MOT SO BAI TAP CAN CHU Y !!!

Round-Robin, SJFS, Bộ nhớ ảo, Bảng FAT, Thuật giải Nhà băng, Sản xuất-Tiêu thụ (semFull-semEmpty), Dining-Philosopers (deadlock, không deadlock).

BÀI TẬP HỆ ĐIỀU HÀNH

Thuật giải Nhà băng

Câu 1:

Một hệ thống có 3 ổ băng từ và 3 tiến trình P1, P2, P3 với trạng thái cấp phát tài nguyên ở thời điểm Ti thể hiện bằng véc-tơ Allocation = (1, 0, 1) và Max = (1, 2, 2):

Dùng thuật giải nhà băng để:

a. Chứng minh trạng thái này an toàn. (1 điểm)

b. Xác định có nên đáp ứng hay không yêu cầu xin thêm 1 ổ nữa của của P3 ? (1 điểm)

Giải:

a. Xét tại thời điểm Ti mà 3 tiến trình được cấp phát như đề bài ta có:

P[i]	Allocation[i]	Max[i]	Need[i]	Available
P_1	1	1	0	1
P_2	0	2	2	
P_3	1	2	1	

Với: Need[i] = Max[i] – Allocation[i] và Available = 3 – (1 + 0 + 1) = 1

Tìm chuỗi an toàn:

Work >=	Need[i]	P[i]	Allocation[i]
1	0	P1	1
2	2	P2	0
2	1	P3	1

Vậy tại thời điểm T0 tồn tại chuỗi an toàn {P1, P2, P3}. Suy ra, hệ thống tại thời điểm Ti ở trạng thái an toàn.

b. Ta thấy, yêu cầu thêm 1 ổ nữa của P3 thoả các điều kiện:
o Request3 <= Need3 và Request1 <= Available
o Hơn nữa việc cấp phát thêm 1 ổ nữa cho P3 thì hệ thống vẫn ở trạng thái an toàn vì tồn tại chuỗi an toàn {P1, P3, P2} trong khi tài nguyên trong hệ thống không còn nữa. Thật vậy:

Work >=	Need[i]	P[i]	Allocation[i]
0	0	P1	1
1	0	P3	2
3	2	P2	0

Do vậy ta có thể cấp thêm cho yêu cầu xin thêm 1 của P3 tại thời điểm này.

Câu 2

Một hệ thống có 3 ổ băng từ và 3 tiến trình P1, P2, P3 với trạng thái cấp phát tài nguyên tại thời điểm T_i thể hiện bằng các véc-tơ Allocation=(0, 2, 1) và Max=(2, 2, 2).
Dùng thuật giải Nhà băng để:
a. Chứng minh trạng thái này an toàn (1,0 điểm)
b. Xác định có đáp ứng được hay không yêu cầu xin thêm 1 ổ nữa của P2 (1,0 điểm)

Trả lời:

a. Chứng minh trạng thái tại thời điểm T_i an toàn:
- Tính Need = Max – Allocation = (2, 0, 1)
- Tính Available=3-(0+2+1)=0
- Theo thuật giải Nhà băng, tìm được 2 chuỗi an toàn là:

	Work	Need(i)	P(i)	Allocation(i)
Chuỗi an toàn 1				
	0	0	P2	2
	2	2	P1	0
	2	1	P3	1
Chuỗi an toàn 2				
	0	0	P2	2
	2	1	P3	1
	3	2	P1	0

Do tồn tại ít nhất 1 chuỗi an toàn (chuỗi nào cũng được), trạng thái hệ thống tại thời điểm T_i là an toàn.
b. Xác định có đáp ứng được hay không yêu cầu xin thêm 1 ổ nữa của P2:
Không được vì:
- $Need_2$=(2-2)=0, nghĩa là đã hết hạn mức ấn định cho P2.
- Mặt khác, Available=0, nghĩa là hệ không còn ổ băng nào.

Câu 3.

Một hệ thống có 5 tiến trình với tình trạng tài nguyên như sau:

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Duøng thuaät giaûi Nhaø baêng ñeå:

a. Chứng minh trạng thái này an toàn. (1 điểm)

b. Xác định có nên đáp ứng yêu cầu (0, 4, 3, 0) của P_1 ? (1 điểm)

Giải:

a. Xét tại thời điểm T0 mà 5 tiến trình được cấp phát như đề bài ta có:

Need[i] = Max[i] – Allocation[i]

Process	Need			
	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	4	4	2

Tìm chuỗi an toàn:

Work >=	Need[i]	P[i]	Allocation[i]
A B C D	A B C D		A B C D
1 5 2 0	0 0 0 0	**P_0**	0 0 1 2
1 5 3 2	1 0 0 2	**P_2**	1 3 5 4
2 8 8 6	0 0 2 0	**P_3**	0 6 3 2
2 14 11 8	0 4 4 2	**P_4**	0 1 1 4
2 15 12 12	0 7 5 0	**P_1**	1 0 0 0

Vậy tại thời điểm T_0 tồn tại chuỗi an toàn {P_0, P_2, P_3, P_4, P_1}. Suy ra, hệ thống tại thời điểm T_0 ở trạng thái an toàn.

b. Ta thấy, yêu cầu thêm (0, 4, 3, 0) của P_1 thoả điều kiện $Request_1 \leq Need_1$, nhưng không thoả điều kiện: $Request_1 \leq Available$ vì tài nguyên C trong hệ thống chỉ còn 2 mà yêu cầu 3. Do vậy, không thể cấp phát thêm (0, 4, 3, 0) cho P1 được.

