

CS222

Operating Systems

Lecture 03

(Section 100001)

ผศ. ดร. กษิณิศ ชาญเชี่ยว

ckasidit@tu.ac.th



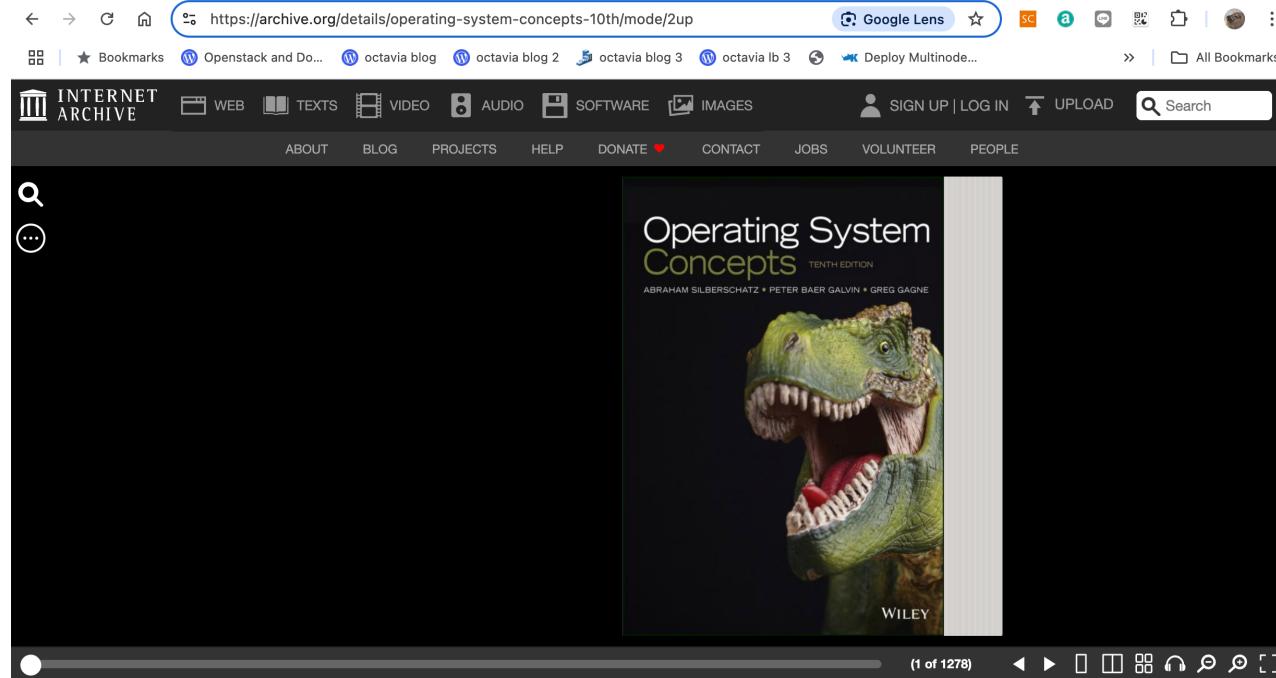
Admin

- อาทิตย์ หน้าเรียน On Site ตามปกติ (ถ้าไม่มีประกาศเรื่อง PM2.5 มห เพิ่มเติม)
- ในวันพุธที่ 5 กพ เวลา 15:00 – 16:30 เรียนที่ห้องเรียน
- ในวันศุกร์ที่ 7 กพ เวลา 15:00 – 18:00 เรียนภาคปฏิบัติ การใช้งานระบบ Linux Lab ที่ห้อง Lab 107 อาคาร บร2





Textbook



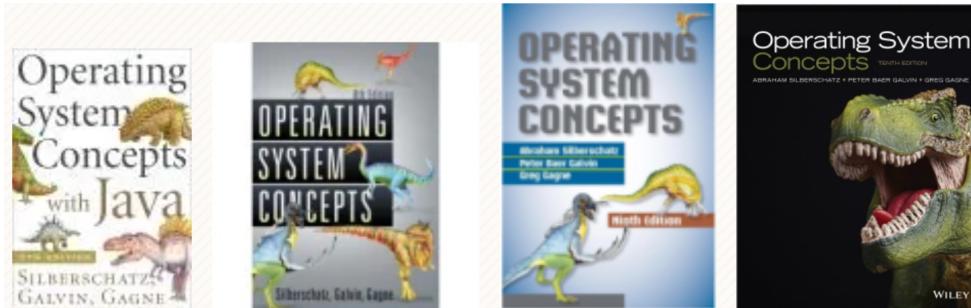
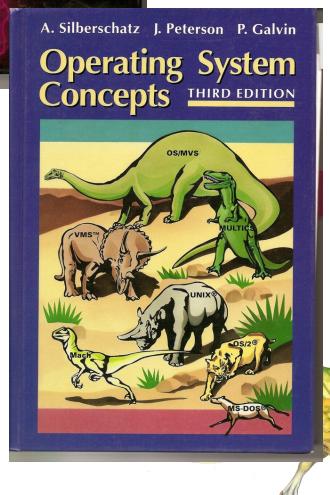
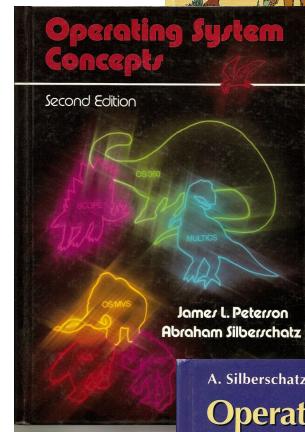
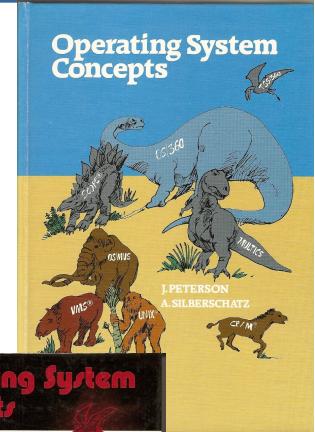
- <https://archive.org/details/operating-system-concepts-10th/mode/2up>
- Original Slides
- <https://www.os-book.com/OS10/slide-dir/index.html>





Why “Dinosaurs”? (คำถามจาก นศ)

- จากคำอธิบายของ Peter Galvin ที่นี่ ไดโนเสาร์เป็นเป็นสัญลักษณ์ของ การวิวัฒนาการของ OS และ การต่อสู้ระหว่าง OS
- Edition 1: ปี 1983 ปกจะมีไดโนเสาร์หลายตัวแต่ละตัวจะมีชื่อ OS กำกับอยู่ ตั้งแต่ OS/360 ของ IBM mainframe และ MULTICS และ UNIX บน minicomputers และ CP/M บน PC (8 bits)
- Edition หลังๆ จะไม่มีการระบุชื่อ OS



Chapter 1: Introduction

- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
- Security and Protection

Objectives

- Describe the components in a modern, multiprocessor computer system
- Illustrate the transition from user mode to kernel mode
- Discuss how operating systems are used in various computing environments

โครงสร้างของระบบคอมพิวเตอร์

- การเก็บข้อมูลในระบบคอมพิวเตอร์
- รูปแบบการประสานงานระหว่าง CPU Memory และ Storage ในระบบคอมพิวเตอร์
- การประมวลผลแบบหลาย Mode

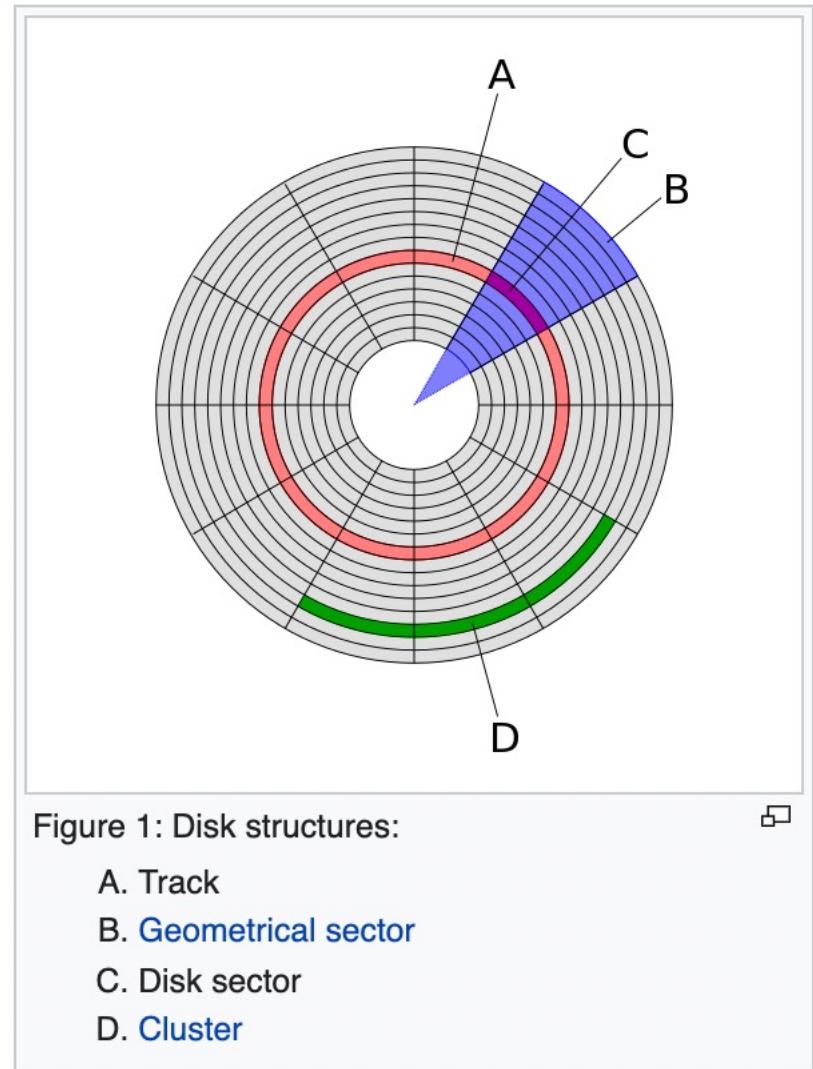
Storage Structure

Storage Structure

- Main memory – only large storage media that the CPU can access directly
 - **Random access**
 - Typically **volatile**
ลบเลื่อนได้
 - Typically **random-access memory** in the form of **Dynamic Random-access Memory (DRAM)**
 - ปิดเครื่องแล้วข้อมูลหาย
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
 - เรียกอีกอย่างว่า Persistent Storage

Storage Structure (Cont.)

- **Hard Disk Drives (HDD)** – rigid metal or glass platters covered with magnetic recording material
 - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
 - The **disk controller** determines the logical interaction between the device and the computer
- **Non-volatile memory (NVM)** devices– faster than hard disks, nonvolatile
 - Various technologies
SSD (SLC, TLC, QLC, etc), Optane DC
 - Becoming more popular as capacity and performance increases, price drops



Storage Definitions and Notation Review

The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes. A **kilobyte**, or KB , is 1,024 bytes; a **megabyte**, or **MB**, is $1,024^2$ bytes; a **gigabyte**, or **GB**, is $1,024^3$ bytes; a **terabyte**, or **TB**, is $1,024^4$ bytes; and a **petabyte**, or **PB**, is $1,024^5$ bytes. Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

หน่วยของข้อมูล

- Bit หมายถึงข้อมูลหนึ่ง ที่มีค่าที่เป็นไปได้สองค่าคือ 1 และ 0
- Byte คือชุดของข้อมูลขนาด 8 bits
- Words คือชุดของข้อมูลที่ประกอบไปด้วย N bytes
 - ถ้าคอมพิวเตอร์มีสถาปัตยกรรม 32 bits จะได้ค่า N คือ $32/8 = 4$ bytes
 - ถ้าคอมพิวเตอร์มีสถาปัตยกรรม 64 bits จะได้ค่า N คือ $64/8 = 8$ bytes
- $1 \text{ KB} = 1024^1 = 2^{10}$, $1 \text{ MB} = 1024^2 = 2^{20}$, $1 \text{ GB} = 1024^3 = 2^{30}$
- $1 \text{ TB} = 1024^4 = 2^{40}$, $1 \text{ PB} = 1024^5 = 2^{50}$

SI vs ICE standards

- International Electrotechnical Commission (IEC)
- Binary prefix
- E.g. 4 KiB, 4 GiB

Factor	Value	Prefix	
		Name	Symbol
$(2^{10})^1$	1 024	kibi	Ki
$(2^{10})^2$	1 048 576	mebi	Mi
$(2^{10})^3$	1 073 741 824	gibi	Gi
$(2^{10})^4$	1 099 511 627 776	tebi	Ti
$(2^{10})^5$	1 125 899 906 842 624	pebi	Pi
$(2^{10})^6$	1 152 921 504 606 846 976	exbi	Ei
$(2^{10})^7$	1 180 591 620 717 411 303 424	zebi	Zi
$(2^{10})^8$	1 208 925 819 614 629 174 706 176	yobi	Yi

- The International System of Units (SI)
- E.g. 4 KB, 4 GB
- ใช้ปัจปนกัน โดยทั่วไป

Factor	Value	Prefix	
		Name	Symbol
$(10^3)^1$	1 000	kilo	k
$(10^3)^2$	1 000 000	mega	M
$(10^3)^3$	1 000 000 000	giga	G
$(10^3)^4$	1 000 000 000 000	tera	T
$(10^3)^5$	1 000 000 000 000 000	peta	P
$(10^3)^6$	1 000 000 000 000 000 000	exa	E
$(10^3)^7$	1 000 000 000 000 000 000	zetta	Z
$(10^3)^8$	1 000 000 000 000 000 000 000	yotta	Y

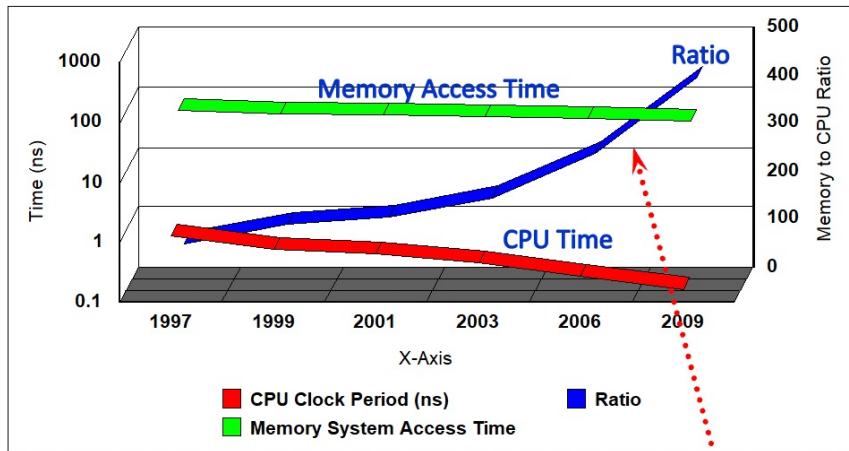
อ้างอิงจาก <https://digilent.com/blog/mib-vs-mb-whats-the-difference/>

Storage Hierarchy

- สาเหตุของการมีลำดับชั้นของการเก็บข้อมูล สืบเนื่องมาจากความแตกต่างของอุปกรณ์เก็บข้อมูลใน สามประเด็นคือ ความเร็วในการเข้าถึงข้อมูล ราคา และความคงอยู่ของข้อมูล
- Storage systems organized in hierarchy
 - Speed
 - Cost
 - Volatility
- **Caching** – copying information into **faster storage system**; main memory can be viewed as a cache for secondary storage
 - จะได้กล่าวถึงต่อไป
- **Device Driver** for each device controller to manage I/O
 - Provides uniform interface between controller and kernel

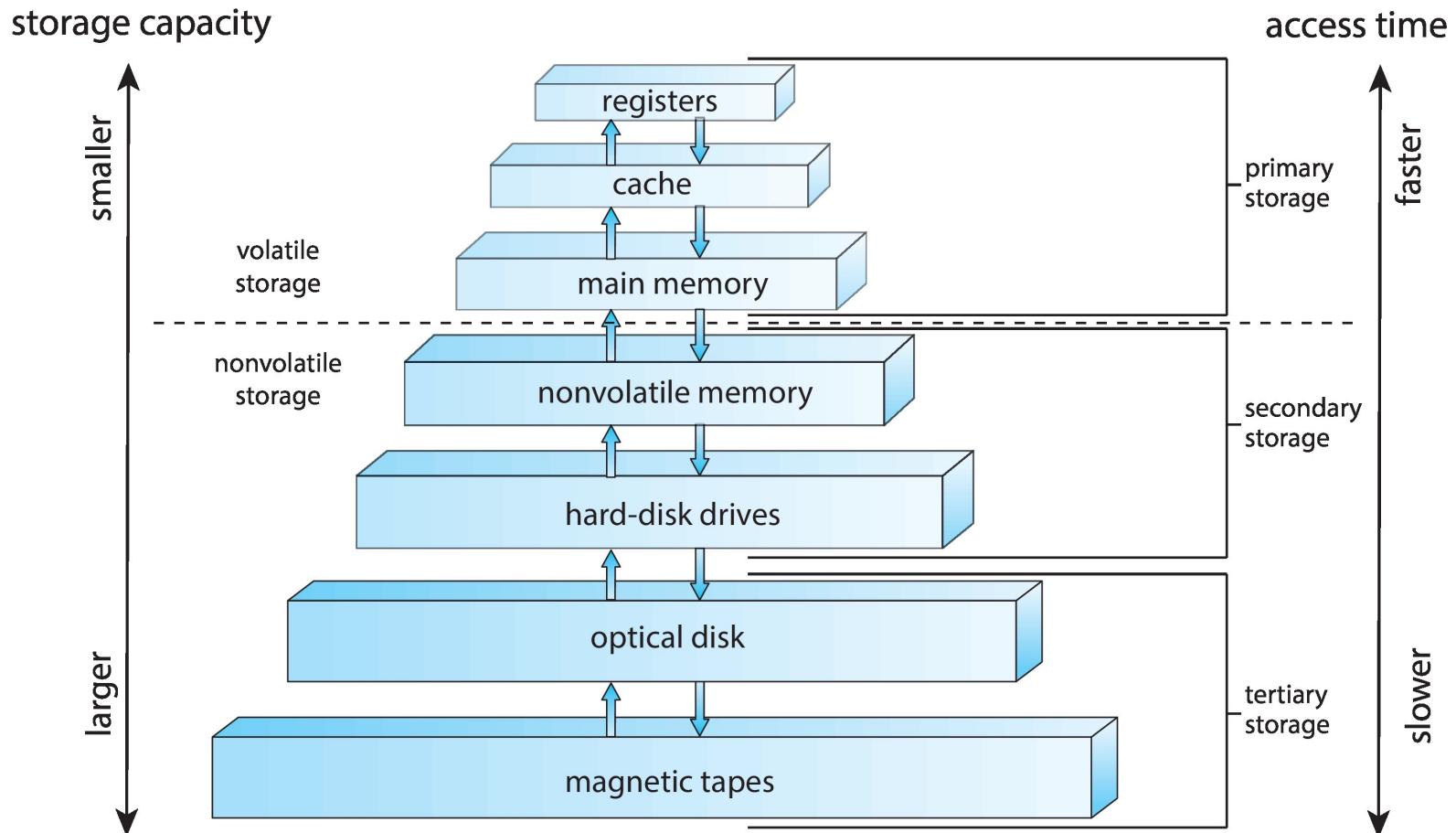
ผลจากการแตกต่างกันของความเร็วในการเข้าถึงข้อมูล

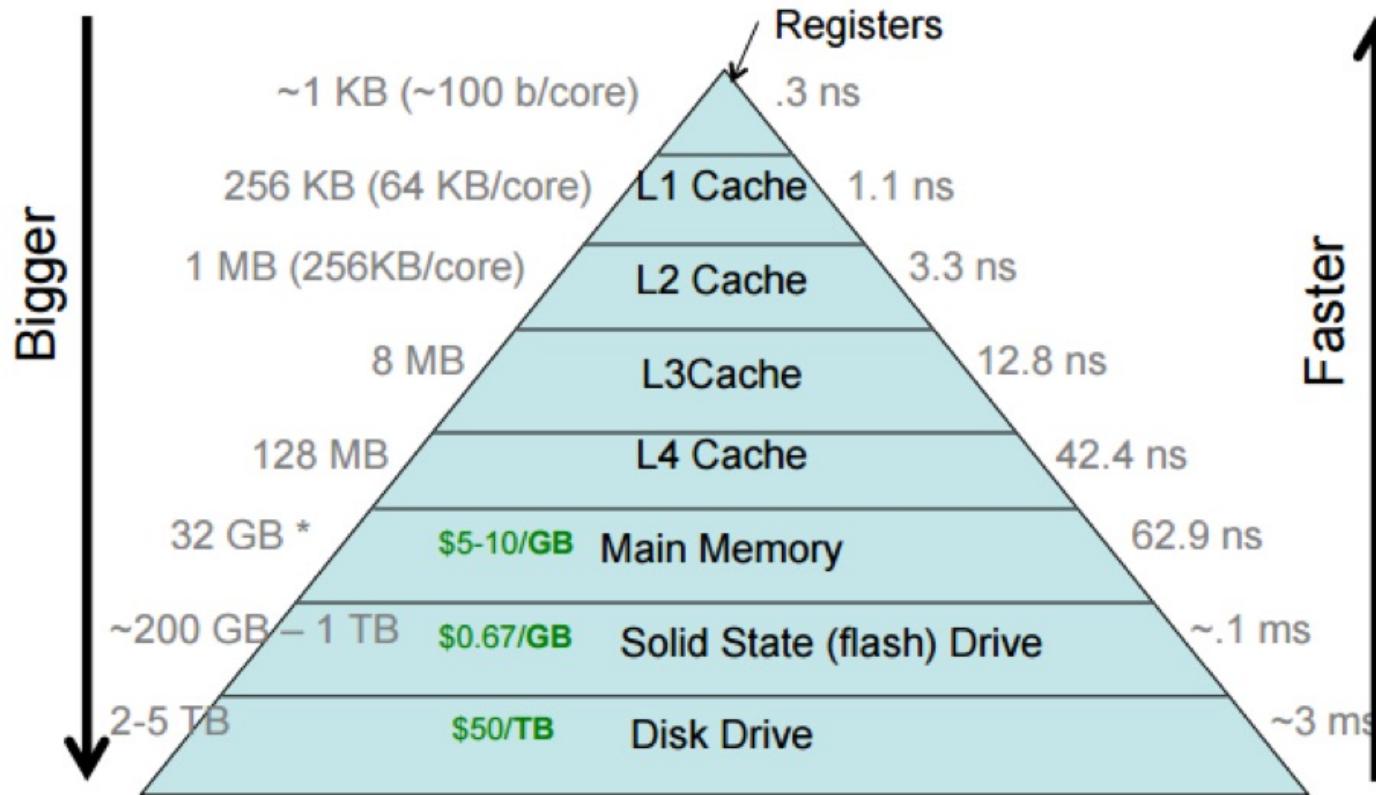
Memory Wall



- พัฒนาการของซีพียู - เวลาในการประมวลผลคำสั่งหนึ่งคำสั่งลดลง
- พัฒนาการของหน่วยความจำ —เวลาในการเข้าถึงข้อมูลจากซีพียูไม่ลดลงมากเหมือนซีพียู
- ทำให้อัตราส่วนของความแตกต่างมากขึ้น
- จำเป็นต้องมี หน่วยความจำแครช cache และหลักการ caching เข้ามาช่วย
- เกิดลำดับชั้นตามความเร็ว

Storage-Device Hierarchy





ที่มา

<https://cs.brown.edu/courses/csci1310/2020/assign/labs/lab4.html?spm=a2c65.11461447.0.0.47497f65vJCjMC>
https://www.alibabacloud.com/blog/the-mechanism-behind-measuring-cache-access-latency_599384





เปรียบเทียบเวลาเข้าถึงอุปกรณ์ (เล่นๆ)

- สมมุติว่า 1 ns เทียบเท่ากับเวลา 1 นาที
- Register: เวลา **.3 ns** ก็เปรียบเหมือน .3 นาที = 18 วินาที นศ เดินไปห้องเรียนข้างๆ
- L1 Cache: **1.1 ns** ประมาณ 1 นาที เดินไปห้องแล็บที่ บร 2
- L2 Cache: **3.3 ns** หรือ 3 นาที เดินไปสหกร มช
- Memory: **62.9 ns** หรือ 62 นาที (1 ชม) ขึ้นรถเมล์+ไฟฟ้าไป Siam
- SSD: 1 ms = 1,000,000 ns ดังนั้น **0.1 ms** = 100,000 ns เทียบกับ 1 แสนนาที = 69 วัน และ 10 ชม = 2 เดือน 9 วัน (sail เรือใบข้ามมหาสมุทร atlantics)
- Disk: **3 ms** = 3,000,000 ns เทียบกับ 3 ล้านนาที = 2083 วัน = 5 ปี 8 เดือน 18 วัน (ขึ้นจรวดไป Saturn)

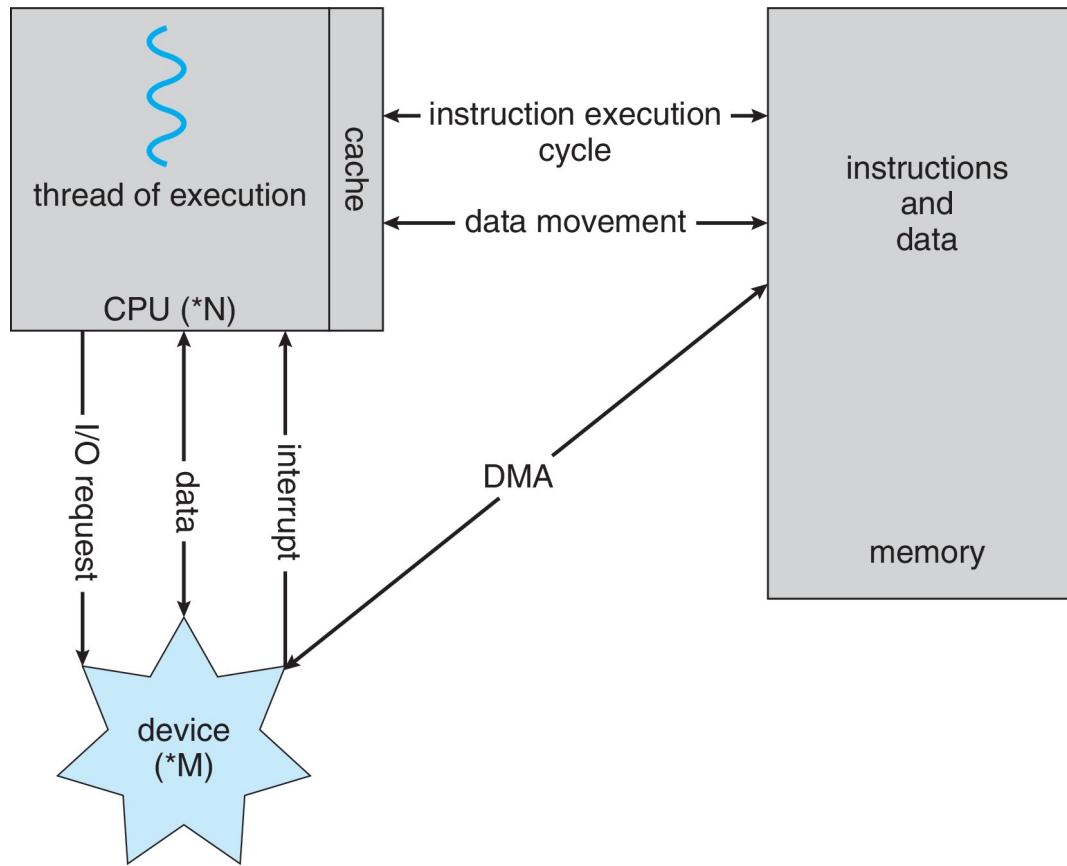
เว็บแปลงเวลา [Convert Milliseconds to Nanoseconds \(calculateme.com\)](http://Convert Milliseconds to Nanoseconds (calculateme.com))



