Karlsruhe Institute of Technology

Microkernel Construction
I.11 – Small Address Spaces (Special Optimization for Untagged TLBs)

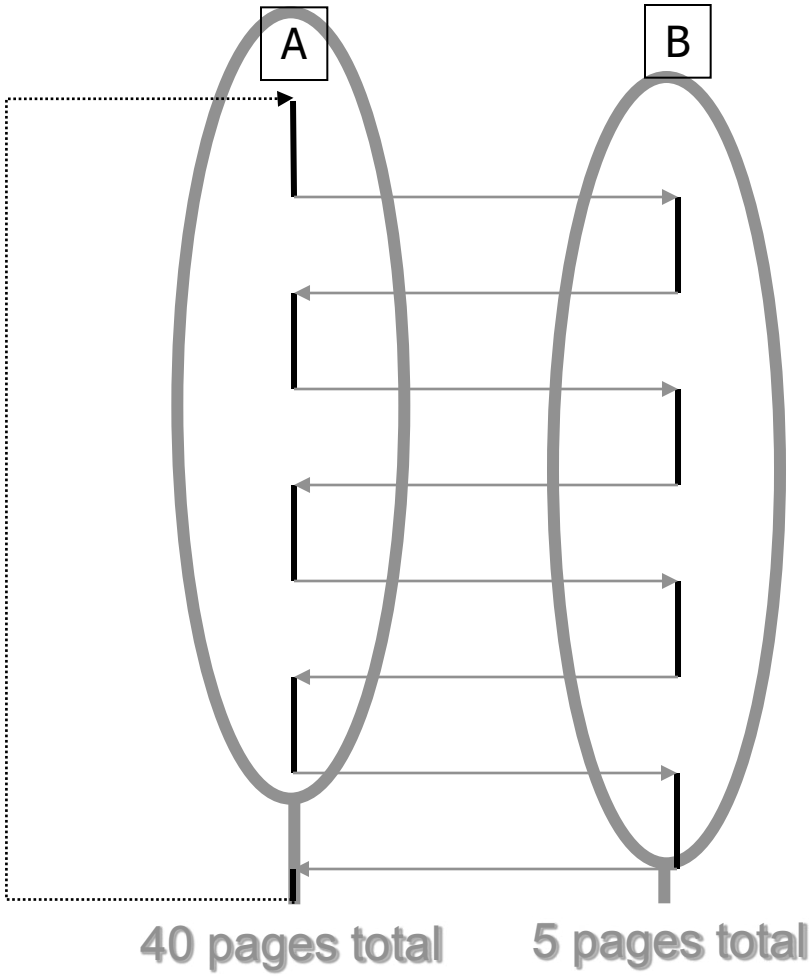
Lecture Summer Term 2017
Wednesday 15:45-17:15 R 131, 50.34 (INFO)

Jens Kehne | Marius Hillenbrand
Operating Systems Group, Department of Computer Science

Untagged TLB Context-Switch Costs

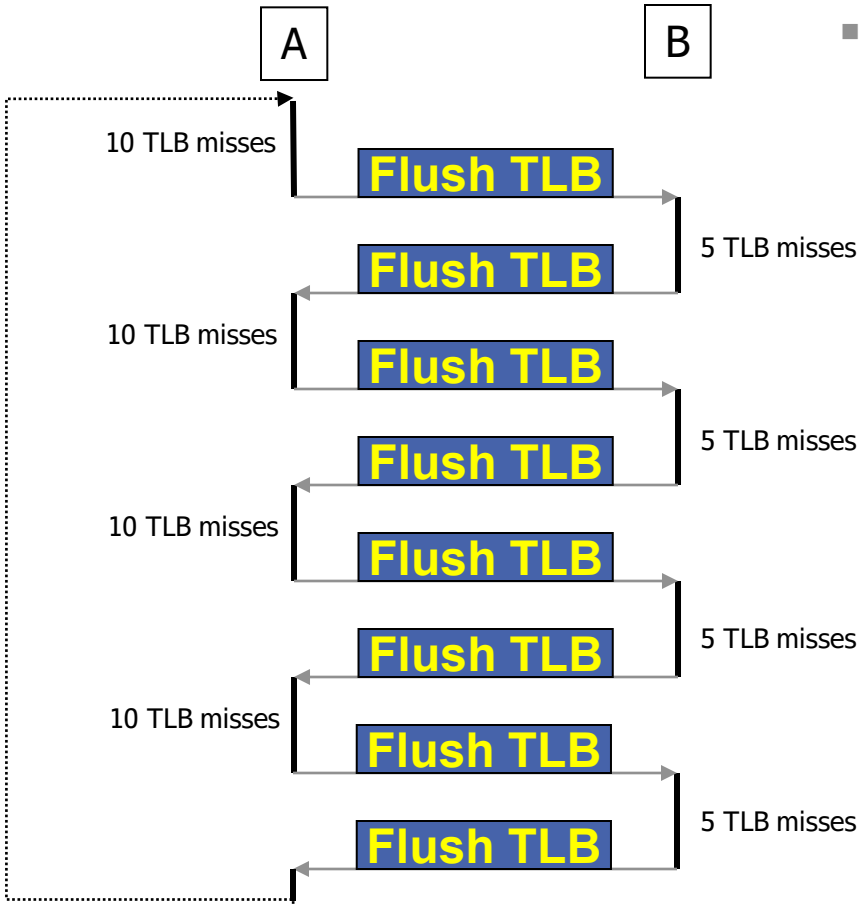
	486 … PIII	Pentium 4
Enter/exit kernel	40 … 200 cycles	150 … 200 cycles
Switch thread	≈ 10 cycles	≈ 10 cycles
Switch address space		
Flush TLB	≈ 50 … 80 cycles	≈ 230 … 250 cycles
Refill TLB	6 … 96 TLB refills 15 … 40 cycles/refill ≈ 100 … 4000 cycles	6 … 192 TLB refills 15 … 500 cycles/refill ≈ 100 … 96000 cycles
Refill L1 caches (P4 only, not P3/Core)		12 K trace cache 8 K data cache 15 … 25 cycles/refill ≈ 100 … 16000 cycles

Untagged TLB Context-Switch Costs



- Even when calling a thread with a very **small TLB working set**
  - Thread A frequently calls thread B
  - Working sets
    - Thread A: 4 different sets of 10 pages between B-calls
    - Thread B: always the same 5 pages

Untagged TLB Context-Switch Costs

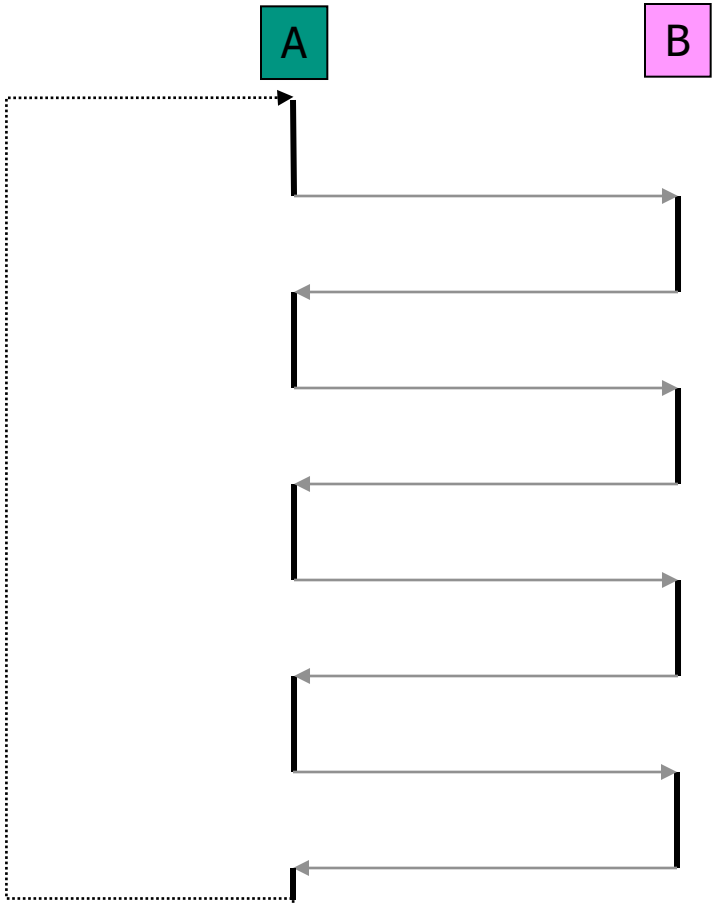


- Even when calling a thread with a very **small TLB working set**
  - Thread A frequently calls thread B
  - Working sets
    - Thread A: 4 different sets of 10 pages between B-calls
    - Thread B: always the same 5 pages

	[cycles]
8 IPCs (w/o AS costs):	1440
60 TLB misses:	900 … 30000
8 TLB flushes:	400 … 2000

Untagged TLB total: 2740 … 33440

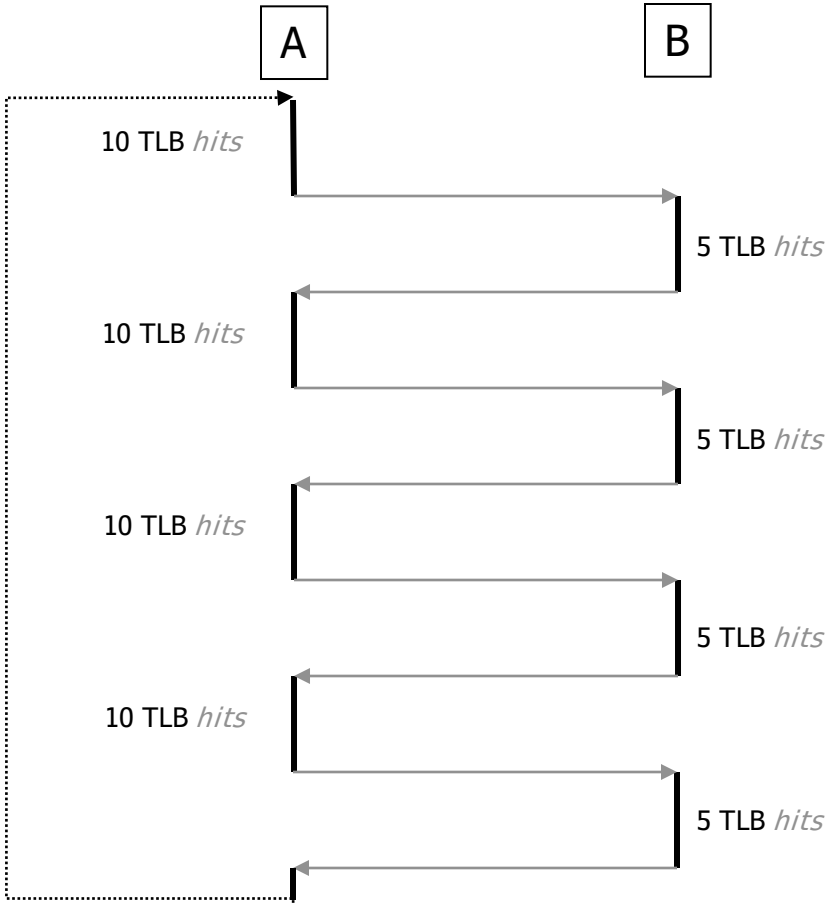
Tagged TLB



- Tagged TLB:
  - Associate Address-Space ID with translations
  - No flushing during AS switch
  - Not available for x86 address spaces (only with virtualization extensions)

ASID	VADDR	PADDR	sz	rwx	U/K
A					
B					
B					
A					
B					
A					
A					

Untagged TLB Context-Switch Costs



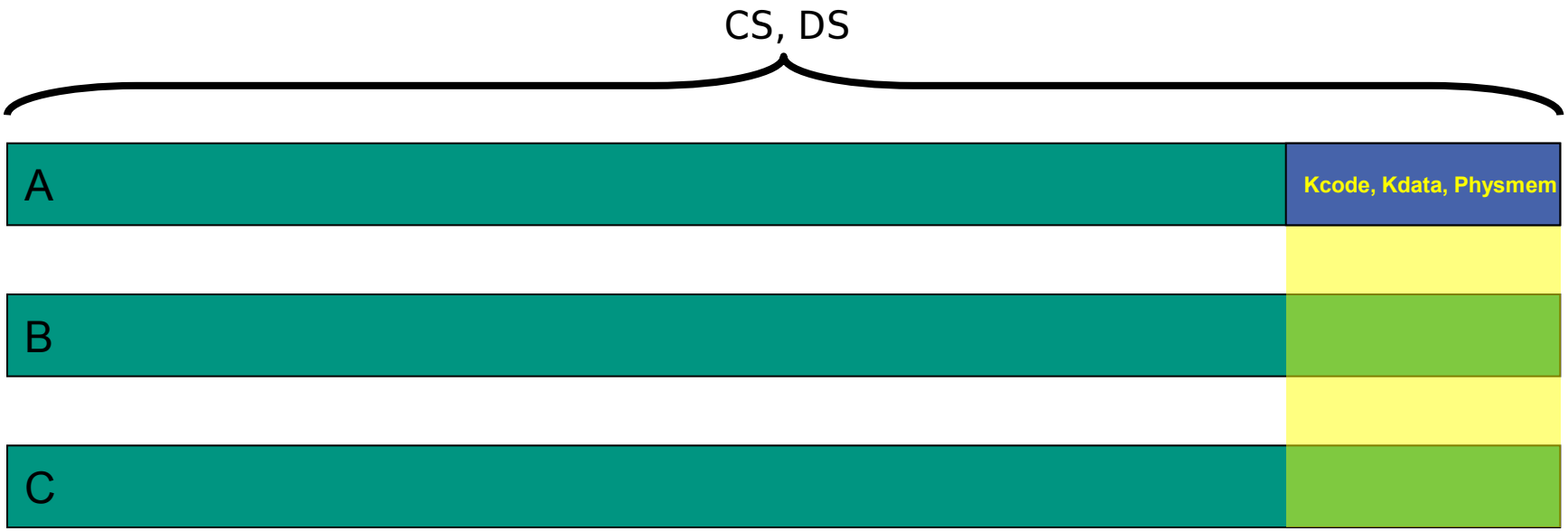
- Even when calling a thread with a very **small TLB working set**
  - Thread A frequently calls thread B
  - Working sets
    - Thread A: 4 different sets of 10 pages between B-calls
    - Thread B: always the same 5 pages

	[cycles]
8 IPCs (w/o AS costs):	1440
60 TLB misses:	900 … 30000
8 TLB flushes:	400 … 2000
Untagged TLB total:	**2740 … 33440**
Tagged TLB total:	**≈ 1500**

