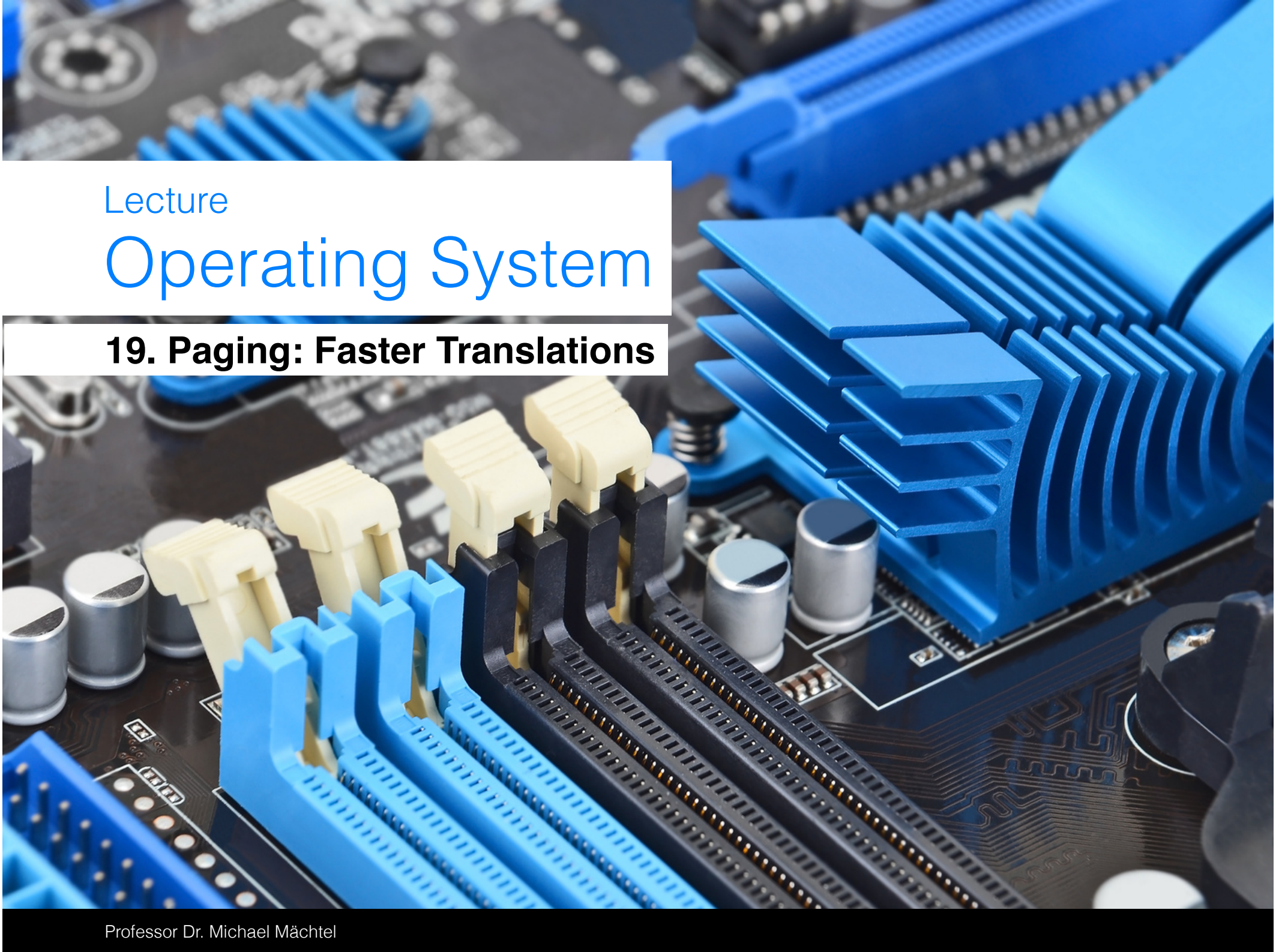


Lecture

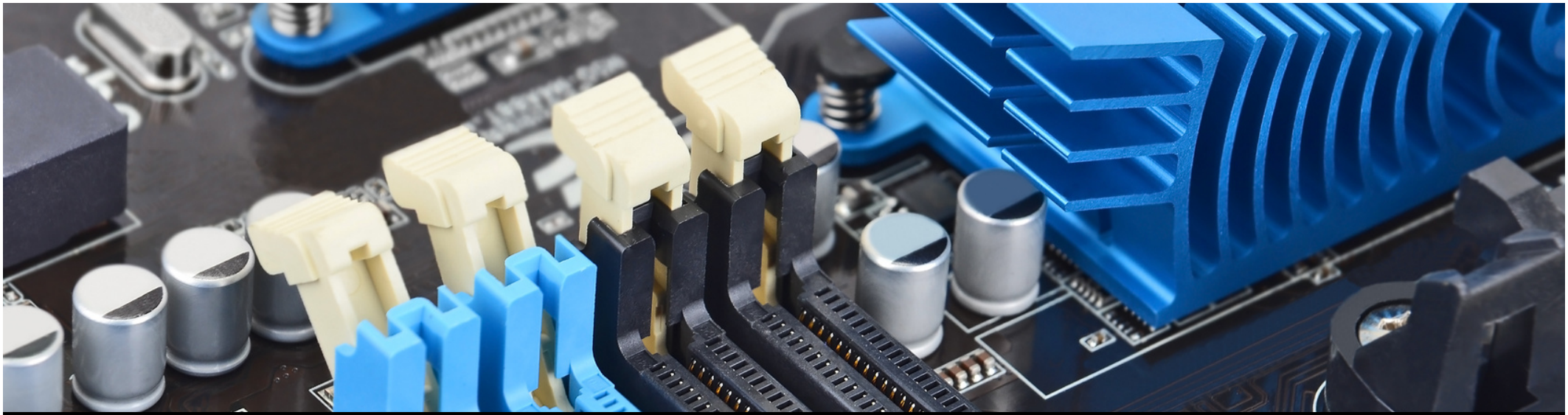
Operating System

19. Paging: Faster Translations

Professor Dr. Michael Mächtel

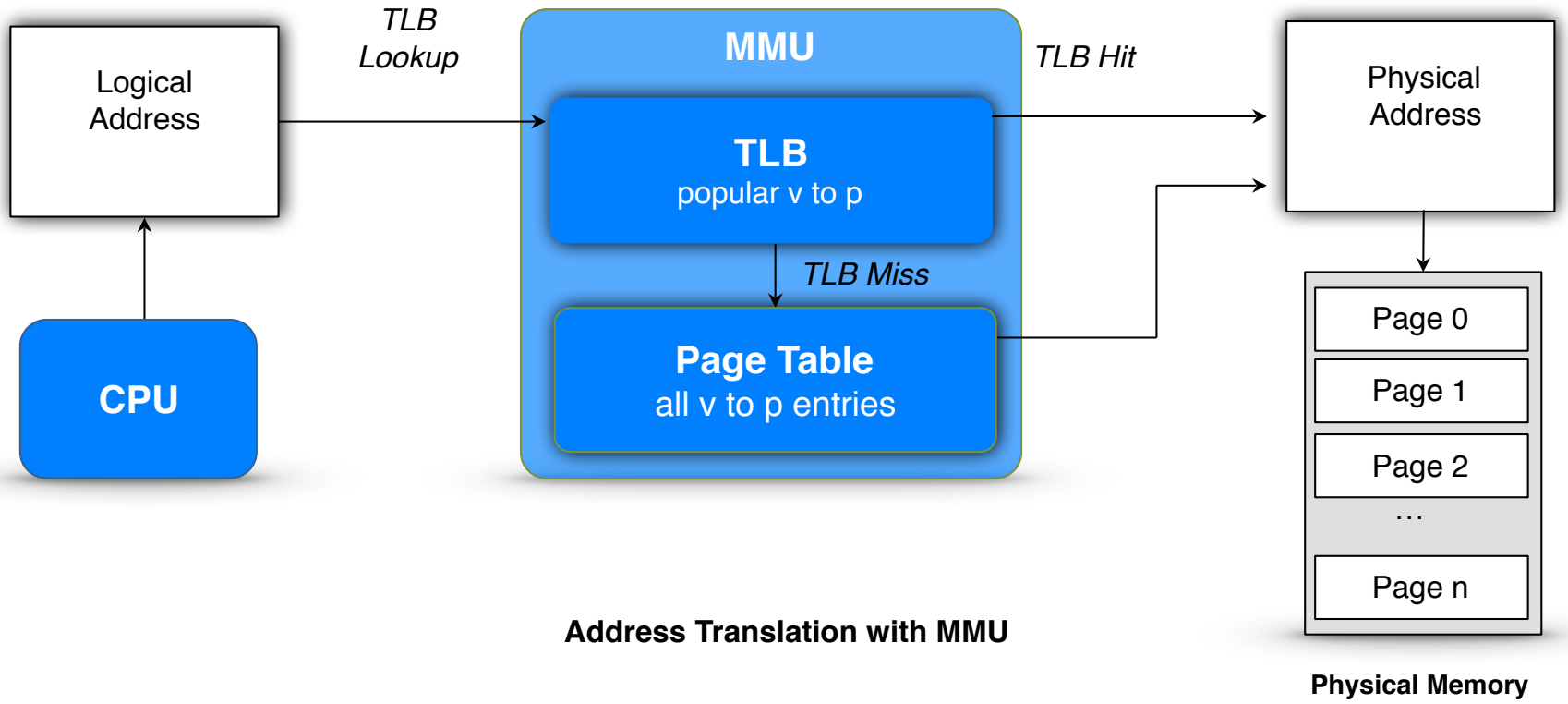
19. Paging: Faster Translations

1. **How can we speed up address translation, and generally avoid the extra memory reference that paging seems to require?**
2. **What hardware support is required?**
3. **What OS involvement is needed?**



TLB: Translation Lookaside Buffer

- 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.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
11      PTEAddr = PTBR + (VPN * sizeof(PTE))
12      PTE = AccessMemory(PTEAddr)
13      if (PTE.Valid == False)
14          RaiseException(SEGMENTATION_FAULT)
15      else if (CanAccess(PTE.ProtectBits) == False)
16          RaiseException(PROTECTION_FAULT)
17      else
18          TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
19          RetryInstruction()
```

extract the virtual page number(VPN).

