+ Phân trang yêu cầu một lượng lớn thông tin ánh xạ. Bởi vì thông tin ánh xạ đó thường được lưu trữ trong bộ nhớ vật lý, việc phân trang theo lôgic yêu cầu tra cứu thêm bộ nhớ trước địa chỉ ảo do chương trình tạo r

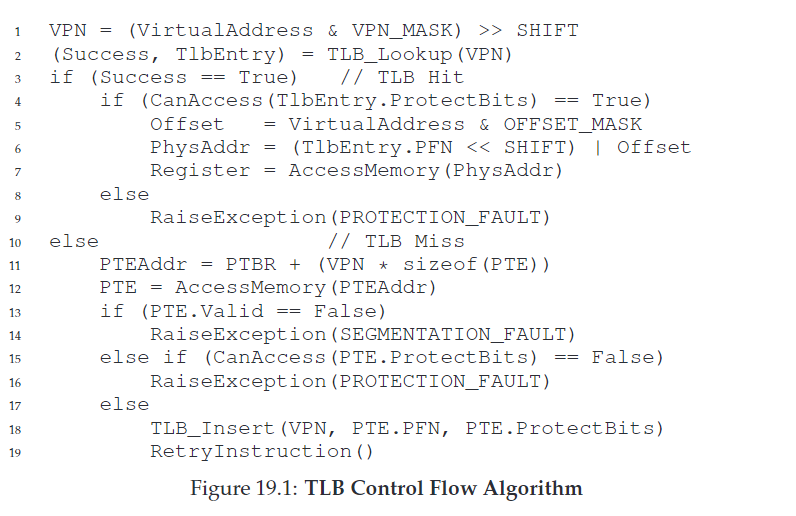
+ Tốc độ truy cập bộ nhớ chậm

* Hệ thống quản lý TLB là gì ?

+ TLB (translation-lookaside buffer) là 1 bộ quản lý bộ nhớ của chip (MMU), làm tăng tốc quá trình dịch địa chỉ ảo -> địa chỉ vật lý.

* Ánh xạ page

+ Trên mỗi tham chiếu địa chỉ ảo, phần cứng sẽ kiểm tra xem page number có nằm trong TLB hay chưa, Nếu page number nằm trong TLB (hit), lấy ngay đc số frame => tiết kiệm được tgian truy cập bộ nhớ chính để lấy số frame trong bảng trang. Ngược lại, vẫn phải truy cập vào bảng trang để lấy số frame bình thường, cập nhật TLB với bản dịch.

19.1 Thuật toán cơ bản của TLB

B1: Trích xuất số trang ảo từ địa chỉ ảo và kiểm tra TLB có chứa VPN đó không (1). Nếu có (2)(TLB hit), => PFN, kết hợp với offset => địa chỉ vật lý => truy cập bộ nhớ (5-7). Giả sử kiểm tra không thất bại (4).

B2: Nếu không tìm thấy (TLB miss), phần cứng truy cập bảng trang để lấy bản dịch (giả sử truy nhập thành công(13,15), cập nhật TLB với bản dịch (18)

19.2 Ví dụ

Hình 19.2, phần tử đầu tiên a[0] bắt đầu từ VPN = 6, offset = 4.

