[Câu 1: Trình bày kỹ thuật phân trang và bộ nhớ ảo. Khảo sát và cho biết giá trị của các thông số quản lý (số bit quản lý địa chỉ ô nhớ, kích thước trang, số bit tối thiểu để quản lý các offset trong trang, số khung trang vật lý, số khung trang logic tối đa trên không gian tiến trình, kích thước phần bộ nhớ ảo,…) trên một hệ thống máy tính cụ thể (và diễn giải) 1](#_Toc93061479)

[1.1 Kỹ thuật phân trang 1](#_Toc93061480)

[**1.1.1 Chuyển đổi địa chỉ trong paging** 3](#_Toc93061481)

[**1.1.2 Bảng phân trang( page table)** 4](#_Toc93061482)

[**1.1.3 Phân loại bảng phân trang** 8](#_Toc93061483)

[Bảng phân trang 2 mức 8](#_Toc93061484)

[Bảng phân trang đa mức 11](#_Toc93061485)

[Bảng phân trang băm 12](#_Toc93061486)

[Chia sẻ các trang nhớ 13](#_Toc93061487)

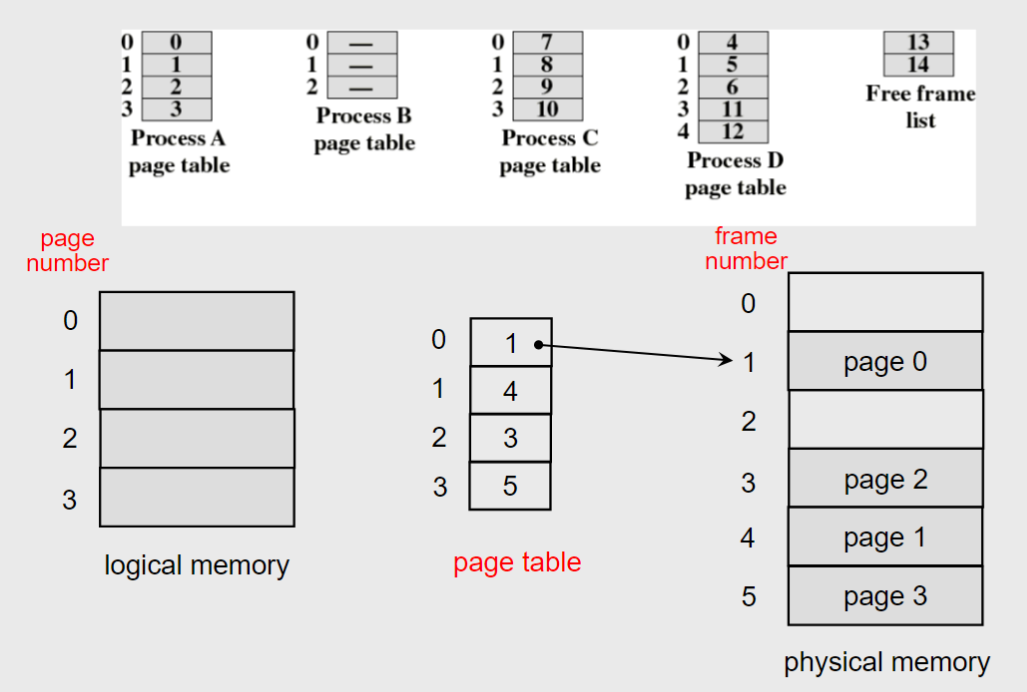
[**1.2 Bộ nhớ ảo( virtual memory)** 13](#_Toc93061488)

[**1.2 Khảo sát laptop:** 14](#_Toc93061489)

