



ĐẠI HỌC ĐÀ NẴNG

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

# NGUYÊN LÝ HỆ ĐIỀU HÀNH



**Khoa Công nghệ thông tin**

ThS. Nguyễn Thị Lệ Quyên

D  
BACH KHOA  
NANG

# Nội dung môn học

- Giới thiệu (6 tiết)
- Tiến trình và luồng (9 tiết)
- ***Thi giữa kỳ***
- Quản lý bộ nhớ (8 tiết)
- Vào/ Ra (4 tiết)
- Hệ thống file (4 tiết)
- Thực hành (15 tiết)



ĐẠI HỌC ĐÀ NẴNG  
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

Khoa CÔNG NGHỆ THÔNG TIN  
ThS. Nguyễn Thị Lệ Quyên



GIỚI THIỆU

D  
BACH KHOA

N  
A  
N  
G

# Giới thiệu

- Hệ điều hành là gì?
- Lịch sử
- Các loại Hệ điều hành
- Phần cứng
- Thành phần Hệ điều hành
- Lời gọi hệ thống (System calls)
- Cấu trúc Hệ điều hành

# Lịch sử

- Internet

D  
BACH KHOA

N  
A  
N  
G

# Các loại hệ điều hành

- Theo loại máy tính
  - Hệ điều hành dành cho máy MainFrame
  - Hệ điều hành dành cho máy Server
  - Hệ điều hành dành cho máy nhiều CPU
  - Hệ điều hành dành cho máy tính cá nhân (PC)
  - Hệ điều hành dành cho máy PDA (Embedded OS – hệ điều hành nhúng)
  - Hệ điều hành dành cho máy chuyên biệt
  - Hệ điều hành dành cho thẻ chip (SmartCard)

# Các loại hệ điều hành

- Theo số chương trình được sử dụng cùng lúc
  - Hệ điều hành đơn nhiệm
  - Hệ điều hành đa nhiệm
- Theo người dùng (truy xuất tài nguyên cùng lúc)
  - Một người dùng
  - Nhiều người dùng
    - Mạng ngang hàng
    - Mạng có máy chủ: LAN, WAN, ...

# Các loại hệ điều hành

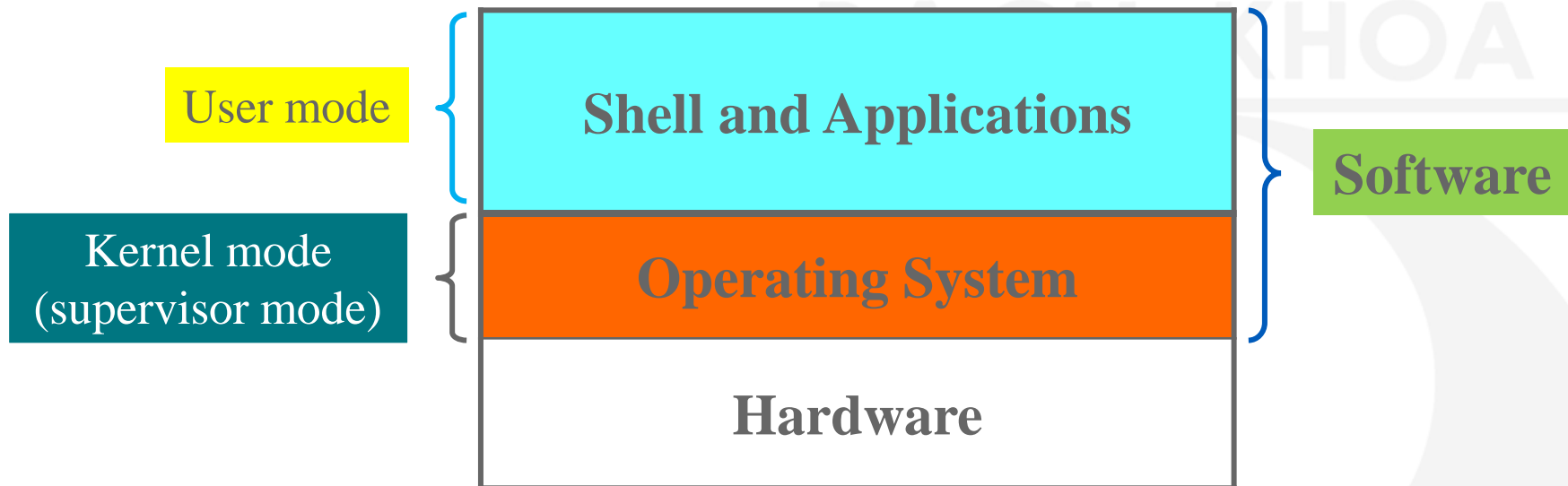
- Theo hình thức xử lý
  - Hệ thống xử lý theo lô
    - Hệ thống đơn chương trình (uniprogramming OS)
    - Hệ thống đa chương trình (multiprogramming OS)
  - Hệ thống chia sẻ thời gian
  - Hệ thống song song
  - Hệ thống phân tán
  - Hệ thống xử lý thời gian thực



# Câu hỏi?

- Tại sao phải cài đặt hệ điều hành trước khi muốn sử dụng và điều khiển máy tính/ thiết bị?
- Người dùng thao tác với phần cứng máy tính (RAM, ổ đĩa, card âm thanh, bàn phím, ...) như thế nào?
- Làm thế nào để máy tính có thể chạy nhiều chương trình cùng một lúc mà không lo hết RAM, CPU? Chúng chạy song song hay theo trình tự?
- Làm thế nào máy tính có thể phân bổ tài nguyên cho nhiều chương trình đang chạy cùng một lúc?
- Vai trò của hệ điều hành trong máy tính là gì? Chức năng chính của nó là gì? Máy tính gặp vấn đề gì khi không có hệ điều hành? Hệ điều hành là phần mềm (software) hay phần cứng (hardware)?

# Tổng quan



# Định nghĩa

- **Phần cứng:** Các thành phần vật lý như CPU, mainboard, ROM, RAM, ổ cứng, USB, ...
- **Phần mềm:** Chương trình truy cập và điều khiển phần cứng
  - **Phần mềm hệ thống** (System software) – phần mềm cơ bản cần phải cài đặt vào máy tính để máy tính có thể sử dụng được
  - **Phần mềm ứng dụng** (Application software) – phần mềm được phát triển dựa theo yêu cầu của người dùng và thực thi trên 1 phần mềm hệ thống cụ thể. Các chương trình hỗ trợ các tiện ích cho người dùng như MS Power Point, MS Word, games, ...
    - HĐH là phần mềm hệ thống

# Chức năng/ vai trò của HĐH

- HĐH được xem như một máy mở rộng giúp che giấu sự phức tạp khi truy cập/ sử dụng phần cứng
- HĐH là một trình quản lý tài nguyên
- HĐH bao gồm nhiều chức năng:
  - Quản lý tất cả tài nguyên phần cứng của hệ thống
  - Cung cấp cho người dùng môi trường để họ có thể:
    - Sử dụng tài nguyên hệ thống
    - Chạy các ứng dụng của riêng họ

# Mục tiêu của HĐH

- Thực thi các chương trình người dùng
- Làm cho phần cứng hiệu quả và dễ sử dụng
- Kiểm soát và điều phối việc sử dụng phần cứng giữa các chương trình ứng dụng khác nhau cho những người dùng khác nhau
- Ẩn các đặc thù (sự khác biệt) của việc truy cập phần cứng; chẳng hạn như đọc/ ghi ổ đĩa hoặc thiết bị I/O

# Thuật ngữ

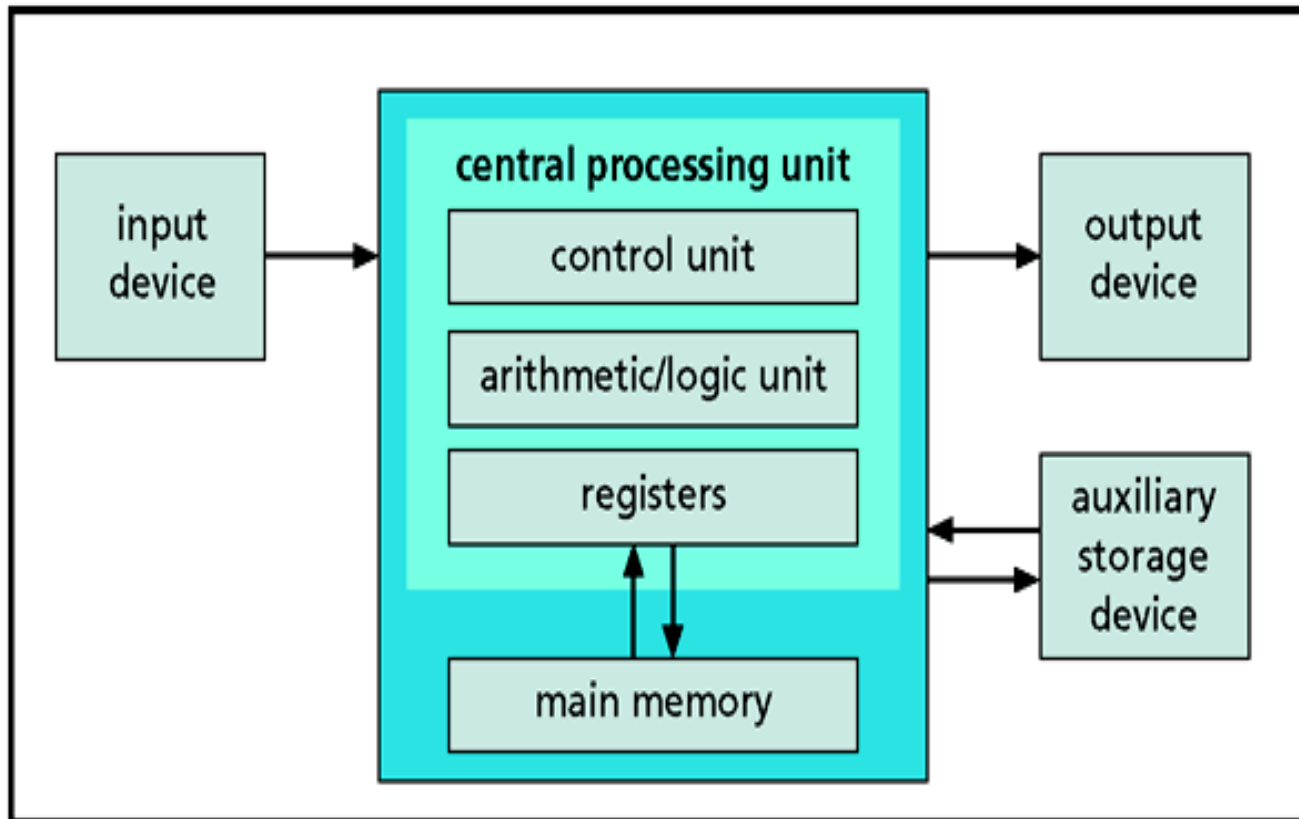
- **Instruction context:** một nhiệm vụ mà phần cứng phải thực hiện. Một lệnh hiện tại có thể thuộc về HĐH(kernel mode) hoặc ứng dụng đang chạy(user mode).
- **Kernel mode/Supervisor mode** (HĐH chạy ở chế độ này)
  - Giành quyền kiểm soát máy tính để truy cập tất cả phần cứng
  - Có thể thực hiện bất kỳ câu lệnh máy nào
  - Hỗ trợ bảo mật: Bảo vệ HĐH khỏi người dùng sai
  - Mọi thứ đang chạy trong chế độ này là một phần của HĐH hoặc được liên kết chặt chẽ với nó.
- **User mode**(Phần mềm người dùng chạy ở chế độ này)
  - Có thể thực thi một tập hợp con các câu lệnh máy ngoại trừ các lệnh để điều khiển máy hoặc thực hiện I/O
- **Mode switching** (chuyển đổi chế độ)
  - Nếu người dùng tương tác với HĐH: user mode → kernel mode
  - Nếu hệ thống chuyển quyền điều khiển cho chương trình người dùng: kernel mode → user mode

