**Câu 1:** Giải thích sự khác biệt giữa địa chỉ logic và địa chỉ physic?

Địa chỉ logic (logic address): là vị trí nhớ được diễn tả trong một chương trình. Mọi tham chiếu bộ nhớ trong mã lệnh chương trình được compiler tạo ra đều là địa chỉ logic. Địa chỉ vật lý (physic address): là một địa chỉ thực trong bộ nhớ chính. Địa chỉ vật lý được chuyển từ địa chỉ logic khi trình biên dịch thức j hiện quá trình linking/loading.

**Câu 2:** Giải thích sự khác biệt giữa hiện tượng phân mảnh nội vi và ngoại vi?

- Phân mảnh ngoại (external fragmentation): là hiện tượng khi kích thước không gian nhớ còn trống đủ để thỏa mãn yêu cầu cấp phát nhưng không gian nhớ này lại không liên tục. Hiện tượng phân mảnh ngoại xảy ra khi bạn thường xuyên cấp phát vùng nhớ mới, sau đó xóa đi những phần vùng nhớ đã cấp phát một cách không thứ tự.

- Phân mảnh nội (internal fragmentation): là hiện tượng sẽ có vùng nhớ dư thừa khi ta cấpphát một vùng nhớ hơi lớn hơn kích thước yêu cầu.

**Câu 3:** Giả sử có một hệ thống sử dụng kỹ thuật phân trang theo yêu cầu. Bảng trang đượclưu trữ trong các thanh ghi. Để xử lý một lỗi trang tốn 8 miliseconds nếu có sẵn một khung trang trống, hoặc trang bị thay thế không bị sửa đổi nội dung, và tốn 20 miliseconds nếu trang bị thay thế bị sửa đổi nội dung. Mỗi truy xuất bộ nhớ tốn 100nanoseconds. Giả sử trang bị thay thế có xác suất bị sửa đổi là 70%. Tỷ lệ phát sinh lỗi trang phải là bao nhiêu để có thể duy trì thời gian truy xuất bộ nhớ (effective access time) không vượt quá 200 nanoseconds ?

- Thời gian truy xuất bộ nhớ: x = 100 nanoseconds

- Thời gian trung bình để xử lý lỗi trang: t = 0,3 \* 8 + 0,7 \* 20 = 16,4 miliseconds

- Thời gian truy xuất bộ nhớ hiệu dụng EAT <= 200 nanoseconds

- EAT = x + a\*t với a là tỉ lệ phát sinh lỗi trang => a = (EAT- x)/t => a <= (200-100)/16,4 = 0,609%

**Câu 4**: Nghịch đảo – 0

Vì khi -0 thì có thể trang ảo sẽ rất lớn, trong khi địa chỉ vật lý lại ít => không hiệu quả. Còn nhịch đảo thì đựa trên bộ nhớ vật lý, gọn hơn , cần lưu thêm id tiến trình (pid).

**Câu 5**: Tại sao kích thước trang luôn là lũy thừa của 2?

Trả lời: Kích thước trang được định nghĩa bởi phần cứng. Kích thước của một trang điển hình là lũy thừa của 2, từ 512 bytes đến 16MB trên trang, tùy thuộc vào kiến trúc máy tính. Chọn lũy thừa 2 cho kích thước trang để thực hiện việc dịch địa chỉ luận lý thành số trang và độ dời trang rất dễ dàng.

**Câu 6:** Hữu ích của bộ nhớ ảo: Lập trình viên không Lolắng với việc các máy tính khác nhau có kích thước bộ nhớ vật lý khác nhau. Thực thi một chương trình có kích thước lớn hơn nhiều so với bộ nhớ chính.

Lượng process trong bộ nhớ cũng nhiều hơn. Trừu tượng hoá bộ nhớ chính thành một mảng array lớn. Người dùng chỉ thấy được bộ nhớ thông qua bộ nhớ luận lý (logical memory). Từ đó giúp người lập trình viên đỡ cơ cực hơn. Bộ nhớ ảo cũng giúp cho các tiến trình chia sẻ dữ liệu với nhau dễ dàng, ngoài ra còn hỗ trợ trong việc cài đặt shared memory (bộ nhớ dùng chung).