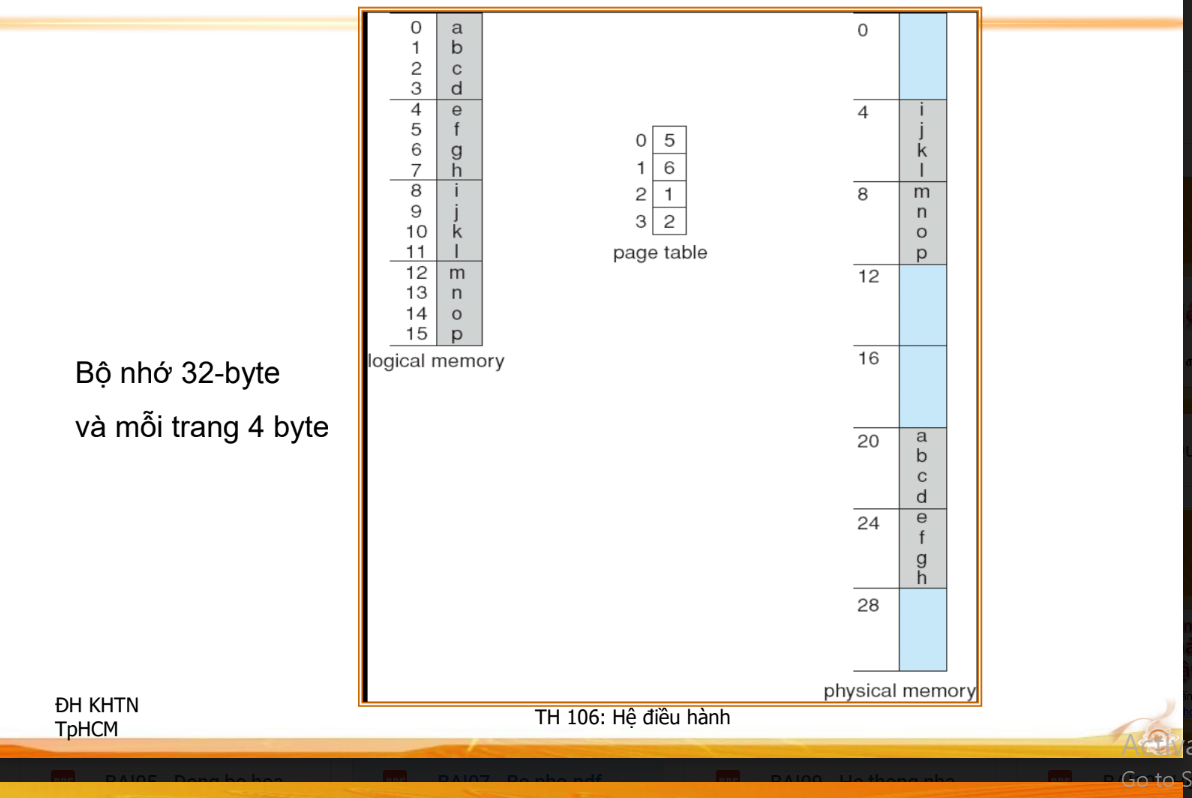
# Câu 1: Trình bày kỹ thuật phân trang và bộ nhớ ảo. Khảo sát và cho biết giá trị của các thông số quản lý (số bit quản lý địa chỉ ô nhớ, kích thước trang, số bit tối thiểu để quản lý các offset trong trang, số khung trang vật lý, số khung trang logic tối đa trên không gian tiến trình, kích thước phần bộ nhớ ảo,…) trên một hệ thống máy tính cụ thể (và diễn giải)

## 1.1 Kỹ thuật phân trang

* Phân trang(paging) là một chương trình quản lý bộ nhớ . Phân trang cho phép một quá trình được lưu trữ trong **không gian quản lý địa chỉ**( physical address space) hay gọi là **bộ nhớ** theo cách không liền kề . Quá trình lưu trữ theo cách không tiếp giáp giải quyết vấn đề phân mảnh bên ngoài .
* Bộ nhớ thực được chia thành các khối cố định và có kích thước bằng nhau gọi là frame.
* Thông thường kích thước của frame là lũy thừa của 2, từ 512 byte đến 16MB
* Bộ nhớ luận lý( logical memory) hay không gian địa chỉ luận lý là tập mọi địa chỉ luận lý của quá trình
* Địa chỉ luận lý có thể được quá trình sinh ra bằng cách dùng indexing, base register, segment register,....
* Bộ nhớ luận lý cũng được chia thành các khối cố định có cùng kích thước gọi là trang nhớ( page)
* Frame và trang nhớ có kích thước bằng nhau
* Hệ điều hành phải thiết lập 1 bảng phân trang( page table) để ánh xạ địa chỉ luận lý thành đại chỉ thực
* Mỗi process được cấp phát 1 bảng phân trang
* Thiết lập bảng phân trang cho process là 1 phần của chuyển ngữ cảnh
* Kỹ thuật phân trang khiến bộ nhớ bị phân mảnh nội nhưng khắc phục được phân mảnh ngoại.
* Hình mô tả:

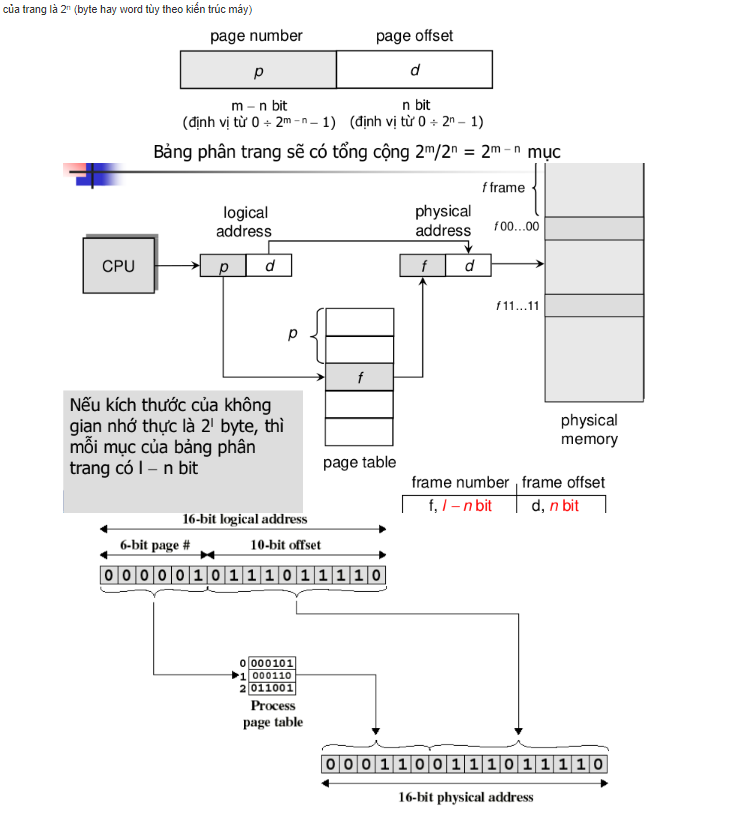


Ví dụ cụ thể:



### **1.1.1 Chuyển đổi địa chỉ trong paging**

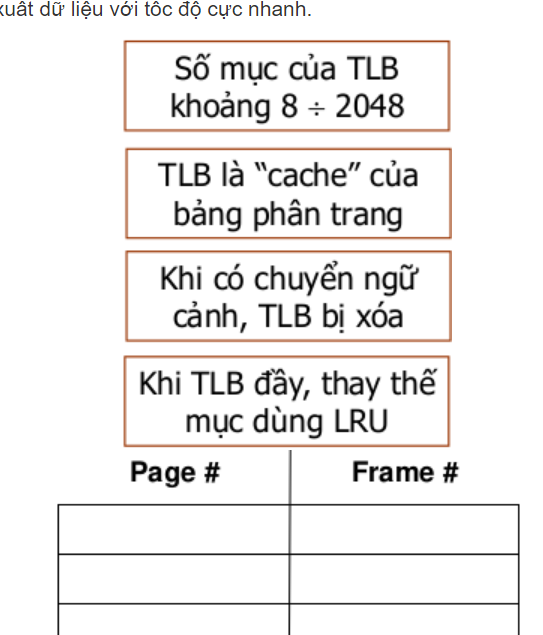
* Địa chỉ luận lý gồm có:  
  + Page number , p , được dùng làm chỉ mục vào bảng phân trang. Mỗi mục (entry) trong bảng phân trang chứa chỉ số frame (còn gọi là số frame cho gọn) của trang tương ứng trong bộ nhớ thực.
  + Page offset , d , được kết hợp với địa chỉ nền (base address) của frame để định vị địa chỉ thực.
* Nếu kích thước của không gian địa chỉ ảo là 2m , và kích thước của trang là 2n (byte hay word tùy theo kiến trúc máy)



