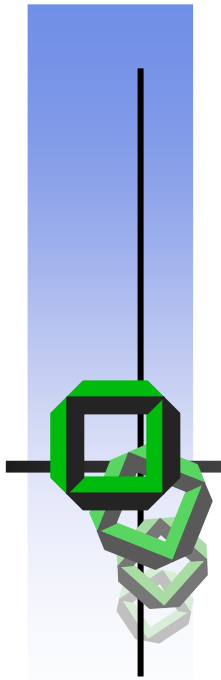
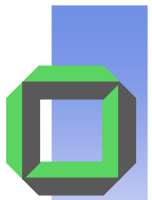


µ-Kernel Construction (8)

Small Address Spaces – IPC

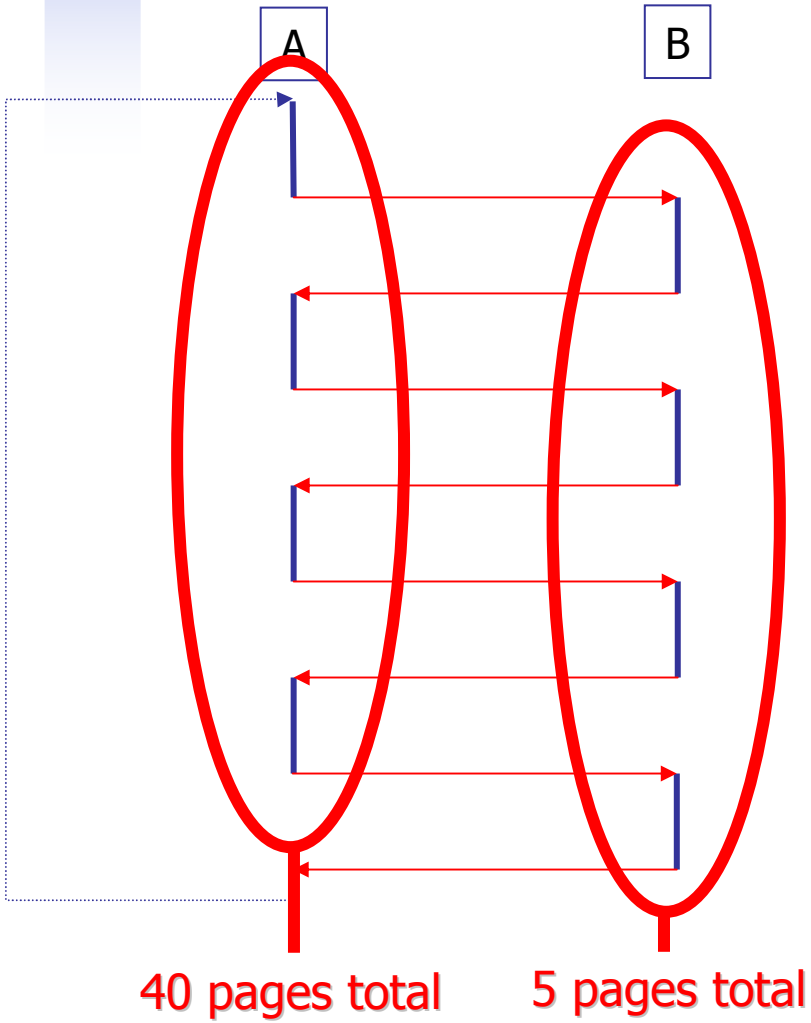
Special Optimization for Untagged TLBs



Untagged TLB Context-Switch Costs

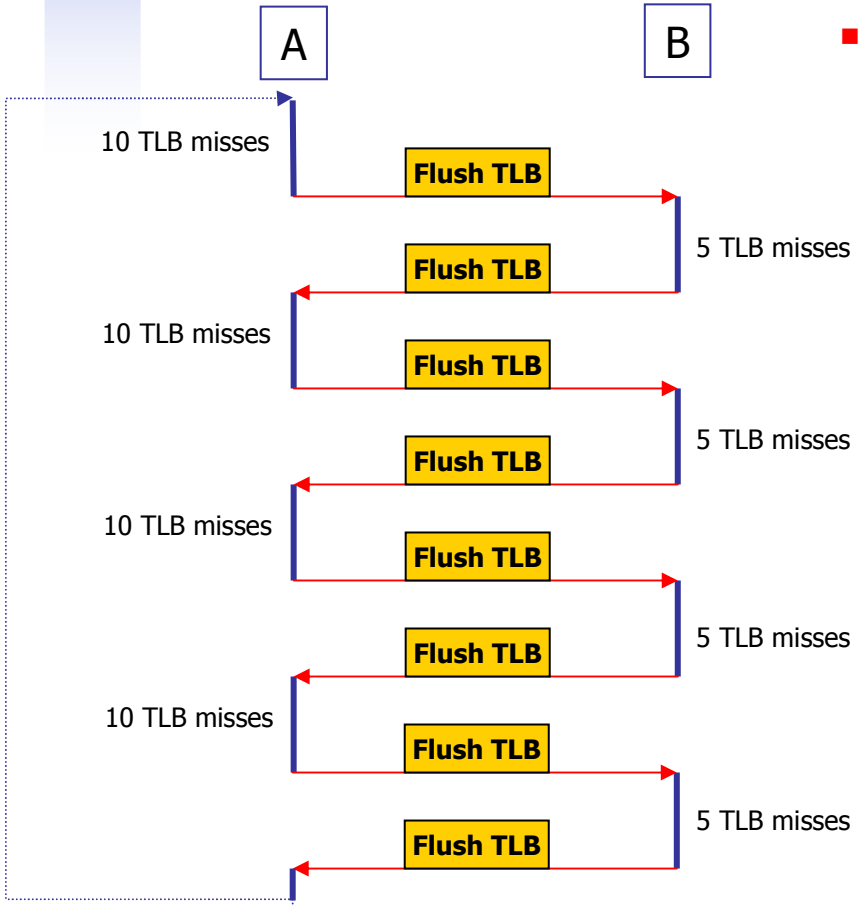
	486 … PIII	Pentium 4
Enter/exit kernel	40 … 200 cycles	150 … 200 cycles
Switch thread	≈ 10 cycles	≈ 10 cycles
Switch address space		
Flush TLB	≈ 50 … 80 cycles	≈ 230 … 250 cycles
Refill TLB	6 … 96 TLB refills 15 … 40 cycles/refill ≈ 100 … 4000 cycles	6 … 192 TLB refills 15 … 500 cycles/refill ≈ 100 … 96000 cycles
Refill L1 caches		12 K Tracecache 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

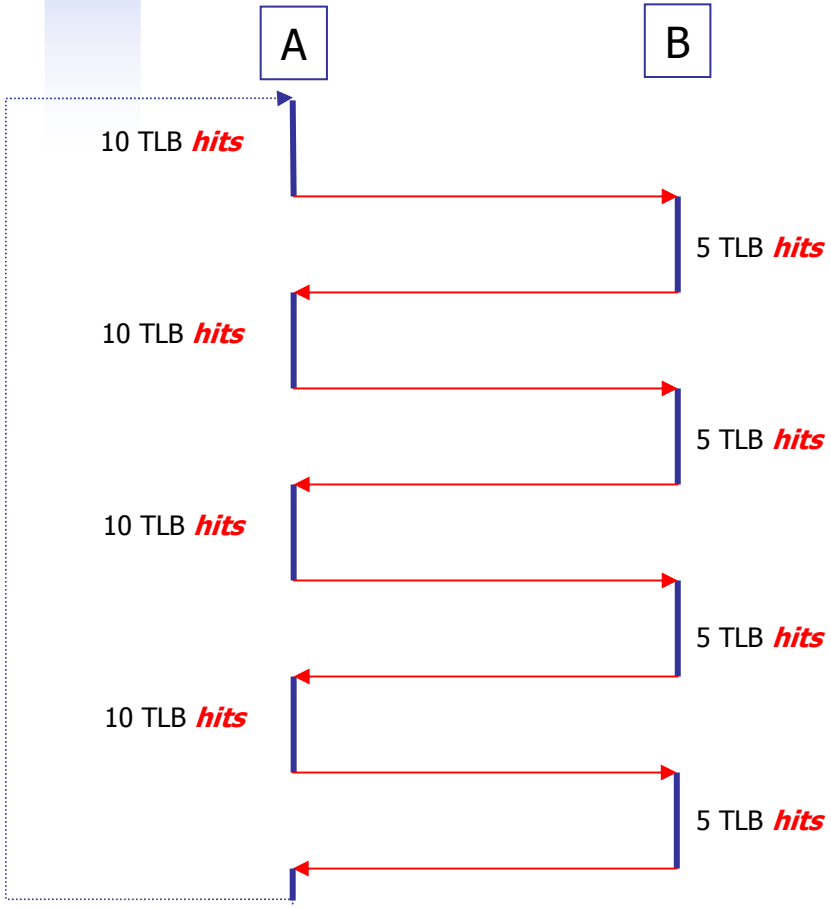


- Even when calling a thread with a very **small TLB working set**
  - Thread A frequently calls thread B
  - Working sets
    - Thread A: 4 different sets of 10 pages between B-calls
    - Thread B: always the same 5 pages

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

Untagged TLB total: 2740 … 33440

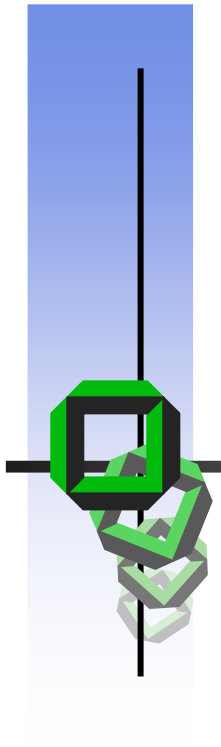
Untagged TLB Context-Switch Costs



- 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**

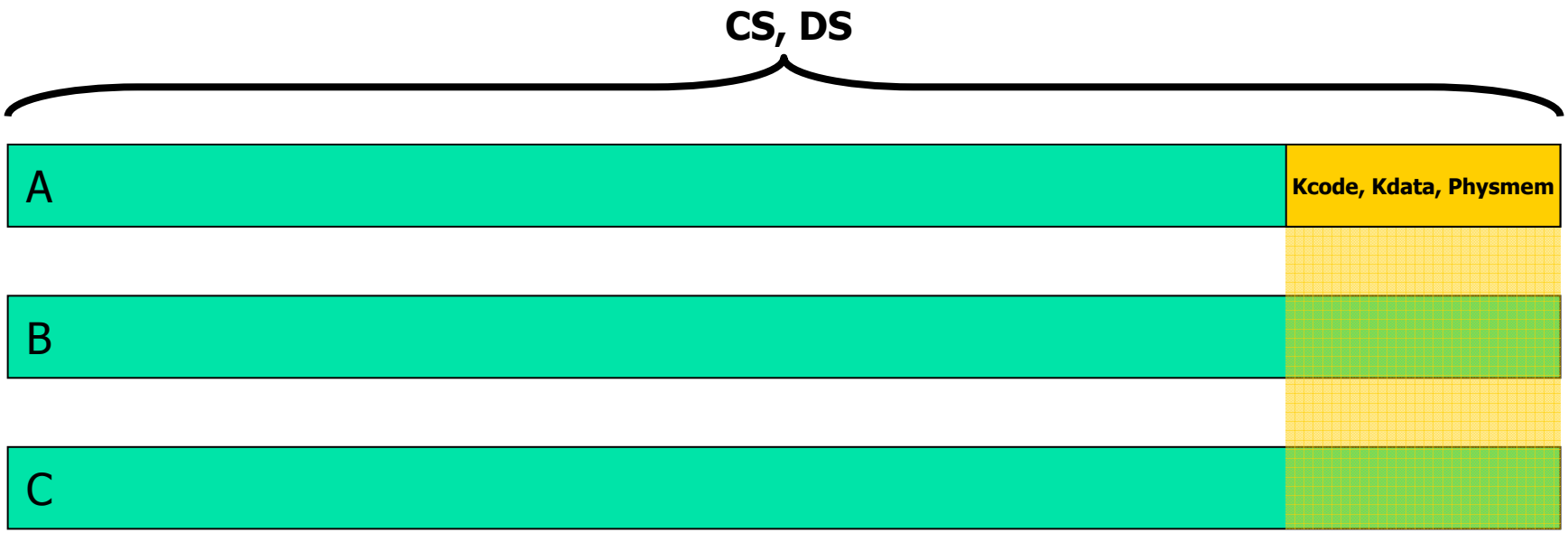
We get "TLB hits" starting at the second iteration, assuming a sufficiently large TLB.