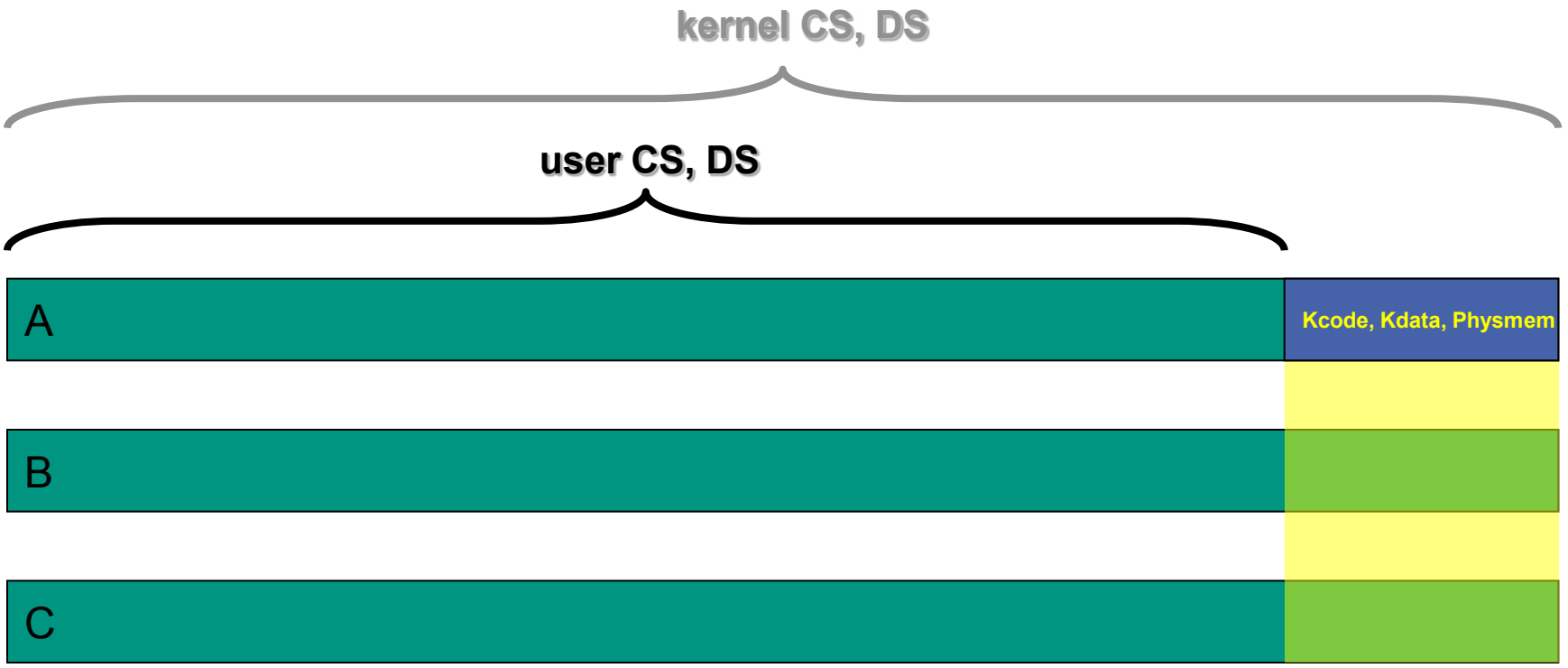
SMALL ADDRESS SPACES ON X86

How to emulate tagged TLBs?

