BÀI 4

QUẢN LÝ BỘ NHỚ

TÌNH HUỐNG KHỞI ĐỘNG

- Nếu đặt toàn bộ không gian địa chỉ vào bộ nhớ vật lý, thì kích thước của chương trình bị giới hạn bởi kích thước vật lý.
- Thực tế, trong nhiều trường hợp, chúng ta không cần phải nạp toàn bộ chương trình vào bộ nhớ vật lý, vì tại một thời điểm chỉ có một chỉ thị của tiến trình được xử lý. Ví dụ, các chương trình đều có một đoạn code xử lý lỗi, nhưng đoạn code này hầu như rất ít sử dụng vì hiếm khi xảy ra lỗi, trong trường hợp này, không cần thiết phải nạp đoạn code xử lý lỗi từ đầu.
- Từ những khó khăn trong việc xử lý một chương trình có kích thước lớn chỉ với một vùng nhớ có kích thước nhỏ. Giải pháp được tìm thấy với khái niệm bộ nhớ ảo.

? Máy tính lưu trữ dữ liệu bằng cách nào? Và như thế nào? Câu trả lời đó là máy tính lưu trong bộ nhớ vật lý. Học xong bài học này chúng ta sẽ biết được cách tổ chức, sắp xếp dữ liệu ra sao.

MỤC TIÊU

- Mô tả các khái niệm cơ bản, các cơ chế tổ chức, chế độ bảo vệ và các chiến lược điều khiển bộ nhớ
- Nắm được các khái niệm cơ bản về bộ nhớ ảo, hiểu được cách tổ chức theo trang, đoạn của bộ nhớ ảo.
- Nắm vững được các chiến lược điều khiển bộ nhớ ảo, chiến lược loại bỏ trang.

NỘI DUNG

1. Khái niệm bộ nhớ vật lý;
2. Tổ chức bộ nhớ ảo;
3. Quản lý bộ nhớ ảo.
4. Thuật toán thay thế trang.

1. BỘ NHỚ VẬT LÝ

1. BỘ NHỚ VẬT LÝ

1.1. Các khái niệm cơ sở;

1.2. Các kiểu địa chỉ nhớ;

1.3. Chuyển đổi địa chỉ nhớ;

1.4. Overlay;

1.5. Swapping;

1.6. Mô hình quản lý bộ nhớ đơn giản

1. BỘ NHỚ VẬT LÝ

1.1. CÁC KHÁI NIỆM CƠ SỞ

- Định nghĩa: Quản lý bộ nhớ là công việc của hệ điều hành với sự hỗ trợ của phần cứng nhằm phân phối, sắp xếp các tiến trình trong bộ nhớ sao cho hiệu quả;
- Mục tiêu: Cần đạt được là nạp càng nhiều tiến trình 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 tiến trình;
- Các yêu cầu đối với việc quản lý bộ nhớ:
  - Cấp phát bộ nhớ cho các tiến trình;
  - Tái định vị (Relocation): Khi Swapping,…;
  - Bảo vệ: Phải kiểm tra truy xuất bộ nhớ có hợp lệ không;
  - Chia sẻ: Cho phép các tiến trình chia sẻ vùng nhớ chung;
  - Kết gán địa chỉ nhớ luận lý của người sử dụng vào địa chỉ thực.

1. BỘ NHỚ VẬT LÝ

1.2. CÁC KIỂU ĐỊA CHỈ 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;
- Các trình biên dịch (Compiler) tạo ra mã lệnh chương trình mà trong đó mỗi tham chiếu bộ nhớ để là địa chỉ luận lý;
- 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;

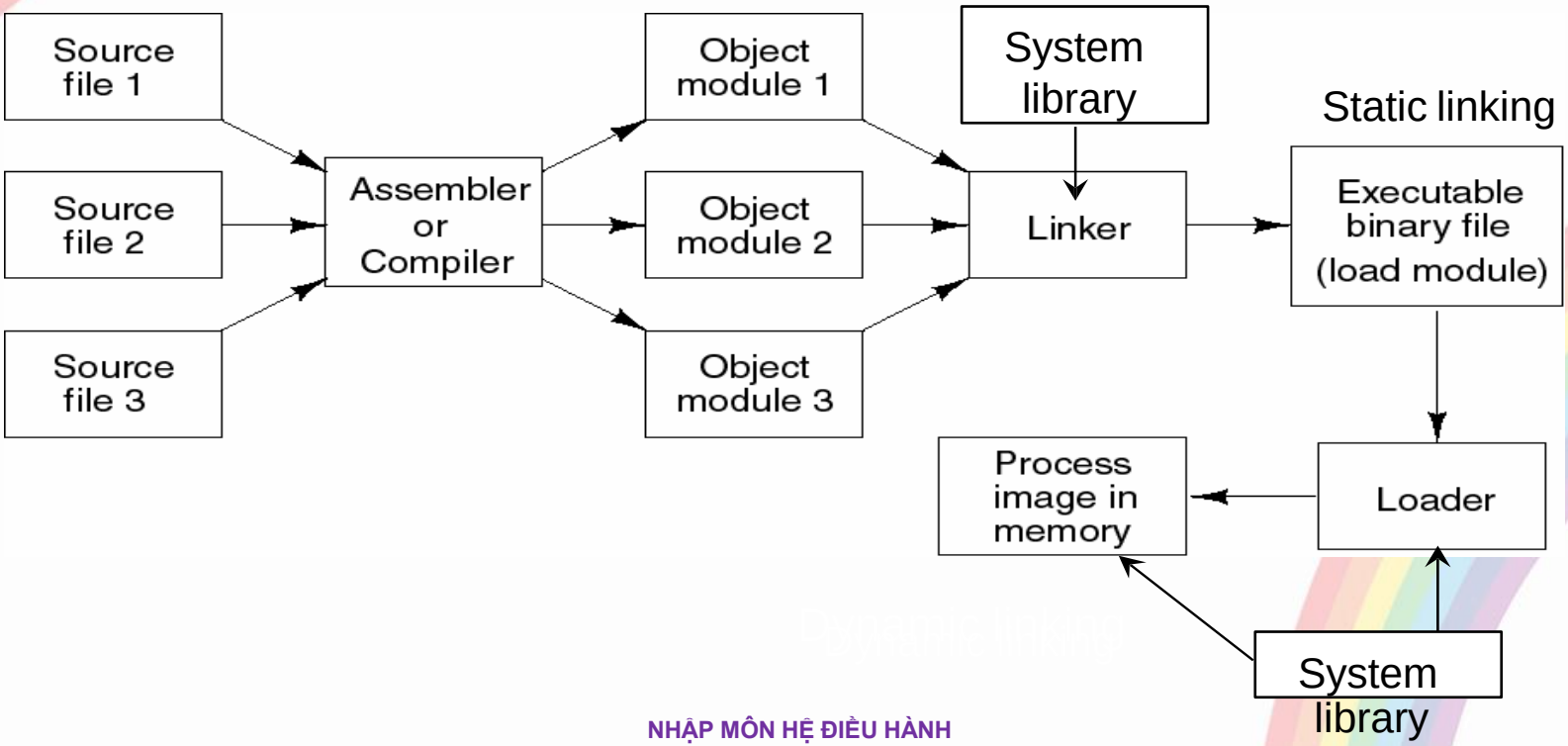
Ví dụ: 12 byte so với vị trí bắt đầu chương trình, …;

- Địa chỉ tuyệt đối (Absolute Address): Địa chỉ tương đương với địa chỉ thực;
- Khi một lệnh được thực thi, các tham chiếu đến địa chỉ luận lý phải được chuyển đổi thành địa chỉ thực. Thao tác chuyển đổi này thường có sự hỗ trợ của phần cứng để đạt hiệu suất cao.

1. BỘ NHỚ VẬT LÝ

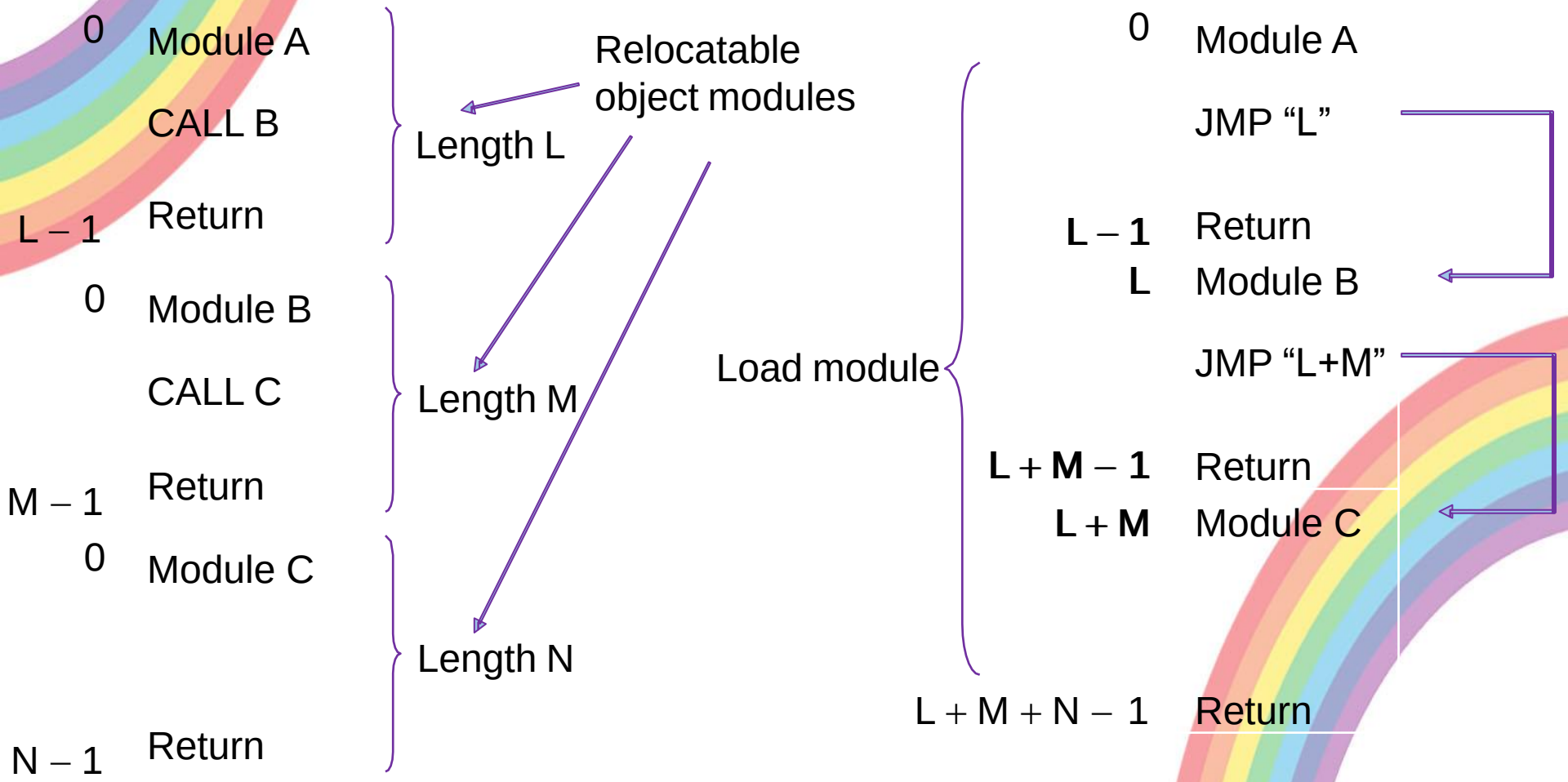
1.2. CÁC KIỂU ĐỊA CHỈ NHỚ
1.2.1. NẠP CHƯƠNG TRÌNH VÀO BỘ NHỚ

- 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.



