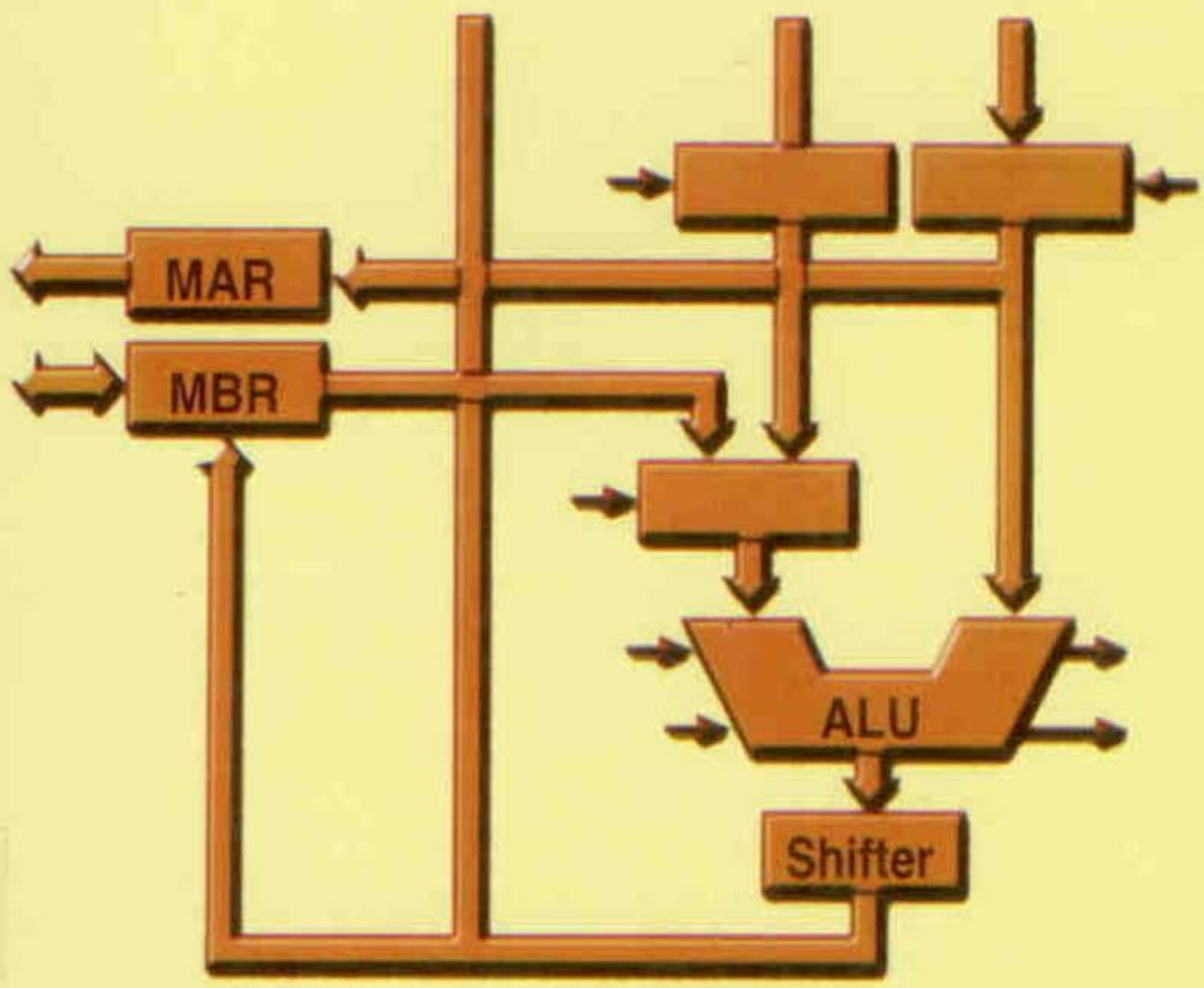
NGUYỄN ĐÌNH VIỆT

TRƯỜNG ĐẠI HỌC
DÂN LẬP HẢI PHÒNG
THƯ VIỆN
617.3
NG 527 D

KIẾN TRÚC MÁY TÍNH

0: mar : = pc; rd;
1: pc : = pc + 1; rd;
2: ir : = mbr; if n then goto 28



NGUYỄN ĐÌNH VIỆT
KIẾN TRÚC MÁY TÍNH

DVL2893

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

THƯ VIỆN
ĐH.DÂN LẬP HP
KÝ HIỆU: 6T7.3
SỐ: NG527Đ

THƯ VIỆN
ĐH.DÂN LẬP HP
SỐ:
KÝ HIỆU:

NGUYỄN ĐÌNH VIỆT

KIẾN TRÚC MÁY TÍNH

(In lần thứ hai)

THƯ VIỆN ĐH.DÂN LẬP HP
PHÒNG ĐỌC
2007 ĐVL 2893

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

LỜI NÓI ĐẦU

Máy tính điện tử (MTĐT) ra đời vào đầu thập kỷ thứ tư của thế kỷ 20 và phát triển như vũ bão, làm nên một cuộc cách mạng trong lĩnh vực tính toán và xử lý thông tin, mở ra một kỷ nguyên mới trong lịch sử nhân loại - Kỷ nguyên MTĐT. Cấu tạo của MTĐT ngày càng tinh vi và phức tạp, bao gồm nhiều thành phần khác nhau. Kiến trúc máy tính (KTMT) là ngành khoa học nghiên cứu việc thiết kế các thành phần cấu thành nên MTĐT mà người lập trình có thể nhận thấy được.

MTĐT không chỉ có các thành phần vật lý, thường được gọi là phần cứng (hardware) mà còn có cả các chương trình để điều khiển sự hoạt động của các thành phần phần cứng, chúng được gọi là phần mềm (software). Việc nghiên cứu thiết kế MTĐT là một bài toán khổng lồ và phức tạp, cần không ngừng được cải tiến. Chính vì vậy mà người ta phải chia bài toán lớn này thành các bài toán nhỏ hơn, mỗi bài toán nhỏ sẽ giải quyết một vấn đề cụ thể, sử dụng một mức trừu tượng thích hợp.

Giáo trình KTMT này trình bày các vấn đề chung nhất, các thành phần cơ bản nhất cấu thành nên MTĐT hiện đại. Giáo trình được biên soạn nhằm phục vụ cho sinh viên ngành Công nghệ thông tin. Tuy nhiên tác giả hy vọng nó cũng hữu ích cho nhiều bạn đọc khác có nhu cầu tìm hiểu lĩnh vực chuyên môn này, nhất là sinh viên các ngành khoa học tự nhiên.

Giáo trình gồm 9 chương, nội dung chính của từng chương như sau:

Chương 1 trình bày các khái niệm và nguyên lý cơ bản; Phác hoạ kiến trúc của MTĐT hiện đại như là một dãy các mức. Mức dưới cùng là mức vật lý, đó là phần cứng, bên trên là các mức phần mềm, mỗi mức này thể hiện ra như một cái máy ảo, mỗi máy ảo sử dụng một ngôn ngữ riêng của nó. Chương 1 cũng điểm qua các cột mốc trong lịch sử phát triển của MTĐT.

Chương 2: Tổ chức hệ thống máy tính, trình bày các thành phần căn bản nhất của MTĐT là Bộ xử lý (Processor), Bộ nhớ (Memory) và Hệ thống vào/ra (Input/Output).

Chương 3: Mức logic số, đây là mức thấp nhất trong kiến trúc MTĐT. Tại mức này chúng ta nhìn thấy các phần tử logic và các phần tử nhớ cơ bản, chúng là những loại tế bào cấu thành MTĐT, được điều khiển bởi các tín hiệu điện biểu diễn các giá trị logic 0 và 1. Mức logic số chính là cái máy vật lý, là phần cứng của MTĐT.

Chương 4: Mức vi chương trình. Máy vật lý bao gồm nhiều đơn vị chức năng khác nhau, mỗi đơn vị này thực hiện một vài thao tác hết sức đơn giản, được coi là sơ cấp. Chương này nghiên cứu cách thức điều khiển và phối hợp hoạt động của các đơn vị chức năng, để máy vật lý thực hiện được các thao tác sơ cấp một cách hiệu quả nhất.

Chương 5: Mức máy thông thường. Tại mức này có một ngôn ngữ thường được các nhà sản xuất MTĐT gọi là "Ngôn ngữ máy". Thực chất đây không phải là ngôn ngữ để "nói chuyện" trực tiếp với phần cứng của MTĐT, mà nó được xây dựng dựa trên các vi chương trình, được nhà sản xuất nhúng sẵn vào trong phần cứng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

(firmware). Mục đích của việc này là để đưa ra cho người lập trình một ngôn ngữ để hiểu hơn so với ngôn ngữ lập vi chương trình.

Chương 6: Mức máy hệ điều hành. Việc "nói chuyện" với MTĐT bằng ngôn ngữ Mức máy thông thường vẫn còn phức tạp, nhất là khi chúng ta muốn máy thực hiện các thao tác vào/ra. Hệ điều hành sẽ khắc phục vấn đề đó, ngoài ra nó còn thực hiện chức năng quản lý bộ nhớ và các tiến trình.

Chương 7: Mức ngôn ngữ assembly. Chương này tác giả chỉ giới thiệu sơ lược vì các bạn đọc là sinh viên ngành Công nghệ Thông tin chắc chắn sẽ có một môn học bắt buộc về lập trình bằng ngôn ngữ assembly.

Chương 8: Trình bày về nguyên lý hoạt động của một số thiết bị ngoại vi thông dụng như bàn phím, màn hình, máy in, ổ đĩa.

Chương 9: Trình bày khá chi tiết về máy vi tính IBM PC, trong đó có các thành phần cơ bản của hệ điều hành DOS và cách DOS tổ chức thông tin trên đĩa. Chương này liên quan đến các kiến thức cụ thể, mang nhiều tính ứng dụng thực tế.

Giáo trình KTMT này đã được sử dụng làm tài liệu giảng dạy từ năm 1989 ở Khoa Toán-Cơ-Tin học và sau đó là Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên thuộc Đại học Quốc gia Hà Nội. Từ năm 1998 đến nay, giáo trình được sử dụng cho sinh viên ngành Công nghệ Thông tin, Khoa Công nghệ nay là Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội.

Tác giả xin bày tỏ sự biết ơn giáo sư Nguyễn Khang Cường đã cho nhiều nhận xét, chỉ bảo xác đáng trong quá trình biên soạn lần thứ nhất, năm 1998. Tác giả cũng xin cám ơn giảng viên Hồ Đắc Phương, Khoa Công nghệ, Đại học Quốc gia Hà Nội, đã đọc giúp bản thảo và cho nhiều ý kiến đóng góp trong lần xuất bản này.

Mặc dù đã cố gắng nhưng do kiến thức còn hạn hẹp, chắc chắn không tránh khỏi việc vẫn còn thiếu sót. Tác giả chân thành mong muốn nhận được các ý kiến đóng góp của các thầy, các bạn đồng nghiệp, các bạn sinh viên và bạn đọc về mọi mặt của giáo trình.

Các ý kiến đóng góp xin gửi về địa chỉ:

Nguyễn Đình Việt, Trường Đại học Công nghệ, ĐHQGHN

144 Đường Xuân Thuỷ, Quận Cầu Giấy, Hà Nội

Tác giả

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 1 MỞ ĐẦU

1.1 NHỮNG KHÁI NIỆM VÀ NGUYÊN LÝ CƠ BẢN

1.1.1 Máy tính

Máy tính theo nghĩa chung nhất dùng để chỉ mọi phương tiện được sử dụng để thực hiện các phép biến đổi toán học, thí dụ bàn tính, máy tính bỏ túi, máy tính điện tử... Tuy nhiên trong thời đại ngày nay, khi nói đến máy tính người ta thường ngầm chỉ máy tính điện tử là máy tính được cấu tạo bởi các linh kiện điện tử.

1.1.2 Nguyên lý xây dựng và sự phân loại máy tính điện tử

1.1.2.1 Các nguyên lý xây dựng máy tính điện tử

Máy tính điện tử làm việc theo hai nguyên lý cơ bản, đó là nguyên lý số và nguyên lý tương tự.

Nguyên lý số (Digital): sử dụng các trạng thái rời rạc của một đại lượng vật lý để biểu diễn số liệu, nguyên lý này còn được gọi là nguyên lý đếm.

Thí dụ về tình trạng rời rạc của đại lượng vật lý theo nguyên lý số được thể hiện trong bảng 1-01.

Bảng 1-01

Linh kiện	Đại lượng vật lý	Trạng thái 1	Trạng thái 2
Chuyển mạch điện tử	Dòng điện	Có (nối mạch)	Không (ngắt mạch)
Lõi ferit	Trường từ tính	Tồn tại	Đảo từ (đảo hướng)
Điôt/transistor	Dòng điện	Dẫn điện	Không dẫn điện

Nguyên lý tương tự (Analog): sử dụng một đại lượng vật lý biến đổi liên tục để biểu diễn số liệu, nguyên lý này còn có tên gọi là nguyên lý đo.

Thí dụ về đại lượng vật lý biến đổi liên tục theo nguyên lý tương tự được thể hiện trong bảng 1-02.

Bảng 1-02

Thiết bị	Đại lượng vật lý
Thước tính	Chiều dài
Máy tính điện tử tương tự	Điện áp

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

1.1.2.2 Phân loại máy tính điện tử

Có nhiều phương pháp và cách phân loại khác nhau, ở đây ta nêu phương pháp phân loại theo nguyên lý xây dựng máy tính điện tử. Theo phương pháp này máy tính được phân chia thành hai lớp lớn là máy tính tương tự và máy tính số. Mỗi lớp lớn này lại có thể được chia thành các lớp con, thí dụ máy tính đa năng và máy tính chuyên dụng...

Máy tính số (digital computer)

Máy tính số là loại máy tính sử dụng các đại lượng vật lý biến thiên rời rạc (dạng số) để biểu diễn các đại lượng cần tính toán. Những thông số cơ bản của máy tính số là: tốc độ hoạt động, hệ thống lệnh và số địa chỉ của các lệnh, các thiết bị nhớ và dung lượng tin của chúng, tổ hợp các thiết bị vào/ra số liệu, kích thước.v.v.

Người ta có thể phân loại máy tính số dựa trên một số cơ sở khác nhau, đó có thể là cách thức thi hành chương trình, là nhiệm vụ mà người thiết kế định ra cho máy tính.v.v. Sau đây là hai thí dụ.

1. Phân loại máy tính số (MTS) theo cách thức thi hành chương trình:

- MTS liên tiếp: Là MTS trong đó chương trình được thi hành từng lệnh một, hết lệnh này đến lệnh khác.
- MTS song song: Là MTS có thể thi hành đồng thời nhiều chương trình. MTS song song cần nhiều trang thiết bị hơn và phức tạp hơn MTS liên tiếp nhưng có tốc độ tác động cao hơn.
- MTS liên tiếp-song song: Là loại MTS trung gian giữa hai loại MTS nêu trên, trong đó các phép tính theo mã của chương trình được liên tiếp đưa vào các bộ phận của máy, còn các bộ phận thi hành các phép tính một cách song song.

Ngày nay trong tất cả các máy tính, kể cả loại máy tính được gọi là liên tiếp, người ta cũng áp dụng các cơ chế thực hiện song song ở các mức độ khác nhau để nâng cao tốc độ hoạt động chung của máy tính điện tử.

2. Phân loại máy tính số theo nhiệm vụ mà người thiết kế định ra cho nó:

- MTS chuyên dụng: Là loại MTS được chế tạo ra để giải một loại bài toán nhất định, nó thường đơn giản và rẻ hơn MTS đa năng nhờ việc có thể giảm bớt một số thành phần của máy và thậm chí cả việc rút gọn tập lệnh của bộ xử lý của máy.

Các MTS chuyên dụng được sử dụng rộng rãi hiện nay là:

- Các máy tính điều khiển, nằm trong các hệ thống điều khiển những đối tượng và quá trình thực.
- Các máy tính để giải một loại bài toán nhất định, chẳng hạn tính tổng các tích 2 số trong các bài toán về dự báo khí tượng, giải hệ phương trình vi phân thường...

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Các máy tính tìm thông tin, dùng để xử lý những lượng lớn các thông tin chữ và số.

- MTS đa năng: Là loại MTS được chế tạo ra để giải một lớp lớn các bài toán mà thành phần của lớp bài toán có thể còn chưa được nêu đầy đủ khi thiết kế máy.

  Những tính chất cơ bản của MTS đa năng là:

  - Có tập lệnh tương đối lớn, bao gồm nhiều nhóm, mỗi nhóm phục vụ cho một loại thao tác nhất định, để có thể thi hành nhiều loại chương trình khác nhau một cách có hiệu quả.
  - Kích thước các thanh ghi trong bộ xử lý và độ lớn của một đơn vị bộ nhớ (mà sau này chúng ta sẽ gọi là word) phải không quá lớn cũng không quá nhỏ, thường bằng bội số của 8 bit. Như vậy chúng có thể chứa vừa vặn một số ký tự ASCII 8 bit hoặc chứa được con số đủ lớn cho hầu hết các chương trình ứng dụng sẽ chạy trên máy này. Các chương trình soạn thảo văn bản, chương trình quản lý.v.v. nói chung đều có dữ liệu kiểu ký tự.
  - Các thiết bị nhớ có thể nhận vào, lưu trữ và đưa ra các số liệu, các kết quả và các chương trình... Còn trong MTS chuyên dụng các chương trình, hằng số... thường được cứng hoá.
  - Hệ thống trao đổi thông tin (giao diện) giữa máy và người thuận tiện, giảm nhẹ các công việc mà con người phải thực hiện trong quá trình máy tính hoạt động.

Máy tính tương tự (Analog Computer)

Máy tính tương tự (MTTT) là loại máy tính sử dụng các đại lượng vật lý biến thiên liên tục để biểu diễn các đại lượng cần tính toán. Đại lượng vật lý đó thường là điện áp hoặc dòng điện.

Mô hình hoá (modelling) là cơ sở cho sự hoạt động của MTTT, trong đó một quá trình vật lý thực sự hay các thành phần của nó được thay thế bằng một mô hình điện có các đặc tính tương tự. Nhờ có mô hình điện mà việc nghiên cứu có thể tiến hành đơn giản, thuận tiện và rẻ hơn.

Các MTTT vận hành rất thuận tiện, thường đưa ra kết quả dưới dạng đồ thị, đặc biệt là với thời gian cực kỳ ngắn.

MTTT có các nhược điểm chính sau: kết quả có độ chính xác không cao lắm, sự hoạt động của nó không mềm dẻo như MST, khả năng giải bài toán phụ thuộc mạnh vào chính phần cứng của máy.

Sự khác nhau cơ bản nhất giữa MTTT và MTS là MTS chỉ làm được các phép tính số học cổ điển như cộng, trừ, nhân, chia; để thực hiện các tổ hợp gồm các phép tính cộng và nhân.v.v. những cái mà bộ cộng của MTTT chỉ phải làm trong nháy mắt thì ở MTS phải có một chương trình đặc biệt để sắp xếp dần dần các phép tính số học chủ yếu thành các tổ hợp cần thiết.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Các MTS ngày nay có khả năng giải được hầu như mọi loại bài toán toán học và các tính toán logic phức tạp với tốc độ thực hiện, độ chính xác và độ tin cậy ngày càng cao. Chính vì vậy mà MTTT với các nhược điểm trên chỉ được sử dụng cho những ứng dụng có tính chất chuyên biệt, không phổ biến như MTS.

Máy tính lai (Hybrid Computer)

Đó là loại máy tính kết hợp cả hai nguyên lý số và tương tự, trong hệ thống này có một nửa là số và một nửa là tương tự. Nửa số về thực chất là một máy tính số hoặc là một tập hợp các phần tử tính toán số. Nửa tương tự là một máy tính tương tự hoặc là một tập hợp các phần tử tính toán tương tự. Trong quá trình tính toán, hai nửa này truyền dữ liệu cho nhau thông qua các bộ chuyển đổi (converter). Việc đồng bộ hoạt động của hai nửa có thể do một đơn vị điều khiển riêng hoặc do đơn vị điều khiển của máy tính số đảm nhiệm.

1.1.3 Ngôn ngữ máy, mức và máy ảo

- **Lệnh máy, ngôn ngữ máy và chương trình**

Các mạch điện tử của máy tính có thể hiểu và thực hiện trực tiếp được một tập hợp hữu hạn các lệnh rất đơn giản, chẳng hạn:

- Cộng hai số với nhau.
- Kiểm tra xem một số có bằng không hay không.
- Vận chuyển một nhóm dữ liệu từ vùng này của bộ nhớ sang một vùng khác.

Các lệnh rất đơn giản này thường được gọi là **chỉ thị** (instruction) **máy** hay **lệnh máy**. Tập các chỉ thị rất đơn giản đó tạo nên một ngôn ngữ, con người có thể sử dụng ngôn ngữ này để giao tiếp với máy tính. Một ngôn ngữ như vậy được gọi là **ngôn ngữ máy (machine language).** Một dãy các chỉ thị mô tả việc thực hiện một nhiệm vụ cụ thể như thế nào được gọi là **chương trình (program).**

(Trong giáo trình này hai thuật ngữ lệnh và chỉ thị thường được sử dụng với nghĩa gần giống nhau. Thuật ngữ lệnh được sử dụng với nghĩa chung nhất, còn thuật ngữ chỉ thị thường dùng để chỉ các lệnh rất đơn giản. Tuy nhiên lệnh nào được coi là rất đơn giản lại tuỳ thuộc vào mức máy mà chúng ta đang nghiên cứu.)

Khi nghiên cứu thiết kế một máy tính mới, người thiết kế phải quyết định ngôn ngữ máy của máy tính đó sẽ bao gồm những chỉ thị nào. Thông thường để làm giảm độ phức tạp và giá thành của các mạch điện tử, người ta cố gắng làm cho các chỉ thị càng đơn giản càng tốt, phù hợp với mục đích sử dụng của máy tính này và các đòi hỏi về hiệu suất máy.

- **Sự cần thiết của các ngôn ngữ cao hơn ngôn ngữ máy và vấn đề dịch**

Vì hầu hết các ngôn ngữ máy rất đơn giản, nên việc sử dụng chúng là khó và tẻ nhạt. Có thể khắc phục vấn đề này bằng cách thiết kế một tập chỉ thị mới dễ sử dụng hơn tập chỉ thị máy đã được xây dựng ngay bên trong máy. Tập các chỉ thị này cũng tạo thành một ngôn ngữ mà chúng ta sẽ gọi là ngôn ngữ mức (Level) 2 -

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

L2, còn tập chỉ thị được xây dựng ngay bên trong máy chúng ta sẽ gọi là ngôn ngữ L1.

Có hai cách để máy tính thi hành được các chương trình viết bằng L2, mà xét cho cùng về thực chất máy tính cũng chỉ có thể thực hiện được các chương trình được viết bằng ngôn ngữ máy của nó là ngôn ngữ L1.

Cách thứ nhất để máy tính thi hành chương trình được viết bằng L2 là đầu tiên thay thế mỗi chỉ thị trong chương trình này bằng một dãy tương đương gồm các chỉ thị trong ngôn ngữ L1. Kết quả thu được một chương trình gồm toàn các chỉ thị thuộc ngôn ngữ L1. Sau đó máy tính thực hiện chương trình mới bằng ngôn ngữ L1 chứ không phải chương trình cũ bằng ngôn ngữ L2. Kỹ thuật này được gọi là **dịch** hay **biên dịch - Compilation** (hay **translation**), còn chương trình thực hiện việc này được gọi là Trình biên dịch - **Compiler**.

Cách thứ hai là viết một chương trình bằng ngôn ngữ L1, có nhiệm vụ làm cho máy tính thi hành chương trình bằng ngôn ngữ L2. Nó lấy chương trình L2 làm dữ liệu vào, đọc và kiểm tra từng chỉ thị L2 một và thực hiện một dãy tương đương các chỉ thị L1 một cách trực tiếp. Cách này không đòi hỏi việc trước tiên phải sinh ra một chương trình mới bằng ngôn ngữ L1, nó được gọi là **thông dịch - interpretation**, còn chương trình thực hiện việc này có tên gọi là Trình thông dịch - **interpreter**.

Biên dịch và thông dịch giống nhau ở chỗ trong cả hai phương pháp, các chỉ thị bằng ngôn ngữ L2 cuối cùng được thực hiện bằng cách thi hành một dãy tương đương các chỉ thị L1. Điều khác nhau là ở chỗ, trong phương pháp dịch, toàn bộ chương trình L2 ban đầu được biến đổi thành chương trình L1, còn chương trình L2 thì có thể bỏ đi, sau đó chương trình L1 mới sinh ra này được thi hành. Trong phương pháp thông dịch, sau khi từng chỉ thị L2 được kiểm tra và giải mã (decoded), nó sẽ được thi hành ngay lập tức, không có việc sinh ra một chương trình L1 mới.

Cả hai phương pháp kể trên đều được sử dụng rộng rãi.

Để cho thuận tiện, chúng ta hình dung có một máy tính tưởng tượng hay gọi là máy tính ảo (virtual machine), mà ngôn ngữ máy là L2. Nếu có thể xây dựng một máy tính như vậy với giá đủ rẻ, thì chúng ta không cần phải có ngôn ngữ L1 và cái máy để thực hiện các chương trình bằng ngôn ngữ L1 nữa, chúng ta chỉ việc viết các chương trình bằng ngôn ngữ L2 và cho máy tính thực hiện trực tiếp. Việc xây dựng máy tính có ngôn ngữ L2 bằng các mạch điện tử là rất đắt, không khả thi, tuy nhiên chúng ta vẫn có thể viết các chương trình cho chiếc máy ảo này. Các chương trình này có thể được thông dịch hoặc dịch bằng một chương trình được viết bằng ngôn ngữ L1, chính chương trình này được thi hành trực tiếp bởi cái máy tính đang tồn tại. Nói theo cách khác, người ta có thể viết chương trình cho các máy tính ảo, cứ như thể chúng tồn tại thực sự.

Để có thể thực hiện được trên thực tế việc dịch và thông dịch, các ngôn ngữ L1 và L2 không được khác nhau nhiều quá. Điều hạn chế này thường có nghĩa là L2

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

mặc dầu tốt hơn L1, nhưng nó cũng chưa thể là một ngôn ngữ lý tưởng để viết hầu hết các chương trình ứng dụng.

Một điều rất tự nhiên là chúng ta có thể tạo ra một tập chỉ thị nữa thuận lợi hơn cho con người so với tập chỉ thị L2. Tập thứ ba này cũng tạo thành một ngôn ngữ, chúng ta sẽ gọi nó là L3. Sau đó chúng ta có thể viết chương trình bằng ngôn ngữ L3 như thể có một máy tính ảo với ngôn ngữ máy L3 tồn tại thực sự. Các chương trình như vậy có thể hoặc là được dịch sang ngôn ngữ L2 hoặc được cho thi hành bằng một trình thông dịch viết bằng ngôn ngữ L2.

Theo ý tưởng trên chúng ta có thể tạo ra cả một dãy các ngôn ngữ, mỗi ngôn ngữ sử dụng ngôn ngữ đứng trước nó làm cơ sở, ngôn ngữ ra đời sau thuận tiện hơn ngôn ngữ ra đời trước nó, cứ tiếp tục mãi cho tới khi có được một ngôn ngữ thích hợp. Vì vậy chúng ta có thể coi một máy tính được xây dựng theo ý tưởng này như một dãy các lớp hay các mức, lớp này nằm trên lớp kia, như minh hoạ ở hình 1-01. Ngôn ngữ (hay lớp) dưới cùng là đơn giản nhất còn ngôn ngữ ở cao nhất là phức tạp nhất.

Giải thích:

Các chương trình viết bằng Ln hoặc là được thông dịch bởi một trình thông dịch chạy trên một máy nằm bên dưới, hoặc là được dịch sang ngôn ngữ máy của một máy tính nằm bên dưới.

Các chương trình viết bằng L3 hoặc là được thông dịch bởi một trình thông dịch chạy trên M2 hoặc M1, hoặc là được dịch sang ngôn ngữ máy L2 hoặc L1.

Các chương trình viết bằng L2 hoặc là được thông dịch bởi một trình thông dịch chạy trên M1, hoặc là được dịch sang ngôn ngữ máy L1.

Các chương trình viết bằng L1 được thực hiện trực tiếp bằng các mạch điện tử.

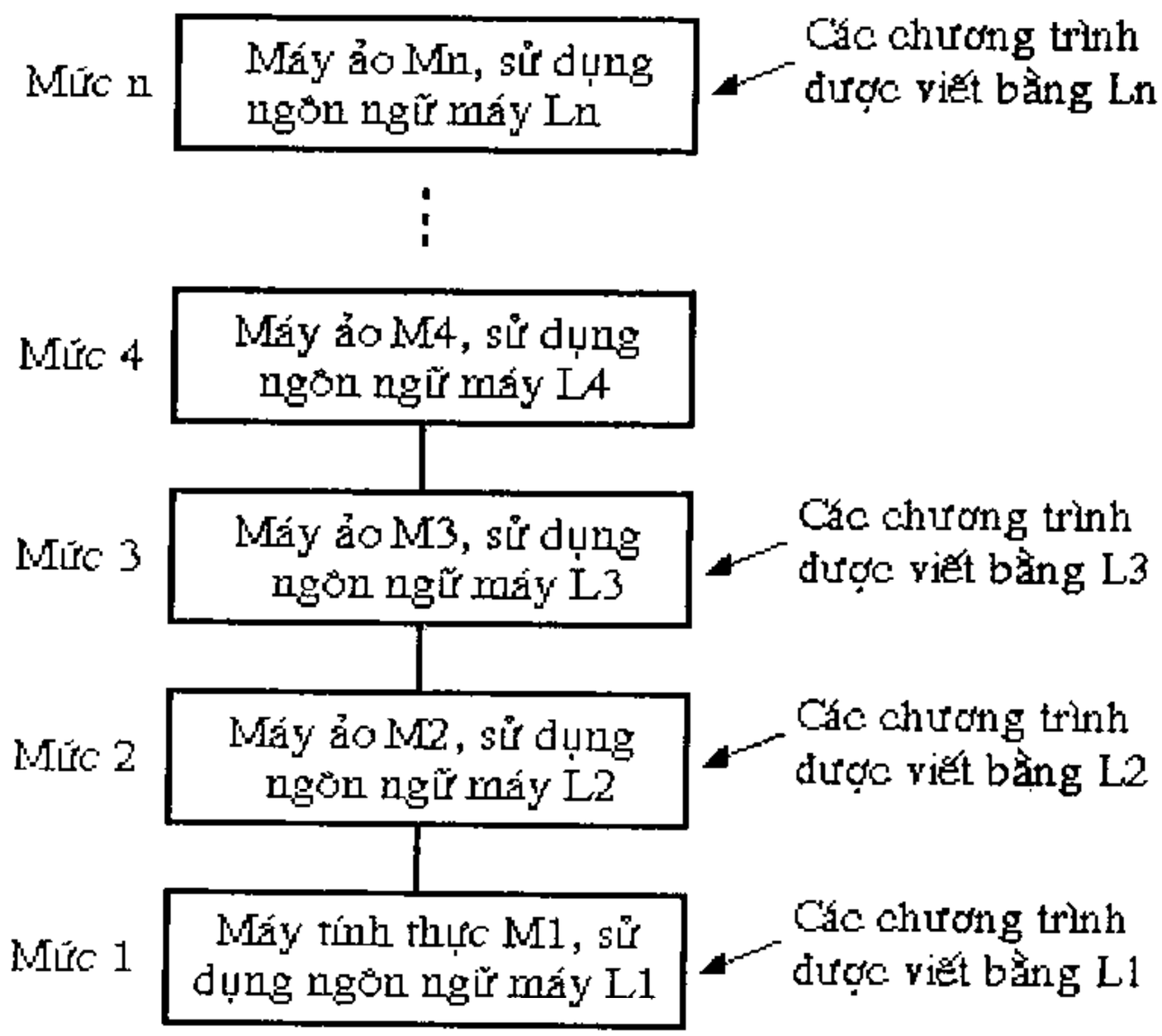
Giữa ngôn ngữ và máy có một mối liên hệ quan trọng, mỗi máy có một ngôn ngữ máy nào đó, bao gồm tất cả các chỉ thị mà máy có thể thi hành. Máy xác định ngôn ngữ và một ngôn ngữ cũng xác định một máy, đó là cái máy có thể thi hành mọi chương trình được viết bằng ngôn ngữ này.

Nếu xây dựng một máy với một ngôn ngữ nhất định trực tiếp bằng các mạch điện tử thì có thể sẽ rất phức tạp và đắt tiền, tuy nhiên chúng ta vẫn có thể hình dung ra được chiếc máy này. Vào thời điểm hiện nay việc xây dựng một máy có ngôn ngữ máy là ngôn ngữ C, Pascal hoặc COBOL có thể coi là quá khó và phi thực tế nhưng rất có thể trong một tương lai không xa những máy tính kiểu như vậy sẽ ra đời.

Một máy với n mức có thể được xem như n máy ảo khác nhau, mỗi máy ảo có một ngôn ngữ máy tương ứng. **Chúng ta sẽ sử dụng thuật ngữ "mức" và "máy ảo" với cùng một ý nghĩa**. Chỉ có các chương trình được viết bằng ngôn ngữ L1 là có thể được thực hiện trực tiếp bằng các mạch điện tử mà không cần sự can thiệp trung gian của việc dịch hoặc thông dịch. Các chương trình được viết bằng L2, L3,..., Ln hoặc là phải được thông dịch bằng một trình thông dịch chạy ở một mức

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

thấp hơn hoặc là phải được dịch sang một ngôn ngữ khác tương ứng với một mức thấp hơn.



Hình 1-01 Một máy tính nhiều mức

Một người làm công việc viết chương trình cho máy tính ảo mức n không nhất thiết phải biết các trình thông dịch hoặc trình dịch nằm ở mức bên dưới. Cấu trúc của máy đảm bảo rằng các chương trình này sẽ được thi hành bằng một cách nào đó. Người viết chương trình không phải bận tâm nhiều về việc chương trình sẽ được thực hiện trực tiếp bởi các mạch điện tử hay là chúng sẽ được thực hiện bởi một trình thông dịch, còn chính trình thông dịch này cũng sẽ lại được thực hiện bởi một trình thông dịch khác. Hầu hết những nhà lập trình sử dụng máy tính mức n chỉ quan tâm đến mức trên cùng, là mức khác xa nhất so với mức ngôn ngữ máy ở dưới cùng. Tuy nhiên những người muốn hiểu biết máy tính thực sự hoạt động như thế nào phải nghiên cứu tất cả các mức. Những người quan tâm đến việc nghiên cứu thiết kế các máy tính mới hoặc là thiết kế các mức mới, nghĩa là máy tính ảo mới phải làm quen với tất cả các mức chứ không phải chỉ riêng mức trên cùng.

1.1.4 Các máy tính nhiều mức hiện đại

Hầu hết các máy tính hiện đại có từ hai mức trở lên. Các máy tính có đến 6 mức như minh hoạ ở hình 1-02 không phải là hiếm. Mức 0 nằm dưới cùng là phần cứng thực sự của máy. Các mạch điện của máy này thực hiện các chương trình ngôn ngữ máy ở mức 1. Thực tế thì còn có một mức nữa nằm bên dưới mức 0, mức này không được vẽ trên hình 1-02 bởi vì nó thuộc lĩnh vực điện tử, không thuộc nội dung giáo trình này, mức đó có tên gọi là **mức thiết bị (device-level)**. Tại mức này người thiết

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

kế nhìn thấy từng transistor, đó là mức thấp nhất đối với những người thiết kế máy tính. Tất nhiên, cũng có thể có những người muốn biết sự hoạt động bên trong của transistor, vấn đề này thuộc lĩnh vực nghiên cứu của vật lý chất rắn!

Mức 0 - mức logic số

Tại mức thấp nhất mà chúng ta sẽ nghiên cứu là mức logic số, các đối tượng chúng ta quan tâm được gọi là các **cổng - gates**. ***Mặc dầu được xây dựng từ các phần tử tương tự, chẳng hạn các transistor, cổng có thể được mô hình hoá một cách chính xác như là các thiết bị số.*** Mỗi cổng có một hoặc một số lối vào tín hiệu số (là các tín hiệu biểu diễn cho các giá trị 0 và 1), nó tính toán để đưa ra một giá trị là một hàm đơn giản của các tín hiệu đầu vào, chẳng hạn hàm logic AND hoặc OR.

Mỗi cổng được xây dựng từ một số transistor. Chúng ta sẽ nghiên cứu chi tiết mức logic số ở chương 3.

Mức 1 - Mức vi chương trình

Mức 1 là mức ngôn ngữ máy thực sự, khác với mức 0, tại đó không có khái niệm về chương trình là một dãy các chỉ thị để thực hiện. Tại mức 1 có một chương trình rõ ràng được gọi là **vi chương trình - microprogram**, nhiệm vụ của nó là thông dịch các chỉ thị ở mức 2. Chúng ta sẽ gọi mức 1 là mức **vi chương trình (microprogramming level)**.

Trên thực tế mức vi chương trình của các máy tính khác nhau có những điểm căn bản giống nhau, tuy không giống nhau hoàn toàn. Thí dụ, một số máy có trên 20 chỉ thị ở mức này và hầu hết trong số đó là các chỉ thị liên quan đến vận chuyển dữ liệu từ vùng này tới vùng khác của máy, hoặc là thực hiện một vài phép kiểm tra đơn giản.

Mức 2 - Mức máy tính thông thường

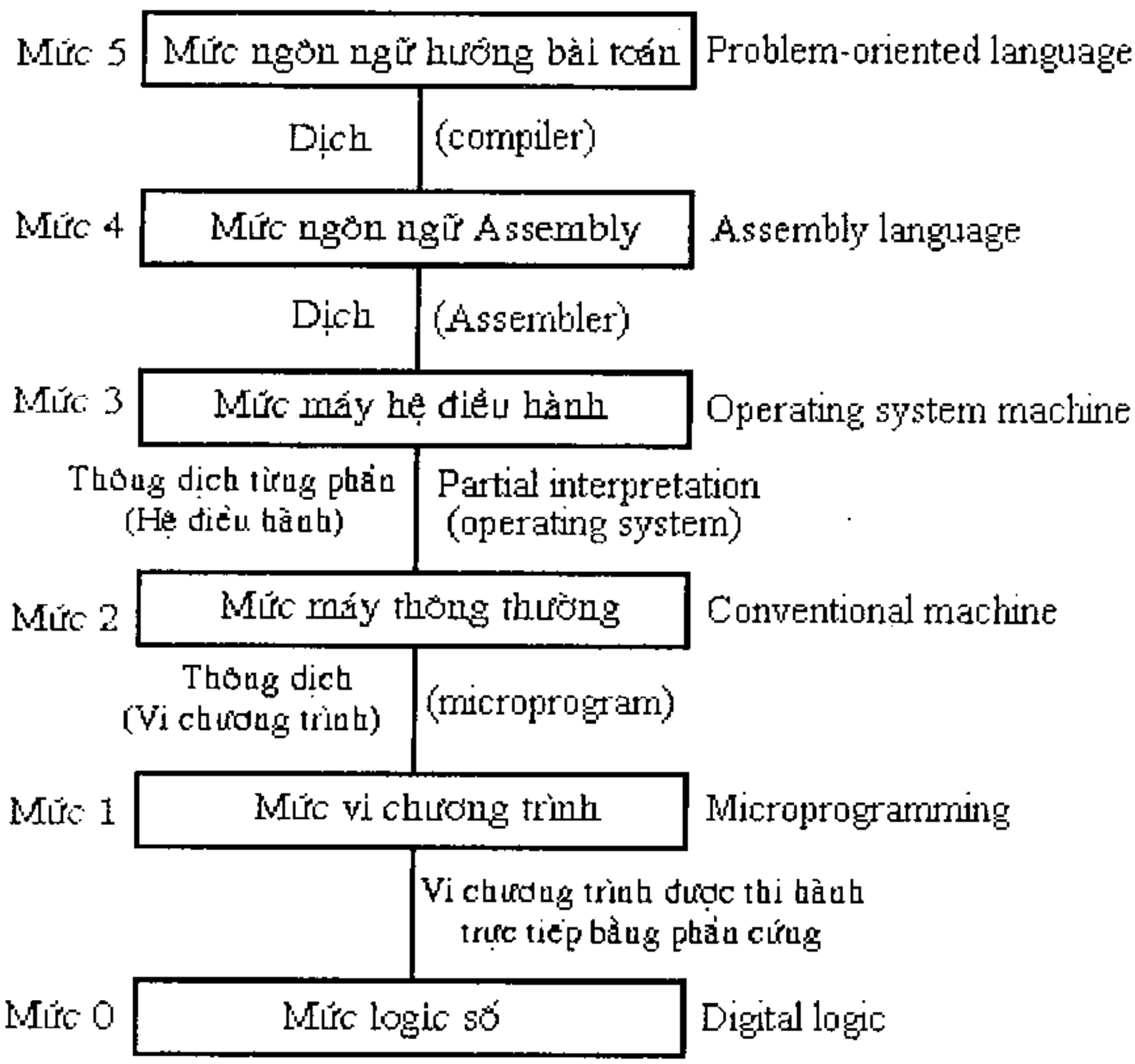
Mỗi máy tính mức 1 có một hoặc một vài vi chương trình có thể chạy trên đó. Mỗi một vi chương trình xác định hoàn toàn một ngôn ngữ mức 2 (L2) và một máy ảo với ngôn ngữ máy là L2. Máy mức 2 của các nhà sản xuất khác nhau có nhiều điểm giống nhau.

Trong nhiều tài liệu nghiên cứu thuộc lĩnh vực kiến trúc máy tính, mức này được gọi là **mức máy tính thông thường** (conventional machine). Với mỗi loại máy tính bán ra, các nhà sản xuất thường công bố một cuốn sách với nhan đề "Sổ tay tra cứu ngôn ngữ máy" ("Machine Language Reference Manual") hoặc là một cái tên khác nào đó kiểu như vậy. Những cuốn sách này thực sự viết về các máy tính ảo mức 2 và ngôn ngữ L2, chứ không phải là máy tính thực sự mức 1 được tạo bởi các mạch điện tử và ngôn ngữ L1. Khi họ mô tả tập chỉ thị của máy, trên thực tế là họ mô tả các chỉ thị được thi hành theo cách thông dịch bằng vi chương trình, chứ không phải là chính tập chỉ thị cho phần cứng. Nếu nhà sản xuất máy tính cung cấp hai trình thông dịch dùng cho một trong các máy tính của họ, để thông dịch hai

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

ngôn ngữ máy mức 2 khác nhau, họ cũng cần phải cung cấp hai cuốn "Sổ tay tra cứu ngôn ngữ máy", mỗi cuốn dùng cho một trình thông dịch.

Cũng cần phải nhấn mạnh rằng một số máy tính không có mức vi chương trình. Trong các máy này các chỉ thị mức máy thông thường được thực hiện trực tiếp bằng các mạch điện tử (mức 0) mà không cần sự tham gia của một trình thông dịch đứng trung gian ở mức 1. Hệ quả là, mức máy thông thường là mức 1 chứ không phải là mức 2; Dù có ngoại lệ này chúng ta vẫn gọi mức máy thông thường là Mức 2.



Hình 1-02 Sáu mức thường có trong các máy tính hiện đại. (Phương pháp hỗ trợ cho từng mức được ghi ở phía dưới mỗi mức, tên của chương trình hỗ trợ được ghi trong dấu ngoặc đơn)

Mức 3 - Mức máy hệ điều hành

Mức 3 thường là một mức lai, hầu hết các chỉ thị có trong ngôn ngữ của nó (L3) cũng có trong ngôn ngữ mức 2 (L2). Người ta bổ sung thêm vào đó một tập các chỉ thị mới, một cách tổ chức bộ nhớ khác, làm cho nó có khả năng cho chạy song song hai hay nhiều chương trình và nhiều đặc tính khác nữa.

Giữa các máy tính khác nhau, tại mức 3 có nhiều điểm khác nhau hơn so với ở mức 1 và mức 2.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Các đặc điểm mới được thêm vào ở mức 3 được thực hiện bởi một trình thông dịch chạy ở mức 2, về mặt lịch sử nó được gọi là **hệ điều hành - operating system**. Một số chỉ thị mức 3 được thông dịch một cách trực tiếp bởi vi chương trình giống như các chỉ thị mức 2; các chỉ thị khác được thông dịch bởi hệ điều hành. Chúng ta đã từng nói mức 3 là một mức lai - "hybrid" chính là nói về điều này. Chúng ta sẽ gọi mức này là **mức máy hệ điều hành** (operating system machine).

Mức 4 - Mức ngôn ngữ assembly

Có một điểm khác nhau cơ bản giữa mức 4 và các mức thấp hơn nó. Ba mức thấp nhất (trừ mức 0, tại đó không có khái niệm về chương trình) được thiết kế không nhằm cho những nhà lập trình thông thường sử dụng trực tiếp. Chúng chủ yếu được nhằm để chạy các trình thông dịch và trình dịch cần thiết cho việc hỗ trợ các mức cao hơn. Các trình thông dịch và trình dịch này được viết bởi những nhà lập trình hệ thống (system programmer), họ chuyên sâu vào việc thiết kế và thực hiện các máy ảo mới. Mức 4 và các mức bên trên nó nhằm phục vụ những người lập trình ứng dụng để giải quyết các bài toán của họ.

Một thay đổi lớn nữa xảy ra ở mức 4 là phương pháp hỗ trợ cho các mức bên trên. Các mức 2 và 3 luôn luôn được thông dịch; Các mức 4, 5 và trên nữa thường (mặc dầu không phải là luôn luôn) được hỗ trợ bằng việc biên dịch. Ngoài ra giữa một bên là các mức 1, 2, 3 và một bên là các mức 4,5 và cao hơn còn có sự nhau ở bản chất của các ngôn ngữ ở các mức này. Các ngôn ngữ máy của các mức 1,2 và 3 là số. Các chương trình viết bằng các ngôn ngữ này chứa một dãy dài các con số, chúng thật là "đẹp" cho cái máy nhưng lại là "xấu" đối với con người. Bắt đầu từ mức 4, các ngôn ngữ gồm các từ (word) và những chữ viết tắt có ý nghĩa đối với con người. Mức 4 được gọi là mức ngôn ngữ assembly, nó thực sự là một dạng ký hiệu (symbolic) cho một trong các ngôn ngữ ở bên dưới. Mức này cung cấp cho những người viết chương trình cho máy mức 1,2 và 3 một ngôn ngữ lập trình dễ sử dụng hơn nhiều so với các ngôn ngữ mức 1, 2 và 3. Các chương trình viết bằng ngôn ngữ assembly đầu tiên được dịch sang ngôn ngữ mức 1, 2 hoặc 3, sau đó được thông dịch bằng một máy thực hay máy ảo phù hợp. Chương trình thực hiện công việc dịch chương trình viết bằng ngôn ngữ assembly sang một ngôn ngữ ở mức thấp hơn được gọi là **assembler**. Ngôn ngữ assembly có một thời là quan trọng, nhưng dần dần nó trở nên kém quan trọng hơn trước.

Mức 5 - Mức ngôn ngữ bậc cao

Mức 5 bao gồm các ngôn ngữ được thiết kế cho những người lập trình ứng dụng sử dụng để giải các bài toán của họ, các ngôn ngữ như vậy được gọi là **ngôn ngữ bậc cao (high-level language)**. Có hàng trăm ngôn ngữ bậc cao từng tồn tại, trong đó có một số ít được nhiều người biết đến là BASIC, C, COBOL, FORTRAN, LISP, Modula 2 và Pascal. Chương trình được viết bằng các ngôn ngữ này thường được dịch sang mức 3 hoặc mức 4 bằng các trình dịch, chúng được gọi là **compiler**, đôi khi chúng cũng được thông dịch.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Bên trên mức 5 có thể còn có một hoặc một số mức cao hơn, mỗi mức này bao gồm một tập hợp các chương trình được thiết kế nhằm tạo ra các máy chuyên dụng cho những ứng dụng nhất định. Chúng ta có thể hình dung ra những máy ảo nhằm cho các ứng dụng quản lý, giáo dục, thiết kế máy tính.v.v. Những mức này là những lĩnh vực nghiên cứu hiện đại.

1.1.5 Kiến trúc máy tính và đối tượng nghiên cứu của nó

Tóm lại vấn đề mấu chốt mà chúng ta cần nhớ là máy tính được thiết kế như một dãy các mức, mỗi mức được xây dựng bên trên mức trước nó. Mỗi mức biểu diễn một sự trừu tượng hoá riêng biệt, trong mỗi mức có những đối tượng và sự hoạt động khác nhau. Bằng việc thiết kế và phân tích máy tính theo cách này, chúng ta có khả năng tạm thời gác lại những chi tiết không thích hợp, làm cho một vấn đề phức tạp trở nên dễ hiểu hơn.

Tập hợp các kiểu dữ liệu, các thao tác và các đặc điểm của mỗi mức được gọi là kiến trúc (architecture). Kiến trúc liên quan tới các đặc điểm của một mức mà người lập trình thấy được, chẳng hạn bộ nhớ có thể sử dụng lớn bao nhiêu là một phần của kiến trúc. Các đặc điểm ở mỗi mức được thực hiện như thế nào, chẳng hạn công nghệ chip nào được sử dụng không phải là một phần của kiến trúc.

Việc nghiên cứu cách thiết kế các phần của hệ thống máy tính mà người lập trình có thể nhận thấy được gọi là **kiến trúc máy tính (Computer architecture)**. Nói chung trong thực tế thuật ngữ **Kiến trúc máy tính (computer architecture) và Tổ chức máy tính (computer organization)** về căn bản nói về cùng một vấn đề.

1.1.6 Phần cứng, Phần mềm và các máy tính nhiều mức

Phần cứng

Các chương trình được viết bằng ngôn ngữ máy của máy tính mức 1 có thể được thi hành trực tiếp bởi các mạch điện tử mà không cần một trình thông dịch hoặc trình dịch trung gian nào cả. Các mạch điện tử như thế, cùng với bộ nhớ và các thiết bị vào/ra tạo thành phần cứng của máy tính, thuật ngữ tiếng Anh là hardware. **Phần cứng bao gồm các đối tượng hữu hình** như cách vi mạch (IC), các bảng mạch in, cáp nối, nguồn điện, bộ nhớ, máy đọc bìa, máy in dòng và terminal chứ không phải là các ý tưởng trừu tượng, các thuật toán hay là các chỉ thị.

Phần mềm

Phần mềm (software) thì trái lại, nó bao gồm các thuật toán và các biểu diễn cho máy tính của chúng, đó chính là các chương trình. Chương trình có thể được biểu diễn trên bìa đục lỗ, trên băng từ, đĩa từ và các môi trường khác, tuy nhiên cái cơ bản nhất của phần mềm chính là tập các chỉ thị tạo nên chương trình chứ không phải là môi trường vật lý được sử dụng để ghi chương trình.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Firmware

Một dạng trung gian giữa phần cứng và phần mềm được gọi là firmware (phần sụn | phần nhão !), nó ***là phần mềm được nhúng vào các mạch điện tử trong quá trình chế tạo các mạch điện tử này.*** Firmware được sử dụng khi các chương trình hiếm khi hoặc không bao giờ cần phải thay đổi, thí dụ trong các đồ chơi hoặc các dụng cụ, máy móc. Firmware cũng được sử dụng khi các chương trình không được phép bị mất đi khi tắt điện. Trong nhiều máy tính, các vi chương trình thuộc Firmware.

Tư tưởng chủ đạo của lĩnh vực nghiên cứu Kiến trúc máy tính là:

Phần cứng và phần mềm là tương đương về mặt logic.

Nói chung một thao tác thực hiện được bằng phần mềm thì cũng có thể xây dựng phần cứng để thực hiện trực tiếp thao tác đó, ngược lại mọi thao tác (hay chỉ thị) được thực hiện bằng phần cứng cũng có thể được mô phỏng bằng phần mềm.

Việc quyết định đưa các chức năng nhất định nào vào phần cứng và các chức năng khác vào phần mềm được dựa trên các yếu tố như giá cả, tốc độ, độ tin cậy và tần suất của những sự thay đổi có thể xảy ra. Không có những quy tắc bắt buộc quy định một cách rõ ràng rằng thao tác X phải được đưa vào trong phần cứng, còn thao tác Y thì phải được thực hiện bằng phần mềm (được lập trình). Những người thiết kế máy tính với các mục tiêu khác nhau có thể và thường quyết định khác nhau.

Trong những máy tính đầu tiên sự tách biệt giữa phần cứng và phần mềm là rõ ràng. Phần cứng thực hiện một số ít chỉ thị đơn giản, chẳng hạn cộng (ADD) và nhảy (JUMP) còn mọi chỉ thị khác được lập trình một cách rõ ràng. Nếu một chương trình cần nhân 2 số, người lập trình phải tự mình viết một chương trình con để nhân, hoặc là mượn nó ở thư viện nếu có ai đó đã xây dựng và đưa vào thư viện. Thời gian trôi đi, những người thiết kế phần cứng nhận thấy rõ rằng một số thao tác được thực hiện khá thường xuyên, cần thiết phải xây dựng những mạch điện phần cứng đặc biệt để thực hiện chúng một cách trực tiếp, làm cho nó được thực hiện nhanh hơn. Kết quả là có một khuynh hướng chuyển các thao tác xuống phần cứng (mức 0). Những cái trước đây được lập trình rất rõ ràng ở mức máy thông thường thì nay thấy được ở bên dưới, trong phần cứng.

Với sự ra đời của kỷ nguyên vi chương trình và các máy tính nhiều mức, một khuynh hướng ngược lại cũng đã xuất hiện. Trong các máy tính đầu tiên, lệnh cộng (ADD) được thi hành bằng phần cứng, không một ai nghi ngờ gì về điều đó. Trong một máy tính được lập vi chương trình, chỉ thị ADD của mức máy thông thường được thông dịch bằng vi chương trình chạy ở mức dưới cùng và được thực hiện như một dãy các bước nhỏ hơn: lấy lệnh (instruction), xác định kiểu của lệnh, định vị dữ liệu sẽ được cộng, lấy dữ liệu từ bộ nhớ, thực hiện phép cộng và cất kết quả. Đây là thí dụ về một chức năng, nó vận chuyển hướng lên phía trên, từ mức phần cứng tới mức vi chương trình.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Khi phát triển một máy tính nhiều mức, người thiết kế phải quyết định sẽ đặt những gì tại mỗi mức, cái gì đặt vào phần cứng và cái gì đặt ở phần mềm. Trong các máy tính hiện đại ngày nay rất nhiều chỉ thị được thực hiện bằng phần cứng hoặc bằng vi chương trình, nhưng khởi thuỷ chúng được lập trình rất rõ ràng tại mức máy thông thường, thí dụ:

1. Các chỉ thị cho các phép nhân và chia số nguyên.
2. Các chỉ thị đối với số dấu phảy động.
3. Các chỉ thị số học có độ chính xác kép (các phép tính số học trên các số có số chữ số có nghĩa nhiều gấp đôi bình thường).
4. Các chỉ thị gọi chương trình con và trở về sau khi thực hiện chương trình con.
5. Các chỉ thị để tăng tốc độ vòng lặp.
6. Các chỉ thị để đếm (cộng 1 vào một biến).
7 Các chỉ thị xử lý dãy ký tự.

...

Vấn đề chủ yếu mà chúng ta vừa bàn luận là để chỉ ra rằng ranh giới giữa phần cứng và phần mềm là tuỳ ý và không ngừng thay đổi. Các phần mềm ngày nay sẽ là phần cứng trong tương lai hoặc ngược lại. Hơn nữa ranh giới giữa các mức khác nhau cũng dễ thay đổi. Theo quan điểm của người lập trình việc một chỉ thị thực tế được thi hành như thế nào không quan trọng, có lẽ trừ một điều là tốc độ thực hiện nó. Một người lập trình ở mức máy thông thường có thể sử dụng chỉ thị nhân của mức này như thể nó là một chỉ thị cho phần cứng mà không phải lo lắng gì cả, thậm chí cũng không cần phải biết nó có thực sự là một chỉ thị cho phần cứng hay không. Phần cứng của người này có thể là phần mềm của người khác.

Sự thực, việc một người lập trình không cần thiết phải hiểu rõ mức mà anh ta đang sử dụng được tạo ra như thế nào dẫn đến ý tưởng về thiết kế máy tính có cấu trúc. Một mức thường được gọi là một máy ảo bởi vì người lập trình nghĩ về nó như là một cái máy vật lý thực sự, mặc dù nó không thực sự tồn tại.

THƯ VIỆN
ĐH DÂN LẬP HP

Bằng việc tạo ra cái máy gồm một dãy các mức, người lập trình làm việc ở mức n không cần thiết phải hiểu rõ mọi chi tiết rối rắm của các mức nằm bên dưới, nhờ đó việc nghiên cứu thiết kế và chế tạo ra các máy tính phức tạp sẽ đơn giản hơn rất nhiều.

SỐ:
BM/OPC

1.2 NHỮNG CỘT MỐC TRONG LĨNH VỰC KIẾN TRÚC MÁY TÍNH

Hàng trăm loại máy tính khác nhau đã từng được thiết kế và chế tạo trong lịch sử tiến hoá của máy tính nói chung và máy tính điện tử số nói riêng; Hầu hết chúng đã bị lãng quên, chỉ còn một số rất ít còn có ảnh hưởng đáng kể đến những tư tưởng hiện đại trong Kiến trúc máy tính. Sau đây là một số cột mốc trong lịch sử phát triển, tiến hoá của kiến trúc máy tính.

THƯ VIỆN ĐH.DÂN LẬP HP
PHÒNG ĐỌC
ĐVL 2893

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

1.2.1 Thế hệ số không - Máy tính cơ khí (1642-1945)

Người đầu tiên chế tạo được một chiếc máy tính hoạt động được là nhà khoa học Pháp Blase **Pascal** (1623-1662). Tên ông đã được đặt cho một ngôn ngữ lập trình - Ngôn ngữ Pascal. Chiếc máy này được chế tạo năm 1642 khi ông 19 tuổi nhằm giúp cho bố mình - một nhân viên thu thuế của chính phủ Pháp. Chiếc máy hoàn toàn là cơ khí, sử dụng các bánh xe răng, năng lượng cung cấp cho máy là sức người - quay tay. Máy tính của Pascal chỉ làm được phép tính cộng và trừ.

Ba mươi năm sau nhà bác học Đức Baron Gottfried von **Leibniz** (1646-1716) đã chế tạo thành công một chiếc máy tính cơ khí khác, ngoài hai phép tính cộng và trừ nó còn có thể thực hiện phép nhân và chia.

Charles **Babbage** (1792-1871) - một giáo sư toán học ở trường đại học tổng hợp Cambridge đã thiết kế và chế tạo chiếc máy tính thực hiện được các phép tính như máy của Pascal để tính các bảng số, sử dụng trong hải quân. Chiếc máy này được thiết kế để chạy một thuật toán duy nhất - phương pháp sai phân hữu hạn sử dụng đa thức. Đặc điểm đáng chú ý nhất của chiếc máy này là phương pháp đưa số liệu ra ngoài: nó đục kết quả lên một tấm đồng - đó là điềm báo hiệu của các thiết bị ghi được một lần (write-once) sau này như: bìa đục lỗ, đĩa quang .v.v.

Mặc dù chiếc máy chạy khá tốt nhưng nó chỉ thực hiện một thuật toán. Babbage đã lao vào nghiên cứu cải tiến nó và đã cho ra đời chiếc máy mới, ông gọi nó là máy phân tích (Analitical Engine), bao gồm 4 phần, ông đặt tên các phần này là:

- "the store" - Bộ nhớ của máy
- "the mill" - Đơn vị tính toán
- "the input section" - Thiết bị vào (máy đọc bìa đục lỗ)
- "the output section" - Thiết bị ra (thiết bị đục bìa)

Đơn vị tính toán nhận các toán hạng từ bộ nhớ, cộng, trừ, nhân hoặc chia chúng rồi lại gửi kết quả ra bộ nhớ.

Đặc điểm nổi bật của chiếc máy tính này là nó đa năng, nó đọc các lệnh từ bìa đục lỗ rồi thực hiện lệnh - do đó nó có thể thực hiện được các công việc tính toán khác nhau.

Chương trình ghi (đục) trên các bìa chính là các 'phần mềm' cho máy. Babbage đã thuê một phụ nữ trẻ tên là Ada Augusta Lovelace làm công việc chuẩn bị 'phần mềm' này - Ada chính là lập trình viên đầu tiên trên thế giới !; Tên của bà sau này đã được đặt cho một ngôn ngữ lập trình hiện đại.

Tư tưởng của Babbage vượt quá xa thời đại của ông, có thể nói rằng *Babbage là ông tổ của máy tính số hiện đại.*

Khoảng năm 1930 một sinh viên Đức tên là Konrad **Zuse** đã chế tạo một loạt máy tính tự động sử dụng rơ-le điện tử, ông không hề biết về các công trình của Babbage. Các máy tính của ông đều bị phe đồng minh ném bom phá huỷ trong chiến tranh thế giới, năm 1944, vì vậy không có ảnh hưởng gì đến các thế hệ máy

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

sau này. Tuy vậy ông vẫn được ghi nhớ như một trong những người đi tiên phong trong lĩnh vực chế tạo máy tính.

1.2.2 Thế hệ thứ nhất - Máy tính dùng đèn điện tử (1945-1955)

Chính chiến tranh thế giới thứ hai đã kích thích sự ra đời của máy tính điện tử. Trong giai đoạn đầu của cuộc chiến tranh, các tàu ngầm Đức trong hoạt động tấn công các tàu Anh đều liên lạc bằng vô tuyến điện với sở chỉ huy ở Berlin, các thông tin đều được mã hoá bằng thiết bị mã hoá có tên là ENIGMA. Phía Anh rất cần giải mã các bản tin thu được, nhưng công việc này đòi hỏi khối lượng tính toán khổng lồ - cần phải có máy tính điện tử. Nhà toán học Anh Alan Turing đã lãnh đạo thiết kế chiếc máy này - chiếc máy tính được đặt tên là **COLOSSUS**; Nó được chính phủ Anh giữ bí mật hoàn toàn trong 30 năm liền.

Năm 1943, John Mauchley và học trò của ông là J. Presper Eckert ở Mỹ đã thuyết phục được quân đội cấp tiền cho việc thiết kế, chế tạo chiếc máy tính điện tử đầu tiên - chiếc máy tính điện tử **ENIAC** (Electronic Numerical Integrator And Calculator), nó bao gồm 18,000 đèn điện tử, 1500 rơ-le, nặng 30 tấn, tiêu thụ công suất điện là 140KW. Về kiến trúc nó có 20 thanh ghi, mỗi thanh ghi chứa được 1 số thập phân 10 chữ số. Chiếc máy được lập trình bằng cách đặt vị trí (set) của 6000 chuyển mạch (switch) - mỗi cái có nhiều vị trí và nối vô số ổ cắm (socket) với một 'rừng' đầu cắm (jumper).

Chiếc máy **ENIAC** mãi đến năm 1946 mới hoàn thành, không đạt được mục đích ban đầu của dự án. Tuy nhiên do chiến tranh đã kết thúc, hai tác giả được phép tổ chức một Trường hè để trình bày về công trình của mình trước các nhà khoa học đồng nghiệp. Chính trường hè này đã gây ra sự bùng nổ về nghiên cứu chế tạo máy tính số cỡ lớn.

Sau trường hè lịch sử đó, có nhiều nhà khoa học khác bắt tay vào xây dựng các máy tính điện tử. Chiếc máy tính đầu tiên đưa vào hoạt động là **EDSAC** (1949), được xây dựng tại trường đại học tổng hợp Cambridge nước Anh, tác giả là Maurice Wilkes. Một số máy khác có thể nêu tên là **JOHNIAC** của công ty Rand, máy tính **ILLIAC** tại trường đại học tổng hợp Illinois, máy tính **MANIAC** của phòng thí nghiệm Los Alamos...

Eckert và Mauchley bắt tay vào việc xây dựng một máy tính mới, có tên là **EDVAC** (Electronic Discrete Variable Automatic Computer), nhưng dự án đó đã bị thất bại nặng nề, sau đó họ rời Pennsylvania đến Philadelphia để thành lập một công ty mới, đó là công ty máy tính Eckert-Mauchley. Sau một loạt các cuộc liên kết, công ty này trở thành công ty Unisys hiện đại.

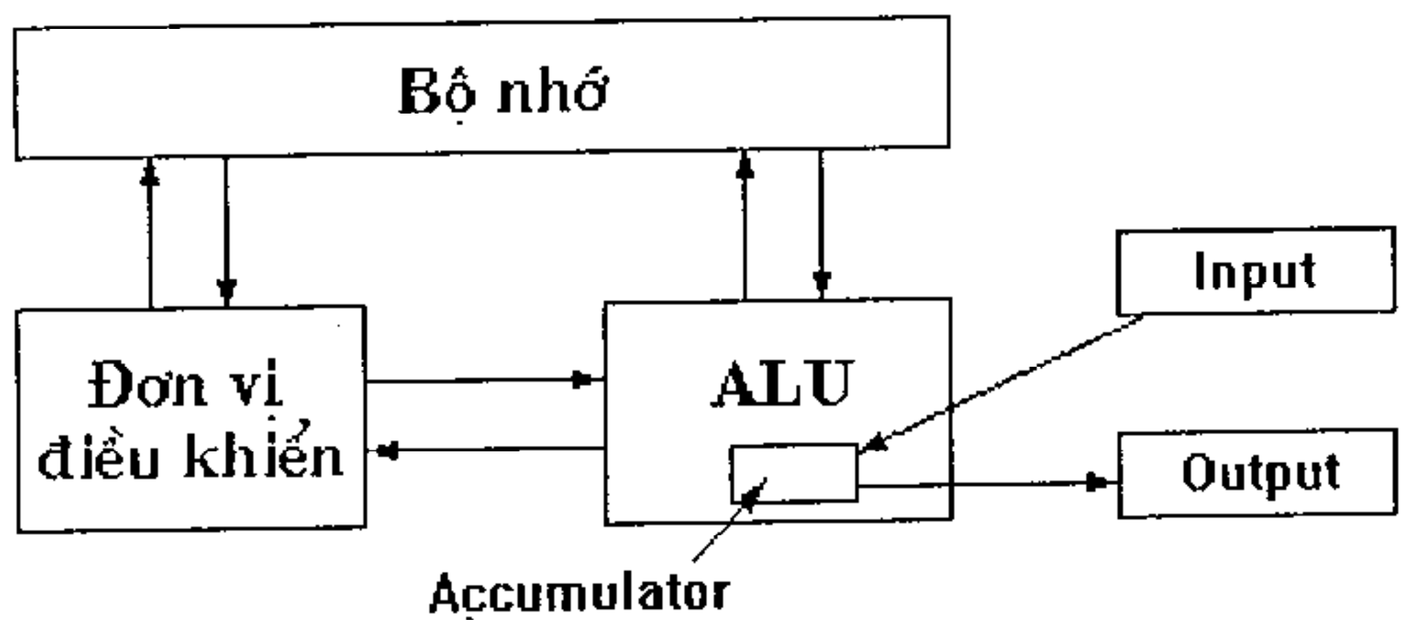
Cùng thời kỳ này, một trong những người bị dự án ENIAC lôi cuốn là John von Neumann, ông đã đến Viện nghiên cứu cao cấp Princeton để xây dựng một phiên bản máy EDVAC của riêng mình, đó là máy **IAS**. Von Neumann là một thiên tài có thể sánh với Leonardo Da Vinci, ông nói được nhiều thứ tiếng, là một chuyên gia về vật lý và toán học, có thể nhớ lại mọi thứ mà ông đã nghe, nhìn thấy hoặc đã đọc.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Ông có khả năng trích dẫn đúng nguyên văn từ trong trí nhớ những đoạn sách mà ông đã đọc từ nhiều năm trước. Khi ông quan tâm đến máy tính, ông đã là một nhà toán học nổi tiếng nhất thế giới.

Một trong những điều mà ông thấy rõ là việc lập trình cho các máy tính có một số rất lớn các switch và cáp là chậm, chán ngắt và không linh hoạt. Ông đi đến nhận xét là chương trình có thể được biểu diễn dưới dạng số và đặt vào trong bộ nhớ của máy tính bên cạnh dữ liệu. Đồng thời ông cũng nhận thấy nhược điểm lớn của cách biểu diễn số trong máy ENIAC; Trong máy này số thập phân được sắp thành dãy, mỗi một chữ số (digit) được biểu diễn bằng 10 đèn điện tử chân không (1 cái on còn 9 cái off). Có thể thay thế cách biểu diễn "vụng về" đó bằng việc sử dụng số nhị phân song song.

Thiết kế cơ sở của ông được gọi là **máy von Neumann** đã được sử dụng trong máy tính EDSAC, ***đó là chiếc máy tính đầu tiên có chương trình được lưu trữ trong bộ nhớ***, cho đến nay vẫn là cơ sở cho hầu hết các máy tính số, cho dù nửa thế kỷ đã trôi qua. Thiết kế này và máy IAS được xây dựng với sự cộng tác của Herman Goldstine đã có một ảnh hưởng rất to lớn, chúng ta sẽ mô tả tóm tắt dưới đây. Khung của kiến trúc được thể hiện trên Hình 1-03.



Hình 1-03 Máy tính von Neumann cổ điển

Máy tính của von Neumann bao gồm 5 thành phần cơ bản là:

- Bộ nhớ (memory)
- Đơn vị số học và logic (ALU - Arithmetic Logical Unit)
- Đơn vị điều khiển chương trình (CU - Control Unit)
- Thiết bị vào (Input)
- Thiết bị ra (Output).

Bộ nhớ gồm 4096 từ, mỗi từ chứa được 40 bit. Như vậy mỗi từ chứa được hoặc là 2 chỉ thị 20 bit hoặc là một số nguyên có dấu dài 39 bit. Các chỉ thị có 8 bit dành riêng để chỉ kiểu của chỉ thị, 12 bit để chứa địa chỉ của 1 trong 4096 từ trong bộ nhớ.

Bên trong ALU, tiền thân của các CPU hiện đại là một thanh ghi 40 bit, có tên gọi là **accumulator**. Có một chỉ thị riêng biệt để cộng một từ của bộ nhớ vào

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

accumulator hoặc chứa nội dung của accumulator vào bộ nhớ. Máy này không thực hiện các phép toán số học dấu phẩy động.

Trong thời kỳ của thế hệ máy tính điện tử thứ nhất IBM còn là một công ty nhỏ, sản xuất các thiết bị đục lỗ... chưa quan tâm gì đến lĩnh vực máy tính điện tử.

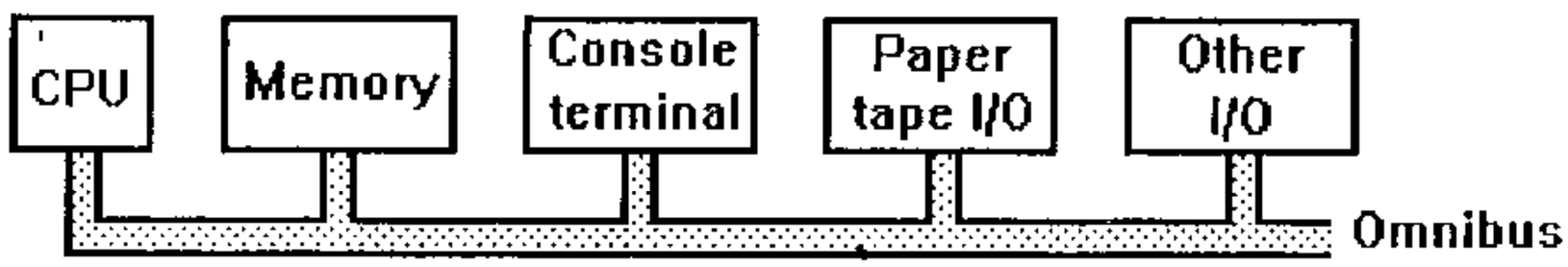
1.2.3 Thế hệ thứ hai - Máy tính dùng Transistor (1955-1965)

Ba nhà khoa học là John Barden, Walter Brattain, và William Shockley đã sáng chế ra transistor ở phòng thí nghiệm Bell năm 1948; Nhờ phát minh này họ đã được tặng giải thưởng Nobel về vật lý năm 1956. Trong khoảng thời gian 10 năm sau đó Transistor đã làm nên một cuộc cách mạng trong lĩnh vực máy tính, cho đến khoảng cuối những năm 1950 các máy tính dùng đèn điện tử đã trở nên lỗi thời.

Chiếc máy tính dùng Transistor đầu tiên được chế tạo ở phòng thí nghiệm Lincoln, nó là máy 16 bit, có tên gọi **TX-0** (Transistorized eXperimental computer 0), nhằm làm một thiết bị để thử nghiệm một máy tính mạnh hơn chính nó là máy tính TX-2.

Máy TX-2 không được lắp ráp nhưng một trong những kỹ sư làm việc ở phòng thí nghiệm là Kenneth Olsen đã thành lập công ty DEC, năm 1957, để sản xuất một máy tính thương mại rất giống TX-0. Vì những nhà tư bản bạo gan góp vốn thành lập DEC tin tưởng chắc chắn rằng sẽ không có thị trường cho máy tính của họ nên DEC hầu như chỉ sản xuất và bán các bảng mạch nhỏ. Bốn năm sau, năm 1961 DEC mới cho ra đời máy PDP-1, PDP-1 có 4K từ 18 bit, một chu kỳ máy là 5μs. Năng suất của máy này bằng nửa năng suất máy 7090 của hãng IBM, là máy tính dùng transistor thế hệ sau của máy 709, đó là chiếc máy tính chạy nhanh chất thế giới vào thời gian đó. Tuy nhiên máy PDP-1 giá chỉ có 120,000$ còn máy IBM 7090 giá hàng triệu đô-la. DEC đã bán được hàng tá máy PDP-1 và công nghiệp sản xuất máy tính mini đã ra đời từ đó.

Một vài năm sau DEC giới thiệu máy PDP-8, đó là một máy tính 12 bit, nhưng có giá rẻ hơn PDP-1 rất nhiều , chỉ còn 16,000$. Máy PDP-8 có một đổi mới chính: chỉ dùng 1 bus, đó là omnibus. Bus là tập hợp các đường dây chạy song song được sử dụng để nối các thành phần của máy tính với nhau. Kiến trúc này xuất phát từ máy IAS có trung tâm là bộ nhớ. Từ đó trở đi hầu hết các máy tính nhỏ đều phỏng theo kiến trúc này. DEC cuối cùng đã bán được 50,000 máy PDP-8, làm cho nó trở thành công ty sản xuất máy tính mini hàng đầu thế giới.



Hình 1-04 Omnibus sử dụng trong máy PDP-8. (Console terminal: Trạm đầu cuối có bàn điều khiển; Paper tape I/O: Thiết bị vào/ra băng giấy; Other I/O: Các thiết bị vào/ra khác)

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Trong thời gian này IBM xây dựng phiên bản transistor hoá của máy 709, đó là máy 7090, như đã đề cập ở trên và sau đó là máy 7094. Máy 7094 có chu kỳ máy 2μs và bộ nhớ chính có 32K từ 36 bit. Các máy 7090 và 7094 đánh dấu sự kết thúc của các máy kiểu ENIAC, nhưng chúng đã thống trị lĩnh vực tính toán khoa học một số năm trong thập kỷ 60.

Đồng thời với việc trở thành một thế lực quan trọng trong lĩnh vực tính toán với máy tính 7094, IBM cũng còn kiếm được rất nhiều tiền bằng việc bán các máy tính cỡ nhỏ hướng phục vụ cho kinh doanh, có tên gọi là 1401. Máy này có thể đọc, ghi băng từ, đọc và đục bìa và in ra kết quả nhanh ngang với 7094, còn giá chỉ bằng một phần nhỏ. Máy này nếu đem dùng cho các tính toán khoa học thì là quá kém nhưng nếu sử dụng cho việc lưu trữ các bản ghi về thương mại thì lại là rất tốt.

Năm 1964, một công ty mới ra đời là CDC giới thiệu máy 6600 trên thị trường, đó là một máy chạy nhanh hơn chiếc IBM 7094 hùng mạnh tới gần một bậc. Trong số các máy tính lớn nó được yêu thích ngay từ đầu và đã thành công trên thị trường. Bí quyết làm cho tốc độ của 6600 nhanh hơn 7094 rất nhiều nằm trong chính CPU, đó là một chiếc máy song song ở mức cao. CPU của 6600 có một số đơn vị chức năng để làm phép tính cộng, một số khác làm phép tính nhân và một số khác nữa làm phép tính chia và tất cả chúng có thể chạy song song. Mặc dầu để tận dụng được hết sức mạnh cần lập trình rất tinh vi nhưng nhờ đó có thể đạt được việc thực hiện 10 chỉ thị đồng thời. Thế vẫn chưa phải là hết, 6600 còn có một số máy tính nhỏ ngay bên trong trợ giúp nó. Điều này có nghĩa là CPU có thể sử dụng toàn bộ thời gian để xử lý các con số, giao tất cả các chi tiết về quản lý công việc và việc vào/ra cho các máy tính nhỏ hơn. Máy CDC 6600 là một cột mốc quan trọng trong lĩnh vực tính toán số.

1.2.4 Thế hệ thứ ba - Máy tính dùng Mạch tích hợp (1965-1980)

Việc phát minh ra mạch tích hợp (còn gọi là mạch vi điện tử - IC) cho phép có thể đặt hàng chục transistor trong một vỏ (chip), nhờ đó người ta có thể chế tạo các máy tính nhỏ hơn, nhanh hơn và rẻ hơn các máy tính dùng transistor ra đời trước nó. Sau đây là một vài loại máy tính nổi bật của thế hệ này.

System/360

Khoảng năm 1964 IBM đã là một công ty máy tính hàng đầu và họ cũng đã vấp phải một vấn đề lớn với 2 máy tính đã rất thành công của mình là 7094 và 1401: chúng hoàn toàn không tương thích gì với nhau cả. Một cái là máy tính số tốc độ cao sử dụng số học số nhị phân song song với các thanh ghi 36 bit, còn một cái là bộ xử lý vào/ra danh tiếng sử dụng số học thập phân liên tiếp trên các từ bộ nhớ có chiều dài thay đổi được. Nhiều khách hàng của công ty có cả hai chiếc máy, nhưng họ không thích gì việc phải có hai phòng lập trình tách biệt, không chung nhau điểm gì. Đã đến lúc phải thay thế hai sê-ri máy này, IBM thực hiện một bước đi căn bản, họ đưa ra một dòng sản phẩm mới, đó là System/360, sử dụng vi mạch, được thiết kế nhằm cho các tính toán khoa học và thương mại.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

System/360 chứa đựng rất nhiều điểm mới, một số điểm mới quan trọng nhất như sau:

- **Tính tương thích cao**

System/360 là cả một họ máy gồm vài thành viên chạy cùng một ngôn ngữ assembly, có kích thước và sức mạnh tăng dần. Phần mềm được viết cho một máy về nguyên tắc có thể chạy trên máy khác. Trên thực tế, phần mềm viết cho máy nhỏ có thể chạy trên máy lớn hơn mà không gặp trục trặc gì, nhưng ngược lại thì chưa chắc đã được, vì bộ nhớ của máy nhỏ có thể không chứa được phần mềm viết cho máy lớn.

Ý tưởng về một họ máy như vậy ngay lập tức được các hãng sản xuất máy tính khác nắm bắt, trong một vài năm hầu hết các nhà sản xuất đã cho ra đời một họ các máy tính giống nhau của mình, chúng có giá và hiệu suất bao trùm một miền rộng.

- **Đặc tính đa chương trình**

Một đổi mới quan trọng khác của System/360 là đặc tính đa chương trình (multiprogramming), nghĩa là có thể có vài chương trình cùng nằm trong bộ nhớ tại một thời điểm và một trong số đó được cho chạy trong khi các chương trình khác chờ hoàn thành các thao tác vào/ra (input/output) của mình.

System/360 là máy tính đầu tiên có thể mô phỏng các máy tính khác. Các mô-đen System/360 loại nhỏ có thể mô phỏng máy 1401 và các mô-đen lớn có thể mô phỏng 7094, nhờ vậy mà các khách hàng có thể tiếp tục chạy các chương trình dạng nhị phân cũ của mình mà không cần phải sửa đổi chúng sang cho System/360. Một số mô-đen System/360 chạy các chương trình cho 1401 nhanh hơn trên chính máy 1401.

- **Không gian địa chỉ rất lớn**

Một đặc điểm quan trọng nữa của System/360 là nó có một không gian địa chỉ rất lớn, tới 2^{24} byte (=16 megabyte). Vào thời đó giá 1 byte là vài đô-la thì 16MB là một không gian nhớ có thể coi là vô hạn. Sau thế hệ System/360 là các thế hệ 370, 4300, 3080 và 3090, tất cả chúng đều có chung một kiến trúc.

Đến khoảng giữa những năm 80, không gian địa chỉ 16MB không còn là lớn nữa, IBM đã phải sử dụng không gian địa chỉ 32 bit trong các máy mới, chấp nhận sự không tương thích về địa chỉ với các máy ra đời trước đây.

PDP-11 của hãng DEC - minicomputer.

Thế giới của các máy tính mini thế hệ thứ ba có một bước tiến quan trọng khi hãng DEC đưa ra máy PDP-11, là máy 16 bit, thế hệ tiếp theo của PDP-8. Máy PDP-11 có nhiều nét giống với hàng em út của dòng IBM 360, cũng như trước đây PDP-1 giống 7094. Cả IBM 360 và PDP-11 đều có các thanh ghi hướng word

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

(word-oriented registers) và bộ nhớ hướng byte (byte-oriented memory) và chúng đều có tỉ số giá/hiệu suất trải trên một miền khá rộng.

PDP-11 đã thành công rực rỡ, đặc biệt là trong các trường đại học và hãng DEC tiếp tục đứng đầu trong các nhà sản xuất máy tính mini.

1.2.5 Thế hệ thứ tư - Máy tính dùng mạch VLSI (1980 -)Vào khoảng năm 1980, công nghệ vi điện tử đã có thể chế tạo các mạch tổ hợp ở mức độ rất cao VLSI (Very Large Scale Integrator), trong một chip có thể có hàng nghìn, hàng trăm nghìn, thậm chí hàng triệu transistor, nhờ đó máy tính ngày càng nhỏ hơn, chạy nhanh hơn và rẻ hơn. Khoảng năm 1980, giá máy tính xuống thấp tới mức "vừa túi" của một cá nhân và Kỷ nguyên máy tính cá nhân đã bắt đầu. Máy tính cá nhân được sử dụng theo cách hoàn toàn khác với các máy tính lớn và các máy tính mini. Chúng được sử dụng để soạn thảo văn bản, làm trang tính và rất nhiều ứng dụng có tính tương tác cao mà các máy tính lớn không thể nào làm tốt được.

Dựa trên kích thước vật lý, hiệu suất và lĩnh vực ứng dụng, hiện nay người ta thường chia máy tính số Thế hệ thứ tư một cách tương đối làm 5 loại chính, các loại có thể trùm lên nhau một phần:

- **Microcomputer:** còn được gọi là PC (Personal Computer), là những máy tính có thể để bàn, hoặc mang xách được (portable), PC có một chip vi xử lý và một số thiết bị ngoại vi như màn hình, bàn phím, ổ đĩa, máy in.v.v. thường dùng cho một người. Ban đầu các PC được thiết kế nhằm cho một người sử dụng độc lập, tuy nhiên sau này xu hướng nối các PC vào mạng máy tính ngày càng trở nên phổ biến.

- **Minicomputer**: Là những máy tính cỡ trung bình, kích thước thường lớn hơn Microcomputer. Nó có thể thực hiện được các ứng dụng mà máy tính cỡ lớn (Mainframe) thực hiện. So với các máy Mainframe thì Minicomputer có bộ nhớ nhỏ hơn, sức mạnh tính toán thấp hơn. Minicomputer được xử dụng rộng rãi trong các ứng dụng thời gian thực, thí dụ trong điều khiển không lưu, trong tự động hoá sản xuất. Minicomputer thường được nối với các Mainframe để thực hiện các thao tác hỗ trợ cho Mainframe.

- **Supermini:** Là những máy Minicomputer có tốc độ xử lý nhanh nhất trong họ Mini ở những thời điểm nhất định, thường được xây dựng dựa trên các bộ xử lý 32 bit và có hàng chục megabyte bộ nhớ (thời kỳ đầu). **Supermini** thường được sử dụng trong các hệ thống phân chia thời gian, thí dụ máy quản gia của mạng (network fileserver) ...

- **Mainframe**: Các máy Mainframe thường là hàng con cháu của IBM-360 và CDC 6600 của Cray. Điểm khác nhau căn bản giữa Mainframe và Supermini là ở năng lực vào/ra và ứng dụng mà nó sử dụng. Trong khi một Supermini thường có vài ổ đĩa cứng 1 GB, thì Mainframe có thể có tới hàng trăm. Supermini thường dùng trong các ứng dụng giao tiếp (interactive) còn Mainframe được sử dụng trong chế độ các công việc sắp theo lô lớn (Large-Batch-Jobs) hoặc xử lý các giao dịch (Transaction Processing), thí dụ trong ngân hàng, đặt vé máy bay.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Supercomputer**: đây là những máy tính được thiết kế đặc biệt để đạt tốc độ thực hiện các phép tính dấu phảy động (FLOPS-FLoating point Operations Per Second) cao nhất có thể được. Chúng thường có kiến trúc song song, chỉ hoạt động có hiệu quả cao trong một số lĩnh vực. Các máy tốc độ nhỏ hơn 1 giga flops/sec không thể gọi là Supercomputer.

Trong nhiều năm, cái tên supercomputer và Seymour **Cray** hầu như là đồng nghĩa, bởi vì Cray đã thiết kế ra CDC 6600 và các thế hệ tiếp theo, thí dụ 7600, sau đó ông đã đứng ra thành lập công ty riêng của mình là **Cray Research** để xây dựng các máy Cray-1 và Cray-2. Năm 1989 ông lại tách ra lập một công ty khác để xây dựng máy Cray-3.

Bảng 1-03: Năm kiểu máy tính hiện nay

Kiểu máy	Tốc độ điển hình (MIPS)	Dung lượng bộ nhớ điển hình (MB)	Thí dụ về tên máy	Thí dụ về ứng dụng
PC	1	1	IBM PS/2	Xử lý văn bản
Minicomputer	2	4	PDP-11/84	Điều khiển thời gian thực
Supermini	10	32	SUN-4	Server trên mạng
Mainframe	30	128	IBM 3090/300	Ngân hàng
Supercomputer	125	1024	Cray-2	Dự báo thời tiết

* **Chú ý:** Sự phân loại trên là tương đối, các loại trên có thể trùm phủ lên nhau, ngoài ra luôn thay đổi theo thời gian.

Bảng 1-04 tóm tắt các thế hệ máy tính điện tử và các linh kiện chính được sử dụng ở từng thế hệ.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Bảng 1-04: Các thế hệ máy tính điện tử

Thế hệ	Linh kiện, đặc điểm chính...
Thế hệ thứ I (1945-1955)	
- Linh kiện, cấu trúc	- Đèn điện tử, mạch rời
- Bộ nhớ chính	- Rơ-le điện
- Thời gian thao tác	- Cỡ ms (mili second)
- Thí dụ	- ENIAC (Mỹ), BESM (LX)
Thế hệ thứ II (1955-1965)	
- Linh kiện, cấu trúc	- Linh kiện bán dẫn, mạch in
- Bộ nhớ chính	- Xuyến pherit
- Thời gian thao tác	- µs (micro second)
- Thí dụ	- TX-0, IBM 1070(Mỹ), MINSK (LX)
Thế hệ thứ III (1965-1980)	
- Linh kiện, cấu trúc	- Mạch tích hợp SSI, MSI (IC)
- Bộ nhớ chính	- Bộ nhớ bằng dây từ, xuyến pherit
- Thời gian thao tác	- µs - ns (Micro second - nano second)
- Thí dụ	- IBM 360, PDP-11(Mỹ), EC (LX)
Thế hệ thứ IV(1980-20??)	
- Linh kiện, cấu trúc	- Mạch vi điện tử LSI và VLSI, bộ vi xử lý
- Bộ nhớ chính	- Bộ nhớ màng mỏng, MOS...
- Thời gian thao tác	- µs - ns (Micro second - nano second)
- Thí dụ	- IBM PS/2, PDP 11/84, SUN-4, IBM 3090/300, CRAY-2 ...

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 2 TỔ CHỨC HỆ THỐNG MÁY TÍNH

Máy tính số là hệ thống bao gồm các bộ xử lý, bộ nhớ và các thiết bị vào/ra được kết nối với nhau. Chương này nghiên cứu 3 thành phần nêu trên và sự kết nối giữa chúng, làm cơ sở cho việc khảo sát chi tiết các mức cụ thể ở những chương tiếp theo. Bộ xử lý, bộ nhớ và các thiết bị vào/ra là các khái niệm mấu chốt sẽ được trình bày ở từng mức, chúng ta bắt đầu việc nghiên cứu kiến trúc máy tính bằng cách lần lượt nghiên cứu 3 khái niệm này.

2.1 KIẾN TRÚC CHUNG CỦA MÁY TÍNH ĐIỆN TỬ

Máy tính điện tử từ khi ra đời cho tới nay mặc dù đã trải qua 4 thế hệ, liên tục được cải tiến, phát triển nhưng nhìn chung vẫn bao gồm 5 đơn vị chức năng chính sau:

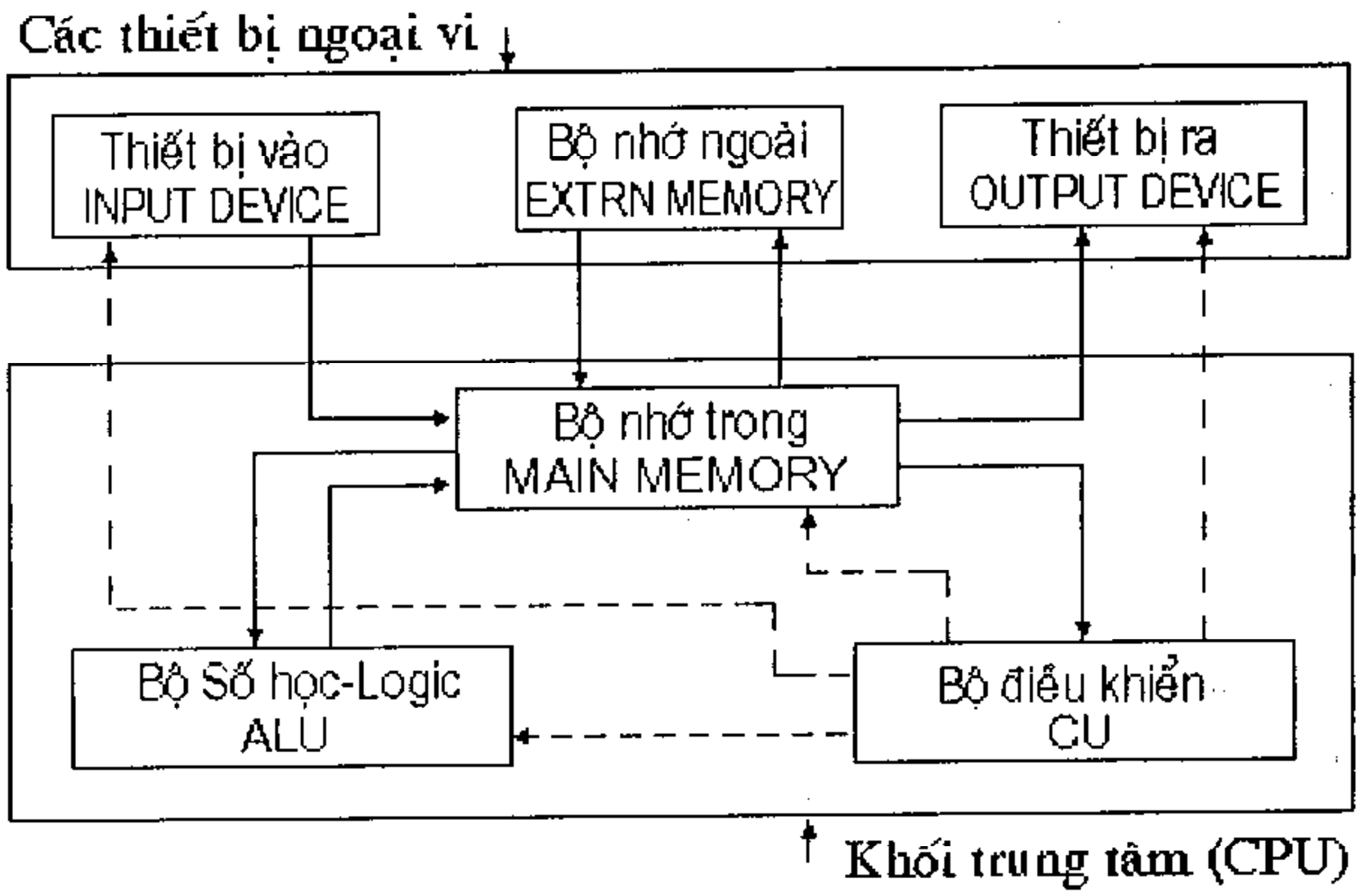
1. Bộ nhớ trung tâm (thuật ngữ tiếng Anh là Central Memory hoặc Main memory). Có nhiệm vụ chứa các chương trình và dữ liệu trước khi chương trình được thi hành.
2. Bộ điều khiển (thuật ngữ tiếng Anh là Control Unit, thường được viết tắt là CU). Có nhiệm vụ điều khiển sự hoạt động của tất các các thành phần của hệ thống máy tính theo chương trình mà nó được giao thi hành.
3. Bộ số học và logic (thuật ngữ tiếng Anh là Arithmetic Logical Unit, thường được viết tắt là ALU). Có nhiệm vụ thực hiện các thao tác tính toán theo sự điều khiển của CU.
4. Thiết bị vào (thuật ngữ tiếng Anh là Input Device). Có nhiệm vụ nhận các thông tin từ thế giới bên ngoài, biến đổi sang dạng số một cách thích hợp rồi đưa vào bộ nhớ trong.
5. Thiết bị ra (thuật ngữ tiếng Anh là Output Device) . Có nhiệm vụ đưa thông tin số từ bộ nhớ trong ra ngoài dưới những dạng mà con người yêu cầu.

Các đơn vị chức năng và mối quan hệ giữa chúng có thể được minh hoạ bằng hình vẽ 2-01 dưới đây. Có thể mô tả sự hoạt động của máy tính một cách khái quát như sau:

- Trước hết các chương trình và số liệu ban đầu được đưa vào bộ nhớ trong, đó thường là bộ nhớ bán dẫn RAM.
- Khi bắt đầu thi hành chương trình, lệnh đầu tiên trong tập lệnh đã được tích luỹ ở bộ nhớ trong được đưa vào thiết bị điều khiển - CU.
- CU tiến hành giải mã lệnh, nếu việc giải mã cho thấy lệnh cần một hay một số toán hạng thì nó sẽ xác định xem toán hạng đó nằm ở đâu trong bộ nhớ, việc này thường được gọi là tính địa chỉ các toán hạng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Sau khi tính địa chỉ toán hạng, CU sẽ phát ra các tín hiệu điều khiển tới các thành phần cần thiết của hệ thống để lấy các toán hạng về, đặt vào các thanh ghi bên trong đơn vị ALU.



Hình 2-01 Các đơn vị chức năng cơ bản của máy tính điện tử. (Các đường vẽ bằng nét đứt chỉ mối quan hệ điều khiển. Các đường vẽ bằng nét liền là các con đường truyền dữ liệu.)

- CU phát tín hiệu điền khiển tới ALU để ALU thực hiện phép toán trên các toán hạng đã lấy về. Kết quả phép toán có thể được để trong ALU để nó tham gia vào các phép toán tiếp theo hoặc được đưa ra bộ nhớ trong. Điều này tuỳ thuộc vào mã lệnh mà CU đã nhận vào và giải mã.
- Nếu CU giải mã và thấy rằng, mã lệnh cho biết cần tiến hành rẽ nhánh chương trình, nó sẽ tính địa chỉ bộ nhớ của lệnh kế tiếp cần thực hiện và phát ra các tín hiệu điều khiển để lấy lệnh kế tiếp về, sau đó mọi việc lại diễn ra tương tự như trên.
- Nếu sau khi giải mã, CU thấy rằng không cần rẽ nhánh chương trình, nó sẽ phát ra các tín hiệu điều khiển để lấy về lệnh đứng ngay sau nó trong bộ nhớ, sau đó mọi việc lại diễn ra tương tự như trên.

Quan hệ giữa bộ nhớ trong và ALU là quan hệ 2 hướng, tức là số liệu sau khi đã được đưa vào xử lý trong ALU theo đúng ý muốn của người lập chương trình sẽ lại được đưa ra bộ nhớ trong để sau đó khi có lệnh từ CU số liệu này có thể được đưa ra thiết bị ra.

Qua mối quan hệ nêu trên ta thấy rằng CU, ALU và Bộ nhớ trong tham gia trực tiếp vào quá trình xử lý số liệu, chính vì vậy mà chúng còn được gọi là ***Đơn vị xử lý trung tâm (CPU - Central Processing Unit).***

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Tập hợp các thiết bị vào và thiết bị ra thường được gọi bằng một cái tên chung là ***thiết bị ngoại vi*** (thuật ngữ tiếng Anh là ***Peripherals,*** hoặc ***I/O Devices***). Có những thiết bị trong quá trình hoạt động của máy tính khi thì đóng vai trò của thiết bị vào, khi thì đóng vai trò của thiết bị ra. Có một số thiết bị vào và một số thiết bị ra là thiết bị nhớ ngoài, chúng tạo nên bộ nhớ ngoài như trên hình vẽ. Thông tin trao đổi giữa bộ nhớ ngoài và hệ thống máy tính luôn thông qua bộ nhớ trong, dưới sự điều khiển của CU. Bộ nhớ ngoài của các máy tính ngày nay được sử dụng thường xuyên nên tốc độ hoạt động của chúng ảnh hưởng nhiều tới tốc độ chung của hệ thống máy tính. Các thiết bị ngoại vi khác cũng có vai trò quan trọng, đó là các máy móc, thiết bị để liên kết máy tính điện tử với thế giới bên ngoài.

2.2 BỘ XỬ LÝ - PROCESSORS

Khi phân tích các đơn vị chức năng của máy tính điện tử được minh hoạ trên Hình 2-01 chúng ta đã gọi ba đơn vị chức năng chính của máy tính điện tử: CU, ALU và bộ nhớ chính là Đơn vị xử lý trung tâm - CPU. Trong các máy tính điện tử thế hệ thứ nhất và thứ hai các đơn vị chức năng này thường được xây dựng thành các mô-đun độc lập, tách biệt nhau rất rõ ràng, người ta có thể mở máy ra và chỉ rõ đâu là các mạch điện thực hiện chức năng ALU, đâu là các mạch điện thực hiện chức năng CU...

Đến thế hệ máy tính thứ ba và thứ tư người ta đã sử dụng các mạch tích hợp (IC), các đơn vị chức năng CU, ALU có thể được xây dựng từ một hoặc một số mạch tích hợp. Các đơn vị CU, ALU và một bộ nhớ đặc biệt có tốc độ cao, nhỏ, được gọi là các thanh ghi (register) thường được bố trí gần nhau tạo thành bộ xử lý (processor) của máy tính, đó chính là là Đơn vị xử lý trung tâm - CPU.

Các máy tính điện tử thế hệ thứ tư đều sử dụng các mạch tích hợp cỡ rất lớn (VLSI), người ta đã tích hợp được vào trong một con chip hàng trăm nghìn, thậm chí hàng triệu transistor và một số linh kiện khác, các linh kiện này đủ để xây dựng nên các đơn vị chức năng CU, ALU và một phần bộ nhớ chính của máy tính. Các chip có thể thực hiện được chức năng như một CPU đầy đủ như vậy được gọi là bộ vi xử lý - microprocessor.

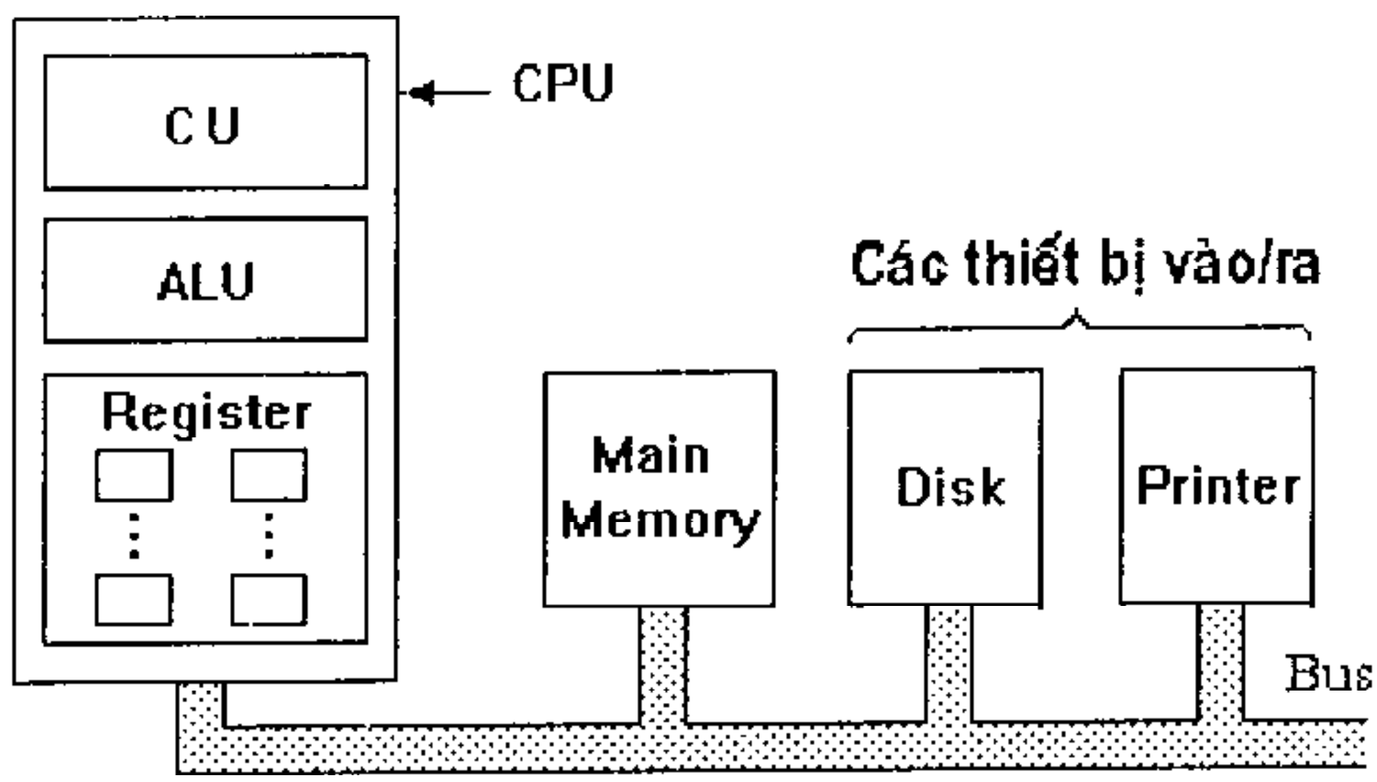
Tổ chức của một máy tính đơn giản, sử dụng một bus được trình bày trên hình 2-02. CPU là bộ não của máy tính, nó bao gồm CU, ALU và các thanh ghi (Register). Chức năng của nó là thi hành các chương trình được chứa trong bộ nhớ của máy tính (main memory) bằng cách lấy các chỉ thị ở nó, kiểm tra và sau đó lần lượt thi hành các chỉ thị.

Đơn vị điều khiển - CU có nhiệm vụ lấy các chỉ thị từ bộ nhớ chính, giải mã và điều khiển ALU cũng như tất cả các thành phần khác thi hành các chỉ thị.

Đơn vị số học và logic - ALU thực hiện các thao tác đơn giản, chẳng hạn cộng, AND logic ... để hoàn thành các chỉ thị.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CPU cũng còn bao gồm một bộ nhớ nhỏ, tốc độ cao để chứa các kết quả tạm thời và những thông tin điều khiển nhất định. Bộ nhớ này gồm một số **thanh ghi - register**, mỗi thanh ghi có một chức năng nhất định. Thanh ghi quan trọng nhất được gọi là **Con đếm chương trình (Program counter - PC)**, nó trỏ đến chỉ thị tiếp theo sẽ được thi hành. Tên là "con đếm chương trình" nhưng thực tế nó không đếm gì cả, tuy vậy thuật ngữ này được sử dụng rất rộng rãi. Một thanh ghi quan trọng khác là **thanh ghi chỉ thị (instruction register - IR)**, nó chứa chỉ thị hiện đang được thi hành. Hầu hết các máy tính đều còn có một số thanh ghi khác nữa, trong số đó một số thanh ghi người lập trình ở mức 2 và 3 có thể sử dụng được để chứa các kết quả trung gian.



Hình 2-02 Tổ chức của một máy tính đơn giản có một CPU và hai thiết bị Vào/Ra

2.2.1 Sự thi hành các chỉ thị

CPU thi hành mỗi chỉ thị theo một dãy các bước nhỏ:

1. Lấy (Fetch) chỉ thị tiếp theo từ trong bộ nhớ đặt vào thanh ghi chỉ thị (IR).
2. Thay đổi con đếm chương trình (PC) để trỏ tới chỉ thị tiếp theo.
3. Xác định kiểu của chỉ thị vừa lấy về.
4. Nếu chỉ thị sử dụng dữ liệu trong bộ nhớ thì xác định vị trí của dữ liệu.
5. Lấy dữ liệu về nếu có, đặt vào các thanh ghi bên trong CPU.
6. Thi hành chỉ thị.
7. Chứa kết quả vào nơi thích hợp.
8. Trở lại bước 1 để bắt đầu thi hành chỉ thị tiếp theo

Dãy các bước này thường được gọi là chu kỳ lấy lệnh - giải mã lệnh - thi hành lệnh, đó chính là trung tâm của sự hoạt động của mọi máy tính.

Thực tế là người ta có thể viết một chương trình để bắt chước chức năng của một CPU, chương trình này không nhất thiết phải được thi hành bằng một CPU phần cứng, gồm toàn các mạch điện tử. Thay vào đó chương trình có thể được thi hành

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

bằng cách bắt một chương trình khác lấy, kiểm tra và thi hành các chỉ thị của nó. Chương trình như vậy có tên gọi là trình thông dịch - interpreter.

Sự tương đương giữa bộ xử lý bằng phần cứng và trình thông dịch có một ứng dụng quan trọng đối với việc tổ chức máy tính. Sau khi đã định rõ ngôn ngữ máy chẳng hạn là L cho một máy tính mới, nhóm thiết kế có thể quyết định hoặc là sẽ xây dựng một bộ xử lý bằng phần cứng để thi hành trực tiếp chương trình bằng ngôn ngữ L hoặc là sẽ viết một trình thông dịch.

Nếu họ chọn việc viết một trình thông dịch, họ cũng bắt buộc phải trình ra một cái máy nào đó để chạy trình thông dịch. Vì trình thông dịch sẽ chia sự thực hiện chỉ thị của cái máy đích của nó thành các bước nhỏ, nên cái máy dùng để chạy trình thông dịch có thể đơn giản hơn nhiều so với bộ xử lý phần cứng mà nó thay thế.

Do mục đích kinh tế cũng như một số lý do khác, các chương trình ở mức máy thông thường của hầu hết các máy tính hiện đại đều được thi hành bởi một trình thông dịch, trình này chạy trên một máy mức 1 hoàn toàn khác và đơn giản hơn nhiều. Mức này đã được chúng ta gọi là mức vi chương trình.

Tập chỉ thị

Các chỉ thị mà người lập trình ở một mức có thể sử dụng được gọi là tập chỉ thị của mức đó. Số chỉ thị trong một tập chỉ thị ở các máy khác nhau hoặc ở các mức khác nhau trong cùng một máy tính là khác nhau. Đối với các mức máy thông thường, kích thước của các tập chỉ thị thường nằm trong khoảng từ 20 đến 300.

Một tập chỉ thị lớn không hẳn là tốt hơn một tập chỉ thị nhỏ, đôi khi trong thực tế lại xảy ra điều ngược lại. Trong một tập chỉ thị lớn thường có nhiều chỉ thị không được sử dụng một cách thường xuyên trong các chương trình. Các trình biên dịch cho các ngôn ngữ lập trình bậc cao như Ada, Modula 2, Pascal nói chung sẽ làm việc tốt hơn trên các máy có tập chỉ thị nhỏ, được lựa chọn tốt chứ không phải là trên các máy có tập chỉ thị lớn, cồng kềnh. Các máy có tập chỉ thị rất nhỏ được gọi là các máy RISC, chúng không sử dụng vi chương trình và chạy cực kỳ nhanh.

Tóm lại tập chỉ thị và tổ chức của mức vi chương trình thực tế là tập chỉ thị và tổ chức của phần cứng - của CPU. Tập chỉ thị và tổ chức của mức máy thông thường thì trái lại được xác định bằng vi chương trình, chứ không phải bằng phần cứng.

2.2.2 Tổ chức CPU

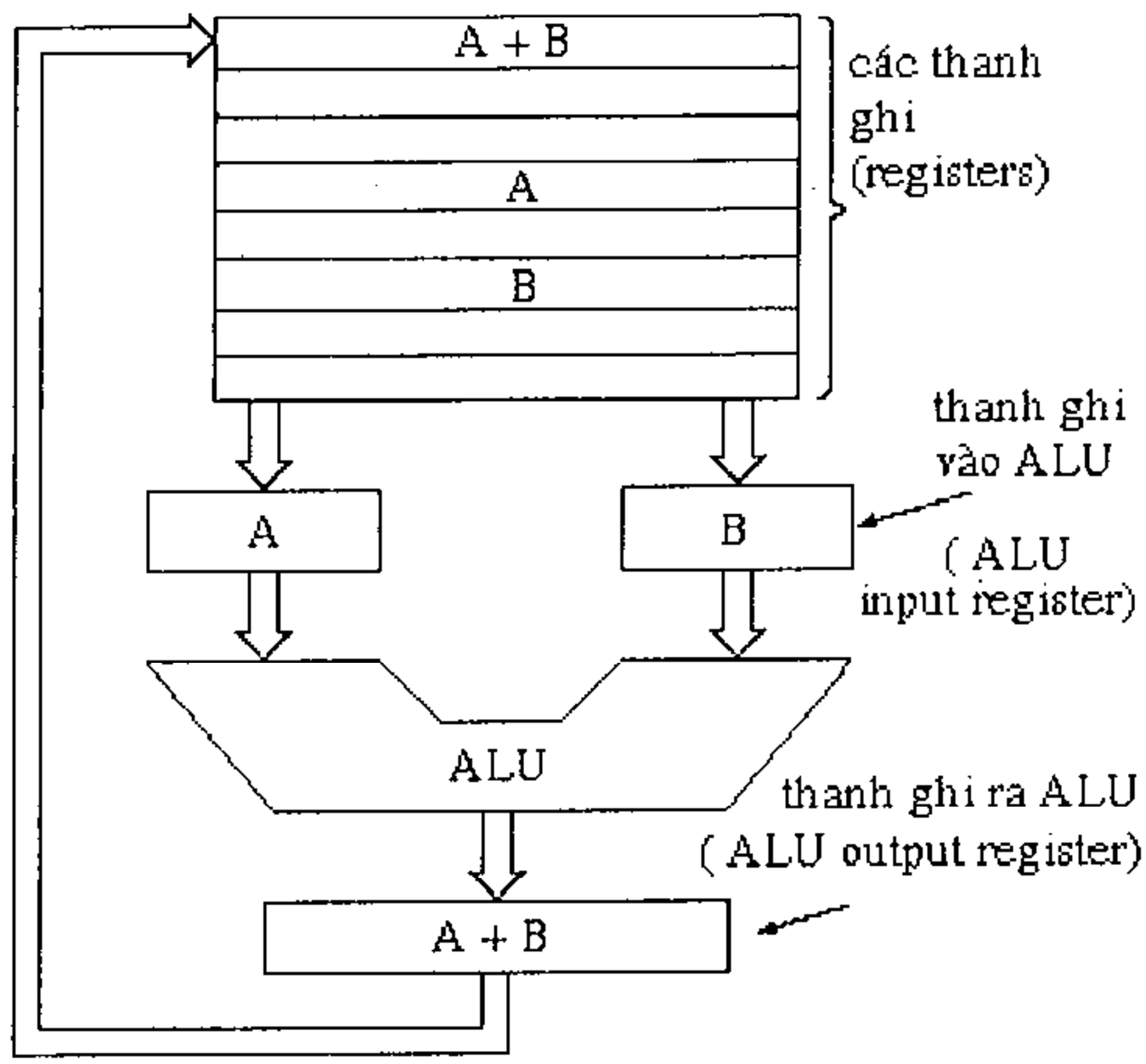
Tổ chức bên trong của một phần của một CPU theo kiểu von Neumann cổ điển được trình bày chi tiết trên hình 2-03. Phần này được gọi là **đường dữ liệu** (data path), nó thường bao gồm từ 1 đến 16 thanh ghi và ALU, các thanh ghi này thường được gọi là các thanh ghi đệm vì chúng thường chứa tạm thời các toán hạng.

Các toán hạng cho phép toán (với hình vẽ minh hoạ 2-03 là phép cộng) mà ALU cần thực hiện đang được chứa trong các thanh ghi đệm được nạp vào 2 thanh ghi vào của ALU là A và B. Các thanh ghi này giữ các toán hạng cho ALU trong khi ALU thực hiện tính toán. Kết quả sinh ra được đưa vào thanh ghi ra (output register).

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Thanh ghi ra có thể lại chuyển nội dung vào một trong các thanh ghi đệm rồi từ đó chuyển ra bộ nhớ nếu cần. (Đơn vị CU sẽ điều khiển các thao tác trên, nhưng không vẽ trên hình, việc điều khiển đường dữ liệu sẽ được nghiên cứu ở chương 4.)

Toán hạng cho các phép toán có thể nằm trong tập các thanh ghi đệm (register), hoặc nằm trong bộ nhớ (memory). Dựa vào vị trí của các toán hạng người ta có thể chia các chỉ thị thành 3 nhóm chính: thanh ghi - bộ nhớ (register-memory), thanh ghi - thanh ghi (register-register) và bộ nhớ - bộ nhớ (memory-memory).



Hình 2-03 Đường dữ liệu của một CPU đang thực hiện phép tính cộng 2 toán hạng A và B

Các lệnh thuộc nhóm register-memory đòi hỏi phải lấy một toán hạng trong bộ nhớ về, đặt vào một thanh ghi đệm nhất định, còn toán hạng thứ hai đang nằm trong một thanh ghi nào đó, sau đó chúng sẽ được ALU sử dụng.

Các lệnh thuộc nhóm register-register sẽ nhận 2 toán hạng từ các thanh ghi đệm, đưa chúng vào các thanh ghi vào của ALU, thực hiện các thao tác trên các toán hạng này sau đó đưa kết quả trở lại các thanh ghi đệm.

Các lệnh thuộc nhóm memory-memory sẽ nhận các toán hạng từ bộ nhớ vào trong các thanh ghi vào của ALU, thực hiện các thao tác tính toán trên các toán hạng này rồi gửi kết quả trở lại bộ nhớ.

Đường dữ liệu là trái tim của hầu hết các CPU. Trong một mức độ nào đó nó xác định việc máy tính có thể làm được những công việc gì và hiệu quả ra sao.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

2.2.3 Sự thực hiện song song các lệnh

Ngay từ khi thiết kế những máy tính điện tử đầu tiên, người ta đã tìm mọi cách để làm cho nó chạy càng nhanh càng tốt. Việc ***nâng cao tốc độ hoạt động của máy tính có thể thực hiện bằng cách nâng cao tốc độ hoạt động của phần cứng***, tuy nhiên luôn có các giới hạn vật lý ngăn cản. Thí dụ, vật lý học đã chứng minh rằng không thể có một dạng chuyển động nào vượt được vận tốc của ánh sáng trong chân không, xấp xỉ bằng 300.000km/s hay 30cm/ns. Trong dây đồng, vận tốc sóng điện từ khoảng 20cm/ns. Dựa trên giới hạn này ta thấy rằng để xây dựng máy tính với tốc độ 1 tỉ phép tính trong một giây (mỗi phép tính thực hiện trong 1ns) thì toàn bộ khoảng cách mà tín hiệu điện phải đi qua trong CPU, tới bộ nhớ và ngược lại không được vượt quá 20cm.. Chính vì vậy các máy tính cực nhanh phải có kích thước cực nhỏ. Các máy tính tốc độ càng cao thì càng sinh nhiều nhiệt, do đó việc tản nhiệt phải được thực hiện thật tốt. Các siêu máy tính thường được nhúng trong các chất lỏng đặc biệt, dẫn nhiệt tốt để làm nguội.

Như vậy để chế tạo ra các máy tính tốc độ ngày càng cao hơn thì phải giải quyết nhiều khó khăn hơn và chi phí cũng sẽ cao hơn. Tuy nhiên cũng có cách tiếp cận khác, đáng lẽ phải sử dụng một CPU tốc độ cao, chúng ta có thể xây dựng một máy tính với nhiều ALU tốc độ thấp (rẻ hơn) hoặc thậm chí với nhiều CPU để đạt được cùng một công suất tính toán nhưng với giá thành thấp hơn.

Đã có nhiều công trình nghiên cứu về các máy tính song song như vậy, dưới đây sẽ giới thiệu tóm tắt một số kỹ thuật. Các máy tính song song có thể được chia thành 3 nhóm, căn cứ vào việc chúng có bao nhiêu dòng chảy lệnh và dữ liệu:

1. SISD (Single Instruction stream, Single Data stream): Máy tính có một dòng chảy của lệnh và một dòng chảy của dữ liệu.
2. SIMD (Single Instruction stream, Multiple Data stream): Máy tính có một dòng chảy của lệnh và nhiều dòng chảy của dữ liệu.
3. MIMD (Multiple Instruction stream, Multiple Data stream): Máy tính có nhiều dòng chảy của lệnh và nhiều dòng chảy của dữ liệu.

Ba loại máy tính này sẽ được giới thiệu dưới đây.

2.2.3.1 Máy tính loại SISD

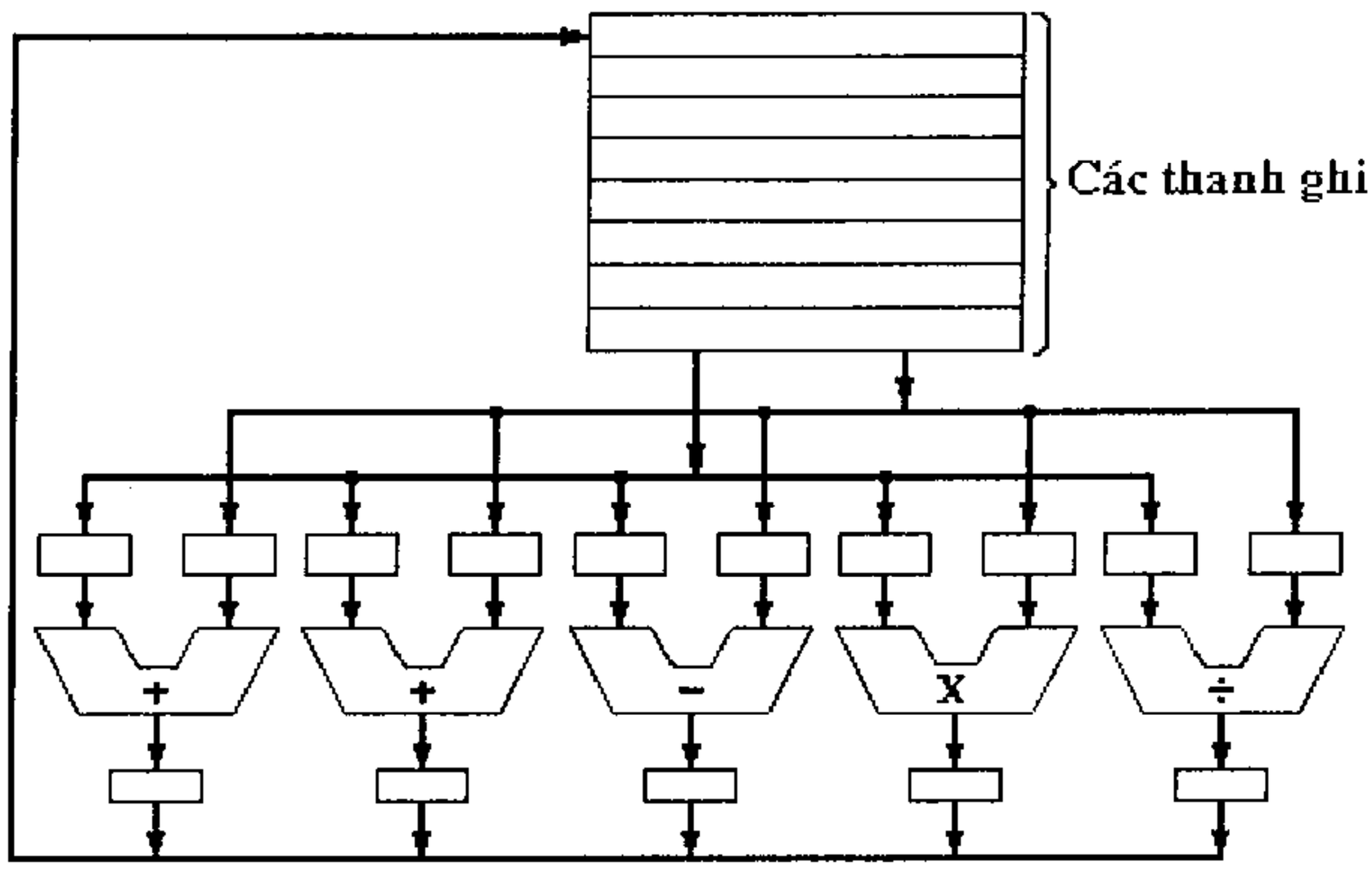
Các máy tính dựa theo nguyên lý Von Neumann truyền thống thuộc loại SISD. Nó có một dòng chảy lệnh (nghĩa là một chương trình) được 1 CPU thi hành và một bộ nhớ chứa dữ liệu của nó. Lệnh đầu tiên được nhận từ bộ nhớ về rồi được thực hiện, sau đó lệnh thứ hai sẽ được nhận về rồi thực hiện.v.v.

Cơ chế thực hiện song song các lệnh trong máy tính loại SISD

Ngay trong mô hình tuần tự này cũng có thể thực hiện cơ chế song song ở một chừng mực nhất định, bằng cách trong khi đang thực hiện một lệnh, nhận về lệnh tiếp theo và bắt đầu thực hiện nó.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Máy CDC 6600 và một số thế hệ sau nó có nhiều ALU như được minh hoạ trên Hình 2-04 mỗi ALU này có thể thực hiện một thao tác đơn với tốc độ cao. Trên hình vẽ ta thấy có 5 ALU, trong đó 2 ALU để thực hiện phép toán cộng, 1 ALU thực hiện phép toán trừ, 1 ALU thực hiện phép toán nhân và 1 ALU thực hiện phép toán chia. Ý tưởng của thiết kế này là: đơn vị điều khiển (CU) nhận về một lệnh rồi chuyển cho một trong các ALU thực hiện, đồng thời CU nhận về lệnh tiếp theo rồi chuyển cho một trong các ALU khác đang rỗi thực hiện. Quá trình này chỉ dừng lại khi mọi ALU đều đã bận. Chiến lược làm việc này có hiệu quả cao khi thời gian thực hiện một lệnh lớn hơn nhiều thời gian lấy lệnh về (fetch). Thí dụ trong các tính toán số dấu phảy động.



Hình 2-04 Một CPU với 5 đơn vị chức năng có thể chạy song song

Một biến thể của cơ chế thực hiện song song các lệnh

Một biến thể của ý tưởng trên là chia việc thực hiện mỗi lệnh thành nhiều giai đoạn, CPU cũng được thiết kế bao gồm một số đơn vị chức năng, mỗi đơn vị chức năng sẽ thực hiện một giai đoạn của quá trình thực hiện một lệnh. Như vậy việc thực hiện lệnh giống như hoạt động của một dây chuyền sản xuất có nhiều công đoạn.

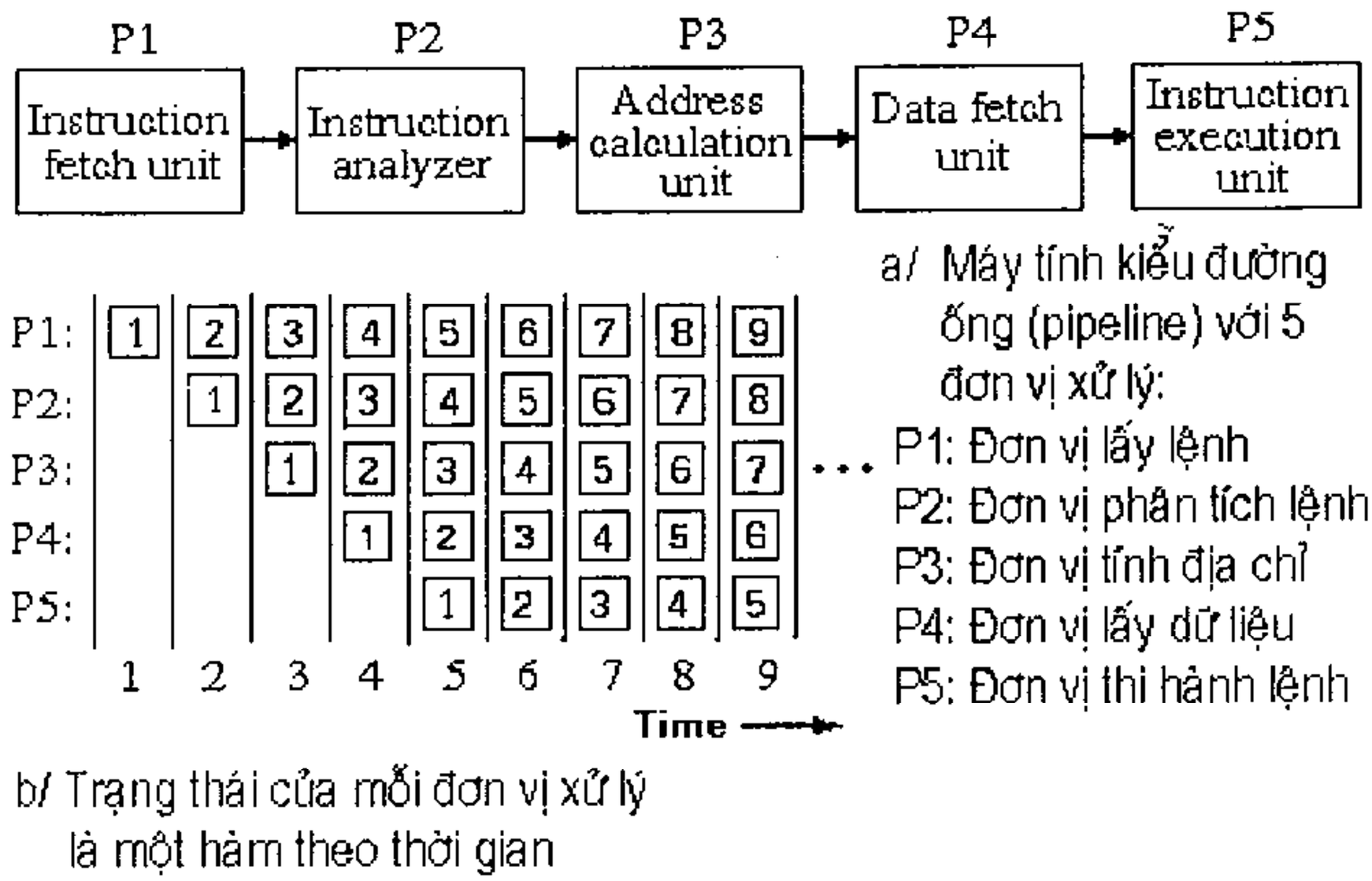
Hình 2-05 là một CPU bao gồm 5 đơn vị xử lý, ký hiệu từ P1 đến P5. Đầu tiên P1 lấy về lệnh thứ nhất từ bộ nhớ, việc này cần một khoảng thời gian nào đó. Trong khoảng thời gian thứ hai lệnh đầu tiên được chuyển cho P2 để phân tích, trong lúc này P1 lấy về lệnh kế tiếp. Trong mỗi khoảng thời gian tiếp theo P1 sẽ lấy về một lệnh mới, còn các lệnh đã nhận sẽ được chuyển cho các đơn vị xử lý kế tiếp.

Máy tính được tổ chức theo kiểu này được gọi là ***máy tính kiểu đường ống (pipeline machine)***. Nếu mỗi khoảng thời gian ta nói ở trên dài n na-nô giây thì mỗi lệnh được thực hiện trong 5n na-nô giây, tuy nhiên trong mỗi khoảng thời gian

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

n na-nô giây có một lệnh được hoàn thành và đi ra khỏi P5, như vậy tốc độ đã được tăng lên 5 lần.

Mặc dù máy tính pipeline có sử dụng cơ chế song song bên trong nhưng nó vẫn thuộc loại SISD, vì chỉ có một chương trình được thi hành và một tập dữ liệu mà chương trình đó sử dụng.



Hình 2-05 Một máy tính kiểu đường ống

2.2.3.2 Máy tính loại SIMD

Máy tính vector

Các máy tính loại này thao tác song song trên nhiều tập dữ liệu, chúng thường được sử dụng để giải các bài toán dự báo thời tiết, chẳng hạn tính toán nhiệt độ trung bình hàng ngày trong suốt 24 giờ của nhiều địa điểm; Đối với mỗi địa điểm công việc tính toán (chương trình) là như nhau, nhưng tập dữ liệu là khác nhau. Kiến trúc thích hợp cho nhiệm vụ này là máy tính vector, như trên Hình 2-06. Đường dữ liệu ở đây tương tự đường dữ liệu trên Hình 2-03, chỉ khác ở chỗ không phải chỉ có một biến đơn cho mỗi lối vào ALU mà chúng ta có một vector với n lối vào. Tương tự như vậy ALU thực sự là một ALU vector, nó có khả năng thực hiện một phép toán như là cộng vector (Hình 2-06b) đối với 2 vector lối vào và đưa ra vector kết quả. Một số siêu máy tính có kiến trúc tương tự như vậy.

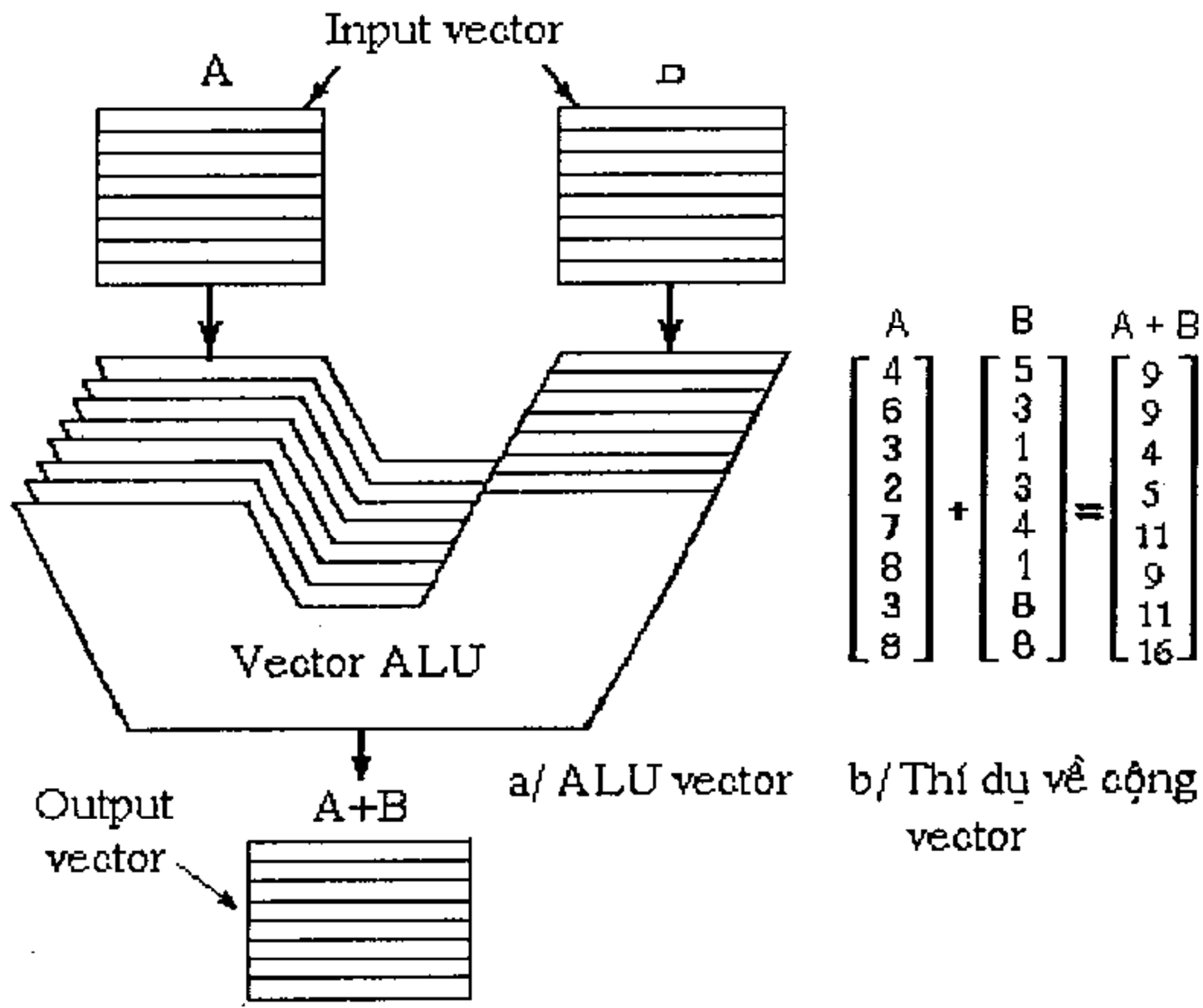
Bộ xử lý mảng (Array processor)

Một cách tiếp cận khác tới SIMD là sử dụng bộ xử lý mảng (array processor). Trường đại học Illinois đi tiên phong trong hướng tiếp cận này, họ đã thiết kế ra máy ILLIAC IV, nguyên lý của nó được minh hoạ trên Hình 2-07.

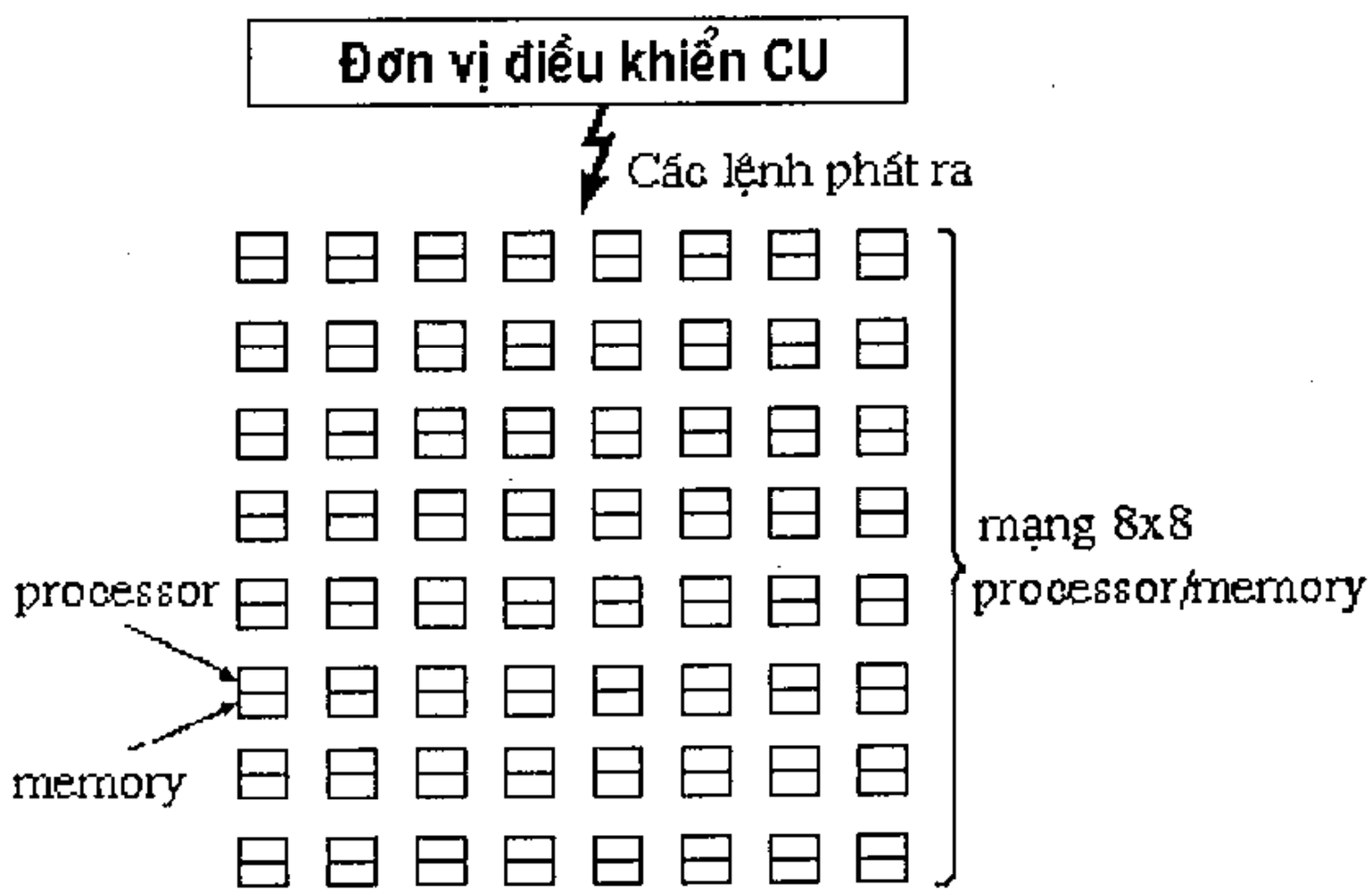
Kiến trúc này bao gồm một mạng lưới hình vuông các phần tử processor/memory (mỗi phần tử như vậy gồm một bộ xử lý- processor và một bộ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

nhớ - memory dành riêng cho bộ xử lý đó làm việc). Đơn vị CU phát ra các lệnh, các lệnh được thực hiện từng bước bởi tất cả các bộ xử lý, mỗi bộ xử lý sử dụng dữ liệu riêng lấy từ bộ nhớ riêng của nó. Bộ xử lý mảng thích hợp với các tính toán ma trận.



