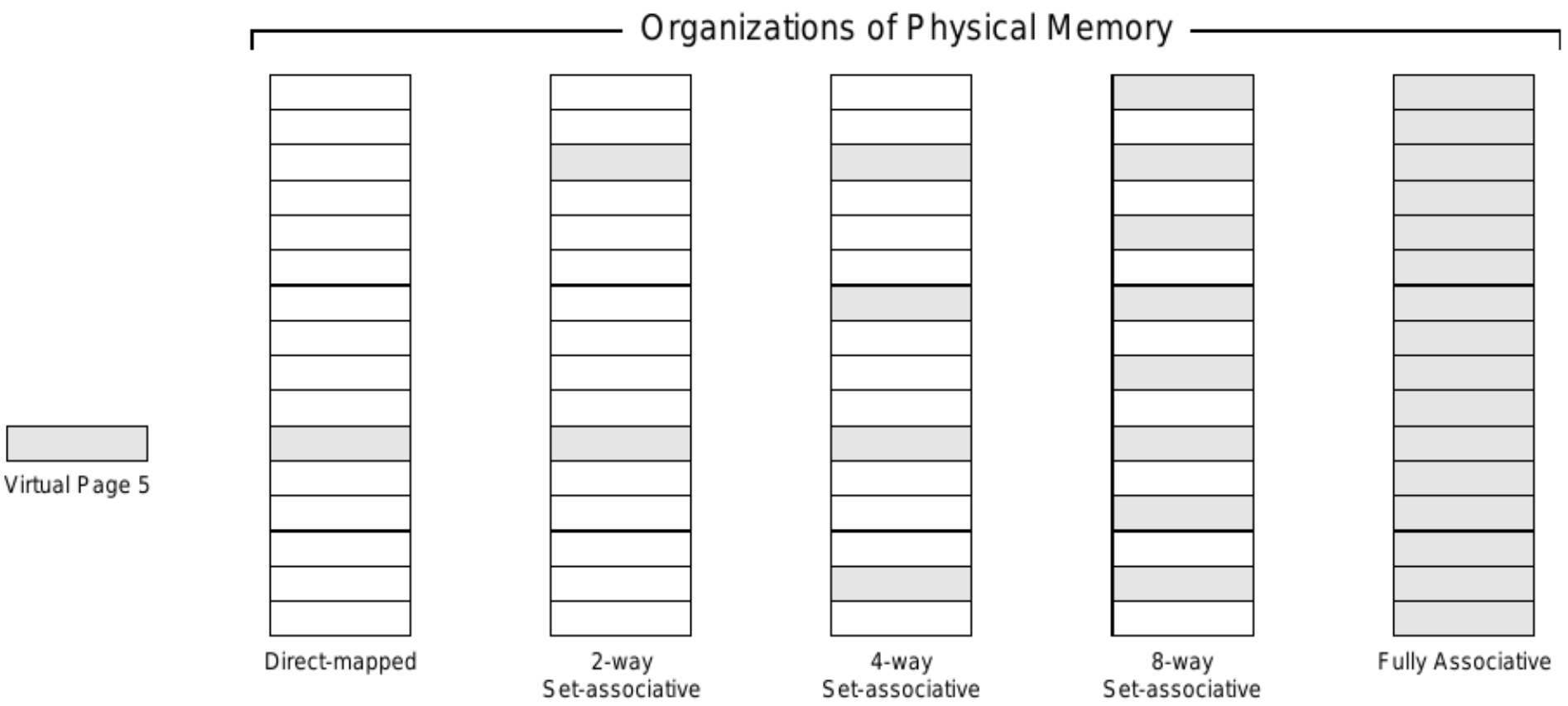
Pamięć wirtualna

Rafał Łasocha

Basic functions of virtual mem.

- create virtual machine environment for every process
- demand-paging – granularity for process residence than an entire address space
- virtual-address aliasing (shared objects at different virtual addresses)
- protection aliasing (shared resources may have different protections dependent on process)
- support for virtual caches

Memory is a cache for permanent store



defs

- page tables – tables which keep mappings between virtual and physical memory
- page table entries – entries of former

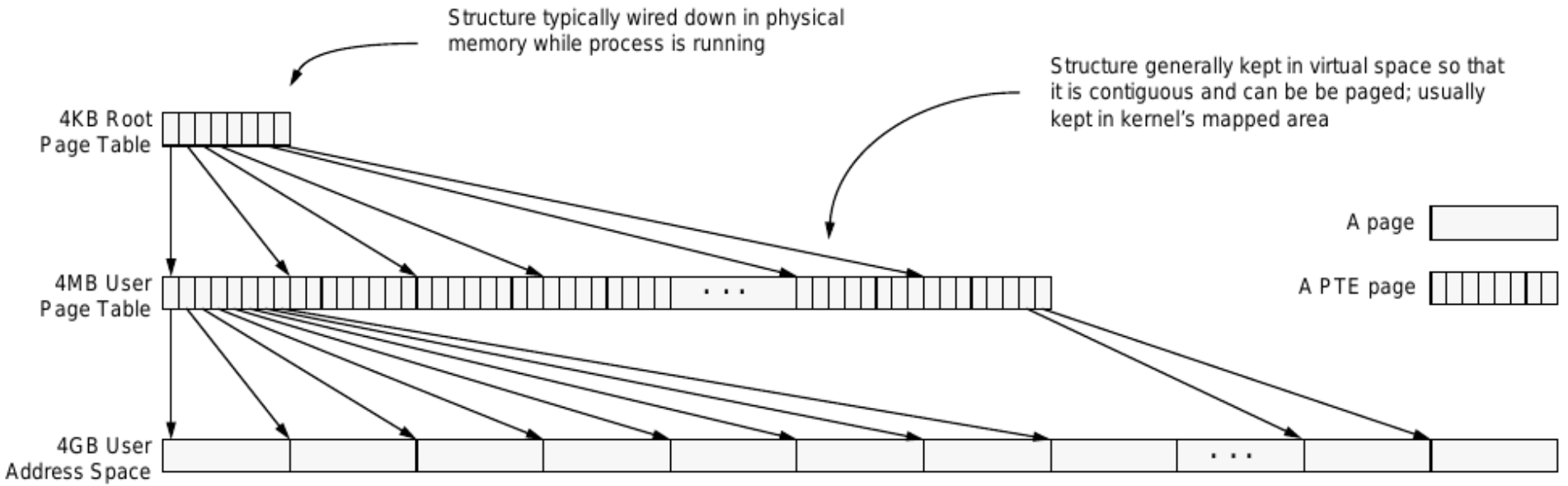
What info OS wants to have from PTE

- virtual page number (VPN)
- physical frame number (PFN) or location on disk
- ID of the page's owner (ASID)
- page's protection information
- aid in making replacement decision (whether it was recently accessed / written)
- is it valid (valid bit)

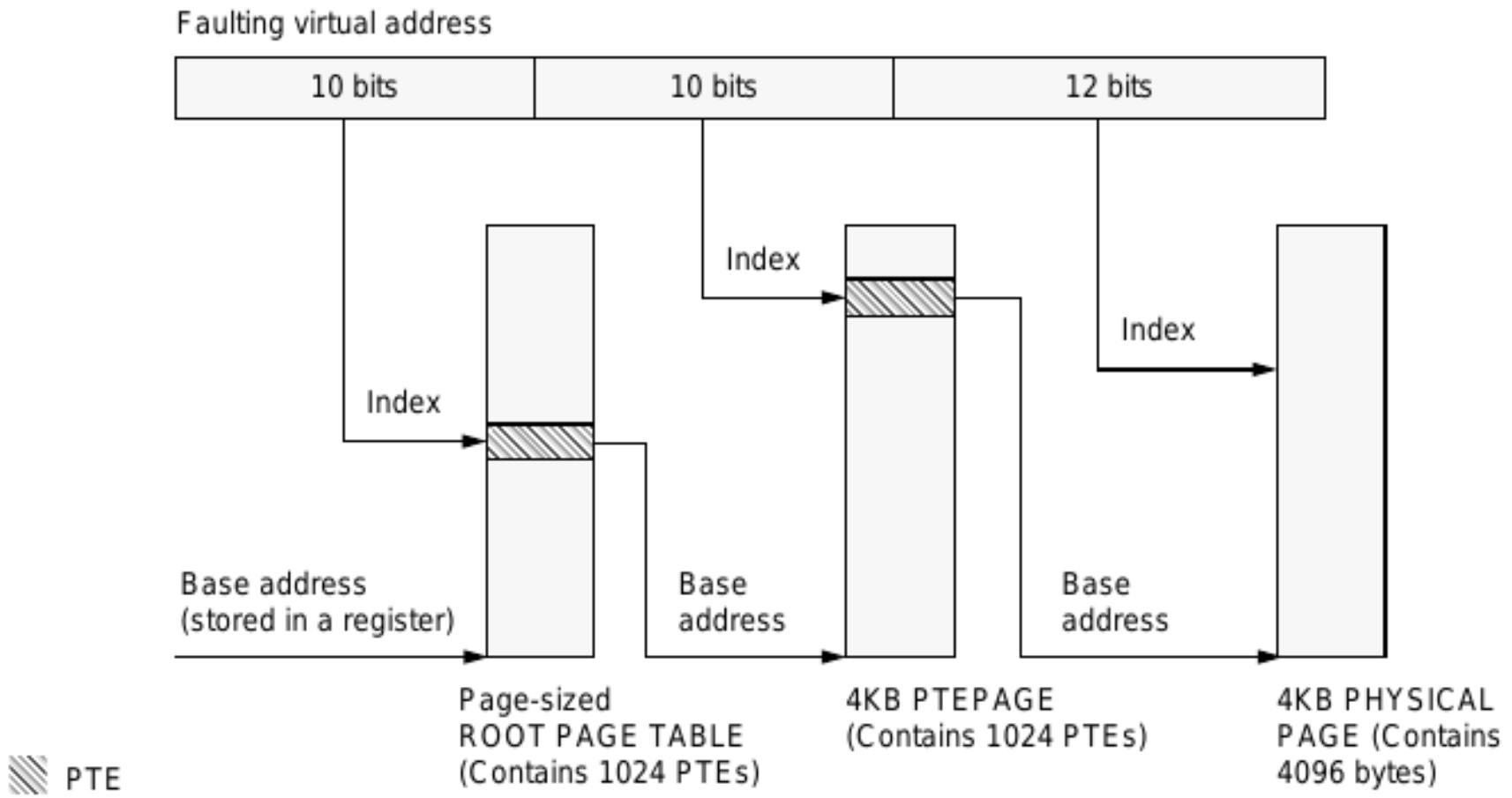
Page table organization

- hierarchical page table
  - top-down traversal (IA-32)
  - bottom-up traversal (MIPS, Alpha)
- inverted page table
  - basic idea
  - modifications in PA-RISC
  - modifications in PowerPC