1. BỘ NHỚ VẬT LÝ

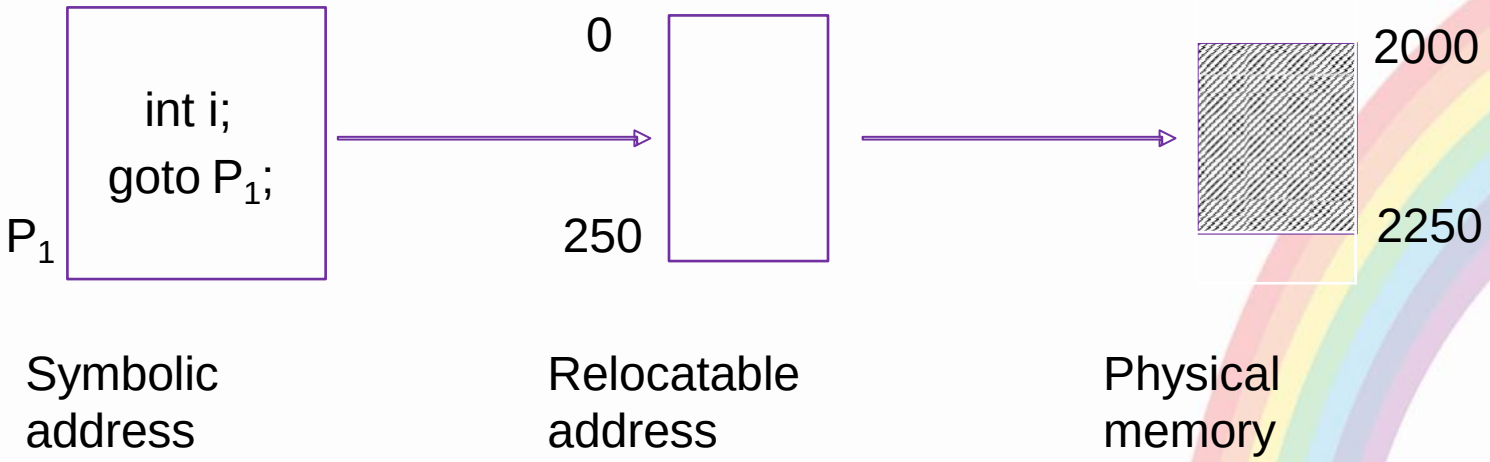
1.2. CÁC KIỂU ĐỊA CHỈ NHỚ
1.2.2. CƠ CHẾ THỰC HIỆN LINKING



1. BỘ NHỚ VẬT LÝ

1.3. CHUYỂN ĐỔI ĐỊA CHỈ

- Chuyển đổi địa chỉ: Quá trình ánh xạ một địa chỉ từ không gian địa chỉ này sang không gian địa chỉ khác.
- Biểu diễn địa chỉ nhớ:
  - 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 14 bytes so với vị trí bắt đầu của Module.
  - Thời điểm Linking/Loading: Có thể là địa chỉ thực. Ví dụ: Dữ liệu nằm tại địa chỉ bộ nhớ thực 2030.



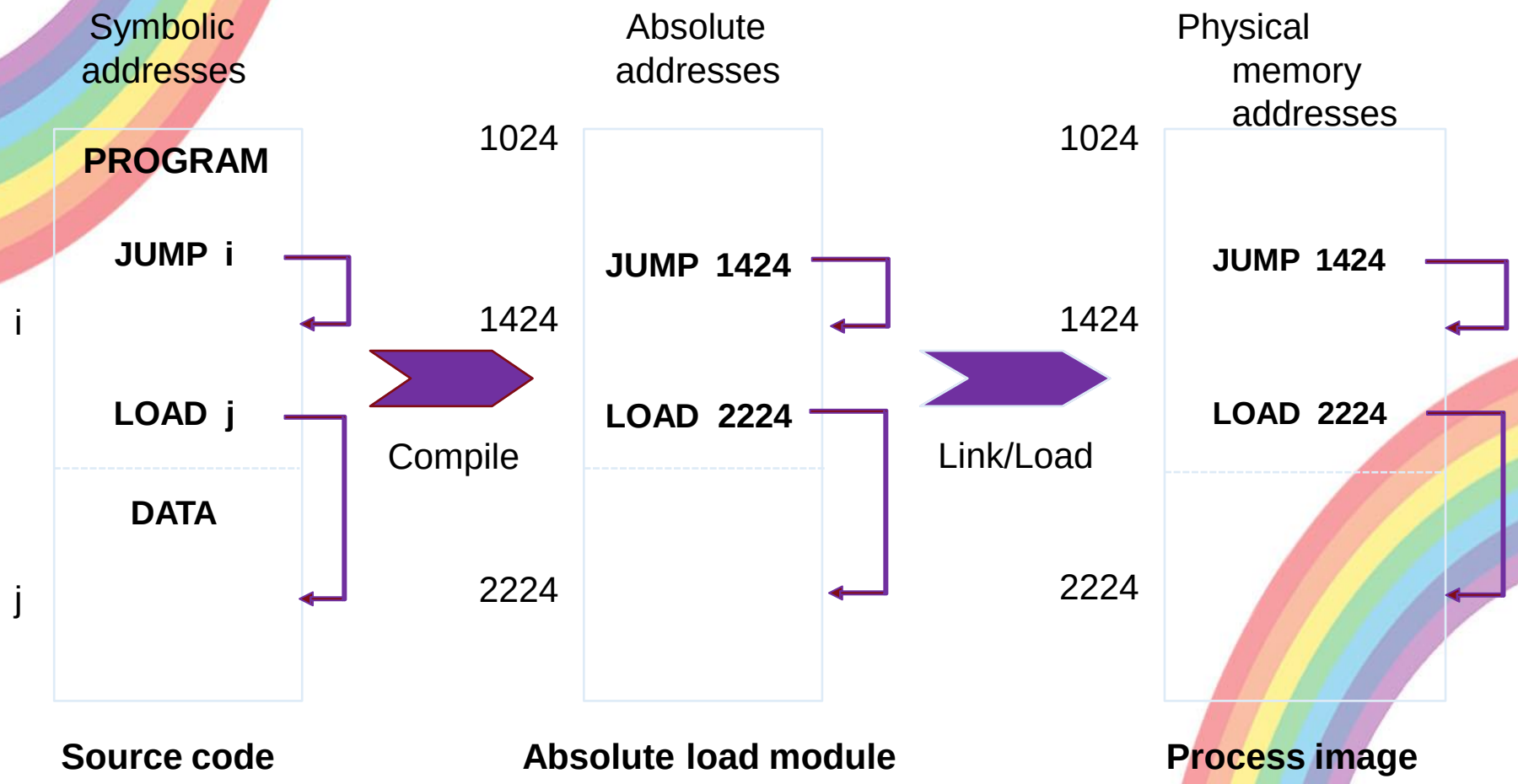
1. BỘ NHỚ VẬT LÝ

1.3. CHUYỂN ĐỔI ĐỊA CHỈ

- Địa chỉ lệnh (Instruction) và dữ liệu (Data) 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:
  - Ví dụ: Chương trình .com của MS-DOS, phát biểu Assembly org xxx.
  - Khuyết điểm: Phải biên dịch lại nếu thay đổi địa chỉ nạp chương trình.
- Load Time: Tại thời điểm biên dịch, nếu chưa biết quá trình sẽ nằm ở đâu trong bộ nhớ thì Compiler phải sinh mã khả tái định vị. Vào thời điểm Loading, Loader phải chuyển đổi địa chỉ khả tái định vị thành địa chỉ thực dựa trên một địa chỉ nền (Base Address).
- Địa chỉ thực được tính toán vào thời điểm nạp chương trình → phải tiến hành Reload nếu địa chỉ nền thay đổi.

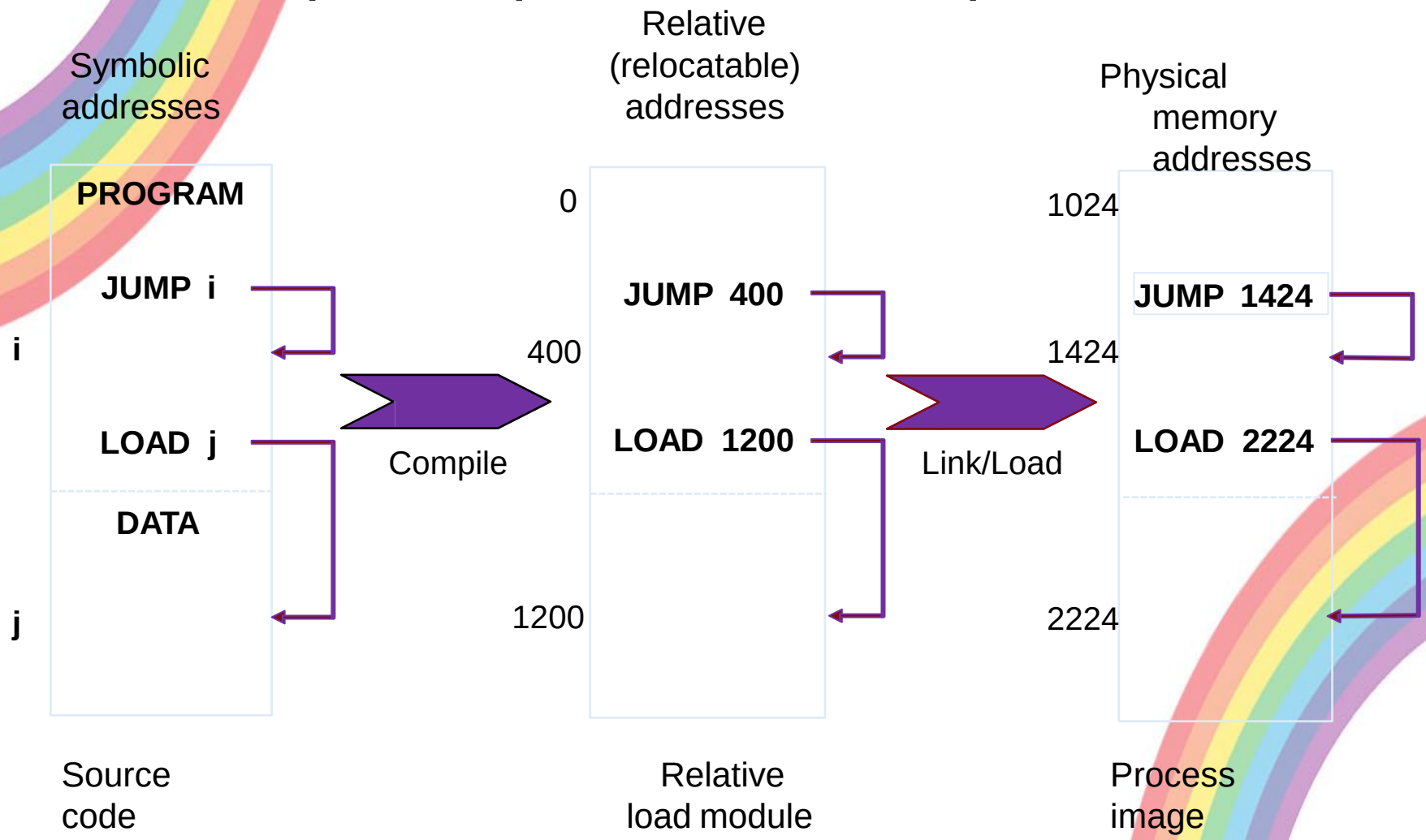
1. BỘ NHỚ VẬT LÝ

1.3. CHUYỂN ĐỔI ĐỊA CHỈ
1.3.1. SINH ĐỊA CHỈ THỰC VÀO THỜI ĐIỂM DỊCH



1. BỘ NHỚ VẬT LÝ

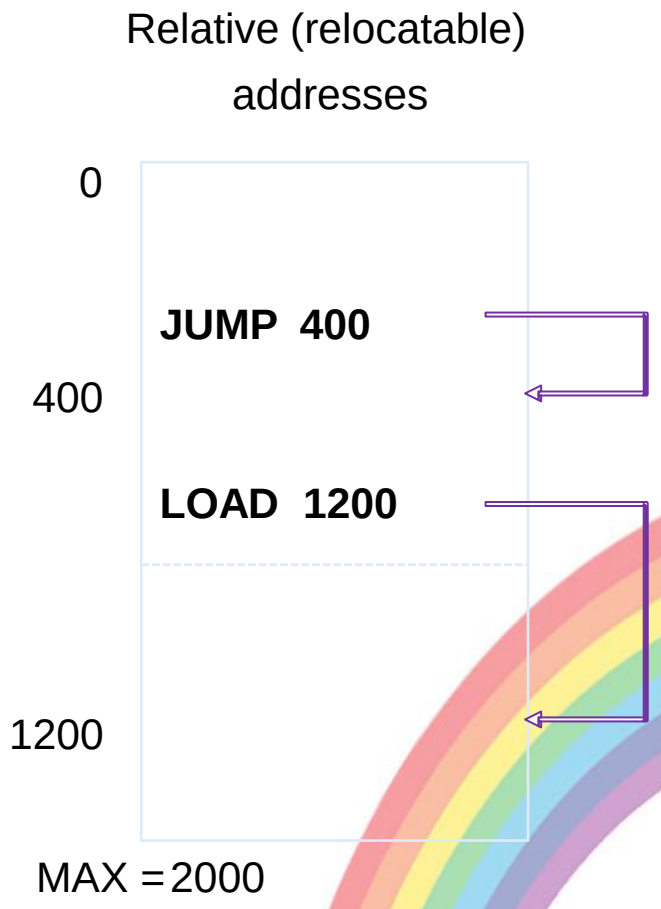
1.3. CHUYỂN ĐỔI ĐỊA CHỈ
1.3.2. SINH ĐỊA CHỈ THỰC VÀO THỜI ĐIỂM NẠP



1. BỘ NHỚ VẬT LÝ

1.3. CHUYỂN ĐỔI ĐỊA CHỈ
1.3.3. CHUYỂN ĐỔI ĐỊA CHỈ

- Execution Time: Khi trong quá trình thực thi, tiến trình 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.
- CPU tạo ra địa chỉ luận lý cho tiến trình.
- Cần sự hỗ trợ của phần cứng cho việc ánh xạ địa chỉ. Ví dụ: Trường hợp địa chỉ luận lý là Relocatable thì có thể dùng thanh ghi Base và Limit,…
- Sử dụng trong đa số các OS đa dụng (General - Purpose) trong đó có các cơ chế Swapping, Paging, Segmentation.



