HỆ ĐIỀU HÀNH NHIỀU BỘ VI XỬ LÝ

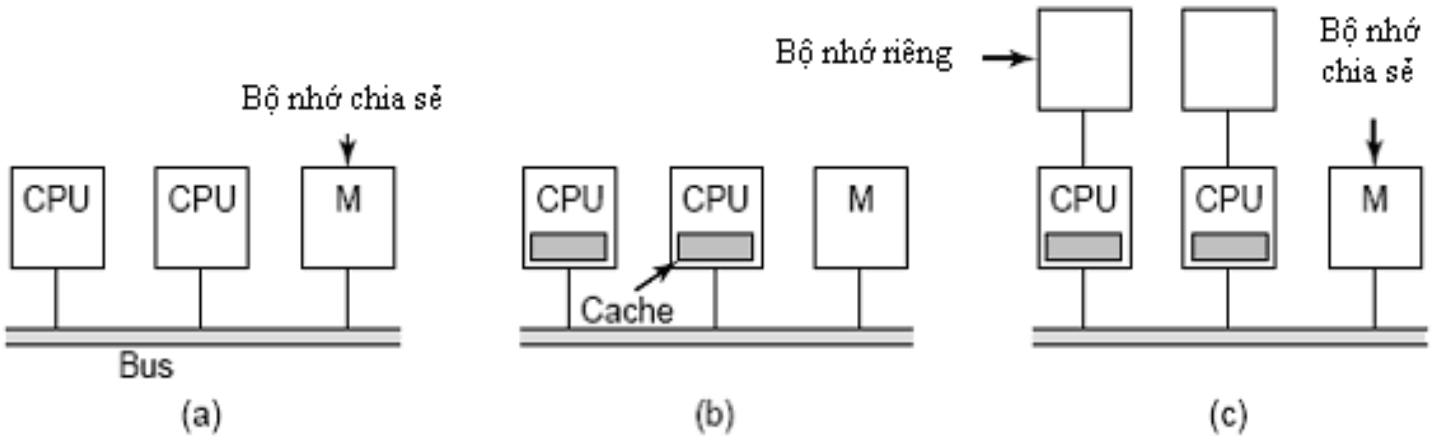
- Cấu hình nhiều processor
- Các loại hệ điều hành hỗ trợ nhiều bộ vi xử lý
- Đồng bộ trong hệ thống đa xử lý
- Điều phối trong hệ thống đa xử lý

CẤU HÌNH NHIỀU PROCESSOR

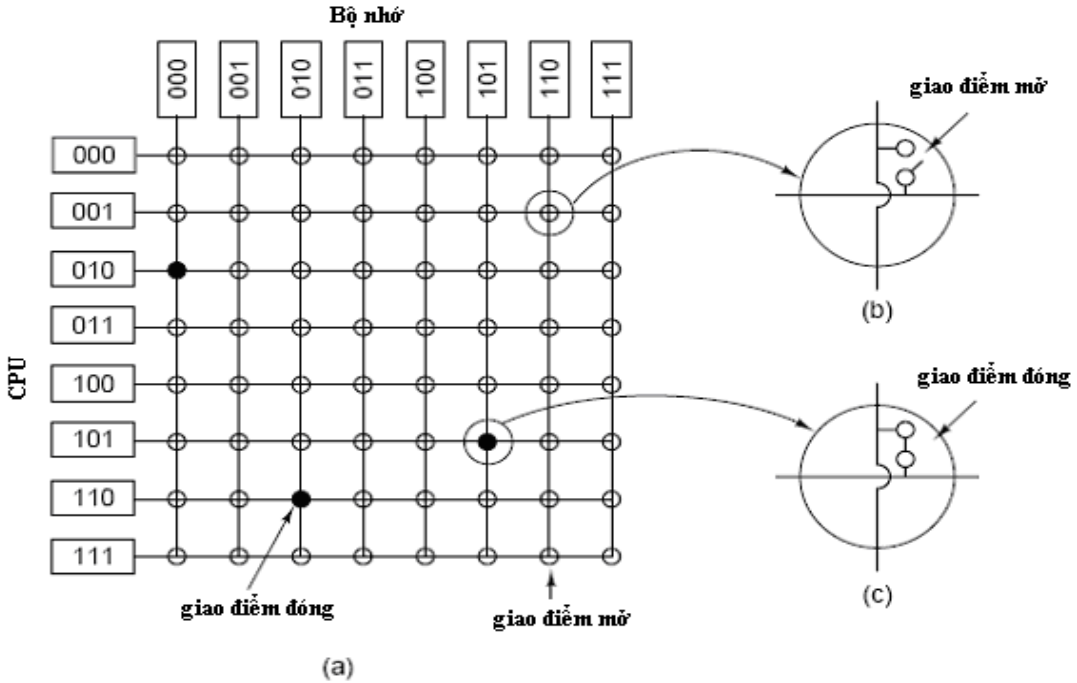
- Đa xử lý dùng bộ nhớ chia sẻ.
- Đa xử lý dùng bộ nhớ riêng.
- Đa xử lý phân tán

Hệ thống đa xử lý UMA dùng mô hình Bus

- 2 hoặc nhiều CPU và một hoặc nhiều bộ nhớ
- Sử dụng một tuyến *bus* để truyền thông.
- CPU phải kiểm tra xem *bus* có rỗi không.

