TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

Môn : Nguyên lý hệ điều hành

- ĐỀ TÀI -

Nghiên cứu tìm hiểu về quản lí bộ nhớ trong trong hệ điều hành Windows

GVHD : Nguyễn Bá Nghiễn

Sinh viên thực hiện : Nhóm 5

1. Phạm Minh Đức	2020600469
2. Phạm Trí Đức	2020600025
3. Nguyễn Tiến Dũng	2020600278
4. Nguyễn Văn Minh	2020601161
5. Nguyễn Vi Phượng	2020601137

Hà Nội - 2022

MỤC LỤC

DANH MỤC HÌNH ẢNH

LỜI NÓI ĐẦU

Bộ nhớ là thiết bị lưu trữ duy nhất mà thông qua đó CPU có thể trao đổi thông tin với môi trường bên ngoài, nó là tài nguyên quan trọng thứ 2 sau CPU trong. Do đó nhu cầu quản lí bộ nhớ là nhiệm vụ hàng đầu của HĐH. Bộ nhớ được tổ chức như mảng một chiều bao gồm các byte hoặc các từ nhớ (word) được đánh địa chỉ. Đây là chỗ chứa các tiến trình và dữ liệu của tiến trình. Việc quản lí và sử dụng bộ nhớ hợp lí ảnh hưởng tới tốc độ và khả năng của toàn bộ hệ thống tính toán.

Các công việc liên quan tới quản lí bộ nhớ bao gồm quản lí bộ nhớ trống, cấp phát bộ nhớ trống cho các tiến trình và giải phóng bộ nhớ đã cấp phát, ngăn chặn việc truy xuất trái phép tới các vùng bộ nhớ, ánh xạ giữa các địa chỉ logic và địa chỉ vật lí. Trong trường hợp yêu cầu về bộ nhớ của các tiến trình lớn hơn dung lượng bộ nhớ vật lí thì HĐH cho phép trao đổi thông tin giữa đĩa và bộ nhớ hoặc tổ chức bộ nhớ ảo để thỏa mãn nhu cầu của tiến trình.

CHƯƠNG I. DẪN NHẬP VÀ KHÁI NIỆM:

1. Dẫn nhập:

Chúng ta thấy rằng CPU có thể được dùng chung bởi nhiều process. Do kết quả định thời CPU, chúng ta có thể cải tiến hiệu suất của CPU lẫn tốc độ đáp ứng của người dùng. Để thực hiện việc làm tăng hiệu quả này chúng ta phải lưu giữ vài quá trình trong bộ nhớ, tức là chúng ta cần phải dùng bộ nhớ dùng chung.

Bộ nhớ là trung tâm hoạt động của hệ thống máy tính hiện đại. Bộ nhớ gồm một dãy lớn của các words hoặc các byte mà mỗi cái đó đều có địa chỉ của riêng chúng.

- Quản lí bộ nhớ là công việc của HĐH với sự hỗ trợ của phần cứng nhằm phân phối, sắp xếp các process trong bộ nhớ sao cho hiệu quả.
- Mục tiêu cần đạt được là nạp càng nhiều process vào bộ nhớ càng tốt (gia tăng mức độ đa chương).
- Trong hầu hết các hệ thống, Kernel (nhân của hệ điền hành) sẽ chiếm một phần cố định của bộ nhớ, phần còn lại phân phối cho các process.

2. Khái niệm:

Địa chỉ luận lí hay còn gọi là địa chỉ ảo (Virtual Address) : là tất cả các địa chỉ do bộ vi xử lí tạo ra. Tập hợp tất cả các địa chỉ luận lí tạo nên không gian địa chỉ luận lí.

Địa chỉ vật lý hay còn gọi là địa chỉ thực : là địa chỉ thực tế mà trình quản lý bộ nhớ nhìn thấy và thao tác. Tập hợp tất cả các địa chỉ vật lý tạo nên không gian địa chỉ vật lý.

Paging & Page File : Paging(Phân trang) là kỹ thuật được sử dụng bởi hệ thống bộ nhớ ảo để đảm bảo rằng dữ liệu của chúng ta cần là tồn tại (available) càng nhanh càng tốt. HĐH copy một số trang nhất định từ thiết bị lưu trữ vào bộ nhớ chính. Khi chương trình cần một trang mà hiện tại không tồn tại trong bộ nhớ chính, HĐH sẽ copy trang cần thiết đó vào bộ nhớ và copy trang khác vào lại ổ đĩa. Page File là một file trên ổ cứng, được Windows sử dụng làm bộ nhớ ảo để lưu trữ các chương trình và dữ liệu, khi bộ nhớ vật lý (RAM) không đủ chỗ chứa.

CHƯƠNG II: HỆ THÔNG WINDOWS QUẢN LÝ BỘ LOGIC THEO CẤU TRÚC PHÂN TRANG(PAGING):

1. Physical Storage(Bộ nhớ vật lí) :

Mức tối đa của dung lượng bộ nhớ vật lý được hệ thống Windows hỗ trợ khoảng từ 2GB->2TB, tùy thuộc vào phiên bản của Windows.

Phiên bản	Hệ 32 bit	Hệ 64 bit
Windows 2000 Professional	4GB	Không hỗ trợ
Windows XP	4GB	128GB
Windows Server 2003 SP2 Datacenter Edition	128GB	2TB
Windows Server 2008 R2 Datacenter	Không hỗ trợ	2TB
Windows 7 Ultimate	4GB	192GB

(Bảng so sánh sự giới hạn bộ nhớ vật lí ở các phiên bản khác nhau của HĐH Windows)

Không gian địa chỉ ảo của một tiến trình có thể nhỏ hơn hoặc lớn hơn tổng dung lượng bộ nhớ vật lý trên máy tính. Tập hợp các không gian địa chỉ ảo của một tiến trình được cư trú trong bộ nhớ vật lý được gọi là " working set " .

2. Virtual Addess Space :

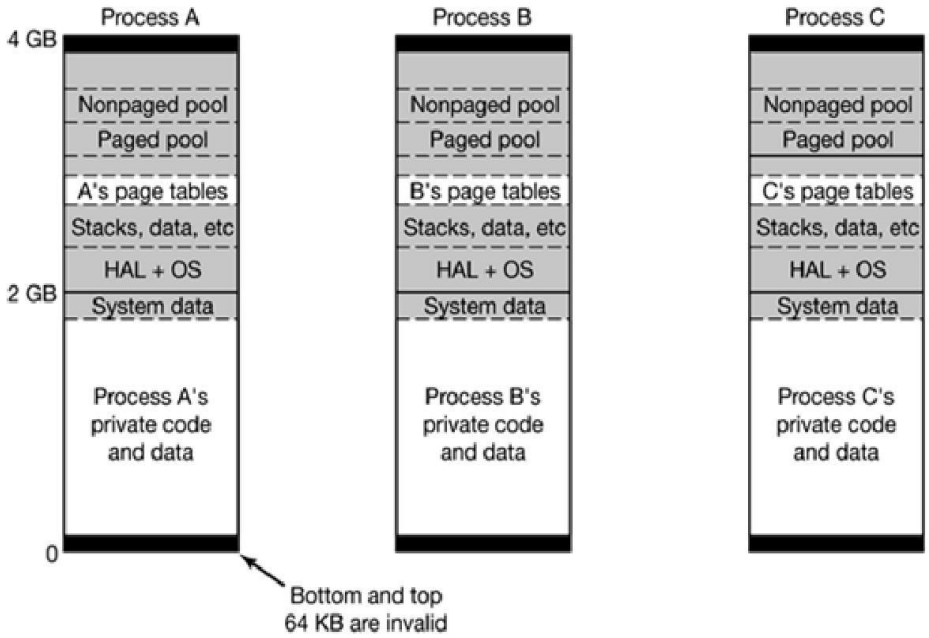
Mỗi tiến trình người dùng trên nền tảng Windows 32-bit được cấp phát một không gian địa chỉ ảo (Virtual Address Space) là 4 GB.Còn ở nền tảng Windows 64-bit, mỗi tiến trình người dùng được cấp phát một không gian địa chỉ ảo lên tới 8TB. Tất cả các tiểu trình của một tiến trình có thể truy cập vào vùng địa chỉ ảo của chính nó, tuy nhiên những tiểu trình đó lại không thể truy cập vào vùng địa chỉ ảo thuộc về một tiến trình khác.

