

# BAN HỌC TẬP CÔNG NGHỆ PHẦN MỀM

TRAINING CUỐI KỲ HỌC KỲ II NĂM HỌC 2022 – 2023



**Sharing is learning**



 **BAN HỌC TẬP**

*Khoa Công nghệ Phần mềm*

*Trường Đại học Công nghệ Thông tin*

*Đại học Quốc gia thành phố Hồ Chí Minh*

 **CONTACT**

*bht.cnpm.uit@gmail.com*

*fb.com/bhtcnpm*

*fb.com/groups/bht.cnpm.uit*

# TRAINING

# HỆ ĐIỀU HÀNH

⌚ Thời gian: 19:30 thứ 6 ngày 23/06/2023

📍 Địa điểm: Microsoft Teams

👤 Trainers: Phạm Long Vũ – MTCL2021

Đặng Phước Sang – KHTN2021

Phạm Thái Bảo – ATCL2021



Sharing is learning

# Chương 5: Đồng bộ



**Sharing is learning**

# Chương 5: Đồng bộ

## Tại sao phải đồng bộ?

- Khảo sát các process/thread thực thi đồng thời và chia sẻ dữ liệu (qua shared memory, file).
- Nếu không có sự kiểm soát khi truy cập các dữ liệu chia sẻ thì có thể đưa đến ra trường hợp không nhất quán dữ liệu (data inconsistency).
- Để duy trì sự nhất quán dữ liệu, hệ thống cần có cơ chế bảo đảm sự thực thi có trật tự của các process đồng thời.

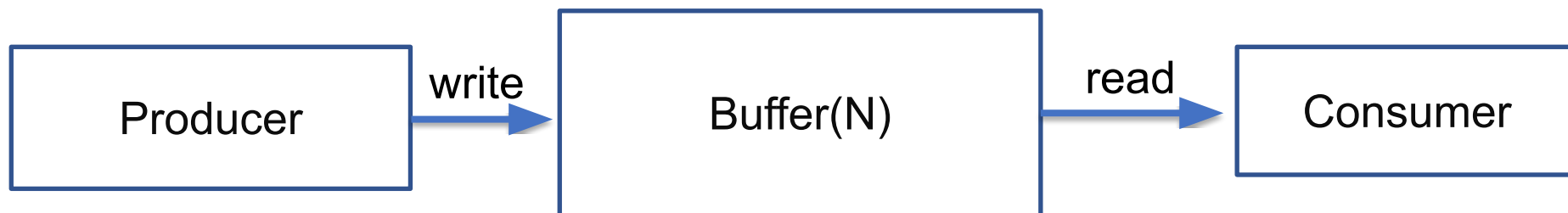


Sharing is learning

# Chương 5: Đồng bộ

## Ví dụ: Producer - Consumer

- P không được ghi dữ liệu vào buffer đã đầy
- C không được đọc dữ liệu từ buffer đang trống
- P và C không được thao tác trên buffer cùng lúc



Sharing is learning

# Chương 5: Đồng bộ

## Ví dụ: Producer - Consumer

### - Quá trình Producer

```
item nextProduce;  
while(1){  
    while(count == BUFFER_SIZE); /*ko lam gi*/  
    buffer[in] = nextProducer;  
    count++;  
    in = (in+1)%BUFFER_SIZE; }
```

### - Quá trình Consumer

```
item nextConsumer;  
while(1){  
    while(count == 0); /*ko lam gi*/  
    nextConsumer = buffer[out];  
    count--;  
    out = (out+1)%BUFFER_SIZE; }
```



Sharing is learning



# Chương 5: Đồng bộ

## Vùng tranh chấp (Critical Section):

- Trong mỗi process có những đoạn code có chứa các thao tác lên dữ liệu chia sẻ. Đoạn code này được gọi là vùng tranh chấp (critical section, CS).
- Vấn đề Critical Section: phải bảo đảm sự loại trừ tương hỗ (mutual exclusion, mutex), tức là khi một process đang thực thi trong vùng tranh chấp, không có process nào khác đồng thời thực thi các lệnh trong vùng tranh chấp.



Sharing is learning

# Chương 5: Đồng bộ

## Yêu cầu của lời giải cho CS Problem:

- Lời giải phải thỏa ba tính chất:
  - (1) Loại trừ tương hỗ (Mutual exclusion): Khi một process P đang thực thi trong vùng tranh chấp (CS) của nó thì không có process Q nào khác đang thực thi trong CS của Q.
  - (2) Progress: Một tiến trình tạm dừng bên ngoài miền giảng hông được ngăn cản các tiến trình khác vào miền giảng
  - (3) Chờ đợi giới hạn (Bounded waiting): Mỗi process chỉ phải chờ để được vào vùng tranh chấp trong một khoảng thời gian có hạn định nào đó. Không xảy ra tình trạng đói tài nguyên (starvation).



Sharing is learning



# Chương 5: Đồng bộ

## Phân loại giải pháp:

### Busy-waiting

- Tiếp tục tiêu thụ CPU trong khi chờ đợi vào CS
- Không đòi hỏi sự trợ giúp của Hệ điều hành

**While (chưa có quyền) do nothing() ;**

CS;

**Từ bỏ quyền sử dụng CS**

- Sử dụng các biến cờ hiệu
- Sử dụng việc kiểm tra luân phiên
- Giải pháp của Peterson
- Cấm ngắt
- Chỉ thị TSL

### Sleep & Wake up

- Từ bỏ CPU khi chưa được vào CS
- Cần Hệ điều hành hỗ trợ

**if (chưa có quyền) Sleep() ;**

CS;

**Wakeup (somebody);**

- Semaphore
- Monitor
- Message



Sharing is learning

# Chương 5: Đồng bộ

**Câu 1: Lí do giải thuật semaphore cần sự hỗ trợ từ hệ điều hành?**

- A. Khi 1 tiến trình đang ở trạng thái chờ thì trạng thái đó được đưa vào trạng thái wakeup
- B. Để tránh việc tiến trình đó sử dụng quá nhiều CPU
- C.** Khi 1 tiến trình ở trạng thái chờ thì đưa tiến trình đó vào trạng thái sleep
- D. Để tăng khả năng xử lý của CPU.



Sharing is learning

## Chương 5: Đồng bộ

**Câu 16: Nhóm giải pháp đồng bộ Sleep And Wakeup có đặc điểm nào dưới đây ?**

A. Tiến trình không có quyền từ bỏ CPU khi chưa được vào vùng tranh chấp

B. Không đảm bảo được vấn đề loại trừ tương hỗ

**C.** Tiến trình rời khỏi vùng tranh chấp sẽ đánh thức tiến trình đã từ bỏ CPU trước đó (nếu có)

D. Được chia thành hai loại Phần mềm và Phần cứng



Sharing is learning

# Chương 5: Đồng bộ

**Câu 11: Câu nào sau đây sai ?**

- A. Loại trừ tương hỗ (Mutual Exclusion – Mutex) là khi một process P đang thực thi trong vùng tranh chấp (CS) của nó thì một process Q nào đó không được thực thi được trong vùng tranh chấp (CS) của Q
- B. Giải thuật kiểm tra luân phiên thỏa mãn tính chất loại trừ tương hỗ.
- C.** Một tiến trình tạm dừng bên ngoài vùng tranh chấp ngăn cản các tiến trình khác vào vùng tranh chấp
- D. Nhóm giải pháp Busy Waiting không đòi hỏi sự hỗ trợ từ hệ điều hành



Sharing is learning

# Chương 5: Đồng bộ

## Busy – waiting: Giải thuật kiểm tra luân phiên

Process P0:

do

**while (turn != 0);**

critical section

**turn := 1;**

remainder section

while (1);

Process P1:

do

**while (turn != 1);**

critical section

**turn := 0;**

remainder section

while (1);

**Câu hỏi: Điều gì xảy ra nếu P0 có RS rất nhỏ còn RS của P1 rất lớn?**

- A. P1 luôn vào vùng tranh chấp trước P0
- ☒ B. P1 có thể cản P0 vào vùng tranh chấp mặc dù P0 đã nằm ngoài vùng tranh chấp.
- C. Cả 2 tiến trình có thể vào vùng tranh chấp cùng 1 lúc.
- D. P0 có thể cản P1 vào vùng tranh chấp mặc dù P1 đã nằm ngoài vùng tranh chấp.

=> Giải thuật trên **thỏa Mutual Exclusion** nhưng **không thỏa Progress** và **Bounded–Waiting**



Sharing is learning

# Chương 5: Đồng bộ

## Busy – waiting: Giải thuật sử dụng biến cờ hiệu:

- Biến chia sẻ :  
boolean flag[ 2 ]; /\* khởi đầu flag[ 0 ] = flag[ 1 ] = false \*/
- Nếu flag[ i ] = true thì Pi “sẵn sàng” vào critical section.

```
do {  
    flag[ i ] = true;    /* Pi “sẵn sàng” vào CS */  
    while ( flag[ j ] ); /* Pi “nhường” Pj      */  
        critical section  
    flag[ i ] = false;  
        remainder section  
} while (1);
```

**Câu hỏi: Giải thuật trên không thỏa mãn tính chất nào trong việc giải quyết CS Problem?**

- A. Không thỏa mãn cả ba tính chất
- ☒ B. Mutual exclusion
- C. Progress
- D. B và C đều đúng



Sharing is learning



# Chương 5: Đồng bộ

## Busy – waiting: Giải thuật Peterson: tiến trình P0, P1

Process  $P_0$

```
do {  
    /* 0 wants in */  
    flag[0] = true;  
    /* 0 gives a chance to 1 */  
    turn = 1;  
    while (flag[1] && turn == 1);  
        critical section  
    /* 0 no longer wants in */  
    flag[0] = false;  
        remainder section  
} while(1);
```

Process  $P_1$

```
do {  
    /* 1 wants in */  
    flag[1] = true;  
    /* 1 gives a chance to 0 */  
    turn = 0;  
    while (flag[0] && turn == 0);  
        critical section  
    /* 1 no longer wants in */  
    flag[1] = false;  
        remainder section  
} while(1);
```

=> Thỏa mãn 3 điều kiện



Sharing is learning

# Chương 5: Đồng bộ

## Busy – waiting: Giải thuật Bakery: n tiến trình (Process)

- Trước khi vào CS, process  $P_i$  nhận một con số. Process nào giữ con số nhỏ nhất thì được vào CS
- Trường hợp  $P_i$  và  $P_j$  cùng nhận được một chỉ số:  
Nếu  $i < j$  thì  $P_i$  được vào trước. (Đối xứng)
- Khi ra khỏi CS,  $P_i$  đặt lại số của mình bằng 0
- Cơ chế cấp số cho các process thường tạo các số theo cơ chế tăng dần, ví dụ 1, 2, 3, 3, 3, 3, 4, 5,...
- Kí hiệu  
 $(a,b) < (c,d)$  nếu  $a < c$  hoặc nếu  $a = c$  và  $b < d$   
 $\max(a_0, \dots, a_k)$  là con số  $b$  sao cho  $b \geq a_i$  với mọi  $i = 0, \dots, k$

```
boolean    choosing[ n ]; /* initially, choosing[ i ] = false */
int        num[ n ];      /* initially, num[ i ] = 0          */

do {
    choosing[ i ] = true;
    num[ i ]      = max(num[0], num[1], ..., num[n - 1]) + 1;
    choosing[ i ] = false;
    for (j = 0; j < n; j++) {
        while (choosing[ j ]);
        while ((num[ j ] != 0) && (num[ j ], j) < (num[ i ], i));
    }
    critical section
    num[ i ] = 0;
    remainder section
} while (1);
```



Sharing is learning

# Chương 5: Đồng bộ

## Busy – waiting: Giải thuật cấm ngắt:

- Trong hệ thống uniprocessor: mutual exclusion được đảm bảo
  - Gây ảnh hưởng đến các thành phần khác của hệ thống có sử dụng ngắt như system clock
  - Cần phải liên tục tạm dừng và phục hồi ngắt dẫn đến hệ thống tốn chi phí quản lý và kiểm soát
- Trong hệ thống multiprocessor: mutual exclusion không được đảm bảo
  - Chỉ cấm ngắt tại CPU thực thi lệnh `disable_interrupts`.
  - Các CPU khác vẫn có thể truy cập bộ nhớ chia sẻ.

