Merry Christmas

UIT Trường Đại học Công nghệ Thông tin | Khoa Khoa học và Kỹ thuật Thông tin

CHƯƠNG 7: QUẢN LÝ BỘ NHỚ

Các khái niệm

Quản lý bộ nhớ là công việc của hệ điều hành với sự hỗ trợ của phần cứng, sắp xếp các process trong bộ nhớ sao cho hiệu quả.

Mục tiêu: Càng nhiều process vào bộ nhớ càng tốt

Trong hầu hết các hệ thống thi kernel sẽ chiếm một phần cố định trong bộ nhớ.

Ví dụ: RAM 8GB chỉ sử dụng được 7,88GB còn lại là của kernel.

Các yêu cầu đối với việc quản lý bộ nhớ

- Cấp phát tài nguyên cho process
- Tái định vị
- Bảo vệ: Kiểm tra truy xuất bộ nhớ có hợp lệ hay 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.

Các kiểu địa chỉ nhớ

	Địa chỉ luận lí (logical)	Địa chỉ vật lý (Physical)
Phạm vi	Phạm vi tiến trình/ chương trình. Còn được gọi là Virtual address.	Toàn cục.
Nguồn gốc	Tạo ra bởi CPU	Tính toán bởi Memory-Management Unit (Đơn vị quản lý bộ nhớ.
Khả năng truy cập	- Người dùng có thể trực tiếp nhìn thấy ĐCLL của một chương trình. - Người dùng sử dụng ĐCLL để truy cập địa chỉ vật lý.	- Người dùng tương tác và truy cập Địa chỉ Vật lý một cách gián tiếp.

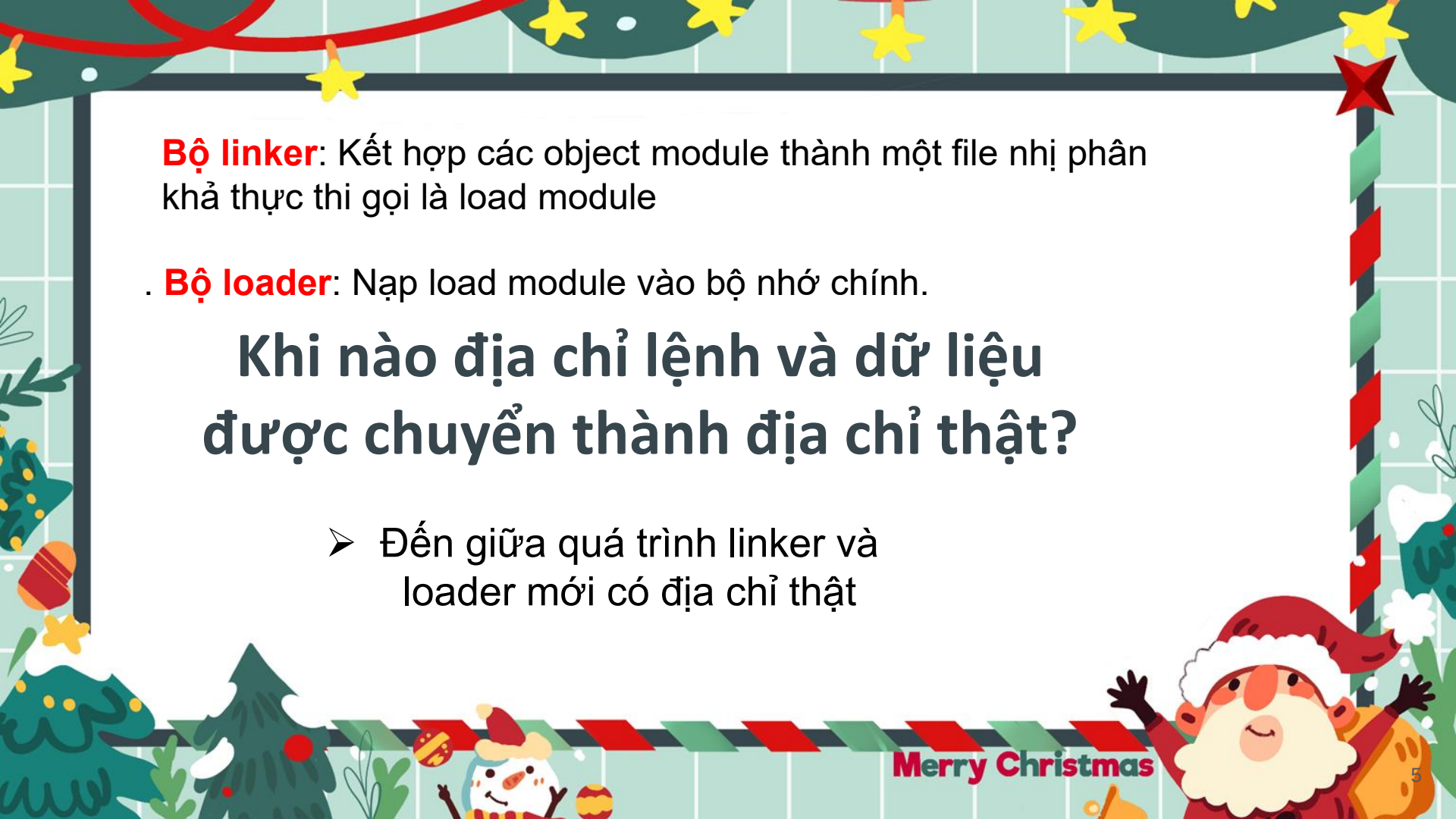
- Ngoài ra có các kiểu địa chỉ khác:
  - **Địa chỉ tương đối** (relative): một kiểu địa chỉ luận lý trong đó các địa chỉ được biểu diễn tương đối so với một vị trí xác định nào đó trong chương trình.
  - **Địa chỉ tuyệt đối** (absolute): địa chỉ tương đương với địa chỉ thực.

Bộ linker: Kết hợp các object module thành một file nhị phân khả thực thi gọi là load module

. **Bộ loader**: Nạp load module vào bộ nhớ chính.

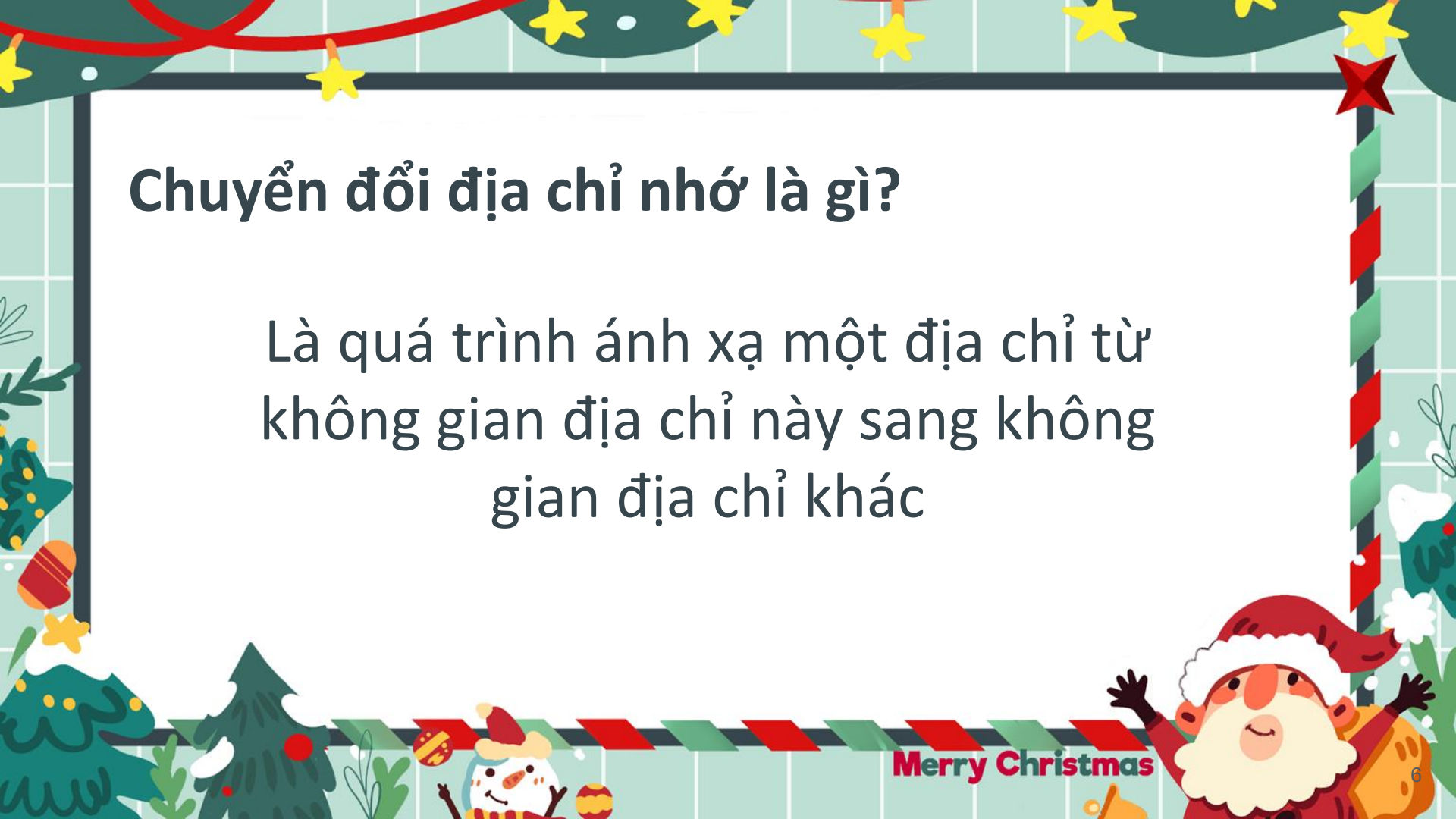
Khi nào địa chỉ lệnh và dữ liệu được chuyển thành địa chỉ thật?

