Lý thuyết hệ điều hành

Giảng viên: TS. Hà Chí Trung

Bộ môn: Khoa học máy tính

Khoa: Công nghệ thông tin

Học viện Kỹ thuật quân sự

Email: hct2009@yahoo.com

Mobile: 01685.582.102

Chương 3. Quản lý bộ nhớ

3.6. Kỹ thuật phân trang (Paging)

3.6.1. Sử dụng thanh ghi truy cập nhanh

3.6.2. Lưu bảng phân trang trong bộ nhớ chính

3.6.3. Sử dụng thanh nhớ kết hợp CAAR

3.7. Chia sẻ trang (Sharing Pages)

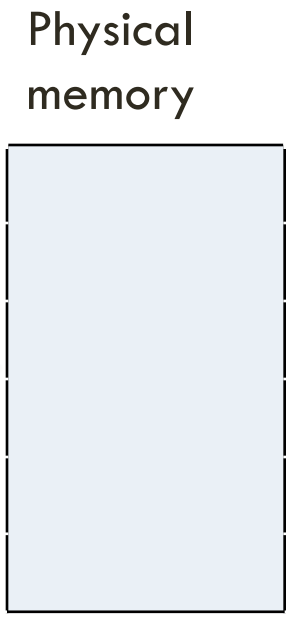
3.8. Kỹ thuật phân đoạn (Segmentation)

3.9. Chia sẻ đoạn (Sharing Segments)

3.10. Kết hợp phân đoạn với phân trang

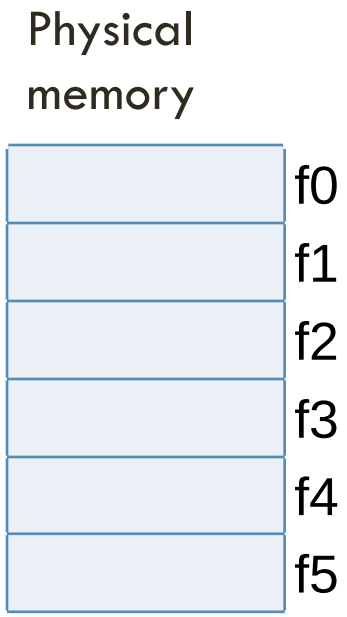
3.6. Kỹ thuật phân trang (Paging)

HĐH chia bộ nhớ vật lý thành các frames có kích cỡ nhỏ và cố định

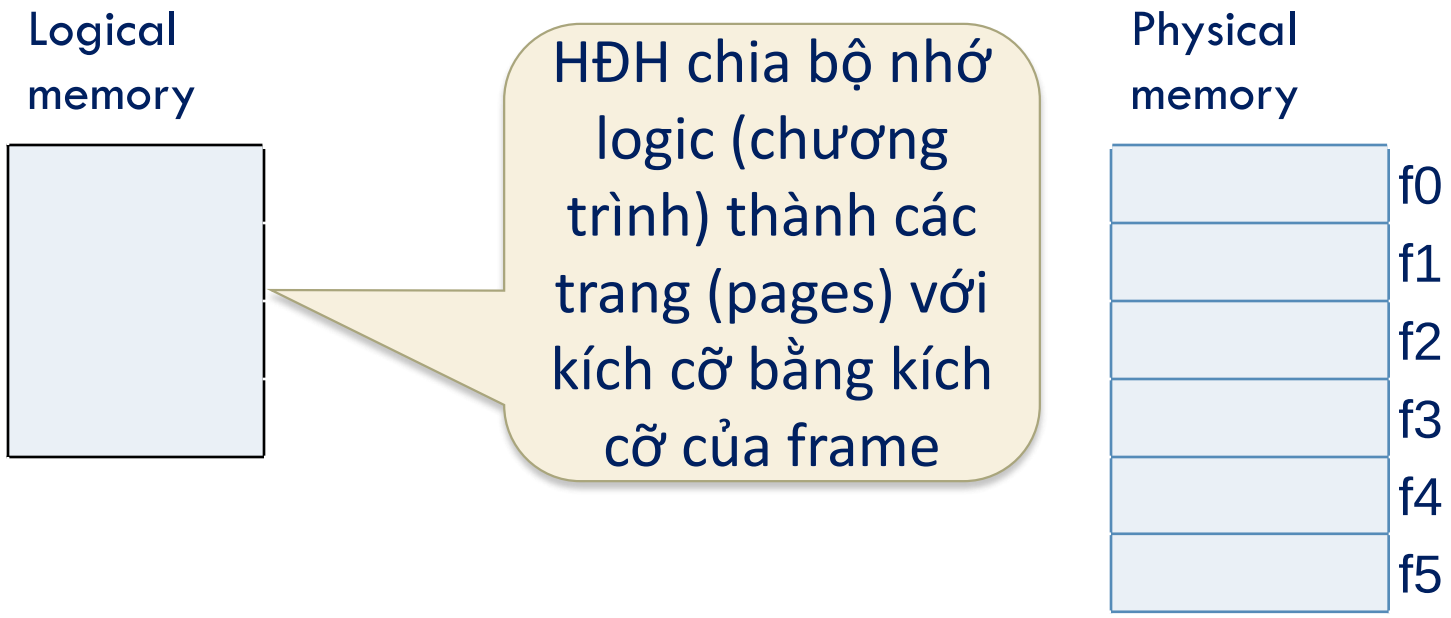


3.6. Kỹ thuật phân trang (Paging)

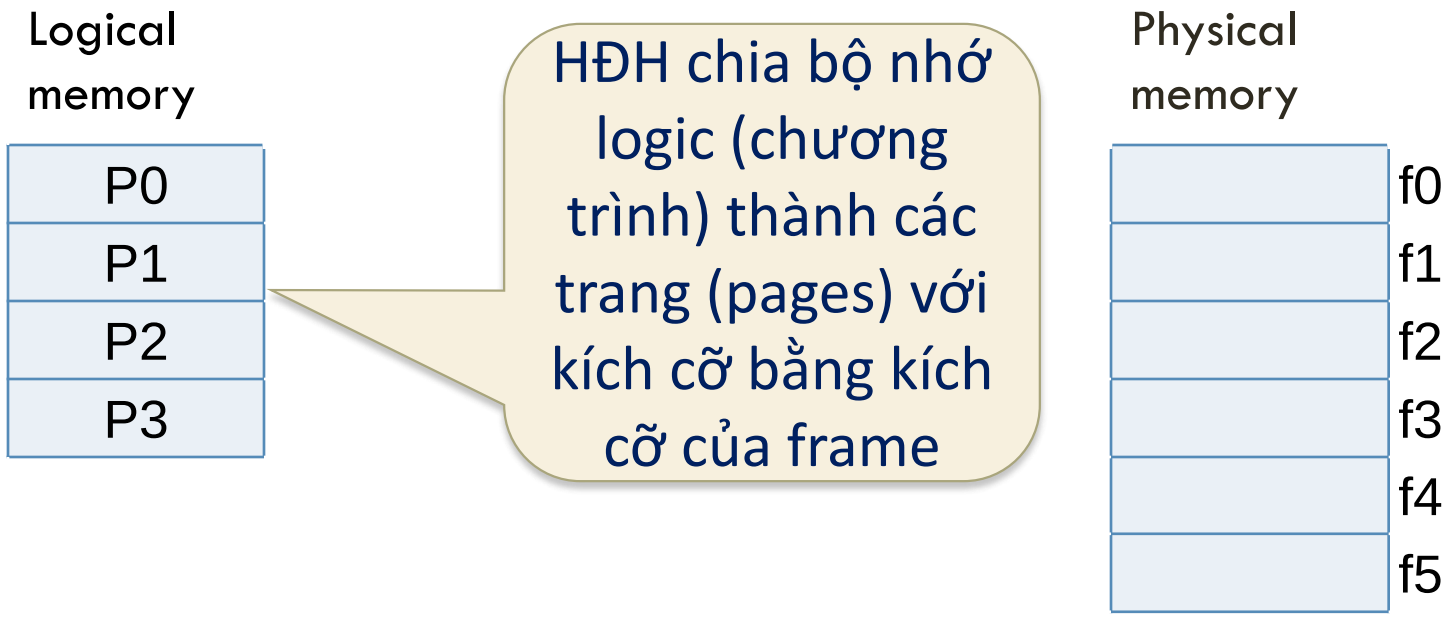
HĐH chia bộ nhớ vật lý thành các frames có kích cỡ nhỏ và cố định



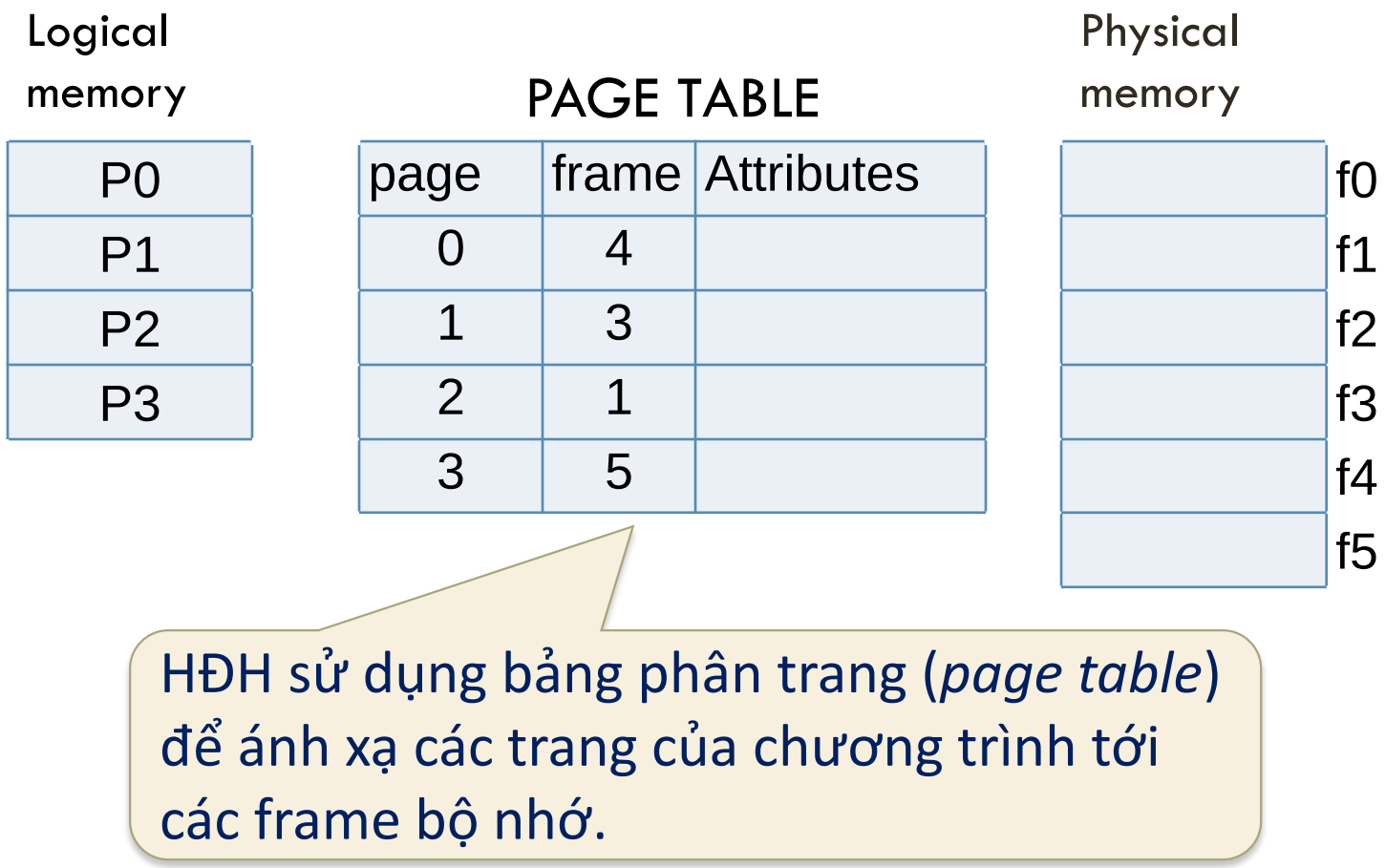
3.6. Kỹ thuật phân trang (Paging)



