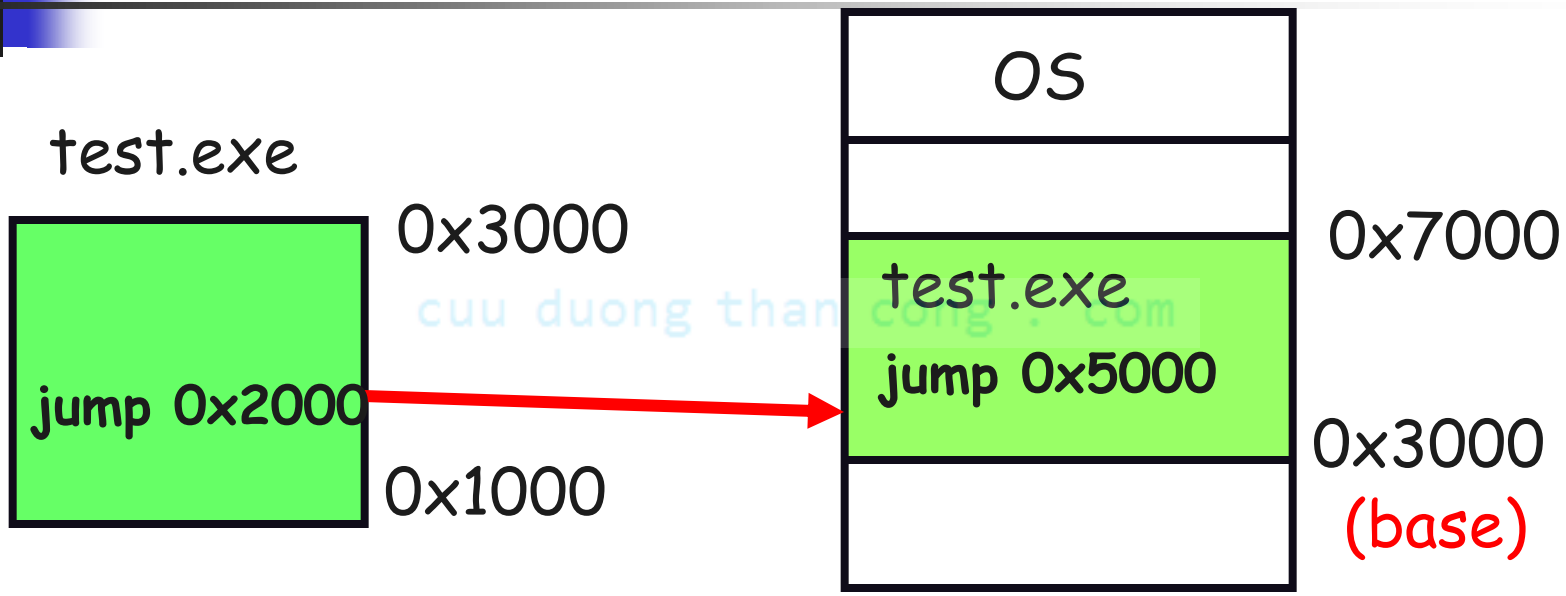