Không gian địa chỉ ảo của một tiến trình là tập hợp tất cả các địa chỉ bộ nhớ ảo mà nó có thể được sử dụng. Các không gian bộ nhớ ảo này được thiết lập riêng tư (private) và các tiến trình khác sẽ không được chia sẻ.

Windows trên hệ thống 32 bit x86 systems có thể truy xuất (access) trên 4GB bộ nhớ vật lý. Do thực tế bus addr của bộ vi xử lý (processor) là 32 lines hay 32 bits chỉ có thể truy xuất vùng addr từ 0x00000000 đến 0xFFFFFFFF tức chỉ có 4GB.

4GB này được chia ra làm hai phần :

- 0->2 GB dưới**:** chứa dữ liệu và lệnh riêng của từng tiến trình.Vùng này hoạt động ở chế độ user-mode, người dùng chỉ thao tác được trên vùng 2GB này.
- 2->4 GB trên: chứa các thành phần dữ liệu thuộc về hệ điều hành, được chia sẻ chung cho các tiến trình, hoạt động ở chế độ kernel-mode, vùng này do HĐH quản lí người dùng không thể tác động vào vùng này.



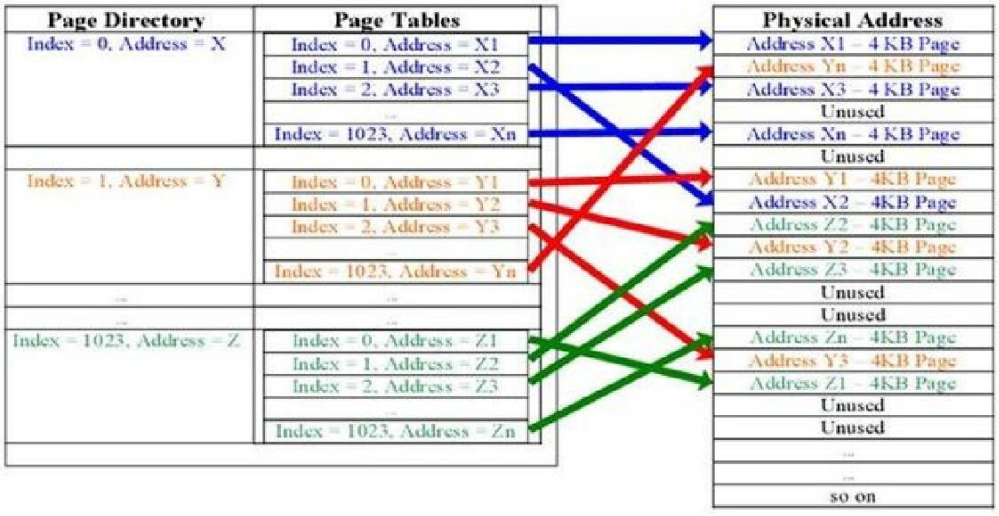
Hình II. 1: Không gian địa chỉ ảo được cấp phát cho 3 tiến trình.

Windows làm thế nào cấp phát vùng addr 4GB cho nhiều processes khi tổng bộ nhớ của nó có thể truy xuất cũng bị giới hạn bởi 4GB? Để đạt được điều này, Windows dùng một đặc tính của x86 processer (386 trở lên) được biết đến là "phân trang" (paging). Paging

cho phép phần mềm sử dụng một địa chỉ nhớ (được biết đến như logical address: địa chỉ luận lý) khác với địa chỉ nhớ vật lý (physical memory address). Paging của processor chuyển đổi logical address thành physical address một cách dễ dàng. Điều này cho phép mọi process trong system có vùng addr logical 4GB của chính nó. Để hiểu điều này chi tiết hơn, chúng ta hãy bắt đầu tìm hiểu cách paging trong môi trường làm việc của x86 Processer.

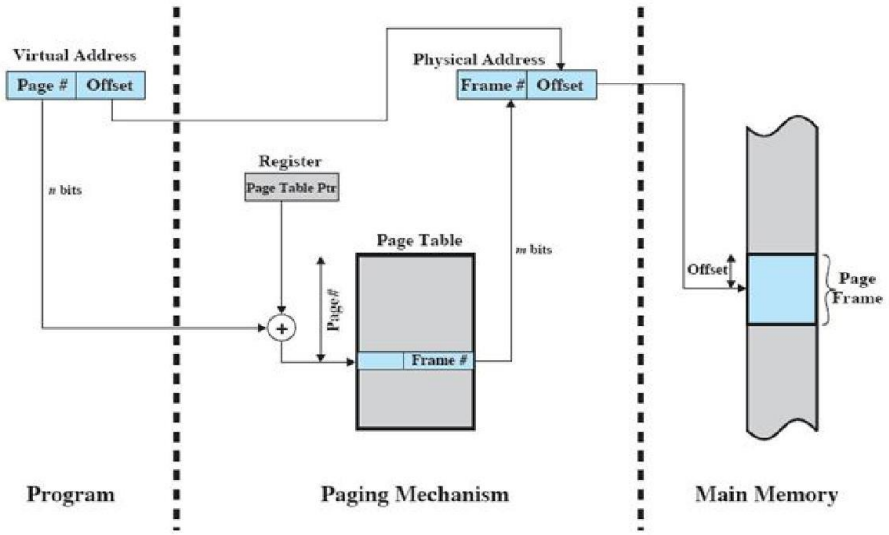
3. Phân trang (paging) :

Trong bộ xử lý x86 vùng địa chỉ vật lý (physical address space) được chia thành các page có kích thước 4KB. Vì vậy để đánh địa chỉ 4GB bộ nhớ, chúng ta cần 1 MB (1024x1024) các trang (pages) có kích thước 4KB.Bộ vi xử lý dùng 2 lớp cấu trúc để tham chiếu đến 1 Mega pages này. Chúng ta có thể nghĩ nó như là một ma trận 2 chiều kích thước là 1024x1024 các phần tử.Chiều thứ nhất được biết đến như là Page Directory và chiều thứ 2 được biết như Page Table.Vì vậy chúng ta cần cài đặt một Page Directory với 1024 thành phần, mỗi thành phần point (trỏ đến) đến một Page Table. Điều này cho phép chúng ta có 1024 Page Table.Mỗi Page Table lại có 1024 thành phần, mỗi thành phần lại trỏ đến 4KB page.



Hình II. 2: Paging in x86 Processor

Mỗi thành phần Page Directory Entry (PDE) có kích thước 4 bytes và trỏ đến một Page Table. Tương tự , mỗi Page Table Entry (PTE) có kthước 4 bytes và trỏ đến một physical address (địa chỉ vật lý) của 4KB page.Để chứa 1024 PDE mà mỗi thành phần lại chứa 1024 PTE, chúng ta cần tổng bộ nhớ là 4x1024x1024 bytes, có nghĩa là 4MB. Vì vậy chia toàn bộ 4GB vùng address cho 4KB page, chúng ta cần 4MB vùng nhớ.



Hình II. 3 Address Translation trong hệ thống phân trang.

