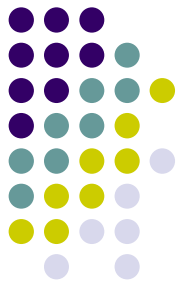


# HỆ ĐIỀU HÀNH

Giáo viên: Đỗ Tuấn Anh  
*Bộ môn Khoa học Máy tính*  
*Khoa Công nghệ Thông tin*  
*ĐHBK Hà Nội*  
[anhdt@it-hut.edu.vn](mailto:anhdt@it-hut.edu.vn)  
0989095167



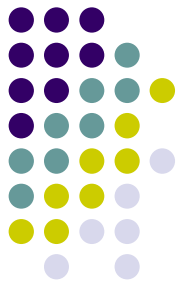
# MỤC ĐÍCH – YÊU CẦU



- Là giáo trình *cơ sở chuyên ngành*:
  - Xét các vấn đề HĐH bất kỳ phải giải quyết,
  - Phương thức giải quyết các vấn đề đó.
  - Hỗ trợ cho các môn khác trong việc xây dựng cơ sở cho Tin học.
  - Những v/d xem xét sẽ không lạc hậu trong tương lai.



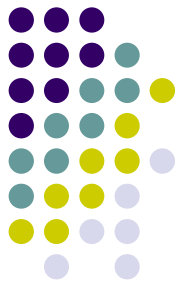
# MỤC ĐÍCH – YÊU CẦU



- Mang yếu tố *chuyên đề*:
  - Minh họa cho các v/đ lý thuyết,
  - Khoảng cách giữa và thực tế công nghệ ở Tin học nói chung và HĐH nói riêng gần như bằng 0.
- Như vậy: đây là một giáo trình khó, khá nặng nề.



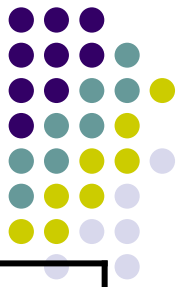
# TÀI LIỆU



- *A. Tanenbaum* Design and Implementation operating system.
- *A. Tanenbaum* Advanced Concepts to Operating Systems.
- *Microsoft Press* Inside to WINDOWS 2000.
- Nguyên lý hệ điều hành:
  - TS. Hà Quang Thụy
  - NXB Khoa học kỹ thuật
- Hệ điều hành: Tác giả: Ths. Nguyễn Thanh Tùng

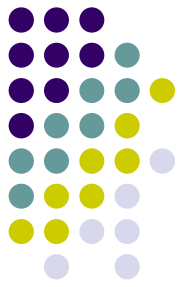


# Thời gian biểu

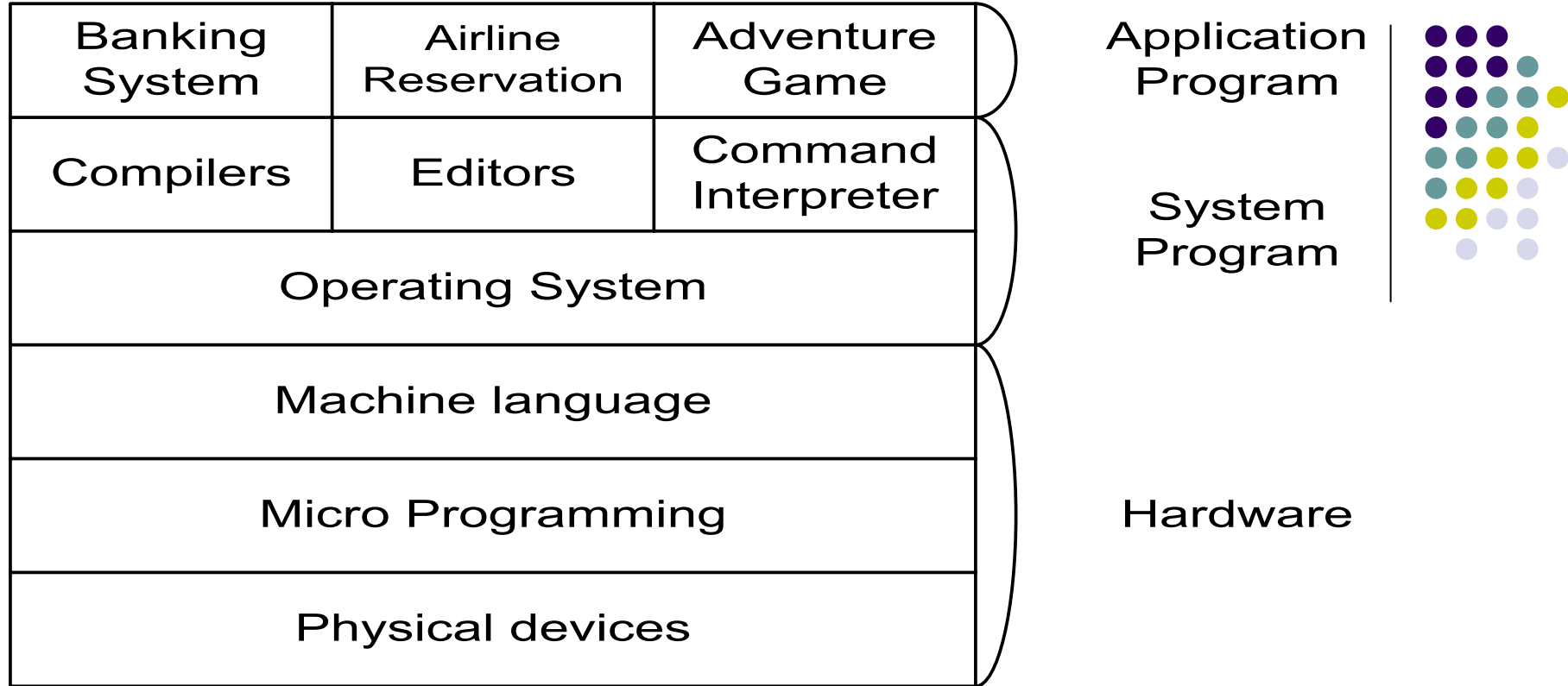


|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Chương 0: Giới thiệu Hệ điều hành

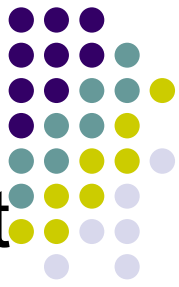


- 1. Giới thiệu về HĐH
  - 1.1 Phần cứng và phần mềm
    - Phần cứng:
      - Ngôn ngữ máy
      - Chương trình vi điều khiển – điều khiển trực tiếp các thiết bị
      - Thiết bị điện tử
    - Phần mềm
      - Chương trình hệ thống: quản lý hoạt động của máy tính
      - Chương trình ứng dụng: giải quyết các bài toán của người dùng.



Phần mềm tạo nên môi trường của hệ thống gọi là Hệ điều hành. Hệ điều hành điều khiển và quản lý tài nguyên và tạo môi trường cho các chương trình ứng dụng thực hiện thao tác với tài nguyên.

- Hệ điều hành thực hiện chế độ đặc quyền
- Trình dịch thực hiện ở chế độ không đặc quyền



# 1.2 Khái niệm Hệ điều hành

*Hệ điều hành là một chương trình hay một hệ chương trình*

- hoạt động giữa người sử dụng và phần cứng của máy tính.
- Chuẩn hóa giao diện người dùng đối với các hệ thống phần cứng khác nhau.
  - Sử dụng hiệu quả tài nguyên phần cứng
  - Khai thác tối đa hiệu suất của phần cứng
- *Hệ điều hành được coi như là **hệ thống quản lý tài nguyên**.*
- *Hệ điều hành được coi như là **phần mở rộng của hệ thống máy tính điện tử**.*



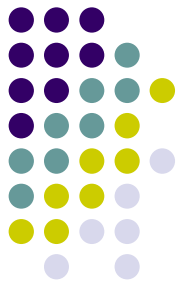


## 2. Lịch sử phát triển của HĐH

Lịch sử phát triển của HĐH luôn gắn liền với sự phát triển của máy tính điện tử

- Thế hệ thứ nhất (1945-1955)
  - Howard Aiken (Harvard) và John von Neumann (Princeton)
    - Xây dựng máy tính dùng bóng chân không
    - Kích thước lớn
    - Với hơn 10000 bóng chân không
  - Ngôn ngữ lập trình và Hệ điều hành chưa được biết đến
  - Đầu những năm 50->phiếu đục lỗ thay cho bảng điều khiển

## 2. Lịch sử phát triển của HĐH



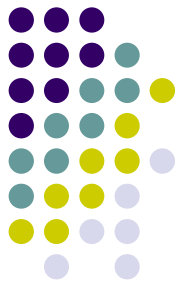
# Chương I. CÁC KHÁI NIỆM CƠ BẢN



- 1- Cấu trúc phân lớp của hệ thống tính toán
- Máy tính điện tử đầu tiên ra đời năm 1944-1945,
- MTĐT được xây dựng và hoạt động theo nguyên lý Von Neuman: *Máy tính được điều khiển bằng chương trình và trong câu lệnh của chương trình người ta chỉ nêu địa chỉ nơi chứa giá trị chứ không nêu trực tiếp giá trị.*

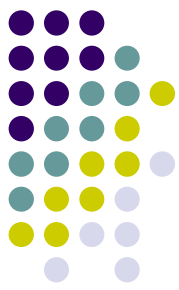


# Chương I. CÁC KHÁI NIỆM CƠ BẢN (tt.)



- Thế hệ thứ 2 (1955-1965)
  - Sự ra đời của thiết bị bán dẫn
  - lập trình FORTRAN và hợp ngữ
  - ***Hệ thống xử lý theo lô***
- Thế hệ thứ 3 (1965-1980)
  - mạch tích hợp (IC)
  - ***hệ điều hành chia sẻ thời gian***
- Thế hệ thứ 4 (1980-nay)
  - máy tính cá nhân (PC-Personal Computer)
  - ***hệ điều hành mạng và hệ điều hành phân tán***

# Cấu trúc phân lớp của hệ thống tính toán

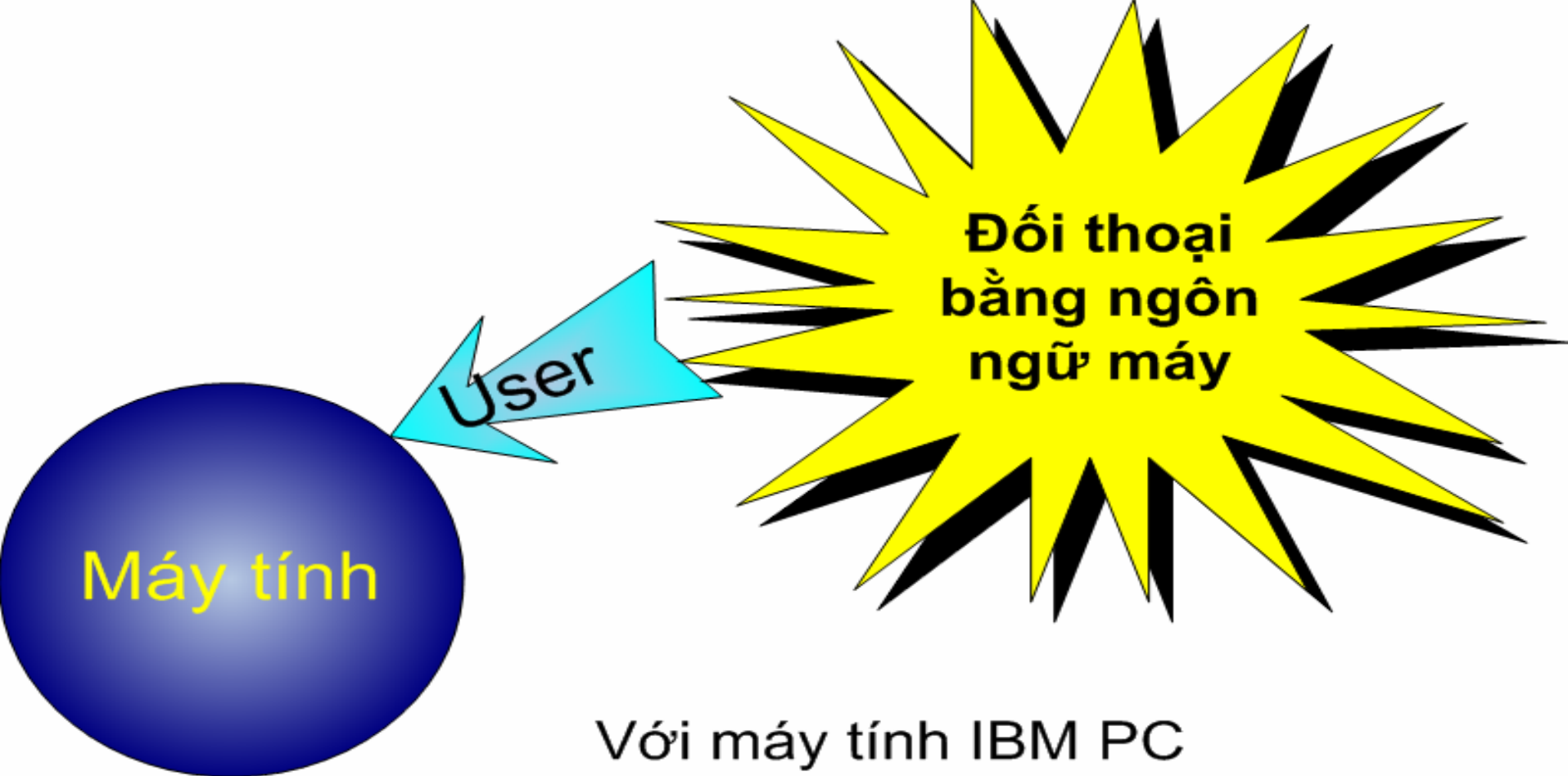


MTĐT

Hệ lệnh = {Mã lệnh}  
Command System =  
{Command Code}

Ngôn ngữ riêng  
(Ngôn ngữ máy)





### Mã lệnh

03 - Lệnh cộng số nguyên (Add)  
2B- Lệnh trừ số nguyên (Sub)  
EA - Lệnh chuyển điều khiển (JMP)  
33 - Cộng bit không nhớ (XOR)  
FA - Xoá trạng thái (CLI)  
.....

### Địa chỉ

0:46C - Lấy nhịp thời gian  
0:417- Trạng thái phím  
.....





- Người lập trình thường nhầm lẫn → năng suất lập trình thấp,
- Đã áp dụng nhiều biện pháp kích thích:
  - Kỷ luật hành chính,
  - Thưởng phạt kinh tế.
- Năng suất chỉ tăng chút ít và ổn định ở mức 8 câu lệnh/ngày công!
- Kết quả nghiên cứu tâm lý học: Bản chất con người không quen làm các công việc đơn điệu, không có tính quy luật, sớm hay muộn cũng sẽ có sai sót!



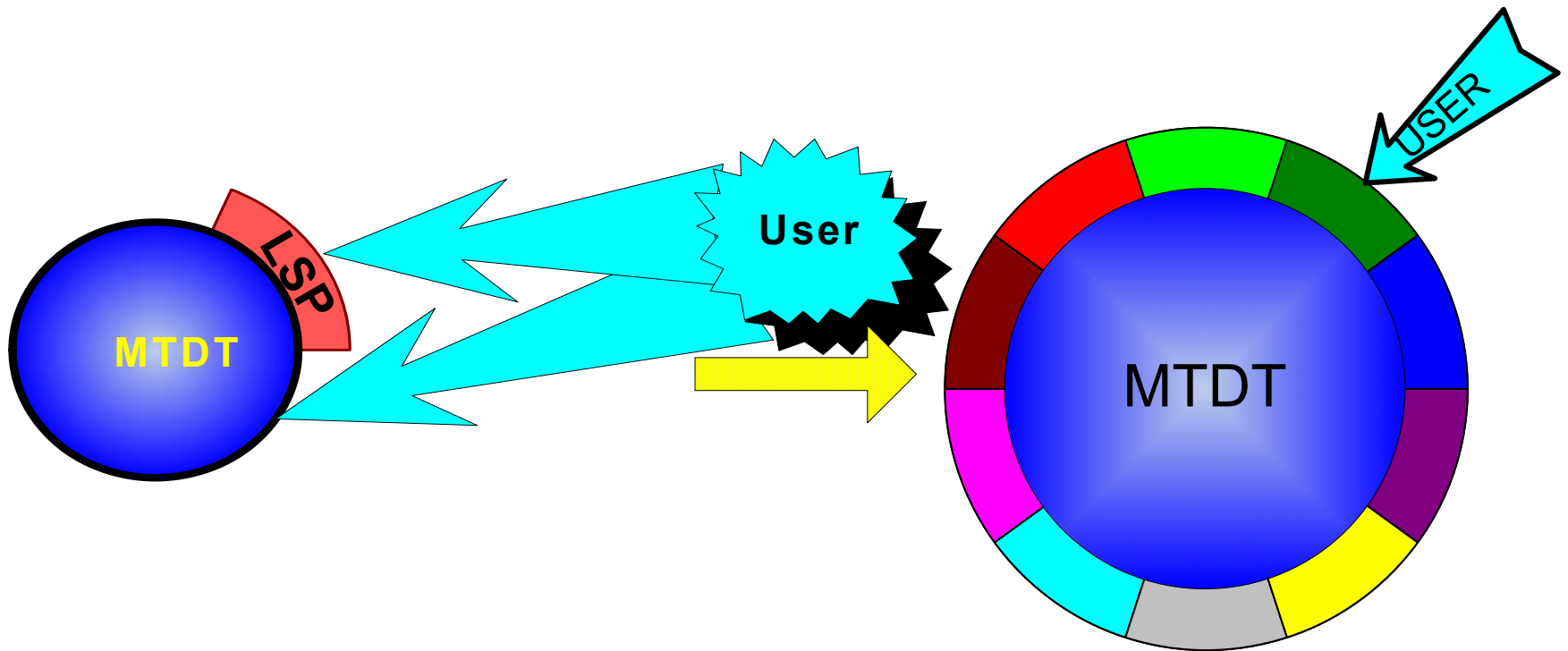
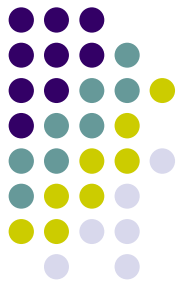
# Cấu trúc phân lớp của hệ thống tính toán

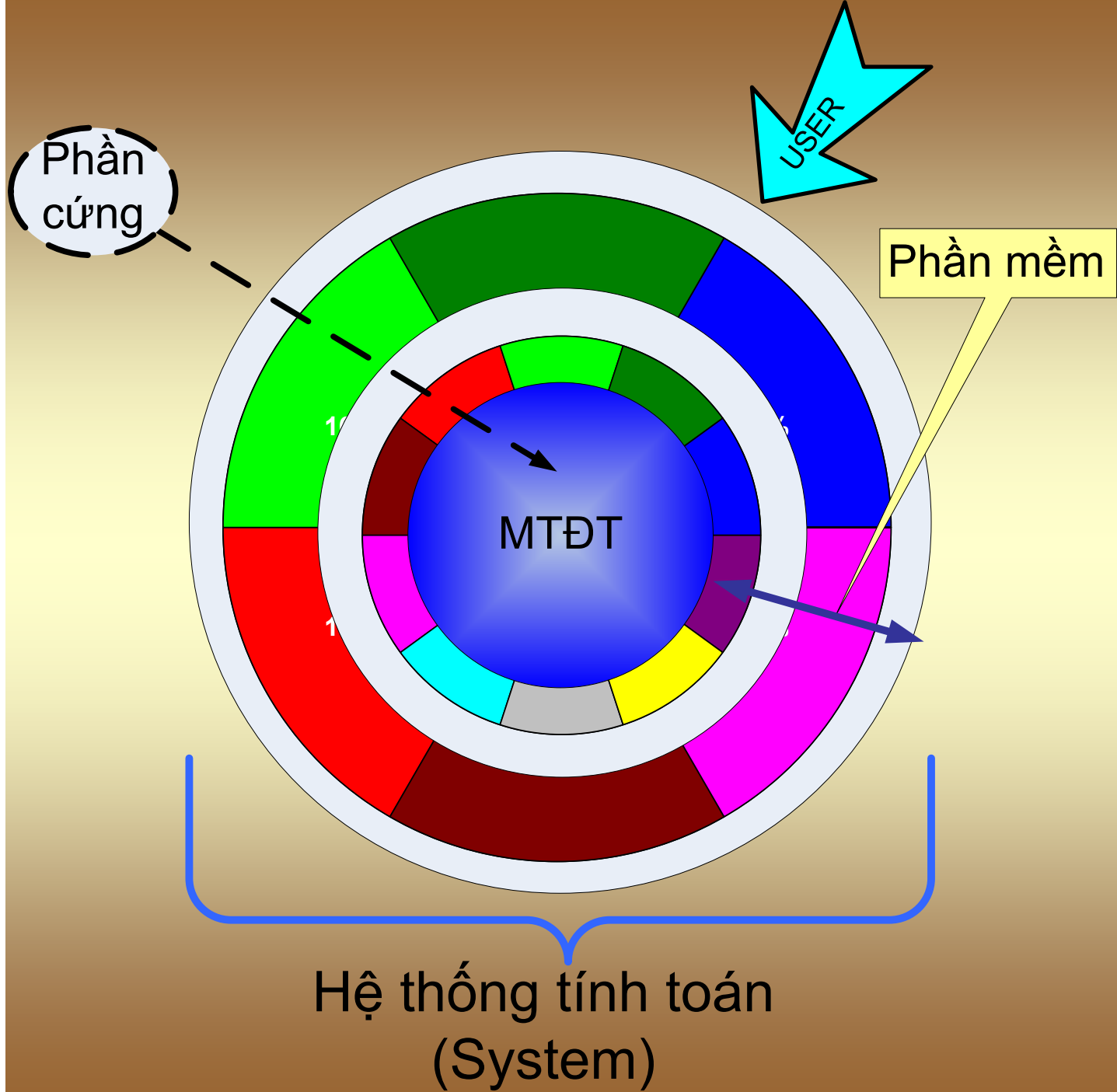


- Như vậy, để nâng cao năng suất - cần tác động vào MTĐT.
- $\exists$  các công việc mọi người và  $\exists$  CT đều cần (V/d – Trao đổi vào ra)  $\rightarrow$  tạo sẵn CT mẫu (Standard Programs – SP) cung cấp cùng với máy.
- Hình thành  $LSP = \{SP\}$

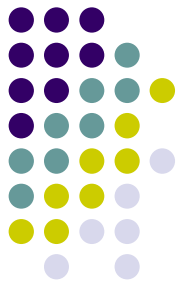








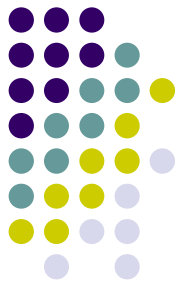
# Tác động phần mềm lên phần cứng



- Cơ sở hoá hệ lệnh:
  - Các lệnh phức tạp như  $x^{1/2}$ ,  $e^x$ ,  $|x|$  . . . dần dần được thay thế bằng CT con,
  - Tăng cường các lệnh xử lý bit.
- Tăng tốc độ của MT,
- Tăng tính vạn năng,
- Tăng độ tin cậy,
- Giảm giá thành,
- Cho phép phân các thiết bị thành từng nhóm độc lập, tăng độ mềm dẻo của cấu hình.



# Tác động phần mềm lên phần cứng



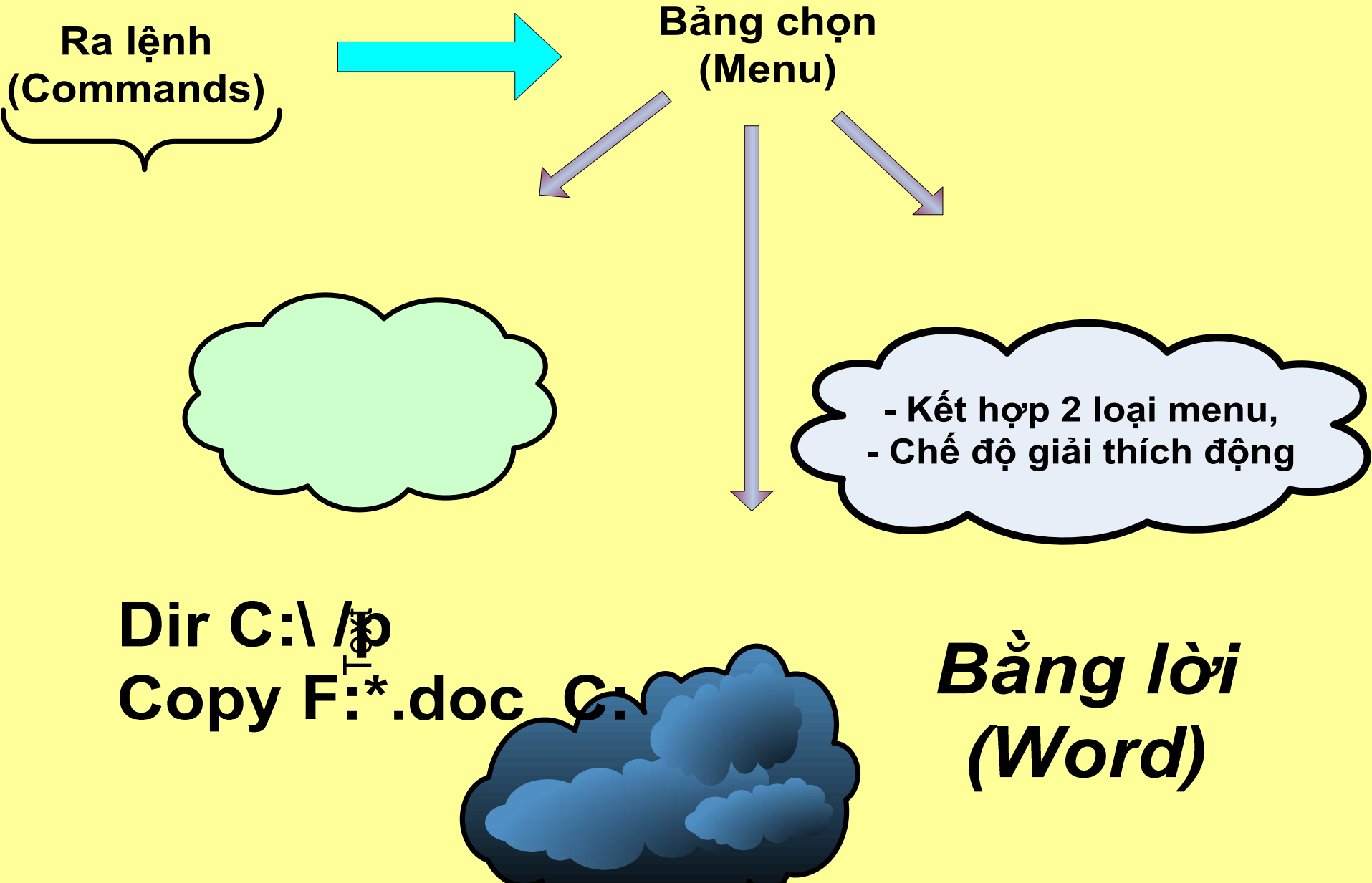
- Các yếu tố trên có sự tác động của tiến bộ công nghệ, nhưng phần mềm đóng vai trò quan trọng, nhiều khi có tính quyết định:
  - Bàn phím,
  - Máy in.

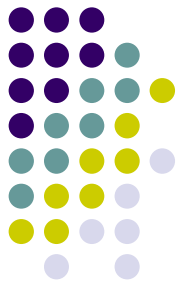




-

# Thay đổi nguyên lý làm việc:

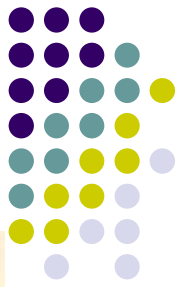




# Tác động phần mềm lên USER

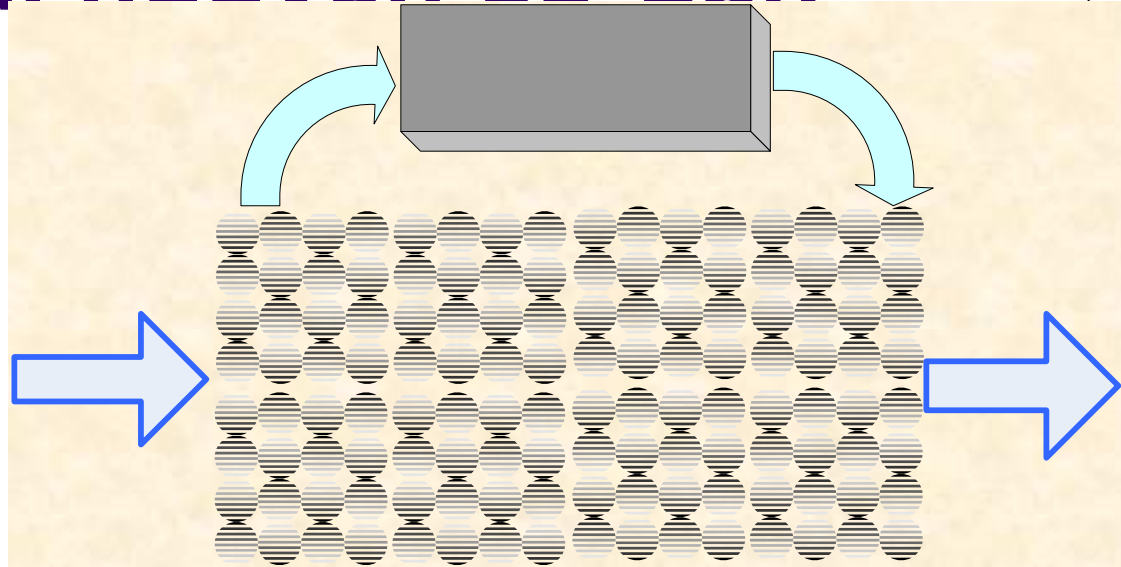
- Hiệu ứng tự đào tạo,
- Nguyên lý WYSIWYG,
- Giải phóng người dùng khỏi sự ràng buộc vào thiết bị vật lý cụ thể.

## 2 – Các tài nguyên cơ bản



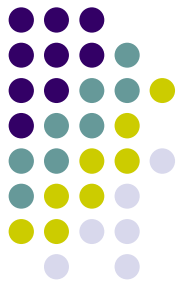
### a) Bộ nhớ:

- Vai trò,
- Gót chân Asin của hệ thống,
- Quan trọng: sử dụng như thế nào?
- Bảo vệ thông tin?



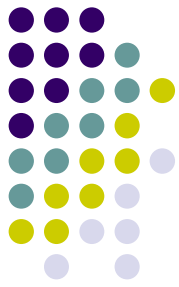
**INPUT**





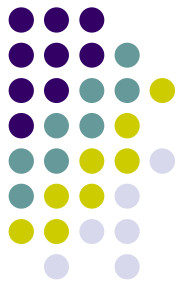
## b) PROCESSOR

- Điều khiển máy tính,
- Thực hiện các phép tính số học, lô gic và điều khiển,
- Có tốc độ rất lớn (vài chục triệu phép tính / giây),
- Thông thường có thời gian rảnh (thời gian “chết”) lớn → hiệu suất sử dụng thấp,
- V/đ: tăng hiệu suất sử dụng (giảm thời gian chết).



## C) THIẾT BỊ NGOẠI VI

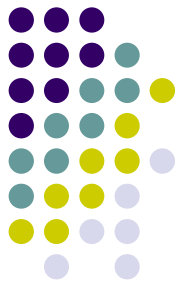
- Số lượng: *Nhiều*,
- Chất lượng: *Đa dạng*,
- Tốc độ: *Cực chậm* (so với Processor),
- V/đ: Phải đảm bảo:
  - Hệ thống *thích nghi* với số lượng và tính đa dạng,
  - Tốc độ thiết bị ngoại vi không ảnh hưởng đáng kể đến *năng suất* hệ thống.



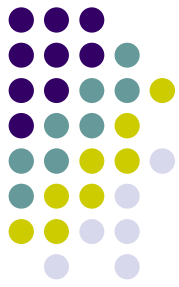
## D) Tài nguyên chương trình

- Cần phải có các chương trình cần thiết,
- Một chương trình được kích hoạt: phục vụ cho nhiều người dùng ( cấu trúc Reenter),
- Khai thác On-Line, RPC,
- Cách tổ chức chương trình: cấu trúc và đảm bảo cho cấu trúc hoạt động,

# Nhiệm vụ của hệ thống đối với tài nguyên



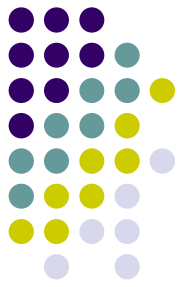
- 2 nhiệm vụ chung(không phụ thuộc vào loại tài nguyên):
  - Phân phối tài nguyên: Cho ai? Khi nào? Bao nhiêu (với loại chia sẻ được)?
  - Quản lý trạng thái tài nguyên: Còn tự do hay không hoặc số lượng còn tự do?
- Tồn tại nhiều giải thuật → Loại hệ thống:
  - Xử lý theo lô,
  - Phân chia thời gian,
  - Thời gian thực.



### 3 - ĐỊNH NGHĨA HỆ ĐIỀU HÀNH

- Có nhiều góc độ quan sát và đánh giá,
- Các đối tượng khác nhau có yêu cầu, đòi hỏi khác nhau đối với OS,
- Xét 4 góc độ:
  - Của người sử dụng,
  - Của nhà quản lý,
  - Của nhà kỹ thuật,
  - Của người lập trình hệ thống.

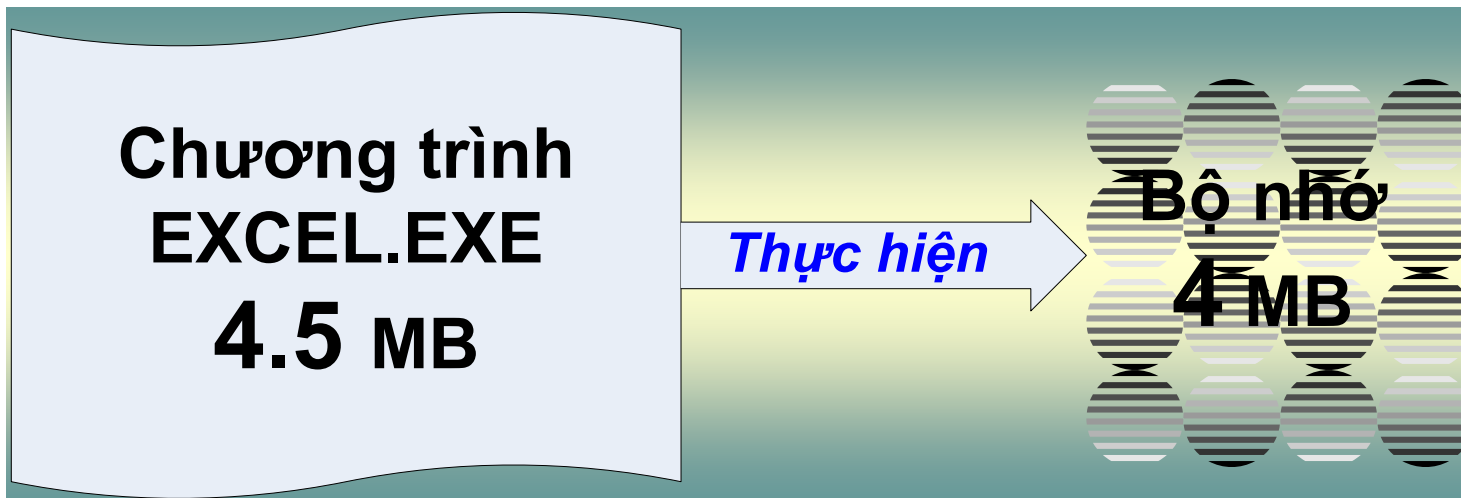
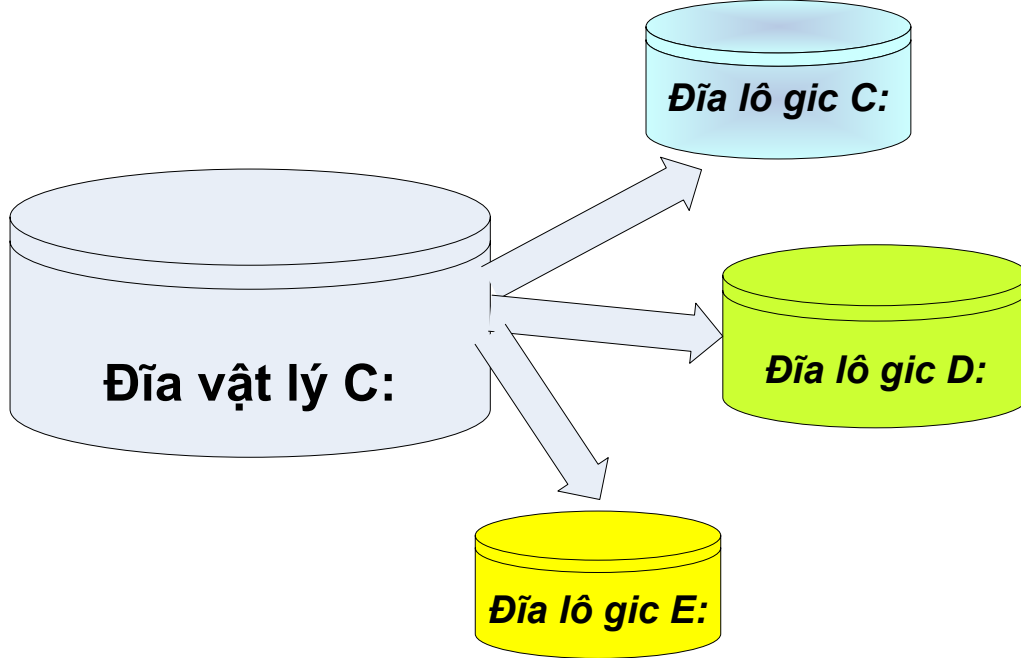
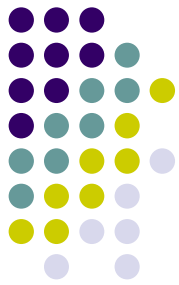
# ĐỊNH NGHĨA HỆ ĐIỀU HÀNH



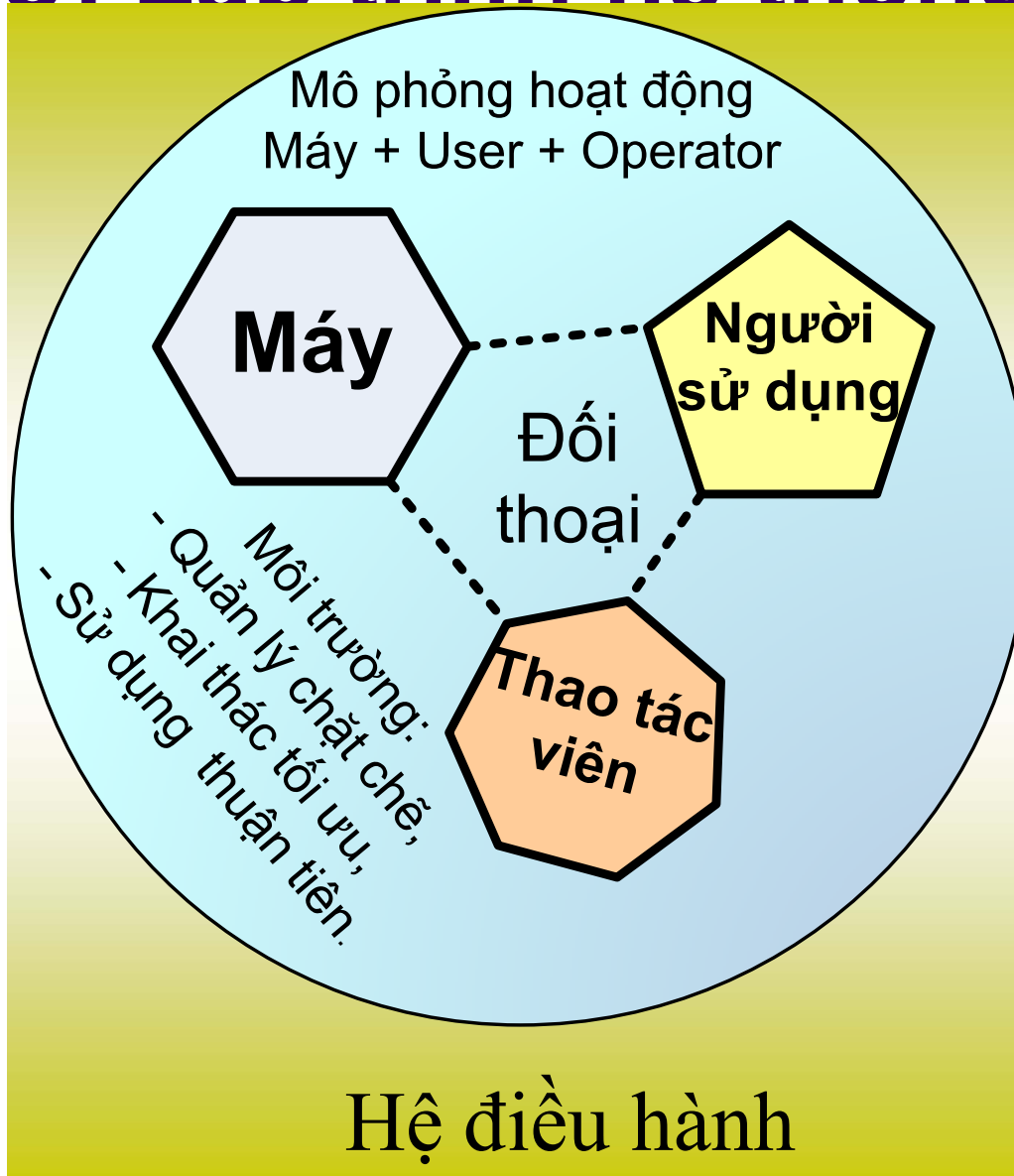
- Người dùng: Thuận tiện,
- Nhà quản lý: Quản lý chặt chẽ, khai thác tối ưu,
- Nhà kỹ thuật:



Hệ điều hành  
Hệ thống chương trình **bao**  
**trùm** lên máy vật lý, tạo ra một  
**máy lô gíc** với những **tài**  
**nguyên** và **khả năng mới**

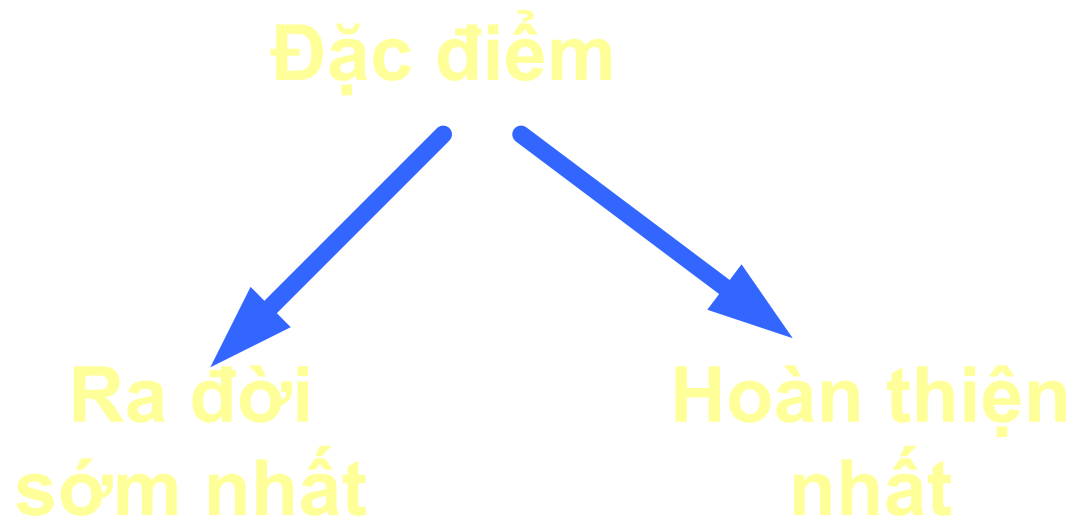


# Người Lập trình hệ thống





- Đối thoại: để hệ thống *gọn nhẹ* + *linh hoạt*,
- Đối thoại  $\rightarrow \exists$  *ngôn ngữ đối thoại* (bằng lời hoặc cử chỉ).
- Ở mô phỏng 2 đối tượng con người  $\rightarrow$  là hệ thống trí tuệ nhân tạo, là *hệ chuyên gia*,





- **Ra đời sớm nhất:**
- 04/1951 xã hội mới biết và tin vào khả năng giải quyết các bài toán phi số của MT,
- 1952 - Von Neuman đề xuất tư tưởng xây dựng “CT tự hoàn thiện” ,
- 1961 – Bell Lab – Các CT trò chơi Animal và Core Ware,
- Khai thác thực tế các hệ CG: 1971-1972.
- OS – xây dựng từ 1950,
- 1965 - Hệ ĐH nổi tiếng OS IBM 360

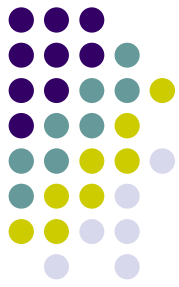


- **Hoàn thiện nhất:**
  - Thống kê UNESCO: 73% số công trình không hoàn thành do khâu đặt v/đ,
  - Các HCG khác: Cán bộ chuyên ngành + Cán bộ lập trình,
  - OS:
    - Người lập trình giải quyết bài toán của chính mình
    - Hiểu rõ: V/đ+khả năng công cụ+ khả năng bản thân
- 1974: 3 công trình xây dựng kỹ thuật tiêu biểu đỉnh cao trí tuệ loài người:
- Hệ thống ĐT tự động liên lục địa,  
Hệ thống Appolo đưa người lên mặt trăng,  
OS IBM 360.