### **1.1.2 Bảng phân trang( page table)**

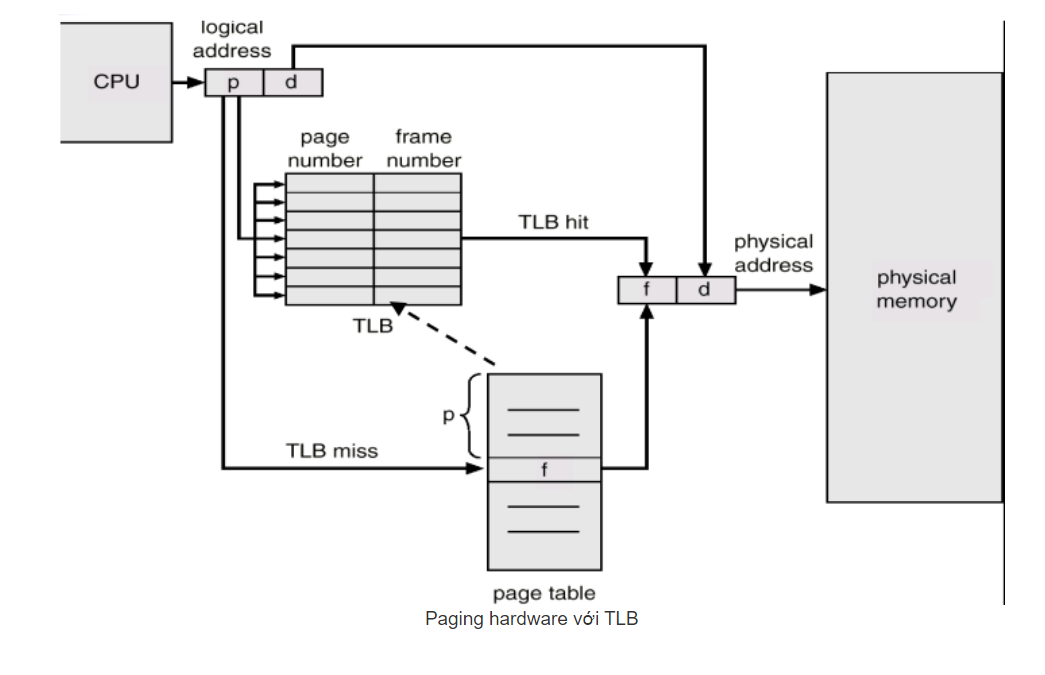
* Bảng phân trang thường được lưu giữ trong bộ nhớ chính  
  + Mỗi process được hệ điều hành cấp một bảng phân trang
  + Thanh ghi page-table base (PTBR) trỏ đến bảng phân trang
  + Thanh ghi page-table length (PTLR) biểu thị kích thước của bảng phân trang (có thể được dùng trong cơ chế bảo vệ bộ nhớ)
* Mỗi truy cập dữ liệu/lệnh cần hai thao tác truy xuất vùng nhớ  
  + Dùng page number p làm index để truy cập mục trong bảng phân trang nhằm lấy số frame, và kế đến là dùng page offset d để truy xuất dữ liệu/lệnh trong frame
* Thường dùng một cache phần cứng có tốc độ truy xuất và tìm kiếm cao, gọi là thanh ghi kết hợp (associative register) hoặc translation look-aside buffers (TLBs)

TLB, là thanh ghi hỗ trợ tìm kiếm truy xuất dữ liệu với tốc độ cực nhanh.



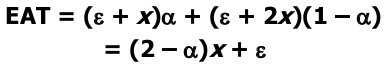
**Ánh xạ page #**

* Nếu page number nằm trong TLB (: hit, trúng) lấy ngay được số frame tiết kiệm được thời gian truy cập bộ nhớ chính để lấy số frame trong bảng phân trang.
* Ngược lại (: miss, trật), phải lấy số frame từ bảng phân trang như bình thường.

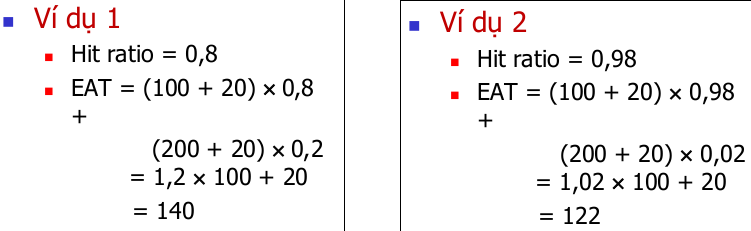


**Effective access time (EAT)**

* Tính thời gian truy xuất hiệu dụng (EAT)
* Thời gian tìm kiếm trong TLB (associative lookup): ε
* Thời gian một chu kỳ truy xuất bộ nhớ: x
* Hit ratio: tỉ số giữa số lần chỉ số trang được tìm thấy (hit) trong TLB và số lần truy xuất khởi nguồn từ CPU,
  + Kí hiệu hit ratio: α
* Thời gian cần thiết để có được địa chỉ thực  
  + Khi chỉ số trang có trong TLB (hit) : ε + x
  + Khi chỉ số trang không có trong TLB (miss): ε + x + x
* Thời gian truy xuất hiệu dụng

