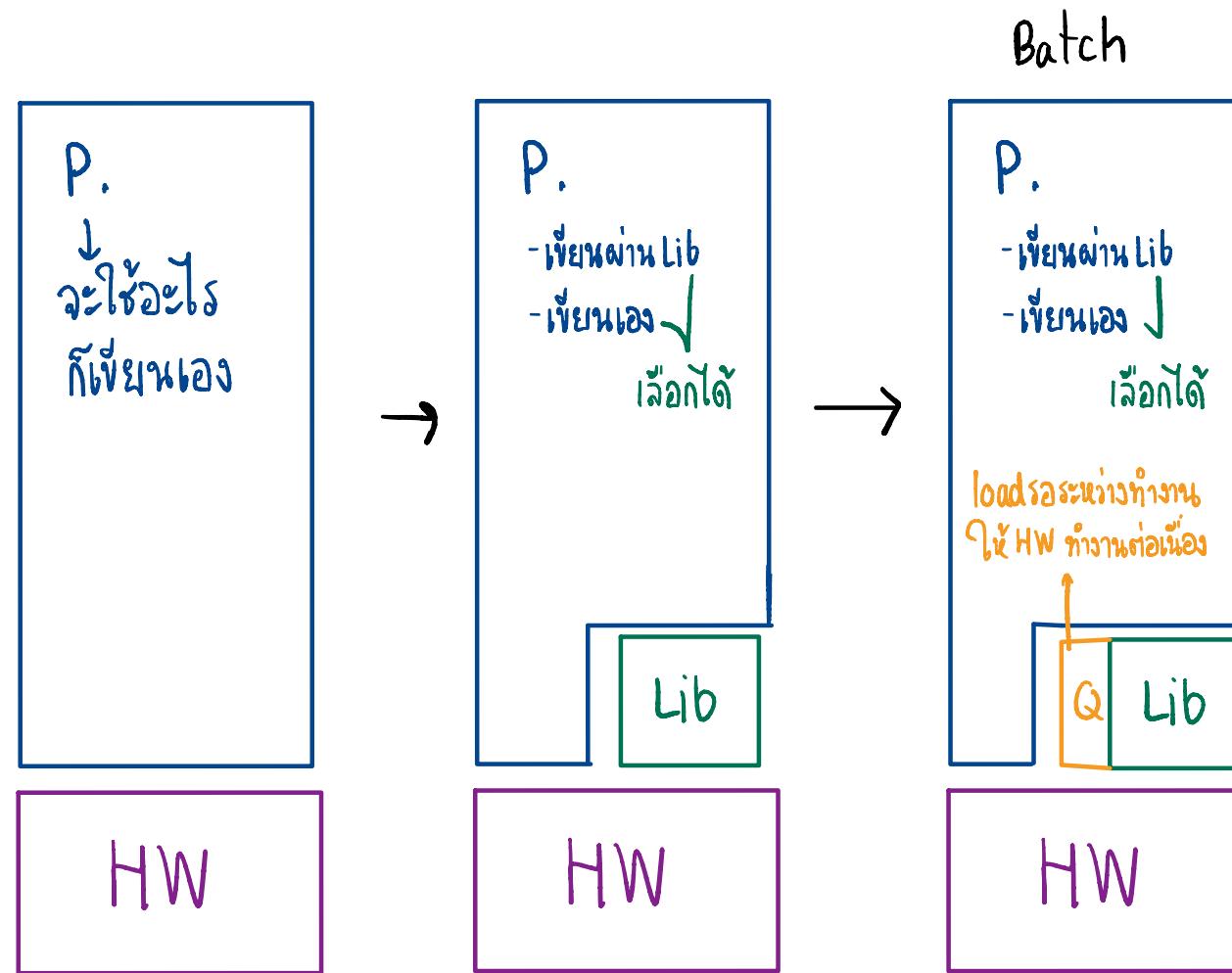


The Kernel Abstraction

Slides adopted from CSE 451 class at UW, CS162 class at Berkeley and CSE 421/521 class at UB