- Đến giữa quá trình linker và loader mới có địa chỉ thật

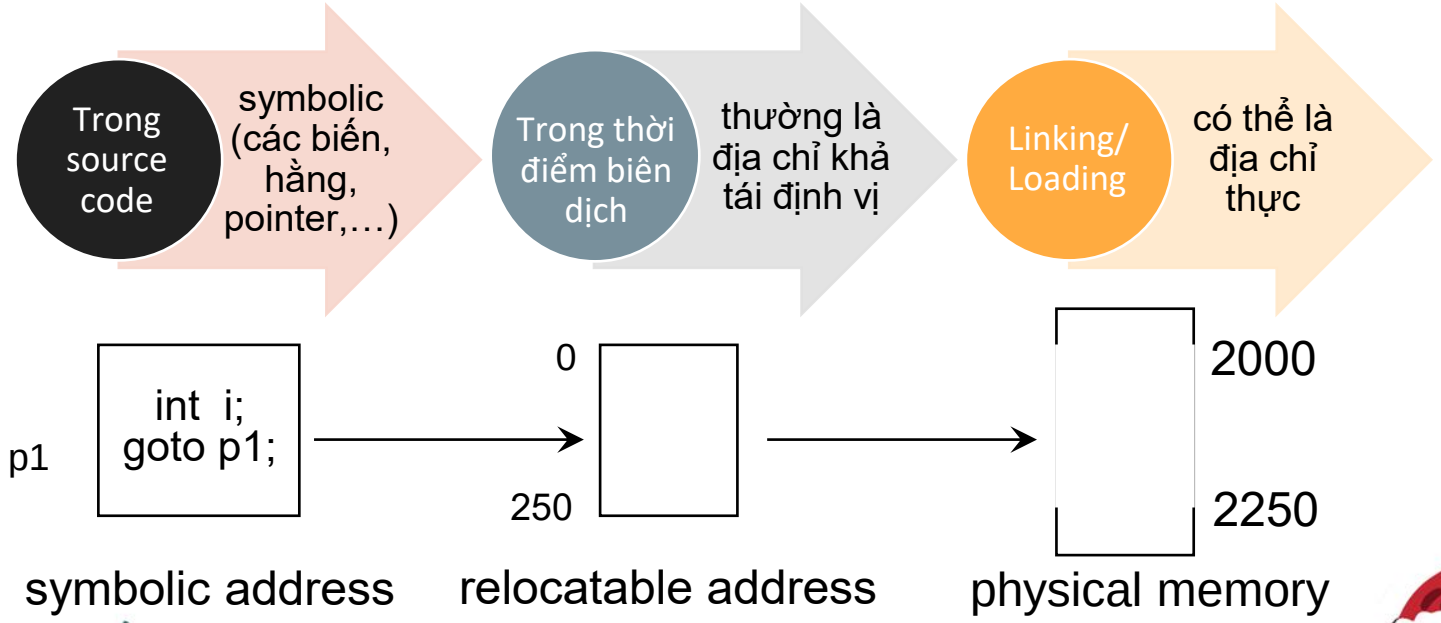


Chuyển đổi địa chỉ nhớ là gì?

Là quá trình ánh xạ một địa chỉ từ không gian địa chỉ này sang không gian địa chỉ khác



Biểu diễn địa chỉ nhớ



Merry Christmas

Biểu diễn 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.

1. Tại thời điểm dịch (compile time):

2. Tại thời điểm nạp (load time):

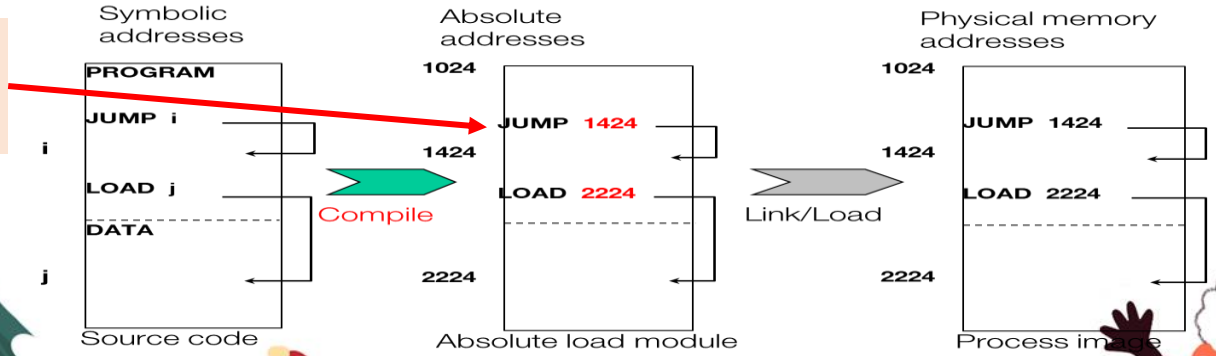
3. Tại thời điểm thực thi (Execution time):

Biểu diễn địa chỉ nhớ

1. Compile time: nếu biết trước địa chỉ bộ nhớ của chương trình thì có thể kết **gán địa chỉ tuyệt đối lúc biên dịch**

- Thường đối với các chương trình hệ thống: chương trình .COM của MS-DOS
- Khuyết điểm: phải biên dịch lại nếu thay đổi địa chỉ nạp chương trình

Gán thẳng địa chỉ thực cho chương trình

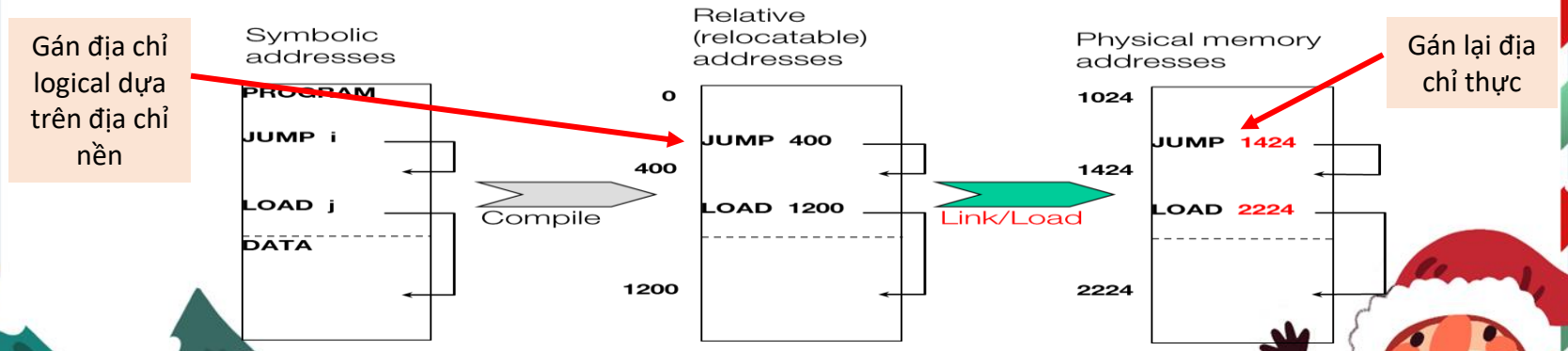


Merry Christmas

Biểu diễn địa chỉ nhớ

2. Load time:

- Gán địa chỉ luận lí cho các biến trong chương trình, dựa trên địa chỉ nền.
- Loader phải chuyển đổi logical address thành physical dựa trên địa chỉ nền.
- Nhược điểm: reload lại chương trình nếu thay đổi địa chỉ nền.



