CHƯƠNG 1. MỞ ĐẦU

1.0. Những quan niệm về hệ điều hành (operating system)

Một hệ điều hành là phần mềm chứa đựng tất cả các chương trình cần thiết để điều hành máy tính thực hiện các ứng dụng khác nhau. Nếu một người sử dụng máy tính chỉ để xử lý text, anh ta sẽ mong rằng, máy tính chứa đựng tất cả các chức năng của ứng dụng xử lý text.

Các phần mềm của hệ điều hành được dùng trên thị trường là những phần mềm đã được chuẩn hoá. Tất cả đều có chung những chức năng là kết nối giữa phần cứng của máy tính (như bộ vi xử lý, bộ nhớ và thiết bị xuất nhập) và các phần mềm ứng dụng (như các files, các chương trình của người sử dụng…).

Ta định nghĩa cô động: Hệ điều hành là tổng hợp các chương trình được sử dụng là phương tiện điều hành để quản lý và điều khiển. Định nghiax này không đòi hỏi sự đa năng và hoàn mỹ mà nó thực nghiệm một cách uyển chuyển để mô phỏng trạng thái tức thời của một khái niệm có thể thay đổi. Sự khác nhau của định nghĩa thứ nhất là ở chỗ, những chương trình dịch vụ (như biên dịch và kết nối) được gắn vào hệ điều hành, nhưng nó lại được tách rời trong định nghĩa thứ hai. Sau đây, chúng ta sẽ chấp nhận một quan điểm trung hoà:

Hệ điều hành là phần mềm có ứng dụng độc lập và cần thiết để điều hành một máy tính.

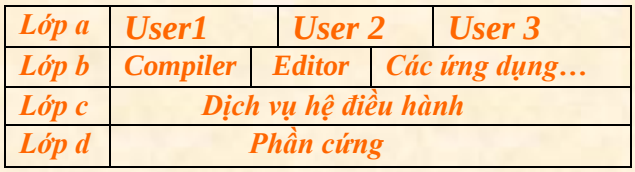
Tuy nhiên, sự cắt nghĩa khái niệm ứng dụng độc lập và cần thiết là chủ quan và do đó điều tất nhiên là phải dẫn tới những lý thuyết mới

1.1 Các lớp của hệ điều hành

Không có một hệ điều hành nào mà không cần tới một sự trợ giúp phù hợp với các yêu cầu của những chương trình ứng dụng. Sự trợ giúp này phụ thuộc vào cấu hình được người sử dụng (NSD) định nghĩa và biến đổi trong quá trình công tác. Nếu trước đây việc quản lý bộ vi xử lý, bộ nhớ và việc xuất nhập thuộc hệ điều hành, thì ngày nay một giao diện người sử dụng được đòi hỏi với các thành phần và độ lớn lớp khác nhau cũng như các chức năng của mạng máy tính. Đặc trưng của một hệ thống máy tính hoạt động độc lập với các phần mềm hiện hữu là cần tới một sự trợ giúp hữu hiệu tương ứng của hệ điều hành (HĐH). Hình 1.1 mô tả những quan hệ của các phần mềm và máy tính.

Hình 1.1.Những quan hệ tương đối của các thành phần.

Điều đó được chỉ ra một cách chặt chẽ hơn trong hình 1.2 dưới đây với mô hình hệ thống các lớp.



Hình 1.2. Mô hình các lớp

1.2 Các giao diện và máy ảo

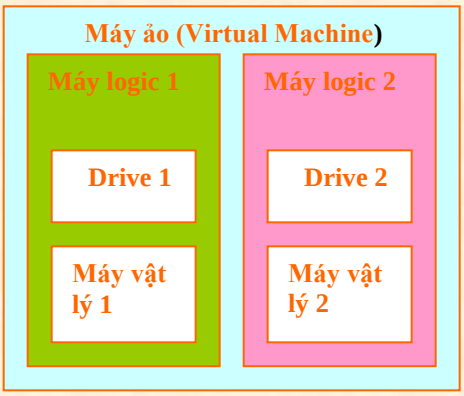
Ở trên chúng ta đã nói tới mô hình của các lớp. Sự tương đối của việc lớp này (thí dụ lớp a) ứng dụng lớp kia (thí dụ lớp b) chỉ ra rằng, lớp b đã dịch vụ lớp a. Đó là trường hợp ở việc sử dụng một procedure dưới lớp b trong một chương trình của lớp a. Nếu chúng ta xuất phát từ đó, rằng tất cả các khả năng dịch vụ được yêu cầu theo một thứ tự xác định, do đó những yêu cầu của người sử dụng đới với các chương trình tiện ích hay các yêu cầu của các chương trình tiện ích đối với các hệ điều hành hoặc các yêu cầu của hệ điều hành đới với phần cứng thì được diễn biến theo trục thời gian… Và những yêu cầu đó được sắp xếp liền kề nhau.