1. BỘ NHỚ VẬT LÝ

1.3. CHUYỂN ĐỔI ĐỊA CHỈ
1.3.4. 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);

Ví dụ trong Windows: Module ngoài là các file.dll còn trong Unix, các Module ngoài là các file.so (Shared Library);

- Load Module chứa các Stub tham chiếu (Refer) đến Routine của External Module;
- Lúc thực thi, khi Stub được thực thi lần đầu (do Process gọi Routine lần đầu), Stub nạp Routine vào bộ nhớ, tự thay thế bằng địa chỉ của Routine và Routine được thực thi;
- Các lần gọi Routine sau sẽ xảy ra bình thường;
- Stub cần sự hỗ trợ của OS (như kiểm tra xem Routine đã được nạp vào bộ nhớ chưa).

1. BỘ NHỚ VẬT LÝ

1.3. CHUYỂN ĐỔI ĐỊA CHỈ
1.3.4. DYNAMIC LINKING

ƯU ĐIỂM CỦA DYNAMIC LINKING

- Thông thường, External Module là một thư viện cung cấp các tiện ích của OS. Các chương trình thực thi có thể dùng các phiên bản khác nhau của External Module mà không cần sửa đổi, biên dịch lại.
- Chia sẻ mã (Code Sharing): Một External Module chỉ cần nạp vào bộ nhớ một lần. Các Process cần dùng External Module này thì cùng chia sẻ đoạn mã của External Module → Tiết kiệm không gian nhớ và đĩa.
- Phương pháp Dynamic Linking cần sự hỗ trợ của os trong việc kiểm tra xem một thủ tục nào đó có thể được chia sẻ giữa các tiến trình hay là phần mã của riêng một tiến trình (bởi vì chỉ có OS mới có quyền thực hiện việc kiểm tra này).

1. BỘ NHỚ VẬT LÝ

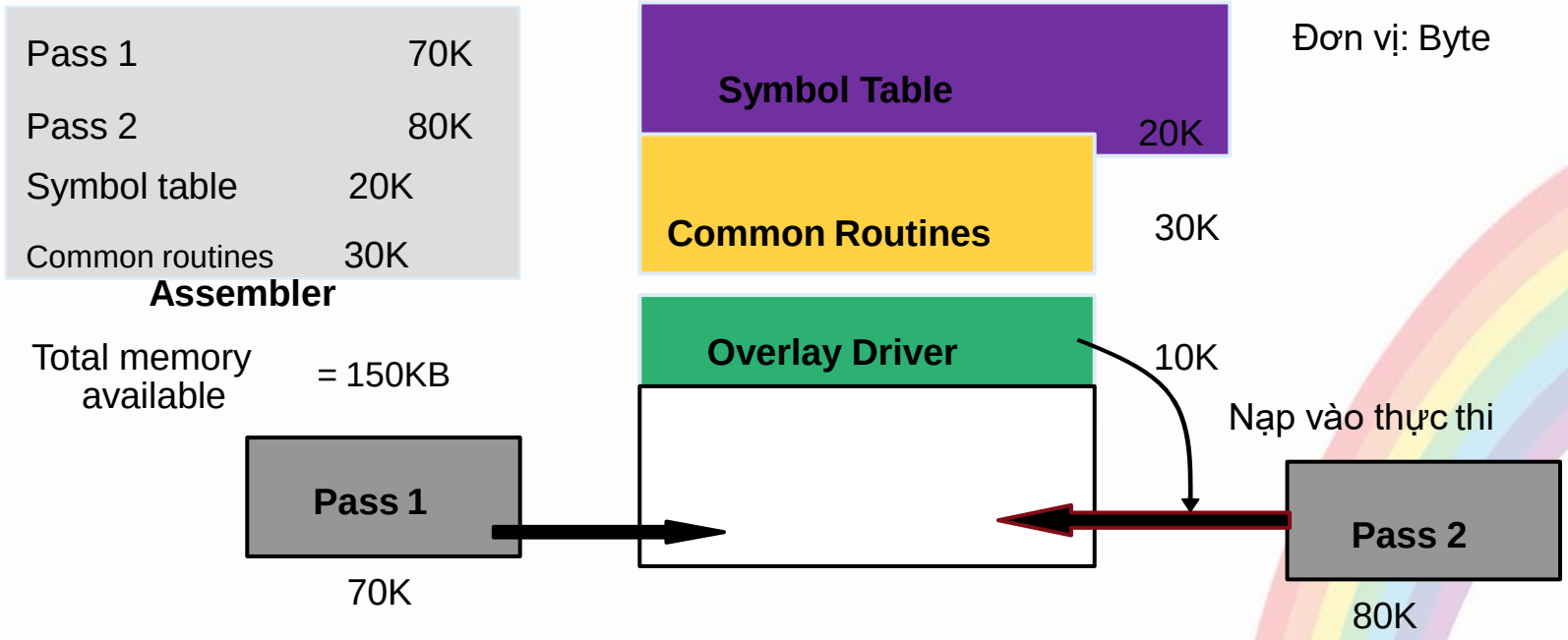
1.3. CHUYỂN ĐỔI ĐỊA CHỈ
1.3.5. DYNAMIC LOADING

- Cơ chế: Chỉ khi nào cần được gọi đến thì một thủ tục mới được nạp vào bộ nhớ chính → tăng độ hiệu dụng của bộ nhớ (Memory Utilization) bởi vì các thủ tục không được gọi đến sẽ không chiếm chỗ trong bộ nhớ.
- 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).
- Hỗ trợ từ hệ điều hành.
- Thông thường, User chịu trách nhiệm thiết kế và hiện thực các chương trình có Dynamic Loading.
- Hệ điều hành chủ yếu cung cấp một số thủ tục thư viện hỗ trợ, tạo điều kiện dễ dàng hơn cho lập trình viên.

1. BỘ NHỚ VẬT LÝ

1.4. CƠ CHẾ OVERLAY

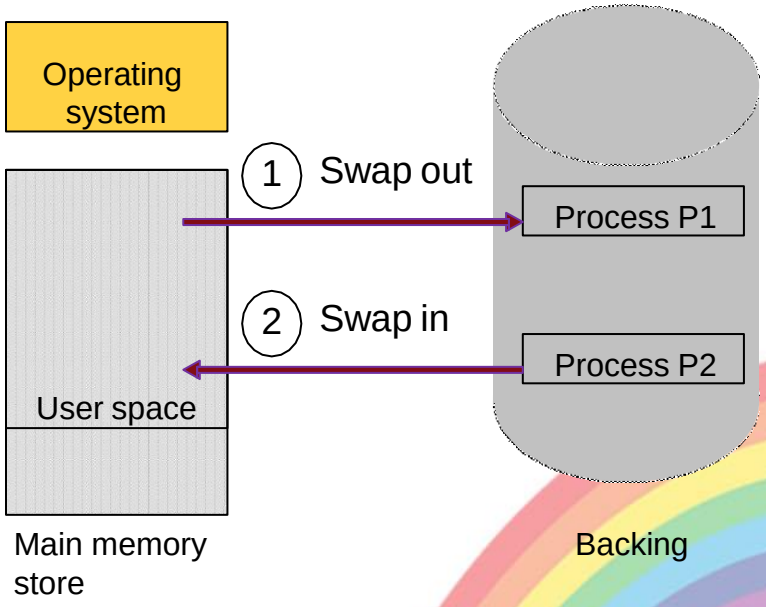
- Tại mỗi thời điểm, chỉ giữ lại trong bộ nhớ những lệnh hoặc dữ liệu cần thiết, giải phóng các lệnh/ dữ liệu chưa hoặc không cần dùng đến.
- Cơ chế này rất hữu dụng khi kích thước một tiến trình lớn hơn không gian bộ nhớ cấp cho tiến trình đó.
- Cơ chế này được điều khiển bởi người sử dụng (thông qua sự hỗ trợ của các thư viện lập trình) chứ không cần sự hỗ trợ của hệ điều hành.



1. BỘ NHỚ VẬT LÝ

1.5. CƠ CHẾ SWAPPING

- Một tiến trình có thể tạm thời bị Swap ra khỏi bộ nhớ chính và lưu trên một hệ thống lưu trữ phụ. Sau đó, tiến trình có thể được nạp lại vào bộ nhớ để tiếp tục quá trình thực thi.
- Swapping policy:
  - Round-Robin: Swap out P1 (vừa tiêu thụ hết Quantum của nó), Swap in P2, thực thi P3,…
  - Roll out, Roll in: Dùng trong cơ chế định thời theo độ ưu tiên (Priority-Based Scheduling).
- Tiến trình có độ ưu tiên thấp hơn sẽ bị Swap out nhường chỗ cho tiến trình có độ ưu tiên cao hơn mới đến được nạp vào bộ nhớ để thực thi.
- Hiện nay, ít hệ thống sử dụng cơ chế Swapping trên.



1. BỘ NHỚ VẬT LÝ

1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ

- Trong chương này, mô hình quản lý bộ nhớ là một mô hình đơn giản, không có bộ nhớ ảo;
- Một tiến trình phải được nạp hoàn toàn vào bộ nhớ thì mới được thực thi (ngoại trừ khi sử dụng cơ chế Overlay);
- 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);
- Phân đoạn đơn giản (Simple Segmentation).

1. BỘ NHỚ VẬT LÝ

1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ
1.6.1. PHÂN MẢNH (Fragmentation)

- 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 → 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 lớn hơn vùng nhớ yêu cầu. Ví dụ: Cấp một khoảng trống 18,464 bytes cho một tiến trình yêu cầu 18,462 bytes.
  - 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 tiến trình được cấp phát theo đơn vị khối.

  Ví dụ: Cơ chế phân trang (Paging).

1. BỘ NHỚ VẬT LÝ

1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ
1.6.1. PHÂN MẢNH (Fragmentation)

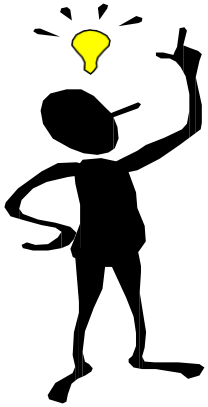
Operating system

(Used)

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 khối 18,464 bytes cho tiến trình sẽ dư ra 2 bytes không dùng!

1. BỘ NHỚ VẬT LÝ

1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ
1.6.1. PHÂN MẢNH (Fragmentation)

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.
- Tiến trình nào có kích thước nhỏ hơn hoặc bằng kích thước Partition thì có thể được nạp vào Partition đó.
- Nếu chương trình có kích thước lớn hơn Partition thì phải dùng cơ chế overlay.
- Nhận xét: Không hiệu quả do bị phân mảnh nội: Một chương trình dù lớn hay nhỏ đều được cấp phát trọn một Partition.

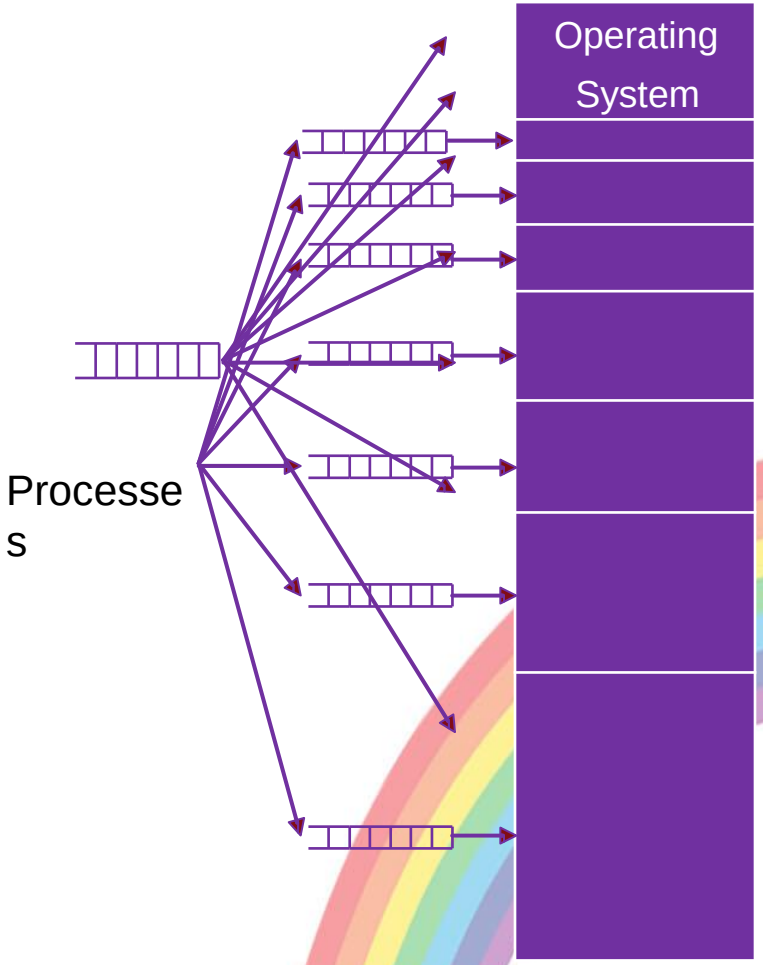
Equal
Size Partitions

Unequal
Size Partitions

1. BỘ NHỚ VẬT LÝ

1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ
1.6.2. CHIẾN LƯỢC PLACEMENT

- Partition có kích thước bằng nhau, giải pháp:
  - Nếu còn partition trống thì sẽ được nạp vào các tiến đó.
  - Nếu không còn partition nào trống nhưng trong đó có tiến trình đang ở trạng thái chờ hay đang bị Blocked thì sẽ chuyển tiến trình → ra bộ nhớ phụ, nhường chỗ cho tiếng trình mới.
- Partition có kích thước không bằng nhau, giải pháp 1:
  - Gán mỗi tiến trình 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 tiến trình với kích thước tương ứng) và hàng đợi dày đặc.