Round-Robin, SJFS,

Câu 2

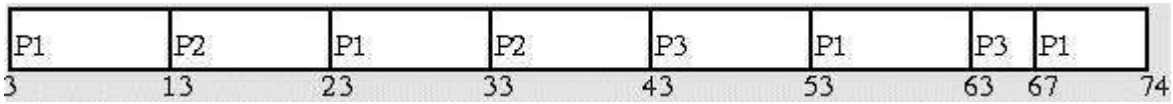
Một hệ thống có 3 tiến trình với thời điểm đến và thời gian sử dụng CPU như sau:

Tiến trình	Thời điểm đến (ms)	CPU-Burst (ms)
P1	3	37
P2	10	20
P3	24	14

Dùng thuật giải Round-Robin với thời lượng 10 ms để điều phối CPU:
a. Thể hiện bằng biểu đồ Gantt (1,0 điểm)
b. Tính thời gian chờ trung bình của các tiến trình (1,0 điểm)

Trả lời:

a. Thể hiện bằng biểu đồ Gantt:



b. Thời gian chờ trung bình của các tiến trình:
(34+13+29)/3 = 76/3 = **25,3** ms

Câu 4

Một hệ thống có 3 tiến trình với thời điểm đến và thời gian sử dụng CPU như sau:

Tiến trình	Thời điểm đến (ms)	CPU-Burst (ms)
P1	5	47

P2	23	15
P3	45	28

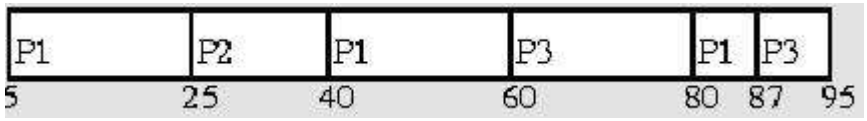
Dùng thuật giải Round-Robin với thời lượng bằng 20 ms để điều phối CPU:

a. Thể hiện bằng biểu đồ Gantt (0,5 điểm)

b. Tính thời gian chờ trung bình của các tiến trình (0,5 điểm)

Trả lời:

a. Thể hiện bằng biểu đồ Gantt:



b. Tính thời gian chờ trung bình của các tiến trình:

- Thời gian chờ của các tiến trình:

P1 = **35** ms

P2 = **2** ms

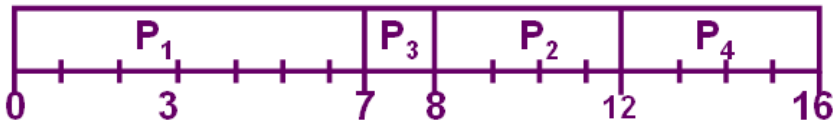
P3 = **22** ms

- Thời gian chờ trung bình = (35 + 2 + 22) / 3 = 59 / 3 = **19,66** ms

SJFS không tiếm quyền
(Non-Preemptive SJFS)

Thời điểm	Tiến trình	Khoảng CPU kế tiếp
0	P_1	7
2	P_2	4
4	P_3	1
6	P_4	4

- Biểu đồ Gantt:

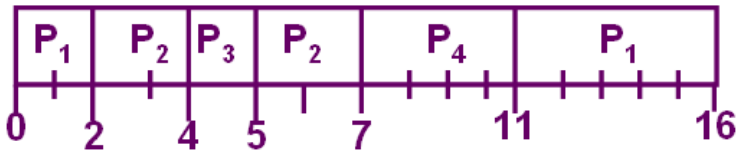


- Thời gian chờ trung bình: (0 + (8-2) + (7-4) + (12-6)) / 4 = 3,75 ms
- Ứng dụng trợ giúp giải bài tập

SJFS có tiếm quyền
(Preemptive SJFS)

Thời điểm	Tiến trình	Khoảng CPU kế tiếp
0	P_1	7
2	P_2	4
4	P_3	1
5	P_4	4

- Biểu đồ Gantt:



- Nếu tiến trình mới đến có khoảng CPU kế tiếp nhỏ hơn so với thời gian còn lại của tiến trình đang vận hành, nó sẽ được ưu tiên chạy thay thế.
- Thời gian chờ trung bình: (9 + 1 + 0 + 2) / 4 = 3 ms
- Ứng dụng trợ giúp giải bài tập

bài tập phân đoạn, tính địa chỉ vật lý cho địa chỉ logic

Câu 8 (1 điểm)

Giả sử trong quá trình quản lý bộ nhớ ảo dạng phân đoạn, hệ điều hành duy trì Segment Table:

Segment	Base	Limit
0	300	700
1	1200	500
2	2000	600

Hãy tính địa chỉ vật lý cho mỗi địa chỉ lô-gic sau: (1, 200), (1, 0), (0, 700), (2, 0), (2, 600)

sau khi tìm hiểu về bài tập này, mình post lên cho mọi người cùng trao đổi 

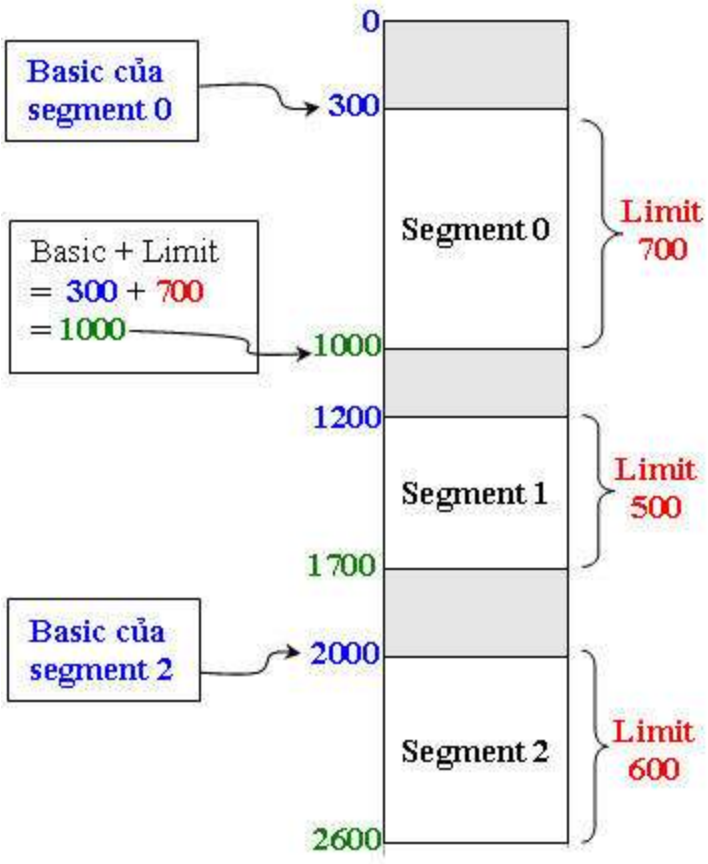
GIẢI

1. Vẽ vùng bộ nhớ Vật lý dạng các đoạn segment

Từ bảng dữ liệu đề bài

segment	Base	Limit
0	300	700
1	1200	500
2	2000	600

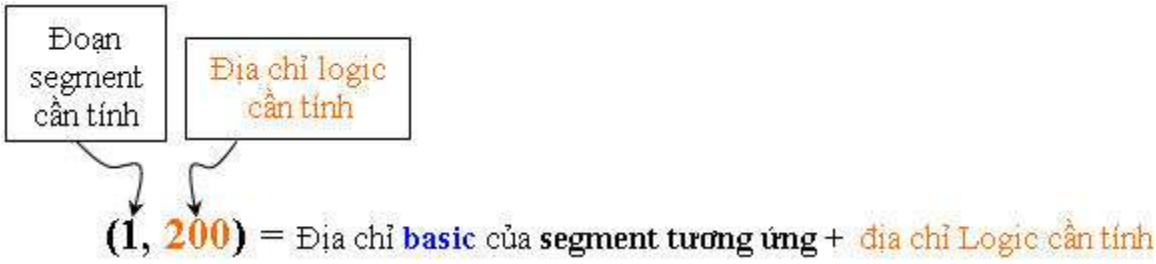
Ta vẽ được vùng bộ nhớ vật lý như sau:



Các bạn nhìn vào hình mình đã hướng dẫn cách tính và cách vẽ, các bạn chú ý phần màu chữ mình sử dụng để nhận ra dễ hơn.
Với **segment 0**: ta có
+ Địa chỉ vật lý cơ sở (basic) là 300
+ Limit là 700
==> địa chỉ vật lý của **segment 0** là từ 300 -> 1000
Với **segment 1**:
+ Địa chỉ vật lý cơ sở (basic) là 1200, nên ta sẽ vẽ bắt đầu từ 1200, như vậy từ 1000->1200 là trống, không có segment nào
+ Limit là 500
==> địa chỉ vật lý của **segment 1**là từ 1200 -> 1700

và **segment 2** các bạn tính tương tự

2. Cách tính địa chỉ logic



Tính địa chỉ vật lý
+ Với dữ liệu đề bài cho là (1,200), ta xác định: tính địa chỉ vật lý của segment 1, địa chỉ logic là 200 (lưu ý: giá trị X tính được này nằm trong segment 1 hay ko (1200<= X<= 1700))
===> (1,200) = 1200 + 200 = 1400 (hợp lệ vì < 1700)
+ (1,0) = 1200 + 0 = 1200 (hợp lệ)
+ (0,700) = 300 + 700 = 1000 (hợp lệ)
+ (2,0) = 2000 + 0 = 2000 (hợp lệ)
+ (2, 600) = 2000 + 600 = 2600 (hợp lệ)

Nhận xét: Để tính địa chỉ vật lý cho các địa chỉ logic thì các địa chỉ logic này hok được vượt quá giới hạn (limit) của segment tương ứng đang xét

Mong các bạn trao đổi để chúng ta ôn tập được tốt hơn nha 

Một số trường hợp không hợp lệ mình đưa ra như sau:
+ (0,800) --> không hợp lệ vì 800 > 700 (limit của segment 0)
+ (2,650) --> không hợp lệ vì 650 > 600 (limit của segment 2)
+ (1,501) --> không hợp lệ vì 501 > 500 (limit của segment 1)
Tóm lại nếu nhìn thấy địa chỉ logic mà đề bài cho > limit của segment đó thì kết luận ngay là không hợp lệ

bài tập phân đoạn, tính địa chỉ vật lý cho địa chỉ logic có trường hợp không hợp lê

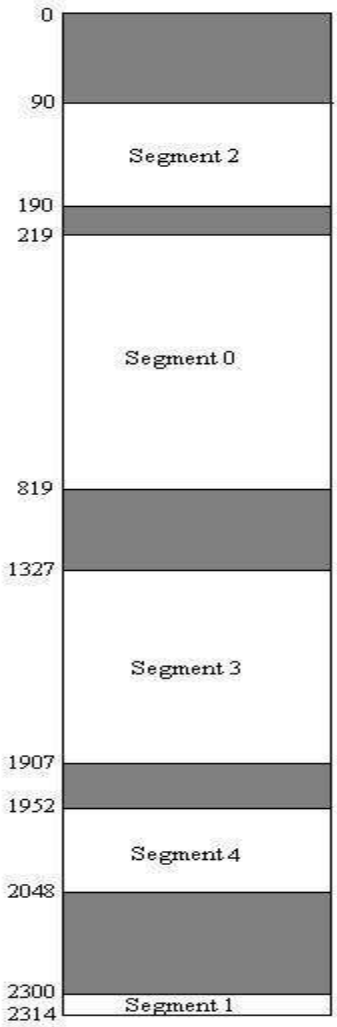
Giả sử có Bảng đoạn sau:

Segment	Base	Limit
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Hãy tính địa chỉ vật lý cho mỗi địa chỉ lô-gic sau:

a. 0430
b. 1010
c. 2500
d. 3400
e. 4112
giải

Tính địa chỉ vật lý
+ Với dữ liệu đề bài cho là (0,430) = 219 +430 = 649 (hợp lệ) Vì nó nằm trong đoạn Segment 0.
+ (1,010) = 2300+ 10 = 2310 (hợp lệ)
+ (2,500) = 90 + 500 = 1400 (không hợp lệ)
+ (3,400) = 1327+ 400 = 1727 (hợp lệ)
+ (4,112) = 1952 + 112 = 2064 (không hợp lệ)

