TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT
KHOA CÔNG NGHỆ THÔNG TIN

THS. PHAN NHƯ MINH
(Bộ môn truyền thông và mạng máy tính)

BÀI GIẢNG
KIẾN TRÚC MÁY TÍNH

DÙNG CHO SINH VIÊN KHOA CÔNG NGHỆ THÔNG TIN

LƯU HÀNH NỘI BỘ
Hà nội 2022

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

LỜI NÓI ĐẦU

Bài giảng Kiến trúc máy tính được biên soạn làm tài liệu giảng dạy của các giảng viên và là tài liệu tham khảo cho sinh viên chuyên ngành ***Công nghệ thông tin, Hệ thống thông tin, Mạng máy tính và truyền số liệu*** của Trường Đại học Công nghệ Giao thông vận tải theo yêu cầu, mục tiêu đào tạo của Nhà trường.

Bài giảng Kiến trúc máy tính cung cấp cho sinh viên các kiến thức cơ sở của kiến trúc máy tính; hệ thống phân cấp của bộ nhớ, bộ nhớ trong, bộ nhớ cache và các loại bộ nhớ ngoài và các thiết bị vào ra. Nội dung của bài giảng được biên soạn thành sáu chương:

Chương 1: Giới thiệu chung về kiến trúc máy tính

Chương 2: Biểu diễn thông tin trong máy tính

Chương 3: Mức logic số

Chương 4: Bộ xử lý trung tâm CPU

Chương 5: Hệ thống nhớ

Chương 6**:** Hệ thống vào ra

Bài giảng được biên soạn dựa trên kinh nghiệm giảng dạy môn học Kiến trúc máy tính tại Đại học Công nghệ GTVT. Tài liệu có thể được sử dụng làm tài liệu học tập, tham khảo cho sinh viên hệ đại học và cao đẳng ngành công nghệ thông tin.

Người biên soạn
Hà Nội, tháng 8 năm 2021

CÁC TỪ VIẾT TẮT

ALU	Arithmetic and Logic Unit
BCD	Binary Coded Decimal
CD	Compact Disk
CD-ROM	Compact Disk Read Only Memory
CD-RW	CD - Recordable
CPU	Central Processing Unit
DMA	Direct memory Access
DRAM	Dynamic RAM
DVD	Digital Video Disk
FR	Flag register
IEEE	Institute of Electrical and Electronic Engineers
LSB	Least Significant Bit
MSB	Most Significant Bit
PIC	Programmable Interrupt Controller
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
ROM	Read Only Memory
USB	Universal Serial Bus

Chương 1
GIỚI THIỆU CHUNG VỀ KIẾN TRÚC MÁY TÍNH

Trang bị cho sinh viên những kiến thức cơ bản về kiến trúc máy tính: Các khái niệm, các thành phần cơ bản, phần mềm của máy tính và lịch sử phát triển của máy tính.

1.1. CÁC KHÁI NIỆM VÀ NGUYÊN LÝ CƠ BẢN

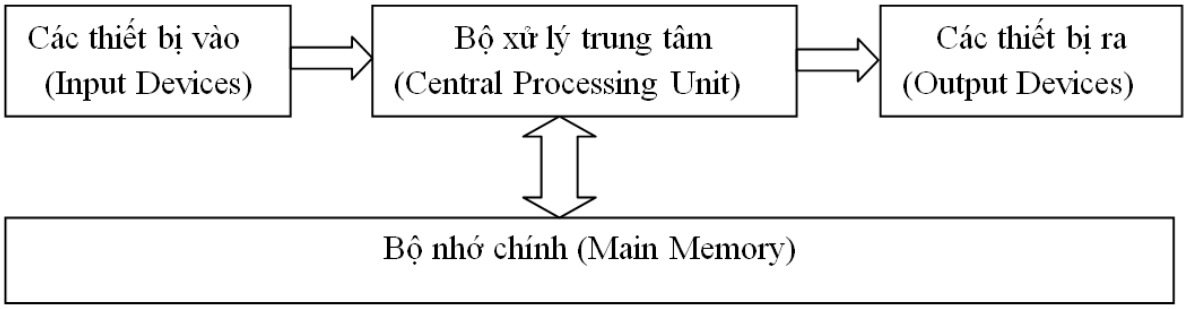
Kiến trúc máy tính là khoa học về việc lựa chọn và kết nối các thành phần phần cứng để tạo ra các máy tính đạt được các yêu cầu về chức năng (functionality), hiệu năng (performance) và giá thành (cost). Yêu cầu chức năng đòi hỏi máy tính phải có thêm nhiều tính năng phong phú và hữu ích; yêu cầu hiệu năng đòi hỏi máy tính phải đạt tốc độ xử lý cao hơn và yêu cầu giá thành đòi hỏi máy tính phải càng ngày càng rẻ hơn. Để đạt được cả ba yêu cầu về chức năng, hiệu năng và giá thành là rất khó khăn. Tuy nhiên, nhờ có sự phát triển rất mạnh mẽ của công nghệ vi xử lý, các máy tính ngày nay có tính năng phong phú, nhanh hơn và rẻ hơn so với máy tính các thế hệ trước.

1.1.1. Khái niệm máy tính

Máy tính (computer) là một thiết bị điện tử hoạt động dưới sự điều khiển của các chỉ thị được lưu trữ trong bộ nhớ. Nó thực hiện các công việc sau:

- Nhận thông tin đầu vào
- Xử lý thông tin theo chương trình được lưu trong bộ nhớ
- Đưa thông tin ra

Máy tính hoạt động theo chương trình.



Hình 1-1. Mô hình máy tính cơ bản

1.1.2. Kiến trúc máy tính và cấu trúc máy tính

- Kiến trúc máy tính (architecture) nghiên cứu những thuộc tính của một hệ thống mà người lập trình có thể nhìn thấy được, những thuộc tính quyết định trực tiếp đến việc thực thi một chương trình tính toán, xử lý dữ liệu

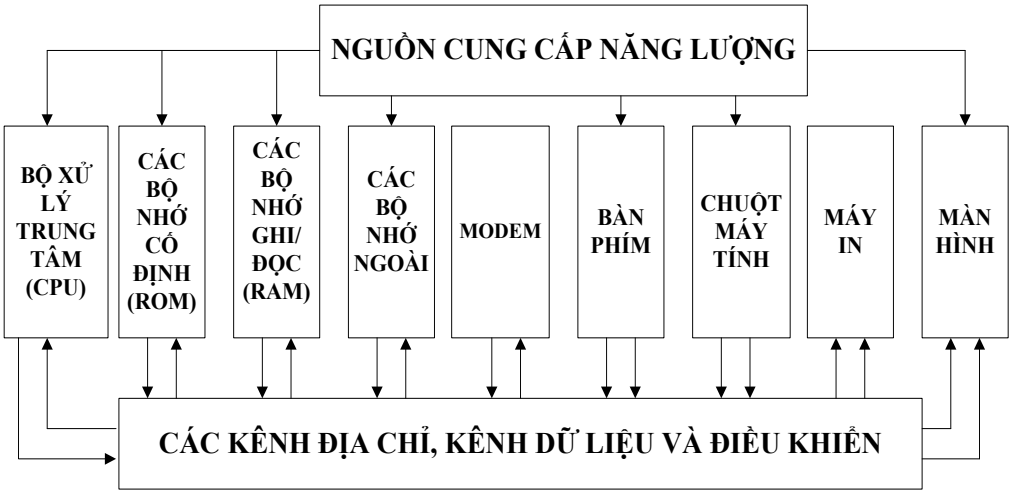
- Cấu trúc máy tính (structure) nghiên cứu về các thành phần chức năng và sự kết nối giữa chúng để tạo nên một máy tính, nhằm thực hiện chức năng và tính toán kỹ thuật của kiến trúc.

Những thuộc tính liên quan đến kiến trúc bao gồm tập lệnh cơ bản mà CPU có thể thực hiện, số bit được sử dụng để biểu diễn các loại dữ liệu khác nhau, cơ chế nhập/xuất dữ liệu và các kỹ thuật đánh địa chỉ ô nhớ,…Cấu trúc máy tính lại bao gồm các thuộc tính kỹ thuật mà người lập trình không nhận biết được như các tín hiệu điều khiển, giao diện giữa máy tính và thiết bị ngoại vi, công nghệ xây dựng bộ nhớ…

Ví dụ việc quyết định máy tính có cần một lệnh cơ bản để thực hiện phép nhân hay không là vấn đề về kiến trúc. Còn thể hiện lệnh nhân bằng các đơn vị vật lý cụ thể nào (chẳng hạn, một đơn vị thuộc phần cứng đặc biệt, hay thực hiện lặp nhiều phép cộng) lại là vấn đề về cấu trúc. Để làm ví dụ minh họa sự khác biệt đó ta có thể xem các máy tính ở Trung tâm nghiên cứu nào đó. Các máy tính này có thể có kiến trúc rất giống nhau theo quan điểm của người lập trình. Chúng có cùng số thanh ghi (tức là thiết bị lưu trữ tạm thời), có cùng một tập lệnh cơ bản và dạng các toán hạng được nạp vào bộ nhớ giống nhau. Tuy nhiên các hệ thống này khác nhau về mặt cấu trúc: số bộ vi xử lý khác nhau, kích thước bộ nhớ của chúng cũng khác hẳn nhau, cách thức dữ liệu được truyền từ bộ nhớ đến bộ vi xử lý cũng không giống nhau. Kiến trúc máy tính thường được ứng dụng trong khoảng thời gian dài, hàng chục năm; trong khi cấu trúc thường thay đổi cùng với sự phát triển của công nghệ. Trên cùng một kiến trúc, các hãng chế tạo máy tính có thể đưa ra nhiều loại máy tính khác nhau về cấu trúc, do đó các đặc trưng về hiệu suất, giá thành cũng khác nhau. Các sản phẩm của IBM là một ví dụ điển hình. Kiến trúc máy tính của IBM vẫn còn được ứng dụng cho tới ngày nay và là ngọn cờ của thương hiệu IBM. Trong lĩnh vực máy PC, người ta thường không phân biệt rõ ràng giữa kiến trúc và cấu trúc vì sự khác biệt giữa hai khái niệm này đã rút ngắn đáng kể. Sự phát triển của công nghệ không chỉ tác động lên cấu trúc mà còn tạo điều kiện phát triển các kiến trúc mạnh hơn và nhiều tính năng hơn; và do đó tác động qua lại giữa kiến trúc và cấu trúc thường xuyên hơn [3,4].

1.2. CÁC THÀNH PHẦN CƠ BẢN CỦA MÁY TÍNH

Để đảm bảo tính tương thích, cấu trúc phần cứng bên trong các máy vi tính cá nhân về cơ bản là giống nhau. Vì thế chúng ta có cấu trúc chung của máy vi tính như sau:



Hình 1-2. Cấu trúc chung của máy vi tính

1.2.1. Bộ nguồn

1.2.1.1. Nguồn cấp điện cho máy lớn

Bộ nguồn có chức năng chuyển điện xoay chiều AC 110 – 220V thành điện một chiều DC để cung cấp cho các mạch điện tử bên trong máy, cũng như các bộ phận ngoại vi.

Nguồn điện là điều kiện cơ bản cho các máy tính hoạt động nên một bộ nguồn hoạt động ổn định và cung cấp đủ công suất rất quan trọng đối với một máy vi tính cá nhân. Tùy theo chủng loại cấu hình, mỗi máy vi tính cá nhân cần một công suất khác nhau.

Ví dụ: Laptop công suất 60 – 80W; để bàn: 200W- 400W.

Để máy tính làm việc tốt thì bộ nguồn cần phải ổn định, làm nguội tốt, hiệu suất cao và phải có khả năng mở rộng. Bộ nguồn cung cấp cho bản mạch chính và các thiết bị ngoại vi những điện thế: ± 5V; ± 12V; ± 3,3V; 0V. Ngày nay, có nhiều bộ nguồn với công nghệ mới nhất và chất lượng tốt nhất điển hình như: Corsair, Seasonic, Superflower....

Bộ nguồn được chia theo nguyên tắc hoạt động thành 2 loại:

Bộ nguồn tuyến tính: Gồm một biến thế để hạ điện áp, một mạch nắn dòng (Dùng 4 Diode công suất) và một hoặc nhiều bộ ổn định hiệu điện thế (Có thể đổi 12V – 5V). Do bộ nguồn tuyến tính giải phóng rất nhiều nhiệt lượng và hao tốn điện năng nên ngày nay gần như không tồn tại trong vi tính cá nhân.

Bộ ổn áp ngắt: Là bộ nguồn rất nhẹ và hiệu suất cao. Năng lượng điện được điều tiết theo nguyên tắc đóng – mở.

1.2.1.2. Nguồn pin cho máy tính xách tay

Thế hệ đầu tiên bộ nguồn dành cho máy tính xách tay là pin NiCad (Nickel Cadmium). Thế hệ sau của nó là pin NiMH (Nickel Metal Hybride).

Năm 1998, trên thị trường xuất hiện thêm pin Li – Ion (Lithium Ion). Pin Li – Ion có thời gian làm việc lâu hơn pin NiMH (Khoảng 3 tiếng làm việc liên tục), nhẹ hơn và không cần xả hết trước khi nạp. Nhược điểm duy nhất của loại pin này là sẽ tự xả hết điện nếu như không được sử dụng trong thời gian dài.

Chính vì lý do trên nên tới năm 1999, đã xuất hiện loại pin mới Li – polymer. Pin này có mật độ điện tích cao hơn nhiều Li- Ion. Thay vì dùng điện môi lỏng, pin dùng điện môi dạng gôm hay rắn ghép giữa các điện cực. Pin có cấu trúc lớp được sử dụng rộng rãi từ năm 2000 cho tới ngày nay [6].

1.2.2. Bản mạch chính

Bản mạch chính (Main Board) chứa đựng những linh kiện điện tử và những chi tiết quan trọng nhất của máy vi tính cá nhân như: Bộ vi xử lý CPU (Central Processing Unit), hệ thống bus và các vi mạch hỗ trợ. Vì vậy, bản mạch chính cần phải: nhỏ gọn, ổn định với nhiễu bên ngoài và an toàn về điện.

1.2.2.1. Bộ xử lý trung tâm (CPU- Central Processing Unit)

Trung tâm đầu não của máy vi tính là bộ xử lý trung tâm, nó có nhiệm vụ quản lý điều hành và phân phối các tài nguyên của hệ thống tới các thiết bị làm việc khác trong hệ thống máy vi tính. Trong suốt quá trình làm việc của máy vi tính, nó thông qua các kênh điều khiển, kênh địa chỉ và kênh dữ liệu, tiến hành tất cả các phép gia công và xử

lý thông tin, các tín hiệu điều khiển đều được thực hiện và điều phối trong bộ xử lý trung tâm.

Trong quá trình làm việc của bộ xử lý trung tâm, nó căn cứ vào lệnh nhận được, sau đó sẽ phát ra tín hiệu điều khiển đưa đến các thiết bị khác để yêu cầu chúng hoạt động, chỉ cho chúng biết phải lấy tài nguyên ở bộ phận nào, địa chỉ nào và chỉ ra hướng truyền thông tin cho chúng. Khả năng, tốc độ của bộ xử lý trung tâm quyết định khả năng và tốc độ của máy vi tính.

1.2.2.2. Bộ nhớ cố định (ROM- Read Only Memory)

ROM chứa thông tin cố định chỉ được phép đọc ra từ nó, đó là thông tin chương trình khởi động máy, chương trình chạy thử máy, chương trình biên dịch ngôn ngữ. Nội dung của ROM sẽ không thay đổi khi bị mất điện. Thông tin ghi nhớ ở trong bộ nhớ ROM thường được nạp sẵn từ khi sản xuất. Ngoài ra để thuận tiện cho người sử dụng, thiết kế, xây dựng hệ thống chuyên dùng người ta còn sử dụng các loại EPROM, PROM (các loại bộ nhớ kiểu này có thể ghi lại bằng thiết bị đặc biệt).

1.2.2.3. Bộ nhớ ghi/đọc (RAM- Random Access Memory)

Bộ nhớ ghi/đọc (RAM) có thể truy xuất dữ liệu một cách ngẫu nhiên, chúng ta có thể ghi vào RAM, hoặc đọc dữ liệu ra từ RAM. Bất kỳ thời điểm nào bộ xử lý trung tâm có yêu cầu trao đổi thông tin thì bộ nhớ ghi/đọc (RAM) đều phải đáp ứng. Chúng thường được sử dụng để ghi nhớ tạm thời dữ liệu trong suốt quá trình hoạt động gia công, xử lý thông tin của máy tính.

Các bộ nhớ ghi/đọc thường có hai loại đó là RAM động (DRAM) và RAM tĩnh (SRAM).

RAM tĩnh được cấu tạo từ các vi mạch nhớ, xây dựng trên các TRIGƠ (flip-flop). Nếu chúng ta liên tục duy trì nguồn cung cấp thì nội dung trong RAM tĩnh được bảo toàn, thao tác trao đổi thông tin với RAM tĩnh đơn giản hơn, tốc độ truy cập nhanh hơn, nhưng dung lượng so với RAM động nhỏ hơn khi chúng cùng thể tích.

Nguyên tắc lưu trữ số liệu ở bộ nhớ RAM động giống như nguyên tắc lưu trữ năng lượng (điện áp) của tụ điện. Mỗi bit nhớ trong RAM động tương ứng với một tụ điện. Như vậy theo thời gian thì năng lượng lưu trữ ở trong RAM động sẽ bị suy giảm, dẫn đến hiện tượng mất dữ liệu trong quá trình làm việc, để đảm bảo duy trì dữ liệu trong bộ nhớ hệ thống máy tính phải liên tục thực hiện thao tác làm tươi bộ nhớ.

Quá trình làm tươi bộ nhớ RAM động thường thực hiện theo chu kỳ từ 2 đến 3 giây, tức là cứ 2 hoặc 3 giây tại các vị trí dữ liệu của bộ nhớ DRAM có mức lôgic 1 máy tính phải tiến hành nạp số liệu lại 1 lần.

1.2.2.4. Các bộ nhớ ngoài

- Ngoài các hệ thống lưu trữ dữ liệu bên trong như ROM, RAM, các hệ máy vi tính còn sử dụng các bộ nhớ ngoài để lưu trữ thông tin với mục đích tăng dung lượng nhớ cho hệ thống, lưu trữ thông tin để bảo quản lâu dài, đồng thời để sử dụng các thông tin đó chuyển sang các hệ máy vi tính khác, cũng như để cập nhật các chương trình điều hành, các chương trình ứng dụng,... Các bộ nhớ ngoài thường được sử dụng hiện nay đó là các hệ thống đĩa cứng, đĩa mềm, đĩa quang, USB,…

- Về nguyên tắc thì hệ máy vi tính quản lý các hệ thống nhớ ngoài thông qua các cổng vào/ra giống như các thiết bị ngoại vi, chúng cũng được địa chỉ hoá và quản lý

chặt chẽ. Trong quá trình làm việc với chúng hệ máy vi tính sẽ tiến hành kiểm tra để báo cho người sử dụng biết chúng có được phối ghép với hệ thống làm việc hay không.

1.2.3. Các thiết bị ngoại vi

1.2.3.1. Bàn phím (Keyboard)

Là thiết bị đưa thông tin vào đơn giản và thông dụng nhất của máy vi tính, chúng là tổ hợp của các công tắc tức thời. Tín hiệu lối ra của các công tắc đưa tới bộ giải mã bàn phím, tại đó chúng được tạo ra các bộ mã tương ứng với chức năng của từng phím, chúng chính là dãy các con số mã nhị phân của các ký tự, ký hiệu, các chữ số thập phân, v,..v,.. Các bộ mã này sẽ được đưa vào hệ máy vi tính để yêu cầu nó thực hiện các chức năng theo ý muốn của người sử dụng.

1.2.3.2. Màn hình (Monitor)

Màn hình chính là thiết bị đưa thông tin ra. Màn hình dùng để hiển thị các thông tin mang nội dung của các số liệu, cũng như các văn bản đưa vào, hay là các thông báo của máy vi tính đối với người sử dụng, trên cơ sở đó người sử dụng đánh giá nó có hoạt động đúng hay không và thực hiện các thao tác tiếp tục theo yêu cầu để máy tính tiếp tục hoạt động, hoặc thực hiện dừng hệ thống khi công việc thực hiện xong.

Ngày nay thường sử dụng hai loại màn hình cơ bản.

+ Hiển thị bằng ống tia điện tử CRT,

+ Tinh thể lỏng (hoặc ma trận điôt màu).

Sử dụng màn hình tinh thể lỏng thì hệ thống gọn, nhẹ, tiết kiệm năng lượng và an toàn hơn đối với người sử dụng.

1.2.3.3. Máy in (Printer)

Được sử dụng trong hệ máy tính để in ra các văn bản, các biểu đồ, các số liệu trên giấy để sử dụng trong các mục đích khác nhau. Chúng ta có thể sử dụng các máy in đen trắng, máy in màu tuỳ theo yêu cầu và chất lượng hình ảnh cần in.

Có nhiều loại máy in khác nhau, nhưng người ta thường phân thành hai loại:

- Máy in tiếp xúc: Là loại tạo nên các ký tự, hình ảnh bằng cách tiếp xúc cơ học của các kim in lên các băng mực đặt trên mặt giấy in. Loại máy này tốc độ chậm, thường chỉ in được đen trắng, độ phân giải thấp, gây ra tiếng ồn trong quá trình làm việc. Ngày nay loại máy in này ít được sử dụng.

- Máy in không tiếp xúc: Loại này thực hiện in ra các ký tự không cần va đập, do đó chúng không gây tiếng ồn khi làm việc, chất lượng in tốt, độ phân giải cao, tốc độ làm việc cao và có thể in các hình ảnh màu. Thông dụng nhất của loại in không tiếp xúc là máy in laze, máy in phun.

1.2.3.4. Modem và các thiết bị ngoại vi khác

Ngoài các thiết bị đã nêu trên người ta còn sử dụng thêm các thiết bị khác. Ví dụ như: chuột máy tính, modem phối ghép.

Chuột máy tính thường được kết nối thông qua cổng COM, USB giúp cho việc điều hành máy vi tính được thuận tiện, nhanh chóng hơn. Chuột máy tính có hai loại: Chuột không dây và chuột có dây, đối với chuột có dây lại có chuột cơ và chuột quang. Còn chuột không dây thì thường là chuột quang. Chuột máy tính là thiết bị chuột có thể nối trực tiếp hoặc nối không dây thông qua cổng thông tin hồng ngoại.

Modem phối ghép: Đó là các mạch điện tử dùng để phối ghép các thiết bị ngoại vi khác nhau như: các thiết bị xử lý âm thanh, xử lý hình ảnh, các thiết bị ghép nối mạng máy tính với mạng internet, mạng thông tin,...

1.3. PHẦN MỀM MÁY TÍNH

Phần mềm máy tính (Chương trình phần mềm): Là tổ hợp các lệnh trong kiến trúc tập lệnh. Đó là toàn bộ các chương trình điều hành máy tính và các chương trình ứng dụng thực hiện các thuật toán trong quá trình gia công và xử lý thông tin.

Các chương trình điều hành và các chương trình ứng dụng có thể được nhập từ bàn phím, đĩa cứng, đĩa mềm hoặc các thiết bị khác. Chương trình được lưu trữ ở dạng tập hợp các bít nhị phân 0 và 1, chương trình được ghi nhớ dưới dạng mã máy. Phần mềm như là linh hồn của máy tính.

Phần mềm máy tính thì có phần mềm hệ thống và phần mềm ứng dụng.

Phần mềm hệ thống (thường được gọi là hệ điều hành): Bao gồm các hệ thống chương trình phần mềm, thực hiện chức năng điều hành hệ thống, nó giúp cho máy tính quản lý, vận hành hệ thống, kể cả việc khởi động và sửa chữa các chương trình.

Chương trình ứng dụng: Được xây dựng đáp ứng cho người sử dụng khai thác và vận hành máy tính theo những mục đích khác nhau.

Ngoài việc tạo ra những phần mềm có ích con người còn tạo ra các phần mềm có hại cho máy tính. Những phần mềm như vậy chúng ta gọi là Virus.

1.4. LỊCH SỬ PHÁT TRIỂN MÁY TÍNH

Nguồn gốc sơ khai của việc tính toán được sử dụng bằng công cụ là bàn tính gẩy tay, nó được phát minh khoảng 3000 năm trước Công nguyên tại thành phố Babilon, sau này hàng loạt các công cụ tính toán khác nhau ra đời. Nhưng phải đến thế kỷ 17 thì máy tính cơ học đầu tiên có thể thực hiện được các phép tính cộng, trừ mới được chế tạo thành công do phát minh của nhà bác học Pascal người Pháp [1,4].

Năm 1930 tại Mỹ máy tính điện tử số được bác học Bush đề cập xây dựng, đến năm 1944 ở Mỹ hãng IBM do giáo sư Aiken chỉ đạo bắt đầu sản xuất máy tính hoàn thiện.

Kể từ đó máy tính phát triển mạnh mẽ và được chia thành các thế hệ như sau:

+ Máy tính thế hệ 1 (1950 – 1959): Là thế hệ dùng bóng điện tử chân không. Năm 1946 máy tính điện tử số ENIAC được xây dựng có chứa khoảng 18.000 bóng đèn điện tử, 1500 rơle, cùng với hàng triệu các linh kiện thụ động khác. Máy tính này nặng khoảng 30 tấn, chiếm diện tích 200m2, trên một toà nhà với không gian rộng lớn với các hệ thống làm mát tiêu thụ khá nhiều năng lượng, hiệu suất và khả năng làm việc rất thấp.

+ Máy tính thế hệ 2 (1959 – 1963): Thế hệ chế tạo các linh biện bán dẫn rời (Diode, Transistor). Máy tính đầu tiên của thế hệ này là TX – 0 (Transistorized Experimental Computer 0).

+ Máy tính thế hệ 3 (1964-1974): Máy tính được xây dựng trên các vi mạch cỡ nhỏ (SSI) và cỡ vừa (MSI), điển hình là thế hệ máy System360 của IBM. Thế hệ máy

tính này có những bước đột phá mới đó là: Tính tương thích cao, đặc tính đa chương trình, không gian địa chỉ lớn.

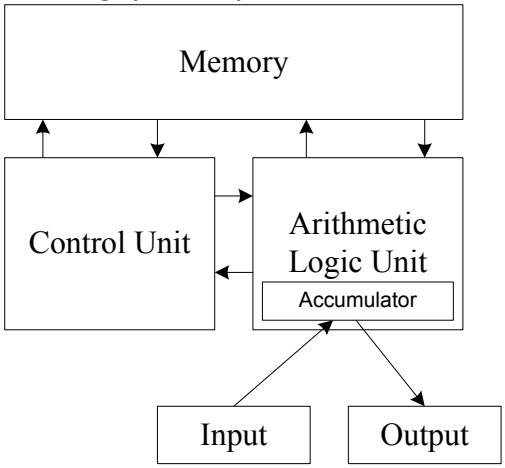
+ Máy tính thế hệ 4 (1974 – 2000): Máy tính được xây dựng trên các vi mạch cỡ lớn (LSI) và cực lớn (VLSI).

+ Máy tính thế hệ 5 (2000 đến nay): Ứng dụng máy tính thông minh của trí tuệ nhân tạo với công nghệ Nơron.

Ngày nay với sự phát triển mạnh mẽ của công nghệ vi điện tử và công nghệ bán dẫn thì cấu trúc của máy tính ngày càng được thu gọn, tốc độ làm việc và khả năng thao tác để giải quyết các bài toán lớn càng được tăng lên. Kể từ nửa sau của thế kỷ 20 máy tính điện tử đã phát triển qua nhiều thời kỳ và nhiều giai đoạn khác nhau với tốc độ cực kỳ nhanh chóng, cấu hình ngày càng hoàn thiện và được nâng cấp liên tục, kết cấu ngày càng được thu gọn và đặc biệt giá thành ngày càng giảm mạnh. Đây chính là thế mạnh của máy vi tính làm cho nó được ứng dụng rộng rãi trong nhiều lĩnh vực đời sống, kinh tế, xã hội,...

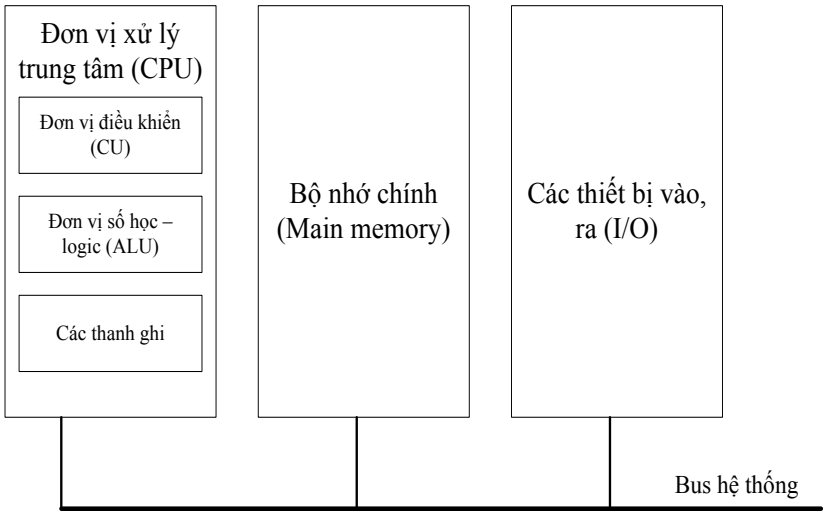
1.5. KIẾN TRÚC MÁY TÍNH VON-NEUMANN

Kiến trúc máy tính von-Neumann được John von-Neumann đưa ra vào năm 1945. Kiến trúc máy tính von-Neumann nguyên thuỷ như Hình 1.3.



Hình 1-3. Kiến trúc máy tính von-Neumann nguyên thủy

Các máy tính hiện đại ngày nay sử dụng kiến trúc máy tính von-Neumann cải tiến còn được gọi là kiến trúc máy tính von-Neumann hiện đại (Hình 1.4).



Hình 1-4. Kiến trúc máy tính von-Neumann hiện đại

Kiến trúc von-Neumann dựa trên các nguyên lý sau:

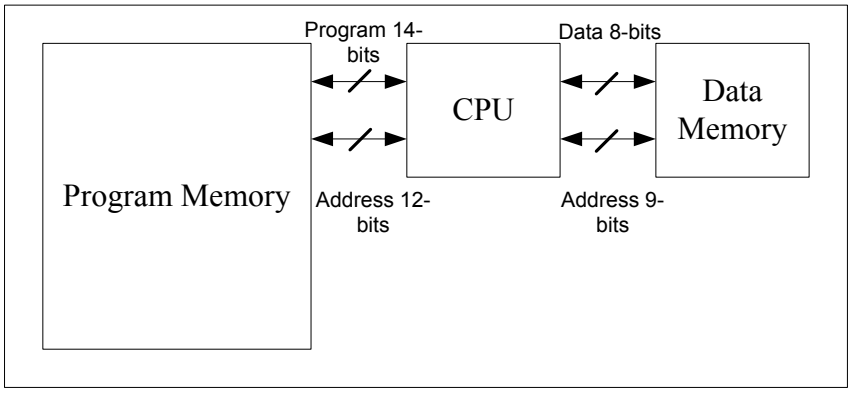
- Lệnh và dữ liệu được lưu trữ trong bộ nhớ đọc ghi chia sẻ - một bộ nhớ duy nhất được sử dụng để lưu trữ cả lệnh và dữ liệu,
- Bộ nhớ được đánh địa chỉ theo vùng, không phụ thuộc vào nội dung nó lưu trữ và.
- Các lệnh của một chương trình được thực hiện tuần tự.

Quá trình thực hiện lệnh được chia thành 3 giai đoạn (stages) chính:

- CPU đọc (fetch) lệnh từ bộ nhớ.
- CPU giải mã và thực hiện lệnh; nếu lệnh yêu cầu dữ liệu, CPU đọc dữ liệu từ bộ nhớ.
- CPU ghi kết quả thực hiện lệnh vào bộ nhớ (nếu có).

1.6. KIẾN TRÚC MÁY TÍNH HAVARD

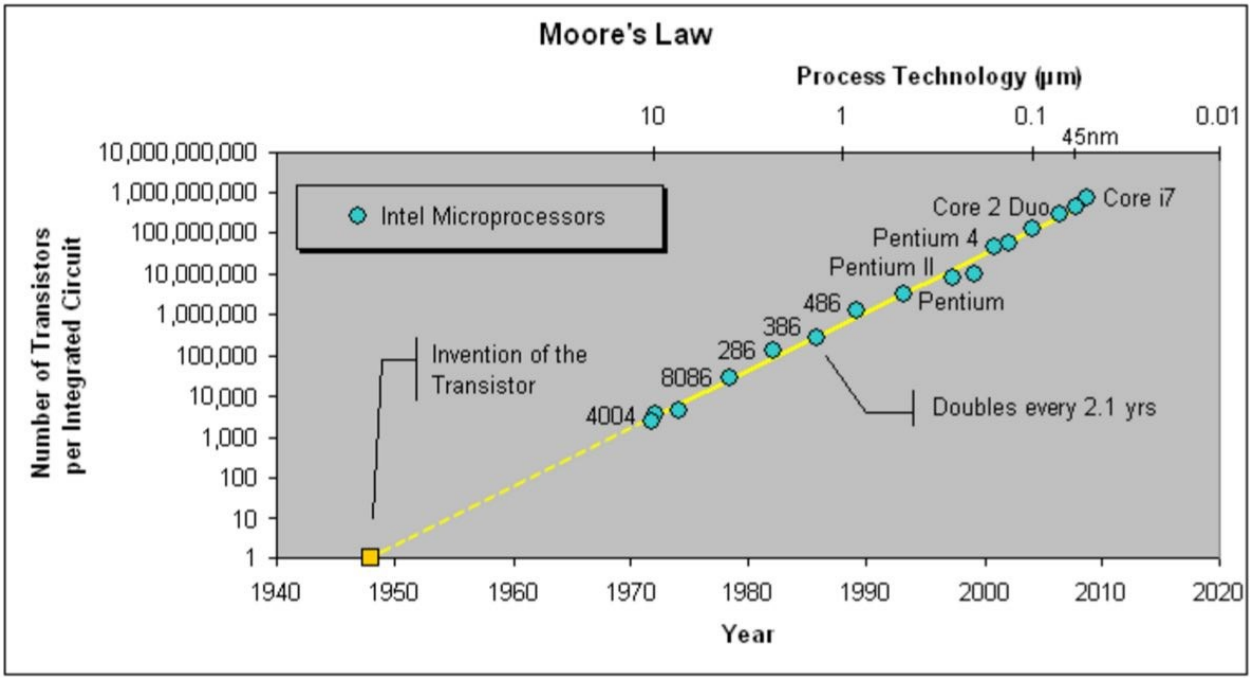
Kiến trúc máy tính Harvard là một kiến trúc như Hình 1.5. Kiến trúc máy tính Harvard có bộ nhớ trong gồm hai thành phần: Bộ nhớ lưu chương trình (Program Memory) và Bộ nhớ lưu dữ liệu (Data Memory). Hai hệ thống bus riêng được sử dụng để kết nối CPU với bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu. Mỗi hệ thống bus đều có đầy đủ ba thành phần để truyền dẫn các tín hiệu địa chỉ, dữ liệu và điều khiển. Máy tính dựa trên kiến trúc Harvard có khả năng đạt được tốc độ xử lý cao hơn máy tính dựa trên kiến trúc von-Neumann do kiến trúc Harvard hỗ trợ hai hệ thống bus độc lập với băng thông lớn hơn. Ngoài ra, nhờ có hai hệ thống bus độc lập, hệ thống nhớ trong kiến trúc Harvard hỗ trợ nhiều lệnh truy nhập bộ nhớ tại một thời điểm, giúp giảm xung đột truy nhập bộ nhớ, đặc biệt khi CPU sử dụng kỹ thuật đường ống (pipeline).



Hình 1-5. Kiến trúc máy tính Havard

1.7. ĐỊNH LUẬT MOORE

Định luật Moore được xây dựng bởi Gordon Moore - một trong những sáng lập viên của tập đoàn sản xuất chip máy tính nổi tiếng Intel. Từ năm 1965, G. Moore đã đưa ra dự đoán: Khả năng của máy tính sẽ tăng lên gấp đôi sau 18 tháng với giá thành là như nhau.



Hình 1-6. Sự phát triển của bộ xử lý Intel theo qui luật Moore

Một số kết luận rút ra từ quy luật Moore:

- Chi phí cho máy tính sẽ giảm.
- Giảm kích thước các linh kiện, máy tính sẽ giảm kích thước.
- Hệ thống kết nối bên trong mạch ngắn: tăng độ tin cậy, tăng tốc độ.
- Tiết kiệm năng lượng cung cấp, toả nhiệt thấp.
- Các IC thay thế cho các linh kiện rời.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 1

Câu 1. Nêu các thành phần cơ bản của máy tính?
Câu 2. Phần mềm của máy tính là gì?
Câu 3. Cấu trúc và kiến trúc máy tính giống và khác nhau như thế nào?
Câu 4. Dựa vào tiêu chuẩn nào người ta phân chia máy tính thành các thế hệ?
Câu 5. Đặc trưng cơ bản của các thế hệ máy tính?
Câu 6. Phần cứng (Hardware) là gì? Phần mềm (Software) là gì? Phần sụn (Firmware) là gì?
Câu 7. Tại sao nói phần cứng và phần mềm là tương đương về mặt logic? Lấy ví dụ minh hoạ?
Câu 8. Khuynh hướng phát triển của các thế hệ máy tính ngày nay là gì?

Chương 2
BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH

Trang bị cho sinh viên kiến thức về cách biến đổi cơ bản của hệ thống số, các bảng mã thông dụng được dùng để biểu diễn các ký tự. Cách chuyển đổi giữa các hệ thống sử dụng trong máy tính.

Sinh viên nắm vững kiến thức về các khái niệm cơ bản liên quan đến các hệ thống số được dùng trong máy tính. Thành thạo các thao tác biến đổi số giữa các hệ thống số.

Các máy tính xử lý thông tin số và chữ. Các thông tin được biểu diễn dưới dạng mã nhất định. Bản chất vật lý của việc biểu diễn thông tin là điện áp (”0” ứng với không có điện áp, ”1” ứng với điện áp ở mức quy chuẩn trong mạch điện tử) và việc mã hóa các thông tin số và chữ được tuân theo chuẩn quốc tế. Mã hiệu để mã hóa các thông tin cho máy tính xử lý là các giá trị của biến nhị phân 0,1. Một biến chỉ nhận được một trong hai giá trị duy nhất là 0 hoặc 1 được gọi là một bít. Hai trạng thái này của bít là các giá trị tương ứng với trạng thái ’tắt” hặc ”đóng” của một công tắc điện, được sử dụng để mã hóa cho tất cả các ký tự (số, chữ, các ký tự đặc biệt).

Các bít được ghép lại thành các đơn vị mang thông tin đầy đủ. Với 8 bít ghép với nhau tạo thành đơn vị dư liệu cơ sở của máy tính gọi là 1 Byte. Do được lưu giữ tương đương với một ký tự nên Byte là đơn vị cơ sở để đo khả năng lưu giữ, xử lý của các máy tính.

2.1. HỆ ĐẾM

2.1.1. Hệ thập phân

Hệ thập phân (hay hệ đếm cơ số 10) là một hệ đếm có 10 ký tự dùng chỉ số lượng.

Trong hệ thập phân, 10 ký tự (còn gọi là con số) khác nhau được dùng để biểu đạt 10 giá trị riêng biệt (0, 1, 2, 3, 4, 5, 6, 7, 8 và 9), tức là 10 con số. Những con số này còn được dùng cùng với dấu thập phân - ví dụ dấu "phẩy" - để định vị phần thập phân sau hàng đơn vị. Con số còn có thể được dẫn đầu bằng các ký hiệu "+" hay "-" để biểu đạt số dương và số âm.

Hệ thập phân là một hệ đếm dùng vị trí định lượng, bao gồm hàng đơn vị, hàng chục, hàng trăm v.v. Vị trí của một con số ám chỉ một phép nhân (mũ 10) với con số ở vị trí đó, và mỗi con số về bên tay trái, có giá trị gấp mười lần con số kế bên, ở bên tay phải [7].