รายละเอียดการประสานงานของ CPU Memory และ I/O

- ที่ผ่านมาเราพูดถึงขั้นตอนในการทำงานแบบที่มี Interrupts เกี่ยวข้องกับการที่ CPU เข้าถึงข้อมูลจาก I/O อย่างคร่าวๆ
- ในอันดับถัดไปเราจะพิจารณาการประสานงานที่ละเอียดขึ้น
- การทำงานแบบ Interrupt Driven I/O นั้น โอบค์สำหรับ กรณีซึ่งพิจัยเข้าถึงข้อมูลขนาดเล็กจากอุปกรณ์ I/O
- แต่จะเสียเวลามากเมื่อโปรแกรมต้องการอ่านข้อมูลขนาดใหญ่
- แนะนำ Direct Memory Access (DMA) เข้ามาช่วยแก้ปัญหานี้

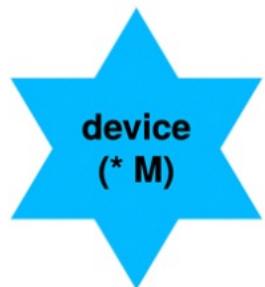
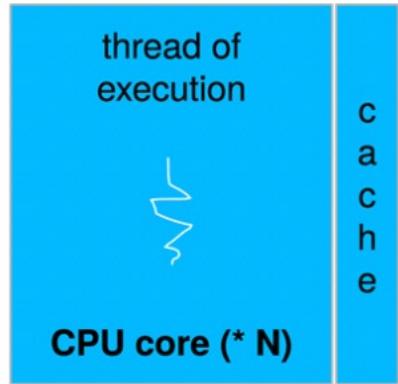
How a Modern Computer Works

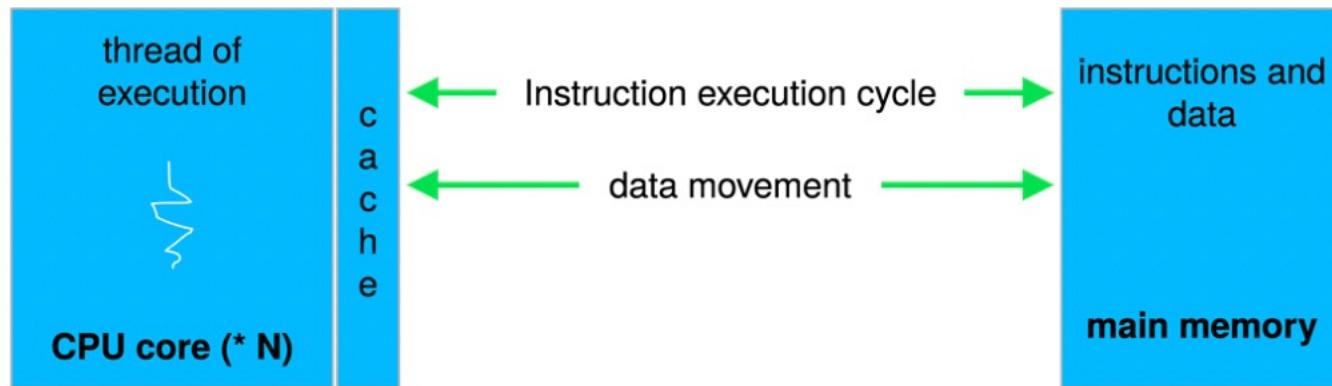


A von Neumann architecture

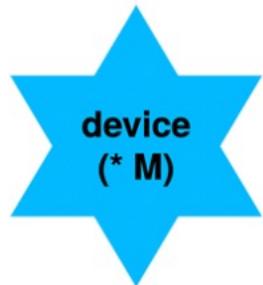
รายละเอียดการประสานงานของ CPU Memory และ I/O

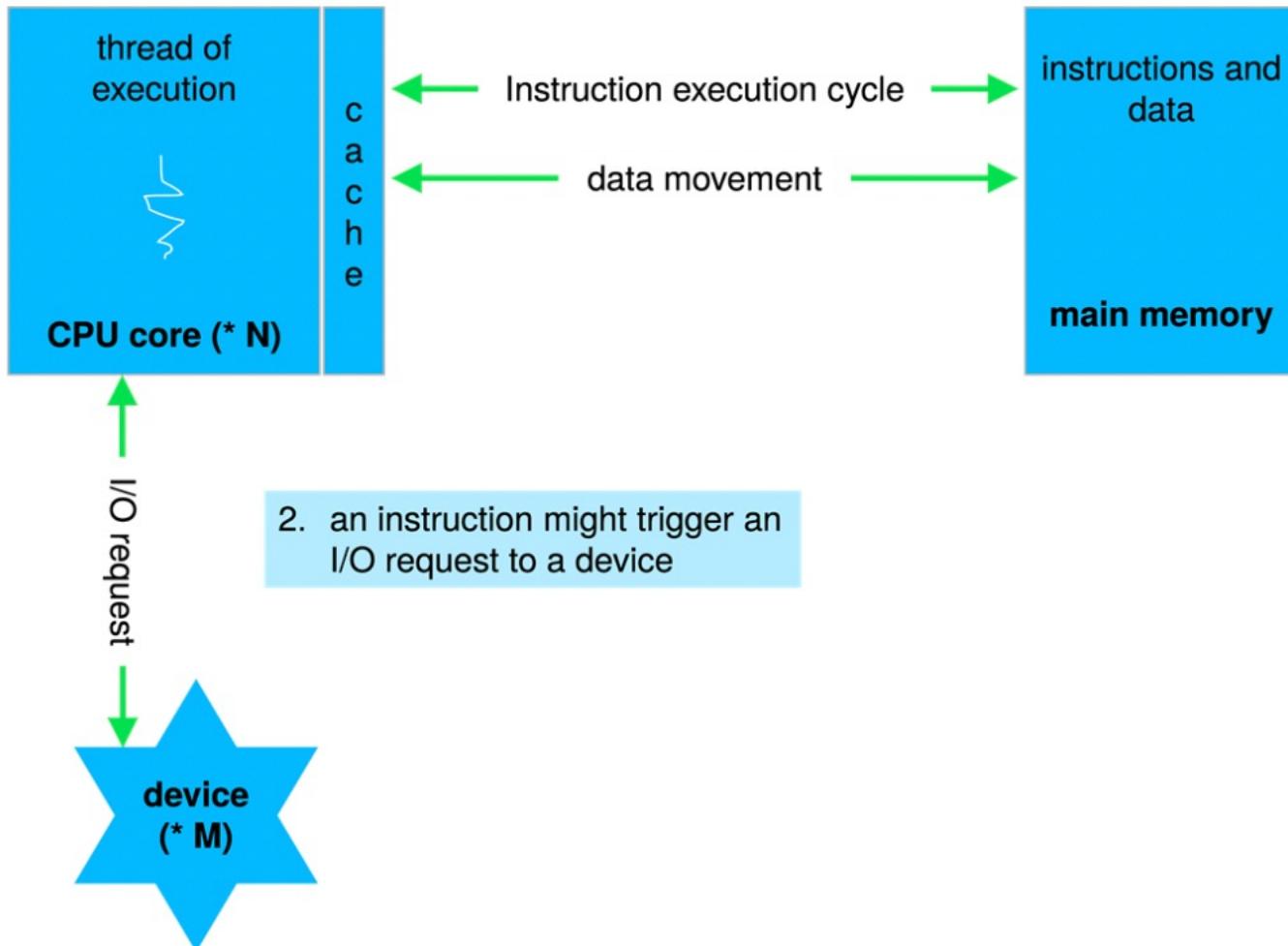
- ที่ผ่านมาเราพูดถึงขั้นตอนในการทำงานแบบที่มี Interrupts เกี่ยวข้องกับการที่ CPU เข้าถึงข้อมูลจาก I/O อย่างคร่าวๆ
- ในอันดับถัดไปเราจะพิจารณาการประสานงานที่ละเอียดขึ้น
- Interrupt Driven I/O
- **แนะนำ Direct Memory Access (DMA)**

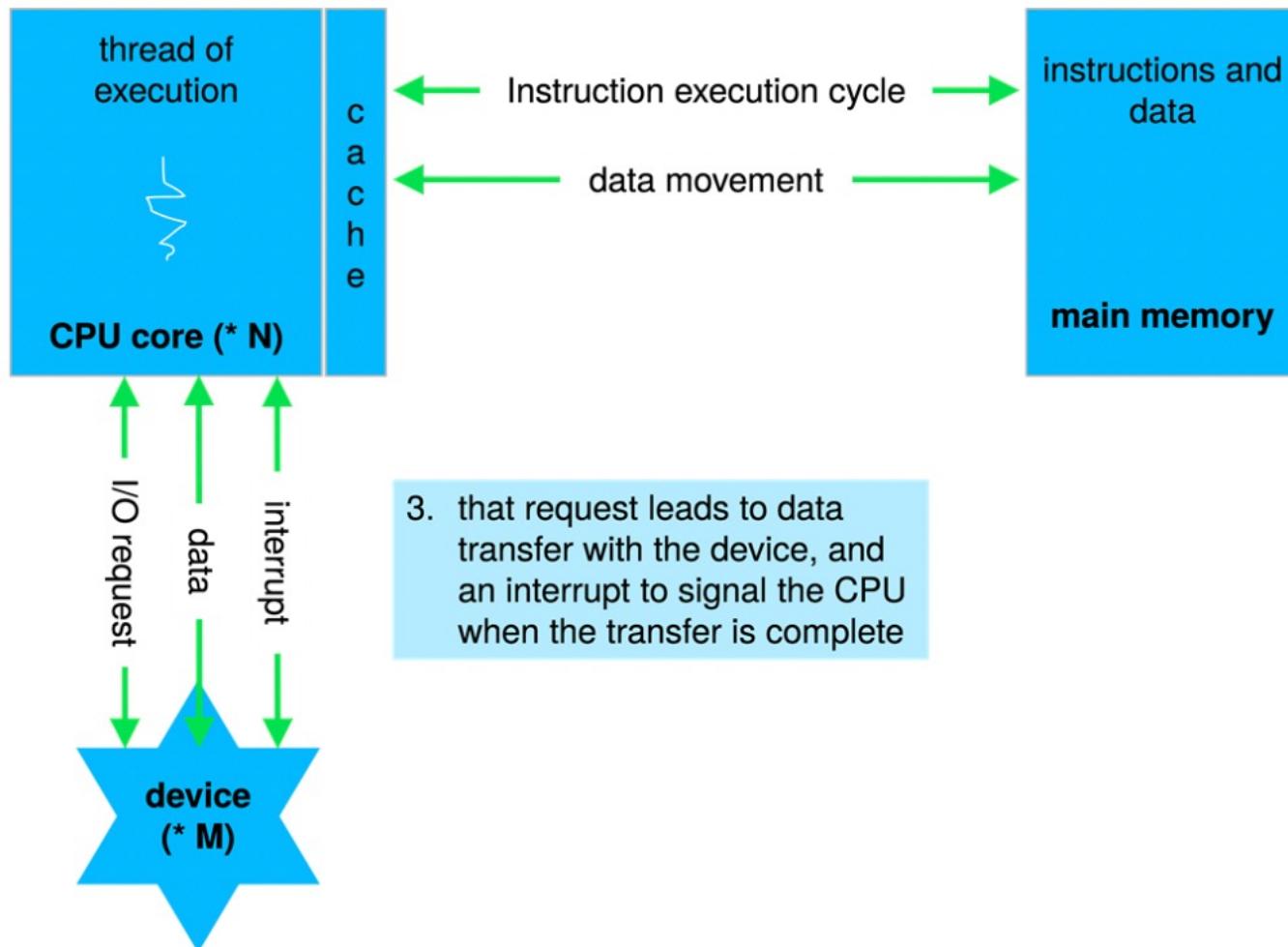




1. the CPU core instruction cycle involves reading instructions and reading and writing data to main memory







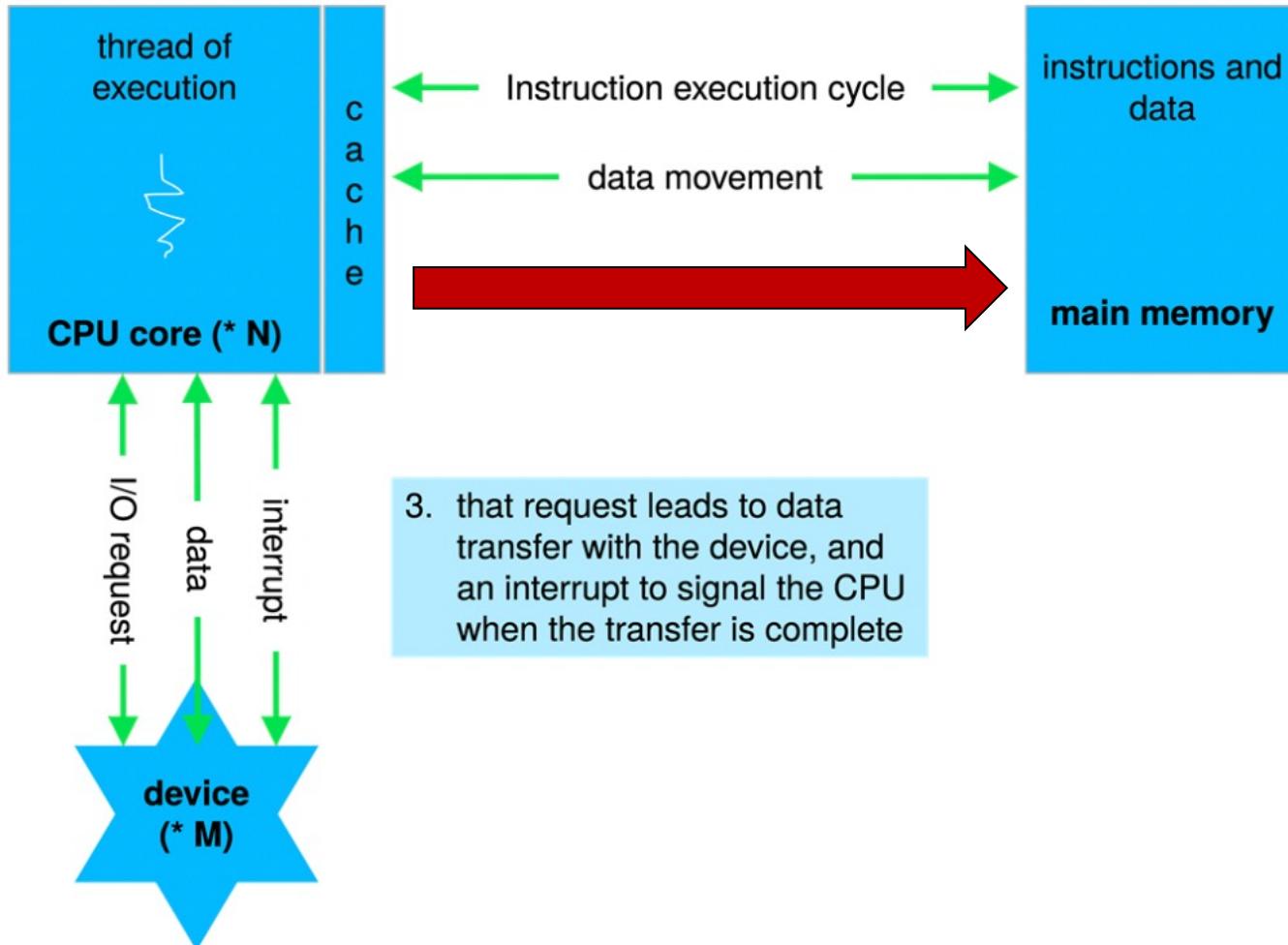
เปรียบเทียบประสานงานของ CPU Memory และ I/O

- สมมุติว่า นศ เป็นผู้จัดการ (CPU) และสั่งให้ลูกน้อง (I/O device) ทำงานที่เขานัด และเมื่อได้ผลมาแล้ว นศ จะเอาผลงานที่ลูกน้องทำไปเก็บในตู้เก็บของ
- ขั้นแรก นศ ก็จะสั่งให้ลูกน้องทำงาน และ นศ ไปทำอย่างอื่น
- ลูกน้องตะโภนแจ้งเมื่องานเสร็จและ นศ หยุดงานที่ทำชั่วคราวและมาเอาผลงานของลูกน้องไปเก็บในตู้ เสร็จแล้วก็กลับมาทำงานที่ค้างต่อ
- ปัญหา: ถ้าผลงานของลูกน้องมีจำนวนมาก นศ อาจต้องเดินทางรอบเพื่อมาหยิบเอาผลงานของลูกน้องไปเก็บในตู้ (ทำให้ นศ เสียเวลาต้องมาทำงาน labour นี้เอง)
- แต่ถ้า นศ มี ผู้ช่วยผู้จัดการ (DMA) นศ (CPU) ก็สามารถสั่งให้ ผู้ช่วยฯ (DMA) ไปสั่งงานลูกน้อง และผู้จัดการไปทำงานอื่น
- เมื่อลูกน้องทำเสร็จก็ให้ ผู้ช่วยฯ ขนผลงานที่ได้ไปเก็บในตู้ จนกระทั่ง xn เสร็จแล้ว จึงค่อยตะโภนบอก ผู้จัดการ

การประสานงานของ CPU Memory และ I/O (แบบไม่ใช้ DMA)

- วิธีการนี้เรียกว่า Programmed-I/O (PIO)
- หลังจากที่ซีพียู ส่ง I/O request ไปให้อุปกรณ์ มันก็อาจหันไปประมวลผลคำสั่งอื่น (หรือรอถ้าไม่มีงานอื่นทำ)
- เมื่อได้รับ Interrupt จากอุปกรณ์ I/O ซีพียูก็จะประมวลผลคำสั่ง Interrupt Handler และ
- อ่านข้อมูลจาก local buffer ของ I/O controller เข้าสู่ registers ของซีพียู
- ซีพียุรัน Interrupt Handler เพื่อนำค่าที่อ่านได้มาประมวลผล หรืออาจนำไปเก็บในหน่วยความจำ (เช่น อ่านค่ามาไปเก็บในตัวแปรของโปรแกรม)
- ในกรณีที่ข้อมูลมีขนาดใหญ่ ซีพียูจะต้องอ่านข้อมูลจากอุปกรณ์โดยลายรอบในการถ่ายโอนข้อมูลจาก local buffers ไปสู่หน่วยความจำ

ແບບໄຟໃໝ່ DMA



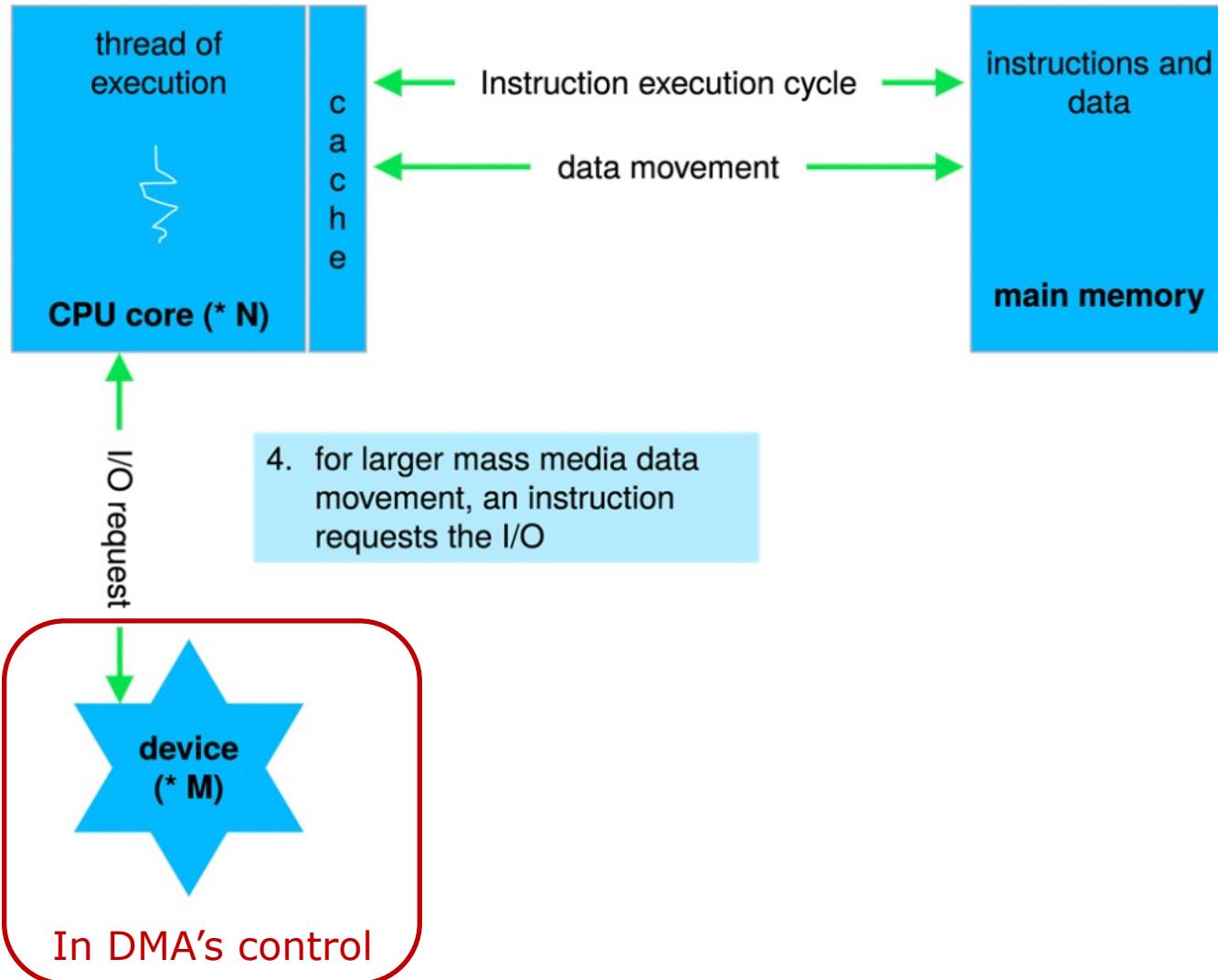


การประสานงานของ CPU Memory และ I/O (แบบใช้ DMA)

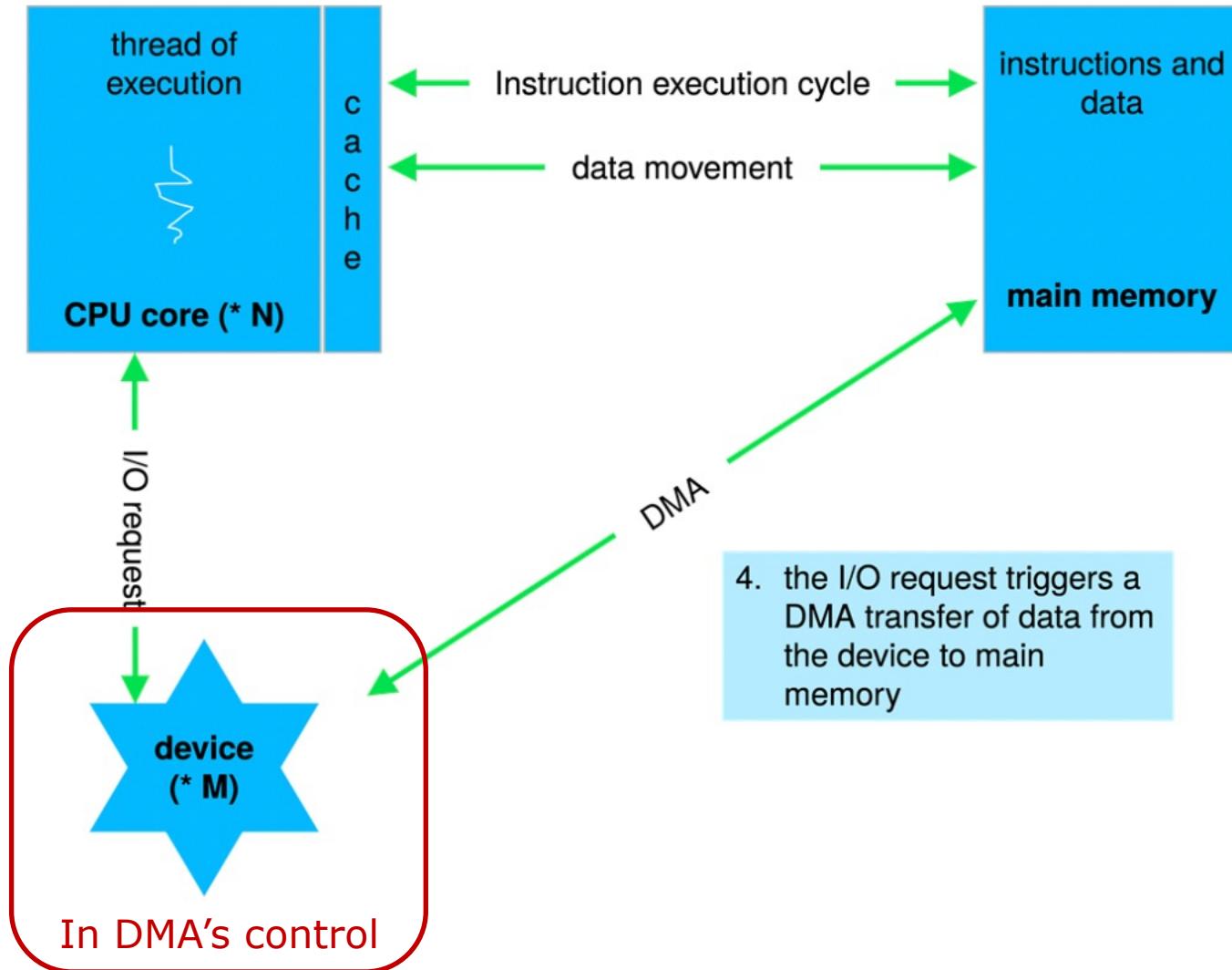
- Direct Memory Access (DMA) คือ Special Purpose Processor ที่ทำหน้าที่ประสานงานกับซีพียูและ Device Controller เพื่อถ่ายโอนข้อมูลจาก Device ไปยังหน่วยความจำโดยตรง แล้วส่งสัญญาณ Interrupt เแจ้งซีพียู เมื่อถ่ายโอนเสร็จสิ้น
- ซีพียู (โดย Device Driver) จะส่งคำสั่ง IO Request ให้ Device Controller แล้วซีพียูไปทำอย่างอื่น
- เมื่อ Device Controller ทำงานและมีข้อมูลใน buffer (หรือ Device memory) พร้อมให้ถ่ายโอน มันจะสั่งให้ DMA เริ่มทำงาน (initiate) โดยที่ DMA จะเข้าถึงหน่วยความจำในตำแหน่ง memory address ที่ต้องการ แล้ว DMA Controller จะถ่ายโอนข้อมูลจาก buffer ไปที่หน่วยความจำ
- เมื่อถ่ายโอนเสร็จมันจะส่งสัญญาณ Interrupt ไปแจ้งซีพียู