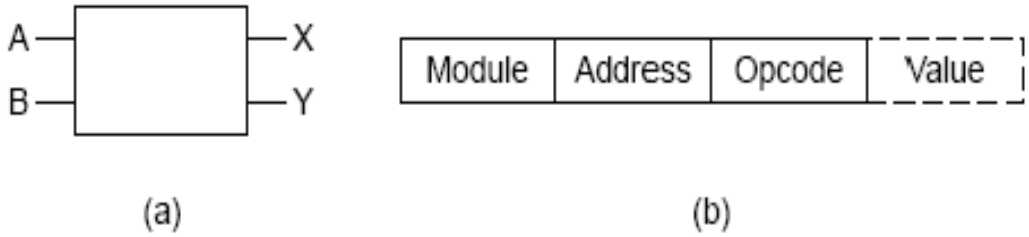


Hệ thống đa xử lý UMA dùng mô hình chuyển mạch chéo (Crossbar Switch)



Hệ thống đa xử lý UMA dùng mô hình mạng chuyển mạch đa tầng (Multistage Switching Network)

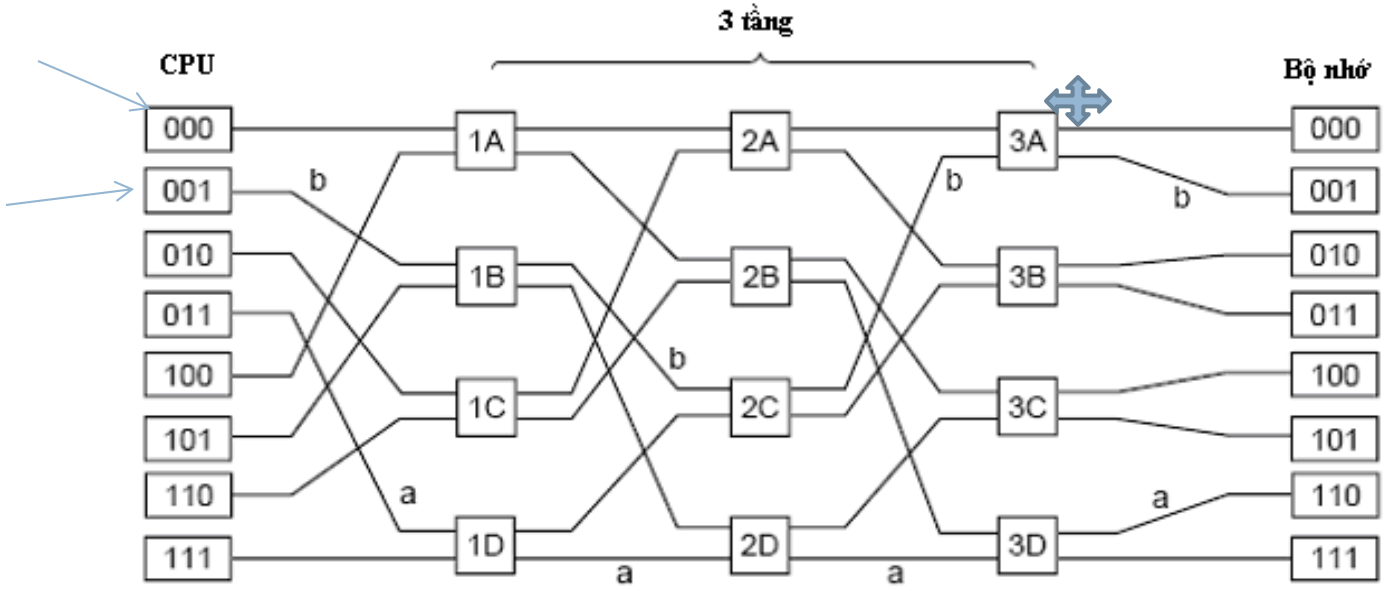
- *Module* : vùng nhớ nào được sử dụng.
- *Address* : địa chỉ nào trong vùng nhớ đó.
- *Opcode* : hoạt động gì sẽ được thực hiện (đọc/ghi).
- *Value* : toán hạng nào sẽ được dùng vào việc đọc/ghi



(a) Chuyển mạch 2x2. (b) Định dạng của *Message*

Multistage Switching Network (tt)

- Với *n* CPU và *n* bộ nhớ, chúng ta cần $log_2 n$ tầng (stage) với *n*/2 switch cho mỗi tầng.



