Tạp Chí Tin Học và Điều Khiển Học
Tập IX(1993), Số 1, 8-18

Một Số Vấn Đề Cơ Bản Của Hệ Điều Hành Thời Gian Thực*

Phan Minh Tân
Viện Tin Học
Viện Khoa Học Việt Nam

I. Mở đầu

* Hiện nay việc sử dụng máy vi tính, kĩ thuật vi xử lí vào các hệ thống đo lường, điều khiển ngày càng nhiều do các đặc điểm ưu việt của chúng như: Tính ổn định, độ tin cậy, chất lượng điều khiển cao và khả năng kết hợp với các nhiệm vụ khác như thống kê, quản lý v.v... . Đối với các hệ thống điều khiển tự động máy tính phải thực hiện các nhiệm vụ điều khiển trong một thời gian nhất định; mà hệ điều hành thông thường như các hệ xử lí theo lô (batch processing) hoặc các hệ phân chia thời gian (timesharing) phần lớn không đáp ứng được. Hệ điều hành thời gian thực có thể tiến hành tính toán và xử lí song song các nhiệm vụ điều khiển kịp với các yêu cầu về thời gian thực của đối tượng điều khiển. Bài này giới thiệu một số nét đặc trưng, cấu trúc và phương pháp thiết kế một hệ điều hành thời gian thực (Real time Operating System: RTOS).

2. Các đặc trưng chủ yếu của hệ điều hành thời gian thực

Xử lí đa nhiệm (Multitasking)

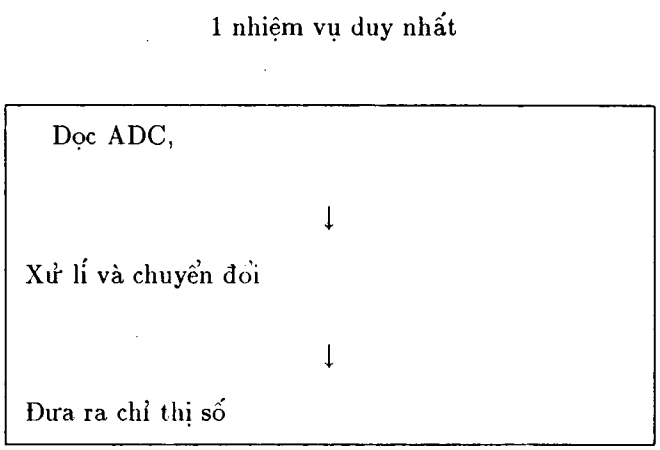
Nhiệm vụ (task) là một đoạn chương trình được sử dụng để thực hiện một công việc. Xử lí đa nhiệm là việc tổ chức phần mềm hệ thống, sao cho nó có khả năng xử lí song song nhiều nhiệm vụ. Khi máy tính được dùng để điều khiển một quá trình công nghệ, thì thông thường phần mềm của nó được chia thành các nhiệm vụ riêng biệt. Bản thân mỗi nhiệm vụ cũng có thể bao gồm các modul hoặc các nhiệm vụ con. Mỗi nhiệm vụ có thể được thực hiện theo một trong những cách sau:

- Tuần tự theo thời gian.

*Bài này viết với sự hỗ trợ của đề tài " Điều khiển thời gian thực" mã số KC-02-09, chương trình tự động hóa.

- Khi có đủ các điều kiện cần thiết thì được khởi động(event driven tasks).
- Chạy theo ngắt (interrupt).

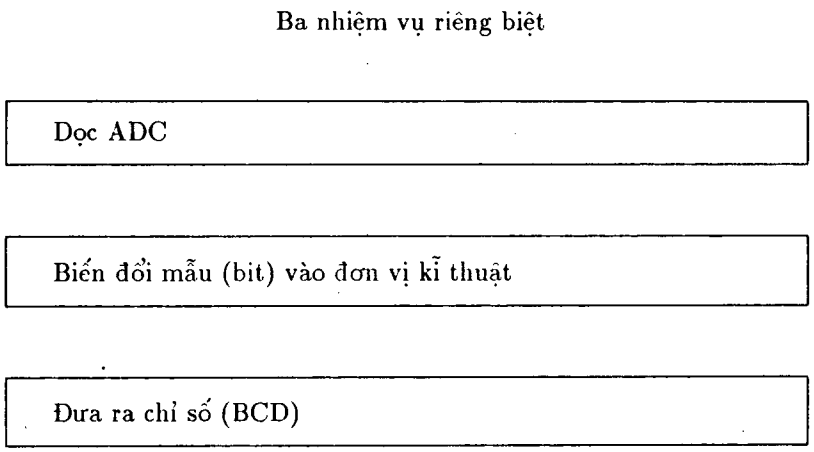
Ta hãy khảo sát một nhiệm vụ đơn giản: biến đổi và chỉ thị một số đại lượng đo nào đó. Nhiệm vụ này bao gồm ba phần như sau (xem hình 1):