Merry Christmas

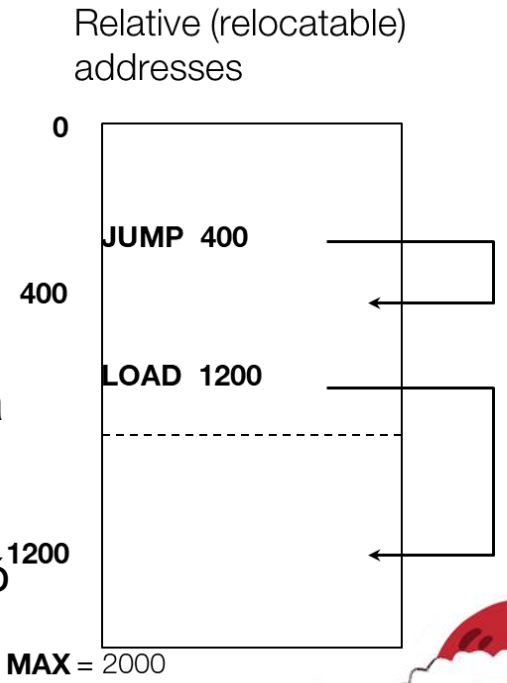
Biểu diễn địa chỉ nhớ

3. 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

- 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



Merry Christmas

Thế nào là dynamic linking?

Là quá trình **liên kết đến một module ngoài** (external module) được thực hiện **sau khi đã tạo xong** loader module.

Ưu điểm

- Cung cấp tiện ích của OS. Chương trình thực thi có thể dùng các module ngoài phiên bản khác nhau mà không cần sửa đổi.
- Chia sẻ mã (code sharing)
- Dynamic linking cần sự hỗ trợ của OS để kiểm tra xem một thủ tục nào đó có thể được chia sẻ giữa các process hay là phần mã của riêng một process (bởi vì chỉ có OS mới có quyền thực hiện việc kiểm tra này).

Thế nào là dynamic loading?

- Cơ chế: Khi cần gọi đến thì một thủ tục mới được nạp vào bộ nhớ chính → tăng độ hiệu dụng của bộ nhớ chính.

- 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.

- User thiết kế chương trình có dynamic loading. Hệ điều hành cung cấp một số thư viện hỗ trợ, tạo điều kiện dễ dàng cho lập trình viên,

Sự khác biệt giữa Linking và Loading

	Linking	Loading
Đầu vào	- Object module của chương trình. - Tất cả các hàm (function) của chương trình. - Các hàm thư viện của thư viện built-in của ngôn ngữ lập trình.	- Executable module của Linking.
Nhiệm vụ	- Liên kết để tạo ra một tập tin sẵn sàng thực thi (executable file) của chương trình.	- Load cái executable file của Linking vào bộ nhớ chính để thực thi.
	Linking liên kết các Module.	Loading phân bổ không gian bộ nhớ chính cho các Executable module.

Sự khác biệt giữa Linking và Loading

Linking tạo ra tệp thực thi của một chương trình trong khi Loading sẽ tải tệp thực thi có được từ Linking vào bộ nhớ chính để thực thi.

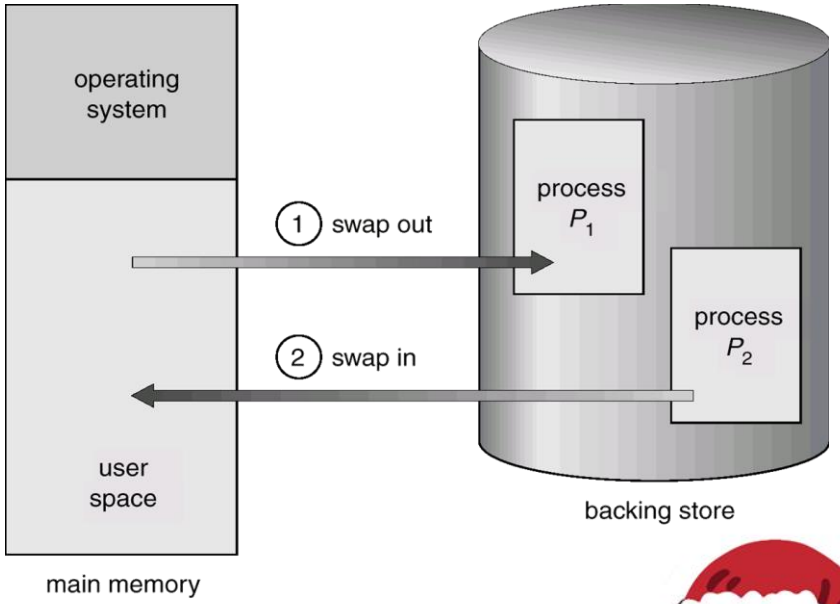
Linking lấy mô-đun đối tượng của chương trình do trình hợp dịch tạo ra. Tuy nhiên, quá trình tải sẽ lấy mô-đun thực thi được tạo bởi Linking.

Linking kết hợp tất cả các mô-đun đối tượng của một chương trình để tạo ra các mô-đun thực thi. Nó cũng liên kết chức năng thư viện trong mô-đun đối tượng với các thư viện tích hợp sẵn của ngôn ngữ lập trình cấp cao. Mặt khác, Loading phân bổ không gian cho một mô-đun thực thi trong bộ nhớ chính.

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.

Hiện nay, ít hệ thống sử dụng cơ chế swapping trên



Các mô hình quản lý bộ nhớ

Mô hình quản lý bộ nhớ là một mô hình đơn giản, không có bộ nhớ ảo.

Một process phải được nạp hoàn toàn vào bộ nhớ thì mới được thực thi.

Phân mảnh (fragmentation)

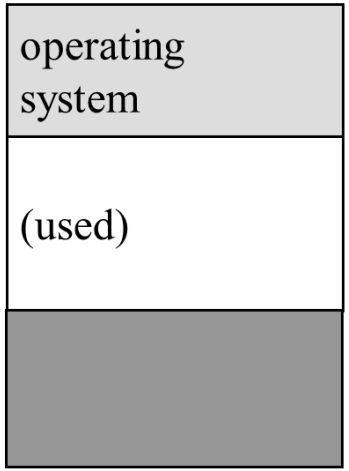
- Phân mảnh ngoại: Kích thước không gian nhớ còn trống đủ để thỏa mãn một yêu cầu cấp phát, tuy nhiên không gian bộ nhớ này không liên tục=> Có thểdùng cơ chế kết khối để gom lại thành một vùng nhớ liên tục.

- Phân mảnh nội: Kích thước vùng nhớ có thể hơi lớn so với yêu cầu. Cần 1GB mà cấp 2GB sẽ lãng phí 1GB. Hệ điều hành sẽ quản lý 1GB không dùng sử dụng cho một mục dích khác.

Phân mảnh (fragmentation)

Internal Fragmentation (Phân mảnh nội)	External Fragmentation (Phân mảnh ngoại)
Xảy ra khi bộ nhớ thực được chia thành các khối cố định.	Xảy ra khi bộ nhớ thực được cấp phát dynamically cho tiến trình.
Sau khi tiến trình chiếm khối nhớ, dư ra không gian bộ nhớ không được sử dụng.	Các không gian bộ nhớ dư ra rải rác, không liên tục, không thể được tận dụng để cấp phát cho các tiến trình khác.
Sự chênh lệch giữa không gian bộ nhớ được cấp và không gian bộ nhớ mà chương trình sử dụng được gọi là phân mảnh nội.	Các không gian bộ nhớ không được sử dụng ở các memory fragments quá nhỏ để cấp cho một tiến trình mới đc gọi là phân mảnh ngoại.
Có thể dùng best-fit để giải quyết.	Compaction, segmentation, và paging

Phân mảnh nội



hole kích thước 18,464 bytes

Yêu cầu kế tiếp là 18,462 bytes !!!

Cần quản lý khoảng trống 2 bytes !?!

OS sẽ cấp phát hẳn khối18,464 bytes cho process
⇒ dư ra 2 bytes không dùng!