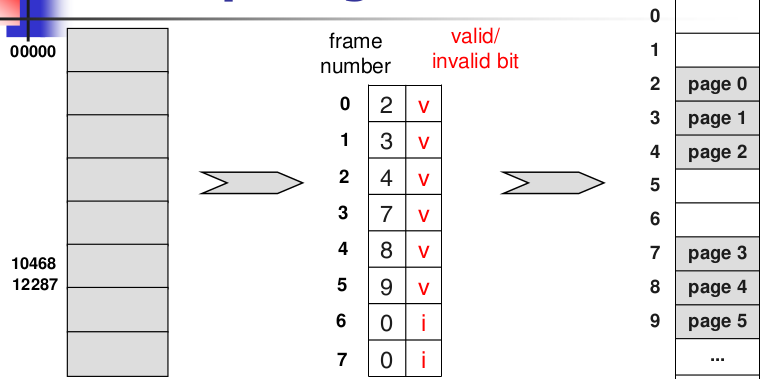


* Giả sử (đơn vị thời gian: nano giây)  
  + Associative lookup = 20
  + Memory access = 100



**Bảo vệ bộ nhớ**

* Việc bảo vệ bộ nhớ được hiện thực bằng cách gắn với frame các bit bảo vệ (protection bit) được giữ trong bảng phân trang. Các bit này biểu thị các thuộc tính sau  
  + read-only, read-write, execute-only
* Ngoài ra, còn có một valid/invalid bit gắn với mỗi mục trong bảng phân trang  
  + “valid”: cho biết là trang của process, do đó là một trang hợp lệ.
  + “invalid”: cho biết là trang không của process, do đó là một trang bất hợp lệ.



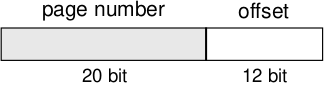
* Mỗi trang nhớ có kích thước 2K = 2048
* Process có kích thước 10.468 ⇒ phân mảnh nội ở frame 9 (chứa page 5), các địa chỉ ảo > 12287 là các địa chỉ invalid.
* Dùng PTLR để kiểm tra truy xuất đến bảng phân trang có nằm trong bảng hay không.

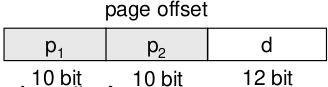
### **1.1.3 Phân loại bảng phân trang**

#### Bảng phân trang 2 mức

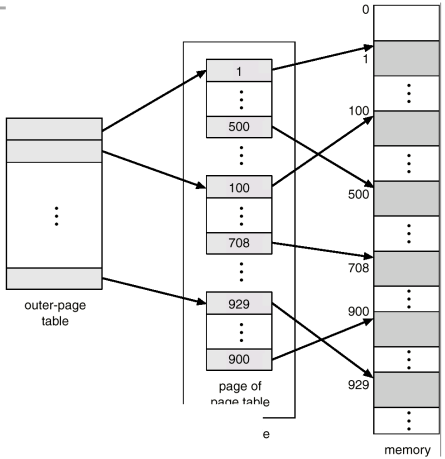
* Các hệ thống hiện đại đều hỗ trợ không gian địa chỉ ảo rất lớn (2^32 đến 2^64 ), ở đây giả sử là 2^32  
  + Giả sử kích thước trang nhớ là 4 KB (= 2^12 ) ⇒ bảng phân trang sẽ có 2^32 / 2^12 = 2^20 = 1 M mục.
  + Giả sử mỗi mục gồm 4 byte thì mỗi process cần 4 MB cho bảng phân trang
* Một giải pháp là, thay vì dùng một bảng phân trang duy nhất cho mỗi process, “paging” bảng phân trang này, và chỉ giữ những bảng phân trang cần thiết trong bộ nhớ ⇒ bảng phân trang 2 mức (two-level page table).