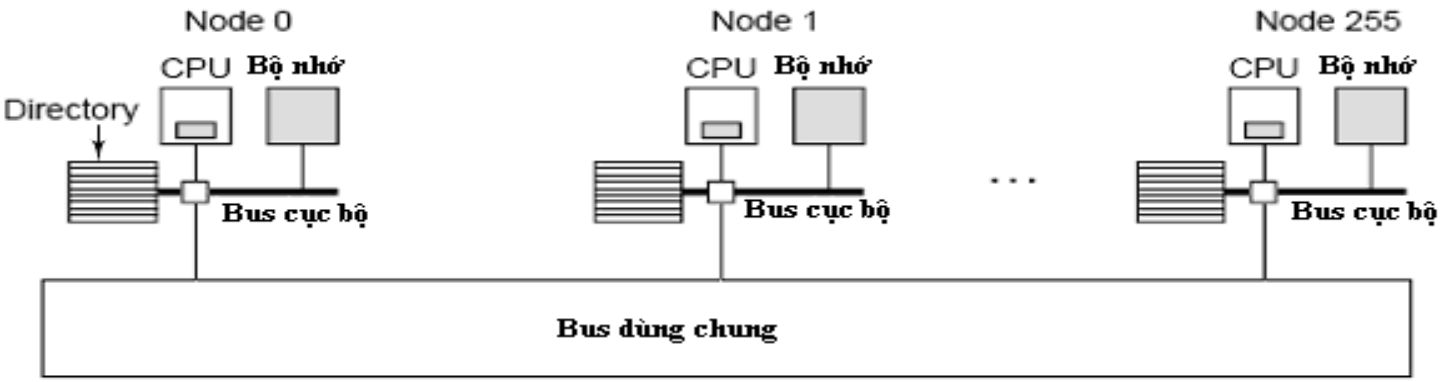
0: ngỏ ra phía trên
1: tuyến bên dưới

Hệ thống đa xử lý NUMA

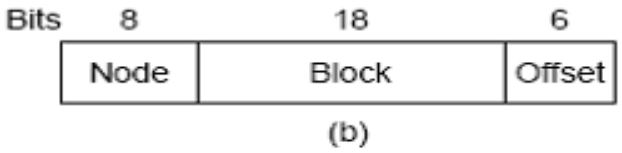
- Có một không gian địa chỉ duy nhất có thể nhìn thấy bởi tất cả các CPU.
- Truy xuất bộ nhớ ở xa thông qua hai lệnh LOAD và STORE.
- Truy xuất bộ nhớ ở xa chậm hơn truy xuất bộ nhớ cục bộ.

- ✓ NC-NUMA (NonCaching NUMA).
- ✓ CC-NUMA (Cache Coherent NUMA)

CC-NUMA

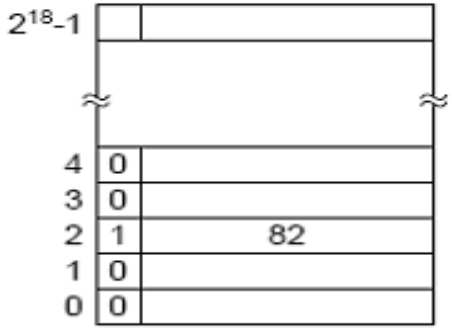


(a)



(b)

node 36, khối cache 4, offset 8



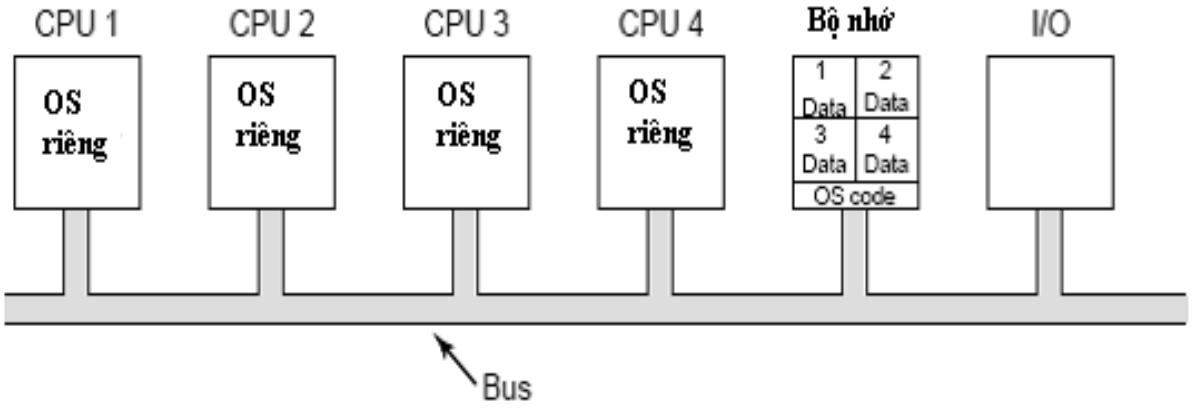
(c)

