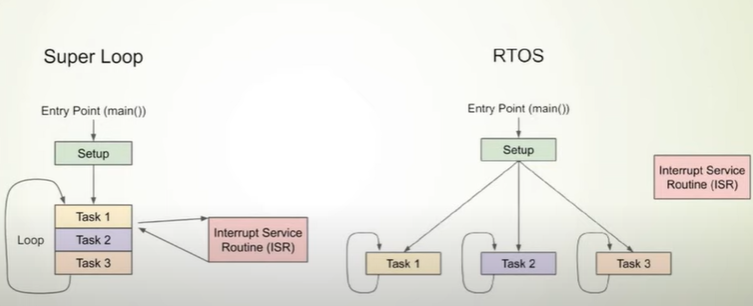
RTOS (Real Time Operating System)

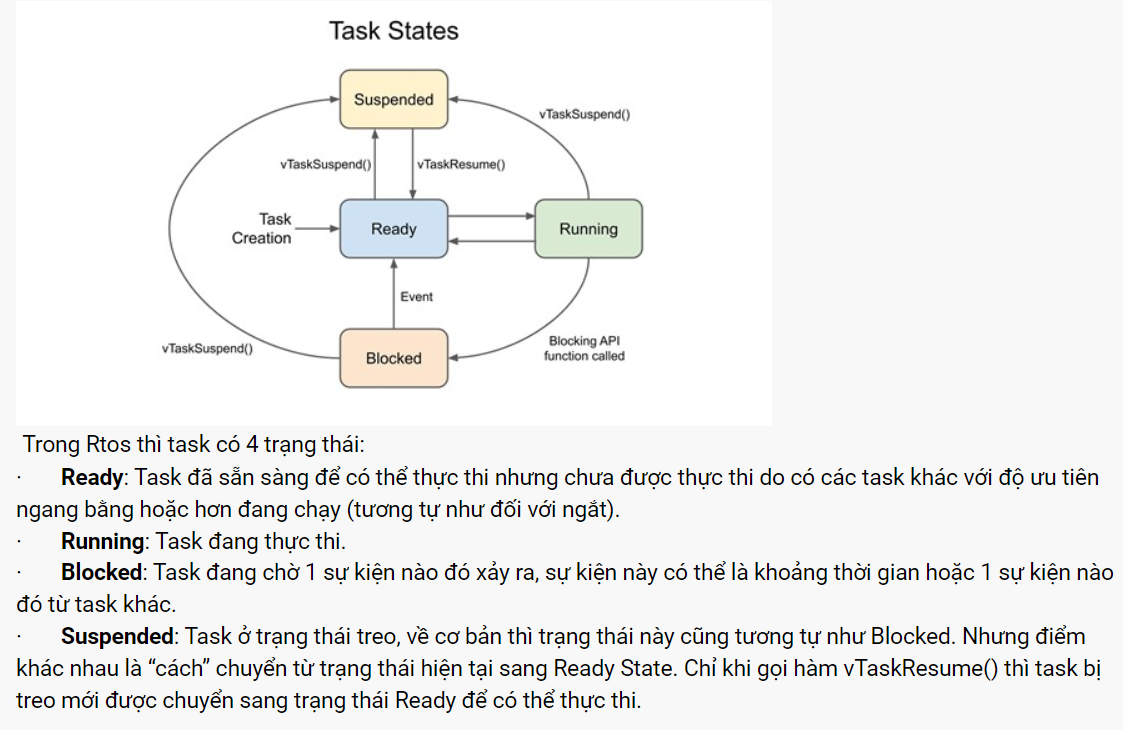
- Cho phép ứng dụng chạy đa tác vụ

- Đáp ứng được deadline theo thời gian thực -> dùng cho các ngành đòi hỏi cao về đáp ứng thời gian như Viễn thông, IOT, Thiết bị y tế,…

- Chia nhỏ project để phân công dự án dễ dàng và cụ thể hơn



- Có thể hiểu Task như Thread nhưng thực tế không phải.



- Mỗi 1 Task trong RTOS được gồm:

+ Tên task

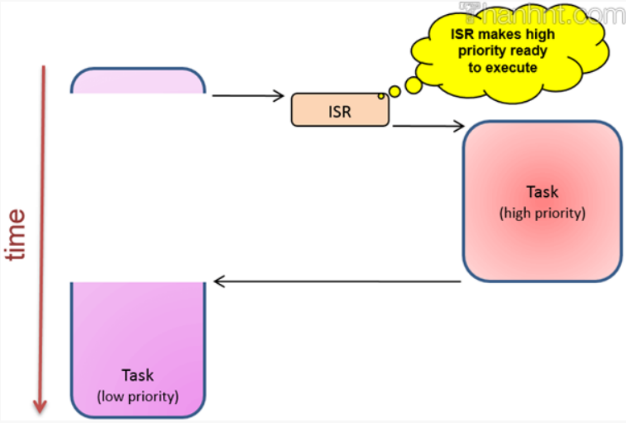
+ Hàm task (Viết một hàm cho task như kiểu viết hàm cho Interreupt trong hệ thống Loop)

