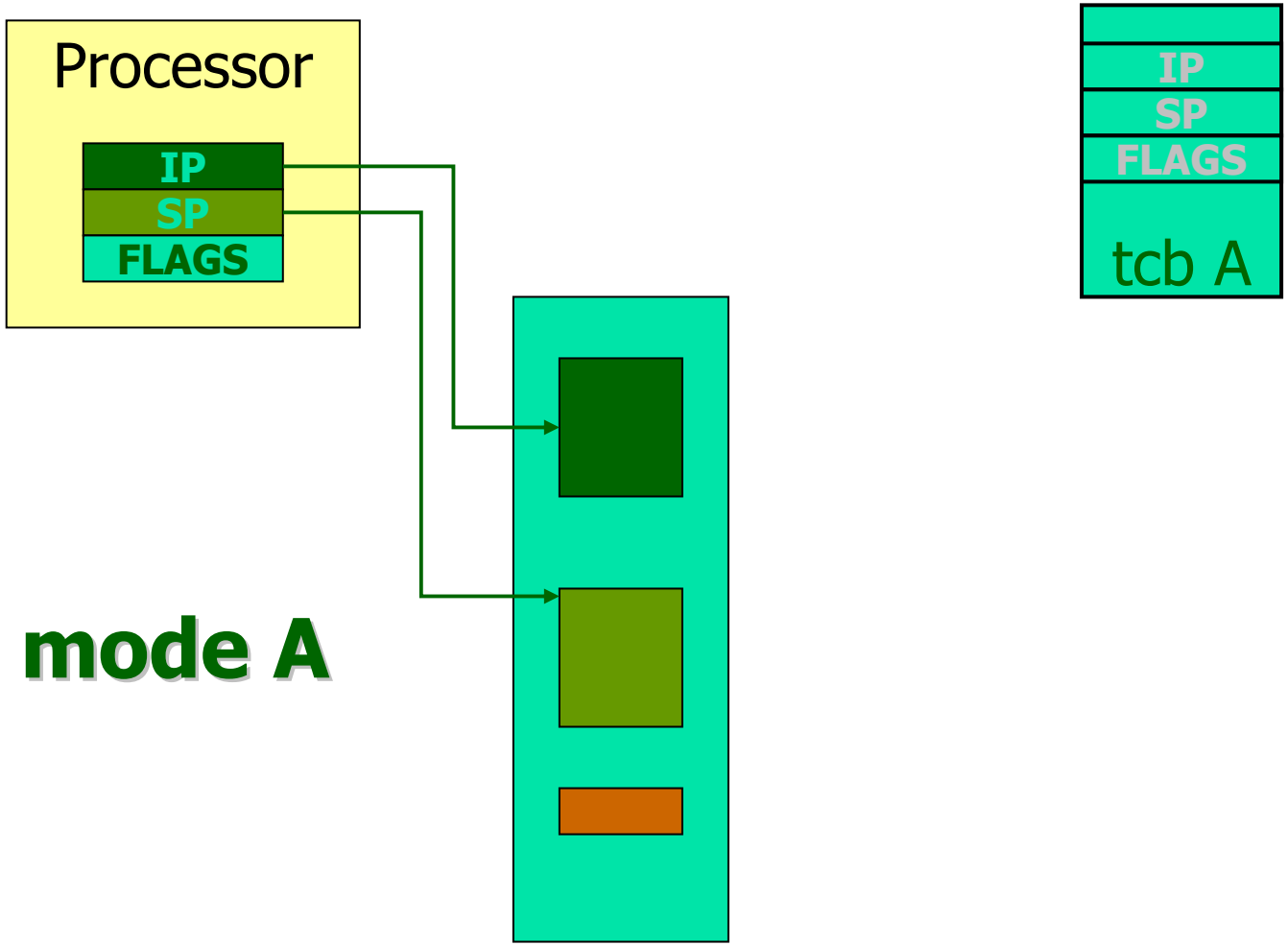
At least partially. We have found some good reasons to implement parts of the TCB in user memory.



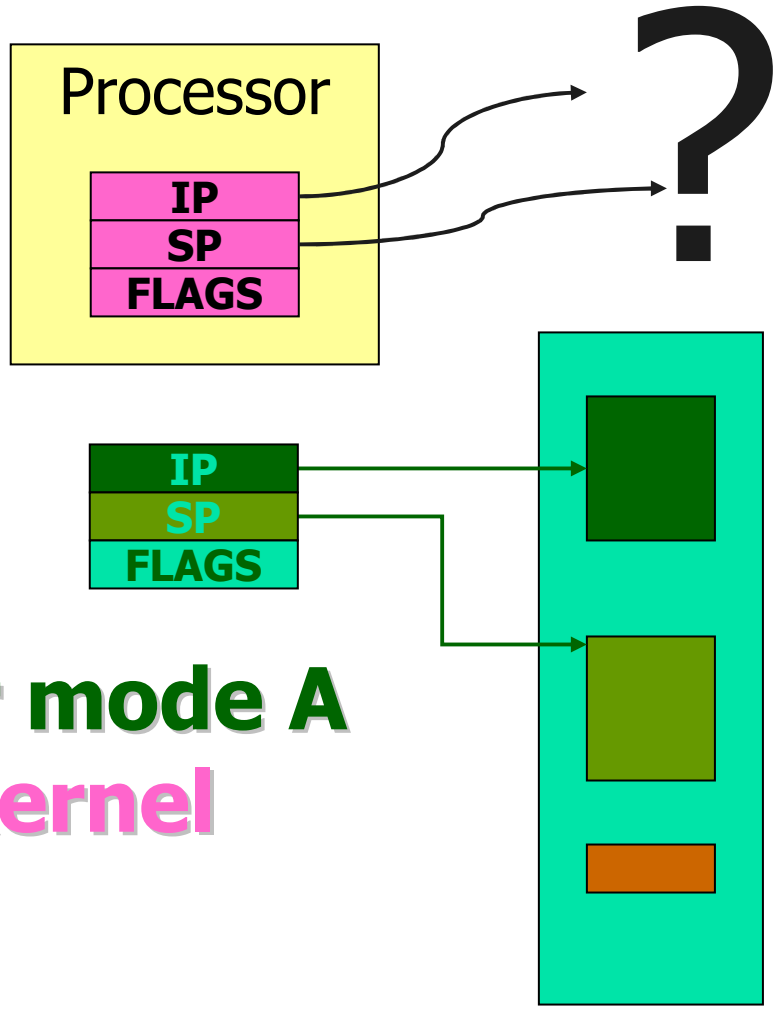
Thread Switch A → B

- Thread A is running in user mode
- Thread A experiences an end-of-time-slice or is preempted by a (device) interrupt
- We enter kernel mode
- The microkernel saves the status of thread A on A's TCB
- The microkernel loads the status of thread B from B's TCB
- We leave kernel mode
- Thread B is running in user mode