Công thức chuyển đổi từ hệ thống số đếm khác sang hệ thập phân:

$$N = \underbrace{a_{n-1}\dots a_0}_{n} = a_0.s^0 + a_1.s^1 + \dots + a_{n-1}.s^{n-1} = \sum_{i=0}^{n-1} a_i s^i \qquad (2.1)$$

Trong đó N là một số nguyên có n chữ số. Chữ số a_i tại vị trí i (i=0...n-1) được gọi là trị số (hay còn gọi là trọng số). Giá trị s là cơ số của hệ đếm. Hệ đếm được đặt tên theo giá trị cơ số s. Chẳng hạn, với s=2 ta có hệ đếm 2, với s=10 ta có hệ đếm 10, với s=16 ta có hệ đếm 16. Giá trị s cũng xác định số ký tự cần dùng để biểu diễn trị số. Chẳng hạn với s=2 hệ đếm này sẽ cần hai ký tự để biểu diễn, vì thế có khái niệm hệ nhị

phân. Tương tự, có hệ đếm 10 và hệ đếm 16 còn được gọi là hệ thập phân và thập lục phân.

***Ví dụ 2.1.1**:* Số nguyên $(1537)_{10}$. Giá trị số nguyên này được tính theo công thức (2.1) như sau:

$$N=1537D= 7x10^0 + 3x10^1 + 5x10^2 + 1x10^3$$

2.1.2. Hệ nhị phân (Binary)

2.1.2.1. Khái niệm

Hệ nhị phân hay hệ đếm cơ số 2 chỉ có hai con số 0 và 1. Đó là hệ đếm dựa theo vị trí. Giá trị của một số bất kỳ nào đó tuỳ thuộc vào vị trí của nó. Các vị trí có trọng số bằng bậc luỹ thừa của cơ số 2. Chấm cơ số được gọi là chấm nhị phân trong hệ đếm cơ số 2. Mỗi một con số nhị phân được gọi là một bit (Binary digit). Bit ngoài cùng bên trái là bit có trọng số lớn nhất (MSB, Most Significant Bit) và bit ngoài cùng bên phải là bit có trọng số nhỏ nhất (LSB, Least Significant Bit)

	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	
MSB	1	0	1	0	.	1	1	LSB

Chấm nhị phân

Dùng n bit có thể biểu diễn 2^n giá trị

Cho một số nhị phân:

$A= a_n\ a_{n-1} \dots a_1\ a_0\ ,\ a_{-1} \dots a_{-m}$

Giá trị của A được tính như sau:

$A= a_n\ 2^n + a_{n-1}\ 2^{n-1} + \dots + a_1\ 2^1 + a_0\ 2^0 + a_{-1}\ 2^{-1} + \dots + a_{-m}\ 2^{-m}$

2.1.2.2. Biến đổi từ nhị phân sang thập phân

Các số nhị phân có thể được biến đổi thành thập phân bằng cách tính tổng của các con số, nhân với giá trị vị trí của nó.

$$N= \underbrace{a_{n-1} \dots a_0}_{n} = a_0.s^0 + a_1.s^1 + \dots + a_{n-1}.s^{n-1} = \sum_{i=0}^{n-1} a_i s^i \qquad (2.2)$$

***Ví dụ 2.1.2**:* Biến đổi số nhị phân $(11001)_2$ thành số thập phân:

Trọng số vị trí:	2^4	2^3	2^2	2^1	2^0
Giá trị vị trí:	16	8	4	2	1
Số nhị phân:	1	1	0	0	1

Số thập phân: $1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0 = (25)_{10}$

2.1.2.3. Biến đổi thập phân sang nhị phân

Có hai phương pháp:

Phương pháp 1: Chia dần cho 2 rồi lấy phần dư.

Phương pháp 2: Phân tích thành tổng của các số 2^i (Nhanh hơn với các số thập phân có giá trị nhỏ).

Phương pháp 1:

Với phần nguyên, ta thực hiện phương pháp chia lặp, tức là chia liên tiếp số thập phân cho cơ số của hệ đếm (cơ số 2) cho đến khi thương số bằng 0 và thực hiện lấy số dư theo thứ tự số dư cuối cùng là chữ số có trọng số lớn nhất và số dư đầu tiên là chữ số có trọng số nhỏ nhất.

Đối với phần lẻ của các số thập phân, số lẻ được nhân với cơ số (cơ số 2) và số nhớ được ghi lại làm một số nhị phân. Trong quá trình biến đổi, số nhớ đầu chính là **bit** MSB và số nhớ cuối là **bit** LSB.

Ví dụ 2.1.3*:* Biến đổi số thập phân $(29.625)_{10}$ thành nhị phân:

Biến đổi số $(29)_{10}$ sang nhị phân:

29/2 = 14 dư 1(LSB)
14/2 = 7 dư 0
7/2 = 3 dư 1
3/2 = 1 dư 1
1/2 = 0 dư 1(MSB)

=> $(29)_{10} = (11101)_2$.

Biến đổi số thập phân $(0.625)_{10}$ thành nhị phân:

0.625x2 = 1.250. Số nhớ là 1, là bit MSB.
0.250x2 = 0.500. Số nhớ là 0
0.500x2 = 1.000. Số nhớ là 1, là bit LSB.
=> $(0.625)_{10} = (0.101)_2$.
Vậy : $(29.625)_{10} = (11101.101)_2$

Ví dụ 2.1.4 *:* Biến đổi số thập phân $(24.25)_{10}$ sang hệ nhị phân:

Biến đổi $(24)_{10}$ sang hệ nhị phân:

24/2 = 12 dư 0
12/2 = 6 dư 0
6/2 = 3 dư 0
3/2 = 1 dư 1
1/2 = 0 dư 1

Vậy : $(24)_{10} = (11000)_2$

Biến đổi $(0.25)_{10}$ sang hệ nhị phân:

0.25x2 = 0.5 số nhớ là 0
0.5x2 = 1.0 số nhớ là 1
Vậy : $(0.25)_{10} = (0.01)_2$

Vậy : $(24.25)_{10} = (11000.01)_2$

Phương pháp 2:

Ta chỉ việc phân tích số thập phân thành tổng của các số 2^i.

Ví dụ 2.1.5 *:* Chuyển đổi $105_{(10)}$ sang hệ nhị phân :

$105 = 64 + 32 + 8 + 1 = 2^6 + 2^5 + 2^3 + 2^0$

Kết quả : $105_{(10)} = 01101001_{(2)}$

Ví dụ 2.1.6 *:* Chuyển đổi 17000_{10} sang hệ nhị phân :

$17000_{10} = 16384 + 512 + 64 + 32 + 8$
$= 2^{14} + 2^9 + 2^6 + 2^5 + 2^3$

Kết quả : $17000_{10} = 0100001001101000_2$

2.1.3. Hệ thập lục phân (Hexadecima)

2.1.3.1. Khái niệm

Các hệ máy tính hiện đại thường dùng một hệ đếm khác là hệ thập lục phân. Hệ thập lục phân là hệ đếm dựa vào vị trí với cơ số là 16. Hệ này dùng các con số từ 0 đến 9 và các ký tự từ A đến F như trong bảng sau:

Bảng 2-1. Hệ thập lục phân

Thập lục phân	Thập phân	Nhị phân
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

2.1.3.2. Biến đổi thập lục phân sang thập phân

Các số thập lục phân có thể được biến đổi thành thập phân bằng cách tính tổng của các con số, nhân với giá trị vị trí của nó.

$$N = \underbrace{a_{n-1}\ldots a_0}_{n} = a_0.s^0 + a_1.s^1 + \ldots + a_{n-1}.s^{n-1} = \sum_{i=0}^{n-1} a_i s^i \qquad (2.3)$$

Ví dụ 2.1.7*:* Biến đổi các số $(2AF)_{16}$ thành thập phân.

Số thập lục phân: 2 A F
Trọng số vị trí: 16^2 16^1 16^0
Giá trị vị trí: 256 16 1
Số thập phân: $2x256 + Ax16 + Fx1 = (687)_{10}$.

2.1.3.3. Biến đổi thập phân sang thập lục phân

Để biến đổi các số thập phân thành thập lục phân, ta sử dụng phương pháp chia lặp, với cơ số 16.

Ví dụ 2.1.8 *:* Biến đổi $(1776)_{10}$ thành thập lục phân.

1776/16 = 111 dư 0 (LSB).
111/16 = 6 dư 15 hoặc F.
6/16 = 0 dư 6 (MSB).
Số thập lục phân: $(6F0)_{16}$.

Ví dụ 2.1.9*:*

Biến đổi $(24A)_{16}$ thành thập phân.
Biến đổi $(586)_{10}$ thành thập lục phân.
Biến đổi $(24A)_{16}$ sang hệ thập phân:
- Số thập lục phân: 2 4 A
Trọng số vị trí: 16^2 16^1 16^0

Giá trị vị trí: 256 16 1

Số thập phân: $2x256 + 4x16 + 10x1 = 586_{10}$

Vậy: $(24A)_{16} = (586)_{10}$

- Biến đổi $(586)_{10}$ sang hệ thập lục phân:

586/16 = 36 dư 10 hay A

36/16 = 2 dư 4

2/16 = 0 dư 2

Vậy: $(586)_{10} = (24A)_{16}$

2.1.3.4. Biến đổi thập lục phân sang nhị phân

Để đổi các số thập lục phân thành nhị phân, chỉ cần thay thế một cách đơn giản từng con số thập lục phân bằng bốn bit nhị phân tương đương của nó.

Ví dụ 2.1.10: *Đổi số thập lục (DF6)16 thành nhị phân:*

D F 6

↓ ↓ ↓

1101 1111 0110

$(DF6)_{16} = (110111110110)_2$.

2.1.3.5. Biến đổi nhị phân sang thập lục phân

Để biến đổi một số nhị phân thành số thập lục phân tương đương thì chỉ cần gộp lại thành từng nhóm gồm 4 bit nhị phân, bắt đầu từ dấu chấm nhị phân.

Ví dụ 2.1.11*:* Biến đổi số nhị phân $(1111101000010000)_2$ thành thập lục phân.

1111 1010 0001 0000

↓ ↓ ↓ ↓

F A 1 0

Số thập lục phân: $(FA10)_{16}$.

Ví dụ 2.1.12*:* Chuyển số $(ABC)_{16}$ sang hệ nhị phân.

Chuyển $(1010\ 1011\ 1100)_2$ sang hệ thập lục phân.

Chuyển số $(ABC)_{16}$ sang hệ nhị phân:

A B C

↓ ↓ ↓

1010 1011 1100

$(ABC)_{16} = (1010\ 1011\ 1100)_2$

Chuyển số $(1010\ 1011\ 1100)_2$ sang hệ thập lục phân:

1010 1011 1100

↓ ↓ ↓

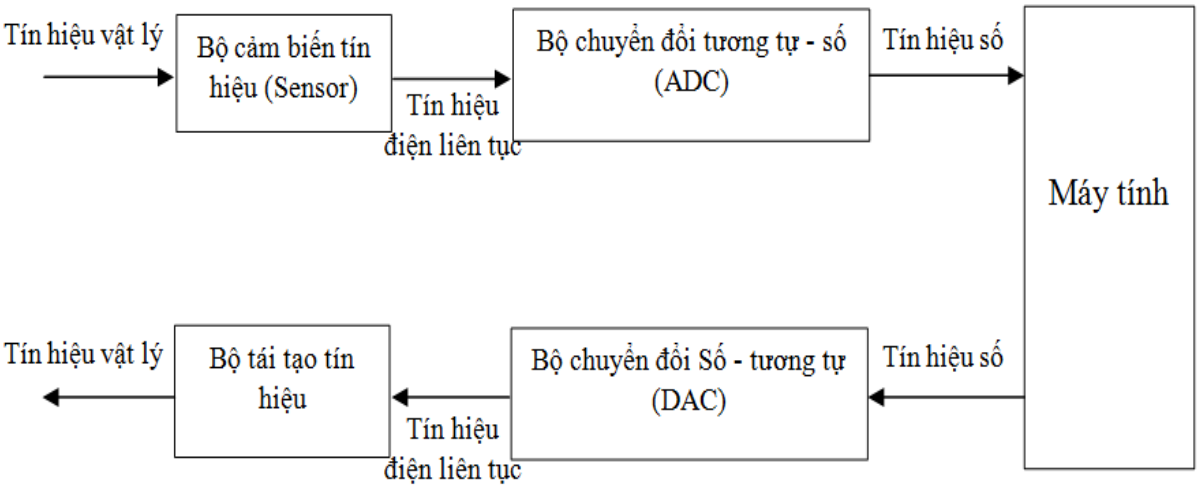
A B C

$(1010\ 1011\ 1100)_2 = (ABC)_{16}$

2.2. BIỂU DIỄN DỮ LIỆU SỐ TRONG MÁY TÍNH

2.2.1. Nguyên tắc chung về mã hóa dữ liệu

- Mọi dữ liệu đưa vào máy tính đều phải mã hóa thành số nhị phân.
- Có hai loại dữ liệu: Dữ liệu nhân tạo và dữ liệu tự nhiên
- Mã hóa dữ liệu nhân tạo:
  + Dữ liệu số nguyên: Mã hóa theo tiêu chuẩn quy ước
  + Dữ liệu số thực: Mã hóa bằng số dấu phẩy động.
  + Dữ liệu ký tự: Mã hóa theo bộ mã ký tự.
- Mã hóa và tái tạo dữ liệu vật lý (Tín hiệu tự nhiên):



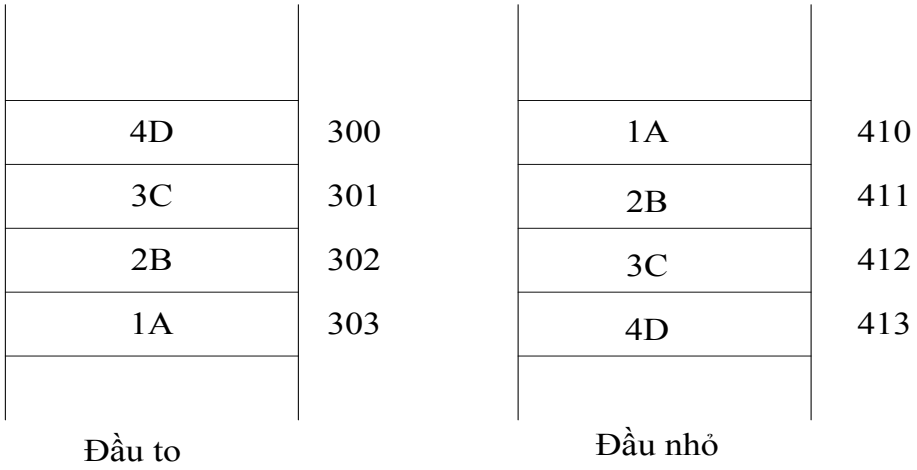
Hình 2-1. Sơ đồ khối mã hóa và tái tạo dữ liệu vật lý

Các dữ liệu vật lý thông dụng: Âm thanh và hình ảnh.

Độ dài dữ liệu: Độ dài từ dữ liệu là số bit được sử dụng để mã hóa loại dữ liệu tương ứng. Độ dài từ dữ liệu thường là bội của 8 – bit: 8, 16, 32, 64 bit

2.2.2. Thứ tự lưu trữ các byte của dữ liệu

- Bộ nhớ chính thường được tổ chức theo byte.
- Độ dài từ dữ liệu có thể chiếm từ 1 đến nhiều byte. Đối với dữ liệu nhiều byte ta cần phải biết thứ tự lưu trữ các byte trong bộ nhớ chính.
- Có hai cách lưu trữ:
  + Đầu nhỏ (Little - endian): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ nhỏ hơn.
  + Đầu to (Big - endian): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ lớn hơn.



Hình 2-2. Lưu trữ các byte của dữ liệu

- Các cách lưu trữ của các bộ xử lý điển hình:
- Intel 80x86 và các loại Pentium lưu trữ kiểu đầu nhỏ
- Motorola 680x0 và các bộ xử lý RISC lưu trữ kiểu đầu to
- Power PC và Itanium lưu trữ cả 2 kiểu

Ví dụ 2.2.1*:* Lưu trữ dữ liệu 32-bit

0001	1010	0010	1011	0011	1100	0100	1101
1	A	2	B	3	C	4	D

2.2.3. Biểu diễn số nguyên

2.2.3.1. Biểu diễn số nguyên không dấu

Tất cả các số cũng như các mã trong máy vi tính đều được biểu diễn bằng các chữ số nhị phân. Để biểu diễn các số nguyên không dấu, người ta dùng n bit.

$A = a_{n-1}\ a_{n-2} \ldots a_1\ a_0$ (a = 1; 0)

Giá trị của A được tính như sau: (2.4)

$A = a_{n-1}\ 2^{n-1} + a_{n-2}\ 2^{n-2} + \ldots + a_1\ 2^1 + a_0\ 2^0$

Với n = 8 bit biểu diễn được các giá trị từ 0…255

Giá trị 255 được biểu diễn là 1111 1111.

```
 1111 1111
+0000 0001
-----------
10000 0000
```

Vậy: 255+1 = 0

Trục số học máy tính: Biểu diễn với n = 8 bit

Theo trục số học thì với n = 8 thì biểu diễn được 256 giá trị từ 0…255. Tới giá trị 255 thì trục số lại quay lại giá trị 0.

Tương ứng với độ dài của số bit được sử dụng, ta có các khoảng giá trị xác định như sau:

Số bit	Khoảng giá trị
n bit:	0.. 2^n - 1
8 bit	0.. 255 Byte
16 bit	0.. 65535 Word

***Ví dụ 2.2.2**:* Biểu diễn các số nguyên không dấu sau bằng thanh ghi 8 bit

$A = (41)_{10}$; $B = (150)_{10}$

Hướng dẫn:

$A = (41)_{10} = 32 + 8 + 1 = 2^5 + 2^3 + 2^0$

$41_{10} = 0010\ 1001$

$B = (150)_{10} = 128 + 16 + 4 + 2 = 2^7 + 2^4 + 2^2 + 2^1$

$150_{10} = 1001\ 0110$

***Ví dụ 2.2.3**:* Cho các số nguyên không dấu M, N được biểu diễn bằng thanh ghi 8 bit như sau:

M= 0001 0010

N= 1011 1001

Xác định giá trị của chúng.

Hướng dẫn:

$M = 0001\ 0010 = 2^4 + 2^1 = 16 + 2 = 18_{10}$

$N = 1011\ 1001 = 2^7 + 2^5 + 2^4 + 2^3 + 2^0$

$= 128 + 32 + 16 + 8 + 1 = 185$

2.2.3.2. Biểu diễn số nguyên có dấu

Số bù một và số bù hai:

Gọi A là một số nhị phân.

Số bù 1 của A nhận được bằng cách đảo giá trị các bit của A

Số bù 2 của A = Số bù 1 của A + 1

***Ví dụ 2.2.4**:* Giả sử A = 0010 0101

Số bù 1 của A = 1101 1010

\+ 1

Số bù 2 của A = 1101 1011

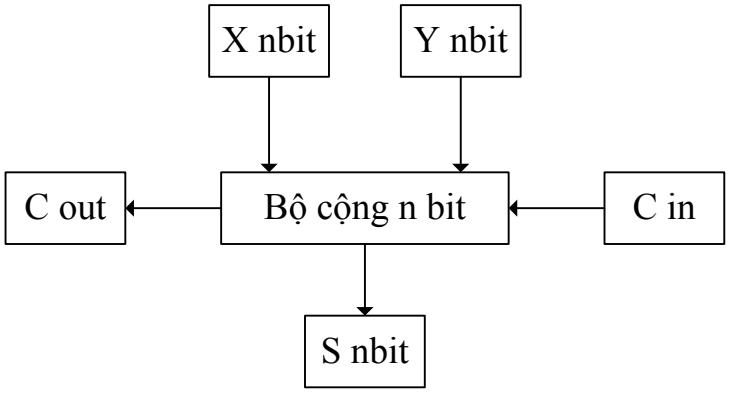
Ta thấy: A + (Số bù 2 của A) = 0

Nên số bù 2 biểu diễn được số âm. Chính vì thế mà số bù 2 dùng để biểu diễn số nguyên có dấu.

Người ta sử dụng bit cao nhất biểu diễn dấu; bit dấu có giá trị 0 tương ứng với số nguyên dương, bit dấu có giá trị 1 biểu diễn số âm. Như vậy khoảng giá trị số được biểu diễn sẽ được tính như sau:

Số bit	Khoảng giá trị:	
n bit	$-2^{n-1}.....2^{n-1}-1$	
8 bit	-128 ÷ 127	Short integer
16 bit	-32768 ÷ 32767	Integer
32 bit	-2147483648 ÷ 2147483647	Long integer

2.2.4. Các phép toán số học với số nguyên



Hình 2-3. Sơ đồ khối phép toán số học với số nguyên

2.2.4.1. Nguyên tắc thực hiện phép toán với số nguyên

Bit có trọng số lớn nhất là bit dấu:

Nếu bit dấu là:

00: Số dương

01: Tràn dương

10: Tràn âm

11: Số âm

2.2.4.2. Phép cộng số nguyên không dấu

Nếu không có nhớ ra khỏi bit MSB thì kết quả nhận được luôn luôn đúng:

$C_{out} = 0$

Nếu có nhớ ra khỏi bit MSB thì kết quả nhận được là sai:

$C_{out} = 1$ (Hiện tượng tràn số)

Hiện tượng tràn số xảy ra khi tổng lớn hơn $2^n - 1$

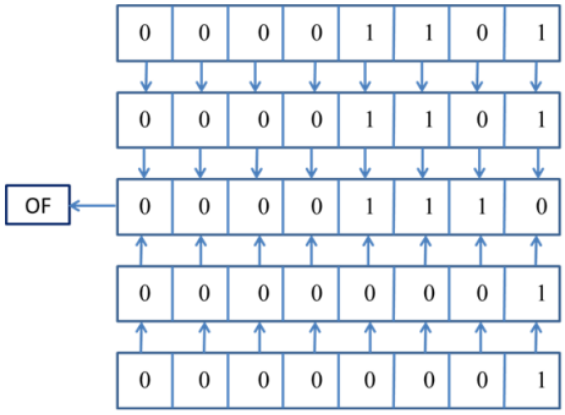
Nếu xuất hiện, hiện tượng tràn số thì ta tăng số lượng các bit lên (tăng n). Ví dụ từ 8 lên 16 bit

Ví dụ 2.2.5*:* Cho C = 1101; D = 0001

S = C + D

Hướng dẫn:

Tính S:



Hình 2-4. Thực hiện phép cộng nhị phân

Kết quả trong thanh ghi là: 00001110. Ta thấy bit dấu là 00 nên kết quả thu được sẽ là số dương 001110 có giá trị là 3

Vậy S = +3

Phép cộng số nguyên có dấu:

Biểu diễn số nguyên có dấu dưới dạng mã bù 2 của số không dấu trong thanh ghi n bit

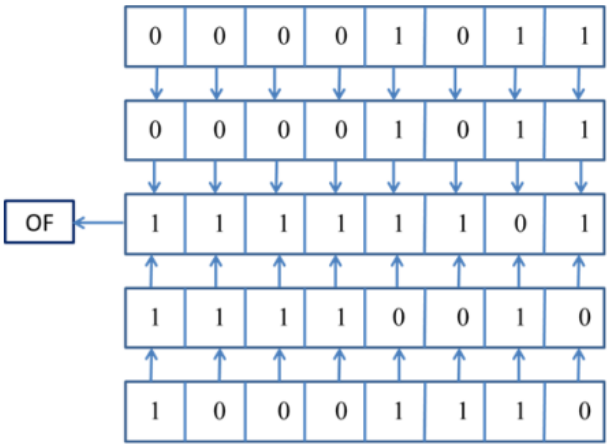
Thực hiện phép cộng hai số có dấu.

Chúng ta phải chú ý hiện tượng tràn số. Nếu xuất hiện tràn số thì ta tăng số lượng các bit lên (tăng n). Ví dụ từ 8 lên 16.

Ví dụ 2.2.6*:* Cho A = 1011; B = -1110. Tính S= A + B

Hướng dẫn:

Tính S:



Hình 2-5. Thực hiện phép cộng nhị phân

Kết quả trong thanh ghi là:

11111101. Ta thấy bit dấu là 11 nên kết quả nhận được là một số âm.

111101 có số bù 2 là: 000011.

Vậy kết quả của S= -3

2.2.5. Biểu diễn số thực

Có hai cách biểu diễn số thực trong một hệ nhị phân: Số có dấu chấm cố định (fixed point number) và số có dấu chấm động (floating point number). Cách thứ nhất được dùng trong những bộ vi xử lý (micro processor) và những bộ vi điều khiển (micro controller) cũ. Cách thứ hai được sử dụng hiện nay có độ chính xác cao hơn. Đối với cách biểu diễn số thực dấu chấm động có khả năng hiệu chỉnh theo giá trị của số thực.

2.2.5.1. Biểu diễn số thực dấu phẩy tĩnh

Quy tắc: ta chuyển đổi từng phần nguyên và lẻ theo quy tắc sau:

Phần nguyên: Chia liên tiếp phần nguyên cho 2 giữ lại các số dư, Số nhị phân chuyển đổi sẽ là dãy số dư liên tiếp tính từ lần chia cuối về lần chia đầu tiên.

Phần lẻ: Nhân liên tiếp phần lẻ cho 2, giữ lại các phần nguyên tạo thành. Phần lẻ của số Nhị phân sẽ là dãy liên tiếp phần nguyên sinh ra sau mỗi phép nhân tính từ lần nhân đầu đến lần nhân cuối.

Ví dụ 2.2.7*:* Chuyển sang hệ Nhị phân số: 13,625

Hướng dẫn:

Phần nguyên:

13:2 = 6 dư 1

6:2 = 3 dư 0

3:2 = 1 dư 1

1:2 = 0 dư 1

=> Phần nguyên của số Nhị phân là 1101

Phần lẻ:

0,625 x 2 = 1,250 Phần nguyên là 1

0,250 x 2 = 0,500 Phần nguyên là 0

0,500 x 2 = 1,000 Phần nguyên là 1 (dừng ở đây vì phân nguyên còn lại =0)

=> Phần lẻ của số Nhị phân là: 0,101

=> Ta viết kết quả là: $(13,625)_{10} = (1101,101)_2$

Chú ý: việc chuyển đổi từ hệ thập phân sang hệ Nhị phân không phải luôn được gọn gàng chính xác, trong trường hợp phép tính chuyển đổi kéo dài, thì tùy theo yêu cầu về độ chính xác mà ta có thể dùng phép tính ở mức độ cần thiết thích hợp.

❖ Nhận xét :

Các số dấu phẩy tĩnh chỉ dùng trong các bài toán yêu cầu độ chính xác không cao và luông cố định, ví dụ điểm môn học trong trường ĐH CN tính chính xác đến 1 số sau dấu phẩy (VD : 4,5). Như vậy các điểm sẽ luôn có độ chính xác trong khoảng đó. Trong thực tế có các bài toán yêu cầu độ chính xác gần như tuyệt đối như các bài toán trong ngân hàng, trong điều khiển máy bay, vệ tinh, ... Các bài toán này không thể quy định trước một độ chính xác, người ta phải sử dụng sô dấu chấm động, khi đó số các số sau dấu phẩy là không hạn chế về mặt biểu diễn.

VD : 12e-101=$12*10^{-101}$

Vấn đề là biểu diễn các số dấu chấm động như thế trên máy tính với cơ số 2

2.2.5.2. Biểu diễn số thực dấu phảy động

Nguyên tắc chung

Một số thực bất kỳ được biểu diễn bằng công thức sau:

$$\mathbf{x = M.R^E}$$

Trong đó : M - Phần định trị.

R - Cơ số.

E - Phần mũ.

Với một hệ thống nào đó thì R là cố định. Trong máy tính ngày nay thông thường R=2. Để biểu diễn được với cơ số2 người chuyển về dạng tương đương như sau:

$$\mathbf{X=(-1)^s\ 1.M^{2E-B}}$$

Trong đó: s: là bit dấu (s=0 phần định trị là dương; s=1 phần định trị là âm)

M : là phần định trị.

E: là số mũ được dịch chuyển đi B đơn vị.

R đã được biết (R=2) máy tính lưu số dấu chấm động bao gồm hai thành phần chính

Chuẩn **IEEE 754-1985** phân định 3 dạng số dấu chấm động cơ bản (IEEE: Institute of Electrical and Electronics Engineers)

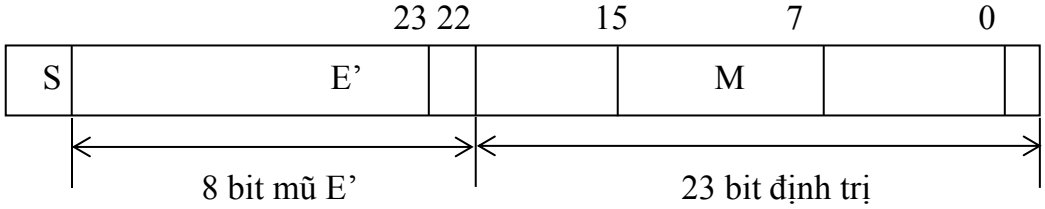
- Số có độ chính xác đơn dài 32 bit (single)
- Số có độ chính xác kép dài 64 bit (double)
- Số có độ chính xác mở rộng dài 128bit (quadruple)

Bảng 2-2. Hệ thập lục phân

Loại	**Single**	**Double**	**Quadruple**
Bề rộng của trường (bit)			
S	**1**	**1**	**1**
E	**8**	**11**	**15**
M	**23**	**52**	**111**
Tổng cộng	**32**	**64**	**128**
E cực đại	**255**	**2047**	**32767**
E cực tiểu	**0**	**0**	**0**
Độ dịch	**127**	**1023**	**16383**

❖ Chuẩn 32 bit:

Xét trường hợp sử dụng 32 bit



Hình 2-6. Biểu diễn số thực chuẩn 32 bit

Phần định trị (M) chiếm 23 bit cho phần sau dấu phẩy nhị phân.

Phần mũ E' chiếm 8 bit.

$X= (-1)^s * 1,M * 2^{E'-127}$ (127 = 7F)

Ví dụ 2.2.8. Giả sử có một dãy 32 bit như sau:

1 0111 11111 100 0000 0000 0000 0000 0000

Hướng dẫn:

bit dấu = 1 (số âm)

$X = (-1)^1 * 1,1 * 2^{127-127} = (-1,1)$

Ví dụ 2.2.*:* Chuyển một số thực hệ 10 sang hệ 2 biểu diễn bằng số thực dấu phẩy động:

Cho số 13,2 hệ 2 $(1101,0011\ 0011\ .....)_2$

Hướng dẫn:

Di chuyển dấu phẩy về sau số 1 đầu tiên được :

$((1,101\ 0011\ 0011\ ...)*2^3) = (+1,101\ 0011\ 0011\ ...)*2^3$

Phần mũ E = E' - 127 = E' - 7F = 3

$\Rightarrow$ E' = E + 127 = 3 + 127 = $(130)_{10} = (1000\ 0010)_2$

S	E’	M
0	1000 0010	101 0011 0011 .....

Các qui ước:

Nếu E’ = 255 và M <> 0 thì x không phải là số hợp lệ

Nếu E’ = 255 và M = 0 thì x = -∞/+∞

Nếu E’ = 0 và M = 0 thì x = 0

❖ Chuẩn 64 bit:

Cũng tương tự như chuẩn 32 bit định dạng chuẩn 64 bit như sau

Bảng 2-3. Biểu diễn số thực chuẩn 64 bit

Dấu	Phần mũ	Phần định trị
1bit	11 bit	52 bit
S	E = E’ - 1023	M

❖ Chuẩn 80 bit:

Bảng 2-4. Hệ thập lục phân

Dấu	Phần mũ	Phần định trị
1bit	15 bit	111 bit
S	E = E’ - 16383	M

Ví dụ 2.2.10: Tính số thực

0100 0011 0 111 0110 0000 0000 0000 0000

Phần định trị: $1 + 2^{-1}+2^{-2}+2^{-3}+2^{-5}+2^{-6}$

$=1+0.5+0.25+0.125+0.03125+0.015625=1.921875$

Phần mũ: $2^7 + 2^2 + 2^1 = 134$

Giá trị thực (bit cao nhất là bit dấu): 134 - 128 =6

Dấu: 0 = số dương

Giá trị số thực là: $R = 1{,}921875.2^6 = +123$

**) Phương pháp đổi số thực sang số dấu phẩy động 32 bit:*

- Đổi số thập phân thành số nhị phân.
- Biểu diễn số nhị phân dưới dạng $\pm 1, xxx.2^y$
- Bit cao nhất 31: Lấy giá trị 0 với số dương, 1 với số âm.
- Xác định bit từ 30-23: được xác định bằng cách:

$y + (7F)_{16}$ hay $y+(127)_{10}$

- Phần xxx là phần định trị, được đưa vào từ bit 22....0

Ví dụ 2.2.11*:* Biểu diễn số thực $A= (9.75)_{10}$ dưới dạng dấu phẩy động 32 bit theo chuẩn IEEE 754.

Hướng dẫn:

Đổi sang dạng nhị phân: $(9.75)_{10} = (1001,11)_2 = 1,00111.2^3$

Bit dấu: do là 1 số âm nên s=1

Xác định bit từ 30-23: $y + (7F)_{16} = 3+127=(130)_{10}=(10000010)_2$

Xác định các bit từ 22...0: 00111...0

Cuối cùng số thực $(9.75)_{10}$ được biểu diễn dưới dạng dấu phẩy động 32 bit như sau:

	1	100 0001 0	001 1100 0000 0000 0000 0000
bit	|31|30	23|22	0|

Ví dụ 2.2.12: Biểu diễn số thực $A= (123)_{10}$ dưới dạng dấu phẩy động 32 bit theo chuẩn IEEE 754

Hướng dẫn:

Ta đổi sang dạng nhị phân: $(123)_{10} = (0111\ 1011)_2 = 0,1111011\ 2^7$

Bit dấu: do là 1 số dương nên bit 31 = 0.

Xác định bit từ 30-23: $7F + 7 = (86)_{16} = 86H = (1000\ 0110)_2$.

Xác định các bit từ 22...0: 1111011...0

Cuối cùng số thực $(123)_{10}$ được biểu diễn dưới dạng dấu phẩy động 32 bit như sau:

	0	100 0011 0	111 1011 0000 0000 0000 0000
bit	|31|30	23|22	0|

Vậy $A= (43760000)_{16}$

2.2.6. Biểu diễn ký tự

Để biểu diễn thông tin mà con người có thể hiểu được, máy vi tính cần một giao diện thích hợp. Nên con người đã xây dựng bộ mã để biểu diễn cho các ký tự cũng như các con số và các ký hiệu đặc biệt khác. Các mã đó gọi là *bộ mã ký tự và số.* Ta có hai bộ mã thông dụng là bộ mã ASCII (American Standard Code For Information Interchange) và bộ mã Unicode. Ngoài ra còn có chuẩn Latin1 dùng 8 bit để biểu diễn các ký tự, chuẩn ISO 10646 dùng 24 bit để biểu diễn các ký tự.

2.2.6.1. Bộ mã ASCII

Bộ mã ASCII do ANSI (American National Standar Institude) thiết kế.

Bảng mã ASCII là mã 7 bit được dùng phổ biến trong các hệ máy tính hiện nay. Với mã 7 bit nên có $2^7 = 128$ tổ hợp mã. Mỗi ký tự (chữ hoa và chữ thường) cũng như các con số thập phân từ 0…9 và các ký hiệu đặc biệt khác đều được biểu diễn bằng một mã số.

Bảng 2-5. Phân bố mã trong ASCII cơ bản

Ký tự	**Mã thập phân**	**Mã thập lục phân**
Null	0	00
31 ký tự điều khiển	1-31	01-1F
Các dấu	32-47	20-2F
Số 0-9	48-57	30-39
Các dấu khác	58-64	3A-40
A-Z	65-90	41-5A
Các dấu khác	91-96	5B-60
a-z	97-122	61-7A
Các dấu khác	123-127	7B-7F

Bảng 2-6. Bảng mã ASCII

Column bits ($B_7B_6B_5$)

Bits(row)						000	001	010	011	100	101	110	111
ROW	B_4	B_3	B_2	B_1		0	1	2	3	4	5	6	7
0	0	0	0	0	→	NUL	DLE	SP	0	@	P	\	p
1	0	0	0	1	→	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	→	STX	DC2	“	2	B	R	b	r
3	0	0	1	1	→	ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	→	EOT	DC4	$	4	D	T	d	t
5	0	1	0	1	→	ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	→	ACK	SYN	&	6	F	V	f	v
7	0	1	1	1	→	BEL	ETB	‘	7	G	W	g	w
8	1	0	0	0	→	BS	CAN	(	8	H	X	h	x
9	1	0	0	1	→	HT	EM	)	9	I	Y	i	y
A	1	0	1	0	→	LF	SUB	*	:	J	Z	j	z
B	1	0	1	1	→	VT	ESC	+	;	K	[	k	{
C	1	1	0	0	→	FF	FS	-	<	L	\	l	|
D	1	1	0	1	→	CR	GS	,	=	M	]	m	}
E	1	1	1	0	→	SO	RS	.	>	N	^	n	~
F	1	1	1	1	→	SI	US	/	?	O	_	o	DEL

Ví dụ 2.2.10: Đổi ký tự BILL thành mã ASC II

Ký tự	B	I	L	L
ASCII	1000010	1001001	1001100	1001100

HEXA 42 49 4C 4C

Một đặc điểm lý thú là mã ký tự chữ cái trừ đi 64 sẽ ra mã điều khiển.

Ví dụ 2.2.11*:* Mã ký tự chữ cái và mã điều khiển

Ký tự D H

ASCII 1000100 1001000

Thực hiện phép tính như hướng dẫn sau:

Sau khi lấy giá trị tương ứng tại bảng mã ASC II của D đem trừ đi 64_{10} ta sẽ được giá trị mới là 0000100.

Giá trị này tra ra bảng là EOT

Tương tự : Ký tự H mã điều khiển tương ứng của nó là BS

Đối với bảng chữ cái không phải là tiếng Anh chẳng hạn như tiếng Việt, ta cần nhiều mã hơn bộ mã ASCII cơ bản. Có hai cách để biểu diễn những ký tự này:

Dùng phần ASCII mở rộng gồm cả 8 bit của một byte. Ta sẽ có cả thẩy 256 ký tự, chúng đủ cho chữ cái của các thứ tiếng châu Âu, tạm đủ cho tiếng Việt, nhưng không đủ cho các chữ cái tượng hình (Hán, Hàn, Nhật).

Dùng nhiều byte để biểu diễn mã một ký tự. Phương án này sẽ khiến văn bản lớn lên rất nhiều lần so với văn bản dùng mã ASCII. Trong phương án này có chuẩn Unicode do hãng Xerox đề nghị dùng 2 byte để mã hóa một ký tự.

2.2.6.2. Bộ mã Unicode

Chuẩn Unicode do hãng Xerox đề nghị dung 2 byte để mã hóa một ký tự.

Bộ mã Unicode do các hãng sản xuất máy tính hàng đầu thiết kế. Nó là một bộ mã 6 bit và là bộ mã đa ngôn ngữ.

Bộ mã có hỗ trợ ký tự tiếng Việt (bao gồm cả chữ Nôm)

2.2.6.3. Mã BCD (Binary Coded Decimal)

Giữa hệ thập phân và hệ nhị phân còn tồn tại một hệ lai: Hệ BCD cho các số *hệ thập phân mã hoá bằng hệ nhị phân*. Ở đây dùng bốn số hệ nhị phân (bốn bit) để mã hoá một số hệ thập phân có giá trị nằm trong khoảng từ 0÷9.

0 -> 0000
1 -> 0001
2 -> 0010
3 -> 0011
4 -> 0100
5 -> 0101
6 -> 0110
7 -> 0111
8 -> 1000
9 -> 1001

Như vậy ở đây ta không dùng hết các tổ hợp có thể có của 4 bit. Vì tầm quan trọng của các số BCD nên các bộ vi xử lý thường có các lệnh thao tác với chúng.

6 tổ hợp không sử dụng là: 1010, 1011, 1100, 1101, 1110, 1111.