Mỗi một lớp không chỉ tạo thành một đơn vị phần mềm như hình 1.2, mà chúng còn được sắp xếp một cách tuần tự cạnh nhau. Những chức năng dịch vụ của một lớp (các procedure, các dữ liệu và các giao thức tiện dụng của chúng) được người ta tập hợp trong một giao diện. Chương trình mà nó mang lại những khả năng dịch vụ được tập hợp thành một dãy các lệnh, mà những dịch vụ này được sử dụng như những khả năng riêng. Lớp dưới cùng được tạo bởi phần cứng của máy tính. Vì các chức năng của chúng được điều chỉnh qua các giao diện, do đó người ta coi chúng như một máy. Tuy nhiên, máy này không tự làm việc được, nó không phải là máy thực và do đó người ta gọi là máy ảo.

Chức năng của các máy ảo tổng thể được tạo bởi sự cộng tác của các máy ảo riêng lẻ. Cho đến nay chúng ta đã có sự phân biệt giữa máy vật lý và máy ảo. Bây giờ có thêm loại thứ 3: máy logic. Một số người cho máy logic là máy ảo, số người khác tách biệt chúng thành máy vật lý và máy ảo.

Một ổ đĩa ảo được mô hình hoá một trường của các khối bộ nhớ mà nó xem đồng nghĩa với một số khối tuần tự. Ngược lại, ổ đĩa logic được mô tả một cái gì cụ thể hơn, nó được hiểu là ổ đĩa cứng với nhiều điểm khác biệt như thời gian trễ và sự ưu tiên khi vận chuyển dữ liệu (data). Hình 1.3 chỉ ra điều đó

Với ý tưởng đó, người ta đưa ra khái niệm quản lý các khối bộ nhớ, trong đó chỗ nhớ của các ổ đĩa cứng được quản lý một cách thống nhất mà không cần phải quan tâm tới giao diện của các ổ đĩa ảo.



Hình 1.3. Minh hoạ máy vật lý, máy logic và máy ảo

Trong kiểu kết hợp thứ 3, ở việc khảo sát một máy logic cụ thể thì độ chính xác đầy đủ của ổ đĩa cứng (thanh ghi trạng thái, thông tin lỗi, địa chỉ buffer đọc viết) phải rõ ràng. Việc quản lý thông tin và việc che phủ của lớp liền kề phía trên (của việc quản lý máy logic) cần tới bộ kích tạo chuyên dụng (specify- driver). Trong trường hợp này, ý nghĩa của các định nghĩa sẽ là:

- Máy ảo = máy logic + Bộ kích tạo quản lý (Manager-driver)
- Máy logic = Máy vật lý + Bộ kích tạo phần cứng (Hardware - driver)

Cả 3 loại máy tạo nên 3 lớp được trình bày như hình 1.3

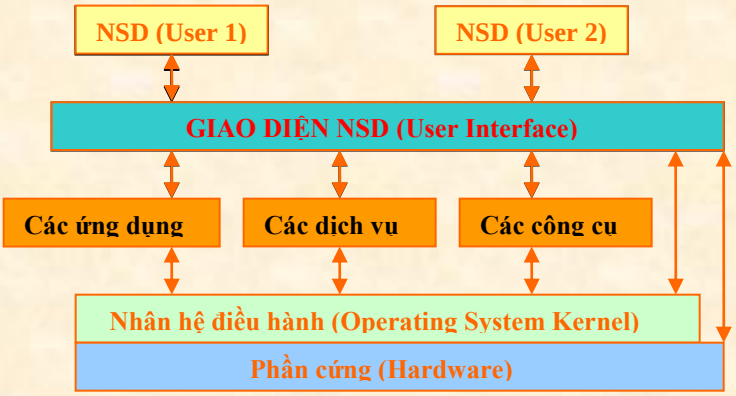
1.3 Kiến trúc hệ điều hành

Để điều hành những ứng dụng độc lập của máy tính, hệ điều hành (operating system) chứa đựng những phần mềm sau:

- Các chương trình dịch vụ và các công cụ: các chương trình tiện dụng như Editor…
- Các chương trình dịch: thông dịch (interpreter), biên dịch (compiler), chuyển đổi (translator)…
- Giao diện người sử dụng: hệ thống giao tiếp text và độ họa với người sử dụng…

Vì một hệ điều hành đầy đủ phải bao gồm hàng trăm MegaByte, do đó với dùng lượng này thì những chức năng ứng dụng thông thường được nạp để làm nhân của hệ điều hành (operating system kernel) ở trong bộ nhớ chính. Hình 1.4 mô tả việc nạp nhân hệ điều hành trong từng lớp của hệ thống máy tính. Đó là các

