Linux Kernel Training. Lecture 3, part 1

Hardware Overview

Oleksandr Redchuk
<oleksandr.redchuk@gmail.com>

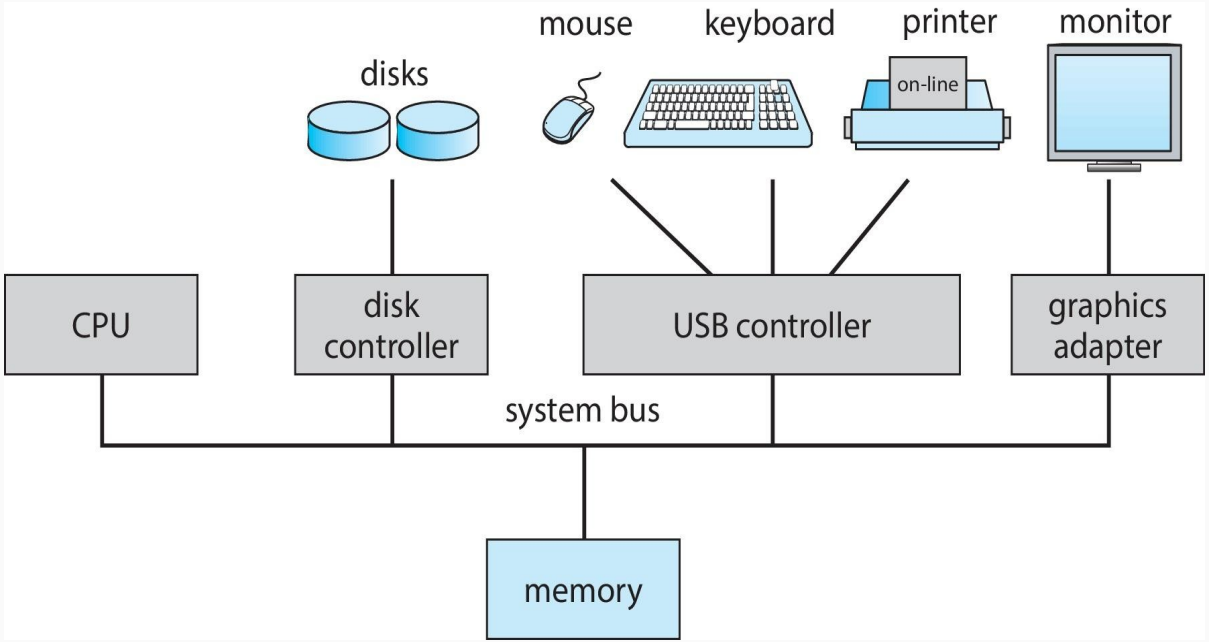
March 30, 2020. GlobalLogic

Operating System

Operating system is a **resource allocator** and **control** program making efficient use of HW and managing execution of user programs

“The one program running at all times on the computer” is the **kernel**, part of the operating system

Users want convenience, ease of use and good performance (Don’t care about resource utilization)



