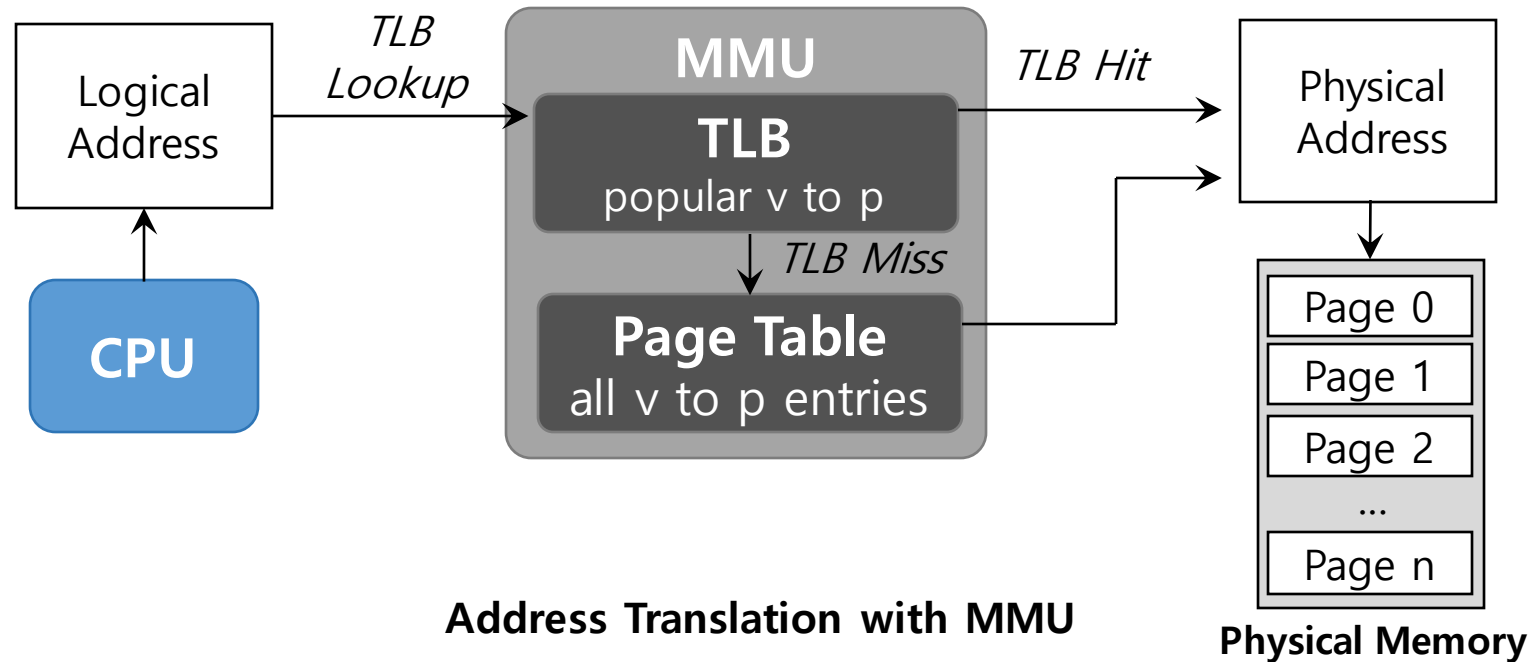


Translation Lookaside Buffer

Operating System (OS)
안인규

TLB

- Part of the chip's **memory-management unit(MMU)**
- A hardware cache of **popular** virtual-to-physical address translation



TLB Basic Algorithms

```
1: VPN = (VirtualAddress & VPN_MASK ) >> SHIFT
2: (Success , TlbEntry) = TLB_Lookup(VPN)
3:     if(Success == TRUE) { // TLB Hit
4:         if(CanAccess(TlbEntry.ProtectBit) == True ) {
5:             offset = VirtualAddress & OFFSET_MASK
6:             PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7:             AccessMemory( PhysAddr )
8:         }else RaiseException(PROTECTION_ERROR)
```

TLB Basic Algorithms (Cont.)

```
11:     }else{ //TLB Miss
12:         PTEAddr = PTBR + (VPN * sizeof(PTE))
13:         PTE = AccessMemory(PTEAddr)
14:         if(PTE.Valid == False)
15:             RaiseException(SEGFAULT) ;
16:         else{
17:             TLB_Insert( VPN , PTE.PFN , PTE.ProtectBits)
18:             RetryInstruction()
19:         }
```