Bài tập

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

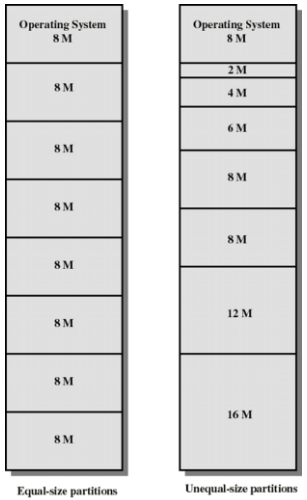
D. Số lỗi trang tăng lên

Các mô hình quản lý bộ nhớ

- Fixed partitioning và chiến lược placement
- Dynamic partitioning và các chiến lược placement

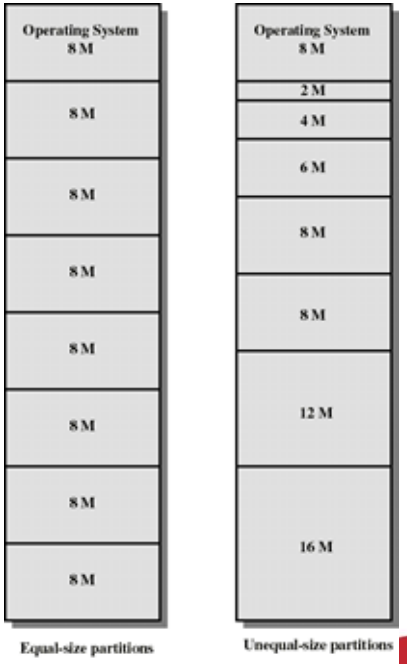
Fixed partitioning

- Khi khởi động hệ thống, bộ nhớ chính được chia thành nhiều phần rời nhau được gọi là partition có kích thước bằng nhau hoặc khác nhau.
- Tiến trình có kích thước < hoặc = kích thước partition thì mới có thể nạp vào partition đó được.
- Nếu chương trình có kích thước lớn hơn dùng cơ chế overlay.



Chiến lược placement

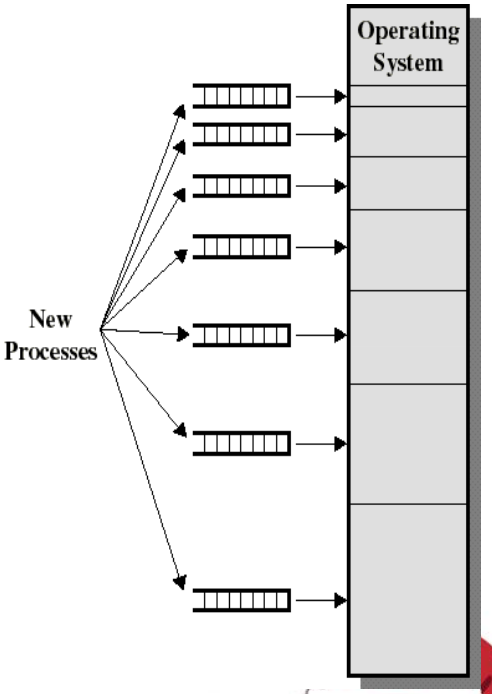
- Chiến lược **placement** đối với fixed partitioning:
  - Partition có kích thước bằng nhau
    - Nếu còn partition trống ⇒ process mới sẽ được nạp vào partition đó
    - Nếu không còn partition trống, nhưng trong đó có process đang bị blocked ⇒ swap process đó ra bộ nhớ phụ nhường chỗ cho process mới.



Merry Christmas

Chiến lược placement

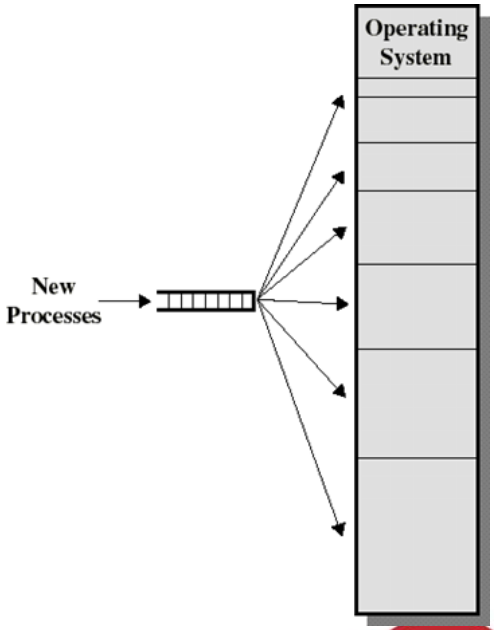
- Chiến lược **placement** đối với fixed partitioning:
  - Partition có kích thước không bằng nhau: giải pháp 1
    - Gán mỗi process vào partition nhỏ nhất phù hợp với nó
    - Có hàng đợi cho mỗi partition
    - Giảm thiểu phân mảnh nội
    - Vấn đề: có thể có một số hàng đợi trống không (vì không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



Merry Christmas

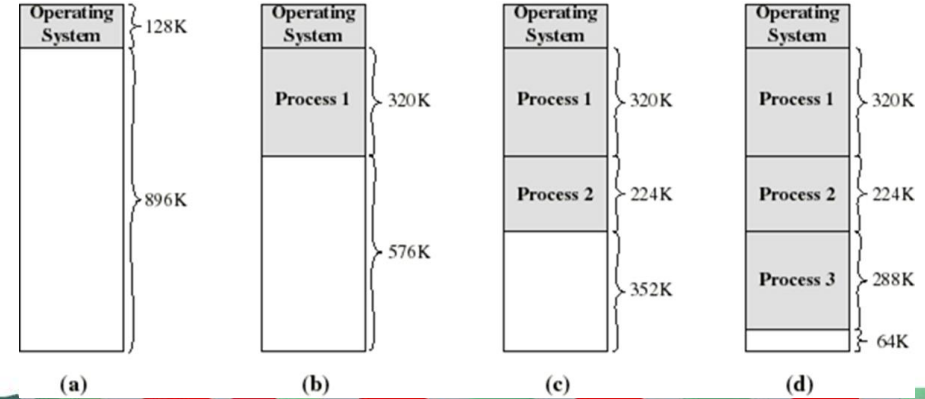
Chiến lược placement

- Chiến lược **placement** đối với fixed partitioning:
  - 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 ⇒ chọn partition nhỏ nhất còn trống



Dynamic partitioning

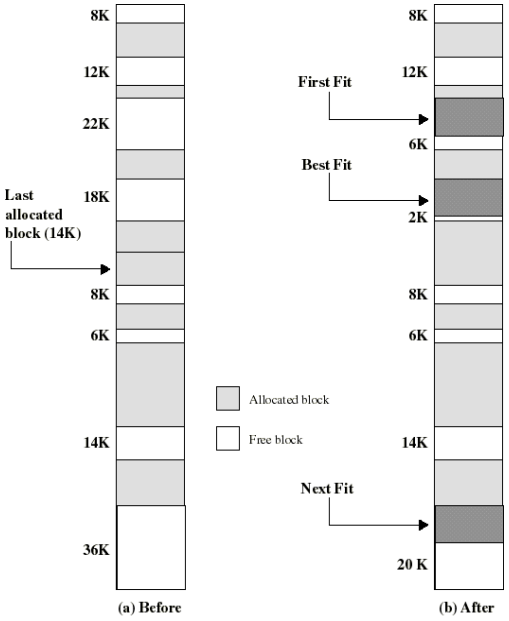
- Số lượng partition là không cố định có thể có kích thước khác nhau
- Mỗi process được cấp dung lượng chính xác cần thiết.
- Gây ra tình trạng phân mảnh ngoại.



Merry Christmas

Chiến lược placement

- Chiến lược **placement** đối với dynamic partitioning:
  - **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*



Example Memory Configuration Before and After Allocation of 16 Kbyte Block

Merry Christmas

Các mô hình quản lý bộ nhớ

Fixed partitioning	Dynamic partitioning
Bộ nhớ được chia thành các ngăn (partition) có kích thước cố định.	Bộ nhớ được chia thành các ngăn có kích thước tương ứng với tiến trình.
Chỉ một tiến trình tại một ngăn.	Tiến trình được cấp phát một phần bộ nhớ trống.
Sử dụng bộ nhớ kém hiệu quả.	Sử dụng bộ nhớ hiệu quả hơn.
Mắc phải vấn đề phân mảnh nội lẫn phân mảnh ngoại.	Mắc phải vấn đề phân mảnh ngoại.
Tính đa chương thấp.	Tính đa chương cao hơn.
Giới hạn kích thước của các tiến trình.	Các tiến trình không bị giới hạn kích thước.

Bài tập

Giả sử bộ nhớ chính được cấp phát các phân vùng có kích thước là 600K, 500K, 200K, 300K (theo thứ tự), sau khi thực thi xong, các tiến trình có kích thước 212K, 417K, 112K, 426K (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào, nếu sử dụng: Thuật toán First fit, Best fit, Next fit, Worst fit?

Thuật toán nào cho phép sử dụng bộ nhớ hiệu quả nhất trong trường hợp trên?

Fisrt fit: 212K sẽ được cấp cho vùng nhớ 600K, 417K được cấp phát cho vùng nhớ 500K, 112K được cấp phát cho vùng nhớ 200K (Có thể lấy 600 -212K = 388K) để cấp cho 112K, 426K phải đợi do không còn vùng nhớ thỏa yêu cầu.