Process  $P_i$ :

```
do {  
    disable_interrupts();  
    critical section  
    enable_interrupts();  
    remainder section  
} while (1);
```



Sharing is learning

# Chương 5: Đồng bộ

## Busy – waiting: Giải thuật dùng Swap:

```
void Swap(boolean *a,  
           boolean *b) {  
    boolean temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

```
Process Pi  
do {  
    key = true;  
    while (key == true)  
        Swap(&lock, &key);  
    critical section  
    lock = false;  
    remainder section  
} while (1)
```

- Biến chia sẻ lock (khởi tạo false)
- Biến cục bộ key
- Process P<sub>i</sub> nào thấy giá trị lock = false thì được vào CS
- Process P<sub>i</sub> vào CS sẽ cho lock = true để ngăn các process khác



Sharing is learning

# Chương 5: Đồng bộ

## Busy – waiting: Giải thuật dùng lệnh Test and set:

```
boolean TestAndSet( boolean  
*target){  
    boolean rv = *target;  
    *target = true;  
    return rv;  
}
```

□ Shared data:  
    boolean lock = false;

□ Process  $P_i$ :

```
do {  
    while (TestAndSet(&lock));  
        critical section  
    lock = false;  
        remainder section  
} while (1);
```

- Đảm bảo được tính chất Mutual Exclusion
- Quá trình chọn lựa process  $P_j$  vào CS kế tiếp là tùy ý
- Không đảm bảo được điều kiện Bounded Waiting=> Xảy ra tình trạng starvation (bị bỏ đói)



Sharing is learning

## Chương 5: Đồng bộ

**Câu 17 : Chọn phát biểu Sai trong các phát biểu dưới đây**

A. Giải thuật Peterson và giải thuật Bakery là các giải pháp đồng bộ Busy Waiting sử dụng phần mềm

**B. Cấm ngắt là giải pháp đồng bộ busy waiting luôn đảm bảo tính chất loại trừ tương hỗ**

C. Trong giải thuật Bakery, trước khi vào vùng tranh chấp, mỗi tiến trình sẽ nhận được một con số

D. Trong giải thuật Peterson, tính chất chờ đợi giới hạn luôn được đảm bảo Cấm ngắt không đảm bảo được tính chất loại trừ tương hỗ trên hệ thống đa xử lý



Sharing is learning



# Chương 5: Đồng bộ

## Sleep & wake up : Semaphore:

- Là công cụ đồng bộ cung cấp bởi OS mà không đòi hỏi busy waiting
- Semaphore S là một biến số nguyên.
- Ngoài thao tác khởi động biến thì chỉ có thể được truy xuất qua hai tác vụ cổ tính đơn nguyên (atomic) và loại trừ (mutual exclusive)
  - + wait(S) hay còn gọi là P(S): giảm giá trị semaphore ( $S=S-1$ ) .  
Kế đó nếu giá trị này thì process thực hiện lệnh wait() bị blocked.
  - + signal(S) hay còn gọi là V(S): tăng giá trị semaphore ( $S=S+1$ ) .  
Kế đó nếu giá trị này không dương, một process đang blocked bởi một lệnh wait() \ sẽ được hồi phục để thực thi.
- Tránh busy waiting: khi phải đợi thì process sẽ được đặt vào một blocked queue, trong đó chứa các process đang chờ đợi cùng một sự kiện.



Sharing is learning

# Chương 5: Đồng bộ

## Sleep & wake up : Hiện thực Semaphore:

```
void wait(semaphore S) {  
    S.value--;  
    if (S.value < 0) {  
        add this process to S.L;  
        block();  
    }  
}  
  
void signal(semaphore S) {  
    S.value++;  
    if (S.value <= 0) {  
        remove a process P from S.L;  
        wakeup(P);  
    }  
}
```



Sharing is learning

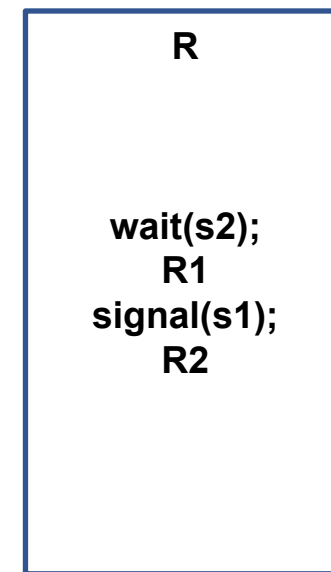
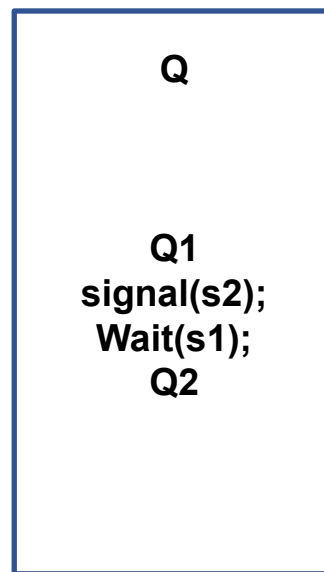
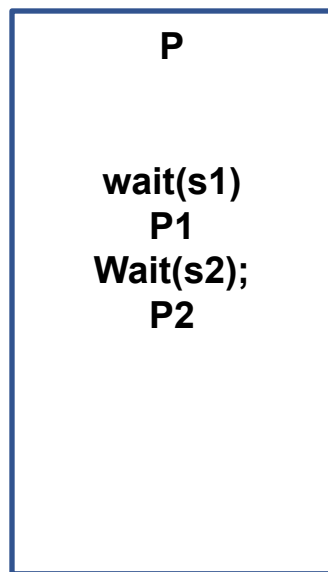
# Chương 5: Đồng bộ

## Sleep & wake up : Ví dụ về Semaphore:

Cho 3 tiến trình P(P1,P2), Q(Q1,Q2), R(R1,R2). Hãy đồng bộ hóa hoạt động của 3 tiến trình sao cho:

- a. R1 hoàn tất trước Q2 và P1.
- b. Q1 hoàn thành trước R1 và P2.

**Giải :**



Sharing is learning

# Chương 5: Đồng bộ

## Các loại Semaphore:

- Counting semaphore: một số nguyên có giá trị không hạn chế.
- Binary semaphore: có trị là 0 hay 1. Binary semaphore rất dễ hiện thực.
- Có thể hiện thực counting semaphore bằng binary semaphore



Sharing is learning

# Chương 5: Đồng bộ

## Các vấn đề đối với Semaphore:

- Semaphore cung cấp một công cụ mạnh mẽ để bảo đảm mutual exclusion và phối hợp đồng bộ các process
- Tuy nhiên, nếu các tác vụ wait(S) và signal(S) nằm rải rác ở rất nhiều processes  $\Rightarrow$  khó nắm bắt được hiệu ứng của các tác vụ này. Nếu không sử dụng đúng  $\Rightarrow$  có thể xảy ra tình trạng deadlock hoặc starvation.
- Một process bị “die” có thể kéo theo các process khác cùng sử dụng biến semaphore.



Sharing is learning

# Chương 5: Đồng bộ

## Sleep & Wake up: Monitor

- Cũng là một cấu trúc ngôn ngữ cấp cao tương tự CR, có chức năng như semaphore nhưng dễ điều khiển hơn
- Xuất hiện trong nhiều ngôn ngữ lập trình đồng thời như Concurrent Pascal, Modula-3, Java,...
- Có thể hiện thực bằng semaphore



Sharing is learning



# Chương 5: Đồng bộ

## Câu hỏi 25: Cho các tính chất sau:

- (1) Một tiến trình tạm dừng bên ngoài vùng tranh chấp không ngăn cản các tiến trình khác vào vùng tranh chấp
- (2) Các tiến trình phải từ bỏ CPU khi chưa vào vùng tranh chấp
- (3) Mỗi tiến trình phải chờ đợi để vào vùng tranh chấp trong một khoảng thời gian có hạn định nào đó. Không xảy ra tình trạng đói tài nguyên
- (4) Khi một tiến trình P đang thực thi trong vùng tranh chấp của nó thì không có tiến trình Q nào khác đang thực thi trong vùng tranh chấp của P

Lời giải dành cho vấn đề vùng tranh chấp cần phải thỏa mãn tính chất nào ?

A.(2), (4), (5)

B.(1), (3), (4)

C.(1), (2), (4)

D.(1), (3)



Sharing is learning

# Chương 5: Đồng bộ

Xét giải pháp đồng bộ sử dụng 3 semaphore full, empty, mutex để giải quyết bài toán bounded buffer như bên dưới. Biết giá trị khởi tạo của các semaphore trên lần lượt là 0, n và 1 với n là kích thước của buffer. Vai trò của semaphore empty trong giải pháp này là gì?

```
producer
do {
    ...
    nextp = new_item();
    ...
    wait(empty);
    wait(mutex);
    ...
    insert_to_buffer(nextp);
    ...
    signal(mutex);
    signal(full);
} while (1);
```