3.6. Kỹ thuật phân trang (Paging)

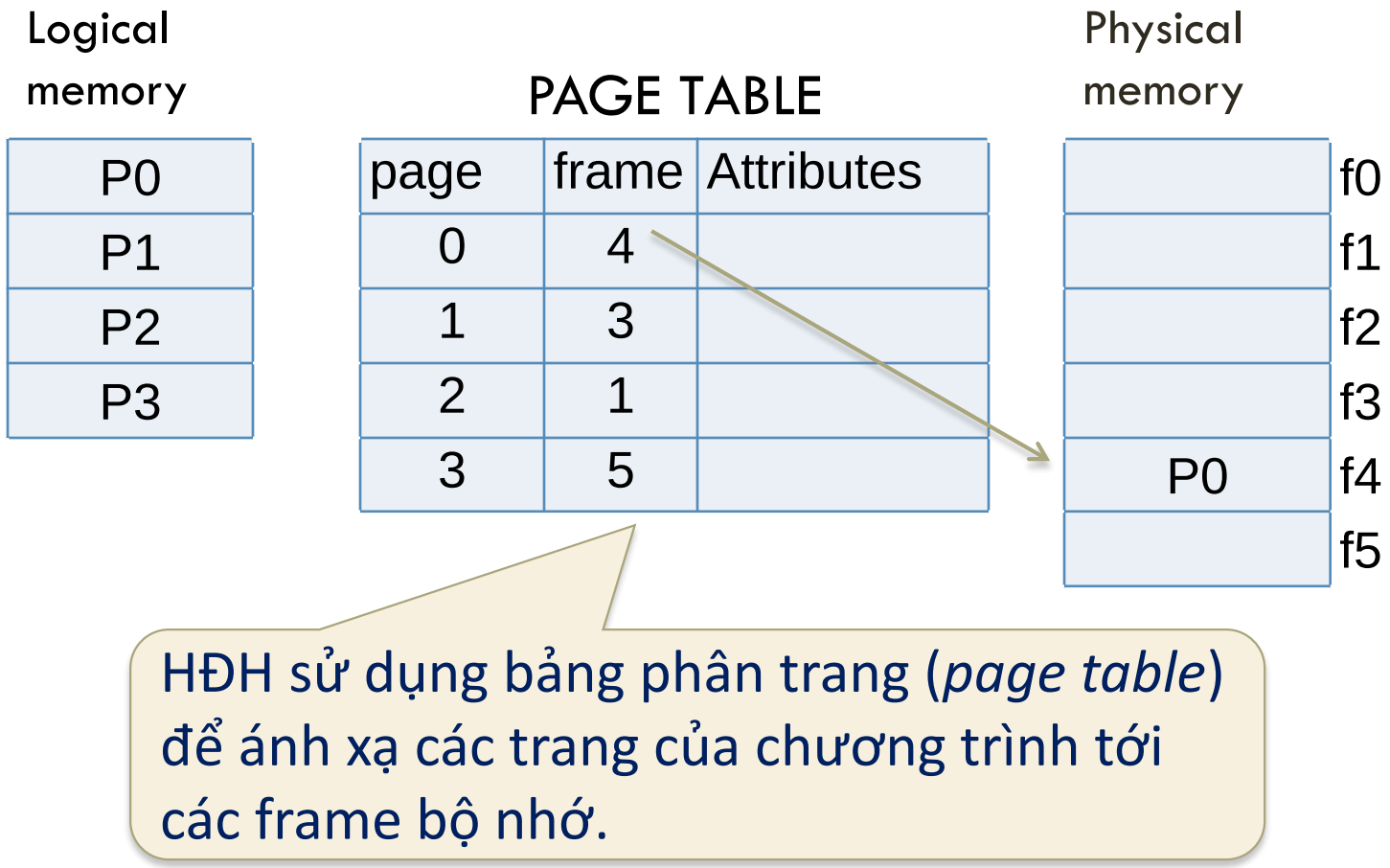


3.6. Kỹ thuật phân trang (Paging)



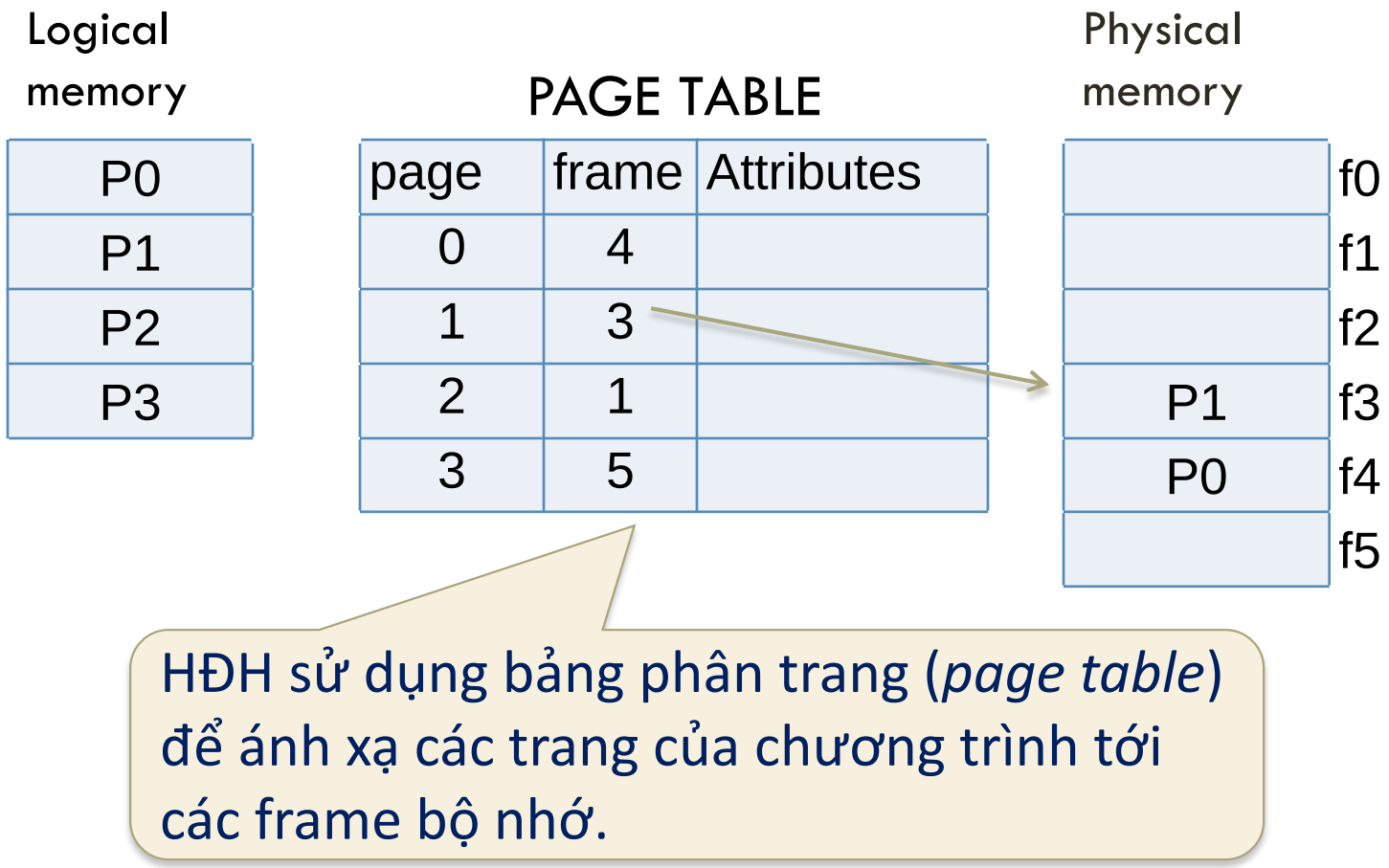
HĐH sử dụng bảng phân trang (*page table*) để ánh xạ các trang của chương trình tới các frame bộ nhớ.

3.6. Kỹ thuật phân trang (Paging)



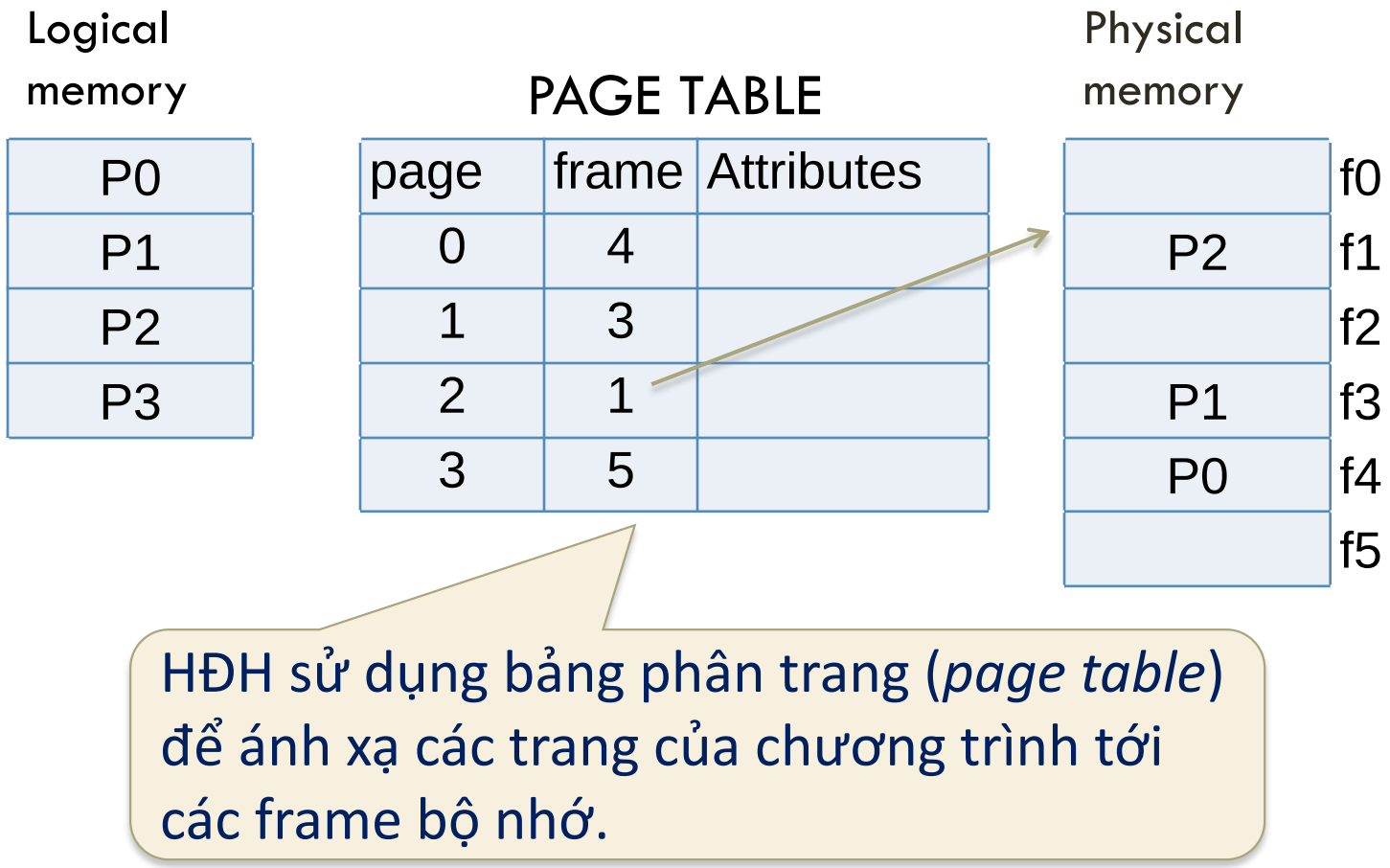
HĐH sử dụng bảng phân trang (*page table*) để ánh xạ các trang của chương trình tới các frame bộ nhớ.

3.6. Kỹ thuật phân trang (Paging)



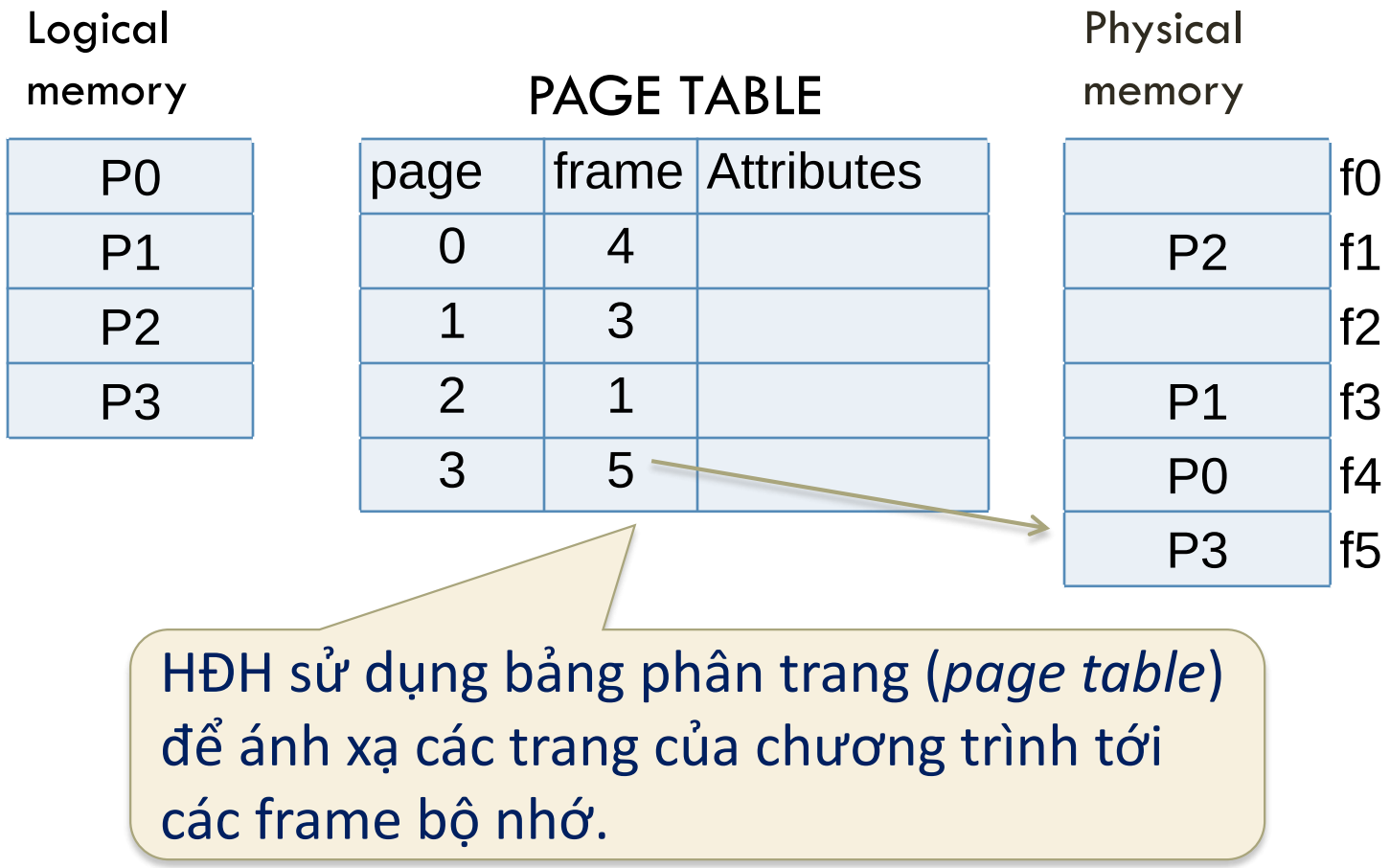
HĐH sử dụng bảng phân trang (*page table*) để ánh xạ các trang của chương trình tới các frame bộ nhớ.

3.6. Kỹ thuật phân trang (Paging)



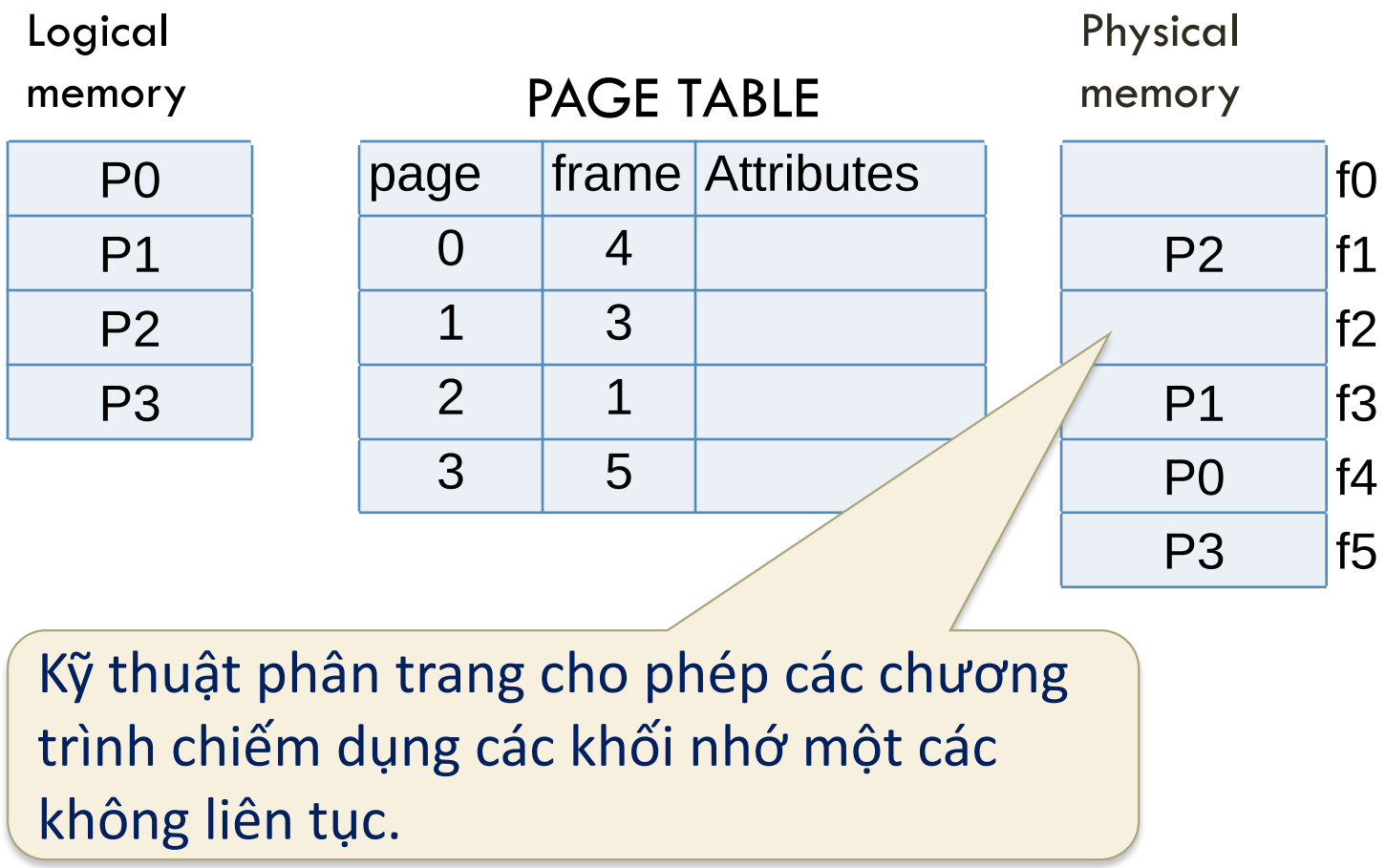
HĐH sử dụng bảng phân trang (*page table*) để ánh xạ các trang của chương trình tới các frame bộ nhớ.