Phép cộng số BCD

Ví dụ 2.2.12: Thực hiện phép cộng BCD

35 + 61

$35 \rightarrow (0011\ 0101)_{BCD}$

$+\ 61 \rightarrow (0110\ 0001)_{BCD}$

$96 \leftarrow (1001\ 0110)_{BCD}$

Ví dụ 2.2.13: Thực hiện phép cộng số BCD

$87 \rightarrow (1000\ 0111)_{BCD}$

$+\ 96 \rightarrow (1001\ 0110)_{BCD}$

$183 \quad (10001\ 1101) \rightarrow$ Kết quả sai

$+\ 0110\ 0110 \leftarrow$ Hiệu chỉnh

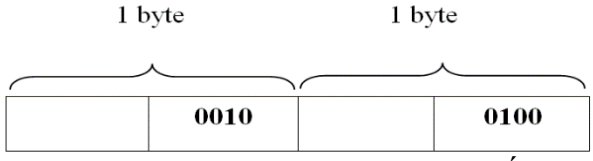
$1\ 1000\ 0011 \rightarrow 183$

Thực hiện bước hiệu chỉnh bằng cách cộng 0110 (6) vào các vị trí có nhớ

Các kiểu lưu trữ số BCD

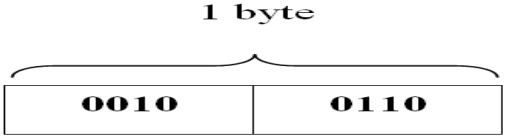
BCD không gói (Unpacked BCD): Mỗi số BCD 4 bit được lưu trữ trong 4 bit thấp của mỗi Byte.

Ví dụ 2.2.14: Số 26 được lưu trữ như sau:



BCD gói (Packed BCD): Hai số BCD được lưu trữ trong một byte

Ví dụ 2.2.15: Số 26 được lưu trữ như sau:



2.3. CÁC PHÉP TOÁN SỐ HỌC TRONG HỆ NHI PHÂN

2.3.1. Khái niệm số bù

Trong hệ thống số thông thường, để biểu diễn số âm ta chỉ cần thêm dấu "-" vào các chữ số. Tuy nhiên, trong hệ thống máy tính, ta không thể biểu diễn được như trên. Phương pháp thông dụng là bit có ý nghĩa lớn nhất MSB làm bit dấu. Nếu MSB = 1 sẽ là số âm còn MSB = 0 là số dương. Khi đó các bit còn lại sẽ biểu diễn độ lớn của nó. Như vậy, nếu ta dùng 8 bit để biểu diễn thì sẽ thu được 256 tổ hợp ứng dụng với các giá trị 0 ÷ ess (số không dấu) hay -127… -0 +0…+127 (số có dấu).

Để thuận tiện hơn trong việc tính toán số có dấu, ta dùng một dạng biểu diễn đặc biệt là số bù hai. Số bù hai của một số nhị phân xác định bằng cách lấy đảo của các bit rồi cộng thêm 1.

Ví dụ 2.3.1:

Số 7 biểu diễn là: $0000\ 0111_2$ có MSB = 0 (biểu diễn số dương).

Số bù 2 là: $1111\ 1000_2 + 1_2 = 1111\ 1001_2$

Số đại diện cho số -7 là: $1111\ 1001_2$ có MSB = 1 (biểu diễn số âm).

Như ta đã biết đảo của đảo của một số sẽ là chính nó:

$-7 = 1111\ 1001_2$

Đảo bit: $0000\ 0110_2 + 1_2 = 0000\ 0111_{(2)} = 7_{10}$

Ta thấy, để thực hiện việc xác định số bù 2 của một số A cần phải:

Biểu diễn số A theo mã bù 2 của nó.

Đảo các bit (tìm số bù 1 của A).

Cộng thêm 1 vào để nhận được số bù 2.

Khi biểu diễn theo số bù 2, nếu sử dụng 8 bit ta sẽ có các giá trị số thay đổi từ -128…127.

Bảng 2-7. Biểu diễn các số theo hệ 2, hệ 2 có dấu và mã bù 2

Số 8 bit hệ hai	Số hệ mười tương đương	Số hệ mười theo mã hệ hai có dấu	Số hệ mười theo mã bù hai
0000 0000	0	+0	+0
0000 0001	1	+1	+1
0000 0010	2	+2	+2
....	....	....	....
0111 1101	125	+125	+125
0111 1110	126	+126	+126
0111 1111	127	+127	+127
1000 0000	128	-0	-128
1000 0001	129	-1	-127
1000 0010	130	-2	-126
....	....	....	....
1111 1101	223	-125	-3
1111 1110	224	-126	-2
1111 1111	225	-127	-1

2.3.2. Các phép toán cộng trừ

2.3.2.1. Phép toán cộng

Phép cộng các số hệ 2 thực hiện giống như đối với các số hệ mười. Quy tắc cộng như sau: Y = A +B

Bảng 2-8. Phép cộng hệ nhị phân

A	**B**	**Y**	**C (Carry – Nhớ)**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ví dụ 2.3.2:

Nhớ	1111 1110
Số hạng 1	0110 0011
Số hạng 2	0101 1111
Tổng	1100 0010

Nếu kết quả của phép toán là dương thì chúng ta có được một số dương trong ký hiệu thập phân bình thường. Còn nếu kết quả là một số âm thì chúng ta có được một số âm ở dạng mã bù hai. Chú ý rằng, trong một vài ví dụ có một vài bit nhớ vượt quá giới hạn của từ thì nó sẽ bị lờ đi. Ở phép cộng bất kỳ thì kết quả có thể lớn hơn độ lớn của từ được sử dụng. Trạng thái này được gọi là tràn trên (overflow). Nếu hai số được cộng có hiện tượng tràn trên thì kết quả là ngược dấu. Khi có hiện tượng tràn trên thì ALU phải báo sự việc này ra ngoài để không sử dụng kết quả này. Chú ý rằng tràn trên có thể xuất hiện có nhớ hoặc là không [1,3,7].

2.3.2.2. Phép toán trừ

Phép trừ các số hệ 2 được thực hiện theo quy tắc sau: Y = A – B

Bảng 2-9. Phép trừ hệ nhị phân

A	**B**	**Y**	**B (Borrow – Mượn)**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Ví dụ 2.3.3:

Mượn	0110	0000
Số trừ	0110	1101
Số bị trừ	0011	0001
Hiệu	0011	1100

Để trừ một số (số bị trừ) cho một số khác (số trừ) có thể thực hiện bằng cách tạo số bù hai của số trừ và cộng với số bị trừ để được kết quả. Vì vậy, phép trừ được thực hiện thực chất là nhờ phép cộng.

Ví dụ 2.3.4:

$M = 2_{10} = 0010_2$
$S = 7_{10} = 0111_2$
$S' = 1001_2$
$M + S' = 0010_2$
$+ \underline{1001_2}$
$1011_2 = -5_{10}$

$M = -5_{10} = 1011_2$
$S = 2_{10} = 0010_2$
$S' = 1110_2$
$M + S' = 1011_2$
$+ \underline{1110_2}$
$1001_2 = -7_2$

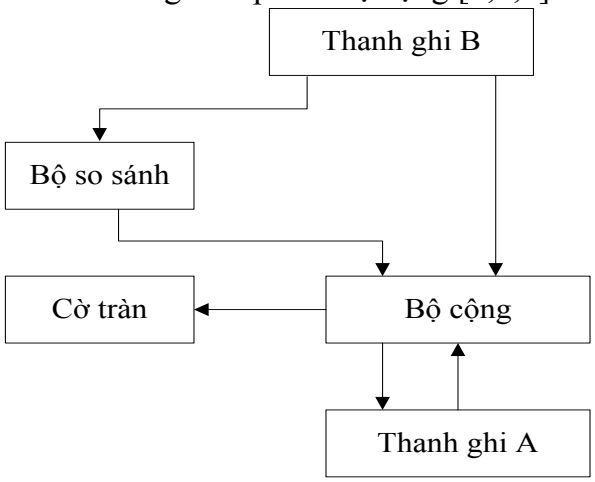
$M = 7 = 0111_2$
$S = -7 = 1001_2$
$S' = 0111_2$
$M + S' = 0111_2$
$+ \underline{0111_2}$
1110_2 = Tràn trên

$M = -6_{10} = 1010_2$
$S = 4_{10} = 0100_2$
$S' = 1100_2$
$M + S' = 1010_2$
$+ \underline{1100_2}$
0110_2 = Tràn trên

S'= Số bù 2.

Hình 2-7. đưa ra đường đi của dữ liệu và yếu tố phần cứng để thực hiện phép cộng và phép trừ. Phần tử chính là một bộ cộng nhị phân, nó nhận hai số vào và đưa ra là tổng và tín hiệu báo nếu tràn. Bộ cộng nhị phân coi hai số như là các số nguyên không dấu.

Đối với phép cộng thì hai số được đưa ra để cộng là từ hai thanh ghi, như trên hình vẽ là thanh ghi A và B. Kết quả được lưu vào một trong các thanh ghi này chứ không phải là thanh ghi thứ ba. Tín hiệu báo tràn được lưu vào trong 1 bit cờ tràn (nếu không tràn thì bit này bằng 0, nếu tràn thì bằng 1). Đối với phép trừ, số bị trừ (thanh ghi B) được gửi đến một bộ tạo mã bù hai và gửi tiếp đến bộ cộng [1,3,7].



Hình 2-7. Sơ đồ khối phần cứng của bộ cộng và trừ

2.3.3. Phép nhân số nguyên không dấu

Phép nhân trong hệ 2 được thực hiện theo quy tắc sau: Y= A x B

Bảng 2-10. Phép nhân số nguyên không dấu

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Ví dụ 2.3.5:

Số bị nhân 1 0 0 1 = 9d

Số nhân 0 1 1 0 = 6d

0 0 0 0

1 0 0 1

1 0 0 1

0 0 0 0

0 1 1 0 1 1 0 = 54d

Theo quy tắc trên ta thấy, phép nhân hệ hai của các số nguyên nhị phân không dấu có thể thực hiện theo thuật toán cộng và dịch như sau:

- Thành phần đầu tiên của tổng tích luỹ thu được là tích của số LSB trong số bị nhân với số nhân. Nếu LSB = 0 thì thành phần này cũng bằng 0, còn nếu LSB =1 thì thành phần này chính bằng số nhân.
- Mỗi thành phần thứ i tiếp theo của tổng tích luỹ sẽ được tính bằng cách tương tự nhưng phải dịch trái i bit (có thể bỏ qua các thành phần bằng 0)
-Tổng của các thành phần là tích cần tìm.
- Kết quả phép nhân hai số nguyên nhị phân n bit có độ dài 2n bit.

Tuy nhiên, chúng ta có thể thực hiện phép nhân một cách có hiệu quả hơn. Đầu tiên, ta thực hiện cộng liên tiếp các tích từng phần cho đến khi kết thúc. Điều này sẽ làm triệt tiêu sự lưu lại của tất cả các tích thành phần, dẫn đến việc sử dụng các thanh ghi ít hơn để lưu giữ các kết quả. Mặt khác, chúng ta có thể loại ra các phân tử phát sinh của tích thành phần. Đối với bit 1 trong số bị nhân thì thực hiện phép cộng và một phép dịch, nhưng đối với bit 0, thì chỉ có phép dịch là được thực hiện.

2.3.4. Phép nhân số nguyên có dấu

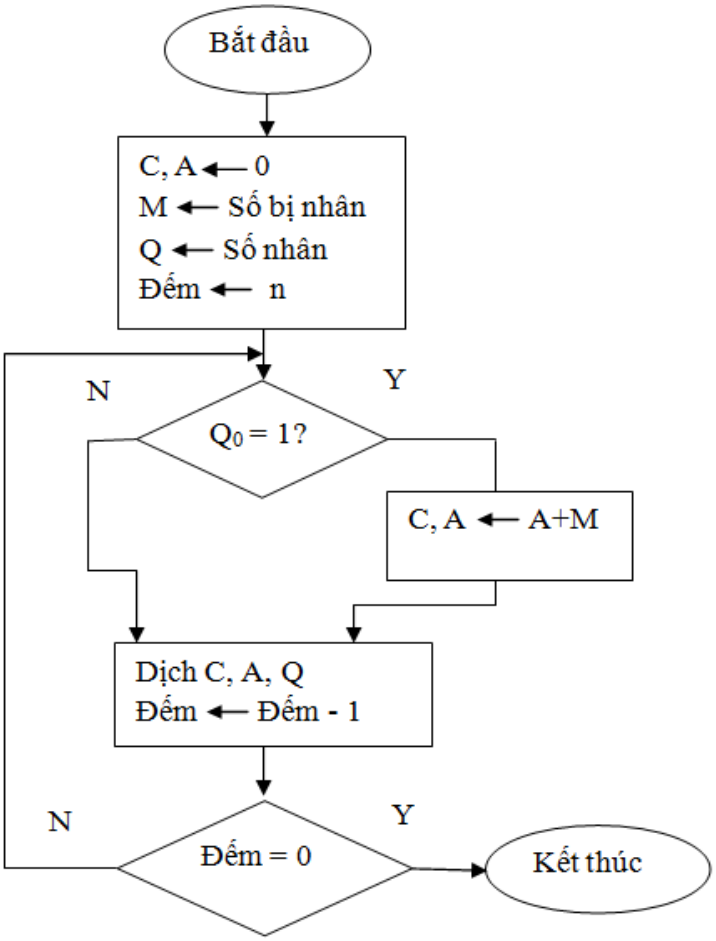
Chúng ta biết rằng phép cộng và phép trừ được số nguyên không dấu được tính như ví dụ sau:

$$\begin{array}{r} 1001_2 \\ +\ 0011_2 \\ \hline 1100_2 = 12_{10} \end{array}$$

Nếu các số trên được xem như là các số nguyên bù 2, chúng ta thấy phép cộng 1001 (-7_{10}) với 0011 (3_{10}) sẽ được 1100 (-4_{10}). Từ đó ta nhận thấy rằng các phép tính đơn giản này sẽ không áp dụng được cho phép nhân. Để thấy rõ điều này ta lấy ví dụ nhân 1011 (11_{10}) với 1101 (13_{10}) và được 10001111 (143_{10}). Nếu xem các số này như các số bù hai thì ta có 1011 (-5_{10}) nhân với 1101 (-3_{10}) sẽ được kết quả là 1000 1111 (-113_{10}). Quá trình thực hiện phép nhân này được diễn giải như sau:

Số nhân và số bị nhân được tải vào hai thanh ghi (Q và M), còn thanh ghi thứ ba là thanh ghi A, ban đầu nó được thiết lập ở mức 0. Thanh ghi 1 bit C được khởi đầu bằng 0 được dùng để lưu bit nhớ của phép cộng. Các bit của số nhân được đọc lần lượt một cách logic. Nếu Qo là 1 thì số bị nhân được cộng với thanh ghi A và kết quả được lưu trong thanh ghi A. Sau đó tất cả các bit của các thanh ghi C, A và Q được dịch sang phải một bit, các bit C dịch tới A_{n-1}, A_0 tới Q_{n-1} và Q_0 thì thôi.

Nếu Q_0 là 0 thì phép cộng không được thực hiện, chỉ thực hiện phép dịch. Quá trình này được thực hiện lặp lại với mỗi bit của số nhân. Kết quả của tích 2n bit được chứa trong các thanh ghi A và Q. Chú ý rằng trong chu kỳ thứ hai, khi một bit của số nhân là 0 thì phép cộng không được thực hiện [1,3,7].



Hình 2-8. Sơ đồ khối phép nhân hai số nhị phân không dấu

Ví dụ 2.3.6: Thực hiện phép tính $(11)_{10}$ x $(13)_{10}$

Hướng dẫn:

Chuyển đổi M= $(11)_{10}$= 1011; Q= $(13)_{10}$=1101

Các bước thực hiện	**C**	**A**	**Q**
Giá trị khởi tạo	0	0000	1101
Chu kỳ 1: Do Q_0=1 Thực hiện phép cộng A ← A+M Dịch C, A, Q	 0 0	 1011 0101	 1101 1110
Chu kỳ 2: Do Q_0=0 Thực hiện dịch C, A, Q	0	0010	1111
Chu kỳ 3: Do Q_0=1 Thực hiện phép cộng A ← A+M Dịch C, A, Q	 0 0	 1101 0110	 1111 1111
Chu kỳ 4:			

Các bước thực hiện	C	A	Q
Do Q_0=1 Thực hiện phép cộng A ← A+M Dịch C, A, Q	1 0	0001 1000	1111 1111
Kết quả:		**1000**	**1111**

Ví dụ này chứng minh rõ phép nhân sẽ không thực hiện được nếu cả hai số nhân và số bị nhân là âm. Trong thực tế nó sẽ không thực hiện được nếu một trong hai số nhân hoặc số bị nhân là âm.

Ta biết rằng, phép nhân một số nhị phân 2n bit được thực hiện nhờ sự dịch chuyển các số đó tới n bit bên trái. Vì lý do do này ta có thể tạo ra tích từng phần bằng cách thực hiện phép nhân theo cách khác.

Tích từng phần sẽ được coi như số 2n bit được tạo ra từ số bị nhân n bit. Vì vậy, cũng như số nguyên không dấu, số bị nhân 4 bit 1011 sẽ được lưu ở trong 1 từ 8 bit là 0000 1011. Mỗi tích từng phần (khác 2^0) có được từ số đó đã được dịch sang trái, các bít ở vị trí bên phải sẽ được điền bằng các số 0.

$$
\begin{array}{ll}
1011 & \\
\times\ 1101 & \\
\hline
00001011 & 1011 \times 1 \times 2^0 \\
00000000 & 1011 \times 0 \times 2^1 \\
00100100 & 1011 \times 1 \times 2^2 \\
01011000 & 1011 \times 1 \times 2^3 \\
\hline
10001111 &
\end{array}
$$

Giờ đây chúng ta có thể làm sáng tỏ được rằng phép nhân sẽ không làm việc nếu số bị nhân là âm.

Vấn đề là mỗi lần số bị nhân là âm thì tích từng phần phải là số âm trên vùng 2n bit. Bit dấu của tích từng phần phải sắp xếp lại.

Ta chứng minh điều này dựa vào ví dụ bảng 3.2, biểu diễn phép nhân số 1001 (9_{10}) với 0011 (3_{10}). Nếu được coi như là số nguyên không dấu thì phép nhân 9×3=27 được thực hiện đơn giản.

Tuy nhiên nếu 1001 thì được coi như là số bù hai thì giá trị của nó là -7. Sau đó mỗi tích từng phần phải được như là số bù hai âm của các bít 2n (8 bit), được biểu diễn như bảng 2-11.

Chú ý rằng phép tính phải được thực hiện bằng cách kéo dài mỗi tích từng phần sang bên trái với số bit nhị phân bằng 1.

Bảng 2-11. Phép nhân số nguyên có dấu

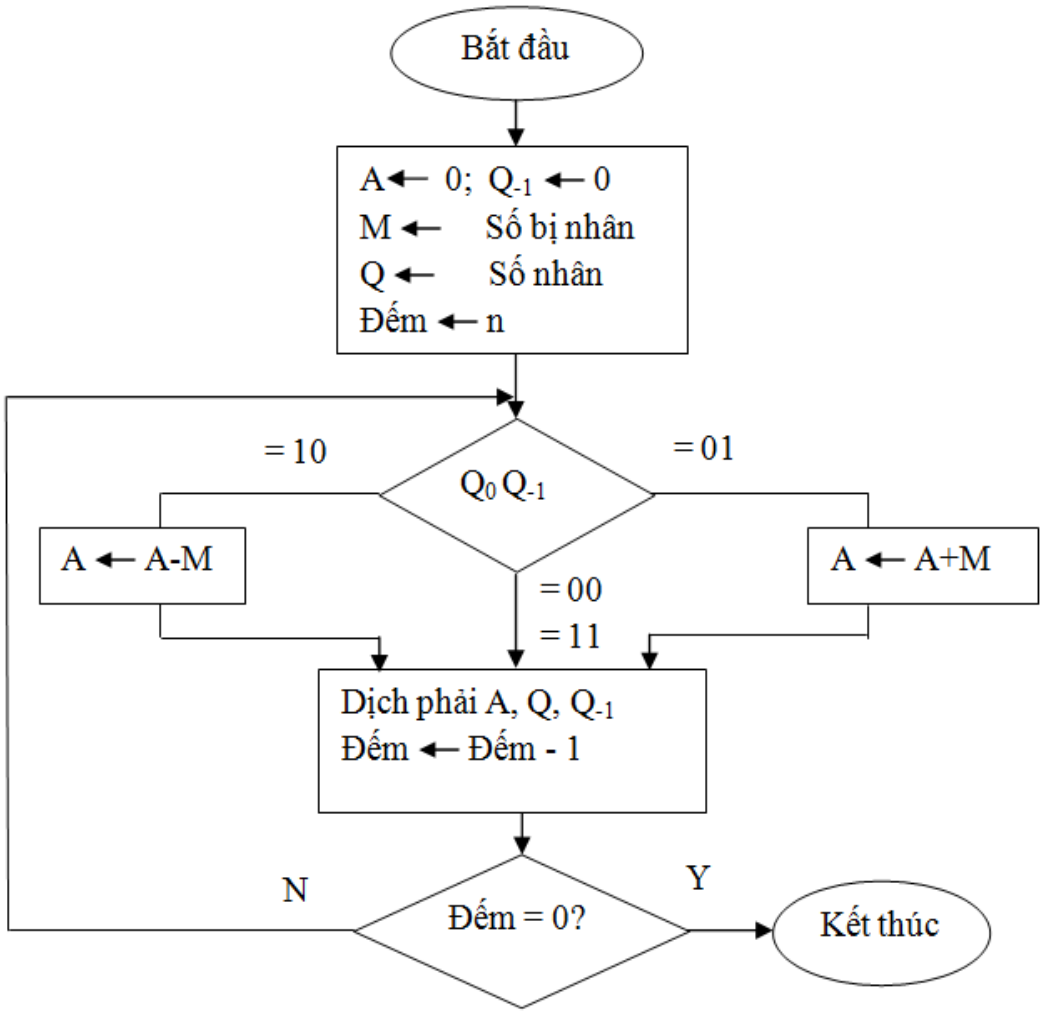
1011 (9)	1001 (-7)
x 0011 (3)	x 0011 (3)
00001001 (1001)x2^0	11111001 (-7)x2^0 = (-7)
00010010 (1001)x2^1	11110010 (-7)x2^1 = (-14)
00011011 (27)	11101011 (-21)
Các số nguyên không dấu	Các số nguyên bù 2

Một trong số những thuật toán được sử dụng để thực hiện phép nhân các số nguyên bù hai là thuật toán Booth. Thuật toán này rất có lợi để tăng tốc độ thực hiện các phép nhân, nó có liên quan đến việc tính toán gần đúng.

**) Thuật toán Booth được miêu tả trong hình 2-8 và được mô tả như sau:*

Giống như phần trên, số nhân và số bị nhân được đặt riêng rẽ trong các thanh ghi Q và M. Ở đây thanh ghi 1 bít được đặt một cách lôgic vào bên phải của bít có nghĩa nhỏ nhất của thanh ghi Q và được định nghĩa là Q-1. Kết của phép nhân sẽ xuất hiện trong các thanh ghi A và Q. A và Q được khởi tạo bằng 0. Giống như trước thì các bít của phép nhân sẽ kiểm tra một cách có hệ thống theo thời gian. Bây giờ, mỗi bít đã được kiểm tra và bít của bên phải của nó cũng được kiểm tra.

Nếu hai bít giống nhau (11 hay 00) thì tất cả các bít của thanh ghi A và Q và Q1 được dịch sang bên phải 1 bít. Nếu hai bít này khác nhau thì số bị nhân được cộng hay trừ từ thanh ghi A, tuỳ theo hai bít là 01 hay 10. Khi thực hiện phép cộng hay phép trừ thì đều được dịch sang phải. Kết quả của tích 2n bit được chứa trong các thanh ghi A và Q. Trong mỗi trường hợp, sự dịch phải là ta thực hiện dịch bít ở phía bên trái nhất của A đó là An-1, không những được dịch vào An-2 mà còn vào An-1. Đây là yêu cầu để giữ nguyên dấu của các số trong A và Q. Đây được hiểu như là phép dịch số học, từ đó sẽ giữ nguyên bit dấu.



Hình 2-9. Thuật toán Booth cho phép nhân số bù hai

Để rõ hơn ta mô tả phép toán trong các ví dụ sau. Ở các ví dụ này ta có thể thấy nó thực hiện phép nhân với bất kỳ các số dương và số âm.

Ví dụ 2.3.7: Thực hiện phép nhân của số (-7) với 5 theo thuật toán Booth.

Hướng dẫn:

Chuyển đổi M= $(-7)_{10}$= 1001; Q= $(5)_{10}$= 0101

Các bước thực hiện	**A**	**Q**	**Q_{-1}**
Giá trị khởi tạo	0000	0101	0
Chu kỳ 1: Do Q_0Q_{-1} =10 Thực hiện phép gán A ← A-M Dịch phải A, Q, Q_{-1}	 0001 0011	 0101 1010	 0 1
Chu kỳ 2: Do Q_0Q_{-1} =01 Thực hiện phép gán A ← A+M Dịch phải A, Q, Q_{-1}	 1100 1110	 1010 0101	 1 0

Các bước thực hiện	A	Q	Q_{-1}
Chu kỳ 3: Do Q_0Q_{-1} =10 Thực hiện phép gán A ← A-M Dịch phải A, Q, Q_{-1}	 0101 0010	 0101 1010	 0 1
Chu kỳ 4: Do Q_0Q_{-1} =01 Thực hiện phép gán A ← A+M Dịch phải A, Q, Q_{-1}	 1011 1101	 1010 1101	 1 0
Kết quả:	**1101**	**1101**	

Ví dụ 2.3.8: Thực hiện phép nhân của số (-5) với (-6)

Hướng dẫn:

Chuyển đổi M= (-5)= 1011; Q= (-6)= 1010

Các bước thực hiện	A	Q	Q_{-1}
Giá trị khởi tạo	0000	1010	0
Chu kỳ 1: Do Q_0Q_{-1} =00 Dịch phải A, Q, Q_{-1}	 0000	 0101	 0
Chu kỳ 2: Do Q_0Q_{-1} =10 Thực hiện phép gán A ← A-M Dịch phải A, Q, Q_{-1}	 0101 0010	 0101 1010	 0 1
Chu kỳ 3: Do Q_0Q_{-1} =01 Thực hiện phép gán A ← A+M Dịch phải A, Q, Q_{-1}	 1101 1110	 1010 1101	 1 0
Chu kỳ 4: Do Q_0Q_{-1} =10 Thực hiện phép gán A ← A-M Dịch phải A, Q, Q_{-1}	 0011 0001	 1101 1110	 0 1
Kết quả:	**0001**	**1110**	

2.3.5. Phép chia số nguyên không dấu

Phép chia là một phép tính phức tạp hơn so với các phép tính khác nhưng về cơ bản thì nó vẫn được dựa trên những nguyên tắc chung.

Đầu tiên là việc xây dựng thuật toán và các phép dịch các bit, phép cộng hoặc trừ. Phép chia có thể được thực hiện bằng các phép trừ và phép dịch liên tiếp cho đến khi không còn gì để trừ hoặc số bị trừ nhỏ hơn số chia.

Ví dụ 2.3.9: 215 / 22 = 9 dư 17. Thực hiện phép chia này trong hệ 2.

Số bị chia 215 = 1 1 0 1 0 1 1 1

Số chia 22 = 1 0 1 1 0

Số bù 1 của số chia = 0 1 0 0 1

Số bù 2 của số chia = 0 1 0 1 0

Số bị chia 1 1 0 1 0 1 1 1 | 1 0 1 1 0

Số bù 2 0 1 0 1 0 | 1 0 0 1

Tổng nhỏ hơn số chia 0 0 1 0 0 1

Hạ hàng 0 0 1 0 0 1 1

Hạ hàng 0 0 1 0 0 1 1 1

Số bù 2 0 0 1 0 1 0 1 0

Số dư 0 0 0 1 0 0 0 1

Ví dụ trên được diễn giải chi tiết như sau: Đầu tiên kiểm tra từng bít của số bị chia từ trái qua phải cho đến khi nào nhóm các bit đó biểu diễn một số lớn hơn hoặc bằng số chia, có nghĩa là số bị chia có thể chia được cho số chia. Nếu còn chưa chia được cho số chi thì ta thêm các số 0 vào thương số từ trái sang phải. Khi chia được cho số chia thì ta thêm 1 vào thương số và lấy số đó trừ đi số chia. Kết quả của phép trừ được gọi là số dư thành phần (partial remainder). Từ đây, phép chia lặp lại các bước như trên. Mỗi lần lặp như vậy ta bổ xung một bit từ số bị chia vào số dư thành phần cho đến khi nào nó lớn hơn số bị chia. Giống như trên, ta lại trừ đi số chia để dược một số dư thành phần mới. Quá trình cứ lặp đi lặp lại như trên cho đến khi tất cả các bit của số bị chia đã hết. Tuy nhiên, phép chia thực hiện theo cách trên rất khó khăn cho các VXL vốn chỉ gồm các phần tử để thực hiện phép cộng và dịch. Để khắc phục các khó khăn này người ta dùng thuật toán sau [1,3,7]:

- Đổi số chia ra số bù 2 của nó.
- Lấy số bị chia cộng với số bù 2 của số chia (trừ đi số chia).

Nếu kết quả này có bit dấu bằng 0 (nghĩa là phần này của số bị chia chia được cho số chia) thì bit tương ứng của thương bằng 1.

Nếu kết quả này có bit dấu bằng 1 (nghĩa là phần này của số bị chia không chia được cho số chia) thì bit tương ứng của thương bằng 0 và buộc phải khôi phục lại giá trị ban đầu của số bị chia bằng cách cộng kết quả với số chia ở mã hệ 2.

- Dịch trái kết quả thu được ở trên và làm lại bước 2 cho đến khi nhận được kết quả là 0 (chia hết) hoặc nhỏ hơn số chia (chia còn dư).

Bây giờ ta thực hiện lại ví dụ trên theo thuật toán này.

Số bị chia 215 = 0 1 1 0 1 0 1 1 1

Số chia 22 = 0 1 0 1 1 0

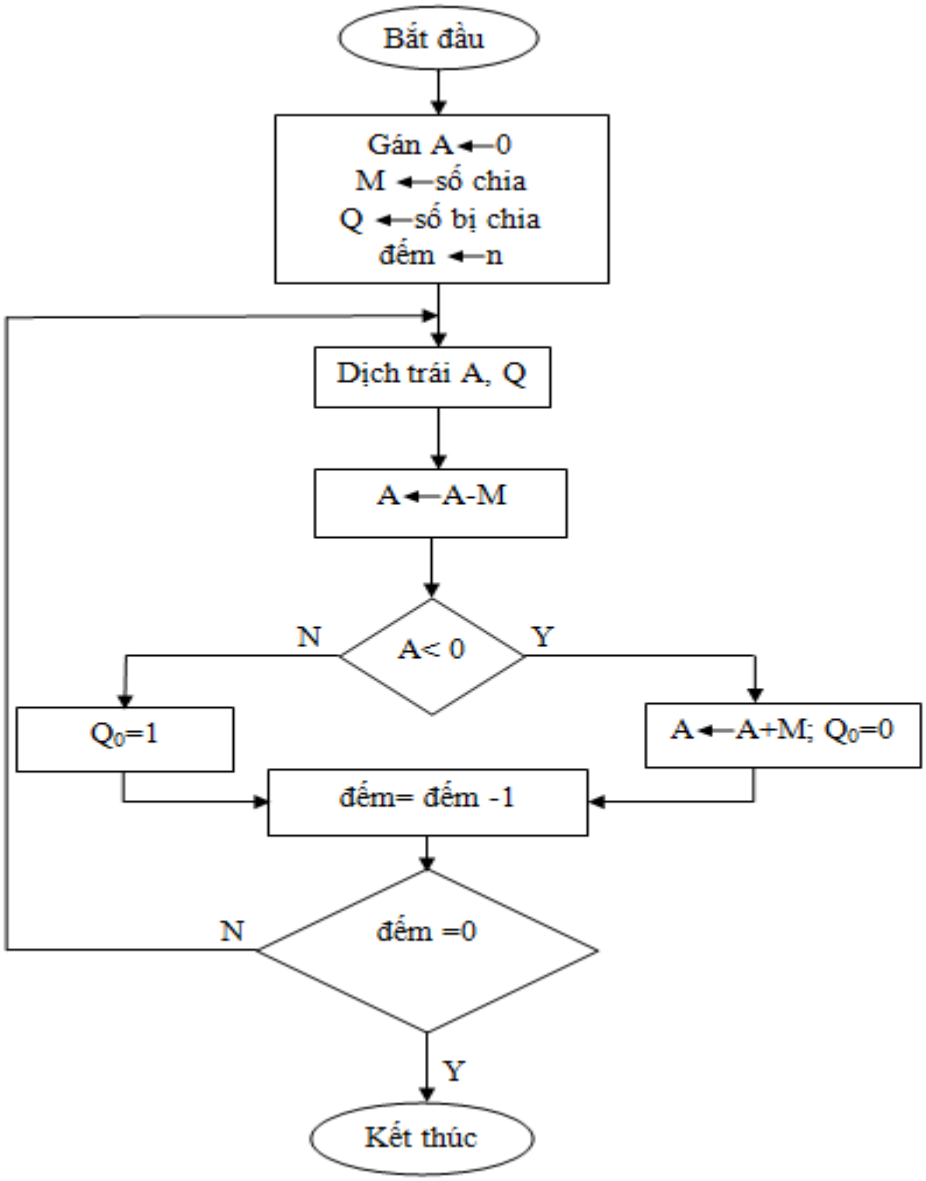
Số bù 2 của số chia = 1 0 1 0 1 0

Các bước thực hiện như sau:

Số bị chia	0 1101 0111	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 1	0 0010 0111	Bit dấu = 0 → thương = 1
Dịch trái tổng 1	0 0100 111	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 2	1 1001 111	Bit dấu = 1 → thương = 0
Số chia ở mã hệ 2 để cộng	0 1011 0	
Số bị chia	0 0100 111	
Dịch trái số bị chia	0 1001 11	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 3	1 1110 11	Bit dấu = 1 → thương = 0
Số chia ở mã hệ 2 để cộng	0 1011 0	
Số bị chia	0 1001 11	
Dịch trái số bị chia	1 0011 1	
Số chia ở mã bù 2 để cộng	1 0101 0	
Tổng 4	0 1000 1	Bit dấu = 0 → thương = 1

Bit dấu

Ta cũng có kết quả tương tự. Hình 3.6 là thuật toán tương ứng để thực hiện phép chia dài. Số chia được đặt trong thanh ghi M, số bị chia thì đặt trong thanh ghi Q. Ở mỗi bước, cùng một lúc thanh ghi A và Q đều được dịch trái một bit. M được trừ A để xác định là có chia phần số dư A hay không. Nếu có thì thì Q_0 nhận được một bit 1, nếu không thì Q_0 nhận được một bit 0 và M phải được cộng ngược trở lại A để trở lại giá trị trước. Bộ đếm sẽ giảm xuống và tiếp tục thực hiện cho đến bước thứ n. Để kết thúc thì thương số phải nằm trong thanh ghi Q và phần dư phải nằm trong thanh ghi A.



Hình 2-10. Lưu đồ thuật toán phép chia số nhị phân không dấu

Ví dụ 2.3.10: Thực hiện phép tính 9:3

Hướng dẫn:

Đổi ra mã nhị phân: Q= $(9)_{10}$= 1001; M= $(3)_{10}$ = 0011

Các bước thực hiện	**A**	**Q**	**M**
Giá trị khởi tạo	0000	1001	0011
Chu kỳ 1:			
Dịch trái A, Q	0001	0010	
Gán A← A-M	1110	0010	
A< 0, thực hiện phép phục hồi A← A+M và gán Q_0=0	0001	0010	
Chu kỳ 2:			
Dịch trái A, Q	0010	0100	

Các bước thực hiện	A	Q	M
Gán A← A-M	1111	0100	
A<0, thực hiện phép phục hồi A← A+M và gán Q_0=0	0010	0100	
Chu kỳ 3:			
Dịch trái A, Q	0100	1000	
Gán A← A-M	0001	1000	
A> 0, thực hiện phép gán Q_0=1	0001	1001	
Chu kỳ 4:			
Dịch trái A, Q	0011	0010	
Gán A← A-M	0000	0010	
A=0, thực hiện phép gán Q_0=1	0000	0011	
Kết quả:	**0000 (Số dư)**	**0011 (Thương số)**	

2.3.6. Phép chia số nguyên có dấu

Phép chia như trên có thể thực hiện được, nhưng gặp một số khó khăn khi mở rộng tới các số âm. Vì vậy, khi số bị chia hoặc số chia là một số âm thì thuật toán có thể được tổng quát thành một số bước như sau [1,3,7]:

- Bước 1. Chuyển đổi số bị chia và số chia về thành số dương tương ứng.
- Bước 2. Sử dụng thuật giải chia số nguyên không dấu để chia hai số dương, kết quả nhận được là thương Q và phần dư A đều là dương.
- Bước 3. Hiệu chỉnh dấu của kết quả như sau:

(Lưu ý: phép đảo dấu thực chất là thực hiện phép lấy bù hai)

Bảng 2-12. Phép chia số nguyên có dấu

Số bị chia	Số chia	Thương	Số dư
Dương	Dương	Giữ nguyên	Giữ nguyên
Dương	Âm	Đảo dấu	Giữ nguyên
Âm	Dương	Đảo dấu	Đảo dấu
Âm	Âm	Giữ nguyên	Đảo dấu

Ví dụ 2.3.11: Thực hiện phép chia (-10): 4.

Hướng dẫn:

Bước 1: Chuyển đổi số bị chia và số chia thành số dương tương ứng:

Q= (10)10= 1010; M=(4)10= 0100

Bước 2: Thực hiện phép chia số nguyên không dấu

Các bước thực hiện	A	Q	M
Giá trị khởi tạo	0000	1010	0100
Chu kỳ 1:			
Dịch trái A, Q	0001	0100	
Gán A← A-M	1101	0100	
	0001	0100	

Các bước thực hiện	A	Q	M
A<0, thực hiện phép phục hồi A← A+M và gán Q_0=0			
Chu kỳ 2: Dịch trái A, Q Gán A← A-M A<0, thực hiện phép phục hồi A← A+M và gán Q_0=0	 0010 1110 0010	 1000 1000 1000	
Chu kỳ 3: Dịch trái A, Q Gán A= A-M A>0, thực hiện phép gán Q_0=1	 0101 0001 0001	 0000 0000 0001	
Chu kỳ 4: Dịch trái A, Q Gán A← A-M A<0, thực hiện phép phục hồi A← A+M và gán Q_0=0	 0010 1110 0010	 0010 0010 0010	
Kết quả:	**0010 Số dư**	**0010 Thương số**	

Bước 3: Hiệu chỉnh dấu kết quả ta được:

Thương số = Q= 1110; Số dư= A= 1110

Ví dụ 2.3.12: Thực hiện phép chia:

a) 10: (-4)

b) (-10): (-4)

Dựa vào phép chia ở ví dụ trên ta chỉ cần thực hiện hiệu chỉnh dấu kết quả và thu được kết quả như sau:

a) Thương số = Q= 1110; Số dư = A= 0010

b) Thương số = Q= 0010; Số dư= A= 1110

2.3.7. Phép toán với số dấu phẩy động

Ở nội dung này trình bày các tính toán cơ bản trong tính toán dấu phẩy động.

Một dấu phẩy động được biểu diễn như sau:

$$A = \pm M_A.2^{\pm E_A}$$

$$B = \pm M_B.2^{\pm E_B}$$

Trong đó:

M_A, M_B là phần định trị

E_A, E_B là các số mũ

Với các phép cộng, phép trừ, chúng ta cần phải đảm bảo là chúng có cùng số mũ (thường đưa về số mũ lớn hơn). Điều này có thể dẫn đến yêu cầu phải thay đổi cơ số của một trong hai toán hạng. Phép chia và phép nhân thì đơn giản hơn nhiều.