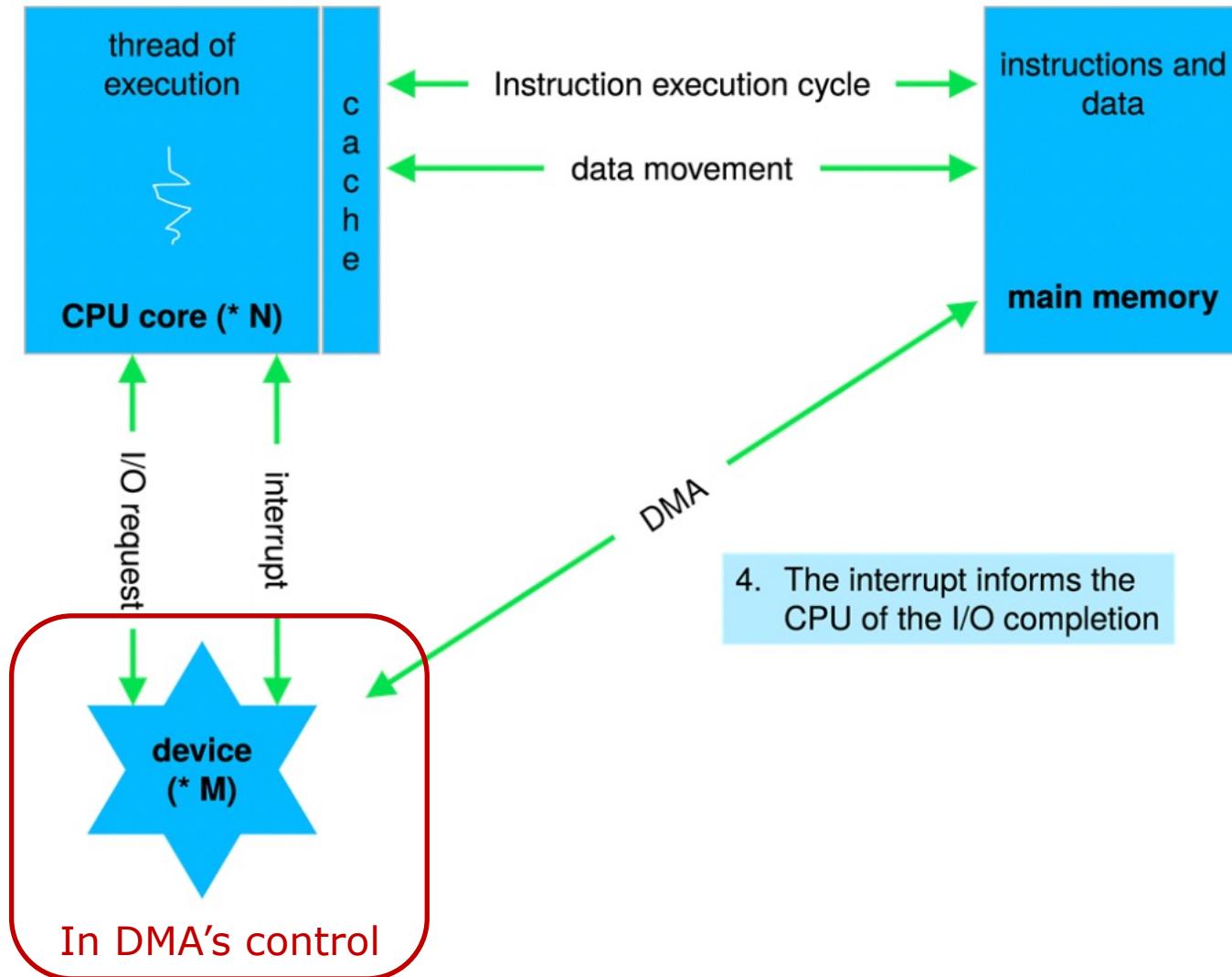
ແບບໃໝ່ DMA



ແບບໃໝ່ DMA



ແບບໃໝ່ DMA



Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
 - ประโยชน์ของ DMA
- Only one interrupt is generated per block, rather than the one interrupt per byte



Computer System Architecture

- Single-Processor System คือคอมพิวเตอร์ที่มีซีพียูเดียว และซีพียูมี core ซึ่งหมายถึงวงจรที่ดำเนินการสร้างเส้นทาง (Flow) ของการประมวลผลเส้นทางเดียว
- ซีพียู ในระบบ Single-Processor System จะสั่งงานอุปกรณ์ฮาร์ดแวร์อื่นในระบบคอมพิวเตอร์ เช่นระบบควบคุม Disk ระบบควบคุม Keyboard ระบบควบคุม Graphic Display แต่ ระบบควบคุมเหล่านี้ไม่ถือว่าเป็น Processor เพราะมันถูกออกแบบมาให้ทำงานเฉพาะ (Special Purpose Computer)
 - ซีพียูเปรียบเหมือน สมอง ในขณะที่ special purpose processor เปรียบเหมือนอวัยวะที่ทำงานของมันโดยอัตโนมัติ (ซีพียูไม่สามารถสั่งได้ทุกอย่าง)





Computer System Architecture

- Multiprocessor System คือคอมพิวเตอร์ที่มีชิปปี้แบบ single core CPU หลายชิปปี้
- Multiprocessor system แบบที่เรียกวันทั่วไปว่า Symmetric Multi-processor (SMP)
- เข้าถึง Mem แบบ **Uniform Mem Access** คือเข้าถึงได้เร็วเท่ากันไม่ว่าจะมาจาก processor ไหน

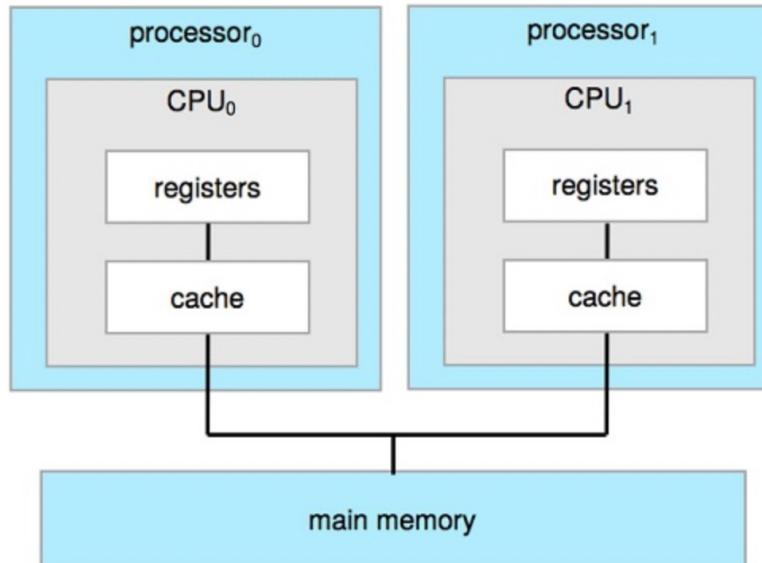


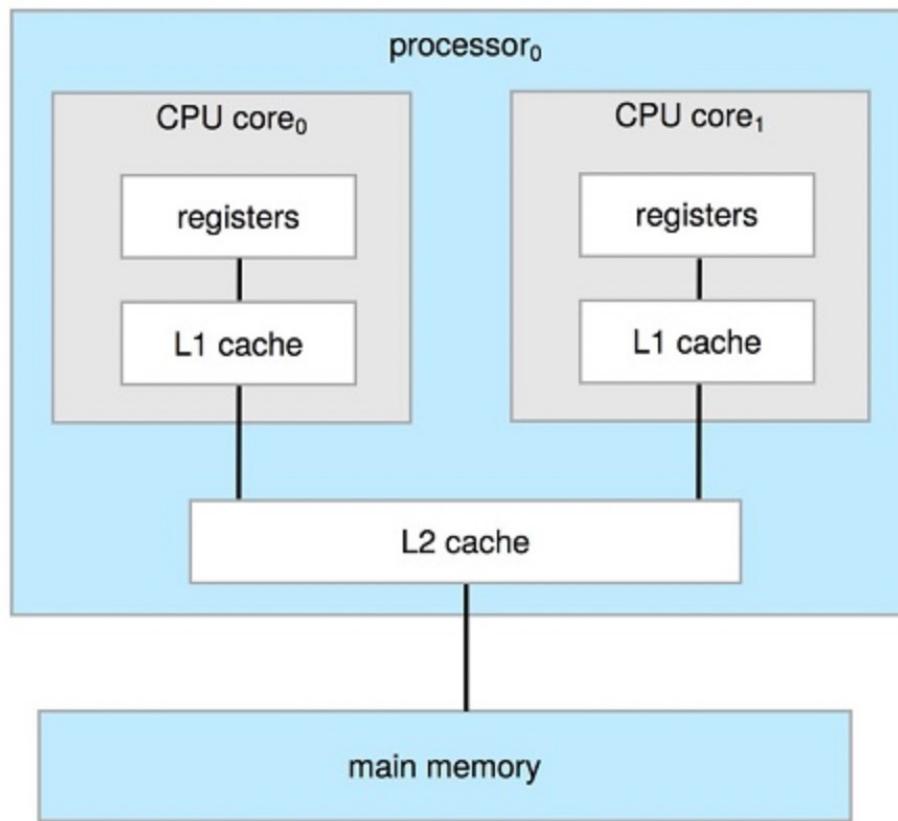
Figure 1.8 Symmetric multiprocessing architecture.





Computer System Architecture

- ระบบ Multicore คือระบบที่สร้างแกนการประมวลผลหลายแกนบนวงจร Chip เดียวกัน





Computer System Architecture

DEFINITIONS OF COMPUTER SYSTEM COMPONENTS

- **CPU**—The hardware that executes instructions.
- **Processor**—A physical chip that contains one or more CPUs.
- **Core**—The basic computation unit of the CPU.
- **Multicore**—Including multiple computing cores on the same CPU.
- **Multiprocessor**—Including multiple processors.

Although virtually all systems are now multicore, we use the general term **CPU** when referring to a single computational unit of a computer system and **core** as well as **multicore** when specifically referring to one or more cores on a CPU.





Computer System Architecture

■ ระบบคอมพิวเตอร์แบบ Non-Uniform

Memory Access (NUMA) คือระบบ

Multiprocrssor แบบที่

- แต่ละ Processor มี Local Memory
- Processor เข้าถึง Memory ของ Processor อื่นได้
- Processor ใช้เวลาเข้าถึง Local Memory ได้เร็วกว่า Remote Memory

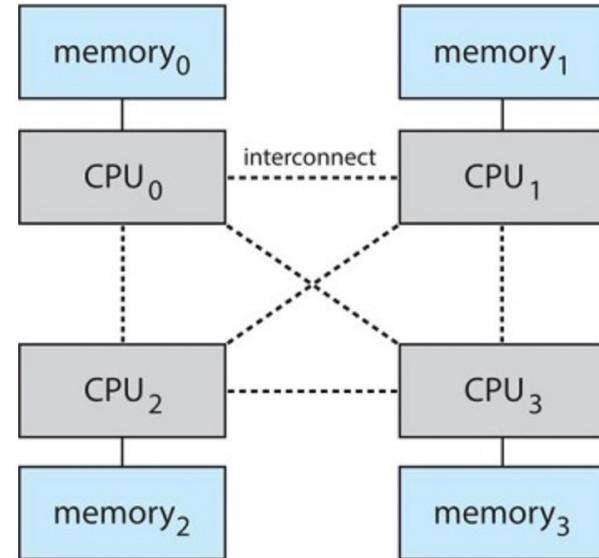


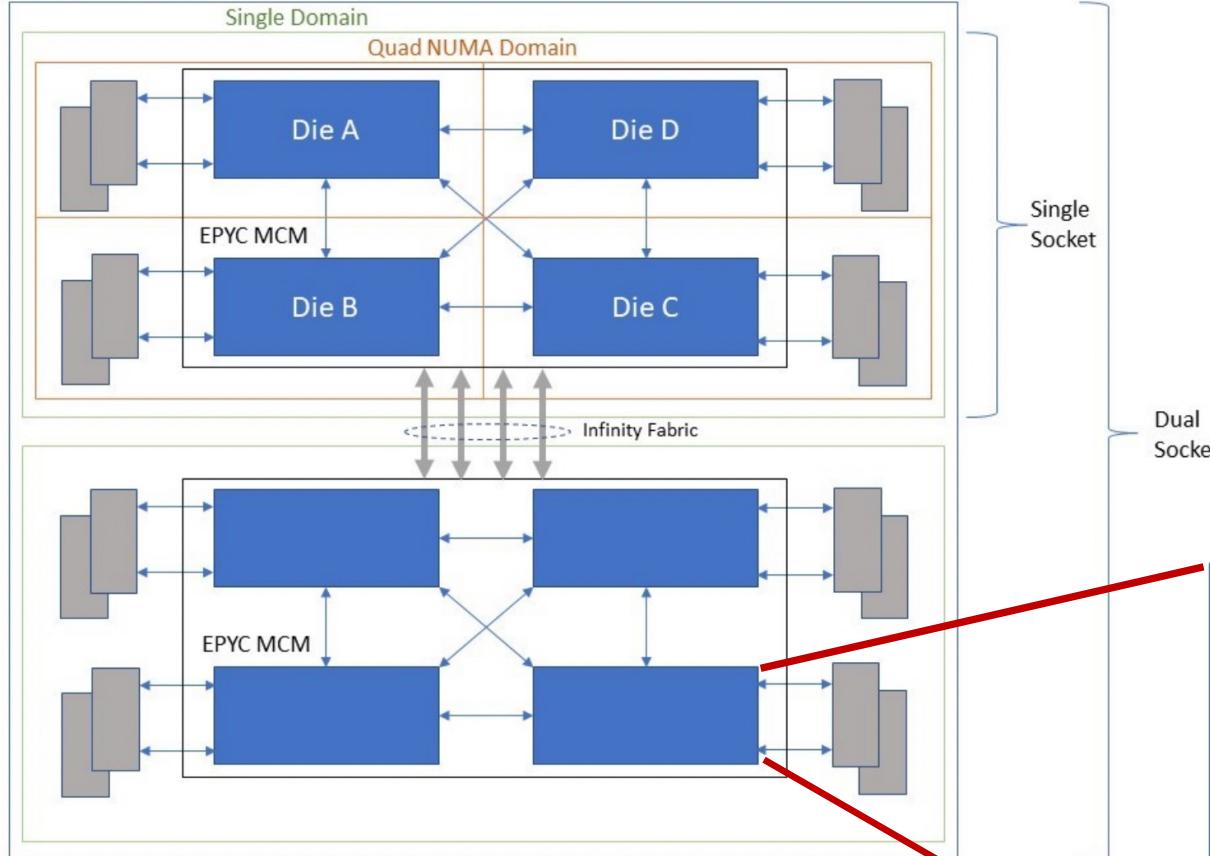
Figure 1.10 NUMA multiprocessing architecture.





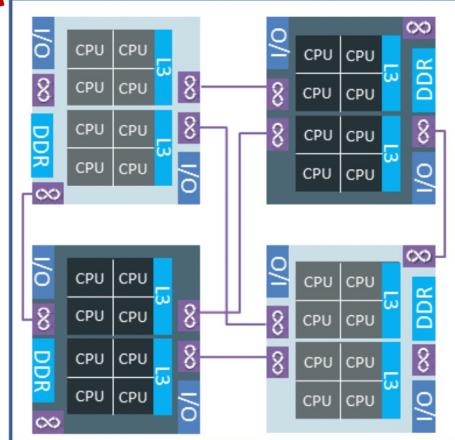
Computer System Architecture

Dual Socket (2P) Domain



Sources: AMD & TIRIAS Research

EPYC MCM



4 x 8C Die Cost

0.59X¹





AMD EPYC vs Intel Xeon (สำหรับ servers)

AMD EPYC CPU codenames^{[15][16][17][18][19]}

Segment	Gen	Year	Name	Product line	Cores	Socket	
Server	1st	2017	Naples	7001 series	32x Zen	SP3 (LGA)	
	2nd	2019	Rome	7002 series	64x Zen 2		
	3rd	2021	Milan	7003 series	64x Zen 3		
			Milan-X				
	4th	2022	Genoa	9004 series	96x Zen 4	SP5 (LGA)	
			Genoa-X				
		2023	Bergamo		128x Zen 4c		
			Siena	8004 series	64x Zen 4c	SP6 (LGA)	
	5th	2024	Turin	TBA	Zen 5	TBA	

Sapphire Rapids-HBM (High Bandwidth Memory/Xeon Max Series) [edit]

Xeon Max processors contain 64 GB of [High Bandwidth Memory](#).

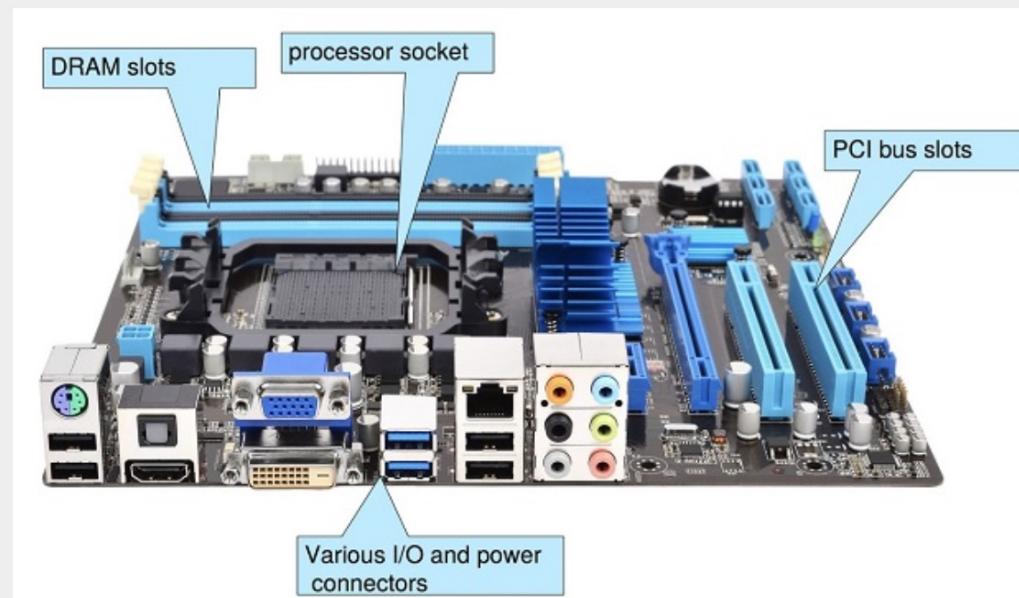
Model number	Cores (Threads)	Base clock	All core turbo boost	Max turbo boost	Smart Cache	TDP	Maximum scalability	Registered DDR5 w. ECC support	UPI links	Release MSRP (USD)
9480	56 (112)	1.9 GHz	2.6 GHz	3.5 GHz	112.5 MB	350 W	2S	4800 MT/s	4	\$12980
9470	52 (104)	2.0 GHz	2.7 GHz		105.0 MB					\$11590
9468	48 (96)	2.1 GHz	2.6 GHz		97.5 MB					\$9900
9460	40 (80)	2.2 GHz	2.7 GHz		75.0 MB				3	\$8750
9462	32 (64)	2.7 GHz	3.1 GHz							\$7995





PC Motherboard

Consider the desktop PC motherboard with a processor socket shown below:

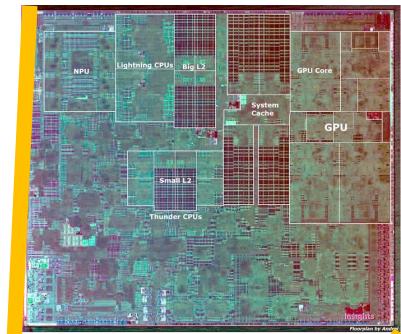
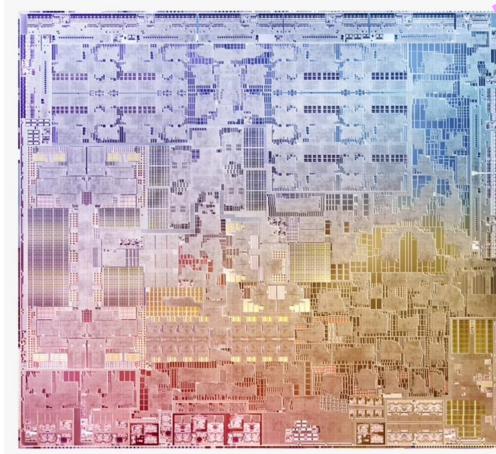
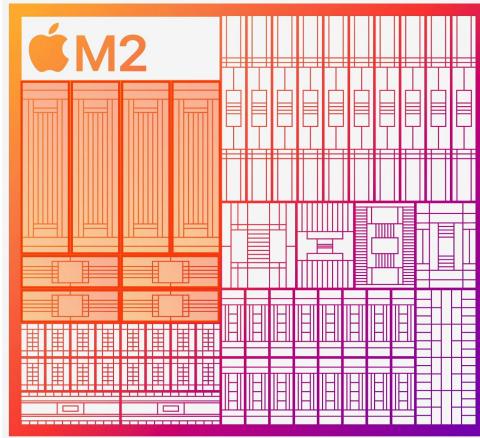


This board is a fully functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

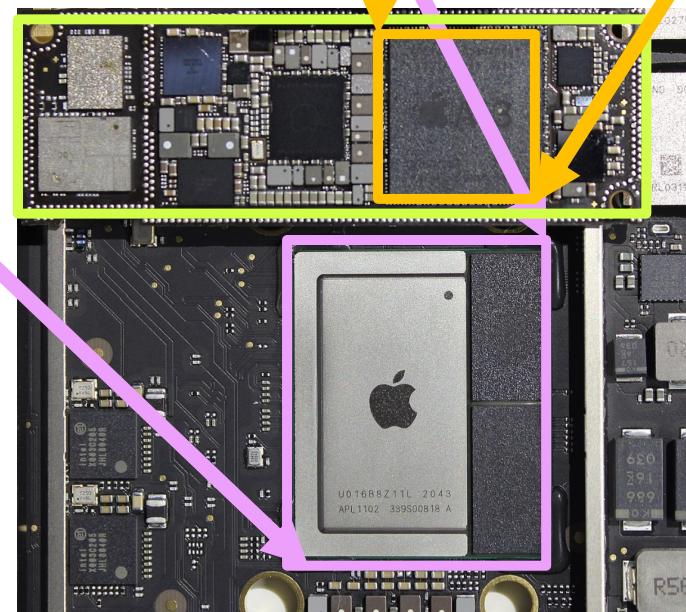




System on Chip (SoC)



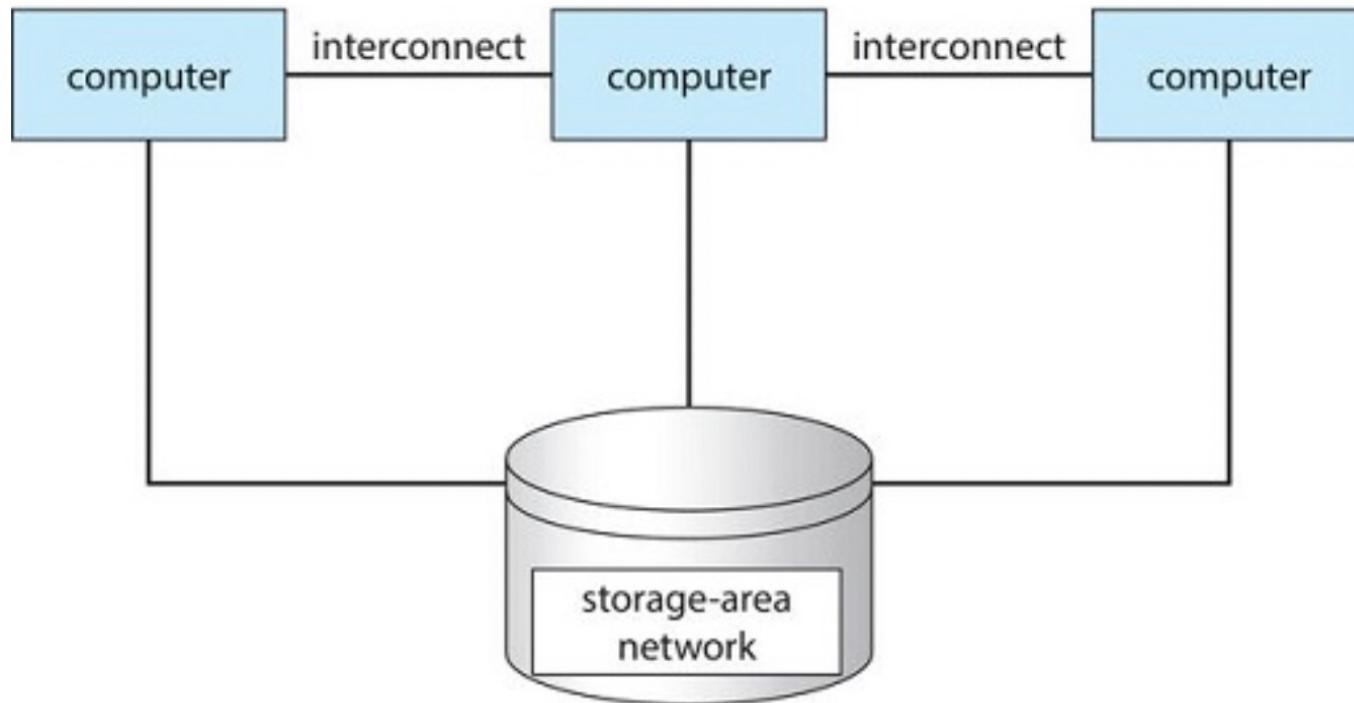
ระบบคอมพิวเตอร์แบบซิสเต็มออนชิป ยกระดับอย่าง เช่น Apple Silicon M1 และ M2 และ A13/A15 Bionic รวม CPU และ GPU และ NPU (ใช้สำหรับ face recognition, background blurring, Games) และ Mem Controller ใน Chip เดียวกัน (ไม่รวม RAM) แทนที่จะแยกอุปกรณ์เหล่านี้และเชื่อมต่อบน Motherboard





Computer System Architecture

- Cluster System คือระบบที่ประกอบไปด้วย คอมพิวเตอร์แบบ multiprocessor หลายเครื่อง เชื่อมต่อกันด้วย **high speed network** และ shared storage system ผ่านระบบเครือข่าย (Storage Area Network)



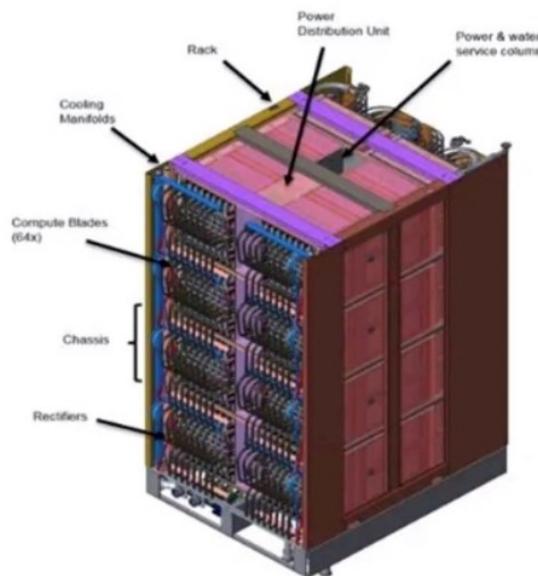


Massively Parallel Computers



System

- 2 EF Peak DP FLOPS
- 74 compute racks
- 29 MW Power Consumption
- 9,408 nodes
- 9.2 PB memory
(4.6 PB HBM, 4.6 PB DDR4)
- Cray Slingshot network with dragonfly topology
- 37 PB Node Local Storage
- 716 PB Center-wide storage
- 4000 ft² foot print



All water cooled, even DIMMs and NICs

AMD node

- 1 AMD "Trento" CPU
- 4 AMD MI250X GPUs
- 512 GiB DDR4 memory on CPU
- 512 GiB HBM2e total per node
(128 GiB HBM per GPU)
- Coherent memory across the node
- 4 TB NVM
- GPUs & CPU fully connected with AMD Infinity Fabric
- 4 Cassini NICs, 100 GB/s network BW

Compute blade

- 2 AMD nodes

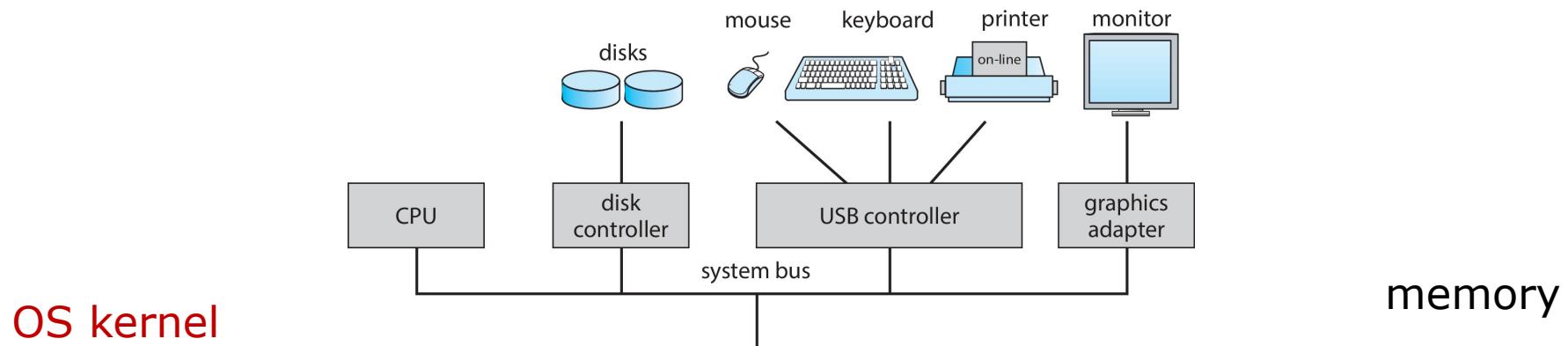


<https://www.nextplatform.com/2022/03/28/early-frontier-supercomputer-tests-show-decent-performance-leaps/>