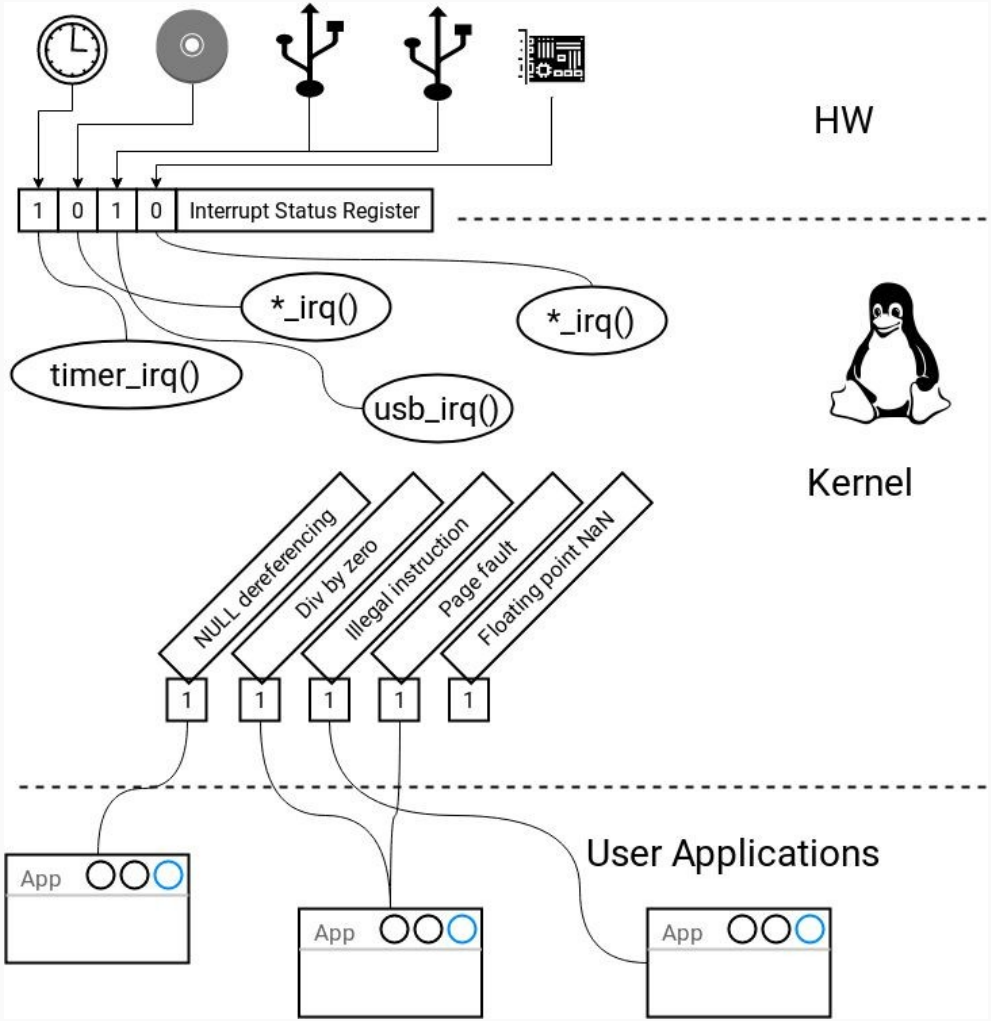
Interrupts and Exceptions

I/O devices and the CPU can execute concurrently

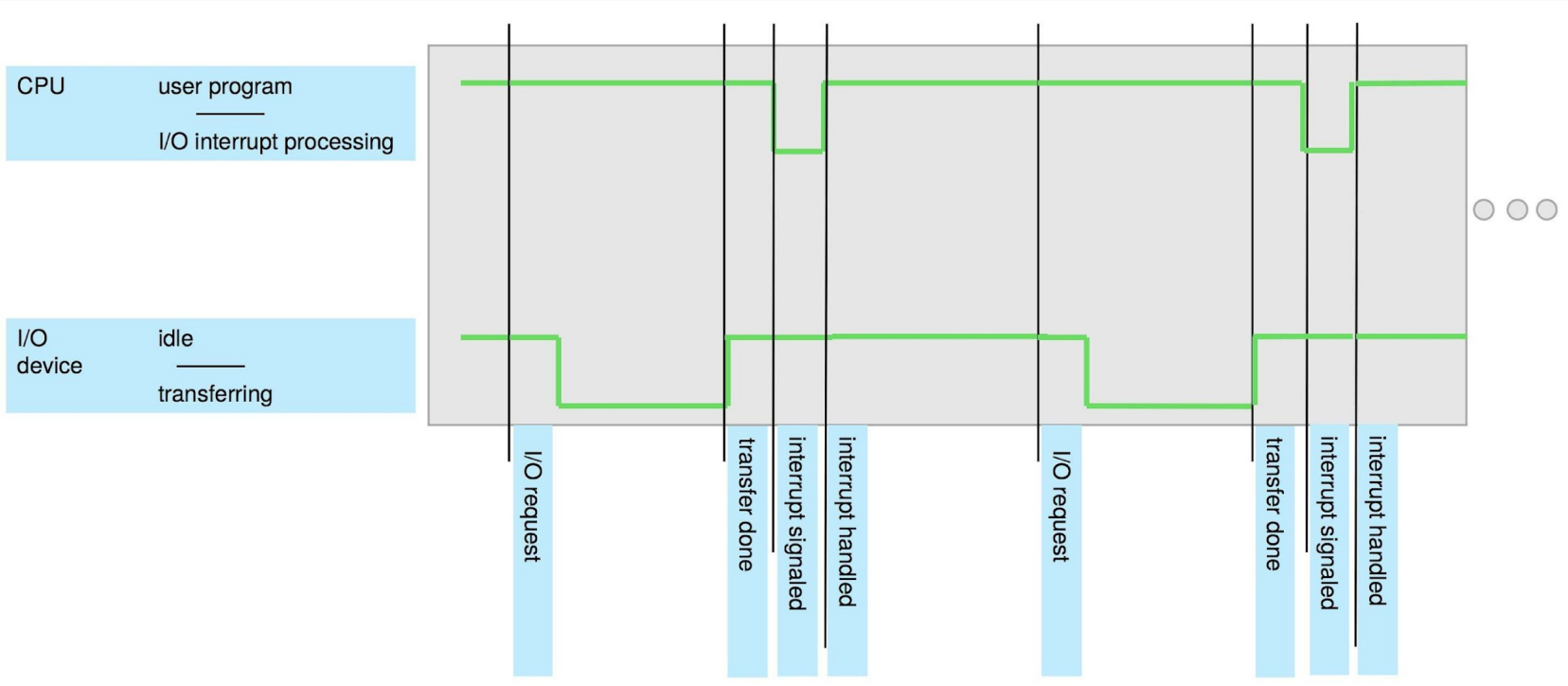
Device controller informs CPU that it has finished its operation by causing an interrupt

A trap or exception is a software-generated interrupt caused either by an error or a user request