- (11-12 lines) The hardware accesses the page table to find the translation
- (16 lines) updates the TLB with the translation

Example: Accessing An Array

- How a TLB can improve its performance
 - An array of 10 4-byte integers
 - 8-bit virtual address space with 16-byte pages of the system

Example: Accessing An Array

- How a TLB can improve its performance

	OFFSET				
	00	04	08	12	16
VPN = 00					
VPN = 01					
VPN = 03					
VPN = 04					
VPN = 05					
VPN = 06		a[0]	a[1]	a[2]	
VPN = 07	a[3]	a[4]	a[5]	a[6]	
VPN = 08	a[7]	a[8]	a[9]		
VPN = 09					
VPN = 10					
VPN = 11					
VPN = 12					
VPN = 13					
VPN = 14					
VPN = 15					

```
0:   int sum = 0 ;
1:   for( i=0; i<10; i++) {
2:       sum+=a[i];
3:   }
```

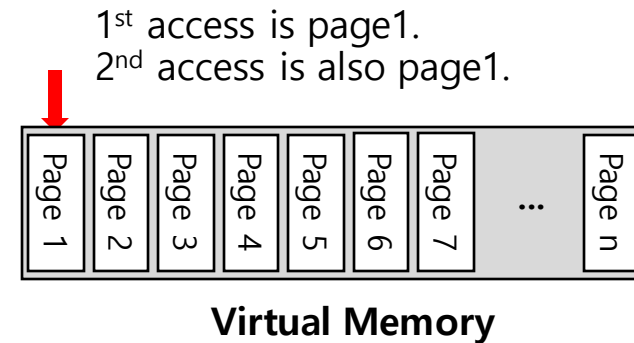
The TLB improves performance
due to **spatial locality**

3 misses and 7 hits.
Thus **TLB hit rate** is 70%.

Locality

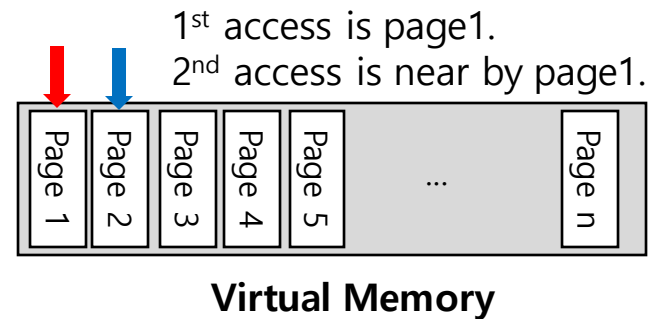
- Temporal Locality

- An instruction or data item that has been recently accessed will likely be re-accessed soon in the future



- Spatial Locality

- If a program accesses memory at address x , it will likely soon access memory near x



Who Handles The TLB Miss?

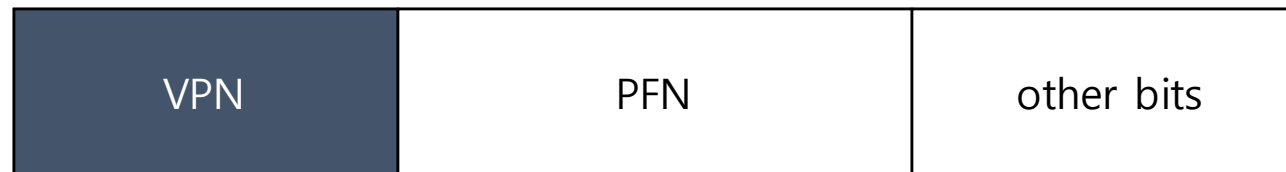
- Hardware handles the TLB miss entirely on **CISC**
 - The hardware has to know exactly where the page tables are located in memory
 - The hardware would “walk” the page table, find the correct page-table entry and **extract** the desired translation, **update** and **retry** instruction
 - **hardware-managed TLB**
 - **Intel x86**
- **RISC** has what is known as a **software-managed TLB**
 - On a TLB miss, the hardware raises exception (trap handler)
 - **Trap handler is code** within the OS that is written with the express purpose of **handling TLB miss**

TLB Control Flow algorithm(OS Handled)