2.3.7.1. Phép cộng và trừ

Trong các phép tính số học với số dấu phảy động, phép cộng và trừ phức tạp hơn nhiều so với phép nhân và phép chia vì nó cần xắp xếp. Thuật toán cho phép cộng và phép trừ có 4 bước cơ bản sau:

Bước 1: Kiểm tra số mũ:

Nếu $E_A > E_B$ lấy E_A làm chuẩn và ngược lại

Bước 2: Quy đồng phần định trị

Nếu $E_A > E_B$ đặt $M'_B = M_B.2^{-(E_A - E_B)}$

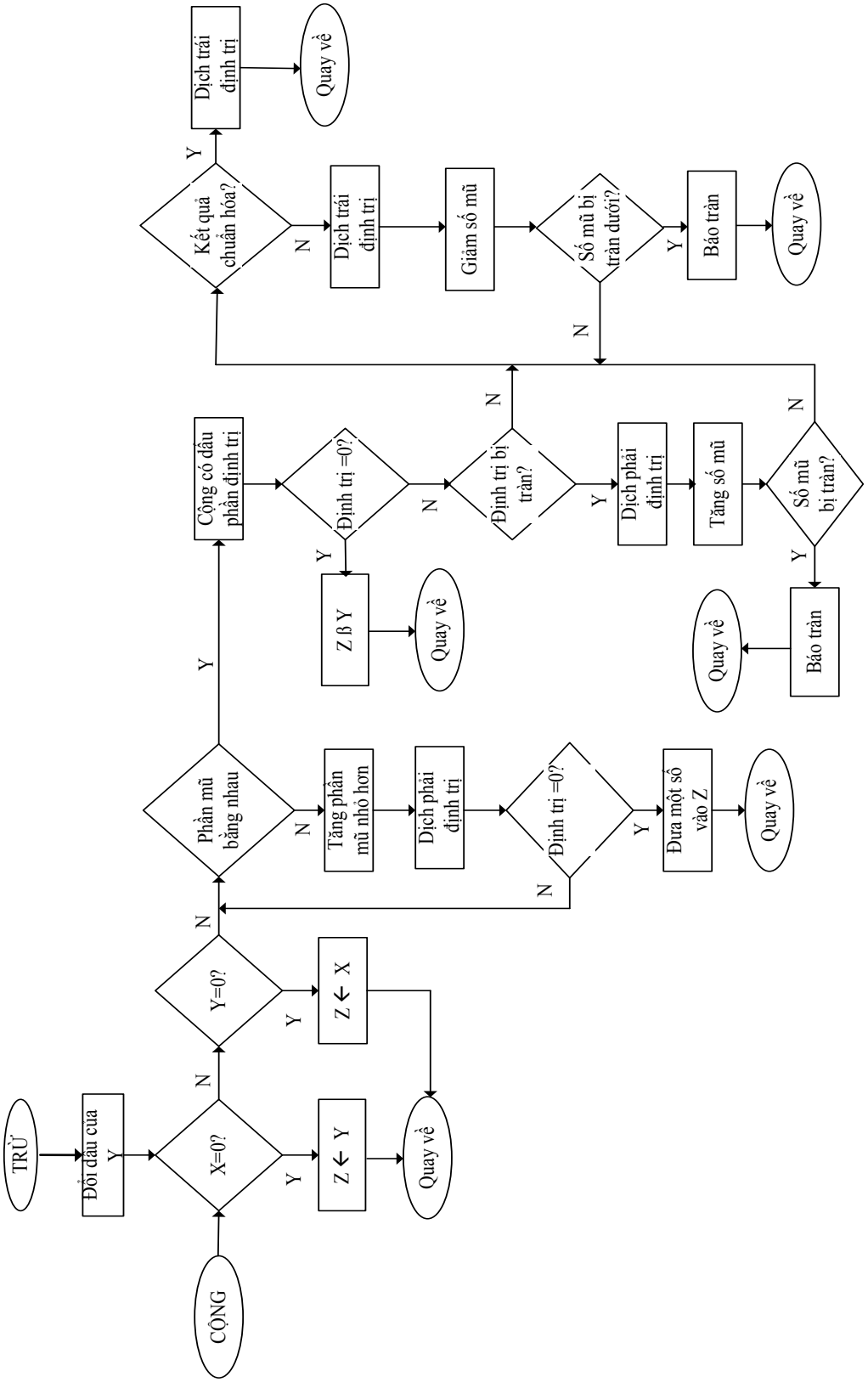
Nếu $E_A < E_B$ đặt $M'_A = M_A.2^{-(E_B - E_A)}$

Bước 3: Cộng hoặc trừ phần định trị

Kết quả = $(M_A \pm M'_B).2^{E_A}$ nếu $E_A > E_B$

Kết quả = $(M'_A \pm M_B).2^{E_B}$ nếu $E_A < E_B$

Bước 4: Chuẩn hóa kết quả và báo tràn nếu có.



Hình 2-11. Lưu đồ các bước chính để thực hiện phép cộng hoặc trừ dấu phẩy động

Chúng ta cần phải đưa các số theo định dạng dấu phảy động. Để thực hiện được các phép cộng và phép trừ cần phải đưa các số vào các thanh ghi mà ta sẽ sử dụng cho ALU.

Nếu định dạng dấu phẩy động mà bit phần định trị được ẩn đi thì bit đó phải ẩn trong quá trình tính toán. Các phần mũ và phần định trị được lưu trong các thanh ghi riêng rẽ và sẽ được tổng hợp lại khi có kết quả.

Vấn đề của phép cộng và phép trừ là đồng nhất dấu, nếu chúng ta thực hiện phép trừ thì quá trình được bắt đầu bằng cách thay đổi dấu của số bị trừ. Tiếp theo, nếu một trong hai toán hạng bằng 0 thì toán hạng kia sẽ được lưu như là kết quả.

Bước tiếp theo là thao tác với các số, trong đó hai phần mũ được cân bằng nhau. Để hiểu rõ điều này, ta coi như cộng số thập phân như sau:

$123x100 + 456x10^{-2}$

Chúng ta không chỉ cộng phần định trị. Đầu tiên các số phải thiết lập về vị trí tương đương nhau, như là, số 4 ở số thứ hai phải thẳng hàng với số 3 ở số thứ nhất. Với các qui định dưới đây, các số mũ sẽ bằng nhau, các số được cộng sẽ được biểu diễn như sau:

$123x100 + 456x10^{-2} = 123x100 + 4,56x100$

Để làm tròn ta di chuyển các số nhỏ hơn về bên phải (tăng số mũ) hoặc di chuyển số lớn hơn về phía trái. Việc này có thể dẫn tới kết quả bị mất các số và kết quả thu được nhỏ hơn số bị làm tròn. Bất kỳ số nào bị mất đi đều có phần định trị tương đối nhỏ. Quá trình làm tròn được lặp đi lặp lại cho đến khi nào bên phải phần định trị của số bị chia có có giá trị là 1 và tăng số mũ tương ứng lên đúng bằng số lần dịch chuyển (chú ý rằng với cơ số 16 thì một lần di chuyển một số là phải di chuyển 4 bit). Nếu quá trình này cho kết quả phần định trị là 0 thì số khác được lưu lại như là kết quả. Vì vậy nếu hai số có cùng phần mũ, nhưng khác nhau phần định trị, thì số nhỏ hơn sẽ bị mất.

Tiếp theo, hai phần định trị được cộng với nhau, kèm theo cả việc tính toán phần dấu của chúng. Dấu của chúng có thể khác nhau do đó kết quả có thể bằng 0. Cũng có các số có phần định trị có khả năng bị tràn trên một số hạng. Cũng như phần định trị của kết quả đã bị di chuyển về phía bên phải và phần mũ đã được tăng. Một số mũ bị tràn có thể xuất hiện như là một kết quả và việc này có thể được lưu lại và kết thúc mọi hoạt động.

Bước tiếp theo là tiêu chuẩn hoá kết quả. Tiêu chuẩn hoá phù hợp là việc di chuyển phần định trị sang trái cho đến khi số cao nhất của phần định trị (1 bit hoặc 4 bit với cơ số 16) là khác 0. Mỗi lần di chuyển tương ứng với một lần giảm số mũ đi 1 và điều này có thể dẫn tới số mũ bị tràn dưới. Tóm lại là kết quả phải được làm tròn và lưu lại.

Ví dụ 2.3.13: Cho A= 0,110. 2^{110}; B= 0,1101. 2^{101}

Thực hiện phép tính:

a) A+B

b) A-B

Hướng dẫn:

a) Thực hiện phép tính A + B

Bước 1:

Xét: E_A= 0110 => $E_{A\ thuận}$= 0000 0110

$-E_B$= -101 => $-E_{B\ thuận}$= 1111 0101=> $-E_{B\ ngược}$= 1111 1010=> $-E_{B\ bù}$= 1111 1011

E_A+ (-E_B)= E_A+ (-$E_{B\ bù}$)= 0000 0110 + 1111 1011= 0000 0001

=> n=1 > 0 chọn E_A làm chuẩn

Bước 2: đặt $M'_B = M_B.2^{-(E_A-E_B)} = 0,111.2^{-1} = 0,0111$

Bước 3: thực hiện phép toán đối với phần định trị

M_A+M'$_B$= 0101 0000+ 0011 1000 = 1000 1000

Kết quả= $(M_A + M'_B).2^{E_A} = 1,0001.2^{110}$

Bước 4: Chuẩn hóa kết quả

Kết quả= $0,10001.2^{111}$

b) Thực hiện phép tính A - B

có M_A= 0101 000; M'$_B$= 0011 1000

M_A-M'$_B$= 0101 0000 - 0011 1000 = 0001 1000 (có tràn)

Kết quả= $(M_A - M'_B).2^{E_A} = 0,0011000.2^{110} = 0,11.2^{100}$

Ví dụ 2.3.14: Cho A= - 0,101. 2^{-101}; B= - 0,1. 2^{1000}

Thực hiện phép tính:

a) A+B

b) A-B

Hướng dẫn:

a) Thực hiện phép tính A + B

Bước 1:

E_A= -101 => $E_{A\ ngược}$= 1111 1010 => $E_{A\ bù}$= 1111 1011

-E_B= 1000 => -$E_{B\ thuận}$= -$E_{B\ bù}$ = 0000 1000

$E_{A\ bù}$ + (-$E_{B\ bù}$)= 1111 1011 + 0000 1000 = 0000 0011

=> n=3 > 0 chọn E_A làm chuẩn

Bước 2: đặt $M'_B = M_B.2^{-(E_A-E_B)} = -0,1.2^{-3} = -0,0001$

Bước 3: thực hiện phép toán đối với phần định trị

có M_A= 0101 0000 => $M_{A\ ngược}$= 1010 1111=> $M_{A\ bù}$= 1011 0000

M'$_B$= 0000 1000 => M'$_{B\ ngược}$= 1111 0111 => M'$_{B\ bù}$ =1111 1000

M_A+M'$_B$= $M_{A\ bù}$+M'$_{B\ bù}$ =1011 0000 + 1111 1000 = 1010 1000

Kết quả= ($M_{A\ bù}$+M'$_{B\ bù}$). 2^{E_A} = 1,0101000. 2^{-101} =0,10101. 2^{-100}

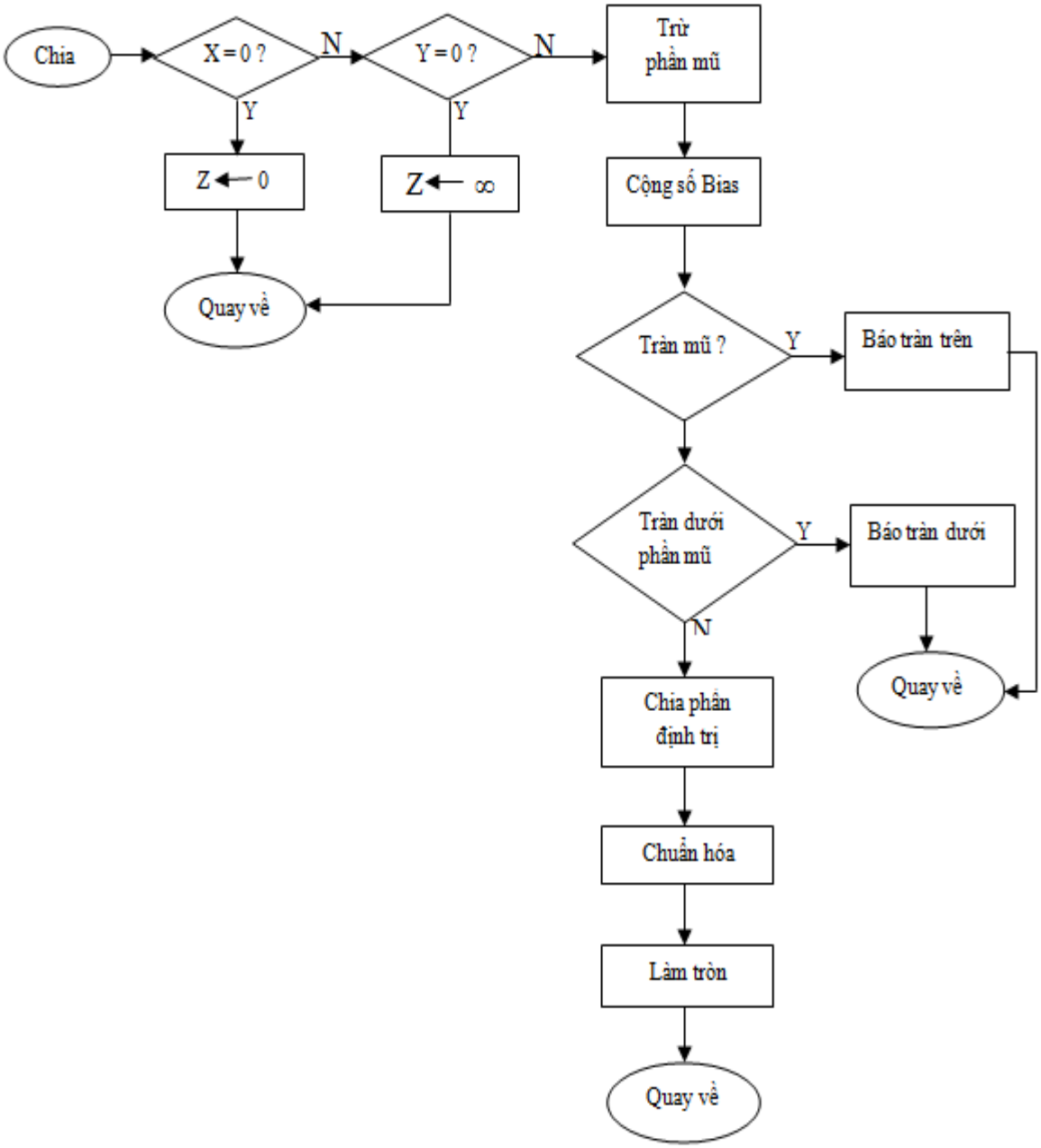
b) Thực hiện phép tính A - B

có M_A-M'$_B$= $M_{A\ bù}$ - M'$_{B\ bù}$ =1011 0000 - 1111 1000= 1101 1000

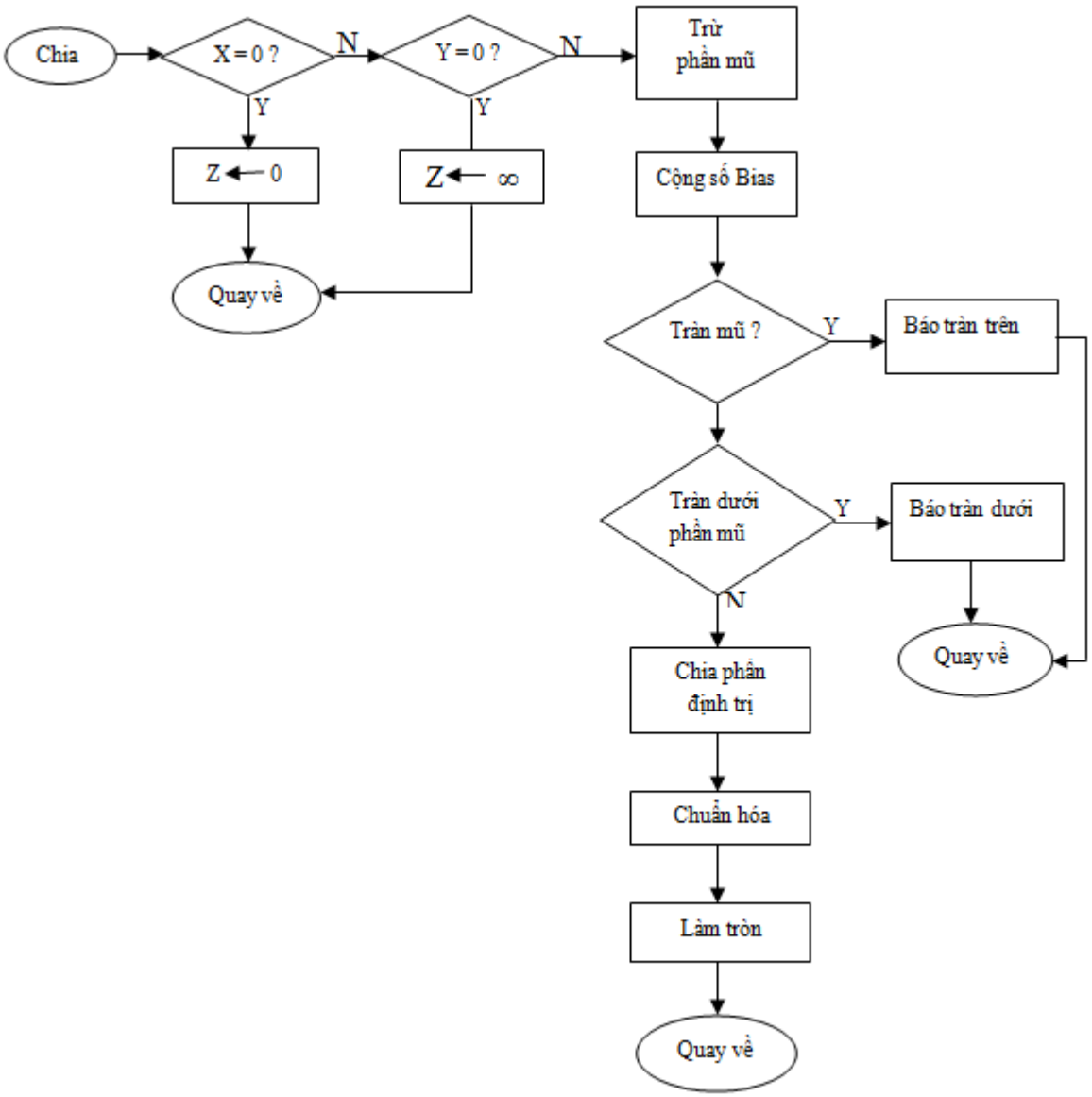
Kết quả= $(M_A - M'_B).2^{E_A} = 1,1011000.2^{-101} = 0,11011.2^{-100}$

2.3.7.2. Phép nhân và chia

Phép nhân và chia dấu phảy động là một việc đơn giản hơn so với phép cộng và phép trừ. Các bước tính toán với phép nhân, được minh hoạ trong hình 2-12. Trước tiên là thực hiện kiểm tra 0, nếu một toán hạng bằng không thì kết quả bằng 0. Bước tiếp theo là cộng số mũ. Nếu số mũ được lưu dưới dạng độ lệch (Bias) thì tổng của số mũ có thể gấp đôi số Bias. Do đó giá trị của độ lệch phải được trừ từ tổng. Kết quả là có thể từng số mũ bị tràn trên hoặc tràn dưới, kết quả này có thể được lưu lại và kết thúc thuật toán [1,3,7]. Các bước tính toán với phép chia dấu phảy động được minh họa trong hình 2-13.



Hình 2-12. Phép nhân dấu phảy động



Hình 2-13. Phép chia dấu phảy động

3.3.7.3. Phép làm tròn

Để hiểu rõ hơn về cách làm tròn. Kết quả của bất kỳ phép tính nào thì phần định trị cũng được lưu ở thanh ghi dài hơn. Khi kết quả được đưa ra dưới dạng dấu phảy động thì các bit thêm vào phải được xắp xếp.

Mỗi một con số đều có phương pháp cho phép làm tròn. Thực ra trong tiêu chuẩn IEEE có 4 phương pháp làm tròn.

- Làm tròn đến số gần nhất: kết quả được làm tròn tới số gần nhất được biểu diễn.
- Làm tròn tới + ∞: kết quả được làm tròn lên đến dương vô cùng.
- Làm tròn tới – ∞: kết quả được làm tròn về âm vô cùng.
- Làm tròn tới 0: kết quả được làm tròn tới giá trị 0.

Mỗi phương pháp có những đặc điểm riêng. Đối với làm tròn tới số gần nhất là là cách làm tròn ngầm định trong tiêu chuẩn IEEE và được định nghĩa như sau: là giá trị

gần nhất được biểu diễn sẽ là: Nếu hai giá trị được biểu diễn gần nhất là gần bằng nhau, thì bít 0 bit nhỏ nhất của phần định trị sẽ được gán bằng.

Với ví dụ, nếu các bit thêm vào mà vượt khỏi 23 bit thì có thể được lưu lại, với số là 10010 thì các bit thêm vào lên đến hơn một nửa của các bit đã được biểu diễn với số dương trước đó. Trong trường hợp này câu trả lời đúng là cộng thêm số binary 1 vào bit cuối cùng, tiếp tục làm tròn lên tới số tiếp theo để biểu diễn. Bây giờ thì các bit thêm vào là 01111. Trong trường hợp này, các bít được thêm vào lên đến hơn một nữa của các bit đã được biểu diễn số dương trước đó. Câu trả lời đúng đơn giản là rơi vào bit thêm vào, mà có kết quả của phép làm tròn xuống tới số tiếp theo được biểu diễn.

Trong tiêu chuẩn cũng chỉ ra trường hợp đặc biệt của các bit thêm vào của định dạng 1000. . .. ở đây kết quả là chưa hoàn toàn chính xác giữa hai giá trị có thể được biểu diễn. một có thể làm tròn theo phương pháp luôn luôn cắt bỏ. Tuy nhiên có một phương pháp khác đơn giản là đưa ra một số nhỏ nhưng độ lệch tích luỹ trong bước tiếp theo của các phép tính. Yêu cầu ở đây là phương pháp không làm tròn độ lệch. Một có thể làm tròn lên hoặc xuống với độ lệch của các số ngẫu nhiên, hoặc trung bình, thì kết quả có thể là không làm tròn độ lệch. Ngược lại việc làm tròn với đối số là không thể dự đoán theo một qui định trước nào. Làm tròn số chẵn theo tiêu chuẩn IEEE là bắt buộc: Nếu kết quả có độ chính xác trung bình giữa hai số có thể được biểu diễn, giá trị được làm tròn lên nếu cuối cùng của số được biểu diễn là 1 và kết quả được làm tròn xuống nếu bit thấp nhất của số được biểu diễn có trạng thái hiện hành là 0.

Tiếp theo là sự lựa chọn thứ hai, việc làm tròn về dương và âm vô cùng, ta biết một phương pháp hoàn toàn có thể thực hiện được như với các phép tính số học. Một số khái niệm về phép tính số học trước đây: Tại bước cuối cùng của hàng loạt phép tính toán dấu phảy động, chúng ta không biết được kết quả chính xác bởi vì điều đó còn phụ thuộc vào phần cứng, mà tạo ra việc làm tròn kết quả. Nếu chúng ta cho phép mỗi một phép tính được thực hiện hai lần liên tục: lần thứ nhất là làm tròn lên và lần thứ hai là làm tròn xuống, và kết quả đúng là trung bình cộng của hai kết quả đó. Nếu giới hạn của số làm tròn lên và làm tròn xuống được thu hẹp lại một cách đầy đủ, thì kết quả thu được thật sự chính xác. Nếu không, thì ít nhất chúng ta cũng biết điều đó và có thể cho phép phân tích đươc phép cộng.

Phương pháp có tính quyết định trong tiêu chuẩn là làm tròn về 0. Đây chỉ đơn giản là sự cắt bỏ: các bít thêm vào bị bỏ qua. Đây quả thật là phương pháp đơn giản nhất. Tuy nhiên kết quả mà bị cắt bỏ các bit đi thì thường nhỏ hơn hoặc bằng với giá trị gốc, quá trình này tạo ra một độ lệch xuống (downward). Đây là một độ lệch liên tiếp mà ta đã bàn tới trước kia, từ độ lệch này ảnh hưởng đến mọi tính toán cho mỗi một bit thêm vào là 0.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 2

Câu 1. Thực hiện phép toán sau bằng thanh ghi 8 bit

a) S1 = 61 - 79
b) S2 = 65 - 9
c) S3 = 12 – 25
d) S4 = 24 - 12

Câu 2. Thực hiện phép toán sau bằng thanh ghi 16 bit

a) S1 = 329 - 163
b) S2 = 365 - 139
c) S3 = 134 – 134
d) S4 = 120 - 86

Câu 3. Thực hiện phép toán sau

a) $S1 = (123)_{BCD} + (546)_{BCD}$
b) $S2 = (789)_{BCD} + (456)_{BCD}$
c) $S3 = (123)_{BCD} + (115)_{BCD}$
d) $S4 = (127)_{BCD} + (240)_{BCD}$

Câu 4. Biểu diễn các số thực sau về dạng dấu phẩy động 32 bít.

$(58,25)_{10}$; $(-25,75)_{10}$

Câu 5. Xây dựng và áp dụng giải thuật nhân cải tiến MxQ.

a) (-15x6).
b) (-14x7).
c) (-13x8).
d) (-12x9).

Câu 6. Xây dựng và áp dụng giải thuật chia cải tiến M:Q.

a) (-15:6).
b) (-14:7).
c) (-13:8).
d) (-12:9).

Câu 7. Thực hiện phép tính: $A \pm B$

Khi biết: $A = -0,111 \times 2^{-1001}$; $B = -0,101 \times 2^{101}$

Chương 3
MỨC LOGIC SỐ

Trang bị cho sinh viên kiến thức cơ bản về mạch logic số trong hệ thống máy tính: Hàm Boole, các phương pháp tối thiểu hóa bằng bìa các nô, cổng (gate) và đại số logic cơ bản, một số mạch logic.

Giúp sinh viên làm quen và hiểu được các mạch logic cơ bản và cách thức thiết kế mạch qua phần mềm mô phỏng bằng Logisim.

3.1. HÀM BOOLE

3.1.1. Giới thiệu chung:

Đại số Boole (đại số logic) là một tập hợp các đối tượng có hai trạng thái: có hoặc không có, mệnh đề đúng hoặc sai; các đối tượng này được biểu diễn bằng biến logic. Khi trạng thái đối tượng là tồn tại thì biến logic biểu diễn có giá trị là 1 và ký hiệu là A; nếu không thì biến logic của nó có giá trị là 0 và ký hiệu là $\bar{A}$

Giữa các biến logic, người ta định nghĩa 3 phép toán cơ sở:

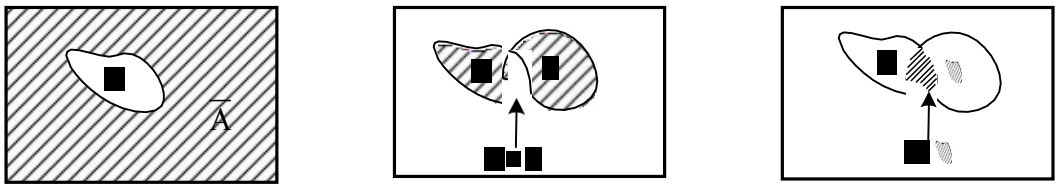
Phép phủ định logic đối với một biến A hay còn gọi là phép đảo. Khi nhận tác động của phép toán này, A sẽ nhận giá trị đảo với giá trị ban đầu và ký hiệu là $\bar{A}$

Phép cộng logic (phép hoặc) được ký hiệu bằng dấu "+". Ví dụ, (A + B), mỗi biến được gọi là một số hạng và kết quả gọi là tổng.

Phép nhân logic (phép và) được ký hiệu bằng dấu ".". Ví dụ, (A . B), mỗi biến được gọi là một thừa số và kết quả gọi là tích.

Có thể dùng giản đồ Venn trong lý thuyết tập hợp để biểu diễn 3 phép toán logic trên.

Một trạng thái của đối tượng nào đó luôn có thì biến logic biểu diễn nó luôn có giá trị 1 ngược là thì nhận giá trị 0. Ta nhận được trong tập hợp này hai hằng số 0 và 1.



Hình 3-1. Đồ thị Venn mô tả ba phép tính cơ bản

Để thể hiện các hàm logic bằng mạch điện người ta sử dụng các cổng logic. Các cổng logic được xây dựng dựa trên cấu hình mạch chuyên biệt được gọi là họ mạch logic điển hình là: Mạch logic Điện trở - Transistor (RTL), Mạch logic Điốt – Transistor (DTL), Mạch logic Transistor – Transistor (TTL), CMOS…

Trong chương này sẽ trình bày các ký hiệu cổng logic chủ yếu và được dùng phổ biến hiện nay.

3.1.2. Đại số Boole

Có ba loại quan hệ logic cơ bản nhất là: ĐẢO, HOẶC, VÀ. Mạch điện thực hiện ba phép toán cơ bản là cổng NOT, OR và AND. Ngoài ba phép toán cơ bản trên còn có các phép toán logic khác như: NAND, NOR, XOR, XNOR…

Các tín hiệu vào còn được gọi là các biến logic vào, tín hiệu ra được gọi là hàm ra. Trong đại số logic, biến số và hàm số đều chỉ lấy hai giá trị là "0" và "1". Mỗi biến số biểu thị một điều kiện để sự kiện có thể phát sinh. Điều kiện đó chỉ có thể có hoặc không. Hàm số biểu thị bản thân sự kiện đó có phát sinh hay không phát sinh.

3.1.2.1. Các định lý cơ bản

Vì trong đại số logic chỉ có thể có hai hằng số 0 và 1 nên các biến logic cũng chỉ lấy một trong hai giá trị đó.

Do đó, xuất hiện các định lý cơ bản sau:

Bảng 3-1. Một số định lý cơ bản trong đại số Boole

STT	Tên gọi	Dạng tích	Dạng tổng
1	Đồng nhất	A.1 = A	A + 0 = A
2	Phần tử 0, 1	A.0 = 0	A + 1 = 1
3	Bù	$A.\overline{A} = 0$	$A + \overline{A} = 1$
4	Bất biến	A.A = A	A + A = A
5	Hấp thụ	A + A.B = A	A.(A + B) = A
6	Hoàn nguyên	$\overline{\overline{A}} = A$	
7	Định lý DeMorgan	$(\overline{A.B.C\ldots}) = \overline{A} + \overline{B} + \overline{C}$ ..	$(\overline{A} + \overline{B} + \overline{C}) = (\overline{A}.\overline{B}.\overline{C})$

3.1.2.2. Các định luật cơ bản

+ Hoán vị: A.B = B.A, A+B = B+A
+ Kết hợp: A.(B.C)=(A.B).C, A+(B+C)=(A+B)+C
+ Phân phối: A.(B+C)=A.B+A.C; (A+B).(A+C)=A+B.C
+ Nhất quán: nếu A + B = B thì A.B = A

3.1.2.3. Ba quy tắc về đẳng thức

a) Quy tắc thay thế

Trong bất kỳ đẳng thức logic nào nếu muốn thay một biến nào đó bằng một hàm số thì đẳng thức vẫn được thiết lập.

Quy tắc này có ứng dụng rất lớn trong việc biến đổi công thức đã biết để tạo ra công

thức mới, mở rộng phạm vi ứng dụng của công thức đã biết.

Ví dụ 3.1.1: Ta có công thức $(\overline{A+B}) = \bar{A}.\bar{B}$ Dùng F= A+C thay vào biến *A, ta có:*

$\overline{(A+C)+B} = \overline{A+C}.\overline{B} = \overline{A}.\overline{C}.\overline{B}$ hay $\overline{A+B+C} = \overline{A}.\overline{B}.\overline{C}$

b) Quy tắc tìm đảo của hàm số

Phép đảo của hàm số được thực hiện bằng cách đổi dấu nhân thành dấu cộng và ngược lại; đổi 0 thành 1 và ngược lại; đổi biến nguyên thành biến đảo và ngược lại. Ngoài ra, những dấu đảo nào của hàm nhiều biến vẫn phải giữ nguyên, và tuân thủ theo quy tắc đổi "nhân trước, cộng sau ".

c) Quy tắc đối ngẫu

Hàm F và F' là đối ngẫu với nhau khi các dấu cộng và dấu nhân; số '0' và số '1' đổi chỗ

cho nhau một cách tương ứng.

Ví dụ 3.1.2: F = A . (B + C) thì F' = A + B .C

Do quy tắc đối ngẫu nên các định lý cơ bản có thể viết dưới 2 dạng đối ngẫu nhau là: dạng tích và dạng tổng.

3.1.3. Các phương pháp biểu diễn hàm Boole

Như đã nói ở trên, hàm logic được thể hiện bằng những biểu thức đại số như các môn toán học khác. Đây là phương pháp tổng quát nhất để biểu diễn hàm logic. Ngoài ra, một số phương pháp khác cũng được dùng để biểu diễn loại hàm này. Mỗi phương pháp đều có ưu điểm và ứng dụng riêng của nó. Dưới đây là nội dung của một số phương pháp thông dụng.

3.1.3.1. Bảng trạng thái

Bảng trạng thái liệt kê giá trị (trạng thái) mỗi biến theo từng cột và giá trị hàm theo một cột riêng (thường là bên phải bảng). Bảng trạng thái còn được gọi là ***bảng sự thật*** hay ***bảng chân lý***.

Đối với hàm *n* biến sẽ có 2^n tổ hợp độc lập. Các tổ hợp này được kí hiệu bằng chữ m_i, với $i = 0$ đến $2^n - 1$ (xem bảng 1-2) và có tên gọi là các **hạng tích** hay còn gọi là **minterm**.

Bảng 3-2. Bảng trạng thái hàm 3 biến

m	A	B	C	f
m_0	0	0	0	0
m_1	0	0	1	1
m_2	0	1	0	1
m_3	0	1	1	0
m_4	1	0	0	1
m_5	1	0	1	0
m_6	1	1	0	0
m_7	1	1	1	1

Đặc điểm của bảng trạng thái:

+ Rõ ràng, trực quan. Sau khi xác định các giá trị biến vào có thể tìm được giá trị đầu ra nhờ bảng trạng thái. Do vậy, trong các sổ tay tra cứu đều giới thiệu bảng trạng thái để độc giả biết được chức năng logic của mạch.

+ Để giải quyết bài toán ở dạng logic thì sử dụng bảng trạng thái là hữu ích nhất. Do vậy, trong quá trình thiết kế mạch số việc đầu tiên nên làm là lập bảng trạng thái.

Nhược điểm chủ yếu của bảng trạng thái là sẽ phức tạp nếu số biến quá nhiều, không thể dùng các công thức và định lý để tính toán.

3.1.3.2. Phương pháp đại số

Có 2 dạng biểu diễn là dạng *tuyển* (*tổng các tích*) và dạng *hội* (*tích các tổng*).

+ Dạng tuyển: Mỗi số hạng của tổng được gọi là một *hạng tích* hay *minterm* (đủ biến)**,**

và thường kí hiệu bằng chữ **"*mi*"** (chỉ số ***i*** được tính trong hệ thập phân).

+ Dạng hội: Mỗi thừa số là *hạng tổng* hay *maxterm* (đủ biến)**,** thường được kí hiệu bằng chữ **"*Mi*".** Nếu trong tất cả mỗi hạng tích hay hạng tổng có đủ mặt các biến, thì dạng tổng các tích hay tích các tổng tương ứng được gọi là dạng *chuẩn*. Dạng chuẩn là duy nhất.

Bảng 3-3. là các minterm và Maxterm của hàm 2 biến

Biến		minterm (m_i)	Maxterm (M_i)
A	B		
0	0	$\bar{A}\,\bar{B} = m_0$	$A + B = M_0$
0	1	$\bar{A}\,B = m_1$	$A + \bar{B} = M_1$
1	0	$A\bar{B} = m_2$	$\bar{A} + B = M_2$
1	1	$AB = m_3$	$\bar{A} + \bar{B} = M_3$

Bảng 3-4 là các minterm và Maxterm của hàm 3 biến

Biến			minterm	Maxterm
A	B	C	(m_i)	(M_i)
0	0	0	$\bar{A}\,\bar{B}\,\bar{C} = m_0$	$A + B + C = M_0$
0	0	1	$\bar{A}\,\bar{B}C = m_1$	$A + B + \bar{C} = M_1$
0	1	0	$\bar{A}B\bar{C} = m_2$	$A + \bar{B} + C = M_2$
0	1	1	$\bar{A}BC = m_3$	$A + \bar{B} + \bar{C} = M_3$
1	0	0	$A\bar{B}\bar{C} = m_4$	$\bar{A} + B + C = M_4$
1	0	1	$A\bar{B}C = m_5$	$\bar{A} + B + \bar{C} = M_5$
1	1	0	$AB\bar{C} = m_6$	$\bar{A} + \bar{B} + C = M_6$
1	1	1	$ABC = m_7$	$\bar{A} + \bar{B} + \bar{C} = M_7$

Tổng quát, hàm logic *n* biến có thể biểu diễn chỉ bằng một dạng tổng các tích:

$$f(X_{n-1}, \dots, X_0) = \sum_{i=0}^{2^n-1} a_i m_i \qquad (1.1)$$

hoặc bằng chỉ một dạng tích các tổng:

$$f(X_{n-1}, \dots, X_0) = \prod_{i=0}^{2^n-1} a_i + M_i \qquad (1.2)$$

Ở đây, a_i chỉ lấy hai giá trị 0 hoặc 1. Đối với một hàm thì minterm và maxterm là bù của nhau.

Ví dụ 3.1.3: *Biểu diễn hàm sau theo dạng minterm:*

F(A, B, C) = A + BC→ Đây là dạng minterm không đầy đủ. Muốn đưa về dạng chuẩn tắc (đủ biến) ta sử dụng một số định lý đã nêu để biến đổi.

$$F(A,B,C) = A + BC = A(B + \bar{B})(C + \bar{C}) + (A + \bar{A})BC$$
$$= ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + ABC + \bar{A}BC$$
$$= ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC$$

Đây là dạng chuẩn minterm.

Tuy nhiên, biểu diễn này khá dài nên mỗi một hạng tích được thay thế bằng ký hiệu *mi* tương ứng (xem bảng 1.3). Lưu ý, nguyên biến (biến không đảo) được thay bằng số "12" và đảo biến được thay bằng số "02". Như vậy, biểu thức có dạng:

$$F(A,B,C) = ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC$$
$$\downarrow\downarrow\downarrow \quad \downarrow\downarrow\downarrow \quad \downarrow\downarrow\downarrow \quad \downarrow\downarrow\downarrow \quad \downarrow\downarrow\downarrow$$
$$111_2 \quad 101_2 \quad 110_2 \quad 100_2 \quad 011_2$$
$$7_{10} \quad 5_{10} \quad 6_{10} \quad 4_{10} \quad 3_{10}$$
$$F(A,B,C) = m_7+m_5+m_6+m_4+m_3 = m_7+m_6+m_5+m_4+m_3 = \sum(3,4,5,6,7)$$

a) Biểu diễn hàm sau theo dạng Maxterm:

F(A, B, C) = A + BC =(A + B)(A + C) → Đây là dạng Maxterm không đầy đủ. Muốn đưa về dạng chuẩn (đủ biến) ta sử dụng một số định lý đã nêu để biến đổi.

$$F(A,B,C) = A + BC = (A + B)(A + C) = (A + B + C\bar{C})(A + C + B\bar{B})$$
$$= (A + B + C)(A + B + \bar{C})(A + C + B)(A + C + \bar{B})$$
$$= (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)$$

Giống như minterm, người ta cũng biểu diễn hàm logic theo ký hiệu Mi. Trong đó nguyên biến được thay thế bằng số "0_2" và đảo biến thay bằng số "1_2". Do đó, ta viết biểu thức thành dạng sau:

$$F(A,B,C) = (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)$$
$$\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$$
$$(0 \quad 0 \quad 0)_2(0 \quad 0 \quad 1)_2(0 \quad 1 \quad 0)_2$$
$$0_{10} \qquad 1_{10} \qquad 2_{10}$$
$$F(A,B,C) = M_0.M_1.M_2 = \prod(0,1,2)$$

Nhận xét:

Đối với dạng minterm: m_i được gọi là số hạng nhỏ nhất. Số hạng nhỏ nhất có các tính chất sau:

+ Bao gồm tất cả các biến của hàm trong một thừa số; mỗi biến số chỉ xuất hiện một lần dưới dạng thừa số hoặc là nguyên biến hoặc là đảo biến.