CÁC LOẠI HỆ ĐIỀU HÀNH HỖ TRỢ NHIỀU BỘ XỬ LÝ

- Mỗi CPU có riêng một hệ điều hành
- Hệ điều hành cho nhiều bộ xử lý hoạt động theo cơ chế Chủ-Tớ (Master-Slave)
- Hệ điều hành cho hệ thống có nhiều bộ xử lý đối xứng (Symmetric multiprocessors)

Mỗi CPU có riêng một hệ điều hành

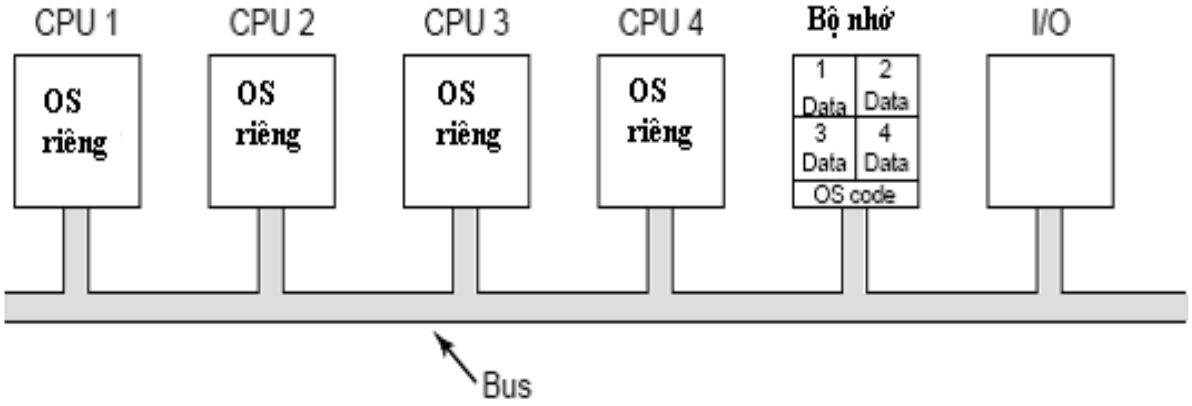
- Phân chia bộ nhớ cho các CPU trong hệ thống đa xử lý
- Cùng chia sẻ chung tập lệnh của hệ điều hành.
- Dữ liệu cũng được lưu trữ riêng cho từng CPU



✓Mỗi CPU được cấp một bộ nhớ riêng và sở hữu một bản sao riêng của hệ điều hành.

✓*n* CPU sau đó sẽ hoạt động như là *n* máy tính độc lập.

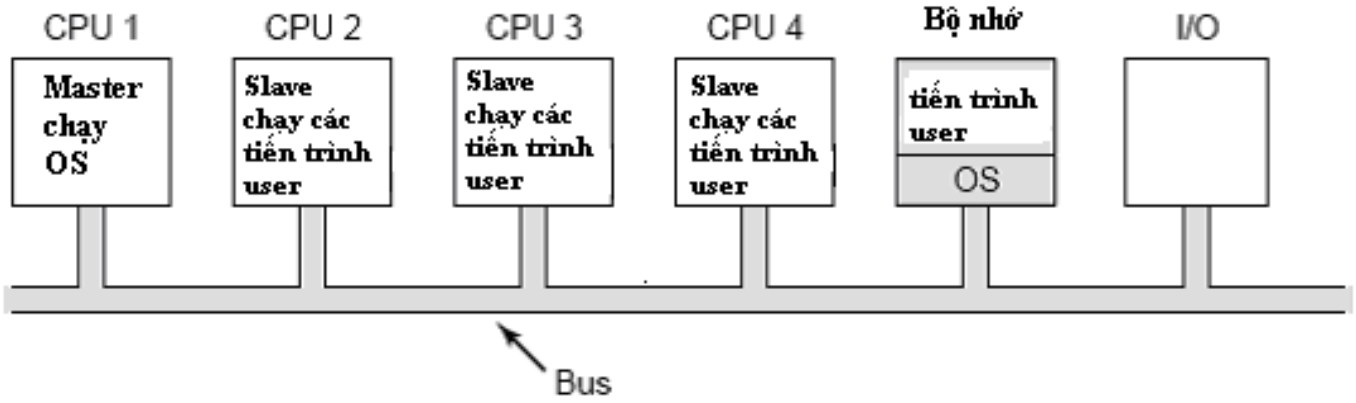
Mỗi CPU có riêng một hệ điều hành – Nhược điểm



- Không có việc chia sẻ tiến trình
- Không có việc chia sẻ tiến trình
- Mỗi hệ điều hành sẽ thao tác trên khối dữ liệu đĩa (cached) một cách độc lập với các hệ điều hành khác.

