Operating Systems

Memory Management II

Mohamed Zahran (aka Z)
mzahran@cs.nyu.edu
http://www.mzahran.com

What is the problem?

- Not enough memory
  - Having enough memory is not possible
    - How do you determine “enough”?
    - Programs keep growing in size.
- Processor does not execute anything that is not in the memory.

Have you ever thought …

- Why the *text* segment (Do you remember heap, stack, text, and data?) of any process starts at the same address?
- Why don't you run out of memory even if the sum of memory requirement of each program you are running at the same time exceeds the amount of memory you have in your machine?
- The address is 64-bit in 64-bit machines. It accesses memory from 0 to $2^{64} - 1$ which is way more than the amount of memory you have on your machine?

How come?

But We Can See That …

- All memory references are logical(virtual) addresses that are dynamically translated into physical addresses at run time
- A process may be broken up into several pieces that don't need to be contiguously located in main memory during execution.

So:

It is not necessary that all the pieces of a process be in main memory during execution.

Definition of Virtual Memory

Mapping from logical (virtual) address space to physical address space

Implementation of Virtual Memory Using Paging

The Story

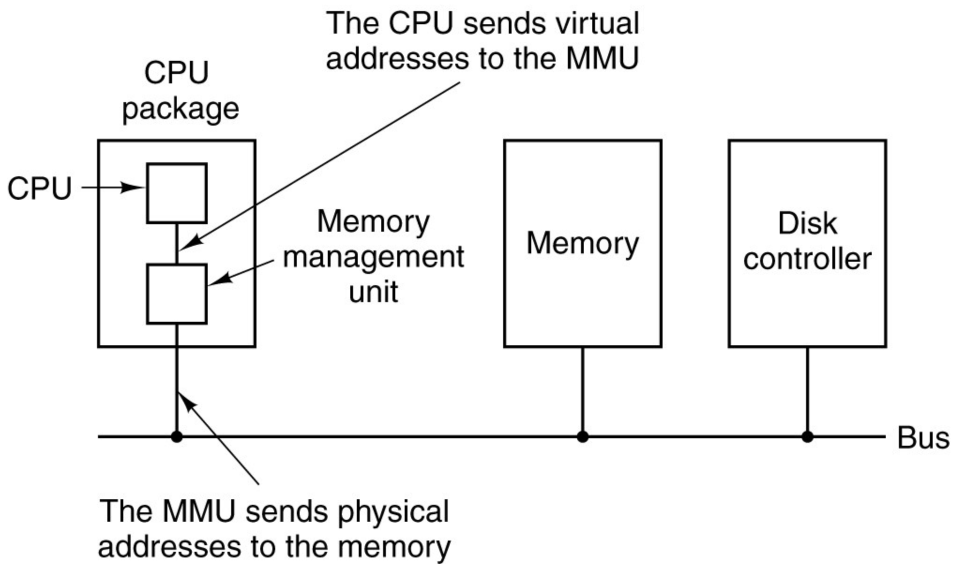
1. Operating system brings into main memory a few pieces of the program.
2. An interrupt is generated when an address is needed that is not in main memory.
3. Operating system places the process in a blocking state.
4. Operating system issues a disk I/O Read request.
5. Another process is dispatched to run while the disk I/O takes place.
6. An interrupt is issued when disk I/O is complete, which causes the operating system to place the affected process in the Ready state
7. Piece of process that contains the logical address is brought into main memory.

The Story

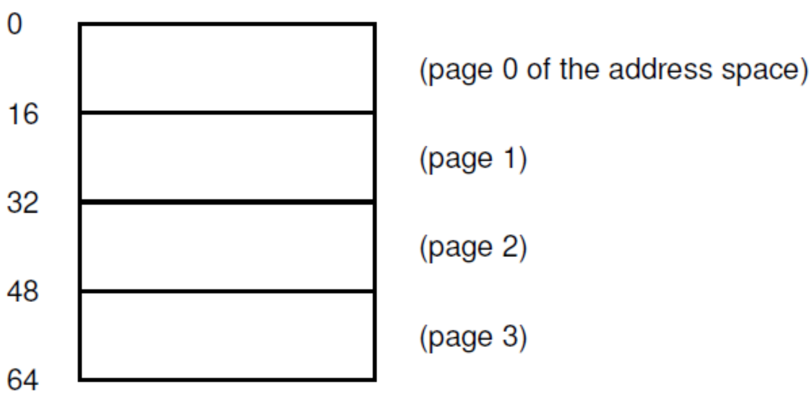
1. Operating system brings into main memory a few pieces of the program. What do you mean by “pieces”?
2. An interrupt is generated when an address is needed that is not in main memory. How do you know it isn’t in memory?
3. Operating system places the process in a blocking state.
4. Operating system issues a disk I/O Read request. Why?
5. Another process is dispatched to run while the disk I/O takes place.
6. An interrupt is issued when disk I/O is complete, which causes the operating system to place the affected process in the Ready state
7. Piece of process that is addressed by the logical address is brought into main memory. What if memory is full?