Không gian địa chỉ ảo được Windows quản lý theo kiểu phân trang, kích thước mỗi trang 4kB = 2^{12} byte; vì 4GB=2^{20} x 4kB => bộ nhớ ảo chứa 2^{20} trang ảo.

Mỗi trang ảo có thể nằm ở một trong 3 trạng thái:

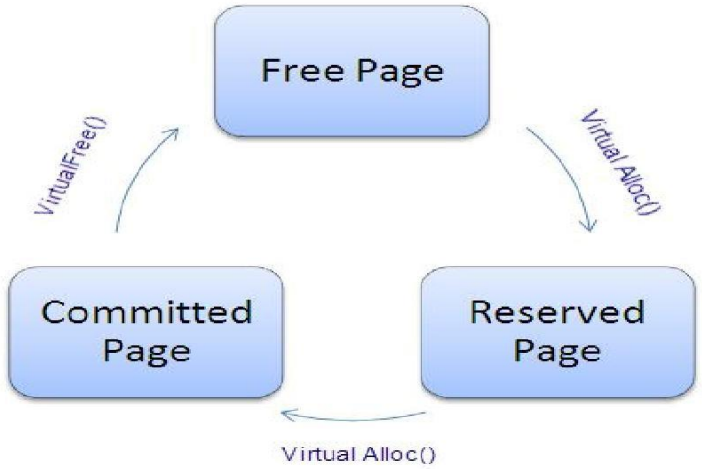
- **Free**: là trang chưa dùng để chứa dữ liệu và có thể được sử dụng bởi bất kỳ tiểu trình nào của tiến trình chứa nó, trang Free không được đưa vào RAM.Tham chiếu đến trang free gây ra lỗi (Page Fault), lỗi này không xử lý được.
- **Committed**: là trang đã được ánh xạ dữ liệu, đang nằm trên RAM hoặc vùng
- **Paging File**. (Paging file là 1 vùng trên bộ nhớ ngoài được tổ chức như RAM, cho cảm giác như RAM được mở rộng và được dùng để chứa nội dung các trang bị đẩy ra từ RAM).Khi CPU gọi đến trang Committed nếu trang đang ở vùng Paging File thì xuất hiện Page Fault, trang được

đẩy vào RAM để hoạt động.Còn nếu trang đang ở RAM thì không xuất hiện Page Fault.

- **Reserved**: là trang hiện tại chưa có trong bộ nhớ vật lý, được đặt trước để chứa dữ liệu hoặc code.Khi CPU gọi đến trang này thì xuất hiện Page Fault. Trang

Ví dụ: Khi 1 tiểu trình được sinh ra thì nó chỉ cần ngay 1 trang committed ở thời điểm hiện tại nhưng cũng có thể đặt trước đến 1MB các trang reserved liên tiếp ở ngay cạnh trang committed, để tiểu trình sử dụng sau này.

Hình vẽ mô tả mối quan hệ giữa 3 trạng thái của trang được xử lý để chuyển sang trạng thái committed



Hình II. 4 Chuyển đổi trạng thái trang ảo bằng hàm API

4. Windows Page Table Management(Quản lí trang bảng) :

Trong Windows, mỗi process có Page Directory và Page Table của chính nó. Vì vậy Windows cấp 4MB của vùng nhớ này cho mỗi process. Khi một process được cài đặt, mỗi thành phần trong Page Directory chứa physical address (địa chỉ vật lý) của Page Table.

Các thành phần trong Page Table hoặc là valid (hợp lệ) , hoặc là invalid (không hợp lệ). Các thành phần valid chứa physical address của 4KB page cấp cho process. Một thành phần invalid (không hợp lệ) chứa một vài bits đặc biệt đánh dấu nó không hợp lệ và các thành phần này được biết như Invalid PTEs (Page Table Entry). Khi memory được cấp cho process, các thành phần trong Page Table được lắp các địa chỉ vật lý của các pages đã cấp. Ở đây là một process không

biết bất kỳ điều gì về địa chỉ vật lý và nó chỉ sử dụng logical address (địa chỉ luận lý) mà thôi. Chi tiết về việc logical address nào tương ứng với physical address nào được quản lý chuyển đổi bởi Windows Memory Manager và Processor (bộ vi xử lý).

Address tại Page Directory nào đó của một process được định vị trong physical memory và được tham chiếu đến như là Page Directory Base address. Page Directory Base address này được chứa trong một thanh ghi đặc biệt của CPU là CR3 (trên nền x86). Để chuyển đổi context khác, Windows tải một giá trị mới của CR3 để trỏ đến một Page Directory base mới của process. Với cách này mỗi process sẽ lấy được các phần phân chia cả 4GB physical address space (không gian địa chỉ vật lý) của chính nó. Tất nhiên, tổng dung lượng bộ nhớ cấp tại một thời điểm cho tất cả các process trong hệ thống là không thể vượt quá số lượng RAM+kích thước pagefile nhưng theo lược đồ đã thảo luận ở trên thì cho phép Windows cấp cho mỗi process vùng address logical (hay Virtual: ảo) 4GB . Chúng ta gọi nó là vùng địa chỉ ảo (Virtual Addres sapce) bởi vì ngay mỗi process có đến cả range (phạm vi) là 4GB address, nó chỉ có thể sử dụng memory cấp cho nó. Nếu một process thử truy xuất (access) một địa chỉ không được cấp phép, nó sẽ gây ra một access violation (sự vi phạm truy xuất) bởi vì PTE tương ứng với address trỏ đến một giá trị ko hợp lệ (invalid value). Cũng vậy, process ko thể cấp memory nhiều hơn những gì nó được phép trong system. Phương thức tách riêng logical memory từ physical memory này có nhiều thuận lợi. Một process có được một vùng address 4GB tuyến tính , do đó các lập trình viên ứng dụng ko còn phải lo lắng về segments và hoàn toàn không giống như những ngày tháng cũ làm việc với DOS. Nó cũng cho phép Windows chạy nhiều prosses cùng một lúc và cho phép chúng dùng physical memory trên máy tính mà không phải lo lắng chúng sẽ đè lên trên vùng address space của process khác. Một logical address trong một process sẽ không bao giờ trỏ đến một physical memory được cấp cho process khác (trừ khi chúng sử dụng phần nào để shared memory). Vì vậy,một process có thể không bao giờ read hay write vào memory của process khác.

Sự chuyển đổi từ logical address (địa chỉ luận lý) sang physical address (địa chỉ vật lý) được thực hiện bởi bộ vi xử lý. Một 32bit logical address được chia thành 3 phần như hình dưới đây:

10 bits	10 bits	12 bits

Vi xử lý sẽ loads physical address của page directory lưu trữ trong CR3. Rồi nó được sử dụng 10 bits thấp từ logical address như là một chỉ mục trong Page directory. Tạo cho processor một page directory entry (PDE) trỏ đến một Page Table. 10 bits kế đến được sử dụng như một chỉ mục trong Page Table. Sử dụng 10 bits này, nó lấy một page table entry (hay PTE) trỏ đến một 4KB physical page. 12 bits thấp nhất được sử dụng đánh địa chỉ các bytes riêng lẻ trên một page.

5. Windows Memory Protection (Bảo vệ bộ nhớ):

Windows hổ trợ sự bảo vệ memory cho tất cả các processes mục đích để một process không thể truy xuất một vùng bộ nhớ của process khác. Điều này đảm bảo các hoạt động của nhiều processes cùng lúc một cách trôi chảy. Windows đảm bảo chế độ bảo vệ này bằng cách theo các bước sau:

- Chỉ đặt physical address của memory được định vị trong PTE cho một process. Điều này đảm bảo rằng process bắt được một access violation nếu nó thử truy xuất một địa chỉ mà không được định vị.
- Một rouge process (tiến trình đang thực thi) có thể cố gắng thay đổi page tables của nó để nó có thể truy xuất physical memory thuộc về một process khác, điều này sẽ dẫn đến lỗi trang.Windows bảo vệ khỏi loại tấn công này bởi cơ chế cất giữ các page tables trong kernel address space.