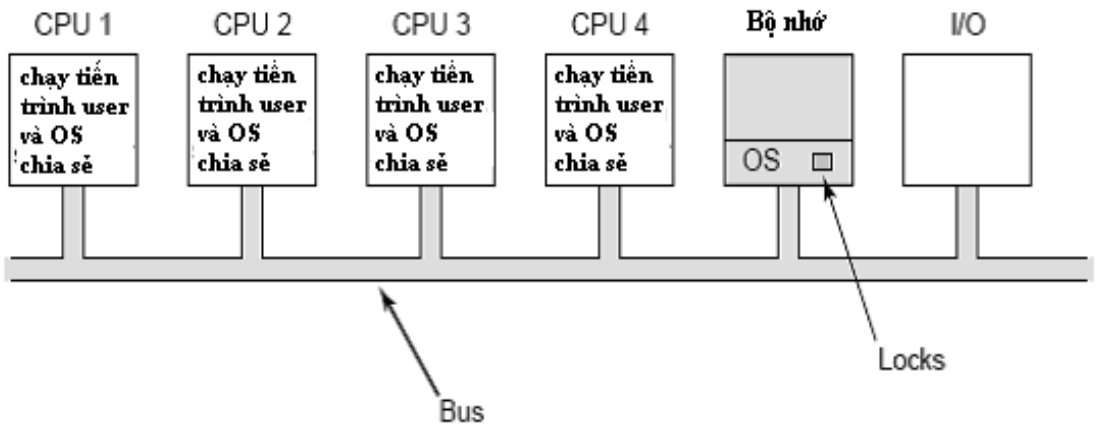
động theo cơ chế Chủ-Tớ (Master-Slave)

- Tất cả các lời gọi hệ thống đều được chuyển đến CPU 1 để được xử lý.
- Khi một CPU muốn đi vào trạng thái rỗi, nó sẽ yêu cầu HĐH gán cho nó một tiến trình để thực thi.
  - nếu được gán thì nó tiếp tục làm việc
  - nếu không nó mới đi vào trạng thái rỗi.



✓Nghẽn cổ chai tại CPU "chủ"

Hệ điều hành cho hệ thống có nhiều bộ xử lý đối xứng (Symmetric multiprocessors)

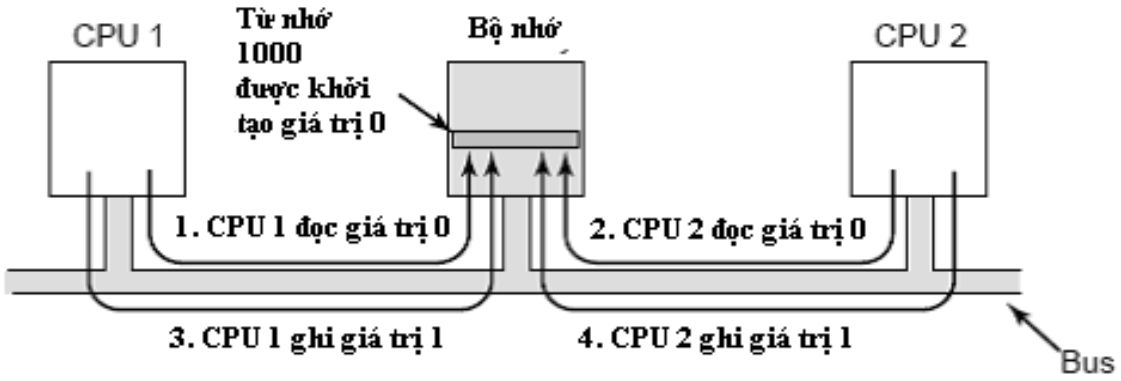


- Điều gì sẽ xảy ra nếu có hai CPU chọn cùng tiến trình để thực thi và yêu cầu cùng trang nhớ rỗi? -> Multex
- Dùng nhiều miền găng (Mỗi miền găng được bảo vệ bởi một biến *mutex* của nó)
- Deadlocks

Hầu hết các hệ thống có nhiều bộ xử lý đều sử dụng mô hình này

ĐỒNG BỘ TRONG HỆ THỐNG ĐA XỬ LÝ

- Chỉ thị lệnh TSL có thể bị lỗi nếu *Bus* không thể bị khóa.