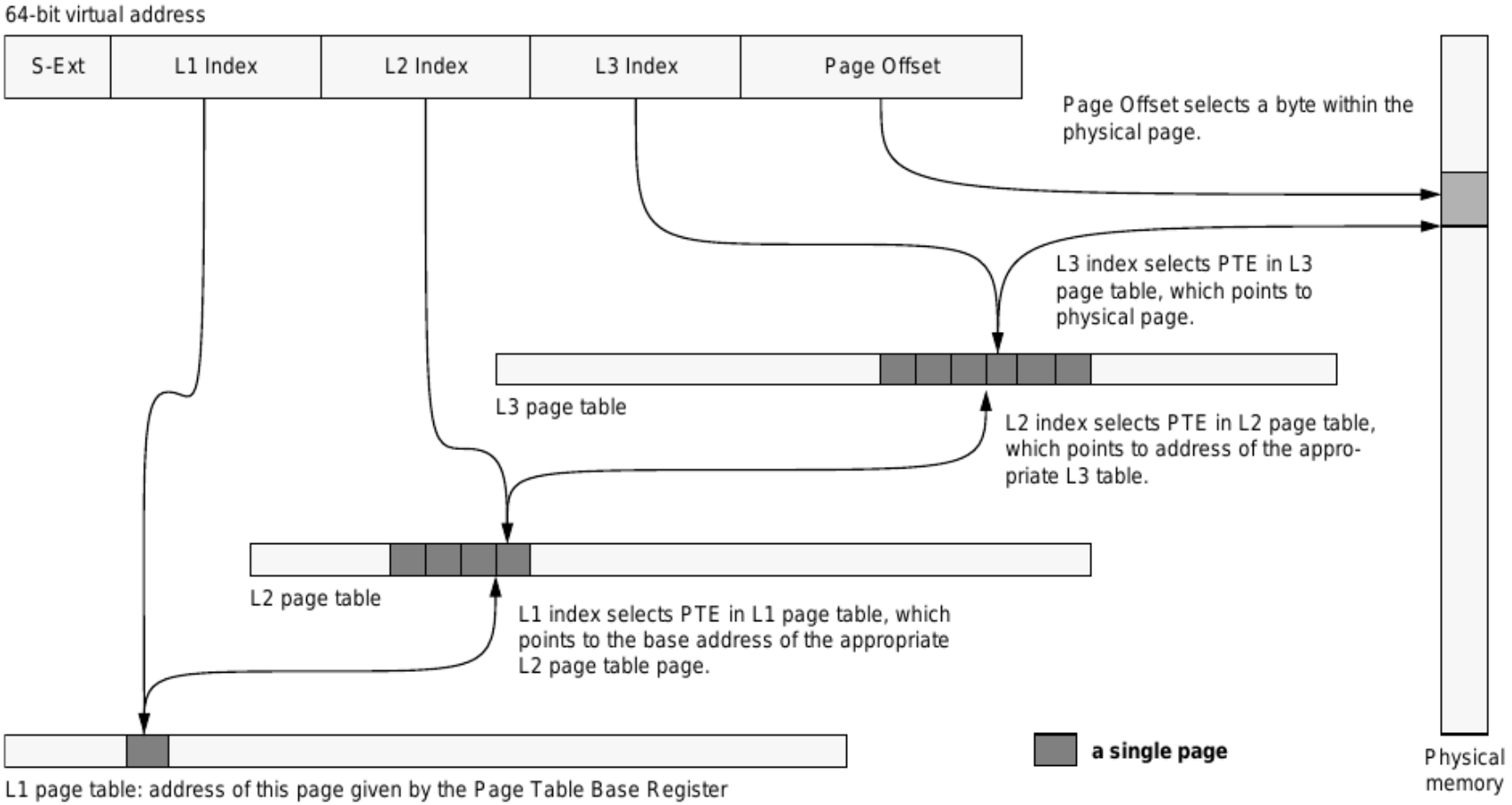
Hierarchical Page Table



Top-down approach



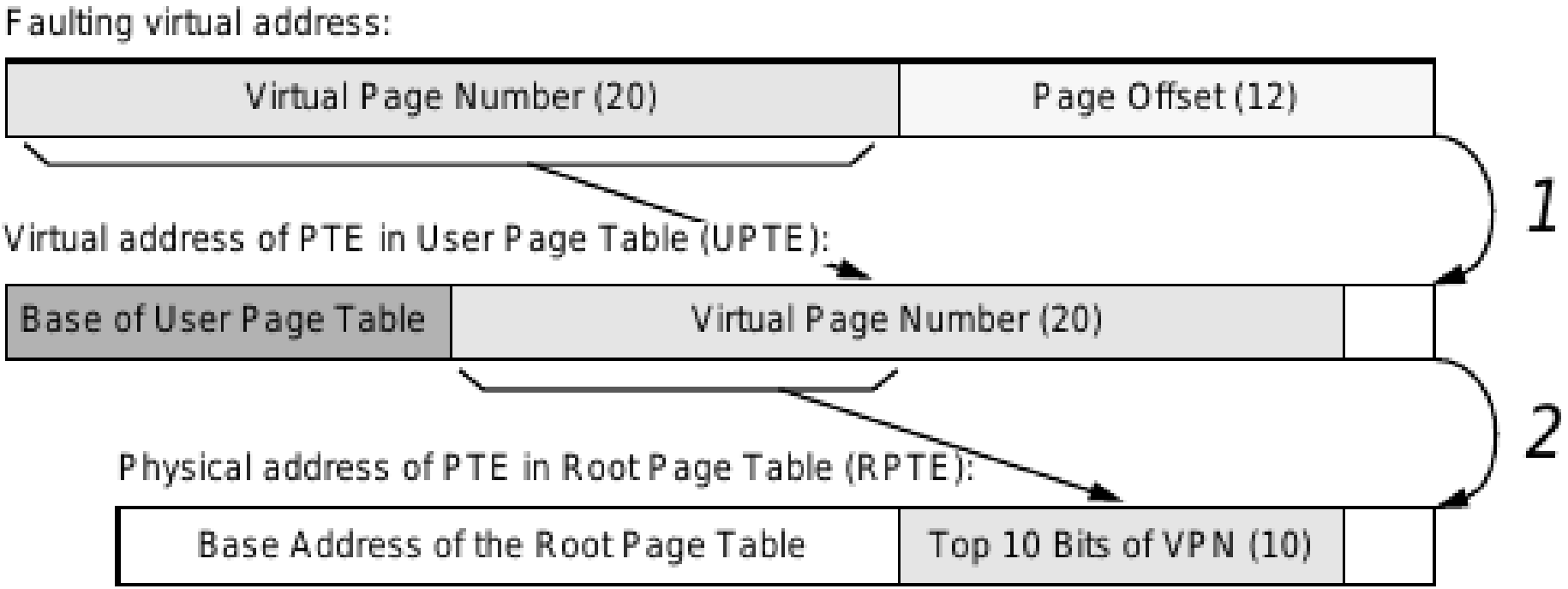
3-level page table in Alpha



Bottom-up approach

- 3 memory references to get data in top-down approach for 32-bit architecture
- even more for 64-bit
- User Page Table is contiguous
- we can use that fact and decrease memory references to 2 in best-case scenario (in both 32 and 64-bit architectures)
- and use top-down approach only when we have page fault

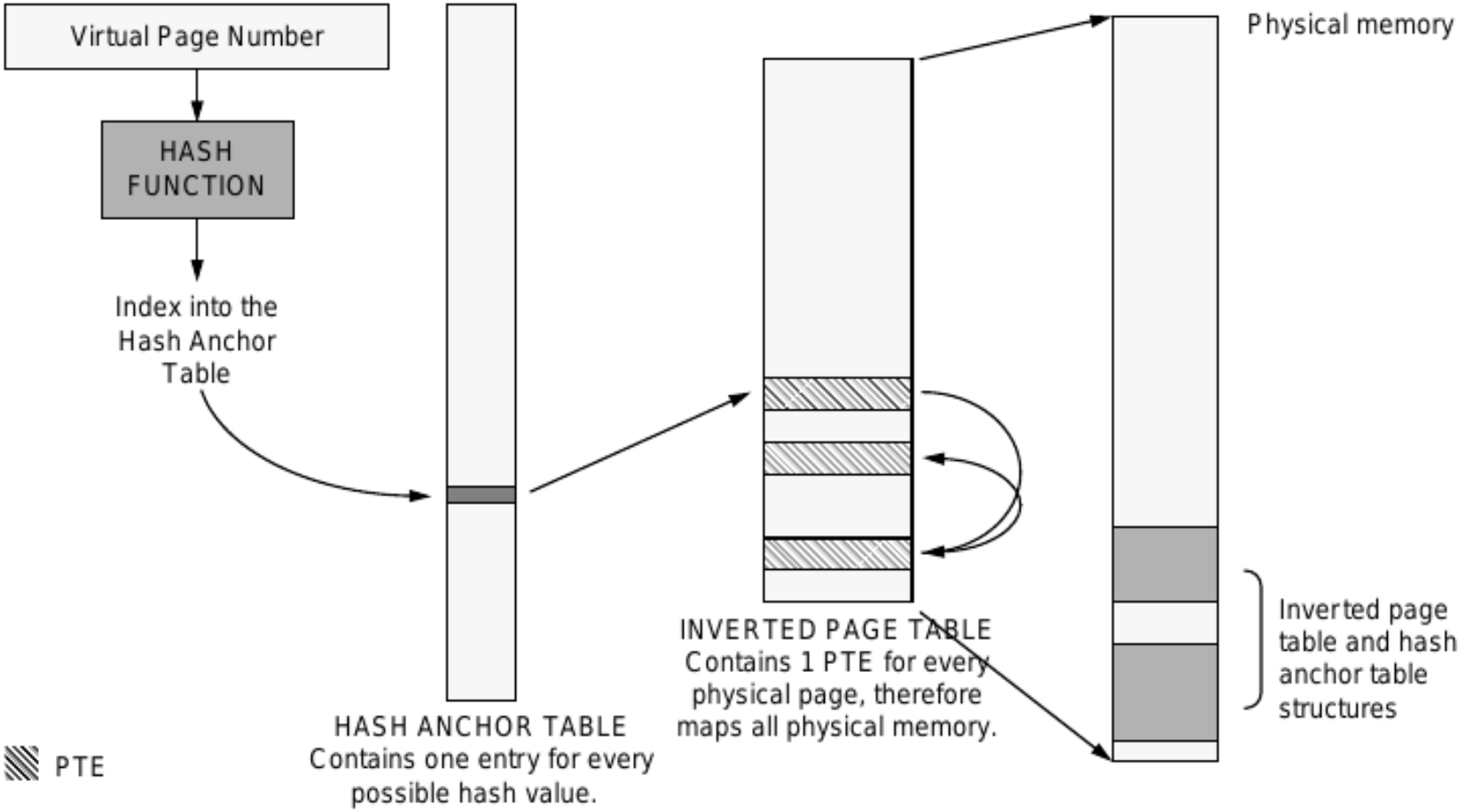
Bottom-up approach



Inverted page table

- instead of table indexed by VPN and entries containing PFN, let's index it with PFN, and keep entries to VPN (and ASID)
- PFN is implicit
- instead of scaling with virtual address space, we scale with physical memory
- we still want use that page table to find *PFN*, given *VPN*
  - linear lookup – nope
  - hashing to the rescue