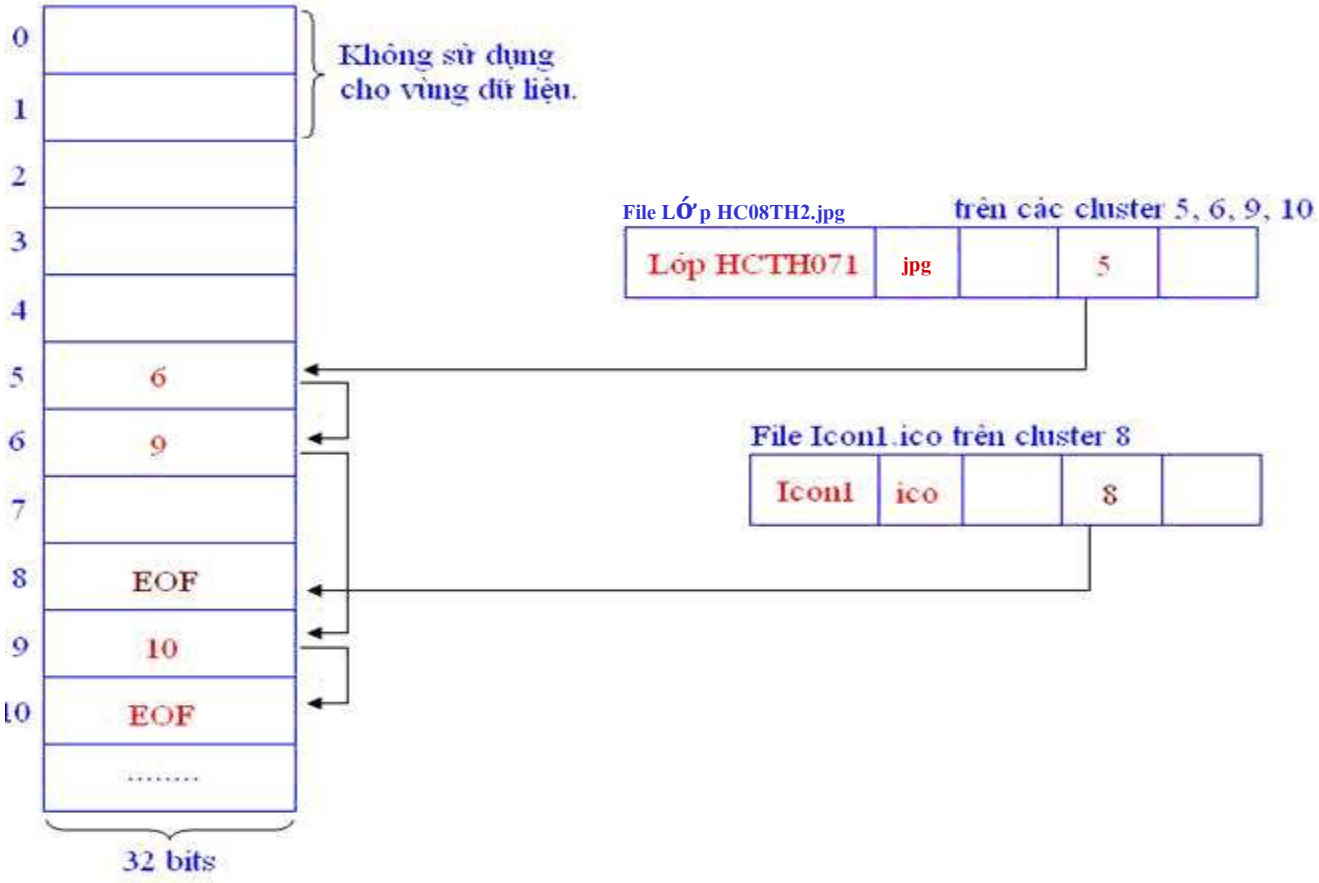


=>Với các địa chỉ logic (0,430); (1,010); (1,500); (3,400); (4,112) ta có các địa chỉ vật lý tương ứng là 649; 2310; không hợp lệ;1727; không hợp lệ.

Bảng FAT,

Trên một hệ tập tin FAT32, tập tin Lớp HC08TH2.jpg có nội dung trải trên các liên cung 5, 6, 9, 10; trong khi Icon1.ico chỉ cần liên cung 8. Hãy thể hiện bằng hình vẽ cấu trúc bảng FAT và các Directory Entry.

Giải:



Cần đến 2 Directory Entry cho Lớp HC08TH2.jpg

Bytes														
2	. j p g			A	0	CK						0		
1	Lớp			A	0	CK	H C 0 8 T					0	H 2	
L	o p H C 0 ~ 1			A	NT	S	Creation time		Last acc	Upp	Last write	Low	Size	

Ghi chú: Lớp gồm 5 ký tự Lowps.

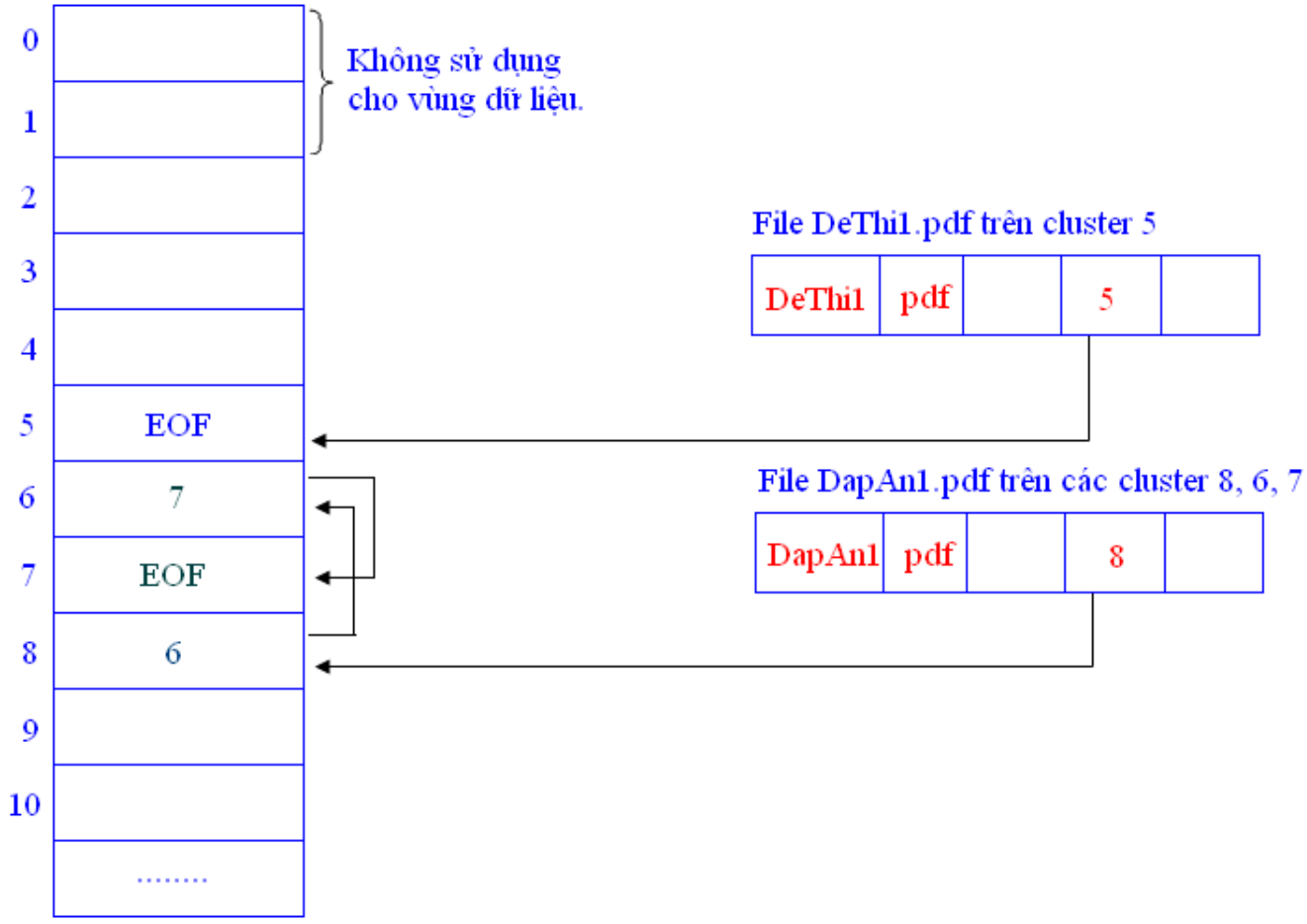
H C 0 8 T có 5 ký tự + 1 ký tự khoảng trắng (_) phía trước là 6.