Hình 1: Hiển thị đại lượng đo

Thông thường ba phần của nhiệm vụ này đòi hỏi thực hiện với các khoảng thời gian khác nhau. Giả sử sự hiển thị cần nhập số liệu 60 giây một lần và quá trình tính toán cho nhiệm vụ chỉ cần 0.25 giây. Trong trường hợp này, bộ xử lí trung tâm CPU chỉ làm việc dưới 0.5% thời gian và trong khoảng thời gian còn lại 99.5% chỉ chờ (extant) hoặc treo (suspend).

Trong nhiều trường hợp ta phải đo và hiển thị nhiều đại lượng, việc tổ chức như trên sẽ không đáp ứng được vì không đủ thời gian. Ta có thể tổ chức tách 3 chức năng trên thành 3 nhiệm vụ riêng biệt (xem hình 2) và cho chúng chạy với tần xuất khác nhau.



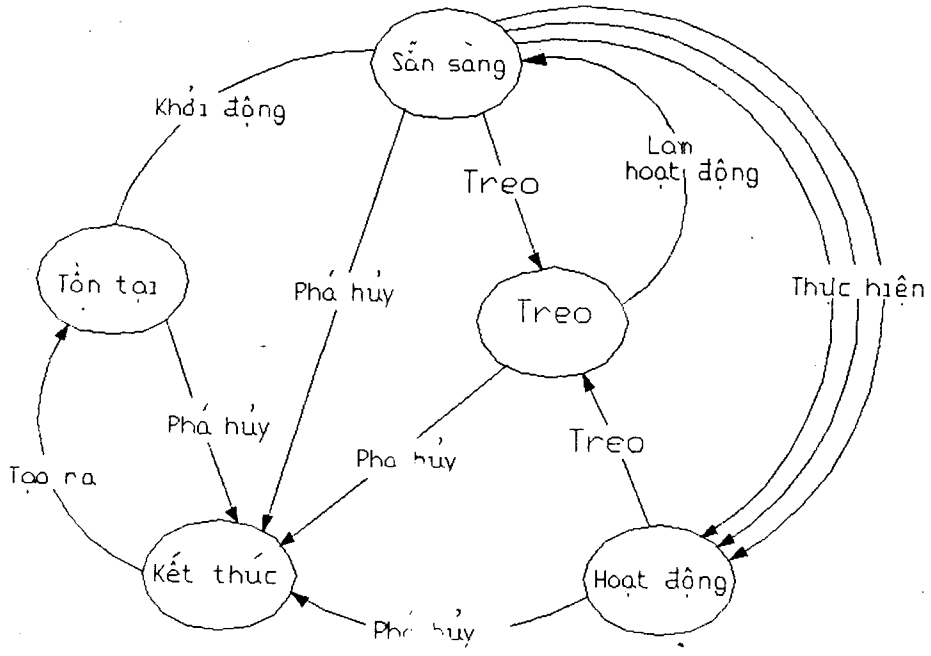
Hình 2: Hiển thị đại lượng đo qua 3 nhiệm vụ độc lập.

Trong trường hợp này ta phải tổ chức việc phối ghép số liệu (vào và ra) phù hợp cho mỗi nhiệm

vụ (hoặc modul). Các nhiệm vụ sẽ được thực hiện ít hoặc nhiều tùy theo yêu cầu và chúng còn được sử dụng trong các hoạt động khác của toàn hệ thống. Khi tổ chức chương trình mỗi nhiệm vụ được gán theo một bảng trạng thái và chúng có thể hoạt động ở một trong số các trạng thái ấy (xem bảng 1). Hình 3 giới thiệu mối quan hệ và bước chuyển đổi giữa các trạng thái. Hệ điều hành thời gian thực (RTOS) quản lí các nhiệm vụ qua các bảng trạng thái của chúng. Hệ điều hành cần phải có khả năng thay đổi trạng thái của tất cả các nhiệm vụ phù hợp với bảng danh mục và các điều kiện quản lí tài nguyên.