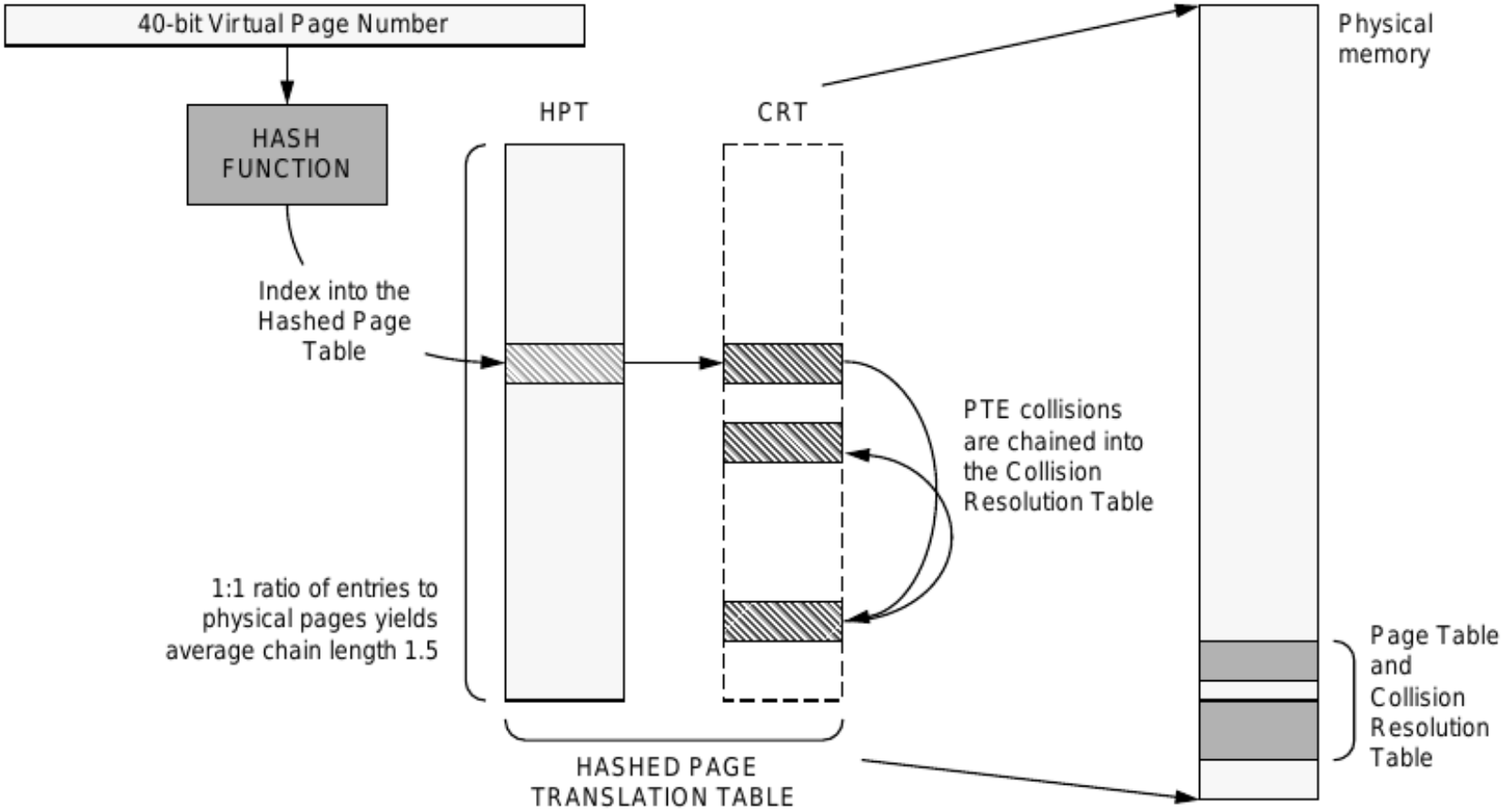
Inverted Page Table



PA-RISC

- at least 2 mem. ref. - we can do better than that
- let's remove access to HAT
- and just make IPT bigger
- disadvantage: we lose implicit information about PFN, so we have to keep it explicitly

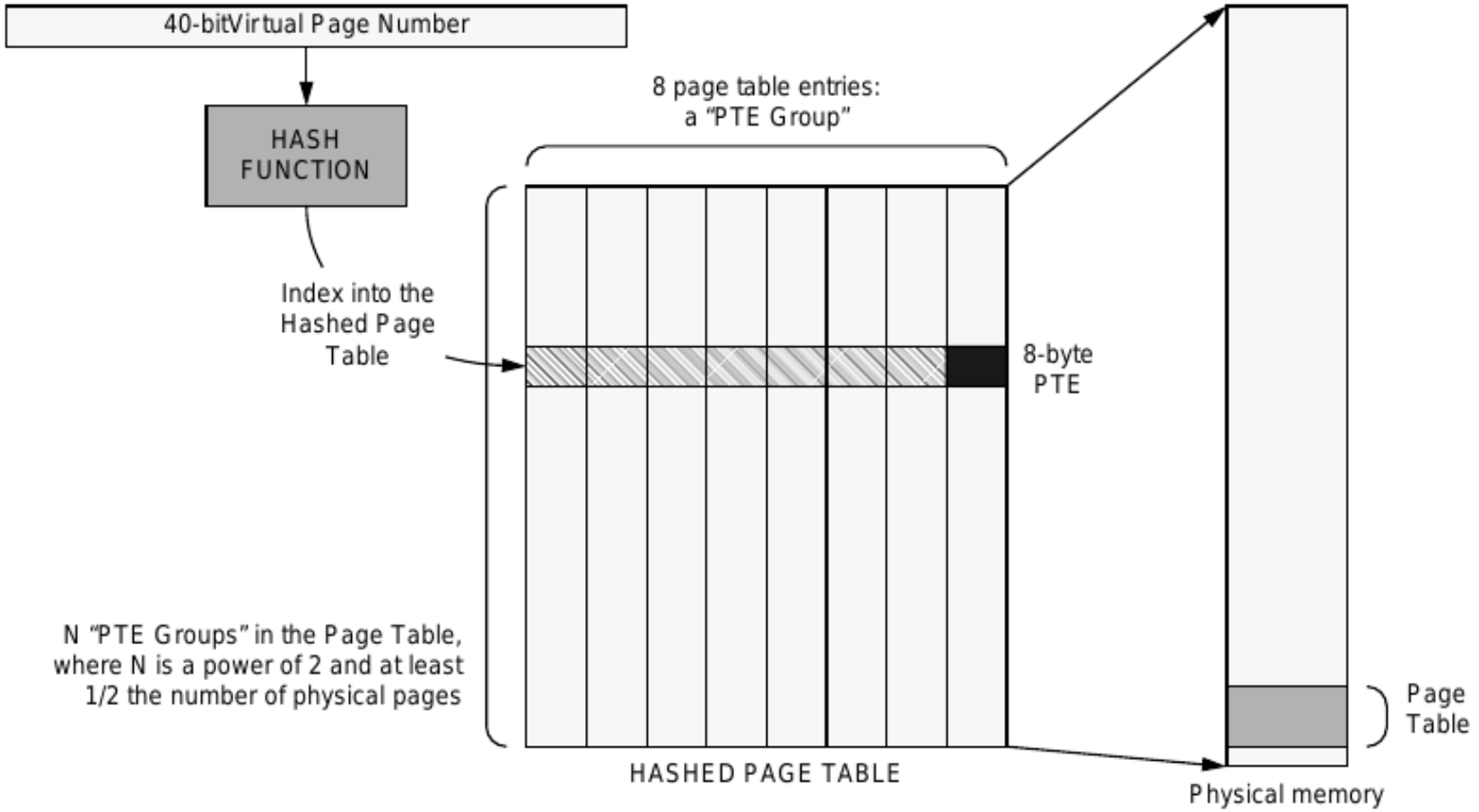
PA-RISC


40-bit Virtual Page Number
HASH FUNCTION
Index into the Hashed Page Table
HPT
CRT
Physical memory
PTE collisions are chained into the Collision Resolution Table
1:1 ratio of entries to physical pages yields average chain length 1.5
Page Table and Collision Resolution Table
HASHED PAGE TRANSLATION TABLE

PowerPC

- let's limit chain's length by constant (8)
- and keep whole chain in contiguous space
- then we can fetch all PTE together and check whole chain at once
- however, if chain's length is bigger than 8, we delegate this problem to OS
- each PTE is also bigger, but we have only one (big) memory reference

PowerPC


40-bitVirtual Page Number
HASH FUNCTION
Index into the Hashed Page Table
8 page table entries: a "PTE Group"
8-byte PTE
N "PTE Groups" in the Page Table, where N is a power of 2 and at least 1/2 the number of physical pages
HASHED PAGE TABLE
Page Table
Physical memory

Inverted vs. Hierarchical

- another structure is needed to keep references to disk
- inverted cannot into shared memory
- bigger PTE make communication with cache harder
- process sparsely used memory is wasting it
-

Translation lookaside buffer (TLB)

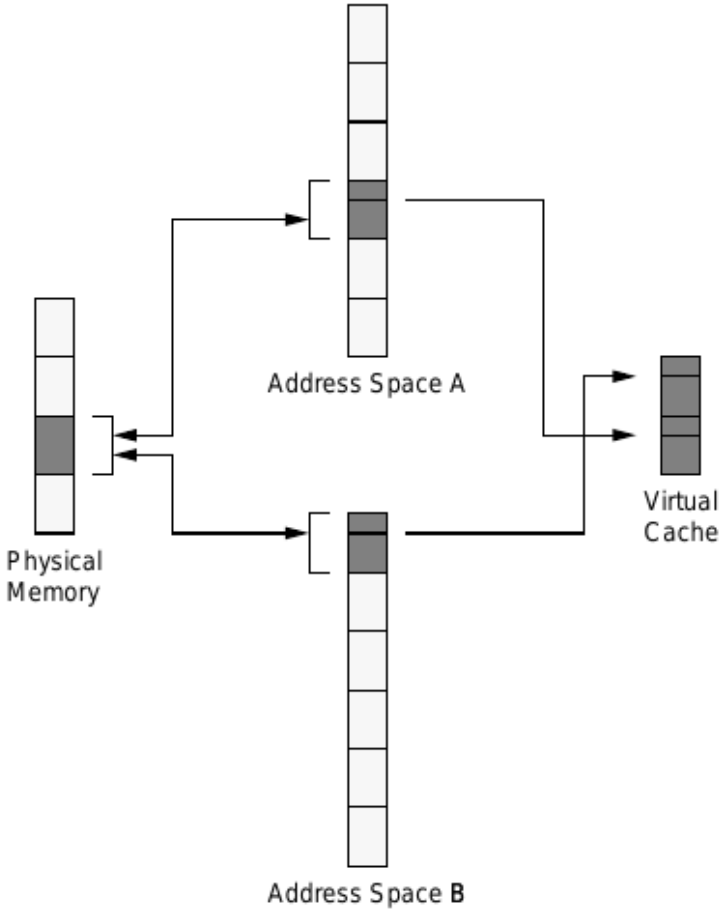
- page table cache (usually on CPU)
- some time ago, when address space were much smaller, one table mapping entire address space was small enough to be in hardware
- protection may block cache
- doesn't scale
- management: hardware, software (in OS)