# Giới thiệu

- Hệ điều hành là gì?
- Lịch sử
- Các loại Hệ điều hành
- **Phần cứng**
- Thành phần Hệ điều hành
- Lời gọi hệ thống (System calls)
- Cấu trúc Hệ điều hành

# Phần cứng

Figure 3-19, Von Neumann architecture





# Kiến trúc Von Neumann

- Có các đặc điểm sau:
  - Chu trình lệnh: Fetch – Decode – Execute
  - Câu lệnh và dữ liệu được lưu trữ trong bộ nhớ chính
  - Việc thực thi câu lệnh được thực hiện bởi bộ xử lý trung tâm (CPU)

# Chu trình fetch-decode-execute

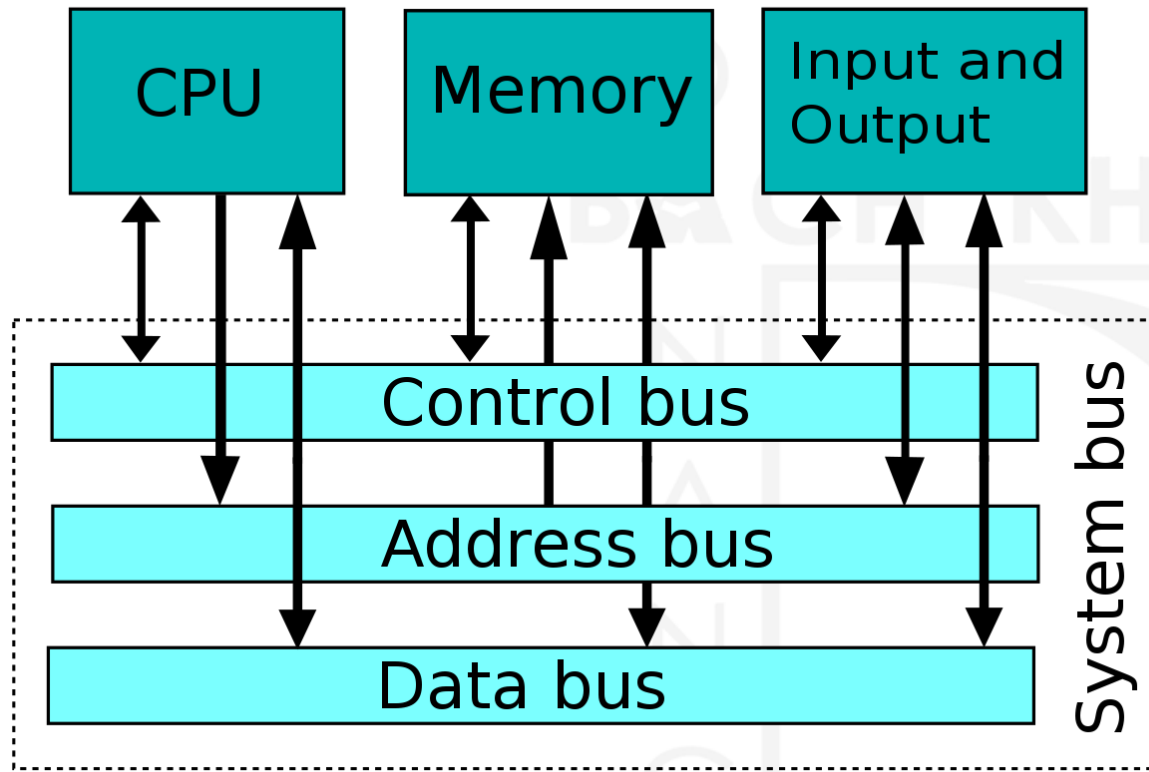
- Phân tích chu trình fetch – decode – execute điển hình:
  - Control unit (đơn vị điều khiển) sử dụng địa chỉ trong thanh ghi bộ đếm chương trình (program counter) để lấy lệnh từ bộ nhớ chính
  - Giải mã câu lệnh
  - Mọi dữ liệu cần thiết được lấy từ bộ nhớ và đặt vào các thanh ghi khác
  - ALU thực thi lệnh sử dụng dữ liệu trong các thanh ghi nếu cần
  - Các hoạt động đầu vào hoặc đầu ra theo yêu cầu của lệnh được thực hiện

# Bus

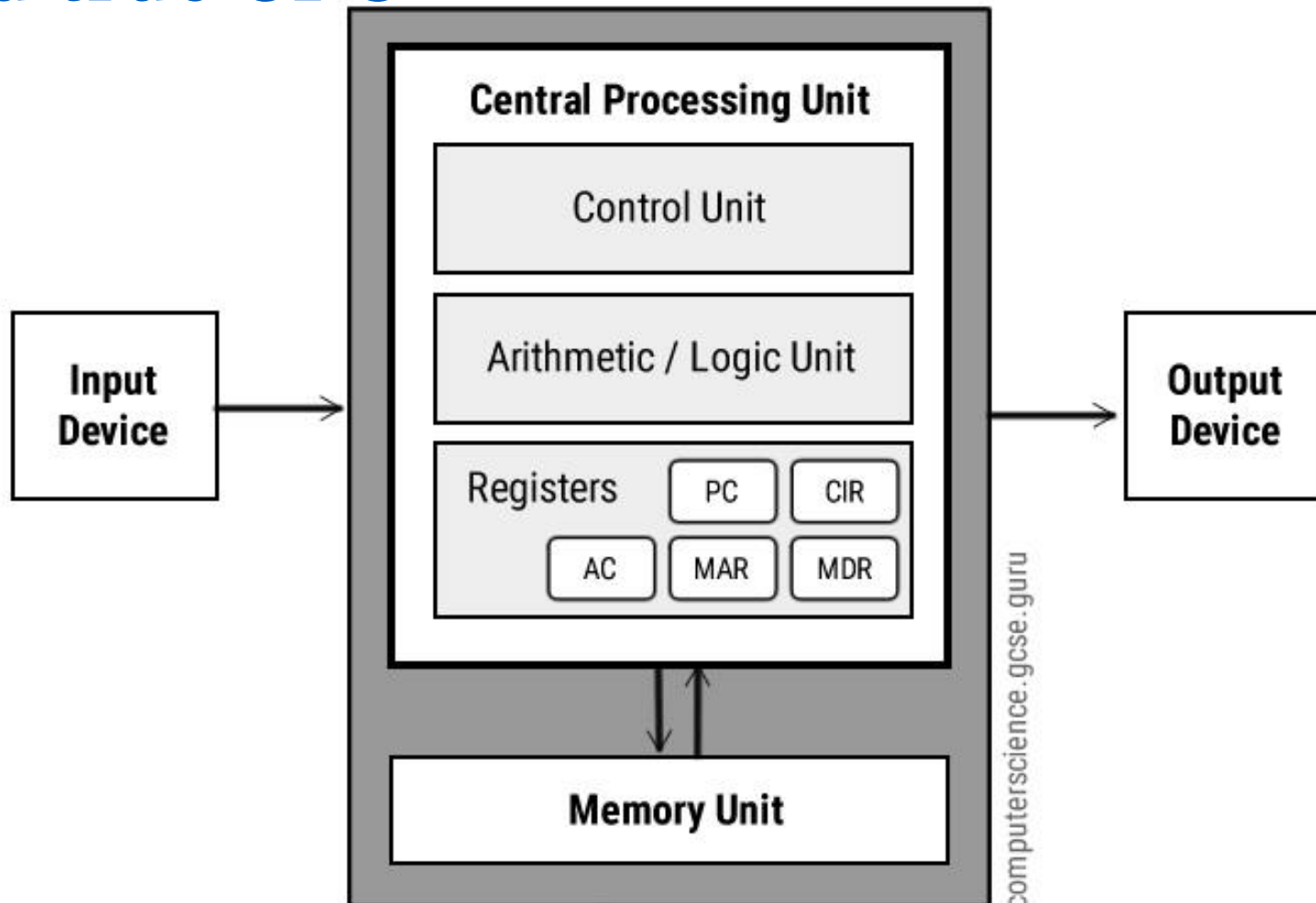
- Là phương tiện truyền dữ liệu từ một phần của máy tính sang phần khác, kết nối tất cả các thành phần chính bên trong với CPU và bộ nhớ
- 1 bus hệ thống CPU tiêu chuẩn bao gồm bus điều khiển, bus dữ liệu và bus địa chỉ

<b>Address Bus</b>	Mang theo địa chỉ của dữ liệu (không phải dữ liệu) giữa bộ xử lý và bộ nhớ
<b>Data Bus</b>	Mang dữ liệu giữa bộ xử lý, bộ nhớ và thiết bị vào/ra
<b>Control Bus</b>	Mang các tín hiệu/lệnh điều khiển từ CPU (và các tín hiệu trạng thái từ các thiết bị khác) nhằm điều khiển và phối hợp mọi hoạt động bên trong máy tính

# Bus



# Cấu trúc CPU



# Cấu trúc CPU

- Đơn vị điều khiển (CU)
  - Điều khiển hoạt động của ALU, bộ nhớ và thiết bị đầu vào/đầu ra của máy tính
  - Cung cấp các tín hiệu điều khiển và thời gian theo yêu cầu của các thành phần máy tính khác
- Đơn vị Số học/Logic (ALU)
  - cho phép thực hiện các phép toán số học (cộng, trừ, v.v.) và logic (VÀ, HOẶC, KHÔNG, v.v.)

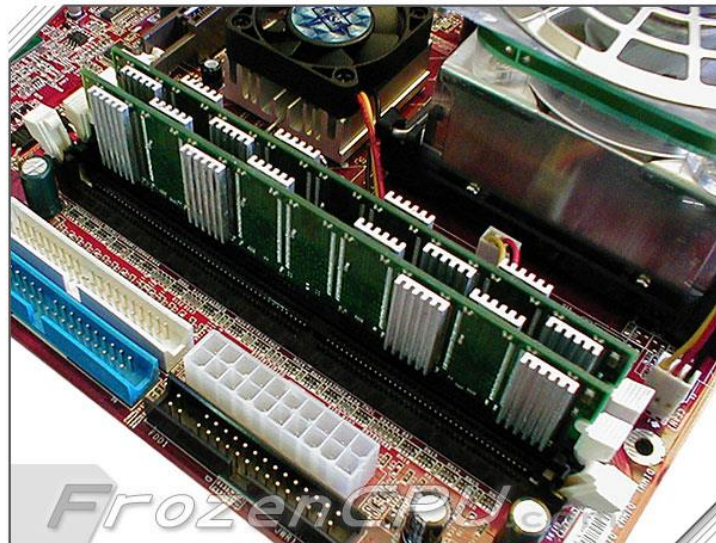
# Cấu trúc CPU

- Thanh ghi (Registers)
  - Thanh ghi là vùng lưu trữ tốc độ cao trong CPU.
  - Tất cả dữ liệu phải được lưu trữ trong một thanh ghi trước khi nó có thể được xử lý.