An operating system is interrupt driven

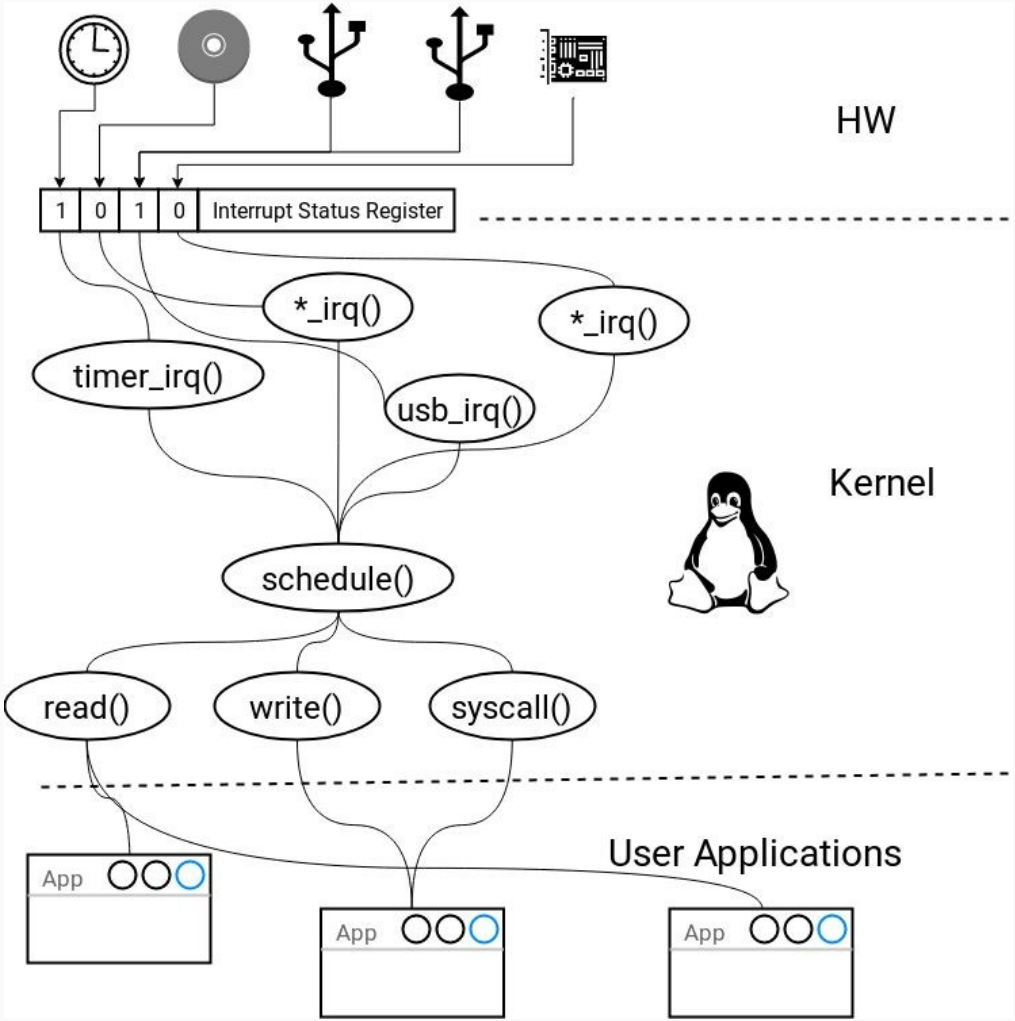


Interrupt Timeline



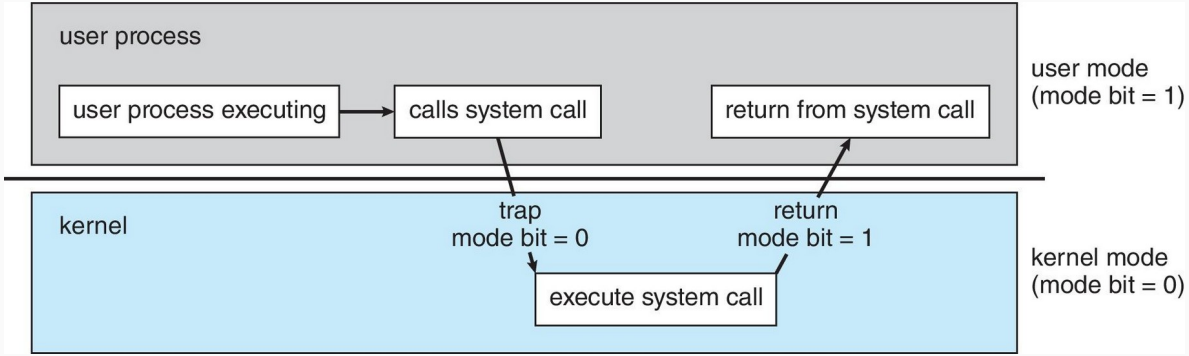
schedule() callers

System call – request to the OS



Dual-mode Operation

User mode and kernel mode



Dual-mode Operation

User to kernel mode

- Exceptions
- Interrupts
- System calls

Kernel to user mode

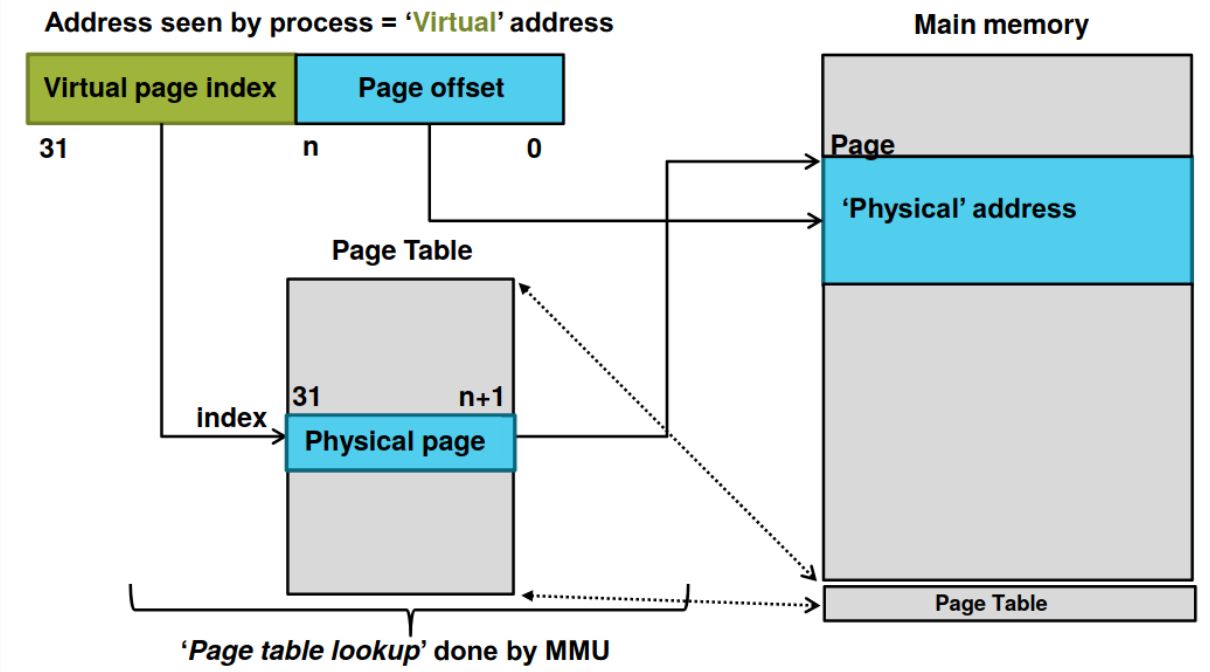
- New process
- Resume after an exception, interrupt or system call
- Switch to a different process
- User-level upcall

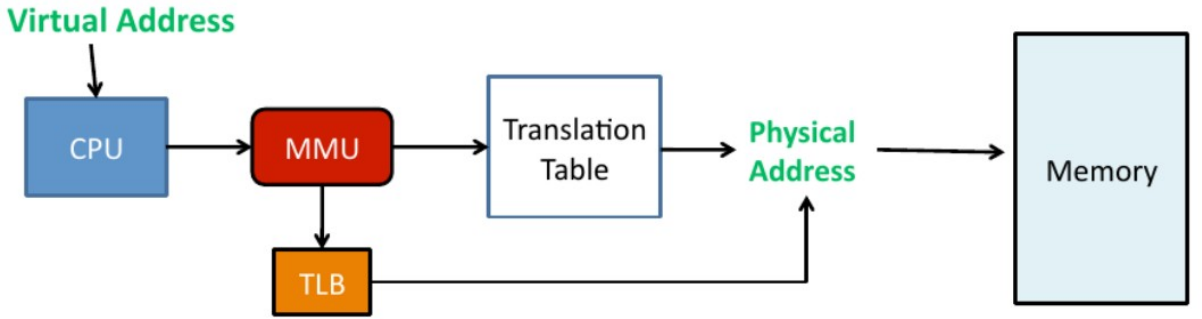
MMU

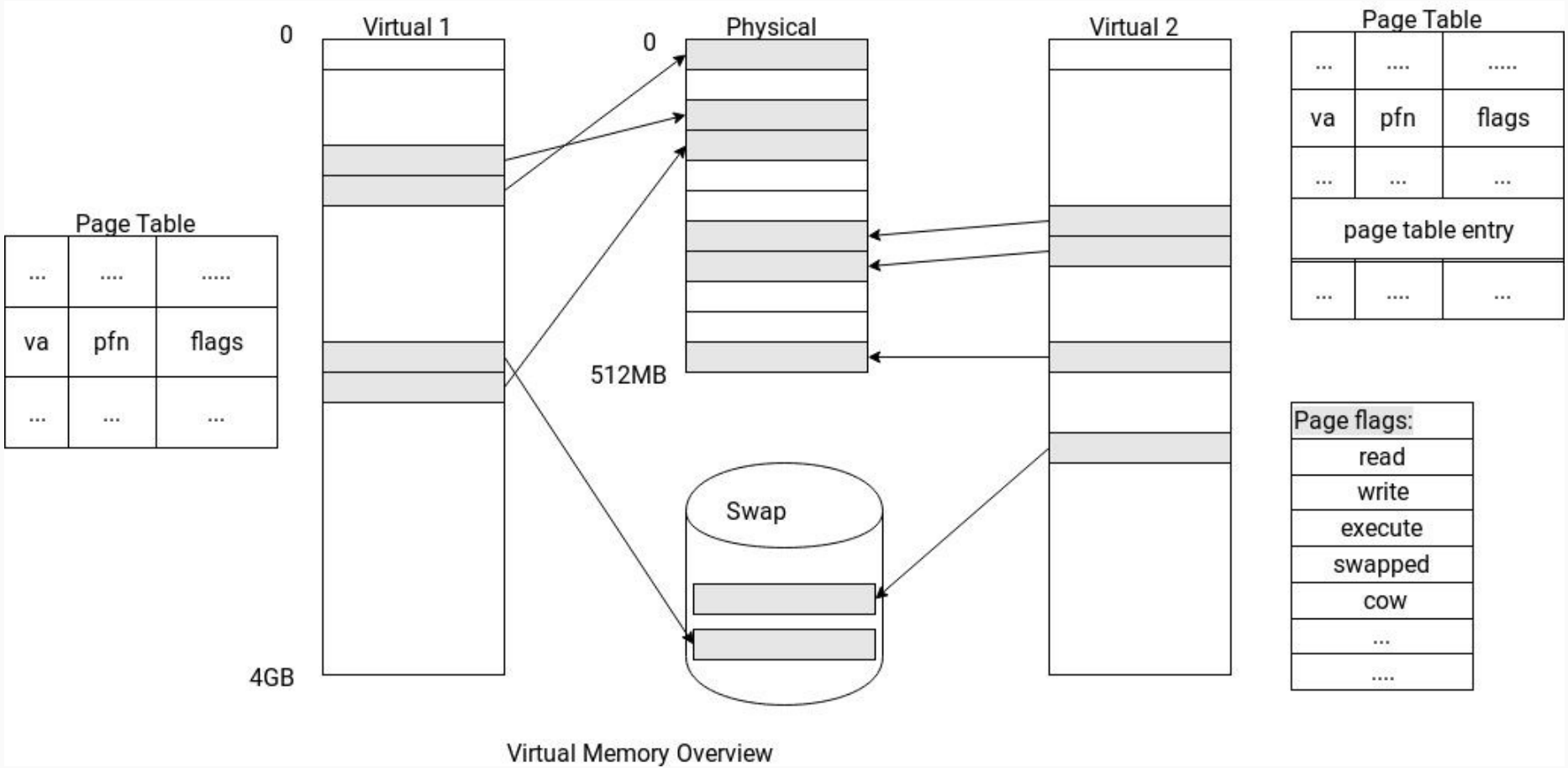
Memory-Management Unit (MMU) device that maps virtual to physical address

Page table — **per process table** used to translate logical to physical addresses