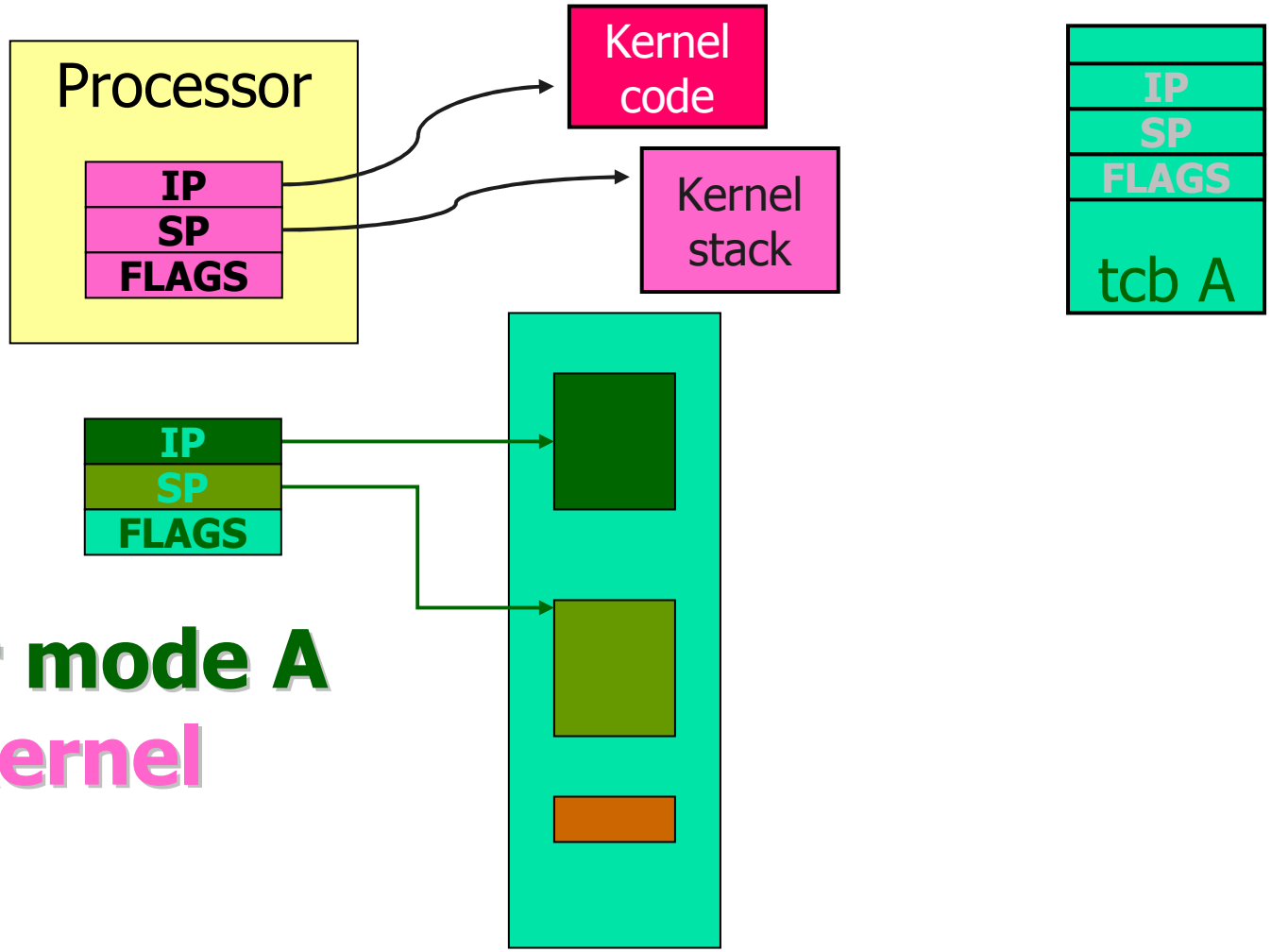
Thread Switch A → kernel → B



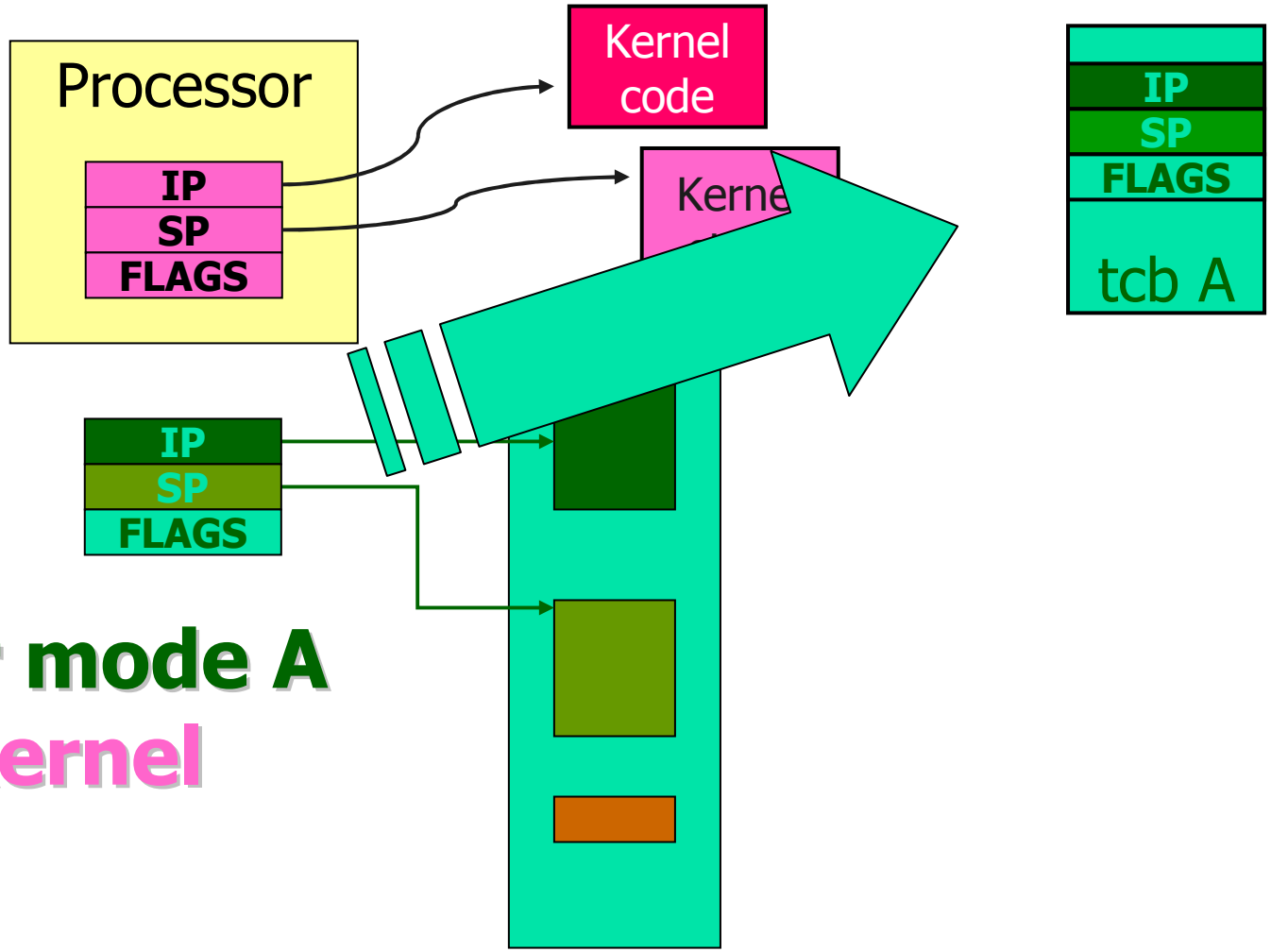
Thread Switch A → kernel → B



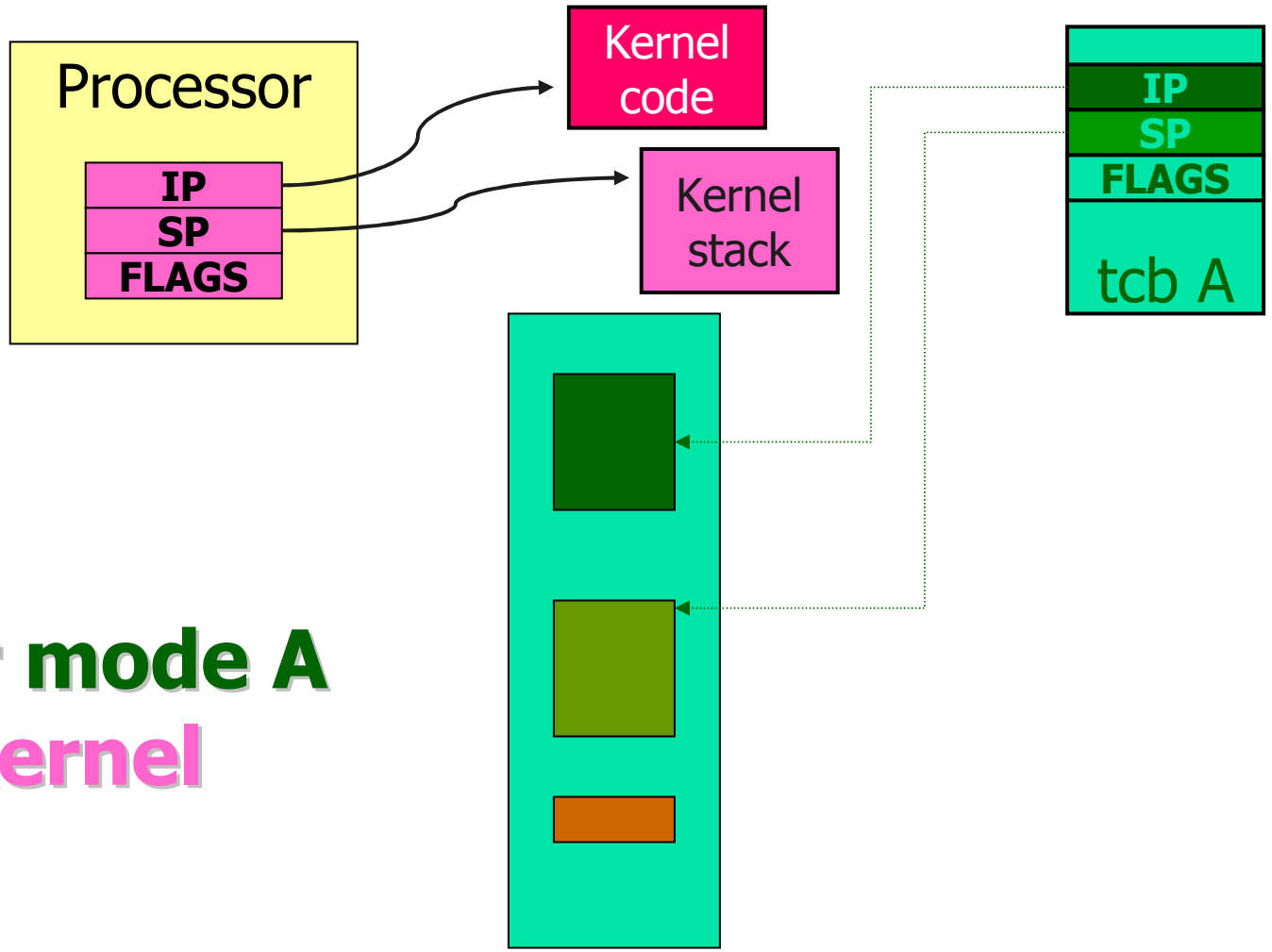
Thread Switch A → kernel → B



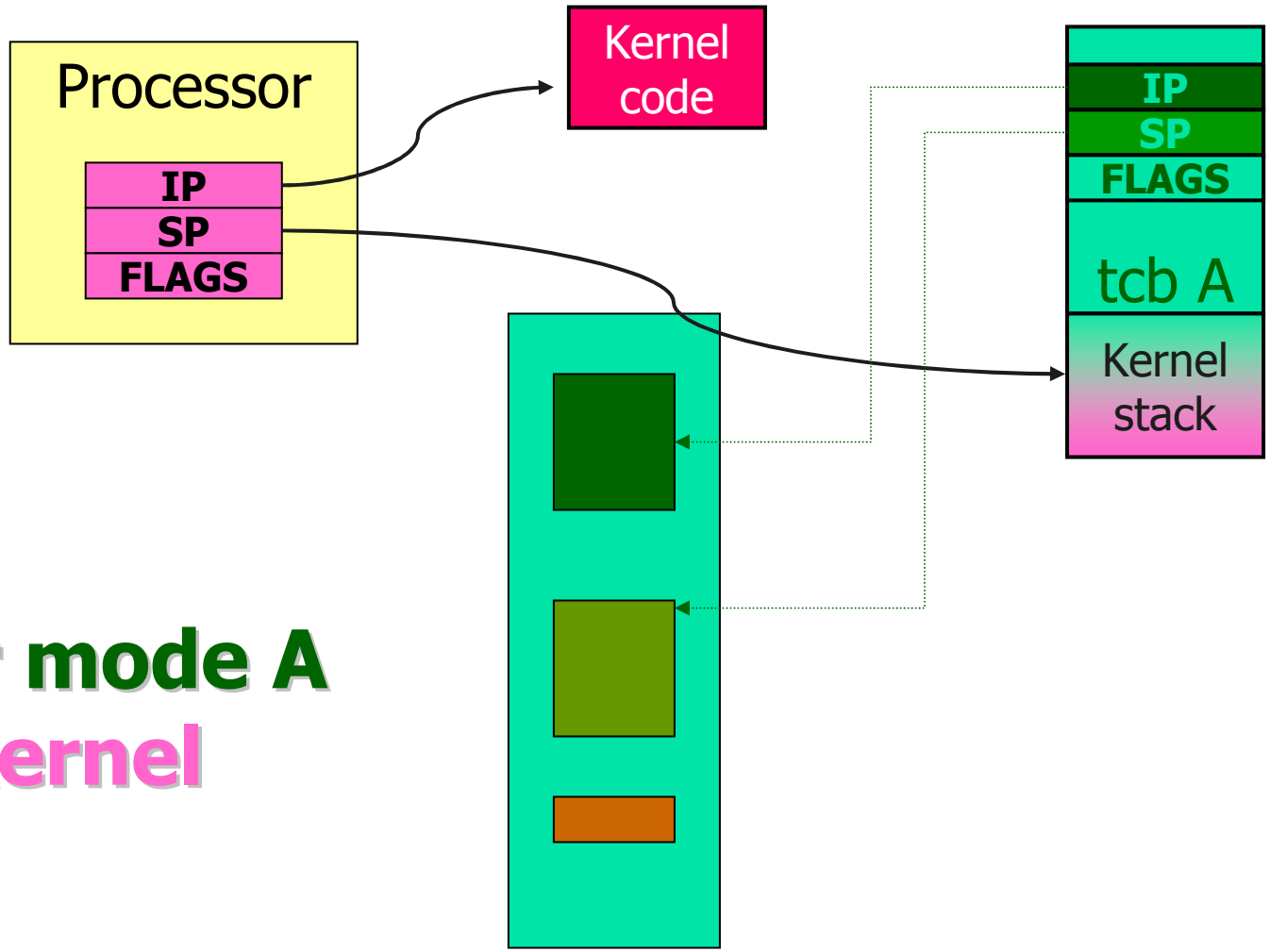
Thread Switch A → kernel → B



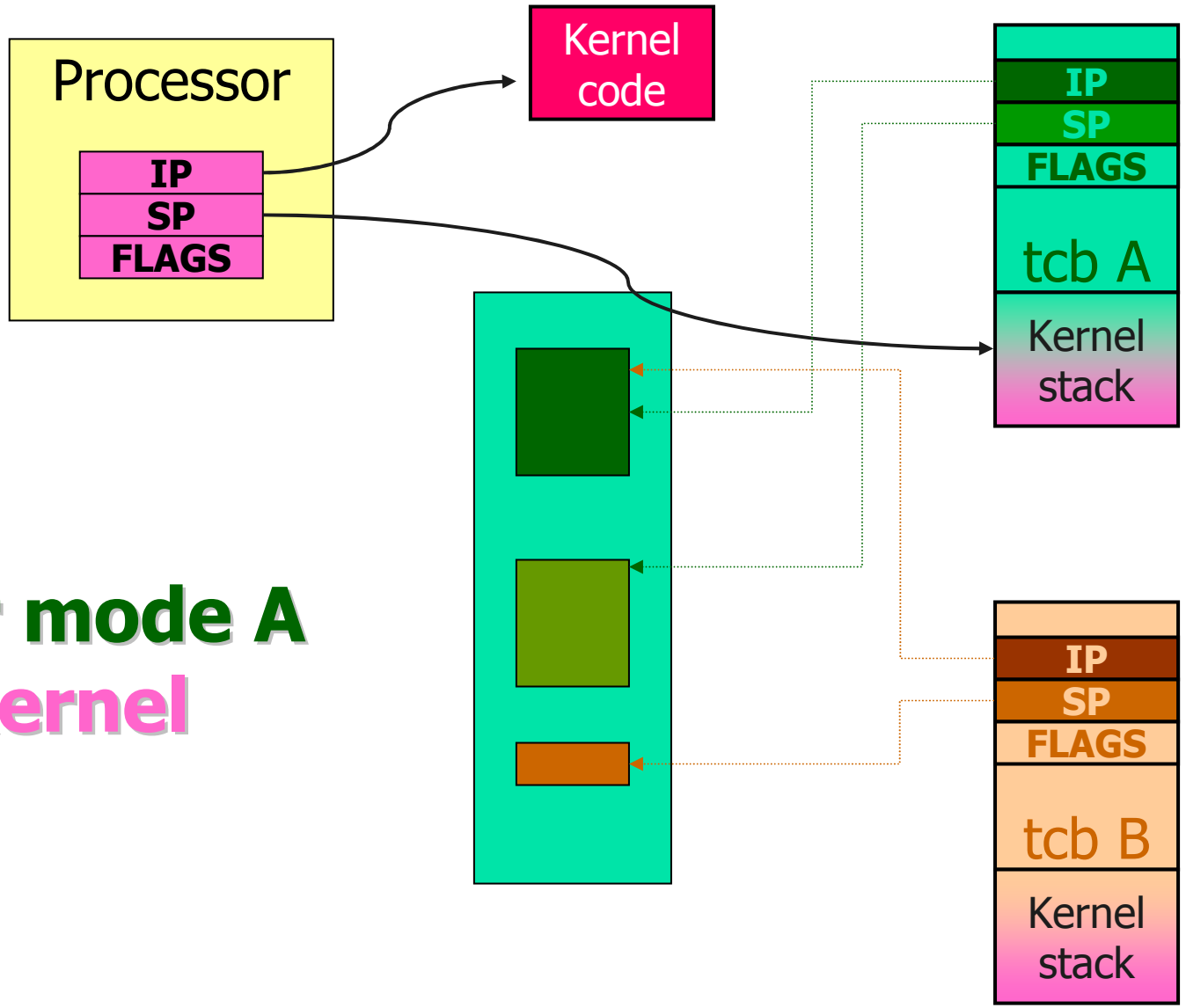
Thread Switch A → kernel → B



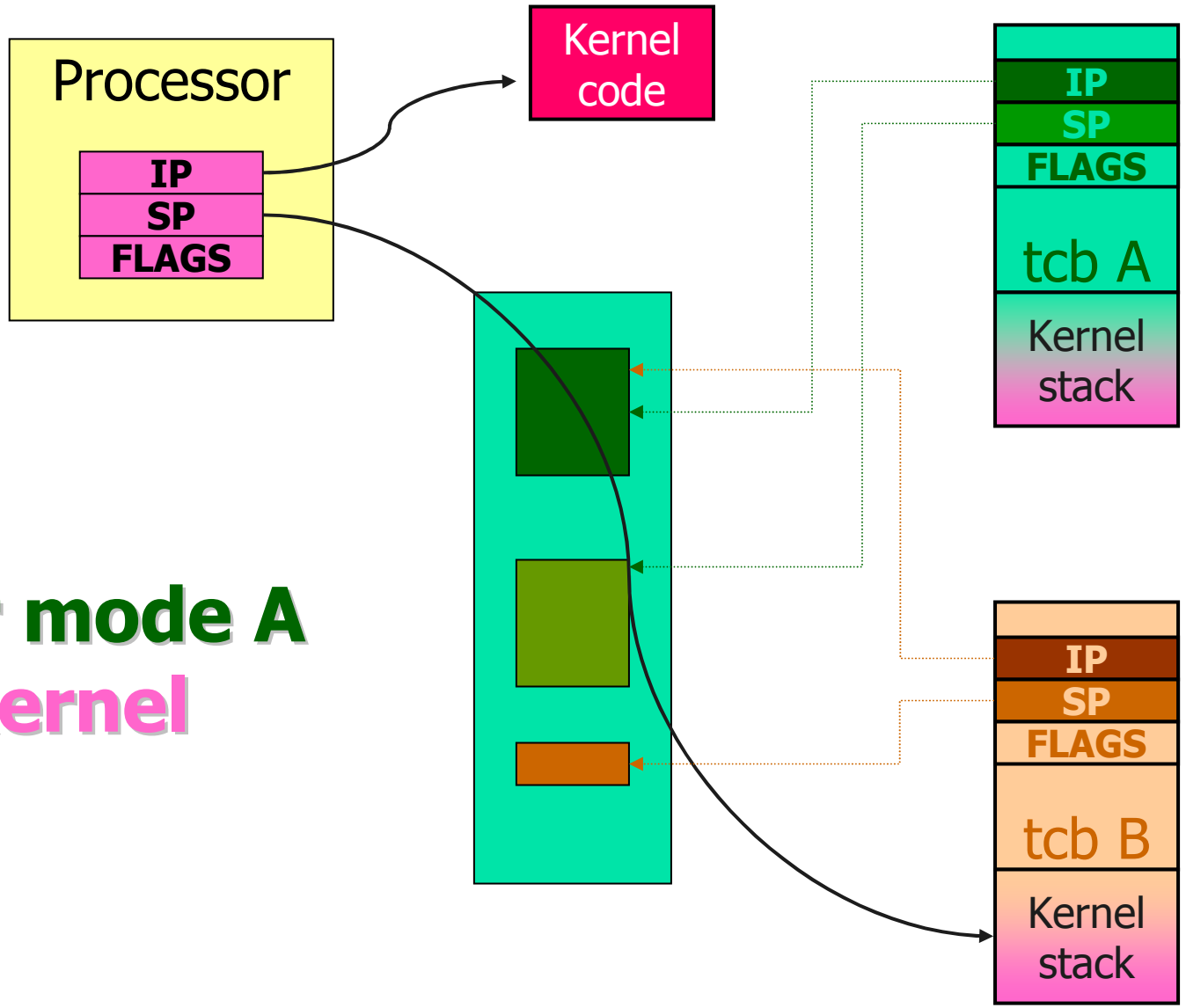
Thread Switch A → kernel → B



Thread Switch A → kernel → B



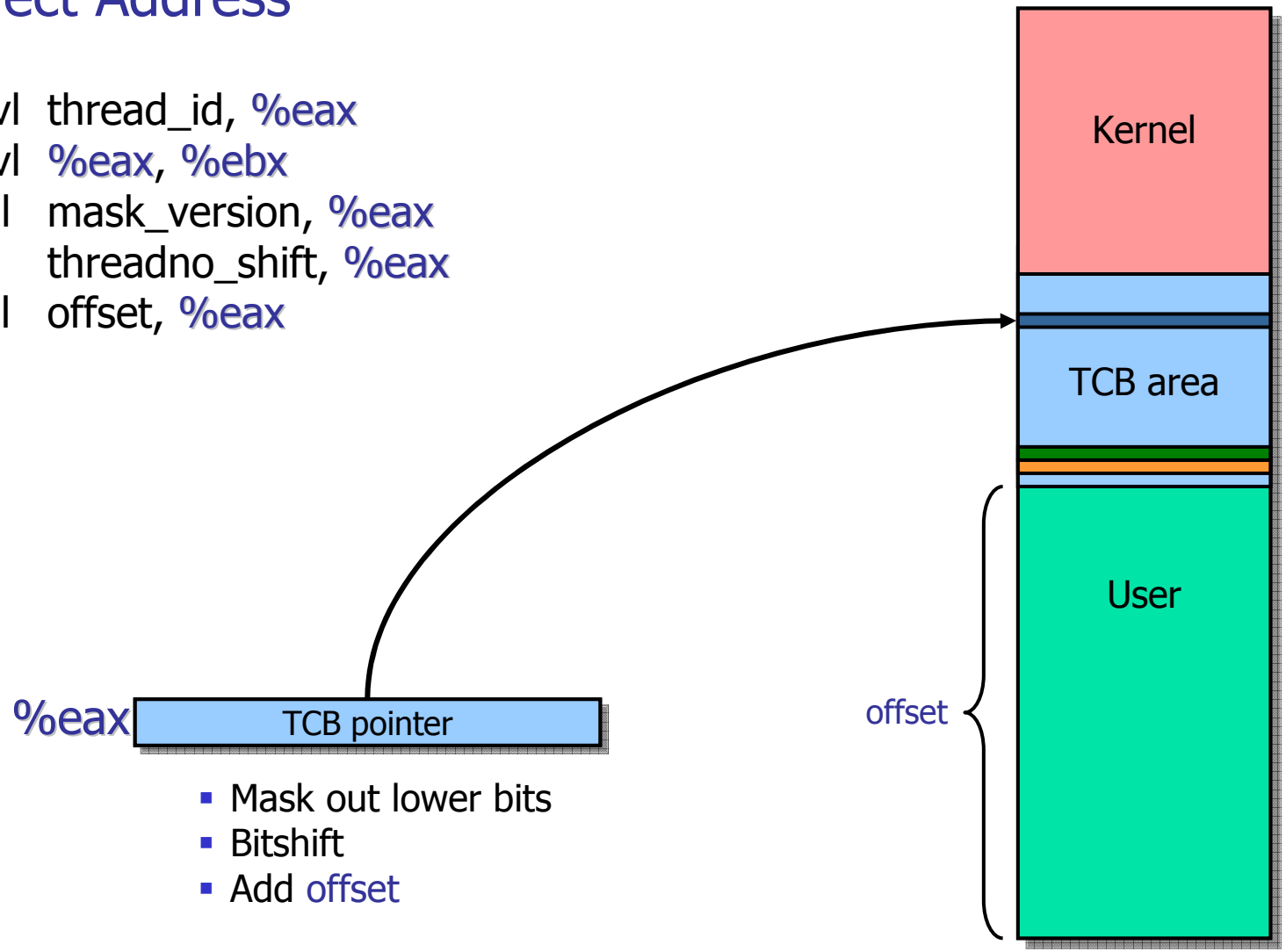
Thread Switch A → kernel → B



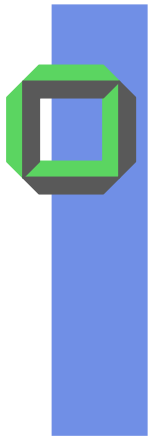
Thread ID → TCB

Direct Address

```
movl  thread_id, %eax
movl  %eax, %ebx
andl  mask_version, %eax
shrl  threadno_shift, %eax
addl  offset, %eax
```

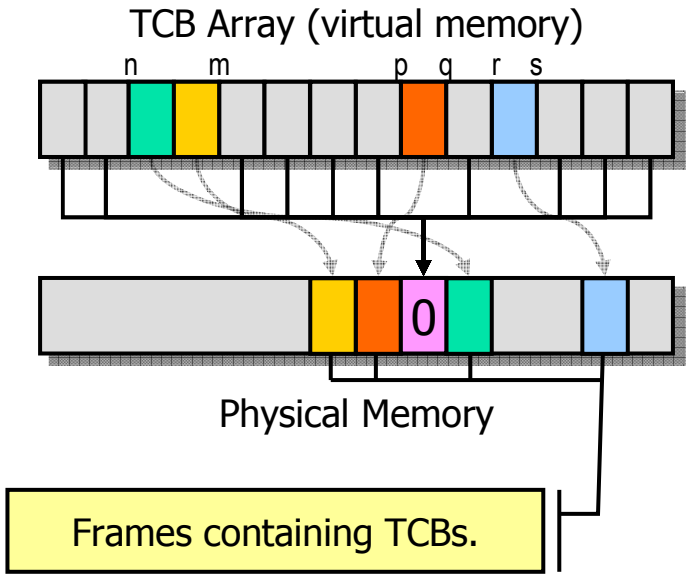


- Mask out lower bits
- Bitshift
- Add offset



0-Mapping Trick
Direct Addressing

- Allocate physical memory for TCBs on demand
  - Dependent on the max number of allocated TCBs
- Map all remaining TCBs to a 0-filled read-only page
  - Any access to unused threads will result in "invalid thread ID" (0)
  - Avoids additional check



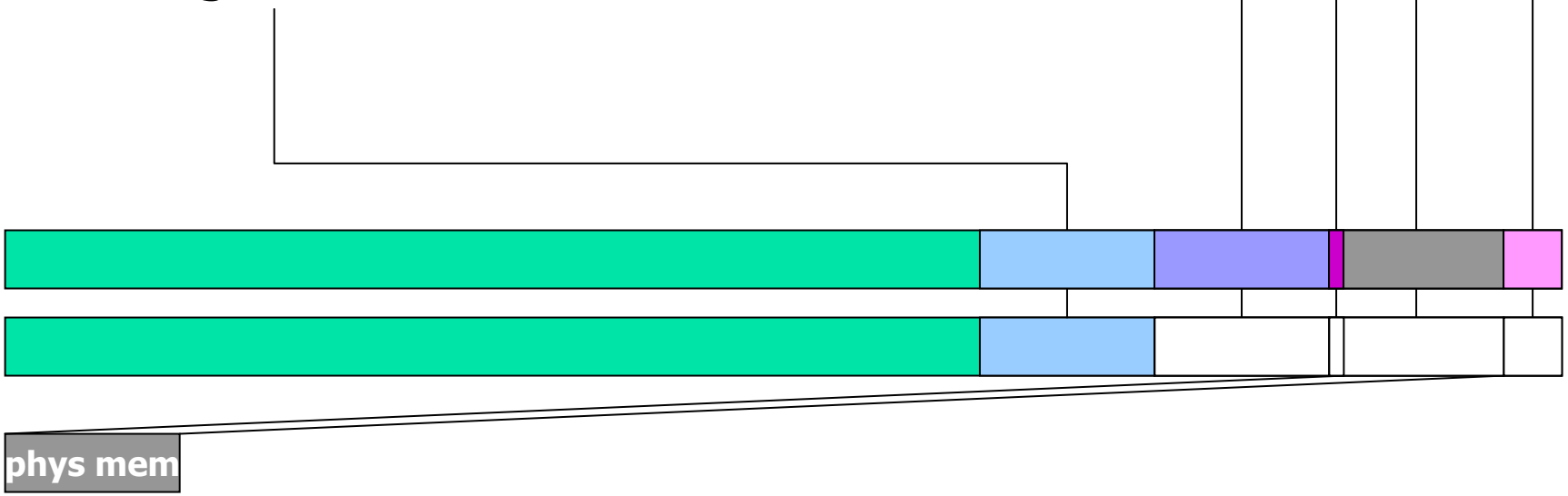
- Virtual TCB array requires ≥ 256 MB virtual memory for 256k potential TCBs

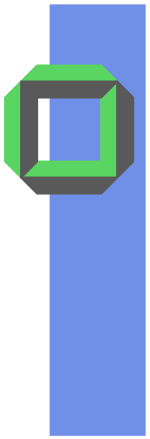
Basic Address-Space Layout

Address-Space Layout

32bits, Virtual TCBs

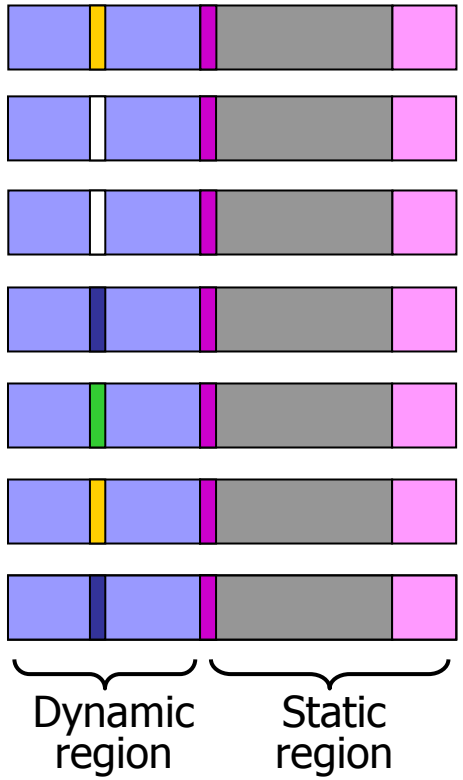
- User regions
- Shared system regions
- Per-space system regions
  - Other kernel tables
  - Physical memory
  - Kernel code
  - TCBs

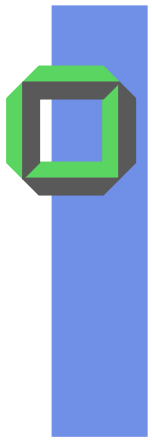




Shared Region Synchronization

- We have
  - Region shared among all address spaces
  - Separate page table per address space
- Updates occur in dynamic region
  - May lead to inconsistencies
- We need
  - Some form of synchronization within dynamic region
  - Make sure valid virtual memory mappings are synchronized



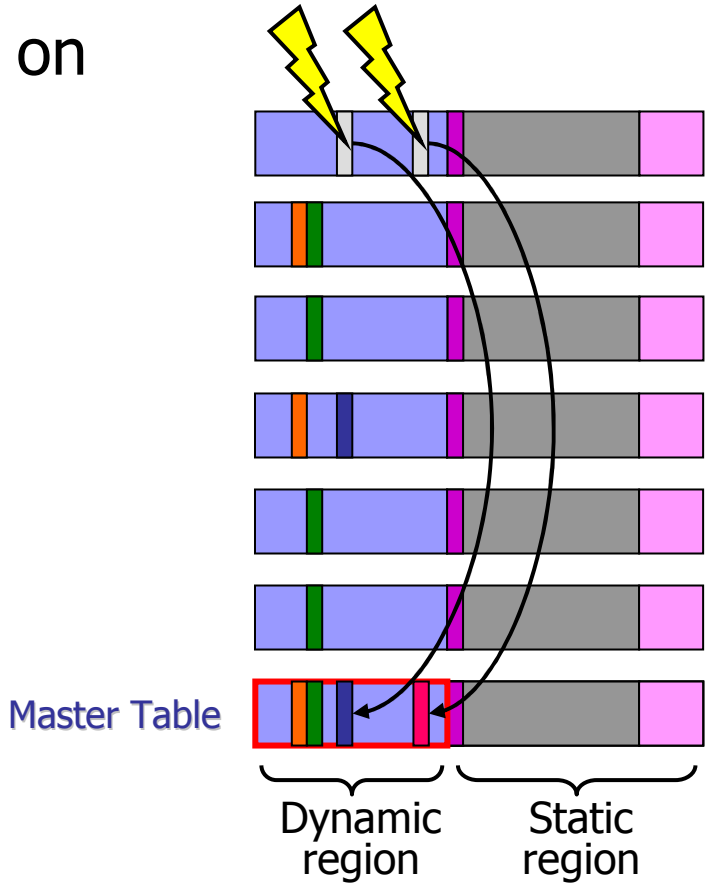


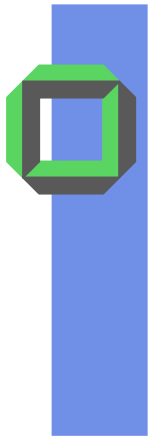
TCB Area Synchronization

Basic Algorithm

- Dedicate one table as master
- Synchronize with master table on page faults
  - Page fault algorithm:

```
if (master entry valid) {
      copy entry from master
} else {
      create new entry in master
      copy entry from master
}
```