+ Tích của hai số hạng nhỏ nhất bất kỳ luôn bằng 0

+ Tổng của tất cả các số hạng nhỏ nhất luôn bằng 1

Đối với dạng Maxterm: Mi được gọi là thừa số lớn nhất. Thừa số lớn nhất có các tính chất sau:

+ Bao gồm tất cả các biến của hàm;

+ Mỗi biến số chỉ xuất hiện một lần dưới dạng tổng của thừa số hoặc là nguyên biến hoặc là đảo biến.

+ Tổng của hai thừa số lớn nhất bất kỳ luôn bằng 1

+ Tích của tất cả các thừa số luôn bằng 0

Ưu điểm của phương pháp đại số:

+ Dùng các ký hiệu logic biểu diễn mối quan hệ logic giữa các biến làm cho cách viết gọn, cách viết này có tính khái quát và trừu tượng cao.

+ Rất tiện sử dụng các công thức và định lý của đại số Boole để biến đổi.

+ Tiện cho việc sử dụng sơ đồ logic để thực hiện hàm số. Chỉ dùng các ký hiệu logic của mạch điện cổng tương ứng thay thế phép toán xét trong biểu thức hàm số thì ta được một sơ đồ logic.

Nhược điểm chính của phương pháp này là khó xác định giá trị hàm ứng với tổ hợp biến một cách trực tiếp đối với các hàm phức tạp (không trực quan như bảng trạng thái).

3.1.3.3. Phương pháp bảng Các nô.

a) Bảng Karnaugh hay phương pháp hình học được tổ chức như sau: Một hàm logic có n biến sẽ có 2^n ô (mỗi ô tương ứng với một minterm m_i của hàm). Các tổ hợp biến phải xếp theo thứ tự mã Gray nghĩa là các hạng tích trong hai *ô kế cận* chỉ khác nhau một biến. Các tổ hợp biến được viết theo một dòng (thường là phía trên) và một cột (thường là bên trái).

b) Tính tuần hoàn của bảng Các nô: Không những các ô kế cận khác nhau một biến mà các ô đầu dòng và cuối dòng, đầu cột và cuối cột cũng chỉ khác nhau một biến (kể cả 4 góc vuông của bảng) nên các ô này cũng gọi là ô kế cận.

c) Cách ghi giá trị hàm trên bảng Cácnô: Muốn thiết lập bảng Các nô của một hàm đã cho dưới dạng chuẩn tổng các tích (minterm), ta chỉ việc ghi giá trị 1 vào các ô ứng với hạng tích có mặt trong biểu diễn (ứng với $a_i = 1$), các ô còn lại sẽ lấy giá trị 0 hoặc được bỏ trống. Nếu hàm cho dưới dạng tích các tổng (Maxterm), cách làm cũng tương tự, các ô ứng với hạng tổng có trong biểu diễn lại lấy giá trị 0 (ứng với $a_i = 0$), và các ô khác lấy giá trị 1.

d) Cấu tạo bảng Các nô cho hàm 3 biến, 4 biến và 5 biến

Bảng 3-5. Các nô cho hàm 3 biến

A \ BC	**00**	**01**	**11**	**10**
0	m_0 $\bar{A}\bar{B}\bar{C}$	m_1 $\bar{A}\bar{B}C$	m_3 $\bar{A}BC$	m_2 $\bar{A}B\bar{C}$
1	m_4 $A\bar{B}\bar{C}$	m_5 $A\bar{B}C$	m_7 ABC	m_6 $AB\bar{C}$

Bảng 3-6. Các nô cho hàm 4 biến

CD / AB	00	01	11	10
00	m_0 $\bar{A}\bar{B}\bar{C}\bar{D}$	m_1 $\bar{A}\bar{B}\bar{C}D$	m_3 $\bar{A}\bar{B}CD$	m_2 $\bar{A}\bar{B}C\bar{D}$
01	m_4 $\bar{A}B\bar{C}\bar{D}$	m_5 $\bar{A}B\bar{C}D$	m_7 $\bar{A}BCD$	m_6 $\bar{A}BC\bar{D}$
11	m_{12} $AB\bar{C}\bar{D}$	m_{13} $AB\bar{C}D$	m_{15} $ABCD$	m_{14} $ABC\bar{D}$
10	m_8 $A\bar{B}\bar{C}\bar{D}$	m_9 $A\bar{B}\bar{C}D$	m_{11} $A\bar{B}CD$	m_{10} $A\bar{B}C\bar{D}$

Bảng 3-7. Bảng Các nô cho hàm 5 biến

CDE / AB	000	001	011	010	110	111	101	100
00	m_0 $\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$	m_1 $\bar{A}\bar{B}\bar{C}\bar{D}E$	m_3 $\bar{A}\bar{B}\bar{C}DE$	m_2 $\bar{A}\bar{B}\bar{C}D\bar{E}$	m_6 $\bar{A}\bar{B}CD\bar{E}$	m_7 $\bar{A}\bar{B}CDE$	m_5 $\bar{A}\bar{B}C\bar{D}E$	m_4 $\bar{A}\bar{B}C\bar{D}\bar{E}$
01	m_8 $\bar{A}B\bar{C}\bar{D}\bar{E}$	m_9 $\bar{A}B\bar{C}\bar{D}E$	m_{11} $\bar{A}B\bar{C}DE$	m_{10} $\bar{A}B\bar{C}D\bar{E}$	m_{14} $\bar{A}BCD\bar{E}$	m_{15} $\bar{A}BCDE$	m_{13} $\bar{A}BC\bar{D}E$	m_{12} $\bar{A}BC\bar{D}\bar{E}$
11	m_{24} $AB\bar{C}\bar{D}\bar{E}$	m_{25} $AB\bar{C}\bar{D}E$	m_{27} $AB\bar{C}DE$	m_{26} $AB\bar{C}D\bar{E}$	m_{30} $ABCD\bar{E}$	m_{31} $ABCDE$	m_{29} $ABC\bar{D}E$	m_{28} $ABC\bar{D}\bar{E}$
10	m_{16} $A\bar{B}\bar{C}\bar{D}\bar{E}$	m_{17} $A\bar{B}\bar{C}\bar{D}E$	m_{19} $A\bar{B}\bar{C}DE$	m_{18} $A\bar{B}\bar{C}D\bar{E}$	m_{22} $ACD\bar{E}$	m_{23} $A\bar{B}CDE$	m_{21} $A\bar{B}C\bar{D}E$	m_{20} $A\bar{B}C\bar{D}\bar{E}$

Ví dụ 3.1.4: Xây dựng bảng Các nô cho hàm logic sau:
F(A, B, C, D) = Σ(0, 1, 5, 7, 10, 14, 15)

CD / AB	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	0	0	0	1

Ưu điểm nổi bật nhất của bảng Các nô là tính kề nhau về logic của các số hạng nhỏ nhất (minterm), nó biểu thị rõ ràng thành sự liền kề hình học của các ô trong bảng. Do vậy, rất dễ dàng tối thiểu hóa hàm.

Nhược điểm là do có quá nhiều ô nên trong trường hợp nhiều biết việc tổ chức bảng rất phức tạp. Do đó, chỉ nên dùng bảng Các nô cho trường hợp hàm logic có số biến nhỏ hơn 6.

3.1.4. Các phương pháp tối thiểu hóa (rút gọn hàm)

Trong thực tế, khi viết một hàm logic dưới dạng nào đó, thì dạng có được không phải là dạng duy nhất. Thông thường nếu biểu thức càng đơn giản thì mạch điện cũng đơn giản. Khi thiết kế mạch phải đảm bảo sao cho các phần tử trong

mạch phải tối thiểu nhất để tiết kiệm chi phí, do vậy, người thiết kế phải sử dụng các phương pháp để tối thiểu hóa hàm logic.

Có ba phương pháp tối thiểu hoá. Nếu số biến số tương đối ít (n ≤ 6) khi đó dùng phương pháp hình vẽ, phương pháp này dùng bảng Cácnô. Nếu số biến tương đối nhiều dùng phương pháp đại số hoặc dùng phương pháp Mc. Quine Cluskey.

3.1.4.1. Phương pháp đại số

Dựa vào các định lý đã học để đưa biểu thức về dạng tối giản.

Ví dụ 3.1.5: Hãy đưa hàm logic về dạng tối giản:

$$F(A,B,C) = \bar{A}B + AC + BC$$

Lời giải:

Áp dụng định lý, $\bar{A} + A = 1;\ A + A.B = A$ ta có:

$f = AB + \bar{A}C + BC(A + \bar{A})$

$= AB + ABC + \bar{A}C + \bar{A}BC$

$= AB + \bar{A}C$

Nhận xét:

Từ ví dụ trên ta thấy: nếu trong tổng các tích, xuất hiện một biến và đảo của biến đó trong hai số hạng khác nhau, các thừa số còn lại trong hai số hạng đó tạo thành thừa số của một số hạng thứ ba thì số hạng thứ ba đó là thừa và có thể bỏ đi.

3.1.4.2. Phương pháp bảng Các nô

Phương pháp này thường được dùng để rút gọn các hàm có số biến không vượt quá 5. Các bước tối thiểu hóa:

a) Đối với minterm:

Gộp các ô kế cận có giá trị '1' (hoặc '0') lại thành từng nhóm *2, 4, ..., 2^i* ô. Số ô trong mỗi nhóm càng lớn kết quả thu được càng tối giản tức là nếu gộp được 2^n ô thì ta tối giản được *n* biến. Một ô có thể được gộp nhiều lần trong các nhóm khác nhau. Nếu gộp theo các ô có giá trị '0' ta sẽ thu được biểu thức bù của hàm.

Thay mỗi nhóm bằng một hạng tích mới, trong đó giữ lại các biến giống nhau theo dòng và cột.

Cộng các hạng tích mới lại, ta có hàm đã tối giản.

b) Đối với Maxterm:

Gộp các ô kế cận có giá trị '0' (hoặc '1') lại thành từng nhóm 2, 4, ..., 2^i ô. Số ô trong mỗi nhóm càng lớn kết quả thu được càng tối giản. Một ô có thể được gộp nhiều lần trong các nhóm khác nhau. Nếu gộp theo các ô có giá trị '1' ta sẽ thu được biểu thức bù của hàm.

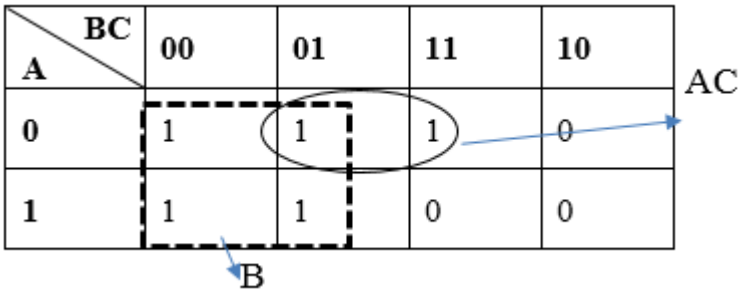
Thay mỗi nhóm bằng một hạng tổng mới, trong đó giữ lại các biến giống nhau theo dòng và cột.

Nhân các hạng tổng mới lại, ta có hàm đã tối giản.

Ví dụ 3.1.6: Hãy dùng bảng Các nô để tối giản hàm:

f (A, B, C) = ∑(0, 1, 3, 4, 5)

Lời giải:



+ Xây dựng bảng Các nô tương ứng với hàm đã cho.

Rút gọn theo minterm

+ Gộp các ô có giá trị 1 kế cận lại với nhau thành hai nhóm (bảng 1-6) Lời giải phải tìm: $f(A,B,C) = \bar{B} + \bar{A}C$

Nếu gộp các ô có giá trị 0 lại theo hai nhóm, ta thu được biểu thức hàm bù f:

$$\overline{f(A,B,C)} = AB + B\bar{C}$$

Rút gọn theo Maxterm:

A \ BC	00	01	11	10
0	1	1	1	0
1	1	1	0	0

$\bar{B} + C$

$\bar{A} + \bar{B}$

$$f(A,B,C) = (\bar{A} + \bar{B})(\bar{B} + C) = \bar{B} + \bar{A}C$$

Nếu gộp các ô có giá trị 1 lại theo hai nhóm, ta thu được biểu thức hàm bù f :

$$\overline{f(A,B,C)} = B(A + \bar{C})$$

CD \ AB	00	01	11	10
00				
01	1			1
11	1			1
10				

a) $B.\bar{D}$

CD \ AB	00	01	11	10
00				
01	1	1	1	1
11				
10				

b) $\bar{A}.D$

CD \ AB	00	01	11	10
00	1			1
01				
11				
10	1			1

c) $\bar{B}.\bar{D}$

CD \ AB	00	01	11	10
00	1	1	1	1
01				
11				
10	1	1	1	1

d) $\bar{B}$

CD \ AB	00	01	11	10
00	1	1		
01	1	1		
11	1	1		
10	1	1		

e) $\bar{C}$

CD \ AB	00	01	11	10
00	1	1	1	1
01				
11	1	1	1	1
10				

f) $\bar{A}.\bar{B} + A.B$

Một số vấn đề cần lưu ý khi tiến hành rút gọn bằng bảng Các nô:

- Vòng gộp càng to càng tốt vì số biến được rút gọn càng nhiều.
- Mỗi vòng gộp bao gộp ít nhất một số hạng nhỏ nhất – minterm (hoặc một thừa số lớn nhất - Maxterm) không có trong vòng khác. Vòng nào bao gồm các số hạng đã có trong các vòng khác thì vòng đó là vòng thừa. Tuy nhiên, một số hạng có thể có mặt trong nhiều vòng khác nhau.
- Phải khoanh vòng sao cho toàn bộ số hạng nhỏ nhất - minterm (hoặc một thừa số lớn nhất - Maxterm) của hàm số đều nằm trong các vòng, không được để sót.

Ví dụ 3.1.7: Hãy dùng bảng Các nô để tối giản hàm :

$$f(A,B,C) = \sum(1,4,5,6,8,12,13,15)$$

Lời giải:

Lập bảng Các nô ở bảng 1-9. Ta thấy vòng (m4 + m5 + m12 + m13) là lớn nhất nhưng các vòng khác đều đã chứa m4, m5 , m12 , m13 nên vòng này là vòng thừa.

CD / AB	00	01	11	10
00	0	1	0	0
01	1	1	0	1
11	1	1	1	0
10	1	0	0	0

Sau khi rút gọn ta có biểu thức hàm như sau:

$$f(A,B,C,D) = \bar{A}\bar{C}D + \bar{A}B\bar{D} + A\bar{C}\bar{D} + ABD$$

3.1.4.3. Phương pháp hàm tùy chọn (don't care).

Trên thực tế, tồn tại một số tổ hợp biến có giá trị không ảnh hưởng đến kết quả của hàm. Ví dụ: số BCD là số mã hóa 10 ký hiệu thập phân thành nhị phân 4 bit. Với 4 bit nhị phân ta có thể biểu diễn được mã Hexa, nhưng các ký hiệu A16(1010), B16(1011), C16(1100), D16(1101), E16(1110), E16(1111) lại không phù hợp với mã BCD.

Do vậy, khi lập bảng Các nô không quan tâm đến các giá trị này. Sáu giá trị này được gọi là các trạng thái tùy chọn (*don't care*). Các trạng thái này có thể có giá trị 1 hoặc 0, tùy thuộc vào mục đích người sử dụng và thông thường chúng được ký hiệu bằng chữ " x ".

Khi tiến hành tối thiểu bằng bảng Các nô: tùy theo yêu cầu, có thể tùy ý khoanh vòng qua điều kiện tùy chọn để hàm tối giản hơn.

Từ đó, có thể viết dạng tổng quát của hàm logic như sau:

Dạng chuẩn minterm:

$$f(A,B,C) = \sum m_i + \sum_d m_j; \qquad (1.3)$$

Σ_d là ký hiệu của điều kiện tùy chọn;

Dạng chuẩn Maxterm:

$$F(A,B,C\ \ldots) = \prod M_i \prod_d M_j \qquad (1.4)$$

$\prod_d$ là ký hiệu của điều kiện tùy chọn;

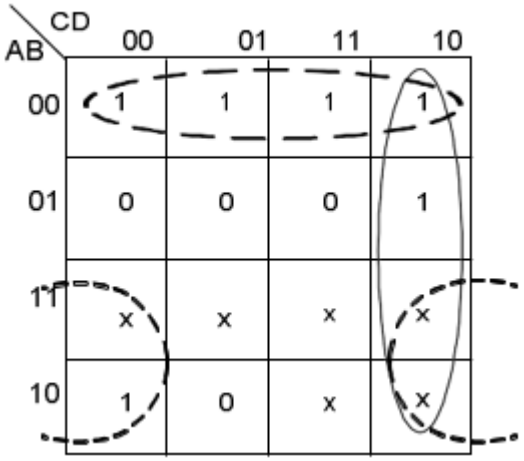
Ví dụ 3.1.8: Tối thiểu hóa hàm

$$F(A,B,C,D) = \sum(0,1,2,3,6,8) + \sum_d(10,11,12,13,14,15)$$

Lập bảng Các nô:

Từ đó, tìm được hàm tối giản sau:

$$F(A,B,C,D) = \bar{A}.\bar{B} + A.\bar{D} + C.\bar{D}$$



3.2. CỔNG VÀ ĐẠI SỐ LOGIC

3.2.1. Cổng (Gate)

Mạch điện tử số là mạch điện trong đó chỉ biểu diễn 2 giá trị logic 0, 1. Thí dụ, tín hiệu điện có điện áp:

0V-1V: biểu diễn một giá trị, chẳng hạn 0 (thấp, mức thấp hoặc low)

2V-5V: biểu diễn giá trị kia, chẳng hạn 1 (cao, mức cao hoặc high).

Các điện áp nằm ngoài miền này là không được phép.

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 hoạt động của chúng.

Khái niệm cổng (Gate): Là những thiết bị điện tử rất nhỏ bé, có một hoặc một số lối vào nhưng chỉ có một lối ra, 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:

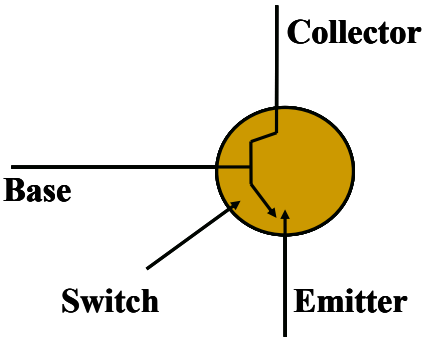
- Mạch logic bởi nó thực hiện các phép tính đại số logic.
- Phần tử ra quyết định (decision making element)

Chính các cổng tạo nên cơ sở phần cứng của tất cả các loại máy tính.

Chi tiết về sự hoạt động bên trong của các cổng không thuộc khuôn khổ của bài giảng này, nó thuộc mức thiết bị - device level (dưới mức 0). 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.

Transistor này có 3 chân nối với thế giới bên ngoài: collector (cực góp), emitter (cực phát) và base (cực gốc).

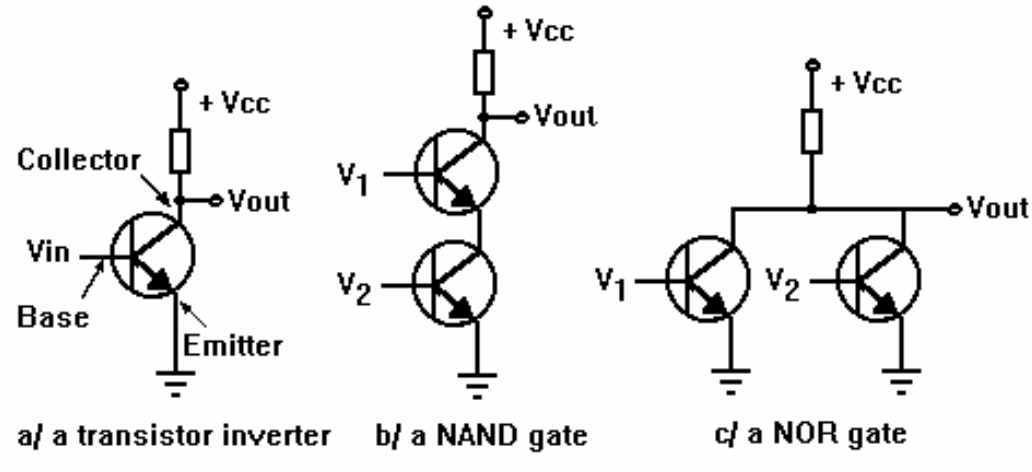


Hình 3-2. Cấu tạo Transistor

Transistor có thể hoạt động theo 2 cơ chế là

- Tuyến tính: được sử dụng trong các bộ khuyếch đại
- Chuyển mạch nhị phân: được sử dụng trong các mạch logic số

Các hình dưới đây mô tả một số cổng cơ bản



Hình 3-3. Một số cổng Logic cơ bản

Hình a mô tả cổng đơn giản nhất - cổng NOT (Inverter); Vout có mức là đảo của Vin.

Hình b mô tả cổng NAND, 2 transitor được chồng nối tiếp lên nhau; Vout có mức thấp (0) khi và chỉ khi cả hai giá trị V1 và V2 là cao (1).

Hình c mô tả cổng NOR, 2 transistor được mắc song song; Vout có mức cao (1) khi và chỉ khi cả hai giá trị V1 và V2 là thấp (0).

Cổng AND: Nếu tín hiệu ra của cổng NAND lại được đưa tới đầu vào của một cổng NOT, 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.

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.

NOT, NAND và NOR là các cổng đơn giản nhất:

- 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.

3.2.2. Đại số logic

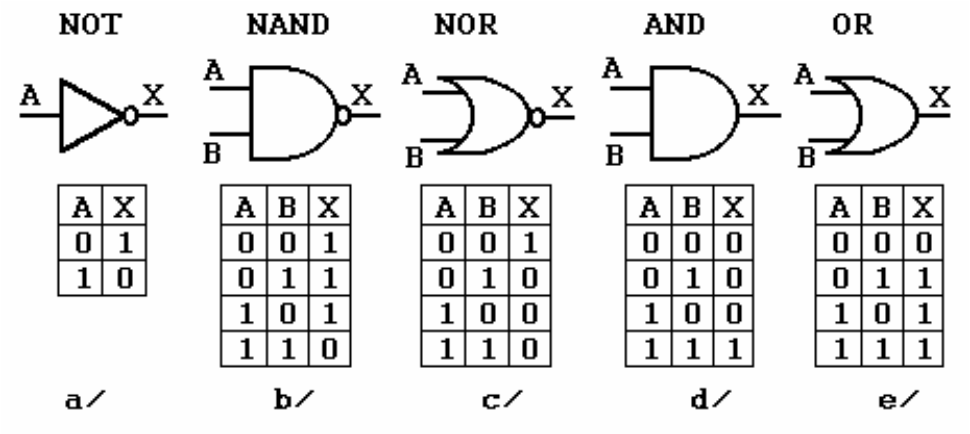
Để mô tả các mạch điện được xây dựng từ các cổng.

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 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.

* Mô tả hàm logic bằng Bảng chân lý (Truth table):

- 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 vẽ sau (Hình 3-4).



Hình 3-4. Mô tả hàm logic bằng bản chân lý

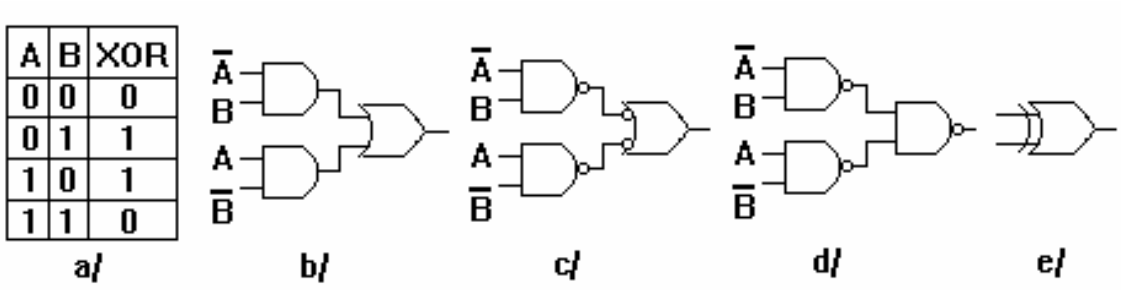
* Mô tả hàm logic bằng số nhị phân:

- Liệt kê các hàng của bảng chân lý theo thứ tự số học (cơ số 2) của tổ hợp biến vào, nghĩa là cho 2 biến theo thứ tự 00, 01, 10 và 11.
- Đọc kết quả (giá trị của hàm) 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.
- 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.
- Trong đại số thông thường, số hàm 2 biến là vô hạn, không thể mô tả bằng bảng các giá trị của hàm theo các giá trị có thể có của 2 biến, bởi vì mỗi biến có thể nhận một giá trị trong một miền vô hạn các giá trị có thể.

3.2.3. Thực hiện các hàm logic

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 hàng mà hàm bằng 1 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.



Hình 3-5. Xây dựng mạch điện bằng hàm logic

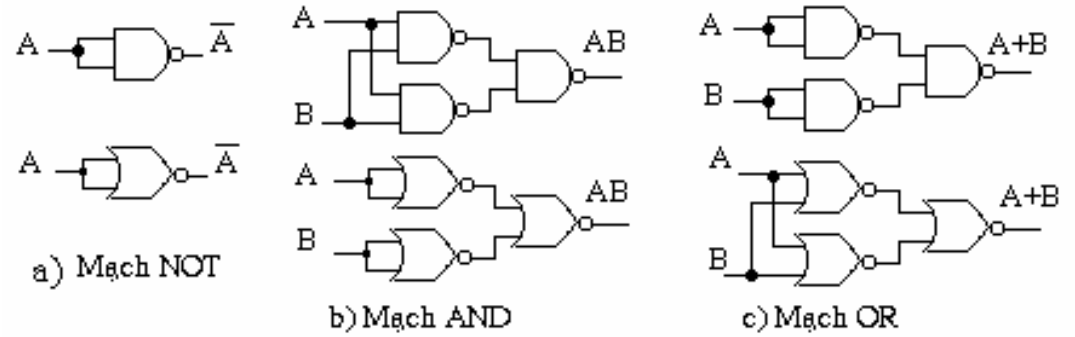
* Sử dụng 1 loại cổng NAND hoặc NOR thường thuận lợi hơn:

- Có thể sử dụng các cổng AND, OR và NOT để xây dựng mọi hàm logic bất kỳ
- Tuy nhiên, nếu giảm được số các loại cổng khác nhau thì tốt hơn:
- Chỉ cần sử dụng một loại cổng NAND hoặc NOR, cũng xây dựng được các cổng AND, OR, NOT. Vì 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. Đồng thời, mạch điện của NAND, NOR đơn giản hơn mạch điện AND, OR
- Do vậy, người ta thường sử dụng hoặc là cổng NAND hoặc là cổng NOR để xây dựng các hàm logic.

3.2.4. Sự tương đương của các mạch

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.

Ví dụ 3.1.9: sử dụng một loại cổng NAND hoặc NOR.



Hình 3-6. Sự tương đương các mạch

* *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

3.3. CÁC MẠCH LOGIC SỐ CƠ BẢN

3.3.1. Mạch tích hợp

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.

- Kích thước chip: rộng cỡ (5-15)mm, dài cỡ (20-50)mm.
- Số chân của một chip: 14, 16, 18, 20 .. 68... Các chip lớn có chân chìa ra ở cả 4 cạnh.

Bảng 3-8. 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

Các chip hiện nay có thể chứa hàng triệu cổng. Tính ra nếu 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), điều này khó có thể chấp nhận được.

Thực tế, trong các chip có nhiều cổng:

- Gồm nhiều đơn vị chức năng, các cổng thuộc mỗi đơn vị chức năng liên hệ với nhau thông qua các đường dây dẫn ngay bên trong chip; đồng thời, các đơn vị chức năng cũng có thể liên hệ với nhau bằng các đường dây dẫn bên trong chip. Do đó, có thể chỉ cần một số ít, thậm chí không cần các chân đưa ra ngoài.
- Một số chân chip có thể được 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.

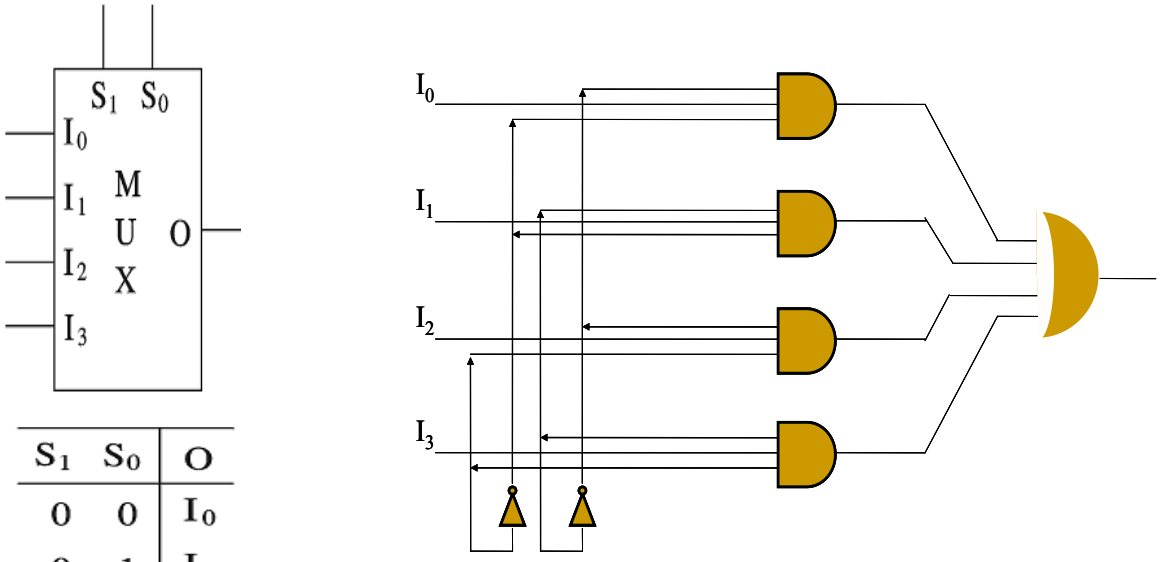
3.3.2. Mạch tổ hợp

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.

Một số mạch tổ hợp mạch tổ hợp điển hình như: mạch dồn kênh, mạch phân kênh, mạch mã hoá, mạch giải mã, mạch so sánh, v.v..

3.3.2.1. Mạch dồn kênh (Multiplexer)

Là một mạch điện với 2n lối vào số liệu, một lối ra số liệu, và n lối vào điều khiển. Một trong số 2n đầu vào số liệu sẽ được chọn để cho đi qua cổng.



S_1	S_0	O
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

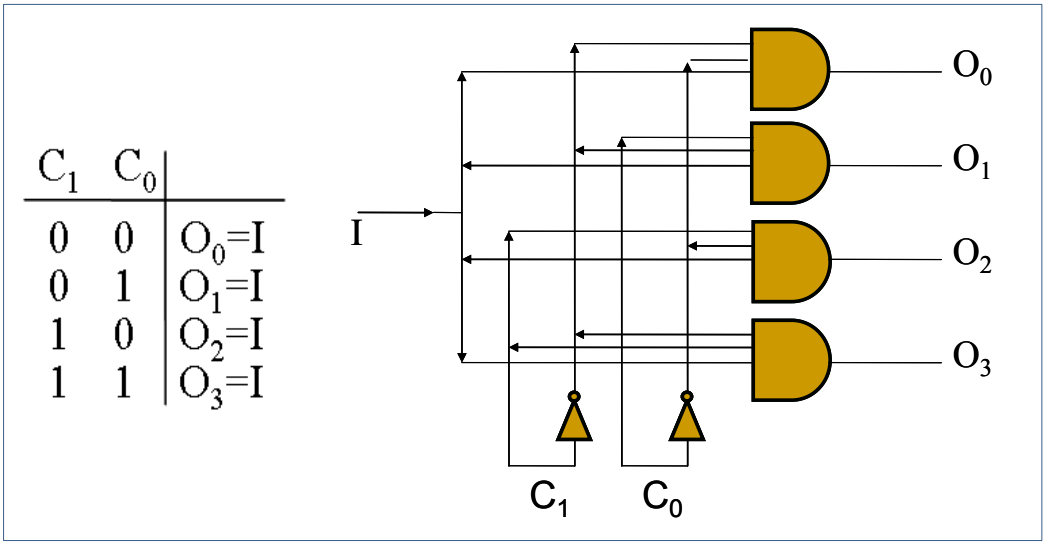
Hình 3-7. Mạch dồn kênh cho 4 đường dữ liệu vào

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.

Một số mạch tổ hợp mạch tổ hợp điển hình như: mạch dồn kênh, mạch phân kênh, mạch mã hoá, mạch giải mã, mạch so sánh, v.v..

3.3.2.2. Mạch phân kênh (Demultiplexe)

Ngược lại với mạch dồn kênh: Có một lối vào số liệu, lối ra là 1 trong 2n lối ra, tuỳ thuộc tín hiệu trên n đầu vào điều khiển



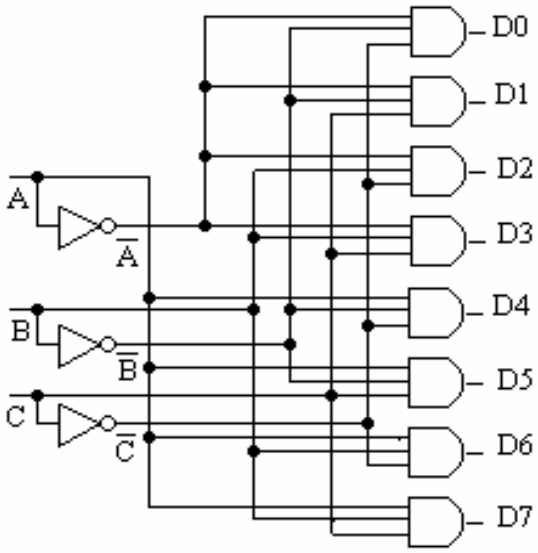
Hình 3-8. Mạch phân kênh 1 đầu vào 4 đầu ra

3.3.2.3. Mạch giải mã (decoder)

Đó là mạch điện nhận con số n-bit đầu vào để chọn (1) cho duy nhất một trong số 2n đầu ra có mức 1.

Ứng dụng của mạch giải mã:

- Giải mã bộ nhớ: khi bộ nhớ được tổ chức thành ma trận nhớ, sử dụng bộ giải mã để chọn 1 trong các hàng và 1 trong các cột
- Chọn 1 trong nhiều chip nhớ
- v.v.

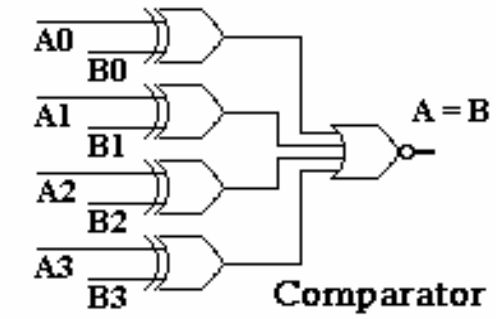


Hình 3-9. Mạch giải mã 3 đầu

3.3.2.4. Mạch so sánh (Comparator)

Thực hiện so sánh 2 từ nhị phân đưa vào bằng cách so sánh các cặp bit tương ứng. Thí dụ trên hình là một bộ so sánh 2 toán hạng nhị phân 4 bit:

- $A = A_3\ A_2\ A_1\ A_0$
- $B = B_3\ B_2\ B_1\ B_0$
- 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 3-10. Mạch so sánh (Comparator)

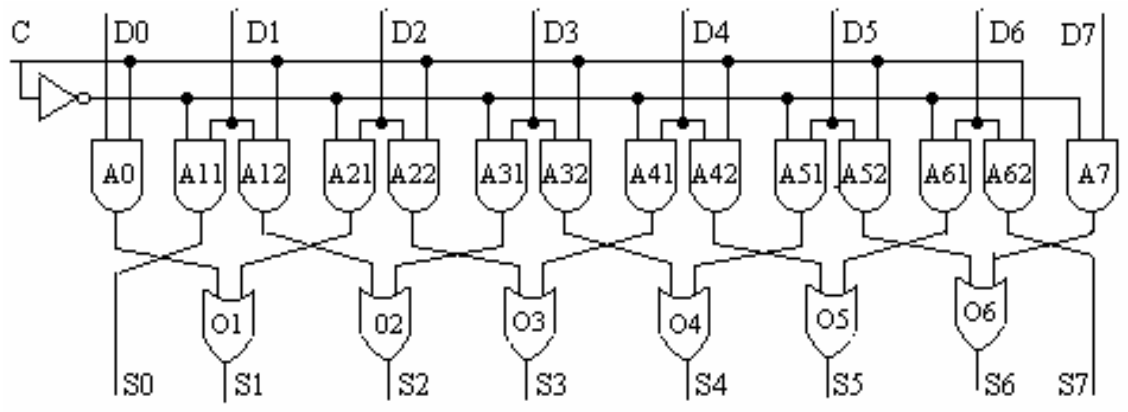
3.3.3. Các mạch số học

3.3.3.1. Bộ dịch (Shifter)

Dịch bit là một thao tác cơ sở trong hoạt động của máy tính.

Trên hình 3-8 là bộ dịch bit 8 bit:

- 8 đầu vào: D0 .. D7; 8 đầu ra: S0 .. S7
- C xác định hướng dịch chuyển của các bit, nếu C=1: dịch phải một vị trí, nếu C=0: dịch trái một vị trí.



Hình 3-11. Mạch số học bộ dịch 8bit

C xác định hướng dịch chuyển của các bit:

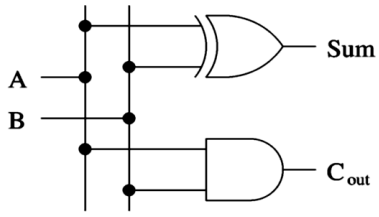
C=1: S0=0, S1=D0, S2=D1, S3=D2, ... S7=D6: dịch phải một vị trí

C=0: S7=0, S6=D7, S5=D6, S4=D5, ... S0=D1: dịch trái một vị trí.

3.3.3.2. Bộ cộng

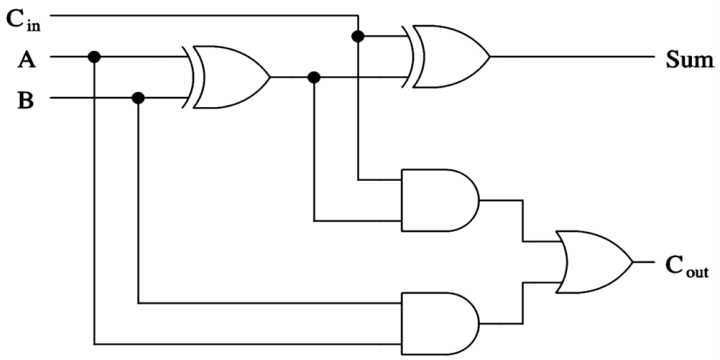
Bộ cộng là một phần rất căn bản của mọi CPU. Hình 3.10 a là mạch tính tổng 2 bit - Haft_Adder. Bộ cộng này không có lối vào cho số nhớ Carry-in cho nên không dùng để cộng các bit bậc cao hơn 0. Hình 3.10 b là mạch tính tổng 2 bit vào và bit nhớ C-in, trả lại tổng - sum và số nhớ C-out**.** Bộ cộng này có lối vào C-in, nó được xây dựng từ 2 bộ Haft-Adder.