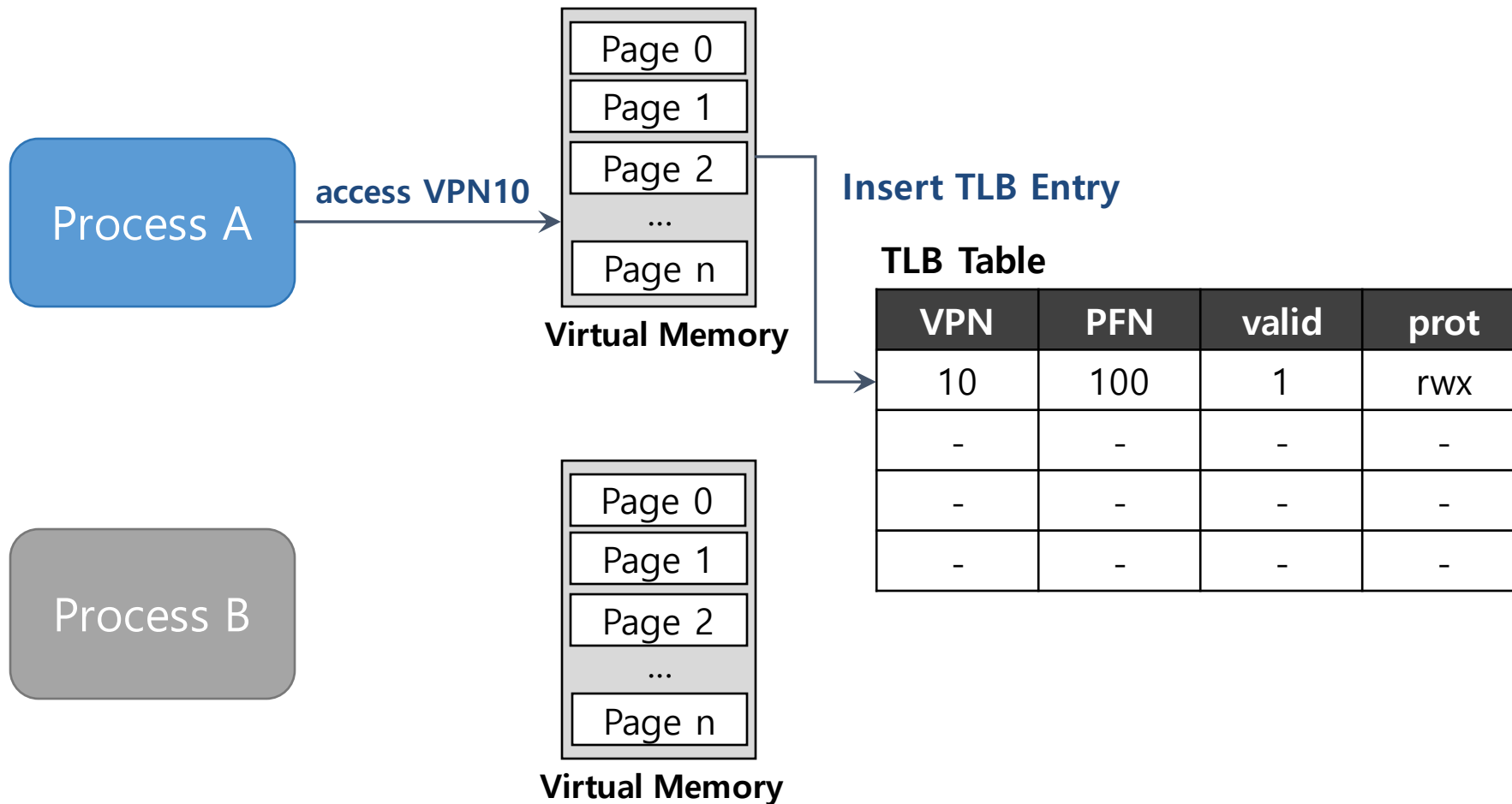
```
1:     VPN = (VirtualAddress & VPN_MASK) >> SHIFT
2:     (Success, TlbEntry) = TLB_Lookup(VPN)
3:     if (Success == True) // TLB Hit
4:         if (CanAccess(TlbEntry.ProtectBits) == True)
5:             Offset = VirtualAddress & OFFSET_MASK
6:             PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7:             Register = AccessMemory(PhysAddr)
8:         else
9:             RaiseException(PROTECTION_FAULT)
10:    else // TLB Miss
11:        RaiseException(TLB_MISS)
```