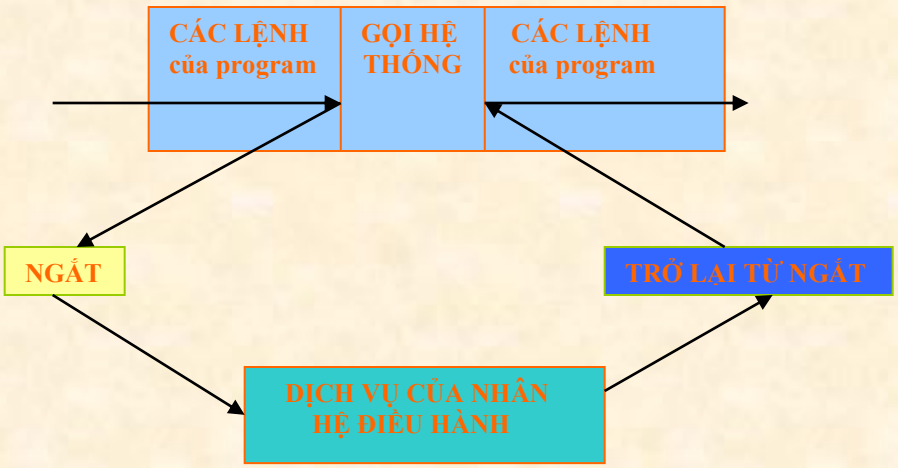
phần mềm để quản lý thiết bị, bộ nhớ và bộ vi xử lý, đồng thời cả những phần mềm quan trọng để quản lý mạng.



Hình 1.4. Cấu trúc khái quát các phần mềm của máy tính

1.3.1Gọi hệ thống (system- call)

Các dịch vụ của nhân được yêu cầu qua việc gọi hệ điều hành và được dẫn tới như gọi các procedure bình thường theo một khuôn khổ nhất định. Vì việc nạp nhân hệ điều hành ở bộ nhớ chính có thể thay đổi, do đó các chương trình tiện dụng luôn luôn được kết nối một cách mới mẻ. Ở hầu hết các nhân hệ điều hành có một cơ cấu gọi đặc biệt để đáp ứng khả năng dịch vụ mà không biết địa chỉ chính xác của procedure. Cơ chế gọi này bao gồm bộ nhớ các tham số ở trên ngăn xếp (*stack*) và khởi động một tín hiệu phần cứng hay bắt đầu một ngắt phần mềm *(softwareinterrupt)*. Như vậy, tại một ngắt phần cứng, bộ vi xử lý lưu trữ trạng thái và địa chỉ lệnh (instructionadress) ở ngăn xếp, đón nhận trạng thái và địa chỉ của lệnh kế sau đó ở bộ nhớ chính và tiếp tục thực hiện các lệnh ở địa chỉ này. Ở việc tự khởi động của hệ thống máy tính thì địa chỉ đưa vào của nhân hệ điều hành được mô tả trên không gian bộ nhớ ngắt, do đó chương ứng dụng tìm thấy dấu đợi lệnh (prompt) hệ điều hành. Hình 1.5 chỉ ra quá trình gọi hệ thống như vậy.



Hình 1.5. Vòng lệnh của việc gọi hệ thống

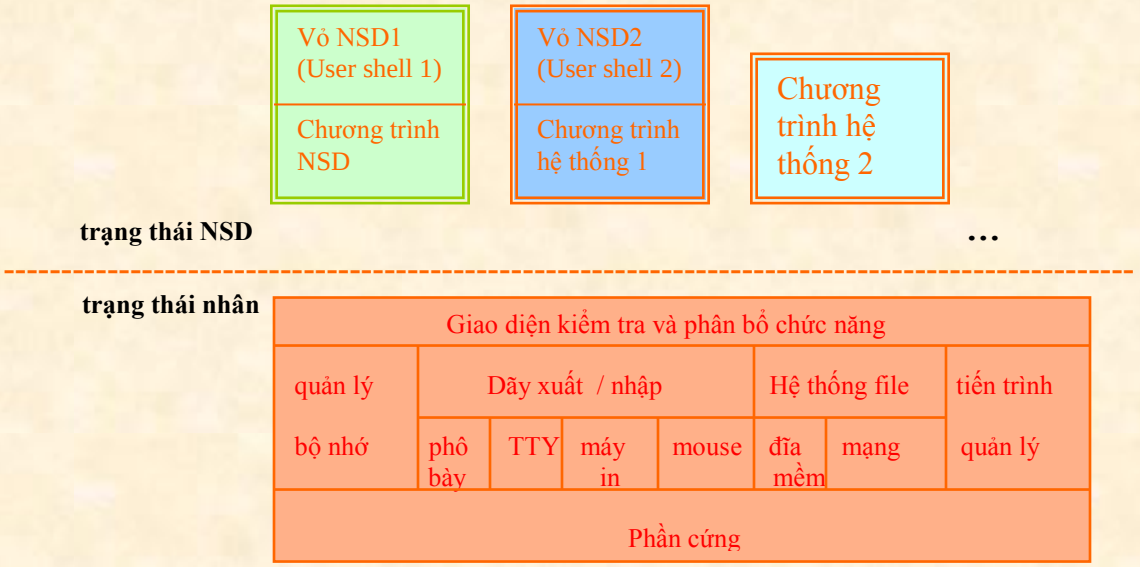
Sau khi gọi hệ thống thì lệnh kế tiếp không thực hiện đồng thời mà vòng lệnh ở địa chỉ nhớ này ngừng đột ngột, do đó ngắt phần mềm được biểu thị là cửa bẩy (trap door).