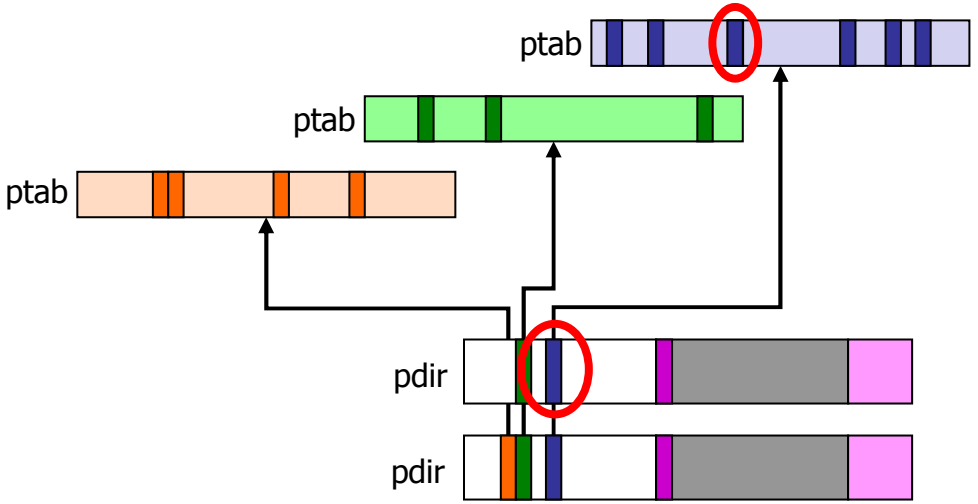


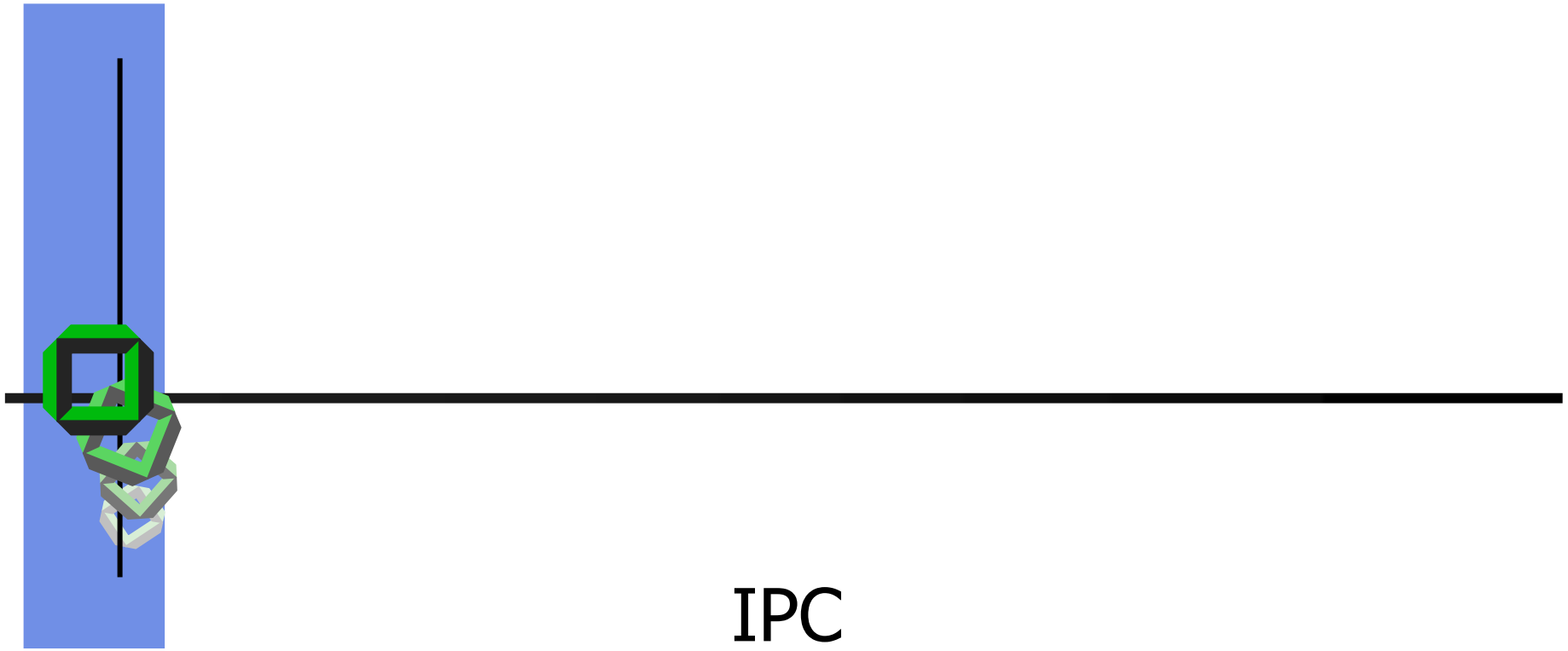


TCB Area Synchronization

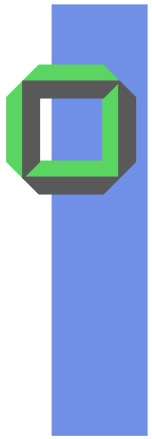
Modifying Mappings

- Page tables have multiple levels
  - IA-32: page directories and page tables
- We only synchronize top level (page directory)
- Modifications in lower levels visible in all spaces
- Works even if entries are invalidated



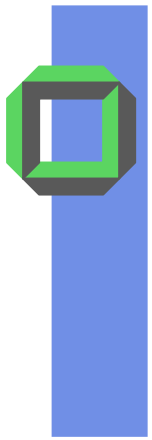


IPC



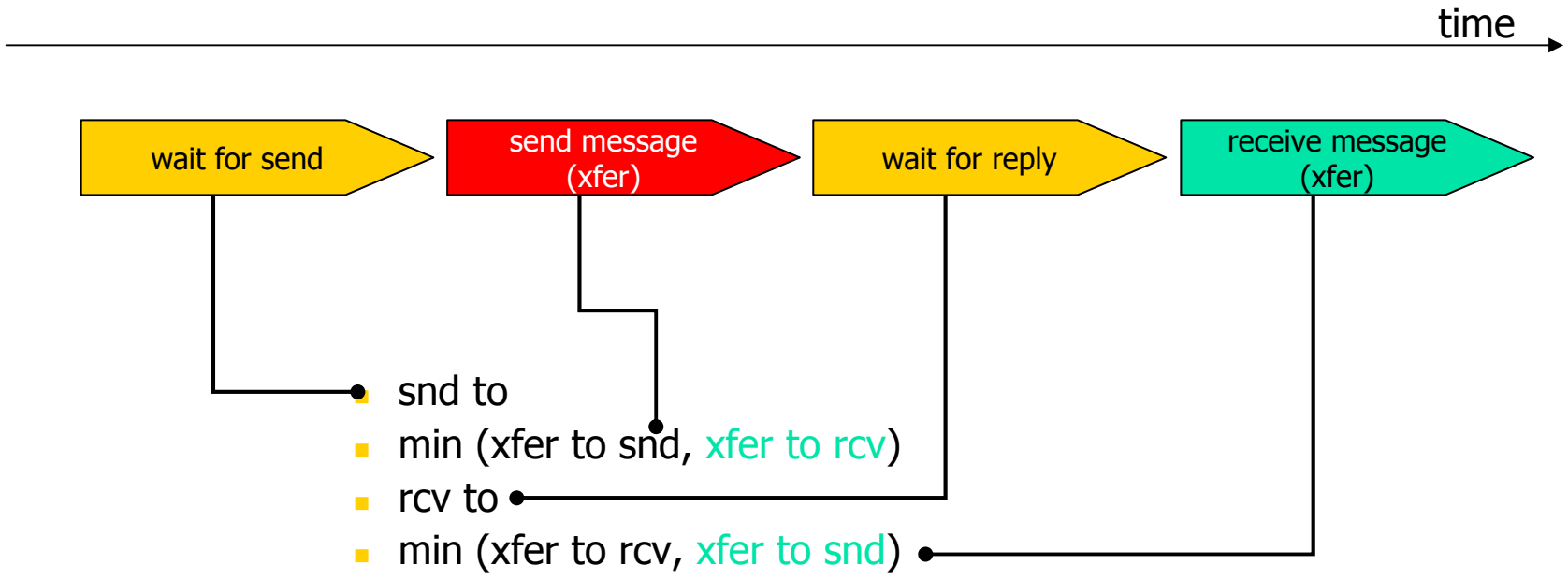
IPC – API

- Operations
  - Send to
  - Receive from
  - Receive
  - **Call**
  - Send to & Receive any
  - Send to & Receive from
- Message Types
  - Registers
  - Strings
  - Map pages



Timeouts

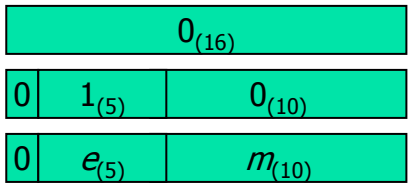
- snd timeout, rcv timeout, xfer timeout snd, xfer timeout rcv



(specified by the partner thread)

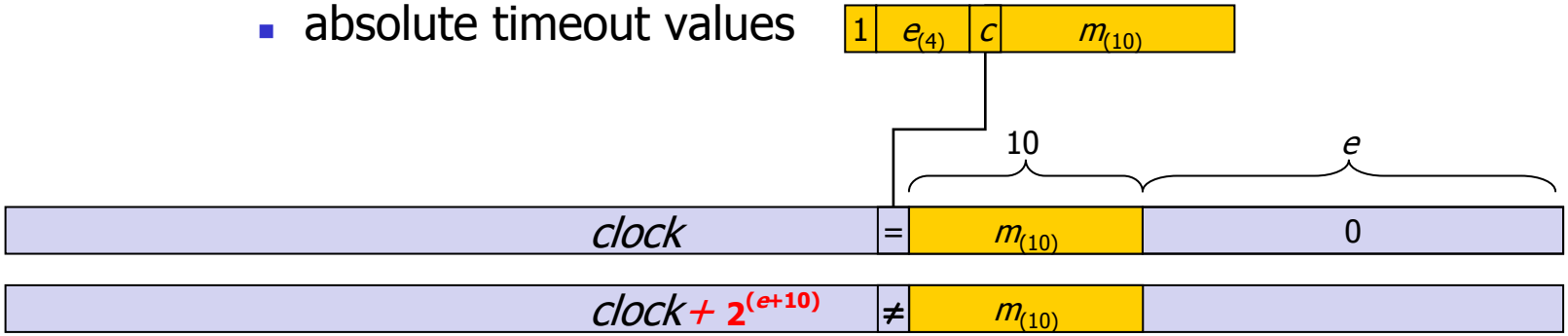
Timeouts

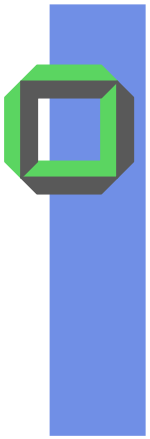
- relative timeout values
  - 0
  - infinite
  - 1 µs … 610 h (log)



$2^e m\ \mu s$

- absolute timeout values

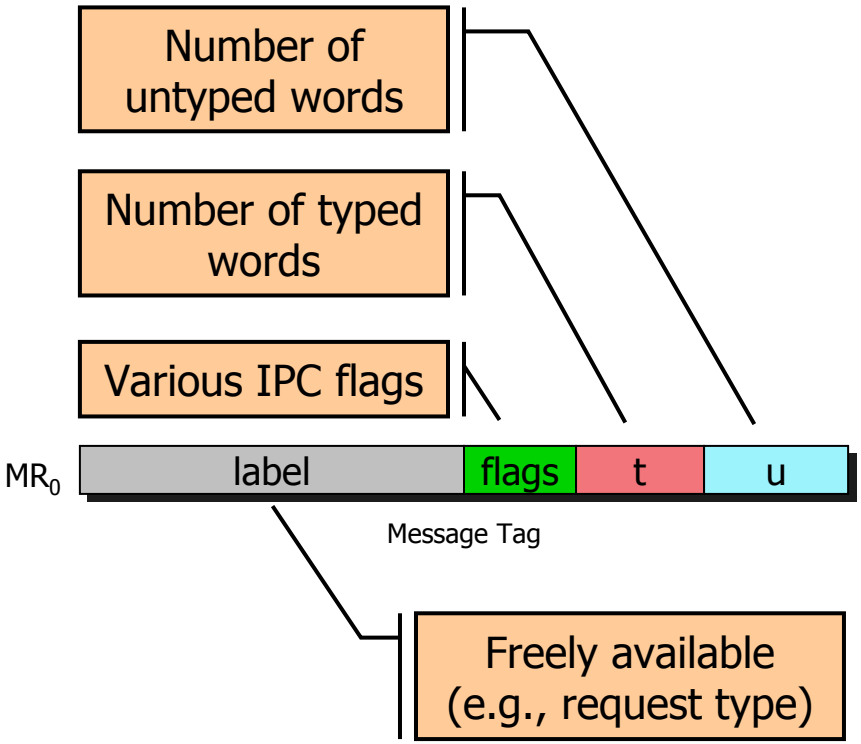


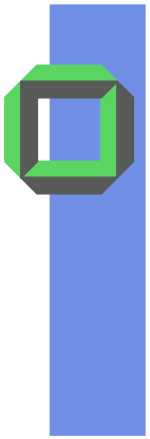


Message Construction

- Messages are stored in registers (MR_0 … MR_{63})
- First register (MR_0) acts as message tag
- Subsequent registers contain
  - Untyped words (u), and
  - Typed words (t)

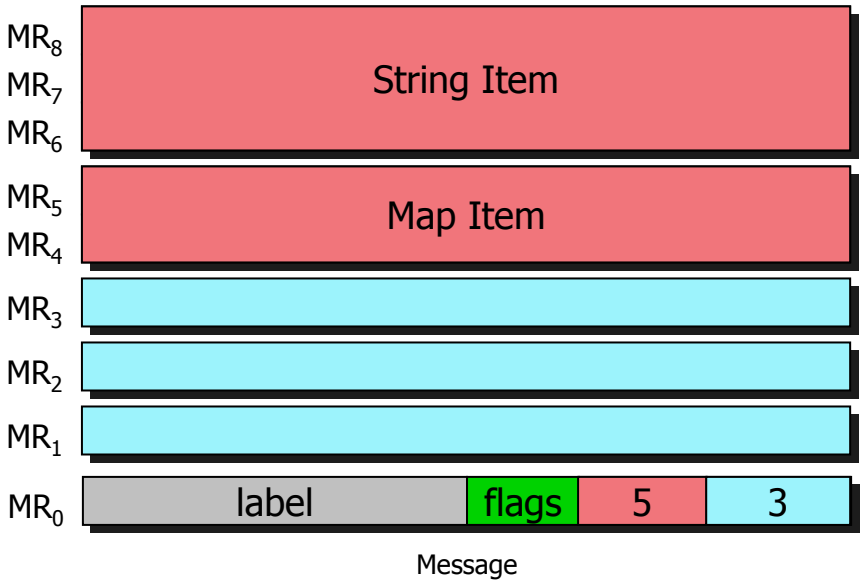
    (e.g., map item, string item)





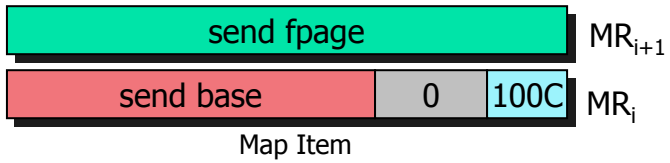
Message Construction

- Typed items occupy one or more words
- Three currently defined items
  - Map item (2 words)
  - Grant item (2 words)
  - String item (2+ words)
- Typed items can have arbitrary order

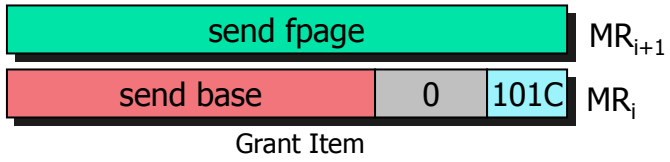


Map and Grant Items

- Two words
  - Send base
  - Fpage
- Lower bits of send base indicates map or grant item



Map Item

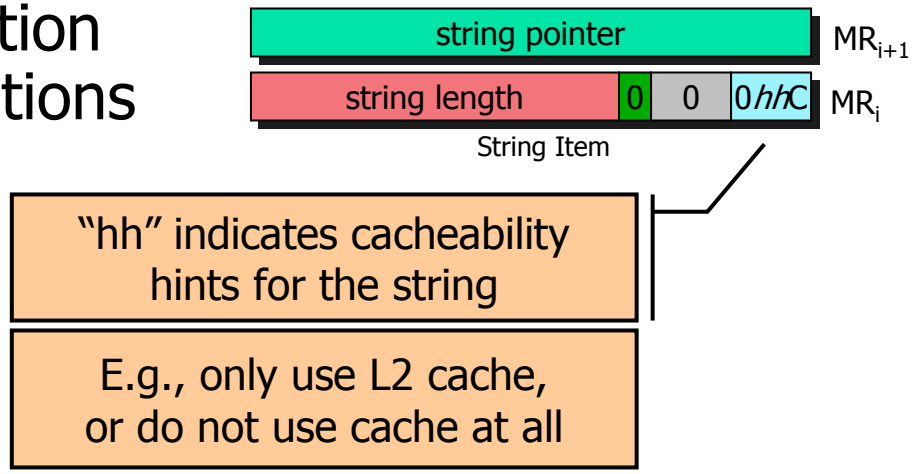


Grant Item

Fpage

String Items

- Up to 4 MB (per string)
- Compound strings supported
  - Allows scatter-gather
- Incorporates cacheability hints
  - Reduce cache pollution for long copy operations

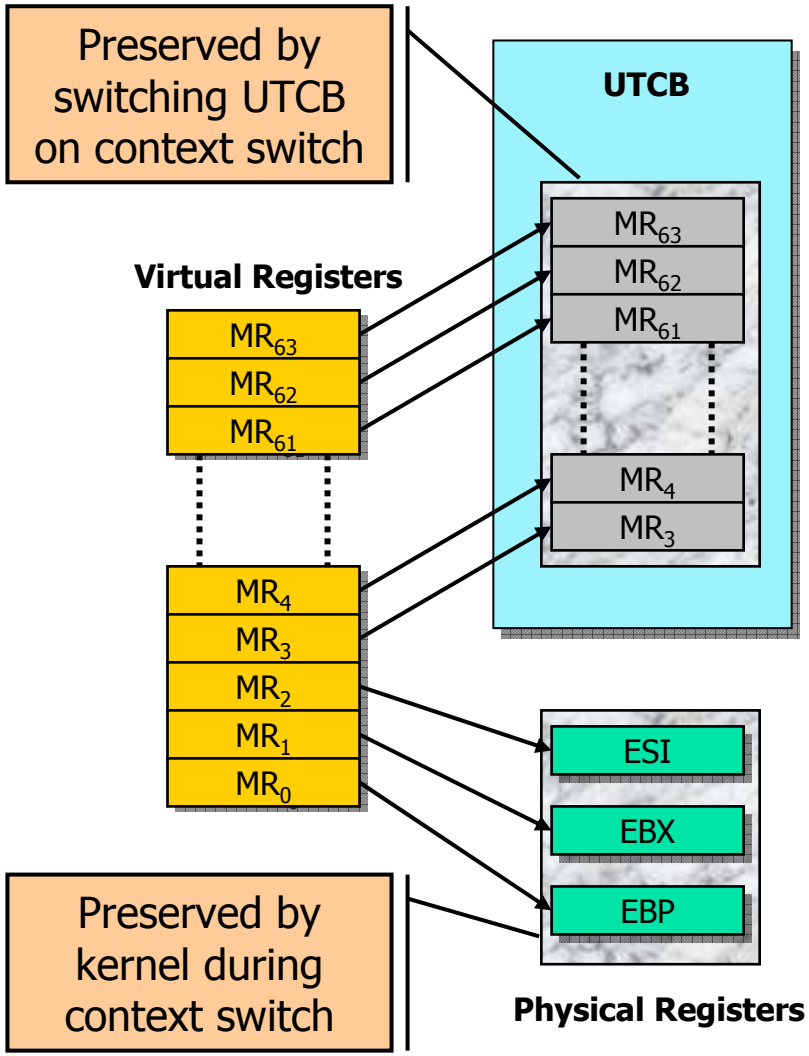


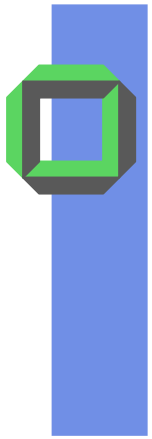
Receiving Messages

- Receiver buffers are specified in registers (BR_0 ... BR_{33})
- First BR (BR_0) contains "Acceptor"
  - May specify receive window (if not nil-fpage)
  - May indicate presence of receive strings/buffers (if s-bit set)

What are Virtual Registers?