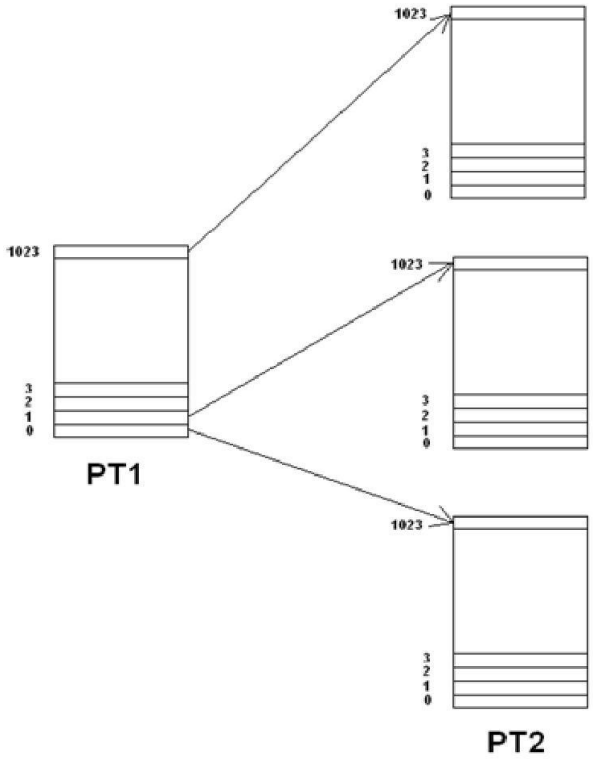
6. Cấu trúc đa bảng trang :

Windows sử dụng hai cấp bảng trang: bảng trang cấp 1 (PT1-page table 1) và bảng trang cấp 2 (PT2-page table 2)để quản lý bộ nhớ nhằm tránh việc quản lý tất cả các bảng trang trong bộ nhớ cùng một lúc, mỗi bảng trang bằng kích thước một trang ảo là 4KB.Mỗi tiến trình có một bảng trang cấp 1, và 1024 bảng trang cấp 2.

Bảng trang cấp 1 quản lý địa chỉ vật lý của bảng trang cấp 2.Bảng trang cấp 1 có 1024 mục mỗi mục 4 byte (hay 32 bit), quản lý địa chỉ vật lý của 1024 bảng trang cấp 2.Trong mỗi

mục, 20 bits đầu dùng chứa địa chỉ vật lý của bảng trang cấp 2 nếu bảng trang cấp 2 đã được nạp vào RAM; 12 bit cuối chứa các thuộc tính của bảng trang đó, trong đó 1 bit Present/Absent bằng 1 nếu trang đã trên RAM, ngược lại nó được gán giá trị 0. Trong trường hợp trang cấp 2 chưa được nạp vào RAM thì 20 bit đầu chứa toàn 0, bit Present/Absent cũng bằng 0.

Bảng trang cấp 2 quản lý địa chỉ vật lý của trang ảo.Bảng trang cấp 2 cũng có1024 mục, mỗi mục 4 byte (hay 32 bit), quản lý địa chỉ của 1024 trang ảo.Như vậy mỗi bảng trang cấp 2 quản lý được địa chỉ vật lý của 4MB trang ảo.Cấu tạo của mỗi mục trong PT2 cũng tương tự như mỗi mục trong PT1. Tức 20 bits đầu dùng chứa địa chỉ vật lý của trang ảo và 12 bits còn lại lưu trữ một số thuộc tính bảo vệ; bit Present/Absent bằng 1 nếu trang đó trên RAM, ngược lại thì bằng 0.



Hình II. 5 Cấu trúc đa bảng trang

7. 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 process lớn hơn không gian bộ nhớ cấp cho process 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.

8. 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.

Swapping policy: hai ví dụ

- *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 chế độ ưu tiên (priority-based scheduling) process có độ ưu tiên thấp hơn sẽ bị swap ot nhường chỗ cho process có độ ưu tiên cao hơn mới đến được nạp vào bộ nhớ để thực thi

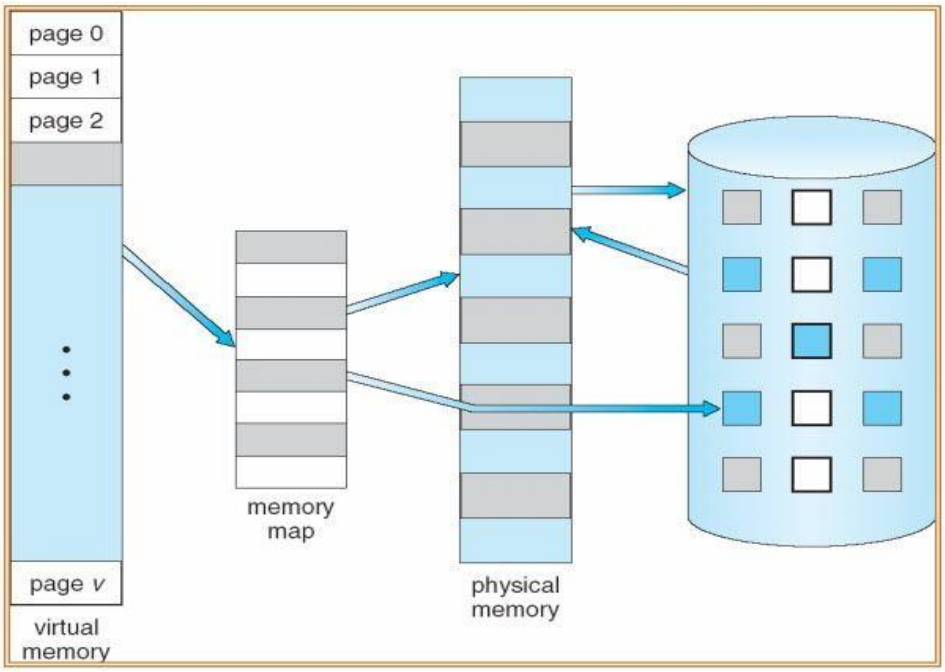
(*hiện nay ít hệ thống sử dụng cơ chế swapping trên)*

CHƯƠNG III. QUẢN LÍ BỘ NHỚ ẢO(BỘ NHỚ LOGIC):

1. Bộ nhớ ảo (Virtual Memory):

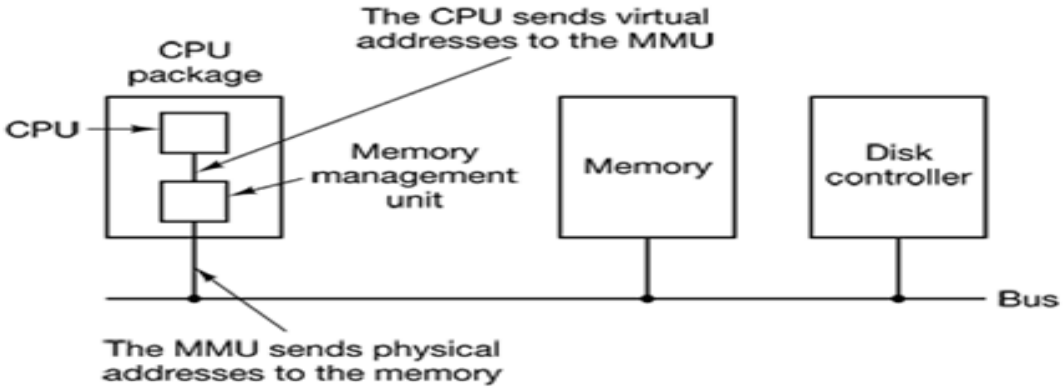
Bộ nhớ ảo là một kĩ năng cho phép xử lí 1 tiến trình không được nạp toàn bộ vào bộ nhớ vật lí. Bộ nhớ ảo là mô hình hóa bộ nhớ như 1 bảng lưu trữ rất lớn và đồng nhất, tách biệt hẳn khái niệm không gian địa chỉ ảo (virtual address space) và không gian vật lý (physical space). Một điểm lợi quan trọng của cơ chế này là các chương trình được chạy có thể lớn hợn bộ nhớ vật lý.