Việc chuyển đổi chương trình người sử dụng tới một hệ điều hành được tiến hành bất kì khi nào nhằm mở rộng mọi khả năng truy cập mã một cách hữu hiệu. Nếu phần lớn các máy tính được bảo vệ trước sự chuyển đổi chương trình người sử dụng thì do đó tất cả các thiết bị bảo vệ được ngắt ra để không cản trở nhân hệ điều hành. Ở việc rút khỏi, nhân hệ điều hành được đóng trở lại một cách tự động mà không cần người sử dụng biết điều hành xảy ra hầu hết qua bộ vi xử lý và không cần người sử dụng thực hiện bằng tay. Những bộ vi xử lý như thế còn chế ngự những mức độ bảo vệ kế tiếp, tuy nhiên nó nâng cao chi phí phần cứng trên vi mạch (chip).

Để khắc phục các lỗi trong khối dấu phẩy trôi (Floating Point Unit: FPU) thì ngắt phần mềm được sử dụng. Việc khắc phục lỗi này không chứa đựng trong hệ điều hành mà nó phó mặc cho chương trình ứng dụng hay chương trình ngoại vi.

1.3.2.Thí dụ về Unix

Trong hệ điều hành Unix truyền thống, ở giao diện người sử dụng có một bộ thông dịch lệnh mà người ta gọi là vỏ (shell). Từ đó, tất cả các chương trình người sử dụng cũng như các chương trình hệ thống của hệ điều hành được khởi động. Sự trao đổi thông tin giữa những người sử dụng và hệ điều hành xảy ra qua các kênh xuất nhập. Trường hợp đơn giản, đó là việc nhập các ký tự qua bàn phím và việc xuất qua thiết bị đầu cuối. Sơ đồ sơ lược được chỉ ra trên hình 1.6



Hình 1.6.Các lớp của hệ điều hành Unix

Cho đến nay, Unix vẫn là một hệ điều hành có nhiều thế mạnh khác nhau. Bởi lẽ, nó không chỉ là một hệ điều hành trợ giúp đồng thời nhiều người sử dụng, mà còn có thể thực hiện đồng thời nhiều chương trình. Nhờ việc thực thi vượt trội với ngôn ngữ lập trình bậc cao C thì nó có thể truy cập một cách dễ dàng và nhanh chóng tới một phần cứng khác. Ở một bộ biên dịch C cũ, người ta muốn thay đổi bộ tạo mã (*Codegenerator*) với câu lệnh mới thì công việc chính phải làm là, phải mô tả lại các chương trình của hệ điều hành cũng như của nhân bằng ngôn ngữ C và sau khi biên dịch, chúng có thể chạy như một máy tính mới.

Tuy nhiên, ở việc chuyển đổi như thế cũng còn vài vấn đề cần phải được hoàn hảo. Các phiên bản đầu tiên của Unix thì phụ thuộc rất nhiều vào phần cứng, trước hết nó phụ thuộc vào bề rộng từ (*word- width*) của CPU. Trong các phiên bản sau này (*Berkeley Unix cũng như Systems IV và V*) đã được tu chỉnh và sửa chữa nhiều. Cho đến nay công việc chuyển đổi không phải không còn vấn đề. Từ sự thực thi này tới sự thực thi khác đều có cấu trúc cơ sở khác biệt của Unix. Do đó loại và số lượng của gọi hệ thống là đa dạng. Việc chuyển đổi chương trình người sử dụng giữa các phiên bản khác nhau thực ra đã được giảm thiểu. Để có sự trợ giúp, thì những tổ chức khác nhau đã được thành lập. Nỗi bậc nhất là nhóm X/Open (tại bắc Mỹ và Châu Âu). Một hiệp định của các công ty và trường đại học đã đưa ra một vài chuẩn khác nhau. Một trong các chuẩn đầu tiên đó là hệ thống chuyển đổi POSIX (*Portable Operating System Interface based on Unix*) của Unix, mà nó được định nghĩa là một lượng các dịch vụ có thể sử dụng khác nhau. Tất nhiên chúng chỉ là những dịch vụ, chứ không được định nghĩa trực tiếp là gọi hệ thống. Điều thú vị là, biểu trưng Unix phù hợp với một giao diện bắt