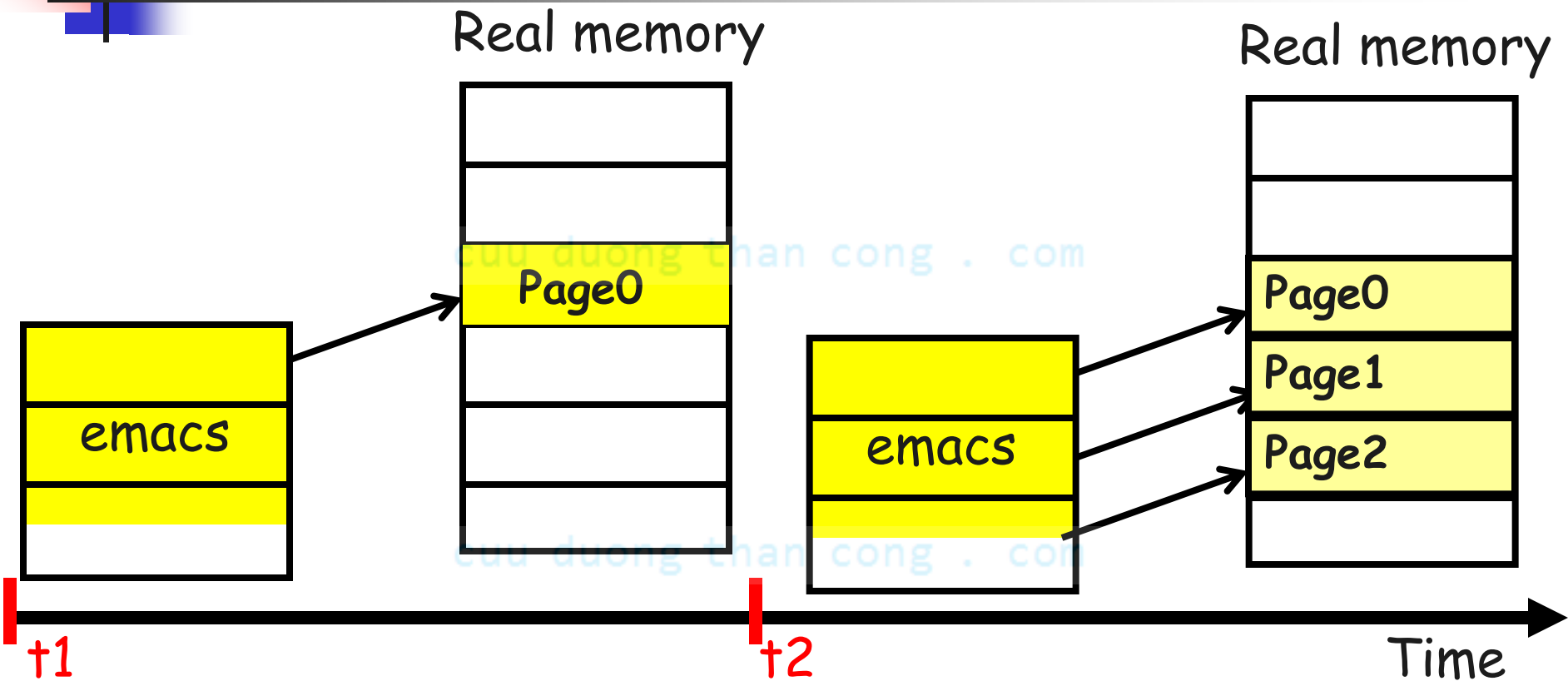
# BÀI 10: BỘ NHỚ ẢO