3.6. Kỹ thuật phân trang (Paging)



HĐH sử dụng bảng phân trang (*page table*) để ánh xạ các trang của chương trình tới các frame bộ nhớ.

3.6. Kỹ thuật phân trang (Paging)



Kỹ thuật phân trang cho phép các chương trình chiếm dụng các khối nhớ một các không liên tục.

3.6. Kỹ thuật phân trang (Paging)

- Kích cỡ trang (S) được định nghĩa và hỗ trợ bởi phần cứng. Thông thường kích cỡ trang được lựa chọn là lũy thừa của 2, chẳng hạn 512 words/page hoặc 4096 words/page,…
- Với cách tổ chức này, các words trong chương trình có một địa chỉ logic (LA) dưới dạng một cặp như sau:

<p,d>

- Trong đó p là số hiệu của trang, p = LA div S; d là độ dịch chuyển (offset), d = LA mod S.

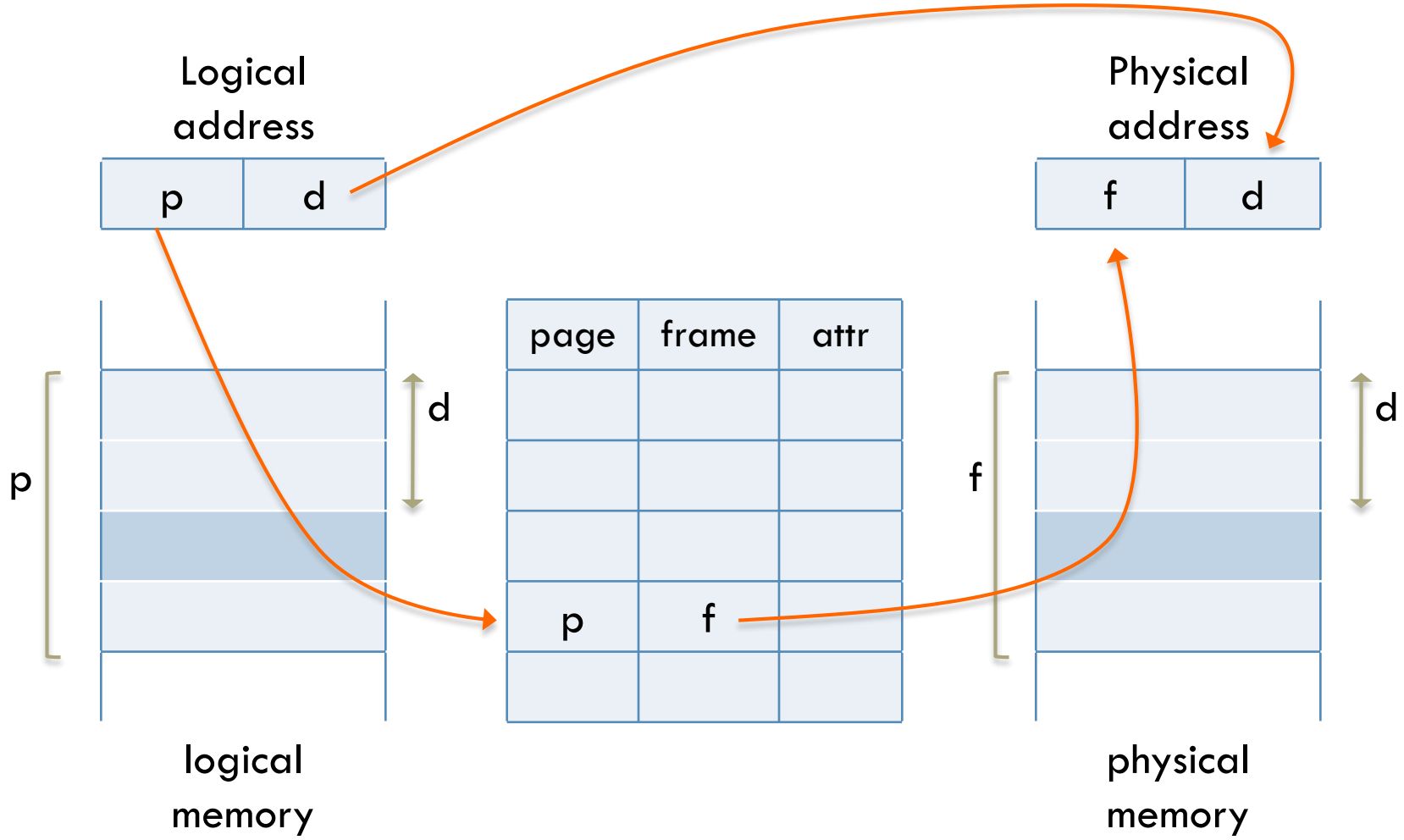
3.6. Kỹ thuật phân trang (Paging)

- Khi chương trình yêu cầu một địa chỉ mới, một địa chỉ logic <p, d> được tạo ra bởi processor, sẽ có frame f tương ứng với số trang p (trong bảng phân trang) được xác định;
- Địa chỉ vật lý (physical address) được tính như sau:

$$PA = (f*S+d),$$

đồng thời chương trình được phép truy cập vào địa chỉ vật lý đó.

3.6. Kỹ thuật phân trang (Paging)

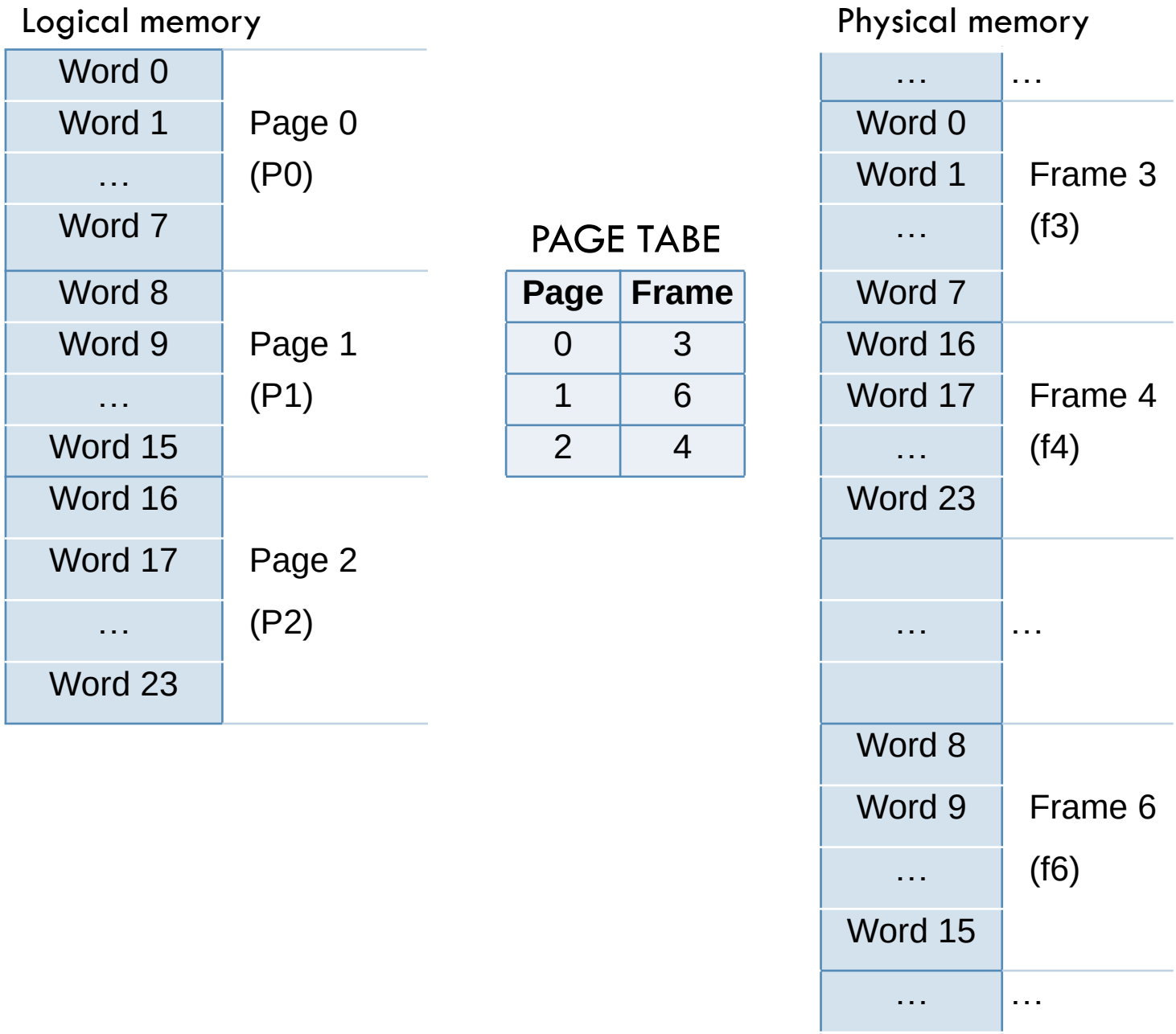


3.6. Kỹ thuật phân trang (Paging)

- **VD:** Xét ví dụ về ánh xạ từ LA sang PA như sau:

  S= 8 words → d: 3 bits

  - Kích thước bộ nhớ vật lý = 128 words. Như vậy sẽ có 128/8 = 16 frames → f: 4 bits
  - Giả sử kích thước chương trình tối đa là 4 trang → p: 2 bits
  - Một chương trình gồm 3 trang P0 → f3; P1 → f6; P2 → f4


Logical memory
Word 0
Word 1
…
Word 7
Page 0
(P0)
Word 8
Word 9
…
Word 15
Page 1
(P1)
Word 16
Word 17
…
Word 23
Page 2
(P2)
PAGE TABE
Page
Frame
0
3
1
6
2
4
Physical memory
…
…
Word 0
Word 1
…
Word 7
Frame 3
(f3)
Word 16
Word 17
…
Word 23
Frame 4
(f4)
…
…
Word 8
Word 9
…
Word 15
Frame 6
(f6)
…
…

3.6. Kỹ thuật phân trang (Paging)

Program Line
Word 0
Word 1
…
Word 7
Word 8
Word 9
…
Word 15
Word 16
Word 17
…
Word 23

3.6. Kỹ thuật phân trang (Paging)

Program Line	Logical Address
Word 0	00 000
Word 1	00 001
…	…
Word 7	00 111
Word 8	01 000
Word 9	01 001
…	…
Word 15	01 111
Word 16	10 000
Word 17	10 001
…	…
Word 23	10 111

3.6. Kỹ thuật phân trang (Paging)

Program Line	Logical Address	Offset
Word 0	00 000	000
Word 1	00 001	001
…	…	…
Word 7	00 111	111
Word 8	01 000	000
Word 9	01 001	001
…	…	…
Word 15	01 111	111
Word 16	10 000	000
Word 17	10 001	001
…	…	…
Word 23	10 111	111

3.6. Kỹ thuật phân trang (Paging)

Program Line	Logical Address	Offset	Page Number
Word 0	00 000	000	00
Word 1	00 001	001	00
…	…	…	…
Word 7	00 111	111	00
Word 8	01 000	000	01
Word 9	01 001	001	01
…	…	…	…
Word 15	01 111	111	01
Word 16	10 000	000	10
Word 17	10 001	001	10
…	…	…	…
Word 23	10 111	111	10

3.6. Kỹ thuật phân trang (Paging)

Program Line	Logical Address	Offset	Page Number	Frame Number
Word 0	00 000	000	00	0011
Word 1	00 001	001	00	0011
…	…	…	…	…
Word 7	00 111	111	00	0011
Word 8	01 000	000	01	0110
Word 9	01 001	001	01	0110
…	…	…	…	…
Word 15	01 111	111	01	0110
Word 16	10 000	000	10	0100
Word 17	10 001	001	10	0100
…	…	…	…	…
Word 23	10 111	111	10	0100

3.6. Kỹ thuật phân trang (Paging)

Program Line	Logical Address	Offset	Page Number	Frame Number	Physical Address
Word 0	00 000	000	00	0011	0011 000
Word 1	00 001	001	00	0011	0011 001
…	…	…	…	…	…
Word 7	00 111	111	00	0011	0011 111
Word 8	01 000	000	01	0110	0110 000
Word 9	01 001	001	01	0110	0110 001
…	…	…	…	…	…
Word 15	01 111	111	01	0110	0110 111
Word 16	10 000	000	10	0100	0100 000
Word 17	10 001	001	10	0100	0100 001
…	…	…	…	…	…
Word 23	10 111	111	10	0100	0100 111

3.6. Kỹ thuật phân trang (Paging)

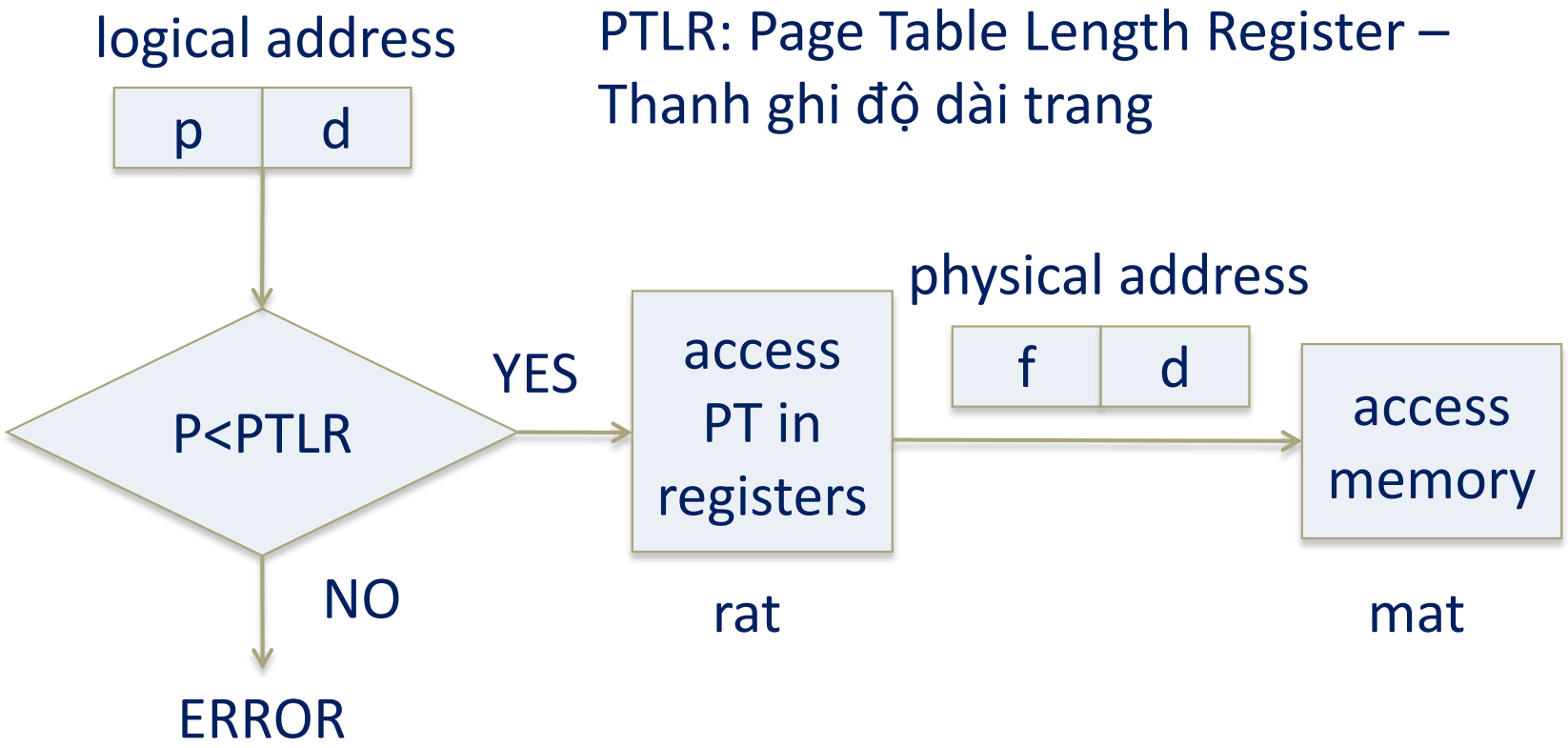
- Mọi truy cập vào bộ nhớ đều thực hiện thông qua bảng phân trang, vì vậy, cần có một cơ chế thực hiện hiệu quả.
- Phương pháp:
  1. Sử dụng thanh ghi truy cập nhanh (**fast dedicated registers**);
  2. Lưu bảng phân trang trong bộ nhớ chính;
  3. Sử dụng thanh nhớ kết hợp nội dung-địa chỉ **CAAR** (**content-addressable associative registers**).