buộc chứ không phải là mọt sự thực thi. Điều đó có ý nghĩa rằng, Unix thực chất là một cơ cấu hệ điều hành ảo chứ không phải thực thi.

1.3.3 Thí dụ về Windows NT

Hệ điều hành *Windows NT* của hảng Microsoft là một hệ thống tương đối hiện đại, nó được phát triển dưới sự lảnh đạo của David Cutler vào năm 1988. Một dự án để sản xuất một hệ điều hành chuyên dụng được Microsoft lần đầu tiên thực nghiệm, nó bao gồm các ý tưởng sau đây:

1. Một hệ điều hành phải bao hàm các chuẩn hiện hữu (MS- DOS, 16 bit-Windows, Unix và OS/2) và được thị trường chấp nhận một cách rộng rãi.
2. Nó phải chắc chắn và bền vững. Nghĩa là các chương trình vừa tương hổ nhau vừa không làm tổn hại hệ điều hành, các lỗi xuất hiện chỉ có hậu quả giới hạn.
3. Nó phải được truy cập một cách dễ dàng trên những phần cứng có phiên bản thấp.
4. Nó không được che phủ tất cả, mà nó phải đáp ứng một cách dễ dàng các yêu cầu chuyển đổi.
5. Điều quan trọng: nó phải là một hệ điều hành mạnh.

Nếu quan sát những dự kiến này, người ta thấy, đó là những điều nan giải. Những yêu cầu đồng thời (như tương thích, bền vững, tiện lợi và mạnh) thì đối lập với chính nó, vì MS-DOS thì tuyệt đối không bền vững, không mạnh và hoàn toàn không tương thích với Unix. Tuy nhiên, tác giả vẫn đạt được mục đích, vì anh ta đã sử dụng kinh nghiệm của các hệ điều hành khác và đã biến điệu một cách mạnh mẽ. Việc phân lớp và những quan hệ gọi của nhân ở bản phác thảo gốc được trình bày trên hình 1.7. Nhân của hệ điều hành này mang tên *Windows NT Executive*, nó là phần dưới của khối chuyển đổi trạng thái người sử dụng và trạng thái hệ điều hành.

Để giải quyết vấn đề thiết kế, dưới đây ba tiêu chí quan trọng của hệ điều hành được nói tới.

- Tính tương thích (*compatible*)

Đó là khả năng đặc biệt của một hệ điều hành mà ta có thể truy cập được vào một hệ điều hành riêng lẻ. Với tư cách là cơ cấu hệ điều hành ảo, nó được chứa đựng trong khả năng dịch vụ của Windows NT Executive (tức là gọi hệ thống được biểu thị bằng mũi tên vẽ đậm trên hình 1.7).

Việc xuất nhập định hướng ký tự được dẫn ra bởi dịch vụ của Win32. Những dịch vụ của hệ thống con (*subsystem*) được yêu cầu qua các thông tin (gọi các *local procedure*, xem mũi tên vẽ nhạt trên hình 1.7) của các chương trình người sử dụng hay của khách hàng. Nếu là server, chúng cũng có quan hệ client/server.

- Độ bền vững (*strengthen*)

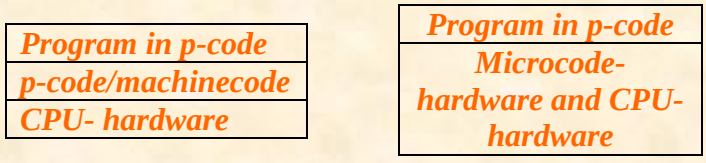
Nó đạt được nhờ sự phân chia chặt chẽ của các chương trình với nhau và nhờ việc luôn luôn giữ vững các quá trình ngoại vi (cơ cấu DOS ảo) đối với các chương trình chạy trên MS-DOS hay trên Windows. Các chức năng này là đồng đều, nhưng việc truy cập(số liệu) trực tiếp được tách ra trên phần cứng, do vậy với các chức năng này thì các chương trình cũ có thể chạy được. Những biện pháp phụ như một hệ thống files có độ chính xác lỗi và những cơ cấu bảo vệ chuyên dụng để kiểm tra sự chọn files, mạng và chương trình đã trợ giúp cho mục đích này.