Virtual Memory

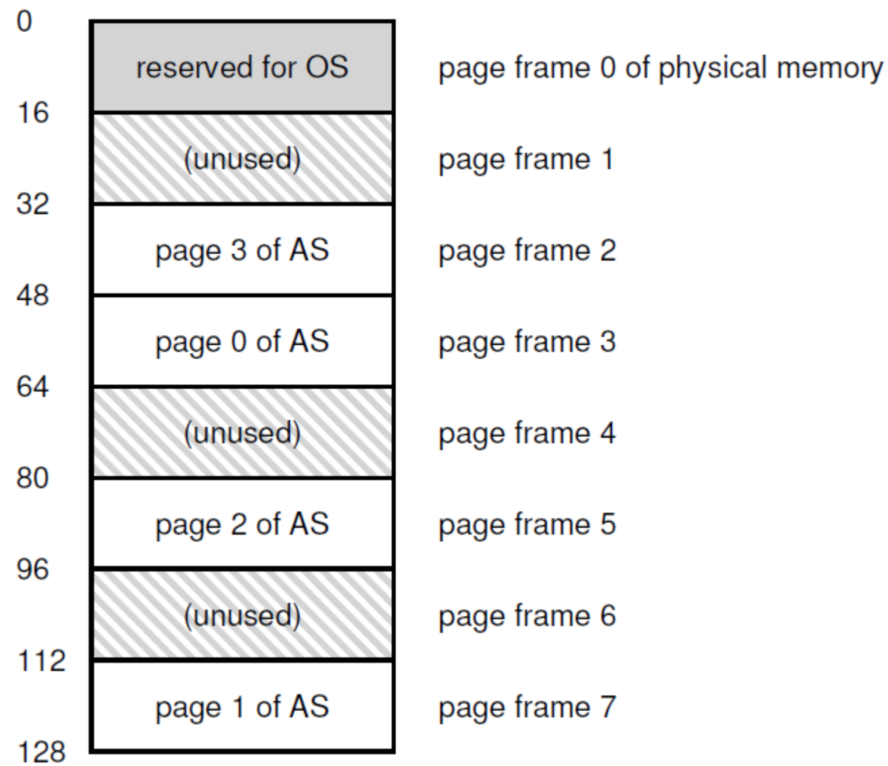
- Each program has its own address space
- This address space is divided into pages
- Pages are mapped into physical memory