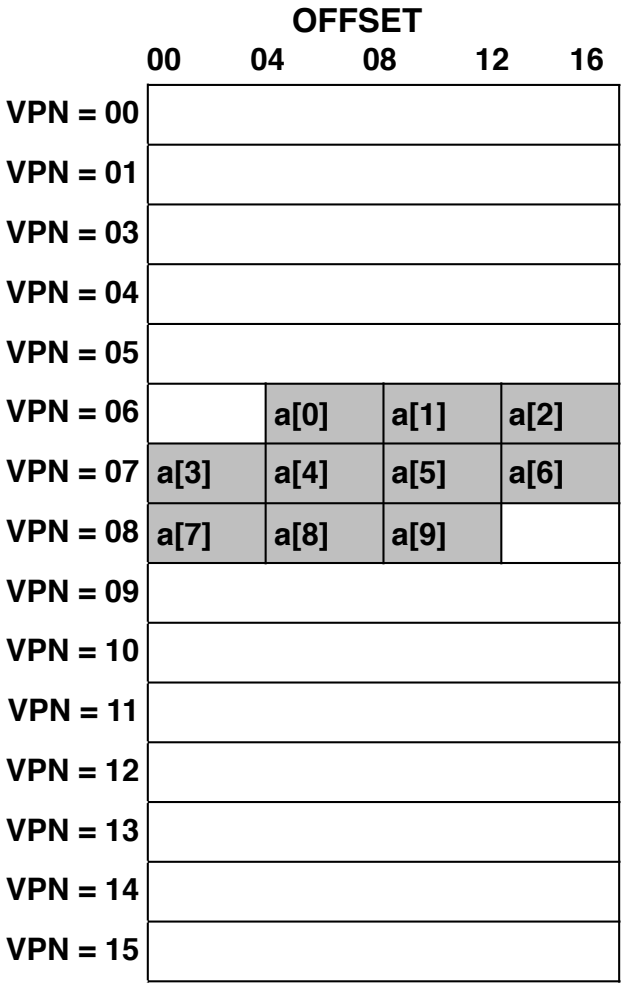
check if the TLB holds the transalation for this VPN.

extract the page frame number from the relevant TLB entry, and form the desired physical address and access memory.

The hardware accesses the page table to find the translation.

updates the TLB with the translation.

Example: Accessing An Array



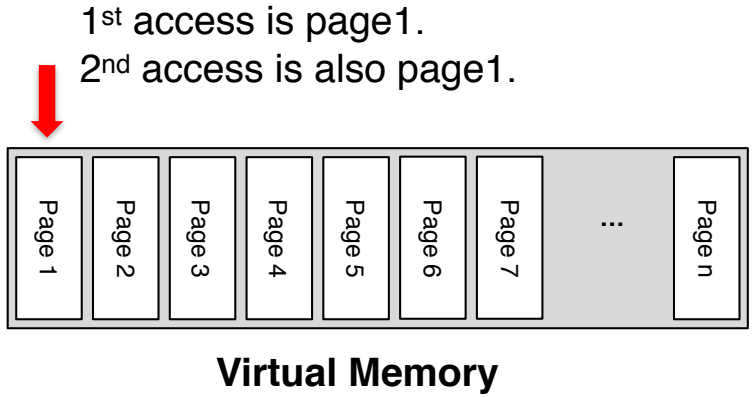

```
0:  int sum = 0 ;
1:  for( i=0; i<10; i++){
2:      sum+=a[i];
3:  }
```

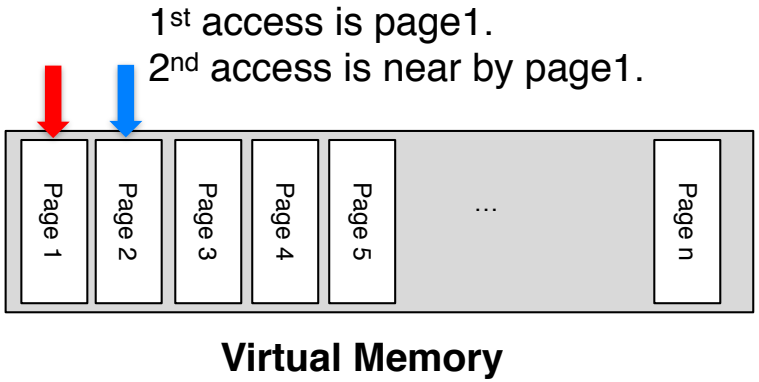
The TLB improves performance due to spatial locality

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

Locality

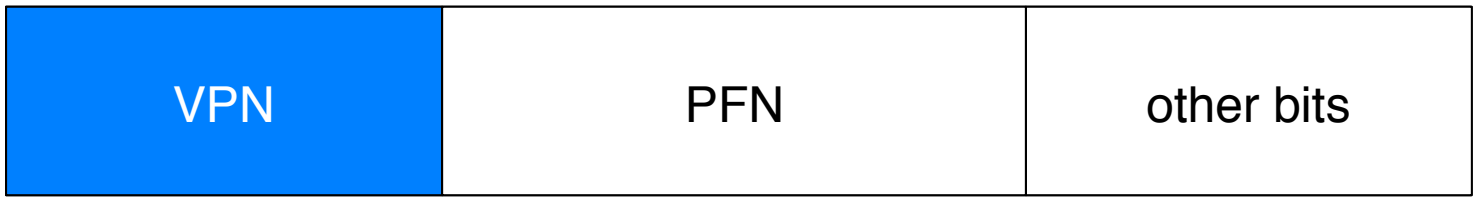
- **Temporal Locality**
  - An instruction or data item that has been recently accessed will likely be re-accessed soon in the future.