<b>MAR</b>	Memory Address Register	Giữ địa chỉ bộ nhớ của dữ liệu cần được truy cập
<b>MDR</b>	Memory Data Register	Giữ dữ liệu đang được chuyển đến hoặc từ bộ nhớ
<b>AC</b>	Accumulator	Nơi lưu trữ kết quả số học và logic trung gian
<b>PC</b>	Program Counter	Chứa địa chỉ của lệnh tiếp theo sẽ được thực hiện
<b>CIR</b>	Current Instruction Register	Chứa câu lệnh hiện tại trong quá trình xử lý

# Storage – lưu trữ

- Storage – các thành phần được sử dụng để lưu trữ chương trình và dữ liệu
- Phân loại
  - Bộ nhớ chính (primary memory)
  - Bộ nhớ thứ cấp (secondary memory) (bộ nhớ chung)

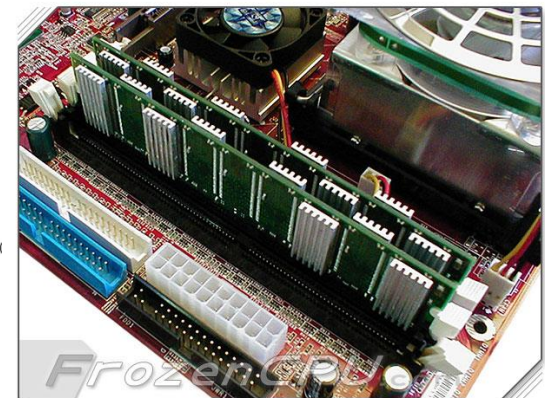
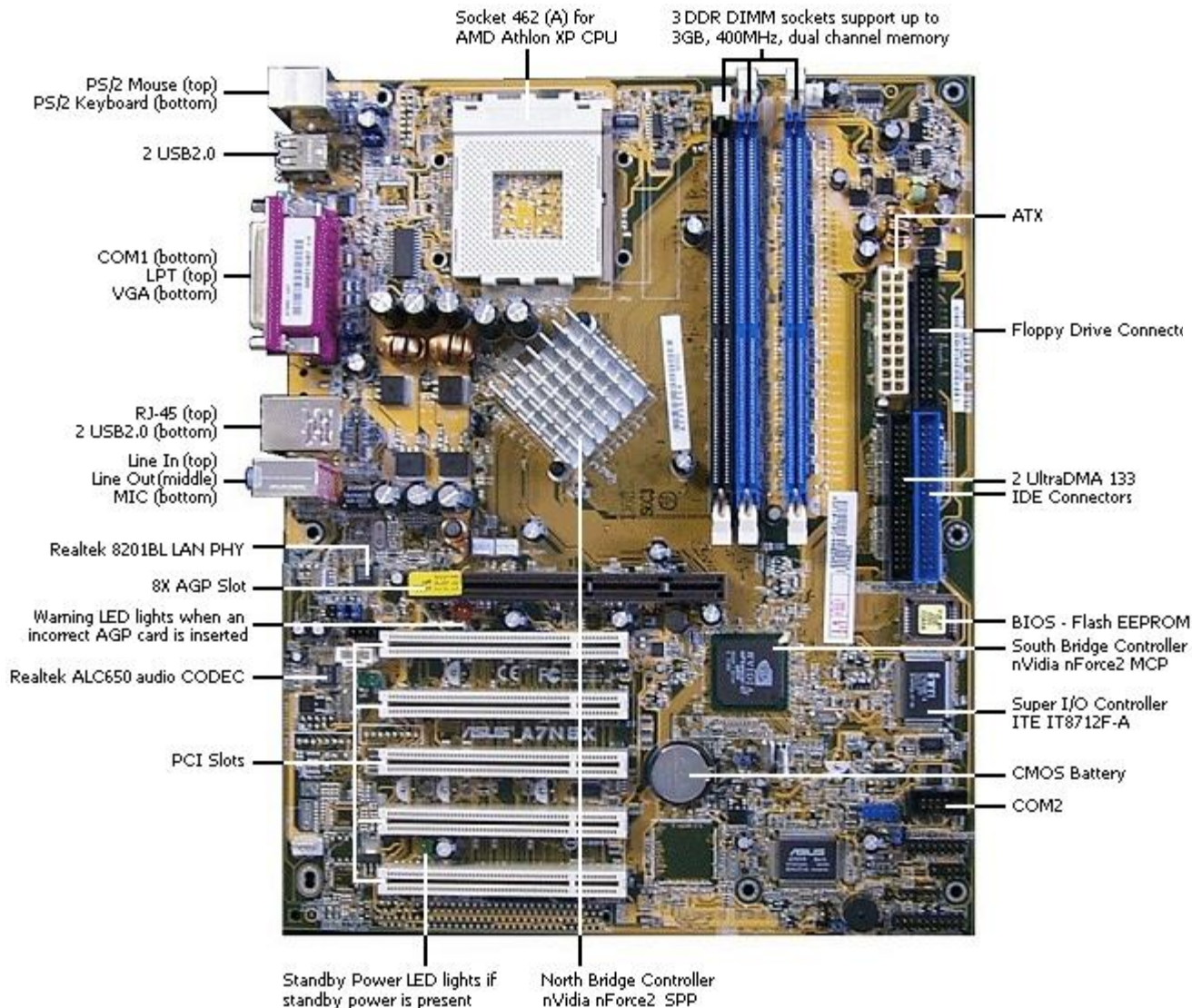




# Bộ nhớ

- 2 loại chính
  - ROM (read-only memory)
    - Bộ nhớ khắc vào chip
    - Nhìn chung không thể sửa đổi
    - Vd: BIOS (basic input/output system)
  - RAM (random access memory)
    - Cho phép tham chiếu bộ nhớ trực tiếp
    - Cho phép đọc/ ghi
    - Volatile
    - CPU lấy lệnh chương trình từ RAM

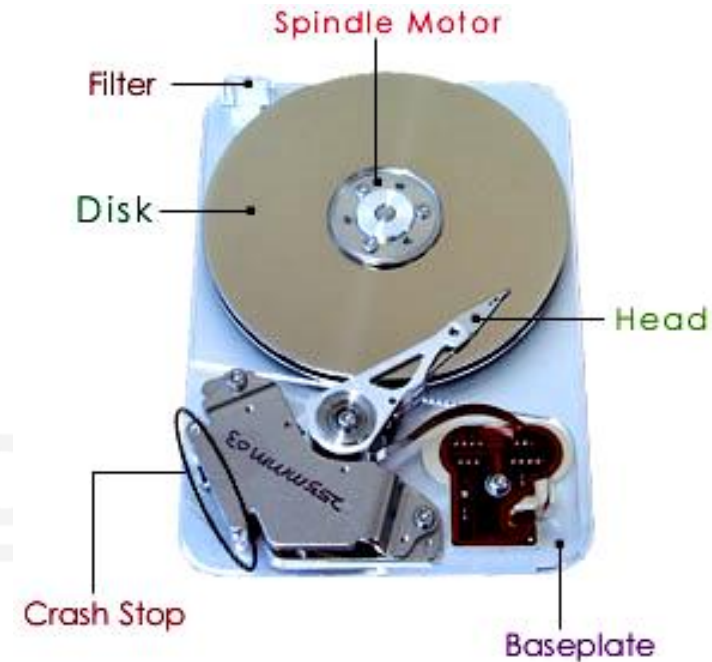
# RAM & ROM



BIOS - Flash EEPROM

# Ổ đĩa

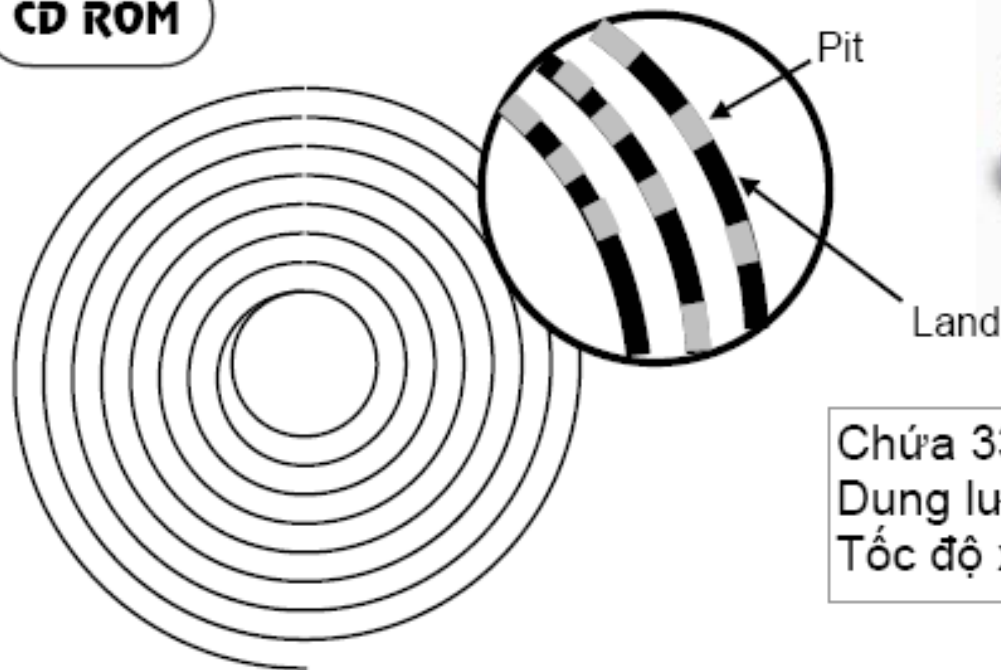
- Ổ cứng (Hard disk drives)
  - Phổ biến
  - Dữ liệu được lưu trữ ở đĩa kim loại từ tính
    - Được tổ chức thành các vòng tròn đồng tâm gọi là rãnh (track)
    - Các track được chia thành sector
    - Đĩa quay với tốc độ khoảng 7200 RPM
    - Sử dụng đầu đọc/ ghi lên mặt đĩa
  - Giá thành thấp so với RAM
  - RAID (Redundant Arrays of Inexpensive Disks)



# Ổ đĩa

- Optical Storage (Quang học)
  - Các định dạng phổ biến: CD và DVD
  - Lưu trữ dữ liệu bằng công nghệ quang học (laser)
  - Các lỗ được ghi vào đĩa được hiểu là dữ liệu nhị phân
  - Dữ liệu được ghi vào đĩa theo hình xoắn ốc liên tục
  - Giống như đĩa cứng, đĩa CD và DVD quay
  - Giao diện đầu đọc/ghi với bề mặt đĩa
  - Chi phí thấp

**CD ROM**



Chứa 330.000 khối dữ liệu.  
Dung lượng 650 MB / 74 min  
Tốc độ x1 = 153.60 KByte/s

Thông tin ghi theo rãnh (track) hình xoắn ốc  
Dùng tia laser đục lỗ 1  $\mu\text{m}$  trên rãnh gọi là Pit.  
Phần không bị đục lỗ trên rãnh gọi là Land.



# Ổ đĩa

- Solid-State Drive
  - Sử dụng bộ nhớ Solid State để lưu trữ thông tin
  - Sử dụng Bộ nhớ flash NAND không bay hơi, cho phép nó giữ lại dữ liệu khi ngắt nguồn điện
  - Những lợi ích:
    - Khởi động SSD nhanh hơn
    - Truy cập nhanh hơn
    - Thời gian khởi chạy ứng dụng nhanh hơn
    - Năng lượng hiệu quả hơn
    - Đáng tin cậy hơn
    - ...