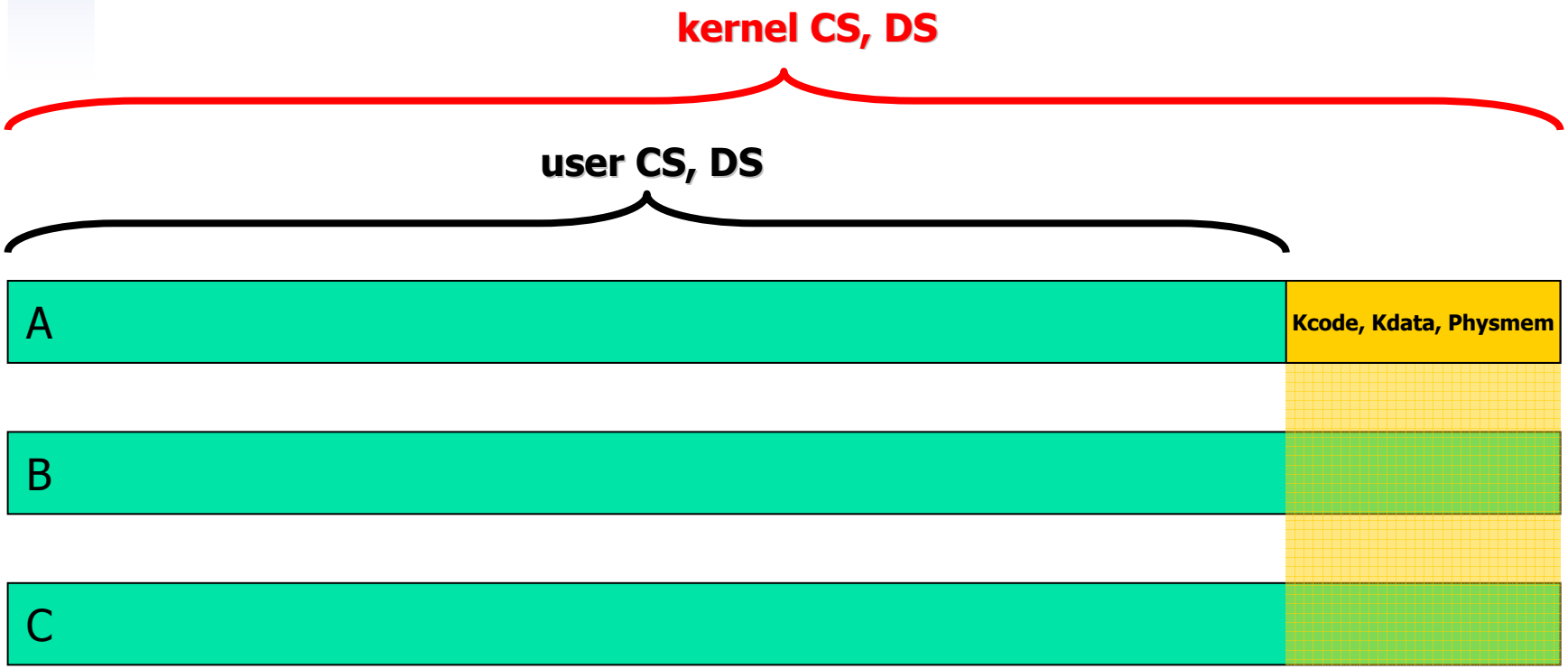
Small Address Spaces on x86

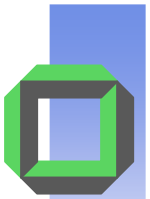
How to emulate tagged TLBs?

Address Spaces

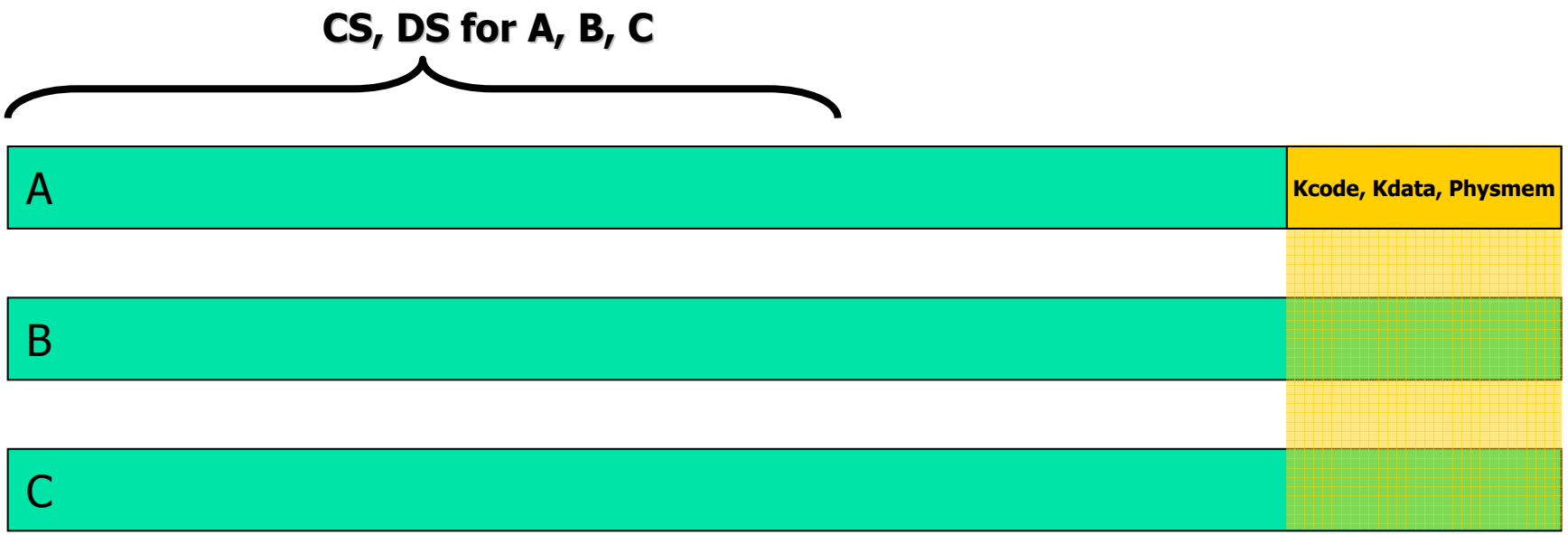


Address Spaces

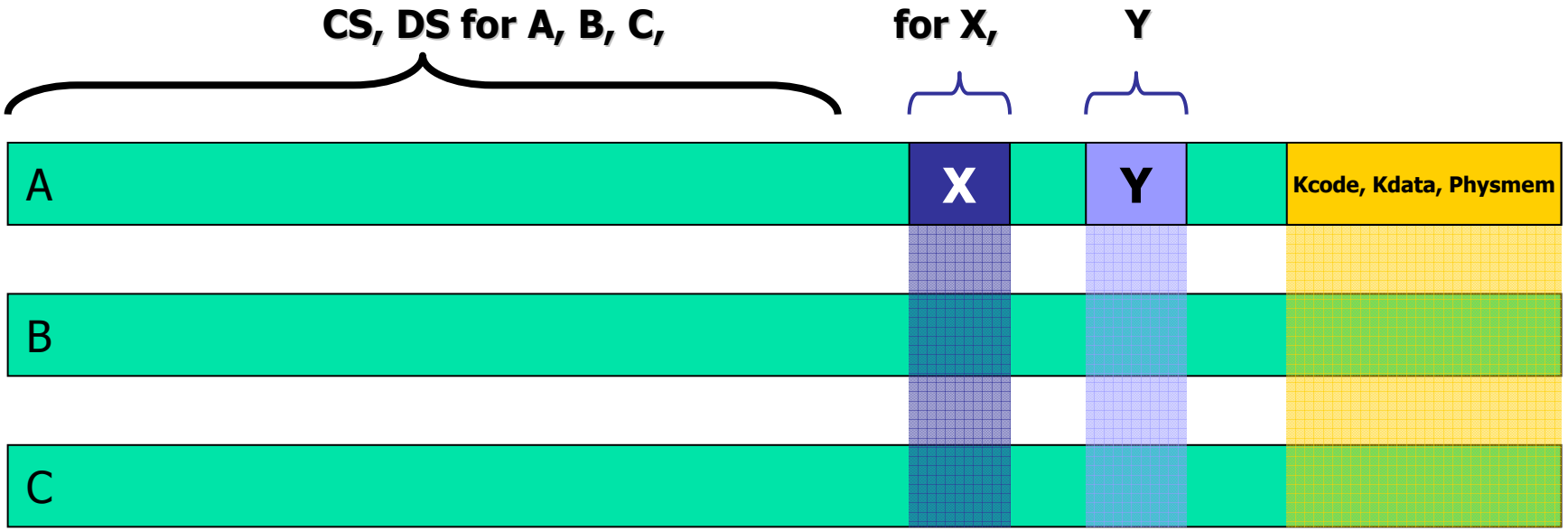




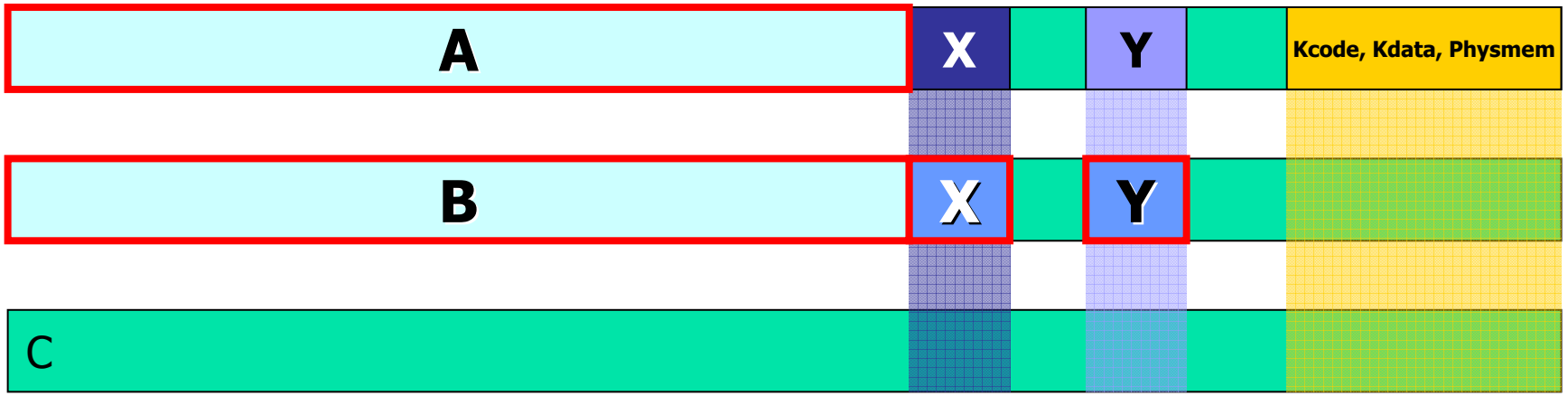
Small Address Spaces



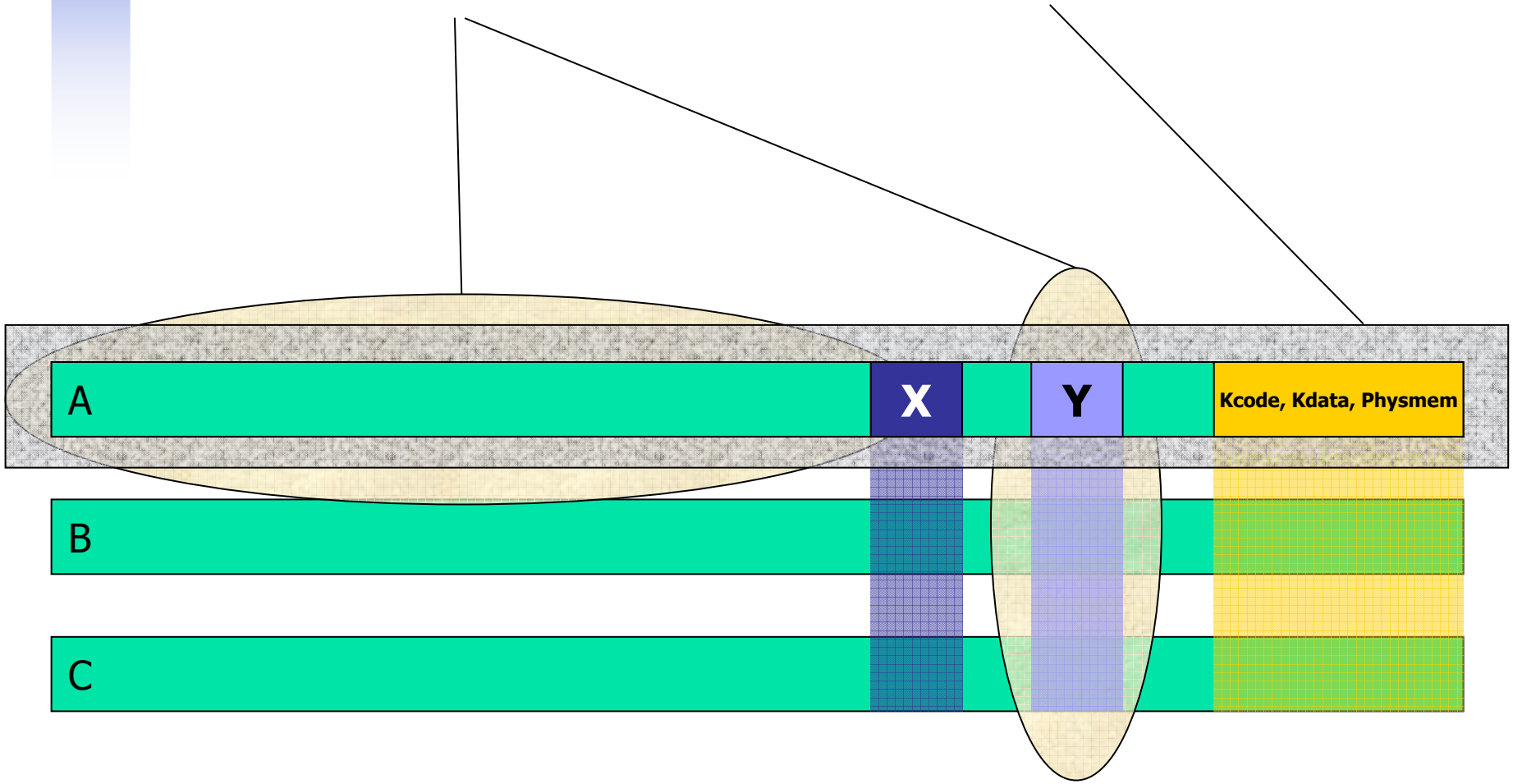
Small Address Spaces



Small Address Spaces

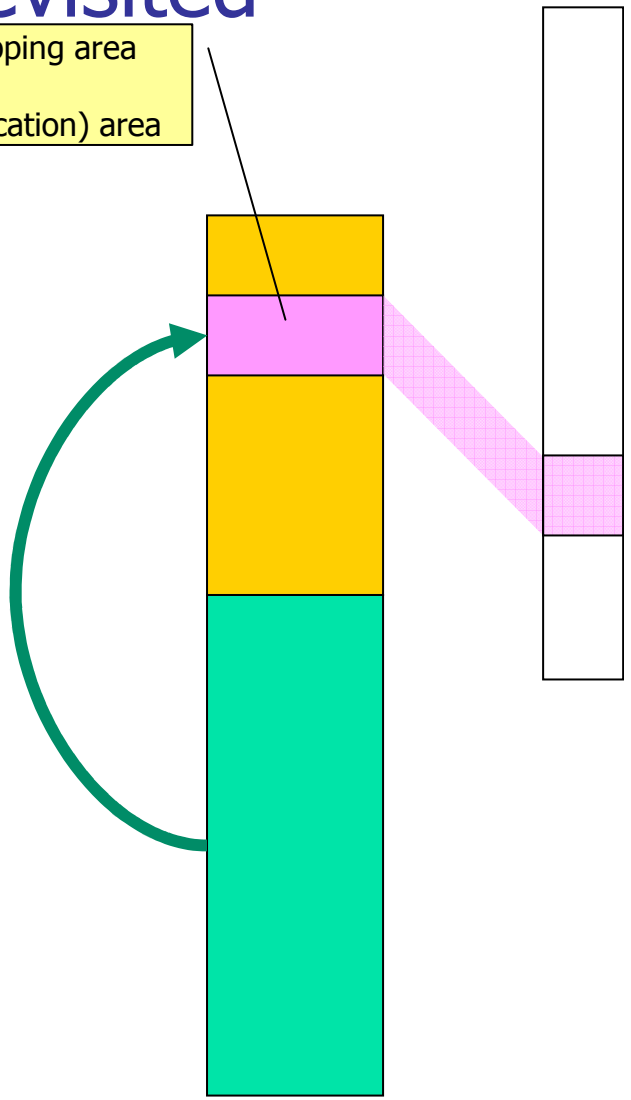


Address Space (AS) – Hardware AS (HwAS)