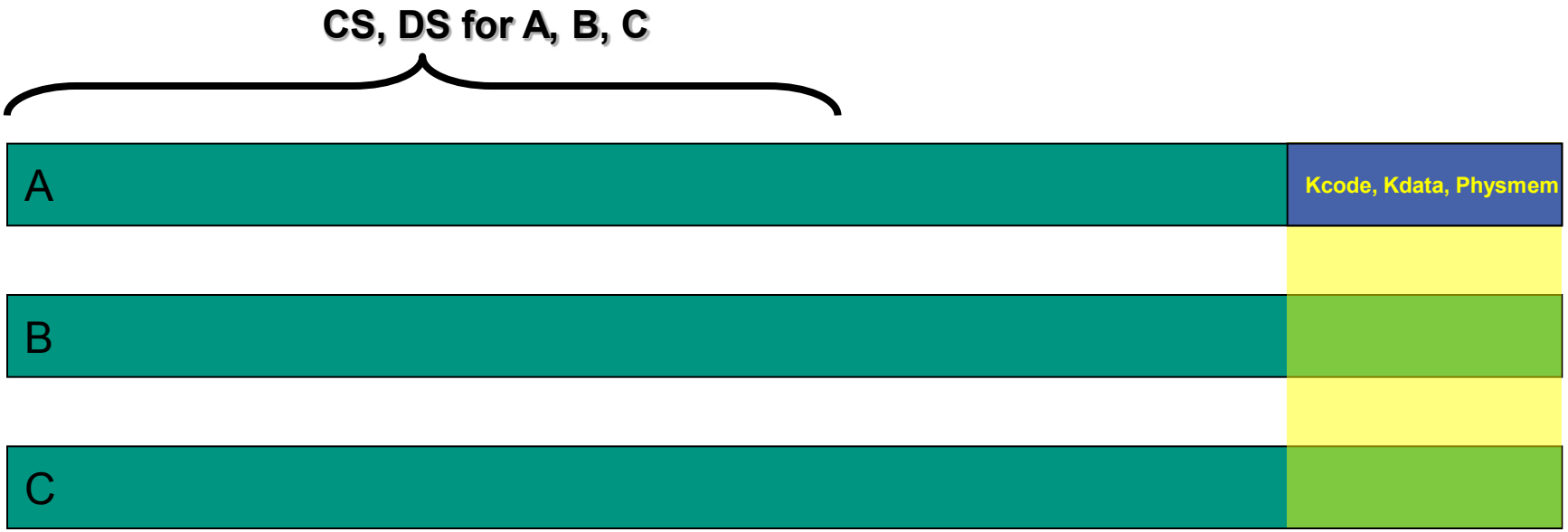
Address Spaces



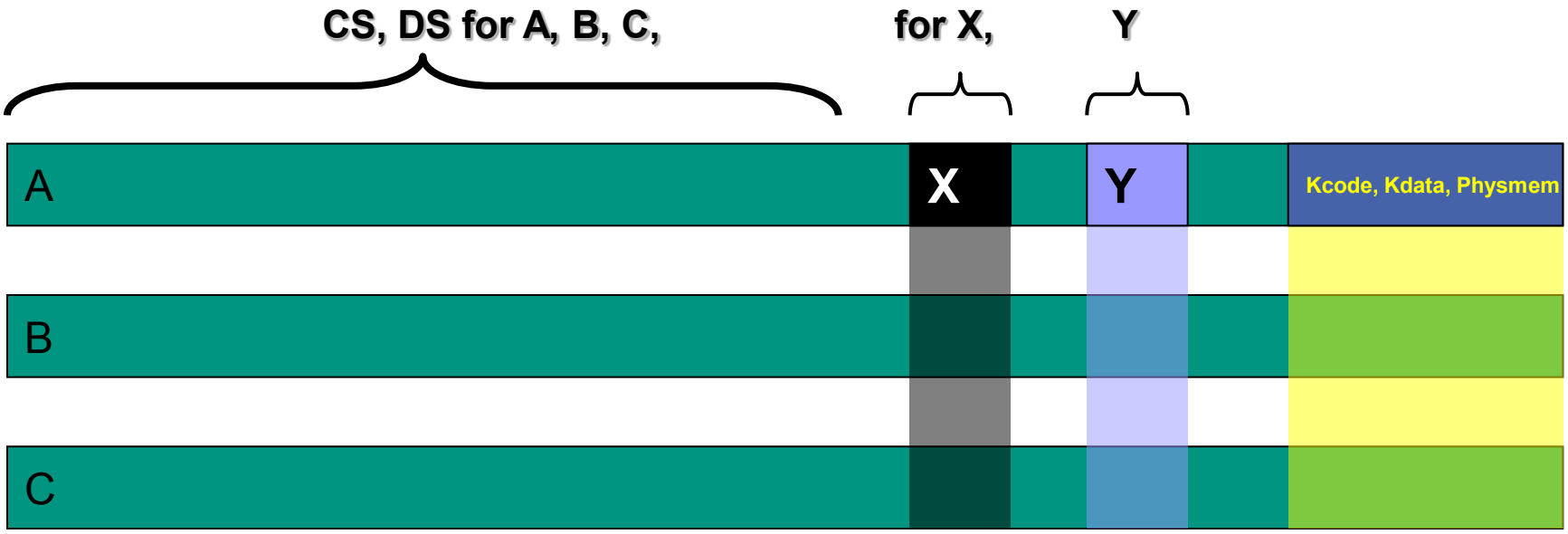
Address Spaces



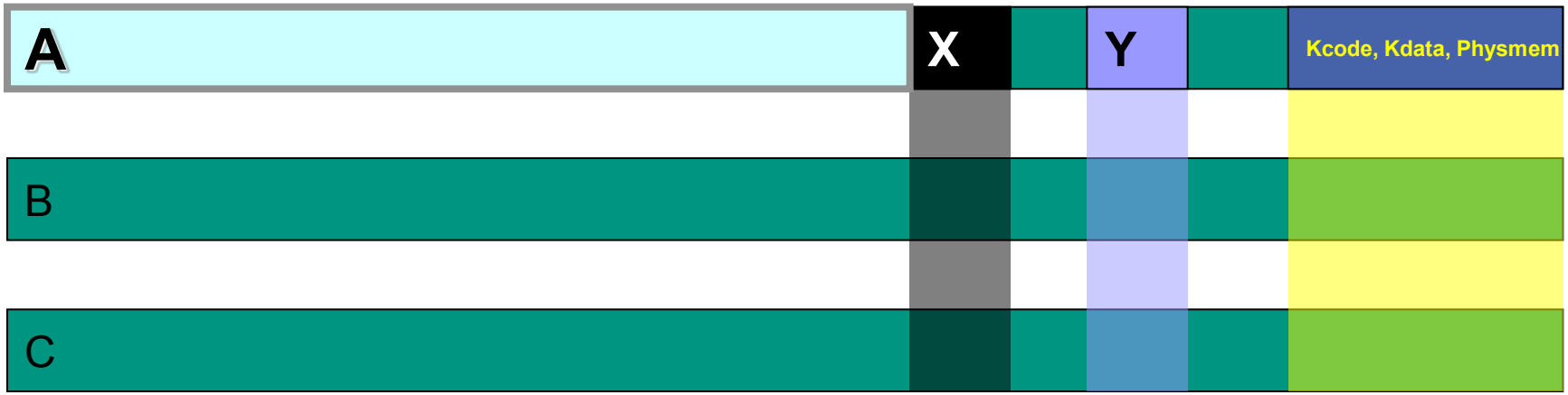
Small Address Spaces



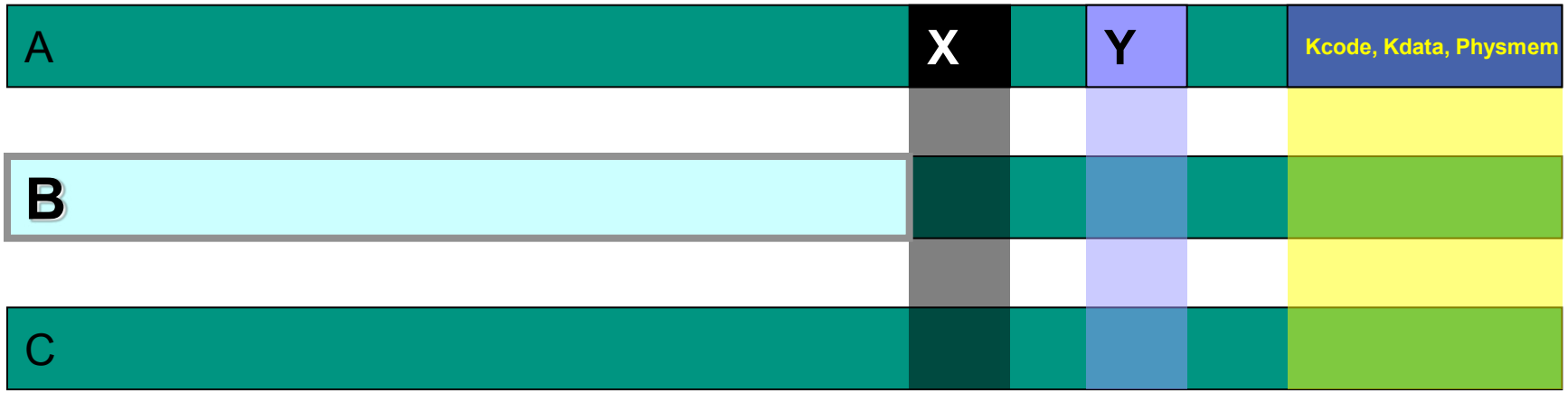
Small Address Spaces



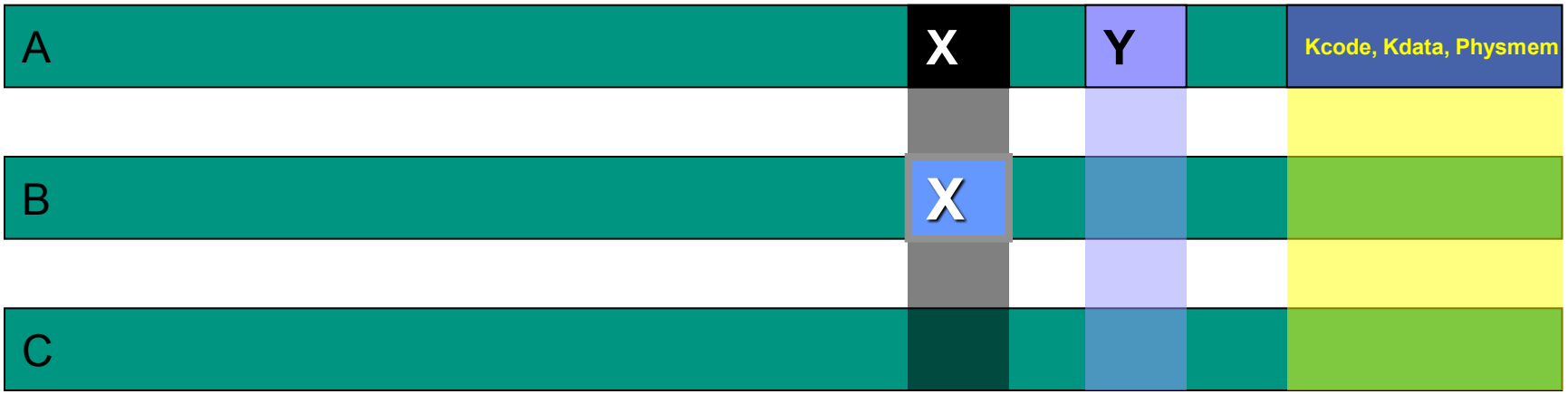
Small Address Spaces



Small Address Spaces

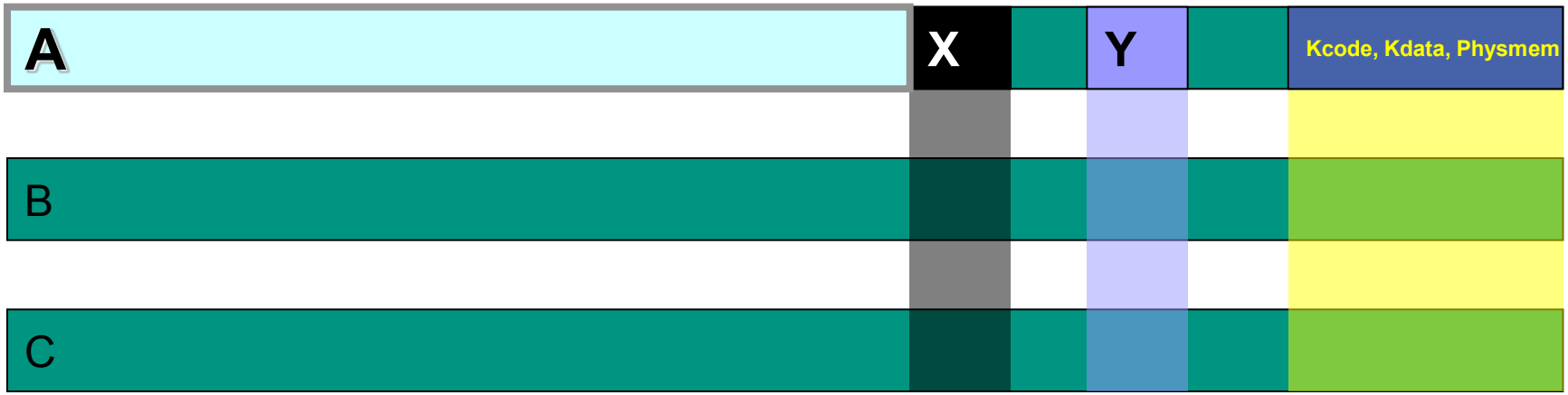


Small Address Spaces

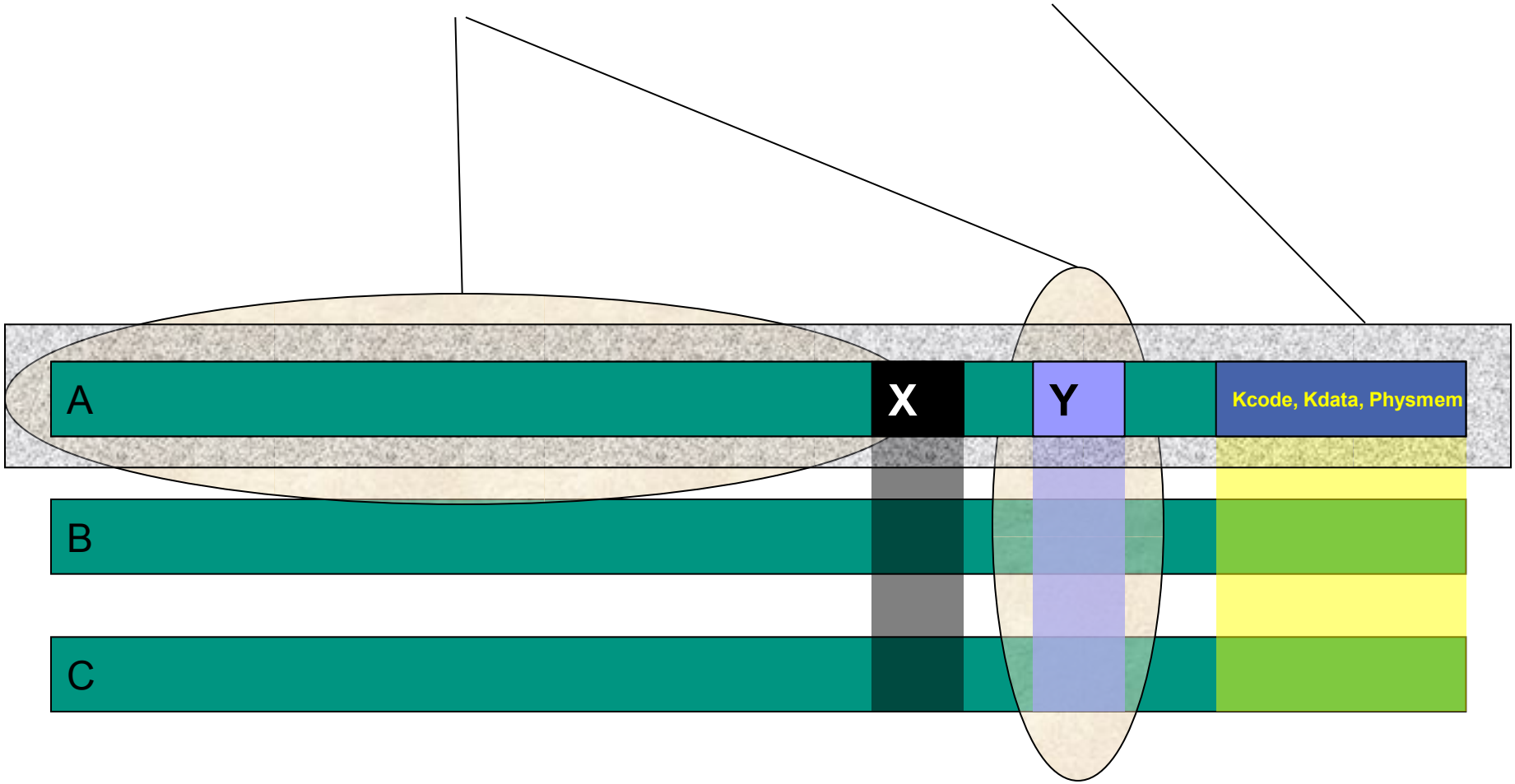


Small Address Spaces

Small Address Spaces



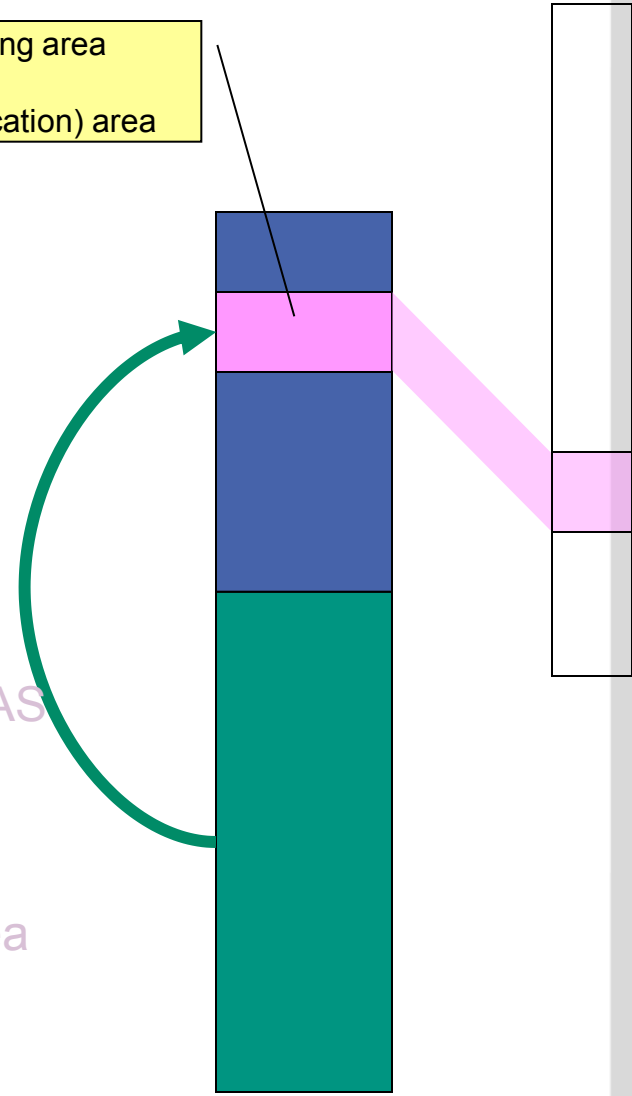
Address Space (AS) vs. Hardware AS (HwAS)



Long-IPC Implementation Revisited

Temporary mapping area
alias
kernel com(munication) area

Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area

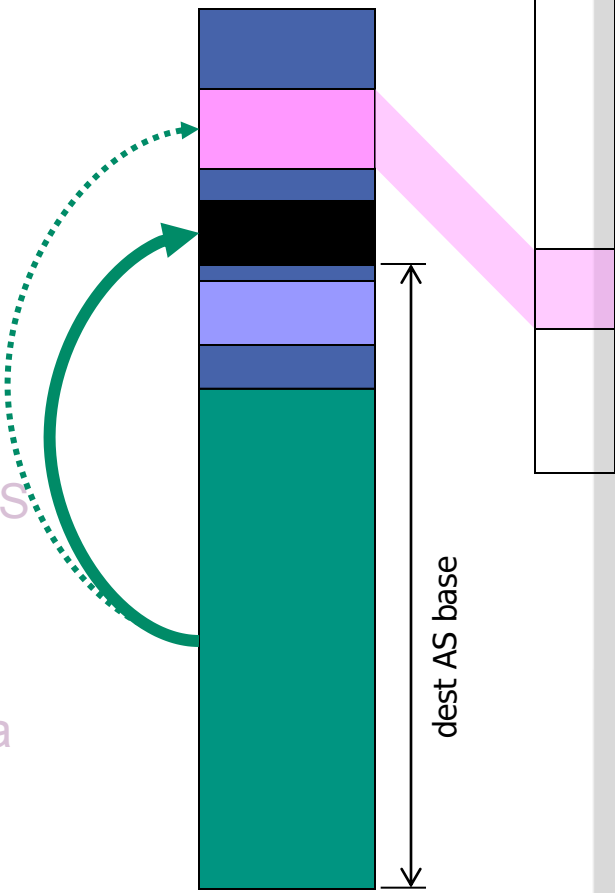


then -direct-

Long-IPC Implementation Revisited

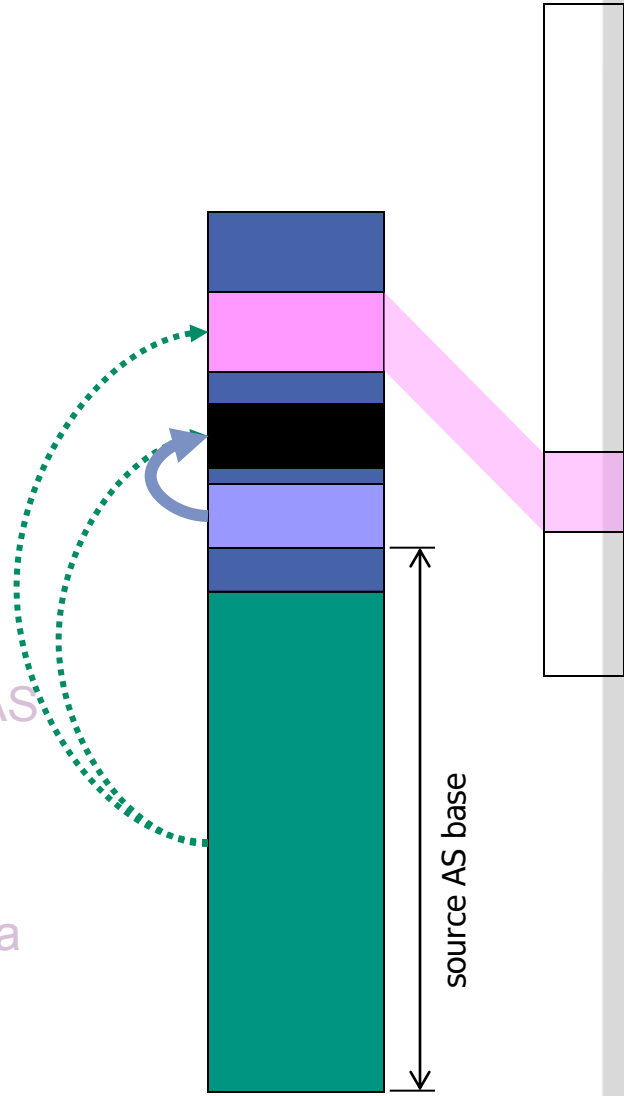
Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area
large / small	direct **- no HwAS switch -**	0	dest AS base

then -direct-



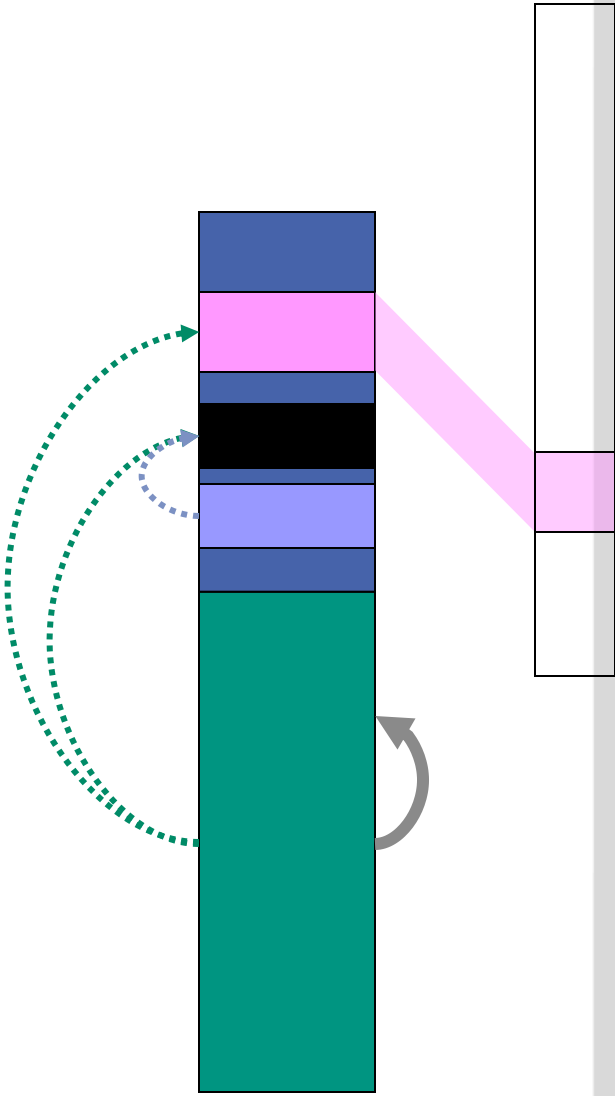
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area
large / small	direct **- no HwAS switch -**	0	dest AS base
small / small	direct **- no HwAS switch -**	source AS base	dest AS base