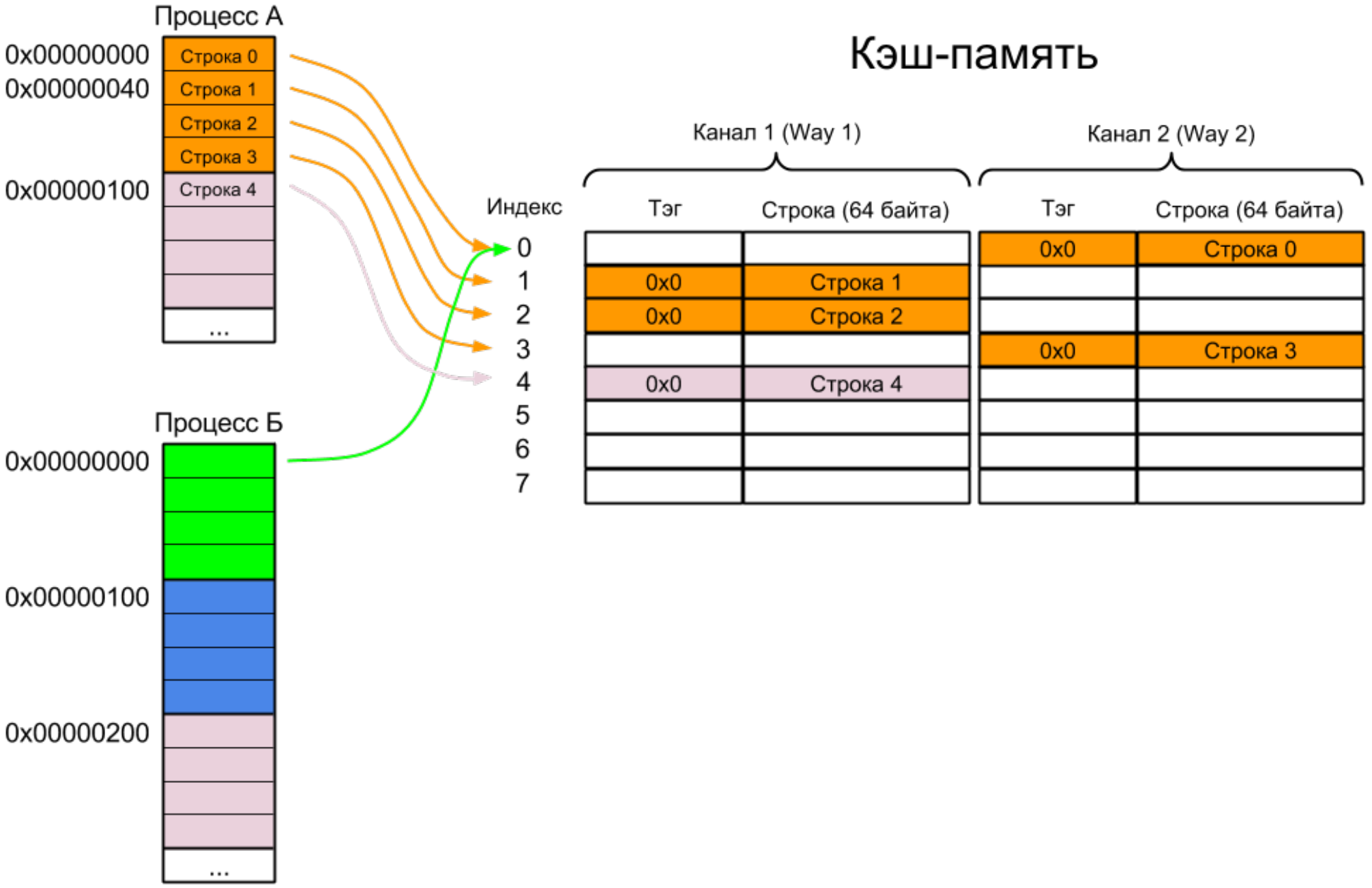
hardware vs. software TLB

- state machine walking page table when TLB miss
- PowerPC, x86
- PA-7200 – hybrid approach, part of HPT in hardware

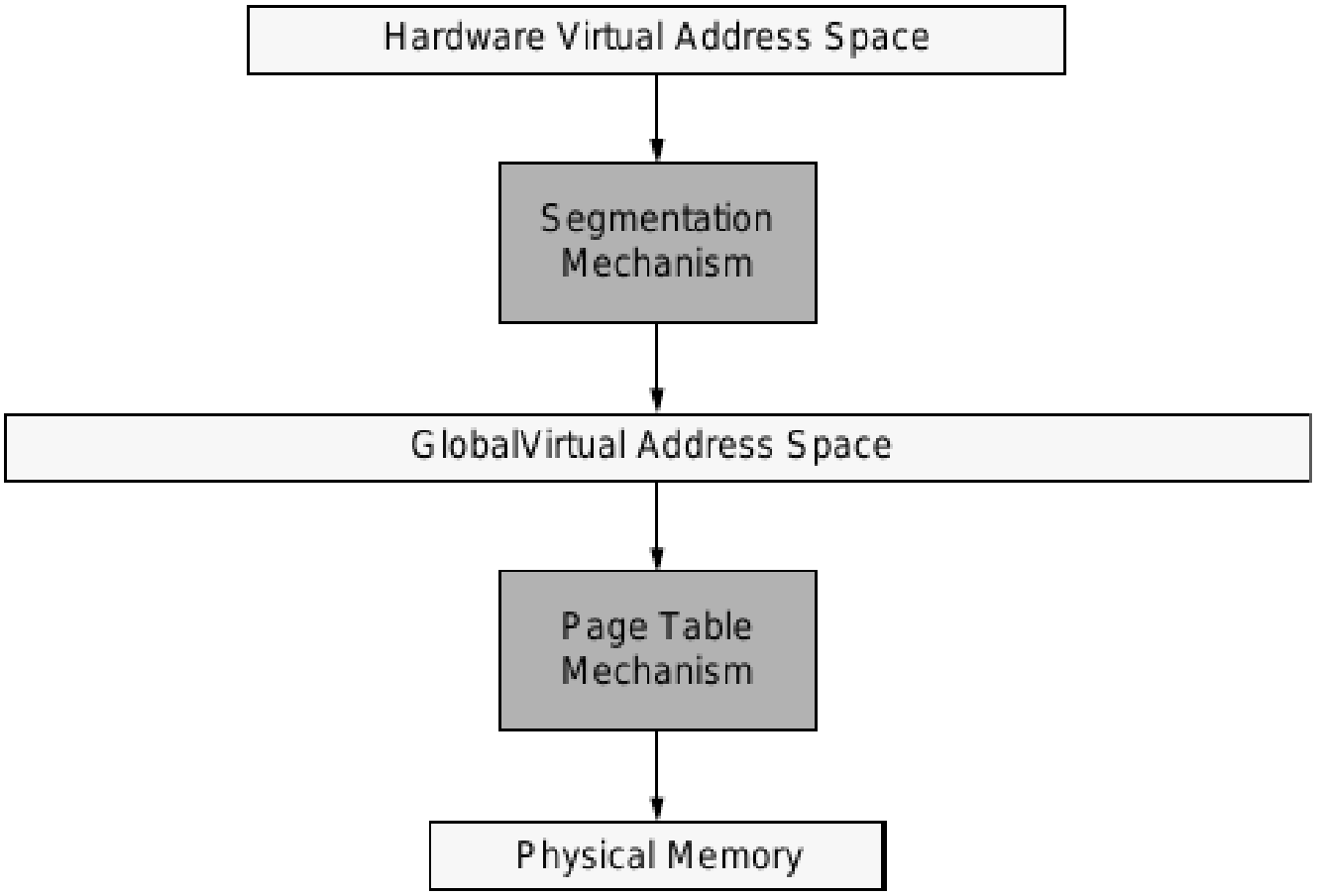
- slower
- interrupts
- flexible
- MIPS, SPARC, Alpha, PA-RISC

Synonym problem




Процесс А
0x00000000
0x00000040
0x00000100
Строка 0
Строка 1
Строка 2
Строка 3
Строка 4
...
Процесс Б
0x00000000
0x00000100
0x00000200
...
Индекс
0
1
2
3
4
5
6
7
Кэш-память
Канал 1 (Way 1)
Канал 2 (Way 2)
Тэг
Строка (64 байта)
Тэг
Строка (64 байта)
0x0
Строка 1
0x0
Строка 2
0x0
Строка 4
0x0
Строка 0
0x0
Строка 3