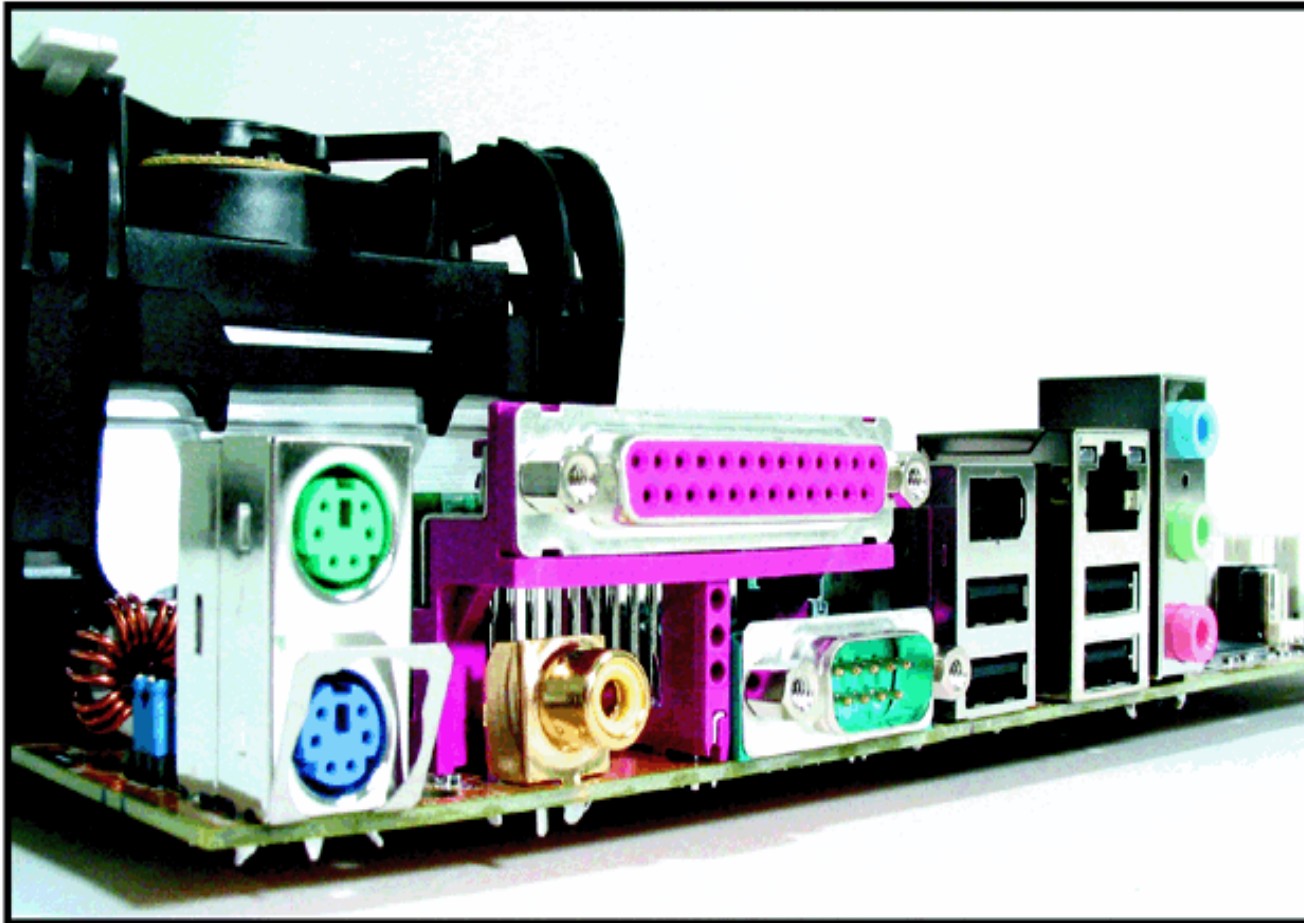
# Thiết bị vào (input devices)

- Bàn phím
  - Thiết bị đầu vào chính cho hầu hết người dùng
  - Kết nối với bo mạch chủ thông qua cổng và sau đó đến CPU bằng mạch điều khiển và bus hệ thống
  - Tổ hợp phím được dịch thành tín hiệu nhị phân để tiêu thụ CPU
- Chuột
  - Dùng kết hợp với bàn phím
  - Cảm nhận chuyển động có thể được dịch thành mã nhị phân
- Các thiết bị khác: bi xoay, bút stylus (dụng cụ hình cây bút nhỏ), miếng cảm ứng/màn hình



# Thiết bị vào (input devices)

Figure 3-21, The motherboard provides numerous ports to connect peripheral devices





# Thiết bị ra (Output devices)

- Kênh giao tiếp với thế giới bên ngoài
- Màn hình
  - Thiết bị đầu ra chính
  - CRT (ống tia âm cực)
    - Sử dụng các kỹ thuật quét raster
    - Chất lượng dựa trên độ phân giải và tốc độ làm mới
  - LCD (Màn hình tinh thể lỏng)
    - Mỏng hơn và mát hơn CRT
    - Sử dụng bóng bán dẫn thay vì chùm điện tử
    - Chất lượng dựa trên độ phân giải và tốc độ làm mới



# Thiết bị khác

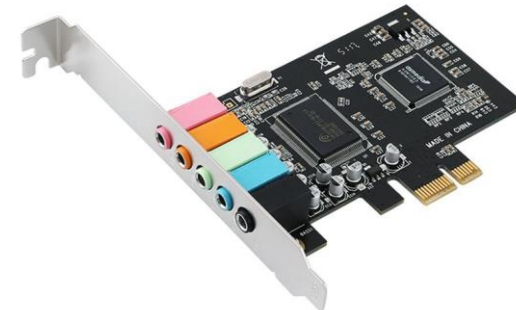
- Máy in

- Thiết bị đầu ra quan trọng
- Các loại chính: máy in phun mực và máy in laser
- Chất lượng được đo bằng độ phân giải (số chấm trên inch) và tốc độ (trang trên phút)



- Card âm thanh

- Vừa với khe cắm mở rộng PCI trên bo mạch chính
- Dùng để số hóa âm thanh để lưu trữ
- Cũng chuyển đổi các tệp âm thanh nhị phân thành âm thanh tương tự



# Interrupts and Polling

- Làm thế nào CPU biết tín hiệu từ I/O?
- Chu kỳ lệnh của CPU bằng tốc độ xung nhịp
- CPU cam kết thực hiện chu kỳ dựa trên nhu cầu xử lý
- Nhu cầu xử lý được xác định bởi (2) kỹ thuật
  - Polling (quét dò tìm định kỳ): CPU liên tục quét dò định kỳ thiết bị I/O
  - Xử lý ngắt (Báo tín hiệu ngắt): Thiết bị I/O khởi tạo yêu cầu dịch vụ

# Cấu trúc dữ liệu

- Linked list (next)
- Stack (First in, last out)
- Queue (First in, first out)

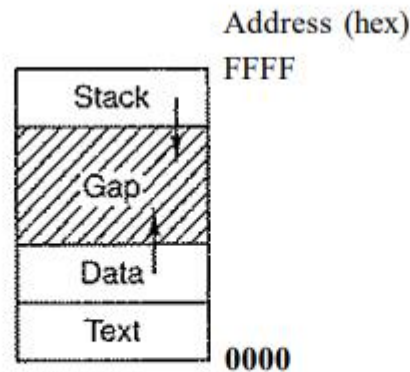


Figure 1-20- Processes have three segments: text, data, and stack.

# Giới thiệu

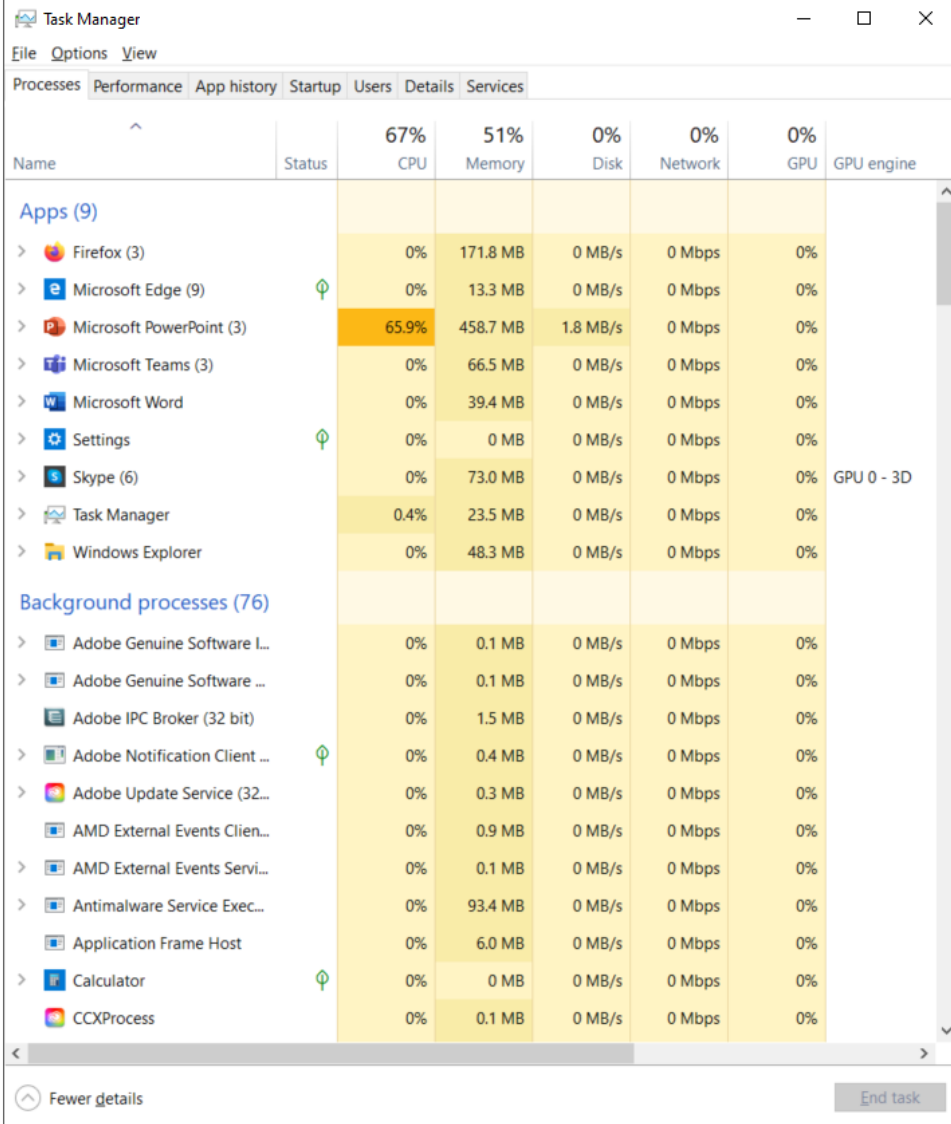
- Hệ điều hành là gì?
- Lịch sử
- Các loại Hệ điều hành
- Phần cứng
- **Thành phần Hệ điều hành**
- Lời gọi hệ thống (System calls)
- Cấu trúc Hệ điều hành

# Thành phần của HĐH

- Tiến trình (process)
- Không gian địa chỉ (Address space)
- Tập tin (File)
- Input/ Output
- Protection
- Shell

# Tiến trình

- Tiến trình là gì? Khi nào tiến trình xuất hiện?
- Khi nào cũng thấy nhiều tiến trình, chúng có chạy song song không? HĐH quản lý các tiến trình như thế nào?