Hình 2-06 Máy tính vector



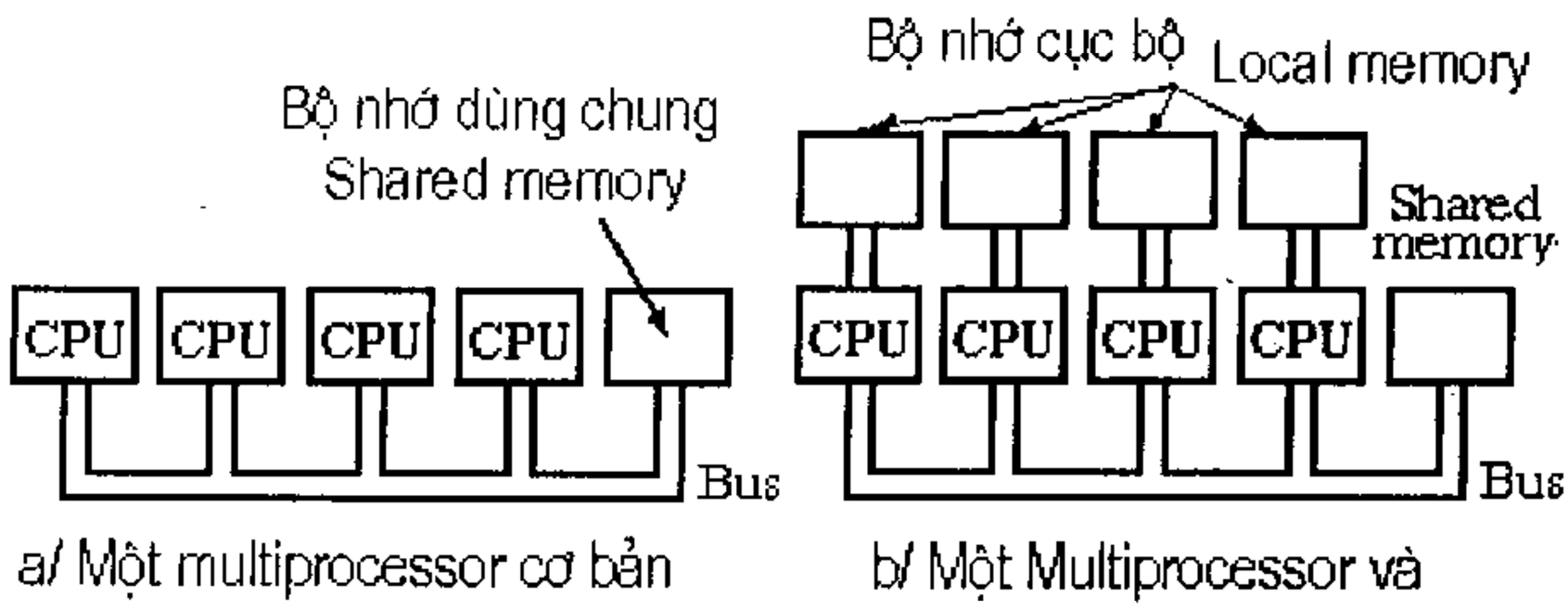
Hình 2-07 Bộ xử lý mảng của máy tính loại ILLIAC IV

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

2.2.3.3 Máy tính loại MIMD

Máy tính loại này có nhiều CPU, mỗi CPU thực hiện một chương trình khác nhau, các CPU có thể cùng chia sẻ một vùng nhớ chung (được ghi là shared memory trên hình 2-08). Thí dụ trong một hệ thống máy tính phục vụ việc đặt chỗ trong ngành hàng không, việc đặt chỗ có thể xảy ra thường xuyên, liên tục và đồng thời, mỗi cuộc đặt chỗ cần một chương trình để thực hiện, các chương trình này thường không bắt đầu song song với nhau từng lệnh một. Vì vậy chúng ta có nhiều dòng chảy (stream) lệnh và nhiều dòng chảy dữ liệu.

Hình 2-08a là một máy tính loại MIMD có nhiều CPU, sử dụng một bộ nhớ dùng chung. Mỗi bộ xử lý truy cập tới bộ nhớ này thông qua một đường truyền tín hiệu chung gọi là bus.



Hình 2-08 Máy tính loại MIMD

Nếu nhiều bộ xử lý có tốc độ cao cùng truy cập bộ nhớ dùng chung thông qua cùng một bus thì có thể xảy ra đụng độ. Người thiết kế máy tính loại MIMD phải giải quyết vấn đề này sao cho giảm được khả năng xảy đụng độ đồng thời vẫn đạt được hiệu quả cao. Có một số cách giải quyết được trình bày sau đây.

Một trong các phương án thiết kế như vậy được minh hoạ trên Hình 2-08b, trong đó mỗi bộ xử lý được trang bị một bộ nhớ cục bộ riêng, các bộ xử lý khác không truy cập được. Bộ nhớ cục bộ được sử dụng để chứa các đoạn mã chương trình hoặc các phần tử dữ liệu không được sử dụng chung. Các bộ xử lý khi truy cập bộ nhớ cục bộ riêng của mình không sử dụng bus chính dùng chung, nhờ vậy làm giảm rất nhiều việc truyền thông trên bus chính.

Có những máy tính loại MIMD khác sử dụng nhiều bus để giảm khả năng xảy ra đụng độ và cũng có những máy MIMD sử dụng bộ nhớ cache. Đó là một kỹ thuật trong đó các lệnh hay dữ liệu trong bộ nhớ nếu được bộ xử lý sử dụng thường xuyên sẽ được cất giữ vào một bộ nhớ nhỏ có tốc độ cao nằm ngay bên trong bộ xử lý. (Bộ nhớ cache sẽ được nghiên cứu ở chương 4)

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

2.3 BỘ NHỚ

Bộ nhớ (memory) là một thành phần quan trọng của máy tính, được sử dụng để lưu trữ các chương trình và dữ liệu. Trong tiếng Anh đôi khi người ta còn dùng thuật ngữ store hay storage.

2.3.1 Bit

Theo nguyên lý số trong máy tính người ta sử dụng các đại lượng vật lý biến thiên rời rạc để biểu diễn dữ liệu. Nếu sử dụng một đại lượng vật lý có rất nhiều giá trị khác nhau thì việc phân biệt các giá trị lân cận sẽ khó khăn và do đó độ tin cậy sẽ giảm đi. Hệ thống số nhị phân chỉ có hai giá trị là 0 và 1, do đó việc sử dụng nó là phương pháp tin cậy nhất để mã hoá (encode) thông tin số. Đơn vị cơ sở của bộ nhớ là bit, một bit có thể chứa số 0 hoặc 1.

2.3.2 Địa chỉ ô nhớ, byte, word

Bộ nhớ bao gồm các ô nhớ (cells; locations) có cùng kích thước (tức là cùng số bit), mỗi ô nhớ được gán một địa chỉ duy nhất. Nếu bộ nhớ có n ô nhớ thì địa chỉ của chúng sẽ nằm trong miền từ 0 tới n-1. Các máy tính khác nhau có thể sử dụng ô nhớ có kích thước khác nhau, thí dụ nêu trong bảng 2-01.

Bảng 2-01 Kích thước ô nhớ của một số máy tính

Tên máy	Kích thước ô nhớ (bit)
Burroughs B1700	1
IBM PC	8
IBM 1130	16
DEC PDP-8	12
DEC PDP-15	18
CDC 3600	48
CDC Cyber	60

Ngày nay hầu hết các máy vi tính có kích thước ô nhớ là 8 bit, còn gọi là **1 byte**, các byte lại được nhóm lại thành word (từ). Mỗi ***word gồm số byte mà hầu hết các lệnh của bộ xử lý thao tác như một đơn vị*** (thao tác làm một lần).

Như vậy một máy tính 16 bit sẽ có các thanh ghi 16 bit và các lệnh để xử lý (cộng, trừ, vận chuyển.v.v.) các word 16 bit.

2.3.3 Trật tự byte trong word, big endian và little endian computer

Các byte trong một word có thể được đánh số từ trái qua phải hoặc từ phải qua trái. Hình vẽ 2-09a minh hoạ một phần bộ nhớ của một máy tính có word dài 32 bit

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

với các byte được đánh thứ tự từ trái qua phải, như trong họ bộ xử lý của hãng Motorola. Hình 2-09b là một phần của bộ nhớ của một máy tính 32 bit với các byte được đánh thứ tự từ phải qua trái, như trong họ bộ xử lý của hãng Intel.

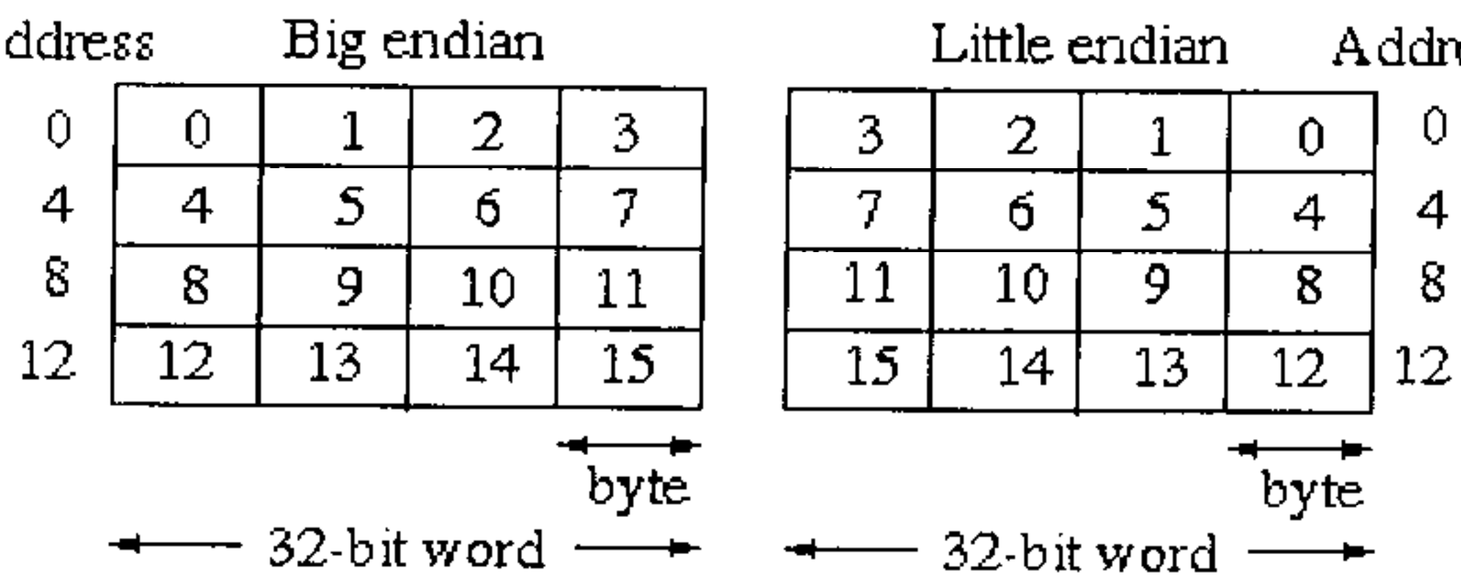
Big endian computer

Máy tính trong đó các từ (word) bộ nhớ có các byte được đánh thứ tự từ trái qua phải, nghĩa là các byte trọng số cao hơn đứng bên phải các byte có trọng số thấp hơn được gọi là ***big endian computer***.

Little endian computer

Máy tính trong đó các từ (word) bộ nhớ có các byte được đánh thứ tự từ phải qua trái, nghĩa là các byte trọng số cao hơn đứng bên trái các byte có trọng số thấp hơn được gọi là ***Little endian computer***.

{Thuật ngữ này được tác giả Jonathan Swift dùng trong tác phẩm "Những cuộc phiêu lưu của Giu-li-vơ" (Gulliver's Travels) để châm biếm các nhà chính trị đã gây ra chiến tranh do tranh luận về việc các quả trứng sẽ bị vỡ ở đầu to hay đầu nhỏ. Thuật ngữ này được dùng lần đầu tiên trong một bài báo của Cohen thuộc lĩnh vực kiến trúc máy tính năm 1981}



a/ Bộ nhớ máy tính Big endian b/ Bộ nhớ máy tính Little endian

Hình 2-09 Trật tự byte trong word của máy tính Big/Little endian

So sánh hai cách đánh thứ tự byte trong word của hai loại máy tính trên:

- Nếu bộ nhớ máy tính chỉ chứa số nguyên thì không có vấn đề gì nảy sinh do việc đánh thứ tự các byte từ trái qua phải hoặc từ phải qua trái. Cả hai cách biểu diễn này đều tốt và phù hợp nếu xét ở bên trong mỗi máy tính.
- Tuy nhiên có các ứng dụng yêu cầu việc lưu trữ dữ liệu xen kẽ các số nguyên, xâu ký tự và các kiểu dữ liệu khác. Thí dụ kiểu dữ liệu bản ghi (record) với các trường xâu ký tự (string) chứa tên người và các trường số nguyên (integer) chứa tuổi (21). Vấn đề sẽ nảy sinh khi một máy tính loại này gửi 1 record cho một máy tính loại khác hoặc ngược lại qua mạng máy tính.
- Giải quyết vấn đề này cũng không đơn giản lắm, một trong các cách giải quyết là thêm vào một trường header ở đầu mỗi record, header này chứa thông tin cho

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

biết record chứa những loại dữ liệu gì, mỗi trường dài bao nhiêu để cho bên nhận thực hiện các chuyển đổi cần thiết với mỗi trường.

2.3.4 Mã sửa sai

Bộ nhớ máy tính có thể có lỗi do các nguyên nhân khác nhau, thí dụ do các xung điện do tia lửa điện gây ra.v.v. Để khắc phục hiện tượng này người ta thường sử dụng các mã phát hiện sai (error-detecting code) hoặc các mã sửa sai (error-correcting code). Khi sử dụng các phương pháp này người ta bổ sung một nhóm bit vào mỗi word khi ghi vào bộ nhớ, khi đọc một word nhóm bit này sẽ được kiểm tra để xem xem word được đọc có lỗi hay không.

Giả sử 1 word có m bit dữ liệu, ta bổ sung thêm r bit kiểm tra, tổng chiều dài của word sẽ là n=m+r. Một đơn vị thông tin gồm m bit dữ liệu và r bit kiểm tra thường được gọi là **từ mã** n-bit (n-bit code word).

Khi cho 2 từ mã, thí dụ 1000 1001 và 1011 0001, ta có thể xác định được có bao nhiêu cặp bit tương ứng của 2 từ mã này là khác nhau. Để xác định con số này người ta thực hiện phép toán XOR 2 từ mã, sau đó đếm số bit 1 của kết quả. Con số vị trí bit mà 2 từ mã khác nhau được gọi là **khoảng cách Harming** (Harming distance). Nếu 2 từ mã có khoảng cách Harming là d thì phải có d bit bị lỗi mới có thể làm chuyển từ mã này thành từ mã kia. Thí dụ cho 2 từ mã 11110001 và 00110000 ta tính được khoảng cách Harming là 3.

Tính chất phát hiện lỗi hay sửa lỗi của một mã phụ thuộc vào khoảng cách Harming của nó. Để phát hiện được d bit lỗi, người ta phải dùng mã có khoảng cách Harming là d+1; tương tự như vậy để sửa được d bit lỗi, phải sử dụng mã có khoảng cách Harming là 2d+1.

Bit chẵn lẻ (Parity bit)

Thí dụ đơn giản nhất về mã phát hiện sai là trường hợp nhóm bit bổ sung chỉ gồm có 1 bit, ta gọi đó là bit chẵn lẻ (parity bit), bit chẵn lẻ được chọn tuỳ thuộc vào số bit 1 trong từ mã là chẵn hay lẻ.

Mã như vậy có khoảng cách Harming là 2, bởi vì bất kỳ một lỗi bit nào cũng làm cho từ mã bị sai tính chẵn lẻ, cho nên nó được sử dụng để phát hiện các lỗi đơn. Khi bộ xử lý đọc 1 word trong bộ nhớ có lỗi parity, nó sẽ phát hiện ra và thực hiện phản ứng cần thiết.

2.3.5 Bộ nhớ phụ

Vì bộ nhớ chính cần tốc độ hoạt động cao để các word của nó có thể được CPU đọc hay ghi một cách nhanh chóng, cho nên giá mỗi đơn vị nhớ cũng tương đối cao. Trong khi đó người sử dụng luôn yêu cầu phải có một bộ nhớ ngày một lớn hơn. Chính vì vậy mà người ta thường trang bị cho các máy tính một bộ nhớ thứ hai, được gọi là bộ nhớ phụ hay bộ nhớ ngoài (secondary/ external memory) có dung lượng lớn hơn nhiều so với bộ nhớ chính, nhưng rẻ hơn, chậm hơn. Hệ điều hành sẽ quản lý bộ nhớ này cùng với bộ nhớ chính sao cho đạt hiệu quả cao nhất.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Một số loại bộ nhớ ngoài đã và đang được sử dụng rộng rãi là băng từ, đĩa từ, đĩa quang. Các thiết bị nhớ ngoài sẽ được trình bày trong chương Các thiết bị ngoại vi thông dụng.

2.4 TỔ CHỨC VÀO/RA (I/O)

2.4.1 Các phương pháp tổ chức I/O

Để máy tính làm công việc giải một bài toán, trước hết phải trao cho nó chương trình và dữ liệu. Sau khi máy tính tìm được lời giải của bài toán, nó cần chuyển kết quả cho con người. Vấn đề đưa thông tin vào (input) và lấy thông tin ra (output) khỏi máy tính được gọi là vào/ra - input/output, thường được gọi ngắn gọn hơn là I/O.

Không phải mọi input đều do con người đưa vào và mọi output đều nhằm gửi ra cho con người. Một kính viễn vọng quan sát mặt trời được vận hành bằng máy tính có thể nhận các dữ liệu vào trực tiếp từ các thiết bị đang quan sát mặt trời. Một máy tính điều khiển một nhà máy hoá học tự động có thể hướng các output của nó tới các máy móc đang điều khiển việc sản xuất trong nhà máy.

Có nhiều phương pháp tổ chức I/O khác nhau, chúng ta có thể chia chúng thành 4 phương pháp chính như sau:

1. I/O được lập chương trình với việc luôn luôn đợi (Programmed I/O with busy waiting)
2. I/O được điều khiển bởi ngắt (Interrupt-driven I/O)
3. I/O sử dụng DMA (DMA I/O)
4. I/O sử dụng kênh dữ liệu (I/O using data channels)

- **Phương pháp I/O được lập chương trình**

Đây là phương pháp I/O đơn giản nhất, thường được sử dụng trong các hệ thống vi xử lý cần giá thành thấp. Trong quá trình hoạt động, CPU thực hiện một vòng lặp trong đó nó lần lượt "nhìn" ra tất cả các cổng để xem xem có cần nhận (input) hoặc gửi (output) dữ liệu hay không. Các lệnh input hoặc output thực hiện nhận/gửi từng byte một.

Nhược điểm lớn nhất của phương pháp này là CPU phải sử dụng hầu hết thời gian để kiểm tra các cổng liên hệ với các thiết bị I/O, chính vì vậy phương pháp này còn có tên gọi là luôn luôn chờ đợi (busy waiting).

- **Phương pháp I/O được điều khiển bởi ngắt**

Để tránh các nhược điểm của phương pháp trên, trong phương pháp này CPU sẽ chỉ khởi động sự làm việc của thiết bị I/O, khi thiết bị hoàn thành công việc thì nó sẽ gửi một tín hiệu báo lại cho CPU biết - đó là tín hiệu yêu cầu ngắt.

- **Phương pháp I/O sử dụng DMA**

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Phương pháp I/O được điều khiển bởi ngắt là một tiến bộ lớn so với phương pháp I/O được lập chương trình, tuy nhiên ta có thể thấy ngay nhược điểm là cứ mỗi lần truyền 1 ký tự lại phải gọi ngắt một lần - chính CPU phải thực hiện chương trình con xử lý ngắt, dẫn đến cần nhiều thời gian của CPU.

Trong phương pháp DMA, người ta sử dụng một đơn vị điều khiển, được gọi là DMAC (DMA Controler), nó có thể đọc/ghi một khối dữ liệu từ/vào bộ nhớ mà không cần sự can thiệp của CPU. Khi cần trao đổi dữ liệu với thiết bị I/O, CPU gửi cho DMAC địa chỉ bộ nhớ chứa dữ liệu, số byte cần đọc/ghi, số hiệu thiết bị và hướng truyền dữ liệu. Sau đó DMAC sẽ thực hiện công việc dựa trên các tham số mà CPU đã truyền cho nó, khi công việc hoàn thành, DMAC sẽ báo lại cho CPU.

- **Phương pháp I/O sử dụng kênh dữ liệu**

Phương pháp DMA có ưu điểm lớn là giải phóng CPU khỏi gánh nặng của công việc quản lý I/O, trong thời gian DMAC làm việc CPU có thể làm các công việc khác, nhưng CPU không được chiếm dụng bus trong khi DMAC làm việc, vì DMAC luôn có quyền ưu tiên chiếm dụng bus cao hơn CPU.

Trong các máy Mainframe, các thao tác I/O xảy ra liên tục và là công việc chính của CPU, nếu áp dụng phương pháp DMA thì chắc chắn bus cũng sẽ bị bão hoà (quá tải). Để giải quyết vấn đề này người ta sử dụng một đơn vị chuyên dụng, gọi là bộ xử lý I/O để hỗ trợ cho CPU chính. Đó là phương pháp I/O sử dụng kênh dữ liệu.

Các máy tính lớn và các máy tính PC thường áp dụng các phương pháp quản lý I/O khác nhau, các máy tính PC thường áp dụng kết hợp phương pháp I/O được điều khiển bởi ngắt và phương pháp I/O sử dụng DMA (phương pháp 2 và 3), các máy tính lớn thường theo phương pháp I/O sử dụng kênh dữ liệu (phương pháp 4). Dưới đây chúng ta sẽ nghiên cứu kỹ hơn các phương pháp I/O được sử dụng trong các máy mainframe và trong các máy PC.

2.4.2 Tổ chức I/O trong các máy tính lớn

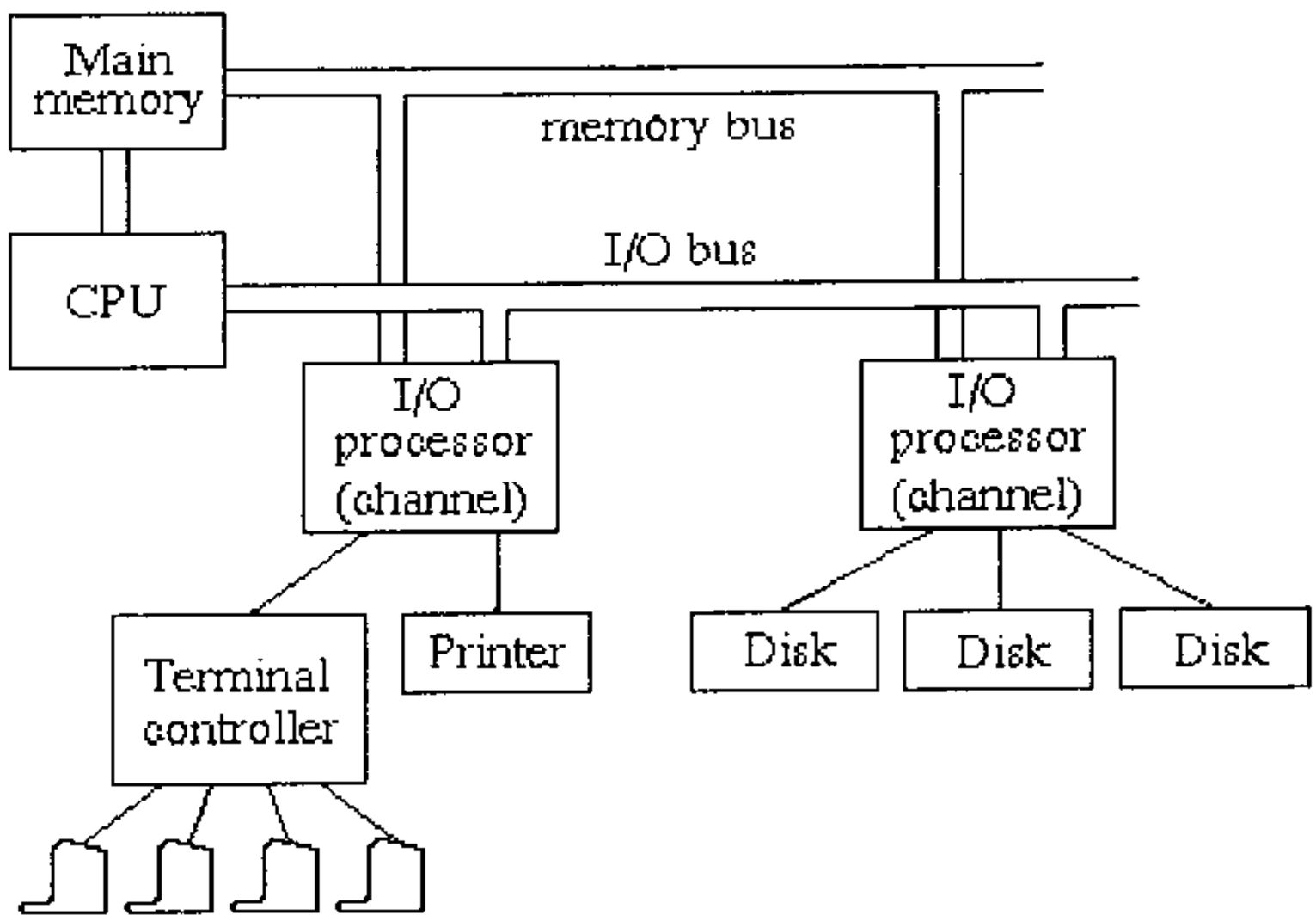
Tổ chức I/O trong các máy tính lớn thường theo phương pháp I/O sử dụng kênh dữ liệu (phương pháp 4), nguyên lý thiết kế như trên hình 2-10. Hệ thống máy tính này thường có một hay một số CPU, bộ nhớ chính (Main memory) và một hay một số bộ xử lý vào/ra chuyên dụng (I/O Processor) - còn được gọi là kênh dữ liệu, tất cả các thiết bị vào/ra được gắn với kênh này.

Khi CPU cần thực hiện một thao tác vào/ra, nó nạp một chương trình chuyên biệt cho một trong các bộ xử lý I/O và ra lệnh cho nó thực hiện chương trình này. Bộ xử lý I/O sẽ tự mình thực hiện các thao tác với bộ nhớ chính, giải phóng CPU khỏi công việc này, do đó CPU có thể thực hiện một công việc khác. Khi bộ xử lý I/O hoàn thành công việc, nó gửi cho CPU một tín hiệu đặc biệt gọi là ngắt (interrupt), buộc CPU ngừng công việc đang làm để chú ý tới đòi hỏi của bộ xử lý I/O.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Các máy Mainframe thường được sử dụng làm server (máy phục vụ) trong mạng máy tính, nhiệm vụ chính của nó là thực hiện các thao tác vào/ra, vì vậy thường được trang bị nhiều bus, tối thiểu là ba.

- Memory bus: cho phép bộ xử lý I/O đọc và viết bộ nhớ chính.
- I/O bus: cho phép CPU truyền các lệnh tới bộ xử lý vào/ra và để bộ xử lý vào/ra ngắt CPU.
- bus thứ ba: cho phép CPU truy cập bộ nhớ mà không phải sử dụng 2 bus kia.



Hình 2-10 Tổ chức vào/ra trong máy Mainframe

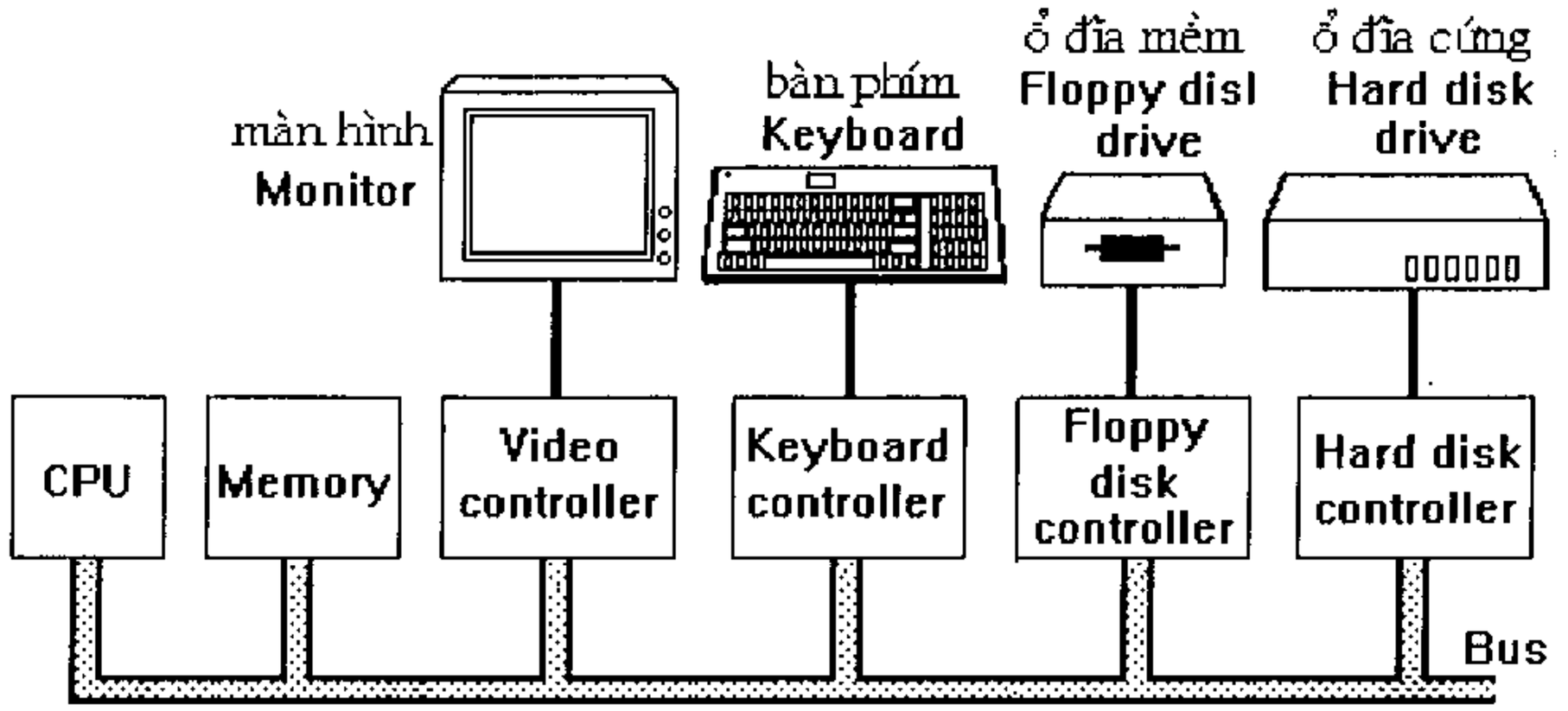
2.4.3 Tổ chức I/O trong các máy tính PC

Các máy tính PC sử dụng một cấu trúc hệ thống đơn giản hơn của Mainframe rất nhiều. Trong các máy PC thường chỉ có một bus, đó là kênh truyền thông tin chung, đôi khi người ta gọi là kênh vào/ra (I/O channel), nó nối tất cả các thành phần của máy tính điện tử với nhau. Cấu trúc logic của máy tính cá nhân được mô tả trên hình 2-11.

Các thiết bị I/O bao gồm 2 phần, một phần chứa hầu hết các mạch điện tử và có tên gọi là bộ điều khiển (controller), phần thứ hai thường chính là thiết bị vào/ra, thí dụ như ổ đĩa. Bộ điều khiển thường được lắp trên một bảng mạch (card) và được cắm vào một khe cắm (slot) trong máy. Bộ điều khiển có nhiệm vụ điều khiển thiết bị I/O, quản lý công việc truy cập bus.

Thí dụ khi một chương trình cần dữ liệu trên đĩa, nó sẽ gửi lệnh tới bộ điều khiển đĩa. Bộ điều khiển đĩa sẽ phát các lệnh tới thiết bị ổ đĩa để tìm dữ liệu và đọc, sau đó thiết bị ổ đĩa sẽ gửi về bộ điều khiển một dòng các bit, bộ điều khiển sẽ nhận và biến đổi dòng bit thành các word và viết vào bộ nhớ chính.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 2-11 Cấu trúc logic của máy tính cá nhân (Video controler: Bộ điều khiển hiển thị; Keyboard controler: Bộ điều khiển bàn phím; Floppy disk controler: Bộ điều khiển đĩa mềm; Hardisk controler: Bộ điều khiển đĩa mềm)

Có thể xảy ra tình huống cả CPU và bộ điều khiển vào/ra cùng đòi chiếm dụng bus. Để tránh xung đột người ta thường sử dụng một chip chuyên dụng làm trọng tài bus (bus arbiter), nó sẽ quyết định đơn vị nào được sử dụng bus trước.

Nói chung các thiết bị I/O được gán quyền ưu tiên cao hơn CPU, bởi vì nhiều thiết bị I/O có các thành phần cơ khí, như ổ đĩa chẳng hạn, nếu bắt chúng phải dừng lại chờ thì có thể bị mất dữ liệu hoặc sẽ làm giảm tốc độ xuống rất nhiều. Khi một thiết bị I/O chiếm dụng bus, nó sẽ dùng bus để truyền dữ liệu, quá trình này được gọi là "**cycle stealing**" ("lấy trộm chu kỳ") và nó làm chậm tốc độ của hệ thống. Các máy tính Mainframe có nhiều bus nên hiện tượng này ít ảnh hưởng tới tốc độ chung của hệ thống.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 3 MỨC LOGIC SỐ

Mức dưới cùng của kiến trúc phân tầng của một máy tính là mức logic số, đó là phần cứng thực sự của máy tính. Chương này nghiên cứu một số vấn đề của logic số, việc nghiên cứu nhấn mạnh đến các máy vi tính không những vì mục đích dễ hiểu mà còn vì các máy vi tính ngày càng trở nên quan trọng hơn.

Các phần tử cơ bản xây dựng nên mọi máy tính số đơn giản đến mức chúng ta sẽ phải ngạc nhiên. Chương này sẽ nghiên cứu các phần tử cơ bản của máy tính đồng thời sẽ nghiên cứu đại số logic dùng để phân tích chúng.

3.1 CÁC CỔNG VÀ ĐẠI SỐ LOGIC

Các mạch điện số có thể được xây dựng từ một số ít các phần tử rất đơn giản bằng cách kết hợp chúng theo những cách không phải là số. Mục này sẽ mô tả các phần tử đơn giản nhất, cách kết hợp chúng và cách sử dụng toán học để phân tích hành trạng của chúng.

3.1.1 Cổng (gate)

Mạch điện tử số là mạch điện trong đó chỉ biểu diễn 2 giá trị logic 0 và 1. Thí dụ tín hiệu điện nằm trong khoảng 0V-1V (V - vôn) biểu diễn một giá trị, chẳng hạn 0 (sau này chúng ta có thể sẽ gọi là thấp, mức thấp hoặc low), còn tín hiệu điện nằm trong khoảng 2V-5V biểu diễn giá trị kia, chẳng hạn 1 (sau này chúng ta có thể sẽ gọi là cao, mức cao hoặc high). Các điện áp nằm ngoài miền này là không được phép.

Những thiết bị điện tử rất nhỏ bé có thể thực hiện các chức năng khác nhau trên các giá trị nhị phân này được gọi là **cổng**, chính các cổng tạo nên cơ sở phần cứng của tất cả các loại máy tính.

Cổng có một hoặc một số lối vào (input) nhưng chỉ có một lối ra (output). Các giá trị vào hoặc ra chỉ có thể nhận một trong hai giá trị là 1 hoặc 0. Cổng còn thường được gọi là mạch logic bởi nó thực hiện các phép tính đại số logic.

Chi tiết về sự hoạt động bên trong của các cổng không thuộc khuôn khổ của giáo trình này, nó thuộc mức thiết bị - device level, nằm bên dưới mức 0 của chúng ta. Tuy nhiên tại mục này chúng ta sẽ nghiên cứu sơ lược để nắm được ý tưởng cơ sở.

Toàn bộ logic số hiện đại dựa trên thực tế là một transistor có thể được chế tạo để hoạt động như một chuyển mạch nhị phân hoạt động rất nhanh. Trên hình 3-01a là một transistor lưỡng cực được nối với một mạch điện đơn giản. Transistor này có 3 chân nối với thế giới bên ngoài, có tên là **collector** (cực góp), **emitter** (cực phát) và **base** (cực gốc).

- **Hình 3-01a**

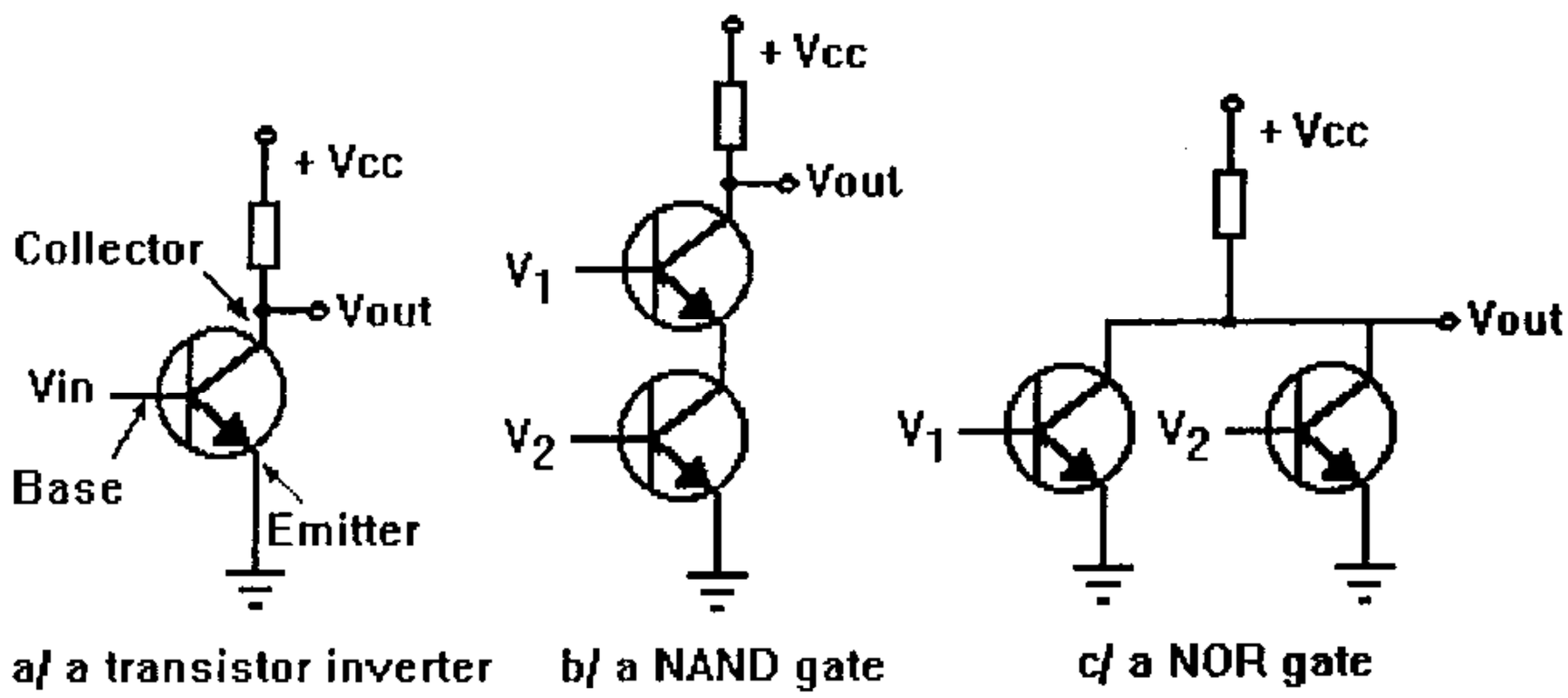
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Khi điện áp vào Vin thấp hơn một giá trị nào đó, transistor sẽ cấm (không có dòng điện đi qua) và thể hiện như một điện trở lớn vô cùng, làm cho lối ra của mạch điện là Vout nhận một giá trị xấp xỉ bằng Vcc, đó là một điện áp được ổn áp ở bên ngoài, thường là +5V.

Khi điện áp vào Vin vượt quá một giá trị giới hạn nào đó, transistor sẽ thông và thể hiện như một dây dẫn, làm cho lối ra của mạch điện là Vout được nối xuống đất, thường được quy ước là 0V.

Điều quan trọng là khi Vin thấp thì Vout cao và ngược lại. Mạch điện này chính là một bộ đảo, đảo giá trị logic 0 thành 1 và 1 thành 0. Điện trở trong mạch điện dùng để hạn chế dòng điện đi qua transistor.

Thời gian cần thiết để transistor chuyển từ trạng thái này sang trạng thái khác thường bằng vài nanô giây (1ns = 10^{-9}s).



Hình 3-01 a) cổng đảo; b) cổng NAND; c) cổng NOR

- **Hình 3-01b**

Trên hình 3-01b hai transitor được chồng nối tiếp lên nhau. Nếu cả V1 và V2 đều có mức cao (1) thì cả hai transitor đều dẫn điện và Vout sẽ được nối đất. Nếu một lối vào có mức thấp (0) thì transitor tương ứng sẽ cấm và Vout sẽ có mức cao. Nói tóm lại Vout có mức thấp khi và chỉ khi cả hai giá trị V1 và V2 là cao.

- **Hình 3-01c**

Trên hình 3-01c hai transistor được mắc song song. Theo cách mắc này nếu một lối vào nào đó có mức cao thì transitor tương ứng sẽ thông và lối ra được nối xuống đất. Nếu cả hai lối vào có mức thấp thì lối ra có mức cao.

Các cổng đơn giản nhất - NOT, NAND và NOR

Cả 3 mạch điện trên hình 3-01 hoặc các mạch tương đương của chúng tạo nên các cổng đơn giản nhất. Chúng được gọi là cổng NOT, cổng NAND và cổng NOR. Cổng NOT còn có tên gọi là inverter. Nếu chúng ta quy ước rằng điện áp cao - high

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

(Vcc) dùng để biểu diễn giá trị logic 1, còn điện áp thấp - low (đất) biểu diễn giá trị logic 0 thì chúng ta có thể biểu diễn giá trị ra như một hàm của các giá trị vào.

Các ký hiệu quy ước được sử dụng để mô tả các cổng này cùng với hành trạng chức năng của chúng được biểu diễn trên hình 3-02a-c.

Cổng AND

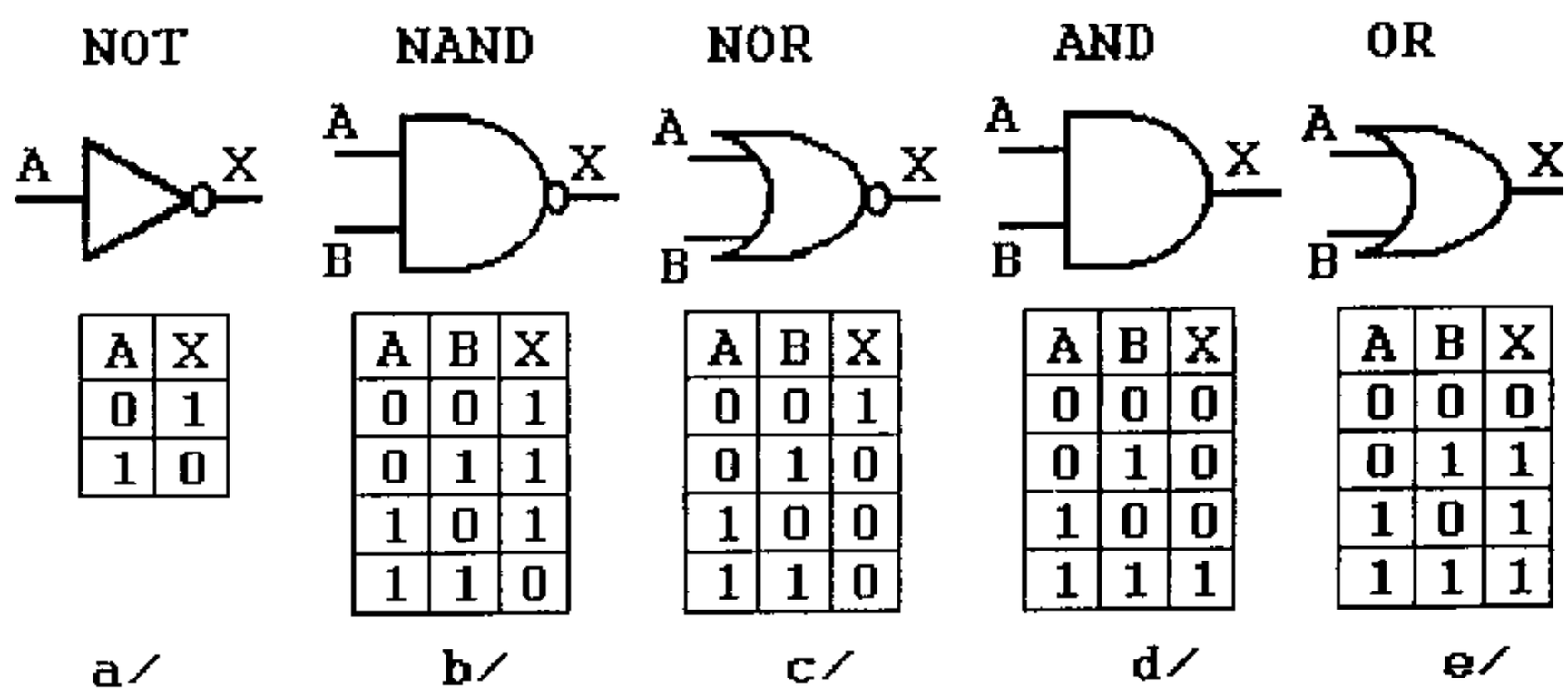
Nếu tín hiệu ra của cổng NAND trên hình 3-02b lại được đưa tới đầu vào của một mạch đảo, thì chúng ta có một mạch mà đầu ra của nó bằng 1 khi và chỉ khi các đầu vào đồng thời bằng 1. Mạch điện đó được gọi là cổng AND, ký hiệu và mô tả chức năng của nó được biểu diễn trên hình 3-02d.

Cổng OR

Tương tự như trên, đầu ra của cổng NOR nếu nối tới đầu vào của một cổng NOT thì tạo thành một mạch điện mà giá trị đầu ra của nó sẽ bằng 0 khi và chỉ khi giá trị ở cả hai đầu vào bằng 0, còn ngược lại thì sẽ bằng 1. Mạch điện đó được gọi là cổng OR, ký hiệu và mô tả chức năng của nó được biểu diễn trên hình 3-02e.

Vòng tròn nhỏ ở đầu ra các cổng NOT, NAND, NOR được gọi là vòng tròn đảo, chúng được sử dụng cả trong các ngữ cảnh khác để chỉ một tín hiệu bị đảo.

Năm cổng trên hình 3-02 là những phần tử xây dựng cơ bản của mức logic số. Từ những nghiên cứu trên ta thấy rằng các cổng NAND và NOR mỗi cổng chỉ cần sử dụng 2 transistor, trong khi đó các cổng AND và OR lại cần đến 3 transistor. Như vậy các cổng NAND và NOR đơn giản hơn các cổng AND và OR. Chính vì lý do này mà nhiều máy tính được xây dựng dựa trên các cổng NAND và NOR chứ không phải là các cổng AND và OR.



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Hình 3-02 Ký hiệu và hành trạng chức năng của 5 cổng cơ bản

3.1.2 Đại số logic

Để mô tả các mạch điện được xây dựng từ các cổng, chúng ta cần sử dụng **đại số logic** (Boolean algebra). Hàm logic có một hoặc một số biến vào, nó sẽ sinh ra một giá trị kết quả phụ thuộc vào các biến vào này. Một hàm *f* đơn giản có thể được

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

định nghĩa như sau: f(A) bằng 1 nếu A bằng 0 và f(A) bằng 0 nếu A bằng 1. Hàm này chính là hàm NOT, như trên hình 3-02a.

Bởi vì một hàm logic n biến chỉ có 2^n tập có thể các giá trị biến vào, cho nên hoàn toàn có thể mô tả hàm bằng một bảng có 2^n hàng, mỗi hàng cho giá trị của hàm ứng với mỗi tổ hợp khác nhau của các biến vào. Bảng như vậy được gọi là **Bảng chân lý** (Truth table), thí dụ các bảng trên hình 3-02.

Nếu chúng ta thống nhất liệt kê các hàng của bảng chân lý theo thứ tự số học (cơ số 2), nghĩa là cho 2 biến theo thứ tự 00, 01, 10 và 11 thì hàm hoàn toàn có thể được mô tả bởi một số nhị phân 2^n bit. Số này nhận được bằng cách đọc kết quả từ bảng chân lý theo cột, từ trên xuống dưới. Như vậy NAND sẽ là 1110, NOR là 1000, AND là 0001 và OR là 0111, như trên hình 3-02 b, c, d và e đã minh hoạ. Rõ ràng rằng chỉ tồn tại 16 hàm logic có hai biến, tương ứng với 16 dãy kết quả dài 4 bit có thể có. Trái lại trong đại số thông thường số hàm 2 biến là vô hạn, không thể mô tả một hàm nào bằng bảng các giá trị ra theo các giá trị vào có thể có, bởi vì mỗi biến có thể nhận một giá trị trong một miền vô hạn các giá trị có thể.

Mặc dù hầu hết các hàm logic có thể được biểu diễn bằng bảng chân lý, nhưng khi số biến tăng lên, cách biểu diễn này trở nên bất tiện vì cồng kềnh. Trong trường hợp đó người ta thường sử dụng một cách biểu diễn khác, được trình bày dưới đây.

Chúng ta có thể nhận thấy rằng mọi hàm logic có thể được chỉ rõ bằng cách cho biết những tổ hợp biến vào nào làm cho giá trị ra bằng 1. Chúng ta quy ước sẽ ghi một dấu gạch ngang bên trên các biến để chỉ rằng các giá trị này được đảo. Ngoài ra sẽ dùng các dấu phép toán nhân và cộng thông thường (dấu . và dấu +) để chỉ các phép toán logic AND và OR. Nếu như vậy việc viết $A\overline{B}C$ có nghĩa là A=1, B=0 và C=1. Tương tự như vậy $A\overline{B}+B\overline{C}$ có nghĩa là (A=1 and B=0) or (B=1 and C=0).

Một hàm n biến vì thế có thể được mô tả bằng cách cho một tổng của nhiều nhất là 2^n hạng thức tích n biến. Phát biểu này đặc biệt quan trọng bởi vì nó dẫn trực tiếp tới việc thực hiện hàm bằng việc sử dụng các cổng theo chuẩn.

3.1.3 Sự thực hiện các hàm logic

Như đã đề cập ở trên, phát biểu một hàm logic như là tổng của 2^n hạng thức tích dẫn trực tiếp tới việc có thể thực hiện được hàm bằng phần cứng. Ta có thể nêu ra cách xây dựng một mạch điện để thực hiện một hàm logic như sau:

1. Viết bảng chân lý của hàm.
2. Sử dụng các cổng NOT để tạo ra các giá trị đảo của từng biến vào.
3. Với mỗi hạng thức có một con 1 ở cột kết quả ta lấy ra một cổng AND.
4. Nối đầu vào các cổng AND tới các tín hiệu vào phù hợp.
5. Đưa các đầu ra của các cổng AND vào các đầu vào của một cổng OR.

Mặc dù chúng ta có thể sử dụng các cổng AND, OR và NOT để xây dựng mọi hàm logic bất kỳ, nhưng trong thực tế nếu chỉ sử dụng một loại cổng thì thường là

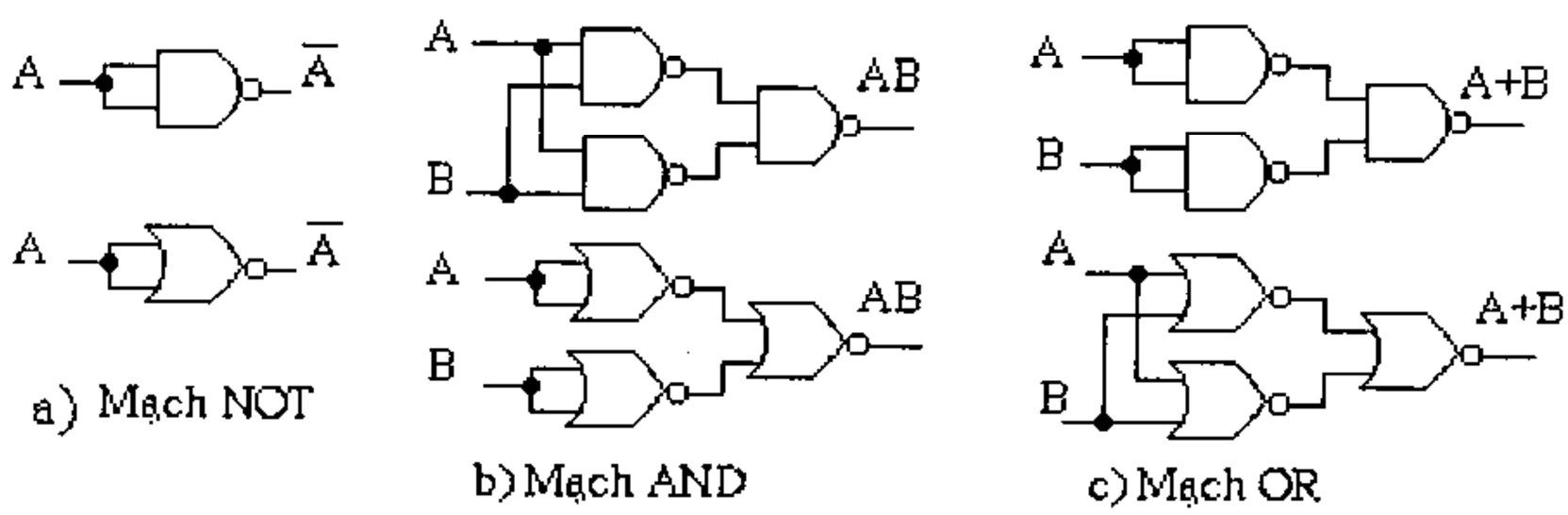
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

thuận lợi hơn. Người ta thường sử dụng hoặc là cổng NAND hoặc là cổng NOR để xây dựng các hàm logic. Cả hai cổng này đều được gọi là đầy đủ (**complete**) bởi vì mọi hàm logic đều có thể được xây dựng từ một trong hai loại cổng này. Không có một cổng nào khác có được tính chất này.

Hình 3-03 minh hoạ cách sử dụng cổng NAND và NOR để xây dựng các mạch logic thực hiện các hàm NOT, AND và OR.

3.1.4 Sự tương đương của các mạch

Những người thiết kế các mạch logic thường muốn giảm số các loại cổng khác nhau trong sản phẩm của mình nhằm giảm giá thành, giảm diện tích của mạch điện và công suất tiêu thụ.v.v. Để làm giảm độ phức tạp của mạch điện, người thiết kế phải tìm những mạch thực hiện được cùng chức năng theo yêu cầu nhưng lại chứa số cổng ít hơn hoặc có các cổng đơn giản hơn, chẳng hạn lấy các cổng có 2 đầu vào thay cho các cổng có 4 đầu vào. Để tìm kiếm được các mạch tương đương, cần sử dụng đại số logic.



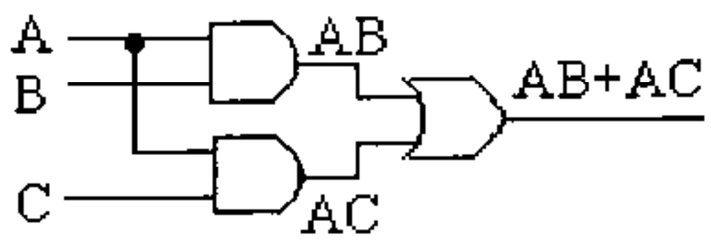
Hình 3-03 Các mạch logic có chức năng tương đương

Chúng ta cùng xem xét thí dụ minh hoạ cho việc sử dụng đại số logic trên hình 3-04. Hình 3-04a là mạch điện và bảng chân lý cho hàm AB+AC. Hình 3-04b là mạch điện và bảng chân lý cho hàm A(B+C). Hai hàm này là tương đương nhau nhưng mạch điện trên hình 3-04b rõ ràng là đơn giản hơn vì nó chứa ít cổng hơn.

Nói chung người thiết kế mạch logic có thể biểu diễn một mạch điện như một hàm logic và sau đó áp dụng các định luật của đại số logic lên biểu diễn này để tìm một biểu diễn đơn giản hơn nhưng tương đương. Từ biểu diễn cuối cùng sẽ xây dựng mạch điện.

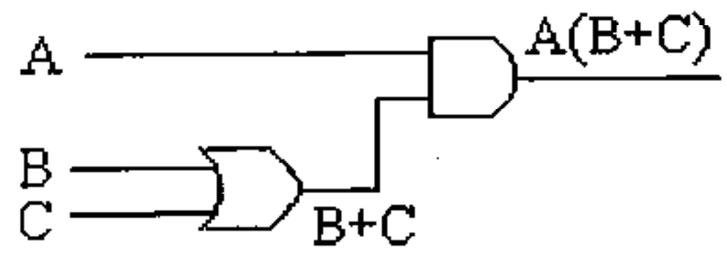
Để có thể sử dụng được cách tiếp cận này, chúng ta cần nắm được một số đồng nhất thức của đại số logic, chúng được trình bày tóm tắt trên bảng 3-01. Điều lý thú là mỗi định luật có 2 dạng đi đôi với nhau. Từ dạng này có thể chuyển sang dạng kia bằng cách hoán đổi giữa AND và OR, và giữa 0 và 1.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



A	B	C	AB	AC	AB+AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

a/



A	B	C	A	B+C	A(B+C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

b/

Hình 3-04 Hai hàm tương đương nhau nhưng số cổng khác nhau

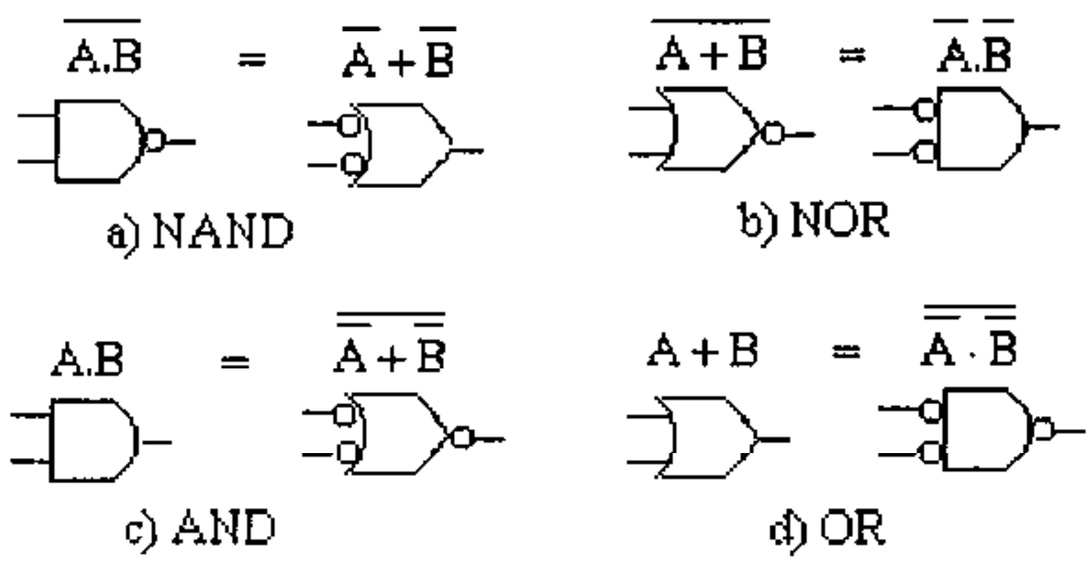
Bảng 3-01 Một số định luật của đại số logic

Tên	Dạng AND	Dạng OR
Định luật đồng nhất thức	1A=A	0+A=A
Định luật về số 0 (Null law)	0A=0	1+A=1
Định luật luỹ đẳng (Idempotent law)	AA=A	A+A=A
Định luật đảo (Inverse law)	$A\overline{A} = 0$	$A + \overline{A} = 1$
Định luật giao hoán (Commutative law)	AB=BA	A+B=B+A
Định luật kết hợp (Associative law)	(AB)C=A(BC)	(A+B)+C=A+(B+C)
Định luật phân phối (Distributive law)	A+BC=(A+B)(A+C)	A(B+C)=AB+AC
Định luật hấp thụ (Absorption law)	A(A+B)=A	A+AB=A
Định luật De Morgan	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\ \overline{B}$

Theo định luật De Morgan có thể dẫn tới các cách ghi ký hiệu khác như trên Hình 3-05. Hình 3-05a: ta thấy mạch NAND có thể được thực hiện bằng một mạch OR với cả 2 lối vào đảo, các lối vào này có khuyên nhỏ thể hiện việc thực hiện đảo

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

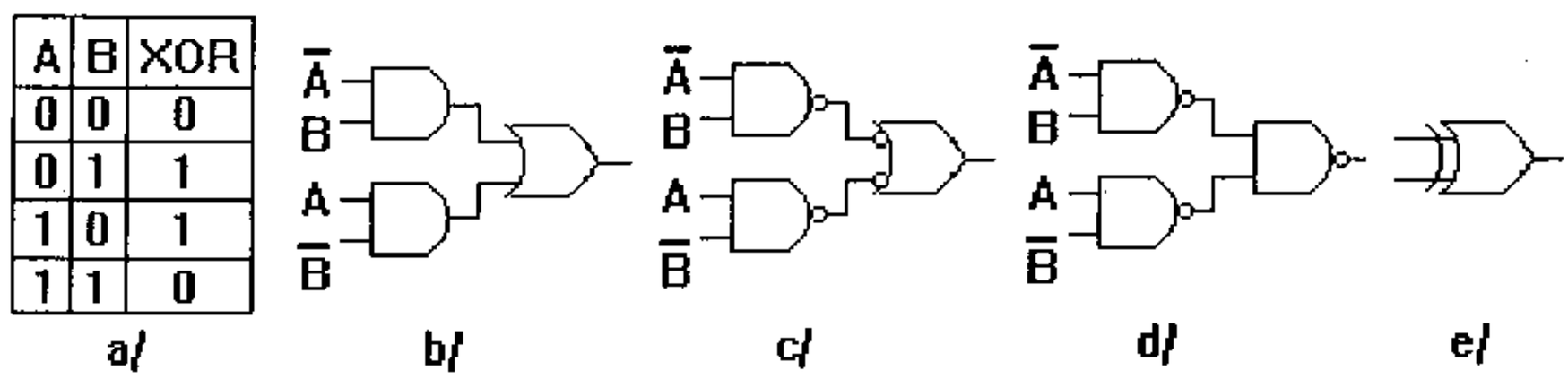
ở đầu vào. Hình 3-05b: cổng NOR có thể thực hiện bằng một cổng AND có 2 lối vào đảo. Các hình c, d cũng là những thể hiện của định luật De Morgan, đó là những ký hiệu tương đương của cổng AND và OR.



Hình 3-05 Một số cách ký hiệu khác của các cổng

Sử dụng các mạch tương đương trên hình 3-05, chúng ta có thể dễ dàng biến đổi dạng biểu diễn tổng các tích của một bảng chân lý thành một dạng gồm toàn NAND hoặc toàn NOR.

Thí dụ xem xét hàm XOR (Exclusive OR) được định nghĩa bằng bảng chân lý trên hình 3-06a. Hình 3-06b là mạch điện tổng của các tích chuẩn. Để chuyển sang dạng NAND, các đường dây nối các đầu ra của các cổng AND tới các đầu vào của cổng OR cần được vẽ lại với 2 vòng trong đảo ở các đầu, như trên hình 3-06c. Cuối cùng áp dụng hình 3-05a chúng ta có được mạch điện như trên hình 3-06d. Hình 3-06e là ký hiệu của mạch XOR trên sơ đồ.



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Hình 3-06 Bảng chân lý của hàm XOR và các mạch điện thực hiện hàm này

Chú ý: Cùng một cổng logic (cổng vật lý) có thể thể hiện các chức năng logic khác nhau tuỳ theo sự quy ước về các mức điện áp nào thể hiện giá trị logic nào!

- Nếu ta quy ước rằng mức điện áp 0v thể hiện mức logic 0 còn mức điện áp +5v thể hiện mức logic 1 thì ta có **logic dương (positive logic).**
- Nếu ta quy ước rằng mức điện áp 0v thể hiện mức logic 1 còn mức điện áp +5v thể hiện mức logic 0 thì ta có **logic âm (negative logic).**
- Khi người ta không nói rõ là sử dụng loại logic nào thì có nghĩa là sử dụng logic dương. Điều đó có nghĩa là các thuật ngữ sau là tương đương:
  + logic 1, true, high
  + logic 0, false, low

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

3.2 CÁC MẠCH LOGIC SỐ CƠ BẢN

Ngày nay rất hiếm vi mạch được chế tạo ra mà chỉ có một cổng, các vi mạch có bán trên thị trường thường chứa một số cổng để giảm giá thành, kích thước.v.v.

3.2.1 Mạch tích hợp - Integrated Circuits (IC)

Các cổng được sản xuất ra không phải dưới dạng mỗi đơn vị sản phẩm chứa một cổng mà trong một đơn vị có chứa đựng nhiều cổng, người ta gọi những đơn vị này là mạch tích hợp, hay còn gọi là IC (Integrated Circuits), hay là chip. Các chip thường có kích thước trong khoảng: rộng cỡ (5-15)mm, dài cỡ (20-50)mm. Số chân của một chip có thể là 14,16,18,20 .. 68... Các chip lớn có chân chìa ra ở cả 4 cạnh.

Người ta thường phân loại chip theo số lượng cổng chứa trong chip một cách tương đối thành một số loại như trong bảng 3-02.

Bảng 3-02 Phân loại chip theo số lượng cổng

Ký hiệu	Số cổng/chip
SSI (Small Scale Integrated)	1-10
MSI (Medium Scale Integrated)	10-100
LSI (Large Scale Integrated)	100-100.000
VLSI (Very Large Scale Integrated)	> 100.000

Hiện nay người ta có thể đặt hàng triệu cổng trong một chip, chip có một triệu cổng có thể cần tới 3,000,002 chân! (1 triệu cổng x 3 chân/cổng + 2 chân cho nguồn nuôi), tuy nhiên điều này không thể chấp nhận được. Trong thực tế các chip có nhiều cổng thường chứa một số đơn vị logic để thực hiện các chức năng nhất định, mỗi đơn vị logic thường được xây dựng từ một số cổng, các cổng này liên hệ với nhau thông qua các đường dây dẫn ngay bên trong chip, có thể chỉ cần một số ít, thậm chí không cần các chân đưa ra ngoài. Các đơn vị chức năng trong một chip cũng có thể liên hệ với nhau bằng các đường dây dẫn bên trong chip. Một số chân của chip có thể được các đơn vị chức năng dùng chung theo kiểu phân chia thời gian. Trong công nghệ chế tạo chip người ta phải thiết kế các chip sao cho có tỉ số cổng/chân ra cao.

3.2.2 Các mạch tổ hợp (Combinational Circuit)

Nhiều ứng dụng logic số cần đến các mạch có nhiều đầu vào và nhiều đầu ra trong đó các giá trị ra hoàn toàn được xác định bởi các giá trị đầu vào ở thời điểm đang xét. Những mạch như vậy được gọi là **Mạch tổ hợp** (Combinational Circuit), các mạch này thường được thể hiện bởi bảng chân lý . Không phải tất cả các mạch

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

điện đều có tính chất này (chẳng hạn các mạch nhớ). Trong mục này sẽ khảo sát một số mạch tổ hợp thông dụng, chúng thường là các chip MSI.

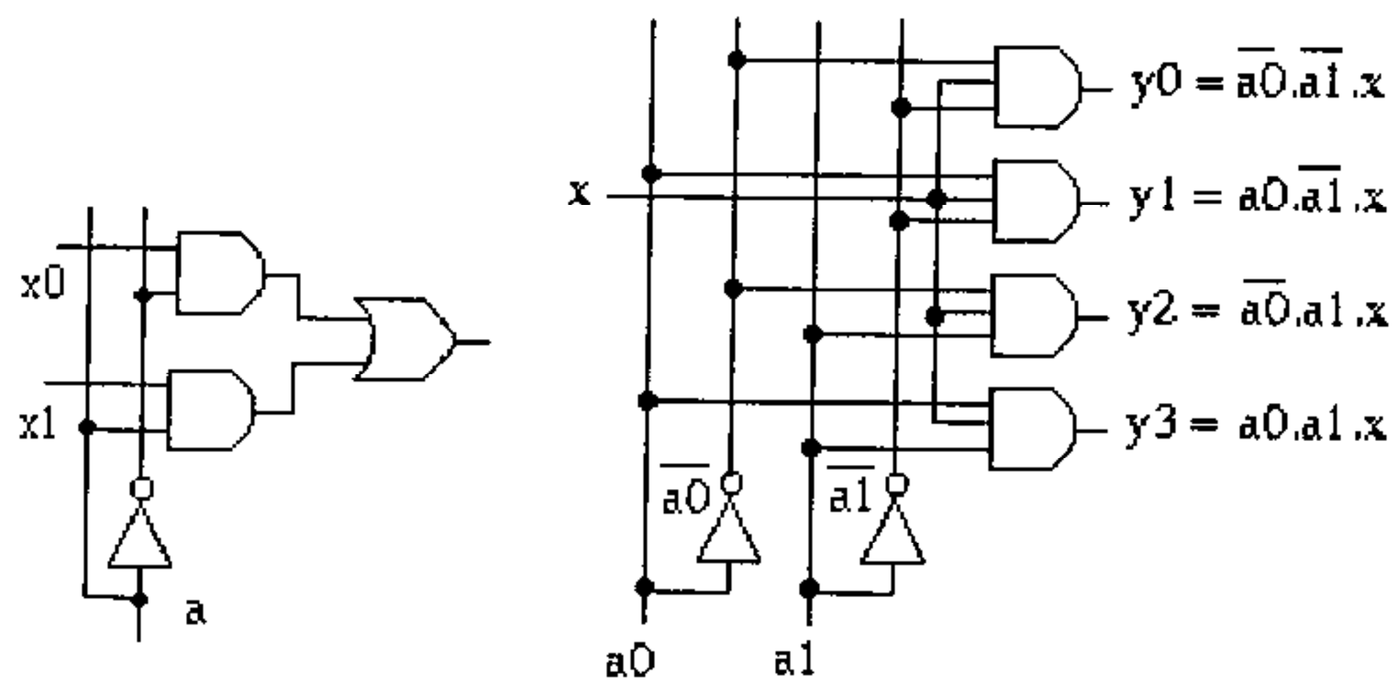
3.2.2.1 Bộ dồn kênh và bộ phân kênh

Bộ dồn kênh - Multiplexer

Là một mạch điện với 2^n lối vào số liệu, một lối ra số liệu, và n lối vào điều khiển. Một trong số 2^n đầu vào số liệu sẽ được chọn để cho đi qua cổng. Hình 3-07a là một Multiplexer có 2 đường vào số liệu, 1 đường vào điều khiển.

Bộ phân kênh - Demultiplexer

Mạch điện thực hiện chức năng ngược với multiplexer được gọi là demultiplexer, nó cho tín hiệu vào (chỉ có một đầu vào) đi ra một trong số 2^n đường ra tuỳ theo tín hiệu trên n đầu vào điều khiển. Nếu giá trị của tín hiệu điều khiển = k thì đường ra thứ k được chọn để đi ra. Hình 3-07b là một demultiplexer có 1 đường vào số liệu x, 2 đường vào điều khiển a0, a1 và 4 đường ra y0, y1, y2 và y3.



a/ Multiplexer **b/ Demultiplexer**

Hình 3-07 Bộ dồn kênh và bộ phân kênh

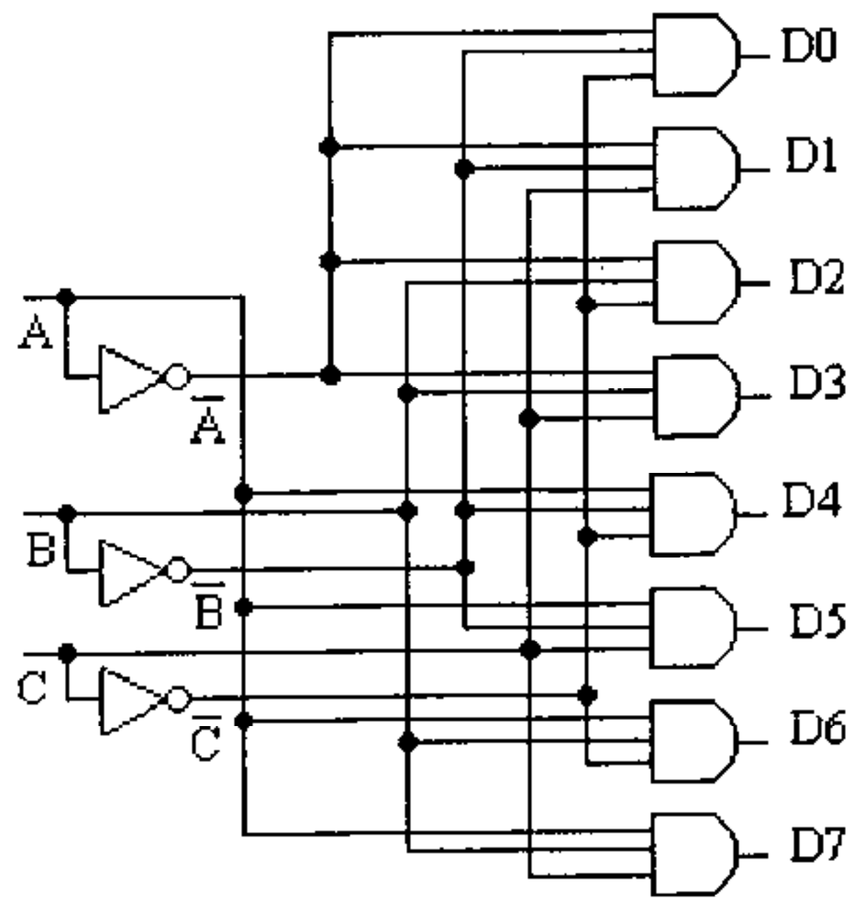
3.2.2.2 Bộ giải mã (Decoder)

Đó là mạch điện nhận con số n-bit đầu vào để chọn (nghĩa là đặt mức logic 1) cho duy nhất một trong số 2^n đầu ra. Hình 3-08 là một Decoder với n=3.

Để thấy được công dụng của decoder, ta hình dung một bộ nhớ có 8 chip nhớ, mỗi chip 8KB, chip 0 có các địa chỉ từ 0 đến 8191, chip 1 có các địa chỉ từ 8192 đến 16383 .v.v.

Khi bộ vi xử lý phát tín hiệu trên bus địa chỉ tới bộ nhớ, 3 bit bậc cao sẽ được dùng để chọn 1 trong 8 chip, đó là 3 bit A, B, C ở hình 3-08. Với 1 giá trị của con số 3 bit A, B, C chỉ có 1 dây Di (i=0..7) có mức 1, các dây còn lại đều có mức 0, các dây này được nối vào đầu Chip Select (CS hoặc CE) của 8 chip, vì vậy chỉ có 1 chip là được phép mở.

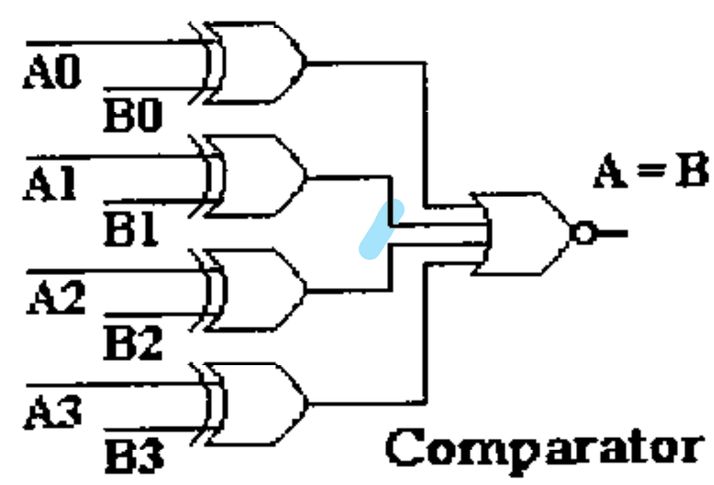
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 3-08 Mạch giải mã "3 to 8"

3.2.2.3 Bộ so sánh (Comparator)

Đó là mạch điện thực hiện so sánh 2 từ đưa vào, trên hình 3-09 là một bộ so sánh 2 toán hạng nhị phân 4 bit. Hai từ đưa vào là A = A3A2A1A0 và B = B3B2B1B0, trong đó Ai và Bi (i=0..3) là các bit bậc i của từ A và B. Đầu ra A=B sẽ có giá trị 1 nếu 2 từ đưa vào là bằng nhau, ngược lại sẽ có giá trị 0.



Hình 3-09 Bộ so sánh 4 bit

Mạch điện này dùng các cổng XOR 2 lối vào và 1 cổng NOR 4 lối vào. Trên thị trường cũng còn có các bộ so sánh với đầu ra A>B hoặc A<B.

3.2.2.4 Mảng logic định trình được - PLA (Programmed Logic Arrays)

Theo đại số logic, một hàm tuỳ ý hay một bảng chân lý có thể được xây dựng bởi việc tính các hạng thức bằng các cổng AND, sau đó cộng (OR) các hạng thức ấy lại. Những chip điển hình để thực hiện cộng các tích lại là chip PLA.

Hình 3-10 là chip PLA có ký hiệu 74S330, chip này có các đường vào cho 12 tín hiệu vào - biến, giá trị đảo của các biến vào được tạo thành bên trong chip, như vậy có 24 biến vào.

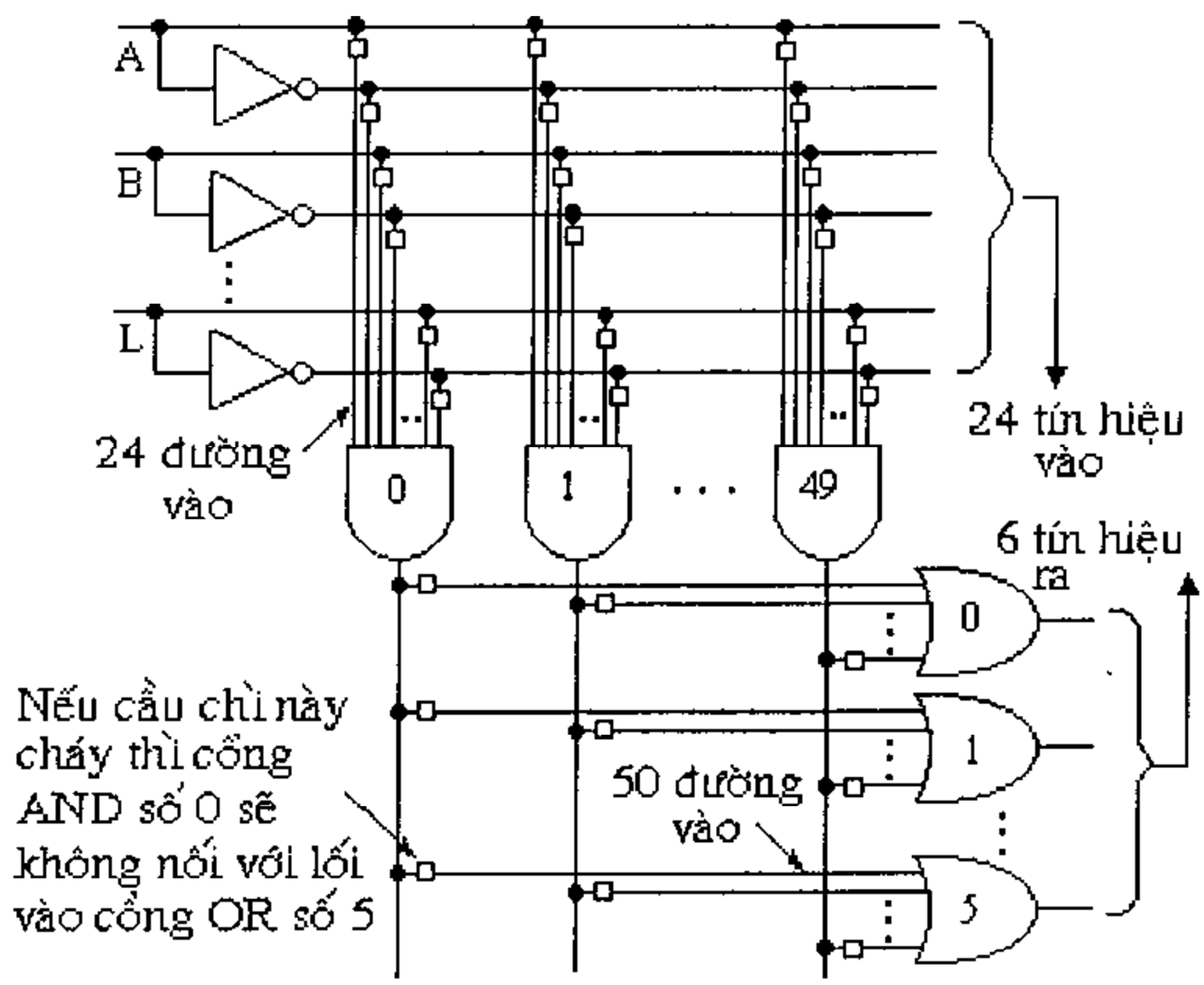
Phần chính của mạch này là một ma trận 50 cổng AND, mỗi cổng này có thể nối các đầu vào của nó với một tập con tuỳ ý của 24 đường tín hiệu vào.

Tín hiệu vào nào đi tới cổng AND nào được quyết định bởi một ma trận 24x50 bit do người sử dụng thiết lập. Mỗi tín hiệu vào trong số 24 tín hiệu đi tới đầu vào của tất cả các cổng phải qua 1 cầu chì. Ban đầu, khi mới xuất xưởng tất cả các cầu chì đều còn thông. Người sử dụng sẽ làm cháy các cầu chì được lựa chọn bằng cách sử dụng một thiết bị chuyên dụng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Phần ra của mạch điện có 6 cổng OR, mỗi cổng này có đến 50 đầu vào ứng với 50 đầu ra của 50 cổng AND, ở phần này ta cũng thấy một ma trận 50x6 do người sử dụng thiết lập các mối nối dây hàng - dây cột thông qua các cầu chì.

Chip 74S330 có 12 chân cho 12 biến vào, 6 chân cho 6 đường dây ra, 1 chân cho nguồn nuôi, 1 chân cho dây đất, tổng cộng là 20 chân ra.



Hình 3-10 Chip PLA 74S330

3.2.3 Các mạch số học

3.2.3.1 Bộ dịch (Shifter)

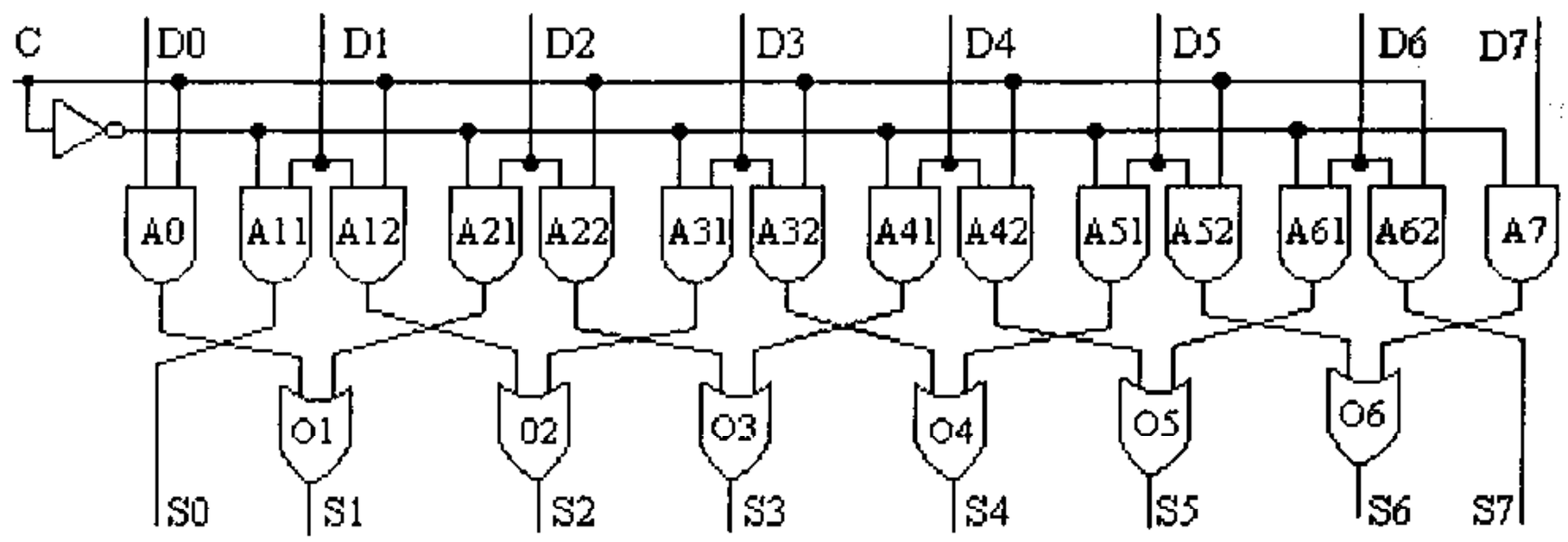
Dịch bit là một thao tác cơ sở, một số phép toán số học và logic của các bộ vi xử lý trong máy tính dựa trên thao tác này. Trên hình 3-11 là bộ dịch bit có 8 đầu vào (D0 .. D7), 8 đầu ra (S0 .. S7), đường điều khiển C xác định hướng dịch chuyển của các bit, nếu C=1 thì dịch phải một vị trí, nếu C=0 thì dịch trái một vị trí.

Khi C=1 các cổng AND bên trái trong các cặp cổng AND, như A11, A21,.. A61 và A7 đều có 1 lối vào bằng 0, do đó đầu ra của các cổng AND này đều bằng 0 và dẫn đến việc tất cả các cổng OR từ O1 đến O6 đều có lối vào bên phải bằng 0, vì vậy các đầu ra S1..S6 sẽ nhận giá trị của các đầu vào bên trái. Kết quả là S1=D0, S2=D1, ...và S6=D5, còn lại S0=0 và S7=D6. Như vậy khi C=1 các bit đưa vào đã được dịch phải một vị trí và đưa ra lối ra.

Cho C=0 và phân tích tương tự như trên ta sẽ thấy rằng các bit đưa vào sẽ được dịch trái một vị trí và đưa ra lối ra.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

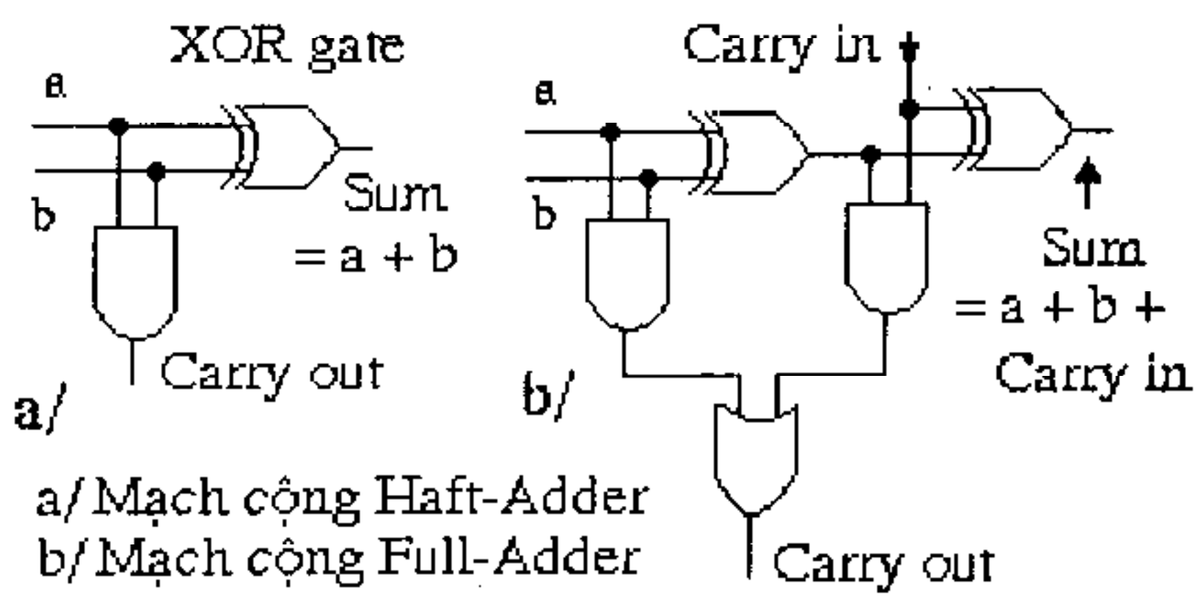
(Chú ý: về hướng dịch sang trái hay sang phải là theo hình 3-11, trong hình này trật tự các bit tính từ trái qua phải, ngược lại với trật tự mà chúng ta thường ghi trên giấy như D7D6...D0, S7S6...S0.)



Hình 3-11 Bộ dịch 1 bit sang trái/phải

3.2.3.2 Bộ cộng - Adder

Các mạch thực hiện phép cộng là những phần rất căn bản của mọi CPU. Hình 3-12a là mạch tính tổng 2 bit, trả lại tổng - sum và số nhớ - carry-out để có thể dùng cho phép cộng 2 bit ở hàng cao hơn, nó còn được gọi là **Haft_Adder.** Bộ cộng này không có lối vào cho số nhớ Carry-in cho nên không dùng để cộng các bit bậc cao hơn 0.



Hình 3-12 Bộ cộng

Hình 3-12b là mạch tính tổng 2 bit vào và bit nhớ carry-in, trả lại tổng - sum và số nhớ - carry-out để có thể dùng cho phép cộng bit bậc cao hơn, nó còn được gọi là **Full_Adder.** Bộ cộng này có lối vào carry-in, nó được xây dựng từ 2 bộ Haft-Adder.

Để xây dựng các bộ cộng với từ dài hơn, chẳng hạn từ 16 bit, chỉ cần lặp lại mạch điện của hình 3-12b 16 lần. Bit nhớ carry-out của một hàng bit được sử dụng làm bit carry-in cho việc cộng 2 bit của hàng cao hơn nó một bậc. Đầu vào carry-in của bit bậc thấp nhất được nối với 0. Một bộ cộng như vậy được gọi là **Ripple Carry Adder** (ripple- làm gợn sóng...), có tên gọi như vậy bởi vì trong trường hợp cộng 2 số: 1 và 111...111 phép cộng chỉ được hoàn thành khi số nhớ đã đi suốt từ bit

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

thấp nhất đến bit cao nhất, điều này làm chậm phép tính. Người ta cũng xây dựng các bộ cộng không có nhược điểm này nhưng chúng phức tạp hơn.

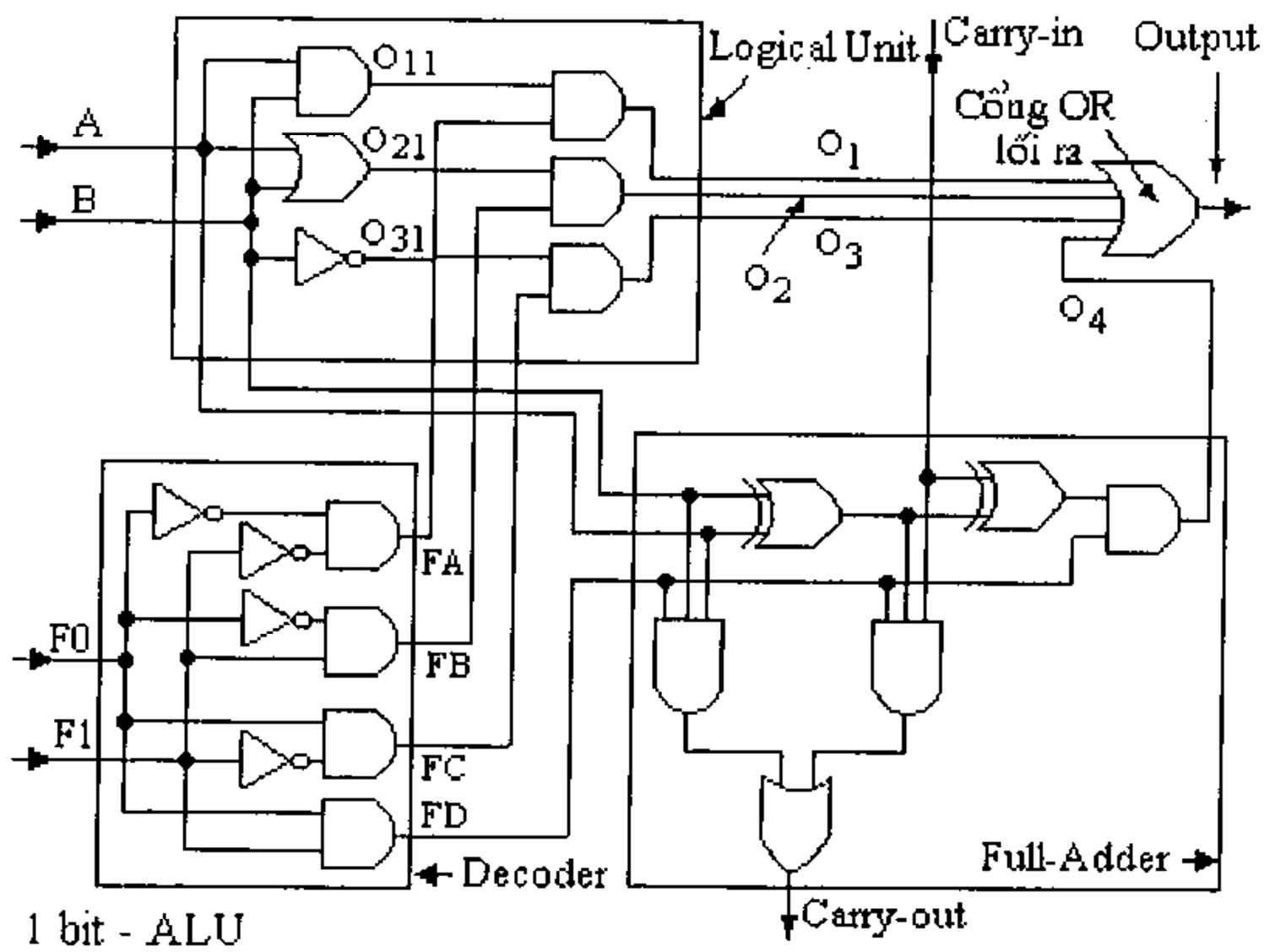
3.2.3.3 Bộ số học và logic - ALU (Arithmetic Logical Unit)

Hầu hết các bộ vi xử lý đều có mạch riêng để thực hiện các phép tính AND, OR và tính tổng của 2 từ máy. Mạch điển hình để tính toán với các từ máy n-bit được xây dựng từ n mạch điện hoàn toàn giống nhau, mỗi mạch này sẽ tính toán với 1 vị trí bit của 2 từ.

Hình 3-13 là một ví dụ đơn giản về mạch tính toán số học - ALU. Tuỳ theo tín hiệu điều khiển chọn chức năng F0 và F1 là 00,01,10 hay 11, nó sẽ thực hiện một trong 4 chức năng sau:

- A AND B
- A OR B
- NOT B (đảo B)
- A + B

Góc dưới bên trái của hình vẽ là bộ giải mã 2-bit (decoder), tuỳ thuộc các giá trị điều khiển đầu vào F0 và F1 mà một trong 4 dây ra FA, FB, FC, FD có mức logic 1, còn 3 dây kia có mức logic 0.



Hình 3-13 Bộ Số học và Logic (ALU) 1 bit

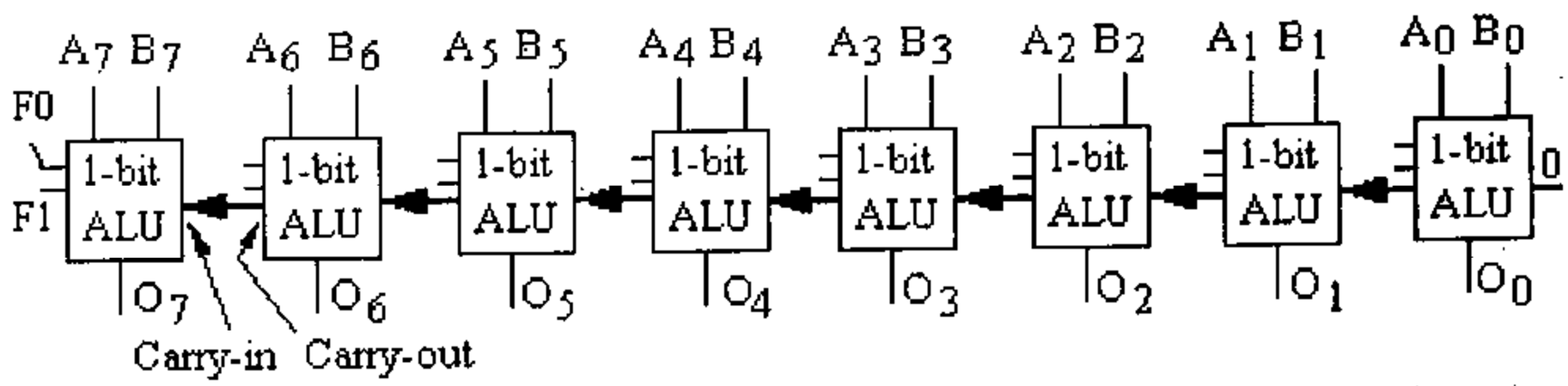
Khối ở góc trên bên trái là khối logic có nhiệm vụ tính A AND B, A OR B, và NOT B. Khối Full-Adder ở góc dưới bên phải hình vẽ thực hiện tính A+B, kể cả việc xử lý số nhớ đưa vào carry-in. Bốn phép tính này cùng được thực hiện nhưng chỉ có một trong bốn kết quả được đưa ra đầu ra Output, bởi vì tín hiệu điều khiển sẽ làm cho 3 trong 4 đầu vào của cổng OR lối ra có giá trị 0, do đó giá trị ra Output sẽ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

bằng giá trị tại đầu vào còn lại, là kết quả của một trong bốn phép tính (tất nhiên kết quả này có thể là 1 hoặc 0).

Các chip như trên hình vẽ 3-13 đã từng có mặt trên thị trường với tên gọi **bit-slices** (slice- miếng mỏng, lát). Sử dụng các chip này song song với nhau người thiết kế có thể xây dựng nên ALU có độ rộng tuỳ ý, thực hiện các tính toán trên word.

Hình 3-14 là một ALU 8-bit được xây dựng từ 8 bộ ALU kiểu bit-slices. Trên thị trường cũng có các chip ALU chứa nhiều bộ ALU kiểu bit-slices, do đó việc thiết kế được đơn giản đi nhiều, số lượng chip cũng giảm. Các chip này cũng có thể kết hợp lại để tạo thành các ALU nhiều bit hơn.



Hình 3-14 ALU 8-bit được tạo bởi "8 mảnh" ALU 1-bit

3.2.3.4 Clock - Bộ tạo tín hiệu thời gian

Trong rất nhiều mạch điện số, trật tự xảy ra các sự kiện là hết sức quan trọng. Đôi khi một sự kiện nhất thiết phải xảy ra trước sự kiện khác, có khi 2 sự kiện lại phải xảy ra đồng thời. Để thực hiện được mối quan hệ về mặt thời gian như mong muốn, người thiết kế phải sử dụng nhiều mạch số sử dụng đồng hồ để cung cấp tín hiệu đồng bộ các quá trình.

Trong lĩnh vực kỹ thuật số, đồng hồ là một mạch điện phát ra chuỗi xung điện, trong đó độ rộng của mỗi xung cũng như chu kỳ lặp lại xung rất ổn định và chính xác. Chu kỳ xung cũng còn được gọi là thời gian chu kỳ đồng hồ. Tần số xung nằm trong khoảng từ 1 đến 100MHz, tương ứng có chu kỳ đồng hồ là 1μs cho đến 10ns. Để đạt được độ chính xác cao và ổn định, người ta thường sử dụng máy phát thạch anh (Crystal Oscilator).

Trong máy tính, nhiều sự kiện có thể xảy ra trong một chu kỳ đồng hồ. Nếu các sự kiện này phải xảy ra theo một trật tự nhất định thì chu kỳ đồng hồ phải được chia ra thành các chu kỳ con (subcycles). Người ta thường sử dụng một giải pháp rất hay là bổ sung thêm một mạch điện, mạch này nối với đường tín hiệu đồng hồ chính qua một bộ phận làm trễ (delay) - như vậy có ngay một tín hiệu đồng hồ thứ hai dịch pha so với tín hiệu đồng hồ chính. (Xem hình 3-15a).

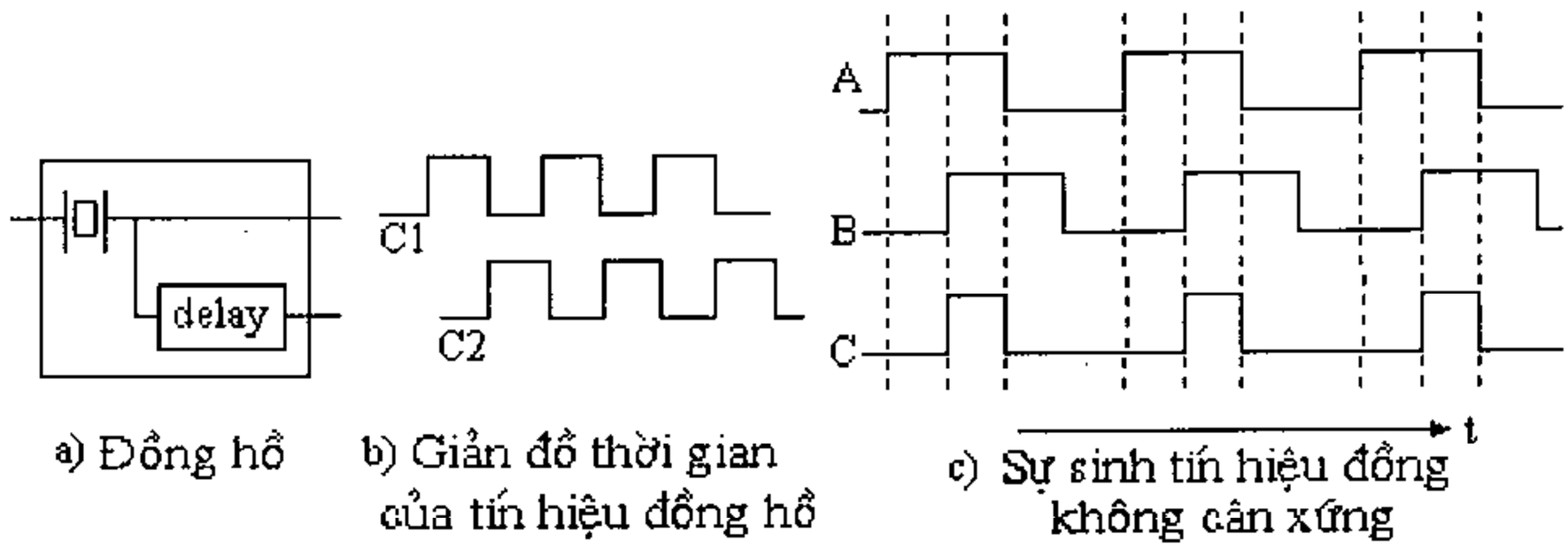
Giản đồ thời gian trên hình 3-15b cho thấy có 4 tín hiệu thời gian cho các sự kiện khác nhau:

1. Sườn dương của C1

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

2. Sườn âm của C1
3. Sườn dương của C2
4. Sườn âm của C2

Bằng cách gắn các sự kiện với các sườn xung khác nhau, người ta thực hiện được việc phân bổ thứ tự của các sự kiện. Nếu hệ thống cần trên 4 tín hiệu đồng hồ trong một chu kỳ máy, thì các đường tín hiệu đồng hồ phụ sẽ được bổ sung vào, mỗi đường có độ trễ khác nhau.



Hình 3-15 Tạo tín hiệu thời gian

Trong một số mạch điện người ta cần các mức tín hiệu đồng hồ kéo dài trong một khoảng thời gian chứ không phải trong các thời điểm rời rạc. Thí dụ một số sự kiện có thể được phép xảy ra trong khoảng thời gian C1 ở mức cao, chứ không phải chính xác ở sườn dương của C1. Nếu cần tới trên 2 tín hiệu như vậy người ta có thể dùng nhiều đường tín hiệu đồng hồ hoặc cho các trạng thái có mức cao của 2 tín hiệu đồng hồ trùm phủ một phần lên nhau (theo trục thời gian). Trong trường hợp dùng 2 tín hiệu đồng hồ trùm phủ một phần lên nhau ta có các khoảng thời gian tách biệt sau:

$$\overline{C1} \text{ AND } \overline{C2},\ \overline{C1} \text{ AND } C2,\ C1 \text{ AND } \overline{C2},\ C1 \text{ AND } C2$$

Trên hình 3-15c, tín hiệu c) có được bằng cách thực hiện AND các tín hiệu a) và b).

3.3 BỘ NHỚ

Bộ nhớ là một trong các thành phần quan trọng nhất của máy tính điện tử, được dùng để lưu trữ các lệnh sẽ được thực hiện cũng như dữ liệu. Bộ nhớ được xây dựng từ các phần tử nhớ cơ bản, mỗi phần tử nhớ cơ bản có thể nhớ được 1 bit thông tin; Đó là các mạch điện có hai trạng thái cân bằng ổn định, thuật ngữ tiếng Anh là flip-flop, flip-flop sẽ nằm mãi ở một trong hai trạng thái cân bằng nếu không có tín hiệu điện phù hợp kích thích vào làm cho nó phải thay đổi trạng thái.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

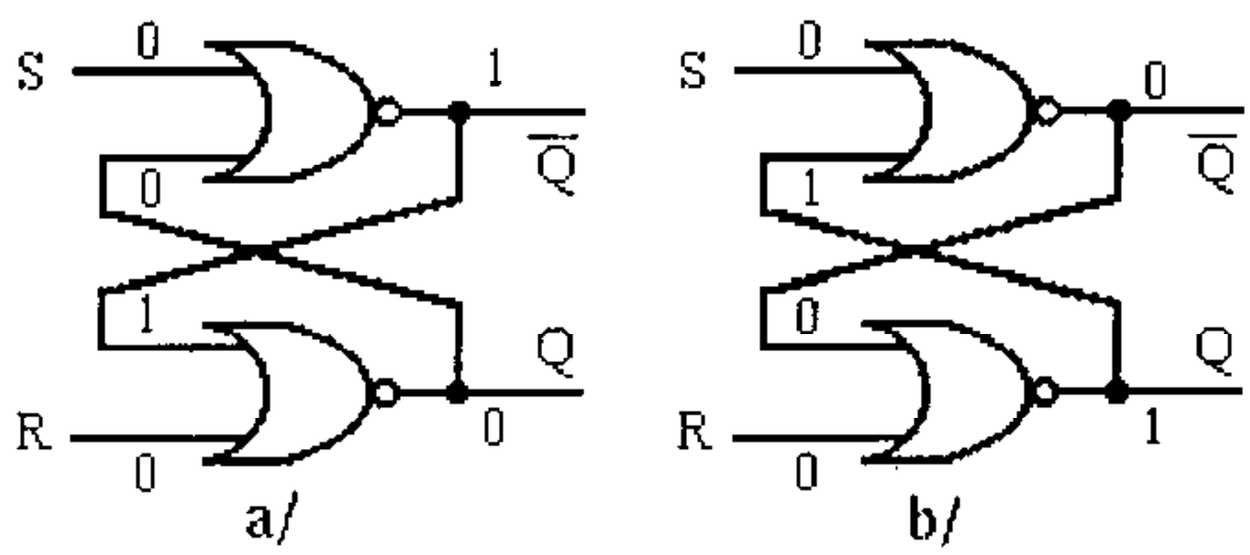
3.3.1 Phần tử nhớ 1 bit

Thanh ghi chốt (Latch) là dạng đơn giản của flip-flop, được xây dựng từ 2 cổng NAND hoặc 2 cổng NOR. Sự thay đổi trạng thái của thanh ghi chốt có thể xảy ra trong thời gian kéo dài của xung đồng hồ chứ không phải trong thời gian sườn xung đồng hồ, người ta gọi đó là sự chuyển mạch theo mức.

Các thanh ghi chốt 1 bit có thể được sử dụng làm các phần tử nhớ cơ bản xây dựng nên bộ nhớ của máy tính.

3.3.1.1 Thanh ghi chốt đơn giản nhất - thanh ghi chốt RS

Để tạo ra bộ nhớ 1-bit, ta cần một mạch điện có khả năng nhớ giá trị vừa đưa vào. Một mạch điện như vậy có thể được xây dựng từ các cổng NOR hoặc NAND, như ở hình 3-16. Nó có tên gọi là thanh ghi chốt RS (hoặc SR).



Hình 3-16 a/ Thanh ghi chốt dùng cổng NOR ở trạng thái 0; b/ Thanh ghi chốt dùng cổng NOR ở trạng thái 1

Thanh ghi chối RS có 2 lối vào là S (Set) - để thiết lập giá trị (cho Q=1), và R (Reset) - để xoá nó (cho Q=0), hai lối ra luôn bù nhau là Q và $\overline{Q}$. Khác với các mạch logic tổ hợp, giá trị lối ra của thanh ghi chốt RS không phải được quyết định duy nhất bởi các giá trị đầu vào hiện thời.

- Cho R=S=0: trong khi thanh ghi RS đang ở trong một trong hai trạng thái cân bằng ổn định (Q=1 hoặc Q=0) giá trị của Q sẽ không thay đổi. (giá trị này của R và S tồn tại trong phần lớn thời gian hoạt động của thanh ghi)
- Cho S=1, R=0 trong khi Q=0 → dẫn tới Q=1, nếu Q=1 thì Q sẽ không thay đổi.
- Cho R=1, S=0 → dẫn tới Q=0 không phụ thuộc vào trạng thái trước đó.

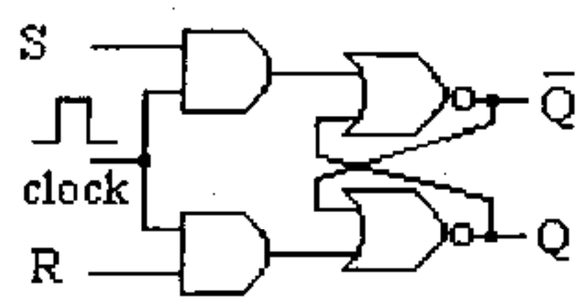
(**Tổ hợp R=S=1 bị cấm** vì khi ra khỏi trạng thái này về trạng thái R=S=0 trạng thái của mạch điện sẽ không xác định.)

Kết luận: Mạch điện nhớ được S hay R vừa có giá trị 1, tức là nhớ được tín hiệu vào (nếu ta đặt S=1, R=0 hoặc R=1, S=0). Sau đó nếu chúng ta cho R=S=0 thì giá trị nhớ sẽ không thay đổi.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

3.3.1.2 Thanh ghi chốt RS hoạt động theo nhịp xung đồng hồ

Để làm cho thanh ghi RS chỉ có thể chuyển trạng thái vào những khoảng thời gian xác định, người ta sử dụng tín hiệu đồng hồ - clock. Thanh ghi như vậy được gọi là Thanh ghi chốt RS hoạt động theo nhịp xung đồng hồ.



Hình 3-17 Thanh ghi chốt RS hoạt động theo nhịp xung đồng hồ

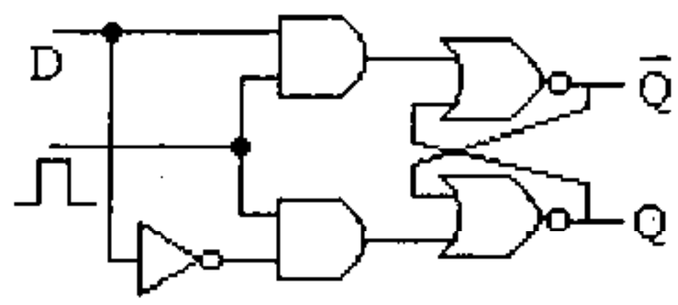
Mạch điện có đầu vào clock, bình thường clock=0, thanh ghi không thể chuyển trạng thái. Chỉ trong khoảng thời gian clock=1 thanh ghi mới phản ứng theo S và R.

Tín hiệu clock này không nhất thiết phải do đồng hồ sinh ra, nó có thể là các tín hiệu như enable - cho phép chip hoạt động.v.v.

3.3.1.3 Thanh ghi chốt D hoạt động theo nhịp xung đồng hồ

Để loại trừ tổ hợp biến vào bị cấm R=S=1, người ta xây dựng thanh ghi chốt D như hình vẽ 3-18.

- Khi D=1, clock=1 thanh ghi chuyển sang trạng thái 1 (Q=1)
- Khi D=0, clock=1 thanh ghi chuyển sang trạng thái 0 (Q=0)



Hình 3-18 Thanh ghi chốt D hoạt động theo nhịp xung đồng hồ

Phương trình trạng thái như sau: $Q_{n+1}=D_n$

Điều đó có nghĩa là tín hiệu ở lối ra Q bằng tín hiệu ở lối vào D tại xung nhịp trước.

3.3.1.4 Phần tử nhớ 1 bit của bộ nhớ RAM tĩnh (SRAM)

Các thanh ghi chốt được trình bày ở trên là các phần tử nhớ cơ bản 1 bit, nó có 2 trạng thái cân bằng ổn định được sử dụng để biểu diễn 2 giá trị nhị phân 0 và 1. Khi phần tử nhớ đã được thiết lập giá trị thì nó nhớ mãi giá trị đó cho đến khi chúng ta thiết lập cho nó một giá trị mới. Vì thế người ta gọi nó là phần tử nhớ RAM tĩnh (Static Random Access Memory). Bộ nhớ được xây dựng từ các phần tử 1 bit như vậy được gọi là bộ nhớ RAM tĩnh.

Hình 3-19 là sơ đồ mạch điện của một phần tử nhớ RAM tĩnh 1 bit S_{ij} và các mạch điện cũng như các tín hiệu để điều khiển sự hoạt động của nó.

Các đường dây truyền tín hiệu trên sơ đồ có ý nghĩa như sau:

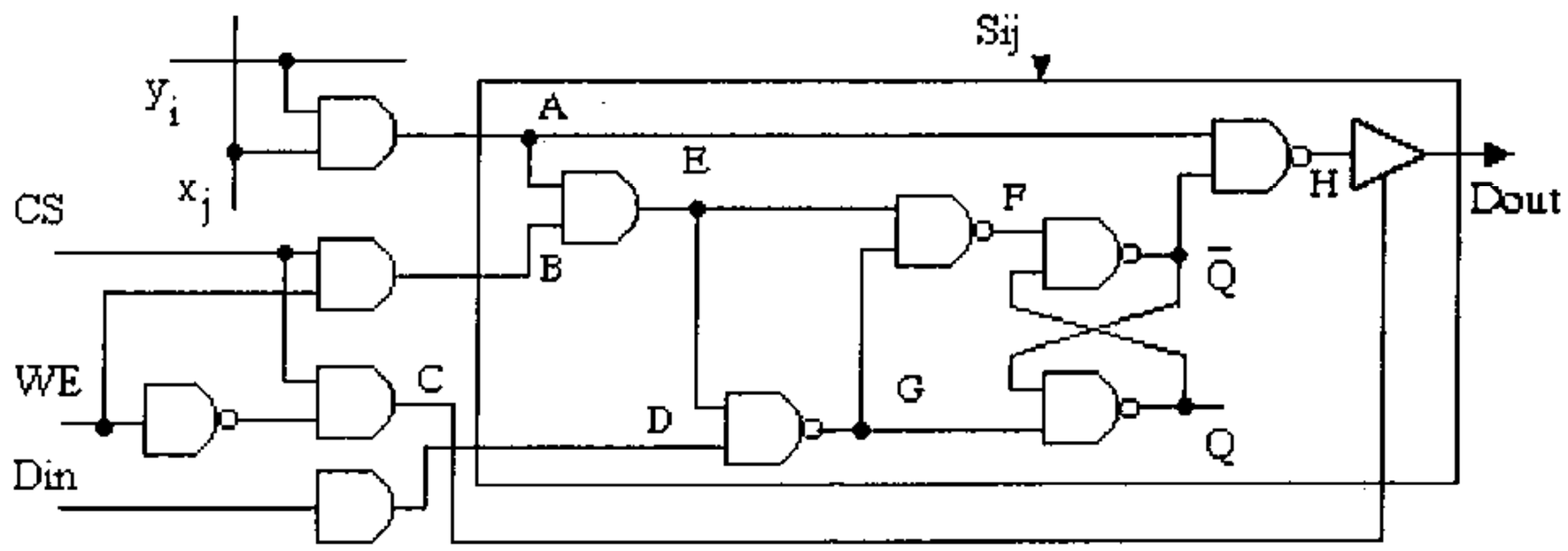
- Din: đầu vào thông tin
- Dout: đầu ra thông tin

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Yi, Xj: là các dây địa chỉ. Nếu ta tổ chức một mạng nhớ hình chữ nhật, trong đó phần tử S_{ij} nằm ở hàng i cột j thì Yi sẽ được nối với hàng i của ma trận, còn Xj sẽ được nối với cột j của ma trận.
- WE (Write Enable): tín hiệu cho phép ghi, khi We = 1: cho phép ghi thông tin Din vào phần tử nhớ, lúc này đầu ra Dout có trở kháng cao (vì nó được nối với đầu ra của bộ đệm ba trạng thái đang ở trạng thái có trở kháng cao), có thể coi Dout bị cách ly khỏi phần tử nhớ. Khi WE = 0: cho phép đọc thông tin từ phần tử nhớ, bộ đệm ba trạng thái đóng (trở kháng thấp), Dout được nối với điểm H và có giá trị bằng Q là nội dung phần tử nhớ, đồng thời E=B=0 làm cho F=G=1 dẫn đến Q không đổi trạng thái, tức là Q không bị phụ thuộc vào Din.
- CS (Chip Select): tín hiệu chọn chip, đôi khi còn được ký hiệu là CE (Chip Enable). Khi có nhiều chip nhớ RAM cùng nối với đường tín hiệu chung (Din, Dout) thì đầu vào CS có nhiệm vụ chọn xem chip RAM nào được truyền thông tin trên bus số liệu.

Điều kiện ghi: CS=1, Xi =1, Yj =1, WE=1, khi đó Q sẽ được thiết lập bằng Din.

Điều kiện đọc: CS=1, Xi =1, Yj =1, WE=0, khi đó Dout sẽ nhận giá trị bằng Q.



Hình 3-19 Mạch điện của một phần tử nhớ RAM tĩnh 1 bit S_{ij}

Khi có các phần tử nhớ 1 bit có cấu tạo như trên, rất dễ kết hợp chúng lại với nhau để tạo nên các bộ nhớ có dung lượng từ (word) và kích thước từ như mong muốn. Vấn đề này được đề cập trong mục tổ chức bộ nhớ, sẽ được trình bày sau.

3.3.1.5 Phần tử nhớ 1 bit của bộ nhớ RAM động (DRAM- Dynamic RAM)

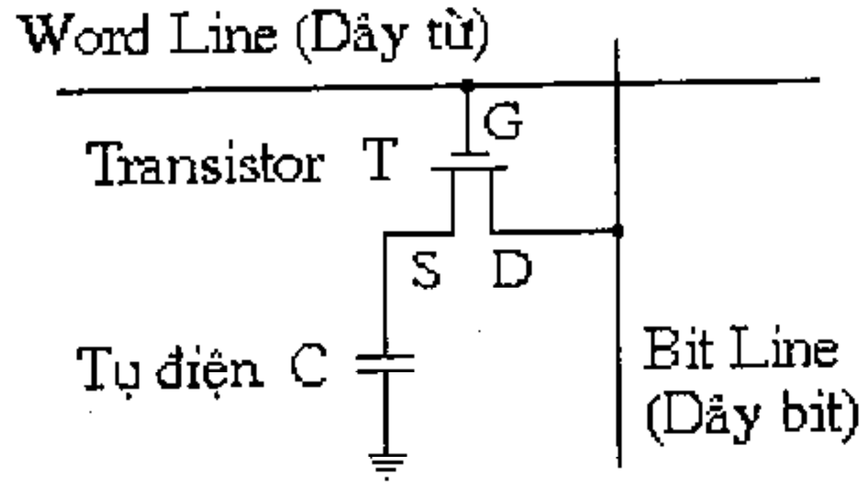
Như chúng ta đã biết, các phần tử nhớ RAM tĩnh (SRAM) được cấu tạo từ các flip-flop, một phần tử SRAM 1 bit như ở hình 3-19 được tạo bởi 6 cổng NAND và 1 bộ đệm 3 trạng thái, nếu mỗi phần tử này được tạo bởi 2 transistor thì cần đến 14 transitor để nhớ được 1 bit thông tin. Ngày nay người ta thường sử dụng các phần tử SRAM đơn giản hơn, chứa từ 4 đến 6 transistor.

Để bố trí được số phần tử nhớ lớn nhất trong một vi mạch, mỗi phần tử nhớ phải được chế tạo sao cho đơn giản nhất. Phần tử nhớ RAM động (DRAM) mà chúng ta tìm hiểu dưới đây chỉ cần 1 transistor cho 1 bit thông tin, chính vì vậy có thể bố trí

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

với mật độ rất cao và có giá thành rẻ. Trong phần tử nhớ này người ta thay flip-flop bằng một tụ điện C, giá trị nhớ trong phần tử nhớ này chính là điện tích nạp trên tụ điện. Ta có thể sử dụng trạng thái tụ được nạp, tức là trên tụ điện C có một điện áp lớn hơn một giá trị nhất định nào đó, biểu diễn giá trị 1 của bit, còn trạng thái không được nạp biểu diễn giá trị 0. Nguyên lý cấu tạo của một phần tử DRAM được minh hoạ trên hình 3-20.

Bộ nhớ DRAM được tổ chức thành một ma trận nhớ (thường là ma trận vuông), trong đó dây từ (Word Line) là một trong các dây hàng của ma trận, còn dây bit (Bit Line) là một trong các dây cột. Phần tử nhớ được đặt ở giao điểm của các dây hàng và cột. Transistor T là một transistor trường (Field Effect Transistor) đóng vai trò một chuyển mạch điện tử. T có 3 cực là cực cổng G (gate), cực máng D (Drain) và cực nguồn S (Source), trong đó G là cực điều khiển, D sẽ được nối với S khi G có mức điện áp cao (1) so với S, ngược lại thì điện trở giữa D và S sẽ rất lớn.



Hình 3-20 Phần tử nhớ DRAM 1 bit

- Việc ghi (Write)

Khi dây từ có mức tích cực (1), T ở trạng thái mở, nối tụ điện C với dây bit. Nếu thao tác là ghi thì giá trị cần ghi phải đặt trên dây bit. Nếu giá trị đó là 1 thì tụ C sẽ được nạp tới điện áp ứng với giá trị 1 trên dây bit, còn nếu giá trị đó là 0 thì tụ C sẽ bị phóng hết điện tích, tức là có giá trị 0.

- Việc đọc (Read)

Việc đọc phức tạp hơn ghi một chút do điện tích trên tụ C ứng với giá trị cần đọc rất nhỏ. Trước khi đặt dây từ lên mức tích cực, cần phải đặt lên dây bit điện áp bằng 1/2 mức chênh lệch giữa điện áp ứng với mức 0 và điện áp ứng với mức 1. Điện áp trên tụ sẽ làm cho điện áp dây bit thay đổi một chút theo chiều hướng tăng hoặc giảm, tuỳ thuộc vào việc nó đang nhớ giá trị 1 (high) hay 0 (low). Sự thay đổi nhỏ của điện áp trên dây bit sẽ được truyền tới đầu vào của một bộ khuếch đại nhạy, tại đầu ra của nó ta nhận được điện áp ứng với giá trị của bit chứa trên tụ C.

- Làm tươi (Refresh)

Vì mọi tụ điện đều có một điện trở rò và transistor T mắc nối tiếp với nó dù ở trạng thái cấm cũng có một điện trở rò nhất định, cho nên sau khi được nạp, điện tích trên tụ C liên tục bị phóng, sau một khoảng thời gian nhất định sẽ làm mất thông tin mà C chứa. Chính vì vậy cần phải nạp điện lại cho tụ C trước khi điện áp trên tụ giảm thấp hơn một ngưỡng nào đó, việc này được gọi là "làm tươi" (refresh). Để làm tươi ô nhớ DRAM, cần phải đọc nội dung của nó rồi viết trở lại. Việc làm tươi phải được tiến hành đều đặn theo một chu kỳ nhất định, gọi là chu kỳ làm tươi; Tên gọi RAM động (Dynamic RAM) xuất phát từ hoạt động này.

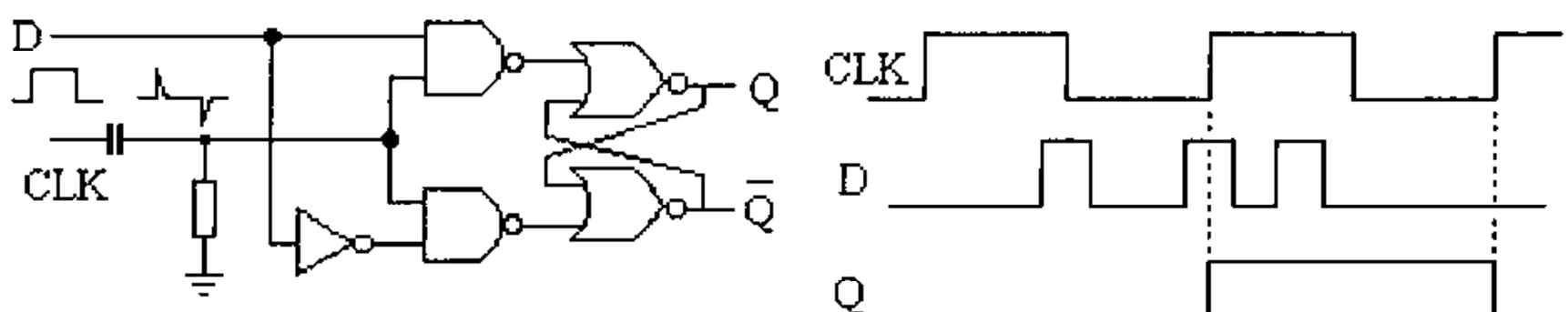
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Trong các chip DRAM trước đây mạch điện bổ sung để thực hiện "làm tươi" thường ở ngoài chip nhớ. Ngày nay các mạch thực hiện làm tươi thường được chế tạo nằm ngay bên trong chip nhớ, nhờ đó chip nhớ loại này vừa có dung lượng cao vừa có giao diện đơn giản, chúng được gọi là ***quasi-static RAM*** **(quasi-giống như).**

3.3.2 Flip-flop và thanh ghi

3.3.2.1 Flip-flop D

Trong nhiều mạch điện thường cần phải lấy mẫu giá trị tín hiệu trên một đường dây nào đó tại một thời điểm cụ thể và ghi nhớ giá trị đó. Về nguyên tắc có thể sử dụng thanh ghi chốt D có khả năng trong khoảng thời gian ứng với xung đồng hồ rất ngắn trên lối vào clock, ghi nhận được giá trị ở lối vào D. Người ta đã phát triển một biến thể của thanh ghi chốt D - gọi là thanh ghi chốt D kích bởi sườn xung (Edge-Triggered D-Flip-flop) và đôi khi vẫn được gọi là flip-flop D; Sự chuyển trạng thái của nó xảy ra trong quá trình tín hiệu đồng hồ chuyển từ 0 lên 1 hoặc ngược lại (chuyển mạch tại sườn xung). Như vậy thời gian kéo dài xung không quan trọng, chỉ cần sự chuyển trạng thái xảy ra đủ nhanh. (Xem Hình 3-21)

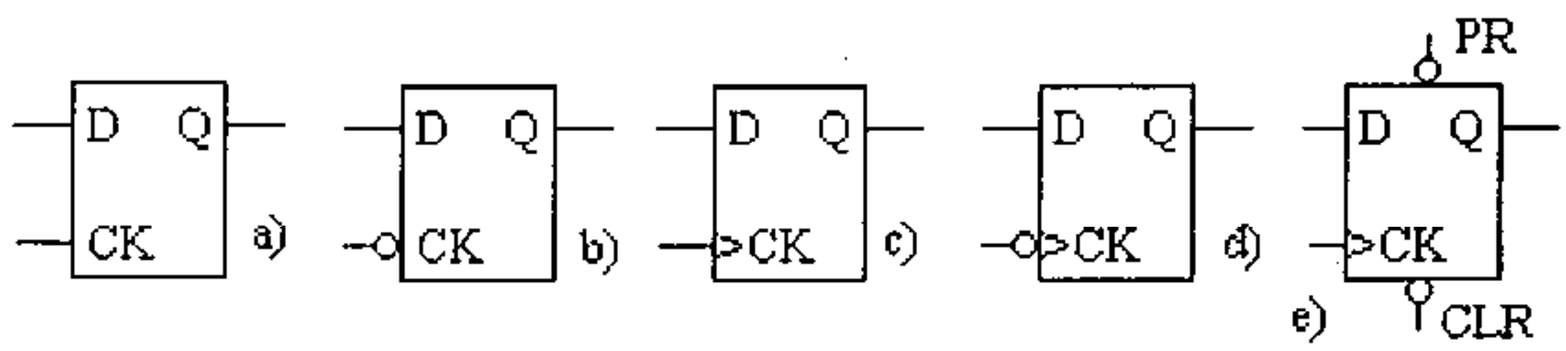


Edge-triggered D-Flip-flop

Hình 3-21 Flip-flop D kích thích bởi sườn xung

Hình 3-22 trình bày ký hiệu chuẩn của một số thanh ghi chốt và flip-flop.

- Hình (a) là thanh ghi chốt D, chuyển trạng thái khi tín hiệu CK=1, bình thường CK=0. (chuyển mạch theo mức.)
- Hình (b) là thanh ghi chốt D, chuyển trạng thái khi tín hiệu CK=0, bình thường CK=1. (chuyển mạch theo mức.)



Hình 3-22 Ký hiệu chuẩn của một số thanh ghi chốt và flip-flop

- Hình (c) và (d) là các Flip-flop chứ không phải các thanh ghi chốt, ở lối vào clock của chúng được vẽ ký hiệu đầu mũi tên '>'. Flip-flop trên hình (c) chuyển

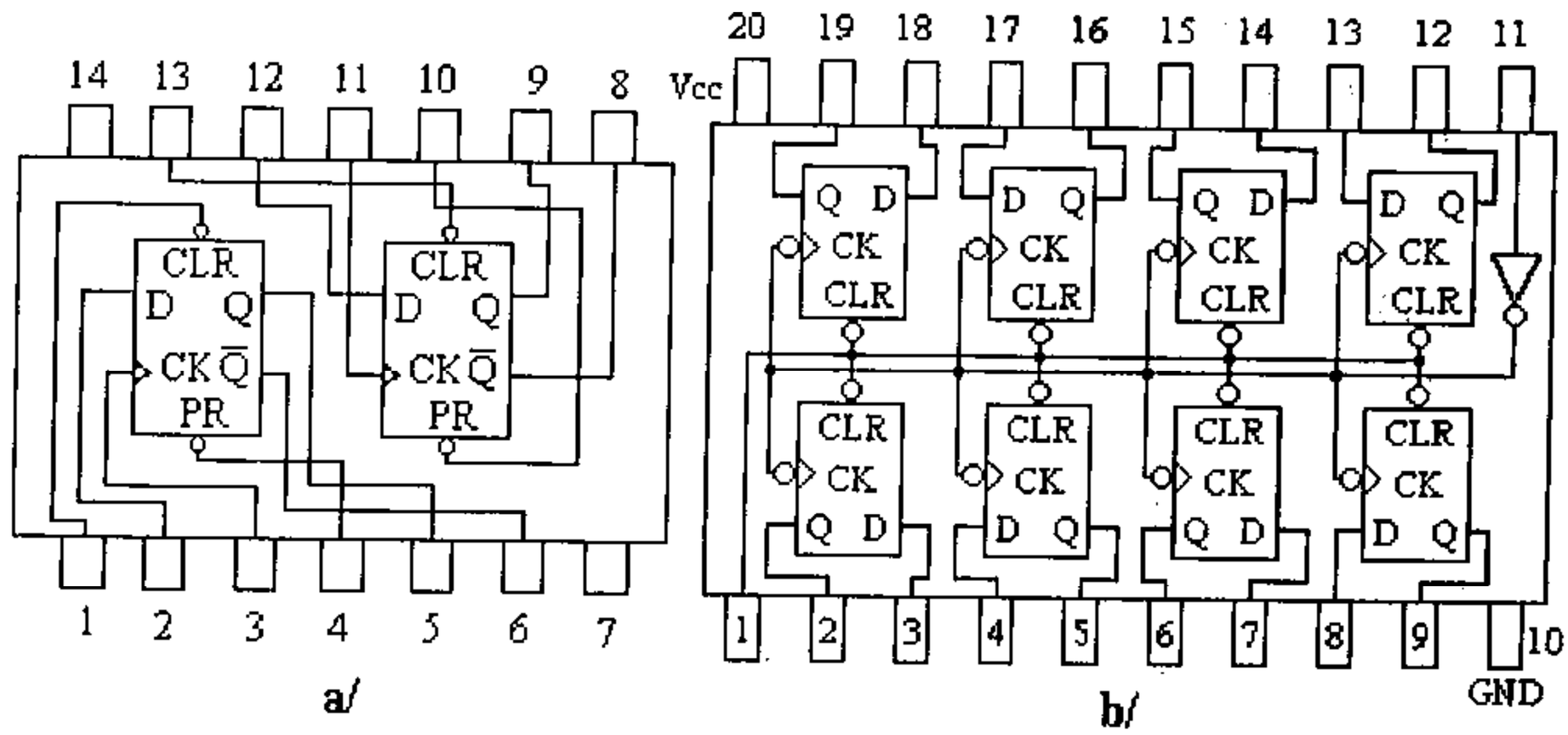
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

trạng thái trong thời gian sườn dương của xung đồng hồ. Flip-flop trên hình (d) chuyển trạng thái trong thời gian sườn âm của xung đồng hồ.

- Hình (e): Nhiều thanh ghi chốt và flip-flop ngoài đầu ra Q còn có đầu ra $\overline{Q}$ và có thêm các đầu vào Set hoặc Preset (để thiết lập Q=1) và Reset hay Clear (để thiết lập Q=0)

Người ta sản xuất ra flip-flop với nhiều cấu hình khác nhau. Một loại đơn giản, trong một chip có chứa 2 flip-flop D độc lập với nhau, có các chân tín hiệu clear và preset được trình bày trên hình 3-23a. Mặc dù 2 flip-flop này được đóng trong cùng một chip 14 chân nhưng chúng không có liên hệ gì với nhau. Một loại chip khác chứa 8 flip-flop D, như trên hình 3-23b. Các flip-flop này không có các đường tín hiệu $\overline{Q}$ và đường reset, đồng thời tất cả các đường tín hiệu đồng hồ được nối lại với nhau và được điều khiển từ chân 11. Các flip-flop này đều có kiểu như trên hình 3-22d nhưng chân 11 của chip lại có một mạch đảo cho nên các flip-flop sẽ được nạp dữ liệu khi tại chân 11 có bước chuyển mức dương của xung đồng hồ (sườn dương). Tất cả các đầu vào xoá (CLR) của cả 8 flip-flop đều nối với chân 1 của chip, vì vậy khi chân này có mức 0 thì các flip-flop đều bị chuyển vào trạng thái 0.

Sở dĩ người ta nối 8 chân xoá CLR với nhau và 8 chân đồng hồ CK với nhau vì người ta dự định dùng cả 8 flip-flop trong chip như một đơn vị - làm thành 1 thanh ghi 8 bit, việc nối bên trong này sẽ giảm được số chân của chip.



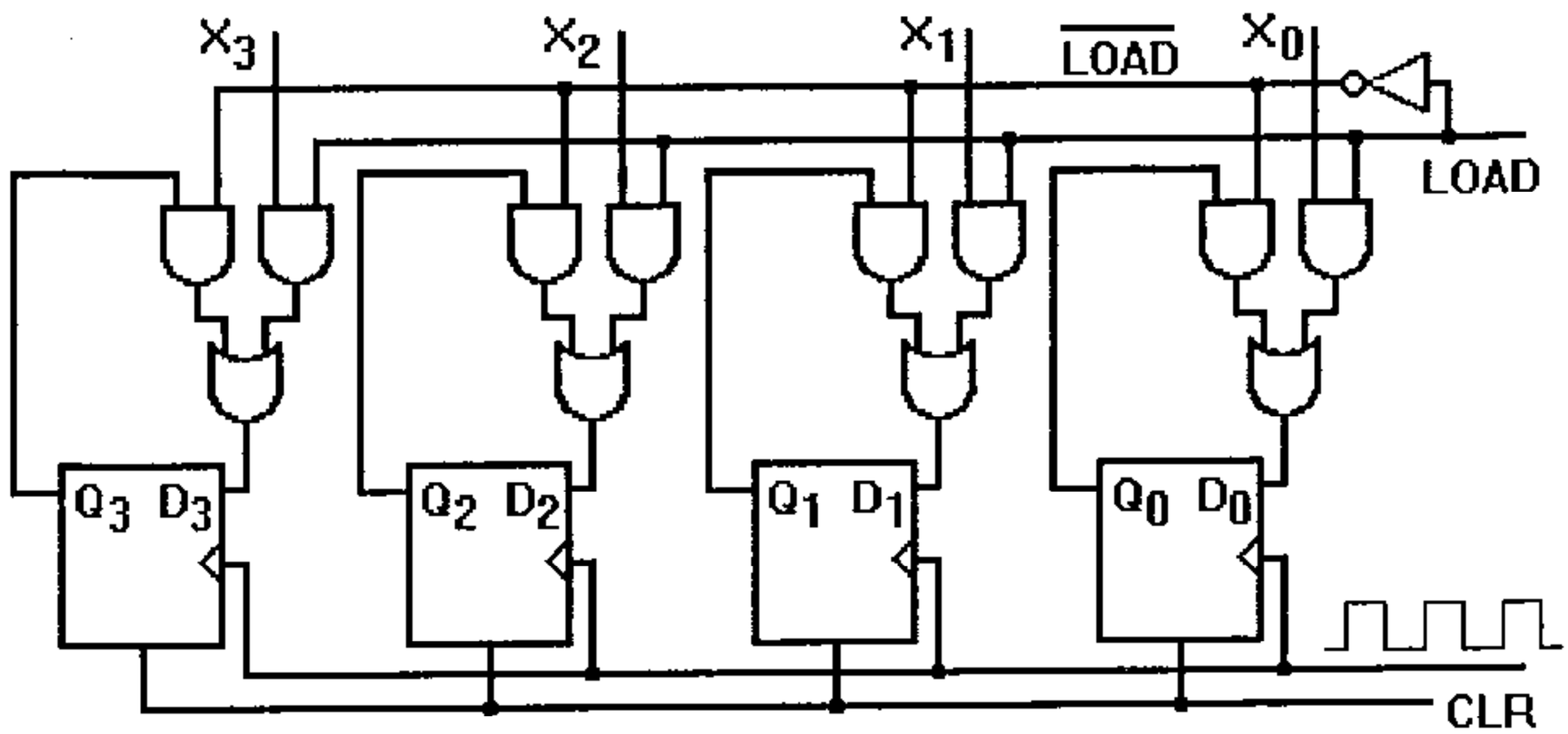
Hình 3-23 a/ Một chip chứa 2 flip-flop D độc lập. b/ Một loại chip khác chứa 8 flip-flop D có thể dùng làm một thanh ghi 8 bit.