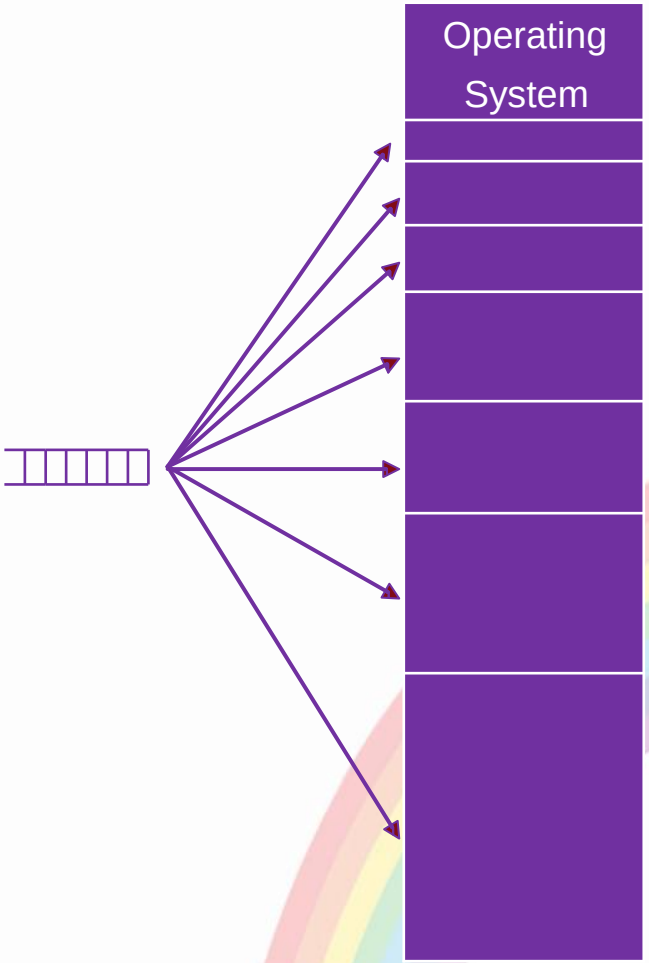


1. BỘ NHỚ VẬT LÝ

1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ
1.6.2. 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 tiến trình vào bộ nhớ chính

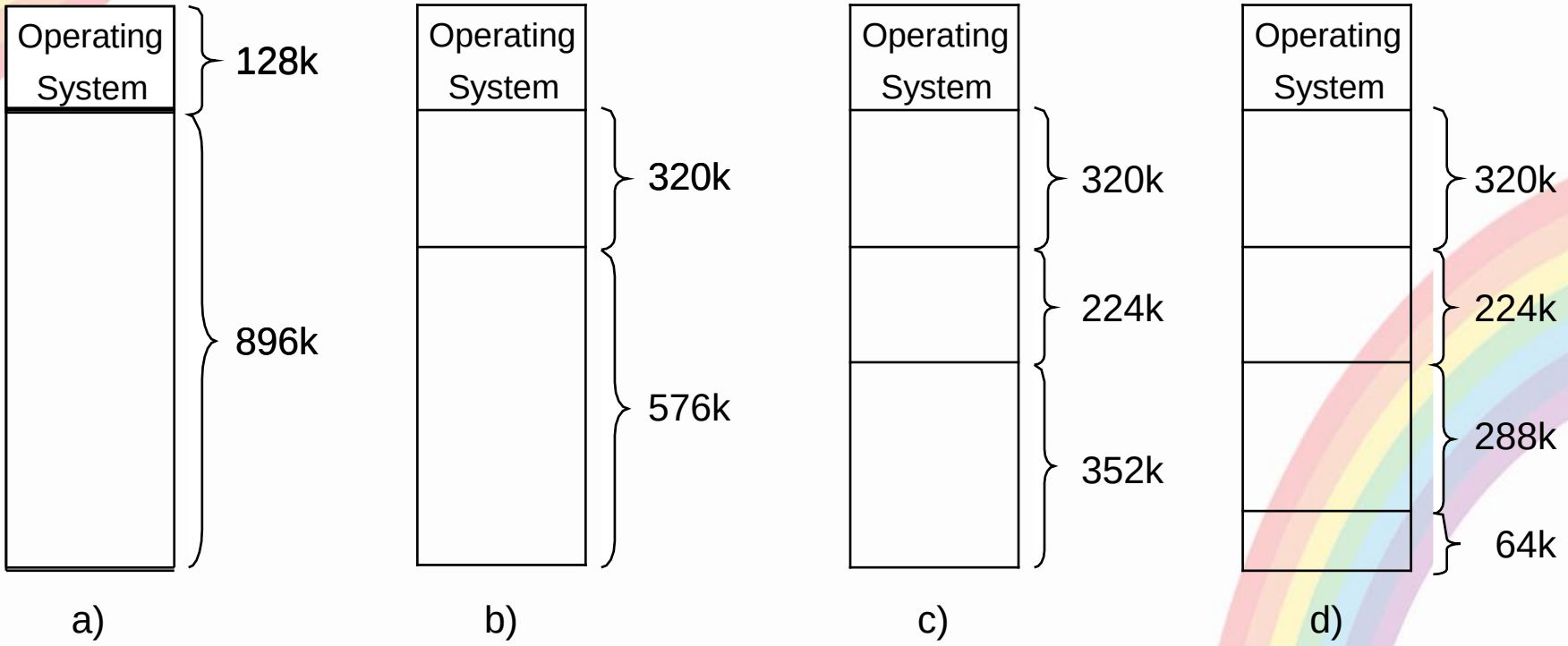
=> Chọn Partition nhỏ nhất còn trống.



1. BỘ NHỚ VẬT LÝ

1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ
1.6.3. DYNAMIC PARTITIONING

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



1. BỘ NHỚ VẬT LÝ

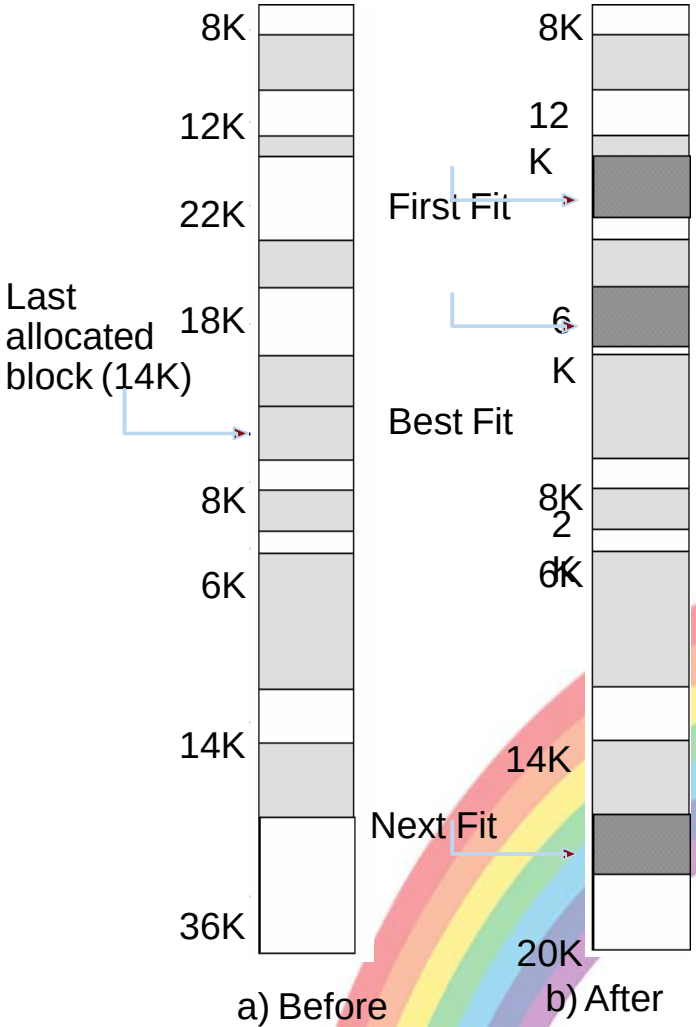
1.6. MÔ HÌNH QUẢN LÝ BỘ NHỚ
1.6.4. CHIẾN LƯỢC PLACEMENT

- Dùng để quyết định cấp phát khối bộ nhớ trống nào cho một tiến trình;
- 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.

Trong đó:

Allocated Block

Free Block



Example Memory Configuration Before and After Allocation of 16 Kbyte Block

2. TỔ CHỨC BỘ NHỚ ẢO

2. TỔ CHỨC BỘ NHỚ ẢO

2.1. Khái niệm bộ nhớ ảo;

2.2. Ánh xạ địa chỉ;

2.3. Kỹ thuật phân trang;

2.4. Vấn đề xác định kích thước trang;

2.5. Kỹ thuật phân đoạn;

2.6. Phối hợp phân trang và phân đoạn.

2. TỔ CHỨC BỘ NHỚ ẢO

2.1. KHÁI NIỆM BỘ NHỚ ẢO

- Là hình ảnh của bộ nhớ thực;
- Tách rời địa chỉ quá trình truy cập và địa chỉ trên bộ nhớ thực:
  - Địa chỉ ảo V: Tham khảo bởi tiến trình;
  - Địa chỉ thực R: Có trong bộ nhớ thực;

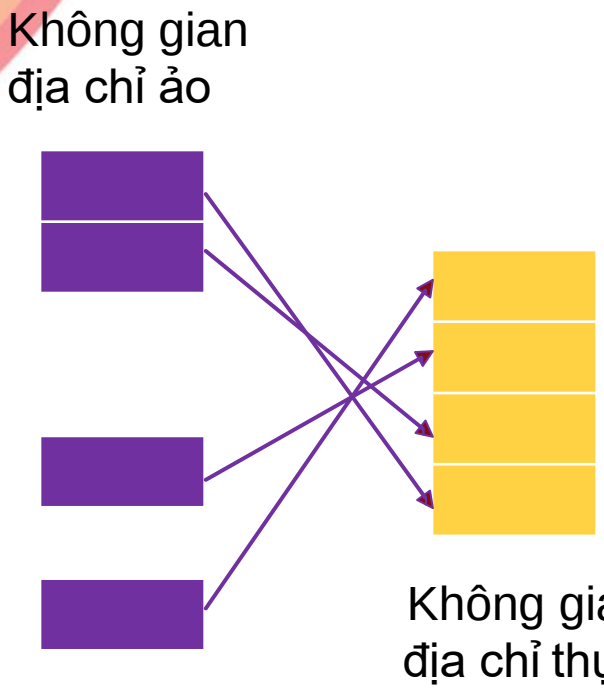
$$|V| >> |R|$$

- Địa chỉ ảo được ánh xạ thành địa chỉ thực mỗi khi quá trình thực thi → Dynamic Address Translation;
- Sự cần thiết của bộ nhớ ảo:
  - Dễ phát triển ứng dụng;
  - Lưu trữ được nhiều quá trình trong bộ nhớ;
  - Tái định vị (Relocation) các quá trình;
  - Cho các quá trình chia xẻ vùng nhớ dễ dàng.

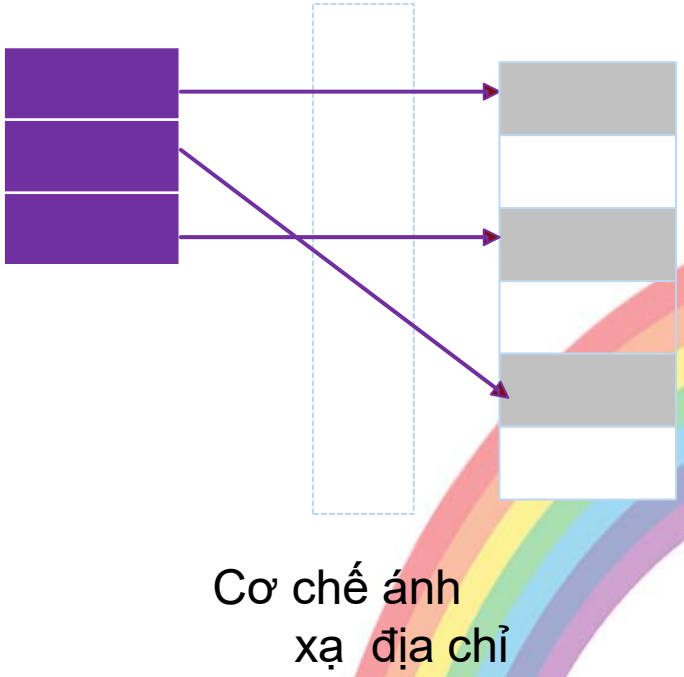
2. TỔ CHỨC BỘ NHỚ ẢO

2.2. ÁNH XẠ ĐỊA CHỈ

- Cách thực hiện: Ánh xạ khối (hình 1);
- Dùng giả lập sự liên tục của bộ nhớ (hình 2).