**Task Manager**  
File Options View

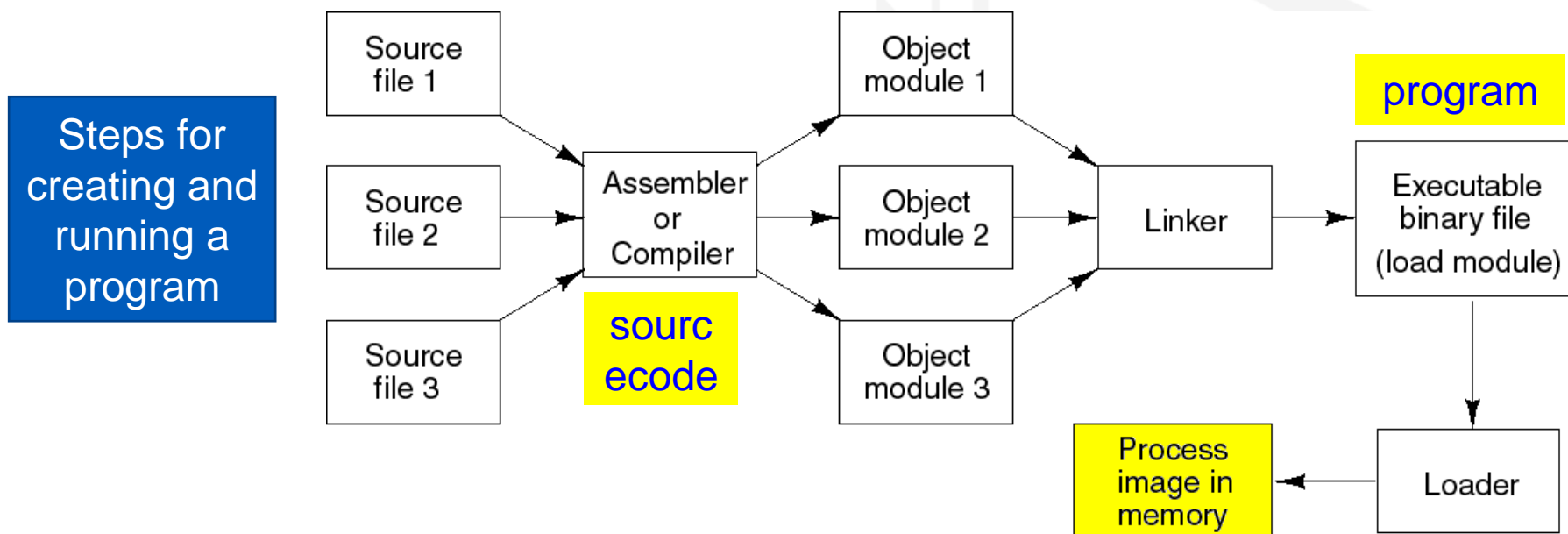
Processes Performance App history Startup Users Details Services

Name	Status	67% CPU	51% Memory	0% Disk	0% Network	0% GPU	GPU engine
<b>Apps (9)</b>							
Firefox (3)		0%	171.8 MB	0 MB/s	0 Mbps	0%	
Microsoft Edge (9)	🟢	0%	13.3 MB	0 MB/s	0 Mbps	0%	
Microsoft PowerPoint (3)		65.9%	458.7 MB	1.8 MB/s	0 Mbps	0%	
Microsoft Teams (3)		0%	66.5 MB	0 MB/s	0 Mbps	0%	
Microsoft Word		0%	39.4 MB	0 MB/s	0 Mbps	0%	
Settings	🟢	0%	0 MB	0 MB/s	0 Mbps	0%	
Skype (6)		0%	73.0 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
Task Manager		0.4%	23.5 MB	0 MB/s	0 Mbps	0%	
Windows Explorer		0%	48.3 MB	0 MB/s	0 Mbps	0%	
<b>Background processes (76)</b>							
Adobe Genuine Software L...		0%	0.1 MB	0 MB/s	0 Mbps	0%	
Adobe Genuine Software ...		0%	0.1 MB	0 MB/s	0 Mbps	0%	
Adobe IPC Broker (32 bit)		0%	1.5 MB	0 MB/s	0 Mbps	0%	
Adobe Notification Client ...	🟢	0%	0.4 MB	0 MB/s	0 Mbps	0%	
Adobe Update Service (32...		0%	0.3 MB	0 MB/s	0 Mbps	0%	
AMD External Events Clie...		0%	0.9 MB	0 MB/s	0 Mbps	0%	
AMD External Events Servi...		0%	0.1 MB	0 MB/s	0 Mbps	0%	
Antimalware Service Exec...		0%	93.4 MB	0 MB/s	0 Mbps	0%	
Application Frame Host		0%	6.0 MB	0 MB/s	0 Mbps	0%	
Calculator	🟢	0%	0 MB	0 MB/s	0 Mbps	0%	
CCXProcess		0%	0.1 MB	0 MB/s	0 Mbps	0%	

⬅ Fewer details End task

# Tiến trình

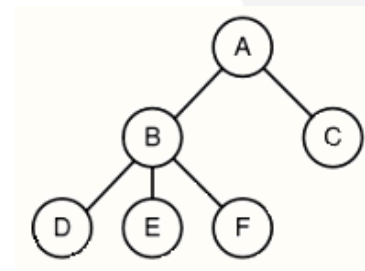
- Tiến trình: 1 chương trình đang được thực thi → Chương trình: 1 file có thể được thực thi nằm trên ổ đĩa
- Với mỗi tiến trình, HĐH cấp phát 1 tập tài nguyên như các khối bộ nhớ cho executable code, data, stack, CPU registers value, PC và các thông tin khác cần thiết để chạy 1 chương trình → mỗi tiến trình có 1 không gian địa chỉ duy nhất (tất cả vị trí bộ nhớ mà tiến trình có thể đọc/ ghi)





# Tiến trình

- HĐH quản lý nhiều tiến trình → Thông tin về tiến trình phải được giữ trong bộ nhớ của HĐH
- Bảng tiến trình (**process table**) được lưu trữ bằng mảng hoặc danh sách liên kết lưu trữ tất cả thông tin của tiến trình
  - 1 hệ thống gồm tập hợp các tiến trình (nhiều chương trình)
  - Dựa vào bảng này, HĐH chuyển đổi CPU giữa các tiến trình
- Phân cấp tiến trình (**process hierarchy**)
  - 1 tiến trình (cha) có thể tạo các tiến trình con
  - 1 tiến trình chỉ có 1 cha nhưng có 1 hoặc nhiều con
  - 1 HĐH cụ thể sẽ triển khai khái niệm phân cấp theo các cách khác nhau



# Cơ chế chuyển đổi

- **Vấn đề:** Dung lượng RAM có giới hạn. Làm thế nào HĐH cho phép nhiều chương trình nạp vào bộ nhớ? → Cơ chế chuyển đổi (**Swap mechanism**)
  - Tại một thời điểm, chỉ có 1 tiến trình giữ quyền điều khiển (tiến trình hiện tại) → Bộ nhớ (memory map) của một tiến trình *không* hiện tại có thể được tự động chuyển đổi khỏi bộ nhớ và sau đó chuyển đổi trở lại bộ nhớ tại một nơi nào đó (HĐH sẽ cập nhật địa chỉ nếu cần).
  - Nhà phát triển cố ý xác định địa chỉ tuyệt đối của chương trình nằm trong bộ nhớ tại thời điểm đó (mô hình lập trình cũ)

# Không gian địa chỉ

- Làm thế nào HĐH có thể xác định địa chỉ truy cập? → **Không gian địa chỉ**
  - Không gian địa chỉ: Tập giá trị địa chỉ có thể truy cập bởi tiến trình
  - Mỗi tiến trình có không gian địa chỉ riêng
  - Tiến trình này không thể truy cập vào địa chỉ của tiến trình khác (cơ chế bảo vệ - protection mechanism)
  - Khiến mỗi chương trình nghĩ rằng chúng có phần cứng riêng (cơ chế bảo vệ)
  - Không gian địa chỉ có thể được tách riêng thành pages/ segments. Mỗi byte được đánh số từ 0 đến maximum. Kích thước page/ segment có thể khác nhau (lớn hơn hoặc nhỏ hơn)

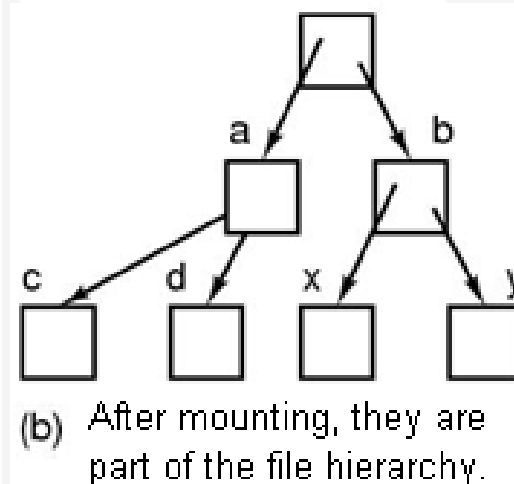
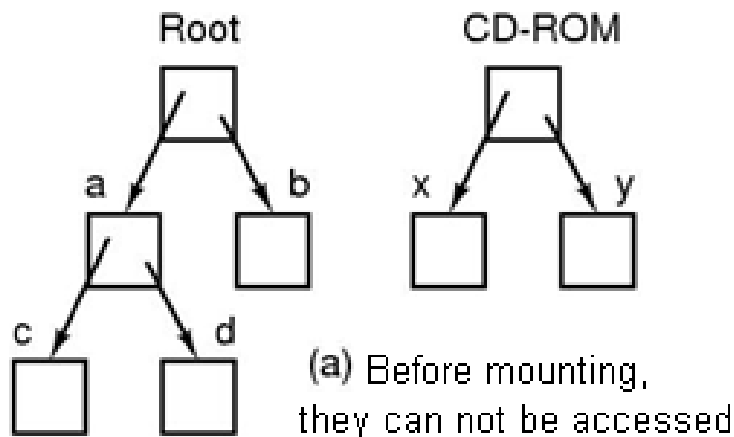
# File

- File
- Đường ống (pipe)
- File đặc biệt
- Thư mục (Directory)

D  
BACH KHOA  
N  
A  
N  
G

# File

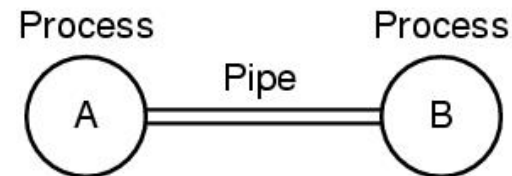
- **File:** 1 tập hợp những thông tin liên quan đến nhau
  - Nó là một bản tóm tắt của dữ liệu được lưu trữ trên ổ cứng
  - Thông tin tương quan
    - Tên đường dẫn (path name), thư mục gốc (root directory), bộ mô tả file (file descriptor) (Windows)
    - Hệ thống file được gắn kết (mount): một số cấu trúc file được liên kết với nhau và được quản lý bởi HĐH.



# File

- **Đường ống (pipe) (UNIX)**

- Cơ chế truyền dữ liệu giữa các tiến trình
- Là một file giả (pseudofile) có thể được sử dụng để kết nối 2 tiến trình
- Cho phép tiến trình con kế thừa kênh liên lạc với tiến trình cha, dữ liệu được ghi vào đầu đường ống có thể được đọc ở đầu kia



# File

- **Special file (UNIX)** - Làm cho các thiết bị I/O trông giống như file được sử dụng để đọc/ghi (mỗi thiết bị I/O được xác định bởi 1 file). Các file này liên kết với trình điều khiển thiết bị (**driver**) thích hợp. Chúng được phân thành 2 loại:
  - Block special file (ví dụ: ổ đĩa): Là một file bao gồm chuỗi các khối được đánh số, mỗi khối có thể được đánh địa chỉ (ngẫu nhiên) riêng lẻ và được truy cập (đọc/ghi bằng đơn vị khối)
  - Character special file (VD: máy in, modem, chuột,...)
    - Được sử dụng cho các thiết bị nhập hoặc xuất luồng ký tự (đọc/ghi sử dụng đơn vị ký tự)
    - Không thể truy cập ngẫu nhiên (không khối và không địa chỉ)
- **Thư mục:** nơi lưu giữ các file/ một cách nhóm các file lại với nhau.