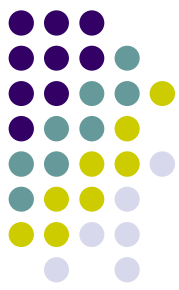
## 4 – TÍNH CHẤT CHUNG CỦA OS

- A) Tin cậy và chuẩn xác,
- B) Bảo vệ,
- C) Kế thừa và thích nghi,
- D) Hiệu quả,
- E) Thuận tiện.



# Tin cậy và chuẩn xác

- Mọi công việc trong hệ thống đều phải có kiểm tra:
  - Kiểm tra môi trường điều kiện thực hiện,
  - Kiểm tra kết quả thực hiện,
- Nhiều chức năng KT: chuyển giao cho phần cứng.
- Ví dụ: Lệnh **COPY A:F1.TXT B:**
- Sau khi KT cú pháp, bắt đầu thực hiện lệnh. Lần lượt hệ thống sẽ KT gì và có thể có thông báo nào?



- Kt CARD I/O,
- Tồn tại ổ đĩa?
- Thiết bị điện tử ổ đĩa?
- Động cơ ổ đĩa?
- Khả năng truy nhập của ổ đĩa?
- Khả năng truy nhập đĩa?
- Tồn tại file F1.TXT?
- Khả năng truy nhập file?
- . . . . .

- So sánh:

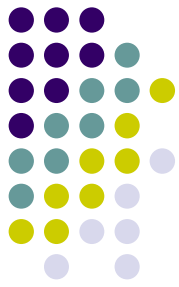
SCANDISK

NDD

DEFRAG

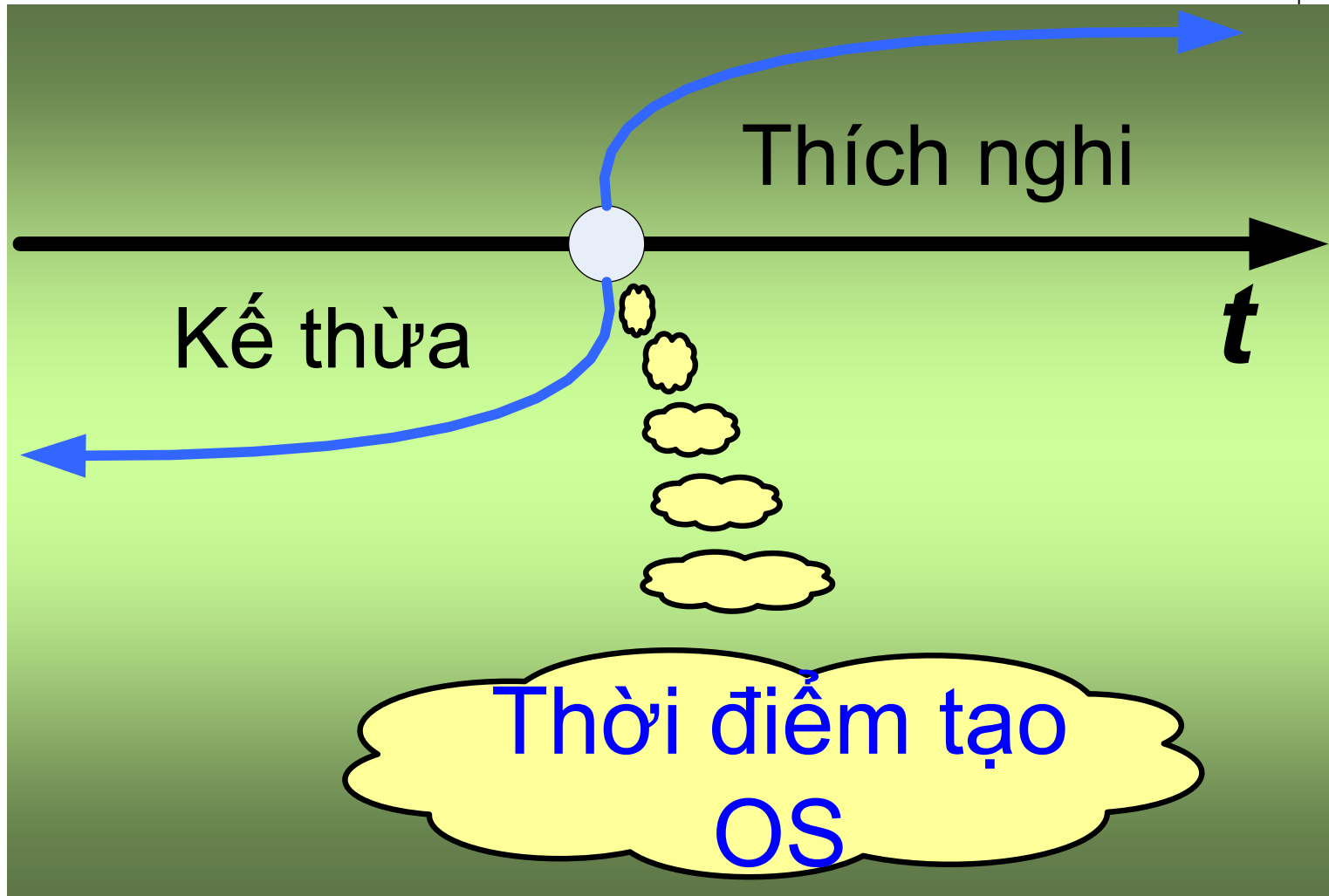
SPEEDISK

# BẢO VỆ



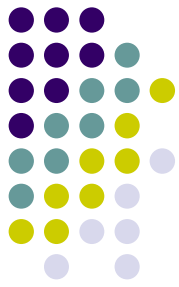
- Hạn chế truy nhập không hợp thức,
- Hạn chế ảnh hưởng sai sót vô tình hay cố ý,
- Bảo vệ:
  - Nhiều mức,
  - Nhiều công cụ,
  - Nhiều thời điểm và giai đoạn khác nhau.
- **Chú ý:** *bảo vệ và chống bảo vệ: cùng mức → không thể đảm bảo an toàn tuyệt đối!*

# Kế thừa và thích nghi

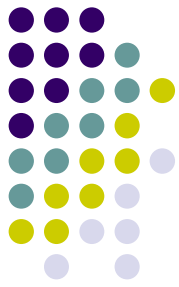




# 5 - NGUYÊN LÝ TỔ CHỨC VÀ HOẠT ĐỘNG

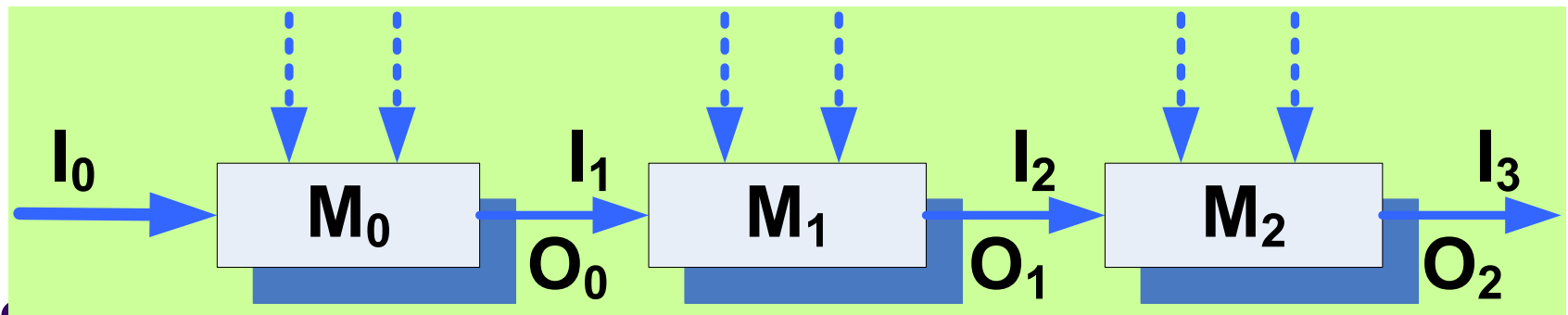


- Nguyên lý mô đun,
- Nguyên lý phủ chức năng,
- Nguyên lý Macroprocessor,
- Nguyên lý bảng tham số điều khiển,
- Nguyên lý giá trị chuẩn,
- Nguyên lý 2 loại tham số.



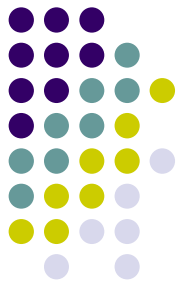
# NGUYÊN LÝ MÔ ĐUN

- Mỗi công việc  $\Leftrightarrow$  mô đun CT độc lập,
- Các mô đun – liên kết với nhau thông qua Input/Output:



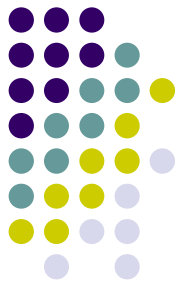
- Các mô đun được thiết kế theo chức năng, thành phần hệ thống.

# NGUYÊN LÝ PHỦ CHỨC NĂNG

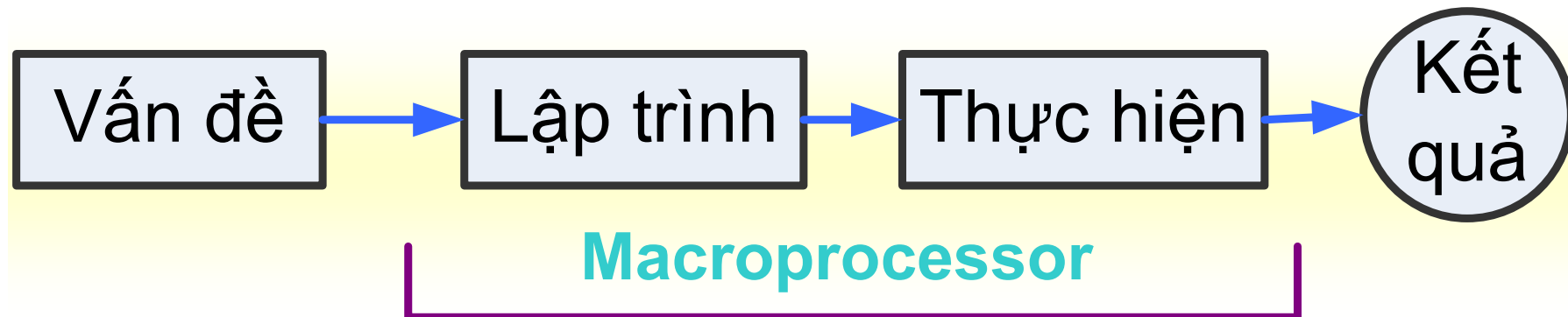


- Mỗi công việc trong hệ thống thông thường có thể thực hiện bằng nhiều cách với nhiều công cụ khác nhau,
- *Lý do:*
- Mỗi mô đun có **hiệu ứng phụ** chức năng,
- Người dùng có quyền khai thác mọi hiệu ứng phụ không phụ thuộc vào việc công bố,
- Lập trình: Phải đảm bảo các tính chất của OS với mọi hiệu ứng phụ,
- *Vai trò:*
  - Đảm bảo thuận tiện cho người dùng,
  - Đảm bảo an toàn chức năng của hệ thống,
- *Ví dụ:* In một file.

# NGUYÊN LÝ MACROPROCESSOR



- Trong OS không có sẵn CT giải quyết v/đ,
- Khi cần thiết: Hệ thống tạo ra CT và thực hiện CT tạo ra:



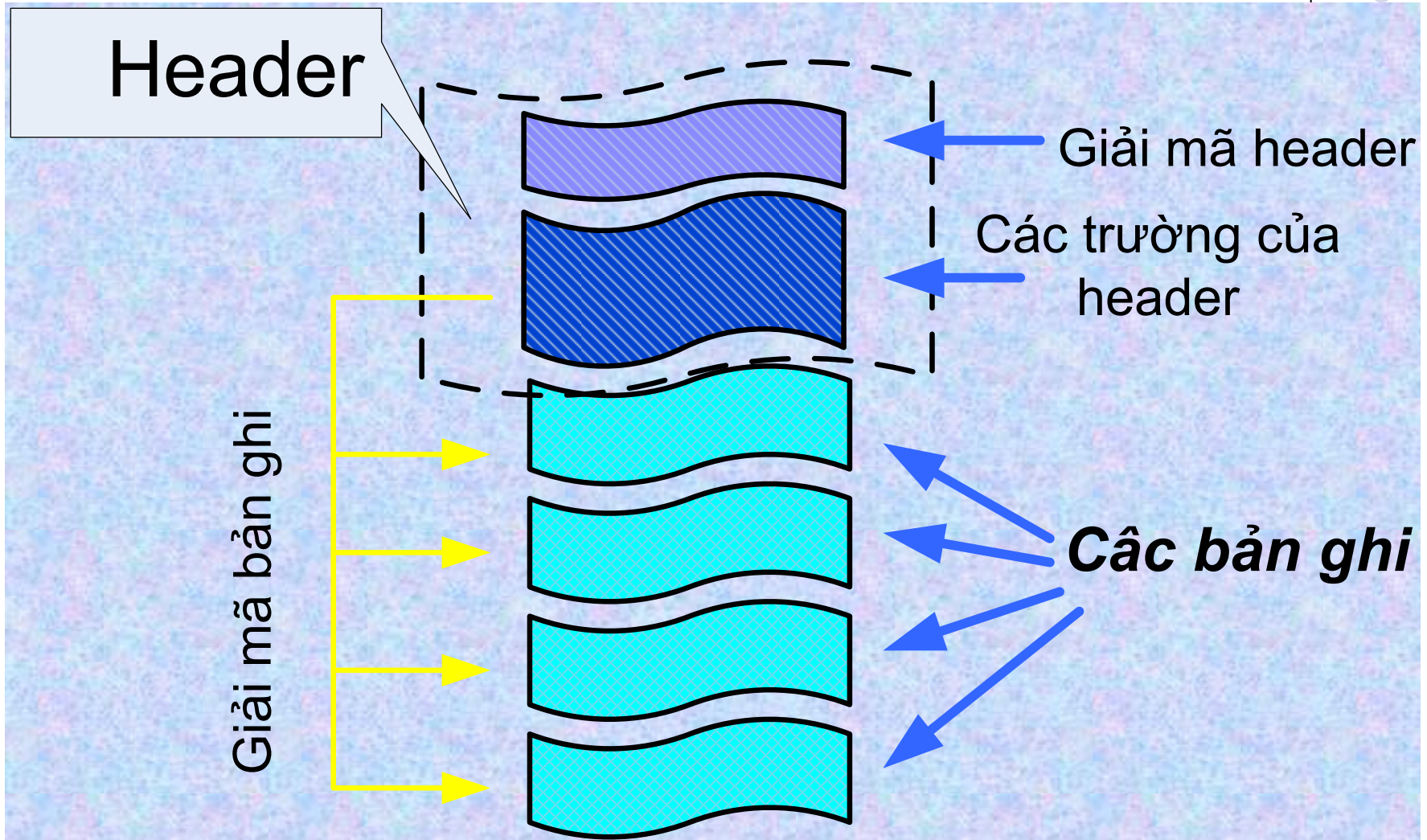
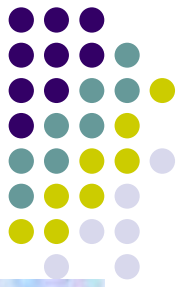
- Nguyên lý này áp dụng với cả bản thân **toàn bộ OS**. Trên địa chỉ có các thành phần. Khi cần các thành phần được lắp ráp thành **HỆ ĐIỀU HÀNH** (Nạp hệ thống).
- **Lưu ý:** Các nguyên lý *Phủ chức năng và Macroprocessor* trái với lý thuyết lập trình có cấu trúc.

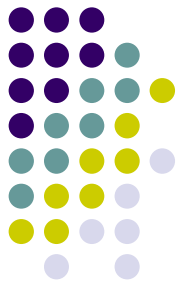
## NGUYÊN LÝ BẢNG THAM SỐ ĐIỀU KHIỂN



- Mỗi đối tượng trong OS  $\Leftrightarrow$  Bảng tham số (Control Table, Control Block),
- Hệ thống không bao giờ tham chiếu tới đối tượng vật lý mà chỉ tham chiếu tới bảng tham số điều khiển tương ứng.
- Với các đĩa từ, CD – bảng tham số ghi ở phần đầu – Vùng hệ thống (System Area),
- Với các files – Header.

# Cấu trúc file định kiểu





## Một số loại bảng tham số :

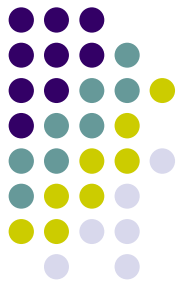
- Cho WINDOWS: Win.ini,
- Cho MS DOS: Config.sys,
- Cho WINWORD: Winword.ini,
- Bảng tham số cấu hình hệ thống: phục vụ cho mọi hệ điều hành: lưu trữ trong CMOS,

# NGUYÊN LÝ GIÁ TRỊ CHUẨN



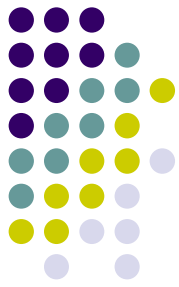
- Cách gọi khác: Nguyên tắc ngầm định (Default),
- Hệ thống chuẩn bị bảng giá trị cho các tham số - bảng giá trị chuẩn,
- Khi hoạt động: nếu tham số thiếu giá trị → OS lấy từ bảng giá trị chuẩn.
- **Vai trò** của nguyên lý:
  - Thuận tiện: không phải nhắc lại những giá trị thường dùng,
  - Người dùng không cần biết đầy đủ hoặc sâu về hệ thống.





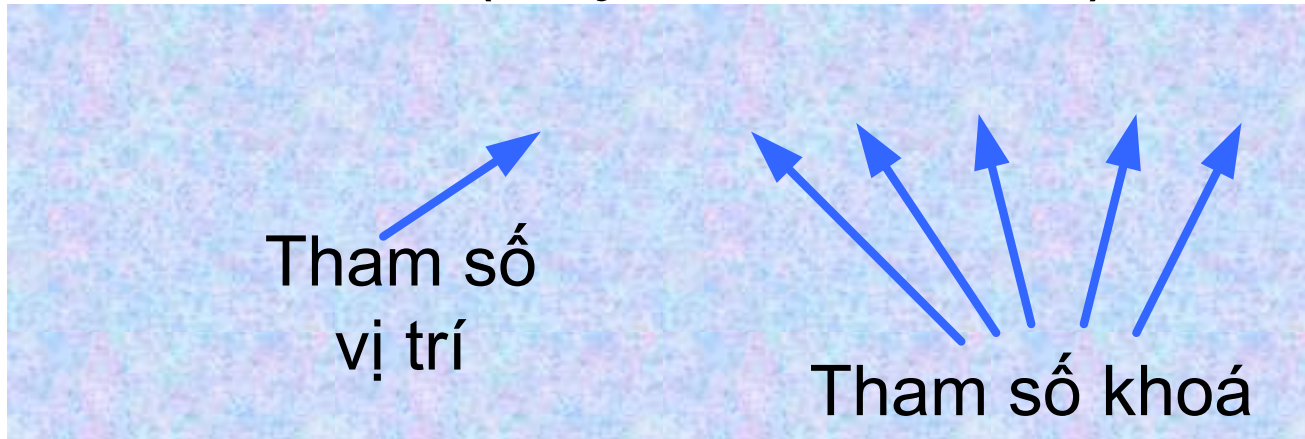
# Nguyên lý giá trị chuẩn

- Tác động lên giá trị tham số hoặc bảng giá trị chuẩn:
  - Startup,
  - Autoexec.bat,
  - Control Panel
- Ví dụ: `c:\csdl>dir`
- Tham số thiếu giá trị:
  - Ổ đĩa?
  - Thư mục?
  - Xem gì?
  - Quy cách đưa ra?
  - Nơi ra?

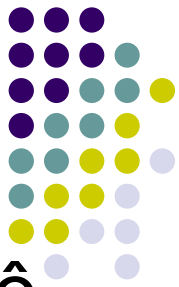


# NGUYÊN LÝ 2 LOẠI THAM SỐ

- 2 loại tham số:
- Tham số vị trí (Position Parameters),
- Tham số khoá (Keyword Param.).



- Tham số khoá – theo trình tự tùy ý.



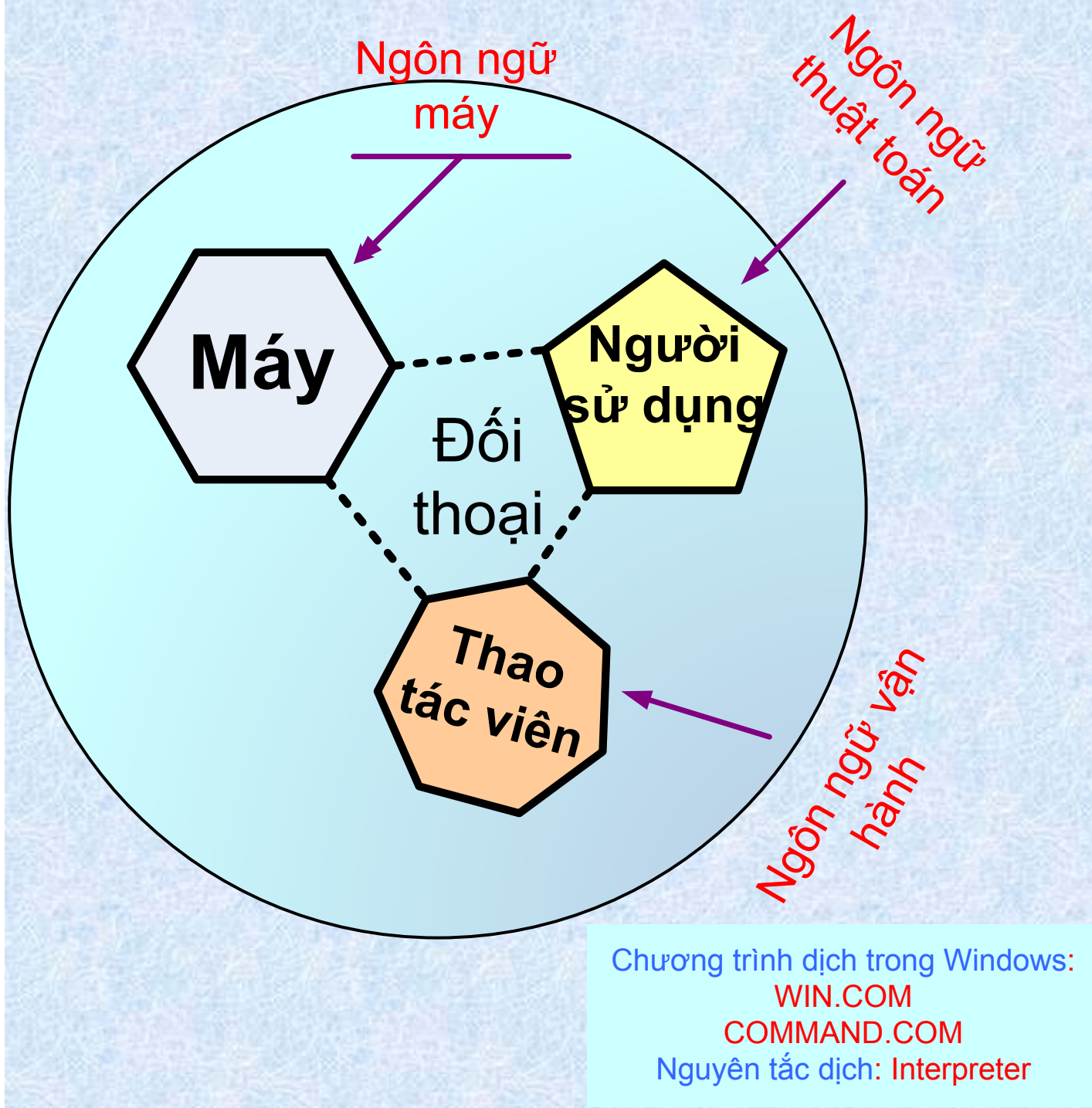
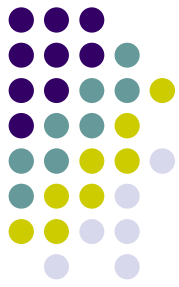
## 6 – THÀNH PHẦN

- Nhiều các phân chia theo chức năng, mức độ chi tiết,
  - Hệ thống Supervisor,
  - Hệ thống quản lý thiết bị ngoại vi,
  - Hệ thống quản lý files,
  - Hệ thống các chương trình điều khiển:
    - Điều phối nhiệm vụ,
    - Monitor,
    - Biên bản hệ thống,
- Các chương trình phục vụ hệ thống.

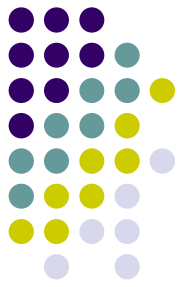
## Thành phần



- Lưu ý: ngôn ngữ không phải là thành phần hệ thống, nhưng trong thành phần hệ thống có một số CT dịch.
- Phân biệt: Chương trình phục vụ hệ thống và chương trình ứng dụng

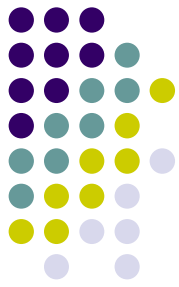


## II – QUẢN LÝ FILES VÀ THIẾT BỊ NGOẠI VI



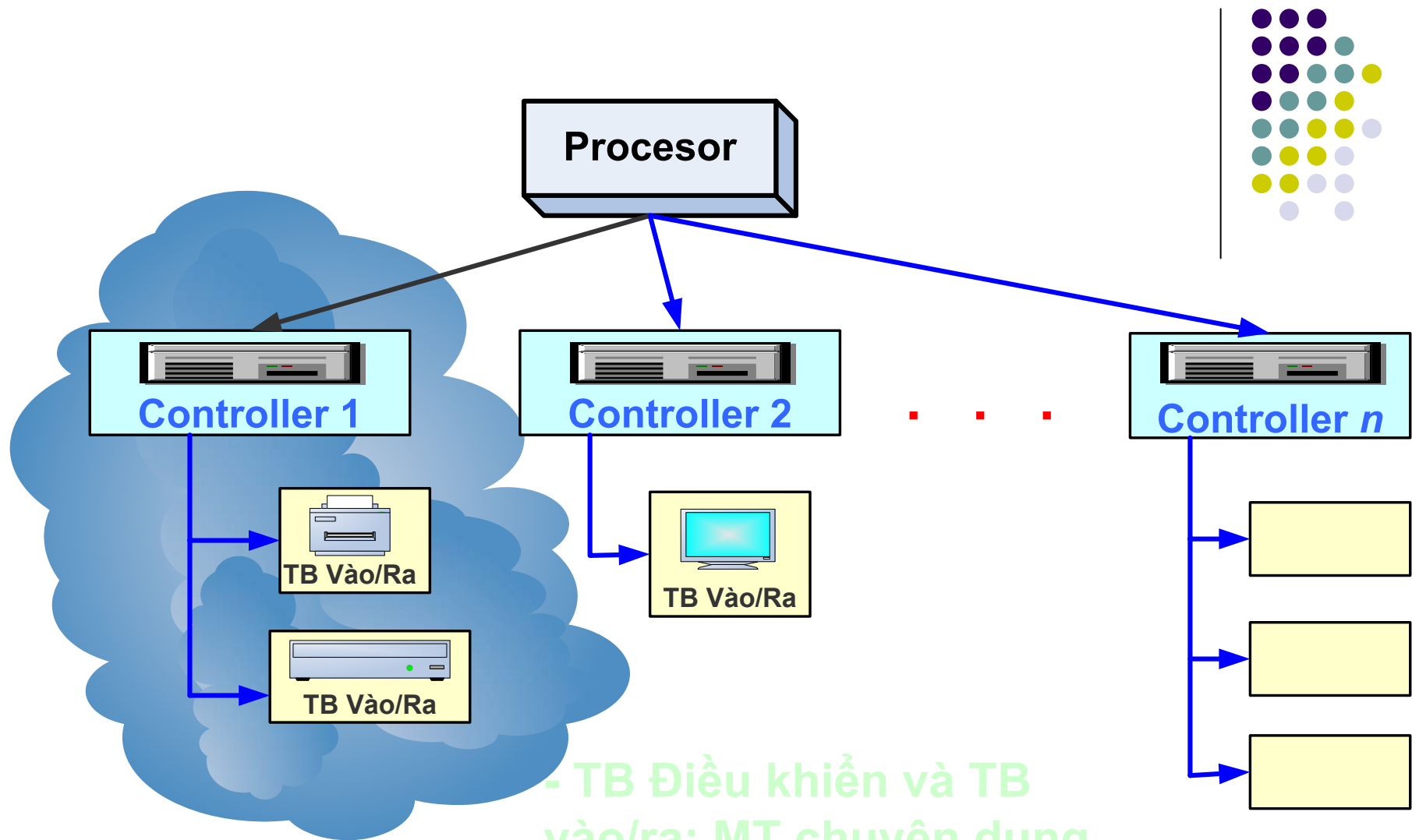
- **Quản lý thiết bị ngoại vi:** Cần đảm bảo hệ thống thích nghi với:
  - Số lượng nhiều,
  - Chất lượng đa dạng,
  - Thuận tiện cho người dùng.
- **Quản lý files:** Cho phép người dùng:
  - Tạo files ở các loại bộ nhớ ngoài,
  - Tìm kiếm, truy nhập files,
  - Đảm bảo độc lập giữa CT và thiết bị

# 1 – Nguyên tắc phân cấp trong quản lý thiết bị ngoại vi



- Máy tính thế hệ I và II: Processor làm việc trực tiếp với thiết bị ngoại vi,
- Hạn chế: Tốc độ - Số lượng - Chứng loại,
- Từ thế hệ III trở lên:

Processor → TB điều khiển → TB ngoại vi  
(Control Devices)  
(Controllers)

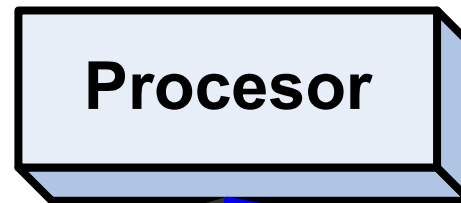


**Kênh  
(Channel)**

- TB Điều khiển và TB vào/ra: MT chuyên dụng
- Hệ lệnh + Ngôn ngữ riêng**
- Hoạt động độc lập với nhau và với Processor.



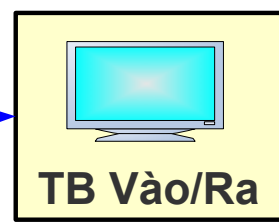
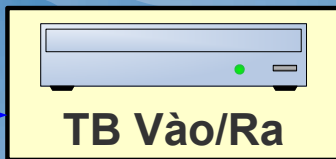
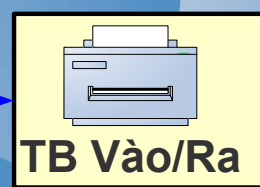
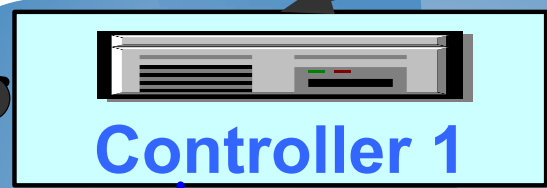
TRAO ĐỔI  
VÀO/RA



Thực hiện công  
việc của mình



CT kênh  
(Channel Prog)



Ngắt vào/ra (I/O Interrupt)  
Mã trở về (Return Code)

T Kênh

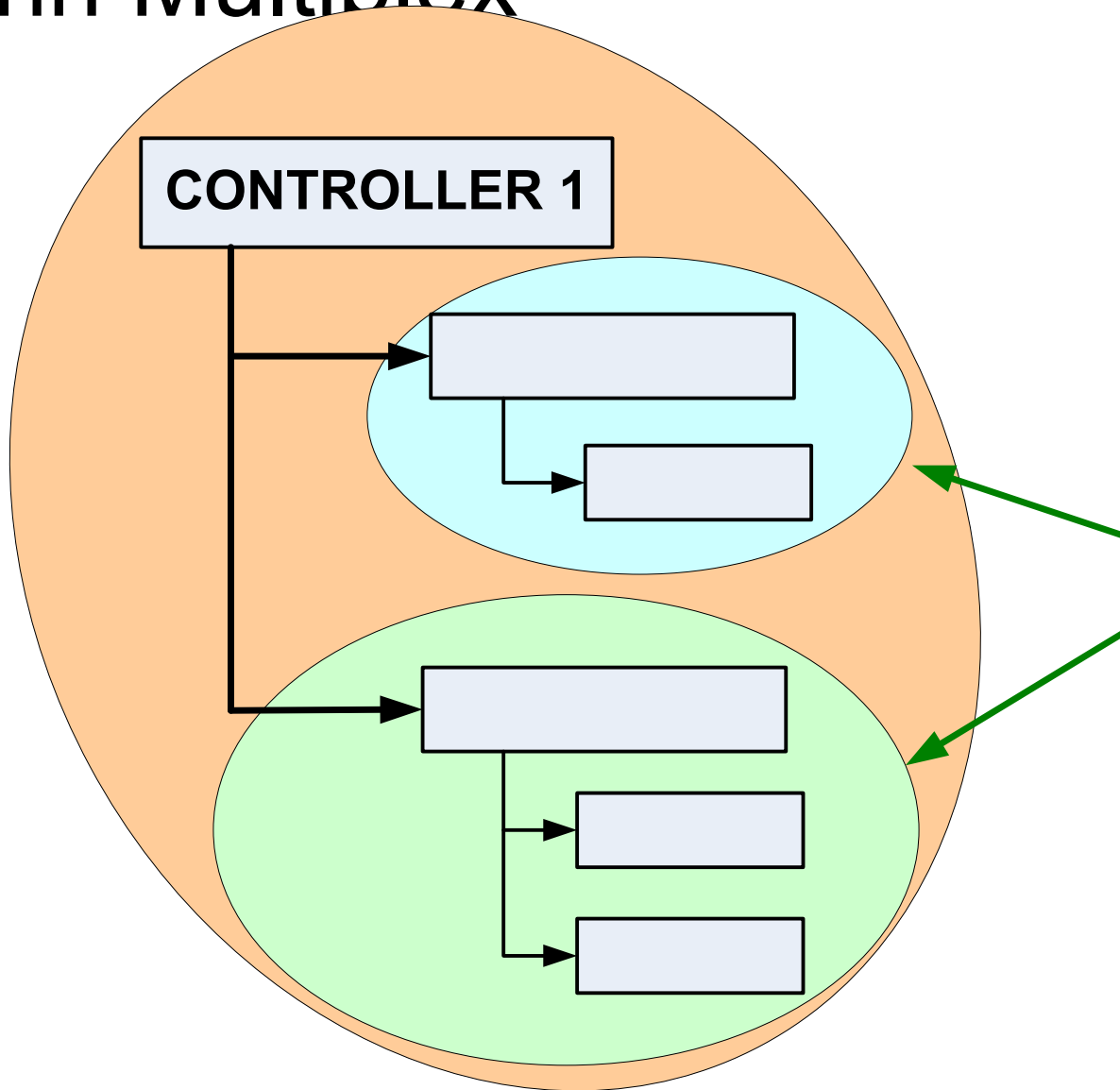
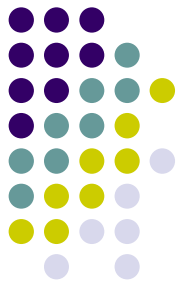
Kênh  
(Channel)



## Nguyên tắc phân cấp trong quản lý thiết bị ngoại vi

- Phép trao đổi vào ra: thực hiện theo nguyên lý Macroprocessor,
- Với máy vi tính: Thiết bị điều khiển vào ra  $\equiv$  I/O Card,
- Máy Card on Board,
- Lập trình trên Card vào/ra: Viết TOOLS khởi tạo chương trình kênh,
- Khái niệm kênh bó (Multiplex), Card Multimedia.

# Kênh Multiplex

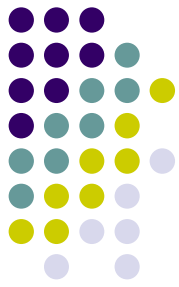
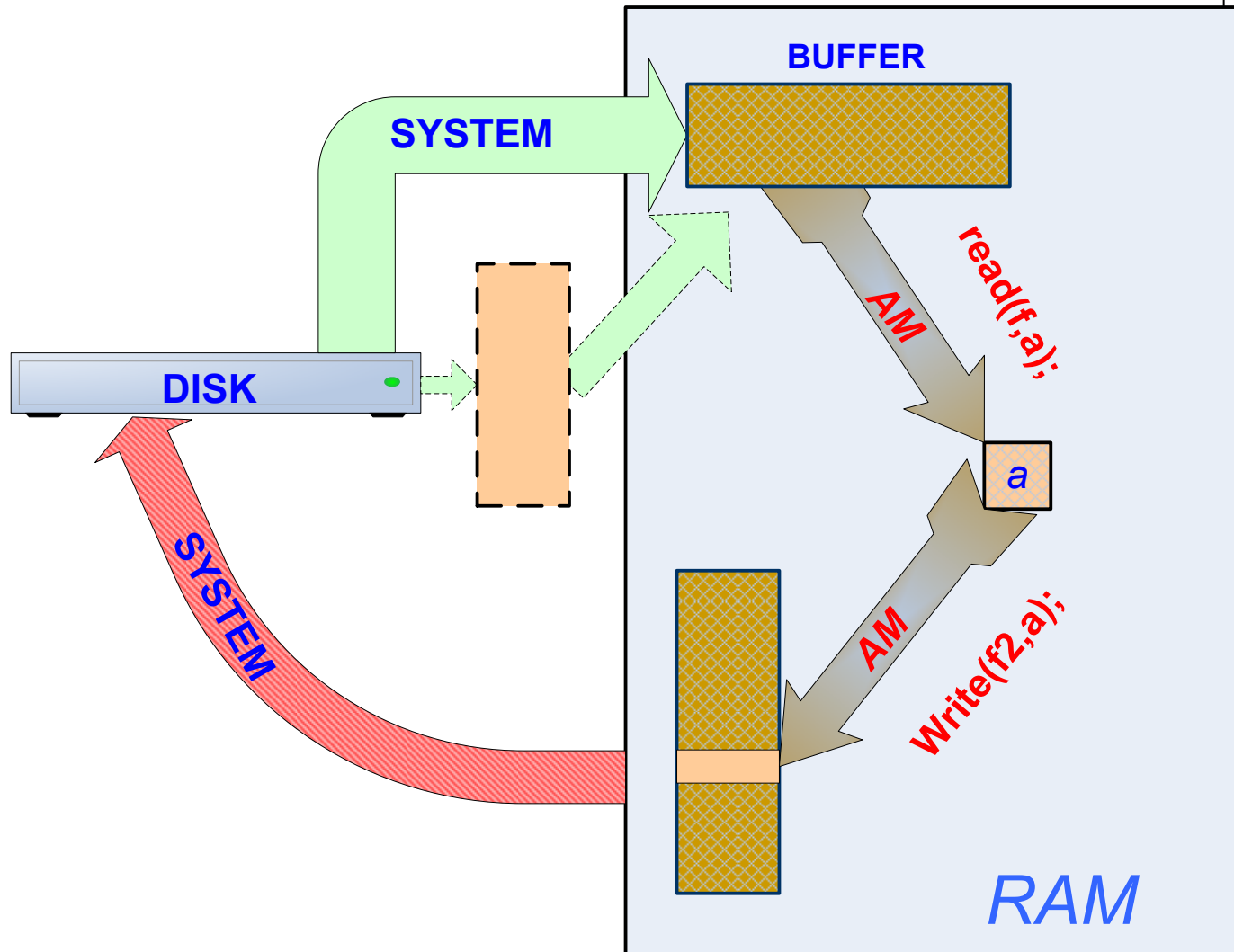


**CTRL**

**Kênh Multiplex**

## 2 - KỸ THUẬT PHÒNG ĐỆM

- Khái niệm phòng đệm (Buffer) của OS.

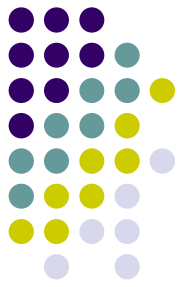


# KỸ THUẬT PHÒNG ĐỆM

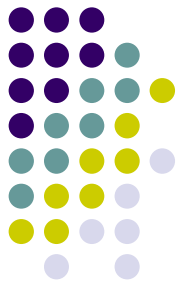


- Cơ chế phục vụ phòng đệm,
- Vấn đề đóng file output, FLUSH(F),
- Vai trò phòng đệm:
  - Song song giữa trao đổi vào ra và xử lý,
  - Đảm bảo độc lập:
    - Thông tin và phương tiện mang,
    - Bản ghi lô gíc và vật lý,
    - Lưu trữ và xử lý,
  - Giảm số lần truy nhập vật lý: Giả thiết mỗi lần truy nhập vật lý: 0.01", truy nhập kiểu BYTE.