Trạng thái	Mô tả
Tồn tại (extant)	Có khả năng hoạt động
	(chờ tài nguyên, CPU, bộ nhớ, vào ra)
Hoạt động (active)	Thực hiện nhiệm vụ
Treo (suspend)	Chờ thựchiện
	(chờ nhiệm vụ khác kết thúc công việc)
Kết thúc (terminated)	Nhiệm vụ bị đưa ra khỏi vòng quản lí của hệ điều hành.
	Nhiệm vụ bị quên đi, tài nguyên được giải phóng).

Bảng 1: Các trạng thái của nhiệm vụ



Hình 3: Các trạng thái của nhiệm vụ và quan hệ của chúng.

Hệ điều hành thời gian thực phải có một số lệnh hệ thống nhằm tạo ra, khởi động, làm hoạt

động, treo và phá hủy nhiệm vsử dụngu. Bảng 2 tóm tắt chức năng của các nhiệm vụ này.

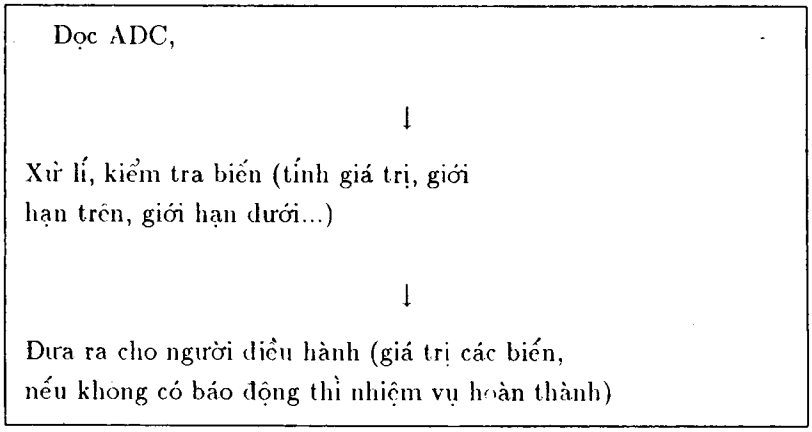
Lệnh	Chức năng
Tạo nhiệm vụ (create)	Công bố một nhiệm vụ mới cho hệ điều hành với trạng thái ban đầu là tồn tại (extant). Lệnh này cung cấp một bảng các thông số mô tả nhiệm vụ cho hệ điều hành.
Khởi động (initiate)	Chuyển nhiệm vụ đang ở trạng thái tồn tại (extant) sang trạng thái sẵn sàng (ready) để thực hiện.
Làm hoạt động (activate)	Chuyển nhiệm vụ bị treo (suspended) vào trạng thái sẵn sàng. (ready).
Treo (Suspended)	Lệnh suspend chuyển nhiệm vụ từ trạng thái active sang trạng thái bị treo (suspended). Tất cả các thông tin về nhiệm vụ được cất giữ cho hoạt động trở lại về sau. Treo xảy ra khi nhiệm vụ quan trọng hơn được thực hiện hoặc khi có yêu cầu ngắt.
Phá hủy (destroy)	Lệnh này dời nhiệm vụ ra khỏi sự quản lí của hệ điều hành. Lệnh destroy hoạt động từ các trạng thái ready, suspended, active. Các thông tin liên quan tới nhiệm vụ đã kết thúc có thể còn lại trong bộ nhớ.

Bảng 2: Các lệnh xử lí của hệ điều hành.