Ngoài ra, bộ nhớ ảo phóng đại bộ nhớ chính thành bộ nhớ luận lý cực lớn khi được hiển thị bởi người dùng. Kỹ thuật này giải phóng người lập trình từ việc quan tâm đến giới hạn kích thước bộ nhớ. Bộ nhớ ảo cũng cho phép các quá trình dễ dàng chia sẻ tập tin và không gian địa chỉ, cung cấp cơ chế hữu hiện cho quá trình.



Hình III. 1: Minh họa bộ nhớ ảo lớn hơn bộ nhớ vật lí.

2. Ánh xạ (dịch) từ bộ nhớ Logic sang bộ nhớ thực :



Hình III. 2 CPU làm việc với MMU

Bộ phận dịch (MMU).MMU là viết tắt của Memory Management Unit. Để thi hành một lệnh nào đó,CPU gửi địa chỉ ảo đến MMU. Thông qua MMU, địa chỉ ảo này sẽ được ánh xạ tương ứng với một địa chỉ vật lý cụ thể và được gửi tới bus địa chỉ. Cuối cùng thông qua bus địa chỉ để truy cập tới 1 vùng nhớ cụ thể trên RAM.

3. Page Faults :

Định nghĩa: Page faults cũng là một vấn đề đối với các loại phần mềm hiện nay, và một phần cũng do hệ thống phần cứng, khi một chương trình truy cập đến một page được ánh xạ trong không gian địa chỉ ảo nhưng chưa được lưu vào bộ nhớ vật lý.

MMU trong bộ vi xử lí chính là phần cứng đóng vai trò phát hiện những trường hợp xảy ra Page faults. Hệ điều hành xử lý page faults bằng cách:

- Tạo ra những page yêu cầu có thể dễ dàng hiểu được và chúng được đặt tại một nơi trong địa chỉ vật lí.
- Loại bỏ những chương trình trong trường hợp chúng có những biểu hiện truy xuất không hợp lệ.

Trái ngược với Page Faults nó không phải là những loại lỗi thường xuyên và cần thiết để gia tăng số lượng bộ nhớ sẵn có để cung cấp cho chương trình cho bất cứ HĐH nào có sử dụng bộ nhớ ảo bao gồm : Microsoft Windows, Mac os X , Linux, * BSD, Solaris, AIX, and HP-US và z/OS. Một điều đáng lưu ý ở đây mà Microsoft dùng thuật ngữ hard fault để định nghĩa page fault :

Lý do gây ra Page Faults:

a. Một lỗi trang xảy ra khi bộ xử lý truy cập tới một địa chỉ mà các trang tương ứng với địa chỉ đó không được đánh dấu trong cácMMU (đơn vị quản lý bộ nhớ) khi được nạp trong bộ nhớ. Các lỗi phần cứng hoặc lỗi phát sinh trong trường hợp này phụ thuộc vào kiến trúc tập lệnh của bộ xử lý. Với một số tập lệnh kiến trúc, các lỗi phần cứng trong câu hỏi có thể được tạo ra bởi các điều kiện khác hơn là một truy cập vào một địa chỉ trong một trang không được tải vào bộ nhớ, điều này có nghĩa là bộ xử lý cho rằng lỗi phần cứng sẽ phải tìm xem nó có tương ứng với một trang lỗi hay không.

b. Một khái niệm có liên quan với Page Fault được gọi là Protection fault được tạo ra để truy cập trang mà các trang tương ứng với địa chỉ yêu cầu được đánh dấu trong các đơn vị quản lý bộ nhớ khi được nạp trong bộ nhớ, nhưng không được đánh dấu khi cho phép các hoạt động mà các bộ vi xử lý đã thực hiện. Ví dụ, trang này có thể được đánh dấu là không cho phép lưu trữ, trong trường hợp cố gắng để lưu trữ vào các trang sẽ tạo ra một lỗi bảo vệ, hoặc nó có thể được đánh dấu là không cho phép thực thi mã, trong trường hợp cố gắng để lấy một hướng dẫn từ trang đó sẽ tạo ra một lỗi bảo vệ. Một lần nữa, các lỗi phần cứng hoặc lỗi phát sinh trong trường hợp này phụ thuộc vào tập lệnh của bộ xử lý.

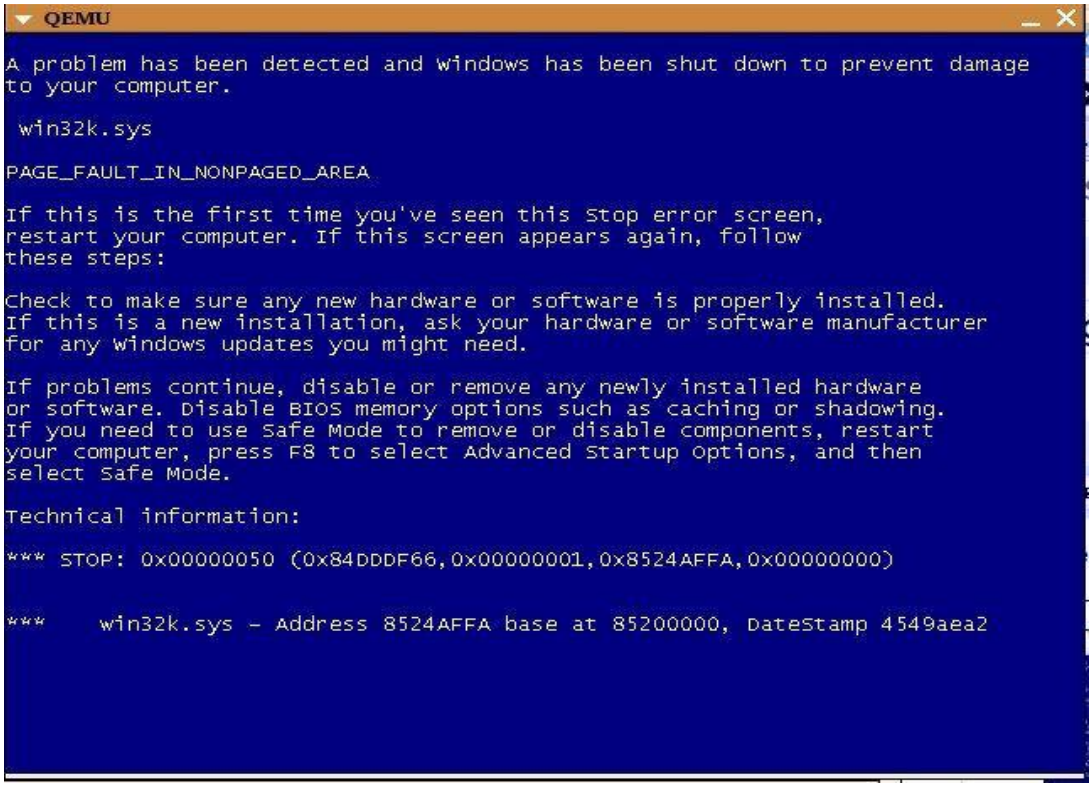
Các thuật ngữ "Page fault" và "Protection Fault" được sử dụng ở đây để cho thấy cách hệ điều hành xử lý lỗi, và không nhất thiết phải là tên dành cho các lỗi phần cứng xảy ra. Ví dụ, trên kiến trúc x86, truy cập vào các page mà không được trình bày và truy cập vào các trang được bảo vệ đều được báo cáo thông qua một lỗi phần cứng được gọi là một lỗi "trang ", và các phần cứng xử lý cung cấp thông tin cho các bộ xử lý lỗi trang cho biết những loại truy cập được kích hoạt lỗi, vì vậy mà các cách xử lý như thế có thể được hệ điều hành phân biệt. Việc sử dụng các lỗi bảo vệ không nên nhầm lẫn với các trường hợp ngoại lệ lỗi x86 nói chung bảo vệ, được sử dụng để vi phạm tín hiệu bộ nhớ truy cập dựa trên phân khúc.