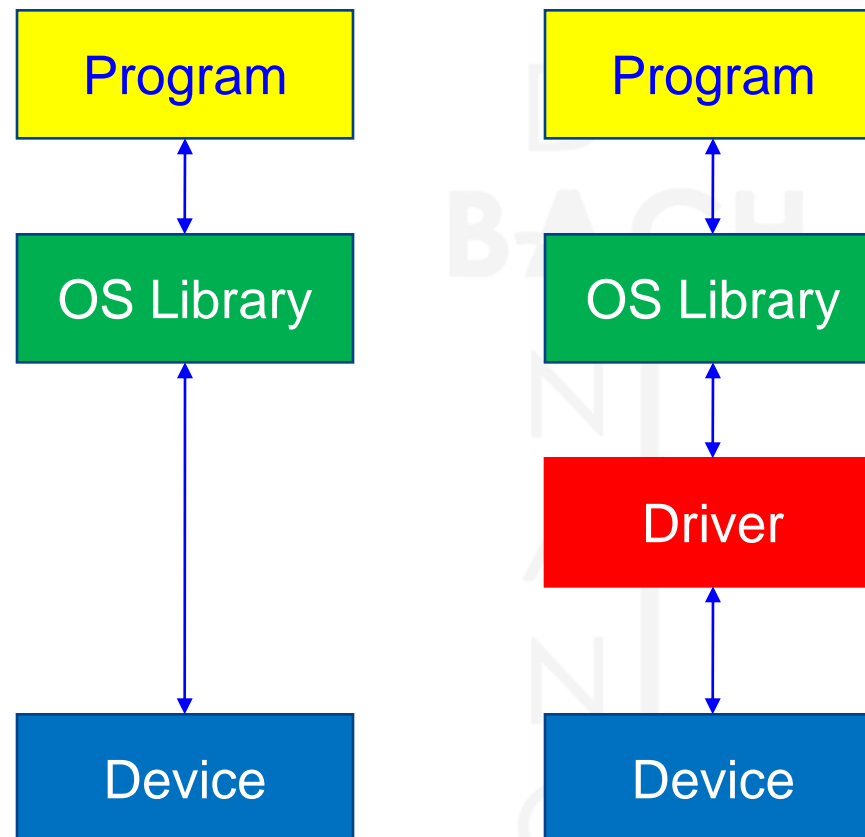
# Input/ Output

- Tất cả các máy tính đều có thiết bị vật lý để nhận đầu vào và tạo đầu ra.
  - Một số trong số chúng có thể được chia sẻ thiết bị
  - Số khác có thể là thiết bị chuyên dụng
- Mọi hệ điều hành đều có một hệ thống con I/O để quản lý các thiết bị I/O
  - Một số phần mềm I/O độc lập với thiết bị (thiết bị tiêu chuẩn).
  - Số khác dành riêng cho các thiết bị I/O cụ thể sử dụng trình điều khiển thiết bị (thiết bị không chuẩn)

→ HĐH sử dụng phần mềm I/O dựa trên phần cứng I/O.



# Input/ Output



# Cơ chế bảo vệ

- Là một cơ chế để kiểm soát quyền truy cập của các tiến trình hoặc người dùng trên các tài nguyên do HĐH xác định; đảm bảo rằng tất cả các quyền truy cập vào tài nguyên hệ thống đều được kiểm soát.
- Bảo vệ code của HĐH trước các tiến trình → Code của shell không thể truy cập không gian địa chỉ của HĐH.
- Bảo vệ một tiến trình trước các tiến trình khác → Mã của một tiến trình không thể truy cập không gian địa chỉ của các tiến trình khác.
- Có thể cải thiện độ tin cậy bằng cách phát hiện các lỗi tiềm ẩn tại giao diện giữa các hệ thống con thành phần
- Kỹ thuật:
  - Xác thực, bảo vệ hệ thống khỏi những kẻ xâm nhập không mong muốn (quan điểm quản lý người dùng)
  - Cung cấp các thuộc tính của file để bảo vệ việc truy cập file → Mỗi người dùng chỉ có thể truy cập một tập hợp con của hệ thống tệp.

# Shell

- Nó là một phần mềm (không phải là một phần của HĐH) về cơ bản cung cấp một loại giao diện cho người dùng cuối truy cập tài nguyên máy tính.
- 2 loại shell:
  - Thông dịch lệnh (command interpreter). Giao diện dòng lệnh: ví dụ: Dấu nhắc lệnh trong Windows.
  - Giao diện người dùng đồ họa (GUI): ví dụ: Window Explorer, Gnome trong Linux, KDE- K Môi trường Desktop
- Mặc dù nó không phải là một phần của HĐH, nhưng nó sử dụng rất nhiều tính năng của HĐH
- Nhiều shell tồn tại như sh – Shell, csh – C Shell, ksh – Korn Shell, và bash – Bourne Again Shell (shell mặc định trong hầu hết hệ thống Linux).

# Shell

- Chức năng

- Được khởi động (khi người dùng đăng nhập) với dấu nhắc (\$) ký tự
- Có thiết bị đầu cuối là đầu vào và đầu ra tiêu chuẩn

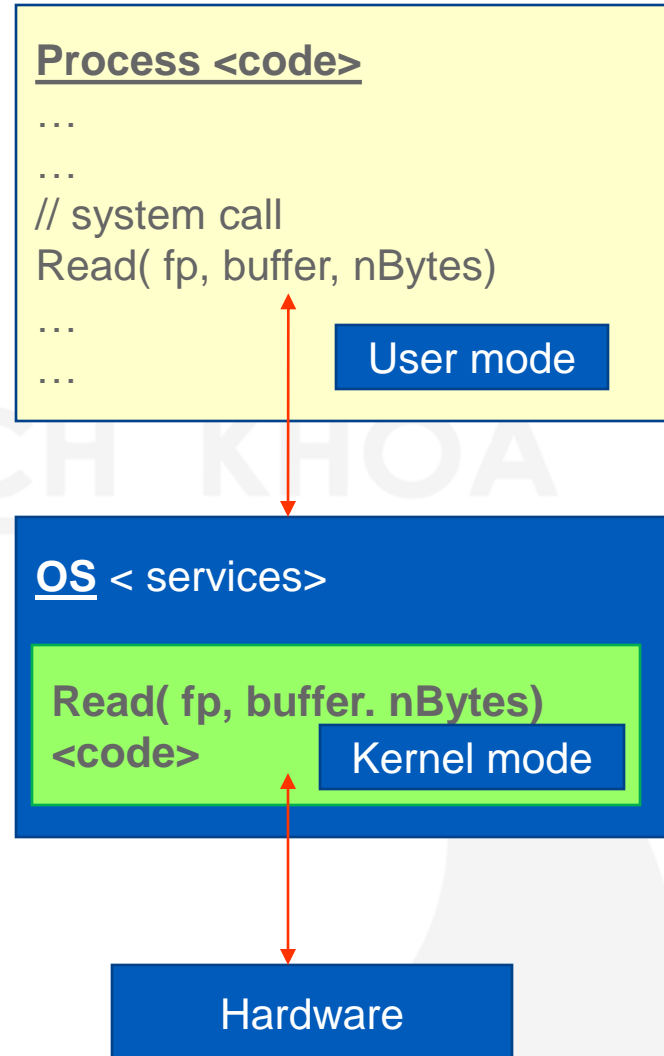
- **Shell scripts:** Một file chứa danh sách các lệnh shell mà chúng được thực thi theo thứ tự (giống như tệp bat trong DOS)

# Giới thiệu

- Hệ điều hành là gì?
- Lịch sử
- Các loại Hệ điều hành
- Phần cứng
- Thành phần Hệ điều hành
- **Lời gọi hệ thống (System calls)**
- Cấu trúc Hệ điều hành

# System calls

- HĐH hỗ trợ giao diện lập trình ứng dụng (API – thư viện HĐH) để truy cập tài nguyên
- Các tiến trình sử dụng API bằng cách sử dụng lời gọi hệ thống.
- Giao diện giữa các chương trình người dùng và HĐH (ẩn với người dùng)
- Cho phép các chương trình ứng dụng truy cập các tài nguyên được bảo vệ
- Thực hiện lời gọi hệ thống giống như thực hiện một cuộc gọi thủ tục đặc biệt nằm trong kernel.



# System calls

- Sự kiện trong 1 lời gọi hệ thống
  - 1 tiến trình đang chạy trong user mode và cần 1 dịch vụ hệ thống, tiến trình thực thi 1 trap instruction (interrupt) để chuyển quyền điều khiển sang cho HĐH (kernel mode)
  - HĐH tìm ra tiến trình muốn gì bằng cách kiểm tra tham số, sau đó nó thực hiện lời gọi hệ thống và trả lại quyền điều khiển cho lệnh sau lời gọi hệ thống.

# System calls

- Các bước thực hiện lời gọi hệ thống:
  1. Chương trình đưa đối số vào thanh ghi hoặc ngăn xếp (chuẩn bị truyền đối số cho hàm HĐH)
  2. Đưa ra các lệnh bẫy (trap instruction) để chuyển từ user mode sang kernel mode tại địa chỉ cố định nơi đặt thủ tục (HĐH lưu vị trí trả về của lệnh tiếp theo của tiến trình).
  3. Kernel mode gửi đến trình xử lý lời gọi hệ thống chính xác (hàm HĐH thực thi).
  4. Khi trình xử lý lời gọi hệ thống đã hoàn thành công việc của nó, các lệnh bẫy được đưa ra để chuyển kernel sang user mode.
  5. Sau đó, thủ tục trả về chương trình người dùng theo cách thông thường gọi thủ tục trả về.

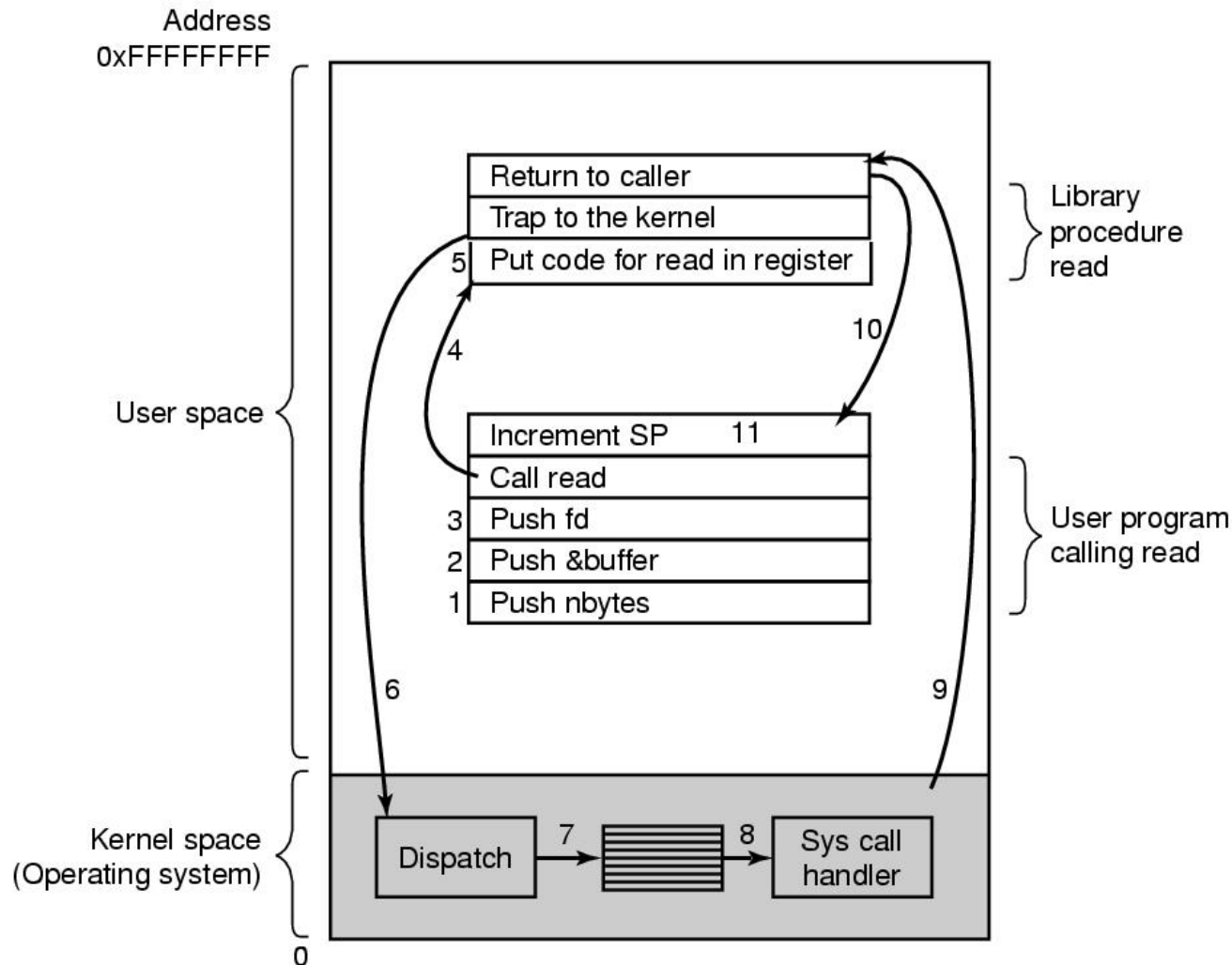


# System calls

- System programs

- Chương trình hệ thống cung cấp môi trường thuận tiện cho việc phát triển và thực thi chương trình (VD thao tác với file, trạng thái, ...)
- Chương trình hệ thống → shell
- Là giao diện người dùng đơn giản cho system calls
  - VD: Norton Commander, Window Explorer, ...