# KỸ THUẬT PHÒNG ĐỆM

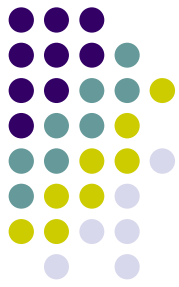


|      | <b>Không có<br/>Buffer</b> | <b>Buffer<br/>512B</b> |
|------|----------------------------|------------------------|
| 1B   | 0.01''                     | 0.01''                 |
| 512B | ~5''                       | 0.01''                 |
| 5KB  | ~50''                      | 0.1''                  |
| 50KB | ~8'                        | 1''                    |



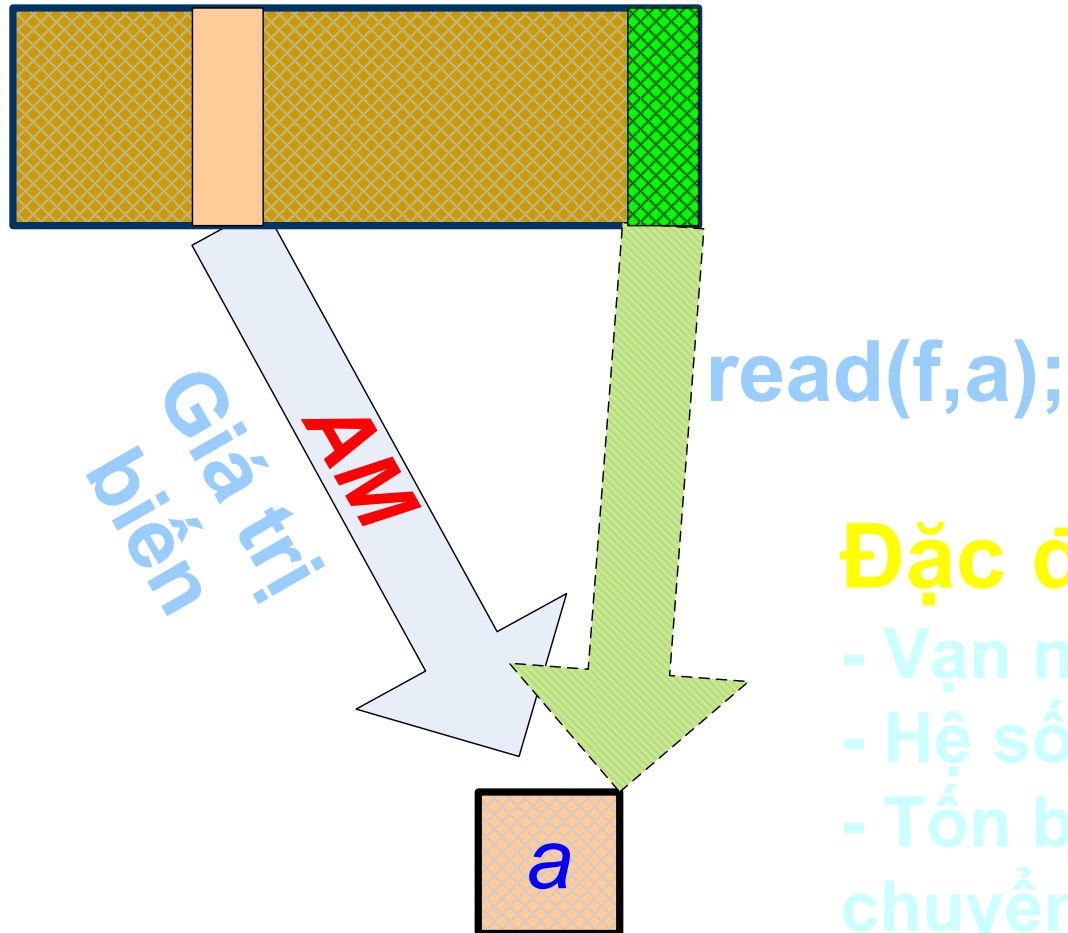
## Các loại phòng đệm

- Phòng đệm chung hoặc gắn với file,
- Các Hệ QTCSDL còn hệ thống phòng đệm riêng để nâng độ linh hoạt và tốc độ xử lý,
- Các loại bộ nhớ Cache và phòng đệm.
- Ba kiểu tổ chức chính:
  - Phòng đệm truy nhập theo giá trị,
  - Phòng đệm truy nhập theo địa chỉ,
  - Phòng đệm vòng tròn.



## Các loại phòng đệm

- A) Phòng đệm truy nhập theo giá trị:

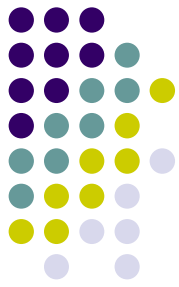


### Đặc điểm:

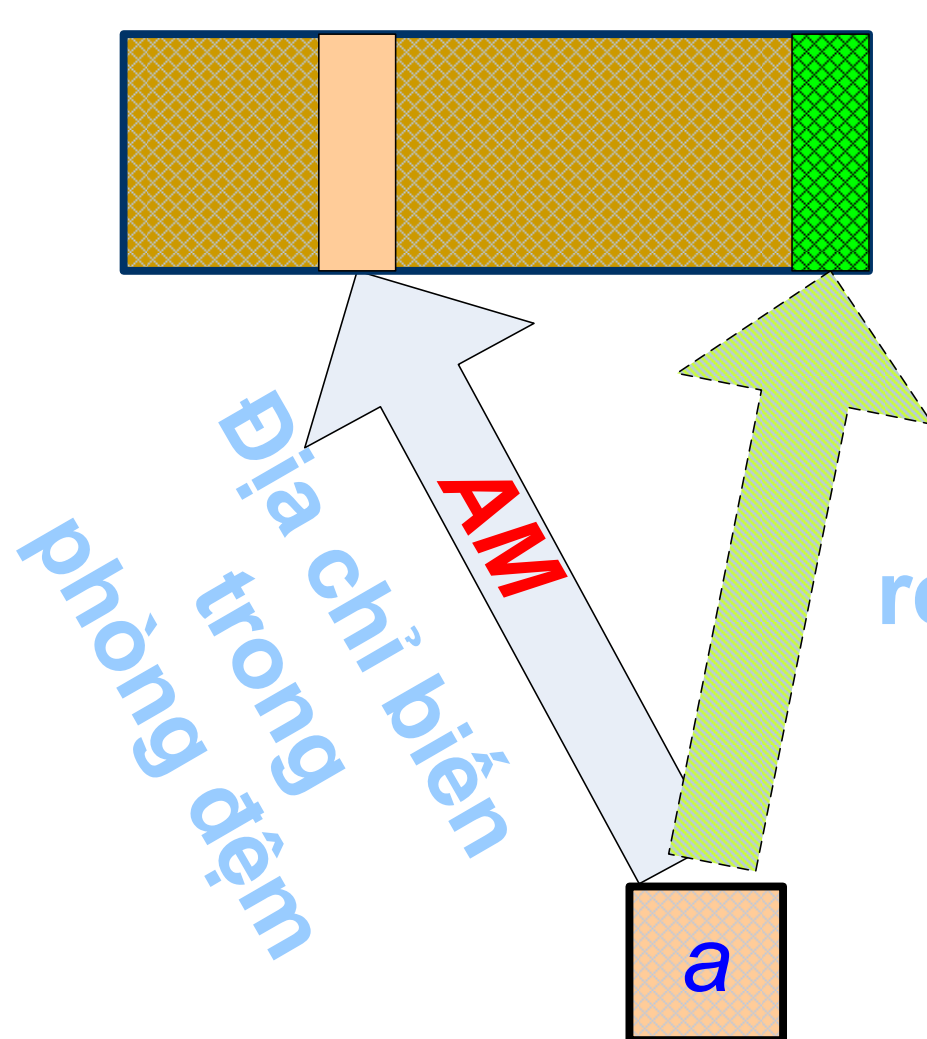
- Vụn năng,
- Hệ số song song cao,
- Tồn bộ nhớ và thời gian chuyển thông tin trong bộ nhớ



## Các loại phòng đệm



- B) Phòng đệm truy nhập theo địa chỉ:



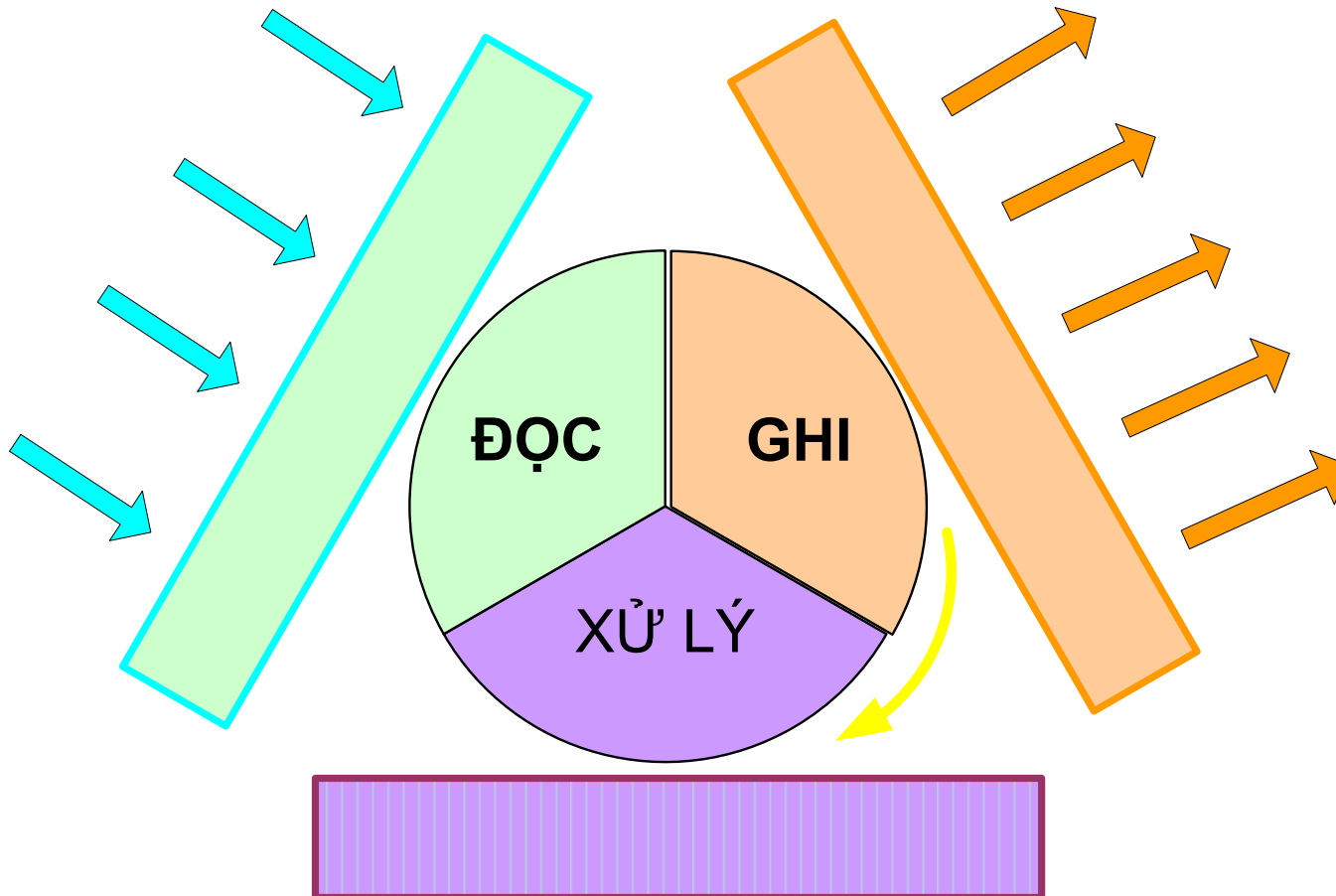
### Đặc điểm:

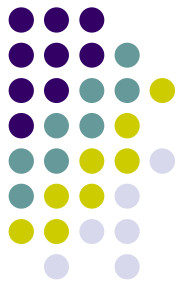
- Kém vận năng,
- Hệ số song song thấp,
- Không tốn bộ nhớ và thời gian chuyển thông tin trong bộ nhớ.

`read(f,a);`

## Các loại phòng đệm

- C) Phòng đệm vòng tròn: thường áp dụng cho các hệ QT CSDL.



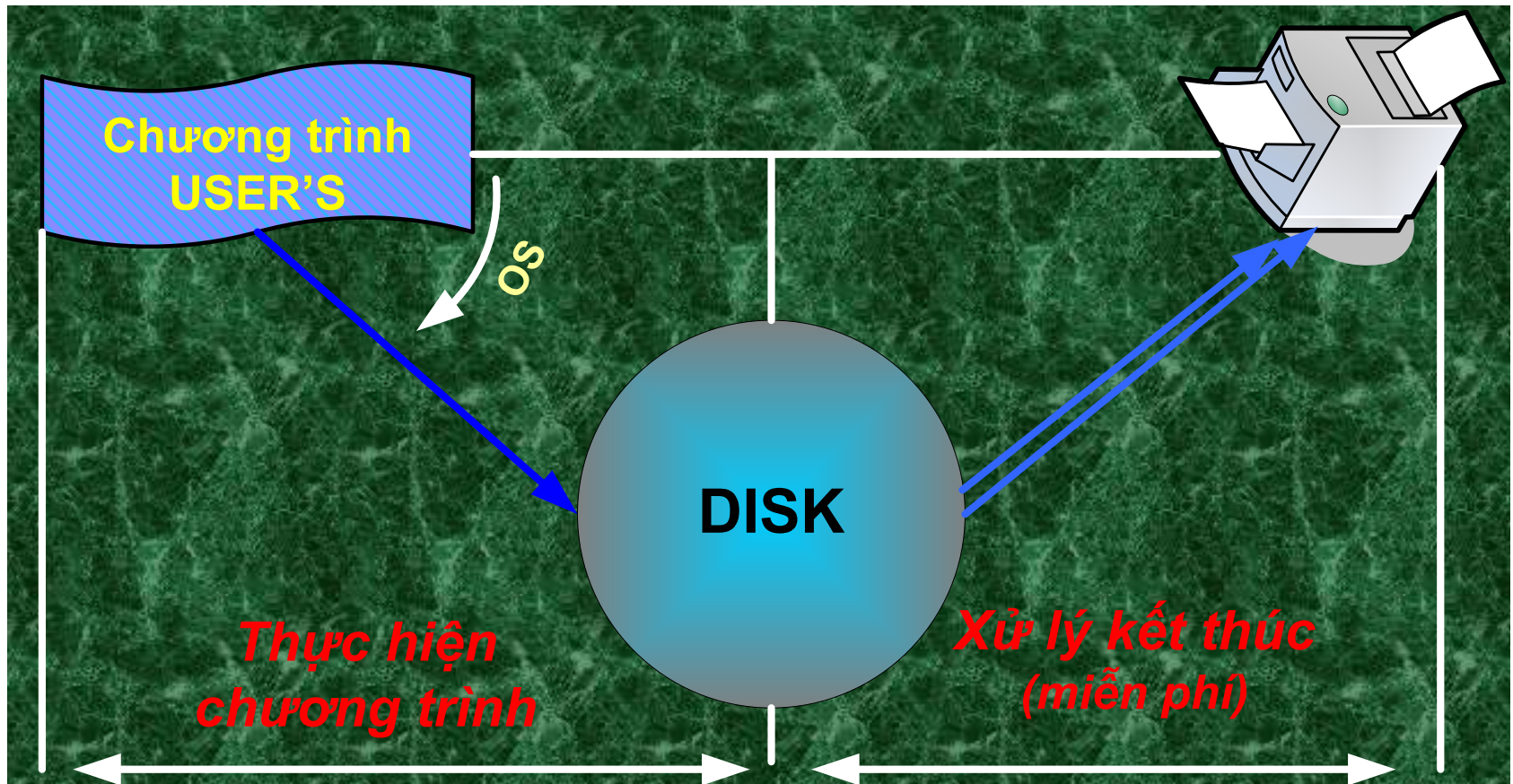


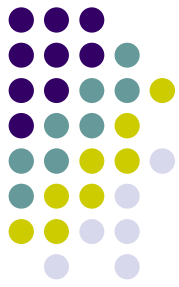
### 3 - SPOOL

- SPOOL – Simultaneous Peripheral Operations On-Line,
- Không can thiệp vào CT người dùng,
- Hai giai đoạn:
  - Thực hiện: thay thế thiết bị ngoại vi bằng thiết bị trung gian (Đĩa cứng),
  - Xử lý kết thúc:
    - Sau khi kết thúc việc thực hiện CT,
    - Đưa thông tin ra thiết bị yêu cầu.
- Chú ý: Đặc trưng của thiết bị trung gian.

# SPOOL

- Đảm bảo song song giữa xử lý một CT với trao đổi vào ra của CT khác.

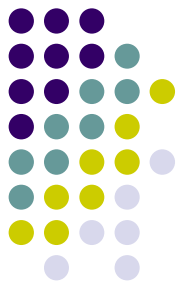




# SPOOL

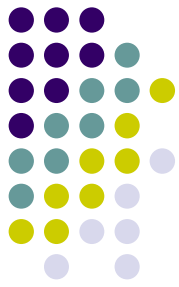
- Giải phóng hệ thống khỏi sự ràng buộc về số lượng thiết bị,
- Khai thác thiết bị ngoại vi tối ưu,
- Kỹ thuật lập trình hiệu quả.
- Hệ thống cung cấp các phương tiện để người dùng tạo SPOOL,
- Ai tạo SPOOL – người đó xử lý kết thúc.

# Tổ chức SPOOL

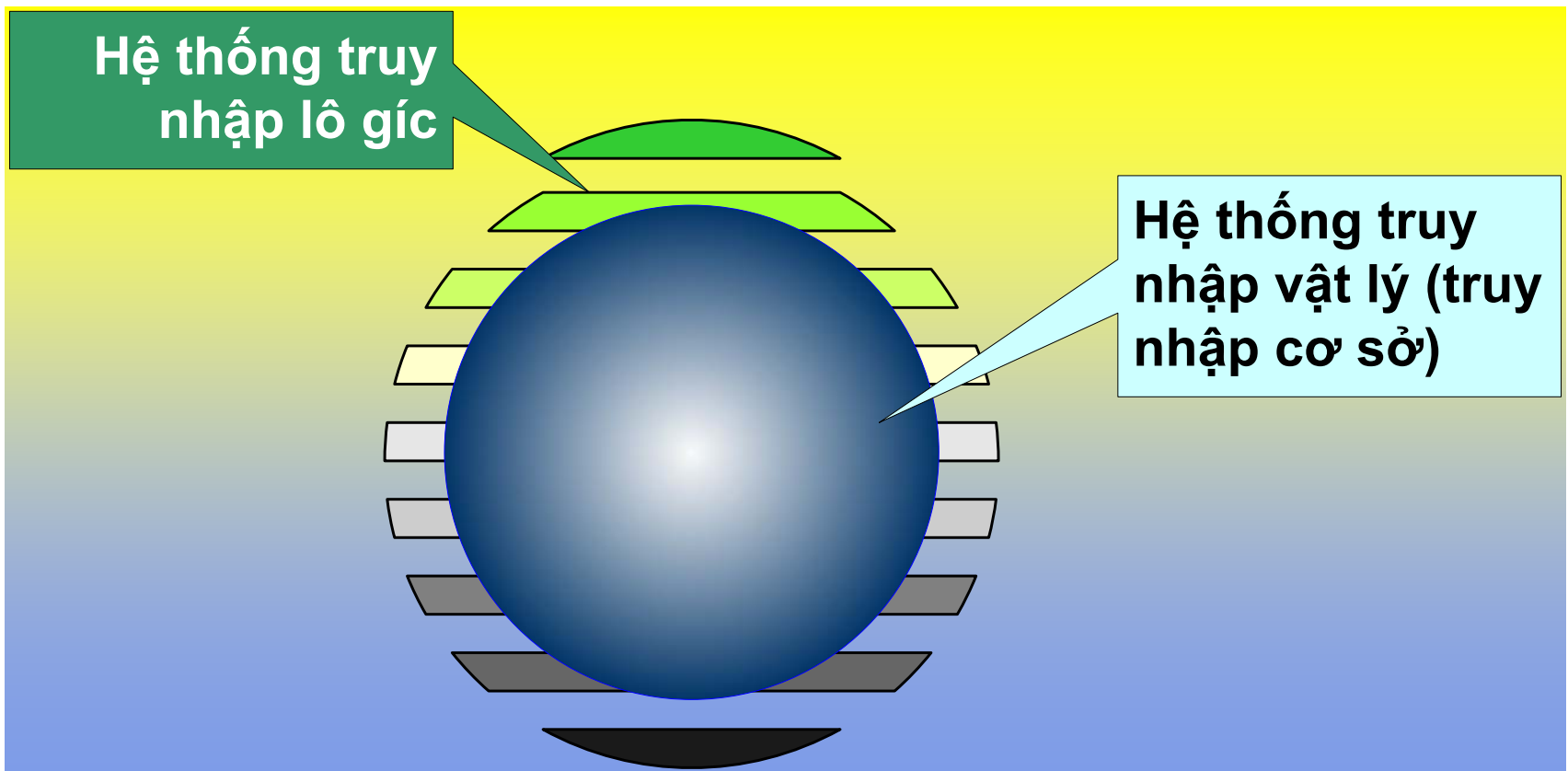


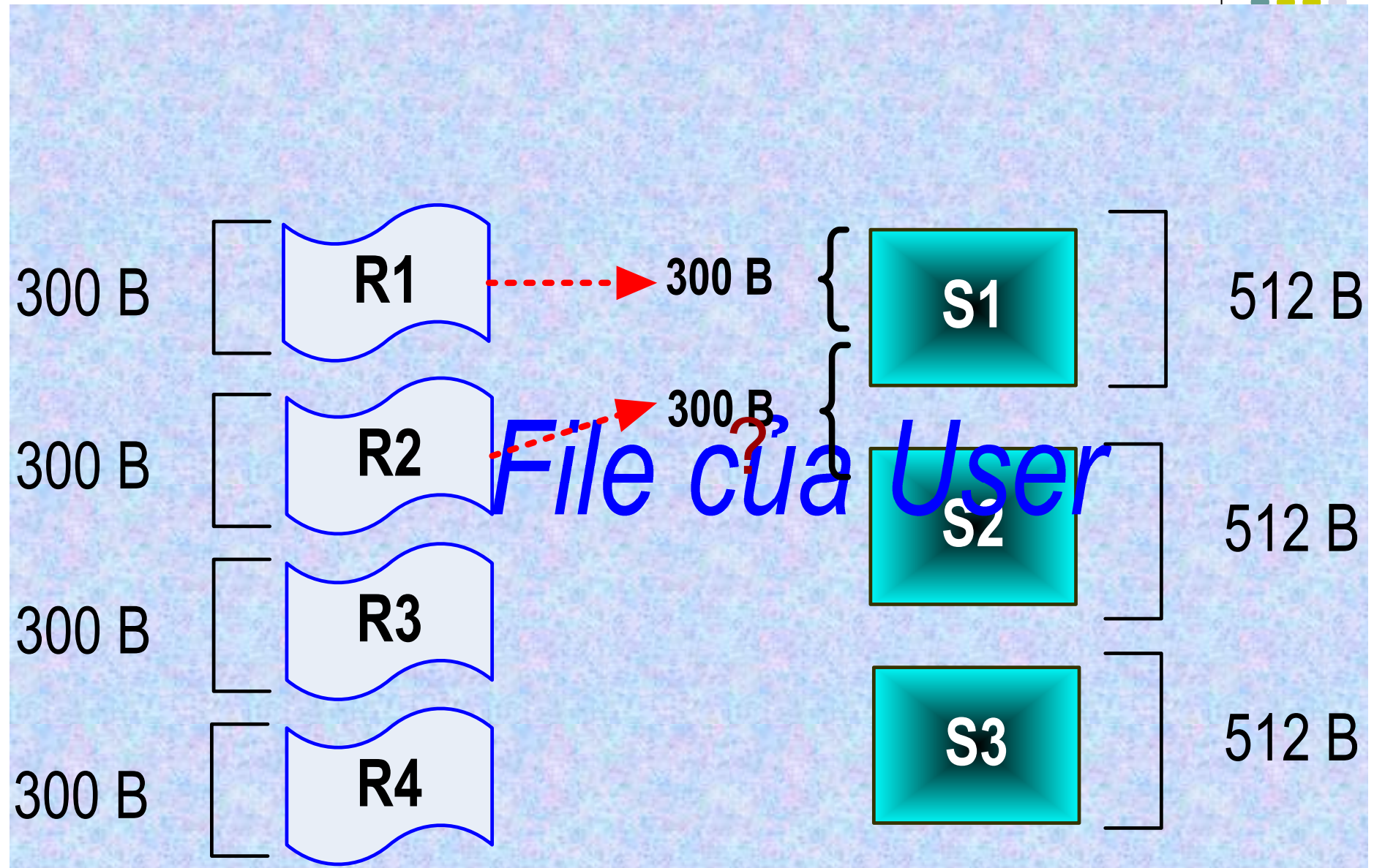
- **Giai đoạn thực hiện:** với mỗi phép trao đổi vào ra hệ thống tạo 2 CT kênh:
  - CT kênh I – theo thiết bị yêu cầu,
  - CT kênh II – phục vụ ghi CT kênh I ra thiết bị trung gian,
- **Xử lý kết thúc:** Đọc CT kênh đã lưu và chuyển giao cho kênh.
- Như vậy, mỗi thiết bị sử dụng → file CT kênh.

# 4 – HỆ THỐNG QUẢN LÝ FILES

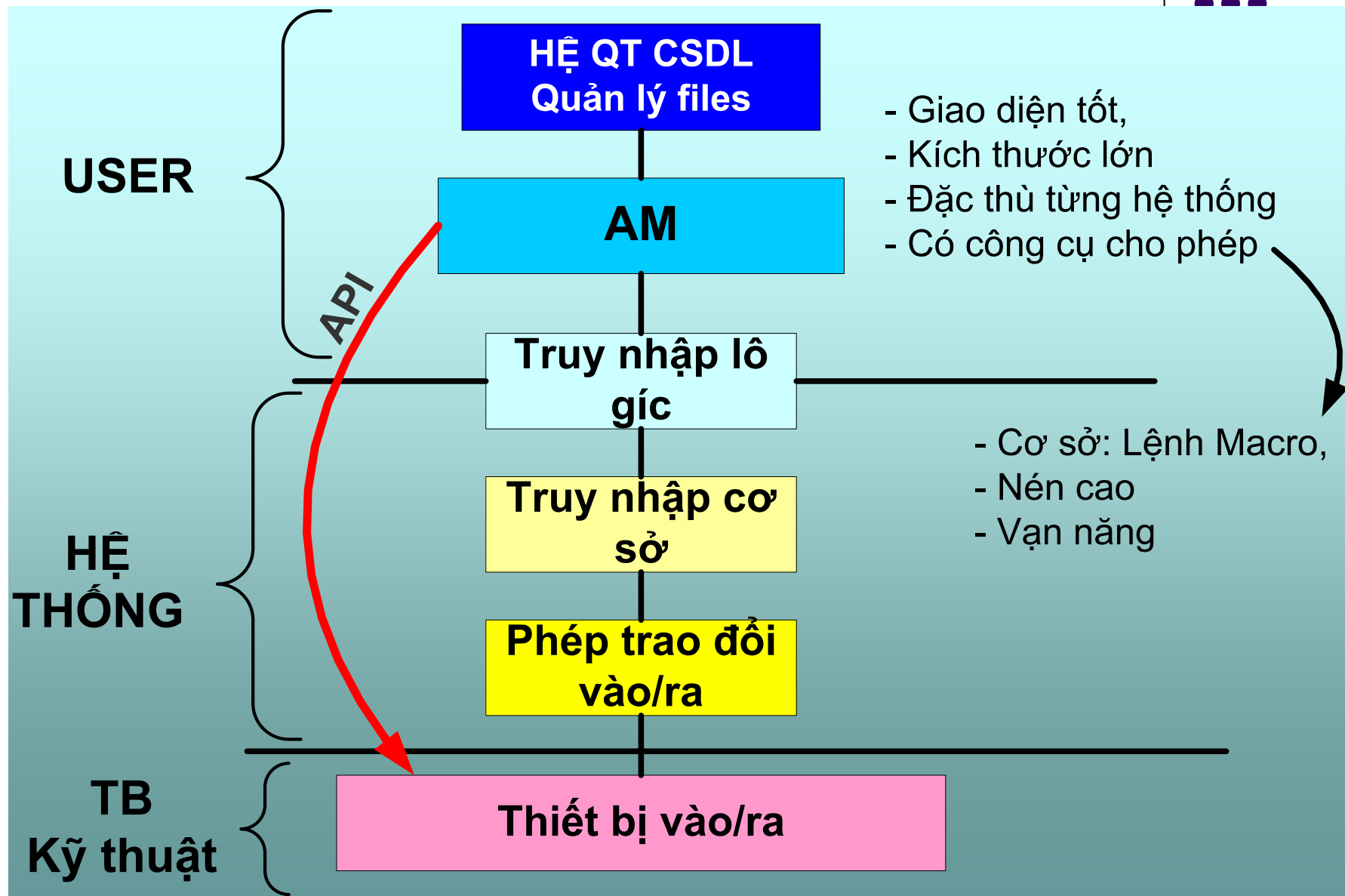


- ∃ CSDL quản lý files,
- Hệ thống quản lý files - Hệ QT CSDL.

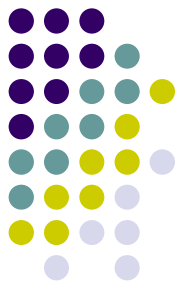






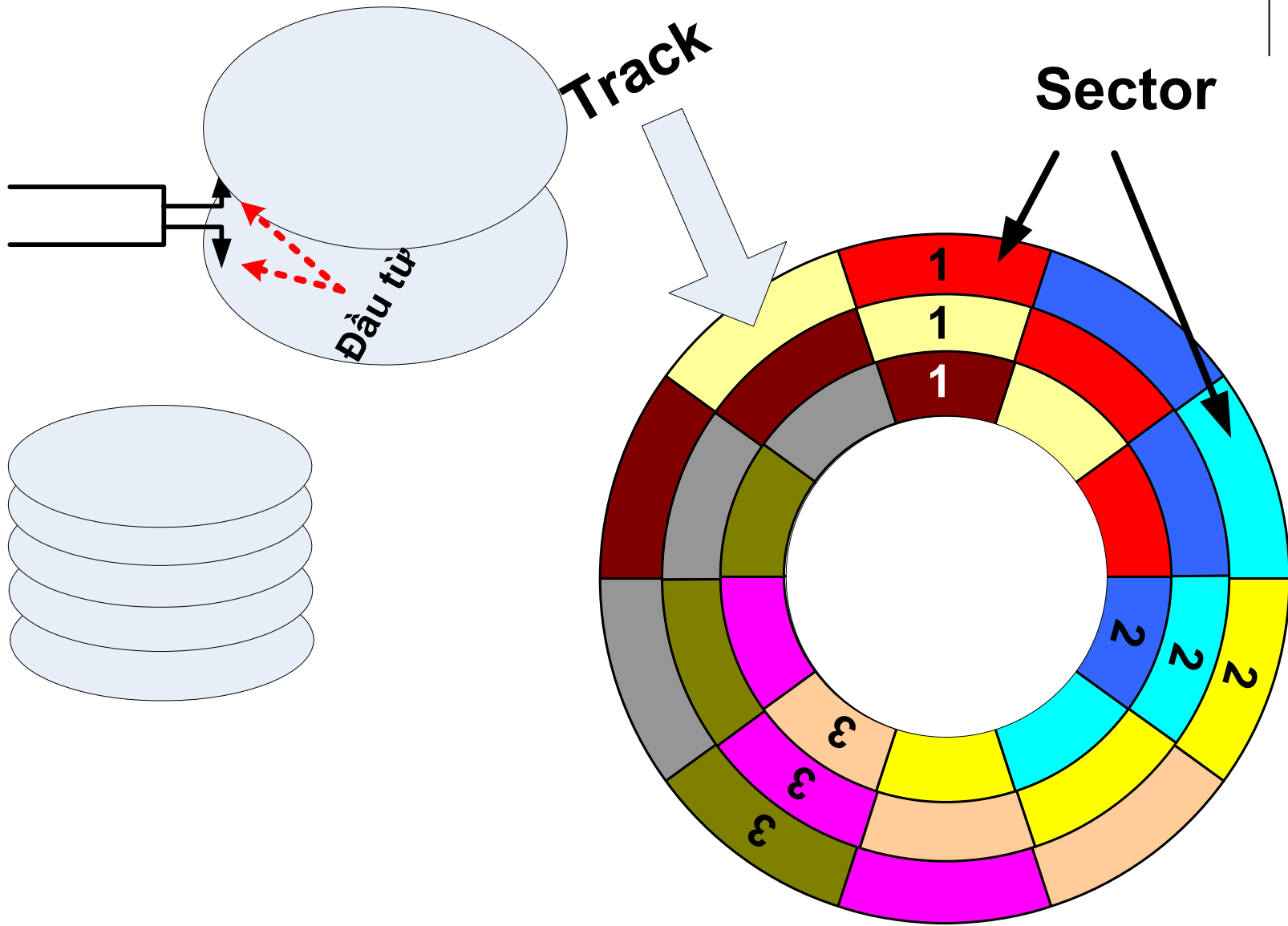


# QUẢN LÝ FILE TRONG WINDOWS

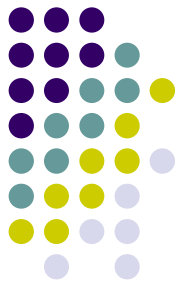


- Mục đích:
  - Minh họa nguyên lý bảng tham số điều khiển,
  - Tính kế thừa và thích nghi,
  - Cơ chế bảo vệ,
  - Cách thể hiện một số chế độ quản lý bộ nhớ (chương tiếp theo).

# TỔ CHỨC THÔNG TIN TRÊN ĐĨA TỪ

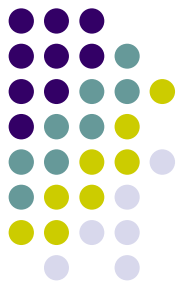


# CÁC KHÁI NIỆM CƠ BẢN

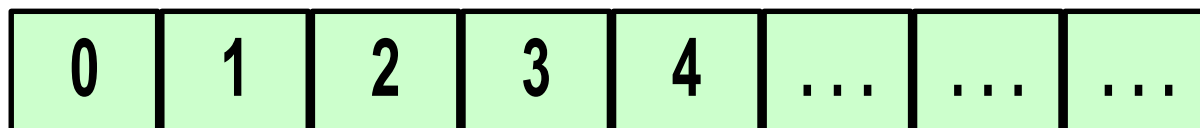


- Sector:
  - Đánh số từ 1,
  - Số Sector/track – Constant,
  - Hệ số đan xen (Interleave) – nguyên tố cùng nhau với số sector/track,
  - Kích thước 1 sector:
    - 128B
    - 256B
    - 512B
    - 1KB

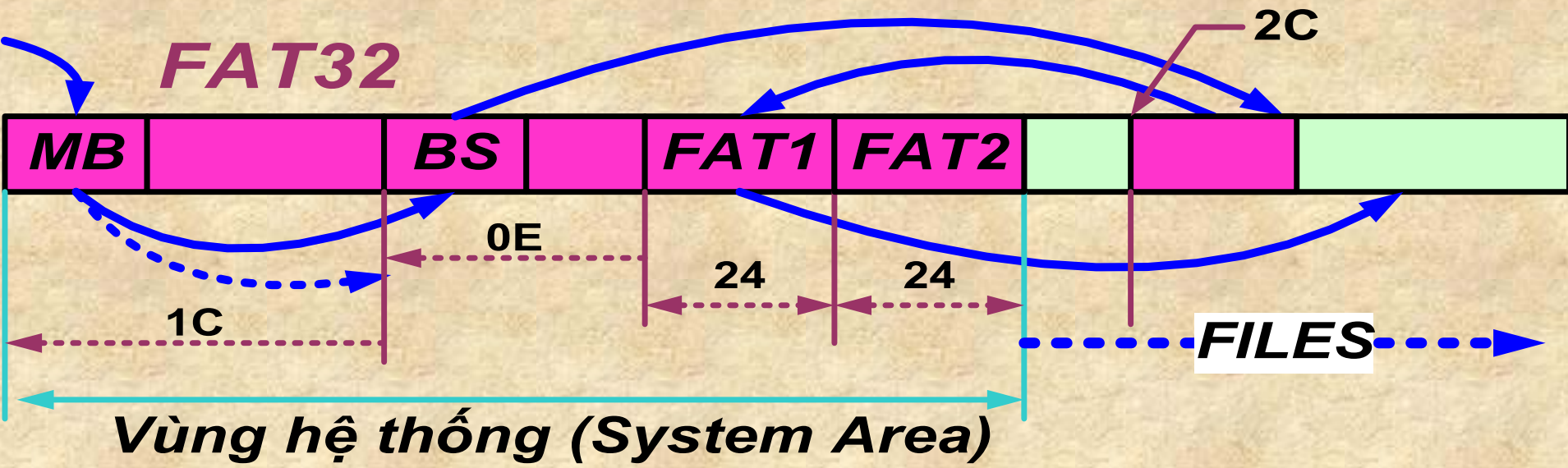
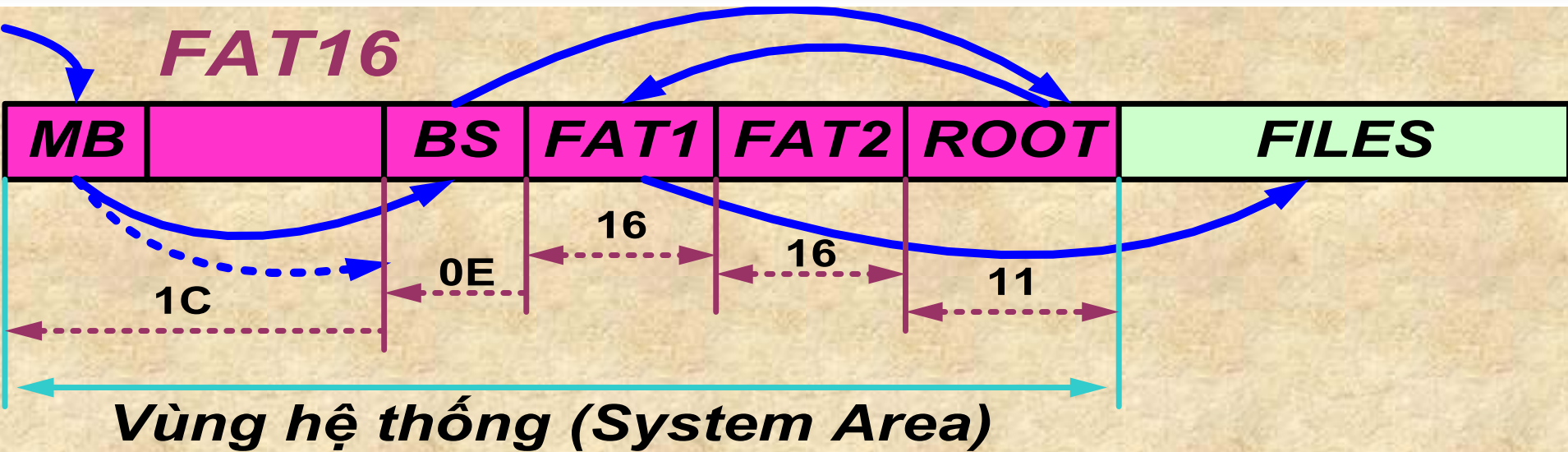
# CÁC KHÁI NIỆM CƠ BẢN



- Cylinder: 0, 1, 2, . . .
- Đầu từ (Header): 0, 1, 2, . . .
- Cluster:
  - Nhóm sectors liên tiếp lôgic,
  - Đơn vị phân phối cho người dùng,
  - Đánh số: 2, 3, 4, . . .
  - Kích thước: 1, 2, 4, 8, 16, 32, 64 (S),
- Địa chỉ vật lý: (H, S, Cyl),
- Địa chỉ tuyệt đối: 0, 1, 2, . . .



# CẤU TRÚC THÔNG TIN TRÊN ĐĨA TỪ



# BOOT SECTOR



***Tham số***

**Đặc thù**

***Chương trình  
mồi  
(Boot Strap  
Loader)***

**Như nhau với mọi  
đĩa cho mỗi loại  
OS**

**55AA**

**Chữ ký  
(Signature)**

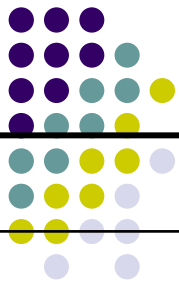
# BOOT SECTOR



| Stt | Offs            | L (Byte) | Ý Nghĩa                                  |
|-----|-----------------|----------|--|
| 1   | 0               | 3        | Lệnh JMP (EB xx 90)                      |
| 2   | 3               | 8        | Tên hệ thống Format đĩa                  |
| 3   | B               | 2        | Kích thước Sector                        |
| 4   | D               | 1        | Sec/Cluster                              |
| 5   | E               | 2        | Địa chỉ tuyệt đối FAT1 trong đĩa lô gíc  |
| 6   | 10 <sub>H</sub> | 1        | Số lượng bảng FAT                        |
| 7   | 11              | 2        | FAT16: Số phần tử ∈ ROOT<br>FAT32: 00 00 |
| 8   | 13              | 2        | Σ sect/Disk (<32MB) hoặc 00 00           |



# BOOT SECTOR

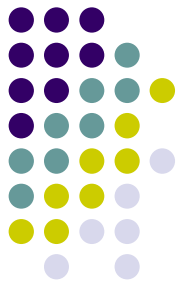


|    |    |   |  |
|----|----|---|--|
| 9  | 15 | 1 | Kiểu đĩa từ (F8 – HD, F0 – 1.44MB)         |
| 10 | 16 | 2 | FAT16: $\Sigma$ Sec/FAT<br>FAT32: 00 00    |
| 11 | 18 | 2 | Sec/ Track                                 |
| 12 | 1A | 2 | số đầu từ                                  |
| 13 | 1C | 4 | Địa chỉ tuyệt BS trong đĩa vật lý          |
| 14 | 20 | 4 | $\Sigma$ Sec / Disk ( $\geq 32$ MB) hoặc 0 |
| 15 | 24 | 4 | $\Sigma$ Sec / FAT                         |
| 16 | 28 | 2 | Flags                                      |
| 17 | 2A | 2 | Version                                    |
| 18 | 2C | 4 | Địa chỉ ROOT (Cluster)                     |



|    |    |           |                          |
|----|----|-----------|--------------------------|
| 19 | 30 | 2         | Inf                      |
| 20 | 32 | 2         | Địa chỉ lưu BS           |
| 21 | 34 | $12_{10}$ | Dự trữ (00...00)         |
| 22 | 40 | 1         | Địa chỉ ổ đĩa ( 80 – C:) |
| 23 | 41 | 1         | 00                       |
| 24 | 42 | 1         | 29 – BIOS mở rộng        |
| 25 | 43 | 4         | Serial Number            |
| 26 | 47 | $11_{10}$ | Volume Name              |
| 27 | 52 | 8         | FAT32                    |

# Boot Sector FAT 16



|    |    |                  |                          |
|----|----|------------------|--------------------------|
| 15 | 24 | 1                | Địa chỉ ổ đĩa ( 80 – C:) |
| 16 | 25 | 1                | 00                       |
| 17 | 26 | 1                | 29 – BIOS mở rộng        |
| 18 | 27 | 4                | Serial Number            |
| 19 | 2B | 11 <sub>10</sub> | Volume Name              |
| 20 | 36 | 8                | FAT16                    |

# Ví dụ



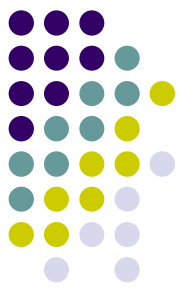
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 58 | 90 | 4D | 53 | 57 | 49 | 4E | 34 | 2E | 31 | 00 | 02 | 08 | 2D | 00 |
| 02 | 00 | 00 | 00 | 00 | F8 | 00 | 00 | 3F | 00 | 40 | 00 | 3F | 00 | 00 | 00 |
| 41 | 0C | 34 | 00 | 03 | 0D | 00 | 00 | 00 | 00 | 00 | 00 | 02 | 00 | 00 | 00 |
| 01 | 00 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 80 | 00 | 29 | D1 | 09 | 47 | 32 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| 20 | 20 | 46 | 41 | 54 | 33 | 32 | 20 | 20 | 20 | FA | 33 | C9 | 8E | 41 | BC |

# THƯ MỤC



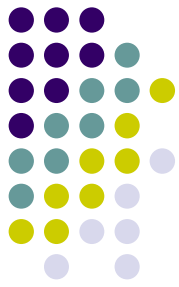
- Đóng vai trò mục lục tra cứu, tìm kiếm,
- Mọi hệ thống đều phải có với những tên khác nhau (Catalog, Directory, Folder, . . .),
- Bao gồm: Thư mục gốc (ROOT) + Thư mục con,
- Các hệ thống của Microsoft và OS IBM – cấu trúc cây,
- UNIX - cấu trúc phân cấp,
- Thư mục = {Phần tử}, mỗi phần tử:  $32_{10}$  B
- Phần tử  $\leftrightarrow$  file,
- Thư mục con và ROOT: File có cấu trúc.