Bytes	8	3	1	1	1	4	2	2	4	2
4										
Icon1.ico	Ext		N T		Creation date/time	Last acces		Last write date/time		File Size

Câu 7 (1 điểm)

Trên một hệ tập tin FAT32, tập tin DeThi1.pdf có nội dung tại liên cung 5, trong khi DapAn1.pdf cần các liên cung 8, 6, 7. Hãy thể hiện bằng hình vẽ cấu trúc bảng FAT và các Directory Entry.

Giải:



RAG

Một hệ thống có 1 máy in laser và 1 ổ băng từ. Hai tiến trình P1 và P2 đang vận hành với trạng thái cấp phát tài nguyên như sau:

Tiến trình	Máy in laser	Ổ băng từ
P1	Yêu cầu	Được cấp
P2	Được cấp	Yêu cầu

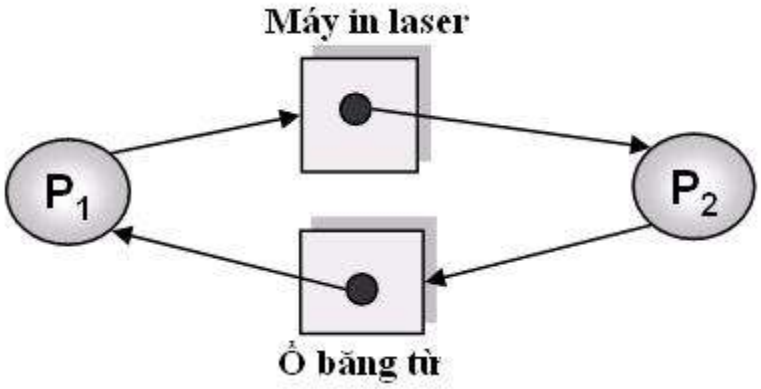
Hãy:

a. Thể hiện bằng RAG

b. Xác định và giải thích trạng thái này.

Giải:

a. Đồ thị cấp phát tài nguyên RAG:



b. Trạng thái này là trạng thái Deadlock .vì mỗi tài nguyên chỉ có một phiên bản và tồn tại chu trình hay vòng tròn khép kín các yêu cầu tài nguyên.

Bộ nhớ ảo

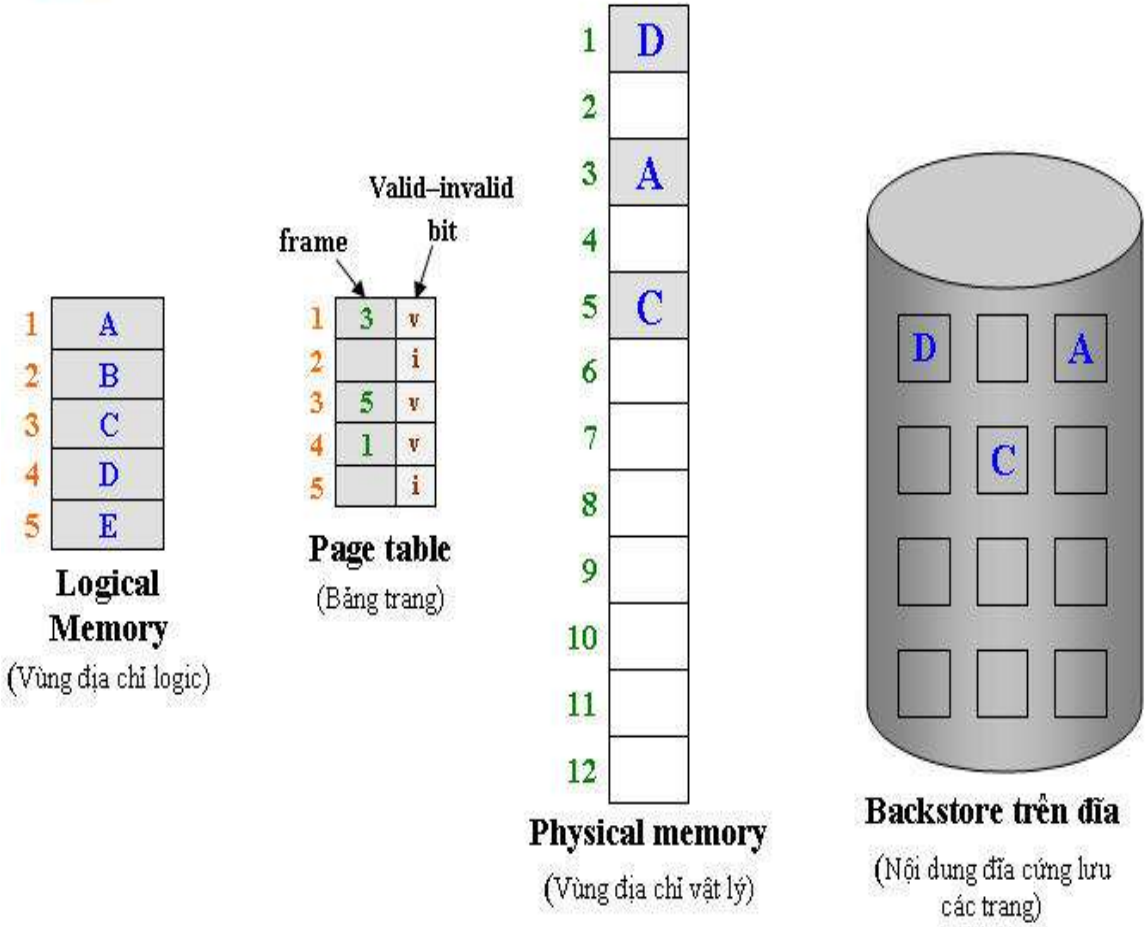
Đề bài: Vùng địa chỉ lô-gic (vùng nhớ ảo) của một tiến trình được chia thành các trang A, B, C, D, E được ánh xạ sang Vùng địa chỉ vật lý theo bảng sau:

STT	Trang lô-gic	Khung trang số
1	A	3
2	B	
3	C	5
4	D	1
5	E	

Hãy minh hoạ bằng hình vẽ các thành phần sau:

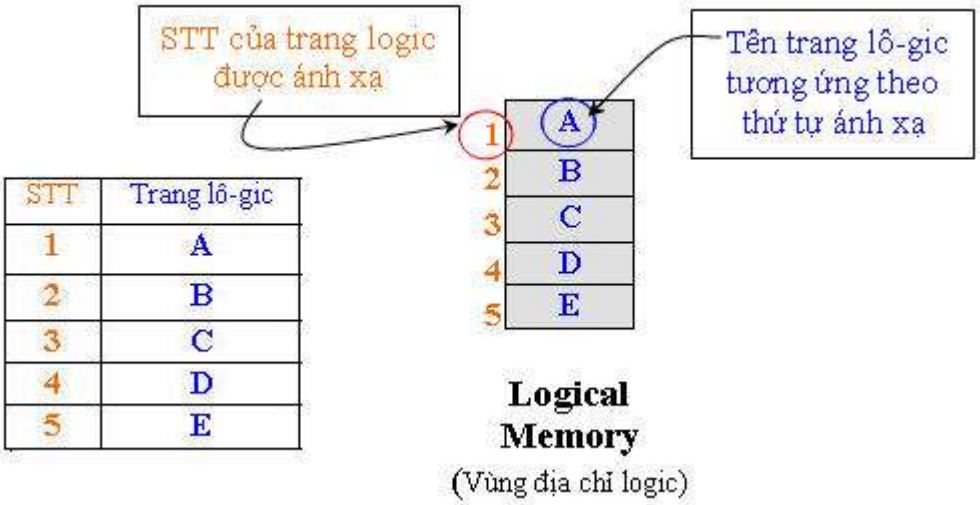
- Vùng địa chỉ lô-gic
- Bảng trang (sử dụng bit "Đúng-Sai")
- Vùng địa chỉ vật lý
- Nội dung đĩa cứng lưu các trang.

Giải:

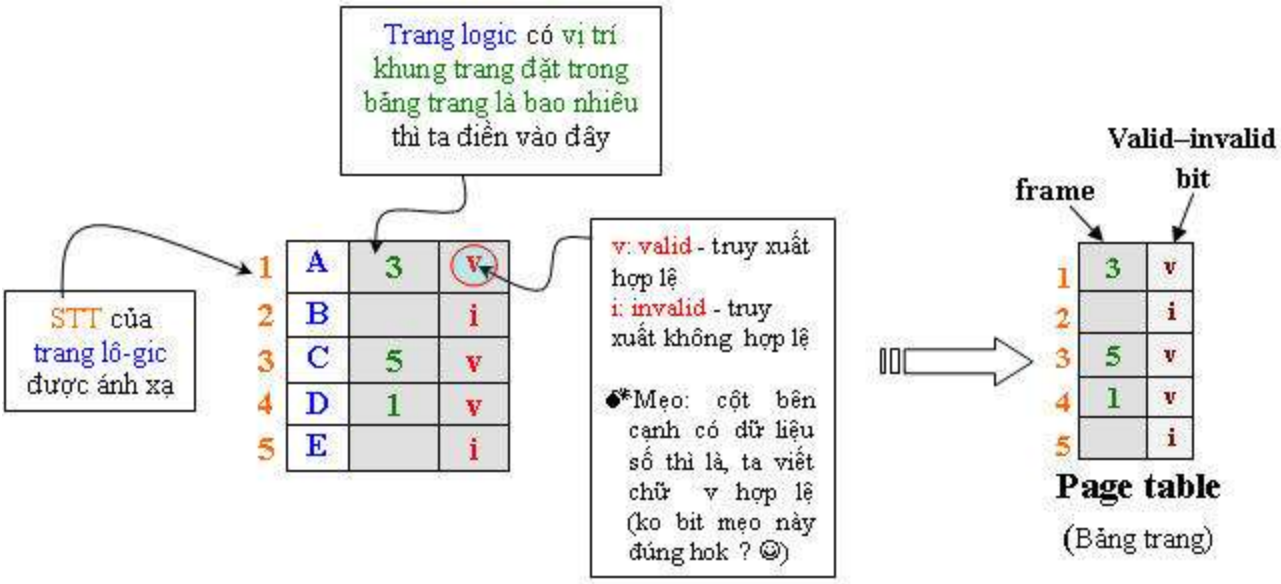


Với đề bài và cách giải trên thì mình xin làm và giải thích theo cách hiểu của mình
Bạn nào có cách giải thích rõ hơn, chính xác hơn thì bổ sung nhé