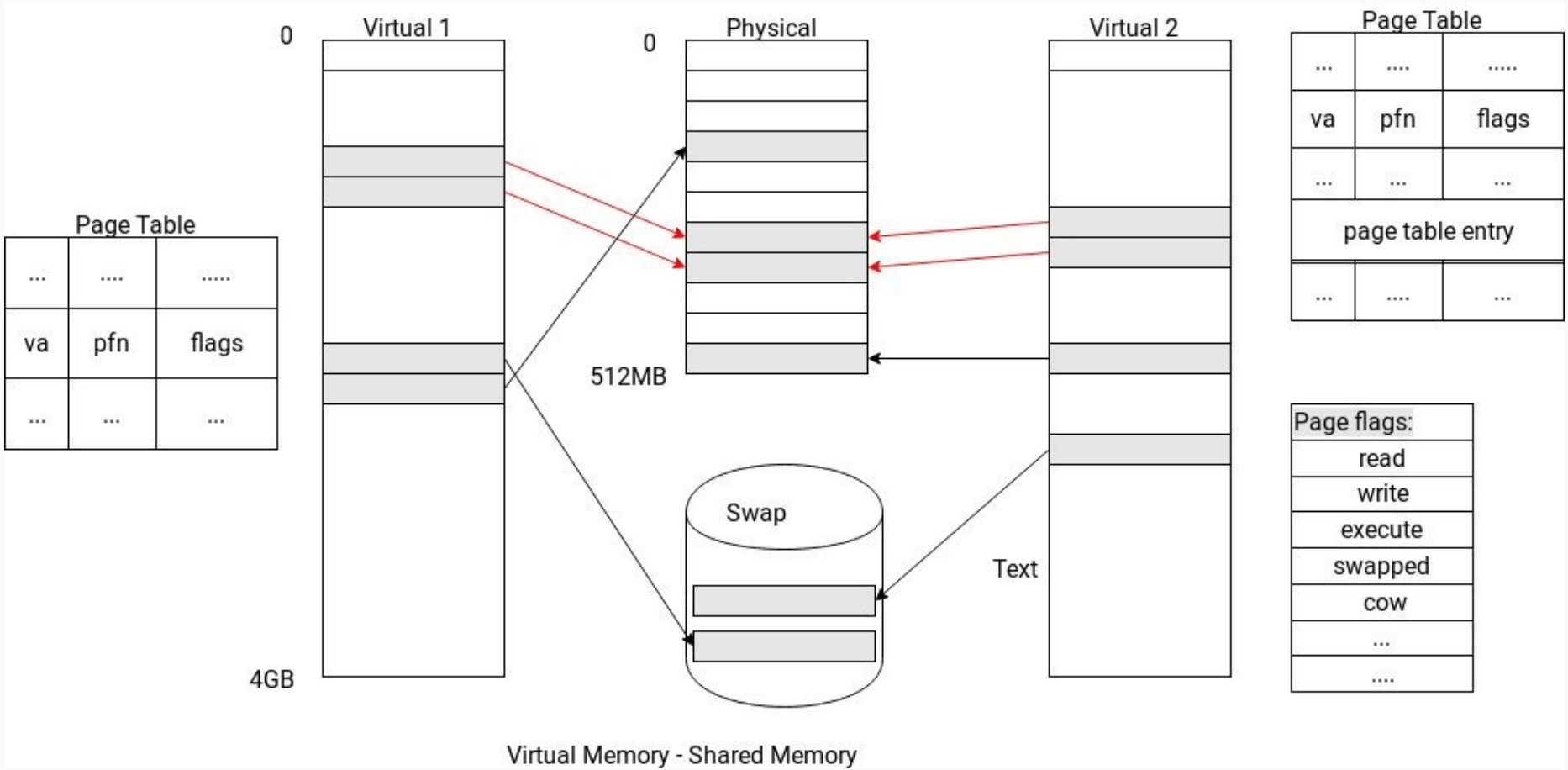
Translation Look-aside Buffer (TLB) is a CPU cache that memory management hardware uses to improve virtual address translation speed