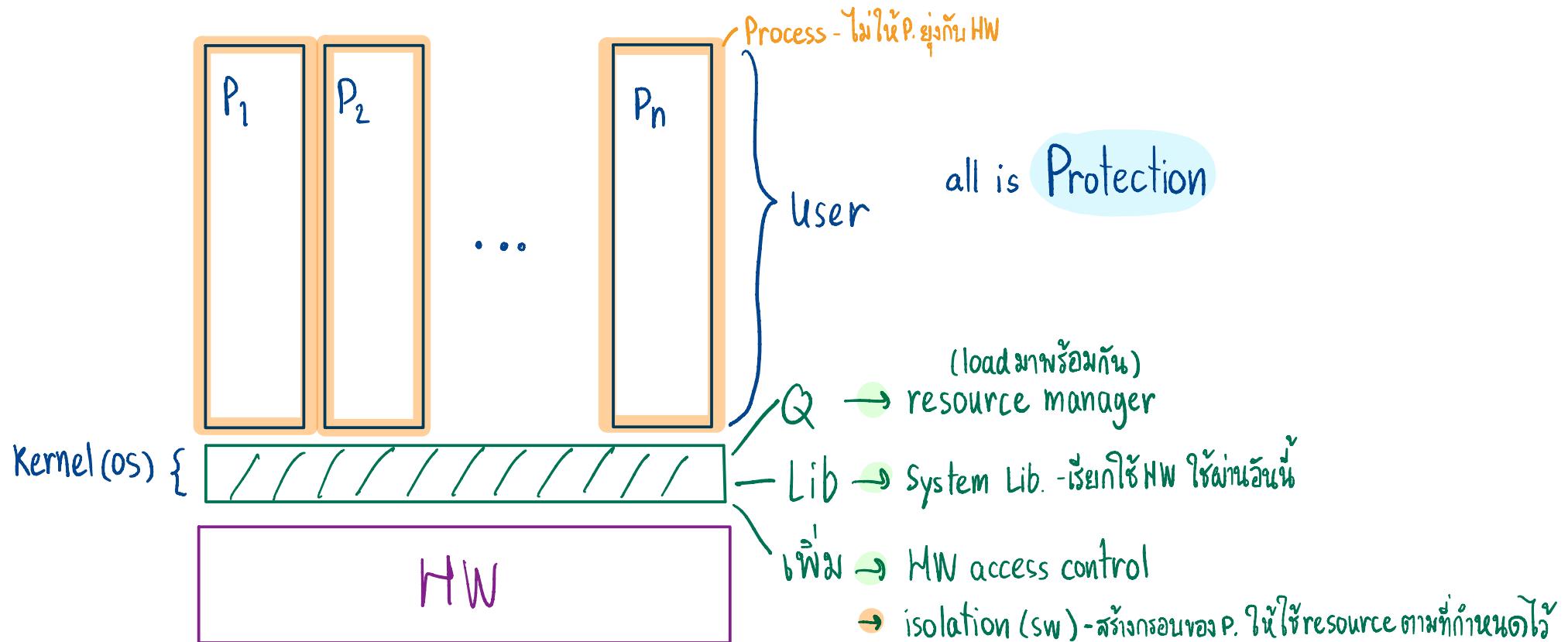
อดีต < 1970

- Single task system



ປົວຈຸບັນ > 1980

- Multitasking system → P. ทำงานพร้อมกัน > 1



Activity #1

- การเปลี่ยนแปลงจากระบบแบบ single task ไปเป็นระบบแบบ Multitask ... สิ่งที่จะต้องมีการปรับแต่งหรือเพิ่มเติมเข้ามาใน OS ได้แก่...

What does an OS do...