Xử lí các mức ưu tiên (Priorities)

Hoạt động dị bộ giữa các nhiệm vụ là đặc trưng của hệ điều hành thời gian thực. Tất cả các nhiệm vụ đều có thể đòi hỏi tài nguyên chung và sự quản lí của hệ thống. Ta hãy khảo sát một ví dụ: hai nhiệm vụ cùng được khởi hành, một để tiến hành kiểm tra báo động (ưu tiên cao) và hai là để hiển thị các đại lượng đo (ưu tiên thấp). Như đã giới thiệu trong hình 1, kiểm tra báo động có dạng ở hình 4:

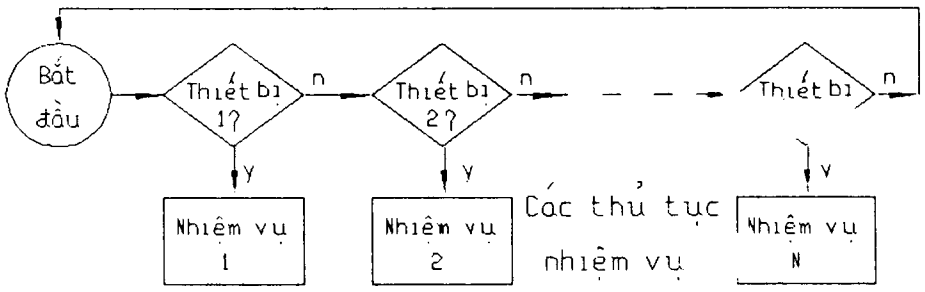
1 nhiệm vụ duy nhất



Hình 4: Nhiệm vụ kiểm tra báo động

Có hai cách xử lí mức ưu tiên như sau:

Hệ thống ưu tiên đồng nhất là cách đơn giản nhất để quyết định chương trình thực hiện nhiệm vụ nào trước. Một thủ tục quét kiểm tra từng nhiệm vụ trong trạng thái tồn tại và nếu có sẵn sàng thì thực hiện nó. Thứ tự ưu tiên của các nhiệm vụ được sắp xếp trước (xem hình 5).

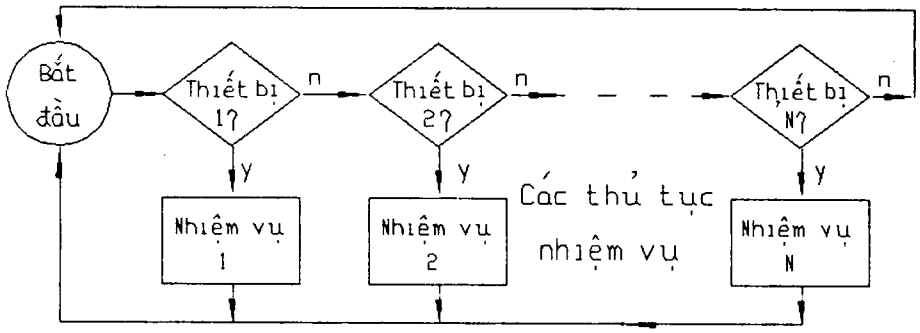


Hình 5: Xử lí nhiệm vụ theo hệ thống ưu tiên đồng nhất

Theo hệ thống này ta có thể thấy ngay rằng nhiệm vụ báo động phải chờ việc xét nhu cầu của nhiệm vụ hiển thị và các nhiệm vụ khác trước khi nó được thực hiện lại. Như vậy có thể dẫn đến khả năng xử lí báo động không kịp thời. Để khắc phục điểm này ta tổ chức lại theo cách là sau khi mỗi nhiệm vụ được hoàn thành, tiến hành quay về kiểm tra từ nhiệm vụ đầu tiên. Như

vậy tất cả các nhiệm vụ đều được kiểm tra từ khởi đầu của trình tự (xem hình 6). Ở đây, báo động là nhiệm vụ đầu tiên được kiểm tra (được xếp ở vị trí thứ nhất). Sau khi mỗi nhiệm vụ thực hiện xong quy trình quay lại từ đầu. Cách xử lí mức ưu tiên này gọi là xử lí ưu tiên không đồng nhất.

Để quyết định thứ tự xử lí nhiệm vụ, hệ điều hành cần có thêm một bảng danh mục có mức ưu tiên cho các nhiệm vụ. Các trạng thái sẽ phức tạp hơn khi mức ưu tiên của mỗi nhiệm vụ có thể thay đổi theo thời gian. Ở đây bảng ưu tiên cần chỉ ra các nhiệm vụ đã đưa vào trong thứ tự ưu tiên (hình 7).