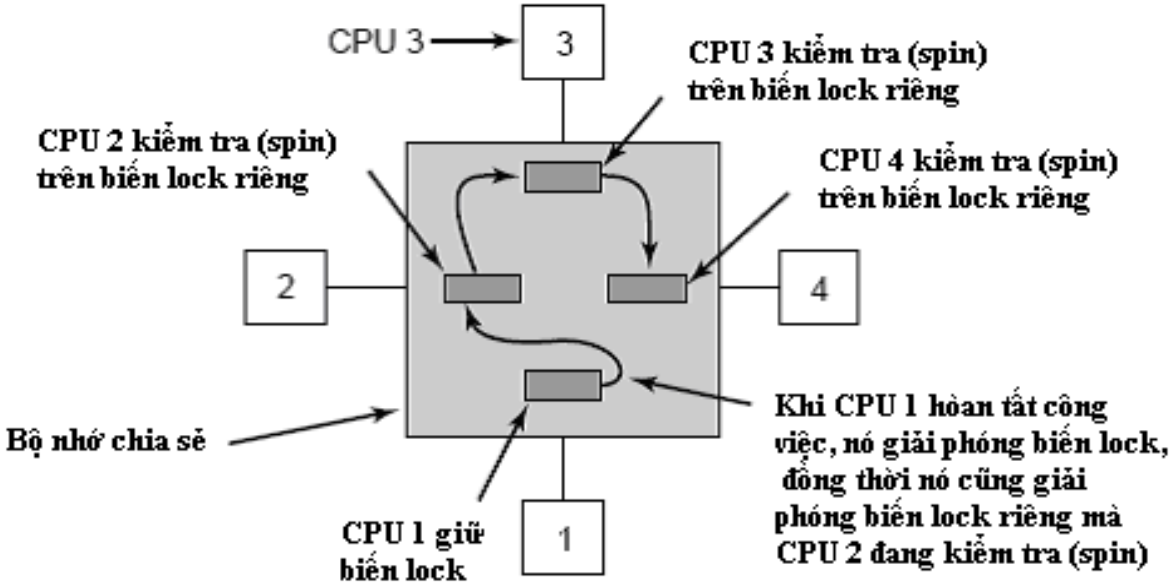
✓Chỉ thị TSL đầu tiên phải khóa *bus*, ngăn chặn các CPU khác truy xuất *bus*.
✓Cho phép cả hai CPU truy xuất đến bộ nhớ.

Hạn chế của TSL

- Biến *lock* sẽ bị chiếm dụng theo cơ chế quay tròn (spinning).
- Các CPU phải lần lượt thực hiện vòng lặp kiểm tra biến lock và chiếm dụng *bus* một cách chặt chẽ.
- Không chỉ làm lãng phí thời gian yêu cầu CPU mà còn gây ra một lượng tải lớn cho *bus* hoặc bộ nhớ, làm giảm nghiêm trọng tốc độ làm việc của các CPU khác.

Giảm lưu lượng bus - mỗi CPU chiếm lấy một *mutex* (biến lock riêng của nó)

- Sử dụng nhiều biến *lock* để tránh *cache thrashing*
- Giải quyết được vấn đề đói tài nguyên