3.3.2.2 Thanh ghi - Register

Thanh ghi là một nhóm các phần tử nhớ cơ bản cùng hoạt động như một đơn vị. Có những thanh ghi chỉ làm nhiệm vụ đơn giản là nhớ một từ nhị phân; có những thanh ghi thực hiện các thao tác tính toán số học - thay đổi từ nhị phân mà nó nhớ bằng cách dịch trái hoặc dịch phải các bit của từ và các thao tác khác phức tạp hơn nữa.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Các thanh ghi làm nhiệm vụ nhớ tạm thời một từ nhị phân thường được xây dựng từ các flip-flop. Các bộ xử lý thường có một tập thanh ghi được sử dụng để chứa tạm (nhớ đệm) dữ liệu hoặc các chỉ thị trong quá trình thi hành chương trình. Các thanh ghi như vậy cần có khả năng hoạt động ở tốc độ cao hơn các thanh ghi được sử dụng trong bộ nhớ chính. Trên hình 3-24 là thanh ghi đệm 4-bit, sử dụng flip-flop D chuyển mạch bởi sườn dương của xung đồng hồ.



Hình 3-24 Thanh ghi đệm

- Xi (i=1..3) là các bit của word X cần nhớ, khi sườn xung dương đầu tiên đi tới, word được nhớ vào trong thanh ghi là $Q_3Q_2Q_1Q_0 = X_3X_2X_1X_0$, hay viết là Q = X. Từ nhị phân có thể lấy ra ở các đầu ra $Q_3Q_2Q_1Q_0$ (trên hình không vẽ đầu đưa tín hiệu ra).
- CLR (Clear) là đầu vào tín hiệu điều khiển xoá nội dung thanh ghi, tích cực ở mức cao, nghĩa là khi CLR là cao (1), tất cả các flip-flop bị xoá và từ được nhớ trở thành Q = 0.
- LOAD là đầu vào điều khiển, tích cực ở mức cao, khi LOAD = 0 (low) các bit vào X không thể đi vào các flip-flop, đồng thời trên đường dây $\overline{LOAD}$ sẽ có mức = 1 (high), làm cho các giá trị ở đầu ra của các flip-flop đưa ngược trở lại lối vào, tại các sườn dương của xung đồng hồ chúng sẽ lại được ghi lại vào trong flip-flop, nói cách khác, khi LOAD=low thì thanh ghi không thay đổi nội dung.
- Khi LOAD = 1 (high) các bit của X được đưa tới lối vào của flip-flop, khi sườn dương của xung đồng hồ đi tới chúng sẽ được ghi vào trong thanh ghi.

3.3.3 Bộ nhớ ROM

ROM (Read Only Memory) là bộ nhớ giữ thông tin cố định, không thay đổi nội dung ngay cả khi ngắt nguồn điện nuôi, nói chung có thể chia ROM thành 3 loại chính sau:

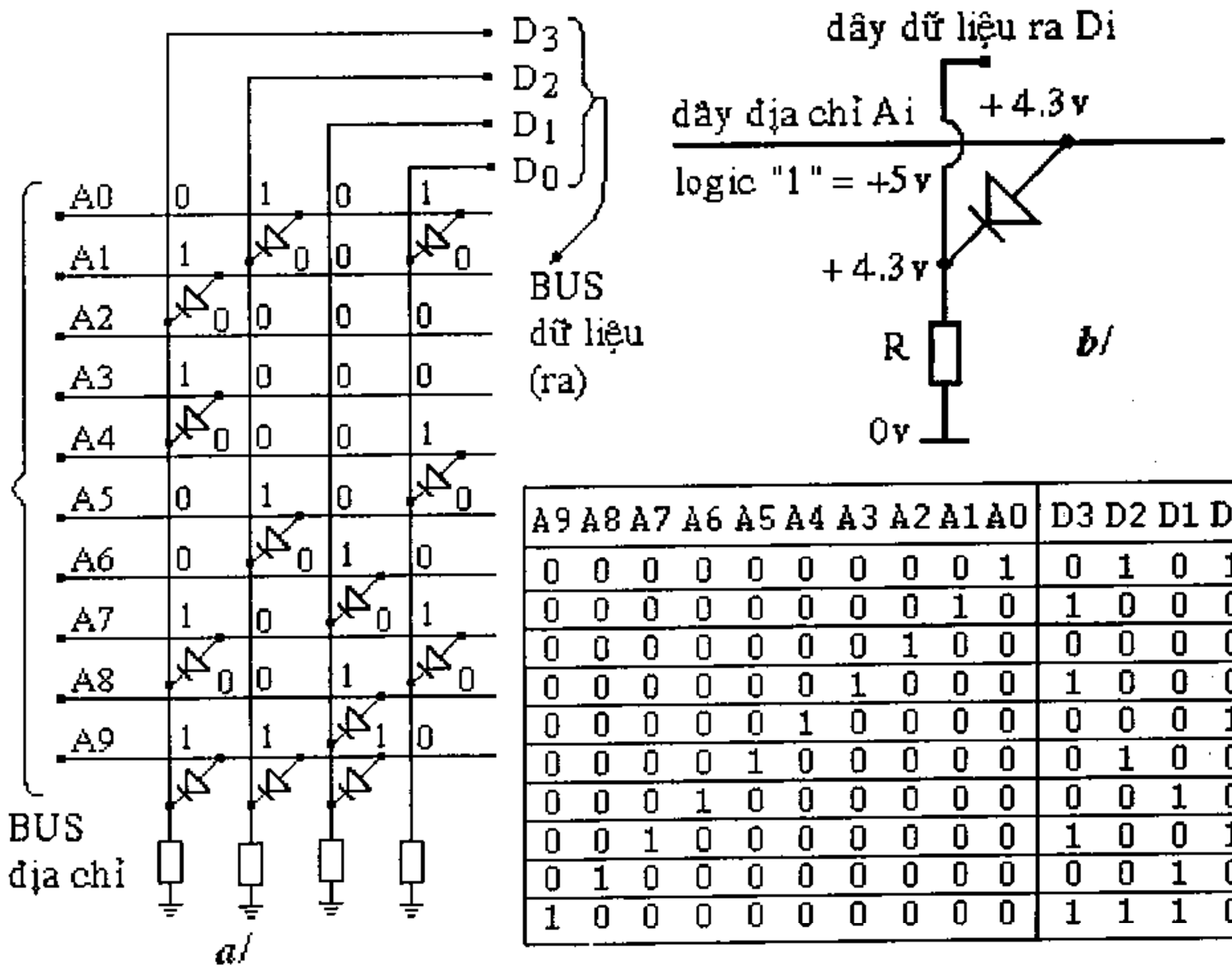
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **ROM mặt nạ (Maskable ROM)**, nội dung của nó được cố định trong quá trình chế tạo vi mạch và không thể thay đổi được sau đó.
- **ROM có thể nạp chương trình (PROM - Programmable ROM)**, là loại ROM mà người sử dụng có thể sử dụng một thiết bị chuyên dụng để nạp nội dung cho nó, việc này thường được gọi là "đốt ROM". Sau khi đã được nạp chương trình, nội dung của PROM cũng không thể thay đổi được nữa.
- **ROM có thể xoá (Erasable ROM)**, là loại ROM mà người sử dụng có thể nạp nội dung cho nó bằng thiết bị chuyên dụng, tuy nhiên người sử dụng sau này có thể xoá nội dung cũ để ghi nội dung mới; Loại ROM này thường là EPROM, EAROM.

3.3.3.1 ROM mặt nạ

ROM mặt nạ thường được tổ chức dưới dạng ma trận như minh hoạ trên hình 3-25a. Trạng thái có hay không có đi-ốt tại vị trí (i,j) của ma trận được sử dụng để biểu diễn một bit dữ liệu là 1 hay 0. Mỗi hàng i của ma trận đi-ôt giống như một thanh ghi lưu trữ một từ (word), các dây địa chỉ Ai được sử dụng để chọn lấy một từ, đưa nội dung của nó ra ngoài qua các dây dữ liệu ra $D_3D_2D_1D_0$.

Hình 3-25b là một "ô nhớ" của ma trận đi-ốt. Hình 3-25c là bảng chân lý của ma trận ROM đi-ốt ở hình 3-25a.



A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	1	0	1	0	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1	1	1	0

Hình 3-25 ROM mặt nạ cấu tạo bởi ma trận đi-ốt

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Việc ghi chương trình cho ROM được thực hiện bằng cách loại bỏ hoặc để lại đi-ốt ở giao điểm của các dây hàng và dây cột của ma trận.

- Nếu đi-ốt tại hàng i, cột j được giữ lại, khi mức điện áp trên dây địa chỉ là 1 (cao), đi-ốt được phân cực thuận sẽ có điện trở rất nhỏ, vì vậy Dj=Ai=1.
- Nếu đi-ốt tại hàng i, cột j đã bị loại bỏ, dù mức điện áp trên dây địa chỉ là cao, nó cũng không ảnh hưởng tới dây dữ liệu ra Dj lúc này được nối "đất" (low,0) qua một điện trở.

Các công nghệ MOS và công nghệ lưỡng cực đều được sử dụng để chế tạo ROM; Thay cho mỗi đi-ốt người ta có thể dùng một transistor lưỡng cực (Bi-polar transistor) hay một transistor đơn cực (MOSFET), tuy vậy nguyên tắc cũng không có gì thay đổi. Thời gian truy cập giữa 2 loại ROM lưỡng cực và MOS là khác nhau, ROM lưỡng cực (Bi-polar ROM) là: 50-90ns, nó tác động nhanh hơn, khả năng kích tốt hơn. Đối với ROM theo công nghệ MOS (MOS ROM) thời gian tác động lớn gấp 10 lần (chậm hơn), nhưng kích thước nhỏ hơn, tiêu thụ ít năng lượng hơn.

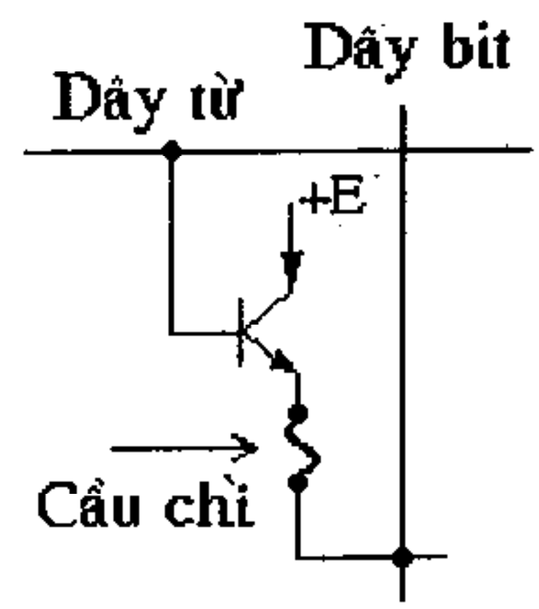
3.3.3.2 PROM - ROM có thể nạp chương trình

Nguyên tắc cấu tạo của PROM có điểm giống ROM mặt nạ ở chỗ các phần tử nhớ được bố trí ở giao điểm của các dây hàng và dây cột của ma trận, giống như cái công tắc, nó có hai trạng thái là đóng (nối) và mở (cắt). Một cầu chì sẽ đóng vai trò của công tắc, nó cũng có hai trạng thái là đứt và không đứt.

Khác với ROM mặt nạ, các chip PROM khi xuất xưởng chưa được ghi nội dung, tất cả các cầu chì đều chưa đứt, người sử dụng có thể tự ghi nội dung cho nó bằng một thiết bị chuyên dụng, làm "cháy" cầu chì ở các vị trí mong muốn của ma trận.

Mô tả bộ nhớ PROM nối bằng cầu chì :

Phần tử nối giữa dây hàng (dây từ) và dây cột (dây bit) là 1 transistor mắc theo kiểu lặp lại Emiter, có cầu chì mắc nối tiếp với Emiter. Cầu chì thường được làm bằng Nichrome hay silic đa tinh thể (Poli-Crystalline Silicon), các vật liệu này có nhiệt độ nóng chảy thấp. Khi muốn ghi vào từ nào thì nâng dây từ tương ứng lên mức cao (1), đồng thời đặt các dây bit lên các mức như sau:



Hình 3-26 Phần tử nhớ PROM

- Với các bit cần ghi giá trị 0, đặt dây bit tương ứng ở mức thấp (0), do đó các transistor có cực Base nối với dây từ đang có mức cao (1) và cực Emitter nối với dây bit qua cầu chì có mức thấp (0) sẽ thông, cho một dòng điện đủ lớn đi qua làm chảy cầu chì → phần tử nhớ tương ứng được ghi giá trị 0.
- Với các bit cần ghi giá trị 1, đặt dây bit tương ứng ở mức cao (1), do đó các transistor có cực Base nối với dây từ đang có mức cao và cực Emitter nối với dây

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

bit qua cầu chì cũng có mức cao sẽ bị cấm, không cho dòng điện đi qua và cầu chì sẽ không bị chảy → phần tử nhớ tương ứng được ghi giá trị 1.

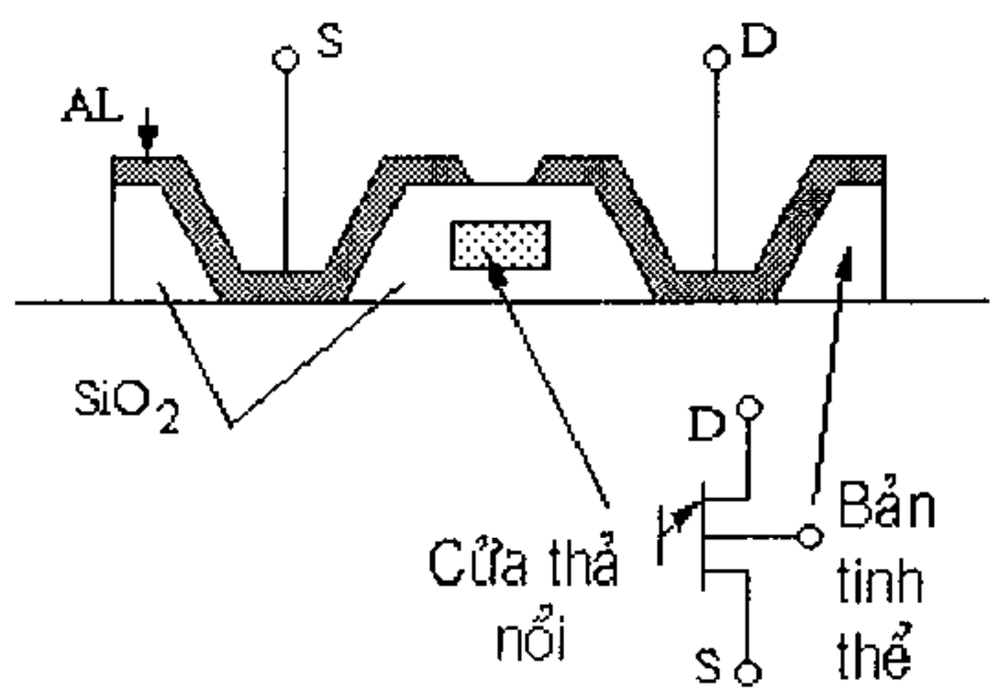
Theo nguyên tắc này mỗi chip PROM chỉ có thể ghi được một lần.

Thiết bị ghi PROM (PROM Programmer hay PROM Writer)

Việc ghi vào PROM đòi hỏi những mạch cung cấp dòng điện đủ lớn để đốt cháy cầu chì ở những bit mong muốn. Các bộ ghi PROM thường là hệ vi xử lý chuyên dụng. Đó là thiết bị tạo ra địa chỉ, số liệu và xung dòng điện ghi cần thiết để làm chảy các cầu chì. Mỗi bộ ghi PROM có thể ghi một hay một họ PROM, có những bộ ghi PROM vạn năng có thể ghi bất kỳ loại PROM nào bằng cách sử dụng một mô-đun riêng. Mô-đun này giúp cho bộ ghi PROM thích hợp ở mức xung ghi, dòng điện và bố trí chân với 1 mạch PROM cụ thể.

Giá thành ghi PROM đắt hơn ghi ROM nên PROM chỉ thuận lợi cho những ứng dụng với số lượng nhỏ.

3.3.3.3 EPROM - ROM có thể xoá và ghi lại



Hình 3-27 Nguyên lý cấu tạo của một phần tử nhớ EPROM

EPROM là một loại bộ nhớ ROM có các phần tử nhớ có thể xoá và sau đó ghi lại nội dung mới.

Bộ nhớ EPROM có cửa thả nổi lần đầu tiên được hãng INTEL giới thiệu vào năm 1971, ký hiệu 1702 có dung lượng 256x8, theo công nghệ PMOS, có 24 chân. Phần tử nhớ của EPROM 1702 là 1 transistor MOS gồm 2 vùng dẫn điện loại P trên miền n. Ba cực của transistor là:

- 1 cực gốc thả nổi
- 1 cực nguồn: S
- 1 cực góp: D

Để ghi mức logic 1 vào ô nhớ, một xung -35V được đặt giữa cực nguồn S và cực góp D được nối với vỏ. Do nguồn năng lượng lớn, các điện tử bề mặt của lớp tiếp xúc P-N sẽ chui vào cửa Silic. Số điện tử lọt vào đó là hàm số của độ lớn và độ rộng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

của xung đặt vào. Khi ngừng tác dụng xung điện áp đó, điện tích được nạp vào cửa silic không có đường dẫn ra, do đó nó được giữ lại ở đó.

Ước tính ở nhiệt độ 125°C, 80% điện tích sẽ được giữ lại sau 100 năm. Do cửa silic không có chân nối ra ngoài, nên không thể khử điện tích lưu trong cửa bằng xung điện, muốn làm việc này phải chiếu tia cực tím có bước sóng λ=0,537 antron làm sinh ra dòng quang điện chạy từ cửa thả nổi vào bản tinh thể Silic.

Nắp che mạng nhớ trên vi mạch là 1 miếng thạch anh trong suốt cố định với vỏ nhưng vẫn cho phép xoá dễ dàng. Chỉ có thể xoá toàn bộ 1 vi mạch EPROM luôn một lần, chứ không thể xoá riêng một phần tử hoặc một phần của vi mạch EPROM. EPROM có thể xoá và ghi lại nhiều lần, giá thành cao hơn PROM. Để xoá nội dung cũ, ghi lại nội dung mới vào EPROM, cần phải tháo nó ra khỏi mạch điện, xoá bằng đèn phát tia cực tím (tử ngoại).

- **EAROM**

EAROM (Electrically Alterable ROM) hay còn gọi EEPROM (Electrically Erasable ROM) có cấu tạo tương tự EPROM nhưng có thể xoá được bằng xung điện. Loại EAROM ER3400 của hãng Genaral Instruments có dung lượng 1024 x 4 có thể giữ nội dung nhớ đến 10 năm, không cần nguồn nuôi. Thời gian ghi là 1ms, thời gian xoá 10ms và thời gian đọc là 900ns.

Một nhược điểm của mạch EAROM là cần các mức điện áp nguồn nuôi khác nhau trong khi xoá, ghi và đọc. Thí dụ mạch ER3400 cần 3 mức điện áp nuôi là +12v, -5v và -30v.

3.3.4 Tổ chức bộ nhớ

3.3.4.1 Tổ chức bên trong các chip nhớ nhằm làm giảm số chân

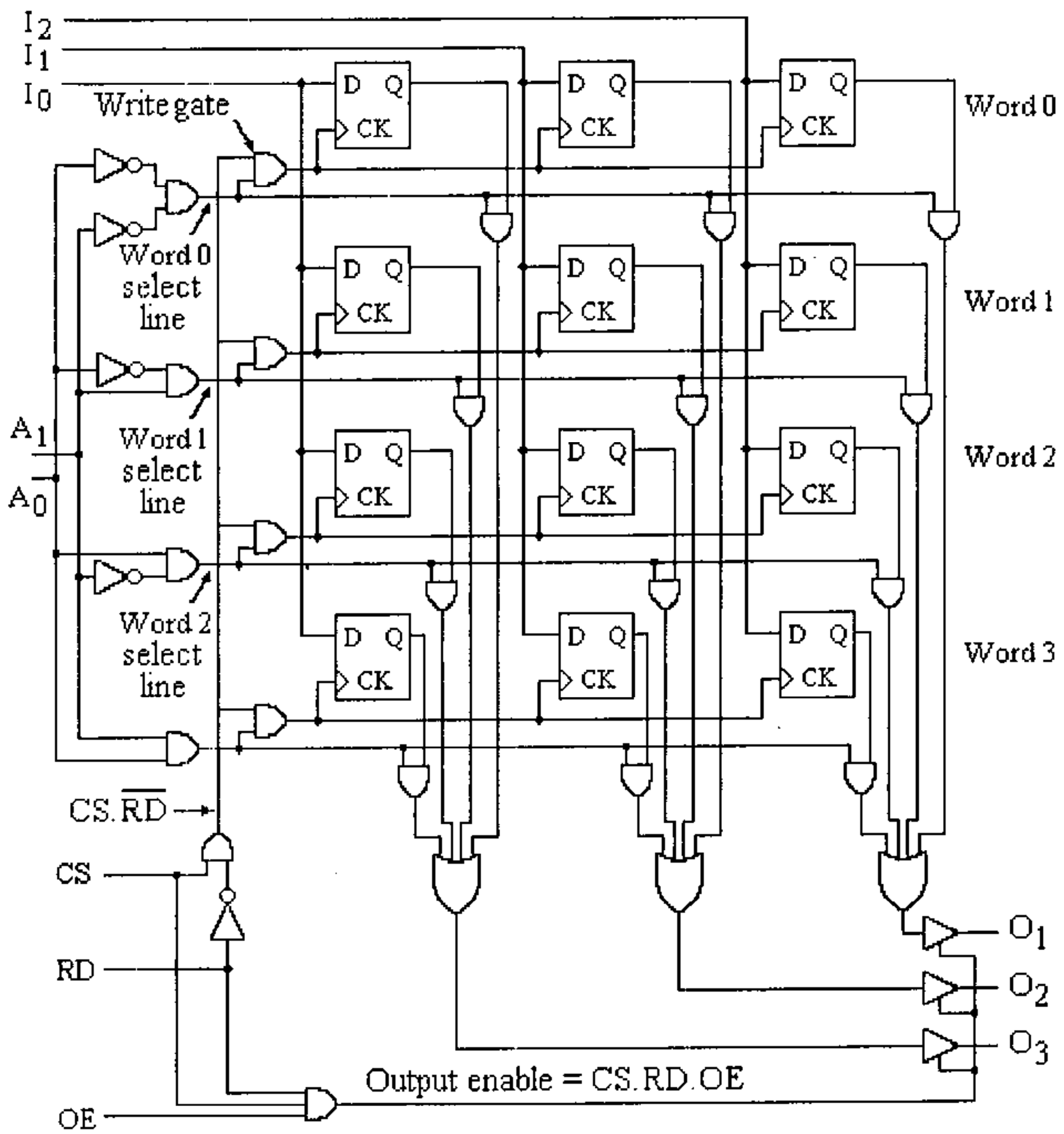
Chúng ta mong muốn có một thiết kế trong đó số lượng chân tăng lên theo logarit chứ không tăng tuyến tính theo dung lượng của chip nhớ. Có một cách tổ chức bộ nhớ đáp ứng được yêu cầu đó được sử dụng rộng rãi, như minh hoạ trên hình 3-28. Đó là một bộ nhớ có 4 từ, mỗi từ 3 bit. Trong khi tổng dung lượng của bộ nhớ là 12 bit, số chân của nó lại ít hơn số chân của chip nhớ 8 bit trên hình 3-23b và điều quan trọng nhất là thiết kế này có thể mở rộng dễ dàng cho các bộ nhớ lớn.

Bộ nhớ có 8 đường dây tín hiệu vào và 3 đường dây tín hiệu ra. Ba đường vào cho dữ liệu là I_0, I_1, I_2; hai đường vào địa chỉ là A_0, A_1 và 3 đường cho tín hiệu điều khiển là: CS (Chip Select) để chọn chip, RD để phân biệt giữa đọc (Read) và ghi (Write) và OE (Output Enable) để đưa dữ liệu ra. Ba đường ra cho dữ liệu là D_0, D_1 và D_2.

- **Tín hiệu chọn chip CS**

Bộ nhớ của máy tính có thể bao gồm nhiều chip như trên, để chọn chip nhớ nào, mạch logic bên ngoài phải đặt cho đầu tín hiệu CS của chip đó có mức = 1 và RD = 1 để đọc hoặc RD = 0 để ghi.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 3-28 Sơ đồ logic một chip nhớ 4x3, mỗi hàng là một từ 3 bit

- **Tín hiệu địa chỉ**

Hai đường dây địa chỉ phải được thiết lập giá trị để chỉ ra một trong 4 từ 3 bit sẽ được đọc hoặc ghi. Đối với thao tác đọc, các đường dây dữ liệu vào không được dùng đến, từ được chọn được đặt lên các đường dây dữ liệu ra. Đối với thao tác ghi, các bit có mặt trên các đường dây dữ liệu vào được nạp vào từ bộ nhớ đã chọn, còn các đường dây dữ liệu ra không được sử dụng.

Bốn cổng AND cùng với 4 cổng NOT (mà các đầu vào nối với các dây địa chỉ A_1A_0) tạo thành một bộ giải mã dùng để chọn từ, mỗi đầu ra của một cổng AND điều khiển một dây chọn từ (word select line), được đánh số từ 0..3.

- **Việc ghi**

Khi chip nhớ được chọn để ghi, đường dây được ghi nhãn CS.$\overline{RD}$ sẽ có mức cao, làm cho đầu ra của một trong 4 cổng AND có ghi nhãn Write gate ở mức tích cực, đó là cổng AND có một đầu vào nối với dây word select có mức cao. Tín hiệu này

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

nối với tất cả các đầu vào CK của từ đã được chọn, làm cho dữ liệu trên các đường dây dữ liệu vào I_0, I_1, I_2 nạp vào các flip-flop của từ đó.

- **Việc đọc**

Việc đọc cũng tương tự việc ghi. Việc giải mã địa chỉ để ghi và đọc hoàn toàn giống nhau. Điểm khác nhau là tín hiệu CS. $\overline{RD}$ phải có mức thấp, vì vậy các cổng AND có nhãn Write gate đều có lối ra ở mức thấp, làm cho tín hiệu ở các chân CK của các flip-flop có mức thấp, do đó các flip-flop không thể thay đổi trạng thái. Một trong bốn đường dây chọn từ (Word select line) có mức cao, làm cho tại 3 đầu ra của 3 cổng AND mà mỗi cổng có một đầu vào nối với nó có giá trị bằng các giá trị tại đầu ra Q của 3 flip-flop. Như vậy một từ đã đưa được nội dung ra. Mỗi bit dữ liệu của từ này được truyền tới một đầu vào trong số 4 đầu vào của một cổng OR ở phía dưới của hình vẽ. Ba đầu vào còn lại của mỗi cổng OR đều có giá trị 0, vì vậy giá trị ở đầu ra của các cổng OR chính là từ mà chúng ta cần đọc.

Phương pháp giải mã

- **Giải mã một bước**

Để lựa chọn một từ nằm ở hàng i nào đó của bộ nhớ trên (để ghi hoặc đọc) ta cần đặt vào các dây địa chỉ A_1A_0 một giá trị địa chỉ 2 bit chính bằng i. Bộ giải mã địa chỉ chỉ cần một bước là thực hiện được việc chọn lấy 1 từ. Phương pháp tổ chức bộ nhớ trong đó tất cả các phần tử nhớ trên một hàng được gắn với một địa chỉ và được thao tác đọc/ghi đồng thời được gọi là ***Tổ chức bộ nhớ theo kiểu giải mã một bước*** hoặc ***Tổ chức bộ nhớ theo từ với sự chọn tuyến tính.***

Ưu điểm: có thời gian truy cập ngắn, áp dụng thuận lợi cho các bộ nhớ có dung lượng nhỏ.

Nhược điểm: khi dung lượng bộ nhớ lớn thì kích thước bộ giải mã (số chân ra) cũng tăng theo. Thí dụ nếu cần tổ chức bộ nhớ có 1024 từ thì bộ giải mã cần 10 dây địa chỉ vào và sẽ có 1024 đầu ra để chọn hàng.

- **Giải mã hai bước**

Kích thước của bộ giải mã địa chỉ có thể được giảm đi bằng cách tổ chức ma trận nhớ và mạch logic chọn từ cho phép chọn trùng hợp (coincident selection) hay còn gọi là giải mã hai bước. Trong bộ nhớ tổ chức theo kiểu này, các ô nhớ được sắp xếp thành một ma trận vuông, các bit trên một hàng được gọi là một từ vật lý, mỗi từ vật lý bao gồm một số từ logic, mỗi từ logic bao gồm một nhóm bit của một từ vật lý được nhận biết và gửi ra cùng một lúc.

Trong mạng nhớ sử dụng giải mã 2 bước, bước một để chọn lấy từ vật lý, bước hai để chọn lấy từ logic. Bộ giải mã hàng để chọn một từ vật lý và bộ giải mã cột gồm một vài bộ dồn kênh, chọn lấy một từ logic trong từ vật lý đã chọn.

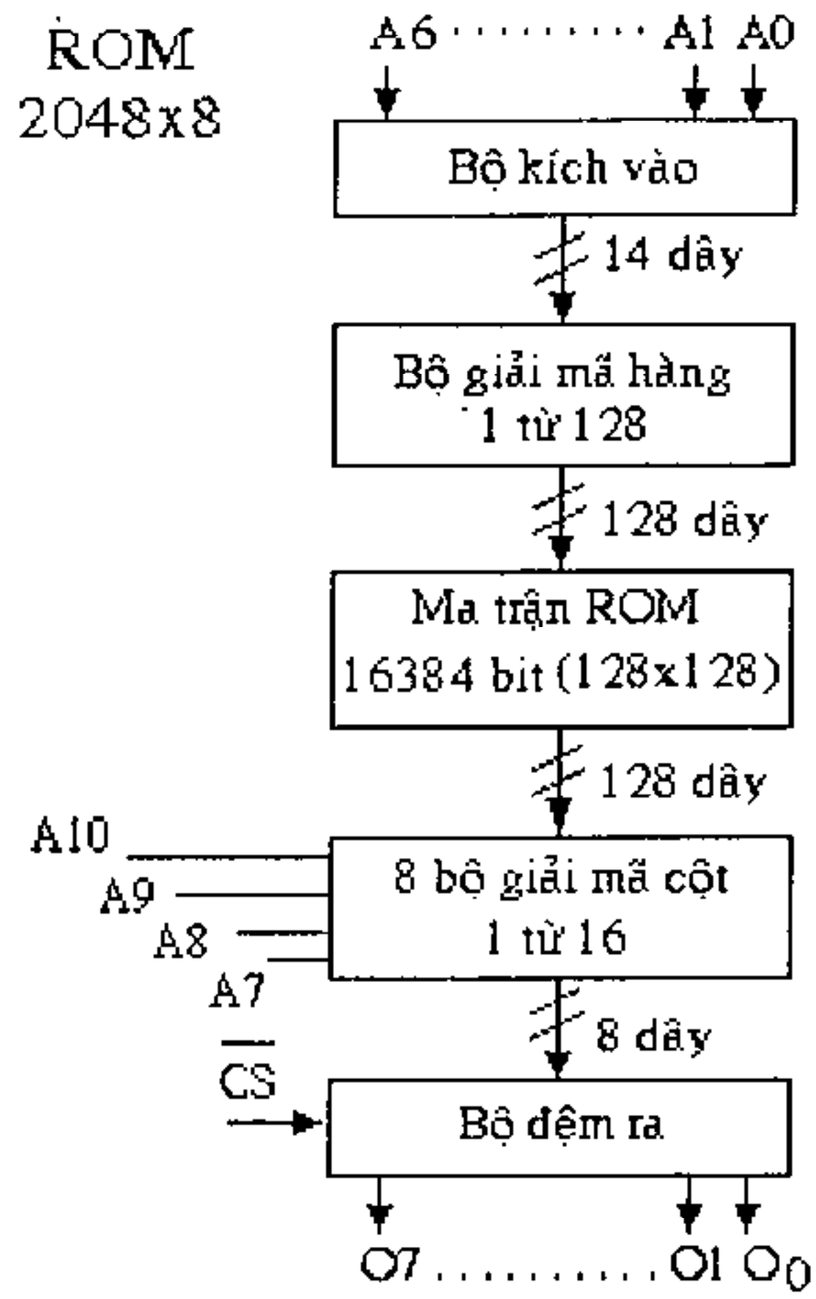
Hình 3-29 là một bộ nhớ ROM 2048x8 (2048 từ x 8bit/1 từ). Bộ nhớ này là một ma trận vuông 128x128, như vậy nó chứa 128 từ vật lý, mỗt từ vật lý được chọn bởi 7 dây địa chỉ hàng A0 - A6 (2^7 =128), bộ giải mã hàng chọn 1 hàng từ 128 hàng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Từ vật lý được chia thành 8 nhóm 16 bit. Nhóm thứ nhất chứa những bit cao nhất của 16 từ logic (bit 7), nhóm thứ hai chứa những bit cao tiếp theo (bit 6) của 16 từ logic, v.v. nhóm cuối cùng chứa những bit thấp nhất (bit 0) của 16 từ logic. Như vậy những bộ giải mã cột gồm 8 bộ dồn kênh (multiplexer) 16 đường vào 1 đường ra để cung cấp 1 từ logic 8 bit đưa ra lối ra. Các bit địa chỉ A7, A8, A9, A10 điều khiển những bộ giải mã cột.

- **Phương pháp địa chỉ multiplex**

Trong các ma trận nhớ có số dây hàng và dây cột lớn người ta thường truyền lần lượt địa chỉ hàng và địa chỉ cột trên cùng một số dây, nhờ đó giảm được một nửa số dây địa chỉ. Tuy nhiên cần phải bổ sung thêm một số tín hiệu điều khiển, ngoài ra tốc độ truy cập bộ nhớ cũng bị giảm theo. Đó là **phương pháp địa chỉ multiplex**, các chip nhớ DRAM thường dùng phương pháp này.



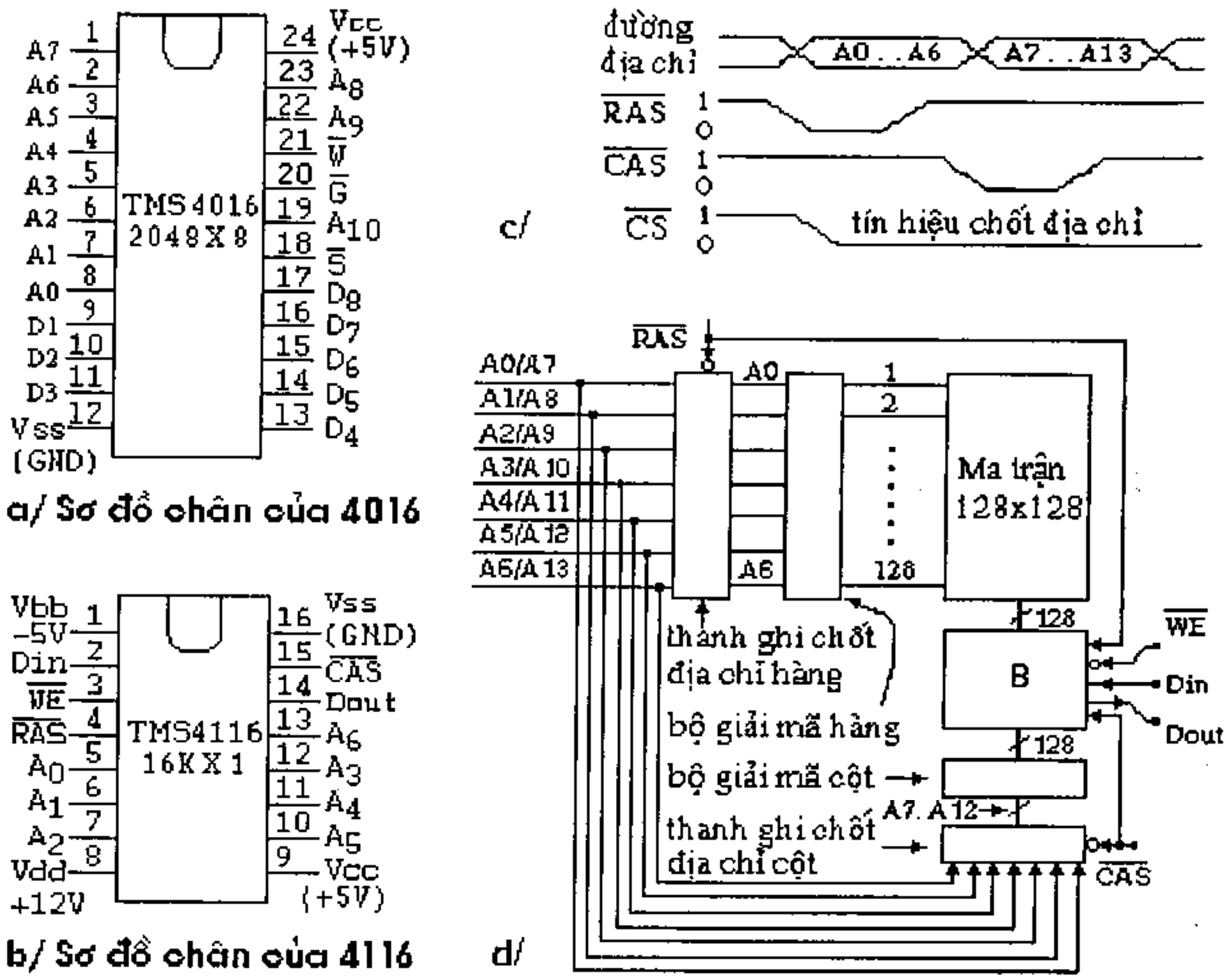
Hình 3-29 Bộ nhớ ROM 2048x8 sử dụng phương pháp giải mã địa chỉ 2 bước

Hình 3-30a là bố trí chân ra của chip nhớ SRAM ký hiệu TMS 4016 dung lượng 2048x8 có bus dữ liệu 8 dây dùng chung cho tín hiệu vào và tín hiệu ra. Hình 3-30b là bố trí chân ra của chip nhớ DRAM ký hiệu TMS 4116 dung lượng 16Kx1 bit có đầu vào và ra dữ liệu tách biệt.

Ta thấy rằng cần có 14 dây địa chỉ để đánh địa chỉ 16.384 ô nhớ (vì 2^{14}=16.384=16Kx1) nhưng thay cho việc dùng 14 dây địa chỉ người ta chỉ dùng có 7 dây. Trong quá trình truy cập bộ nhớ, 7 đường dây địa chỉ đầu tiên truyền thông tin về địa chỉ hàng rồi sau đó truyền thông tin về địa chỉ cột. Để kiểm soát các thao tác này người ta dùng các đường dây truyền tín hiệu điều khiển $\overline{RAS}$ (Row Address Select) và $\overline{CAS}$ (Column Address Select) như trên hình 3-30c và 3-30d. Khi dây tín hiệu $\overline{RAS}$ tích cực (thấp) thì tín hiệu trên các đường địa chỉ sẽ được mở thông qua thanh ghi chốt địa chỉ hàng. Khi tín hiệu $\overline{CAS}$ tích cực (thấp) thì tín hiệu trên các đường địa chỉ sẽ được mở thông qua thanh ghi chốt địa chỉ cột. Hai tín hiệu điều khiển này không bao giờ cùng ở mức tích cực. Tín hiệu $\overline{WE}$ tích cực ở mức thấp được sử dụng để xác định thao tác là ghi hay đọc. Khi nó là tích cực thì bit dữ liệu đặt tại đầu vào Din sẽ được ghi vào ô nhớ có địa chỉ đã được chỉ ra; còn khi nó ở mức cao thì bit dữ liệu bên trong ô nhớ được đánh địa chỉ sẽ được đưa ra ngoài, lên đường dữ liệu Dout.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Khối ký hiệu B trên sơ đồ có chức năng khuếch đại nhậy, đệm và làm tươi (Buffers-Sense amps. and refresh).



Hình 3-30 Sử dụng phương pháp địa chỉ Multiplex để giảm số chân ra của chip nhớ

- **Chuyển mạch 3 trạng thái và việc sử dụng chúng trong tổ chức bus**

Trong sơ đồ trên hình 3-28, chúng ta có 3 đường dây dữ liệu vào I_0, I_1, I_2 và 3 đường dây dữ liệu ra. Qua phân tích chúng ta cũng đã thấy rằng khi sử dụng các đường dây dữ liệu vào thì không sử dụng các đường dây dữ liệu ra và ngược lại. Chính vì vậy trên thực tế người ta ***dùng chung các đường dây cho dữ liệu vào và ra***. Để thực hiện được điều này người ta sử dụng một chuyển mạch điện tử đặc biệt để nối đầu ra của các cổng OR với đường dây dữ liệu dùng chung. Nó sẽ thông trong trạng thái đọc và cấm trong trạng thái ghi. Chuyển mạch điện tử này có tên gọi chung là bộ đệm 3 trạng thái (tri-state buffer) bởi vì nó có thể truyền qua các giá trị 0, 1 hoặc không truyền qua giá trị nào.

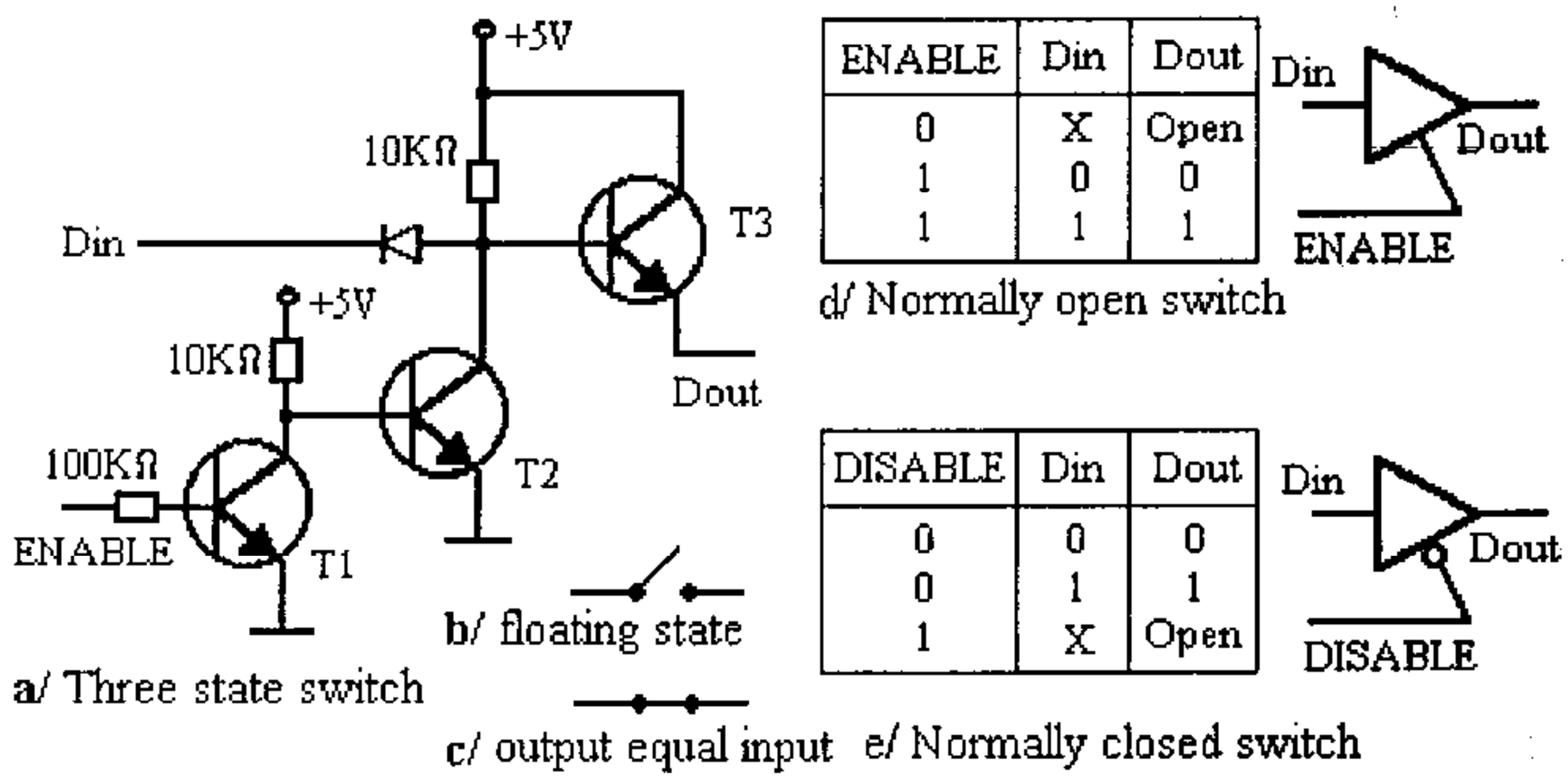
Hình 3-31a là một thí dụ về chuyển mạch 3 trạng thái. Đầu vào ENABLE có thể là low hoặc high. Khi ENABLE = low, transistor T1 bị cấm và T2 thì bão hoà (thông hoàn toàn), vì vậy làm cho mức điện áp ở cực base của T3 hạ xuống bằng mức điện áp của đất (0v), làm cấm lớp tiếp giáp base-emitter của T3, nói cách khác mạch base-emitter của T3 như một diode có cực dương nối đất, trở kháng của diode này bằng vô cùng (trạng thái trở kháng cao). Chính vì vậy đầu ra Dout được thả nổi,

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

bị cách ly khỏi Din và hoàn toàn không phụ thuộc vào nó. Trạng thái thả nối này tương đương với một công tắc mở, như trên hình 3-31b.

Khi ENABLE = high, transistor T1 thông bão hoà và T2 bị cấm, vì vậy T3 hoạt động như một bộ lặp lại emitter, mạch điện lúc này tương đương với một công tắc đóng như trên hình 3-31c, Dout = Din, tức là Dout sẽ là low hoặc high giống như Din.

Các chuyển mạch 3 trạng thái trong thực tế phức tạp hơn, tuy nhiên chức năng thì cũng đơn giản như sơ đồ nguyên lý trên, đầu ra Dout sẽ nằm ở một trong ba trạng thái: low, high hoặc thả nổi.



Hình 3-31 Chuyển mạch 3 trạng thái

Chuyển mạch bình thường là mở

Trên hình 3-31d là ký hiệu của một bộ đệm 3 trạng thái không đảo, nếu ENABLE=low thì lối ra Dout được thả nổi, nếu ENABLE=high thì lối ra sẽ là 0 hoặc 1 giống lối vào Din, người ta gọi đó là chuyển mạch bình thường mở (Normally Open Switch).

Chuyển mạch bình thường là đóng

Trên hình 3-31e là ký hiệu của chuyển mạch bình thường đóng (Normally Closed Switch). Lối vào điều khiển DISABLE tích cực ở mức thấp, nói cách khác chuyển mạch đóng khi DISABLE=low và mở khi DISABLE=high.

Bộ nhớ mà chúng ta đã tìm hiểu như trên hình 3-28 về nguyên tắc có thể đặt vào trong một chip có 14 chân, kể cả chân nguồn nuôi và chân nối đất. Những chip RAM có dung lượng cỡ như vậy trước kia được sử dụng rộng rãi, các chip RAM ngày nay có dung lượng lớn hơn rất nhiều, có thể nhiều megabit, được tạo nên từ hàng triệu cổng. Chính vì vậy việc nâng cao tỉ số cổng/chân của IC được những người thiết kế chip đặc biệt quan tâm, nó có liên quan đến ***sự tổ chức bên trong của các chip nhớ.***

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Ngoài ra bộ nhớ của máy tính cũng thường rất khác nhau, từ vài trăm KB tới vài trăm MB, độ dài từ cũng thường khác nhau, chẳng hạn 1 bit, 8 bit, 16 bit, 32 bit hoặc lớn hơn. Nói chung độ dài từ máy thường lớn hơn độ dài từ của một chip nhớ.

Mặc dù công nghiệp vi điện tử đã sản xuất được các chip nhớ có dung lượng lớn, nhưng dù sao mỗi chip RAM cũng chỉ có sức chứa giới hạn, cần ***nối ghép các chip nhớ để có được độ dài từ và tổng số từ cần thiết***. Để có thể ghép nối các chip nhớ với nhau, các chip nhớ bán dẫn được thiết kế sao cho mỗi chip có đầy đủ những chức năng của một bộ nhớ như:

- Là 1 mảng những phần tử nhớ, mỗi phần tử chứa 1 bit hoặc một nhóm bit.
- Có mạch logic để giải mã địa chỉ cho từng phần tử nhớ.
- Có mạch logic cho phép đọc được nội dung của phần tử nhớ.
- Có mạch logic cho phép ghi (nếu là loại bộ nhớ cho phép ghi).

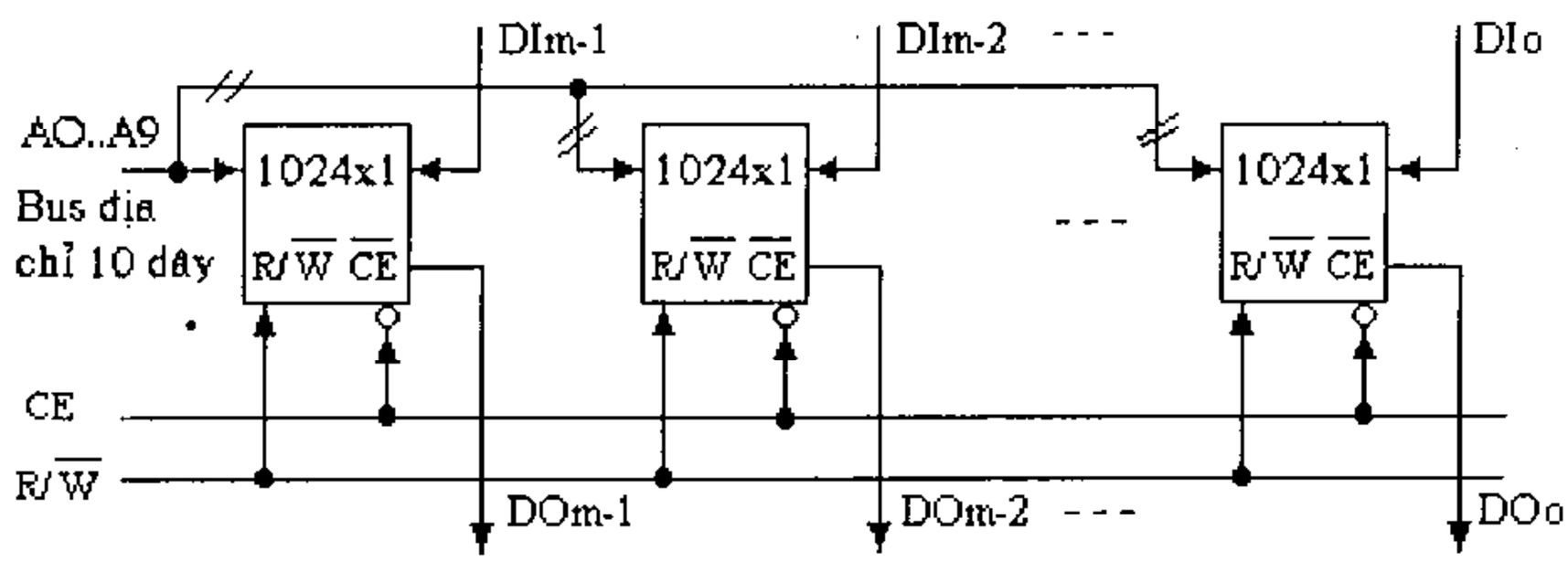
Ngoài ra để việc nối ghép giữa các chip nhớ với nhau và giữa chúng với các mạch logic khác được dễ dàng, các chip nhớ chứa những bộ đệm vào, bộ kích ra và phần mạch mở rộng địa chỉ.

Mạng nhớ bán dẫn được tổ chức sao cho đạt được tốc độ hoạt động cao, dung lượng lớn và giá thành mạch nhớ và các mạch phụ cận thấp.

3.3.4.2 Tổ chức bộ nhớ có dung lượng, độ dài từ mong muốn từ các chip nhớ

- **Tổ chức bộ nhớ có độ dài từ mong muốn**

Muốn tăng độ dài từ, ta đặt đầu ra của 2 hay nhiều vi mạch nhớ song song với nhau (song song chứ không phải là nối lại với nhau), mỗi đầu ra này sẽ ứng với 1 bit của từ. Hình 3-32 là thí dụ về việc tăng độ dài từ bằng cách sử dụng m chip nhớ 1024x1 ghép song song với nhau để được bộ nhớ 1024xm (bộ nhớ có dung lượng 1024 từ, mỗi từ có m bit).



Hình 3-32 Bộ nhớ 1024xm được xây dựng từ m chip nhớ loại 1024x1

Tín hiệu điều khiển CE khi có mức tích cực (thấp) sẽ cho phép cả m chip hoạt động (đọc/ghi), nếu có mức cao thì chúng bị cấm. Bus địa chỉ 10 dây A0-A9 đưa vào m chip nhớ, các bộ giải mã hàng trong từng chip này sẽ chọn lấy một hàng (trên

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

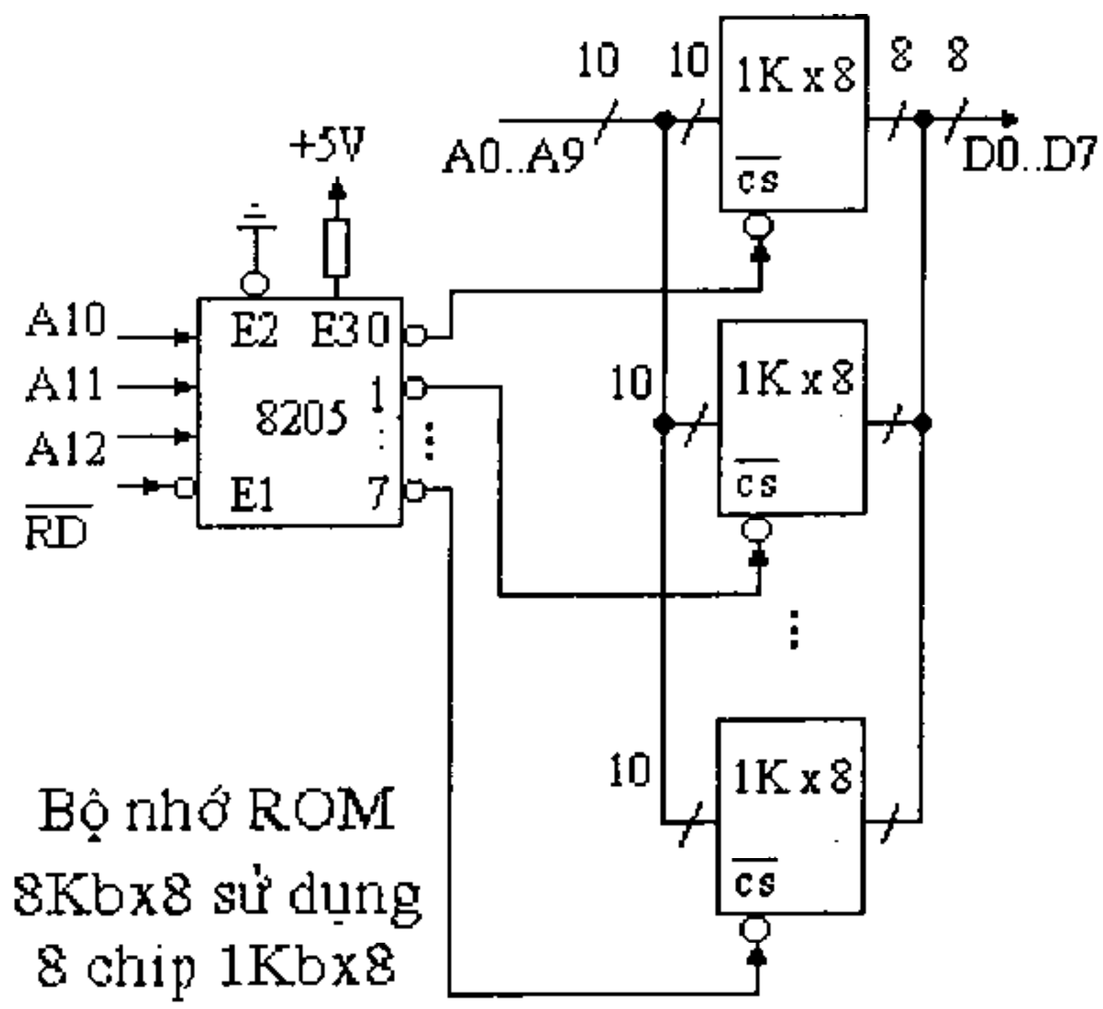
mỗi hàng chỉ có 1 phần tử nhớ 1 bit), m phần tử được chọn trong m chip sẽ nhận dữ liệu vào hoặc đưa dữ liệu ra theo tín hiện điều khiển $R/\overline{W}$.

- **Tổ chức bộ nhớ có dung lượng mong muốn**

Muốn tăng dung lượng bộ nhớ ta dồn đầu ra của 2 hay nhiều vi mạch nhớ. Các vi mạch nhớ bán dẫn có những đặc điểm giúp thực hiện việc này dễ dàng. Hình 3-33 dưới đây minh hoạ cách tăng dung lượng bộ nhớ bằng cách dồn đầu ra của một số vi mạch nhớ.

Đầu vào $\overline{CS}$ trên vi mạch nhớ cho phép chọn vi mạch nhớ đó. Khuyên nhỏ ở đầu vào $\overline{CS}$ cho biết rằng tín hiệu chọn chip $\overline{CS}$ tích cực ở mức thấp. Khuyên nhỏ ở đầu vào E1, E2 cho biết 2 đầu vào Chip Enable này có mức tích cực là mức thấp, còn E3 tích cực ở mức cao. Các đầu ra có khuyên nhỏ cho biết rằng khi ở mức tích cực chúng có mức logic thấp.

Chip 8205 trên Hình 3-33 sử dụng 3 đường địa chỉ $A_{12}A_{11}A_{10}$ để tạo ra 8 tín hiệu chọn chip cho một bộ nhớ 8kx8 bit được xây dựng từ 8 chip nhớ 1Kx8. Khi bộ giải mã được sử dụng theo kiểu này, những chip nhớ được chọn một cách tuyến tính cũng giống như cách chọn một từ bởi bộ giải mã địa chỉ bên trong chip nhớ.



Hình 3-33 Tăng số lượng từ trong bộ nhớ

Như vậy, để chọn được một từ trong bộ nhớ 8Kbx8 cần một địa chỉ 13 bit, 10 bit thấp được đưa tới đầu vào địa chỉ của cả 8 chip nhớ, 3 bit cao được đưa tới đầu vào địa chỉ của bộ giải mã chọn chip. Một trong 8 đầu ra của chip giải mã này sẽ có mức tích cực (thấp, low), chip nhớ nào nối với nó sẽ được phép hoạt động đọc/ghi.

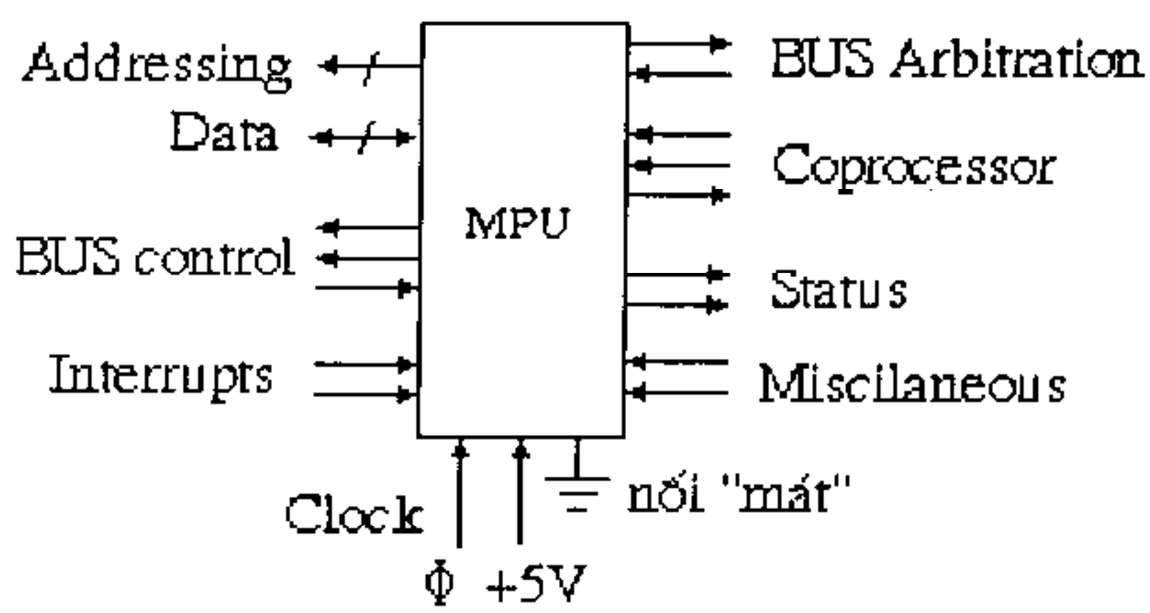
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

3.4 CHIP VI XỬ LÝ VÀ CÁC BUS

3.4.1 Chip vi xử lý (MPU - Microprocessor Unit)

Đó là những chip có chức năng như một CPU đầy đủ, nghĩa là chip này chứa đựng cả 3 đơn vị chức năng số học-logic, điều khiển và bộ nhớ trong của máy tính điện tử, vì vậy đôi khi người ta cũng coi chip vi xử lý là CPU. Chip MPU là vi mạch có chức năng hoạt động rất linh hoạt và công dụng nhất trong các loại vi mạch số, nó là hạt nhân của hệ vi xử lý, thực hiện các phép tính logic, số học và điều khiển toàn bộ hoạt động của hệ, nó nhận các lệnh từ bộ nhớ, giải mã lệnh và thực hiện lệnh.

Chip vi xử lý thường có từ 40 cho đến 132 chân, mọi sự liên hệ với thế giới bên ngoài đều phải thông qua các chân này. Một số chân để gửi thông tin từ bộ vi xử lý ra ngoài, một số chân để nhận thông tin từ ngoài vào bên trong bộ vi xử lý, cũng có một số chân thực hiện cả hai việc này.



Hình 3-34 Các chân ra của một MPU điển hình

Các chân của chip vi xử lý có thể được chia thành 3 nhóm: địa chỉ, dữ liệu và điều khiển. Các chân này được nối với các chân tương ứng của các chip nhớ và các chip I/O thông qua các đường dây chạy song song gọi là bus.

Để nhận lệnh, bộ vi xử lý đầu tiên đặt địa chỉ của lệnh đó lên các chân địa chỉ, sau đó nó truyền tín hiệu Read (đọc) một từ lên đường điều khiển, bộ nhớ đáp lại bằng cách đặt từ mà bộ vi xử lý cần lên đường truyền dữ liệu nối tới các chân dữ liệu của bộ vi xử lý, sau đó phát một tín hiệu báo cho vi xử lý là công việc đã được thực hiện. Khi đó bộ vi xử lý nhận tín hiệu (lệnh) này vào và cho thực hiện. Lệnh này lại có thể đòi hỏi đọc hoặc ghi dữ liệu, như vậy quá trình lại tiếp tục tương tự như vậy.

Cần lưu ý là trên một số chân của bộ vi xử lý tín hiệu ứng với +5V kích hoạt một công việc nào đó, nhưng trên một số chân khác tín hiệu -5V lại là tín hiệu kích hoạt một công việc nào đó khác. Để tránh nhầm lẫn ta sẽ dùng thuật ngữ kích hoạt (hoặc tích cực) chứ không nói tín hiệu cao (high) hay thấp (low).

Hai tham số quan trọng của bộ vi xử lý là số chân địa chỉ và số chân số liệu. Một chip có m chân địa chỉ có thể đánh địa chỉ được tới 2^m vị trí nhớ. Trong thực tế

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

m thường có các giá trị là 16, 20, 24, 32. Tương tự như vậy một chip có n chân số liệu có thể đọc hoặc ghi một từ n-bit trong một thao tác đọc hoặc ghi. Trong thực tế n thường có các giá trị là 8, 16, 32. Một MPU có 8 chân dữ liệu sẽ cần 4 thao tác để đọc 1 từ 32 bit, trong khi đó MPU với 32 chân dữ liệu chỉ cần có 1 thao tác.

Ngoài các chân địa chỉ và số liệu, MPU còn có các chân điều khiển, chúng điều khiển dòng số liệu và phân định thời gian cho các tín hiệu đi vào và đi ra khỏi MPU.

Các chân điều khiển có thể được chia thành các nhóm sau:

1. Điều khiển bus (bus control): các tín hiệu đi ra từ MPU tới các bus, báo cho các đơn vi khác như bộ nhớ hay các chip I/O biết MPU muốn đọc/ghi hay làm một công việc gì khác.
2. Ngắt (Interrupts): đó là các chân truyền tín hiệu từ các thiết bị bên ngoài (I/O Device) đi vào MPU, yêu cầu MPU phục vụ. Trong hầu hết các hệ thống, chính MPU có thể ra lệnh cho các thiết bị I/O bắt đầu thực hiện một thao tác nào đó, thí dụ như kiểm tra xem có xảy ra lỗi không.
3. Trọng tài bus (bus arbitration): đó là các chân được sử dụng để truyền tín hiệu điều hoà sự lưu thông trên các bus, tránh việc có trện một thiết bị cố chiếm dụng bus, gây đụng độ.
4. Phát tín hiệu cho bộ đồng vi xử lý (Coprocessor signaling)

   Có nhiều chip MPU được thiết kế để có thể cùng làm việc với bộ đồng vi xử lý - coprocessor, các chip coprocessor này thường là các chip tính toán số dấu phảy động, cũng có khi là các chip xử lý đồ hoạ .v.v. Để việc trao đổi dữ liệu giữa MPU và bộ đồng vi xử lý được thuận lợi, MPU thường có các chân đưa ra các tín hiệu trao đổi với coprocessor.
5. Trạng thái (Status): các chân này thông báo trạng thái của MPU hoặc nhận thông tin trạng thái từ các thiết bị khác.
6. Các tín hiệu khác (Miscelaneous: linh tinh): các MPU khác nhau có thể có các tín hiệu khác nhau, thí dụ tín hiệu để reset MPU .v.v.

Bộ vi xử lý ngày càng được sử dụng nhiều trong thực tế: trong đồng hồ, lò nung siêu cao tần, máy giặt, các rô-bốt, các thiết bị kiểm soát nhiệt độ trong các cao ốc, bộ khống chế nhiên liệu trong ô tô, trong các động cơ máy bay và tất nhiên là trong máy tính điện tử.

Hiện nay trên thế giới có một số hãng sản xuất bộ vi xử lý, trong đó Intel là một trong các hãng lớn nhất.

Bộ nhớ trong của bộ vi xử lý

Bộ nhớ trong của MPU là bộ nhớ nhỏ, tốc độ cao được sử dụng để chứa tạm thời lệnh, dữ liệu, kết quả trung gian và những thông tin điều khiển nhất định. Bộ nhớ này bao gồm các thanh ghi và có thể cả bộ nhớ ROM, RAM.

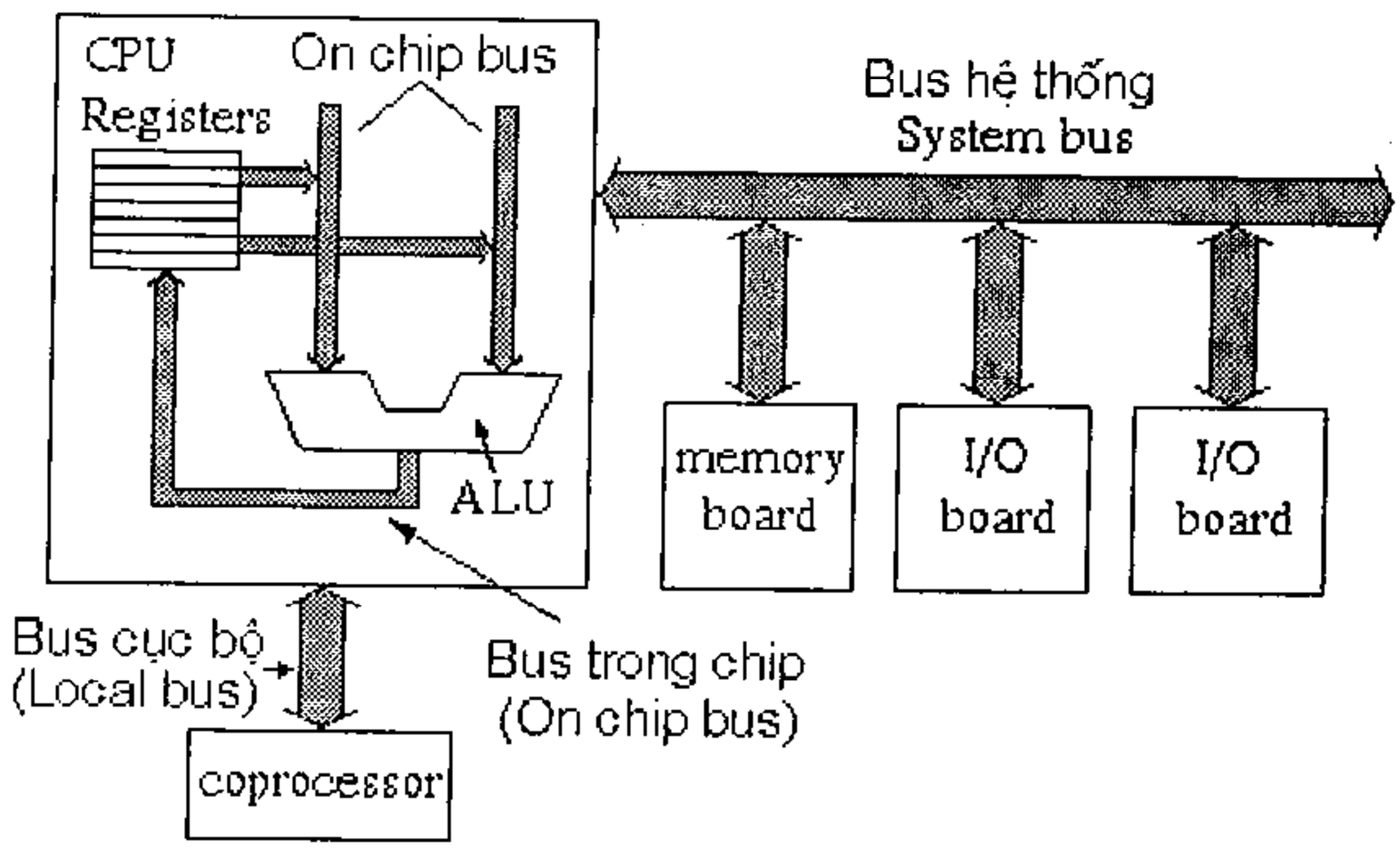
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Mỗi thanh ghi của MPU thường có những chức năng nhất định, thí dụ thanh ghi PC (Program Counter), nó trỏ đến lệnh tiếp theo sẽ được thực hiện; thanh ghi lệnh IR (Instruction Register) chứa lệnh đang được thực hiện .v.v.
- Bộ nhớ ROM bên trong bộ vi xử lý thường chứa các vi chương trình, chương trình này là tập các vi lệnh, nhằm thực hiện các lệnh của bộ vi xử lý.
- Bộ nhớ RAM bên trong bộ vi xử lý thường để chứa các lệnh sắp được thực hiện hoặc các số liệu được sử dụng thường xuyên và được gọi là "cache" nội.

3.4.2 Các bus của máy tính

Bus là đường truyền tín hiệu điện chung nối các thiết bị khác nhau trong một hệ thống máy tính. Bus thường bao gồm từ 50 đến 100 dây dẫn được gắn chặt với Main board, trên các dây này có các đầu nối đưa ra, các đầu này được sắp xếp và cách nhau những khoảng quy định để có thể cắm vào đó các bảng mạch điều khiển vào ra hoặc bộ nhớ.

Cũng có những bus được dùng cho mục đích chuyên biệt, thí dụ nối một bộ vi xử lý với một hoặc nhiều bộ vi xử lý khác hoặc nối với bộ nhớ cục bộ.



Hình 3-35 Các bus trong một hệ thống máy tính

Trong chính bộ vi xử lý cũng có một số bus để nối các thành phần bên trong của bộ vi xử lý với nhau. Người thiết kế MPU có thể tuỳ ý lựa chọn loại bus bên trong MPU, còn với các bus liên hệ với bên ngoài MPU, cần phải xác định rõ các quy tắc làm việc cũng như các đặc điểm kỹ thuật về điện và cơ khí của bus để những người thiết kế Main board có thể ghép nối MPU với các thiết bị khác. Nói cách khác, các bus này cần phải tuân theo một chuẩn nào đó. Tập các quy tắc của chuẩn còn được gọi là nghi thức bus (bus protocol).

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

giới máy tính có rất nhiều loại bus khác nhau được sử dụng, các bus là không tương thích với nhau. Sau đây là một số bus được sử dụng

Tên bus	Lĩnh vực áp dụng
- Camac	Vật lý hạt nhân.
- EISA	Một số hệ thống có chip 80386
- IBM PC, PC/AT	Máy IBM PC, IBM/PC/AT
- Massbus	Máy PDP-11 và VAX
- Microchannel	Máy PS/2
- Multibus I	Một số hệ thống có 8086
- Multibus II	Một số hệ thống có chip 80386
- Versabus	Một số hệ thống có chip vi xử lý của Motorola
- VME	Một số hệ thống có chip vi xử lý họ 68x0 của Motorola

Người ta thường phân loại bus theo 3 cách sau:

1. Theo tổ chức phần cứng (như các loại bus nêu bên trên).
2. Theo nghi thức truyền thông (bus đồng bộ và không đồng bộ).
3. Theo loại tín hiệu truyền trên bus (bus địa chỉ, bus dữ liệu...)

Sự làm việc của các bus

Thường có nhiều thiết bị nối với bus, một số thiết bị là tích cực (active) và có thể đòi hỏi truyền thông tin trên bus, trong khi đó lại có các thiết bị thụ động chờ các yêu cầu từ các thiết bị khác. Các thiết bị tích cực được gọi là **chủ bus (master)** còn các thiết bị thụ động là **tớ (slave).**

Khi CPU ra lệnh cho bộ điều khiển đĩa đọc/ghi một khối dữ liệu thì CPU là master còn bộ điều khiển đĩa là slave. Tuy nhiên khi bộ điều khiển đĩa ra lệnh cho bộ nhớ nhận dữ liệu mà nó đọc từ đĩa thì nó lại giữ vai trò của master. Bảng 3-03 là một số cặp master - slave điển hình.

Bus Driver và Bus Receiver

Tín hiệu điện mà các thiết bị trong máy tính phát ra thường không đủ mạnh để điều khiển được bus, nhất là khi bus khá dài và có nhiều thiết bị nối với nó. Chính vì thế mà hầu hết các bus master được nối với bus thông qua một chip gọi là ***bus Driver***, về căn bản đó là một bộ khuyếch đại tín hiệu số. Tương tự như vậy hầu hết các slave được nối với bus thông qua ***bus Receiver***. Đối với các thiết bị có thể khi thì đóng vai trò master khi thì đóng vai trò slave, người ta sử dụng một chip kết hợp gọi là ***transceiver***. Các chip này đóng vai trò ghép nối và thường là các thiết bị 3 trạng thái, cho phép nó có thể ở trạng thái thứ ba - hở mạch (còn gọi thả nổi).

Giống như MPU, bus có các đường địa chỉ, đường số liệu và đường điều khiển. Tuy nhiên không nhất thiết phải có ánh xạ một - một giữa các tín hiệu ở các chân ra của MPU và các đường dây của bus. Thí dụ một số MPU có 3 chân ra, truyền ra

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

các tín hiệu báo MPU đang thực hiện các thao tác Memory Read, Memory Write, I/O Read, I/O Write hoặc thao tác khác. Một bus điển hình thường có một đường Memory Read, đường thứ hai cho Memory Write, đường thứ ba cho I/O Read, đường thứ tư cho I/O Write .v.v.

Bảng 3-03 Một số cặp master-slave điển hình

Master	Slave	Thí dụ
CPU	Memory	Lấy lệnh và dữ liệu
CPU	I/O	Khởi đầu việc truyền dữ liệu
CPU	Coprocessor	CPU chuyển nhiệm vụ tính toán số dấu phảy động cho Coprocessor
I/O	Memory	DMA (Direct Memory Access)
Coprocessor	Memory	Lấy toán hạng

Những vấn đề quan trọng nhất liên quan đến thiết kế bus là: Nhịp đồng hồ bus (sự phân chia thời gian, hay còn gọi bus cloking), cơ chế trọng tài bus (bus arbitration), xử lý ngắt và xử lý lỗi.

Các bus có thể được chia theo nghi thức truyền thông tin thành hai loại riêng biệt là bus đồng bộ và bus không đồng bộ phụ thuộc vào việc sử dụng nhịp đồng hồ bus.

3.4.3 Bus đồng bộ (Synchronous bus)

Bus đồng bộ có một đường dây điều khiển bởi một bộ dao động thạch anh, tín hiệu trên đường dây này có dạng sóng vuông, với tần số thường nằm trong khoảng 5MHz - 50MHz. Mọi hoạt động bus xảy ra trong một số nguyên lần chu kỳ này và được gọi là chu kỳ bus.

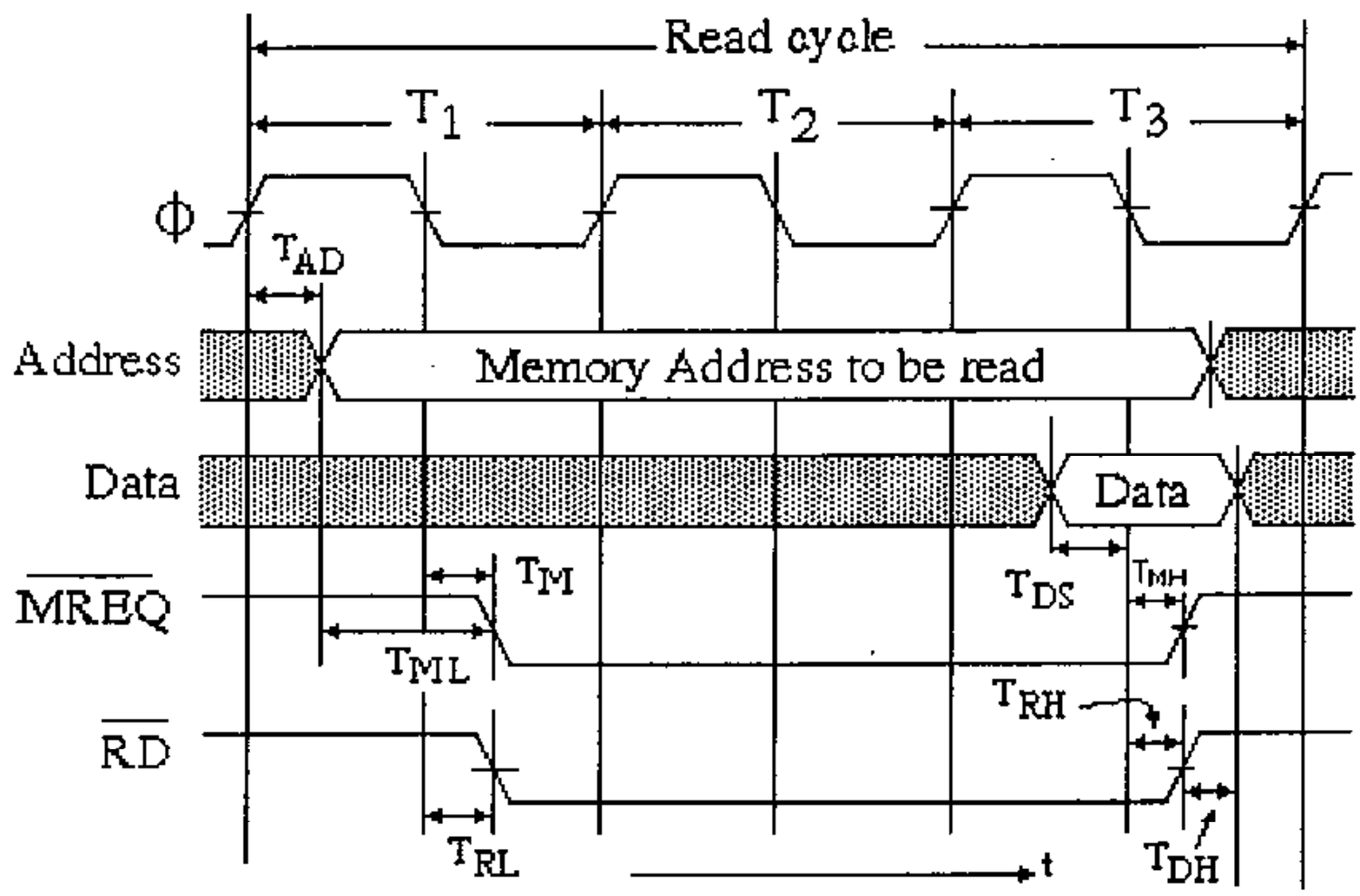
Trên Hình 3-36 là giản đồ thời gian của một bus đồng bộ với tần số đồng hồ là 4 MHz, như vậy chu kỳ bus là 250ns.

Giả sử việc đọc 1 byte từ bộ nhớ chiếm 3 chu kỳ bus (750ns), tương ứng với T1, T2, T3 trên hình vẽ. Vì tất cả các tín hiệu điện thay đổi mức không phải là tức thời, nên trên hình vẽ có các sườn xung, ta giả sử các sườn xung kéo dài 10ns.

- **T1** bắt đầu bằng sườn lên của tín hiệu đồng hồ ϕ, trong một phần thời gian của T1, MPU đặt địa chỉ của byte cần đọc lên bus địa chỉ, trên hình vẽ phần được tô xám chỉ rằng các giá trị trong khoảng thời gian này là không quan trọng. Tương tự như vậy các giá trị ứng với miền xám của tín hiệu Data cũng không quan trọng. Sau khi tín hiệu địa chỉ được thiết lập giá trị mới, MPU đặt các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ tích cực (mức thấp). Tín hiệu $\overline{MREQ}$ - truy cập bộ nhớ (Memory

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

REQest) chứ không phải thiết bị I/O; còn tín hiệu $\overline{RD}$ - chọn Read chứ không phải Write.



Hình 3-36 Chu kỳ đọc trong bus đồng bộ

- **T2** - là thời gian cần thiết để bộ nhớ giải mã địa chỉ và đưa dữ liệu lên bus dữ liệu.
- **T3** - Tại sườn xuống của T3, MPU nhận dữ liệu trên bus dữ liệu, chứa vào thanh ghi bên trong MPU và chốt dữ liệu. Sau đó MPU đảo các tín hiệu $\overline{MREQ}$ và $\overline{RD}$.

Như vậy thao tác đọc đã hoàn thành, tại chu kỳ máy tiếp theo MPU có thể thực hiện một thao tác khác.

Các giá trị cụ thể về thời gian của hình vẽ trên được ghi trong bảng 3-04, có thể giải thích chi tiết hơn như sau:

- **T_{AD} :** Theo giản đồ thời gian $T_{AD} \leq 110$ns, nghĩa là nhà sản xuất chip MPU đảm bảo rằng trong mọi chu kỳ đọc toán hạng từ bộ nhớ, MPU sẽ đưa ra tín hiệu địa chỉ không chậm hơn 110ns tính từ thời điểm giữa sườn lên của T1.

T_{DS} : Giá trị nhỏ nhất là 50ns, có nghĩa là nhà sản xuất chip nhớ phải đảm bảo rằng dữ liệu đã ổn định trên bus dữ liệu ít nhất là 50ns trước thời điểm giữa sườn xuống của T3. Yêu cầu về thời gian này đảm bảo cho MPU đọc dữ liệu tin cậy.

Khoảng thời gian bắt buộc đối với T_{AD} và T_{DS} cũng nói lên rằng, trong trường hợp tồi nhất, bộ nhớ chỉ có 250+250+125-110-50=465ns tính từ thời điểm có tín hiệu địa chỉ cho tới khi nó sinh ra dữ liệu trên bus dữ liệu. Nếu bộ nhớ không có khả năng đáp ứng đủ nhanh, nó cần phát tín hiệu đòi chờ $\overline{WAIT}$ trước sườn xuống của T2. Thao tác này đưa thêm vào một trạng thái chờ- **wait state** (cũng tức là đưa

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

thêm vào một số chu kỳ bus), khi bộ nhớ đã đưa ra tín hiệu ổn định, nó sẽ đảo tín hiệu $\overline{WAIT}$ thành WAIT. (Chú ý: tín hiệu wait không vẽ ở sơ đồ này).

Bảng 3-04 Giá trị tới hạn của một số thông số thời gian

Ký hiệu	Tham số	Min (ns)	Max (ns)
T_{AD}	Thời gian trễ của tín hiệu địa chỉ		110
T_{ML}	Thời gian địa chỉ ổn định trước tín hiệu $\overline{MREQ}$	60	
T_M	Thời gian trễ của $\overline{MREQ}$ so với sườn xuống của tín hiệu đồng hồ T1		85
T_{RL}	Thời gian trễ của $\overline{RD}$ so với sườn xuống của tín hiệu đồng hồ T1		85
T_{DS}	Thời gian thiết lập dữ liệu trước sườn xuống của tín hiệu đồng hồ	50	
T_{MH}	Thời gian trễ của $\overline{MREQ}$ so với sườn xuống của tín hiệu đồng hồ T3		85
T_{RH}	Thời gian trễ của $\overline{RD}$ so với sườn xuống của tín hiệu đồng hồ T3		85
T_{DH}	Thời gian lưu trữ dữ liệu từ lúc đảo tín hiệu $\overline{RD}$	0	

- **T_{ML} :** Đảm bảo rằng tín hiệu địa chỉ sẽ được thiết lập trước tín hiệu $\overline{MREQ}$ ít nhất là 60ns. Khoảng thời gian này là quan trọng nếu tín hiệu $\overline{MREQ}$ điều khiển sự tạo ra tín hiệu chọn chip (CS hoặc CE) bởi vì một số chip nhớ đòi hỏi phải nhận được tín hiệu địa chỉ trước tín hiệu chọn chip.

  Qua các số liệu được phân tích trên, ta thấy rằng với bus này người thiết kế máy tính sẽ không thể chọn chip nhớ với thời gian thiết lập là 75ns.

- **T_M , T_{RL} :** Các giá trị bắt buộc đối với 2 đại lượng này có nghĩa là cả hai tín hiệu $\overline{MREQ}$ và $\overline{RD}$ sẽ là tích cực trong khoảng thời gian 85ns tính từ thời điểm xuống của xung đồng hồ T1. Trong trường hợp tồi nhất, chip nhớ chỉ có 250+250-85-50=365ns sau khi hai tín hiệu trên là tích cực để đưa dữ liệu ra bus dữ liệu. Sự bắt buộc về thời gian này bổ sung thêm sự bắt buộc thời gian với tín hiệu đồng hồ.

- **T_{MH} , T_{RH} :** Hai đại lượng này cho biết cần có bao nhiêu thời gian để các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ được đảo sau khi dữ liệu đã được MPU nhận vào.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- T_{DH} : Cho biết bộ nhớ cần phải giữ data trên bus bao lâu sau khi tín hiệu $\overline{RD}$ đã đảo.

Giản đồ thời gian một chu kỳ đọc trên bus đồng bộ hình 3-36 đã được đơn giản hoá so với thực tế, trong đó số tín hiệu cần sử dụng lớn hơn nhiều.

Block Transfer

Ngoài các chu kỳ đọc/ghi, một số bus đồng bộ còn hỗ trợ truyền dữ liệu theo khối. Khi một thao tác đọc khối bắt đầu, bus master báo cho slave biết có bao nhiêu byte cần được truyền đi, thí dụ truyền đi con số này trong chu kỳ T1, sau đó đáng lẽ truyền đi một byte, slave đưa ra trong mỗi chu kỳ một byte cho tới khi đủ số byte được thông báo. Như vậy khi đọc dữ liệu theo khối, n byte dữ liệu cần n+2 chu kỳ đồng hồ chứ không phải 3n chu kỳ.

Cách khác làm cho bus truyền dữ liệu nhanh hơn là làm cho các chu kỳ ngắn lại. Trong thí dụ trên mỗi byte được truyền đi trong 750ns, vậy bus có giải thông là 1.33 Mbytes/sec. Nếu đồng hồ có tần số 8MHz, thời gian một chu kỳ chỉ còn một nửa, giải thông sẽ là 2.67MBytes/sec.

Tuy vậy việc giảm chu kỳ bus dẫn đến các khó khăn về mặt kỹ thuật, các tín hiệu truyền trên các đường dây khác nhau không phải luôn có cùng vận tốc, dẫn đến một hiệu ứng gọi là **bus skew** (skew: nghiêng, xiên, ghềnh). Điều quan trọng là thời gian chu kỳ phải dài hơn so với skew để tránh việc những khoảng thời gian được số hoá lại trở thành các đại lượng biến thiên liên tục.

Khi nghiên cứu về bus cũng cần quan tâm đến một vấn đề nữa, đó là tín hiệu tích cực là thấp hay cao, điều này tuỳ thuộc vào người thiết kế bus xác định tín hiệu nào thuận lợi hơn.

3.4.4 Bus không đồng bộ (asynchronous bus)

Bus không đồng bộ không sử dụng đồng hồ giữ nhịp, chu kỳ của nó có thể kéo dài tuỳ ý và có thể khác nhau đối với các cặp thiết bị khác nhau.

Làm việc với các bus đồng bộ dễ dàng hơn do nó được định thời một cách gián đoạn, tuy vậy chính đặc điểm này cũng dẫn đến nhược điểm. Thứ nhất là: mọi công việc được tiến hành trong những khoảng thời gian là bội số của nhịp đồng hồ bus, nếu một thao tác nào đó của MPU hay bộ nhớ có thể hoàn thành trong 3.1 chu kỳ, thì nó cũng sẽ phải kéo dài thành 4 chu kỳ. Điều dở hơn nữa là khi đã chọn chu kỳ bus và đã xây dựng bộ nhớ, I/O Card cho bus này thì khó có thể tận dụng được những tiến bộ của công nghệ. Chẳng hạn sau khi đã xây dựng bus với sự định thời (phân chia thời gian) như trên, công nghệ mới đưa ra các MPU và chip nhớ có thời gian chu kỳ là 100ns (chứ không còn là 250ns như cũ), chúng vẫn cứ phải chạy với tốc độ thấp như các MPU và chip nhớ loại cũ, bởi vì nghi thức bus đòi hỏi chip nhớ phải đưa được dữ liệu ra và ổn định ngay trước thời điểm ứng với sườn xuống của T3. Nếu có nhiều thiết bị khác nhau nối với một bus, trong đó một số thiết bị có thể

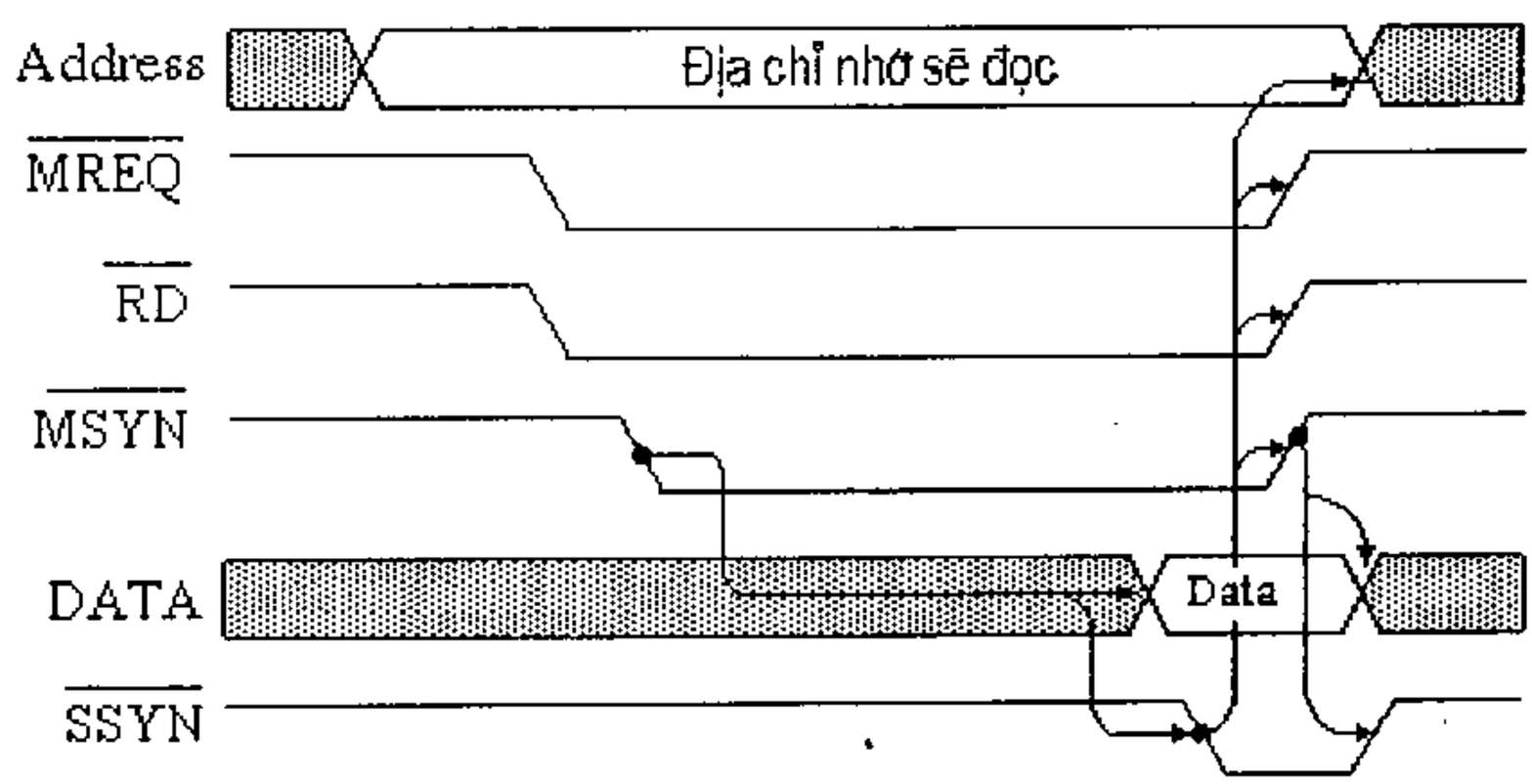
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

hoạt động nhanh hơn các thiết bị khác thì cần phải đặt bus hoạt động phù hợp với thiết bị có tốc độ thấp nhất.

Bus không đồng bộ ra đời chính là nhằm khắc phục các nhược điểm của bus đồng bộ. Hình 3-37 minh hoạ sự hoạt động của bus không đồng bộ, trong đó master yêu cầu đọc bộ nhớ.

Trước hết master cần phát ra địa chỉ nhớ (Address) mà nó muốn truy cập, sau đó phát tín hiệu $\overline{MREQ}$ tích cực để báo rằng nó muốn truy cập bộ nhớ (Memory) chứ không phải cổng I/O. Tín hiệu này là cần thiết vì bộ nhớ và các cổng I/O có thể sử dụng chung một miền địa chỉ. Sau khi phát địa chỉ, bên master cũng phải phát tín hiệu $\overline{RD}$ tích cực để bên slave biết rằng master sẽ thực hiện thao tác đọc (Read) chứ không phải là ghi (Write).

Các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ được đưa ra sau tín hiệu địa chỉ bao lâu tuỳ thuộc vào tốc độ hoạt động của master. Sau khi hai tín hiệu này đã ổn định, bên master sẽ phát ra một tín hiệu đặc biệt gọi là $\overline{MSYN}$ (Master SYNchronization) ở mức tích cực để báo cho bên slave biết rằng các tín hiệu cần thiết đã sẵn sàng trên bus, slave có thể nhận lấy. Khi slave nhận được tín hiệu này, nó sẽ thực hiện công việc với tốc độ nhanh nhất có thể thực hiện được, ở đây là nhanh chóng đưa dữ liệu của ô nhớ được yêu cầu lên bus dữ liệu. Khi hoàn thành công việc, slave sẽ phát tín hiệu $\overline{SSYN}$ (Slave SYNchronization) tích cực.



Hình 3-37 Hoạt động của bus không đồng bộ

Khi master nhận được tín hiệu $\overline{SSYN}$ tích cực, nó biết rằng dữ liệu của slave đã sẵn sàng và thực hiện việc chốt dữ liệu lại, sau đó đảo các đường địa chỉ cũng như các tín hiệu $\overline{MREQ}$, $\overline{RD}$ và $\overline{MSYN}$.

Khi slave nhận được sự đảo tín hiệu $\overline{MSYN}$ thành không tích cực, nó biết rằng một chu kỳ đã kết thúc và đảo tín hiệu $\overline{SSYN}$. Bây giờ bus lại trở lại trạng thái ban đầu, mọi tín hiệu đều là không tích cực, tất cả chờ bus master mới.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Trên giản đồ thời gian của bus không đồng bộ, ta sử dụng mũi tên để thể hiện nguyên nhân và kết quả. Việc đưa $\overline{MSYN}$ lên mức tích cực dẫn đến việc truyền dữ liệu ra bus dữ liệu và đồng thời cũng dẫn đến việc slave phát ra tín hiệu $\overline{SSYN}$ tích cực; đến lượt mình, tín hiệu $\overline{SSYN}$ lại gây ra sự đảo mức của các đường địa chỉ, $\overline{MREQ}$, $\overline{RD}$ và $\overline{MSYN}$. Cuối cùng sự đảo mức của $\overline{MSYN}$ lại gây ra sự đảo mức tín hiệu $\overline{SSYN}$ và kết thúc một chu kỳ đọc.

Full handshake

Tập các tín hiệu phối hợp với nhau như vậy được gọi là Full handshake, nó chủ yếu gồm có 4 tín hiệu (4 sự kiện):

1. $\overline{MSYN}$ được đặt tích cực
2. $\overline{SSYN}$ được đặt tích cực để đáp lại tín hiệu $\overline{MSYN}$
3. $\overline{MSYN}$ được đảo để đáp lại tín hiệu $\overline{SSYN}$ (tích cực)
4. $\overline{SSYN}$ được đảo để đáp lại sự đảo tín hiệu $\overline{MSYN}$ thành không tích cực.

Ta có thể nhận xét thấy Full handshake là độc lập thời gian, nghĩa là mỗi sự kiện được gây ra bởi một sự kiện trước đó chứ không phải bởi một xung đồng hồ. Nếu một cặp master-slave nào đó hoạt động chậm thì cặp master-slave kế tiếp không hề bị ảnh hưởng.

Tuy ưu điểm của bus không đồng bộ rất rõ ràng, nhưng trong thực tế phần lớn các bus đang được sử dụng lại là loại đồng bộ. Lý do căn bản là các hệ thống sử dụng bus đồng bộ dễ thiết kế hơn. MPU chỉ cần chuyển các mức tín hiệu cần thiết sang trạng thái tích cực là các chip nhớ đáp lại ngay, không cần tín hiệu phản hồi. Chỉ cần các chip được chọn phù hợp thì mọi hoạt động đều trôi trảy, không cần phải "bắt tay" (handshake).

3.4.5 Trọng tài bus (bus arbitration)

Trong hệ thống máy tính không phải chỉ có CPU làm chủ bus (bus master), thực tế các chip I/O cũng có lúc phải làm chủ bus để có thể đọc hoặc viết vào bộ nhớ và để gọi ngắt; Các bộ đồng vi xử lý cũng có thể làm chủ bus. Như vậy nảy sinh vấn đề: điều gì xảy ra nếu có từ hai thiết bị trở lên đồng thời muốn trở thành chủ bus? Câu trả lời là cần có một cơ chế trọng tài để tránh sự hỗn loạn của hệ thống. Cơ chế trọng tài có thể là tập trung hoặc không tập trung.

3.4.5.1 Trọng tài bus tập trung

Đầu tiên ta sẽ xem xét cơ chế trọng tài tập trung. Hình 3-38 là một thí dụ đơn giản. Trong thí dụ này, một trọng tài bus duy nhất sẽ quyết định "ai" sẽ đến lượt tiếp theo. Nhiều bộ vi xử lý có đơn vị trọng tài được chế tạo nằm ngay bên trong chip CPU, trong một số máy tính mini, đơn vị trọng tài nằm ngoài chip CPU.

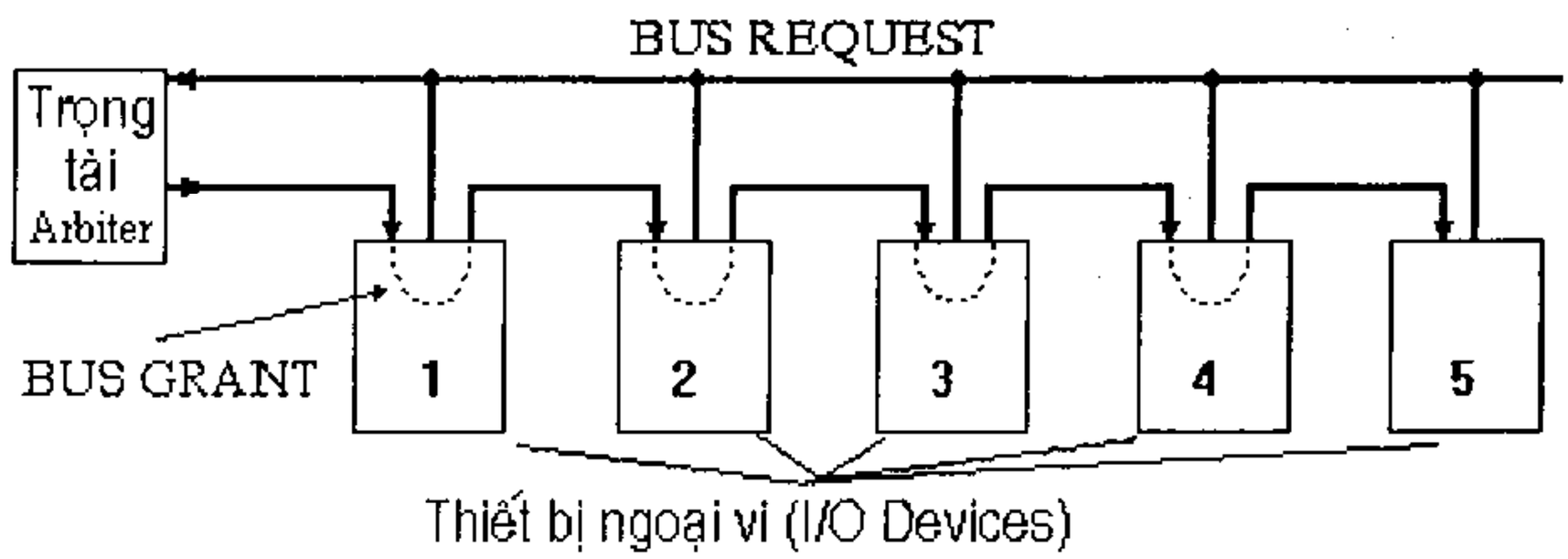
Theo cơ chế này trọng tài chỉ có thể biết là có yêu cầu chiếm dụng bus hay không, chứ không biết được có bao nhiêu đơn vị muốn chiếm dụng bus. Khi trọng tài bus nhận được một yêu cầu, nó sẽ phát ra một tín hiệu cho phép trên đường dây

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

bus grant (cho sử dụng bus). Đường dây này nối qua tất cả các thiết bị vào/ra theo kiểu nối tiếp.

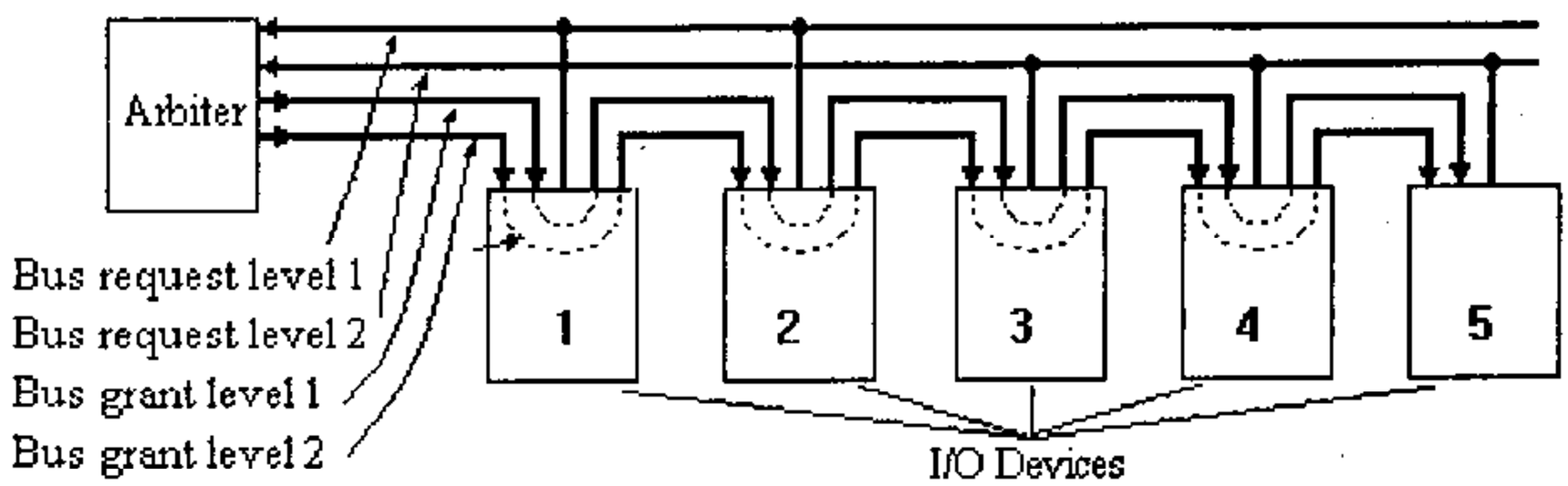
Khi thiết bị nằm gần trọng tài nhất nhận được tín hiệu cho phép (grant), nó sẽ kiểm tra xem có phải chính nó đã phát ra yêu cầu không? Nếu đúng thì nó sẽ chiếm lấy bus và không truyền tiếp tín hiệu cho phép trên đường dây. Nếu nó kiểm tra thấy nó không phát tín hiệu yêu cầu, thì nó sẽ truyền tín hiệu cho phép tới thiết bị kế tiếp trên đường dây, với thiết bị này sự việc xảy ra giống thiết bị trước nó, quá trình cứ tiếp diễn cho tới khi có một thiết bị chiếm lấy bus.

Sơ đồ như vậy có tên gọi là daisy chaining (tràng hoa cúc). Điểm nổi bật của sơ đồ này là các thiết bị được gán thứ tự ưu tiên tuỳ thuộc vào vị trí của nó so với trọng tài, thiết bị gần trọng tài hơn thì có mức độ ưu tiên cao hơn.



Hình 3-38 Trọng tài bus tập trung có một mức, dùng cách mắc nối tiếp

Một số loại bus có nhiều mức độ ưu tiên, với mỗi mức ưu tiên có một đường dây yêu cầu bus (bus request) và một dường dây cho chiếm bus (bus grant). Trên hình 3-39 là một thí dụ, bus này có 2 mức ưu tiên 1 và 2 (các bus trên thực tế thường có 4, 8 hay tới 16 mức). Mỗi thiết bị trong hệ thống máy tính nối với một trong các mức yêu cầu bus, các thiết bị thường được sử dụng hơn được gắn với đường dây có mức ưu tiên cao hơn. Trên hình vẽ, các thiết bị 1 và 2 sử dụng mức ưu tiên 1, còn các thiết bị 3, 4 và 5 sử dụng mức ưu tiên 2.



Hình 3-39 Trọng tài bus tập trung có hai mức

Nếu có một số thiết bị ở các mức ưu tiên khác nhau cùng yêu cầu, trọng tài bus sẽ chỉ phát ra tín hiệu cho phép (grant) đối với yêu cầu có mức ưu tiên cao nhất.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Trong số các thiết bị có cùng mức ưu tiên, thiết bị gần trọng tài bus hơn sẽ có ưu tiên cao hơn.

Với sơ đồ ở hình 3-39, về mặt kỹ thuật có thể không cần nối đường dây "Bus grant level 2" thông qua thiết bị 1 và 2 bởi vì các thiết bị này không bao giờ có thể đòi hỏi bus ở mức 2. Tuy nhiên trong thực tế, để thuận tiện cho việc lắp đặt người ta vẫn hay làm như vậy - nối tất cả các đường dây grant thông qua tất cả các thiết bị, như vậy sẽ dễ dàng hơn là nối các đường dây grant một cách riêng biệt, căn cứ vào việc xem xét thiết bị nào có quyền ưu tiên cao hơn.

Một số trọng tài bus có đường dây thứ ba nối tới các thiết bị để các thiết bị khẳng định việc nhận được tín hiệu cho phép (grant) và chiếm dụng bus - gọi là đường dây biên nhận - acknowledgement, thường được gọi ngắn gọn là đường dây ACK. Ngay sau khi một thiết bị phát tín hiệu tích cực trên đường dây ACK, có thể đảo tín hiệu trên các đường dây request và grant xuống mức không tích cực. Kết quả là các thiết bị khác có thể đòi hỏi chiếm dụng bus trong khi thiết bị đầu tiên đang sử dụng bus. Khi sự truyền thông trên bus kết thúc, bus master kế tiếp sẽ được lựa chọn. Cách làm việc như vậy làm tăng hiệu quả sử dụng bus, nhưng cần thêm một đường truyền tín hiệu và cấu trúc của các thiết bị cũng phải phức tạp hơn. Các chip trong các máy tính PDP-11 và các chip Motorola làm việc với các bus như vậy.

3.4.5.2 Trọng tài bus không tập trung

Trong cơ chế trọng tài bus không tập trung, không cần sử dụng một đơn vị riêng làm trọng tài bus, nhờ vậy giảm được giá thành của phần cứng. Trong một số loại máy tính khác nhau, người ta đã sử dụng một vài kiểu trọng tài bus theo cơ chế này, sau đây là một thí dụ.

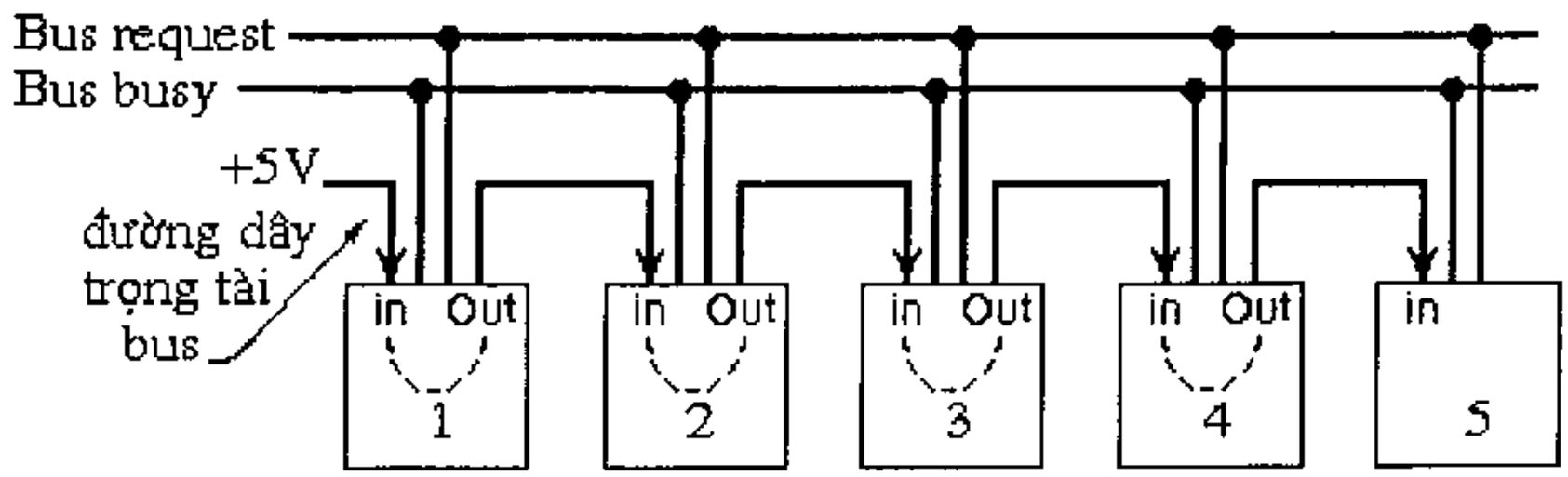
Trọng tài bus không tập trung trong Multibus

Trong Multibus, người ta cho phép có thể lựa chọn sử dụng trọng tài bus tập trung hoặc không tập trung, cơ chế trọng tài bus không tập trung được thực hiện theo sơ đồ trên hình 3-40.

Người ta chỉ sử dụng 3 đường dây, không phụ thuộc vào số lượng thiết bị nối với bus. Đường thứ nhất, trên cùng là đường yêu cầu chiếm dụng bus - Bus request; Đường thứ hai, ở giữa gọi là đường báo bận - Bus busy, đường này sẽ được đơn vị đang là bus master đặt ở mức tích cực; Đường thứ ba, ở dưới được dùng làm trọng tài bus, nó được mắc nối tiếp thành một chuỗi xích qua tất cả các thiết vị ngoại vi. Đầu của chuỗi này được gắn cố định với mức điện áp +5v của nguồn nuôi.

Khi không có đơn vị nào yêu cầu chiếm dụng bus, đường dây trọng tài bus truyền mức tích cực tới tất cả các thiết bị trong chuỗi xích. Khi một đơn vị nào đó muốn chiếm dụng bus, đầu tiên nó kiểm tra xem bus có rỗi không và đầu vào IN của đường trọng tài bus có mức tích cực không, nếu không (not active) thì nó không được trở thành bus master. Nếu đầu vào IN có mức tích cực, nó sẽ đảo đầu ra OUT thành mức không tích cực, làm cho tất cả các thiết bị đứng sau nó trong chuỗi xích có đầu vào IN ở mức không tích cực.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



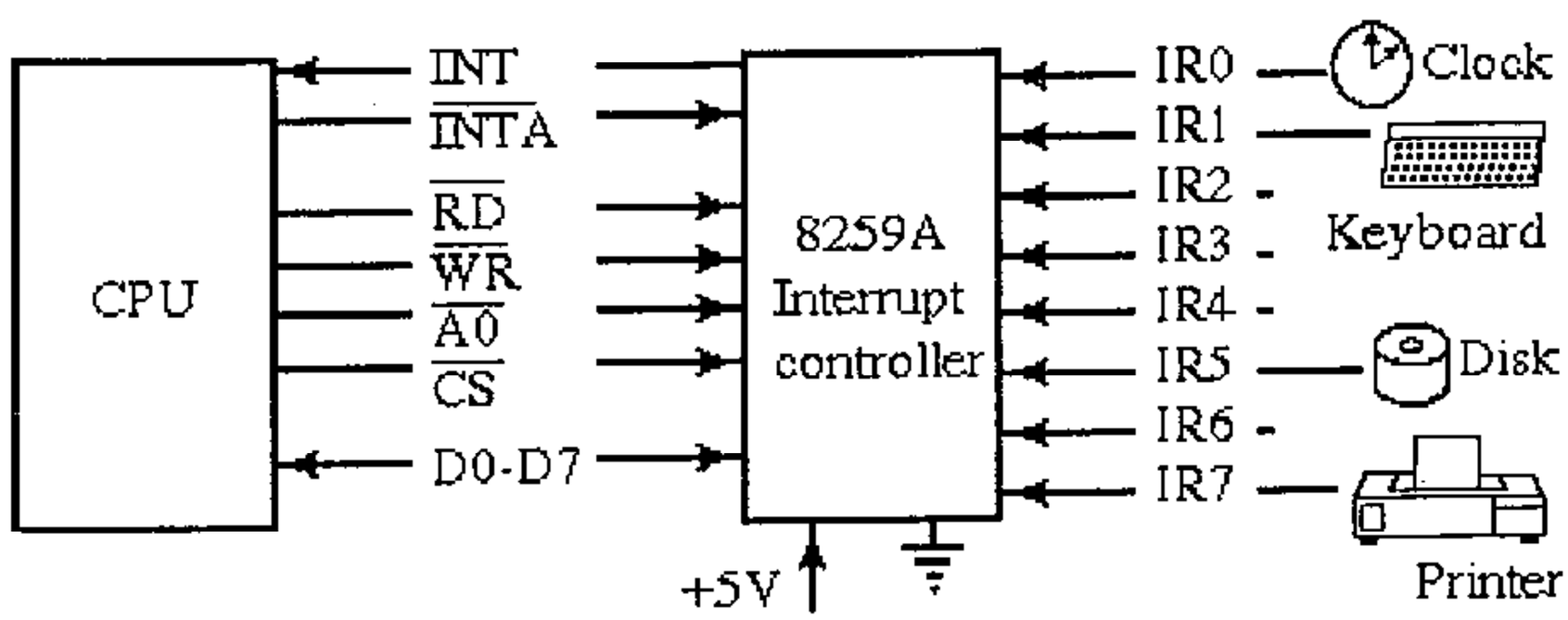
Hình 3-40 Trọng tài bus không tập trung trong Multibus

Khi trạng thái có thể hiểu lầm (khoảng thời gian tín hiệu trên các đầu IN và OUT của các thiết bị đang thay đổi) qua đi, chỉ còn lại duy nhất một thiết bị có đầu vào IN có mức tích cực và đầu ra OUT được đưa xuống mức không tích cực. Thiết bị này trở thành bus master, nó sẽ đặt dây báo bận - busy lên mức tích cực và bắt đầu truyền tin trên bus.

3.4.6 Việc xử lý ngắt

Cho đến đây chúng ta mới chỉ thảo luận các chu kỳ bus thông thường, trong đó master nhận hoặc gửi thông tin từ/đến slave. Một ứng dụng quan trọng nữa của bus là để xử lý ngắt. Khi CPU ra lệnh cho một thiết bị I/O làm một việc gì đó, nó thường chờ đợi tín hiệu ngắt do thiết bị I/O phát ra khi hoàn thành công việc được CPU yêu cầu. Khi nhận được tín hiệu ngắt, CPU sẽ phản ứng ngay, đó có thể là việc nhận dữ liệu do thiết bị I/O đã truyền về, cũng có thể là việc gửi tiếp dữ liệu ra thiết bị I/O hoặc CPU sẽ sử dụng bus cho một thao tác khác... Như vậy chính ngắt phát ra tín hiệu yêu cầu sử dụng bus.

Vì có thể xảy ra trường hợp nhiều thiết bị ngoại vi cùng phát ra ngắt, cho nên cũng cần có một cơ chế trọng tài giống như đối với các bus thông thường. Giải pháp thường dùng là gán các mức độ ưu tiên cho các thiết bị và sử dụng một trọng tài tập trung để trao quyền ưu tiên cho các thiết bị quan trọng thường xuyên được sử dụng. Hiện trên thị trường có những chip điều khiển ngắt được tiêu chuẩn hoá và được sử dụng rộng rãi. Thí dụ các máy IBM PC, PC/AT, PS/2 và cả dòng máy tương thích với IBM PC đều sử dụng chip 8259A, như được minh hoạ trên hình 3-41.



Hình 3-41 Việc sử dụng bộ điều khiển ngắt 8259A

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Có thể nối trực tiếp đến 8 chip điều khiển I/O tới các đầu vào IRx (Interrupt Request) của chip 8259A. Khi có một thiết bị nào đó muốn ngắt, nó đặt mức tích cực lên một chân IRx. Khi 8259A nhận được tín hiệu tích cực ở một hay một số đầu vào IRx nó sẽ đặt mức tích cực lên đầu dây INT (INTerrupt), tín hiệu trên dây này truyền trực tiếp tới chân interrupt của CPU. Khi CPU có thể xử lý được ngắt, nó gửi trở lại một tín hiệu chấp nhận ngắt cho 8259A trên đường dây INTA (INTerrupt Acknowledge). Vào thời điểm này CPU chờ 8259A chỉ ra lối vào (input) nào đã yêu cầu ngắt bằng cách gửi số hiệu của lối vào đó lên bus dữ liệu (D0-D7) để đi tới CPU. Sau đó phần cứng của CPU sẽ sử dụng con số đó để tính chỉ số trong một bảng con trỏ, có tên gọi là bảng **vector ngắt - interrupt vector** để tìm địa chỉ một chương trình con, cho chạy chương trình này để phục vụ cho ngắt. Các chương trình con như thế được gọi là chương trình con xử lý ngắt.

Chip 8259A có một số thanh ghi bên trong, CPU có thể ghi vào các thanh ghi này bằng cách dùng các chu kỳ bus thông thường và các chân tín hiệu $\overline{RD}$ (ReaD), $\overline{WR}$ (WRite), $\overline{CS}$ (Chip Select) và A0. Khi số lượng thiết bị ngoại vi lớn hơn 8, có thể sử dụng 2 hoặc nhiều chip 8259A. Trong trường hợp cần số thiết bị ngoại vi lớn, 8 đầu vào IRQ_0 .. IRQ_7 có thể nối với 8 đầu ra INT của 8 chip 8259A bổ sung, 8 chip bổ sung này có tổng cộng 64 đầu vào, như vậy có thể nối với 64 thiết bị I/O. Chip 8259A còn có một số chân được sử dụng khi ghép nhiều chip này với nhau nhưng không vẽ trên hình để đơn giản hoá vấn đề.

3.5 THÍ DỤ VỀ MỘT SỐ BUS

3.5.1 Bus IBM PC

Đây là thí dụ điển hình về một loại bus được sử dụng trong các hệ thống cần giá thành thấp, nó được sử dụng rộng rãi trong các hệ thống vi xử lý dựa trên chip 8088. Hầu hết họ PC, kể cả các máy tương thích, sử dụng bus này. Chính bus IBM PC tạo nên cơ sở cho bus IBM PC/AT và nhiều bus khác. Bus này có 62 đường dây trong đó có 20 đường địa chỉ, 8 đường số liệu và các đường tín hiệu khác.v.v. được liệt kê ở bảng 3-05.

- Các cột có nhãn In: chỉ ra các tín hiệu đi từ bus vào motherboard (mainboard).
- Các cột có nhãn Out: chỉ ra các tín hiệu được sinh ra trên motherboard và đi vào bus.

Về mặt vật lý, bus IBM PC được gắn chặt vào Motherboard và có khoảng gần chục đầu nối dạng khe (slot) để cắm các card mở rộng, trên mỗi khe cắm 62 dây được chia thành 2 hàng. IBM gọi các đầu nối này là I/O Channel.

(Hình 9-05 là sơ đồ đã được đơn giản hoá của phần chính yếu nhất, là "trái tim" của máy tính họ PC, nó bao gồm 7 chip chính. Một số chip phụ không vẽ trên sơ đồ, bộ nhớ cũng không được vẽ vì về mặt logic nó được nối với MPU thông qua bus.)

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Bảng 3-05 Các tín hiệu của PC bus

Tín hiệu	Số dây	In	Out	Giải thích
OSC	1		X	70ns (14.31818MHz): T.hiệu đồng hồ
CLK	1		X	210ns (4.77 MHz): Tín hiệu đồng hồ
RESET	1		X	Tín hiệu để reset CPU và thiết bị I/O
A0-A19	20		X	20 Address lines: đường dây địa chỉ
D0-D7	8		X	8 Data lines: đường truyền dữ liệu
ALE	1		X	Address latch enable: Chốt địa chỉ
(MEMR)	1		X	Memory read: đọc bộ nhớ
(MEMW)	1		X	Memory Write: ghi vào bộ nhớ
(IOR)	1		X	I/O read: đọc cổng I/O
(IOW)	1		X	I/O write: ghi ra cổng I/O
AEN	1	X		Address ENable: Yêu cầu CPU thả nổi bus
(IOCHCHK)	1	X		I/O CHannel ChecK
IOCHRDY	1	X		I/O CHannel ReaDY
IRQ2-IRQ7	6	X		Các đường yêu cầu ngắt
DRQ1-DRQ3	3	X		DMA ReQuest: Yêu cầu DMA
DACK0-DACK3	4		X	DMA ACKnowledge lines
T/C	1		X	Terminal/Count: báo hoàn thành thao tác DMA
Power	5			±5Volts, ±12Volts: các dây nguồn
GND	3			Ground: dây đất ("mát")
Reserved	1			Không sử dụng đến

- **Tín hiệu đồng hồ OSC và CLK**

Các máy IBM PC đầu tiên sử dụng đồng hồ thạch anh tần số 14.31818 MHz, tần số này không phải được chọn tuỳ tiện mà nhằm để tạo ra tín hiệu đồng bộ màu (color burst) theo hệ truyền hình màu NTSC, vì ban đầu IBM cho rằng nhiều khách hàng sẽ dùng máy thu hình làm thiết bị hiển thị để tiết kiệm tiền! Tuy thực tế điều đó nói chung không xảy ra nhưng tần số đồng hồ thạch anh đã chọn, IBM không thay đổi nữa. Tần số này cao hơn so với chuẩn 8088 - tần số cực đại là 5 MHz, do đó nó được chia ba thành 4.77 MHz (việc chia 3 dùng chip 8284A). ***Tần số 4.77***

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

MHz được sử dụng làm đồng hồ chính (master) để xác định chu kỳ bus. Tín hiệu tần số 4.77 MHz cũng có trên bus IBM PC và ký hiệu là CLK. Tín hiệu này không cân xứng như tín hiệu đồng hồ thạch anh 14.31818 MHz, trong một chu kỳ 2/3 thời gian có mức thấp, 1/3 thời gian có mức cao. Sau này cũng có một số máy trong họ PC sử dụng chip 8088 có tốc độ cao hơn, tới 8 MHz, vì vậy trong các hệ này tín hiệu CLK cũng tăng lên một cách tỉ lệ.

- **Tín hiệu RESET**

Tín hiệu RESET trên bus do chip 8284A sinh ra. Để reset CPU, các mạch điện bên ngoài gửi tín hiệu tới 8284A, nó sẽ đặt tín hiệu reset lên mức tích cực, buộc CPU và các thiết bị I/O khởi tạo lại chính chúng.

- **Các đường địa chỉ và dữ liệu A0-A19 và D0-D7**

MPU không nối trực tiếp với các đường địa chỉ và đường số liệu của bus, mà thông qua các chip khác. Các đường địa chỉ được chốt bằng cách dùng 3 chip 74LS373 - mỗi chip là một bộ 8 thanh ghi chốt, tuy vậy chỉ sử dụng 20 trong số 24 đường. Cần phải chốt tín hiệu địa chỉ vì một số tín hiệu địa chỉ và số liệu được truyền thông qua một số chân dùng chung của MPU. Đầu mỗi chu kỳ bus của CPU, 8088 đưa ra các tín hiệu địa chỉ, các thanh ghi chốt địa chỉ sẽ chốt các giá trị địa chỉ lại (giữ cho nó không thay đổi) trong khoảng thời gian còn lại của chu kỳ bus, khi các tín hiệu địa chỉ trên các chân của MPU đã mất đi. Chính việc sử dụng các chip chốt địa chỉ làm ẩn đi việc sử dụng chung các chân của MPU (phương pháp multiplexing).

Các đường dữ liệu sẽ được lấy mẫu (đọc nhanh giá trị) hoặc cung cấp giá trị trong những khoảng thời gian xác định, chẳng hạn trong sườn dương của một tín hiệu đồng hồ nào đó, vì vậy không cần chốt. Các đường dữ liệu của bus được điều khiển bởi bus transceiver - chip 74LS245. Chân DIR (DIRection) xác định hướng của tín hiệu là đi tới hay ra khỏi MPU.

Lý do chính của việc nối các chân của MPU với bên ngoài thông qua các bộ đệm chính là vì nó được chế tạo theo công nghệ MOS, MPU không có khả năng cung cấp đủ dòng điện để điều khiển tất cả các phần tử nối với bus; các chip đệm thuộc loại TTL có khả năng cung cấp đủ dòng cho cả bus.

Ngoài ra còn có lý do khác nữa, đó là khi có một thiết bị nào đó không phải MPU muốn trở thành bus master (thí dụ đơn vị DMAC), MPU cần phải thả nổi các bus. Phương pháp đơn giản nhất được áp dụng là thiết bị đó phát tín hiệu AEN (Address ENable) để đảo tín hiệu cho phép đưa ra trên các thanh ghi chốt và transceiver, làm cho các bus được thả nổi.

- **Tín hiệu ALE (Address Latch Enable)**

Tín hiệu ALE được đặt mức tích cực khi MPU đang điều khiển các đường tín hiệu địa chỉ, cho phép các chip 74LS373 biết khi nào cần chốt địa chỉ lại. Tín hiệu này của bus cũng cho bộ nhớ và các chip I/O biết khi nào các tín hiệu trên bus địa chỉ là hợp lệ.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Các đường tín hiệu $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$**

Để điều khiển việc đọc/ghi vào bộ nhớ hoặc các thiết bị vào/ra. Nhờ các tín hiệu này, bus cung cấp hai không gian địa chỉ riêng biệt, một cho MEM và một cho I/O. Bộ nhớ sẽ không phản ứng gì khi thấy tín hiệu $\overline{IOR}$ và $\overline{IOW}$ ở mức tích cực.

MPU sử dụng các tín hiệu $\overline{S0}$ - $\overline{S2}$ đưa vào chip điều khiển bus 8288 để tạo ra các tín hiệu $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$ cùng với tín hiệu ALE. Chip điều khiển bus cũng nhận tín hiệu điều khiển AEN từ bus, tín hiệu này do một thiết bị muốn trở thành bus master phát ra, khi nhận được tín hiệu AEN, chip điều khiển bus sẽ phát tín hiệu điều khiển các chip chốt địa chỉ và chip bus transceiver thả nối bus.

- **Tín hiệu $\overline{IOCHCHK}$ (I/O CHannel ChecK)**

Tín hiệu này sẽ tích cực khi có lỗi chẵn/lẻ (parity) bị phát hiện trên bus. Tín hiệu này sẽ kích một ngắt không thể che.

- **Tín hiệu IOCHRDY (I/O CHannel ReaDY)**

Tín hiệu này do bộ nhớ phát ra khi tốc độ hoạt động của nó thấp, yêu cầu MPU đợi bằng cách chèn wait states vào các chu kỳ đọc, ghi bộ nhớ.

- **Các tín hiệu IRQ2-IRQ7**

Là các tín hiệu do thiết bị ngoại vi sinh ra, đưa vào chip điều khiển ngắt 8259A. Khi có tín hiệu gửi đến chip điều khiển ngắt 8259A nó sẽ theo dõi, phát ra một tín hiệu yêu cầu ngắt tới MPU và đặt số hiệu vector ngắt lên đường dữ liệu khi MPU cần đến. IRQ0 thường được dùng bởi chip đồng hồ - timer, IRQ1 thường được bàn phím sử dụng.

- **Các tín hiệu liên quan đến DMA**

Các tín hiệu còn lại nói chung liên quan đến hoạt động DMA, chẳng hạn khi MPU yêu cầu ổ đĩa đọc một khối dữ liệu, chip điều khiển ổ đĩa sẽ chờ nhận được byte đầu tiên từ đơn vị ổ đĩa đi đến, sau đó phát ra một yêu cầu trở thành bus master để ghi byte đó vào bộ nhớ.

Chip 8237A được INTEL thiết kế nhằm quản lý các nghi thức bus và thực hiện DMA trong đó có việc tăng địa chỉ bộ nhớ và giảm con đếm sau khi truyền mỗi byte. Việc này nó làm thay cho các thiết bị I/O, giúp giảm giá thành của chúng.

Về căn bản chip 8237A là một CPU nhỏ có các chương trình được ghi sẵn bên trong. Khi 8088 muốn bắt đầu hoạt động DMA đối với một thiết bị ngoại vi nào đó, nó nạp số hiệu thiết bị, địa chỉ nhớ, số byte (byte count), hướng truyền và các thông tin khác vào các thanh ghi bên trong chip 8237A. Khi chip điều khiển đã sẵn sàng đọc hoặc ghi byte đầu tiên, nó đặt mức tích cực lên một trong các đường DRQx của bus để đưa vào chip 8237A. Khi nhận được tín hiệu, 8237A đòi chiếm bus và sẵn sàng truyền 1 byte. Chip 8237A phát ra tín hiệu $\overline{DACKx}$ tới chip điều khiển báo cho nó biết hãy ghi byte của mình (đối với thao tác đọc) hoặc đọc byte của mình (đối

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

với thao tác ghi). Trong khoảng một chu kỳ này, chip 8237A điều khiển hoạt động như bus master còn bộ nhớ hoạt động như bus slave.

Chip 8237A có 4 kênh độc lập và có thể quản lý đồng thời 4 đường truyền.

- **Tín hiệu T/C**

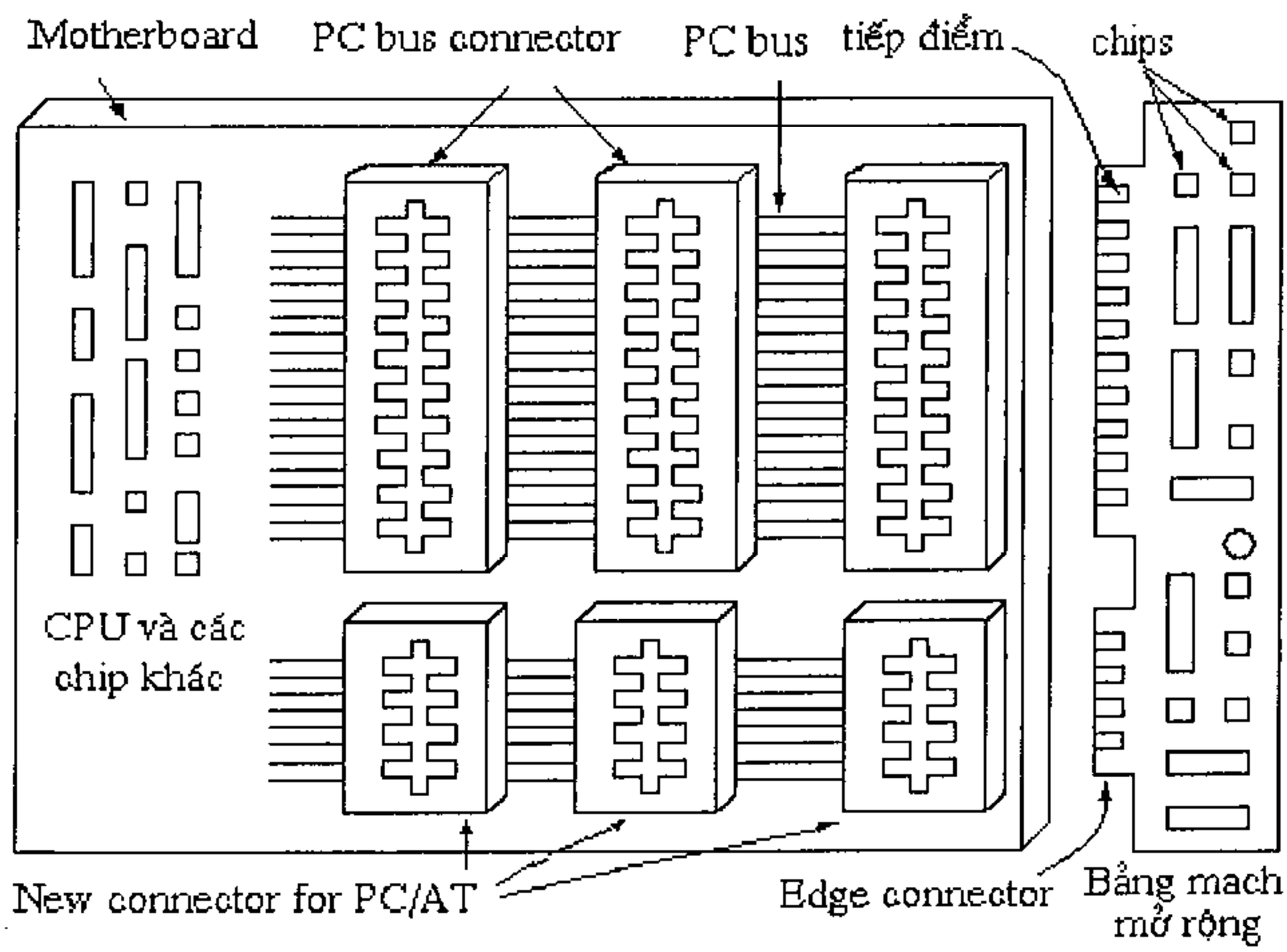
Đường T/C được chip 8237A đặt mức tích cực khi con đếm byte (byte count) bằng 0, báo cho bộ điều khiển I/O biết rằng công việc yêu cầu đã hoàn thành, đã đến lúc báo hiệu cho 8259A gọi ngắt.

- **Các đường dây còn lại**

8 đường dây còn lại của bus là các đường nguồn nuôi, trong đó có 2 đường +5V, một đường -5V, một đường +12V, một đường -12V và 3 đường nối đất.

3.5.2 Bus IBM PC/AT

Khi hãng IBM đưa ra loại máy IBM PC/AT, họ phải giải quyết một vấn đề quan trọng, đó là làm sao vừa kế thừa được thế hệ IBM PC vừa phát huy được những khả năng hơn hẳn của bộ vi xử lý 80286 so với 8088, với bus địa chỉ 24 dây do đó đánh địa chỉ được 16MB bộ nhớ và có bus dữ liệu 16 bit.