- Độ thuận tiện (*commodity*)

Đó là khả năng có thể bảo quản và mở rộng, qua đó, một hệ điều hành được viết bằng ngôn ngữ C, có thể trao đổi mạnh và được phân lớp ngay từ đầu. Với lớp HAL (xem hình 1.7), phần cứng được mô phỏng như là máy ảo và ở việc truy cập tới các bộ vi xử lý khác nó thu hẹp sự thay đổi cần thiết trên các đơn thể. Điều thú vị là từ các version 4.0 của hệ thống Win32 thì đã tiết kiệm thời gian để gọi hệ thống từ Win32 tới Win NT. Sự trợ giúp của chuẩn khác (như hệ thống file OS/2 với version 4.0) thì hệ điều hành Win NT đã được điều chỉnh.

1.4 Sự giao kết phần cứng và phần mềm.

Cấu tạo của các máy ảo cho phép giữ lại giao diện một cách tương tự tới các đơn thể và thay đổi sự thực thi. Do đó, nó sẽ có điều kiện để thực hiện việc thực thi nhờ một sự giao trộn qua lại từ phần cứng và phần mềm. Đối với khả năng dịch vụ thì điều đó không quan trọng. Vì phần cứng làm việc hầu như nhanh hơn, nhưng giá đắt hơn; ngược lại phần mềm làm việc chậm hơn, nhưng giá lại rẻ hơn và có thể thay đổi nhanh chóng hơn. Do đó, khi thiết kế cấu trúc một máy tính , người ta phải lưu ý giải quyết hai nguyên do kể trên. Hình 1.8 là một thí dụ, cho thấy sự phân lớp theo mã máy tượng trưng (trên hình vẽ ký hiệu *p-code*), mà nó hoặc là được giao kết bằng phần mềm qua bộ biên dịch (compiler) và bộ thông dịch(interpreter), hoặc là có thể được thực hiện bằng phần cứng qua các lệnh máy. Trong trường hợp thứ hai, lớp ở giữa được di chuyển trong phần cứng (xem hình 1.8 ở phía bên trái). Trong đó, mỗi lệnh p-code được thực hiện một chức năng (tức là được lập trình bằng mã *microcode*) ở trong CPU.



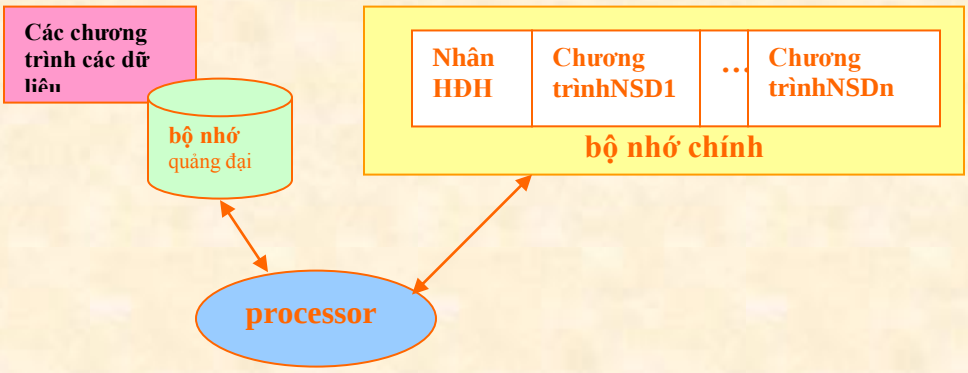
Hình 1.8. Sự giao kết phần cứng và phần mềm bằng mã p-code

Đối với vấn đề vừa nêu, một điều khác cần phải nói, đó là sự trợ giúp của các chức năng mạng. Với các bộ điều khiển mạng rẻ tiền thì hầu hết có chỉ chứa đựng tập hợp các chip theo tiêu chuẩn, nó chỉ là cấu nối tín hiệu và có thời hạn làm việc nhất định. Tất cả những chức năng và giao thức cao hơn để kết hợp các gói dữ liệu hoặc để tìm thấy lộ trình của mạng đều phải được bộ nhớ chính và phần mềm tích hợp thực hiện, điều đó làm giảm đi đáng kể khả năng của bộ vi xử lý đối với các nhiệm vụ khác(như giao diện người sử dụng, xử lý text…). Trên cơ sở đó, ở các bộ điều khiển mạng giá đắt hơn, các chức năng điều khiển mạng và quản lý dữ liệu được di chuyển trên phần cứng, nghĩa là bộ vi xử lý phải thực hiện một số quy trình để tiếp cận những chức năng ở mức độ cao.