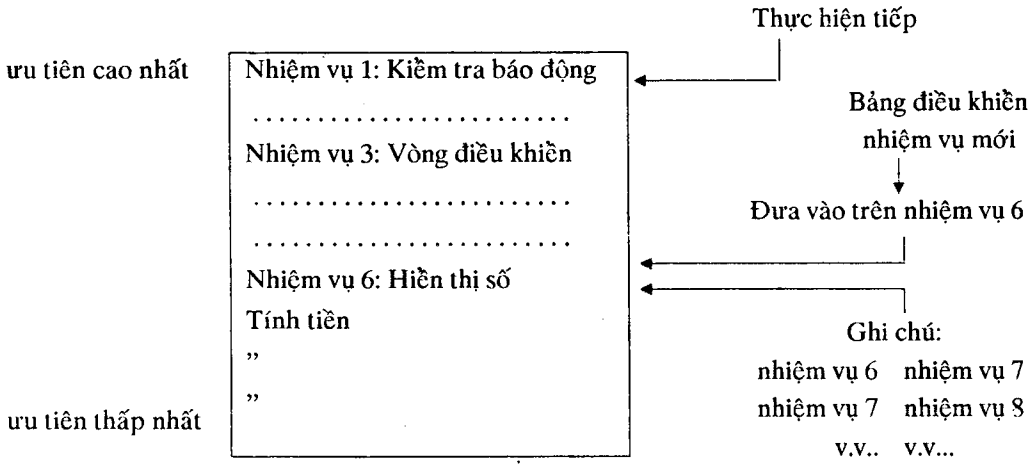
Hình 6: Xử lí nhiệm vụ ưu tiên không đồng nhất

Xử lí ngắt (Interprrupt)

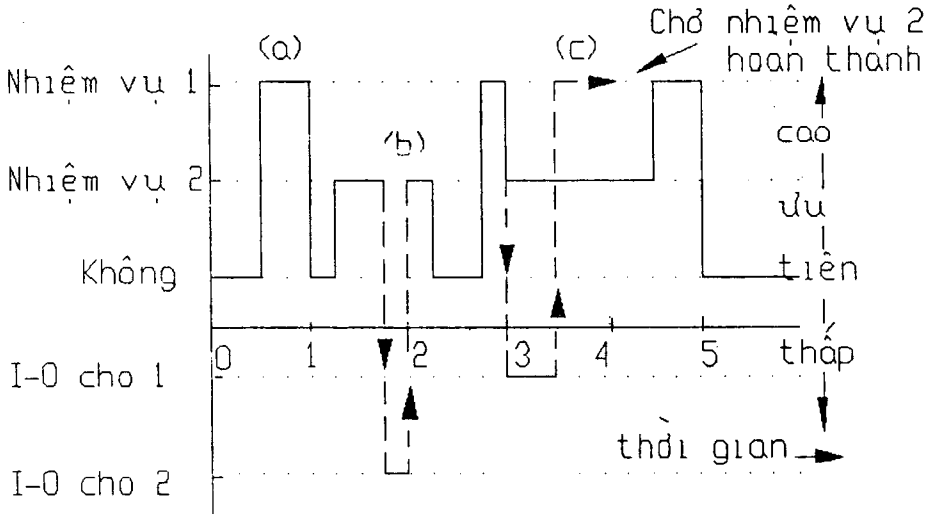
Trong hệ thống xử lí theo các mức ưu tiên kể trên mỗi lần một nhiệm vụ bắt đầu thực hiện, nó phải tiến hành cho tới hoàn tất trừ phi nó bị treo để chờ số liệu vào ra. Điều này trong điều khiển các quá trình thời gian thực không thoả mãn được các nhu cầu thực tiễn. Hãy xem một ví dụ sau:

Giả sử nhiệm vụ 1 có mức ưu tiên cao hơn nhiệm vụ 2. Theo hình 8 ta thấy nhiệm vụ 1 đang chạy, nó không đòi hỏi I/O và kết thúc tại (a). Tiếp đến nhiệm vụ 2 tiến hành, nó yêu cầu cổng I/O và kết thúc tại (b). Nhiệm vụ 1 chạy lại và yêu cầu cổng I/O, lúc này nhiệm vụ 2 cũng chạy lại (c). Nhiệm vụ 2 cần thời gian dài hơn thời gian cổng I/O thực hiện cho nhiệm vụ 1, nên nhiệm vụ 1 không thể bắt đầu lại việc xử lí mà phải chờ cho đến khi nhiệm vụ 2 kết thúc. Như vậy là nhiệm vụ 2 có mức ưu tiên thấp đã khóa nhiệm vụ 1 có mức ưu tiên cao.