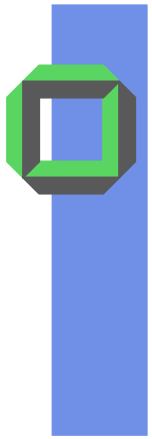
- Virtual registers are backed by either
  - Physical registers, or
  - Non-pageable memory
- UTCBs hold the memory backed registers
  - UTCBs are thread local
  - UTCB can not be paged
    - No page faults
    - Registers always accessible





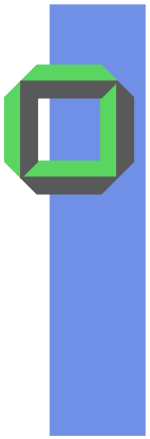
Implementation Goal

- Most frequent kernel op: short IPC
  - Thousands of invocations per second
- Performance is critical
  - Structure IPC for speed
  - **Structure entire kernel to support fast IPC**
- What affects performance?
  - Cache line misses
  - TLB misses
  - Memory references
  - Pipe stalls and flushes
  - Instruction scheduling



Requirements for Fast Path IPC

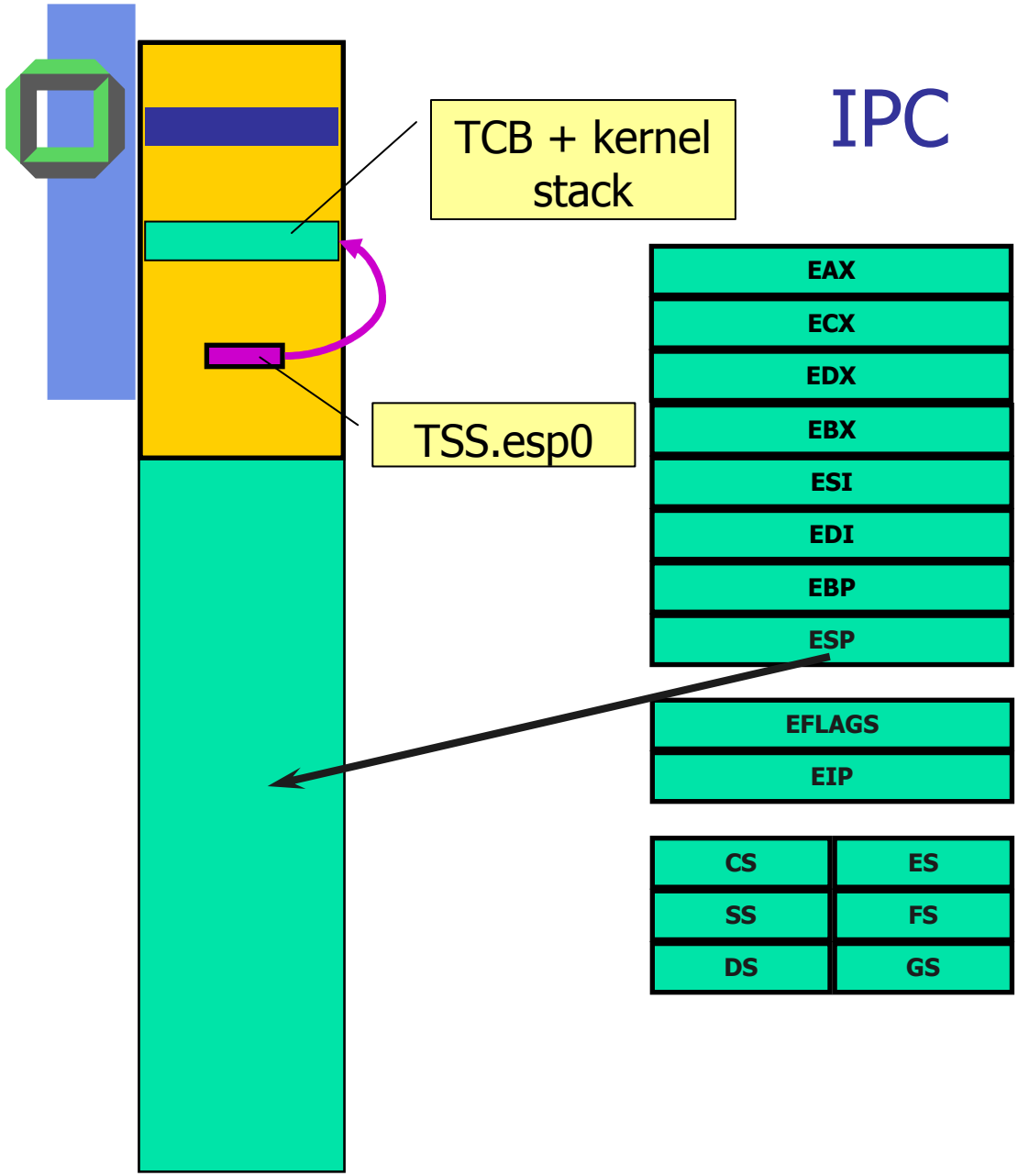
- Untyped message
- Single runnable thread after IPC
  - Must be valid call-like IPC
    - Send phase
      - Target is already waiting
    - Receive phase
      - Sender is **not** ready to couple, causing up to block
  - Switch threads, originator blocks
- No receive timeout
  - Send timeout can be ignored: receiver is waiting
  - Xfer timeouts do not apply for untyped messages



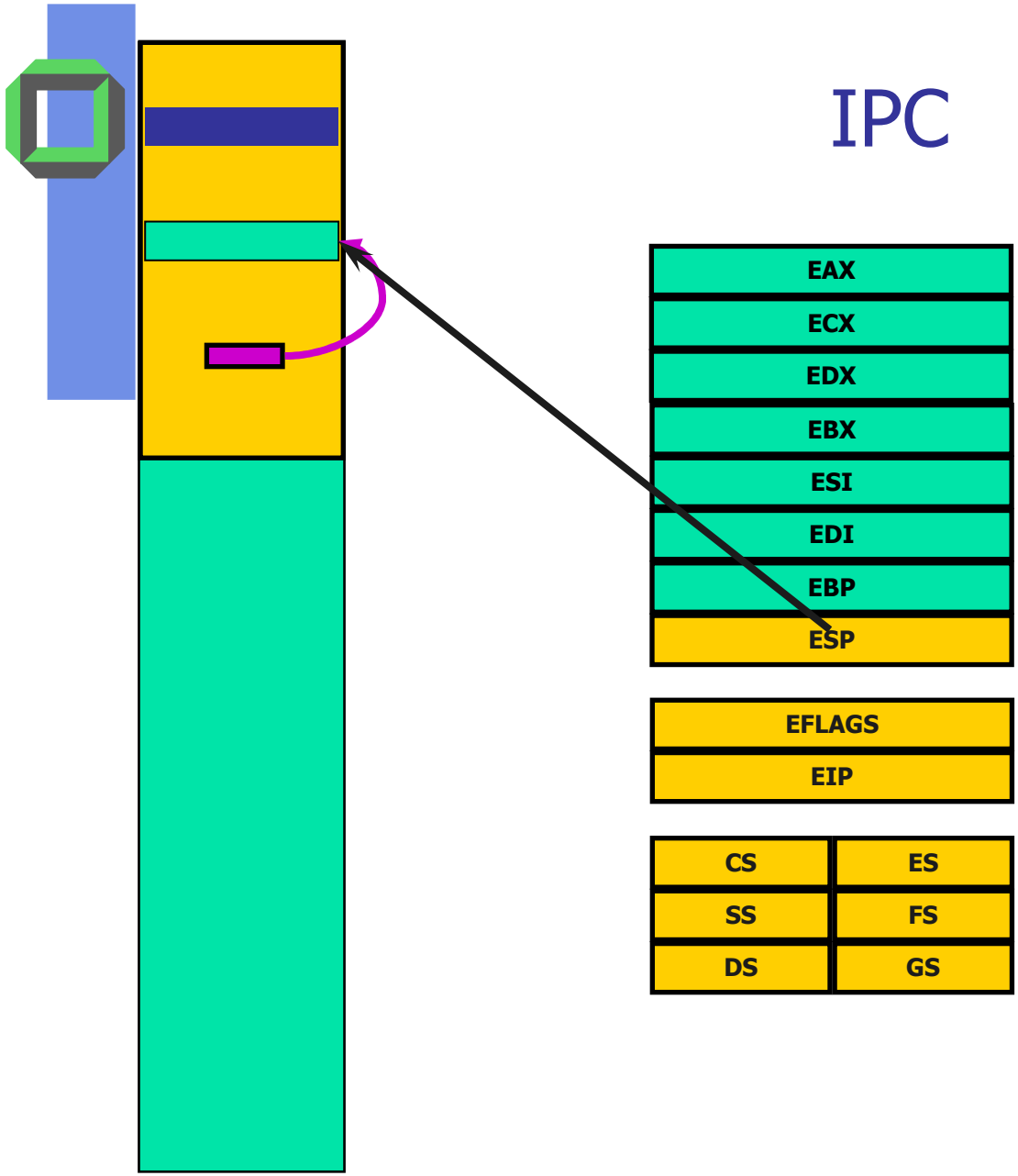
Memory is Forbidden

- Memory references are slow
  - Avoid in IPC
    - E.g., use lazy scheduling
  - Avoid in common case
    - E.g., (xfer) timeouts
- Microkernel should minimize artifacts
  - Cache pollution
  - TLB pollution
  - Memory bus