# Cấu trúc phần tử thư mục tên ngắn (Phần tử 8.3)



| <i>Stt</i> | <i>Offs</i>     | <i>L</i> | <i>Ý nghĩa</i>                 |
|------------|-----------------|----------|--------------------------------|
| 1          | 0               | 8        | Tên (Name)                     |
| 2          | 8               | 3        | Phần mở rộng (Extention)       |
| 3          | B               | 1        | Thuộc tính (Attribute)         |
| 4          | C               | 2        | Thời điểm tạo file             |
| 5          | E               | 2        | Ngày tạo file                  |
| 6          | 10 <sub>H</sub> | 2        | Ngày truy nhập gần nhất        |
| 7          | 12              | 1        | 00 (Cho NT)                    |
| 8          | 13              | 1        | Số 0.1" của thời điểm tạo file |

## Phần tử 8.3



|    |    |   |                                    |
|----|----|---|------------------------------------|
| 9  | 14 | 2 | 2 bytes cao của cluster xuất phát  |
| 10 | 16 | 2 | Thời điểm cập nhật cuối cùng       |
| 11 | 18 | 2 | Ngày cập nhật cuối cùng            |
| 12 | 1A | 2 | 2 bytes thấp của cluster xuất phát |
| 13 | 1C | 4 | Kích thước file (Byte)             |

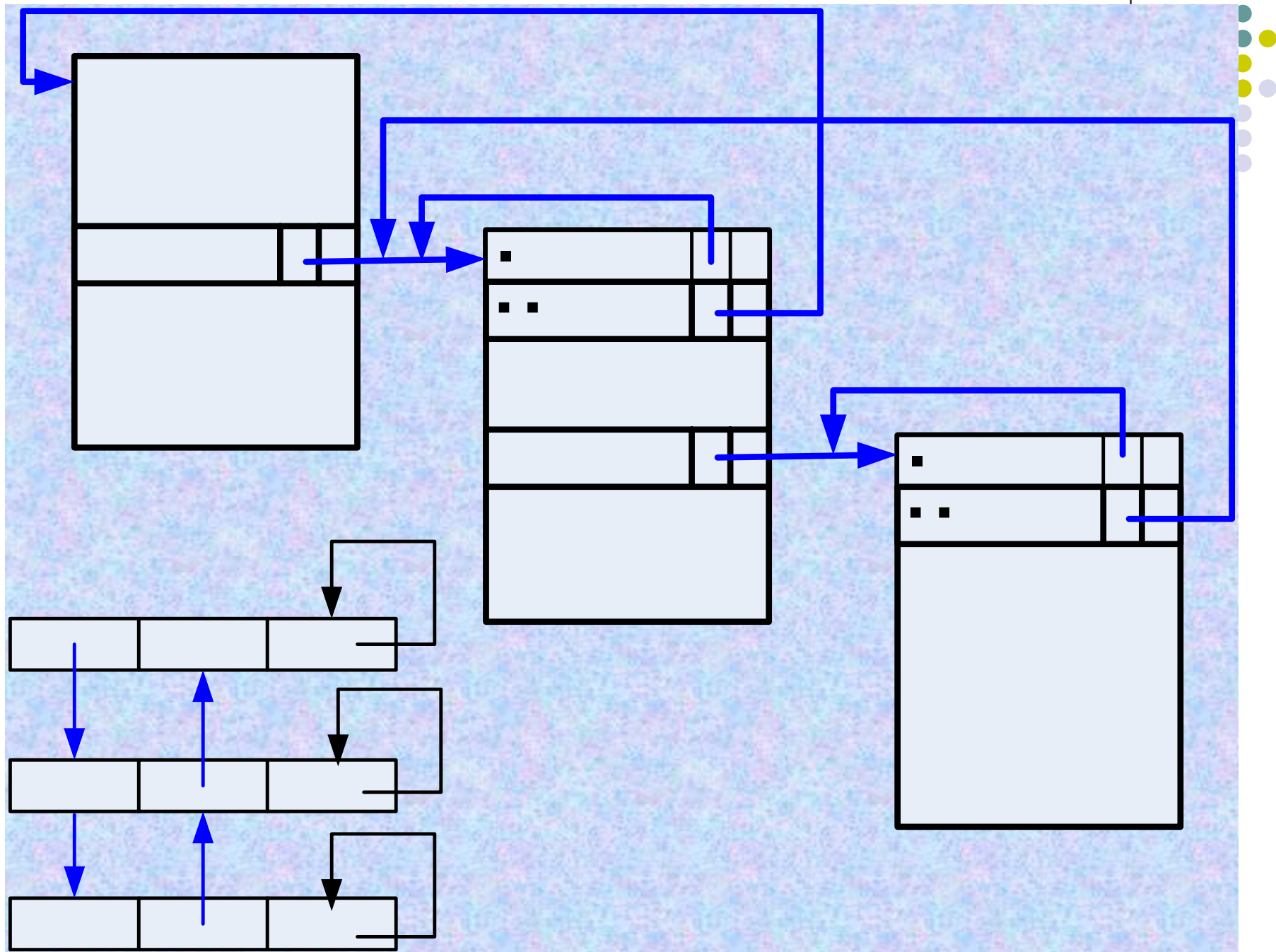
# Phần tử 8.3

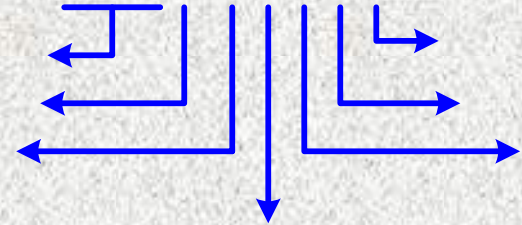


- Byte số 0: Vai trò đặc biệt.
- 00 – Chưa sử dụng, phần tử chưa sử dụng đầu tiên - dấu hiệu kết thúc thư mục,
- E5 – ( $\sigma$ ) Đã bị xoá,
- 05 – Tên bắt đầu bằng ký tự  $\sigma$ ,
- 2E 20 (. ) – Phần tử thứ I của thư mục con,
- 2E 2E (..) – Phần tử thứ II của thư mục con

|    |  |
|----|--|
| xx |  |
| xx |  |
| xx |  |
| xx |  |
| 00 |  |
|    |  |
|    |  |
|    |  |







## Time



## Date



1998 – 18

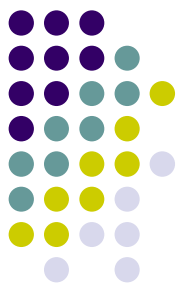
1999 – 19

2000 – 20

2001 – 21

Dự trữ thời gian:  
 $1980 + 2^7 - 1 = 2107$

# Tên dài



- Không quá 255 ký tự,
- Unicode,
- Hệ thống phân biệt theo 66 ký tự đầu tiên,
- Lưu trữ theo cách đưa vào,
- Nhận dạng: Đưa về chữ hoa.

## ***Lưu trữ tên dài***

***Phần tử tên dài  $n$***

***Phần tử tên dài  $n-1$***

⋮

***Phần tử tên dài 1***

***Phần tử 8.3***

# Cấu trúc phần tử tên dài



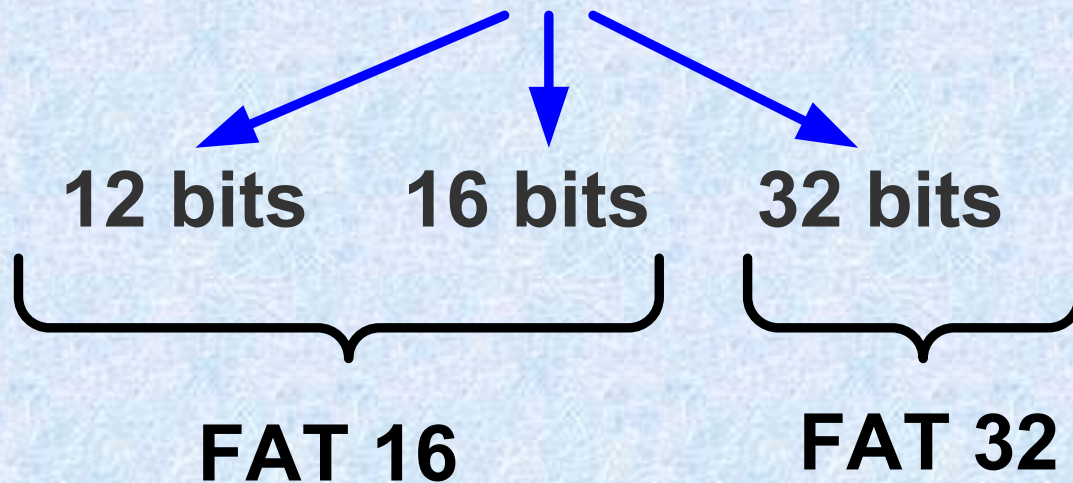
| Stt | Offs | L                | Ý nghĩa                                 |
|-----|------|------------------|---|
| 1   | 0    | 1                | Số thứ tự i (64+i)                      |
| 2   | 1    | 10 <sub>10</sub> | 5 ký tự C <sub>1</sub> – C <sub>5</sub> |
| 3   | B    | 1                | Thuộc tính (00001111 <sub>B</sub> )     |
| 4   | C    | 1                | 00 – dấu hiệu phần tử tên dài           |
| 5   | D    | 1                | Σ <sub>K</sub> phần tử 8.3              |
| 6   | E    | 12 <sub>10</sub> | C <sub>6</sub> – C <sub>11</sub>        |
| 7   | 1A   | 2                | 00 00                                   |
| 8   | 1C   | 4                | C <sub>12</sub> – C <sub>13</sub>       |

# File Allocation Table (FAT)

- Chức năng:
  - Quản lý bộ nhớ phân phối cho từng file,
  - Quản lý bộ nhớ tự do trên đĩa,
  - Quản lý bộ nhớ kém chất lượng.
- $FAT = \{phần\ tử\}$
- Phần tử:
  - Đánh số: 0, 1, 2, . . .
  - Từ phần tử số 2: phần tử  $\leftrightarrow$  Cluster



# Phần tử



Phần tử 0:    Fxx

FFF8

FFFFFFFF8

Phần tử 1:    FFF

xx11FF8

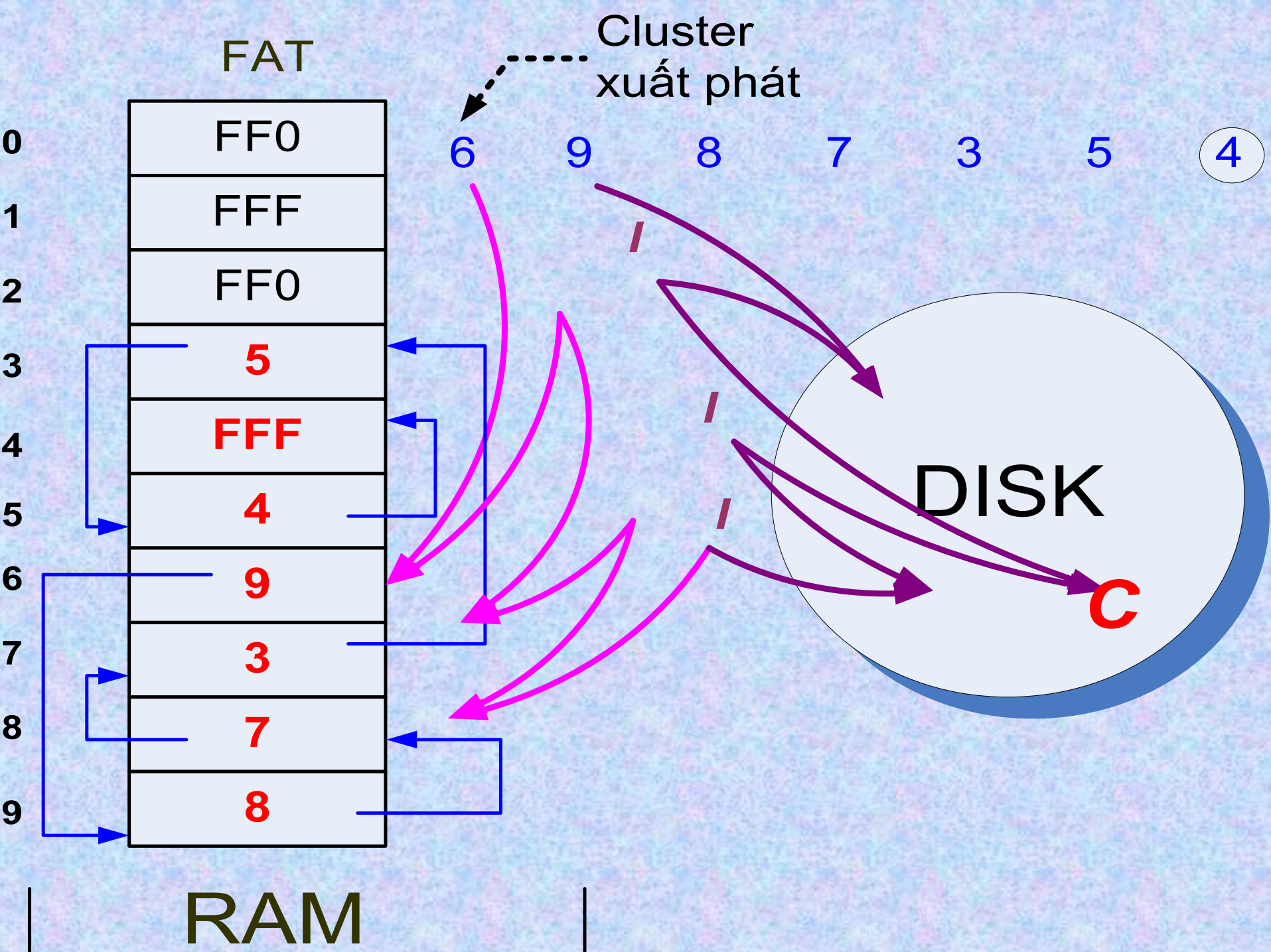
0xx11FFFFFF8



# FAT

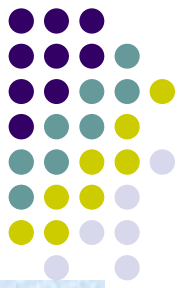


- Bit Shutdown = 1 – Ra khỏi hệ thống đúng cách
- Bit Diskerror = 1 – không có lỗi truy nhập đĩa ở lần truy nhập cuối cùng.
- Từ phần tử 2 trở đi:
- Giá trị 0 – Cluster tự do,
- FF7 (FFF7, 0FFFFFFF7) – Bad cluster,
- Các giá trị khác – đã phân phối,
- Các phần tử tương ứng những Clusters của một file - tạo thành một danh sách móc nối,
- EOC (End of Cluster Chain) – FFF (FFFF, FFFFFFFF).





# MASTER BOOT



*Chương trình  
nhận biết cấu  
trúc  
(Master Boot  
Record - MBR)*

Như nhau với mọi  
đĩa cho mỗi loại  
OS

FDISK /MBR

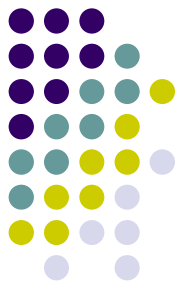
Đặc thù

*Bảng phân  
vùng*

Chữ ký  
(Signature)

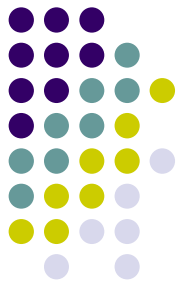
55AA

# MASTER BOOT

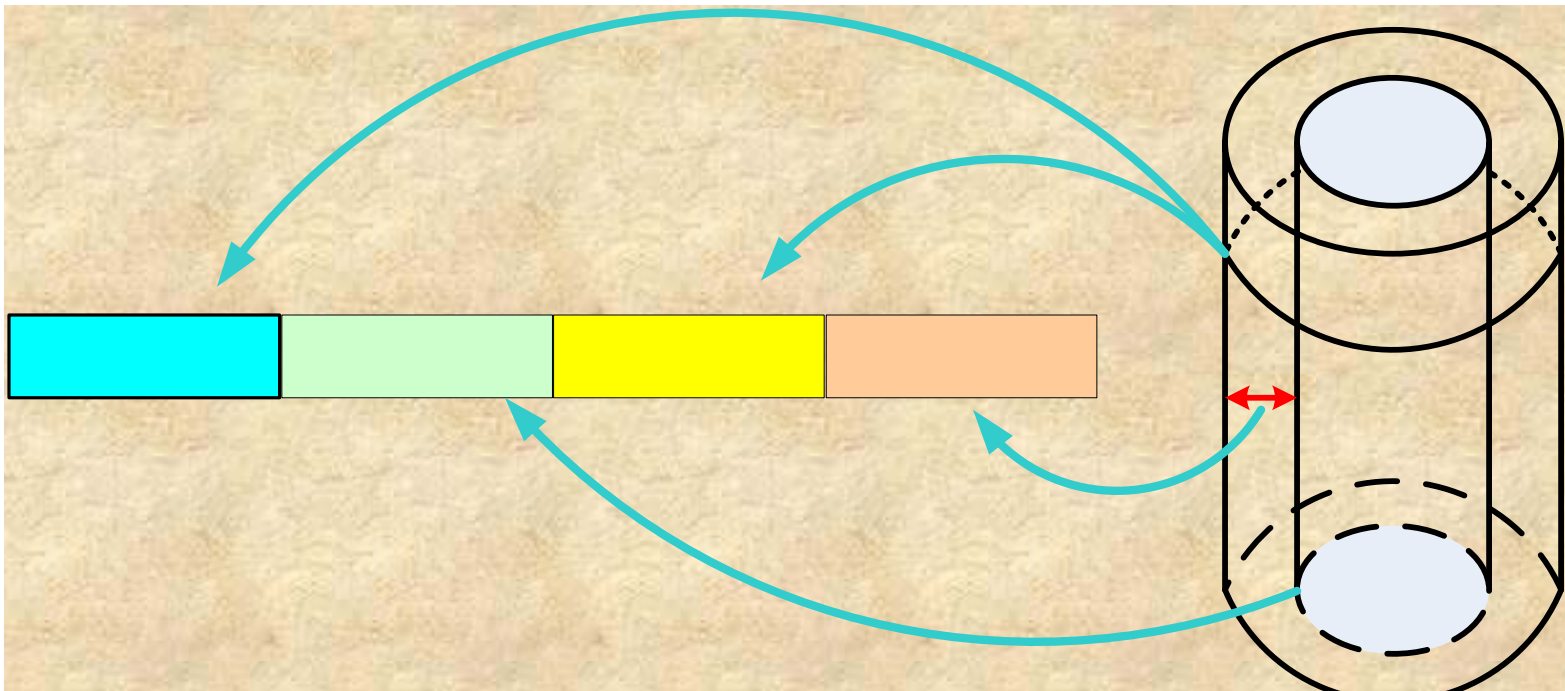


- Nguyên tắc khai thác HD:
  - Chia HD thành các phần, mỗi phần có kích thước cố định,
  - Mỗi phần sử dụng như một đĩa từ độc lập: Đĩa lô gic ( Logical Volume).
- OS cho phép tạo các đĩa kích thước động trong mỗi đĩa lô gic.

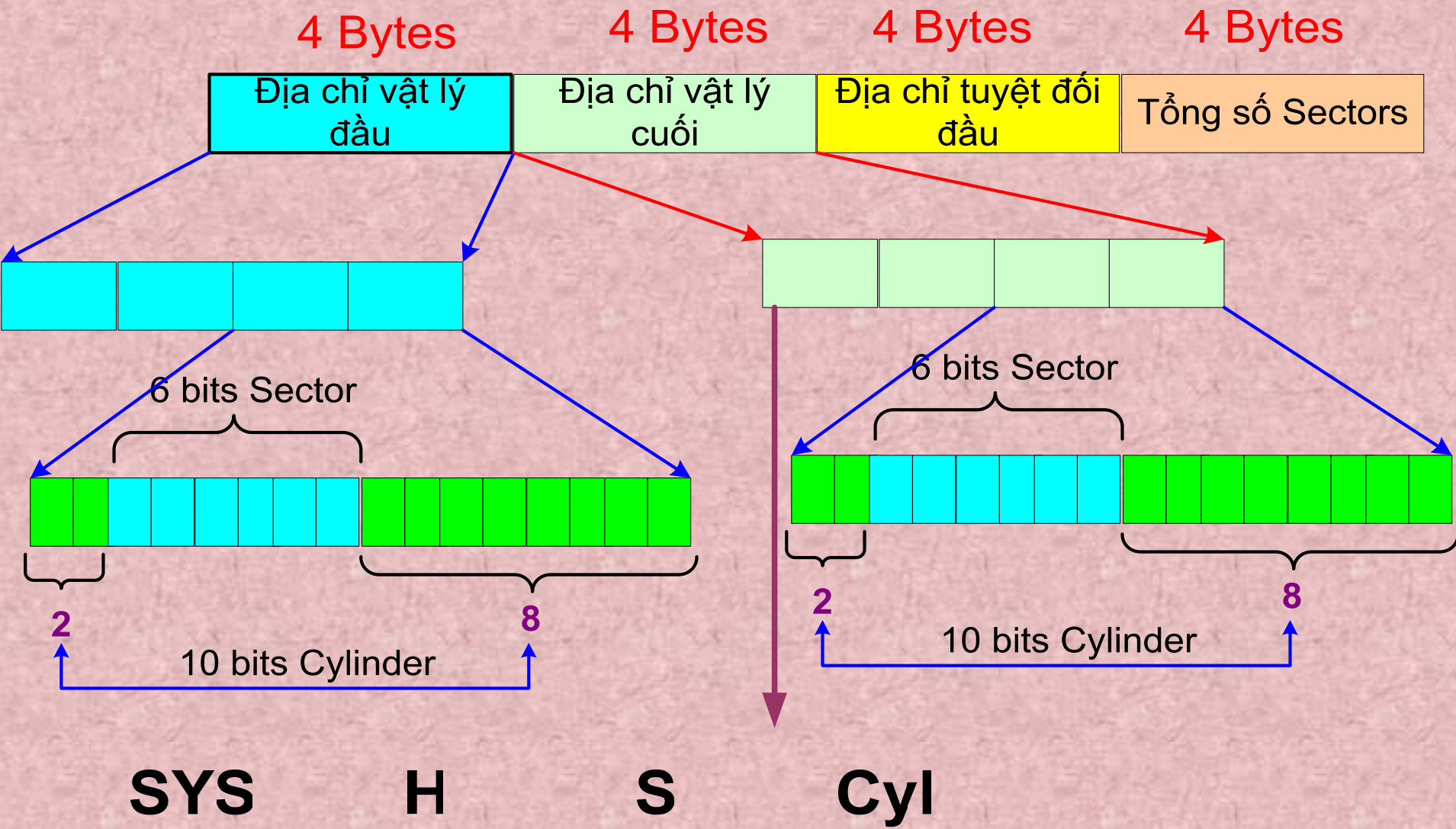
# Cấu trúc bảng phân vùng



- Bảng phân vùng bắt đầu từ địa chỉ  $1BE_H$ ,
- Bảng phân vùng = {4 phần tử},
- Mỗi phần tử sử dụng  $\leftrightarrow$  *Đĩa lô gíc*,
- Tồn tại cơ chế cho phép tạo *nhiều hơn 4* đĩa lô gíc trên một đĩa vật lý.



# Cấu trúc phần tử bảng phân vùng



# Bảng phân vùng



80 01 01 00 0B 3F FF 4D 3F 00 00 00 41 0C 34 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

55AA

|          |          |          |          |
|----------|----------|----------|----------|
| 80010100 | 010511BF | 11000000 | 6F4C0000 |
| 000001C0 | 5105511F | 804C0000 | 40260000 |
| 00004120 | 510551DF | C0720000 | 804C0000 |
| 000041E0 | 5105D132 | 40BF0000 | 12870000 |


55AA

# Truy nhập Boot Sector



- Dùng các hàm API,
- Chương trình đọc và đưa ra màn hình nội dung BS của đĩa mềm A: (Hexa và ASCII):

```
Program R_BS_A;  
Uses Crt, Dos;  
Const s16: string[16]='0123456789ABCDEF' ;  
Var  B: array[0..511] of char;  
      reg: registers;  
      i,j,k: integer;  
      c: char;  
  
BEGIN  
  clrscr;  
  fillchar(b,sizeof(b),0) ;  
  writeln('Cho dia vao o A: va bam phim bat ky. ');  
  c:=readkey;
```



```
I := 0;
```

```
Repeat
```

```
  with reg do
```

```
    begin
```

```
      dl := 0; { 0 -> A:, 128 -> C: }
```

```
      dh := 0; {Đầu từ}
```

```
      cl := 1; {Sector}
```

```
      ch := 0; {Cylinder}
```

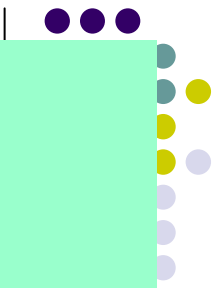
```
      al := 1; {So Sectors can doc}
```

```
      ah := 2; {2 -> Read; 3 -> Write;. . .}
```

```
      es := seg(b) ;
```

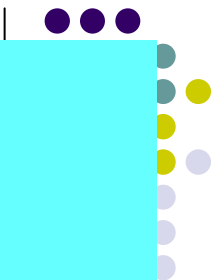
```
      bx := ofs(b)
```

```
    end;
```

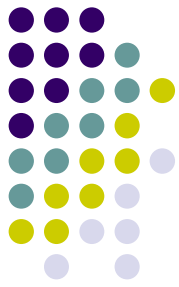


```
intr($13,reg) ;  
    inc(i)  
Until i = 2;  
for i := 0 to 511 do  
    begin  
        j := b[i] shr 4 + 1;  
        k := b[i] and $0F + 1;  
        write(s16[j]:2, s16[k]);  
        if (i+1) mod 16 = 0 then  
            begin write(` `:5);  
                for j := i-15 to i do  
                    if (b[j] <32) or (b[j] = 255) then  
                        write(`.`)  
                    else write(chr(b[j]));
```





```
writeln;  
  if i = 255 then c:= readkey  
end  
end;  
Repeat  
Until keypressed  
END.
```

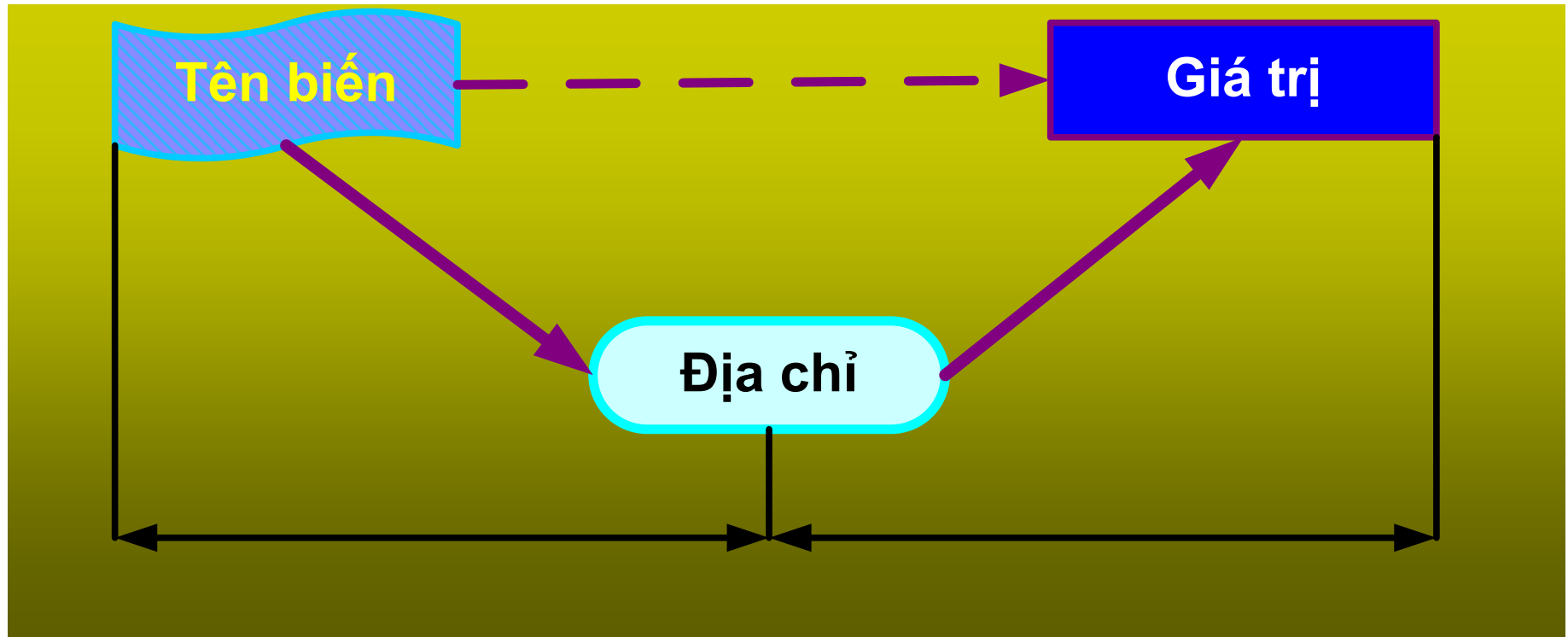


# III – QUẢN LÝ BỘ NHỚ

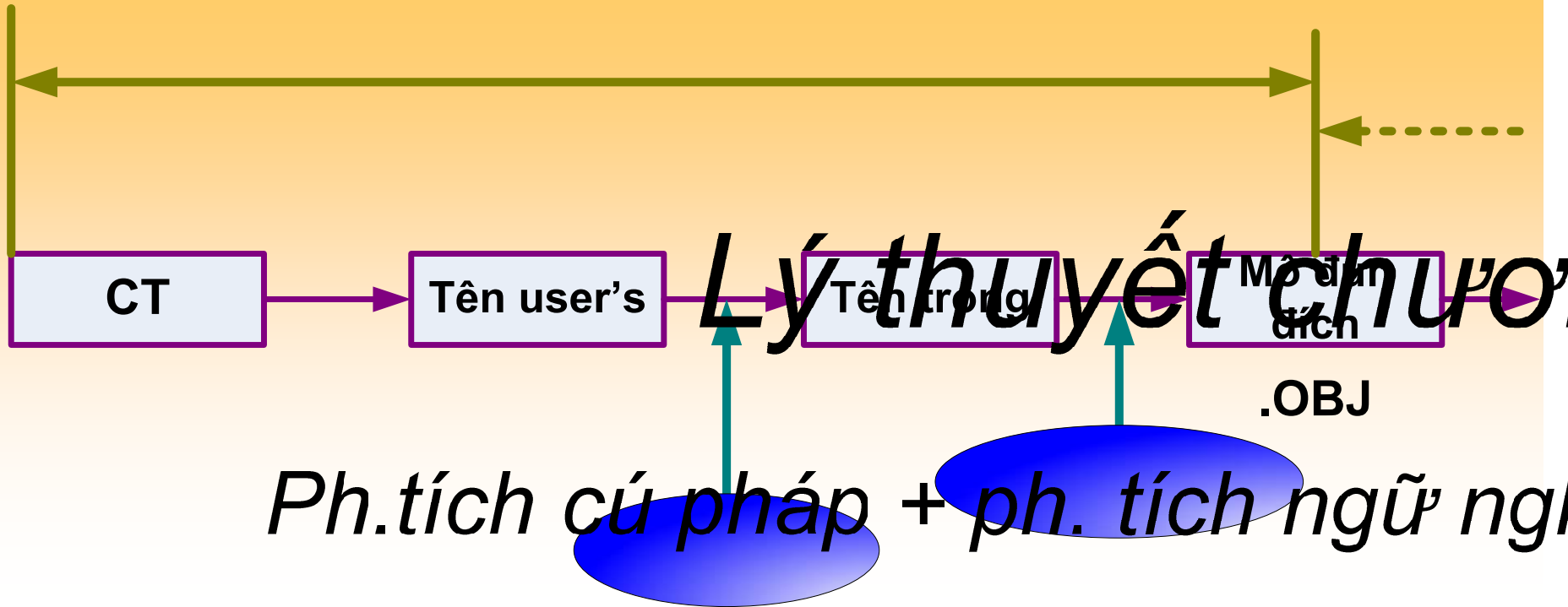
- Bộ nhớ tác động nhiều lên độ phức tạp của giải thuật,
- Phải giải quyết 2 v/đ trái ngược nhau:
- Tiết kiệm bộ nhớ,
- Tận dụng tối đa bộ nhớ cho phép.
- Phần lớn các chương trình: viết trên ngôn ngữ lập trình: Assembler, VB, JAVA, VC++, . . .
- Với người lập trình: CT và thực hiện CT là **ánh xạ từ tên sang giá trị**.

# QUẢN LÝ BỘ NHỚ

- Với hệ thống:

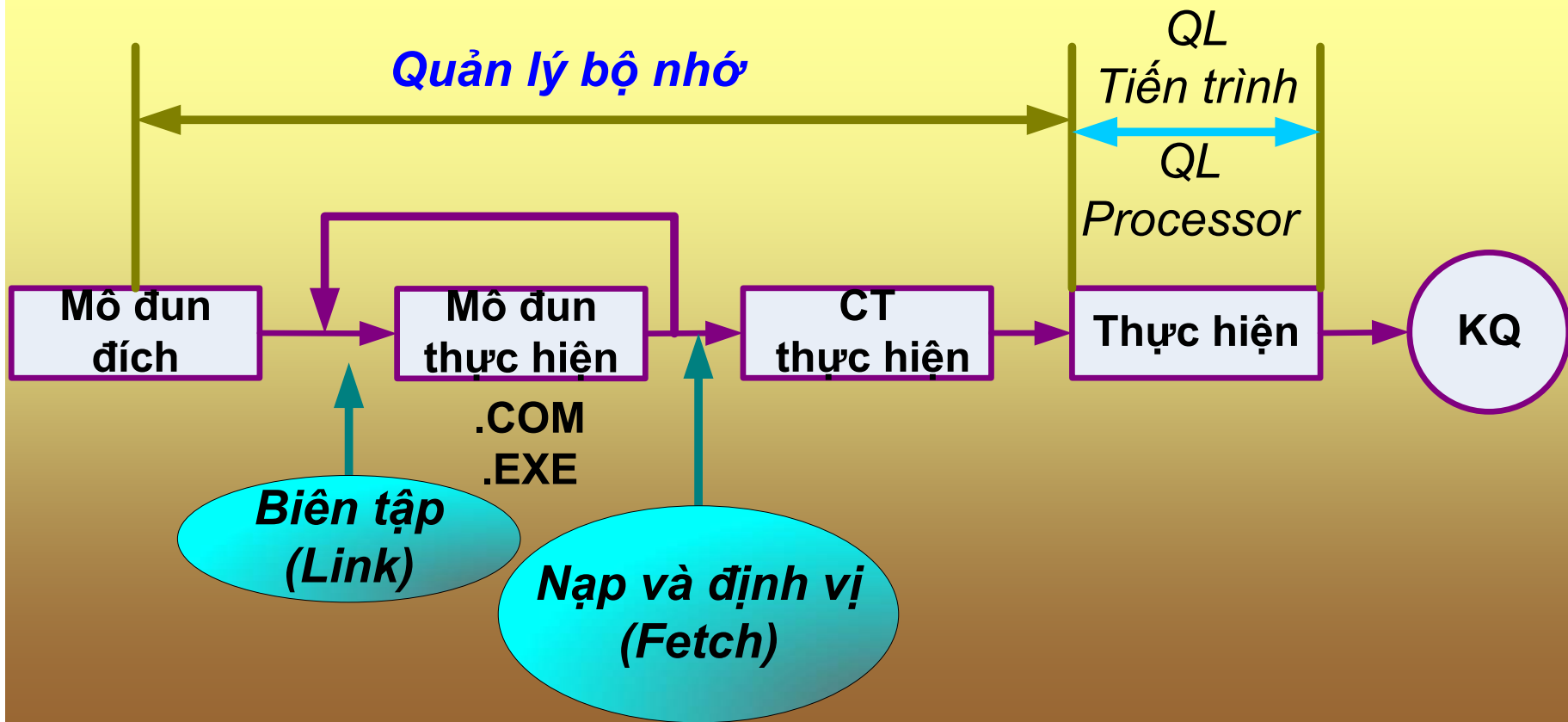


# 1 – CÁC BƯỚC XỬ LÝ CT



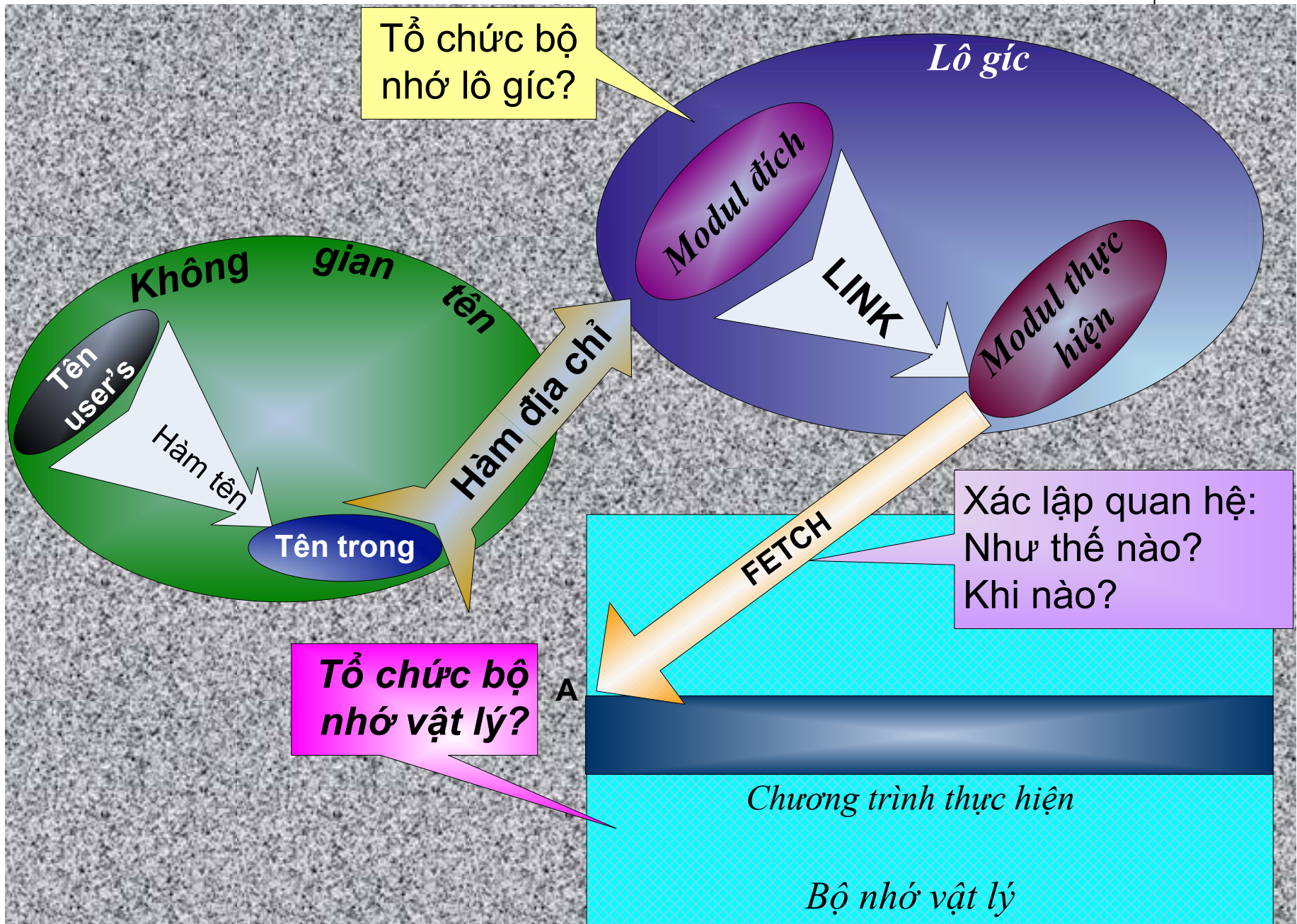
- I + J
- A + B
- A + I

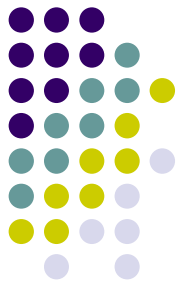
# CÁC BƯỚC XỬ LÝ CT



- Vai trò của Biên tập (Input/Output),
- Khái niệm bộ nhớ lô gíc.

# CÁC BƯỚC XỬ LÝ CT





## 2 – CẤU TRÚC CHƯƠNG TRÌNH

- Bộ nhớ lô gíc:
  - Không gắn với máy tính cụ thể,
  - Không giới hạn về kích thước,
  - Chỉ chứa 1 mô đun hoặc 1 CT,
  - Chỉ phục vụ lưu trữ, không thực hiện.
- Quản lý bộ nhớ lô gíc ~ tổ chức chương trình,
- Mỗi cách tổ chức CT  $\Leftrightarrow$  cấu trúc CT,
- Mọi cấu trúc: đều được sử dụng trong thực tế.

# CẤU TRÚC CHƯƠNG TRÌNH



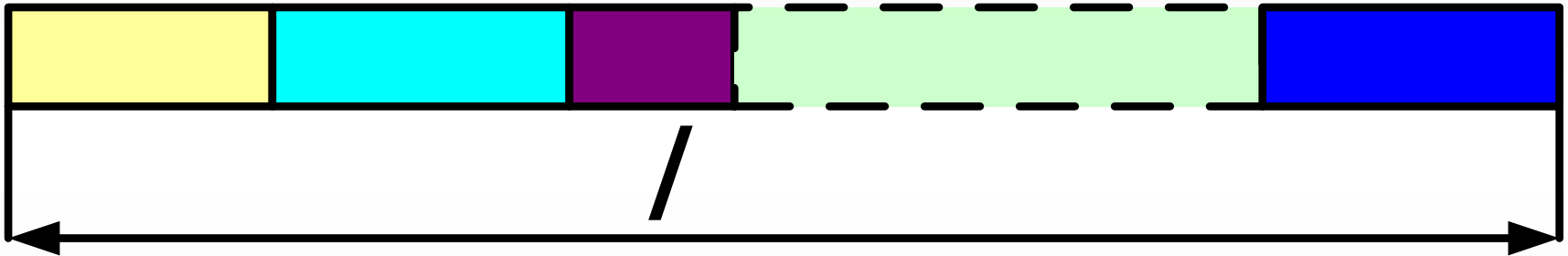
- Đặc trưng mô đun đích (**Object Modul**): chứa thông tin về các moduls khác liên quan (các móc nối) → kích thước lớn.
- Nhiệm vụ biên tập (**Linked**): Giải quyết các móc nối.
- Các loại cấu trúc chính:
  - Cấu trúc tuyến tính,
  - Cấu trúc động (Dynamic Structure),
  - Cấu trúc Overlay,
  - Cấu trúc mô đun,
  - Cấu trúc phân trang.
- Một chương trình thực hiện có thể chứa nhiều cấu trúc khác nhau.