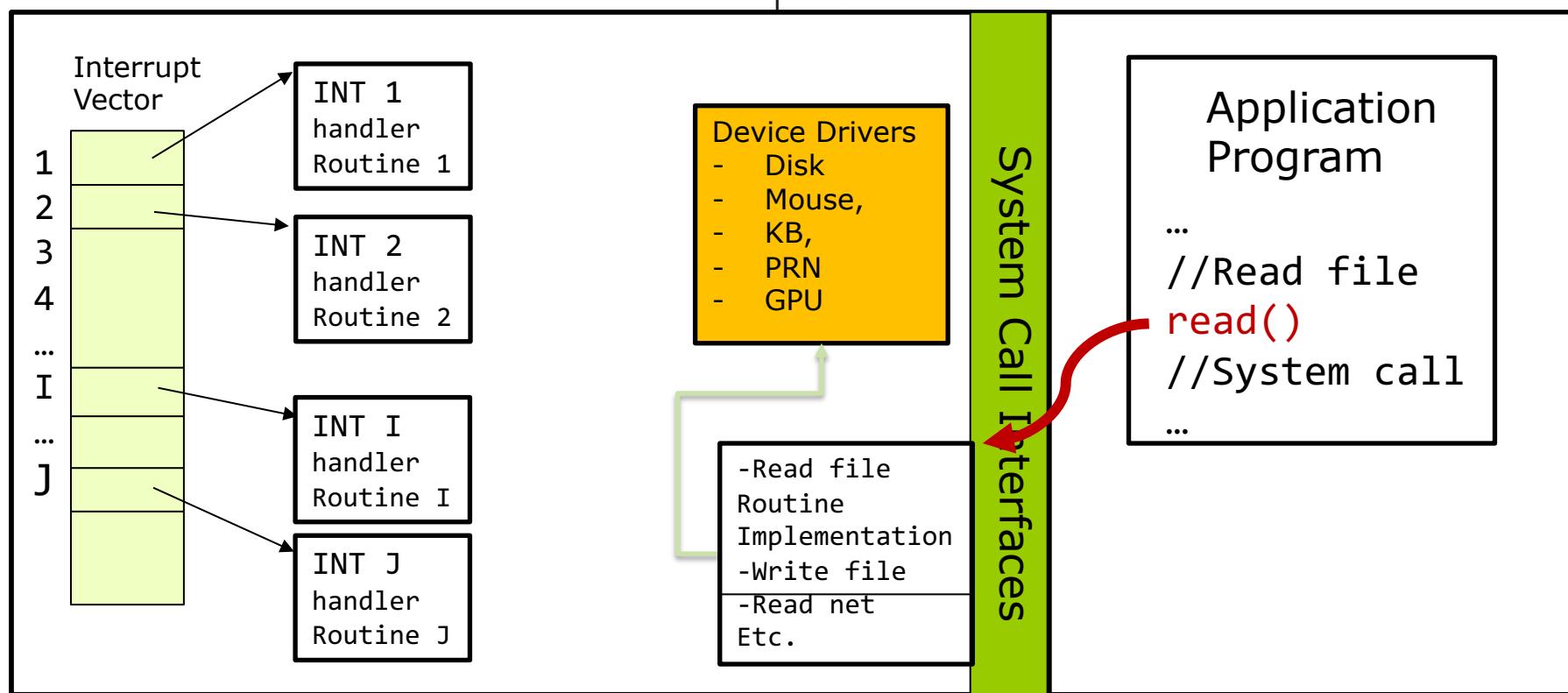


Operating-System Operations

- Bootstrap program – simple code to initialize the system, load the kernel
- Kernel loads
- Starts **system daemons** (services provided outside of the kernel)
- Kernel **interrupt driven** (hardware and software)
 - Hardware interrupt by one of the devices
 - Software interrupt (**exception** or **trap**):
 - ▶ Software error (e.g., division by zero)
 - ▶ Request for operating system service – **system call**
 - **System call trap** เป็นค่าตัวเลข **trap number** ค่าหนึ่งที่จะทำให้ชิปปุ่ย ประมวลผล **trap handler** ซึ่งจะเช็คค่าพารามิเตอร์ที่ผู้ใช้ส่งมาด้วย เพื่อพิจารณา ว่าผู้ใช้ต้องการรัน **system call routines** อะไร
 - ▶ Other process problems include infinite loop, processes modifying each other or the operating system



OS kernel



โครงสร้างจะง่ายขึ้นอีก... จะได้กล่าวถึงต่อไป



System Call

- Programming interface to the services provided by the OS
- เป็นวิธีการที่ User Process ใช้เรียกบริการ (หรือ Function) ของ OS
- ถ้าเรามองให้ OS เหมือนเป็นผู้ให้บริการจะเปรียบเหมือน
 1. เมื่อ hardware มีเหตุการณ์มันจะส่งสัญญาณ Interrupt เพื่อเรียก routine ของ OS
 2. เช่นเดียวกันเมื่อ Application มีความต้องการมันจะเรียก system call เพื่อเรียก routine ของ OS
- กลไกการประมวลผลของ System Call มีสองแบบใหญ่ๆ ได้แก่ Legacy System Call กับ Fast System Call
- เรา ยกตัวอย่าง Linux Legacy และ Fast System Call Implementations



Quote from Silberschatz Operating Systems Concepts:

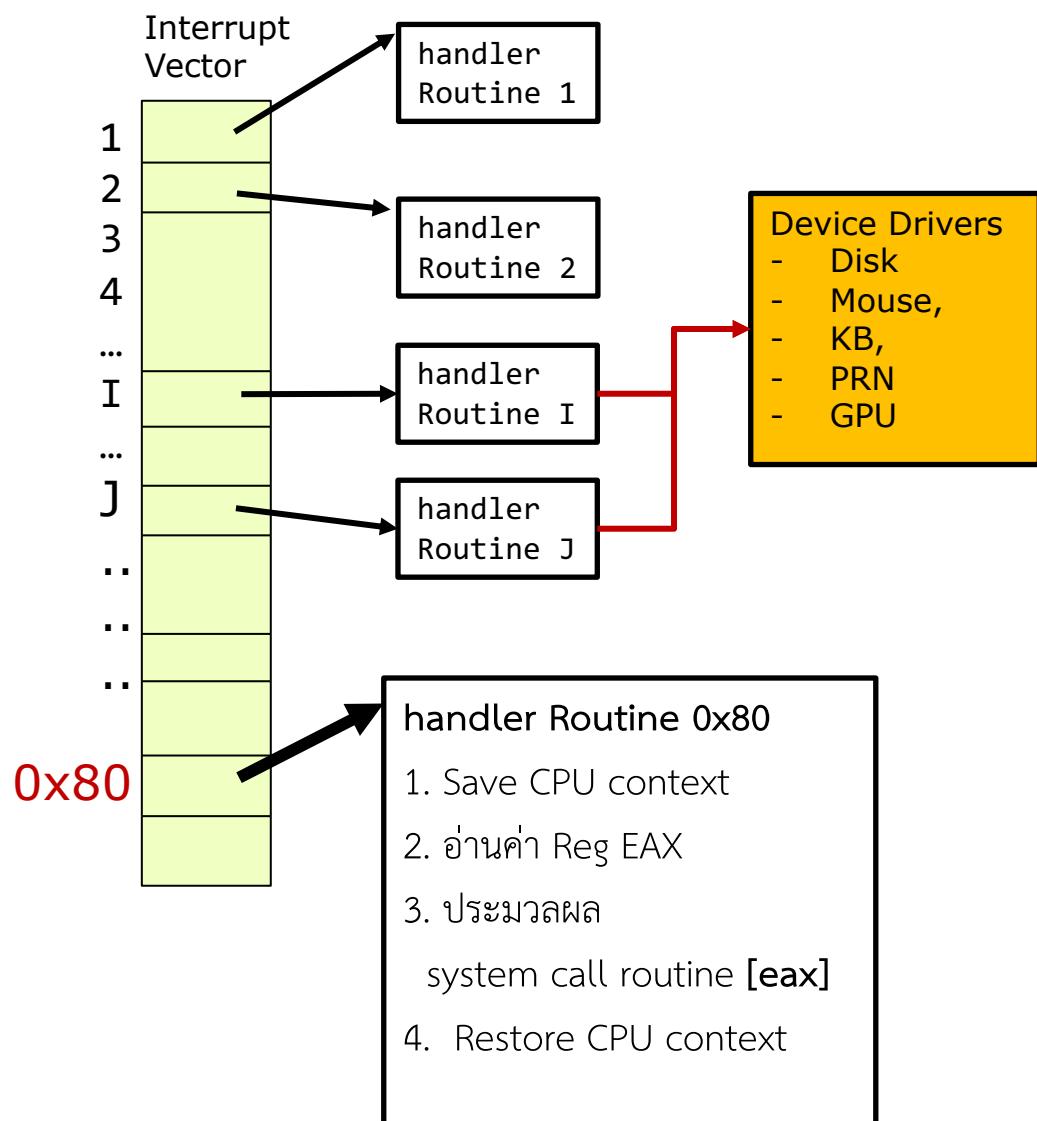
A system call usually takes the form of a trap to a specific location in the **interrupt vector**. This trap can be executed by a generic trap instruction, although some systems (such as MIPS) have a specific `syscall` instruction to invoke a system call.

When a system call is executed, it is typically treated by the hardware as a software interrupt. Control passes through the **interrupt vector** to a service routine in the operating system, and the mode bit is set to kernel mode. The **system-call service routine** is a part of the operating system. The kernel examines the interrupting instruction to determine what system call has occurred; a parameter indicates what type of service the user program is requesting. Additional information needed for the request may be passed in registers, on the stack, or in memory (with pointers to the memory locations passed in registers). The kernel verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call.

<https://stackoverflow.com/questions/29656136/is-there-a-system-call-service-routine-in-the-interrupt-vector#:~:text=A%20system%20call%20usually%20takes,to%20invoke%20a%20system%20call.>

Legacy System Call

memory



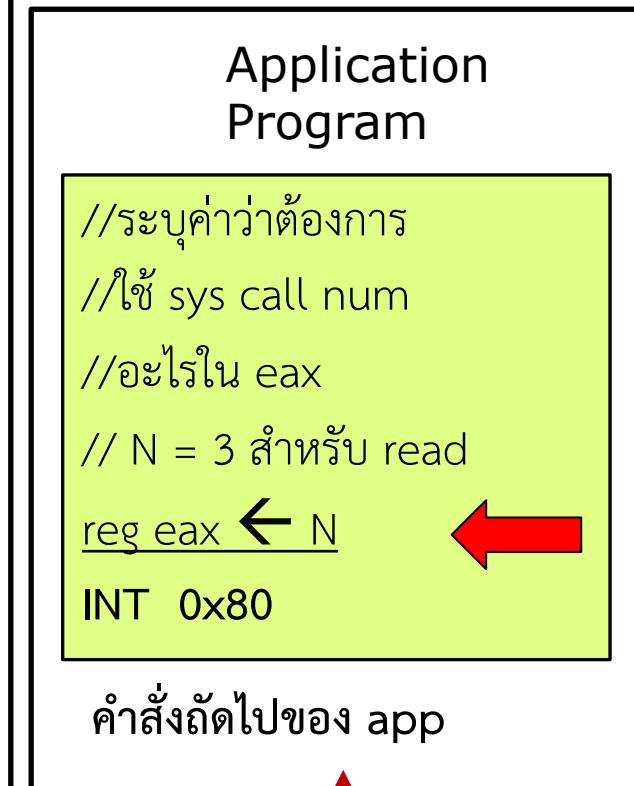
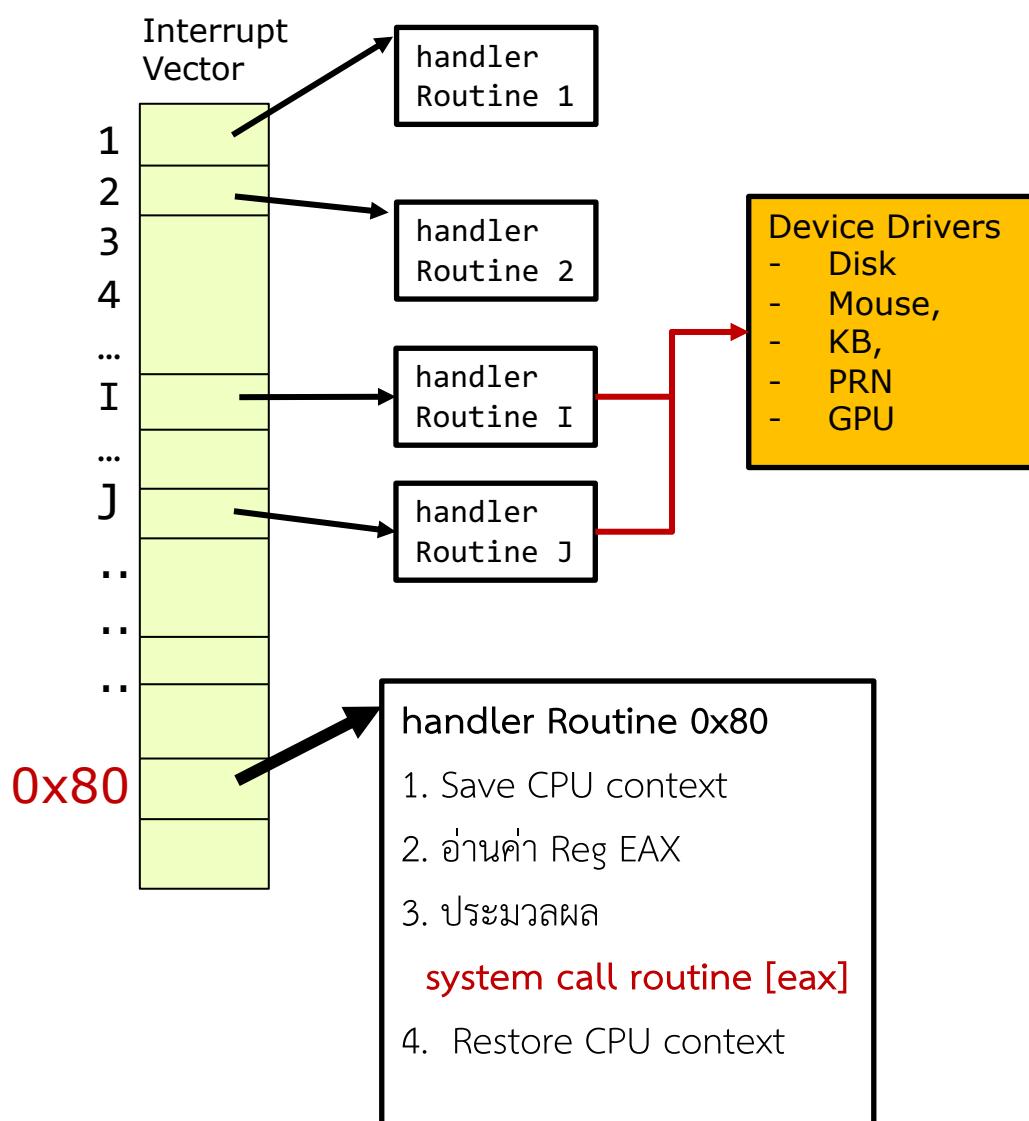
Application Program

```
// อ่านค่าจากไฟล์  
// เรียก libc API  
read(...)
```

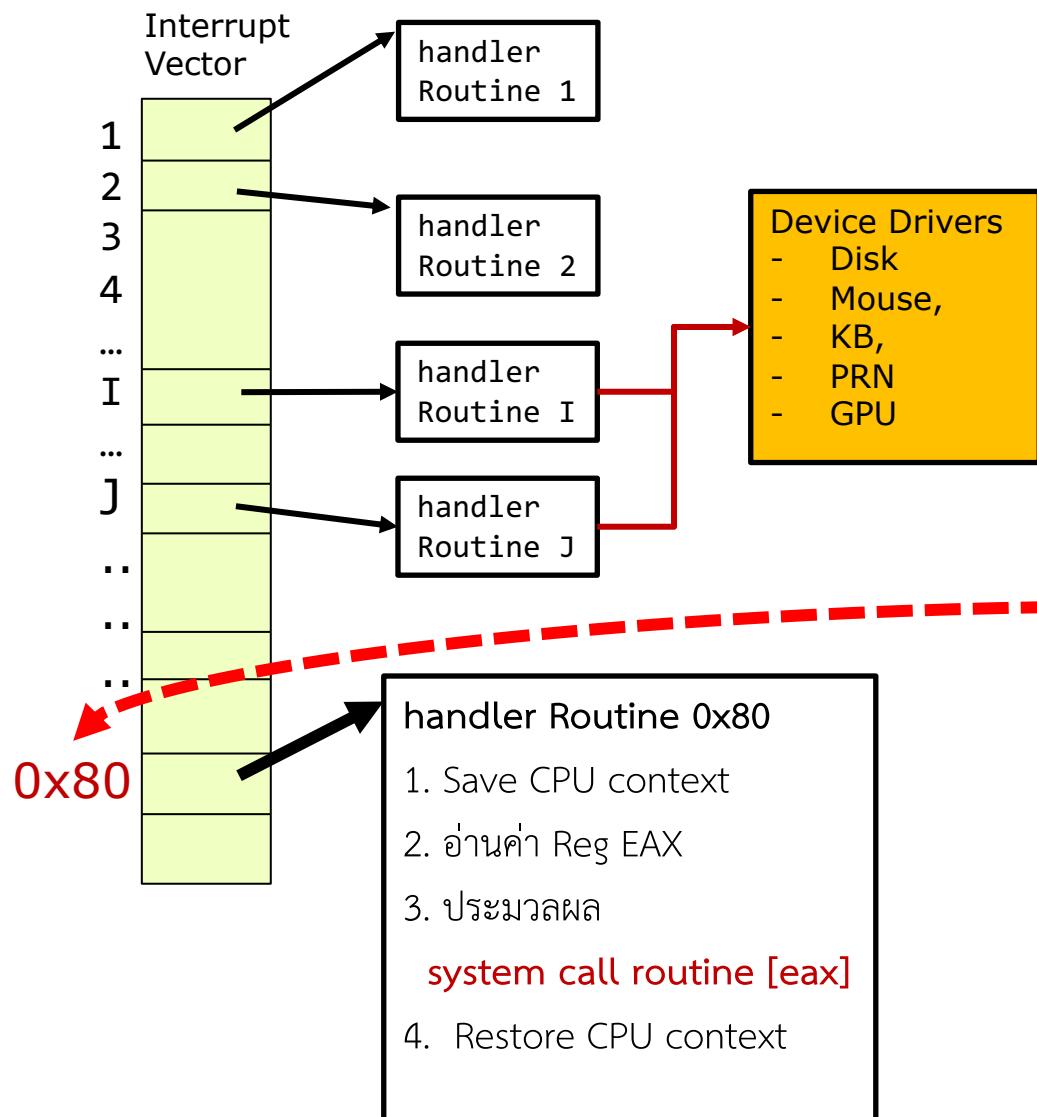
คำสั่งถัดไปของ app

OS kernel

memory



ข้างใน implementation
ของ read()



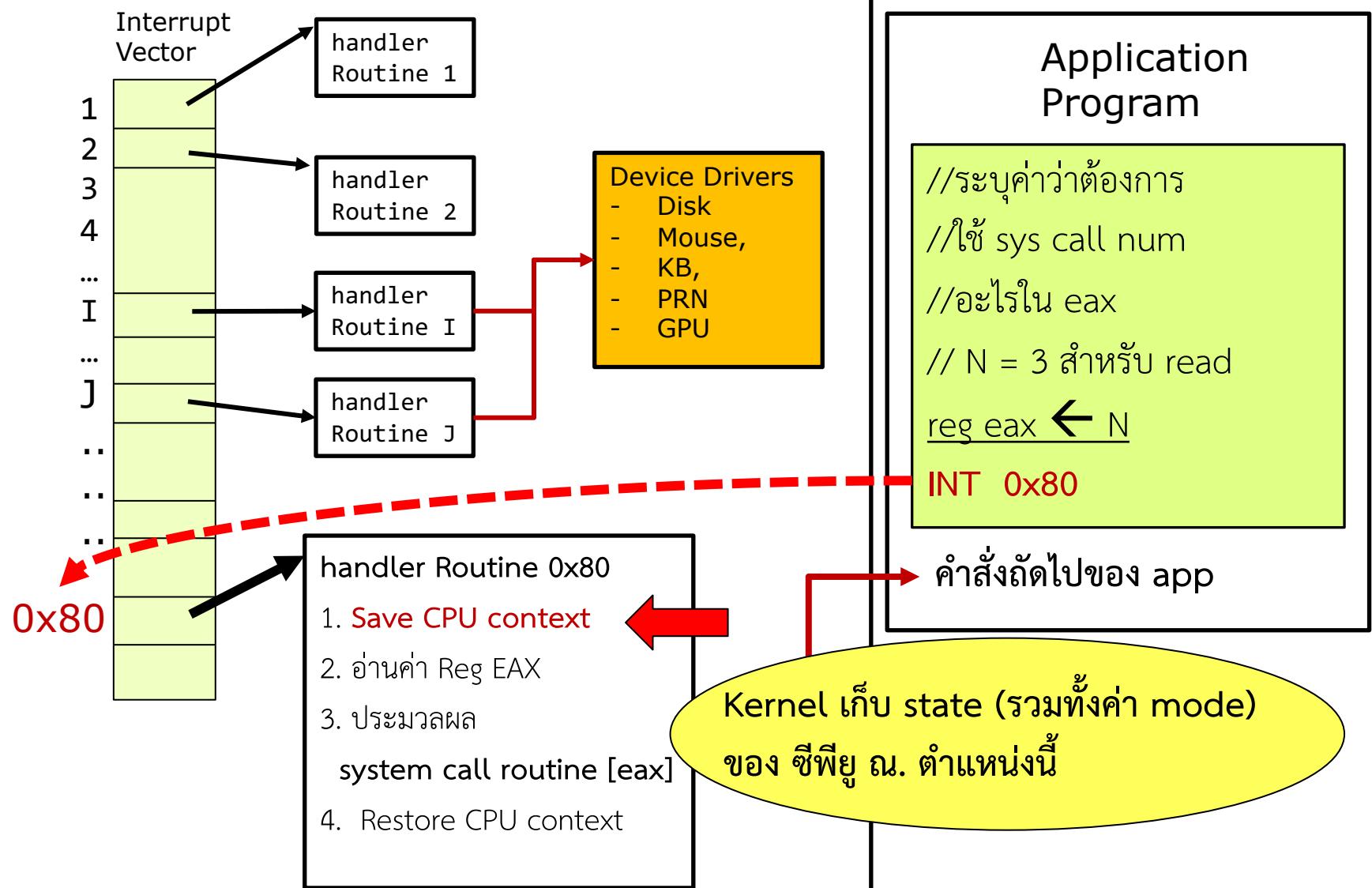
Application Program

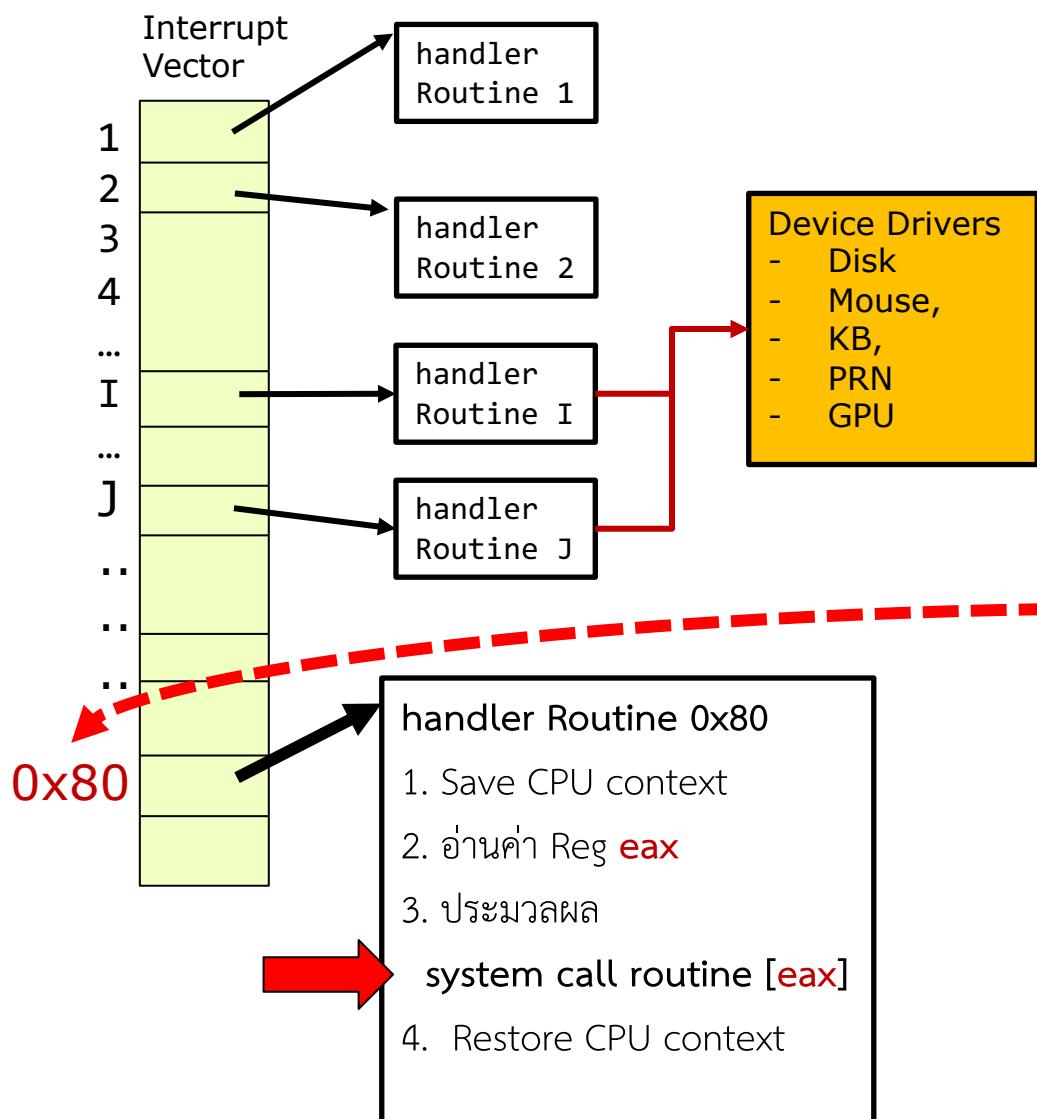
```
//ระบุค่าที่ต้องการ
//ใช้ sys call num
//อธิบายใน eax
// N = 3 สำหรับ read
reg eax <- N
```

INT 0x80

คำสั่งถัดไปของ app

ข้างใน implementation
ของ read()