Best fit: 212K sẽ được cấp cho bộ nhớ 300K, 417K sẽ được cấp cho bộ nhớ 500K, 112K sẽ được cấp cho bộ nhớ 200K, 426K sẽ được cấp cho bộ nhớ 600K

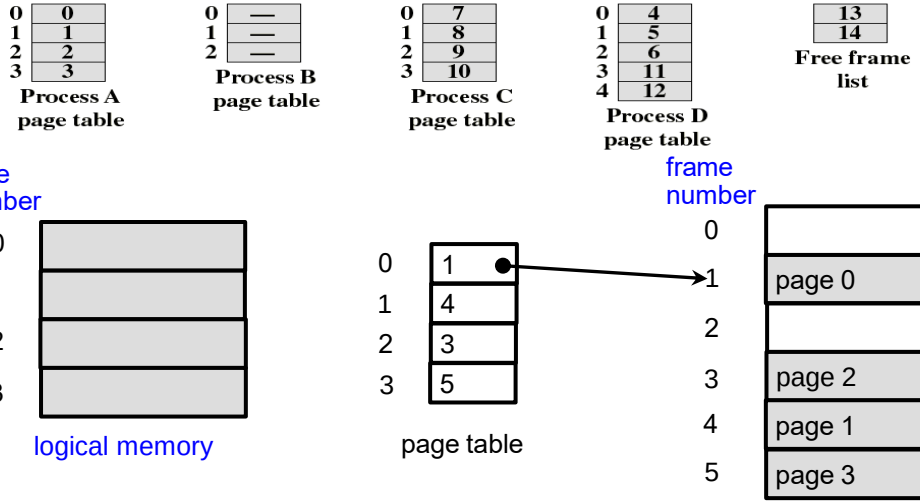
Worst fit: 212K sẽ được cấp cho vùng nhớ 600K, 417K được cấp phát cho vùng nhớ 500K, 112K được cấp phát cho vùng nhớ 300K, 426K phải đợi do không còn vùng nhớ thỏa yêu cầu.

Next fit: 212K sẽ được cấp cho vùng nhớ 600K, 417K được cấp phát cho vùng nhớ 500K, 112K được cấp phát cho vùng nhớ 200K (Có thể lấy 600 -212K = 388K) để cấp cho 112K, 426K phải đợi do không còn vùng nhớ thỏa yêu cầu.

Cơ chế phân trang

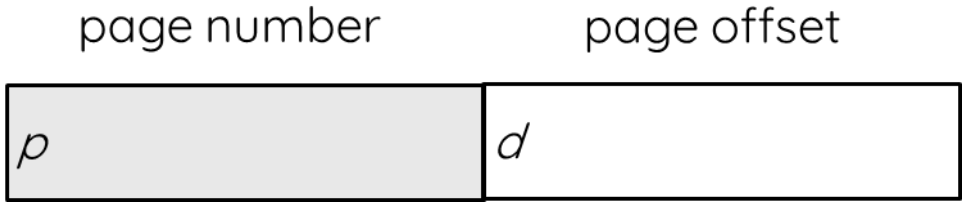
- Bộ nhớ vật lý → khung trang (frame).
  - Kích thước của frame là lũy thừa của 2, từ khoảng 512 byte đến 16MB.
- Bộ nhớ luận lý là tập hợp mọi địa chỉ luận lý mà một chương trình bất kì có thể phát sinh → page.
- Bảng phân trang là bảng dùng để ánh xạ địa chỉ luận lý sang địa chỉ vật lý

Cơ chế phân trang



Lợi ích của cơ chế paging tận dụng các không gian trống rời rạc một cách linh hoạt.

- Địa chỉ luận lý gồm có:
  - Số hiệu trang (Page number) p
  - Địa chỉ tương đối trong trang (Page offset) d



- Kích thước địa chỉ ảo là 2^m, kích thước trang là 2^n
- Kích thước địa chỉlà 2^{m-n}
- Số mục = 2^m / 2^n

Bài tập

1/ Xét một không gian địa chỉ gồm 12 trang, mỗi trang có kích thước 2K. Ánh xạ vào bộ nhớ vật lý có 32 khung trang, Địa chỉ logic gồm bao nhiêu bit? Địa chỉ vật lý gồm bao nhiêu bit? Bảng trang có bao nhiêu mục?

Ta có 12 trang cần 4 bit để lưu địa chỉ.
Offset là 2K = 2048 byte = 2^{11}
➔ Vậy cần tất cả là 15 bit để lưu đại chỉ logic
➔ Số mục là $2^{15} / 2^{11}$ = 16 mục.
Ta có 32 khung frame cần 5 bit để lưu địa chỉ
Offset là 2K = 2^{11}
➔ Vậy cần tất là 16 bit để lưu địa chỉ vật lý.

Bài tập

2/ Một hệ thống máy tính với bộ nhớ chính có kích thước 320MB. Hệ thống sử dụng địa chỉ logic 48 bit. Kích thước trang được sử dụng là 8KB. Yêu cầu xác định các thông số sau:

a. Cho biết số bit dùng cho địa chỉ offset.

b. Số khung trang vật lý.

c. Số trang logic trong không gian tiến trình.

d. Cho địa chỉ logic 20030, yêu cầu đổi sang dạng <p,d>

Bài tập

a) Kích thước một trang: 8KB = 8*1024 = 213 B → cần 13 bit đểbiểu diễn offset

b) $$\frac{\text{Kích thước bộ nhớ vật lý}}{\text{Kích thước trang}} = \frac{320\ \text{MB}}{8\ {}^{\text{KB}}/_{\text{trang}}} = \frac{320 * 2^{20}\ \text{B}}{8 * 2^{10}\ {}^{\text{B}}/_{\text{trang}}} = 40 * 2^{10} = 40960\ \text{trang}$$

c) $$\frac{\text{Kích thước không gian tiến trình}}{\text{Kích thước trang}} = \frac{2^{48}\ \text{B}}{8\ {}^{\text{KB}}/_{\text{trang}}} = \frac{2^{48}\ \text{B}}{2^{13}\ {}^{\text{B}}/_{\text{trang}}} = 2^{35}\ \text{trang}$$

d) Do kích thước trang là 8*1024 = 8192
→ 20030 / 8192 = 2 dư 3646
→ 20030 ~ <p=2, d = 3646>

Bài tập

3/ 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 ảo như hình.

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

Page Table

Ta có địa chỉ vật lý 4100/1024 (kích thước frame) =4 (dư 4).

Ánh xạ vào bảng trang 4 nằm ở địa số trang thứ 1.

Địa chỉ vật ý ở vị trí 1*1024 (1Kbyte frame)+4 = 1028

Cài đặt bảng trang (paging hardware)

- Bảng phân trang được lưu trữ 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 lengh (PTLR) biểu thị kích thước của bảng phân trang

- Có hai cách cài cài đặt là PTBR và TLB (bộ nhớ tạm hay bộ nhớ phụ)
  - Ưu điểm của TLB: tìm kiếm nhanh
  - Nhược điểm: Chưa chắc tìm được, tìm lại lâu.

Effective Access time (EAT)

- Thời gian truy xuất hiệu dụng (EAT)
- Thời gian tìm kiếm trong TLB: ε
- Thời gian 1 chu kỳ xuất hiện bộ nhớ: x
- Thời gian cần thiết để có được chỉ số frame
  - Khi chỉ số trang có trong TLB (hit) $\varepsilon + x$
  - Khi chỉ số trang không có trong TLB (miss) $\varepsilon + x + x$
- Hit ratio: α là tỉ số giữa số lần chỉ số trang được tìm thấy (hit) trong TLB và số lần truy xuất khởi nguồn từ CPU.
  Ví dụ: Tổng số lần truy xuất là 10 nhưng số lần tìm thấy là 5 thì tỉ số 5/10
- EAT = (2- α)x + ε.

Bài tập

Xét một hệ thống sử dụng kỹ thuật phân trang, với bảng trang được lưu trữ trong bộ nhớ chính.

a) Nếu thời gian cho một lần truy xuất bộ nhớ bình thường là 200ns thì mất bao nhiêu thời gian cho một thao tác truy xuất bộ nhớ trong hệ thống này?

b) Nếu sử dụng TLBs với hit-ratio là 75%, thời gian để tìm trong TLBs xem như bằng 0, tính thời gian truy xuất bộ nhớ trong hệ thống

a) Một lần truy xuất bộ nhớ bình thường là 200ns. Một thao tác thực hiện 2 lần truy xuất Vậy thời gian là 2*200 = 400ns.

b) EAT = (2- 0.75) ∗ 200 + 0 =250ns

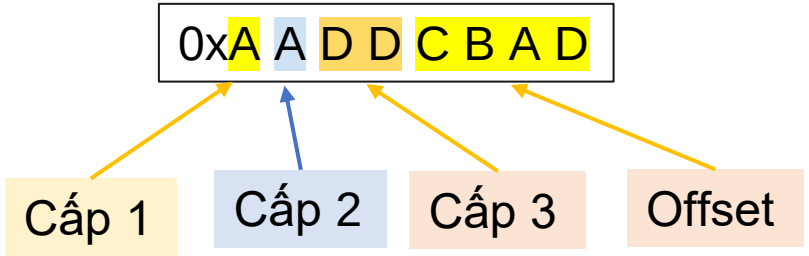
Bảo vệ bộ nhớ

Việc bảo vệ bộ nhớ được hiện thực bằng cách gắn với frame các bit bảo vệ (proctection bits) được giữ trong bảng phân trang. Các biến này được biêu diễn: read -only, read-write, execute-only.