- **Spatial Locality**
  - If a program accesses memory at address x, it will likely soon access memory near x.





TLB entry

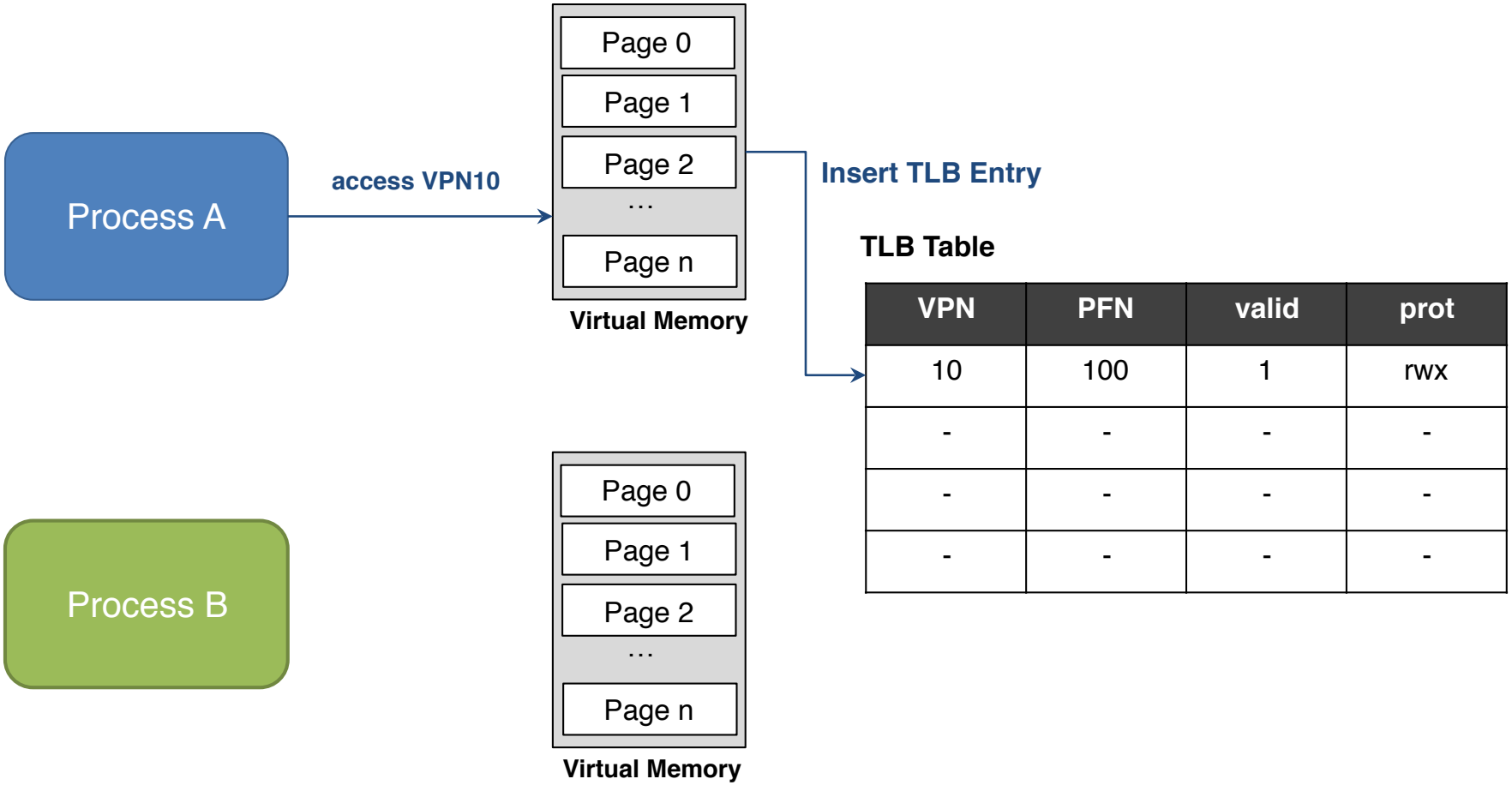
- TLB is managed by **Full Associative** method.
  - A typical TLB might have 32, 64, or 128 entries.
  - Hardware search the entire TLB in **parallel** to find the desired translation.