- Cho đến nay : Nạp toàn bộ tiến trình vào bộ nhớ rồi thực hiện nó...
  - Chậm, lãng phí bộ nhớ
  - Nếu kích thước tiến trình lớn hơn dung lương bộ nhớ chính ?
  - Lưu ý : tại 1 thời điểm chỉ có một chỉ thị được thực hiện

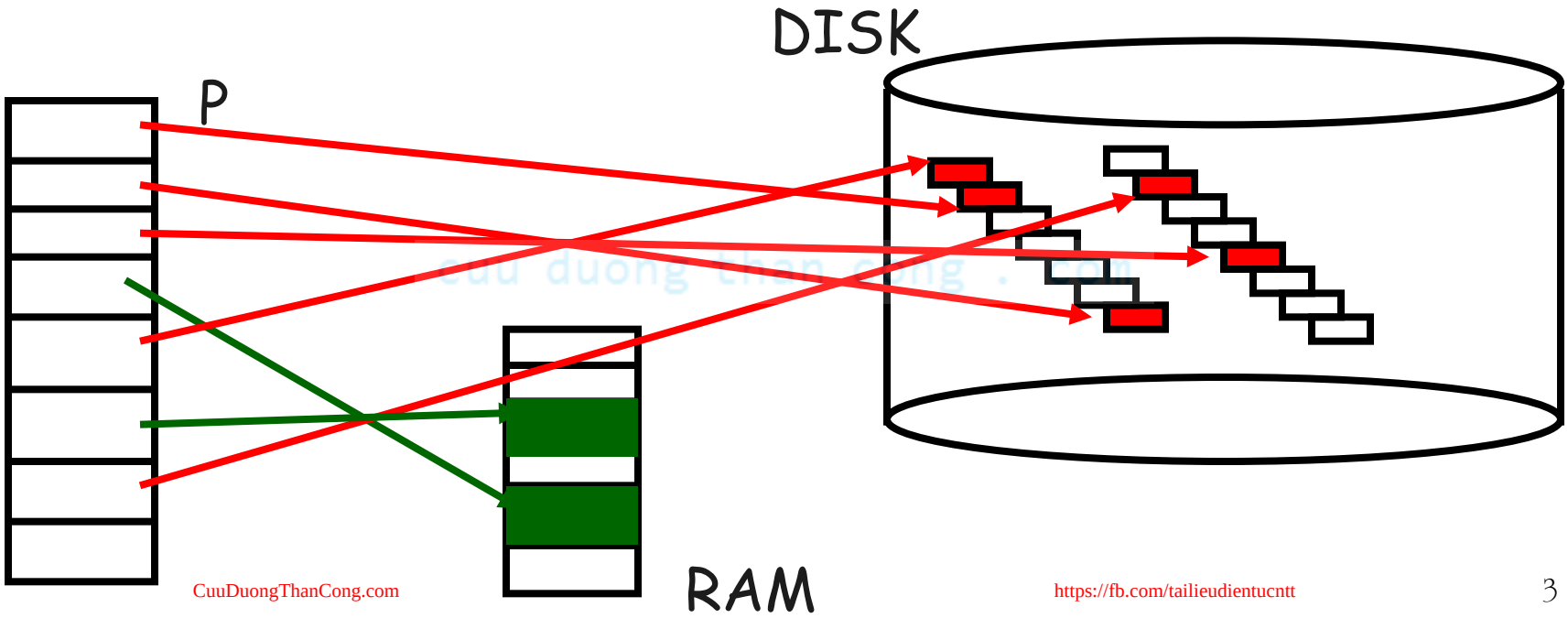
# Giải pháp




- Nạp từng phần chương trình khi cần thiết
- Demand paging

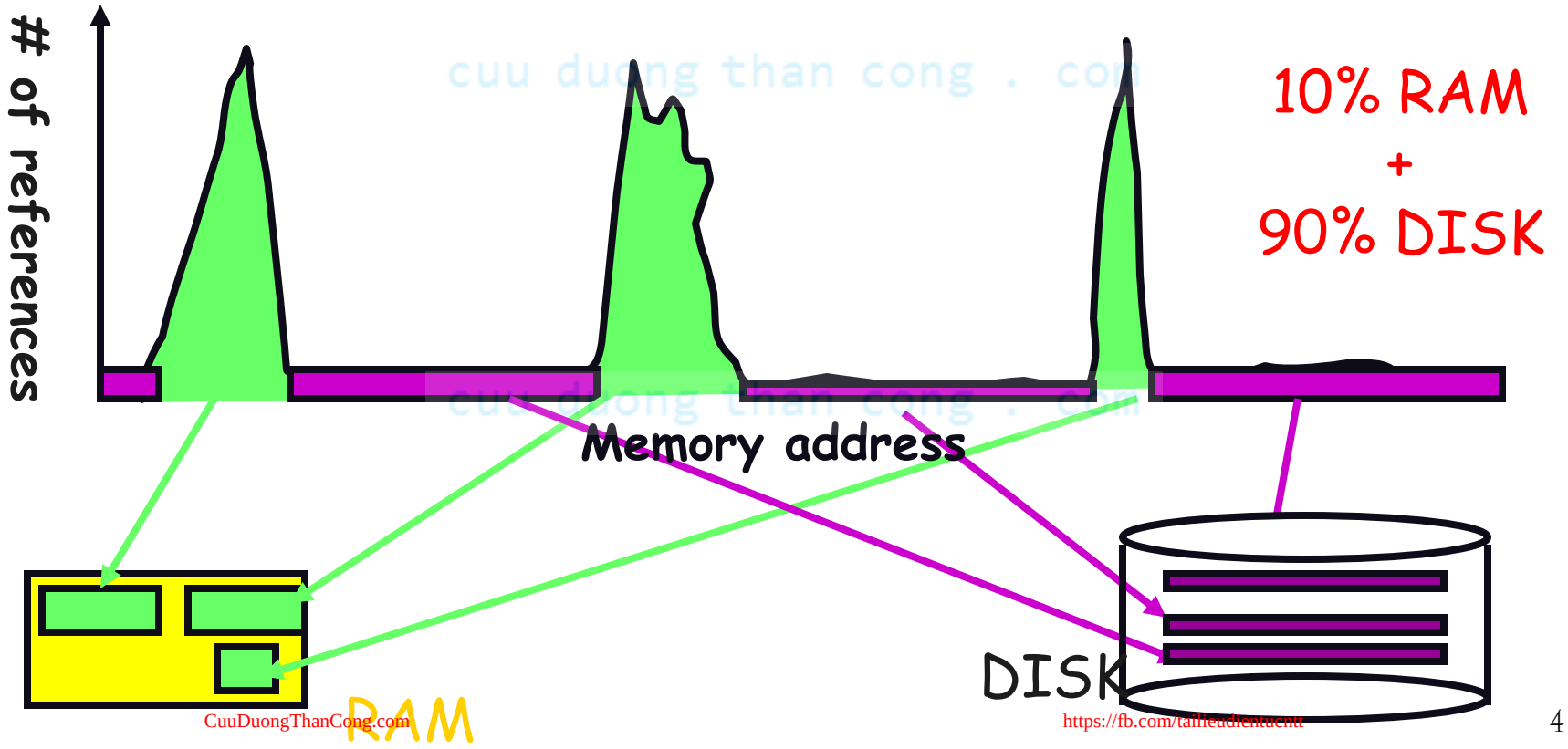