Ví dụ

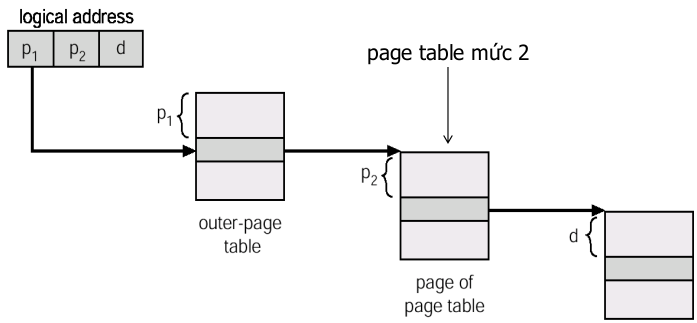
* Một địa chỉ luận lý trên hệ thống 32-bit với trang nhớ 4K được chia thành các phần sau:
  + Page number: 20 bit, Nếu mỗi mục dài 4 byte ⇒ Cần 220 4 byte = 4 MB cho mỗi page table
* Bảng phân trang cũng được chia nhỏ nên page number cũng được chia nhỏ thành 2 phần:  
  + 10-bit page number
  + 10-bit page offset
*   
  Vì vậy, một địa chỉ luận lý sẽ như hình vẽ bên  
  + p 1 : chỉ số của mục trong bảng phân trang mức 1 (outer-page table)
  + p 2 : chỉ số của mục trong bảng phân trang mức 2

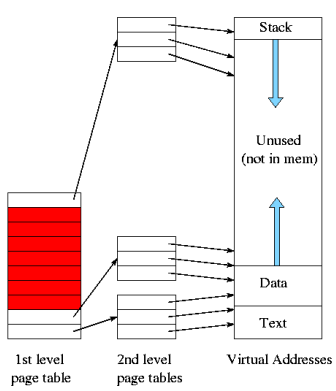


* Có 2 p1 mục trong bảng phân trang mức 1
* Mỗi bảng phân trang mức 2 chứa 2 p2 mục

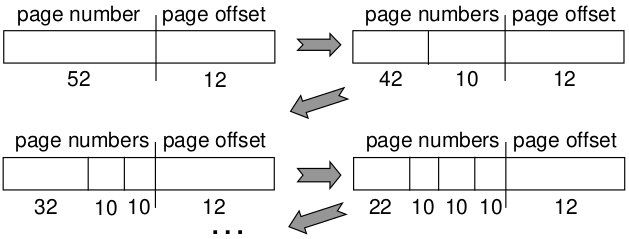


các bảng phân trang mức 2

Sơ đồ chuyển đổi địa chỉ (address-translation scheme) cho kỹ thuật phân trang 2 mức, với 32- bit địa chỉ

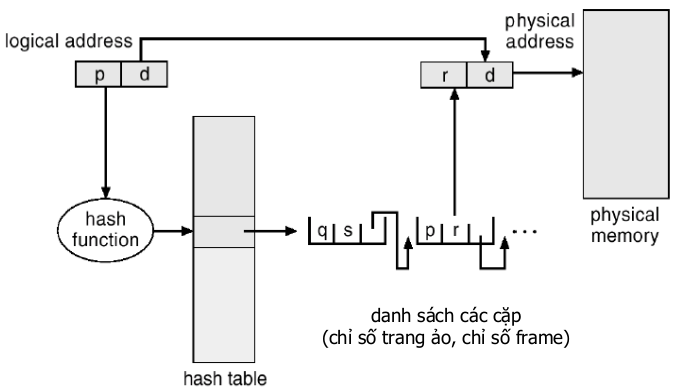
* Bảng phân trang 2 mức giúp tiết kiệm bộ nhớ: Vùng màu đỏ tương ứng với phần chưa được sử dụng của không gian địa chỉ ảo. Các mục màu đỏ được đánh dấu là không có frame nên sẽ gây ra page fault nếu được tham chiếu đến; khi đó OS sẽ tạo thêm bảng phân trang mức 2 mới.  
  

#### Bảng phân trang đa mức

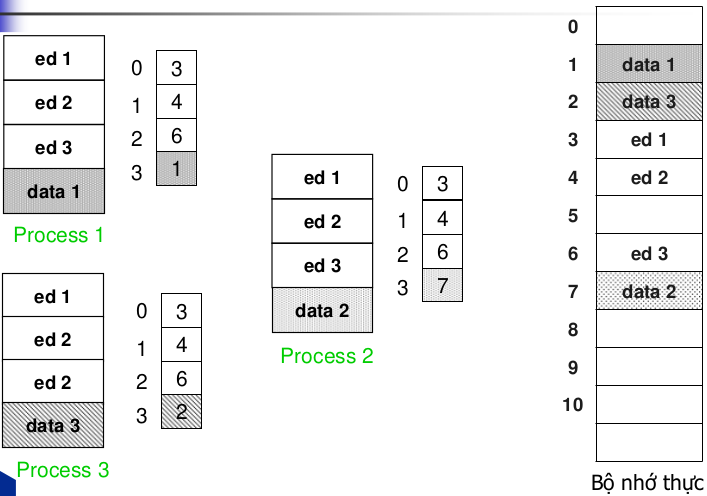
* Ví dụ: Không gian địa chỉ luận lý 64-bit với trang nhớ 4K. Bảng phân trang 2-mức vẫn còn quá lớn! Tương tự bảng phân trang 2 mức, ta có bảng phân trang 3, 4,…, n mức  
  