Để giải quyết tình trạng này ta phải dừng ngắt, nghĩa là cho phép các nhiệm vụ khởi động ngắt cứng khi thấy cần thiết hoặc khi cổng I/O liên quan đã hoàn thành (xem hình 9). Trong trường hợp này nhiệm vụ 1 khởi động lại tại (d) sau khi kết thúc yêu cầu cổng I/O bằng cách treo nhiệm



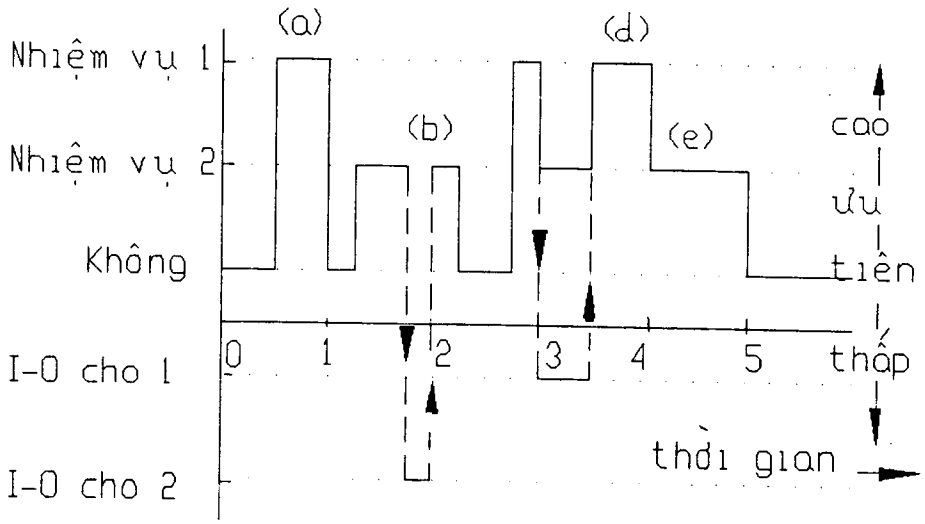
Hình 7: Bảng ưu tiên cho các nhiệm vụ



Hình 8: Nhiệm vụ ưu tiên thấp hơn gây trở ngại cho nhiệm vụ ưu tiên cao

vụ 2. Nhiệm vụ 2 sẽ kết thúc chậm hơn tại (e). Thí dụ trên đây cho thấy rằng một nhiệm vụ có thể treo các nhiệm vụ khác. Để khởi động lại nhiệm vụ bị treo phải bảo tồn các trạng thái của nó. Điều này chỉ đạt được bằng cách sử dụng thủ tục phục vụ ngắt. Sử dụng ngắt bên trong hệ điều hành đòi hỏi sự chú ý lớn. Trình tự của ngắt thường được tổ chức như sau:

1. Ngắt được khởi động.
2. CPU kết thúc lệnh hiện hành tại thời điểm xảy ra ngắt.



Hình 9: Ưu tiên của nhiệm vụ được thiết lập bằng ngắt

3. Điều khiển được chuyển vào vị trí cất giữ cho trước.

4. Nếu ngắt được chấp nhận, điều khiển được chuyển cho chương trình ngắt để nó tiếp tục.

5. Nội dung của thanh ghi địa chỉ và các số liệu trạng thái chương trình bị ngắt được cất vào vùng cho trước.

6. Thực hiện thủ tục phục vụ ngắt.

7. Nạp lại trạng thái chương trình bị ngắt và thực hiện việc khởi động lại chương trình.

Nếu hệ thống có nhiều nguồn ngắt thì trình tự thực hiện sẽ phức tạp hơn.

Thí dụ minh họa.