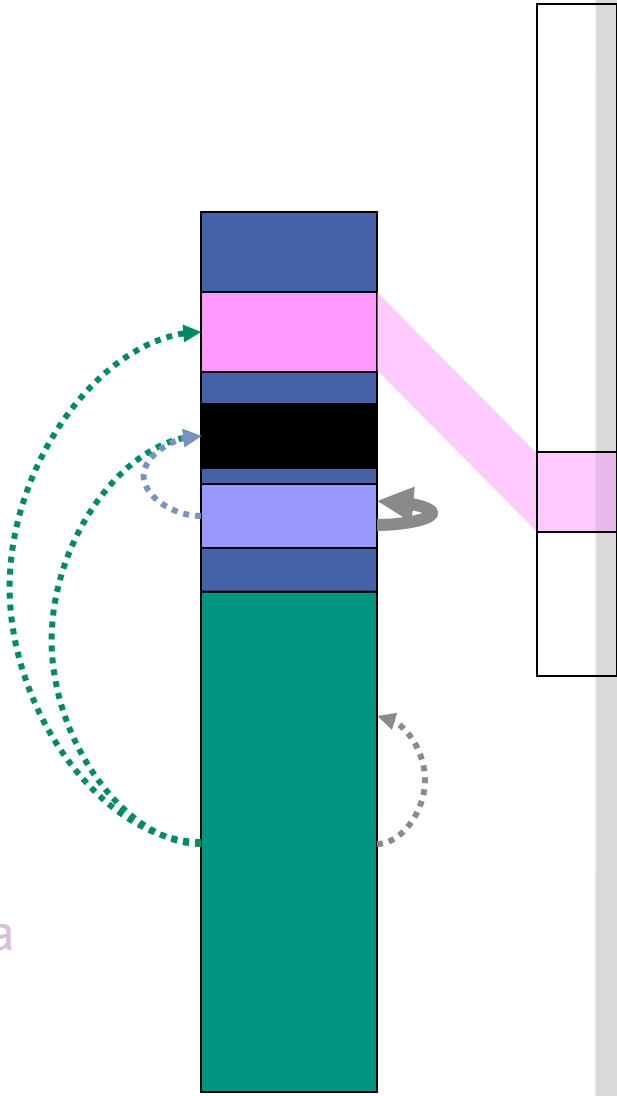
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area
large / small	direct **- no HwAS switch -**	0	dest AS base
small / small	direct **- no HwAS switch -**	source AS base	dest AS base
large / same	direct **- no HwAS switch -**	0	0



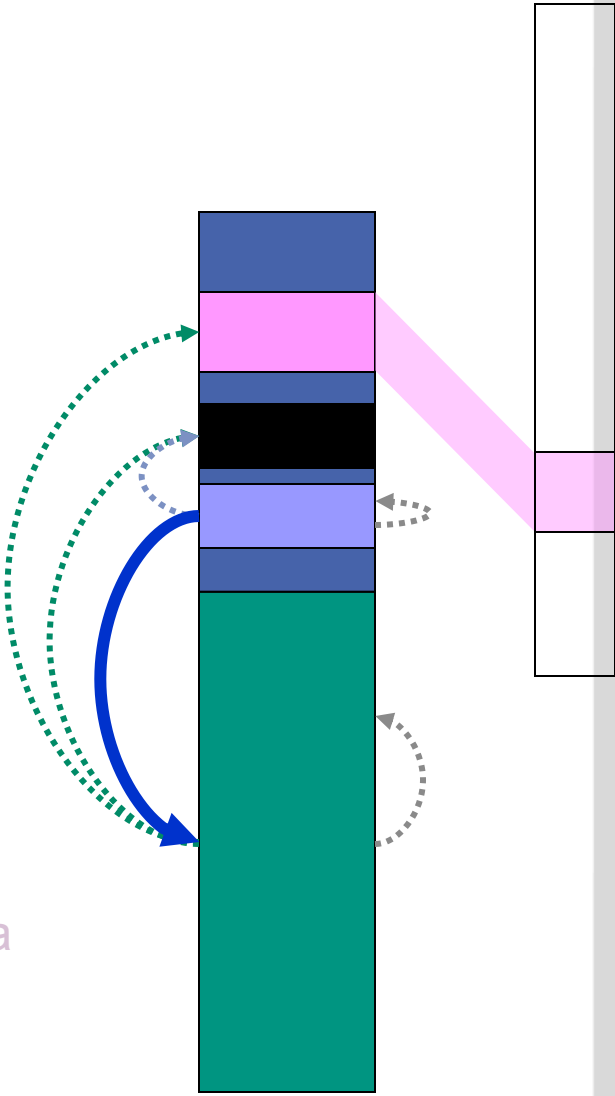
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area
large / small	direct **- no HwAS switch -**	0	dest AS base
small / small	direct **- no HwAS switch -**	source AS base	dest AS base
large / same	direct **- no HwAS switch -**	0	0
small / same	direct **- no HwAS switch -**	source AS base	source AS base



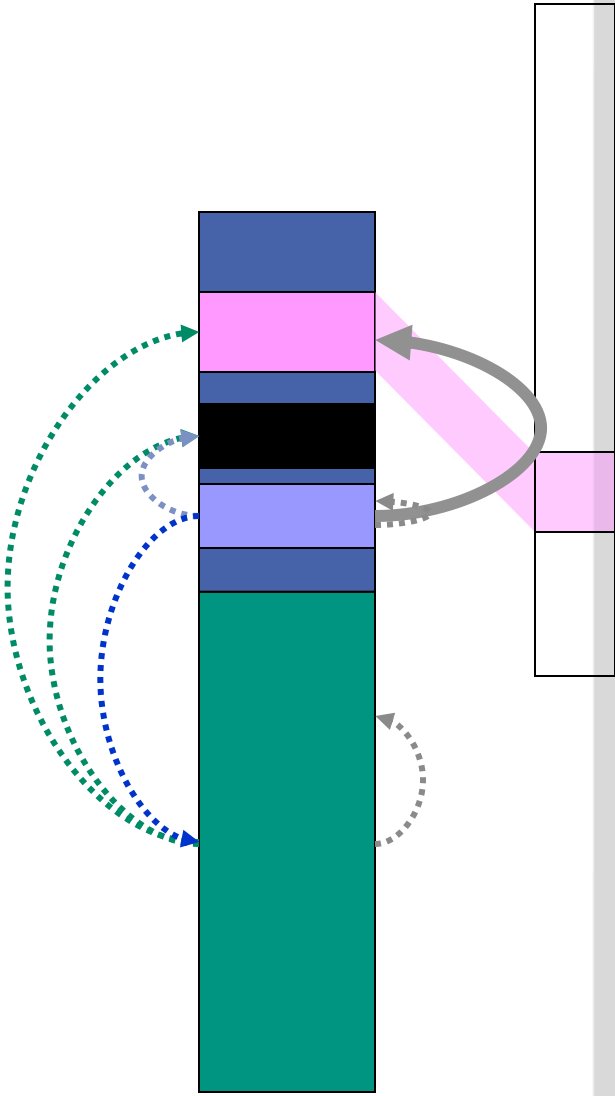
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area
large / small	direct **- no HwAS switch -**	0	dest AS base
small / small	direct **- no HwAS switch -**	source AS base	dest AS base
large / same	direct **- no HwAS switch -**	0	0
small / same	direct **- no HwAS switch -**	source AS base	source AS base
small / large = Current HwAS	direct **- no HwAS switch -**	source AS base	0



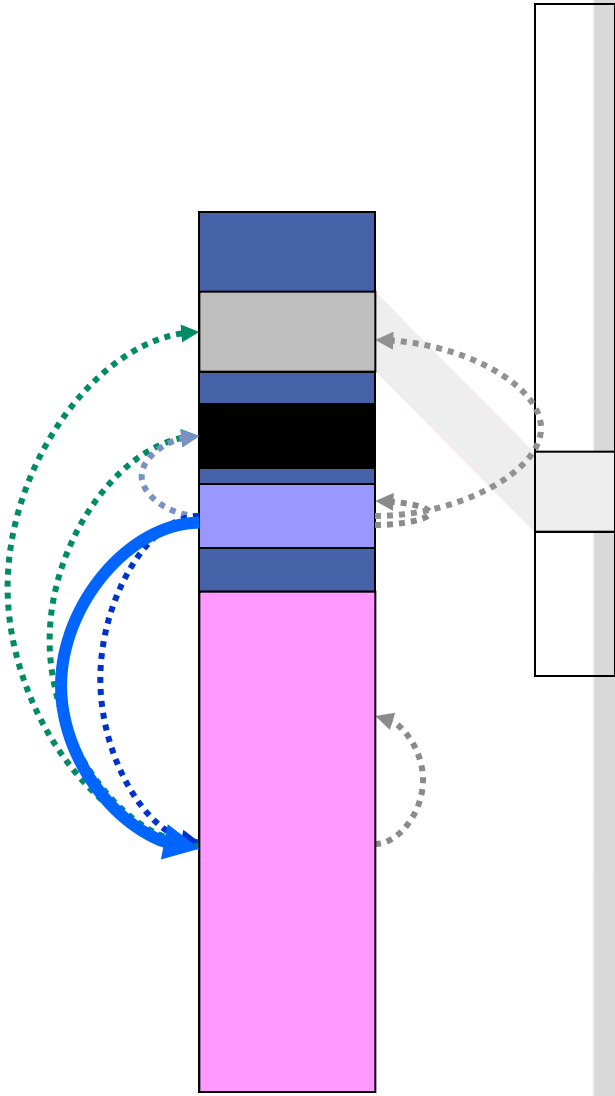
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area
large / small	direct **- no HwAS switch -**	0	dest AS base
small / small	direct **- no HwAS switch -**	source AS base	dest AS base
large / same	direct **- no HwAS switch -**	0	0
small / same	direct **- no HwAS switch -**	source AS base	Source AS base
small / large = Current HwAS	direct **- no HwAS switch -**	Source AS base	0
small / large ≠ Current HwAS	temp mapping **switch HwAS**	source AS base	kernel com area



Long-IPC Implementation Revisited

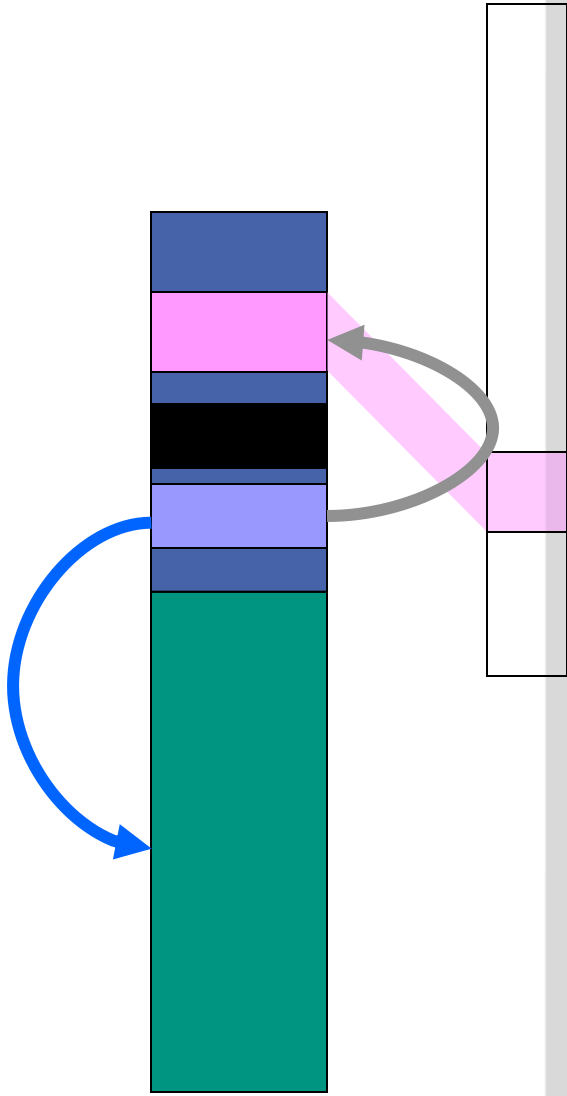
Source / Destination	Method	Source offset	Dest
large / large	temp mapping **switch HwAS**	0	kernel com area
large / small	direct **- no HwAS switch -**	0	dest AS base
small / small	direct **- no HwAS switch -**	source AS base	dest AS base
large / same	direct **- no HwAS switch -**	0	0
small / same	direct **- no HwAS switch -**	source AS base	source AS base
small / large = Current HwAS	direct **- no HwAS switch -**	source AS base	0
small / large ≠ Current HwAS	temp mapping **switch HwAS** **– or –** ***first* switch HwAS** ***then* direct**	source AS base	kernel com area **– or –** 0



Long-IPC Implementation Revisited

- A **global bit** in page table entry indicates that TLB entry should not be flushed on TLB flushes
  - Used for not flushing kernel entries

Source / Destination	Method	Source	Dest
large / large			
large / small			
small / small			
large / same			
small / same	- no HwAS sw	base	base
small / large = Current HwAS	direct - no HwAS sw	source base	0
small / large ≠ Current HwAS	temp mapping **switch HwAS** **– or –** ***first* switch HwAS** ***then* direct**	source AS base	kernel com area **– or –** 0

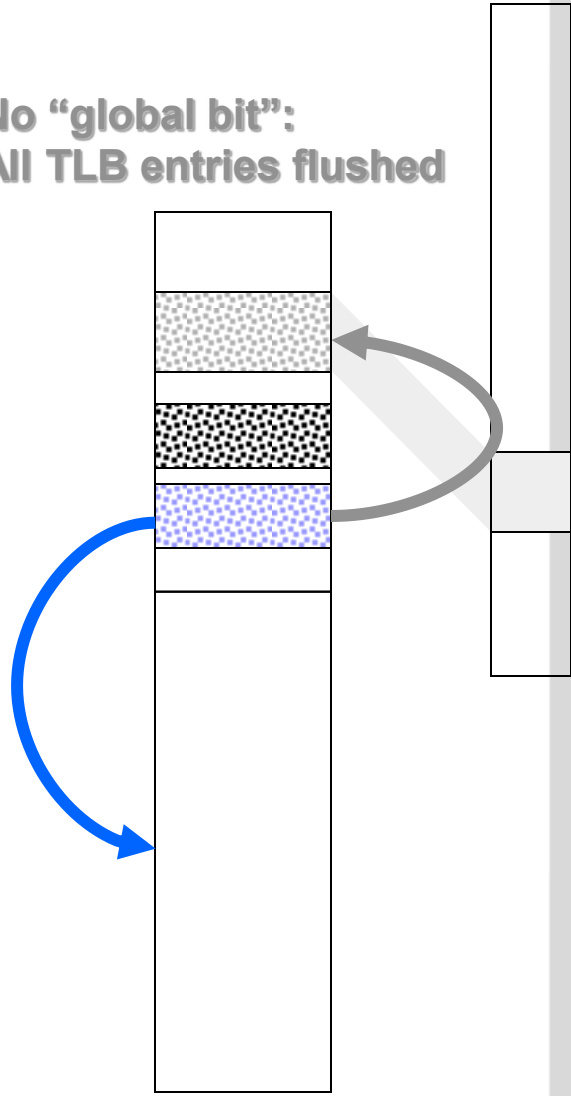


Long-IPC Implementation Revisited

- A **global bit** in page table entry indicates that TLB entry should not be flushed on TLB flushes
  - Used for not flushing kernel entries