Hình 3-42 Bus PC/AT gồm 2 phần: bus PC nguyên thuỷ và phần mở rộng

Hãng IBM đã chọn giải pháp mở rộng PC bus, bổ sung thêm vào các khe cắm cũ một đoạn khe cắm ngắn, trên đó có 36 dây tín hiệu, tăng thêm cho bus địa chỉ 4 dây, bus dữ liệu 8 dây, các đường yêu cầu ngắt, kênh DMA cũng như dây nguồn

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

nuôi và dây đất. Nhờ giải pháp này các bảng mạch mở rộng trước đây vẫn dùng cho IBM PC nay vẫn có thể cắm vào và hoạt động trong máy IBM PC/AT.

Ngoài việc mở rộng bus, tần số tín hiệu đồng hồ bus cũng được tăng từ 4.77MHz ở PC bus thành 8MHz, nhờ đó tốc độ truyền thông trên bus cũng tăng lên nhiều. Xem hình 3-42.

Năm 1991 tổ chức IEEE (Institute of Electrical and Electronic Engineers) đã đưa ra tiêu chuẩn quốc tế cho bus của máy AT, gọi là bus ISA (Industrial Standard Architecture).

Các nhà sản xuất PC khác đã đưa ra một chuẩn của họ, đó là bus EISA (Extended Industry Standard Architecture), về căn bản bus này là sự mở rộng bus PC/AT thành 32 bit, giữ nguyên tính tương thích với các máy tính và các card mở rộng đã có.

Khi IBM thiết kế thế hệ PS/2, đó là thế hệ sau của IBM PC và IBM PC/AT họ đã xây dựng một bus hoàn toàn mới - Microchannel và đăng ký bản quyền.

3.5.3 Bus VME

Trong lĩnh vực máy tính PC, bus IBM PC và các thế hệ cải tiến của nó được sử dụng rộng rãi. Trong các lĩnh vực cần sử dụng các bus có khả năng truyền thông lượng lớn, người ta cũng đã phát triển một số bus, trong đó phải kể đến bus VME (VME - Versa Module Eurocard), nó được phát triển từ Versabus của hãng Motorola, nhưng lại sử dụng bảng mạch có kích thước được tiêu chuẩn hoá của Châu Âu - 160mm x 200mm.

Vào cuối những năm 1970, sau khi hãng Motorola sản xuất ra chip 68000, họ cũng quyết định sản xuất hệ thống máy tính dựa trên chip này, hệ thống đầu tiên có tên gọi Exormacs, trên bảng mạch chính của nó có bus được gọi là Versabus. Sau đó họ quyết định sử dụng kích thước được tiêu chuẩn hoá của Châu Âu cho bus này và cho ra đời bus VME, đó là một chuẩn được chấp nhận rộng rãi (de facto standard). Chuẩn VME được sử dụng rộng rãi trong các ứng dụng cần tốc độ truyền thông cao (32 bit), đặc biệt trong lĩnh vực công nghiệp.

Một số đặc điểm kỹ thuật của bus VME

Các đường dây của bus này có thể được chia thành 4 nhóm:

1. Các dậy truyền dữ liệu
2. Các dây trọng tài bus
3. Các dây truyền tín hiệu yêu cầu ngắt có gán các mức ưu tiên khác nhau.
4. Các dây truyền các tín hiệu phục vụ cho các ứng dụng khác

Bus VME hỗ trợ việc truyền 8,16 và 32 bit cũng như việc đánh địa chỉ 16, 24 và 32 bit. Nó cũng hỗ trợ việc truyền khối (Block Transfer), kích thước khối không quá 256 byte.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Một số ưu điểm của bus VME so với bus IBM PC/AT

Các đơn vị dùng bus VME dễ làm việc với nhau hơn vì người ta đã mô tả rất chi tiết về bus này, do đó các hãng sản xuất card điều khiển khác nhau có thể áp dụng đúng đắn. Trong khi đó các định nghĩa về bus của IBM không đầy đủ bằng, khiến cho có thể xảy ra tình trạng từng card điều khiển có thể hoạt động tốt với CPU nhưng khi có nhiều card cùng hoạt động với CPU thì lại xảy ra trục trặc (người ta thường gọi là "đụng độ").

Hiệu suất truyền thông của bus VME cao vì nó là bus không đồng bộ. Mỗi board liên kết với bus có thể truyền thông với tốc độ cao nhất của nó, tốc độ này được quyết định bởi trình độ hiện thời của công nghệ. Trong thực tế chu kỳ bus có thể chỉ còn 100ns (theo tài liệu năm 1990), tức là có thể truyền 4 byte (32 bit) trong 100ns, tính ra được giải thông là 40 MB/s.

Bus VME có độ tin cậy cao hơn bus IBM PC, nó không sử dụng 2 hàng các tiếp điểm dẹt trên card mà sử dụng tới 3 hàng 32 chân cắm tròn, đảm bảo tiếp xúc tốt hơn, tuy nhiên chi phí sẽ cao hơn. Ngoài ra còn nhiều tính năng làm cho độ tin cậy cao hơn.

3.6 GIAO DIỆN - INTERFACING

Một hệ thống máy tính điển hình từ cỡ nhỏ tới cỡ trung bình bao gồm một chip vi xử lý, các chip nhớ và một số chip điều khiển vào/ra, tất cả các thành phần này được nối với nhau bằng một bus. Chúng ta đã nghiên cứu khá chi tiết bộ nhớ, bộ vi xử lý và các bus. Phần này sẽ nghiên cứu thành phần cuối cùng là các chip vào/ra. Máy tính của chúng ta liên hệ với thế giới bên ngoài thông qua các chip này.

3.6.1 Các chip vào/ra (I/O chips)

Rất nhiều chip vào/ra đã từng có mặt trên thị trường và các chip mới cũng không ngừng được đưa ra. Trong số các chip thông dụng phải kể đến các chip điều khiển truyền thông UART, USART, chip điều khiển màn hình CRTC, chip điều khiển các đơn vị ổ đĩa HDC/FDC và chip điều khiển vào/ra qua các cổng song song PIO (Parallel Input/Output). Sau đây chúng ta sẽ tìm hiểu một số trong các chip này.

- **UART - Chip Thu-Phát không đồng bộ**

Chip UART (Universal Asynchronous Receiver Transmitter) có thể đọc một byte từ bus dữ liệu và gửi từng bit của nó ra đường dây nối tiếp tới terminal hoặc nhận dữ liệu từ terminal. Các UART thường hoạt động ở các tốc độ từ 50bps tới 19.2 kbps; các ký tự có độ rộng từ 5 tới 8 bit; số bit dừng (stop bit) là 1, 1.5 hoặc 2; và hỗ trợ tính chẵn lẻ là even (chẵn) hoặc odd (lẻ). Tất cả các tham số trên đều có thể điều khiển được bằng chương trình.

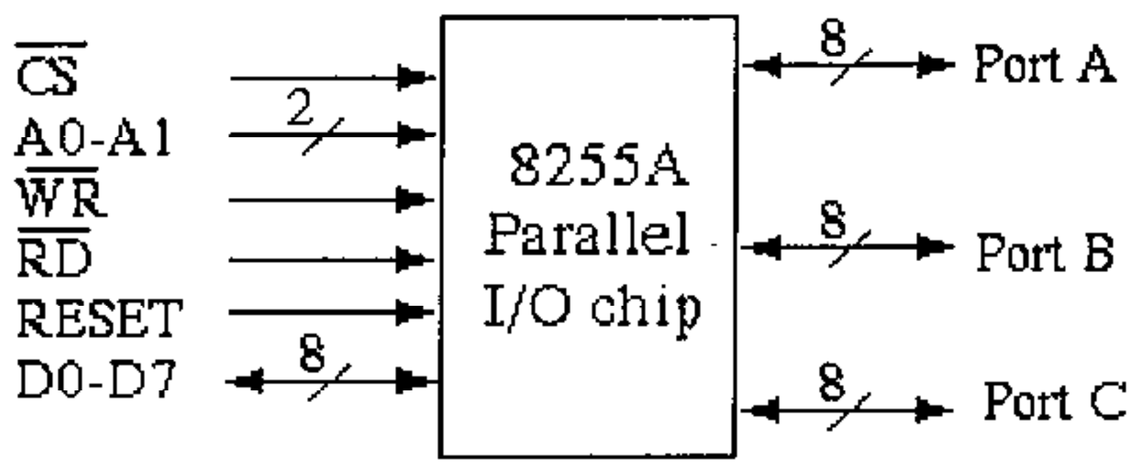
- **USART- Chip Thu-Phát đồng bộ/không đồng bộ**

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Chip USART (Universal Synchronous Asynchronous Receiver Transmitter) có thể quản lý việc truyền tin đồng bộ bằng việc sử dụng nhiều giao thức khác nhau cũng như có thể sử dụng tất cả các chức năng của UART.

- **Các chip PIO (Parallel Input/Output)**

Một chip PIO điển hình là chip 8255A, như thể hiện trên hình 3-43. Nó có 24 đường I/O, có thể ghép nối với mọi thiết bị tương thích TTL, thí dụ bàn phím, các chuyển mạch, đèn hay máy in. Chương trình chạy trong CPU có thể ghi các giá trị 0 hay 1 lên mọi đường hoặc đọc giá trị trên mọi đường, làm cho sự hoạt động của chip này hết sức linh hoạt. Một hệ thống nhỏ dựa trên bộ vi xử lý có sử dụng PIO có thể thay thế cho một bảng mạch chứa đầy các chip SSI, MSI.



Hình 3-43 Chip PIO 8255A

CPU có thể định cấu hình cho 8255A bằng cách nạp giá trị cho các thanh ghi trạng thái bên trong chip, sau đây chúng ta sẽ tập trung tìm hiểu một số mode hoạt động đơn giản.

Mode hoạt động đơn giản nhất

Cách thức đơn giản nhất của việc sử dụng 8255A là sử dụng nó như 3 cổng 8 bit độc lập là A, B và C. Mỗi cổng này được gắn với một thanh ghi chốt 8 bit. Để thiết lập giá trị trên các đường của cổng, CPU chỉ phải ghi một con số 8 bit vào một thanh ghi tương ứng, con số đó sẽ xuất hiện trên các đường dây ra và cứ nằm ở đó cho tới khi thanh ghi được viết lại. Để sử dụng cổng làm lối vào, CPU chỉ việc đọc các thanh ghi tương ứng.

Các mode hoạt động khác

Các mode hoạt động khác có khả năng bắt tay (handshaking) với các thiết bị bên ngoài. Thí dụ, để gửi dữ liệu ra một thiết bị không luôn luôn sẵn sàng nhận dữ liệu, 8255A có thể đặt dữ liệu lên một cổng ra và chờ thiết bị đó gửi trở lại một xung báo rằng nó đã chấp nhận dữ liệu và muốn nhận thêm. Những mạch logic cần thiết để chốt những xung như vậy lại và làm cho CPU có thể sử dụng được chúng đã được đặt ngay bên trong phần cứng của 8255A.

Trên sơ đồ khối chúng ta thấy rằng ngoài 24 chân dùng cho 3 cổng, chip 8255A còn có 8 đường nối trực tiếp với bus dữ liệu, một đường dây chọn chip (CS), một đường chọn đọc (RD), một đường chọn ghi (WR), 2 đường địa chỉ để chọn 1 trong 4

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

thanh ghi bên trong tương ứng với 3 cổng A,B,C và một thanh ghi trạng thái. Thanh ghi này có các bit dùng để xác định cổng nào làm input, cổng nào làm output và một số chức năng khác. Thông thường hai đường dây địa chỉ được nối với các bit thấp nhất của bus địa chỉ.

3.6.2 Việc giải mã địa chỉ

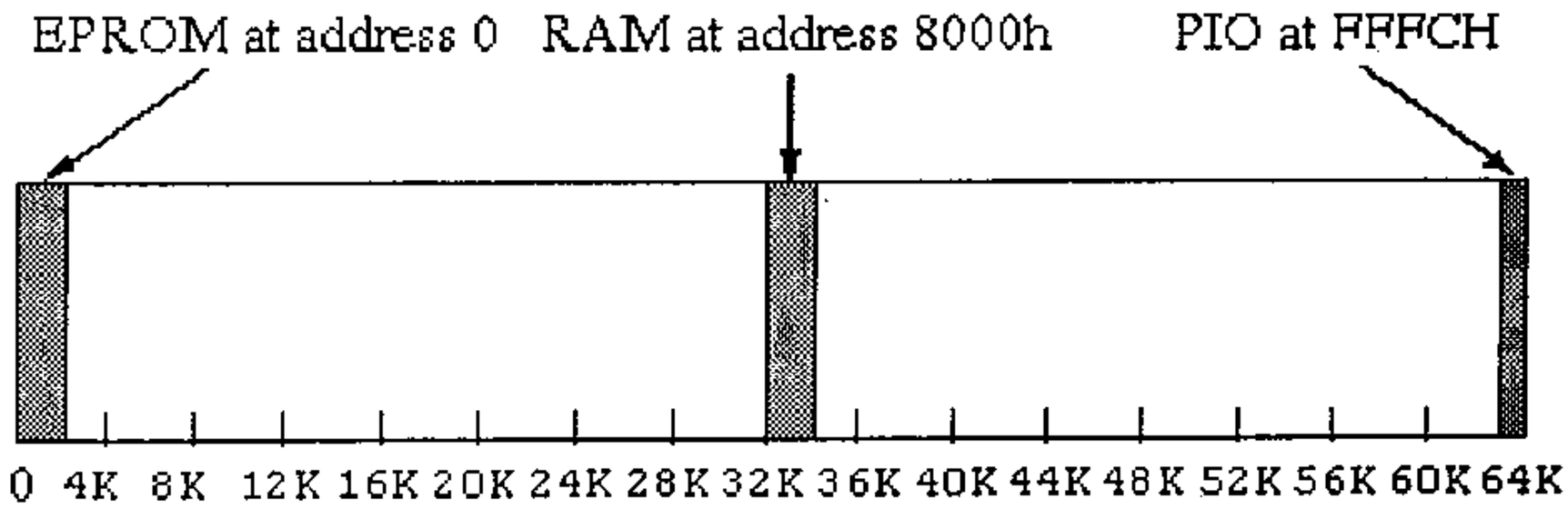
Chúng ta lấy một máy vi tính có một CPU, một chip EPROM dung lượng 2Kx8 chứa chương trình, một chip RAM dung lượng 2Kx8 để chứa dữ liệu và một chip PIO làm thí dụ để nghiên cứu vấn đề giải mã địa chỉ. Một hệ thống nhỏ như vậy có thể là mô hình mẫu cho bộ não của một đồ chơi hoặc thiết bị.

Chip PIO có thể được chọn bằng một trong hai cách, hoặc như một thiết bị vào/ra thực sự hoặc như một phần của bộ nhớ. Nếu ta chọn cách sử dụng chip PIO như một thiết bị vào/ra thì chúng ta bắt buộc phải chọn nó bằng cách sử dụng một địa chỉ cụ thể, được chỉ ra rõ ràng thuộc không gian I/O, thí dụ tín hiệu $\overline{IORQ}$ trong bus IBM PC.

Nếu chúng ta sử dụng cách tiếp cận khác, được gọi là **memory-mapped I/O**, thì chúng ta phải gán cho ba cổng và một thanh ghi điều khiển của nó bốn địa chỉ (địa chỉ là số một byte) thuộc không gian địa chỉ của CPU.

Việc lựa chọn một trong hai cách là tuỳ ý. Chúng ta sẽ chọn cách memory-mapped I/O bởi vì nó minh hoạ một số vấn đề thú vị trong việc ghép nối vào/ra.

EPROM cần 2K của không gian địa chỉ, RAM cũng cần 2K và PIO chỉ cần có 4 địa chỉ (4 byte). Vì không gian địa chỉ trong thí dụ của chúng ta là 64K nên chúng ta phải thực hiện lựa chọn việc đặt 3 thiết bị chiếm những miền địa chỉ nào. Một lựa chọn có thể thực hiện, được thể hiện trên hình 3-44. EPROM chiếm các địa chỉ từ 0 cho đến 2K, RAM chiếm các địa chỉ từ 32K tới 34K và PIO chiếm 4 byte địa chỉ cao nhất của không gian địa chỉ, từ 65532-65535. Dưới góc nhìn của người lập trình, việc sử dụng địa chỉ nào không có gì là quan trọng, tuy nhiên với việc ghép nối thì có vấn đề. Nếu chúng ta đã chọn địa chỉ cho PIO qua không gian I/O tách biệt thì nó không cần địa chỉ bộ nhớ như ở hình vẽ này.



Hình 3-44 Sử dụng phương pháp memory-mapped I/O

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Với việc gán địa chỉ như trên hình 3-44, EPROM phải được chọn bằng các địa chỉ 16 bit, có dạng nhị phân là 0000 0xxx xxxx xxxx. Nói cách khác, mọi địa chỉ 16 bit mà 5 bit bậc cao đều là 0, đều rơi vào miền 2K thấp nhất của bộ nhớ, đó chính là miền của EPROM. Vì vậy để chọn chip EPROM cần sử dụng một bộ so sánh 2 con số 5 bit, một trong hai đầu vào (5 bit) được gắn cố định với giá trị 00000, đầu vào thứ hai là 5 bit cao của địa chỉ 16 bit. Đầu ra của bộ so sánh nối với đầu vào chọn chip của chip EPROM. Khi hai số đầu vào bằng nhau thì đầu ra của bộ so sánh có giá trị tích cực, sẽ kích hoạt chip EPROM.

Có một cách khác để đạt được cùng một kết quả là sử dụng một cổng OR có 5 lối vào, các lối vào này gắn với các đường địa chỉ A11-A15. Chỉ khi cả 5 dây này cùng bằng 0 thì ở lối ra của cổng OR mới có giá trị 0, làm cho dây $\overline{CS}$ có mức tích cực (mức thấp). Vì trong các chip SSI được tiêu chuẩn hoá, không có chip nào chứa cổng OR có 5 lối vào, nên chúng ta có thể chọn loại chip gần đúng yêu cầu nhất có trên thị trường là các cổng NOR có 8 lối vào. Ba lối vào không dùng đến chúng ta nối xuống đất (trên hình không vẽ 3 lối vào này) và đảo tín hiệu ở đầu ra. Làm như vậy chúng ta sẽ nhận được tín hiệu đúng theo yêu cầu, xem hình 3-45a.

Nguyên tắc đó cũng có thể đem áp dụng cho RAM. Tuy nhiên chip RAM cần đáp lại yêu cầu đối với các địa chỉ có dạng 1000 0xxx xxxx xxxx, vì thế cần phải có thêm một mạch đảo nối với dây địa chỉ A_{15} như chúng ta thấy trên hình vẽ 3-45a.

Việc giải mã địa chỉ cho PIO phức tạp hơn, bởi vì nó được chọn bởi 4 địa chỉ có dạng 1111 1111 1111 11xx. Trên hình vẽ chúng ta sử dụng một mạch điện, nó sẽ đặt mức tín hiệu ở chân $\overline{CS}$ của chip PIO tích cực, chỉ khi địa chỉ đúng xuất hiện trên bus địa chỉ. Mạch điện này sử dụng 2 cổng NAND có 8 lối vào, đầu ra của 2 cổng NAND được đưa vào 2 đầu vào của một cổng OR, xem hình 3-45a.

Việc giải mã địa chỉ một phần (Partial address decoding)

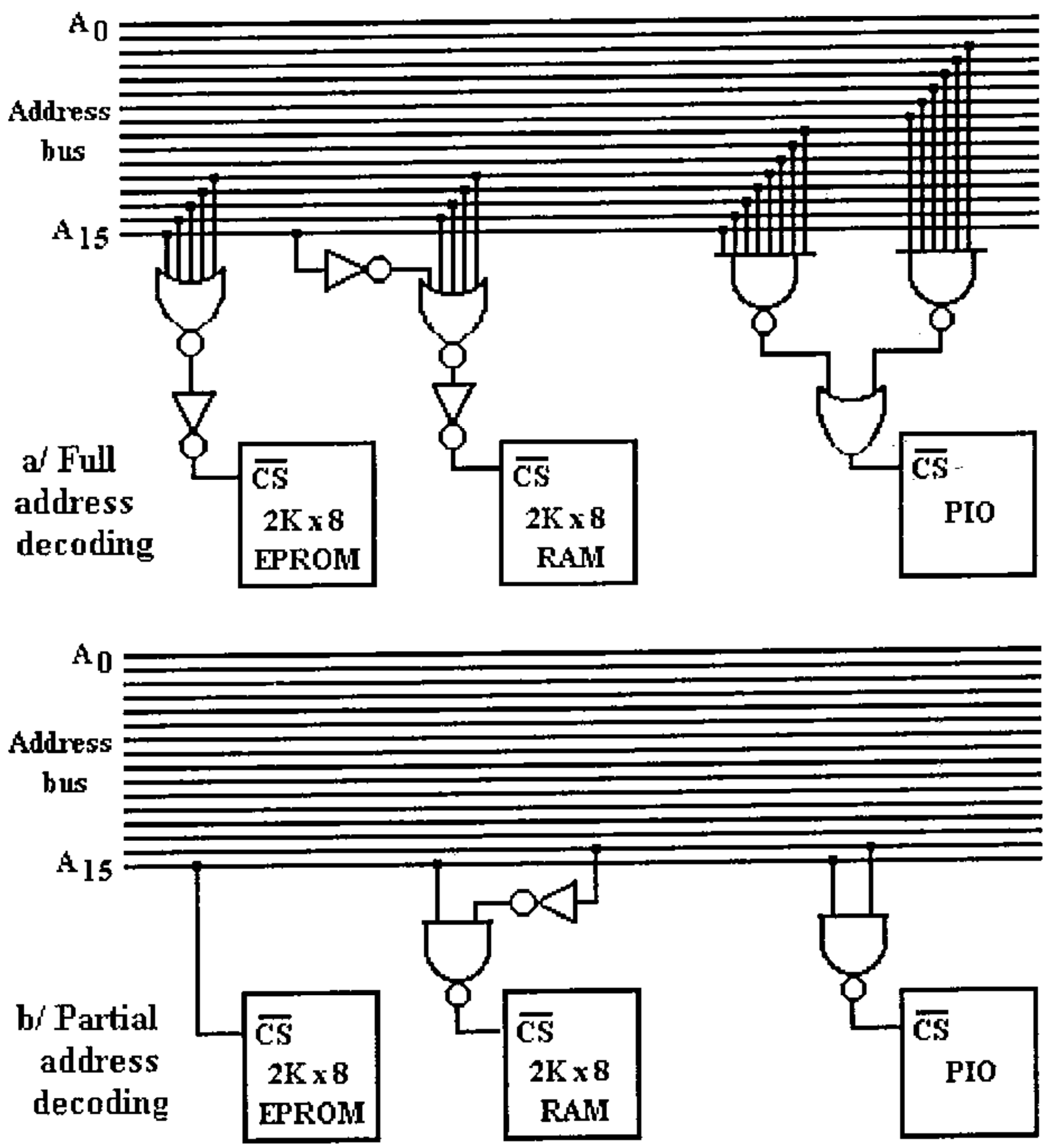
Nếu máy tính thực sự chỉ có một CPU, hai chip nhớ và một chip PIO với dung lượng mô tả như trên, thì chúng ta có thể dùng một "mánh" nhỏ, giúp việc giải mã địa chỉ đơn giản hơn rất nhiều. Việc này dựa trên thực tế là tất cả địa chỉ của EPROM và chỉ có địa chỉ của EPROM mới có bit cao bằng 0 (A15=0). Chính vì vậy chúng ta có thể nối trực tiếp chân $\overline{CS}$ của chip EPROM vào dây địa chỉ A15, như trên hình 3-45b. Sau khi làm như vậy thì việc quyết định đặt RAM tại địa chỉ 8000H không còn được tự do như trước. Để thực hiện việc giải mã cho RAM, chúng ta dựa trên nhận xét là chỉ có các địa chỉ có dạng 10xx xxxx xxxx xxxx là nằm trong RAM, vì vậy chỉ cần 2 bit cho bộ giải mã là đủ. Tương tự như vậy, mọi địa chỉ có 2 bit cao nhất là 11 chắc chắn là địa chỉ PIO. Như vậy toàn bộ mạch giải mã gồm có 2 cổng NAND và 1 cổng đảo.

Mạch giải mã như trên hình 3-45b được gọi là giải mã địa chỉ một phần (Partial address decoding), bởi vì mạch giải mã này không sử dụng hết toàn bộ số dây địa chỉ. Nó có một điểm đặc biệt là việc đọc từ các địa chỉ 0001 0000 0000 0000, 0001 1000 0000 0000 hoặc 0010 0000 0000 0000 sẽ cho cùng một kết quả. Trên thực tế mọi địa chỉ ở nửa dưới của không gian địa chỉ sẽ chọn EPROM. Bởi vì các địa chỉ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

thừa ra không được sử dụng, cho nên không có gì nguy hiểm cả, tuy nhiên nếu chiếc máy tính mà chúng ta đang thiết kế sẽ được mở rộng trong tương lai, đặc biệt là mở rộng bộ nhớ, thì nên tránh giải mã địa chỉ một phần vì chúng ta sẽ phải thay đổi gần như hoàn toàn các mạch giải mã cho RAM, EPROM và PIO.

Một kỹ thuật giải mã địa chỉ thông dụng khác là sử dụng một chip giải mã, như chip trên hình 3-08 (chip giải mã "3 to 8"). Bằng cách nối 3 đầu vào của mạch giải mã với 3 dây địa chỉ cao nhất, chúng ta có 8 lối ra, tương ứng với các địa chỉ trong khối 8K thứ nhất, 8K thứ hai .v.v. Đối với một máy tính có 8 chip RAM loại 8Kx8 chỉ cần dùng một chip giải mã như vậy là có thể giải mã đầy đủ. Đối với một máy tính có 8 chip RAM loại 2Kx8 dùng một chip giải mã như vậy cũng được, miễn là mỗi chip RAM được đặt ở đầu các khối 8K địa chỉ trong không gian địa chỉ.



Hình 3-45 a) Giải mã địa chỉ toàn phần b) Giải mã địa chỉ một phần

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 4 MỨC VI CHƯƠNG TRÌNH

Người ta không xác định thật rõ ràng ranh giới giữa phần cứng và phần mềm, hơn nữa nó cũng không ngừng dịch chuyển. Trong những máy tính đầu tiên tất cả các chỉ thị thực hiện các phép toán số học, logic, dịch, so sánh, lặp.v.v. đều được thực hiện bằng phần cứng. Với mỗi chỉ thị có một mạch điện phần cứng cụ thể thực hiện chỉ thị đó. Về nguyên tắc, người ta có thể tháo ốc, mở nắp máy và chỉ ra những mạch điện tử thực hiện một chỉ thị cụ thể, chẳng hạn chỉ thị chia.

Trong máy tính nhiều mức hiện đại, người ta không thể chỉ ra mạch thực hiện phép chia bởi vì không có mạch thực hiện phép chia. Tất cả các chỉ thị có thể sử dụng được ở mức máy thông thường (thí dụ các chỉ thị thực hiện các phép toán số học, logic, dịch, so sánh, lặp) được thực hiện từng bước một bằng một trình thông dịch chạy ở mức vi chương trình. Việc tìm ra mạch thực hiện phép chia ngày nay tương đương với việc lấy ra danh sách các vi chương trình và tìm phần vi chương trình thực hiện việc dịch chỉ thị chia thành một dãy các thao tác đơn giản hơn.

Mặc dầu mọi chương trình ở mọi mức đều có thể thi hành bằng trình thông dịch và mặc dầu chính trình thông dịch này lại có thể được thi hành bởi một trình thông dịch khác, nhưng trình tự như vậy không thể nào cứ kéo dài mãi. Ở mức dưới cùng nhất thiết phải có một cái máy phần cứng vật lý, chế tạo bởi các vi mạch (IC), nguồn nuôi và các thành phần "cứng" tương tự như thế. Những thành phần "cứng" như vậy là đối tượng mà chúng ta đã nghiên cứu ở chương 3. Chương này nghiên cứu việc các thành phần phần cứng này được điều khiển bởi vi chương trình như thế nào và vi chương trình thông dịch mức máy thông thường như thế nào. Một điều đặc biệt cần nêu ngay ở đây là các máy RISC không có mức vi chương trình.

Bởi vì kiến trúc của mức vi chương trình, còn được gọi là ***vi kiến trúc*** được xác định bởi phần cứng cho nên nó thường rất sơ đẳng và rất khó lập trình, chẳng hạn việc định thời (timing) cho các thao tác phần cứng.

Mức vi chương trình có một chức năng chuyên biệt là: thi hành các trình thông dịch cho các máy ảo khác. Mục đích thiết kế này tự nhiên dẫn tới một sự tổ chức được tối ưu hoá ở mức cao, hướng tới việc lấy chỉ thị, giải mã chỉ thị và thi hành các chỉ thị mức máy thông thường. Trong một số trường hợp thực hiện các công việc trên ngay cả đối với các chỉ thị phức tạp hơn.

Chương này sẽ khảo sát các vấn đề liên quan tới việc tổ chức và thiết kế mức vi chương trình cũng như những sự cân nhắc lựa chọn các phương pháp.

Chúng ta sẽ bắt đầu nghiên cứu mức vi chương trình bằng cách ôn lại tóm tắt các thành phần cơ bản xây dựng nên máy tính đã được nghiên cứu ở chương 3, chúng là một phần của kiến trúc của mức vi chương trình, là mối quan tâm của những người lập vi chương trình. Sau đó sẽ tiến đến nghiên cứu phần chính yếu của chương là các chỉ thị phức tạp hơn có thể được xây dựng nên từ một dãy các chỉ thị sơ cấp như thế nào. Chủ đề này sẽ được thảo luận chi tiết và minh hoạ bằng thí dụ cụ thể. Sau

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

đó chúng ta sẽ khảo sát các yếu tố khác nhau cần phải tính đến khi thiết kế mức vi chương trình của một máy tính, nhằm hiểu tốt hơn tại sao người ta đã làm như vậy. Chúng ta cũng xem các cách nâng cao hiệu suất của máy tính.

4.1 CÁC VẤN ĐỀ CHÍNH CỦA MỨC LOGIC SỐ MÀ NGƯỜI LẬP VI CHƯƠNG TRÌNH QUAN TÂM

Công việc của người lập vi chương trình là viết chương trình điều khiển các thanh ghi, các bus, các ALU, bộ nhớ và các phần tử phần cứng khác của máy. Các thiết bị này đã được nghiên cứu khá chi tiết ở chương trước, ở đầu của chương 4 này chúng ta chỉ ôn lại một cách tóm tắt để có thể hiểu được thực chất công việc của người lập vi chương trình.

4.1.1 Thanh ghi

Thanh ghi là một thiết bị có khả năng chứa thông tin. Mức vi chương trình luôn luôn có một số thanh ghi để giữ các thông tin cần thiết cho việc xử lý chỉ thị hiện đang được thông dịch. Về mặt khái niệm các thanh ghi cũng giống như bộ nhớ chính, sự khác nhau là về mặt vật lý. Các thanh ghi được đặt ngay bên trong bộ xử lý, vì vậy việc đọc/ghi thông tin ở đó nhanh hơn so với đọc/ghi các từ trong bộ nhớ, các từ của bộ nhớ luôn nằm ngoài chip xử lý. Những máy tính lớn và đắt tiền thường có nhiều thanh ghi hơn các máy nhỏ và rẻ, các máy nhỏ và rẻ buộc phải sử dụng bộ nhớ chính để chứa các kết quả trung gian. Trong một số máy tính, một nhóm các thanh ghi được đánh số 0, 1, 2, ..., n-1 có thể sử dụng được ở mức vi chương trình và chúng được gọi là ***bộ nhớ cục bộ hay bộ nhớ nháp (scratchpad storage)***.

Thanh ghi được đặc trưng bằng số bit mà nó có thể chứa. Hình vẽ 4-01 là một thanh ghi 16 bit, với các bit được đánh số thứ tự từ phải qua trái theo thói quen. Quá trình đọc thông tin từ một thanh ghi không làm ảnh hưởng tới nội dung của thanh ghi; Nói một cách khác, khi một thanh ghi bị đọc, có một bản copy nội dung của nó được tạo ra, còn giá trị gốc không bị suy chuyển vẫn để trong thanh ghi.

Bit number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	1	1	0	0	1	0	1	1	0	0	1	0

Hình 4-01 Thanh ghi 16 bit chứa được 16 bit thông tin

4.1.2 Bus

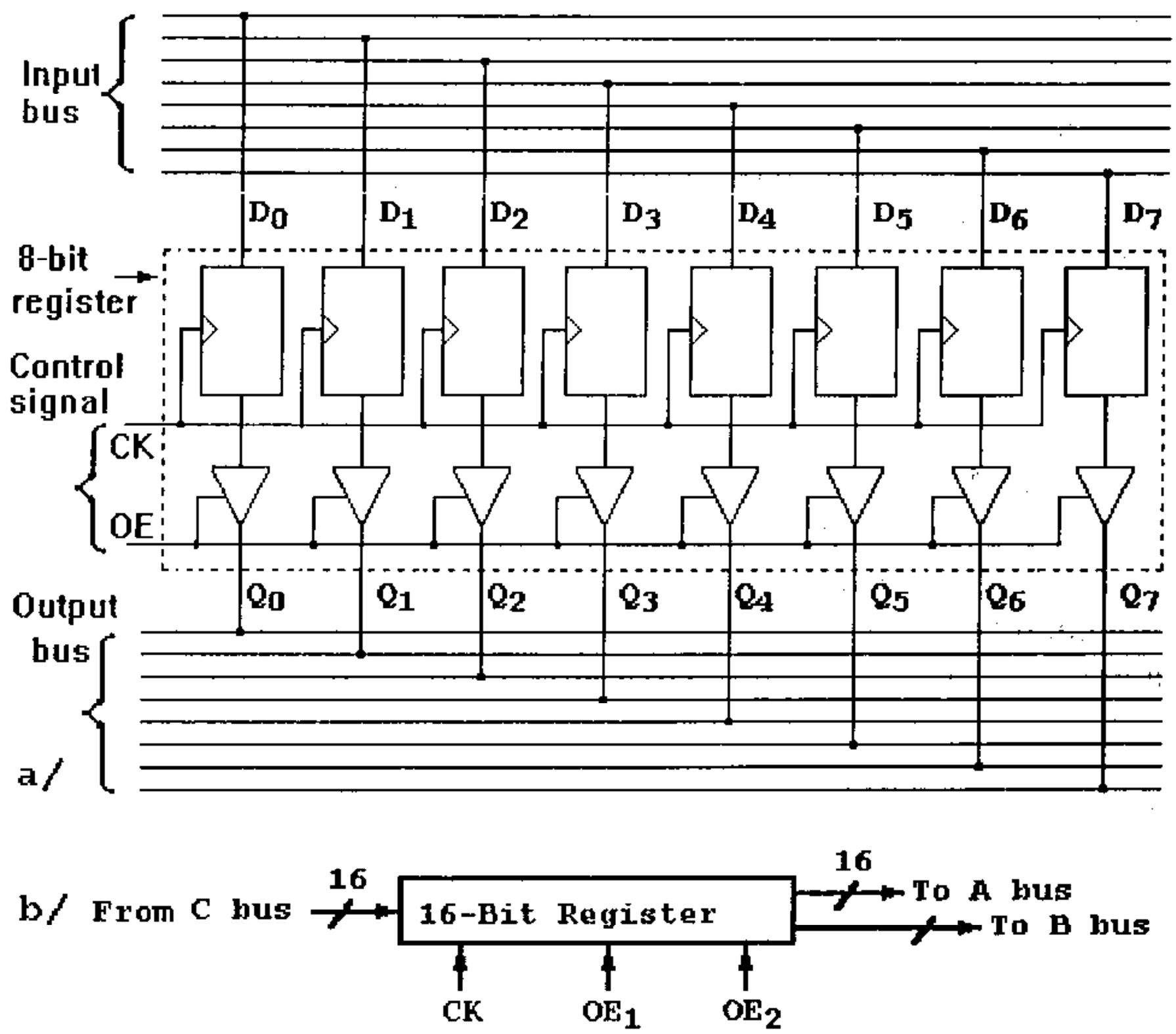
Bus là tập hợp các đường dây được sử dụng để truyền song song các tín hiệu. Thí dụ dùng để copy nội dung thanh ghi này sang một thanh ghi khác. Người ta sử

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

dụng bus vì việc truyền cùng một lúc nhiều bit đi song song nhanh hơn truyền lần lượt từng bit.

Bus có thể là một hoặc hai chiều. Bus một chiều chỉ có thể truyền dữ liệu theo một chiều, trong khi đó bus hai chiều có thể truyền dữ liệu đi theo cả hai chiều, nhưng không thể truyền theo cả hai chiều cùng một lúc được. Bus một chiều thường được sử dụng để nối hai thanh ghi, trong đó một thanh ghi luôn luôn là nguồn còn thanh ghi kia luôn luôn là đích. Bus hai chiều hay được sử dụng khi một thanh ghi bất kỳ trong tập hợp các thanh ghi có thể là nguồn và một thanh ghi bất kỳ khác là đích.

Nhiều thiết bị có khả năng nối và tách chính nó về mặt điện ra khỏi bus mà nó được nối về mặt vật lý, việc nối và cắt có thể được thực hiện trong vài na-nô giây. Bus mà các thiết bị của nó có tính chất này được gọi là **bus 3 trạng thái**, bởi vì mỗi đường dây có thể có giá trị 0 hay 1 hoặc là bị tách ra. Các bus 3 trạng thái thường được sử dụng khi cần nối nhiều thiết bị vào bus, mỗi thiết bị này đều có thể đổ thông tin lên bus.



Hình 4-02 a/ Thanh ghi 8 bit nối với một bus vào (input bus) và một bus ra (output bus). b/ Ký hiệu của một thanh ghi 16 bit với một bus vào và hai bus ra

Trong hầu hết các vi kiến trúc, một số thanh ghi được nối với một hoặc một vài bus vào và một hoặc một vài bus ra. Hình 4-02a mô tả một thanh ghi 8 bit được nối

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

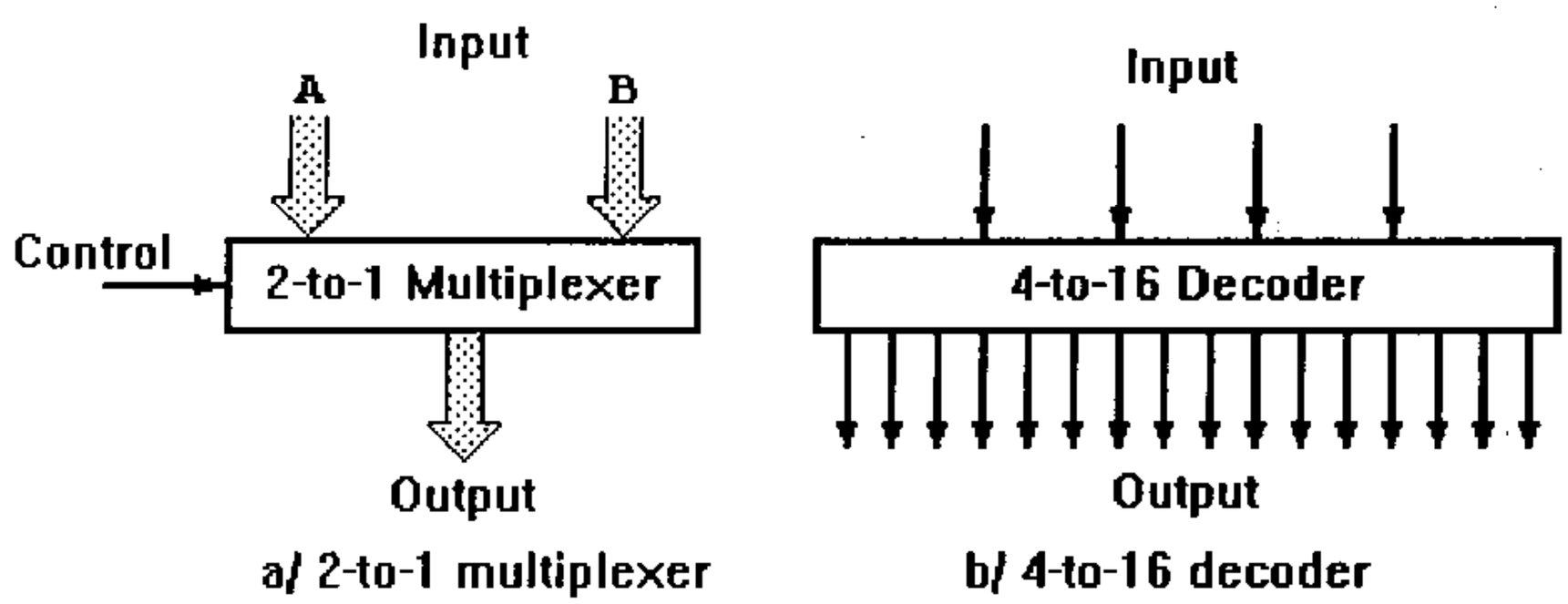
với một bus vào và một bus ra. Thanh ghi này gồm có 8 flip-flop loại D, mỗi flip-flop nối với bus ra qua một bộ đệm, mỗi flip-flop chứa 1 bit thông tin. Thanh ghi có hai tín hiệu điều khiển, CK (ClocK) điều khiển nạp vào thanh ghi và OE (Output Enable) - cho phép thanh ghi đưa dữ liệu ra, cả hai tín hiệu này nối với mọi flip-flop. Lúc bình thường cả hai tín hiệu này nằm ở trạng thái không tích cực, nhưng đôi khi chúng sẽ được đặt mức tích cực, làm xảy ra một hành động tương ứng.

Khi tín hiệu CK không tích cực, nội dung của thanh ghi không bị ảnh hưởng của các tín hiệu trên bus vào (input bus). Khi CK được đặt tích cực, thanh ghi được nạp từ bus vào. Khi tín hiệu OE bị đặt không tích cực, thanh ghi bị tách khỏi bus ra (output bus) và có thể coi là không còn ảnh hưởng gì đối với các thanh ghi khác trên bus. Khi OE được đặt tích cực, nội dung của thanh ghi được đưa lên bus ra.

Nếu một thanh ghi R khác có bus vào nối với bus ra của thanh ghi này, thì có thể truyền thông tin từ thanh ghi này tới thanh ghi R. Để làm việc này, phải đặt $\overline{OE}$ mức tích cực và giữ mức này đủ lâu sao cho tín hiệu mà thanh ghi đưa ra bus ra trở nên ổn định. Sau đó đường dây tín hiệu CK của thanh ghi R phải được đặt tích cực để nạp thông tin trên bus cho R. Thao tác mở cổng một thanh ghi vào một bus để cho một thanh ghi khác có thể nạp giá trị từ bus vào thường xảy ra ở mức vi chương trình, chúng ta sẽ nghiên cứu vấn đề này ở một phần dưới đây. Hình 4-02b là một thí dụ thứ hai về thanh ghi và bus, hình vẽ mô tả một thanh ghi 16 bit có hai bus ra, mỗi bus ra được điều khiển bằng một tín hiệu OE khác nhau.

4.1.3 Bộ dồn/phân kênh và bộ giải mã

Bộ dồn/phân kênh - Multiplexer



Hình 4-03 a/ Bộ dồn kênh "2-to-1"; b/ Bộ giải mã "4-to-16"

Mạch điện có một hoặc nhiều đường dây vào và đưa ra một hay nhiều giá trị hoàn toàn được xác định bởi các giá trị vào hiện thời được gọi là mạch tổ hợp. Hai trong số các mạch tổ hợp quan trọng nhất là mạch multiplexer và decoder. Multiplexer có 2^n đầu vào dữ liệu (mỗi đầu vào này có thể là một đường dây riêng biệt hoặc là một bus), một đầu ra dữ liệu có cùng độ rộng với đầu vào dữ liệu và một đầu vào điều khiển n bit, đầu vào điều khiển sẽ chọn một trong 2^n đầu vào và phân

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

luồng đi cho nó tới đầu ra. Hình 4-03a là một multiplexer có hai bus lối vào. Tín hiệu điều khiển 1 bit (control) chọn bus A hoặc bus B đưa ra đầu ra là Output.

Mạch điện làm công việc ngược lại với multiplexer được gọi là demultiplexer, nó phân luồng đi cho một lối vào để đi ra một trong 2^n lối ra, tuỳ thuộc vào giá trị của n đường dây điều khiển.

Bộ giải mã - Decoder

Một mạch tổ hợp quan trọng khác nữa là decoder, nó có n đường dây lối vào và 2^n đường dây lối ra, được đánh số từ 0..(2^n-1). Nếu số nhị phân trên các đường dây lối vào là k, thì đường dây ra thứ k sẽ có giá trị 1 và tất cả các đường dây còn lại sẽ có giá trị 0. Decoder luôn luôn có duy nhất 1 đường dây ra bằng 1, các dây ra còn lại bằng 0. Hình 4-03b minh hoạ bằng ký hiệu một decoder loại chọn 4 từ 16 (4-to-16), tức là có 4 lối vào và 2^4 = 16 lối ra.

Mạch điện thực hiện công việc ngược lại với decoder là ***encoder***, nó có 2^n đầu vào và n đầu ra. Chỉ có duy nhất một đầu vào được phép nhận giá trị 1 và con số đó được biểu diễn ở lối ra như một số nhị phân.

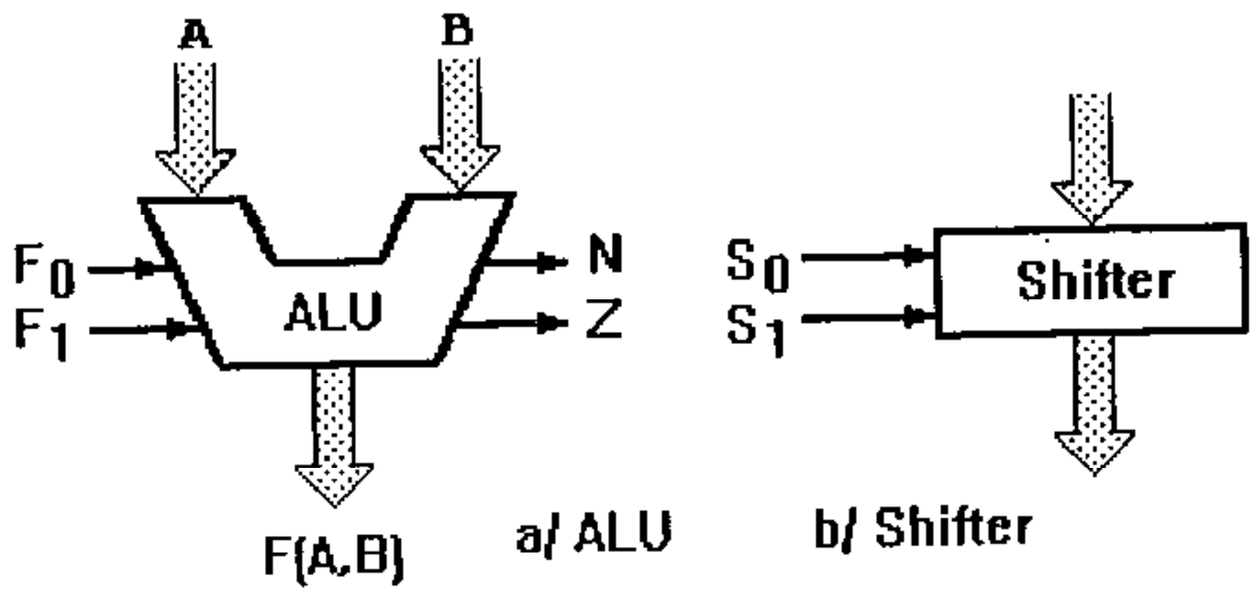
4.1.4 Đơn vị Số học-Logic và Bộ dịch

Mỗi máy tính đều cần thực hiện các phép tính số học bằng một cách nào đấy. Mạch điện đơn giản nhất chính là mạch cộng - adder, nó nhận vào hai giá trị n bit và sinh ra tổng của hai số ở lối ra. Mạch điện số học tổng quát hơn là ALU, nó cũng có hai lối vào dữ liệu và một lối ra dữ liệu nhưng cũng còn có một số lối vào điều khiển và lối ra điều khiển. ALU trên hình 4-04a có hai bit chức năng F_0 và F_1, chúng xác định chức năng mà ALU thực hiện. Chúng ta sẽ nghiên cứu một ALU có thể tính A+B, A AND B, A cũng như $\overline{A}$. Hai chức năng đầu và chức năng cuối không có gì cần bàn thêm, còn chức năng thứ ba, có nghĩa là chỉ copy A ra lối ra. Chức năng này có vẻ như không để làm gì cả, nhưng chúng ta sẽ thấy ứng dụng của nó trong các phần sau. Hình 3-13 ở chương 3 là một ALU có bốn chức năng, nó thao tác trên dữ liệu 1 bit. Bằng cách sử dụng n mạch ALU như vậy, đưa carry-out của ALU cộng hai bit bậc i vào đầu vào carry-in của ALU cộng hai bit bậc i+1, chúng ta có thể tạo ra một ALU rộng n bit thực hiện được các phép tính AND, OR, NOT và cộng trên các toán hạng n bit.

ALU cũng có thể có các đầu ra trạng thái, các đầu ra trạng thái điển hình là các đầu ra bằng 1 khi kết quả phép tính là âm, bằng 0 hoặc có carry-out ở bit bậc cao nhất hoặc khi có sự tràn số. ALU trên hình 4-04a có hai đầu ra trạng thái là: N - chỉ báo rằng kết quả ra của ALU là âm (Negative) và Z - chỉ báo rằng kết quả ra của ALU bằng không (Zero). Thực chất, bit N chỉ là copy của bit có bậc cao nhất của kết quả đưa ra. Bit Z là NOR của tất cả các bit của kết quả đưa ra.

Mặc dầu một số ALU có thể thực hiện thao tác dịch (shift), nhưng từ trước tới nay người ta vẫn thường xây dựng đơn vị thực hiện phép dịch riêng biệt. Mạch điện này có thể dịch một số nhiều bit đưa vào sang trái hoặc phải 1 bit hay là không dịch bit nào cả. Hình 4-04b là ký hiệu của bộ dịch.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

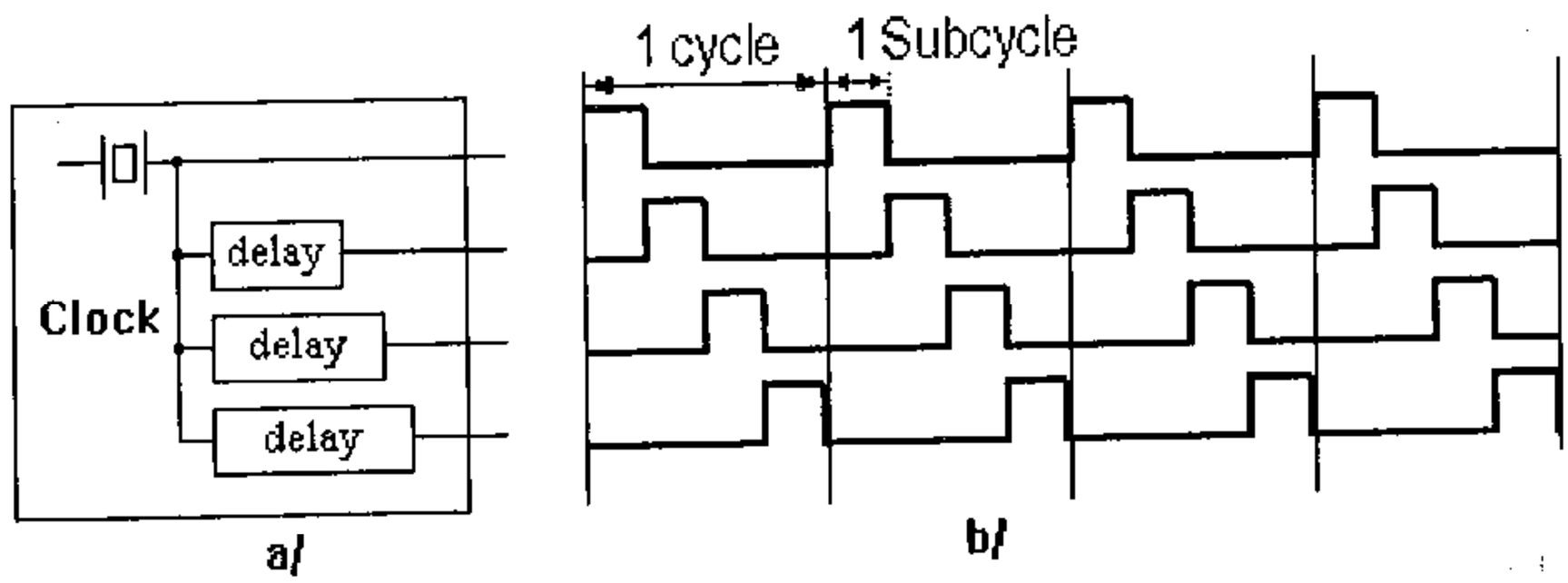


Hình 4-04 a/ Ký hiệu một ALU; b/ Ký hiệu một Shifter

4.1.5 Đồng hồ

Các mạch điện của máy tính thường được điều khiển bằng một đồng hồ, đó là một thiết bị sinh ra một dãy đều đặn các xung điện. Trong mỗi chu kỳ máy có một số hành động xảy ra, chẳng hạn các bước để thực hiện một vi lệnh. Người ta thường phải chia một chu kỳ máy thành nhiều chu kỳ con để cho các bước khác nhau của vi lệnh có thể được thi hành theo một trật tự đã được xác định rõ. Thí dụ các giá trị đưa vào ALU đã phải có và đã ổn định trước khi kết quả đưa ra được cất đi.

Hình 4-05a là ký hiệu một đồng hồ có 4 lối ra. Lối ra trên cùng là lối ra cơ sở, các lối ra còn lại nhận được từ lối ra này bằng cách chèn các bộ trễ có độ trễ khác nhau vào lối ra. Tín hiệu đồng hồ cơ sở, đường trên cùng của hình 4-05b, có độ rộng xung bằng 1/4 chu kỳ. Các tín hiệu ra khác đã được làm trễ một, hai và 3 lần độ rộng xung, kết quả là mạch điện chia mỗi chu kỳ thành 4 chu kỳ con dài bằng nhau.



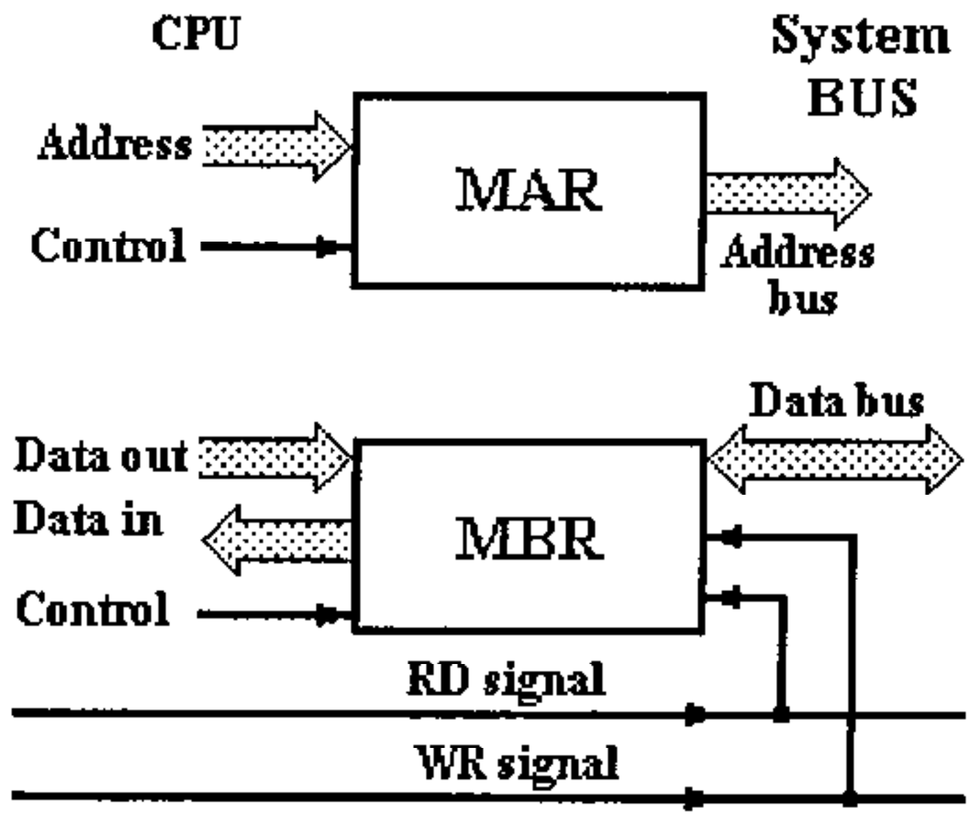
Hình 4-05 a/ Đồng hồ có 4 đầu ra; b/ Tín hiệu trên các đầu ra

4.1.6 Bộ nhớ chính

Các bộ xử lý cần phải có khả năng đọc dữ liệu từ bộ nhớ và ghi dữ liệu vào bộ nhớ. Hầu hết các máy tính có một bus địa chỉ (Address), một bus dữ liệu (Data) và một bus điều khiển (Control) để truyền thông giữa CPU và bộ nhớ.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Để đọc từ bộ nhớ, CPU đặt một địa chỉ lên bus địa chỉ và thiết lập các tín hiệu điều khiển một cách phù hợp, chẳng hạn đặt tín hiệu RD (ReaD - đọc) tích cực, sau đó bộ nhớ đưa dữ liệu mà nó bị yêu cầu lên bus dữ liệu. Trong một số máy tính, việc đọc/ghi bộ nhớ là đồng bộ; nghĩa là bộ nhớ phải đáp ứng được yêu cầu đọc/ghi đối với nó trong một khoảng thời gian cố định. Trong một số máy tính khác, bộ nhớ có thể chậm bao nhiêu cũng được, để thực hiện được việc này phải bổ sung thêm cho bus một đường tín hiệu điều khiển, khi nào bộ nhớ hoàn thành công việc nó sẽ báo cho đơn vị đã có yêu cầu dữ liệu biết.



Hình 4-06 Các thanh ghi điều khiển bus địa chỉ và bus dữ liệu

Việc ghi vào bộ nhớ được thực hiện tương tự. CPU đặt dữ liệu cần ghi vào bộ nhớ lên bus dữ liệu và đặt địa chỉ ô nhớ sẽ chứa dữ liệu lên bus địa chỉ, sau đó đặt dây tín hiệu WR (WRite) lên mức tích cực. (Có trường hợp người ta dùng tín hiệu MREQ thay cho hai tín hiệu RD và WR).

Một thao tác truy cập bộ nhớ (đọc/ghi) nhìn chung thường lâu hơn nhiều thời gian cần thiết để thực hiện một chỉ thị. Hệ quả là vi chương trình phải duy trì các giá trị trên các bus địa chỉ và bus dữ liệu trong suốt thời gian thi hành một số vi lệnh. Để cho công việc này được đơn giản và thuận tiện, người ta thường dùng hai thanh ghi là MAR (Memory Address Register : thanh ghi địa chỉ bộ nhớ) và thanh ghi MBR (Memory Buffer Register : thanh ghi đệm bộ nhớ), chúng sẽ điều khiển bus địa chỉ và bus dữ liệu. Để tiện cho việc nghiên cứu, chúng ta sắp xếp các bus như trên hình 4-06. Cả hai thanh ghi nêu trên nằm giữa CPU và bus hệ thống (System BUS). Bus địa chỉ là một chiều ở cả hai bên thanh ghi MAR, nó được CPU nạp giá trị lên khi tín hiệu control được đặt mức tích cực. Đầu ra của MAR nối với các đường dây của bus địa chỉ hệ thống luôn luôn được đặt tích cực (có trường hợp chỉ được đặt tích cực khi ghi hoặc đọc). Đường dây control của MBR làm cho dữ liệu từ MBR thông qua bus "Data in" nạp vào trong CPU. Đường dây "Data out" luôn luôn là tích cực. Bus dữ liệu của hệ thống là hai chiều, đưa dữ liệu từ MBR ra khi tín hiệu WR tích cực và nạp vào MBR khi tín hiệu RD là tích cực.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

4.2 MỘT VI KIẾN TRÚC LẤY LÀM THÍ DỤ

Tới lúc này chúng ta đã nghiên cứu tất cả các thành phần cơ bản để xây dựng nên mức vi chương trình, đã đến lúc chúng ta xem xét việc chúng được nối với nhau như thế nào. Trong lĩnh vực này có rất ít nguyên lý chung và chúng khác xa nhau, vì vậy chúng ta sẽ nghiên cứu vấn đề thông qua một thí dụ chi tiết.

4.2.1 Đường dữ liệu (Data path)

Đường dữ liệu là một phần của CPU, nó có chứa ALU, các đầu vào và đầu ra. Trên hình 4-07 là một đường dữ liệu của một vi kiến trúc mà chúng ta lấy làm thí dụ; Nó bao gồm 16 thanh ghi 16 bit giống nhau, được chúng ta gắn nhãn PC, AC, SP.v.v. chúng tạo nên một bộ nhớ tạm thời chỉ có thể truy cập được ở mức vi chương trình. Các thanh ghi được gắn nhãn 0, +1, -1 sẽ được sử dụng để chứa các hằng 0, 1 và -1, ý nghĩa của các tên thanh ghi khác sẽ được giải thích sau. Sự thực là thanh ghi 0 không hề được sử dụng trong thí dụ đơn giản của chúng ta, nhưng có thể cần đến nó trong các máy phức tạp hơn, vì vậy chúng ta đã thêm nó vào. Mỗi thanh ghi có thể đưa nội dung của mình ra ngoài, lên một hay cả hai bus bên trong, đó là bus A và bus B. Mỗi thanh ghi có thể được nạp từ một bus trong thứ ba là bus C.

Các bus A và B nạp dữ liệu vào ALU rộng 16 bit, ALU đó có thể thực hiện 4 chức năng: A+B, A AND B, A và NOT A. Hai đường điều khiển F_0 và F_1 định rõ chức năng sẽ được ALU thực hiện. ALU sinh ra hai bit trạng thái dựa trên kết quả ra hiện thời của nó: N có giá trị 1 khi kết quả ra là âm, Z có giá trị 1 khi kết quả ra bằng 0 (Zê-rô).

Đầu ra của ALU đưa vào một thanh ghi dịch - shifter, nó có thể dịch các bit 1 vị trí theo cả hai hướng hoặc không dịch. Có thể thực hiện dịch trái hai bit một thanh ghi R, bằng cách tính R+R trong ALU, sau đó dịch tổng một bit nữa sang trái bằng thanh ghi dịch.

Cả bus A lẫn bus B đều không nạp trực tiếp vào ALU, thay vào đó mỗi bus nạp vào một thanh ghi chốt, đến lượt mình các thanh ghi chốt nạp vào ALU. Cần phải sử dụng các thanh ghi chốt bởi vì ALU là một mạch tổ hợp, nó liên tục tính và đưa ra kết quả theo các giá trị vào hiện thời và theo mã chức năng. Sự tổ chức này có thể gây ra những vấn đề, thí dụ khi đang tính A:=A+B. Vì thanh ghi A đang được chứa kết quả vào nên giá trị trên bus A bắt đầu thay đổi, nó sẽ làm cho đầu ra của ALU và do đó làm cho bus C cũng thay đổi theo. Hệ quả là giá trị sai có thể sẽ được chứa vào trong thanh ghi A. Nói cách khác, trong phép gán A:=A+B, A ở vế phải là giá trị ban đầu của A, không phải là sự trộn lẫn từng bit một của các giá trị mới và cũ. Bằng cách chèn các thanh ghi chốt A latch và B latch vào các bus A và B, chúng ta có thể cho các thanh ghi chốt này chứa và đưa đến hai đầu vào ALU các giá trị ban đầu ở bus A và bus B. Vì thế ALU được bảo vệ không bị thay đổi do các thay đổi trên các bus A và B khi một giá trị mới đang được chứa vào bộ nhớ tạm (là tập 16 thanh ghi ở phía trên của hình). Các tín hiệu L_0 và L_1 điều khiển việc nạp giá trị trên bus A và bus B vào các thanh ghi chốt.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Để truyền thông với bộ nhớ chúng ta đã bổ sung thêm hai thanh ghi MAR và MBR trong vi kiến trúc. MAR có thể được nạp từ thanh ghi chốt B song song với một thao tác của ALU. Đường dây M_0 điều khiển việc nạp của MAR.

Trong các thao tác ghi vào bộ nhớ chính, MBR được nạp giá trị từ đầu ra của thanh ghi dịch, giá trị này cũng có thể đồng thời được chứa vào một trong các thanh ghi của bộ nhớ tạm. M_1 điều khiển việc nạp của MBR từ đầu ra của thanh ghi dịch, M_2 và M_3 điều khiển việc đọc và ghi bộ nhớ. Khi đọc, dữ liệu đọc được từ bộ nhớ qua MBR có thể có mặt ở đầu vào bên trái của ALU thông qua một bộ dồn kênh, trên sơ đồ ghi là Amux. Đường dây điều khiển A_0 xác định sẽ nạp vào ALU từ thanh ghi chốt A latch hay từ MBR.

Vi kiến trúc trên hình vẽ 4-07 tương tự vi kiến trúc của một số chip đã từng được bán trên thị trường.

4.2.2 Vi chỉ thị - microinstruction

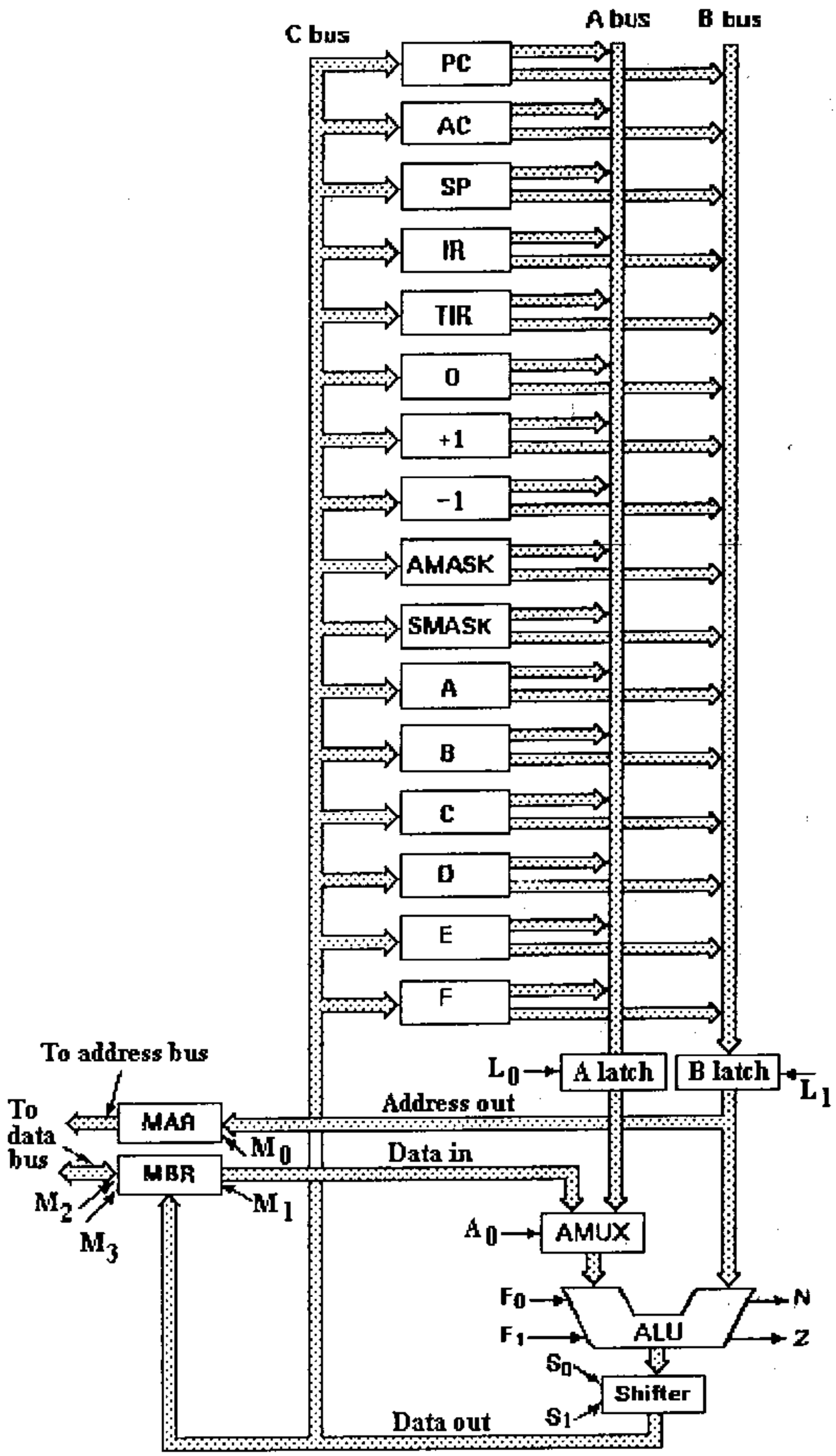
Để điều khiển đường dữ liệu trên hình 4-07 chúng ta cần 61 tín hiệu, chúng có thể được chia thành 9 nhóm chức năng như mô tả dưới đây.

- 16 tín hiệu để điều khiển việc nạp cho bus A từ bộ nhớ tạm (có 16 thanh ghi).
- 16 tín hiệu để điều khiển việc nạp cho bus B từ bộ nhớ tạm.
- 16 tín hiệu để điều khiển việc nạp cho bộ nhớ tạm từ bus C.
- 2 tín hiệu để điều khiển các thanh ghi chốt A và B.
- 2 tín hiệu để điều khiển chức năng ALU.
- 2 tín hiệu để điều khiển bộ dịch.
- 4 tín hiệu để điều khiển MAR và MBR (M_0, M_1, M_2,M_3).
- 2 tín hiệu để chỉ rõ việc đọc bộ nhớ và ghi vào bộ nhớ.
- 1 tín hiệu để điều khiển Amux.

Khi đã có các giá trị của 61 tín hiệu, chúng ta có thể thực hiện một chu kỳ của đường dữ liệu. Một chu kỳ bao gồm việc mở cổng cho các giá trị trong bộ nhớ tạm đi vào bus A và bus B, chốt chúng lại trong hai thanh ghi chốt bus A latch và B latch, cho các giá trị từ các thanh ghi chốt chạy qua ALU và shifter, cuối cùng là việc chứa các kết quả vào trong bộ nhớ tạm hoặc vào MBR. Ngoài ra MAR cũng có thể được nạp, sau đó một chu kỳ bộ nhớ sẽ bắt đầu.

Chúng ta sẽ lấy một thí dụ gần đúng: sử dụng một thanh ghi điều khiển 61 bit, mỗi bit dùng cho một tín hiệu điều khiển. Bit bằng 1 có nghĩa là tín hiệu được đặt tích cực, còn bằng 0 nghĩa là không tích cực. Tuy nhiên nếu chúng ta chấp nhận trả giá cho việc bổ sung thêm một số mạch điện, thì có thể giảm đi được nhiều bit mà chúng ta cần dùng để điều khiển đường dữ liệu. Chúng ta bắt đầu xem xét từ 16 bit để điều khiển việc đưa dữ liệu vào bus A, với 16 bit này có thể chọn một trong 2^{16} tổ hợp các thanh ghi nguồn, nhưng chỉ có 16 trong số các tổ hợp này là được phép, vì chúng ta cũng chỉ có 16 thanh ghi. Chính vì vậy chúng ta có thể lập mã (encode) thông tin điều khiển bus A chỉ bằng 4 bit và dùng một bộ giải mã để sinh ra 16 tín hiệu điều khiển. Đối với bus B cũng như vậy.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 4-07 Đường dữ liệu của một vi kiến trúc lấy làm thí dụ

Đối với bus C tình hình có hơi khác. Về nguyên tắc có thể chứa giá trị trên bus C đồng thời vào nhiều thanh ghi của bộ nhớ tạm, nhưng trong thực tế đặc điểm này không đem lại lợi ích gì và vì vậy hầu hết các phần cứng không hỗ trợ việc đó. Vì thế chúng ta sẽ mã hoá tín hiệu điều khiển bus C chỉ bằng 4 bit.

Như vậy cho tới nay chúng ta đã tiết kiệm được 3x12bit = 36 bit, bây giờ chúng ta chỉ còn cần 25 bit (61-36=25) điều khiển để vận hành đường dữ liệu. Hai tín hiệu

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

L_0 và L_1 luôn luôn cần đến ở những thời điểm nhất định, vì vậy chúng sẽ được một đồng hồ cung cấp, chúng ta bớt đi được hai bit điều khiển, còn lại 23 bit. Có một tín hiệu điều khiển nữa không bắt buộc phải có nhưng thường là hữu ích, đó là tín hiệu cho phép hoặc không cho phép chứa giá trị trên bus C vào bộ nhớ tạm. Trong một số tình huống người ta chỉ muốn thực hiện một thao tác ALU để sinh ra các tín hiệu trạng thái N và Z chứ không cần lưu trữ kết quả. Với bit thêm vào này, ta đặt tên là ENC (ENable C), chúng ta có thể chỉ ra rằng có cất giá trị trên bus C (cho ENC=1) hay không cất (cho ENC=0). Như vậy tới thời điểm này chúng ta có thể điều khiển đường dữ liệu bằng một con số 24 bit.

Với nhận xét là tín hiệu RD có thể dùng để điều khiển việc chốt MBR tách nó khỏi bus dữ liệu hệ thống và WR để nối MBR với bus này, chúng ta có thể không cần sử dụng tín hiệu M_2 và M_3 nữa, làm giảm số tín hiệu điều khiển độc lập từ 24 xuống còn 22 (RD và WR thuộc nhóm 4 tín hiệu để điều khiển MAR và MBR, tham khảo hình 4-06).

AMUX	Điều khiển lối vào bên trái của ALU: 0 = A latch, 1 = MBR
ALU	Chức năng ALU: 0 = A+B, 1 = A AND B, 2 = A, 3 = $\overline{A}$
SH	Chức năng thanh ghi dịch: 0 = no shift, 1 = right, 2 = left
MBR	Nạp cho MBR từ thanh ghi dịch: 0 = không nạp, 1 = nạp
MAR	Nạp cho MAR từ thanh ghi chốt B: 0 = không nạp, 1 = nạp
RD	Yêu cầu đọc bộ nhớ: 0=không đọc, 1=nạp cho MBR từ bộ nhớ
WR	Yêu cầu ghi bộ nhớ: 0 = không ghi, 1 = ghi MBR vào bộ nhớ
ENC	Điều khiển việc chứa vào bộ nhớ tạm: 0=không chứa, 1=chứa
C	Chọn thanh ghi để chứa vào (nếu ENC = 1): 0 = PC, 1 = AC, .v.v.
B	Chọn nguồn cho bus B: 0 = PC, 1 = AC, .v.v.
A	Chọn nguồn cho bus A: 0 = PC, 1 = AC, .v.v.

Bước kế tiếp trong việc thiết kế vi kiến trúc là tạo ra một khuôn dạng vi chỉ thị chứa 22 bit. Hình 4-08 là một khuôn dạng vi chỉ thị như vậy, nó bao gồm 13 trường, trong đó có hai trường mới được bổ sung thêm là COND và ADDR, 11 trong số đó sẽ được mô tả ngay dưới đây.

Trật tự của các trường là hoàn toàn tuỳ ý, trật tự mà chúng ta đã chọn trên hình vẽ 4-08 là để giảm thiểu sự cắt nhau của các đường dây ở các hình vẽ sau này.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Bits	1	2	2	2	1	1	1	1	1	4	4	4	8
	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR

AMUX	COND	ALU	SH	MBR,MAR,RD, WR,ENC
0 = A latch 1 = MBR	0= No jump 1= Jump if N=1 2= Jump if Z=1 3= Jump always	0= A+B 1= A AND B 2= A 3 = $\overline{A}$	0=No shift 1=Shift right 1 bit 2=Shift left 1 bit 3=(not used)	0 = No 1 = Yes

Hình 4-08 Diễn giải một vi chỉ thị điều khiển đường dữ liệu

4.2.3 Việc định thời vi chỉ thị

Mặc dù chúng ta đã thảo luận gần như đầy đủ về việc vi chỉ thị có thể điều khiển đường dữ liệu như thế nào trong một chu kỳ, cho tới điểm này chúng ta vẫn còn bỏ qua một vấn đề là định thời, đó là việc định trình tự xảy ra các sự kiện trong quá trình thực hiện một vi chỉ thị.

Một chu kỳ ALU cơ sở bao gồm việc thiết lập giá trị cho các thanh ghi chốt A và B, dành thời gian cho ALU và bộ dịch - shifter thực hiện công việc của nó và cất các kết quả. Trình tự như vậy là rất hiển nhiên; Nếu chúng ta cứ chứa giá trị trên bus C vào bộ nhớ tạm trước khi các thanh ghi chốt A và B được nạp xong, thì chỉ có rác được chứa chứ không phải là các dữ liệu hữu ích.

Để đạt được việc định đúng trình tự các sự kiện, bây giờ chúng ta đưa ra một đồng hồ 4 pha, tức là một đồng hồ có 4 chu kỳ con, như trên hình vẽ 4-05. Các sự kiện chủ yếu xảy ra trong mỗi một trong số 4 chu kỳ con như sau:

1. Nạp vi chỉ thị tiếp theo sẽ được thi hành vào một thanh ghi có tên là MIR - MicroInstruction Register.
2. Mở cổng các thanh ghi tạm đi vào bus A và bus B và giữ chúng trong các thanh ghi chốt A và B (A latch và B latch).
3. Khi các giá trị đưa vào ALU ổn định, dành thời gian cho ALU và shifter để chúng sinh ra giá trị ra ổn định; nạp MAR nếu cần thiết.
4. Khi giá trị ra của shifter ổn định, chứa giá trị trên bus C vào bộ nhớ tạm và nạp cho MBR nếu việc này cũng được yêu cầu.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Hình 4-09 là sơ đồ khối chi tiết của vi kiến trúc đầy đủ của chiếc máy mà chúng ta lấy làm thí dụ. Thoạt nhìn nó có vẻ thật đồ sộ nhưng rất đáng để chúng ta nghiên cứu tỉ mỉ. ***Khi hiểu đầy đủ về từng khối và từng đường dây trên sơ đồ chúng ta sẽ bước vững chắc trên con đường nghiên cứu mức vi chương trình của mình.*** Sơ đồ khối có hai phần, đường dữ liệu phía bên trái đã được chúng ta đã bàn luận chi tiết và phần điều khiển phía bên phải sẽ được xem xét tại đây.

Phần tử lớn nhất và quan trọng nhất trong phần điều khiển của máy tính là ***bộ nhớ điều khiển - Control store***. Bộ nhớ tốc độ cao và đặc biệt này là nơi giữ các vi chỉ thị. Trong một số máy đó là bộ nhớ chỉ đọc và trong một số máy khác lại là bộ nhớ đọc/ghi. Trong thí dụ của chúng ta các vi chỉ thị có độ rộng 32 bit (22 bit đã được mô tả ở phần trên, ngoài ra còn có thêm trường ADDR rộng 8 bit và trường COND rộng 2 bit) và không gian địa chỉ vi chỉ thị bao gồm 256 từ, vì thế bộ nhớ điều khiển sẽ chiếm tối đa là 256x32 = 8192 bit.

Bộ nhớ điều khiển và bộ nhớ chính là hoàn toàn khác nhau, bộ nhớ điều khiển chứa vi chương trình còn bộ nhớ chính chứa chương trình ngôn ngữ máy thông thường. Giống như mọi bộ nhớ khác, bộ nhớ điều khiển cần một MAR và một MBR. Chúng ta sẽ gọi MAR của bộ nhớ điều khiển là MPC (MicroProgram Counter: Con đếm hay Con trỏ vi chương trình), bởi vì chức năng duy nhất của nó là trỏ đến vi chỉ thị tiếp theo sẽ được thực hiện còn MBR của bộ nhớ điều khiển là MIR như đã nhắc đến ở trên.

- Chu kỳ con thứ nhất:

Từ hình vẽ 4-09 ta có thể thấy rõ rằng bộ nhớ điều khiển không ngừng truyền các bit của vi chỉ thị đã được định địa chỉ bởi MPC vào MIR. Tuy nhiên MIR chỉ được nạp trong chu kỳ con thứ nhất, như đã được chỉ ra bằng đường đứt nét nối từ đồng hồ tới nó. Trong khoảng thời gian 3 chu kỳ con khác nó không bị ảnh hưởng, dù có điều gì xảy ra với MPC đi nữa.

- Chu kỳ con thứ hai:

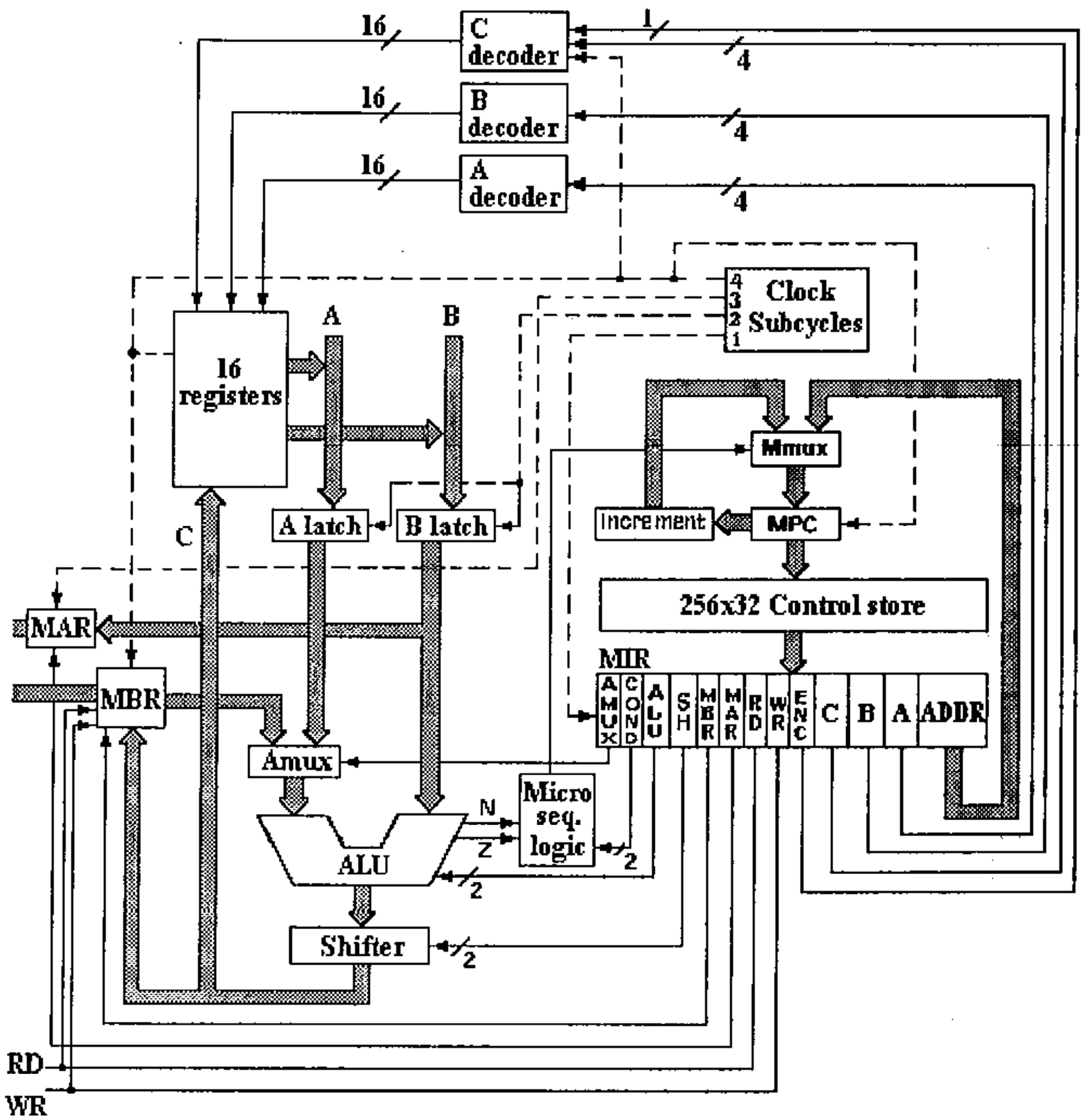
Trong khoảng thời gian của chu kỳ con thứ hai, MIR không thay đổi và các trường khác nhau của MIR bắt đầu điều khiển đường dữ liệu.

Các khối có ghi A decoder và B decoder trên sơ đồ khối làm công việc giải mã 4 ra 16 đường ("4-to-16"), với 4 đường dây vào mỗi khối được nối với 4 bit của từng trường A hoặc B trong MIR. Một trong 16 đầu ra của mỗi decoder này có mức tích cực, tín hiệu đó sẽ điều khiển các đường dây OE1 và OE2 trong các thanh ghi nối với chúng (xem hình 4-02b). Hai trong số 16 thanh ghi tạm được các bộ giải mã A decoder và B decoder chọn sẽ mở cổng ra, đưa nội dung của chúng lên các bus A và bus B.

Đồng hồ kích hoạt các thanh ghi chốt A latch và B latch trong khoảng thời gian chu kỳ con này, chúng sẽ chốt dữ liệu trên bus A và bus B và cung cấp tín hiệu vào ALU ổn định trong thời gian còn lại của chu kỳ. Trong khi dữ liệu đang được mở cổng đi vào các bus A và bus B thì đơn vị Increment trong bộ phận điều khiển tính

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

MPC+1, để chuẩn bị cho việc nạp vi chỉ thị theo trình tự tiếp theo trong chu kỳ tiếp theo. Việc gối đầu hai thao tác này, có thể làm tăng tốc việc thực hiện chỉ thị.



Hình 4-09 Sơ đồ khối đầy đủ của một vi kiến trúc

- Chu kỳ con thứ ba:

Trong chu kỳ con thứ ba ALU và shifter được dành thời gian để sinh ra kết quả đúng. Trường Amux của vi chỉ thị xác định giá trị vào nhánh trái của ALU; còn giá trị vào nhánh phải luôn luôn là thanh ghi chốt B. Mặc dầu ALU là một mạch tổ hợp, thời gian mà nó cần để tính tổng được xác định bởi thời gian lan truyền carry, chứ không phải là độ trễ của một cổng thông thường. Thời gian lan truyền carry tỉ lệ với số bit trong từ. Trong khi ALU và shifter đang tính toán thì MAR được nạp từ bus B nếu trường MAR trong vi chỉ thị bằng 1.

- Chu kỳ con thứ tư:

Trong thời gian chu kỳ con thứ tư và là chu kỳ con cuối cùng, bus C có thể được chứa ngược trở lại vào bộ nhớ tạm và MBR, tuỳ thuộc vào các trường ENC và MBR

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

của vi chỉ thị. Khối có ghi nhãn "C decoder" nhận ở đầu vào các tín hiệu ENC, đường tín hiệu đồng hồ thứ tư và trường C của vi chỉ thị, nó sinh ra 16 tín hiệu điều khiển. Bên trong nó thực hiện giải mã "4-to-16" của trường C của MIR, sau đó thực hiện AND từng tín hiệu này với một tín hiệu nhận được bằng cách AND tín hiệu trên đường dây chu kỳ con thứ tư với ENC. Chính vì thế một thanh ghi của bộ nhớ tạm chỉ được nạp từ bus C nếu 3 điều kiện sau đây được đồng thời thoả mãn:

1. ENC = 1.
2. Đang trong chu kỳ con thứ 4.
3. Thanh ghi đã được chọn bởi trường C của vi chỉ thị.

MBR cũng được nạp trong chu kỳ con thứ tư nếu trường MBR=1.

Hai tín hiệu điều khiển bộ nhớ RD và WR được đặt tích cực khi các trường tương ứng RD và WR trong MIR là tích cực. Hệ quả là các trường tương ứng của MIR thể hiện như các thanh ghi chốt: giá trị mà nó nhớ luôn được đưa ra, còn đầu vào bị chốt lại, làm cho nội dung mà nó nhớ không thay đổi.

4.2.4 Sự định trình tự các vi chỉ thị

Chỉ còn lại một vấn đề là vi chỉ thị tiếp theo sẽ được chọn như thế nào. Mặc dầu đôi khi chỉ cần lấy vi chỉ thị kế tiếp theo thứ tự về là đủ, nhưng cũng cần một cơ chế cho phép thực hiện việc nhảy có điều kiện trong vi chương trình. Chính vì lý do này mà trong mỗi vi chỉ thị chúng ta đã đưa vào thêm hai trường: ADDR là địa chỉ của vi chỉ thị có thể sẽ được thực hiện tiếp sau vi chỉ thị hiện thời và trường COND, nó xác định sẽ lấy vi chỉ thị tiếp theo từ địa chỉ MPC+1 hay từ địa chỉ ADDR trong bộ nhớ điều khiển.

Mọi vi chỉ thị đều có thể bao gồm cả việc nhảy có điều kiện. Người ta làm như vậy bởi vì nhảy có điều kiện rất thường gặp trong các vi chương trình và điều đó cho phép mọi vi chỉ thị đều có hai vi chỉ thị kế tiếp để lựa chọn thực hiện. Việc nhảy có điều kiện sẽ nhanh hơn việc thiết lập một số điều kiện trong một vi chỉ thị rồi thử nó trong vi chỉ thị tiếp theo. Hầu hết các vi kiến trúc hiện nay sử dụng chiến lược mà chúng ta vừa nêu dưới dạng này hay dạng khác.

Việc lựa chọn vi chỉ thị tiếp theo được thực hiện bởi khối có ghi nhãn "Micro sequencing logic", trong chu kỳ con thứ tư, khi các tín hiệu ra N và Z của ALU đang có tác dụng. Tín hiệu ra của khối này điều khiển multiplexer Mmux, nó chọn hoặc MPC+1 hoặc là ADDR cho đi ra MPC, tín hiệu tại đầu ra MPC sẽ điều khiển việc lấy vi chỉ thị tiếp theo. Người lập vi chương trình có 4 cách lựa chọn vi chỉ thị tiếp theo, để chỉ ra lựa chọn cần thiết, ta có thể thiết lập trường COND (trường này rộng 2 bit) như sau:

0 = Không nhảy, lấy vi chỉ thị tiếp theo từ MPC+1

1 = Nhảy tới ADDR nếu N =1

2 = Nhảy tới ADDR nếu Z =1

3 = Nhảy tới ADDR vô điều kiện.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Đơn vị Micro sequencing logic kết hợp 2 bit N và Z của ALU và 2 bit L (Left - trái) và bit R (Right - phải) của trường COND để sinh ra kết quả ra. Tín hiệu này là:

$$\text{Mmux} = \overline{L}RN + L\overline{R}Z + LR = RN + LZ + LR$$

Trong đó dấu + là dấu phép toán OR. Có thể diễn đạt bằng lời như sau: tín hiệu điều khiển đưa đến Mmux bằng 1 (chọn đường cho ADDR đi tới MPC) nếu LR = 01_2 và N=1, hoặc LR = 10_2 và Z=1, hoặc LR = 11_2. Ngược lại thì nó sẽ bằng 0 và vi chỉ thị kế tiếp theo thứ tự sẽ được lấy về. Mạch điện để tính tín hiệu Mmux có thể được xây dựng từ các chip PLA như trên hình 3-10.

Để chiếc máy tính thí dụ của chúng ta giống với máy thật, chúng ta sẽ giả thiết rằng chu kỳ bộ nhớ chính dài hơn chu kỳ của vi chỉ thị. Cụ thể là nếu một vi chỉ thị bắt đầu việc đọc bộ nhớ chính bằng cách đặt RD=1, thì khi vi chỉ thị tiếp theo được thực hiện cũng vẫn phải có RD=1. Dữ liệu sẽ có sau khi thực hiện hai vi chỉ thị kể từ lúc bắt đầu đọc. Nếu vi chương trình không có việc hữu ích gì khác để làm tiếp sau một vi chỉ thị cần đọc bộ nhớ thì chúng ta sẽ bị lãng phí thời gian "ngồi chờ" dữ liệu của bộ nhớ. Tương tự như vậy việc ghi bộ nhớ cũng chiếm thời gian bằng hai vi chỉ thị không có truy cập bộ nhớ ngoài.

4.3 THÍ DỤ VỀ MỘT KIẾN TRÚC MỨC MÁY THÔNG THƯỜNG

4.3.1 Kiến trúc mức máy thông thường - Vĩ kiến trúc

Để nghiên cứu tiếp thí dụ về mức vi chương trình, bây giờ chúng ta cần chuyển sang nghiên cứu kiến trúc của mức máy thông thường, mức này sẽ được hỗ trợ bởi một trình thông dịch chạy trên máy mức 1 như ở hình 4-09. Để cho tiện chúng ta sẽ gọi kiến trúc của các máy ở mức 2 và 3 là **vĩ kiến trúc - macroarchitecture** cho tương phản với kiến trúc ở mức 1 đã được gọi là vi kiến trúc - microarchitecture. Tương tự như vậy chúng ta sẽ gọi các chỉ thị của mức 2 là **vĩ chỉ thị - macroinstruction.**

Với lý do vừa nêu, trong chương này các chỉ thị bình thường như ADD, MOVE và các chỉ thị khác của mức máy thông thường sẽ được gọi là vĩ chỉ thị, tuy nhiên thuật ngữ này không liên quan gì đến thuật ngữ "macro" trong một số ngôn ngữ assembly. Ngoài ra thỉnh thoảng chúng ta cũng sẽ gọi máy mức 1 trong là Mic-1, máy mức 2 là Mac-1.

4.3.2 Vấn đề đánh địa chỉ bộ nhớ chính và bộ nhớ ngăn xếp

Một vĩ kiến trúc hiện đại cần phải được thiết kế nhằm phục vụ yêu cầu của các ngôn ngữ lập trình bậc cao. Một trong những vấn đề thiết kế quan trọng nhất ở mức 2 là việc đánh địa chỉ.

- **Bộ nhớ ngăn xếp - stack và các thao tác trên stack**

Những ngôn ngữ có cấu trúc kiểu khối như Pascal thường được cài đặt sao cho khi một thủ tục hoặc hàm kết thúc, thì bộ nhớ đã được sử dụng cho các biến cục bộ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

được giải phóng. Cách dễ nhất để đạt được mục tiêu này là sử dụng một cấu trúc dữ liệu có tên gọi là **stack**, đó là một khối bộ nhớ nằm liền nhau chứa dữ liệu, được tổ chức theo kiểu vào sau - ra trước (LIFO = Last In First Out). Chúng ta sẽ sử dụng một **con trỏ stack** - SP (Stack Pointer) để cho biết đâu là đỉnh của stack (đáy của stack nằm ở một địa chỉ cố định và chúng ta không quan tâm đến nó).

Người ta thường định nghĩa một số thao tác trên stack. Hai thao tác quan trọng nhất là PUSH và POP. PUSH X làm dịch con trỏ stack và sau đó đặt X vào bộ nhớ tại vị trí lúc này được trỏ bởi SP. PUSH X sẽ làm tăng stack lên bằng kích thước phần tử X. POP Y thì trái lại sẽ làm giảm kích thước của stack do chứa phần tử ở trên cùng của stack vào Y sau đó loại bỏ phần tử này bằng cách tăng con trỏ stack một lượng bằng kích thước của phần tử được lấy ra.

Đối với stack, ngoài hai thao tác trên, thường cần đến một thao tác khác đó là làm dịch chuyển con trỏ stack mà thực tế không đẩy một dữ liệu nào lên stack cả. Việc này thường được thực hiện để dành sẵn chỗ cho các biến cục bộ khi một thủ tục hoặc hàm được đưa vào.

- **Ba phương thức đánh địa chỉ bộ nhớ chính**

Giả sử bộ nhớ chính có dung lượng $2^{12} = 4096$ từ và tại mức máy thông thường chúng ta sử dụng 3 thanh ghi là PC (Program Counter), SP và AC (ACcumulator) cho các thao tác với bộ nhớ. SP được sử dụng cho các thao tác với bộ nhớ stack; PC được sử dụng làm con trỏ lệnh - trỏ tới lệnh sắp được thực hiện; còn AC được sử dụng để vận chuyển dữ liệu, để tính toán số học và cho các mục đích khác nữa. Các thanh ghi này người lập trình ở mức máy thông thường có thể "nhìn thấy" được, nghĩa là có thể sử dụng chúng.

Phương thức đánh địa chỉ là cách thức chỉ ra vị trí của dữ liệu, thường được gọi ngắn gọn là mode địa chỉ, dữ liệu ở đây có thể là toán hạng, có thể là các chỉ thị của chương trình. Ba mode địa chỉ thường được sử dụng là trực tiếp, gián tiếp và cục bộ.

Các chỉ thị sử dụng mode địa chỉ trực tiếp có chứa một địa chỉ bộ nhớ 12 bit, nằm ở 12 bit thấp để đánh được địa chỉ của toàn bộ 2^{12} (=4096) từ trong bộ nhớ. Những chỉ thị như vậy cần phải có để truy cập các biến toàn cục.

Việc đánh địa chỉ gián tiếp cho phép người lập trình tính địa chỉ nhớ, đặt nó vào trong thanh ghi AC, sau đó đọc hoặc ghi word đã được đánh địa chỉ. Dạng đánh địa chỉ này rất phổ biến và thường được dùng để truy cập các phần tử của mảng.

Việc đánh địa chỉ cục bộ chỉ ra một vị trí tương đối (độ dịch hay offset) so với SP và được dùng để truy cập các biến cục bộ.

Cả 3 mode đánh địa chỉ nêu trên tạo ra một hệ thống đánh địa chỉ đơn giản nhưng thích hợp.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

4.3.3 Tập vĩ chỉ thị

Tập chỉ thị cho máy Mac-1 được trình bày trên bảng 4-01. Mỗi chỉ thị bao gồm một trường mã phép toán - opcode và đôi khi có thêm một trường chứa địa chỉ bộ nhớ của toán hạng hay chứa một giá trị hằng số. Cột 1 (Binary) ghi mã nhị phân của chỉ thị (trong đó có tất cả các trường của nó). Cột 2 (Mnemonic) là ký hiệu gợi trí nhớ bằng ngôn ngữ Assembly. Cột 3 (Instruction) là tên chỉ thị, còn cột 4 (Meaning) là mô tả công việc mà chỉ thị thực hiện bằng các mẩu chương trình Pascal.

Trong các mẩu chương trình này *m*[x] là word bộ nhớ có địa chỉ *x*. Như vậy chỉ thị LODD nạp cho AC từ một word nhớ có địa chỉ được chỉ ra trong 12 bit thấp của chỉ thị. LODD là chỉ thị có mode địa chỉ trực tiếp. Trong khi đó LODL nạp cho AC từ một word nằm ở khoảng cách x bên trên SP, vì vậy LODL thuộc mode địa chỉ cục bộ. LODD, STOD, ADDD và SUBD thực hiện 4 chức năng cơ bản sử dụng mode địa chỉ trực tiếp, còn LODL, STOL, ADDL và SUBL thực hiện cùng các chức năng, chúng sử dụng mode đánh địa chỉ cục bộ.

Trong tập chỉ thị có 5 chỉ thị nhảy, trong đó có một chỉ thị nhảy không điều kiện JUMP và bốn chỉ thị nhảy có điều kiện là JPOS, JZER, JNEG và JNZE. Chỉ thị JUMP luôn luôn copy 12 bit thấp của nó vào PC còn 4 chỉ thị nhảy có điều kiện khác chỉ thực hiện việc đó nếu các điều kiện được chỉ ra thoả mãn.

LOCO nạp một hằng số 12 bit có giá trị nằm trong miền 0..4095 vào AC. Chỉ thị PSHI sẽ đẩy word có địa chỉ nằm trong AC lên stack. Thao tác ngược lại là POPI, nó lấy 1 word ra khỏi stack và chứa nó vào một word của bộ nhớ có địa chỉ nằm trong AC. PUSH và POP rất hữu ích cho việc quản lý stack theo nhiều cách khác nhau.

SWAP thực hiện hoán vị nội dung của AC và SP, nó sẽ có ích khi SP phải được tăng lên hoặc giảm đi một lượng còn chưa biết được lúc biên dịch. Nó cũng có ích cho việc khởi tạo SP lúc mới bắt đầu thi hành chương trình. INSP và DESP được sử dụng để thay đổi SP một lượng đã biết được trong lúc biên dịch. Do thiếu không gian để lập mã, offset ở đây đã bị hạn chế chỉ có 8 bit.

Cuối cùng chỉ thị CALL là để gọi thủ tục sau khi đã cất địa chỉ trở về vào stack, và RETN để trở về từ thủ tục bằng cách lấy địa chỉ trở về trong stack và đặt vào PC.

Để cho đơn giản, chiếc máy trên của chúng ta không được trang bị một chỉ thị input/output nào, máy của chúng ta có thể không cần đến chúng, vì chúng ta sẽ sử dụng phương pháp memory-mapped I/O, đã được giới thiệu ở mục 3.6.2 Việc giải mã địa chỉ. Theo phương pháp này việc truy cập bộ nhớ (memory) và các cổng vào/ra (I/O) là như nhau.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Bảng 4-01 Tập chỉ thị của Mac-1

Mã nhị phân (Binary)	Ký hiệu (Mnemonic)	Thực hiện	Giải thích
0000xxxxxxxxxxxx	LODD	Load direct	ac:=m[x]
0001xxxxxxxxxxxx	STOD	Store direct	m[x]:=ac
0010xxxxxxxxxxxx	ADDD	Add direct	ac:=ac+m[x]
0011xxxxxxxxxxxx	SUBD	Subtract direct	ac:=ac-m[x]
0100xxxxxxxxxxxx	JPOS	Jump positive	if ac ≥ 0 then pc:=x
0101xxxxxxxxxxxx	JZER	Jump zero	if ac=0 then pc:=x
0110xxxxxxxxxxxx	JUMP	Jump	pc:=x
0111xxxxxxxxxxxx	LOCO	Load constant	ac:=x (0 ≤ x ≤ 4095)
1000xxxxxxxxxxxx	LODL	Load local	ac:=m[sp+x]
1001xxxxxxxxxxxx	STOL	Store local	m[x+sp]:=ac
1010xxxxxxxxxxxx	ADDL	Add local	ac:=ac+m[sp+x]
1011xxxxxxxxxxxx	SUBL	Subtract local	ac:=ac-m[sp+x]
1100xxxxxxxxxxxx	JNEG	Jump negative	if ac < 0 then pc:=x
1101xxxxxxxxxxxx	JNZE	Jump nonzero	if ac <> 0 then pc:=x
1110xxxxxxxxxxxx	CALL	Call procedure	sp:=sp-1; m[sp]:=pc; pc:=x
1111000000000000	PSHI	Push indirect	sp:=sp-1; m[sp]:=m[ac]
1111001000000000	POPI	Pop indirect	m[ac]:=m[sp]; sp:=sp+1
1111010000000000	PUSH	Push onto stack	sp:=sp-1; m[sp]:=ac
1111011000000000	POP	Pop from stack	ac:=m[sp]; sp:=sp+1
1111100000000000	RETN	Return	pc:=m[sp]; sp:=sp+1
1111101000000000	SWAP	Swap ac, sp	tmp:=ac; ac:= sp; sp:=tmp
11111100YYYYYYYY	INSP	Increment sp	sp:=sp+y (0 ≤ y ≤ 255)
11111110YYYYYYYY	DESP	Decrement sp	sp:=sp-y (0 ≤ y ≤ 255)

xxxxxxxxxxxx là địa chỉ máy 12 bit; ở cột 4 nó được ghi là x

YYYYYYYY là một hằng số 8 bit; ở cột 4 nó được ghi là y

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

4.4 THÍ DỤ VỀ MỘT VI CHƯƠNG TRÌNH

Sau khi nghiên cứu khá chi tiết vi kiến trúc và vĩ kiến trúc, vấn đề còn lại là việc thực hiện: chương trình chạy trên vi kiến trúc dịch chương trình của vĩ kiến trúc trông ra sao và nó làm việc như thế nào? Trước khi có thể trả lời những câu hỏi này chúng ta phải cân nhắc cẩn thận việc sẽ sử dụng ngôn ngữ lập trình nào để lập vi chương trình.

4.4.1 Ngôn ngữ Vi assembly

Về mặt nguyên tắc chúng ta có thể viết chương trình bằng số nhị phân, mỗi vi chỉ thị là 32 bit. Tuy nhiên chắc chắn là chẳng có ai muốn làm theo cách đó cả, vì lập trình theo cách đó rất nặng nề, dễ nhầm lẫn... Chính vì vậy chúng ta cần một ngôn ngữ bằng các ký hiệu để thể hiện các vi chương trình. Người lập trình có thể sử dụng một cách ghi ký hiệu là trên mỗi dòng ghi một vi chỉ thị, trên dòng đó ghi tên các trường khác 0 và giá trị của chúng. Thí dụ để cộng AC với A và chứa kết quả vào AC, chúng ta có thể viết:

ENC = 1, C = 1, B = 1, A = 10

Nhiều ngôn ngữ vi chương trình trông giống như vậy, tuy nhiên việc ghi chương trình như thế trông có vẻ kỳ quái. Một ý tưởng tốt hơn nhiều là sử dụng cách ghi của ngôn ngữ lập trình bậc cao, trong khi vẫn giữ nguyên ý tưởng cơ bản là trên mỗi dòng chương trình nguồn ghi một vi chỉ thị.

Việc viết vi chương trình bằng một ngôn ngữ bậc cao thông dụng về nguyên tắc là có thể chấp nhận được, nhưng rất kém hiệu quả vì ngôn ngữ bậc cao khác rất xa ngôn ngữ máy thực thụ. Hiệu quả của vi chương trình kém bao nhiêu sẽ làm máy chạy chậm đi đúng bấy nhiêu.

Bởi vì đối với các vi chương trình tính hiệu quả là cực kỳ quan trọng nên chúng ta sẽ sử dụng ngôn ngữ assembly. Đó là một ngôn ngữ ký hiệu, trong đó có một ánh xạ một-một từ tập các chỉ thị của ngôn ngữ assembly vào tập chỉ thị của máy. Chúng ta gọi ngôn ngữ assembly bậc cao để viết vi chương trình là ngôn ngữ vi assembly, viết tắt là - MAL (Micro Assembly Language). Chúng ta sẽ cố gắng làm cho nó trông giống với ngôn ngữ lập trình Pascal quen thuộc. Trong ngôn ngữ MAL, việc chứa vào 16 thanh ghi của bộ nhớ tạm hoặc vào MAR hoặc vào MBR được ký hiệu bằng chỉ thị gán. Như vậy thí dụ trên (ENC = 1, C = 1, B = 1, A = 10) được viết bằng MAL trở thành ac:=a+ac.

Để chỉ rõ việc sử dụng các chức năng 0, 1, 2 và 3 của ALU, chúng ta có thể viết, thí dụ:

ac:=a+ac, a:=band(ir,smask), ac:=a và a:=inv(a)

Tương ứng với 4 chức năng trên, trong đó band là "Boolean and" và inv là "invert". Các thao tác dịch trái và dịch phải có thể được ký hiệu là lshift và rshift, như:

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

tir:=lshift(tir+tir)

Nó sẽ đặt tir lên cả 2 bus A và B, thực hiện phép cộng và dịch tổng sang trái 1 bit trước khi chứa nó trở lại vào tir.

Bảng 4-02 Một số chỉ thị MAL và các vi chỉ thị tương ứng

Statement (Chỉ thị ngôn ngữ MAL)	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR
Mar := pc; rd	0	0	2	0	0	1	1	0	0	0	0	0	00
Rd	0	0	2	0	0	0	1	0	0	0	0	0	00
ir := mbr	1	0	2	0	0	0	0	0	1	3	0	0	00
pc := pc+1	0	0	0	0	0	0	0	0	1	0	6	0	00
Mar := ir; mbr := ac; wr	0	0	2	0	1	1	0	1	0	0	3	1	00
alu := tir; if n then goto 15	0	1	2	0	0	0	0	0	0	0	0	4	15
ac := inv (mbr)	1	0	3	0	0	0	0	0	1	1	0	0	00
tir := lshift (tir); if n then goto 25	0	1	2	2	0	0	0	0	1	4	0	4	25
alu := ac; if z then goto 22	0	2	2	0	0	0	0	0	0	0	0	1	22
ac :=band(ir, amask); goto 0	0	3	1	0	0	0	0	0	1	1	8	3	00
sp := sp + (-1); rd	0	0	0	0	0	0	1	0	1	2	2	7	00
tir := lshift(ir+ir); if n then goto 69	0	1	0	2	0	0	0	0	1	4	3	3	69

Việc nhảy vô điều kiện được thực hiện bằng chỉ thị goto; việc nhảy có điều kiện có thể kiểm tra n hoặc z; thí dụ:

if n then goto 27

Việc gán và nhảy có thể được kết hợp lại trên cùng một dòng, bởi vì chúng sử dụng các trường khác nhau của cùng một vi chỉ thị (mỗi vi chỉ thị có 13 trường khác nhau).

Tuy nhiên có một vấn đề nhỏ phải giải quyết là nếu chúng ta chỉ muốn kiểm tra thanh ghi chứ không muốn chứa kết quả, tức là không muốn chứa giá trị của thanh ghi đó vào bất cứ đâu. Làm thế nào để chúng ta có thể chỉ ra thanh ghi muốn kiểm tra? Để giải quyết vấn đề này chúng ta đưa ra một giả biến alu, nó có thể được gán

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

một giá trị là nội dung một thanh ghi nào đó, giá trị này sẽ được cho chạy qua ALU. Thí dụ:

alu := tir; if n then goto 27

Có nghĩa là tir sẽ chạy qua ALU (trường ALU = 2), vì vậy bit bậc cao của nó có thể kiểm tra được. Chú ý rằng việc sử dụng alu cũng có nghĩa là ENC = 0.

Để chỉ ra việc đọc hoặc ghi bộ nhớ, chúng ta chỉ việc đặt rd và wr trong chương trình nguồn. Thứ tự các phần khác nhau của một vi chỉ thị được viết bằng MAL về nguyên tắc là tuỳ ý, nhưng để cho dễ đọc chúng ta cố gắng sắp xếp chúng theo trật tự mà chúng sẽ được thi hành. Bảng 4-02 dẫn ra một số thí dụ về các chỉ thị MAL cùng với các vi chỉ thị (microinstruction) tương ứng với chúng.

4.4.2 Thí dụ về một vi chương trình

Cuối cùng chúng ta cũng đã chuẩn bị đủ các kiến thức cần thiết để có thể ghép nối chúng lại với nhau. Hình 4-10 là vi chương trình chạy trên Mic-1 và thông dịch các chương trình được viết bằng ngôn ngữ của máy Mac-1. Đó là một chương trình ngắn đến mức đáng ngạc nhiên, chỉ có 79 dòng. Cho tới lúc này chúng ta có thể thấy được sự lựa chọn tên cho các thanh ghi của bộ nhớ tạm trên hình 4-08 là có ý nghĩa rõ ràng: PC, AC và SP được sử dụng để giữ 3 thanh ghi của MAC-1. IR là thanh ghi lệnh (Instruction Register) và giữ vĩ chỉ thị hiện đang được thi hành. TIR (Temporary Instruction Register) là một thanh ghi chứa tạm thời bản sao của IR dùng để giải mã vĩ chỉ thị (opcode). Ba thanh ghi tiếp theo giữ các hằng số được chỉ ra trên thanh ghi (0, +1 và -1). Thanh ghi AMASK là mặt nạ địa chỉ 007777 (hệ cơ số 8, số nhị phân tương ứng là 0000.1111.1111.1111), được sử dụng để tách các bit của trường địa chỉ khỏi các bit của trường opcode trong chỉ thị MAC-1. SMASK (Stack MASK) là mặt nạ stack, có giá trị 000377 (hệ cơ số 8, số nhị phân tương ứng là 0000.0000.1111.1111) được sử dụng trong các chỉ thị INSP và DESP để tách địa chỉ offset 8 bit. Sáu thanh ghi còn lại (A, B, C, D, E, F) không được gán trước các chức năng và có thể được sử dụng theo ý của người lập vi chương trình.

Giống như mọi chương trình thông dịch, vi chương trình trên hình 4-10 có một vòng lặp chính, thực hiện việc lấy chỉ thị về (fetch), giải mã (decode) và thi hành các chỉ thị của chương trình đang được thông dịch, trong trường hợp này đó là các chỉ thị ở mức 2, của máy MAC-1. Vòng lặp chính của chương trình bắt đầu từ dòng 0, tại đó nó bắt đầu việc lấy về vĩ chỉ thị tại địa chỉ chứa trong PC. Trong khi chờ chỉ thị về đến nơi, vi chương trình tăng PC và tiếp tục giữ tín hiệu RD trên bus ở mức tích cực.

Khi chỉ thị về đến nơi, tại dòng 2, nó được chứa vào IR và bit bậc cao (bit 15) được kiểm tra ngay lập tức. Nếu bit 15 là 1 thì việc giải mã được thực hiện ở dòng 28, còn ngược lại nó sẽ được tiếp tục ở dòng 3.

Với giả thiết tạm thời rằng chỉ thị là LODD, bit 14 được kiểm tra trên dòng 3 và TIR được nạp chỉ thị ban đầu sau khi đã dịch trái 2 bit (dịch trái 1 bit bằng việc dùng bộ cộng và dịch trái một bit nữa sử dụng thanh ghi dịch - shifter). Chú ý rằng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

bit trạng thái ALU là N được xác định bằng đầu ra của ALU trong đó bit 14 của chỉ thị lấy về lúc này là bit bậc cao nhất, bởi vì IR + IR làm dịch IR sang trái 1 bit. Đầu ra của thanh ghi dịch không làm ảnh hưởng đến các bit trạng thái của ALU.

Mọi chỉ thị có 2 bit cao nhất là 00 cuối cùng đi tới dòng 4 để bit 13 được kiểm tra, các chỉ thị có 3 bit cao là 000 đi tới dòng 5 và những chỉ thị có 3 bit cao là 001 đi tới dòng 11.

Dòng 5 là một thí dụ về vi chỉ thị với ENC=0; nó chỉ kiểm tra TIR chứ không làm thay đổi nó. Tuỳ thuộc vào kết quả của sự kiểm tra này mà mã của LODD hoặc STOD sẽ được lựa chọn.

Đối với LODD, vi mã lệnh trước hết phải lấy về word được đánh địa chỉ trực tiếp bằng cách nạp 12 bit thấp của IR vào MAR. Trong trường hợp này (đang giải mã chỉ thị LODD) cả 4 bit cao bằng 0 nhưng đối với STOD và các chỉ thị khác thì không phải như vậy. Tuy nhiên vì MAR chỉ rộng có 12 bit nên các bit mã lệnh không ảnh hưởng tới sự lựa chọn word đã được đọc. Trên dòng 7 vi chương trình không có việc gì để làm cả, vì thế nó chỉ có chờ. Khi word đi đến nơi nó sẽ được copy vào AC và vi chương trình nhảy tới đỉnh của vòng lặp.

STOD, ADDD và SUBD là tương tự. Chỉ có một điểm đáng ghi nhớ liên quan đến việc thực hiện phép trừ (SUBD), nó sử dụng số âm dạng bù 2:

$$x-y = x + (-y) = x + (\overline{y}+1) = x + 1 + \overline{y}$$

Việc cộng 1 vào AC được thực hiện ở dòng 16, song song với việc giữ tín hiệu RD trên bus ở mức tích cực bằng vi mã lệnh rd. Trên dòng 13 chỉ có một vi mã lệnh rd, là 1 trong 13 trường điều khiển của 1 vi chỉ thị, thật là phí, nhưng vi chương trình cũng không có công việc gì khác để làm.

Vi mã lệnh cho JPOS bắt đầu ở dòng 21. Nếu AC<0, thì việc rẽ nhánh không được thực hiện và JPOS bị kết thúc ngay lập tức bằng việc nhảy trở lại đầu vòng lặp chính. Tuy nhiên nếu AC >= 0, 12 bit thấp của IR được trích ra bằng cách AND nó với mặt nạ 007777 và chứa kết quả vào PC. Ở đây chúng ta không cần phải loại bỏ các bit opcode (4 bit cao trong thanh ghi IR), vì AMASK là thanh ghi 16 bit chứa nội dung cố định, với 4 bit cao bằng 0.

JZER (dòng 23) làm công việc ngược lại với JPOS. Với JPOS nếu điều kiện được thoả mãn, việc nhảy không xảy ra và điều khiển trả về cho vòng lặp chính. Với JZER nếu điều kiện được thoả mãn, việc nhảy sẽ xảy ra. Bởi vì mã cho sự thực hiện nhảy là giống nhau đối với mọi chỉ thị nhảy, chúng ta có thể tiết kiệm vi mã lệnh bằng cách đi đến dòng 22 bất cứ khi nào có thể được. Kiểu lập trình như vậy trong các chương trình ứng dụng thường có thể bị xem là vụng về nhưng trong một vi chương trình người ta không xem như vậy, hiệu suất là trên hết.

JUMP và LOCO không phức tạp, chúng ta bỏ qua, vì thế chương trình con thi hành đáng quan tâm tiếp theo là LODL. Đầu tiên địa chỉ bộ nhớ tuyệt đối sẽ được tham khảo được tính toán bằng cách cộng offset được chứa bên trong chỉ thị vào SP. Sau đó việc đọc bộ nhớ bắt đầu được thực hiện. Bởi vì phần còn lại của mã của

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

LODL và LODD là giống nhau, chúng ta có thể sử dụng các dòng 7 và 8 cho cả hai chỉ thị này mà không gặp trở ngại gì, việc này giúp tiết kiệm bộ nhớ điều khiển mà không làm giảm tốc độ thi hành. Mã tương tự như vậy được dùng cho STOL, ADDL và SUBL. Mã cho JNEG và JNZE giống với mã cho JZER và JPOS (không có cách nào khác).

CALL đầu tiên làm giảm SP, sau đó đẩy địa chỉ trở về lên stack, và cuối cùng nhảy tới thủ tục. Dòng 49 gần giống dòng 22; nếu như chúng hoàn toàn giống nhau thì chúng ta đã có thể loại bỏ dòng 49 bằng cách đặt một lệnh nhảy vô điều kiện tới dòng 22 ở chỗ dòng 48. Nhưng tại dòng 49 chúng ta phải tiếp tục giữ tín hiệu WR tích cực cho một vi chỉ thị khác nữa, điều này làm cho dòng lệnh 49 khác dòng lệnh 22.

Các vĩ chỉ thị còn lại đều có 4 bit cao bằng 1111, việc giải mã tương tự như trên, xin dành công việc còn lại này cho bạn đọc nghiên cứu như một bài tập.

```
0:  mar := pc; rd;                          {main loop}
1:  pc := pc+1; rd;                         {increment pc}
2:  ir  := mbr; if n then goto 28;          {save, decode mbr}
3:  tir :=lshift(ir+ir); if n then goto 19;
4:  tir := lshift(tir); if n then goto 11;  {000x or 001x?}
5:  alu := tir; if n then goto 9;           {0000 or 0001?}
6:  mar := ir; rd;                          {0000 = LODD}
7:  rd;
8:  ac := mbr; goto 0
9:  mar := ir; mbr := ac; wr;               {0001 = STOD}
10:  wr; goto 0;
11:  alu := tir; if n then goto 15;         {0010 or 0011?}
12:  mar := ir; rd;                         {0010 = ADDD}
13:  rd;
14:  ac := mbr + ac; goto 0
15:  mar := ir; rd;                         {0011 = SUBD}
16:  ac := ac + 1; rd;                      {Note: x - y = x + 1 + not  y}
17:  a := inv(mbr);
18:  ac := ac + a; goto 0;
19:  tir := lshift(tir); if n then goto 25; {010x 011x?}
20:  alu := tir; if n then goto 23;         {0100 or 0101?}
21:  alu := ac; if n then goto 0            {0100 = JPOS}
22:  pc := band(ir, amask); goto 0;         {perform the jump}
23:  alu := ac; if z then goto 22;          {0101 = JZER}
```


127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

```
24: goto 0;                                  {jump failed}
25: alu := tir; if n then goto 27;           {0110 or 0111?}
26: pc := band(ir, amask); goto 0;           {0110 = JUMP}
27: ac := band(ir, amask); goto 0;           {0111 = LOCO}
28: tir := lshift(ir+ir); if n then goto 40; {10xx or 11xx?}
29: tir := lshift(tir); if n then goto 35;   {100x or 101x?}
30: alu := tir; if n then goto 33;           {1000 or 1001?}
31: a := ir + sp;                            {1000 = LODL}
32: mar := a; rd; goto 7;
33: a := ir + sp;                            {1001 = STOL}
34: mar := a; mbr := ac; wr; goto 10;
35: alu := tir; if n then goto 38;           {1010 or 1011?}
36: a := ir + sp;                            {1010 = ADDL}
37: mar := a; rd; goto 13;
38: a := ir + sp;                            {1011 = SUBL}
39: mar := a; rd; goto 16;
40: tir := lshift(tir); if n then goto 46;   {110x or 111x?}
41: alu := tir; if n then goto 44;           {1100 or 1101?}
42: alu := ac; if n then goto 22;            {1100 = JNEG}
43: goto 0;
44: alu := ac; if z then goto 0;             {1101 = JNZE}
45: pc := band(ir, amask); goto 0;
46: tir := lshift(tir); if n then goto 50;
47: sp := sp + (-1);                         {1110 = CALL}
48: mar := sp; mbr := pc; wr;
49: pc := band(ir, amask); wr; goto 0;
50: tir := lshift(tir); if n then goto 65;   {1111, examine addr}
51: tir := lshift(tir); if n then goto 59;
52: alu := tir; if n then goto 56;
53: mar := ac; rd;                           {1111000 = PSHI}
54: sp := sp + (-1); rd;
55: mar := sp; wr; goto 10;
56: mar := sp; sp := sp + 1; rd;             {1111001 = POPI}
57: rd;
58: mar := ac; wr; goto 10;
59: alu := tir; if n then goto 62;
60: sp := sp + (-1);                         {1111010 = PUSH}
```


127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

```
61: mar := sp; mbr := ac; wr; goto 10;
62: mar := sp; sp := sp + 1; rd;                {1111011 = POP}
63: rd;
64: ac := mbr; goto 0;
65: tir := lshift(tir); if n then goto 73;
66: alu := tir; if n then goto 70;
67: mar := sp; sp := sp + 1; rd;                {1111100 = RETN}
68: rd;
69: pc := mbr; goto 0;
70: a := ac;                                    {1111101 = SWAP}
71: ac := sp;
72: sp := a; goto 0;
73: alu := tir; if n then goto 76;
74: a := band(ir, smask);                       {1111110 = INSP}
75: sp := sp + a; goto 0;
76: a := band(ir, smask);                       {1111111 = DESP}
77: a := inv(a);
78: a := a + 1; goto 75;
```

Hình 4-10 Vi chương trình

4.4.3 Nhận xét về vi chương trình

- Bản thân phần cứng cho phép thực hiện cơ chế song song kiểu đường ống

Mặc dầu chúng ta đã thảo luận khá chi tiết về vi chương trình, nhưng vẫn còn có một số điểm cần phải lưu ý. Trên hình 4-10 chúng ta tăng PC lên ở dòng 1. Việc này lẽ ra có thể thực hiện được ở dòng 0 cũng tốt, để giải phóng dòng 1 cho một việc khác trong khi đang chờ. Trong chiếc máy này không có việc gì khác để làm cả nhưng trong các máy thực tế vi chương trình có thể phải sử dụng cơ hội này để kiểm tra các thiết bị vào/ra đang chờ đợi dịch vụ, làm tươi RAM động hoặc một việc khác nào đó.

Nếu chúng ta cứ để dòng 1 như đã thấy, chúng ta vẫn có thể nâng cao tốc độ của máy bằng cách sửa dòng 8 để đọc:

```
mar:=pc; ac:=mbr; rd; goto 1;
```

Nói cách khác là chúng có thể bắt đầu lấy về chỉ thị tiếp theo trước khi chúng ta thực sự hoàn thành chỉ thị hiện đang thực hiện. Khả năng này tạo ra một dạng cơ sở của việc cho chỉ thị đi theo đường ống. Gợi ý đó cũng có thể đem áp dụng cho các chương trình con thi hành khác.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Rõ ràng là phần lớn thời gian thi hành mỗi vĩ chỉ thị là dùng cho việc giải mã nó từng bit một. Quan sát này gợi ý rằng nếu có thể nạp cho MPC dưới sự điều khiển của vi chương trình thì thật có ích. Trong nhiều máy tính hiện nay vi kiến trúc có phần cứng hỗ trợ cho việc trích lấy các mã vĩ chỉ thị và đặt nó trực tiếp vào MPC để thực hiện sự rẽ nhánh theo nhiều đường. Thí dụ nếu như chúng ta có thể dịch IR đi 9 bit sang phải, xoá 9 bit cao và đặt con số kết quả vào MPC, thì chúng ta có 128 cách rẽ nhánh tới các vị trí từ 0 đến 127. Mỗi một trong các word này chứa vi chỉ thị đầu tiên của vĩ chỉ thị tương ứng với nó. Mặc dầu cách tiếp cận này làm lãng phí bộ nhớ điều khiển, nhưng nó tăng tốc độ của máy lên rất nhiều, chính vì vậy trong thực tế người ta luôn luôn áp dụng một cách thức gần giống như thế.

- Thiết kế máy tính phân mức nhằm đạt được tính hiệu quả và tính đơn giản.

Chúng ta nghiên cứu mức vi chương trình xuất phát từ ý tưởng cơ bản là bắt đầu bằng một máy tính có phần cứng rất đơn giản. Cái máy mà chúng ta lấy làm thí dụ gồm có hơn 22 thanh ghi một chút, một bộ nhớ ROM nhỏ làm bộ nhớ điều khiển, một bộ cộng, một bộ dịch và một số mạch tổ hợp dùng cho việc phân/hợp kênh, giải mã và định trình. Bằng việc sử dụng các phần cứng này chúng ta có khả năng xây dựng một trình thông dịch bằng phần mềm để thi hành các chỉ thị của máy mức 2. Với sự trợ giúp của một compiler, chúng ta có thể dịch các chương trình ngôn ngữ bậc cao sang các chỉ thị bậc 2 và sau đó lần lượt thông dịch các chỉ thị này.

Như vậy để chạy một chương trình được viết bằng ngôn ngữ lập trình bậc cao, trước hết chúng ta phải dịch nó sang ngôn ngữ mức 2 và sau đó thông dịch các chỉ thị bậc 2 nhận được. Mức 2 được sử dụng một cách có hiệu quả, làm giao diện giữa trình dịch (compiler) và trình thông dịch (interpreter). Mặc dầu về nguyên tắc trình dịch có thể trực tiếp sinh ra vi mã lệnh, nhưng việc này rất phức tạp và gây lãng phí bộ nhớ. Mỗi vĩ chỉ thị của chúng ta chiếm một word 16 bit, cần trung bình khoảng 4 vi chỉ thị 32 bit để thực hiện. Nếu chúng ta biên dịch trực tiếp sang mức 1, bộ nhớ tổng cộng cần đến có thể tăng khoảng tám lần. Hơn nữa bộ nhớ tăng thêm lên đó lại là bộ nhớ điều khiển có thể ghi vào được, nó rất đắt tiền vì là bộ nhớ tốc độ cao. Người ta tránh không dùng bộ nhớ chính để chứa vi mã lệnh vì nó làm cho máy chạy chậm đi nhiều.

Dưới ánh sáng của những thí dụ cụ thể này, lý do tại sao ngày nay các máy tính thường được thiết kế như một dãy các mức trở nên rõ ràng, đó là nhằm đạt được hiệu quả cao và tính đơn giản, bởi vì mỗi mức giải quyết một mức độ trừu tượng hoá khác nhau. Người thiết kế mức 0 lo suy nghĩ về việc làm thế nào để chắt bóp ra được vài na-nô giây cuối cùng của ALU, bằng việc sử dụng một thuật toán mới "đẹp mắt" nào đó để làm giảm thời gian lan truyền carry. Những người lập vi chương trình lo nghĩ về việc làm thế nào để sử dụng có lợi nhất từng vi chỉ thị, điển hình là việc cố gắng khai thác đến mức tối đa cơ chế song song vốn có trong phần cứng. Người thiết kế tập vĩ chỉ thị lo nghĩ về việc làm thế nào để cung cấp được một giao diện mà cả người viết compiler lẫn người lập vi chương trình có thể học và yêu thích, đồng thời có hiệu quả cao. Rõ ràng là mỗi mức có những mục tiêu, vấn đề phải giải quyết và kỹ thuật khác nhau và nói chung là cách nhìn nhận cái máy khác

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

nhau. Bằng việc chia việc thiết kế toàn bộ cái máy thành các công việc nhỏ hơn, chúng ta có thể nắm bắt được sự phức tạp vốn có trong việc thiết kế một máy tính hiện đại.

4.5 NHỮNG VẤN ĐỀ CẦN CÂN NHẮC, LỰA CHỌN KHI THIẾT KẾ MỨC VI CHƯƠNG TRÌNH

Cũng như mọi vấn đề trong lĩnh vực khoa học máy tính, việc thiết kế vi kiến trúc cũng chứa đầy những sự lựa chọn thoả hiệp. Phần tiếp theo chúng ta xem xét một số vấn đề về thiết kế và những sự lựa chọn thoả hiệp liên quan đến chúng.

4.5.1 Vi chỉ thị theo chiều đứng và theo chiều ngang

- **Vi chỉ thị được thiết kế theo chiều ngang**

Có lẽ việc cân nhắc chủ yếu là đưa mức độ lập mã (encoding) nhiều đến đâu vào các vi chỉ thị. Nếu như có một người nào đó định xây dựng Mic-1 bằng 1 chip VLSI, anh ta có thể lờ đi những khái niệm trừu tượng như thanh ghi, ALU ... và chỉ xem xét đến các cổng. Để làm cho máy chạy, cần phải có một số tín hiệu nhất định, chẳng hạn 16 tín hiệu OE để mở cổng các thanh ghi cho dữ liệu đi lên bus A và các tín hiệu để điều khiển các chức năng ALU. Khi chúng ta nhìn vào bên trong ALU, chúng ta thấy rằng các mạch điện bên trong thực tế bị điều khiển bằng 4 đường dây, chứ không phải là 2, bởi vì góc dưới bên trái của ALU (hình 3-13) là một mạch giải mã "2-to-4".

Nói tóm lại, đối với mỗi máy, một tập n tín hiệu điều khiển nào đó được sử dụng ở những chỗ phù hợp có thể làm cho máy chạy mà không cần giải mã. Quan điểm này dẫn tới một khuôn dạng vi chỉ thị khác: đó là vi chỉ thị được tạo ra với độ rộng n bit, mỗi bit dùng cho một tín hiệu điều khiển. Các vi chỉ thị được thiết kế theo nguyên tắc này được gọi là theo chiều ngang (horizontal).

- **Vi chỉ thị được thiết kế theo chiều đứng**

Thái cực ngược lại là các vi chỉ thị chứa một số nhỏ các trường được lập mã ở mức độ cao. Các vi chỉ thị này được gọi là theo chiều đứng (vertical). Những tên gọi này xuất phát từ việc các hoạ sĩ có thể vẽ phác hoạ bộ nhớ điều khiển như thế nào: các thiết kế theo chiều nằm ngang có một số lượng khá ít các vi chỉ thị rộng; còn các thiết kế theo chiều đứng có nhiều chỉ thị hẹp.

- **Vi chỉ thị được thiết kế theo kiểu hỗn hợp**

Giữa hai thái cực này có nhiều kiểu thiết kế hỗn hợp. Các vi chỉ thị của chúng ta có các bit thí dụ như MAR, MBR, RD, WR và AMUX, chúng trực tiếp điều khiển các chức năng của phần cứng. Mặt khác các trường A, B, C và ALU lại cần một số logic giải mã trước khi chúng được đưa đến các cổng riêng biệt.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

(Thực chất các vi chỉ thị mà chúng ta đã nghiên cứu được thiết kế theo kiểu hỗn hợp, nhưng gần với kiểu thiết kế nằm ngang hơn là kiểu thiết kế đứng. Để cho đơn giản dưới đây chúng ta coi các vi chỉ thị đã nghiên cứu của mình là loại nằm ngang).

Một vi chỉ thị hoàn toàn theo chiều đứng (đây là một thái cực) có thể chỉ có một mã lệnh, nó chỉ là một sự tổng quát hoá của trường ALU, ngoài ra là một số toán hạng, chẳng hạn như các trường A, B và C. Theo cách tổ chức như vậy, cần phải có các mã lệnh để đọc, ghi bộ nhớ chính, tạo ra các vi lệnh nhảy (microjump).v.v. bởi vì các trường điều khiển các chức năng này không còn có trong máy.

4.5.2 Lập siêu vi chương trình - Nanoprogramming

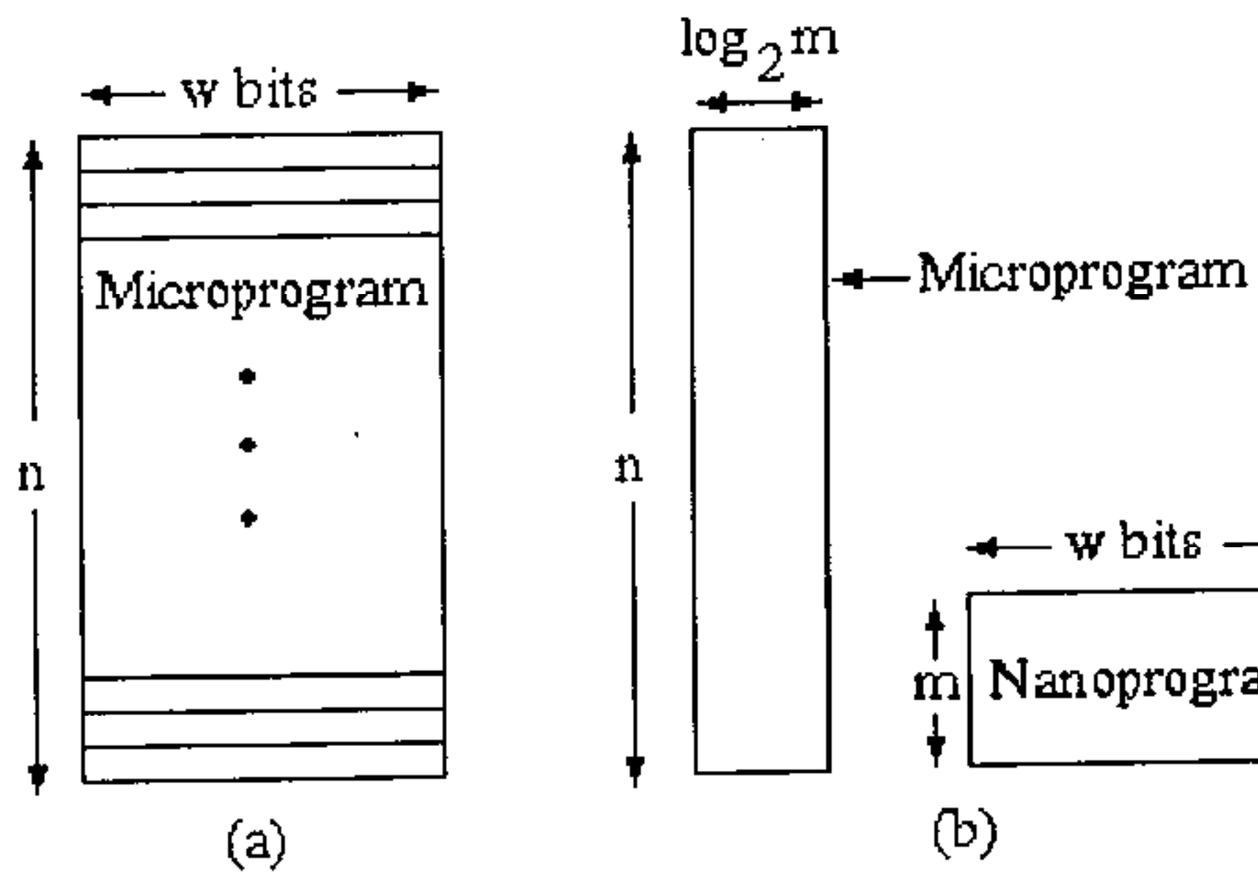
Theo các thiết kế mà chúng ta đã bàn luận ở trên, cần phải có hai bộ nhớ: bộ nhớ chính dùng để chứa các chương trình mức 2 và bộ nhớ điều khiển dùng để chứa vi chương trình.

Có một cách thiết kế đạt được các ưu điểm của cả cách thiết kế vi chỉ thị theo chiều ngang lẫn cách thiết kế vi chỉ thị theo chiều đứng, đây là một phương pháp thoả hiệp, có thể áp dụng hiệu quả khi có nhiều vi chỉ thị xảy ra nhiều lần. Phương pháp này có tên gọi là Lập siêu vi chương trình - Nanoprogramming, sử dụng một bộ nhớ thứ ba, được gọi là nanostore.

Vi chương trình được trình bày trên hình 4-10 không có thuộc tính này. Chỉ thị thường xảy ra nhất là chỉ thị có chứa rd, nó xuất hiện 5 lần.

- **Khái niệm nanoprogramming**

Hình 4-11 minh hoạ khái niệm nanoprogramming. Ở hình 4-11a là một vi chương trình gồm n vi chỉ thị, mỗi vi chỉ thị rộng w bit. Bộ nhớ điều khiển - control store cần có dung lượng nw bit để chứa vi chương trình.



Hình 4-11 a) Một vi chương trình thông thường; b) Nanoprogram tương ứng nếu chỉ có m vi chỉ thị khác nhau trong vi chương trình

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Giả sử sau khi nghiên cứu kỹ lưỡng vi chương trình, chúng ta phát hiện ra rằng chỉ có m vi chỉ thị khác nhau được sử dụng trong tổng số 2^w vi chỉ thị có thể có, với m rất nhỏ hơn n. Nếu như vậy, chúng ta có thể sử dụng một bộ nhớ đặc biệt là nanostore gồm m word, mỗi word rộng w bit để chứa các chỉ thị này. Khi đó mỗi vi chỉ thị trong chương trình ban đầu có thể được thay thế bằng địa chỉ của một word trong bộ nhớ nanostore có chứa vi chỉ thị đó. Bởi vì chỉ có m word trong nanostore nên bộ nhớ điều khiển control store chỉ cần có độ rộng $\log_2 m$ là đủ, xem hình 4-11b.