Hình 1

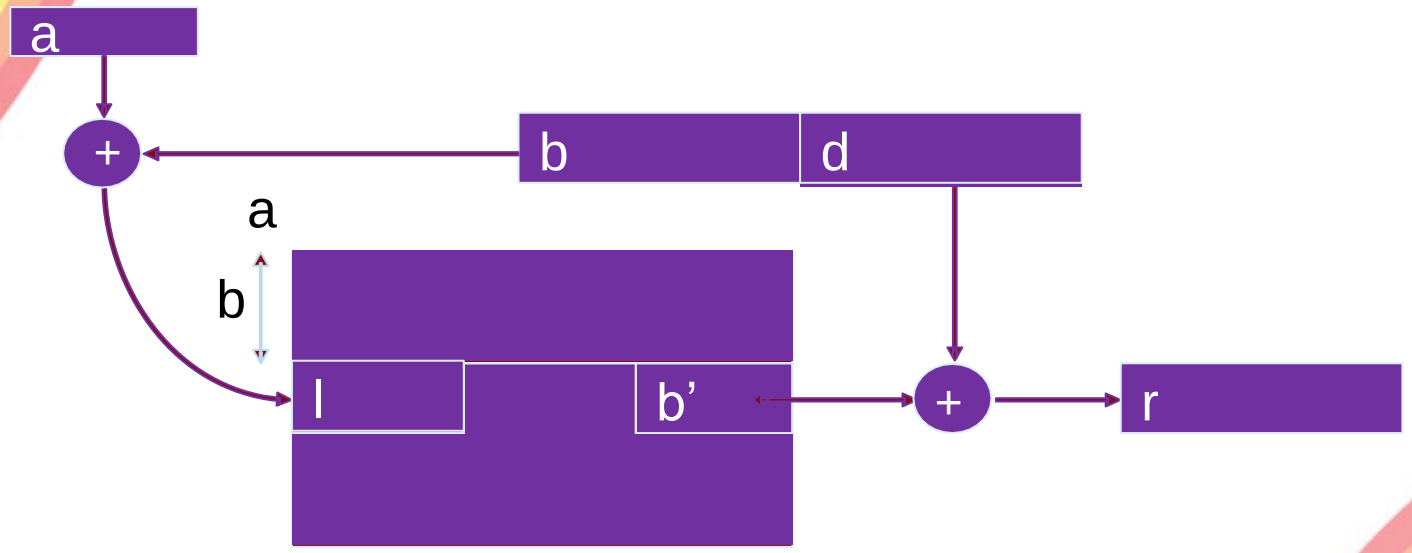


Hình 2

2. TỔ CHỨC BỘ NHỚ ẢO

2.2. ÁNH XẠ ĐỊA CHỈ

CÁCH THỰC HIỆN ÁNH XẠ KHỐI



Bảng ánh xạ khối

Trong đó:

a: Đại chỉ bảng ánh xạ khối; b: Chỉ số khối;

b': Chỉ số khối thực;

d: Độ dời trong khối; l: Bít hiện diện;

2. TỔ CHỨC BỘ NHỚ ẢO

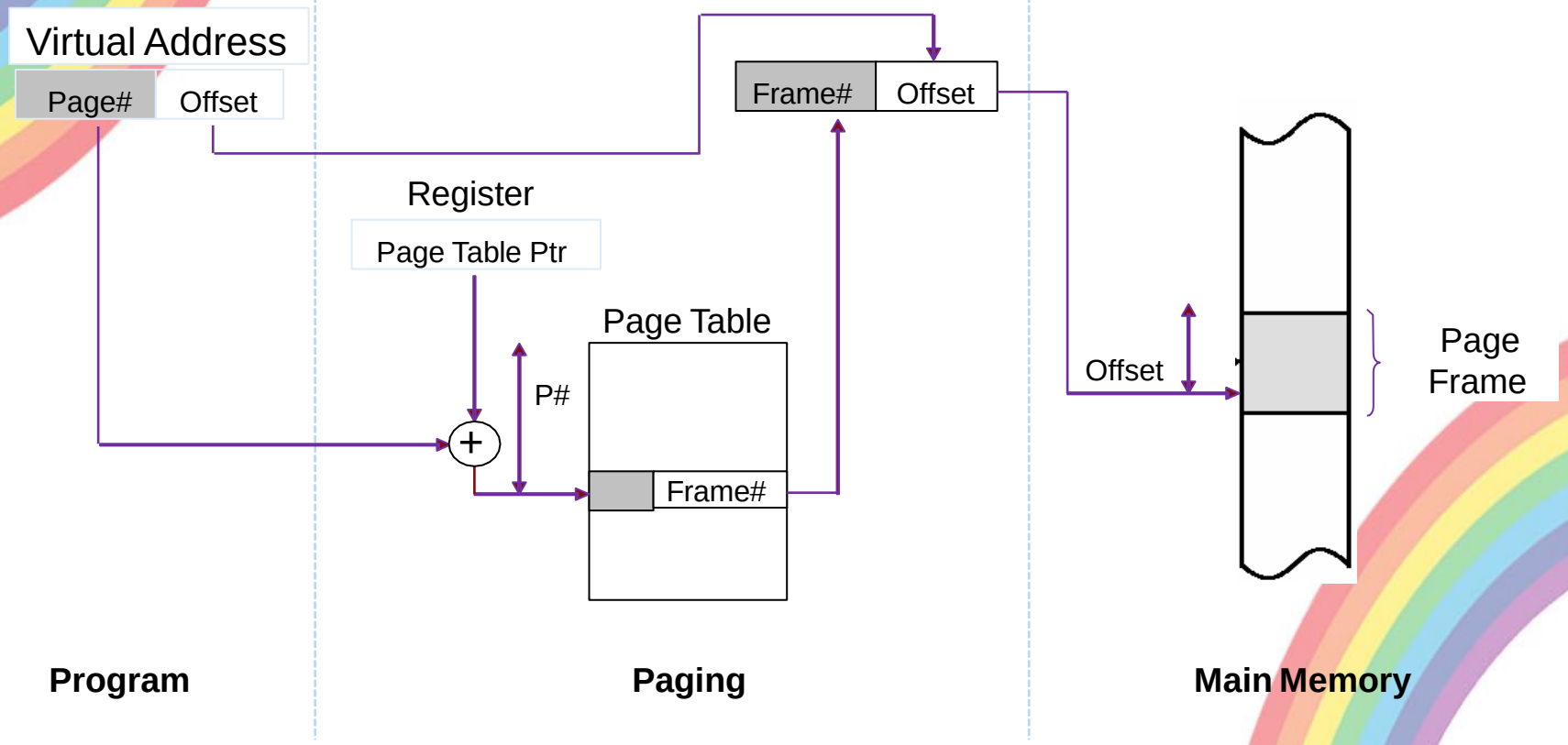
2.3. KỸ THUẬT PHÂN TRANG (Paging)

- Các khối bộ nhớ có kích thước bằng nhau:
  - Khối trên bộ nhớ ảo: Trang (Page);
  - Khối trên bộ nhớ thực: Page Frame.
- Mỗi địa chỉ ảo có hai thành phần:
  - Chỉ số trang (Page Number);
  - Độ dời của ô nhớ trong trang đó (Offset).
- Mỗi quá trình có một bảng ánh xạ trang (Page Table);
- Mỗi mục (Entry) của bảng ánh xạ trang chứa;
  - Present bit;
  - Secondary Storage Address;
  - Page Frame Number;
  - Modified bit ;
  - Các bít điều khiển khác.
- Dùng 1 Register chứa địa chỉ thực của bảng ánh xạ trang của quá trình đang chạy

2. TỔ CHỨC BỘ NHỚ ẢO

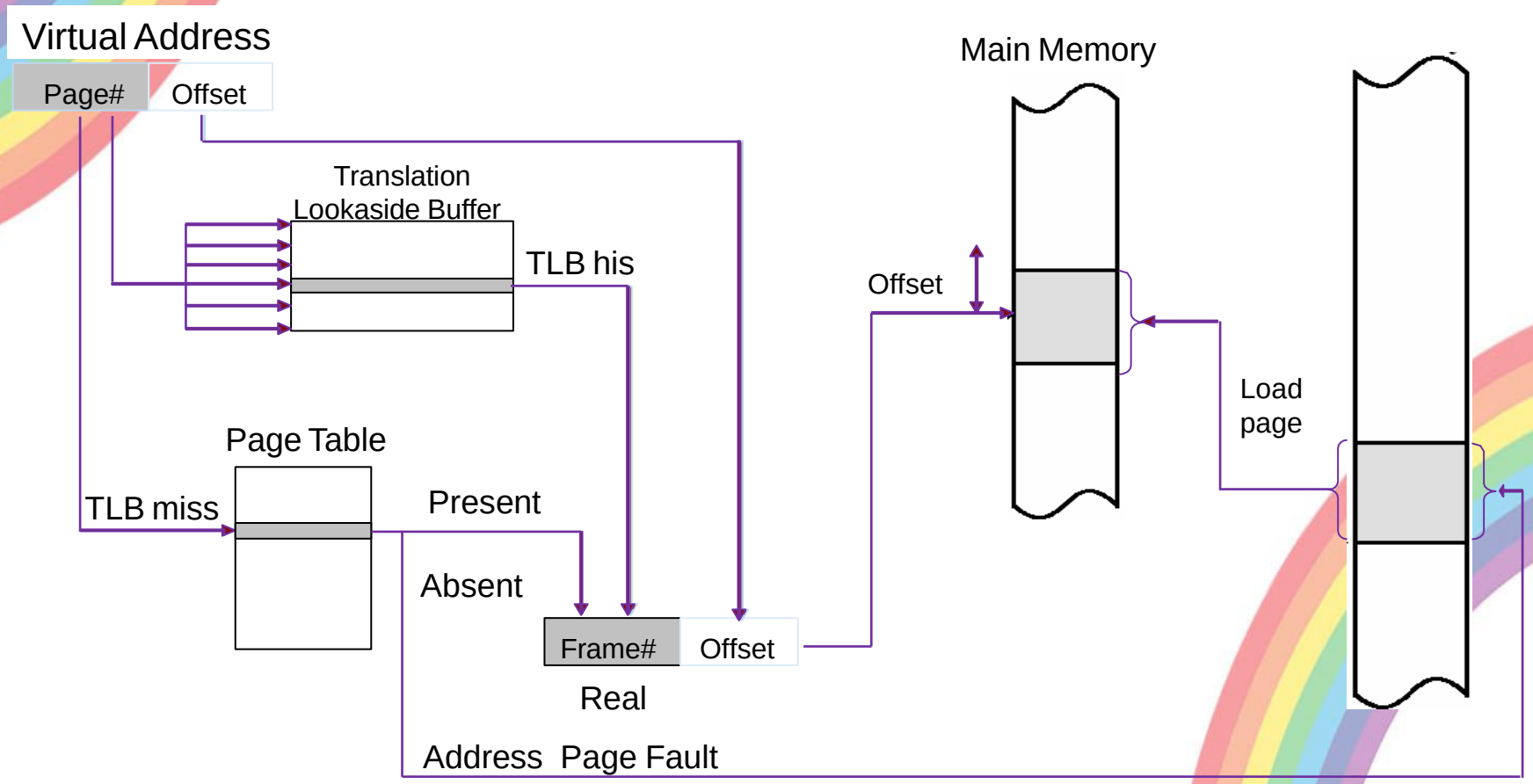
2.3. KỸ THUẬT PHÂN TRANG (Paging)