Trắc nghiệm

1/ 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 trình truy xuất địa chỉ 0xAADDCBAD thì offset là bao nhiêu?

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



Merry Christmas

Trắc nghiệm

α

2/ 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-ratio (tỉ lệ tìm thấy) là 90% thì thời gian truy xuất bộ nhớ trong hệ thống (effective memory reference time) là 240 ns. Biết thời gian để tìm trong TLBs là 20 ns, hãy xác định thời gian truy xuất bộ nhớ trong hệ thống nếu tỉ lệ tìm thấy giảm xuống còn 85%?

EAT

ε

A. 200 B. 20 C. 230 D. 250

EAT = (2 – α)x + ε => x = (EAT – ε) / (2 – α) = (240 – 20) / (2- 0.9) = 200

EAT = (2 - 0.85)*200 + 20 = 250

Merry Christmas

Trắc nghiệm

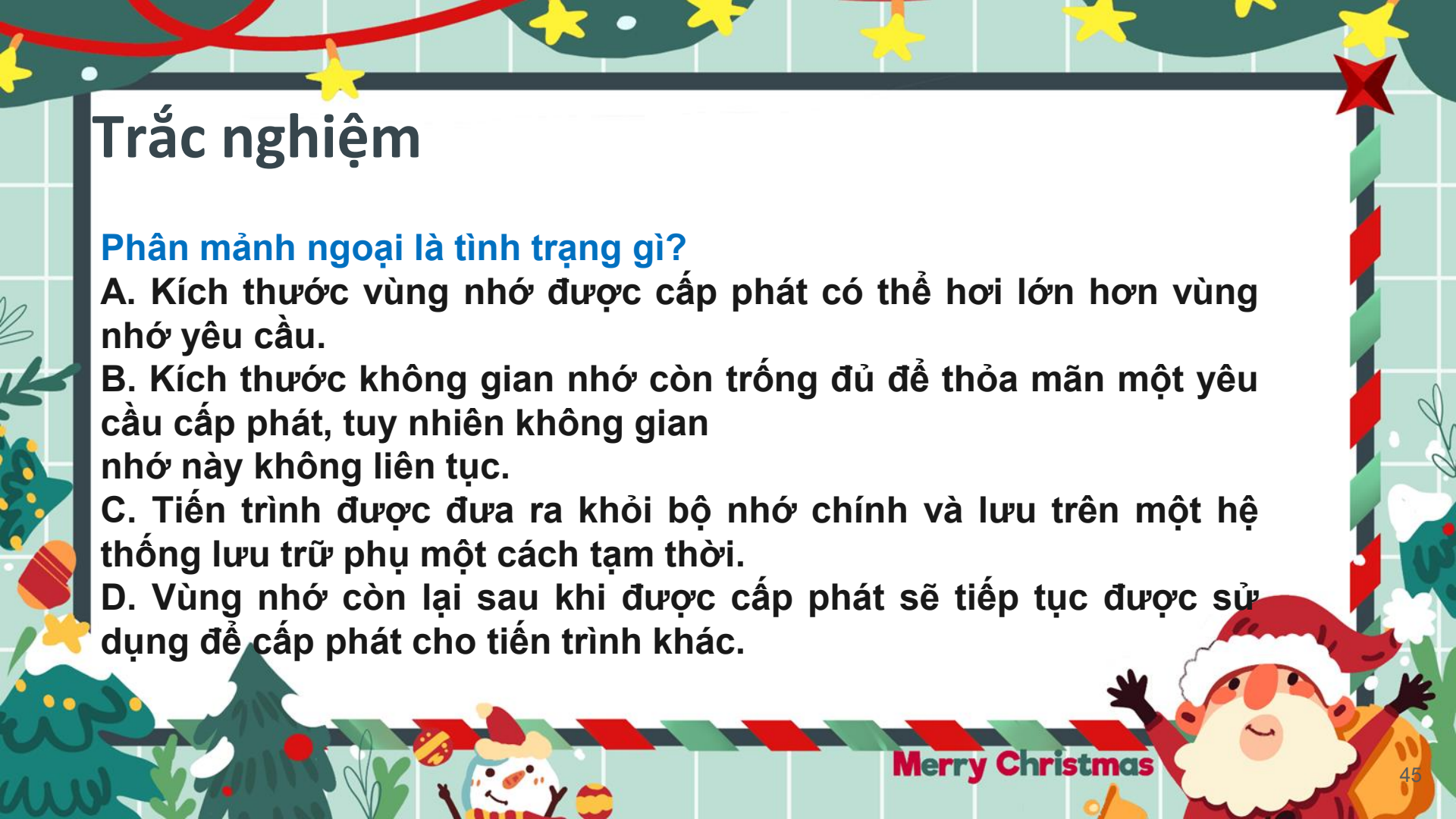
Phân mảnh ngoại là tình trạng gì?

A. Kích thước vùng nhớ được cấp phát có thể hơi lớn hơn vùng nhớ yêu cầu.

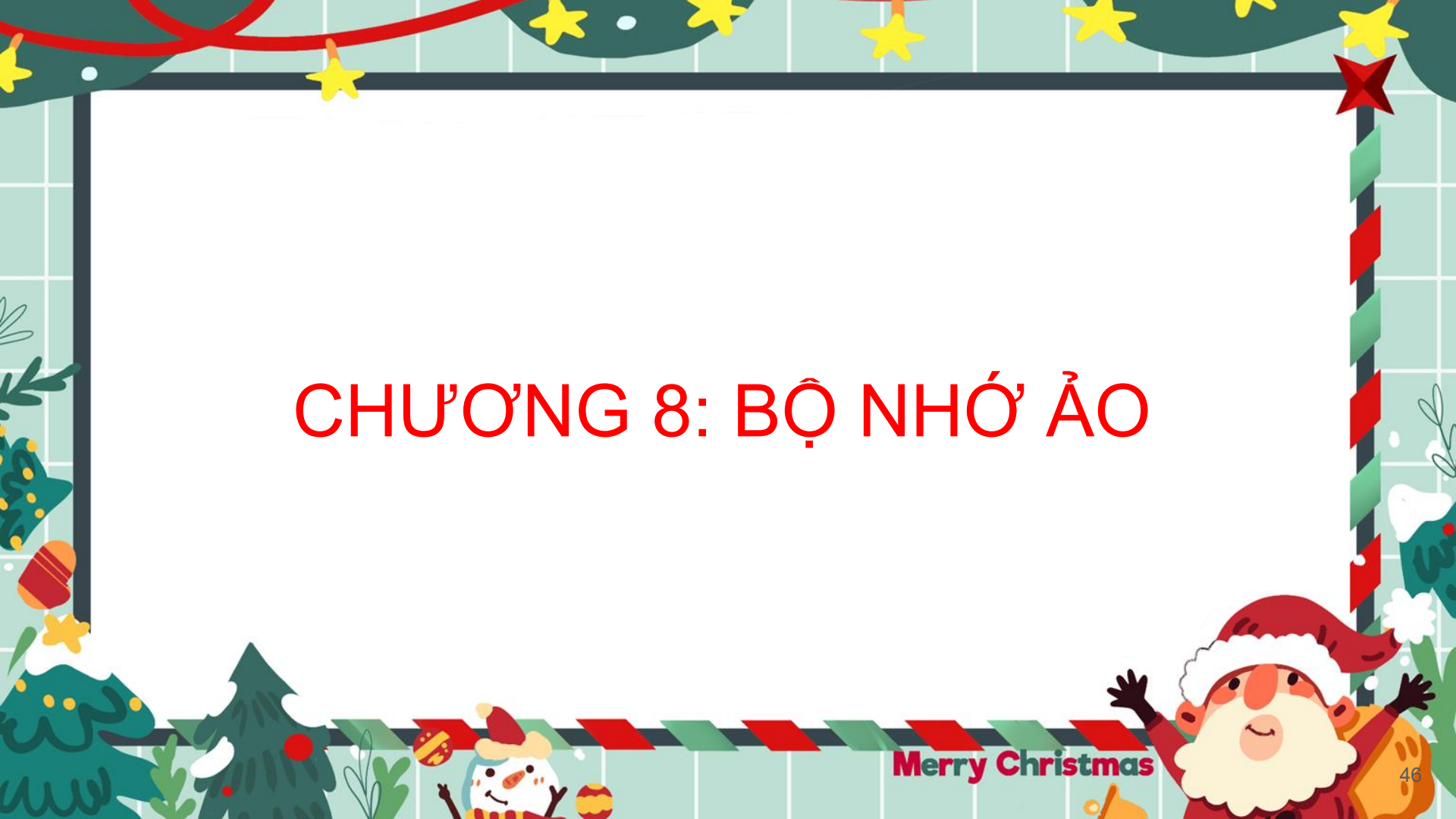
B. Kích thước không gian nhớ còn trống đủ để thỏa mãn một yêu cầu cấp phát, tuy nhiên không gian nhớ này không liên tục.