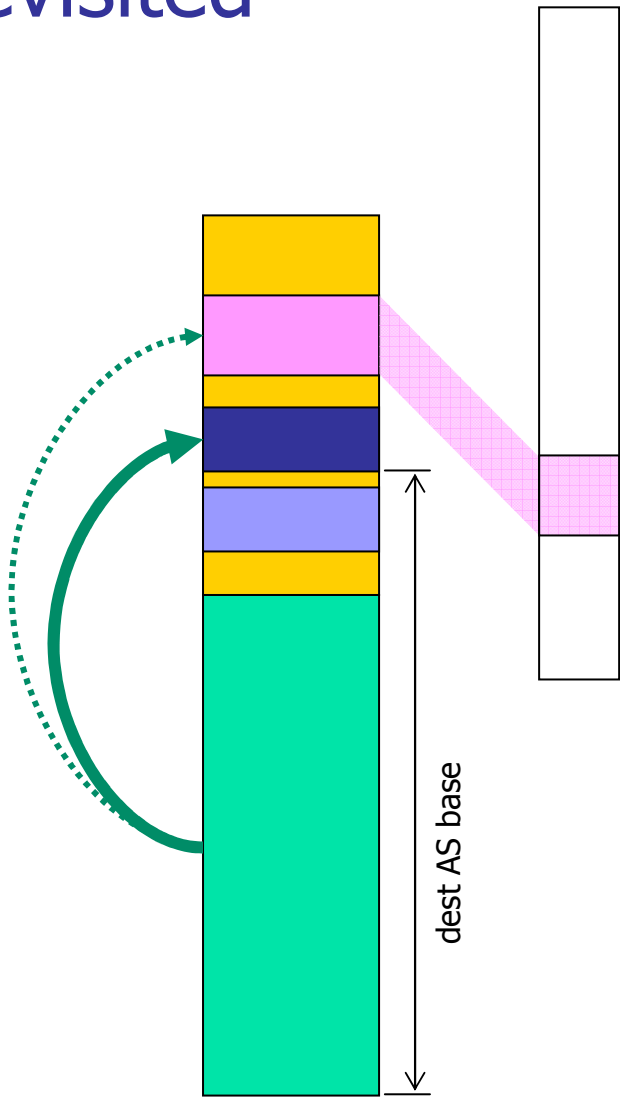
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest offset
Large / Large	temp mapping **switch HwAS**	0	Kernel com area



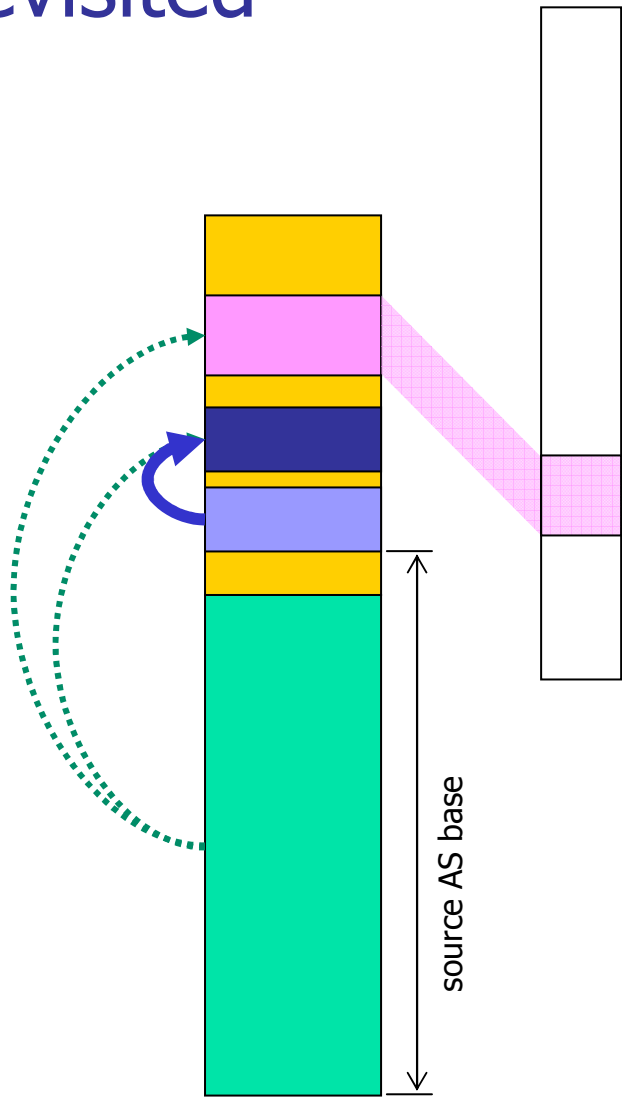
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest offset
Large / Large	temp mapping **switch HwAS**	0	Kernel com area
Large / Small	direct **- no HwAS switch -**	0	dest AS base



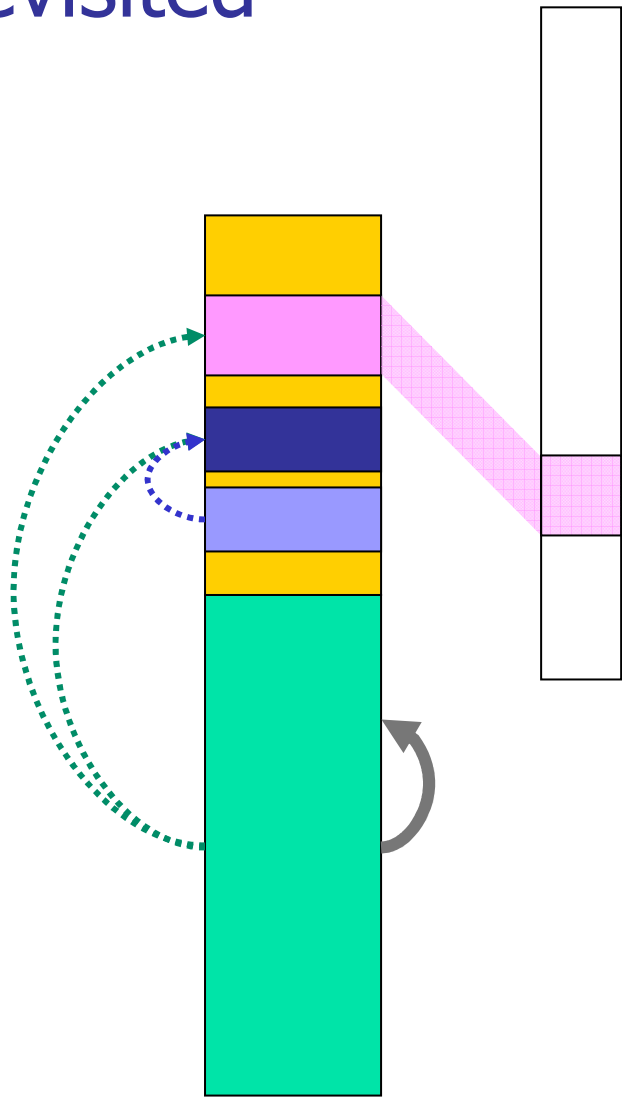
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest offset
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



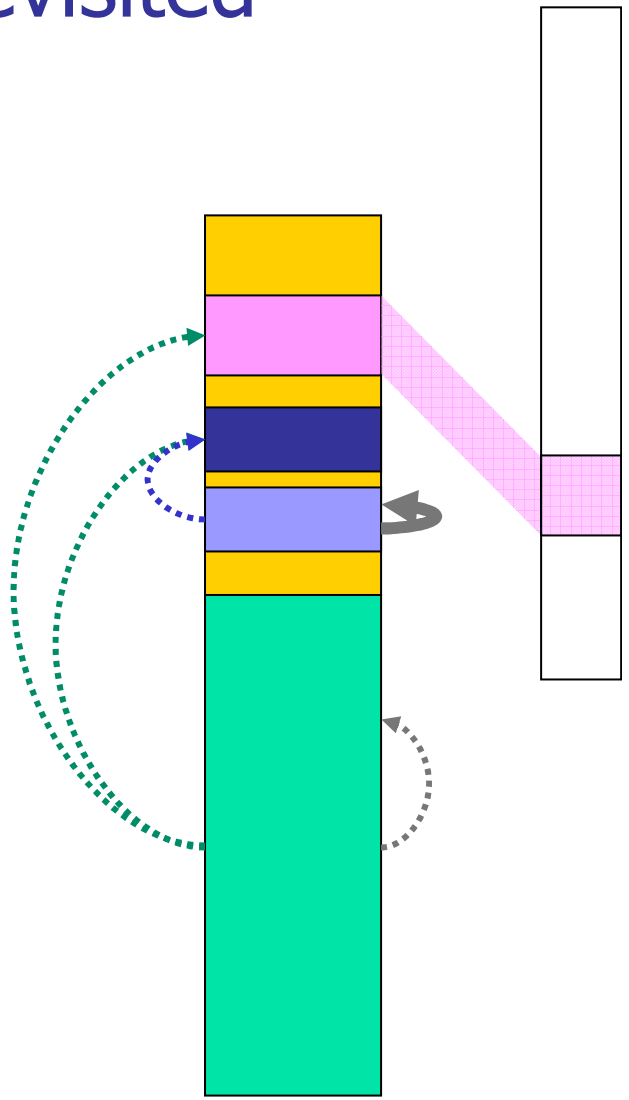
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest offset
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



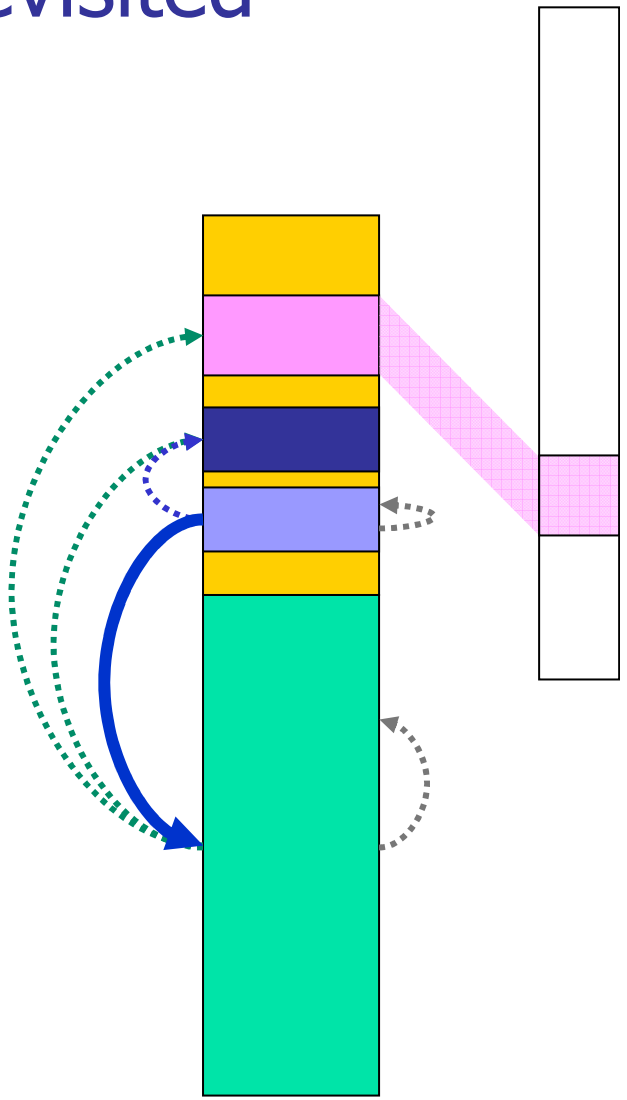
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest offset
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



