TẠP CHÍ KHOA HỌC TÍNH TOÁN VÀ ĐIỀU KHIỂN TẬP III (1987), Số 1 (25 - 28)

MỘT PHƯƠNG PHÁP XÂY DỰNG PHẦN MỀM CHO HỆ VI XỬ LÍ ỨNG DỤNG THỜI GIAN THỰC

ĐẶNG VĂN ĐỨC

Những năm gần đây, các hệ vi xử lí xuất hiện ngày càng nhiều. Ngoài việc ứng dụng vào quản lí, thống kê, xử lí văn bản.... chúng còn thâm nhập sâu rộng vào các ứng dụng thời gian thực : điều khiển quá trình, trao đổi thông tin, đo lường.... Để làm tăng sức mạnh của phần cứng, cần phải có phần mềm hệ thống hoàn hảo. Ở đây chúng ta không đề cập đến lĩnh vực tự động hóa trong phân tích hệ thống: đặc trưng lí thuyết và vật lí, mô hình toán học của quá trình vật lý, mô hình lí thuyết điều khiển. Chúng ta chỉ đề cập đến một khía cạnh quan trọng của lĩnh vực tin học: lựa chọn phương pháp lập trình cho hệ ứng dụng thời gian thực trên cơ sở bộ vi xử lí.

I – LỰA CHỌN CÔNG CỤ LẬP TRÌNH

Hệ thời gian thực là hệ có khả năng trả lời các cơ chế xung quanh liên quan sau một khoảng thời gian xác định để bảo đảm ổn định của toàn bộ hệ thống. Các ứng dụng thời gian thực rất khác nhau, nhưng chúng thường có các đặc tính sau đây:

– Vào/ra của hệ rất khác nhau.

– Thời gian trả lời yêu cầu từ bên ngoài rất ngắn.

– Có nhiều công việc xảy ra đồng thời.

– Yêu cầu an toàn cho toàn hệ là khắc nghiệt.

– Phần cứng ngày càng thay đổi nhanh chóng.

– Việc thích nghi phần mềm giữa các hệ ứng dụng và bảo hành phần mềm rất phức tạp.

Để xây dựng hệ ứng dụng thời gian thực, trước hết phải mô tả được dãy biến cố và các hoạt động của hệ vật lý. Do các ứng dụng thời gian thực ngày càng hoàn hảo và phức tạp, nhiều kết quả nghiên cứu về việc tìm kiếm phương tiện mô tả, phân tích và lập trình cho hệ đã được công bố: Sơ đồ các chương trình song song của Karp và Miller, đồ thị các ứng dụng thời gian thực của Mendelbaum.

Một hệ ứng dụng thời gian thực là tập hợp các chương trình con, mỗi chương trình con thực hiện một công việc (task). Mỗi công việc được gắn một mức ưu tiên. Phân chia hệ thống thành các công việc là nhiệm vụ khó khăn nhất trong thiết kế phần mềm hệ thời gian thực. Việc phân chia này được dựa theo một số nguyên lí sau:

– Mỗi công việc chính được coi là một công việc riêng biệt.

– Các chức năng nhỏ của hệ mà yêu cầu thời gian trả lời và thời gian chiếm giữ bộ vi xử lí khác nhau được chia thành các công việc riêng biệt.

– Các chức năng nhỏ có tầm quan trọng khác nhau được chia thành các công việc khác nhau.

Việc phân chia hệ thống thành các công việc làm giảm gánh nặng cho người lập trình và tối ưu hóa thời gian xâm chiếm vi xử lí, có nghĩa làm tăng tốc độ của hệ. Tùy theo yêu cầu của hệ ứng dụng, người lập trình lựa chọn công cụ mô tả các công việc của hệ. Việc chọn ngôn ngữ máy và assembler làm tăng tốc độ của hệ ứng dụng. Việc chọn ngôn ngữ bậc cao thời gian thực như LTR, Pearl, Ada... làm nhẹ bớt công việc lập trình. Nhưng chúng có

hạn chế là khó thích nghi vào quản lí các thiết bị ngoại vi đặc biệt. Vì vậy một giải pháp tối ưu là sử dụng ngôn ngữ lập trình để mô tả các công việc và monitor thời gian thực để quản lí chúng.