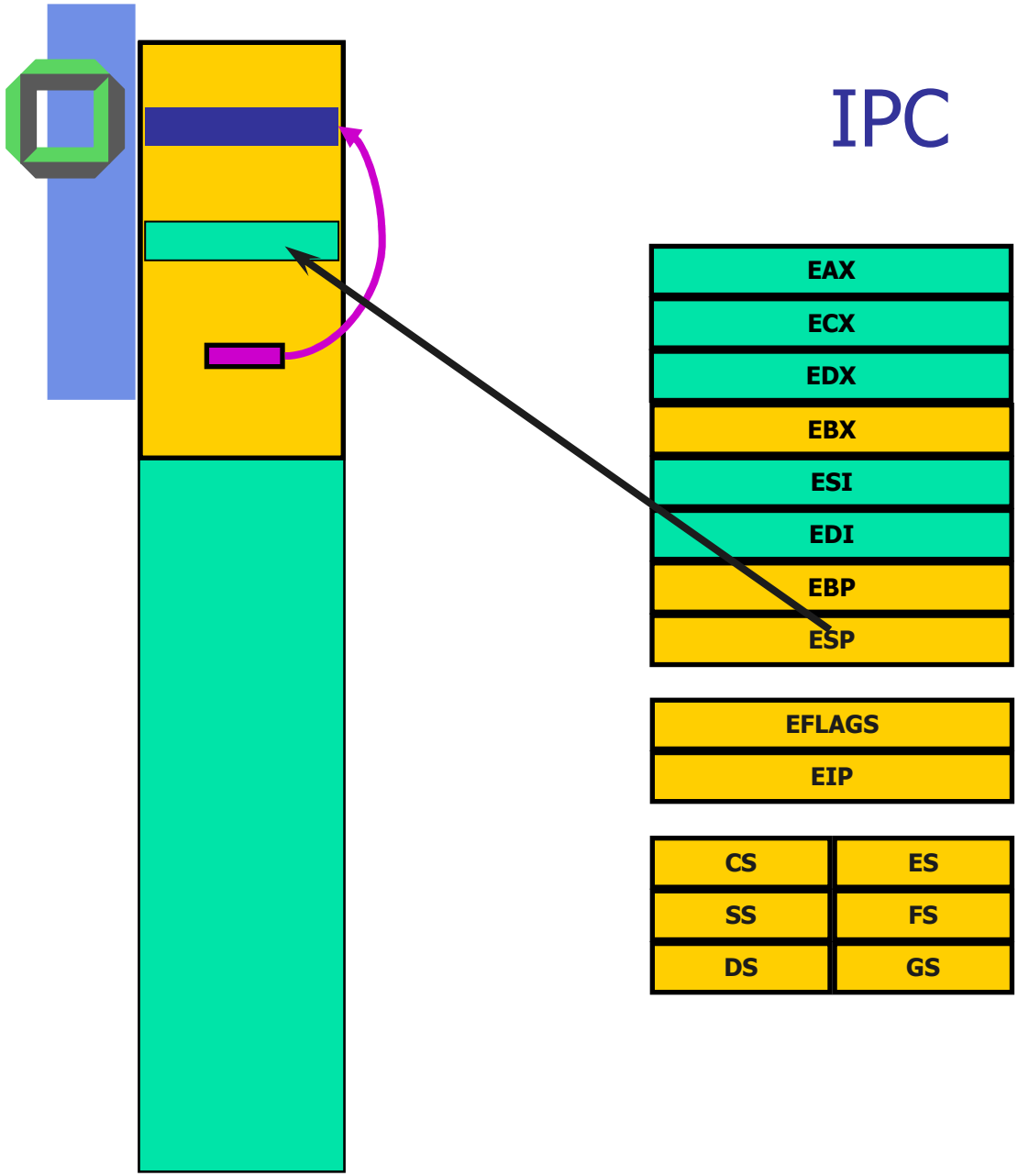
IPC



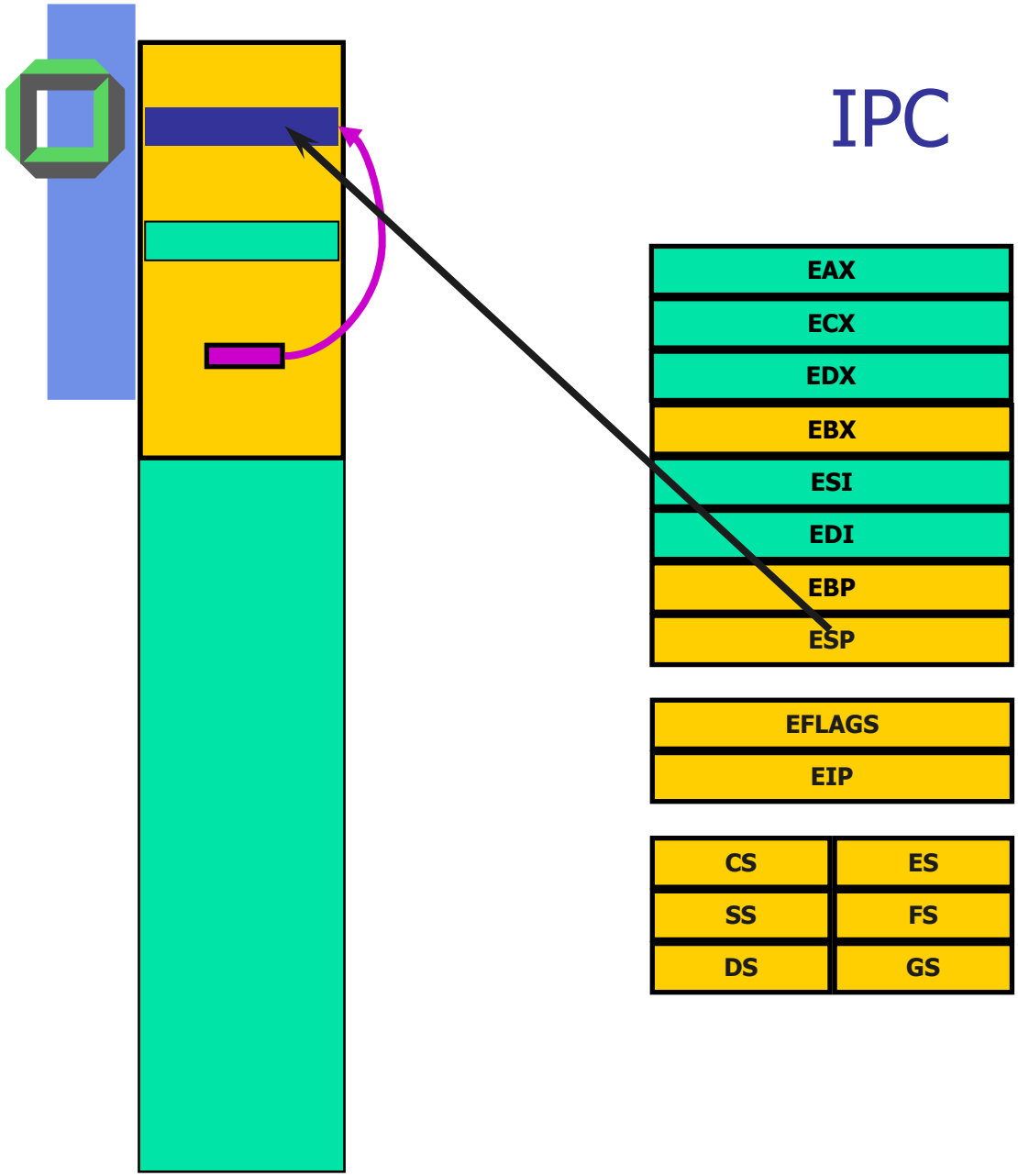
IPC



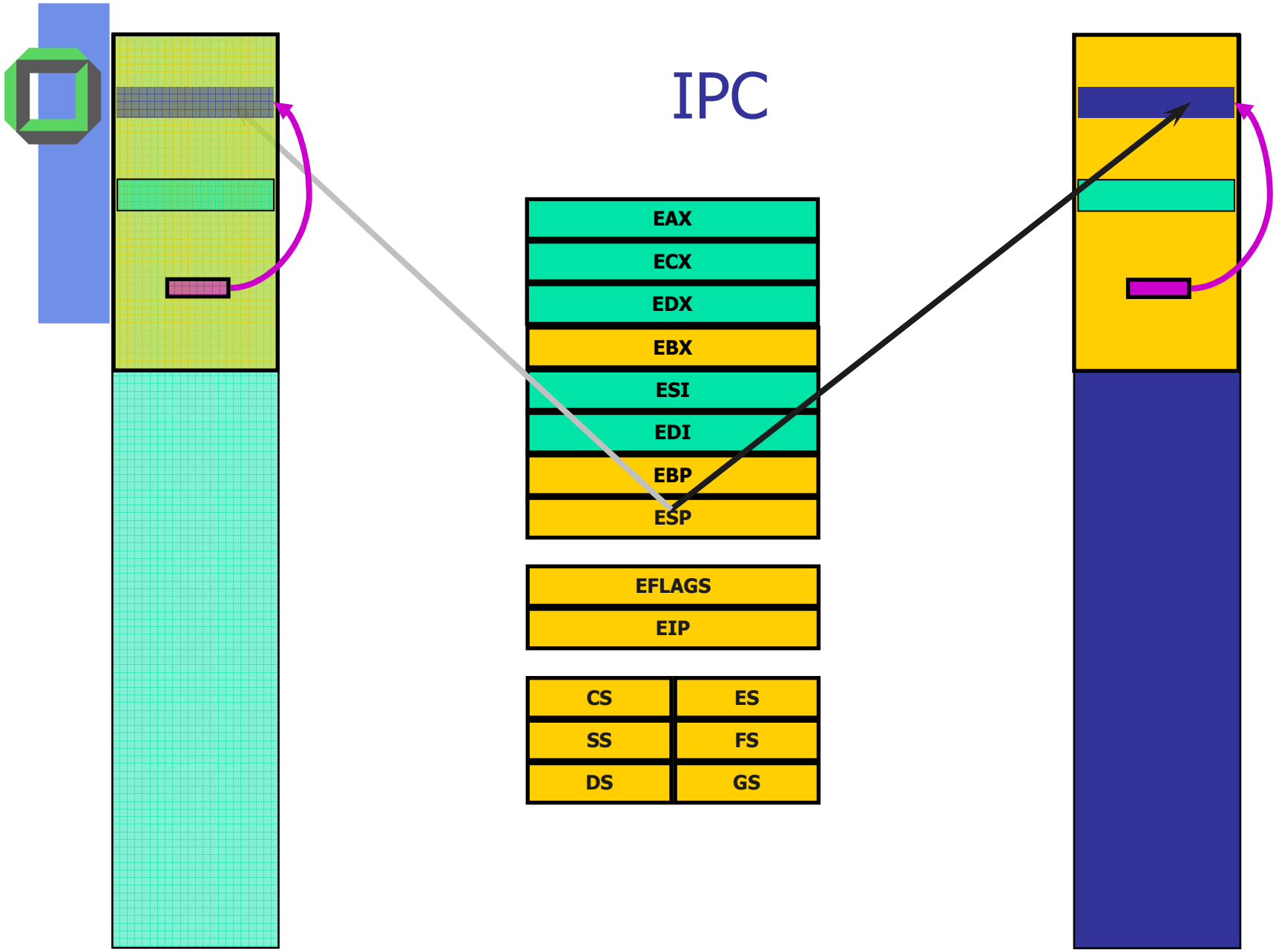
IPC

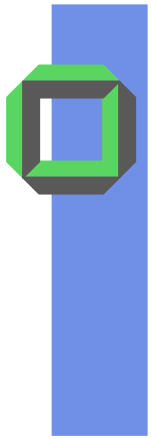


IPC

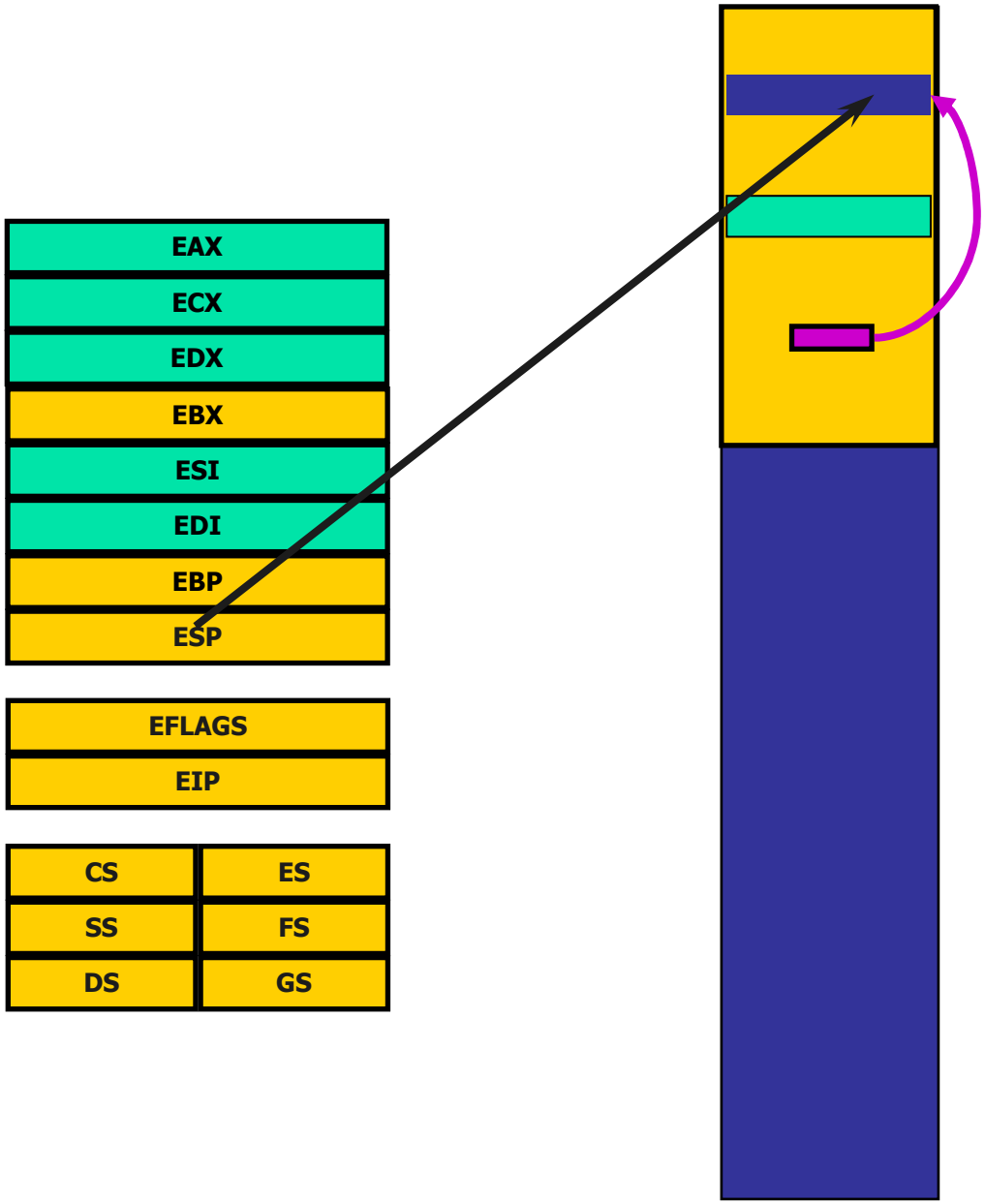


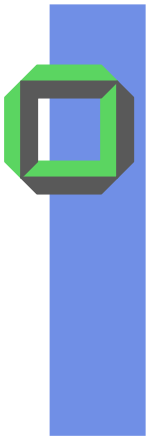
IPC



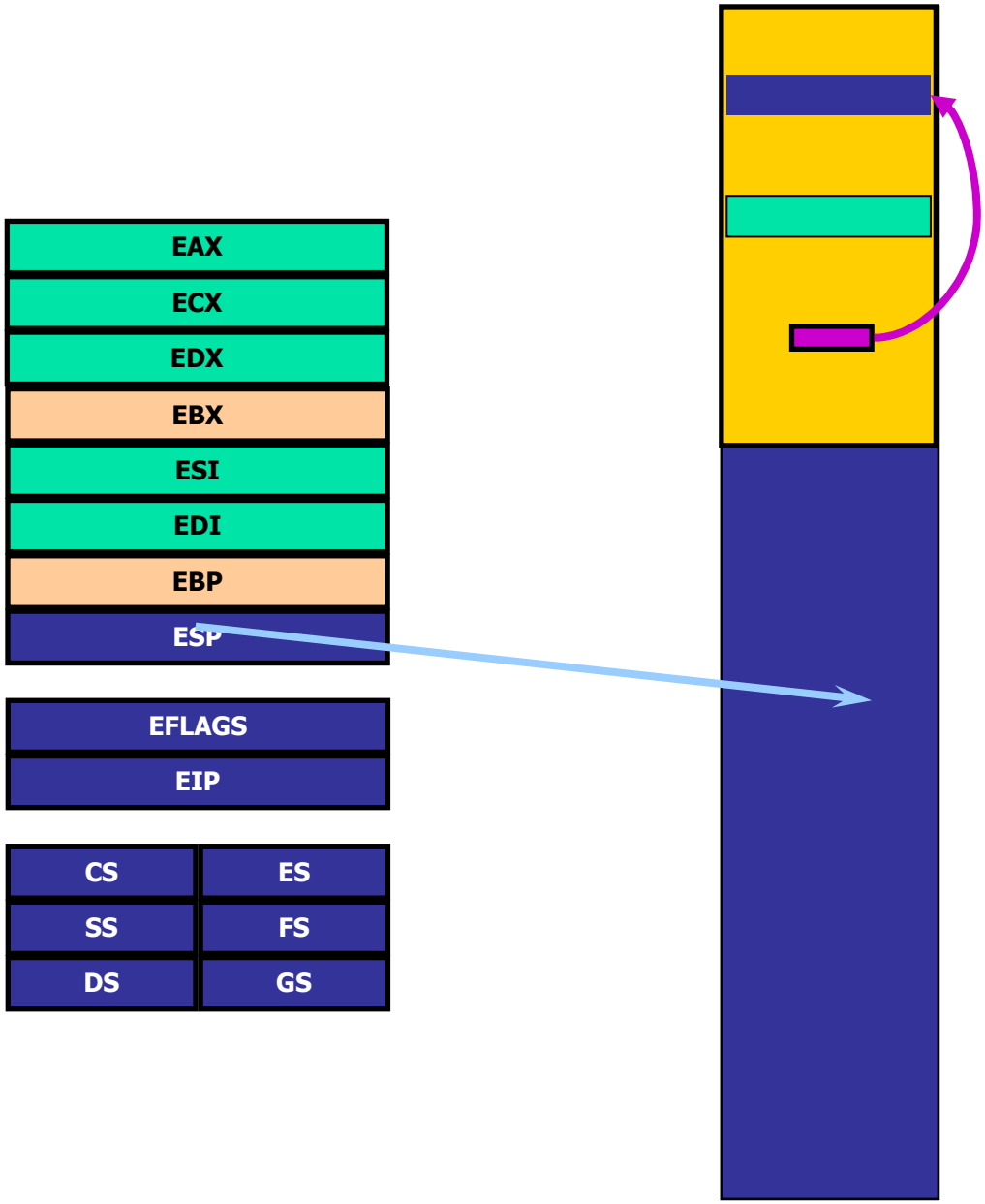


IPC



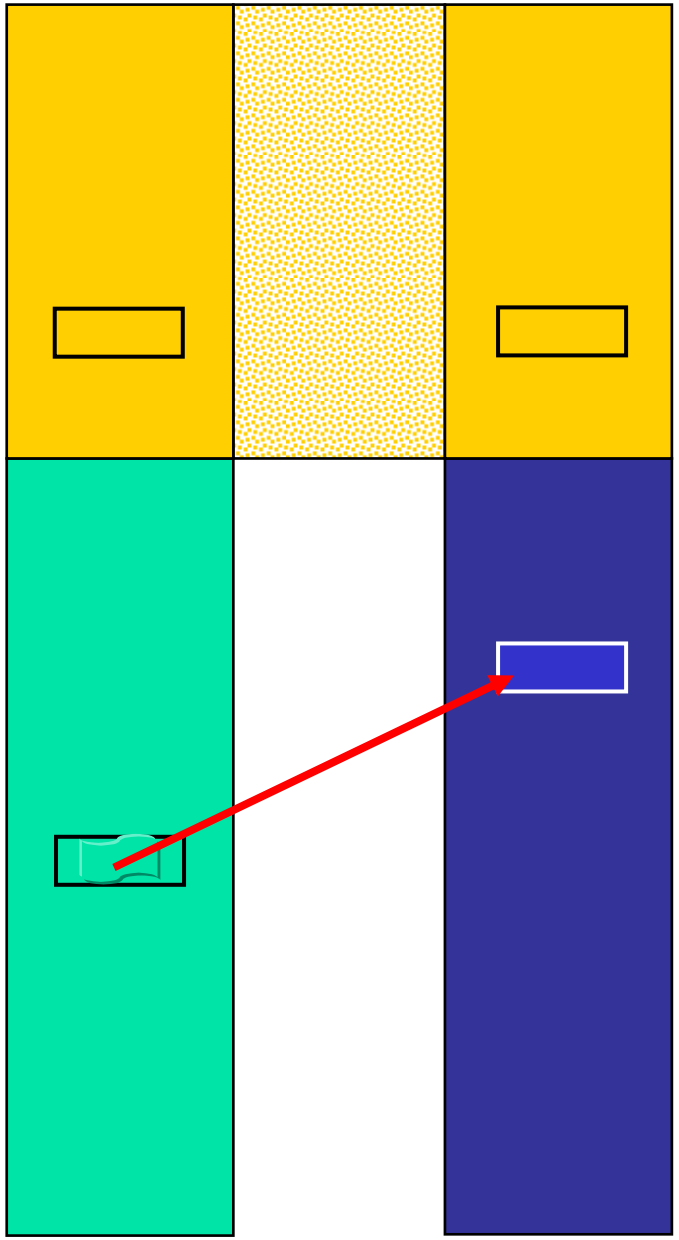


IPC

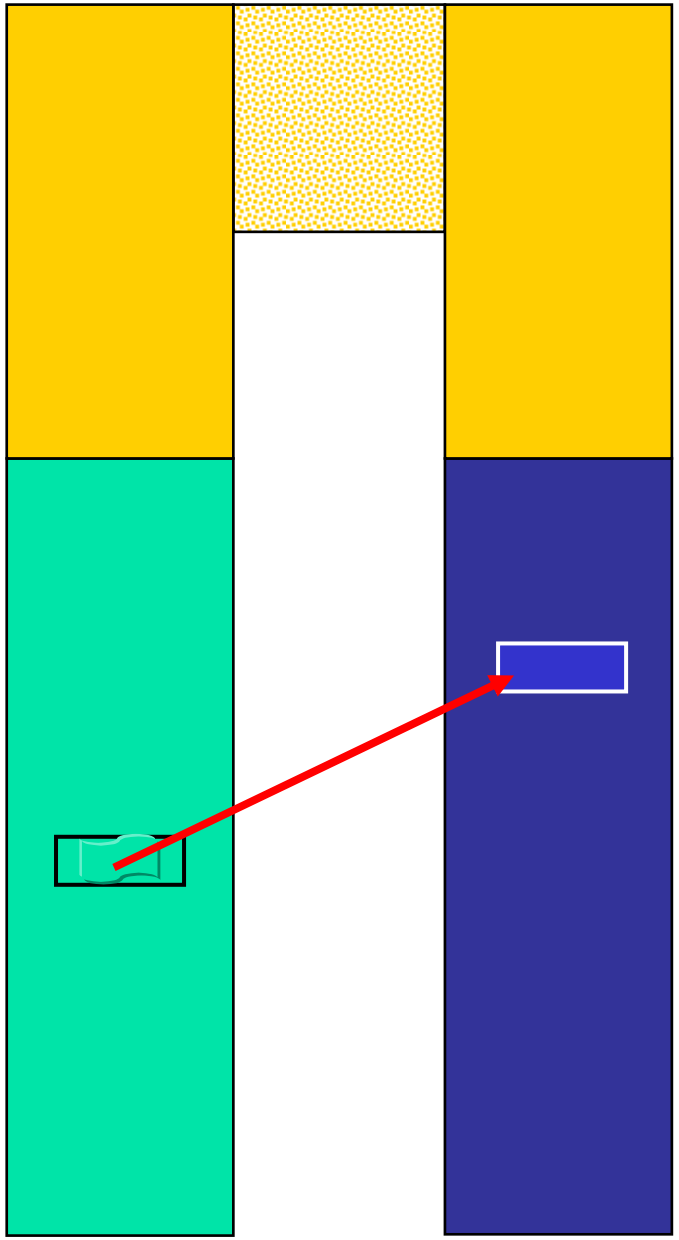


String IPC / memcpy

- Why?
  - Trust
  - Granularity
  - Synchronous ("atomic") transfer

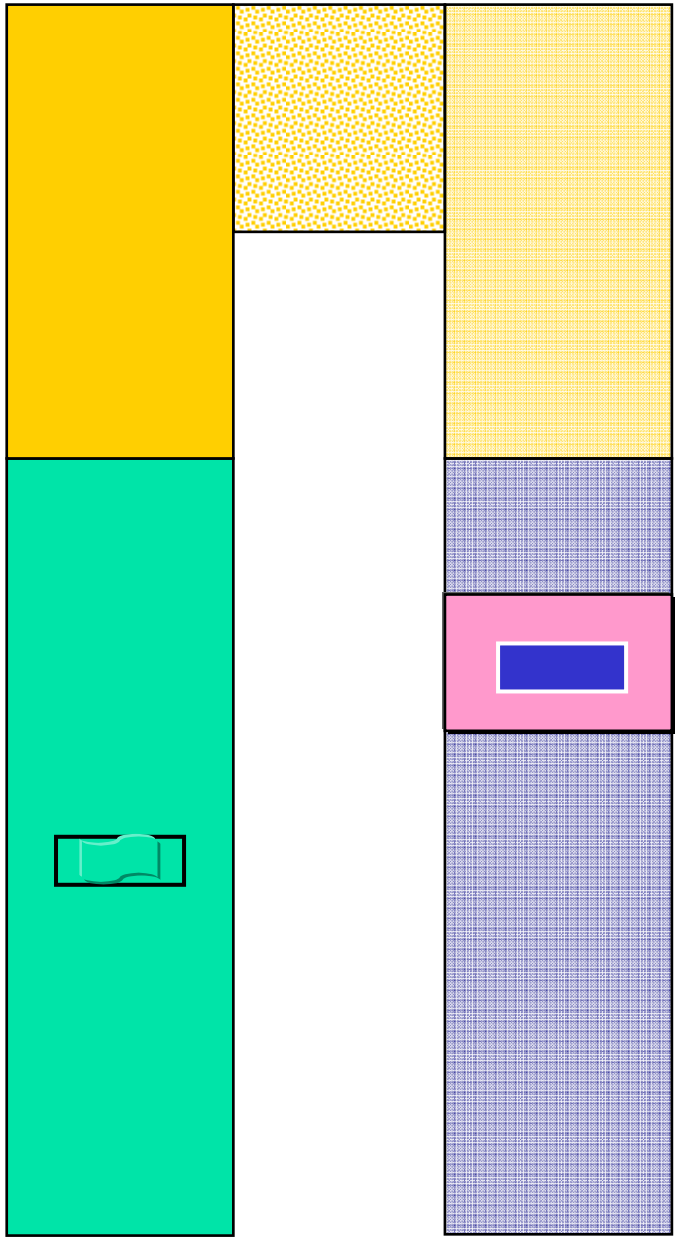


Temporary Mapping



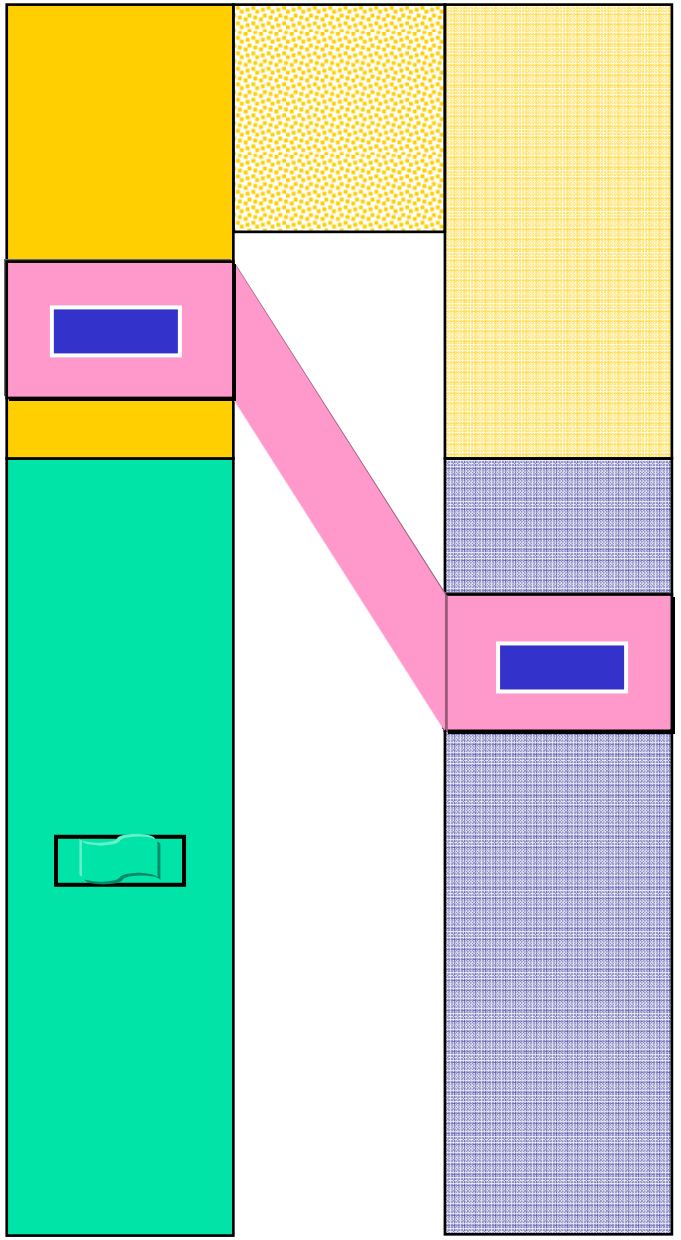
Temporary Mapping

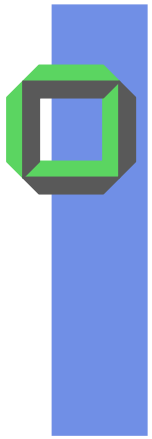
- Select dest area (2x4 MB)



Temporary Mapping

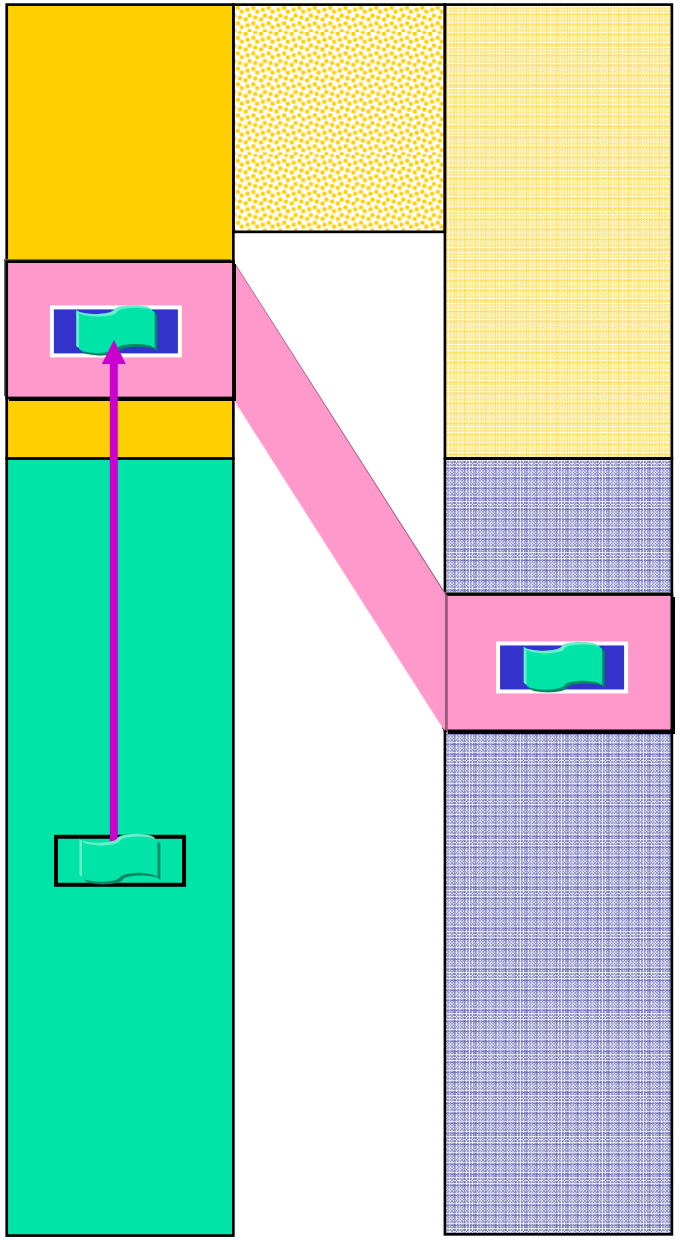
- Select dest area (2x4 MB)
- Map into source AS (kernel)

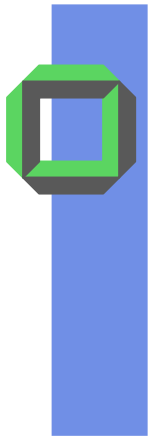




Temporary Mapping

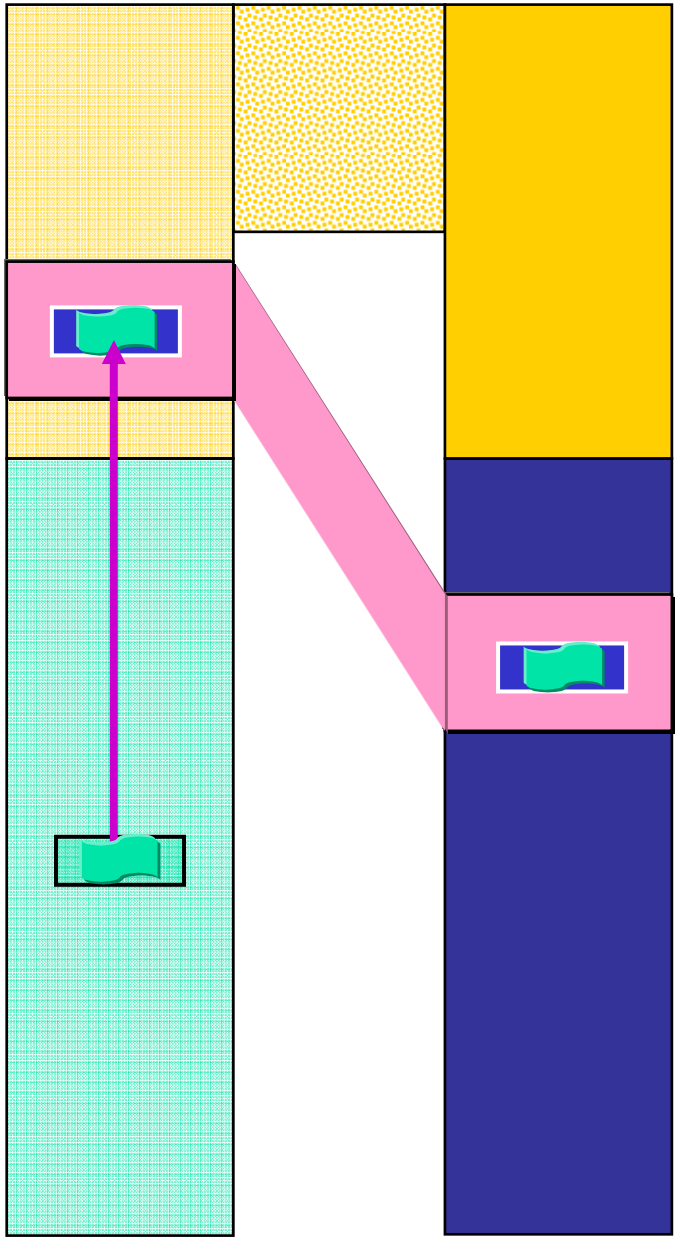
- Select dest area (2x4 MB)
- Map into source AS (kernel)
- Copy data