2.3.1. ÁNH XẠ ĐỊA CHỈ TRỰC TIẾP TRONG HỆ THỐNG PHÂN TRANG



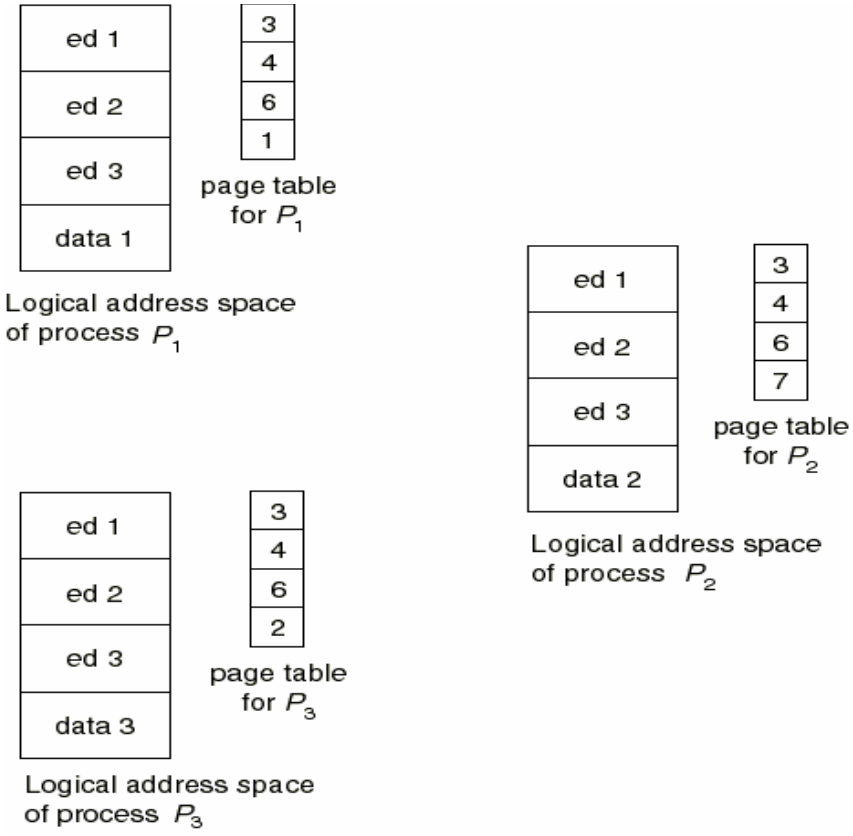
2. TỔ CHỨC BỘ NHỚ ẢO

2.3. KỸ THUẬT PHÂN TRANG (Paging)
2.3.2. ÁNH XẠ TRANG DÙNG BỘ NHỚ KẾT HỢP



2. TỔ CHỨC BỘ NHỚ ẢO

2.3. KỸ THUẬT PHÂN TRANG (Paging)
2.3.3. DÙNG CHUNG BỘ NHỚ



2. TỔ CHỨC BỘ NHỚ ẢO

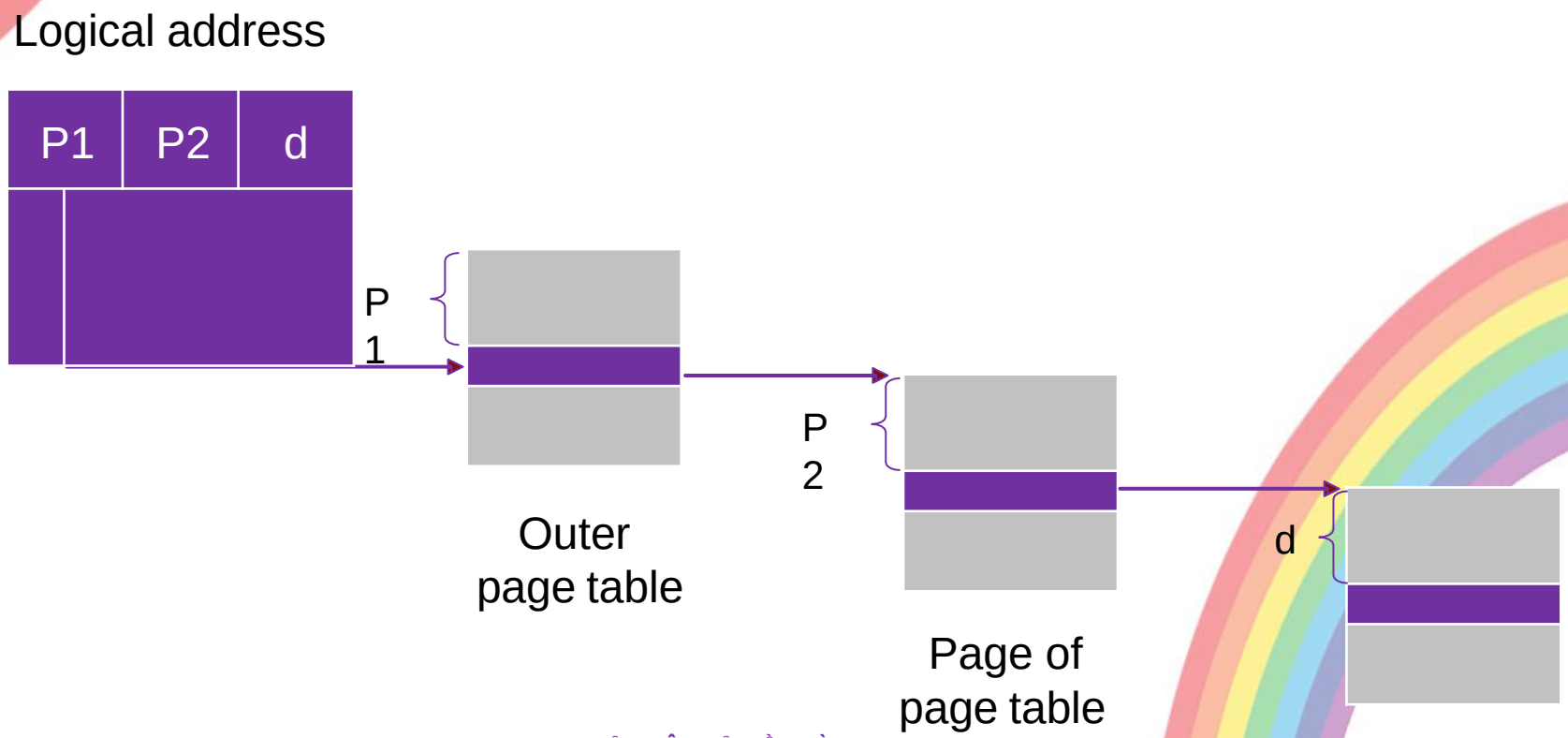
2.3. KỸ THUẬT PHÂN TRANG (Paging)
2.3.4. LƯU TRỮ BẢNG ÁNH XẠ TRANG

- Không gian địa chỉ ảo rất lớn:
  - Dùng 32 → 64 bit địa chỉ;
  - Với 32 bít địa chỉ, trang có size 4kb, bảng ánh xạ trang sẽ có 220 mục.
- Giải pháp để lưu trữ bảng ánh xạ trang của mọi quy trình:
  - Với 32 bít địa chỉ, trang có size 4kb, bảng ánh xạ trang sẽ có 220 mục.
  - Lưu trữ Page Table trong bộ nhớ ảo và phân trang nó;
  - Một số hiện thực:
    - Bảng ánh xạ trang đa cấp;
    - Bảng ánh xạ trang ngược.

2. TỔ CHỨC BỘ NHỚ ẢO

2.3. KỸ THUẬT PHÂN TRANG (Paging)
2.3.5. BẢNG ÁNH XẠ TRANG ĐA CẤP

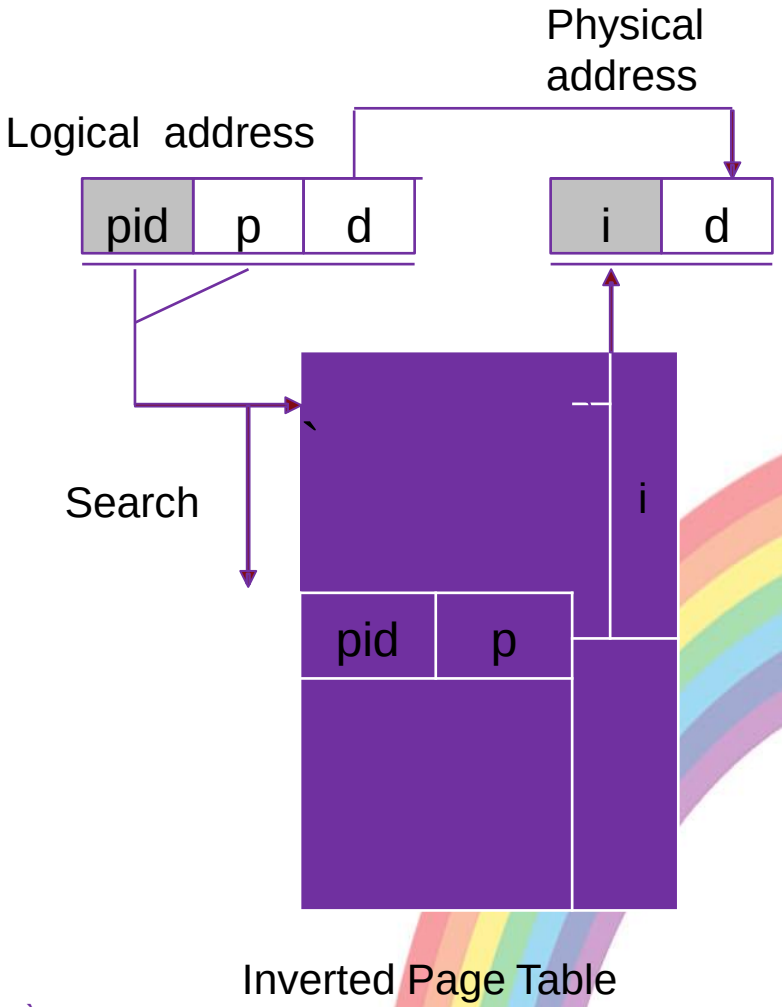
- Chỉ số trang được chia ra thành n chỉ số nhỏ;
- Ví dụ: 386, Pentium dùng n = 2;
- Chỉ số trang được chia làm 2 chỉ số P1 và P2.



2. TỔ CHỨC BỘ NHỚ ẢO

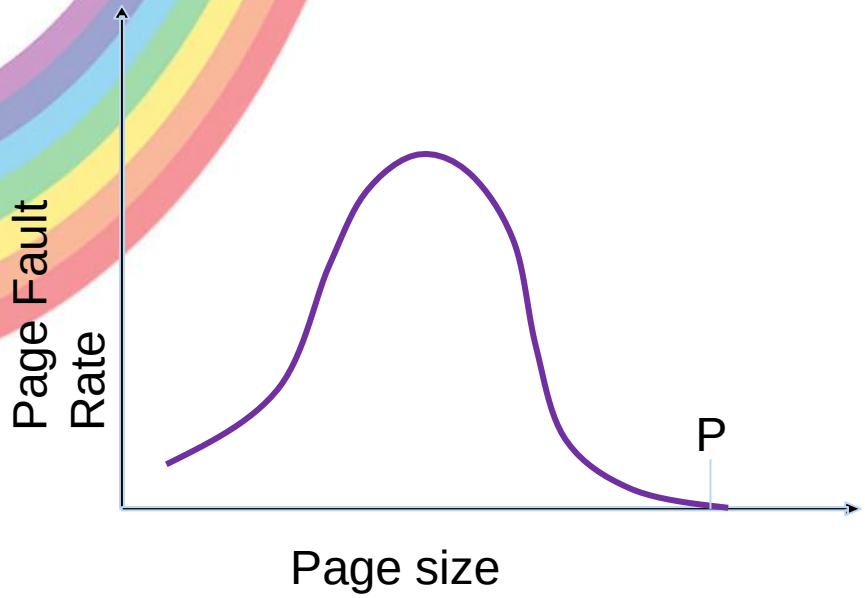
2.3. KỸ THUẬT PHÂN TRANG (Paging)
2.3.6. BẢNG ÁNH XẠ TRANG NGƯỢC (Inverted Page Table – IPT)

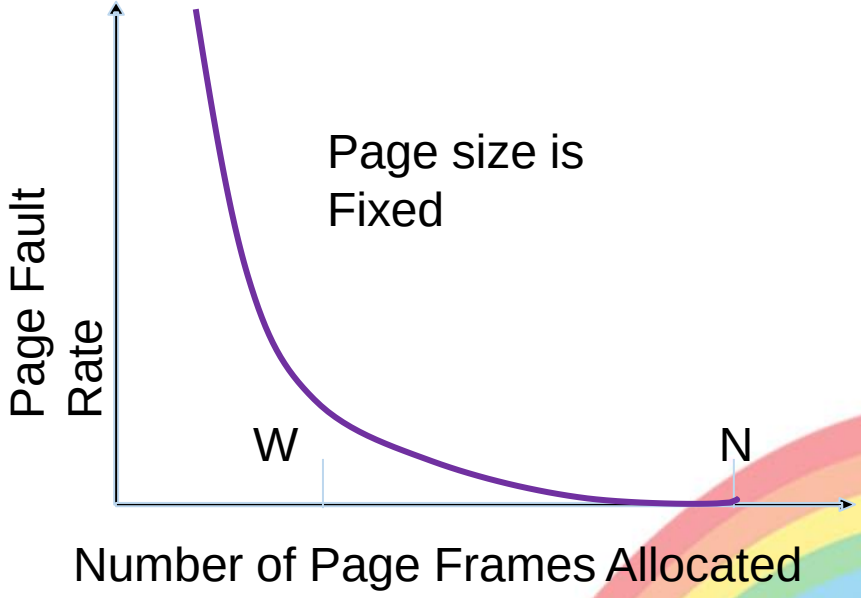
- Dùng trong PowerPC;
- Toàn hệ thống có một IPT;
- 1 mục của IPT;
- Tương ứng với 1 Frame bộ nhớ thực;
- Chứa chỉ số trang ảo được ánh xạ vào Frame đó và PID của quá trình tương ứng dùng trang ảo này;
- Dùng PID+Page# để tìm trong bảng IPT, từ đó suy ra Frame#.