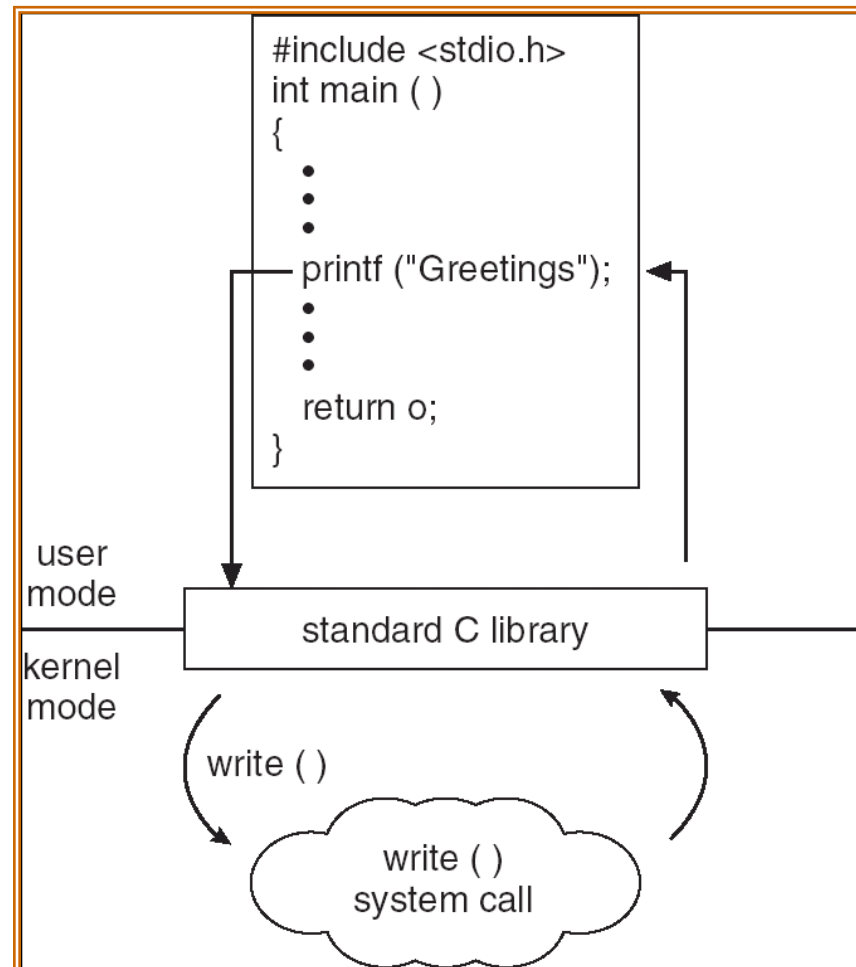
# System calls: Example



C, C++ compilers will put parameters to the stack in reverse order.

There are 11 steps executing the system call: read (fd, buffer, nbytes)

# System calls: Example



# System calls: Example

- Quản lý tiến trình:
  - Create a child process - `fork()`,
  - Wait for a child process to terminate – `waitpid(...)`
  - Execute a process - `exec()`,
  - Terminate process and return status - `exit(status)` ...
- Quản lý file:
  - `open()`, `close()`, `read()`, `write()`, `lseek()`,
  - Get file status: `stat()`
  - ...

# System calls: Example

- Quản lý file hệ thống:
  - mkdir(...), rmdir(...),
  - mount(...), unmount(...),
  - link(), unlink(), chown() - change file owner
  - ...
- Khác:
  - Change the working directory: chdir(),
  - Change the file's protection bits: chmod(),
  - Send a signal to a process: kill()
  - Get the elapsed time since Jan, 1, 1970: time()
  - ....

# System calls: Example

- UNIX System Calls vs Windows Win32 API

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

# Giới thiệu

- Hệ điều hành là gì?
- Lịch sử
- Các loại Hệ điều hành
- Phần cứng
- Thành phần Hệ điều hành
- Lời gọi hệ thống (System calls)
- **Cấu trúc Hệ điều hành**

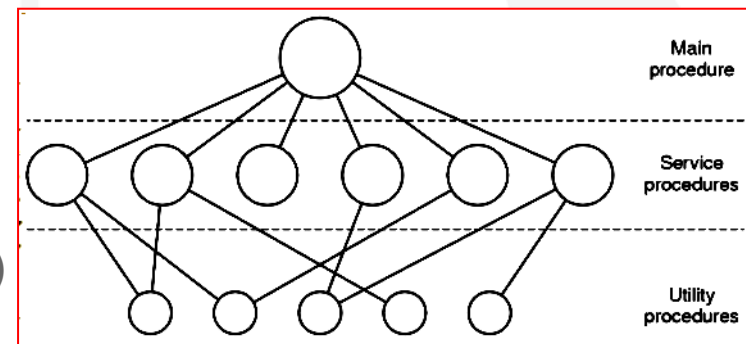
# Cấu trúc HĐH

- Monolithic Systems – Hệ nguyên khối
- Layered Systems – Hệ phân lớp
- Micro Kernels – Hệ vi nhân
- Client-Server Model
- Virtual Machines: Máy ảo



# Hệ nguyên khối

- Toàn bộ HDH chạy như 1 chương trình duy nhất ở kernel mode
- HDH được viết dưới dạng tập hợp các thủ tục, được liên kết với nhau thành 1 chương trình lớn để có thể thực thi được
- **Cấu trúc cơ bản:**
  - 1 chương trình chính gọi thủ tục dịch vụ được yêu cầu
  - 1 tập hợp các thủ tục dịch vụ thực hiện các system calls
  - Đối với mỗi system call, sẽ có 1 thủ tục dịch vụ xử lý và thực thi nó
- **Tập hợp các thủ tục tiện ích**
  - Hỗ trợ các thủ tục dịch vụ
  - Thực hiện những việc cần thiết trong một số quy trình dịch vụ (vd tìm nạp dữ liệu)



# Hệ nguyên khối

- Ưu điểm
  - Đơn giản → chạy nhanh
- Nhược điểm
  - Không linh hoạt (Chỉ chạy một chương trình nhị phân thực thi duy nhất)
  - Không bảo vệ mà giấu thông tin
  - Hệ thống có thể gọi bất kì system call
  - Có thể sử dụng ngôn ngữ bậc thấp (assembly)
  - Có hàng ngàn thủ tục có thể gọi nhau
  - Sử dụng thủ tục (tĩnh) → không thể quản lý môi trường người dùng
- VD: MS-DOS

# Hệ phân lớp

- HDH được chia thành nhiều lớp (tầng)
- Lớp thấp nhất là hardware, lớp cao nhất là giao diện người dùng, các lớp được chọn sao cho mỗi lớp chỉ sử dụng các chức năng, hoạt động và dịch vụ của các lớp thấp hơn
- Các lớp thấp hơn (thường được gọi là kernel) chứa các chức năng cơ bản nhất để quản lý tài nguyên hệ thống
- Có 6 lớp:

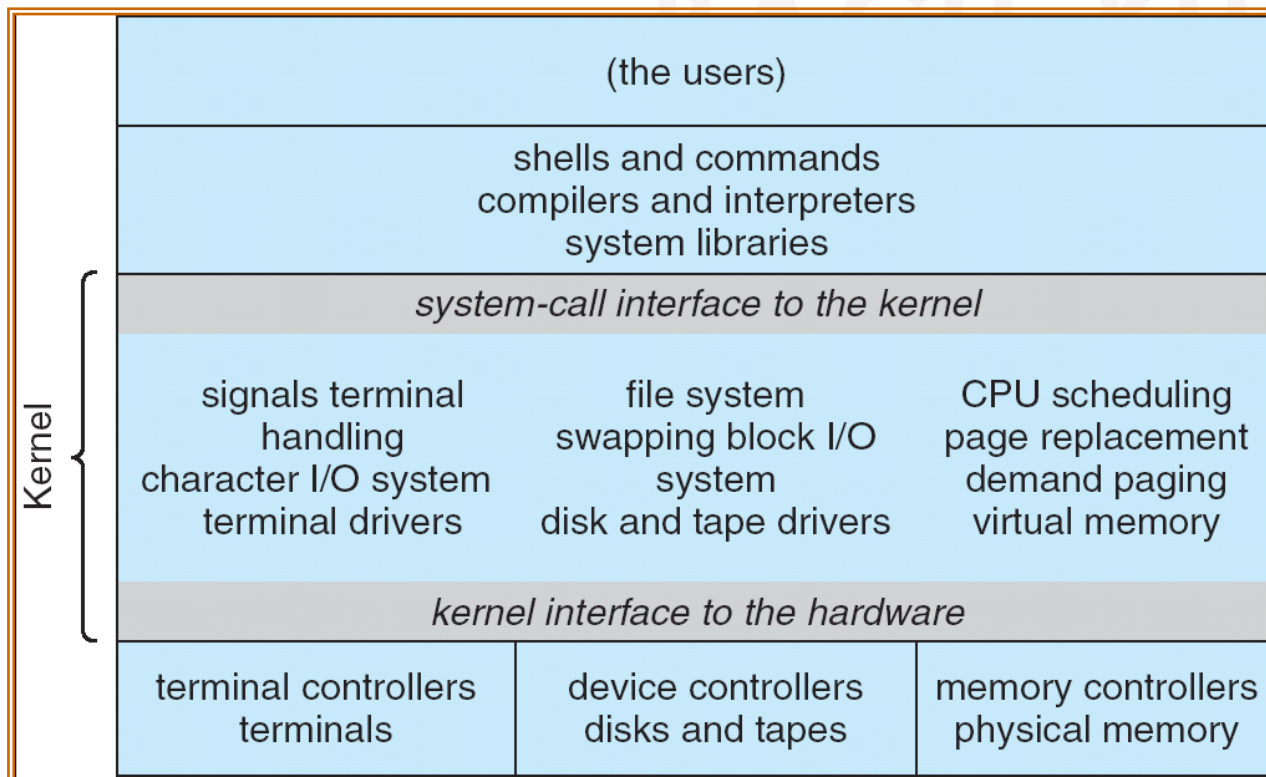
Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

# Hệ phân lớp

- Ưu điểm
  - Dễ bảo trì, hỗ trợ, quản lý lỗi
  - Mô đun hóa, đóng gói
  - VD: OS/2, Windows NT, Vista và Unix
- Nhược điểm
  - Bao nhiêu lớp?
  - Mỗi lớp có chức năng gì?
  - Chậm