Vi chương trình được thi hành như sau. Word đầu tiên được lấy từ bộ nhớ điều khiển về. Sau đó nó được sử dụng để chọn một word trong nanostore, word này được lấy về và đặt trong thanh ghi vi chỉ thị. Sau đó các bit của thanh ghi này được dùng để điều khiển các cổng trong một chu kỳ. Vào cuối chu kỳ, word tiếp theo trong bộ nhớ điều khiển được lấy về và quá trình lại được lặp lại.

- **Một thí dụ**

Chúng ta lấy một thí dụ cụ thể, giả sử rằng vi chương trình ban đầu gồm 4096x100 bit (có chứa 4096 vi chỉ thị, mỗi vi chỉ thị rộng 100 bit) nhưng trong đó chỉ có 128 vi chỉ thị khác nhau. Một bộ nhớ nanostore kích thước 128x100 bit là đủ để chứa tất cả các vi chỉ thị sẽ cần đến. Bộ nhớ điều khiển khi đó trở thành 4096x7 bit, mỗi word (7 bit) của nó trỏ tới một siêu vi chỉ thị (nanoinstruction). Trong thí dụ này bộ nhớ điều khiển tiết kiệm được:

$$4096x100 - 4096x7 - 128x100 = 368.128 \text{ bit}$$

Hoặc tính tỉ lệ thì bộ nhớ điều khiển có kích thước giảm đi chỉ bằng 7% kích thước ban đầu. Giá phải trả cho sự tiết kiệm bộ nhớ điều khiển là sự thi hành chương trình chậm hơn.

Máy có bộ nhớ điều khiển hai mức này sẽ chạy chậm hơn so với máy đầu tiên bởi vì chu kỳ lấy lệnh bây giờ cần hai lần truy cập bộ nhớ, một lần truy cập bộ nhớ control store và một lần truy cập bộ nhớ nanostore. Hai lần lấy lệnh này không thể thực hiện gối lên nhau. Nanoprogramming sẽ có hiệu quả cao khi các vi chỉ thị được sử dụng rất nhiều lần.

4.5.3 Đường ống dẫn chỉ thị và các chỉ thị nhảy

- **Tạo đường ống dẫn chỉ thị - Pipelining**

Pipelining chính là một trong các cách để tăng tốc độ của máy, bằng cách xây dựng phần cứng bao gồm một số đơn vị chức năng và liên kết chúng thành một đường ống. (Trên hình 2-05 ở chương 2 chúng ta đã thấy một đường ống gồm 5 đơn vị chức năng làm việc như thế nào).

Trên hình 4-12 chúng ta thấy rằng nếu chỉ thị 1 được lấy về trong chu kỳ 1, thì nó sẽ được thi hành trong chu kỳ 5. Tương tự như vậy nếu chỉ thị 4 được lấy về trong chu kỳ 8, được giải mã trong chu kỳ 9 .v.v, nó sẽ được thi hành trong chu kỳ 12. Trong các điều kiện tối ưu, trong mỗi chu kỳ tiếp theo sẽ có một chỉ thị được thi

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

hành, tốc độ thi hành chỉ thị trung bình sẽ là một chỉ thị trong một chu kỳ, chứ không phải là một chỉ thị trong 5 chu kỳ.

- **Tác động của lệnh nhảy tới đường ống**

Điều đáng tiếc là các chỉ thị nhảy (jump) làm hỏng đường ống của chúng ta, chúng chiếm khoảng 30% trong tổng số các chỉ thị của các chương trình. Các chỉ thị nhảy có thể được phân làm 3 loại: không điều kiện, có điều kiện và vòng lặp.

Chúng ta hãy xem điều gì xảy ra với đường ống trên hình 4-12 khi gặp một chỉ thị nhảy có điều kiện, được đánh dấu bằng chữ B. Chỉ thị tiếp theo sẽ được thi hành có thể là chỉ thị đi theo sau chỉ thị jump, nhưng đó cũng có thể là chỉ thị nằm tại địa chỉ được nhảy đến. Bởi vì đơn vị lấy chỉ thị chỉ biết được sẽ phải lấy chỉ thị nào khi chỉ thị nhảy đã được thi hành, cho nên nó dừng lại và chờ cho đến khi chỉ thị jump được thi hành xong rồi mới có thể làm tiếp. Hệ quả là đường ống bị rỗng, chỉ khi chu kỳ 7 đã kết thúc nó mới biết được chỉ thị tiếp theo là chỉ thị nào.

Chỉ thị jump thực sự đã làm mất của ta 4 chu kỳ, người ta gọi số chu kỳ bị mất này là **jump penalty** (Sự phạt do lệnh nhảy). Nếu trong ba chỉ thị lại có một chỉ thị jump thì rõ ràng là hiệu suất của máy bị mất đi đáng kể.

Đã có rất nhiều công trình nghiên cứu về vấn đề giành lại một phần trong phần hiệu suất bị mất đi này. Điều đơn giản nhất có thể làm là cứ cho rằng lệnh nhảy sẽ không nhảy và tiếp tục đổ các chỉ thị tiếp theo vào đường ống, coi chỉ thị jump cũng như các chỉ thị số học đơn giản. Nếu sau đó chỉ thị jump thực tế cũng không nhảy thật thì chúng ta không bị mất gì cả. Còn nếu chỉ thị jump có nhảy thì chúng ta sẽ phải loại bỏ các chỉ thị hiện đang nằm trong đường ống, và bắt đầu từ đầu. Điều này được gọi một cách hình tượng là "bóp bẹp" đường ống - **squashing**.

	Chu kỳ - Cycle												
	1	2	3	4	5	6	7	8	9	10	11		
Lấy chỉ thị (fetch)	1	2	B					4	5	6	7	8	9
Giải mã (decode)		1	2	B					4	5	6	7	8
Tính địa chỉ			1	2	B					4	5	6	7
Lấy toán hạng				1	2	B					4	5	6
Thi hành					1	2	B					4	5

Hình 4-12 Một đường ống gồm 5 đoạn. Con số (1, 2,...) là nhãn của chỉ thị, chỉ thị có nhãn B là chỉ thị nhảy

Chúng ta sẽ xây dựng một mô hình đơn giản của việc bị mất hiệu suất:

Gọi P_j là xác suất mà một chỉ thị là chỉ thị jump.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Gọi P_t là xác suất mà chỉ thị jump sẽ nhảy.

Gọi b là jump penalty.

Thời gian thi hành trung bình một chỉ thị tính bằng chu kỳ (ký hiệu là average_instruction_time) có thể tính được như là tổng có trọng số của hai trường hợp, chỉ thị là chỉ thị nhảy và không phải là chỉ thị nhảy:

$$\text{average_instruction_time} = (1 - P_j)(1) + P_j[P_t(1+b) + (1 - P_t)(1)]$$

Thực hiện vài tính toán đại số, có thể tính được:

$$\text{average_instruction_time} = 1 + bP_jP_t.$$

Như vậy hiệu suất thi hành sẽ là: $1/(1 + bP_jP_t)$. Với b=4, P_j=0.3 và P_t=0.65 (đây là các giá trị điển hình mà người ta đo được), có thể thấy được máy chạy với một tốc độ dưới 60% tốc độ mà nó có thể chạy được nếu không có các lệnh nhảy.

- **Giảm ảnh hưởng của lệnh nhảy đến hiệu suất của máy**

Cần phải làm gì để nâng cao hiệu suất? Nếu chúng ta có thể dự đoán được mỗi lệnh nhảy sẽ làm gì, nhảy hay không nhảy, thì chúng ta có thể lấy về đúng chỉ thị cần thiết và tránh được việc bị phạt - jump penalty. Trong công thức tính hiệu suất của chúng ta ở trên, chúng ta có thể thay P_t bằng P_w, là xác suất đoán sai.

Có thể có hai loại dự đoán: tĩnh (static), hay còn được gọi là dự đoán trong thời gian dịch chương trình (compile time) và động (dynamic), hay còn được gọi là dự đoán trong thời gian chạy chương trình (run time). Theo cách dự đoán tĩnh, compiler thực hiện việc dự đoán đối với từng chỉ thị nhảy mà nó phát sinh ra. Thí dụ với các chỉ thị loop, việc đoán rằng nó sẽ nhảy trở về đầu vòng lặp hầu như là luôn đúng. Khi kiểm tra một điều kiện không chắc đã xảy ra, thí dụ một lời gọi hệ thống sẽ trả lại mã lỗi, việc jump sẽ không nhảy hầu như là luôn đúng. Trong nhiều trường hợp người ta sử dụng các chỉ thị khác nhau cho các trường hợp nêu trên, chỉ cần nhìn vào mã lệnh (opcode) là có thể thấy được sự gợi ý.

Những người thiết kế máy tính thường sử dụng một cách tỉ mỉ hơn, đó là cung cấp hai mã lệnh cho mỗi loại chỉ thị jump và đòi hỏi compiler sử dụng mã lệnh thứ nhất nếu nó nghĩ rằng lệnh jump sẽ nhảy và mã lệnh thứ hai trong trường hợp ngược lại.

Đối với các chương trình sẽ được sử dụng rất thường xuyên, người ta thường áp dụng một cách khác, đầu tiên cho chương trình chạy trong điều kiện mô phỏng và ghi lại hành vi thực của mỗi lệnh nhảy. Sau đó chương trình mã máy có thể được sửa chữa bằng việc thay thế từng chỉ thị nhảy bằng một mã chỉ thị phù hợp, tương ứng với khả năng sẽ xảy ra và khả năng sẽ không xảy ra sự nhảy.

Phương pháp dự đoán khác là dự đoán động - dynamic. Trong khi thi hành, vi chương trình xây dựng một bảng chứa các địa chỉ của các chỉ thị jump và theo dõi sát hành vi của từng chỉ thị này. Phương pháp này có khuynh hướng làm cho máy chạy chậm đi vì phải liên tục ghi chép, vì thế có thể sử dụng các mạch phần cứng để

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

trợ giúp công việc này. Các phép đo đạc đã chỉ ra rằng bằng phương pháp này việc đạt được độ chính xác 90% không phải là khó.

Việc dự đoán sự nhảy không phải là cách giải quyết vấn đề duy nhất. Có một cách giải quyết khác là cố gắng xác định sớm việc lệnh nhảy có nhảy hay không ngay bên trong đường ống. Một số phép kiểm tra, chẳng hạn nhảy nếu bằng dễ thực hiện hơn nhiều so với nhảy nếu nhỏ hơn. Phép kiểm tra bằng có thể được thực hiện bằng bộ so sánh (comparator), trong khi đó phép kiểm tra nhỏ hơn lại cần đến cả một chu kỳ đường dữ liệu để thực hiện phép trừ. Theo cách tiếp cận này, cứ mỗi khi gặp một lệnh nhảy, vi chương trình sẽ làm một phép kiểm tra nhanh trong công đoạn đầu của đường ống để "ngay lập tức" biết được lệnh nhảy có nhảy hay không. Nếu có nhảy thì nó biết được nơi phải lấy lệnh tiếp theo về.

4.5.4 Bộ nhớ cache

Trong lịch sử phát triển của máy tính, CPU luôn có tốc độ cao hơn bộ nhớ, khi bộ nhớ được cải tiến để nâng cao tốc độ thì CPU cũng được cải tiến, đạt được tốc độ lớn hơn. Vấn đề này không phải là do có khó khăn về công nghệ chế tạo bộ nhớ mà vì lý do kinh tế, các bộ nhớ tốc độ cao có giá thành cao, làm giảm khả năng cạnh tranh của sản phẩm.

4.5.4.1 Tính cục bộ của các chương trình và ý tưởng về cache

Người ta đã nắm được kỹ thuật kết hợp một bộ nhớ nhỏ tốc độ cao với một bộ nhớ lớn tốc độ thấp để có được một bộ nhớ có tốc độ của bộ nhớ nhỏ tốc độ cao và dung lượng của bộ nhớ lớn tốc độ thấp với giá vừa phải. Bộ nhớ nhỏ tốc độ cao được gọi là **cache**, cache hoạt động theo sự điều khiển của vi chương trình.

Việc tạo ra một bộ nhớ như vậy dựa trên nguyên lý cục bộ (Principle of locality). Khi phân tích các chương trình, người ta đã phát hiện ra rằng: Tại mỗi thời điểm các chương trình chỉ truy cập một phần tương đối nhỏ không gian địa chỉ của nó . Tính chất cục bộ có hai kiểu khác nhau:

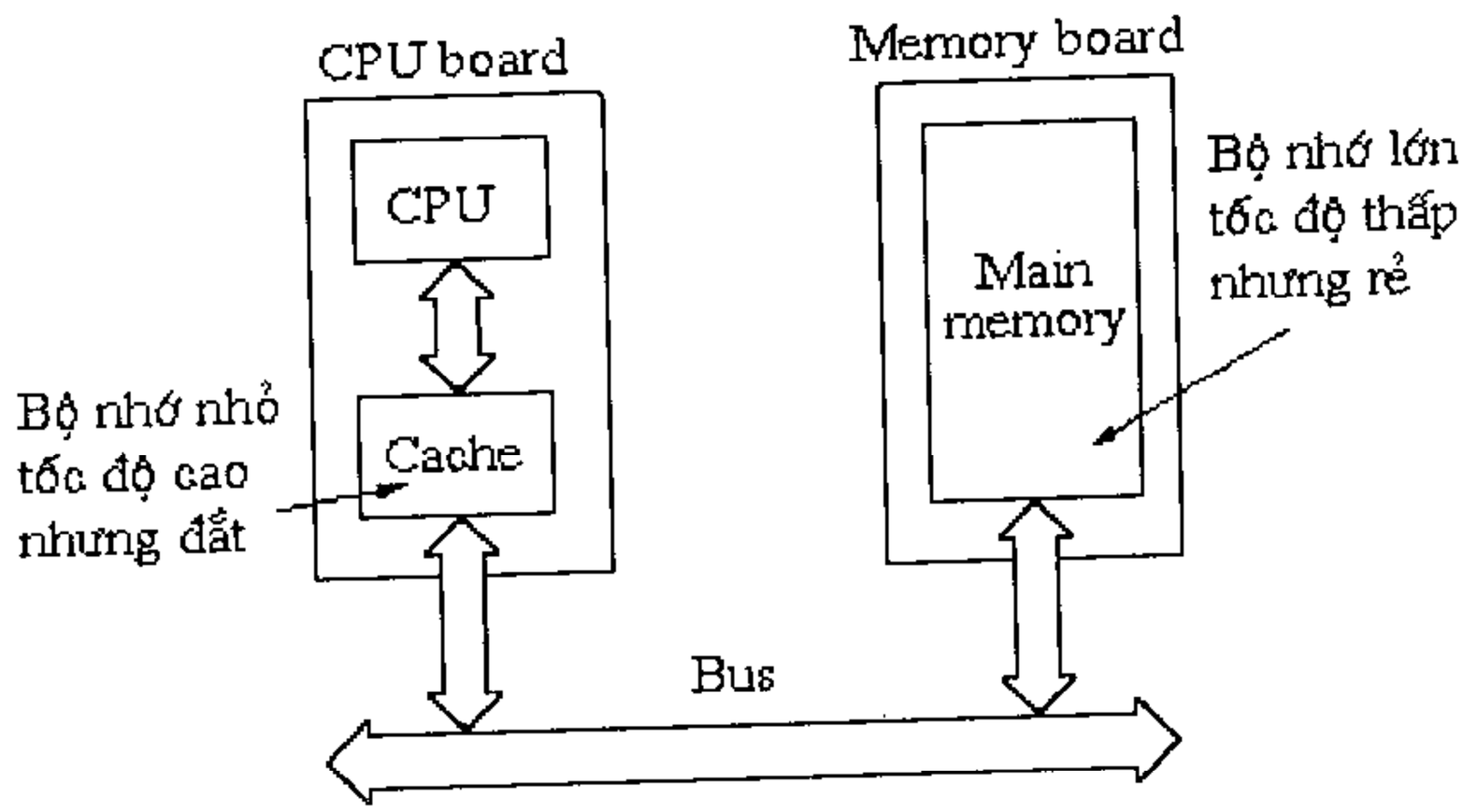
- Cục bộ về thời gian (Locality in time): Nếu một phần tử nào đó của chương trình được tham khảo thì không lâu sau nó dễ được tham khảo lại.
- Cục bộ về không gian (Spatial locality): Nếu một phần tử nào đó của chương trình được tham khảo thì sau đó các phần tử nằm gần phần tử đó dễ được tham khảo.

Thí dụ về tính cục bộ của chương trình:

1. Hầu hết các chương trình đều chứa vòng lặp, vì vậy các chỉ thị và dữ liệu rất có thể được truy cập lặp đi lặp lại, thể hiện tính cục bộ về thời gian ở mức độ cao.
2. Các chỉ thị trong chương trình thường được truy cập tuần tự - thể hiện tính cục bộ về không gian ở mức độ cao. Việc truy cập dữ liệu cũng thể hiện tính cục bộ về không gian một cách rất tự nhiên, chẳng hạn khi chúng ta truy cập một phần tử của mảng hay một record thì sau đó thường sẽ truy cập phần tử mảng hay record lân cận.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Ý tưởng chung là: khi cần truy cập một từ, từ đó sẽ được chuyển từ bộ nhớ lớn tốc độ thấp (bộ nhớ chính - main memory) về bộ nhớ nhỏ tốc độ cao (bộ nhớ cache), để lần sau nếu được sử dụng thì có thể truy cập nhanh hơn.



Hình 4-13 Sử dụng bộ nhớ cache

Nếu một từ được đọc hoặc viết k lần trong một khoảng thời gian ngắn thì bộ xử lý chỉ cần truy cập bộ nhớ lớn tốc độ thấp 1 lần và truy cập bộ nhớ nhỏ tốc độ cao k-1 lần. Số k càng lớn thì hiệu quả của cache càng cao.

Chúng ta có thể hình thức hoá sự tính toán này bằng cách đưa ra c - là thời gian truy cập bộ nhớ cache, m - là thời gian truy cập bộ nhớ chính, h - là tỉ số tìm thấy (hit ratio), là tỉ số của tổng số lần truy cập được thoả mãn từ bộ nhớ cahce trên số lần truy cập bộ nhớ. Trong ví dụ chúng ta vừa dẫn ra ở phần trên, h = (k-1)/k. Cũng có những tác giả định nghĩa tỉ số không tìm thấy - miss ratio, đó là 1-h.

Với các định nghĩa này chúng ta có thể tính được thời gian truy cập trung bình (mean access time) như sau:

$$\text{mean access time} = c + (1-h)m$$

Khi $h \rightarrow 1$, mọi sự truy cập bộ nhớ có thể được thoả mãn nhờ cache, thời gian truy cập trung bình tiến tới c. Mặt khác khi h $h \rightarrow 0$, luôn cần phải truy cập bộ nhớ chính, vì vậy thời gian truy cập trung bình tiến tới c+m, đầu tiên là thời gian c để kiểm tra bộ nhớ cache (không thành công), và sau đó là thời gian m để truy cập bộ nhớ chính. Trong một số hệ thống, việc truy cập bộ nhớ (chính) được bắt đầu song song với việc tìm kiếm trong bộ nhớ cache, vì thế nếu không thấy trong cache thì chu kỳ bộ nhớ đã được bắt đầu rồi. Chiến lược này đòi hỏi bộ nhớ phải dừng ngay lại khi tìm thấy trong cache, làm cho nó trở nên phức tạp. Thuật toán cơ bản để tìm kiếm trong bộ nhớ cache và để bắt đầu hoặc dừng việc truy cập bộ nhớ tuỳ thuộc vào kết quả của sự tìm kiếm trong cache được thực hiện bằng vi chương trình.

Có hai cách tổ chức bộ nhớ cache khác nhau một cách căn bản được người ta sử dụng, ngoài ra còn có một cách thứ ba là sự lai của hai cách trên. Trong cả 3 cách

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

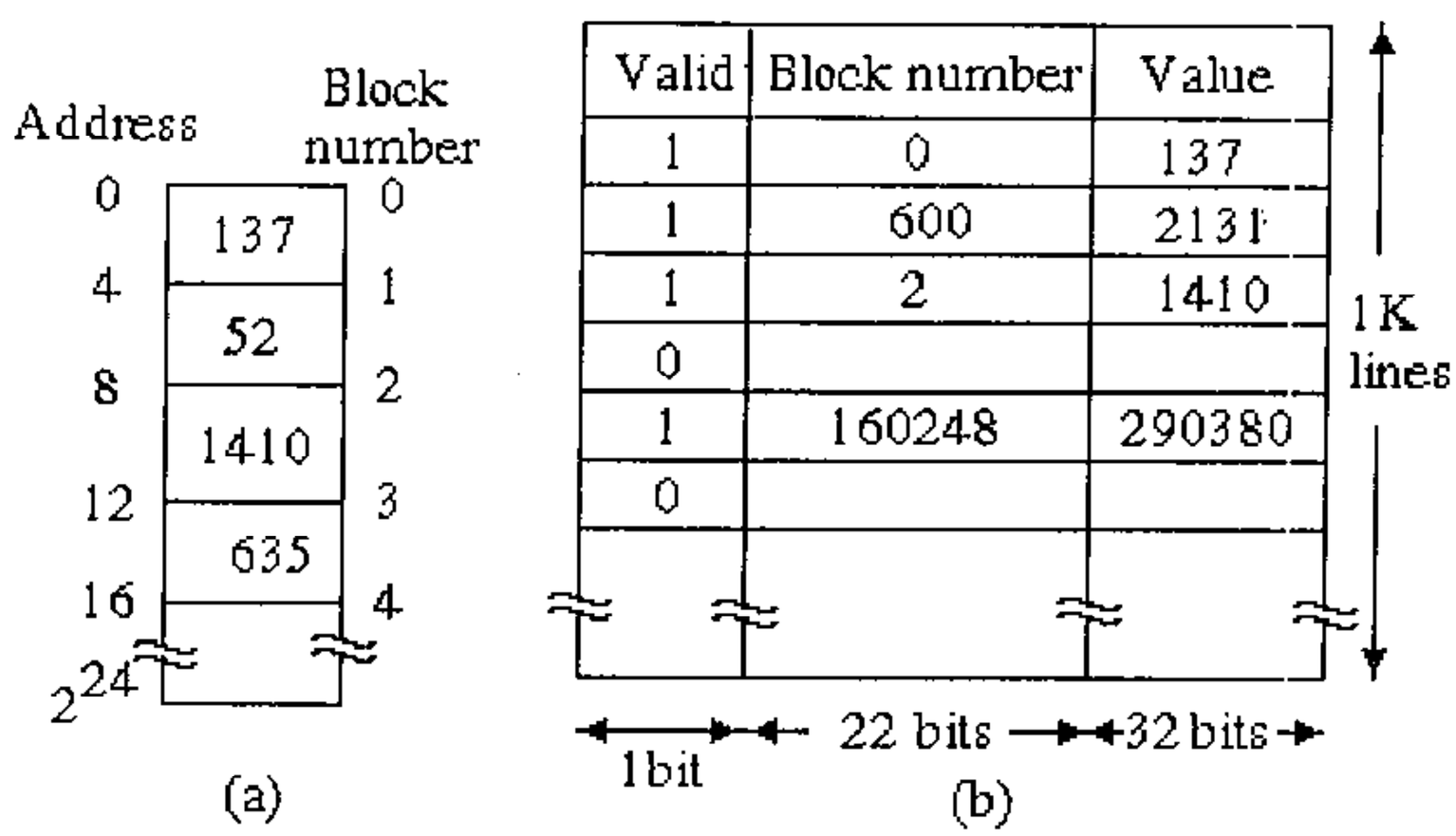
tổ chức, giả sử bộ nhớ có 2^m byte, được chia tưởng tượng thành các khối liên tiếp có kích thước b byte, tổng cộng có $2^m/b$ khối. Mỗi khối có một địa chỉ là bội số của b. Kích thước b của khối thường là số mũ của 2.

4.5.4.2 Cache liên kết (associative cache)

Kiểu cache thứ nhất được gọi là cache liên kết, trên hình 4-14 là một thí dụ về kiểu cache này. Nó bao gồm một số dòng (slot, line), mỗi dòng giống như 1 record trong ngôn ngữ lập trình Pascal và có 3 trường. Trường Value chứa nội dung một khối nhớ của bộ nhớ chính, trường Block number chứa số hiệu khối nhớ tương ứng và trường Valid kích thước 1 bit cho biết rằng hiện thời slot đó có được sử dụng hay không. Mỗi dòng của bộ nhớ cache đôi khi còn được gọi là đề mục cache (cache entry).

Thí dụ trên hình 4-14 minh hoạ một cache có 1024 slot và một bộ nhớ có 2^{24} byte được chia thành 2^{22} khối 4 byte. Trong cache loại liên kết, thứ tự đưa các giá trị vào cache là ngẫu nhiên.

Khi máy tính được reset, tất cả các trường Valid đều được đặt bằng 0, để chỉ ra rằng không có một slot nào của cache được sử dụng. Giả sử chỉ thị đầu tiên truy cập một word 32 bit tại địa chỉ 0. Vi chương trình sẽ kiểm tra tất cả các đề mục của cache để tìm một đề mục có trường Valid =1, đồng thời trường số hiệu khối - Block number = 0. Khi không tìm thấy nó sẽ phát ra tín hiệu yêu cầu sử dụng bus (bus request, xem phần trọng tài bus) để lấy word 0 từ bộ nhớ về, đặt vào trường value của một đề mục cache hiện chưa được sử dụng, ghi vào trường block number của đề mục này số hiệu khối (word có địa chỉ 0 thuộc khối 0) và sau đó đánh dấu đề mục này là đã được sử dụng bằng cách gán cho trường Valid của nó = 1. Nếu word này lại được truy cập nữa, thì lần sau nó sẽ được truy cập trong cache chứ không phải trong bộ nhớ chính, loại trừ được việc phải thao tác bus.



Hình 4-14 Thí dụ về một cách tổ chức cache a) Bộ nhớ chính có các khối 4 byte; b) Cache loại liên kết có 1024 dòng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Quá trình trên cứ tiếp diễn và càng ngày càng có nhiều đề mục cache được đánh dấu là đã sử dụng (trường Valid =1). Nếu chương trình sử dụng dưới 1024 word cho cả chương trình lẫn dữ liệu, thì cuối cùng toàn bộ chương trình và dữ liệu của nó đều xuất hiện trong cache và chương trình sẽ chạy với tốc độ cao nhất, không còn phải truy cập bộ nhớ chính qua bus nữa.

Nếu chương trình cần trên 1024 word thì đến một lúc nào đó cache sẽ đầy và một đề mục cũ phải được loại bỏ đi để lấy chỗ cho một đề mục mới. Trong thực tế việc quyết định sẽ vứt đề mục nào đi phải được thực hiện rất nhanh, cỡ na-nô giây. Trong máy VAX và nhiều máy khác người ta thực hiện vứt bỏ đề mục cache một cách ngẫu nhiên. Người ta cũng đã nghiên cứu và áp dụng một số thuật toán khác nữa, nhưng chúng ta sẽ không đi sâu vào lĩnh vực này.

Một đặc điểm của cache liên kết làm cho nó nổi bật so với các loại cache khác là trong mỗi slot có chứa số thứ tự của khối của bộ nhớ chính và nội dung của khối đó. Khi có một địa chỉ bộ nhớ được truy cập, vi chương trình bắt buộc phải tính số thứ tự khối có liên quan (việc này là dễ) sau đó tìm số thứ tự khối đó bên trong cache (việc này là khó). Để tránh khỏi phải tìm kiếm tuyến tính, trong cache liên kết người ta sử dụng một phần cứng đặc biệt, nó có thể so sánh một cách đồng thời mọi trường Block number của tất cả các đề mục cache với một số thứ tự khối được đưa ra, chứ không phải thực hiện bằng một vòng lặp của vi chương trình. Phần cứng này làm cho cache liên kết trở nên đắt.

4.5.4.3 Cache ánh xạ trực tiếp (Direct-mapped cache)

Để giảm giá thành bộ nhớ cache, người ta đã phát minh ra một loại cache khác, gọi là cache ánh xạ trực tiếp. Loại cache này tránh được việc tìm kiếm như ở cache liên kết, bằng cách đặt mỗi khối lấy từ bộ nhớ chính vào một slot mà số thứ tự của slot có thể tính được dễ dàng từ số thứ tự khối. Thí dụ, số thứ tự slot có thể bằng số thứ tự khối đem modulo cho tổng số slot của cache. (Có thể giải thích vắn tắt về phép tính modulo sử dụng ở đây như sau: đó là phép tính trên số nguyên, kết quả của việc đem số thứ tự khối chia nguyên cho số slot và lấy phần dư của phép chia.)

Thí dụ một bộ nhớ cache có 1024 slot và các khối có kích thước 4 byte (1 word), số slot cho word tại địa chỉ A là A/4 modulo 1024. Trong thí dụ trên hình 4-15a chúng ta thấy rằng các word tại các địa chỉ 0, 4096, 8192 .v.v. cùng được ánh xạ vào slot số 0 .v.v.

Cache ánh xạ trực tiếp loại bỏ được vấn đề tìm kiếm, nhưng nó cũng làm nảy sinh ra một vấn đề mới, đó là chúng ta đã tạo ra 1024 lớp tương đương, dựa trên số thứ tự khối modulo cho kích thước của bộ nhớ cache.

Làm thế nào để biết được rằng word nào trong số nhiều word cùng được ánh xạ lên 1 slot đã cho hiện đang nằm trong slot đó?. Trong thí dụ này, slot 0 có thể chứa một word bất kỳ trong số các word 0, 4096, 8192 .v.v.

Một cách để thông báo word nào đang nằm trong slot là mọi slot được bổ sung thêm một trường, gọi là trường Tag, trường này chứa một phần của địa chỉ của word

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

đã được copy từ bộ nhớ chính vào slot này của cache, phần này không thể tính được từ số slot.

- **Thí dụ minh hoạ:**

Địa chỉ bộ nhớ là một con số 24 bit; Một chỉ thị (chẳng hạn move - là một chỉ thị vận chuyển dữ liệu) tại địa chỉ 8192 chuyển 1 word tại địa chỉ 4100 tới địa chỉ 12296.

Số của khối tương ứng với địa chỉ nhớ 8192 tính được bằng cách chia nó cho 4 (kích thước khối trong thí dụ của chúng ta =4), ta nhận được 2048. Sau đó số của slot được tính bằng cách lấy 2048 modulo 1024, việc này tương tự với việc sử dụng 10 bit thấp của 2048 ($1024=2^{10}$). Ta nhận được số của slot là 0. Mười hai bit cao chứa số 2, vì vậy đó là tag. Hình 4-15a chỉ ra cache sau khi tất cả 3 địa chỉ đã được xử lý.

Hình 4-15b thể hiện việc các địa chỉ được chia ra thành các phần như thế nào. Hai bit thấp luôn luôn là 00 bởi vì cache làm việc với cả khối, chúng là bội số của kích thước khối (trong thí dụ của chúng ta là 4 byte, 4 = 100b). Tiếp theo là số slot (10 bit), và cuối cùng là Tag (12 bit). Việc xây dựng phần cứng để trực tiếp trích số slot và Tag ra khỏi các địa chỉ (24 bit, trong thí dụ này) là một việc dễ.

Slot	Valid	Tag	Value	Addresses that use this slot
0	1	2	12130	0, 4096, 8192, 12288, ...
1	1	1	170	4, 4100, 8196, 12292, ...
2	1	3	2142	8, 4104, 8200, 12296, ...
3	0			12, 4108, 8204, 12300, ...
4	0			16, 4112, 8208, 12304, ...
5	0			20, 4116, 8212, 12308, ...
≈	≈	≈	≈	
1023	1			4092, 8188, 12284, ...

a)

Bits	12	10	2
	Tag	Slot	00

b)

Hình 4-15 a) Cache ánh xạ trực tiếp với 1024 slots, mỗi slot chứa 1 khối 4 byte; b) Việc tính số slot và trường tag từ địa chỉ bộ nhớ chính 24-bit .

Việc nhiều khối cùng ánh xạ vào một slot của cache có thể gây ra vấn đề. Giả sử rằng chỉ thị move của chúng ta vận chuyển word tại địa chỉ 4100 tới địa chỉ 12292 chứ không phải 12296. Cả hai địa chỉ này đều ánh xạ vào slot 1 (4100

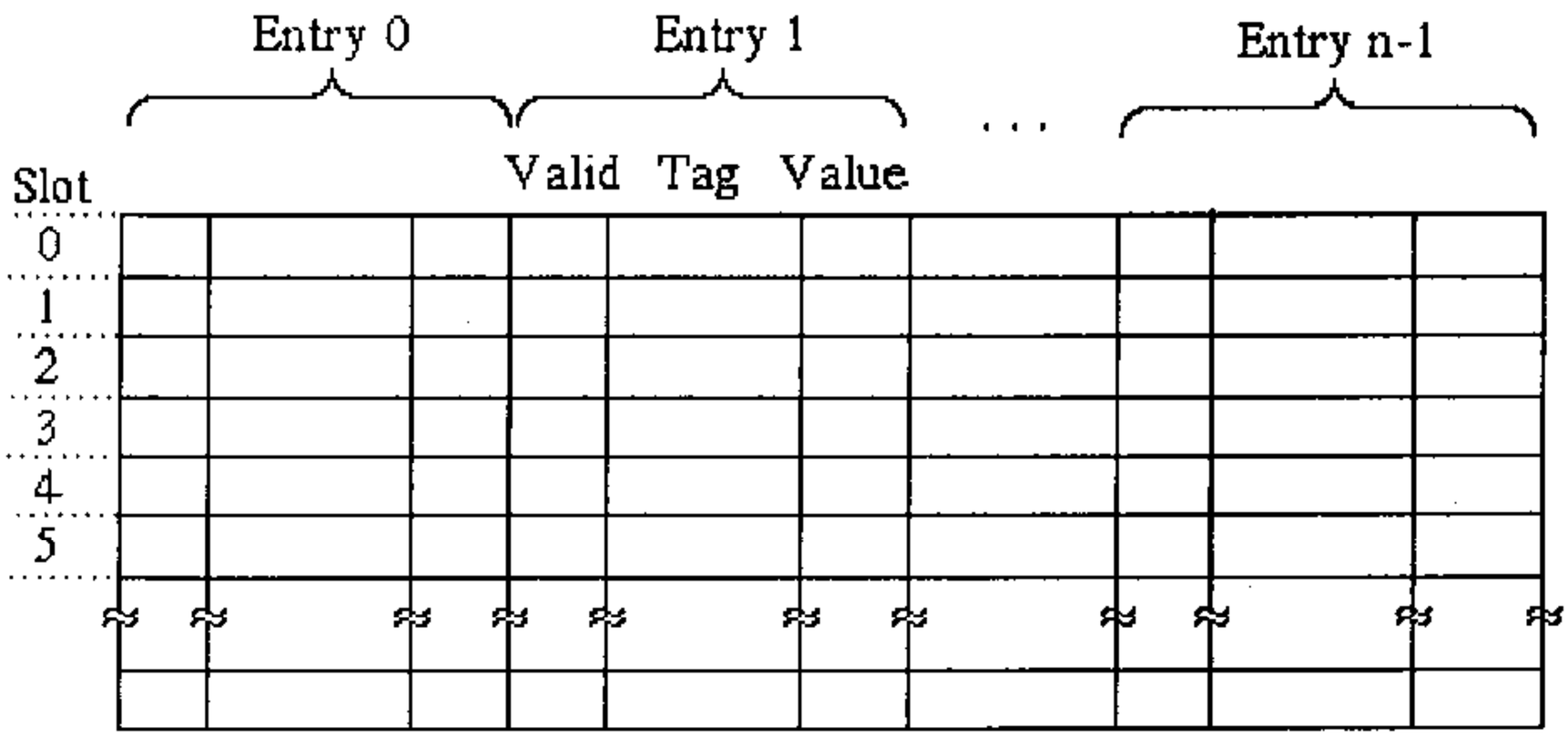
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

modulo 1024 = 12292 modulo 1024 = 4). Tuỳ thuộc vào các chi tiết của vi chương trình cụ thể, word nào có địa chỉ được tính sau sẽ được copy vào trong cache, còn word kia sẽ bị tẩy bỏ khỏi cache mặc dù nó có thể vừa mới được copy vào. Việc này sẽ làm giảm hiệu suất của cache.

- **cache liên kết thành bộ - set associative cache**

Một cách để khắc phục khó khăn nêu trên là mở rộng cache ánh xạ trực tiếp để mỗi slot có trên 1 đề mục. Thí dụ ở máy PDP-11/70 mỗi slot có hai đề mục. Những cache ánh xạ trực tiếp với mỗi slot có nhiều đề mục được gọi là **cache liên kết thành bộ - set associative cache** và được minh hoạ trên hình 4-16.

Cache liên kết và cache ánh xạ trực tiếp thực tế đều là các trường hợp đặc biệt của loại cache liên kết thành bộ. Nếu chúng ta giảm số slot của cache liên kết thành bộ xuống 1 thì tất cả các đề mục của cache là thuộc cùng một slot và chúng ta phải phân biệt chúng hoàn toàn bằng trường Tag, bởi vì tất cả chúng đều ánh xạ vào cùng một địa chỉ. Trường hợp này chính là cache loại liên kết. Nếu số đề mục trong mỗi slot n=1, chúng ta lại trở lại trường hợp của cache ánh xạ trực tiếp, trong đó mỗi slot là một đề mục.



A set associative cache with n entries per slot

Hình 4-16 Cache liên kết thành bộ với mỗi slot chứa n đề mục

4.5.4.4 So sánh cache liên kết và cache ánh xạ trực tiếp

Hai loại cache nêu trên có những điểm mạnh và điểm yếu khác nhau. Việc xây dựng loại cache ánh xạ trực tiếp đơn giản hơn, rẻ hơn và nó có thời gian truy cập nhanh hơn, bởi vì có thể tìm đúng slot dựa trên địa chỉ bộ nhớ chính.

Cache liên kết có ***tỉ lệ tìm thấy - hit ratio*** cao hơn đối với mọi số slot bởi vì không bao giờ có sự xung đột. Không thể xảy ra việc có k word quan trọng không thể đồng thời đưa vào cache do việc chúng chẳng may ánh xạ vào cùng một slot của cache.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Mỗi khi thiết kế một máy tính thực, người ta phải tăng cường việc mô phỏng cache để biết được với mỗi giá phải trả thì có được hiệu suất là bao nhiêu. Ngoài việc xác định số lượng slot, người thiết kế cũng còn phải chọn kích thước khối. Trong thí dụ, chúng ta đã sử dụng khối có kích thước bằng một word 32 bit cho đơn giản, nhưng cũng có thể chọn 2, 4, 8 hoặc nhiều word hơn nữa. Trong thực tế người ta cũng thường sử dụng như vậy.

Việc sử dụng kích thước khối lớn có ưu điểm là có tổng chi phí thời gian thấp hơn trong việc lấy về một khối 8 word so với việc lấy về 8 khối 1 word, đặc biệt nếu bus cho phép truyền khối (block transfer). Tuy nhiên nó cũng có nhược điểm là không phải mọi word mang về cache đều cần đến, vì vậy một phần của chi phí nêu trên bị bỏ phí.

4.5.4.5 Các chiến lược quản lý việc ghi vào bộ nhớ cache

Một vấn đề thiết kế cache quan trọng nữa là quản lý việc ghi như thế nào. Có hai chiến lược được người ta sử dụng rộng rãi. Chiến lược thứ nhất được gọi là ***write through***, khi một word đang nằm trong cache cần thay đổi, nó sẽ được ghi, đồng thời word tương ứng trong bộ nhớ chính cũng lập tức được ghi lại với cùng giá trị. Cách tiếp cận này đảm bảo rằng các đề mục cache luôn luôn giống với các đề mục tương ứng của bộ nhớ, nói cách khác khi bản sao (word trong cache) thay đổi thì bản gốc (word tương ứng trong bộ nhớ chính) cũng lập tức được cập nhật.

Chiến lược thứ hai được gọi là *copy back*, nó không cập nhật bộ nhớ chính ngay khi cache bị thay đổi. Thay vào đó bộ nhớ chính chỉ được cập nhật khi đề mục tương ứng trong cache bị loại bỏ để giải phóng chỗ cho một đề mục khác. Khi sử dụng copy back, cần phải sử dụng 1 bit trong mỗi đề mục cache để cho biết đề mục cache đã bị thay đổi từ khi nó được nạp vào trong cache hay chưa.

Hai chiến lược ghi vào cache vừa được trình bày có ưu và nhược điểm của nó. Write through rõ ràng là làm sinh ra nhiều sự vận chuyển trên bus hơn là copy back. Nếu sử dụng copy back, người thiết kế cache phải giải quyết một số vấn đề phát sinh. Thí dụ nếu CPU bắt đầu một thao tác vận chuyển 1 word từ bộ nhớ đến đĩa, trong khi đó word trong bộ nhớ chính còn chưa được cập nhật còn word tương ứng hiện ở trong cache đã bị thay đổi, thì dữ liệu không đúng sẽ được ghi lên đĩa. Vấn đề này có thể tránh được nhưng nó sẽ làm cho hệ thống thêm phức tạp.

Nếu tỉ số đọc/ghi của chương trình là rất cao, sử dụng write through là đơn giản nhất và chúng ta chấp nhận việc thỉnh thoảng phải vận chuyển trên bus.

Tuy nhiên nếu tỉ số ghi/đọc của chương trình là rất cao, sử dụng copy back là tốt hơn và chúng ta buộc chính vi chương trình xoá bỏ toàn bộ cache trước khi bắt đầu một thao tác I/O.

Một vấn đề thiết kế khác liên quan đến việc ghi là sẽ phải làm gì nếu word tại địa chỉ cần ghi chưa có trong cache (sự kiện này được gọi là ***cache miss***). Có một chiến lược là lấy word cần ghi từ bộ nhớ chính về cache rồi cập nhật nó trong cache, nếu sử dụng write through thì viết trở lại bộ nhớ chính giá trị mới này. Một chiến

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

lược khác là viết trực tiếp vào bộ nhớ chứ không lấy word từ bộ nhớ về cache trừ khi đọc. Hiện nay người ta sử dụng cả hai phương pháp nêu trên.

Chúng ta đã xem xét mọi vấn đề, sử dụng cache là một việc quan trọng để nâng cao hiệu suất của hệ thống. Từ trước đến nay hầu hết các máy tính loại trung và loại lớn đều sử dụng một dạng cache nào đó. Ngày nay ngay cả các máy tính cá nhân (PC) nói chung cũng có bộ nhớ cache.

4.6 TÓM TẮT MỨC VI CHƯƠNG TRÌNH

Tại mức vi chương trình có thể coi CPU bao gồm hai thành phần căn bản là: đường dữ liệu - data path và đơn vị điều khiển - control section. Đường dữ liệu chủ yếu bao gồm một bộ nhớ tạm là một tập thanh ghi và một phần là ALU và shifter. Một chu kỳ bao gồm việc tách một số toán hạng khỏi bộ nhớ tạm, cho chúng chạy qua ALU/shifter và có thể bao gồm cả việc chứa kết quả trở lại vào bộ nhớ tạm.

Đơn vị điều khiển có chứa bộ nhớ điều khiển - control store, đó là nơi chứa vi chương trình. Mỗi vi chỉ thị trong control store điều khiển các cổng trong đường dữ liệu trong suốt một vi chu kỳ. Trong thí dụ đã trình bày vi chu kỳ được chia thành các chu kỳ con (subcycle) được sinh ra bởi một đồng hồ. Trong chu kỳ con đầu tiên, một vi chỉ thị được lấy từ control store về và đặt vào một thanh ghi bên trong. Trong chu kỳ con thứ hai các giá trị vào - input của ALU được chốt lại để giữ cho chúng không thay đổi trong suốt thời gian thực hiện vi chỉ thị. Trong chu kỳ con thứ ba ALU và shifter thực hiện công việc của chúng. Cuối cùng, trong chu kỳ con thứ tư kết quả có thể được chứa trở lại vào bộ nhớ tạm nếu vi chỉ thị tiếp theo cần sử dụng đến nó.

Để định trình tự các vi chỉ thị, một số ít máy tính có con đếm chương trình riêng tại mức vi chương trình. Tuy nhiên nói chung các vi chỉ thị có chứa địa chỉ cơ sở của vi chỉ thị sau nó. Địa chỉ cơ sở này thường được OR với một vài bit trạng thái để sinh ra địa chỉ cuối cùng. Trong nhiều máy, địa chỉ cơ sở của một vi chỉ thị được kết hợp với các bit trạng thái do vi chỉ thị trước đó sinh ra để tạo ra một địa chỉ nhảy, địa chỉ này chỉ có ảnh hưởng đối với vi chỉ thị tiếp theo.

Các vi chỉ thị có thể được tổ chức theo chiều ngang, trong đó mỗi bit dùng làm một tín hiệu điều khiển. Chúng cũng có thể được tổ chức theo chiều đứng, trong đó chỉ có một số ít trường và đòi hỏi việc giải mã phức tạp. Ngoài ra cũng còn có cách tổ chức trung gian giữa hai cách trên.

Cách tổ chức theo chiều ngang dẫn đến các máy tính có word dài, chạy nhanh và tổ chức song song ở mức độ cao. Cách tổ chức theo chiều đứng dẫn đến các máy tính chạy chậm hơn và có bộ nhớ điều khiển nhỏ hơn.

Có thể nâng cao hiệu suất bằng nhiều kỹ thuật khác nhau. Một trong các kỹ thuật đó là lập siêu vi chương trình (nanoprogramming), trong đó vi chương trình chứa các con trỏ ngắn để trỏ đến các siêu vi chỉ thị (nanoinstruction) dài, các siêu vi chỉ thị này thực tế điều khiển các cổng. Nanoprogramming làm giảm độ lớn không

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

gian chip dành cho ROM chứa vi chương trình, nhưng giá phải trả là thi hành chậm hơn.

Các cách khác để nâng cao hiệu suất là: sử dụng tín hiệu đồng hồ có chu kỳ thay đổi được, rẽ nhánh nhiều đường (multiway branching), tạo đường ống (pipelining) và sử dụng bộ nhớ cache. Tuy nhiên các kỹ thuật nêu trên cũng đều dẫn đến sự tăng độ phức tạp.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

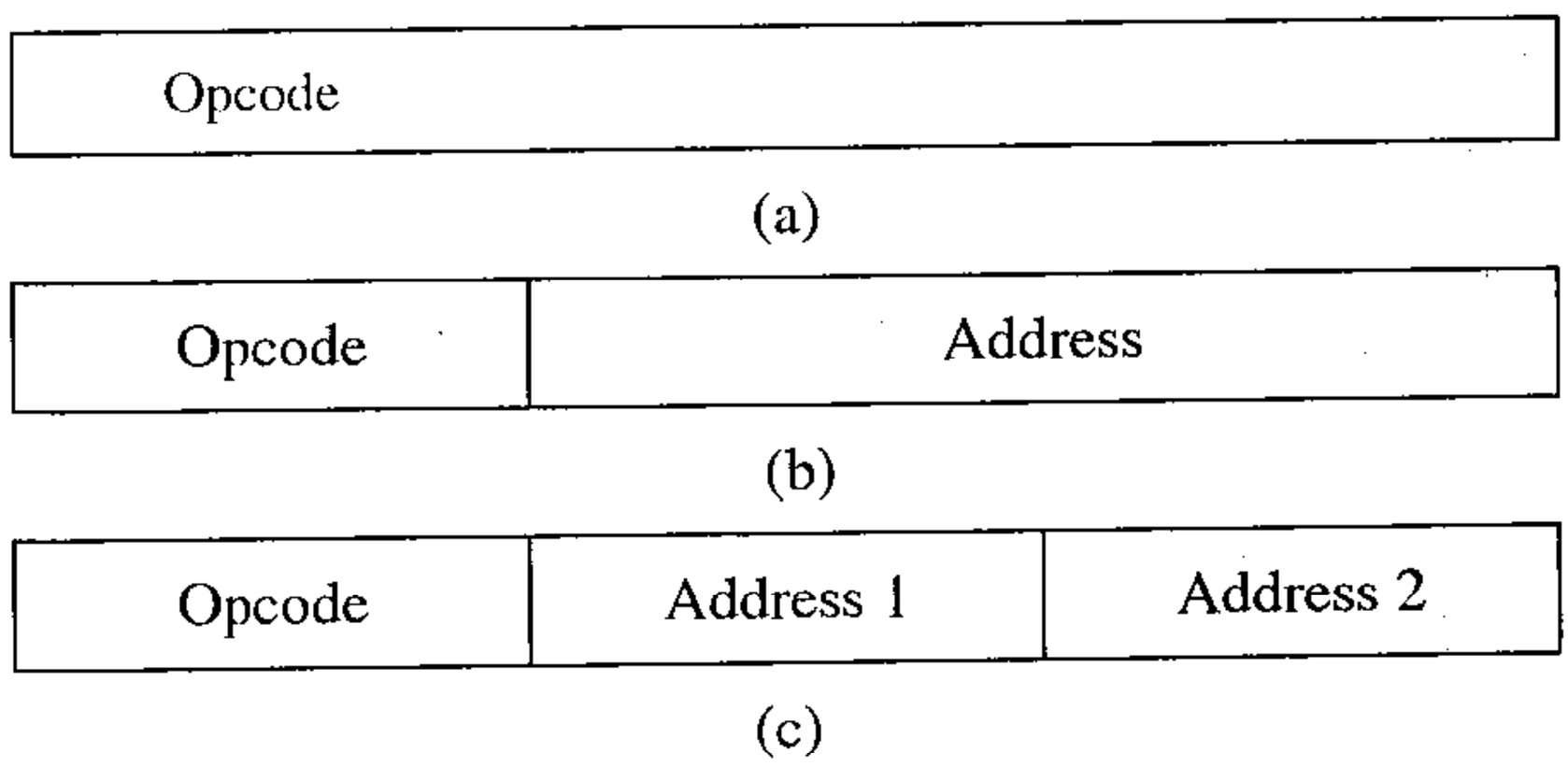
CHƯƠNG 5 MỨC MÁY THÔNG THƯỜNG

Trong lịch sử phát triển của máy tính, mức 2 luôn được thiết kế trước mọi mức khác và do đó được gọi là Mức máy thông thường (Conventional Machine Level). Tại mức này có một ngôn ngữ thường được nhiều người gọi một cách không đúng là **ngôn ngữ máy - machine language**. Sở dĩ có sự gọi sai như vậy là do ở nhiều máy tính vi chương trình được ghi trong ROM, do đó người sử dụng không thể viết chương trình cho mức 1. Hơn nữa ngay cả trong các máy mà người sử dụng có thể lập vi chương trình được, độ phức tạp rất cao của kiến trúc mức 1 cũng thường làm cho mọi người lập trình e ngại, không thực hiện lập trình cho máy mức 1.

5.1 KHUÔN DẠNG CỦA CHỈ THỊ

Chương trình bao gồm một dãy các chỉ thị, mỗi chỉ thị chỉ rõ một hành động cụ thể nào đó. Một phần của chỉ thị, được gọi là ***mã phép toán - operation code*** hay gọi ngắn gọn là ***opcode*** cho biết hành động nào sẽ được thi hành. Nhiều chỉ thị có chứa hoặc chỉ ra vị trí của dữ liệu được chỉ thị sử dụng (address). Thí dụ một chỉ thị so sánh 2 ký tự để xem xem chúng có giống nhau không bắt buộc phải chỉ rõ những ký tự nào sẽ được so sánh với nhau. Việc chỉ ra nơi đặt toán hạng, nghĩa là địa chỉ của chúng được gọi là đánh địa chỉ và sẽ được thảo luận ở mục 5.3.

Trên hình 5-01 là một số khuôn dạng điển hình của các chỉ thị mức 2. Ở một số máy mức 2, mọi chỉ thị đều có cùng chiều dài; còn ở một số máy khác có thể có 2 hoặc 3 chiều dài khác nhau. Ngoài ra các chỉ thị có thể ngắn hơn, bằng hoặc dài hơn chiều dài từ (word).



Hình 5-01 Ba khuôn dạng chỉ thị điển hình a) Chỉ thị không có địa chỉ; b) chỉ thị có một địa chỉ; c) chỉ thị có 2 địa chỉ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

5.1.1 Các tiêu chuẩn thiết kế khuôn dạng chỉ thị

Khi các nhà thiết kế máy tính lựa chọn khuôn dạng chỉ thị cho chiếc máy tính của họ, họ buộc phải xem xét một số yếu tố hay là các tiêu chuẩn thiết kế sau đây.

- **Tiêu chuẩn thiết kế thứ 1:** ***Chỉ thị ngắn tốt hơn chỉ thị dài.***

Đây là tiêu chuẩn quan trọng nhất. Một chương trình bao gồm n chỉ thị 16 bit chiếm một không gian nhớ chỉ bằng một nửa chương trình gồm n chỉ thị 32 bit. Sử dụng bộ nhớ phải mất tiền, vì vậy những nhà thiết kế không muốn lãng phí nó.

Mỗi bộ nhớ có một tốc độ truyền thông tin khác nhau, tốc độ đó được xác định bởi công nghệ và sự thiết kế bộ nhớ. Nếu tốc độ truyền của một bộ nhớ cụ thể là t bps (bit per second) và chiều dài chỉ thị trung bình là r bit, bộ nhớ có thể truyền đi nhiều nhất là t/r chỉ thị trong một giây. Vì vậy mà tốc độ thực hiện các chỉ thị, cũng tức là tốc độ của bộ xử lý phụ thuộc vào chiều dài của chỉ thị. Chỉ thị ngắn hơn có nghĩa là bộ xử lý chạy nhanh hơn.

Nếu thời gian cần thiết để thực hiện một chỉ thị dài hơn so với thời gian cần thiết để lấy nó từ bộ nhớ về, thì thời gian lấy chỉ thị về sẽ không phải là quá quan trọng. Tuy nhiên đối với các CPU nhanh, bộ nhớ thường giống như là cái cổ chai, vì vậy việc tăng số chỉ thị được lấy về trong một giây là một chỉ tiêu thiết kế quan trọng.

- **Tiêu chuẩn thiết kế thứ hai:** ***Có đủ chỗ trong chỉ thị để biểu diễn tất cả các thao tác.***

Một máy tính với 2^n thao tác không thể sử dụng chỉ thị hẹp hơn n bit. Bởi vì như vậy sẽ không có đủ chỗ trong mã lệnh (opcode) để chỉ ra chỉ thị nào được gọi đến.

- **Tiêu chuẩn thiết kế thứ ba:** ***Độ dài word của máy là bội số nguyên của chiều dài ký tự.***

Nếu mã ký tự có k bit thì chiều dài word cần phải là k, 2k, 3k, 4k, 5k .v.v.; để tránh lãng phí không gian nhớ khi chứa các ký tự.

Tất nhiên có thể chứa 3,5 ký tự trong một word, nhưng việc đó sẽ làm giảm hiệu quả một cách trầm trọng khi truy cập các ký tự. Hạn chế vừa nêu về chiều dài word cũng ảnh hưởng đến chiều dài chỉ thị, bởi vì một chỉ thị tốt nhất là chiếm một số nguyên các byte hoặc một số chỉ thị nằm vừa trong một word.

- **Tiêu chuẩn thiết kế thứ tư:** ***Số bit trong trường địa chỉ càng ngắn càng tốt.***

Tiêu chuẩn này liên quan đến việc chọn kích thước ô nhớ. Chúng ta thử xem xét một thiết kế máy với ký tự 8 bit và một bộ nhớ chính phải chứa được 2^{16} ký tự. Nhà thiết kế có thể chọn để gán các địa chỉ liên tiếp cho các nhóm 8, 16, 24 hoặc 32 bit...

Nếu chọn ô nhớ có kích thước 8 bit (1 byte) thì cần sử dụng các giá trị địa chỉ trong miền 0..65535, tương ứng với một số nhị phân 16 bit.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Nếu chọn ô nhớ có kích thước 32 bit (4 byte) thì cần sử dụng các giá trị địa chỉ trong miền 0..16383, tương ứng với một số nhị phân 14 bit.

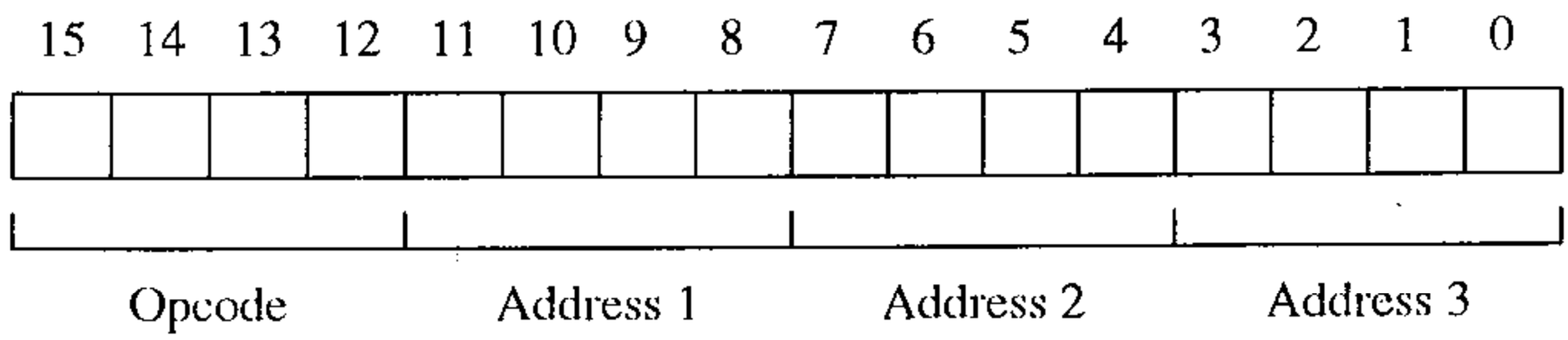
Việc chọn kích thước ô nhớ 32 bit có lợi là để đánh địa chỉ bộ nhớ 2^{16} byte chỉ cần 2^{14} địa chỉ khác nhau, như vậy chiều dài trường địa chỉ trong các chỉ thị chỉ cần 14 bit, so với 16 bit nếu chọn kích thước ô nhớ 8 bit. Địa chỉ ngắn hơn cũng có nghĩa là chỉ thị ngắn hơn, chiếm ít không gian nhớ hơn và thời gian lấy chỉ thị cũng ngắn hơn.

Thực tế có những máy tính mà chiều dài word chỉ có 1 bit (thí dụ máy Burroughs B1700), lại cũng có máy tính mà word rất dài, tới 60 bit (thí dụ máy CDC Cyber).

5.1.2 Việc mở rộng mã chỉ thị

Trong phần này chúng ta nghiên cứu những sự cân nhắc, thoả hiệp liên quan đến mã phép toán - opcode và địa chỉ. Chúng ta xem xét một chỉ thị dài (n+k) bit, trong đó opcode dài k bit và địa chỉ là n bit. Với khuôn dạng này có thể có 2^k chỉ thị khác nhau và có thể đánh địa chỉ được 2^n ô nhớ. Cũng với chỉ thị dài (n+k) bit đó, nếu opcode dài (k-1) bit và địa chỉ dài (n+1) bit thì số chỉ thị có thể có chỉ còn bằng một nửa nhưng số ô nhớ có thể đánh địa chỉ lại tăng gấp đôi.

Khái niệm về mã phép toán mở rộng sẽ được thí dụ sau đây làm rõ. Chúng ta xem xét một máy có các chỉ thị dài 16 bit như trên hình 5-02. Thí dụ chúng ta sử dụng một thiết kế khuôn dạng chỉ thị trong đó mỗi chỉ thị có trường opcode 4 bit và 3 trường địa chỉ 4 bit, như vậy có được 16 chỉ thị 3 địa chỉ. Các con số lấy làm thí dụ này là có lý đối với một máy có 16 thanh ghi, vì có thể dùng 4 bit để đánh địa chỉ các thanh ghi.



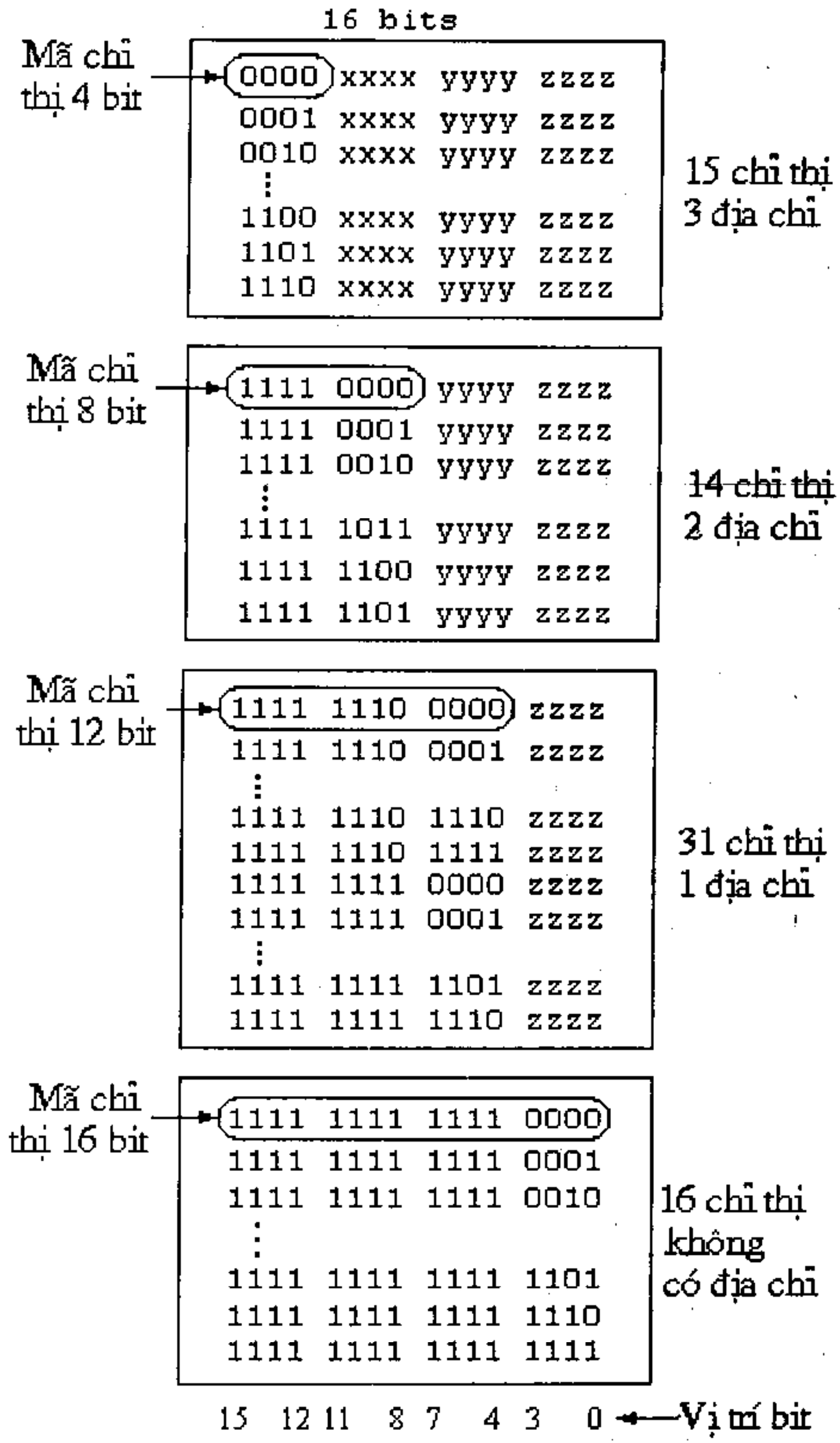
Hình 5-02 Một chỉ thị có opcode 4 bit và 3 trường địa chỉ 4 bit.

Tuy nhiên nếu các nhà thiết kế cần đến 15 chỉ thị 3 địa chỉ, 14 chỉ thị 2 địa chỉ, 31 chỉ thị 1 địa chỉ và 16 chỉ thị không có địa chỉ thì họ có thể làm như sau:

- Sử dụng các opcode có giá trị từ 0 đến 14 làm các chỉ thị 3 địa chỉ, nhưng họ lại thông dịch opcode 15 theo cách khác (xem hình 5-03).
- Opcode 15 (dạng nhị phân là 1111) có ý nghĩa là opcode được chứa trong các bit từ 8 tới 15., chứ không phải trong các bit từ 12 đến 15. Các bit từ 0..3 và từ 4..7 tạo thành 2 địa chỉ, như bình thường.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Mười bốn chỉ thị 2 địa chỉ đều có 4 bit trái là 1111 và các con số từ 0000 đến 1101 nằm trong các bit từ 8 đến 11.



Một mã lệnh mở rộng cho phép tạo ra 15 chỉ thị 3 địa chỉ, 14 chỉ thị 2 địa chỉ, 31 chỉ thị 1 địa chỉ và 16 chỉ thị 0 địa chỉ. Các trường được đánh dấu xxxx, yyyy và zzzz là các trường địa chỉ 4 bit.

Hình 5-03 Một thí dụ về mở rộng mã chỉ thị

- Các chỉ thị có 4 bit trái là 1111 và các bit từ 8..11 là 1110 hoặc 1111 sẽ được xử lý một cách khác. Chúng sẽ được coi như có opcode nằm trong các bit từ 4..15. Kết quả là có 32 opcode mới. Bởi vì người ta chỉ cần 31 chỉ thị nên opcode 1111 1111 1111 được thông dịch để chỉ ra rằng opcode thực sự nằm trong các bit từ 0..15, đó là opcode của 16 chỉ thị không có địa chỉ.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Nếu chúng ta cứ làm tiếp theo cách đã thảo luận, opcode sẽ càng ngày càng dài ra; điều đó có nghĩa là các chỉ thị 3 địa chỉ có opcode dài 4 bit, các chỉ thị 2 địa chỉ có opcode dài 8 bit, các chỉ thị 1 địa chỉ có opcode dài 12 bit và các chỉ thị không có địa chỉ có opcode dài 16 bit.

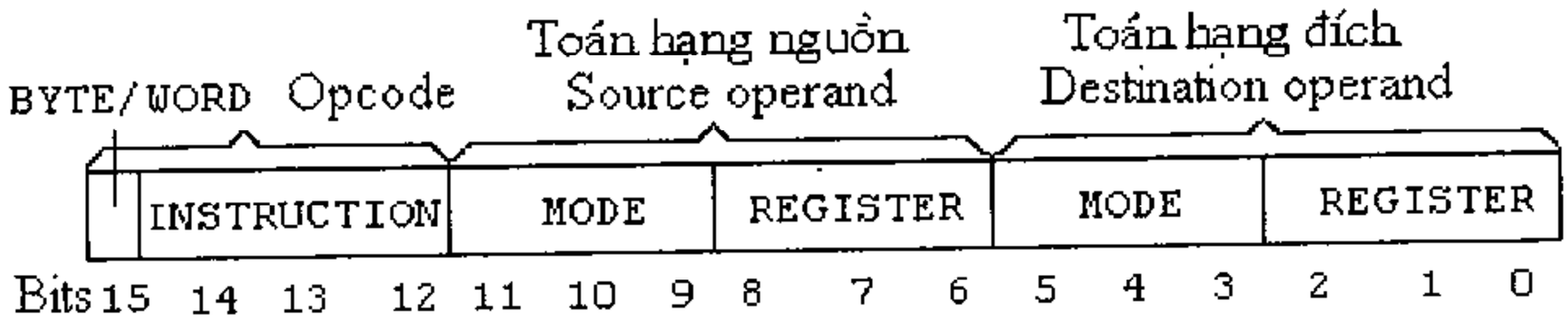
Các mã lệnh mở rộng trong thực tế không có dáng vẻ đơn giản và đẹp mắt như trong thí dụ của chúng ta, chúng thường cần có nhiều loại chiều dài và khuôn dạng khác nhau.

5.1.3 Thí dụ về khuôn dạng chỉ thị của máy PDP-11

Trong phần này chúng ta sẽ tìm hiểu khuôn dạng chỉ thị ở mức máy thông thường của máy PDP-11. Chúng ta đưa ra máy PDP-11 bởi vì tập chỉ thị của nó được người ta xem là một điển hình mẫu mực về tính đơn giản và tính hợp lý.

Đa số các chỉ thị 2 địa chỉ của PDP-11 được lập mã như trên hình 5-04. Mỗi chỉ thị bao gồm một trường opcode 4 bit và 2 trường địa chỉ 6 bit. Bit trái nhất của trường opcode chỉ rõ việc chỉ thị thao tác trên byte hay trên word. Các trường địa chỉ lại được chia tiếp thành một trường mode 3 bit và một trường register 3 bit (PDP-11 có 8 thanh ghi). Trường mode cho biết toán hạng nằm trong thanh ghi hay nằm trong bộ nhớ hay là một hằng.v.v. có 8 mode giống nhau cho toán hạng nguồn và cho toán hạng đích. Mọi mã chỉ thị (opcode) đều có thể sử dụng bất cứ toán hạng nguồn và toán hạng đích nào.

Một tập chỉ thị như các chỉ thị nêu trên của PDP-11, trong đó phương pháp chỉ ra địa chỉ của toán hạng độc lập với opcode được các nhà viết chương trình dịch yêu thích, vì nó làm cho công việc của họ đơn giản hơn.



Hình 5-04 Việc lập mã chỉ thị ở máy PDP-11

Đối với một số ít chỉ thị khác, trong đó có các chỉ thị 1 toán hạng, PDP-11 sử dụng một cách tạo mã lệnh mở rộng, theo cách này các opcode dạng x111 được sử dụng để tránh việc phải sử dụng các mã lệnh dài hơn. Với hầu hết các chỉ thị 1 toán hạng, PDP-11 sử dụng mã lệnh 10 bit và một trường mode/register 6 bit, như vậy độ dài chỉ thị vẫn là 16 bit giống như các chỉ thị 2 toán hạng. Trường mã lệnh 10 bit bao gồm 4 bit của trường opcode cộng với 6 bit của trường mode/register thứ nhất như ở hình vẽ 5-04.

Các chỉ thị PDP-11 có đánh địa chỉ bộ nhớ, chúng có 1 hay 2 word 16 bit đi theo sau chỉ thị để chỉ ra các địa chỉ này.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

5.2 VIỆC ĐÁNH ĐỊA CHỈ

Các chỉ thị có thể được phân loại theo số lượng địa chỉ mà chúng sử dụng, các địa chỉ này được dùng để chỉ ra vị trí của các toán hạng mà chỉ thị cần đến.

Chúng ta cũng cần luôn nhớ rằng tập hợp các thanh ghi được đánh số của CPU, thực tế cũng tạo thành một bộ nhớ tốc độ cao và xác định một không gian địa chỉ. Một chỉ thị cộng thanh ghi 1 với thanh ghi 2 cũng là một chỉ thị 2 địa chỉ bởi vì chỉ thị này phải chỉ ra các thanh ghi nào sẽ được cộng lại với nhau cũng giống như chỉ thị cộng 2 word bộ nhớ phải chỉ rõ đó là các word nào.

Những chỉ thị chỉ ra một, hai hoặc ba địa chỉ đều là bình thường cả. Nhiều máy làm các phép tính số học chỉ sử dụng có một địa chỉ, khi đó nó sử dụng một thanh ghi đặc biệt gọi là accumulator để chứa toán hạng thứ hai. Trong những máy như vậy, địa chỉ luôn luôn là địa chỉ của một word bộ nhớ m là nơi đặt toán hạng.

Chỉ thị cho phép tính cộng chỉ ra địa chỉ m có hiệu quả như sau:

accumulator := accumulator + memory[m]

Các chỉ thị cộng 2 địa chỉ sử dụng một địa chỉ cho nguồn và địa chỉ kia cho đích. Khi đó nguồn (source) được cộng vào đích (destination):

destination := destination + source

Các chỉ thị 3 địa chỉ chỉ ra hai nguồn và 1 đích, 2 nguồn được cộng lại và đặt vào đích.

5.2.1 Đánh địa chỉ tức thời- Immediate Addressing

Cách thức đơn giản nhất để một chỉ thị chỉ ra một toán hạng là ***đặt luôn toán hạng vào trường địa chỉ của chỉ thị*** chứ không đặt vào đó địa chỉ của toán hạng hay là một thông tin khác mô tả việc toán hạng nằm ở đâu. Một toán hạng như vậy được gọi là toán hạng tức thời bởi vì nó được lấy từ bộ nhớ về một cách tự động đồng thời với việc lấy chỉ thị về; chính vì vậy nó lập tức sẵn có để sử dụng.

Việc đánh địa chỉ tức thời có ưu điểm là không cần thêm một lần truy cập bộ nhớ nữa để lấy toán hạng về. Nhưng nó cũng có nhược điểm là kích thước của toán hạng bị hạn chế bằng kích thước của trường địa chỉ. Trong một chỉ thị sử dụng địa chỉ 3 bit (tức là trường register), các toán hạng bị hạn chế là 3 bit, điều này làm hạn chế tính khả dụng.

5.2.2 Đánh địa chỉ trực tiếp - Direct Addressing

Đó là một phương pháp đơn giản khác để chỉ ra một toán hạng, bằng cách ***đặt địa chỉ của word bộ nhớ nơi đặt toán hạng vào trường địa chỉ của chỉ thị***. Tất nhiên phải có cách để máy tính có thể biết được địa chỉ nào là tức thời, địa chỉ nào là trực tiếp. Nói chung có các cách tiếp cận: sử dụng các opcode khác nhau hoặc sử dụng một mode đánh địa chỉ đặc biệt cho mỗi loại toán hạng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Việc đánh địa chỉ trực tiếp cần một lần truy cập bộ nhớ để lấy toán hạng về. Trong trường hợp tổng quát bộ nhớ gồm cả các thanh ghi chứ không phải chỉ là bộ nhớ chính.

5.2.3 Đánh địa chỉ thanh ghi - Register Addressing

Trong kiểu đánh địa chỉ này, ***trường địa chỉ của chỉ thị chứa số của thanh ghi đang chứa toán hạng***. Việc đánh địa chỉ thanh ghi về mặt khái niệm là giống việc đánh địa chỉ trực tiếp.

Một máy có 16 thanh ghi và 65536 word bộ nhớ thực tế có hai không gian địa chỉ. Chúng ta có thể hình dung một địa chỉ trong một máy như vậy sẽ có 2 phần:

- Một bit cho biết chỉ thị đang cần một word thanh ghi hay là một word bộ nhớ
- Một trường địa chỉ cho biết cần word thanh ghi hay word bộ nhớ nào.

Bởi vì số thanh ghi ít hơn số word bộ nhớ nên người ta thường sử dụng các chỉ thị có khuôn dạng khác nhau cho toán hạng thanh ghi và cho toán hạng bộ nhớ.

Các máy được thiết kế có các thanh ghi vì 2 lý do sau: (a) Các thanh ghi hoạt động nhanh hơn bộ nhớ chính và (b) số lượng thanh ghi là rất ít vì vậy cũng chỉ cần một số ít bit để đánh địa chỉ chúng.

5.2.4 Đánh địa chỉ gián tiếp- Indirect Addressing

Đánh địa chỉ trực tiếp là cách trong đó trường địa chỉ của thỉ thị chỉ ra word bộ nhớ nào hay thanh ghi nào chứa toán hạng. Đánh địa chỉ gián tiếp là cách trong đó ***trường địa chỉ của chỉ thị chỉ ra word bộ nhớ nào hoặc thanh ghi nào có chứa địa chỉ của toán hạng***. Word nhớ hoặc thanh ghi đó chính là một con trỏ, nó trỏ tới toán hạng.

Việc đánh địa chỉ gián tiếp cần hai lần truy cập bộ nhớ, lần thứ nhất để lấy con trỏ về, lần thứ hai để lấy toán hạng về.

5.2.5 Chỉ số - Indexing

Nhiều thuật toán đòi hỏi việc thực hiện một số thao tác trên dãy các cấu trúc dữ liệu được chứa trong các vị trí nhớ liên tiếp. Thí dụ chúng ta xem xét một khối gồm n từ máy chiếm các vị trí nhớ:

A, A+1, A+2, ..., A+N-1

Chúng cần phải được chuyển tới các vị trí nhớ:

B, B+1, B+2, ...,B+N-1

Giả sử rằng máy có chỉ thị:

MOVE A,B

Chỉ thị này chuyển nội dung vị trí nhớ A tới vị trí nhớ B và máy có thể thi hành chỉ thị này, sau đó sửa lại chính chỉ thị đó thành:

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

MOVE A+1,B+1

rồi thi hành chỉ thị này, rồi lại sửa, lại thi hành ... cứ lặp lại chu trình này cho tới khi tất cả n word được copy xong.

Mặc dầu các chương trình có thể sửa đổi chính nó có một thời đã từng là phổ biến, nhưng ngày nay cách lập trình đó bị xem là dở, bởi vì việc gỡ rối (debug) các chương trình loại này và dùng chung cho nhiều người sử dụng sẽ gặp nhiều khó khăn.

Bài toán copy cũng có thể giải quyết được bằng cách đánh địa chỉ gián tiếp. Một thanh ghi hay một word bộ nhớ sẽ được nạp địa chỉ A; một thanh ghi hay word bộ nhớ khác sẽ được nạp địa chỉ B. Chỉ thị MOVE sử dụng chúng như là 2 con trỏ. Mỗi lần copy xong một word, từng con trỏ được tăng lên 1. Con trỏ là một phần của dữ liệu chứ không phải là một phần của chương trình do đó tất nhiên những người khác dùng chung chương trình này không động đến.

Một cách giải quyết thứ hai là sử dụng một hoặc vài thanh ghi, chúng được gọi là các thanh ghi chỉ số - index register, chúng làm việc như mô tả sau đây.

Các địa chỉ sẽ có hai phần: một con số của thanh ghi chỉ số và một hằng số. Địa chỉ toán hạng là tổng của hằng số đó và nội dung của thanh ghi chỉ số. Trong thí dụ trên nếu cả hai địa chỉ đều được đánh chỉ số sử dụng một thanh ghi chỉ số đang chứa số nguyên k, thì chỉ thị MOVE A,B sẽ chuyển nội dung của vị trí nhớ A+k tới B+k. Bằng cách khởi tạo cho thanh ghi chỉ số giá trị 0 và tăng nó lên bằng kích thước một word mỗi khi copy xong một word, thì chúng ta chỉ cần sử dụng một thanh ghi cho vòng lặp copy. Hơn nữa việc tăng giá trị của thanh ghi lại nhanh hơn việc tăng giá trị của ô nhớ.

Việc đánh chỉ số được sử dụng rộng rãi để đánh địa chỉ một trường tại một khoảng cách đã biết tính từ điểm đầu của một cấu trúc dữ liệu đã cho. Các biến cục bộ trong một thủ tục được truy cập theo cách này.

Việc phải tăng hoặc giảm giá trị thanh ghi chỉ số trước hoặc sau khi nó được sử dụng là việc thường xuyên xảy ra, vì vậy trong một số máy tính người ta đưa ra các chỉ thị hoặc phương thức (mode) đánh địa chỉ chuyên biệt, hoặc thậm chí là các thanh ghi chỉ số đặc biệt, tự chúng có thể tăng hoặc giảm. Việc sửa đổi tự động một thanh ghi chỉ số được gọi là ***đánh chỉ số tự động - autoindexing***.

5.2.6 Đánh địa chỉ stack

Chúng ta đã nêu ra nhận xét rằng chỉ thị máy càng ngắn càng tốt để tiết kiệm thời gian của CPU và tiết kiệm bộ nhớ. Giới hạn cuối cùng của việc giảm chiều dài địa chỉ là làm cho chỉ thị không còn trường địa chỉ nữa, chỉ còn có opcode. Thật đáng ngạc nhiên là điều này có thể thực hiện được bằng cách sử dụng một cấu trúc dữ liệu có tên gọi là stack.

Stack chứa các phần tử dữ liệu (words, characters, bits, .v.v..) theo trật tự liên tiếp trong bộ nhớ. Phần tử đầu tiên được đẩy lên stack được gọi là nằm ở đáy của

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

stack, phần tử sau cùng vừa mới được đẩy lên stack được gọi là nằm ở đỉnh của stack. Mỗi stack được gắn với một thanh ghi hay word bộ nhớ chứa địa chỉ đỉnh của stack, nó được gọi là con trỏ stack - **stack pointer**.

5.3 CÁC KIỂU CHỈ THỊ

Có thể phân chỉ thị thành các kiểu sau:

1. Các chỉ thị vận chuyển dữ liệu

   Copy dữ liệu từ nơi này sang nơi khác là thao tác cơ bản nhất trong các thao tác.

2. Các chỉ thị cho phép toán hai ngôi

   Đó là các phép toán kết hợp 2 toán hạng để sinh ra một kết quả.

3. Các chỉ thị cho phép toán một ngôi

   Đó là các phép toán chỉ có một toán hạng và chỉ sinh ra một kết quả. Bởi vì so với loại chỉ thị trên, nó bớt đi được một địa chỉ, nên thường là ngắn hơn.

4. Các chỉ thị so sánh và nhảy có điều kiện

   Hầu như mọi chương trình đều cần thực hiện so sánh các dữ liệu của nó để dựa vào kết quả thực hiện rẽ nhánh chương trình.

5. Các chỉ thị gọi chương trình con

   Chương trình con (procedure) là một nhóm các chỉ thị thực hiện một nhiệm vụ nào đó và có thể được gọi ra thực hiện từ một vài nơi trong chương trình. Khi hoàn thành nhiệm vụ, nó phải trở về chỉ thị đứng sau chỉ thị gọi, chính vì vậy địa chỉ trở về cần phải được truyền cho chương trình con. Địa chỉ trở về có thể để ở một trong 3 nơi: trong bộ nhớ, trong một thanh ghi hoặc trong stack.

6. Các chỉ thị lặp

   Trong các chương trình thường xuyên xảy ra việc phải cho thi hành một nhóm chỉ thị một số lần cố định, vì vậy một số máy tính có các chỉ thị để có thể dễ dàng làm việc này. Các phương pháp khác nhau đều liên quan đến một con đếm, nó sẽ được tăng lên hoặc giảm đi một hằng số nào đó sau mỗi lần lặp. Nếu một điều kiện nhất định nào đó thoả mãn thì vòng lặp sẽ kết thúc.

7. Các chỉ thị vào/ra (I/O)

   Đây là nhóm chỉ thị mà giữa các máy khác nhau có nhiều điểm khác nhau nhất. Hiện nay có các phương pháp I/O được người ta sử dụng là:

   - I/O được định trình với việc luôn luôn đợi (Programmed I/O with busy waiting)
   - I/O được điều khiển bởi ngắt (Interrupt-driven I/O)
   - I/O sử dụng phương pháp truy cập trực tiếp bộ nhớ (DMA I/O)
   - I/O sử dụng kênh dữ liệu (I/O using data channel)

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

5.4 TÓM TẮT MỨC MÁY THÔNG THƯỜNG

Ngôn ngữ của mức máy thông thường thường được mọi người xem là ngôn ngữ máy - machine language. Tại mức này máy có bộ nhớ hướng byte hay word, nghĩa là kích thước ô nhớ bằng 1 byte hoặc bội số của byte, dung lượng bộ nhớ nằm trong khoảng từ hàng chục KB tới hàng MB và có các chỉ thị như là MOVE, ADD...

Hầu hết các máy tính điện tử hiện đại có bộ nhớ được tổ chức như một dãy các byte, trong đó cứ 2 hay 4 byte được nhóm lại với nhau thành một word. Thông thường máy tính có từ 8 đến 32 thanh ghi, mỗi thanh ghi chứa được một word. Có nhiều máy tính tạo thành cả một họ, thí dụ họ Intel, họ Motorola.

Các chỉ thị thường có một hoặc hai toán hạng, chúng được đánh địa chỉ bằng các phương thức (mode) đánh địa chỉ tức thời, trực tiếp, gián tiếp, chỉ số và một vài mode địa chỉ khác nữa. Một số máy có nhiều mode đánh địa chỉ phức tạp.

Chỉ thị thường có các loại: vận chuyển dữ liệu, thực hiện phép toán hai ngôi, phép toán một ngôi, các phép toán số học, logic, các chỉ thị nhảy, gọi thủ tục, lặp và cũng có thể gồm cả các chỉ thị I/O. Những máy tính có tập gồm trên 100 chỉ thị là điều hoàn toàn bình thường.

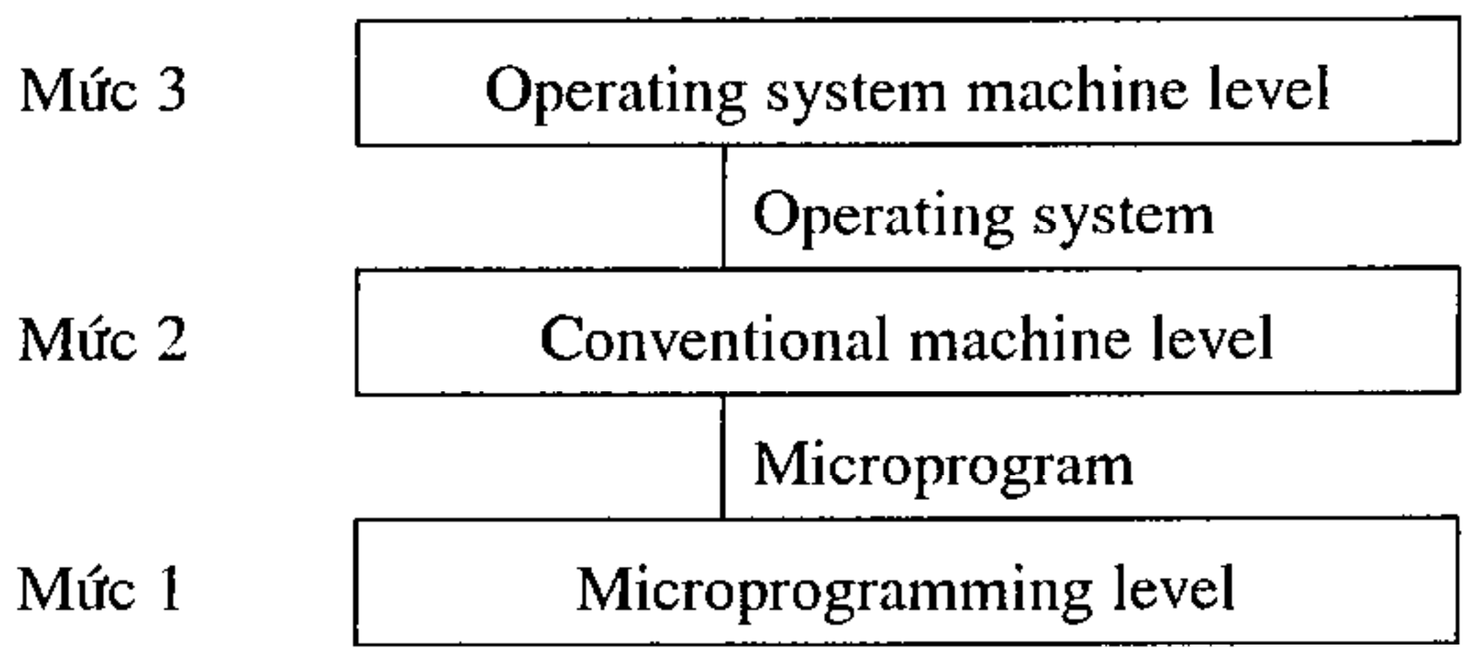
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 6 MỨC MÁY HỆ ĐIỀU HÀNH

6.1 GIỚI THIỆU MỨC MÁY HỆ ĐIỀU HÀNH

Trong máy tính có vi chương trình, các chỉ thị ở mức máy thông thường, chẳng hạn các chỉ thị gọi chương trình con, nhân và lặp không được thi hành một cách trực tiếp bằng phần cứng. Trái lại, chúng được lấy về, kiểm tra và thi hành như một dãy liên tiếp các bước nhỏ bằng vi chươngg trình. Máy ở mức 2 có thể được lập trình bởi những người lập trình không biết gì về máy mức 1 và trình thông dịch của nó, trong phạm vi mà họ quan tâm, máy mức 2 được sử dụng như một cái máy bằng phần cứng thực sự.

Giống như trình thông dịch chạy trên máy mức 1 có thể thông dịch các chương trình được viết bằng ngôn ngữ máy mức 2, một trình thông dịch chạy trên máy mức 2 có thể thông dịch các chương trình được viết bằng ngôn ngữ máy mức 3. Vì lý do lịch sử trình thông dịch chạy trên máy mức 2 để hỗ trợ cho máy mức 3 được người ta gọi là **hệ điều hành - operating system**, như minh hoạ trên hình 6-01. Chính vì vậy chúng ta sẽ gọi mức 3 là **mức máy hệ điều hành**.



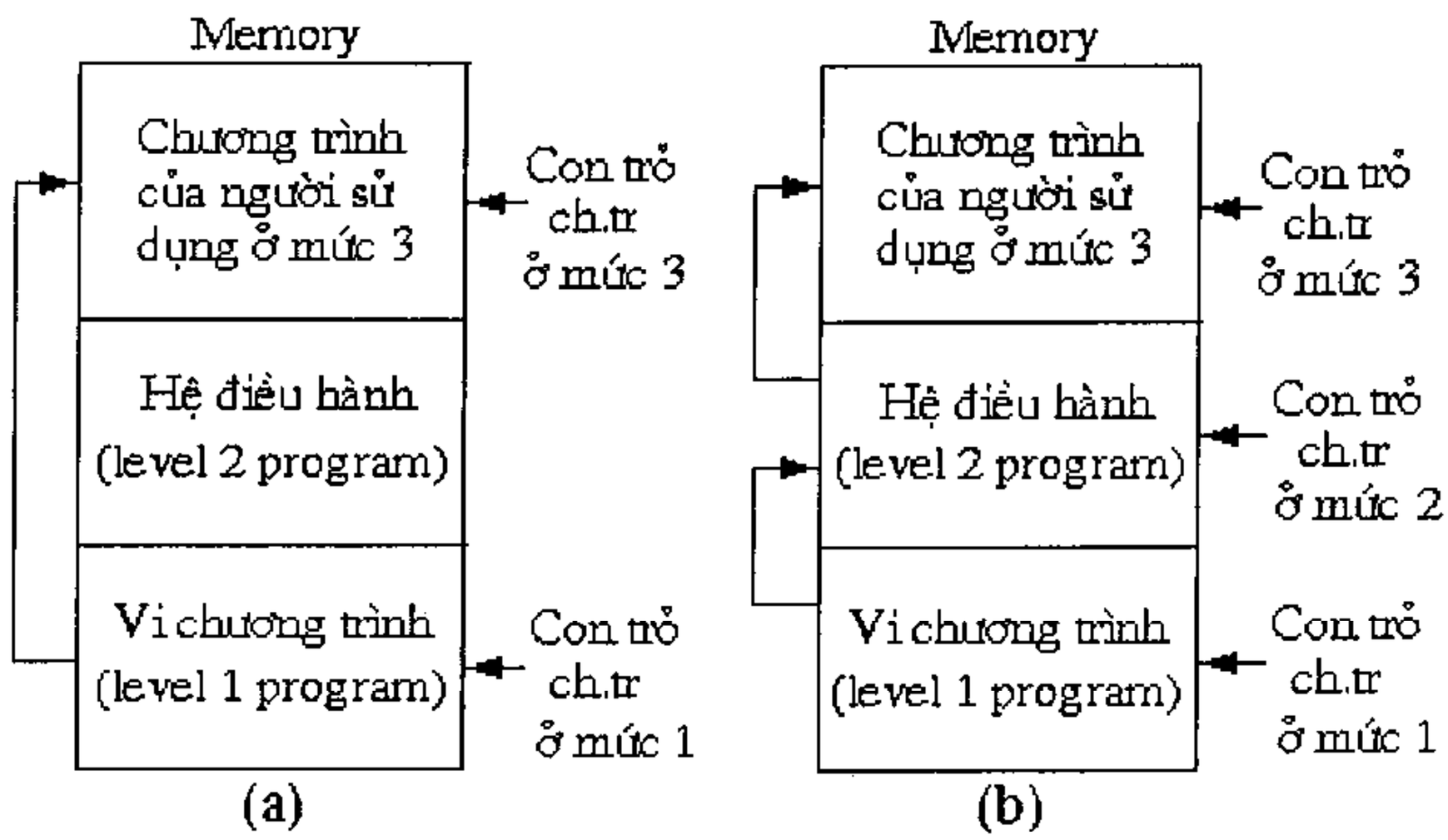
Hình 6-01 Cả mức 2 và mức 3 đều được hỗ trợ bởi phần mềm

Có một sự khác nhau quan trọng giữa cách mà mức máy hệ điều hành được hỗ trợ và cách mà mức máy thông thường được hỗ trợ. Sự khác nhau này là do mức máy hệ điều hành đã tiến hoá dần lên từ mức máy thông thường. Hầu hết các chỉ thị của mức máy hệ điều hành cũng có ở mức máy thông thường. Chúng ta sẽ gọi các chỉ thị này là các chỉ thị mức 3 thông thường, bởi vì chúng bao gồm cả các thao tác thông thường như thao tác số học, logic, dịch .v.v. Còn các chỉ thị mức 3 khác, không có ở mức 2, chúng ta sẽ gọi là các chỉ thị OSML (Operating System Machine Level - Mức máy hệ điều hành), để nhấn mạnh rằng chúng chỉ tồn tại ở mức máy hệ điều hành.

Mặc dầu có thể bắt hệ điều hành thông dịch tất cả các chỉ thị của mức 3, nhưng làm như vậy vừa không hiệu quả vừa không cần thiết. Các chỉ thị mức 3 thông thường có thể được thông dịch trực tiếp bằng vi chương trình. Tình huống được

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

minh hoạ trên hình 6-02a là trường hợp một máy tính có một bộ nhớ duy nhất dùng để chứa mọi chương trình. Khi chỉ có các chỉ thị thông thường được thực hiện, vi chương trình lấy các chỉ thị trực tiếp từ chương trình của người sử dụng, kiểm tra chúng và thi hành các chỉ thị này.



Hình 6-02 a) Các chỉ thị thông thường được thông dịch trực tiếp bởi vi chương trình; (b) Các chỉ thị OSML được thông dịch bởi hệ điều hành, còn hệ điều hành được thông dịch bằng vi chương trình

Tuy nhiên ngay khi gặp một chỉ thị OSML thì tình hình sẽ thay đổi. Vi chương trình dừng việc thông dịch chương trình của người sử dụng và thay vào đó là việc bắt đầu thông dịch hệ điều hành. Từ lúc này hệ điều hành kiểm tra chỉ thị OSML trong chương trình của người sử dụng và thi hành chỉ thị này. Khi chỉ thị OSML đã được thi hành xong, hệ điều hành sẽ thi hành một chỉ thị nhất định, chỉ thị này làm cho vi chương trình tiếp tục lấy và thi hành các chỉ thị của chương trình của người sử dụng. Tất nhiên nếu chỉ thị tiếp theo trong chương trình của người sử dụng cũng lại là một chỉ thị OSML thì hệ điều hành lại được khởi động.

Phương pháp để thực hiện chương trình ở mức 3 như vậy có nghĩa là trong một phần thời gian, máy tính có chức năng như một máy có ba mức, còn trong một phần thời gian khác nó giống như một máy có hai mức. Trong lúc thi hành chỉ thị OSML có 3 chương trình đang chạy, mỗi chương trình chạy trên một máy ảo, mỗi chương trình có trạng thái của riêng nó, kể cả con đếm chương trình riêng. Về mặt khái niệm, con đếm chương trình ở mức 3 trỏ tới chỉ thị OSML (hoặc chỉ thị đứng sau nó, tuỳ thuộc vào việc khi nào thì con đếm chương trình được tăng lên). Con đếm chương trình ở mức 2 trỏ tới chỗ mà chỉ thị hệ điều hành hiện đang thi hành. Con đếm chương trình ở mức 1 cho phần cứng thực sự biết vi chỉ thị nào sẽ được thực hiện.

Để cho đơn giản trong chương này hệ điều hành sẽ được coi như là một mức đơn, nhưng điều đó không có nghĩa là mọi hệ điều hành đều được tổ chức chỉ có

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

một mức. Trái lại một số hệ điều hành tiên tiến được xây dựng như là một dãy gồm vài mức con. Tuy nhiên chủ đề thiết kế hệ điều hành không phải là mục đích của giáo trình này.

Hầu hết các hệ điều hành cho các máy tính lớn là các ***hệ thống đa chương trình*** (multiprogramming systems), điều đó có nghĩa là hệ điều hành hành không phải chỉ hỗ trợ cho một máy ảo mức 3 mà nó hỗ trợ một vài máy ảo mức 3 một cách song song. Nếu mỗi một trong số các máy ảo này được nối với một terminal ở xa, thì hệ thống này được gọi là ***hệ thống chia sẻ thời gian*** (*time-sharing system*). Nếu không có một terminal ở xa nào thì hệ thống được gọi là ***hệ thống đa chương trình theo lô*** (*batch multiprogramming system*). Các dạng lai, trong đó một số máy ảo được sử dụng trực tuyến (on line) còn các máy ảo khác thì không trực tuyến là phổ biến. Nhiệm vụ quan trọng hàng đầu của hệ điều hành là giải quyết việc quản lý mọi máy ảo chứ không phải là vấn đề thông dịch các chỉ thị OSML.

Khi nghiên cứu hệ điều hành, cần tập trung vào 3 chủ đề chính, quan trọng nhất sau đây.

Chủ đề thứ nhất là bộ nhớ ảo (virtual memory), đó là kỹ thuật được nhiều hệ điều hành sử dụng, làm cho máy thể hiện như có một bộ nhớ lớn hơn bộ nhớ mà nó thực sự có.

Chủ đề thứ hai là vào/ra file (file I/O), đây là một khái niệm có mức độ trừu tượng cao hơn khái niệm về các chỉ thị vào/ra (I/O instruction).

Chủ đề thứ ba là xử lý song song - làm thế nào để nhiều quá trình có thể được đồng thời thi hành ở mức 3. Tại thời điểm này chúng ta có thể coi quá trình là một chương trình đang chạy cùng với tất cả các thông tin trạng thái của nó như bộ nhớ, các thanh ghi, con đếm chương trình, trạng thái I/O.v.v.

Chương này chỉ giới thiệu rất tóm tắt về hệ điều hành, hạn chế trong 2 chủ đề trên. Chủ đề thứ ba thường được nghiên cứu kỹ trong các giáo trình về hệ điều hành.

6.2 BỘ NHỚ ẢO

Trong những ngày đầu tiên của kỷ nguyên máy tính, bộ nhớ của máy tính thường nhỏ và đắt tiền. Máy IBM 650 là một máy tính cho nghiên cứu khoa học dẫn đầu lúc đó (cuối những năm 1950) chỉ có bộ nhớ 2000 word. Một trong các compiler đầu tiên cho ALGOL 60 đã được viết cho các máy tính có bộ nhớ chỉ có 1024 word. Thời kỳ này các nhà lập trình thường phải dành phần lớn thời gian nhằm làm sao cho chương trình càng bé càng tốt để có thể lọt vào được một bộ nhớ rất nhỏ.

Một giải pháp truyền thống cho vấn đề này là sử dụng bộ nhớ phụ, chẳng hạn đĩa. Người lập trình chia chương trình ra làm một số phần, gọi là overlay, mỗi overlay có thể nằm lọt trong bộ nhớ. Để chạy chương trình, overlay thứ nhất được đọc vào và nó sẽ chạy được một lúc. Trước khi kết thúc, nó sẽ đọc overlay tiếp theo vào và gọi overlay đó ra làm việc và việc đó cứ tiếp diễn như thế. Người lập trình

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

phải lo chia chương trình thành các overlay, phải quy định từng overlay sẽ được giữ ở đâu trong bộ nhớ phụ, phải thu xếp để vận chuyển các overlay giữa bộ nhớ chính và bộ nhớ phụ và nói chung là phải tự mình quản lý toàn bộ quá trình overlay.

Kỹ thuật này đã được sử dụng rộng rãi trong nhiều năm, nhưng nó đòi hỏi nhiều công sức của người lập trình cho việc quản lý overlay. Năm 1961 một nhóm nhà khoa học ở Manchester, nước Anh, đã đề xuất một phương pháp ***thực hiện quá trình overlay một cách tự động***, thậm chí không đòi hỏi người lập trình phải biết điều gì đang xảy ra. Phương pháp này được gọi là ***bộ nhớ ảo - virtual memory***. Phương pháp này bắt đầu được sử dụng trong một số máy tính vào những năm 1960, hầu hết liên quan tới các dự án nghiên cứu về thiết kế hệ thống máy tính. Đầu những năm 1970 bộ nhớ ảo đã trở nên thông dụng trong các hầu hết các máy tính. Ngày nay thậm chí cả các bộ vi xử lý, trong đó có 80386 của Intel và 68030 Motorola cũng có các hệ thống bộ nhớ ảo rất tinh vi.

6.2.1 Việc phân trang - Paging

Ý tưởng mà nhóm các nhà khoa học ở Manchester đưa ra đã làm tách biệt các khái niệm về không gian địa chỉ và các vị trí nhớ. Chúng ta hãy xem xét một thí dụ về một máy tính trong đó các chỉ thị có trường địa chỉ 16 bit và có bộ nhớ 4096 word. Một chương trình chạy trên máy tính này có thể đánh địa chỉ 65536 word bộ nhớ. Số word có thể đánh địa chỉ chỉ phụ thuộc vào số bit trong trường địa chỉ của chỉ thị và chẳng có gì liên quan tới số word mà bộ nhớ thực sự có. Không gian địa chỉ của máy tính này bao gồm các số 0, 1, 2, , 65535.

Khi còn chưa phát minh ra bộ nhớ ảo, người ta phải phân biệt giữa các địa chỉ nhỏ hơn 4096 và các địa chỉ lớn hơn hoặc bằng 4096. Có thể coi miền thứ nhất là không gian địa chỉ khả dụng và miền thứ hai là không gian địa chỉ không khả dụng. Người ta thường không phân biệt rõ ràng giữa không gian địa chỉ và các địa chỉ bộ nhớ thực sự, bởi vì phần cứng bắt phải có sự tương ứng một-một giữa chúng. Chương trình không được phép sử dụng các địa chỉ không khả dụng.

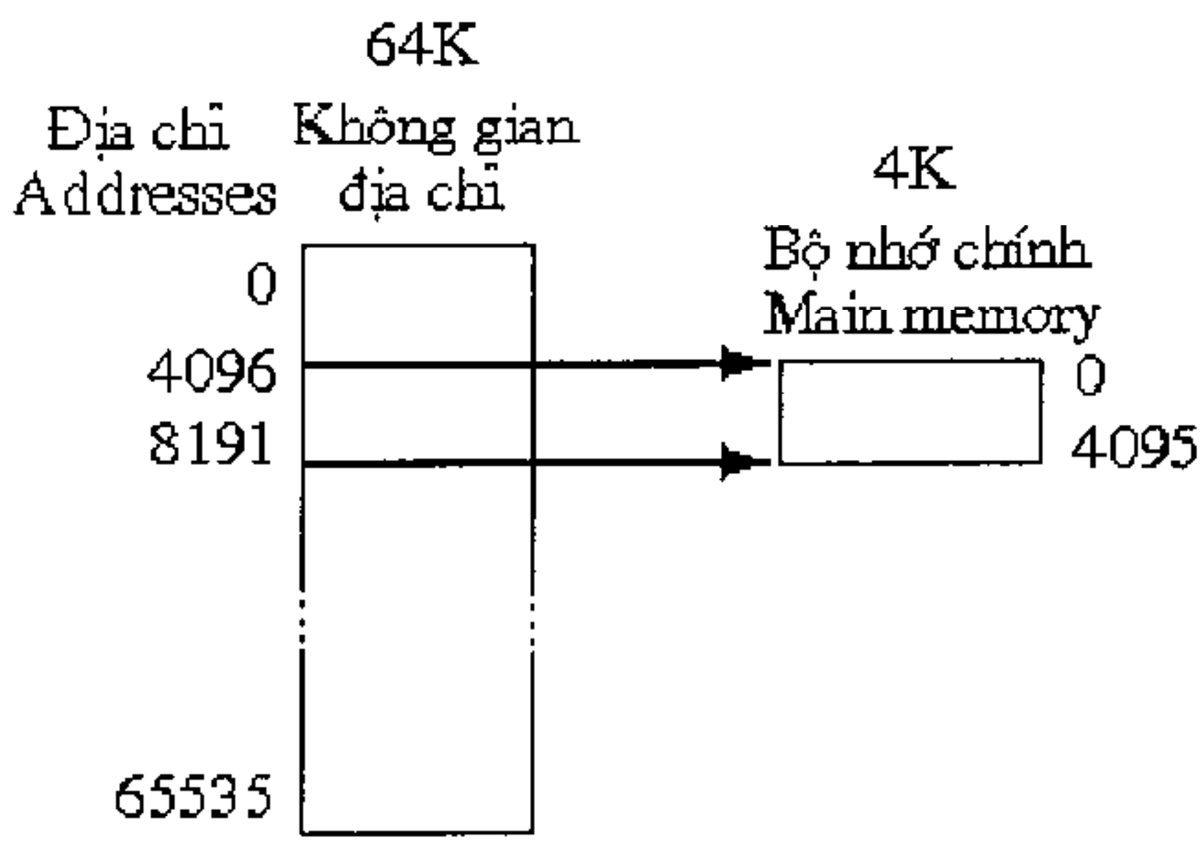
Ý tưởng về việc làm tách biệt không gian địa chỉ và các địa chỉ bộ nhớ thực (bộ nhớ vật lý) như sau. Vào bất cứ thời điểm nào 4096 word bộ nhớ có thể được truy cập trực tiếp, nhưng chúng không nhất thiết phải tương ứng với các địa chỉ từ 0..4095. Thí dụ chúng ta có thể "bảo" cho máy tính rằng từ lúc này trở đi, mỗi khi địa chỉ 4096 được truy cập thì word 0 của bộ nhớ sẽ được sử dụng. Mỗi khi địa chỉ 4097 được truy cập thì word 1 của bộ nhớ sẽ được sử dụng, mỗi khi địa chỉ 8191 được truy cập thì word 4095 của bộ nhớ sẽ được sử dụng.v.v. Nói cách khác, chúng ta đã định nghĩa một sự ánh xạ từ không gian địa chỉ vào các địa chỉ bộ nhớ thực, như được minh hoạ trên hình 6-03.

Bằng cách ánh xạ các địa chỉ của không gian địa chỉ vào các vị trí nhớ thực như trên hình vẽ 6-03, một máy với 4K bộ nhớ nếu không có bộ nhớ ảo thì chỉ có một sự xạ cố định các địa chỉ từ 0..4095 vào 4096 word bộ nhớ. Có một câu hỏi thú vị là: điều gì sẽ xảy ra nếu chương trình nhảy tới một địa chỉ nằm trong khoảng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

8192..12287? Trong máy tính không có bộ nhớ ảo, chương trình này sẽ gây ra một bẫy lỗi, nó sẽ in ra màn hình một thông báo, chẳng hạn "Nonexistent memory referenced" rồi dừng chương trình. Trong máy tính có bộ nhớ ảo, một dãy các bước sau có thể sẽ lần lượt diễn ra:

1. Nội dung của bộ nhớ chính được cất vào bộ nhớ phụ.
2. Các word 8192..12287 đang nằm trong bộ nhớ phụ được nạp vào bộ nhớ chính.
3. Ánh xạ địa chỉ sẽ được thay đổi để ánh xạ các địa chỉ 8192..12287 vào các vị trí nhớ 0..4095.
4. Chương trình tiếp tục như không có điều gì bất thường xảy ra.



Hình 6-03 Việc ánh xạ các địa chỉ 4096..8191
vào các địa chỉ bộ nhớ chính 0..4095

Kỹ thuật thực hiện overlay tự động được gọi là ***phân trang - paging***, các đoạn chương trình được đọc vào bộ nhớ chính từ bộ nhớ phụ được gọi là các **trang**. Người ta cũng đã nghiên cứu các cách tinh vi hơn để ánh xạ các địa chỉ từ không gian địa chỉ vào các địa chỉ bộ nhớ thực.

Để nhấn mạnh chúng ta sẽ gọi miền địa chỉ mà chương trình có thể truy cập là ***không gian địa chỉ ảo*** (virtual address space) còn các địa chỉ bộ nhớ thực, bằng các mạch điện, được gọi là ***không gian địa chỉ vật lý*** (physical address space). Ánh xạ bộ nhớ liên hệ các địa chỉ ảo với các địa chỉ vật lý.

Trên thực tế người lập trình có thể viết chương trình mà không cần phải biết đến sự tồn tại của bộ nhớ ảo. Đối với anh ta máy tính dường như có một bộ nhớ chính rất lớn. Việc phân trang làm cho người lập trình có ảo giác về bộ nhớ chính tuyến tính, liên tục và lớn, có kích thước bằng kích thước của không gian địa chỉ; trong khi đó trên thực tế bộ nhớ chính có thể nhỏ hơn không gian địa chỉ nhớ. Sự mô phỏng bộ nhớ chính lớn như thế không thể phát hiện ra được bằng chương trình (trừ việc cho chạy các chương trình kiểm tra thời gian), vì vậy cơ chế phân trang có thể coi là trong suốt đối với người lập trình.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Đặc điểm trên của phương pháp phân trang bộ nhớ rất quan trọng, làm cho nó khác với phương pháp phân đoạn bộ nhớ (segmentation), trong đó người lập trình phải ý thức về sự tồn tại của các phân đoạn. Tuy nhiên giáo trình này sẽ không trình bày vấn đề phân đoạn bộ nhớ.

Ý tưởng về việc người lập trình có thể sử dụng một số đặc điểm không tồn tại thực sự mà không cần phải quan tâm đến việc nó hoạt động như thế nào không phải là điều mới mẻ đối với chúng ta. Thí dụ tập chỉ thị của một máy tính mức 2 là không có thực theo nghĩa là các chỉ thị này không phải là các thao tác sơ cấp của phần cứng, trên thực tế chúng được thi hành bằng phần mềm ở mức 1. Tương tự như vậy một người lập trình ở mức 3 có thể sử dụng bộ nhớ ảo mà chẳng cần bận tâm gì về việc nó hoạt động như thế nào, chỉ có những người viết hệ điều hành là phải biết nó làm việc như thế nào.

6.2.2 Thực hiện việc phân trang

Một đòi hỏi cốt yếu của bộ nhớ ảo là phải có bộ nhớ phụ để chứa toàn bộ chương trình. Nếu ta coi bản copy của chương trình trong bộ nhớ phụ là bản gốc, còn các phần của chương trình thỉnh thoảng được đưa vào bộ nhớ chính là các bản copy thì một số khái niệm mà chúng ta sử dụng sau này sẽ đơn giản hơn. Một công việc tự nhiên và quan trọng là cập nhật cho bản gốc, những thay đổi xảy ra với bản copy trong bộ nhớ chính, cuối cùng cần phải được phản ánh vào trong bản gốc.

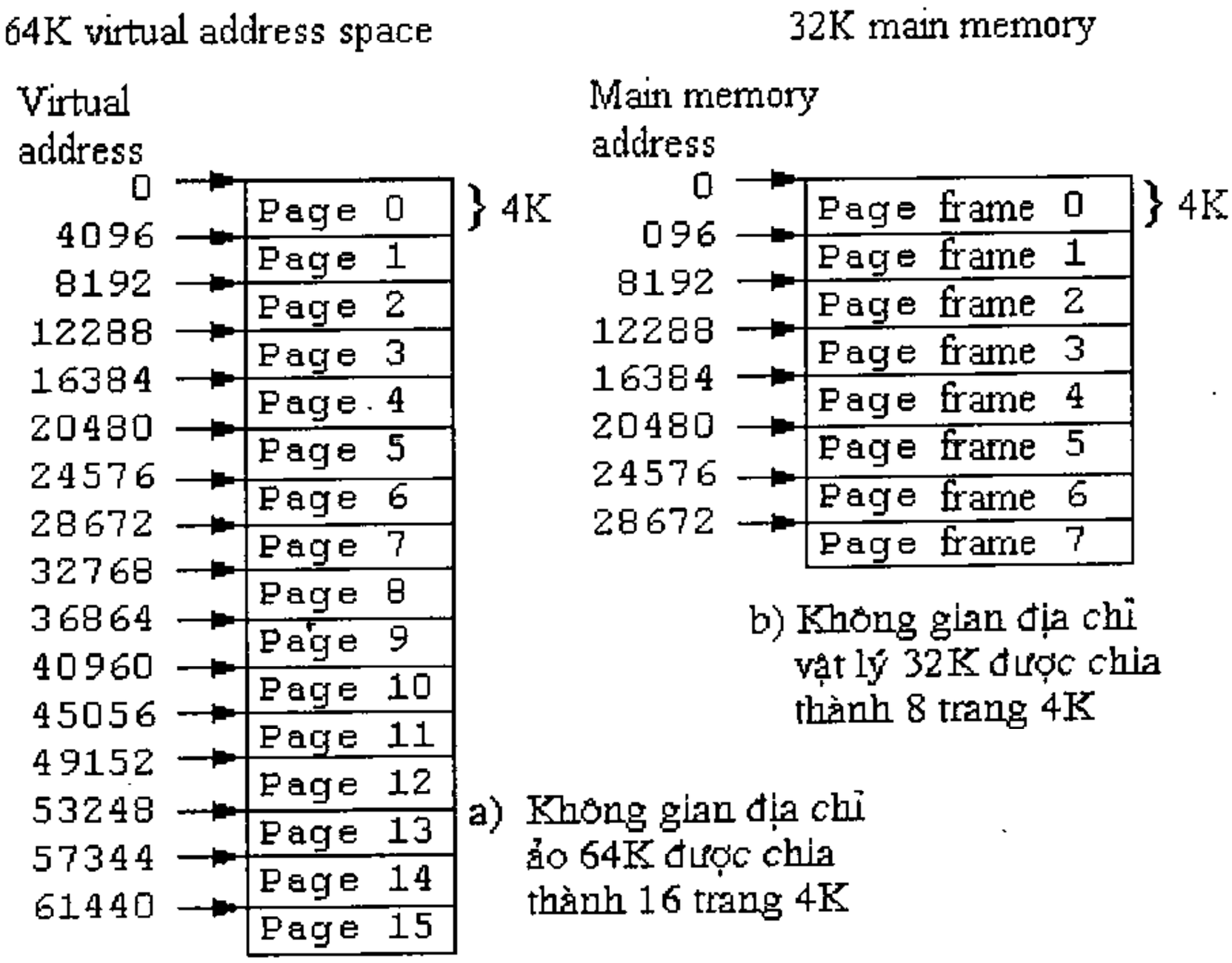
Không gian địa chỉ ảo được chia thành các trang có kích thước bằng nhau. Hiện nay người ta thường sử dụng kích thước trang nằm trong khoảng 512..4096. Nó luôn được chọn là số mũ của 2. Tương tự như vậy, không gian địa chỉ vật lý cũng được chia thành các mảnh, mỗi mảnh có cùng kích thước với trang, để cho mỗi mảnh của bộ nhớ chính có thể chứa được vừa vặn một trang. Các mảnh của bộ nhớ chính mà các trang sẽ được đưa vào được gọi là ***khung trang - page frame***. Trên hình 6-03 bộ nhớ chính chỉ có 1 khung trang. Trong các thiết kế thực, bộ nhớ chính của các máy tính lớn có thể có hàng chục, hàng trăm hay thậm chí hàng nghìn khung trang.

Hình 6-04 minh hoạ một cách chia không gian địa chỉ 64K. Bộ nhớ ảo trên hình này có thể được triển khai thực hiện ở mức 2 bằng cách sử dụng một ***bảng phân trang - page table*** có 16 word (bảng phân trang sẽ được trình bày ngay dưới đây). Khi một chương trình truy cập bộ nhớ, dù là để lấy dữ liệu, chứa dữ liệu, lấy chỉ thị hay là nhảy, đầu tiên nó sẽ sinh ra một địa chỉ 16 bit tương ứng với một địa chỉ ảo nằm trong miền 0..65535.

Trong thí dụ này, địa chỉ 16 bit được sử dụng với 4 bit cao làm số trang ảo và 12 bit thấp làm địa chỉ trong trang ảo đã được chọn, như thể hiện trên hình 6-05a. Trong hình này địa chỉ 16 bit có giá trị là 12310 (số nhị phân trên hình vẽ là 0011 0000 0001 0110), theo cách sử dụng trên nó sẽ được xem là địa chỉ 22 của trang 3. Mối liên hệ giữa trang và địa chỉ ảo trong thí dụ này được thể hiện trên hình 6-05b. Nếu địa chỉ ảo 0 của trang 3 là tại địa chỉ vật lý 12288, thì địa chỉ ảo 22 phải là tại địa chỉ vật lý 12310.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Sau khi đã tính ra được rằng cần trang ảo số 3, hệ điều hành phải tìm ra được trang ảo số 3 nằm ở đâu. Có 9 khả năng xảy ra: trong đó 8 khả năng là trang ảo số 3 nằm ở một trong số 8 khung trang trong bộ nhớ chính; khả năng thứ chín là trang ảo không có trong bộ nhớ chính mà nằm ở đâu đó trong bộ nhớ phụ.



Hình 6-04 Một cách chia không gian địa chỉ

- **Bảng phân trang**

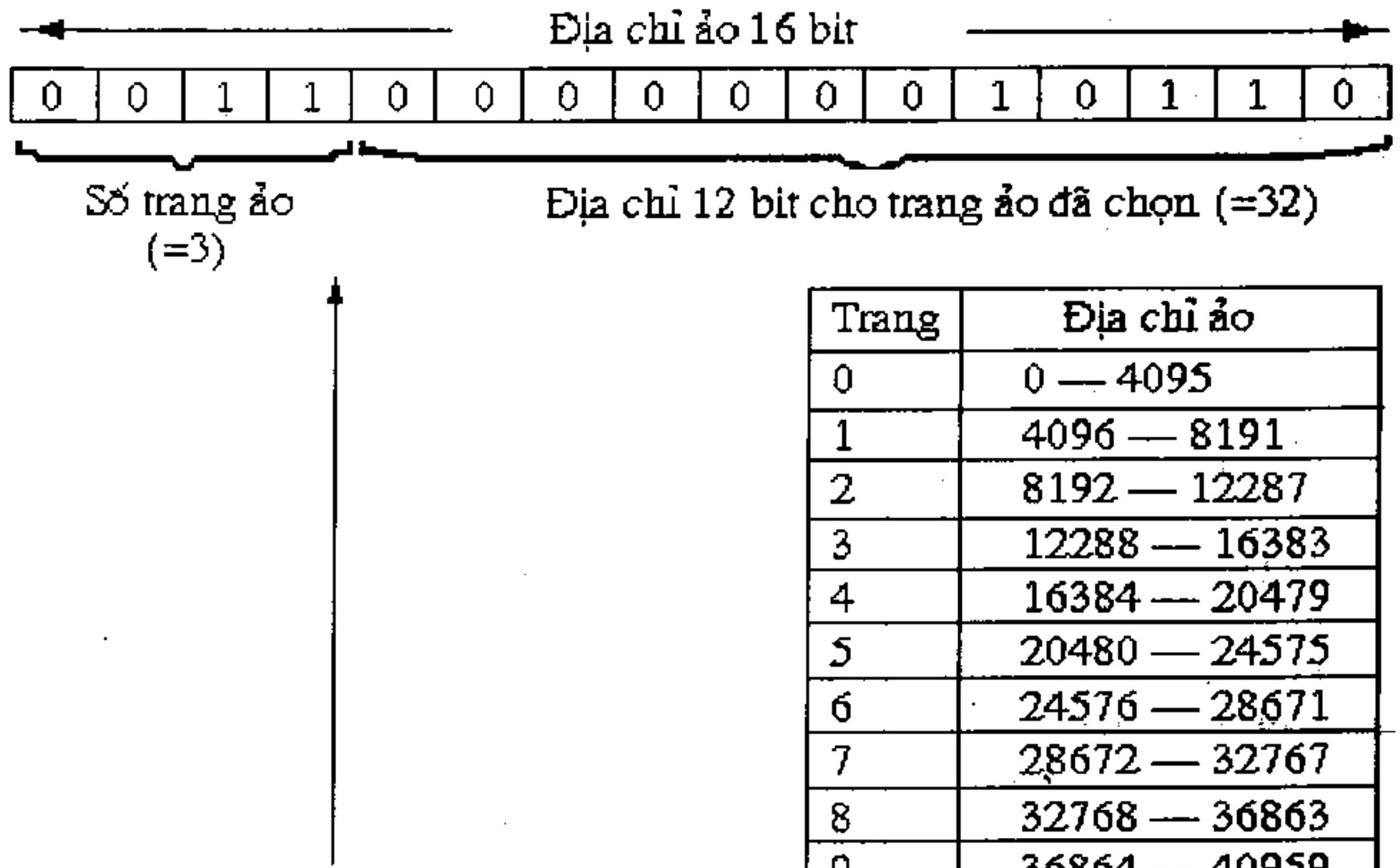
Để tìm ra được khả năng nào trong 9 khả năng nói trên là đúng, hệ điều hành sẽ nhìn lên bảng phân trang. Bảng phân trang có số đề mục (entry) bằng số lượng trang ảo, mỗi đề mục giống như một bản ghi (record) có 3 trường, xem hình 6-06.

Trường đầu tiên kích thước 1 bit, nó sẽ bằng 0 nếu trang ảo tương ứng không nằm trong bộ nhớ chính và sẽ bằng 1 nếu ngược lại.

Trường thứ hai chứa địa chỉ nơi chứa trang ảo trong bộ nhớ phụ khi nó không nằm trong bộ nhớ chính (thí dụ số hiệu track hay sector trên đĩa). Phải có địa chỉ này để khi cần có thể tìm được trang trong bộ nhớ phụ và đọc nó vào bộ nhớ chính và sau này ghi nó trở lại vị trí ban đầu trong bộ nhớ phụ khi nó không còn cần phải nằm trong bộ nhớ chính nữa.

Trường thứ ba kích thước 3 bit chứa một con số chỉ khung trang, là nơi mà trang nằm nếu nó ở trong bộ nhớ chính. Nếu trang không nằm trong bộ nhớ chính thì trường này không có ý nghĩa gì và được lờ đi.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

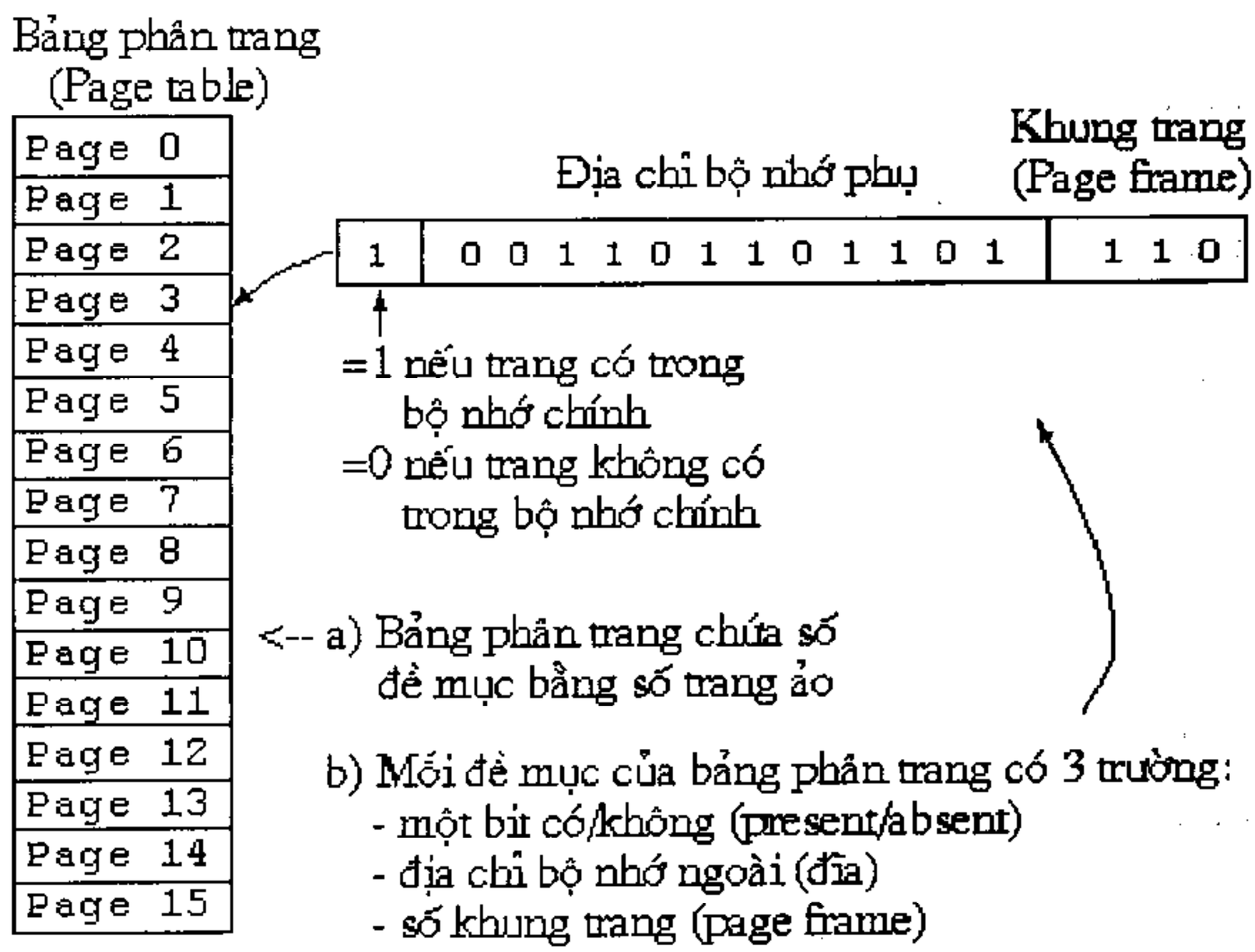


a) Một địa chỉ ảo bao gồm số trang ảo 4-bit và địa chỉ tương đối (offset) 12 bit

b) Số trang và địa chỉ ảo của chúng --->

Trang	Địa chỉ ảo
0	0 — 4095
1	4096 — 8191
2	8192 — 12287
3	12288 — 16383
4	16384 — 20479
5	20480 — 24575
6	24576 — 28671
7	28672 — 32767
8	32768 — 36863
9	36864 — 40959
10	40960 — 45055
11	45056 — 49151
12	49152 — 53247
13	53248 — 57343
14	57344 — 61439
15	61440 — 65535

Hình 6-05 Thí dụ một địa chỉ ảo

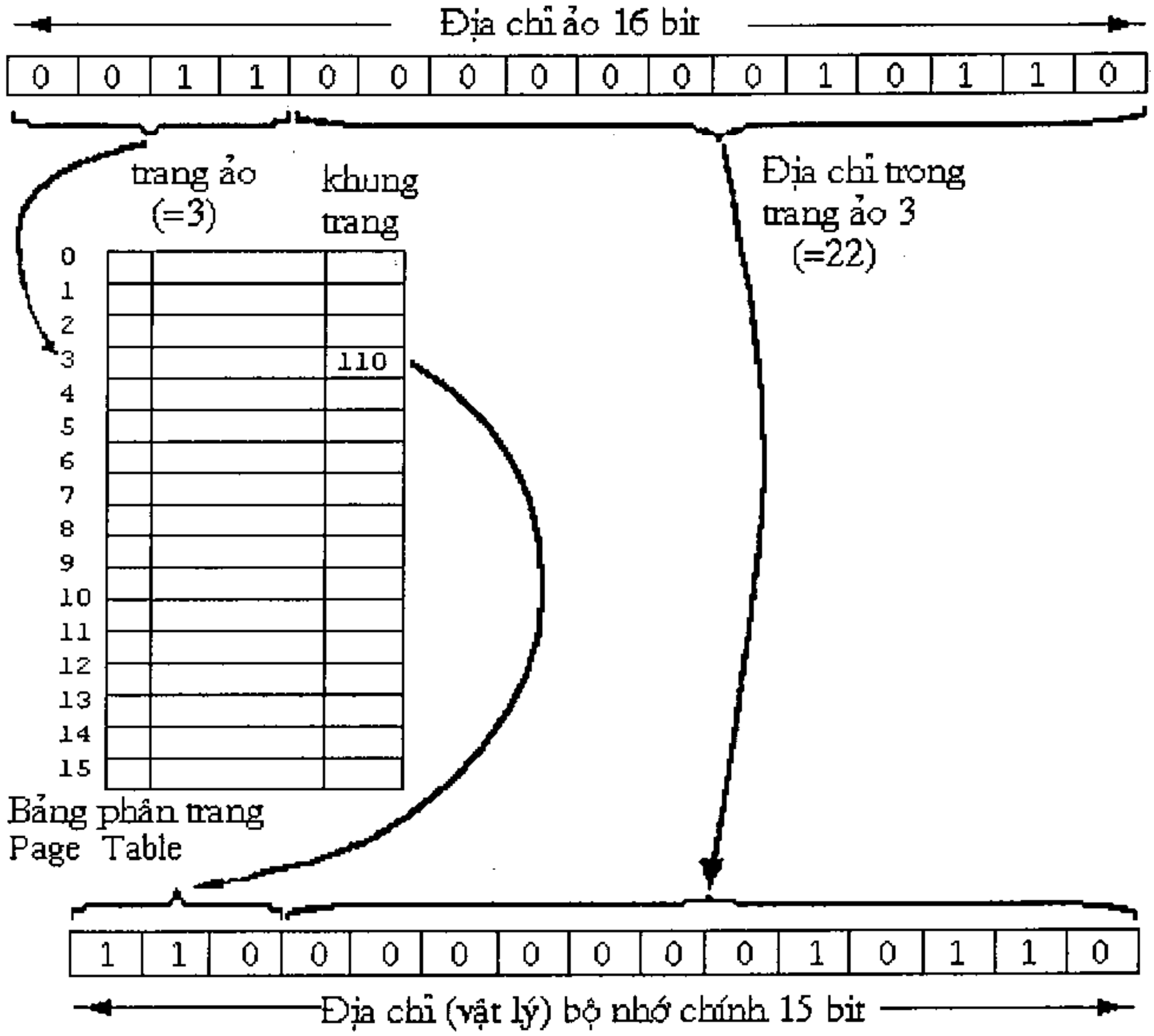


Hình 6-06 Thí dụ một Bảng phân trang

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Cách tạo địa chỉ bộ nhớ chính (bộ nhớ vật lý) từ địa chỉ ảo**

Giả sử rằng trang ảo nằm trong bộ nhớ chính, trường khung trang 3 bit sẽ chỉ ra trang nằm ở đâu. Con số 3 bit này sẽ được nạp vào 3 bit bên trái nhất của thanh ghi địa chỉ bộ nhớ MAR, còn 12 bit bên phải là địa chỉ trong trang ảo sẽ được nạp vào 12 bit bên phải của MAR. Đó là cách tạo thành địa chỉ bộ nhớ chính, được minh hoạ trên hình 6-07.



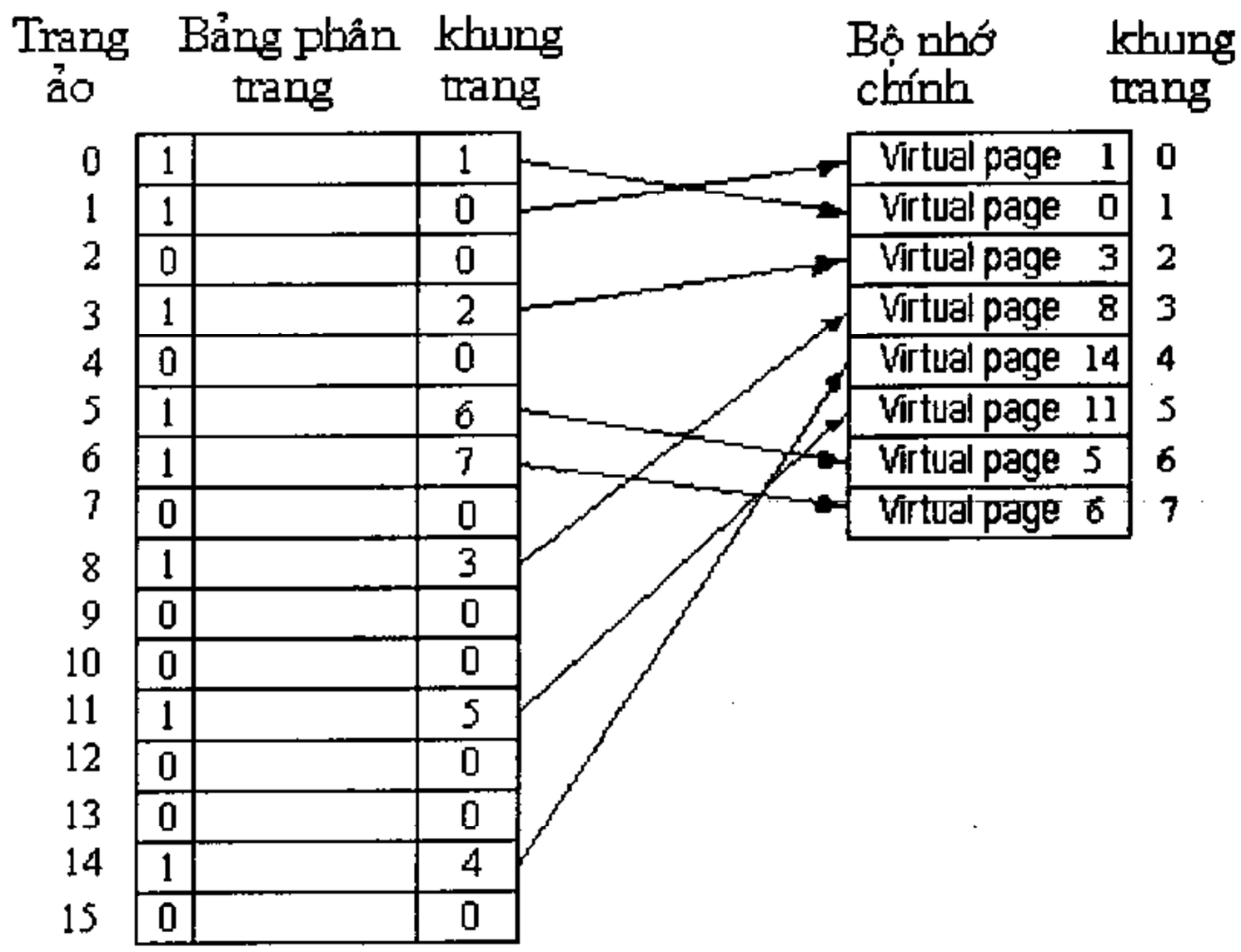
Hình 6-07 Cách tạo ra địa chỉ bộ nhớ chính từ địa chỉ ảo

Ba bit số chỉ khung trang cộng với 12 bit offset tạo thành một địa chỉ 15 bit, đó chính là địa chỉ cần cho bộ nhớ chính dung lượng 32K như trên hình 6-04 mà chúng ta đã xem xét. Bây giờ phần cứng có thể sử dụng địa chỉ này và lấy về word mà nó cần để nạp vào thanh ghi đệm bộ nhớ MBR (nếu thao tác với bộ nhớ là đọc - Read) hoặc có thể chứa nội dung của thanh ghi MBR vào một word cần thiết nào đó (nếu thao tác với bộ nhớ là ghi - Write).

Hình 6-08 cho thấy một ánh xạ có thể giữa các trang ảo và các khung trang vật lý. Trang ảo số 0 nằm ở khung trang số 1. Trang ảo số 1 nằm ở khung trang số 0. Trang ảo số 2 không nằm trong bộ nhớ chính. Trang ảo số 3 nằm ở khung trang số 2. Trang ảo số 4 không nằm trong bộ nhớ chính. Trang ảo số 5 nằm ở khung trang số 6 .v.v.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Nếu hệ điều hành phải chuyển đổi mỗi địa chỉ ảo của chỉ thị máy mức 3 thành địa chỉ thực thì một máy mức 3 có bộ nhớ ảo có thể chạy chậm hơn một máy mức 3 không có bộ nhớ ảo rất nhiều lần và như vậy toàn bộ ý tưởng sẽ là không có ý nghĩa thực tế.



Hình 6-08 Một thí dụ về ánh xạ từ không gian địa chỉ ảo lên không gian bộ nhớ chính có 8 khung trang

Để tăng tốc độ việc chuyển đổi địa chỉ ảo thành địa chỉ vật lý, bảng phân trang thường được giữ trong các thanh ghi phần cứng đặc biệt, việc chuyển đổi được thực hiện trực tiếp bằng phần cứng. Cách này đòi hỏi các chi phí phần cứng.

Một cách khác để tăng tốc là giữ bảng phân trang trong các thanh ghi tốc độ cao và dùng vi chương trình thực hiện chuyển đổi bằng lập trình trực tiếp đối với các thanh ghi. Tuỳ thuộc vào kiến trúc của mức vi chương trình, chuyển đổi bằng cách này có thể gần nhanh bằng cách chuyển đổi trực tiếp bằng phần cứng mà không đòi hỏi phải có các mạch điện đặc biệt.

6.2.3 Phương pháp Cấp trang khi có yêu cầu và Mô hình tập làm việc

Trong thực tế bộ nhớ chính nói chung không đủ lớn để chứa tất cả các trang. Sự truy cập tới một địa chỉ thuộc một trang mà trang đó không có trong bộ nhớ chính được gọi là *lỗi trang - page fault*. Khi lỗi trang xảy ra, hệ điều hành cần phải đọc trang được yêu cầu từ bộ nhớ phụ vào bộ nhớ chính, nhập vị trí nhớ vật lý mới của nó vào bảng phân trang và sau đó lặp lại chỉ thị đã gây ra lỗi trang.

6.2.3.1 Phương pháp Cấp trang khi có yêu cầu

Với máy có bộ nhớ ảo, có thể khởi động một chương trình ngay cả khi không có phần nào của chương trình này nằm trong bộ nhớ chính, chỉ cần thiết lập cho bảng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

phân trang để chỉ ra là mọi trang ảo nằm trong bộ nhớ phụ chứ không phải trong bộ nhớ chính. Khi CPU lấy chỉ thị đầu tiên, lập tức nó gặp một lỗi trang, điều này làm cho trang có chứa chỉ thị đầu tiên được nạp vào và bảng phân trang được cập nhập thông tin. Sau đó chỉ thị đầu tiên có thể bắt đầu được thực hiện. Nếu chỉ thị đầu tiên có hai địa chỉ mà chúng lại nằm ở các trang khác với trang chứa chính chỉ thị, thì lại xảy ra hai lỗi trang nữa và sẽ lại có thêm hai trang được nạp vào bộ nhớ chính trước khi chỉ thị này được thi hành. Các chỉ thị tiếp theo của chương trình cũng có thể gây ra các lỗi trang nữa và quá trình trên lại tiếp diễn.v.v.

Phương pháp vận hành bộ nhớ ảo như thế được gọi là ***cấp trang khi có yêu cầu - demand paging***, nó giống với thuật toán nổi tiếng demand feeding (*cho ăn khi đòi*) đối với trẻ con: khi đứa trẻ khóc thì bạn cần cho nó ăn, chứ không cho nó ăn theo những giờ đã định trong ngày. Trong phương pháp cấp trang khi có yêu cầu, các trang chỉ được nạp vào bộ nhớ chính khi xuất hiện một yêu cầu thực sự về trang, chứ không phải là được nạp từ trước.