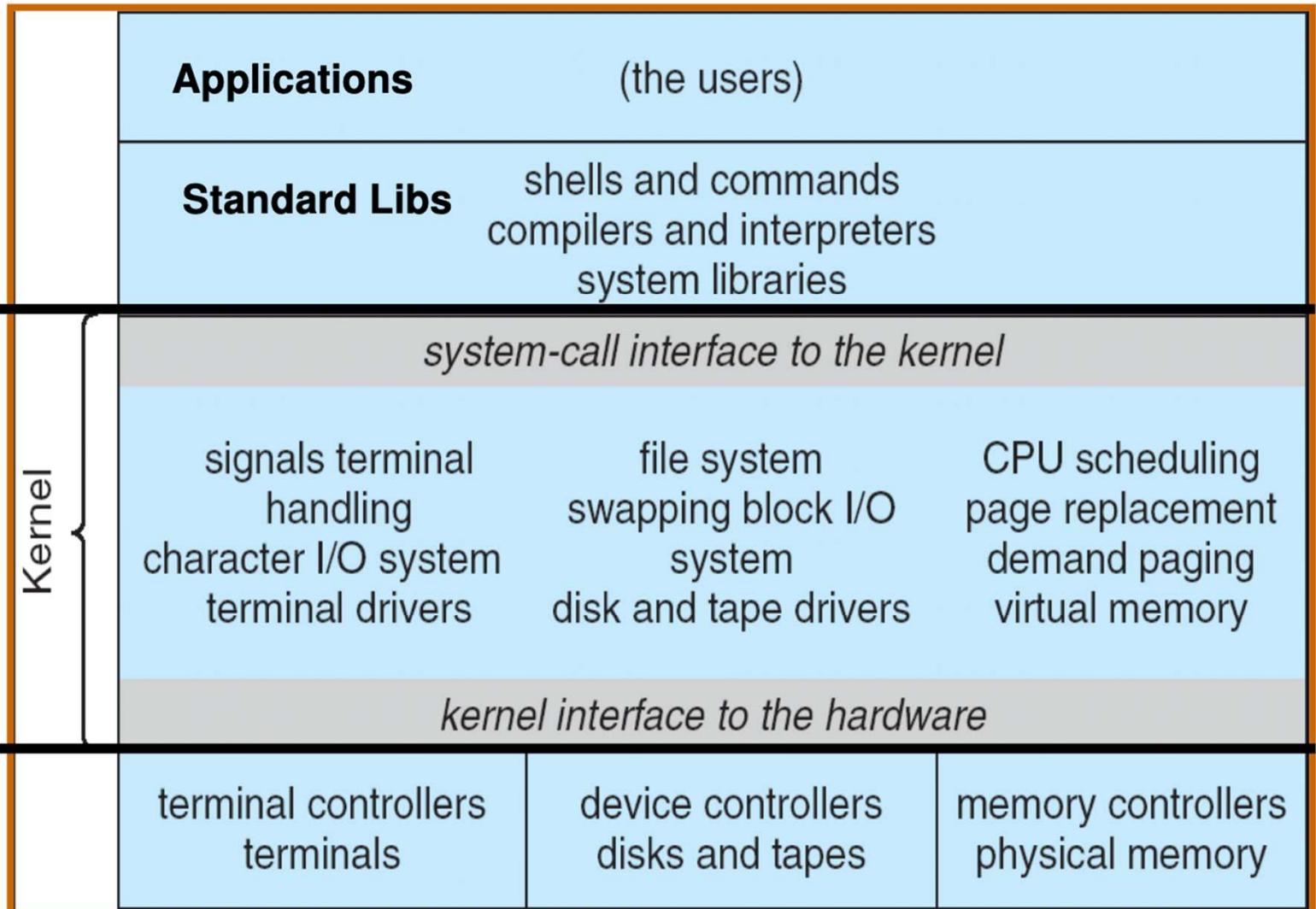
- Hiding Complexity
 - Variety of HW
 - E.g. different CPU, amount of RAM, I/O devices
 - (ទេស P.)
 - ស្ថិតិវិធី environment បានសម្រាប់ process
 - អនុញ្ញាត P. គ្រប់ resource all
- Kernel is the part of the OS that running all the time on the computer
 - Core part of OS
 - Manages system resources
 - Acts as a bridge bet. app, HW