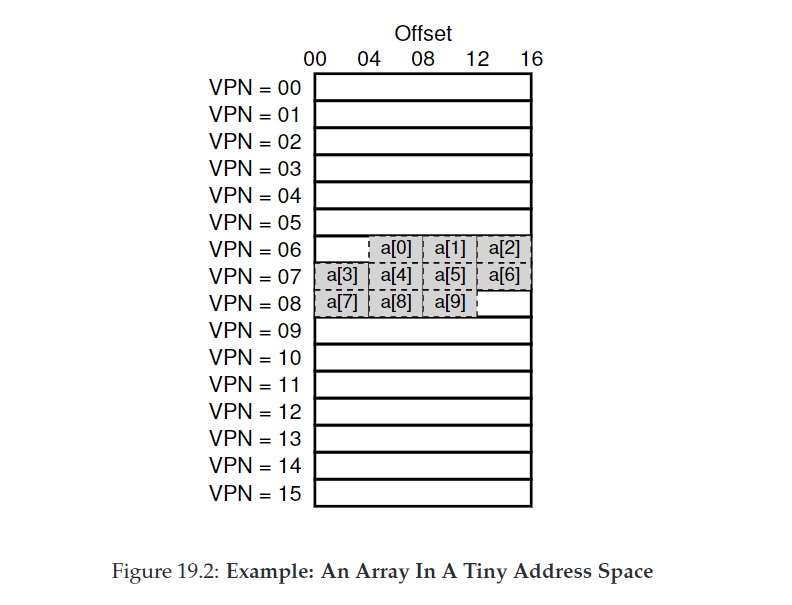
Xét vòng lặp trong C:

int sum = 0;

for (i = 0; i < 10; i++) {

sum += a[i];

}



* Khi phần tử đầu tiên a[0] được truy cập, CPU tìm ra được VPN = 6 và kiểm tra xem TLB có chứa bản dịch đó chưa. Kết quả lần kiểm tra đầu tiên sẽ trả về miss.
* Truy cập tiếp theo tại a[1] vẫn nằm tại VPN 06, lúc này khi kiểm tra TLB đã có sẵn bản dịch của VPN, do đó có thể trích xuất luôn được số khung và suy ra địa chỉ vật lý.
* Tương tự với các lần truy cập còn lại, ta có thứ tự TLB thực hiện: miss, hit, hit,miss, hit, hit, hit, miss, hit, hit.
* Ta thấy có 70% hit, tốc độ truy cập cũng được cải thiện.
* Đặc biệt, sau vòng lặp được thực hiện, nếu ta truy cập lại mảng một lần nữa, hiệu suất sẽ là 100%

19.3. Đơn vị thực hiện tlB miss.

Đơn vị nào thực hiện TLB miss: phần cứng hay HDH ?

* Thời gian đầu, phần cứng (có cấu trúc phức tạp) là đơn vị thực hiện TLB miss. Phần cứng cần biết được rõ vị trí của bảng trang trong bộ nhớ, trích xuất bản dịch, cập nhật TLB và thực hiện lại lệnh
* Kiến trúc máy tính hiện đại hơn: có phần mềm quản lý TLB. Khi TLB miss, phần cứng đưa ra một ngoại lệ dừng lệnh, chuyển về chế độ nhân và nhảy đến chế độ thực hiện bẫy lỗi. Mã chương trình bẫy lỗi được thực hiện bởi HDH, cập nhật lạiTLB và thực hiện lại lệnh.

19.4. Nội dung của TLB

- 1 TLB điển hình có thể có 32, 64 hoặc 128 mục.

- Bất kỳ bản dịch được cho nào cũng có thể nằm ở bất cứ đâu trong TLB, và do đó phần cứng có thể tìm kiếm bất cứ các mục của TLB một cách song song để tìm kiếm bản dịch mong muốn.

VPN | PFN | other bits

* VPN và PFN là 2 mục luôn có trong một bản dịch.
* Other bits:

+ A valid bit: Chứa thông tin liệu xem bản dịch có giá trị hay không.

+ Protection bits: xác định cách mà trang được truy cập.

+ ...