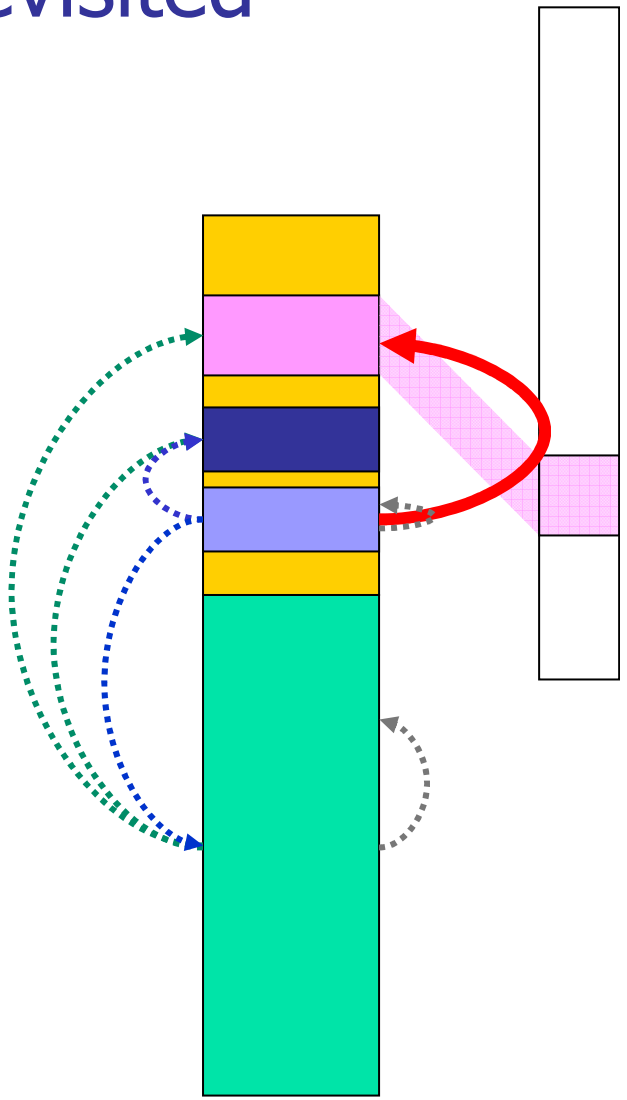
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest offset
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



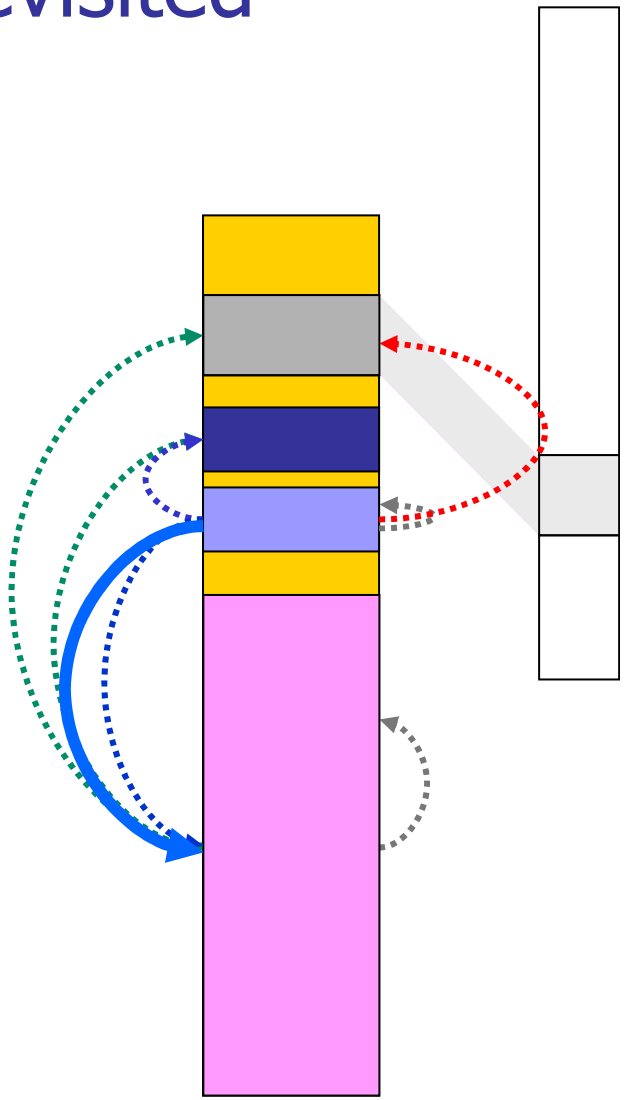
Long-IPC Implementation Revisited

Source / Destination	Method	Source offset	Dest offset
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

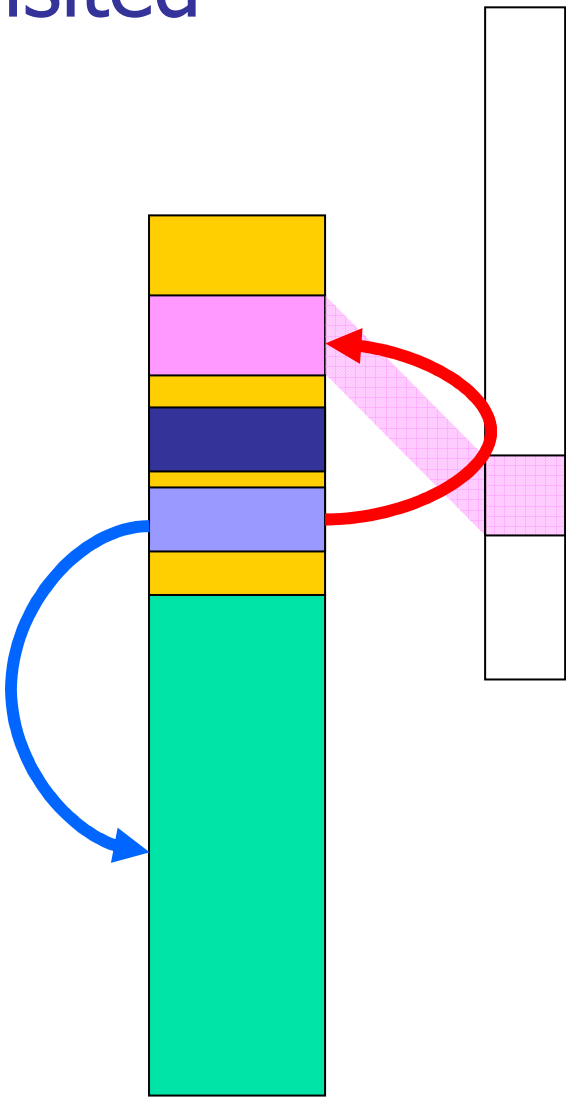
Source / Destination	Method	Source offset	Dest offset
Large / Large	temp mapping **switch HwAS**	0	Kernel com area
Large / Small	direct **- no HwAS switch -**	0	dest AS base
Small / Small	direct **- no HwAS switch -**	source AS base	dest AS base
Large / Same	direct **- no HwAS switch -**	0	0
Small / 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**

Small / Large ≠ Current HwAS	temp mapping **switch HwAS** **– or –** ***first*** **switch HwAS** ***then*** **direct**	source AS base	Kernel com area **– or –** 0