+ Mức độ ưu tiên của task (như kiểu đặt mức độ ưu tiên cho Interreupt trong hệ thống Loop)

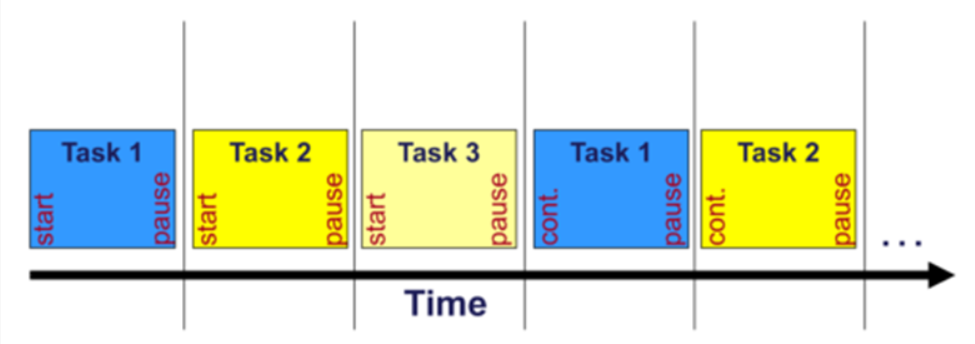
+ Tham số truyền vào (thường là 0)

+ Kích thước Task (Task được tạo ra ở vùng nhớ Heap -> mỗi Task phải tự while(1), nếu không thì khi kết thúc Task phải xóa Task )

- Chính vì một số đặt tính trên của Task nên ta có thể vận hành các task theo 3 cách chính:

 + Cách 1: Từng Task sẽ thực hiện hết chức năng của mình rồi mới qua Task khác và cứ tuần tự như vậy (vận hành như hệ thống Loop bình thường)

+ Cách 2: Vận hành các Task theo mức độ ưu tiên của Task (Task nào có mức độ ưu tiên cao hơn thì chạy trước, như kiểu Interrupt trong Loop)

 + Cách 3: Vận hành theo kiểu chia thời gian (mỗi Task sẽ chạy trong 1 khoảng thời gian nhất định, ví dụ như 10ms, nếu Task đó chưa chạy xong (ví dụ Task phải chạy trong 20ms) thì cũng sẽ tạm ngưng lại và chuyển sang chạy Task khác. Và cứ như vậy đến khi qua lại thì Task chưa hoàn thành sẽ chạy tiếp ở đoạn đã tạm ngững trước đó)

- Interrupt trong RTOS cũng sẽ khác so với Interrupt thông thường: Khi xảy ra Interrupt RTOS thì sẽ nhảy vào hàm thủ tục ngắt và hàm này sẽ lại phát đi 1 tín hiệu để thực hiện chức năng của ngắt Unblocking và hàm này có mức ưu tiên cao hơn tất cả các mức độ ưu tiên các Task nên sẽ ưu tiên thực hiện ngay lập tức.

- Một số dạng tương tác với kho lưu trữ dữ liệu trong RTOS:

+ Queue: là dạng dữ liệu được lấy ra theo thứ tự dạng FIFO (First in First out). Hiểu đơn giản ta có một cái ống rỗng vào liên tục đẩy vật vào đầu A, thì sẽ lấy vật ra ở đầu B

+ Semaphore: Hãy tưởng tượng chúng ta có 1 cái kho lưu trữ và chỉ có 1 vài chìa khóa (Tùy vào ta quy định sẽ có bao nhiêu). Tại một thời điểm nào đó thì chỉ những ai có chìa khóa mới được vào kho lấy hàng, những người không có chìa khóa phải đợi cho đến khi có chìa khóa mới vào lấy được. Tuy nhiên, dù đang giữ Semaphore và đang lấy Data mà có Task nào đó có mức ưu tiên cao hơn cần thực hiện công việc của Task đó thì vẫn phải tạm ngưng việc lấy dữ liệu cho Task đó thực hiện.

+ Mutex: Giống Semaphore nhưng khác ở chỗ là tại 1 thời điểm chỉ có duy nhất 1 Mutex và Task nắm giữ Mutex sẽ không bị gián đoạn bởi bất cứ Task nào dù Task đó có mức ưu tiên cao hơn (Tính kế thừa mức độ ưu tiên)