# Cơ chế

- Sử dụng bộ nhớ phụ để lưu trữ tạm thời các trang chưa sử dụng
- Ai chịu trách nhiệm chuyển đổi ?
  - Lập trình viên : Overlay
  - Hệ điều hành : Bộ nhớ ảo ( Virtual Memory )



# Bộ nhớ ảo = "lời nói dối vĩ đại"

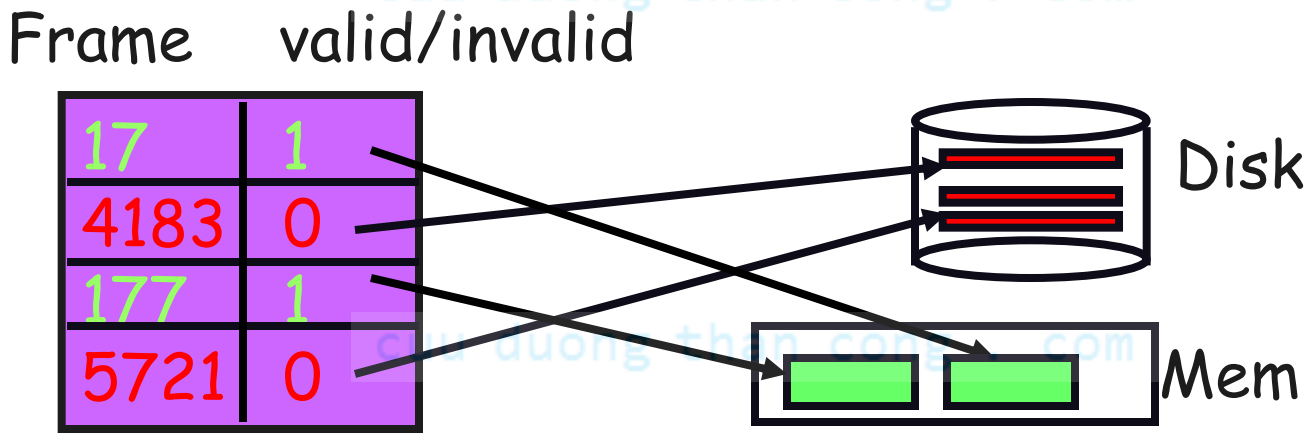
- Người dùng : sở hữu bộ nhớ "vô hạn", "riêng biệt"
- Hệ điều hành : "thầm lặng" thực hiện quá trình swapping