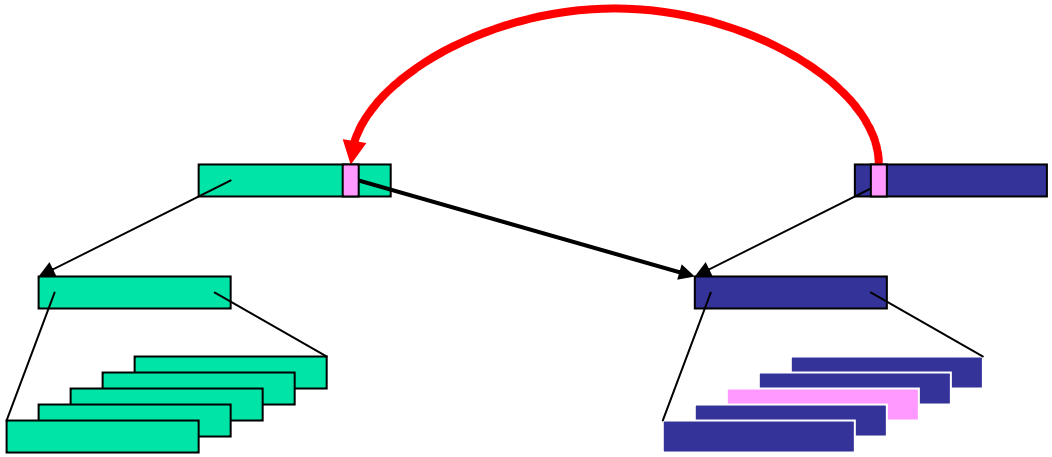
Temporary Mapping

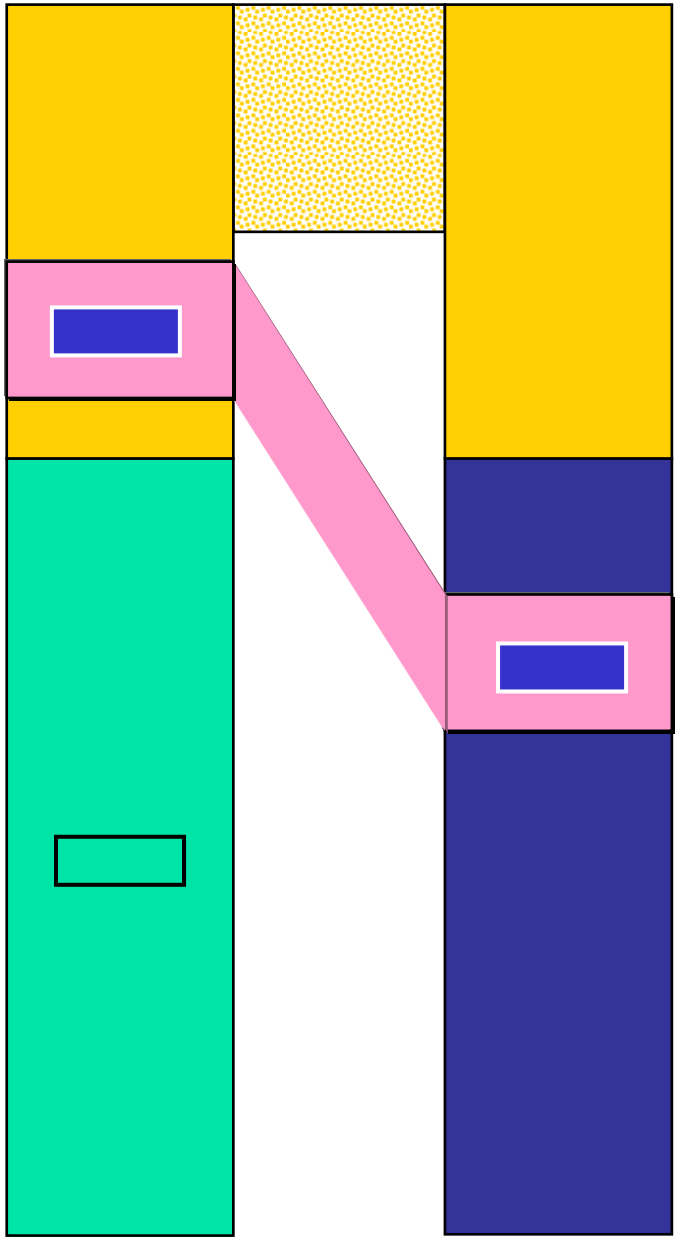
- Select dest area (2x4 MB)
- Map into source AS (kernel)
- Copy data
- Switch to dest space

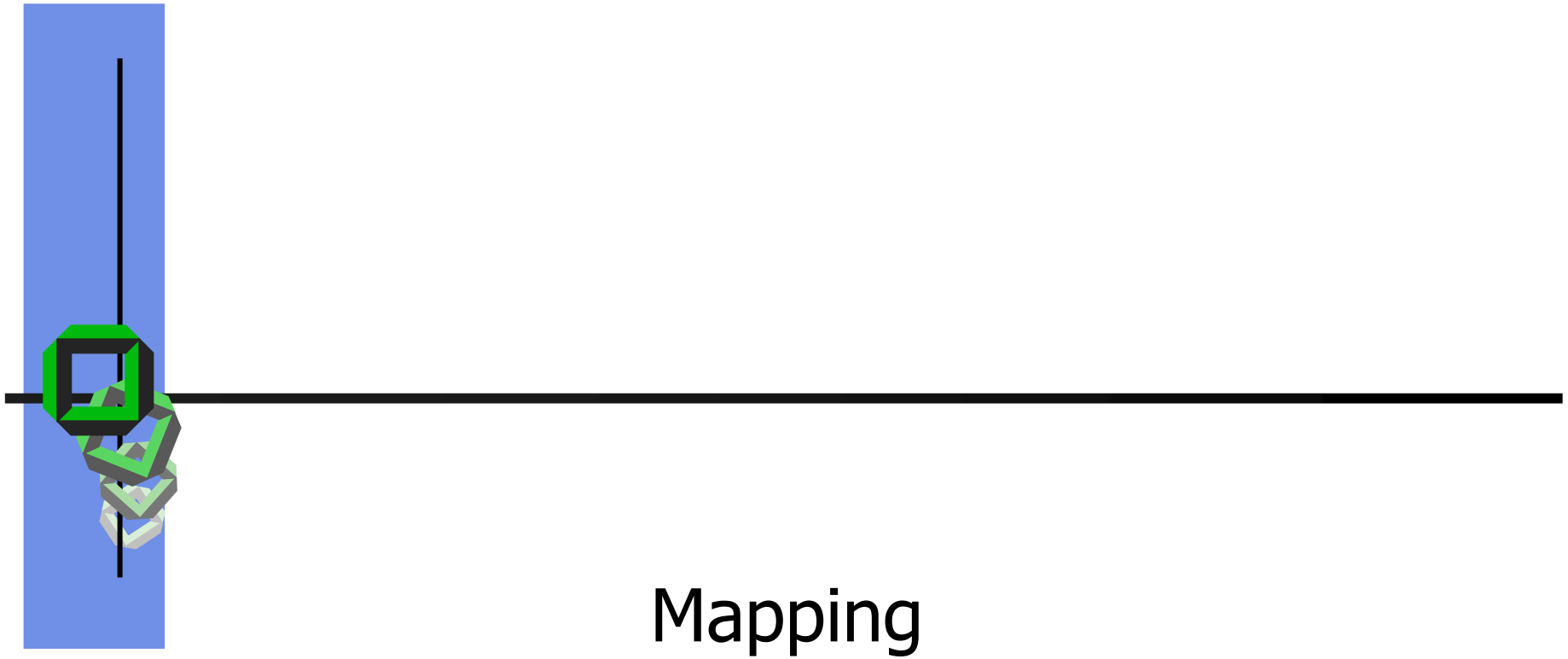


Temporary Mapping

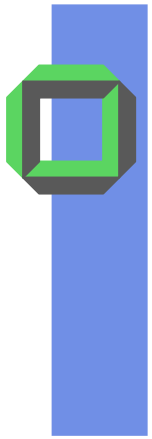
- Copy page directory entry (PDE) from dest
  - Addresses in temporary mapping area are resolved using dest's page *table*







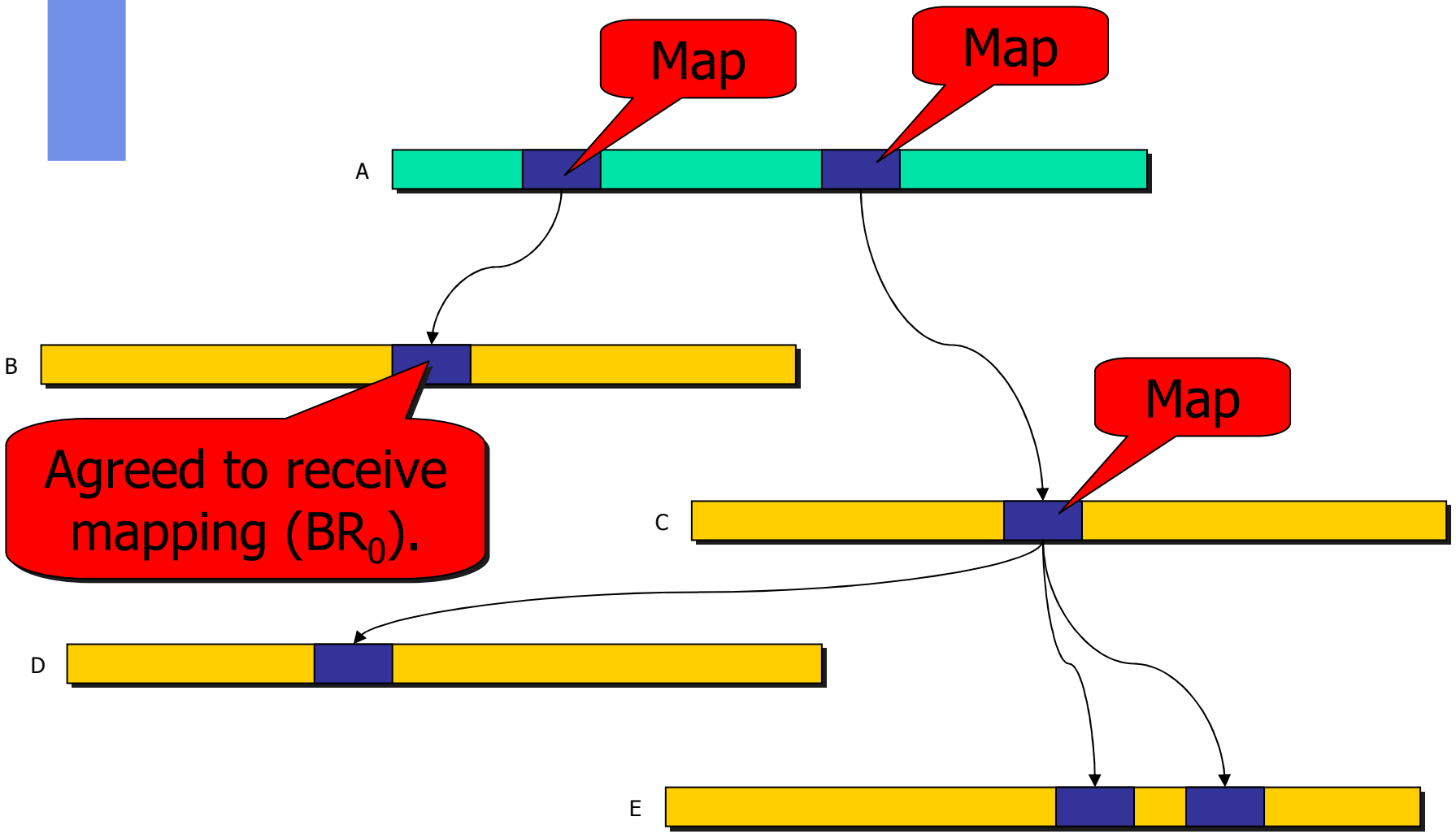
Mapping



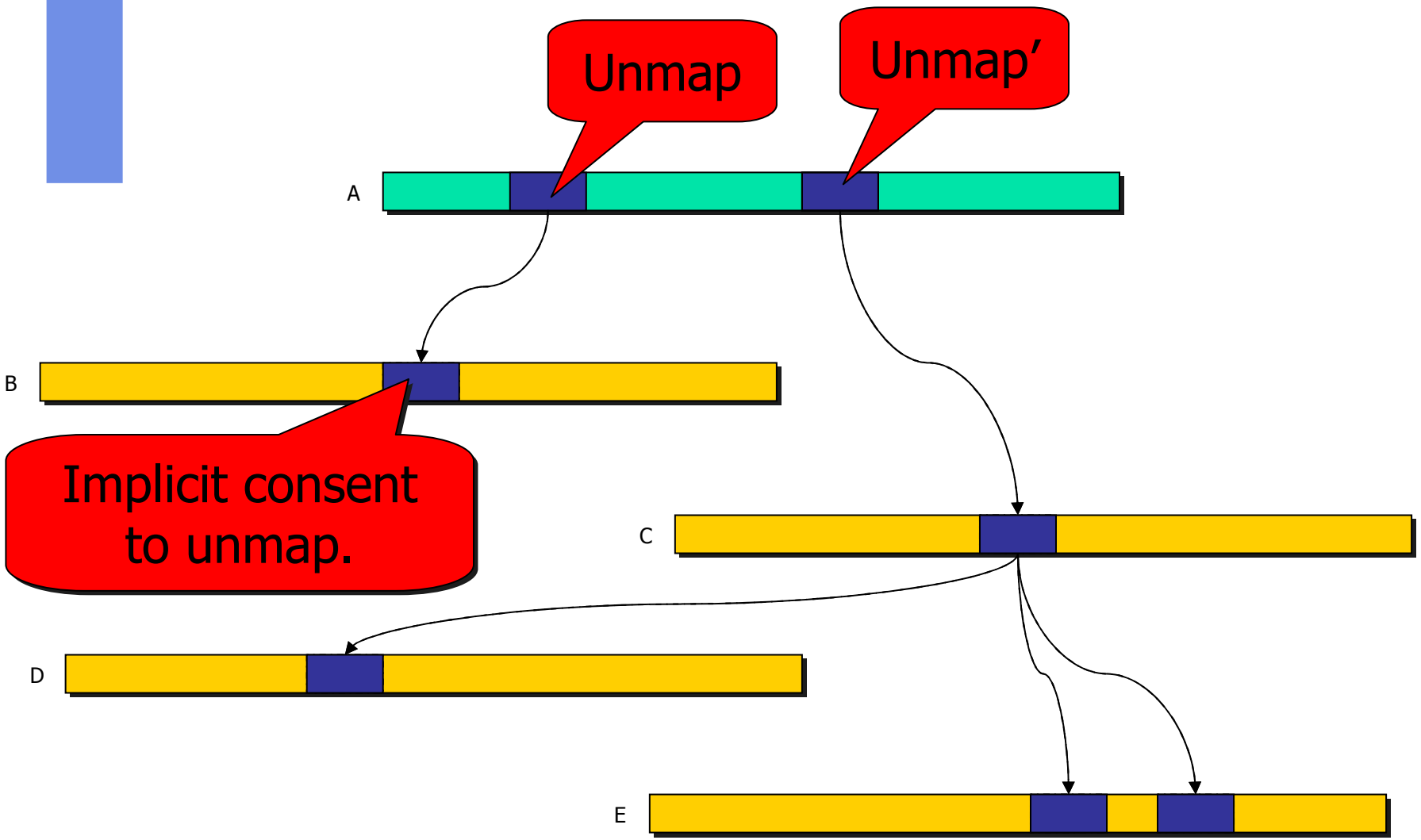
Mechanisms

- We need tools to build address spaces
  - Map
  - Unmap
- We need security
  - Access permissions [rwx]
- We need resource control
  - Use bits [accessed or dirty]
  - Page fault messages [detect page use]