* Tiết kiệm chổ trong bộ nhớ chính bằng cách chỉ giữ trong bộ nhớ chính các bảng phân trang mà process hiện đang cần.

#### Bảng phân trang băm

* Dùng kỹ thuật băm để giảm không gian bảng phân trang. Phổ biến trong các hệ thống có địa chỉ lớn hơn 32 bit.
* Để giải quyết đụng độ, mỗi mục của bảng băm được gắn một danh sách liên kết. Mỗi phần tử của danh sách là một cặp (chỉ số trang ảo, chỉ số frame). Chỉ số trang ảo (virtual page number) được biến đổi qua hàm băm thành một hashed value . Cặp (chỉ số trang ảo, chỉ số frame) sẽ được lưu vào danh sách liên kết tại mục có chỉ số là hashed value.
* Giải thuật tìm trang: Chỉ số trang ảo được biến đổi thành hashed value. Hashed value là chỉ số của mục cần truy cập trong bảng băm. Sau đó, tìm trong danh sách liên kết phần tử chứa chỉ số trang ảo để trích ra được chỉ số frame tương ứng.



#### **Chia sẻ các trang nhớ**

****

## **1.2 Bộ nhớ ảo( virtual memory)**

* Bộ nhớ ảo là sự trừu tượng hóa của HĐH, nó cung cấp người lập

trình một không gian địa chỉ lớn hơn không gian địa chỉ vật lý

thật sự

* Bộ nhớ ảo có thể được triển khai bằng cách phân trang hoặc

phân đoạn, hiện tại phân trang thông dụng hơn

* Mô hình kết hợp cũng thường được dùng, phân đoạn thường khá

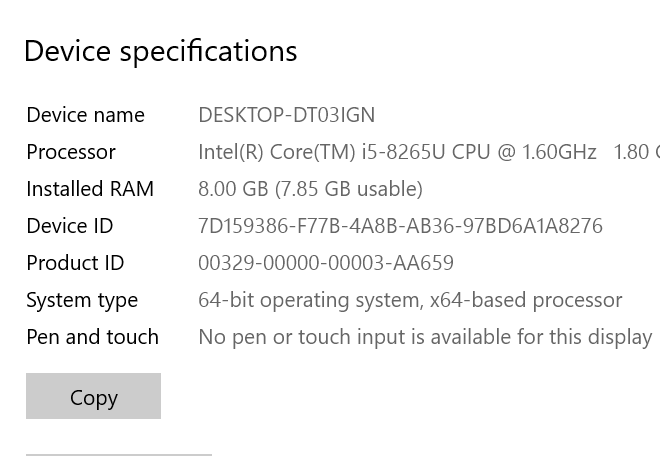
đơn giản (v.d., một số lượng xác định các đoạn cùng kích thước)

* Ưu điểm của bộ nhớ ảo:
* Lập trình viên không lo lắng với việc các máy tính khác nhau có