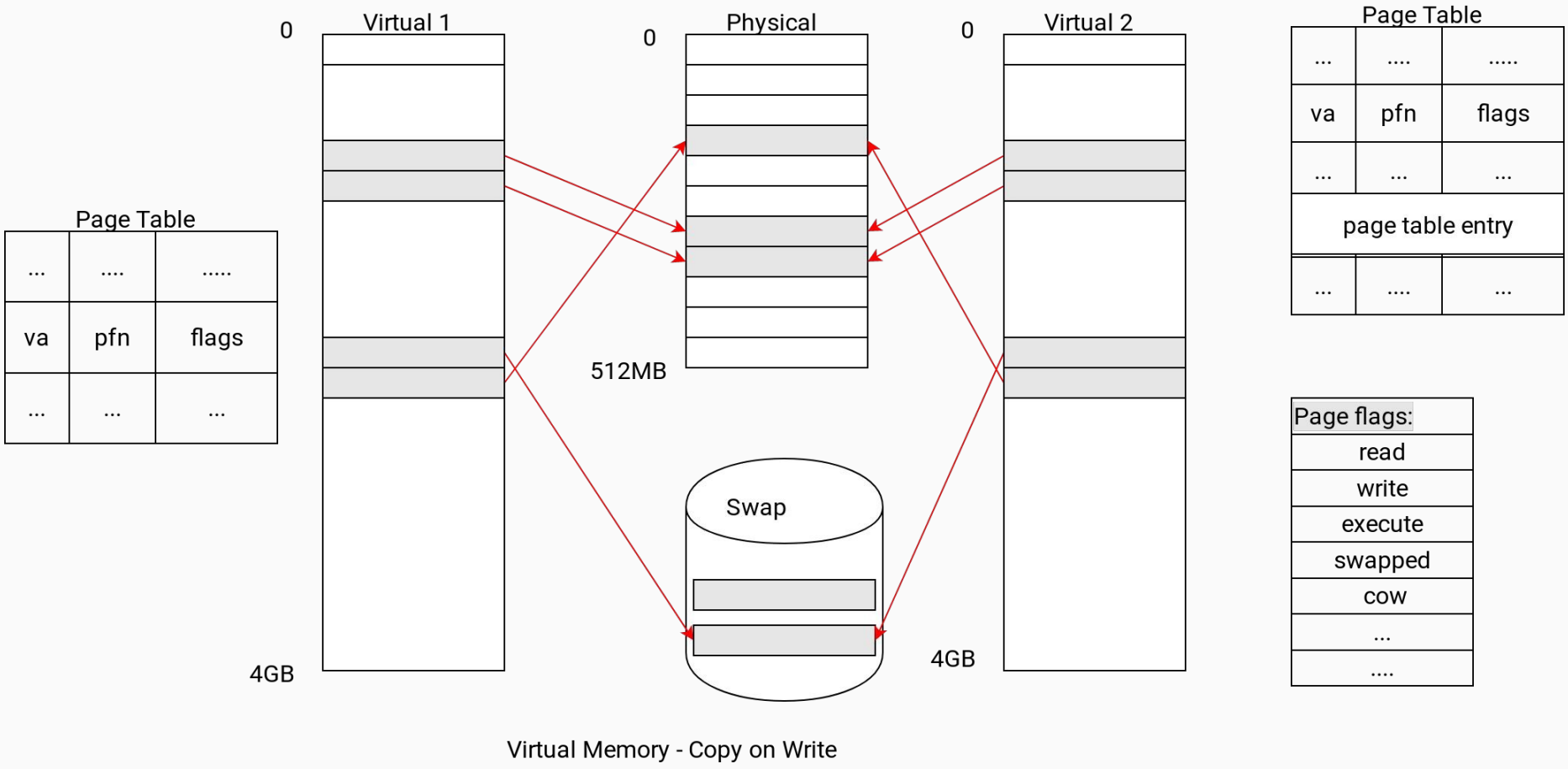




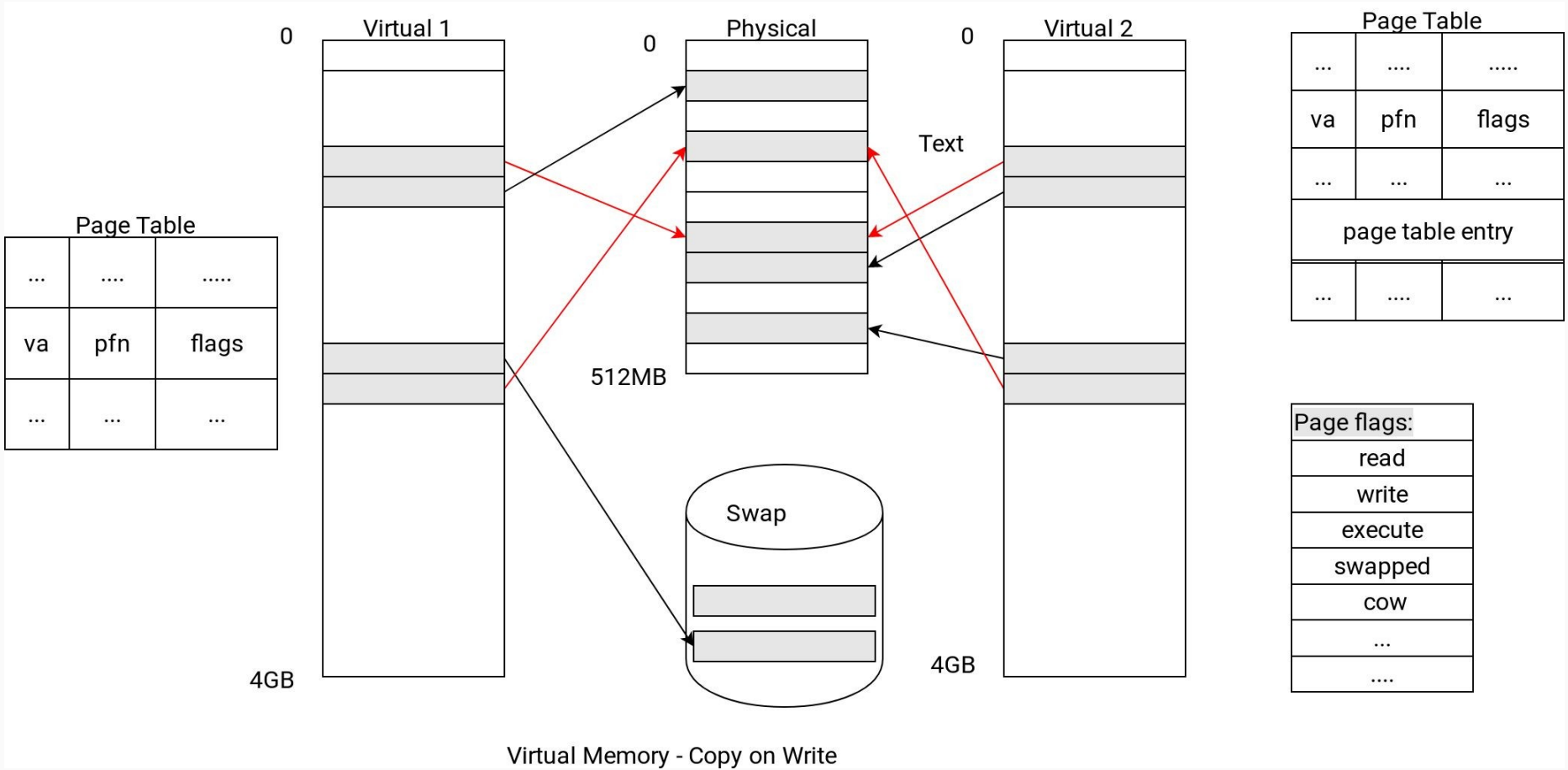
Virtual Memory Overview



Virtual Memory - Shared Memory

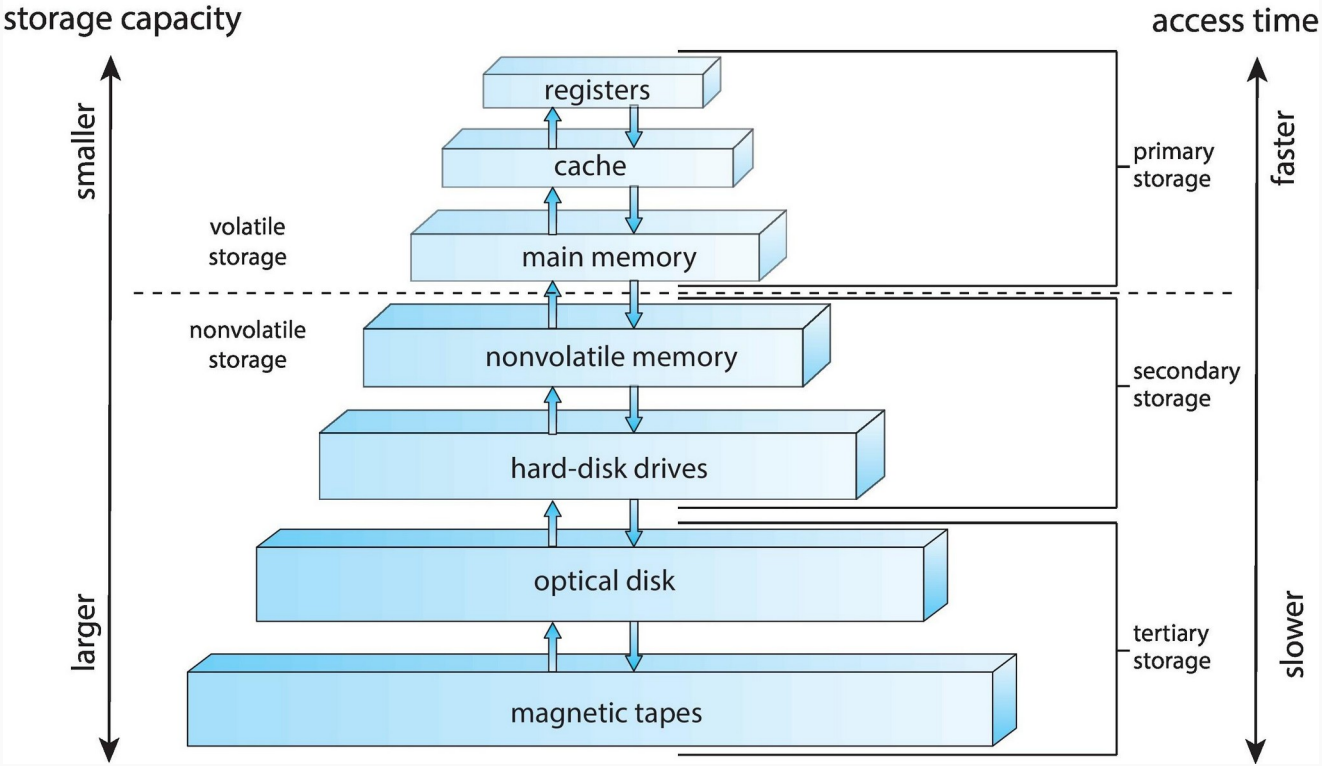


Virtual Memory - Copy on Write



Virtual Memory - Copy on Write

Storage-Device Hierarchy