Các loại Page Faults có thể khắc phục :

- Loại 1: Truy nhập đến 1 trang reserved tức là trang này mới được đặt trước mà chưa được đưa vào RAM. Khi đó Page Fault xảy ra trang reserved sẽ được xử lí để thành trang committed tức là ánh xạ trang đó vào trong RAM.

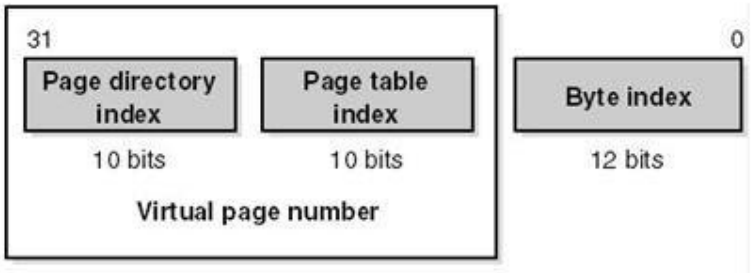
- Loại 2: Page Fault xảy ra trong kĩ thuật copy-on-write.

Khi xảy ra lỗi trang, cần phải mang trang vắng mặt vào bộ nhớ. Nếu không có một khung trang nào trống, hệ điều hành cần thực hiện công việc thay thế trang – nghĩa là chọn một trong trong bộ nhớ mà không được sử dụng tại thời điểm hiện tại và chuyển nó ra không gian swapping trên đĩa để giải phóng một khung trang dành chỗ nạp trang cần truy xuất vào bộ nhớ.



Hình III. 3 Một "Blue Screen" xuất hiện khi xảy ra PAGE FAULT

4. Quá trình dịch địa chỉ ảo :

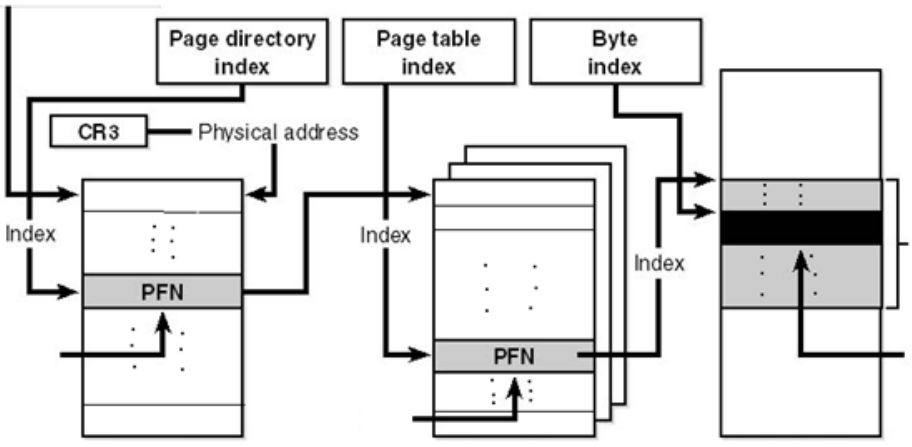


Hình III. 4 Tổ chức 32-bits địa chỉ ảo

Một địa chỉ ảo trang Windows được chia làm 3 phần

Giả sử CPU phát sinh một địa chỉ ảo là 1 số 32 bit để tìm đến 1 byte nhớ. Bộ phận dịch Memory Management Unit (MMU) nhận địa chỉ và thực hiện thao tác dịch:

- Bước 1: MMU nhận 10 bít đầu tiên tìm trong PT1 để lấy địa chỉ vật lý của PT2. Nếu PT2 reserved thì Page Fault và nạp PT2 vào RAM.
- Bước 2**:** Khi PT2 đã có trong RAM, MMU dùng 10 bits tiếp theo để tìm trong PT2 lấy địa chỉ vật lý của trang chứa byte cần tìm.Nếu trang ở trạng thái reserved thì xảy ra Page Fault và nạp trang vào RAM.
- Bước 3: Khi trang đã có trong RAM, MMU dùng 12 bits cuối để tìm đến byte cụ thể ở trong khung trang và trả về cho CPU địa chỉ vật lý cụ thể của byte cần tìm.



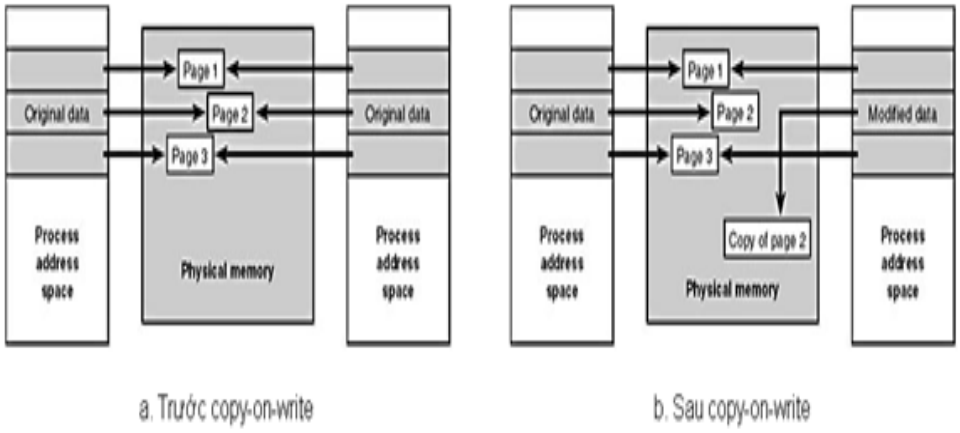
Hình III. 5 Minh họa quá trình dịch.

5. Kỹ thuật Copy-on-Write:

Windows cho phép nhiều tiến trình khác nhau chia sẻ cùng một physical page trên RAM để tiết kiệm bộ nhớ. Các trang dùng chung này có thể cho phép các tiến trình dùng thay đổi nội dung hoặc không, căn cứ vào thuộc tính bảo vệ của trang chia sẻ đó.

Trong trường hợp thuộc tính bảo vệ không cho phép các tiến trình chỉnh sửa trang để tránh việc một tiến trình khi thay đổi nội dung trang sẽ làm ảnh hưởng đến tiến trình khác, Windows sử dụng kỹ thuật copy-on-write với nguyên lý như sau:

" Tất cả các tiến trình cùng ánh xạ đến một trang dùng chung cho đến khi một tiến trình nào đó làm thay đổi nội dung của trang. Khi đó, Page Fault xảy ra báo cho hệ thống xử lý tình huống như sau: tiến trình làm trang thay đổi sẽ copy một bản của trang dùng chung ra một vùng bộ nhớ riêng và thao tác trên vùng nhớ đó; các tiến trình còn lại vẫn sử dụng trang nhớ cũ."