Chương 3. Quản lý bộ nhớ

3.6.1. Sử dụng thanh ghi truy cập nhanh

- Bảng phân trang được lưu trong các thanh ghi, chỉ HĐH mới có quyền tác động đến những thanh ghi này.
- Nhận xét: nếu bảng phân trang lớn, phương pháp này trở nên rất đắt đỏ vì đòi hỏi nhiều thanh ghi.

3.6.1. Sử dụng thanh ghi truy cập nhanh



rat = Register Access Time: thời gian truy cập thanh ghi.

mat = Memory Access Time: thời gian truy cập bộ nhớ

Effective Memory Access Time:

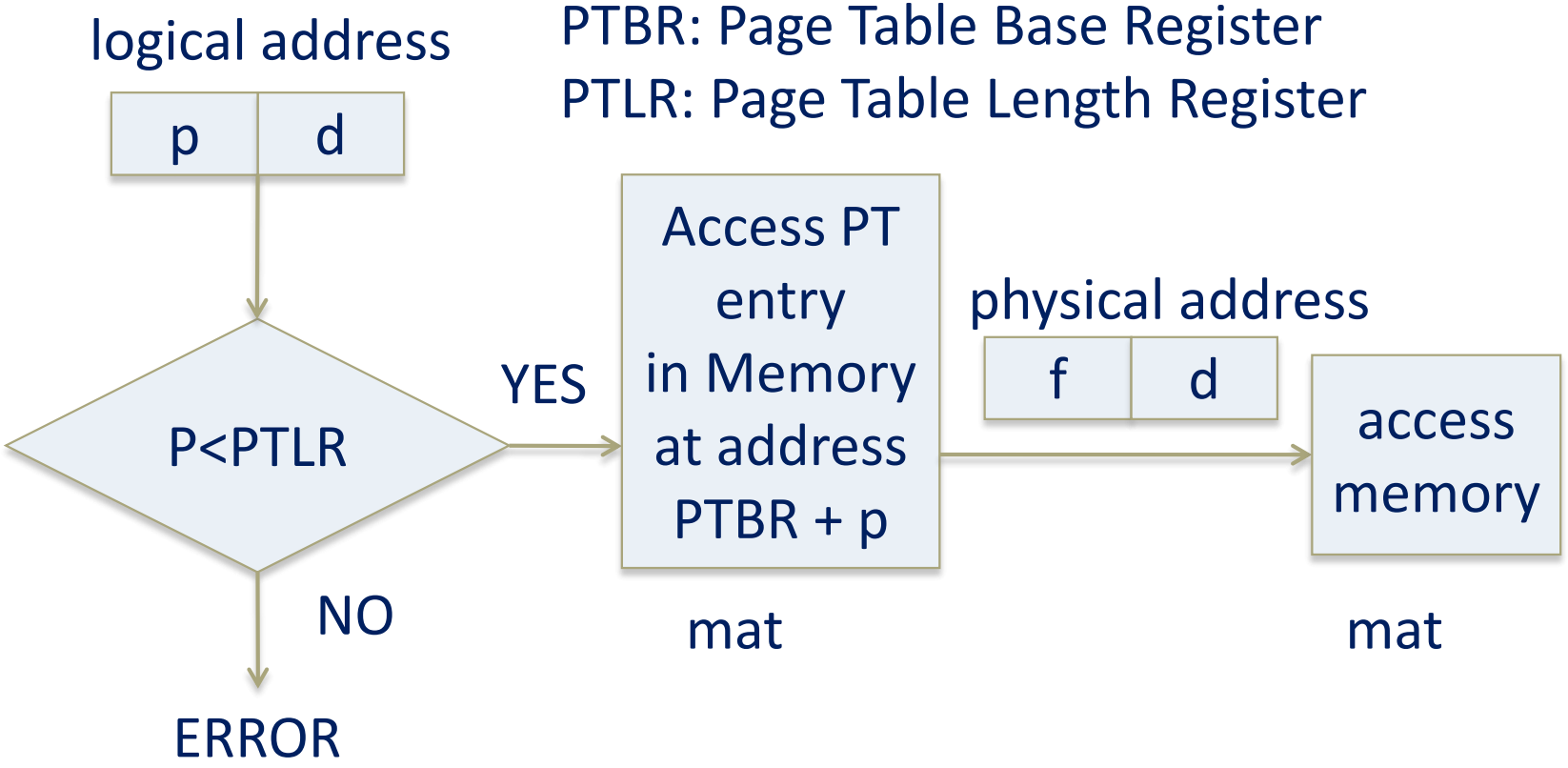
emat = rat + mat

Chương 3. Quản lý bộ nhớ

3.6.2. Lưu bảng phân trang trong MM

- Trong phương pháp này, HĐH lưu bảng phân trang trong bộ nhớ chính, thay vì trong thanh ghi.
- Với mỗi tham chiếu tới địa chỉ logic, đòi hỏi 2 truy cập tới bộ nhớ:
    1. Truy cập tới bảng phân trang để tìm ra frame tương ứng;
    2. Truy cập tới word trong frame.
- Nhận xét: phương pháp đơn giản dễ cài đặt nhưng tốn thời gian.

3.6.2. Lưu bảng phân trang trong MM



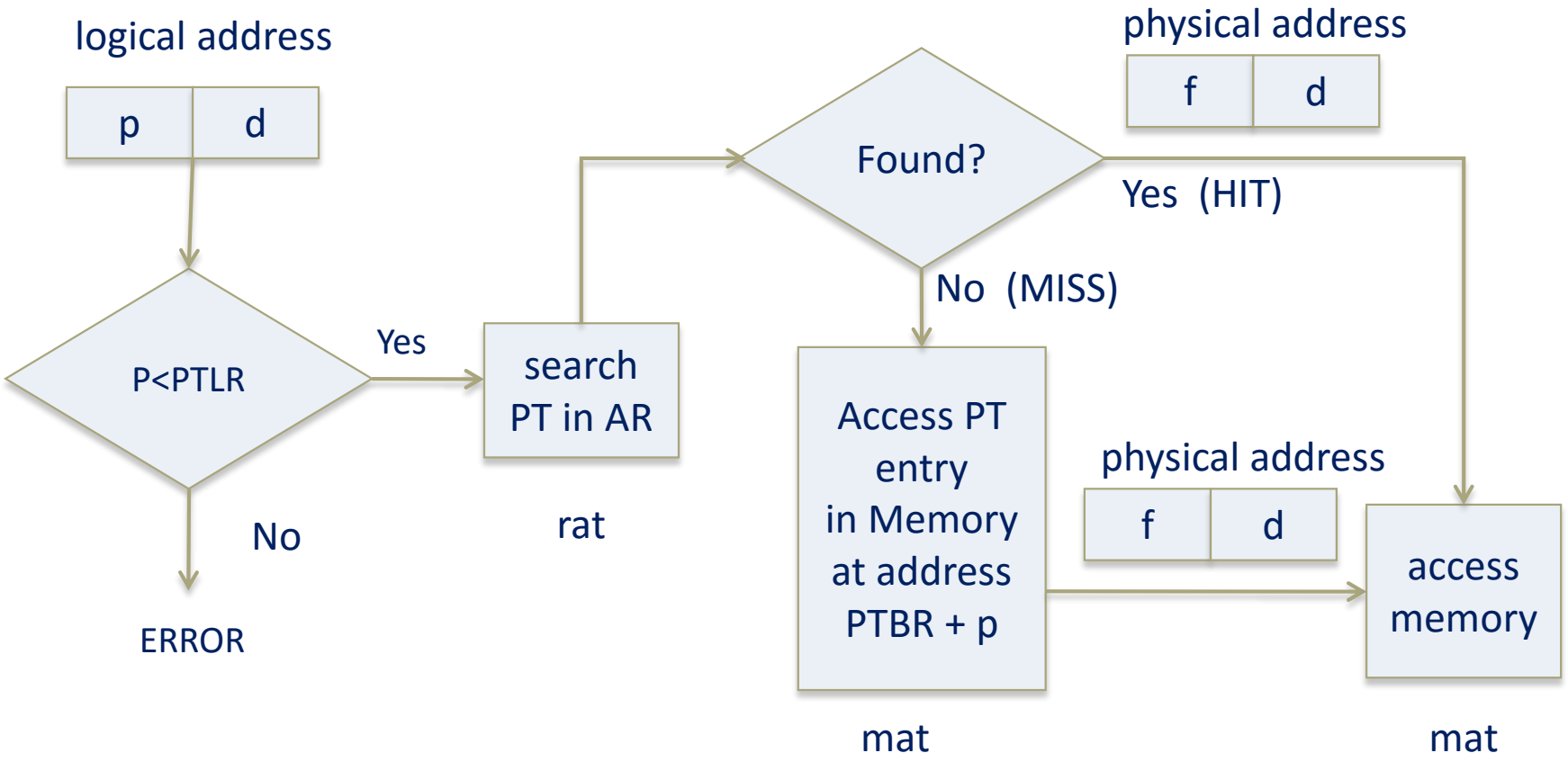
Effective Memory Access Time:
emat = mat+mat = 2mat

Chương 3. Quản lý bộ nhớ

3.6.3. Sử dụng thanh nhớ kết hợp CAAR

- Kết hợp của 2 phương phương pháp trên.
  - Sử dụng các thanh nhớ kết hợp có dung lượng nhỏ, tốc độ truy cập nhanh được thiết kết đặc biệt trong phần cứng cho phép tìm kiếm nhanh trên nội dung của nó, chẳng hạn cho phép tìm kiếm đồng thời trên tất cả các thanh ghi trong cùng một thời điểm.
- Nhận xét: Nhanh chóng, tuy nhiên thanh nhớ kết hợp rất đắt về giá thành, chỉ áp dụng với quy mô nhỏ.

3.6.3. Sử dụng thanh nhớ kết hợp CAAR



Effective Memory Access Time: h: hit ratio

emat = h *$emat_{HIT}$ + (1-h) * $emat_{MISS}$ = h(rat+mat)+(1-h)(rat+mat+mat)

3.6.3. Sử dụng thanh nhớ kết hợp CAAR

- Giả sử ta có hệ phân trang sử dụng thanh nhớ kết hợp. Các thanh có tốc độ truy cập là 30 ns, và tốc độ truy cập bộ nhớ là 470 ns. Hệ thống có hiệu suất là 90%.

rat=30 ns; mat=470ns ; h=0.9

- Nếu số trang được tìm thấy ở trong thanh nhớ kết hợp thì $emat_{HIT}$ được tính như sau:

$$emat_{HIT} = 30 + 470 = 500 \text{ ns.}$$

- vì hệ thống đòi hỏi 1 truy cập tới thanh nhớ kết hợp và 1 truy cập tới bộ nhớ.

3.6.3. Sử dụng thanh nhớ kết hợp CAAR

rat=30 ns, mat=470ns, h=0.9

- Mặt khác, nếu số trang không được tìm thấy trong thanh nhớ kết hợp thì $emat_{MISS}$ được tính như sau:

$$emat_{MISS} = 30 + (470+470) = 970 \text{ ns.}$$

- Vì cần đến 1 truy cập tới thanh nhớ kết hợp và 2 truy cập tới bộ nhớ chính.
- Khi đó, emat được tính như sau:

$$emat = h * emat_{HIT} + (1-h) * emat_{MISS}$$

$$= 0.9 * 500 + 0.1 * 970$$

$$= 450 + 97 = 547 \text{ ns}$$

Chương 3. Quản lý bộ nhớ

3.7. Chia sẻ trang (Sharing Pages)

- **Kỹ thuật chia sẻ trang:**
  - Cho phép chia sẻ các thủ tục hay các chương trình hệ thống, thủ tục hay chương trình người dùng, hoặc có thể chia sẻ cả vùng dữ liệu.
  - Trong các hệ chia sẻ thời gian, chia sẻ trang có lợi thế đặc biệt trong các hệ chia sẻ thời gian.
- **Mã chỉ đọc** (non-self-modifying code = read only) mã không thay đổi trong suốt quá trình thực hiện. Vì vậy:
  - nhiều tiến trình có thể thực hiện các đoạn code chỉ đọc trong cùng một thời gian;
  - Mỗi tiến trình có lưu trữ riêng và bản sao của các thanh ghi để chứa dữ liệu cho việc thực thi chương trình chia sẻ.