```
consumer
do {
    wait(full);
    wait(mutex);
    ...
    nextc = get_buffer_item(out);
    ...
    signal(mutex);
    signal(empty);
    ...
    consume_item(nextc);
    ...
} while (1);
```

- A. Đảm bảo producer và consumer không được thao tác trên buffer cùng lúc.
- ☒ B. Đảm bảo producer không được ghi dữ liệu vào buffer đã đầy.
- C. Đảm bảo consumer không được đọc dữ liệu từ buffer đang trống.
- D. Đảm bảo không có deadlock hoặc starvation xảy ra.



Sharing is learning

# Chương 6: Deadlocks



**Sharing is learning**

# Chương 6: Deadlock

## Định nghĩa :

- Một tiến trình gọi là deadlock nếu nó đang đợi một sự kiện mà sẽ không bao giờ xảy ra. Thông thường, có nhiều hơn một tiến trình bị liên quan trong một deadlock.
- Một tiến trình gọi là trì hoãn vô hạn định nếu nó bị trì hoãn một khoảng thời gian dài lặp đi lặp lại trong khi hệ thống đáp ứng cho những tiến trình khác.

Ví dụ: Một tiến trình sẵn sàng để xử lý nhưng nó không bao giờ nhận được CPU



Sharing is learning

# Chương 6: Deadlock

Điều kiện xảy ra deadlock:

- **Loại trừ tương hỗ:** ít nhất một tài nguyên được giữ theo nonsharable mode
- **Giữ và chờ cấp thêm tài nguyên:** Một tiến trình đang giữ ít nhất một tài nguyên và đợi thêm tài nguyên do quá trình khác giữ
- **Không trưng dụng:** tài nguyên không thể bị lấy lại mà chỉ có thể được trả lại từ tiến trình đang giữ tài nguyên đó khi nó muốn
- **Chu trình đợi:** một tập hợp các process đang chờ đợi lẫn nhau ở dạng vòng tròn (chu trình)



Sharing is learning

# Chương 6: Deadlock

## Các phương pháp giải quyết deadlock:

- Bảo đảm rằng hệ thống không rơi vào tình trạng deadlock bằng cách ngăn hoặc tránh deadlock
  - + **Ngăn deadlock:** không cho phép (ít nhất) một trong 4 điều kiện cần cho deadlock
  - + **Tránh deadlock:** các tiến trình cần cung cấp thông tin về tài nguyên nó cần để hệ thống cấp phát tài nguyên một cách thích hợp
- Cho phép hệ thống vào trạng thái deadlock, nhưng sau đó phát hiện deadlock và phục hồi hệ thống



Sharing is learning



# Chương 6: Deadlock

## Các phương pháp giải quyết deadlock:

- Bỏ qua mọi vấn đề, xem như deadlock không bao giờ xảy ra trong hệ thống  
Khá nhiều hệ điều hành sử dụng phương pháp này  
Deadlock không được phát hiện, dẫn đến việc giảm hiệu suất của hệ thống. Cuối cùng, hệ thống có thể ngưng hoạt động và phải khởi động lại



Sharing is learning

# Chương 6: Deadlock

**Câu 1: Thuật ngữ “deadlock” được hiểu như thế nào là đúng:**

A. do thông lượng tiến trình xử lý trên 1 giây quá nhỏ.

B. do xung đột tài nguyên làm treo máy.

☒ C. do thiếu tài nguyên đáp ứng cho các tiến trình cùng yêu cầu.

D. là điểm chết của các tiến trình bị khóa.



Sharing is learning

# Chương 6: Deadlock

Câu 2: “Các tiến trình cần cung cấp thông tin về tài nguyên nó cần để hệ thống cấp phát tài nguyên một cách thích hợp” là đặc điểm của phương pháp giải quyết deadlock nào?

- A. Ngăn deadlock.
- ☒ B. Tránh deadlock.
- C. Bỏ qua deadlock.
- D. Phát hiện và phục hồi deadlock.



Sharing is learning

# Chương 6: Deadlock

**Ngăn deadlock: bằng cách ngăn một trong 4 điều kiện cần của deadlock**

- Ngăn **Mutual Exclusion**

Đối với tài nguyên không chia sẻ (printer): không làm được

Đối với tài nguyên chia sẻ (read-only file): không cần thiết

- Ngăn **Hold and Wait**:

Cách 1: Mỗi tiến trình yêu cầu toàn bộ tài nguyên cần thiết một lần. Nếu có đủ tài nguyên thì hệ thống sẽ cấp phát, nếu không đủ tài nguyên thì tiến trình phải bị block

Cách 2: Khi yêu cầu tài nguyên, tiến trình không được giữ tài nguyên nào. Nếu đang có thì phải trả lại trước khi yêu cầu



Sharing is learning

# Chương 6: Deadlock

Ngăn deadlock: bằng cách ngăn một trong 4 điều kiện cần của deadlock

- Ngăn **No preemption**: nếu tiến trình A có giữ tài nguyên và đang yêu cầu tài nguyên khác nhưng tài nguyên này chưa được cấp phát ngay thì:

Cách 1: Hệ thống lấy lại mọi tài nguyên mà A đang giữ. A chỉ bắt đầu lại khi có lại tài nguyên đã bị lấy cùng tài nguyên yêu cầu.

Cách 2: Hệ thống sẽ xem tài nguyên mà A yêu cầu.



Sharing is learning

# Chương 6: Deadlock

**Ngăn deadlock:** bằng cách ngăn một trong 4 điều kiện cần của deadlock

- Ngăn **Circular wait**: Gán số thứ tự cho tất cả các tài nguyên trong hệ thống.

Mỗi tiến trình chỉ có thể yêu cầu thực thể của một loại tài nguyên theo thứ tự tăng dần (định nghĩa bởi hàm  $F$ ) của loại tài nguyên.

Ví dụ: Chuỗi yêu cầu thực thể hợp lệ: tape driver  $\rightarrow$  disk drive  $\rightarrow$  printer

Khi một tiến trình yêu cầu một thực thể của loại tài nguyên  $R_j$  thì nó phải trả lại các tài nguyên  $R_i$  với  $F(R_i) > F(R_j)$



Sharing is learning

# Chương 6: Deadlock

Câu 3: Cho các thuật ngữ sau: “Banker”, “Elphick” , “Mutual exclusion”, “No preemption”. Thuật ngữ nào là điều kiện gây ra tắc nghẽn?

- A. Banker, Mutual exclusion.
- ☒ B. No preemption, Mutual exclusion
- C. Elphick, Banker.
- D. No preemption, Elphick, Banker.

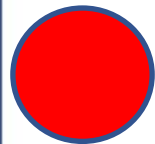


Sharing is learning



# Chương 6: Deadlock

**Câu 4: Chọn câu sai trong các câu sau:**

-  Thứ tự quá trình xử lý deadlock: tránh, ngăn ngừa, phát hiện, phục hồi.
- B. Xác định deadlock có chu trình bằng đồ thị có hướng.
- C. Ngăn chặn tất nghẽn với điều kiện “chiếm giữ và yêu cầu thêm tài nguyên” cả hai giải pháp cấp phát đều vi phạm tính toàn vẹn của dữ liệu.
- D. Ngăn chặn tất nghẽn dạng tài nguyên không thể chia sẻ nhưng cho phép kết xuất người ta dùng spooling điều phối tài nguyên.



Sharing is learning

# Chương 6: Deadlock

## Tránh deadlock

- Ngăn deadlock sử dụng tài nguyên không hiệu quả.
- Tránh deadlock vẫn đảm bảo hiệu suất sử dụng tài nguyên tối đa đến mức có thể.
- Yêu cầu mỗi tiến trình khai báo số lượng tài nguyên tối đa cần để thực hiện công việc.
- Giải thuật tránh deadlock sẽ kiểm tra trạng thái cấp phát tài nguyên để đảm bảo hệ thống không rơi vào deadlock.
- Trạng thái cấp phát tài nguyên được định nghĩa dựa trên số tài nguyên còn lại, số tài nguyên đã được cấp phát và yêu cầu tối đa của các tiến trình.



Sharing is learning

# Chương 6: Deadlock

## Trạng thái safe và unsafe

Một trạng thái của hệ thống gọi là an toàn (safe) nếu tồn tại một chuỗi an toàn

Một chuỗi quá trình  $\langle P_1, P_2, \dots, P_n \rangle$  là một chuỗi an toàn nếu: Với mọi  $i = 1, 2, \dots, n$  yêu cầu tối đa về tài nguyên của  $P_i$  có thể được thỏa bởi:

Tài nguyên mà hệ thống đang có sẵn sàng

Cùng với tài nguyên mà tất cả các  $P_j$  ( $i < j$ ) đang giữ

Nếu hệ thống ở trạng thái **safe**  $\Rightarrow$  **không deadlocks**

Nếu hệ thống đang ở trạng thái **unsafe**  $\Rightarrow$  **có thể bị deadlocks**

$\Rightarrow$  Tránh deadlocks bằng cách đảm bảo hệ thống không đi đến trạng thái unsafe



Sharing is learning

# Chương 6: Deadlock

## Các giải thuật tránh deadlock

- Mỗi tài nguyên chỉ có một thực thể -> Giải thuật đồ thị cấp phát tài nguyên
- Mỗi tài nguyên có nhiều thực thể -> Giải thuật Banker



Sharing is learning

# Chương 6: Deadlock

## Giải thuật tránh deadlock: Giải thuật Banker

Gọi  $n$  là số process,  $m$  là số loại tài nguyên

Các cấu trúc dữ liệu:

Available: vector độ dài  $m$ . Với  $Available[j] = k$  tức là loại tài nguyên  $R_j$  có  $k$  instance sẵn sàng.

Max: ma trận  $n \times m$ . Với  $Max[i, j] = k$  tức là tiến trình  $P_i$  yêu cầu tối đa  $k$  instance của loại tài nguyên  $R_j$ .

Allocation: ma trận  $n \times m$ . Với  $Allocation[i, j] = k$  tức là  $P_i$  đã được cấp phát  $k$  instance của  $R_j$ .

Need: ma trận  $n \times m$ . Với  $Need[i, j] = Max[i, j] - Allocation[i, j]$



Sharing is learning

# Chương 6: Deadlock

## Giải thuật Banker: tìm chuỗi an toàn

1. Gọi Work và Finish là hai vector độ dài là m và n. Khởi tạo: Work = Available, Finish[i] = false, với  $i = 0, 1, \dots, n-1$ .
2. Tìm i thỏa Finish[i] = false,  $Need_i \leq Work$  (hàng thứ i của Need)  
Nếu không tồn tại i như vậy, đến bước 4.
3.  $Work = Work + Allocation$ , Finish[i] = true, quay về bước 2.
4. Nếu Finish[i] = true,  $i = 1, \dots, n$ , thì hệ thống đang ở trạng thái safe



Sharing is learning

# Chương 6: Deadlock

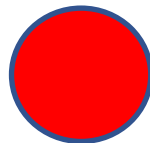
Cho sơ đồ sau:

Tiến trình	Allocation				Max			
	R1	R2	R3	R4	R1	R2	R3	R4
P1	1	2	2	1	3	2	4	3
P2	3	1	1	3	8	2	1	6
P3	5	1	4	2	7	5	5	5
P4	3	1	2	2	3	4	7	6
P5	1	2	1	4	4	6	3	7

Available			
R1	R2	R3	R4
4	2	3	5

**Câu 5: Lựa chọn nào dưới đây là một chuỗi an toàn của hệ thống?**

- A. <P4, P2, P1, P5, P3>    B. <P2, P4, P3, P1, P5>  
C. <P3, P1, P5, P4, P2>    D. <P1, P2, P3, P4, P5>



Sharing is learning



# Chương 6: Deadlock

	Allocation				Max			
Tiến trình	R1	R2	R3	R4	R1	R2	R3	R4
P1	1	2	2	1	3	2	4	3
P2	3	1	1	3	8	2	1	6
P3	5	1	4	2	7	5	5	5
P4	3	1	2	2	3	4	7	6
P5	1	2	1	4	4	6	3	7

Available			
R1	R2	R3	R4
4	2	3	5

Need			
R1	R2	R3	R4
2	0	2	2
5	1	0	3
2	4	1	3
0	3	5	4
3	4	2	3



Sharing is learning

# Chương 6: Deadlock

**Giải thuật Banker: Yêu cầu tài nguyên cho một tiến trình**

B1: Nếu  $\text{request}_i \leq \text{need}_i$  thì đến bước 2. Nếu không, báo lỗi vì tiến trình đã vượt yêu cầu tối đa

B2: Nếu  $\text{request}_i \leq \text{available}$  thì qua bước 3. Nếu không,  $P_i$  phải chờ vì tài nguyên không còn đủ để cấp phát

B3: Giả định cấp phát tài nguyên đáp ứng yêu cầu của  $P_i$  bằng cách cập nhật trạng thái hệ thống như sau:

$$\text{available} = \text{available} - \text{request}_i$$

$$\text{allocation}_i = \text{allocation}_i + \text{request}_i$$

$$\text{need}_i = \text{need}_i - \text{request}_i$$



Sharing is learning

# Chương 6: Deadlock

Cho sơ đồ sau:

Tiến trình	Allocation				Max			
	R1	R2	R3	R4	R1	R2	R3	R4
P1	1	2	2	1	3	2	4	3
P2	3	1	1	3	8	2	1	6
P3	5	1	4	2	7	5	5	5
P4	3	1	2	2	3	4	7	6
P5	1	2	1	4	4	6	3	7

Available			
R1	R2	R3	R4
4	2	3	5

**Câu hỏi 6: Yêu cầu cấp phát nào sau đây sẽ được đáp ứng?**

- A. P4 yêu cầu thêm tài nguyên (1, 2, 3, 4)
- B. P3 yêu cầu thêm tài nguyên (2, 3, 1, 3)
- C. P5 yêu cầu thêm tài nguyên (3, 2, 4, 3)
- ☒ D. P2 yêu cầu thêm tài nguyên (2, 1, 0, 2)



Sharing is learning

# Chương 6: Deadlock

**Câu 1 (1đ)(TL):** Xét một hệ thống máy tính có 5 tiến trình: P1, P2, P3, P4, P5 và 4 loại tài nguyên: R1, R2, R3, R4. Tại thời điểm  $t_0$ , trạng thái của hệ thống như sau:

Tiến trình	Allocation				Max			
	R1	R2	R3	R4	R1	R2	R3	R4
P1	1	2	2	3	2	3	4	3
P2	3	1	3	1	3	8	6	1
P3	2	1	4	5	7	7	5	7
P4	3	1	5	2	5	4	6	7
P5	1	4	4	2	1	6	7	3

Available			
R1	R2	R3	R4
3	4	4	3

Tại thời điểm  $t_1$ , nếu tiến trình P4 yêu cầu thêm tài nguyên (2, 3, 1, 3), hệ thống có đáp ứng không và giải thích tại sao? Biết hệ điều hành dùng giải thuật Banker để kiểm tra độ an toàn của hệ thống.



Sharing is learning

# Chương 6: Deadlock

**Giải:**

B1: Kiểm tra: request 1 (2,3,1,3) < available (3,4,4,3) => **True**

=> Tại thời điểm T1, Nếu P4 yêu cầu thêm tài nguyên(2, 3, 1, 3) thì hệ thống được đáp ứng.

**New state:**

	ALLOCATION				MAX				NEED				AVAILABLE				
Process	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	
P1	1	2	2	3	2	3	4	3	1	1	2	0	1	1	3	0	
P2	3	1	3	1	3	8	6	1	0	7	3	0	2	3	5	3	P1
P3	2	1	4	5	7	7	5	7	5	6	1	2	7	7	11	8	P4
P4	5	4	6	5	5	4	6	7	0	0	0	2	10	8	14	9	P2
P5	1	4	4	2	1	6	7	3	0	2	3	1	12	9	18	14	P3
													13	13	22	16	P5

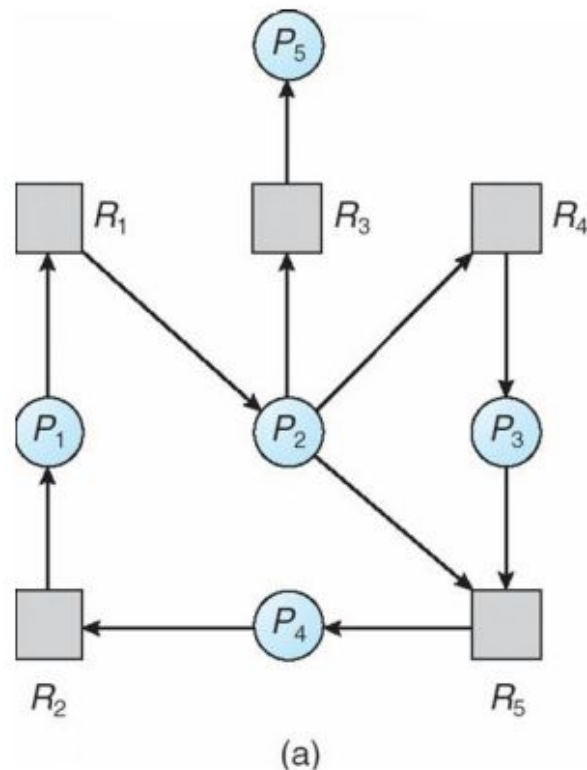
Chuỗi an toàn: P1 – P4 – P2 – P3 – P5



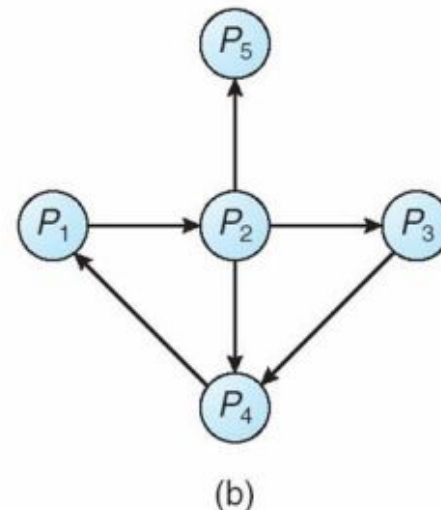
Sharing is learning

# Chương 6: Deadlock

Giải thuật phát hiện deadlock: Tài nguyên chỉ có 1 thực thể



Resource-Allocation Graph



Corresponding wait-for graph



Sharing is learning

# Chương 6: Deadlock

**Giải thuật phát hiện deadlock: Tài nguyên có nhiều thực thể**

Thực hiện tương tự giải thuật Banker

Thay cột Need bằng cột Request

Liệt kê chuỗi an toàn => Tiến trình không có trong chuỗi an toàn thì deadlock xảy ra tại tiến trình đó



Sharing is learning



# Chương 6: Deadlock

## Phục hồi deadlock:

- Khi deadlock xảy ra, để phục hồi
  - + Báo người vận hành
  - + Hệ thống tự động phục hồi bằng cách bề gãy chu trình deadlock:

Chấm dứt một hay nhiều tiến trình

Lấy lại tài nguyên từ một hay nhiều tiến trình



Sharing is learning

# Chương 6: Deadlock

## Phục hồi deadlock:

- Chấm dứt quá trình bị deadlock
  - + Chấm dứt lần lượt từng tiến trình cho đến khi không còn deadlock
  - + Sử dụng giải thuật phát hiện deadlock để xác định còn deadlock hay không



Sharing is learning

# Chương 6: Deadlock

## Phục hồi deadlock:

Những yếu tố để chấm dứt tiến trình:

- Độ ưu tiên của tiến trình
- Thời gian đã thực thi của tiến trình và thời gian còn lại
- Loại tài nguyên mà tiến trình đã sử dụng
- Tài nguyên mà tiến trình cần thêm để hoàn tất công việc Số lượng tiến trình cần được chấm dứt
- Tiến trình là interactive hay batch



Sharing is learning

# Chương 7: Quản lý bộ nhớ



**Sharing is learning**

# Chương 7: Quản lý bộ nhớ

1. Khái niệm cơ sở
2. Các kiểu địa chỉ nhớ
3. Chuyển đổi địa chỉ nhớ
4. Mô hình quản lý bộ nhớ
5. Cơ chế phân mảnh (fragmentation)
6. Cơ chế phân trang
7. Cơ chế swapping
8. Cài đặt bảng trang
9. Effective Access Time (EAT)



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Khái niệm cơ sở

- Chương trình phải được mang vào trong bộ nhớ và đặt nó trong một tiến trình để được xử lý.
- Input Queue – Một tập hợp của những tiến trình trên đĩa mà đang chờ để được mang vào trong bộ nhớ để thực thi.
- User programs trải qua nhiều bước trước khi được xử lý



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Khái niệm cơ sở

- Quản lý bộ nhớ là công việc của hệ điều hành với sự hỗ trợ của phần cứng nhằm **phân phối, sắp xếp** các process trong bộ nhớ cho hiệu quả.
- Mục tiêu cần đạt được là nạp càng nhiều process vào bộ nhớ càng tốt (gia tăng mức độ đa chương)
- Trong hầu hết các hệ thống, kernel sẽ chiếm một phần cố định của bộ nhớ; phần còn lại phân phối cho các process.



Sharing is learning



# Chương 7: Quản lý bộ nhớ

## Khái niệm cơ sở

Các yêu cầu đối với việc quản lý bộ nhớ

- Cấp phát/ thu hồi bộ nhớ cho các process
- Tái định vị (relocation): khi swapping,...
- Bảo vệ: phải kiểm tra truy xuất bộ nhớ có hợp lệ không
- Chia sẻ: cho phép các process chia sẻ vùng nhớ chung
- Kết gán địa chỉ nhớ luận lý của user vào địa chỉ thực



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Địa chỉ bộ nhớ