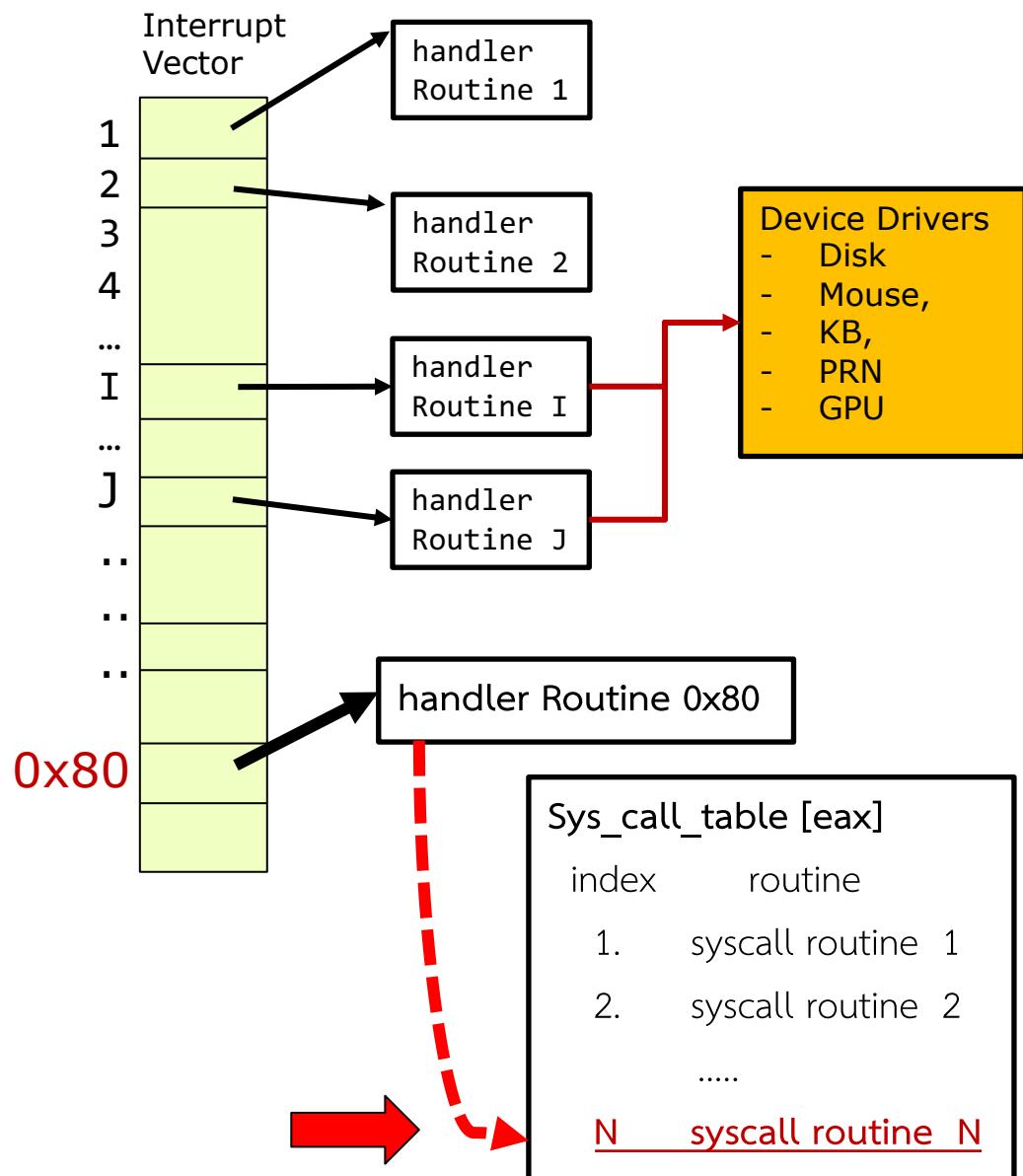




Application Program

```
//ระบุค่าที่ต้องการ
//ใช้ sys call num
//อะไรมีใน eax
// N = 3 สำหรับ read
reg eax ← N
INT 0x80
```

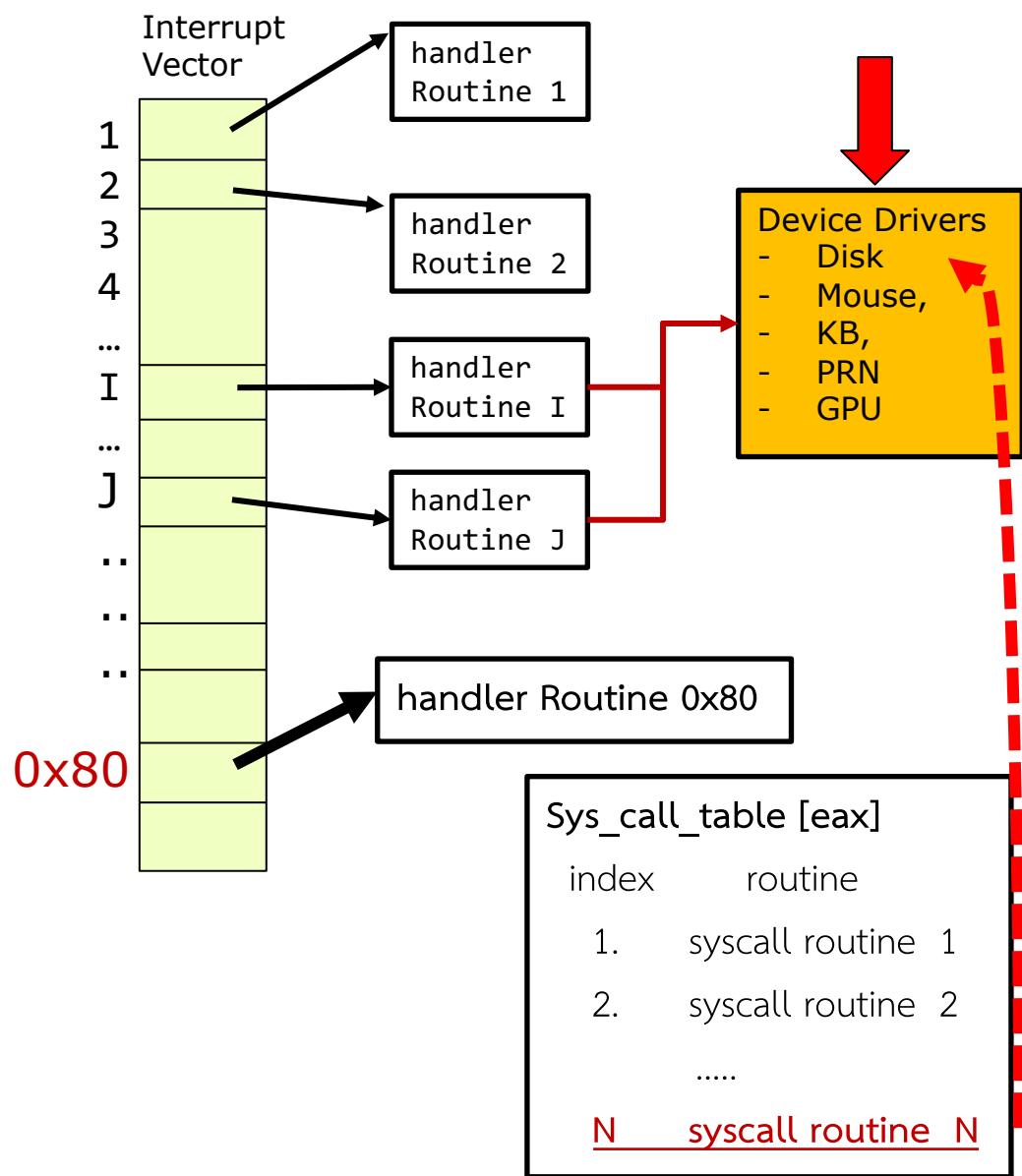
คำสั่งถัดไปของ app



Application Program

```
//ระบุค่าที่ต้องการ
//ใช้ sys call num
//อะไรมีใน eax
// N = 3 สำหรับ read
reg eax ← N
INT 0x80
```

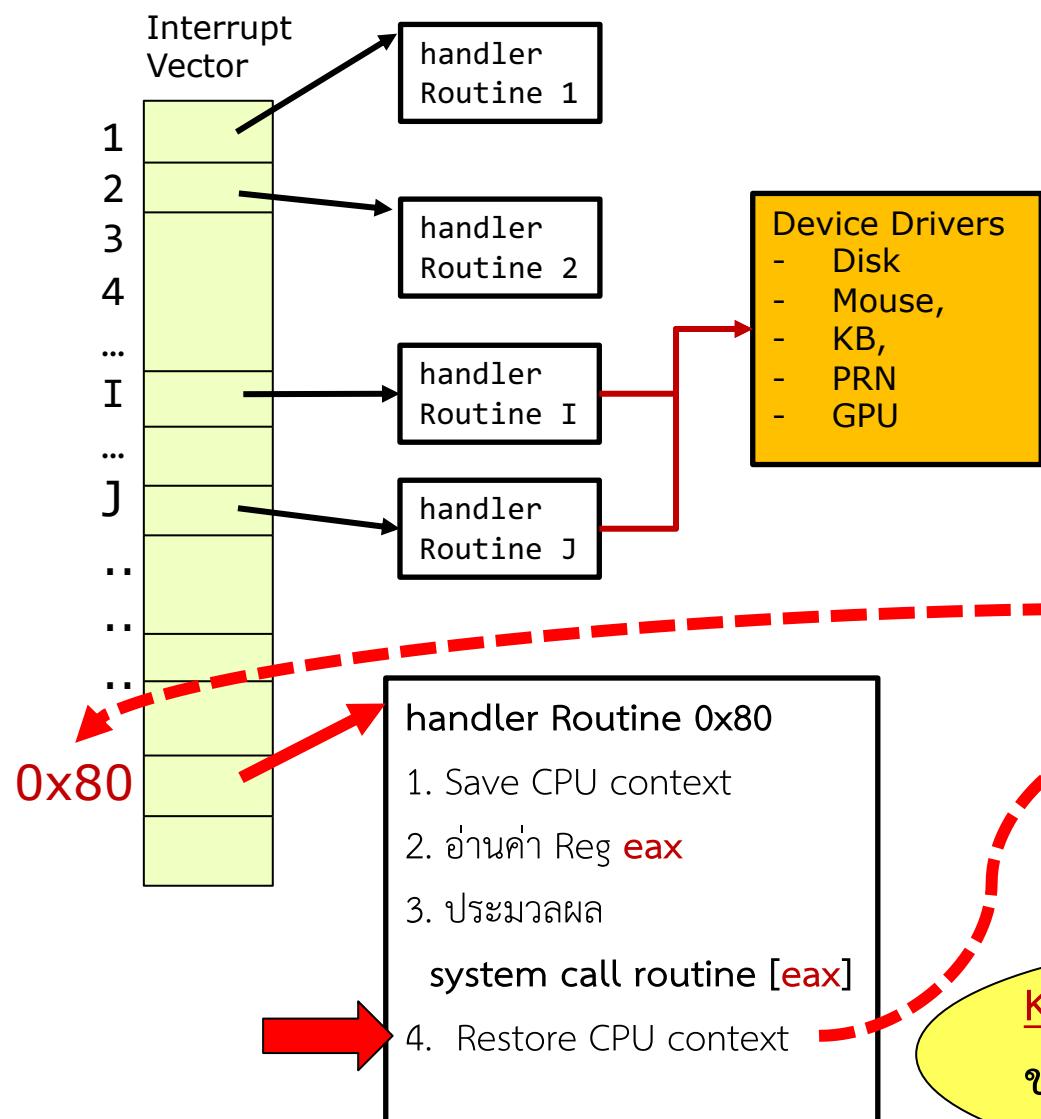
คำสั่งถัดไปของ app

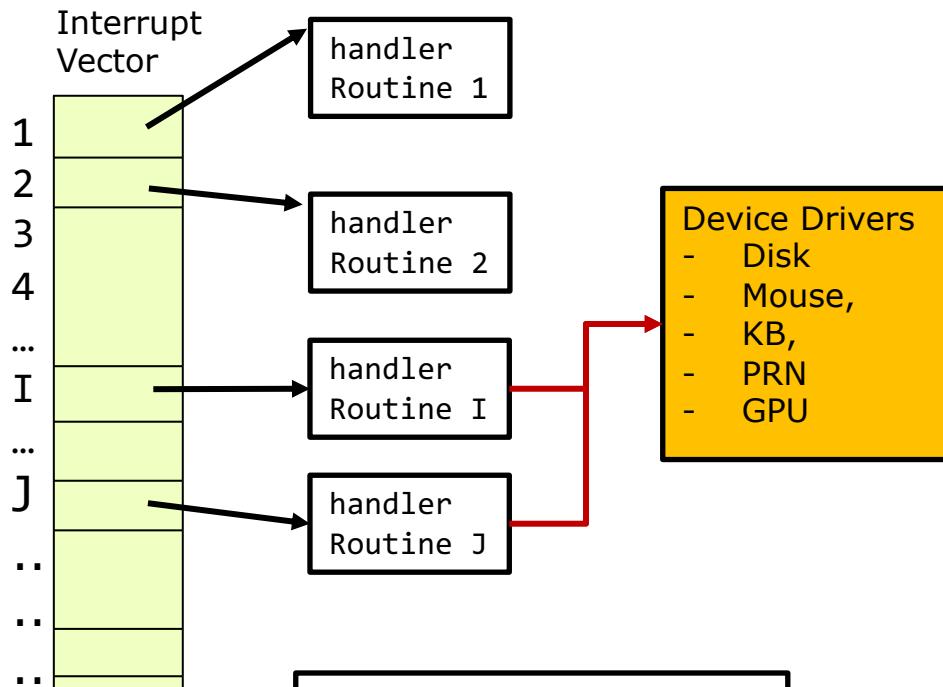


Application Program

```
//ระบุค่าที่ต้องการ
//ใช้ sys call num
//อะไรมีใน eax
// N = 3 สำหรับ read
reg eax ← N
INT 0x80
```

คำสั่งถัดไปของ app





syscall handler Routine

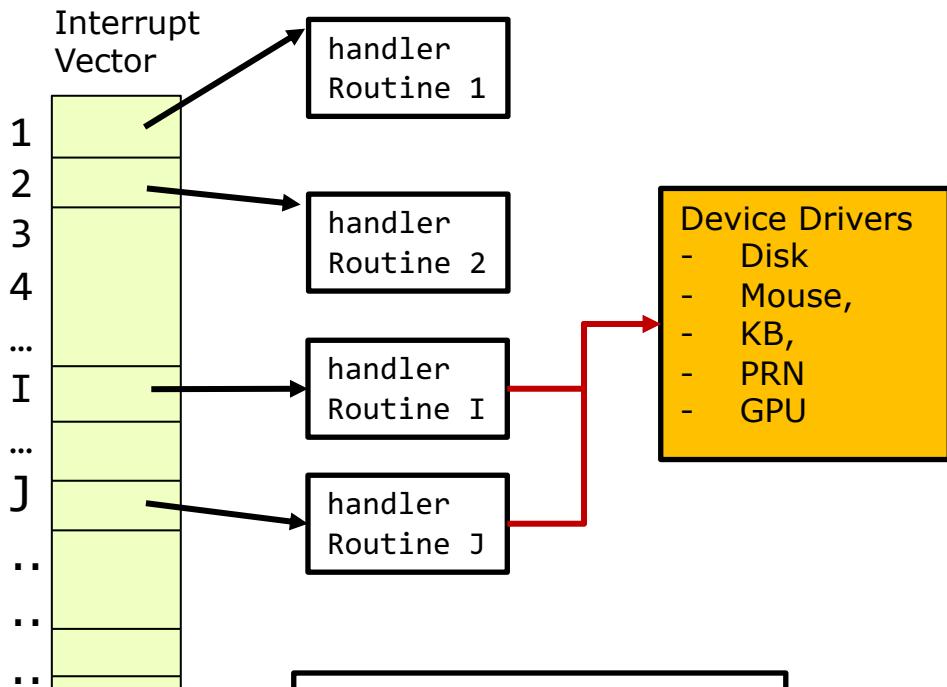
1. Save CPU context
2. อ่านค่า Reg EAX
3. ประมวลผล system call routine [eax]
4. Restore CPU context

Application Program

```
// อ่านค่าจากไฟล์
// เรียก libc API
read(...)
```

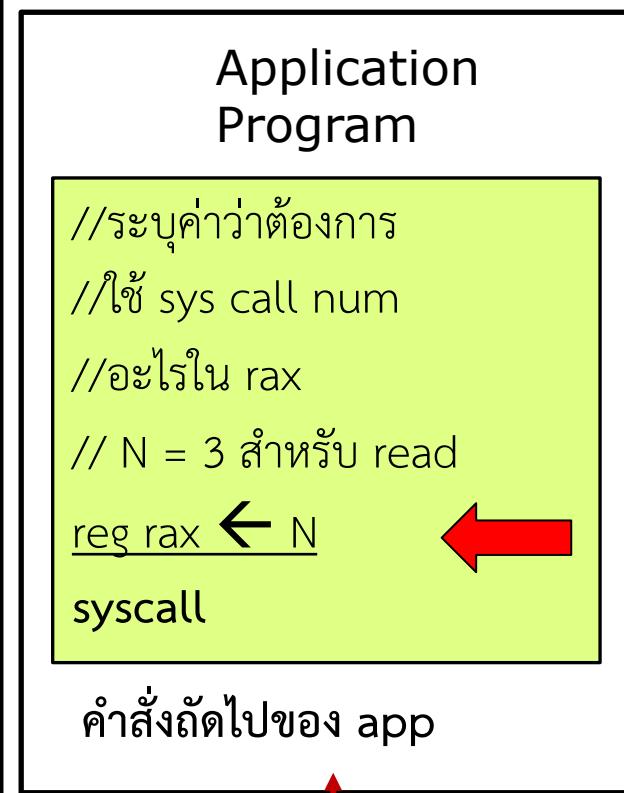
คำสั่งถัดไปของ app

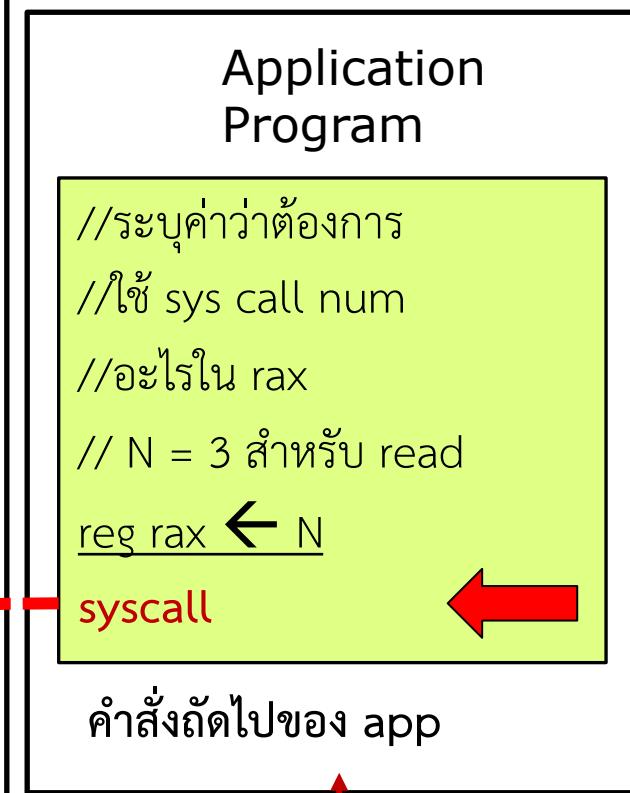
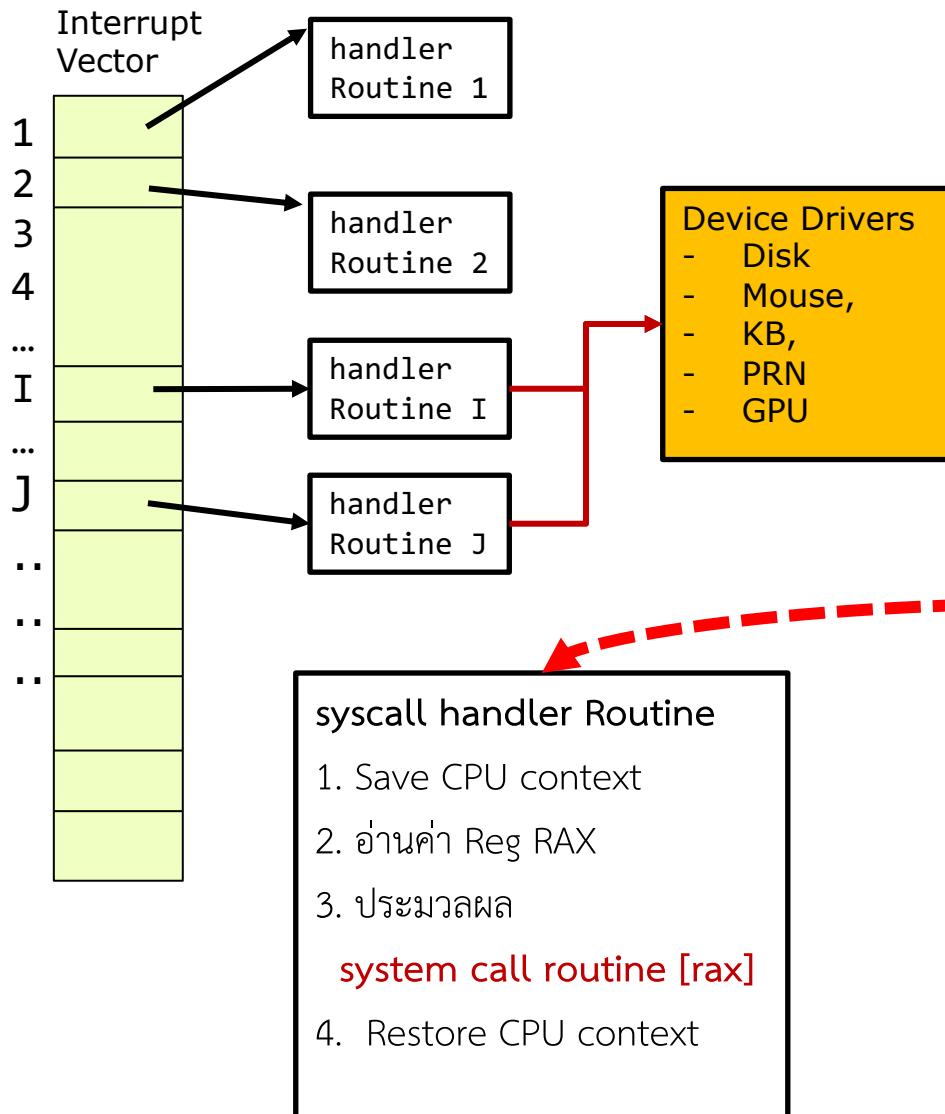
ปรับ hardware
เพื่อ Software



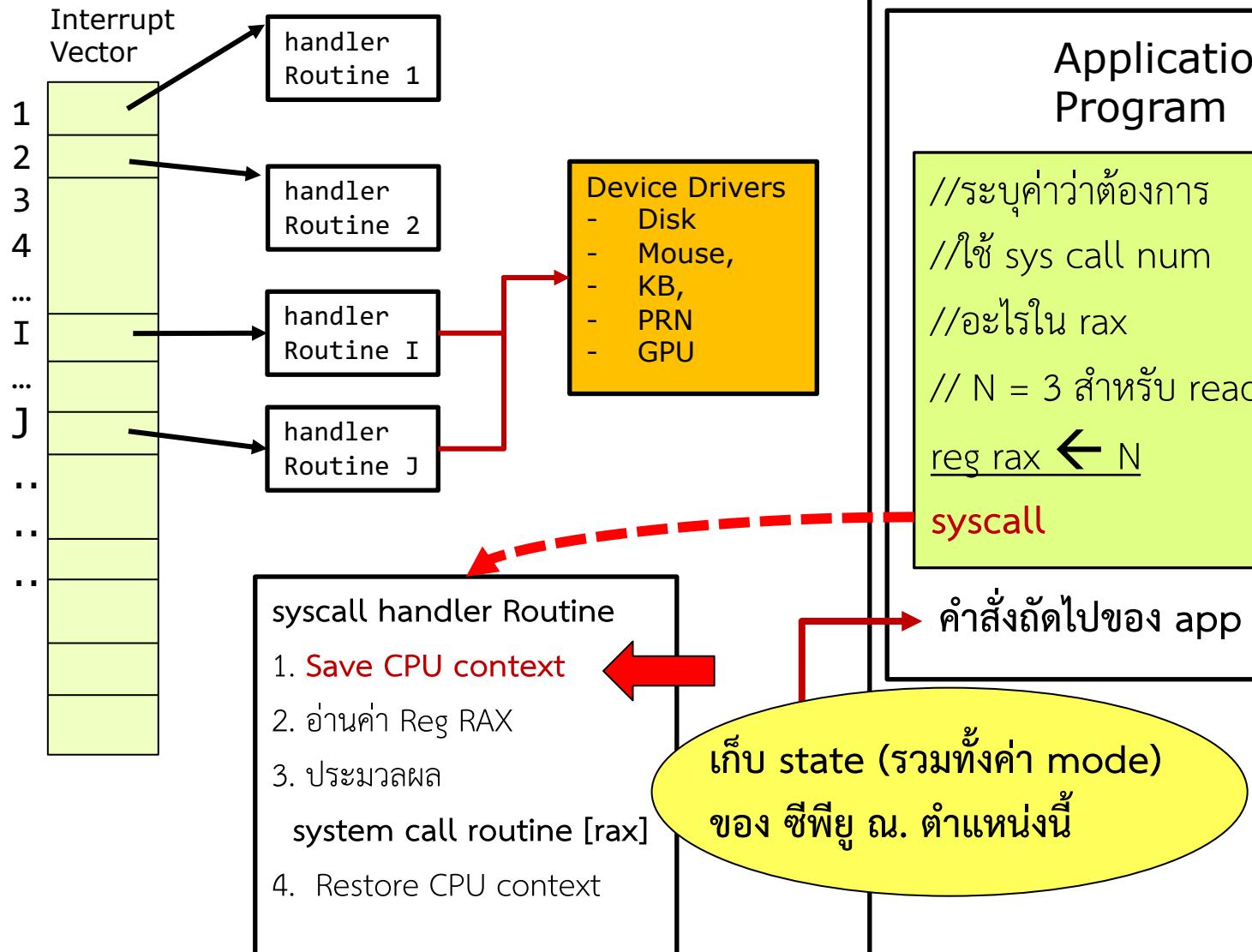
syscall handler Routine

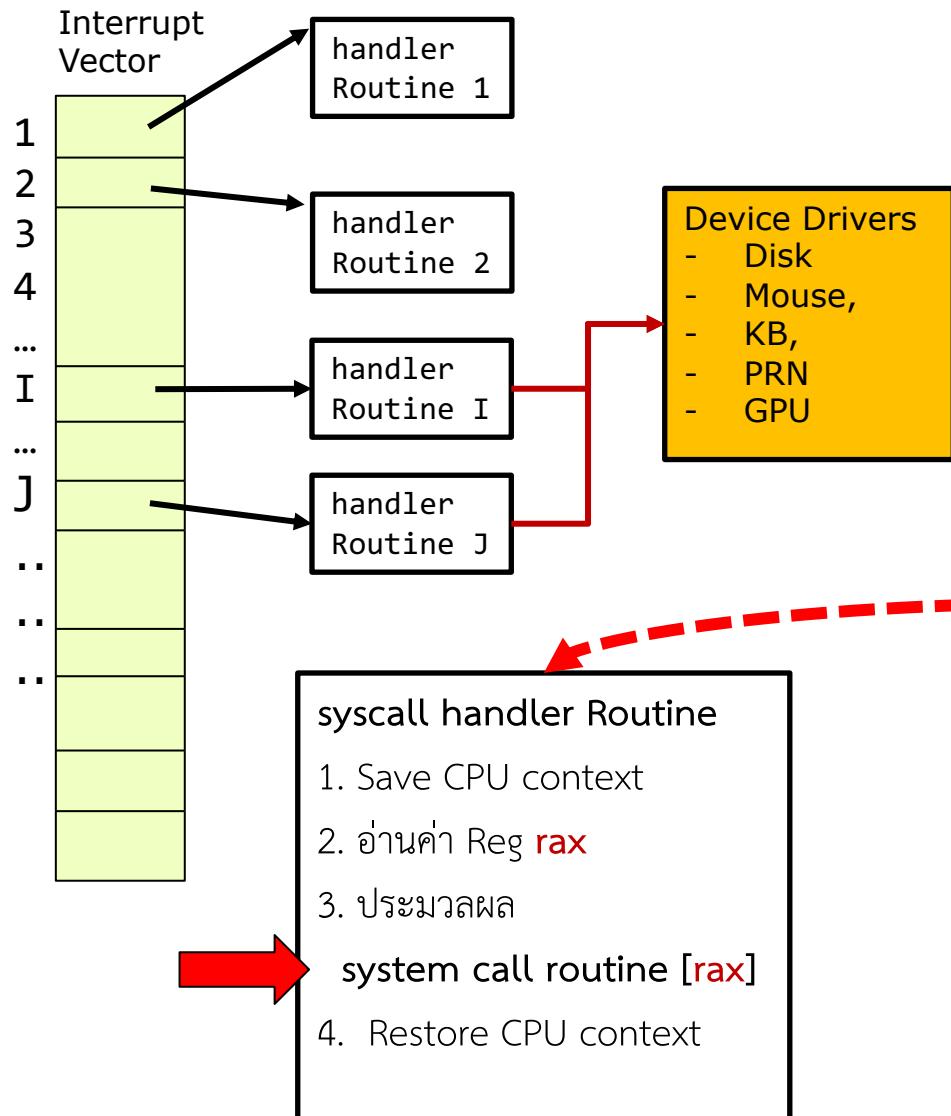
1. Save CPU context
2. อ่านค่า Reg rax
3. ประมวลผล
- system call routine [rax]**
4. Restore CPU context





ข้างใน implementation
ของ read()