Ở đây, điều chẳng có gì quan trọng đó là: những chức năng của bộ điều khiển mạng đã được hoàn thiện nhờ một bộ vi xử lý riêng lẻ hoặc nhờ kết cấu thích hợp của chip. Ngoài ra, nếu một *ngôn ngữ hình thức* được sử dụng để thiết kế các lệnh cho máy thì sự khác nhau của hai vấn đề trên không còn điều gì để nói. Điều quyết định còn lại cần phải được xem xét, đó là giá thành, chất lượng tiêu chuẩn và sự mong muốn của khách hàng.

1.5 Cấu trúc nhiều bộ vi xử lý (*multi- processorsystem*)

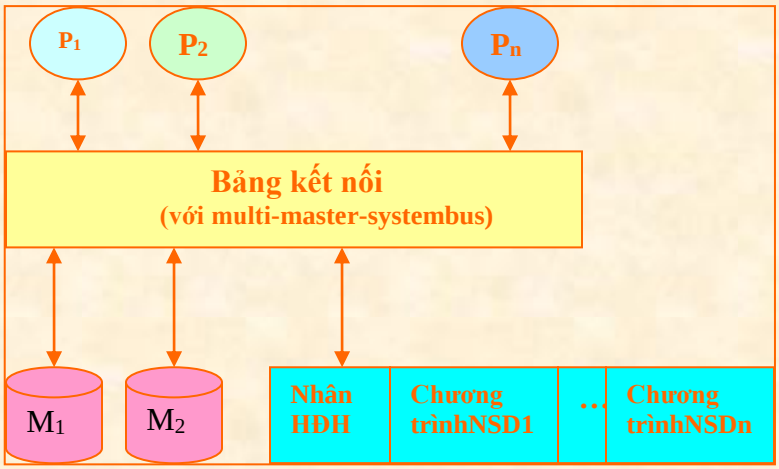
Đối với vấn đề quản lý các phương tiện điều hành thì điều quan trọng là phải xem xét còn tồn tại những quan hệ và những sự phụ thuộc nào giữa chúng. Điều đó độc lập với việc sử dụng kiểu bộ vi xử lý, kiểu bus, hay loại sản phẩm vi mạch các bộ nhớ, vấn đề là, người ta phải biết phân biệt một vài kiểu cấu hình cơ bản. Trong trường hợp kinh điển và đơn giản, trước hết ta khảo sát hệ thống với chỉ một bộ vi xử lý được dùng làm bộ nhớ quảng đại (*massen- memory*) và bộ nhớ chính (*main memory*) để xây dựng hệ điều hành và thực hiện các chương trình của người sử dụng (xem hình 1.9). Ở đây, các thiết bị vào- ra (như màn hình, bàn phím, chuột…) không chỉ ra trên hình vẽ.



Hình 1.9. Hệ thống một bộ vi xử lý đơn

Từ việc nghiên cứu hệ thống đơn vi xử lý như vừa nêu, người ta có thể thiết lập một hệ thống với nhiều bộ vi xử lý. Mỗi lần người ta kết nối các bộ vi xử lý khác nhau thì sẽ nhận cấu trúc hệ thống khác nhau. Cấu trúc đơn giản nhất cho thấy, đó là một CPU mà các bộ vi xử lý đơn được mắc song song qua một bảng kết nối, chẳng hạn qua một bus đa nhiệm (multi-master- systembus). Hình 1.10 mô tả một hệ thống nhiều bộ vi xử lý; trong đó P_1….P_n là các bộ vi xử lý đơn và M_1…M_n là các bộ nhớ quảng đại.

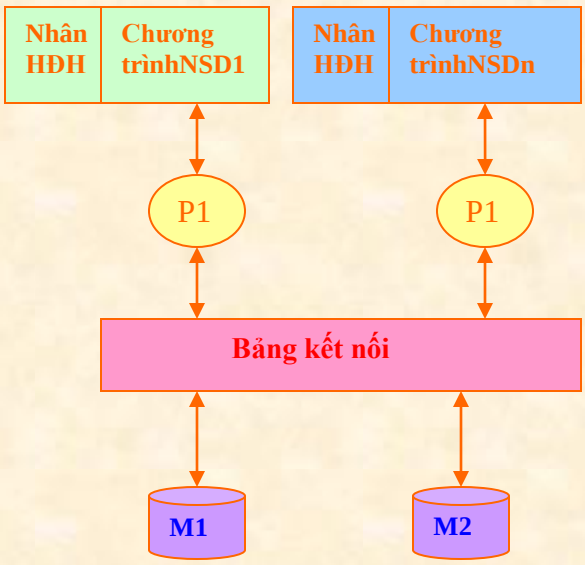
Các đơn thể bộ vi xử lý được định vị trên một phía bảng kết nối, còn phía kia là các đơn thể bộ nhớ. Đối với mỗi việc truy cập mã chương trình hay mã dữ liệu thì một sự kết nối giữa chúng được tạo ra, sự kết nối này tồn tại trong suốt thời gian yêu cầu.