A Tiny Example



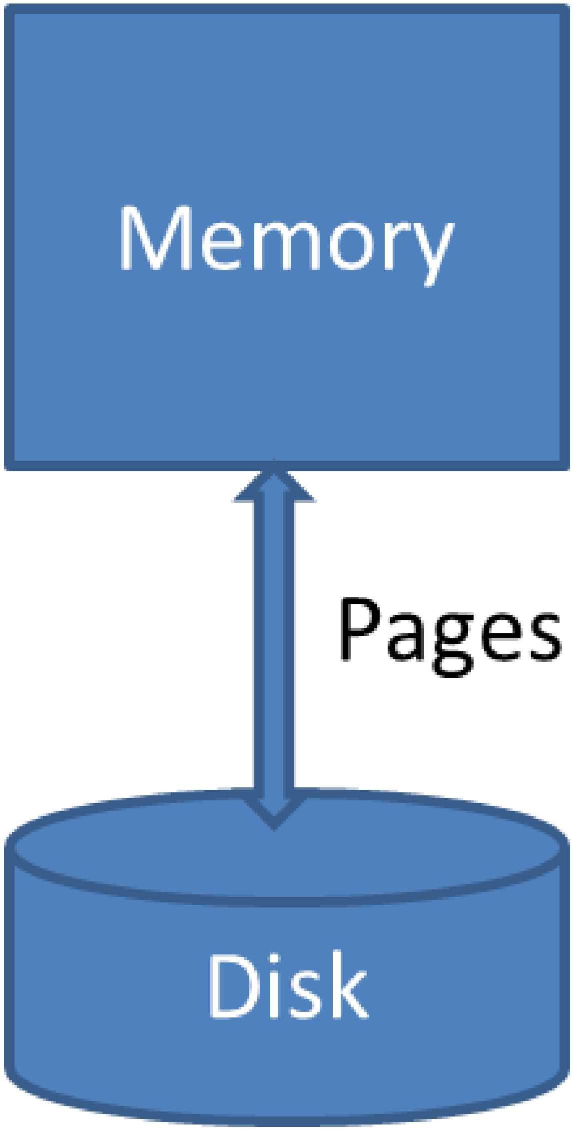
Virtual Address Space



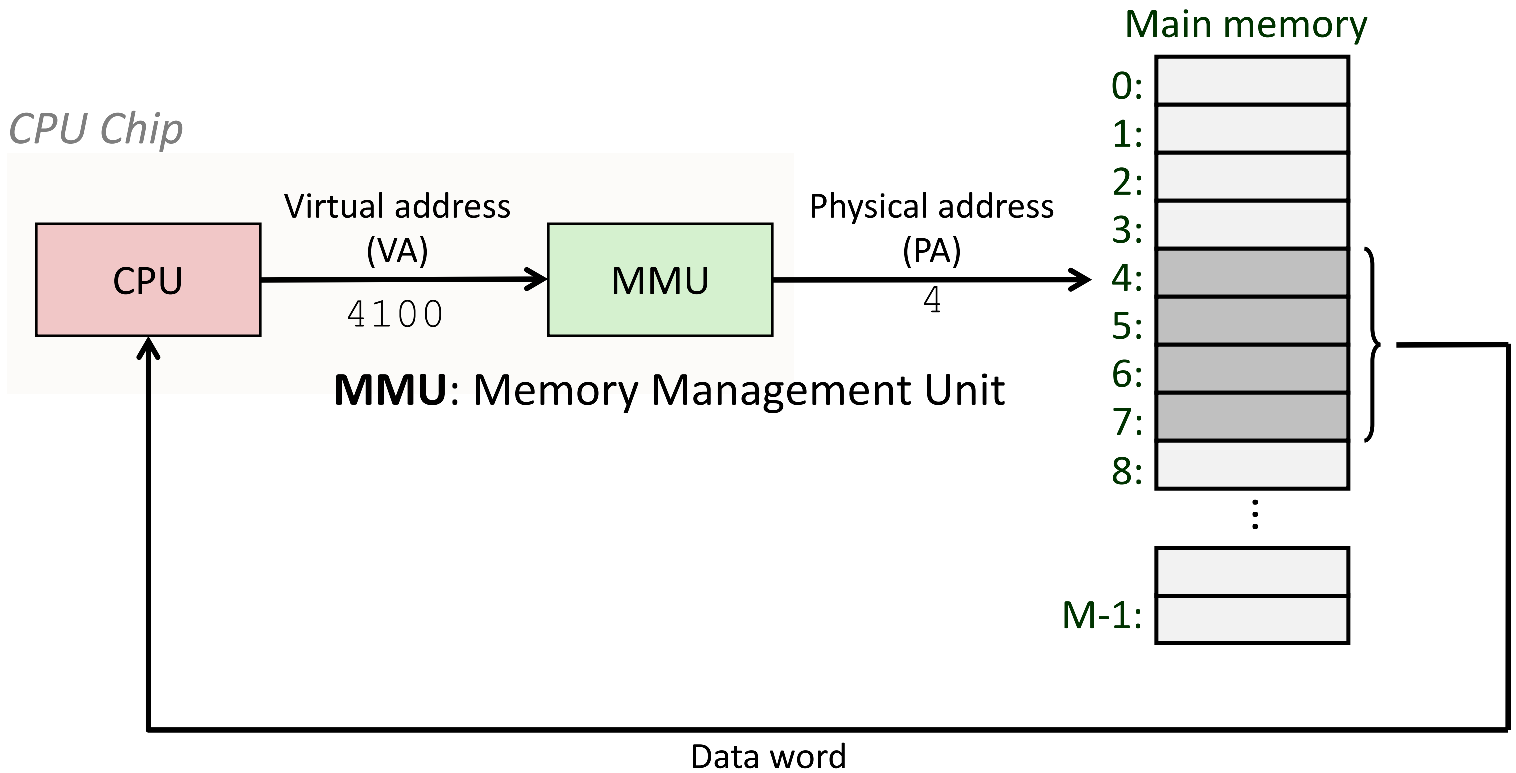
Physical Memory

In real life, the virtual address space is much bigger than the physical memory.

Use part of the disk as extension to the memory!

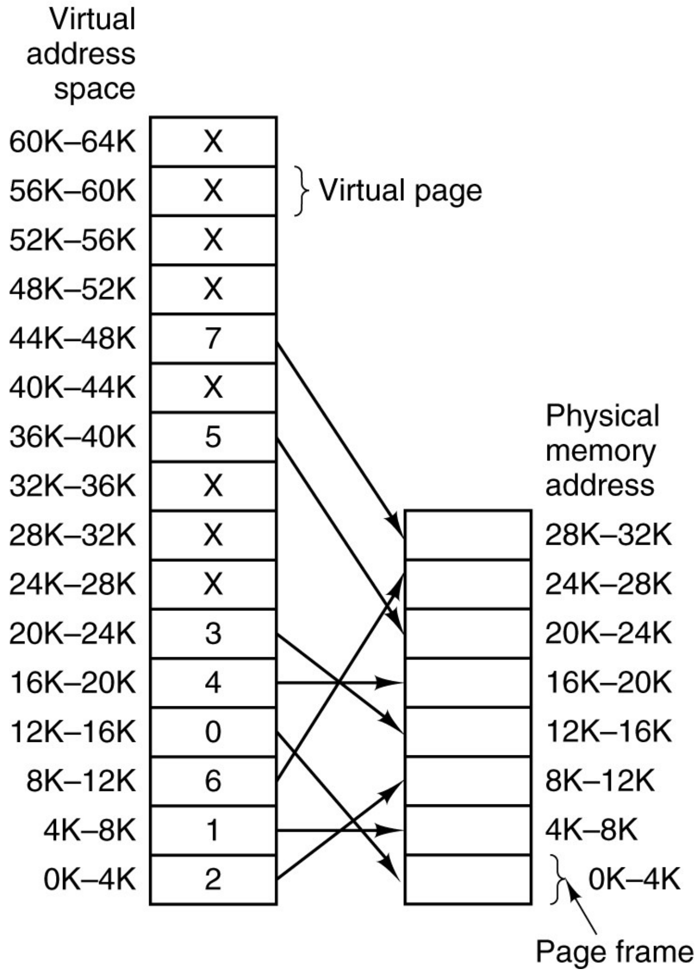


Virtual Addressing



- Used in all modern machines.