  - other bits: valid bits[1], protection bits, address-space identifier, dirty bit



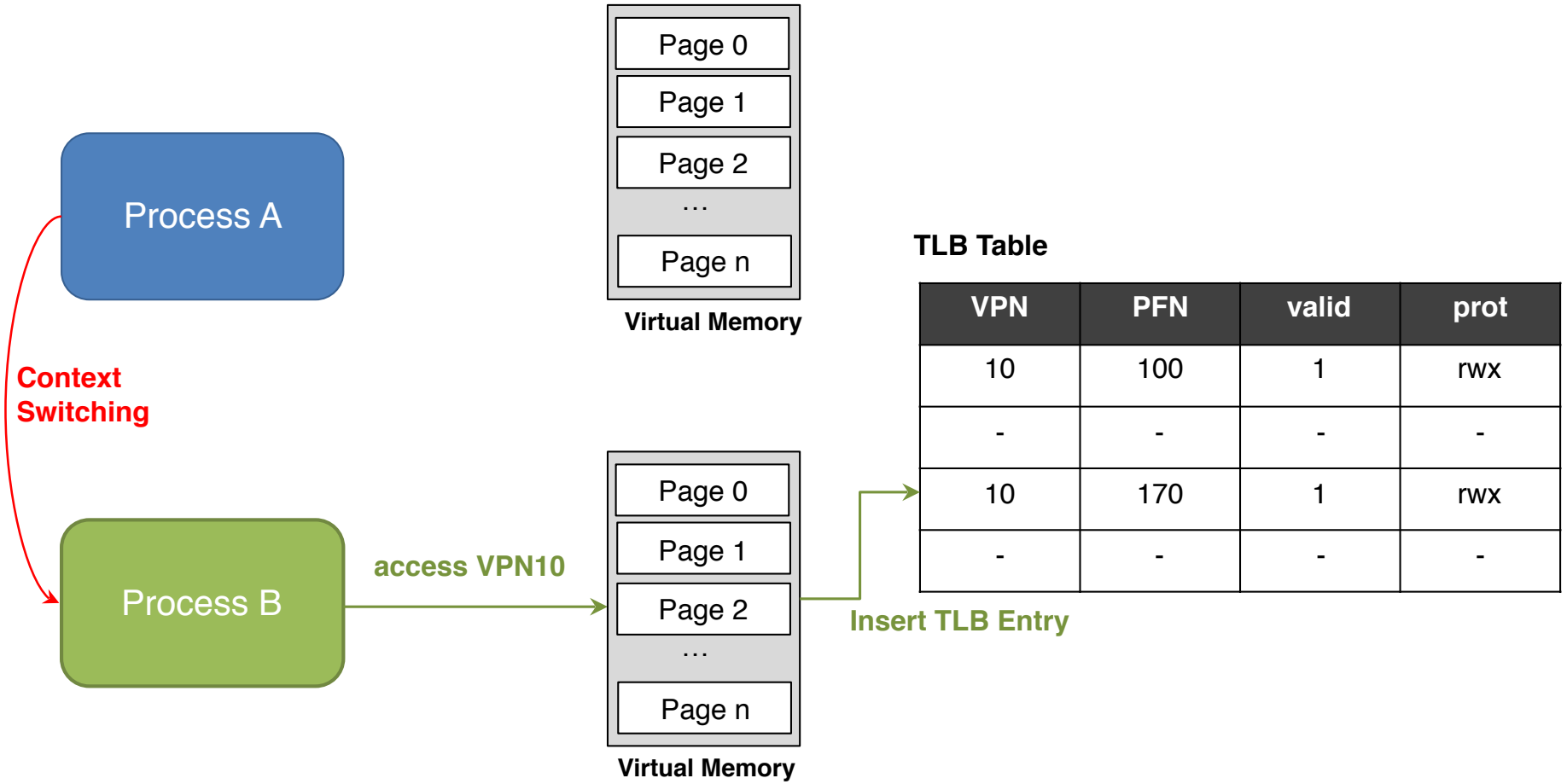
Typical TLB entry look like this

1: valid has in TLB different meaning than in Page Table!

TLB Issue: Context Switching



TLB Issue: Context Switching

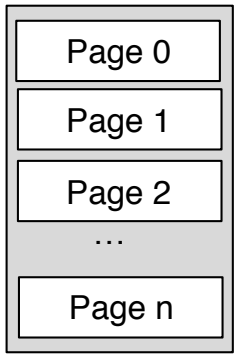


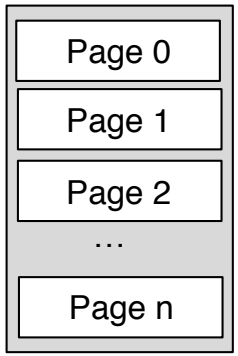
VPN	PFN	valid	prot
10	100	1	rwx
-	-	-	-
10	170	1	rwx
-	-	-	-

TLB Issue: Context Switching

Process A

Process B





TLB Table

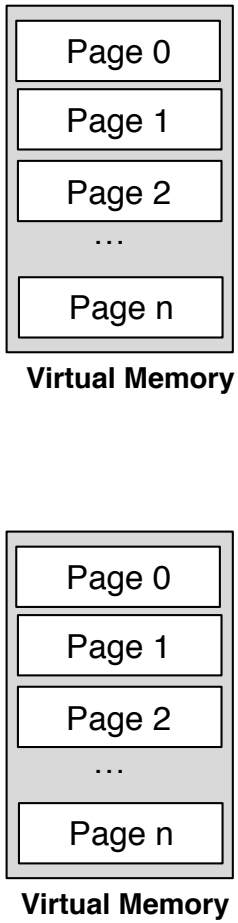
VPN	PFN	valid	prot
10	100	1	rwx
-	-	-	-
10	170	1	rwx
-	-	-	-

Can't Distinguish which entry is meant for which process

To Solve Problem