Hình III. 6 Minh họa kỹ thuật Copy-On-Write

6. Những thành phần được nạp vào RAM:

Bộ nhớ RAM chia làm hai phần:

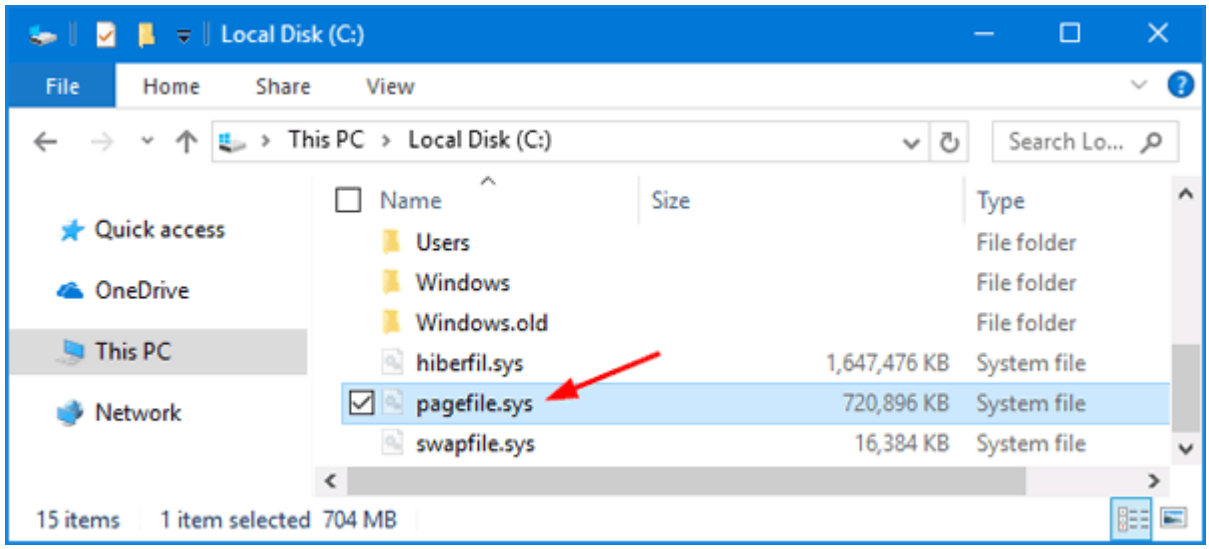
- The Non-Paged are **:** Có một số phần của hệ điều hành rất quan trọng và không bao giờ được phân trang. Khu vực của RAM được dùng cho những phần này được gọi là "Non-Paged are", chỉ dành cho những code lõi của hệ thống (core code of the system).
- The Page Pool **:** Được dùng để lưu trữ:

- Mã chương trình.
- Pages đã có dữ liệu được ghi.
- Một phần dung lượng cơ bản dành cho các "file cache", lưu trữ thông tin các tập tin vừa được xử lý đọc/ghi từ ổ cứng.

Bất kỳ lượng RAM còn lại nào sẽ được sử dụng để làm dung lượng bộ nhớ Cache lớn hơn.

9. Page File ở đâu :

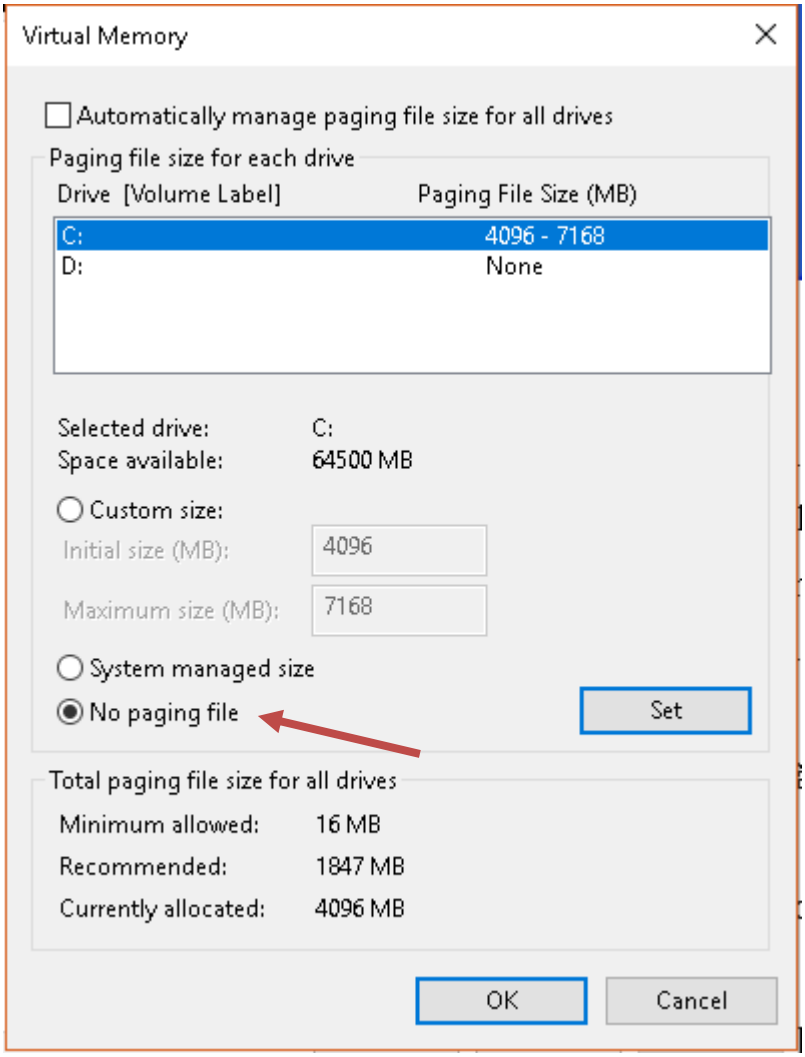
Trong hệ thống Windows, page file là một tệp tin ẩn được lưu với tên pagefile.sys.File này được tạo ra mỗi lần hệ điều hành được boot. Để xem file này, chúng ta vào ổ đĩa lưu page file vào Folder Options View chọn "Show hidden files and folders" và bỏ chọn mục "Hide Protected mode System files".



Hình III. 7 Một Page File trong hệ thống Windows 10

Theo mặc định, Windows tự set dung lượng của "page file" gấp 1.5 lần dung lượng bộ nhớ RAM trên máy tính.

Để tắt bộ nhớ ảo, chúng ta vào Control Panel System, chọn tab Advance và sau đó click vào nút Settings trong Performance, tiếp tục click vào tab Advanced rồi click vào nút Change.Chọn "No paging file" và click nút Set để xóa file pagefile.sys.



Hình III. 8 Tắt "page file" trên hệ thống Windows 10

Tuy nhiên việc không sử dụng bộ nhớ ảo sẽ làm tiêu tốn khá nhiều tài nguyên RAM, và có thể dẫn tới hiệu suất hoạt động của hệ thống bị giảm.Lý do ở đây là trên thực tế, các chương trình khi thực thi thường yêu cầu một lượng bộ nhớ lớn hơn lượng bộ nhớ thực sự được đưa vào sử dụng.Các yêu cầu địa chỉ ô nhớ này sẽ được hệ thống giao cho một nới nào đó. Nếu tệp tin trang (page file) là có sẵn, hệ thống có thể giao cho nó.Nhưng nếu chúng ta tắt "page file" (không sử dụng bộ nhớ ảo) thì hệ thống sẽ giao

toàn bộ cho RAM, điều này làm lãng phí khá nhiều tài nguyên RAM, có khi đến vài trăm megabytes mỗi chương trình đang chạy.