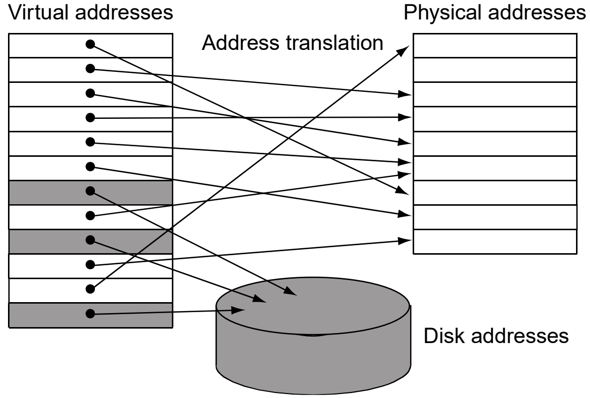
Virtual Memory



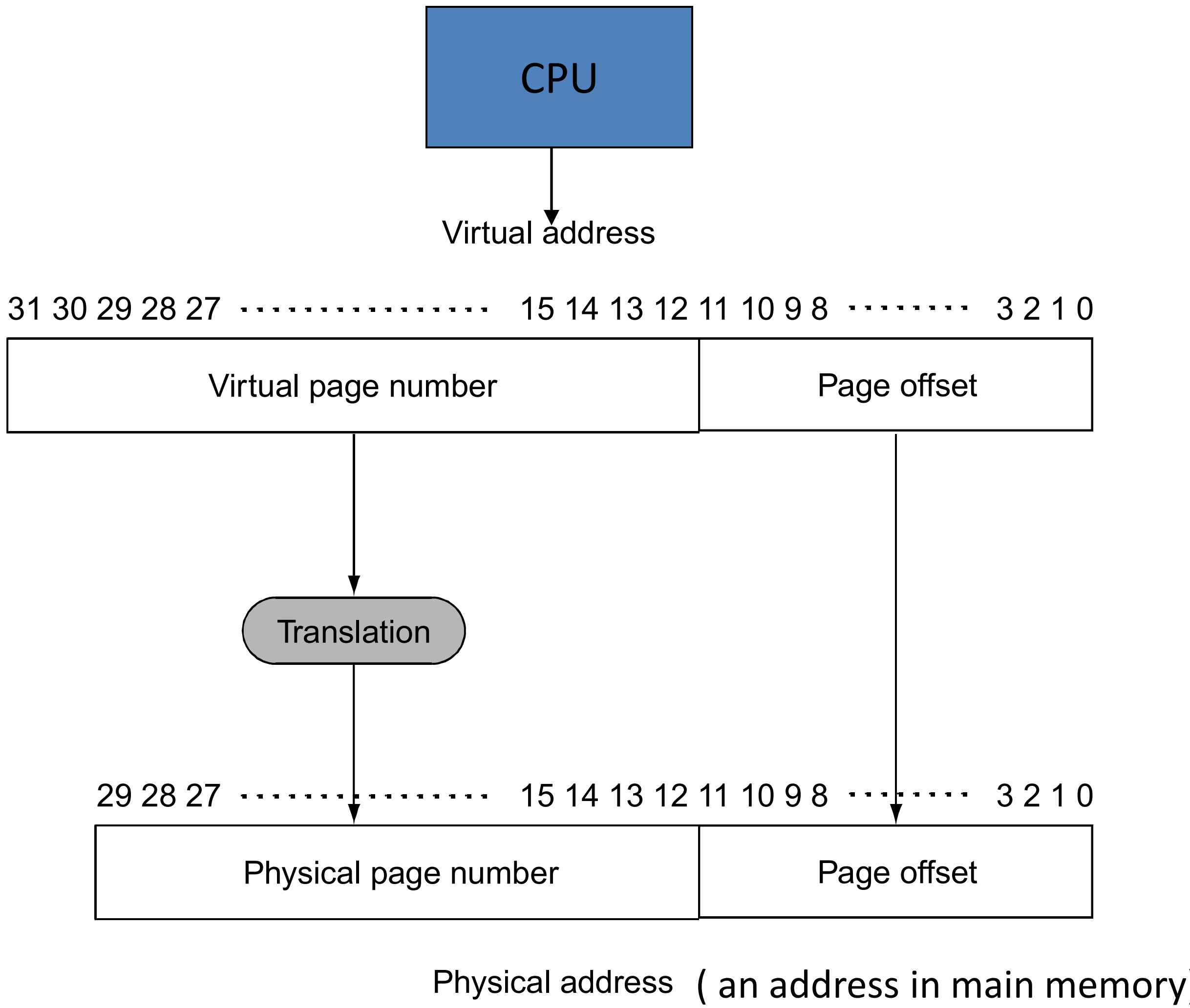
Virtual Address Space **Physical Memory**