2. TỔ CHỨC BỘ NHỚ ẢO

2.4. XÁC ĐỊNH KÍCH THƯỚC TRANG





- Phụ thuộc phần cứng (Size của Frame);
- Kích thước trang nên lớn hay nhỏ;
- Tỷ lệ page fault phụ thuộc vào page size và số frame cấp cho quá trình.
- Kích thước trang thông thường từ 1KB – 4 KB

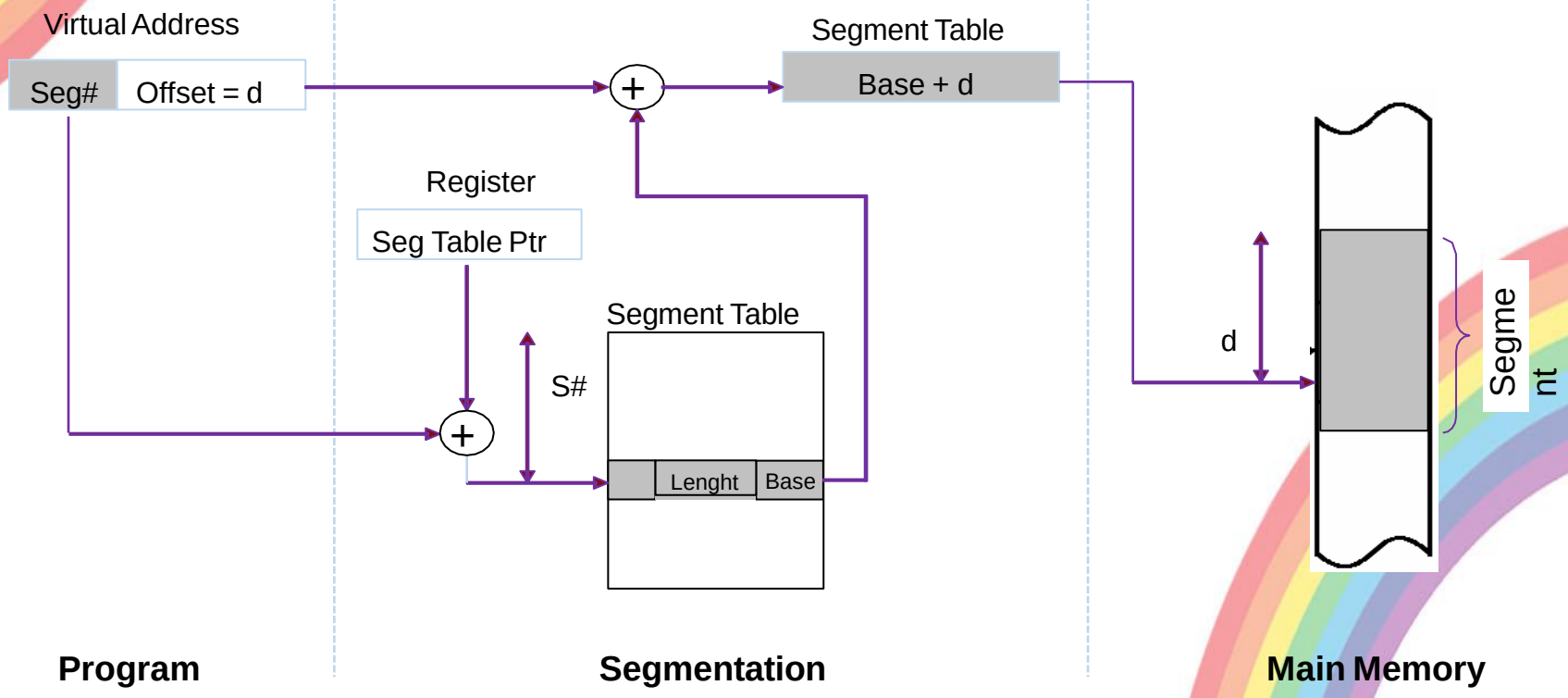
2. TỔ CHỨC BỘ NHỚ ẢO

2.5. KỸ THUẬT PHÂN ĐOẠN (Segmentation)

- Các khối bộ nhớ có kích thước khác nhau tùy thuộc yêu cầu của quá trình.
- Địa chỉ ảo:
  - Chỉ số đoạn (Segment Number);
  - Độ dời của ô nhớ trong đoạn (Displacement).
- Ưu điểm:
  - Dễ dàng mở rộng Segment, thay đổi và tái biên dịch chương trình mà không cần Link hay Load lại;
  - Cho phép chia sẻ, bảo vệ giữa các tiến trình.
- Mỗi quá trình có một bảng ánh xạ đoạn (Segment Table);
- Mỗi mục (Entry) của bảng ánh xạ đoạn chứa:
  - Present bit;
  - Secondary Storage Address;
  - Chỉ số Segment, chiều dài Segment;
  - Modified bit;
  - Các bít điều khiển khác.

2. TỔ CHỨC BỘ NHỚ ẢO

2.5. KỸ THUẬT PHÂN ĐOẠN (Segmentation)
2.5.1. ÁNH XẠ ĐỊA CHỈ TRONG HỆ THỐNG PHÂN ĐOẠN



2. TỔ CHỨC BỘ NHỚ ẢO

2.5. KỸ THUẬT PHÂN ĐOẠN (Segmentation)
2.5.2. CƠ CHẾ BẢO VỆ VÀ CHIA XẺ BỘ NHỚ TRONG HỆ THỐNG PHÂN TRANG

Có thể chia sẻ đoạn như chia sẻ trang

Mode	Read	Write	Exec
Security	N	N	N
Copy prevention	N	N	Y
Data protection	Y	N	N
Data protection	Y	N	Y
Run prevention	Y	Y	N
Full right	Y	Y	Y

2. TỔ CHỨC BỘ NHỚ ẢO

2.6. KẾT HỢP PHÂN TRANG VÀ PHÂN ĐOẠN

- Có nhiều cách kết hợp. Sau đây là một cách đơn giản, gọi là Segmentation with paging. Mỗi tiến trình sẽ có:
  - Một bảng phân đoạn;
  - Nhiều bảng phân trang: Mỗi đoạn có một bảng phân trang.
- Một địa chỉ luận lý (địa chỉ ảo) bao gồm:
  - Segment number: Là chỉ số của một mục trong bảng phân đoạn, mục này chứa địa chỉ nền (Base address) của bảng phân trang cho đoạn đó;
  - Page number: Là chỉ số của một mục trong bảng phân trang, mục này chứa chỉ số frame trong bộ nhớ thực;
  - Offset: Độ dời của vị trí nhớ trong Frame nói trên.

2. TỔ CHỨC BỘ NHỚ ẢO

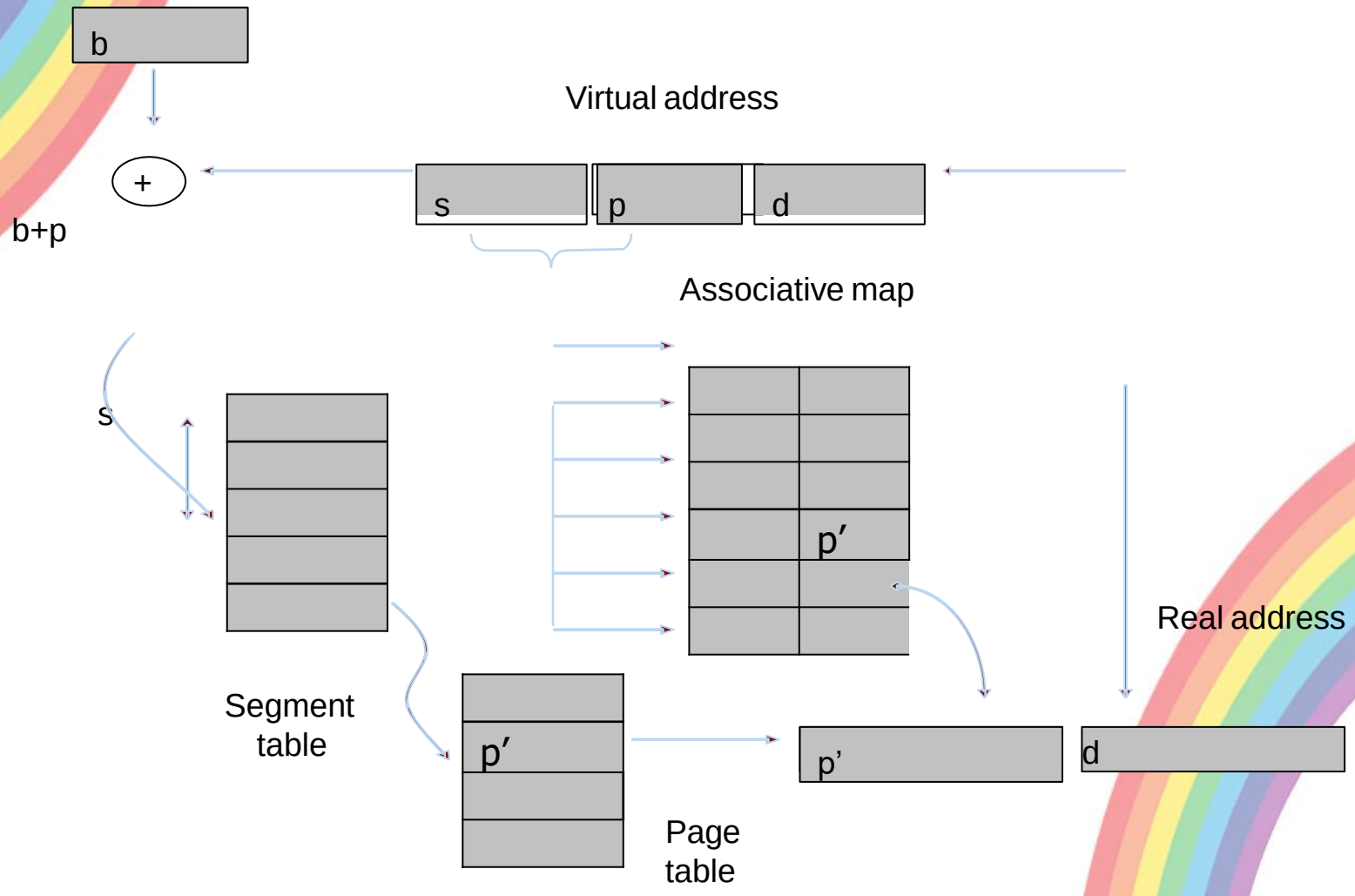
2.6. KẾT HỢP PHÂN TRANG VÀ PHÂN ĐOẠN

- Địa chỉ ảo v = (s, p, d):
  - s: Chỉ số đoạn (segment #);
  - p: Chỉ số trang trong đoạn (page #);
  - d: Độ dời offset của ô nhớ trong trang (displacement).
- Địa chỉ thực r = (p', d'):
  - p' : chỉ số trang thực (frame #);
  - d' : độ dời của ô nhớ trong trang thực.
- Ánh xạ địa chỉ:
  - (s, p) → Associate memory → p'; Hoặc s → s' (s', p) → p'.
  - s' : Địa chỉ đầu bảng ánh xạ trang.

2. TỔ CHỨC BỘ NHỚ ẢO

2.6. KẾT HỢP PHÂN TRANG VÀ PHÂN ĐOẠN

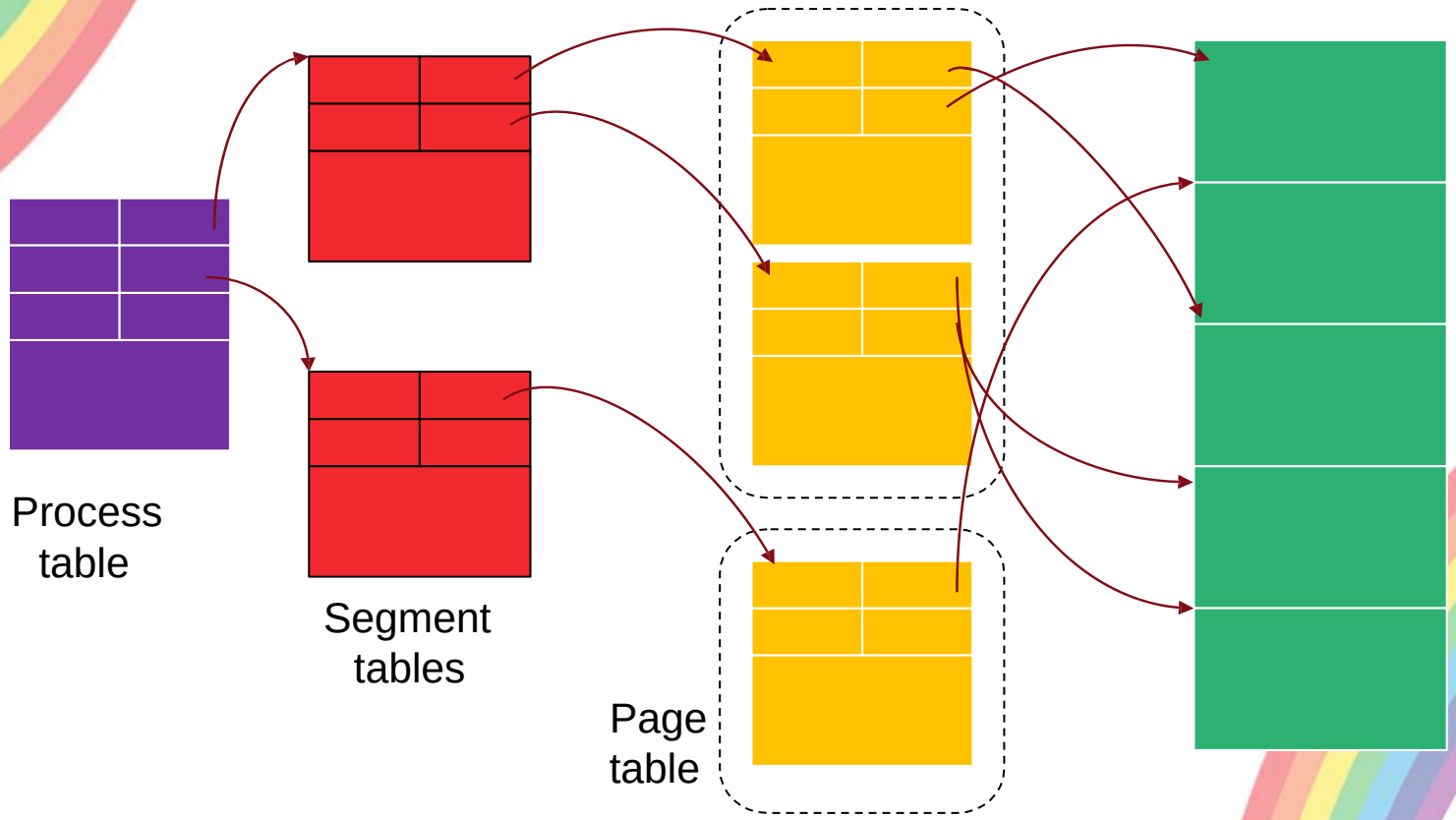
2.6.1. ÁNH XẠ ĐỊA CHỈ TRONG HỆ THỐNG PHÂN ĐOẠN KẾT HỢP PHÂN TRANG