Process A

Process B



TLB Table

VPN	PFN	valid	prot	ASID
10	100	1	rwx	1
-	-	-	-	-
10	170	1	rwx	2
-	-	-	-	-

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

Another Case

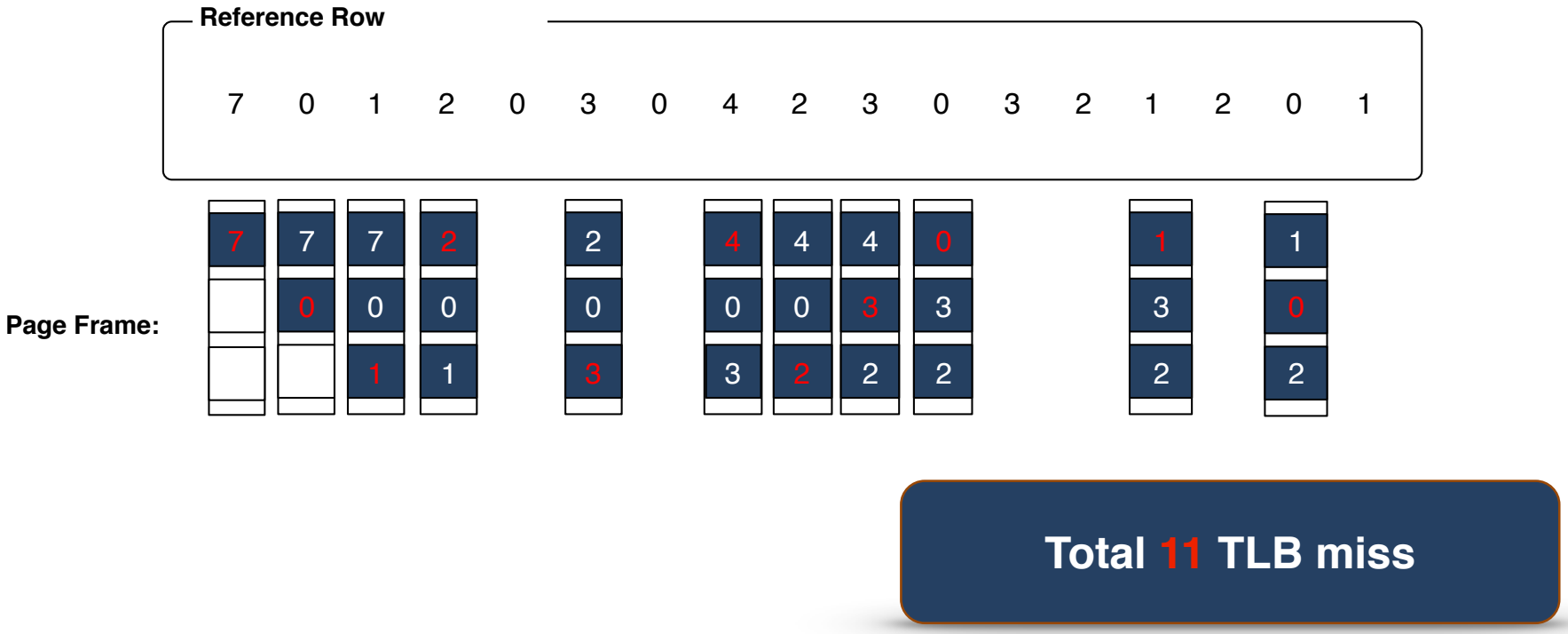
- Two processes **share a page**.
  - Process 1 is sharing physical page 101 with Process2.
  - P1 maps this page into the 10th page of its address space.
  - P2 maps this page to the 50th page of its address space.

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

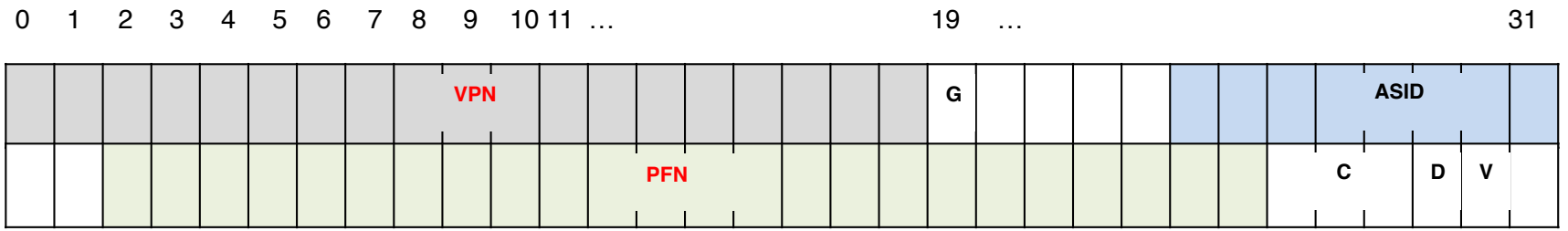
Sharing of pages is useful as it reduces the number of physical pages in use.

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.