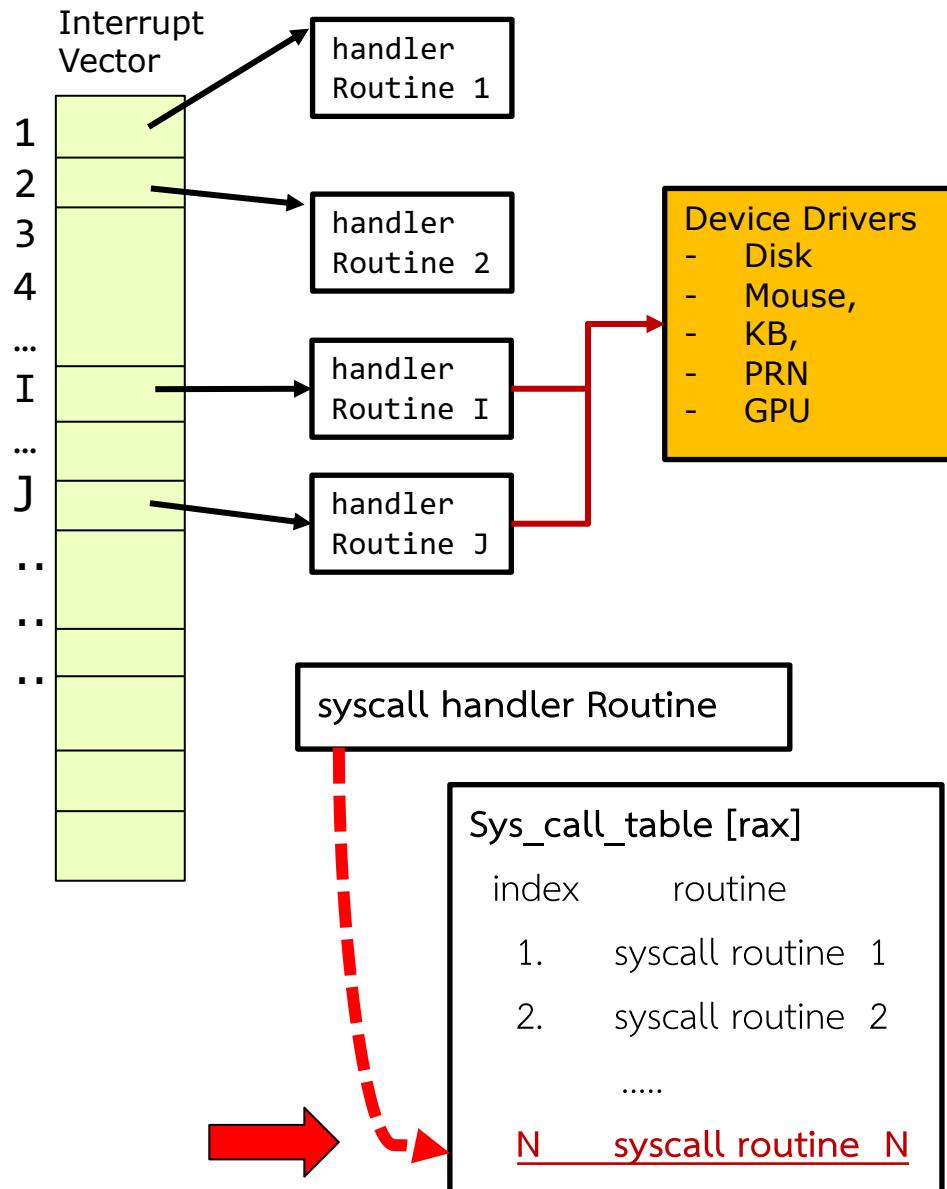




Application Program

```
//ระบุค่าที่ต้องการ
//ใช้ sys call num
//อ่าเรื่องใน rax
// N = 3 สำหรับ read
reg rax ← N
syscall
```

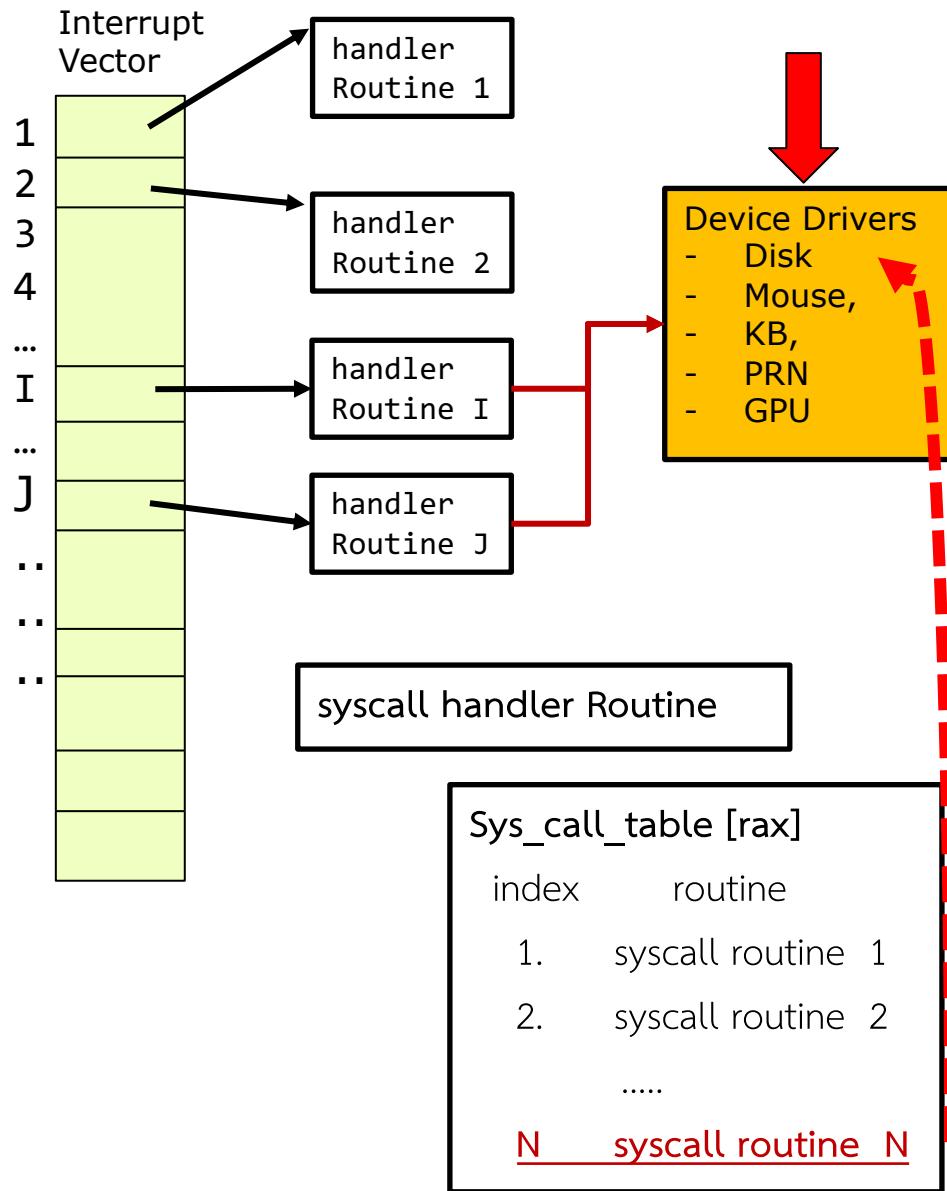
คำสั่งถัดไปของ app



Application Program

```
//ระบุค่าที่ต้องการ
//ใช้ sys call num
//อะไรมีใน rax
// N = 3 สำหรับ read
reg rax < N
syscall
```

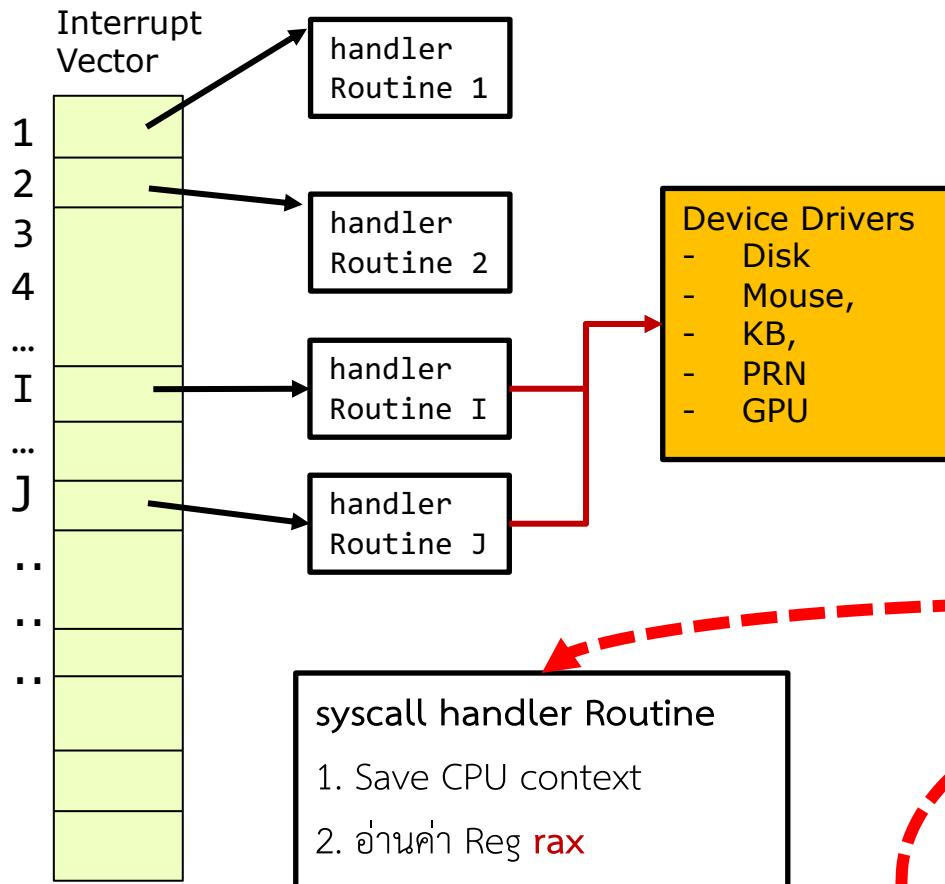
คำสั่งถัดไปของ app



Application Program

```
//ระบุค่าที่ต้องการ
//ใช้ sys call num
//อะไรมีใน rax
// N = 3 สำหรับ read
reg rax < N
syscall
```

คำสั่งถัดไปของ app



Application Program

```
// ระบุค่าที่ต้องการ
// ใช้ sys call num
// อ่าไว้ใน rax
// N = 3 สำหรับ read
reg rax ← N
```

syscall

คำสั่งถัดไปของ app

syscall handler Routine

1. Save CPU context

2. อ่านค่า Reg **rax**

3. ประมวลผล

system call routine [**rax**]

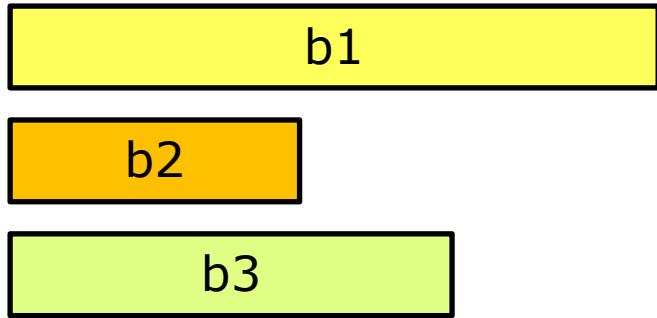
4. Restore CPU context

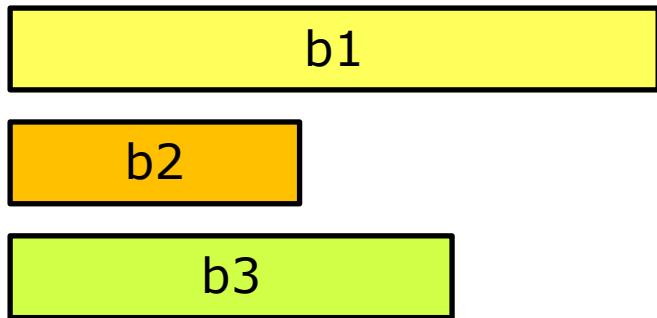
คืนค่า state (รวมทั้งค่า mode)
ของซีพียู ณ. ตำแหน่งนี้



Multiprogramming (Batch system)

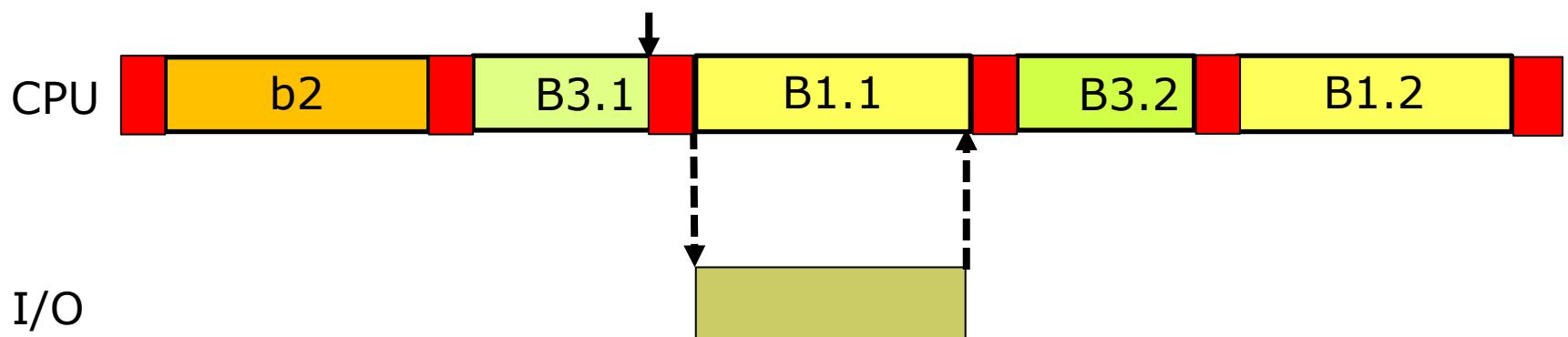
- Single user cannot always keep CPU and I/O devices busy
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via **job scheduling**
- When job has to wait (for I/O for example), OS switches to another job
- Batch job คือโปรแกรมที่อ่านอินพุตจากผู้ใช้ประมวลผลเป็นเวลานานและส่งคืนເອົາພຸທໃຫ້ຜູ້ໃຊ້
- Process คือโปรแกรมที่กำลังประมวลผล
- เรามักจะเรียก job และ task ที่รันบน OS ว่า Process





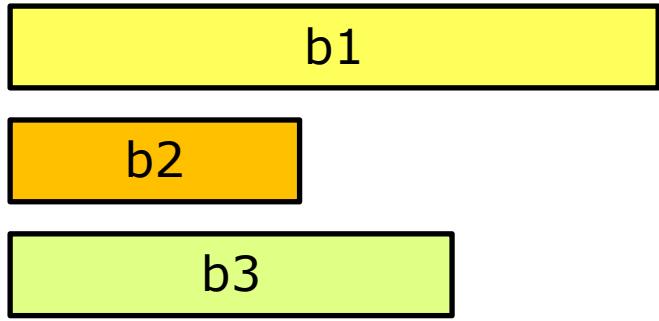
OS/Job Scheduler

B3 อ่านข้อมูลจากไฟล์

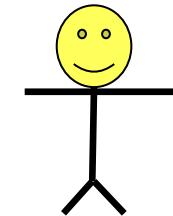
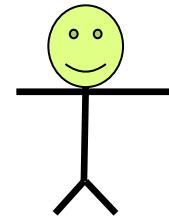
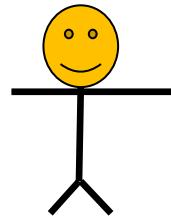


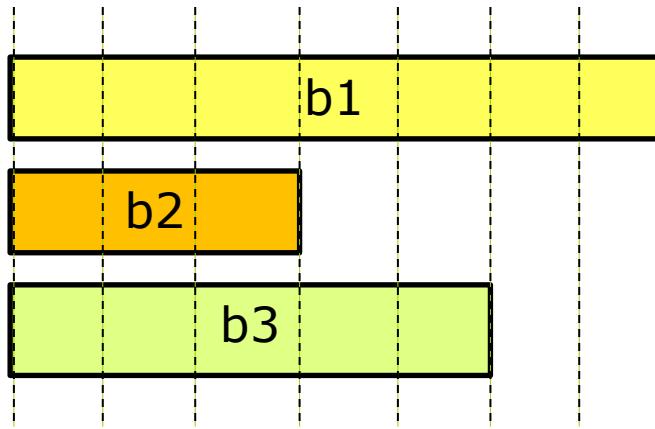
Multitasking (Timesharing)

- A logical extension of Batch systems— the CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
 - **Response time** should be < 1 second
 - **Each user** has at least one program executing in memory
⇒ **process**
 - If several jobs ready to run at the same time ⇒ **CPU scheduling**
 - If processes don't fit in memory, **swapping** moves them in and out to run
 - **Virtual memory** allows execution of processes not completely in memory



An oval with a thick black border and a red fill. Inside the oval, the text 'OS/Job Scheduler' is written in white.



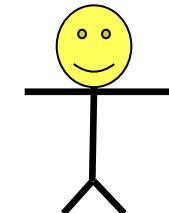
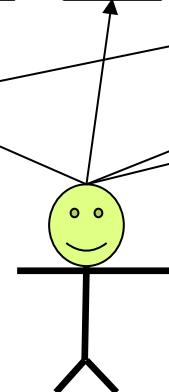
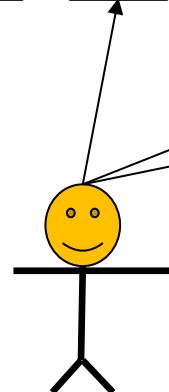
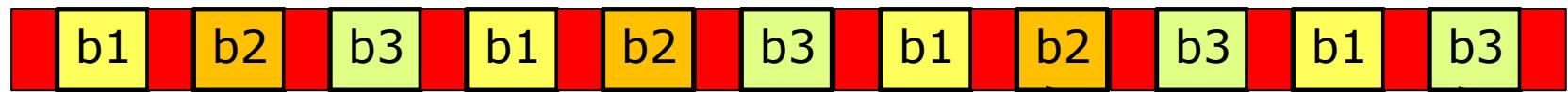


OS แบ่งเวลาเรียกว่า

Time slice < 1 วินาที
ให้แต่ละ process

OS/Job Scheduler

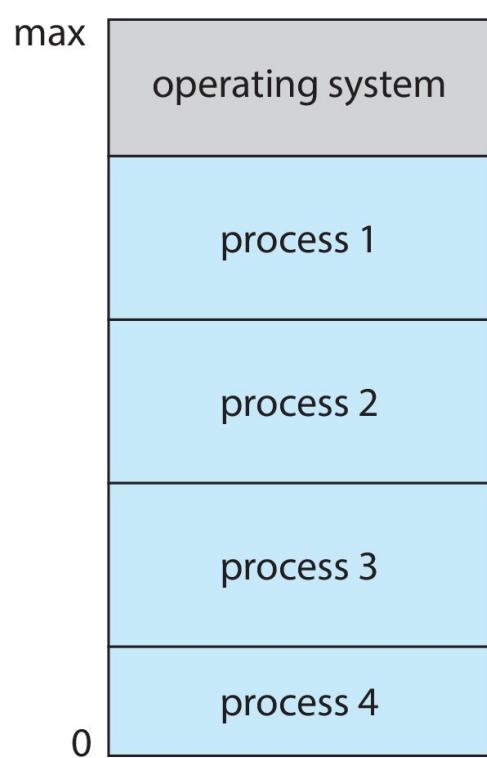
CPU



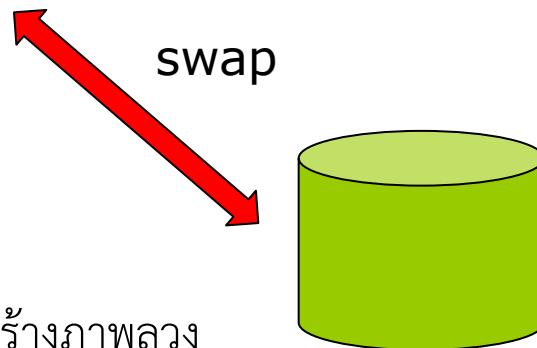
Memory Layout for Multiprogrammed System

1) Process จะเกิดขึ้นได้
ต้องมีเนื้อหาโหลด (load)
เข้ามาอยู่ใน
Main memory จน
กระทั่งจบการประมวลผล

2) Multiprogramming
คือความสามารถของ OS
ที่สามารถรันโปรแกรม
หลายโปรแกรมได้พร้อมกัน
จะเกิดขึ้นได้ ต้องมีเนื้อหา
ของโปรแกรมเหล่านั้นอยู่ใน
Main memory



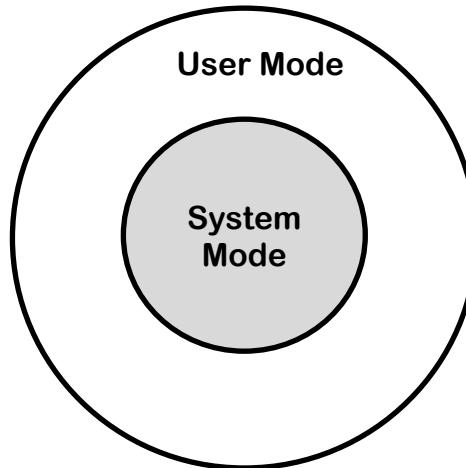
4) ถ้าพื้นที่ไม่พอ ก็ต้องนำ
เนื้อหาของบาง Process สลับ
ไปไว้ใน Disk เเรียกว่า การ Swap
(อ่าน สวอป)



3) Virtual memory คือ การสร้างภาพลง
ให้แต่ละโปรแกรมคิดว่า ข้อมูลของมันอยู่ใน
Memory ทั้งหมดแต่จริงๆแล้วมีบางส่วนเท่านั้น
อยู่ใน Memory จริงของเครื่อง



Protection Modes ของซีพีयู



- ซีพีyu จะออกแบบมาสนองความต้องการให้รองรับ multiprogramming และ multiusers
- ซีพีyu จะออกแบบให้ทำงานเป็นหลาย Mode เพื่อรักษาความปลอดภัยให้ผู้ใช้
- Mode คือสถานการณ์การทำงานของ CPU ที่มีได้หลายระดับ เราเรียนแต่ละระดับว่า Ring Level



เปรียบเทียบเรื่องโหมดการทำงานของซีพียู

Dual-Mode Operation

- สมมุติว่า นศ (OS kernel) เป็นเจ้าของโน๊ตบุ๊คเรื่องหนึ่ง (CPU) และมีน้องเล็กๆ ชนอยู่คนหนึ่ง (Application Program) (อาจมีหลายคนก็ได้) และ นศ จะต้องใช้โน๊ตบุ๊คร่วมกับน้อง แต่เวลาที่น้องใช้ นศ จะต้อง set ค่า parental control (CPU Mode) ไม่ให้น้องสามารถ web site ที่ไม่เหมาะสมกับเด็กได้
- ตอนแรก นศ ใช้โน๊ตบุ๊คออยู่ นศ ใช้โหมดผู้ใหญ่ (System Mode) เมื่อจะให้น้องใช้ ก็เปลี่ยนโหมดเป็นโหมดเด็ก (User Mode)
- พอน้องใช้ไป แล้วไปทำในสิ่งที่โหมดของเขาทำไม่ได้ หรือใช้ไปแล้วเกิดเหตุการณ์บางอย่างขึ้น เขาก็ต้องวิงมาหาพี่ (Trap หรือ Interrupts)
- นศ ก็จะรับเครื่องมาและเปลี่ยนโหมดให้เป็นผู้ใหญ่ (System Mode) และ แก้ปัญหาให้ แล้วเปลี่ยนโหมดกลับให้เป็นโหมดเด็ก (User Mode) แล้วส่งเครื่องให้น้องใช้ต่อ

Dual-mode Operation

- **Dual-mode** operation allows OS to **protect** itself and other system components
 - **User mode** and **kernel mode**
- **Mode bit** provided by hardware
 - Provides ability to distinguish when system is running user code or kernel code.
 - When a user is running \Rightarrow mode bit is “user”
 - When kernel code is executing \Rightarrow mode bit is “kernel”
- How do we guarantee that user does not explicitly set the mode bit to “kernel”?
 - System call changes mode to kernel, return from call resets it to user
- Some instructions designated as **privileged**, only executable in kernel mode

ชนิดของชุดคำสั่งของซีพียู

- ชุดคำสั่งของซีพียูเรียกว่า Instruction Set Architecture (ISA)
- CPU รองรับการประมวลผลชุดคำสั่งสองแบบได้แก่ User ISA และ System ISA
- User ISA คือชุดคำสั่งสำหรับการประมวลผลทั่วไปสำหรับการอ่านและเขียนข้อมูลบนหน่วยความจำ และคำสั่งสำหรับประมวลผล Arithmetic Logic Unit
- System ISA คือชุดคำสั่งที่เกี่ยวข้องกับการควบคุมการประมวลผลของระบบคอมพิวเตอร์และการเข้าถึงและสั่งงานอุปกรณ์ I/O และจัดการทรัพยากรในระบบคอมพิวเตอร์ (เรียกอีกอย่างว่า Privilege Instruction)

User ISA

Memory Instr.	Integer Instr.	Floating-Point	Branch Instr.
Load byte	Add	Add single	Branch
Load word	Compare logical	Multiply double	Branch if
Store byte	Exclusive OR	Multiplyadd	negative
Store multiple	Count leading	double	Jump to
Load double	zeros	Convert to	subroutine.
...	Rotate left w.	integer	Return
...	carry	Compare double	...