- Địa chỉ vật lý (physical address) (**địa chỉ thực**) là một vị trí thực trong bộ nhớ chính
- Địa chỉ luận lý (logical address) là một vị trí nhớ được diễn tả trong một chương trình (còn gọi là **địa chỉ ảo** virtual address): Các trình biên dịch (compiler) tạo ra mã lệnh chương trình mà trong đó mọi tham chiếu bộ nhớ đều là địa chỉ luận lý



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Địa chỉ bộ nhớ

- Địa chỉ tuyệt đối (absolute address): địa chỉ tương đương với địa chỉ thực.
- Địa chỉ tương đối (relative address) (địa chỉ khả tái định vị, relocatable address) là một kiểu địa chỉ luận lý trong đó các địa chỉ được biểu diễn tương đối so với một vị trí xác định nào đó trong chương trình.
- Ví dụ: 12 byte so với vị trí bắt đầu chương trình,...



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK2(2021-2022) Địa chỉ có dạng "18 byte so với vị trí bắt đầu chương trình" là địa chỉ gì?

- ☒ A. Relative address
- ☐ B. Physical address
- ☐ C. Absolute address
- ☐ D. Invalid address



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK1(2018-2019) Cho bảng phân đoạn của một tiến trình P<sub>i</sub> như hình bên dưới, hỏi địa chỉ vật lý tương ứng với địa chỉ logic <1,150> là bao nhiêu?

- A. 2169
- ☒ B. 1480
- C. 340
- D. 330

Segment	Base	Length
0	2019	500
1	1330	180
2	190	300



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK1(2018-2019) Quy trình tính toán địa chỉ vật lý trong mô hình quản lý bộ nhớ được thực hiện như thế nào nếu địa chỉ luận lý  $\langle s, d \rangle$ ?

- A. Dựa vào  $s$  để tìm ra limit và base, so sánh  $d$  với limit, nếu  $d$  nhỏ hơn limit thì địa chỉ vật lý bằng  $\text{base} + d$
- B. Dựa vào  $s$  để tìm ra limit và base, so sánh  $d$  với base, nếu  $d$  nhỏ hơn base thì địa chỉ vật lý bằng  $\text{limit} + d$
- C. Dựa vào  $s$  để tìm ra limit và base, so sánh  $d$  với limit, nếu  $d$  nhỏ limit thì địa chỉ vật lý bằng  $\langle \text{base}, d \rangle$
- D. Dựa vào  $s$  để tìm ra limit và base, so sánh  $d$  với base, nếu  $d$  nhỏ hơn base thì địa chỉ vật lý bằng  $\langle \text{limit}, d \rangle$



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK1(2019-2020) Cho bảng phân đoạn của một tiến trình như sau. Địa chỉ luận lý nào dưới đây **KHÔNG** hợp lệ?

- ☒ A. 2, 215
- ☐ B. 1, 178
- ☐ C. 3, 399
- ☐ D. 0, 42

Segment	Base	Length
0	2017	146
1	564	223
2	900	75
3	1242	680



Sharing is learning



# Chương 7: Quản lý bộ nhớ

## Nạp chương trình vào bộ nhớ

- Bộ linker: kết hợp các object module thành một file nhị phân thực thi gọi là load module
- Bộ loader: nạp load module vào bộ nhớ chính



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chuyển đổi địa chỉ nhớ

- Quá trình ánh xạ một địa chỉ từ không gian địa chỉ này sang không gian địa chỉ khác.
- Biểu diễn:
  - Trong source code: symbolic (các biến, hằng, pointer,...)
  - Thời điểm biên dịch: thường là địa chỉ khả tái định vị  
(Ví dụ: a ở vị trí 12 byte so với vị trí bắt đầu module)
  - Thời điểm linking/ loading: có thể là địa chỉ thực  
(Ví dụ: dữ liệu nằm tại địa chỉ bộ nhớ thực 2030 )



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chuyển đổi địa chỉ nhớ

- Địa chỉ lệnh và dữ liệu được chuyển đổi thành địa chỉ thực có thể xảy ra tại ba thời điểm khác nhau
- Compile time: nếu biết trước địa chỉ bộ nhớ của chương trình thì có thể kết gán địa chỉ tuyệt đối lúc biên dịch (phải biên dịch lại nếu thay đổi địa chỉ nạp chương trình)
- Load time: vào thời điểm loading, loader phải chuyển đổi địa chỉ khả tái định vị thành địa chỉ thực dựa trên một địa chỉ nền (Địa chỉ thực được tính toán vào thời điểm nạp chương trình => phải tiến hành reload nếu địa chỉ nền thay đổi)



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chuyển đổi địa chỉ nhớ

- Excution time: khi trong quá trình thực thi, process có thể được di chuyển từ segment này sang segment khác trong bộ nhớ thì quá trình chuyển đổi địa chỉ được trì hoãn đến thời điểm thực thi. Lưu ý:
  - Cần sự hỗ trợ của phần cứng cho việc ánh xạ địa chỉ
  - Sử dụng trong đa số các OS đa dụng trong đó có các cơ chế swapping, paging, segmentation



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chuyển đổi địa chỉ nhớ - Dynamic Linking

- Quá trình link đến một module ngoài (external module) được thực hiện sau khi đã tạo xong load module (i.e. file có thể thực thi, executable)
- Load module chứa các stub (cần sự hỗ trợ của OS) tham chiếu (refer) đến routine của external module.
- Cơ chế: chỉ khi nào cần được gọi đến thì một thủ tục mới được nạp vào bộ nhớ chính  $\Rightarrow$  tăng độ hiệu dụng của bộ nhớ



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chuyển đổi địa chỉ nhớ - Ưu điểm của Dynamic Linking

- Dùng các phiên bản khác nhau của external module mà không cần sửa đổi, biên dịch lại
- Chia sẻ mã (code sharing)
- Cần sự hỗ trợ của OS trong việc kiểm tra: xem một thủ tục nào đó có thể được chia sẻ giữa các process hay là phần mã của riêng một process
- Rất hiệu quả trong trường hợp tồn tại khối lượng lớn mã chương trình có tần suất sử dụng thấp, không được sử dụng thường xuyên (ví dụ các thủ tục xử lý lỗi)



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Mô hình quản lý bộ nhớ

- Mô hình đơn giản, **không có bộ nhớ ảo**
- Một process phải được nạp hoàn toàn vào bộ nhớ thì mới được thực thi.
- Các cơ chế quản lý bộ nhớ sau đây rất ít (hầu như không còn) được dùng trong các hệ thống hiện đại: Phân chia cố định (fixed partitioning), Phân chia động (dynamic partitioning), Phân trang đơn giản (simple paging) và Phân đoạn đơn giản (simple segmentation)



Sharing is learning



# Chương 7: Quản lý bộ nhớ

## Cơ chế phân mảnh

- Phân mảnh ngoại (external fragmentation): Kích thước không gian nhớ còn trống đủ để thỏa mãn một yêu cầu cấp phát, tuy nhiên không gian nhớ này không liên tục  $\Rightarrow$  có thể dùng cơ chế kết khối (compaction) để gom lại thành vùng nhớ liên tục.
- Phân mảnh nội (internal fragmentation): Kích thước vùng nhớ được cấp phát có thể hơi lớn hơn vùng nhớ yêu cầu.
- Hiện tượng phân mảnh nội thường xảy ra khi bộ nhớ thực được chia thành các khối kích thước cố định (fixed-sized block) và các process được cấp phát theo đơn vị khối. Ví dụ: cơ chế phân trang (paging).



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK1(2019-2020) Nếu hệ thống cấp phát vùng nhớ có kích thước 20480 byte cho tiến trình yêu cầu 20324 byte thì sẽ dẫn đến tình trạng gì?

- ☒ A. Phân mảnh nội
- B. Phân mảnh ngoại
- C. Deadlock
- D. Số lỗi trang tăng lên



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Fixed partitioning

- Khi khởi động hệ thống, bộ nhớ chính được chia thành nhiều phần rời nhau gọi là các partition có kích thước bằng nhau hoặc khác nhau
- Process nào có kích thước nhỏ hơn hoặc bằng kích thước partition thì có thể được nạp vào partition đó (gây ra hiện tượng phân mảnh nội).
- Nếu chương trình có kích thước lớn hơn partition thì phải dùng cơ chế overlay.



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chiến lược placement

- Partition có kích thước **bằng nhau**
- Nếu còn partition trống  $\Rightarrow$  process mới sẽ được nạp vào partition đó