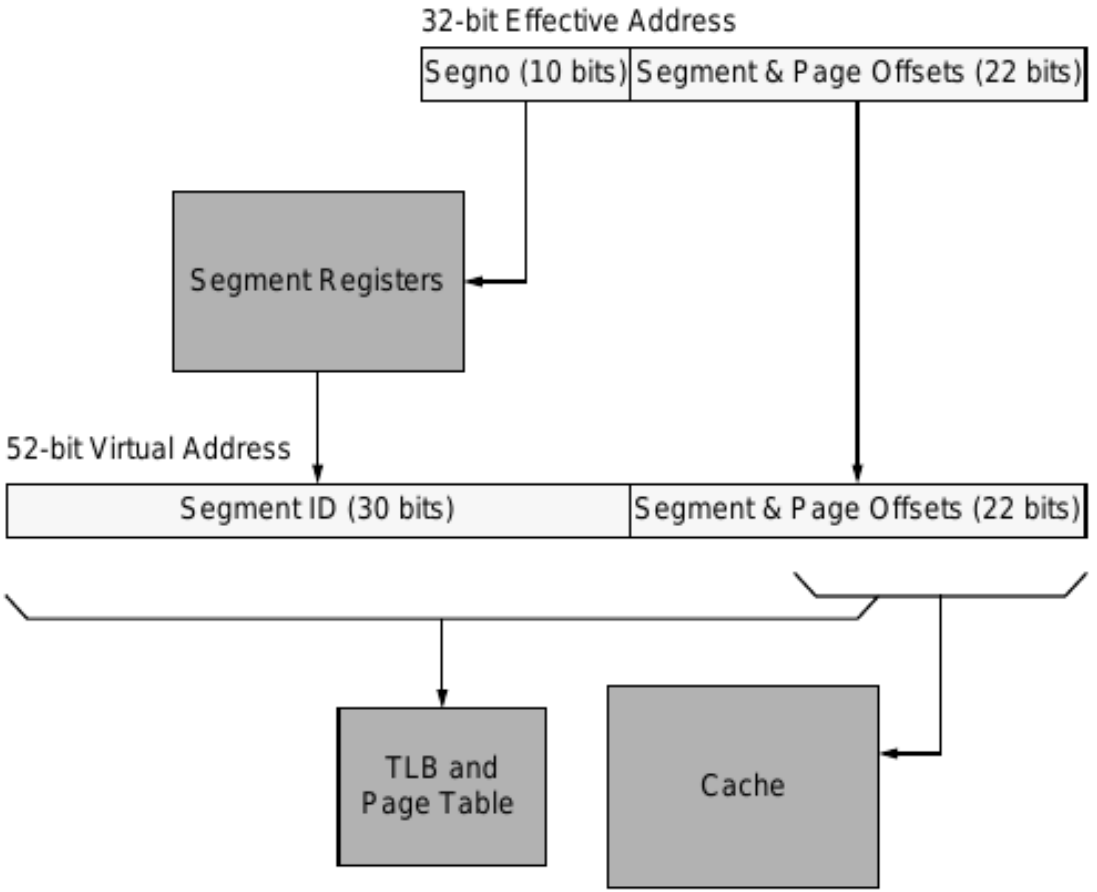
Global segmented space



Global segmented space

- if there's one global space, synonym problem disappears immediately
- segments are groups of pages
- segments may be scattered trough whole global space
- automatic address-space protection

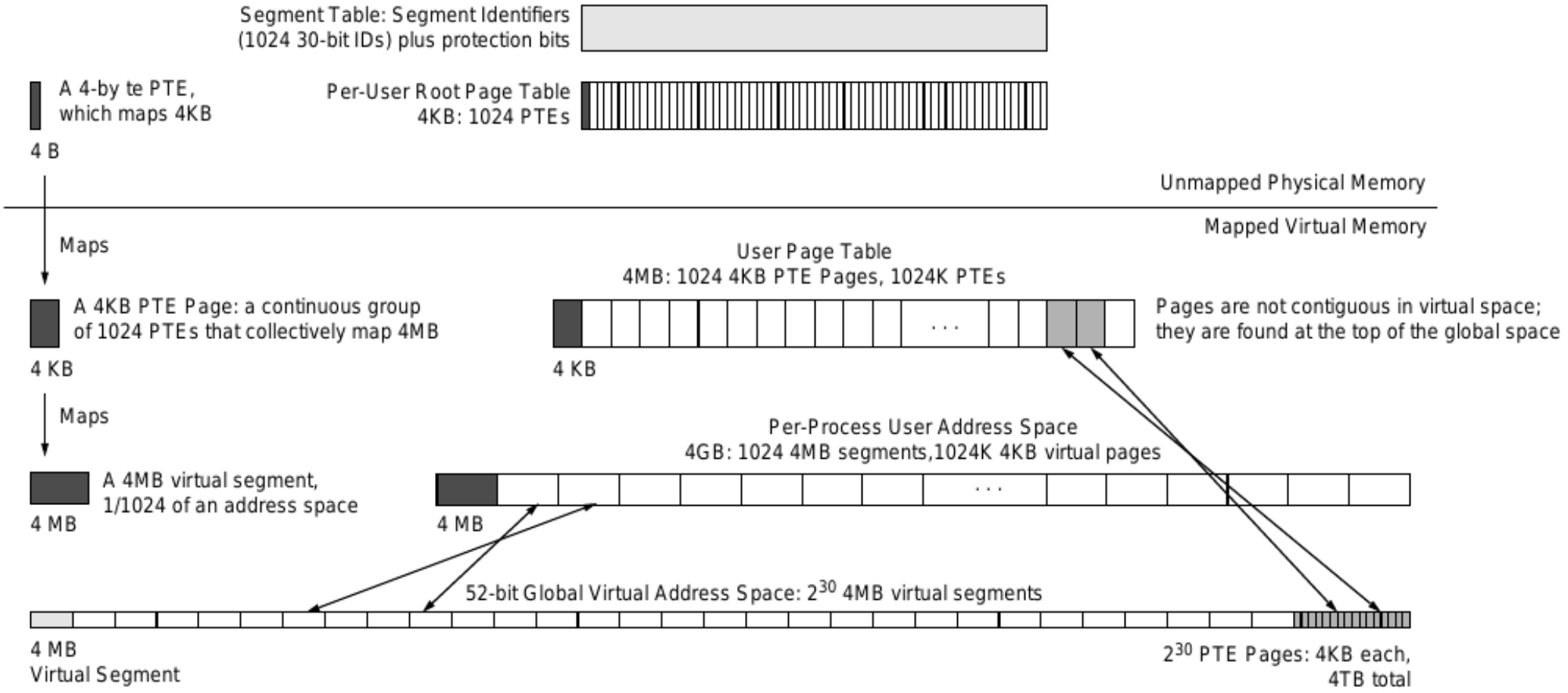
Global segmented space (example)



Global segmented space (example)

- per-user RPT – 4KB, UPT – 4MB, User Address Space - 4GB
- whole space: 2^{52} = 4PB
- Global Page Table – contiguous, 4TB in the end of whole global space
- per-user RPT are small enough (just like RPT) to be wired down in physical memory

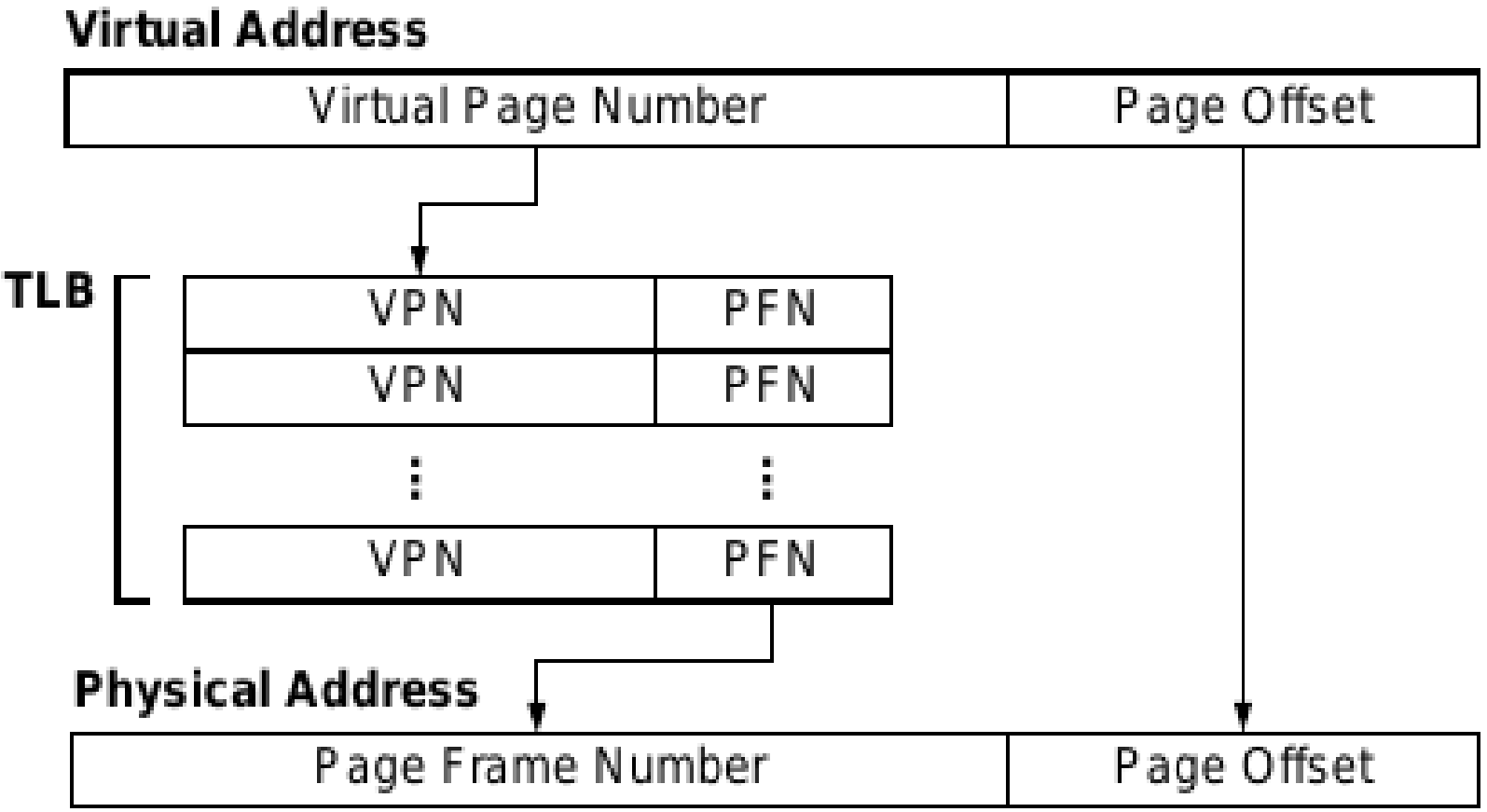
Global segmented space (example)