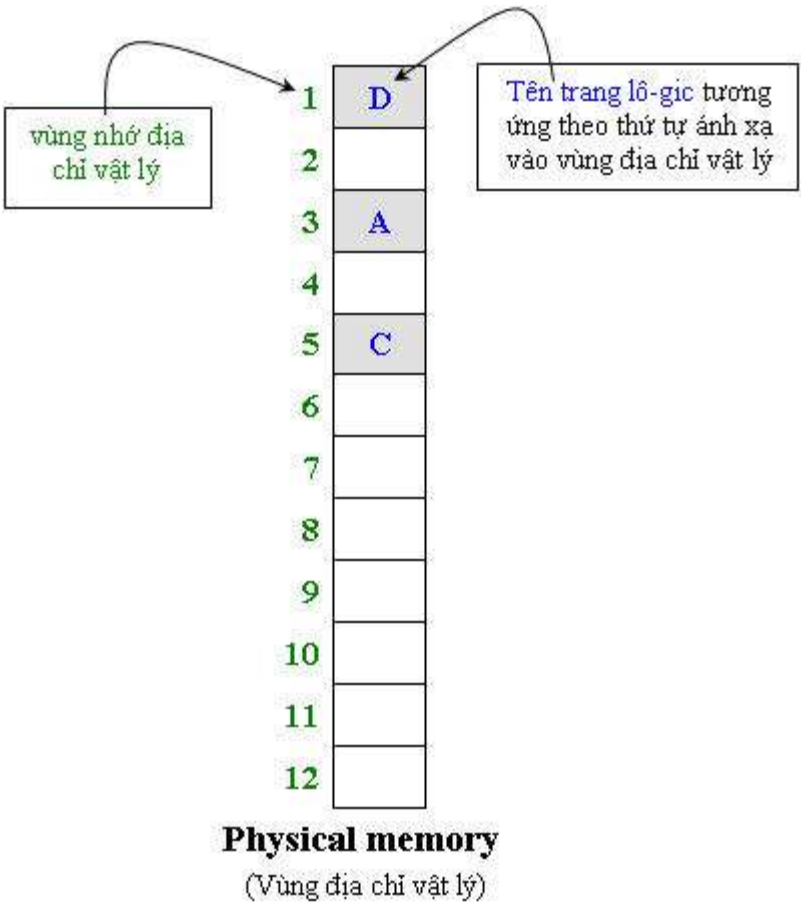
+ Hình 1: Logical Memory



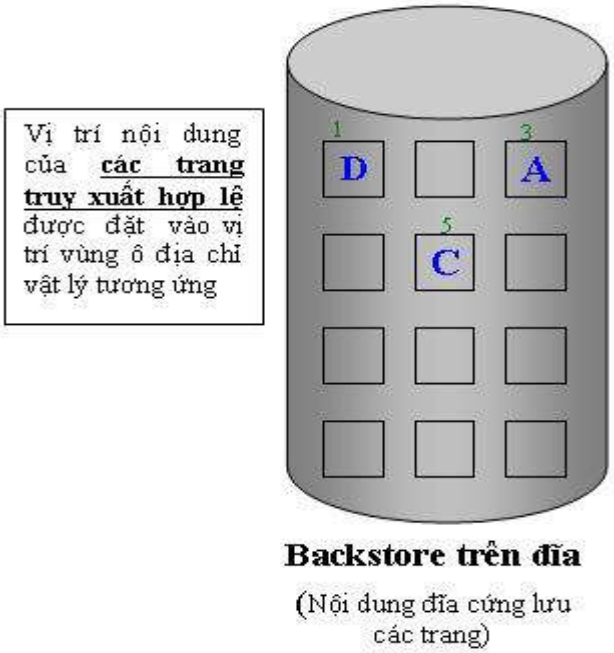
+ Hình 2: Page table



+ Hình 3: Physical memory



+ Hình 4: Backstore



Một hệ thống có Bộ nhớ trong chia thành 8 khung trang với Khung 0 dành cho Hệ điều hành và các khung còn lại dành cho 2 tiến trình đang vận hành là P0 (gồm các trang C, D, E, F) và P1 (gồm các trang O, P, Q, R). Bằng hình vẽ, với kỹ thuật tổ chức bộ nhớ ảo dạng phân trang, hãy tìm cách:
a. Phân bổ ngẫu nhiên các trang của P0 và P1 vào Bộ nhớ trong kể trên
b. Tổ chức lại các bảng trang sao cho trang chưa nạp (do hết chỗ) bây giờ được nạp

Câu a: phân bố ngẫu nhiên

P0:

0	C
1	D
2	E
3	F

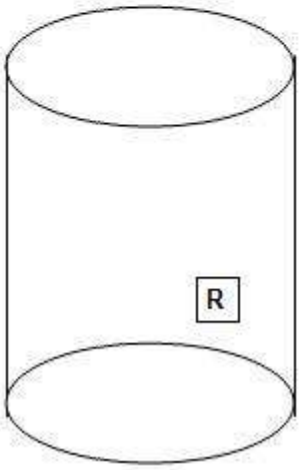
0	1	v
1	3	v
2	4	v
3	6	v

0	
1	C
2	O
3	D
4	E
5	P
6	F
7	Q

P1:

0	O
1	P
2	Q
3	R

0	2	v
1	5	v
2	7	v
3		i



Câu b: phân bố lại

P0:

0	C
1	D
2	E
3	F

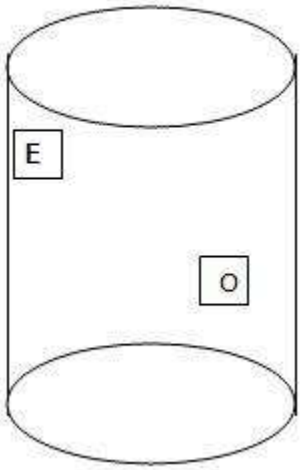
0	1	v
1	3	v
2		i
3	6	v

0	
1	C
2	O
3	D
4	O
5	P
6	F
7	Q

P1:

0	O
1	P
2	Q
3	R

0	2	v
1	5	v
2	7	v
3	4	v



Sản xuất-Tiêu thụ (semFull-semEmpty),

Phát biểu bài toán Sản xuất-Tiêu thụ với thuật giải đồng bộ hoá bằng 2 đèn hiệu semFull và semEmpty.

Giải:

Bài toán người sản xuất-người tiêu thụ (Producer-Consumer) thường được dùng để hiển thị sức mạnh của các hàm cơ sở đồng bộ hoá. Hai quá trình cùng chia sẻ một vùng đệm có kích thước giới hạn n. Biến semaphore **mutex** cung cấp sự loại trừ hỗ tương để truy xuất vùng đệm và được khởi tạo với giá trị **1**. Các biến semaphore **empty** và **full** đếm số khe trống và đầy tương ứng. Biến semaphore **empty** được khởi tạo tới giá trị **n**; biến semaphore **full** được khởi tạo tới giá trị **0**.