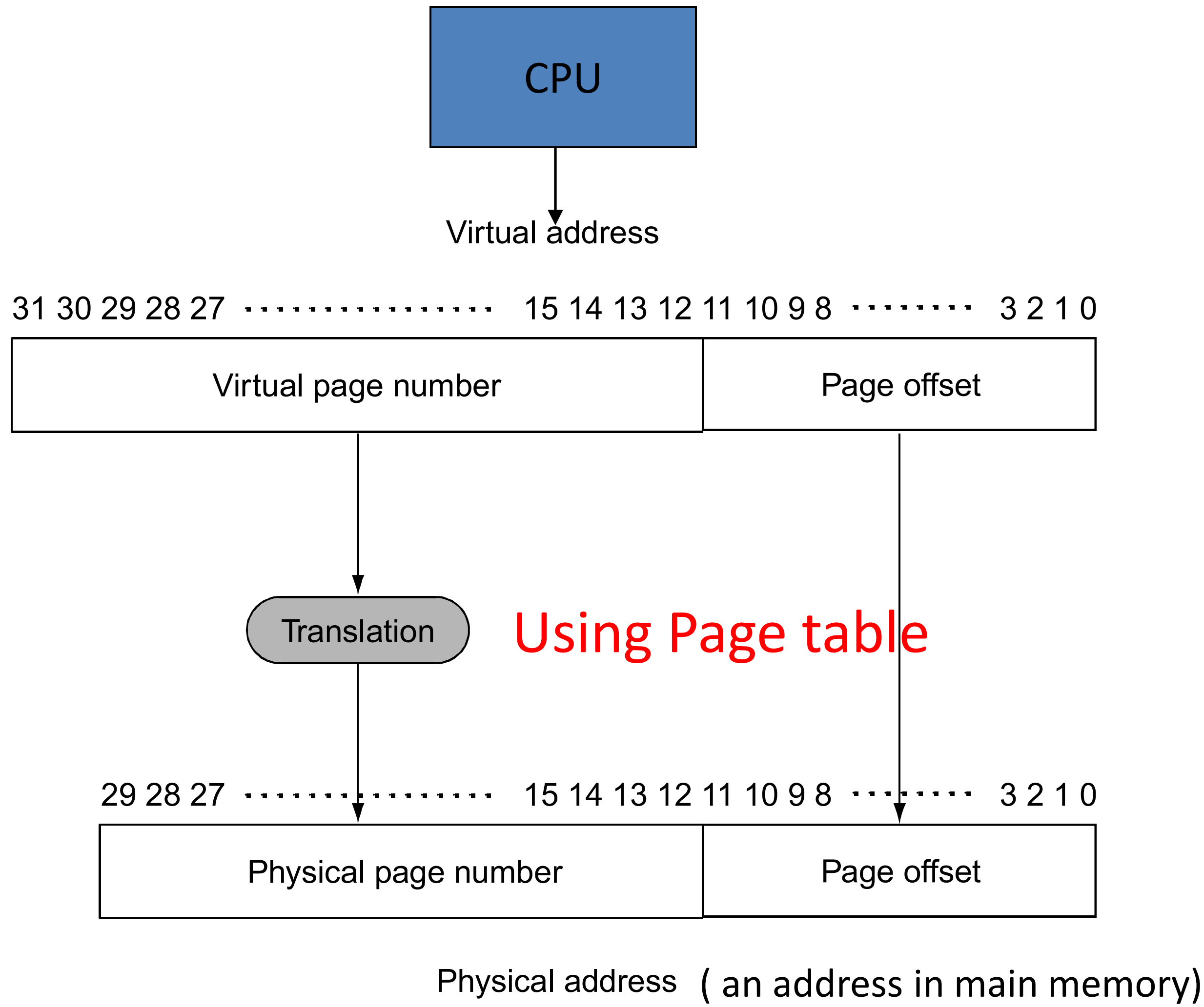
Virtual Memory

- We can think of memory as acting as a cache to the secondary storage (disk)



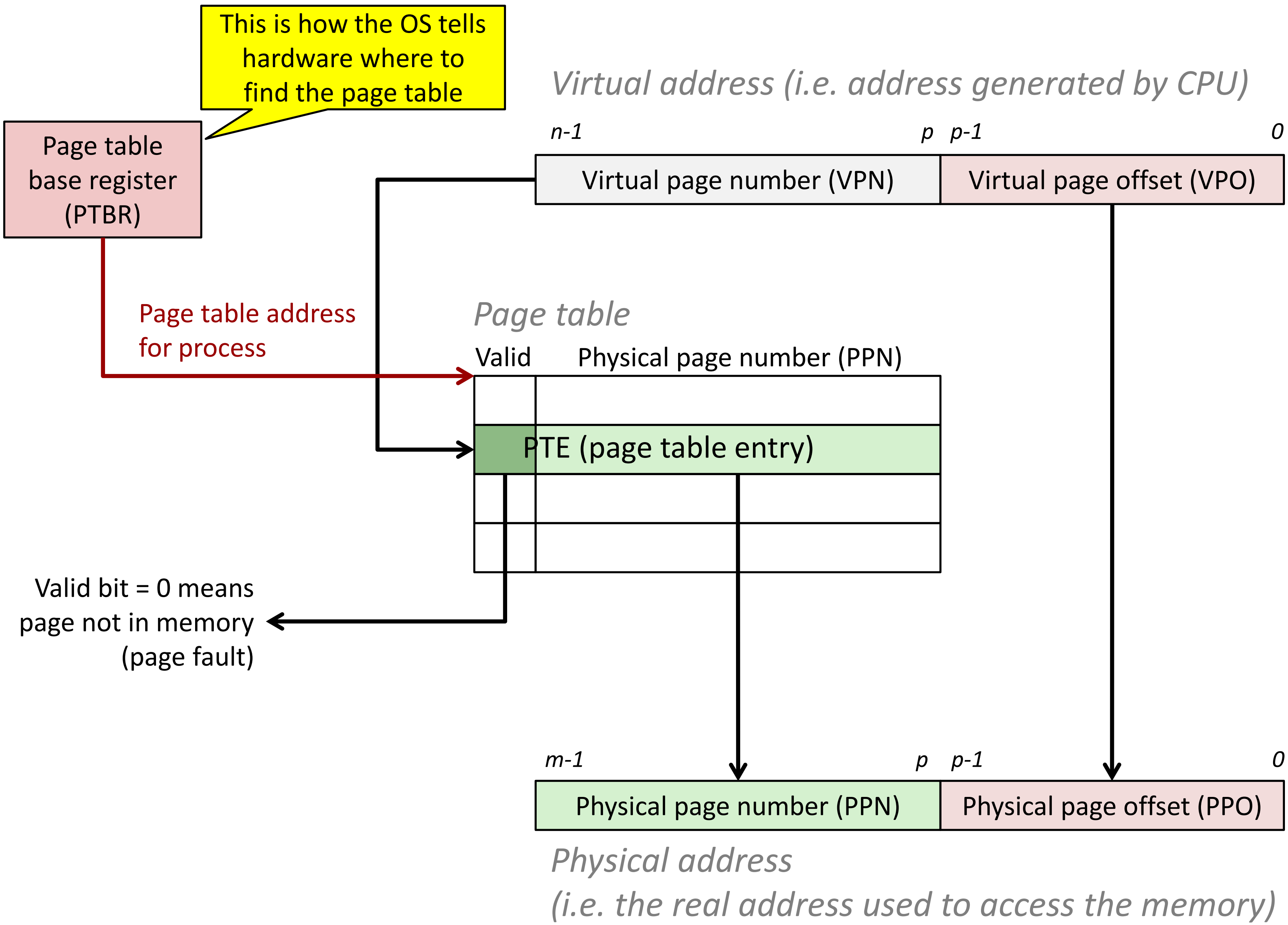
- Advantages:
  - illusion of having more physical memory
  - program relocation
  - protection