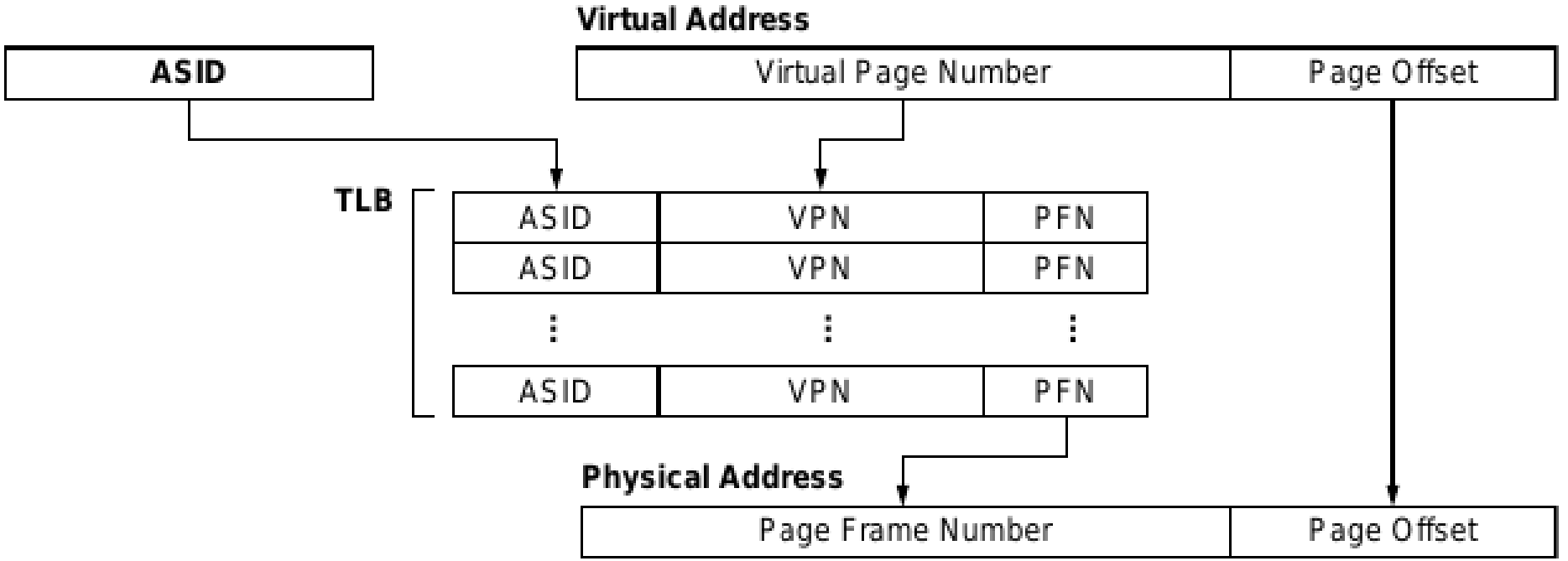
Address Space Organizations

- virtual space may be categorized by:
  - owner:
    - single owner (basic implementation)
    - multiple owner (segmentation)
  - used identifiers (protection)
    - no ID
    - single ID
    - multiple ID