Say đây là ví dụ minh họa cho quan hệ tương hỗ của các nhiệm vụ, các ưu tiên và các ngắt. Hình 10 giới thiệu cấu trúc tổng thể các mức ưu tiên trong máy tính điều khiển quá trình.

+ Đỉnh ưu tiên (Kích bằng sự kiện bên ngoài)	⟶ các mức ngắt cứng và các chương trình dịch vụ ngắt cứng.
+ Ưu tiên trung bình	⟶ Hệ điều hành thời gian thực.
+ Dãy ưu tiên thấp nhất (Thứ tự sắp xếp công việc)	⟶ Các mức ưu tiên mềm gồm tất cả các nhiệm vụ điều khiển và yêu cầu số liệu.

Hình 10: Tổ chức các mức ưu tiên trong máy tính điều khiển quá trình

Hình 11 liệt kê chi tiết hơn việc sắp xếp các nhiệm vụ thường gặp trong các hệ thống điều khiển quá trình công nghệ.

Các mức ngắt cứng

1. Nguồn nuôi hỏng.
2. Sai số của xử lí số liệu (chia cho 0).
3. Ngắt khẩn từ bàn phím, bảng điều khiển.
4. Các yêu cầu ngắt cứng khác của hệ thống như ngắt của đồng hồ thời gian thực, chuyển dữ liệu vào đĩa.

Hệ điều hành thời gian thực

Các mức ưu tiên mềm.

1. Quét đầu vào tương tự và biến đổi. Đầu vào số và phát hiện báo động. Hiển thị báo động (CRT).
2. Xử lí tín hiệu. Tính thuật toán DDC. Phát lệnh điều khiển cơ cấu chấp hành v.v...
3. Phục vụ bảng điều khiển.
4. Đưa ra máy in.
5. Tính toán mô hình tối ưu quá trình. Tính toán hiệu quả và thủ tục quản lí khác.
6. Định nghĩa lại vòng điều khiển. Phát triển chương trình khác.

Hình 11: Ví dụ về cấu trúc ưu tiên trong máy tính điều khiển quá trình

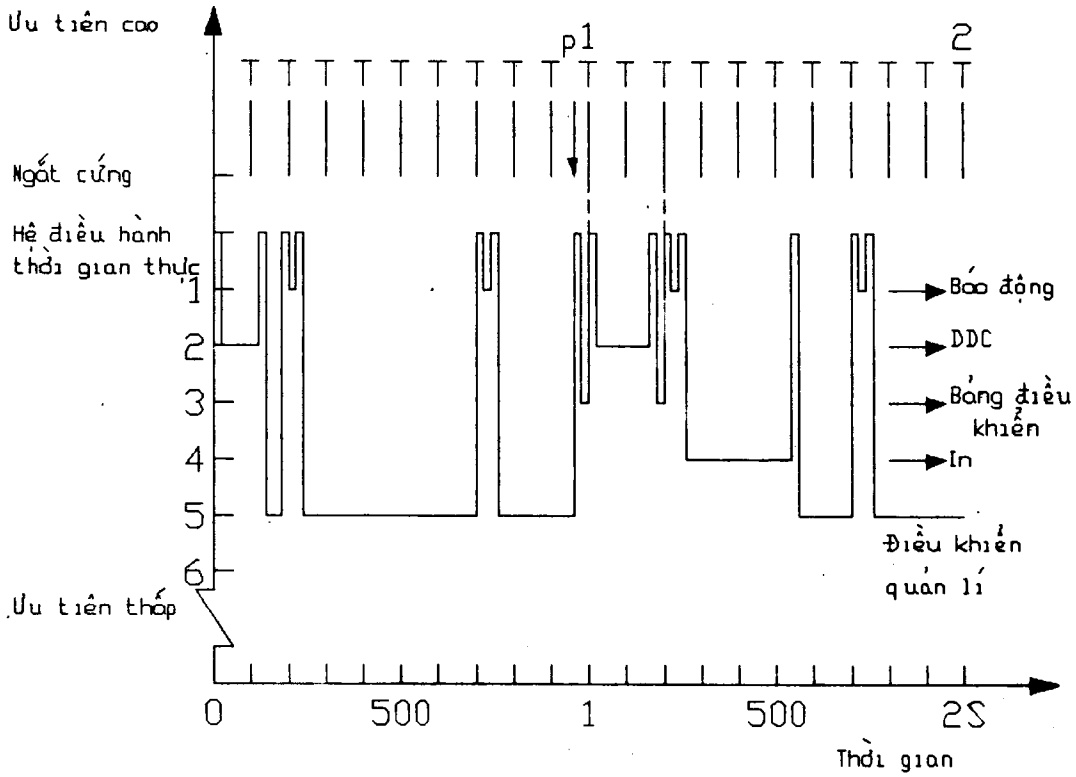
Ví dụ 1.