# Thực hiện Bộ nhớ ảo

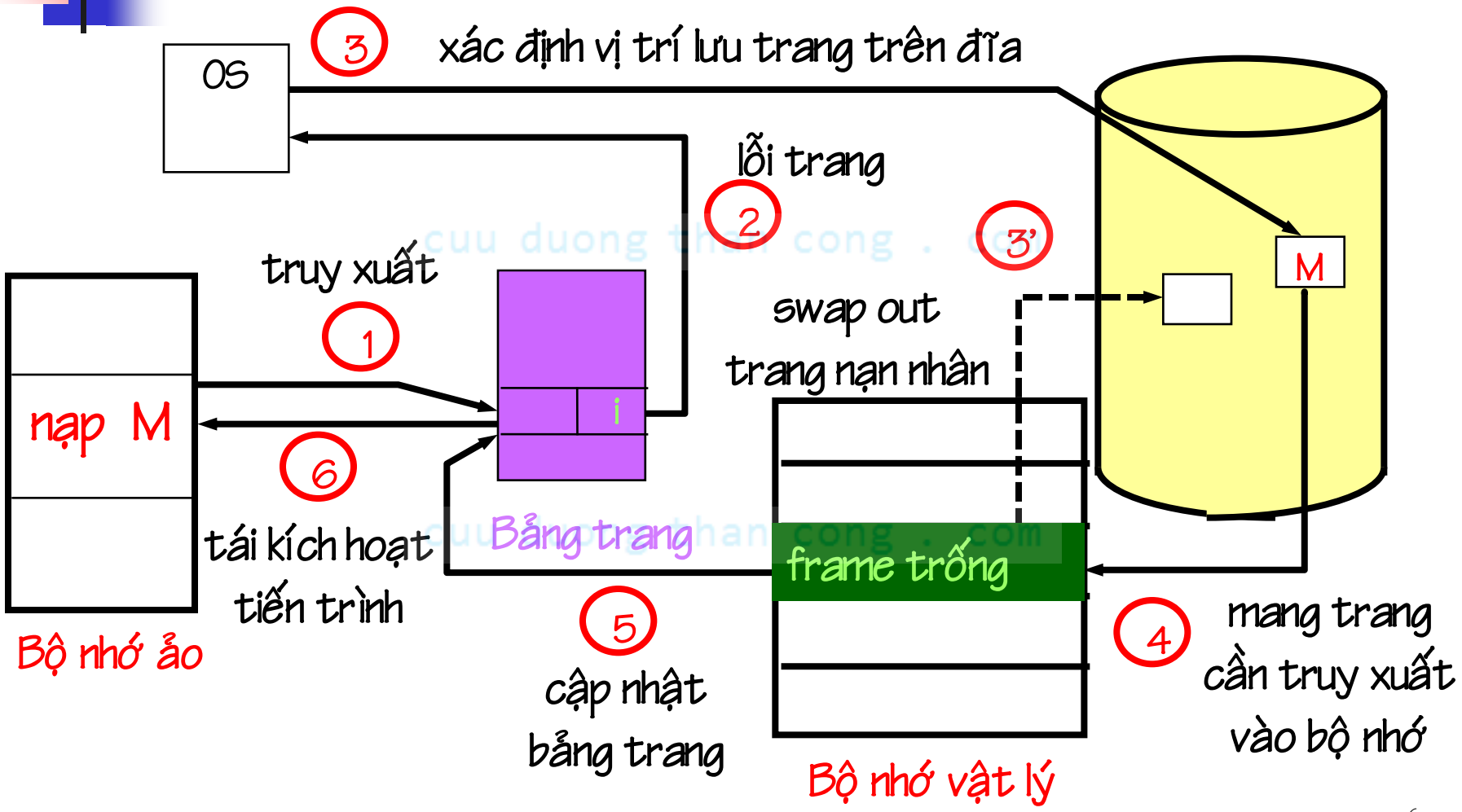
- Bảng trang : thêm 1 bit valid/invalid để nhận diện trang đã hay chưa được nạp vào RAM

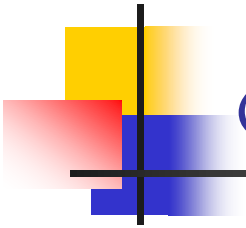





- Truy xuất đến một trang chưa được nạp vào bộ nhớ : lỗi trang (page fault)

# Xử lý lỗi trang





# Các câu hỏi

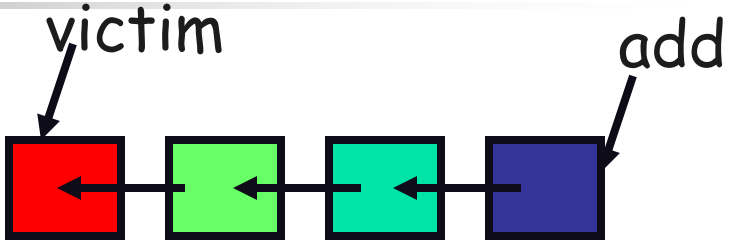
1. Chọn trang nạn nhân ? => Chiến lược thay thế trang