UNIX Structure

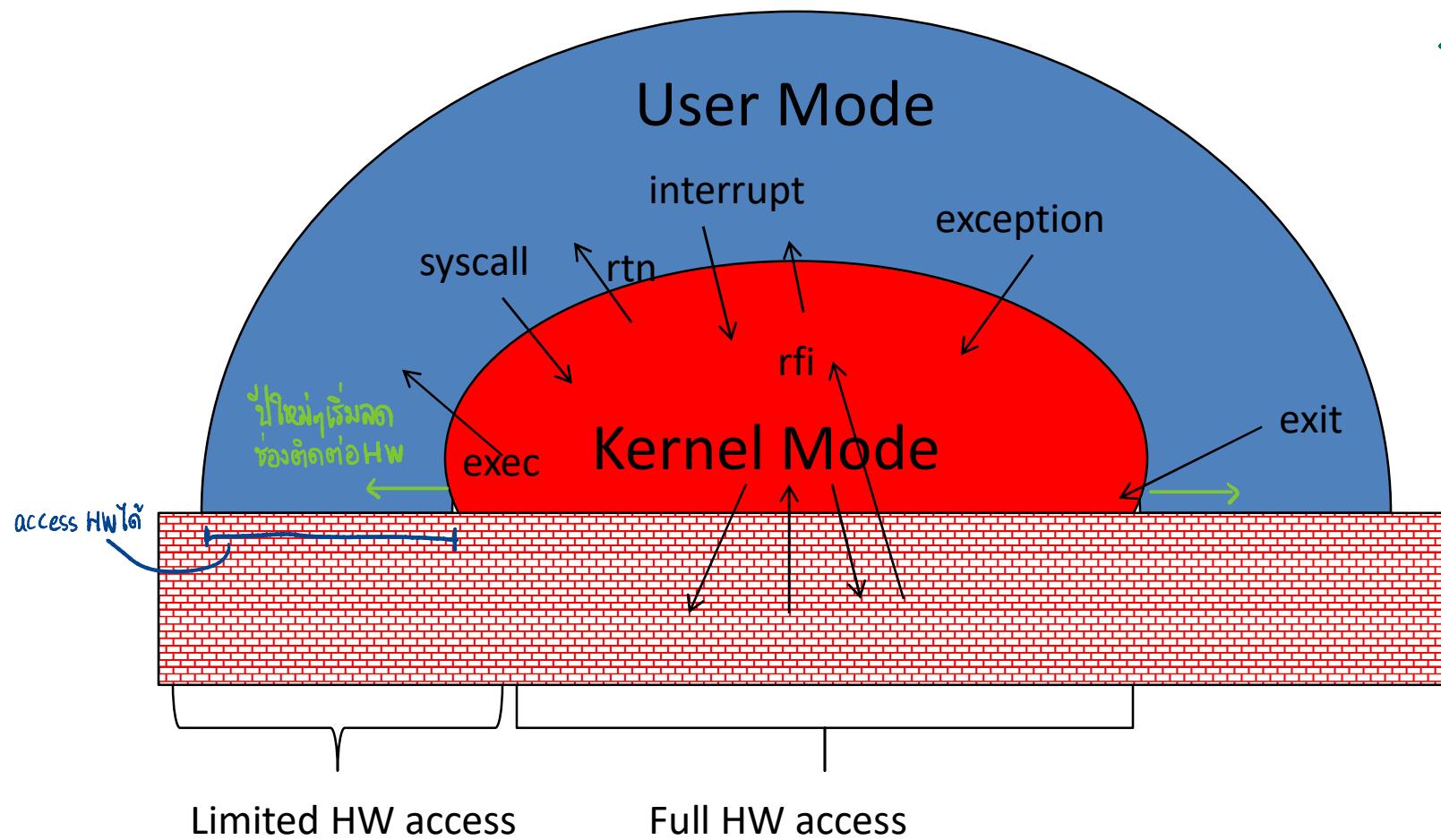
User Mode

Kernel Mode

Hardware



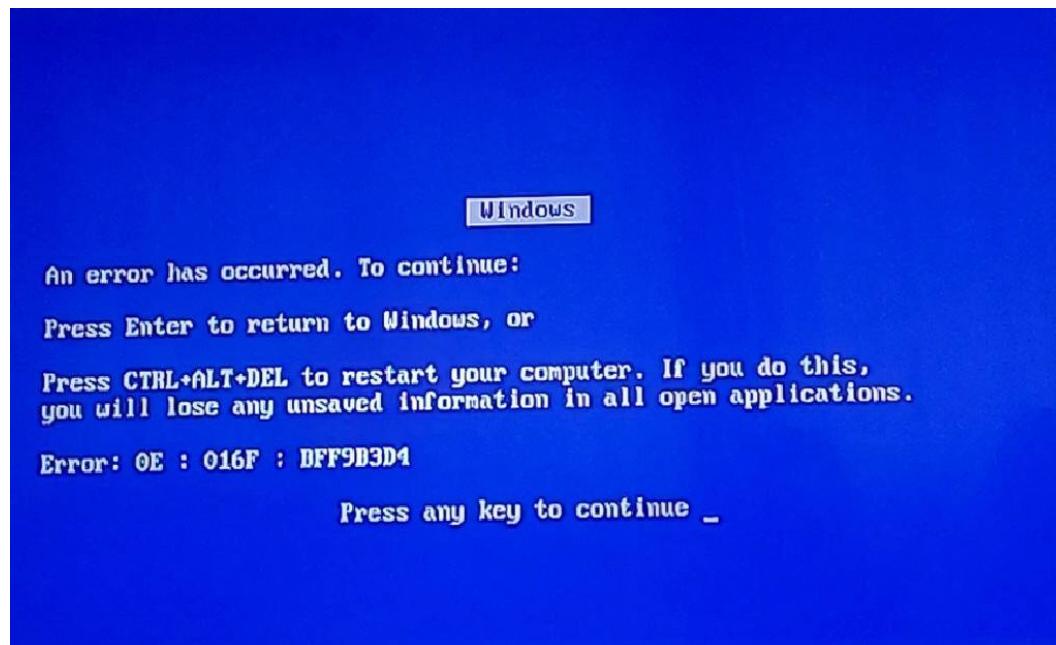
User/Kernel (Privileged) Mode



ใช้ยังไง B. ใช้ R. เก่าได้ (not all)
B. สำหรับในตั้งต่อ HW ได้
↓
HW บางตัวมีช่องนี้อยู่
Ex P.1 ใช้อยู่ P.2 ถ้ายังไม่มี
↓
รับบันไดเข้าสู่ยาร์

One of the major goals of OS is...