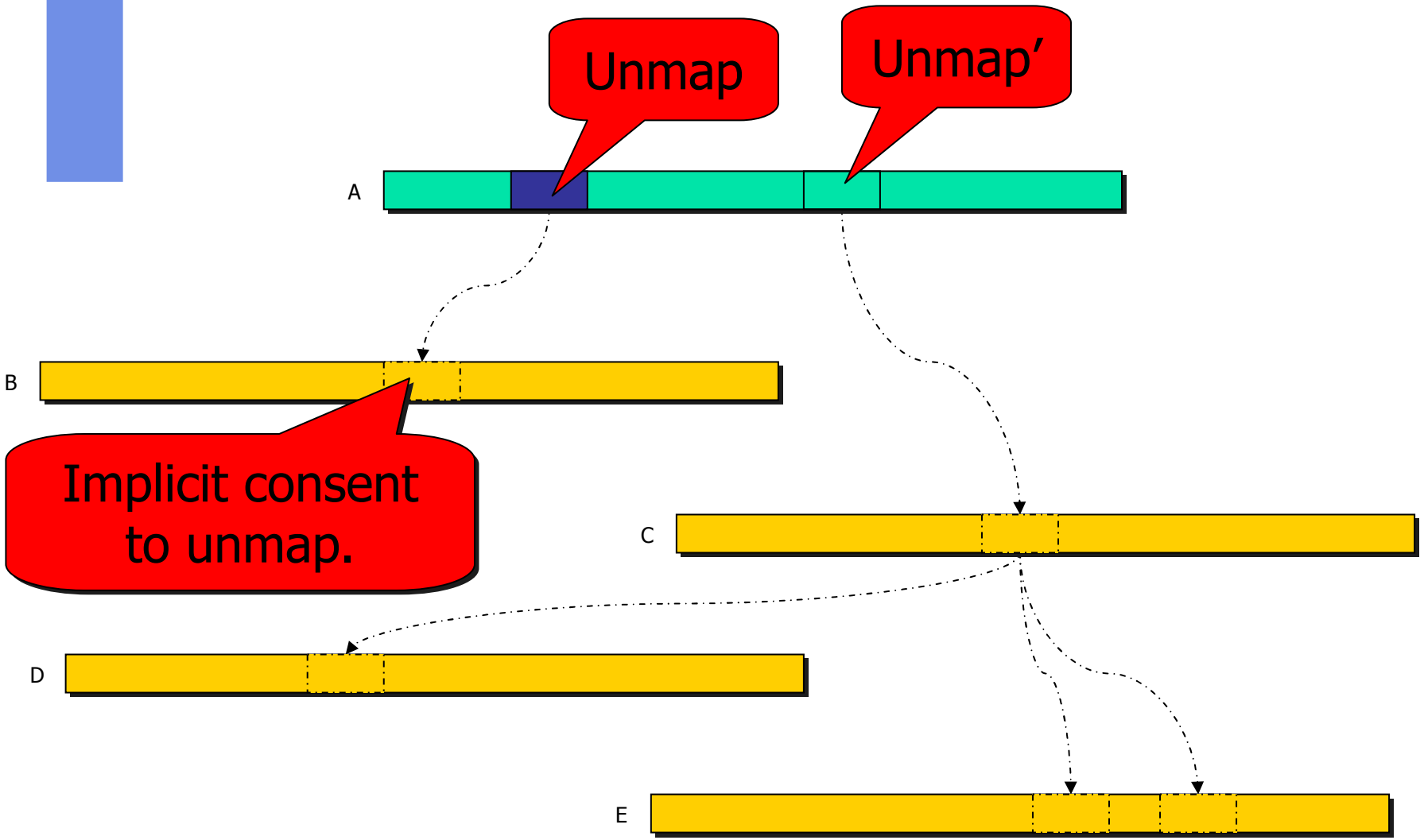
Map



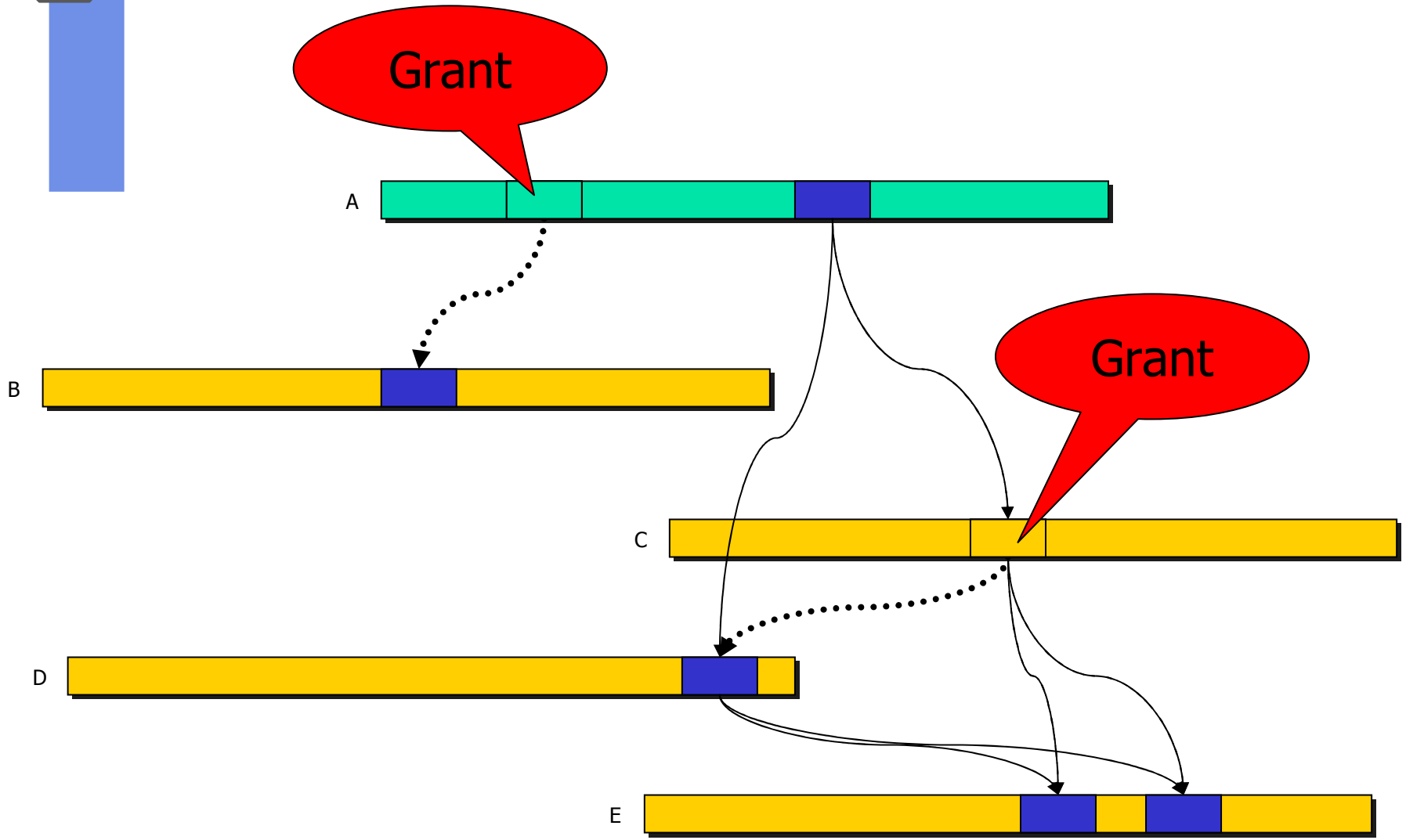
Unmap



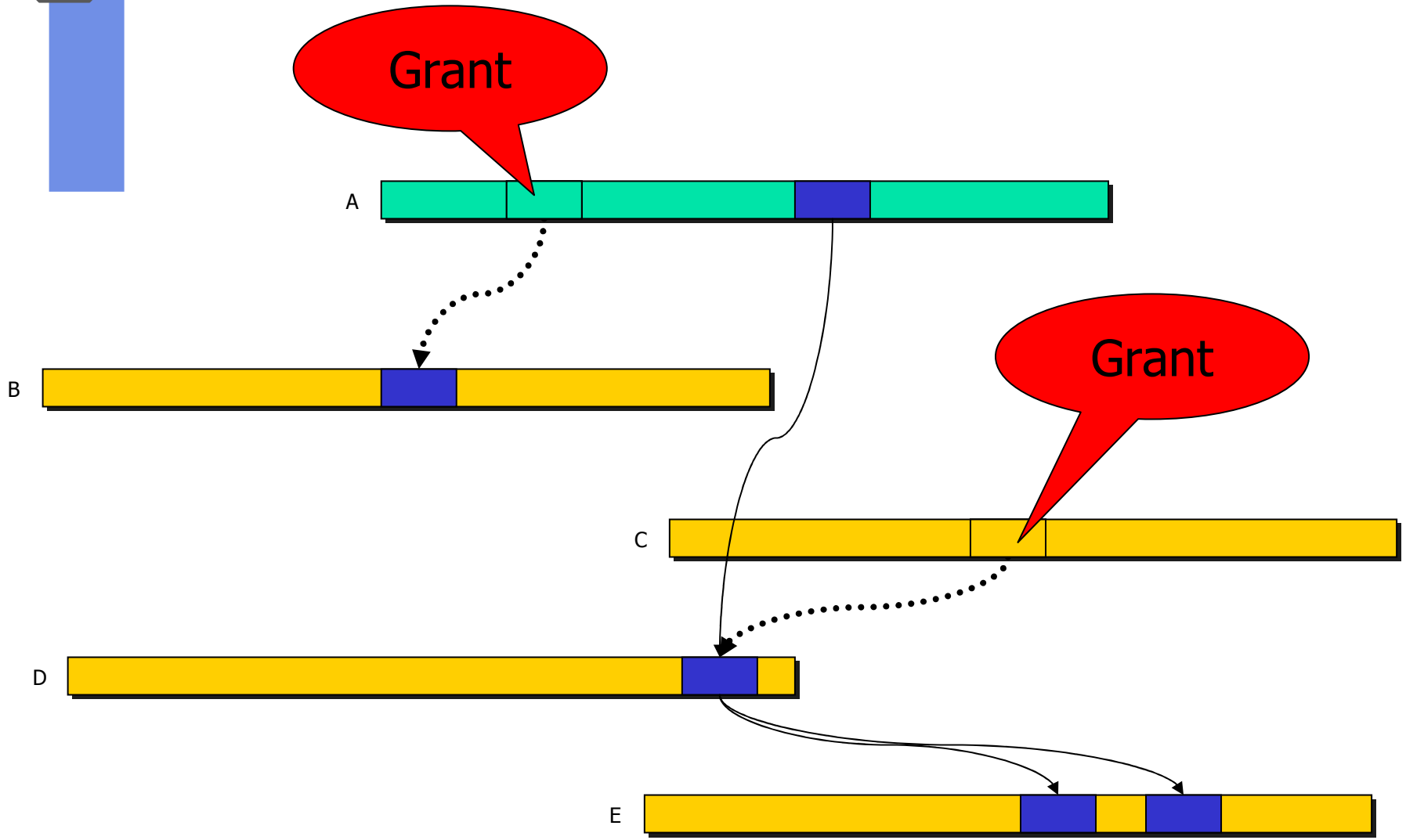
Unmap



Grant

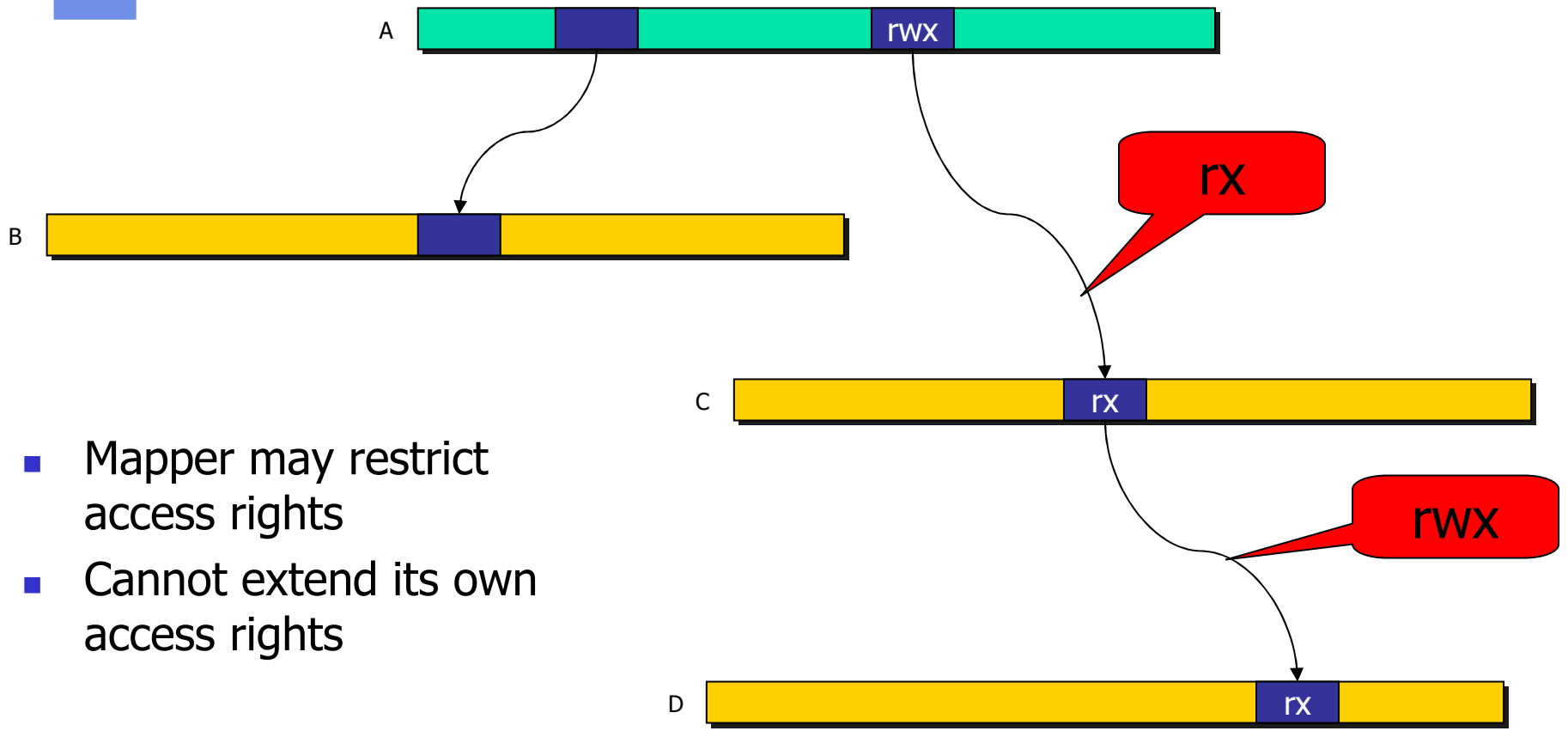


Grant



Access Rights – Map

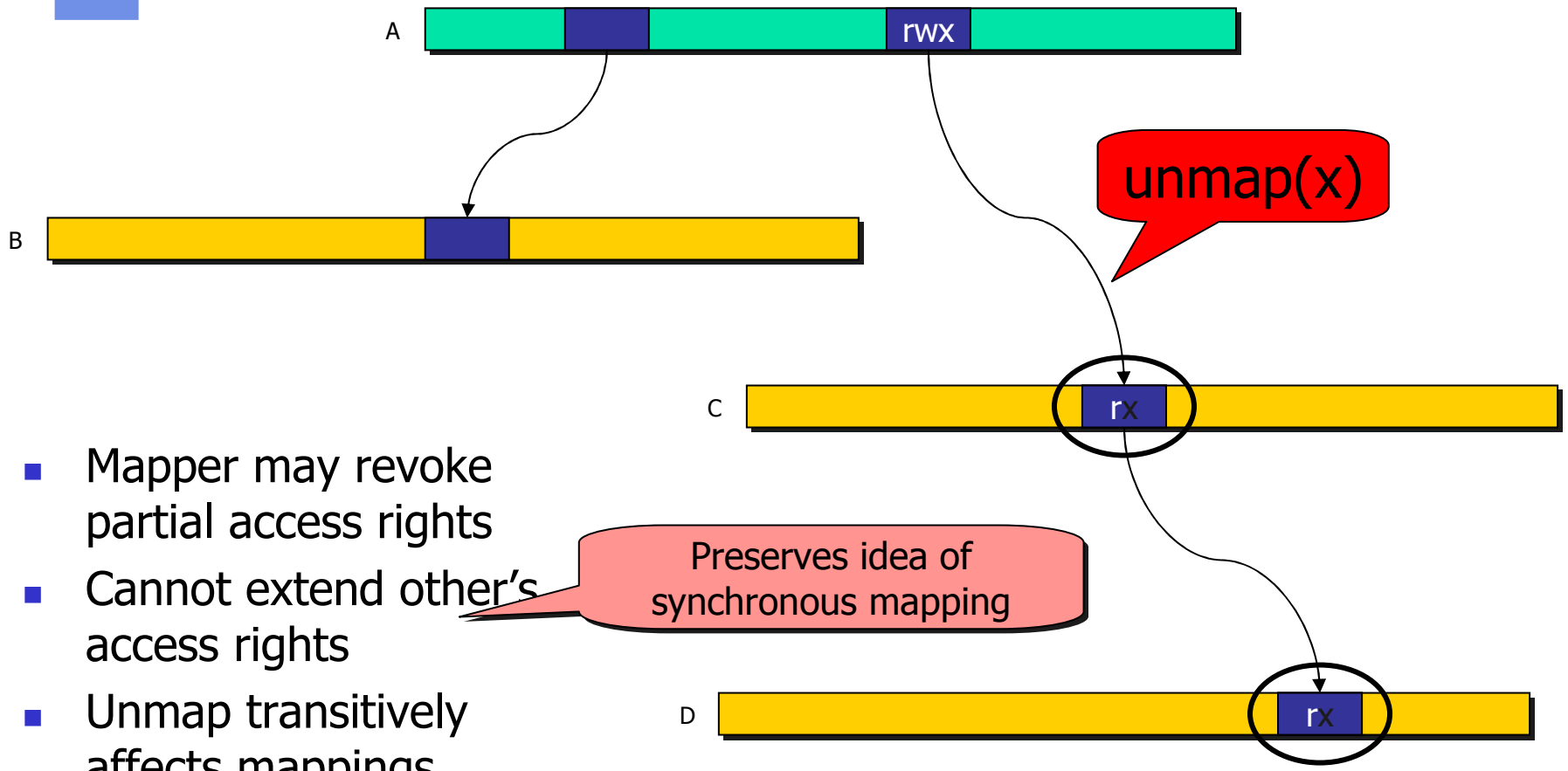
r = Read
w = Write
x = eXecute



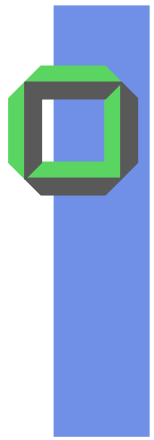
- Mapper may restrict access rights
- Cannot extend its own access rights

Access Rights – Unmap

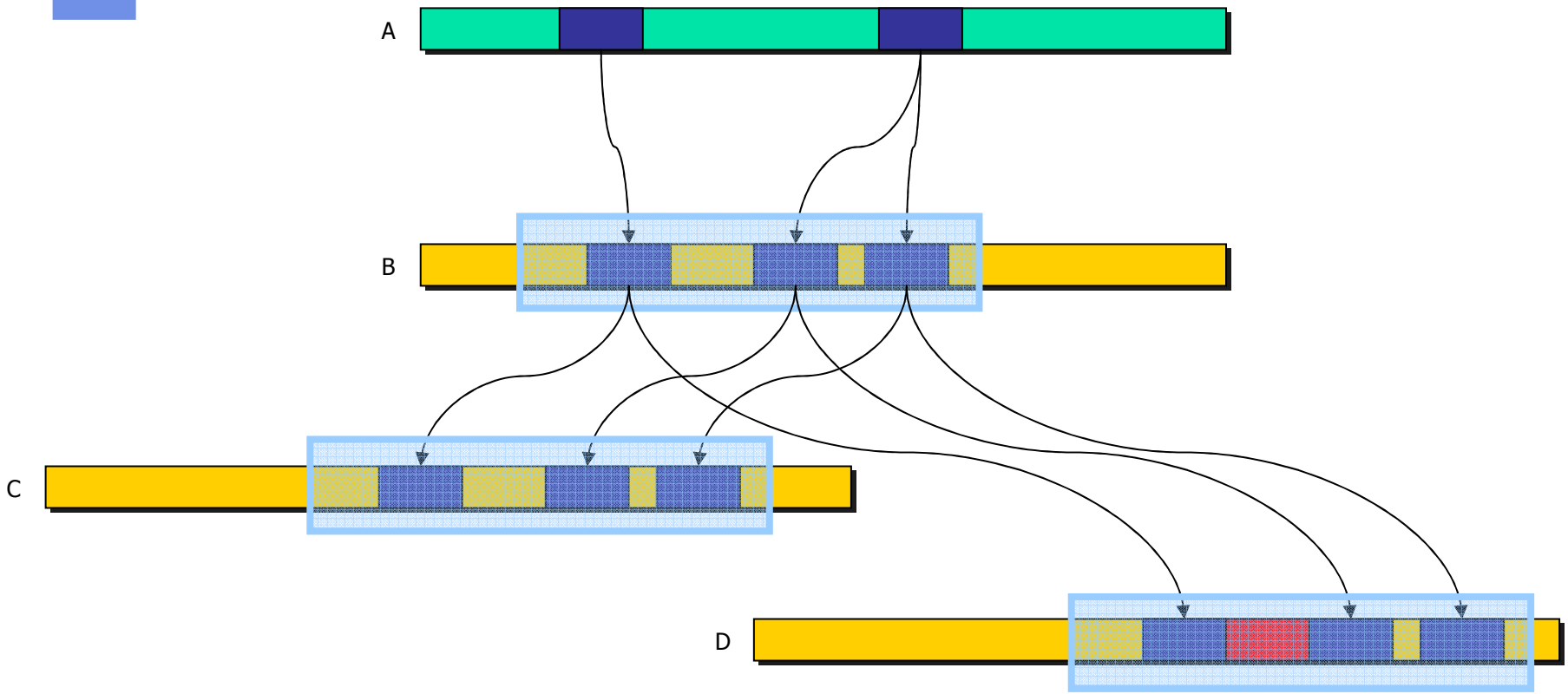
r = Read
w = Write
x = eXecute

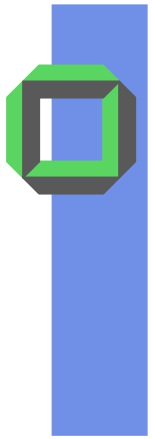


- Mapper may revoke partial access rights
- Cannot extend other's access rights
- Unmap transitively affects mappings



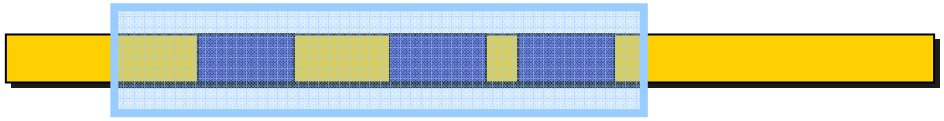
Mapping Regions



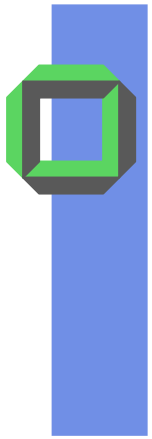


Mapping Regions: Flex Pages

- Abstraction: flex page
  - Contiguous region of virtual address space
    - Sparse physical mappings possible
  - Called **fpage**
  - Abstracts from architecture's page sizes



- Fpage semantics
  - Inseparable object
  - Aligned to its size
  - Size is power of 2, min. $1024=2^{10}$ byte



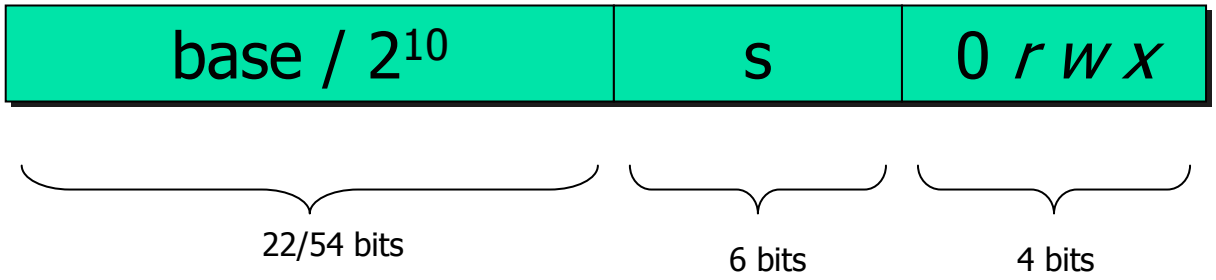
Fpage Encoding

- Special cases
  - Complete address space (base=0, s=1)
  - Nothing: nilpage (0)

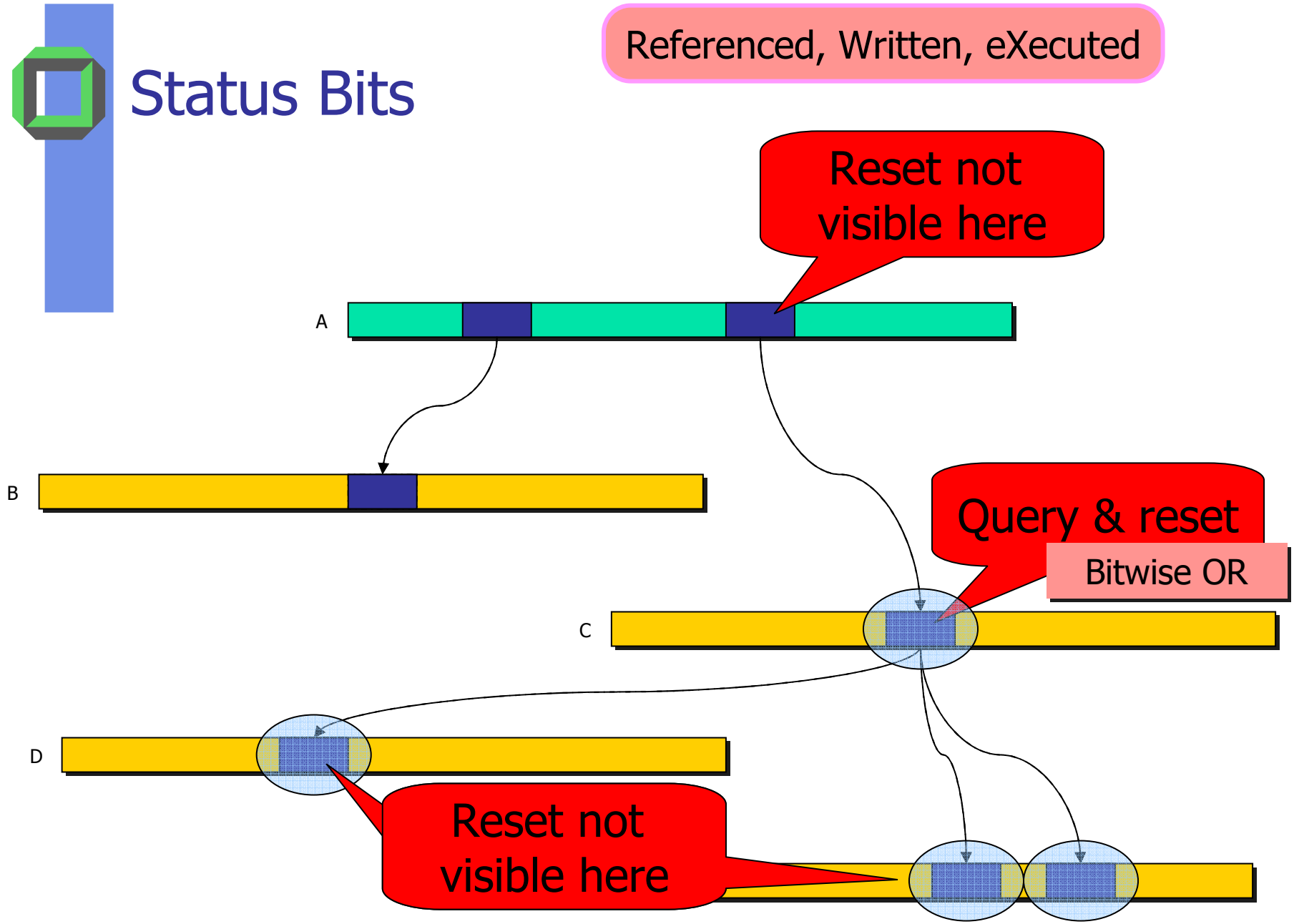
fpage(base, size=2^s)