Các công việc (các chương trình con thường trực trong bộ nhớ) gọi monitor để đồng bộ, liên lạc với nhau.

II–CẤU TRÚC VÀ THIẾT KẾ MONITOR THỜI GIAN THỰC

Monitor thời gian thực được cài đặt trên một hệ có phần cứng đảm bảo các cơ chế sau đây:

– Gọi chương trình con và trở về.

– Cấm ngắt và cho phép ngắt.

– Bảo vệ và phục hồi các thanh ghi của bộ vi xử lí.

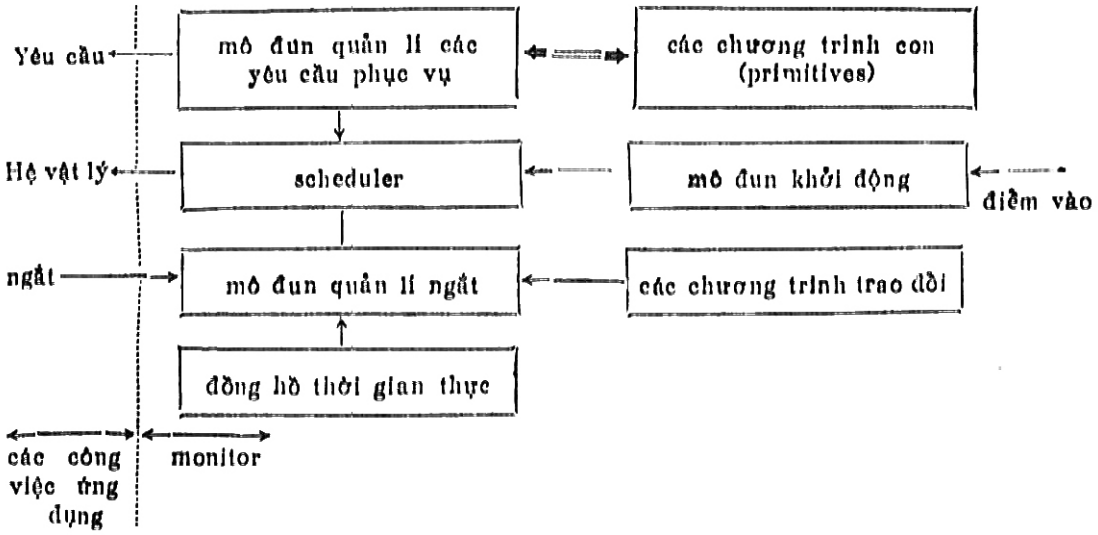
Cấu trúc của các hệ vi xử lí đều thỏa mãn nhu cầu trên. Monitor phải có các chức năng chính sau:

– Phân chia thời gian của bộ vi xử lí: các công việc của hệ thường phụ thuộc vào các biến cố bên ngoài. Vì không kiểm soát được chúng, nên có nhiều công việc cạnh tranh thời gian bộ vi xử lí (MP). Vậy monitor phải có một chương trình con gọi là scheduler để lựa chọn công việc sẽ thực hiện.

– Bảo vệ các tài nguyên không thể phân chia. Có thể một hoặc nhiều công việc đồng thời yêu cầu ngoại vi. Ngoại vi của hệ có thể là phân chia được hoặc không phân chia được giữa các công việc. Monitor đảm bảo việc phân chia loại trừ các tài nguyên không phân chia được.

– Quản lí các ngắt. Monitor xác định nơi phát sinh ngắt để khởi động các công việc đang nghỉ hoặc khởi động các công việc không tích cực.