TLB entry

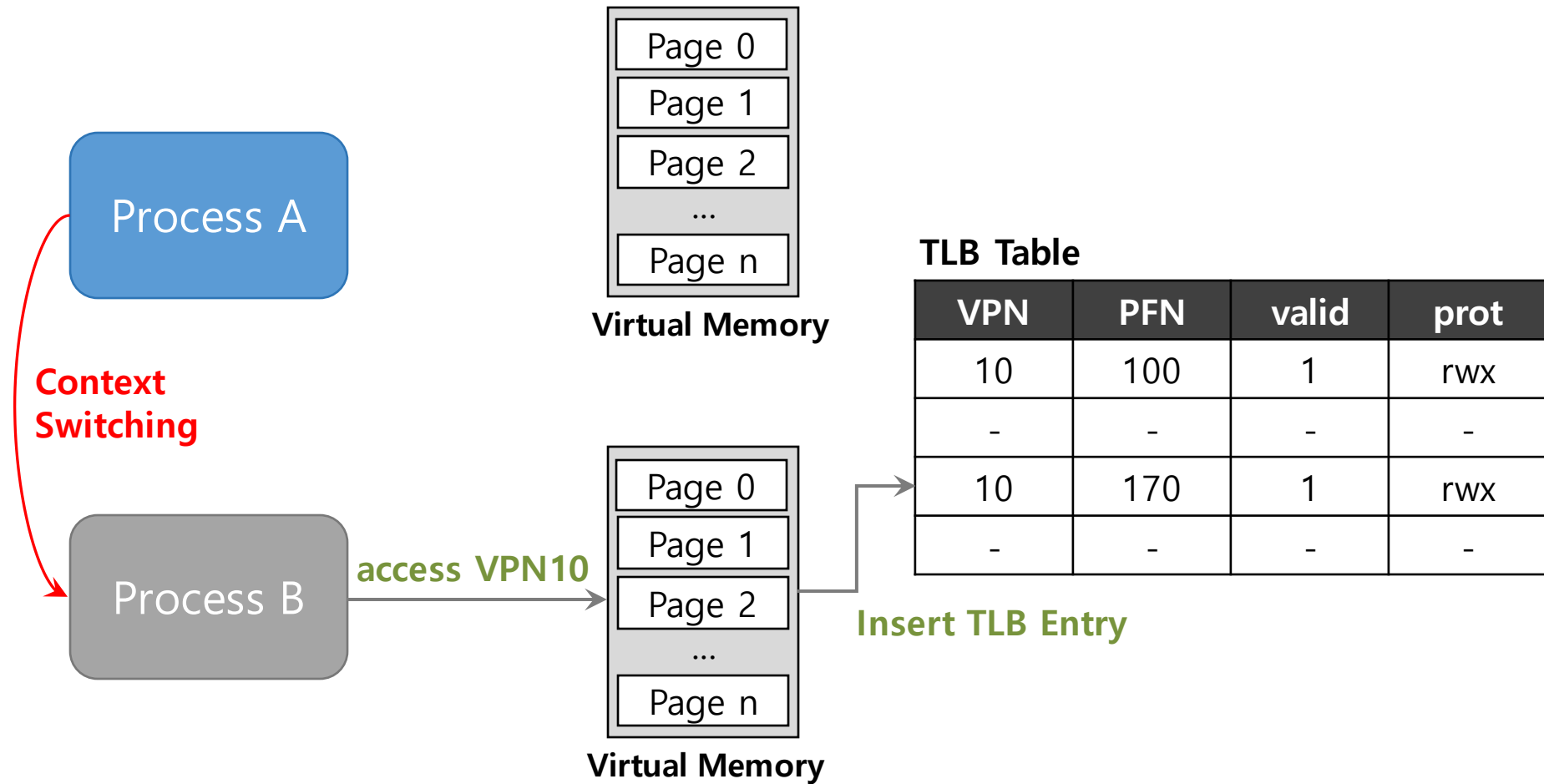
- TLB is managed by **Full Associative** method.
 - A typical TLB has 32,64, or 128 entries.
 - Hardware searches the entire TLB in parallel to find the desired translation.
 - other bits: valid bits , protection bits, address-space identifier, dirty bit