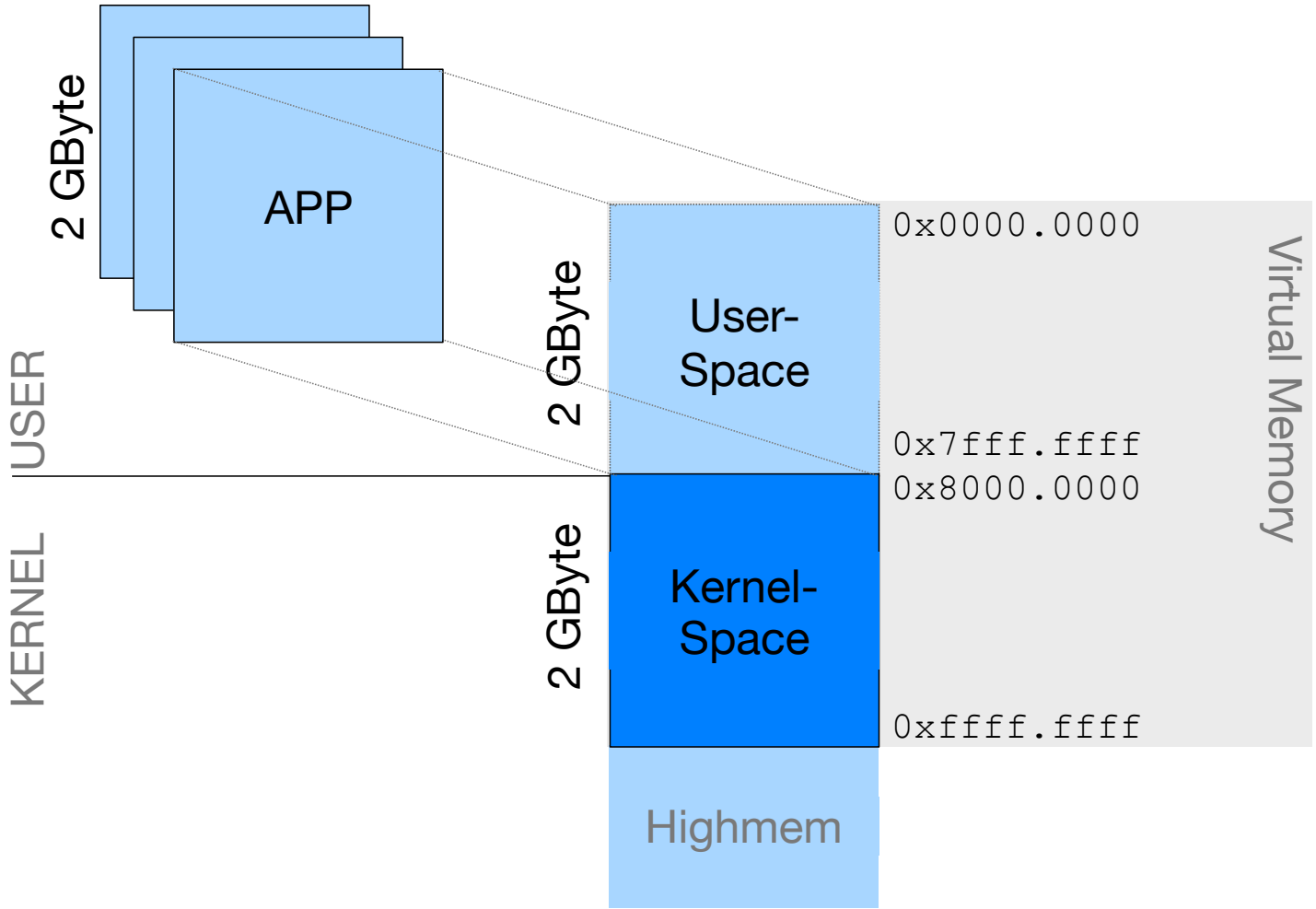
A Real TLB Entry (64 bits MIPS)



Flag	Content
19-bit VPN	The rest reserved for the kernel.
24-bit PFN	Systems can support with up to 64GB of main memory (2^{24} * 4KB 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.