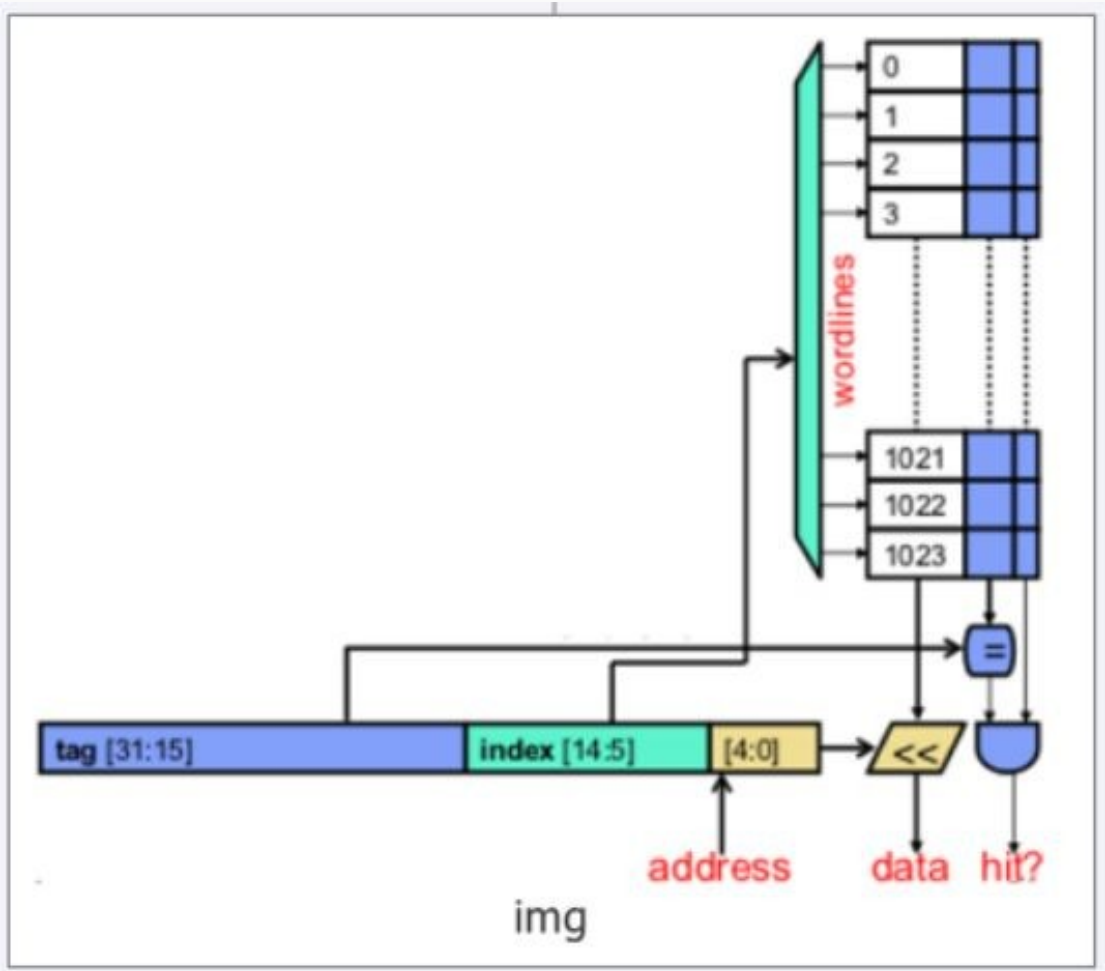


Characteristics of Various Types of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

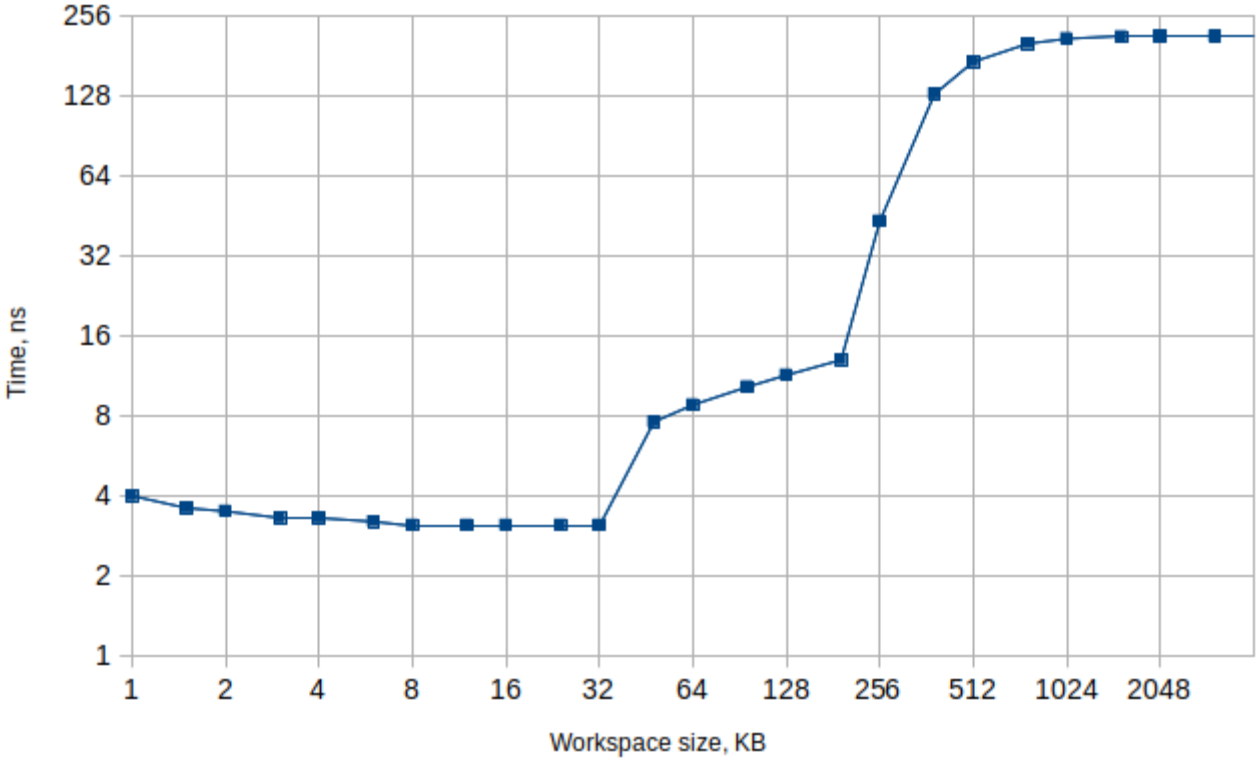
Caches

Caching – copying information into faster storage system; main memory can be viewed as a cache for secondary storage.