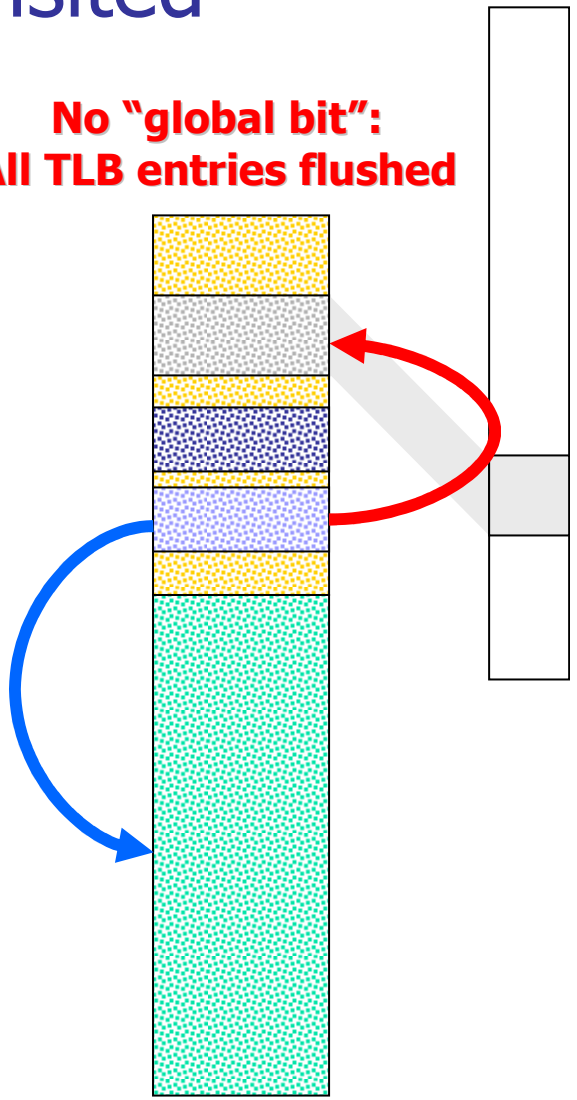


Long-IPC Implementation Revisited

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

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

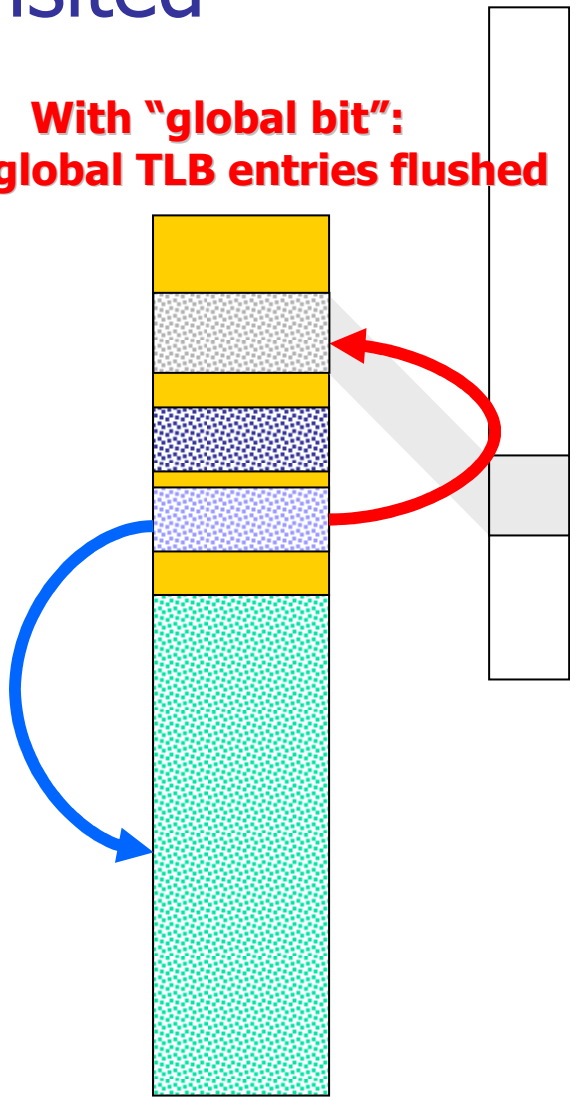


Long-IPC Implementation Revisited

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

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

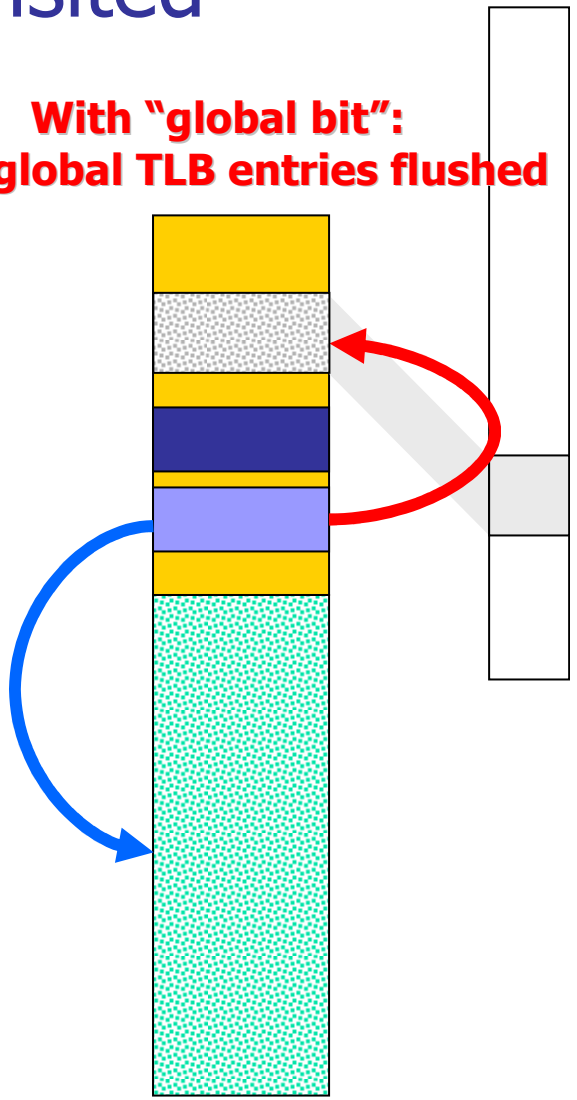


Long-IPC Implementation Revisited

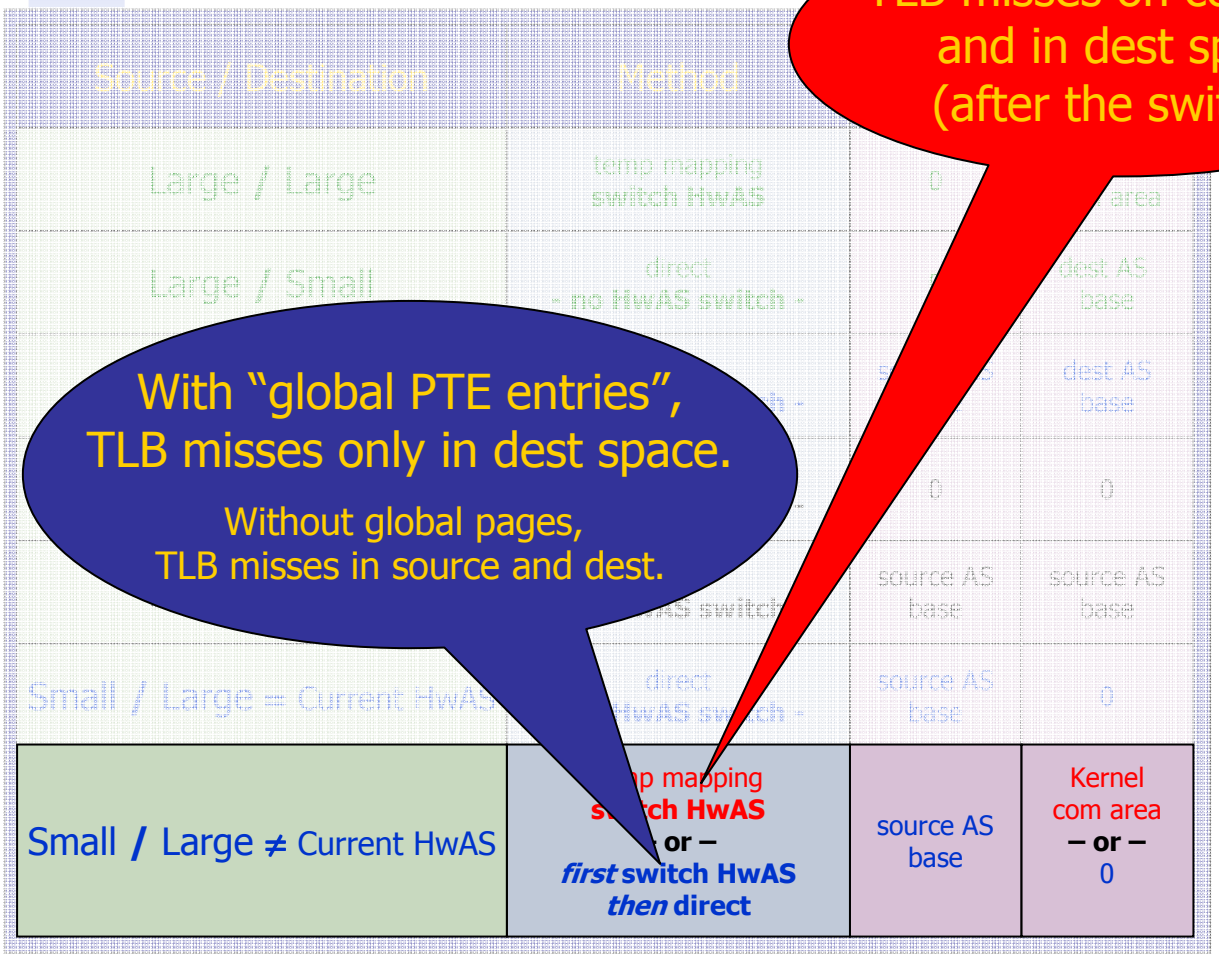
- **A global bit in page table entry indicates that TLB entry should not be flushed on TLB flushes**
  - **Used for not flushing kernel entries**
- **Small spaces are global over all hardware address spaces**
  - **Mark small spaces' PTEs as global**

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



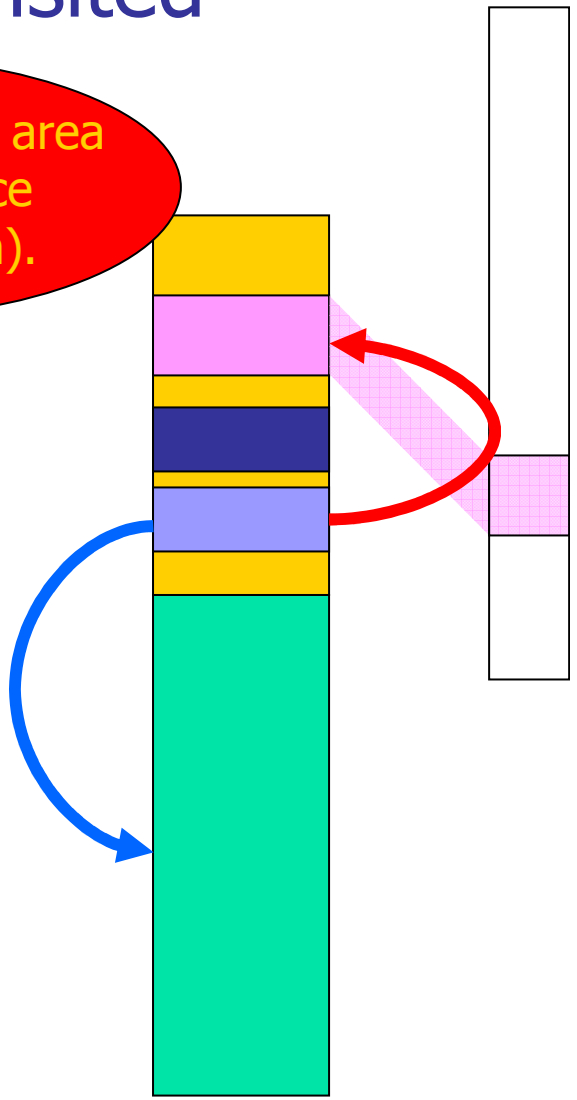
Long-IPC Implementation Revisited



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.



Temporary Mapping Revisited

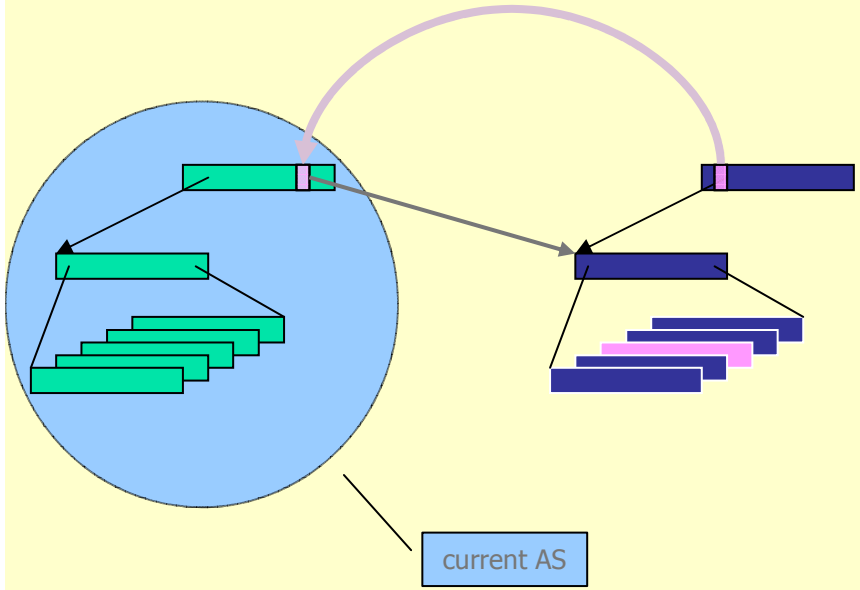
Leave thread:

if mytcb.partner ≠ nilthread **then**

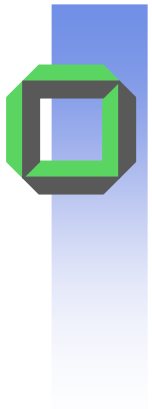
myPDE.TMarea := nil ;

if dest AS = my AS **then**

flush TLB



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

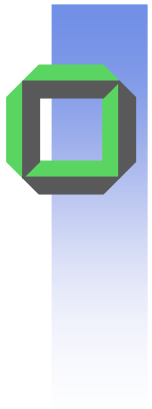


Evicting the Temporary Mapping Area

- 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			
Large AS			

[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]



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

Assuming small spaces use the temp. mapping area.

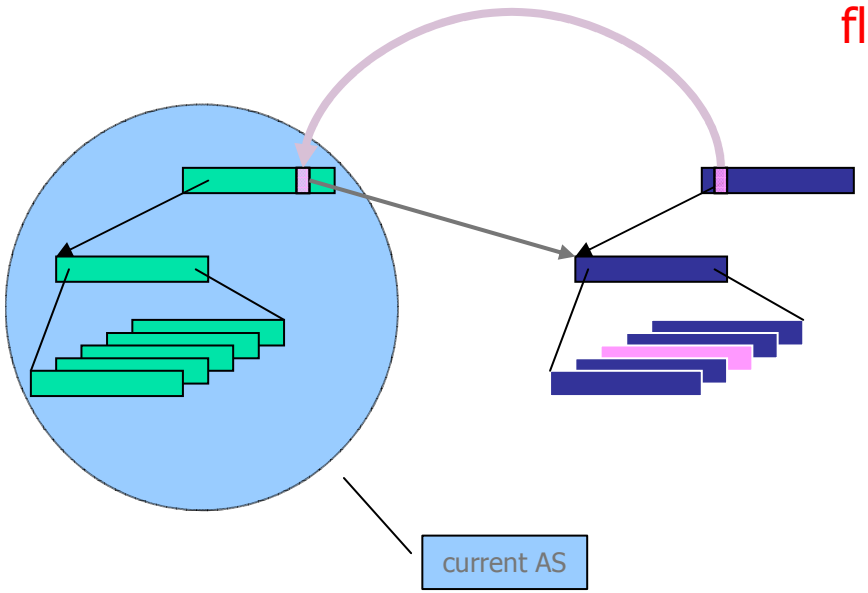
Leave thread:

if mytcb.partner ≠ nilthread **then**

myPDE.TMarea := nil ;

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

flush TLB



Temporary Mapping Revisited

Assuming small spaces **never** use the temp. mapping area.

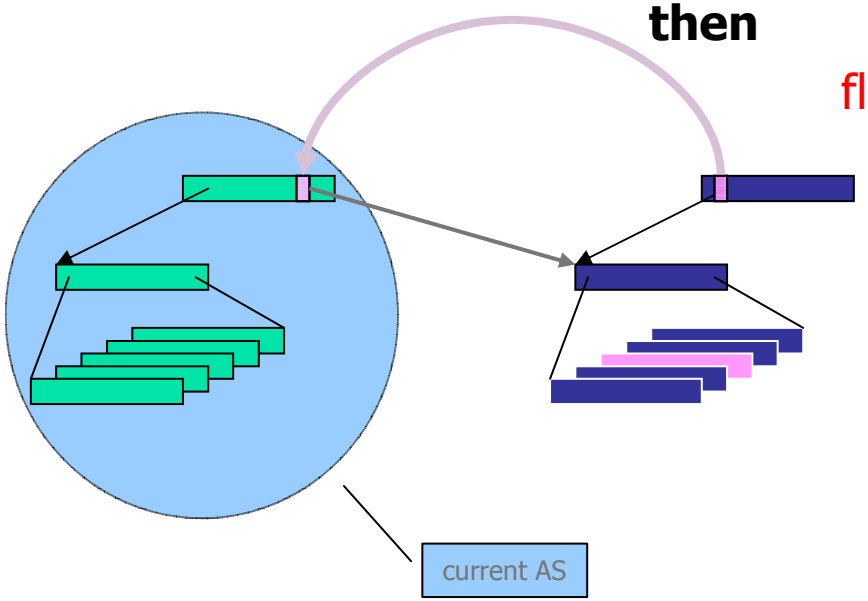
Leave thread:

if mytcb.partner ≠ nilthread **then**

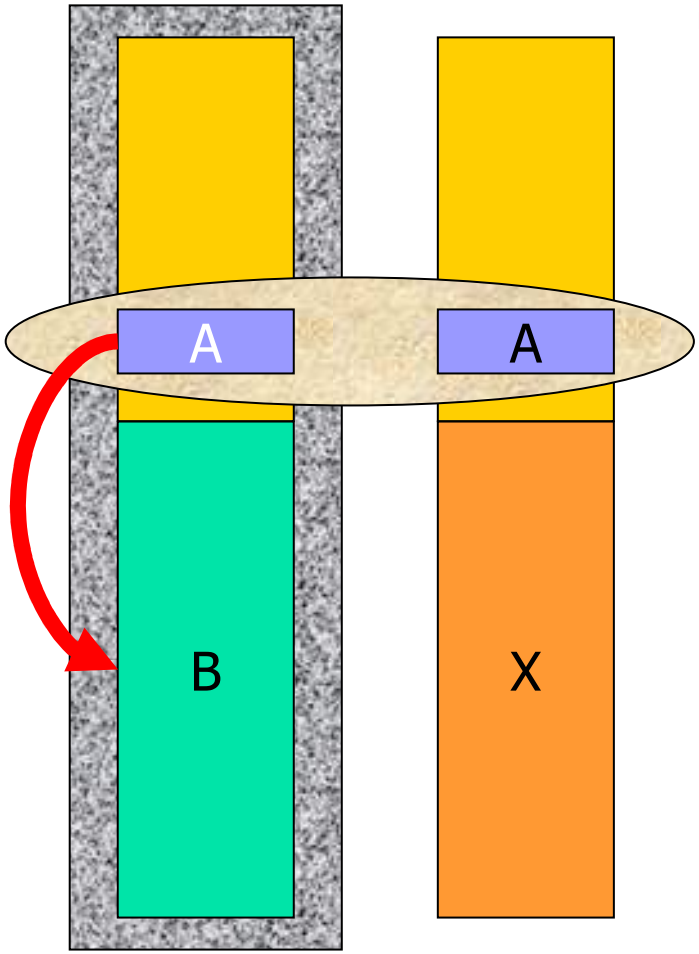
myPDE.TMarea := nil ;