Source / Destination	Method	Source	Dest
large / large			
large / small			
small / small			
large / small			
small / same			
small / large = Current HwAS	direct - no HwAS sw	source base	0
small / large ≠ Current HwAS	temp mapping **switch HwAS** **– or –** ***first* switch HwAS** ***then* direct**	source AS base	kernel com area **– or –** 0

No “global bit”:
All TLB entries flushed

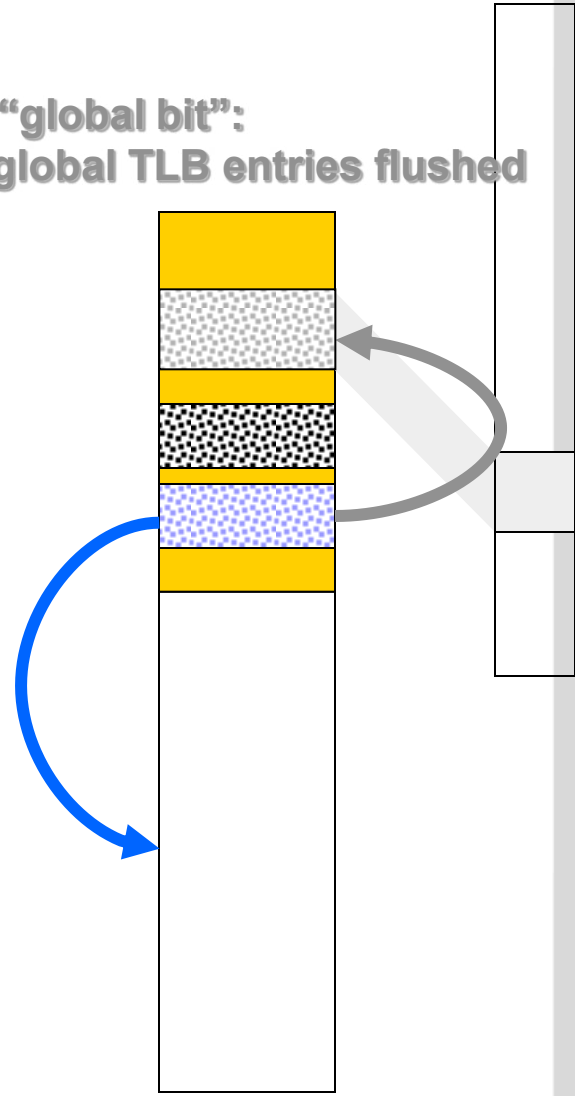


Long-IPC Implementation Revisited

- A **global bit** in page table entry indicates that TLB entry should not be flushed on TLB flushes
  - Used for not flushing kernel entries

Source / Destination	Method	Source	Dest
large / large	- no HwAS sw	base	base
large / small	direct	source	0
small / small	- no HwAS sw	base	
large / same			
small / same			
small / large = Current HwAS			
small / large ≠ Current HwAS	temp mapping **switch HwAS** **– or –** ***first* switch HwAS** ***then* direct**	source AS base	kernel com area **– or –** 0

With “global bit”:
Non-global TLB entries flushed

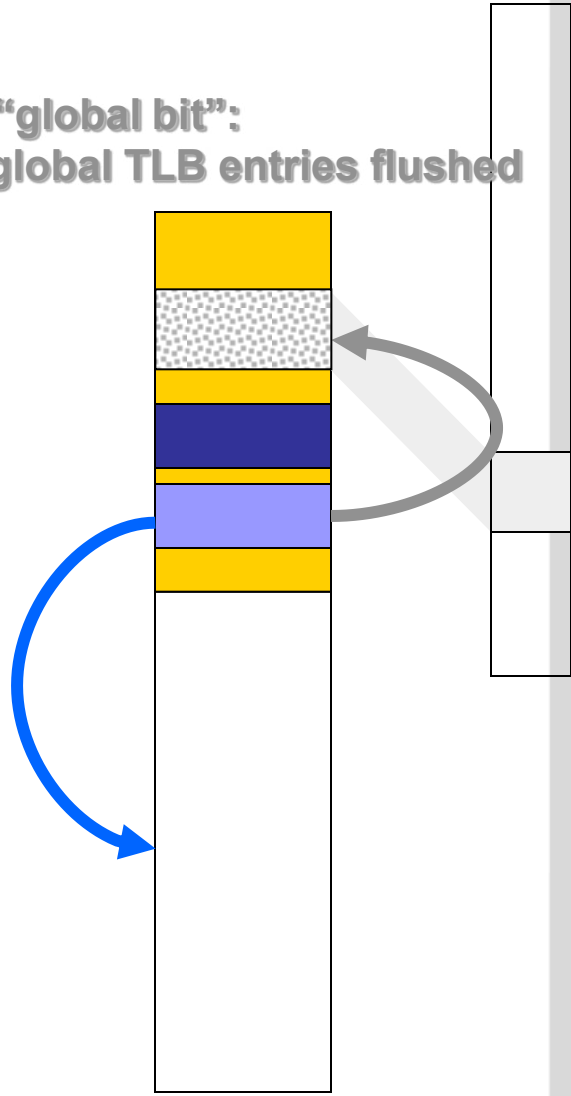


Long-IPC Implementation Revisited

- A **global bit** in page table entry indicates that TLB entry should not be flushed on TLB flushes
  - Used for not flushing kernel entries
- small spaces are global over all hardware address spaces
  - Mark small spaces' PTEs as global

Source / Destination	Method	Source	Dest
large / large			
large / small			
small / small			
large / same			
small / same	- no HwAS sw	base	base
small / large = Current HwAS	direct - no HwAS sw	source base	0
small / large ≠ Current HwAS	temp mapping **switch HwAS** **– or –** ***first* switch HwAS** ***then* direct**	source AS base	kernel com area **– or –** 0

With "global bit":
Non-global TLB entries flushed



Operating Systems Group
Department of Computer Science

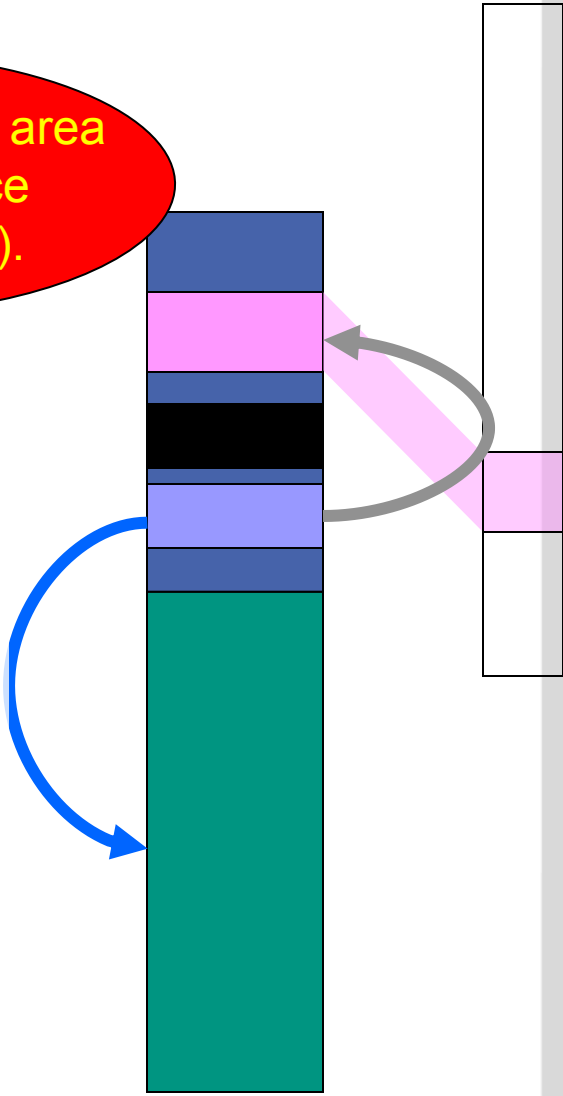
Long-IPC Implementation Revisited

TLB misses on com area and in dest space (after the switch).

With “global PTE entries”, TLB misses only in dest space.

Without global pages, TLB misses in source and dest.

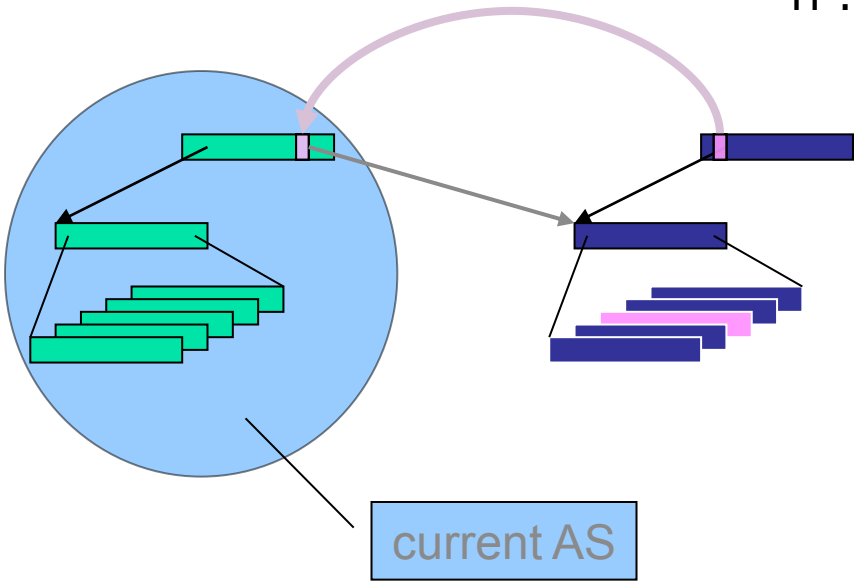
Source / Destination	Method		
large / large	temp mapping **switch HwAS**	0	
large / small	direct **no HwAS switch -**	0	est AS base
sm	ch -	AS	dest AS base
	ch		0
small	HwAS switch -	source base	source AS base
small / large = Current HwAS	HwAS switch -	source base	0
small / large ≠ Current HwAS	temp mapping switch HwAS **– or –** ***first* switch HwAS** ***then* direct**	source AS base	kernel com area **– or –** 0



Temporary Mapping Revisited

Leave thread:

```
if mytcb.partner ≠ nilthread then
    myPDE.TMarea := nil ;
    if dest AS = my AS then
        flush TLB
    fi
fi .
```



Optimization: avoids second TLB flush if subsequent thread and AS switch would flush TLB anyway

Evicting the Temporary Mapping Area

- We must evict the temp. mapping from the TLB when switching from … to …
- Depending on whether small spaces use the temp. mapping / or not at all

from \ to	small AS	Active large AS	Inactive large AS
small AS	Yes / No	Yes / No[1]	No / No
large AS	Yes / Yes[2]	Yes	No

[1]: assuming the temp. mapping is invalidated when switching to the small space
[2]: to prevent T3 from using T1's mapping area after T1 (large AS A) → T2 (small AS B) → T3 (large AS A), possible due to [1]

Temporary Mapping Revisited

Leave thread:

if mytcb.partner ≠ nilthre

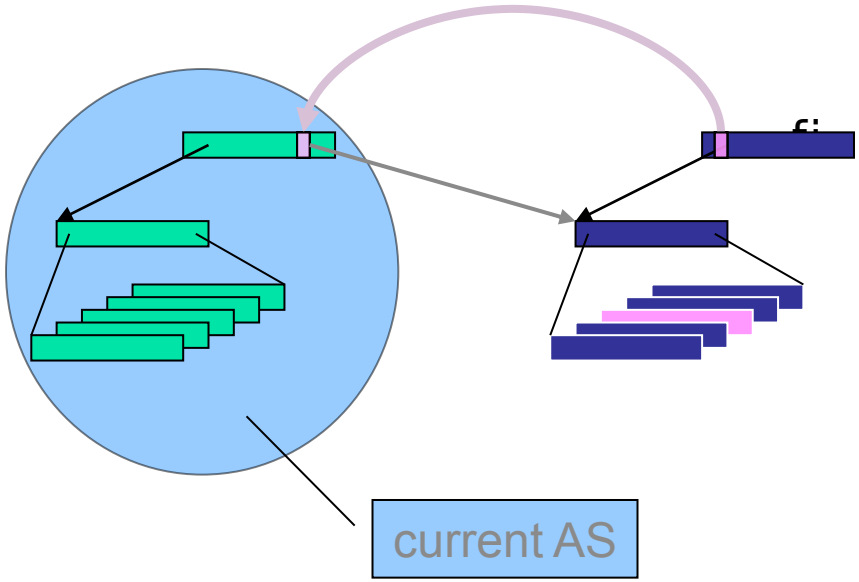
myPDE.TMarea := nil;

if (dest is small or
dest HwAS = curr HwAS)
then

flush TLB

fi

Assuming small spaces use the temp. mapping area.