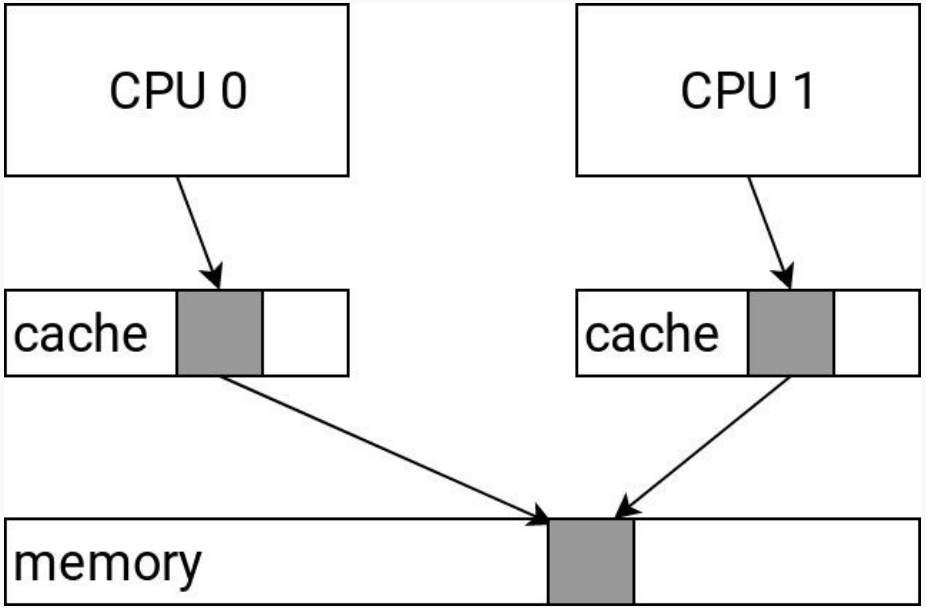
img

BBB: L1D, L2 and memory access time

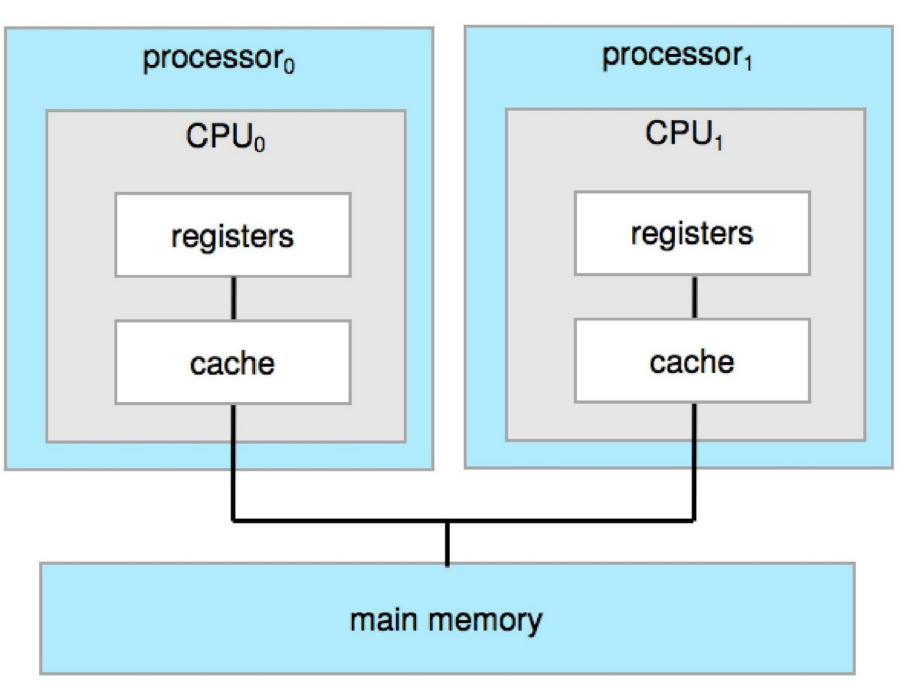


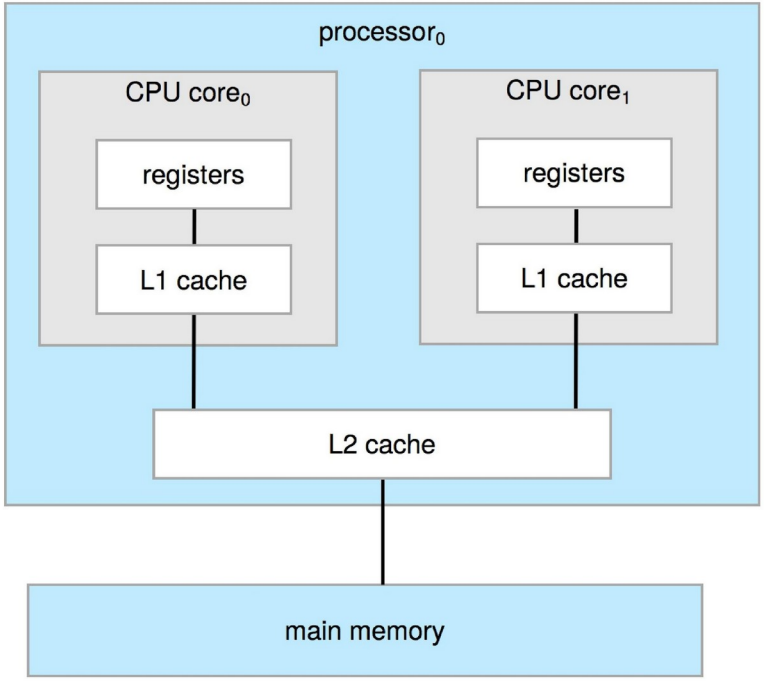
- Ulrich Drepper. What every programmer should know about memory (LWN.net)

Caches and memory access



SMP vs Multicore Design





Atomic Operations

GCC flavor:

```
type __sync_fetch_and_add
type __sync_fetch_and_sub
type __sync_fetch_and_or
type __sync_fetch_and_and
type __sync_fetch_and_xor
type __sync_fetch_and_nand

bool __sync_bool_compare_and_swap
type __sync_val_compare_and_swap

void __sync_synchronize
```

C++ 11: <atomic>

```
atomic_store
atomic_store_explicit

atomic_exchange
atomic_exchange_explicit

atomic_fetch_add
atomic_fetch_add_explicit

atomic_thread_fence
```

CPU pipelines

The root of the single cycle processor's problems:

- The cycle time has to be long enough for the slowest instruction

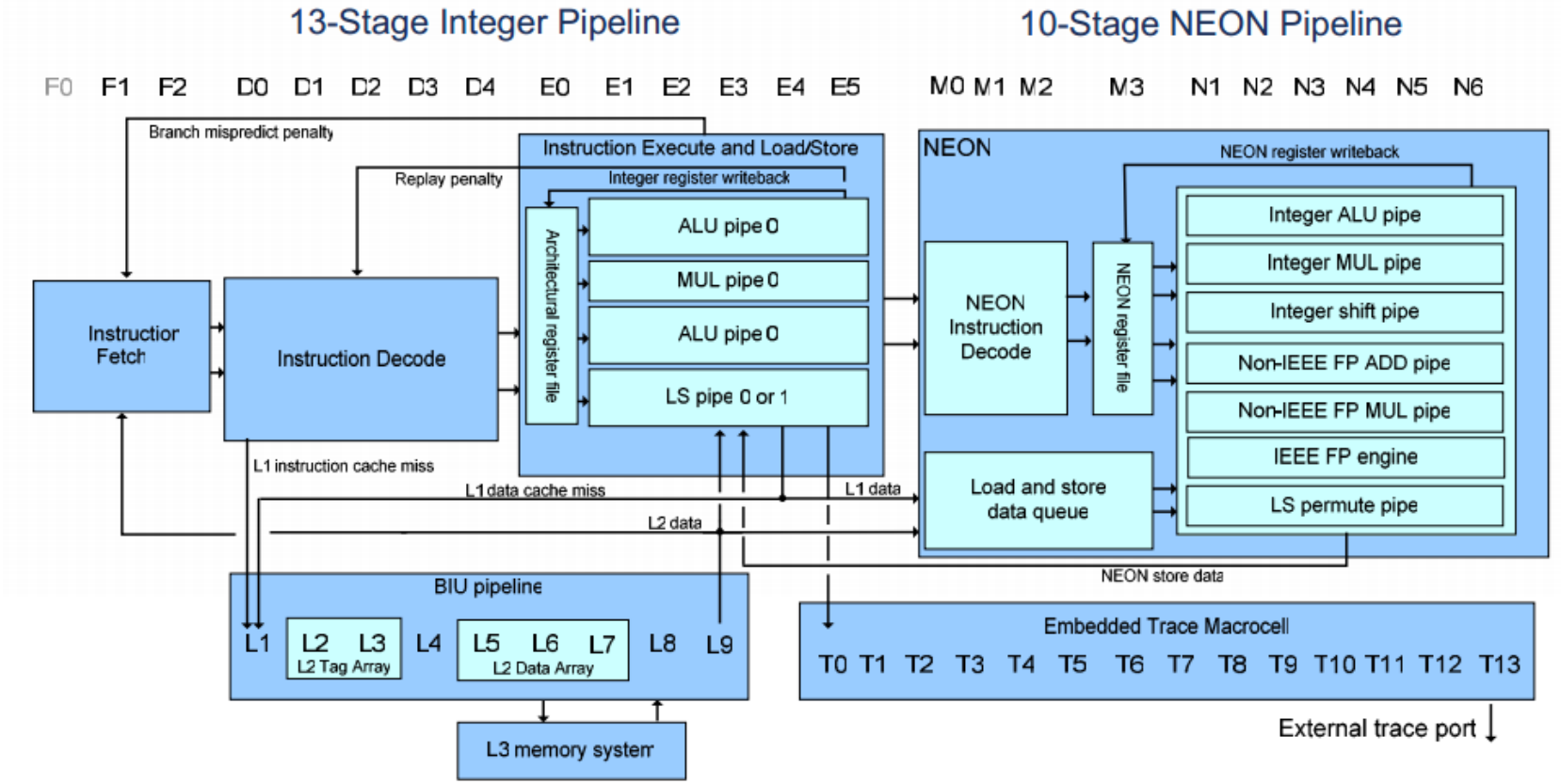
Solution:

- Break the instruction into smaller steps
- Execute each step (instead of the entire instruction) in one cycle
- Cycle time: time it takes to execute the longest step
- Keep all the steps to have similar length

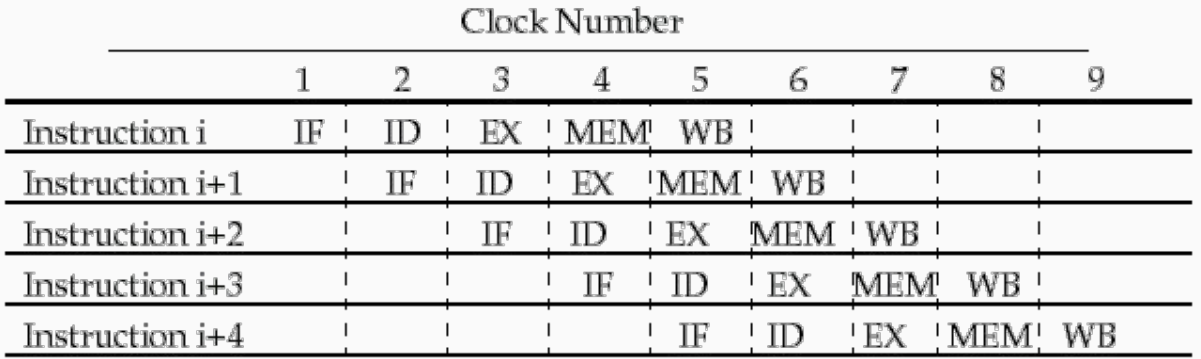
The advantages of the multiple cycle processor:

- Cycle time is much shorter
- Different instructions take different number of cycles to complete
- Allows a functional unit to be used more than once per instruction

Cortex-A8 pipelines

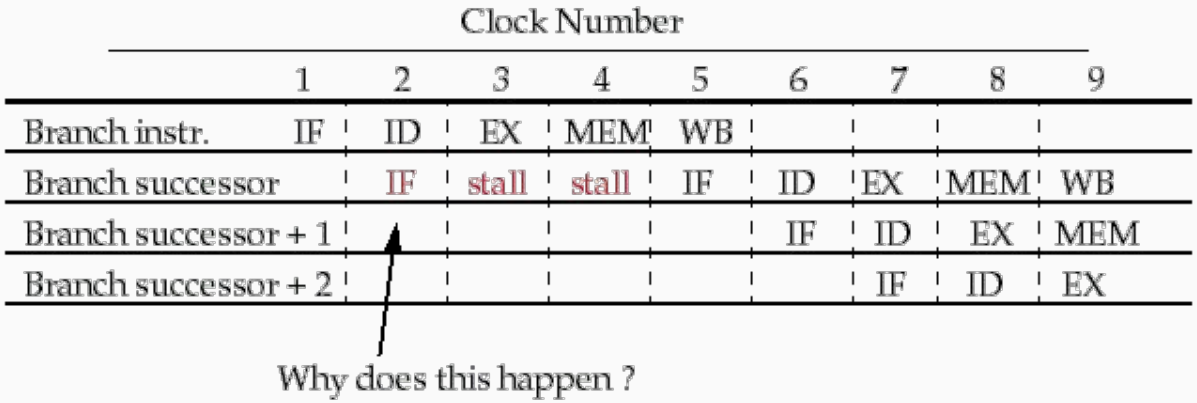


CPU pipelines



	Clock Number								
	1	2	3	4	5	6	7	8	9
Instruction i	IF	ID	EX	MEM	WB				
Instruction i+1		IF	ID	EX	MEM	WB			
Instruction i+2			IF	ID	EX	MEM	WB		
Instruction i+3				IF	ID	EX	MEM	WB	
Instruction i+4					IF	ID	EX	MEM	WB

CPU pipelines



CPU pipelines

	Clock Number																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LD F4, 0(R2)	F	D	E	M	W												
MULTD F0, F4, F6		F	D	sta	M1	M2	M3	M4	M5	M6	M7	M	W				
ADDD F2, F0, F8			F	sta	D	sta	sta	sta	sta	sta	sta	A1	A2	A3	A4	M	
SD 0(R2), F2					F	sta	sta	sta	sta	sta	sta	D	E	sta	sta	sta	M

Assumes full bypassing and forwarding.

	Clock Number										
	1	2	3	4	5	6	7	8	9	10	11
MULTD F0, F4, F6	IF	D	M1	M2	M3	M4	M5	M6	M7	M	WB
...		F	D	E	M	W					
...			F	D	E	M	W				
ADDD F2, F4, F6				F	D	A1	A2	A3	A4	M	WB
...					F	D	E	M	W		
...						F	D	E	M	W	
LD F2, 0(R2)							F	D	E	M	WB

Memory barriers

[X ==0; Y == 0]

Thread 1:

```
X = 1;

Y = 2;
```

Thread 2:

```
while (Y != 2)
        /* wait */;

if (X == 1)
        printf(“OK”);
else
        printf (“WOW!”);
```

Memory barriers

Compiler only:

- asm volatile("": : :"memory")

Hardware:

- __sync_synchronize
- atomic_thread_fence

Advanced types of memory barriers:

- Read
- Write
- Full
- SMP
- etc.

Hardware support for OS

- **Privilege levels**: user and kernel
- **Privileged instructions**: instructions available only in kernel mode
- **Memory translation**: to prevent user programs from accessing kernel data structures, and to aid in memory management
- **Exceptions**: trap to the kernel on a privilege violation or other unexpected event
- **Timer interrupts**: return control to the kernel on time expiration
- **Device interrupts**: return control to the kernel to signal I/O completion
- **System calls**: trap to the kernel to perform a privileged action on behalf of a user program
- **Return from interrupt**: switch from kernel-mode to user-mode, to a specific location in a user program

To support threads, one additional mechanism is needed:

- **Atomic instructions**: instructions to atomically read and modify a memory location, used to implement synchronization in multithreaded programs

Thanks!