- Protecting **Process** and the **Kernel**
 - Running multiple programs
 - Keep them from interfering with the OS - kernel
 - Keep them from interfering with each other



→ จดฟ้า
ເກົດຈາກ OS ແລະ ປໍ່
ມາຮອນເຫັນໄວ້
ແລຍຂຽດຮະບັນ

Activity #2: Protection: WHY?

เวลา 10 นาที

การ protect Process และ Kernel ทำให้เกิด impact
อะไรกับระบบบ้าง และยังต้อง protect อะไรอีกบ้าง เพื่ออะไร

- Reliability : buggy program only hurt themselves
- Security and privacy : trust P. less
- Fairness : enforce share of disk , CPU ไม่จัดคิ่งให้ได้ก็ร

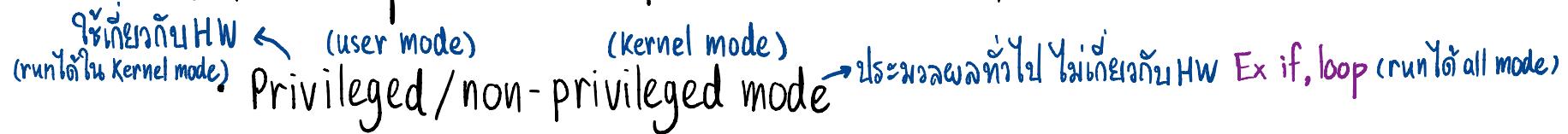
Protection: How? (HW/SW)

เวลา 10 นาที

- 2 Main HW Mechanism

- Mem addr. translation → map virtual addr. กับ addr.จริง

- Dual mode operation - microprocessor แยก成 2 กลุ่ม



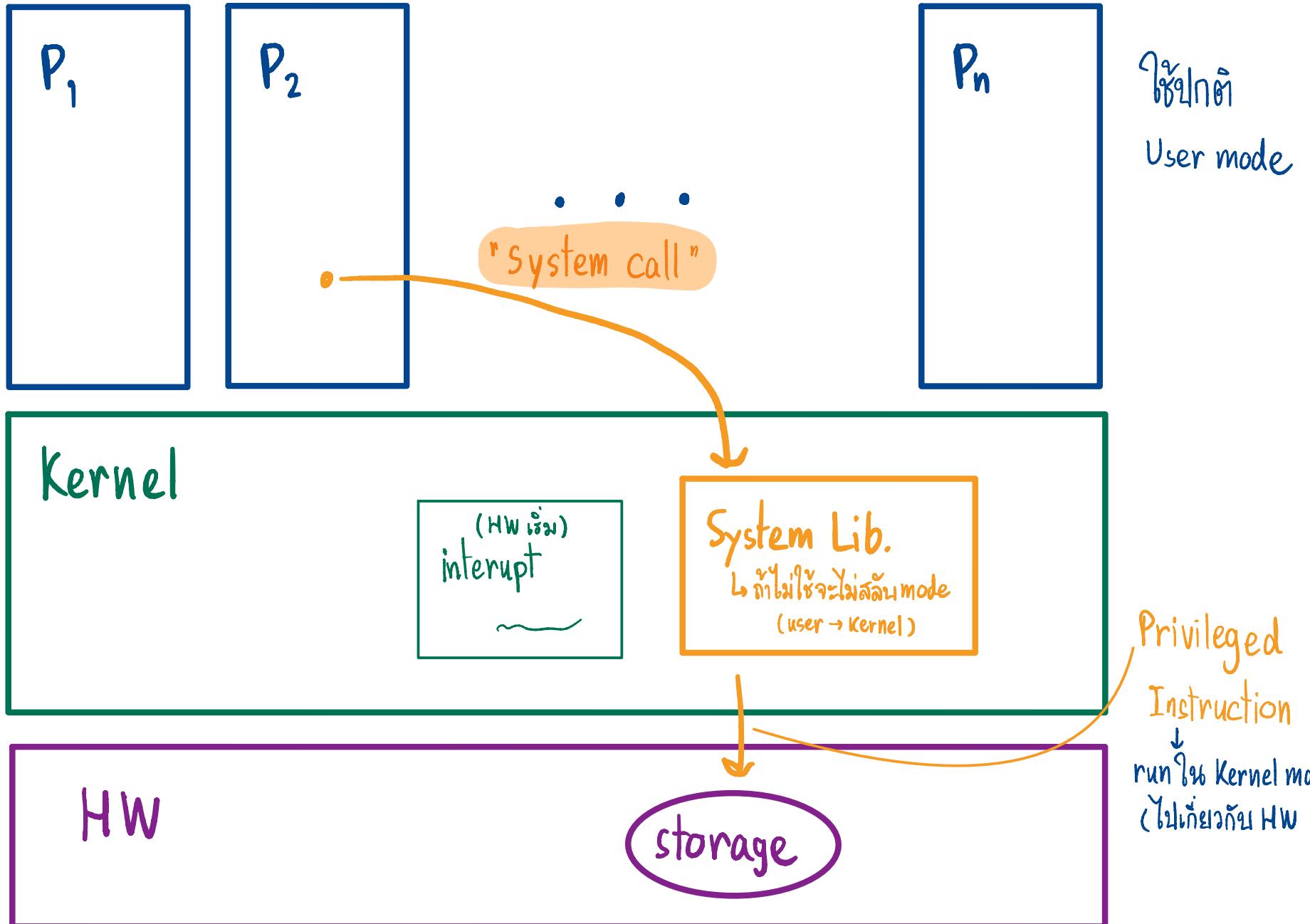
- System calls (SW ร่อง) คล้าย interrupt (HW ร่อง)