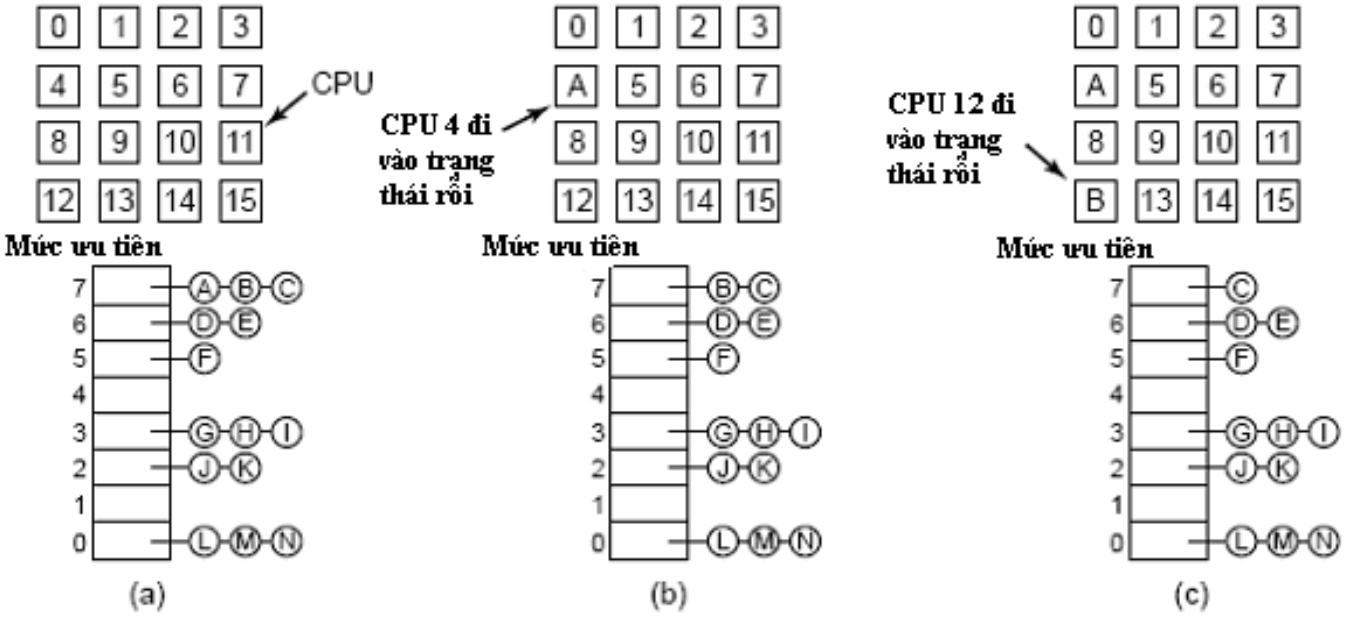


ĐIỀU PHỐI TRONG HỆ THỐNG ĐA XỬ LÝ

- Bộ điều phối phải quyết định tiến trình nào chạy với CPU nào -> điều phối 2 hướng.
- Các kỹ thuật điều phối:
  - Điều phối theo phương pháp chia sẻ thời gian (Time sharing)
  - Điều phối theo phương pháp chia sẻ không gian (Space Sharing)
  - Gang Scheduling

Điều phối theo phương pháp chia sẻ thời gian (Time sharing)

- Áp dụng cho các tiến trình độc lập nhau
- Dùng một cấu trúc dữ liệu cho toàn hệ thống



Time sharing – Ưu/nhược

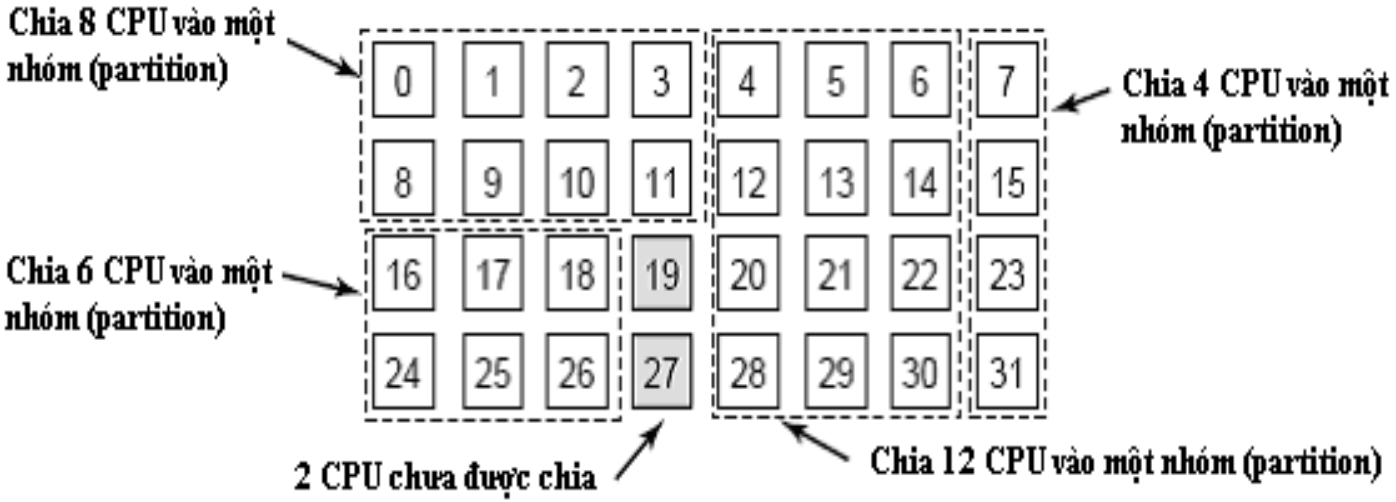
- Ưu : cân bằng tải -> không có trường hợp một CPU nào đó rỗi trong khi các CPU khác thì làm việc quá tải.
- Hạn chế:
  - Cạnh tranh cấu trúc dữ liệu điều phối khi số lượng CPU tăng lên
  - Lượng *overhead* lớn khi thực hiện chuyển đổi ngữ cảnh trong trường hợp một tiến trình bị khóa để đợi thao tác nhập xuất