A	B	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(a) Half-adder truth table and implementation

A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



(b) Full-adder truth table and implementation

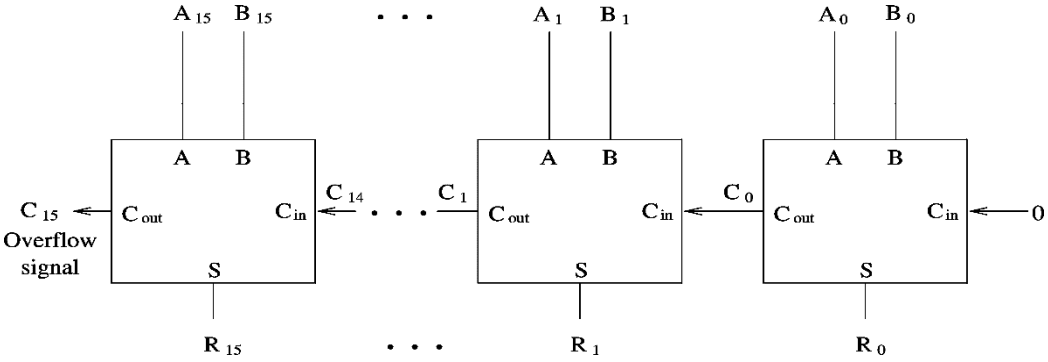
Hình 3-12. Mạch bộ cộng bán phần và toàn phần

Để xây dựng các bộ cộng với từ dài hơn, chẳng hạn từ 16 bit, cần sử dụng 16 bộ Full_Adder:

- 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...), việc lan truyền bit nhớ làm chậm phép tính.

Người ta cũng xây dựng các bộ cộng không có nhược điểm này nhưng chúng phức tạp hơn. Hình dưới đây mô tả một 16-bit ripple-carry adder

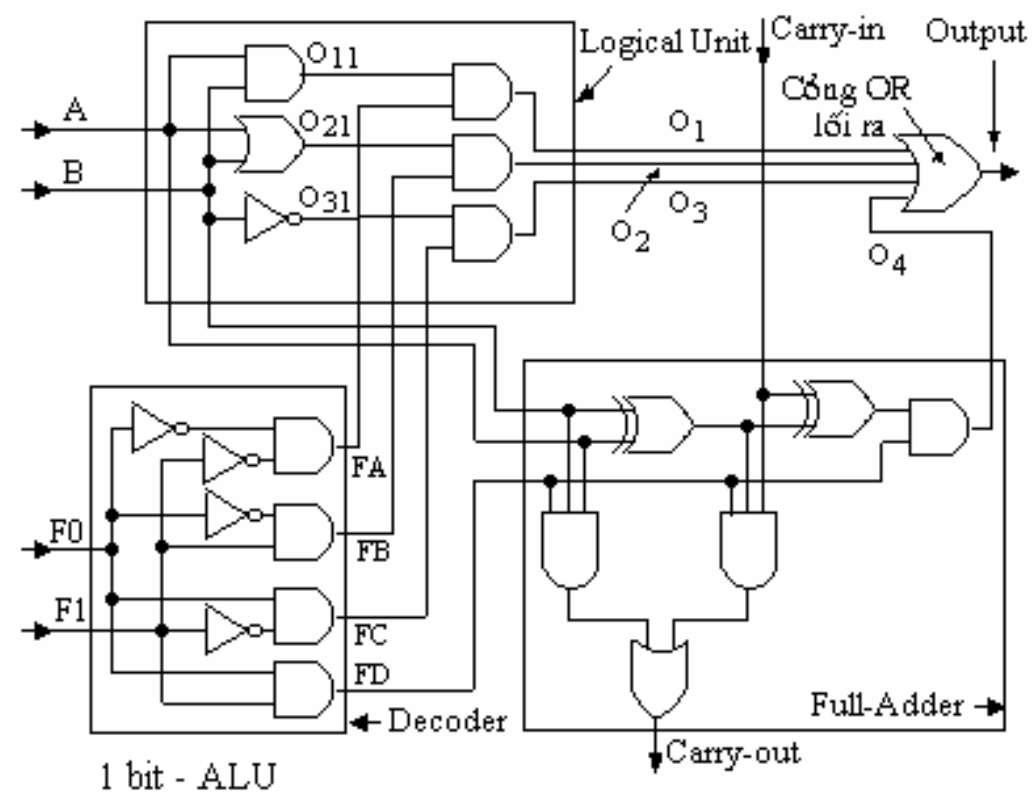


Hình 3-13. Xây dựng mạch bộ cộng 16-bit ripple-carry adder

3.3.3.3. Bộ tính toán số học và logic – ALU (Arithmetic Logical Unit)

Hầu hết các bộ vi xử lý đều có mạch riêng (ALU) để thực hiện các phép tính AND, OR và tính tổng của 2 từ máy. Hình 3.12 là một ALU đơn giản, thực hiện 1 trong 4 chức năng sau tuỳ theo tín hiệu điều khiển chọn chức năng F0 và F1:

- F0F1=00: A AND B
- F0F1=01: A OR B
- F0F1=10: NOT B (đảo B)
- F0F1=11: A + B



Hình 3-14. Cấu tạo bộ tính toán và logic số học ALU

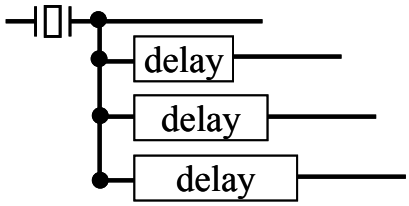
3.3.3.4. Clock - Bộ tạo tín hiệu thời gian

Trong rất nhiều mạch điện số, trật tự xảy ra các sự kiện là hết sức quan trọng. Cần thiết phải sử dụng đồng hồ để cung cấp tín hiệu đồng bộ các quá trình:

- Đồng hồ là một mạch điện phát ra chuỗi xung điện có chu kỳ rất ổn định và chính xác.
- Tần số xung nằm trong khoảng từ 1..100MHz, tương ứng có chu kỳ 1ms..10ns.
- Thường sử dụng máy phát thạch anh (Crystal Oscilator).

Chu kỳ con (subcycles):

Trong máy tính, nhiều sự kiện có thể xảy ra trong một chu kỳ đồng hồ, nếu chúng phải xảy ra theo một trật tự nhất định thì chu kỳ đồng hồ phải được chia ra thành các chu kỳ con. Giải pháp đưa ra là cần 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.



Hình 3-15. Bộ tạo tín hiệu thời gian

Các mạch của máy tính được điều khiển bởi một mạch tạo các chuỗi xung tuần hoàn gọi là xung Clock. Xung Clock xác định các chu kỳ của máy tính.

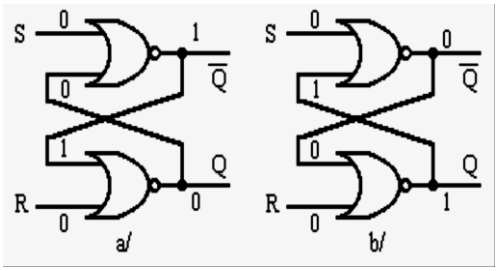
3.3.4. Mạch Thanh ghi chốt

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.

3.3.4.1. Thanh ghi chốt RS

Thanh ghi chốt RS là một dạng thanh ghi đơn giản nhất và được xây dựng từ các cổng NOR. Trong đó,

- 2 lối vào: S (Set) - để thiết lập giá trị (cho Q=1) và R (Reset) - để xoá nó (cho Q=0)
- 2 lối ra luôn bù nhau là Q và $\overline{Q}$



Hình 3-16. Mạch thanh ghi chốt RS

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.

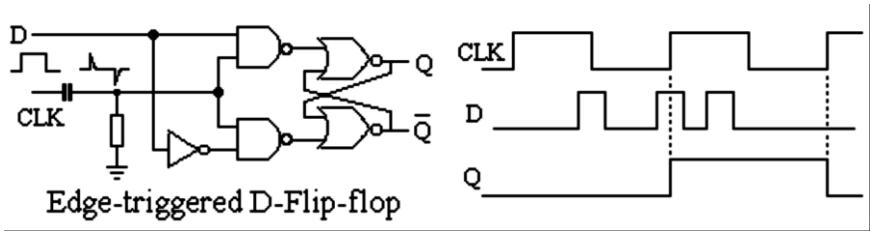
- R=S=0: Q sẽ không thay đổi. (trạng thái này tồn tại trong phần lớn thời gian hoạt động của thanh ghi)
- S=1, R=0, Q=1 không phụ thuộc vào trạng thái trước đó.
- R=1, S=0, Q=0 không phụ thuộc vào trạng thái trước đó.
- Tổ hợp R=S=1 bị cấm.

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.

3.3.4.2. Mạch Flip-Flop

Trong nhiều mạch điện thường cần phải lấy mẫu (sample) giá trị tín hiệu trên một đường dây nào đó tại một thời điểm cụ thể và ghi nhớ giá trị đó.

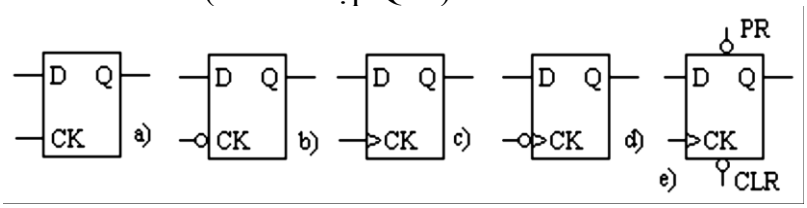
Flip-flop là một biến thể của thanh ghi chốt D, có khả năng trong khoảng thời gian ứng với xung đồng hồ rất ngắn trên lối vào clock, ghi nhận được giá trị ở lối vào D.Như vậy thời gian kéo dài xung không quan trọng, chỉ cần sự chuyển trạng thái xảy ra đủ nhanh



Hình 3-17. Mạch Flip - Flop

** 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 c) và d) là các Flip-flop, ở lối vào clock của chúng được vẽ ký hiệu đầu mũi tên '>'.
    - Flip-flop trên hình (c) chuyển 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 và có thêm các đầu vào Set hoặc Preset (để thiết lập Q=1) và Reset hay Clear (để thiết lập Q=0).



3.3.4.3. Thanh ghi

Thanh ghi là một nhóm các phần tử nhớ cơ bản cùng hoạt động như một đơn vị. Có các loại thanh ghi thực hiện các nhiệm vụ khác nhau: nhớ, tính toán số học - dịch trái, dịch phải hoặc các thao tác khác phức tạp hơn nữa.

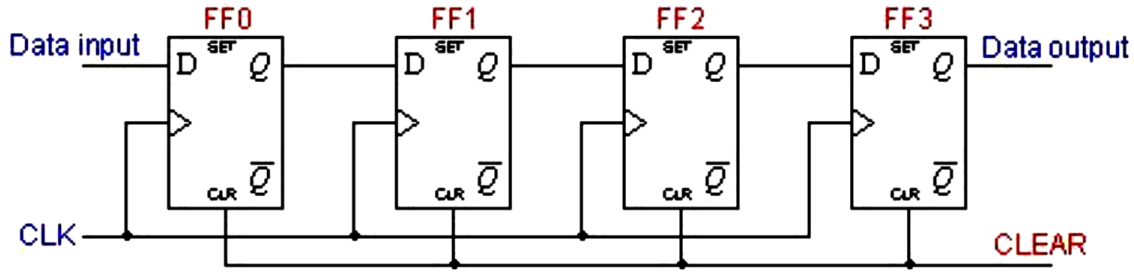
Các thanh ghi làm nhiệm vụ nhớ thường được xây dựng từ các flip-flop, chúng cần có khả năng hoạt động ở tốc độ cao hơn các thanh ghi được sử dụng trong bộ nhớ chính.

Do mạch Flip-Flop có thể lưu trữ 1bit và khi có xung kích hoạt (Ck) thì bit đó mới được truyền tới đầu ra (đảo hoặc không đảo). Khi cần nhớ nhiều bit ta chỉ cần mắc nối tiếp nhiều Flip-Flop lại với nhau. Nếu mạch có khả năng ghi lại dữ liệu và dịch chuyển nó thì gọi là thanh ghi dịch.

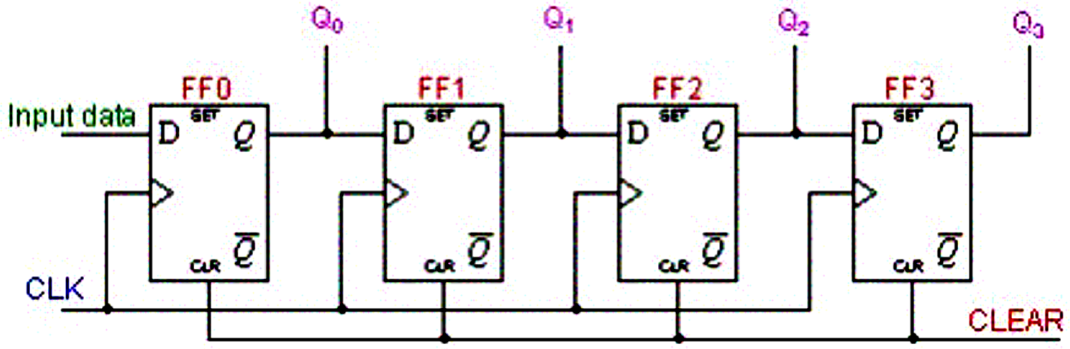
Có một số dạng kết nối thanh ghi dịch như sau:

SISO (Serial In Serial Output - vào nối tiếp ra nối tiếp)
SIPO (Serial In Parallel Output - vào nối tiếp ra song song)
PISO (Parallel In Serial Output - vào song song ra nối tiếp)
PIPO (Parallel In Parallel Output - vào song song ra song song)

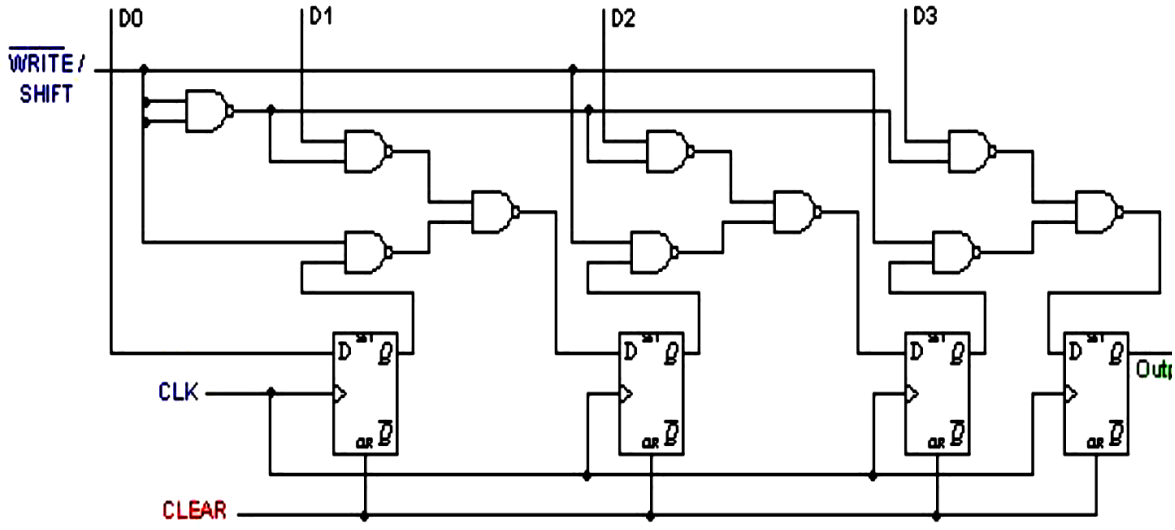
Cấu trúc các dạng kết nối được mô tả trong hình (a), (b), (c), (d) dưới đây:



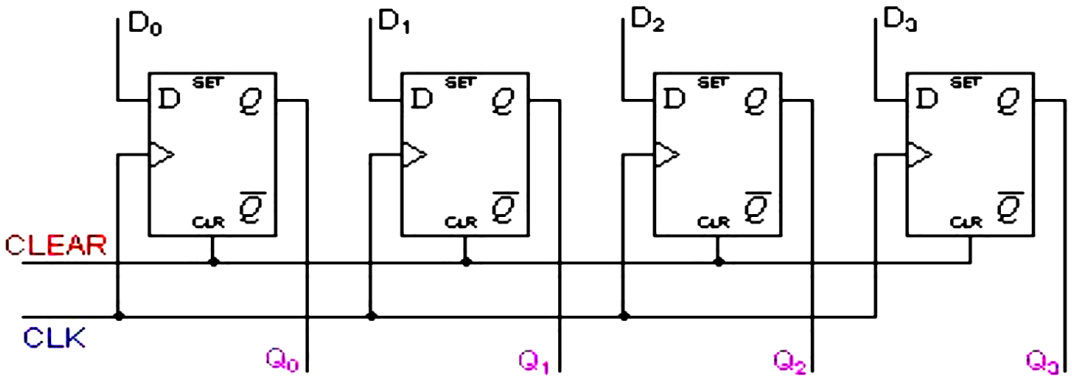
(a) Kết nối dạng SISO: Dữ liệu cần dịch chuyển được đưa vào ngõ D của mạch FF đầu tiên (FF0). Ở mỗi xung kích lên của đồng hồ ck, sẽ có 1 bit được dịch chuyển từ trái sang phải, nối tiếp từ tầng này qua tầng khác và đưa ra ở đầu Q của mạch sau cùng (FF3)



(b) Kết nối dạng SIPO: Dữ liệu sẽ được lấy ra ở 4 đầu ra Q của mạch FF, vì chung nhịp đồng hồ nên dữ liệu cũng được lấy ra cùng lúc



(c) Kết nối dạng PISO: đầu ra dữ liệu là nối tiếp (tại đầu ra Q hoặc $\overline{Q}$), còn dữ liệu đầu vào là song song



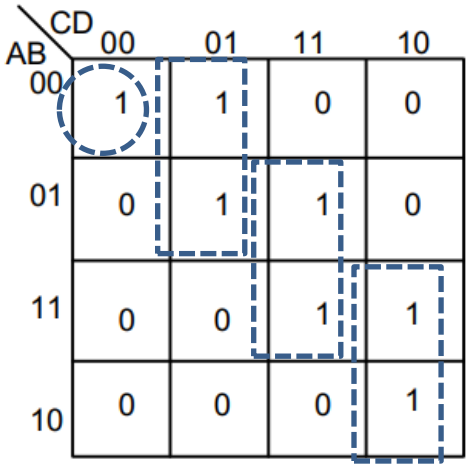
(d) Kết nối dạng PIPO: dữ liệu được đưa vào cùng một lúc và lấy ra cùng một lúc

Hình 0-18. Có một số dạng kết nối thanh ghi dịch

3.3.5. Một số ví dụ cơ bản

Ví dụ 3.3.1. Xây dựng bảng Các nô cho hàm logic sau:

F(A, B, C, D) = Σ(0, 1, 5, 7, 10, 14, 15):



$$F(A, B, C, D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{C}D + BCD + AC\bar{D}$$

Ví dụ 3.3.2. Hãy đưa hàm logic về dạng tối giản:

f = AB + $\bar{A}$C + BC

Lời giải: Áp dụng định lý, A + $\bar{A}$ = 1, A + A.B =A

Ta có:

f = AB + $\bar{A}$C + BC (A + $\bar{A}$)

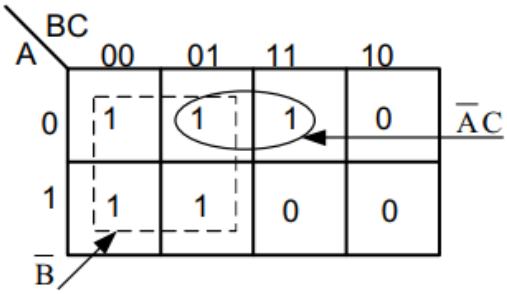
= AB + ABC + $\bar{A}$C + $\bar{A}$BC

= AB + $\bar{A}$C

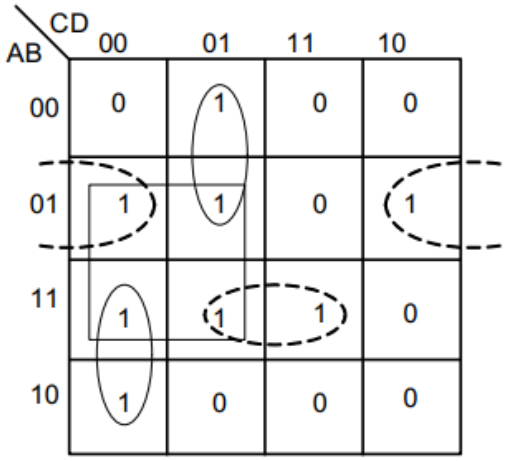
Ví dụ 3.3.3. Hãy dùng bảng Các nô để tối giản hàm:

f (A,B,C) = Σ(0, 1, 3, 4, 5)

Lời giải:



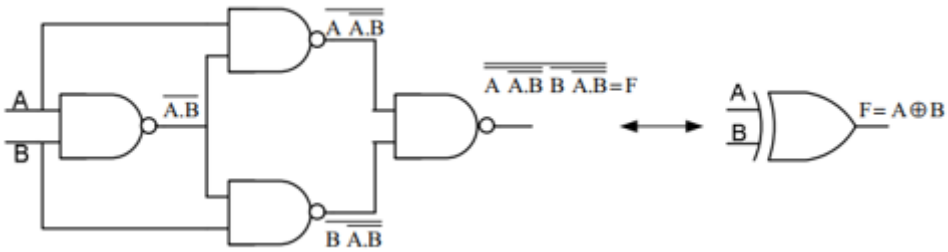
Ví dụ 3.3.4. Hãy dùng bảng Các nô để tối giản hàm:
f (A,B,C) = ∑(1, 4, 5,6,8,12,13,15)
Lời giải:



Sau khi rút gọn ta có biểu thức hàm như sau:
f (A,B,C,D) = $\overline{A}\overline{C}D + \overline{A}B\overline{D} + A\overline{C}\overline{D} + ABD$
Ví dụ 3.3.5. Cho hàm logic $F = A\overline{B} + \overline{A}B$, hãy xây mạch về dạng toán NAND
Giải:

$$F = A\overline{B} + \overline{A}B = A\overline{B} + A\overline{A} + \overline{A}B + B\overline{B} = A(\overline{A} + \overline{B}) + B(\overline{A} + \overline{B})$$
$$= A\overline{AB} + B\overline{AB} = \overline{\overline{A\overline{AB}} + \overline{B\overline{AB}}} = \overline{\overline{A\overline{AB}}.\overline{B\overline{AB}}}$$

Từ biểu thức biến đổi trên vẽ được sơ đồ logic

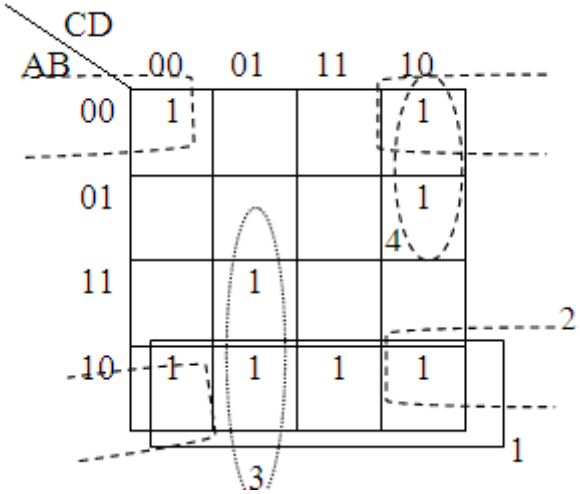


Ví dụ 3.3.6. Dùng bản đồ Karnaugh rút gọn hàm $f(A,B,C,D) = \sum(0,2,6,8,9,10,11,13)$ và vẽ sơ đồ mạch của hàm f dùng các cổng AND và OR.

Trả lời:
$f(A,B,C,D) = \sum(0,2,6,8,9,10,11,13)$

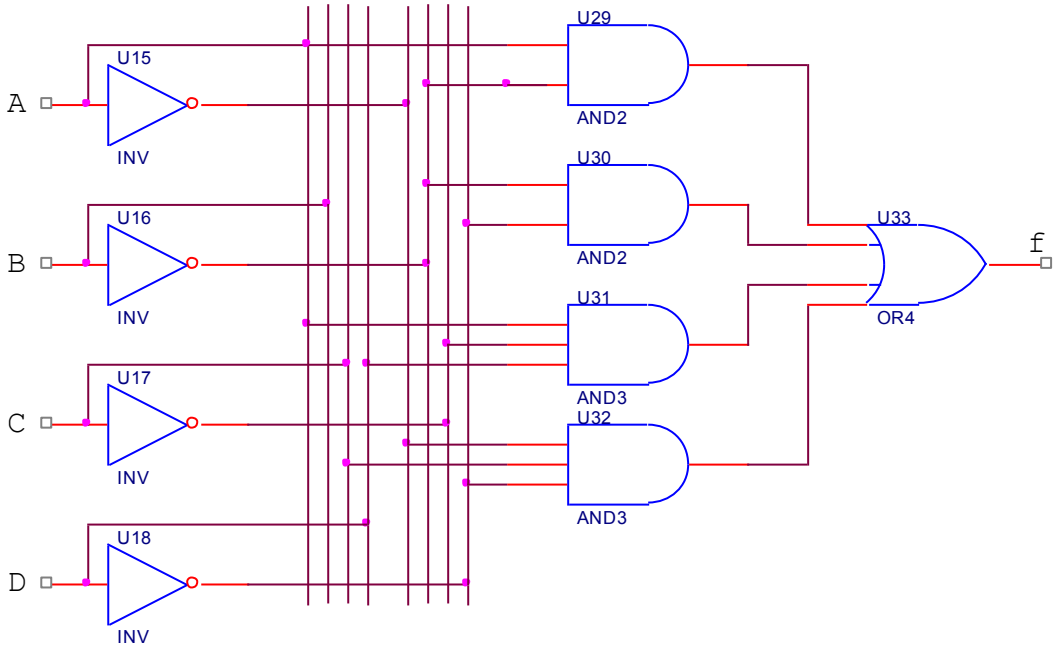
AB \ CD	00	01	11	10
00	1			1
01				1
11		1		
10	1	1	1	1

Sau khi nhóm:



Kết quả hàm rút gọn: $f(A,B,C,D) = A\overline{B} + \overline{B}\overline{D} + A\overline{C}D + \overline{A}C\overline{D}$

Sơ đồ mạch:



Ví dụ 3.3.7. Cho hàm số: F(ABCD) = S (3,5,7,11,13,15)

a) Viết biểu thức đại số đầy đủ cho hàm

b)Viết biểu thức dạng tối thiểu hóa cho hàm

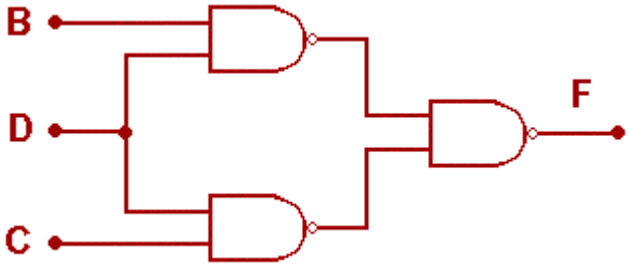
c) Vẽ sơ đồ logic cho hàm dùng cổng NAND 2 đầu vào
d) Vẽ sơ đồ logic cho hàm dùng cổng NOR 2 đầu vào

Trả lời:

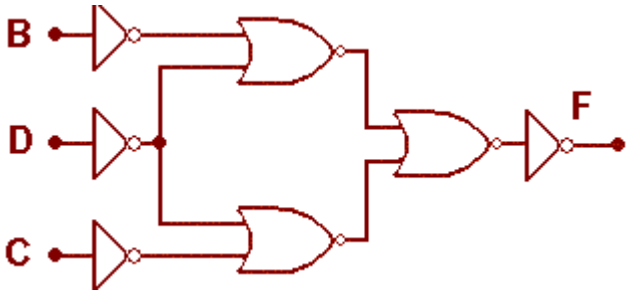
a) $F(ABCD) = \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}CD$
$= AB\overline{C}D + ABCD$

b) F(ABCD) = BD + CD

c) Sơ đồ cổng NAND



d) Sơ đồ cổng NOR



CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 3

Câu 1. Xây dựng sơ đồ và đánh giá OUTPUTs của mạch logic:

a) $\overline{A.B} + \left(\overline{C.D + E.\bar{F}}\right)$

b) $(A + B).(\overline{A.B.C}).(A + C)$

c) $(A + \bar{A}).(A.\bar{B} + \bar{A}\,B).(\overline{A.B.C})$

d) $(A.\bar{A}).(\bar{B} + B).(A.B.C)$

e) $(A.C).(\overline{A.B}).(\overline{A.B} + B.C)$

f) $(A.\bar{E}).(\overline{D + C.B}) + (\bar{A}.\bar{B} + B.\bar{D})$

g) $(A + A.B).\left(\overline{E.F + \overline{C.D}}\right).(\overline{A.B.C.D})$

h) $C.\bar{D} + \left(\overline{\bar{A}.B + \bar{B}.A}\right).(A.B.C.D)$

i) $(E.F + \overline{A.B}).(\overline{E.F} + A.B).(\overline{C + D})$

j) $(A.B.C + \bar{A}.\bar{B}.\bar{C}).\left(\overline{A.C + \bar{C}.B}\right)$

k) $\left(\overline{A.F} + \overline{\bar{B}.C} + \bar{C}.\bar{D}\right).(\overline{A.B.C.D})$

l) $\left(\overline{\bar{A} + \bar{B}}\right).(A.B + \overline{A.B}).(\overline{E.F})$

m) $\overline{(\bar{A}.C + A.\bar{B}).(\overline{E.F} + \overline{C.D})}$

n) $(\overline{A + B}).(A.B + \overline{A.B}).(\bar{A}.\bar{B})$

o) $\overline{(\bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C})}$

p) $(A.\bar{F} + \bar{A}.F + A.F + \overline{A.F}).\left(\overline{B.C + \overline{D.E}}\right)$

q) $(A.C + \bar{A}.B.\bar{C}).\left(\overline{\bar{B}.D + \bar{D}.B}\right)$

r) $\overline{(A + C).(B + D)}.(A.\bar{B}.C + A.B.\bar{C})$

s) $(\overline{A.B} + A.B.C).\left(\overline{E.F + A.\overline{E.F}}\right)$

t) $(A.\overline{B.C} + B.E).(A.\bar{C}.F + E.C)$

u) $\overline{AB + \bar{A}\bar{B}} = \bar{A}B + A\bar{B}$

v) $AB + \bar{A}C = (A + C)(\bar{A} + B)$

w) $A\overline{C + B\bar{C}} = \bar{A}C + \bar{B}\bar{C}$

Câu 2. Cho hàm logic dạng tuyển sau:

$$Z = F(A, B, C) = \sum(1,2,3,5,7)$$

Hãy tối giản hóa bằng phương pháp đại số.

Câu 3. Cho hàm logic dạng hội sau:

$$Z = F(A, B, C) = \prod(0,4,6)$$

Hãy tối giản hóa bằng phương pháp đại số.

Câu 4. Cho hàm logic dạng tuyển sau:

$$Z = F(A,B,C,D) = \sum(1,2,4,5,6,8,9,10,14)$$

Xây dựng sơ đồ mạch hàm Z và đánh giá OUTPUTs của mạch logic trên.

Câu 5. Cho hàm logic dạng hội sau:

$$M = F(A,B,C,D) = \prod(1,2,4,5,6,8,9,10,14)$$

Xây dựng sơ đồ mạch hàm M và đánh giá OUTPUTs của mạch logic trên.

Câu 6. Cho hàm số:

$$Y = F(A,B,C,D) = \prod(0,1,3,7,8,9,11,12,13,15)$$

Xây dựng sơ đồ mạch hàm Y và đánh giá OUTPUTs của mạch logic trên.

Câu 7: Cho hàm logic dạng tuyển sau:

$$Z = F(A,B,C) = \sum(1,2,3,5,7)$$

Thiết kế mạch logic trên

Câu 8: Chứng minh các biểu thức sau:

a) $\overline{AB + \bar{A}\bar{B}} = \bar{A}B + A\bar{B}$

b) $AB + \bar{A}C = (A + C)(\bar{A} + B)$

c) $A\overline{C + B\bar{C}} = \bar{A}C + \bar{B}\bar{C}$

Câu 9. Rút gọn các hàm sau dùng các định lý của Boolean algebra:

a) $x = ACD + \bar{A}BCD$

b) $y = AB + A(CD + C\bar{D})$

c) $z = (B\bar{C} + \bar{A}D)(A\bar{B} + C\bar{D})$

Câu 10. Tối thiểu hóa các hàm sau bằng phương pháp đại số:

$F(A,B,C,D) = (A + BC) + \bar{A}(\bar{B} + \bar{C})(AD + C)$.

Câu 11. Tối thiểu hóa các hàm sau bằng phương pháp đại số:

$$F(A,B,C,D) = (A + B + C)(A + B + \bar{C})(\bar{A} + B + C)(\bar{A} + B + \bar{C})$$

Câu 12. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \sum(0,2,5,6,9,11,13,14)$$

Câu 13. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \sum(0,1,3,5,8,9,13,14,15)$$

Câu 14. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \sum(2,4,5,6,7,9,12,13)$$

Câu 15. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \prod(1,4,6,7,9,10,12,13)$$

Câu 16. Tối thiểu hóa các hàm sau bằng bìa Các-nô*:

$$F(A,B,C,D,E) = \sum(0,1,9,11,13,15,16,17,20,21,25,26,27,30,31)$$

Câu 17. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \prod(0,2,3,4,6,7,9,12,13)$$

Câu 18. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \prod(0,2,8,9,10,11,13,14)$$

Câu 19. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \sum(0,2,6,8,9,10,11,13)$$

Câu 20. Tối thiểu hóa các hàm sau bằng bìa Các-nô:

$$F(A,B,C,D) = \sum(0,1,2,3,4,6,7,8,9,10,11,13)$$

Câu 21. Tối thiểu hóa các hàm sau bằng bìa Các-nô.

$$F(A,B,C,D) = \sum(1,3,4,5,7,9,11,14) + d(6,12,13)$$

Câu 22. Cho hàm bool dùng bản đồ Các_nô để:

$$F(A,B,C,D) = \sum(0,1,6,8,9,11,14,15) + d(2,3,10)$$

a. Xác định dạng chuẩn tổng các tích của hàm f (gọi là hàm g)
b. Xác định dạng chuẩn tích các tổng của hàm f (gọi là hàm h)
c. So sánh hai hàm g và h
d. Xây dựng sơ đồ hàm g và đánh giá OUTPUTs của mạch logic trên.

Câu 23. Cho hàm bool dùng bản đồ Các_nô để:

$$F(A,B,C,D) = \sum(3,4,5,7,10,12,13) + d(8,9,11)$$

a. Xác định dạng chuẩn tổng các tích của hàm f (gọi là hàm g)
b. Xác định dạng chuẩn tích các tổng của hàm f (gọi là hàm h)
c. So sánh hai hàm g và h
d. Xây dựng sơ đồ hàm g và đánh giá OUTPUTs của mạch logic trên.

Câu 24. Cho hàm bool f(A, B, C, D) = ∏(3, 4, 5, 6, 10, 12, 13) + d(8, 11), Dùng bản đồ Karnaugh để rút gọn theo :

a. Dạng tổng các tích của hàm f
b. Dạng tích các tổng của hàm f
c. Vẽ sơ đồ mạch cho câu a và b

Câu 25. Cho hàm bool f(A, B, C, D) = $(\overline{A}+\overline{B}+C+D)(\overline{A}+B+\overline{C}+\overline{D})(\overline{A}+B+\overline{C}+D)(\overline{A}+B+C+D)(\overline{A}+B+\overline{C}+D)$,

a. Đơn giản hàm f.
b. Vẽ sơ đồ mạch hàm f mà chỉ sử dụng cổng NAND.

Câu 26. Cho hàm bool f(A, B, C, D) = ∏(3, 4, 5, 7, 10, 12, 13) + D(8, 9, 11), Dùng bản đồ Karnaugh để :

c. Xác định dạng chuẩn tổng các tích của hàm f (gọi là hàm g).
d. Xác định dạng chuẩn tích các tổng của hàm f (gọi là hàm h).
e. So sánh hai hàm g và h.
f. Vẽ sơ đồ mạch hàm g mà chỉ sử dụng cổng NOR 2 ngõ vào.

Câu 27. Vẽ và phân tích hoạt động của mạch dồn kênh (Multiplexer)? Lấy ví dụ minh họa?

Câu 28. Vẽ và phân tích hoạt động của mạch phân kênh (Demultiplexer)? Lấy ví dụ minh họa?

Câu 29 Vẽ và phân tích hoạt động của mạch giải mã (Decoder)? Lấy ví dụ minh họa?

Câu 30. Vẽ và phân tích hoạt động của mạch mạch so sánh (Comparator)? Lấy ví dụ minh họa?

Câu 31. Vẽ và phân tích hoạt động của mạch dịch (Shifter)? Lấy ví dụ minh họa?

Câu 32. Vẽ và phân tích hoạt động của mạch cộng bán phần (Half adder)? Lấy ví dụ minh họa?

Câu 33. Vẽ và phân tích hoạt động của mạch cộng toàn phần (Full adder)? Lấy ví dụ minh họa?

Chương 4
BỘ XỬ LÝ TRUNG TÂM CPU

Cấu trúc, chức năng của bộ xử lý trung tâm CPU. Giúp sinh viên nắm vững kiến thức về các thanh ghi, đơn vị số học và logic, đơn vị điều khiển nằm bên trong bộ xử lý.

Bộ điều khiển tạo ra sự vận chuyển tín hiệu bên trong bộ xử lý nhằm thực hiện tập lệnh tương ứng với kiến trúc phần mềm đã đề ra. Mô tả diễn tiến thi hành một lệnh mã máy, đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong máy tính.

Cấu trúc của bộ xử lý trung tâm và diễn tiến thi hành một lệnh mã máy, vì đây là cơ sở để hiểu được các hoạt động xử lý lệnh trong các kỹ thuật xử lý thông tin trong máy tính.

Ngôn ngữ lập trình Assebly bằng phần mềm Emu8086

4.1. BỘ XỬ LÝ TRUNG TÂM

Trang bị cho sinh viên kiến thức về cấu trúc, chức năng của bộ xử lý trung tâm CPU. Giúp sinh viên nắm vững kiến thức về các thanh ghi, đơn vị số học và logic, đơn vị điều khiển nằm bên trong bộ xử lý.

4.1.1. Cấu trúc, chức năng của bộ xử lý

4.1.1.1. Chức năng của bộ xử lý

Bộ xử lý trung tâm CPU là cốt lõi của một máy vi tính. CPU thực hiện mọi tính toán và xử lý của hệ thống (ngoại trừ xử lý tăng cường tính toán đặc biệt trong những hệ thống có một chip đơn vị đồng xử lý toán, mà chip này cũng đã được tích hợp ngay trong các CPU hiện nay).

Một trong những bộ xử lý điển hình thuộc họ 80x86 của Intel là bộ xử lý 8088. Đây là bộ xử lý khá đơn giản. Vì vậy việc tìm hiểu nó là tương đối dễ đối với những người bắt đầu thâm nhập vào lĩnh vực vi xử lý, mặt khác việc nắm vững các vấn đề kỹ thuật của bộ vi xử lý 8088 sẽ là cơ sở để nắm bắt được các kỹ thuật của các bộ xử lý khác trong họ 80x86 của Intel, của các họ khác và của các bộ xử lý hiện đại ngày nay.

4.1.1.2. Cấu trúc của bộ vi xử lý

CPU có 3 bộ phận chính:

- Khối điều khiển (CU): Nhận lệnh của chương trình từ bộ nhớ trong đưa vào CPU. Nó có nhiệm vụ giải mã các lệnh, tạo ra các tín hiệu điều khiển công việc của các bộ phận khác của máy tính theo yêu cầu người sử dụng.

- Khối tính toán số học và logic (ALU): Bao gồm các thiết bị thực hiện các phép toán số học, phép tính logic và các tính quan hệ. Độ dài của các toán hạng được đưa vào tính toán trực tiếp ở khối ALU. Độ dài phổ biến với các máy tính hiện nay là 32 hay 64 bit.

- Tập các thanh ghi: Các thanh ghi mang chức năng chuyên dụng tăng tốc độ trao đổi thông tin trong máy tính.

4.1.2. Các thanh ghi

4.1.2.1. Các thanh ghi đa năng (general registers)

Có nhiệm vụ ghi tham số cho mã lệnh, đây cũng là nơi lệnh trả kết quả về sau khi được thực hiện. Những thanh ghi đa năng của vi xử lý 16 bit là [6,7]:

- *AX (accumulator)* rộng 16 bit, được chia làm hai phần: 1 byte cao AH và 1 byte thấp AL. Đây là thanh ghi quan trọng nhất và chuyên được dùng để chứa kết quả các thao tác lệnh. Cả ba cách viết AX, AH, AL đều có thể sử dụng như những thanh ghi riêng biệt.