หมายความว่า interrupt (ใน Kernel)

- SW

- Process

- System calls



Hardware Support: Dual-Mode Operation

- Kernel mode
 - Execution with the full privileges of the hardware
 - Ex – Read/write to any memory, access any I/O device, read/write any disk sector, send/read any packet
- User mode $+, -, \times, \div, \text{if}, \text{loop}$
 - Limited privileges
 - Only those granted by the operating system kernel
- On the x86, mode stored in EFLAGS register
- On the MIPS, mode in the status register

Hardware Support: Dual-Mode Operation

- Privileged instructions
 - Available to kernel
 - Not available to user code
- Limits on memory accesses - เหตุที่ virtual addr. ไม่ใช่ physical addr.
 - To prevent user code from overwriting the kernel → ป้องกันไม่ให้พื้นที่ของ kernel ถูกเขียนใหม่ในพื้นที่ของ user process
- Timer กำหนดให้ P. ทำงานได้นานแค่ไหน? → ทำให้เกิด interrupt ตามกำหนดเวลา
 - เอาคืน To regain control from a user program in a loop → ให้ Kernel คิด ว่าจะทำอะไรต่อ?
- Safe way to switch from user mode to kernel mode,
and vice versa
Ex P. ทำงานครุณ 10 s → interrupt → กลับ kernel → kernel เลือก P. ที่จะทำต่อ
↑
CPU ทำงาน

Privileged instructions

- Examples?
- What should happen if a user program attempts to execute a privileged instruction?