CPU
Virtual address
31 30 29 28 27 15 14 13 12 11 10 9 8 3 2 1 0
Virtual page number
Page offset
Translation
29 28 27 15 14 13 12 11 10 9 8 3 2 1 0
Physical page number
Page offset
Physical address
(an address in main memory)



MMU = Memory Management Unit

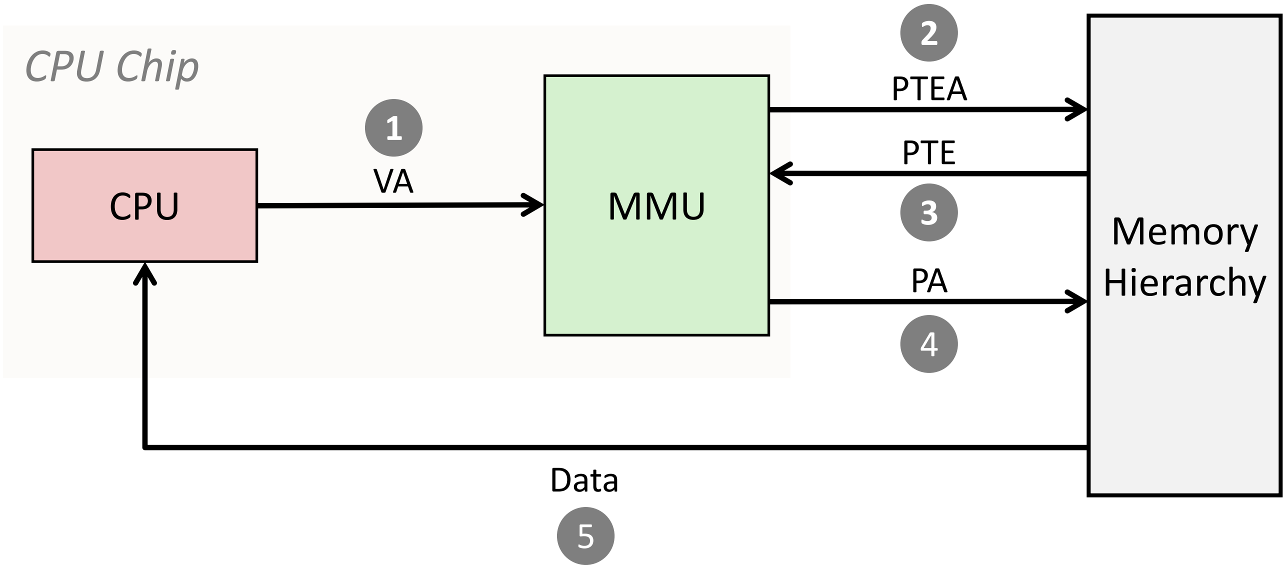
Address Translation w/ a Page Table



Page Table Base Register (PTBR)

- The operating system maintains information about each process in a process control block.
- The page table base address for the process is stored there.
- The operating system loads this address into the PTBR whenever a process is scheduled for execution.
- Only the kernel (i.e. the OS) can access PTBR