- **Dữ liệu chia sẻ:**

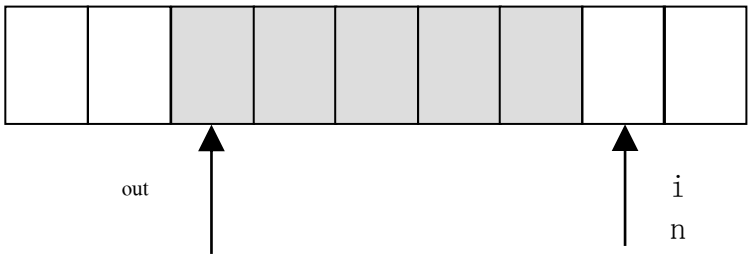
  SEMAPHORE full, empty, mutex;

- **Khởi tạo:**

  full = 0;

empty = BUFFER_SIZE;
mutex = 1;

Buffer xoay vòng



PRODUCER

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

CONSUMER

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

Câu 2 (1 điểm)

Phát biểu bài toán Sản xuất-Tiêu thụ với thuật giải đồng bộ hoá bằng 3 đèn hiệu semFull, semEmpty và Mutex.

Trả lời:

- Tiến trình sản xuất (Producer) tạo ra dòng thông tin để tiến trình tiêu thụ (Consumer) sử dụng.

- Ví dụ: Compiler và Assembler vừa là nhà sản xuất vừa là nhà tiêu thụ. Compiler tạo ra mã dùng cho Assembler, tiếp theo Assembler sản sinh mã máy làm đầu vào cho Loader hoặc Linkage Editor.

- Phát biểu bài toán: Bộ nhớ đệm Buffer bao gồm một số hữu hạn các khoang chứa (Items). Producer lần lượt đưa các sản phẩm S1, S2,…vào các khoang của Buffer. Consumer lấy sản phẩm ra theo đúng thứ tự. Công việc của các tiến trình phải đồng bộ với nhau: không đưa ra sản phẩm khi hết chỗ trống, không lấy được sản phẩm khi chưa có.

- Thuật giải đồng bộ hoá bằng 3 đèn hiệu: semFull (quản lý số sản phẩm có trong bộ đệm, giá trị ban đầu bằng 0), semEmpty (quản lý số khoang còn trống, giá trị ban đầu bằng số khoang của bộ đệm) và Mutex (đảm bảo tính loại trừ tương hỗ, nghĩa là mỗi thời điểm chỉ có 1 tiến trình sản xuất hay tiêu thụ được truy cập/cập nhật tài nguyên dùng chung, giá trị ban đầu bằng 1).

o Thuật giải cho Producer:

```
wait(semEmpty);

wait(Mutex);

// Đưa sản phẩm vào Buffer

.....

signal(semFull);

signal(Mutex);
```

o Thuật giải cho Consumer:

```
wait(semFull);

wait(Mutex);

// Lấy sản phẩm từ Buffer

.....

signal(semEmpty);
signal(Mutex);
```

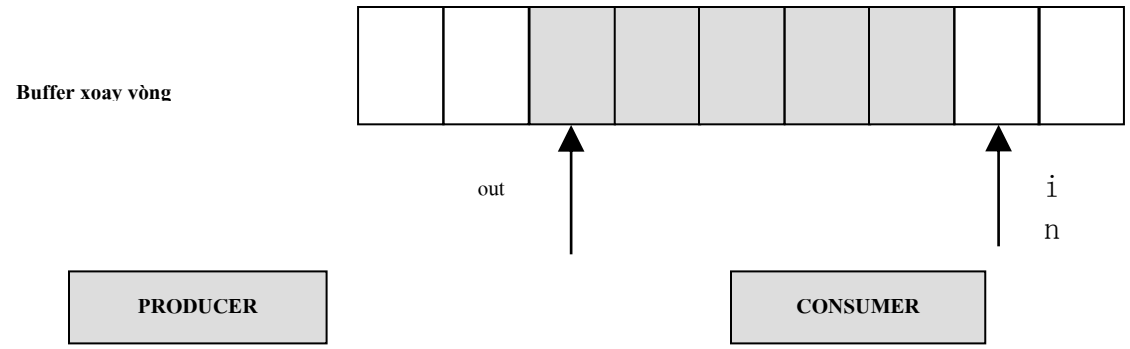
4.1. Phát biểu bài toán Sản xuất-Tiêu thụ và trình bày Thuật giải với Bộ đệm thực thi bằng mảng xoay vòng.

Giải:

Phát biểu bài toán:

- Giả sử có Bộ nhớ đệm (Buffer) bao gồm nhiều khoang (Items) được tiến trình Producer lần lượt đưa các sản phẩm S_1, S_2,... vào.
- Tiến trình Consumer lần lượt lấy sản phẩm ra theo đúng thứ tự.
- Công việc của Producer phải đồng bộ với Consumer: Không được đưa sản phẩm vào khi Buffer đầy, Không được lấy ra khi chưa có.

Trình bày giải thuật:




```
item nextProduced;
while (1)
{
        while(((in+1)%BUFFER_SIZE)==out); //quẩn tại đây khi buffer đầy.
        buffer[in] = nextProduced;
        in = (in+1)%BUFFER_SIZE;
```

```
item nextConsumed;
while (1)
{
        while(in==out); //quẩn khi buffer rỗng
        nextConsumed = buffer[out];
        out = (out+1)%BUFFER_SIZE;
}
```

Dining-Philosopers (deadlock, không deadlock).

7.1. Phân tích thuật giải sai bài toán Dining-Philosophers (dẫn đến Deadlock).

Giải:

Dữ liệu chia sẻ:

semaphore chopstick[5];

Khởi đầu các biến đều là: **1**.

```
while (1)
{
        wait(chopstick[i])
        wait(chopstick[(i+1) % 5 ] )
         …
        eat
         …
        signal(chopstick[i]);
        signal(chopstick[(i+1) % 5] );
         …
        think
         …
}
```

Giải pháp trên có thể gây ra deadlock

Khi tất cả triết gia đói bụng cùng lúc và đồng thời cầm một chiếc đũa bên tay trái ⇒ deadlock

Có thể xảy ra trường hợp ách vô hạn định (starvation).

7.2. Phân tích thuật giải đúng bài toán Dining-Philosophers (dùng đèn hiệu).

Giải:

Dữ liệu chia sẻ:

semaphore chopstick[5];

Khởi đầu các biến đều là: **1**.

HANDLE cs[2]

```
while (1)
{
        cs[0] = chopstick[i];
        cs[0] = chopstick[(i+1) % 5 ]
        WaitForMultipleObjects(2, cs, TRUE, INFINITE);
         …
        eat
         …
        ReleaseMutex(chopstick[i];
        ReleaseMutex(chopstick[(i+1) % 5 ])
         …
        think
         …
}
```