Việc xem xét có sử dụng phương pháp cấp trang khi có yêu cầu hay không chỉ liên quan tới lúc bắt đầu chạy một chương trình lần đầu tiên. Khi chương trình này đã chạy được một lúc rồi, thì các trang cần đến đã được tập hợp trong bộ nhớ chính.

Nếu máy tính hoạt động trong chế độ chia sẻ thời gian và người sử dụng được chuyển đổi kiểu quay vòng, thí dụ mỗi chương trình của người sử dụng được cho chạy 100ms rồi tạm dừng để cho chạy một chương trình khác.v.v. thì từng chương trình sẽ được khởi động lại nhiều lần trong quá trình chạy của nó.

Bởi vì mỗi chương trình có một bản đồ bộ nhớ duy nhất và bản đồ này sẽ bị thay đổi khi các chương trình được chuyển đổi, cho nên vấn đề thay đổi lặp đi lặp lại đó trở thành một vấn đề nghiêm trọng, làm giảm hiệu suất của hệ thống.

6.2.3.2 Phương pháp Mô hình tập làm việc

Một cách tiếp cận khác dựa trên sự quan sát thấy rằng hầu hết các chương trình không truy cập không gian địa chỉ của chúng đồng đều như nhau mà có khuynh hướng tập trung vào một số ít các trang. Sự truy cập bộ nhớ có thể là lấy một chỉ thị, chỉ thị này có thể lấy dữ liệu hoặc chứa dữ liệu. Tại một thời điểm t nào đó, tồn tại một tập bao gồm tất cả các trang được sử dụng bởi k lần truy cập bộ nhớ gần đó nhất. Người ta gọi tập này là tập làm việc - working set, ký hiệu w(k,t).

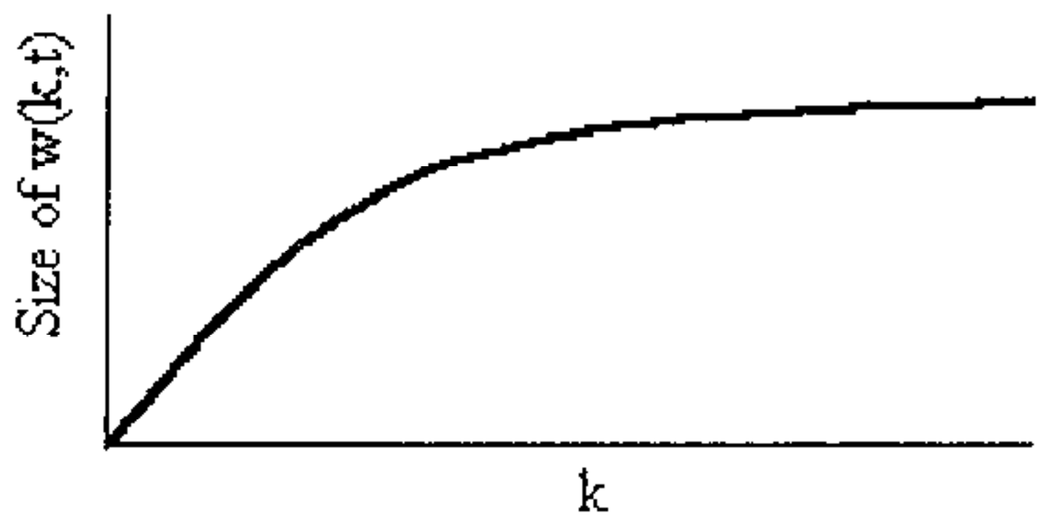
Bởi vì k+1 sự truy cập bộ nhớ mới đây nhất chắc chắn đã sử dụng tất cả các trang được k lần truy cập bộ nhớ mới đây nhất sử dụng và có thể cả các lần truy cập khác nữa, cho nên w(k,t) là một hàm đơn điệu tăng của k. Tuy nhiên giới hạn của w(k,t) khi k cứ tăng dần lên là hữu hạn, bởi vì một chương trình không thể tham khảo nhiều trang hơn số trang của không gian địa chỉ của nó. Có một số ít chương trình chỉ sử dụng có một trang.

Hình 6-09 miêu tả kích thước của tập làm việc là hàm của k. Trong thực tế hầu hết các chương trình truy cập ngẫu nhiên một số ít các trang nhưng tập này thay đổi

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

chậm theo thời gian, điều đó giải thích cho sự tăng nhanh ban đầu của đường cong và sau đó tăng chậm khi k lớn.

Thí dụ một chương trình đang thi hành một vòng lặp chiếm hai trang, chương trình sử dụng dữ liệu chiếm bốn trang, nó có thể tham khảo cả sáu trang cứ mỗi khi thi hành 1000 chỉ thị. Tuy nhiên lần tham khảo mới nhất tới một trang khác nào đó có thể là lần tham khảo một triệu tới trang đó tính từ lúc chương trình bắt đầu chạy. Do hành trạng có tính tiệm cận này nên nội dung của tập làm việc không nhạy cảm đối với giá trị đã được chọn của k, hay nói một cách khác, tồn tại một miền rộng các giá trị của k trong đó working set là không thay đổi.



Hình 6-09 Hàm w(k,t) là kích thước của tập làm việc tại thời điểm t

Lợi ích của Phương pháp Mô hình tập làm việc

Vì tập làm việc thay đổi chậm theo thời gian, cho nên có thể đoán ra được các trang nào sẽ cần đến khi một chương trình được cho chạy trở lại dựa trên tập làm việc của lần chạy trước đó. Các trang này có thể được nạp vào trong khi một chương trình khác nào đó vẫn còn đang chạy, trước khi chương trình được cho chạy trở lại. Nhờ đó khi chương trình được cho chạy trở lại, nó sẽ không sinh ra ngay lập tức một số lớn các lỗi trang làm lãng phí thời gian.

Người ta đã biết rằng thời gian cần để đọc một trang từ đĩa vào bộ nhớ chính thường bằng hoặc lớn hơn thời gian thực hiện 20.000 chỉ thị.

Nhược điểm của phương pháp sử dụng tập làm việc

Nếu tập làm việc được đưa vào bộ nhớ chính trước khi một chương trình được cho chạy trở lại, nhưng chương trình này khi chạy lại cần một tập làm việc khác (chương trình nằm ở lúc chuyển đổi giữa tập làm việc này và tập làm việc khác), thì việc nạp trước đó là thừa, và chương trình vẫn phải đợi cho đến khi các trang mà nó cần được nạp vào.

Theo phương pháp cấp trang khi có yêu cầu thì tình huống này không thể xảy ra, các trang được nạp vào là các trang thực sự cần đến. Hiện nay người ta vẫn còn tranh luận về phẩm chất của cả 2 chiến lược này.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

6.3 CÁC CHỈ THỊ VÀO/RA ẢO

Bình thường, tập chỉ thị mức 2 hoàn toàn khác tập chỉ thị mức 1, cả các thao tác mà chúng thực hiện lẫn khuôn dạng chỉ thị là hoàn toàn khác. Các chỉ thị giống nhau nằm ở cả mức 1 và mức 2 chỉ là điều tình cờ. Trái lại, tập chỉ thị mức 3 chứa hầu hết các chỉ thị của mức 2 và được bổ sung thêm một số ít chỉ thị mới nhưng quan trọng. Một số ít chỉ thị mức 2 có khuynh hướng dễ gây hỏng cho các máy ở mức dưới được loại đi, không có ở mức 3.

Input/Output là lĩnh vực trong đó máy mức 2 và máy mức 3 khác nhau đáng kể. Có hai nguyên nhân chính dẫn đến sự khác nhau này:

Thứ nhất là để đảm bảo an ninh cho hệ thống. Nếu để cho một người sử dụng ở mức 3 có thể thi hành các chỉ thị thực sự của mức 2 đọc được các dữ liệu bí mật được chứa trong hệ thống hoặc là viết vào terminal của người sử dụng khác, và nói chung có thể gây trục trặc cho chính anh ta, thì đó là những mối đe doạ đối với sự an toàn của hệ thống.

Thứ hai là, bình thường không một người sử dụng nào lại muốn phải tự mình viết các chương trình I/O cho riêng mình. Họ thường muốn hệ điều hành cung cấp các chương trình I/O.

6.3.1 Các chỉ thị vào/ra ảo đối với các file tuần tự

Một cách để tổ chức vào/ra ảo là quan niệm dữ liệu ở bộ nhớ phụ sẽ được đọc hoặc ghi là một dãy các bản ghi logic, trong đó bản ghi logic là một đơn vị thông tin có ý nghĩa đối với người lập trình. Các bản ghi có thể có chiều dài cố định hoặc chiều dài thay đổi được. Một dãy các bản ghi logic được gọi là một file.

Chỉ thị vào ảo cơ sở - READ đọc bản ghi tiếp theo từ một file đã được chỉ ra và đặt nó vào các ô nhớ liên tiếp trong bộ nhớ chính, bắt đầu từ một địa chỉ được chỉ rõ. Để thực hiện thao tác này, chỉ thị ảo phải chỉ ra một cách trực tiếp hay gián tiếp ít nhất là 2 thông tin:

- File sẽ được đọc
- Địa chỉ bộ nhớ chính sẽ chứa bản ghi đọc được.

Không cần phải chỉ ra địa chỉ bên trong file, các chỉ thị READ liên tiếp nhau sẽ lấy các bản ghi logic liên tiếp từ file.

Chỉ thị ra ảo cơ sở - WRITE viết 1 bản ghi vào file. Các chỉ thị WRITE liên tiếp nhau sẽ tạo ra các bản ghi logic liên tiếp trên file.

Ngoài ra còn có thể cần đến chỉ thị ảo REWIND để định vị đầu file.

Nhiều hệ điều hành đòi hỏi phải mở file trước khi sử dụng nó, khi đó cần cung cấp chỉ thị OPEN.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

6.3.2 Các chỉ thị vào/ra ảo đối với các file truy cập ngẫu nhiên

File kiểu tuần tự được nói đến ở phần trên không đánh địa chỉ được. Chỉ thị READ ảo chỉ đọc bản ghi logic kế tiếp. Chương trình không cần cung cấp số hiệu bản ghi logic. Nhiều thiết bị I/O về bản chất là tuần tự, chẳng hạn thiết bị đọc bìa, mỗi chỉ thị đọc mức 3 sẽ đọc vào 1 bìa.

Trong nhiều ứng dụng các chương trình cần đọc/ghi các bản ghi logic một cách ngẫu nhiên. Hầu hết các hệ điều hành cung cấp các chỉ thị ảo để đọc hoặc ghi bản ghi logic thứ n.

Có một dạng tổ chức file khác trong đó các bản ghi logic được đánh địa chỉ không phải bằng vị trí của nó trong file mà bằng nội dung của một trường nào đó bên trong từng bản ghi, trường này được gọi là **khoá - key**.

6.3.3 Việc cài đặt các chỉ thị vào/ra ảo

Để có thể hiểu được các chỉ thị I/O ảo được cài đặt như thế nào ở máy mức 2, cần phải tìm hiểu việc các file được tổ chức và lưu trữ như thế nào.

Vấn đề cơ bản mà hệ thống file phải giải quyết là việc sắp đặt bộ nhớ ngoài. Trong đó có hai vấn đề chính là:

- Kích thước của đơn vị nhớ ngoài sẽ được dùng để sắp đặt chỗ cho file.
- File sẽ được chứa trong các đơn vị nhớ ngoài liên tiếp hay không?

6.4 TÓM TẮT MỨC MÁY HỆ ĐIỀU HÀNH

Hệ điều hành có thể được xem như một trình thông dịch cho các đặc điểm kiến trúc nhất định không có ở mức 2. Các đặc điểm quan trọng nhất của hệ điều hành là bộ nhớ ảo, các chỉ thị vào/ra ảo và các tiện nghi cho việc xử lý song song.

Bộ nhớ ảo là một đặc điểm của kiến trúc máy mức 3, mục đích của nó là cho phép các chương trình sử dụng không gian địa chỉ lớn hơn bộ nhớ vật lý mà máy có.

Sự trừu tượng hoá việc vào/ra quan trọng nhất ở mức 3 là file. Nó bao gồm một dãy các bản ghi logic có thể được đọc hoặc ghi mà không cần phải biết về việc các thiết bị vào/ra như đĩa, băng từ... làm việc như thế nào.

Xử lý song song thường có ở mức 3 và được cài đặt bằng cách mô phỏng nhiều bộ xử lý bằng việc chia sẻ thời gian của một bộ xử lý.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 7 GIỚI THIỆU MỨC NGÔN NGỮ ASSEMBLY

Trong các chương 4, 5 và 6 chúng ta đã thảo luận ba mức khác nhau có trong hầu hết các máy tính ngày nay: Mức Vi chương trình, Mức máy thông thường và Mức máy hệ điều hành. Chương này tập trung vào một mức khác, mức này cũng có trong hầu hết các máy tính hiện đại: Mức ngôn ngữ assembly, nó khác hẳn các mức nêu trên ở chỗ nó được thực hiện bằng cách biên dịch chứ không phải bằng thông dịch.

7.1 GIỚI THIỆU NGÔN NGỮ ASSEMBLY

7.1.1 Ngôn ngữ assembly là gì ?

Các chương trình thực hiện chuyển đổi chương trình của người sử dụng được viết bằng một ngôn ngữ nào đó sang một ngôn ngữ khác được gọi là ***chương trình dịch (translator)***. Ngôn ngữ được sử dụng để viết chương trình nguồn được gọi là ngôn ngữ nguồn còn ngôn ngữ chương trình nguồn chuyển đổi sang gọi là ngôn ngữ đích. Cả ngôn ngữ nguồn và ngôn ngữ đích đều xác định các mức máy.

Dựa vào mối quan hệ giữa ngôn ngữ nguồn và ngôn ngữ đích có thể chia các chương trình dịch thành hai nhóm. Khi ngôn ngữ nguồn về căn bản là một sự biểu diễn bằng ký hiệu cho một ngôn ngữ máy bằng số thì chương trình dịch được gọi là ***assembler*** và ngôn ngữ nguồn được gọi là ngôn ngữ assembly. Khi ngôn ngữ nguồn là một ngôn ngữ bậc cao như là C hoặc Pascal và ngôn ngữ đích là một ngôn ngữ máy bằng số hoặc là một biểu diễn bằng ký hiệu cho một ngôn ngữ như vậy thì chương trình dịch được gọi là ***compiler***.

Ngôn ngữ assembly thuần khiết là ngôn ngữ trong đó mỗi chỉ thị của nó khi được dịch sinh ra đúng một chỉ thị máy. Nói cách khác, có tương ứng một - một giữa các chỉ thị máy và các chỉ thị trong ngôn ngữ assembly. Nếu mỗi dòng trong chương trình assembly chứa một chỉ thị assembly và mỗi word máy chứa một chỉ thị máy, thì chương trình assembly dài n dòng sẽ sinh ra một chương trình ngôn ngữ máy dài n word.

Sử dụng ngôn ngữ assembly để lập trình dễ hơn sử dụng ngôn ngữ máy rất nhiều. Việc sử dụng tên và địa chỉ bằng ký hiệu thay cho số nhị phân (hoặc số hệ 8, hệ 16) tạo nên một sự khác biệt rất lớn. Mọi người đều có thể nhớ được các ký hiệu viết tắt cho cộng (add), trừ (substract), nhân (multiply) và chia (divide) là ADD, SUB, MUL và DIV; nhưng ít người có thể nhớ được các chỉ thị máy cho các phép toán đó là 24576, 57344, 28672 và 29184 (cho máy PDP-11). Người lập trình bằng ngôn ngữ assembly chỉ cần nhớ các tên bằng ký hiệu gợi nhớ ADD, SUB, MUL và DIV, bởi vì assembler sẽ dịch chúng sang các chỉ thị máy. Tuy nhiên nếu có ai đó muốn lập trình bằng ngôn ngữ máy thì họ cần phải nhớ, nếu không thì sẽ phải liên tục tra cứu các giá trị này.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Đối với địa chỉ cũng có thể rút ra các nhận xét tương tự. Người lập trình bằng ngôn ngữ assembly có thể đặt tên bằng ký hiệu gợi nhớ cho các vị trí nhớ và giao phó cho assembler lo cung cấp đúng các giá trị bằng số; trong khi đó người lập trình bằng ngôn ngữ máy luôn luôn phải làm việc với các giá trị bằng số của các địa chỉ.

Hệ quả là từ khi assembler ra đời cho đến nay không còn một ai viết chương trình bằng ngôn ngữ máy nữa.

Ngoài ánh xạ một-một của các chỉ thị assembly vào các chỉ thị máy, ngôn ngữ assembly còn có một tính chất khác nữa làm cho nó khác hẳn các ngôn ngữ lập trình bậc cao, đó là người lập trình bằng ngôn ngữ assembly có thể truy cập tới tất cả các đặc điểm và các chỉ thị có trong máy đích. Người lập trình bằng ngôn ngữ bậc cao không làm được như vậy. Thí dụ nếu máy đích có một bit báo tràn số (overflow bit), thì chương trình bằng ngôn ngữ assembly có thể kiểm tra trực tiếp bit này, trong khi đó chương trình bằng ngôn ngữ Pascal không thể kiểm tra trực tiếp được. Nếu ở bàn điều khiển của một máy tính có các công tắc thì chương trình bằng ngôn ngữ assembly có thể đọc trạng thái của chúng. Một chương trình như vậy có thể thi hành từng chỉ thị trong tập chỉ thị của máy đích, còn chương trình bằng ngôn ngữ bậc cao không thể làm như vậy.

Một sự khác nhau quan trọng nữa giữa chương trình assembly và chương trình bằng ngôn ngữ bậc cao, là chương trình bằng ngôn ngữ assembly chỉ có thể chạy được trên một họ máy, trong khi đó chương trình được viết bằng ngôn ngữ bậc cao nói chung có thể chạy được trên nhiều họ máy, đây chính là một ưu điểm lớn của nó so với ngôn ngữ assembly.

Tóm lại tất cả các việc có thể thực hiện được bằng ngôn ngữ máy đều có thể thực hiện được bằng ngôn ngữ assembly, tuy nhiên ngôn ngữ bậc cao không làm được như vậy một cách có hiệu quả. Các ngôn ngữ dùng cho lập trình hệ thống thường là giao của ngôn ngữ bậc cao và ngôn ngữ assembly, trong đó cú pháp là của ngôn ngữ lập trình bậc cao nhưng lại truy cập được máy bằng ngôn ngữ assembly.

7.1.2 Khuôn dạng chỉ thị ngôn ngữ assembly

Mặc dù cấu trúc của một chỉ thị ngôn ngữ assembly (trong chương này chúng ta sẽ thường gọi ngắn gọn là chỉ thị hay instruction) rất giống cấu trúc của một chỉ thị máy mà nó biểu diễn, các ngôn ngữ assembly cho các máy khác nhau và cho các mức khác nhau lại khá giống nhau, điều đó cho phép chúng ta có thể thảo luận chung về ngôn ngữ assembly. Trên hình 7-01 là đoạn chương trình ngôn ngữ assembly cho các máy dùng bộ vi xử lý 80386, thực hiện việc tính N = I + J + K. Trong đoạn chương trình lấy làm thí dụ, phần chương trình nằm bên trên dòng có dấu chấm thực hiện việc tính toán, còn các dòng nằm bên dưới là các lệnh cho assembler dành ra các ô nhớ cho các biến I, J, K và N chứ không phải là các biểu diễn ký hiệu cho các chỉ thị máy. Các chỉ thị là mệnh lệnh cho assembler thường được gọi là ***chỉ thị assembler***.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Trường nhãn	Trường mã phép toán	Trường toán hạng	Trường chú thích
FORMUL:	MOV	EAX, I	;Nạp giá trị biến I vào thanh ghi EAX
	ADD	EAX, J	;Cộng giá trị biến J vào thanh ghi EAX
	ADD	EAX, K	;Cộng giá trị biến K vào thanh ghi EAX
	MOV	N, EAX	;Chứa kết quả I+J+K vào biến N
	...		
I:	DD	2	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 2
J:	DD	3	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 3
K:	DD	4	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 4
N:	DD	0	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 0

Hình 7-01 Tính N = I + J + K

Chỉ thị - Instruction

Instruction là chỉ thị ngôn ngữ assembly, nó có 4 trường (phần): trường nhãn, trường mã phép toán, trường toán hạng và trường chú thích. Có những chỉ thị chỉ có một trường mã phép toán, có chỉ thị có thêm trường các toán hạng, còn trường chú thích có hay không tuỳ thuộc người viết chương trình.

Nhãn	Mã phép toán	Toán hạng	Chú thích

- **Nhãn (Label):**

Nhãn là một dãy các ký hiệu chữ cái, chữ số... được sử dụng như một tên, gán cho các địa chỉ bộ nhớ, các lệnh trong chương trình có thể dùng tên này thay cho địa chỉ bộ nhớ mà nhãn đại diện. Nếu nhãn đứng trước các chỉ thị của ngôn ngữ assembly thì các lệnh nhảy có thể dùng nhãn này thay cho địa chỉ đích nếu chúng cần nhảy tới lệnh được gán nhãn. Nhãn cũng cần cho các **chỉ thị assembler** có nhiệm vụ sắp đặt bộ nhớ , thí dụ DD (Define Double Word) - lệnh cho assembler dành sẵn 1 word kép (4 byte), để có thể truy cập được dữ liệu cất ở đó thông qua tên nhãn.

Các ngôn ngữ assembly cụ thể có các yêu cầu khác nhau về chiều dài của dãy ký tự dùng làm nhãn, các ký tự được phép sử dụng trong nhãn.v.v.

Trên hình 7-01 chúng ta thấy có 5 nhãn: FORMUL, I, J, K và N, các nhãn kết thúc bằng ký tự ":", đây chỉ là một quy ước cụ thể trong một số ngôn ngữ assembly, không phải là điều gì đặc biệt quan trọng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Mã phép toán** (Opcode - Operation code)

Mã phép toán là chữ viết tắt gợi trí nhớ của ngôn ngữ assembly đại diện cho một chỉ thị máy bằng số. Thí dụ ADD, MOV trên hình 7-01.

- **Trường toán hạng** (operand)

Báo cho bộ xử lý biết phải tìm các toán hạng ở đâu, hay nói cách khác trường này chỉ ra địa chỉ của các toán hạng nếu phép toán có toán hạng. Tất nhiên cũng có những chỉ thị assembly không có toán hạng.

- **Trường chú thích**

Chú thích (comment) thường được quy ước phải viết sau ký tự ';' và kết thúc bởi dấu xuống dòng. Chú thích dùng để diễn giải dòng lệnh trong chương trình để việc đọc được dễ hiểu hơn. Trong một chương trình dài các chú thích rất quan trọng và cần thiết.

Giả chỉ thị - Pseudoinstruction

Trong chương trình viết bằng ngôn ngữ assembly, các **chỉ thị cho assembler** trông cũng giống như các chỉ thị cho bộ vi xử lý thực hiện, vì thế nó còn được gọi là giả chỉ thị - **Pseudoinstruction.** Chỉ thị assembler gồm có 4 trường như sau:

Tên	Chỉ thị assembler	Đối số	Chú thích

- **Tên**

Cũng giống như trường nhãn, chỉ thị assembler có thể có hoặc không có trường tên. Tên là một dãy các ký hiệu chữ cái, chữ số... Có thể dùng tên để gán cho các địa chỉ bộ nhớ và sử dụng tên này như các biến trong các chương trình ngôn ngữ lập trình bậc cao. Cũng có thể gán tên cho các hằng số mà chương trình sẽ sử dụng.

- **Chỉ thị assembler**

Cũng tương tự như chỉ thị (instruction), đó là chữ viết tắt gợi trí nhớ của ngôn ngữ assembly, nhưng đó là lệnh cho chính chương trình dịch - assembler, báo cho nó biết phải làm gì hoặc làm như thế nào. Nói chung không có quy tắc nào giúp ta phân biệt chỉ thị (instruction) và chỉ thị assembler (Pseudoinstruction) cả, chúng đều là các từ viết tắt gợi nhớ, người lập trình bằng ngôn ngữ assembly chỉ có cách tra cứu và dần dần thuộc "mặt" chúng.

- **Trường đối số (argument)**

Tuỳ thuộc vào chỉ thị assembler cụ thể, đối số có thể là một địa chỉ nhớ hoặc một số để sử dụng cùng với chỉ thị và do chỉ thị xác định.

- **Trường chú thích**

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Chú thích thường viết sau ký hiệu ';' và kết thúc bởi dấu xuống dòng (Enter). Các chú thích dùng để diễn giải dòng lệnh trong chương trình, để việc đọc lại được dễ hiểu hơn.

7.1.3 So sánh ngôn ngữ assembly và các ngôn ngữ bậc cao

Trước đây khi cần tạo ra các chương trình sẽ chạy thường xuyên, thí dụ các chương trình của hệ điều hành, nói chung người ta đều viết bằng ngôn ngữ assembly nhằm đạt tốc độ chạy cao và tiết kiệm bộ nhớ. Ngày nay người ta không làm như vậy nữa. Có thể kể ra một số nguyên nhân chính sau đây.

Thứ nhất, giá bộ nhớ giảm không ngừng, việc cố gắng viết chương trình sao cho càng nhỏ càng tốt để tiết kiệm bộ nhớ không còn mấy ý nghĩa nữa. Ngoài ra tốc độ bộ nhớ và CPU cũng ngày càng tăng, việc sử dụng ngôn ngữ assembly nhằm tạo ra các chương trình chạy nhanh hơn trong phần lớn các trường hợp cũng không đem lại ý nghĩa thực tế gì.

Nguyên nhân thứ hai là các chương trình ngày càng lớn và phức tạp, được nhiều người cùng tham gia viết trong thời gian dài, vấn đề dễ đọc, dễ bảo trì, dễ sửa đổi và sử dụng lại có ý nghĩa cực kỳ quan trọng.

Nguyên nhân thứ ba liên quan tới các chương trình dịch (compiler), việc viết các chương trình dịch đã trở thành một công nghệ phát triển, các chương trình dịch ngày nay có thể nhận vào các chương trình nguồn viết bằng ngôn ngữ bậc cao và sinh ra các chương trình mã máy rất hiệu quả, chúng vừa nhỏ vừa chạy nhanh.

Chúng ta sẽ rút ra được các kết luận hữu ích khi so sánh hai hệ điều hành là MULTICS và TSS/67 cho máy IBM 360/67. Cả hai hệ điều hành này đều được bắt đầu viết vào cùng một khoảng thời gian và chúng có kích thước xấp xỉ nhau. Một phần lớn, khoảng 95% hệ điều hành MULTICS được viết bằng ngôn ngữ bậc cao PL/I, còn hệ điều hành TSS/67 của IBM được viết hoàn toàn bằng ngôn ngữ assembly.

Các hệ điều hành lớn như MULTICS và TSS/67 phải thực hiện nhiều nhiệm vụ khác nhau, chẳng hạn phải điều khiển tất cả các thiết bị I/O, giải quyết tất cả các tình huống được định thời chặt chẽ, phải quản lý các cơ sở dữ liệu lớn.v.v. Hiệu suất cao là một vấn đề có tính cốt yếu đối với hệ điều hành, chính vì vậy nó có thể được dùng làm thước đo kiểm tra đối với ngôn ngữ lập trình được sử dụng để viết nên hệ điều hành.

- **Năng suất lập trình tính trung bình theo số dòng lệnh không phụ thuộc ngôn ngữ lập trình được sử dụng**

Kết quả thực hiện hai dự án hệ điều hành trên cho thấy, chúng được hoàn thành trong các khoảng thời gian ngang nhau. Tuy nhiên điều đặc biệt đáng chú ý là nhóm viết MULTICS chỉ có 50 người và chi phí ước tính 10 triệu đôla, trong khi nhóm viết TSS/67 gồm 300 người và chi phí ước tính là 50 triệu đôla (theo Graham, 1970). Việc sử dụng ngôn ngữ lập trình bậc cao PL/I đã tiết kiệm cho dự án

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

MULTICS hàng chục triệu đôla, chi phí cho việc viết chương trình chỉ bằng 20% so với dự án TSS/67.

Các nghiên cứu đã chỉ ra rằng, nếu một người lập trình làm việc cho một dự án trong khoảng thời gian vài năm, thì tính trung bình hàng tháng anh ta viết được từ 100 đến 200 dòng lệnh, không phụ thuộc vào ngôn ngữ lập trình được sử dụng (theo Corbató, 1969). Người lập trình chỉ có thể hy vọng đạt được năng suất lao động lập trình cao với các chương trình nhỏ. Bởi vì một chỉ thị ngôn ngữ PL/I tương đương với từ 5 đến 10 chỉ thị ngôn ngữ assembly, cho nên năng suất của người lập trình PL/I sẽ cao gấp từ 5 đến 10 lần năng suất của người lập trình assembly. Đối với các ngôn ngữ lập trình bậc cao khác, kết quả cũng tương tự như vậy.

- **Chương trình bằng ngôn ngữ bậc cao dễ đọc, dễ hiểu hơn**

Có một lý do quan trọng khác nữa khiến người ta tránh sử dụng ngôn ngữ assembly; Đó là ngay cả một người lập trình assembly chuyên nghiệp nói chung cũng khó mà hiểu được chương trình assembly do người khác viết, nhất là khi chương trình đó dài.

Chương trình nguồn đầy đủ của hệ điều hành MULTICS bằng ngôn ngữ PL/I dài khoảng 300 trang, với chiều dài đó hiểu và nắm vững đã là một việc không hề dễ dàng. Tuy nhiên việc đó cũng chỉ là tầm thường nếu đem so với việc phải đọc 30.000 trang chương trình viết bằng assembly. Mặc dầu không có ai thử cố gắng đọc toàn bộ chương trình nguồn của MULTICS, nhưng cũng có những người cố gắng tìm hiểu các chương trình con cụ thể, chúng có chiều dài trung bình 4 trang bằng ngôn ngữ PL/I.

Trong các dự án lớn sự thay đổi nhân sự hàng năm tính trung bình là 15%, như vậy sau 5 năm trong nhóm lập trình chỉ còn lại rất ít người đã tham gia từ đầu dự án. Nếu như những người lập trình mới gia nhập nhóm không hiểu được chương trình của những người trước họ để lại thì các dự án sẽ gặp một trở ngại rất lớn.

7.1.4 Tinh chỉnh chương trình - Program Tuning

Các nghiên cứu thống kê cho biết rằng, đối với hầu hết các chương trình, chỉ có một phần nhỏ mã lệnh của chúng hoạt động trong phần lớn thời gian thi hành. Tính chung trong thời gian thi hành một chương trình, có 1% của chương trình được sử dụng trong 50% thời gian và 10% được sử dụng trong 90% thời gian. Tình huống điển hình như vậy cũng xảy ra với các compiler, trong đó việc tìm kiếm trong bảng ký hiệu có thể chiếm nhiều thời gian hơn tất cả các công việc còn lại.

Dựa trên kết quả quan sát đó có thể đưa ra một cách tiếp cận mới, có hiệu quả. Đầu tiên sẽ viết chương trình bằng ngôn ngữ bậc cao, sau đó thực hiện một loạt phép đo để xác định các phần của chương trình được sử dụng trong phần lớn thời gian chương trình hoạt động. Chẳng hạn tính thời lượng cho từng thủ tục (procedure), đếm chính xác số lần thực hiện các vòng lặp.v.v. Chúng ta sẽ tìm cách cải tiến, tối ưu hoá một phần nhỏ nhưng quan trọng của chương trình bằng cách viết lại nó bằng ngôn ngữ assembly, quá trình này được gọi là ***Tuning***.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Chúng ta cùng nghiên cứu một thí dụ, giả sử cần 10 năm lao động của một người lập trình để viết một compiler lớn bằng ngôn ngữ bậc cao và compiler đó dịch một chương trình thử nghiệm (test) của chúng ta trong 100s. Nếu viết toàn bộ compiler bằng ngôn ngữ assembly có thể cần từ 50 đến 100 năm lao động, tuy nhiên compiler này dịch chương trình test trên nhanh hơn, chỉ cần có 33s, tức là nhanh hơn 3 lần.

Giả sử 10% của chương trình (viết bằng ngôn ngữ bậc cao) được thực hiện trong 90% thời gian thi hành chương trình, điều này cũng có nghĩa là với một công việc làm trong 100s thì 90s được sử dụng cho các chỉ thị chỉ chiếm 10% của chương trình. Chúng ta sẽ tìm cách cải tiến 10% quan trọng này của chương trình bằng cách viết lại nó bằng ngôn ngữ assembly. Giả sử cần thêm 5 năm lao động để viết lại các thủ tục quan trọng, nhờ đó thời gian thi hành chương trình sẽ giảm từ 90s xuống còn 30s. Các số liệu của thí dụ minh hoạ này cho trên hình 7-02.

	Thời gian viết chương trình (năm)	Thời gian thi hành chương trình (giây)
Ngôn ngữ Assembly	50	33
Ngôn ngữ bậc cao	10	100
Assembly+Ngôn ngữ bậc cao, chưa có sự tinh chỉnh		
10% quan trọng	1	90
90% còn lại	9	10
Tổng cộng =	10	100
Assembly+Ngôn ngữ bậc cao, có tinh chỉnh		
10% quan trọng	6	30
90% còn lại	9	10
Tổng cộng =	15	40

Hình 7-02 So sánh việc lập trình bằng ngôn ngữ assembly và bằng ngôn ngữ bậc cao

Như vậy chương trình sử dụng ngôn ngữ assembly một cách thuần tuý sẽ nhanh hơn 20% ((40-33)/33) so với chương trình sử dụng ngôn ngữ bậc cao hỗn hợp với ngôn ngữ assembly nhưng phải chịu một chi phí gấp 3 lần (50/15). Cách tiếp cận hỗn hợp trên thực tế còn đem lại nhiều ưu điểm hơn thế.

Một người lập trình sử dụng ngôn ngữ bậc cao không bị chìm đắm vào các chi tiết quá vụn vặt và dễ có được sự hiểu thấu đáo về bài toán, cho phép đưa ra các cải tiến thực sự về hiệu suất.

Thế thì tại sao lại phải nghiên cứu việc lập trình bằng ngôn ngữ assembly?, ít nhất có thể nêu ra 3 lý do rất có lý sau:

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Thứ nhất, sự thành công hay thất bại của một dự án lớn có thể phụ thuộc vào việc có thể chắt bóp để nâng cao hiệu suất 5 đến 10 lần một số thủ tục quan trọng không. Khả năng lập trình assembly tốt là rất quan trọng trong những tình huống như vậy.

Thứ hai, trong trường hợp bị thiếu bộ nhớ, sử dụng ngôn ngữ assembly có thể là lựa chọn duy nhất.

Thứ ba, compiler bắt buộc phải sinh ra chương trình đích cho một assembler hoặc là chính nó phải thực hiện quá trình assembly, vì thế hiểu biết ngôn ngữ assembly là cơ sở để hiểu được sự làm việc của compiler.

7.2 QUÁ TRÌNH ASSEMBLY

Mục này sẽ giới thiệu tóm tắt các công việc chính mà assembler phải thực hiện. Mặc dù mỗi máy có một ngôn ngữ assembly khác nhau nhưng quá trình assembly ở các máy rất giống nhau và được trình bày chung ở đây.

7.2.1 Các assembler duyệt hai lần (Two-pass Assembler)

Bởi vì chương trình bằng ngôn ngữ assembly bao gồm một dãy các các dòng, trên mỗi dòng chỉ có duy nhất một chỉ thị nên công việc có vẻ rất tự nhiên của assembler là đọc từng chỉ thị, dịch nó sang ngôn ngữ máy và sau đó ghi mã máy mà nó sinh ra vào một file. Quá trình này cần được lặp lại cho đến khi toàn bộ chương trình được dịch hết. Đáng tiếc là phương pháp "có vẻ rất tự nhiên" này không thể thực hiện được.

Chúng ta xem xét tình huống khi chỉ thị đầu tiên trong chương trình assembly là một chỉ thị nhảy đến L. Assembler không thể dịch được chỉ thị này khi mà nó còn chưa biết được địa chỉ của chỉ thị có nhãn là L. Chỉ thị có nhãn L có thể nằm ở gần cuối chương trình, assembler không thể biết được địa chỉ của chỉ thị này khi nó còn chưa đọc đến vị trí của chỉ thị. Khó khăn này được gọi là ***bài toán tham khảo trước*** (forward reference), trong đó một ký hiệu, chẳng hạn L đã được sử dụng trước khi nó được định nghĩa.

Có thể giải quyết bài toán này bằng hai cách. Cách thứ nhất, assembler thực tế phải đọc chương trình nguồn 2 lần, mỗi lần đọc chương trình nguồn được gọi là một lần duyệt (pass). Tất cả các chương trình dịch có duyệt chương trình nguồn 2 lần đều được gọi là chương trình dịch duyệt 2 lần (two-pass). Trong lần duyệt thứ nhất (pass one), các định nghĩa của các ký hiệu, kể cả nhãn của các chỉ thị được thu thập và chứa vào một bảng. Khi bắt đầu duyệt lần hai (pass two), giá trị của tất cả các ký hiệu đều đã biết, vì thế không còn vấn đề tham khảo trước nữa và mỗi chỉ thị có thể được đọc, dịch và gửi ra. Mặc dù phương pháp này cần đến hai lần duyệt chương trình nguồn nhưng nó rất đơn giản, dễ thực hiện.

Cách thứ hai, chỉ duyệt chương trình nguồn một lần trong quá trình dịch. Cứ mỗi khi gặp một chỉ thị không thể dịch được do chỉ thị này tham khảo trước, thì

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

assembler không sinh ra output, thay vào đó nó sinh ra một đề mục (entry) trong một bảng để chỉ ra rằng chỉ thị có sự tham khảo trước còn chưa được dịch. Cuối của quá trình assembly, mọi ký hiệu đều đã được định nghĩa, vì thế tất cả các chỉ thị trong bảng ghi các chỉ thị còn chưa được dịch nay có thể dịch được.

Phương pháp thứ hai này sinh ra các output theo thứ tự khác với phương pháp thứ nhất, nhưng điều này không gây ra vấn đề gì nghiêm trọng nếu sau quá trình dịch là quá trình ghi vào file, chương trình ghi sẽ sắp xếp lại các đoạn mã theo đúng thứ tự cần thiết. Các assembler loại duyệt một lần sẽ gặp phải một vấn đề nếu có nhiều chỉ thị tham khảo trước, khi đó bảng chứa các chỉ thị chưa được dịch sẽ trở nên quá lớn, có thể không có đủ bộ nhớ cho nó. Ngoài ra phương pháp này cũng có độ phức tạp cao hơn.

Do các nguyên nhân nêu trên, hầu hết các assembler là loại duyệt hai lần.

7.2.2 Lần duyệt thứ nhất - Pass One

Chức năng chính của lần duyệt thứ nhất là xây dựng một bảng gọi là ***bảng ký hiệu (symbol table)*** chứa giá trị của tất cả các ký hiệu. Ký hiệu có thể là một nhãn hoặc là một tên được gán giá trị bằng một giả chỉ thị, thí dụ:

BUFSIZE EQU 100 (EQU là giả chỉ thị, gán cho tên BUFSIZE giá trị 0)

Để có thể sử dụng nhãn như là địa chỉ, trong quá trình duyệt, assembler phải biết được chính xác chỉ thị mà đó đang duyệt sẽ có địa chỉ là bao nhiêu nếu chương trình được thi hành. Muốn thực hiện được điều này assembler phải duy trì một biến trong quá trình assembly có tên gọi là con đếm vị trí chỉ thị hay được viết tắt là ILC (Instruction Location Counter). Biến này được thiết lập bằng 0 ở đầu lần duyệt thứ nhất và cứ mỗi khi xử lý xong một chỉ thị lại được tăng lên bằng chiều dài chỉ thị đó. Hình 7-03 minh hoạ sự thay đổi của ILC trong lần duyệt thứ nhất; Đó là một thí dụ cho bộ xử lý 80386.

Trong lần duyệt thứ nhất các assembler sử dụng ít nhất 2 bảng: bảng ký hiệu (symbol table) và bảng mã lệnh (opcode table).

- Bảng ký hiệu

Trong bảng ký hiệu (minh hoạ trên hình 7-04), ứng với mỗi ký hiệu của chương trình nguồn có một đề mục (entry). Các ký hiệu được định nghĩa bằng cách sử dụng chúng như là nhãn của chỉ thị hoặc là định nghĩa dạng hiển, thí dụ bằng chỉ thị EQU đối với ngôn ngữ assembly cho 80386. Mỗi đề mục của bảng ký hiệu chứa chính ký hiệu đó (hay một con trỏ trỏ tới đó), giá trị bằng số tương ứng với nó và đôi khi có cả các thông tin khác nữa. Các thông tin bổ sung này có thể là:

1. Chiều dài trường dữ liệu gắn với ký hiệu.
2. Các bit báo hiệu tái định vị (ký hiệu có thay đổi giá trị không nếu chương trình được nạp tại một địa chỉ khác với địa chỉ mà assembler giả định?)
3. Ký hiệu có truy cập đến được từ bên ngoài thủ tục hay không.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Nhãn Label	Mã lệnh Operation	Toán hạng Operands	Chú thích Comments	Chiều dài chỉ thị	Con đếm ILC trong khi dịch
	...				
SUZANNE:	MOV	EAX,I	EAX=I	5	100
	MOV	EBX,J	EBX=J	6	105
MARIA:	MOV	ECX,K	ECX=K	6	111
	IMUL	EAX,EAX	EAX=I*I	2	117
	IMUL	EBX,EBX	EBX=J*J	3	119
	IMUL	ECX,ECX	ECX=K*K	3	122
MARILYN:	ADD	EAX,EBX	EAX=I*I+J*J	2	125
	ADD	EAX,ACX	EAX=I*I+J*J+K*K	5	127
CAROL:	MOV	N,EAX	N=I*I+J*J+K*K	5	129
	JMP	DONE	JUMP TO DONE	5	134

Hình 7-03 Con đếm vị trí chỉ thị ILC bám sát địa chỉ mà các chỉ thị sẽ được nạp vào bộ nhớ. Trong thí dụ này các chỉ thị đứng trước SUZANNE chiếm 100 byte.

Ký hiệu	Giá trị	Các thông tin khác
SUZANNE	100	
MARIA	111	
MARILYN	125	
CAROL	129	

Hình 7-04 Bảng ký hiệu cho chương trình trên hình 7-03

- Bảng mã lệnh

Bảng mã lệnh chứa các đề mục, với mỗi mã lệnh bằng ký hiệu ngôn ngữ assembly phải có ít nhất một đề mục. Hình 7-05 là một phần của bảng mã lệnh. Trong mỗi đề mục có chứa mã lệnh bằng ký hiệu, 2 toán hạng, giá trị bằng số của mã lệnh, chiều dài mã lệnh và một con số chỉ lớp (class number), để phân chia các mã lệnh thành các nhóm tuỳ thuộc vào số của mã lệnh và kiểu của các toán hạng.

Thí dụ chúng ta xem xét mã lệnh ADD. Nếu chỉ thị ADD chứa EAX ở toán hạng thứ nhất và một hằng số 32 bit (IMMED32) là toán hạng thứ hai thì opcode 05H được sử dụng và chiều dài chỉ thị là 5 byte (các hằng có thể được biểu diễn bằng 8 hay 16 bit sẽ sử dụng các opcode khác nhau, nhưng trên hình vẽ không thể

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

hiện điều này). Nếu lệnh ADD có hai toán hạng là thanh ghi thì chỉ thị dài 2 byte với opcode là 01H.

Mã lệnh Opcode	Toán hạng thứ nhất	Toán hạng thứ hai	Mã lệnh (số hexa)	Chiều dài chỉ thị	Lớp của chỉ thị
AAA	-	-	37	1	6
ADD	EAX	IMMED32	05	5	4
ADD	REG	REG	01	2	19
AND	EAX	IMMED32	25	5	4
AND	REG	REG	21	2	19

Hình 7-05 Một phần của một bảng mã lệnh

7.2.3 Lần duyệt thứ hai - Pass Two

Chức năng của lần duyệt thứ hai là sinh ra một chương trình đối tượng (object program), ngoài ra có thể thực hiện một vài chức năng phụ khác, chẳng hạn sinh ra một file dạng văn bản, thường được gọi là listing. Trong file listing có toàn bộ chương trình nguồn, đồng thời được bổ sung thêm một số thông tin, như địa chỉ của các dòng lệnh... để người lập trình có thể nghiên cứu.

7.3 KẾT NỐI - LINKING

Hầu hết các chương trình đều bao gồm nhiều thủ tục, các compiler và các assembler thường dịch từng thủ tục một và ghi bản dịch của nó lên đĩa. Trước khi chương trình có thể chạy, cần phải tìm tất cả các thủ tục đã được dịch của nó và kết nối lại với nhau một cách đúng đắn. Chương trình thực hiện chức năng này được gọi là linker, ngoài ra trong một số tài liệu nó còn được gọi bằng một số tên khác như: loader, linking loader và linkage editor.

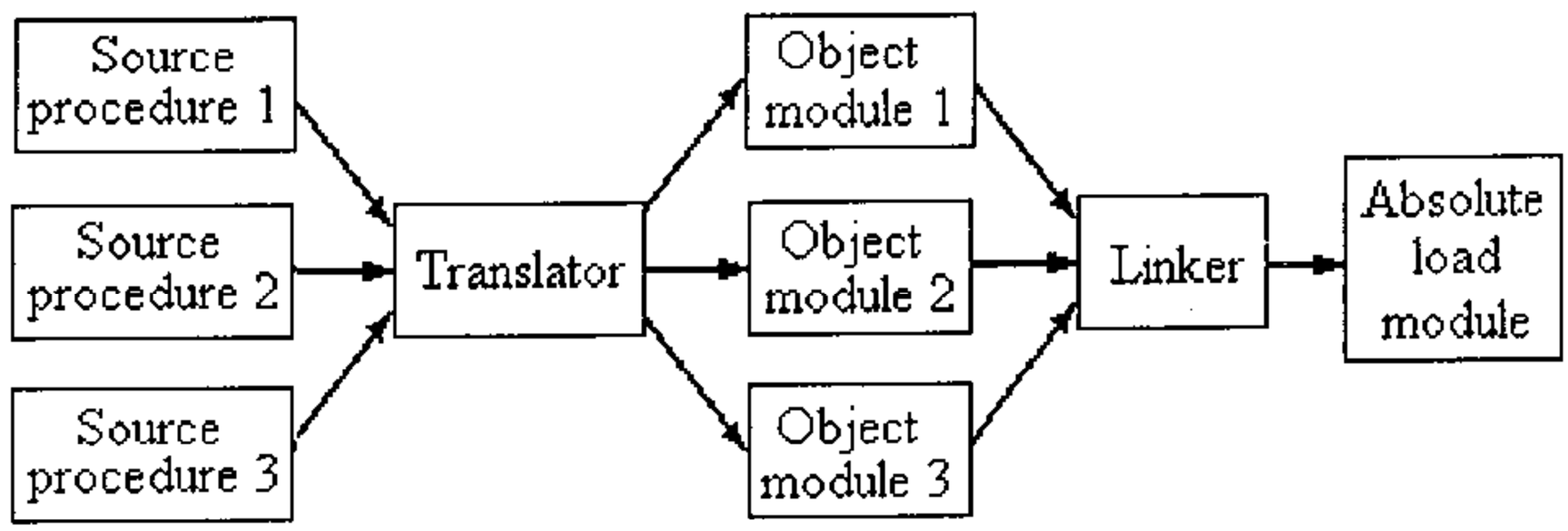
Như vậy việc dịch một chương trình assembly cần hai bước (xem hình 7-06):

1. Dịch các thủ tục của chương trình nguồn (Source procedure) thành các mô-đun đối tượng (Object module).
2. Kết nối các mô-đun đối tượng lại với nhau thành chương trình (Absolute load module).

Bước một được thực hiện bằng assembler hoặc compiler, bước hai sử dụng linker.

Quá trình dịch các thủ tục nguồn thành các mô-đun đối tượng làm thay đổi về mức, bởi vì ngôn ngữ nguồn và ngôn ngữ đíoh có các chỉ thị khác nhau. Quá trình kết nối (linking) không làm thay đổi về mức, bởi vì cả input và output của các chương trình linker đều là các chương trình cho cùng một máy ảo.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 7-06 Hai bước dịch một chương trình assembly

Chức năng của chương trình linker là thu thập các thủ tục đã được dịch một cách riêng rẽ và kết nối chúng lại với nhau để có thể cho chạy như một chương trình.

Mô-đun đối tượng bao gồm 6 phần, được mô tả trên hình 7-07.

Identification
Entry point table
External reference table
Machine instructions and constants
Relocation dictionary
End of module

Hình 7-07 Cấu trúc bên trong của mô-đun đối tượng do assembler sinh ra

Phần thứ nhất - phần định danh (Identification) chứa tên của mô-đun, một số thông tin nhất định cần cho linker, chẳng hạn chiều dài các phần khác nhau của mô-đun đối tượng và đôi khi là ngày thực hiện quá trình assembly.

Phần thứ hai - bảng điểm vào/ bảng ký hiệu toàn cục (entry point table) chứa danh sách các ký hiệu được định nghĩa trong mô-đun cùng với giá trị của chúng, các ký hiệu này có thể tham khảo được từ các mô-đun khác. Thí dụ, nếu mô-đun chương trình nguồn chứa một thủ tục tên là BIGBUG thì entry point table sẽ chứa dãy ký tự "BIGBUG", tiếp theo nó là địa chỉ tương ứng với nó. Người lập trình bằng ngôn ngữ assembly chỉ ra các ký hiệu nào được khai báo là điểm vào (entry point) hoặc là ký hiệu ngoại lai (external symbols) bằng cách sử dụng các giả chỉ thị.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Phần thứ ba - bảng tham chiếu ký hiệu ngoại lai (external reference table), chứa danh sách các ký hiệu được sử dụng trong mô-đun này nhưng lại được định nghĩa trong các mô-đun khác, kèm theo là một danh sách cho biết chỉ thị nào sử dụng ký hiệu nào. Chương trình linker cần danh sách thứ hai này để có thể điền các địa chỉ đúng vào các chỉ thị có sử dụng các ký hiệu được định nghĩa bên ngoài mô-đun này. Một thủ tục có thể gọi các thủ tục khác được dịch một cách độc lập với nó, bằng cách khai báo tên của thủ tục sẽ bị gọi là loại ngoại lai - exterrnal. Người lập trình assembly chỉ ra ký hiệu nào được khai báo là external bằng cách dùng một giả chỉ thị (trong ngôn ngữ của Macro assembler, đó là extrn). Trên một số máy tính hai bảng entry point và external reference được kết hợp lại làm một.

Phần thứ tư (Machine instructions and constants) của mô-đun đối tượng chứa các mã máy và các hằng số. Chỉ có phần thứ tư này của mô-đun đối tượng là sẽ được nạp vào bộ nhớ khi chương trình được cho thi hành. Năm phần còn lại chỉ để cho linker sử dụng và sẽ được loại bỏ sau khi nó hoàn thành nhiệm vụ của mình.

Phần thứ năm chứa một bảng được gọi là Từ điển tái định vị (Relocation dictionary). Bởi vì trong phần thứ tư của mô-đun đối tượng có chứa cả các chỉ thị máy lẫn các hằng cho nên cần phải có từ điển này để cho linker có thể định lại địa chỉ của các chỉ thị khi nó kết nối các mô-đun đối tượng lại với nhau.

Phần thứ sáu - chỉ báo kết thúc mô-đun đối tượng (End of module). Một số assembler ghi trước chỉ báo kết thúc này một đoạn dữ liệu, gọi là checkssum, dùng cho việc phát hiện sai trong file đối tượng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 8 CÁC THIẾT BỊ NGOẠI VI

8.1 KHÁI QUÁT

8.1.1 Khái niệm về thiết bị ngoại vi

Trong hệ thống máy tính, thiết bị ngoại vi (peripherals) là một thuật ngữ dùng để chỉ một miền rất rộng các thiết bị, từ các vi mạch, như vi mạch thu/phát dị bộ đa năng (UART: Universal Asynchronization Receiver-Transmitter), cho đến các hệ hoàn chỉnh như ổ đĩa mềm, ổ đĩa cứng, bàn phím, màn hình, máy in, modem.v.v. Nói một cách tổng quát, mọi mạch logic liên hệ với bộ vi xử lý, trừ mạch tạo xung đồng hồ và bộ nhớ trong, cần thiết cho hoạt động cơ sở của máy tính, đều có thể xem là thiết bị ngoại vi.

Các thiết bị ngoại vi thường liên hệ với CPU thông qua các cổng vào/ra và cung cấp cho hệ thống những khả năng logic phụ, hoặc được sử dụng để nối ghép hệ vi xử lý với những hệ thống phi điện tử. Trong một số trường hợp, những chức năng của thiết bị ngoại vi có thể được thực hiện bằng phần mềm, cho phép thay thế hoàn toàn phần cứng (thí dụ như việc in các font chữ khác nhau trên máy in kim) hoặc kết hợp giữa phần mềm và các thiết bị ngoại vi rẻ tiền để nâng cao tính năng hoạt động của chúng. Khi thiết kế các thiết bị ngoại vi, người ta đặc biệt chú ý tới sự kết hợp giữa phần cứng và phần mềm.

Các thiết bị ngoại vi có thể được chia làm 2 loại: thiết bị vào và thiết bị ra. Có một số thiết bị thực hiện cả 2 chức năng vào và ra, thí dụ các ổ đĩa.

8.1.2 Nhiệm vụ cơ bản và yêu cầu đối với thiết bị ngoại vi

Nhìn chung các thiết bị ngoại vi có 2 nhiệm vụ cơ bản:

- Làm phương tiện trao đổi thông tin giữa hệ thống máy tính và con người hoặc giữa các hệ thống máy tính với nhau.
- Làm các bộ nhớ trung gian, với các dung lượng khác nhau.

Yêu cầu chung đối với các thiết bị ngoại vi:

- Thuận lợi cho người sử dụng
- Tốc độ vào/ra số liệu phải cao
- Các lỗi xuất hiện trong quá trình vào/ra phải được xử lý kịp thời và dễ dàng ngay sau khi xuất hiện.

8.1.3 Nguyên lý cấu tạo và đặc điểm của thiết bị ngoại vi

Nhiều thiết bị ngoại vi áp dụng nguyên lý cơ điện và cũng có những thiết bị ngoại vi làm nhiệm vụ giao tiếp giữa người và máy tính, vì vậy tốc độ hoạt động của các thiết bị ngoại vi này phụ thuộc rất nhiều vào tốc độ thao tác của người sử dụng và vào chính các giác quan của con người.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Do các đặc điểm nêu trên, các thiết bị ngoai vi thường có chung một nhược điểm lớn, đó là tốc độ làm việc chậm hơn tốc độ hoạt động của CPU rất nhiều. Thí dụ, tốc độ của một số máy in kim chỉ nằm trong khoảng một vài trăm ký tự/giây, tốc độ đưa dữ liệu ra màn hình lại phụ thuộc hiện tượng lưu ảnh trên võng mạc.v.v.

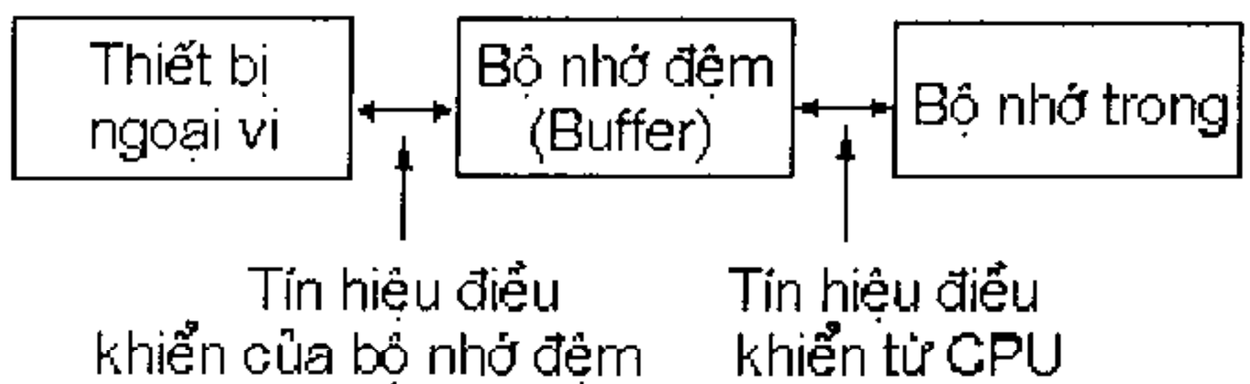
Để khắc phục nhược điểm nêu trên của thiết bị ngoại vi, người ta thường sử dụng phương pháp nhớ đệm. Một vùng nhớ RAM có thể là thuộc CPU hoặc thuộc chính thiết bị ngoại vi được sử dụng làm bộ nhớ đệm - buffer. Theo phương pháp này sự truyền thông tin giữa CPU và thiết bị ngoại vi được chia thành 2 bước:

Bước 1: Truyền thông tin giữa bộ nhớ trong (CPU) và bộ nhớ đệm. Tốc độ truyền bằng tốc độ hoạt động của CPU, quá trình này thực hiện theo các lệnh từ CPU.

Bước 2: Truyền thông tin giữa bộ nhớ đệm và thiết bị ngoại vi. Tốc độ truyền được quyết định bởi tốc độ hoạt động của thiết bị ngoại vi, quá trình truyền thông tin được điều khiển nhờ tín hiệu điều khiển của bộ nhớ đệm còn bộ nhớ đệm lại được điều khiển bởi tín hiệu của CPU. (Xem hình 8-01)

(Với quá trình truyền thông tin từ thiết bị ngoại vi về CPU, các bước hơi khác trên một chút, không trình bày ở đây)

Trong khoảng thời gian của bước 2 (trao đổi thông tin giữa bộ nhớ đệm và thiết bị ngoại vi) thì bộ nhớ trong sẽ không liên kết trực tiếp với bộ nhớ đệm, nhờ vậy mà CPU có thể làm các việc khác, chẳng hạn xử lý các thông tin ở bộ nhớ trong hoặc gửi/nhận thông tin với các thiết bị ngoại vi khác.v.v. Việc này làm cho tốc độ hoạt động tương đối thấp của thiết bị ngoại vi không gây ảnh hưởng trực tiếp đến khả năng làm việc của CPU; Người ta có thể tổ chức để cho một CPU có thể làm việc đồng thời với nhiều thiết bị ngoại vi, theo nguyên tắc phân chia thời gian.



Hình 8-01 Trao đổi dữ liệu giữa CPU và thiết bị ngoại vi

Theo cơ chế hoạt động này mỗi một bộ nhớ đệm phải được trang bị một thiết bị chỉ báo trạng thái bận, nếu thông tin chứa đựng trong một bộ nhớ đệm nào đó chưa được đưa hết ra, thì các lệnh Input/Output với bộ nhớ đệm này không được phép thực hiện. Trong khi thực hiện truyền tin giữa CPU và bộ nhớ đệm thì thiết bị ngoại vi có thể coi như nằm trong trạng thái nghỉ, tuy nhiên thời gian này nói chung là rất ngắn, vì cả hai bộ nhớ đó đều là RAM.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

8.2 BÀN PHÍM - KEYBOARD

8.2.1 Giới thiệu

Bàn phím là thiết bị lối vào thuộc loại đơn giản nhất trong một hệ thống máy tính, nó bao gồm một tập hợp các công tắc, thường được bố trí thành một ma trận. Tín hiệu lối ra của ma trận công tắc này được đưa vào mạch tạo mã bàn phím, với mỗi tổ hợp phím xác định được ấn xuống mạch tạo mã sẽ tạo ra một con số nhận diện cho phím đó, sau đó con số này được gửi vào CPU.

Thiết kế của bàn phím luôn luôn được cải tiến và rất khác nhau giữa các nhà sản xuất, từ dạng của các tiếp điểm tới cách tổ chức các tiếp điểm. Trong các loại bàn phím cũ, lối ra của bàn phím là một bó dây song song, nối từ các công tắc tới CPU, sau này người ta tìm cách giảm số dây nối xuống chỉ còn vài dây (4 hoặc 5 trong các bàn phím dùng cho máy IBM PC hoặc tương thích với IBM PC).

Trong nhiều loại bàn phím người ta sử dụng các tiếp điểm điện, trạng thái được ấn xuống của một phím sẽ chuyển thành trạng thái nối mạch điện của một công tắc, sự thay đổi trạng thái của công tắc sẽ được biến đổi thành sự thay đổi mức logic.

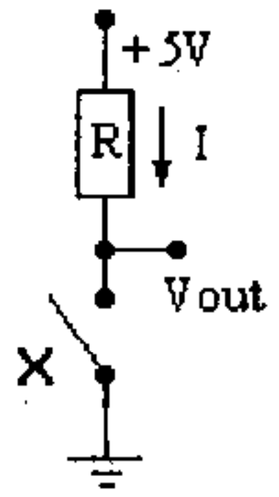
Cũng có loại công tắc sử dụng sự thay đổi điện dung hay điện cảm của một mạch điện, sự thay đổi đó cũng sẽ được biến đổi thành sự thay đổi mức logic tương ứng, ở loại bàn phím này không có tiếp xúc điện của tiếp điểm, nên cũng có một số ưu điểm nhất định.

8.2.2 Chuyển mạch cơ khí

Cho tới giai đoạn hiện nay, các loại bàn phím sử dụng chuyển mạch cơ khí vẫn chiếm tỷ lệ lớn, chúng có ưu điểm là đơn giản, giá thành thấp, độ tin cậy cao, nhưng cũng có nhược điểm là các tiếp điểm dễ bị bẩn, các lò xo lâu ngày thường giảm đàn hồi, hay bị 'dính phím' .v.v.

- **Việc tạo mức logic**

Nguyên lý của việc tạo mức logic tương ứng với trạng thái nối hoặc không nối hai tiếp điểm của một công tắc điện được minh hoạ ở hình vẽ 8-02.



Hình 8-02 Nguyên lý tạo mức logic ứng với trạng thái của chuyển mạch cơ khí

Khi chuyển mạch nối mạch thì Vout bị nối xuống 'đất', mức điện áp ra bằng 0v, tương ứng với mức logic 0. Dòng điện chạy qua tiếp điểm là cực đại: I = 5v/R, lúc này R đóng vai trò điện trở hạn chế dòng điện đi qua tiếp điểm.

Khi chuyển mạch hở (vị trí ngược lại với trước) điện áp ra là Vout = 5v-IR, dòng I≠0 do mạch điện tử nối với tiếp điểm có tiêu thụ một dòng điện nhỏ, tuy nhiên thường là không đáng kể, do đó Vout xấp xỉ bằng 5v, tương ứng với mức logic 1.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

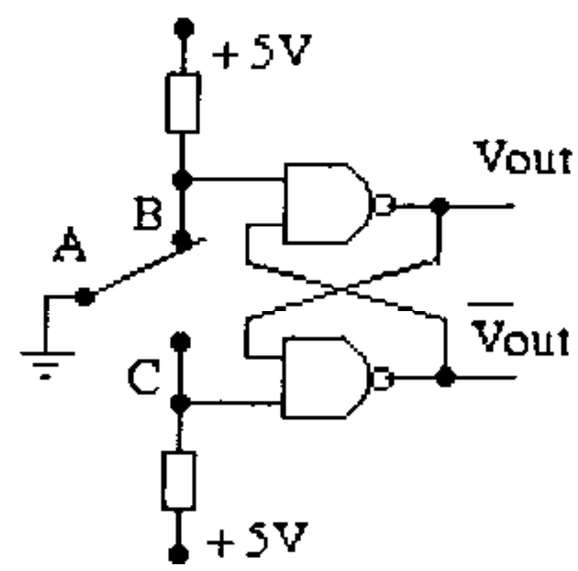
- **Sự rung của tiếp điểm**

Các nghiên cứu thực nghiệm cho biết, ở các loại tiếp điểm cơ khí, khi đóng hoặc mở có ***hiện tượng rung*** - đó là sự đóng mở công tắc nhiều lần trước khi ổn định, hiện tượng này cần phải được khử bỏ, nếu không CPU sẽ ghi nhận như là công tắc được ấn, nhả nhiều lần chứ không phải một lần như ta mong muốn. Việc khử nhiễu rung có thể thực hiện bằng phần cứng hoặc bằng phần mềm.

Khử nhiễu rung bằng phần cứng:

Sử dụng một Flip-Flop loại RS, flip-flop chỉ thay đổi trạng thái một lần khi chuyển mạch thay đổi vị trí. Xem hình vẽ minh hoạ 8-03.

Nếu A nối với B thì $V_{out} = 1$, dù tiếp điểm tại B có bị rung và tách ra rồi lại nối lại nhiều lần. V_{out} chỉ thay đổi (đảo) trạng thái khi A nối với C (thì $V_{out} = 0$).



Hình 8-03 Mạch điện khử ảnh hưởng của hiện tượng rung tiếp điểm cơ khí

Khử nhiễu rung bằng phần mềm:

Bộ vi xử lý chuyên dụng trong thiết bị bàn phím đọc đầu ra của chuyển mạch, nếu giá trị đọc được giống nhau trong một khoảng thời gian xác định nào đó thì chỉ ghi nhận một giá trị.

Mặc dù chuyển mạch đã được khử nhiễu rung cơ khí bằng phần cứng hay phần mềm, bộ vi xử lý vẫn phải đọc giá trị đầu ra của chuyển mạch một cách đều đặn để xác định được trạng thái của các phím.

8.2.3 Các phương pháp tạo mã bàn phím

- **Các phím được sắp xếp song song**

Ngày nay hầu như tất cả các bàn phím đều có mạch tạo mã bàn phím nằm ngay bên trong nó, nhưng trước đây ở các loại bàn phím cổ, các dây nối với các chuyển mạch đều được đưa song song tới bộ tạo mã nằm trong máy, vì vậy tổng số dây thường là khá lớn.

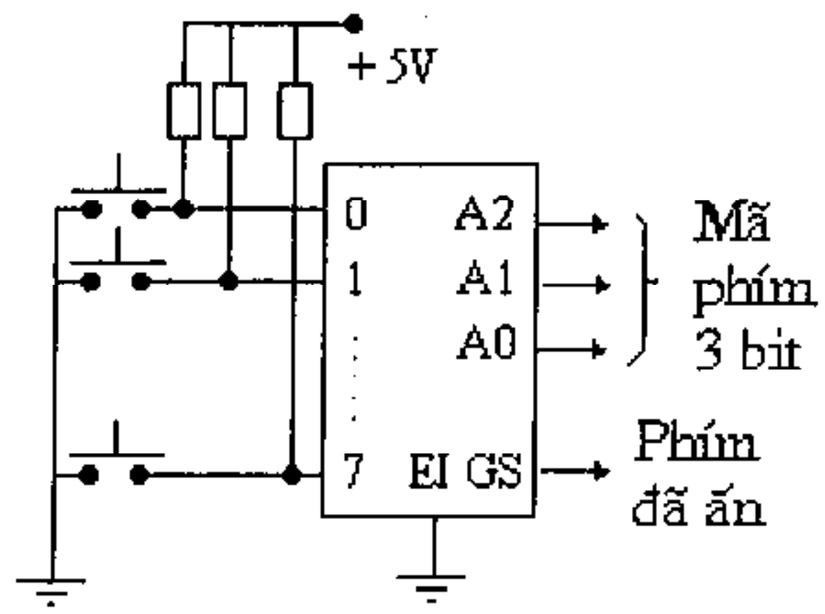
Hình 8-04 là mạch tạo mã 8 đầu vào, 3 đầu ra là A0, A1,A2, ngoài ra còn có một đầu ra đặc biệt GS, nó sẽ ở mức logic tích cực khi có một trong các phím được ấn, nếu GS có mức không tích cực thì tín hiệu trên các dây A0, A1 và A2 sẽ không được CPU đọc. Mức logic ở đầu ra GS này có thể được một mạch điện hay một chương trình sử dụng để khử nhiễu rung cơ khí, hoặc được chốt lại và sử dụng để tạo ngắt, yêu cầu bộ vi xử lý đọc đầu ra của mạch tạo mã.

Cách sắp xếp song song các phím chỉ thuận tiện khi số phím là nhỏ, chẳng hạn không quá 16 phím.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Các phím được sắp xếp thành một ma trận**

Các bàn phím hiện nay thường có số lượng phím lớn, nếu vẫn sắp xếp các phím theo kiểu song song thì số dây dẫn và số đầu vào, ra của mạch tạo mã sẽ rất lớn. Để khắc phục, người ta bố trí các phím thành một ma trận vuông hoặc chữ nhật, các tiếp điểm (phím) nằm ở chỗ giao nhau của các dây hàng và các dây cột, bằng cách này mạch điện tử lập mã phím sẽ đơn giản hơn.

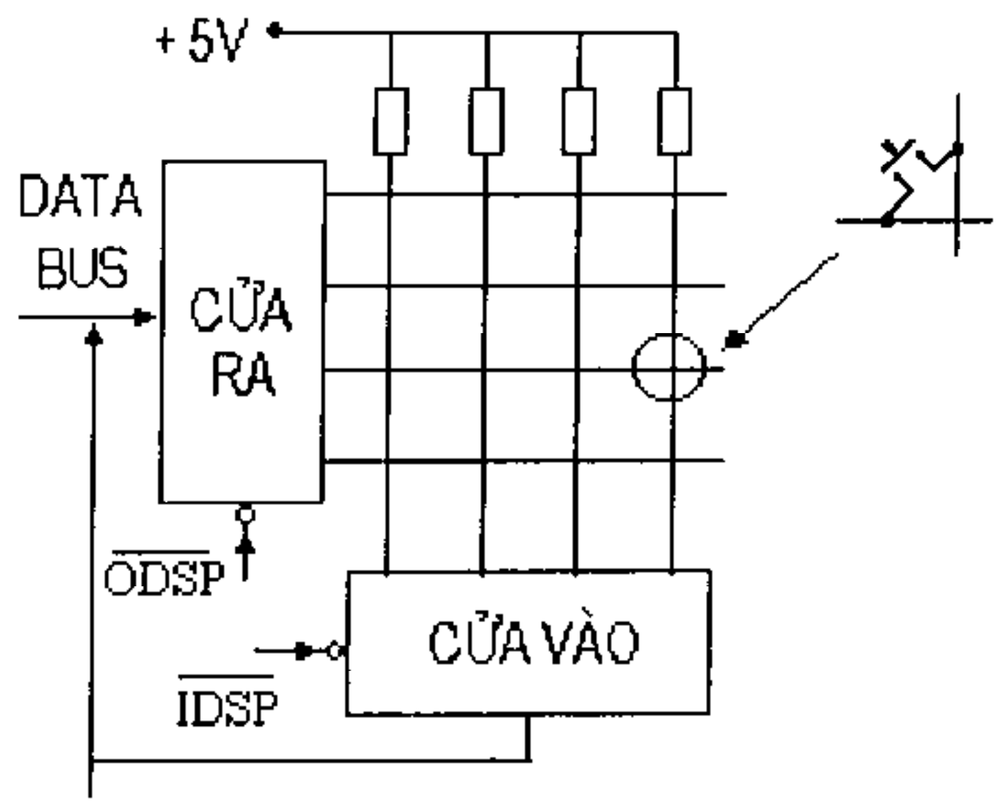


Hình 8-04 Mạch tạo mã bàn phím có 8 phím

Lập mã phím (Encoding)

Để tạo mã phím cần phải thực hiện một công việc gọi là quét bàn phím (hoặc quét ma trận), việc này có thể thực hiện bằng phần cứng cũng như bằng phần mềm. Việc quét bàn phím bằng phần mềm được thực hiện như sau (xem hình 8-05):

- Lần lượt đặt từng dây hàng của ma trận xuống mức logic 0, trong khi các dây hàng khác của ma trận vẫn ở mức logic 1. Việc này được đơn vị logic "Cửa ra" thực hiện, đó thực chất là một bộ giải mã (decoder) với tất cả các đầu ra được đảo 1 thành 0 và 0 thành 1. Đầu vào chính là "data BUS", giá trị trên đầu vào này sẽ chọn lấy 1 hàng ra bằng 0, còn các hàng khác bằng 1. Tín hiệu điều khiển $\overline{ODSP}$ (Output Device Select Pulse) khi có mức tích cực (thấp) sẽ chốt đầu vào, làm cho đơn vị "Cửa ra" được cách ly khỏi tín hiệu trên bus vào "data BUS". Do đó bus này có thể được sử dụng để truyền tín hiệu khác.



Hình 8-05 Cấu trúc phần cứng của ma trận phím để quét bằng phần mềm

- Trong khi một hàng của ma trận đang có mức 0, kiểm tra từng dây cột xem có dây nào có mức logic 0 hay không, nếu có thì chứng tỏ phím nối với cột đó và hàng đang xét đang được ấn. Việc này được đơn vị logic "Cửa vào" thực hiện. Đó thực chất là một encoder (xem mục 4.1.3 Bộ dồn/phân kênh và bộ giải mã) với các lối vào được đảo, chức năng của nó ngược lại so với decoder. Nó nhận vào giá trị trên các dây cột và gửi ra đầu ra (tới "data BUS") một giá trị nhị phân ứng với cột có mức 0. Tín hiệu điều khiển $\overline{IDSP}$ (Input Device Select Pulse) khi có mức tích cực (thấp) sẽ chốt đầu ra của đơn vị "Cửa vào", cách ly nó khỏi tín hiệu trên bus vào "data BUS", khi đơn vị "Cửa ra" nhận dữ liệu từ "data BUS".

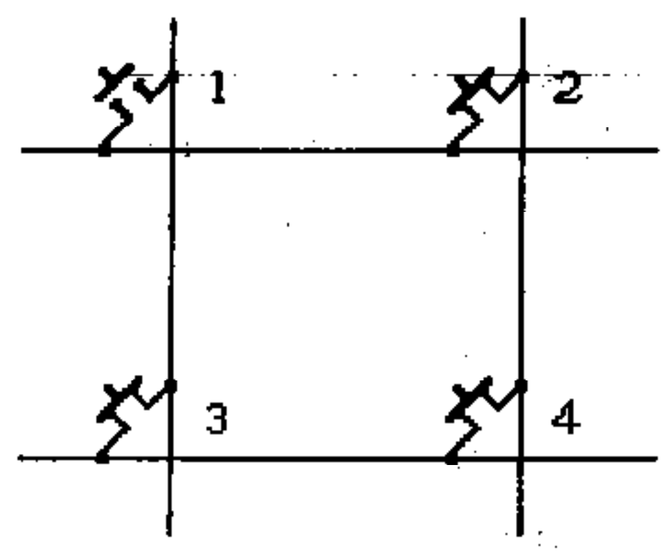
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Chương trình quét bàn phím sẽ lần lượt gửi địa chỉ từng hàng ra "data BUS", sau đó nhận địa chỉ cột có phím bị ấn (nếu có) gắn với hàng đó. Quá trình này phải được lặp đi lặp lại đủ nhanh để tránh hiện tượng có phím được ấn và nhả mà không được phát hiện.

Một số nhiệm vụ khác của mạch lập mã phím:

- Mạch tạo mã phải loại trừ các tổ hợp phím không cho phép, ví dụ khi xảy ra hiện tượng đường dẫn lén, là hiện tượng có những phím không được ấn nhưng mạch quét bàn phím xác định nhầm là có ấn. Thí dụ minh hoạ trên hình vẽ 8-06 dưới đây, phím số 1 dường như được ấn, trong khi đó thực tế chỉ có các phím 2, 3 và 4 được ấn.

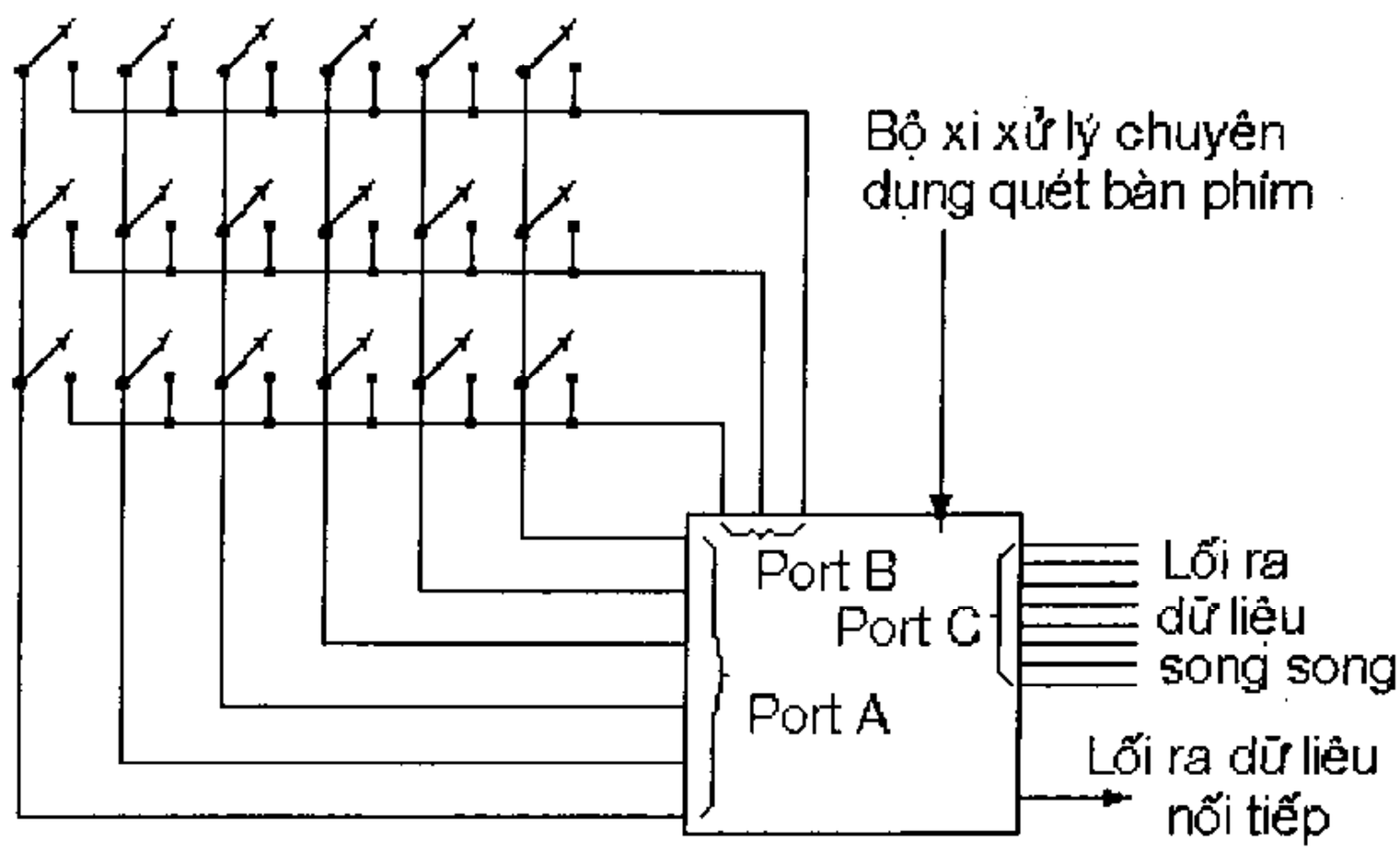
- Ngoài ra chương trình con quét bàn phím còn thực hiện việc khử nhiễu rung cơ khí và phân biệt một phím đã được ấn nhiều lần hay chỉ được ấn một lần song được giữ xuống trong một một khoảng thời gian dài.



Hình 8-06 Hiện tượng đường dẫn lén

- Xử lý trạng thái ấn nhiều phím cùng lúc (trạng thái Rollover). Chương trình con quét bàn phím phải giải quyết vấn đề này, nó thực hiện thuật toán sau: nếu có nhiều phím cùng được ấn thì giá trị của tổ hợp phím được ghi nhận phụ thuộc vào thứ tự phát hiện của chương trình con chứ không phụ thuộc vào thứ tự mà các phím được ấn. Một số tổ hợp phím được xem là hợp lệ - sẽ được tạo một mã tương ứng, các tổ hợp không hợp lệ sẽ không được tạo mã.

Ngày nay, nhờ sự phát triển của kỹ thuật vi điện tử, các nhà sản xuất bàn phím đã có thể thu gọn toàn bộ mạch tạo mã vào trong một con chip, đó là một bộ vi xử lý chuyên dụng tiến hành quét ma trận phím sau đó tạo mã tín hiệu ra như minh hoạ trên hình 8-07.



Hình 8-07 Bộ tạo mã bàn phím sử dụng một chip vi xử lý

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Một chương trình mô phỏng theo hoạt động của mạch phần cứng sẽ tiến hành quét ma trận phím và mã hoá vị trí của phím đang được ấn thành một số nhị phân. Việc chuyển đổi thành mã số tương ứng với các phím này chỉ được thực hiện bằng chương trình chứ không yêu cầu thêm một mạch phần cứng nào cả.

Việc làm này rất có hiệu quả kinh tế và rất linh hoạt. Dữ liệu sau khi được mã hoá, lại được chính mạch vi xử lý này chuyển thành dạng bit nối tiếp. Chính nhờ vậy mà số dây truyền dữ liệu từ bàn phím về CPU rút xuống chỉ còn 3,4 hay 5, làm cho cáp nối có thể làm rất nhỏ và mềm mại.

8.2.4 Mã Scan của bàn phím máy IBM PC hoặc tương thích

Bàn phím IBM PC chuẩn chứa chip vi xử lý điều khiển bàn phím 8048, nó thực hiện rất nhiều công việc khác nhau đỡ cho CPU:

- Theo dõi xem có phím nào được ấn không thì báo cho CPU thông qua ngắt số 9. Nếu có một phím nào đó bị ấn quá 1/2 giây thì 8048 sẽ lặp lại phím này sau những khoảng thời gian nhất định, việc lặp bằng phần mềm như vậy được gọi là Typematic.
- Có một số khả năng hạn chế về chẩn đoán lỗi
- Có vùng đệm có thể lưu trữ 20 tác động phím, nếu như CPU chưa tiếp nhận kịp (tình huống này ít khi xảy ra).

Máy IBM PC/AT sử dụng chip vi xử lý khác là 8042, với các chức năng chủ yếu cũng tương tự như vậy.

Mã Scan

Mỗi lần ta ấn một phím, các mạch điện của bàn phím sẽ tạo ra một mã dài một byte gọi là mã scan, đặc trưng cho vị trí trên bàn phím của phím có tác động, giá trị nằm trong khoảng 1.. 83. Khi ta nhả phím đang bị ấn ra, bàn phím tạo ra một mã scan khác với mã scan lúc phím bị ấn, có giá trị bằng mã trước cộng thêm 128 (80H), nghĩa là đổi bit 7 của byte mã scan trước đó từ 0 thành 1. Ví dụ khi ta ấn chữ z, mã scan của nó là 44, lúc nhả phím ra, bàn phím lại tạo mã scan 172 (=44+128).

8.2.5 Liên lạc giữa bàn phím và CPU ở IBM PC/XT/AT

Cứ mỗi lần có một tác động phím (nhấn hoặc nhả), mạch điện tử của bàn phím lại gây ra ngắt 9, gọi một chương trình phục vụ bàn phím của ROM_BIOS, đó là INT 9 theo cách gọi của ngôn ngữ assembly (INTerrupt - ngắt), INT 9 sẽ đọc cổng 60h để biết tác động phím nào đã xảy ra - tức là đọc mã scan tương ứng.

INT 9 sẽ chuyển đổi mã scan đó thành mã dài 2 byte, byte thấp chứa mã ASCII của phím đó, byte cao chứa mã scan. Với các phím đặc biệt như: các phím chức năng, các phím ở phần bàn phím số (numeric keypad), byte thấp chứa 0 và byte cao chứa mã scan. Sau đó INT 9 sẽ đặt mã 2 byte này vào hàng đợi bàn phím, bắt đầu tại địa chỉ 0: 041EH, kéo dài 32 byte ứng với 16 phím. Đầu vùng đệm này được trỏ bởi vector 0:041AH, đuôi trỏ bởi vector 0: 041CH, nếu 2 vector trùng nhau, có nghĩa là bộ đệm bàn phím rỗng. Các mã này sẽ nằm yên ở đây cho đến khi được

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

một chương trình như DOS, BASIC.v.v. lôi dần ra sử dụng, mỗi khi chúng cần đọc từ bàn phím.

CPU dành riêng 2 byte tại địa chỉ 0:417 và 0:418 để lưu trạng thái của các phím lật, đó là các phím Shift, CapsLock, Numlock.v.v. Chương trình phục vụ bàn phím INT 9 trong ROM_BIOS thường xuyên kiểm tra các byte này trước khi chuyển mã scan sang mã 2 byte.

Các phím tổ hợp

Khi chương trình của ROM_BIOS đang chuyển đổi mã scan, nó luôn luôn kiểm tra một số tổ hợp phím, đặc biệt là các tổ hợp Ctrl_Alt_Del, Shift_PrtScr, Ctrl_NumLock, Ctrl_Break. Bốn tác động phím này có tác dụng như các lệnh, buộc chương trình của ROM_BIOS phải thực hiện ngay tức khắc chứ không lưu lại mã của các phím tổ hợp này vào trong hàng đợi bàn phím.

8.3 MÀN HÌNH (VIDEO DISPLAY, MONITOR)

Đối với các máy vi tính hiện nay, phương tiện đối thoại chủ yếu người-máy là bàn phím + màn hình. Màn hình là phương tiện hiển thị thông tin thuận lợi và kinh tế. Thông tin hiển thị là chữ, số (text) hoặc đồ hoạ (graphics).

Hiện nay có hai loại màn hình đang được sử dụng phổ biến là: Màn hình tia âm cực (CRT - Cathode Ray Tube) và Màn hình tinh thể lỏng (LCD - Liquid Crystal Display).

Loại CRT thông dụng hơn do giá thành thấp hơn nhiều và có khả năng hiển thị thông tin, màu sắc rất phong phú, đồng thời việc điều khiển CRT đã do các mạch LSI chuyên dụng đảm nhiệm.

Loại màn hình LCD có ưu điểm là mỏng, gọn nhẹ và tiêu thụ ít năng lượng, tuy vậy hiện tại khả năng thể hiện màu sắc và ánh sáng còn chưa bằng loại CRT, hơn nữa giá thành còn cao, chính vì vậy nó chưa được sử dụng phổ biến.

8.3.1 Nguyên lý của phương pháp hiển thị hình ảnh video

Màn hình máy tính tuy có kích thước hữu hạn nhưng về mặt toán học lại chứa vô hạn điểm. Để có thể hiển thị thông tin cần phải lưu trữ các thông tin mô tả thuộc tính sáng của từng điểm trên màn hình trong bộ nhớ, ngoài ra còn phải truyền các thông tin của từng điểm từ bộ nhớ ra thiết bị màn hình. Việc lưu trữ hoặc truyền một lượng thông tin lớn vô hạn là không thể thực hiện được, chính vì vậy mà người ta phải nghiên cứu kỹ các đặc điểm của thị giác con người để dựa vào đó xây dựng các phương pháp hiển thị tốt nhất có thể thực hiện được một cách hữu hiệu.

- **Khả năng phân giải hữu hạn của mắt người**

Khả năng phân giải của mắt người là khoảng 1' (góc 1 phút), nghĩa là nếu chúng ta nhìn 2 điểm dưới một góc nhỏ hơn 1' thì sẽ cảm nhận thấy chúng dính vào nhau, góc đó còn được gọi là ***góc phân giải***. Điều này hết sức quan trọng đối với việc hiển thị thông tin, vì chúng ta chỉ cần hiển thị một số hữu hạn điểm của màn hình,

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

khi nhìn màn hình ở một khoảng cách nào đó, chúng ta sẽ vẫn có cảm giác hình ảnh là mịn.

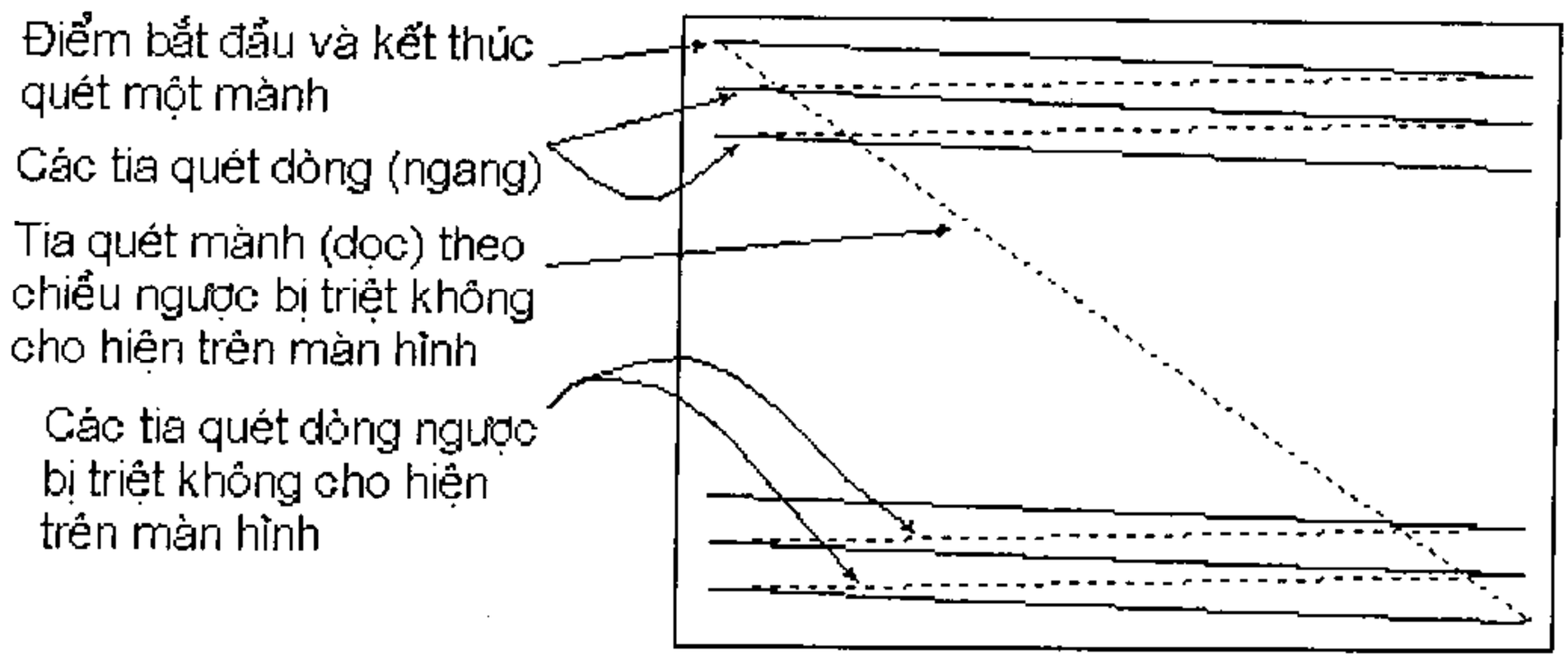
- **Hiện tượng lưu ảnh trên võng mạc**

Khi một hình ảnh hiện rồi lại tắt với tần số lớn hơn 25 lần/giây, mắt người không nhận ra được sự nhấp nháy đó mà có cảm giác hình ảnh tồn tại liên tục. Đó là hiện tượng lưu ảnh trên võng mạc.

Dựa trên độ phân giải hữu hạn và hiện tượng lưu ảnh trên võng mạc của mắt người, người ta đã xây dựng nên phương pháp quét mành để hiển thị thông tin tĩnh và động trên màn hình.

8.3.2 Phương pháp quét mành

Hình ảnh trên màn hình được chia thành một số hữu hạn dòng, tập hợp các dòng tạo nên một hình được gọi là một mành. Trên mỗi dòng của mành chỉ cần hiển thị một số hữu hạn điểm, mỗi điểm được gọi là một **pixel** (Picture element). Như vậy pixel là đơn vị nhỏ nhất của hình ảnh, nó có kích thước hữu hạn.



Hình 8-08 Nguyên lý quét mành

Để hiển thị hình ảnh chúng ta sẽ hiển thị lần lượt từng dòng một, sau khi đã hiển thị xong tất cả các dòng của một mành chúng ta lại lặp lại quá trình trên. Nếu mành được chia thành số dòng đủ lớn thì mắt ta sẽ thấy nó là mịn (không thấy được cấu trúc dòng), nếu tần số hiển thị mành đủ lớn thì ta sẽ cảm thấy hình ảnh tồn tại liên tục chứ không phải là gián đoạn về thời gian. Nếu mỗi mành là hình ảnh của một đối tượng chuyển động ở một thời điểm nhất định thì việc hiển thị các mành theo thứ tự thời gian sẽ tạo nên cảm giác hình ảnh chuyển động.

Số dòng của một mành

Số dòng của một mành được xác định theo công thức sau đây (xem hình 8-09):

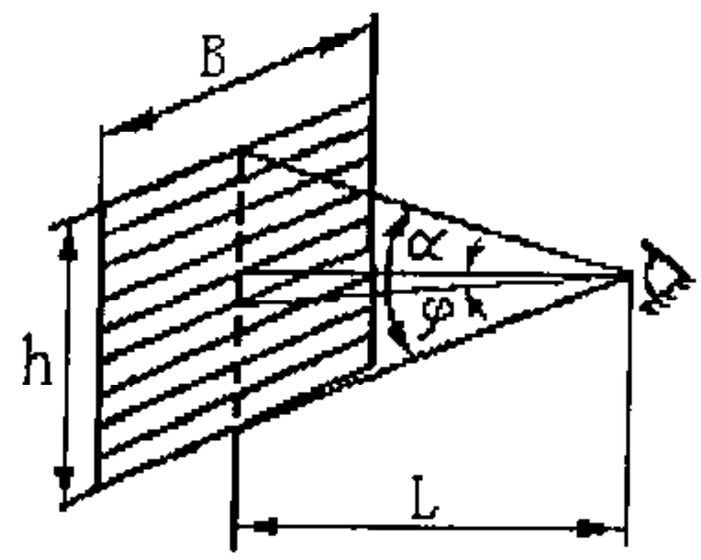
$z = \alpha/\varphi = (2\text{arctg}(h/2L))/\ \varphi$, với:

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

$\varphi = 1'$ là góc phân giải.

h là chiều cao mằn ảnh.

L là khoảng cách từ mắt tới màn ảnh.



Hình 8-09 Xác định số dòng của mành

Khoảng cách từ mắt tới màn ảnh phải lớn hơn 4..6 lần chiều cao h của màn ảnh để mắt người khi quan sát màn ảnh dưới một góc α nhỏ không cần phải quay đầu theo hình ảnh chuyển động mà nó vẫn nằm trong miền thu nhận nhạy của mắt.

Tỉ lệ kích thước màn hình k=B/h thường được chọn bằng 4/3 căn cứ vào việc khi nhìn một hình ảnh dưới góc 8^0 theo chiều ngang và 6^0 theo chiều đứng thì hình ảnh hoàn toàn nằm trong miền thu nhận nhạy của mắt (điểm vàng).

Như vậy z = 2arctg[h/(2(4..6))h]/(1/60) = 600..840 dòng.

8.3.2.1 Nguyên lý cấu tạo và hoạt động của CRT

Đặc điểm của chất huỳnh quang

Những chất khi ta bắn một chùm tia điện tử vào, thì bức xạ ánh sáng có bước sóng nhất định - quyết định màu sắc phát ra được gọi là chất huỳnh quang. Sau khi ngừng tác động của chùm tia điện tử chất huỳnh quang cũng ngừng phát ra ánh sáng sau một khoảng thời gian nhất định, khoảng thời gian này tùy thuộc vào từng chất huỳnh quang cụ thể.

Người ta chọn những chất huỳnh quang có hiệu suất phát quang cao, ánh sáng phát ra phù hợp với thị giác của con người để chế tạo CRT.

Đèn hình (xem hình 8-10)

Đèn hình là một chuông thuỷ tinh được rút chân không, bên trong có các bộ phận để thực hiện chức năng của một "súng điện tử" điều khiển được, có thể "bắn" lên trên toàn màn hình, theo từng dòng của mành. Cường độ tia điện tử của súng được điều chế bởi tín hiệu hình ảnh. Đáy chuông thuỷ tinh chính là màn hình mà chúng ta cần hiển thị thông tin trên đó. Mặt trong của màn hình được phủ một lớp chất huỳnh quang. Bộ phận điều khiển tia điện tử quét trên màn hình thường nằm bên ngoài đèn hình, đó là hệ thống lái tia.

- **Súng điện tử**

Súng điện tử bao gồm một số bộ phận bên trong đèn hình, có nhiệm vụ tạo ra một chùm tia điện tử mảnh, chuyển động đủ nhanh bay lên đập vào một điểm của lớp huỳnh quang trên màn hình, làm cho điểm đó phát sáng. Sau đây chúng ta xem xét một số bộ phận của súng điện tử.

1. Ca-tốt: ca-tốt của CRT khi được đốt nóng sẽ phát xạ ra một đám mây điện tử.

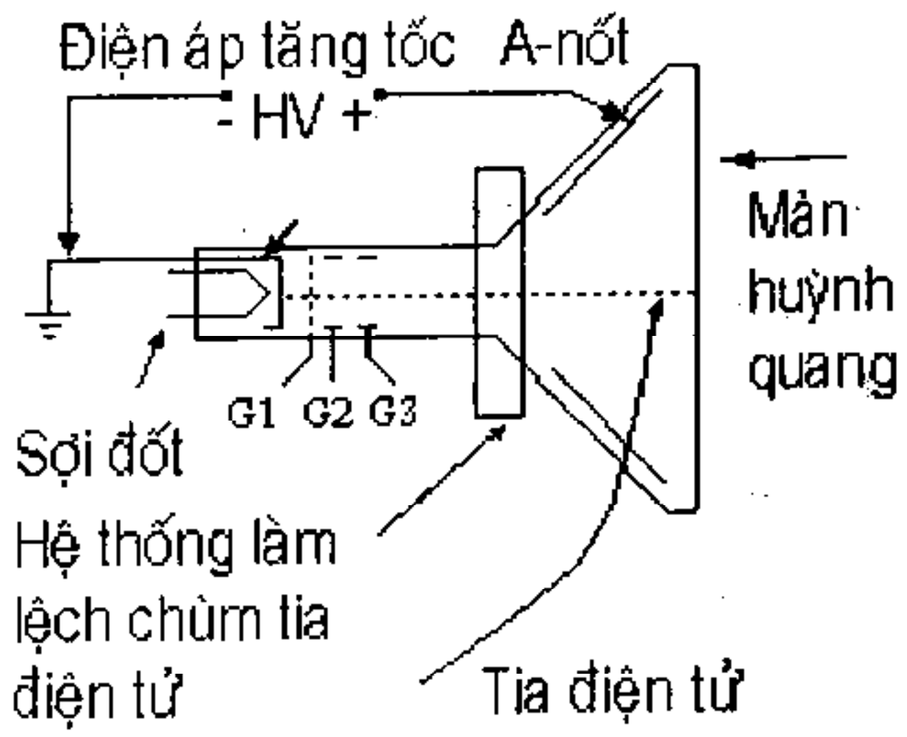
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

2. Các điện cực gia tốc cho chùm tia điện tử: Điện cực chính để gia tốc chùm tia điện tử nằm ở phía màn hình, đó là a-nốt, điện áp của nó rất cao, thường từ 14000V đến 25000V, tuỳ thuộc kích thước CRT. Đám mây điện tử do ca-tốt phát ra bị hút về phía a-nốt của CRT nhưng không bị hút ngay vào a-nốt này mà đập vào lớp huỳnh quang ở mặt trong của màn hình. Điều đó thực hiện được là nhờ a-nốt có hình dạng đặc biệt - hình nón cụt, với đáy lớn là màn hình và chùm tia điện tử chuyển động nhanh dần, khi tới gần màn hình đã có tốc độ rất lớn.

3. Hệ thống các điện cực hội tụ chùm tia điện tử (G2 và G3 trên hình vẽ): Trên đường chuyển động của tia điện tử gần ca-tốt, người ta bố trí các điện cực có hình dáng phù hợp và đặt các điện áp thích hợp, tạo nên một "thấu kính điện tử", có tác dụng làm cho chùm tia điện tử bị hội tụ lại thành một tia rất mảnh. Ngoài tác dụng hội tụ, các điện cực này cũng thường có chức năng gia tốc cho chùm tia điện tử.

- **Hệ thống lái tia điện tử**

Việc lái tia điện tử có thể thực hiện bằng điện trường hoặc từ trường. Trong các CRT dùng cho máy tính nói chung đều thực hiện lái tia điện tử bằng từ trường.



Hình 8-10 CRT với hệ thống lái tia bằng từ trường

Trên hình 8-10 là một CRT có bộ phận lái tia điện tử bằng từ trường. Gần cổ đèn hình người ta đặt 2 cặp cuộn dây song song với nhau và cho các dòng điện có dạng biến thiên phù hợp và tần số nhất định chạy qua, dòng điện sẽ tạo ra từ trường biến thiên theo phương trùng với trục của cuộn dây. Trong lòng đèn hình có 2 vector từ trường theo 2 phương nằm ngang và thẳng đứng, chúng tác động lên dòng điện tử, làm nó lệch hướng theo quy luật của dòng điện trong các cặp cuộn dây. Tia điện tử sẽ quét trên mặt đèn hình theo các dòng, tập hợp các dòng kẻ trên màn hình được gọi là một mành (Raster).

- **Điều chế dòng tia điện tử**

Khi súng điện tử quét lần lượt các dòng của mành, nếu làm thay đổi cường độ của chùm tia điện tử theo quy luật thay đổi của tín hiệu hình ảnh nơi phát đi, thì sẽ tái tạo lại hình ảnh trên màn hình. Việc điều khiển dòng điện tử theo quy luật của hình ảnh cần hiển thị được gọi là điều chế dòng điện tử.

Trên hình 8-10 điện cực G1 là lưới điều khiển, nếu đặt giữa G1 và ca-tốt một điện áp thay đổi theo quy luật thay đổi của tín hiệu hình ảnh thì dòng điện tử của súng điện tử sẽ bị điều chế theo quy luật của tín hiệu.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

8.3.2.2 Nguyên lý tổng hợp màu sắc và cấu tạo đèn hình màu

Phổ tần số của ánh sáng trắng

Khi phân tích chùm ánh sáng trắng (ánh sáng mặt trời) người ta thấy rằng nó là tổ hợp của các bước sóng có giá trị thay đổi liên tục, có thể nêu tên các màu chính như sau: đỏ, da cam, vàng, lục, lam, chàm, tím. Nếu đem các màu thành phần tổ hợp lại thì ta lại thu được chùm ánh sáng trắng.

Ba màu cơ bản: Red, Green, Blue

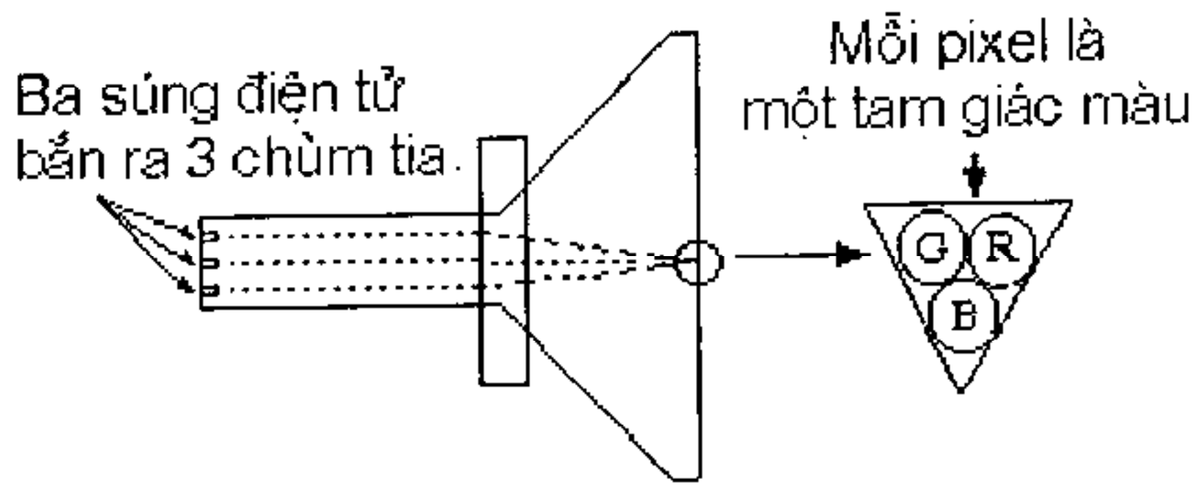
Người ta nghiên cứu và thấy rằng nếu đem 3 chùm tia đỏ (Red), lục (Green) và lam (Blue) trộn lại theo các tỷ lệ nhất định, thì có thể tạo lại cảm giác màu của tất cả các màu có trong thiên nhiên, chính vì vậy 3 màu đó được gọi là 3 màu cơ bản.

Về mặt kỹ thuật, các hãng sản xuất đèn hình có thể chọn các bộ ba màu cơ bản có bước sóng không hoàn toàn giống nhau. Đây là một trong các nguyên nhân làm cho các CRT màu của các hãng có chất lượng màu khác nhau ít nhiều.

Việc trộn màu được thực hiện trên mặt CRT màu, do khả năng phân giải của mắt ≤1', người ta bố trí 3 hạt màu cơ bản gần nhau, chúng tạo thành một pixel, mắt ta sẽ cảm nhận thấy một màu là tổ hợp của 3 thành phần này, như vậy sự tổng hợp màu sắc xảy ra trong mắt người.

Bộ 3 màu (Triad màu) và các kiểu màn hình

Thời kỳ đầu của sự phát triển công nghệ CRT màu, người ta bố trí 3 hạt màu trên đỉnh của một tam giác đều, vì vậy bộ ba này còn có tên gọi ***tam giác màu***. Cách bố trí này tạo cảm giác màu tổng hợp tốt nhất, nhưng lại kèm theo một số nhược điểm kỹ thuật như hiệu suất đèn hình thấp (hệ số thấu quang của mặt nạ thấp, chỉ cỡ 20%).



Hình 8-11 Nguyên lý hiển thị hình ảnh màu

Hãng Sony đưa ra giải pháp bố trí 3 hạt màu nằm trên đường thẳng nằm ngang, toàn bộ màn hình như tạo bởi 1 ma trận gồm các hàng và các cột, giữa chúng có các dải phân cách, tránh hiện tượng nhoè màu. Trong công nghệ ***Trinitron*** này sự thụ cảm màu tổ hợp không tốt bằng trong công nghệ trên, nhưng hệ số thấu quang của mặt nạ bên trong CRT lớn hơn nhiều (> 60 %) do đó dòng cao áp giảm, ngoài ra việc hội tụ động cũng đơn giản hơn nhiều.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Công nghệ ***Trinitron*** dẫn đến giảm giá thành của CRT nhiều đồng thời lại tăng được độ tin cậy, do đó ngày nay đèn hình của các TV màu và của máy tính đều được chế tạo theo công nghệ này.

Khả năng hiển thị màu sắc và độ phân giải của các loại màn hình

Độ phân giải màu phụ thuộc vào mật độ Triad màu, đồng thời còn phụ thuộc vào dải tần số truyền qua của kênh truyền tín hiệu hình ảnh.

Số tổ hợp màu có thể nhận được phụ thuộc vào khả năng điều khiển cường độ của từng màu cơ bản, ngoài ra để có thể hiển thị được nhiều màu hơn, cần phải dành nhiều bộ nhớ hơn để lưu trữ thông tin về thuộc tính của các pixel. Chúng ta thử làm một vài con tính:

- Nếu mỗi màu cơ bản chỉ có 2 mức sáng và tối thì số tổ hợp màu khác nhau có thể có là (số giá trị có thể có của 1 số nhị phân 3 chữ số):

  $2^3 = 8$ màu (kể cả 'màu' đen và 'màu' trắng)

- Nếu mỗi màu cơ bản có 3 mức: mức sáng 1, mức sáng 2 và mức tối thì số tổ hợp màu khác nhau có thể có là:

  $2^6 - 3*2^4 = 64 - 48 = 16$ màu (mỗi màu có mức sáng thể hiện bởi 2 bit, nhưng không sử dụng trường hợp cả 2 bit bằng 1)

- Nếu mỗi màu cơ bản có 3 mức sáng và 1 mức tối thì số tổ hợp màu khác nhau có thể có là:

  $2^6 = 64$ màu (mỗi màu có mức sáng thể hiện bởi 2 bit, có thể nhận các giá trị 00,01,10,11)

- Nếu mỗi màu cơ bản có 4 mức sáng và 1 mức tối thì số tổ hợp màu khác nhau có thể có là:

  $2^6 + 3*2^6 = 4*64 = 256$ màu

  (mỗi màu có mức sáng thể hiện bởi 3 bit, có thể nhận các giá trị 000, 001, 010, 011, 100)

8.3.3 Làm tươi hình ảnh bằng phương pháp DMA

Việc làm tươi hình ảnh

Nếu chỉ hiển thị một mành của hình ảnh, mắt ta sẽ thấy hình ảnh đó sẽ loé lên rồi mất đi. Muốn nhận được hình ảnh tồn tại liên tục, dù là ảnh tĩnh hay là ảnh động, thì khi cảm giác về một mành còn chưa mất đi trong hệ thống thần kinh thị giác, cần phải lặp lại việc hiển thị mành. Việc này được gọi là làm tươi hình ảnh. Tần số lặp lại, hay còn gọi là tần số làm tươi cần phải đủ lớn để mắt không cảm thấy hình ảnh bị lập loè. Người ta thường chọn tần số làm tươi không thấp hơn 50 mành/giây.

Sử dụng phương pháp DMA để làm tươi hình ảnh

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Quá trình làm tươi hình ảnh trên màn hình thực chất là quá trình liên tục chuyển nội dung của bộ nhớ hiển thị lên màn hình. Việc này đòi hỏi phải vận chuyển một lượng thông tin lớn, nhất là trong hiển thị đồ hoạ chất lượng cao, với các chu kỳ xác định. Những người thiết kế hệ thống máy tính không muốn CPU phải làm việc này, họ sử dụng một đơn vị chức năng riêng, chuyên trách việc vận chuyển dữ liệu. Phương pháp này được gọi là phương pháp truy cập trực tiếp bộ nhớ DMA (Direct Memory Access).

Làm tươi bằng phương pháp DMA đòi hỏi hệ thống phải có đơn vị điều khiển DMAC (DMA Controller), một cặp bộ đệm dòng và bộ nhớ hiển thị.

DMAC thực chất là một bộ vi xử lý chuyên dụng, nó đọc nội dung của bộ nhớ hiển thị rồi chuyển ra một cổng dẫn tới mạch điều chế dòng tia điện tử. Người ta áp dụng một số phương pháp đặc biệt để việc DMAC đọc bộ nhớ hiển thị không cản trở việc CPU đọc/ghi vào bộ nhớ này. Nhờ đó đối với các chương trình chạy trong hệ thống, hoạt động làm tươi hình ảnh là trong suốt, nghĩa là các chương trình (và người lập trình) không nhận thấy hoạt động này.

Cặp bộ đệm dòng thường là một phần của đơn vị điều khiển màn hình CRTC (CRT Controller), mỗi bộ đệm dòng chứa được lượng thông tin của một dòng của mành. Cặp bộ đệm dòng được sử dụng trong quá trình chuyển số liệu bằng phương pháp DMA giữa bộ nhớ hiển thị với bộ phận hiển thị.

Bộ nhớ hiển thị thường là một phần của bộ nhớ trong của hệ vi xử lý. Kích thước cần phải có của bộ nhớ hiển thị phụ thuộc vào tổng số pixel trên một mành và số màu có thể có của mỗi pixel. Số màu càng nhiều thì lượng bộ nhớ hiển thị để lưu trữ thuộc tính của pixel càng lớn.

Trong các máy IBM PC/AT và tương thích đều có card màn hình, trên đó có CRTC, toàn bộ hoặc một phần bộ nhớ hiển thị.

Phương pháp làm tươi sử dụng DMA được minh hoạ trên hình 8-12.

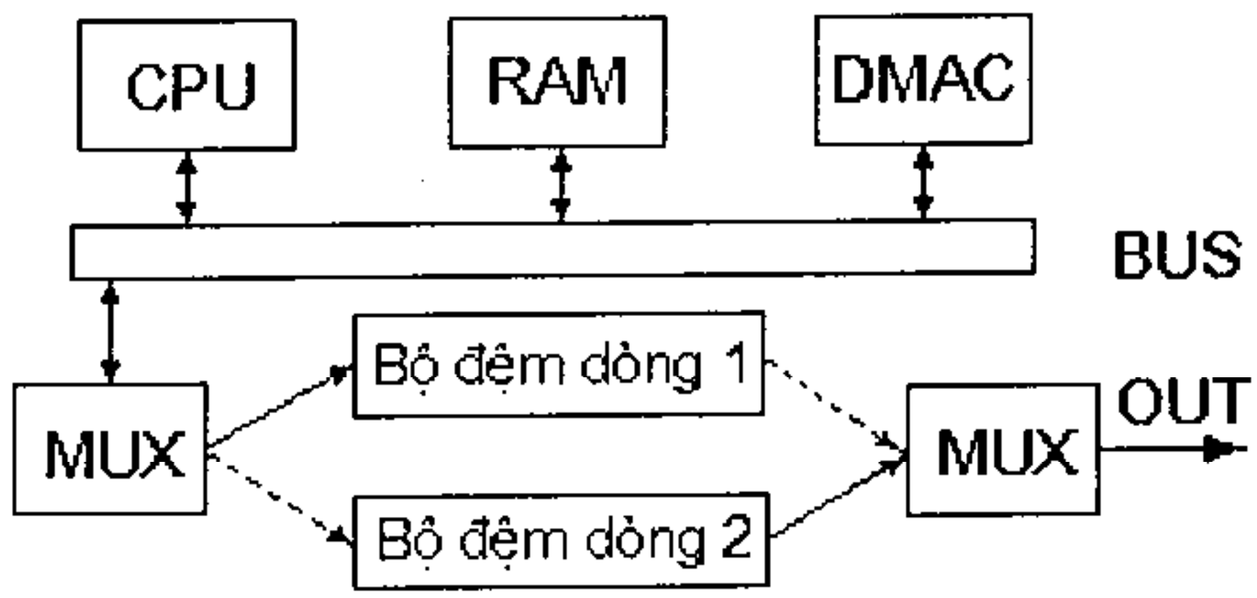
- Khi DMAC nạp số liệu vào bộ đệm dòng số 1, bộ đệm dòng số 2 đã được nạp đầy trước đó lần lượt đẩy nội dung của mình ra cho bộ tạo chữ (Character Generator). Thời gian để bộ đệm dòng đẩy hết nội dung ra ngoài thường lớn hơn thời gian nạp đầy bộ đệm vì tốc độ chuyển DMA rất cao còn tốc độ đẩy từ bộ đệm ra lại phụ thuộc vào tốc độ quét của tia điện tử trên màn hình.

- Khi bộ đệm dòng thứ hai đã đẩy hết nội dung ra ngoài thì bộ đệm dòng số 1 chắc chắn đã nạp đầy và quá trình làm tươi được tiếp tục với sự đổi hướng của 2 bộ phân kênh và dồn kênh (ghi là MUX trên hình 8-12):

  - Số liệu từ bộ đệm dòng 1 được đẩy ra ngoài trong lúc đó DMAC nạp số liệu vào bộ đệm dòng 2 từ bộ nhớ hiển thị.

  - Khi bộ đệm dòng 1 đã đẩy hết số liệu, bộ MUX lại đổi hướng và quá trình làm tươi cứ thế được thực hiện một cách liên tục.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 8-12 Làm tươi hình ảnh theo phương pháp DMA

8.3.4 Một số chuẩn của màn hình máy tính

Chuẩn Hercule (đơn sắc)

Số màu: 1

Ma trận chữ: 9 x 14

Độ phân giải màn hình: 720 x 348

Chuẩn CGA (Color Graphics Adapter)

Số màu: 16 (có thể chọn 4 trong 16)

Ma trận chữ: 8 x 8

Độ phân giải màn hình: 640 x 200 trong chế độ đơn sắc
320 x 200 trong chế độ có 4 màu

Chuẩn EGA (Enhanced Graphics Adapter)

Số màu: 64 (có thể chọn 16 trong 64)

Ma trận chữ: 8 x 14

Độ phân giải màn hình: 640 x 350 trong chế độ 16 màu

Có bộ nhớ RAM để tạo font chữ.

Chuẩn VGA (Video Graphics Array)

Số màu: 256 (có thể chọn 64 trong 256)

Ma trận chữ: 9 x 14

Độ phân giải màn hình: 640 x 480 (x 16/256 màu)
(Số màu tối đa tuỳ thuộc dung lượng video RAM)

Có bộ nhớ RAM để tạo Font chữ.

8.3.5 So sánh một số đặc điểm của màn hình máy tính và TV

Về dải tần số truyền qua và tần số quét dòng

Người ta đã nghiên cứu và thấy rằng độ phân giải theo chiều ngang và dọc của mắt người là như nhau, như vậy nếu một mành có z dòng, tức là theo chiều đứng có

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

z điểm thì theo chiều nằm ngang có kz điểm, với k=B/h. Một mành sẽ có kzz điểm. Nếu một giây truyền f_r mành, thì cần truyền một lượng thông tin ứng với f_r*k*z*z điểm.

Giả sử nội dung hình ảnh đen trắng là phức tạp nhất, ứng với trạng thái các điểm lân cận trên một dòng có trạng thái sáng tối trái ngược nhau, thì cứ 2 điểm sẽ ứng với 1 xung tín hiệu điện hay là một chu kỳ. Như vậy để hiển thị hình ảnh, tín hiệu điện chuyển ra màn hình sẽ có tần số tối đa là:

$f_{max} = f_r * k * z * z /2$. Tần số nhỏ nhất $f_{min}= 0$ ứng với trường hợp cả màn hình có cùng một mức tín hiệu.

Tần số quét dòng (số dòng quét trong 1 giây) $f_l = f_r*z$.

(Các tính toán trên là gần đúng, chưa xét thời gian quét ngược dòng và mành).

Một số thí dụ cụ thể:

- Theo tiêu chuẩn truyền hình của Liên Xô cũ (OIRT), mỗi mành có 625 dòng, f_r=50. Giả sử tỉ lệ kích thước màn hình là 4/3 và sử dụng phương pháp quét xen kẽ các bán mành chẵn và lẻ. Thay k = 4/3, z = 625 ta có $2f_{max}$= 50 * 4/3 * 625 * 625/2 = 13 020 833hz, vậy $f_{max}\approx$6.5MHz. Như vậy dải tần số truyền qua của kênh Video là 6,5 MHz, tần số quét dòng f_l = 50*625/2 = 15625hz.

- Theo tiêu chuẩn truyền hình của mỹ (FCC), mỗi mành có 455 dòng, mỗi giây truyền 60 bán mành. Tính tương tự trường hợp trên, chúng ta sẽ nhận được dải tần số của kênh video là 4.5 MHz; tần số quét dòng f_l = 60*455/2 = 13650hz

- Màn hình tiêu chuẩn CGA80 (640 x 200), có dải tần số truyền qua:

  $f_{max}\approx$ 640 x 200 x 60/2 = 3 840 000 hz = 3.84 MHz

  Tần số quét dòng là: f_l = 60*200 = 12 000Hz (nếu không quét xen kẽ). Dải tần này thấp hơn của Tivi và tần số quét dòng cũng xấp xỉ bằng tần số quét dòng của Tivi (60*455/2=13650) do đó có thể dùng màn hình Tivi làm phương tiện hiển thị cho máy tính (tất nhiên là trong mode CGA).

- Màn hình tiêu chuẩn Hercule (720 x 348)

  Dải tần số truyền qua tính gần đúng như sau:

  $f_{max}\approx$ 720 x 348 x 60/2 = 7 516 000 hz = 7.516 MHz

  Tần số quét dòng là: $f_l \approx$ 60 x 348 = 20880 Hz

  Ta thấy dải tần số truyền qua cũng như tần số quét dòng đều cao hơn của Tivi, do đó không thể sử dụng màn hình Tivi làm thiết bị hiển thị cho máy tính theo tiêu chuẩn này.

Việc quét xen kẽ

Cần phải tạo lại trên 25 hình/1 giây để tránh hiện tượng hình ảnh bị nhấp nháy. Giải pháp quét xen kẽ mành chẵn (mành gồm toàn các dòng chẵn), mành lẻ (mành

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

gồm toàn các dòng lẻ) làm giảm số mành cần phải quét trong 1 giây chỉ còn một nửa, do đó làm giảm dải thông (bandwidth) cũng như tần số quét dòng. Có một số loại màn hình sử dụng quét xen kẽ (interlaced) nhằm hạ giá thành, tuy vậy chất lượng sẽ thấp hơn những loại màn hình không sử dụng phương pháp quét xen kẽ (none-interlaced).

Đặc điểm sử dụng

Với màn hình máy tính người quan sát thường ngồi khá gần, khoảng cách từ mắt tới màn hình thường không lớn hơn chiều cao màn hình từ 4..6 lần như chúng ta đã phân tích ở trên, do đó màn hình máy tính thường được làm với kích thước nhỏ (8", 12", 14") đồng thời số dòng / mành cũng lớn hơn trên màn hình TV.

Các tín hiệu video số của máy tính thường có cường độ thay đổi đột ngột ngay cả giữa 2 dòng kế tiếp nên gây ra sự thay đổi cường độ sáng nhảy bậc trên màn hình, nếu sử dụng quét xen kẽ dễ gây cảm giác hình ảnh bị nhấp nháy. Trong khi đó các tín hiệu độ xám trong truyền hình thường thay đổi từ từ theo nhiều mức khác nhau, sự cảm nhận được trung hoà bởi thị giác, hình ảnh cảm nhận thường mịn hơn.

8.3.6 Hiển thị thông tin

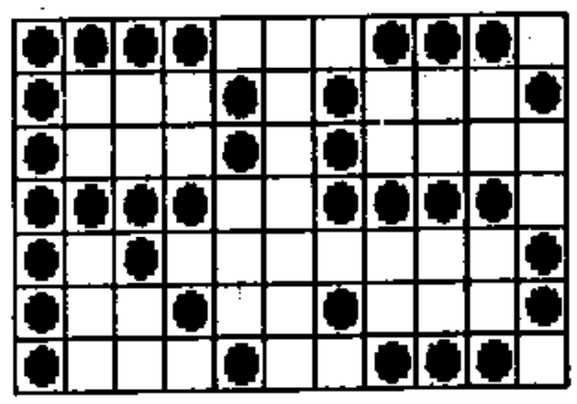
8.3.6.1 Hiển thị thông tin chữ số và nửa đồ hoạ

Mỗi kí tự và mẫu đồ hoạ đơn giản có một mã ASCII tương ứng, hình dạng của nó được thể hiện trong khuôn khổ của một ma trận điểm, ma trận điểm có thể có một số giá trị kích thước chuẩn sau: 5x7, 8x8, 7x9, 8x14, 9x14.v.v.

Các ma trận này thường được xây dựng sẵn và đặt trong bộ nhớ của máy tính. Người sử dụng có thể lựa chọn các bộ ma trận ký tự khác nhau và cũng có thể xây dựng lấy cho mình các bộ ma trận ký tự. Kích thước ma trận càng lớn thì độ phân giải càng cao, nhưng dẫn tới số bit cần thiết cho một ma trận (một kí tự hay một kí hiệu) càng lớn, vùng nhớ để lưu trữ cũng cần lớn hơn; hơn nữa còn kéo theo dải tần số truyền qua của thiết bị màn hình phải tăng theo.

- **Cách lưu trữ ma trận ký tự trong bộ nhớ**

Chúng ta sẽ lấy ma trận 5x7, như trên hình 8-13 làm thí dụ, đây là ma trận ký tự đơn giản nhất, kinh tế nhất, đã từng được sử dụng trong thực tế.



Hình 8-13 Hiển thị ký tự bằng ma trận điểm 5x7

Giả sử các điểm của ma trận ký tự chỉ có trạng thái sáng và tối, như vậy mỗi điểm có thể được biểu diễn bằng 5 bit, điểm sáng biểu diễn bằng giá trị 1, còn tối bằng 0. Ký tự R trên hình 8-13 sẽ được biểu diễn bằng 7 nhóm 5 bit như sau:

11110, 10001, 10001, 11110, 10100, 10010, 10001

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Nếu mỗi địa chỉ nhớ ứng với 1 byte (họ IBM PC là như vậy), người ta sẽ lấy nhóm 5 bit đầu tiên (11110) và 3 bit tiếp theo của của nhóm 5 bit thứ hai (100) ghi vào 1 byte. Hai bit còn lại của nhóm 5 bit thứ hai (01), cả nhóm 5 bit thứ ba (10001) và bit đầu tiên của nhóm 5 bit thứ tư (1), cộng lại thành 8 bit được ghi vào byte tiếp theo. Quá trình này cứ tiếp diễn, hết ký tự R sang ký tự S .v.v. mỗi ma trận ký tự cần 35 bit bộ nhớ. Toàn bộ các ma trận ký tự của bảng mã ASCII được lưu trữ như vậy.

- **Cách hiển thị**

Trên dòng quét ngang đầu tiên trên màn ảnh, tia điện tử hiển thị tất cả các hàng thứ nhất (mỗi hàng có 5 điểm) của các kí tự thuộc dòng chữ trên cùng của màn hình. Lần quét ngang tiếp theo, hiển thị các hàng thứ hai trong ma trận điểm của các kí tự cùng dòng, cứ như thế, tia điện tử quét tất cả các hàng của ma trận điểm của từng dòng kí tự trên màn hình.

Giữa hai dòng kí tự thường có vài dòng quét phân cách - không có tín hiệu của các kí tự để các dòng chữ không gần nhau quá, giúp chúng ta dễ đọc.

Khi quét hết các dòng thứ 7 của các ma trận ký tự ứng với dòng chữ dưới cùng của màn hình tia điện tử đã hoàn thành việc quét một mành, nó sẽ được điều khiển chuyển lên góc trên bên trái của màn hình để lặp lại quá trình quét.

- **Đọc ma trận ký tự gửi tới bộ phận điều chế dòng điện tử**

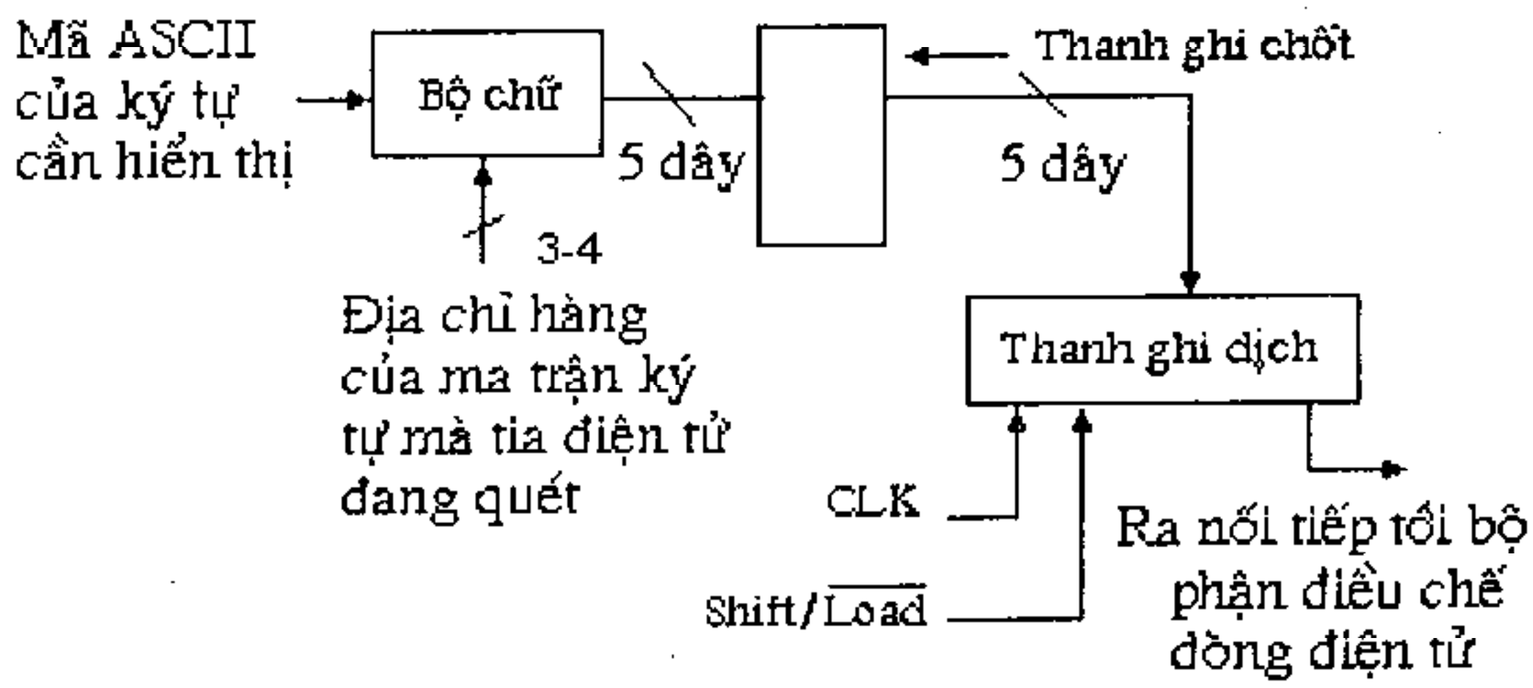
Theo cách lưu trữ ma trận ký tự như đã nghiên cứu ở trên, chúng ta thấy rằng, nếu biết địa chỉ đầu của vùng nhớ chứa ma trận ký tự và kích thước của ma trận thì khi nhận được mã ASCII của một ký tự cần hiển thị nào đó có thể tính ra địa chỉ byte đầu tiên của nhóm byte chứa ma trận ký tự đó. Nếu biết được tia điện tử đang quét dòng thứ bao nhiêu của màn hình thì có thể trích ra nhóm bit cần thiết ứng với một hàng của ma trận ký tự cần hiển thị. Nhóm bit này cần được chuyển từ dạng song song sang nối tiếp để đưa tới bộ phận điều chế tia điện tử.

Quá trình trên được thực hiện theo sơ đồ trên hình 8-14 và được giải thích kỹ hơn dưới đây.

- Bộ chữ (character generator) là một đơn vị chức năng, trong đó có ROM, RAM để chứa các ma trận ký tự của bảng mã ASCII, cũng như dành sẵn chỗ cho người lập trình có thể tạo ra các bộ ký tự của họ. Ngoài ra bộ chữ còn có các mạch giải mã cần thiết phục vụ cho việc đọc thông tin về các hàng của các ký tự .v.v.
- Tín hiệu mã ASCII chọn một kí tự trong bộ chữ
- Tín hiệu địa chỉ hàng (3-4 bit) chọn một hàng trong ma trận của kí tự đã chọn trên. Bộ chữ đưa ra tín hiệu 5 bit (của một hàng), 5 bit này được đưa song song vào thanh ghi chốt 8 bit, 3 bit còn lại của thanh ghi chốt được nạp mức logic 0 để làm khoảng ngăn cách giữa 2 ký tự trong cùng một dòng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Tín hiệu Shift/Load nếu có giá trị tích cực (high) sẽ điều khiển việc đưa dữ liệu trong thanh ghi dịch ra đầu ra nối tiếp từng bit một theo nhịp của tín hiệu CLK. Nếu Shift/Load có giá trị không tích cực (low) sẽ điều khiển việc nhận dữ liệu từ thanh ghi chốt vào thanh ghi dịch một cách song song.
- Tín hiệu đồng hồ nhịp CLK từ bộ dao động đi đến sẽ đẩy nội dung thanh ghi dịch lần lượt ra đầu ra nối tiếp, rồi đưa ra thiết bị CRT. Trong màn hình đơn sắc, ở chế độ chỉ có 2 mức sáng và tối mỗi bit này sẽ điều khiển độ sáng của một điểm trên dòng đang được quét.



Hình 8-14 Đọc ma trận ký tự gửi tới bộ phận điều chế dòng điện tử

8.3.6.2 Hiển thị thông tin đồ hoạ

Hiển thị thông tin dưới dạng đồ hoạ có ý nghĩa rất lớn nhờ tính trực giác của nó, hình ảnh đồ hoạ có thể cho người sử dụng nắm được các thông tin cơ bản nhanh hơn những dòng giải thích bằng văn bản đầy cả màn hình. Trong một số lĩnh vực, đồ hoạ là một yếu tố không thể thiếu được

Trước đây, khả năng đồ hoạ thuộc về lớp máy Mini hoặc các máy mạnh hơn, vì đồ hoạ cần phải có bộ nhớ lớn, số lượng tính toán lớn và phần cứng phức tạp. Ngày nay do giá thành của phần cứng đã giảm đi rất nhiều lần nên hầu hết các máy vi tính cũng có khả năng đồ hoạ.

Người ta đã sản xuất ra các chip LSI làm nhiệm vụ điều khiển hiển thị đồ hoạ, gọi tắt là GDC (Graphics Display Controller), các GDC đều làm việc theo nguyên lý quét mành, chúng không trực tiếp điều khiển CRT mà điều khiển bộ nhớ hiển thị. Mỗi điểm ảnh (pixel) ứng với 1 bit trong trường hợp hiển thị đen trắng, hoặc ứng với 1 nhóm bit trong trường hợp hiển thị màu hoặc đen trắng với các mức đậm nhạt khác nhau.

Quá trình hiển thị đồ hoạ, chương trình con vẽ điểm

Để tìm hiểu quá trình này, ta xem xét chức năng cơ sở nhất: chức năng vẽ điểm, tất cả các khả năng đồ hoạ khác đều dựa trên cơ sở vẽ điểm bằng màu mong muốn tại một toạ độ cần thiết. Thời gian thực hiện chương trình con vẽ điểm là một yếu tố

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

ảnh hưởng rất lớn đến tốc độ của cả hệ đồ hoạ. Trong hệ hiển thị đồ hoạ kiểu quét mành, một điểm trên màn hình ứng với 1 hay một nhóm bit của bộ nhớ hiển thị, do đó chương trình con vẽ điểm phải chuyển đổi toạ độ điểm (x,y) trên màn hình sang địa chỉ bộ nhớ. Thuật toán chuyển đổi hệ toạ độ phụ thuộc vào GDC.

Thí dụ loại mạch điều khiển MC 6847, khi làm việc ở chế độ có độ phân giải cao nhất: 256x192 công thức chuyển đổi địa chỉ sẽ là:

địa chỉ = ((x+xoff)/8 - (y-yoff)*32) + Screen + SCRoff, trong đó

- Screen: địa chỉ đầu của bộ nhớ hiển thị
- xoff và yoff: là số bù để cho gốc toạ độ nằm giữa màn hình (với MC6847, ở chế độ phân giải cao nhất, xoff=128; yoff=96)
- SCRoff (=2*yoff*32) : độ dài của của vùng nhớ màn hình tính theo đơn vị byte trừ đi 32 (thực chất là địa chỉ của byte đầu tiên của dòng cuối cùng trên màn hình)
- Giá trị theo hoành độ (x+xoff) phải chia nguyên cho 8 vì mỗi điểm ứng với 1 bit, cho nên sẽ có các nhóm 8 điểm trên mỗi dòng cùng chung nhau 1 byte, tức là có cùng một địa chỉ bộ nhớ.
- Giá trị theo tung độ (y-yoff) phải nhân với 32 vì mỗi đường quét ngang gồm 256 điểm ứng với 32 byte. Việc quét thêm 1 dòng, tức là tung độ y tăng thêm 1, cũng có nghĩa là đã duyệt thêm được 32 byte trong bộ nhớ hiển thị.

Sau khi tính được địa chỉ của byte chứa bit ứng với điểm cần vẽ trên màn hình, ta cần tiếp tục tính vị trí của bit trong byte, trong trường hợp này địa chỉ bit chính là phần dư của phép chia nguyên (x+xoff)/8 theo công thức trên. Từ địa chỉ bit tính được, ta lập mặt nạ bit để che tất cả các bit còn lại trong byte, sau đó ghi vào byte có địa chỉ tính được theo công thức trên thông qua mặt nạ bit.

(Thuật toán vừa trình bày được cụ thể hoá bằng một chương trình viết bằng ngôn ngữ assembly, trong giáo trình "Hợp ngữ và lập trình hệ thống" của cùng tác giả.)

8.4 ĐĨA TỪ VÀ ĐĨA QUANG

8.4.1 Giới thiệu về các loại thiết bị nhớ phụ

Bộ nhớ phụ (auxiliary memory, secondary memory) của máy tính dùng để lưu trữ các chương trình và dữ liệu không được sử dụng ngay trong quá trình hoạt động, nội dung của bộ nhớ phụ không bị mất đi khi tắt máy. Trong lịch sử phát triển của máy tính, bộ nhớ phụ luôn đóng vai trò quan trọng, ngày nay vai trò đó càng trở nên quan trọng hơn, tới mức không thể thiếu được trong sự hoạt động của các máy tính điện tử hiện đại.

Bìa và băng đục lỗ là các thiết bị nhớ phụ ra đời sớm nhất, được sử dụng để nhớ các chương trình và số liệu, chúng có nhược điểm là không có tính thuận nghịch và tốc độ tương đối chậm, mật độ thông tin thấp.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Băng từ ra đời sau này có ưu điểm lớn là có thể đọc/ghi nhiều lần, mật độ thông tin cao, nhưng tốc độ vẫn còn rất chậm so với tốc độ của CPU. Ngày nay băng từ vẫn còn được sử dụng rộng rãi trong các ứng dụng cần lưu trữ lượng thông tin lớn, lâu dài, tần suất truy cập thấp. Có một số chuẩn về băng từ như: cassette, catridge, tape. Các chuẩn này mô tả các đặc tính kỹ thuật như: độ rộng đường ghi thông tin, tốc độ chạy băng, dải tần tín hiệu đọc/ghi .v.v.

Đĩa từ lần đầu tiên được đưa vào sử dụng trong các thiết bị tính toán của hãng IBM vào đầu những năm 1970, ngày nay đã được sử dụng rất phổ biến trong các máy vi tính và đã trở thành một trong các thiết bị ngoại vi chuẩn của các máy micro và mini. Đĩa từ có nhiều ưu điểm so với các thiết bị nhớ phụ ra đời trước nó: dung lượng nhớ lớn, tốc độ trao đổi dữ liệu cao, thời gian truy cập tương đối ngắn, tổ chức đọc/ghi tín hiệu mềm dẻo, giá rẻ, gọn nhẹ, dễ bảo quản, sử dụng.