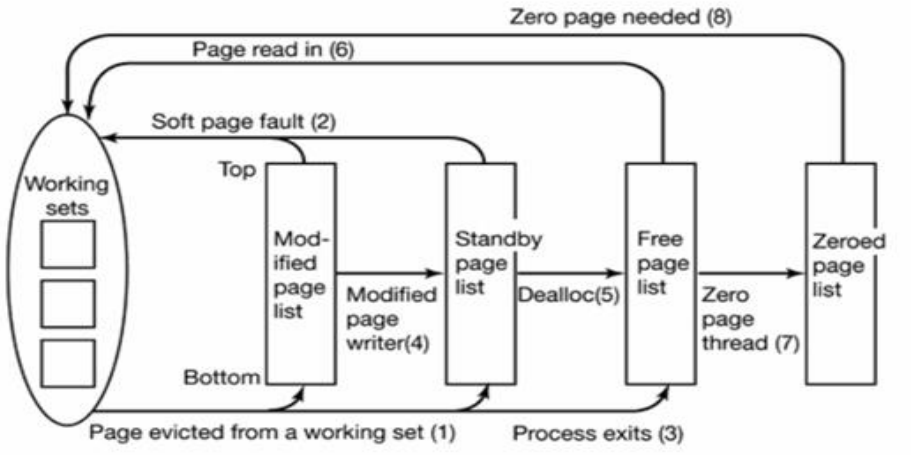
CHƯƠNG IV. QUẢN LÍ BỘ NHỚ VẬT LÝ:

1. Phân chia vùng trong RAM:

- Working Set: Tập các trang đang hoạt động.
- Modified List: Các trang bị loại khỏi Working Set, vẫn còn liên quan đến tiến trình đã gọi nó.
- Standby List: Các trang bị loại ra khỏi Modified List, cũng còn liên quan đến tiến trình gọi nó, nhưng có 1 bản sao trên vùng Paging File ở bộ nhớ ngoài, vì vậy có thể xóa bản gốc trên RAM nếu cần.
- Free List: Các trang bị loại khỏi Standby List vì không còn gắn với tiến trình nào nữa.
- Zeroed List: Các trang chuyển từ Free List và được ghi lại hoàn toàn bằng mã 0.

2. Cách thức chuyển đổi giữa các vùng trong RAM:

Giải thuật thay trang đảm bảo cho các trang ảo được nạp vào RAM khi cần và được xóa khỏi RAM khi không cần dùng nữa để thay bằng trang khác. Sơ đồ sau mô tả giải thuật thay trang.



Hình III. 9 Các vùng trên RAM

- Khi tiến trình gọi đến 1 trang thì nó được nạp vào vùng Working Sets.
- Cứ khoảng 4s, nếu 1 trang của 1 tiến trình nào đó rỗi, nó sẽ bị đẩy từ Working Set sang đáy Modified List hoặc Standby List(tùy trường hợp cụ thể), biểu diễn bởi (1). Các trang ở 2 vùng này vẫn có giá trị và có thể được tiến trình gọi lại và nạp vào Working Set, biểu diễn bởi (2). Khi tiến trình kết thúc và trang không còn chia sẻ với tiến trình nào khác, trang từ Working Set bị đẩy sang Free List, biểu diễn bởi (3).
- Sau một thời gian nhất định các trang ở Modified List bị đẩy sang vùng Standby List, biểu diễn bởi (4). Sự khác biệt giữa 2 vùng này là Modified List có thể được đẩy vào Working Set nhanh hơn, còn Standby List có 1 bản backup ở bộ nhớ ngoài, vì vậy các trang ở Standby List có thể được xóa đi nếu RAM đầy, khi cần thì nạp trang backup.
- Các trang ở Standby List khi không còn gắn với tiến trình nào nữa thì bị đẩy ra vùng Free List, biểu diễn bởi (5). Các trang ở Free List vẫn chứa dữ liệu nhưng đã không còn giá trị, và có thể bị ghi đè bởi 1 trang mới chuyển vào Working Set hoạt động, biểu diễn bởi (6).
- Trong một số trường hợp đặc biệt, các tiến trình đòi hỏi các trang hoàn toàn chưa chứa dữ liệu đê ghi thông tin mới, các trang này được lấy từ vùng Zeroed List. Zeroed List có được nhờ xóa dữ liệu các trang ở Free List, biểu diễn bởi (7)

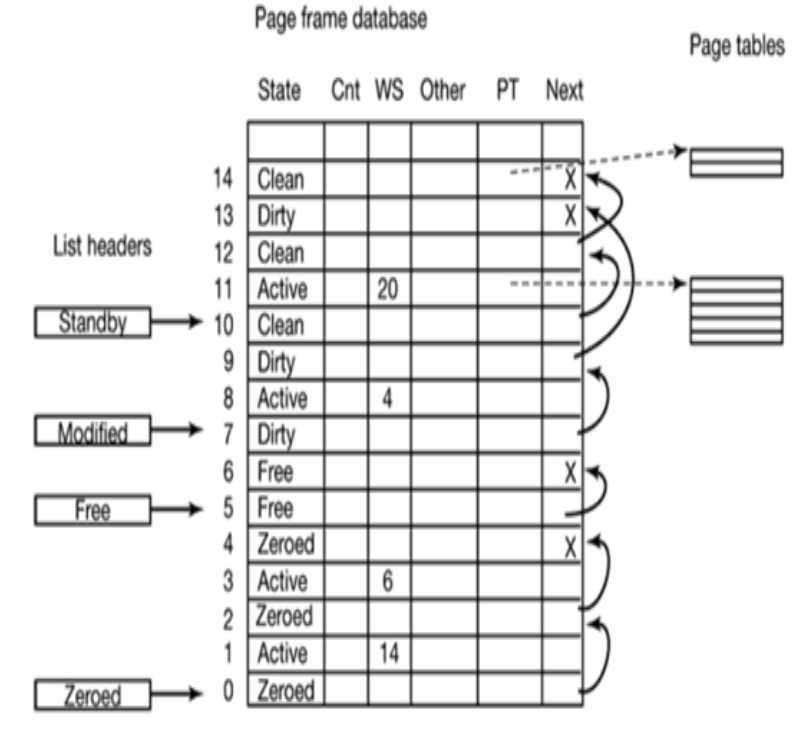
3. **CSDL về khung trang**:

Để quản lý RAM, các vùng trang trên RAM và tiện lợi cho việc ánh xạ bộ nhớ ảo vào RAM, Windows sử dụng 1 bảng dữ liệu về các khung trang. Các thông tin có thể đọc được từ bảng này gồm có:

- Số thứ tự trang trên bộ nhớ vật lý.
- Vùng mà trang đang tồn tại.
- Số bảng trang đang trỏ đến trang 1 trang.

- Con trỏ trỏ đến bảng trang đang sử dụng trang.

Con trỏ đến trang tiếp theo trong cùng vùng List trên RAM



Hình III. 10 Bảng dữ liệu khung trang

KẾT LUẬN

- Trong suốt khoảng thời gian nghiên cứu và học tập vừa qua, nhóm đã từng bước cố gắng và hoàn thiện được bài tập lớn với các mục tiêu, nhiệm vụ đề ra như sau:
  - Trình bày khái quát về cách thức quản lí bộ nhớ HDH Windows
  - Chúng em đã khái quát khái niệm, đăng nhập, trình bày sơ lược về hệ thống windows quản lý bộ nhớ logic bảng phân trang và cuối cùng là cách thức quản lý bộ nhớ logic và bộ nhớ vật lý của hệ điều hành windows.
- Những đóng góp, lời khuyên của quý thầy cô là những kinh nghiệm quý báu tạo cơ hội cho chúng em tổng hợp và hệ thống hóa lại kiến thức đã học từ đó hoàn thành tốt bài báo cáo của chúng em.

Chúng em xin chân thành cảm ơn!