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

flush TLB

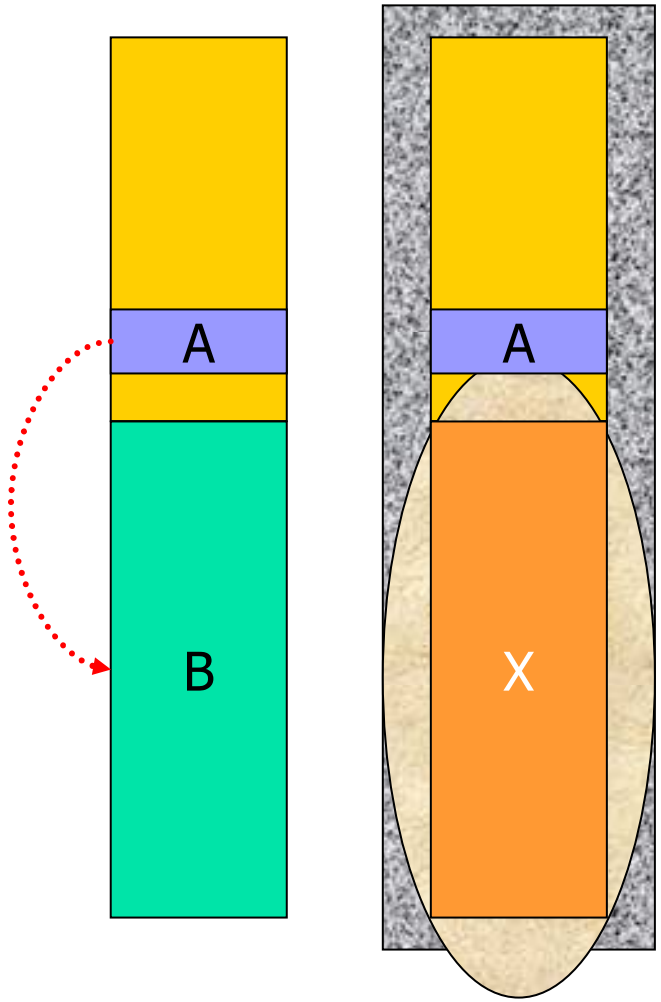


Thread Switching Revisited



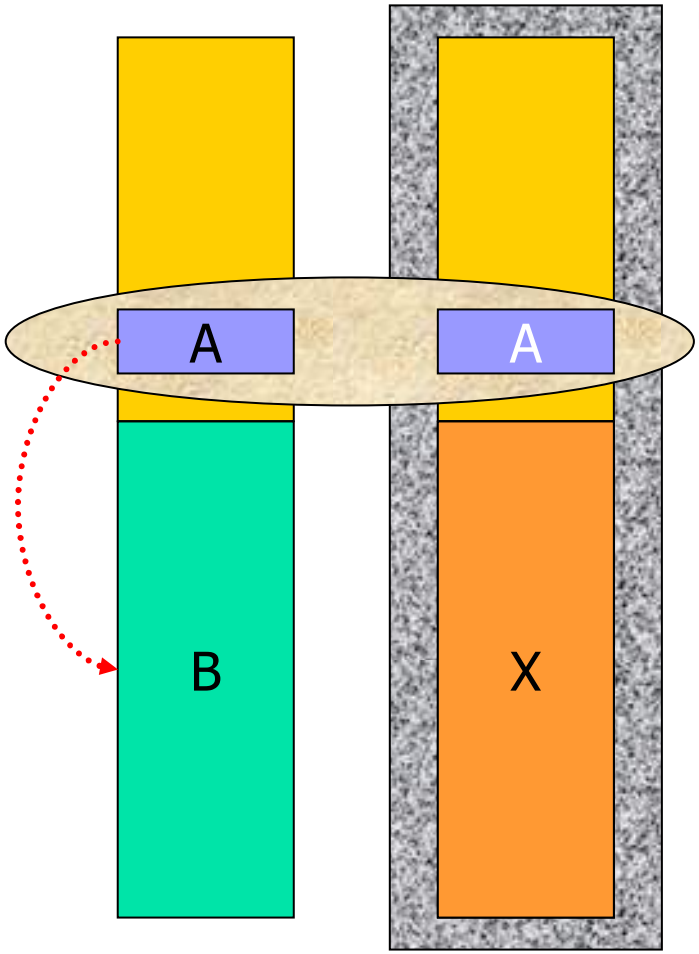
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**

Thread Switching Revisited



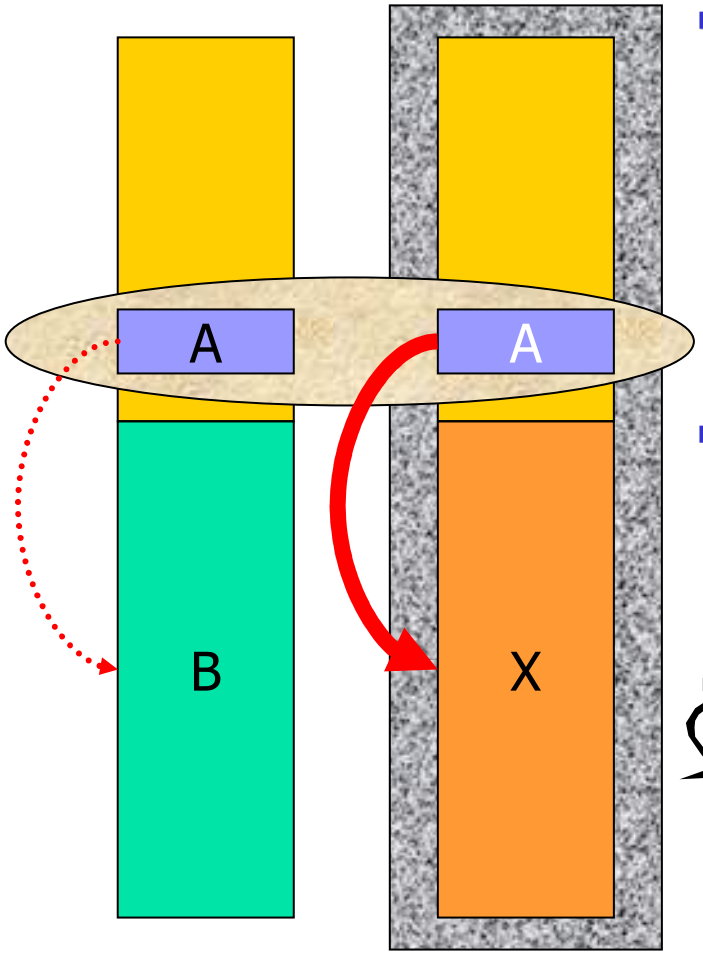
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**

Thread Switching Revisited



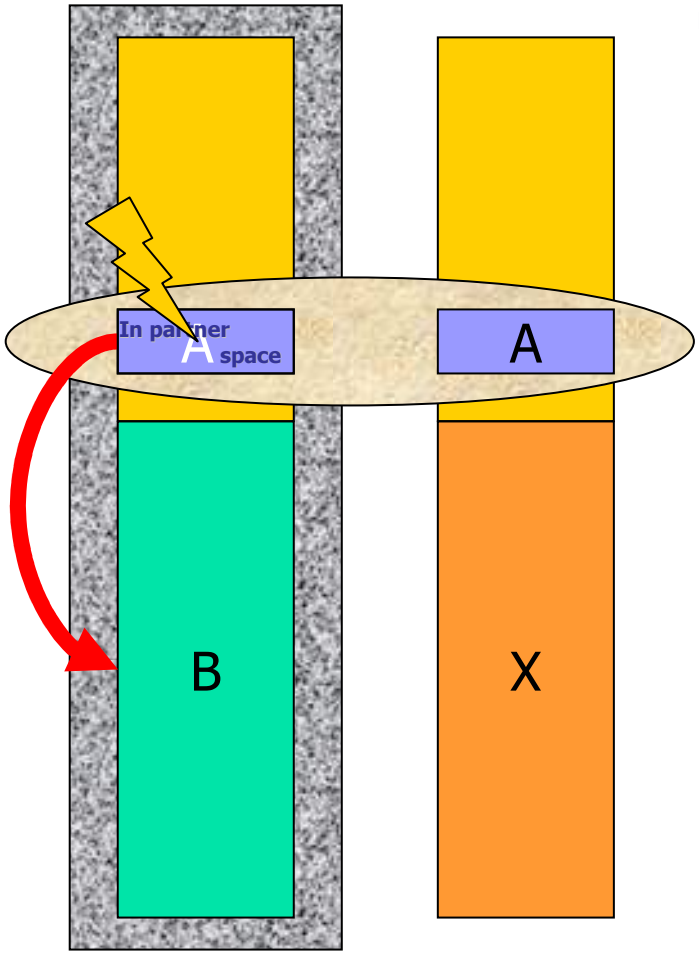
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**

Thread Switching Revisited



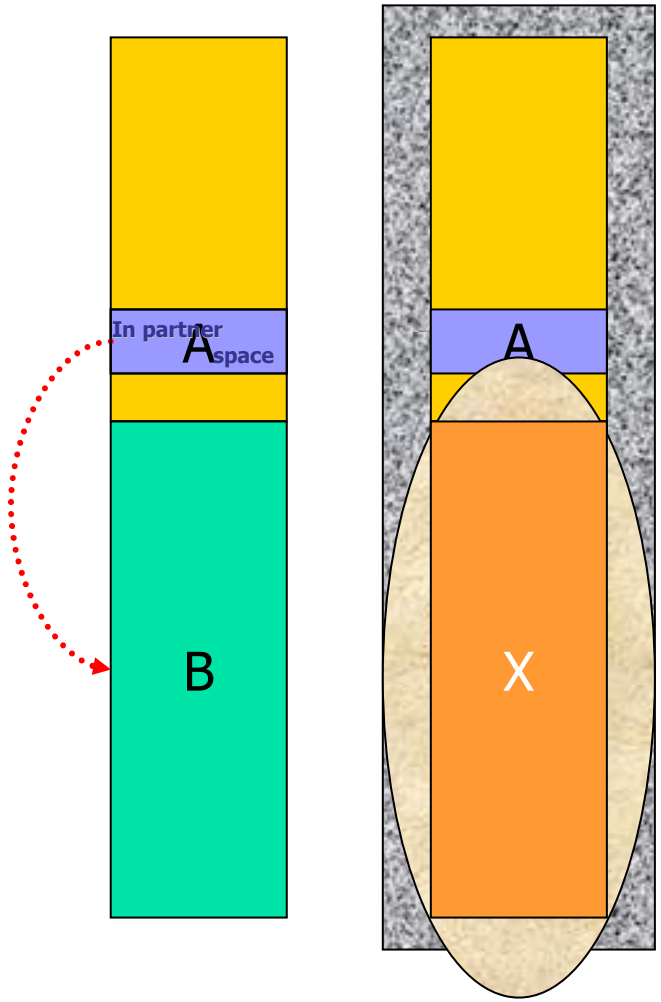
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**
- Preemption or PF resumes in **A**, but **A** now executes in HwAS **X**

Thread Switching Revisited
A Solution



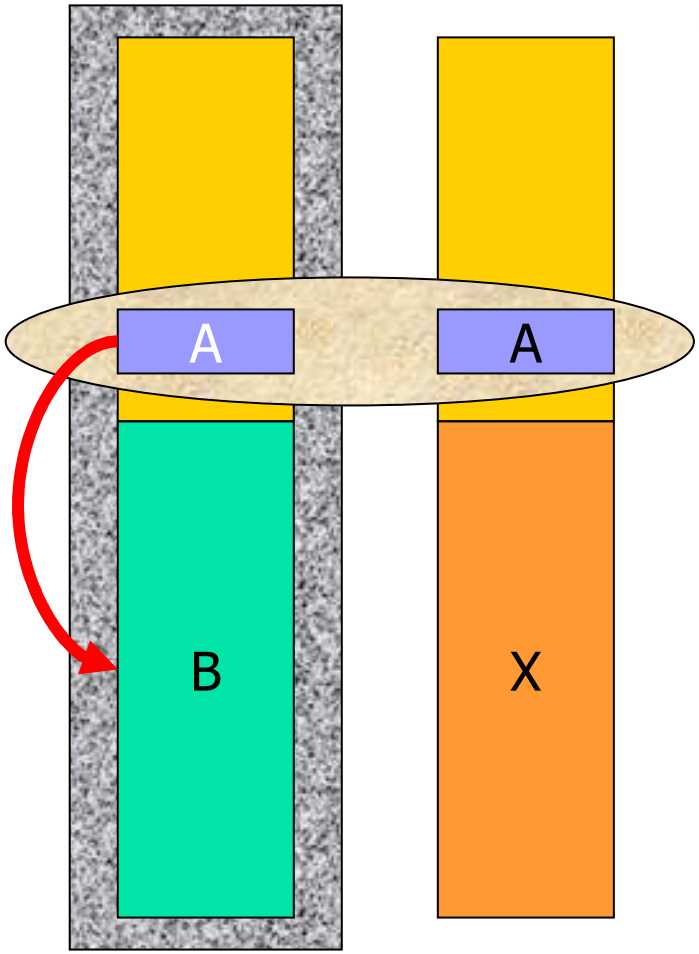
- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
    - Mark **A** **"in partner space"**

Thread Switching Revisited
A Solution

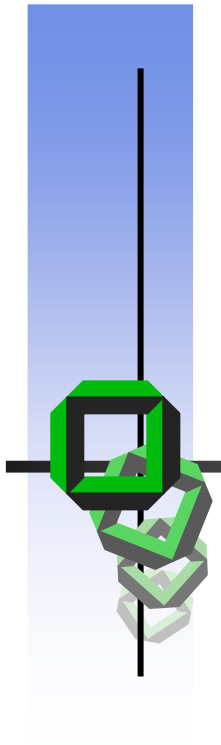


- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
    - Mark **A** **"in partner space"**
  - Thread switch from **A** to **X**