- Nếu không còn partition trống, nhưng trong đó có process đang bị blocked  $\Rightarrow$  swap process đó ra bộ nhớ phụ nhường chỗ cho process mới.



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chiến lược placement

- Partition có kích thước **không bằng nhau**: Giải pháp 1
  - Gán mỗi process vào partition nhỏ nhất phù hợp với nó
  - Có hàng đợi cho mỗi partition
  - Giảm thiểu phân mảnh nội
- Vấn đề: có thể có một số hàng đợi trống không (vì không có process với kích thước tương ứng) và hàng đợi dày đặc



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chiến lược placement

- Partition có kích thước **không bằng nhau**: Giải pháp 2
- Chỉ có một hàng đợi chung cho mọi partition
- Khi cần nạp một process vào bộ nhớ chính  $\Rightarrow$  chọn partition nhỏ nhất còn trống



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Dynamic partitioning

- Số lượng partition không cố định và partition có thể có kích thước khác nhau
- Mỗi process được cấp phát chính xác dung lượng bộ nhớ cần thiết
- Gây ra hiện tượng phân mảnh ngoại



Sharing is learning



# Chương 7: Quản lý bộ nhớ

## Chiến lược placement

- Dùng để quyết định cấp phát khối bộ nhớ trống nào cho một process.
- Mục tiêu: giảm chi phí compaction
- Các chiến lược placement:
  - Best-Fit: chọn khối nhớ trống nhỏ nhất
  - First-Fit: chọn khối nhớ trống phù hợp đầu tiên kể từ đầu bộ nhớ
  - Next-Fit: chọn khối nhớ trống phù hợp đầu tiên kể từ vị trí cấp phát cuối cùng
  - Worst-Fit: chọn khối nhớ trống lớn nhất



Sharing is learning

## Chương 7: Quản lý bộ nhớ

**HK1(2019-2020)** Giả sử bộ nhớ chính được phân chia thành các phân vùng cố định theo thứ tự như sau: 1 (250 KB), 2 (300 KB), 3 (100 KB), 4 (400 KB), 5 (300 KB). Biết con trỏ đang nằm ở vùng nhớ thứ 2, vùng nhớ thứ 2 đã được cấp phát, các vùng nhớ khác vẫn còn trống. Hỏi tiến trình P có kích thước 160 KB, sẽ được cấp phát trong vùng nhớ nào, nếu dùng giải thuật next-fit?

- A. 1
- B. 2
- ☒ C. 4
- D. 5



Sharing is learning

## Chương 7: Quản lý bộ nhớ

**HK2(2021-2022)** Giả sử bộ nhớ chính được phân chia thành các phân vùng cố định theo thứ tự như sau: 1 (200 KB), 2 (180 KB), 3 (140 KB), 4 (220 KB), 5 (360 KB). Biết con trỏ đang nằm ở vùng nhớ thứ 2, vùng nhớ thứ 2 đã được cấp phát, các vùng nhớ khác vẫn còn trống. Hỏi tiến trình P có kích thước 196 KB, sẽ được cấp phát trong vùng nhớ nào, nếu dùng giải thuật best-fit?

- A. 4
- B. 3
- C. 5
- ☒ D. 1



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Cơ chế phân trang

- Bộ nhớ vật lý: khung trang (frame)
- Bộ nhớ luận lý (logical memory) hay không gian địa chỉ luận lý là tập mọi địa chỉ luận lý mà một chương trình bất kỳ có thể sinh ra (page)
- Bảng phân trang (page table) để ánh xạ địa chỉ luận lý thành địa chỉ thực

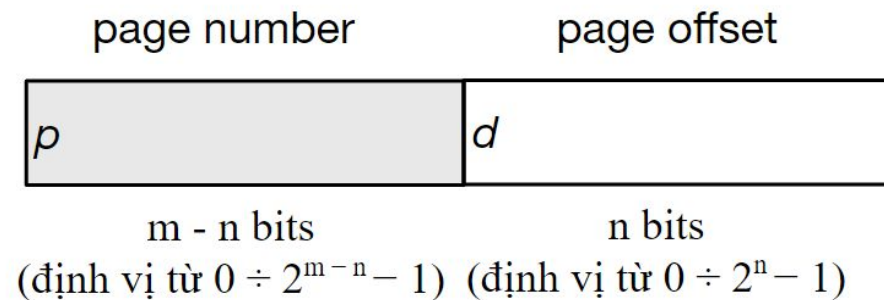


Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Cơ chế phân trang

- Địa chỉ luận lý gồm có:
  - Số hiệu trang (Page number)  $p$
  - Địa chỉ tương đối trong trang (Page offset)  $d$
- Nếu kích thước của không gian địa chỉ ảo là  $2^m$ , và kích thước của trang là  $2^n$  (đơn vị là byte hay word tùy theo kiến trúc máy) thì



Bảng trang sẽ có tổng cộng  $2^m/2^n = 2^{m-n}$  mục (entry)



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK1(2018-2019) Xét một máy tính có không gian địa chỉ luận lý 32 bit và kích thước 1 trang là 4 KByte. Hỏi bảng trang (page table) có bao nhiêu mục (entry) ?

- ☒ A.  $2^{20}$
- B.  $2^{21}$
- C.  $2^{22}$
- D.  $2^{23}$



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK2(2021-2022) Xét một không gian địa chỉ ảo có 108 trang, mỗi trang có kích thước 2048 byte được ánh xạ vào bộ nhớ vật lý có 64 khung trang. Địa chỉ luận lý gồm bao nhiêu bit?

A. 108

B. 7

☒ C. 18

D. 11



Sharing is learning



# Chương 7: Quản lý bộ nhớ

HK2(2021-2022) Xét một không gian địa chỉ ảo có 108 trang, mỗi trang có kích thước 2048 byte được ánh xạ vào bộ nhớ vật lý có 64 khung trang. Bảng phân trang có tất cả bao nhiêu mục (entry)?

- ☒ A. 108
- B. 4096
- C. 64
- D. 18



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK1(2019-2020) Bộ vi xử lý MIPS R2000 có không gian địa chỉ ảo 32 bit với kích thước trang là 4096 byte. Hỏi kích thước của mỗi mục (entry) trong bảng trang là bao nhiêu nếu bảng trang có kích thước 2 MB?

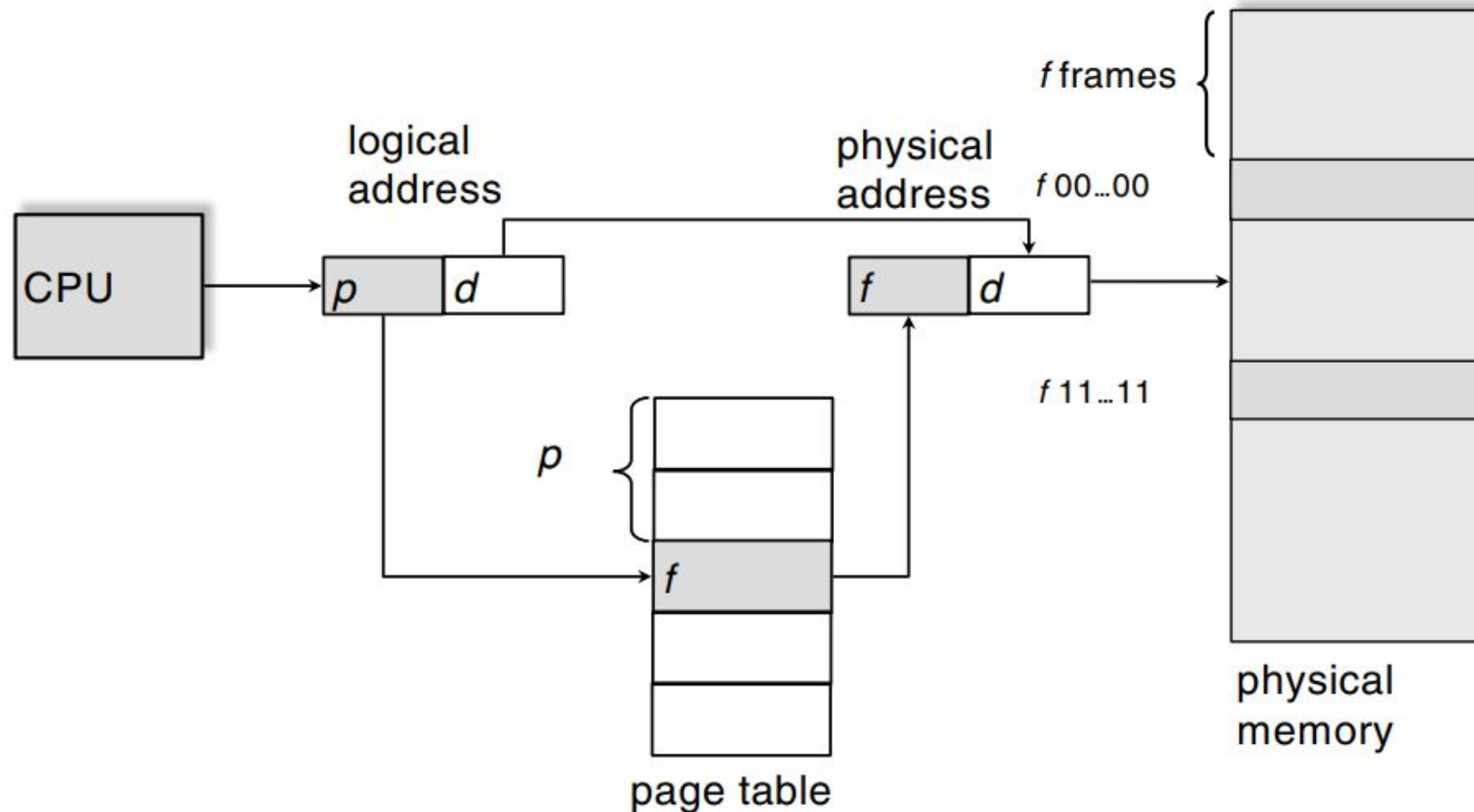
- A. 8 bit
- ☒ B. 16 bit
- C. 24 bit
- D. 32 bit



Sharing is learning

# Chương 7: Quản lý bộ nhớ

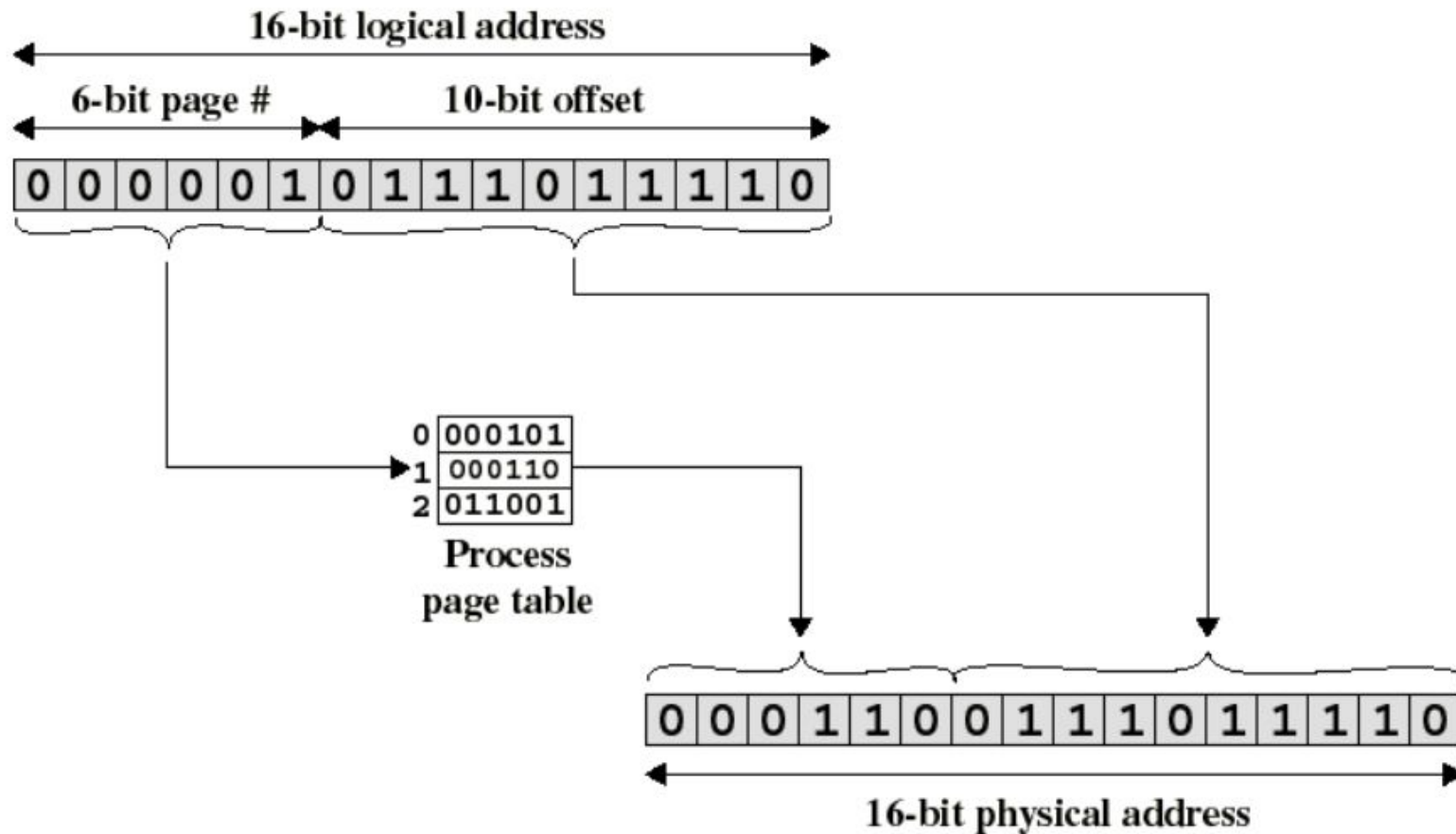
## Chuyển đổi địa chỉ trong paging



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Chuyển đổi địa chỉ trong paging



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK1(2018-2019) Cho địa chỉ vật lý là 4100 sẽ được chuyển thành địa chỉ ảo bao nhiêu? Biết rằng kích thước mỗi frame là 1K bytes, và bảng ánh xạ địa chỉ

- A. 4100
- B. 1024
- ☒ C. 1028
- D. 5124

0	6
1	4
2	5
3	7
4	1
5	9



Sharing is learning

# Chương 7: Quản lý bộ nhớ

**HK1(2018-2019)** Xét một không gian địa chỉ luận lý có 32 trang, mỗi trang có kích thước 2 MByte. Ánh xạ vào bộ nhớ vật lý có 16 khung trang. Địa chỉ luận lý và địa chỉ vật lý gồm bao nhiêu bit?

- A. Địa chỉ luận lý cần 15 bits, địa chỉ vật lý cần 16 bits
- B. Địa chỉ luận lý cần 25 bits, địa chỉ vật lý cần 26 bits
- C. Địa chỉ luận lý cần 16 bits, địa chỉ vật lý cần 15 bits
- ☒ D. Địa chỉ luận lý cần 26 bits, địa chỉ vật lý cần 25 bits



Sharing is learning

## Chương 7: Quản lý bộ nhớ

HK1(2019-2020) Xét một hệ thống có bộ nhớ được cấp phát theo cơ chế phân trang với kích thước trang và khung trang là 2048 byte. Biết địa chỉ ảo 4532 được ánh xạ thành địa chỉ vật lý 6580. Hỏi trang 2 của bộ nhớ ảo được nạp vào khung trang nào của bộ nhớ vật lý?

- A. 2
- ☒ B. 3
- C. 4
- D. 5



Sharing is learning

# Chương 7: Quản lý bộ nhớ

**HK1(2019-2020)** Một máy tính có không gian địa chỉ ảo 32 bit, quản lý bộ nhớ bằng cách sử dụng bảng trang 3 cấp. Trong đó 4 bit đầu tiên là dành cho bảng trang cấp 1, 4 bit kế tiếp dành cho bảng trang cấp 2, 8 bit kế tiếp dành cho bảng trang cấp 3, số bit còn lại dành cho offset. Khi tiến hành truy xuất địa chỉ 0xAADDCCBAD thì offset là bao nhiêu?

- A. 0xAA
- ☒ B. 0xCBAD
- C. 0xDDCB
- D. 0xAD



Sharing is learning



# Chương 7: Quản lý bộ nhớ

HK2(2021-2022) Xét một hệ thống có bộ nhớ được cấp phát theo cơ chế phân trang với kích thước trang và khung trang là 2048 byte. Biết các trang 1, 2, 3, 4 của bộ nhớ ảo lần lượt được nạp vào khung trang 3, 6, 5, 1 của bộ nhớ vật lý. Hỏi địa chỉ ảo 4542 được nạp vào khung trang nào của bộ nhớ chính?

- ☒ A. 6
- B. 5
- C. 3
- D. 1



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Cơ chế swapping

Một process có thể tạm thời bị swap ra khỏi bộ nhớ chính và lưu trên một hệ thống lưu trữ phụ. Sau đó, process có thể được nạp lại vào bộ nhớ để tiếp tục quá trình thực thi.



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Cài đặt bảng trang

- Bảng phân trang thường được lưu giữ trong bộ nhớ chính
- Mỗi process được hệ điều hành cấp một bảng phân trang
- Thanh ghi page-table base (PTBR) trỏ đến bảng phân trang
- Thanh ghi page-table length (PTLR) biểu thị kích thước của bảng phân trang (có thể được dùng trong cơ chế bảo vệ bộ nhớ)