Đĩa quang hay còn gọi đĩa laser (Compact Disc; Optical Disc) có ưu điểm nổi bật nhất là dung lượng cực lớn, độ tin cậy rất cao. Tuy ra đời sau các thiết bị nhớ phụ trên nhưng đã được nghiên cứu cải tiến liên tục. Đĩa quang nói chung chỉ thua kém đĩa từ (đĩa cứng) ở tốc độ truy cập mà thôi.

Nhìn chung các thiết bị nhớ phụ đều có các ưu và nhược điểm, một số loại ngày nay không còn được sử dụng nữa, một số loại khác vẫn tồn tại song song và phát huy ưu điểm của mình trong những ứng dụng cụ thể. Phương hướng phát triển trong tương lai là nhằm đạt dung lượng, tốc độ và độ tin cậy ngày càng cao hơn, giá rẻ hơn.

8.4.2 Nguyên lý ghi từ, đầu từ

Nguyên lý ghi từ

Trong việc ghi từ, trạng thái đảo ngược từ thông được ghi trên các đường trên băng hay đĩa từ thể hiện cho thông tin. Trên mặt băng hay đĩa từ có phủ một lớp bột vật liệu sắt từ trộn với keo dính. Các hạt từ này có khả năng duy trì từ tính sau khi tác động lên chúng từ trường mang nội dung thông tin cần ghi.

Tính thẩm từ (Permeable): là tính chất có thể cho từ thông đi xuyên qua một cách dễ dàng, nói cách khác, đó là tính chất dẫn từ.

Tính duy trì từ tính (Retentivity), còn gọi là tính bị nhiễm từ, thể hiện khả năng lưu lại từ tính sau khi ngừng tác dụng từ trường ngoài.

Chất sắt từ là những chất có độ thẩm từ cao và tính duy trì từ tính cao.

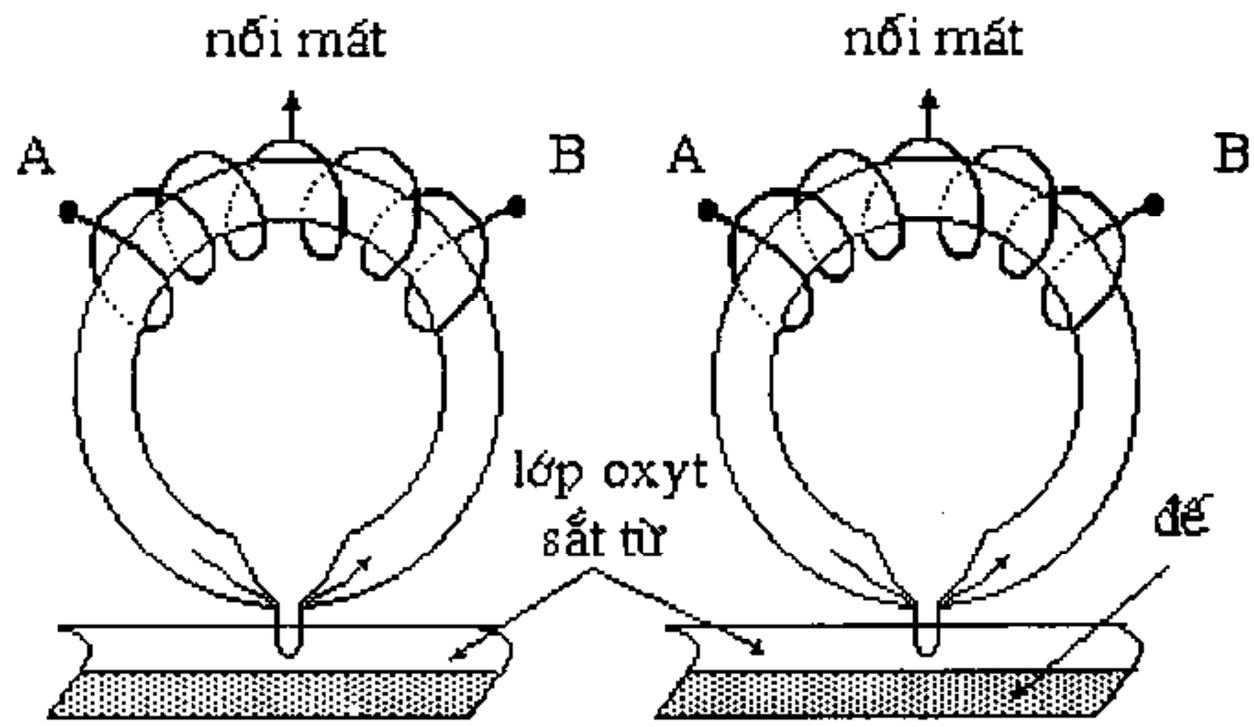
Đầu từ và việc đọc/ghi

Bộ phận then chốt trong việc đọc/ghi là đầu từ. Trong chế độ đọc hay chế độ ghi, có các yêu cầu kỹ thuật khác nhau đối với đầu từ, tuy nhiên để cho đơn giản người ta thường sử dụng một đầu từ làm cả hai nhiệm vụ này, khi đó nó có tên gọi là đầu từ đọc/ghi (read/write head) với các thông số kỹ thuật trung gian.

Nguyên lý cấu tạo đầu từ gần giống nam châm điện, trong đó lõi đầu từ được làm bằng hợp kim có độ thẩm từ cao nhưng không có tính duy trì từ tính. Lõi hình

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

khuyên, có một khe hở nhỏ đồng thời là điểm tiếp xúc với lớp oxyt của băng hay đĩa từ. Quanh lõi có quấn một cuộn dây, thường có điểm giữa nối 'đất' để khử nhiễu.



Hình 8-15 Nguyên lý cấu tạo đầu từ

- **Khi ghi**

Dòng điện chạy trong cuộn dây AB có cường độ tương ứng với các bit thông tin cần ghi, dòng điện này tạo ra một từ trường xác định trong lõi hình khuyên. Qua khe hở, từ thông của từ trường này đi xuyên xuống lớp oxyt sắt từ, 'sắp xếp' (từ hoá) các hạt chất sắt từ của lớp oxyt sắt từ chạy qua khe hở đầu từ theo một hướng nhất định, chỉ phụ thuộc vào chiều của đường sức đó. Sự sắp xếp (hướng từ hoá) các phần tử này chỉ thay đổi khi có sự thay đổi về chiều của từ trường gây ra bởi sự thay đổi chiều của dòng điện chạy trong cuộn dây AB. Còn dòng điện trong cuộn dây AB thay đổi theo quy luật của tín hiệu cần ghi. ***Từ trường dọc theo đường ghi thay đổi theo quy luật của dòng điện mang thông tin đi qua cuộn dây AB.***

- **Khi đọc**

Ngược lại với quá trình ghi, khi đọc thông tin, sự thay đổi chiều sắp xếp các phần tử từ (từ trường) dọc theo đường ghi sẽ tạo nên sự thay đổi chiều của từ trường trong lõi đầu từ, thông qua khe hở đầu từ. Sự thay đổi này sinh ra dòng điện cảm ứng trong cuộn dây AB, dòng điện này mang thông tin đã được ghi trên đĩa. Các thông tin không bị xoá trong quá trình đọc.

8.4.3 Các phương pháp mã hoá số liệu ghi lên đĩa

8.4.3.1 Giới thiệu về điều chế

- **Điều chế**

Đó là quá trình gắn hàm tin tức m(t) vào một tải tin (carrier) bằng cách làm biến đổi một hay nhiều thông số gọi là thông số điều chế của tải tin đó theo quy luật biến thiên của hàm m(t).

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Có thể thực hiện điều chế biên độ, tần số hoặc pha của tải tin (sóng mang thông tin), sau đây chúng ta tìm hiểu về phương pháp điều chế biên độ và điều chế tần số.

- **Phương pháp điều biên AM (Amplitude Modulation)**

Điều chế biên độ là làm cho biên độ của tải tin biến đổi theo quy luật của hàm mang tin. Biểu thức thời gian của tín hiệu điện áp điều biên như sau:

$U_{AM}(t) = Uo\ [1 + m(t)].\cos(\omega_o t) \equiv K(t)\ .\ U(t)$

với $K(t) = 1 + m(t)$; $U(t) = Uo.Cos(\omega_o t)$.

- **Phương pháp điều tần FM (Frequency Modulation)**

Nguyên tắc chung của phương pháp điều tần là dùng hàm số mang tin tức m(t) khống chế trực tiếp tần số của mạch tạo dao động tải tin (carrier) . Các mạch dao động tạo tải tin thông dụng thường thuộc một trong hai loại LC và RC mà tần số tạo ra chủ yếu phụ thuộc vào L, hay R, hay C của mạch; ta sẽ khống chế các giá trị này, làm nó thay đổi theo quy luật của m(t).

Biểu thức thời gian của tín hiệu điều tần là:

$U_{FM}(t) = U_o * Sin\ (\omega_o + m(t))\ t$.

Các phương pháp mã hoá số liệu thông dụng đối với đĩa từ:

- Phương pháp điều tần FM (Frequency Modulation)
- Phương pháp điều tần cải biên MFM (Modified Frequency Modulation)
- Phương pháp điều tần cải biên hai lần MMFM hay M^2FM (Modified Modified Frequency Modulation)
- Phương pháp mã nhóm GCR (Group Code Recording)

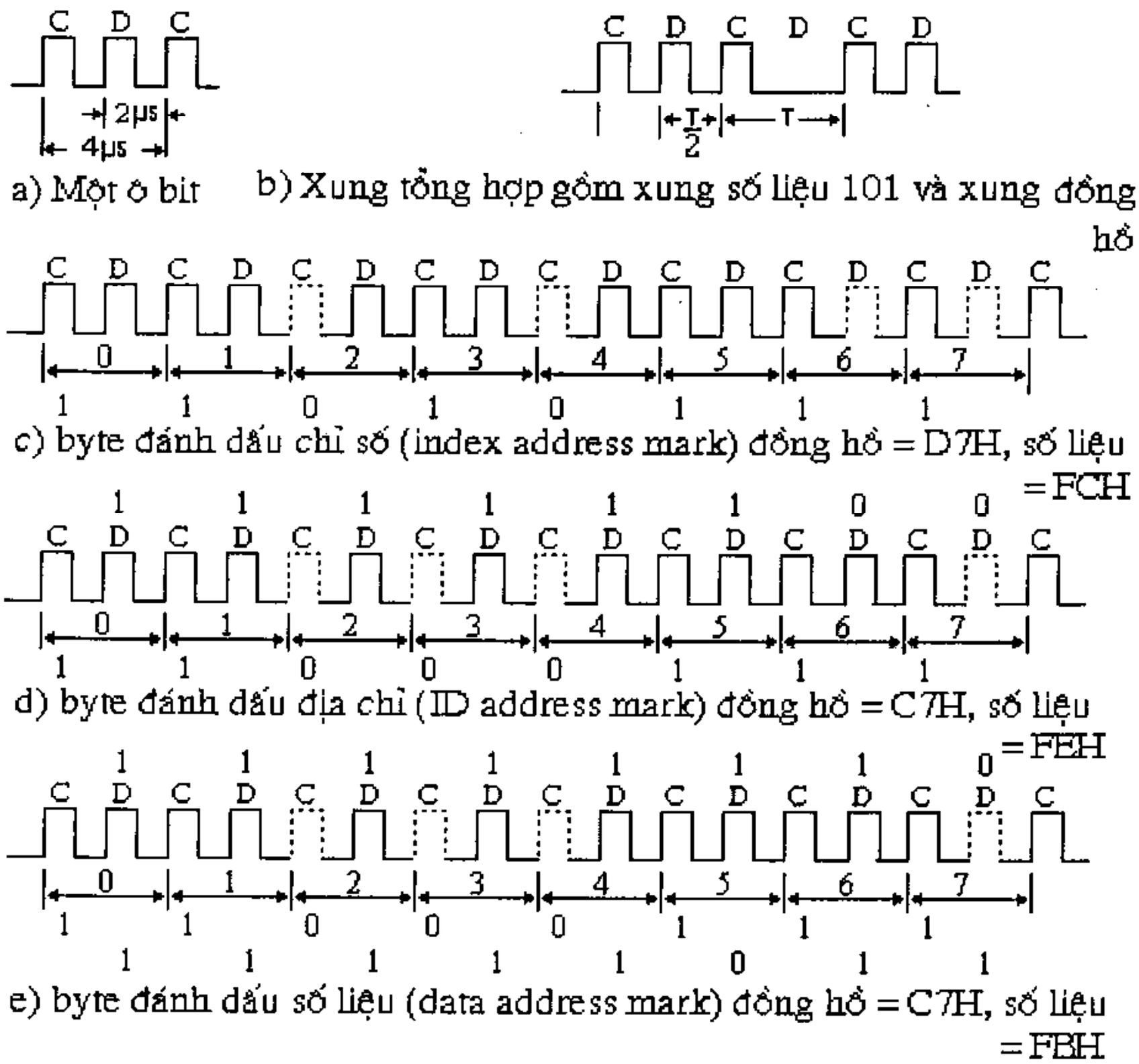
8.4.3.2 Phương pháp mã hoá FM (Frequency Modulation)

Các xung đồng hồ được ghi đều đặn, khoảng cách giữa hai xung đồng hồ liên tiếp được gọi là một ô bit, xung số liệu được ghi ở giữa ô bit. Nếu có xung số liệu thì có nghĩa là bit số liệu là 1, nếu không có xung số liệu (hay nói chính xác hơn là biên độ xung số liệu =0) thì có nghĩa là bit số liệu = 0, xem hình 8-16a.

Như vậy 1 byte số liệu (8 bit số liệu) ghi trên đĩa bao gồm giá trị thực của byte số liệu đó và giá trị FFH của byte đồng hồ (số hexa FF = 1111 1111 nhị phân).

Nếu ký hiệu tần số xung đồng hồ là f, chu kỳ xung đồng hồ là T (hình 8-16b), thì f = 1/T =1/4μs = 250000 (T=4μs là một giá trị chuẩn của phương pháp FM); còn tần số của dãy xung tổng hợp từ các xung đồng hồ và các xung số liệu có giá trị cực đại là 2f khi tất cả các số liệu = 1. Như vậy ta thấy rằng số liệu thay đổi theo thời gian thì tần số của dãy xung tổng hợp thay đổi theo số liệu; Đó chính là việc điều chế tần số lặp lại của xung.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 8-16 Phương pháp mã hoá FM (Frequency Modulation)

- **Byte đánh dấu chỉ** số (xem hình 8-16c)

(Ý nghĩa của các byte đánh dấu, trong đó có byte đánh dấu chỉ số sẽ được giải thích rõ hơn ở phần sau, khi nghiên cứu về tổ chức rãnh của IBM, mục "Đĩa mềm...")

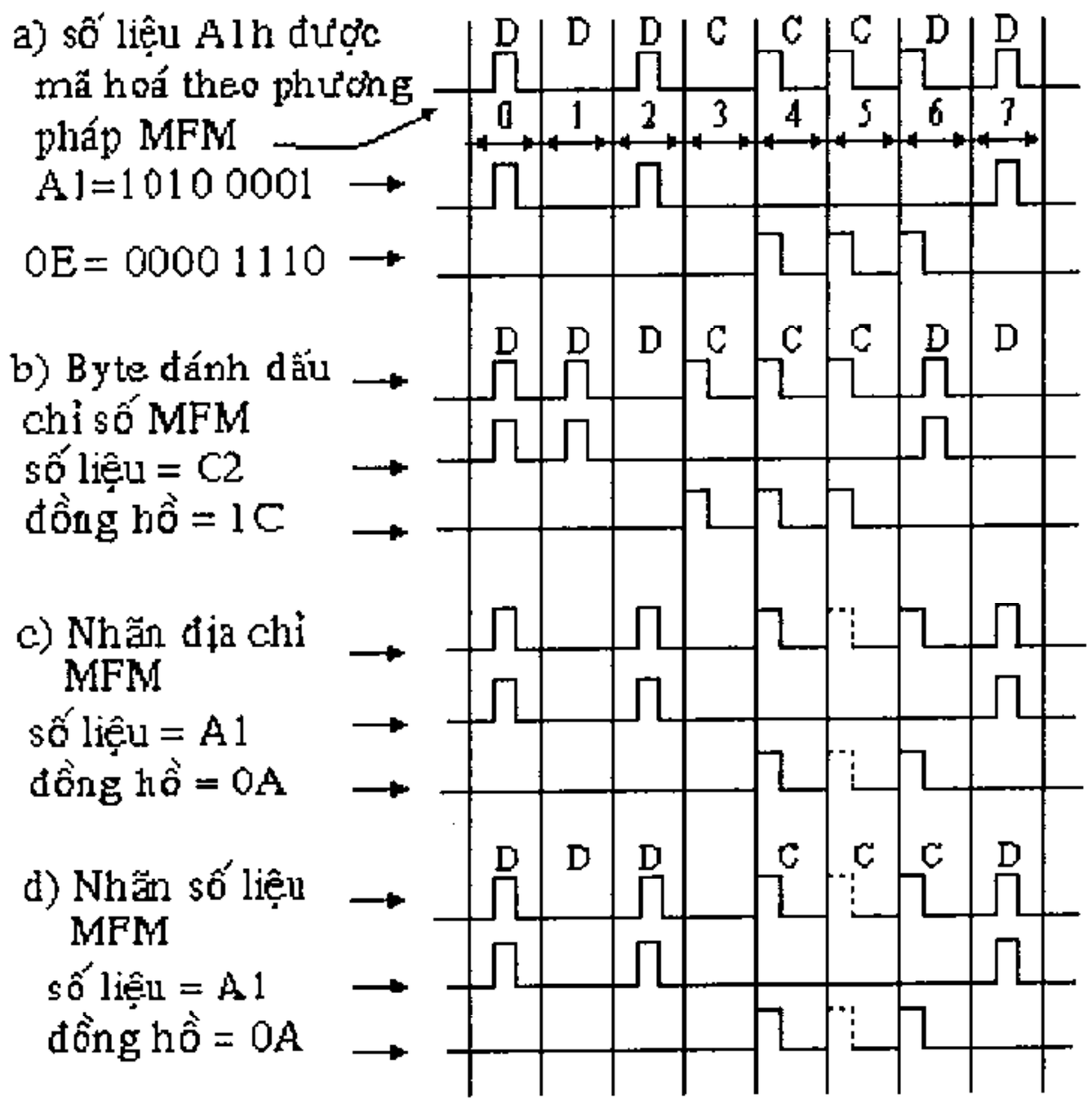
Khác với các byte số liệu thông thường, ngoài giá trị số liệu FCH, giá trị của byte đồng hồ không phải là FFH mà là D7H; Chính nhờ cấu trúc đặc biệt này mà đơn vị điều khiển đĩa phân biệt được các byte đánh dấu chỉ số với các byte số liệu thông thường có cùng giá trị số liệu và đồng bộ được quá trình biến đổi nối tiếp/song song khi đọc số liệu.

- **Byte đánh dấu địa chỉ** (xem hình 8-16d): Giá trị byte đồng hồ là C7H, giá trị byte số liệu là FEH
- **Byte đánh dấu số liệu** (xem hình 8-16e): Giá trị byte đồng hồ là C7H, giá trị byte số liệu là FBH.
- Ưu, nhược điểm của phương pháp FM: ưu điểm của phương pháp này bộ tách số liệu đơn giản mà vẫn bảo đảm hoạt động tin cậy. Nhược điểm của phương pháp FM là độ dư thừa thông tin tới 50%, do đó lãng phí dung lượng của đĩa.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

8.4.3.3 Phương pháp mã hoá MFM (Modified Frequency Modulation)

Thời gian được chia thành những khoảng đều nhau gọi là ô bit, xung số liệu được ghi ở giữa mỗi ô bit, trong khi xung đồng hồ chỉ được ghi ở đầu mỗi ô bit nếu trong ô bit này và ô bit trước đó bit số liệu cùng bằng 0. Xung đồng hồ sẽ xuất hiện liên tục khi các bit số liệu liên tục bằng 0 và do đó đồng bộ bit được bảo đảm trong cả hai trường hợp, có hoặc không có xung số liệu. Các byte đánh dấu được phân biệt với các byte số liệu nhờ cấu trúc đặc biệt là thiếu một số xung đồng hồ, giống như ở phương pháp FM. Xem hình 8-17.



Hình 8-17 Phương pháp mã hoá MFM

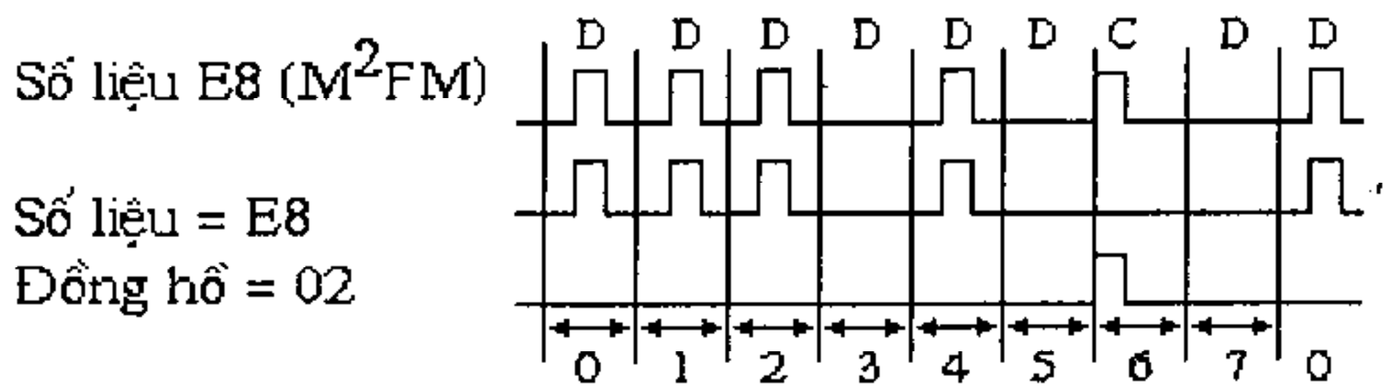
Ưu điểm của phương pháp MFM so với phương pháp FM: tăng tỉ lệ thông tin hữu ích ghi trên đĩa lên gấp đôi, nhờ đó tốc độ truyền số liệu cũng tăng gấp đôi dù tần số xung đồng hồ không thay đổi.

Nhược điểm: dãy xung số liệu và đồng hồ được ghi lên đĩa với nhiều tần số hơn: 1F, 2F, 4/3F... Việc có tới 5 tần số ghi làm phức tạp thêm mạch tách số liệu (giải mã).

8.4.3.4 Phương pháp mã hoá M^2FM (Modified MFM)

Phương pháp này có nhiều điểm giống phương pháp MFM, số liệu được ghi ở giữa các ô bit còn các xung đồng hồ được ghi ở đầu các ô bit, nếu tại ô bit đó bit số liệu =0 và các bit số liệu tại các ô bit trước *và sau đó* cũng bằng 0. (Đặc điểm khác nhau của hai phương pháp được in nghiêng và gạch chân). Xem hình 8-18.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 8-18 Số liệu, đồng hồ mã hoá theo phương pháp M^2FM

Như vậy so với phương pháp MFM, mật độ ghi thông tin hữu ích trên đĩa không thay đổi, vì vậy tốc độ truyền số liệu cũng không thay đổi. Dãy xung tổng hợp (số liệu cộng với đồng hồ) được ghi với 4 tần số 1F, 2F, 4/3F, 4/5F.

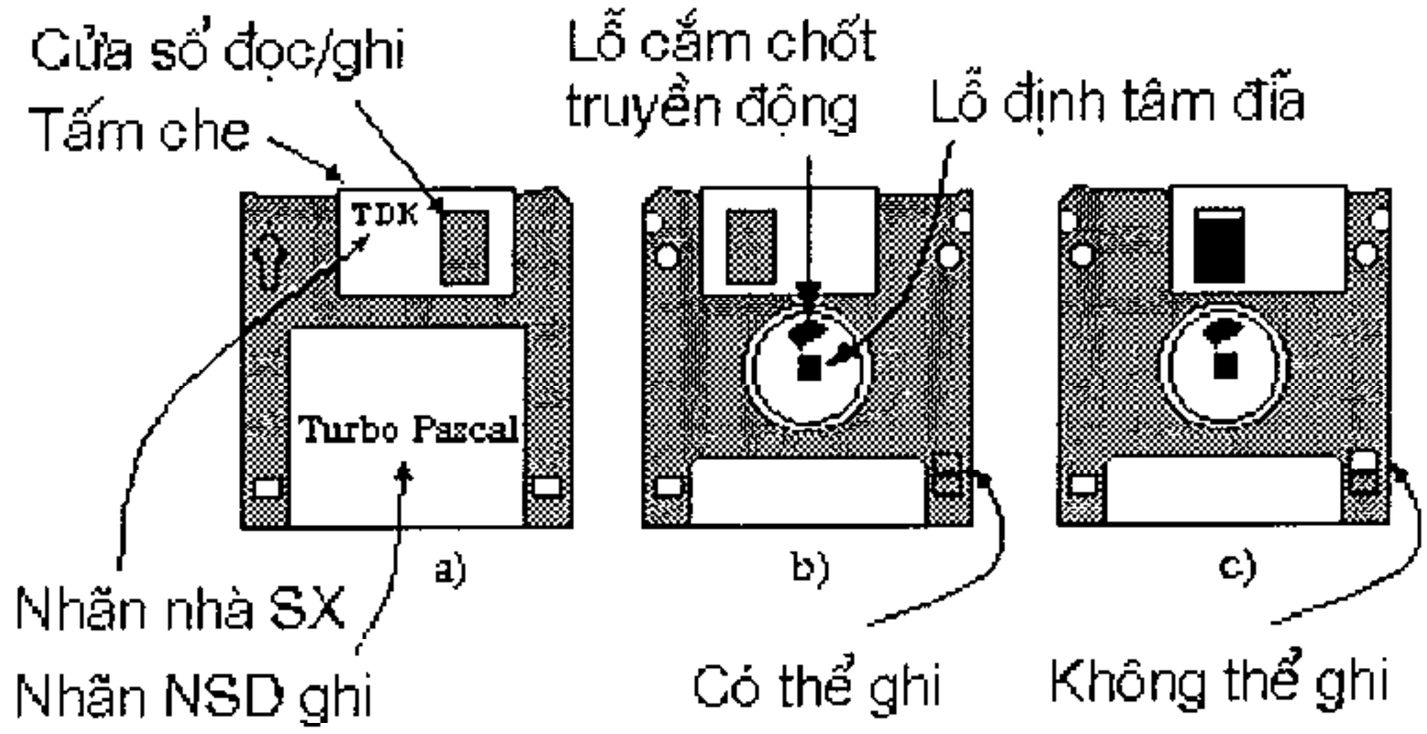
Ưu điểm của phương pháp M^2FM so với phương pháp MFM:

- Việc tách số liệu đơn giản hơn vì số tần số có thể có trong dãy xung tổng hợp ít hơn (chỉ còn 4).
- Giảm được số xung đồng hồ cần ghi, bởi vì chỉ khi nào có trên hai bit dữ liệu liên tiếp bằng 0 mới phải ghi xung đồng hồ. Theo nghiên cứu thống kê nhờ đó có thể giảm tới 1/3 số xung đồng hồ cần ghi.

8.4.4 Đĩa mềm và ổ đĩa mềm

8.4.4.1 Cấu tạo và các chỉ tiêu kỹ thuật chính của đĩa mềm

Đĩa mềm (floppy disk) được làm bằng chất dẻo Milar, dày 76μm, trên bề mặt được phủ một lớp bột vật liệu sắt từ (oxyt sắt, coban .v.v.) trộn với keo dính dày từ 1,3μm đến 1,4μm. Đĩa mềm được đặt trong một phong bì tương đối cứng, thường được dán một lớp chất xốp ở mặt trong của phong bì để lau bụi và làm giảm hệ số ma sát. Đĩa mềm có 3 loại kích thước: 8", 51/4" và 31/2", trong đó hiện nay loại 31/2" được sử dụng rộng rãi hơn cả.



Hình 8-19 Đĩa mềm 3 1/2" a/ Mặt trên; b/ và c/ Mặt dưới

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Mặt đĩa - Side/Head**

Thông tin có thể ghi trên một hoặc cả hai mặt đĩa, mỗi mặt đĩa cần phải có một đầu từ đọc/ghi. Các đĩa sử dụng 1 mặt được ký hiệu là - SS (Single Sided), 2 mặt - DS (Double Sided). Các đĩa ghi số liệu theo mật độ đơn, sử dụng phương pháp FM được ký hiệu là SD (Single Density). Các đĩa ghi số liệu theo mật độ kép, sử dụng phương pháp MFM được ký hiệu là DD (Double Density).

- **Rãnh □ Track; Cylinder**

Số liệu được ghi trên đĩa theo các đường tròn đồng tâm hay còn gọi là rãnh, được đánh số từ ngoài vào tâm theo thứ tự từ nhỏ tới lớn, rãnh ngoài cùng là rãnh 0. Trên hai mặt đĩa luôn có các cặp rãnh có cùng chỉ số, mỗi cặp được gọi là một **cylinder** (từ trụ), tham số cylinder cũng thường được sử dụng giống như track.

- **Cung - Sector**

Mỗi rãnh được chia làm nhiều phần bằng nhau gọi là cung - sector, số cung/rãnh có thể bằng 8, 9, 15 hay 18 tuỳ thuộc cách phân chia và loại đĩa. Việc chia sector được thực hiện bằng một trong 2 phương pháp sau đây:

1. Chia sector bằng phần mềm (***Soft sectoring***):

   Đĩa 5 1/4" có một lỗ nhỏ gần tâm gọi là lỗ chỉ số, nó được cảm biến quang điện của đơn vị ổ đĩa nhận biết, đơn vị ổ đĩa căn cứ vào vị trí lỗ chỉ số để ghi các tín hiệu phân chia rãnh thành các phần bằng nhau. Đĩa 3 1/2" không có lỗ chỉ số nhưng nhờ có lỗ cắm chốt truyền động nên đĩa luôn luôn được kẹp vào hệ thống quay đĩa tại một vị trí xác định. Nhờ vậy đơn vị ổ đĩa có thể ghi các tín hiệu phân chia rãnh thành các phần bằng nhau như ở ổ đĩa 5 1/4".

2. Chia sector cố định (***Hard sectoring***): ngoài lỗ chỉ số với các chức năng như trên, còn có các lỗ cung cũng được nhận biết như lỗ chỉ số - chúng đánh dấu điểm bắt đầu cung số liệu trên rãnh. Phương pháp này hiện nay không còn được sử dụng.

Bảng 8-01 Một số số liệu về hai loại đĩa mềm thông dụng

Loại đĩa	5"1/4		3"1/2	
Dung lượng (bytes)	360K	1.2M	720K	1.44M
Số rãnh	40	80	80	80
Số sector/rãnh	9	15	9	18
Số đầu từ (mặt)	2	2	2	2
Tốc độ quay (vòng/phút)	300	300	300	300
Tốc độ truyền dữ liệu (kbps)	250	500	250	500

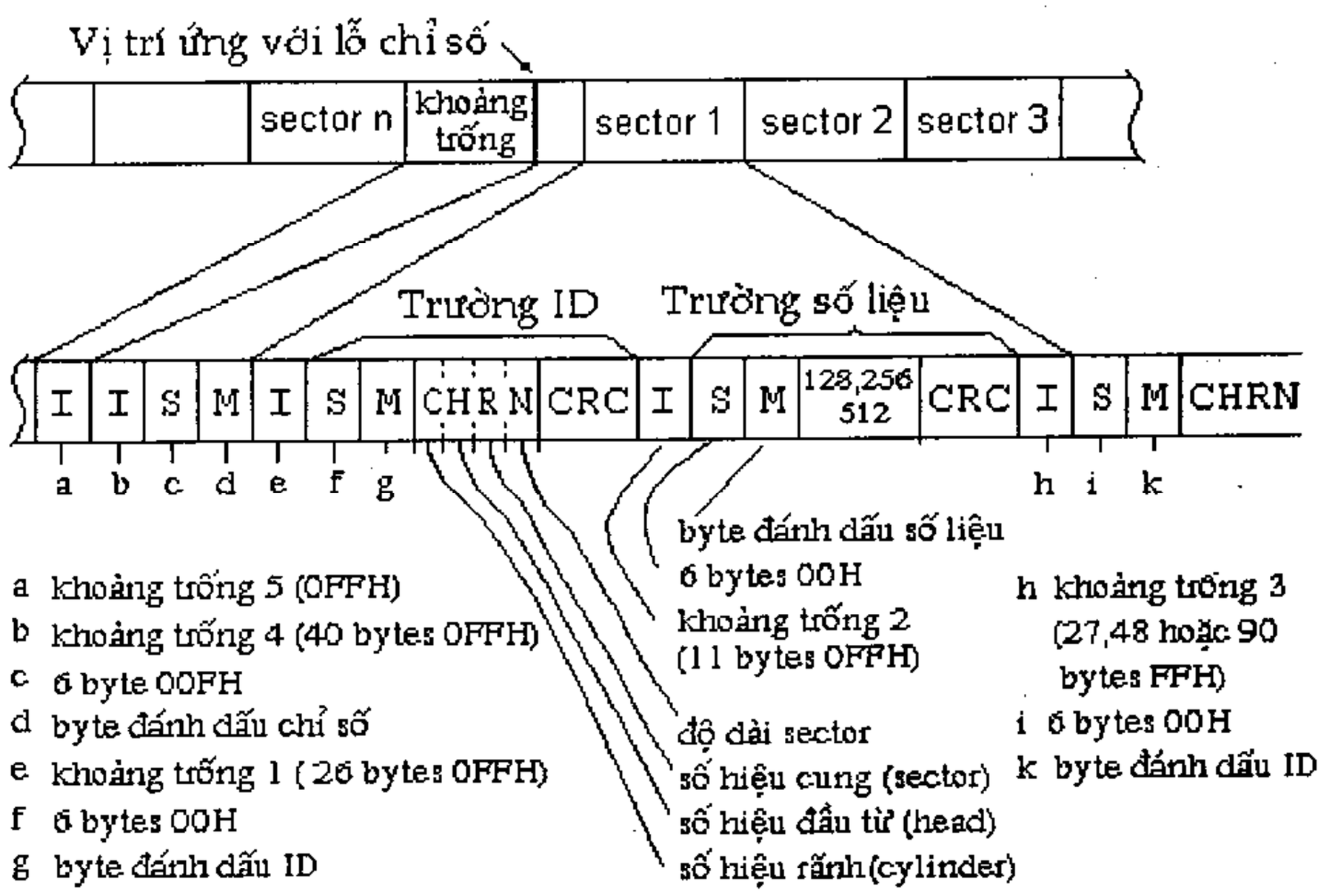
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

8.4.4.2 Tổ chức rãnh theo tiêu chuẩn IBM 3740 (FM)

Việc phân chia đĩa được thực hiện khi ta FORMAT đĩa, việc chia thực hiện theo quy cách thống nhất của hệ điều hành. Thường mỗi cung (sector) gồm có các trường chính sau:

- Trường địa chỉ - ID (Identifier Field)
- Trường số liệu
- Trường đồng bộ: được ghi ở trước mỗi trường địa chỉ, gồm các byte 0FFH và 00H dùng để nhận biết đúng các byte đánh dấu.
- Khoảng trống (gaps): phân cách các các trường và các sector, trên đó có ghi các số liệu hợp lý, cùng được tạo ra một lần với các trường địa chỉ và số liệu khi ta FORMAT đĩa.

Trên hình 8-20 là tổ chức rãnh theo tiêu chuẩn IBM 3740 (FM). Hệ điều hành DOS sử dụng các sector đĩa mềm có cùng kích thước 512 byte, mặc dù chiều dài sector trên các rãnh khác nhau là không bằng nhau, nhờ đó việc quản lý thông tin đơn giản hơn.



Hình 8-20 Tổ chức rãnh theo tiêu chuẩn IBM 3740 (FM)

8.4.4.3 Ổ đĩa mềm - FDD (Floppy Disk Drive)

Cấu tạo và nguyên lý hoạt động

Ổ đĩa mềm bao gồm các bộ phận cơ học và điện tử cần thiết để thực hiện các chức năng sau đây:

1. quay đĩa mềm với tốc độ quy định

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

2. dịch chuyển đầu từ đọc/ghi theo phương bán kính đĩa đến rãnh mong muốn
3. hạ đầu từ đọc/ghi, cho nó tiếp xúc với mặt đĩa.
4. thông báo trạng thái của ổ đĩa (vị trí của đầu từ, trạng thái sẵn sàng làm việc, có cấm viết không, lỗi đọc/ghi .v.v.)
5. ghi và đọc số liệu

Ổ đĩa mềm sử dụng trong IBM PC là loại theo tiêu chuẩn công nghiệp 5 1/4" hoặc 3 1/2", đĩa được quay với tốc độ 300 Vg/ph, mật độ rãnh là 48 TPI, 96 TPI hoặc 135 TPI (đĩa 3 1/2", 1,44MB có mật độ rãnh 135 TPI).

Tốc độ ghi với phương pháp MFM là: 250 000 bit/giây

Tốc độ ghi với phương pháp FM là: 125 000 bit/giây

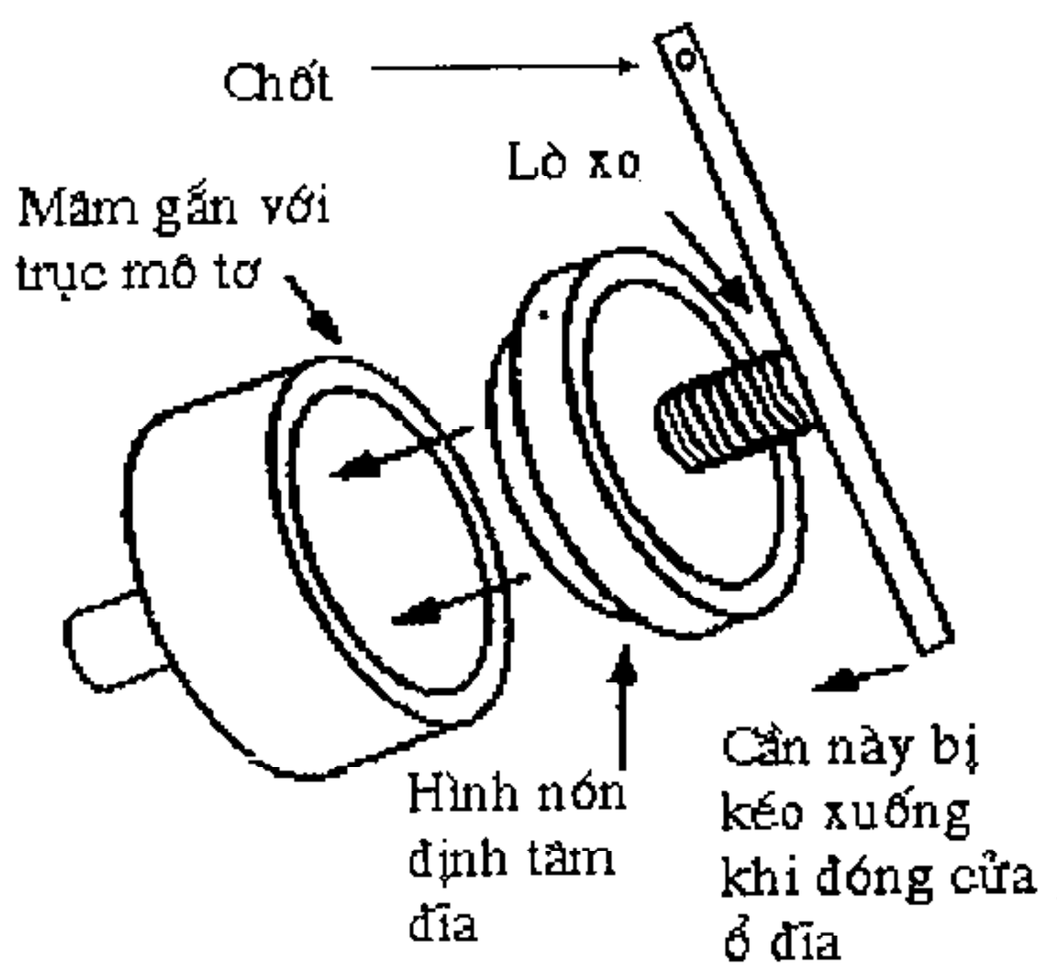
Các bộ phận chính của ổ đĩa:

- **Bộ phận quay đĩa** (Hình 8-21)

Thực hiện giữ đĩa ở tâm trục quay và quay nó với vận tốc 300 vg/ph. Trục quay gắn trực tiếp với trục mô tơ điện một chiều, không dùng dây cu-roa như ở các ổ đĩa loại cổ, tránh hiện tượng "trượt" dây cu-roa.

Mô tơ quay đĩa là loại đồng bộ nhiều pha, không dùng chổi than, tránh hiện tượng mòn chổi và tránh gây nhiễu, phương pháp điều khiển trực tiếp này là một thiết kế có độ tin cậy cao, chính xác.

Đĩa mềm có liên hệ cơ học với trục quay thông qua bộ phận ngàm của trục quay. Bộ phận này cũng thực hiện định tâm đĩa vào trục bằng bộ phận hình nón bằng chất dẻo, được gọi là hình nón định tâm. Hình nón định tâm này được lắp khít vào đường kính trong của bộ phận trục quay.



Hình 8-21 Hình nón ngàm và định tâm đĩa (FDD 5 1/4")

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Đầu từ và thanh mang** (Hình 8-22)

Đó là bộ phận quan trọng nhất của ổ đĩa, bao gồm một đầu từ đọc/ghi gắn trên một thanh mang chuyển động trượt theo đường thẳng. Thanh mang đầu từ được điều khiển chuyển động bằng một cơ cấu chấp hành điện-cơ, thực chất là một mô tơ bước nhiều pha.

Các đầu từ đọc/ghi trong các ổ đĩa mềm của các hãng sản xuất đều giống nhau. So với đầu từ ghi băng, có khác một điều: 2 cực của đầu từ xoá nằm theo chiều dọc, theo cả hai bên của đầu từ đọc/ghi. Thiết kế đầu xoá này được gọi là xoá đường hầm (tunnel), nó xoá sạch cả rãnh ghi thông tin và 2 bên mép của rãnh để tránh hiện tượng "xuyên âm".

Đầu từ 0 thường nằm phía dưới (đĩa), gắn chặt trên thanh mang. Còn đầu từ 1 được gắn chặt trên một nhíp lò xo. Người ta làm như vậy để cả 2 đầu từ cùng ép sát lên mặt đĩa.

- **Cơ cấu điều khiển thanh mang đầu từ**

Cơ cấu này sử dụng mô tơ bước, có 3 loại chính được sử dụng rộng rãi: dùng vít dẫn, dùng băng kéo và dùng đĩa xoắn.

Cơ cấu dùng vít dẫn: hoạt động giống như vít và ecu, trục mô tơ bước được nối trực tiếp với trục vít dẫn, vít dẫn được xâu theo ren xoắn xuyên qua thanh mang đầu từ. Khi mô tơ bước quay thì thanh mang sẽ được kéo lui hay đẩy tới theo phương trục xoắn. Xem minh hoạ trên hình 8-22.

Cơ cấu dùng băng kéo: hoạt động giống như dây cu-roa và puli, băng kéo được bắt chặt một đầu vào thanh mang ở cuối tầm trượt và quấn một vòng quanh puli của mô tơ, đầu thứ hai được gắn chặt vào puli, khi mô tơ quay thì thanh mang đầu từ trượt trên trục dẫn hướng.

Cơ cấu đĩa rãnh xoắn: hoạt động gần giống như đĩa hát. Trục mô tơ bước được gắn với đĩa rãnh xoắn. Trên thanh mang có một chốt dẫn, trên đầu chốt có đính một hạt hình cầu nhỏ lăn theo rãnh xoắn trên đĩa, khi đĩa rãnh xoắn quay thì thanh mang chuyển động vào/ra theo một đường thẳng.

- **Các bộ cảm biến**

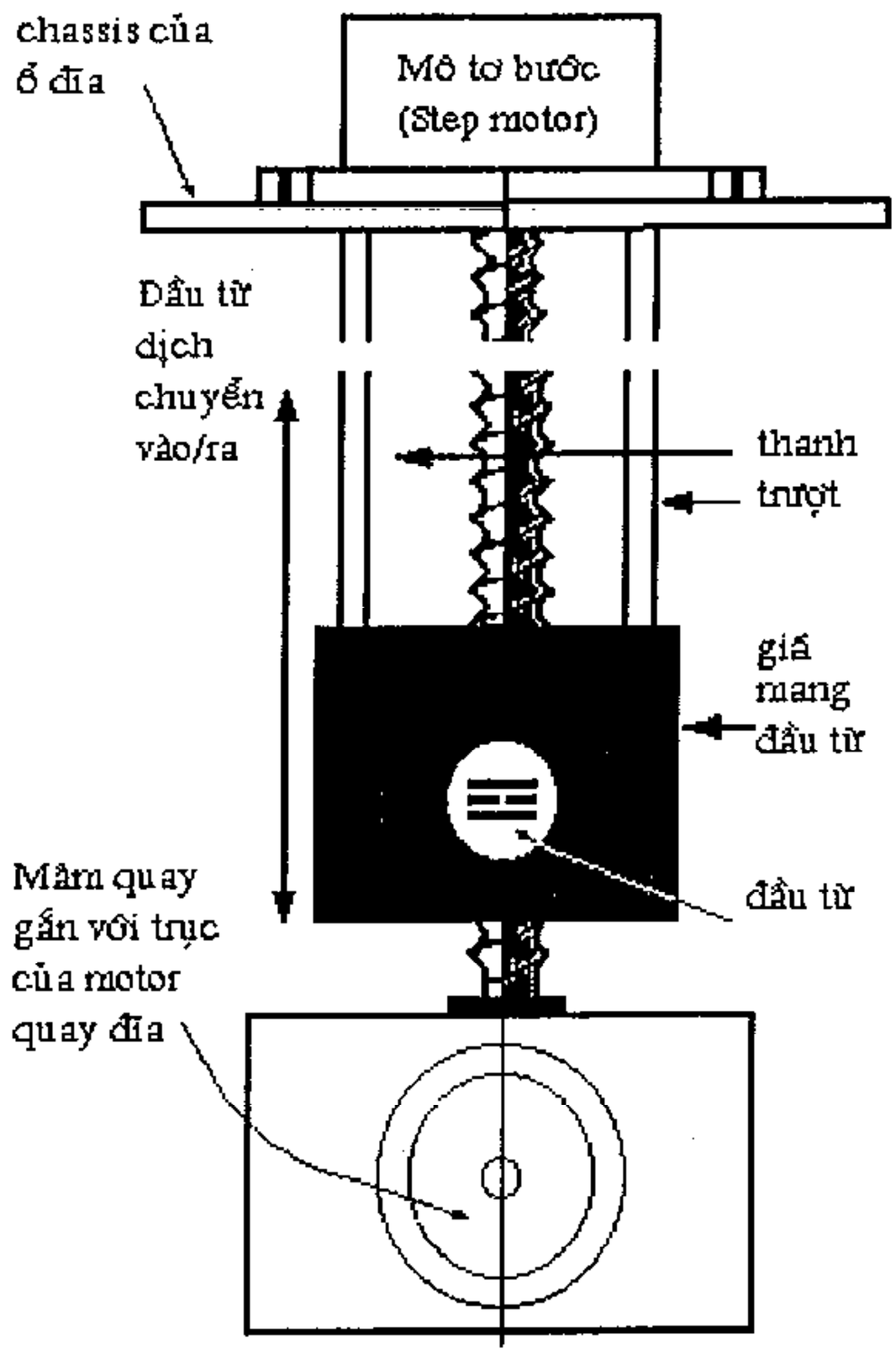
Cảm biến lỗ chỉ số: dùng trong loại ổ đĩa 5 1/4" để phát hiện lỗ chỉ số, các mạch điện tử kiểm soát nó để biết đâu là nơi bắt đầu sector, và khi nào thì bắt đầu tìm sector; đồng thời biết được khi nào một vòng quay đã hết. Cảm biến dùng một cặp Phototransistor-LED (Light Emitted Diot - điôt phát quang), LED được gắn bên dưới lỗ chỉ số, còn phototransistor được gắn bên trên. Xung điện tạo ra bởi Phototransistor được dùng để tạo ra xung đánh dấu.

Cảm biến rãnh 0: khi mới bật máy, mạch điện tử điều khiển ổ đĩa không biết đầu từ đang ở trên rãnh nào, nên cần phải có một lệnh chuẩn lại vị trí ban đầu. Lệnh này phát tới mạch điều khiển, mạch này phát ra những xung bước cho tới khi đầu từ tiến đến đúng rãnh 0, vị trí này được xác định bởi cảm biến rãnh 0. Cảm biến dùng 1

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

cặp Phototransistor-LED, trên thanh mang có lắp 1 cờ nhỏ (miếng kim loại nhỏ), còn Phototransistor + LED lắp trên khung (chassis) phía sau của ổ đĩa, khi đầu từ chuyển động đến vị trí rãnh 0, thì cờ hiệu cũng nằm đúng giữa LED và Phototransistor, báo cho mạch kiểm soát biết vị trí của đầu từ.

Cảm biến chống ghi: có nhiệm vụ xác định xem khe hở "write protect" ở góc đĩa có bị chắn hay không, nếu có thì tạo một tín hiệu logic báo cho đơn bị ổ đĩa biết là đĩa đã được gạt "bảo vệ chống ghi". Cảm biến bảo vệ cấm ghi thường dùng kiểu cảm quang như ở cảm biến lỗ chỉ số, đôi khi có hãng dùng công tắc cơ khí.



Hình 8-22 Cơ cấu điều khiển đĩa bằng vít dẫn

- **Board mạch điện tử**

Board này chứa các mạch tương thích lối ra của các cảm biến với mạch kiểm soát, các mạch điều khiển mô tơ, các mạch khuyếch đại. Đối với các ổ đĩa 2 mặt còn có mạch điện để chuyển mạch đầu từ.v.v.

8.4.5 Đĩa cứng - HDD (Hard Disk Drive)

Nhu cầu của các ứng dụng, nhất là trong lĩnh vực quản lý đòi hỏi phải có thiết bị nhớ ngoài dung lượng lớn, tốc độ cao và tin cậy đã thúc đẩy việc nghiên cứu, chế tạo và phát triển các công nghệ sản xuất đĩa cứng. Các công nghệ sản xuất đĩa cứng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

đã trải qua nhiều giai đoạn, số sản phẩm rất phong phú, sự khác nhau cơ bản giữa các công nghệ là ở phương pháp định vị đầu từ trên mặt đĩa.

8.4.5.1 Đặc điểm chung

Đĩa cứng thuật ngữ tiếng Anh là Hard Disk, đôi khi người ta còn dùng thuật ngữ Fixed Disk hoặc None-Removable Disk để chỉ đặc tính quan trọng của nó là trong thiết bị ổ đĩa cứng các đĩa được gắn cố định với hệ thống quay đĩa, không thể tháo rời ra được. Thông thường khái niệm đĩa cứng được hiểu là ổ đĩa cứng có các đĩa được gắn cố định bên trong nó cùng với hệ thống định vị đầu từ .v.v.

Các đĩa được làm bằng vật liệu cứng, trên đó phủ chất sắt từ. Nhờ đĩa cứng và rất phẳng nên đầu từ có thể được định vị rất chính xác, không cần tiếp xúc trực tiếp mà chỉ cần bay sát mặt đĩa cũng có thể đọc/ghi thông tin. Ngoài ra còn có thể nâng cao tốc độ quay của đĩa cứng lớn hơn tốc độ quay của ổ đĩa mềm rất nhiều mà không sợ ma sát giữa đầu từ và mặt đĩa gây hư hỏng.

Các phương pháp mã hoá thông tin được sử dụng cho đĩa cứng: FM, MFM, M^2FM, GCR...

8.4.5.2 Nguyên lý cấu tạo

Trên hình 8-23 là minh hoạ một mặt đĩa của một đĩa cứng, nó được gắn chặt vào trục quay của mô-tơ và quay với tốc độ từ 3600 đến 5400 vòng/phút. Đầu đọc/ghi (R/W head) được gắn chặt vào một cái cần, còn cần này thì lại được gắn vào một trục quay khác. Trục quay thứ hai này cho phép điều khiển vị trí của đầu từ bằng cách quay. Mỗi mặt đĩa có một đầu từ phụ trách đọc/ghi, tất cả các cần gắn đầu từ trên các mặt đĩa được gắn vào cùng một trục và cùng quay với nhau.

Hình vẽ 8-23 chỉ để minh hoạ nguyên tắc làm việc, các bộ phận được vẽ trên hình cũng không theo cùng một tỉ lệ với các bộ phận của một đĩa cứng thực. Trong các đĩa cứng ngày nay, mật độ rãnh vào khoảng 130 tracks/mm (≈4000TPI), một bit chiếm một chiều dài 0.3μm và chiều rộng 7μm trên rãnh (diện tích ≈ $2.10^{-8}cm^2$).

Chúng ta bắt đầu nghiên cứu với một câu hỏi đầu tiên, hết sức hiển nhiên: Bằng cách nào để các bộ phận cơ khí của ổ đĩa có thể đọc/ghi đúng các bit trên đĩa với mật độ bit cao đến như vậy?

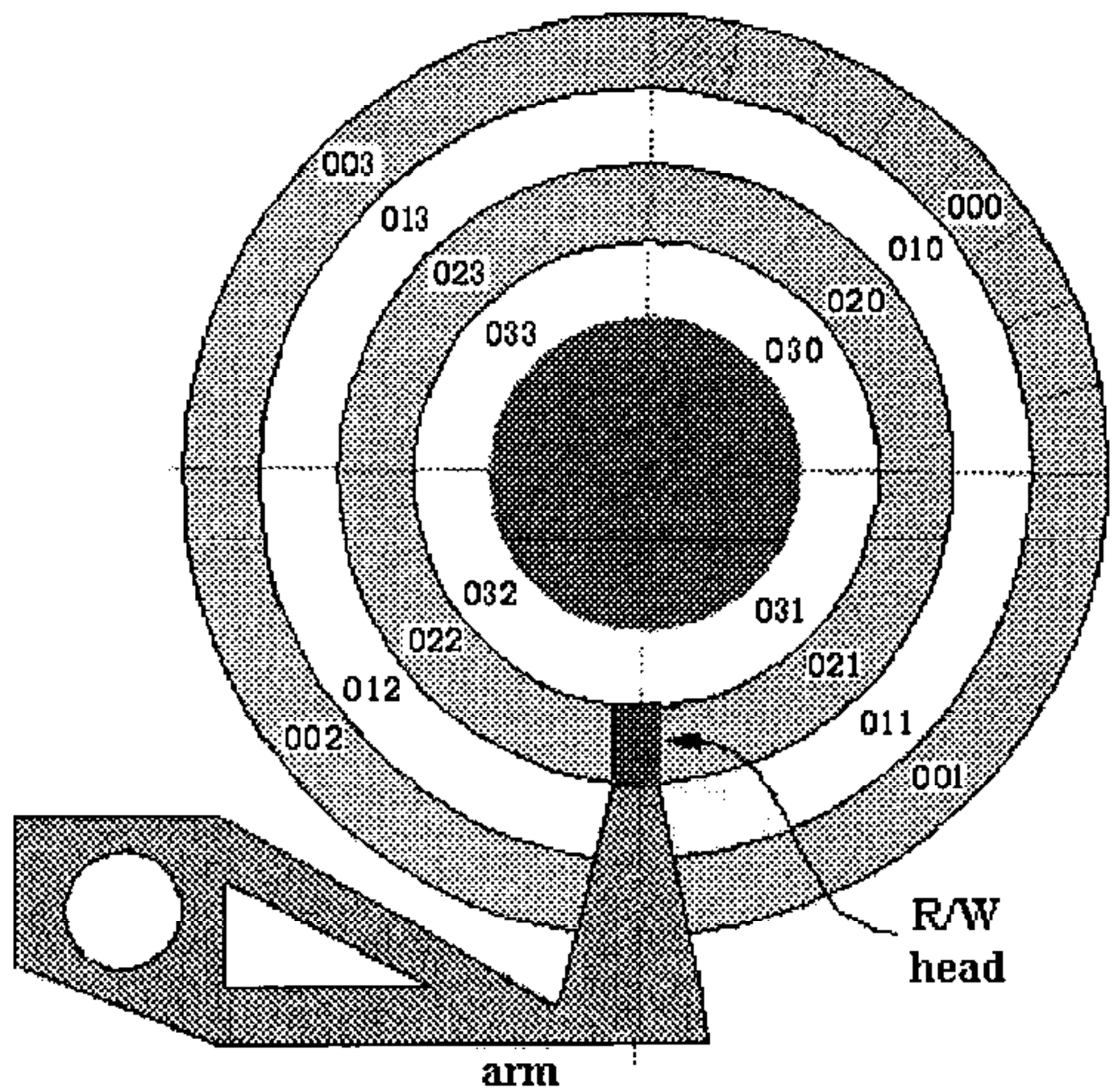
Đối với hệ thống cơ khí này, một vị trí trên đĩa có thể chỉ ra bởi một toạ độ (hay địa chỉ) 3 chiều. Toạ độ thứ nhất là độ cao Z, để chọn mặt đĩa. Toạ độ thứ hai là bán kính R, để chọn cylinder. Toạ độ thứ ba là góc θ, để chọn cả sector lẫn vị trí bit bên trong sector. Nhìn chung việc điều khiển hệ thống định vị đầu từ thường không phải chỉ dựa trên các tham số Z, R và θ mà chủ yếu là các gia số ΔZ, ΔR và Δθ.

Việc chọn đúng đầu từ bằng tham số Z rất đơn giản, rất dễ điều khiển, có thể thực hiện được bằng điện tử và không cần phải có phản hồi.

Vị trí của cần mang đầu từ được điều khiển thông qua tham số R, làm quay trục quay chung của tất cả các cần mang đầu từ gắn chặt với nó.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Tham số thứ ba θ, được thiết lập bằng thời gian. Làm thế nào để có thể điều khiển nó với độ chính xác cần thiết có lẽ là vấn đề khó nhất, chúng ta cần phải biết được một bit bắt đầu và kết thúc ở đâu, nó là 1 hay là 0.



Hình 8-23 Một mặt đĩa của đĩa cứng

Thiết kế hệ thống đĩa là một cuộc cạnh tranh kinh tế, đích mà mọi người cùng nhắm tới là chứa được trên đĩa càng nhiều bit càng tốt trong khi đó vẫn đảm bảo ghi, đọc tin cậy. Trở ngại chính là các giới hạn nằm trong hệ thống định vị cơ khí, kích thước đầu từ, độ nhạy của đầu từ, kích thước miền từ (magnetic domain) và việc khắc phục lỗi. Tất cả các vấn đề khó khăn này có mối quan hệ tương tác với nhau và tạo nên một xác suất xảy ra lỗi mà chúng ta không phát hiện ra được hoặc không thể khắc phục được trong một chu kỳ ghi và đọc file. Xác suất này không được vượt quá một giá trị mà chúng ta coi là mức có thể chấp nhận được.

8.4.5.3 Việc định vị đầu từ

Hầu hết các ổ đĩa đều có nhiều đĩa, trên mỗi bề mặt của các đĩa đều có một thanh mang trên đó gắn đầu từ đọc/ghi. Các thanh mang này được gắn chặt với nhau và có chung một trục quay, chúng được điều khiển như một đơn vị, vì thế tại mọi thời điểm các đầu từ đều nằm trên cùng một cylinder.

- Trong các ổ đĩa cứng chất lượng cao, đắt tiền

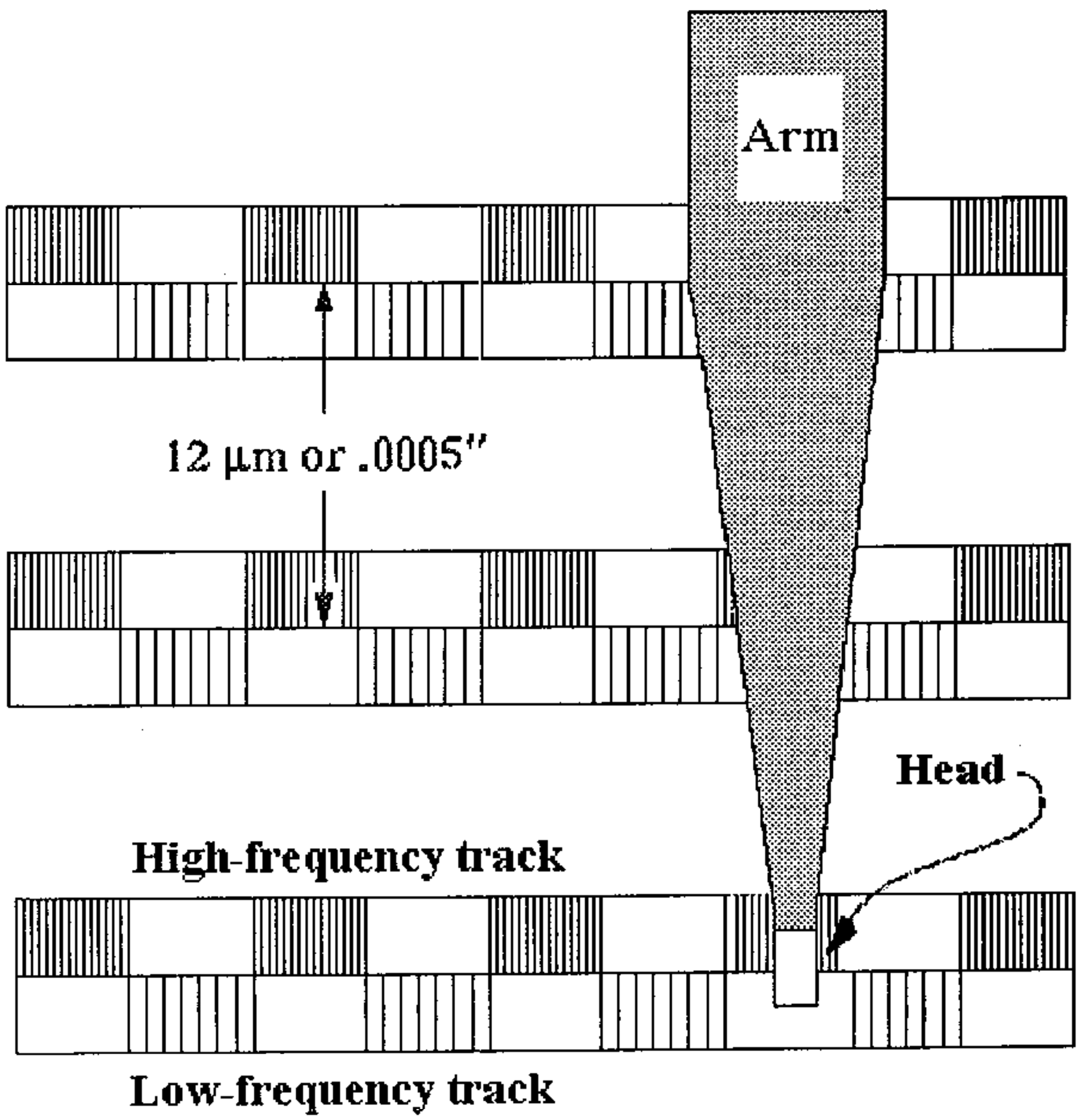
Trong các ổ đĩa cứng chất lượng cao, dung lượng lớn và có nhiều đĩa, người ta dành hẳn một mặt đĩa cho việc bám sát thông tin (***tracking***), các mặt đĩa còn lại chứa dữ liệu. Các mặt đĩa dữ liệu giống như trên hình 8-23. Trong mỗi sector, phần

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

đầu là trường ID, tiếp theo mới là dữ liệu. Dữ liệu được sắp xếp một cách tuần tự dọc theo mỗi rãnh. Thực tế đó là dữ liệu lấy từ bộ nhớ trong, được bổ sung thêm một số nhóm bit theo một số thuật toán phát hiện lỗi hoặc sửa lỗi. Mặt đĩa có chứa các thông tin tracking được gọi là ***đĩa servo***, nó được nhà sản xuất hoặc công ty bán ra đơn vị ổ đĩa này ghi trước. Việc làm này định nghĩa "vĩnh viễn" vị trí của các rãnh trên tất cả các đĩa.

Sự mòn trục ổ đĩa, chênh lệch nhiệt độ, các nhiễu loạn cơ học như xung động của không khí, rung động do di chuyển máy tính không cẩn thận... tất cả đều có thể gây nên lỗi cho việc bám sát rãnh. (Trên thực tế các ổ đĩa 3 1/2 " quay với tốc độ 5400vòng/phút thường không có vấn đề lỗi do mòn trục)

Để lấy được dữ liệu, các sector trên các mặt đĩa dữ liệu được phân cách với nhau bởi các khối ngắn các tín hiệu tracking. Các tín hiệu tracking này cũng được ghi trước giống như ghi trước các thông tin tracking trên mặt đĩa servo.



Hình 8-24 Một Cylinder ghi tín hiệu bám sát rãnh (tracking) để sinh ra tín hiệu servo

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Trong các ổ đĩa cứng loại rẻ tiền

Các ổ đĩa cứng loại rẻ tiền thường không có đĩa servo, chỉ có tín hiệu servo trên các mặt đĩa dữ liệu. Bởi vì các tín hiệu servo trên mặt đĩa là tương đối thưa thớt, nên các tính năng tracking kém hơn đáng kể so với loại đĩa có đĩa servo, điều này làm cho thời gian ổn định dài hơn.

Với việc sử dụng kết hợp cả đĩa servo dành riêng và các tín hiệu servo trên mặt đĩa dữ liệu, có thể thực hiện được việc bám sát (tracking) rãnh trong phạm vi vài micro mét một cách dễ dàng.

Các thành phần của sơ đồ tracking được thể hiện trên hình 8-24. Mô-tơ quay các cần mang đầu từ có thể đặt đầu từ vào rãnh mong muốn một cách tương đối chính xác. Mỗi rãnh có 2 nửa, mỗi nửa được ghi trước các tín hiệu có tần số khác nhau. Nếu đầu từ bị lệch khỏi tâm rãnh thì một trong hai tín hiệu sẽ có biên độ lớn hơn tín hiệu kia. Mạch điện tử phát hiện ra sẽ điều khiển lái đầu từ theo hướng làm cho tín hiệu yếu mạnh lên. Bởi vì các rãnh sinh ra các tín hiệu này được ghi trước khi người ta ghi dữ liệu lên đĩa nên chúng định nghĩa các cylinder.

Khoảng cách giữa các track rất nhỏ, vì vậy hệ thống phải có khả năng phát hiện được việc đầu từ của nó bị đặt vào một cylinder không đúng. Điều này có thể thực hiện được nhờ việc trên mỗi sector của các mặt đĩa dữ liệu đều có trường ID, chúng sẽ được hệ thống thông minh của ổ đĩa đọc. Nếu nhận thấy ID là của track lân cận thì lại phải dịch chuyển đầu từ. Bằng cách này vòng phản hồi là đóng và hệ thống có thể đảm bảo việc bám sát đúng rãnh. Quá trình bám sát rãnh có một độ trễ được gọi là độ trễ ổn định (settling delay).

Các hệ thống cơ học có năng lượng hữu hạn và có độ đàn hồi, vì thế có độ trễ ổn định trên rãnh vào cỡ mi-li giây. Cùng với độ trễ ổn định, còn có độ trễ do chuyển đầu từ từ rãnh này sang rãnh khác cũng như chờ sector được chọn đọc/ghi chạy đến dưới đầu từ. Thành phần trễ nêu sau cùng đó, tính trung bình bằng một nửa thời gian một vòng quay, ở các ổ đĩa tốc độ 3600 vòng/phút bằng 8.3ms. Các ổ đĩa tốc độ 5400vòng/phút, thời gian đó giảm xuống còn 5.6ms nhưng phải trả giá là năng lượng tiêu thụ tăng lên nhiều. Với gia tốc đầu từ bằng 22g (g là gia tốc trọng trường) hoặc lớn hơn, thời gian trung bình để tới được một rãnh ngẫu nhiên là khoảng 1/3 thời gian để vượt qua toàn bộ các cylinder, khoảng 4ms hay 5ms. Thời gian ổn định khi chuyển đến nơi, chờ cho hết rung cần thêm khoảng 1ms nữa.

Tổng các thời gian trễ nêu trên được gọi là ***thời gian tìm - seek time***, độ lớn của nó vào khoảng 8-10ms, đó là với một thiết kế hiện đại. Khi mà các thành phần chủ yếu của ổ đĩa vẫn là cơ khí, khó có hy vọng cải thiện một cách đáng kể thời gian tìm - seek time.

8.4.5.4 Việc định vị đầu từ sát trên mặt đĩa và các sự cố đĩa cứng

Còn một vấn đề chưa được tìm hiểu kỹ, đó là điều khiển đầu từ theo chiều Z. Khi ổ đĩa không hoạt động, đầu từ tì vào mặt đĩa do sức căng đàn hồi tĩnh của chính cần mang đầu từ. Bằng cách đó, vị trí của đầu từ so với mặt đĩa không phụ thuộc

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

vào hướng đặt đĩa so với trọng trường. Chúng ta đã biết diện tích đĩa ứng với 1 bit chỉ vào khoảng 0.3μm x7μm, làm thế nào để có thể giữ được đầu từ đủ gần mặt đĩa để có thể lấy lên hoặc ghi xuống các tín hiệu từ trường biến đổi rất nhỏ trên một diện tích như vậy?

Các ổ đĩa cứng thông thường quay ở tốc độ từ 3600 vòng/phút đến 5400 vòng/phút, tốc độ đó không cho phép đầu từ quệt xuống bề mặt đĩa, đồng thời cũng không cho phép đầu từ tách xa vị trí bit mà nó đang đọc hoặc ghi thêm một phần nhỏ micro mét. Khoảng cách đó nhỏ hơn bước sóng của ánh sáng nhìn thấy, giữ một khoảng cách nhỏ đến như vậy một cách chính xác bằng cách nào?

Giải pháp cho vấn đề này là áp dụng ***hiệu ứng mặt đất (ground effect)***. Hiệu ứng này có thể thấy được rõ ràng khi một số loại máy bay nhỏ muốn hạ cánh trong khi bay ngang bên trên mặt đất ở tốc độ tương đối cao, nó không thể tiếp đất được nếu không giảm tốc độ. Người trên máy bay cảm thấy dường như có một quả bóng đàn hồi nằm dưới máy bay của họ. Thực tế là có, nhưng quả bóng đó là bằng không khí.

Trong đĩa cứng, đầu từ giống như đôi cánh máy bay và sức nâng tác dụng chống lại sức căng đàn hồi đang giữ đầu từ tì vào mặt đĩa. Dù cho mặt đĩa không phải là hoàn toàn phẳng, đầu từ cũng sẽ bám theo những đường hơi lồi lõm đó mà không có vấn đề gì. Ngày nay trong các ổ đĩa cứng, chiều cao điển hình của đầu từ so với mặt đĩa là 0.2μm, bằng khoảng nửa bước sóng ánh sáng màu da trời (blue).

Nếu trên mặt đĩa có lỗi hoặc các hạt bụi thì sao? Mặt đĩa phải hoàn toàn sạch ngay từ khi lắp ráp trong nhà máy và trong quá trình làm việc. Với khoảng cách 0.2μm một hạt bụi khói thuốc lá sẽ tương tự một hòn đá cao nửa mét trên đường cái ô tô. Mặc dầu đĩa cứng được lắp ráp trong điều điện cực kỳ sạch nhưng không thể đạt được sự sạch tuyệt đối. Nguyên nhân hỏng đĩa thường gặp nhất là bề mặt đĩa bị phá huỷ do bụi bẩn tích tụ trên đầu từ (lớp vật liệu sắt từ phủ trên mặt đĩa chỉ dày không quá 0.1μm, gấp khoảng 2000 lần đường kính 1 nguyên tử).

Chính vì vậy chúng ta luôn phải nhớ đến một câu cách ngôn đúng và nghiêm túc là:

□***Đừng có hỏi nếu đĩa cứng hỏng thì sao mà phải hỏi khi nào thì nó hỏng***□

Ngoài bụi bẩn, có thể kể ra một nguyên nhân khác có thể làm hỏng đầu từ hoặc mặt đĩa, đó là những lúc đĩa quay không đạt tốc độ tối đa. Đĩa cứng cần khoảng thời gian cỡ 1 phút để gia tốc, trong khoảng thời gian dài như vậy và trong khi đĩa không quay, tốt nhất là không để đầu từ tiếp xúc với vùng chứa dữ liệu trên mặt đĩa. Việc này thường được gọi là ***cho đầu từ hạ cánh (parking the head)***. Các bộ điều khiển đĩa (HDC) phải được thiết kế sao cho nó có thể cho đầu từ hạ cánh đúng đắn khi tắt nguồn điện, dù đó là sự tắt điện đúng trình tự hay là bị mất điện đột ngột. Ngay cả khi được thiết kế tốt, đầu từ đôi khi vẫn quệt xuống mặt đĩa trong khi đĩa đang quay với tốc độ tối đa.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Để bảo vệ đầu từ và mặt đĩa, người ta thường phủ một màng mỏng chống mài mòn và có cả tác dụng bôi trơn. Một trong các vật liệu như vậy là các bon vô định hình (amorphous carbon), nó có độ cứng gần bằng kim cương và hệ số ma sát với đầu từ rất nhỏ.

8.4.5.5 Một vài công nghệ đĩa cứng và các phương hướng chung để cải tiến

- **Công nghệ đĩa cứng Winchester**

Công nghệ đĩa cứng Winchester ra đời năm 1973, sử dụng động cơ tuyến tính hay động cơ bước với hệ thống servo có vòng mở hay khép kín để định vị đầu từ. Các ổ đĩa dung lượng lớn theo công nghệ này thường sử dụng đĩa servo.

Một đặc điểm quan trọng của công nghệ này là hệ thống các đĩa và đầu từ được đóng trong một vỏ kín. Đĩa cứng Winchester có các cỡ đĩa: 14", 8", 5"1/4.

Các hãng sản xuất khác nhau sử dụng hoặc cải tiến công nghệ này theo các hướng sử dụng nhất định.

- **Công nghệ đĩa cứng có đầu từ cố định**

Ngoài công nghệ Winchester và một số công nghệ khác người ta đã từng nghiên cứu và đưa vào sản xuất đĩa cứng có đầu từ cố định. Các đầu từ được gộp thành nhóm 8 hay 16 cái; mỗi rãnh có đầu từ riêng, loại bỏ được việc dịch chuyển đầu từ. Các ổ đĩa này có thời gian truy cập rất ngắn, nhưng dung lượng bị hạn chế do bị hạn chế về số lượng đầu từ.

- **Phương hướng chung để cải tiến**

1. Tăng số đĩa.
2. Tăng mật độ rãnh càng cao càng tốt.
3. Tăng mật độ tuyến tính (mật độ bit theo chiều dài track).
4. Cải tiến phương pháp mã hoá, sử dụng kỹ thuật nén số liệu.

8.4.5.6 Bộ điều khiển ổ đĩa cứng và các chuẩn

Những bước tiến trong công nghệ sản xuất đĩa cứng có liên quan tới 4 kiểu bộ điều khiển đĩa - HDC (Hard Disk Controller): ST506, ESDI, SCSI và IDE. Chúng được sử dụng rộng rãi và được sử dụng làm các chuẩn công nghiệp.

Khuôn dạng theo đó dữ liệu được cất trên đĩa không chỉ phụ thuộc vào bộ điều khiển mà còn phụ thuộc vào tốc độ truyền dữ liệu giữa máy tính và ổ đĩa. Bảng sau cho biết tốc độ truyền dữ liệu cực đại theo lý thuyết của các bộ điều khiển ổ đĩa cứng máy PC:

Tốc độ truyền đạt được trên thực tế phụ thuộc vào nhiều yếu tố khác nhau, chẳng hạn:

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Tốc độ của các chương trình giao tiếp với bộ điều khiển ở nhiều mức độ khác nhau: BIOS, DOS, chương trình ứng dụng.v.v.
- Tốc độ của bus.

Bảng 8-01 Tốc độ của một số bộ điều khiển đĩa

Tên bộ điều khiển	Tốc độ truyền dữ liệu cực đại
ST506	1 MB/sec
ESDI	2.5 MB/sec
IDE	4 MB/sec
SCSI	5 MB/sec

Chuẩn ST506 (ST - Seagate Technology)

Các ổ đĩa cứng được sử dụng rộng rãi trong các máy IBM PC đầu tiên được điều khiển bởi bộ điều khiển ST506 do hãng Seagate Technology đưa ra. Theo chuẩn ST506, ổ đĩa cứng (HDD) và bộ điều khiển ổ đĩa (HDC) là 2 thành phần tách biệt. Bộ điều khiển nằm trên một card mở rộng cắm vào 1 trong các slot trên mainboard. Mỗi bộ điều khiển có thể điều khiển được 2 ổ đĩa cứng. Các tín hiệu ghép nối HDD với HDC được chia làm 2 nhóm: nhóm các tín hiệu điều khiển - 34 dây và nhóm các tín hiệu số liệu - 20 dây.

Các ổ đĩa loại này nhãn thường được ghi "MFM/RLL", đó là chữ viết tắt của 2 phương pháp mã hoá dữ liệu mà đơn vị điều khiển đĩa ST506 sử dụng:

1. MFM: Nếu sử dụng phương pháp MFM thì có thể đạt tốc độ truyền dữ liệu 5Mbit/sec. Mỗi cylinder có thể được chia thành 17 sector.
2. RLL (Run Length Limited): Nếu sử dụng phương pháp RLL thì có thể đạt tốc độ truyền dữ liệu 7.5Mbit/sec. Mỗi cylinder có thể được chia thành thành 26 sector.

Khi máy IBM PC/XT mới xuất hiện, trên thị trường chỉ mới có bộ điều khiển ST506 và vì thế ROM-BIOS chỉ cung cấp các dịch vụ đĩa dựa trên các đặc tính kỹ thuật của bộ điều khiển này, cụ thể như sau:

1. Số ổ đĩa tối đa là 2.
2. Số từ trụ (cylinder) tối đa là 1024 (được đánh số 0..1023)
3. Số cung/rãnh (sector/track) tối đa là 63
4. Số đầu từ (head/side) tối đa là 16
5. Kích thước sector cố định là 512 bytes

Một ổ đĩa theo chuẩn ST506 nếu tính theo các giá trị cực đại nêu trên, có thể có dung lượng $=16*63*1024*512/2^{20}$ MB = 504MB.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Do chuẩn ST506 có những hạn chế về tốc độ trao đổi thông tin và về dung lượng ổ đĩa nên ngày nay người ta không sử dụng nữa.

Chuẩn ESDI (Enhanced Small Device Interface)

Tháng 12/1983 một số nhà sản xuất đĩa cứng và tủ băng từ đã thống nhất cải tiến chuẩn ST506 thành chuẩn ESDI, chuẩn này vẫn tương thích với chuẩn ST506 và vì vậy các máy tính mà ROM-BIOS chỉ hỗ trợ cho bộ điều khiển ST506 vẫn có thể làm việc với bộ điều khiển ESDI.

Trong cả 2 chuẩn ST506 và ESDI, dòng thông tin mà đầu từ đọc về (bit stream) bao gồm cả dữ liệu lẫn các tín hiệu điều khiển. Trong chuẩn ST506, dòng dữ liệu và các thông tin điều khiển đều cùng truyền trên bus dữ liệu. Trong bộ điều khiển ESDI có bộ nhớ đệm cho một sector, nhờ vậy có thể chọn hệ số đan xen (interleave) là 1:1. Về hệ số đan xen chúng ta sẽ so sánh hai chuẩn với nhau, ST506 yêu cầu hệ số đan xen từ 3 đến 6, nếu chọn bằng 6 có nghĩa là để đọc trọn vẹn một track đĩa cần phải quay đủ 6 vòng, nếu chọn bằng 3 - cần 3 vòng. Như vậy có khả năng tăng tốc độ tuyến tính từ 3 tới 6 lần nhờ giảm hệ số đan xen.

Trong chuẩn ESDI có một đặc điểm quan trọng khác với ST506 là, trên HDD theo chuẩn ESDI chỉ có dữ liệu thực sự mới được gửi về bộ điều khiển, các tín hiệu điều khiển được xử lý ngay trên HDD bằng một đơn vị logic có tên gọi là "***bộ tách dữ liệu***" **(data seperator)**, nó hoạt động song song với bộ điều khiển, nhờ vậy tốc độ truyền dữ liệu theo chuẩn này cao hơn chuẩn ST506. Hầu hết các bộ điều khiển ESDI làm việc với tốc độ truyền 10Mbit/sec, một số khác có thể đạt tốc độ 15, 20 thậm chí tới 24 Mbit/sec. Bộ tách dữ liệu được coi là một thành phần thông minh, hơn thế nữa, trong HDD loại ESDI còn được trang bị bộ nhớ các thông số vật lý của nó cũng như địa chỉ các sector hỏng, nhờ đó mà bộ điều khiển có thể thực hiện việc setup lại HDD.

Với các HDD loại ST506 số sector/track chỉ có thể là 17 hoặc 26; với các HDD loại ESDI con số tương ứng là 34 hoặc 36. Khi chúng ta sử dụng trình SETUP của BIOS để thiết lập thông số cho HDD loại ESDI, nếu thiết lập sai, chẳng hạn chúng ta khai báo 26 sector/track thì sẽ gây ra "mất" 1/3 dung lượng đĩa.

Nếu chúng ta sử dụng ổ đĩa ESDI với bộ điều khiển ESDI thì có thể tránh được vấn đề này. Khi chúng ta setup HDD ESDI, nếu khai báo số sector/track thấp hơn thực tế, do HDD ESDI có thể gửi các đặc tính vật lý của nó cho bộ điều khiển, nên bộ điều khiển có thể sử dụng một khả năng đặc biệt của nó là **"sector translation"** để chuyển đổi các thông số logic của ổ đĩa (thông số của BIOS) thành các thông số của ổ đĩa vật lý. Sự chuyển đổi này tốn thời gian nhưng đảm bảo cho ổ đĩa ESDI có thể sử dụng được với hầu hết các tham số truyền cho BIOS mà không gây "mất" dung lượng đĩa.

Khả năng chuyển đổi này phụ thuộc hoàn toàn vào bộ điều khiển ESDI. Một số bộ điều khiển ESDI chỉ hỗ trợ một số tham số của BIOS, cũng có những bộ điều khiển mềm dẻo hơn, có thể hỗ trợ mọi tham số truyền cho BIOS. Đó chính là ưu điểm nổi bật của chuẩn ESDI, nó giúp chúng ta vượt qua những giới hạn mà các

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

dịch vụ BIOS được xây dựng dựa trên chuẩn ST506 đặt ra. Thí dụ ổ đĩa ESDI có thể có trên 1024 cylinder; nếu dùng bộ điều khiển ST506 chúng ta sẽ không sử dụng được các cylinder có số hiệu trên 1023, còn với bộ điều khiển ESDI, nó sẽ mô phỏng số sector/track lớn hơn. Mặc dù HDD ESDI chỉ có thể có 34 sector/track nhưng nó làm như thể là có tới 63 sector/track, bộ điều khiển ESDI sẽ dịch tham số logic cylinder/sector/head thành thông số vật lý của ổ đĩa.

Chuẩn SCSI (Small Computer System Interface)

Tên viết tắt SCSI thường được phát âm "scuzzy". Về thực chất nó không phải là một chuẩn về giao diện với HDD mà là một chuẩn nêu lên cách nối tới 8 thiết bị hoàn toàn khác nhau với một máy PC. Ngoài HDD ra theo chuẩn này có thể nối thiết bị băng từ, ổ CD-ROM hoặc scanner .v.v. với một bộ giao diện SCSI.

Đặc điểm khác với các chuẩn nêu trên là chuẩn SCSI được sử dụng không những trong các máy họ IBM PC mà cả trong các hệ thống khác như Macintosh, Atari ST .v.v. nhờ việc mạch giao diện SCSI liên hệ với các thiết bị qua một bus được tách biệt với PC bus.

Tất cả các đặc tính đường truyền cũng như các lệnh để điều khiển thiết bị đều được chuẩn hoá. Các thiết bị SCSI có thể được sử dụng cho các hệ thống máy tính khác nhau, chỉ có bộ điều khiển SCSI là phải phù hợp với hệ thống máy tính mà nó làm việc.

Bus SCSI có 80 dây, trong đó có 8 dây truyền dữ liệu. Một version mới của SCSI có ký hiệu là SCSI II có 16 dây dữ liệu. Về lý thuyết tốc độ truyền của bus SCSI có thể đạt tới 4 hoặc 5MByte/sec, tuy vậy HDD thường không đạt được tốc độ này.

Chuẩn SCSI cũng phát triển theo khuynh hướng của ESDI là ngày càng tập trung nhiều đơn vị logic điều khiển trên HDD, làm cho bộ điều khiển trở nên độc lập với thiết bị, không cần quan tâm tới các đặc tính vật lý của thiết bị. Cũng giống như HDD ESDI, các HDD chuẩn SCSI cũng có bộ nhớ, HDD có thể gửi cho bộ điều khiển các thông số về đặc tính vật lý của nó khi được bộ điều khiển yêu cầu. Chính vì vậy bộ điều khiển đĩa SCSI có thể sử dụng các tham số logic của ổ đĩa không có trên thực tế.

Các hệ thống SCSI thường được trang bị BIOS riêng cho nó (ROM mở rộng) và không hỗ trợ các dịch vụ BIOS cho các ổ đĩa theo chuẩn ST506. Khi cài đặt thiết bị SCSI, các dịch vụ BIOS của nó sẽ thay thế các dịch vụ BIOS vốn có của hệ thống.

Chuẩn IDE (Intelligent Device Electronics)

Đây là một chuẩn mới tiên tiến về giao diện với thiết bị ổ đĩa cứng. Chuẩn này được các hãng Compaq kết hợp với Western Digital bắt đầu phát triển từ năm 1984, nhằm tạo ra một bộ điều khiển tương thích với ST506 nhưng choán ít chỗ trong máy.

Ổ đĩa IDE bao gồm cả ổ đĩa lẫn bộ điều khiển. Một cáp nối 40 dây trong đó có các dây truyền dữ liệu và cả các dây truyền tín hiệu điều khiển nối ổ đĩa IDE trực

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

tiếp với bus hệ thống, chính vì vậy đôi khi người ta gọi ổ đĩa IDE là ổ đĩa bus AT ("AT bus drive") mặc dù nó cũng có thể làm việc với bus XT. Một số loại máy có đầu nối với cáp của ổ IDE ngay trên mainboard. Các máy khác vẫn sử dụng một bảng mạch mở rộng, trên đó có đầu nối cáp của ổ IDE xuống bus hệ thống.

Việc kết hợp HDD với bộ điều khiển ngay trên HDD làm cho nó có tất cả các ưu điểm của bộ điều khiển SCSI, trong đó phải kể đến khả năng làm việc tốt với mọi khuôn dạng đĩa và khả năng "caching track" - đọc cả một track của đĩa và chứa vào cache bên trong nó để dùng cho yêu cầu đọc lần sau nếu cần. Ổ đĩa IDE có thể sử dụng hệ số đan xen là 1:1, nhờ vậy tốc độ truyền tăng.

Các ổ đĩa IDE tiêu thụ ít năng lượng, nên rất phù hợp với các máy laptop và notebook. Một số ổ IDE có chế độ tiết kiệm năng lượng, nó sẽ hoạt trong chế độ này theo lệnh do một trình điều khiển chuyên biệt hoặc do BIOS hỗ trợ nó phát ra.

8.4.6 Đĩa quang (optical disc, Compact Disc)

Ngày nay đĩa quang đã được sử dụng phổ biến, chúng có mật độ ghi thông tin cao hơn đĩa từ thông thường rất nhiều. Ban đầu các đĩa quang được nghiên cứu và phát triển nhằm ghi các chương trình truyền hình và âm nhạc, sau đó được sử dụng làm phương tiện nhớ ngoài cho máy tính.

Tín hiệu truyền hình có giải thông là 6MHz, trong thực tế thường giải thông 1 Hz có thể truyền thông tin với tốc độ 1 bps (bps - bit per second - bit/giây), vì vậy đĩa quang loại 1 giờ sẽ có dung lượng lý thuyết là $6*10^6*60*60 \approx 20$ Gbit ($\approx$2.5GByte). Các hệ thống thực tế đạt không quá 50% dung lượng lý thuyết này.

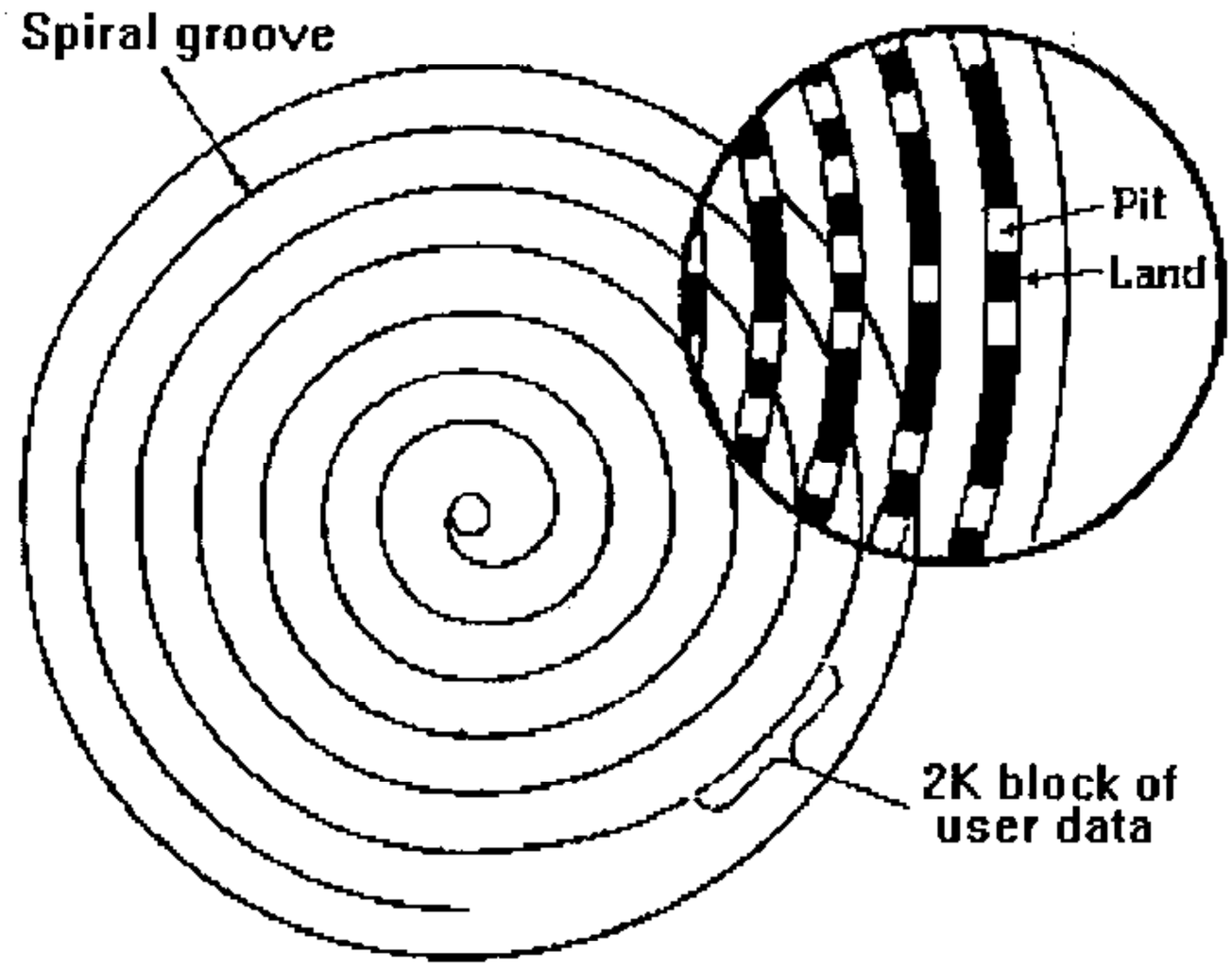
Vì có thể lưu trữ thông tin với dung lượng cực lớn nên đĩa quang được rất nhiều hãng đầu tư nghiên cứu và đã tiến hoá nhanh chóng đến mức người ta khó có thể hình dung nổi. Thế hệ đầu tiên đã được hãng Philips phát minh ra, sau đó Philips hợp tác với Sony để phát triển nó, các đĩa quang này dựa trên cùng một công nghệ được sử dụng trong Compact Disc để ghi âm và được gọi tên là CD ROMs (Compact Disc Read Only Memory).

8.4.6.1 Nguyên lý chế tạo (ghi)

Người ta chuẩn bị tạo ra các đĩa CD ROM bằng cách dùng một tia laser mạnh đốt chảy các hốc đường kính 1μm trên một đĩa chủ (master), từ đĩa chủ này sẽ tạo ra một cái khuôn để tạo ra các bản copy trên các đĩa chất dẻo, giống như việc người ta sản xuất các đĩa hát bằng nhựa trước đây. Sau đó người ta phủ một lớp nhôm mỏng lên trên mặt đĩa và lại phủ một lớp chất dẻo trong suốt lên trên lớp nhôm để bảo vệ. Lớp nhôm có tác dụng phản xạ tia laser.

Các hốc nhỏ được gọi là pit, còn diện tích không bị đốt nóng xung quanh được gọi là land, chúng có độ phản xạ khác nhau, nhờ đó có thể phân biệt được pit và land. Xem hình 8-25.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 8-25 Nguyên lý chế tạo CD-ROM

8.4.6.2 Tổ chức thông tin

Thông tin trên CD-ROM được ghi theo một đường xoắn ốc duy nhất. Dữ liệu được ghi thành từng nhóm 24 byte, đầu tiên mỗi byte được mở rộng từ 8 bit thành 14 bit bằng việc sử dụng mã sửa sai Reed-Solomon. Ba bit đặc biệt được thêm vào giữa các nhóm và một byte đồng bộ được bổ sung để tạo thành 1 frame. Một nhóm 98 frame tạo thành 1 block chứa 2KB dữ liệu của người sử dụng, block là đơn vị cơ bản địa chỉ hoá được. Mỗi CD-ROM có thể chứa 270.000 block, cho dung lượng là 553 MB.

Mặc dù cách ghi thông tin này là phức tạp và gây sự lãng phí lớn dung lượng đĩa, nhưng nó tạo cho ta một độ tin cậy cao với một môi trường ghi thông tin rẻ tiền.

8.4.6.3 Cách đọc thông tin

Các đĩa CD-ROM được đọc bằng một thiết bị giống như máy nghe nhạc CD, một đầu dò (detector) đo năng lượng phản xạ từ bề mặt đĩa khi chiếu lên đó một tia laser công suất nhỏ. Dữ liệu được đọc với một vận tốc tuyến tính không thay đổi là 75 inches/sec, cho ta tốc độ đọc dữ liệu là 153.60 kbytes/sec.

8.4.6.4 Ưu điểm, nhược điểm và các biện pháp khắc phục

- **Ưu điểm**

CD-ROM được sản xuất bằng phương pháp rập khuôn chứ không giống như việc ghi thông tin lên đĩa mềm, làm cho giá thành sản xuất một đĩa CD-ROM rất rẻ khi sản xuất hàng loạt. Việc đọc sử dụng tia laser không cần sự tiếp xúc cơ khí, hơn nữa bền mặt đĩa được phủ lớp bảo vệ nên các đĩa quang có độ tin cậy rất cao so với đĩa mềm và nhiều thiết bị nhớ ngoài khác.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Vì có dụng lượng rất lớn, độ tin cậy cao và giá rẻ nên CD-ROM được sử dụng rộng rãi để chứa các cơ sở dữ liệu lớn, đặc biệt là các cơ sở dữ liệu có chứa văn bản và hình ảnh. Cả một bộ luật hay một bộ bách khoa toàn thư có thể để trên một CD-ROM.

- **Nhược điểm và các biện pháp khắc phục**

Việc rập khuôn các đĩa bằng chất dẻo được phủ nhôm khó đạt được độ chính xác cao và đồng đều nên các thông tin số thường có chứa nhiều lỗi.

Để khắc phục tình trạng lỗi người ta sử dụng 2 biện pháp sau:

1. Thứ nhất, đầu đọc trong ổ đĩa có một gương chính xác, được điều khiển bởi một cơ cấu servo nhằm bám sát (tracking) bề mặt đĩa, bù trừ các lỗi do sản xuất gây ra.
2. Thứ hai, sử dụng một loại mã sửa sai phức tạp có tên là **Crossinterleaved Reed-Solomon** trong việc ghi dữ liệu lên đĩa. Mã này sử dụng nhiều bit hơn mã Hamming, có hiệu quả sửa lỗi cao hơn.
3. Ngoài ra, thay cho việc sử dụng một hốc (pit) để biểu diễn 0, và land để biểu diễn 1 (hay ngược lại), người ta sử dụng một cặp chuyển đổi pit-land hay land-pit để biểu diễn bit 1, khoảng cách giữa hai sự chuyển đổi cho biết có bao nhiêu bit 0 nằm giữa 2 bit 1.

8.4.6.5 Các đĩa quang ghi được một lần (WORM) và nhiều lần

Các CD-ROM loại thông thường như mô tả ở trên là không thể ghi được, sau này người ta đã chế ra các ***đĩa quang "thế hệ thứ hai" là WORM*** (Write Once Read Many - Ghi một lần, đọc được nhiều lần). Thiết bị này cho phép người sử dụng tự mình ghi thông tin lên các đĩa quang, tuy nhiên sau khi đã tạo ra các hốc (pit) rồi thì không thể thay đổi được nữa.

Các ***đĩa quang "thế hệ thứ ba"*** là những môi trường quang học có thể xoá được, chúng sử dụng công nghệ quang-từ (magneto - optical). Đĩa chất dẻo được phủ một lớp hợp kim, hợp kim này có tính chất đáng chú ý là ở nhiệt độ thấp chúng không nhậy cảm với từ trường, nhưng ở nhiệt độ cao cấu trúc phân tử của chúng sắp xếp lại theo từ trường tác động vào.

Để sử dụng tính chất này vào việc ghi thông tin, đầu ghi của ổ đĩa có một laser và một nam châm. Tia laser chiếu một xung ánh sáng ngắn vào kim loại, nâng nhiệt độ của nó tức thời lên cao, nhưng chưa đủ làm chảy thành hốc (pit), đồng thời nam châm phát ra một từ trường theo một trong hai hướng để sắp xếp lại các phần tử của chất có tính chất quang từ. Khi xung laser hết, chỗ kim loại bị chiếu tia laser đã bị từ hoá theo một trong hai hướng biểu diễn bit 0 hoặc 1. Thông tin này được đọc lại như với các CD-ROM thông thường bằng cách sử dụng một tia laser yếu hơn rất nhiều.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Đĩa CD-ROM loại này có thể xoá đi viết lại như lần viết đầu tiên. Tuy nhiên loại đĩa CD-ROM này hiện nay cũng không thể thay thế các ổ đĩa cứng Winchester vì hai lý do chính:

- Thời gian tìm rãnh (seek time) lớn hơn ở các ổ đĩa Winchester một bậc
- Tốc độ đọc dữ liệu cũng chậm hơn một bậc.

8.5 MÁY IN - PRINTER

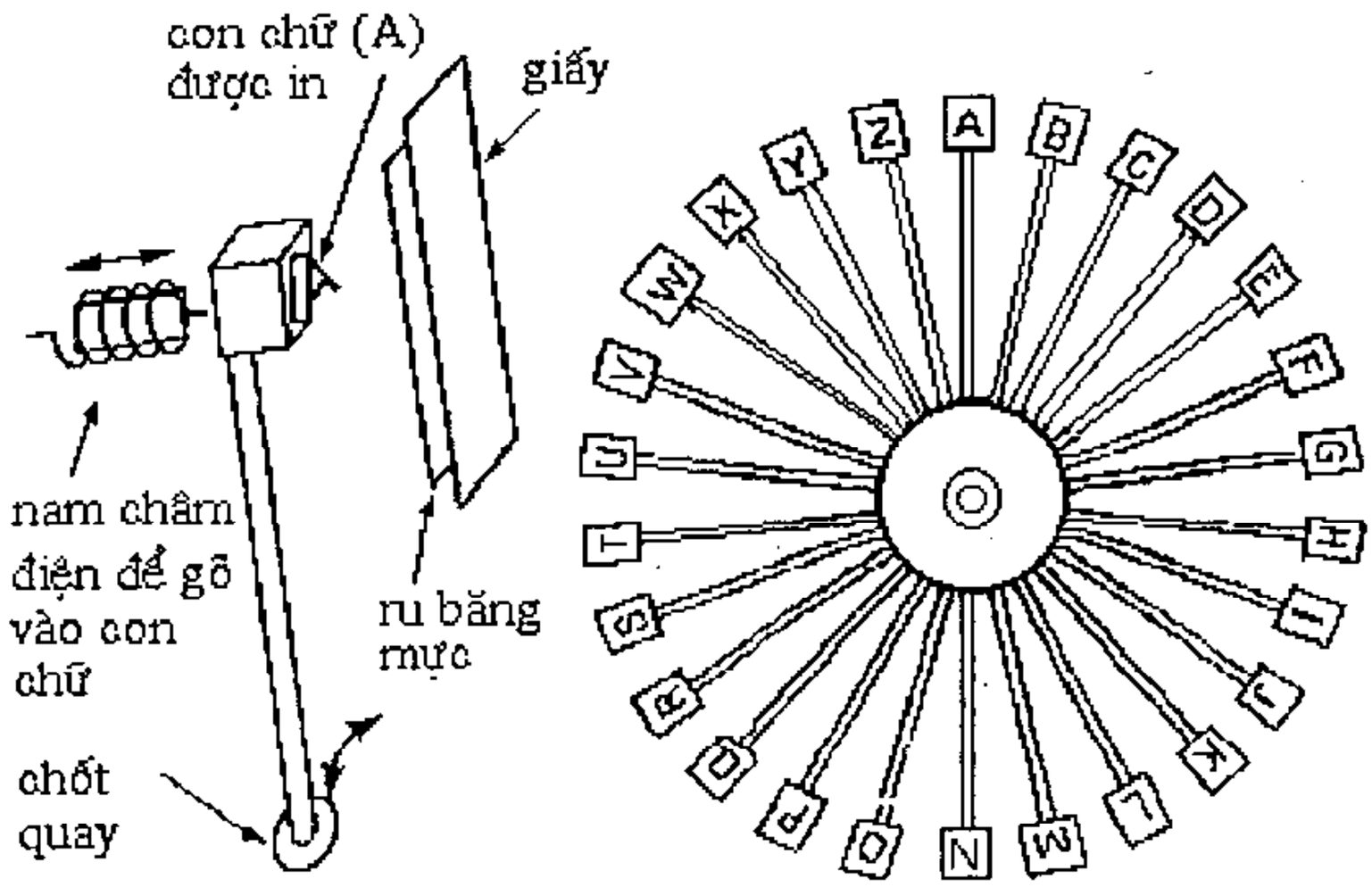
Máy in là thiết bị điện tử-cơ khí, tạo ra được các thông tin có thể đọc được trên giấy trong quá trình đưa số liệu ra.

Các kỹ thuật mới đã đem lại những hiệu quả to lớn trong sự phát triển của máy in. Những tiến bộ về công nghệ trong lĩnh vực cơ khí và điện tử đã biến chiếc máy in cồng kềnh ngày trước thành một thiết bị nhỏ gọn có thể để trên bàn, thậm chí có cả những máy in có thể xách theo tay nhẹ nhàng. Có thể phân máy in thành 2 loại chính là:

1. máy in tiếp xúc (impact printer): tạo nên các ký tự bằng tiếp xúc cơ học với băng mực đặt trên giấy, ví dụ: máy in búa, máy in kim ...
2. máy in không tiếp xúc (nonimpact printer) ví dụ máy in phun mực, máy in laser...

8.5.1 Máy in búa

Các con chữ thường được bố trí như một bánh xe, trông giống hình bông hoa cúc, nên còn có tên gọi là "Máy in bánh xe bông cúc" (Daisy wheel printer). Bánh xe và búa được mang trên một cơ cấu mang, có thể trượt ngang giống như trong máy in kim. Loại máy in này cho chất lượng in tốt, chữ nét liền.



Hình 8-26 Nguyên lý cấu tạo của máy in búa

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Nhược điểm: mô tơ phải quay bánh xe để định vị ký tự đúng giữa búa và băng mực. Sau đó búa gõ xuống 'cánh hoa' (con chữ) làm con chữ in lên băng mực và giấy để tạo nên một ký tự, chính vì vậy mà tốc độ chậm, ngoài ra nó có một nhược điểm lớn là không có khả năng in đồ hoạ.

8.5.2 Máy in dòng

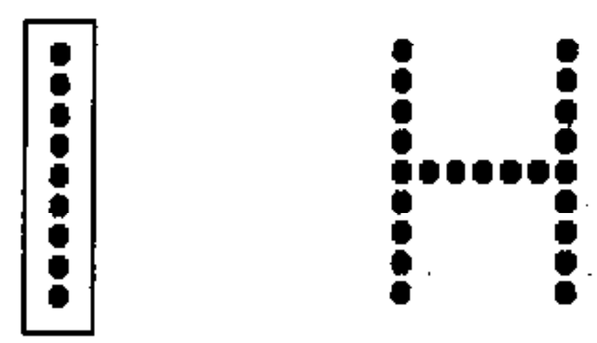
Sử dụng một trống hình trụ có thể quay được, trên đó có gắn các bộ con chữ. Băng mực được đặt giữa mặt giấy cần in lên và trống chữ, thiết bị điều khiển sẽ chọn con chữ cần thiết tại mỗi vị trí trong một dòng và làm cho nó mổ vào giấy.

Ưu điểm: Tốc độ in rất cao, từ vài trăm tới 1000 dòng/ 1 phút.

Nhược điểm: font chữ nghèo nàn, người sử dụng không thể can thiệp được.

8.5.3 Máy in kim

Thay cho việc sử dụng bộ chữ có sẵn như ở máy in dòng, loại máy này sử dụng một đầu in (Printer head), trong đầu in có một số kim bố trí thành cột. Khi đầu in quét theo một dòng trên giấy thì các kim trên "Ma trận cột" đó sẽ mổ xuống giấy, tổ hợp kim nào sẽ mổ xuống phụ thuộc vào ký tự sẽ được in và cột nào của ký tự đó.



a/ Printer head b/ Dot matric

Hình 8-27 a/ Đầu in; b/ Ký tự là một ma trận điểm

Tốc độ máy in kim khá nhanh so với máy in búa, nhưng chữ không liền nét. Một số loại máy in kim đạt chất lượng gần như chữ liền bằng cách in nhiều chấm hơn trong mỗi ký tự. Phần lớn các máy in kim đều có khả năng in đồ hoạ, nhờ có cơ cấu kim như mô tả, cộng với bộ phận điều khiển định vị cơ khí rất chính xác. Các máy in kim có thể in một lần từ một đến 5 bản nhờ kẹp giấy than, khi đó phải điều khiển khoảng cách giữa đầu in và giấy.

8.5.4 Máy in laser

Máy in laser sử dụng phương pháp chụp hình điện tử các hình ảnh bằng nguồn sáng laser. Máy in laser cũng là sự phát triển tiếp theo của máy photocopy thông thường. Các đèn và mặt thuỷ tinh của máy photocopy được thay bằng một chùm tia laser quét đã được điều chế. (xem hình 8-28)

Như hình vẽ minh hoạ, nguồn sáng laser được điều chế bởi một cửa chắn âm-quang học; cửa này tắt và mở chùm tia laser bằng các tín hiệu điện tử. Chùm tia laser đã được điều chế chiếu vào một gương quay nhiều cạnh. Gương quay sẽ làm cho chùm laser, sau khi hội tụ qua thấu kính sẽ quét qua mặt của trống cảm quang. Đây là trống dẫn điện, trên mặt phủ một lớp vật liệu quang dẫn như Sulfide cadmium hoặc chất selen. Trước khi quay qua chỗ có thể bị tia laser chiếu vào, toàn bộ trống đã được nạp một điện áp tĩnh điện. Sau đó, những chỗ bị tia laser chiếu

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

trực tiếp vào, điện áp tĩnh điện sẽ giảm bớt. Hình ảnh điện tích sẽ lưu lại trên bề mặt trống cảm quang.

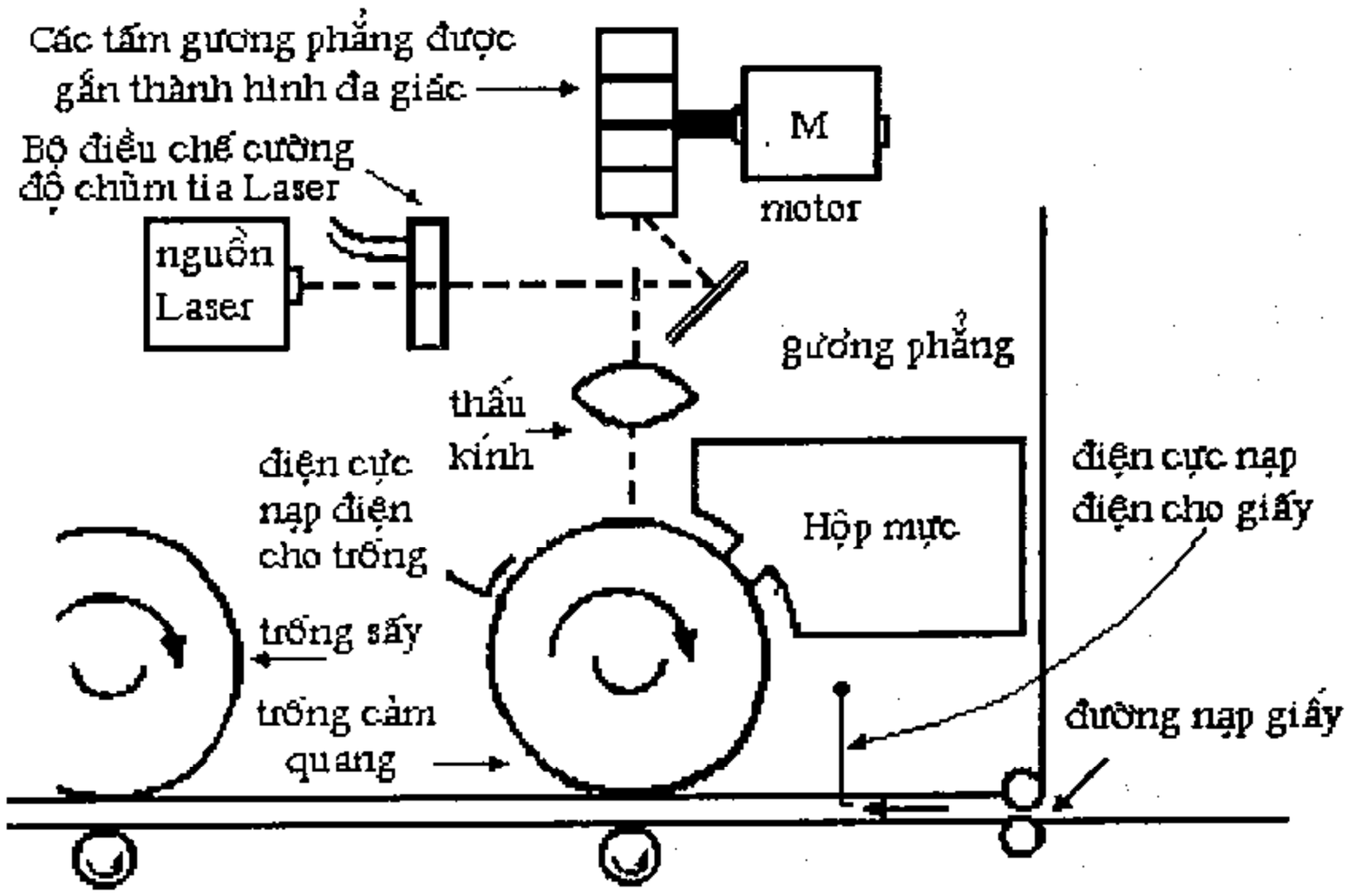
Gương nhiều cạnh quay liên tục tạo nên các đường quét ngang của tia laser trên mặt trống, còn trống lại quay với tốc độ chậm hơn nhiều, kết quả là hình thành một mành các đường ngang, giống như mành trên màn hình CRT.

Lớp mực bột đã bị làm nhiễm tĩnh điện từ trước sẽ áp vào trống. Bột mực tích điện chỉ bắt vào các chỗ tích điện trái dấu trên mặt trống, do vậy trên mặt trống có hình ảnh tạo bởi các hạt mực.

Tiếp theo, mặt giấy cũng đã được làm nhiễm tĩnh điện trái dấu với bột mực được áp vào dưới trống, với cùng tốc độ quay của trống. Hình ảnh tạo bởi các hạt mực trên trống sẽ được chuyển sang giấy vì nó tích điện trái dấu cao hơn.

Tờ giấy mang hình ảnh tạo bởi các hạt mực được quay qua trống sấy nóng, ở đây bột mực chảy ra và khô ngay, in cố định lên mặt giấy.

Các mạch điện tử dùng để điều chế tia laser cũng tương tự như mạch điện tử điều chế dòng điện tử của ca-tốt CRT đã trình bày trước đây.



Hình 8-28 Nguyên lý hoạt động của máy in Laser

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CHƯƠNG 9 MÁY VI TÍNH IBM PC

9.1 GIỚI THIỆU

9.1.1 Sự ra đời của máy IBM PC

Ý tưởng về việc sản xuất các máy tính kích thước nhỏ, dành cho một người sử dụng, có thể để được trên bàn có từ rất sớm. Đã có nhiều hãng sản xuất máy tính đưa ra thị trường các máy như vậy trước khi hãng IBM (International Business Machine Company) đưa ra thị trường thế hệ máy vi tính đầu tiên của mình là IBM PC (Personal Computer).

Trước thế hệ IBM PC, hãng IBM đã sản xuất loại máy System/23 DataMaster, tuy nhiên loại máy này sử dụng bộ vi xử lý 8-bit 8085 của hãng Intel lúc này đã bị coi là lạc hậu. Năm 1980 hãng Intel đưa ra thị trường bộ vi xử lý 16 bit 8086 có nhiều tính năng hơn hẳn loại 8085, hãng IBM bắt đầu tiến hành thiết kế loại máy tính mới của mình là IBM PC sử dụng chip vi xử lý 8086, mở đầu một cuộc cách mạng về máy tính cá nhân.

9.1.2 Họ 80xx của hãng Intel

Các thế hệ kế tiếp nhau của dòng họ IBM PC đều sử dụng các bộ vi xử lý của hãng Intel. Hình 9-01 dưới đây là dòng họ 80xx của Intel.

Bộ vi xử lý 8086 được sử dụng trong IBM PC, 8088 được sử dụng trong IBM PC/XT. Các thế hệ 80186, 80286, 80386, 80486 ... được sử dụng trong các máy AT ra đời sau này.

9.1.2.1 Bộ vi xử lý 8086 và 8088

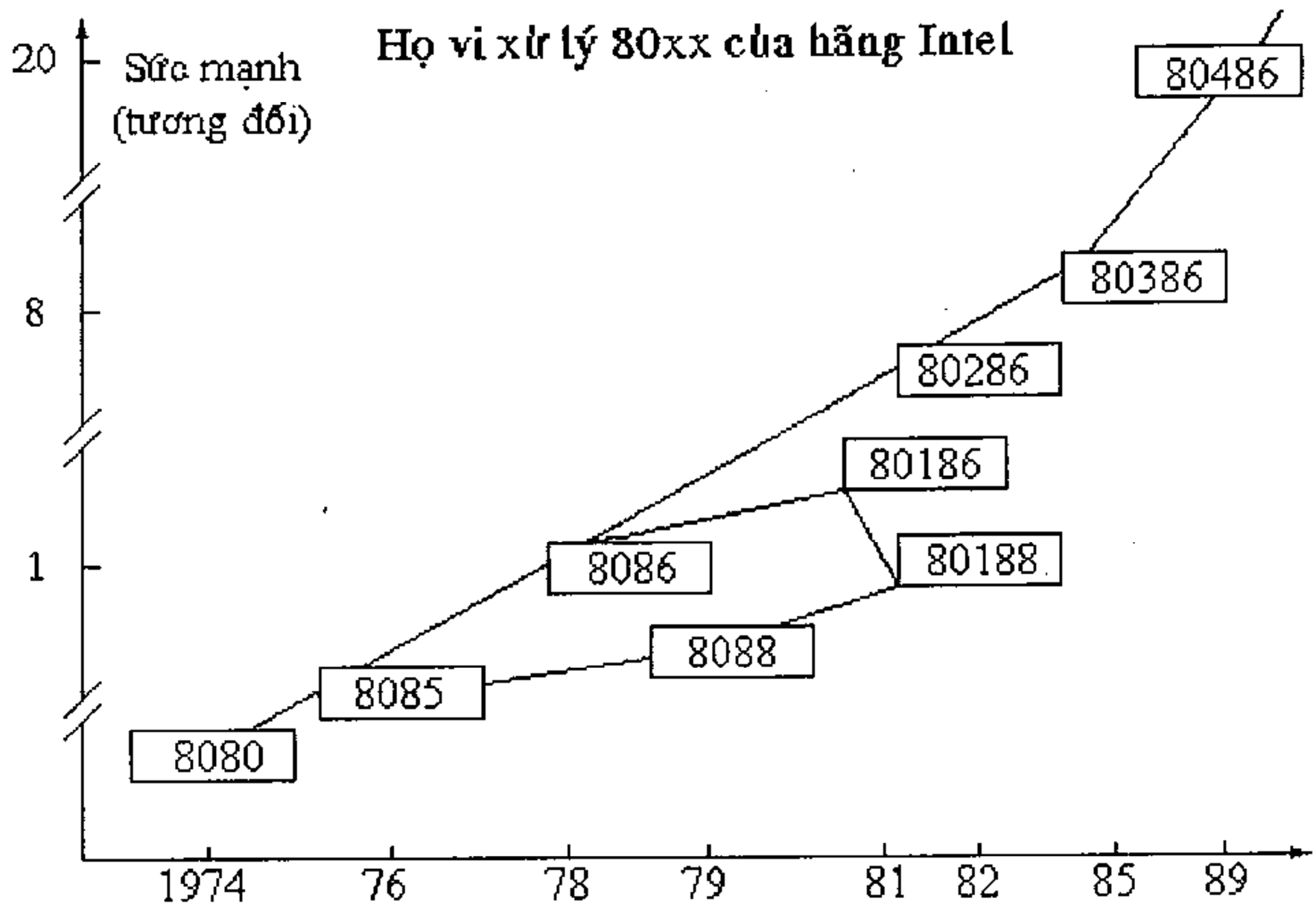
Các đặc tính kỹ thuật chủ yếu

Số thanh ghi:	14 thanh ghi 16 bit
Bus địa chỉ:	20 dây
Bus số liệu:	16 ở 8086 và 8 dây ở 8088
Tập lệnh:	115 lệnh (đây là số lệnh được công bố trong nhiều tài liệu)
Tốc độ chuẩn:	4.77MHz
Số chân:	40

- **Chip 8086**

Chip 8086 bắt đầu được phát triển từ năm 1978 và đưa ra thị trường năm 1980. Đây là bộ vi xử lý 16-bit, các thanh ghi bên trong là 16-bit và nó xử lý 16-bit số liệu cùng một lúc; 8086 liên hệ với kênh số liệu bên ngoài bằng bus dữ liệu 16-bit, bus địa chỉ 20 bit.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 9-01 Họ vi xử lý 80xx của hãng Intel

- **Chip 8088**

Chip 8088 ra đời sau 8086, nó có cấu trúc bên trong và tập lệnh hoàn toàn giống 8086, chỉ khác 8086 ở kênh truyền số liệu với thế giới bên ngoài. 8088 sử dụng bus dữ liệu 8-bit nên việc truyền dữ liệu giữa nó và bộ nhớ chậm hơn so với 8086. Đây thật sự là một bước lùi về kỹ thuật nhưng đem lại nhiều lợi ích về kinh tế. Tại thời điểm đó, bus DataMaster 8-bit đang được sử dụng rộng rãi, nhiều loại card mở rộng và các chip hỗ trợ sẵn có trên thị trường là loại 8-bit nên giá thấp hơn loại 16-bit tương ứng. Việc sử dụng kênh dữ liệu 8-bit giúp cho người sử dụng khi nâng cấp máy có thể tận dụng các card 8-bit trong máy cũ, đồng thời cũng giúp cho những nhà sản xuất các máy tương thích với IBM PC có thêm nhiều cơ hội lựa chọn sản phẩm của các hãng khác nhau. Chiến lược phát triển có tính kế thừa như vậy đã góp phần làm cho khách hàng của IBM ngày một gia tăng.

- **Sơ đồ khối của bộ vi xử lý 8086 và 8088** (hình 9-02)

Quá trình hoạt động của bộ vi xử lý thực tế là việc thực hiện lặp đi lặp lại 2 thao tác chính: lấy lệnh (Fetch) và thực hiện lệnh (Execute). Cấu trúc của bộ vi xử lý cũng có thể được chia ra thành 2 đơn vị chính: EU (Execution Unit) - là đơn vị thực hiện lệnh và BIU (Bus Interface Unit) - là đơn vị giao tiếp bus.

Đơn vị EU

Đơn vị EU của 8086 và 8088 giống nhau, bao gồm ALU, thanh ghi cờ, các thanh ghi đệm và các thanh ghi đa năng. Các kênh truyền dữ liệu bên trong EU đều là 16 bit.

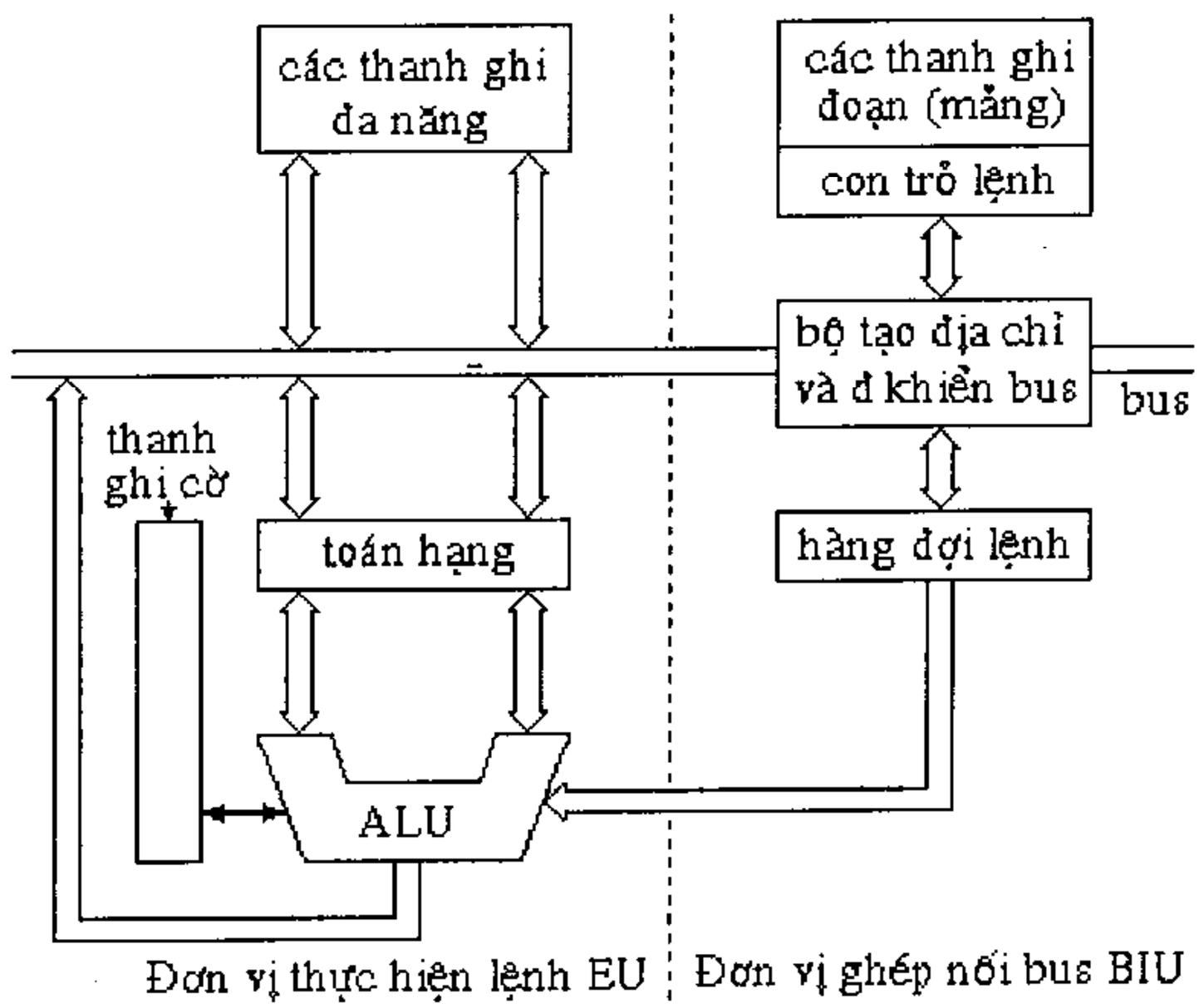
127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

EU không nối trực tiếp với thế giới bên ngoài, nó nhận lệnh từ hàng đợi lệnh bên trong BIU. Nếu lệnh cần truy cập bộ nhớ hoặc cổng I/O (thiết bị ngoại vi) thì EU yêu cầu BIU lấy hoặc gửi dữ liệu. Tất cả các địa chỉ mà EU thao tác đều là 16 bit, khi gửi sang cho BIU thì BIU sẽ sắp đặt lại để tạo thành địa chỉ vật lý 20 bit phát ra các chân ra địa chỉ của chip.

Đơn vị BIU

Đơn vị BIU thực hiện tất cả các thao tác với bus mà EU yêu cầu, ngoài ra khi EU đang thực hiện một lệnh, BIU có thể lấy lệnh từ bộ nhớ về đặt trong hàng đợi lệnh. Hàng đợi lệnh của 8086 dài 6 byte, của 8088 dài 4 byte. Việc lấy lệnh của BIU và việc thực hiện lệnh của EU được thực hiện song song với nhau (trừ một số trường hợp ngoại lệ) làm cho năng suất làm việc của bộ vi xử lý tăng lên.

Đơn vị BIU của 8086 và 8088 về chức năng thì giống nhau nhưng sự làm việc có những điểm khác nhau, do bus dữ liệu của chúng không giống nhau.



Hình 9-02 Sơ đồ khối của bộ vi xử lý 8086/8088

- **Các thanh của 8086/8088**

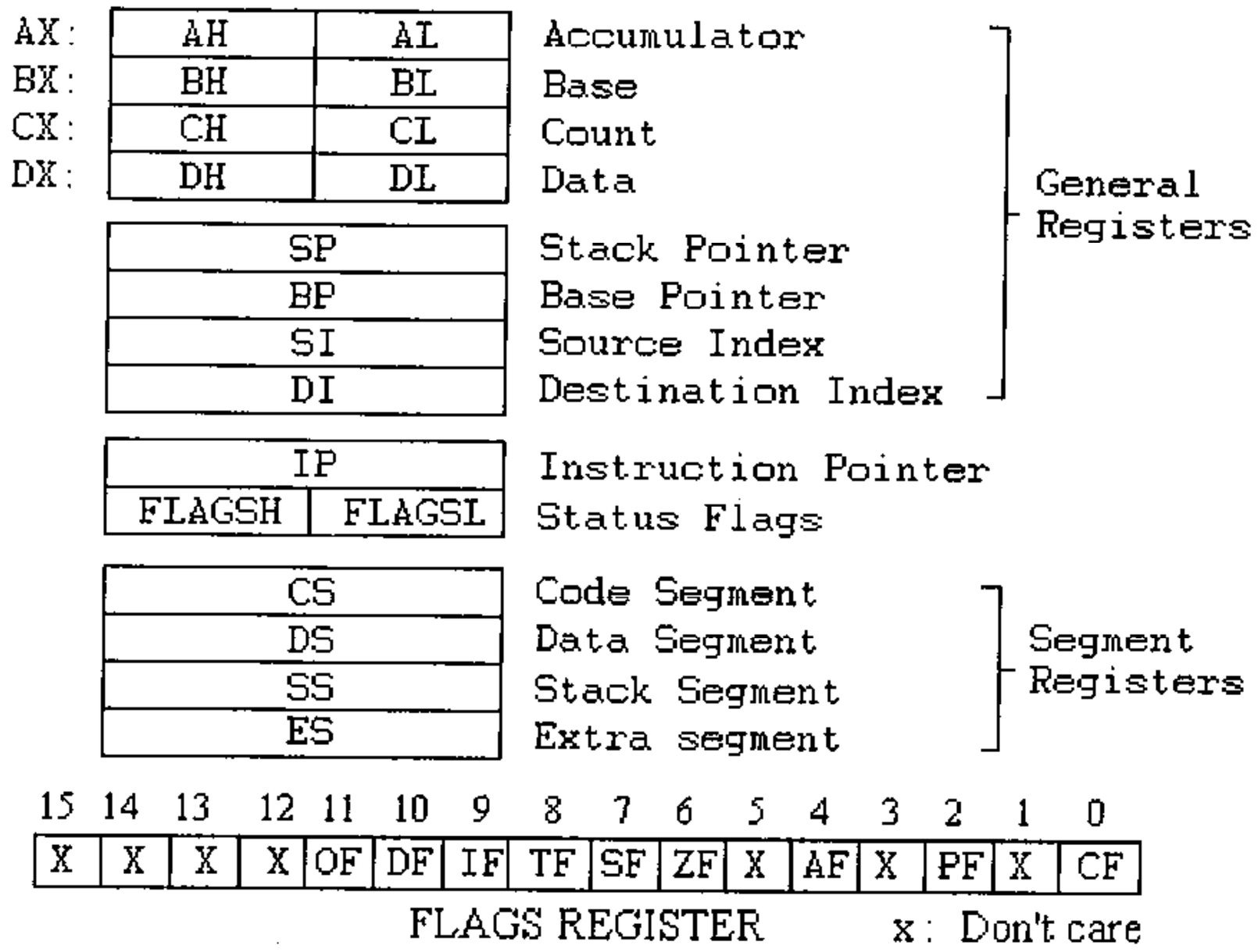
Mô hình các thanh ghi của bộ vi xử lý 8086/8088 được trình bày trên hình 9-03. Có thể chia 14 thanh ghi trên thành 4 nhóm như sau:

1) Các thanh ghi đa năng: AX, BX, CX, DX

Đây là các thanh ghi nhiều công dụng 16 bit nhưng có thể sử dụng từng nửa 8 bit của chúng như những thanh ghi độc lập. Thí dụ thanh ghi AX gồm 2 nửa AH và AL; khi sử dụng AX như một thanh ghi 16 bit thì nửa AL sẽ chứa byte thấp, AH chứa byte cao của một con số 16 bit trong các thao tác số.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Khi thực hiện một số lệnh, 8086/8088 tự động sử dụng một số thanh ghi đa năng vào các mục đích định trước.



Hình 9-03 Các thanh ghi của bộ vi xử lý 8086/8088

1) Các thanh ghi địa chỉ đoạn CS, DS, ES, SS

CS (Code Segment): là thanh ghi địa chỉ đoạn mã lệnh, chứa địa chỉ cơ sở (địa chỉ đoạn) của chương trình đang được thực hiện.

DS (Data Segment): là thanh ghi địa chỉ đoạn dữ liệu, chứa địa chỉ đoạn của vùng dữ liệu mà chương trình đang được thực hiện sử dụng. Vùng này thường chứa các biến của chương trình.

ES (Extra Segment): là thanh ghi địa chỉ đoạn dữ liệu bổ sung mà chương trình đang được thực hiện sử dụng. Vùng này cũng thường chứa các biến của chương trình.

SS (Stack Segment): là thanh ghi địa chỉ đoạn bộ nhớ ngăn xếp (stack) của chương trình đang chạy.

3) Các thanh ghi con trỏ và chỉ số: IP, SP, BP, SI, DI

IP (Instruction Pointer): con trỏ lệnh, nó luôn trỏ tới lệnh sắp được thực hiện trong đoạn bộ nhớ do CS đang trỏ tới, như vậy thực tế thì cặp CS:IP mới là con trỏ lệnh vì nó chứa địa chỉ đầy đủ của một lệnh trong bộ nhớ. Sau khi thực hiện xong mỗi lệnh, bộ vi xử lý thay đổi IP để nó trỏ tới lệnh kế tiếp sẽ được thực hiện.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

SP (Stack Pointer): con trỏ stack, nó luôn kết hợp với SS thành cặp SS:SP để trỏ tới đỉnh tức thời của stack. Sau mỗi thao tác cất một word vào stack (thao tác PUSH), SP được tự động giảm 2 đơn vị, còn sau thao tác lấy một word khỏi stack (thao tác POP), SP được tự động tăng 2 đơn vị.

BP (Base stack Pointer): là thanh ghi con trỏ để đánh dấu stack (Stack Marker). Trong nhiều thao tác với stack cần sử dụng thêm thanh ghi này, chẳng hạn việc truyền tham số thông qua stack.

SI và DI (Source Index, Destination Index): con trỏ chỉ số nguồn và con trỏ chỉ số đích. Chúng được thiết kế để chuyên sử dụng trong các thao tác vận chuyển số liệu. Trong một số lệnh của 8086/8088 các thanh ghi này luôn kết hợp với các thanh ghi đoạn DS và ES để trỏ tới "nguồn" và "đích" của thao tác vận chuyển dữ liệu, đó là các cặp DS:SI và ES:DI.

4) Thanh ghi cờ (Flags register)

Đây là thanh ghi 16 bit, mỗi bit được sử dụng để thể hiện một trạng thái của bộ vi xử lý tại một thời điểm nhất định trong quá trình thực hiện chương trình, trong bộ vi xử lý 8086/8088 mới chỉ có 9 bit được sử dụng, người ta gọi mỗi bit đó là một cờ. Giá trị của mỗi cờ thường được biểu diễn bằng các ký hiệu gợi trí nhớ như cách biểu diễn của chương trình DEBUG.EXE của DOS.

Trong phần trình bày về cờ ở dưới đây, giá trị thứ nhất của cờ tương ứng với giá trị 1 của bit, còn giá trị thứ hai tương ứng với giá trị 0 của bit. Khi nói cờ giương lên có nghĩa là bit =1.

CF (Carry Flag): cờ nhớ, có 2 giá trị là CY (CarrY) và NC (No Carry). Cờ này giương lên nếu số nhớ tràn khỏi bit có trọng số lớn nhất, thường được sử dụng khi thực hiện các phép cộng, trừ các số nhiều byte.

PF (Parity Flag) : cờ chẵn lẻ, có 2 giá trị là PE (Parity Even) và PO (Parity Odd). Cờ này nhận giá trị PE/PO khi byte số liệu nhận hoặc gửi qua một cổng có số bit 1 là chẵn/lẻ.

AF (Auxiliary Flag): cờ bổ trợ, có 2 giá trị là AC (Auxiliary Carry) và NA (No Auxiliary carry). Cờ này giương lên (AC) nếu có số nhớ hoặc số mượn vượt ra khỏi 4 số có trọng số nhỏ nhất.

ZF (Zero Flag): cờ zêro, có 2 giá trị là ZR (ZeRo) và NZ (Non-Zero). Khi kết quả của phép tính bằng 0 thì cờ ZF = ZR.

SF (Sign Flag): cờ dấu, có 2 giá trị là NG (NeGative) và PL (PLus). Trong 8086/8088 các số âm được biểu diễn dưới dạng bù 2, nên phải dùng cờ SF để chỉ thị dấu của kết quả.

TF (Trap - Single step Flag): cờ chạy từng bước. Nếu TF=1 thì bộ vi xử lý sẽ chạy từng bước để có thể gỡ rối chương trình, sau mỗi lệnh bộ vi xử lý sẽ phát ra một lệnh ngắt (INT) để có thể kiểm tra chương trình.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

IF (Interrupt Flag): cờ ngắt, có 2 giá trị là EI (Enable Interrupt) và DI (Disable Interrupt). Nếu đặt cờ IF=EI thì bộ vi xử lý cho phép ngắt; nếu đặt IF=DI thì cấm được các ngắt loại có thể che (maskable).

DF (Direction Flag): cờ hướng, có 2 giá trị là DN (DowN) và UP. Nếu DF=UP thì các lệnh vận chuyển miền dữ liệu sẽ theo hướng địa chỉ tăng dần, ngược lại thì sẽ được tiến hành theo hướng địa chỉ giảm dần.

OF (Overflow Flag): cờ tràn, có 2 giá trị là OV (OVerflow) và NV (Non OVerflow). Cờ này giương lên khi có hiện tượng tràn số học. Đó là trường hợp một số bị mất vì số thành có cỡ quá lớn, vượt quá khỏi dung lượng nơi được gửi tới.

- **Việc đánh địa chỉ phân đoạn**

Bộ vi xử lý 8086/8088 có bus địa chỉ 20 bit (đánh địa chỉ được 1MB ô nhớ), trong khi đó các thanh ghi đều là 16 bit. Trong sự hoạt động của các chương trình, địa chỉ cũng là các con số và có thể tính toán với nó. Khi thực hiện tính toán, các địa chỉ cũng được chứa trong các thanh ghi 16 bit như mọi con số 16 bit thông thường khác. Để tạo ra tín hiệu địa chỉ 20 bit ở các chân ra địa chỉ của bộ vi xử lý (địa chỉ vật lý), người ta sử dụng ***giải pháp đánh địa chỉ phân đoạn*** (Segmental Addressing), theo phương pháp này, 2 thanh ghi chứa địa chỉ 16 bit được kết hợp với nhau như minh hoạ ở hình 9-04.