2. TỔ CHỨC BỘ NHỚ ẢO

2.6. KẾT HỢP PHÂN TRANG VÀ PHÂN ĐOẠN

2.6.2. CẤU TRÚC ÁNH XẠ BỘ NHỚ TRONG HỆ THỐNG PHÂN ĐOẠN KẾT HỢP PHÂN TRANG



3. QUẢN LÝ BỘ NHỚ ẢO

3. QUẢN LÝ BỘ NHỚ ẢO

3.1. Các chiến lược quản lý bộ nhớ ảo;

3.2. Các giải thuật thay thế trang:

- Nguyên tắc tối ưu;
- Các giải thuật: OPT, FIFO, LRU, LFU, NUR, dịp may thứ hai.

3.3. Tính cục bộ (Locality);

3.4. Lý thuyết về tập làm việc (Working Set).

3.1. CÁC CHIẾN LƯỢC QUẢN LÝ BỘ NHỚ ẢO

- Các chiến lược quản lý:
  - Chiến lược nạp (Fetch strategies);
  - Chiến lược sắp đặt (Placement strategies);
  - Chiến lược thay thế(Replacement strategies);
- Chiến lược nạp:
  - Nạp trang theo yêu cầu (Demand paging);
  - Nạp trang tiên đoán (Anticipatory paging);
  - Page fault và các bước xử lý Page fault.
- Chiến lược sắp đặt;
- Chiến lược thay thế.

3. QUẢN LÝ BỘ NHỚ ẢO

3.2. CÁC GIẢI THUẬT THAY THẾ TRANG

- Yêu cầu: Tối thiểu số Page fault;
- Nguyên tắc tối ưu: Chọn trang thay thế là:
  - Trang không còn dùng nữa;
  - 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;
- Một số giải thuật thay thế trang:
  - Thay thế trang ngẫu nhiên;
  - FIFO, LRU, giải thuật xấp xỉ LRU, LFU, NUR.

3. QUẢN LÝ BỘ NHỚ ẢO

3.2. CÁC GIẢI THUẬT THAY THẾ TRANG

3.2.1. GIẢI THUẬT TỐI ƯU (OPT)

Chọn trang thay thế là trang sẽ không được tham khảo trong thời gian lâu nhất.

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

Thời điểm	0	1	2	3	4	5	6	7	8	9	10	11

Bộ nhớ thực có 3 Frame

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

3. QUẢN LÝ BỘ NHỚ ẢO

3.2. CÁC GIẢI THUẬT THAY THẾ TRANG
3.2.2. GIẢI THUẬT FIFO

- Chọn trang thay thế là trang ở trong bộ nhớ thực trong khoảng thời gian lâu nhất;
- Nghịch lý belady.

	1	2	3	4	1	2	5	1	2	3	4	5
Bộ nhớ thực có 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

Bộ nhớ thực có 4 frame	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

10 page fault

3. QUẢN LÝ BỘ NHỚ ẢO

3.2. CÁC GIẢI THUẬT THAY THẾ TRANG

3.2.3. GIẢI THUẬT LRU (Least Recently Used)

Chọn trang thay thế là trang đã không được tham khảo trong thời gian lâu nhất.

Thời điểm t	0	1	2	3	4	5	6	7	8	9	10	11
Chuỗi tham khảo	1	2	3	4	1	2	5	1	2	3	4	5

Bộ nhớ thực có 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

3. QUẢN LÝ BỘ NHỚ ẢO

3.2. CÁC GIẢI THUẬT THAY THẾ TRANG
3.2.4. GIẢI THUẬT NUR (Not Used Recently)

R	M	Ý nghĩa đối với trang nhớ
0	0	Chưa tham chiếu, chưa sửa đổi.
0	1	Chưa tham chiếu, đã sửa đổi.
1	0	Đã tham chiếu, chưa sửa đổi.
1	1	Đã tham chiếu, đã sửa đổi.

Thứ tự ưu tiên thay thế trang giảm dần

3. QUẢN LÝ BỘ NHỚ ẢO

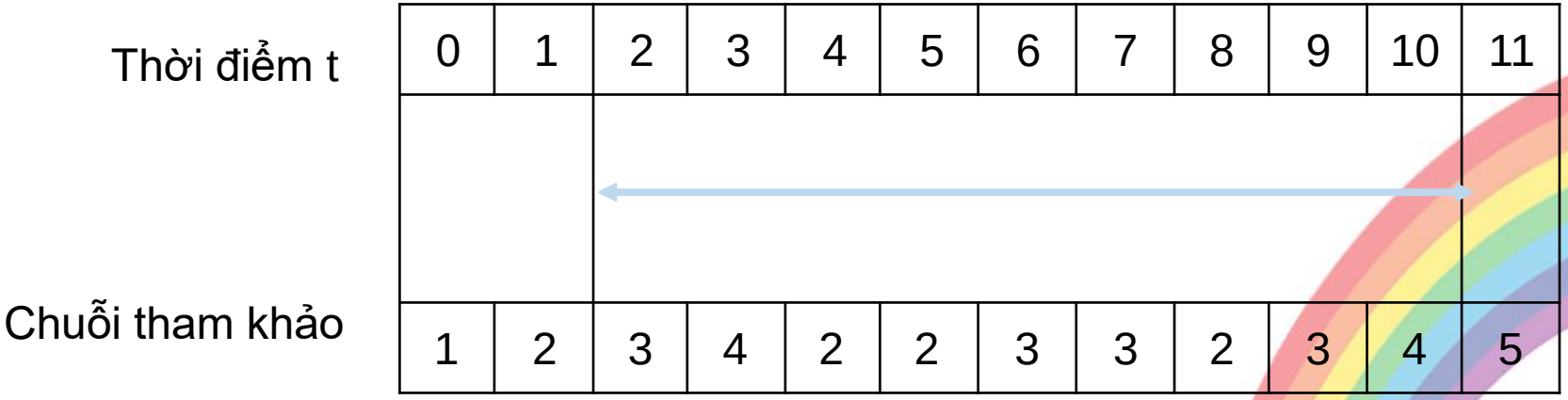
3.2. CÁC GIẢI THUẬT THAY THẾ TRANG
3.2.5. DỊP MAY THỨ HAI (SECOND CHANCE)

- Là giải thuật xấp xỉ LRU;
- Còn gọi là giải thuật FIFO cải tiến;
- Mỗi trang có 1 bit tham chiếu R, lúc đầu là 0;
- Trang được chọn xét thay thế theo kiểu FIFO:
  - Trang có R = 0 sẽ được thay thế ngay;
  - Trang có R = 1 được đưa vào cuối hàng và đặt lại r = 0. hệ thống chọn lựa các trang còn lại trong hàng đợi.

3. QUẢN LÝ BỘ NHỚ ẢO

3.2. CÁC GIẢI THUẬT THAY THẾ TRANG
3.2.6. GIẢI THUẬT LFU (Least Frequently Used)

- Là giải thuật xấp xỉ LRU;
- Chọn trang thay thế là trang có tần số 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.

3. QUẢN LÝ BỘ NHỚ ẢO

3.3. LÝ THUYẾT VỀ TÍNH CỤC BỘ (Locality)

- Tính cục bộ về thời gian (Temporal Locality):
  - Các sự việc xảy ra ở thời điểm t rất có thể đang 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 rất có thể đang xảy ra ở các vùng lân cận;
  - Ví dụ: Những vùng nhớ đang được tham khảo gần đây thường kề nhau.
- Ý nghĩa:
  - Trong lập trình;
  - Trong OS: Giải thuật thay thế trang.

3. QUẢN LÝ BỘ NHỚ ẢO

3.3. LÝ THUYẾT VỀ TÍNH CỤC BỘ (Locality)
3.3.1. KỸ THUẬT ĐỆM TRANG (Page Buffering)

- Tạm thời giữ lại các trang được chọn để thay thể → tránh tác động của giải thuật thay thế trang kém hiệu quả. Sử dụng 2 danh sách:
  - Free Page List;
  - Modified Page List.
- Khi có Page Fault, hệ thống tìm xem trang cần nạp có còn trong bộ nhớ không trước khi nạp trang:
  - Trang nạp sẽ nạp vào đầu Free Page List;
  - Modified Page List dùng để ghi các trang ra theo từng cụm nhiều trang
    - → giảm chi phí I/O.

3. QUẢN LÝ BỘ NHỚ ẢO

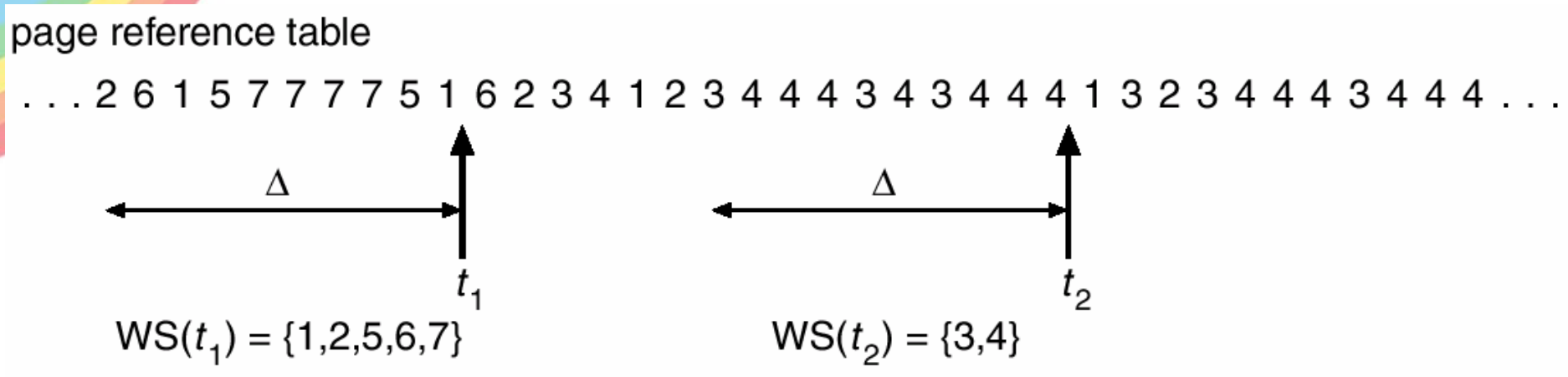
3.3. LÝ THUYẾT VỀ TÍNH CỤC BỘ (Locality)
3.3.2. CÁC VẤN ĐỀ KHÁC

- Tầm vực thay thế trang (Resident Scope):
  - Tầm vực cục bộ: Chỉ chọn trang thay thế trong những trang của quá trình liên quan;
  - Tầm vực toàn cục: Chọn bất kỳ trang nào không bị lock để thay thế.
- Số Frame cấp cho quá trình(Resident Set Size):
  - Không đổi (Fixed Allocation): Chia đều/ theo tỉ lệ kích thước quá trình;
  - Thay đổi trong quá trình chạy (Variable Allocation).
- Điều khiển tải (Load Control): Số quá trình cần nạp vào bộ nhớ?

3. QUẢN LÝ BỘ NHỚ ẢO

3.4. LÝ THUYẾT VỀ TẬP LÀM VIỆC

Tập làm việc (Working Set - WS) = tập những trang quá trình cần sử dụng để làm việc trong thời gian Δ (hình vẽ):



- Lý tưởng: WS của quá trình nằm hoàn toàn trong bộ nhớ chính.
- Theo dõi working set của các quá trình như thế nào?

TÓM LƯỢC CUỐI BÀI

- Nắm được khái niệm bộ nhớ vật lý;
- Nắm được cách tổ chức và điều khiển bộ nhớ thực;
- Nắm được khái niệm bộ nhớ ảo, cách tổ chức và cơ chế chuyển địa chỉ ảo sang địa chỉ thực;
- Nắm được các chiến lược quản lý bộ nhớ ảo.