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Cài đặt bảng trang

- Thường dùng một bộ phận cache phần cứng có tốc độ truy xuất và tìm kiếm cao, gọi là thanh ghi kết hợp (associative register) hoặc translation look-aside buffers (TLBs)



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK2(2021-2022) Chọn phát biểu **SAI** về cơ chế phân trang?

- A. Bộ nhớ vật lý được chia thành các khung trang còn bộ nhớ luận lý được chia thành các trang
- B. Bảng phân trang dùng để ánh xạ địa chỉ luận lý thành địa chỉ thực
- C. Mỗi tiến trình được hệ điều hành cấp một bảng phân trang
- ☒ D. Kích thước của bảng phân trang được xác định bởi thanh ghi page-table base (PTBR)



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK2(2021-2022) Trong cơ chế phân trang, chỉ số khung  $f$  được xác định như thế nào từ địa chỉ luận lý ( $p$ ,  $d$ )

- ☒ A. Chỉ số khung  $f$  nằm ở mục (dòng)  $p$  trong bảng phân trang
- B. Chỉ số khung  $f$  nằm ở mục (dòng)  $d$  trong bảng phân trang
- C. Chỉ số khung  $f$  nằm ở mục (dòng)  $p+d$  trong bảng phân trang
- D. Chỉ số khung  $f$  bằng  $p + d$



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Effective Access Time (EAT)

- Thời gian tìm kiếm trong TLB (associative lookup):  $\epsilon$  (ns)
- Thời gian một chu kỳ truy xuất bộ nhớ:  $x$  (ns)
- Hit ratio: tỉ số giữa số lần chỉ số trang được tìm thấy (hit) trong TLB và số lần truy xuất khởi nguồn từ CPU (Kí hiệu hit ratio:  $\alpha$ )
- Thời gian cần thiết để có được chỉ số frame
- Khi chỉ số trang có trong TLB (hit)  $\epsilon + x$
- Khi chỉ số trang không có trong TLB (miss)  $\epsilon + x + x$



Sharing is learning

# Chương 7: Quản lý bộ nhớ

## Effective Access Time (EAT)

Thời gian truy xuất hiệu dụng

$$\text{EAT} = (\varepsilon + x)\alpha + (\varepsilon + 2x)(1 - \alpha) = (2 - \alpha)x + \varepsilon$$



Sharing is learning



# Chương 7: Quản lý bộ nhớ

**HK1(2018-2019)** Xét một hệ thống sử dụng kỹ thuật phân trang, với bảng trang được lưu trữ trong bộ nhớ chính. Nếu sử dụng TLBs với hit-radio (tỉ lệ tìm thấy) là 90%, thời gian để tìm trong TLBs bằng 30ns, thì thời gian truy xuất bộ nhớ trong hệ thống (effective memory reference time) là 250ns. Hỏi thời gian một lần truy xuất bộ nhớ bình thường là khoảng bao nhiêu?

- ☒ A. 200ns
- ☐ B. 110ns
- ☐ C. 220ns
- ☐ D. 250ns



Sharing is learning

## Chương 7: Quản lý bộ nhớ

HK1(2019-2020) Xét một hệ thống sử dụng kỹ thuật phân trang với bảng trang được lưu trữ trong bộ nhớ chính. Nếu sử dụng TLBs với hit-radio (tỉ lệ tìm thấy) là 90% thì thời gian truy xuất bộ nhớ trong hệ thống (effective memory reference time) là 240ns. Biết thời gian để tìm trong TLBs là 20ns, hãy xác định thời gian truy xuất bộ nhớ trong hệ thống nếu tỉ lệ tìm thấy giảm xuống còn 85%?

- A. 200
- B. 20
- C. 230
- ☒ D. 250



Sharing is learning

# Chương 7: Quản lý bộ nhớ

HK2(2021-2022) Xét một hệ thống sử dụng kỹ thuật phân trang với bảng trang được lưu trữ trong bộ nhớ chính. Nếu sử dụng TLBs với hit-radio  $\alpha = 0.9$  thì thời gian truy xuất bộ nhớ trong hệ thống (effective memory reference time)  $EAT = 250\text{ns}$ . Biết thời gian một chu kỳ truy xuất bộ nhớ ( $x$ ) là  $210\text{ns}$ . Hỏi thời gian để tìm trong TLBs ( $\epsilon$ ) là bao nhiêu?

- ☒ A. 19ns
- B. 231ns
- C. 269ns
- D. 189ns



Sharing is learning

# Chương 8: Bộ nhớ ảo



**Sharing is learning**

# Chương 8: Bộ nhớ ảo

1. Tổng quan về bộ nhớ ảo
2. Cài đặt bộ nhớ ảo: Xử lý lỗi trang và thay thế trang
3. Các giải thuật thay trang (Page Replacement Algorithms)
4. Vấn đề cấp phát Frames
5. Vấn đề Thrashing



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Tổng quan về bộ nhớ ảo

- Bộ nhớ ảo là một kỹ thuật cho phép xử lý một tiến trình không được nạp toàn bộ vào bộ nhớ vật lý
- Không gian trao đổi giữa bộ nhớ chính và bộ nhớ phụ (swap space)
- Ưu điểm của bộ nhớ ảo
  - Số lượng process trong bộ nhớ nhiều hơn
  - Một process có thể thực thi ngay cả khi kích thước của nó lớn hơn bộ nhớ thực
  - Giảm nhẹ công việc của lập trình viên



Sharing is learning

# Chương 8: Bộ nhớ ảo

HK2(2021-2022) Cho các đặc điểm sau. Đặc điểm nào **KHÔNG** phải là ưu điểm của bộ nhớ ảo?

A. (1), (2)

B. (2)

☒ C. (4)

D. (3), (4)

(1) Số lượng tiến trình trong bộ nhớ nhiều hơn

(2) Một tiến trình có thể thực thi ngay cả khi kích thước của nó lớn hơn bộ nhớ thực

(3) Giảm nhẹ công việc của lập trình viên

(4) Tốc độ truy xuất bộ nhớ nhanh hơn



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Cài đặt bộ nhớ ảo

- Có hai kỹ thuật: Demand paging, Demand Segmentation
- Phần cứng memory management phải hỗ trợ paging và/hoặc segmentation
- OS phải quản lý sự di chuyển của trang/đoạn giữa bộ nhớ chính và bộ nhớ thứ cấp



Sharing is learning



# Chương 8: Bộ nhớ ảo

HK1(2018-2019) Yêu cầu nào trong các yêu cầu dưới đây **KHÔNG** phải là điều kiện cần để có thể cài đặt bộ nhớ ảo?

- A. Phần cứng memory management phải hỗ trợ paging và/hoặc segmentation
- B. Hệ điều hành phải quản lý sự di chuyển của trang/đoạn giữa bộ nhớ chính và bộ nhớ thứ cấp
- ☒ C. Bộ nhớ thứ cấp phải có dung lượng lớn hơn bộ nhớ chính
- D. Tất cả yêu cầu trên



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Cài đặt bộ nhớ ảo – Phân trang theo yêu cầu

- Demand paging: các trang của quá trình chỉ được nạp vào bộ nhớ chính khi được yêu cầu.
- Khi có một tham chiếu đến một trang mà không có trong bộ nhớ chính (valid bit) thì phần cứng sẽ gây ra một ngắt (gọi là page-fault trap) kích khởi page-fault service routine (PFSR) của hệ điều hành.



Sharing is learning

# Chương 8: Bộ nhớ ảo

HK1(2018-2019) Chọn phát biểu **ĐÚNG** về phân trang theo yêu cầu?

- A. Số lượng tiến trình trong bộ nhớ nhiều hơn so với phân đoạn theo yêu cầu
- ☒ B. Các trang của tiến trình chỉ được nạp vào bộ nhớ chính khi yêu cầu
- C. Phân trang theo yêu cầu làm giảm thời gian truy xuất bộ nhớ
- D. Phân trang theo yêu cầu có thể thực hiện mà không cần sự hỗ trợ của phần cứng



Sharing is learning

# Chương 8: Bộ nhớ ảo

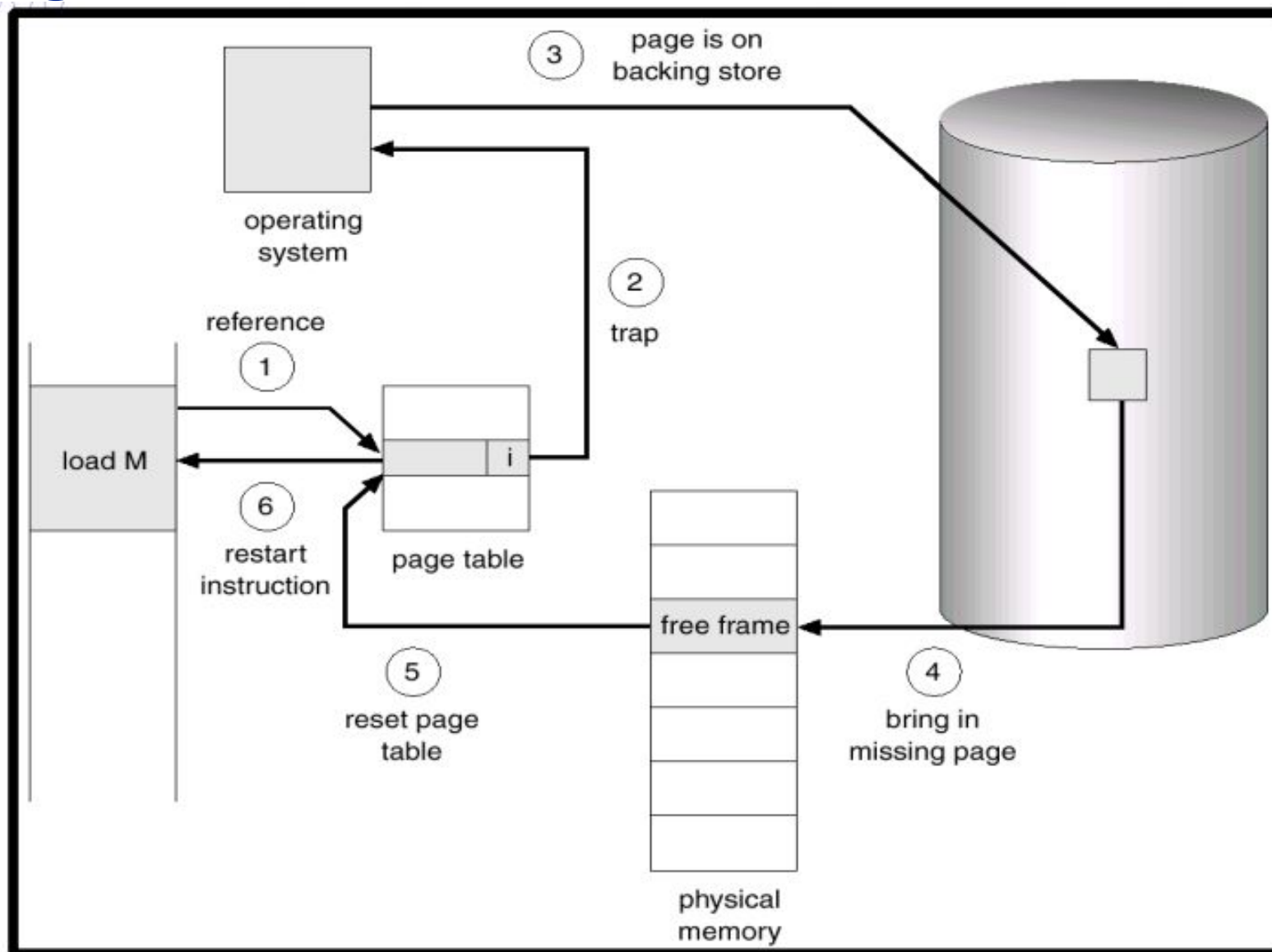
## Cài đặt bộ nhớ ảo – Phân trang theo yêu cầu

- PFSR:
  1. Chuyển process về trạng thái blocked
  2. Phát ra một yêu cầu đọc đĩa để nạp trang được tham chiếu vào một frame trống; trong khi đợi I/O, một process khác được cấp CPU để thực thi
  3. Sau khi I/O hoàn tất, đĩa gây ra một ngắt đến hệ điều hành; PFSR cập nhật page table và chuyển process về trạng thái ready.



Sharing is learning

# Chương 8: Bộ nhớ ảo



# Chương 8: Bộ nhớ ảo

## Thay thế trang nhớ

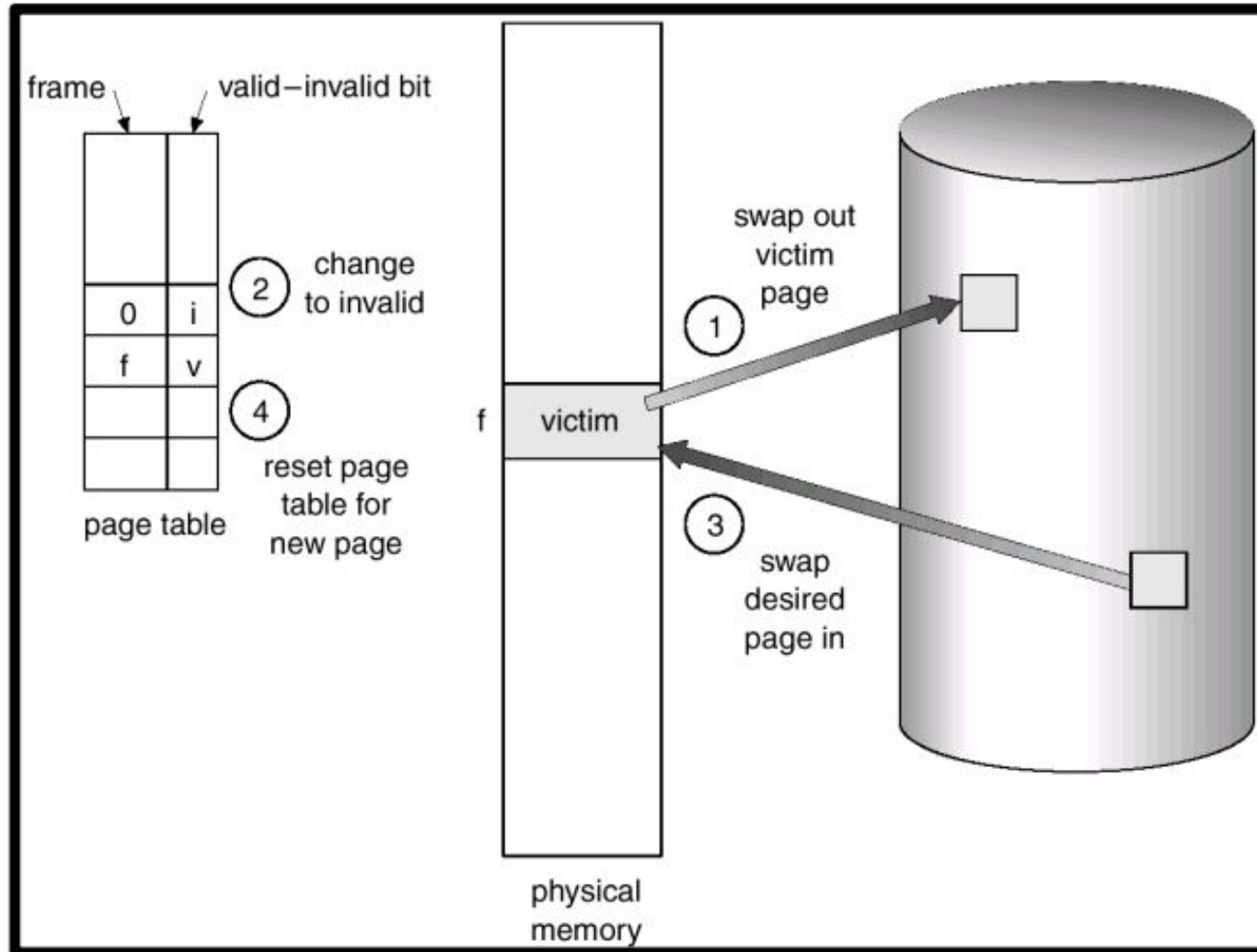
Bước 2 của PFSR giả sử phải thay trang vì không tìm được frame trống, PFSR được bổ sung:

- Xác định vị trí trên đĩa của trang đang cần
- Tìm một frame trống:
  - Nếu có frame trống thì dùng nó
  - Nếu không có frame trống thì dùng giải thuật thay trang để chọn một trang hy sinh (victim page)
- Ghi victim page lên đĩa; cập nhật page table và frame table tương ứng
- Đọc trang đang cần vào frame trống (đã có được từ bước 2); cập nhật page table và frame table tương ứng



Sharing is learning