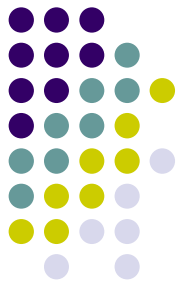
# CẤU TRÚC CHƯƠNG TRÌNH



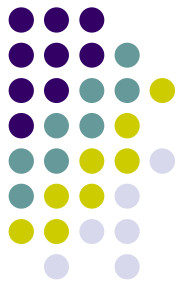
- A) **Cấu trúc tuyến tính**: CT biên tập tìm và lắp ráp các mô đun thành một mô đun duy nhất, chứa đầy đủ thông tin để thực hiện CT,



# Cấu trúc tuyến tính

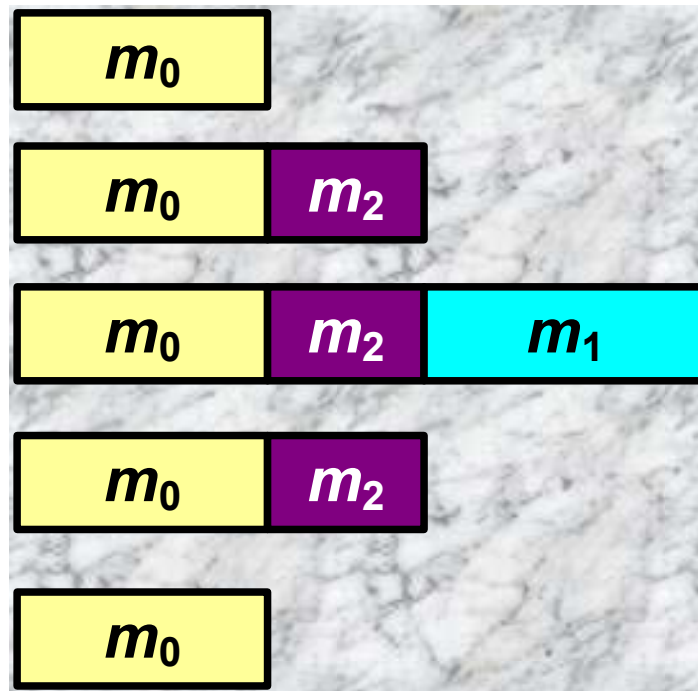


- Đặc điểm:
  - Đơn giản,
  - Thời gian thực hiện: min,
  - Lưu động (mobile) cao,
  - Tồn bộ nhớ: với mỗi bộ dữ liệu chỉ có 13% - 17% câu lệnh đóng vai trò tích cực.
  - Không dùng chung mô đun CT.



## B) CẤU TRÚC ĐỘNG

- Trong CT nguồn: phải dùng các lệnh macro hệ thống để nạp, móc nối, xoá (Load, Attach, Delete) . . . các mô đun khi cần thiết,



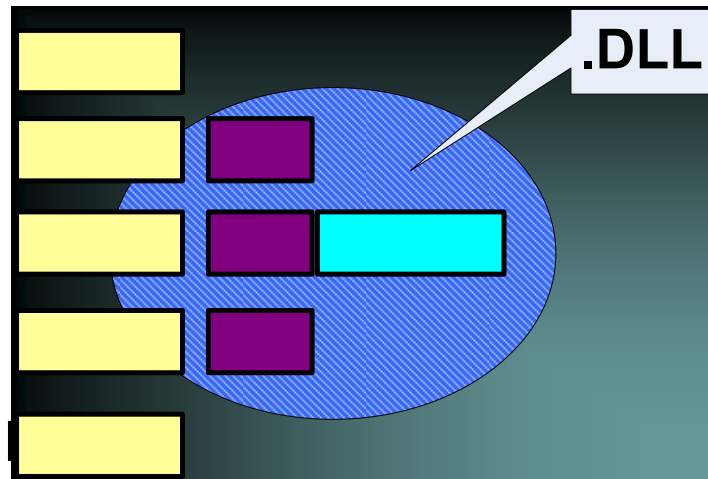
# CẤU TRÚC ĐỘNG



- Đặc điểm:
  - Đòi hỏi user phải biết cơ chế và công cụ quản lý bộ nhớ,
  - Thời gian thực hiện lớn: song song thực hiện với tìm kiếm, nạp và định vị,
  - Tiết kiệm bộ nhớ,
  - Kém lưu động → khó nạp, cập nhật, xoá.
- Được sử dụng rộng rãi những năm 60-70 và từ 90 đến nay.
- Thích hợp cho các CT hệ thống.

# CẤU TRÚC ĐỘNG

- Các mô đun nạp trong quá trình thực hiện → vào các files .DLL ( dynamic Link Library)

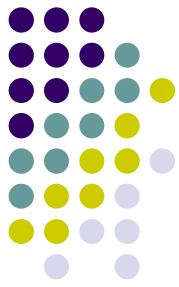


- WINDOWS 98, WINDOWS NT, WINDOWS 2000, SYSTEM32,
- Biên bản cài đặt, uninstall.
- Winword, Excel, Vietkey . . .
- Các ngôn ngữ lập trình:  $\exists$  công cụ tổ chức DLL.

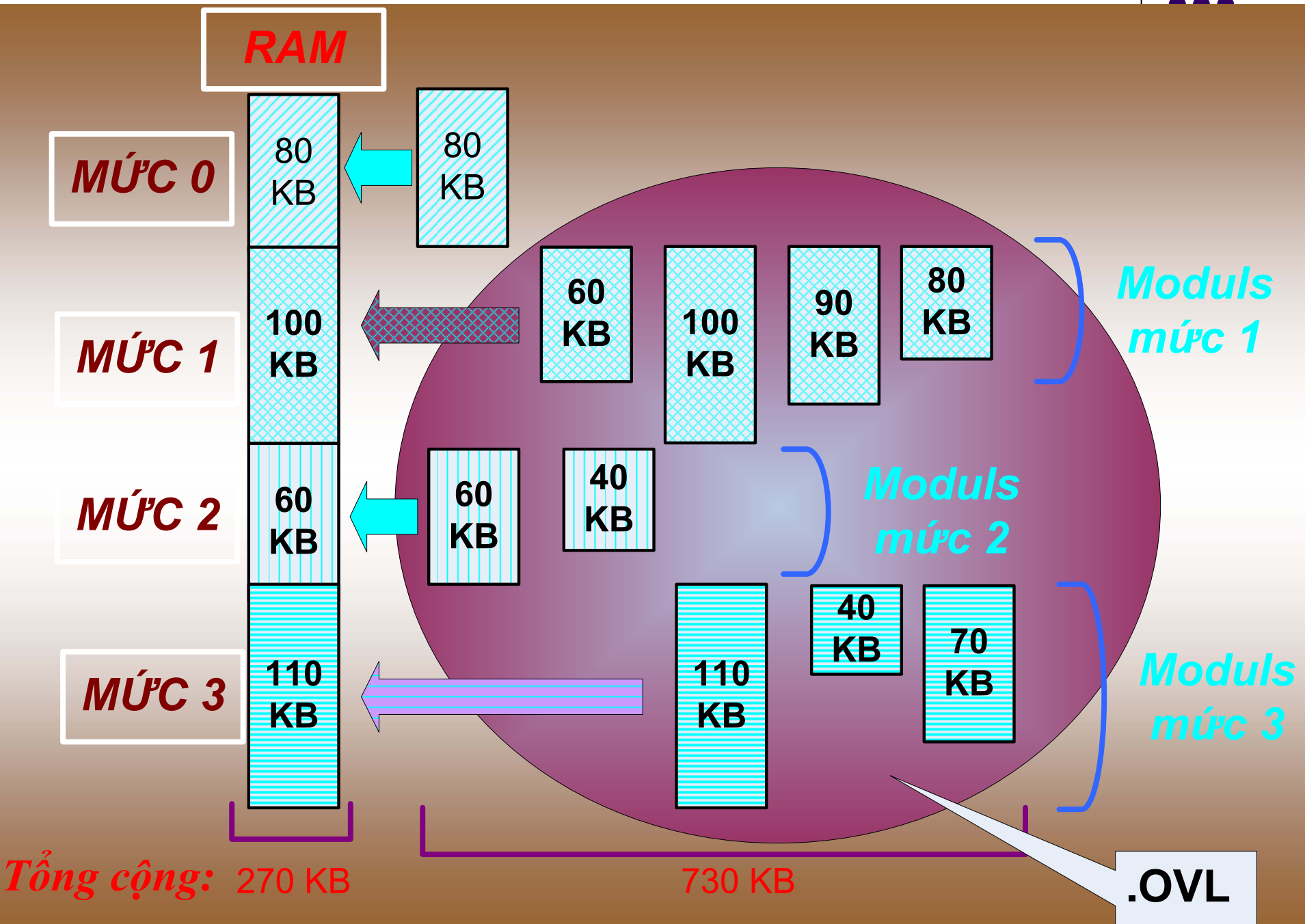
$m_0$

$m_0$

$m_0$



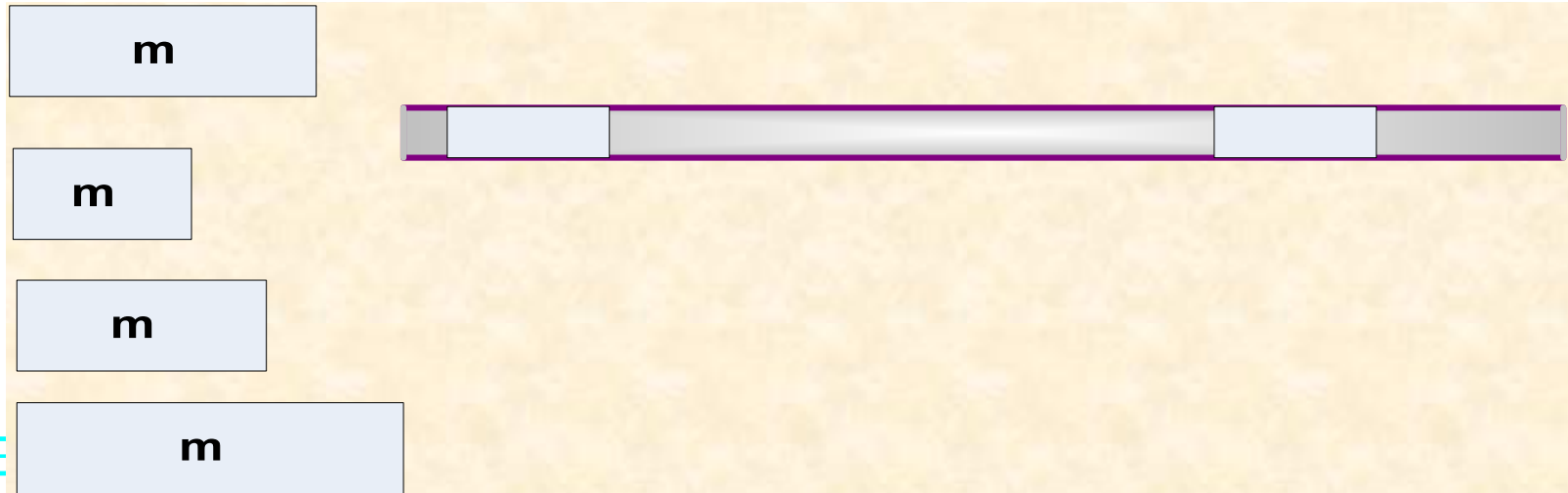
- Moduls → các lớp, lớp = {các moduls không tồn tại đồng thời}
- Moduls lớp i được gọi bởi moduls lớp i-1,
- Thông tin về các lớp: **Sơ đồ tổ chức overlay**, do user cung cấp cho Link,
- Link tạo sơ đồ quản lý overlay,
- Supervisor Overlay tổ chức thực hiện.
- Đặc điểm:
  - Phân phối bộ nhớ theo sơ đồ tĩnh,
  - Files .OVL
- Ví dụ: FOXPRO, PCSHELL. . . .



# D) CẤU TRÚC MODULS



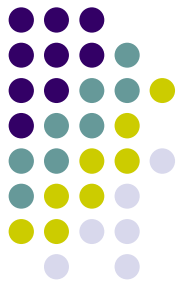
- Biên tập riêng từng mô đun,
- Tạo bảng quản lý mô đun để điều khiển thực hiện,



- Tự động hoàn toàn,
- Không cần phân phối bộ nhớ liên tục,
- Hiệu quả phụ thuộc vào cấu trúc ban đầu của CT nguồn,
- Dễ dàng sử dụng chung mô đun.

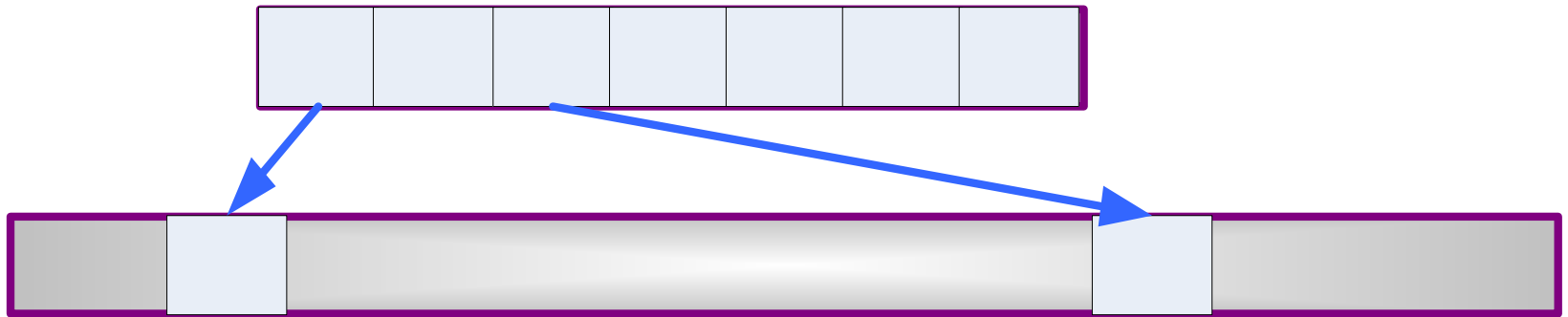
**m<sub>0</sub>**





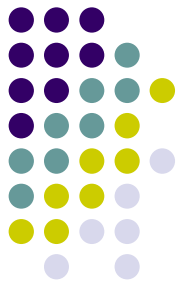
## E) CẤU TRÚC PHÂN TRANG

- CT biên tập như cấu trúc tuyến tính,
- Chia thành các phần bằng nhau – trang,
- Tạo bảng quản lý trang.



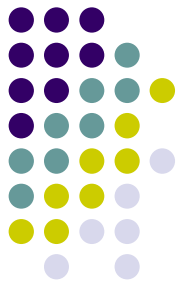
- **Đặc điểm:**
- Tiết kiệm bộ nhớ,
- Hiệu quả không phụ thuộc vào cấu trúc ban đầu của CT nguồn.

# 3 - QUẢN LÝ BỘ NHỚ VẬT LÝ



- **Đặc điểm:**
  - Có kích thước cụ thể,
  - Có cấu hình sử dụng cụ thể.
- **Phục vụ giai đoạn thực hiện CT:**
  - Bảo vệ thông tin,
  - Bộ nhớ cho dữ liệu.
- **Vấn đề:**
  - Cách tổ chức?
  - Xác lập quan hệ với bộ nhớ lô gíc: như thế nào và khi nào?
  - Tình huống thiếu bộ nhớ?

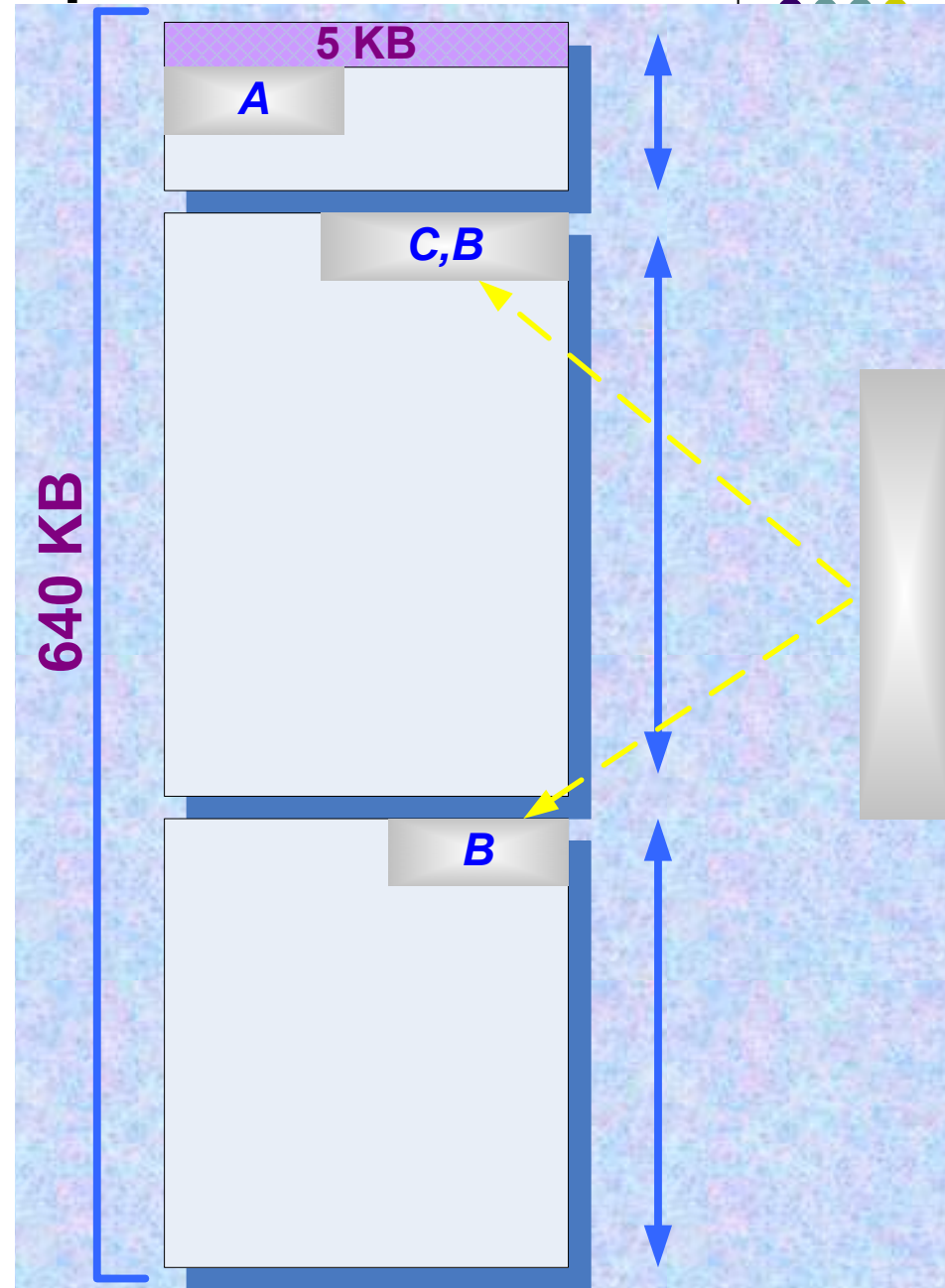
# QUẢN LÝ BỘ NHỚ VẬT LÝ



- Các chế độ quản lý bộ nhớ vật lý:
  - Chế độ phân vùng cố định,
  - Chế độ phân vùng động,
  - Chế độ mô đun,
  - Chế phân trang,
- Chế độ kết hợp mô đun và phân trang.
- Mọi chế độ: đều đang được sử dụng.

## a) Chế độ phân vùng cố định

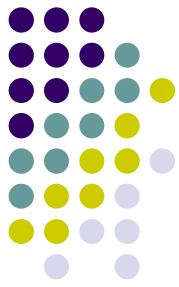
- Bộ nhớ → **n phần**,  
mỗi phần có **kích thước cố định**  
(không nhất thiết bằng nhau),  
sử dụng như một **bộ nhớ độc lập**,  
phục vụ **thực hiện 1 CT**.



# Chế độ phân vùng cố định



- **Đặc điểm:**
  - Mỗi vùng có một danh sách quản lý bộ nhớ tự do,
  - Mỗi vùng: thực hiện một CT ứng dụng,
  - Sơ đồ bảo vệ thông tin: theo toàn vùng.
  - Một số CT điều khiển phải được copy vào từng vùng.
- **Phân lớp** CT phục vụ để hạn chế lãng phí bộ nhớ,
- **Mô hình:** Tổ chức đĩa cứng.



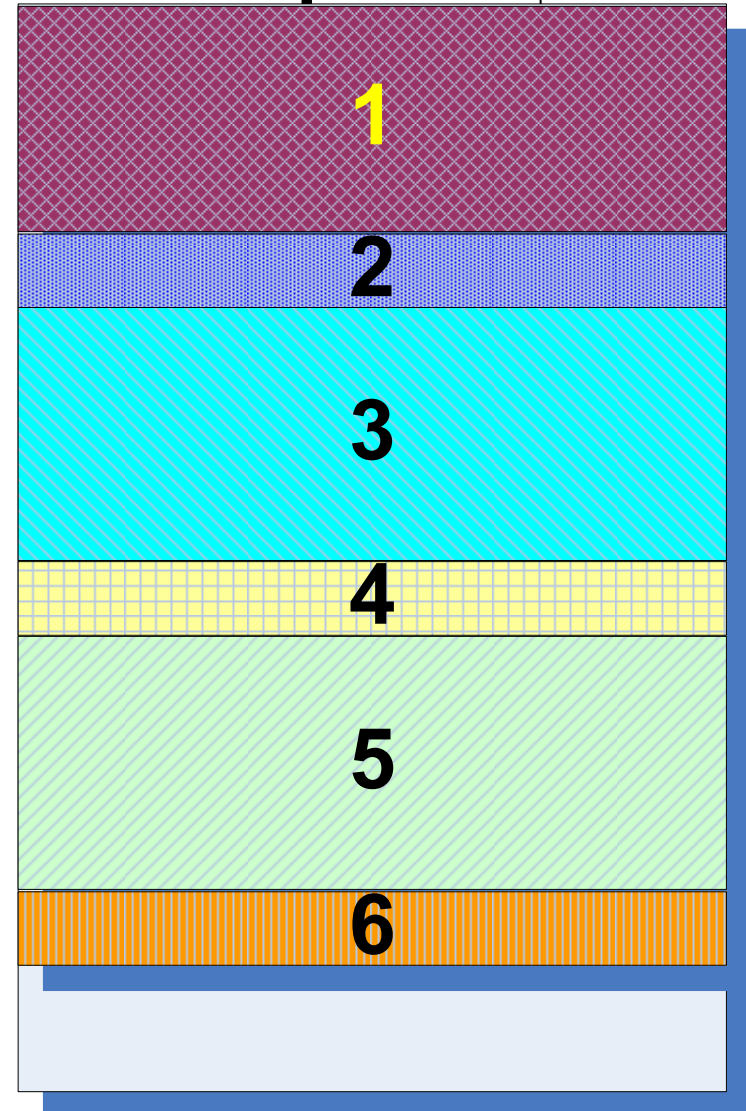
## Chế độ phân vùng cố định

- Công cụ phân bố lại bộ nhớ (**SWAPPING**):
  - **Lệnh OP,**
  - **Do OP thực hiện,**
  - **Những vùng nào biên thay đổi: mất thông tin.** Lý do: làm lại DSQL bộ nhớ tự do.
- Ví dụ: với đĩa cứng: FDISK.
- CT điều khiển hệ thống: **đơn giản.**
- Hệ số song song **cố định.**

## b) CHẾ ĐỘ PHÂN VÙNG ĐỘNG



- CT → Phân phối vùng bộ nhớ **liên tục** đủ thực hiện và quản lý như **bộ nhớ độc lập**.
- $\exists$  **một danh sách QL** bộ nhớ tự do duy nhất.



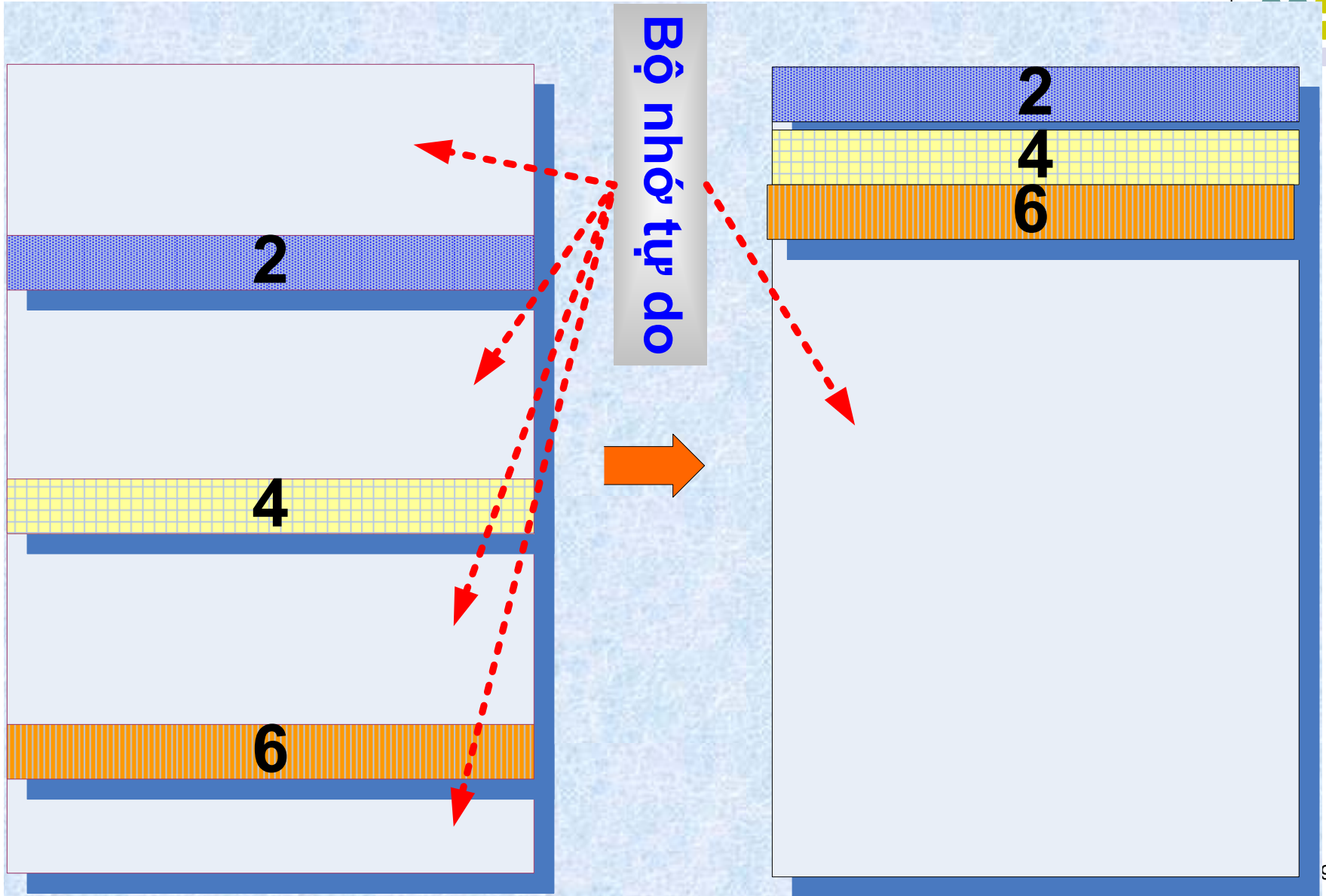
# CHẾ ĐỘ PHÂN VÙNG ĐỘNG



- **Đặc điểm:**
  - Hệ số song song biến thiên,
  - $\exists$  hiện tượng phân đoạn ngoài (External Fragmentation) → SWAPPING,
- **Công cụ SWAPPING:**
  - Lệnh OP,
  - Do OP thực hiện,
  - Không mất thông tin.
- **Nội dung SWAPPING.**
- **Phức tạp** của Swapping.
- Mô hình quản lý đĩa từ **SUBST, DRVSPACE**

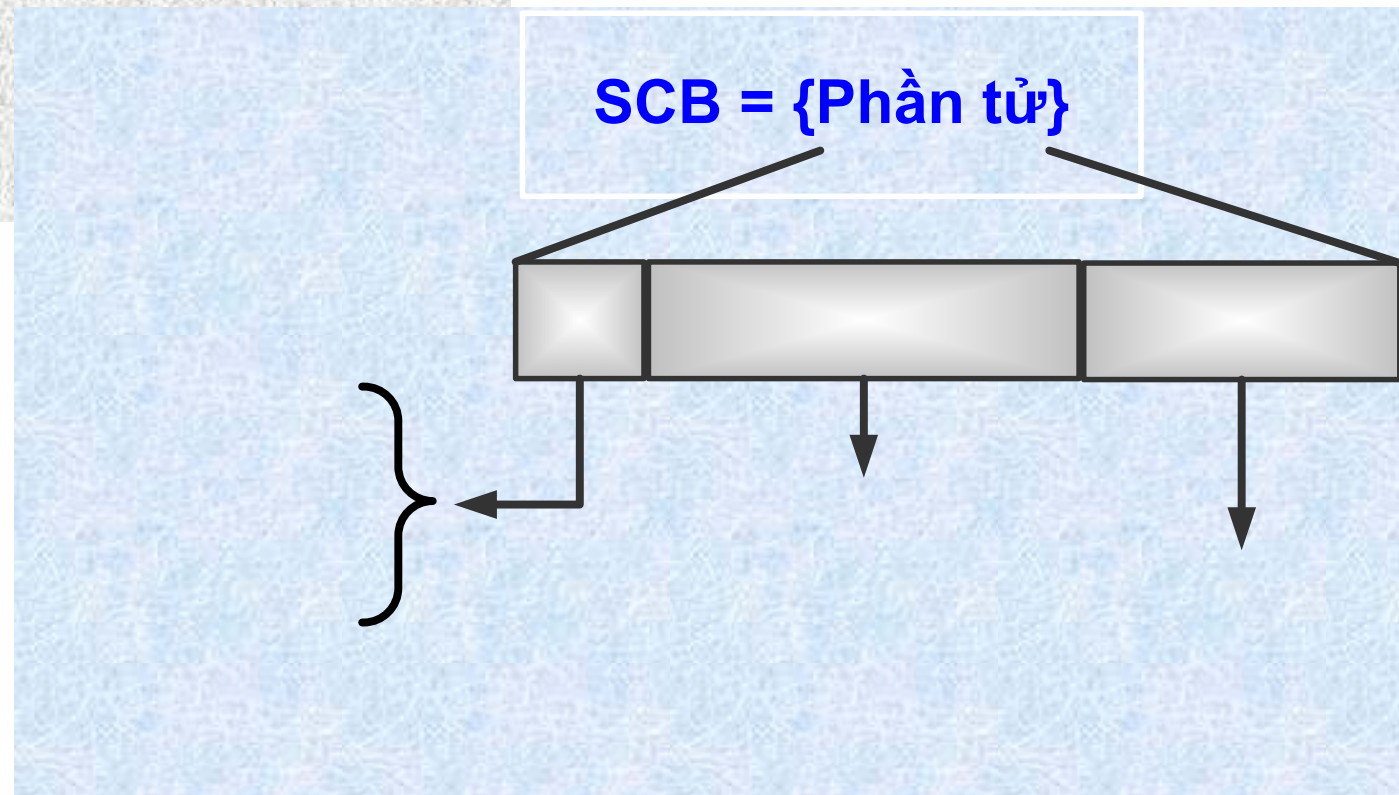
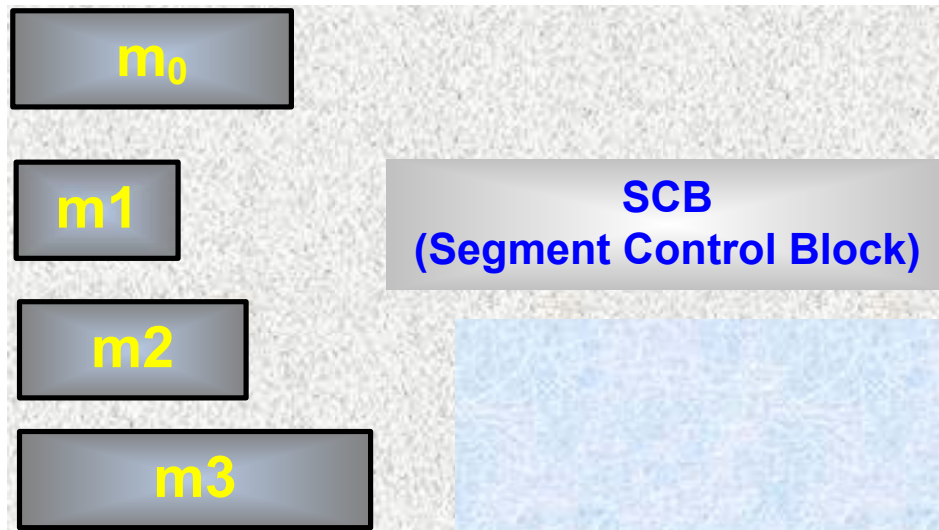
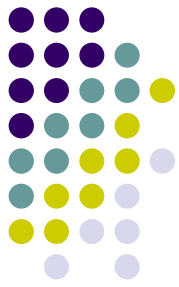


# CHẾ ĐỘ PHÂN VÙNG ĐỘNG

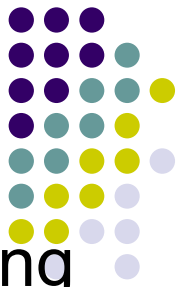


# C) CHẾ ĐỘ QUẢN LÝ THEO MÔ ĐUN

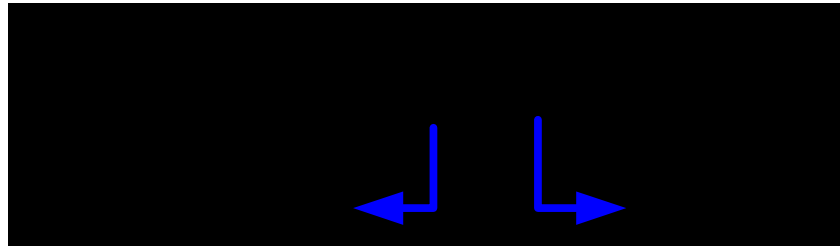
- CT – cấu trúc mô đun,



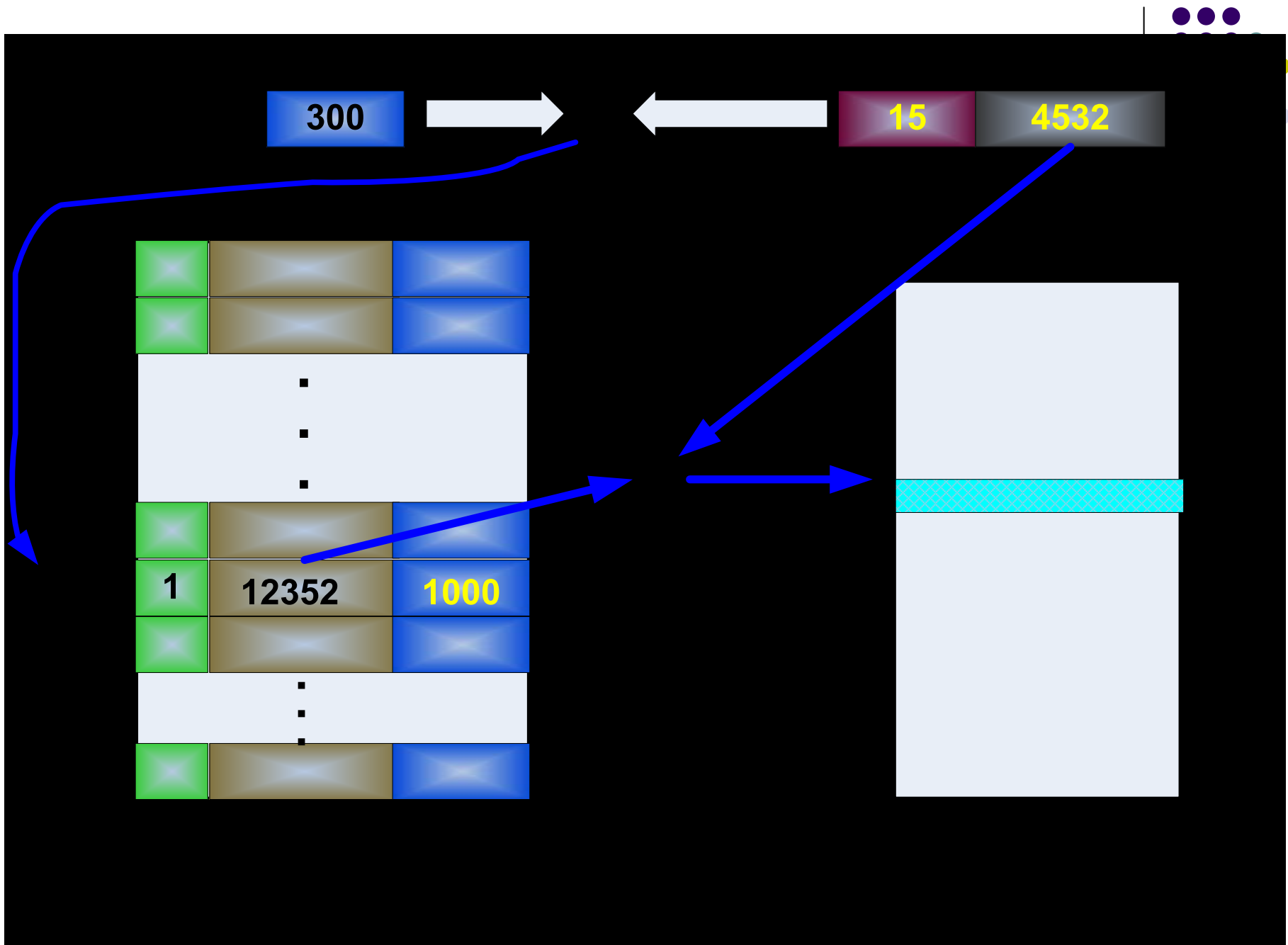
# CHẾ ĐỘ QUẢN LÝ THEO MÔ ĐUN



- Thực hiện CT: địa chỉ dữ liệu phải biểu diễn dưới dạng một cặp



- SCB  $\rightarrow$  RAM, địa chỉ đầu của SCB  $\rightarrow R_s$ - Segment Register.
- Để đọc /ghi dữ liệu: cần 2 lần truy nhập tới bộ nhớ:
  - \*  $(R_s) + s \rightarrow$  truy nhập tới phần tử thứ  $s \in$  SCB,
  - \*\* Khi  $D = 1$ :  $A+d \rightarrow$  truy nhập tới dữ liệu.

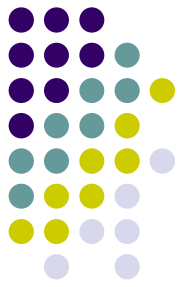


# CHẾ ĐỘ QUẢN LÝ THEO MÔ ĐUN

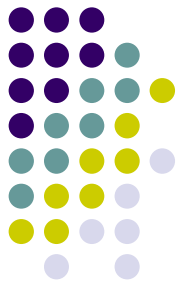


- **Đặc điểm:**
  - Không cần phân phối bộ nhớ liên tục,
  - Không đòi hỏi công cụ đặc biệt → có thể áp dụng cho mọi MTĐT,
  - Dễ dàng sử dụng chung mô đun giữa các CT,
  - Hiệu quả phụ thuộc vào cấu trúc CT nguồn,
  - Tồn tại hiện tượng phân đoạn ngoài (External Fragmentation).
- Thiếu bộ nhớ, phân đoạn ngoài → **Swapping**

# CHẾ ĐỘ QUẢN LÝ THEO MÔ ĐUN

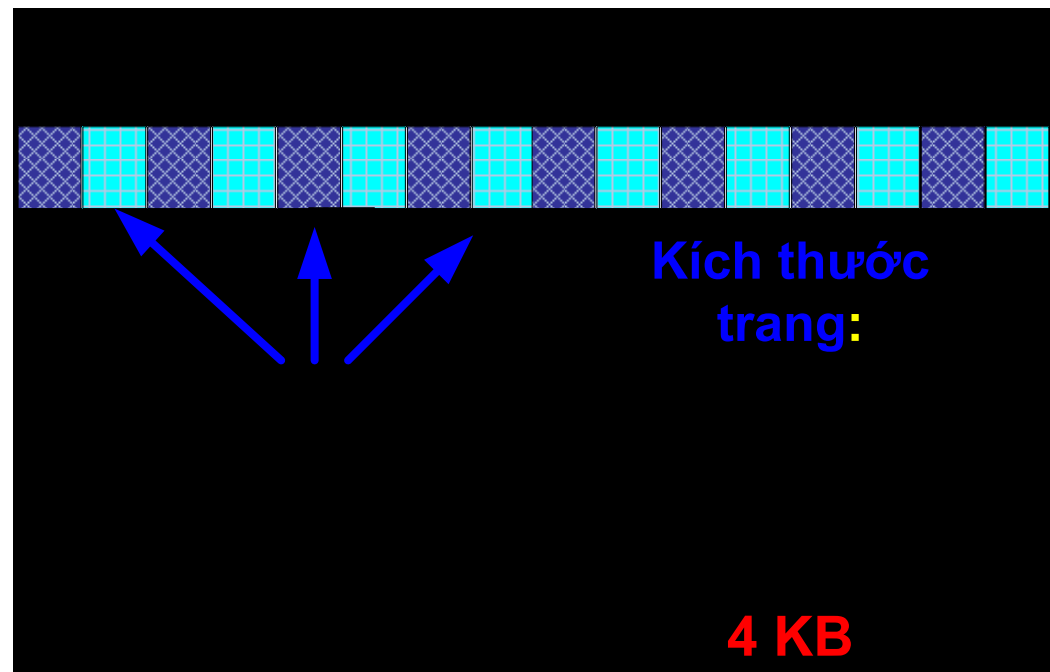


- **SWAPPING:**
  - Do hệ thống đảm nhiệm,
  - Không mất thông tin,
  - Nội dung swapping: đưa một hoặc một số mô đun ra bộ nhớ ngoài, giải phóng chỗ nạp mô đun mới.
- **Cách chọn mô đun đưa ra:** Option
  - Mô đun tồn tại lâu nhất trong bộ nhớ,
  - Mô đun có lần sử dụng cuối cùng cách đây lâu nhất,
  - Mô đun có tần xuất sử dụng thấp nhất.
- **IBM PC 286 trở lên:**
  - Một trong 2 chế độ của 286 và một trong 3 chế độ của 386 trở lên,
  - Swapping - ngầm định – **tiêu chuẩn 2.**



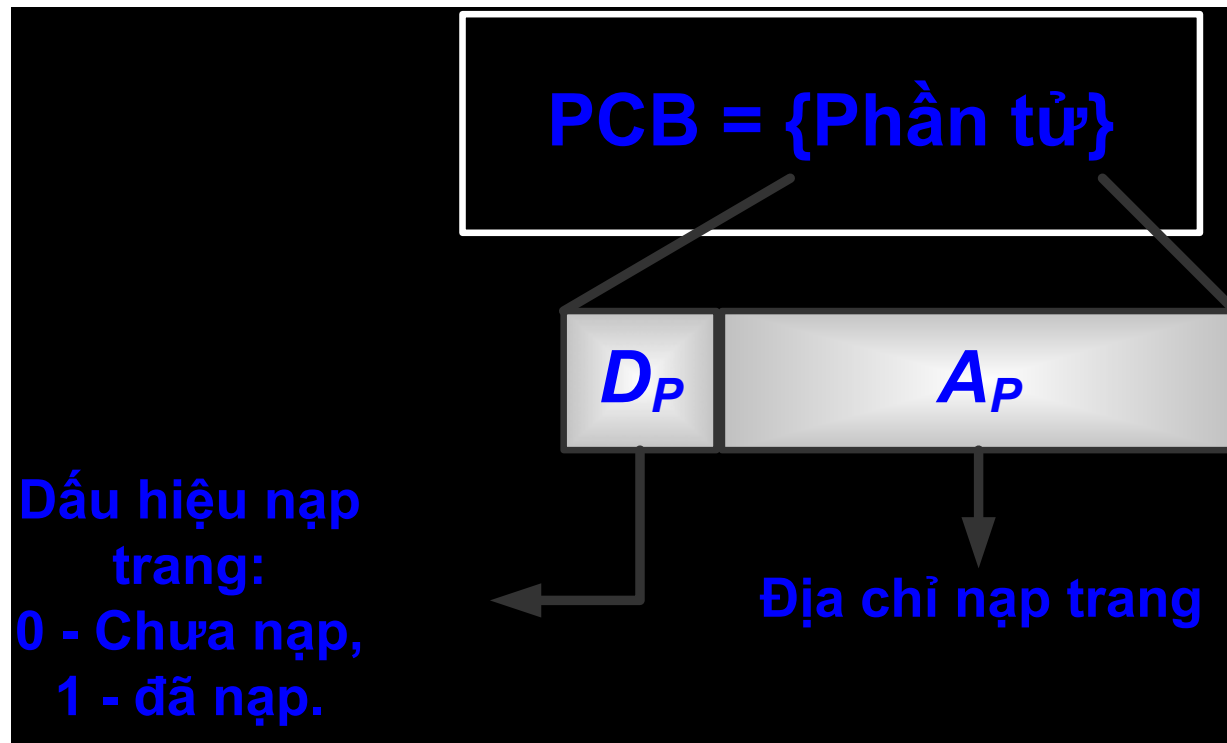
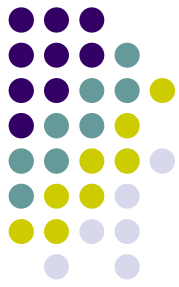
## D) CHẾ ĐỘ PHÂN TRANG

- Bộ nhớ được chia thành các phần bằng nhau – các trang (Pages),
- Các trang – đánh số 0, 1, 2, . . . - địa chỉ trang.