Sau đây là cấu trúc tổng quát của một monitor:



Như vậy minitor là tập hợp các chương trình con cho phép thay đổi trạng thái công việc, quản lí các ngắt, quản lí đồng hồ thời gian thực, quản lí bộ nhớ và quản lí các lỗi xảy ra.

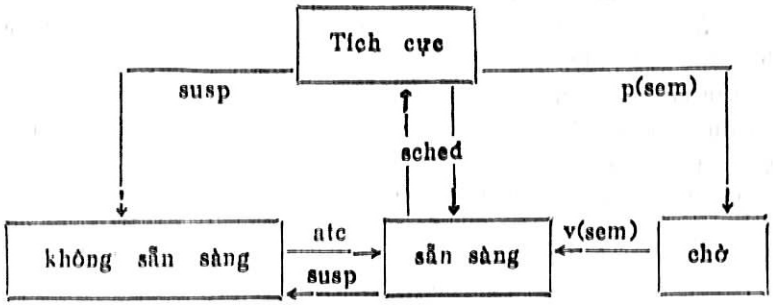
Thiết kế monitor thời gian thực

1. Các công việc (tasks)

Mỗi công việc có mức ưu tiên, vùng nhớ để bảo vệ các thanh ghi MP và vùng nhớ để chứa mã lệnh, số liệu. Các công việc được chia làm hai loại: cứng (task hardware – TH)

và mềm (task software–TS). TH được khởi động bằng ngắt. Mức ưu tiên của chúng tương ứng với mức ưu tiên ngắt do phần cứng quản lí. Khi xảy ra ngắt, TH được thực hiện từ đầu, chúng chỉ bị ngắt bởi TH khác có mức ưu tiên cao hơn.

Các TS nhận điều khiển qua scheduler. Chúng chọn TS có mức ưu tiên cao nhất và sẵn sàng để trao điều khiển. TS có thể bị ngắt bởi TH hoặc bị mất điều khiển khi có TS với mức ưu tiên cao hơn sẵn sàng. Một TS ở một trong các trạng thái sau:



– Không sẵn sàng: TS không muốn chiếm giữ MP.

– Sẵn sàng: TS muốn chiếm MP, đã có đủ tài nguyên để thực hiện (trừ MP).

– Tích cực: TS đang chiếm MP.

– Chờ: TS chưa đủ tài nguyên để thực hiện.

susp, act, v(sem), p(sem) là các primitives, chúng cùng scheduler biến đổi trạng thái của các công việc.

2. Các chương trình con (primitives)

a) MSK cấm ngắt mức ưu tiên NVTH.

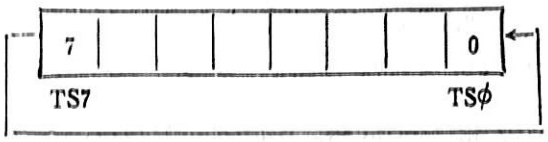
b) DMSK cho phép ngắt mức ưu tiên NVTH

c) INT chương trình xử lí ngắt: bảo vệ các thanh ghi MP của THi hoặc TSi đang thực hiện, thực hiện TH tương ứng với ngắt vừa xảy ra, chuyển điều khiển cho scheduler.

d) Scheduler – chọn 8 bit (1 byte) làm cờ biểu thị trạng thái của 8 TS như sau:

bit i = 0 TSi không sẵn sàng hoặc đang chờ,

bit i = 1 TSi sẵn sàng và tích cực.