Temporary Mapping Revisited

Leave thread:

if mytcb.partner ≠ nilthre

myPDE.TMarea := nil ;

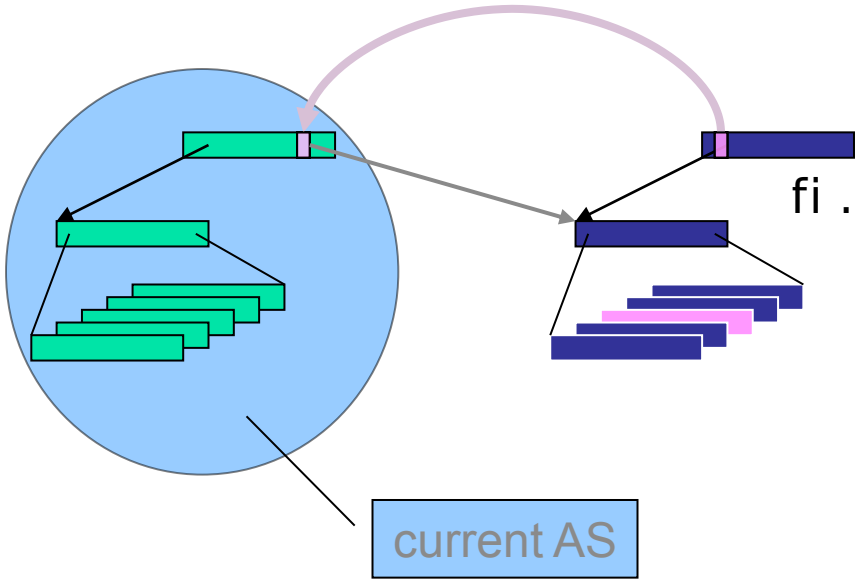
if (dest is small or
dest HwAS = curr HwAS) and
not curr is small
then

flush TLB

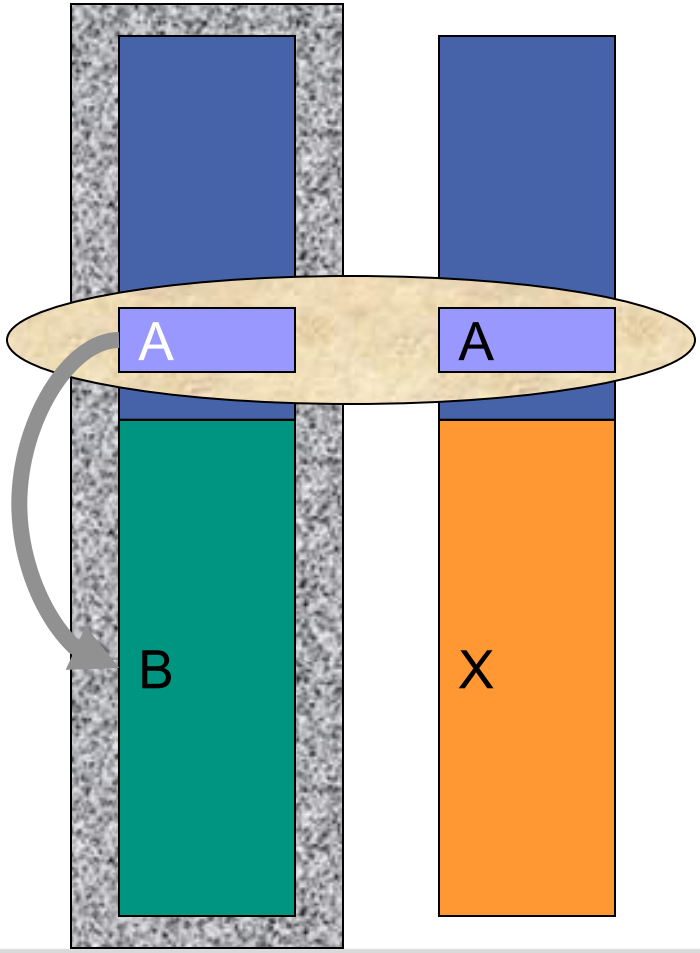
fi

fi .

Assuming small spaces never use the temp. mapping area.

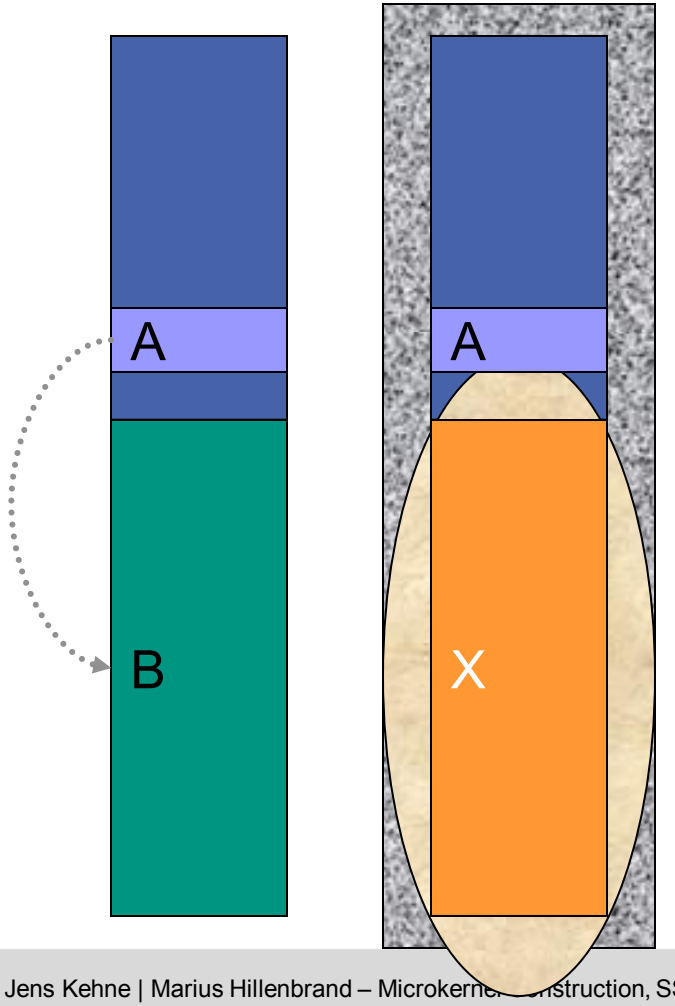


Thread Switching Revisited



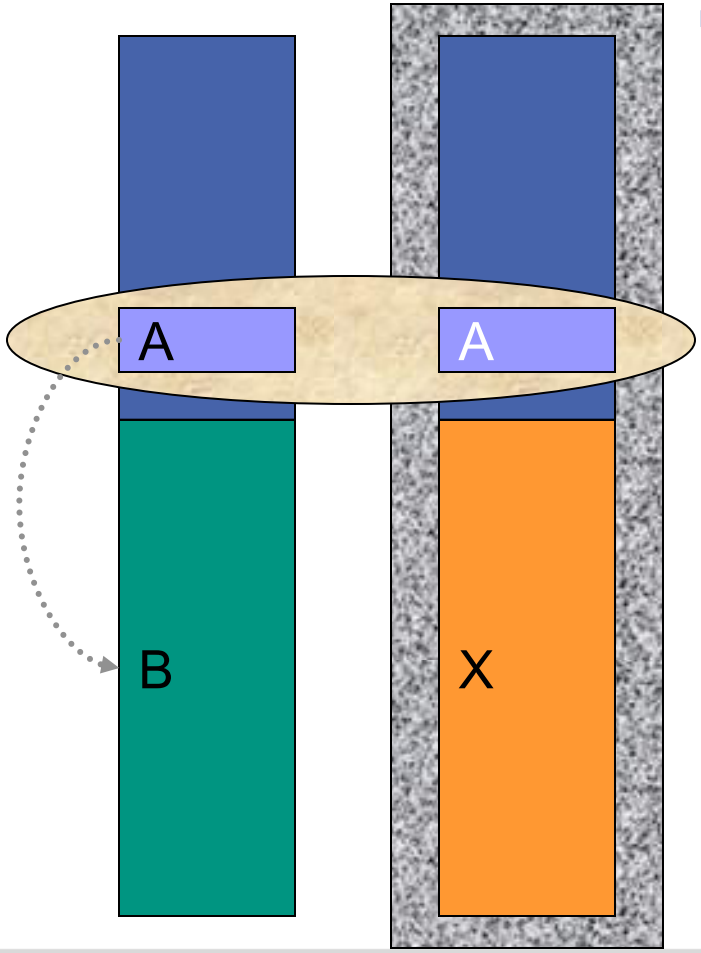
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**

Thread Switching Revisited



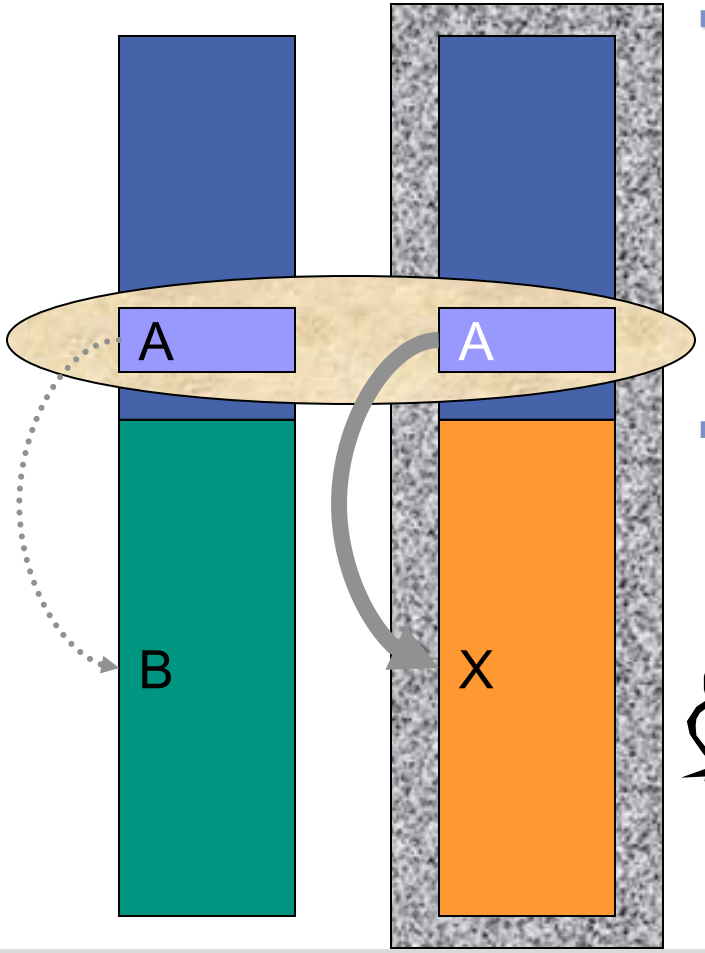
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**

Thread Switching Revisited



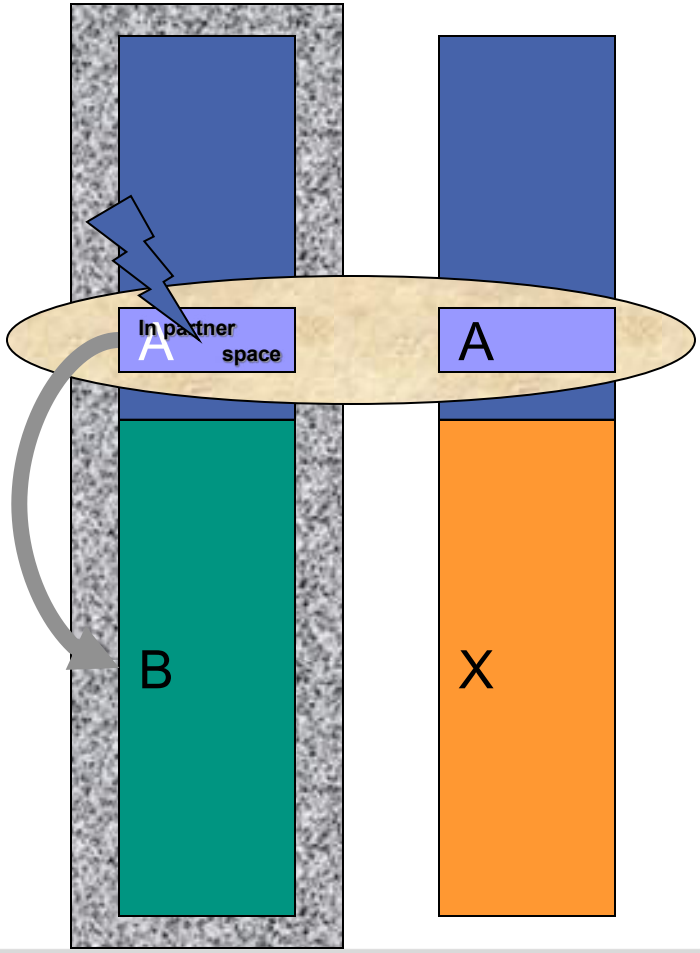
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**

Thread Switching Revisited



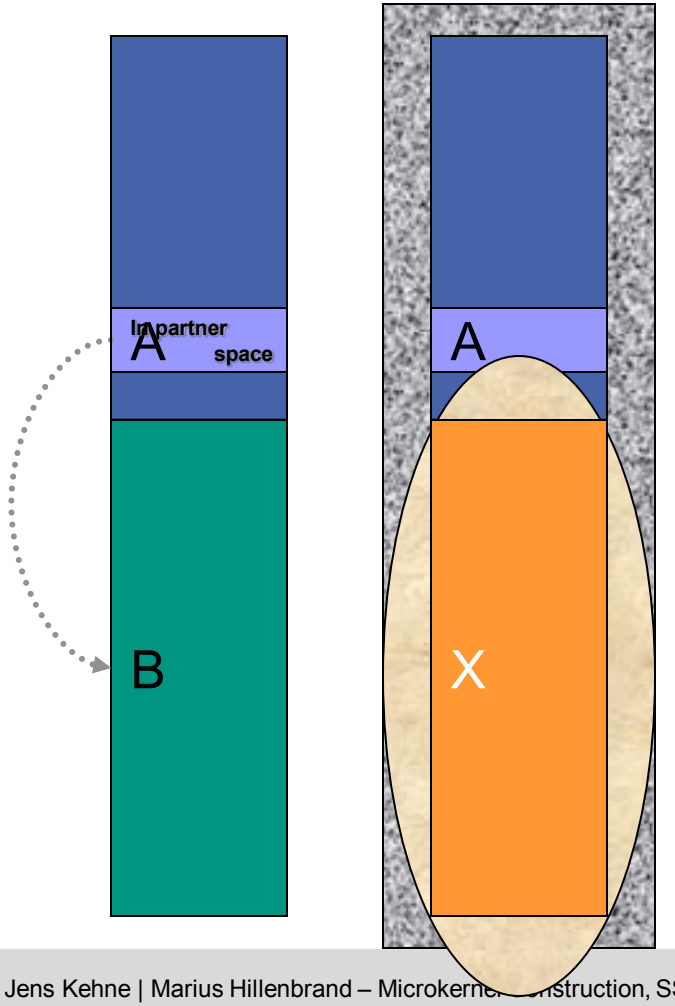
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**
- Preemption or PF resumes in **A**, but **A** now executes in HwAS **X**