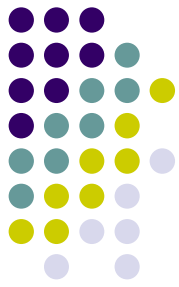
# CHẾ ĐỘ PHÂN TRANG

- CT - cấu trúc phân trang,
- Bảng quản lý trang PCB (Page Control Block),

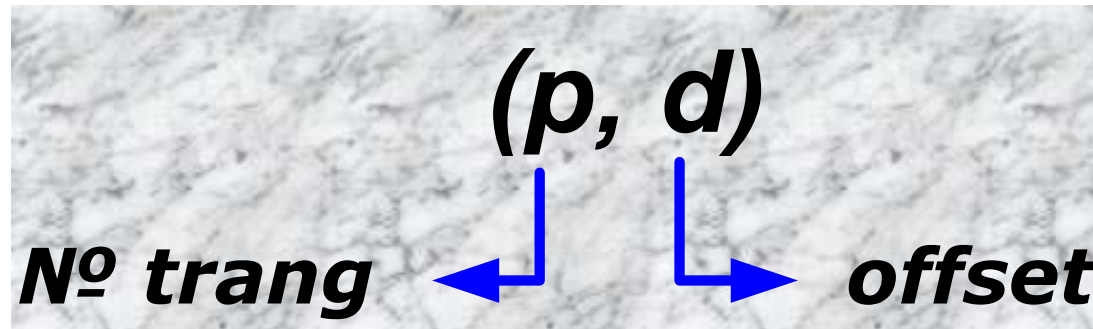




# CHẾ ĐỘ PHÂN TRANG



- Thực hiện CT: địa chỉ dữ liệu phải biểu diễn dưới dạng một cặp



- PCB  $\rightarrow$  RAM, địa chỉ đầu của PCB  $\rightarrow R_p$ - Page Register.
- Để đọc /ghi dữ liệu: cần 2 lần truy nhập tới bộ nhớ:
  - \*  $(R_p) + p \rightarrow$  truy nhập tới phần tử thứ  $p \in$  PCB,
  - \*\* Khi  $D_p = 1$ :  $A \cup d \rightarrow$  truy nhập tới dữ liệu.

R =

400

+

p

81

d

532

400

123000

p

480

481

1

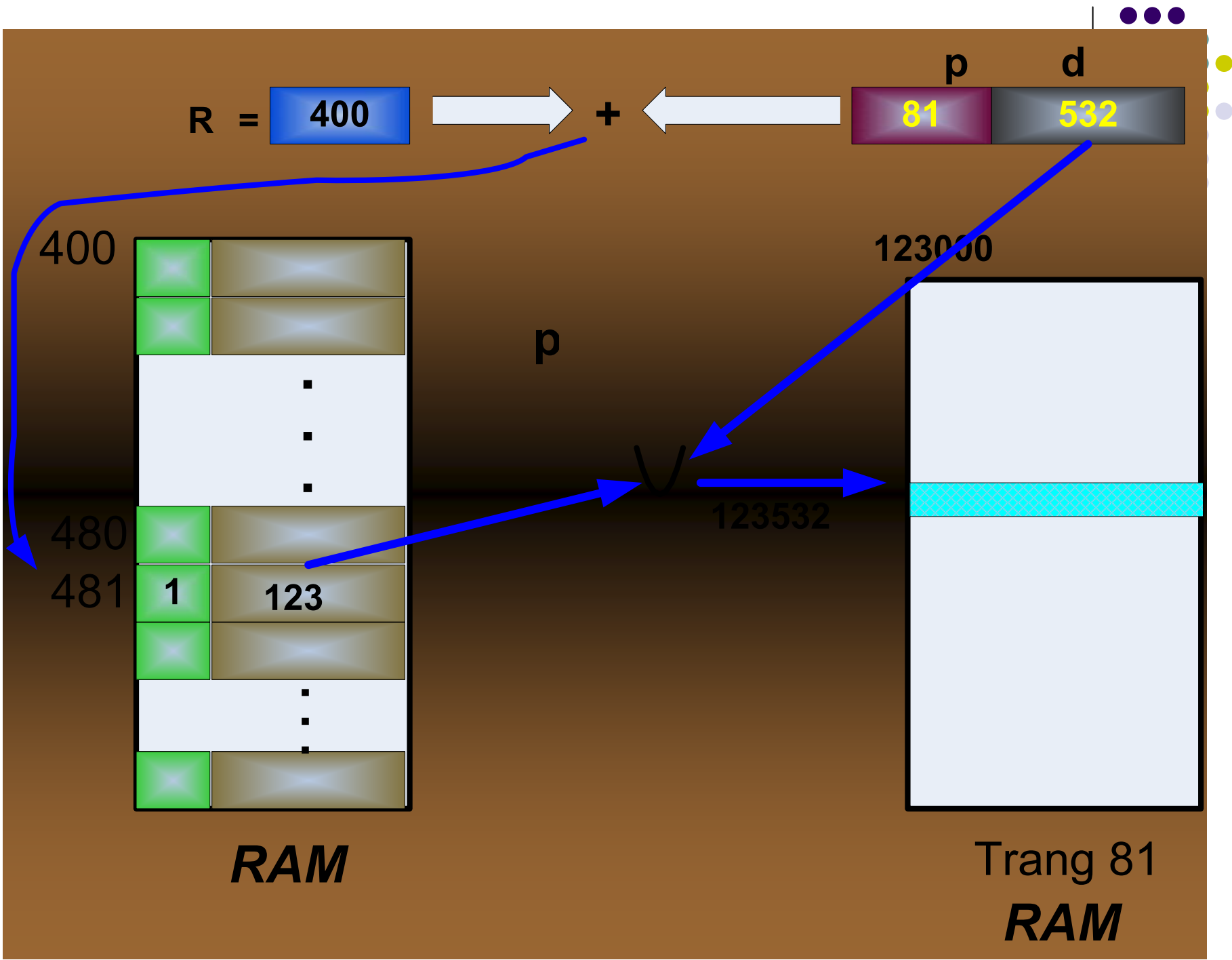
123

123532

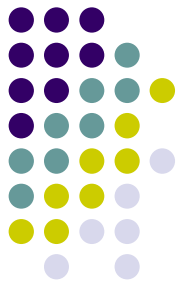
*RAM*

Trang 81

*RAM*



# CHẾ ĐỘ PHÂN TRANG



- **Đặc điểm:**
  - Không cần phân phối bộ nhớ liên tục,
  - Phải có công cụ kỹ thuật hỗ trợ định vị trang,
  - Không sử dụng chung mô đun giữa các CT,
  - Hiệu quả không phụ thuộc vào cấu trúc CT nguồn,
  - Bảng PCB có thể rất lớn,
  - Không bị phân đoạn ngoài.
- Thiếu bộ nhớ (mọi trang đều đã được sử dụng) → **Swapping**



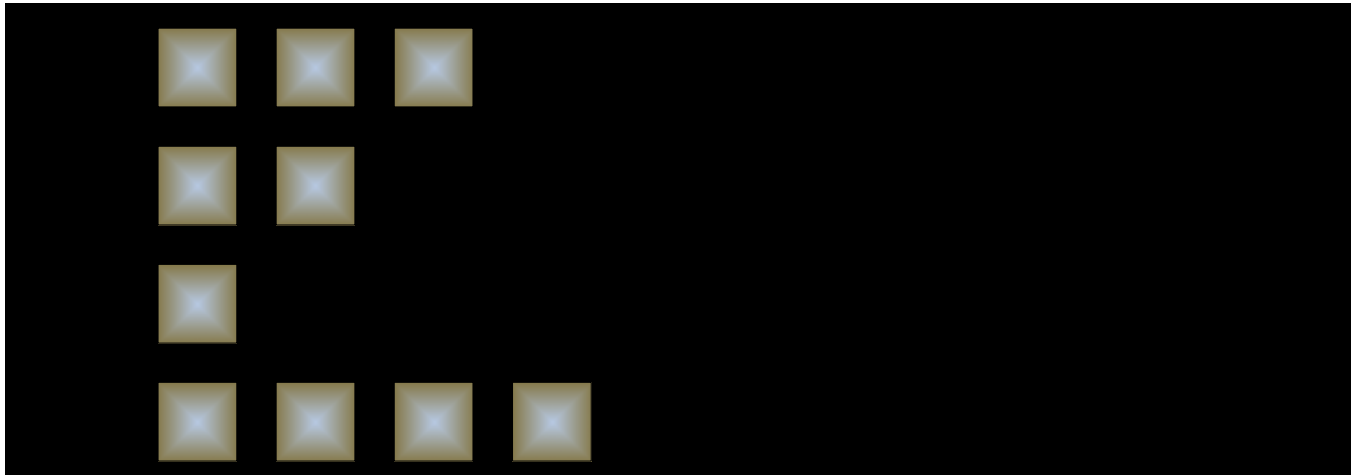
# CHẾ ĐỘ PHÂN TRANG

- **SWAPPING:**
  - Do hệ thống đảm nhiệm,
  - Không mất thông tin,
  - Nội dung swapping: đưa một trang ra bộ nhớ ngoài, giải phóng chỗ nạp trang mới.
- **Cách chọn trang đưa ra:** Option
  - Trang tồn tại lâu nhất trong bộ nhớ,
  - Trang có lần sử dụng cuối cùng cách đây lâu nhất,
  - Trang có tần xuất sử dụng thấp nhất.
- IBM PC 386 trở lên: ngầm định – **tiêu chuẩn 2.**

# E) CHẾ ĐỘ KẾT HỢP MÔ ĐUN – PHÂN TRANG



- Bộ nhớ vật lý – phân trang,
- CT – cấu trúc mô đun,
- Mỗi mô đun – phân trang:





# CHẾ ĐỘ KẾT HỢP MÔ ĐUN – PHÂN TRANG

**SCB = {Phần tử}**

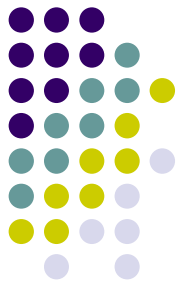


**Dấu hiệu nạp  
PCB:**

**0 - Chưa nạp,  
1 - đã nạp.**

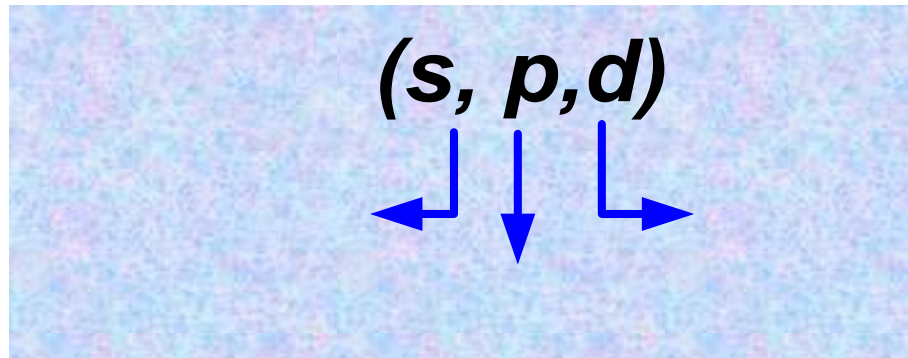
**Địa chỉ nạp PCB**

**Độ dài PCB**  
- Cấp phát bộ  
nhớ,  
- Bảo vệ,

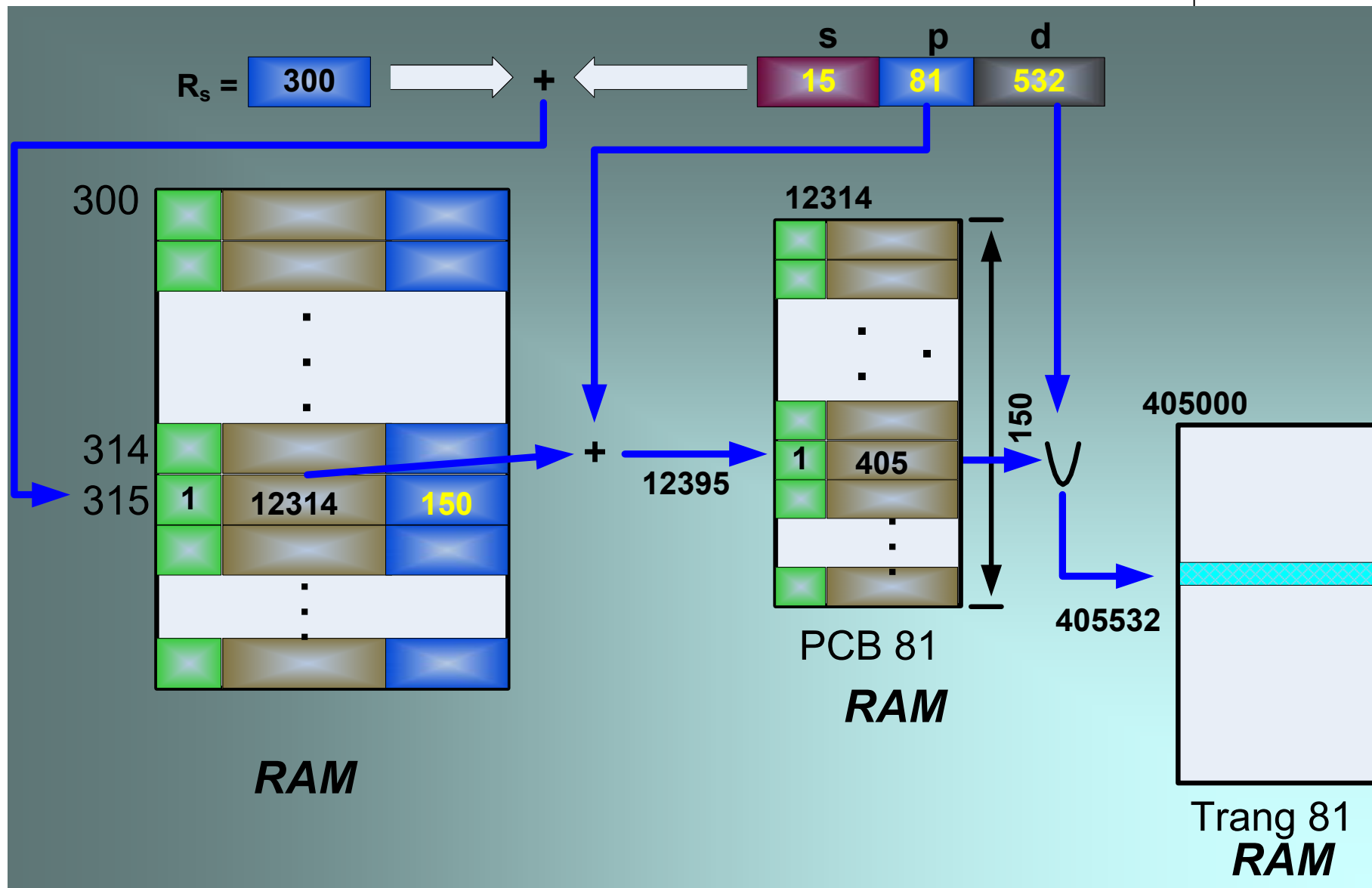


# CHẾ ĐỘ KẾT HỢP MÔ ĐUN – PHÂN TRẠNG

- Thực hiện CT: địa chỉ dữ liệu phải biểu diễn dưới dạng một nhóm 3:

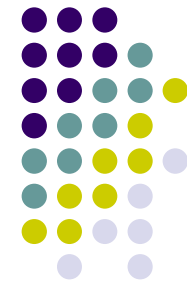


- SCB  $\rightarrow$  RAM, địa chỉ đầu của SCB  $\rightarrow R_s$ - Segment Register.
- Để đọc /ghi dữ liệu: cần 3 lần truy nhập tới bộ nhớ:
  - \*  $(R_s) + s \rightarrow$  truy nhập tới phần tử thứ  $s \in$  SCB,
  - \*\* Khi  $D = 1$ :  $A + d \rightarrow$  truy nhập tới  $PCB_s \in$  SCB,
  - \*\*\* Khi  $D_p = 1$ :  $A \cup d \rightarrow$  truy nhập tới dữ liệu.





# 4 - QUẢN LÝ BỘ NHỚ TRONG IBM PC

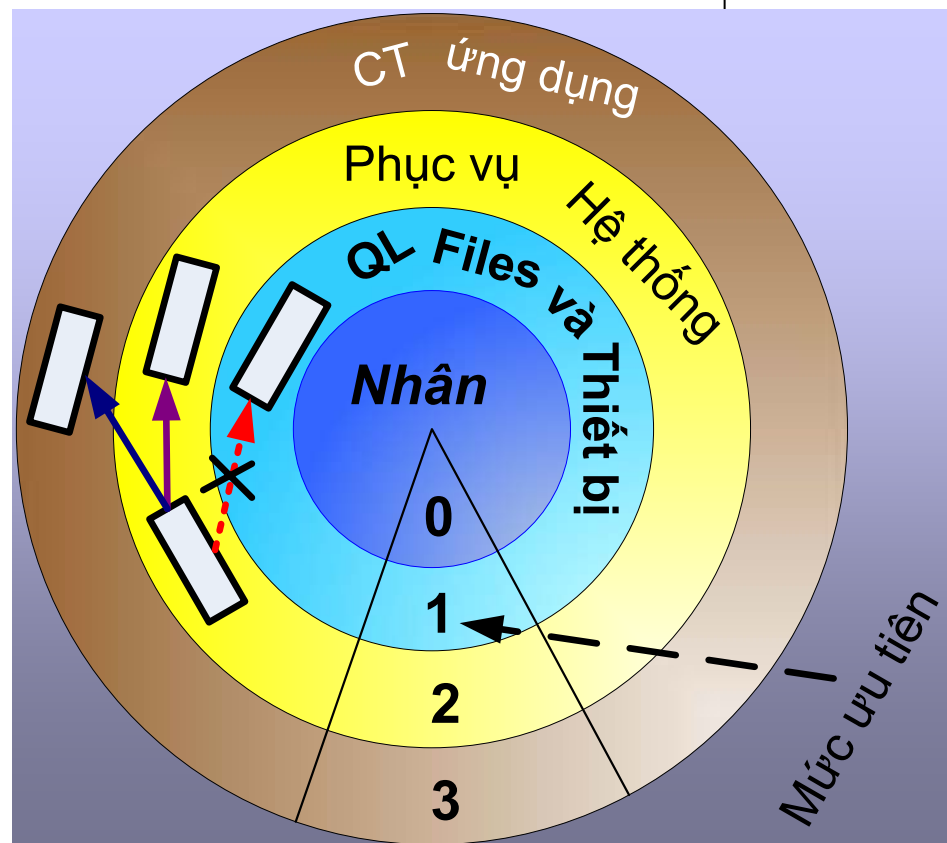


- Bốn mức ưu tiên

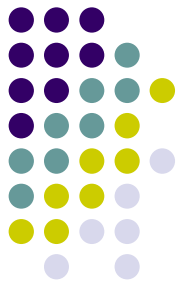
(Privilege Levels)

thực hiện CT:  $0 \div 3$ , cao nhất – 0, thấp nhất – 3.

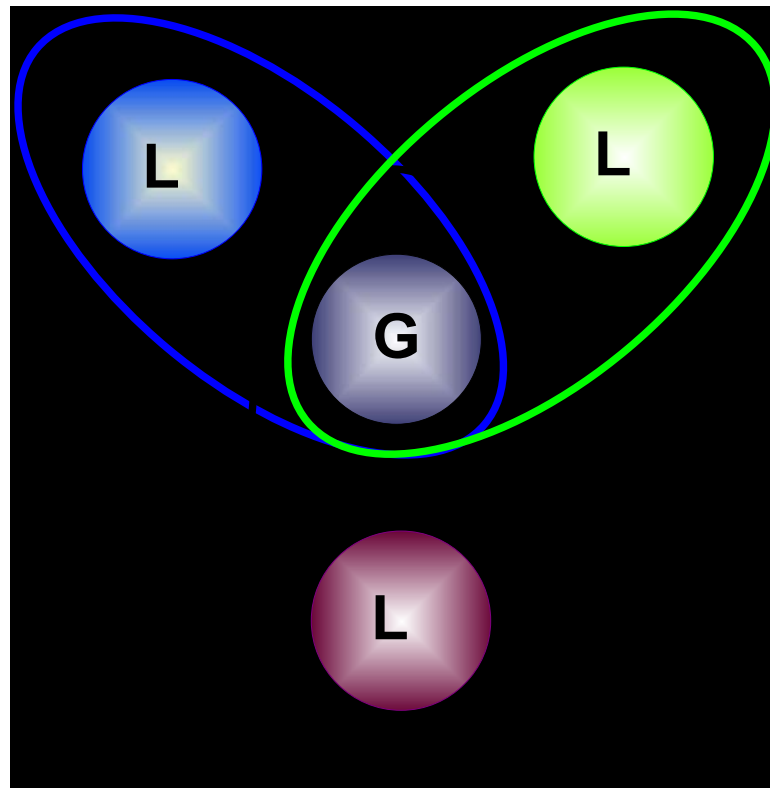
- Nguyên tắc truy nhập: một CT chỉ được quyền truy nhập tới CT và dữ liệu của CT bằng hoặc kém ưu tiên hơn.



# IBM PC

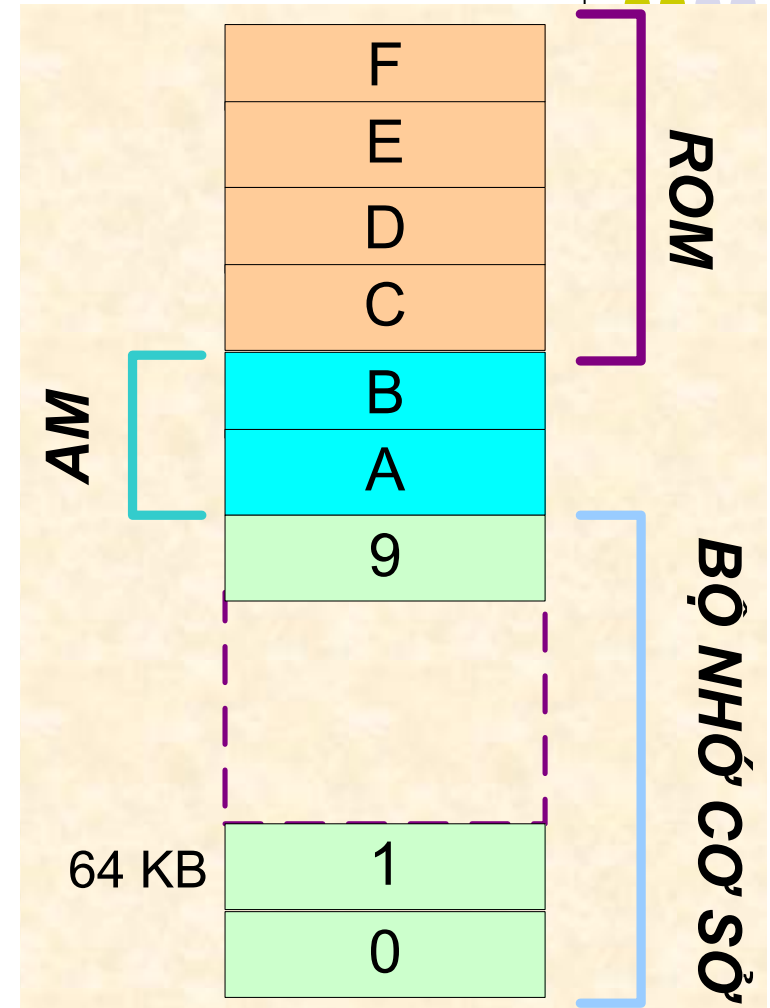


- Bộ nhớ phân phối cho CT - 2 loại: bộ nhớ chung (G) và bộ nhớ riêng (L).

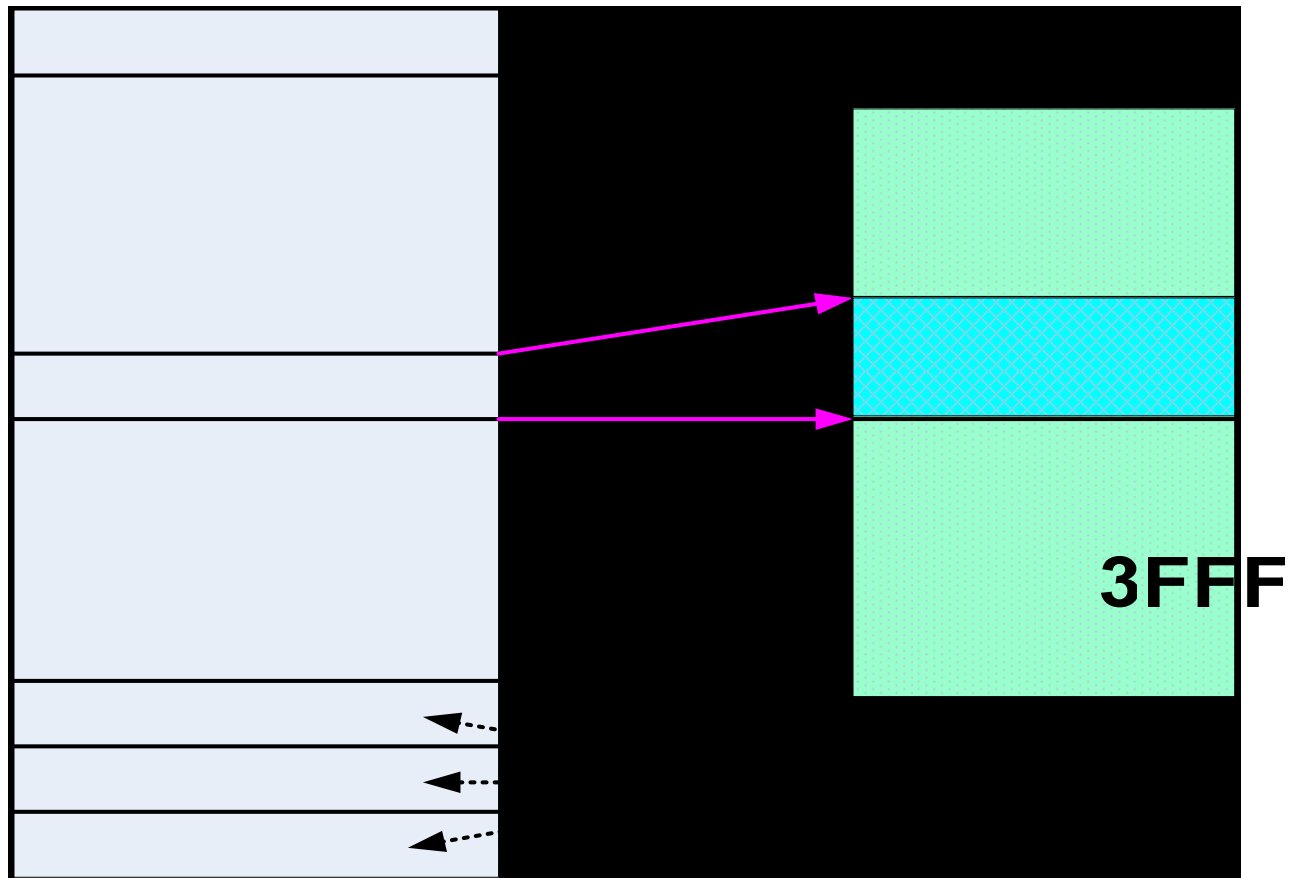
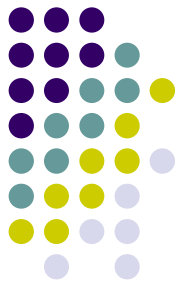


# IBM PC

- 2 chế độ: Chế độ thực (XT) và chế độ bảo vệ (AT).
- Chế độ Real Mode:



# Chế độ Protected Mode

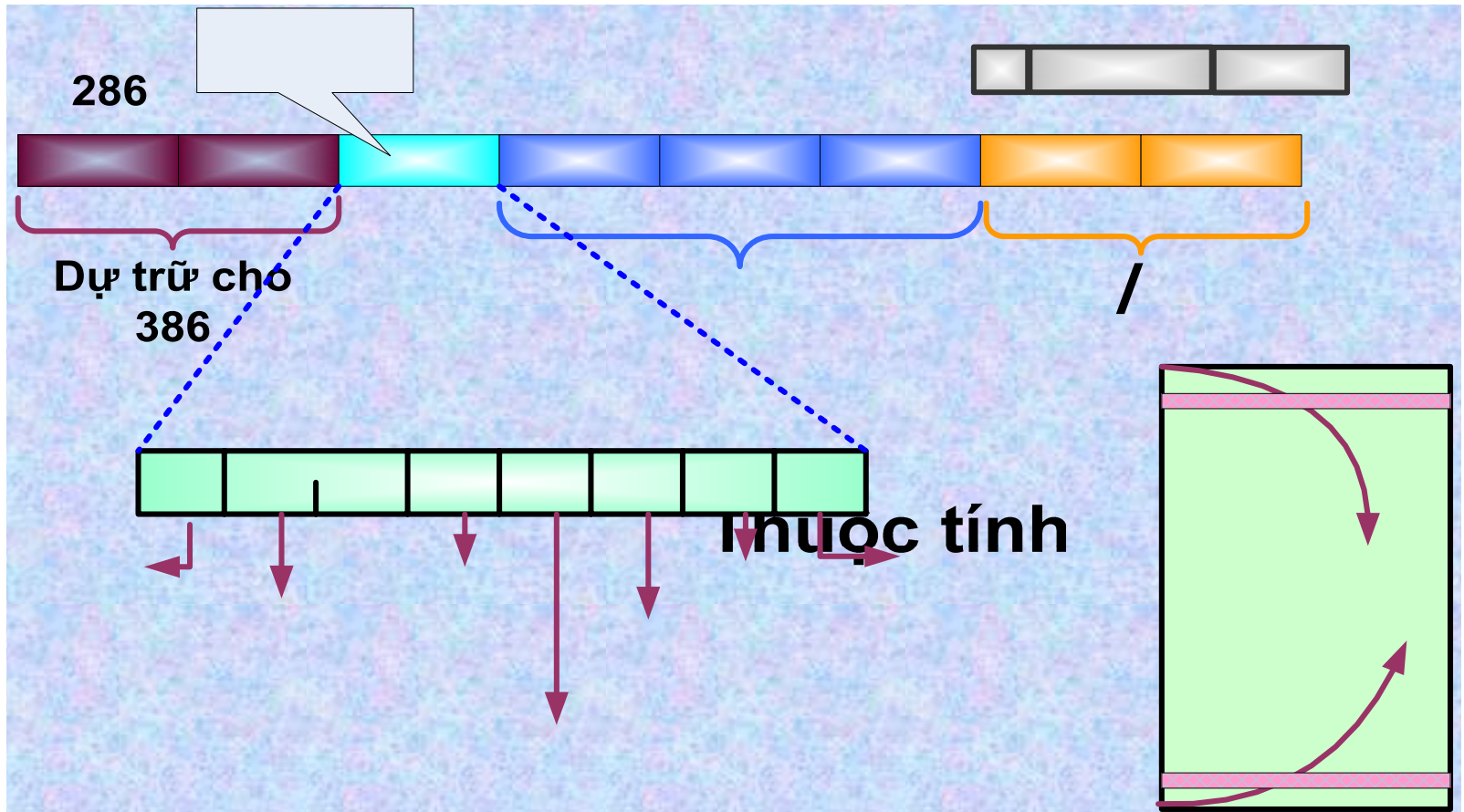
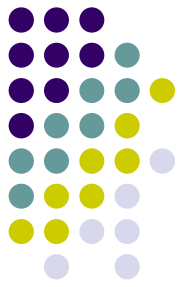


# Chế độ Protected Mode

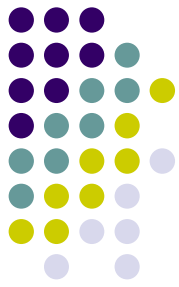


- Mỗi khối  $\Leftrightarrow$  MCB (Memory Control Block)
- Bộ nhớ chung  $\Leftrightarrow \{\text{MCB}\} \rightarrow \text{GDT}$   
(Global Descriptor Table).
- Bộ nhớ riêng  $\Leftrightarrow \{\text{MCB}\} \rightarrow \text{LDT}$   
(Local Descriptor Table).
- MCB: 8 Bytes/phần tử.
- Thực hiện CT: GDT  $\rightarrow$  RAM, GDTR  
LDT  $\rightarrow$  RAM, LDTR  
Lệnh: LGDTR, SGDTR, LLDTR, SLDTR

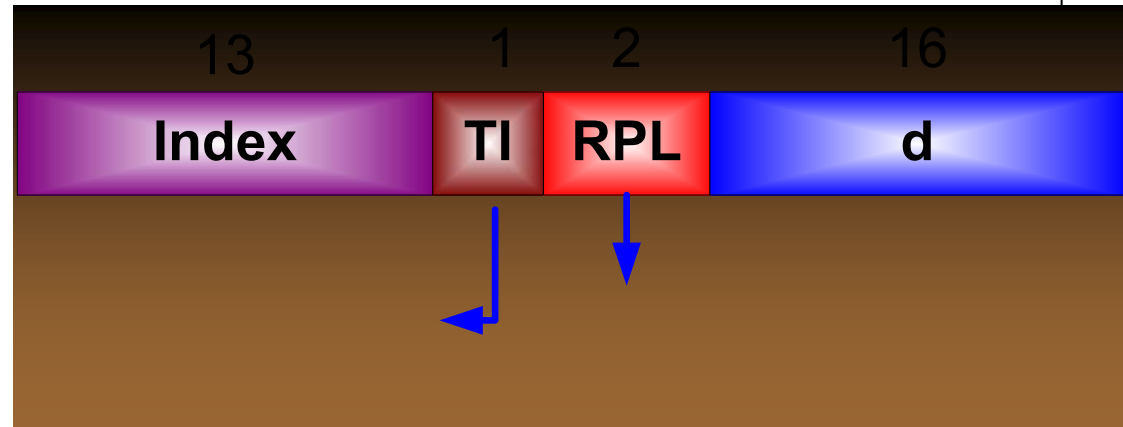
# MCB



# 80286



- Địa chỉ tuyến tính: 32 bits.

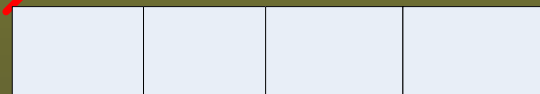
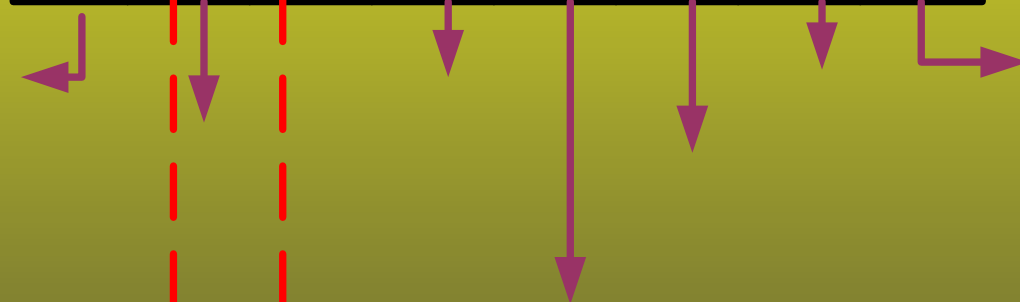
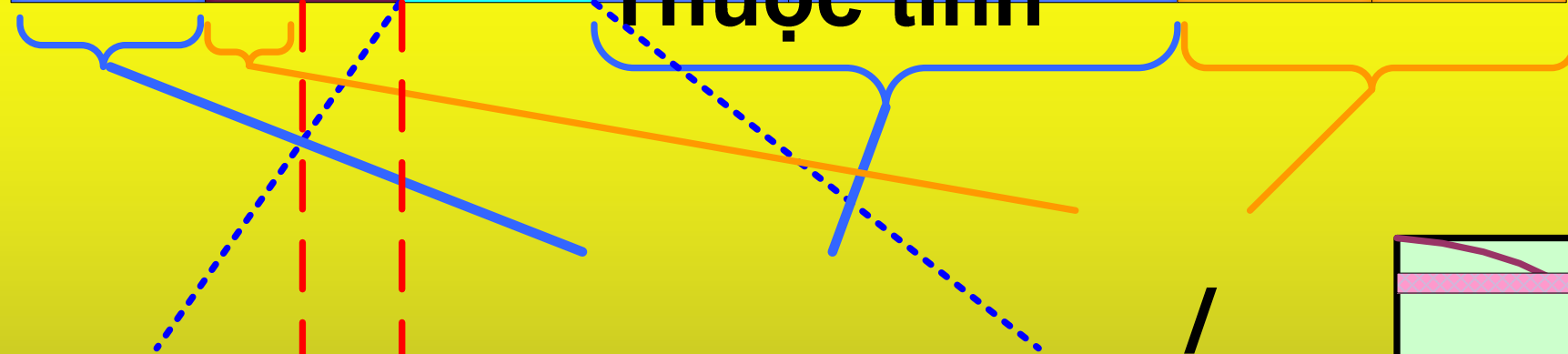


- Khả năng:
  - Vật lý: AR – 24 bits
  - $V_{ph} = 2^{24}$
  - $= 16\text{MB}$
  - Lô gic:  $V_{lg} = 2^{13} \times 2^1 \times 2^{16}$
  - $= 2^{30}$
  - $= 1\text{ GB}$

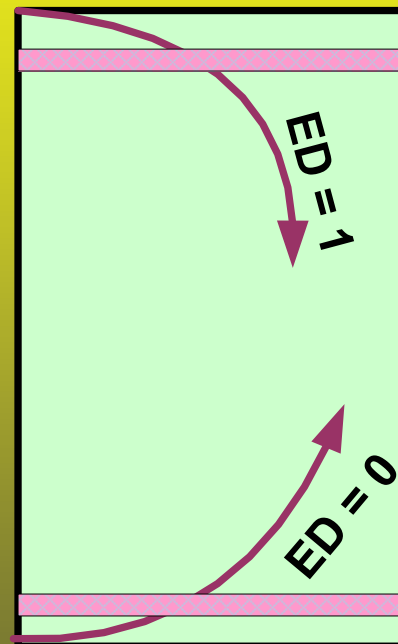
386

MCB

Thuộc tính



/



4 Bytes  
A





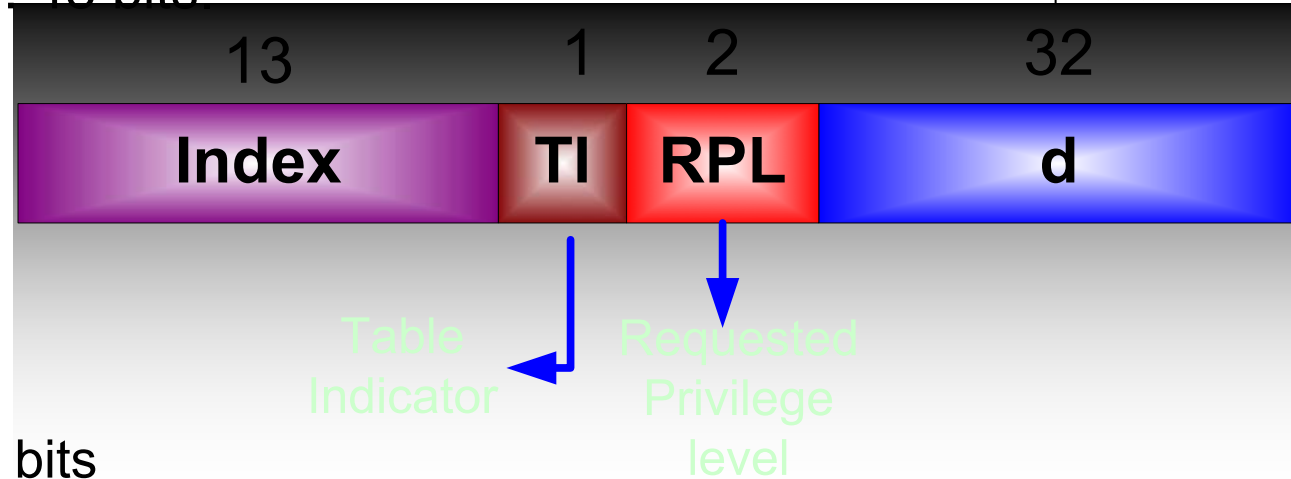
## 80386 - PENTUM

- G = 0 - Chế độ mô đun, đơn vị tính kích thước khối – **Byte** →  $L = 2^{20} = 1 \text{ MB}$ .
- G = 1 - Chế độ phân trang, đơn vị tính kích thước khối – **trang** (4 KB) →  $L = 2^{20} \text{ P} = 2^{20} \times 2^{12} = 2^{32} = 4 \text{ GB}$ .
- D = 0 - Chế độ dữ liệu 16 bit,
- D = 1 - Chế độ dữ liệu 32 bit.