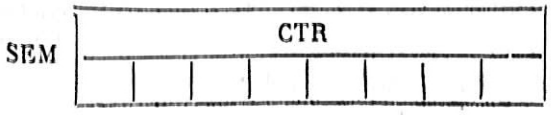


TSϕ có mức ưu tiên cao nhất. Trong các TS sẵn sàng thì chỉ có 1 TS tích cực. Số hiệu của TS tích cực được ghi vào ô nhớ gọi là TS–active. Scheduler tuyển chọn TS từ bit thấp nhất đến bit cao nhất theo chiều mũi tên.

e) ACT chuyển trạng thái của TSi.

f) SUSP chuyển trạng thái của TSi.

g) Semaphore là ảnh lôgic của một tài nguyên và các TS liên quan. Một semaphore gồm 2 ô nhớ: một làm bộ đếm, cho biết tổng số ST có thể đồng thời xâm nhập tài nguyên. Ô nhớ còn lại chứa số hiệu các TS đang chờ tài nguyên.



nếu CTR > 0, có thể sử dụng tài nguyên;

CTR = 0, tài nguyên đã bị chiếm, không có TS chờ;

CTR < 0, tài nguyên đã bị chiếm hết, |CTR| = tổng số TS chờ.

Các toán tử liên quan đến semaphore:

P(sem) – yêu cầu tài nguyên CTR = CTR – 1

nếu CTR $\geq$ 0, TS yêu cầu P(sem) tiếp tục chiếm MP;

nếu CTR $<$ 0, TS yêu cầu P(sem) vào hàng chờ, scheduler chọn TS khác.

V(sem) – giải phóng tài nguyên CTR = CTR + 1

nếu CTR $>$ 0, TS yêu cầu V(sem) tiếp tục chiếm MP;

nếu CTR $\leq$ 0, chọn trong hàng chờ, lấy TS có mức ưu tiên cao nhất để chuyển vào trạng thái sẵn sàng. Scheduler chọn TS có mức ưu tiên cao nhất.

h) Liên lạc giữa các công việc.

Các bảng tin được trao đổi qua vùng đệm. Vùng đệm có cấu trúc FIFO.

ENVOI – gửi 1 byte vào FIFO.

RETRAIT – lấy 1 byte ra khỏi FIFO.

III – KẾT LUẬN

Phần mềm ứng dụng xây dựng theo phương pháp trên đây được sử dụng một cách mềm dẻo. Vì có cấu trúc mô đun nên dễ thích nghi sang các hệ mới. Số lượng công việc được quản lí phụ thuộc vào dung lượng nhớ và thời gian trả lời yêu cầu.

Trong thực tế có nhiều hệ điều hành thời gian thực như iRMX (Intel), MTOS (Industrial Programming Inc), VRTX (Hunter Ready Inc)... Trong số đó có nhiều lõi của hệ được cứng hóa (chứa trong ROM); các hệ này thường có độ dài từ 2KB đến 16KB, thời gian trả lời $<$ 100 ms, có tập chương trình con khá phong phú. Nhưng do tính chất của các ứng dụng thời gian thực, khó có thể tìm kiếm được monitor thỏa mãn mọi nhu cầu của người ứng dụng. Vì vậy việc xây dựng monitor thời gian thực cho hệ vi xử lí vẫn là cần thiết.

TÀI LIỆU THAM KHẢO

1. Projet SCEPTRE, Rapport BNI, 1982.

2. Structuration dans la conception et la réalisation des systèmes en temps réel, Mendelbaum.

3. Mini et micro, N° 66, 218, 219.

4. Help a real–time multitasking OS by carefully defining each task, Electronic disig 1979.

5. Real–time OS prove difficult to evaluate, Marrin, EDN, 1985.

ABSTRACT

A METHOD OF DESIGNING SOFTWARE FOR REAL–TIME APPLICATION SYSTEM ON THE BASIS OF MICROPROCESSOR

The article refers to the software for real–time application system on the basis of microprocessor. The system is devided into task. These tasks are managed by real–time monitor. The real–time monitor is designed so that it had following mechanisms:

– Dividing the time of microprocessor

– Safeguapding the resources mon–divided

– Managing the interruptions

The Real–time Monitor is written by assembler for increasing the speed of the system.