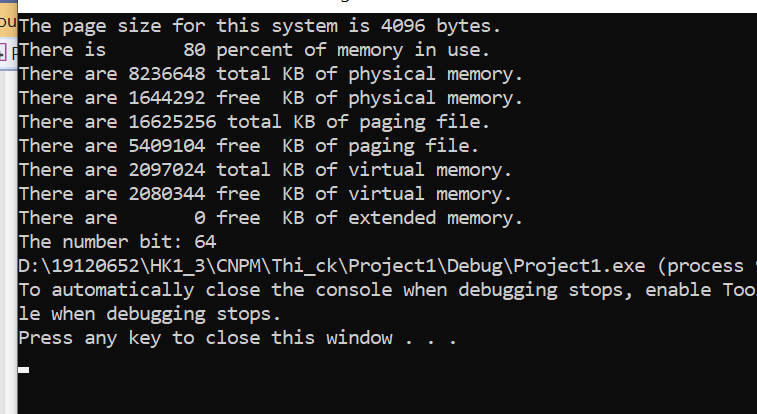
kích thước bộ nhớ vật lý khác nhau

* Phân mảnh trong môi trường đa chương
* Cho phép nhiều ứng dụng hơn chạy cùng lúc.
* Cho phép chạy các ứng dụng lớn hơn trong các hệ thống không có đủ RAM vật lý để chạy chúng.
* Cung cấp cách tăng bộ nhớ ít tốn kém hơn so với việc mua thêm RAM vật lý. Tuy nhiên, hiện tại giá RAM vật lý đã giảm rất nhiều so với trước đó.
* Cung cấp cách tăng bộ nhớ trong một hệ thống có mức dung lượng RAM tối đa mà phần cứng và hệ điều hành có thể hỗ trợ.
* Nhược điểm:
* Không mang lại được hiệu suất tương tự như RAM vật lý. Chậm hơn rất nhiều kể cả SSD, càng chậm nếu là HDD.
* Có thể ảnh hưởng tiêu cực đến hiệu suất tổng thể của một hệ thống.
* Chiếm một phần dung lượng lưu trữ, mà vốn được sử dụng để lưu trữ dữ liệu lâu dài.

## **1.2 Khảo sát laptop:**



* Hệ thống quản lý địa chỉ trên hệ thống: 64 bit (ở system type x64). Là 32 bit nếu x84
* Số bit quản lý địa chỉ ô nhớ: 48 bit
* Hệ thống 64 bit có không gian địa chỉ lý thuyết là 2 ^ 64 = 16 Exabyte.
* Điều này nằm ngoài phạm vi của phần cứng hệ thống có sẵn trên thị trường cho đến thời điểm hiện tại. Vì vậy, các hệ thống 64 bit được thu nhỏ để hoạt động trong 48 bit với không gian địa chỉ là 2 ^ 48 = 256 TB.



* Sử dụng C++ để xem các thông tin trên qua:
* kiểu MEMORYSTATUSEX. Ví dụ MEMORYSTATUSEX test;
* hàm GetSystemInfo(SYSTEM\_INFO);
* Kích thước trang(page size or page table): 4KB = 4096 bytes
* Bộ nhớ ảo đã được cấu hình 8GB
* physical memory là RAM
* Chúng ta sử dụng paging file để cấu hình bộ nhớ ảo nhưng dung lượng nó = RAM + Bộ nhớ ảo
* Bộ nhớ ảo = paging file - physical memory = 16625256 - 8236648 = 8388608 KB = 8 GB.
* Số khung trang vật lý:
* (virtual memory size) / (page size) = 8236608/4 = 2 097 152 trang
* page size = 4096 bytes = 2^12 =>Số bit tối thiểu để quản lý các offset trong trang(page offset) = 12 bit => page number = 48 – 12 = 36 bit
* Số khung trang logic vật lý trên không gian tiến trình: 2^48/4KB = 2^48/2^12 = 2^36 trang

Tài liệu tham khảo

1. Hệ điều hành kỹ thuật phân trang: Web ZUN.vn

link: <http://www.zun.vn/tai-lieu/he-dieu-hanh-ky-thuat-phan-trang-42393/>

1. Sự khác biệt giữa phân trang và phân đoạn trong hệ điều hành. Web: Gadfet-info.com

link: <https://vi.gadget-info.com/difference-between-paging>

1. Kỹ thuật phân mảnh bộ nhớ. WEB: cuuduongthancong

link:<https://cuuduongthancong.com/atc/1144/ky-thuat-phan-manh-bo-nho-(he-dieu-hanh)>

1. Vì sao hệ thống có 64 bit mà chỉ sử dụng 48 bit?

link:<https://qastack.vn/programming/6716946/why-do-x86-64-systems-have-only-a-48-bit-virtual-address-space>

1. Bài tập bộ nhớ: hướng dẫn tính số bit offset, số khung vật lý, khung logic

link:<https://hutechos.weebly.com/uploads/1/6/9/3/16936172/bt-bonho.pdf>

1. Cấu trúc của SYSTEM\_INFO

link:<https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/ns-sysinfoapi-system_info>

1. Code kiểu MEMORYSTATUSEX

link:<https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-globalmemorystatusex?fbclid=IwAR3VT7S7hkCcwVUfxWu_VaPGbTb4Kb6r8LBS5XNozbE15IqiUwRRcvcbp2A>