3.7. Chia sẻ trang (Sharing Pages)

- **VD:** Giả sử hệ thống có kích cỡ trang là 30 MB.
  - Có 3 người dùng đang thực thi một chương trình soạn thảo có kích cỡ là 90 Mb (3 pages) và mỗi người cần 30Mb cho soạn thảo;
  - Để hỗ trợ cả 3 người dùng, HĐH cần 3 * (90+30) = 360 Mb.
  - Tuy nhiên, nếu chương trình soạn thảo là bất biến (non-self-modifying code = read only), thì nó có thể được chia sẻ cho mọi người dùng cùng lúc, chỉ cần 1 copy của chương trình soạn thảo. Vì vậy chỉ cần:

$$90 + 30 * 3 = 180 \text{ Mb}$$

3.7. Chia sẻ trang (Sharing Pages)

User-1

P0	e1
P1	e2
P2	e3
P3	data1

PT-1

Page#	Frame#
0	8
1	4
2	5
3	7

Physical Memory

f0	OS
f1	OS
f2	OS
f3	
f4	e2
f5	e3
f6	
f7	data1
f8	e1
f9	
f10	
f11	
f12	
f13	
f14	
f15	

3.7. Chia sẻ trang (Sharing Pages)

User-1

P0	e1
P1	e2
P2	e3
P3	data1

PT-1

Page#	Frame#
0	8
1	4
2	5
3	7

User-2

P0	e1
P1	e2
P2	e3
P3	data2

PT-2

Page#	Frame#
0	8
1	4
2	5
3	12

Physical Memory

f0	OS
f1	OS
f2	OS
f3	
f4	e2
f5	e3
f6	
f7	data1
f8	e1
f9	
f10	
f11	
f12	data 2
f13	
f14	
f15	

3.7. Chia sẻ trang (Sharing Pages)

User-1

P0	e1
P1	e2
P2	e3
P3	data1

PT-1

Page#	Frame#
0	8
1	4
2	5
3	7

User-2

P0	e1
P1	e2
P2	e3
P3	data2

PT-2

Page#	Frame#
0	8
1	4
2	5
3	12

User-3

P0	e1
P1	e2
P2	e3
P3	data3

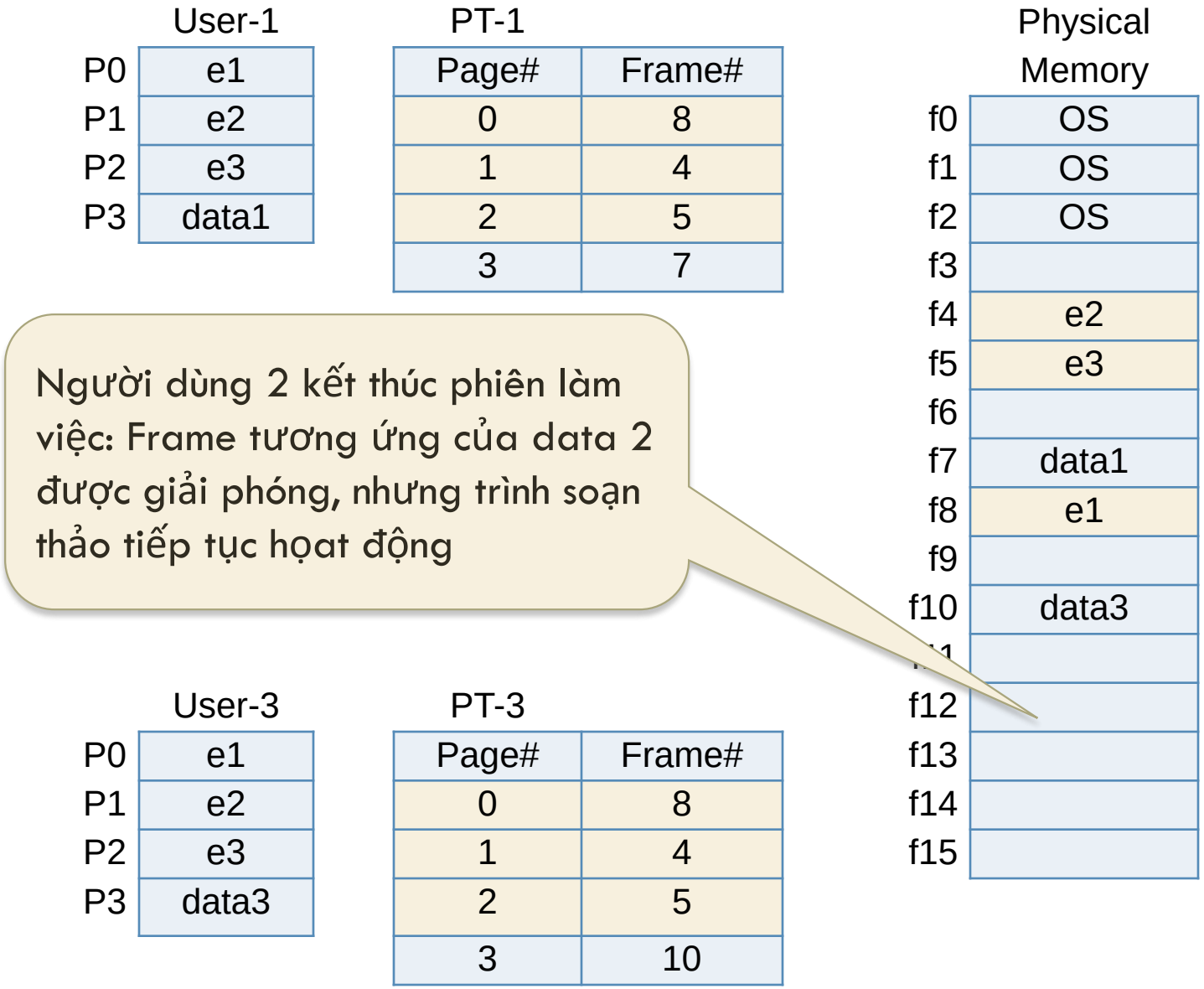
PT-3

Page#	Frame#
0	8
1	4
2	5
3	10

Physical Memory

f0	OS
f1	OS
f2	OS
f3	
f4	e2
f5	e3
f6	
f7	data1
f8	e1
f9	
f10	data3
f11	
f12	data 2
f13	
f14	
f15	

3.7. Chia sẻ trang (Sharing Pages)



3.7. Chia sẻ trang (Sharing Pages)

Người dùng 1 kết thúc: data1 cũng được giải phóng

Physical Memory

f0	OS
f1	OS
f2	OS
f3	
f4	e2
f5	e3
f6	
f7	
f8	e1
f9	
f10	data3
f11	
f12	
f13	
f14	
f15	

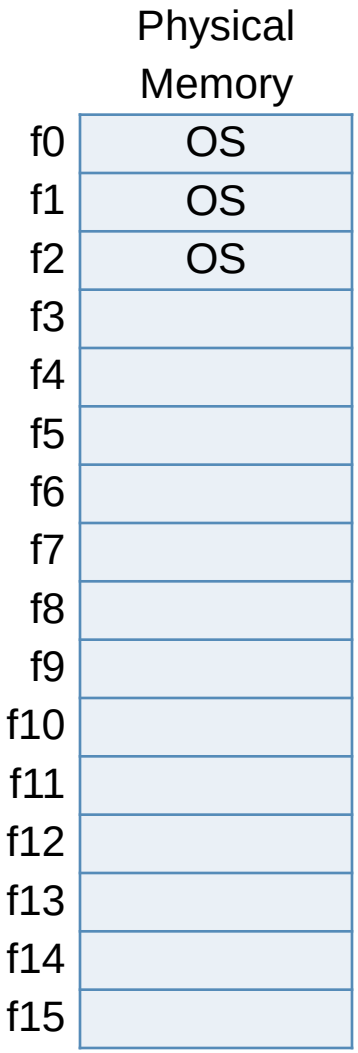
User-3

P0	e1
P1	e2
P2	e3
P3	data3

PT-3

Page#	Frame#
0	8
1	4
2	5
3	10

3.7. Chia sẻ trang (Sharing Pages)



Chỉ khi người dùng 3 kết thúc làm việc: Data-3 và chương trình đồng thời được giải phóng khỏi bộ nhớ.

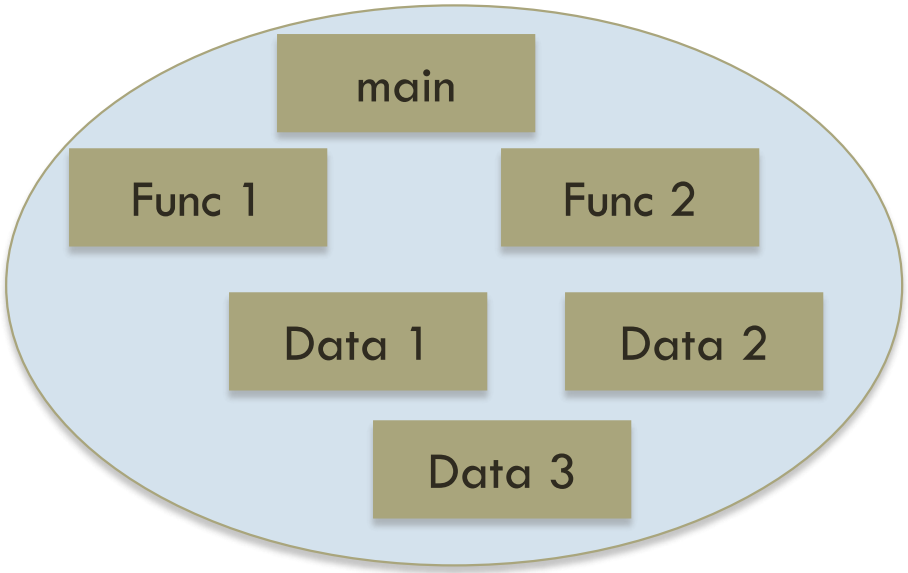
Chương 3. Quản lý bộ nhớ

3.8. Phân đoạn (Segmentation)

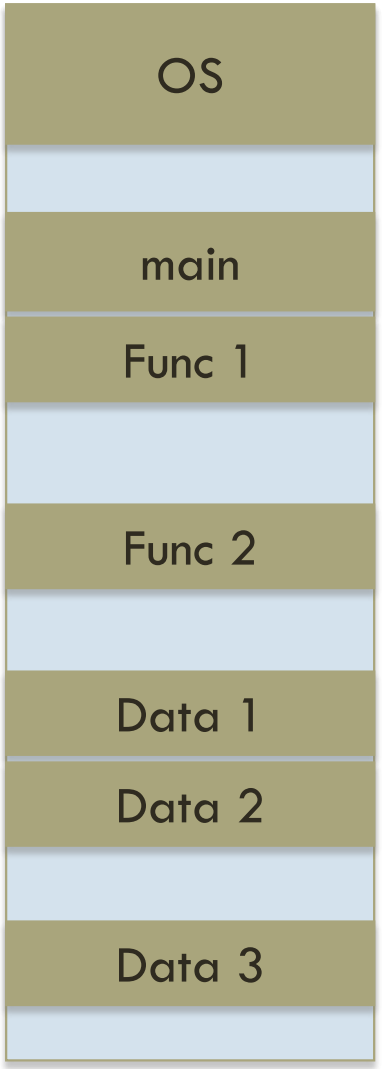
- Trong kỹ thuật phân đoạn, chương trình được chia ra thành các đoạn có kích cỡ động, thay vì kích cỡ cố định như phân trang.
- Giống như kỹ thuật phân vùng động, nhưng ở đây chương trình được chia ra thành các phần nhỏ.
- Mỗi một LA được định dạng bởi một tên đoạn và độ dời chuyển trong đoạn đó, các đoạn được đánh số. Trên thực tế, chương trình được phân đoạn tự động bởi trình biên dịch hoặc trình dịch hợp ngữ.

3.8. Phân đoạn (Segmentation)

- VD, Chẳng hạn trình dịch C có thể chia chương trình thành các đoạn như:
  - Code của mỗi hàm số;
  - Các biến cục bộ của từng hàm;
  - Biến toàn cục của chương trình;



Logical memory

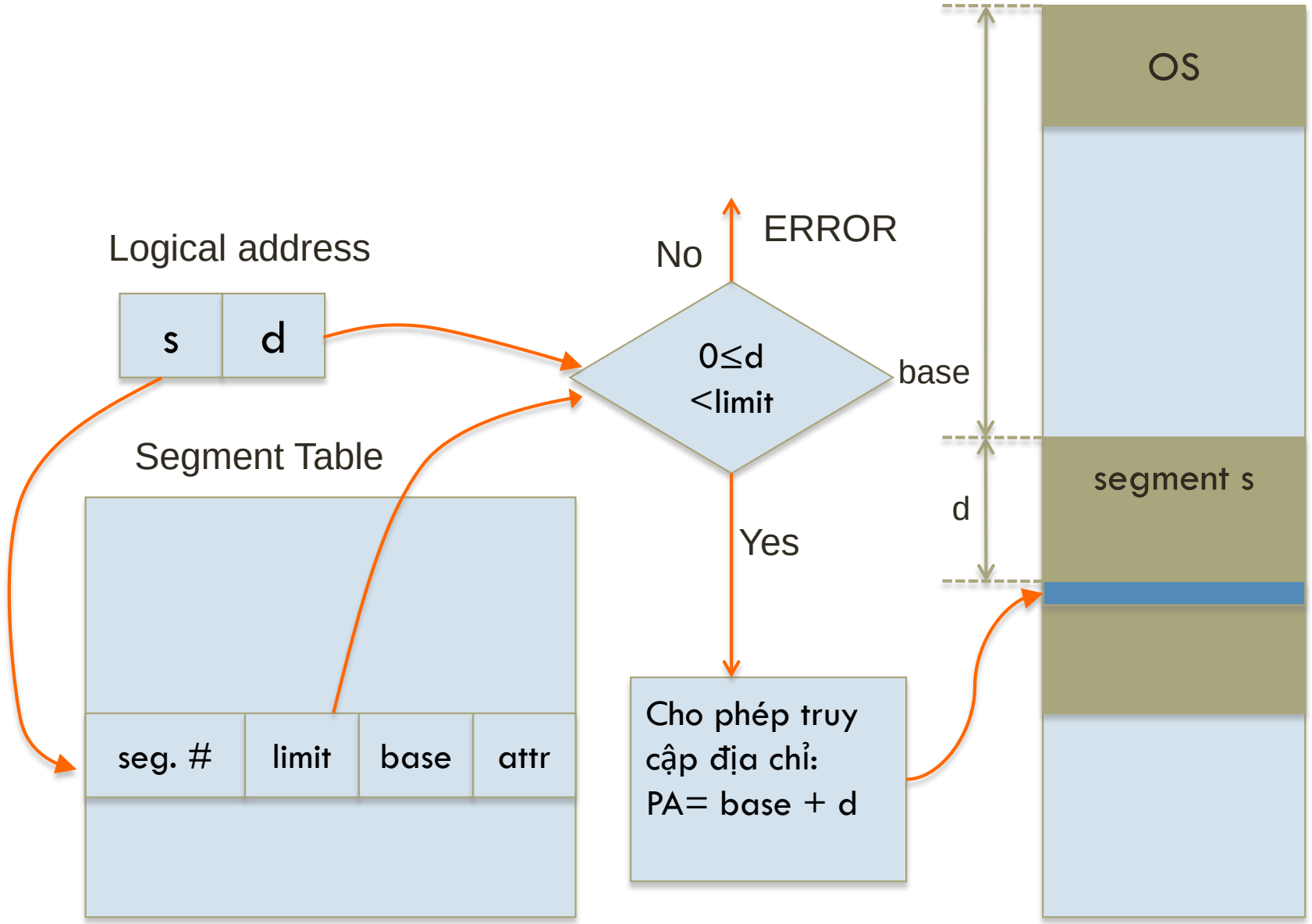


Physical memory

3.8. Phân đoạn (Segmentation)

- Để ánh xạ từ LA sang PA, hệ thống sử dụng bảng phân đoạn (*segment table*). Khi một địa chỉ logic <s, d> được tạo ra bởi processor:
  1. Địa chỉ cơ sở và giới hạn tương ứng với các đoạn được xác định qua bảng phân đoạn.
  2. OS kiểm tra khi nào thì d hợp lệ ($0 \leq d < limit$).
  3. Khi đó địa chỉ vật lý được tính bằng (base + d) và cho phép truy cập đến vùng nhớ tương ứng.

3.8. Phân đoạn (Segmentation)



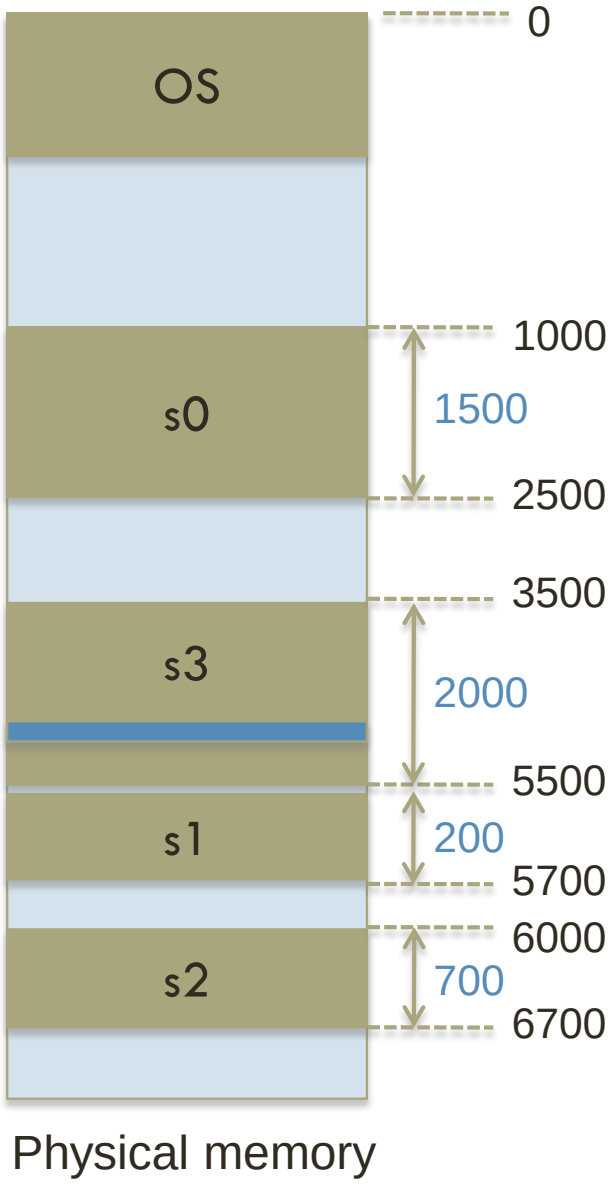
3.8. Phân đoạn (Segmentation)

- **VD:** Tạo ra ánh xạ bộ nhớ theo bảng phân đoạn cho ở sau, giả sử rằng địa chỉ logic được tạo ra là <3,1123>, hãy tìm địa chỉ vật lý tương ứng.