- *BX (base)* thanh ghi cơ sở, rộng 16 bit, cũng được chia ra làm BH và BL. Đây là thanh ghi thường dùng chứa địa chỉ cơ sở của một bảng dùng trong lệnh XLAT. Cả ba cách viết BX, BH, BL đều có thể sử dụng như những thanh ghi riêng biệt.

- *CX (count)* bộ đếm, rộng 16 bit. Được chia ra làm CH và CL. Thanh ghi CX được dùng để chứ số lần lặp trong trường hợp các lệnh LOOP. Thanh ghi thấp CL được dùng để chứa (nhớ) số lần quay hoặc dịch của các lệnh quay (rotate) và dịch (shift).

- *DX (data)* thanh ghi dữ liệu, rộng 16 bit. Thanh ghi này cùng thanh ghi AX tham gia vào các thao tác của phép nhân hoặc chia các số 16 bit. DX còn dùng để chứa địa chỉ 16 bit của các cổng cứng (dài hơn 8 bit) trong các lệnh truy nhập các cổng ngoại vi (I/O port).

4.1.2.2. Các thanh ghi đoạn (segment registers)

Các thanh ghi đoạn dùng để ghi địa chỉ một đoạn bộ nhớ. Vi mạch 8088/8086 có 20 đường dây trên bus địa chỉ. Do các thanh ghi con trỏ, thanh ghi chỉ số chỉ rộng 16 bit nên không thể định địa chỉ cho toàn bộ nhớ vật lý của máy tính là (2^{20} = 1.048.576 = 1Mbyte). Vì vậy trong chế độ thực (real mode) bộ nhớ được chia làm nhiều đoạn để một thanh ghi con trỏ 16 bit có thể quản lý được. Các thanh ghi đoạn 16 bit sẽ chỉ ra địa chỉ đầu của 4 đoạn trong bộ nhớ, dung lượng lớn nhất của mỗi đoạn nhớ sẽ dài 2^{16} = 64 Kbyte và tại một thời điểm nhất định bộ vi xử lý chỉ làm việc được với 4 đoạn nhớ 64Kbyte này. Việc thay đổi giá trị của các thanh ghi đoạn làm cho các đoạn có thể dịch chuyển linh hoạt trong không gian 1 Mbyte, vì vậy các đoạn có thể nằm cách nhau khi thông tin cần lưu trong chúng đòi hỏi dung lượng đủ 64 Kbyte hoặc cũng có thể nằm trùm nhau do có những đoạn không dùng hết độ dài 64 Kbyte và vì thế các đoạn khác có thể bắt đầu nối tiếp ngay sau đó. Địa chỉ của ô nhớ nằm ở đầu đoạn được ghi trong một thanh ghi đoạn 16 bit, địa chỉ này gọi là *địa chỉ cơ sở*. Mười sáu bit này tương ứng với các đường dây địa chỉ từ A4 đến A20. Như vậy giá trị vật lý của địa chỉ đoạn là giá trị trong thanh ghi đoạn dịch sang trái 4 vị trí. Điều này tương đương với phép nhân với 2^4 = 16. Địa chỉ của các ô nhớ khác nằm trong đoạn tính được bằng cách cộng thêm vào địa chỉ cơ sở một giá trị gọi là địa chỉ lệch hay độ lệch (offset), gọi như thế vì nó ứng với khoảng lệch của toạ độ một ô nhớ cụ thể nào đó so với ô đầu đoạn. Độ lệch này được xác định bởi các thanh ghi 16 bit khác đóng vai trò thanh ghi lệch (offset register). Nguyên tắc này dẫn đến công thức tính địa chỉ vật lý (physical address) từ địa chỉ đoạn (segment) trong thanh ghi đoạn và địa chỉ lệch (offset) trong thanh ghi con trỏ như sau:

Địa chỉ vật lý = Thanh ghi đoạn x 16 + Thanh ghi lệch

Việc dùng hai thanh ghi để nhớ thông tin về địa chỉ thực chất tạo ra một loại địa chỉ gọi là địa chỉ logic và được ký hiệu như sau:

Thanh ghi đoạn: Thanh ghi lệch hay *segment:offset.*

Địa chỉ kiểu *segment: offset* là logic vì nó tồn tại dưới dạng giá trị của các thanh ghi cụ thể bên trong CPU và khi cần thiết truy nhập ô nhớ nào đó thì nó phải đổi ra địa chỉ vật lý để rồi đưa lên bus địa chỉ. Việc chuyển đổi này do một bộ tạo địa chỉ thực hiện.

Vi xử lý 16 bit có 4 thanh ghi đoạn như sau:

- *CS (code segment)* là thanh ghi đoạn mã 16 bit. Thanh ghi này phối hợp với con trỏ lệnh IP để ghi địa chỉ mã lệnh trong bộ nhớ. Địa chỉ đầy đủ là CS:IP.

- *DS (data segment)* là thanh ghi đoạn 16 bit cho một đoạn dữ liệu. Thanh ghi này phối hợp với hai thanh ghi chỉ số SI và DI để đánh địa chỉ cho dữ liệu. Địa chỉ đầy đủ cho dữ liệu cần đọc vào là DS:SI, cho dữ liệu cần ghi ra là DS:DI.

- *SS (stack segment)* là thanh ghi đoạn 16 bit cho một ngăn xếp. Địa chỉ đỉnh của ngăn xếp được biểu diễn cùng với con trỏ ngăn xếp SP là SS:SP.

- *ES (extra segment)* là thanh ghi dữ liệu phụ có chiều dài 16 bit. Thường được dùng để đánh địa chỉ một chuỗi. ES:DI là địa chỉ chuỗi cần viết đến (chuỗi đích) và DS:SI là địa chỉ chuỗi đọc vào (chuỗi nguồn).

4.1.2.3. Các thanh ghi con trỏ và chỉ số

Các thanh ghi con trỏ và chỉ số có thể được dùng như một thanh ghi đa năng 16 bit. Vi mạch 8088 có tất cả ba thanh ghi con trỏ là (IP, BP, SP) và hai thanh ghi chỉ số (SI, DI). Nhiệm vụ của từng thanh ghi như sau:

- *IP (instruction pointer)* là con trỏ chỉ tới lệnh máy tiếp theo. Lệnh này nằm trong bô nhớ mà địa chỉ đoạn được ghi trong CS. Như vậy địa chỉ của mã k=lệnh này là CS:IP.

- *BP (base pointer)* là con trỏ cơ sở trỏ về dữ liệu bộ nhớ mà địa chỉ đoạn được ghi trong SS. Địa chỉ đầy đủ sẽ là SS:BP.

- *SP (stack pointer)* là con trỏ ngăn xếp luôn trỏ vào đỉnh ngăn xếp mà địa chỉ đoạn được ghi trong SS. Địa chỉ đầy đủ của dữ liệu là DS:SP.

- *SI (source index)* là chỉ số nguồn, trỏ vào dữ liệu mà địa chỉ đoạn được ghi trong DS. Địa chỉ đầy đủ của dữ liệu là DS:SI.

- *DI (destination index)* là chỉ số đích, cũng trỏ vào đoạn dữ liệu mà địa chỉ đoạn ghi trong DS. Địa chỉ đầy đủ của đoạn dữ liệu là DS:SI.

4.1.2.4. Thanh ghi cờ FR (flag register)

Đây là thanh ghi khá đặc biệt trong CPU, dùng để ghi trạng thái kết quả các phép xử lý trong đơn vị số học và logic ALU hoặc một trạng thái hoạt động của EU. Dựa vào các cờ này người lập trình có thể có các lệnh thích hợp tiếp theo cho bộ vi xử lý (các lệnh nhảy có điều kiện). Thanh ghi này là một thanh ghi 16 bit trong 8088/8086. Nhưng chỉ có 9 bit trong thanh ghi được định nghĩa và sử dụng [2], đó là:

x	x	x	x	O	D	I	T	S	Z	x	A	x	P	x	C

x: bit không được định nghĩa.

Hình 4-1. Sơ đồ thanh ghi cờ của bộ vi xử lý 8086/8088

- *Bit 0: CF* (carry flag) cờ nhớ, CF=1 khi có nhớ hoặc mượn từ MSB.

- *Bit 2: PF* (parity flag) cờ parity, PF phản ảnh tính chẵn (parity) của tổng số bit 1 có trong kết quả. Cờ PF =1 khi tổng số bit 1 trong kết quả là chẵn (even parity, parity chẵn).

- *Bit 4: AF* (auxliary carry flag) cờ nhớ phụ dùng cho các phép tính với mã BCD. AF = 1 khi có nhớ hoặc mượn từ một số BCD thấp (4 bit thấp) sang một số BCD cao (4 bit cao).

- *Bit 6: ZF* (zero flag) cờ rỗng, ZF = 1 khi kết quả bằng 0.

- Bit 7: SF (sing flag) cờ dấu, SF = 1 khi kết quả âm.

- *Bit 8: TF* (trap flag) cờ bẫy, TF = 1 khi vi xử lý ở trong chế độ chạy từng lệnh (chế độ này dùng khi cần tìm lỗi trong một chương trình).

- Bit 9: IF (interrupt enable flag) cờ cho phép ngắt, IF = 1 cho phép các yêu cầu ngắt che được (maskable interrupt) được tác động.

- Bit A: DF (direction flag) cờ hướng. DF = 1 khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải sang trái (lùi).

- *Bit B: OF* (overflow) cờ tràn, OF =1 khi kết quả vượt ra ngoài giới hạn, xảy ra đối với phép tính có dấu.

4.1.3. Đơn vị số học và Logic

Đơn vị số học và logic (Arithmetic and Logic Unit - ALU) thực hiện các phép toán số học và logic trên các dữ liệu cụ thể. ALU bao gồm các thiết bị thực hiện các phép tính số học và các phép tính logic. Thông thường các phép tính cơ bản được thực hiện ở đây như: cộng, trừ (các số nhị phân) hai toán hạng; các phép toán logic như AND, OR, NOR, NOT hai toán hạng; các phép toán đảo, quay, dịch các bít ... Mặt khác chức năng của ALU còn bao gồm cả việc quyết định các trình tự thao tác đối với hệ thống, nó hình thành và quản lý toàn bộ các tín hiệu điều khiển để xắp xếp hợp lý các phép toán và dòng dữ liệu bên trong cũng như bên ngoài ALU, nó điều khiển dòng dữ liệu của BUS địa chỉ, BUS dữ liệu, quản lý và biên dịch các tín hiệu điều khiển trên BUS điều khiển của hệ thống.

4.1.4. Đơn vị điều khiển

Chức năng:

- Điều khiển nhận lệnh từ bộ nhớ đưa vào thanh ghi lệnh.
- Tăng nội dung của PC để trỏ sang lệnh kế tiếp.
- Giải mã lệnh đã nhận được và thao tác với lệnh yêu cầu.
- Phát ra các tín hiệu điều khiển thực hiện lệnh.
- Nhận các tín hiệu từ BUS hệ thống và đáp ứng các yêu cầu đó.
- Đơn vị điều khiển tạo ra các thao tác xảy ra trong CPU.

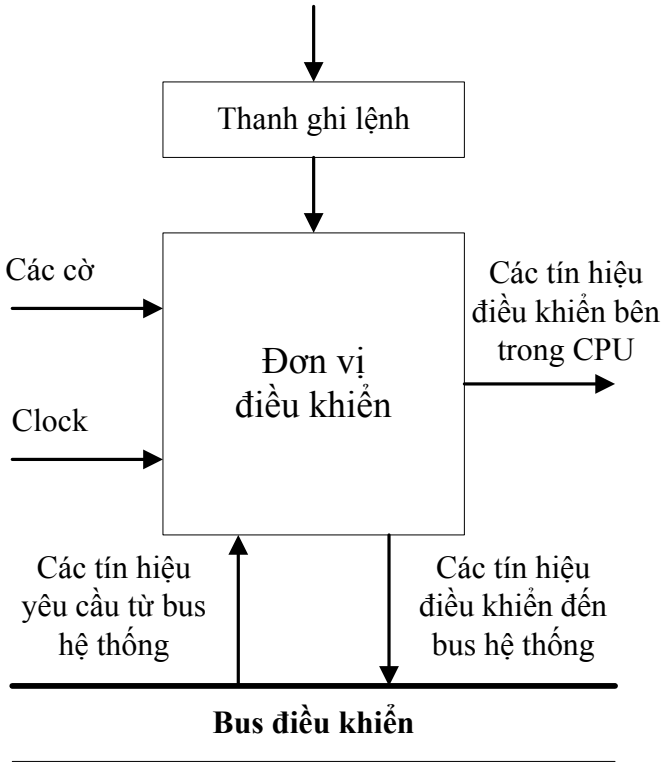
Đơn vị điều khiển thực hiện hai thao tác chính sau:

Sự sắp xếp chuỗi (Sequencing): Đơn vị điều khiển CPU sắp xếp chuỗi vi thao tác vào một chuỗi liên tục thích hợp, dựa trên chương trình đang được thực hiện.

Sự thi hành (Execution): Đơn vị điều khiển mỗi vi thao tác được thực hiện. Đơn vị điều khiển thao tác dựa vào việc sử dụng các tín hiệu điều khiển.

4.1.4.1. Tín hiệu điều khiển

Đối với đơn vị điều khiển để thực hiện được chức năng của nó, nó phải có dữ liệu và cho phép nó xác định trạng thái của hệ thống và mục ra cho phép nó điều khiển tác động của hệ thống. Nội tại, đơn vị điều khiển phải có logic yêu cầu thực hiện chuỗi vi thao tác và thi hành các chức năng.



Hình 4-2. Mô hình kết nối đơn vị điều khiển

Tín hiệu điều khiển được thể hiện trong hình trên bao gồm tín hiệu vào và tín hiệu ra.

a) Các tín hiệu vào

- *Clock:* Đây là cách đơn vị điều khiển giữ thời gian. Đơn vị điều khiển tạo ra một vi thao tác (hoặc một tập các thao tác đồng thời) được thực hiện với mỗi xung đồng hồ.
- *Thanh ghi chỉ lệnh:* Được dùng để xác định vi thao tác nào được thực hiện trong chu kỳ thi hành.
- *Cờ:* Xác định trạng thái của CPU và kết quả của thao tác ALU.
- *Các tín hiệu điều khiển từ BUS điều khiển:* BUS điều khiển của BUS hệ thống cung cấp tín hiệu cho đơn vị điều khiển, như là tín hiệu ngắt và sự công nhận.

b) Các tín hiệu ra

- *Tín hiệu điều khiển trong CPU*: Khiến dữ liệu di chuyển từ một thanh ghi tới các thanh ghi khác và làm hoạt động các chức năng ALU cụ thể.
- Các tín hiệu điều khiển BUS: Có tín hiệu điều khiển bộ nhớ và tín hiệu điều khiển module vào ra.

4.1.4.2. Đơn vị điều khiển vi chương trình

Để thực hiện một lệnh, đơn vị logic tuần tự đưa ra một lệnh đọc tới bộ nhớ điều khiển.

- Từ mã địa chỉ được xác định trong thanh ghi địa chỉ điều khiển được đọc vào thanh ghi bộ đệm điều khiển.
- Nội dung của thanh ghi bộ đệm điều khiển phát ra tín hiệu điều khiển và thông tin địa chỉ tiếp theo cho đơn vị logic tuần tự.
- Đơn vị logic tuần tự tải địa chỉ mới vào trong thanh ghi địa chỉ điều khiển dựa vào thông tin địa chỉ tiếp theo từ thanh ghi bộ đệm điều khiển và các cờ ALU.

Tất cả xảy ra trong một xung đồng hồ.

4.1.5. Các đặc trưng cơ bản của lệnh máy

4.1.5.1. Giới thiệu chung về tập lệnh

Nếu coi phần mạch điện tử của CPU là "phần xác" thì tập lệnh (Instruction Set) chính là "phần hồn" của bộ não máy tính. Nhờ có tập lệnh, CPU có khả năng lập trình được để thực hiện các công việc hữu ích cho người dùng.

Có thể định nghĩa lệnh máy tính một cách đơn giản: Lệnh máy tính (Computer Instruction) là một từ nhị phân (binary word) được gán một nhiệm vụ cụ thể. Các lệnh của chương trình được lưu trong bộ nhớ và chúng lần lượt được CPU đọc, giải mã và thực hiện. Tập lệnh máy tính thường gồm nhiều lệnh có thể được chia thành một số nhóm theo chức năng: nhóm các lệnh vận chuyển dữ liệu (data movement), nhóm các lệnh tính toán (computational), nhóm các lệnh điều kiện và rẽ nhánh conditonal and branching) và một số lệnh khác.

Việc thực hiện lệnh có thể được chia thành các pha (phase) hay giai đoạn (stage). Mỗi lệnh có thể được thực hiện theo 4 giai đoạn: (1) Đọc lệnh (Instruction fetch - IF): lệnh được đọc từ bộ nhớ về CPU; (2) Giải mã (Instruction decode - ID): CPU giải mã lệnh; (3) Thực hiện lệnh (Instruction execution – EX): CPU thực hiện lệnh; và (4) Lưu kết quả (Write back - WB): kết quả thực hiện lệnh (nếu có) được lưu vào bộ nhớ [2].

Mỗi bộ xử lý đều có một tập lệnh xác định.

Tập lệnh thường có hàng chục tới hàng trăm lệnh.

Mỗi lệnh là một chuỗi số nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định.

Các lệnh được miêu tả bằng các ký hiệu gợi nhớ (Chính là các lệnh của hợp ngữ).

4.1.5.2. Các thành phần của lệnh máy

Dạng tổng quát của lệnh máy tính gồm có 2 phần chính: (1) mã lệnh hay mã thao tác (opcode – operation code) và (2) địa chỉ của các toán hạng (Addresses of Operands). Mỗi lệnh có một mã lệnh riêng và được biểu biễn bằng một số bit. Chẳng hạn, mã lệnh của CPU Intel 8086 được biểu diễn bởi 6 bit. Mỗi lệnh có thể có một hoặc nhiều toán hạng và mỗi toán hạng là một địa chỉ.

Mã thao tác	Địa chỉ các toán hạng

Mã thao tác (Operation code – opcode): Mã hóa cho thao tác mà bộ xử lý phải thực hiện.

Địa chỉ toán hạng (Operand): Chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động.

Toán hạng nguồn: Dữ liệu vào của thao tác.

Toán hạng đích: Dữ liệu ra của thao tác.

Không phải lệnh nào cũng có Địa chỉ các toán hạng.

Một lệnh chỉ cần chỉ ra:

Chức năng của lệnh

Nơi chứa dữ liệu

Nơi lưu kết quả

4.1.5.3. Mô tả lệnh

Lệnh máy là nhị phân.

Để dễ hiểu và dễ nhớ đối với con người, người ta mô tả lệnh bằng các ký hiệu gợi nhớ. VD: ADD, SUB, LOAD…

Toán hạng có thể được miêu tả như sau: ADD A, B

4.1.5.4. Các kiểu lệnh

Xử lý dữ liệu

Lưu trữ dữ liệu (Store, Load)

Vận chuyển dữ liệu (Vào/ra – IN, OUT, IOR, IOW)

Lệnh làm việc với thanh ghi, bộ nhớ

Lệnh nhảy (Điều khiển thay đổi trật tự của chương trình): JMP, BR..

Lệnh biến đổi dữ liệu

4.1.5.5. Các thao tác khi thực hiện lệnh

IF: Instruction Fetch: Nhận lệnh

ID: Instruction Decode: Giải mã lệnh

DF: Data: Nhận dữ liệu

EX: Execution: Thực hiện

DS: Data Store: Lưu trữ kết quả

Thông thường địa chỉ của lệnh tiếp theo → PC (+1)

4.1.5.6. Các vấn đề về thiết kế tập lệnh

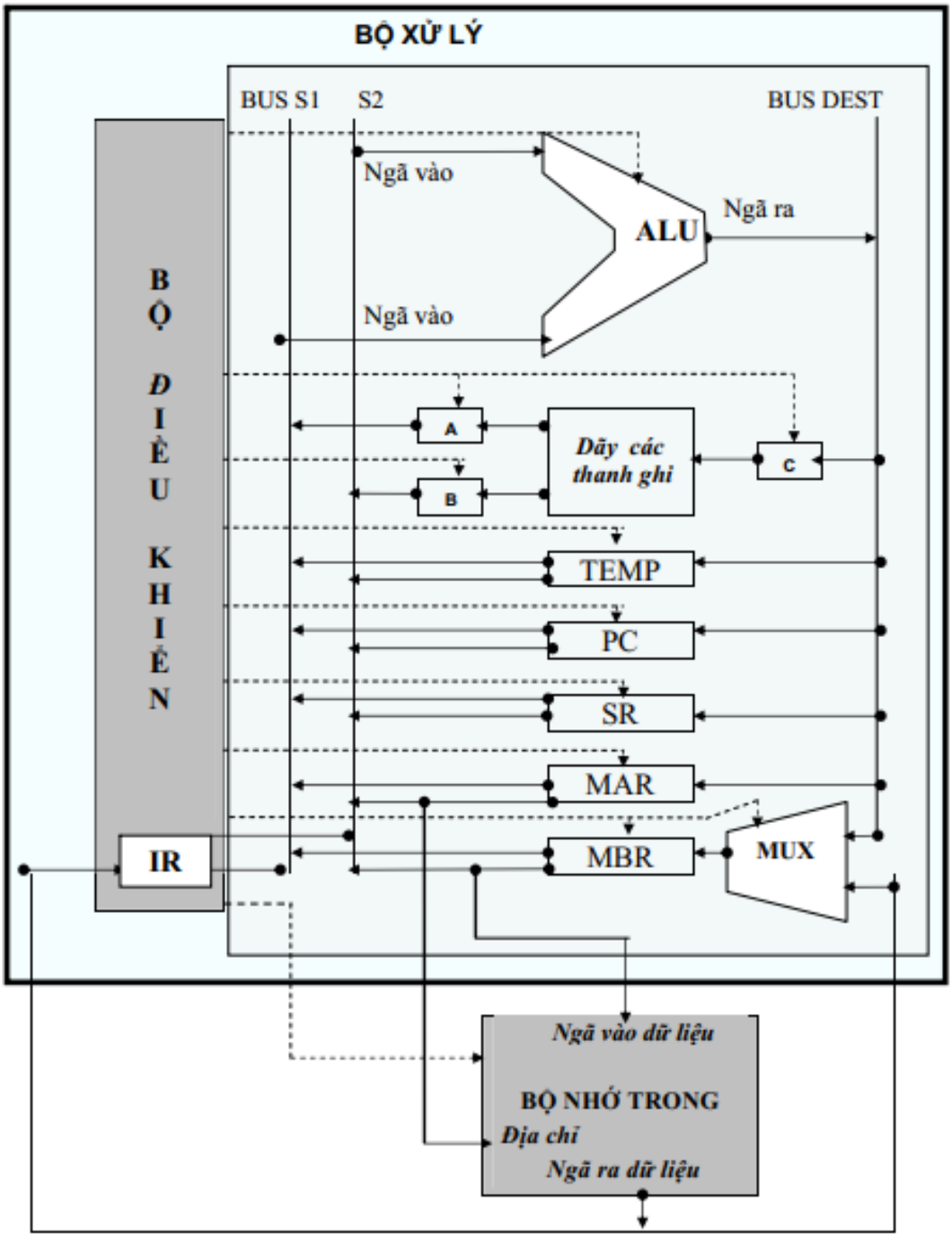
- Về thao tác. Chúng ta phải quan tâm tới các vấn đề như: Bao nhiêu thao tác? Các thao tác như thế nào? Mức độ phức tạp của thao tác.
- Các kiểu dữ liệu.
- Các khuôn dạng lệnh. Độ dài của trường mã thao tác, số lượng địa chỉ toán hạng.
- Các thanh ghi: Quan tâm tới số thanh ghi của CPU được sử dụng, các thao tác nào được sử dụng trên các thanh ghi.
- Các phương pháp địa chỉ (Addressing Modes)
- RISC hay CISC (Reduced Instruction Set Computing, Complex Instruction Set Computing)

4.2. ĐƯỜNG ĐI CỦA DỮ LIỆU

Phần đường đi dữ liệu gồm có bộ phận làm tính và luận lý (ALU: Arithmetic and Logic Unit), các mạch dịch, các thanh ghi và các đường nối kết các bộ phận trên. Phần này chứa hầu hết các trạng thái của bộ xử lý. Ngoài các thanh ghi tổng quát, phần đường đi dữ liệu còn chứa thanh ghi đếm chương trình (PC: Program Counter), thanh ghi trạng thái (SR: Status Register), thanh ghi đệm TEMP (Temporary), các thanh ghi địa chỉ bộ nhớ (MAR: Memory Address Register), thanh ghi số liệu bộ nhớ (MBR: Memory Buffer Register), bộ đa hợp (MUX: Multiplexor), đây là điểm cuối của các kênh dữ liệu - CPU và bộ nhớ, với nhiệm vụ lập thời biểu truy cập bộ nhớ từ CPU và các kênh dữ liệu, hệ thống bus nguồn (S1, S2) và bus kết quả (Dest).

Nhiệm vụ chính của phần đường đi dữ liệu là đọc các toán hạng từ các thanh ghi tổng quát, thực hiện các phép tính trên toán hạng này trong bộ làm tính và luận lý ALU và lưu trữ kết quả trong các thanh ghi tổng quát. Ở ngã vào và ngã ra các thanh ghi tổng quát có các mạch chốt A, B, C. Thông thường, số lượng các thanh ghi tổng quát là 32.

Phần đường đi của dữ liệu chiếm phân nửa diện tích của bộ xử lý nhưng là phần dễ thiết kế và cài đặt trong bộ xử lý.



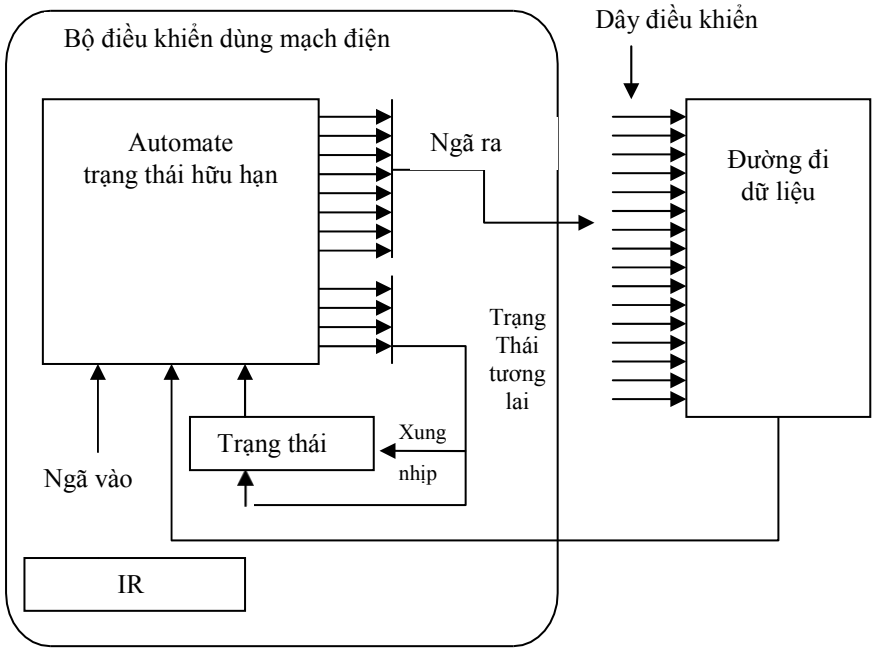
Hình 4-3. Tổ chức của một xử lý điển hình
(Các đường không liên tục là các đường điều khiển)

Bộ điều khiển tạo các tín hiệu điều khiển di chuyển số liệu (tín hiệu di chuyển số liệu từ các thanh ghi đến bus hoặc tín hiệu viết vào các thanh ghi), điều khiển các tác vụ mà các bộ phận chức năng phải làm (điều khiển ALU, điều khiển đọc và viết vào bộ nhớ trong...). Bộ điều khiển cũng tạo các tín hiệu giúp các lệnh được thực hiện một cách tuần tự.

Việc cài đặt bộ điều khiển có thể dùng một trong hai cách sau: dùng mạch điện tử hoặc dùng vi chương trình (microprogram).

4.2.1. Bộ điều khiển mạch điện tử

Để hiểu được vận hành của bộ điều khiển mạch điện tử, chúng ta xét đến mô tả về Automate trạng thái hữu hạn: Có nhiều hệ thống hay nhiều thành phần mà ở mỗi thời điểm xem xét đều có một trạng thái (state). Mục đích của trạng thái là ghi nhớ những gì có liên quan trong quá trình hoạt động của hệ thống. Vì chỉ có một số trạng thái nhất định nên nói chung không thể ghi nhớ hết toàn bộ lịch sử của hệ thống, do vậy nó phải được thiết kế cẩn thận để ghi nhớ những gì quan trọng. Ưu điểm của hệ thống (chỉ có một số hữu hạn các trạng thái) đó là có thể cài đặt hệ thống với một lượng tài nguyên cố định. Chẳng hạn, chúng ta có thể cài đặt Automate trạng thái hữu hạn trong phần cứng máy tính ở dạng mạch điện hay một dạng chương trình đơn giản, trong đó, nó có khả năng quyết định khi chỉ biết một lượng giới hạn dữ liệu hoặc bằng cách dùng vị trí trong đoạn mã lệnh để đưa ra quyết định.

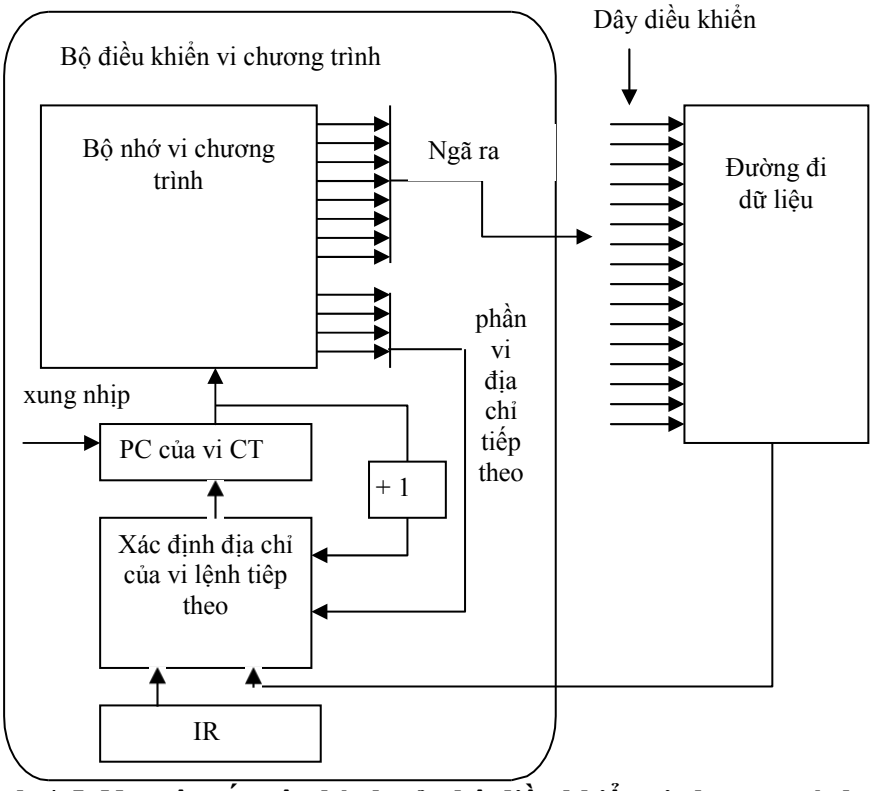


Hình 4-4. Nguyên tắc vận hành của bộ điều khiển dùng mạch điện

Hình 4-5. cho thấy nguyên tắc của một bộ điều khiển bằng mạch điện. Các đường điều khiển của phần đường đi số liệu là các ngã ra của một hoặc nhiều Automate trạng thái hữu hạn. Các ngã vào của Automate gồm có thanh ghi lệnh, thanh ghi này chứa lệnh phải thi hành và những thông tin từ bộ đường đi số liệu. Ứng với cấu hình các đường vào và trạng thái hiện tại, Automate sẽ cho trạng thái tương lai và các đường ra tương ứng với trạng thái hiện tại. Automate được cài đặt dưới dạng là một hay nhiều mạch mảng logic lập trình được (PLA: Programmable Logic Array) hoặc các mạch logic ngẫu nhiên.

Kỹ thuật điều khiển này đơn giản và hữu hiệu khi các lệnh có chiều dài cố định, có dạng thức đơn giản. Nó được dùng nhiều trong các bộ xử lý RISC.

4.2.1.1. Bộ điều khiển vi chương trình:



Hình 4-5. Nguyên tắc vận hành của bộ điều khiển vi chương trình

Sơ đồ nguyên tắc của bộ điều khiển dùng vi chương trình được trình bày ở hình 4-5. Trong kỹ thuật này, các đường dây điều khiển của bộ đường đi dữ liệu ứng với các ngã ra của một vi lệnh nằm trong bộ nhớ vi chương trình. Việc điều khiển các tác vụ của một lệnh mã máy được thực hiện bằng một chuỗi các vi lệnh. Một vi máy tính nằm bên trong bộ điều khiển thực hiện từng lệnh của vi chương trình này. Chính vi máy tính này điều khiển việc thực hiện một cách tuần tự các vi lệnh để hoàn thành tác vụ mà lệnh mã máy phải thực hiện. Các tác vụ của lệnh mã máy cũng tuỳ thuộc vào trạng thái của phần đường đi dữ liệu.

Bộ điều khiển bằng vi chương trình được dùng rộng rãi trong các bộ xử lý CISC. Bộ xử lý này có tập lệnh phức tạp với các lệnh có chiều dài khác nhau và có dạng thức phức tạp. Trong các bộ xử lý CISC, người ta cài đặt một lệnh mã máy bằng cách viết một vi chương trình. Như vậy công việc khá đơn giản và rất hữu hiệu. Các sai sót trong thiết kế automat điều khiển cũng dễ sửa đổi.

4.2.2. Diễn biến thi hành lệnh mã máy

Việc thi hành một lệnh mã máy có thể chia thành 5 giai đoạn:

- *Đọc lệnh* (IF: Instruction Fetch)
- *Giải mã lệnh* (ID: Instruction Decode)
- *Thi hành lệnh* (EX: Execute)
- *Thâm nhập bộ nhớ trong hoặc nhảy* (MEM: Memory access)
- *Lưu trữ kết quả* (RS: Result Storing).

Mỗi giai đoạn được thi hành trong một hoặc nhiều chu kỳ xung nhịp.

4.2.2.1. Đọc lệnh:

MAR ← PC

IR ← M[MAR]

Bộ đếm chương trình PC được đưa vào MAR . Lệnh được đọc từ bộ nhớ trong, tại các ô nhớ có địa chỉ nằm trong MAR và được đưa vào thanh ghi lệnh IR.

4.2.2.2. Giải mã lệnh và đọc các thanh ghi nguồn:

A ←Rs1
B ←Rs2

PC ← PC + 4

Lệnh được giải mã. Kế đó các thanh ghi Rs1 và Rs2 được đưa vào A và B. Thanh ghi PC được tăng lên để chỉ tới lệnh kế đó. Để hiểu rõ giai đoạn này, ta lấy dạng thức của một lệnh làm tính tiêu biểu sau đây:

Mã lệnh	Thanh ghi Rs1	Thanh ghi Rs2	Thanh ghi Rd	Tác vụ
bit *6*	*5*	*5*	*5*	*11*

Các thanh ghi nguồn Rs1 và Rs2 được sử dụng tuỳ theo tác vụ, kết quả được đặt trong thanh ghi đích Rd.

Ta thấy việc giải mã được thực hiện cùng lúc với việc đọc các thanh ghi Rs1 và Rs2 vì các thanh ghi này luôn nằm tại cùng vị trí ở trong lệnh.

4.2.2.3. Thi hành lệnh:

Tuỳ theo loại lệnh mà một trong ba nhiệm vụ sau đây được thực hiện:

- Liên hệ tới bộ nhớ

MAR ← Địa chỉ do ALU tính tuỳ theo kiểu định vị (Rs2). MBR ← Rs1

Địa chỉ hiệu dụng do ALU tính được đưa vào MAR và thanh ghi nguồn Rs1 được đưa vào MBR để được lưu vào bộ nhớ trong.

- Một lệnh của ALU

Ngã ra ALU ← Kết quả của phép tính

ALU thực hiện phép tính xác định trong mã lệnh, đưa kết quả ra ngã ra.

- Một phép nhảy

Ngã ra ALU ← Địa chỉ lệnh tiếp theo do ALU tính.

ALU cộng địa chỉ của PC với độ dời để làm thành địa chỉ đích và đưa địa chỉ này ra ngã ra. Nếu là một phép nhảy có điều kiện thì thanh ghi trạng thái được đọc quyết định có cộng độ dời vào PC hay không.

4.2.2.4. Thâm nhập bộ nhớ trong hoặc nhảy lần cuối

Giai đoạn này thường chỉ được dùng cho các lệnh nạp dữ liệu, lưu giữ dữ liệu và lệnh nhảy.

a) Tham khảo đến bộ nhớ:

MBR ← M[MAR] hoặc M[MAR] ← MBR

Số liệu được nạp vào MBR hoặc lưu vào địa chỉ mà MAR trỏ đến.

b) Nhảy:

If (điều kiện), PC ← ngả ra ALU

Nếu điều kiện đúng, ngã ra ALU được nạp vào PC. Đối với lệnh nhảy không điều kiện, ngả ra ALU luôn được nạp vào thanh ghi PC.

4.2.2.5. Lưu trữ kết quả

Rd ← Ngã ra ALU hoặc Rd ← MBR Lưu trữ kết quả trong thanh ghi đích.

4.2.3. Ngắt quãng (INTERRUPT)

Ngắt quãng là một sự kiện xảy ra một cách ngẫu nhiên trong máy tính và làm ngưng tính tuần tự của chương trình (nghĩa là tạo ra một lệnh nhảy). Phần lớn các nhà sản xuất máy tính (ví dụ như IBM, INTEL) dùng từ ngắt quãng để ám chỉ sự kiện này, tuy nhiên một số nhà sản xuất khác dùng từ "ngoại lệ", "lỗi", "bẫy" để chỉ định hiện tượng này.

Bộ điều khiển của CPU là bộ phận khó thực hiện nhất và ngắt quãng là phần khó thực hiện nhất trong bộ điều khiển. Để nhận biết được một ngắt quãng lúc đang thi hành một lệnh, ta phải biết điều chỉnh chu kỳ xung nhịp và điều này có thể ảnh hưởng đến hiệu quả của máy tính.

Người ta đã nghỉ ra "ngắt quãng" là để nhận biết các sai sót trong tính toán số học, và để ứng dụng cho những hiện tượng thời gian thực. Bây giờ, ngắt quãng được dùng cho các công việc sau đây:

- Ngoại vi đòi hỏi nhập hoặc xuất số liệu.
- Người lập trình muốn dùng dịch vụ của hệ điều hành.

- Cho một chương trình chạy từng lệnh.
- Làm điểm dừng của một chương trình.
- Báo tràn số liệu trong tính toán số học.
- Trang bộ nhớ thực sự không có trong bộ nhớ.
- Báo vi phạm vùng cấm của bộ nhớ.
- Báo dùng một lệnh không có trong tập lệnh.
- Báo phần cứng máy tính bị hư.
- Báo điện bị cắt.

Dù rằng ngắt quãng không xảy ra thường xuyên nhưng bộ xử lý phải được thiết kế sao cho có thể lưu giữ trạng thái của nó trước khi nhảy đi phục vụ ngắt quãng. Sau khi thực hiện xong chương trình phục vụ ngắt, bộ xử lý phải khôi phục trạng thái của nó để có thể tiếp tục công việc.

Để đơn giản việc thiết kế, một vài bộ xử lý chỉ chấp nhận ngắt sau khi thực hiện xong lệnh đang chạy. Khi một ngắt xảy ra, bộ xử lý thi hành các bước sau đây:

1. *Thực hiện xong lệnh đang làm.*
2. *Lưu trữ trạng thái hiện tại.*
3. *Nhảy đến chương trình phục vụ ngắt*
4. *Khi chương trình phục vụ chấm dứt, bộ xử lý khôi phục lại trạng thái cũ của nó và tiếp tục thực hiện chương trình mà nó đang thực hiện khi bị ngắt.*

4.2.4. Kỹ thuật ống dẫn (PIPELINE)

Đây là một kỹ thuật làm cho các giai đoạn khác nhau của nhiều lệnh được thi hành cùng một lúc.