Thread Switching Revisited
A Solution

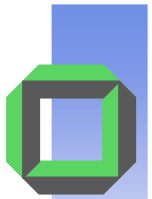


- **A** sends to **B**, executes in HwAS **B**
  - **A** is preempted or PF in **A**
    - Mark **A** **"in partner space"**
  - Thread switch from **A** to **X**
  - Switch back from **X** to **A**
    - Also switch HwAS to HwAS **B**



Small Address Spaces and Fast System Calls

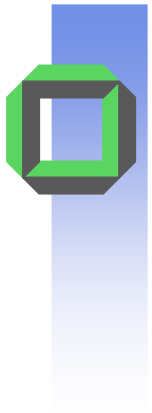
Getting around automatic segment register reloading



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**



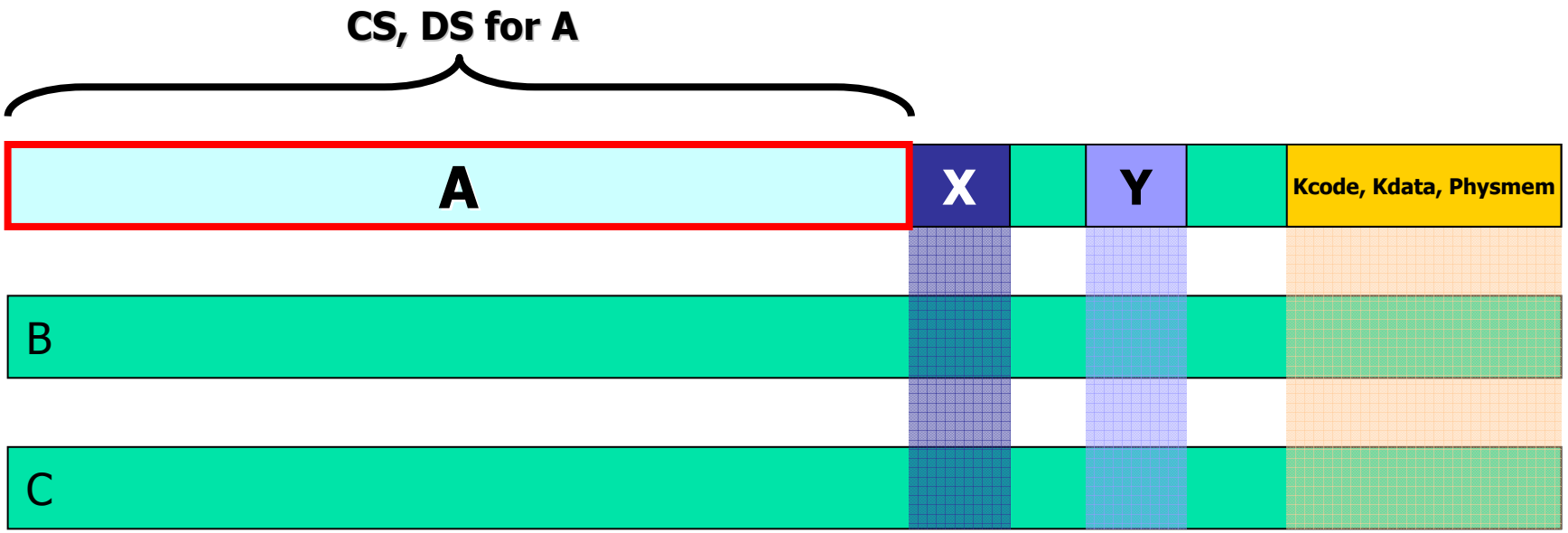
Fast System Calls

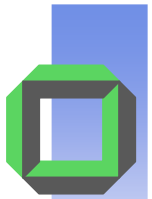
- Optimized instructions
  - sysenter/sysexit (Intel)
  - syscall/sysret (AMD)
- Faster than software interrupts
  - Can avoid certain checks
  - Unconditionally reload segment registers (page based protection)

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

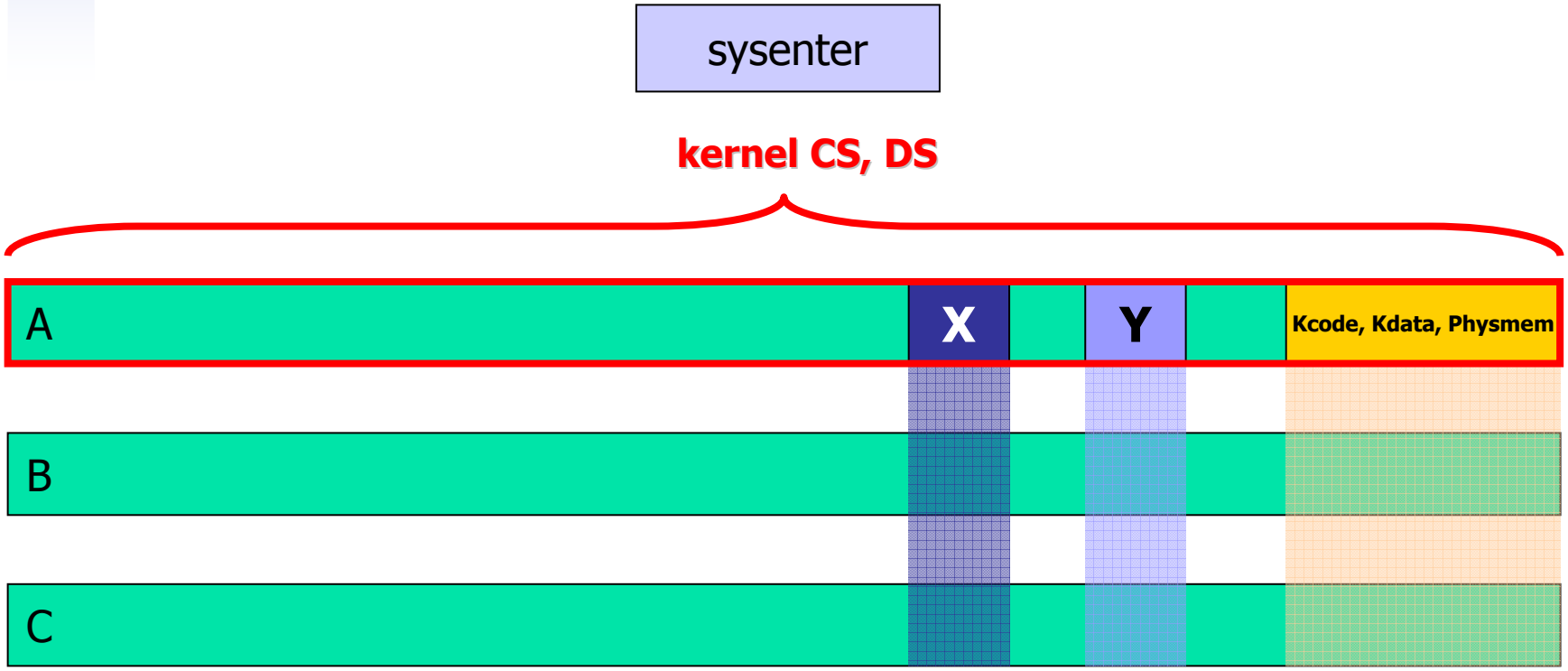
Problematic for small spaces (segment based protection)

Fast System Calls
Automatic Segment Register Reloading

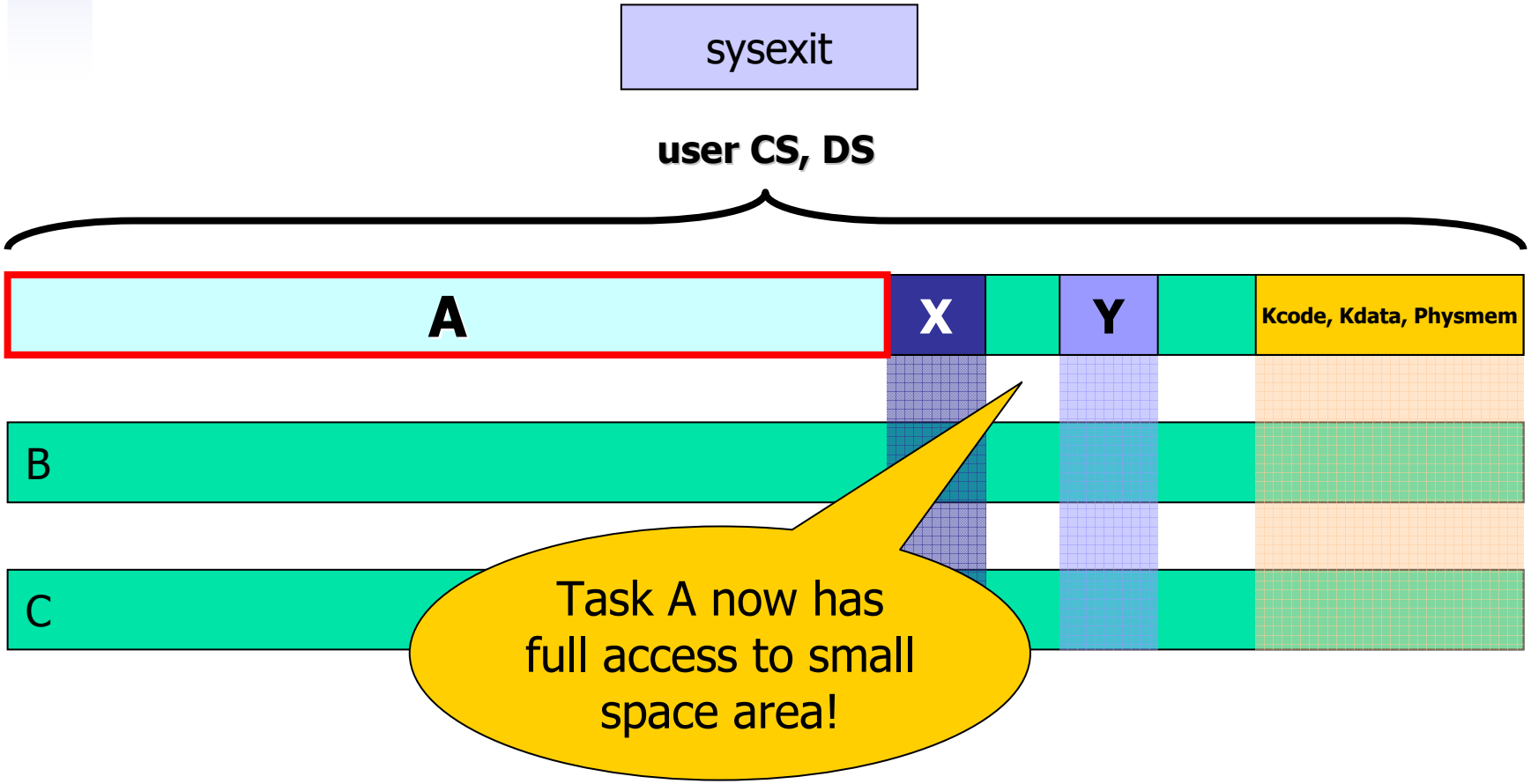




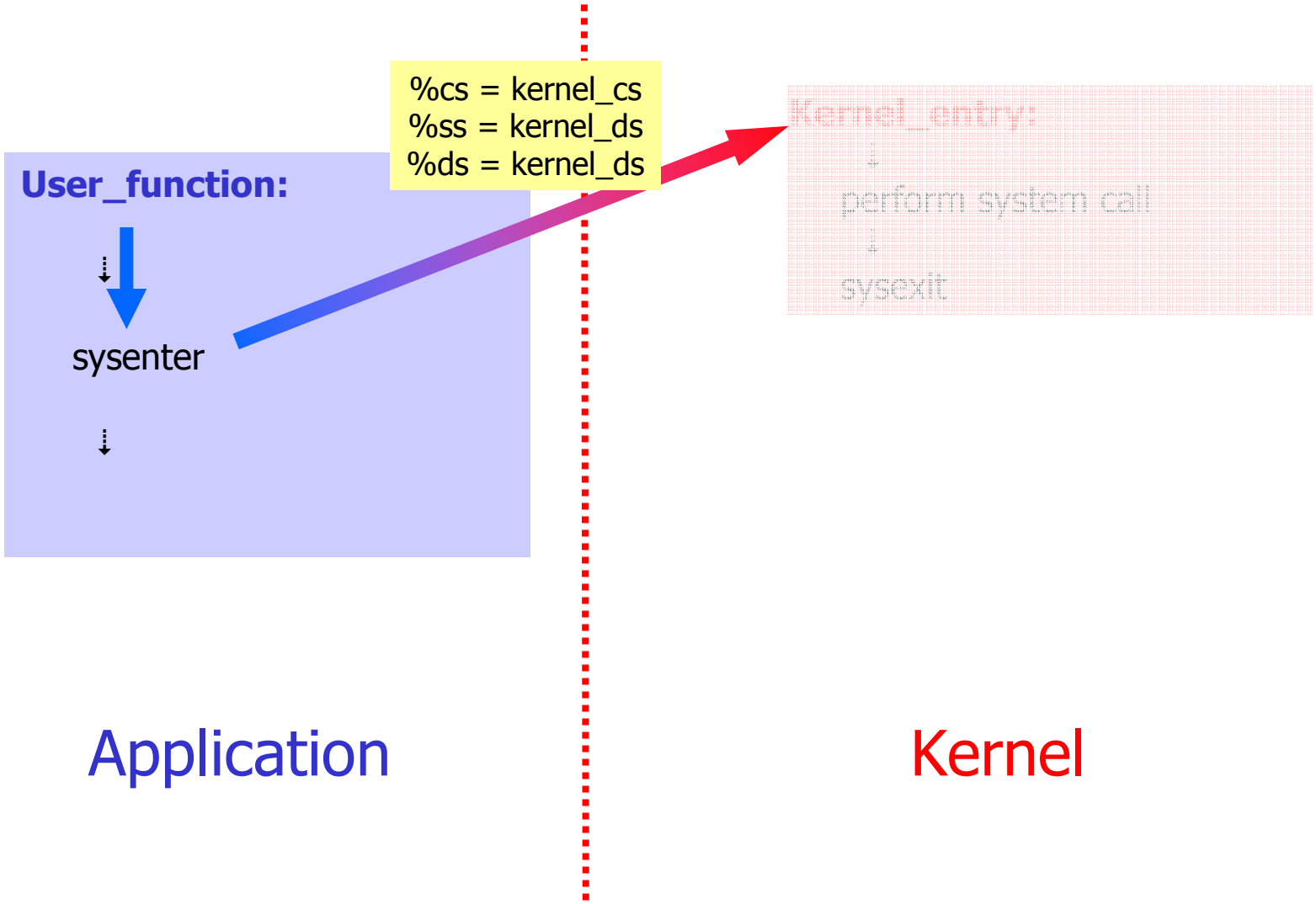
Fast System Calls
Automatic Segment Register Reloading