Segment	Limit	Base
0	1500	1000
1	200	5500
2	700	6000
3	2000	3500

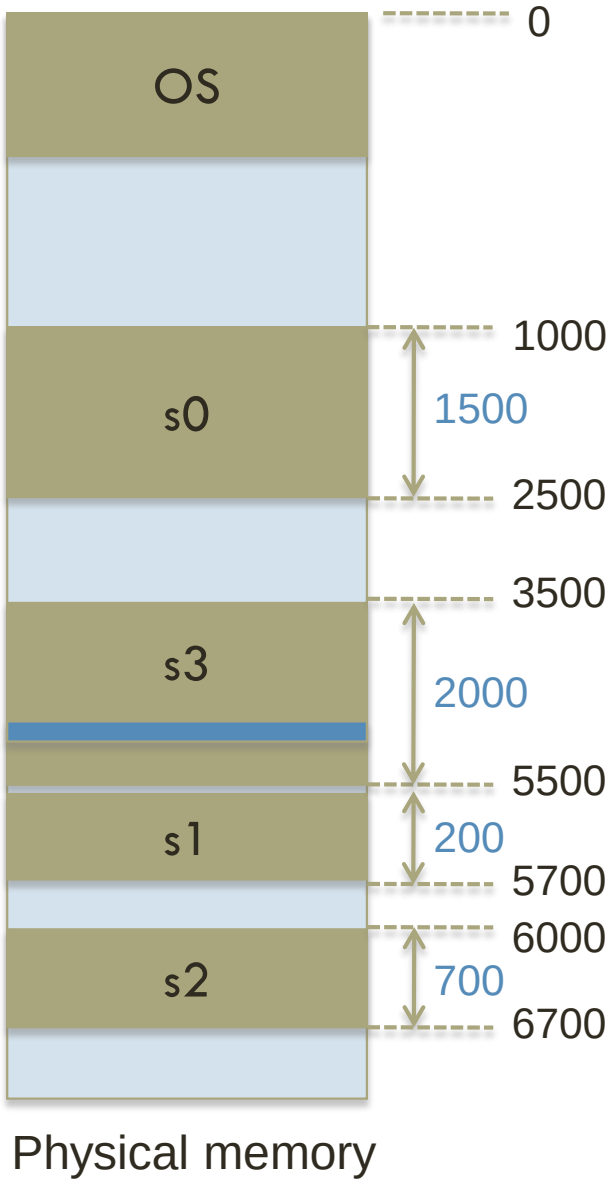
3.8. Phân đoạn (Segmentation)

Segment	Limit	Base
0	1500	1000
1	200	5500
2	700	6000
3	2000	3500



3.8. Phân đoạn (Segmentation)

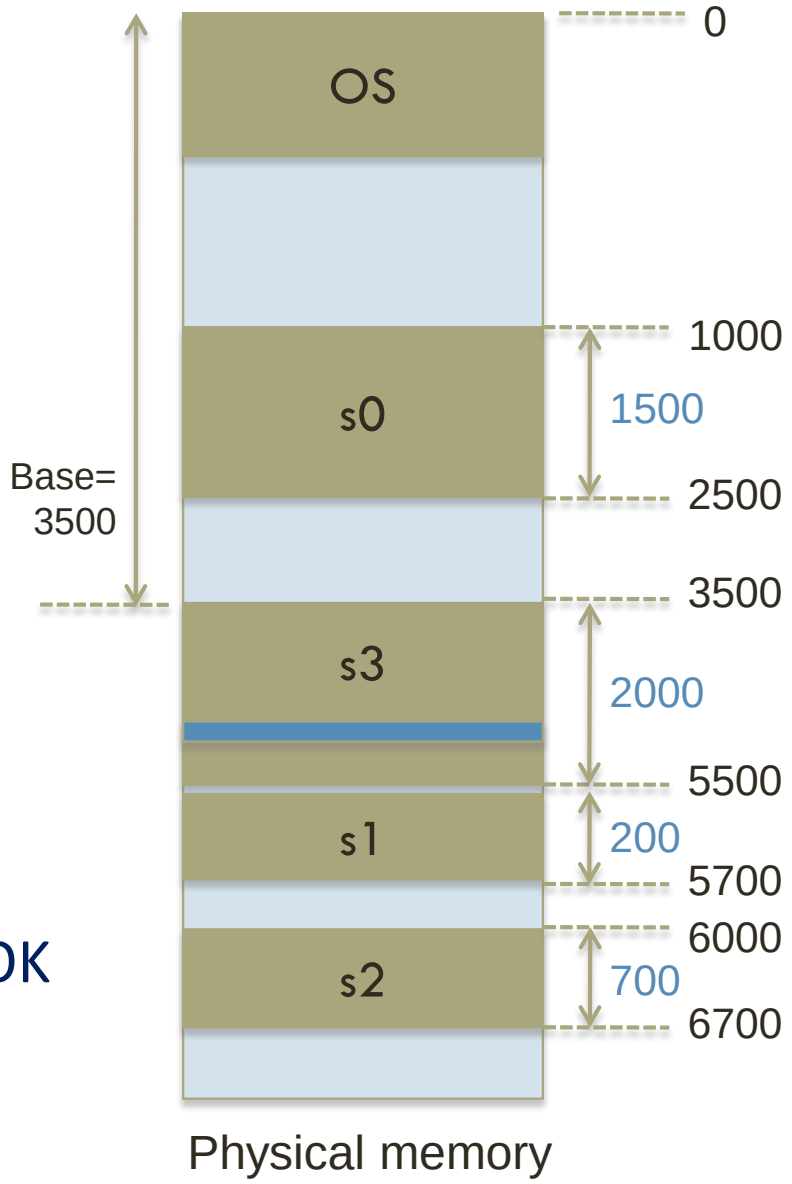
Segment	Limit	Base
0	1500	1000
1	200	5500
2	700	6000
3	2000	3500



Logical address: <3,1123>
s=3, d=1123
Check if d<limit? 1123<2000, OK
Physical address=
base+d=3500+1123=4623

3.8. Phân đoạn (Segmentation)

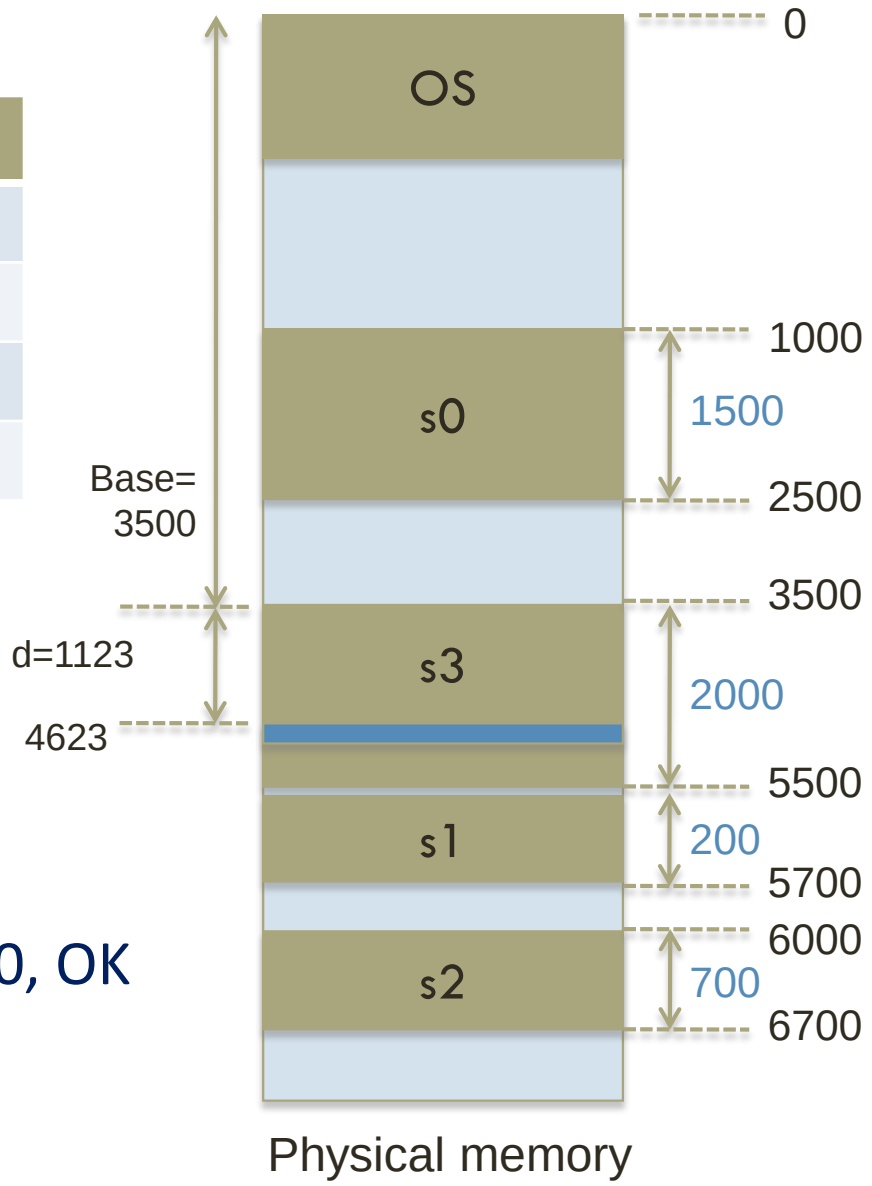
Segment	Limit	Base
0	1500	1000
1	200	5500
2	700	6000
3	2000	3500



Logical address: <3,1123>
s=3, d=1123
Check if d<limit? 1123<2000, OK
Physical address=
base+d=3500+1123=4623

3.8. Phân đoạn (Segmentation)

Segment	Limit	Base
0	1500	1000
1	200	5500
2	700	6000
3	2000	3500



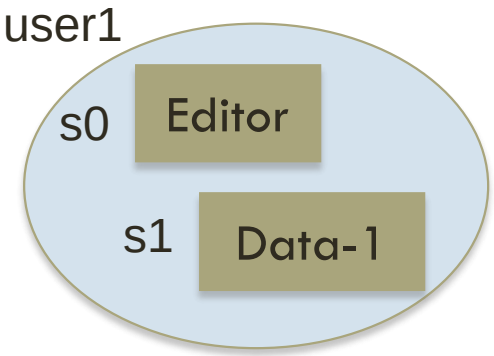
Logical address: <3,1123>
s=3, d=1123
Check if d<limit? 1123<2000, OK
Physical address=
base+d=3500+1123=4623

Chương 3. Quản lý bộ nhớ

3.9. Chia sẻ đoạn (Sharing Segments)

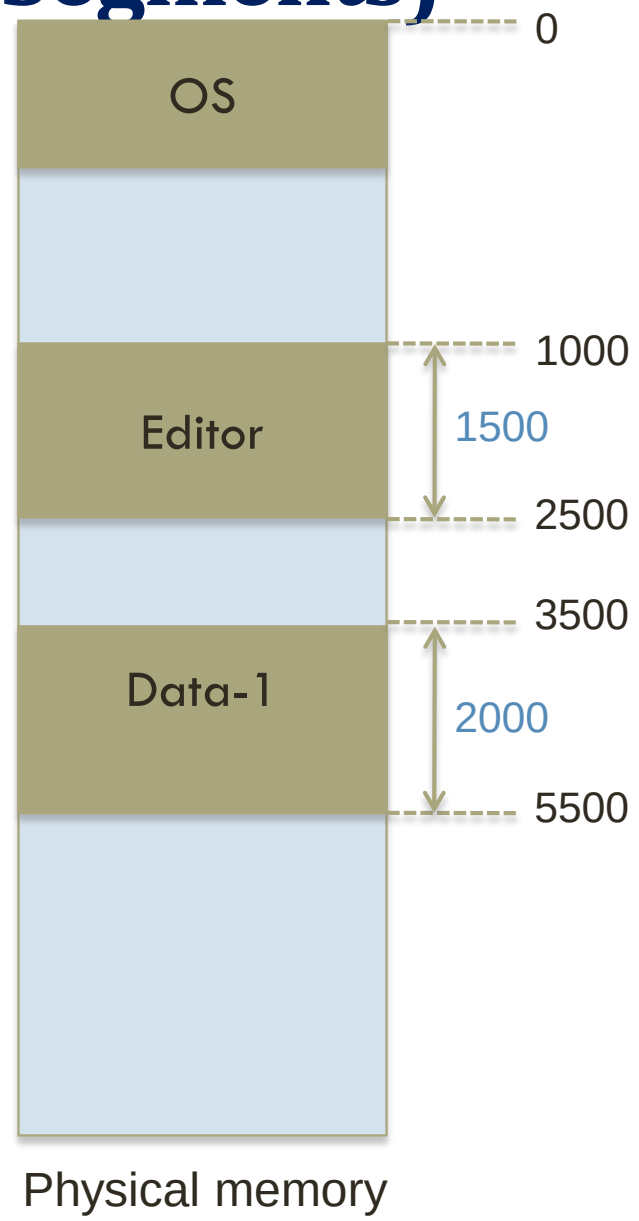
- Bảng phân đoạn cũng có thể được triển khai trong bộ nhớ chính hoặc trong thanh nhớ kết hợp, tương tự như trường hợp phân đoạn.
- Như vậy, có những đoạn cũng có thể được chia sẻ giống như chia sẻ trang. Các đoạn chia sẻ cũng phải là đoạn mã chỉ đọc và được gán cùng số đoạn (segment number).