Giải thuật điều phối hai mức

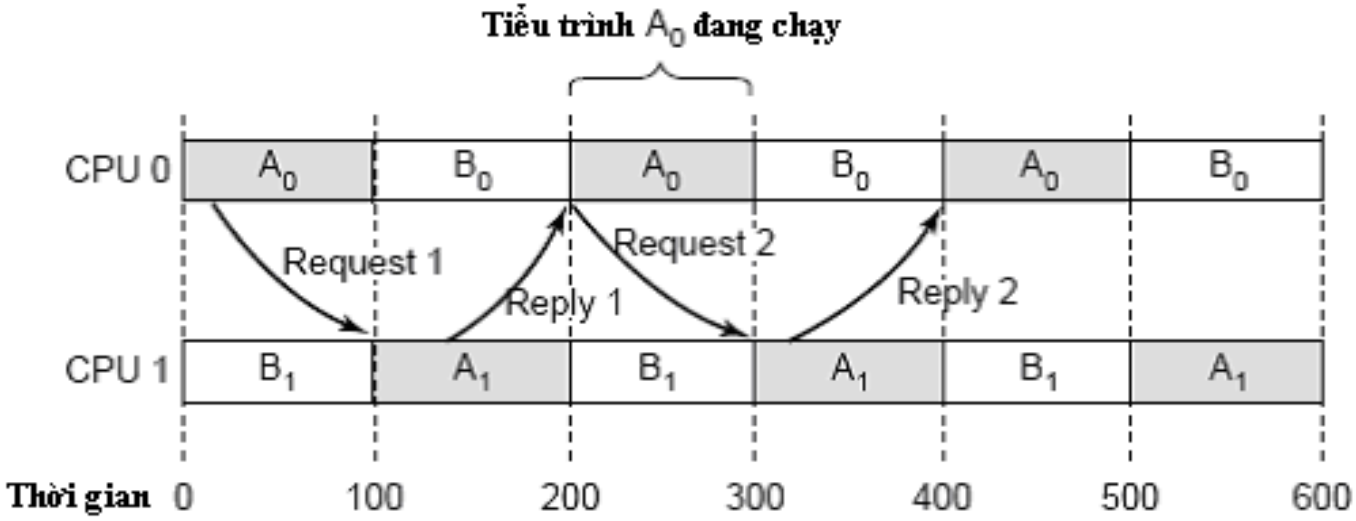
- Khi một tiến trình được tạo ra, nó được gán cho một CPU (được thực hiện ở mức cao)
  - Mỗi CPU chiếm giữ một tập các tiến trình của chính nó.
- Việc điều phối thực sự cho các tiến trình diễn ra ở mức thấp.
  - Điều phối được thực hiện bởi từng CPU riêng lẻ sử dụng các quyền ưu tiên hoặc dựa vào một vài yếu tố khác.
- Nếu một CPU không có tiến trình nào để thực thi, nó sẽ lấy một tiến trình nào đó từ một CPU khác hơn là phải đi vào trạng thái rỗi.
- Ưu điểm:
  - Phân bố tải đều trên các CPU hiện có.
  - *cache* được tận dụng tối đa (giữ một tiến trình trên cùng một CPU)
  - Việc tranh chấp danh sách sẵn sàng được giảm thiểu (mỗi CPU có một danh sách các tiến trình sẵn sàng của chính nó).

Điều phối theo phương pháp chia sẻ không gian (Space Sharing)

- Áp dụng cho các tiến trình có liên quan với nhau
- Các tiểu trình thuộc cùng một nhóm được tạo tại cùng một thời điểm.
- Nếu có nhiều CPU rỗi cho các tiểu trình này -> mỗi tiểu trình được cấp cho một CPU tương ứng và tất cả đều bắt đầu.
- Nếu không đủ CPU, không có tiểu trình nào được thực hiện.
- Mỗi tiểu trình sẽ nằm giữ CPU của nó cho đến khi kết thúc



Gang Scheduling



- Các tiểu trình liên quan được điều phối như là một nhóm.
- Tất cả các thành viên của một nhóm thực thi đồng thời, trên các CPU chia sẻ thời gian.
- Tất cả các thành viên bắt đầu và kết thúc các khe thời gian cùng nhau.

Gang Scheduling (tt)

- Tất cả các tiểu trình của một tiến trình chạy cùng nhau
- Nếu một trong chúng gửi yêu cầu cho một tiểu trình khác
  - nó sẽ nhận được gần như là lập tức và có thể hồi đáp trở lại cũng hầu như tức thời

Khe thời gian	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 5
0	A_0	A_1	A_2	A_3	A_4	A_5
1	B_0	B_1	B_2	C_0	C_1	C_2
2	D_0	D_1	D_2	D_3	D_4	E_0
3	E_1	E_2	E_3	E_4	E_5	E_6
4	A_0	A_1	A_2	A_3	A_4	A_5
5	B_0	B_1	B_2	C_0	C_1	C_2
6	D_0	D_1	D_2	D_3	D_4	E_0
7	E_1	E_2	E_3	E_4	E_5	E_6