# 80386 - PENTUM

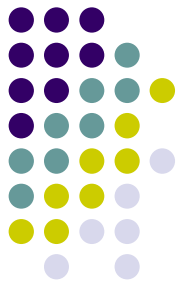


- Địa chỉ tuyến tính: 48 bits

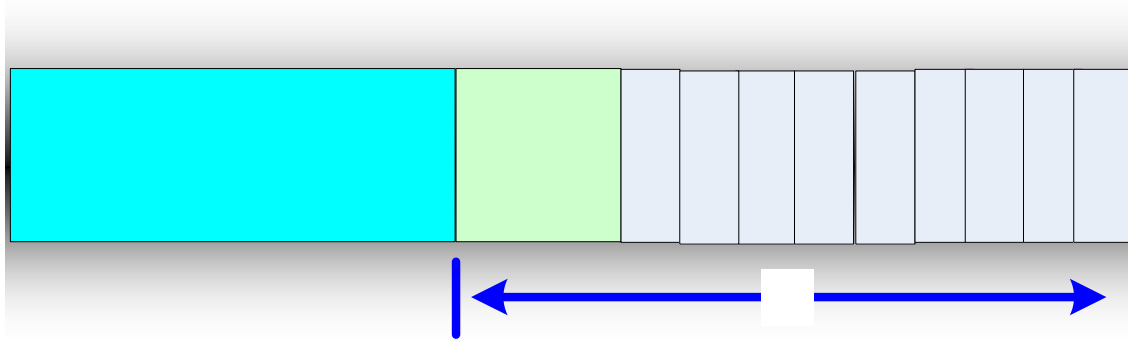


- Khả năng:
  - Vật lý: AR – 32 bits
  - $V_{ph} = 2^{32}$
  - = 4GB
  - Lô gic:  $V_{lg} = 2^{13} \times 2^1 \times 2^{32}$
  - $= 2^{46}$
  - $= 2^6 \times 2^{40}$
  - = 64 TB

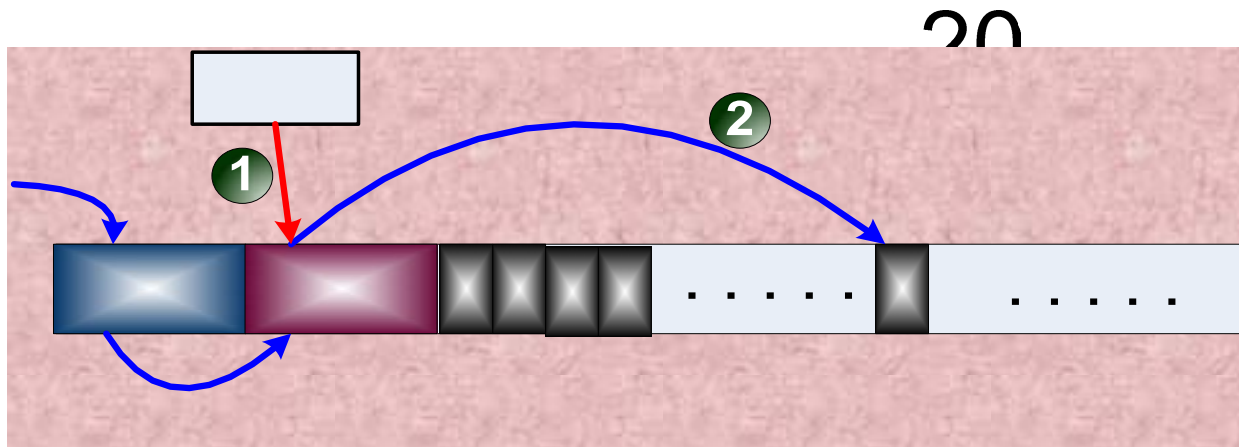
# 80386 - PENTUM



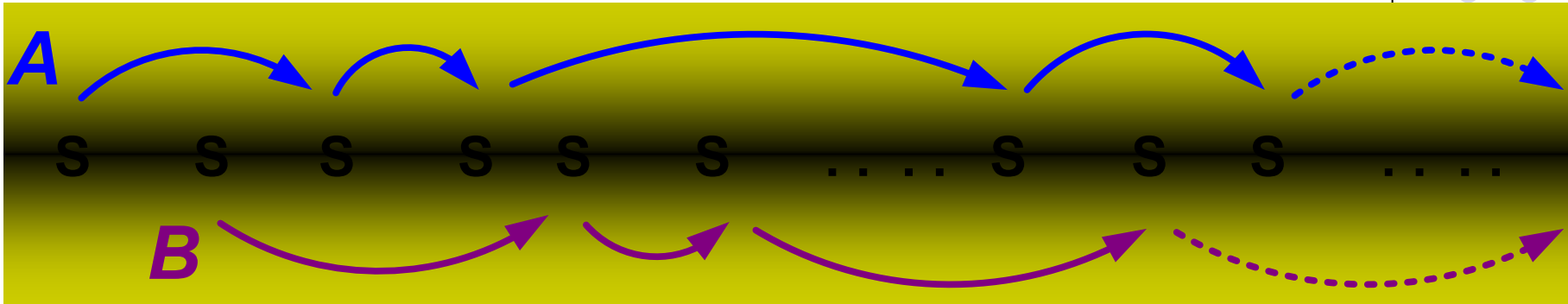
- Chế độ kết hợp mô đun – phân trang:



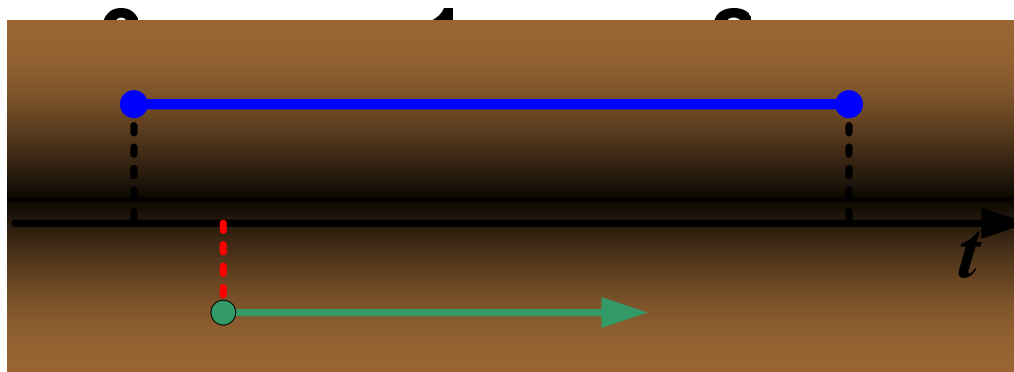
- Phân phối bộ nhớ:



# IV – QUẢN LÝ TIẾN TRÌNH (PROCESS)

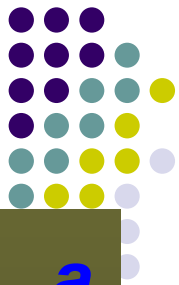


- 2 – **Phân loại**: kế tiếp và song song,
- Tiến trình song song:

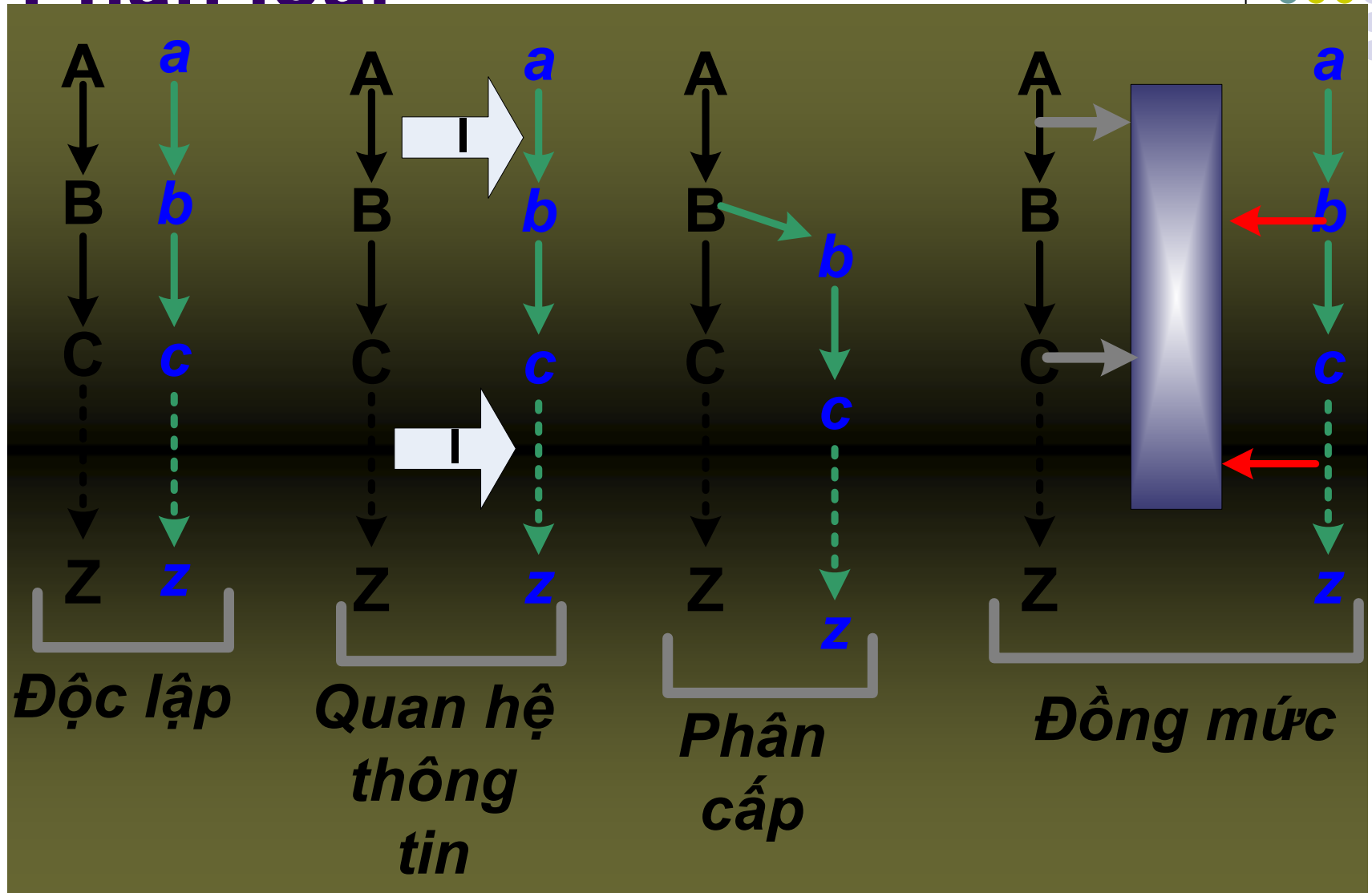


3

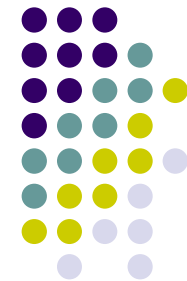
4



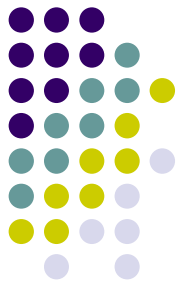
# Phân loại



# Phân loại



- a) Độc lập: Bảo vệ thông tin,
- b) Quan hệ thông tin:
  - Tiến trình nhận: Tồn tại? Ở đâu? Giai đoạn nào?
  - Cơ chế truyền tin:
    - Hộp thư,
    - I/O Ports,
    - Monitor/



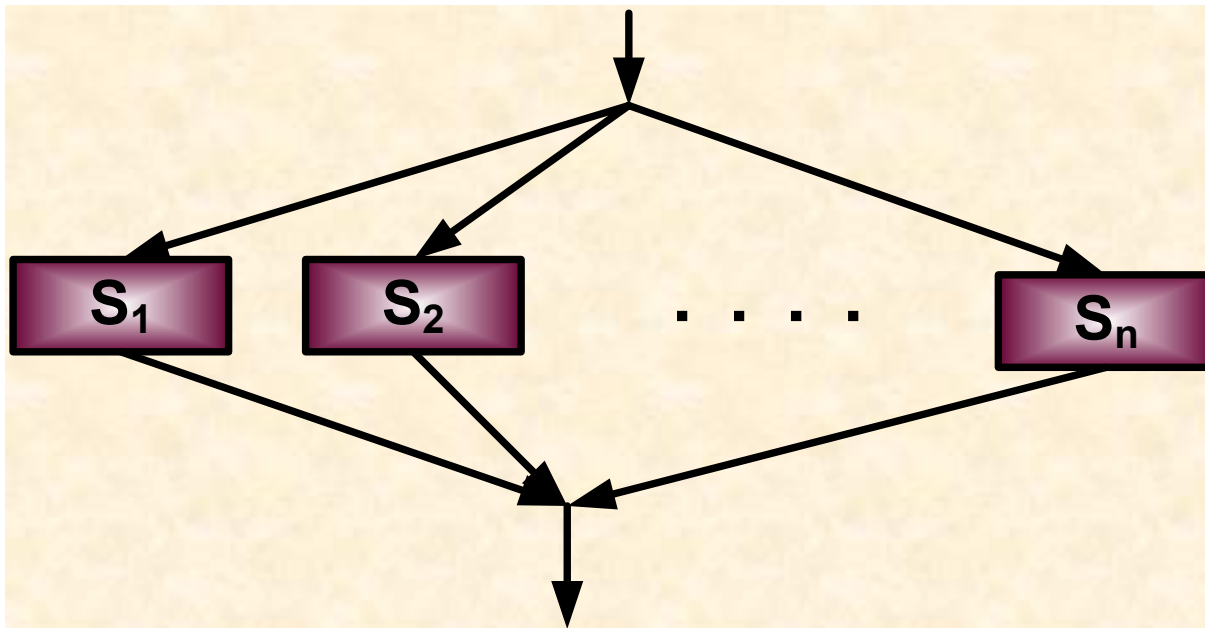
# Phân loại

- c) Phân cấp:
- Tài nguyên cho tiến trình con:
  - Hệ thống QL tài nguyên tập trung: từ hệ thống,
  - Hệ thống QL tài nguyên phân tán: từ vốn tài nguyên tiến trình chính,
- QL phân tán: Tiến trình chính phải kết thúc sau tiến trình con → POST, WAIT.
- d) Đồng mức:
- Sử dụng chung theo nguyên tắc lần lượt,
- Các hệ thống mô phỏng, trò chơi, . . .

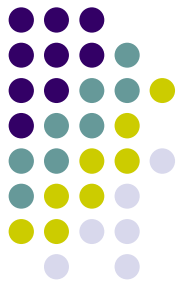
# 3 - BIỂU DIỄN TIỀN TRÌNH SONG SONG



- Giả thiết:  $S_1, S_2, \dots, S_n$  – các công việc thực hiện song song (Trên 1 hoặc nhiều máy).







## BIỂU DIỄN

- 2 cách mô tả phổ biến:

### PARBEGIN

$S_1$  ;

$S_2$  ;

.....

$S_n$

### PAREND;

### COBEGIN

$S_1$  ;

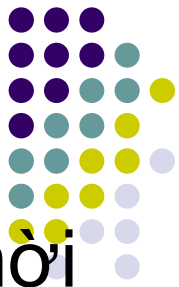
$S_2$  ;

.....

$S_n$

### COEND;

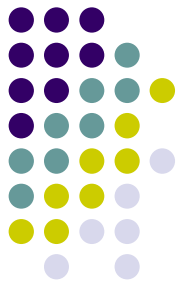
Các công việc  $S_i$  được mô tả chính xác bằng một ngôn ngữ lập trình cụ thể.



## 4 – TÀI NGUYÊN GẮNG và ĐOẠN GẮNG

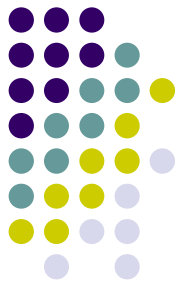
- Tài nguyên găng: Khả năng phục vụ đồng thời bị hạn chế, thông thường - bằng 1.
- Ví dụ: Máy in, quá trình bán vé máy bay . . .
- Đoạn găng (chỗ hẹp) của tiến trình,
- Điều độ tiến trình qua đoạn găng: Tổ chức cho mọi tiến trình qua được chỗ hẹp của mình.
- Giải thuật điều độ phải đảm bảo 4 yêu cầu.

# Yêu cầu



- i) Đảm bảo tài nguyên gắng không phải phục vụ quá khả năng của mình,
- ii) Không để tiến trình nằm vô hạn trong đoạn gắng,
- iii) Nếu có xếp hàng chờ thì sớm hay muộn tiến trình cũng qua được đoạn gắng,
- iv) Nếu có tiến trình chờ đợi và nếu tài nguyên gắng được giải phóng, thì tài nguyên gắng phải phục vụ ngay cho tiến trình đang chờ đợi.

# Công cụ điều độ



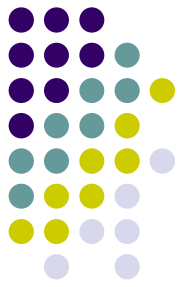
- Công cụ điều độ: 2 loại:
  - **Cấp cao**: do hệ thống đảm nhiệm, nằm ngoài tiến trình được điều độ,
  - **Cấp thấp**: cài đặt ngay vào trong tiến trình được điều độ.
- Các giải thuật điều độ cấp thấp: **3 lớp giải thuật**:
  - Phương pháp khoá trong,
  - Phương pháp kiểm tra và xác lập,
  - Kỹ thuật đèn báo.

# 5 – CÁC GIẢI THUẬT ĐIỀU ĐỘ CẤP THẤP

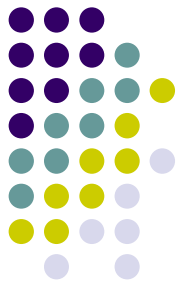


- *Phương pháp khoá trong:*
- Nguyên lý:
  - Mỗi tiến trình (TT) đặt tương ứng tài nguyên găng với 1 biến  $\in G$ ,
  - TT dùng biến này để đánh dấu việc mình đang sử dụng tài nguyên găng,
  - Trước khi vào đoạn găng TT phải kiểm tra biến tương ứng của các TT khác và chỉ vào đoạn găng khi không có TT nào đang sử dụng tài nguyên găng.

# Phương pháp khoá trong



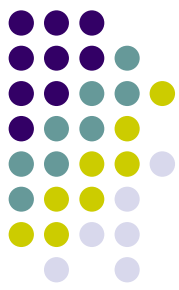
- **Môi trường ví dụ:** Xét trường hợp:
  - 2 tiến trình,
  - Mỗi TT có một đoạn găng ở đầu,
  - 1 tài nguyên găng với khả năng phục vụ:1,
  - Các tiến trình lặp vô hạn.
- **Tránh nhầm lẫn giữa 2 khái niệm:**
  - **Sơ đồ nguyên lý:** nêu ý tưởng chung,
  - **Giải thuật điều độ:** sơ đồ hành động để đảm bảo điều độ.



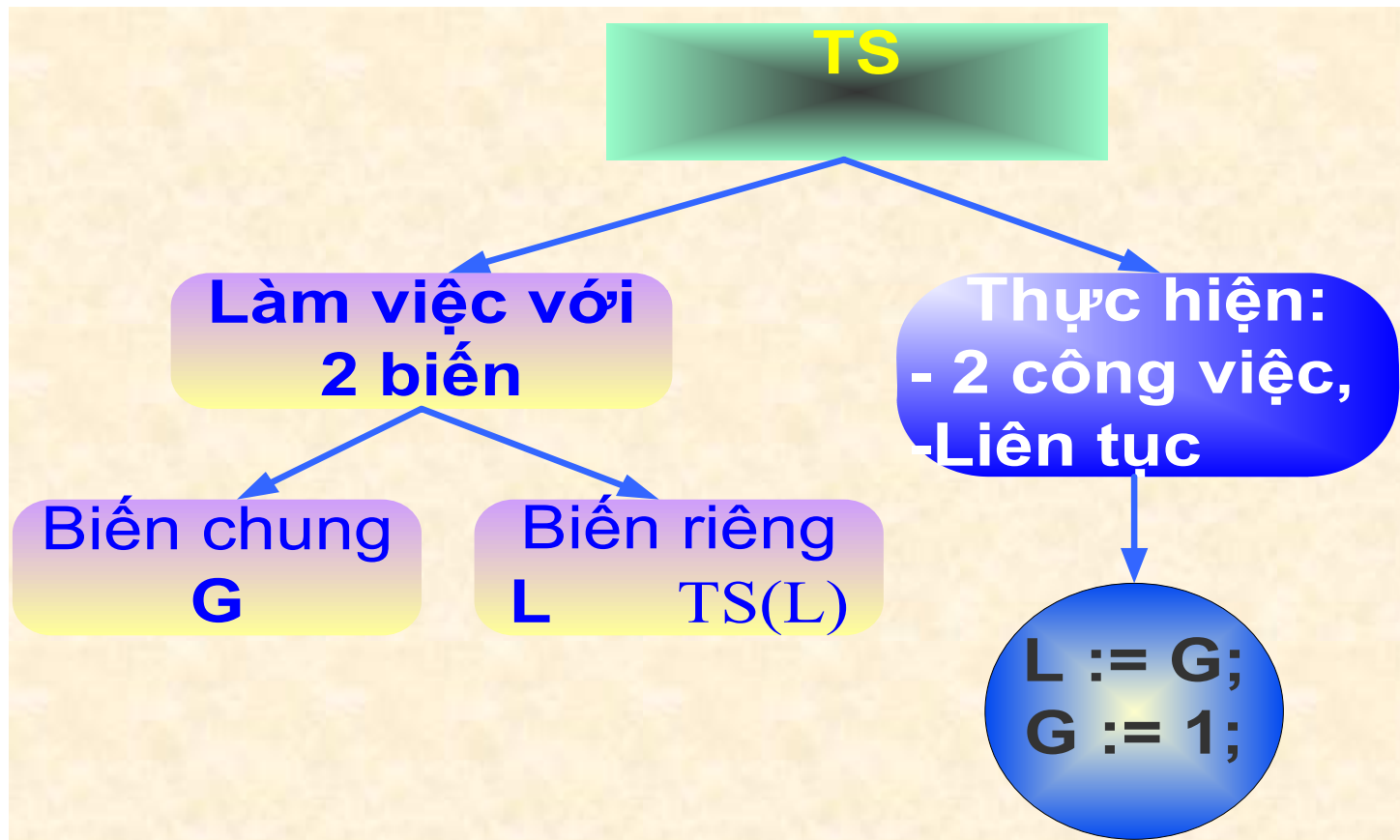
- **SEMI-NGUYÊN LÝ**

- Ban đầu  $k = 0$ ; khoá mở
- **PARBEGIN**
- tt1: while  $k = 1$  do; <-chờ đợi tích cực
- $k := 1$ ;
- <đoạn găng tt1>
- $k := 0$ ;
- <phần còn lại của tt1>
- tt2 : while  $k = 1$  do; <-chờ đợi tích cực
- $k := 1$ ;
- <đoạn găng tt2>
- $k := 0$ ;
- <phần còn lại của tt2>
- **PAREND**
- **END**

# KIỂM TRA VÀ XÁC LẬP (TEST and SET)

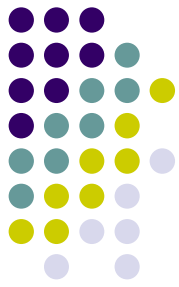


- Cơ sở: dùng lệnh máy TS có từ các máy tính thế hệ III trở đi.





# TEST and SET



- IBM 360/370:  $\exists$  1 lệnh TS ( mã 92<sub>H</sub>),
- IBM PC: Nhóm lệnh BTS (Binary Test and Set):

|     |  |  |  |  |
|-----|--|--|--|--|
| L:= |  |  |  |  |
| G:= |  |  |  |  |

# TEST and SET

- Sơ đồ điều độ:

```
Var l1, l2, g: Integer;
```

```
BEGIN
```

```
  g := 0;
```

```
  PARBEGIN
```

```
    TT1: Repeat
```

```
      l1 := 1;
```

```
      While l1 = 1 do TS(l1);
```

```
      {Đoạn găng TT1}
```

```
      g := 0;
```

```
      {Phần còn lại của TT1}
```

```
    Until false;
```

```
    TT2: Repeat
```

```
      l2 := 1;
```

```
      While l2 = 1 do TS(l2);
```

```
      {Đoạn găng TT2}
```

```
      g := 0;
```

```
      {Phần còn lại của TT2}
```

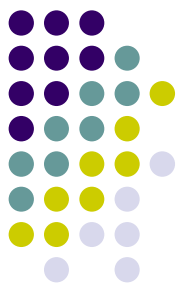
```
    Until false
```

```
  PAREND
```

```
END.
```

| g | l1 | l2 |
|---|----|----|
| 0 | 1  |    |
| 1 | 0  | 1  |
| 1 |    | 1  |
| 1 |    | 1  |
| 1 |    | 1  |
| 1 |    | 1  |
| 0 |    | 1  |
| 1 | 1  | 0  |
| 1 | 1  |    |
| 1 | 1  |    |
| 1 | 1  |    |

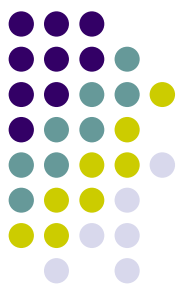
# TEST and SET



## Đặc điểm:

- Đơn giản, độ phức tạp không tăng khi số tiến trình tăng. Nguyên nhân: Cục bộ hoá biến và tính liên tục của KT & XL,
- Tồn tại hiện tượng chờ đợi tích cực. Nguyên nhân: Mỗi TT phải tự đưa mình vào đoạn găng.

# KỸ THUẬT ĐÈN BÁO(Semaphore)



- Dijkstra đề xuất 1972.
- **Đề xuất:**
  - Mỗi tài nguyên găng được đặt tương ứng với một *biến nguyên đặc biệt S* (Semaphore),
  - Ban đầu:  $S \leftarrow \text{Khả năng phục vụ t.ng. găng}$ ,
  - $\exists$  *2 lệnh máy P(S)* và *V(S)* thay đổi giá trị của S, mỗi lệnh làm *2 công việc* và làm một cách *liên tục*.



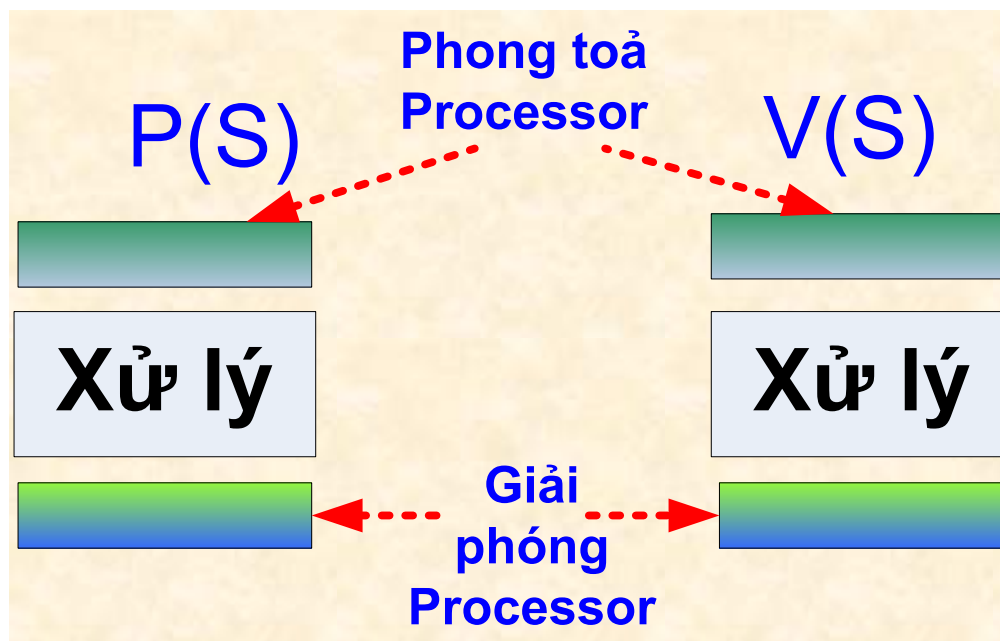
## KỸ THUẬT ĐÈN BÁO

- Nội dung lệnh P(S):
  - \* Dec(s);
  - \*\* **If  $S < 0$  then** *Đưa TT đi xếp hàng.*
- Nội dung lệnh V(S):
  - \* Inc(s);
  - \*\* **If  $S \leq 0$  then** *Kích hoạt TT đang xếp hàng.*

# KỸ THUẬT ĐÈN BẢO



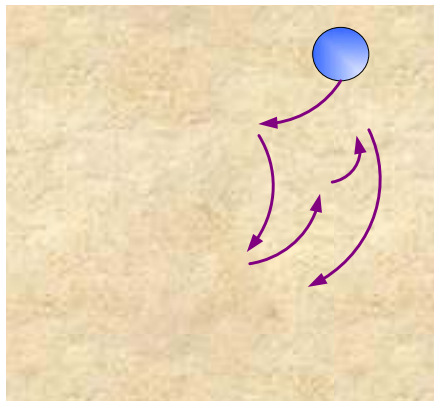
- Thực hiện:
  - Vì nhiều lý do, không thể chế tạo MT với 2 lệnh trên,
  - Lệnh  $P(S)$ ,  $V(S)$  → *thủ tục* tương ứng.
- Đảm bảo tính liên tục:



# KỸ THUẬT ĐÈN BÁO



- Sơ đồ điều độ:



```
Var s:Integer!  
BEGIN
```

```
  s := 1;
```

```
  PARBEGIN
```

```
    TT1:Repeat
```

```
      P(s);
```

```
      {Đoạn găng TT1}
```

```
      V(s);
```

```
      {Phần còn lại TT1}
```

```
    Until false;
```

```
    TT2:Repeat
```

```
      P(s);
```

```
      {Đoạn găng TT2}
```

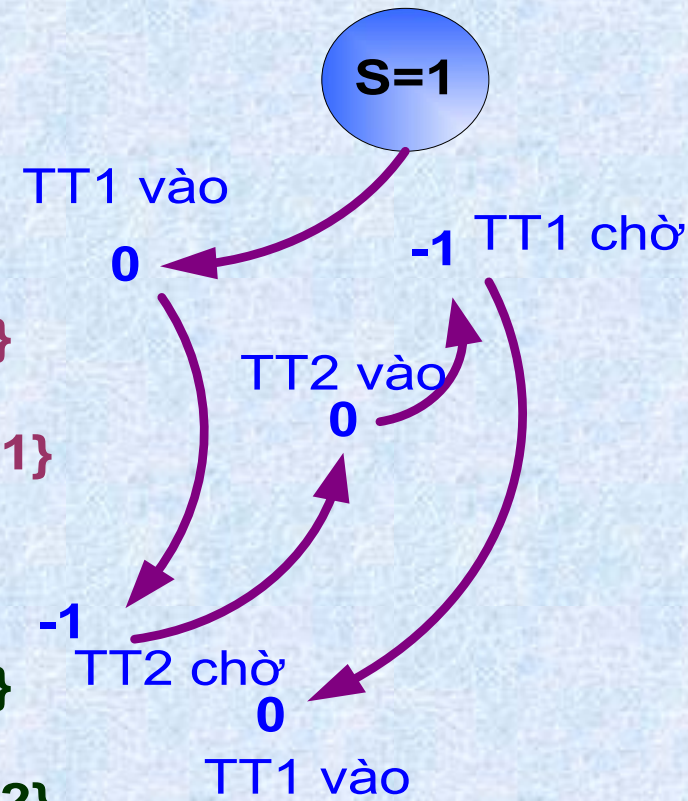
```
      V(s);
```

```
      {Phần còn lại TT2}
```

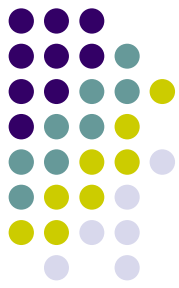
```
    Until false
```

```
  PAREND
```

```
END.
```



# KỸ THUẬT ĐÈN BÁO



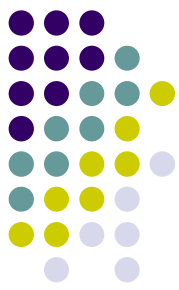
## Semaphore nhị phân:

- Phần lớn các tài nguyên gắng có khả năng phục vụ = 1  $\rightarrow$  S nhị phân.
- P(S):  
**If**  $s = 0$  **then** *Xếp\_hàng* **Else**  $s := 0$ ;
- V(S):  
**If**  $dòng\_xếp\_hàng \neq NULL$  **then** *Kích\_hoạt*  
**Else**  $s := 1$ ;

Vấn đề đặt tên các thủ tục P và V.



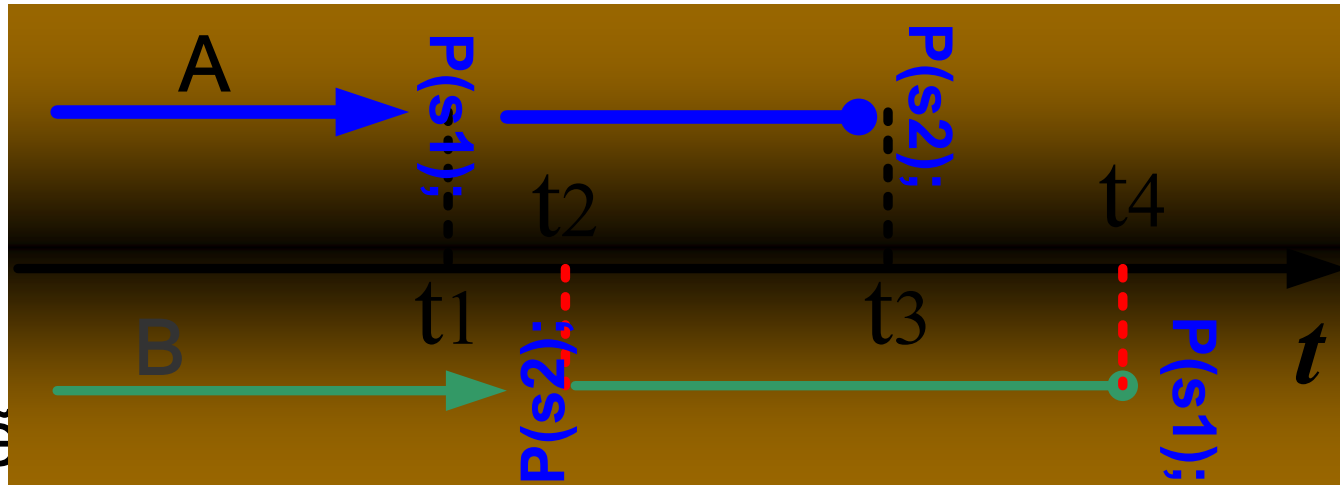
# 6 – CÔNG CỤ ĐIỀU ĐỘ CẤP CAO



- Đoạn găng quy ước,
- Biến điều kiện quy ước,
- Monitor hỗ trợ điều độ: cung cấp giá trị cho biến điều kiện quy ước.
- Monitor đóng vai trò vỏ bọc bảo vệ ngăn cách giữa tài nguyên găng và công cụ truy nhập tới nó.

# 7 - BẾ TẮC và CHỖNG BẾ TẮC

- Khái niệm bế tắc (Deadlock):
- Cùng chờ đợi,
- Vô hạn nếu không có tác động từ bên ngoài.



- Sẽ chờ đợi mãi mãi nếu không có tác động từ bên ngoài.

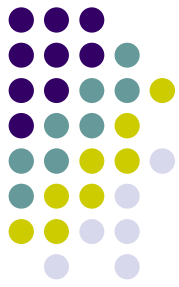
# BẾ TẮC và CHỐNG BẾ TẮC



- Điều kiện xuất hiện bế tắc: hội đủ *đồng thời* 4 điều kiện:
  - $\exists$  tài nguyên găng,
  - Có tổ chức xếp hàng chờ đợi,
  - Không phân phối lại tài nguyên,
  - $\exists$  hiện tượng chờ đợi vòng tròn.
- Chống bế tắc: *3 lớp giải thuật*:
  - Phòng ngừa,
  - Dự báo và tránh,
  - Nhận biết và khắc phục.



- Điều kiện áp dụng:
  - Xác xuất xuất hiện bế tắc lớn,
  - Các biện pháp Tồn thất lớn.
- Biện pháp: tác động lên một hoặc một số điều kiện gây bế tắc để 4 điều kiện không xuất hiện đồng thời.
- Các giải pháp: được áp dụng để *nâng cao hiệu quả* của hệ thống.

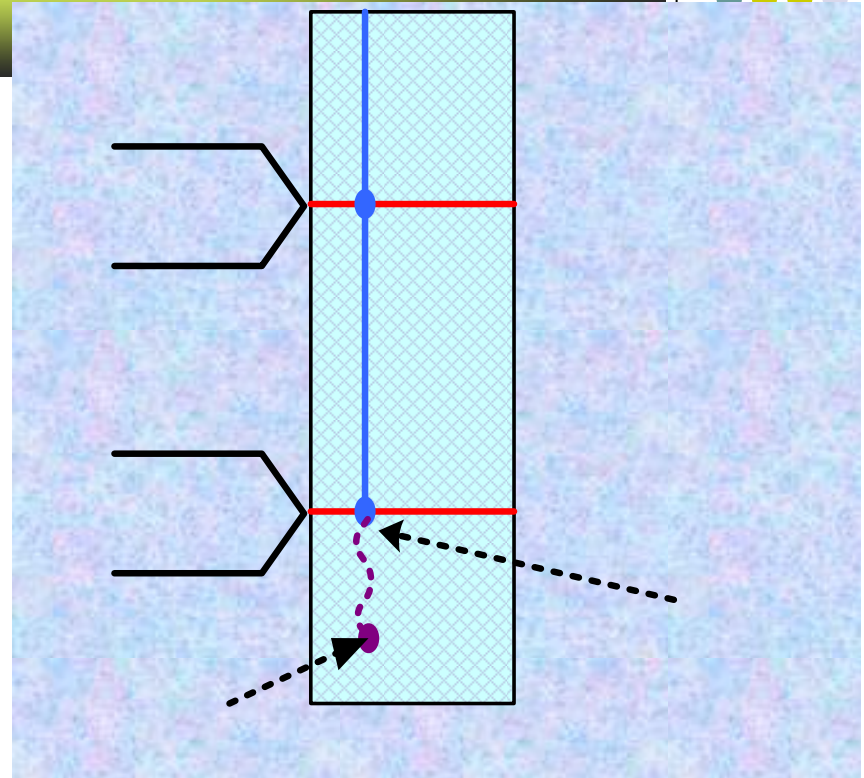


## Phòng ngừa

- Chống tài nguyên găng:
  - Tổ chức hệ thống tài nguyên lô gíc,
  - 2 mức truy nhập,
  - SPOOL.
- Chống xếp hàng chờ đợi:
  - Chế độ phân phối sơ bộ,
  - Trước khi ngắt TT: lưu trạng thái (Dump),
  - Công cụ:
    - Điểm gác (Control Points),
    - Điểm ngắt (Break Points)

# Phòng ngừa

- Đặt điểm gác:
  - Cố định trong CT,
  - Theo tác nhân ngoài (vd: thời gian)
- Ứng dụng:
  - Hiệu chỉnh CT,
  - Thực hiện các CT dài,
  - Với toàn bộ hệ thống: Hibernating.





- Phân phối lại tài nguyên:
  - Các tài nguyên quan trọng (Bộ nhớ, Processor . . .) luôn luôn được phân phối lại,
  - Chủ yếu: chỉ cần lưu ý các tài nguyên riêng,
  - Hệ thống tài nguyên lô gíc: giảm nhu cầu phân phối lại.
  - Để phân phối lại: Lưu và khôi phục trạng thái tài nguyên.

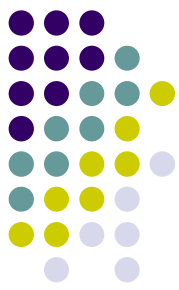
# Phòng ngừa



- Chống chờ đợi vòng tròn:
  - Phân lớp tài nguyên, tạo thành hệ thống phân cấp,
  - Nguyên tắc phân phối: Khi chuyển lớp - phải giải phóng tài nguyên lớp cũ.

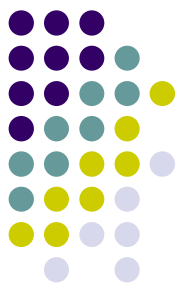


# DỰ BÁO VÀ TRÁNH



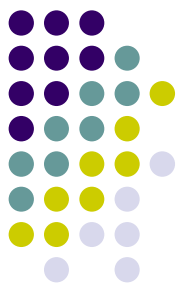
- Mỗi lần phân phối một tài nguyên: kiểm tra xem việc phân phối này có thể dẫn đến *nguy cơ* bế tắc cho một số tiến trình nào đó hay không và là những tiến trình nào?
- Điều kiện môi trường:
  - Xác xuất xảy ra bế tắc nhỏ,
  - Tổn thất (nếu có bế tắc) – lớn.

# DỰ BÁO VÀ TRÁNH



- Giải thuật tiêu biểu: “*Người chủ ngân hàng*”.
- Giả thiết:
  - Xét 1 loại tài nguyên, số lượng  $\rightarrow$  tstb,
  - n tiến trình,
  - $\text{Max}_i$ ,
  - $\text{Ffoi}_i$ ,
  - $\text{Kt}_i$  – boolean,
    - True – chắc chắn kết thúc được,
    - False – trong trường hợp ngược lại.

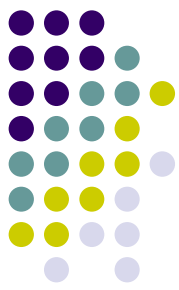
# DỰ BÁO VÀ TRÁNH



```
ts := tstb;  
{Thông kê}  
For i := 1 to n do  
begin  
    clai[i] := max[i] - ffoi[i];  
    ts := ts - ffoi[i];  
    kt[i] := false  
end;
```

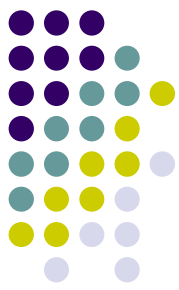
```
{Đánh giá}  
Flag := true;  
While flag do  
begin flag := false;  
    for i := 1 to n do  
        if not kt[i] and (clai[i] <= ts) then  
            begin  
                kt[i] := true;  
                ts := ts + ffoi[i];  
                flag := true  
            end  
    end;  
end;  
If ts <> tstb then Kh_An_Toan;
```

# DỰ BÁO VÀ TRÁNH



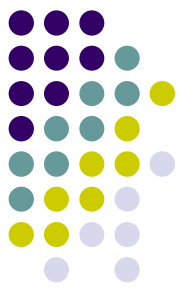
- Tiêu chuẩn dự báo: ngặt,
- Dựa vào  $Kt_i \rightarrow$  biết các TT có nguy cơ bế tắc,
- Xử lý trước khi TT bị bế tắc.
- Đặc điểm giải thuật:
  - Đơn giản,
  - Input:  $Max_i$  – tin cậy,
  - Mỗi loại tài nguyên  $\Leftrightarrow$  thủ tục,
  - Mỗi lần phân phối  $\rightarrow$  kiểm tra.