Thread Switching Revisited
A Solution



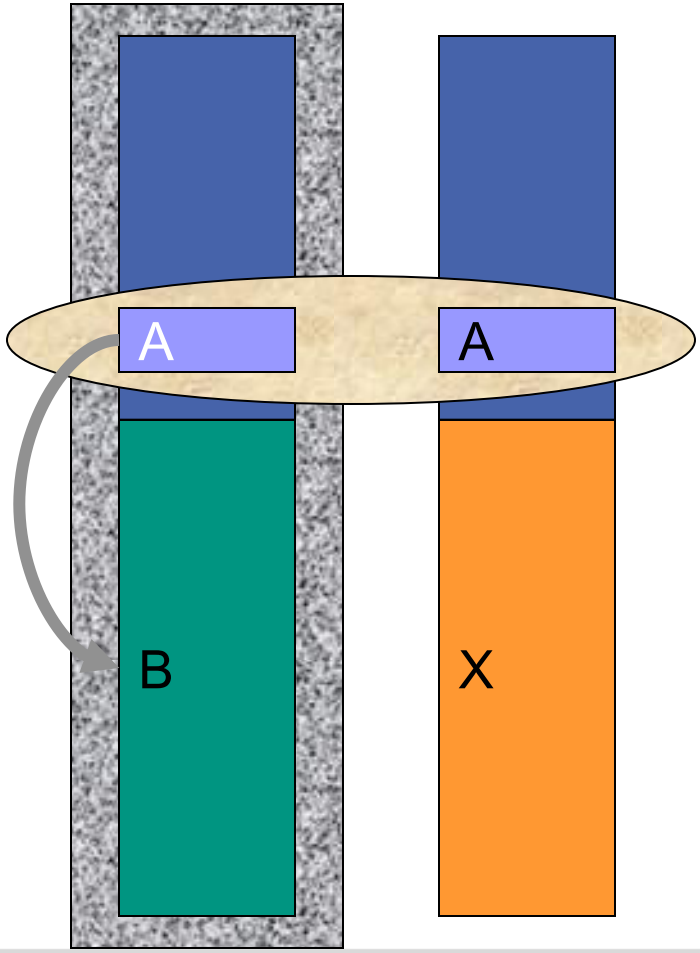
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
    - Mark **A “in partner space”**

Thread Switching Revisited
A Solution



- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
    - Mark **A "in partner space"**
  - Thread switch from **A** to **X**

Thread Switching Revisited
A Solution



- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
    - Mark **A "in partner space"**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**
    - Also switch HwAS to HwAS **B**

SMALL ADDRESS SPACES AND FAST SYSTEM CALLS

Getting around automatic
segment register reloading

Fast System Calls

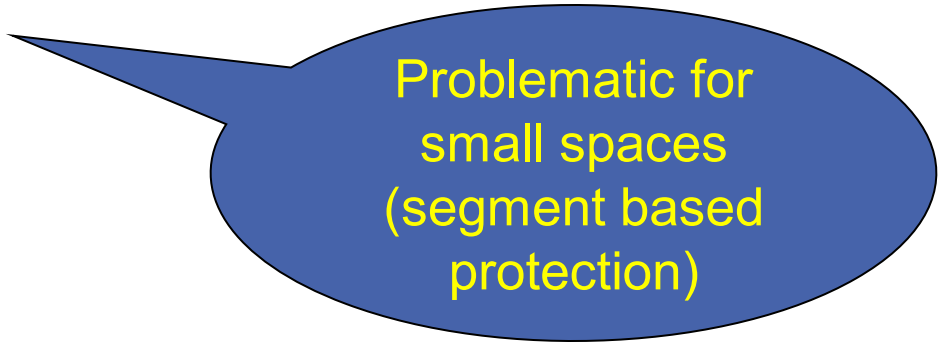
- Optimized instructions
  - sysenter/sysexit (Intel)
  - syscall/sysret (AMD)
- Faster than software interrupts
  - Can avoid certain checks
  - Unconditionally reload segment registers (page based protection)

	450 MHz PIII
Int / Iret	280
Sysenter / Sysexit	50

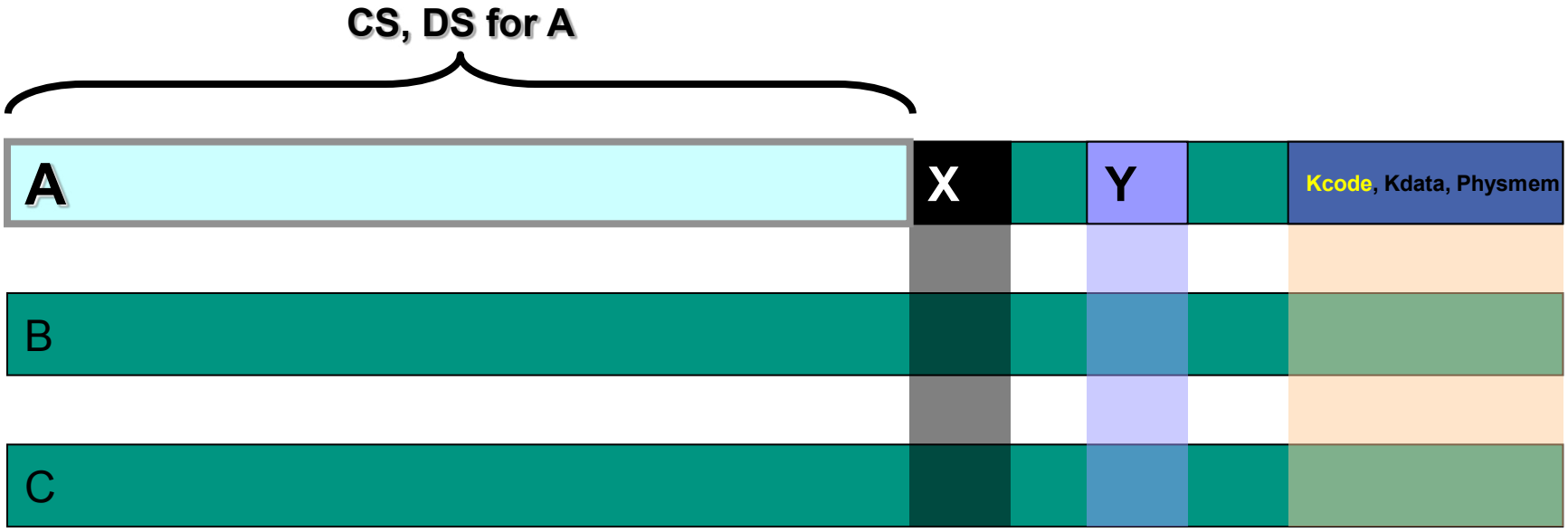
Fast System Calls

- Optimized instructions
  - sysenter/sysexit (Intel)
  - syscall/sysret (AMD)
- Faster than software interrupts
  - Can avoid certain checks
  - Unconditionally reload segment registers (page based protection)

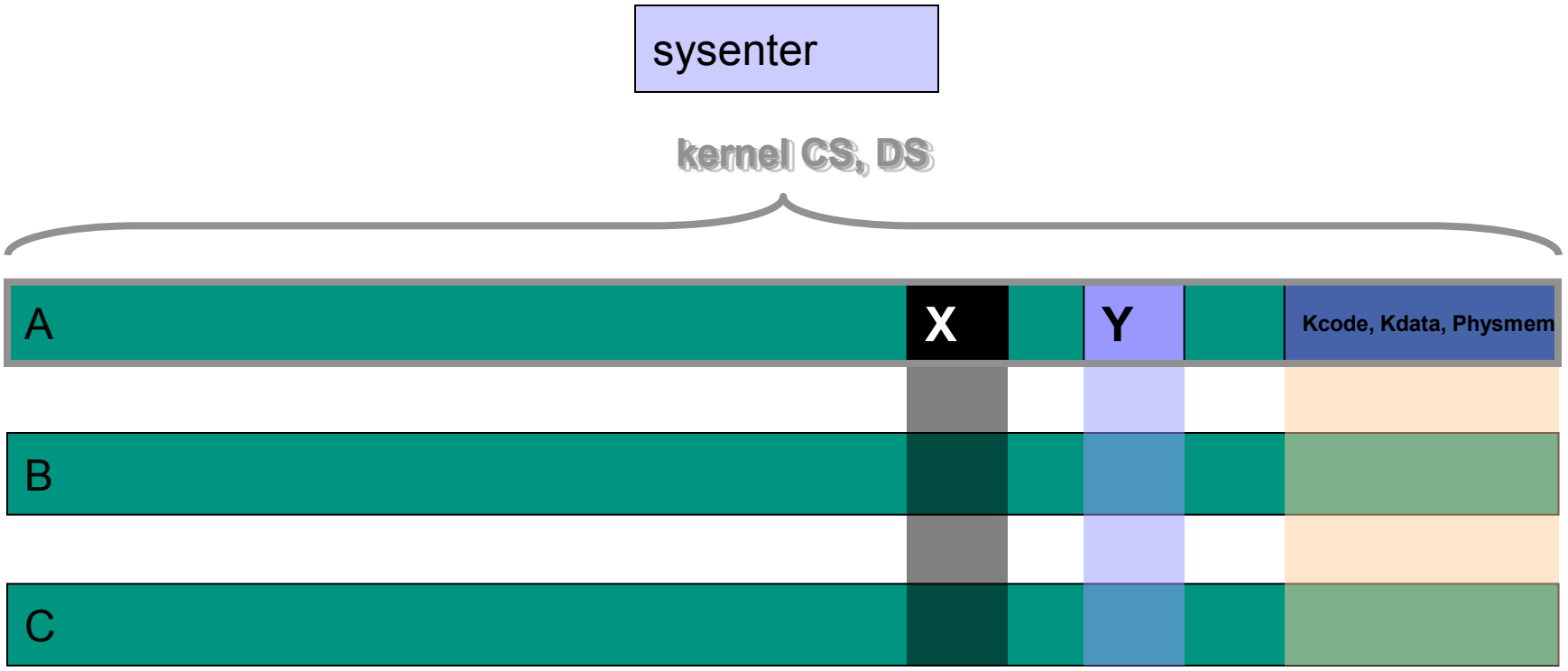
	450 MHz PIII	1.5 GHz P4
Int / Iret	280	1600
Sysenter / Sysexit	50	140



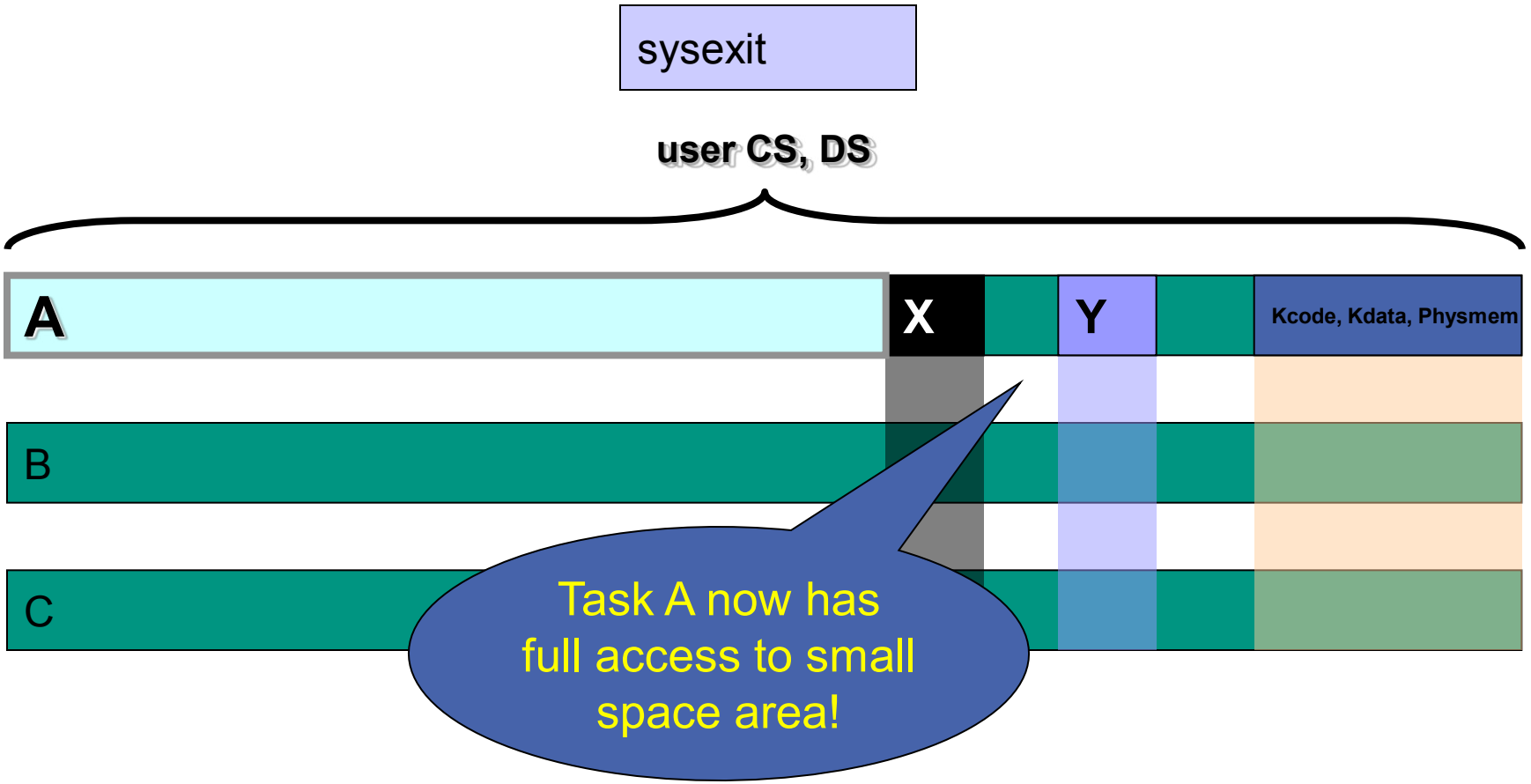
Fast System Calls
Automatic Segment Register Reloading



