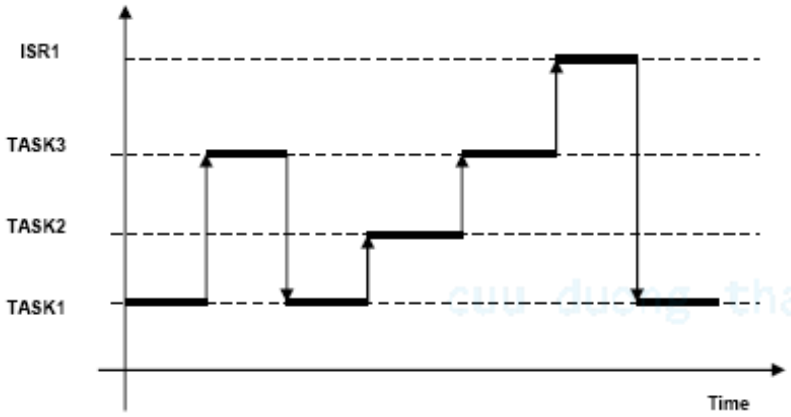
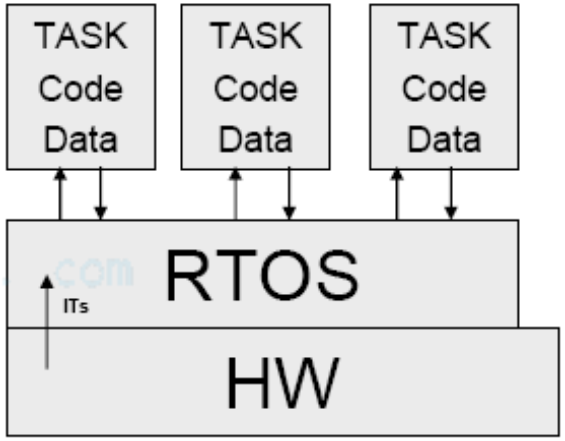
CHƯƠNG 5: RTOS – HỆ ĐIỀU HÀNH THỜI GIAN THỰC