Hình 1.10. Hệ thống đa vi xử lý

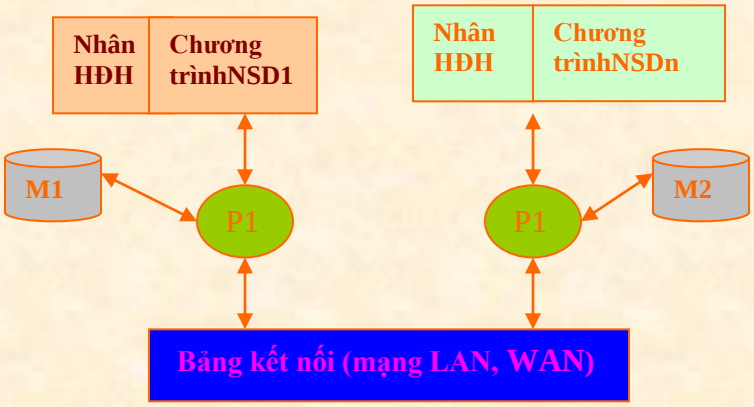
Một cấu trúc như vậy dẫn tới một cách dễ dàng sự giảm sút hiệu suất thực thi, vì bảng kết nối quá tải khi phải thường xuyên truy cập bộ nhớ. Trong trường hợp này, một sự trợ giúp được dẫn ra từ việc quan sát, rằng các bộ vi xử lý hầu như chỉ tham chiếu giới hạn hẹp một phần chương trình.

Ở đây, bộ nhớ có thể được phân chia và được dẫn ra một cách dày đặc hơn tới mỗi một bộ vi xử lý (xem hình 1.11). Tất nhiên, sự phân chia này phải được trợ giúp bởi bộ biên dịch để nó phân chia chương trìng người sử dụng một cách thích hợp.



Hình 1.11. Hệ thống nhiều máy tính

Mạng máy tính tồn tại như là một kiểu kết nối thứ ba. Ở đây, các máy tính làm việc độc lập với hệ điều hành riêng lẻ (nghĩa là không cần thiết phải giống nhau), được kết nối với nhau trong một mạng (xem hình 1.12)



Hình 1.12. Mạng máy tính - cấu trúc đa vi xử lý

Nếu có một phần mềm được tạo lập trên máy tính, mà máy tính này làm việc như là một máy chủ (server), nó cung cấp các dịch vụ (các tài nguyên) cho các máy tính khác (client) thì người ta gọi cấu trúc này có quan hệ khách- chủ (client-server). Thông thường, máy chủ có những khả năng dịch vụ là quản lý chương trình (number- cruncher), quản lý in ấn (print server), quản lý dữ liệu (file server)…

Mỗi một kiểu cấu trúc máy hệ điều hành máy tính được trình bày ở trên có những lợi thế và yếu thế khác nhau. Đối với chúng ta, điều quan trọng là, sử dụng cơ cấu nào để đạt được sự trao đổi thông tin qua các bộ vi xử lý và để đạt được sự

truy cập đồng bộ ở các phương tiện điều hành. Đó chính là phương hướng được tiếp tục xem xét kỷ lưởng ở các chương sau.

1.6. Các bài tập của chương 1

Bài tập 1.1. Về hệ điều hành

Mục đích của hệ điều hành bao gồm việc phân bổ các phương tiện điều hành theo sự thỉnh cầu của người sử dụng.

a). Vì sao phải tạo thô một hệ điều hành ?

b). Có những phương tiện điều hành nào bạn biết?

c). Người sử dụng nào có thể được thỉnh cầu ? (Ở đây khái niệm người sử dụng là một khái niệm khái quát).

d). Người sử dụng đòi hỏi những yêu cầu gì ở hệ điều hành?

e). Bạn hãy giải thích các khái niệm máy ảo và giao diện.

Bài tập 1.2. Về hệ điều hành Unix

a). Trên phạm vi các máy tính của cơ quan bạn, hệ điều hành Unix có bao nhiêu gọi hệ thống ?

b). Các gọi hệ thống được phân thành máy nhóm chức năng ?

c). Một gọi hệ thống được thực thi trên máy tính của bạn như thế nào?

Gợi ý:Nếu bạn mở quyển sổ tay về lập trình Assembler hay C, hoặc nếu bạn thực hiện trình Debugger dịch trở lại một gọi hệ thống được viết bằng ngôn ngữ C thành Assembler; khi đó các tệp tin include của C (như syscall.h. trap.h, proc.h, kernel.h…)sẽ đưa ra nhiều trợ giúp thú vị.