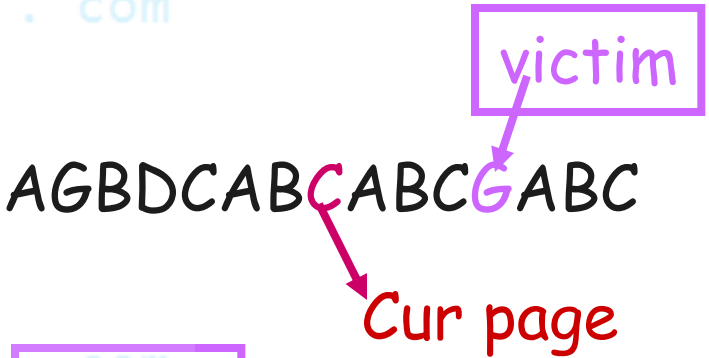


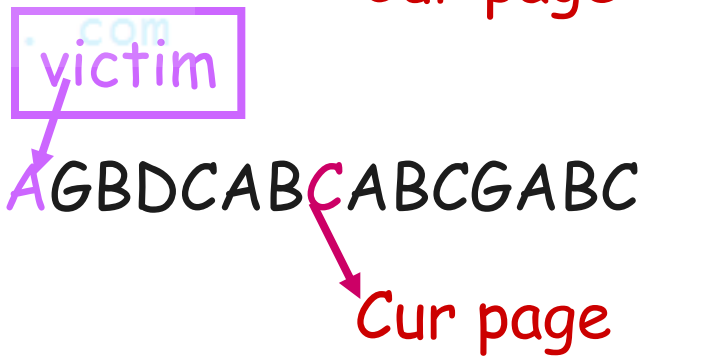
2. Chọn trang nào để nạp ? => Chiến lược nạp

# Chiến lược thay thế trang

- FIFO: trang "già" nhất
  - Công bằng ?
  - Không xét đến tính sủ dụng !
- TỐI ƯU : trang lâu sử dụng đến nhất trong tương lai
  - Tần suất lỗi trang thấp nhất
  - Không khả thi !
- LRU :trang lâu nhất chưa sử dụng đến trong quá khứ
  - Dự đoán tương lai LRU = MIN ?



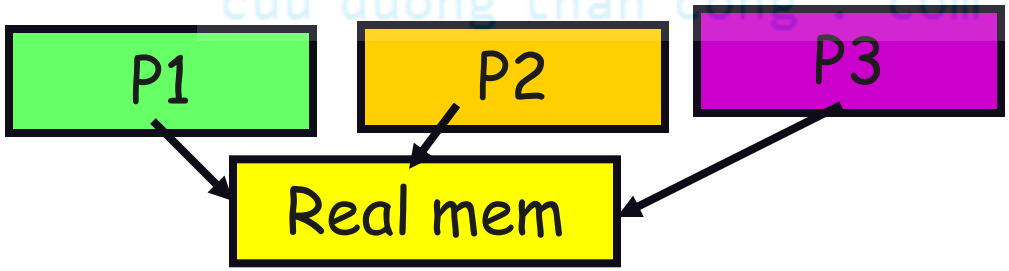
# Chiến lược nạp

- Demand paging : nạp trang được yêu cầu
  - Khi nào ?
  - Nạp sau : tần suất lỗi trang cao ? => pure demand paging
  - Nạp trước : làm sao biết ? =>prepaging


# Thrashing = ảo tưởng sụp đổ !

- Các tiến trình trong hệ thống yêu cầu bộ nhớ nhiều hơn khả năng cung cấp của hệ thống !








- Tất cả tiến trình đều bận rộn xử lý lỗi trang !
- IO hoạt động 100 %, CPU rảnh !
- Hệ thống ngừng trệ

# Nguyên nhân Thrashing

1. Tiến trình không tái sử dụng bộ nhớ (quá khứ != tương lai)
2. Tiến trình tái sử dụng bộ nhớ, nhưng với kích thước lớn hơn
3. Quá nhiều tiến trình trong hệ thống





- Chỉ có thể kiểm soát thrashing do nguyên nhân 3.

# Giải quyết thrasing với mô hình Working set

- Working set = tập hợp các trang tiến trình đang truy xuất tại 1 thời điểm.
- Hệ điều hành :
  - Chỉ nạp một tiến trình khi có đủ khung trang tự do cho working set của nó.
  - Kiểm soát mức độ đa chương của hệ thống : Nếu tổng số khung trang yêu cầu của các tiến trình trong hệ thống vượt quá các khung trang có thể sử dụng, chọn một tiến trình để tạm dừng, ngược lại, khi tổng working set bé hơn số khung trang tự do, nạp thêm tiến trình.