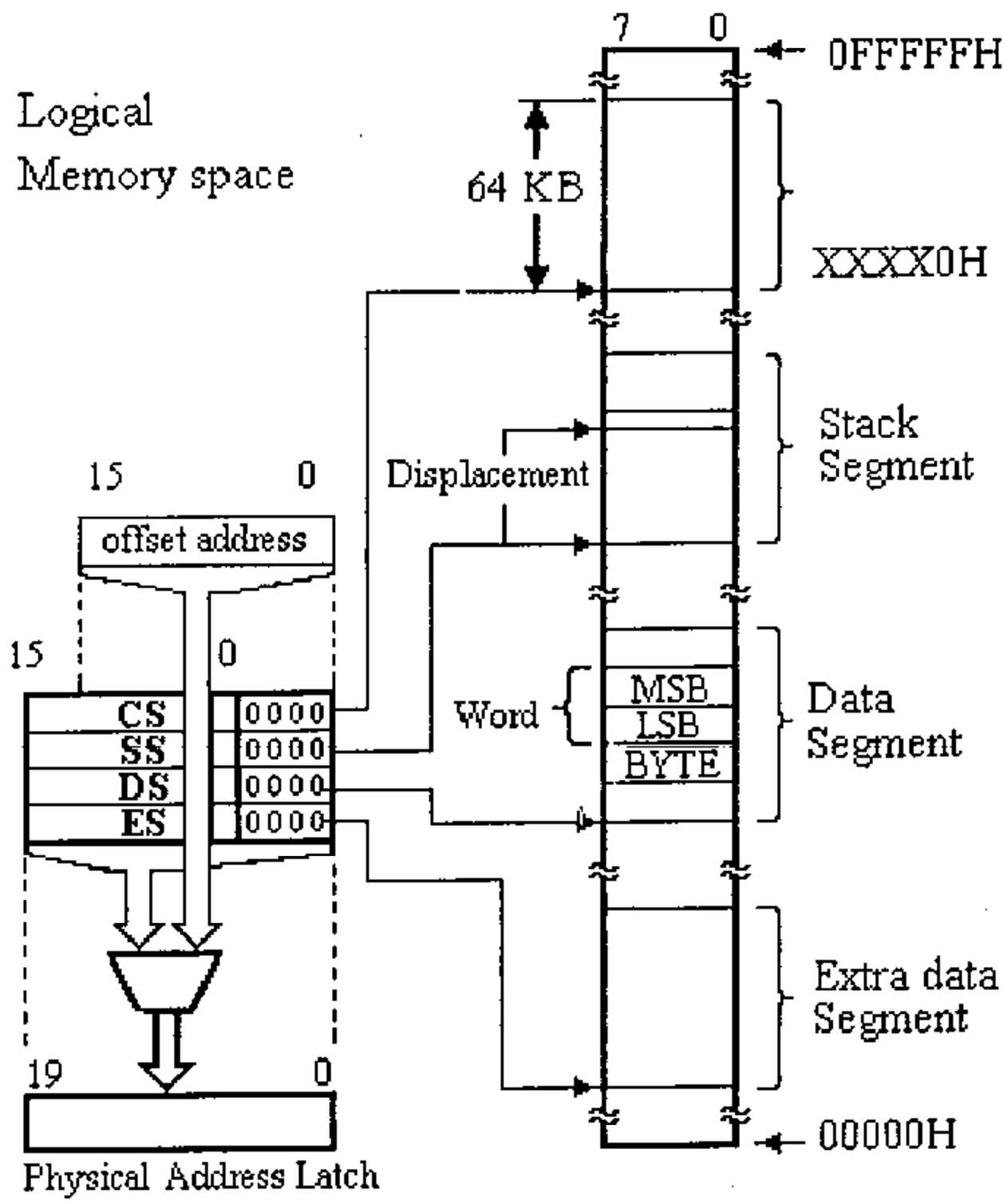
Khi cần truy cập một ô nhớ nào đó, 8086/8088 sử dụng 2 thanh ghi, trong đó một thanh ghi thuộc nhóm các thanh ghi đoạn (CS, DS, SS, ES) và một thanh ghi nữa thuộc nhóm các thanh ghi con trỏ và chỉ số hoặc BX. Giá trị của hai thanh ghi này được kết hợp với nhau theo cách sau:

* Giá trị địa chỉ trong thanh ghi đoạn được dịch trái 4 vị trí, 4 bit thấp được điền 0, tạo thành 1 con số 20 bit với 4 bit thấp bằng 0. Giá trị địa chỉ thứ hai giữ nguyên.

* Hai giá trị địa chỉ trên được đưa vào bộ cộng 20 bit, kết quả ở đầu ra của bộ cộng là một số 20 bit, đó chính là địa chỉ vật lý của ô nhớ cần truy cập. Giá trị này sẽ được truyền ra các chân địa chỉ của bộ vi xử lý 8086/8088. Việc tạo địa chỉ vật lý được thực hiện ngay bên trong bộ vi xử lý, bằng phần cứng.

Địa chỉ (16 bit) chứa trong thanh ghi đoạn được gọi là địa chỉ đoạn. Khi nó tăng/giảm 1 đơn vị sẽ làm cho địa chỉ vật lý tăng/giảm 16 đơn vị, ứng với 16 byte trong bộ nhớ. Vùng bộ nhớ có kích thước 16 byte thường được gọi là 1 paragraph. Địa chỉ 16 bit chứa trong thanh ghi còn lại (thanh ghi offset) được gọi là địa chỉ offset, hay địa chỉ tương đối/ địa chỉ dịch chuyển.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012



Hình 9-04 Đánh địa chỉ phân đoạn

9.1.2.2 Bộ vi xử lý đồng hành 8087

Các bộ vi xử lý 8086/8088, 80286, 80386 chỉ có thể làm các phép tính số học với số nguyên, không thực hiện được các phép tính số học với số dấu phảy động (số thực) một cách trực tiếp. Khi cần làm các phép tính số học với số dấu phảy động thì có thể sử dụng một trong hai giải pháp:

Giải pháp phần mềm: bằng cách lập các chương trình con, đây là giải pháp được dùng nhiều nhất, chương trình con có thể tạo ra các kết quả dấu phảy động bởi các phép tính logic và số học trên các số nguyên.

Giải pháp phần cứng: sử dụng bộ vi xử lý đồng hành 8087, nó được thiết kế đặc biệt để có khả năng tính toán với số dấu phảy động và một số thao tác khác nữa. Bộ vi xử lý 8088 chuyển các nhiệm vụ tính toán cho 8087 thông qua các đường nối tín hiệu đặc biệt và nhận về từ đó các kết quả. Muốn dùng 8087 trong PC, phải có 8088 và các chương trình đặc biệt để tạo ra các mã lệnh riêng cho 8087 hoạt động. Các chương trình này được ghi trong ROM, được cắm vào bảng mạch mở rộng (nếu người sử dụng mua nó) hoặc được ghi lên đĩa kèm theo 8087.

Các bộ vi xử lý 80486DX và mới hơn có chứa cả các bộ vi xử lý đồng hành ngay bên trong nó.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

9.1.3 Bộ vi xử lý 80286

Các đặc tính kỹ thuật chủ yếu

Số thanh ghi: 15 thanh ghi 16 bit, so với 8086/8088 nó có thêm một thanh ghi là MSW (Machine Status Word); ngoài ra trong thanh ghi cờ có thêm 2 cờ mới là NT (Nested Task) và IOPL (I/O Privilege Level), các cờ này được sử dụng trong chế độ bảo vệ.

Bus địa chỉ: 24 dây

Chế độ hoạt động:

1. Chế độ thực dùng các đường địa chỉ A_0 - A_{19}, tương thích với các bộ vi xử lý 8086/8088 ra đời trước nó.
2. Chế độ bảo vệ hay còn gọi chế độ ảo, có khả năng đánh địa chỉ được 16MB bộ nhớ vật lý và 1GB bộ nhớ ảo.

Bus số liệu: 16 dây

Tốc độ chuẩn: 6; 8; 10; 12MHz. Cùng một chương trình chạy trên AT286 8MHz nhanh hơn chạy trên máy dùng 8086, 4.77MHz đến 5 lần.

Số chân: 68 chân, bố trí theo chu vi 2 hình vuông.

9.1.4 Bộ vi xử lý 80386

Các đặc tính kỹ thuật chủ yếu

Số thanh ghi: 32 thanh ghi, trong đó các thanh ghi đa năng (EAX, EBX, ECX, EDX), các thanh ghi con trỏ và chỉ số (ESI, EDI, EBP, ESP), thanh ghi cờ (EFLAGS), con trỏ lệnh (EIP) đều là thanh ghi 32 bit nhưng có thể sử dụng từng nửa 16 bit như những thanh ghi độc lập. Riêng nửa thấp của các thanh ghi đa năng có thể sử dụng từng nửa 8 bit như những thanh ghi độc lập để đảm bảo tính tương thích với các bộ vi xử lý thế hệ trước. Thanh ghi cờ 32 bit đã sử dụng tới 18 cờ. Nhóm thanh ghi đoạn có tới 6 thanh ghi 16 bit. Các phép tính nhân, chia thực hiện bằng phần cứng.

Bus địa chỉ: 32 dây

Chế độ hoạt động:

1. Trong chế độ thực 80386 tương thích với các bộ vi xử lý 8086/8088 nhưng có thể truy cập các thanh ghi 32 bit.
2. Chế độ bảo vệ hay còn gọi chế độ ảo, có khả năng đánh địa chỉ được 4 GB bộ nhớ vật lý và 64 TB bộ nhớ ảo.

Bus số liệu: 32 dây, có thể làm việc với các loại dữ liệu 8, 16 và 32 bit.

Số chân: 132 chân, bố trí theo chu vi 3 hình vuông.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Bộ vi xử lý 386SX có cấu trúc nội tại tương tự 386 nhưng bus dữ liệu liên hệ với bên ngoài là 16 bit. Khả năng địa chỉ hoá vật lý là 16MB. Tốc độ đồng hồ thường là 16, 20, 33, 40MHz.

Bộ vi xử lý 386DX là bộ vi xử lý 32 bit thực sự, có các thanh ghi bên trong và bus dữ liệu liên hệ với bên ngoài là 32 bit. Khả năng địa chỉ hoá vật lý là 4GB. Tốc độ đồng hồ thường là 33, 40MHz.

9.1.5 Bộ vi xử lý 80486

Bộ vi xử lý này là sự phát triển cao hơn của 386DX, trong đó có sử dụng công nghệ RISC. Với cùng tần số đồng hồ, chip 486 chạy nhanh gấp 2 chip 386DX. Bộ vi xử lý 486 có đơn vị đồng vi xử lý ngay bên trong. Các bộ vi xử lý 486SX và 486DX đều là các bộ vi xử lý 32 bit thực thụ; không gian địa chỉ vật lý và không gian địa chỉ ảo là 4GB. Bộ vi xử lý 486DX2 có bộ tăng tốc bên trong - tần số đồng hồ bên trong cao gấp đôi tần số đồng hồ bên ngoài nên sẽ hoạt động nhanh gấp 2 các bộ vi xử lý 486DX thông thường.

9.1.6 Bộ vi xử lý Pentium

- Pentium và Pentium Pro

Pentium được đưa ra thị trường năm 1992 và Pentium Pro năm 1995, chúng là các thế hệ tiếp theo của bộ vi xử lý 486, được thiết kế nhằm đạt hiệu suất cao hơn. Về phương diện người lập trình, hai bộ xử lý này chỉ được bổ sung thêm 4 lệnh so với tập lệnh của 486: trong đó 3 lệnh hỗ trợ cho đa xử lý và một lệnh vận chuyển dữ liệu có điều kiện.

Pentium là bộ xử lý 32 bit, có mọi đặc điểm của 486 và được bổ sung thêm nhiều tính năng kỹ thuật mạnh hơn, chẳng hạn:

- Nó có 2 "đường ống" dẫn lệnh có thể hoạt động độc lập.
- Đơn vị xử lý số dấu phẩy động hoạt động nhanh hơn, với một số phép toán có thể nhanh gấp 10 lần 486.
- Các thanh ghi chứa dữ liệu là 64 bit, các "on chip" bus là 64 bit, có thể truyền song song 8 byte. Hơn thế nữa một thao tác truyền cũng cần ít chu kỳ đồng hồ hơn.

- Pentium MMX

Sau khi Pentium và Pentium Pro ra đời, hãng Intel đã thiết kế một chip vi xử lý mới mạnh hơn, có khả năng hỗ trợ tốt cho việc truyền thông đa phương tiện. Chip này được gọi là Pentium MMX (Multi-Media eXtentions), so với Pentium và Pentium Pro tập lệnh của nó được bổ sung thêm 57 lệnh nhằm tăng tốc cho các chương trình ứng dụng truyền thông và multimedia. Các lệnh MMX này thường thao tác trên các nhóm phần tử dữ liệu ngắn một cách đồng thời, theo kiến trúc song song SIMD.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

9.2 MÁY IBM PC NGUYÊN THUỶ

Kiến trúc của IBM PC nguyên thuỷ đã được đơn giản hoá như minh hoạ trên hình 9-05.

9.2.1 BUS và các khe cắm mở rộng

Bus hệ thống của máy IBM PC đã được trình bày ở Chương 3. Tất cả các bảng mạch mở rộng của IBM PC được ghép nối với máy thông qua các khe cắm nối (slot), từ mỗi một khe cắm có 62 dây dẫn rẽ nhánh ra. Thông qua 62 dây dẫn này máy tính và các thiết bị bên ngoài có thể liên hệ với nhau nhờ các tín hiệu truyền trên chúng. Cả 62 dây dẫn hoạt động song song. Mỗi bảng mạch mở rộng có thể cắm vào một khe cắm bất kỳ vì sự bố trí 62 dây dẫn ở các khe cắm là như nhau, tín hiệu trên các chân ở các khe cắm là như nhau. Đây là sự phát triển tiếp tục của ý tưởng kênh số liệu chung, kênh này người ta quen gọi là kênh I/O.

Về mặt chức năng, ta có thể chia 62 dây này làm 4 nhóm:

1. Nhóm nguồn nuôi: có 8 dây, dẫn các điện áp nuôi ở các mức khác nhau, +5v, -5v, +12v, -12v, "đất".
2. Nhóm dây số liệu: có 8 dây, đó là kênh số liệu chung.
3. Nhóm dây địa chỉ: có 20 dây. Khi diễn ra sự trao đổi số liệu giữa CPU và các thiết bị vào/ra, ngoài số liệu 8 bit, ta còn cần phải xác định địa chỉ nữa. Địa chỉ đó chỉ tới vị trí bộ nhớ tương ứng hoặc số hiệu (nhận diện) của thiết bị vào/ra. Khi sử dụng bộ nhớ trên bảng mạch mở rộng thì cả 20 đường địa chỉ đều cần thiết để có thể đánh địa chỉ cả 1024 KB bộ nhớ, với các thiết bị vào/ra thì chỉ dùng 9 đường để đánh địa chỉ của 512 thiết bị khác nhau.
4. Các đường điều khiển. Các tín hiệu này là các tín hiệu ra lệnh đọc bộ nhớ, ghi vào bộ nhớ, đọc, ghi vào các thiết bị vào/ra ..., ví dụ:

   $\overline{RD}$, $\overline{WR}$, $IO/\overline{M}$, $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$

Các thiết bị vào/ra nối với các khe cắm liên tục theo dõi các tín hiệu trên kênh vào/ra, chúng sẽ nhận ra lệnh điều khiển chúng. Khi một thiết bị nào đó nhận được lệnh, ta nói nó "được chọn" và sau đó nó sẽ nhận được các lệnh điều khiển và số liệu.

9.2.2 Các chip hỗ trợ

Bộ vi xử lý nói chung không trực tiếp liên hệ với bộ nhớ cũng như các cổng vào ra, có hai lý do chính như sau:

Một là, các chân đưa tín hiệu địa chỉ, dữ liệu và điều khiển từ bộ vi xử lý ra ngoài không có khả năng cung cấp dòng điện đủ lớn để duy trì mức tín hiệu đúng trên bus, nơi có thể có từ vài chục tới hàng nghìn phần tử logic nối vào, mỗi phần tử sẽ tiêu thụ một dòng điện chạy vào lối vào của nó.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Hai là, có nhiều nhiệm vụ không cần thiết phải yêu cầu bộ vi xử lý giải quyết, có thể xây dựng các đơn vị logic thực hiện các chức năng nhất định nào đó thay cho bộ vi xử lý, sau đây là các chip thực hiện các chức năng như vậy trong máy IBM PC.

Chip điều khiển DMA 8237

Máy PC sử dụng DRAM, do đó cần phải làm tươi, việc làm tươi được giao cho chip điều khiển DMA 8237 thực hiện.

Việc sử dụng phương pháp DMA thường có nhiều lợi ích trong các hệ thống sử dụng bộ vi xử lý tốc độ thấp, lượng thông tin vào ra vận chuyển trên bus không lớn lắm. Trong các máy tính sử dụng bộ vi xử lý tốc độ cao, lợi ích thu được do sử dụng DMA không nhiều, tuy nhiên các đơn vị DMAC vẫn được sử dụng cho các mục đích khác với mục đích ban đầu người ta đặt ra cho nó.

Chip điều khiển ngắt 8259

Các thiết bị ngoại vi khi cần bộ vi xử lý nhận hoặc gửi thông tin cho nó sẽ phát tín hiệu gửi về bộ vi xử lý qua đường dây yêu cầu ngắt (IRQ), các đường dây này nối với một đơn vị logic gọi là Interrupt Controller (hoặc Interrupt Handler) khi nhận được tín hiệu nó sẽ báo cho CPU biết, CPU sẽ gọi một chương trình con tương ứng ra thực hiện, đáp ứng yêu cầu của thiết bị ngoại vi. Các đường IRQ được gán các mức ưu tiên khác nhau.

Chip ghép nối với thiết bị ngoại vi lập trình được 8255

Chip 8255 này nối bộ vi xử lý với các thiết bị ngoại vi, thí dụ bàn phím và loa. Chip này do bộ vi xử lý điều khiển để truyền các tín hiệu từ bộ vi xử lý tới thiết bị ngoại vi mà nó cần liên lạc.

Chip đồng hồ 8248

Nếu ví bộ vi xử lý như bộ não của máy tính thì đồng hồ có thể ví là trái tim, "trái tim" này đập hàng triệu lần trong một giây (trong máy PC là 14.3 MHz) để giữ nhịp cho bộ vi xử lý cũng như tất cả các chip hỗ trợ khác trong hệ thống máy tính. Tuy nhiên hầu như tất cả các chip hỗ trợ không thể hoạt động được ở tần số cao như vậy, nên trong chip đồng hồ này có mạch chia tần số đồng hồ cho thấp xuống, phù hợp với yêu cầu của các chip hỗ trợ.

Chip định thời (Timer) 8253

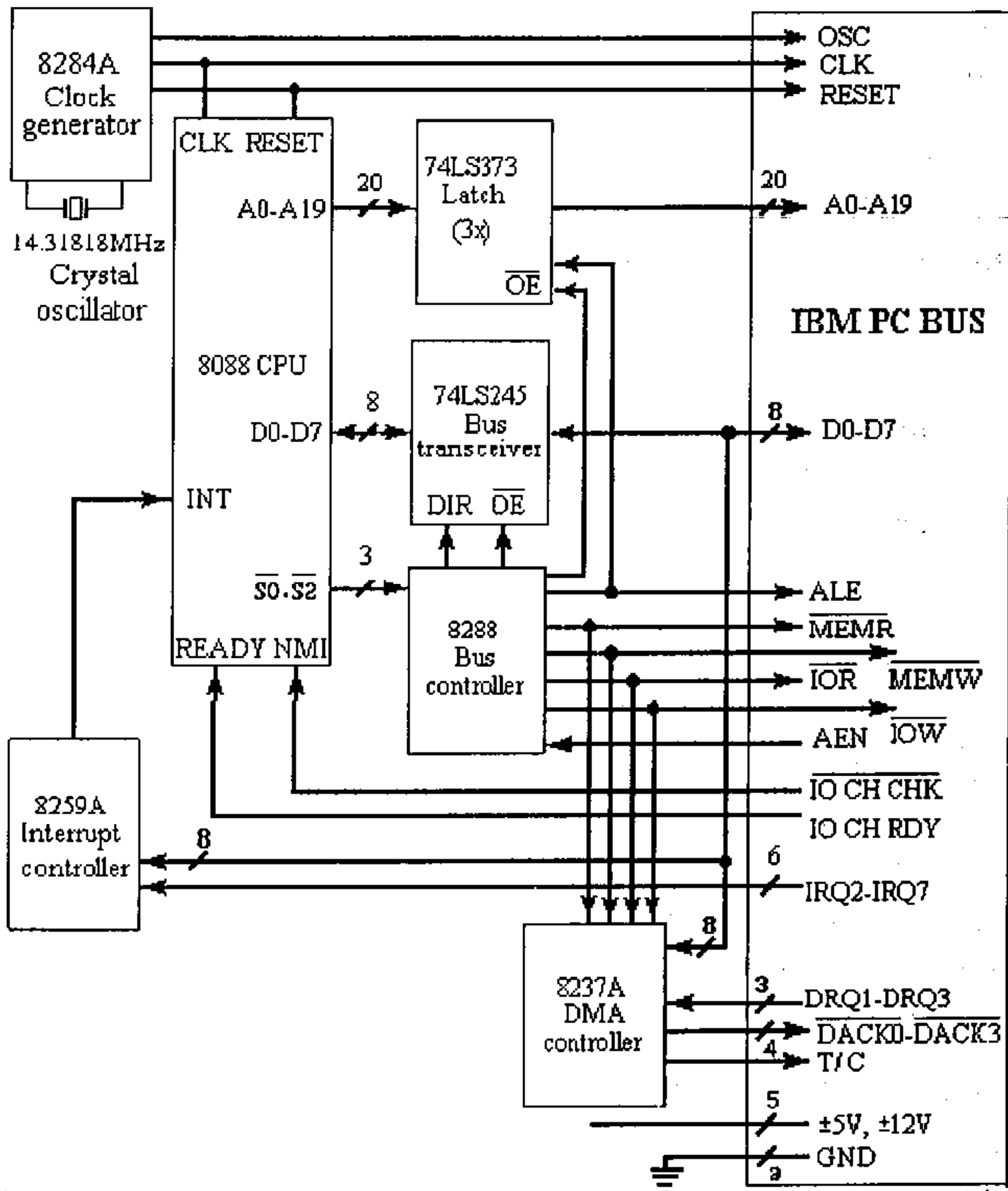
Chip này được sử dụng làm con đếm và làm đơn vị giữ thời gian (timekeeper), nó phát ra các xung điện đều đặn từ một trong các chân ra. Tần số của xung phát ra có thể lập chương trình được nếu cần và mỗi chân ra có thể có một tần số riêng. Tín hiệu từ mỗi chân ra có thể đưa đến một thành phần khác của hệ thống, trong đó có một đường tín hiệu dẫn đến loa, một đường dẫn đến đơn vị điều khiển ngắt, mỗi xung của đường tín hiệu này kích ngắt số 8 một lần, gọi chương trình con xử lý ngắt số 8 để tăng con đếm thời gian lên 1 đơn vị.

Chip điều khiển màn hình 6845

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Khác với các chip trên, chip CRTC (CRT Controller) này không nằm trên mainboard mà nằm trên card điều khiển màn hình. Ban đầu, khi IBM mới đưa ra thị trường máy PC, chip CRTC có ký hiệu 6845 do hãng Motorola sản xuất, card màn hình này điều khiển màn hình loại CGA và MDA.

Một đặc điểm khác biệt giữa chip CRTC và các chip hỗ trợ khác là bộ vi xử lý không liên hệ trực tiếp với CRTC, thay vào đó ROM-BIOS làm việc với chip CRTC, giải phóng bộ vi xử lý khỏi công việc hiển thị.



Hình 9-05 Sơ đồ được đơn giản hoá của một CPU họ PC cùng các chip hỗ trợ

Chip điều khiển đĩa 765

Giống như chip CRTC, chip này cũng thường nằm trên card mở rộng, được hệ điều hành đánh địa chỉ và nó điều khiển việc thực hiện các chức năng của ổ đĩa. Ngoài ra chip điều khiển đĩa cũng được đánh địa chỉ bởi ROM-BIOS chứ không

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

phải bởi bộ vi xử lý. (Máy PC đầu tiên không có ổ đĩa mà chỉ sử dụng ổ băng cassette làm thiết bị nhớ ngoài.)

Chip đồng xử lý toán học 8087

Chip này cũng có thể được coi là một chip hỗ trợ cho bộ vi xử lý chính 8088.

9.2.3 Các linh kiện khác của máy IBM PC

Để bộ vi xử lý có thể làm việc được, nó cần nhiều linh kiện phụ trợ nữa. IBM PC sử dụng rất ít linh kiện đặt riêng mà phần lớn là các linh kiện đã được tiêu chuẩn hoá, kể cả bộ vi xử lý 8088 trở đi. Điều làm cho IBM PC nổi bật lên giữa các máy cùng loại là ở thiết kế của nó - được cấu tạo từ các linh kiện tiêu chuẩn. Khi xem xét cách thức bố trí, sắp xếp các phần tử của IBM PC, ta có thể chia các linh kiện của nó làm 2 nhóm:

- **Các linh kiện nằm trên bảng mạch hệ thống** hay còn gọi là bảng mạch chủ (Mother board; Main board; System board). Tất cả các phần tử chính, có trong tất cả các máy IBM PC nằm trên một bảng mạch in duy nhất - bảng mạch chính. Các phần tử trên bảng mạch này đảm bảo cho máy tính có thể hoạt động và sử dụng được mà không cần thêm bất cứ một bảng mạch tuỳ chọn nào. Trên bảng mạch này có bộ vi xử lý 8088 (hoặc 8086, 80286, 80386 ...), RAM (64 KB ở các máy thế hệ đầu tiên, 512 KB, 640 KB... ở các máy thế hệ sau này), ROM.v.v. Bảng mạch chính là phần không thể tách rời của IBM PC, đó là thiết kế của hãng IBM.

- **Các linh kiện nằm trên các bảng mạch mở rộng** (Extension cards). Các bảng mạch mở rộng là tuỳ chọn đối với người mua, ta có thể lắp thêm vào máy bất cứ cái gì cũng được, miễn là chúng tuân theo các quy định liên quan tới kích thước của bảng mạch chính, các tiêu chuẩn ghép nối điện, điều kiện toả nhiệt.v.v. Có thể kể tên các bảng mạch mở rộng sau: RAM, ROM, CGA Card, VGA Card, Disk Controller.v.v...

Ngoài bảng mạch chính và các bảng mạch mở rộng, trong hộp máy IBM PC còn có bộ nguồn nuôi chung cho toàn bộ máy, kể cả các bảng mạch mở rộng cũng như các ổ đĩa cứng và các ổ đĩa mềm.v.v.

9.2.4 Những chuyển mạch đặt hệ thống (system switch)

Các chuyển mạch này được bố trí thành 2 hàng ở phía bên trái của Main board (ở các máy tương thích, vị trí của các chuyển mạch này có thể khác đi). Nhiệm vụ của chúng là đánh dấu hệ thống được hình thành như thế nào, cấu hình ra sao, thí dụ: số ổ đĩa, loại ổ đĩa, dung lượng bộ nhớ.v.v. thực tế thì các chuyển mạch này không điều chỉnh hay chuyển mạch cái gì cả, chúng chỉ đóng vai trò logic. Khi ta khởi động máy, chương trình khởi động sẽ đọc trạng thái của các chuyển mạch và trên cơ sở đó thiết lập giá trị cho các ô nhớ có địa chỉ xác định. Về sau, các chương trình hệ thống hoặc chương trình của người sử dụng, khi cần số liệu liên quan tới cấu hình

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

của hệ thống như dung lượng bộ nhớ là bao nhiêu? có mấy ổ đĩa, loại ổ đĩa.v.v. thì chúng chỉ cần đọc giá trị của các ô nhớ tương ứng này.

Khi cần, ta có thể thay đổi giá trị ghi về cấu hình của hệ thống, thậm chí có thể làm hiện diện (ảo) các thiết bị trong thực tế không được nối với IBM PC.

9.2.5 Bộ nhớ trong của IBM PC

9.2.5.1 Kích thước ô nhớ và kích thước toán hạng (độ dài từ)

Bộ nhớ trong là nơi lưu trữ các thông tin tạm thời, nó là tập hợp các ô nhớ, mỗi ô nhớ có dung lượng 1 byte và có 1 địa chỉ riêng duy nhất. Trong các loại máy tính khác, 1 ô nhớ có thể có kích thước lớn hơn: 2 byte, 4 byte, 64 byte ...

Việc chọn ô nhớ kích thước 1 byte có ưu điểm là tiết kiệm bộ nhớ. Các thông tin lưu trữ trong bộ nhớ phần lớn là mã ASCII của các ký tự, cũng như mã lệnh có kích thước 1 byte, vì vậy kích thước 1 địa chỉ nhớ bằng 1 byte là kinh tế nhất.

Trong hệ lệnh của 8086/8088 có nhiều lệnh xử lý số 2 byte, nếu con số đó là toán hạng bộ nhớ thì 8086/8088 có thể thao tác liền trên 2 ô nhớ một byte, ta gọi số 2 byte như vậy là một từ (word), trong mỗi từ, byte có trọng số thấp hơn luôn nằm ở ô nhớ có địa chỉ thấp, byte có trọng số cao hơn nằm ở địa chỉ cao hơn.

Các số liệu dài hơn (3,4, ... byte) cũng có thể dùng được nhưng việc dùng chúng cần các chương trình con dành riêng.

Bộ xử lý đồng hành 8087 sử dụng một số kiểu số khác với 8088: số nguyên 4 byte, các số dấu phảy động 2, 4, 10 byte, thậm chí cả dạng số thập phân 20 chữ số nữa. Còn 8088 không hiểu được các dạng số liệu trên.

9.2.5.2 Tổ chức bộ nhớ của IBM PC

Có thể trình bày cách tổ chức bộ nhớ của IBM PC như dưới đây, hình 9-06.

Bộ nhớ quy ước (Conventional memory)

Là miền nhớ từ địa chỉ thấp nhất cho tới 640 KB (Kilobyte), các Version khác nhau của DOS quản lý vùng nhớ này do đó ta không cần phải có chương trình quản lý riêng. Phần lớn các chương trình phần mềm chạy trong môi trường DOS chỉ sử dụng vùng nhớ này.

Vùng nhớ phía trên - UMA (Upper Memory Area)

Đó là vùng nhớ 384 KB phía trên vùng 640 KB bộ nhớ quy ước (từ 640 KB đến 1 MB). UMA được sử dụng bởi phần cứng, chẳng hạn bộ điều khiển hiển thị (CRTC). Tuy nhiên bộ điều khiển hiển thị và Video RAM chưa chiếm hết toàn bộ miền UMA này, phần địa chỉ nhớ chưa dùng thường lại là các miền không liên tục, bị phân mảnh nên còn được gọi là các khối nhớ phía trên (Upper Memory Blocks-UMBs). Trong chế độ địa chỉ thực (Real Mode) của bộ vi xử lý, DOS Version trước 5.0 không đánh địa chỉ trực tiếp được miền này. Từ Version 5.0 DOS hỗ trợ việc sử dụng UMBs, vì vậy có thể sử dụng UMBs để chạy các trình điều khiển thiết bị và

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

các chương trình thường trú vì các chương trình này đều nhỏ, kích thước không vượt quá 64KB. Vùng bộ nhớ còn trống cho UMBs phụ thuộc vào dung lượng Video RAM, các Card mở rộng và ROM-BIOS.

Các chương trình quản bộ nhớ thực hiện việc chuyển bộ nhớ mở rộng xuống vùng UMB như EMM386.EXE, 386Max, QUEMM. Để DOS có thể dùng được UMBs cần đặt các khai báo sau trong tệp CONFIG.SYS:

DOS = HIGH, UMB hoặc DOS = LOW, UMB

Khi thực hiện một trong 2 lệnh này DOS tìm trình điều khiển EMM386.EXE để nó thực hiện nhiệm vụ phân phối bộ nhớ UMBs.

Các lệnh của DOS như DEVICEHIGH, LOAD HIGH được sử dụng để nạp các Driver và các chương trình của DOS vào UMB

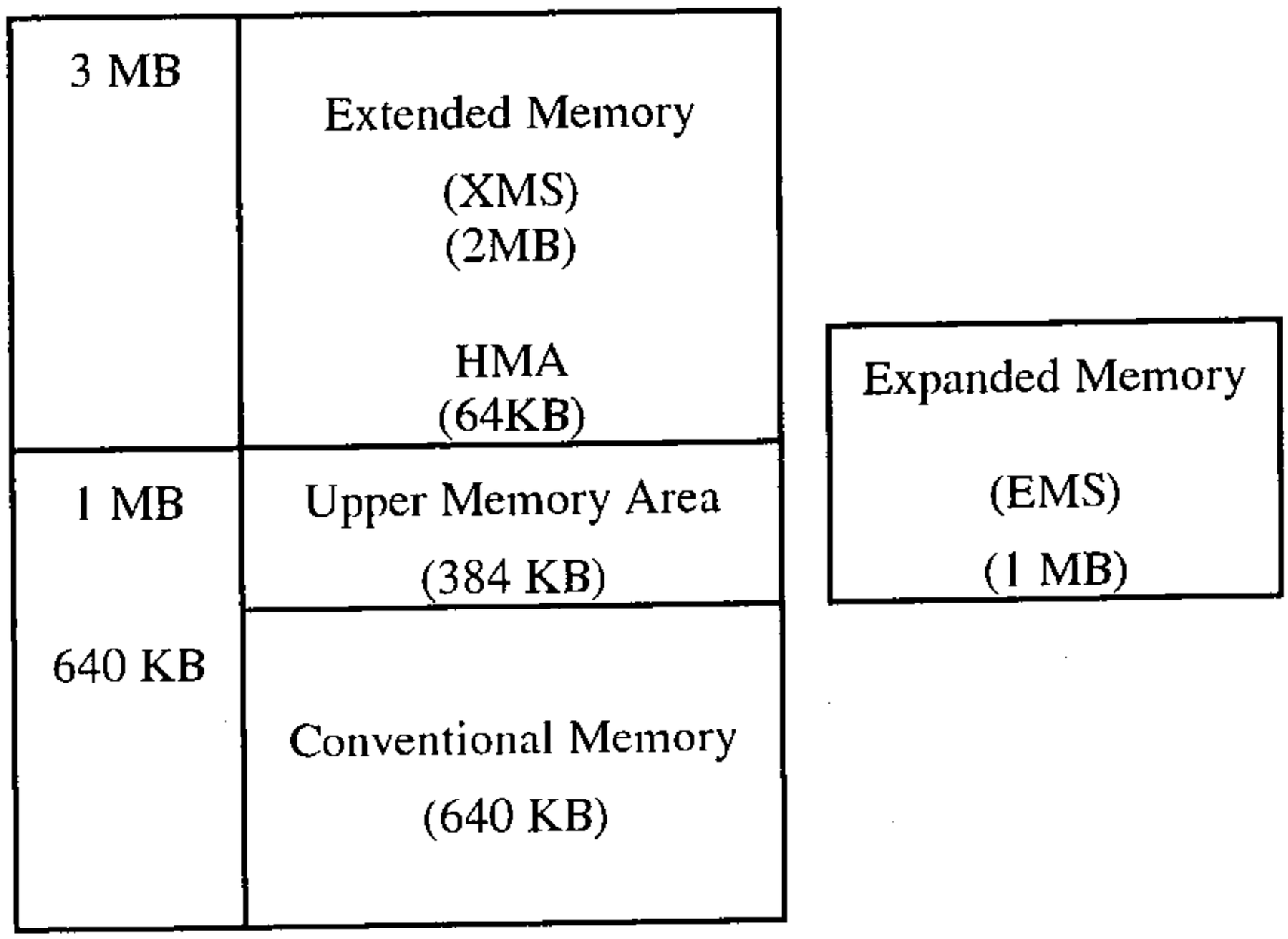
Hình vẽ 9-06 là cấu hình bộ nhớ của một máy tính với bộ vi xử lý 80386 có 640 KB RAM quy ước và 2 MB RAM mở rộng, ngoài ra còn có 1 MB RAM bành trướng.

Bộ nhớ mở rộng - XMS (Extended Memory)

Đó là vùng nhớ bên trên miền 1 MB ở các máy tính dùng bộ vi xử lý 80286, 80386 hoặc 80486. Bộ nhớ mở rộng cần trình điều khiển, thí dụ HIMEM.SYS. Windows và các chương trình ứng dụng chạy trong môi trường này cần phải sử dụng bộ nhớ mở rộng.

Bộ nhớ bành trướng - EMS (Expanded Memory)

Đó là bộ nhớ bổ sung thêm cho bộ nhớ quy ước (640KB), nhiều loại máy tính



Hình 9-06 Bản đồ bộ nhớ của một máy tính

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

PC có thể cắm thêm EMS đồng thời phải có trình điều khiển cho nó. Một số chương trình ứng dụng dựa trên DOS có thể sử dụng bộ nhớ này, các chương trình chỉ dùng được tối đa 64 KB EMS tại mỗi thời điểm bằng cách đánh địa chỉ một phần của UMA - gọi là một khung trang EMS (EMS page frame). Chính điều này làm cho việc sử dụng EMS chậm hơn việc sử dụng XMS. EMM386 có thể sử dụng XMS theo cách mô phỏng EMS cho các chương trình cần EMS. Windows và các chương trình dựa trên Windows không sử dụng EMS.

Bản đồ bộ nhớ của IBM PC

Hình 9-07 là bản đồ vùng nhớ quy ước (1MB đầu tiên) của các máy IBM/PC

0000:0000 - 0000:0400 9000:FFFF	Bảng vector ngắt (1KB) Bộ nhớ quy ước 640 KB
A000:0000 - A000:FFFF	Video RAM bổ sung cho card EGA/VGA
B000:0000 - B000:7FFF B800:0000 - B000:FFFF	Video RAM cho card điều khiển màn hình đơn sắc Video RAM cho card điều khiển màn hình màu CGA/EGA/VGA
C000:0000 - C000:FFFF	ROM-BIOS mở rộng
D000:0000 - D000:FFFF	IBM không mô tả cụ thể
E000:0000 - E000:FFFF	IBM không mô tả cụ thể
F000:0000 - F000:FFFF	ROM BIOS

Hình 9-07 Bản đồ bộ nhớ 1MB đầu tiên của máy tính IBM/PC

- 1KB địa chỉ thấp nhất: Bảng vector ngắt
- Sau bảng vector ngắt: Dành cho hệ điều hành
- Sau phần cho hệ điều hành cho đến sát biên giới 640KB: cho các chương trình ứng dụng chạy trong môi trường DOS ("for DOS").

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

9.2.6 Các cổng - Ports

Bộ vi xử lý liên hệ với các thiết bị ngoại vi và các đơn vị phần cứng của hệ thống thông qua các cổng. Cổng giống như một thanh ghi 8 bit nối với 1 đơn vị phần cứng cụ thể để vào/ra số liệu, mỗi cổng được gán một địa chỉ có giá trị nằm trong khoảng 0..65535.

Bộ vi xử lý sử dụng bus hệ thống (bao gồm bus điều khiển, bus địa chỉ và bus dữ liệu) để liên lạc với các cổng. Khi bộ vi xử lý cần truy cập một cổng, nó truyền ra bus điều khiển một tín hiệu báo cho các phần cứng nối với bus biết rằng nó muốn truy cập một cổng chứ không phải là RAM hay ROM; Còn địa chỉ cổng được đặt ở 16 bit thấp nhất của bus địa chỉ. Tuỳ theo bộ vi xử lý muốn gửi (write) hay nhận (read) đối với cổng mà dữ liệu sẽ truyền từ các chân ra dữ liệu của bộ vi xử lý tới cổng hay theo hướng ngược lại, thông qua bus dữ liệu.

Bảng 9-01 Địa chỉ cổng của một số Chip

Tên thiết bị	Máy PC/XT	Máy AT
Chip điều khiển DMA (8237A-5)	000 - F	000 - 01F
Chip điều khiển ngắt (8259A)	020 - 021	020 - 03E
Đơn vị giao diện với máy in thứ hai (LPT2)	Không có	278 - 27F
Second serial interface (COM2)	2F8 - 2FF	2F8 - 2FF
NetWork card (card mạng - NIC)	Không có	360 - 36E
Đơn vị giao diện với máy in thứ 1 (LPT1)	378 - 37F	378 - 37F
Đơn vị điều khiển đĩa	3F0 - 3F7	3F0 - 3F7
Đơn vị giao diện nối tiếp thứ nhất (COM1)	3F8 - 3FF	3F8 - 3FF

Họ vi xử lý 80xx có 2 lệnh để điều khiển quá trình này:

- Lệnh OUT: gửi data từ bộ vi xử lý ra một cổng.
- Lệnh IN: chuyển data từ một cổng vào trong bộ vi xử lý.

Mỗi thiết thị phần cứng chiếm một miền các địa chỉ cổng. Chính vì lý do này nếu ta cắm vào các khe cắm mở rộng một số card điều khiển thiết bị mà chúng lại được đặt trước các giá trị địa chỉ (cổng) trùng nhau thì sẽ xảy ra xung đột - conflict. Để giải quyết vấn đề này, hầu hết các card điều khiển đều có các bộ công tắc khai báo (DIP switches) để người sử dụng có thể tự đặt cho card của mình một địa chỉ (cổng) mà card sẽ đáp lại khi có lời gọi. Một số card điều khiển được bán kèm theo chương trình để setup nó, trong số các tham số có thể setup được thường có địa chỉ cổng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

9.2.7 Các ngắt (Interrupts)

Ngắt là cơ chế bắt bộ vi xử lý phải dừng công việc (hoặc chương trình) mà nó đang thực hiện để thực hiện một công việc khác.

9.2.7.1 Các ngắt phần cứng

Trong máy tính ngoài bộ vi xử lý còn có các chip hỗ trợ cho nó, các chip này quản lý và thực hiện các nhiệm vụ mà bộ vi xử lý không thể tự nó quản lý nổi. Các chip này còn có tên gọi là ***Controller*** bởi vì chúng điều khiển một số đơn vị phần cứng thay cho bộ vi xử lý để thực hiện nhiệm vụ của cả hệ thống.

Trong quá trình hoạt động, máy tính cần phản ứng với những sự kiện xảy ra bên ngoài bộ vi xử lý - ví dụ phản ứng với một phím được gõ ở bàn phím.v.v. Điều này có thể được thực hiện bằng 2 cách:

Cách thứ nhất: máy tính sau mỗi khoảng thời gian nhất định, đủ ngắn phải kiểm tra tất cả các điểm mà bộ vi xử lý liên hệ với thế giới bên ngoài, để xác định xem có sự kiện nào cần phải phản ứng xuất hiện không?

Cách thứ hai: máy tính cứ thực hiện công việc của mình, không để ý gì tới xung quanh cho tới khi các thiết bị ngoại vi phát tín hiệu yêu cầu - đây là hệ thống của các lệnh ngắt bằng phần cứng. Sự ứng dụng các lệnh ngắt là một yếu tố then chốt trong sự hoạt động của máy tính, nhờ nó máy tính không phải kiểm tra mãi xem có việc gì cần phải làm cho các thiết bị I/O không, nếu xảy ra điều gì thì bản thân nhiệm vụ sẽ tìm đến bộ vi xử lý.

Các ngắt phần cứng (4):

Int 2: báo hiệu có sự cố phần cứng, không thể che (non maskable)

Int 8: ngắt tín hiệu đồng hồ

Int 9: ngắt bàn phím

Int 14: ngắt báo hiệu yêu cầu của ổ đĩa

Ngoài 4 ngắt phần cứng nêu trên, còn có 7 sự ngắt phần cứng dành cho các thiết bị ngoại vi, đó là các ngắt: 6, 7, 10, 11 , 12, 13, 15. Trong đó ngắt 12 dành cho bộ điều khiển thông tin; ngắt 15 dành cho khối ghép nối máy in.

9.2.7.2 Các ngắt logic và các ngắt phần mềm

- **Các ngắt logic** (4)

Đó là các ngắt do chính bộ vi xử lý phát ra khi nó gặp phải điều gì đó khác thường.

Int 0: khi gặp phép chia cho 0

Int 1: step (chạy từng bước)

Int 3: lệnh điểm dừng (break point) sinh ra ngắt này

Int 4: sinh ra do tràn số (kết quả không chứa vừa trong một thanh ghi)

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Các ngắt phần mềm**

Ngắt mềm là sự gọi một chương trình bằng chỉ thị INT của bộ vi xử lý, để nó thực hiện một chương trình phục vụ nào đó của DOS hay BIOS... Các chương trình được gọi bằng chỉ thị INT thường được gọi là ***chương trình con xử lý ngắt***. Địa chỉ của chương trình này phải được chứa trong bảng vector ngắt trước khi gọi chương trình. Các ngắt phần mềm cực kỳ quan trọng trong sự hoạt động của máy tính IBM PC, cho khả năng thực hiện tất cả những gì mà các thiết bị bên ngoài có thể thực hiện được, có thể tạo ra sự ngắt; đồng thời cho ta khả năng sử dụng được tất cả các dịch vụ mà phần cứng và phần mềm đưa ra.

- **Các ưu điểm của việc gọi chương trình bằng ngắt mềm**

Trong trường hợp cần thiết có thể sửa đổi các vector ngắt cũng như chính chương trình con xử lý ngắt một cách dễ dàng, ngay cả khi chương trình sau khi sửa dài hơn chương trình ban đầu hoặc được đặt tại các địa chỉ nhớ khác. Nhờ đó có thể sửa đổi, cải tiến hoặc thay thế các chương trình dịch vụ cho phù hợp với mục đích riêng của chúng ta, nếu chúng ta muốn.

9.2.7.3 Các ngắt khác

- Ngoài các ngắt cứng và mềm nêu trên còn có một loại ngắt không trỏ đến chương trình con nào cả, chúng trỏ đến các bảng số liệu quan trọng của hệ thống. Đó là các ngắt sau:

INT 1DH : trỏ đến Bảng thông số VIDEO, còn gọi bảng khởi tạo màn hình.

INT 1EH : trỏ đến Bảng thông số đĩa (bảng cơ sở đĩa)

INT 1FH : trỏ đến Bảng các ký tự đồ thị chứa các tham số dùng trong ROM-BIOS khi khởi động hệ thống.

- Các ngắt công dụng chung, do người lập trình tạo ra: các ngắt này được thiết lập bởi chương trình của chúng ta với mục đích sử dụng tạm thời. Các số hiệu giành cho chúng là từ 93 đến 103. (Miền số hiệu này là của DOS 5.0, với các version sau này có thể thay đổi, bạn đọc nếu cần dùng phải tham khảo các tài liệu tương ứng của DOS.)

9.3 HỆ MÁY IBM PC/XT, IBM PC/AT

9.3.1 Các đặc tính kỹ thuật chính của IBM PC/XT

(XT = Extent Technology)

- Sử dụng bộ vi xử lý 8088 hoặc 8088-2, ở các máy tương thích, có thể thấy các bộ vi xử lý có ký hiệu khác.
- Bộ đồng vi xử lý: 8087 hoặc 8087-2 (tuỳ chọn)
- Tốc độ đồng hồ chủ: 4.77 MHz hoặc 8 MHz

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Bus địa chỉ: 20 dây
- Bus số liệu: 8 dây
- Số kênh DMA: 4
- RAM: 640 KB "On board"
- Số ổ đĩa mềm (FDD): 2 ổ 51/4", 360 KB
- Bàn phím: 84 hoặc 101 phím
- Số khe cắm mở rộng (Extension slot): 8
- Hệ điều hành PC DOS, MS DOS
- Các bảng mạch mở rộng chuẩn:
  Bảng mạch điều khiển ổ đĩa mềm - FDC
  Bảng mạch điều khiển màn hình màu hoặc đơn sắc/máy in
  Cổng truyền thông không đồng bộ RS 232-C
- Các tuỳ chọn:
  Đĩa cứng (HDD): 1 ổ 20 MB
  Màn hình đơn sắc hoặc màu CGA hoặc Card EGA và màn hình EGA

9.3.2 Các đặc tính kỹ thuật chính của IBM PC/AT

(AT=Advanced Technology)

- Sử dụng bộ vi xử lý 80286 hoặc 80386, 80486; ở các máy tương thích, có thể thấy các bộ vi xử lý có ký hiệu khác.
- Bộ đồng vi xử lý: 80287 đi với 80286, 80387 đi với 80386v.v... Các chip vi xử lý từ 80486DX có chứa cả bộ đồng vi xử lý bên trong (Built-in Coprocessor).
- Tốc độ đồng hồ chủ: 6 MHz hoặc 8 , 10, 16 MHz ở AT 286, các máy AT 386, AT 486 sử dụng tốc độ đồng hồ cao hơn (25, 33, 45, 100... MHz)
- Bus địa chỉ: 24 dây ở AT 286, 32 ở AT 386, AT 486
- Bus số liệu: 16 dây
- Số kênh DMA: 7
- RAM: 1 MB (chuẩn) + 15 MB ở chế độ địa chỉ ảo được bảo vệ (Máy AT286). (Trong các máy sử dụng bộ xử lý 80386, 80486 hoặc Pentium, dung lượng RAM lớn hơn con số nêu trên rất nhiều. Vào năm 2000 Các máy AT thường có ít nhất 32MB RAM)
- Đĩa mềm (FDD): có thể có tới 2 ổ loại 51/4", hoặc 31/2"
- Đĩa cứng (HDD): có thể có tới 2 ổ (một số máy mới có thể quản lý được tới 4 ổ) với dung lượng từ 20MB ở các máy AT cổ tới vài chục GB ở các máy đời mới.
- Bàn phím: 84 hoặc 101 phím (một số bàn phím mới có nhiều phím hơn nữa)
- Số khe cắm mở rộng: 8

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

(Ngày nay con số này có thể rất khác nhau trong các máy IBM PC/AT và các máy tương thích với chúng. Các khe cắm trong một máy cũng có thể có một vài loại, tuân theo các chuẩn ghép nối khác nhau của một số loại bus)

- Hệ điều hành PC DOS, MS DOS.

  (Trong một vài năm trở lại đây, nói chung các máy IBM PC/AT và tương thích thường được cài đặt hệ điều hành WINDOWS-95 hoặc các phiên bản mới hơn, còn rất ít người sử dụng hệ điều hành PC DOS hoặc MS DOS)

- Các bảng mạch mở rộng chuẩn:
  - Bộ điều khiển đĩa mềm - FDC, đĩa cứng - HDD
  - Card điều khiển màn hình đơn sắc/máy in hoặc CGA; EGA; VGA
  - Cổng truyền thông không đồng bộ RS 232-C

(Các thế hệ máy AT gần đây thường có các đơn vị điều khiển thiết bị ngoại vi gắn ngay trên Mainboard, người ta gọi là "on-board card", xu hướng này ngày càng tăng lên cùng với sự phát triển của các công nghệ phần cứng.)

9.4 HỆ ĐIỀU HÀNH DOS CỦA MÁY IBM PC

Vai trò và nhiệm vụ của hệ điều hành: Hệ điều hành là tập hợp các công cụ phần mềm, cho phép sử dụng hữu hiệu phần cứng cho các mục đích khác nhau.

Trước những năm 90 hệ điều hành DOS được sử dụng rộng rãi trên các máy tính cá nhân. Trên máy IBM-PC có PC-DOS là một hệ điều hành của hãng IBM. Trên máy tương thích với IBM-PC thường có MS-DOS, là một hệ điều hành của hãng Microsoft, nó hoàn toàn thương thích với PC-DOS về phương diện sử dụng.

Thực tế một máy tính có thể được cài đặt và hoạt động với một số hệ điều hành khác nhau, ví dụ DOS, WINDOWS, CP-M86, XENIX, VENIX.v.v. Trong số đó DOS là một hệ điều hành từng được sử dụng sớm và hết sức rộng rãi.

Hệ điều hành PC-DOS bao gồm 6 thành phần cơ bản:

1. ROM-BIOS
2. Record khởi động
3. IBMBI0.COM (IBMIO.SYS, IO.SYS)
4. IBMDOS.COM (MSDOS.SYS, DOS.SYS)
5. COMMAND.COM
6. Các lệnh ngoại trú

(Chú ý: tên của thành phần thứ ba và thứ tư nêu trên thay đổi tuỳ thuộc vào phiên bản của hệ điều hành và vào nhà sản xuất là IBM hay Microsoft).

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

9.4.1 ROM-BIOS

ROM-BIOS là một thành phần nằm trong ROM nói chung. Chip ROM trên mainboard của IBM-PC có thể là ROM kiểu mặt nạ hay EPROM, phần mềm ghi trong ROM được coi là một trong các thành phần cơ bản của hệ điều hành PC-DOS. Tất nhiên chúng cũng có thể được dùng cho bất kể loại hệ điều hành nào khác cài đặt trên IBM-PC. ROM chứa các chương trình phục vụ các chức năng sơ đẳng nhất của hệ điều hành và của hệ thống máy tính. Phần lớn nội dung của ROM là các chương trình, ngoài ra còn có một số bảng số liệu về hệ thống.

Việc ghi các chương trình phục vụ cơ sở, thường dùng vào trong ROM có lợi ích rất lớn như sau:

- Không phải nạp từ đĩa mỗi khi khởi động hệ điều hành, nhờ đó có thể sử dụng các chương trình này ngay từ đầu.
- Các chương trình trong ROM được dùng làm cơ sở để xây dựng các chương trình khác, kể cả hệ điều hành. Chúng chính là thành phần dùng chung cho mọi phần mềm chạy trên hệ thống máy tính.

Trong họ máy IBM-PC và các máy tương thích, ROM có 4 thành phần là:

1. Chương trình khởi động, làm những công việc đầu tiên cần cho việc khởi động máy. Nó thường bao gồm một số chương trình con khởi động.
2. ROM-BIOS: chứa tập các chương trình ngôn ngữ máy, thực hiện các thao tác được coi là sơ đẳng nhất của hệ thống máy tính, phục vụ cho hoạt động liên tục của máy tính.
3. ROM-BASIC: làm hạt nhân cho ngôn ngữ lập trình BASIC. Thành phần này thực sự chỉ có trong thế hệ máy đầu tiên.
4. ROM mở rộng: gồm các chương trình thêm vào cho ROM chính, khi người ta thêm vào hệ thống máy tính một thiết bị tuỳ chọn nào đấy.

Phần bộ nhớ cao nhất dành cho ROM, bắt đầu từ đoạn F000, các máy PC khác nhau sử dụng lượng bộ nhớ khác nhau trong vùng 64K này, PC nguyên thuỷ dùng 40K, AT dùng cả 64K.

9.4.1.1 ROM khởi động (Start-up ROM)

Khi bật máy, một chương trình có tên POST (Power On Self Test = tự kiểm tra khi bật máy) bắt đầu chạy, nó thực hiện các chức năng chính sau đây: Giám sát quá trình khởi động máy, kiểm tra nhanh độ tin cậy của toàn bộ hệ thống máy tính, kể cả các chương trình trong ROM để khẳng định mọi thứ có thể làm việc được. POST là bước đầu tiên quan trọng nhất để xác định chắc chắn máy vi tính đã sẵn sàng làm việc. Các chương trình con trong POST đều rất ngắn.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Kiểm tra bộ nhớ**

POST thường kiểm tra bộ nhớ theo khối, với mỗi khối POST chỉ kiểm tra một số byte đầu, nếu các byte được kiểm tra này bình thường thì POST coi cả khối là bình thường. Việc kiểm tra bộ nhớ thường cần nhiều thời gian vì bộ nhớ thường lớn.

Nếu bộ nhớ bị hỏng ở vùng địa chỉ thấp, nơi bộ vi xử lý cần sử dụng thì POST thường thông báo bằng các tiếng "bip bip" ở loa.

Nếu bộ nhớ bị hỏng ở vùng địa chỉ cao hơn thì POST thường thông báo ra màn hình địa chỉ của ô nhớ hoặc đoạn bộ nhớ bị hỏng, sau đó dừng lại ("treo" máy).

Trong khi kiểm tra bộ nhớ, POST thường thông báo trên màn hình dung lượng nhớ mà nó đã kiểm tra và thấy là bình thường.

Khi đã kiểm tra xong toàn bộ bộ nhớ và không thấy có vấn đề gì, POST gửi tín hiệu ra loa, chúng ta sẽ nghe thấy một tiếng "bip".

- **Khởi động (Reset) các thiết bị ngoại vi chuẩn được nối với máy**

Đó là các thiết bị như bàn phím, màn hình, máy in, ổ đĩa. Chúng ta có thể nhận thấy điều này ở các thiết bị có đèn báo trạng thái. Thí dụ:

Với thiết bị bàn phím, các đèn báo trạng thái Caps Lock, Scroll Lock, Num Lock sẽ loé sáng khi POST gửi tín hiệu reset tới.

Với màn hình chúng ta không nhận thấy vì POST gửi tín hiệu reset tới chip điều khiển màn hình - CRTC (chip này nằm trên card điều khiển màn hình).

Với máy in, POST gửi tín hiệu reset tới chip điều khiển giao diện với máy in, nếu máy in nối với máy tính và đã bật điện từ trước, chúng ta có thể thấy đèn Ready của máy in nhấp nháy. (Nếu máy in là loại máy in kim, chúng ta sẽ thấy đầu in được đưa về đầu dòng...)

Đối với các ổ đĩa, POST gửi tín hiệu reset tới các đơn vị điều khiển chúng, chúng ta có thể thấy các đèn báo trên các ổ đĩa mềm và đèn báo của ổ đĩa cứng (HDD led) lần lượt loé sáng. Trình tự thông thường là đèn của các ổ A, B, C rồi D loé sáng, tuy nhiên trong một số máy trình tự này có thể thay đổi, tuỳ thuộc vào việc máy đã được setup như thế nào.

Chương trình POST thực hiện reset các thiết bị ngoại vi chuẩn nhằm biết được danh sách các thiết bị chuẩn nối với máy cũng như trạng thái của các thiết bị này, bởi vì hầu hết các thiết bị ngoại vi đều có khả năng thông báo trạng thái của nó cho POST khi bị reset.

Danh sách thiết bị mà POST phát hiện được sẽ được nó ghi vào một vùng nhớ định trước, các chương trình chạy trên hệ thống sau này có thể đọc để biết (và tất nhiên cũng có thể thay đổi và thậm chí có thể làm "biến mất" một thiết bị nào đó).

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- **Tạo bảng các vectơ ngắt**

POST đặt các giá trị định sẵn cho các véctơ ngắt, các giá trị định sẵn này hoặc là trỏ tới một chương trình xử lý ngắt chuẩn nằm sẵn trong ROM-BIOS, hoặc là trỏ tới các địa chỉ bộ nhớ mà nội dung sẽ được các chương trình nạp vào sau này cung cấp.

- **Kiểm tra xem có ROM mở rộng không**

Khi bổ sung thêm vào hệ thống máy tính một (hay một số) thiết bị ngoại vi tuỳ chọn nào đó, không phải là thiết bị ngoại vi chuẩn, thiết bị này sẽ không được các chương trình phục vụ có sẵn trong ROM-BIOS hỗ trợ, có nghĩa là nó sẽ không làm việc được. Vì vậy các nhà sản xuất thiết bị ngoại vi này thường ghi sẵn chương trình phục vụ thiết bị của mình vào trong một chip ROM, lắp trên card điều khiển bán kèm theo thiết bị. Chip ROM này được gọi là ROM mở rộng, các chương trình phục vụ trong đó bổ sung thêm vào tập các chương trình phục vụ trong ROM-BIOS. Cũng có khi chúng là các chương trình cải tiến, thay thế cho các chương trình đã có trong ROM-BIOS.

Trong họ máy IBM PC người ta đã đưa ra các quy định về ROM mở rộng. Trong đó có các quy định về việc chương trình trong ROM mở rộng phải bắt đầu tại các địa chỉ nào, tên nhà sản xuất ROM mở rộng ghi ở đâu.v.v.

Nếu POST phát hiện thấy có ROM mở rộng thì POST sẽ chuyển điều khiển cho nó để nó tự khởi động lấy, sao cho chương trình trong ROM mở rộng sau này luôn sẵn sàng làm việc. Chương trình trong ROM mở rộng sau khi tự khởi động luôn phải trả lại điều khiển cho chương trình POST để POST tìm các ROM mở rộng khác và làm nốt các công việc còn lại.

- Gọi chương trình **tải Boot-strap**

Công việc cuối cùng của chương trình POST là gọi một chương trình con có nhiệm vụ nạp thành phần tiếp theo của hệ điều hành đang nằm trên đĩa vào bộ nhớ, chương trình con này thường được gọi là chương trình *tải Boot-strap*. Đó là một chương trình ngắn trong ROM thực hiện đọc một sector đặc biệt trên đĩa gọi là ***bản ghi khởi động (boot record)***, nếu thành công thì sẽ chuyển điều khiển cho chương trình trong bản ghi này, nếu thất bại sẽ đưa ra màn hình thông báo. Nếu chương trình tải Boot-strap đọc thành công bản ghi khởi động thì chương trình ở bản ghi này sẽ được cho chạy, nó sẽ nạp thành phần tiếp theo của hệ điều hành vào bộ nhớ. Quá trình như vậy được lặp lại cho tới khi tất cả các thành phần cần thiết của hệ điều hành được nạp vào bộ nhớ, sau đó hệ điều hành sẽ làm việc.

9.4.1.2 ROM-BIOS

ROM-BIOS (Basic Input Output System) là một phần của ROM, chứa các chương trình phục vụ các thao tác vào (Input), ra (Output) sơ đẳng và cơ bản nhất của hệ thống máy tính, nó luôn luôn được sử dụng tích cực trong suốt thời gian máy tính làm việc.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Phần lớn nội dung của ROM-BIOS là các chương trình con xử lý ngắt, các chương trình này điều khiển các thiết bị ngoại vi chuẩn của máy tính như: màn hình, bàn phím, máy in, ổ đĩa, truyền tin không đồng bộ.v.v. Trong ROM-BIOS còn chứa các chương trình quản lý ngày, giờ.v.v.

Các chương trình của ROM-BIOS chỉ dành cho những người lập trình hệ thống chứ không dành cho người sử dụng bình thường, bởi vì chúng rất sơ đẳng, sử dụng không thuận tiện, chẳng hạn không có khả năng xử lý tệp.

9.4.1.3 ROM mở rộng

Chứa các chương trình phục vụ các thiết bị ngoại vi bổ sung. Các chương trình này có mức độ sơ đẳng ngang các chương trình trong ROM-BIOS. Chúng là các chương trình bổ sung, mở rộng hoặc sửa đổi các chương trình đã có trong ROM-BIOS.

9.4.2 Record khởi động

Đây là sector vật lý đầu tiên trên đĩa, vị trí của nó trên đĩa cứng và đĩa mềm là như nhau:

Mặt đĩa (Head/Side)	=	0
Rãnh/Từ trụ (Track/cylinder)	=	0
Cung (Sector)	=	1

- ***Sector vật lý***

Sector vật lý được chỉ ra bằng 3 tham số như trên, các tham số này còn được gọi là địa chỉ vật lý hoặc toạ độ vật lý của sector. Trên mỗi rãnh các sector vật lý được đánh số thứ tự bắt đầu từ 1.

- ***Sector logic***

Hệ điều hành DOS có thể tổ chức một ổ đĩa cứng như một số ***ổ đĩa logic***. Trong mỗi ổ đĩa logic DOS gán cho mỗi sector một số thứ tự, gọi là ***số thứ tự sector logic hoặc địa chỉ logic***, bắt đầu từ số 0, thường viết tắt là Relsec (Relative Sector number). Như vậy sector logic là các sector được gán số thứ tự logic trong các ổ đĩa logic. Có thể tính toán để chuyển đổi từ địa chỉ logic sang địa chỉ vật lý và ngược lại, công thức tính được trình bày trong mục "Truy cập bằng địa chỉ logic của các sector nhờ DOS" ở phần sau của chính chương này.

- ***Record khởi động - Boot Record***

Record khởi động của đĩa mềm và của ổ đĩa logic có chức năng giống nhau, đó là tìm và nạp thành phần tiếp theo của hệ điều hành, file IBMBIO.COM nằm trên đĩa, vào bộ nhớ và trao điều khiển cho nó. Việc này thực tế cũng không phải là đơn giản bởi vì khi chương trình trong boot record chạy, nó chỉ có thể sử dụng các lệnh máy và các dịch vụ sở đẳng trong ROM-BIOS. Bởi vì chưa có các dịch vụ file nên khi cần đọc đĩa, nó phải sử dụng dịch vụ đọc sector theo địa chỉ vật lý của ROM-

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

BIOS, phải thao tác trực tiếp với bảng FAT và bảng thư mục gốc để lần ra dãy liên tiếp các sector vật lý chứa nội dung file IBMBIO.COM, nạp chúng vào bộ nhớ rồi trao điều khiển. (Các vấn đề về tổ chức thông tin trên đĩa, trong đó có bảng FAT và Bảng thư mục gốc sẽ được trình bày ở phần sau của chương này.)

Nếu chương trình tải Boot-strap của ROM không đọc được record khởi động trên ổ đĩa A, nó sẽ tìm trên ổ đĩa C, nếu cũng không đọc được thì sẽ khởi động ROM BASIC (các máy hiện nay không có ROM BASIC). Trong một số máy IBM PC và tương thích, chương trình tải Boot-strap cho phép lựa chọn nạp hệ điều hành từ đĩa mềm (A:) hay từ đĩa cứng (C:) hay thậm chí từ CD-ROM thông qua khai báo được ghi lại ở RAM CMOS.

Chú ý virus khởi động:

Record khởi động là thành phần đầu tiên nằm trên đĩa của hệ điều hành được nạp vào bộ nhớ và trao điều khiển. Trong quá trình máy tính hoạt động, nó có thể bị các chương trình chạy trên máy sửa đổi, thậm chí thay thế. Nếu điều này xảy ra, trong lần khởi động sau chương trình thay thế hoặc đã sửa đổi này sẽ hoạt động theo ý đồ của người đã viết ra nó. Các harker thường làm như vậy, họ copy nội dung trong boot record cất đi đâu đó mà chỉ có họ biết, sau đó ghi vào record khởi động một chương trình virus. Chương trình virus này khi được chạy sẽ tìm cách thường trú (ẩn nấp đâu đó trong bộ nhớ) và thực hiện một số hành động do harker lập trình trước (thí dụ: lây lan, phá hoại...). Sau cùng nó sẽ đọc boot record "nguyên thuỷ" mà nó biết nơi cất giấu vào bộ nhớ rồi cho chạy, để chính chương trình "nguyên thuỷ" này nạp hệ điều hành.

- **Master Boot Record (MBR)**

Với đĩa cứng, DOS có thể chia một ổ đĩa thành một số phần, trên mỗi phần người sử dụng có thể cài đặt một hệ điều hành khác nhau. Trên mỗi phần do DOS quản lý, nó có thể tổ chức thông tin như trên một ổ đĩa độc lập duy nhất. Các ổ đĩa này được gọi là ổ đĩa logic, mỗi ổ được DOS gán cho một tên là một chữ cái, bắt đầu từ C, kết thúc là Z. (Thực tế cách tổ chức của DOS phức tạp hơn thế nhiều, các độc giả muốn tìm hiểu kỹ hơn có thể tham khảo cuốn sách điện tử "Techhelp" chẳng hạn.)

Tổ chức thông tin trên ổ đĩa logic giống tổ chức thông tin trên đĩa mềm, trong đó sector đầu tiên được gọi là sector khởi động (boot sector/boot record). Tuy nhiên với đĩa cứng, record khởi động thực tế là Master Boot Record (MBR), nó ***không thuộc một ổ đĩa logic nào cả***, nó chứa ***Chương trình khởi động chủ***. Chương trình này nằm gọn trong một sector 512 byte, có nhiệm vụ chính là xác định xem trong các phần khác nhau của ổ đĩa vật lý (mỗi phần thường được gọi là một partition), phần nào đã được đánh dấu là tích cực (active) thì nạp hệ điều hành trên phần đó vào bộ nhớ và trao điều khiển. Việc nạp này thực tế được chia thành nhiều bước, bước đầu tiên là đọc boot sector trong ổ đĩa logic vào bộ nhớ rồi cho chạy chương trình trong đó. Chính chương trình này sẽ nạp thành phần tiếp theo vào bộ nhớ và

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

trao điều khiển.v.v. quá trình nạp hệ điều hành từ bước này trở đi giống quá trình nạp hệ điều hành từ đĩa mềm.

- **Một số điểm cần chú ý về MBR:**

Chương trình khởi động chủ trong MBR có nhiệm vụ xác định xem sẽ nạp hệ điều hành từ phần nào (nói đơn giản hơn là từ ổ đĩa logic nào) trên đĩa cứng, trong khi đó mỗi phần có thể đã được cài đặt một hệ điều hành khác nhau, cho nên chương trình khởi động chủ về nguyên tắc không phải là riêng cho một hệ điều hành cụ thể nào. Nếu chúng ta phân chia đĩa bằng chương trình FDISK của DOS, nó sẽ ghi chương trình khởi động chủ vào MBR, chương trình này không phụ thuộc vào version của DOS. Nếu chúng ta phân chia đĩa bằng các phần mềm khác thì các chương trình này sẽ ghi vào MBR các chương trình khởi động chủ của chúng. Tuy nhiên nhiệm vụ chính của chương trình khởi động chủ không thay đổi.

9.4.3 IBMBIO.COM

IBMBIO.COM là file đầu tiên trên đĩa, được ghi trên các sector liên tiếp. Tên của nó cũng phải là tên file đầu tiên trong thư mục gốc. DOS quy định như vậy nhằm giúp cho chương trình trong boot record dễ dàng tìm, đọc file này ngay khi các chương trình dịch vụ về file của hệ điều hành còn chưa được nạp vào bộ nhớ. File này được DOS đặt thuộc tính chỉ đọc (Read Only), ẩn (Hidden) và hệ thống (System).

Nhiệm vụ của IBMBIO.COM là mở rộng ROM-BIOS, nghĩa là bổ sung, thay thế hoặc sửa đổi các chương trình phục vụ các thao tác vào/ra sơ đẳng đã có trong ROM-BIOS. Các chương trình phục vụ trong IBMBIO.COM có mức độ sơ đẳng tương đương các chương trình trong ROM-BIOS. Vì IBMBIO.COM được lưu trên đĩa nên nó là một phương pháp tiện lợi để thay đổi ROM-BIOS mà không cần thay chip ROM của máy. Bất kỳ khi nào có một thiết bị mới được thêm vào máy tính thì các chương trình hỗ trợ cho nó phải được đưa thêm vào File IBMBIO.COM hoặc phải có các chương trình điều khiển thiết bị được đưa vào ROM mở rộng để tránh cho việc phải thay đổi ROM-BIOS.

Khi IBMBIO.COM được nạp vào bộ nhớ và trao điều khiển, nó sẽ tìm file CONFIG.SYS, là file chứa một số thông tin cấu hình của hệ thống và trong đó có chứa các khai báo tên các ***trình điều khiển thiết bị ngoại vi bổ sung - driver*** cần được nạp vào bộ nhớ. Các trình điều khiển này là sự mở rộng, bổ sung hoặc sửa đổi các trình điều khiển trong ROM-BIOS và trong IBMBIO.COM. Tên các trình điều khiển trong file CONFIG.SYS được viết sau khai báo: DEVICE =.

Vì IBMBIO.COM phải tìm đọc file CONFIG.SYS trong khi chưa có các dịch vụ file của DOS, cho nên DOS đòi hỏi file CONFIG.SYS phải có trong thư mục gốc, như vậy việc tìm và đọc sẽ dễ dàng hơn.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Sau khi tìm, đọc và nạp các trình điều khiển được khai báo trong CONFIG.SYS, chương trình trong IBMBIO.COM tìm, đọc và nạp file IBMDOS.COM vào vùng kế tiếp với nó trong bộ nhớ và trao điều khiển.

- **File CONFIG.SYS**

Là một file dạng text có chứa các lệnh để định cấu hình (configure) cho các thành phần phần cứng của máy tính (bộ nhớ - memory, bàn phím - keyboard, chuột - mouse, máy in - printer.v.v.) sao cho MS-DOS và các chương trình ứng dụng có thể sử dụng được chúng.

- **File DBLSPACE.BIN**

Là một phần của MSDOS, nhờ nó có thể truy nhập được ổ đĩa nén. IBMBIO.COM nạp file này cùng với các thành phần khác của hệ điều hành trước khi thực hiện các lệnh trong Config.sys và Autoexec.bat. DBLSPACE.BIN luôn luôn được tải vào đỉnh của vùng nhớ quy ước. Khi IBMBIO.COM thực hiện các lệnh trong CONFIG.SYS nó sẽ tải trình điều khiển thiết bị DBLSPACE.SYS vào bộ nhớ, trình này khi chạy sẽ chuyển DBLSPACE.BIN từ đỉnh của vùng nhớ quy ước tới vị trí cuối cùng trong bộ nhớ , sát với vùng nhớ mà IBMBIO.COM đã nằm.

(Khi chúng ta chạy chương trình DBLSPACE.EXE, nó sẽ tự động bổ sung khai báo DEVICE=DBLSPACE.SYS vào file CONFIG.SYS, trình điều khiển này không tạo cho ta khả năng truy nhập đến ổ đĩa nén, nó chỉ xác định vị trí cuối cùng trong bộ nhớ của DBLSPACE.BIN.)

9.4.4 IBMDOS.COM

IBMDOS.COM được ghi trên các sector liên tiếp sau vùng dành cho file IBMBIO.COM. Tên của nó cũng phải là tên file thứ hai, sau tên file IBMBIO.COM trong thư mục gốc. DOS quy định như vậy nhằm giúp cho chương trình trong IBMBIO.COM dễ dàng tìm, đọc file này ngay khi các chương trình dịch vụ về file của hệ điều hành (nằm chính trong IBMDOS.COM) còn chưa được nạp vào bộ nhớ. File này cũng được DOS đặt thuộc tính chỉ đọc (Read Only), ẩn (Hidden) và hệ thống (System).

IBMDOS.COM được IBMBIO.COM nạp vào bộ nhớ ngay sau vùng mà IBMBIO.COM nằm (nếu trình quản lý ổ đĩa nén DBLSPACE.BIN đã được nạp vào trước đó thì nó nằm sau DBLSPACE.BIN).

IBMDOS.COM là hạt nhân của hệ điều hành DOS, nó chứa hầu như tất cả các dịch vụ của DOS, trong đó có tất cả các dịch vụ về file. Mỗi chương trình dịch vụ này có thể được các chương trình khác gọi bằng chỉ thị ngắt - INT. Ngắt quan trọng nhất là ngắt 21H, nó bao gồm hầu như tất cả các dịch vụ (function/service) của DOS.

IBMDOS.COM quản lý file và I/O ở mức trung gian. Khi các chương trình ứng dụng gọi các chức năng của DOS, IBMDOS.COM nhận các yêu cầu ở dạng mức cao

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

rồi dịch thành một hoặc nhiều lời gọi đến các chương trình phục vụ ở mức sơ đẳng hơn trong IBMBIO.COM để thực hiện.

Các file IBMBIO.COM và IBMDOS.COM chứa các chương trình phục vụ sơ đẳng của hệ thống và của hệ điều hành, người ta cố gắng viết chúng thật nhỏ (và tất nhiên chúng chạy càng nhanh càng tốt) để chiếm ít bộ nhớ, dành chỗ trống cho các chương trình của người sử dụng.

9.4.5 COMMAND.COM

Thành phần này còn có tên gọi là ***Bộ thông dịch lệnh (Command Interpretter),*** trong đó có chứa tất cả các lệnh nội trú (thường trú) của hệ điều hành DOS. Nó thực hiện nhiệm vụ hiển thị thông báo chờ lệnh trên màn hình, đọc và dịch các câu lệnh nhập vào từ bàn phím để hệ thống máy tính thi hành.

COMMAND.COM được nạp vào bộ nhớ và nằm thường trực trong đó cùng với IBMDOS.COM và các trình điều khiển thiết bị khai báo trong CONFIG.SYS. Chương trình này đóng vai trò giao tiếp giữa người sử dụng và hệ thống máy tính, do đó nó được viết ra theo khuynh hướng có giao diện càng thuận tiện cho người sử dụng càng tốt. Đây cũng là một trong các lý do chính làm cho kích thước của COMMAND.COM tăng lên không ngừng theo các phiên bản (version) của hệ điều hành DOS. Khi phần bộ nhớ mà hệ điều hành DOS chiếm tăng lên thì phần còn lại cho các chương trình chạy trong môi trường DOS sẽ giảm đi tương ứng, hệ quả là một số chương trình ứng dụng có thể sẽ không có đủ bộ nhớ để chạy nếu dùng phiên bản DOS mới.

Để tránh sự cố trên, người ta tạo ra command.com sao cho khi được nạp vào bộ nhớ, nó được tách ra thành 2 phần: thường trú và lưu trú (bán trú).

- Phần thường trú

Phần thường trú được nạp vào bộ nhớ kế tiếp các thành phần đã được nạp vào trước của DOS, sau đó được bảo vệ chống ghi đè. DOS không phân phối cho các chương trình vùng nhớ được nó bảo vệ. Phần thường trú chứa các chương trình đảm bảo thực hiện các chức năng tối quan trọng của command.com và có kích thước tương đối nhỏ.

- Phần bán thường trú (lưu trú)

Phần bán trú được nạp vào bộ nhớ ở vùng địa chỉ cao nhất, sát vùng dành cho ROM, không được bảo vệ chống ghi đè. DOS cho phép các chương trình toàn quyền sử dụng vùng nhớ mà nó không bảo vệ, bao gồm cả vùng chứa phần bán trú của chính nó.

Khi một chương trình ứng dụng chạy, nó có thể ghi đè lên phần bán trú của command.com; Khi chương trình này kết thúc, phần thường trú sẽ phát hiện ra, khi đó nó sẽ đọc file command.com trên đĩa để khôi phục lại phần bán trú của mình trong bộ nhớ. (Vị trí của file command.com trong những lần đọc như vậy có thể

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

không phải ở thư mục gốc của ổ đĩa khởi động, người sử dụng có thể khai báo đường dẫn tới một file command.com ở chỗ khác.)

Command.com sau khi được nạp vào bộ nhớ và trao điều khiển sẽ tìm file autoexec.bat tại thư mục gốc của đĩa khởi động, nếu thấy thì cho chạy các chương trình khai báo trong đó.

Việc cuối cùng của command.com là hiển thị thông báo chờ lệnh trên màn hình; Lúc này quá trình nạp hệ điều hành đã hoàn thành, máy đã khởi động xong, sẵn sàng nhận lệnh và thi hành.

9.5 TRUY CẬP IBM PC

9.5.1 Các thông tin cấu hình máy trong RAM CMOS

Họ máy IBM-PC/AT có một đồng hồ thời gian thực chạy pin (RTC-Real Time Clock) và một bộ nhớ RAM dung lượng 64 byte chế tạo theo công nghệ CMOS - tiêu thụ ít năng lượng, thường được gọi là RAM CMOS cũng chạy bằng nguồn pin trên. Trong bộ nhớ này có chứa nhiều thông tin khác nhau, trong đó có ngày, giờ, cấu hình phần cứng.v.v.

Khi máy đang ở trong quá trình POST, nếu chương trình POST phát hiện thấy một số thông số phần cứng trong hệ thống không khớp với các thông tin được lưu trong RAM CMOS thì nó sẽ nhắc ta chạy chương trình setup để khai báo lại cấu hình cho đúng. Bảng 9-02 là bản đồ bộ nhớ RAM CMOS.

Cách truy cập dữ liệu trong RAM CMOS

Để đọc một byte trong RAM CMOS, trước hết ta phải gửi địa chỉ của byte đó ra cổng 70H bằng lệnh OUT, sau đó nhận nội dung của byte mà ta vừa yêu cầu thông qua cổng 71H bằng lệnh IN.

Để ghi vào một byte trong RAM CMOS, trước hết ta phải gửi địa chỉ của byte đó ra cổng 70H bằng lệnh OUT, sau đó gửi giá trị (kích thước 1 byte) qua cổng 71H bằng lệnh OUT.

Thí dụ để đọc kiểu của đĩa cứng (HDD), thực hiện đoạn chương trình sau:

```
mov  al,12H    ;kiểu HDD được ghi ở byte thứ 12H của RAM CMOS.
out  70H,al    ;báo cho RAM CMOS sẽ truy cập byte thứ 12H
jmp  $+2       ;thực hiện delay, vì RAM CMOS có tốc độ thấp
in   al,71H    ;nhận giá trị (0-15) kiểu HDD vào AL
```

(Đây chỉ là một thí dụ minh hoạ việc đọc từ RAM CMOS, các đĩa cứng ngày nay đều có kiểu >15, do đó chúng ta sẽ phải đọc các ô nhớ 19h hoặc 1Ah.)

Trong bộ nhớ RAM CMOS, miền nhớ từ địa chỉ 10H tới 20H được bảo vệ bởi kỹ thuật checksum, là một kỹ thuật phát hiện lỗi. Khi chúng ta setup máy, chương trình setup ghi các thông tin cấu hình mà chúng ta khai báo hoặc nó phát hiện được vào RAM CMOS, đồng thời nó tính tổng của tất cả các byte trong miền cần bảo vệ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

theo một thuật toán đặc biệt (không phải phép cộng thông thường), kết quả nhận được 1 word, gọi là word checksum, được ghi vào địa chỉ 2EH và 2FH của RAM CMOS.

Mỗi lần khởi động máy, chương trình POST tính lại checksum rồi so sánh với word checksum được ghi từ trước. Nếu hai con số khác nhau thì chứng tỏ rằng nội dung vùng nhớ được bảo vệ đã bị thay đổi vì một nguyên nhân nào đó, không phải do chương trình setup thực hiện. Nguyên nhân thay đổi có thể do pin nuôi RAM CMOS bị kiệt hoặc các thông tin sai lạc bị viết vào miền này, chương trình POST sẽ thông báo cho người sử dụng (chương trình POST trong một số máy PC cho chạy luôn trình setup mà không đòi hỏi mật khẩu).

Bảng 9-02 Diễn giải nội dung bộ nhớ RAM CMOS

Địa chỉ	Nội dung	
00H-0dH	Đồng hồ hệ thống (real-time clock)	
0eH	Byte trạng thái chẩn đoán của chương trình POST	
0fH	Byte trạng thái lúc tắt máy (shutdown status)	
10H	Kiểu FDD	Miền được tính checksum để phát hiện sai (10H-20H)
11H	Dự trữ (reserved)	
12H	Kiểu HDD, nếu < 15)	
13H	Dự trữ (reserved)	
14H	Byte danh sách thiết bị	
15H-16H	Kích thước bộ nhớ cơ sở	
17H-18H	Kích thước bộ nhớ mở rộng trên 1MB	
19H	Kiểu ổ đĩa cứng C, nếu > 15)	
1aH	Kiểu ổ đĩa cứng D, nếu > 15)	
1bH-20H	Dự trữ (reserved)	
21H-2dH	Dự trữ (reserved)	
2eH-2fH	Chứa word checksum của miền 10H..20H	
30H-31H	Kích thước bộ nhớ mở rộng trên 1MB	
32H	Thế kỷ dạng số BCD (thí dụ, 19H)	
33H	Các thông tin linh tinh khác	
34H-3fH	Dự trữ (reserved)	

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

9.5.2 Tổ chức thông tin trên đĩa mềm

Tổ chức vật lý của đĩa mềm đã được nghiên cứu ở chương 8 Các thiết bị ngoại vi; Tại mục này chúng tìm hiểu cách mà hệ điều hành DOS tổ chức thông tin (ta sẽ gọi là tổ chức logic) trước khi tìm hiểu việc truy cập thông tin trên đĩa.

Về tổ chức vật lý, vị trí của mỗi sector được xác định bằng 3 tham số: mặt đĩa (head/side), cung (sector) và rãnh (track/cylinder); các sector không có mối quan hệ logic với nhau, không có khái niệm file. Truy cập đĩa theo tổ chức vật lý là một cách khó, phải sử dụng các dịch vụ của ROM-BIOS.

Theo cách tổ chức logic của DOS, vị trí của mỗi sector được xác định chỉ bằng một tham số, đó là số thứ tự của sector trong ổ đĩa logic, tính bắt đầu từ 0. Dưới đây chúng ta sẽ ký hiệu số thứ tự này là Relsec. Khi truy cập đĩa ở mức sector bằng các dịch vụ của DOS, chúng ta chỉ cần tham số Relsec, chính DOS sẽ tính ra vị trí vật lý của sector này trên đĩa, việc chuyển đổi này DOS thực hiện theo công thức sau:

RelSec = CylNo*SecsPerTrack*Heads)+(HeadNo*SecsPerTrack)+(SecNo-1)

Các tham số:

Heads	Số đầu từ/mặt đĩa
HeadNo	Số hiệu mặt đĩa (địa chỉ đầu từ/mặt đĩa theo toạ độ vật lý)
SecNo	Số hiệu sector (địa chỉ sector theo toạ độ vật lý)
CylNo	Số hiệu từ trụ/rãnh (địa chỉ cylinder theo toạ độ vật lý)
SecsPerTrack	Số lượng sector trên 1 rãnh

Với một ổ đĩa được DOS tổ chức (được format bằng DOS), các tham số Heads và SecsPerTrack là các hằng xác định, khi cho Relsec có thể tính được (HeadNo, SecNo, CylNo) một cách dễ dàng và ngược lại cũng dễ như vậy vì các tham số trên đều là các số nguyên, nằm trong một miền xác định.

DOS định khuôn dạng đĩa theo quy định của DOS (sử dụng chương trình FORMAT), vì vậy DOS có thể không đọc được các đĩa được định dạng theo các quy định khác, thí dụ đĩa của máy Macintosh. Tuy nhiên nếu sử dụng các dịch vụ của ROM-BIOS thì có thể đọc một cách vật lý số liệu trên nhiều loại đĩa khác nhau. Nhờ các dịch vụ của ROM-BIOS, có thể thực hiện việc chống sao chép số liệu trên đĩa bằng cách thay đổi tham số của đĩa (số byte/sector, sector/track, hoặc địa chỉ của cung hay rãnh.v.v...) làm cho khác với chuẩn của DOS.

Các vùng thông tin trên đĩa: Chương trình FORMAT định khuôn dạng đĩa thành các phần như trên hình 9-08.

9.5.2.1 Record khởi động

Địa chỉ vật lý của record khởi động trên đĩa mềm:

Head/Side	= 0
Track/Cylinder	= 0
Sector	= 1

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Record khởi động chủ yếu chứa một chương trình ngắn để khởi động quá trình nạp hệ điều hành DOS từ đĩa mềm. Tất cả các đĩa mềm đều có record khởi động dù chúng không chứa hệ điều hành, tức không phải là đĩa khởi động. Nội dung một số phần của nó thay đổi tuỳ theo khuôn dạng đĩa. Ngoài chương trình khởi động ra trên record khởi động còn có chứa một số thông tin khác: Bảng tham số đĩa mềm, chữ ký của hệ điều hành.v.v.

Sector khởi động (Boot sector) Chiếm 1 sector
Các sector để dành DOS dự trữ cho các phát triển trong tương lai; Nếu sử dụng DOS 6.22 hoặc các phiên bản cũ hơn, số lượng sector này bằng 0
Bảng FAT (có hai bảng FAT) Bảng FAT chiếm một số sector, kích thước phụ thuộc vào dung lượng ổ đĩa logic và kích thước đơn vị đĩa mà DOS sử dụng (trang, cluster)
Bảng thư mục gốc (Root Directory) Chiếm một số sector, phụ thuộc vào dung lượng ổ đĩa logic
Phần còn lại để phân phối cho các file dữ liệu Được chia thành các trang (cluster), là đơn vị đĩa mà DOS cấp phát cho các file. Mỗi trang có thể gồm một số sector (2 ở đĩa 360KB, 1 ở đĩa 1.2MB và 1.44MB)

Hình 9-08 Các vùng thông tin trên đĩa mềm

Dưới đây là một số nội dung của record khởi động:

- Khối tham số đĩa mềm dùng cho BIOS (BIOS PARAMETER BLOCK-BPB).
- Số nhận diện đĩa - Serial Number, được ghi ở 4 byte trên record khởi động, bắt đầu từ byte thứ 27H. Thí dụ nếu ta đọc 4 byte liên tiếp đó vào bộ nhớ và chúng có giá trị như sau: CB 13 25 37 (số hexa), nếu dùng lệnh LABEL của DOS thì nó hiện Serial Number là : 3725-13CB.
- Chương trình khởi động, có nhiệm vụ nạp các phần còn lại của hệ điều hành vào bộ nhớ của máy.
- Chữ ký của hệ điều hành 55 AA ở 2 byte cuối cùng (số hệ 16).

Khối tham số đĩa mềm - BPB

Trên các đĩa mềm được format bằng DOS trước 4.0, BPB gồm 13 byte, từ byte 0Bh tới 17h của record khởi động, nếu đĩa được format bằng DOS từ 4.0 trở đi, BPB sẽ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

được bổ sung thêm 12 byte nữa, ngay sau vùng 13 byte nói trên. Trong 25 byte của BPB có chứa các trường thông tin, ý nghĩa của chúng cho trong bảng 9-03. Bảng 9-04 là thí dụ về một khối tham số đĩa mềm cụ thể.

Bảng 9-03 BPB - Khối tham số đĩa mềm

Địa chỉ byte	Kích thước trường	Ý nghĩa
+0	2	Kích thước sector tính bằng byte (DOS chọn =512)
+2	1	Kích thước trang (số sector trong 1 cluster)
+3	2	Tổng số sector để dành và sector khởi động
+5	1	Số bảng FAT, DOS luôn chọn =2
+6	2	Số cực đại đề mục thư mục trong thư mục gốc, mỗi đề mục có kích thước 32 byte
+8	2	= Tổng số sector trong đĩa, nếu đĩa <32MB = 0 nếu đĩa >=32MB, tổng ghi bắt đầu tại byte thứ 15H
+0AH	1	Byte mô tả môi trường đĩa (Media descriptor), giống byte FAT_ID (được mô tả sau)
+0BH	2	Kích thước một bảng FAT tính bằng đơn vị sector
+0DH	2	Số sector trên một track
+0FH	2	Số đầu đọc/ghi (= số mặt đĩa)
+11H	4	Số sector ẩn (DOS trước 4.0 chỉ dùng word thấp)
+15H	4	Tổng số sector trong đĩa, nếu đĩa <32MB

Bảng tham số đĩa mềm (Diskette Parameter Table)

Đây là một bảng số liệu kích thước 11 byte, mô tả các thông số của các ổ đĩa, giá trị của bảng này được xác lập bằng trình Setup của ROM-BIOS (hoặc bằng các chuyển mạch đặt cấu hình hệ thống ở các thế hệ máy IBM PC đầu tiên). Vị trí của bảng được trỏ tới bởi vector ngắt 1EH, vì thế vector này còn được gọi là Con trỏ bảng tham số đĩa mềm (Diskette Parameter Pointer).

Mỗi khi chúng ta cho một đĩa mềm mới vào ổ đĩa để làm việc, DOS đều đọc record khởi động, lấy thông tin trong Khối tham số đĩa mềm (BPB) để cập nhật cho Bảng tham số đĩa mềm, đảm bảo cho các thao tác với đĩa mềm này phù hợp với các tham số mà nó đã được định dạng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Bảng tham số đĩa mềm có thể được DOS hoặc người sử dụng sửa đổi, thậm chí thay thế để cải thiện các thao tác đĩa hoặc thực hiện việc bảo vệ thông tin chống sao chép.

Bảng 9-04 Thí dụ về một khối tham số đĩa mềm

Địa chỉ byte	Kích thước trường	Nội dung	Ý nghĩa
+0	2	0200H	Kích thước sector = 200H = 512 byte
+2	1	01	Kích thước trang = 1 sector
+3	2	0001	Tổng số sector để dành và sector khởi động =1, nghĩa là số sector để dành =0.
+5	1	02	Số bảng FAT = 2
+6	2	00E0H	Số đề mục cực đại = E0H = 224 đề mục 32 byte
+8	2	0B40H	Tổng số sector trong đĩa = B40H = 2880 sector
+0AH	1	F0H	Đây là đĩa mềm 31/2", 2 mặt, 80 rãnh/1 mặt, 18 sector/rãnh
+0BH	2	0009	Một bảng FAT chiếm 9 sector
+0DH	2	0012H	Số sector trên một track
+0FH	2	0002	Số mặt đĩa sử dụng = 2
+11H	4	0000	Không quan tâm
+15H	4	0000	Không quan tâm

(số theo hệ hexa có chữ H đi sau)

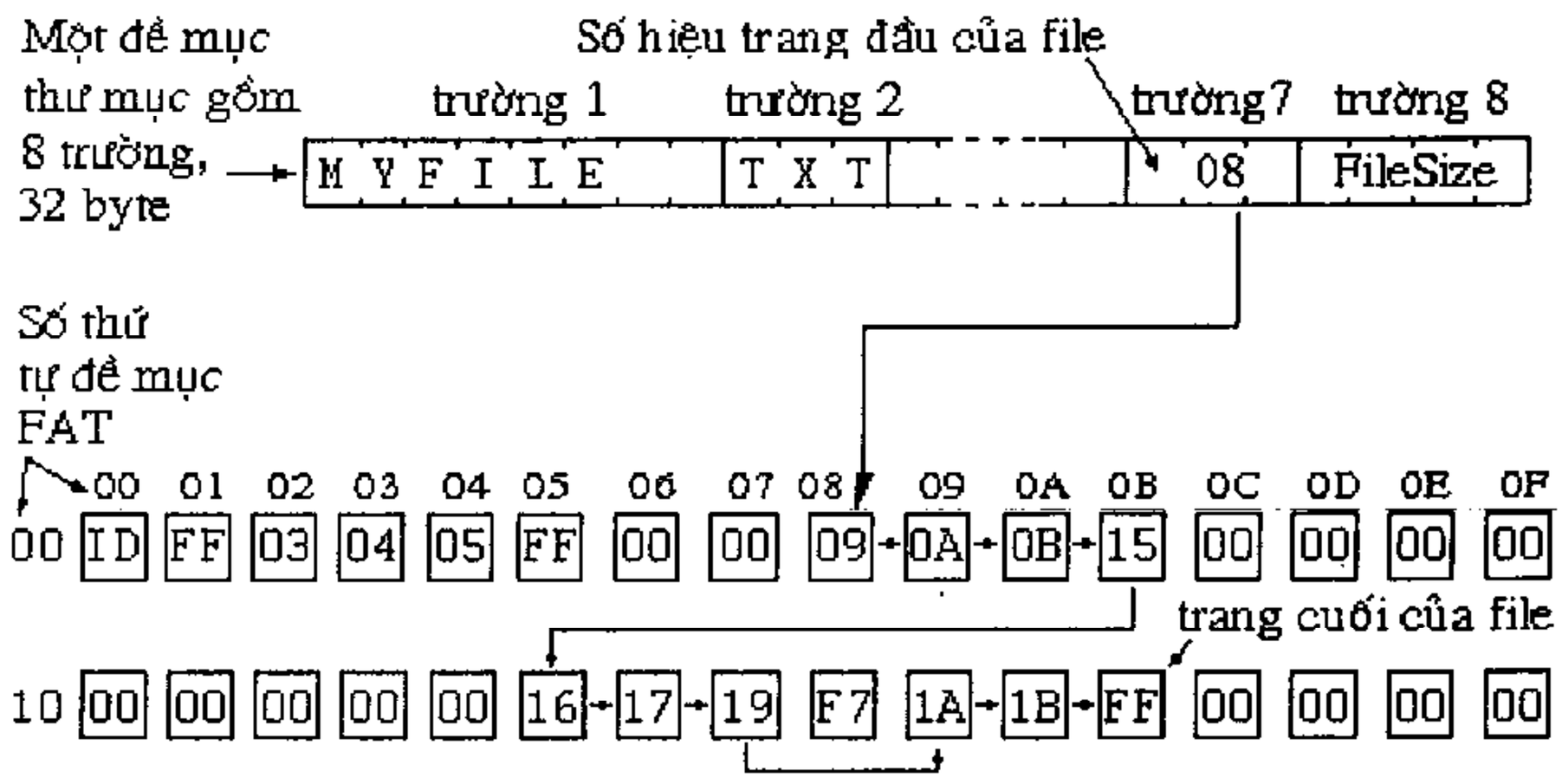
9.5.2.2 Bảng FAT (File Allocation Table)

FAT là một bảng danh sách móc nối mà DOS sử dụng cùng với bảng thư mục để theo dõi sát vị trí vật lý của dữ liệu (file) trên đĩa và để sắp đặt các chỗ trống để lưu trữ các file mới. FAT nằm ngay sau Sector khởi động (và các sector để dành nếu có). Trên đĩa DOS bố trí 2 bảng FAT nằm liên tiếp trên các sector đầu tiên, nội dung của 2 bảng này giống hệt nhau. DOS làm như vậy vì FAT rất quan trọng, các trục trặc của FAT sẽ làm ta mất (không truy nhập được) thông tin trên đĩa.

- Với đĩa 360 KB, 1 bảng FAT chiếm 2 sector
- Với đĩa 1.2 MB, 1 bảng FAT chiếm 7 sector
- Với đĩa 1.44 MB - 9 sector

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Hình 9-09 thể hiện cách mà DOS sử dụng bảng FAT và thư mục để quản lý thông tin trên đĩa. Nội dung các đề mục (ô) trong bảng FAT có ý nghĩa như cho trên bảng 9-05.



Hình 9-09 Mối liên hệ giữa đề mục thư mục và bảng FAT

Bảng 9-05 Nội dung đề mục FAT và ý nghĩa

Nội dung	Ý nghĩa
000	Báo cho DOS rằng trang tương ứng còn trống
002-FEFH	Chỉ ra trang tiếp theo của File
FF0H-FF6H	Trang dành cho mục đích khác, không sử dụng được
FF7H	Trang hỏng (bad), DOS không phân phối cho các file
FF8H-FFFH	Trang tương ứng là kết thúc của File

DOS chia phần đĩa để lưu trữ dữ liệu thành các trang (cluster), khi cấp phát các vùng nhớ trên đĩa cho File, DOS cấp từng trang một, một file không nhất thiết phải chiếm các trang liên tục mà có thể chiếm các trang nằm rải rác trên đĩa. Chính vì lý do này, cần phải lưu trữ bản đồ các trang - đó chính là FAT. FAT được chia thành các phần bằng nhau gọi là đề mục (entry), mọi trang trên đĩa đều có một đề mục tương ứng với nó, đề mục chứa một con số chỉ ra số của trang tiếp theo, hoặc một con số biểu diễn các trạng thái khác của trang.

Trừ hai đề mục đầu tiên là đề mục 0 và 1 có ý nghĩa khác biệt sẽ mô tả sau, đề mục thứ hai sẽ gắn với trang đầu tiên trên đĩa có thể cấp phát cho file, đề mục thứ ba sẽ gắn với trang thứ hai.v.v. Chính vì sự tương ứng này mà DOS đánh số thứ tự các trang bắt đầu từ 2 chứ không phải là từ 1 hoặc từ 0.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Số đề mục trong FAT phải lớn hơn hoặc bằng số trang trên đĩa, và kích thước (số bit) của mỗi đề mục FAT phải chứa được số hiệu trang của trang cuối cùng trên đĩa.

Ta có thể hình dung FAT là một cái bảng mà mỗi phần tử là một đề mục. Thông qua nội dung trong các đề mục này mà DOS biết các trang nào còn trống để cấp phát cho các file, các trang đã cấp là thuộc File nào, trang nào bị hỏng không dùng.v.v.

Với đĩa mềm, mỗi đề mục FAT chỉ cần rộng 12 bit (một byte rưỡi) là đủ, vì nó chứa được con số lớn nhất là 4095, lớn hơn tổng số trang có trên đĩa mềm. Với đĩa cứng thì 12 bit là không đủ, vì vậy đề mục FAT trên đĩa cứng cần phải lớn hơn, thường là 16 bit.

Bảng 9-06 FAT ID - Byte nhận diện ổ đĩa

FAT ID	Loại đĩa
F0	3 1/2", đĩa 2 mặt, 80 rãnh, 18 sector/rãnh
F9	3 1/2", đĩa 2 mặt, 80 rãnh, 9 sector/rãnh (720 KB)
F8	Đĩa cứng
F9	5 1/4", đĩa 2 mặt, 80 rãnh, 15 sector/rãnh
FA	5 1/4", đĩa 1 mặt, 80 rãnh, 8 sector/rãnh
FB	5 1/4", đĩa 2 mặt, 80 rãnh, 8 sector/rãnh
FC	5 1/4", đĩa 1 mặt, 40 rãnh, 9 sector/rãnh
FD	5 1/4", đĩa 2 mặt, 40 rãnh, 9 sector/rãnh
FE	5 1/4", đĩa 1 mặt, 40 rãnh, 8 sector/rãnh
FF	5 1/4", đĩa 2 mặt, 40 rãnh, 8 sector/rãnh

FAT ID

Byte đầu tiên trong bảng FAT được gọi là FAT ID (IDentification), DOS sử dụng để nhận diện loại đĩa. Giá trị của byte này và ý nghĩa tương ứng được cho trong bảng 9-06.

9.5.2.3 Bảng thư mục (Directory)

Bảng thư mục chứa danh sách các file và thư mục con *tại thư mục gốc* trên đĩa. Ở đây, hệ thống lưu trữ tất cả mọi thông tin liên quan đến File. Mỗi file (hoặc thư mục con) cần 1 đề mục trong bảng thư mục, mỗi đề mục dài 32 byte, như vậy mỗi sector có thể chứa được 16 đề mục thư mục. Kích thước bảng thư mục thay đổi tuỳ

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

theo dung lượng đĩa như cho trong bảng 9-07. Mỗi đề mục thư mục có thể được chia làm 8 trường, kích thước và ý nghĩa của các trường như trong bảng 9-08.

Bảng 9-07 Kích thước thư mục của một số loại đĩa mềm

Dung lượng	Số Sector của bảng thư mục	Số đề mục tối đa (Root Directory Entry)
360 KB	7	112
1.2 MB	14	224
1.44 MB	14	224

Bảng 9-08 Diễn giải các trường của một đề mục thư mục

Địa chỉ	Kích thước	Nội dung	Ý nghĩa, chú thích
+ 0	8	Tên file	Căn trái, chèn ký tự trống bên phải
+ 8	3	Tên mở rộng	Căn trái, chèn ký tự trống bên phải
+ 0BH	1	Thuộc tính file	01: Read only; 02: Hidden; 04 : system; 08: Volume; 10h: subdir; 20h: Archive.
+ 0CH	0AH	0000000000	Dự trữ, DOS chưa sử dụng, được đặt =0
+ 16H	2	Giờ (Time)	
+ 18H	2	Ngày (Date)	
+ 1AH	2	ClstNo	Số hiệu trang đầu tiên của file
+ 1CH	4	File size	Kích thước file (đơn vị là byte)

Giải thích chi tiết hơn về các trường như sau:

1) Trường tên file

Dài 8 byte, từ byte 0 đến byte 7, nếu tên dài dưới 8 byte, DOS tự động thêm các ký tự trống (32) vào sau tên cho đầy trường này. Khi ta xoá 1 file, DOS chỉ thay ký tự đầu tiên bằng E5H, còn các phần khác của bảng thư mục không thay đổi.

2) Trường tên mở rộng của file

Dài 3 byte, từ byte 8 đến byte10. Nếu File không có phần mở rộng thì DOS điền vào các ký tự trống.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

3) Trường thuộc tính của file

Kích thước 1 byte, là byte thứ 11. Một file có thể có các thuộc tính sau: System, Hidden, Read Only, Archive.v.v... Bảng 9-09 ghi tóm tắt ý nghĩa các bit trong byte thuộc tính file.

- Bit 0: Thuộc tính Read Only - làm cho File không xoá được bằng các lệnh của DOS
- Bit 1: Thuộc tính Hidden - làm cho File không nhìn thấy được khi dùng các lệnh bình thường của DOS như lệnh DIR
- Bit 2: thuộc tính hệ thống (System): không mang ý nghĩa gì đặc biệt, nó tồn tại do kế thừa hệ điều hành CP/M
- Bit 3: thuộc tính nhãn đĩa (Volume ID Lable): nếu bit này = 1 thì DOS hiểu rằng đề mục thư mục này là nhãn đĩa (Volume ID Label); Khi đó đề mục này phải ở thư mục gốc và chỉ sử dụng một số trong số 8 trường thôi. Trong trường hợp này cả hai trường 1 và 2 được dùng để lưu nhãn đĩa và được hiểu như một trường duy nhất, ngoài ra các trường ngày, giờ cũng được sử dụng, còn các trường khác thì không được sử dụng.
- Bit 4: Thuộc tính thư mục con, được dùng để nhận diện đề mục thư mục con, phân biệt với đề mục tên file. Thư mục con được cất giữ trên đĩa giống như một file chứa dữ liệu thông thường, kiểu văn bản (file dạng text), vì vậy cần sự hỗ trợ của một đề mục loại thư mục. Trường hợp này tất cả các trường của đề mục đều được sử dụng, trừ trường kích thước File sẽ bằng 0. Kích thước thật sự của thư mục con chỉ được biết nhờ lần theo chuỗi số hiệu trang tương ứng trong FAT.

Bảng 9-09 Ý nghĩa các bit trong byte thuộc tính file.

Bit	giá trị		Ý nghĩa
7 6 5 4 3 2 1 0	Dec	Hex	
. 1	1	1	Chỉ đọc (Read-only)
. . . . . 1 .	2	2	Ẩn (Hidden)
. . . . . 1 . .	4	4	Hệ thống (System)
. . . . 1 . . .	8	8	Nhãn đĩa (Volume Label)
. . . 1	16	10	Thư mục con (Subdirectory)
. . 1	32	20	Lưu trữ bằng Back Up (Archive)
. 1	64	40	Không dùng
1	128	80	Không dùng

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

- Bit 5: thuộc tính lưu trữ (Achived)
  - Với đĩa cứng: bit 5 = 0 đối với tất cả các tập tin chưa bị sửa đổi lần nào kể từ lần backup gần đây nhất. Nếu ta sửa một file, bit này sẽ được DOS đặt = 1 để lệnh backup nhận biết.
  - Với đĩa mềm: DOS luôn đặt bit này = 1, như vậy thuộc tính này không có ý nghĩa gì với các file trên đĩa mềm.

4) Trường 4: dự trữ

Dài 10 byte, dành cho tương lai. Cả 10 byte được đặt = 00h

5) Trường 5: Giờ (Time)

Dài 2 byte, chứa thời điểm file được tạo ra hoặc sửa đổi gần nhất. Trường thời gian được coi như số nguyên không dấu dài 1 word, bao gồm giờ, phút và giây, được tính theo công thức:

Thời gian = giờ * 2048 + phút * 32 + giây /2 (= 0..23)

Do chỉ sử dụng 1 word nên không đủ bit (thiếu 1 bit) để chứa tất cả các giây nên DOS lưu trữ theo đơn vị 2 giây (0..29). Thí dụ số 5 có nghĩa là 5*2 = 10 giây. Thời gian 11:32:10 sẽ được ghi là 23557.

6) Trường 6: Ngày tháng (Date)

Dài 2 byte, tương tự trường 5.

ngày tháng = (năm - 1980)*512 + tháng*32 + ngày

Công thức này "nén" số năm lại bằng cách trừ đi 1980, ví dụ năm 1984 sẽ được tính là 4. Ngày 12 tháng 12 năm 1984 sẽ được ghi là 0000.1001.1000.1100B (=2444D). Ngày 31/12/1999 sẽ là 0010.0111.1001.1111B = 10143D.

7) Trường 7: Số hiệu trang đầu tiên của file

Dài 2 byte, dựa vào đây DOS (hoặc chúng ta) biết được số của trang đầu tiên của file, nếu đọc đề mục FAT tương ứng (có số hiệu = giá trị của trường này) sẽ lần ra được trang tiếp theo của File, cứ như vậy có thể lần đến trang cuối cùng kết thúc file. Với các file không được cấp trang nào và các đề mục loại nhãn thì DOS đặt trường này = 0, chứ không dùng giá trị FFFH trong FAT để báo hiệu hết File.

8) Trường 8: Kích thước file

Trường này dài 4 byte, chứa số nguyên không dấu, giá trị cực đại của số nguyên 4 byte không dấu này có thể lớn hơn kích thước của cả đĩa.

Thư mục con

Đề mục thư mục con cũng gần giống đề mục tên file, chỉ khác ở giá trị của trường thuộc tính phải bằng 10H và trường 8 (kích thước file) phải bằng 0. File đặc biệt này có dạng văn bản (text), khi cần truy cập thư mục con tương ứng DOS sẽ đọc và xử lý từng khối 32 byte của nó như đối với một đề mục trong bảng thư mục. Tất

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

nhiên một đề mục như vậy có thể lại là một đề mục thư mục con (bên trong thư mục con).

9.5.3 Tổ chức thông tin trên đĩa cứng

Khi nghiên cứu tổ chức thông tin trên đĩa cứng, chúng ta cần quan tâm đến cả tổ chức logic và tổ chức vật lý.

- **Tổ chức vật lý**

Tổ chức vật lý bao gồm các yếu tố sau:

1. số mặt đĩa (head/side)
2. số rãnh (track/cylinder)
3. số sector trên một rãnh (sector per track)
4. Kích thước sector (sector size)

Không giống như đĩa mềm, tổ chức vật lý của đĩa cứng được thiết lập trước khi ta dùng - do hãng sản xuất thực hiện, người sử dụng nói chung không thể thay đổi được tổ chức vật lý của đĩa cứng.

- **Tổ chức logic**

Xác định cách tổ chức thông tin trên đĩa và các loại thông tin khác nhau có thể đặt ở đâu trên đĩa. Tổ chức logic thông tin trên đĩa cứng bao gồm các vấn đề chính sau:

1. Đĩa cứng được chia làm bao nhiêu vùng (partition); mỗi vùng chiếm một khoảng liên tục các Cylinder nào trên đĩa vật lý.

   Việc phân chia đĩa thành các vùng có thể thực hiện bằng chương trình FDISK.EXE hoặc một số phần mềm khác. Mỗi vùng được DOS coi như một đĩa hoàn chỉnh (ổ đĩa logic), có thể có chương trình khởi động riêng ở sector khởi động của nó (sector logic số 0). Các Partition khác nhau có thể được tổ chức theo các tham số khác nhau của hệ điều hành được cài đặt trên đó. Các thông tin liên quan đến việc chia đĩa được ghi ở Master Boot Record (MBR), đó là sector vật lý đầu tiên, địa chỉ = (Head=0, Cyl=0, Sec=1).

2. Phần nào trong số các phần là tích cực (active), tức là sẽ khởi động hệ điều hành được cài đặt trên đó.

   Sau khi khởi động máy tính, chương trình trong MBR sẽ chuyển điều khiển cho record khởi động, là record logic đầu tiên của ổ đĩa logic.

3. Lựa chọn kích thước trang (Cluster) bằng bao nhiêu sector.

   Việc tăng kích thước của trang sẽ làm giảm kích thước của bảng FAT nhưng việc sử dụng đĩa sẽ bị lãng phí hơn vì trang cuối cùng của các file tính chung chỉ được sử dụng 50%. Nếu giảm kích thước trang sẽ có tác dụng ngược lại.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Tổ chức logic thông tin trên đĩa cứng của hệ điều hành DOS thực tế phức tạp hơn những điều được trình bày trên. Bạn đọc có thể tham khảo tài liệu điện tử "Techhelp" để có thêm các thông tin chi tiết.

9.5.4 Truy cập thông tin trên đĩa

Có 3 mức truy cập thông tin trên đĩa, không phân biệt đĩa cứng hay đĩa mềm. Mỗi mức truy cập sử dụng một số dịch vụ thuộc một mức nhất định của hệ điều hành DOS.

- **Truy cập bằng địa chỉ vật lý nhờ dịch vụ của ROM-BIOS**

Tất cả các dịch vụ đĩa của ROM-BIOS đều có thể gọi bằng ngắt 13H; các chương trình con được gọi đều sử dụng các bảng tham số đĩa mà hệ thống lưu trữ. Nếu hệ thống có đĩa cứng thì vector ngắt 41h trỏ tới bảng tham số đĩa cứng, còn vector ngắt 40h trỏ tới bảng tham số đĩa mềm.

Các tham số truyền cho chương trình xử lý ngắt 13H thông qua các thanh ghi như sau:

AH - số hiệu dịch vụ (0 - Reset, 2 - đọc, 3- ghi...)
AL - số lượng sector cho thao tác đọc/ghi
ES:BX - địa chỉ vùng đệm dữ liệu
CX - số hiệu SecCyl
DH - số hiệu đầu từ
DL - số hiệu ổ đĩa vật lý (0: ổ A; 1: ổ B; 80H: ổ đĩa vật lý thứ nhất...)

Chú ý rằng mặc dù AL có thể chứa được giá trị lớn nhất là 0FFH, nhưng với đĩa cứng tổng số sector cực đại truy cập được trong một lần chỉ là 80h, đồng thời cũng không vượt quá số lượng sector trên một rãnh.

- **Truy cập bằng địa chỉ logic sector nhờ DOS**

DOS cung cấp các chương trình dịch vụ đọc ghi đĩa tuyệt đối, không quan tâm đến tổ chức file của thông tin, đó là các chương trình được gọi qua ngắt 25h và 26h, chúng ta viết ngắn gọn là INT 25H và INT 26H. Hai chương trình dịch vụ này có thể đọc/ghi sector có số hiệu bất kỳ theo địa chỉ logic của chúng. Các dịch vụ này dựa trên các dịch vụ cơ sở của ROM-BIOS hoặc dựa trên các trình điều khiển thiết bị (driver) đĩa đã được cài đặt.

Khi sử dụng INT 25H và INT 26H cần truyền tham số qua các thanh ghi như sau:

AL: số hiệu ổ đĩa (0: A, 1: B .v.v...)
CX: tổng số cung cần đọc/ghi
DX: số hiệu (địa chỉ logic) sector bắt đầu đọc
DS:BX: địa chỉ vùng đệm đọc/ghi số liệu

Nếu sử dụng DOS version 4.0 hoặc mới hơn thì các tham số cần truyền như sau:

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CX	= ffffH nếu thao tác trên partition >32M
DS:BX	địa chỉ của transfer packet (10 bytes)
DS:[BX+0]	số hiệu sector logic đầu tiên, là số 4 byte (DWORD)
DS:[BX+4]	số lượng sector đọc/ghi, là số 2 byte (WORD)
DS:[BX+6]	địa chỉ offset của vùng đệm dữ liệu, là số 2 byte (WORD)
DS:[BX+8]	địa chỉ segment của vùng đệm dữ liệu, là số 2 byte (WORD)

Kết quả trả lại:

- AX chứa mã lỗi nếu CF = 1 (CY). Nội dung tất cả các thanh ghi bị phá huỷ, trừ các thanh ghi DS, ES, SS, SP.
- Các dịch vụ này sau khi thực hiện xong còn để lại trên stack 1 word, người lập trình hệ thống thường cần phải loại nó đi.

- **Truy cập bằng tên File**

Có thể truy cập các file trên đĩa bằng các lệnh của hệ điều hành DOS. Cũng có thể truy cập file bằng các chương trình ngôn ngữ bậc cao và cả ngôn ngữ assembly. Các ngôn ngữ này cho phép gọi các chương trình dịch vụ file của DOS rất dễ ràng, người lập trình chỉ phải chỉ ra tên file và đường dẫn.

Truy cập file thông qua tên file là cách truy cập rất đơn giản và tiện lợi cho người sử dụng. DOS có INT 21h với đầy đủ các chức năng quản lý File.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

CÂU HỎI VÀ BÀI TẬP CHO CÁC CHƯƠNG

Chương 1

1. Máy tính điện tử là gì ? Các nguyên lý cơ bản để xây dựng máy tính điện tử ? So sánh ưu, nhược điểm của các máy tính được xây dựng dựa trên các nguyên lý này.
2. Trình bày các khái niệm về lệnh máy, ngôn ngữ máy và máy ảo.
3. Phân biệt sự khác nhau giữa biên dịch (compilation) và thông dịch (interpretation).
4. Trình bày các mức thường có trong một máy tính hiện đại, đặc điểm của các mức.
5. Phần cứng (hardware), phần mềm (software) và phần mềm được cứng hoá (firmware) giống nhau và khác nhau như thế nào ?.
6. Hãy điểm qua các cột mốc trong lịch sử phát triển máy tính điện tử; các thế hệ máy tính điện tử được phân biệt dựa vào những đặc điểm chủ yếu gì ?
7. Hãy nêu các đặc điểm chính để phân biệt các loại máy tính điện tử: microcomputer, minicomputer, superminicomputer, mainframe và supercomputer.

Chương 2

1. Trình bày cấu trúc chung của máy tính điện tử, các khối chức năng và quan hệ giữa các khối chức năng của máy tính điện tử.
2. Trình bày các khái niệm CPU, Bộ xử lý (processor) và Bộ vi xử lý (Microprocessor); sự khác nhau giữa các khái niệm này.
3. Các bước trong quá trình CPU thi hành một chỉ thị ?.
4. Trình thông dịch (interpreter) và phần cứng tương đương với nhau như thế nào? ý nghĩa của sự tương đương này?
5. Trình bày hoạt động của đường dữ liệu của CPU theo kiểu von Neumann cổ điển.
6. Phân biệt các loại máy tính SISD, SIMD và MIMD.
7. Trình bày các cơ chế song song trong máy tính loại SISD. Phân tích cơ chế hoạt động của máy tính song song kiểu đường ống.
8. Trình bày các khái niệm bit, byte, word, ô nhớ và địa chỉ ô nhớ.
9. Khoảng các Harming là gì? thế nào là mã sửa sai, mã phát hiện sai? bit chẵn/lẻ (parity bit). Cho một vài thí dụ.
10. Cho biết các phương pháp tổ chức I/O và so sánh chúng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Chương 3

1. Cổng là gì? Trình bày sơ đồ nguyên lý, ký hiệu, phương trình logic biểu diễn chức năng của các cổng logic cơ bản AND, OR, NOT.
2. Trình bày các cổng logic NAND, NOR, XOR. Hãy xây dựng bộ so sánh (Comparator) bằng các cổng XOR.
3. Hãy xây dựng các mạch logic AND, OR, NOT từ các cổng NAND hoặc NOR.
4. Phân tích sự hoạt động của Multiplexer, Demultiplexer, Decoder, PLA.
5. Phân tích sự hoạt động của Bộ dịch (Shifter), các bộ cộng Haft-Adder và Full-Adder.
6. Phân tích sự hoạt động của bộ Số học và Logic 1-bit (ALU).
7. Xây dựng một ALU nhiều bit từ các bộ ALU 1-bit.
8. Mô tả sự hoạt động của các thanh ghi chốt RS và thanh ghi chốt D.
9. Hãy xây dựng thanh ghi chốt RS từ các cổng NAND.
10. Thế nào là sự chuyển mạch của các mạch lưỡng bền ? Thế nào là sự chuyển mạch bằng sườn dương ?, sườn âm ?, bằng mức của xung kích thích ? Thanh ghi chốt (Latch) và Flip-Flop khác nhau ở đặc điểm gì ? Ký hiệu trên sơ đồ của chúng.
11. Phân tích hoạt động của phần tử nhớ RAM tĩnh (SRAM) 1 bit.
12. Thanh ghi là gì ? nêu một số chức năng của thanh ghi.
13. Hãy trình bày về các loại bộ nhớ ROM, nguyên tắc chế tạo và hoạt động của chúng?
14. Hãy trình bày phương pháp tổ chức bộ nhớ có độ dài từ và dung lượng theo yêu cầu, cho thí dụ (có thể sử dụng chip RAM 1024x1).
15. Bus là gì ? Nêu các bus thường có trong một máy tính.
16. Hãy trình bày các cách phân loại bus?
17. Phân tích hoạt động của bus đồng bộ.
18. Phân tích hoạt động của bus không đồng bộ.
19. Trọng tài bus là gì ?, trình bày nguyên lý hoạt động của trọng tài tập trung có một và hai mức ưu tiên.
20. Hãy trình bày phương pháp Memory-mapped I/O; Giải mã địa chỉ toàn phần (Full address decoding) và Giải mã địa chỉ một phần (Partial address decoding).

Chương 4

1. Hãy cho biết các tín hiệu cần thiết để điều khiển đường dữ liệu trên hình 4-07.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

2. Trình bày việc định thời vi chỉ thị, các sự kiện xảy ra trong từng chu kỳ con. (Dựa trên hình 4-09.)
3. Trình bày việc định trình tự các vi chỉ thị và vấn đề lệnh nhảy.
4. Ngôn ngữ vi assembly (MAL) được trình bày trong chương 4 có các đặc điểm gì so với các ngôn ngữ khác?
5. Hãy phân tích kỹ bảng 4-02 để thấy được sự tương đương của mỗi dòng chỉ thị MAL với một vi chỉ thị.
6. Phân tích tỉ mỉ vi chương trình trên hình 4-10 để hiểu rõ hoạt động của trình thông dịch này.
7. Hãy so sánh, bình luận việc thiết kế các chỉ thị theo chiều ngang và theo chiều đứng.
8. Trình bày về lập siêu vi chương trình (nanoprogramming).
9. Hãy tính thời gian thi hành chỉ thị trung bình trong một máy tính đường ống (average_instruction_time).
10. Thế nào là tính cục bộ của các chương trình ?, cho thí dụ. Trình bày ý tưởng cơ bản về bộ nhớ cache ?
11. Trình bày về cache liên kết (Associative cache).
12. Trình bày về cache ánh xạ trực tiếp (Direct-mapped cache).
13. So sánh cache liên kết và cache ánh xạ trực tiếp.
14. So sánh các chiến lược quản lý việc ghi vào bộ nhớ cache: copy-back và write-through.

Chương 5

1. Trình bày về khuôn dạng chỉ thị và các tiêu chuẩn thiết kế khuôn dạng chỉ thị.
2. Thế nào là mở rộng mã chỉ thị, cho thí dụ?
3. Trình bày về việc đánh địa chỉ (các mode địa chỉ).

Chương 6

1. Trình bày các chủ đề chính trong việc thiết kế mức máy hệ điều hành.
2. Trình bày khái niệm về bộ nhớ ảo, so sánh nó với bộ nhớ cache, nêu các điểm giống nhau và khác nhau.
3. Trình bày về việc phân trang (paging).
4. Trình bày về cách thực hiện phân trang.
5. So sánh 2 phương pháp: Cấp trang khi có yêu cầu và Mô hình tập làm việc.
6. Trình bày khái niệm chỉ thị vào/ra ảo.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Chương 7

1. Ngôn ngữ assembly là gì? so sánh nó với các ngôn ngữ lập trình khác như Pascal, C.
2. Trình bày khuôn dạng một chỉ thị ngôn ngữ assembly.
3. Nêu ý nghĩa của việc tinh chỉnh chương trình, cho thí dụ.
4. Quá trình assembly là gì? thế nào là một assembler two-pass.
5. Nhiệm vụ của quá trình kết nối - linking là gì ?.

Chương 8

1. Thế nào là thiết bị ngoại vi ? Nêu nhiệm vụ cơ bản của thiết bị ngoại vi và Đặc điểm chung của các thiết bị ngoại vi.
2. Trình bày các phương pháp tạo mã bàn phím.
3. Mã SCAN của bàn phím là gì ? ý nghĩa của việc sử dụng nó ?.
4. Cho biết cơ sở của phương pháp quét mành; tại sao phải làm tươi hình ảnh ?.
5. Trình bày Nguyên lý tổng hợp màu sắc và cấu tạo đèn hình màu; quan hệ giữa khả năng hiển thị màu và lượng bộ nhớ cần thiết dành cho mỗi điểm ảnh (pixel).
6. Thế nào là ma trận ký tự ? Cách lưu trữ, cách hiển thị thông tin kiểu ký tự trên màn hình máy tính?
7. Điều chế thông tin là gì ? Trình bày các phương pháp FM, MFM để mã hoá thông tin ghi lên đĩa ? cho thí dụ.
8. Mô tả tổ chức thông tin trên đĩa mềm.
9. Trình bày các đặc điểm chung của đĩa cứng, nguyên lý cấu tạo ổ đĩa cứng.
10. Trình bày việc định vị đầu từ trong ổ đĩa cứng, vấn đề tracking, thời gian tìm (seek time) bao gồm các thành phần nào ?.
11. Trình bày việc định vị đầu từ sát trên mặt đĩa và các sự cố đĩa cứng.
12. Trình bày các phương hướng cải tiến đối với đĩa cứng nhằm nâng cao tốc độ và dung lượng?
13. Trình bày các chuẩn bộ điều khiển ổ đĩa cứng (HDC) ST506, ESDI, SCSI, IDE.
14. Trình bày nguyên lý chế tạo đĩa quang (CD-ROM), cách tổ chức thông tin, các ưu và nhược điểm so với đĩa cứng và các biện pháp khắc phục.
15. Mô tả nguyên lý hoạt động của các loại máy in, so sánh.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Chương 9

1. Trình bày và so sánh các đặc điểm kỹ thuật chính của máy vi tính IBM-PC/XT và IBM-PC/AT.
2. Mô tả cách tổ chức bộ nhớ của IBM PC.
3. Thế nào là ngắt ?, Vector ngắt ?, bảng Vector ngắt ?, Chương trình con xử lý ngắt và ưu điểm của việc sử dụng ngắt ?
4. Trình bày quá trình khởi động IBM-PC ?
5. Nêu các thành phần cơ bản của hệ điều hành DOS của IBM-PC và vai trò của chúng.
6. Trình bày cách DOS tổ chức thông tin trên đĩa mềm ? bảng FAT và bảng thư mục đĩa. Hãy chỉ rõ mối liên hệ giữa bảng thư mục đĩa và bảng FAT.
7. Mô tả tổ chức thông tin trên đĩa cứng nói chung (Tổ chức vật lý) và cách DOS tổ chức thông tin trên đĩa cứng.
8. Nêu các mức truy cập đĩa, trình bày ngắt 13H, 25H, 26H; so sánh cách sử dụng chúng.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

TÀI LIỆU THAM KHẢO

1. Albert Paul Malvino, An Introduction to Micro Computer, Mc Graw-Hill, 1989.
2. Andrew S, Tanenbaum, Structured Computer Organization, Printice Hall, 1990.
3. James M. Feldman, CHARLES T. RETTER, Computer Architecture. MIT Press&Mc Graw-Hill, Singapore, 1994.
4. Michael Titcher, PCINTERN-System programming: The Encyclopedia of DOS programming Know How. Arbacus, 1992.
5. Peter Norton, Programmer's Guide to The IBM PC&PS/2, Microsoft Press (Người dịch: Nguyễn Ngọc Minh Hằng). NXB Thống Kê, Hà Nội, 1990.
6. Tech Help! 6.0 (Electronics Book), Flambeaux Software, Inc.
7. Nguyễn Chí Công, Kỹ thuật vi xử lý (sách dịch). NXB Thống Kê, Hà Nội, 1993.
8. Nguyễn Văn Tam, Các bộ vi xử lý thông dụng 16/32 bit. NXB Thống Kê, Hà Nội, 1990.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

MỤC LỤC

LỜI NÓI ĐẦU **3**

CHƯƠNG 1 MỞ ĐẦU **5**

1.1 Những khái niệm và nguyên lý cơ bản **5**
1.1.1 Máy tính 5
1.1.2 Nguyên lý xây dựng và sự phân loại máy tính điện tử 5
1.1.3 Ngôn ngữ máy, mức và máy ảo 8
1.1.4 Các máy tính nhiều mức hiện đại 11
1.1.5 Kiến trúc máy tính và đối tượng nghiên cứu của nó 15
1.1.6 Phần cứng, Phần mềm và các máy tính nhiều mức 15
1.2 Những cột mốc trong lĩnh vực kiến trúc máy tính **17**
1.2.1 Thế hệ số không - Máy tính cơ khí (1642-1945) 18
1.2.2 Thế hệ thứ nhất - Máy tính dùng đèn điện tử (1945-1955) 19
1.2.3 Thế hệ thứ hai - Máy tính dùng Transistor (1955-1965) 21
1.2.4 Thế hệ thứ ba - Máy tính dùng Mạch tích hợp (1965-1980) 22
1.2.5 Thế hệ thứ tư - Máy tính dùng mạch VLSI (1980 -) 24

CHƯƠNG 2 TỔ CHỨC HỆ THỐNG MÁY TÍNH **27**

2.1 Kiến trúc chung của máy tính điện tử **27**
2.2 Bộ xử lý - Processors **29**
2.2.1 Sự thi hành các chỉ thị 30
2.2.2 Tổ chức CPU 31
2.2.3 Sự thực hiện song song các lệnh 33
2.3 Bộ nhớ **38**
2.3.1 Bit 38
2.3.2 Địa chỉ ô nhớ, byte, word 38
2.3.3 Trật tự byte trong word, big endian và little endian computer 38
2.3.4 Mã sửa sai 40
2.3.5 Bộ nhớ phụ 40
2.4 Tổ chức Vào/Ra (I/O) **41**
2.4.1 Các phương pháp tổ chức I/O 41
2.4.2 Tổ chức I/O trong các máy tính lớn 42
2.4.3 Tổ chức I/O trong các máy tính PC 43

CHƯƠNG 3 MỨC LOGIC SỐ **45**

3.1 Các cổng và đại số logic **45**
3.1.1 Cổng (gate) 45
3.1.2 Đại số logic 47
3.1.3 Sự thực hiện các hàm logic 48
3.1.4 Sự tương đương của các mạch 49
3.2 Các mạch logic số cơ bản **52**
3.2.1 Mạch tích hợp - Integrated Circuits (IC) 52
3.2.2 Các mạch tổ hợp (Combinational Circuit) 52

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

3.2.3 Các mạch số học 55
3.3 Bộ nhớ 59
3.3.1 Phần tử nhớ 1 bit 60
3.3.2 Flip-flop và thanh ghi 64
3.3.3 Bộ nhớ ROM 66
3.3.4 Tổ chức bộ nhớ 70
3.4 Chip vi xử lý và các bus 78
3.4.1 Chip vi xử lý (MPU - Microprocessor Unit) 78
3.4.2 Các bus của máy tính 80
3.4.3 Bus đồng bộ (Synchronous bus) 82
3.4.4 Bus không đồng bộ (asynchronous bus) 85
3.4.5 Trọng tài bus (bus arbitration) 87
3.4.6 Việc xử lý ngắt 90
3.5 Thí dụ về một số BUS 91
3.5.1 Bus IBM PC 91
3.5.2 Bus IBM PC/AT 95
3.5.3 Bus VME 96
3.6 Giao diện - Interfacing 97
3.6.1 Các chip vào/ra (I/O chips) 97
3.6.2 Việc giải mã địa chỉ 99

CHƯƠNG 4 MỨC VI CHƯƠNG TRÌNH 102

4.1 Các vấn đề chính của mức logic số mà người lập vi chương trình quan tâm . 103
4.1.1 Thanh ghi 103
4.1.2 Bus 103
4.1.3 Bộ phân/dồn kênh và bộ giải mã 105
4.1.4 Đơn vị Số học-Logic và Bộ dịch 106
4.1.5 Đồng hồ 107
4.1.6 Bộ nhớ chính 107
4.2 Một vi kiến trúc lấy làm thí dụ 109
4.2.1 Đường dữ liệu (Data path) 109
4.2.2 Vi chỉ thị - microinstruction 110
4.2.3 Việc định thời vi chỉ thị 113
4.2.4 Sự định trình tự các vi chỉ thị 116
4.3 Thí dụ về một kiến trúc mức máy thông thường 117
4.3.1 Kiến trúc mức máy thông thường - Vĩ kiến trúc 117
4.3.2 Vấn đề đánh địa chỉ bộ nhớ chính và bộ nhớ ngăn xếp 117
4.3.3 Tập vĩ chỉ thị 119
4.4 Thí dụ về một vi chương trình 121
4.4.1 Ngôn ngữ Vi assembly 121
4.4.2 Thí dụ về một vi chương trình 123
4.4.3 Nhận xét về vi chương trình 127
4.5 Những vấn đề cần cân nhắc, lựa chọn khi thiết kế mức vi chương trình 129
4.5.1 Vi chỉ thị theo chiều đứng và theo chiều ngang 129
4.5.2 Lập siêu vi chương trình - Nanoprogramming 130
4.5.3 Đường ống dẫn chỉ thị và các chỉ thị nhảy 131
4.5.4 Bộ nhớ cache 134

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

4.6 Tóm tắt mức vi chương trình **141**

CHƯƠNG 5 MỨC MÁY THÔNG THƯỜNG **143**

5.1 Khuôn dạng của chỉ thị **143**
5.1.1 Các tiêu chuẩn thiết kế khuôn dạng chỉ thị 144
5.1.2 Việc mở rộng mã chỉ thị 145
5.1.3 Thí dụ về khuôn dạng chỉ thị của máy PDP-11 147
5.2 Việc đánh địa chỉ **148**
5.2.1 Đánh địa chỉ tức thời- Immediate Addressing 148
5.2.2 Đánh địa chỉ trực tiếp - Direct Addressing 148
5.2.3 Đánh địa chỉ thanh ghi - Register Addressing 149
5.2.4 Đánh địa chỉ gián tiếp- Indirect Addressing 149
5.2.5 Chỉ số - Indexing 149
5.2.6 Đánh địa chỉ stack 150
5.3 Các kiểu chỉ thị **151**
5.4 Tóm tắt mức máy thông thường **152**

CHƯƠNG 6 MỨC MÁY HỆ ĐIỀU HÀNH **153**

6.1 Giới thiệu mức máy hệ điều hành **153**
6.2 Bộ nhớ ảo **155**
6.2.1 Việc phân trang - Paging 156
6.2.2 Thực hiện việc phân trang 158
6.2.3 Phương pháp Cấp trang khi có yêu cầu và Mô hình tập làm việc 162
6.3 Các chỉ thị vào/ra ảo **165**
6.3.1 Các chỉ thị vào/ra ảo đối với các file tuần tự 165
6.3.2 Các chỉ thị vào/ra ảo đối với các file truy cập ngẫu nhiên 166
6.3.3 Việc cài đặt các chỉ thị vào/ra ảo 166
6.4 Tóm tắt mức máy hệ điều hành **166**

CHƯƠNG 7 GIỚI THIỆU MỨC NGÔN NGỮ ASSEMBLY **167**

7.1 Giới thiệu ngôn ngữ assembly **167**
7.1.1 Ngôn ngữ assembly là gì ? 167
7.1.2 Khuôn dạng chỉ thị ngôn ngữ assembly 168
7.1.3 So sánh ngôn ngữ assembly và các ngôn ngữ bậc cao 171
7.1.4 Tinh chỉnh chương trình - Program Tuning 172
7.2 Quá trình assembly **174**
7.2.1 Các assembler duyệt hai lần (Two-pass Assembler) 174
7.2.2 Lần duyệt thứ nhất - Pass One 175
7.2.3 Lần duyệt thứ hai - Pass Two 177
7.3 Kết nối - Linking **177**

CHƯƠNG 8 CÁC THIẾT BỊ NGOẠI VI **180**

8.1 Khái quát **180**
8.1.1 Khái niệm về thiết bị ngoại vi 180
8.1.2 Nhiệm vụ cơ bản và yêu cầu đối với thiết bị ngoại vi 180
8.1.3 Nguyên lý cấu tạo và đặc điểm của thiết bị ngoại vi 180

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

8.2 Bàn phím - Keyboard **182**
8.2.1 Giới thiệu 182
8.2.2 Chuyển mạch cơ khí 182
8.2.3 Các phương pháp tạo mã bàn phím 183
8.2.4 Mã Scan của bàn phím máy IBM PC hoặc tương thích 186
8.2.5 Liên lạc giữa bàn phím và CPU ở IBM PC/XT/AT 186
8.3 Màn hình (Video Display, Monitor) **187**
8.3.1 Nguyên lý của phương pháp hiển thị hình ảnh video 187
8.3.2 Phương pháp quét mành 188
8.3.3 Làm tươi hình ảnh bằng phương pháp DMA 192
8.3.4 Một số chuẩn của màn hình máy tính 194
8.3.5 So sánh một số đặc điểm của màn hình máy tính và TV 194
8.3.6 Hiển thị thông tin 196
8.4 Đĩa từ và đĩa quang **199**
8.4.1 Giới thiệu về các loại thiết bị nhớ phụ 199
8.4.2 Nguyên lý ghi từ, đầu từ 200
8.4.3 Các phương pháp mã hoá số liệu ghi lên đĩa 201
8.4.4 Đĩa mềm và ổ đĩa mềm 205
8.4.5 Đĩa cứng - HDD (Hard Disk Drive) 210
8.4.6 Đĩa quang (optical disc, Compact Disc) 220
8.5 Máy in - Printer **223**
8.5.1 Máy in búa 223
8.5.2 Máy in dòng 224
8.5.3 Máy in kim 224
8.5.4 Máy in laser 224

CHƯƠNG 9 MÁY VI TÍNH IBM PC **226**

9.1 Giới thiệu **226**
9.1.1 Sự ra đời của máy IBM PC 226
9.1.2 Họ 80xx của hãng Intel 226
9.1.3 Bộ vi xử lý 80286 233
9.1.4 Bộ vi xử lý 80386 233
9.1.5 Bộ vi xử lý 80486 234
9.1.6 Bộ vi xử lý Pentium 234
9.2 Máy IBM PC nguyên thuỷ **235**
9.2.1 BUS và các khe cắm mở rộng 235
9.2.2 Các chip hỗ trợ 235
9.2.3 Các linh kiện khác của máy IBM PC 238
9.2.4 Những chuyển mạch đặt hệ thống (system switch) 238
9.2.5 Bộ nhớ trong của IBM PC 239
9.2.6 Các cổng - Ports 242
9.2.7 Các ngắt (Interrupts) 243
9.3 Hệ máy IBM PC/XT, IBM PC/AT **244**
9.3.1 Các đặc tính kỹ thuật chính của IBM PC/XT 244
9.3.2 Các đặc tính kỹ thuật chính của IBM PC/AT 245
9.4 Hệ điều hành DOS của máy IBM PC **246**
9.4.1 ROM-BIOS 247

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

9.4.2 Record khởi động 250
9.4.3 IBMBIO.COM 252
9.4.4 IBMDOS.COM 253
9.4.5 COMMAND.COM 254
9.5 Truy cập IBM PC 255
9.5.1 Các thông tin cấu hình máy trong RAM CMOS 255
9.5.2 Tổ chức thông tin trên đĩa mềm 257
9.5.3 Tổ chức thông tin trên đĩa cứng 266
9.5.4 Truy cập thông tin trên đĩa 267

CÂU HỎI VÀ BÀI TẬP CHO CÁC CHƯƠNG 269

TÀI LIỆU THAM KHẢO 274

MỤC LỤC 275

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

16 Hàng Chuối - Hai Bà Trưng - Hà Nội
Điện thoại: (04) 9718312; (04) 9724770. Fax: (04) 9714899
E-mail: nxb@ vnu.edu.vn

Chịu trách nhiệm xuất bản:

Giám đốc: PHÙNG QUỐC BẢO

Tổng biên tập: PHẠM THÀNH HƯNG

Biên tập: NGUYỄN THUỶ
ĐÌNH VIỆT

Trình bày bìa: VĂN SÁNG

KIẾN TRÚC MÁY TÍNH

Mã số: 1L-107 ĐH2006
In 1000 cuốn, khổ 19 x 27 cm tại Trung tâm in tranh Tuyên truyền Cổ động
Số xuất bản: 85 - 2006/CXB/73 - 01/ĐHQGHN, ngày 24/01/2006
Quyết định xuất bản số: 271 LK/XB
In xong và nộp lưu chiểu quý III năm 2006.

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012

Kiến trúc máy tính

* 2 0 0 7 D V L 2 8 9 3 *

Giá: 32.000 đ

THƯ VIỆN ĐHDL HP

127.0.0.1 downloaded 112- Kientruc Maytinh.pdf at Mon Aug 06 08:59:01 ICT 2012