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

VPNs: Shared between User and Kernel Adress



Example: 64 Bit Linux System

cat /proc/8080/maps

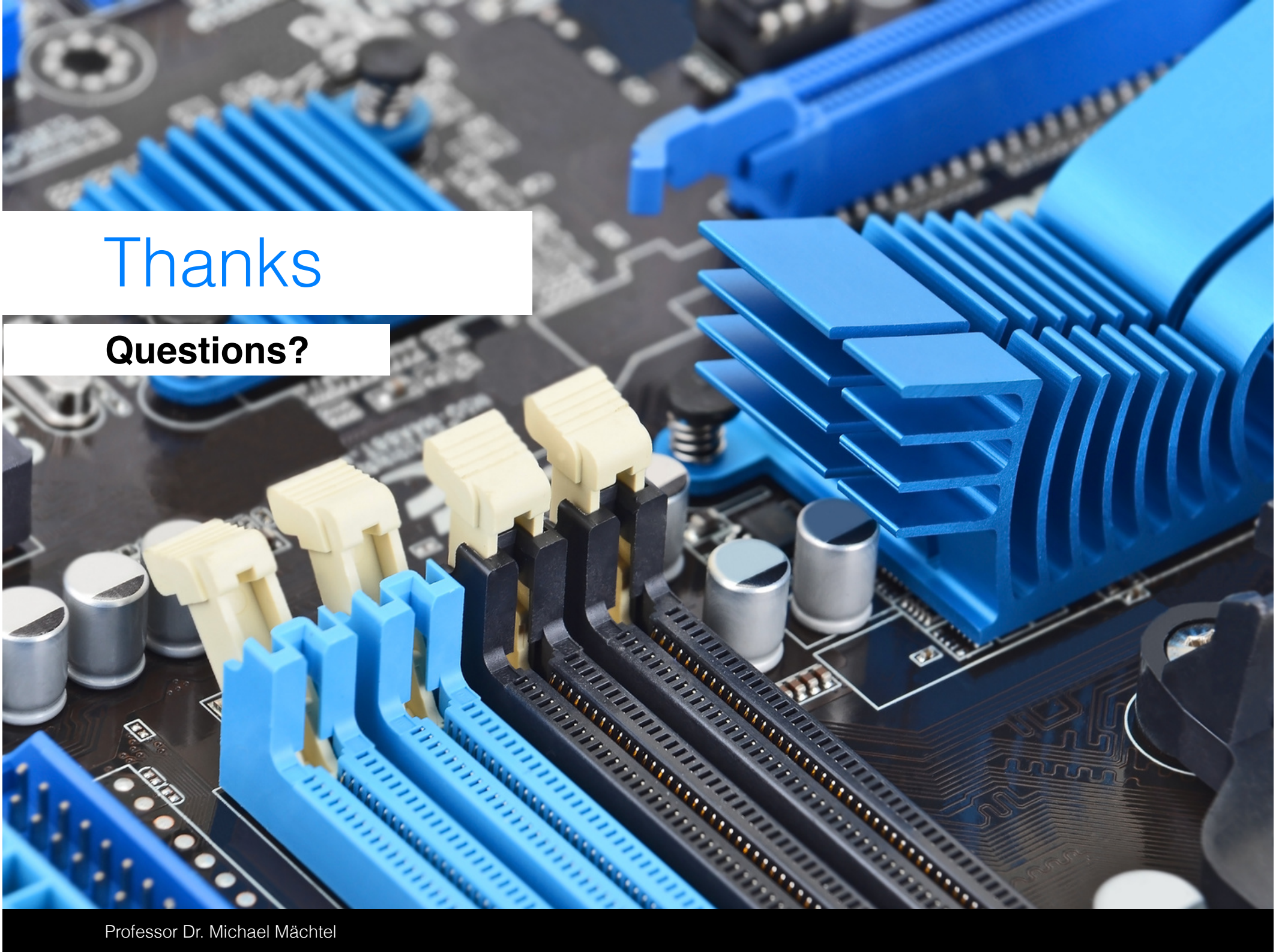
```
# address                         Perms     Offset           Dev    Inode       Pathname
00400000-00406000                 r-xp      00000000         03:05  1437882     /usr/sbin/metalog
00505000-00506000                 rw-p      00005000         03:05  1437882     /usr/sbin/metalog
00506000-00527000                 rw-p      00506000         00:00  0           [heap]
2b76d29df000-2b76d29f5000         r-xp      00000000         03:02  61500       /lib64/ld-2.3.6.so
2b76d29f5000-2b76d29fc000         rw-p      2b76d29f5000     00:00  0
2b76d2af5000-2b76d2af6000         r--p      00016000         03:02  61500       /lib64/ld-2.3.6.so
2b76d2af6000-2b76d2af7000         rw-p      00017000         03:02  61500       /lib64/ld-2.3.6.so
2b76d2af7000-2b76d2b0c000         r-xp      00000000         03:05  311804      /usr/lib64/libpcre.so.0.0.1
2b76d2b0c000-2b76d2c0c000         ---p      00015000         03:05  311804      /usr/lib64/libpcre.so.0.0.1
2b76d2c0c000-2b76d2c23000         rw-p      00015000         03:05  311804      /usr/lib64/libpcre.so.0.0.1
2b76d2c23000-2b76d2d41000         r-xp      00000000         03:02  62921       /lib64/tls/libc-2.3.6.so
2b76d2d41000-2b76d2e41000         ---p      0011e000         03:02  62921       /lib64/tls/libc-2.3.6.so
2b76d2e41000-2b76d2e44000         r--p      0011e000         03:02  62921       /lib64/tls/libc-2.3.6.so
2b76d2e44000-2b76d2e47000         rw-p      00121000         03:02  62921       /lib64/tls/libc-2.3.6.so
2b76d2e47000-2b76d2e4e000         rw-p      2b76d2e47000     00:00  0
7fffd80b5000-7fffd80cb000         rw-p      7fffd80b5000     00:00  0           [stack]
ffffffffff600000-ffffffffffe00000 ---p      00000000         00:00  0           [vdso]
```

cat /proc/790/maps

```
00400000-004af000                 r-xp      00000000         03:02     62290    /bin/bash
005ae000-005b9000                 rw-p      000ae000         03:02     62290    /bin/bash
005b9000-00728000                 rw-p      005b9000         00:00     0        [heap]
.....
```

TLB Summary

- **Hardware** can help us **make address translation faster**.
  - By TLB memory references often will be handled without having to access the page table in main memory.
  - the performance of the program will be almost as if memory isn't being virtualized at all.
- Locality depends on program:
  - **exceeding** the **TLB coverage**
  - TLB access can easily become a **bottleneck** in the **CPU pipeline**



Thanks

Questions?