Address Translation: Page Hit



VA: Virtual Address
PA: Physical Address
PTE: Page Table Entry
PTEA: PTE Address

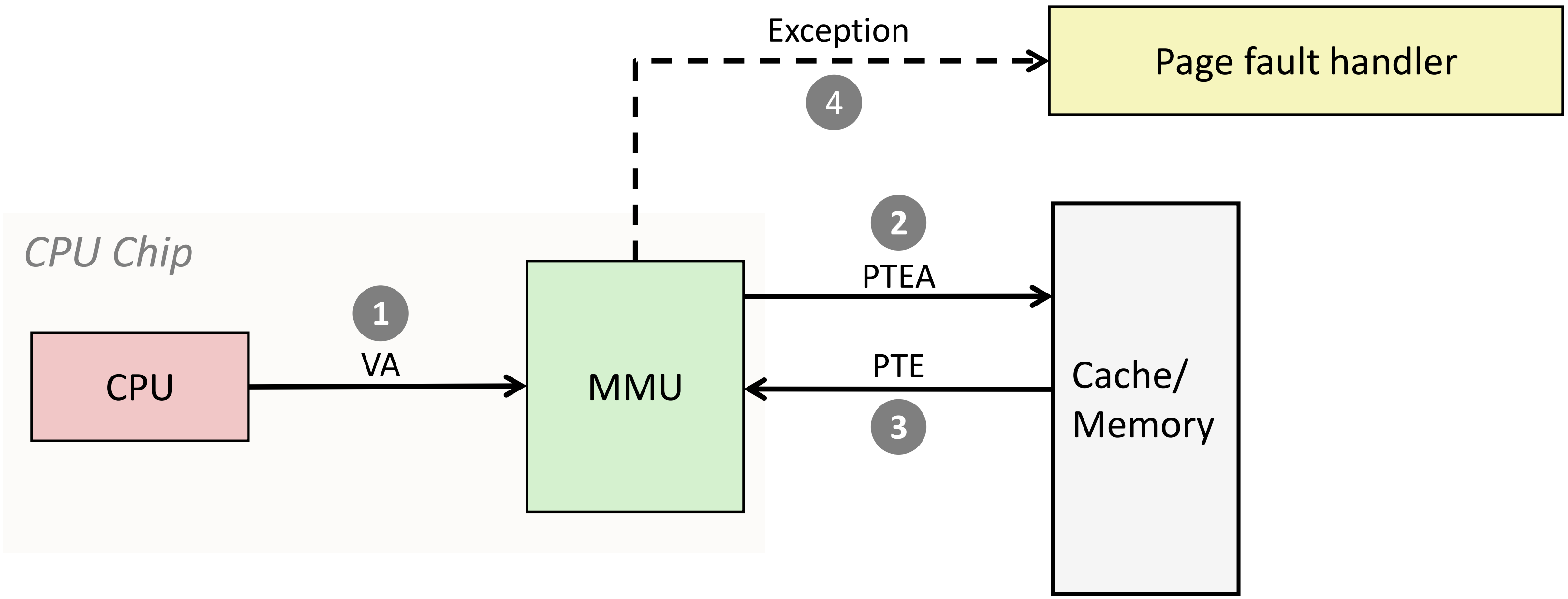
1) Processor sends virtual address to MMU

2-3) MMU fetches PTE from page table in memory

4) MMU sends physical address to cache/memory

5) Cache/memory sends data word to processor

Address Translation: Page Fault



1) Processor sends virtual address to MMU

2-3) MMU fetches PTE from page table in memory

4) Valid bit is zero, so MMU triggers page fault exception in kernel

If VA is invalid, then kill process (SIGSEGV)

If VA has been paged out to disk, then swaps in faulted page, update page table, resume faulted process

There are two challenges

- **Speed**: VA to PA translation means we need to access the memory twice for each CPU memory request!

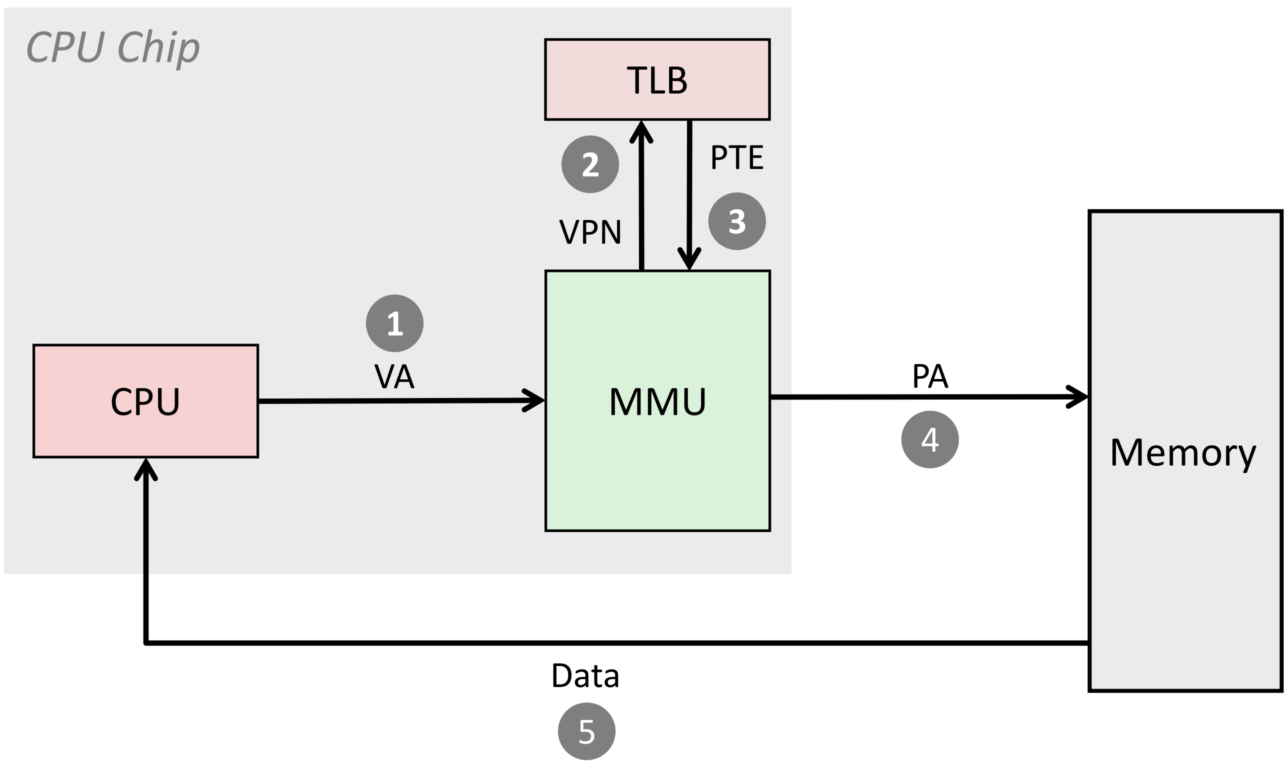
- **Size**: page table can be huge. And we have a page table per process.