Bài 9: RTOS và Kỹ thuật lập lịch

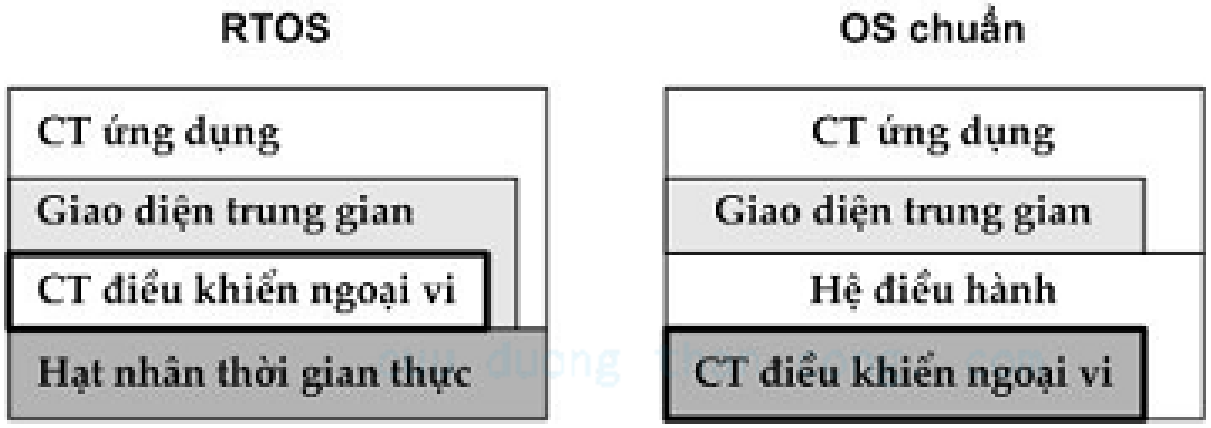
RTOS

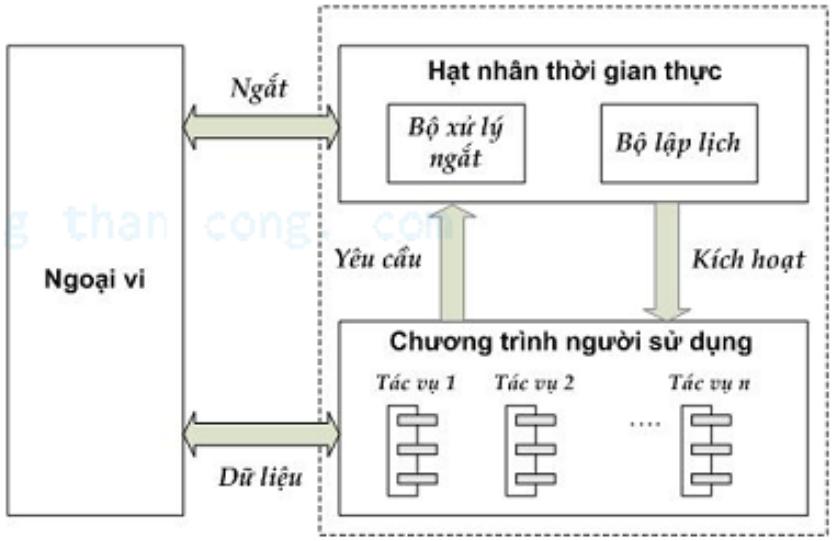
- Phần lõi (Kernel): Thực hiện việc lập lịch (schedules tasks)
- Tác vụ (Tasks): Là các hoạt động hiện tại với các trạng thái riêng của nó (PC, registers, stack, etc.)





RTOS





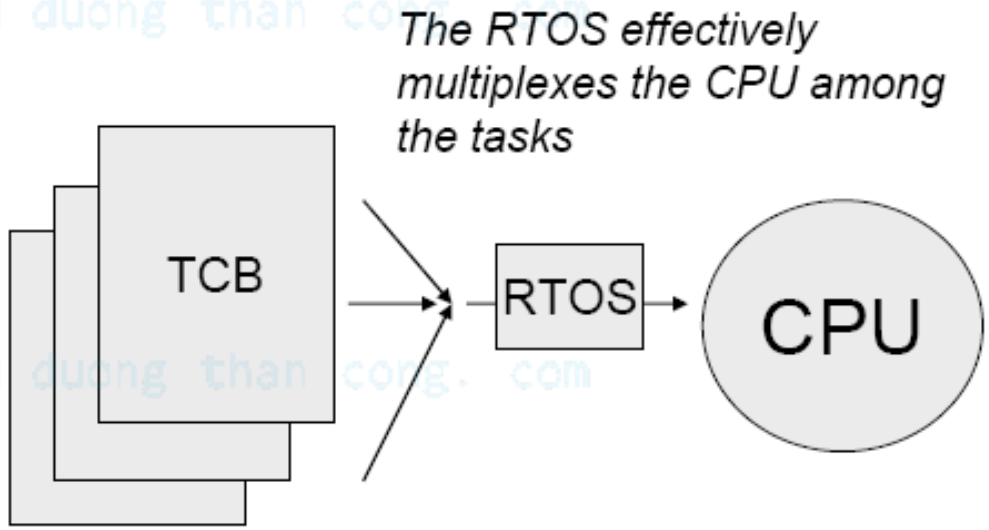
CÁC YÊU CẦU VỚI RTOS

- Kích thước nhỏ (lưu trữ toàn bộ trong ROM)
- Sử dụng hệ thống ngắt
- Không nhất thiết phải có các cơ chế bảo vệ
- Tăng tốc độ truyền thông giữa các quá trình
- Khi các quá trình ứng dụng đang thực hiện thì các yêu cầu hệ thống điều hành có thể được thực hiện thông qua các lời gọi hàm thay vì sử dụng cơ chế ngắt mềm

CÁC NHIỆM VỤ (Tasks)

- Các nhiệm vụ = Code + Data + State (trạng thái)
- Trạng thái của nhiệm vụ được lưu trữ trong khối điều khiển nhiệm vụ (Task Control Block - TCB) khi nhiệm vụ không được thực hiện trên CPU
- Một TCB điển hình:

ID
Priority
Status
Registers
Saved PC
Saved SP

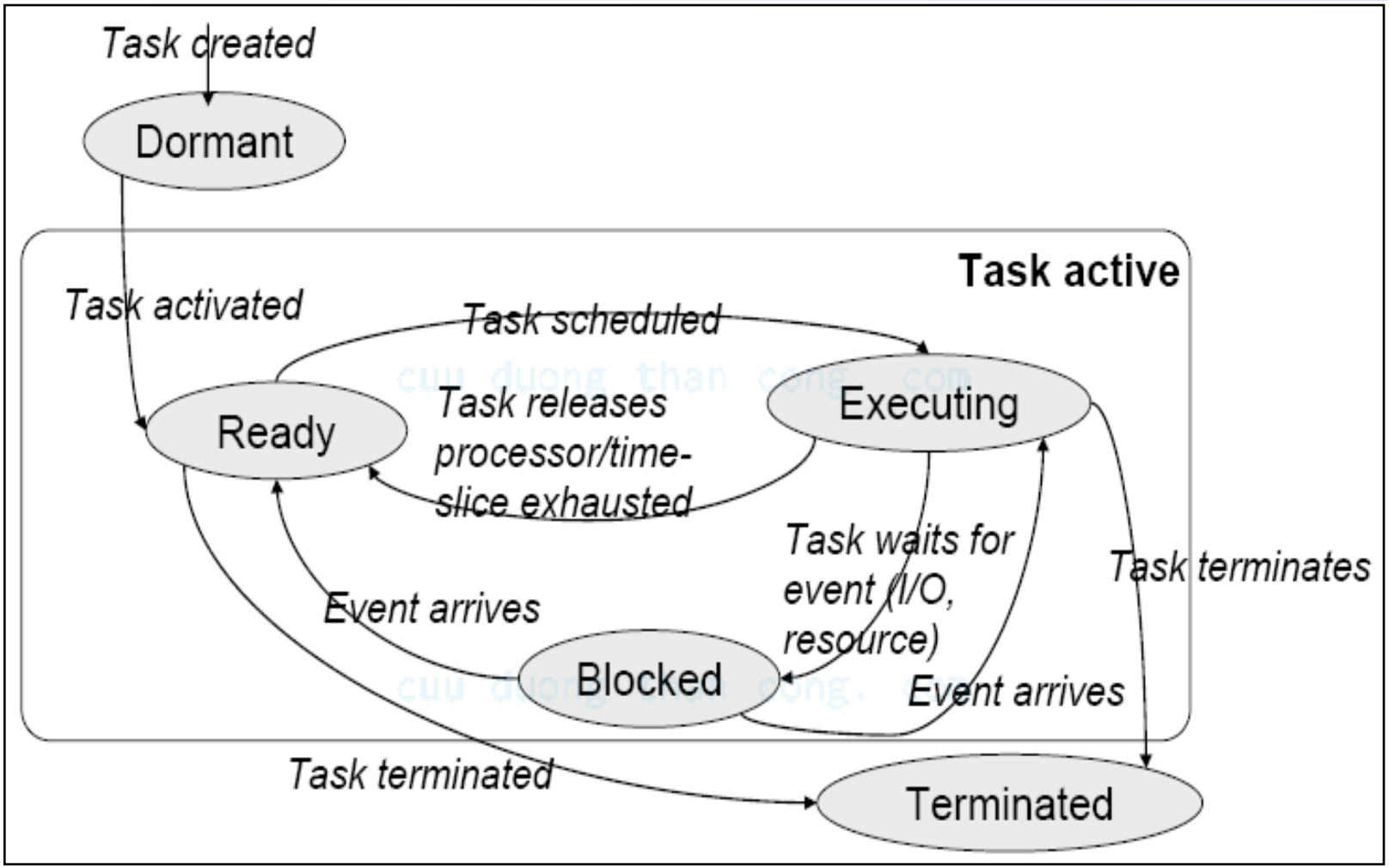


Các trạng thái của nhiệm vụ

- Executing: Đang thực hiện trên CPU
- Ready: Có thể chạy trên CPU nhưng một nhiệm vụ khác đang sử dụng CPU
- Blocked: Đợi sự kiện (I/O, signal, resource, etc.)
- Dormant: Tạo ra nhưng chưa được thực hiện
- Terminated: Không còn tác động nữa

RTOS thực hiện một cơ chế chuyển trạng thái cho mỗi nhiệm vụ và quản lý quá trình chuyển trạng thái.

Task States Transitions



Bộ lập lịch RTOS

- Thực hiện cơ chế chuyển trạng thái
- Chuyển giữa các nhiệm vụ
- Thuật toán chuyển:
  1. Lưu trữ trạng thái hiện tại vào TCB
  2. Tìm TCB mới
  3. Khôi phục trạng thái từ TCB mới
  4. Tiếp tục
- Chuyển đổi giữa trạng thái EXECUTING -> READY:
  1. Nhiệm vụ được thực hiện tuần tự: **NON-PREEMPTIVE**
  2. RTOS chuyển trạng thái cho các nhiệm vụ ưu tiên cao hơn: **PREEMPTIVE**

Quá trình lập lịch

Mục đích: Đảm bảo yêu cầu về thời gian

- Lập lịch trước khi chạy (*static*): Xác định chính xác giản đồ thời gian cho các nhiệm vụ tại thời điểm thiết kế
- Lập lịch khi chạy chương trình (*dynamic*): Lập lịch được thực hiện tự động bởi RTOS, dựa trên sự ưu tiên.

Phân bố các nhiệm vụ (1)

Một chu kỳ của nhiệm vụ (Đối với các nhiệm vụ có chu kỳ - periodic tasks):

• Các ràng buộc ưu tiên: Xét xem có bất cứ nhiệm vụ nào cần được ưu tiên không.