$s \geq 10$

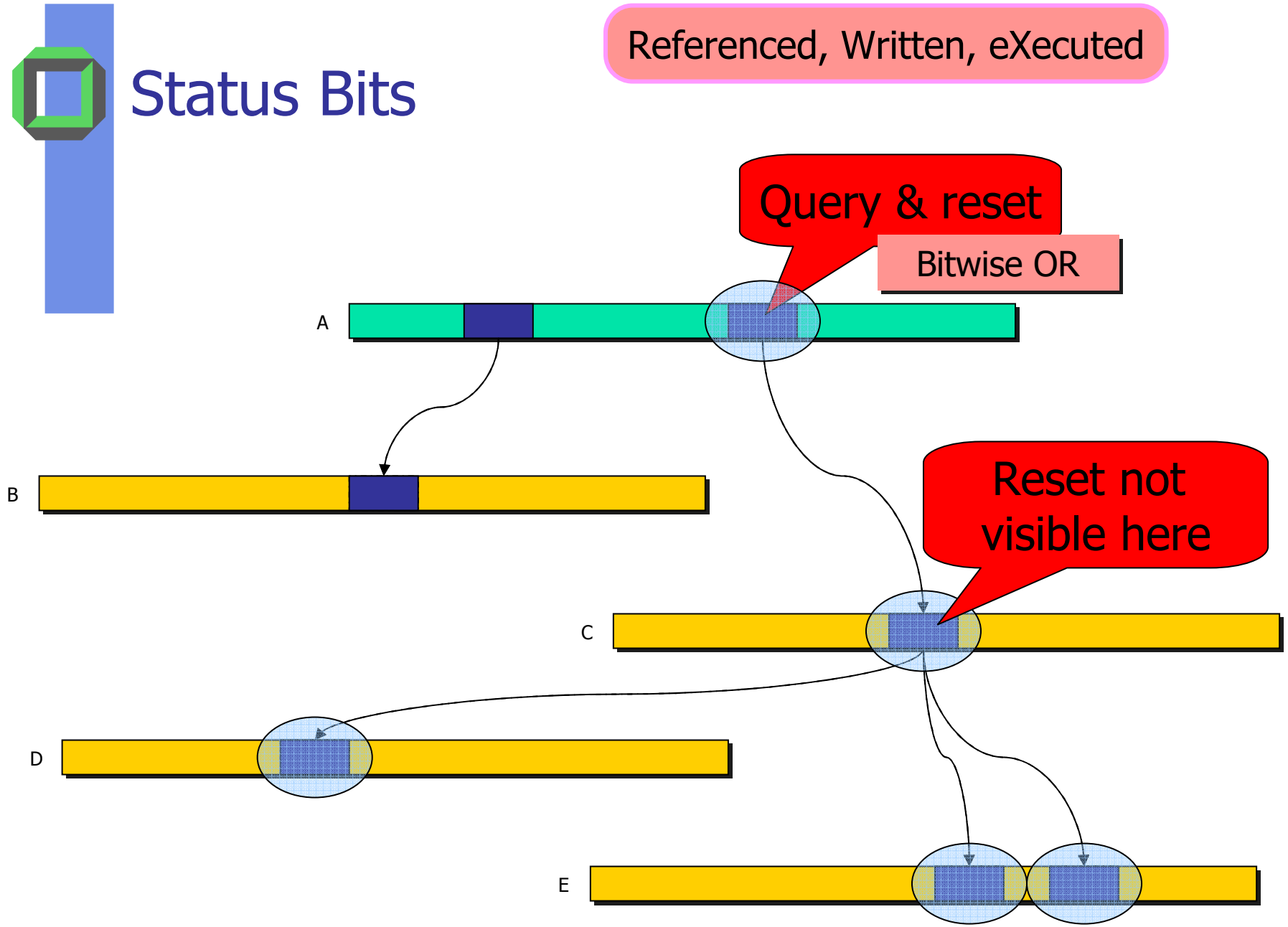
base mod $2^s = 0$

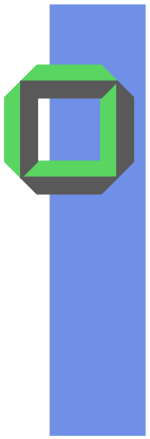


Status Bits



Status Bits

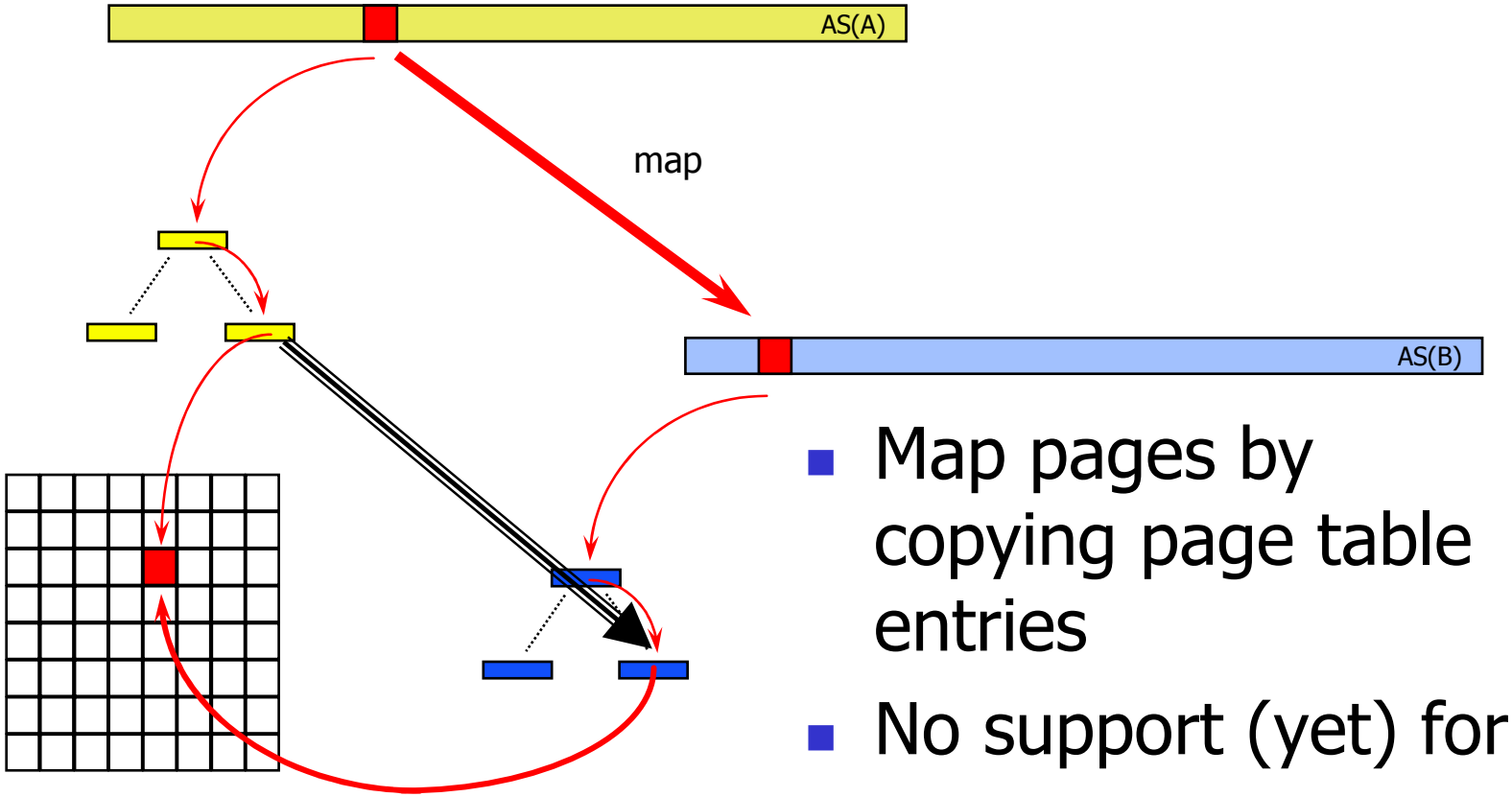




Mapping Regions

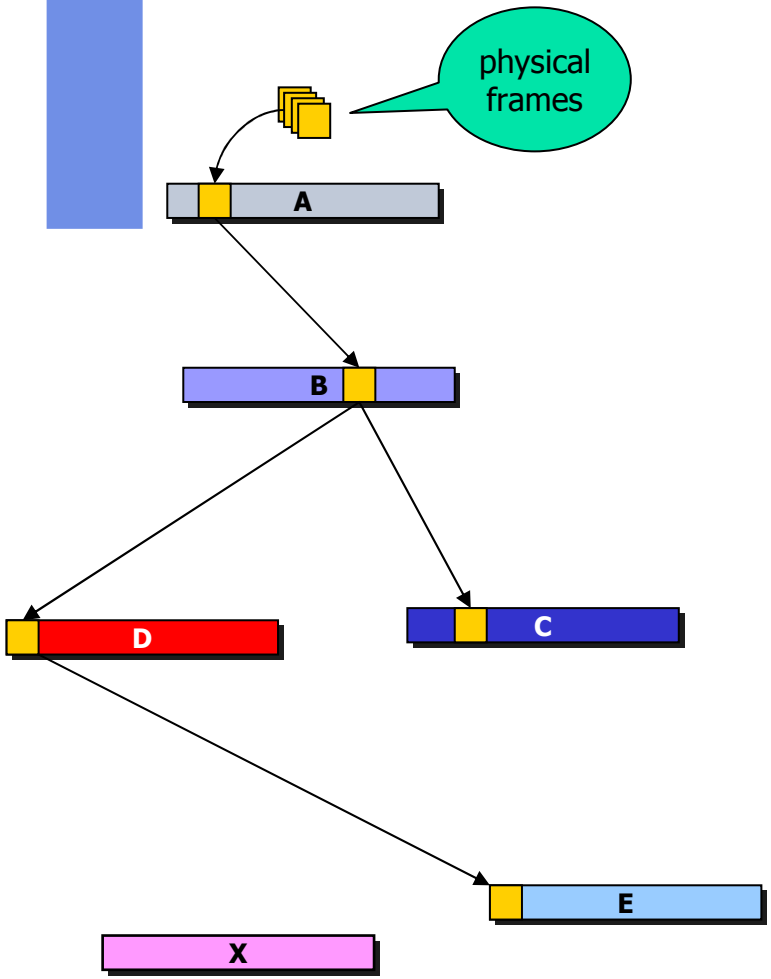
- Implementation
  - Based on page tables
  - Physical page (frame)
    - Basic mapping unit
    - Determines minimum alignment
- Minimum fpage size
  - Physical page size

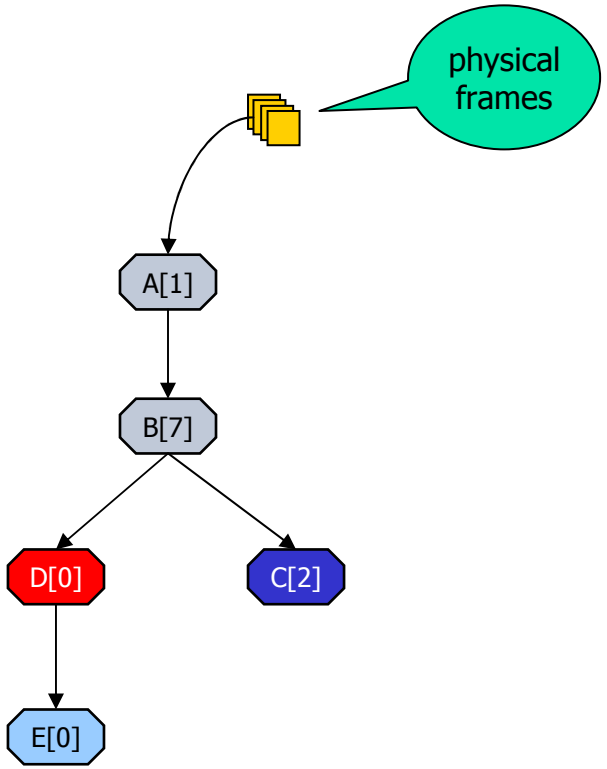
Mapping Pages



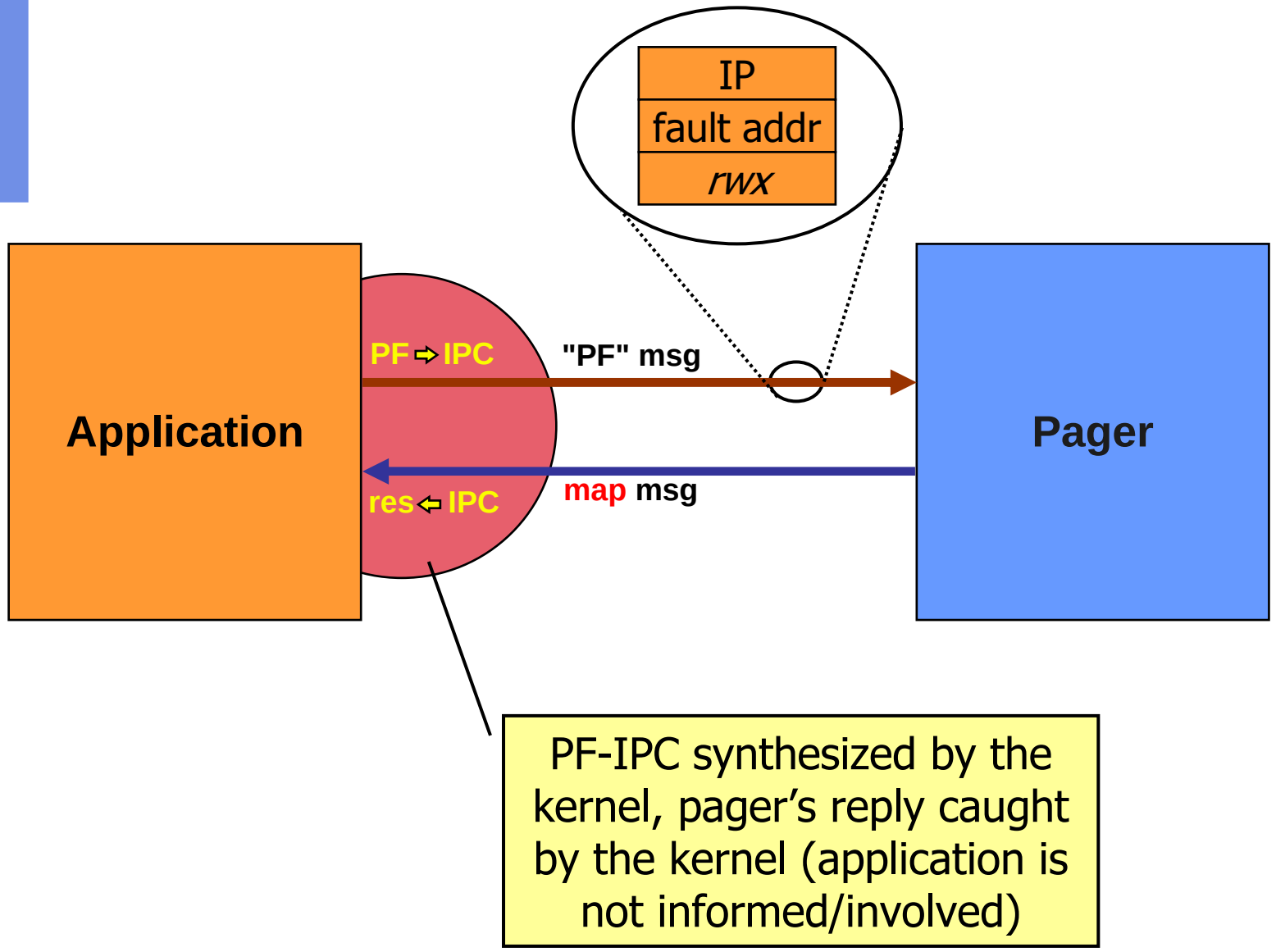
- Map pages by copying page table entries
- No support (yet) for
  - Recursive unmap
  - Combined status bits

Mapping Database

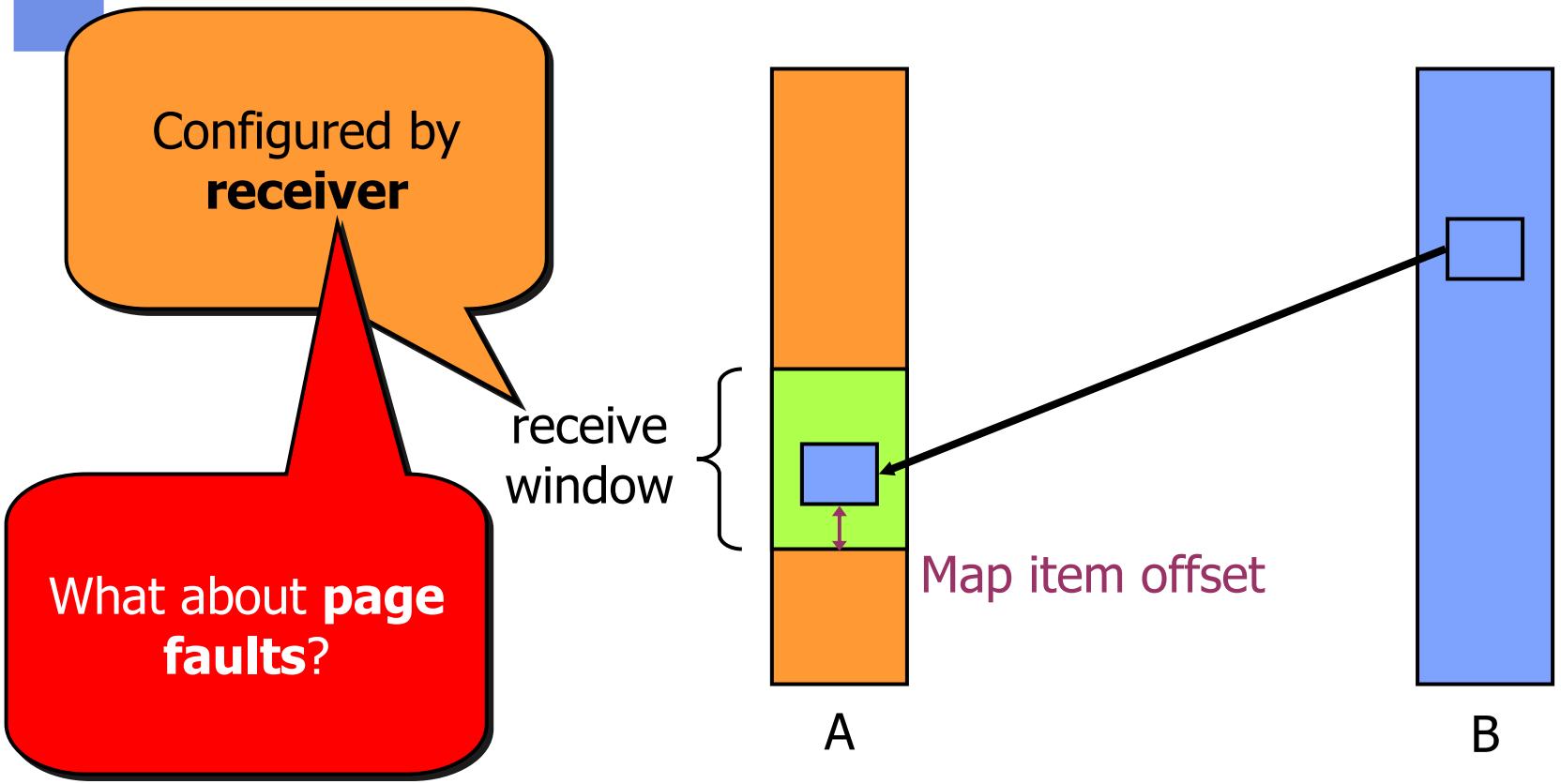




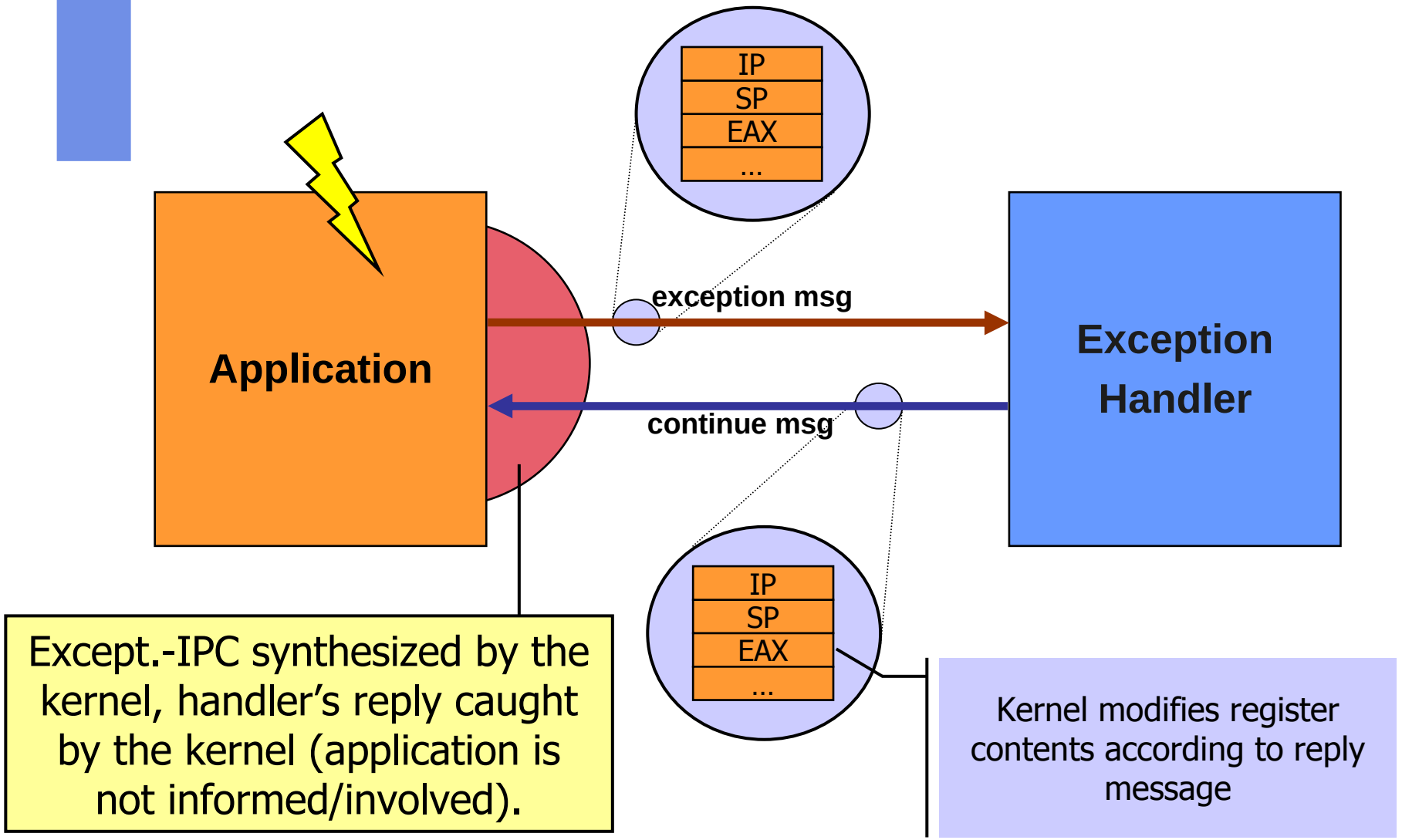
Page Fault IPC

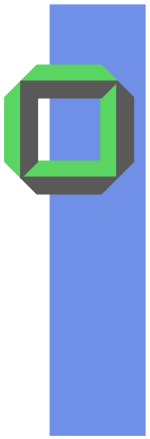


IPC Map



New Exception Handling Model





Other Key Ideas

- Avoid memory
  - No indirection (TCB area)
  - Lazy scheduling
- Make clever use of HW features
  - Sysenter/sysexit
  - Segmentation (➔ small spaces)
- Serialize recursive algorithms
  - “Recursive” unmap