C. Tiến trình được đưa ra khỏi bộ nhớ chính và lưu trên một hệ thống lưu trữ phụ một cách tạm thời.

D. Vùng nhớ còn lại sau khi được cấp phát sẽ tiếp tục được sử dụng để cấp phát cho tiến trình khác.



CHƯƠNG 8: BỘ NHỚ ẢO



Tại sao cần phải có bộ nhớ ảo?

- Bộ nhớ ảo là một kĩ thuật cho phép nạp một phần tiến trình vào bộ nhớ.
- Có những phần không cần phải luôn luôn nạp vào bộ nhớ:
  - Những lệnh điều kiện rất ít khi xảy ra.
  - Các tính năng ít dùng…
- Ưu điểm:
  - Giúp nạp nhiều hơn chương trình vào bộ nhớ.
  - Có thể thực thi những chương trình yêu cầu bộ nhớ lớn hơn bộ nhớ thực.
  - Giảm gánh nặng của lập trình viên.
- Kỹ thuật cài đặt bộ nhớ ảo:
  - Phân trang theo yêu cầu (Demand Paging)
  - Phân đoạn theo yêu cầu (Demand Segmentation)

Phân trang theo yêu cầu

- **Demand paging**: các trang của quá trình chi được nạp vào bộ nhớ chính khi có yêu cầu.

- Khi có một tham chiếu đến trang mà trong đó không nằm trong bộ nhớ chính (present bit =0) thì phần cứng sẽ gây ra một ngắt (page-fault trap) kích khởi page –fault service routine (PFSR) của OS..

PFSR:

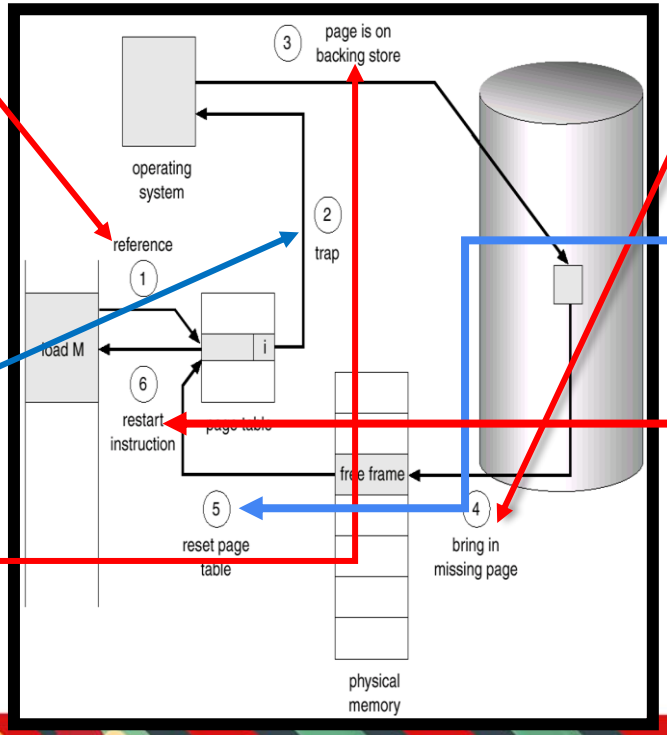
- Chuyền process về trạng thái blocked.
- Phát yêu cầu đọc đĩa để nạp trang được tham chiếu vào 1 frame trống. Lúc này một process khác vào CPU để thực thi.
- Sauk hi I/O hoàn tất, đĩa lại gây ra 1 ngắt đến hệ điều hành. PFSR sẽ cập nhật lại pagetable và chuyển process về trạng thái ready

Phân trang theo yêu cầu (Demand Paging)

1. Chương trình tham chiếu đến một trang không có trong bộ nhớ chính.

2. Phần cứng sẽ gây ra một ngắt (page-fault trap) khởi động dịch vụ *page-fault service routine (PFSR)* của Hệ điều hành

3. Trang mà chương trình cần đang nằm trên bộ nhớ thứ cấp (bộ nhớ tạm, đĩa,…)



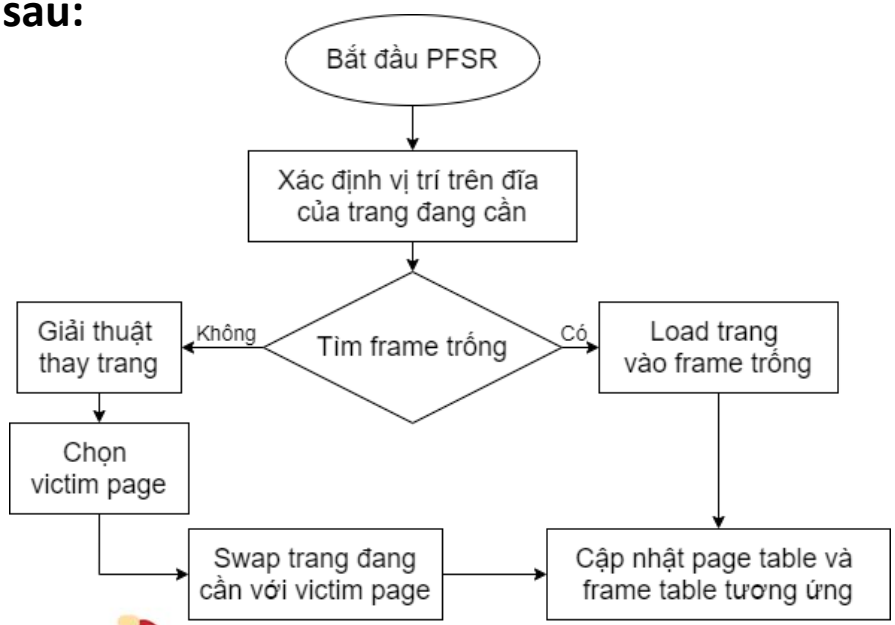
4. Tìm một frame trống và load trang vào frame trống ấy.
- Hoặc sử dụng các giải thuật thay thế trang (FIFO, LRU, OPT).

5. Cập nhật lại page table và frame table tương ứng.

6. Khởi động lại process khi đã có nội dung page mà nó yêu cầu.

Thay thế trang nhớ

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



Thay thế trang nhớ

Có 2 vấn đề cần giải quyết:

- Cấp phát cho process bao nhiêu frame của bộ nhớ thực?
- Giải thuật thay thế trang như thế nào là phù hợp?
  - Mục tiêu là làm sao cho lượng page fault nhỏ nhất.
  - Được đánh giá bằng cách thực thi giải thuật đối với một chuỗi tham chiếu bộ nhớ và xác định số lần xảy ra page fault

Bài tập

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 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**

Chiến lược cấp phát động (variable-allocation)
Số frame cấp cho mỗi tiến trình có thể thay đổi trong khi nó thực thi:
Nếu tỷ lệ page-fault cao ⇒ cấp thêm frame
Nếu tỷ lệ page-fault thấp ⇒ giảm bớt frame
OS phải mất chi phí để ước định các tiến trình

Giải thuật thay trang

- FIFO (tới trước thay trước)
- LRU (Least recently used)
- OPT (Giải thuật optimal)

Giải thuật thay trang

	FIFO	OPT	LRU
Chọn trang	Trang ở trong bộ nhớ lâu nhất.	Trang sẽ được tham chiếu trễ nhất trong tương lai.	Trang ít được tham chiếu nhất.
Nhược điểm	Cài đặt đơn giản nhưng dễ mắc phải nghịch lý Belady.	Khó hiện thực vì không thể dự đoán chính xác thời điểm một trang được tham chiếu.	Đòi hỏi sự trợ giúp từ phần cứng để sắp xếp thứ tự các trang theo thời điểm tham chiếu.