Ans: close P. រៀបចំនេវត្បូ → ចាប់ផ្តើម

User Mode

- Application program
 - Running in process

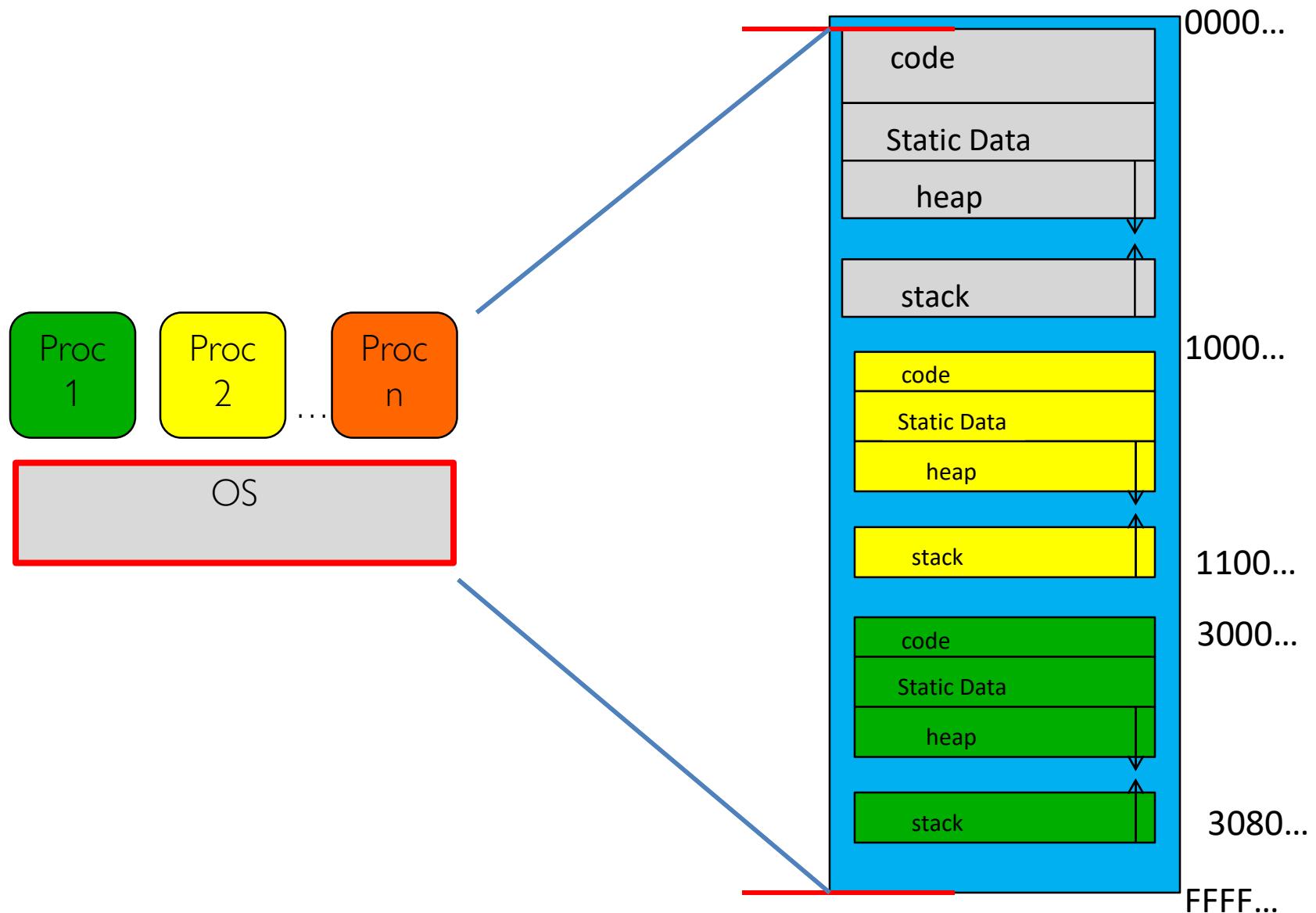
Virtual Machine: VM

- Software emulation of an abstract machine
 - Give programs illusion they own the machine
 - Make it look like HW has feature you want
- 2 types of VM
 - * – Process VM หมายความว่า OS เดี่ยวๆ กันทำงานได้
 - Supports the execution of a single program (one of the basic function of the OS)
 - System VM → จำลองระดับ HW
 - Supports the execution of an entire OS and its applications

Process VMs

- GOAL:
 - Provide an isolation to a program ป้องกัน
• Processes unable to directly impact
 - Portability (Program) รันต่าง HW ได้ Ex เพื่อน P. แล้ว run บน HW ที่ต่างกันได้

Kernel mode & User mode

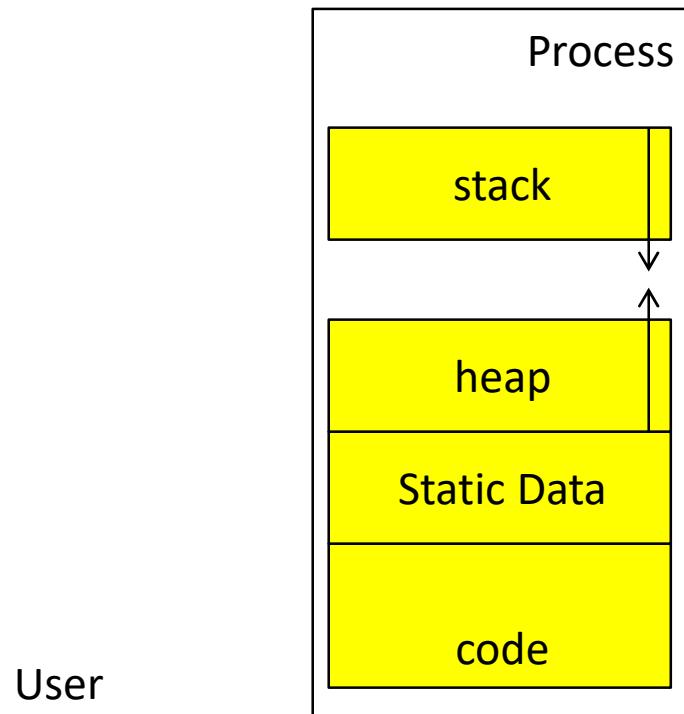


Process Abstraction

- Process: an *instance* of a program, running with limited rights
 - Thread: a sequence of instructions within a process
 - Potentially many threads per process (for now 1:1)
 - Address space: set of rights of a process *เริ่มต้น เลิกกัน*
 - Memory that the process can access
 - Other permissions the process has (e.g., which system calls it can make, what files it can access)