3.9. Chia sẻ đoạn (Sharing Segments)

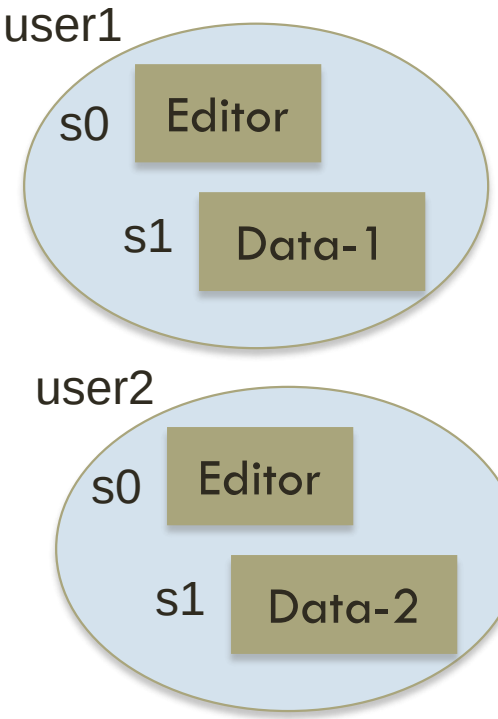


ST1

seg	lim	base
0	1500	1000
1	2000	3500



3.9. Chia sẻ đoạn (Sharing Segments)

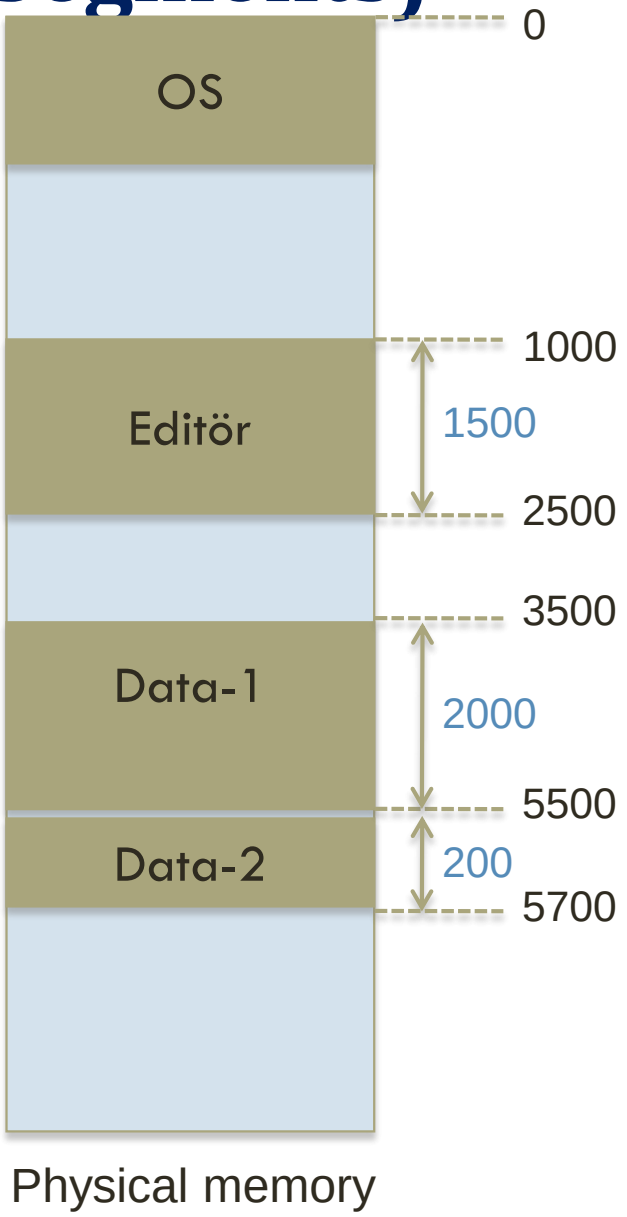


ST1

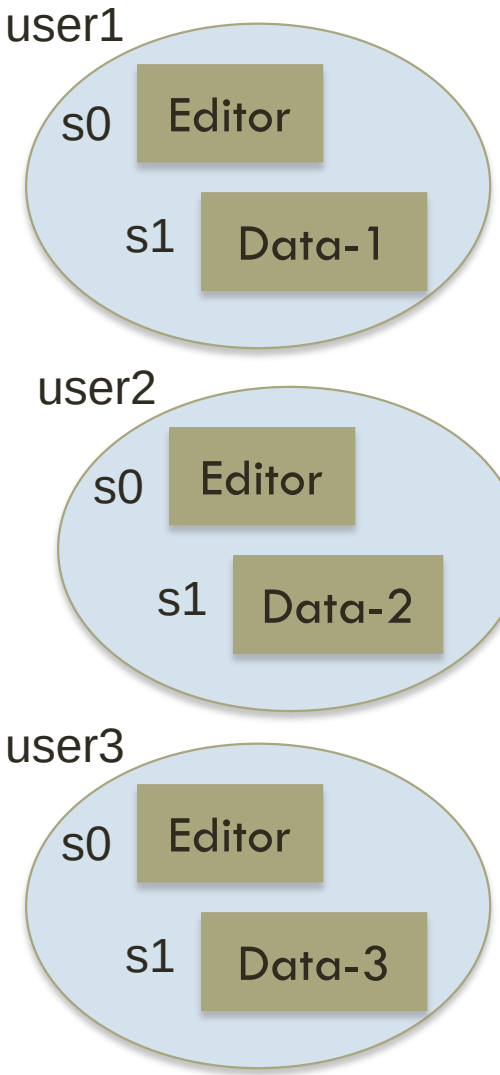
seg	lim	base
0	1500	1000
1	2000	3500

ST2

seg	lim	base
0	1500	1000
1	200	5500



3.9. Chia sẻ đoạn (Sharing Segments)



ST1

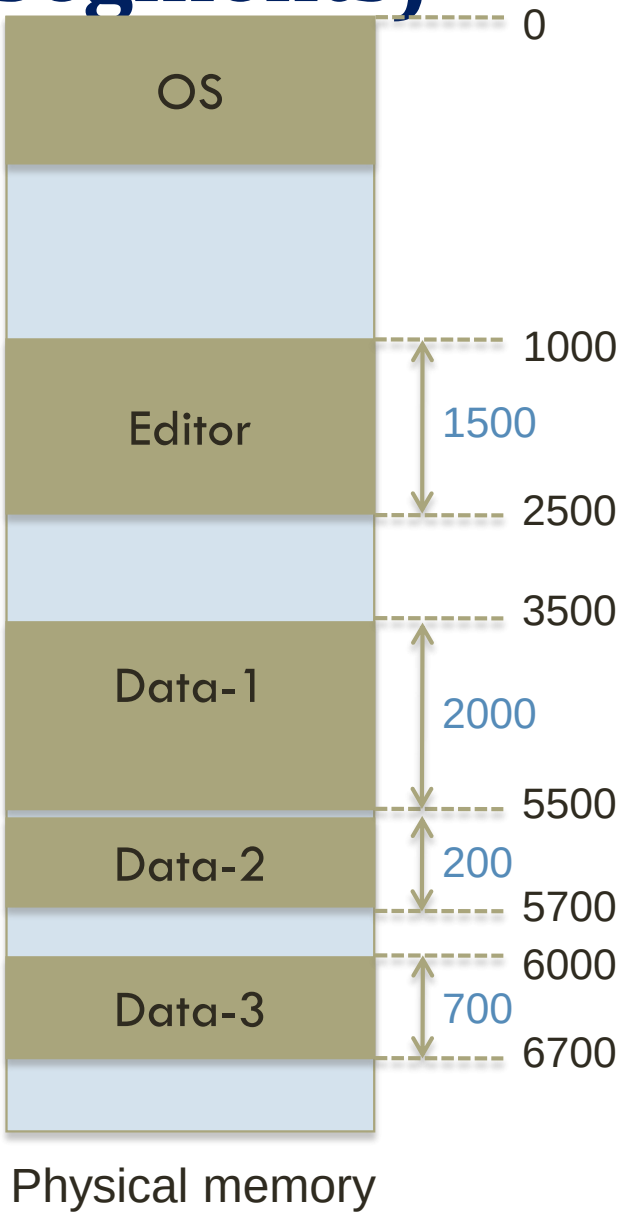
seg	lim	base
0	1500	1000
1	2000	3500

ST2

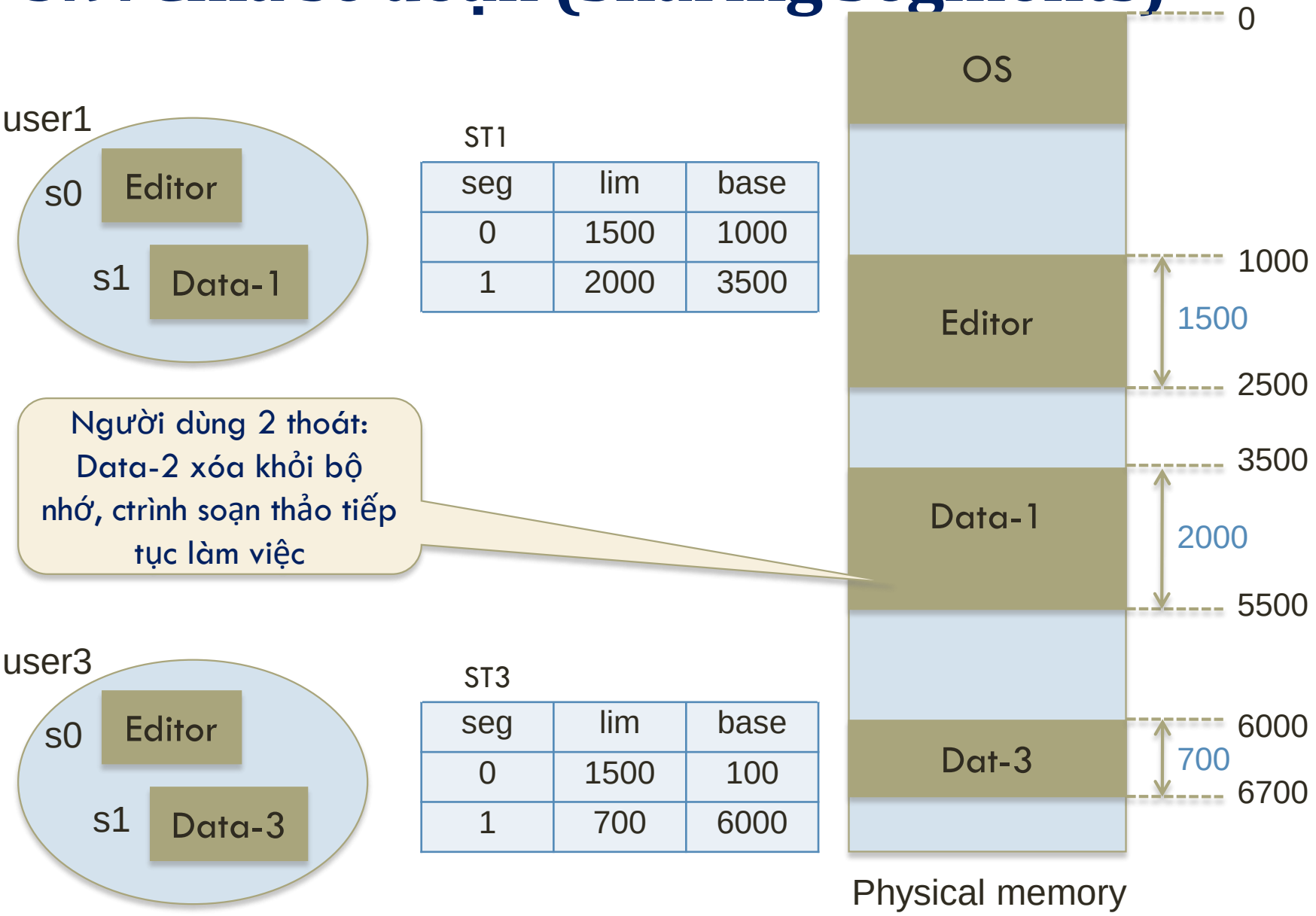
seg	lim	base
0	1500	1000
1	200	5500

ST3

seg	lim	base
0	1500	1000
1	700	6000

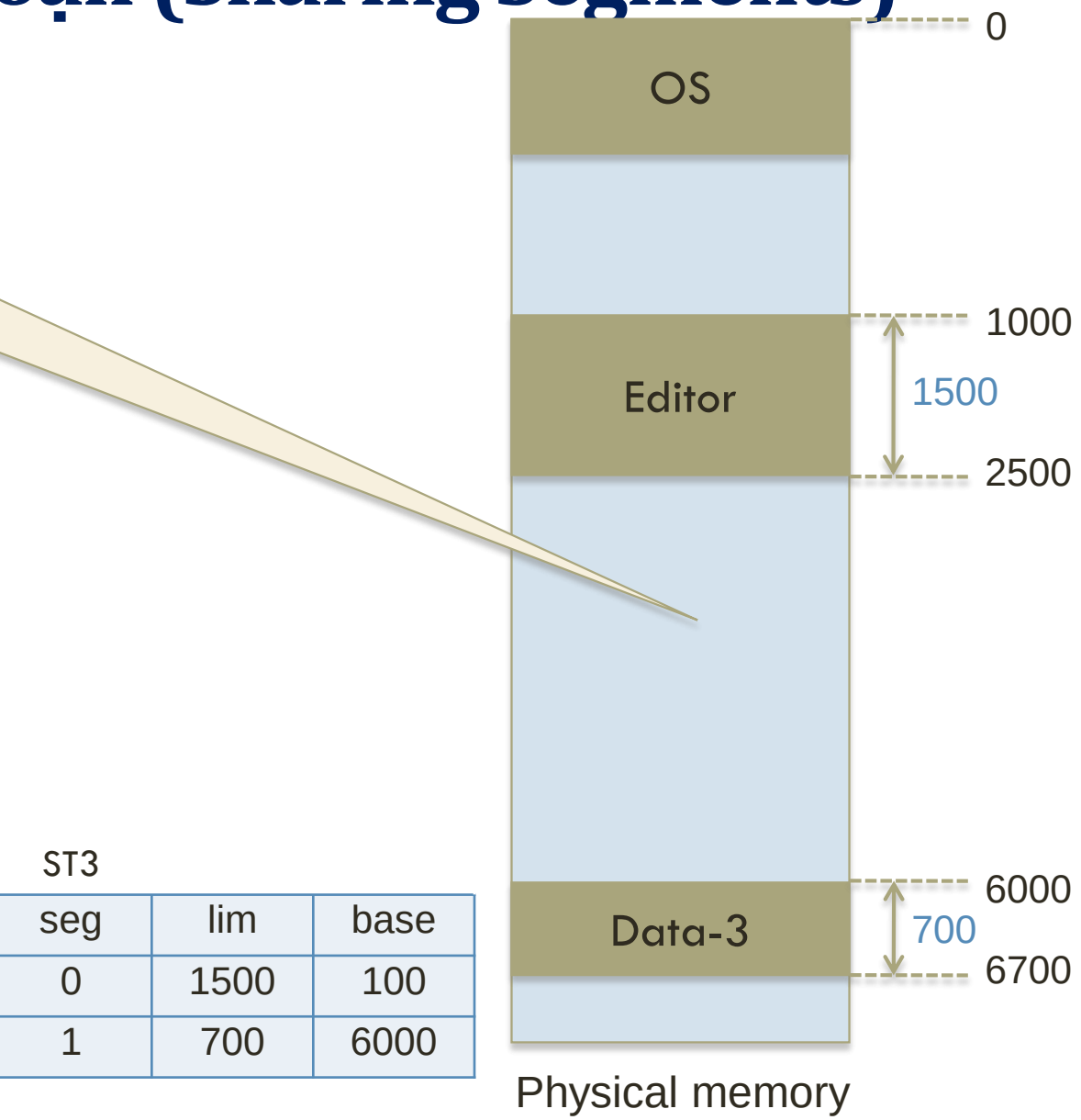


3.9. Chia sẻ đoạn (Sharing Segments)

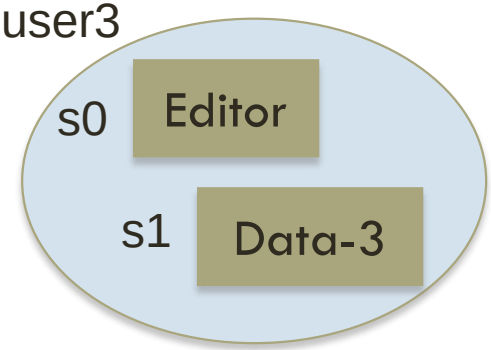


3.9. Chia sẻ đoạn (Sharing Segments)

User 1 thoát:
Data-1 được giải phóng khỏi bộ nhớ.