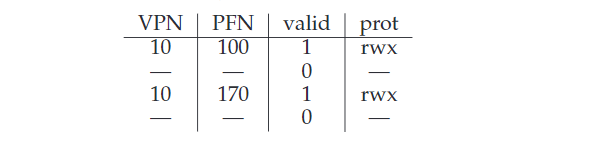
19.5. Vấn đề của TLB: Chuyển đổi ngữ cảnh

- Trong TLB, có một số vấn đề nảy sinh khi chuyển đổi giữa các tiến trình. Khi chuyển đổi giữa một tiến trình này với tiến trình khác, phần cứng và OS phải cẩn thận đảm bảo không sử dụng bản dịch của một số tiến trình đã chạy trước đó

- Ví dụ:

+ Khi một tiến trình (P1) đang chạy, giả sử TLB có chứa 1 bản dịch của P1 và có giá trị, được trích xuất từ bảng trang của P1. Giả sử VPN = 10, PFN = 100.

+ Giả sử có 1 tiến trình (P2) đang tồn tại, hệ điều hành muốn chạy P2, trang ảo thứ 10 của P2 ánh xạ tới khung 170, ta có bảng của TLB như sau:



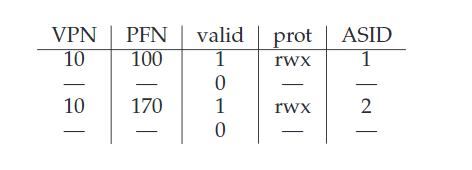
+ VPN 10 đều dịch tới cả PFN 100 (P1) và PFN 170 (P2), nhưng phần cứng không phân biệt được PFN nào có nghĩa cho tiến trình cần chạy.

* Giải pháp:

1. Giải phóng nội dung của TLB bằng cách set bit giá trị về 0.

* Giải quết được vấn đề dịch sai địa chỉ trong TLB, tuy nhiên mỗi khi tiến trình chạy, TLB lại quay lại trạng thái miss. Nếu hệ điều hành chuyển đổi giữa các tiến trình này nhiều lần, hiệu suất của TLB sẽ bị giảm đáng kể.

1. Sử dụng trường định danh không gian địa chỉ (ASID – Address space identifier)



Ta có thể thấy, các tiến trình có thể chia sẻ một cách rõ ràng: chỉ có trường ASID mới có thể phân biệt được các bản dịch giống nhau

19.6. Cách thức thay thế

- Câu hỏi: Nên thay thế mục nào trong khi có một mục mới cần được cập nhật vào TLB ?

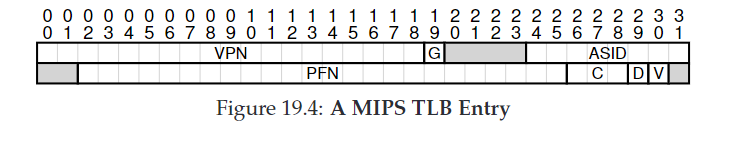
- 2 cách phổ biến:

+ Last recently used (LRU): Thay thế mục chưa sử dụng đến xa nhất trong TLB.

+ Thay thế ngẫu nhiên: hữu ích do tính chất đơn giản và có thể trách được các trường hợp góc. Ví dụ như 1 chương trình lặp lớn hơn n+1 trang với TLB cỡ n. Trong trường hợp này, LRU miss với mọi truy cập, trong khi thay thế ngẫu nhiên có thể làm tốt hơn.

19.7. Các mục của TLB trong thực tế

Hệ thống MIPS R4000[H93] được miêu tả như sau:



* MIPS hỗ trợ một không gian địa chỉ 32 bits với trang 4kB
* Chỉ có 19 bits đầu được sử dụng để lưu VPN, địa chỉ người dùng sẽ chỉ đến từ một nửa không gian địa chỉ (phần còn lại dành cho nhân) và vì thế chỉ cần 19 bits của VPN. VPN dịch tới số khung vật lý 24 bit, do đó có thể hỗ trợ bộ nhớ chính lên tới 64GB.
* Bit G (global): được sử dụng cho những trang được chia sẻ toàn cục giữa các tiến trình. Nếu G được set, ASID sẽ bị bỏ qua.
* ASID 8 bit: được HDH sử dụng để phân biệt giữa các không gian địa chỉ