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

Bài tập

Giả sử có 3 khung trang và các khung trang ban đầu là rỗng. Xác định số Page Fault sử dụng FIFO, LRU, OPT?

0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

T0: tham chiếu **page 0**, chưa có trong bộ nhớ nên lỗi trang.

T1: tham chiếu **page 2**, chưa có trong bộ nhớ nên lỗi trang

T2: tham chiếu **page 1**, chưa có trong bộ nhớ nên lỗi trang.

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	6	6							
	2	2	2	4							
		1	1	1							
*	*	*	*	*							

T3: tham chiếu **page 6**, chưa có trong bộ nhớ nên lỗi trang. Chọn page 0 vì p0 ở lâu nhất.

T4: tham chiếu **page 4**, chưa có trong bộ nhớ nên lỗi trang. Chọn page 2.

Merry Christmas

Bài tập

FIFO: Tiếp tục tương tự như bảng tổng cộng 9 lỗi trang.

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	6	6	6	1	1	1	1	1	1
	2	2	2	4	4	4	4	3	3	3	3
		1	1	1	0	0	0	0	0	2	2
*	*	*	*	*	*	*		*		*	

Bài tập

OPT: 7 lỗi trang

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	0	0	0	0	0	3	3	2	2
	2	2	6	4	4	4	4	4	4	4	4
		1	1	1	1	1	1	1	1	1	1
*	*	*	*	*				*		*	

Bài tập

LRU: 9 lỗi trang.

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	6	6	6	1	1	1	1	1	1
	2	2	2	4	4	4	4	3	3	3	3
		1	1	1	0	0	0	0	0	2	2
*	*	*	*	*	*	*		*		*	

Bài tập

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

Có bao nhiêu lỗi trang xảy ra khi sử dụng các thuật toán thay thế sau đây, giả sử có lần lượt là 2, 3, 4, 5 khung trang.

a. LRU

b. FIFO

c. Chiến lược tối ưu (OPT)

Bài tập

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

a) LRU vơi 2 khung trang 18 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	3	3	2	2	5	5	2	2	2	2	7	7	3	3	1	1	3	3
	2	2	4	4	1	1	6	6	1	1	3	3	6	6	2	2	2	2	6
x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x		x	x

LRU với 3 khung trang 15 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6
x	x	x	x		x	x	x	x	x		x	x	x		x	x			x

Bài tập

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

LRU, 4 khung trang 10 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1
x	x	x	x			x	x				x	x	x			x			

LRU, 5 khung trang 8 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
x	x	x	x			x	x				x	x							

Bài tập

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

b) FIFO vơi 2 khung trang 18 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	3	3	2	2	5	5	2	2	2	3	3	6	6	2	2	2	3	3
	2	2	4	4	1	1	6	6	1	1	1	7	7	3	3	1	1	1	6
x	x	x	x	x	x	x	x	x	x		x	x	x	x	x	x		x	x

FIFO với 3 khung trang 16 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
		3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3
x	x	x	x		x	x	x	x	x		x	x	x		x	x		x	x

Bài tập

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

FIFO, 4 khung trang 14 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	5	5	5	5	5	3	3	3	3	3	1	1	1	1
	2	2	2	2	2	2	6	6	6	6	6	7	7	7	7	7	7	3	3
		3	3	3	3	3	3	2	2	2	2	2	6	6	6	6	6	6	6
			4	4	4	4	4	4	1	1	1	1	1	1	2	2	2	2	2
x	x	x	x			x	x	x	x		x	x	x		x	x		x	

FIFO, 5 khung trang 10 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
x	x	x	x			x	x		x	x	x	x							

Bài tập

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

c) OPT vơi 2 khung trang 17 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	3	4	4	1	5	6	6	1	1	3	3	3	3	3	1	1	3	6
	2	2	2	2	2	2	2	2	2	2	2	7	6	6	2	2	2	2	2
x	x	x	x	x	x	x	x		x		x	x	x		x	x	x	x	x

OPT với 3 khung trang 11 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	6
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	1
x	x	x	x			x	x				x	x			x	x			x

Trắc nghiệm

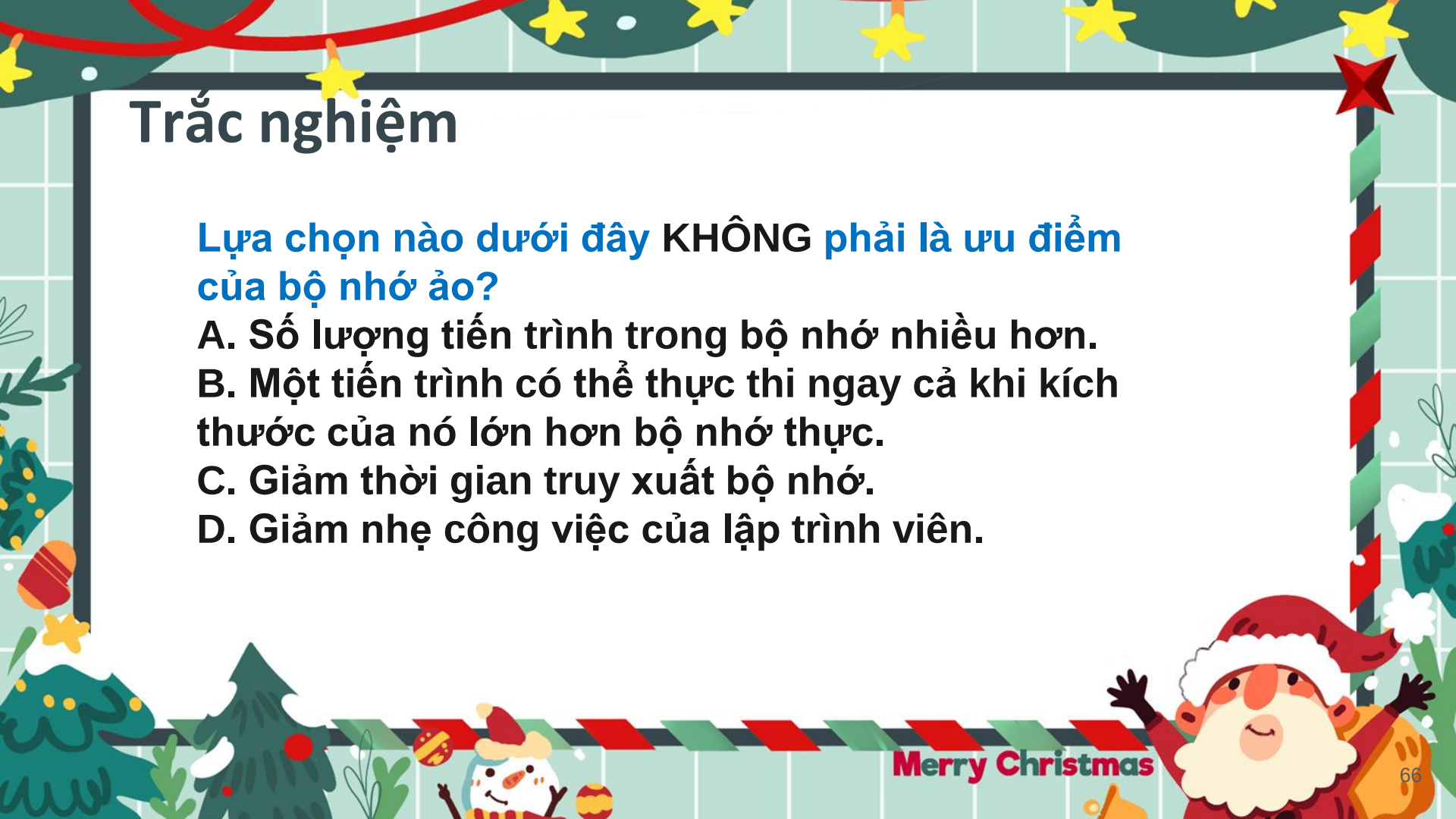
Lựa chọn nào dưới đây KHÔNG phải là ưu điểm của bộ nhớ ảo?

A. Số lượng tiến trình trong bộ nhớ nhiều hơn.

B. Một tiến trình có thể thực thi ngay cả khi kích thước của nó lớn hơn bộ nhớ thực.

C. Giảm thời gian truy xuất bộ nhớ.

D. Giảm nhẹ công việc của lập trình viên.



CHÚC CÁC BẠN THI ĐẠT KẾT QUẢ TỐT

Merry Christmas

ĐIỂM DANH VÀ GOP Ý TRAINING

SCAN ME

Trắc nghiệm

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 cho một tiến trình sẽ thay đổi như thế nào nếu tỷ lệ lỗi trang (page fault) cao?

A. Giảm xuống

B. Tăng lên

C. Không thay đổi

D. Bị hệ thống thu hồi toàn bộ