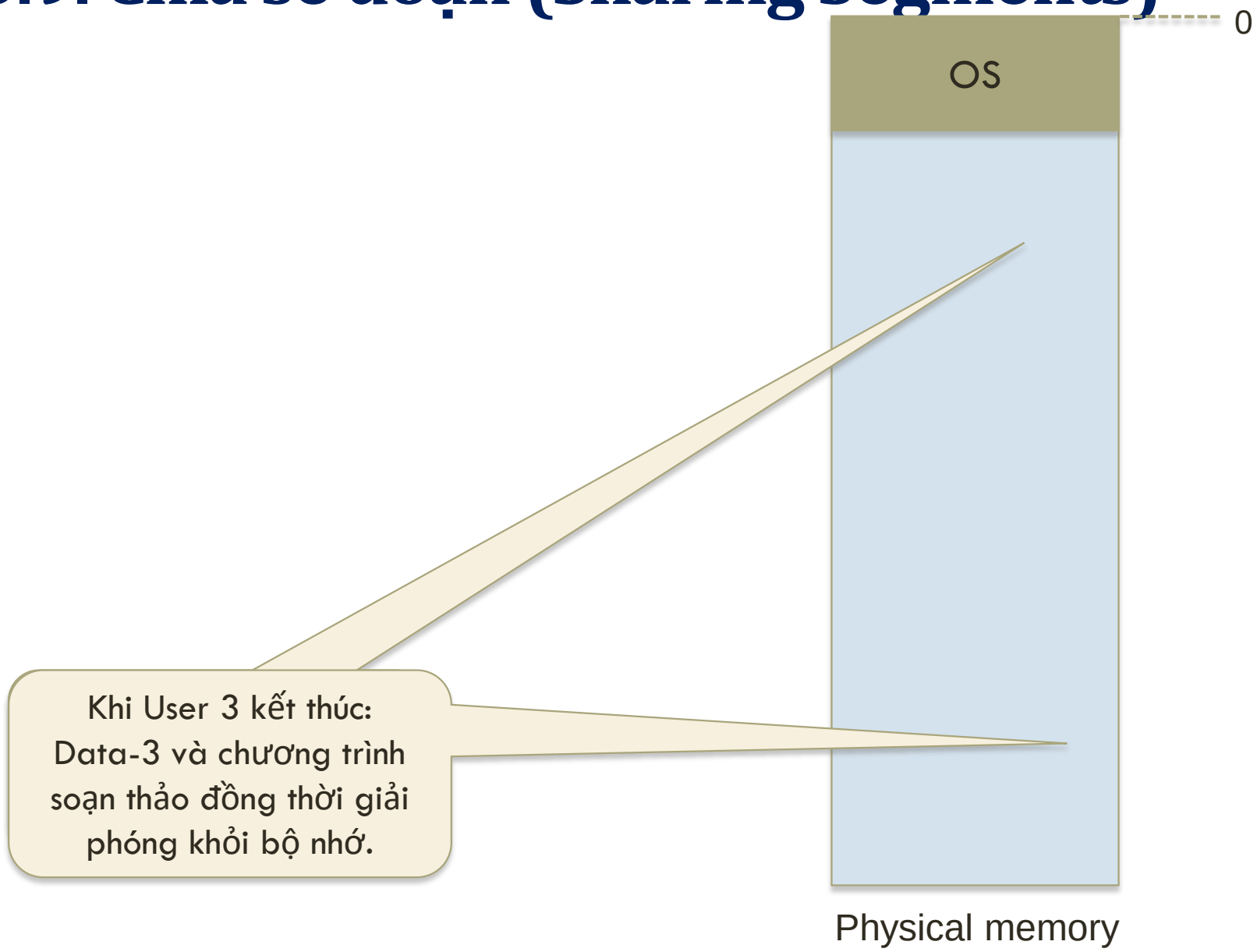
user3



ST3

seg	lim	base
0	1500	100
1	700	6000

3.9. Chia sẻ đoạn (Sharing Segments)



Chương 3. Quản lý bộ nhớ

3.10. Kết hợp phân đoạn với phân trang

- Kỹ thuật kết hợp:
  - Bộ nhớ chính được chia thành các đoạn sau đó mỗi đoạn lại được chia thành các trang.
  - Các địa chỉ tham chiếu bây giờ gồm ba thành phần: (segment, page, offset)

Bộ nhớ ảo

- Nhận xét: Các phần của một chương trình không nhất thiết phải nạp vào bộ nhớ chính tại cùng một thời điểm
  - Đoạn mã điều khiển các lỗi hiếm khi xảy ra
  - Các arrays, list, tables được cấp phát bộ nhớ (cấp phát tĩnh) nhiều hơn yêu cầu cần thiết
  - Một số tính năng ít khi được dùng của một chương trình
- - Có thể hình dung không gian bộ nhớ ảo bao gồm bộ nhớ chính và bộ nhớ thứ cấp. Để đạt hiệu quả làm việc cao cần phải đọc/ghi đĩa trực tiếp với các khối dữ liệu lớn.
- - Thông thường phần bộ nhớ thứ cấp tham gia vào bộ nhớ ảo được lưu trữ ở một vùng đặc biệt gọi là không gian hoán đổi (swap space). Ví dụ file system swap trong Unix/Linux, file pagefile.sys trong W2K/XP

Bộ nhớ ảo

- - Bộ nhớ ảo hoạt động trên nguyên lý cục bộ (locality principle) sau:
  - Tính cục bộ về thời gian (temporal locality): Các sự việc xảy ra ở thời điểm t rất có thể là đã hoặc sẽ xảy ra ở các thời điểm lân cận (t – dt, t + dt)
    - Ví dụ : một vùng nhớ đang được tham khảo có thể sẽ được tham khảo đến trong tương lai gần
  - Tính cục bộ về không gian(spatial locality): Biến cố xảy ra ở một vùng nhớ rất có thể là đã hoặc sẽ xảy ra ở các vùng lân cận
    - Ví dụ : những vùng nhớ đang được tham khảo gần đây thường kề nhau
- Ý nghĩa: Nguyên lý cục bộ là cơ sở trong các giải thuật thay thế trang
- Ưu điểm của cơ chế bộ nhớ ảo?
  - Dễ phát triển ứng dụng
  - Lưu trữ được nhiều tiến trình trong bộ nhớ

Sự thực thi với bộ nhớ ảo của process

- Hệ điều hành chỉ nạp một phần nhỏ của chương trình vào bộ nhớ
- Khi có một lệnh tham chiếu đến phần chương trình chưa có trong bộ nhớ chính:
  - - HĐH sẽ kích hoạt một ngắt mềm gọi là memory fault (page fault, segmentation fault)
  - - Sau đó Hệ điều hành chuyển tiến trình về trạng thái blocked
  - - Sau đó HĐH phát ra một yêu cầu đọc đĩa để nạp phần chương trình được tham chiếu vào bộ nhớ chính. Trong khi đó, một tiến trình khác sẽ chiếm được CPU để thực thi.
  - - Sau khi đọc ghi đĩa hoàn tất, một ngắt mềm được kích hoạt, báo cho hệ điều hành để chuyển tiến trình tương ứng trở lại trạng thái sẵn sàng trong ready queue.

Các thuật toán thay thế trang

- Để cơ chế bộ nhớ ảo làm việc hiệu quả, hệ thống cần thỏa mãn hai yêu cầu sau:
  - Phải có sự hỗ trợ của phần cứng cho cơ chế phân trang hay phân đoạn
  - Hệ điều hành cần có bộ phận quản lý việc hoán chuyển các trang/đoạn giữa bộ nhớ thứ cấp và bộ nhớ thực.
- Khi xảy ra page fault, hệ điều hành thực hiện các bước sau
  - Chuyển trạng thái của tiến trình sang Waite
  - Chọn một trang để thay thế (page replacement algorithm)
  - Khởi động việc nạp trang mới từ đĩa vào bộ nhớ
  - Chuyển thực thi cho tiến trình khác trong lúc đang thực hiện việc vào ra
  - Nhận interrupt báo I/O hoàn tất (i.e. đã nạp xong trang nhớ mới)
  - Chuyển trạng thái process về ready
  - Khi bộ nhớ chính có chỗ trống thì chúng ta tiến hành nạp chương trình vào trang nhớ đó→ page fault. Tuy nhiên, khi so sánh các giải thuật thì chúng ta có thể bỏ qua số page-fault khởi đầu vì đại lượng này như nhau đối với mọi giải thuật

Các thuật toán thay thế trang

- Yêu cầu : Tối thiểu số page fault
- Nguyên tắc tối ưu : Chọn trang thay thế là
  - 1. Trang không còn dùng nữa
  - 2. Trang sẽ không dùng lại trong thời gian xa nhất
- Các tiêu chuẩn (thực tế) để chọn trang thay thế
  - Các trang không bị thay đổi
  - Các trang không bị khóa
  - Các trang không thuộc quá trình nhiều page fault
  - Các trang không thuộc tập làm việc của quá trình
  - Các giải thuật thay thế trang phụ thuộc vào resident set (số frame cấp cho mỗi process)

Giải thuật tối ưu (MIN)

- - Giải thuật MIN hay optimal (OPT)
  - Thay thế trang nhớ được tham chiếu trễ nhất trong tương lai
  - Mỗi trang nhớ được gắn nhãn là một số có giá trị bằng số lệnh sẽ được thực thi trước khi tham chiếu đến trang đó. Các trang sẽ không được truy cập tiếp kề từ vị trí hiện thời sẽ có nhãn là vô cùng lớn.
  - Trang nhớ có nhãn lớn nhất sẽ bị thay thế.
  - Tối ưu số page faults
  - Không thể hiện thực được. Vì sao?
- Ví dụ: Thứ tự các trang nhớ được tham chiếu như sau và Giả sử resident set = 3

Các thuật toán thay thế trang

- Ví dụ: Thứ tự các trang nhớ được tham chiếu như sau và Giả sử resident set = 3
- Chú ý: 2 cột gần cuối cùng: - Thay trang 3 vào vị trí trang 2 vì trang 2 ko được truy cập nữa cho nên có nhãn VCL. Tương tự thay trang 4 vào vị trí trang 3.

	1	2	3	4	1	2	5	1	2	3	4	5
Thời điểm t	0	1	2	3	4	5	6	7	8	9	10	11
Bộ nhớ thực có 3 frames	1	1	1	1	1	1	1	1	1	1	1	1
		2	2	2	2	2	2	2	2	3	4	4
			3	4	4	4	5	5	5	5	5	5

7 page fault

Least Recently Used (LRU)

- Giải thuật thay thế trang "lâu nhất chưa sử dụng" Least Recently Used (LRU)
  - Trong bảng phân trang, mỗi trang được ghi nhận thời điểm được tham chiếu.
  - Chọn trang thay thế là trang đã không được tham khảo trong thời gian lâu nhất

Thôøi ñieåm t	0	1	2	3	4	5	6	7	8	9	10	11
Chuoãi tham khaûo	1	2	3	4	1	2	5	1	2	3	4	5
Boänhôù thöïc coù 3 frame	1	1	1	4	4	4	5	5	5	3	3	5
		2	2	2	1	1	1	1	1	1	4	4
			3	3	3	2	2	2	2	2	2	2

Giải thuật FIFO

- Xem các frame được cấp phát cho process như là circular buffer
- Trang nhớ cũ nhất sẽ được thay thế: first-in, first-out

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

Boänhôù thöïc coù 3 frame

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

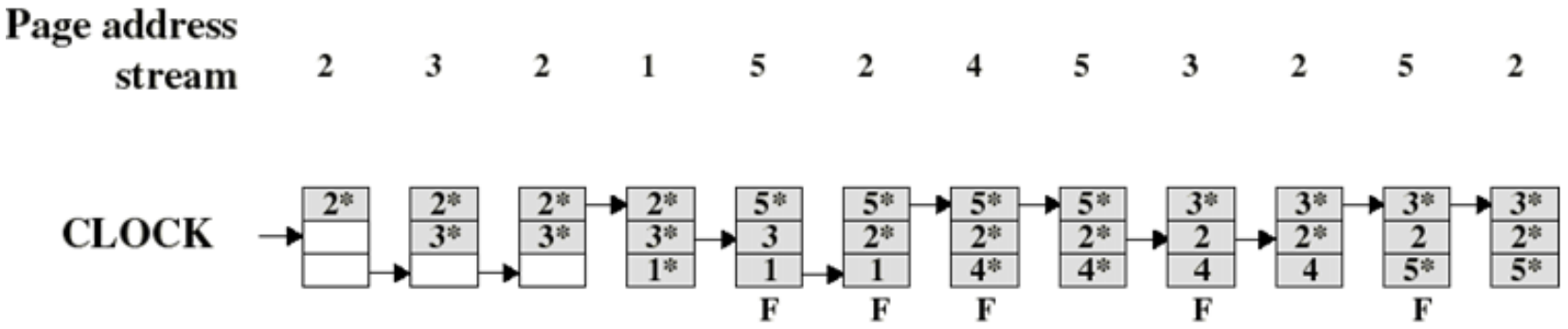
9 page fault

Giải thuật thay thế trang FIFO cải tiến

- Giải thuật thay thế trang FIFO cải tiến (Second Chance – cơ hội thứ 2Ý tưởng của **Second chance** tương tự FIFO nhưng cho cơ hội thứ hai còn có tên là đồng hồ vì dùng bộ đệm quay vòng giống đồng hồ.
- Giải thuật
  - -Resident set – số frame của mỗi tiến trình được hiện thực dạng quay vòng
  - Khi một trang được thay thế, con trỏ sẽ chỉ đến frame kế tiếp trong bộ đệm quay vòng
  - Mỗi frame có một use-bit. Bit này được thiết lập trị 1 khi
    - Trang nhớ được nạp lần đầu vào frame
    - Có tham chiếu tới địa chỉ thuộc trang chứa trong frame
  - Trang được chọn xét thay thế theo kiểu FIFO
  - Khi cần thay thế một trang nhớ, trang nhớ nằm trên frame đầu tiên có use bit bằng 0 sẽ được thay thế.
  - Trong suốt quá trình tìm trang nhớ thay thế, giải thuật clock sẽ reset về giá trị 0 các use-bit của frame trên đường đi qua.

Giải thuật thay thế trang FIFO cải tiến

- Dấu *: use bit tương ứng được thiết lập trị 1
- Giải thuật Clock bảo vệ các trang thường được tham chiếu bằng cách thiết lập use bit bằng 1 với mỗi lần tham chiếu
- Một số kết quả thực nghiệm cho thấy clock có hiệu suất gần với LRU



NRU - Not Recently Used

- **Giải thuật thay thế trang ít được sử dụng gần đây (NRU - Not Recently Used)**
  - Mỗi mục trong page table có thêm 2 bit là M (modified) và R (referenced: read, write)
  - Khởi đầu: R = M = 0
  - Khi trang nhớ được tham chiếu thì thiết lập R = 1
  - Khi có thay đổi nội dung trang nhớ thì thiết lập M = 1
  - Khi có page fault xảy ra, hệ điều hành xem xét tất cả trang nhớ và chia thành 4 loại dựa trên giá trị của R và M
  - Loại 1: không tham chiếu (R=0), không cập nhật (M=0)
  - Loại 2: không tham chiếu (R=0), có cập nhật (M=1)
  - Loại 3: có tham chiếu (R=1), không cập nhật (M=0)
  - Loại 4: có tham chiếu (R=1), có cập nhật (M=1)
  - NRU sẽ thay trang nhớ đầu tiên ở loại nhỏ hơn trước.

LFU (Least Frequently Used)

- Là giải thuật xấp xỉ LRU
- Chọn trang thay thế là trang có tần suất được tham khảo là nhỏ nhất trong 1 khoảng thời gian nhất định
- Tại t=11, nếu trong bộ nhớ còn 3 trang 2, 3, 4 ta sẽ chọn trang 4 để thay thế
-