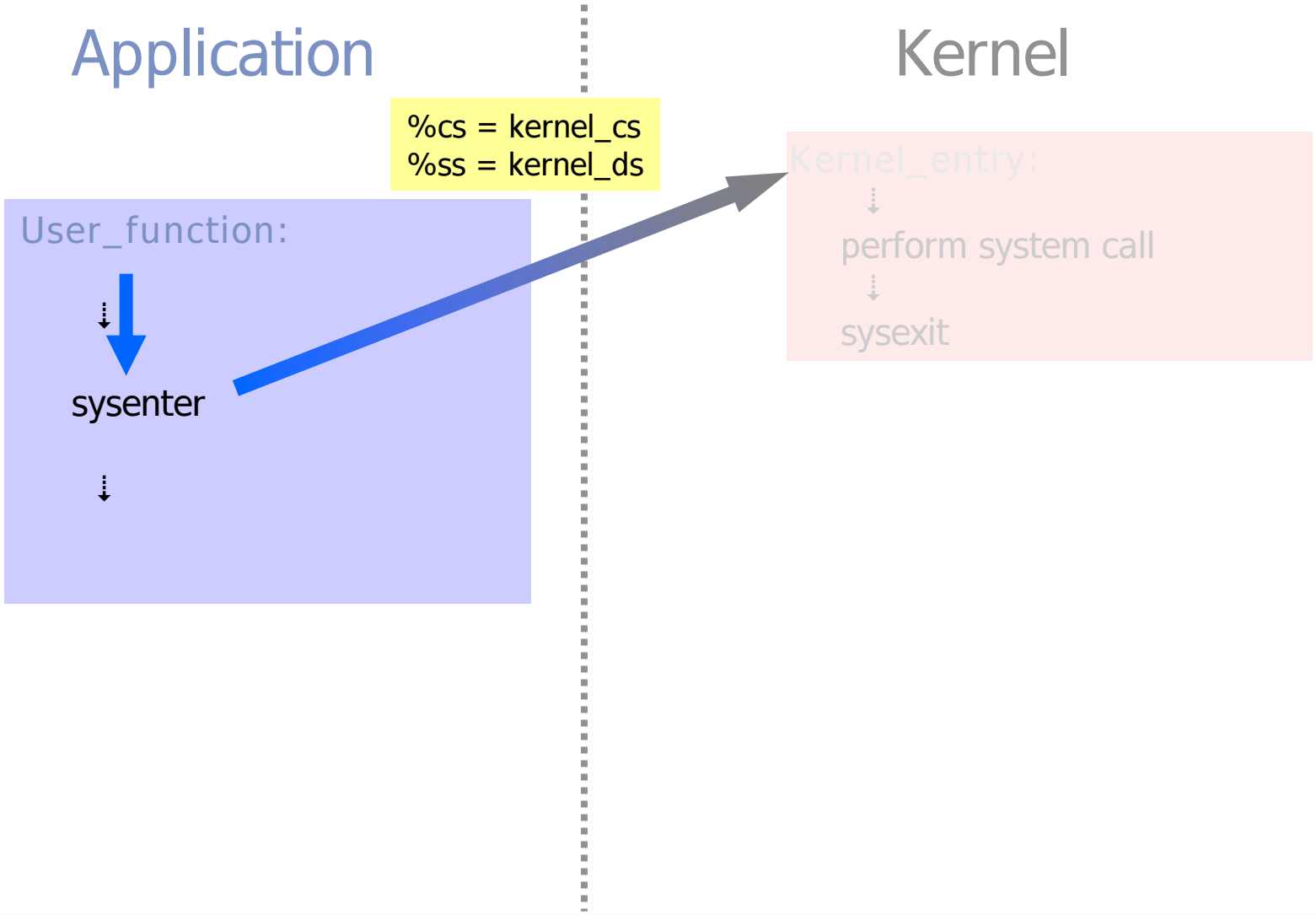
Fast System Calls
Automatic Segment Register Reloading



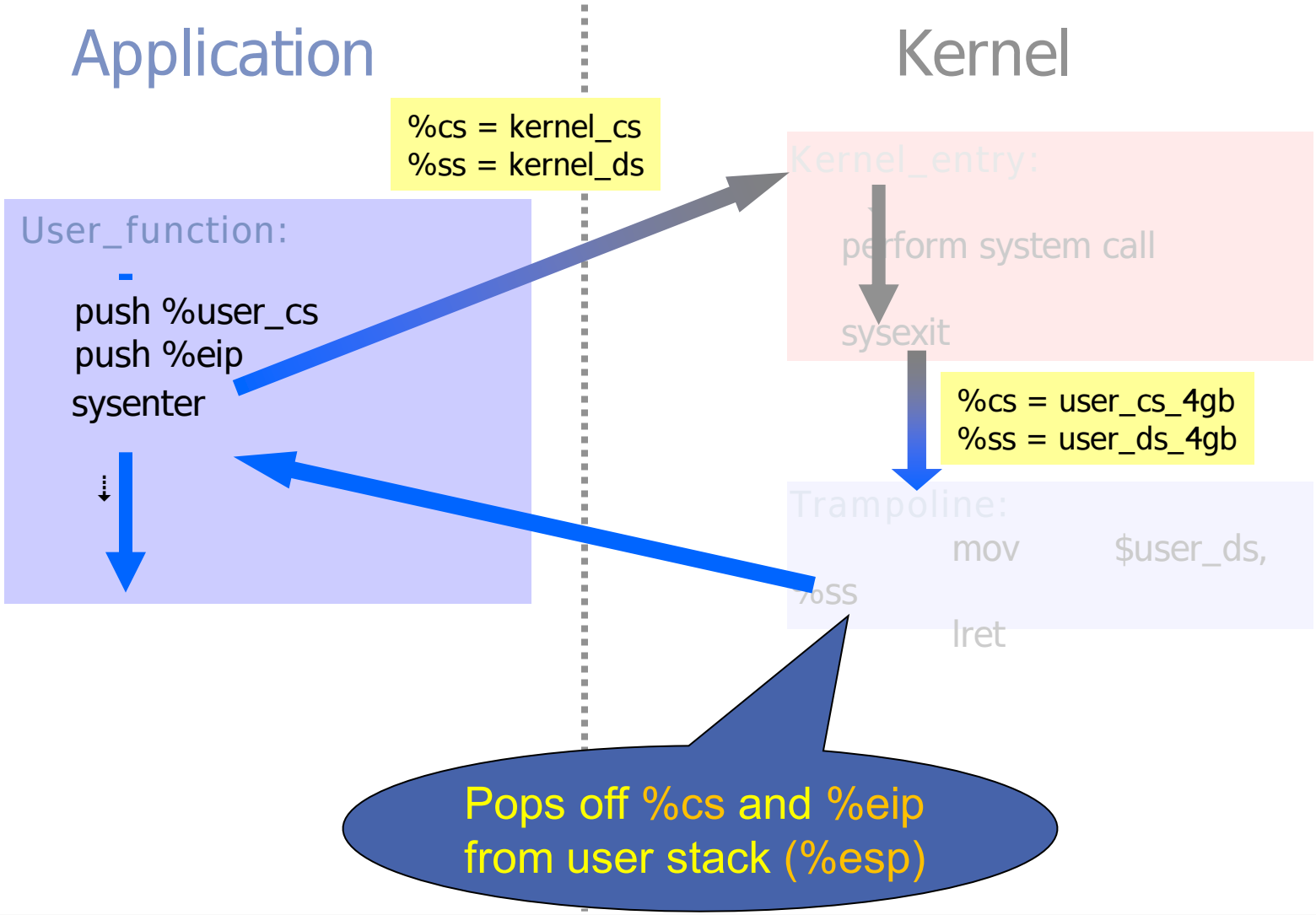
Fast System Calls
Automatic Segment Register Reloading



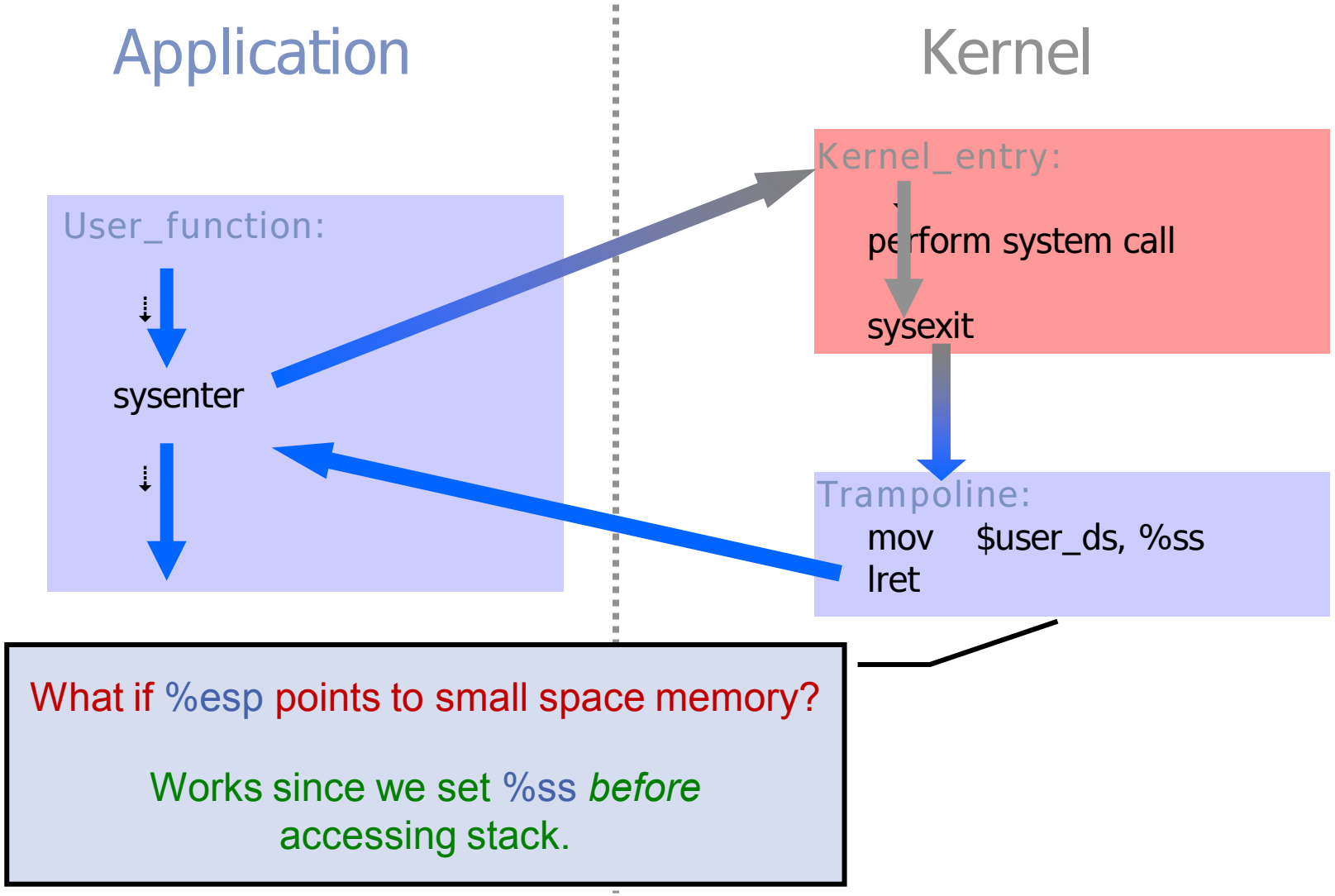
Automatic Segment Register Reloading Solution – In-Kernel Trampoline



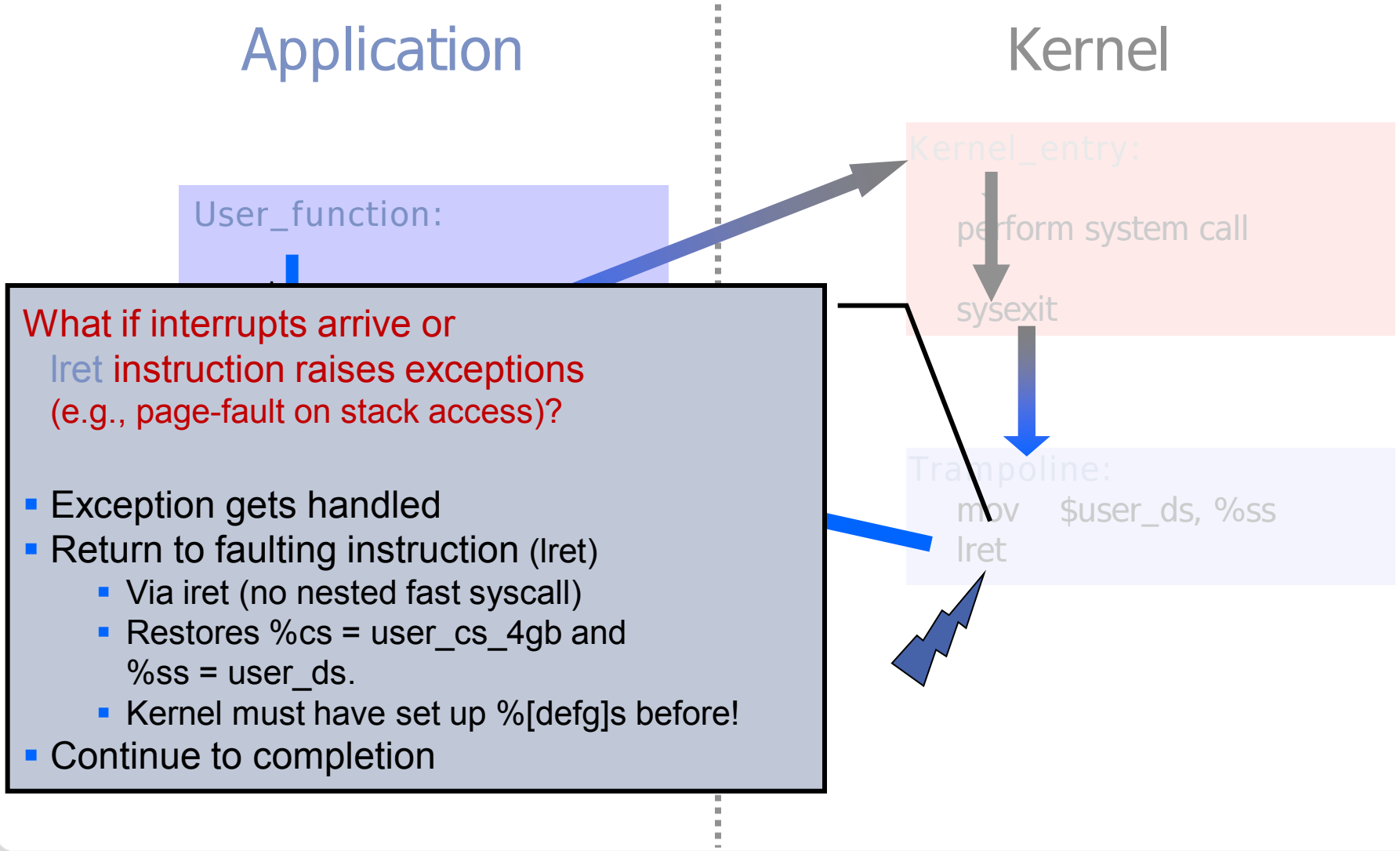
Automatic Segment Register Reloading Solution – In-Kernel Trampoline



Automatic Segment Register Reloading Solution – In-Kernel Trampoline



Automatic Segment Register Reloading Solution – In-Kernel Trampoline



What if interrupts arrive or
lret instruction raises exceptions
(e.g., page-fault on stack access)?

- Exception gets handled
- Return to faulting instruction (lret)
  - Via iret (no nested fast syscall)
  - Restores %cs = user_cs_4gb and %ss = user_ds.
  - Kernel must have set up %[defg]s before!
- Continue to completion