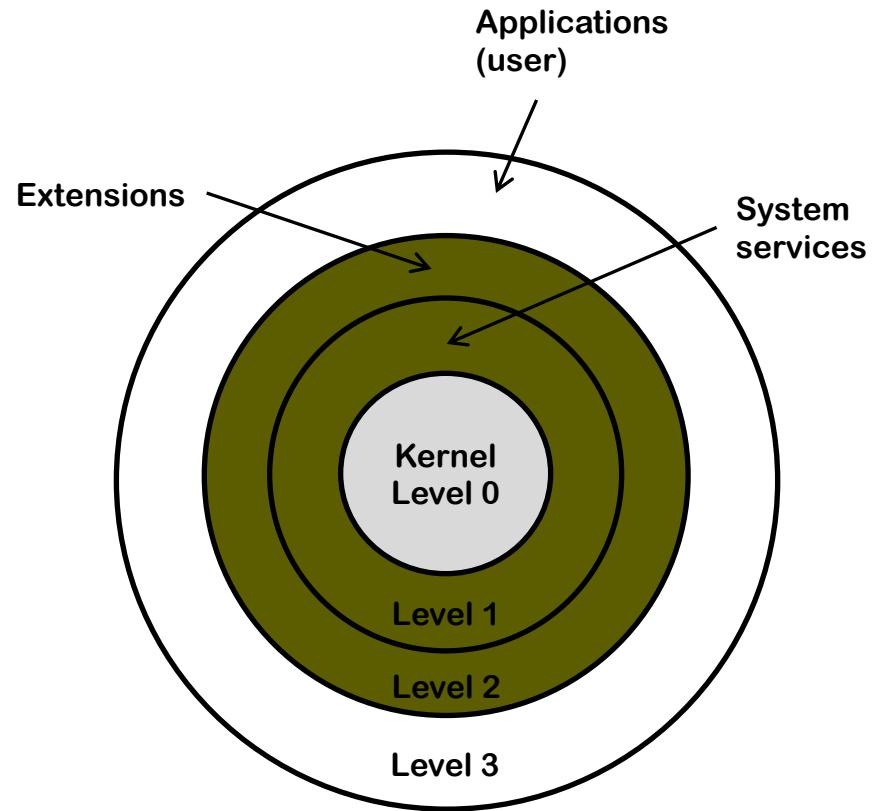
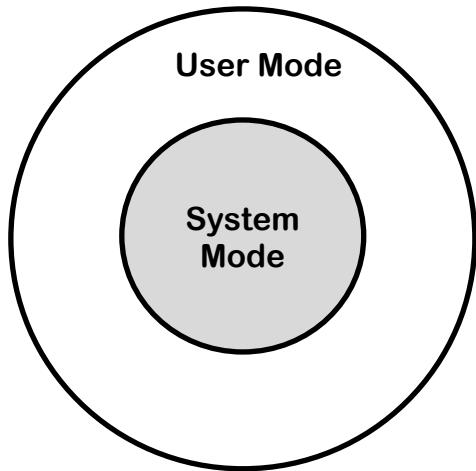
System ISA (Privilege Inst)

- คำสั่งสำหรับสั่งให้อ่านและเขียนข้อมูลจากอุปกรณ์ I/O เช่น HDD หรือ SSD หรือ Network
- คำสั่งสำหรับจับจองพื้นที่ในหน่วยความจำ
- คำสั่งสำหรับกำหนดค่าอุปกรณ์สาร์ดแวร์ เช่น วิจารณ์ Timer (จับเวลา และตั้งเวลาการทำงานพิเศษ)
- คำสั่งเพื่อเปลี่ยนโหมดการประมวลผลของชีพียู

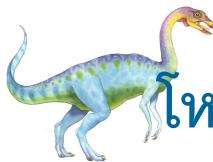
โหมดการประมวลผลของซีพีyu

- เวลาที่ซีพีyuประมวลผล มันจะปฏิบัติงานอยู่ในโหมดของการประมวลผลแบบใดแบบหนึ่งในสองแบบคือ User Mode หรือ System Mode
- เวลาที่ OS ประมวลผลซีพีyuจะทำงานใน System Mode
- เวลาที่ Application Programs ประมวลผล ซีพีyuจะทำงานใน User Mode
- ซีพีyuจะประมวลผลคำสั่ง System ISA ได้ เมื่อมันทำงานอยู่ใน System Mode เท่านั้น
- ซีพีyuสามารถประมวลผลคำสั่ง User ISA ได้เสมอไม่ว่าจะอยู่ใน System Mode หรือ User Mode
- เวลาที่ Application Programs ประมวลผลแล้วเกิดความผิดพลาด (Trap) หรือมีสัญญาณขัดจังหวะ (Interrupts) ส่งมาให้ซีพีyu ซีพีyuจะเปลี่ยน mode เป็น system Mode โดยอัตโนมัติ เพื่อรัน Trap หรือ Interrupt handler

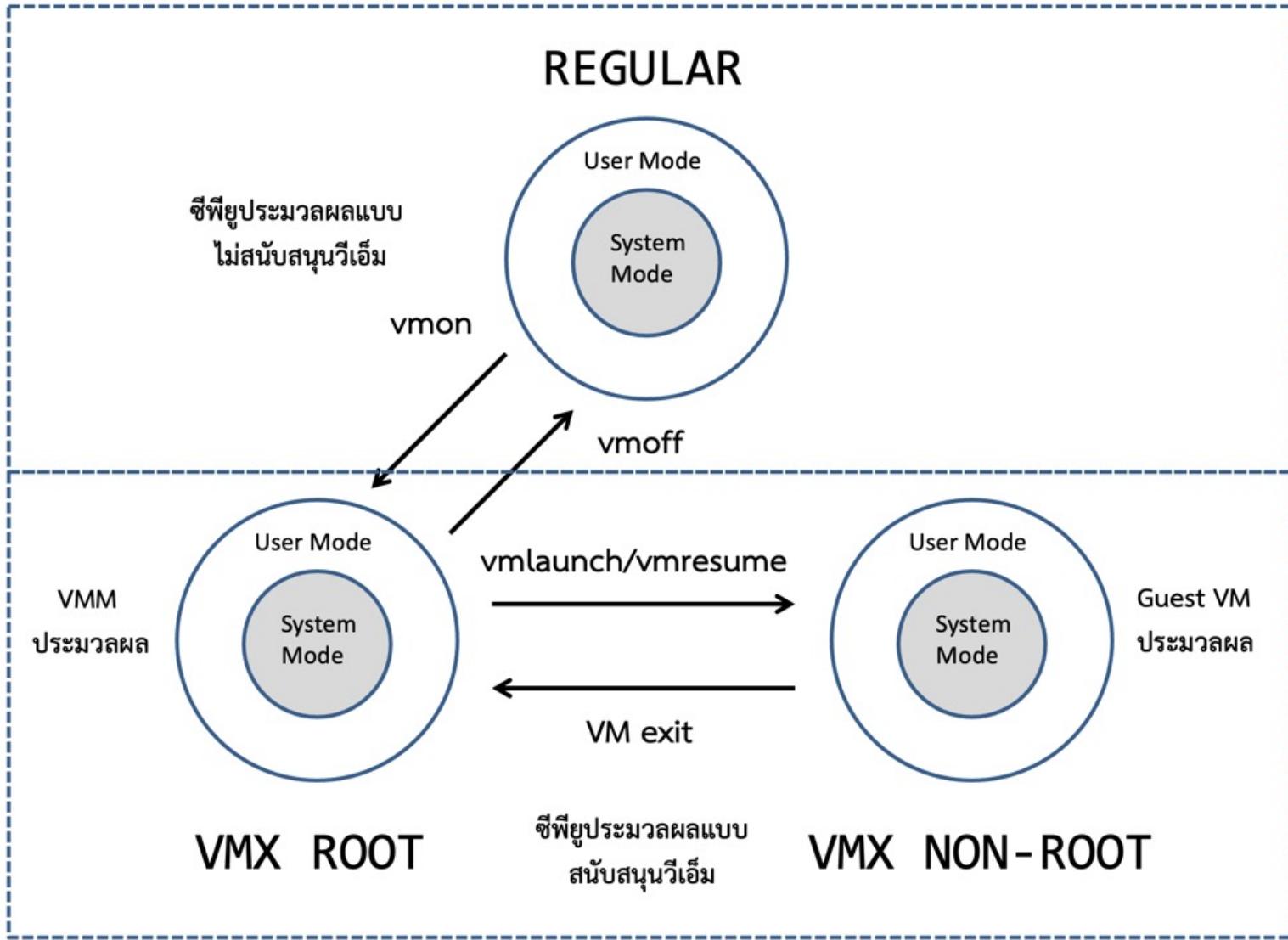
โหมดการประมวลผลของชิปปู



- Mode คือสถานะการทำงานของ CPU
- ตัวอย่าง CPU Ring ของ CPU Intel x86 32 bits

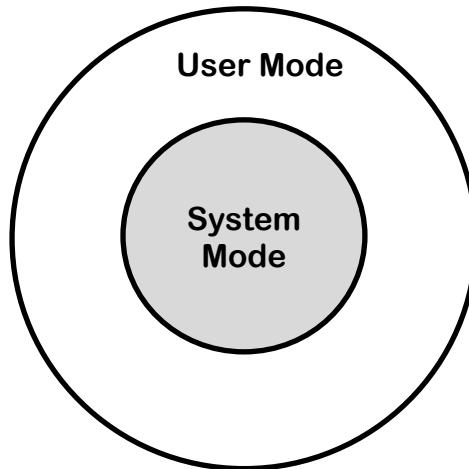


ໂຄນດກາປະໜມວລົດຂອງໜີ່ພູ້ (ແບບສັນບສຸນ Virtualization)





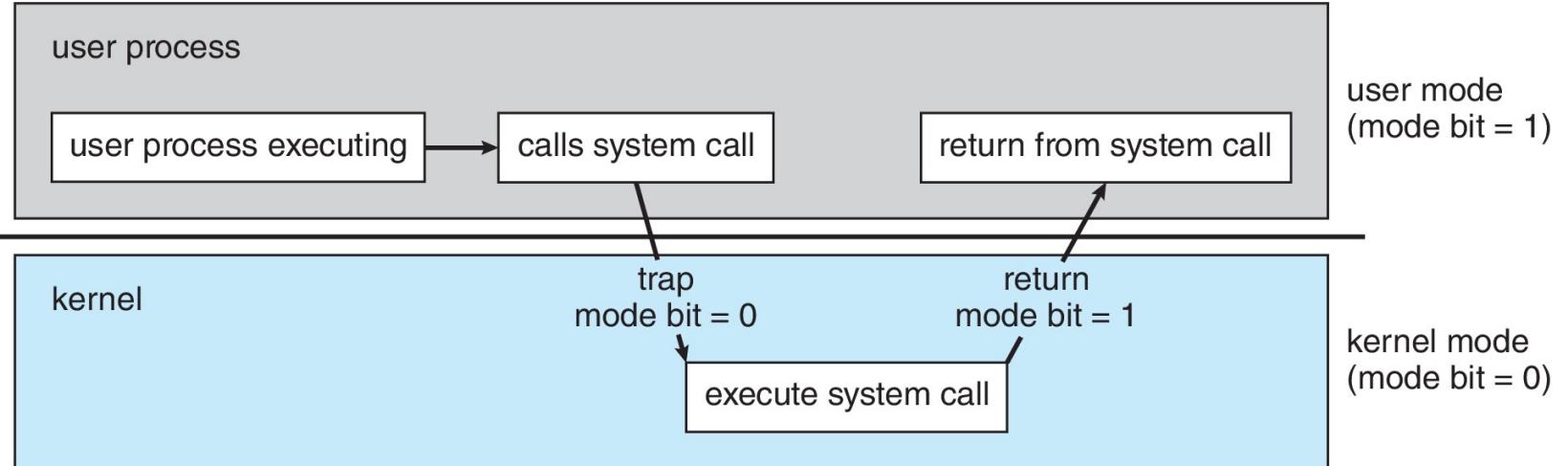
เราจะ FOCUS ที่ Dual Mode ซีพียู



- Mode คือสถานะการทำงานของ CPU

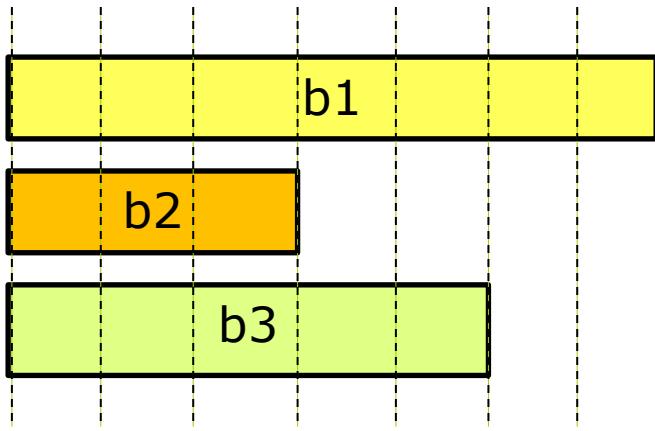


Transition from User to Kernel Mode



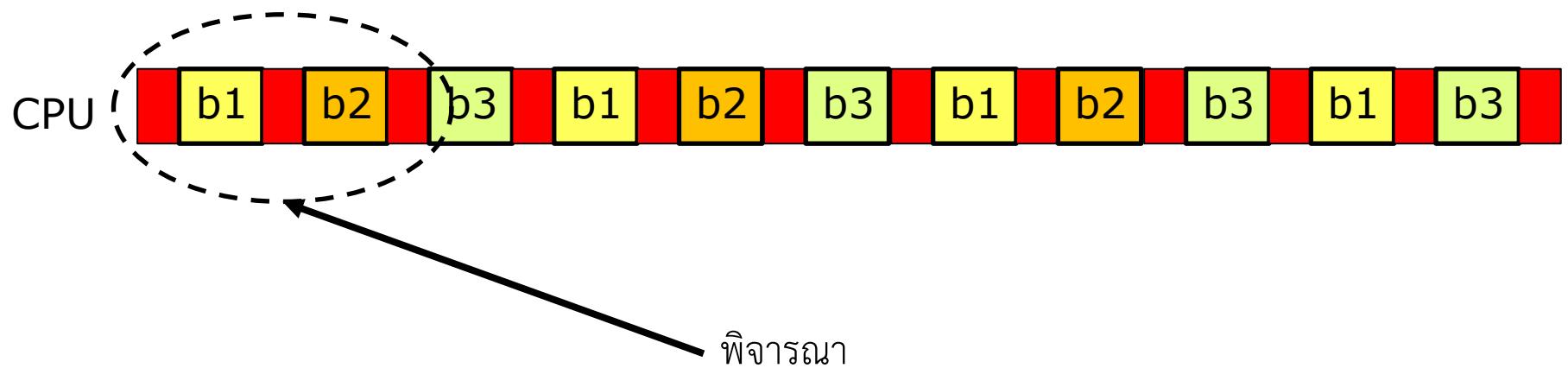
Timer

- Timer to prevent infinite loop (or process hogging resources)
 - Timer is set to interrupt the computer after some time period (เรียกว่า Time Slice)
 - Keep a counter that is decremented by the physical clock
 - Operating system set the counter (**privileged instruction**)
 - When counter zero generate an interrupt
 - ▶ นับเวลาอยหลัง เมื่อถึง 0 จึงส่งสัญญาณ interrupt
 - ▶ บางแบบใช้วิธีตั้งเวลาแบบนาฬิกาปลุก ถึงเวลา ก็ส่ง interrupt
 - Set up before scheduling process to regain control or terminate program that exceeds allotted time



OS แบ่งเวลาเรียกว่า
Time slice < 1 วินาที
ให้แต่ละ process

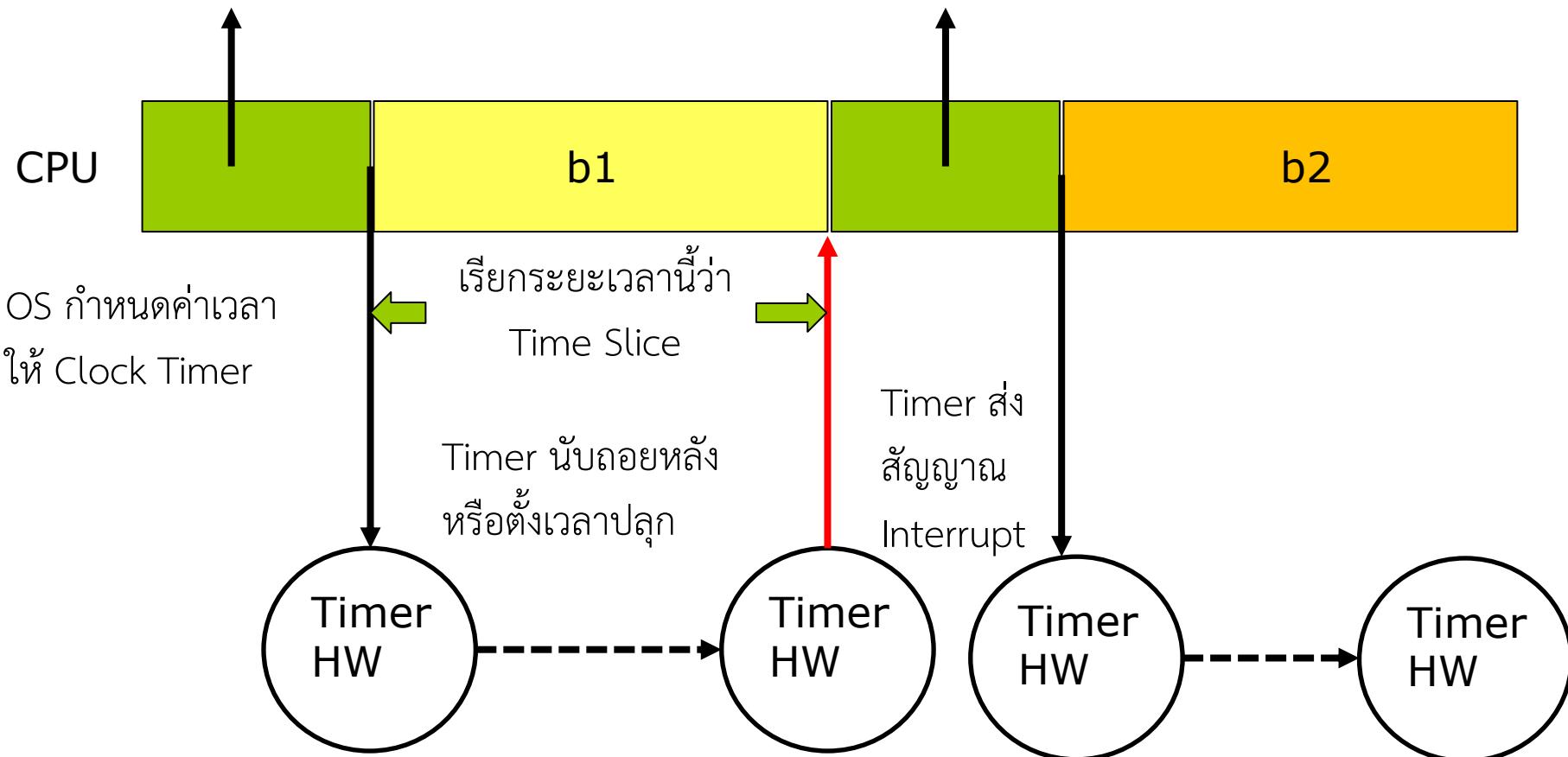
OS/Job Scheduler



เวลา

OS เลือก process
เข้ามาให้ซีพียูประมวลผล

Timer Interrupt
Handler ประมวลผลและ
ส่งงานต่อให้ OS เลือก Process ใหม่



Process Management

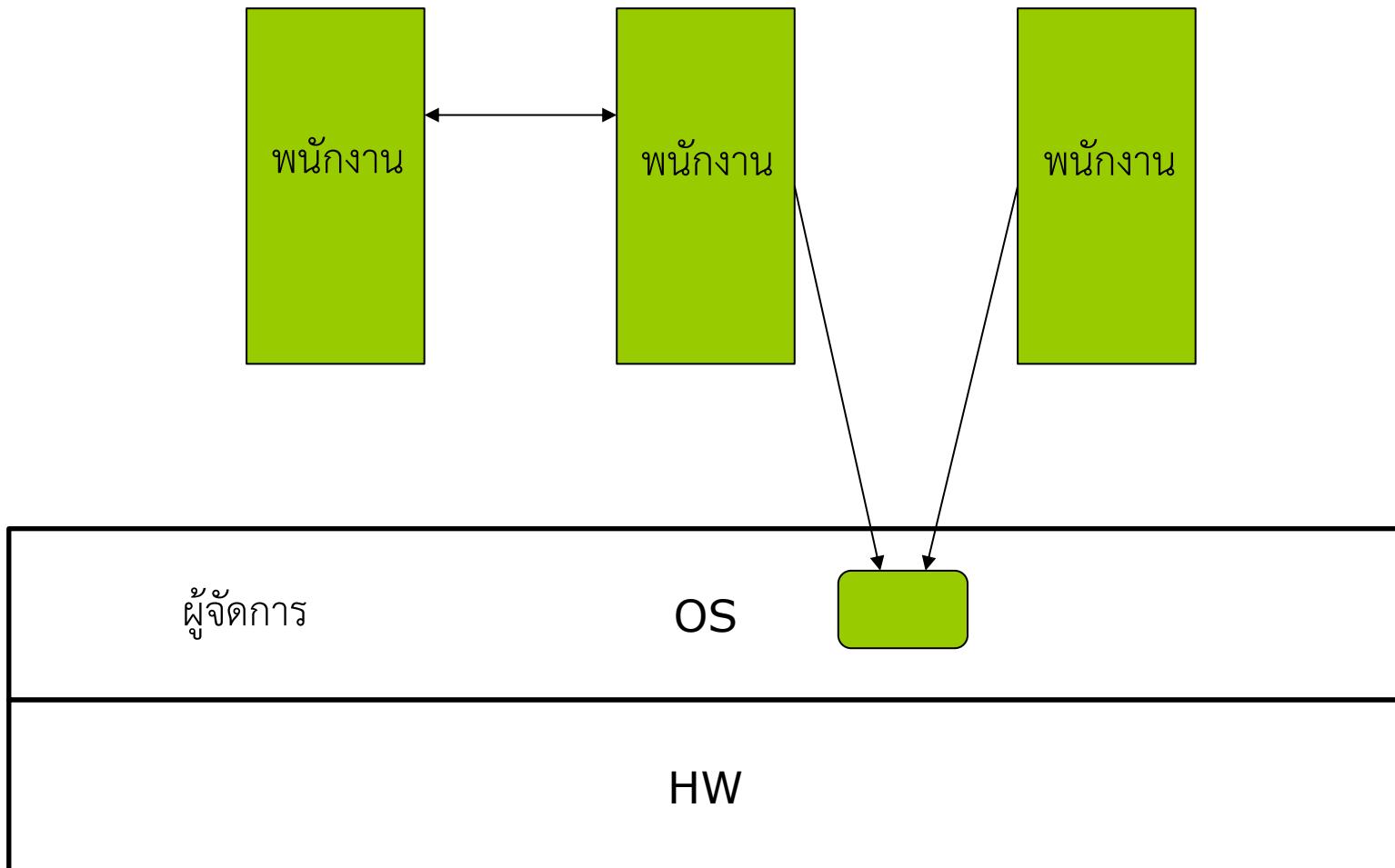
- A process is a program in execution. It is a unit of work within the system. Program is a ***passive entity***; process is an ***active entity***.
- Process needs resources to accomplish its task
 - CPU, memory, I/O, files
 - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
 - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
 - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

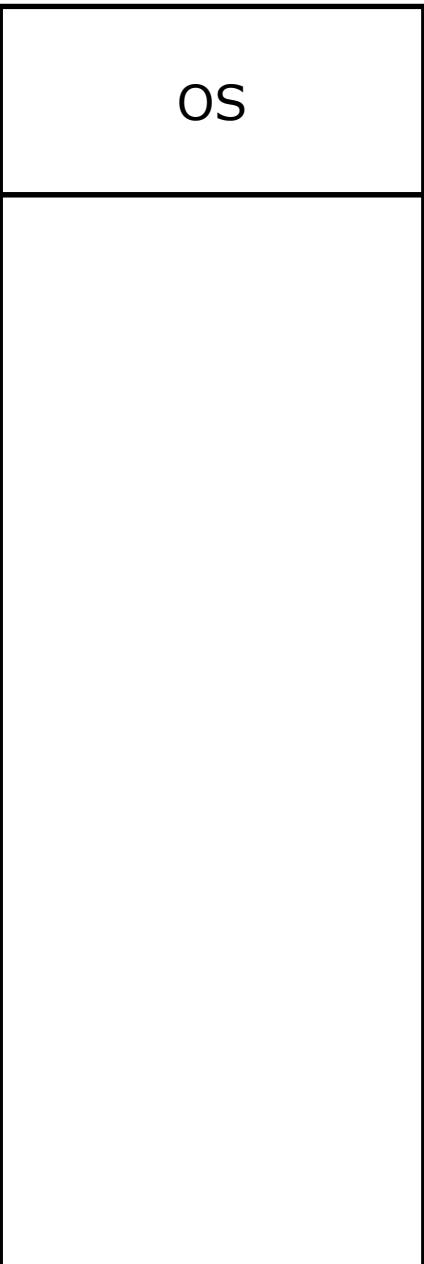
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

OS គឺជាដក្ឋាន ឱ្យខ្សោតិំទរពយារកទាំងអស់



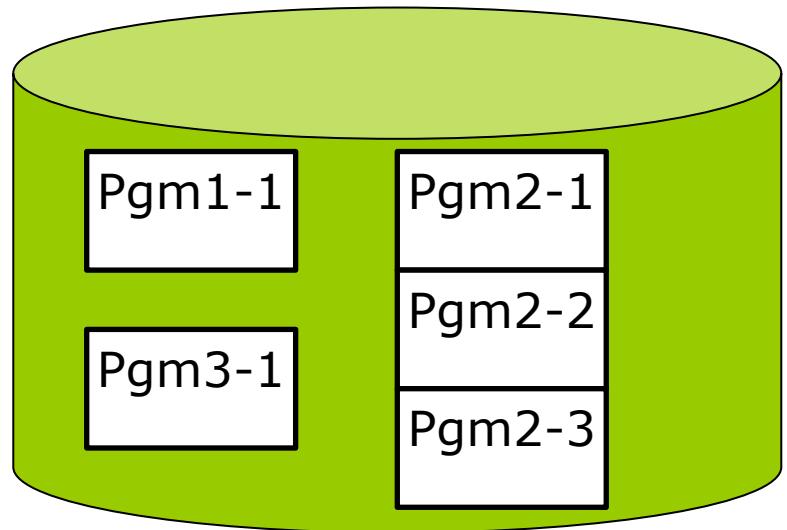
Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory
- Memory management determines what is in memory and when
 - Optimizing CPU utilization and computer response to users
- Memory management activities
 - Keeping track of which parts of memory are currently being used and by whom
 - Deciding which processes (or parts thereof) and data to move into and out of memory
 - Allocating and deallocating memory space as needed



ผู้จัดการการใช้งานหน่วย
ความจำหลัก

Keep track การใช้งาน
หน่วยความจำหลัก



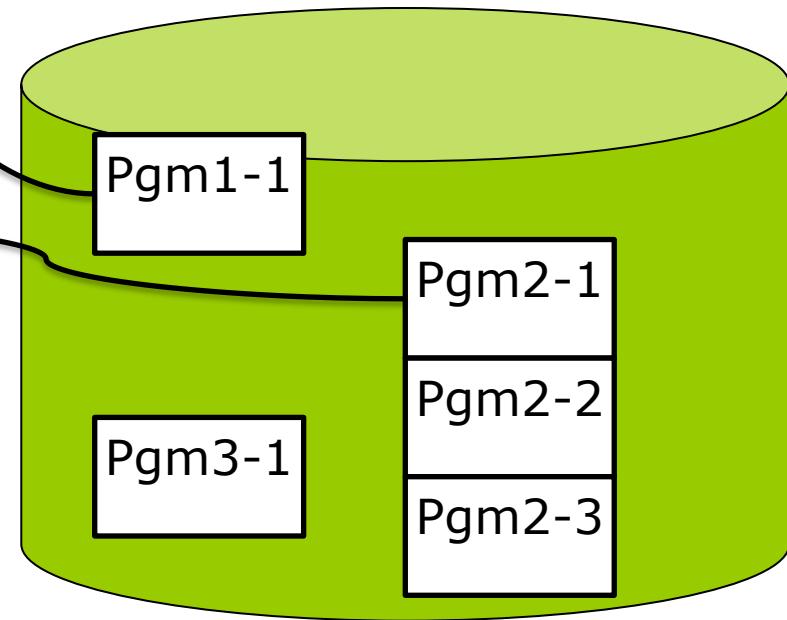


1) นำเข้า (โหลดส่วนของโปรแกรม)

2) นำออก (โหลดส่วนของโปรแกรม)

เมื่อหน่วยความจำเต็ม

3) ขยาย/ลด พื้นที่การใช้งาน



File-system Management

- OS provides uniform, logical view of information storage
 - Abstracts physical properties to logical storage unit - **file**
 - Each medium is controlled by device (i.e., disk drive, tape drive)
 - ▶ Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
 - Files usually organized into directories
 - **Access control** on most systems to determine who can access what
 - OS activities include
 - ▶ Creating and deleting files and directories
 - ▶ Primitives to manipulate files and directories
 - ▶ Mapping files onto secondary storage (**Mount**)
 - ▶ Backup files onto stable (non-volatile) storage media

ระบบจัดการไฟล์

bin

data

home

user1

user2

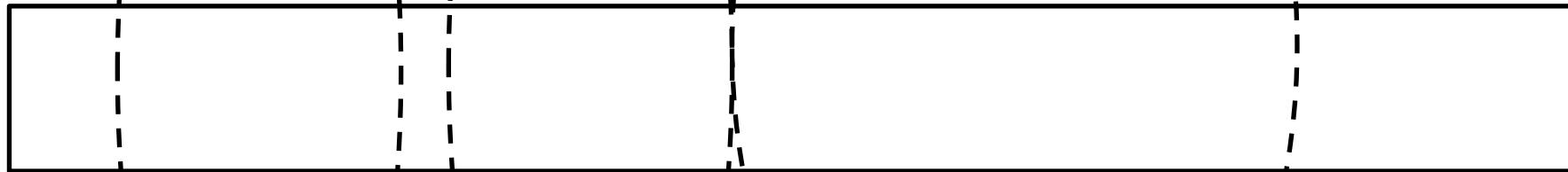
ระบบจัดการ ไอโอ



ดูแลไอโอ
อื่นด้วย

ระบบจัดการไฟล์

C:
D:
E:
user1
user2



ระบบจัดการ ไอโอ



ดูแลไอโอ
อื่นด้วย

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
 - Mounting and unmounting: บริการ map FS บน Devices
 - Free-space management: บริหารจัดการ การใช้พื้นที่ว่างใน storage
 - Storage allocation: บริหารจัดการ จัดสรรพื้นที่
 - Disk scheduling: บริหารการให้บริการ อ่านเขียนข้อมูล จะทำงานไห่งก่อนหลัง
 - Partitioning: การแบ่งพื้นที่ Storage การสร้าง Storage เสมือน
 - Protection: การตรวจความถูกต้อง การพิสูจน์ตัวตน การเข้ารหัสข้อมูล การสำรองข้อมูล การสร้างข้อมูลซ้ำซ้อน การสร้างความทนทานต่อความผิดพลาด