# Chương 8: Bộ nhớ ảo



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Các giải thuật thay trang - FIFO

- Các dữ liệu cần biết ban đầu: Số khung trang, tình trạng ban đầu và chuỗi tham chiếu
- Thay thế trang nhớ khi được tham chiếu sớm nhất trong quá khứ



Sharing is learning



# Chương 8: Bộ nhớ ảo

## Các giải thuật thay trang - FIFO

Ví dụ: Sử dụng 3 khung trang, ban đầu cả 3 đều trống, chuỗi tham chiếu: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

Ta có được bảng sau với \* là các lỗi trang(page fault).

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Các giải thuật thay trang - FIFO

Sử dụng 3 khung trang sẽ có 9 lỗi trang

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4
*	*	*	*	*	*	*			*	*	

Sử dụng 4 khung trang sẽ có 10 lỗi trang

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
*	*	*	*			*	*	*	*	*	*



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Các giải thuật thay trang – FIFO. Nghịch lý Belady

Bất thường (anomaly) Belady: số page fault tăng mặc dầu quá trình đã được cấp nhiều frame hơn.



Sharing is learning

# Chương 8: Bộ nhớ ảo

HK2(2021-2022) Tình trạng số lỗi trang tăng khi được cấp nhiều khung trang hơn được gọi là gì?

- A. ☒ Nghịch lý Belady
- B. ☐ Deadlock
- C. ☐ Starvation
- D. ☐ Hệ thống đang ở trạng thái không an toàn



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Các giải thuật thay trang - OPT

- Thay thế trang nhớ sẽ được tham chiếu trễ nhất trong tương lai
- Ví dụ: cho chuỗi : 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 với 3 khung trang, ban đầu đều trống.

Sử dụng 3 khung trang với 9 lỗi trang

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
*	*	*	*		*		*			*			*				*		



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Các giải thuật thay trang - LRU

- Thay thế trang nhớ được tham chiếu sớm nhất trong quá khứ.  
(chiều ngược lại với OPT)
- Ví dụ: cho chuỗi : 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1 với 3 khung trang, ban đầu đều trống.

Sử dụng 3 khung trang với 12 lỗi trang

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
*	*	*	*		*		*	*	*	*			*		*		*		



Sharing is learning



# Chương 8: Bộ nhớ ảo

HK1 (2018-2019) Khi dùng bộ nhớ ảo và cần thay thế trang, thuật toán nào dưới đây dùng thời điểm trang sẽ được sử dụng để xem xét việc chọn trang thay thế

- A. FIFO
- B. LRU
- ☒ C. Optimal
- D. Tất cả đều đúng



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Vấn đề cấp phát Frames

- OS phải quyết định cấp cho mỗi process bao nhiêu frame
- Chiến lược cấp phát tĩnh (fixed-allocation): Số frame cấp cho mỗi process không đổi (cấp phát bằng nhau, theo tỉ lệ hoặc độ ưu tiên). Được xác định vào thời điểm loading(nạp) và có thể tùy thuộc vào từng ứng dụng (kích thước của nó,...)
- Chiến lược cấp phát động (variable-allocation):
  - Số frame cấp cho mỗi process có thể thay đổi trong khi nó chạy
  - Nếu tỷ lệ page-fault cao  $\Rightarrow$  cấp thêm frame
  - Nếu tỷ lệ page-fault thấp  $\Rightarrow$  giảm bớt frame
  - OS phải mất chi phí để ước định các process



Sharing is learning



# Chương 8: Bộ nhớ ảo

**HK1(2019-2020)** Trong kỹ thuật cài đặt bộ nhớ ảo sử dụng phân trang theo yêu cầu, khi sử dụng chiến lược cấp phát động, số lượng khung trang (frame) được cấp phát cho một tiến trình sẽ thay đổi như thế nào nếu tỷ lệ lỗi trang (page fault) thấp?

- ☒ A. Giảm xuống
- B. Tăng lên
- C. Không thay đổi
- D. Bị hệ thống thu hồi toàn bộ



Sharing is learning

# Chương 8: Bộ nhớ ảo

HK2(2021-2022) Đặc điểm của chiến lược cấp phát tĩnh trong cơ chế quản lý bộ nhớ ảo là gì?

- A. Số khung trang cấp cho mỗi tiến trình không đổi, được xác định vào thời điểm biên dịch và có thể tùy thuộc vào từng ứng dụng
- ☒ B. Số khung trang cấp cho mỗi tiến trình không đổi, được xác định vào thời điểm nạp và có thể tùy thuộc vào từng ứng dụng
- C. Số khung trang cấp cho mỗi tiến trình có thể thay đổi trong khi nó chạy
- D. Số khung trang cấp cho mỗi tiến trình có thể thay đổi trong khi nó chạy với lượng cấp phát ban đầu phụ thuộc vào kích thước của nó



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Vấn đề Thrashing

- hiện tượng các trang nhớ của một process bị hoán chuyển vào/ra liên tục
- Nguyên nhân: Khi tổng size of locality > memory size
- Nguyên lý locality (locality principle)
- Locality là tập các trang được tham chiếu gần nhau
- Một process gồm nhiều locality, và trong quá trình thực thi, process sẽ chuyển từ locality này sang locality khác



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Vấn đề Thrashing - Giải pháp tập làm việc

- Được thiết kế dựa trên nguyên lý locality.
- Xác định xem process thực sự sử dụng bao nhiêu frame.
- Định nghĩa:

$WS(t)$  - số lượng các tham chiếu trang nhớ của process gần đây nhất cần được quan sát.

$\Delta$  - khoảng thời gian tham chiếu

- Định nghĩa: Working set của process  $P_i$ , ký hiệu  $WS_i$ , là tập gồm  $\Delta$  các trang được sử dụng gần đây nhất.



Sharing is learning

# Chương 8: Bộ nhớ ảo

HK2(2021-2022) Trong cơ chế quản lý bộ nhớ ảo, đối với mỗi tiến trình, tập gồm  $\Delta$  các trang được sử dụng gần đây nhất ( $\Delta$  là khoảng thời gian tham chiếu) được gọi là gì?

- A. working set
- B. page fault
- C. locality
- D. variable-allocation



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Vấn đề Thrashing - Giải pháp tập làm việc

Nhận xét:

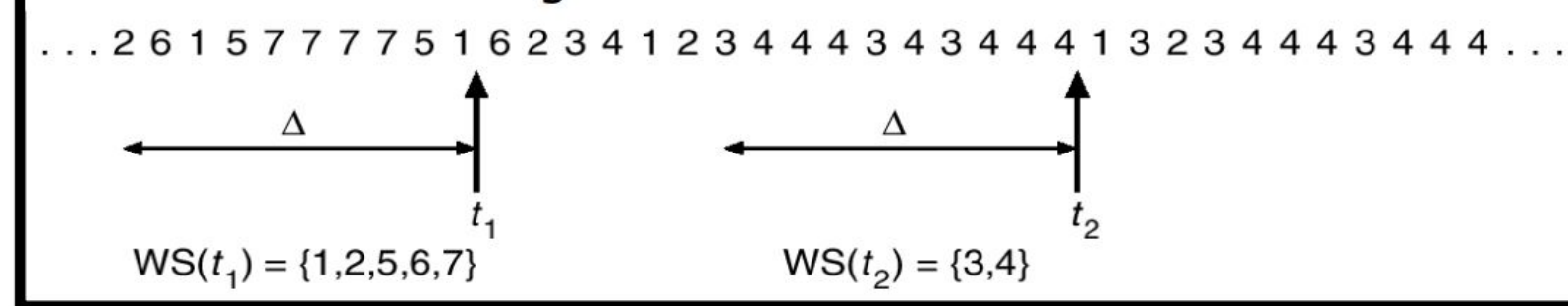
$\Delta$  quá nhỏ  $\Rightarrow$  không đủ bao phủ toàn bộ locality.

$\Delta$  quá lớn  $\Rightarrow$  bao phủ nhiều locality khác nhau.

$\Delta = \infty \Rightarrow$  bao gồm tất cả các trang được sử dụng

Ví dụ:  $\Delta = 10$  và

chuỗi tham khảo trang



Sharing is learning

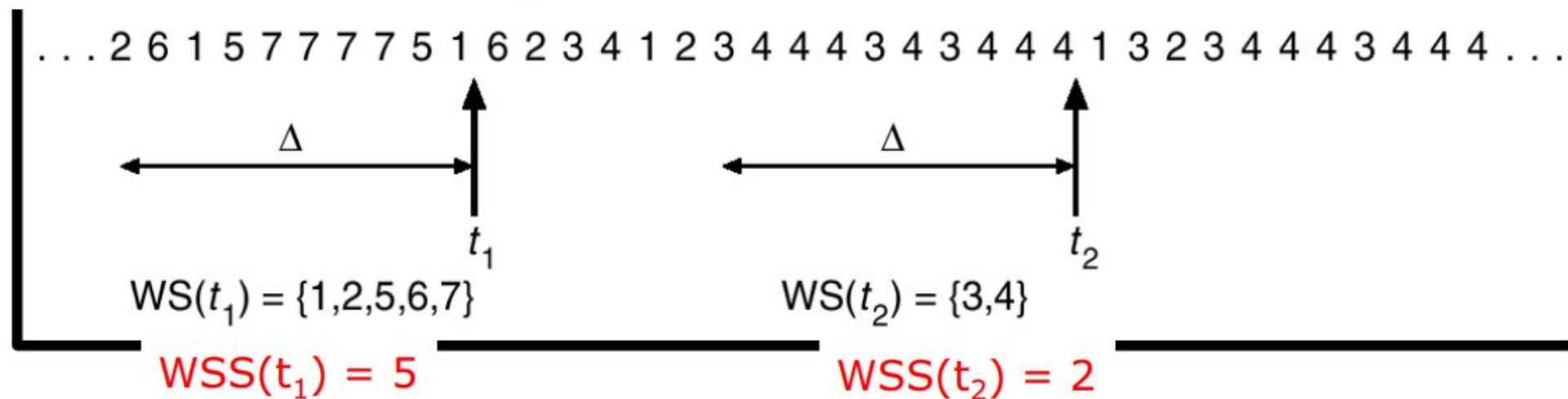
# Chương 8: Bộ nhớ ảo

## Vấn đề Thrashing - Giải pháp tập làm việc

- Định nghĩa:  $WSS_i$  là kích thước của working set của  $P_i$ :

$WSS_i$  = số lượng các trang trong  $WS_i$

Ví dụ:  $\Delta = 10$  và  
chuỗi tham khảo trang



Sharing is learning

# Chương 8: Bộ nhớ ảo

## Vấn đề Thrashing - Giải pháp tập làm việc

- Đặt  $D = \sum WSS_i$  = tổng các working-set size của mọi process trong hệ thống. Nếu  $D > m$  (số frame của hệ thống)  $\Rightarrow$  sẽ xảy ra thrashing.
- Giải pháp working set:
  - Khi khởi tạo một quá trình: cung cấp cho quá trình số lượng frame thỏa mãn working-set size của nó.
  - Nếu  $D > m \Rightarrow$  tạm dừng một trong các process.



Sharing is learning



# Chương 8: Bộ nhớ ảo

HK1(2019-2020) Giải pháp tập làm việc được sử dụng để giải quyết vấn đề gì?

- A. Phát hiện deadlock
- B. Trì trệ trên toàn bộ hệ thống do hoán chuyển trang nhớ
- ☒ C. Đồng bộ hoạt động giữa các tiến trình
- D. Thay thế trang nhớ



Sharing is learning

# FORM ĐIỂM DANH



Sharing is learning

# BAN HỌC TẬP CÔNG NGHỆ PHẦN MỀM

TRAINING GIỮA KỲ HỌC KỲ I NĂM HỌC 2022 – 2023



**Sharing is learning**

# HẾT

CẢM ƠN CÁC BẠN ĐÃ THEO DÕI  
CHÚC CÁC BẠN CÓ KẾT QUẢ THI THẬT TỐT!



*Khoa Công nghệ Phần mềm*

*Trường Đại học Công nghệ Thông tin*

*Đại học Quốc gia thành phố Hồ Chí Minh*



**CONTACT**

*bht.cnpm.uit@gmail.com*

*fb.com/bhtcnpm*

*fb.com/groups/bht.cnpm.uit*