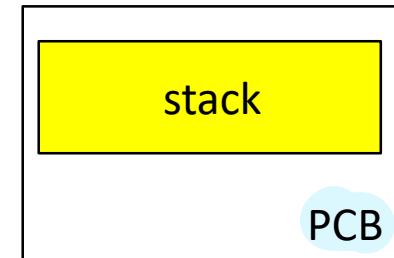
Process

- 2 parts
 - PCB in kernel
 - Others in user



User

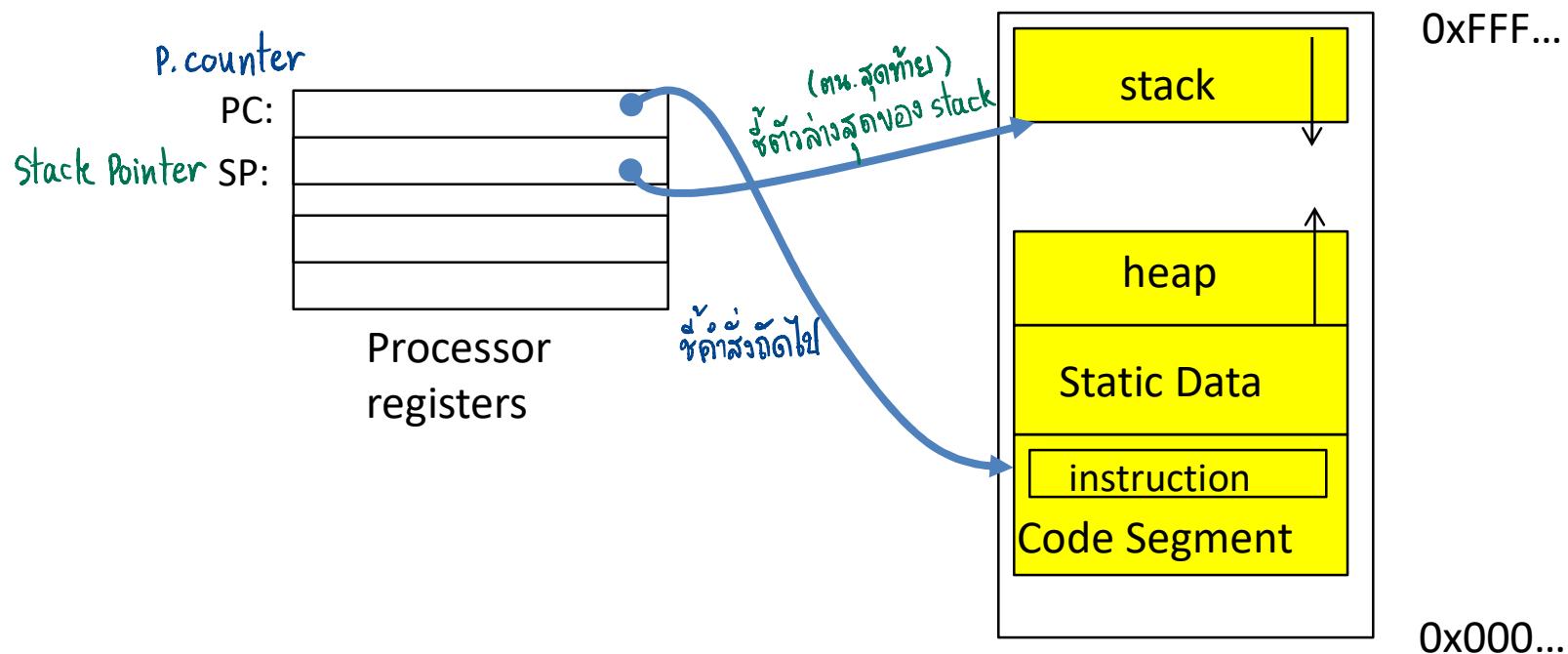
Kernel



Process Control Block: PCB

- Kernel represents each process as a process control block (PCB)
 - Status (running, ready, blocked, ...)
 - Registers, SP, ... (when not running)
 - Process ID (PID), User, Executable, Priority, ...
 - Execution time, ...
 - Memory space, translation tables, ...
- Kernel Scheduler maintains a data structure containing the PCBs
- Scheduling algorithm selects the next one to run
 - ↳ process ที่ต้องเข้ามาทำงาน

Address Space: In a Picture



Break