để miêu tả sự hoạt động nhịp nhàng của hệ điều hành thời gian thực như xử lí các nhiệm vụ có mức ưu tiên khác nhau ta hãy xét ví dụ sau: Bảng dưới đây mô tả 5 nhiệm vụ của một hệ điều khiển số.

Mức ưu tiên	Chức năng	Tần xuất
1	Kiểm tra mức báo động	0.5s (khởi hành 0.25)
2	Vòng điều khiển số DDC	1 s
3	Phục vụ bảng điều khiển	theo yêu cầu
4	In số liệu	theo cần thiết
5	Các công việc quản lí	vùng sau (background)

Bảng các nhiệm vụ và mức ưu tiên của một hệ thống điều khiển số

Hình 12 mô tả hoạt động của các nhiệm vụ có các mức ưu tiên khác nhau theo trục thời gian. Các mức ưu tiên cao thấp được vẽ trên trục đứng. Kí hiệu "T" là chu kỳ cắt mẫu của đồng hồ thời gian 0.1 s và "P" là yêu cầu ngẫu nhiên của người sử dụng bảng điều khiển. Trục ngang là trục thời gian.



Hình 12: Hoạt động của các nhiệm vụ theo trục thời gian.

Chú ý rằng tần xuất của vòng điều khiển DDC bằng 1/2 của vòng kiểm tra báo động. Khi thiết kế hệ thống cần giảm xung đột ưu tiên, ta dịch việc kiểm tra báo động chậm đi 0.2 giây.

Hoạt động bắt đầu ở thời điểm không. RTOs quét danh sách các nhiệm vụ hoạt động (List of Active Programs: LAP) và tìm thấy vòng DDC yêu cầu ở mức ưu tiên 2 (PL2). Chương trình DDC thực hiện xong trả điều khiển trở lại cho RTOS. Tiếp theo các chương trình quản lí ở mức PL5 được thực hiện. Sau 0.2 giây hệ điều hành cho khởi động chương trình kiểm tra báo động ở mức PL1. Nhiệm vụ này kết thúc trong một thời gian rất ngắn và điều hkiển được trở về hệ điều hành xung nhịp, rồi tiếp tục các chương trình ở mức PL5. Quy trình kiểm tra báo động lại xảy ra ở thời điểm 0.7 giây. Sau đó chương trình quản lí bị treo để thực hiện thủ tục phục vụ bảng điều khiển ở mức PL3. Tại thời điểm 1 giây chương trình DDC được chạy lại ở mức PL2 và như vậy 2 chương trình bị treo, một ở PL5 và một ở PL3. Khi DDC kết thúc, điều khiển trả lại RTOs và nó cho phép chương trình bảng điều khiển (PL3) hoàn thành. Tại thời điểm 1.2 RTOS kiểm tra lại các chu kì thời gian và cho các chương trình báo động chạy lại ở mức PL1 và sau đó thực hiện việc in ở mức PL4. Các hoạt động tiếp có thể giải thích tương tự như trên.

Tài liệu tham khảo

1. Barney G.C., Intelligent instrumentation, Prentice Hall-International.
2. Verker C., Introduction to real time operation systems, September, 19, 1990. Real time College. Trieste 1-27 October 1990.
3. Real time operating system RT/68.
4. Auslander D.M. & Sagues P., Microprocessors for measurement and control, Osborn/Mc Graw-Hill Berkeley, California

Abstract

Realtime Operating Systems

This paper is concerned with the design of operating systems for realtime or online computer systems. An online or realtime operating system must respond to all critical demands in restricted times promply. After presenting the functions and principles of design, some examples of using Realtime Operating System in measurement and control are given for illustration.