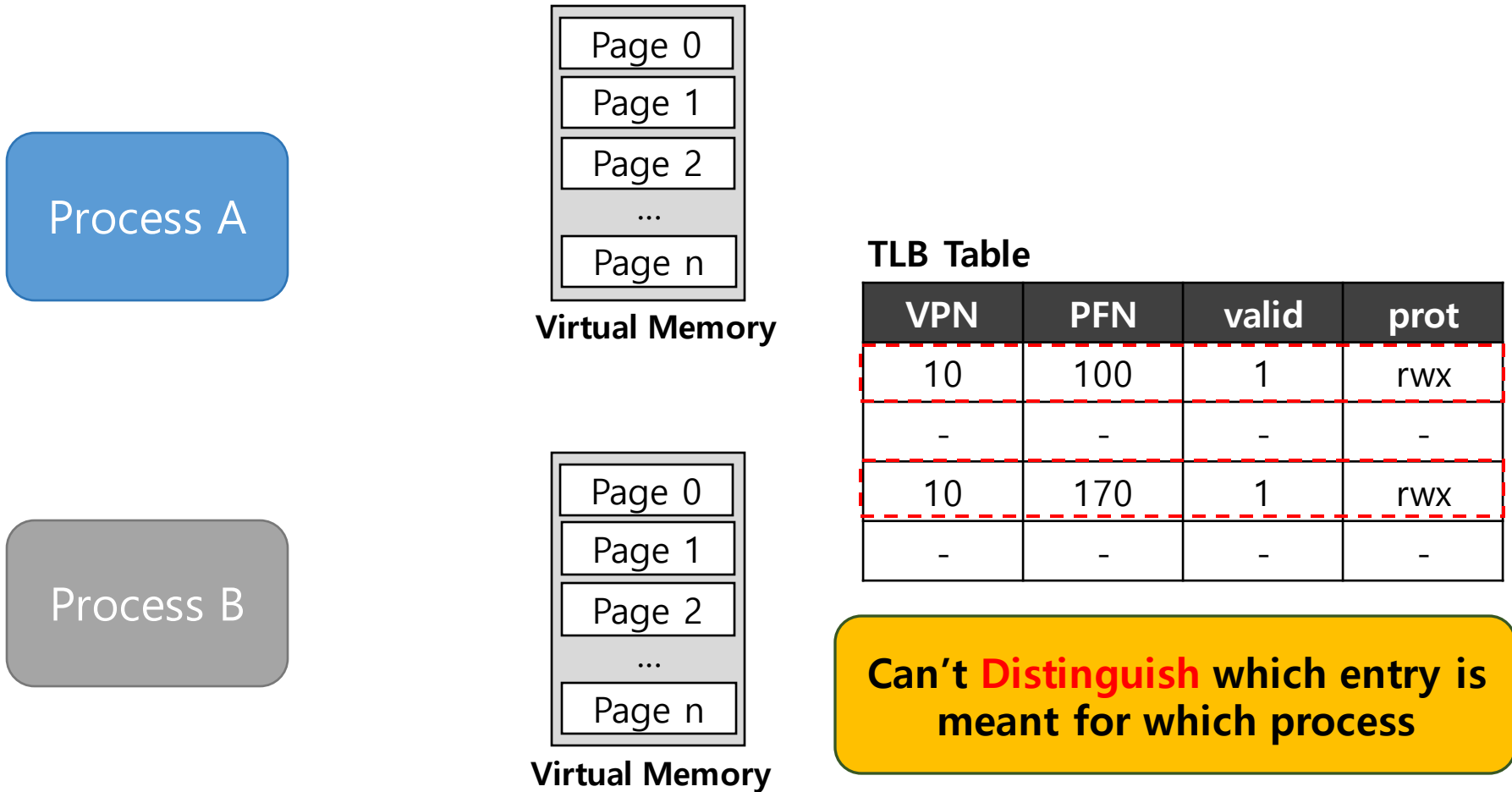
TLB Issue: Context Switching



TLB Issue: Context Switching

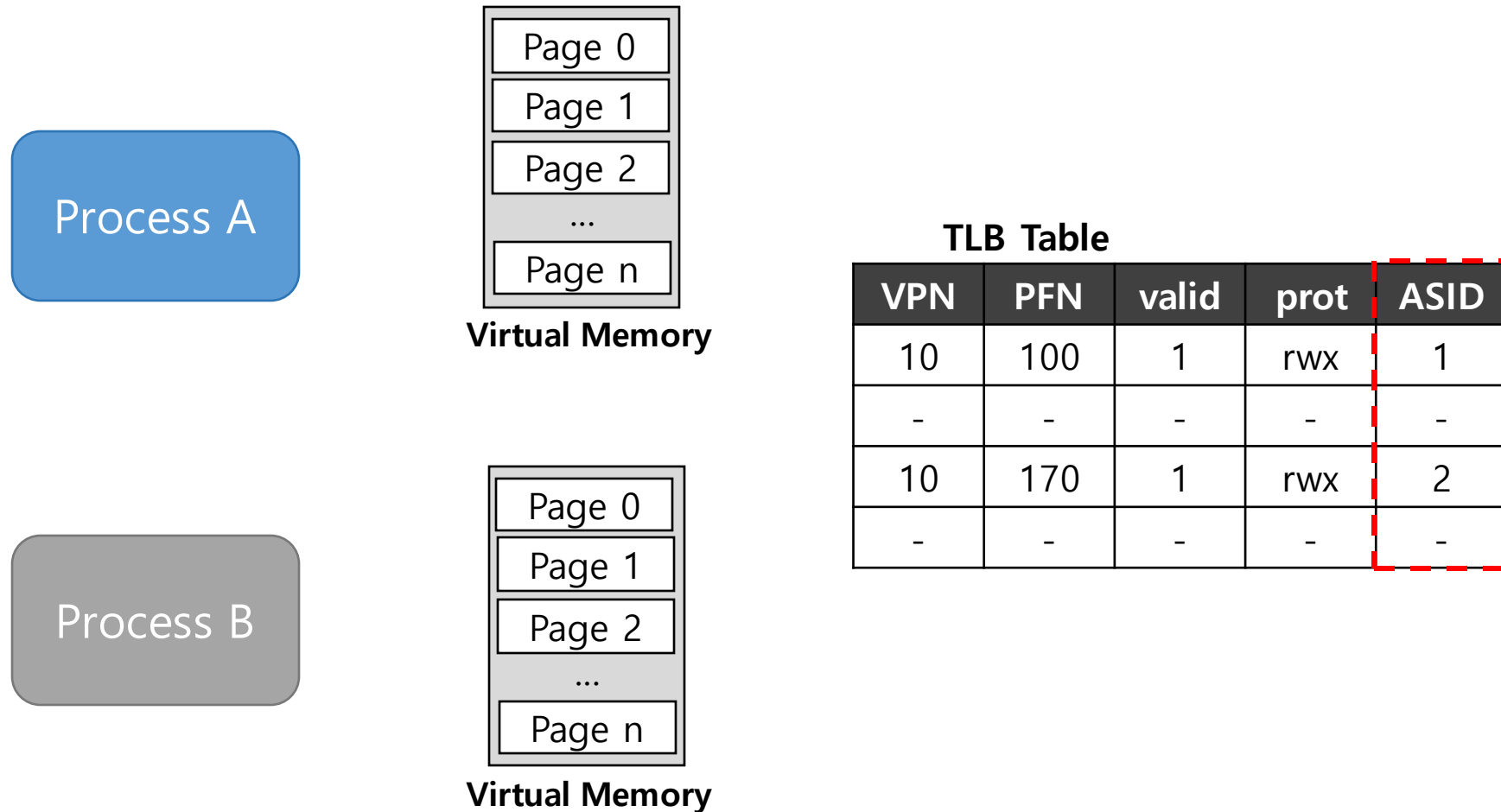


TLB Issue: Context Switching



To Solve Problem

- Provide an address space identifier(ASID) field in the TLB.