**Câu 8** Nếu một tiến trình không có đủ các khung trang để chứa những trang cần thiết cho xử lý, thì nó sẽ thường xuyên phát sinh các lỗi trang , và vì thế phải dùng đến rất nhiều thời gian sử dụng CPU để thực hiện thay thế trang. Một hoạt động phân trang như thế được gọi là sự trì trệ ( thrashing)

Starvation: các tiến trình độ ưu tiên thấp có thể không bao giờ thực thi được

**Câu 1 :** Khi nào thì xảy ra tranh chấp race condition ?

**Race condition** là tình trạng nhiều process truy xuất và thao tác đồng thời lên dữ liệu chia sẻ. Kết quả cuối cùng của việc truy xuất đồng thời này phụ thuộc thứ tự thực thi của các lệnh thao tác dữ liệu.

Và vấn đề vùng tranh chấp (**Critical Section Problem**) là vấn đề về việc tìm một cách thiết kế một giao thức (một cách thức) nào đó để các process có thể phối hợp với nhau hoàn thành nhiệm vụ của nó.

**Câu 7:** Phát sinh deadlock: sử dụng tài nguyễn 0 thể chia sẻ + process chiếm giữ và yêu cầu thêm tìa nguyên + hệ thống ko thể thu hồi tài nguyên nếu process ko từ bỏ + tồn tại chu trình trong đồ thị cấp phát

Giải quyết : Ngăn deadlock: Không cho phép ít nhất một trong 4 điều kiện cần cho deadlock xảy ra. Hoạt động bằng cách giới hạn yêu cầu tài nguyên của process. Tránh deadlock: Các process cần cung cấp thông tin về tài nguyên nó cần, để hệ thống cấp phát tài nguyên một cách thích hợp. Chúng ta có thể cho phép hệ thống rơi vào trạng thái deadlock, kiểm tra deadlock và phục hồi hệ thống. Chúng ta có thể bỏ qua deadlock và xem như deadlock chưa từng xuất hiện (Windows/Linux). Tuy nhiên, deadlock không được phát hiện, về lâu dài sẽ dẫn đến việc giảm hiệu suất của hệ thống. Cuối cùng, hệ thống có thể ngưng hoạt động và phải khởi động lại.

**ngăn deadlock ?** Chúng ta ngăn deadlock bằng cách ngăn một trong 4 điều kiện cần của deadlock.

* Ngăn loại trừ tương hỗ (Mutual Exclusion) : Đối với tài nguyên không chia sẻ (printer) : Không làm được. Đối với tài nguyên chia sẻ (read-only file) : Không cần thiết.
* Ngăn giữ và chờ (Hold and Wait) : Cách 1 : Mỗi tiến trình yêu cầu toàn bộ tài nguyên cần thiết một lần. Nếu có đủ tài nguyên thì hệ thống sẽ cấp phát, nếu không đủ tài nguyên thì tiến trình phải bị block. Cách 2 : Khi yêu cầu tài nguyên, tiến trình không được giữ tài nguyên nào. Nếu đang có thì phải trả lại trước khi yêu cầu thêm.
* Ngăn không trưng dụng (no preemption) : Nếu tiến trình A có giữ tài nguyên và đang yêu cầu tài nguyên khác nhưng tài nguyên này chưa được cấp phát ngay thì : Cách 1 : Hệ thống lấy lại mọi tài nguyên mà A đang giữ. A chỉ bắt đầu lại được khi có được các tài nguyên đã bị lấy lại cùng với tài nguyên đang yêu cầu. Cách 2 : Hệ thống sẽ xem tài nguyên mà A yêu cầu : Nếu tài nguyên được giữ bởi một tiến trình khác đang đợi thêm tài nguyên, tài nguyên này được hệ thống lấy lại và cấp phát cho A. Nếu tài nguyên được giữ bởi tiến trình không đợi tài nguyên, A phải đợi và tài nguyên của A bị lấy lại. Tuy nhiên hệ thống chỉ lấy lại các tài nguyên mà tiến trình khác yêu cầu.
* Ngăn Circular Wait : Gán một thứ tự cho tất cả các tài nguyên trong hệ thống :