• Thời gian xuất hiện-*ai (arrival time)*: Là khoảng thời gian khi sự kiện xảy ra và nhiệm vụ tương ứng được kích hoạt.

Thời điểm bắt đầu thực thi *ri* (*release time*): Thời điểm sớm nhất khi việc xử lý đã sẵn sàng và có thể bắt đầu.

• Thời điểm bắt đầu thực hiện - *si* (*starting time*): Là thời điểm mà tại đó tác vụ bắt đầu việc thực hiện của mình.

• Thời gian tính toán/thực thi - *ci* (*Computation time*): Là khoảng thời gian cần thiết để bộ xử lý thực hiện xong nhiệm vụ của mình mà không bị ngắt.

• Thời điểm hoàn thành - *fi* (*finishing time*): Là thời điểm mà tại đó tác vụ hoàn thành việc thực hiện của mình.

• Thời gian rủi ro/xấu nhất - *wi* (*worst case time*): khoảng thời gian thực hiện lâu nhất có thể xảy ra.

• Thời điểm kết thúc - *di* (*due time*): Thời điểm mà tác vụ phải hoàn thành.

Model nhiệm vụ

Mô hình nhiệm vụ đơn giản:

- Tất cả các nhiệm vụ phải có yêu cầu khắt khe về chu kỳ thực hiện.
- Thời gian để hoàn thành nhiệm vụ chính là chu kỳ của nhiệm vụ.
- Tất cả các nhiệm vụ độc lập – không có ràng buộc về quyền ưu tiên.
- Không có nhiệm vụ nào có bộ phận không ưu tiên
- Chỉ có nhiệm vụ xử lý.

Các kỹ thuật lập lịch

1. **First Come First Serve (FCFS)**
- Các quá tình được xử lý theo thứ tự mà nó xuất hiện yêu cầu và cho đến khi hoàn thành
- Cơ chế lập lịch này thuộc loại không ngắt được và có ưu điểm là dễ dàng thực thi
- Không phù hợp cho hoạt động đáp ứng thời gian thực

2. **Shortest Job First (SJF)**
- Tác vụ có thời gian thực thi ngắn nhất sẽ có quyền ưu tiên cao nhất và sẽ được phục vụ trước
- Không biết trước được thời gian thực thi của các tác vụ tham gia trong chương trình và thông thường phải áp dụng cơ chế tiên đoán và đánh giá dựa vào kinh nghiệm về các tác vụ thực thi trong hệ thống
- Có thể áp dụng cho các tác vụ cả loại ngắt được và không ngắt được

Các kỹ thuật lập lịch

3. Rate Monotonic (RM)

- Phương pháp này dựa trên một số giả thiết sau:
  - Tất cả các tác vụ tham gia hệ thống phải có *deadline* kiểu chu kỳ
  - Tất cả các tác vụ độc lập với nhau
  - Thời gian thực hiện của các tác vụ biết trước và không đổi
  - Thời gian chuyển đổi ngữ cảnh thực hiện là rất nhỏ và có thể bỏ qua
- Thuật toán RM được thực thi theo nguyên lý ***gán mức ưu tiên cho các tác vụ dựa trên chu kỳ của chúng*** (chu kỳ nhỏ thì mức ưu tiên cao)
- Với các tác vụ chu kỳ không thay đổi thì RM sẽ là phương pháp lập lịch cho phép ngắt và mức ưu tiên cố định

Các kỹ thuật lập lịch

3. Earliest Deadline First (EDF)

- Sử dụng *deadline* của tác vụ như điều kiện ưu tiên để xử lý điều phối hoạt động
- Tác vụ có *deadline* gần nhất sẽ có mức ưu tiên cao nhất và các tác vụ có *deadline* xa nhất sẽ nhận mức ưu tiên thấp nhất

4. Minimum Laxity First (MLF)

- Cơ chế lập lịch này sẽ ưu tiên tác vụ nào còn ít thời gian còn lại để thực hiện nhất trước khi nó phải kết thúc để đảm bảo yêu cầu thực thi đúng
- Cơ chế lập lịch gán quyền ưu tiên động và dễ đạt được sự tối ưu về hiệu suất thực hiện và sự công bằng trong hệ thống

Các kỹ thuật lập lịch

5. Round Robin (RR)

- Mỗi một tác vụ được xử lý/phục vụ trong một khoảng thời gian nhất định và lặp lại theo một chu trình xuyên suốt toàn bộ các tác vụ tham gia trong hệ thống
- Khoảng thời gian phục vụ cho mỗi tác vụ trong quá trình là một sự thoả hiệp giữa thời gian thực hiện của các tác vụ và thời gian thực hiện một chu trình