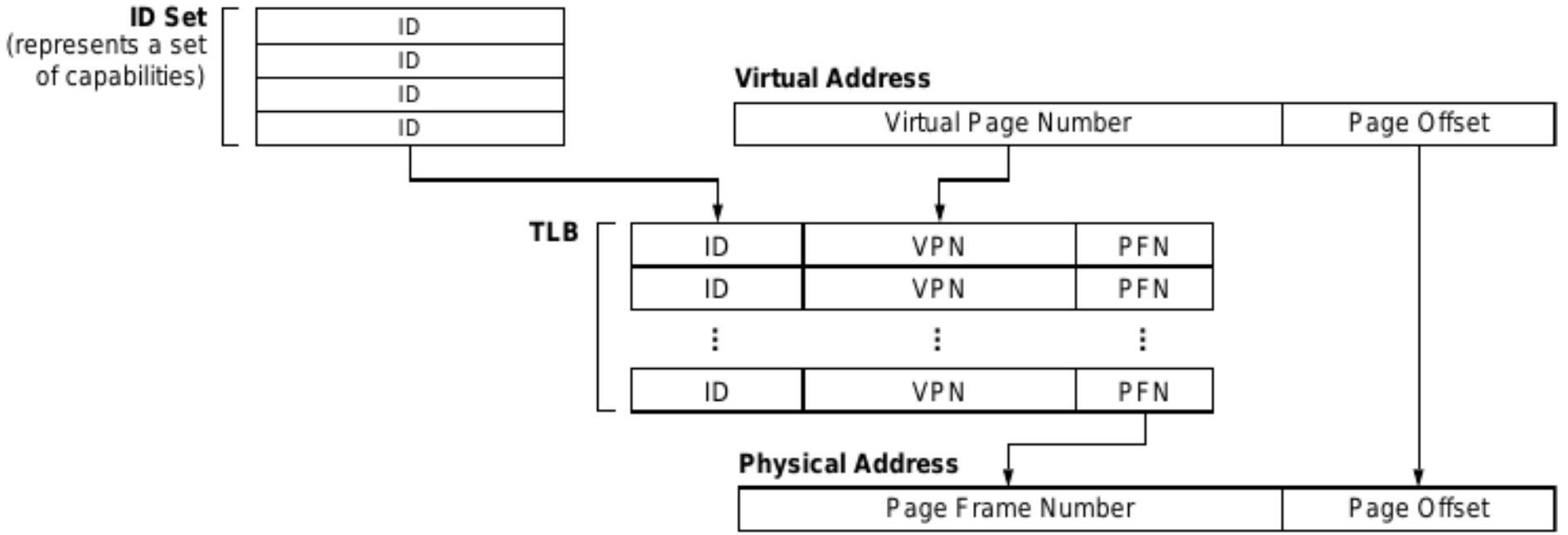
Single owner, no ID



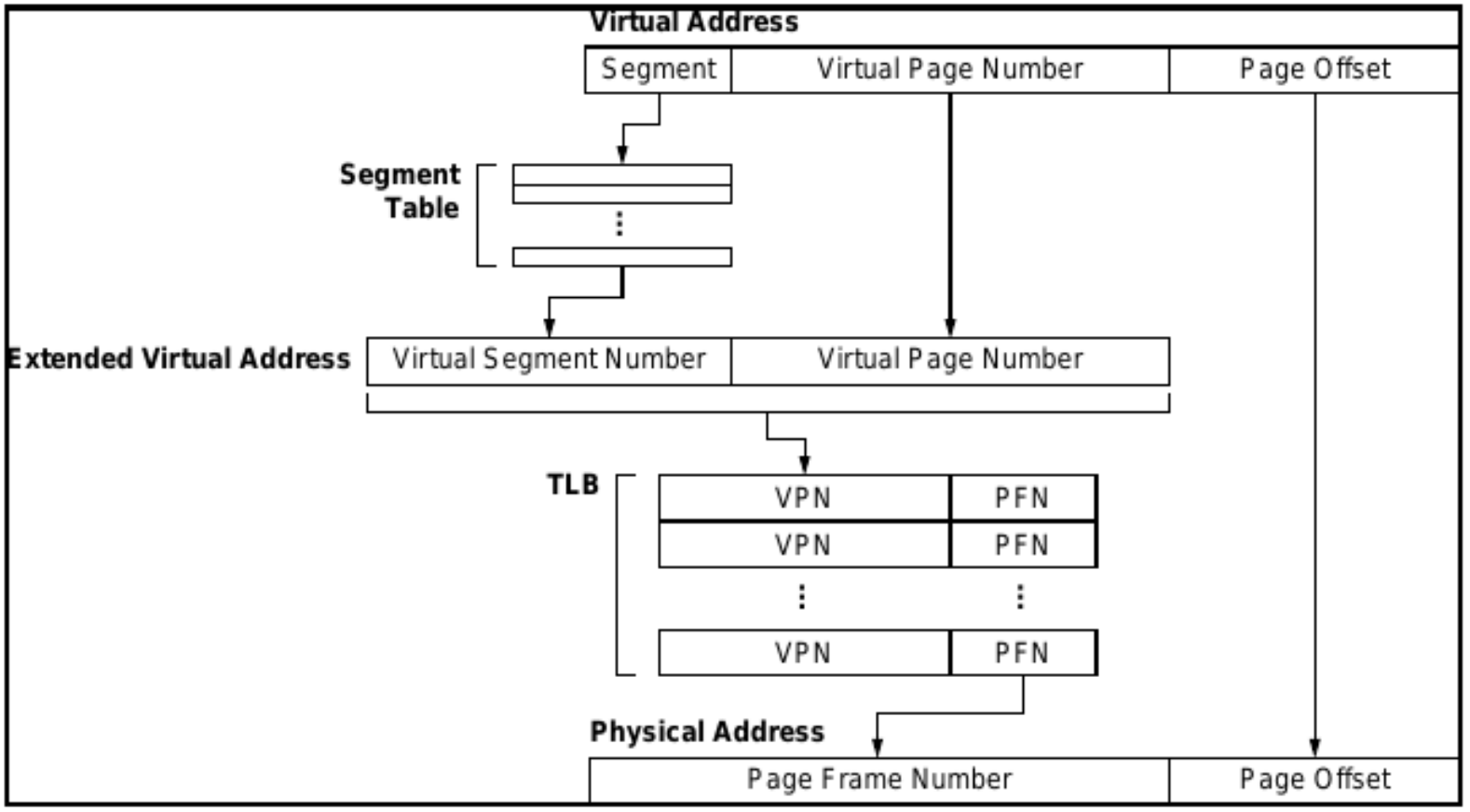
Single owner, single ID



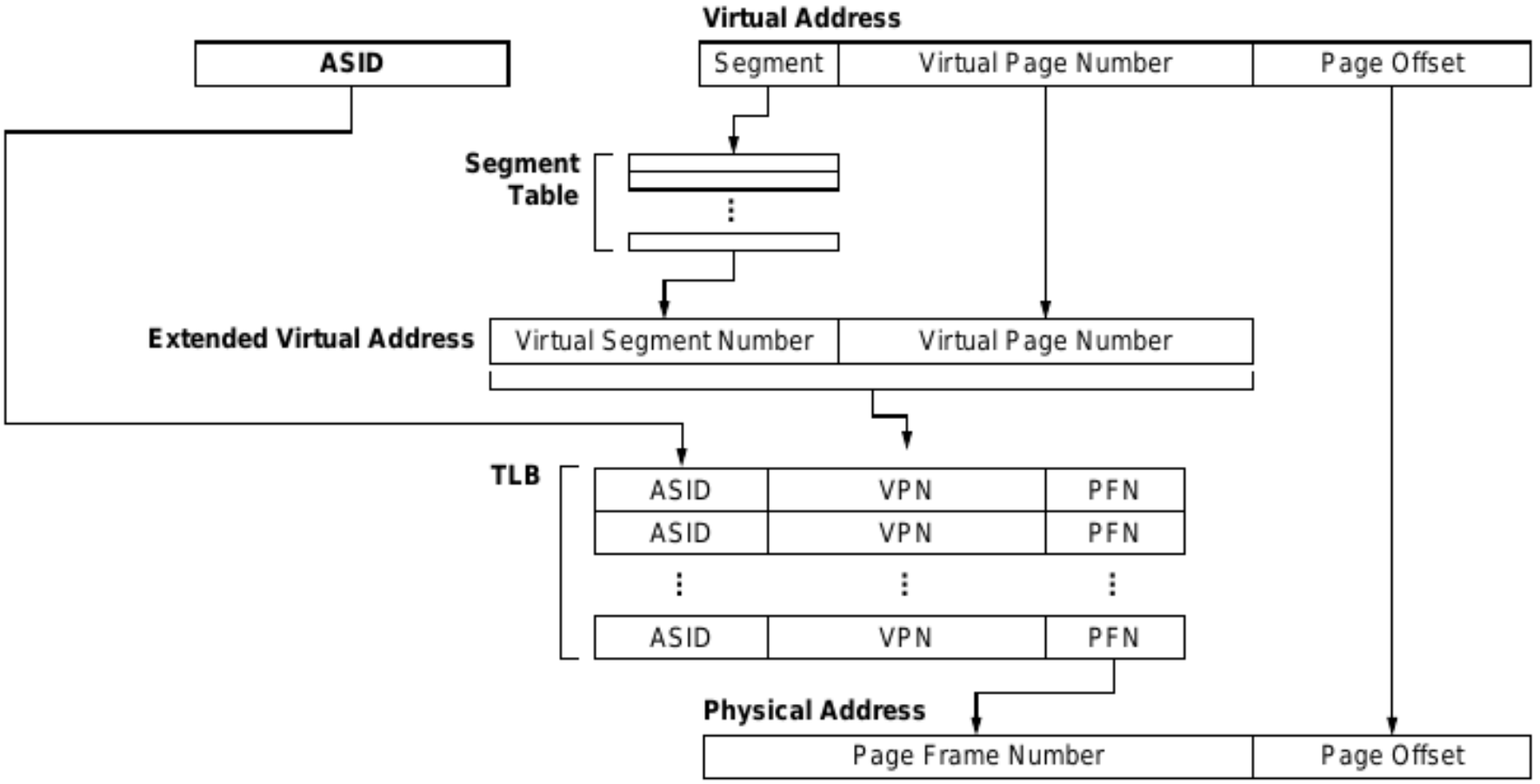
Single owner, multiple ID



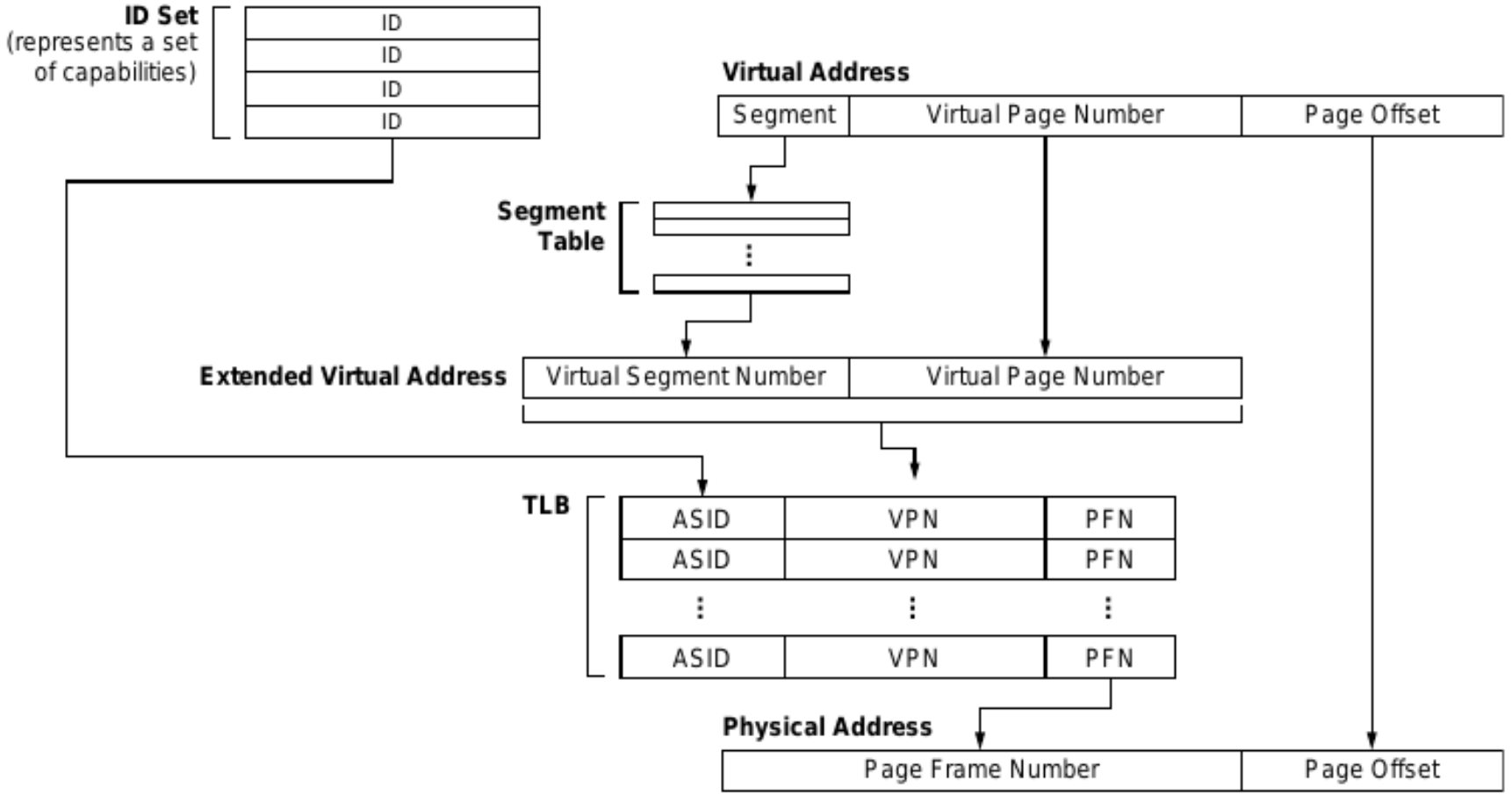
Multiple owner, no id

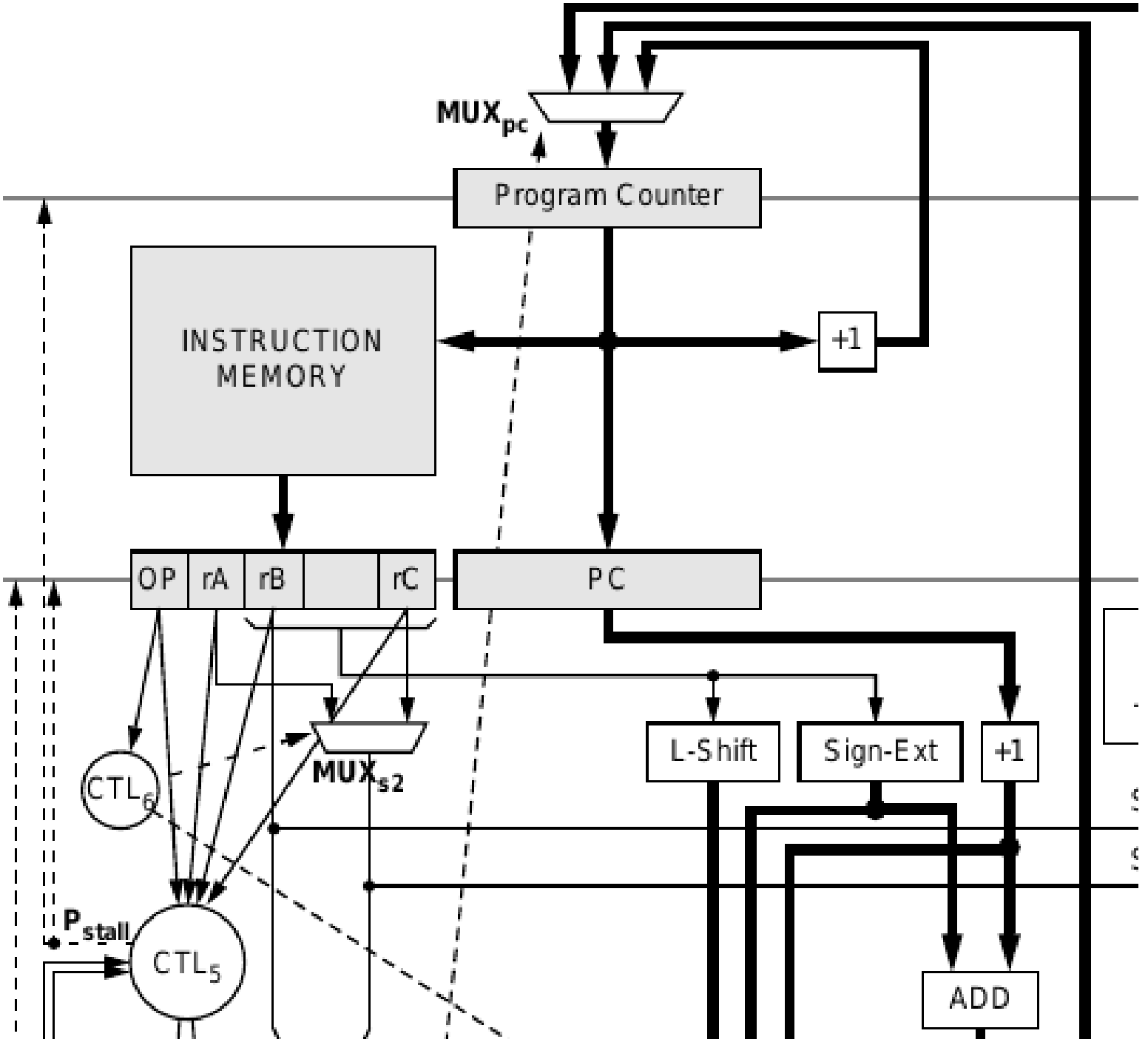


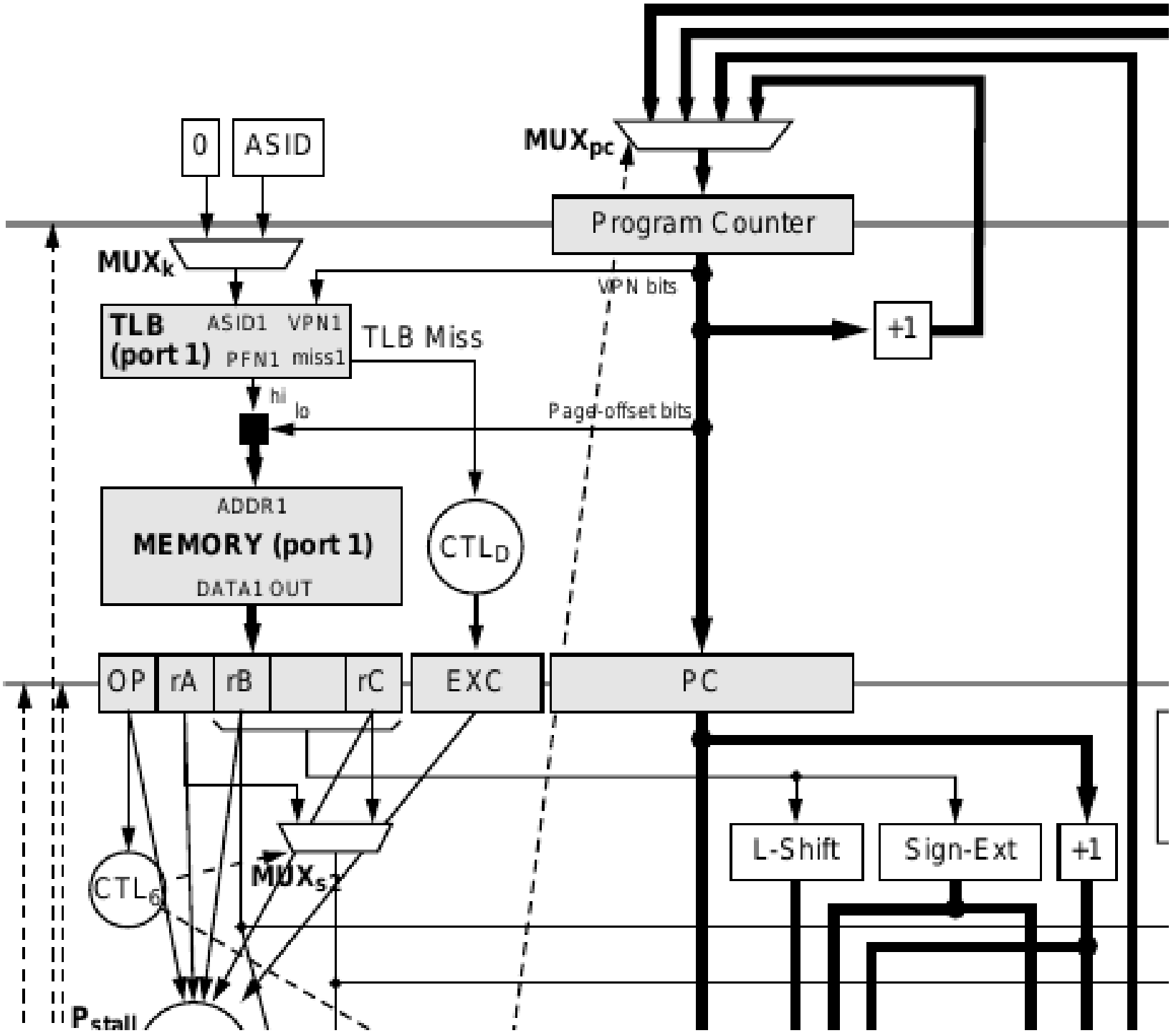
Multiple owner, single ID

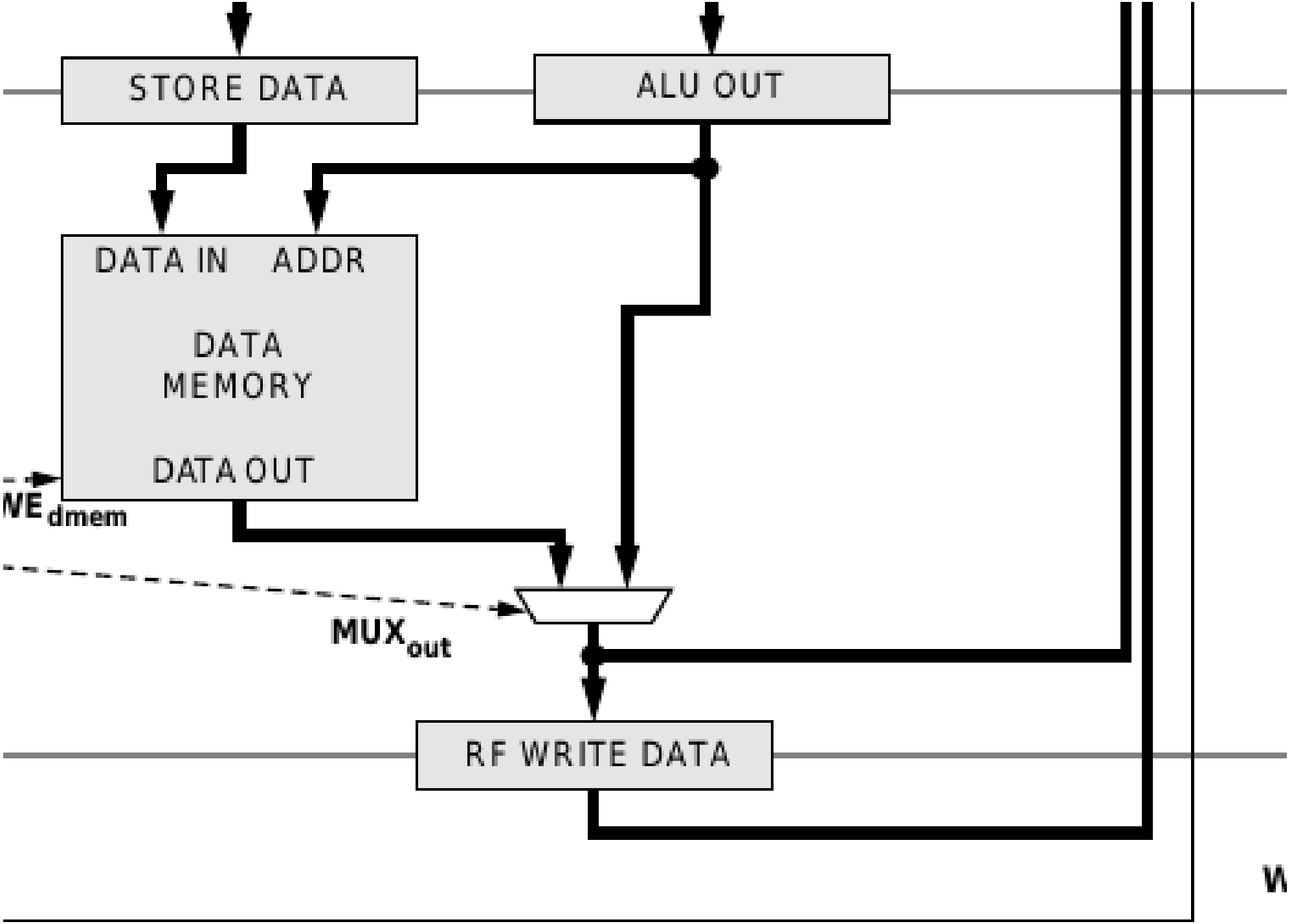


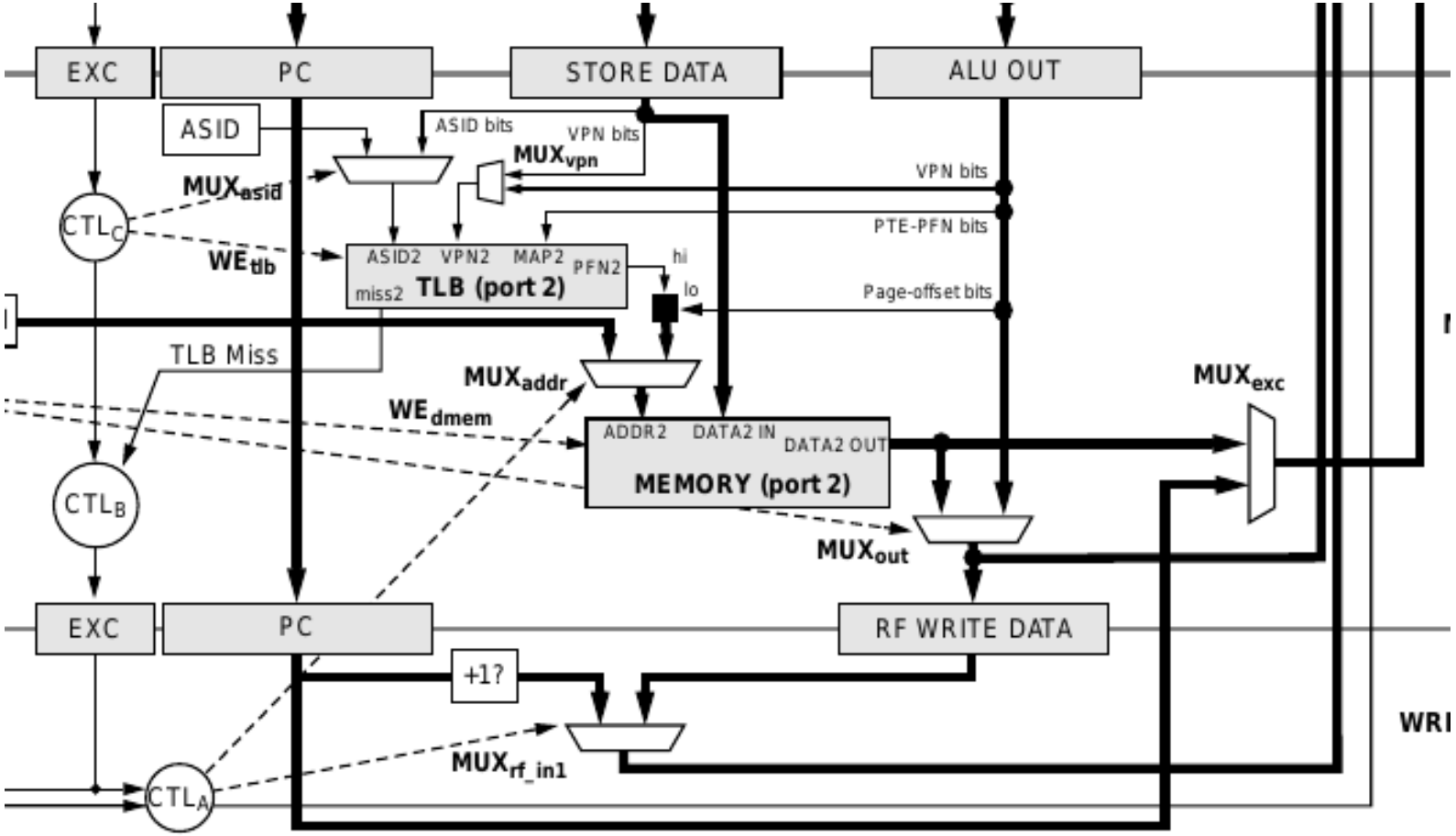
Multiple owner, multiple ID




MUX_pc
Program Counter
INSTRUCTION MEMORY
+1
OP
rA
rB
rC
PC
MUX_s2
CTL_6
L-Shift
Sign-Ext
+1
P_stall
CTL_5
ADD


0
ASID
MUXpc
Program Counter
MUXk
VPN bits
TLB (port 1)
ASID1 VPN1
PFN1 miss1
TLB Miss
+1
hi
lo
Page-offset bits
ADDR1
MEMORY (port 1)
DATA1 OUT
CTLD
OP
rA
rB
rC
EXC
PC
L-Shift
Sign-Ext
+1
CTL6
MUXs
Pstall


STORE DATA
ALU OUT
DATA IN
ADDR
DATA
MEMORY
DATA OUT
WE dmem
MUX out
RF WRITE DATA


EXC
PC
STORE DATA
ALU OUT
ASID
ASID bits
VPN bits
MUX_vpn
MUX_asid
VPN bits
CTL_C
WE_tlb
PTE-PFN bits
ASID2 VPN2 MAP2 PFN2
hi
lo
miss2
TLB (port 2)
Page-offset bits
TLB Miss
MUX_addr
MUX_exc
WE_dmem
ADDR2 DATA2 IN DATA2 OUT
MEMORY (port 2)
CTL_B
MUX_out
EXC
PC
RF WRITE DATA
+1?
MUX_rf_in1
CTL_A

questions?