Another Case

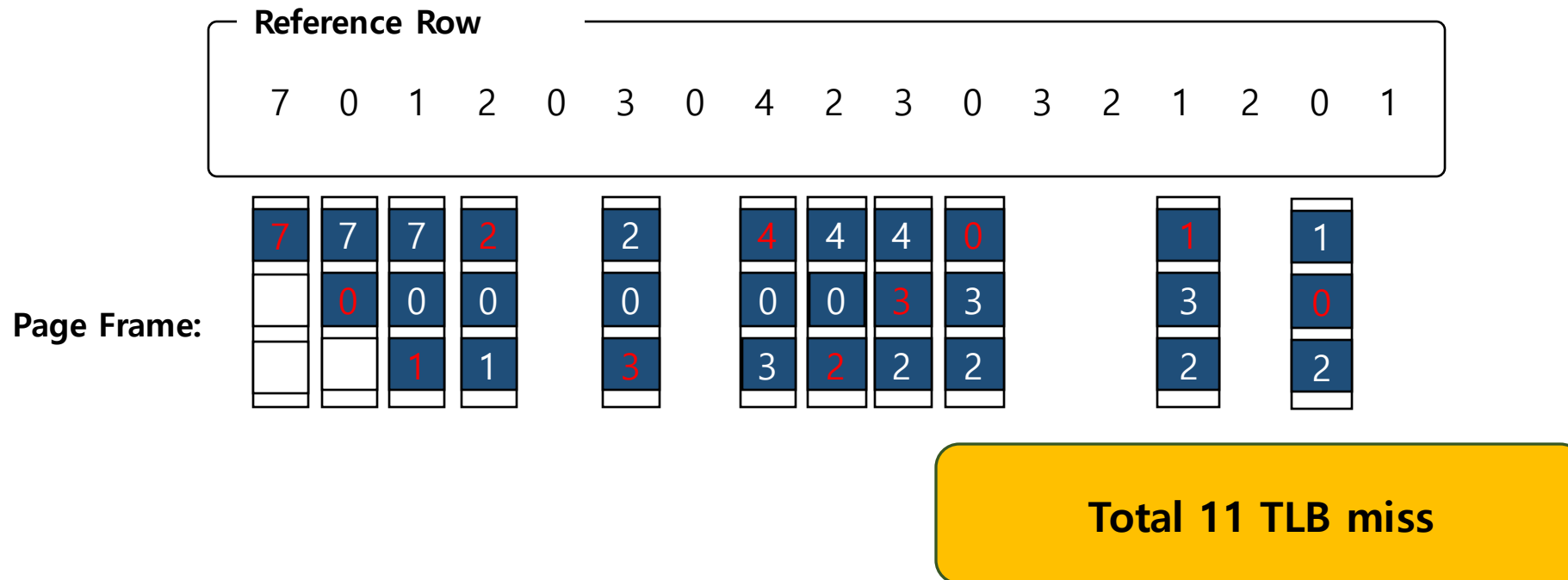
- Two processes **share a page**
 - Process 1 is sharing physical page 101 with Process2.
 - P1 maps this page into the 10th page of its address space.
 - P2 maps this page to the 50th page of its address space.

VPN	PFN	valid	prot	ASID
10	101	1	rwX	1
-	-	-	-	-
50	101	1	rwX	2
-	-	-	-	-

Sharing of pages is **useful** as it reduces the number of physical pages in use.

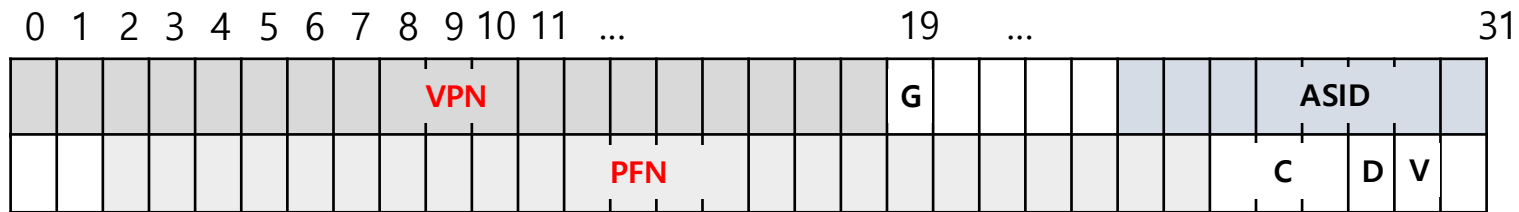
TLB Replacement Policy

- LRU(Least Recently Used)
 - Evict an entry that has not recently been used.
 - Take advantage of *locality* in the memory-reference stream.



A Real TLB Entry

All 64 bits of this TLB entry(example of MIPS R4000)



Flag	Content
19-bit VPN	The rest reserved for the kernel.
24-bit PFN	Systems can support with up to 64GB of main memory(pages).
Global bit(G)	Used for pages that are globally-shared among processes.
ASID	OS can use to distinguish between address spaces.
Coherence bit(C)	determine how a page is cached by the hardware.
Dirty bit(D)	marking when the page has been written.
Valid bit(V)	tells the hardware if there is a valid translation present in the entry.