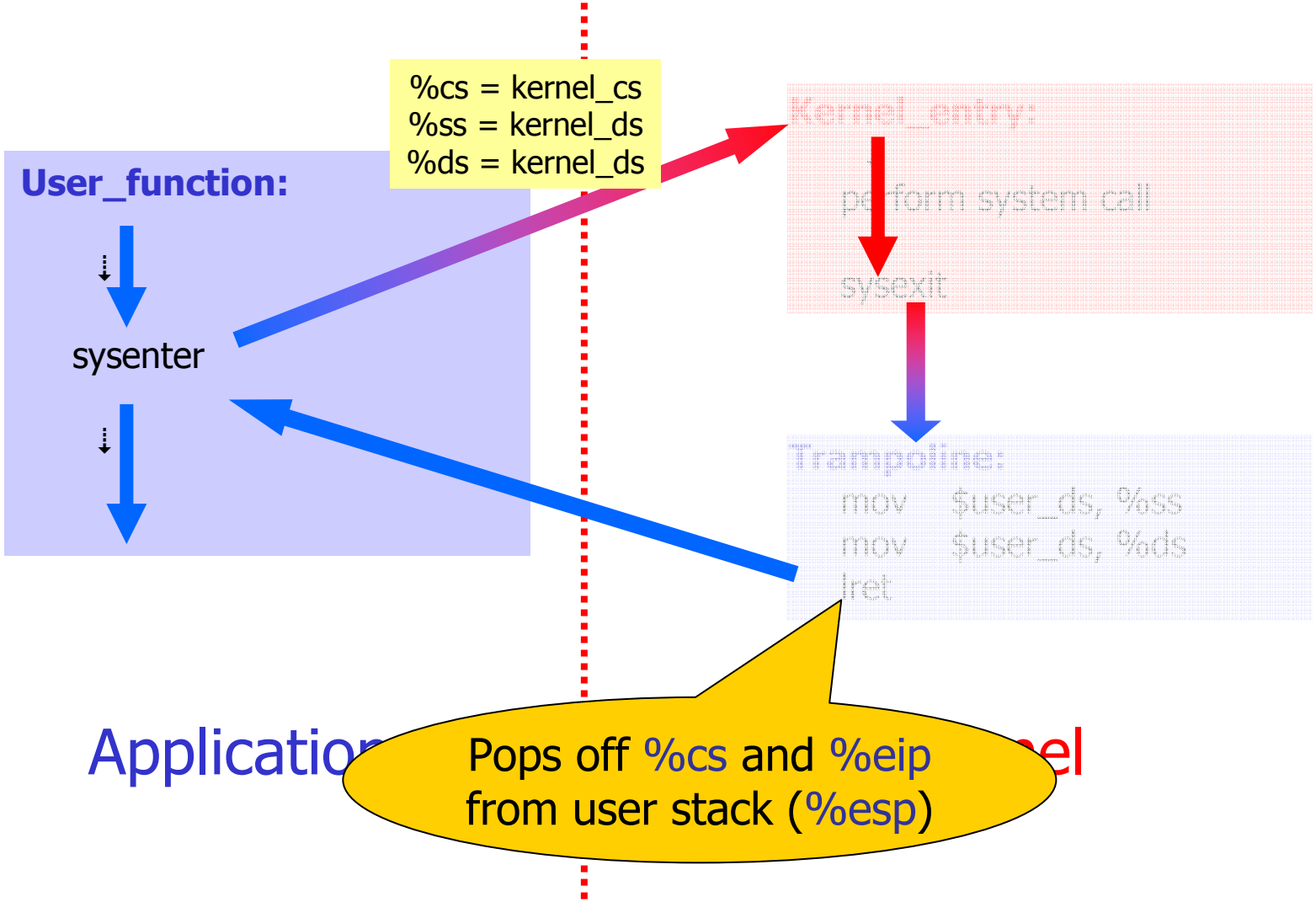
Fast System Calls
Automatic Segment Register Reloading



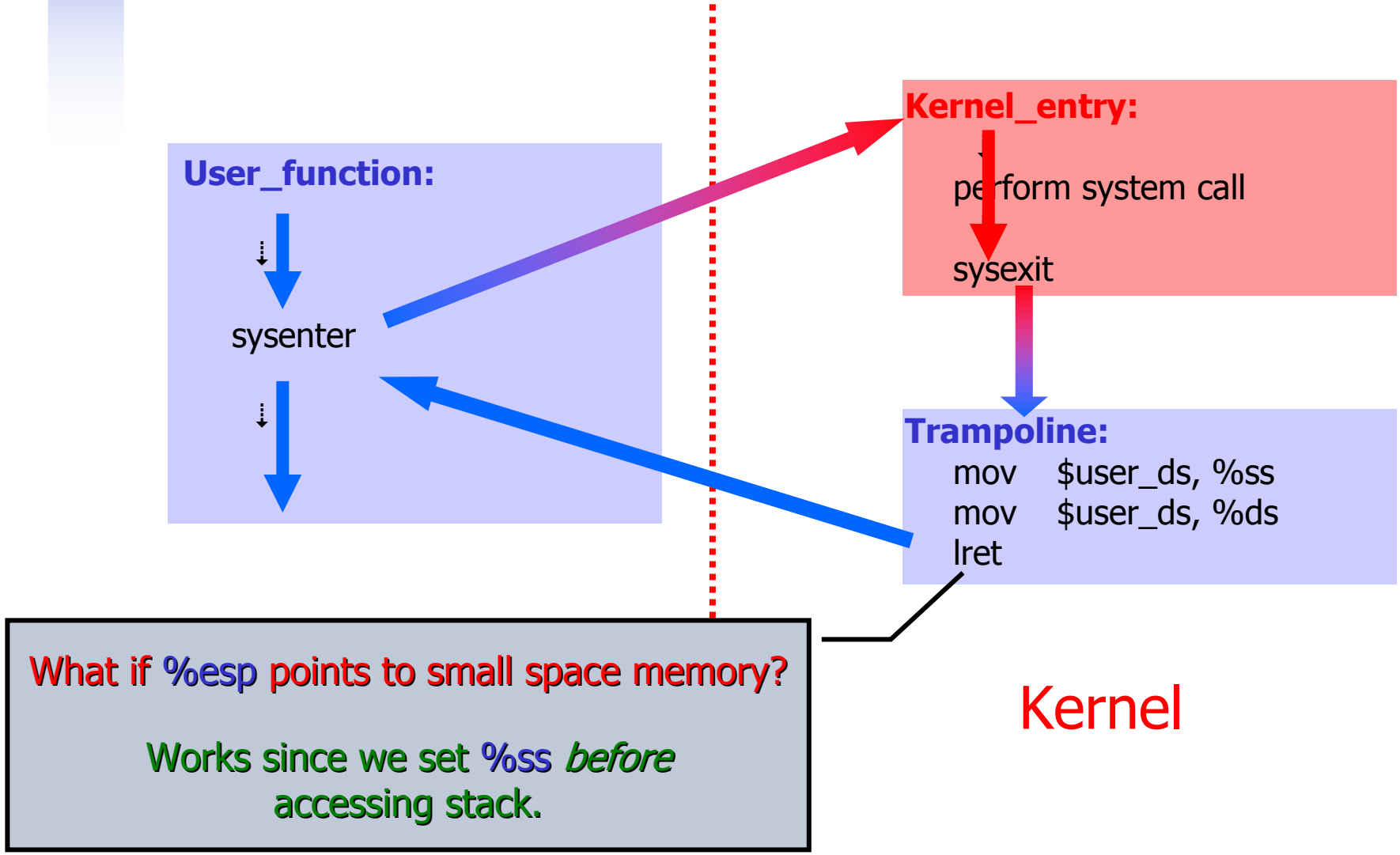
Automatic Segment Register Reloading Solution – In-Kernel Trampoline



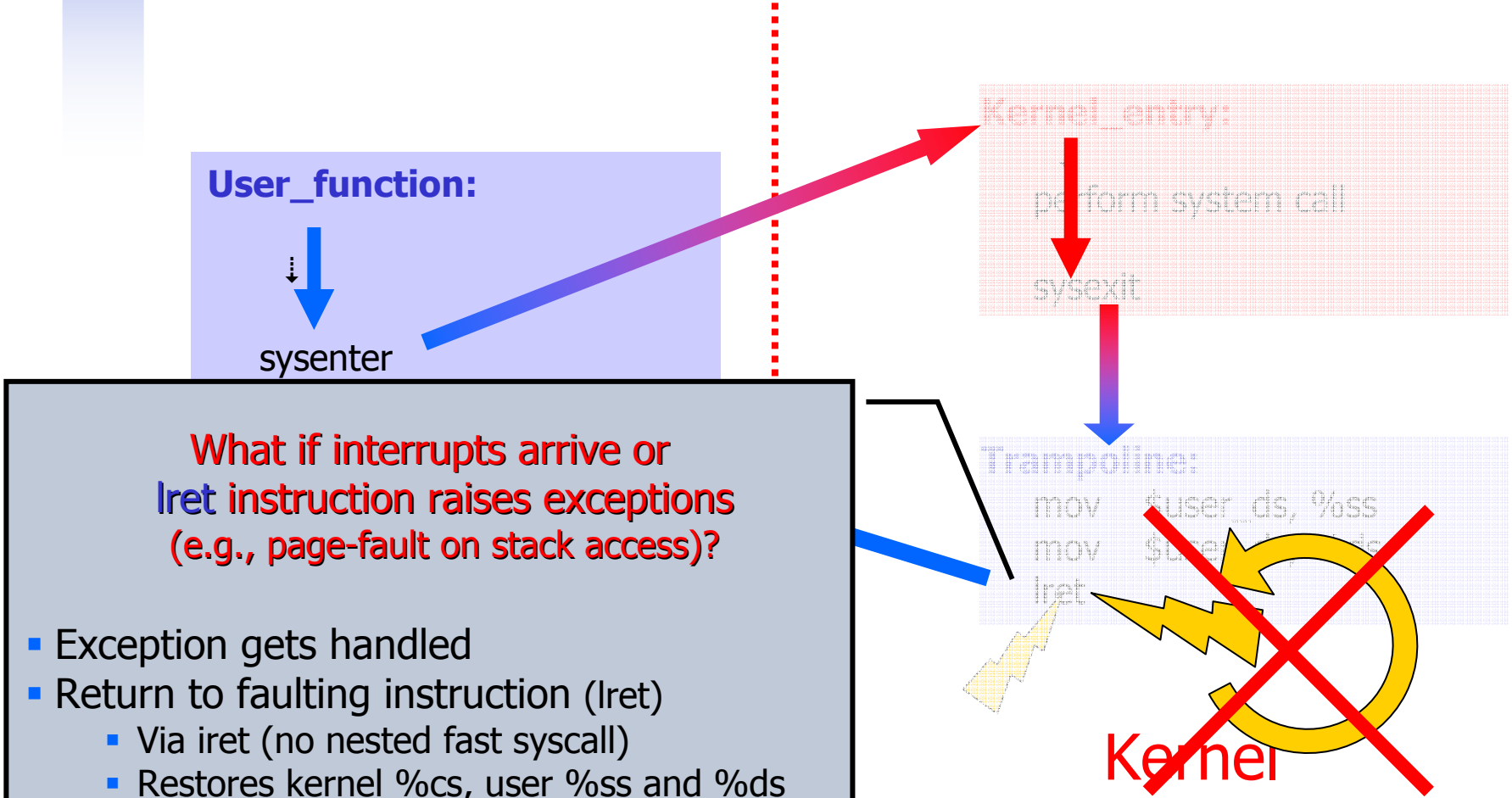
Automatic Segment Register Reloading Solution – In-Kernel Trampoline



Automatic Segment Register Reloading Solution – In-Kernel Trampoline

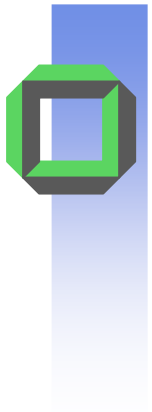


Automatic Segment Register Reloading Solution – In-Kernel Trampoline



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

- Exception gets handled
- Return to faulting instruction (Iret)
  - Via iret (no nested fast syscall)
  - Restores kernel %cs, user %ss and %ds
- Continue to completion



Outlook on IA32 + TLB Tagging
AMD Opteron

- "Flush Filter", Patent 6,604,187
  - Automatic ASN tagging
- Lookup cache, storing ptab lookup chain
- Tag TLB as "to-be-flushed"
  - If entry in cache gets modified (mem snooping)
  - If cache overflows
- Selective flushing on CR3 reloads

- Pistachio IPC: 750 cycles → 240 cycles