Speeding Up Virtual Memory: Translation Lookaside Buffer (TLB)

TLB

- **Observation**: most programs tend to make a large number of references to a small number of pages -> only fraction of the page table is heavily used
- **Solution:** *Translation Lookaside Buffer* (TLB)
  - Small hardware cache inside the MMU (i.e. on chip)
  - Maps virtual page numbers to physical page numbers address without going to the page table
  - Contains complete page table entries for small number of pages

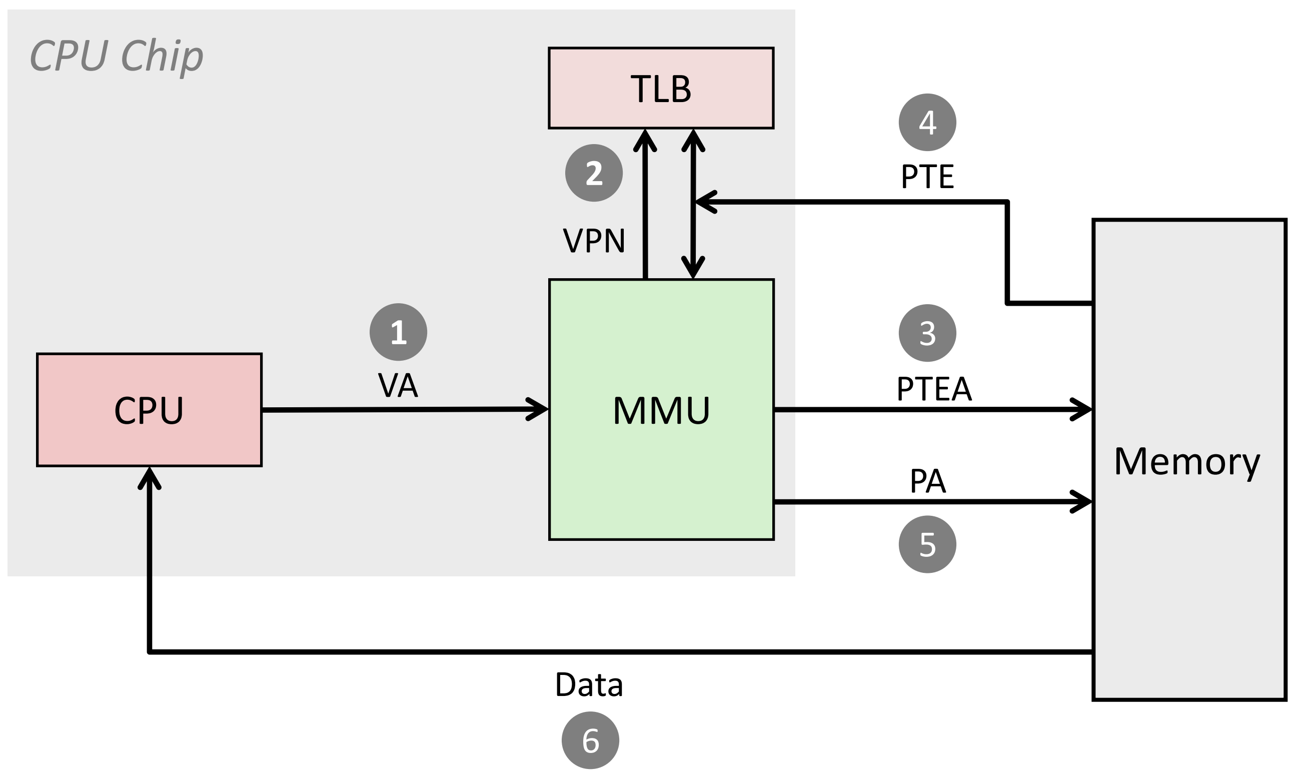
TLB Hit



A TLB hit eliminates a memory access

VA: Virtual Address **PA:** Physical Address **PTE:** Page Table Entry

TLB Miss



A TLB miss incurs an additional memory access (the PTE)

Fortunately, TLB misses are rare.

TLB

- In case of TLB miss (did not find the address translation you want) -> MMU accesses page table
- TLB misses occur more frequently than page faults

Example of contents of a TLB

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Page frame is the physical page number.

TLB In Case of Context Switch

- TLB contains translations from the page table of a process.
- If a new process is schedule for execution, the page table of that new process is used.
- **Solution 1:** TLB must be flushed.
- **Solution 2:** TLB augmented with process ID.

Conclusions

- Virtual memory is very widely used
- The main pieces of the puzzle are:
  - Page
  - Page frame
  - MMU
  - TLB
  - PTBR