# NHẬN BIẾT VÀ KHẮC PHỤC



- Định kỳ kiểm tra các TT chờ đợi để phát hiện bế tắc,
- Điều kiện áp dụng:
  - Xác xuất xảy ra bế tắc bé,
  - Tổn thất (nếu có bế tắc) – bé.
- Áp dụng với phần lớn OS trong thực tế,
- Do OP đảm nhiệm.

# NHẬN BIẾT VÀ KHẮC PHỤC



- Lệnh OP → các nhóm lệnh phục vụ nhận biết và khắc phục,
- Nhóm lệnh xem trạng thái (Display Status),
- Nhóm lệnh tác động lên dòng xếp hàng TT,
- Nhóm lệnh tác động lên TT,
- Quan trọng: các lệnh huỷ tiến trình,
- Các biện pháp hỗ trợ và ngăn chặn tự động.

# 8 - GỌI TIỀN TRÌNH



- TT có thể cạnh tranh hoặc tương tác với nhau,
- Mỗi quan hệ tương tác: tuần tự hoặc song song,
- Xác lập quan hệ:
  - Lời gọi,
  - Cơ chế xử lý sự kiện (Sẽ xét ở chương sau),
- Các cách gọi:
  - Trong phạm vi một hệ thống,
  - Giữa các hệ thống:
    - RI (Remote Invocation),
    - RPC (Remote Procedure Call),
  - Lý thuyết chung: RMI (Remote Methods Invocation)

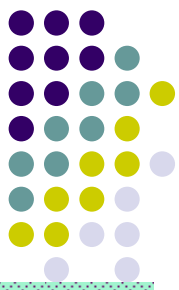
# GỌI TIỀN TRÌNH



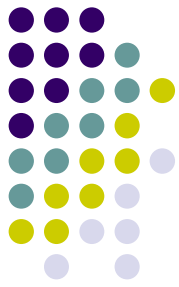
- Sơ đồ gọi:
  - Không đối xứng,
  - Đối xứng.
- Kỹ thuật truyền tham số:
  - Theo giá trị,
  - Theo địa chỉ,
  - CR (Call by Copy/Restore).



# GỌI TIỀN TRÌNH



- Thông tin tối thiểu để lưu và khôi phục TT:
  - Nội dung các thanh ghi,
  - Địa chỉ lệnh,
  - Vùng bộ nhớ RAM liên quan,
  - Vùng bộ nhớ phục vụ của hệ thống,
  - Các sự kiện chưa xử lý.
- Phân biệt sơ đồ gọi đối xứng và đệ quy.

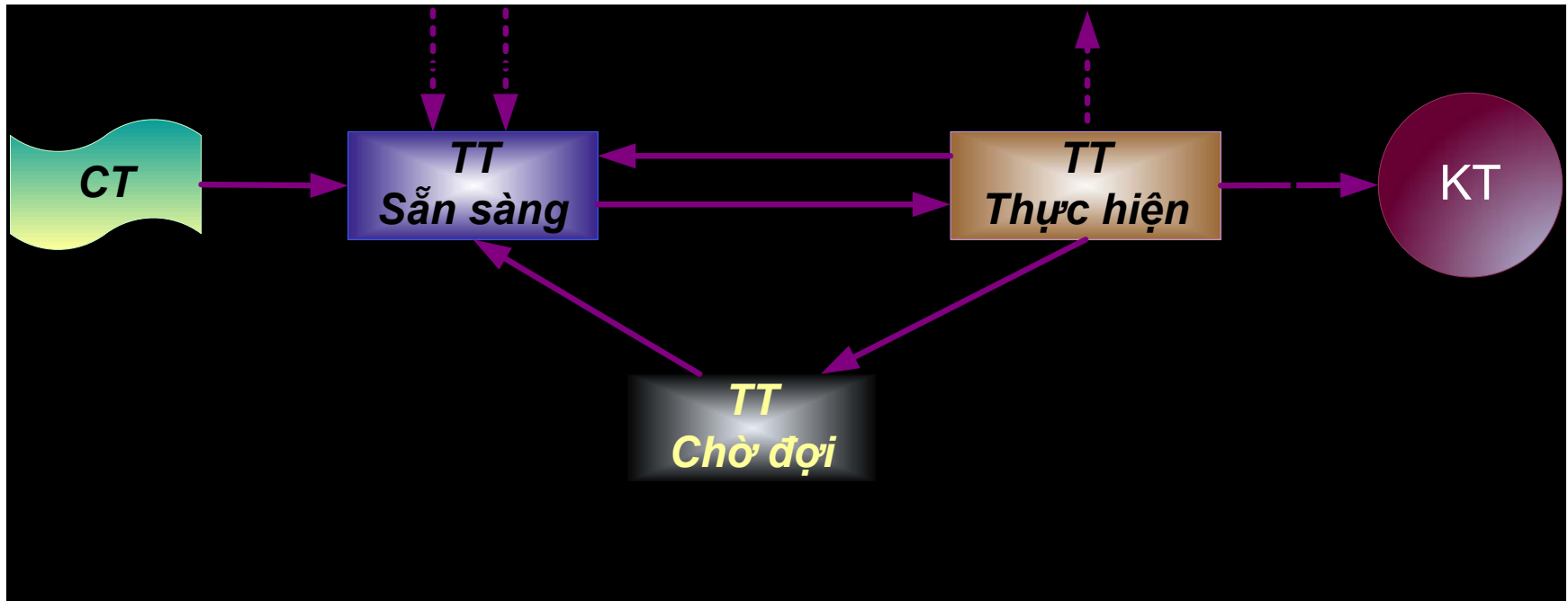
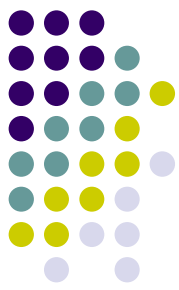


# V – QUẢN LÝ PROCESSOR

- Mục đích: Giảm thời gian chết của Processor  
→ nâng cao hiệu quả hệ thống,
- Vai trò thiết bị trung tâm: liên kết các bộ phận độc lập (cứng và mềm) thành hệ thống hoạt động đồng bộ.
- Trong phần này: xét hoạt động của 1 CPU.

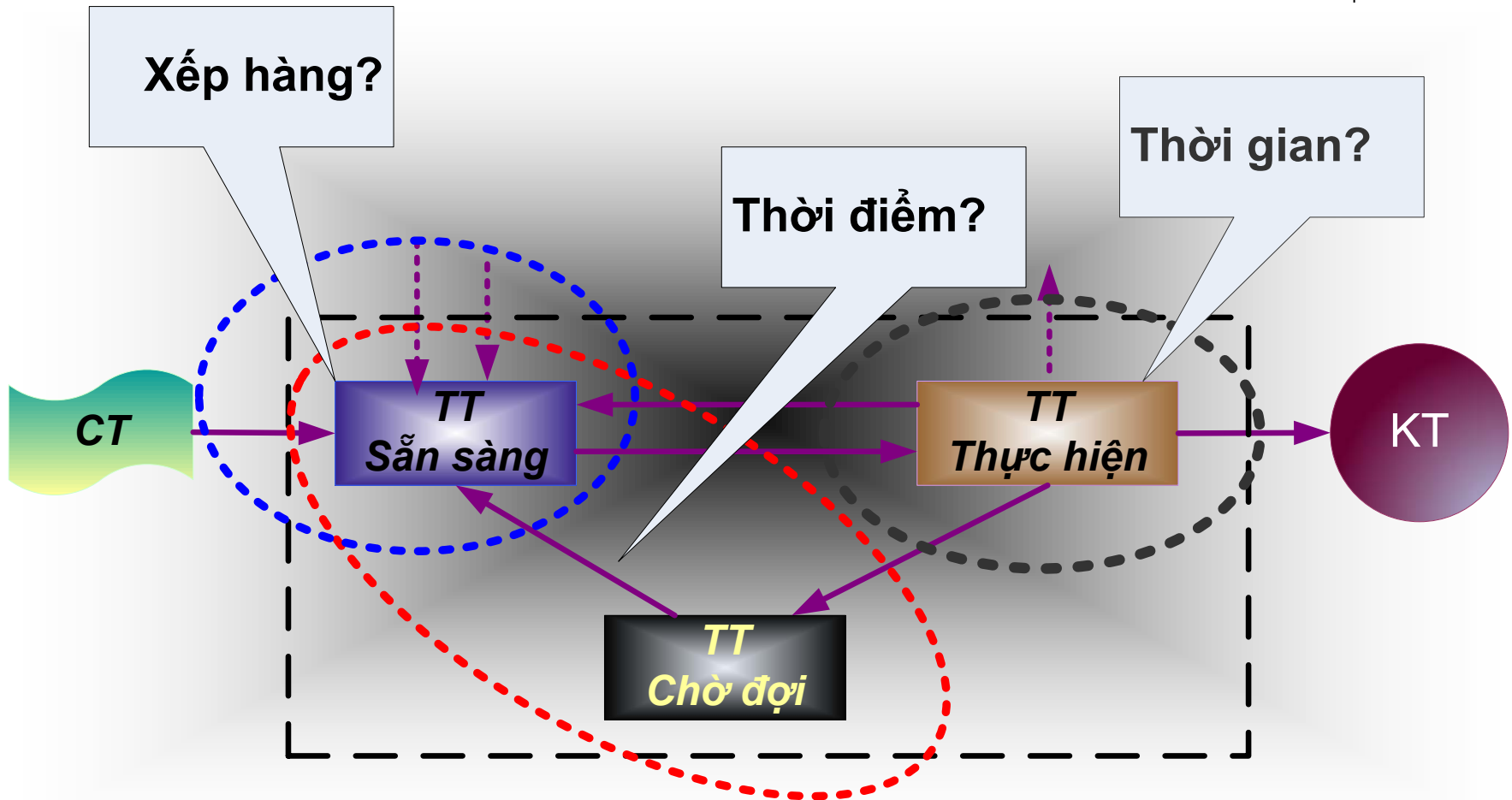


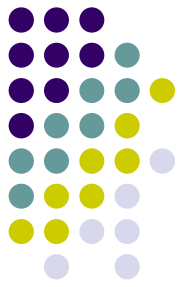
## 2 – CÁC TRẠNG THÁI CƠ BẢN CỦA TIỀN TRÌNH



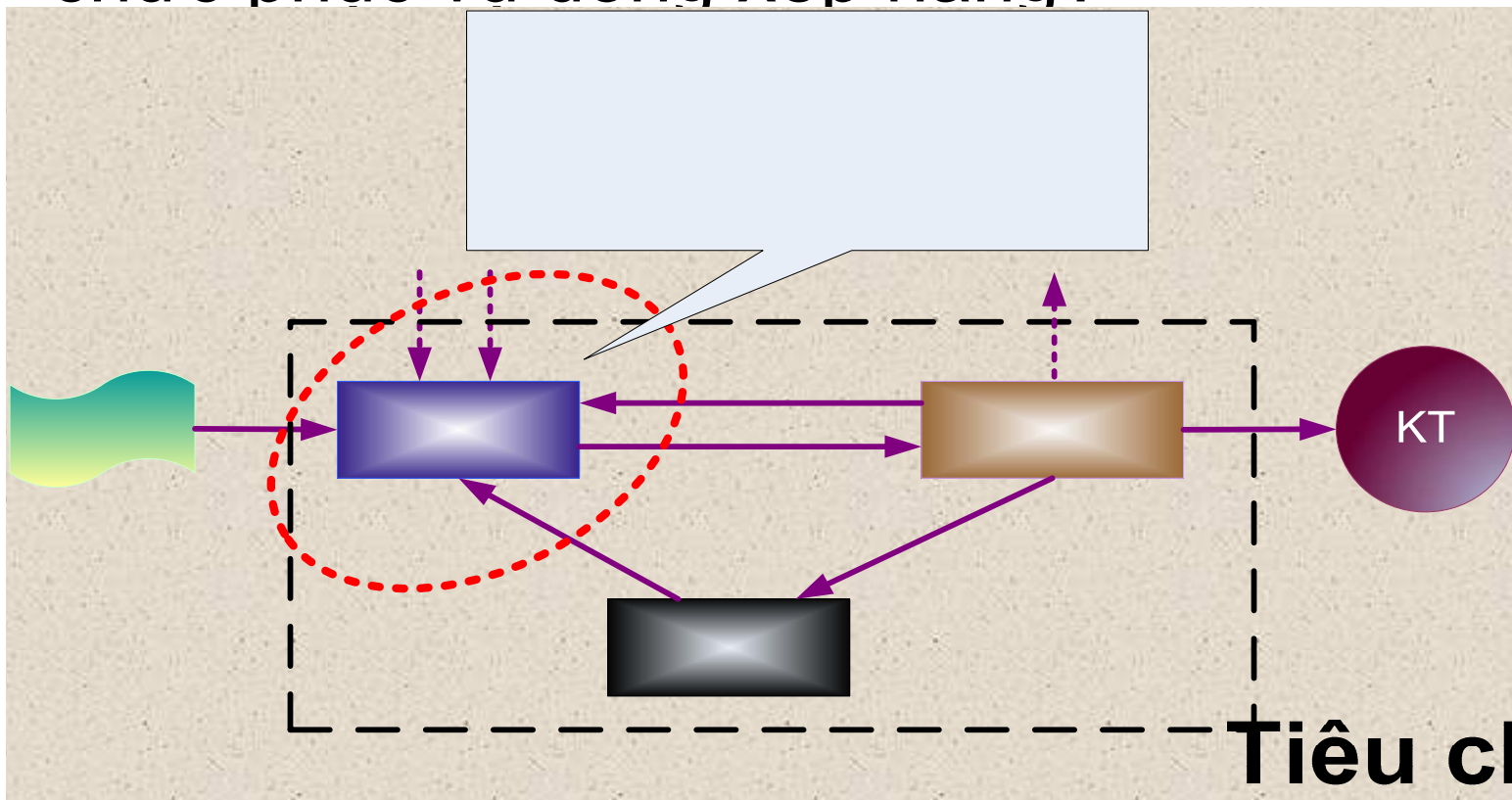
- Đặc trưng các loại trạng thái,
- Vấn đề cần giải quyết: 3 loại.

# VẤN ĐỀ





a) Liên quan tới dòng TT sẵn sàng: Cách tổ chức phục vụ dòng xếp hàng?



## A decorative graphic consisting of a grid of colored dots in the bottom right corner. The dots are arranged in a roughly triangular shape, with colors including dark purple, teal, yellow, and light purple. The colors transition from dark purple on the left to light purple on the right, with teal and yellow in the middle. The dots are of varying sizes and are scattered across the bottom right area of the slide.

- |  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |

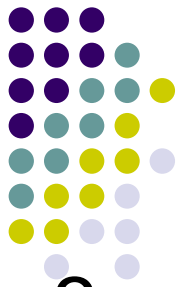
199

## A decorative graphic consisting of a grid of colored dots in the bottom right corner. The dots are arranged in a roughly triangular shape, with colors including dark purple, teal, yellow, and light purple. The colors transition from dark purple on the left to light purple on the right, with teal and yellow in the middle.

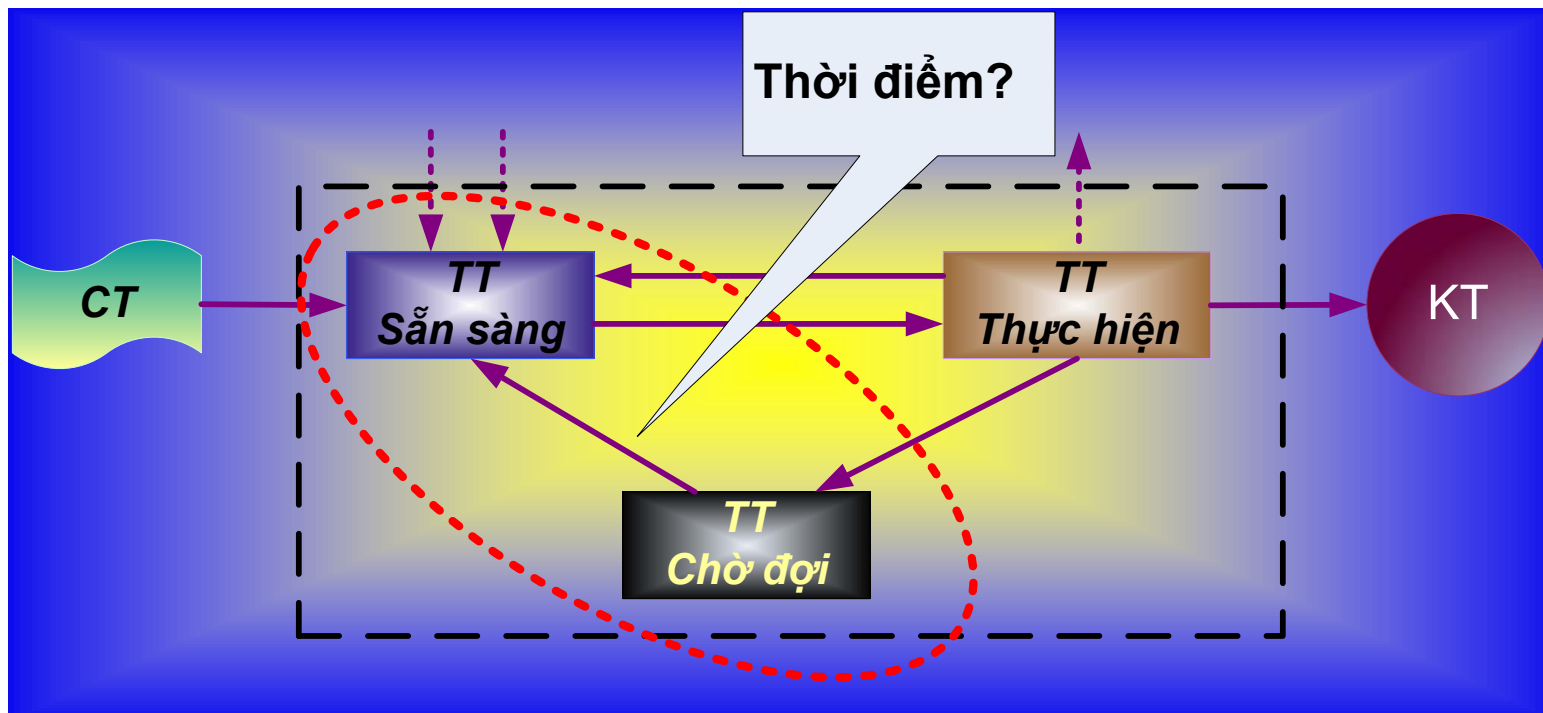
- 
- 2 chế độ phục vụ
- TT  
Thực hiện
- KT
- TT  
Chờ đợi

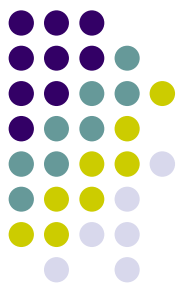


# VẤN ĐỀ



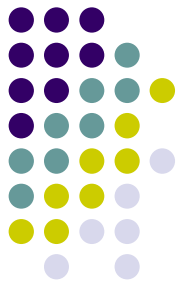
- c) Thời điểm đưa TT chờ đợi trở lại sẵn sàng?  
Cơ chế *sự kiện* và *ngắt*.





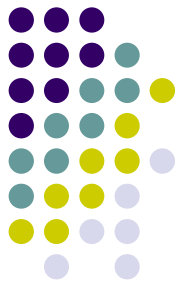
- TT  $\Leftrightarrow$  thứ tự ưu tiên phục vụ,
- Yêu cầu:
  - $t_w \rightarrow \min.$
  - TT kết thúc.
- Chế độ:
  - Một dòng xếp hàng,
  - Nhiều dòng xếp hàng.

# Chế độ một dòng xếp hàng



- a) FCFS (First come – First served):
  - Đơn giản,
  - $\forall$  TT kết thúc được,
  - Không cần input bổ sung,
  - $T_w$  – lớn,
  - Non-Preemptive.

# Chế độ một dòng xếp hàng



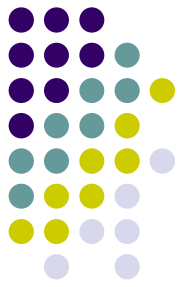
- b) SJN (Shortest Job – Next):
  - Thời gian thực hiện ít → ưu tiên cao,
  - $T_w$  giảm,
  - TT dài có nguy cơ không kết thúc được,
  - Khó dự báo thời điểm phục vụ TT,
  - Non-Preemptive,
  - Input: Thời gian thực hiện TT.

# Chế độ một dòng xếp hàng

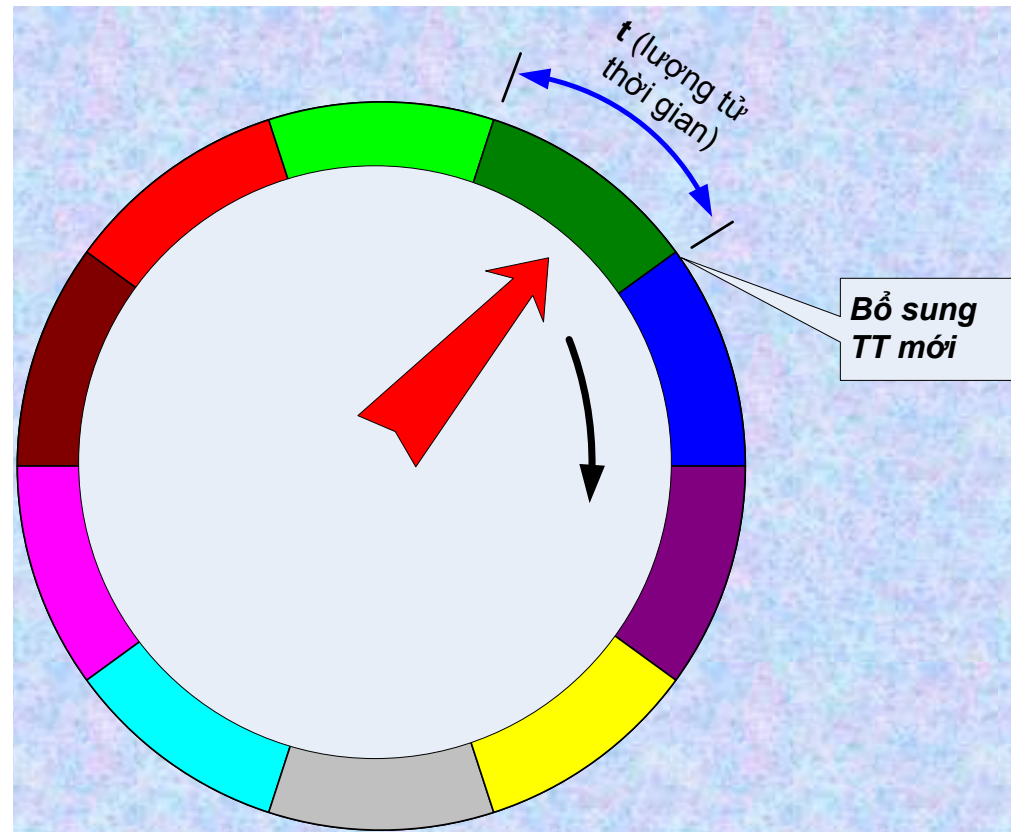


- c) SRT (Shortest Remaining Time):
  - Thứ tự ưu tiên phục vụ: xác định theo lượng thời gian còn lại cần thiết để kết thúc TT,
  - $t_w$  giảm mạnh,
  - Các đặc trưng khác: tương tự như SJN,
  - TT dài càng có nguy cơ không kết thúc được!
- Ở các chế độ Non-Preemptive: cần có  $t_{lim}$  → huỷ TT hoặc đưa về thứ tự ưu tiên thấp nhất.

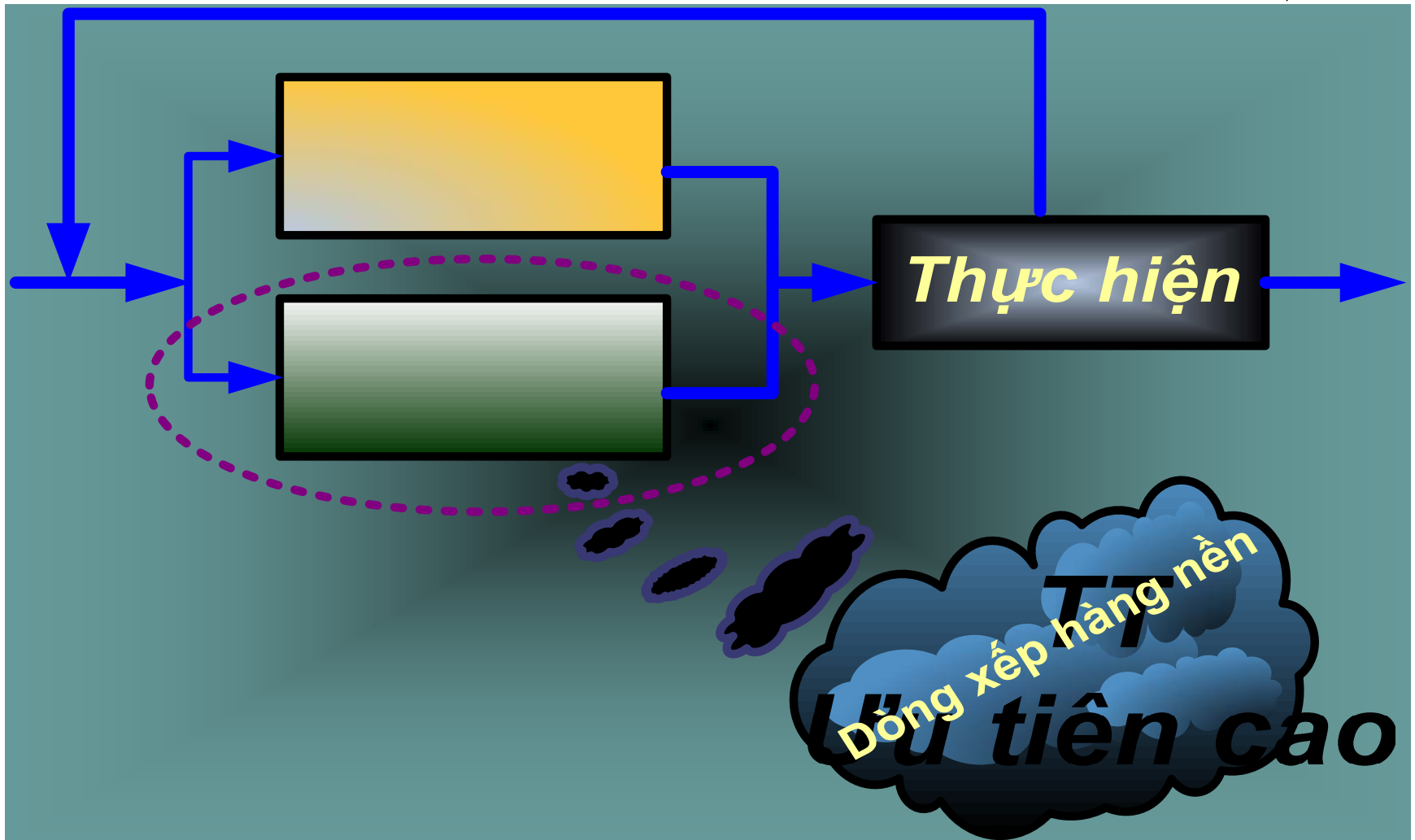
# Chế độ một dòng xếp hàng



- d) RR (Round Robin):
  - Preemptive,
  - $\forall$  TT - kết thúc được,
  - Khả năng đối thoại với TT,
  - Ưu tiên thích đáng với TT dài: phân lớp phục vụ với  $t$  lớn hơn.



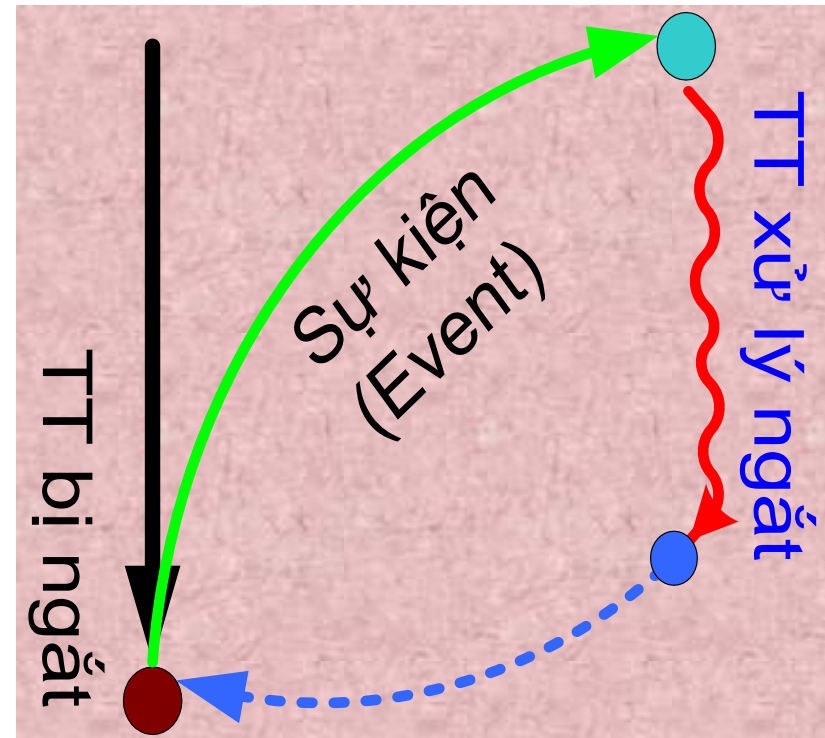
# Chế độ nhiều dòng xếp hàng



# 4 - NGẮT và XỬ LÝ NGẮT

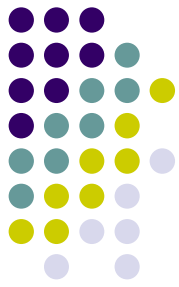


- Định nghĩa ngắt (Interrupt):
  - Cơ chế *Sự kiện* và *Ngắt*: từ MT thế hệ III,
  - IBM 360/370 – 7 loại sự kiện,
  - IBM PC – 256 loại sự kiện.

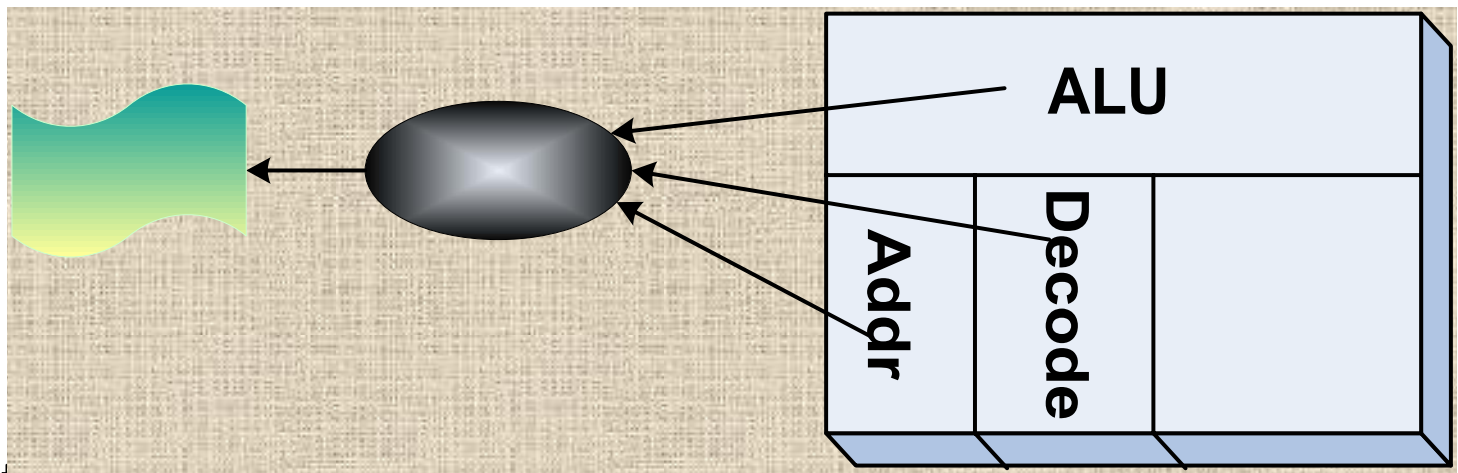




# PHÂN LOẠI NGẮT



- Ngắt trong và ngắt ngoài,

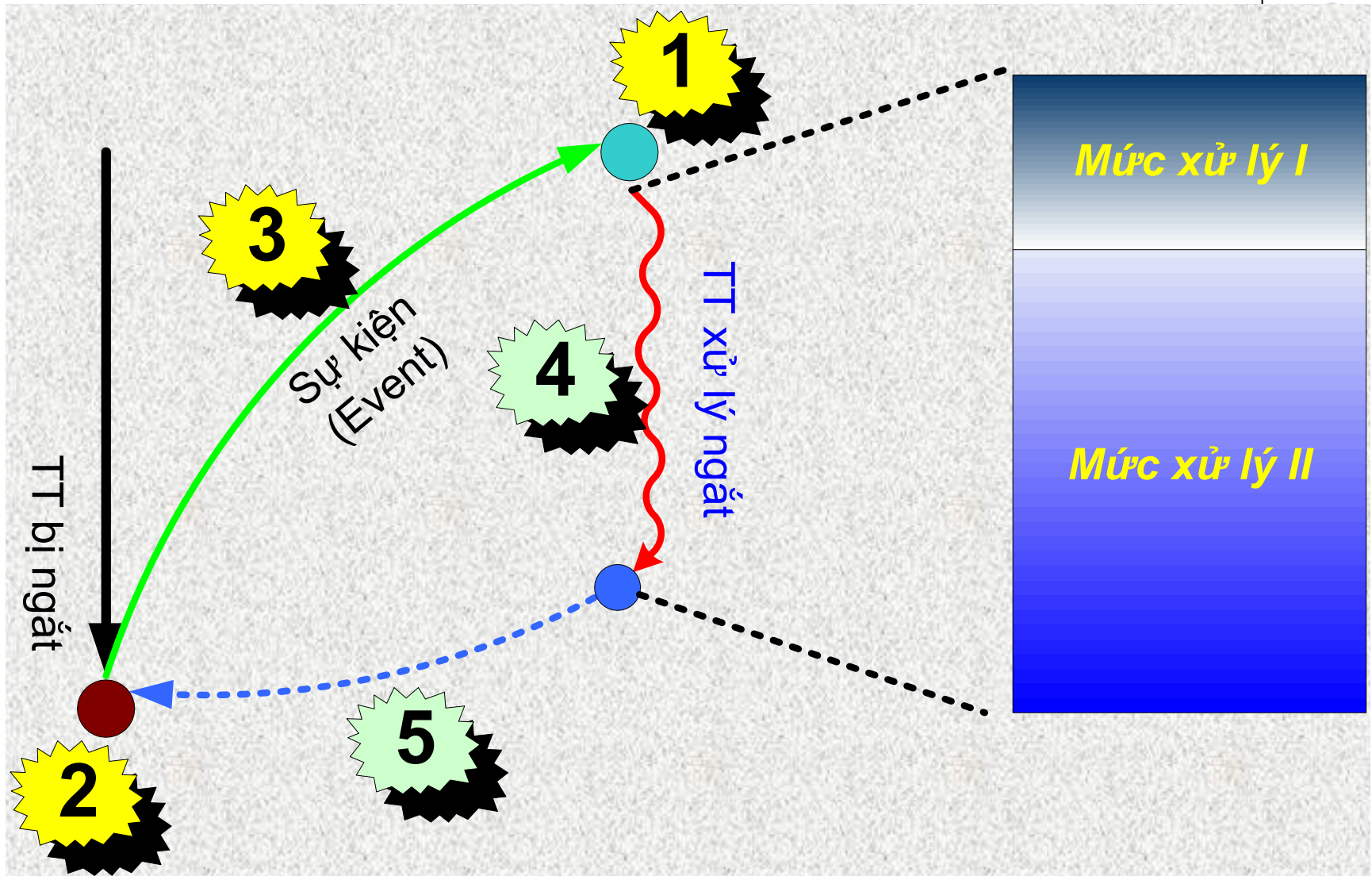


- Ngắt trong: /0, tran 0, . . .
- Ngắt ngoài: I/O Int, Timer, . . .
- Ngắt chặn được và không chặn được:
  - Chặn được: i/o Int,
  - Không chặn được: Timer Int.
- Ngắt cứng và ngắt mềm.

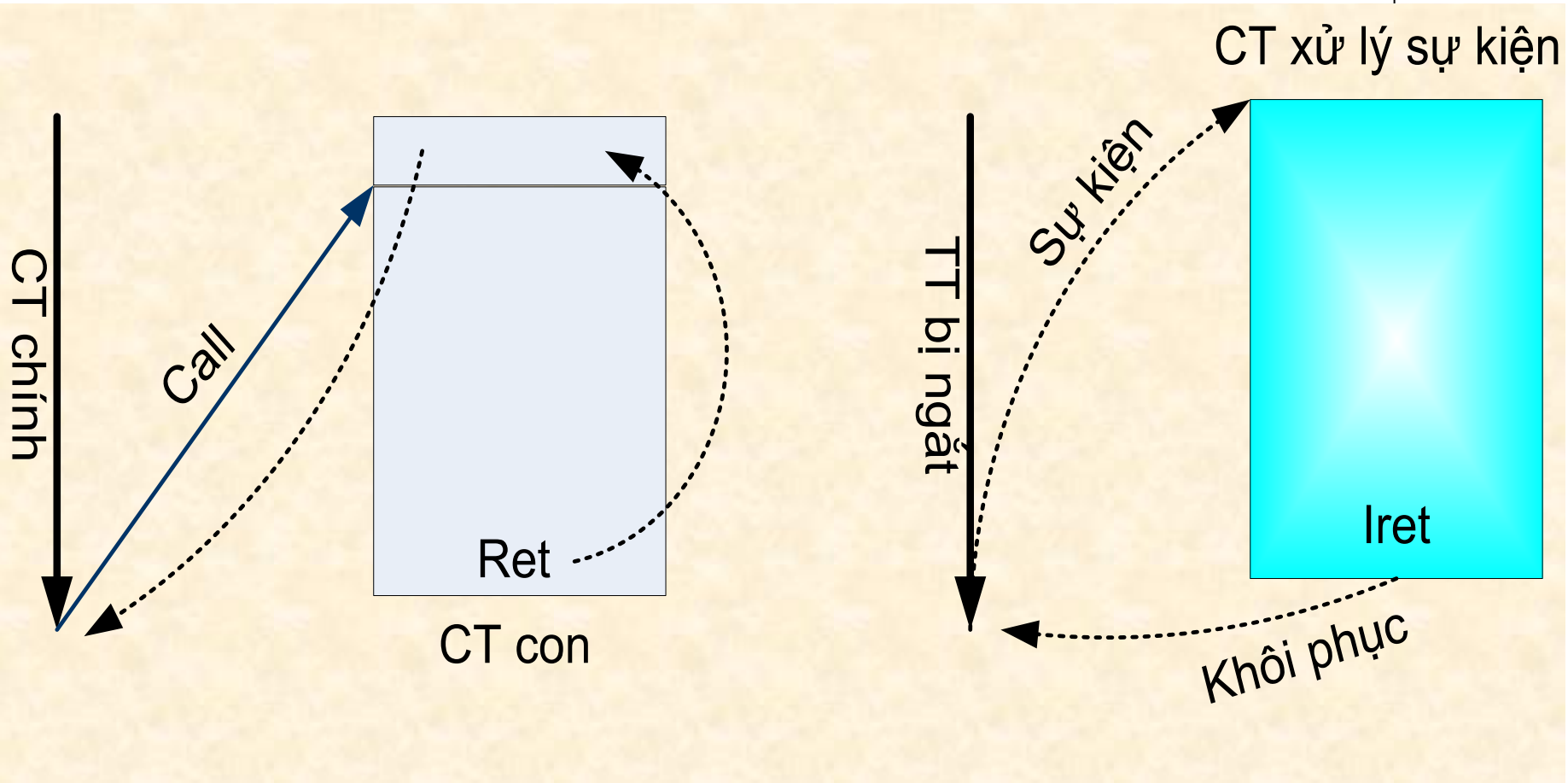
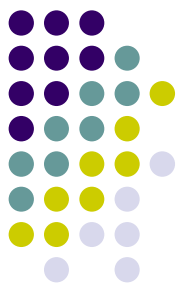
**Ngắt  
trong**

Sự kiện

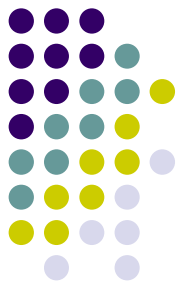
# XỬ LÝ NGẮT



# CT con và CT xử lý ngắt

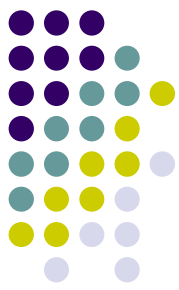


# 5 - Xử lý ngắt trong IBM PC



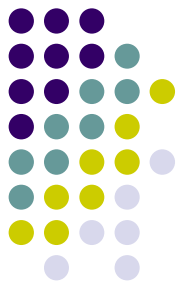
- Ngắt  $\Leftrightarrow$  Pointer (4 bytes),
- Véc tơ ngắt = {Pointers} (1 KB),
- Khối bộ nhớ xử lý ngắt,
- Nét đặc biệt:
  - $\exists$  các ngắt | Pointer  $\rightarrow$  Bảng tham số (Int 11, 1E, 41, ...),
  - Ngắt KT CT – Int 20, Ngắt thường trú CT Int 27,
  - Ngắt R/W đĩa theo địa chỉ tuyệt đối – Int 25, 26,
  - $\exists$  ngắt tương ứng với việc bấm phím (Int 05, 1B),
  - Ngắt OS mô phỏng xử lý các sự kiện (Int 21),
  - Một số sự kiện: dành cho user tạo ngắt mềm  $\rightarrow$  Lập trình hưởng sự kiện (EOP).

# VI - CẤU HÌNH và QUẢN LÝ HỆ THỐNG



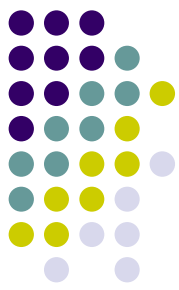
- 1 - Hệ thống nhiều Processors.
- Các loại cấu hình:
  - Cấu hình phân cấp,
  - Liên kết linh hoạt,
  - Đẳng cấu,
- Quản lý tiến trình:
  - S – tài nguyên găng,
  - $TS \rightarrow S \rightarrow$  điều độ,
- Đảm bảo toàn vẹn chức năng và toàn vẹn cấu hình.

# CẤU HÌNH và QUẢN LÝ HỆ THỐNG



- 2 - Bảo vệ hệ thống:
- Nguy cơ:
  - Mất hoặc hỏng dữ liệu,
  - Sử dụng tài nguyên với mục đích xấu,
  - Truy nhập không đăng ký,
  - Dò rỉ thông tin.
- Cơ chế bảo vệ:
  - Nguyên lý ngăn chặn,
  - Nguyên lý cho phép.
- Giải thuật và biện pháp bảo vệ: linh hoạt, thường xuyên thay đổi.

# CẤU HÌNH và QUẢN LÝ HỆ THỐNG



- 3 – Thiết kế và xây dựng hệ thống:
- Nguyên lý tập trung: WINDOWS, UNIX, OS IBM, . . .
- Nguyên lý “Thử và sai”: LINUX:
  - Không có đề xuất hướng chung,
  - Mã nguồn mở cho phép mọi người nghiên cứu, bổ sung sửa đổi,
  - Phát triển theo nguyên lý tự điều chỉnh,
  - Giao diện: User tự trang bị.

# 4 - Hệ thống của Microsoft