ระบบจัดการไฟล์



device 1



device 2

ระบบจัดการ ไอโอ

ดูแลไอโอ
อื่นด้วย

/

ระบบจัดการไฟล์

Create a file system
and root / directory



device 1



device 2

ระบบจัดการ ไอโอ

ดูแลไอโอ
อื่นด้วย

ระบบจัดการไฟล์
mounting

Mount / (root) To device 1

MP1: / (root) → device 1

ระบบจัดการ ไอโอ



device 2

ดูแลไอโอ
อื่นด้วย

device 1



MP = Mount Point เก็บข้อมูล Mapping ของ FS → Dev

ระบบจัดการไฟล์
mounting

Create a sub directory /bin

MP1: / (root) → device 1



device 1



device 2

ระบบจัดการ ไอโอ

ดูแลไอโอ
อื่นด้วย

ระบบจัดการไฟล์

Create /home

MP1: / (root) → device 1

bin

home



device 1



device 2

ระบบจัดการ ไอโอ

ดูแลไอโอ
อื่นด้วย

ระบบจัดการไฟล์

bin

mount /home dev2

MP1: / (root) → device 1
MP2: /home → device 2

home

ระบบจัดการ ไอโอ



device 2

ดูแลไอโอ
อื่นด้วย



device 1

ระบบจัดการไฟล์

bin

home

user1

user2

Create user1&2 subdir

MP1: / (root) → device 1

MP2: /home → device 2

ระบบจัดการ ไอโอ

ดูแลไอโอ
อื่นด้วย



device2



device1

ระบบจัดการไฟล์

bin

home

user1

user2

Add device 3

MP1: / (root) → device 1

MP2: /home → device 2



device 1



device 3



device 2

ระบบจัดการ ไอโอ

ดูแลไอโอ
อื่นด้วย

ระบบจัดการไฟล์

create /data subdir

MP1: / (root) → device 1
MP2: /home → device 2

bin

data

home

user1

user2

ระบบจัดการ ไอโอ



device 1



device 3



device 2

ดูแลไอโอ
อื่นด้วย

ระบบจัดการไฟล์

mount /data device3

MP1: / (root) → device 1
MP2: /home → device 2
MP3: /data → device 3

bin

data

home

user1

user2

ระบบจัดการ ไอโอ

ดูแลไอโอ
อื่นด้วย



device 1



device 3

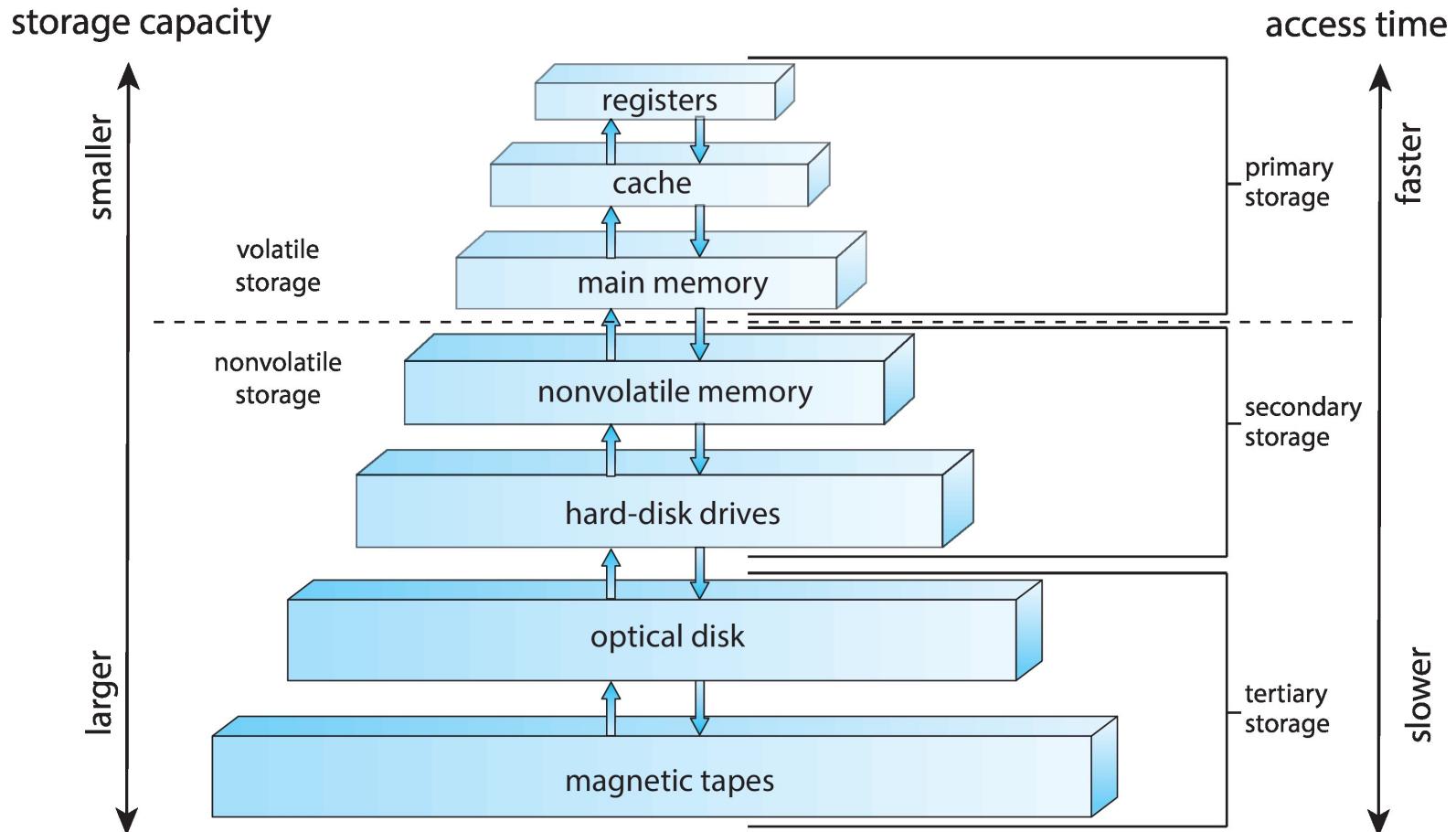


device 2

Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
 - If it is, information used directly from the cache (fast)
 - If not, data copied to cache and used there
- Cache smaller than storage being cached
 - Cache management important design problem
 - Cache size and replacement policy

Storage-Device Hierarchy



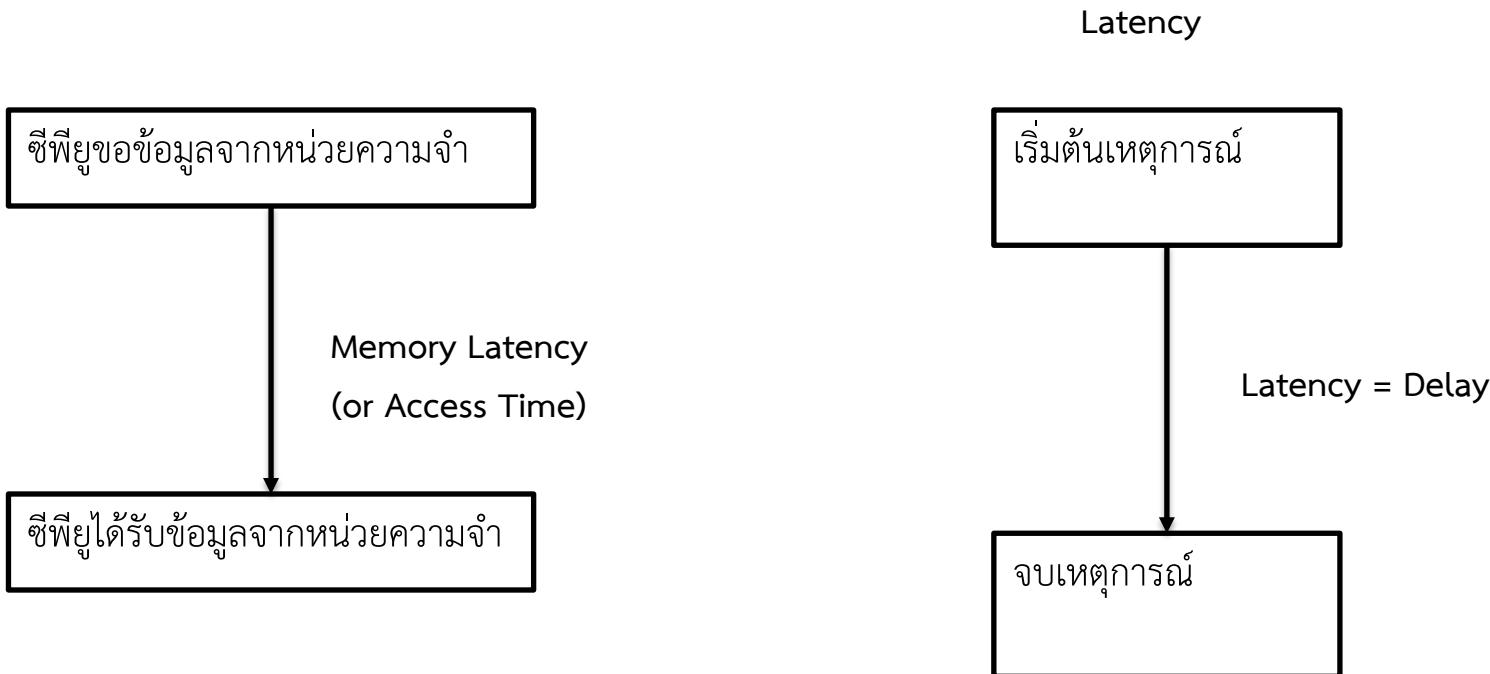
Characteristics of Various Types of Storage

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

Movement between levels of storage hierarchy can be explicit or implicit

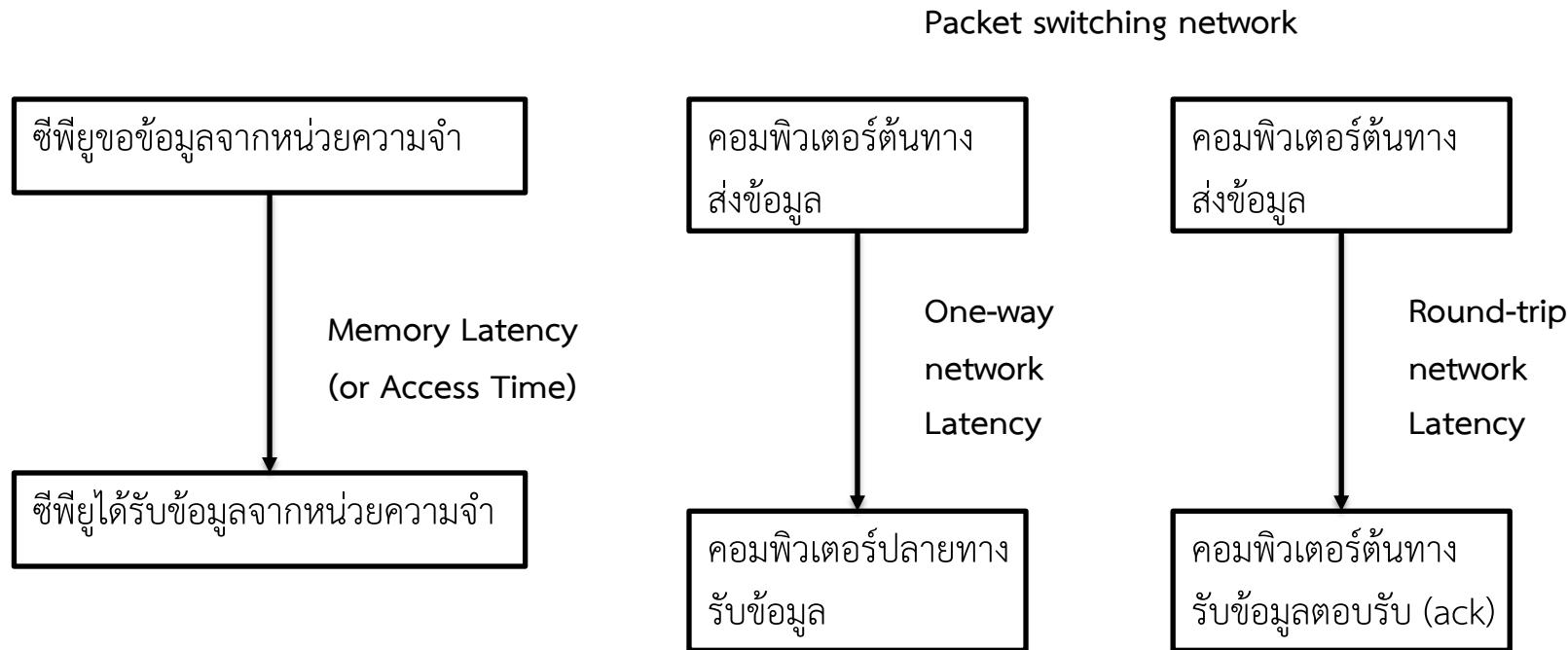
Access Time and Latency

- Memory access time คือระยะเวลาที่ใช้ในการอ่านข้อมูลจากหน่วยความจำ
- Latency คือระยะเวลาของการ delay ที่จะทำอะไรอย่างหนึ่ง



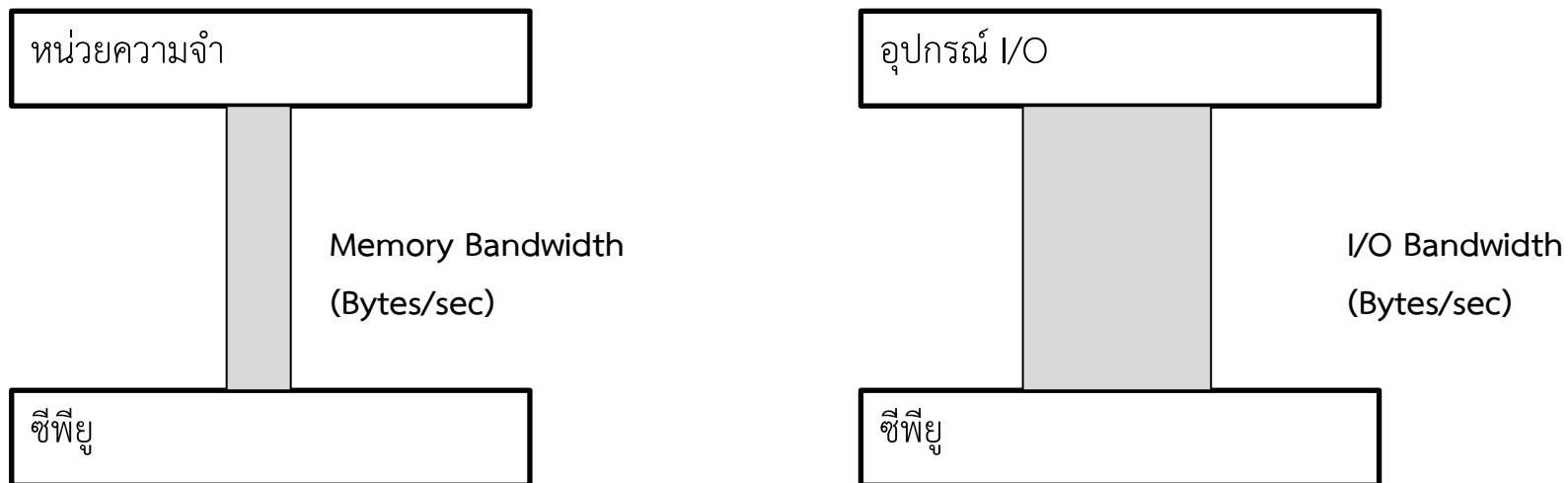
Access Time and Latency

- Memory access time คือระยะเวลาที่ใช้ในการอ่านข้อมูลจากหน่วยความจำ
- Latency คือระยะเวลาของการ delay ที่จะทำอะไรอย่างหนึ่ง



Bandwidth

- Memory bandwidth คือ อัตราการอ่านหรือเขียนข้อมูลระหว่าง ชิปีญ กับ หน่วยความจำ วัดเป็น bytes/sec (https://en.wikipedia.org/wiki/Memory_bandwidth)
- I/O Read bandwidth อัตราการอ่านข้อมูลจากอุปกรณ์ I/O สู่ ชิปีญ
- I/O Write bandwidth อัตราการเขียนข้อมูลจาก ชิปีญ ออกสู่ อุปกรณ์ I/O



ศึกษาเพิ่มเติม

- ค้นคว้าหา memory access time และ memory bandwidth ของ DDR4 และ DDR5

Bandwidth computation and nomenclature [edit]

The nomenclature differs across memory technologies, but for commodity **DDR SDRAM**, **DDR2 SDRAM**, and **DDR3 SDRAM** memory, the total bandwidth is the product of:

- **Base DRAM clock frequency**
- **Number of data transfers per clock**: Two, in the case of "double data rate" (DDR, DDR2, DDR3, DDR4) memory.
- **Memory bus (interface) width**: Each DDR, DDR2, or DDR3 memory interface is 64 bits wide. Those 64 bits are sometimes referred to as a "line."
- **Number of interfaces**: Modern personal computers typically use two memory interfaces (**dual-channel** mode) for an effective 128-bit bus width.

For example, a computer with dual-channel memory and one DDR2-800 module per channel running at 400 MHz would have a theoretical maximum memory bandwidth of:

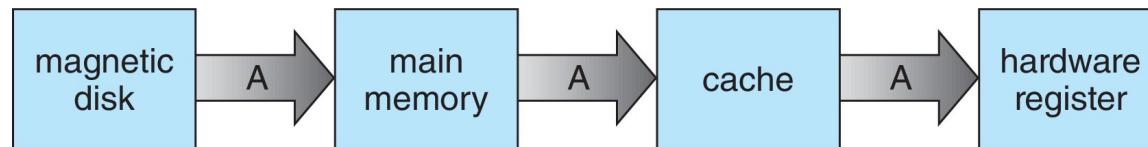
400,000,000 clocks per second \times 2 lines per clock \times 64 bits per line \times 2 interfaces =

102,400,000,000 (102.4 billion) bits per second (in bytes, 12,800 MB/s or 12.8 GB/s)

(https://en.wikipedia.org/wiki/Memory_bandwidth)

Migration of data “A” from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy

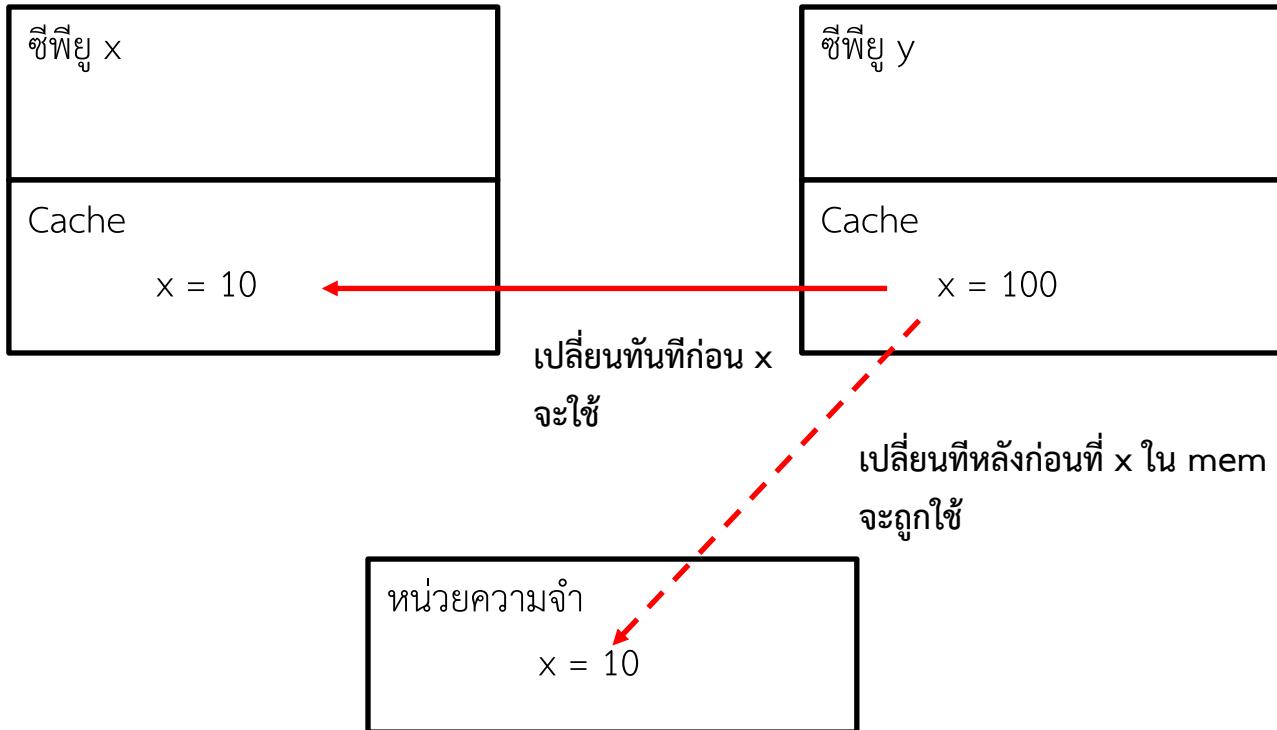


- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
 - Several copies of a datum can exist
 - Various solutions covered in Chapter 19
 - ยกตัวอย่างเช่น cookies ใน browser และ web caching

cache coherency

- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache

|



I/O Subsystem

- One purpose of OS is to hide peculiarities (ลักษณะที่แปลกและแตกต่าง) of hardware devices from the user
 - User ความองเห็น I/O devices และมีวิธีการเข้าถึงแบบเดียวกัน
- I/O subsystem responsible for
 - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
 - General device-driver interface
 - Drivers for specific hardware devices

ระบบจัดการไฟล์

ระบบจัดการ ไอโอ

device driver interfaces (มีหลายแบบ ขึ้นอยู่ชนิดของ I/O)

HDD Driver

Memory buffer
For Disk Caching

Printer driver

Spooler

NIC Driver

Memory buffer
For Network packets



Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
 - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
 - User identities (**user IDs**, security IDs) include name and associated number, one per user
 - User ID then associated with all files, processes of that user to determine access control
 - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
 - **Privilege escalation** allows user to change to effective ID with more rights
 - ▶ เจ้าของไฟล์หรือ admin อนุญาตให้ ผู้อื่นมีสิทธิ ใช้โปรแกรมหรือข้อมูลบางอย่างของตนได้ (แบบชั่วคราวหรือเฉพาะกิจ) โดยเพิ่มสิทธิการใช้งานให้
 - ▶ ช่วยเพิ่มการใช้งาน แต่ต้องระวังไม่ให้ผิดพลาด



Protection & Security

- **Protection** คือกลไก (Mechanism) ที่ OS ใช้กำหนดว่า กระบวนการทำงาน(Process) หรือผู้ใช้ (User) ได สามารถใช้งานทรัพยากรได้ได้บ้าง
 - ใช้ความเป็นเจ้าของเป็นเครื่องมือ
- **Security** คือกลไกสำหรับป้องกันการโจมตีจากภายในและภายนอก
 - ภายนอก: hackers พยายามเดา passwords, DDOS, etc
 - ภายใน: User A พยายามเข้าถึงข้อมูลของ User อื่น
- OS แยกสิทธิของ User แต่ละคนมีสิทธิต่างกัน
 - แต่ละคนมี User ID
 - User ID จะถูกแบ่งเข้ากับสิ่งต่างๆในระบบ เช่น ไฟล์ และ Process เพื่อแสดงว่า User คนนั้นเป็นเจ้าของสิ่งเหล่านั้น
 - Group ID คือ การที่ OS สามารถกำหนดให้ Users มีกลุ่ม และใช้ ID นี้กำหนดว่า ไฟล์ และ Process ในระบบ อยู่ในกลุ่มใด
 - มีกลไก **Privilege Escalation** ที่ใช้เปลี่ยนสิทธิของ User จาก User หนึ่งเป็นอีก user หนึ่งชั่วคราว เพื่อทำงานได่ง่ายหนึ่ง





Privilege Escalation เปรียบเทียบ

- สมมุติว่า พนักงานทั่วไปคนหนึ่ง เป็นพนักงานบริษัททั่วไป เวลาทำงานในบริษัทเขาก็จะมีสิทธิเข้าถึง ห้องต่างๆ ได้จำกัด
- ผู้จัดการท่านนั้นที่มีสิทธิ เข้าห้องเก็บความลับได้
- พนักงานทั่วไป ไม่มีสิทธิเข้าห้องเก็บข้อมูลความลับของบริษัท
- แต่ เวลาที่พนักงานทำงาน A พนักงานจำเป็นต้องเข้าห้องเก็บความลับจริงจะทำงานให้เสร็จได้
- **Privilege Escalation:** ผู้จัดการแก้ปัญหา โดยเปลี่ยนสิทธิของพนักงานให้เป็นผู้จัดการชั่วคราว เวลาที่ทำงาน A **เท่านั้น**
- พนักงานสามารถทำงาน A ได้ เมื่อเสร็จแล้วจึงกลับมา มีสิทธิเหมือนเดิม





Privilege Escalation Example



- สมมุติว่า User P มี ID 1000 เวลา rันโปรแกรม Process จะมี Effective User ID = 1000 ด้วย
- User Root มี ID = 0 มีแค่ Root เท่านั้นที่อ่านไฟล์ X ที่ Root เป็นเจ้าของได้
- เวลา Root รัน Process Process จะมี Effective User ID = 0 ซึ่งทำให้ Process อ่านข้อมูลในไฟล์ X ได้
- User P ไม่สามารถ อ่าน เขียน ข้อมูลใน X ได้
- Root สร้างโปรแกรม A ขึ้นให้ User ทุกคนเรียกใช้ได้ **แต่โปรแกรม A ต้องอ่านข้อมูลจากไฟล์ X**
- **Privilege Escalation:** User Root กำหนดให้โปรแกรม A มี Flag พิเศษแบบ setuid ซึ่งทำให้เวลา User 1000 รันโปรแกรมนี้ OS จะสร้าง Process ที่มี Real User ID =1000 แต่มี Effective ID = 0 ที่ทำให้สามารถเข้าถึงไฟล์ได้
 - โปรแกรม A สามารถเปลี่ยน Effective User ID ไประหว่าง 1000 และ 0 ได้ ปกติมันจะเปลี่ยนเป็น 0 เฉพาะตอนที่จำเป็นเท่านั้น
- Sys Admin Root ต้องโอด่าว่าโปรแกรม A ปลอดภัย



สรุป

- ระบบเก็บข้อมูล
- การประมวลผลแบบที่มี DMA
- การประมวลผลแบบหลายโหมด
- แนะนำส่วนประกอบอื่นที่สำคัญของ OS



BACKUP & EXTRA





เงื่อนไขในการประมวลผลชุดคำสั่งของซีพีयู

- User ISA ถูกออกแบบมาให้โปรแกรมได้ใช้งานก็ได้ (โปรแกรม Apps + OS)
- System ISA ถูกออกแบบมาให้โปรแกรม OS เรียกใช้เท่านั้น
- เมื่อ OS ประมวลผลซีพีयูจะอยู่ใน System Mode
- ซีพีyuจะประมวลผลคำสั่ง System ISA ได้เมื่อมี mode = system เท่านั้น
 - ถ้ามีการรัน System ISA ในขณะที่ซีพีyuอยู่ใน user mode จะเกิด trap
- OS จะเปลี่ยนโหมดการประมวลผลของซีพีyuให้เป็น user mode เมื่อมันต้องการกำหนดให้ซีพีyuประมวลผล Application Process
- Application Process ไม่สามารถเปลี่ยน mode ของซีพีyuได้เนื่องจากคำสั่งสำหรับเปลี่ยน mode เป็น Privilege instruction
- ซีพีyuจะเข้าสู่ system mode เมื่อได้รับสัญญาณ Interrupt หรือ trap