# Hệ vi nhân

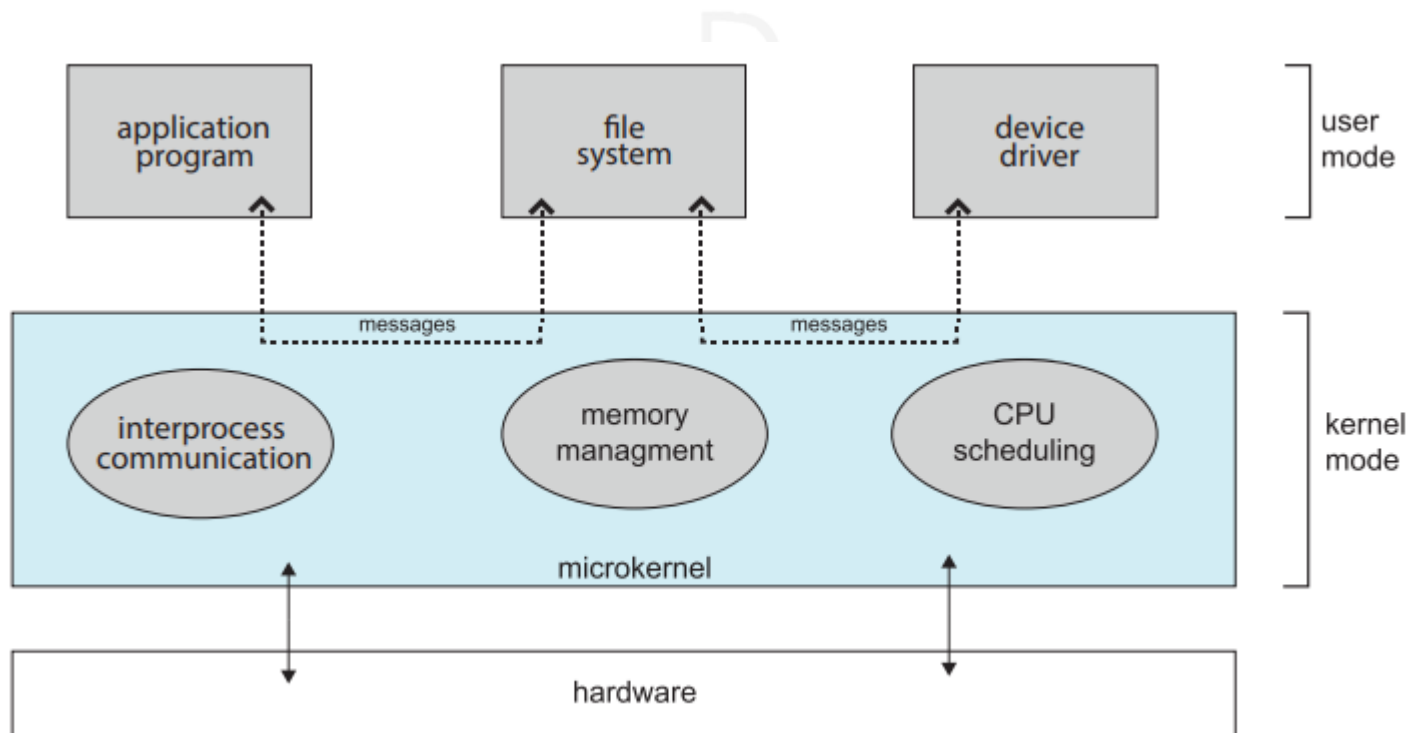
- Kernel HDH gặp các vấn đề
  - 10 bugs/ 1000 dòng code
  - Tất cả các lớp đều nằm trong kernel (phức tạp, lớn, khó quản lý)



# Hệ vi nhân

- Loại bỏ tất cả các thành phần không cần thiết khỏi kernel và triển khai chúng ở user mode (chia HDH thành các mô đun nhỏ, được xác định rõ ràng)
- Chạy từng tác vụ như một tiến trình độc lập → chống lại việc làm hỏng toàn bộ hệ thống
- Các mô đun chạy user mode → bảo vệ khỏi lỗi

# Hệ vi nhân



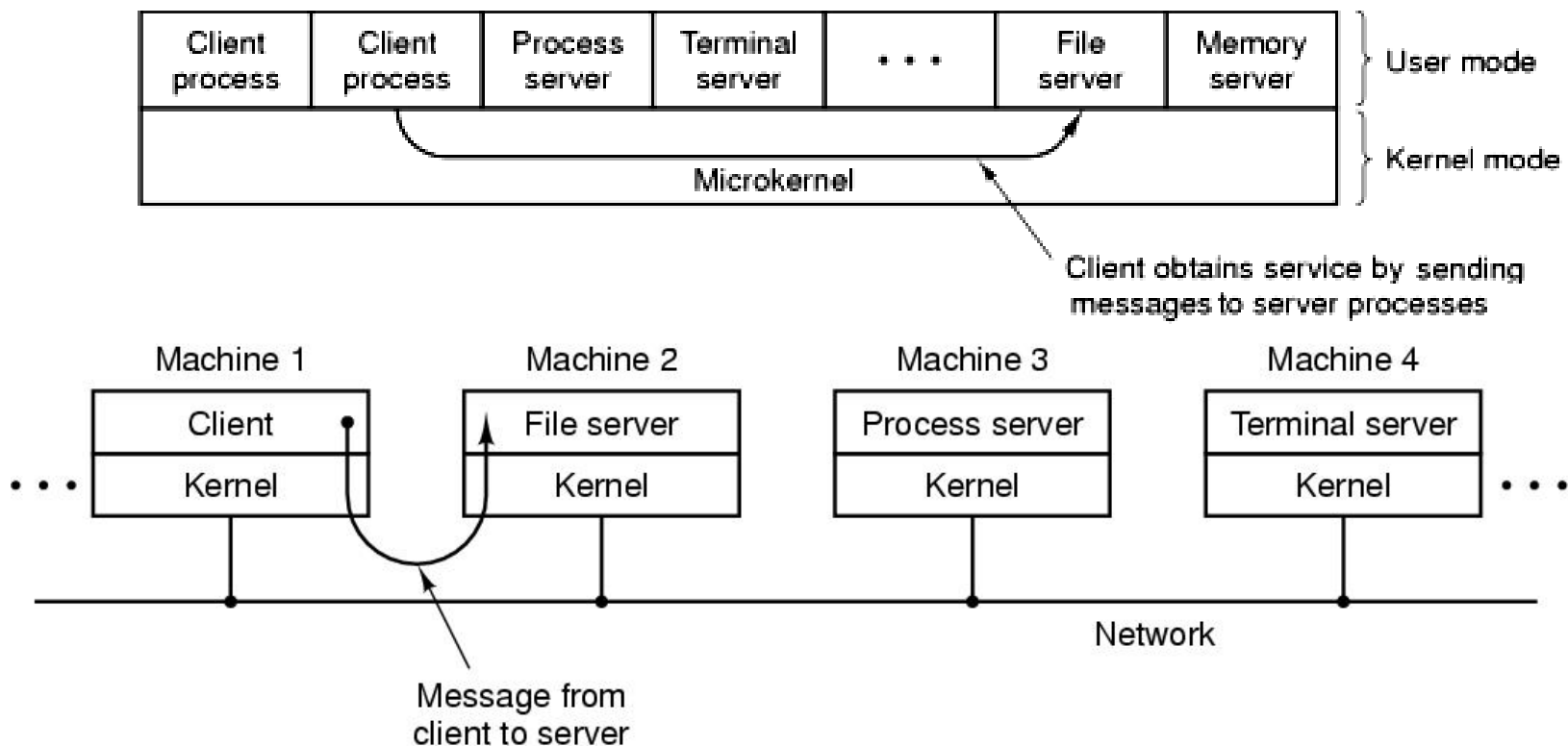
# Hệ vi nhân

- Ưu điểm
  - Dễ mở rộng và chuyển HDH sang các kiến trúc mới
  - Bảo mật & độ tin cậy cao (ít mã chạy hơn ở kernel mode)
- Nhược điểm
  - Định nghĩa thành phần cần thiết và không cần thiết như thế nào?
  - Hiệu suất quá tải của việc giao tiếp giữa user space và kernel space
- Ex: Symbian, Apple MacOS X Server



# Mô hình Client-Server

- Server cung cấp dịch vụ phục vụ yêu cầu của client
- Client sử dụng dịch vụ của server
- Liên lạc bằng cách truyền tin nhắn (message passing)
- Có thể chạy trên các máy tính khác nhau hoặc cùng một máy



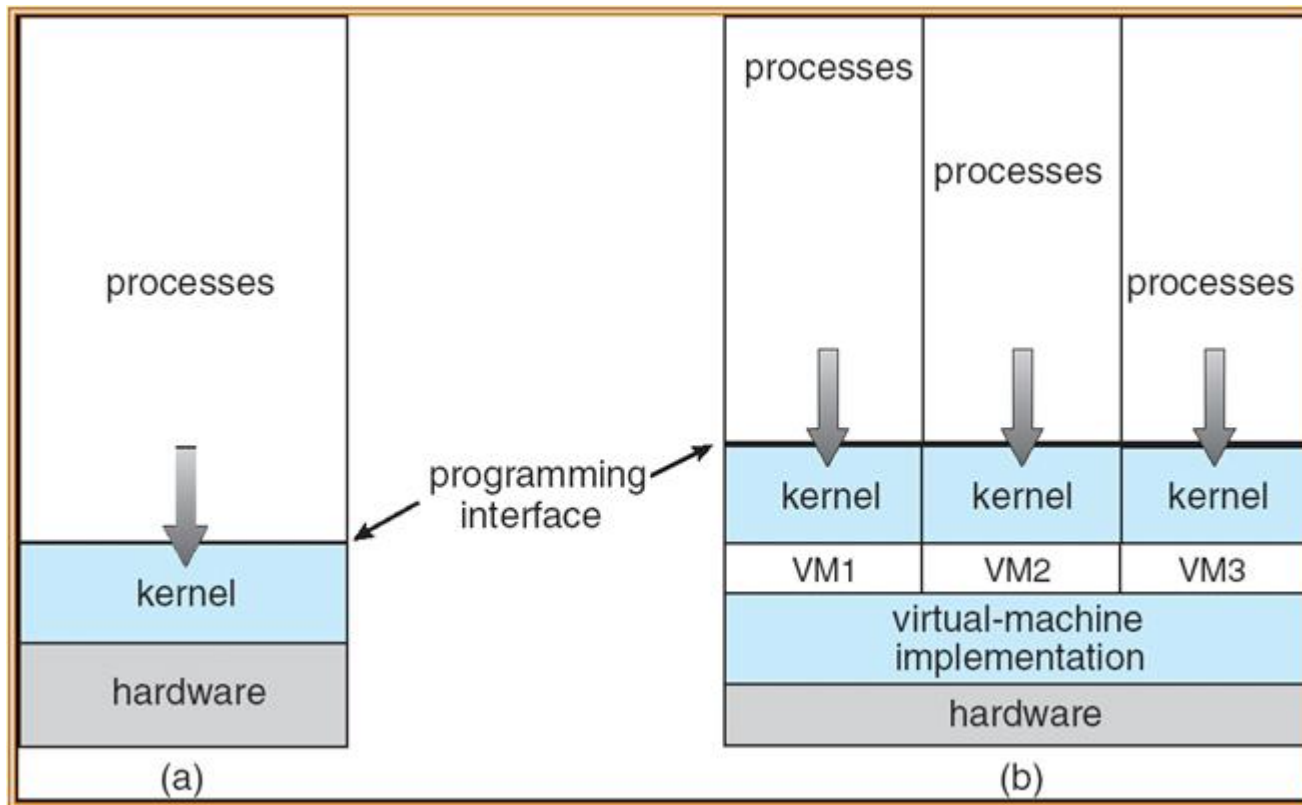
# Máy ảo

- Mỗi máy ảo
  - Không phải là máy mở rộng mà tất cả phần cứng đều được trừu tượng hóa bằng code
  - Giống hệt với phần cứng thực sự (cung cấp bản sao chính xác của máy thật)
  - Có thể chạy bất kỳ HDH nào và chạy trực tiếp trên phần cứng
- Các máy ảo khác nhau có thể chạy các HDH khác nhau
- Hỗ trợ chạy nhiều HDH cùng lúc (Windows, Linux)
- Phần mềm
  - VMware Workstation
  - OS running on CD (Linux, Ubuntu ...)
  - JVM (Java Virtual Machine)

# Máy ảo

- Ưu điểm
  - Hoàn toàn bảo vệ tài nguyên hệ thống
  - Giải quyết vấn đề tương thích hệ thống
  - Không làm gián đoạn (phá vỡ) hoạt động bình thường của hệ thống
- Nhược điểm
  - Không thể cấp phát toàn bộ disk cho Máy ảo

# Máy ảo



(a) Nonvirtual machine

(b) virtual machine

Virtual-machine implementation is called **virtual machine monitor**, or **virtualization layer**, or **hypervisor**