Ví dụ: Chúng ta có những lệnh đều đặn, mỗi lệnh được thực hiện trong cùng một khoản thời gian. Giả sử, mỗi lệnh được thực hiện trong 5 giai đoạn và mỗi giai đoạn được thực hiện trong 1 chu kỳ xung nhịp. Các giai đoạn thực hiện một lệnh là: lấy lệnh (IF: Instruction Fetch), giải mã (ID: Instruction Decode), thi hành (EX: Execute), thâm nhập bộ nhớ (MEM: Memory Access), lưu trữ kết quả (RS: Result Storing). Hình 4-7. cho thấy chỉ trong một chu kỳ xung nhịp, bộ xử lý có thể thực hiện một lệnh (bình thường lệnh này được thực hiện trong 5 chu kỳ).

Chuỗi lệnh	**Chu kỳ xung nhịp**								
	1	2	3	4	5	6	7	8	9
Lệnh thứ i	IF	ID	EX	MEM	RS				
Lệnh thứ i+1		IF	ID	EX	MEM	RS			
Lệnh thứ i+2			IF	ID	EX	MEM	RS		
Lệnh thứ i+3				IF	ID	EX	MEM	RS	RS
Lệnh thứ i+4					IF	ID	EX	MEM	

Hình 4-6. Các giai đoạn khác nhau của nhiều lệnh được thi hành cùng một lúc

So sánh với kiểu xử lý tuần tự thông thường, 5 lệnh được thực hiện trong 25 chu kỳ xung nhịp, thì xử lý lệnh theo kỹ thuật ống dẫn thực hiện 5 lệnh chỉ trong 9 chu kỳ xung nhịp.

Như vậy kỹ thuật ống dẫn làm tăng tốc độ thực hiện các lệnh. Tuy nhiên kỹ thuật ống dẫn có một số ràng buộc:

- Cần phải có một mạch điện để thi hành mỗi giai đoạn của lệnh vì tất cả các giai đoạn của lệnh được thi hành cùng lúc. Trong một bộ xử lý không dùng kỹ thuật ống dẫn, ta có thể dùng bộ làm toán ALU để cập nhật thanh ghi PC, cập nhật địa chỉ của toán hạng bộ nhớ, địa chỉ ô nhớ mà chương trình cần nhảy tới, làm các phép tính trên các toán hạng vì các phép tính này có thể xảy ra ở nhiều giai đoạn khác nhau.

- Phải có nhiều thanh ghi khác nhau dùng cho các tác vụ đọc và viết. Trên hình 4-7, tại một chu kỳ xung nhịp, ta thấy cùng một lúc có 2 tác vụ đọc (ID, MEM) và 1 tác vụ viết (RS).

- Trong một máy có kỹ thuật ống dẫn, có khi kết quả của một tác vụ trước đó, là toán hạng nguồn của một tác vụ khác. Như vậy sẽ có thêm những khó khăn mà ta sẽ đề cập ở mục tới.

- Cần phải giải mã các lệnh một cách đơn giản để có thể giải mã và đọc các toán hạng trong một chu kỳ duy nhất của xung nhịp.

- Cần phải có các bộ làm tính ALU hữu hiệu để có thể thi hành lệnh số học dài nhất, có số giữ, trong một khoảng thời gian ít hơn một chu kỳ của xung nhịp.

- Cần phải có nhiều thanh ghi lệnh để lưu giữ lệnh mà chúng ta phải xem xét cho mỗi giai đoạn thi hành lệnh.

- Cuối cùng phải có nhiều thanh ghi bộ đếm chương trình PC để có thể tái tục các lệnh trong trường hợp có ngắt quãng.

4.2.5. Khó khăn trong kỹ thuật ống dẫn

Khi thi hành lệnh trong một máy tính dùng kỹ thuật ống dẫn, có nhiều trường hợp làm cho việc thực hiện kỹ thuật ống dẫn không thực hiện được như là: thiếu các mạch chức năng, một lệnh dùng kết quả của lệnh trước, một lệnh nhảy.

Ta có thể phân biệt 3 loại khó khăn: *khó khăn do cấu trúc, khó khăn do số liệu* và *khó khăn do điều khiển*.

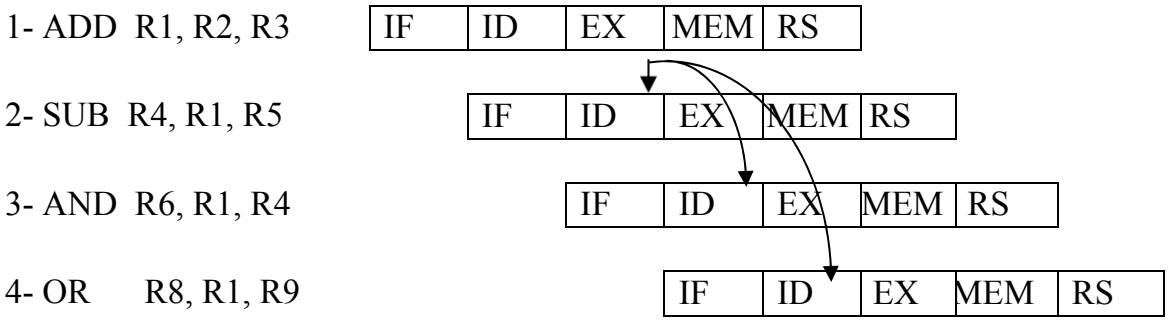
4.2.5.1. Khó khăn do cấu trúc:

Đây là khó khăn do thiếu bộ phận chức năng, ví dụ trong một máy tính dùng kỹ thuật ống dẫn phải có nhiều ALU, nhiều PC, nhiều thanh ghi lệnh IR ... Các khó khăn này được giải quyết bằng cách thêm các bộ phận chức năng cần thiết và hữu hiệu.

4.2.5.2. Khó khăn do số liệu:

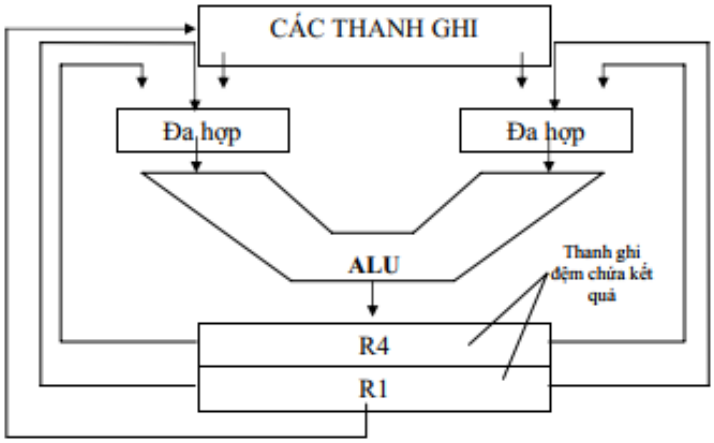
Lấy ví dụ trường hợp các lệnh liên tiếp sau:
Lệnh 1: ***ADD R1, R2, R3***
Lệnh 2: ***SUB R4, R1, R5***
Lệnh 3: ***AND R6, R1, R7***
Lệnh 4: ***OR R8, R1, R9***

Hình 4-8. cho thấy R1, kết quả của lệnh 1 chỉ có thể được dùng cho lệnh 2 sau giai đoạn MEM của lệnh 1, nhưng R1 được dùng cho lệnh 2 vào giai đoạn EX của lệnh 1. Chúng ta cũng thấy R1 được dùng cho các lệnh 3 và 4.



Hình 4-7. Chuỗi lệnh minh hoạ khó khăn do số liệu.

Để khắc phục khó khăn này, một bộ phận phần cứng được dùng để đưa kết quả từ ngã ra ALU trực tiếp vô một trong các thanh ghi ngã vào như trong hình 4-9.



Hình 4-8. ALU với bộ phận phần cứng đưa kết quả tính toán trở lại ngã vào

Khi bộ phận phần cứng nêu trên phát hiện có dùng kết quả của ALU làm toán hạng cho liệt kê, nó tác động vào mạch đa hợp để đưa ngã ra của ALU vào ngã vào của ALU hoặc vào ngã vào của một đơn vị chức năng khác nếu cần.

4.2.5.3. Khó khăn do điều khiển:

Các lệnh làm thay đổi tính thi hành các lệnh một cách tuần tự (nghĩa là PC tăng đều đặn sau mỗi lệnh), gây khó khăn về điều khiển. Các lệnh này là lệnh nhảy đến một địa chỉ tuyệt đối chứa trong một thanh ghi, hay lệnh nhảy đến một địa chỉ xác định một cách tương đối so với địa chỉ hiện tại của bộ đếm chương trình PC. Các lệnh nhảy trên có thể có hoặc không điều kiện.

Trong trường hợp đơn giản nhất, tác vụ nhảy không thể biết trước giai đoạn giải mã (xem hình 4-7). Như vậy, nếu lệnh nhảy bắt đầu ở chu kỳ C thì lệnh mà chương trình

nhảy tới chỉ được bắt đầu ở chu kỳ C+2. Ngoài ra, phải biết địa chỉ cần nhảy đến mà ta có ở cuối giai đoạn giải mã ID. Trong lệnh nhảy tương đối, ta phải cộng độ dời chứa trong thanh ghi lệnh IR vào thanh ghi PC. Việc tính địa chỉ này chỉ được thực hiện vào giai đoạn ID với điều kiện phải có một mạch công việc riêng biệt.

Vậy trong trường hợp lệnh nhảy không điều kiện, lệnh mà chương trình nhảy đến bắt đầu thực hiện ở chu kỳ C+2 nếu lệnh nhảy bắt đầu ở chu kỳ C.

Cho các lệnh nhảy có điều kiện thì phải tính toán điều kiện. Thông thường các kiến trúc RISC đặt kết quả việc so sánh vào trong thanh ghi trạng thái, hoặc vào trong thanh ghi tổng quát. Trong cả 2 trường hợp, đọc điều kiện tương đương với đọc thanh ghi. Đọc thanh ghi có thể được thực hiện trong phân nửa chu kỳ cuối giai đoạn ID.

Một trường hợp khó hơn có thể xảy ra trong những lệnh nhảy có điều kiện. Đó là điều kiện được có khi so sánh 2 thanh ghi và chỉ thực hiện lệnh nhảy khi kết quả so sánh là đúng. Việc tính toán trên các đại lượng logic không thể thực hiện được trong phân nửa chu kỳ và như thế phải kéo dài thời gian thực hiện lệnh nhảy có điều kiện. Người ta thường tránh các trường hợp này để không làm giảm mức hữu hiệu của máy tính.

Vậy trường hợp đơn giản, người ta có thể được địa chỉ cần nhảy đến và điều kiện nhảy cuối giai đoạn ID. Vậy có chậm đi một chu kỳ mà người ta có thể giải quyết bằng nhiều cách.

Cách thứ nhất là đóng băng kỹ thuật ống dẫn trong một chu kỳ, nghĩa là ngưng thi hành lệnh thứ i+1 đang làm nếu lệnh thư i là lệnh nhảy. Ta mất trắng một chu kỳ cho mỗi lệnh nhảy.

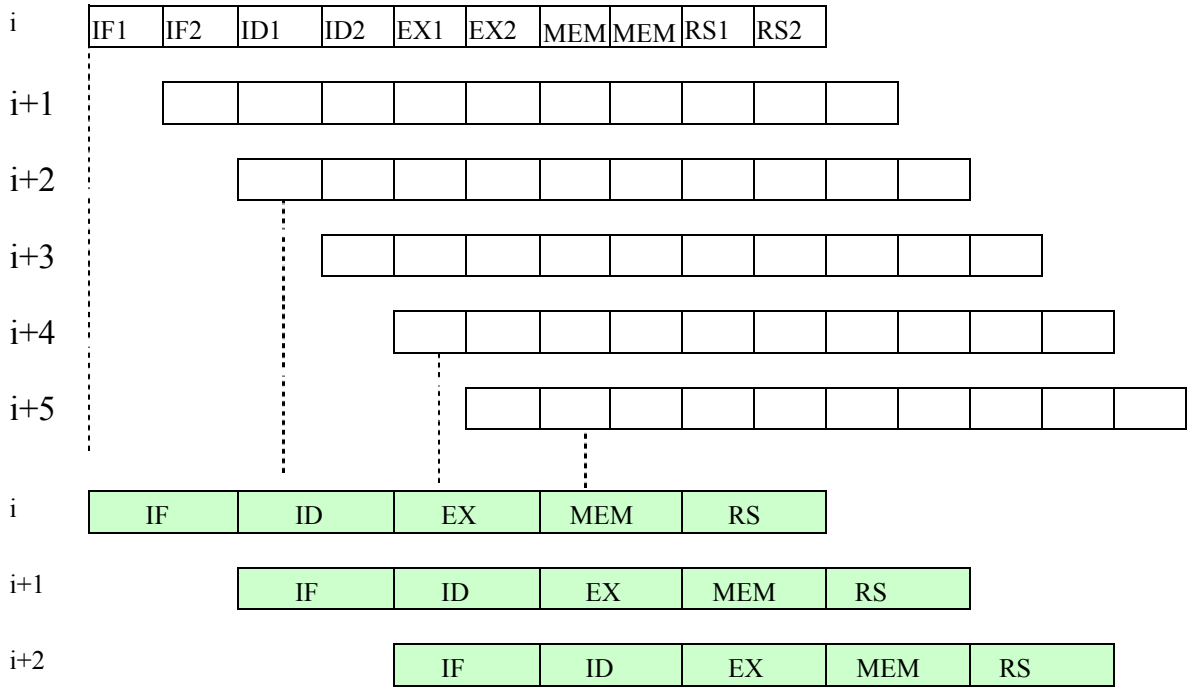
Cách thứ hai là thi hành lệnh sau lệnh nhảy nhưng lưu ý rằng hiệu quả của một lệnh nhảy bị chậm mất một lệnh. Vậy lệnh theo sau lệnh nhảy được thực hiện trước khi lệnh mà chương trình phải nhảy tới được thực hiện. Chương trình dịch hay người lập trình có nhiệm vụ xen vào một lệnh hữu ích sau lệnh nhảy.

Trong trường hợp nhảy có điều kiện, việc nhảy có thể được thực hiện hay không thực hiện. Lệnh hữu ích đặt sau lệnh nhảy không làm sai lệch chương trình dù điều kiện nhảy đúng hay sai.

Bộ xử lý RISC SPARC có những lệnh nhảy với huỷ bỏ. Các lệnh này cho phép thi hành lệnh sau lệnh nhảy nếu điều kiện nhảy đúng và huỷ bỏ thực hiện lệnh đó nếu điều kiện nhảy sai.

4.2.6. Siêu ống dẫn

Máy tính có kỹ thuật siêu ống dẫn bậc n bằng cách chia các giai đoạn của kỹ thuật ống dẫn đơn giản, mỗi giai đoạn được thực hiện trong khoản thời gian Tc, thành n giai đoạn con thực hiện trong khoản thời gian Tc/n. Độ hữu hiệu của kỹ thuật này tương đương với việc thi hành n lệnh trong mỗi chu kỳ Tc. Hình III.7 trình bày thí dụ về siêu ống dẫn bậc 2, có so sánh với siêu ống dẫn đơn giản. Ta thấy trong một chu kỳ Tc, máy dùng kỹ thuật siêu ống dẫn làm 2 lệnh thay vì làm1 lệnh trong máy dùng kỹ thuật ống dẫn bình thường. Trong máy tính siêu ống dẫn, tốc độ thực hiện lệnh tương đương với việc thực hiện một lệnh trong khoảng thời gian Tc/n. Các bất lợi của siêu ống dẫn là thời gian thực hiện một giai đoạn con ngắn Tc/n và việc trì hoãn trong thi hành lệnh nhảy lớn. Trong ví dụ ở hình 4-11, nếu lệnh thứ i là một lệnh nhảy tương đối thì lệnh này được giải mã trong giai đoạn ID, địa chỉ nhảy đến được tính vào giai đoạn EX, lệnh phải được nhảy tới là lệnh thứ i+4, vậy có trì trệ 3 lệnh thay vì 1 lệnh trong kỹ thuật ống dẫn bình thường.

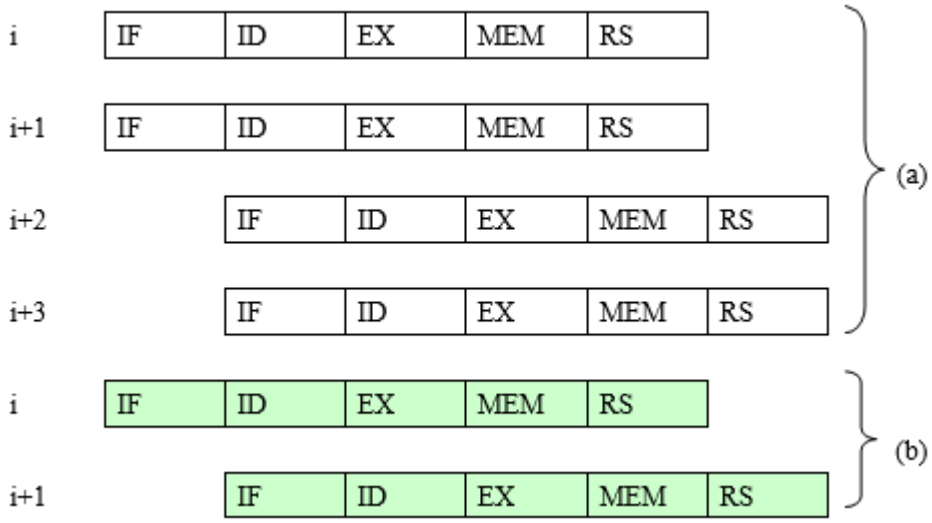


Hình 4-9. Siêu ống dẫn bậc 2 so với siêu ống dẫn đơn giản.

Trong khoảng thời gian Tc, máy có siêu ống dẫn làm 2 lệnh thay vì 1 lệnh như trong máy có kỹ thuật ống dẫn đơn giản.

4.2.7. Siêu vô hướng (SUPERSCALAR)

Máy tính siêu vô hướng bậc n có thể thực hiện đồng thời n lệnh trong một chu kỳ xung nhịp Tc. Hình 4-10. trình bày một ví dụ về sự vận hành của một máy tính siêu vô hướng bậc 2 so với một máy tính dùng kỹ thuật ống dẫn.



Hình 4-10. Siêu vô hướng (a) so với kỹ thuật ống dẫn (b).

Trong một máy tính siêu vô hướng phần cứng phải quản lý việc đọc và thi hành đồng thời nhiều lệnh. Vậy nó phải có khả năng quản lý các quan hệ giữa số liệu với nhau. Cũng cần phải chọn các lệnh có khả năng được thi hành cùng một lúc. Những bộ xử lý đầu tiên đưa ra thị trường dùng kỹ thuật này là các bộ xử lý Intel i860 và IBM RS/6000. Các bộ xử lý này có khả năng thực hiện song song nhiều tác vụ trên số nguyên và trên số lẻ.

Năm 1992, người ta thấy xuất hiện các bộ xử lý có nhiều bộ thực hiện tác vụ độc lập với nhau (nhiều ALU, bộ tính toán số lẻ, nạp dữ liệu, lưu dữ liệu, nhảy), có thể thực hiện song song nhiều lệnh (lệnh tính số nguyên, số lẻ, lệnh bộ nhớ, lệnh nhảy...). Số lệnh có thể được thi hành song song càng nhiều thì phần cứng thực hiện việc này càng phức tạp.

4.2.8. Lệnh VLIW (VERY LONG INSTRUCTION WORD)

Máy tính siêu vô hướng có thể thực hiện 2 hoặc 3 lệnh trong mỗi chu kỳ xung nhịp. Do kỹ thuật ống dẫn đòi hỏi các lệnh phải phụ thuộc vào nhau nên rất khó thực hiện nhiều lệnh trong một chu kỳ. Như vậy, thay vì cố thực hiện nhiều lệnh trong một chu kỳ, người ta tìm cách đưa vào nhiều lệnh trong một từ lệnh dài. Một lệnh VLIW có thể chứa hai tác vụ tính toán số nguyên, hai tác vụ tính toán số lẻ, hai tác vụ thâm nhập bộ nhớ và một lệnh nhảy. Một lệnh như vậy được chia thành nhiều trường, mỗi trường có thể có từ 16 đến 24 bít và chiều dài của lệnh VLIW là từ 112 đến 168 bít. Có nhiều kỹ thuật tạo ra một lệnh VLIW trong đó tất cả các trường đều được dùng. Giá thành và độ phức tạp của một máy tính có lệnh thật dài tăng lên rất nhiều nếu người ta tăng số trường trong một lệnh VLIW.

4.2.9. Máy tính Vectơ

Một máy tính vectơ bao gồm một bộ tính toán vô hướng bình thường dùng kỹ thuật ống dẫn và một bộ làm tính vectơ. Bộ tính toán vô hướng, giống như bộ xử lý dùng kỹ thuật ống dẫn, thực hiện các phép tính vô hướng, còn bộ làm tính vectơ thực hiện các phép tính vectơ. Đa số các máy tính vectơ cho phép làm các phép tính trên vectơ số nguyên, vectơ số lẻ và vectơ số logic (số Boolean).

Có 2 kiểu kiến trúc máy tính vectơ: kiểu *vectơ ô nhớ - ô nhớ* và *kiểu thanh ghi vectơ*. Trong máy tính loại vectơ bộ nhớ - bộ nhớ, các phép tính vectơ được thực hiện trong bộ nhớ. Kiến trúc kiểu thanh ghi vectơ được thực hiện trong các siêu máy tính CRAY - 1, CRAY - 2, X - MP, Y - MP, trong các siêu máy tính của Nhật NEC SX/2, Fujitsu VP200 và Hitachi S820. Các máy này có một bộ nhiều thanh ghi vectơ và những tác vụ vectơ được thực hiện trên các thanh ghi này ngoại trừ các tác vụ nạp dữ liệu và lưu dữ liệu. Máy CRAY-2 (1995) có 8 thanh ghi vectơ, mỗi thanh ghi có thể chứa 64 vectơ, mỗi vectơ có chiều dài 64 bít.

4.2.10. Máy tính song song

Trong các máy tính siêu ống dẫn, siêu vô hướng, máy tính vectơ, máy tính VLIW, người ta đã dùng tính thực hiện song song các lệnh ở các mức độ khác nhau để làm tăng hiệu quả của chúng. Giới hạn về khả năng tính toán của loại máy trên cùng với sự phát triển của công nghệ máy tính khiến người ta nghĩ tới giải pháp song song theo đó người ta tăng cường hiệu quả của máy tính bằng cách tăng số lượng bộ xử lý.

Các máy tính có thể sắp xếp vào 4 loại sau:

1- **SISD** (Single Instructions Stream, Single Data Stream): Máy tính một dòng lệnh, một dòng số liệu.

2- **SIMD** (Single Instructions Stream, Multiple Data Stream): Máy tính một dòng lệnh, nhiều dòng số liệu.

3- **MISD** (Multiple Instructions Stream, Single Data Stream):Máy tính nhiều dòng lệnh, một dòng số liệu.

4- **MIMD** (Multiple Instruction Stream, Multiple Data Stream): Máy tính nhiều dòng lệnh, nhiều dòng số liệu.

Kiểu phân loại này đơn giản, dễ hiểu, vẫn còn hiệu lực đến hôm nay, mặc dù có những máy tính dùng kiến trúc hỗn tạp.

Các máy tính SISD tương ứng với các máy một bộ xử lý mà chúng ta đã nghiên cứu.

Các máy MISD kiểu máy tính này không sản xuất thương mại.

Các máy SIMD có một số lớn các bộ xử lý giống nhau, cùng thực hiện một lệnh giống nhau để xử lý nhiều dòng dữ liệu khác nhau. Mỗi bộ xử lý có bộ nhớ dữ liệu riêng, nhưng chỉ có một bộ nhớ lệnh và một bộ xử lý điều khiển, bộ này đọc và thi hành các lệnh. Máy CONNECTION MACHINE 2 (65536 bộ xử lý 1 bít) của công ty Thinking Machine Inc, là một ví dụ điển hình của SIMD. Tính song song dùng trong các máy SIMD là tính song song của các dữ liệu. Nó chỉ có hiệu quả nếu cấu trúc các dữ liệu dễ dàng thích ứng với cấu trúc vật lý của các bộ xử lý thành viên. Các bộ xử lý véc-tơ và mảng thuộc loại máy tính này

Các máy MIMD có kiến trúc song song, những năm gần đây, các máy MIMD nổi lên và được xem như một kiến trúc đương nhiên phải chọn cho các máy nhiều bộ xử lý dùng trong các ứng dụng thông thường, một tập hợp các bộ xử lý thực hiện một chuỗi các lệnh khác nhau trên các tập hợp dữ liệu khác nhau. Các máy MIMD hiện tại có thể được xếp vào ba loại hệ thống sẽ được giới thiệu trong phần tiếp theo của chương trình là: *SMP (Symmetric Multiprocesors), Cluster* và *NUMA (Nonunifrom Memory Access)*

a) Một hệ thống SMP bao gồm nhiều bộ xử lý giống nhau được lắp đặt bên trong một máy tính, các bộ xử lý này kết nối với nhau bởi một hệ thống bus bên trong hay một vài sự sắp xếp chuyển mạch thích hợp. Vấn đề lớn nhất trong hệ thống SMP là sự kết hợp các hệ thống cache riêng lẻ. Vì mỗi bộ xử lý trong SMP có một cache riêng của nó, do đó, một khối dữ liệu trong bộ nhớ trong có thể tồn tại trong một hay nhiều cache khác nhau. Nếu một khối dữ liệu trong một cache của một bộ xử lý nào đó bị thay đổi sẽ dẫn đến dữ liệu trong cache của các bộ xử lý còn lại và trong bộ nhớ trong không đồng nhất. Các giao thức cache kết hợp được thiết kế để giải quyết vấn đề này.

b) Trong hệ thống cluster, các máy tính độc lập được kết nối với nhau thông qua một hệ thống kết nối tốc độ cao (mạng tốc độ cao Fast Ethernet hay Gigabit) và hoạt động như một máy tính thống nhất. Mỗi máy trong hệ thống được xem như là một phần của cluster, được gọi là một nút (node). Hệ thống cluster có các ưu điểm:

- Tốc độ cao: Có thể tạo ra một hệ thống cluster có khả năng xử lý mạnh hơn bất cứ một máy tính đơn lẻ nào. Mỗi cluster có thể bao gồm hàng tá máy tính, mỗi máy có nhiều bộ xử lý.
- Khả năng mở rộng cao: có thể nâng cấp, mở rộng một cluster đã được cấu hình và hoạt động ổn định.
- Độ tin cậy cao: Hệ thống vẫn hoạt động ổn định khi có một nút (node) trong hệ thống bị hư hỏng. Trong nhiều hệ thống, khả năng chịu lỗi (fault tolerance) được xử lý tự động bằng phần mềm.
- Chi phí đầu tư thấp: hệ thống cluster có khả năng mạnh hơn một máy tính đơn lẻ mạnh nhất với chi phí thấp hơn.

c) Một hệ thống NUMA (Nonunifrom Memory Access) là hệ thống đa xử lý được giới thiệu trong thời gian gần đây, đây là hệ thống với bộ nhớ chia sẻ, thời gian truy cập các vùng nhớ dành riêng cho các bộ xử lý thì khác nhau. Điều này khác với kiểu quản lý bộ nhớ trong

hệ thống SMP (bộ nhớ dùng chung, thời gian truy cập các vùng nhớ khác nhau trong hệ thống cho các bộ xử lý là như nhau). Hệ thống này có những thuận lợi và bất lợi như sau:

Thuận lợi:

- Thực hiện hiệu quả hơn so với hệ thống SMP trong các xử lý song song.
- Không thay đổi phần mềm chính.
- Bộ nhớ có khả năng bị nghẽn nếu có nhiều truy cập đồng thời, nhưng điều này có thể được khắc phục bằng cách:

+ Cache L1&L2 được thiết kế để giảm tối thiểu tất cả các thâm nhập bộ nhớ.
+ Cần các phần mềm cục bộ được quản lý tốt để việc các ứng dụng hoạt động hiệu quả.
+ Quản trị bộ nhớ ảo sẽ chuyển các trang tới các nút cần dùng.

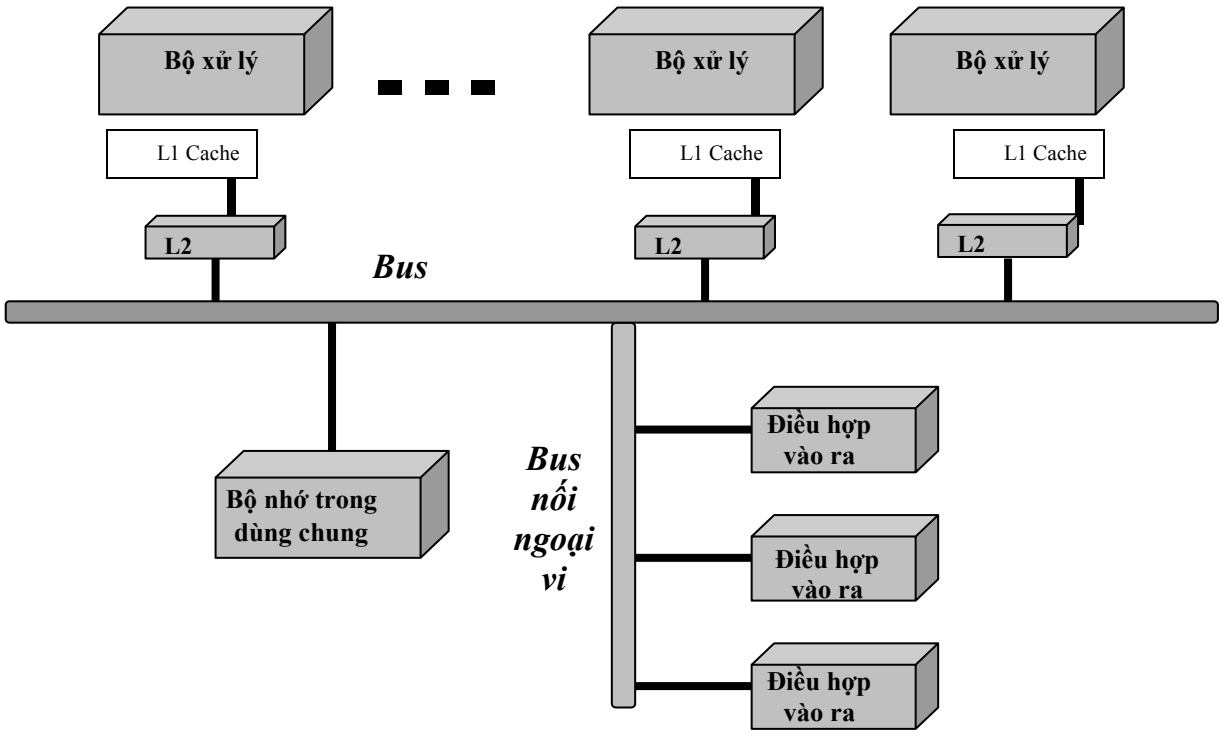
Bất lợi:

- Hệ thống hoạt động không trong suốt như SMP: việc cấp phát các trang, các quá trình có thể được thay đổi bởi các phần mềm hệ thống nếu cần.
- Hệ thống phức tạp.

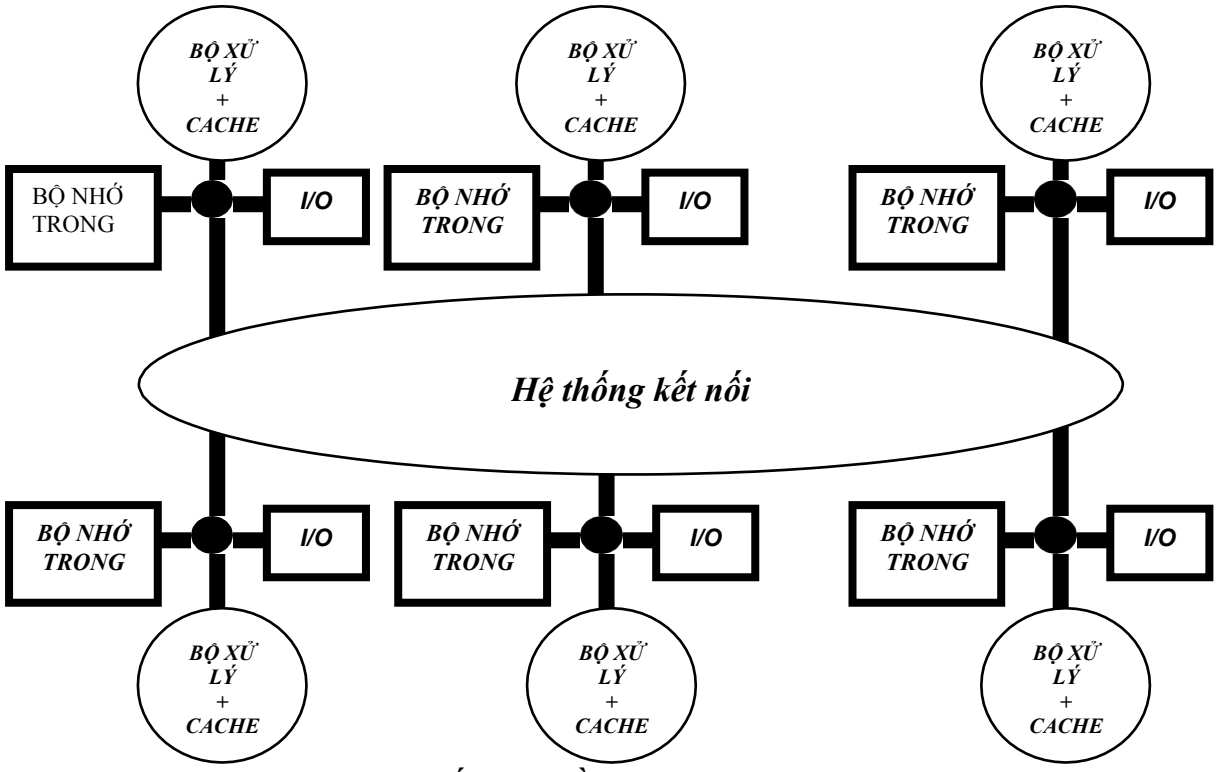
Liên quan đến bộ nhớ trong các máy tính song song, chúng ta có thể chia thành hai nhóm máy:

Nhóm máy thứ nhất, mà ta gọi là máy có kiến trúc bộ nhớ chia sẻ, có một bộ nhớ trung tâm duy nhất được phân chia cho các bộ xử lý và một hệ thống bus chia sẻ để nối các bộ xử lý và bộ nhớ. Vì chỉ có một bộ nhớ trong nên hệ thống bộ nhớ không đủ khả năng đáp ứng nhu cầu thâm nhập bộ nhớ của một số lớn các bộ xử lý. Kiểu kiến trúc bộ nhớ chia sẻ được dùng trong hệ thống SMP.

Nhóm máy thứ hai bao gồm các máy có bộ nhớ phân tán vật lý. Mỗi máy của nhóm này gồm có các nút, mỗi nút chứa một bộ xử lý, bộ nhớ, một vài ngã vào ra và một giao diện với hệ thống kết nối giữa các nút (hình 4-14).



Hình 4-11. Máy tính song song với bộ nhớ dùng chung, hệ thống bus dùng chung



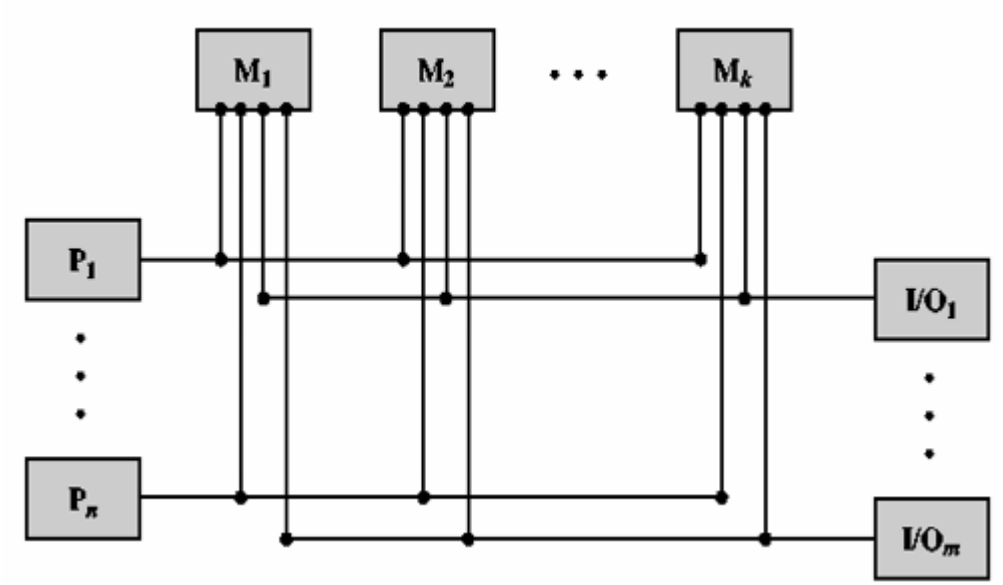
Hình 4-12. Cấu trúc nền của một bộ nhớ phân tán

Việc phân tán bộ nhớ cho các nút có hai điểm lợi. *Trước hết*, đây là một cách phân tán việc thâm nhập bộ nhớ. *Thứ hai*, cách này làm giảm thời gian chờ đợi lúc thâm nhập bộ nhớ cục bộ. Các lợi điểm trên làm cho kiến trúc có bộ nhớ phân tán được dùng cho các máy đa xử lý có một số ít bộ xử lý. Điểm bất lợi chính của kiến trúc máy tính này là việc trao đổi dữ liệu giữa các bộ xử lý trở nên phức tạp hơn và mất nhiều thời gian hơn vì các bộ xử lý không cùng chia sẻ một bộ nhớ trong chung. Cách thực hiện việc trao đổi thông tin giữa bộ xử lý và bộ nhớ trong, và kiến trúc logic của bộ nhớ phân tán là một tính chất đặc thù của các máy tính với bộ nhớ phân tán.

Có 2 phương pháp được dùng để truyền dữ liệu giữa các bộ xử lý.

Phương pháp thứ nhất là các bộ nhớ được phân chia một cách vật lý có thể được thâm nhập với một định vị chia sẻ một cách logic, nghĩa là nếu một bộ xử lý bất kỳ có quyền truy xuất, thì nó có thể truy xuất bất kỳ ô nhớ nào. Trong phương pháp này các máy được gọi có kiến trúc bộ nhớ chia sẻ phân tán (DSM: Distributed Sharing Memory). Từ bộ nhớ chia sẻ cho biết không gian định vị bị chia sẻ. Nghĩa là cùng một địa chỉ vật lý cho 2 bộ xử ý tường ứng với cùng một ô nhớ.

Phương pháp thứ hai, không gian định vị bao gồm nhiều không gian định vị nhỏ không giao nhau và có thể được một bộ xử lý thâm nhập. Trong phương pháp này, một địa chỉ vật lý gắn với 2 máy khác nhau thì tương ứng với 2 ô nhớ khác nhau trong 2 bộ nhớ khác nhau. Mỗi mô-đun bộ *xử lý-bộ nhớ* thì cơ bản là một máy tính riêng biệt và các máy này được gọi là đa máy tính. Các máy này có thể gồm nhiều máy tính hoàn toàn riêng biệt và được nối vào nhau thành một mạng cục bộ.



Hình 4-13. Tổ chức kết nối của máy tính song song có bộ nhớ phân tán

Kiến trúc song song phát triển mạnh trong thời gian gần đây do các lý do:

Việc dùng xử lý song song đặc biệt trong lãnh vực tính toán khoa học và công nghệ. Trong các lãnh vực này người ta luôn cần đến máy tính có tính năng cao hơn.

Người ta đã chấp nhận rằng một trong những cách hiệu quả nhất để chế tạo máy tính có tính năng cao hơn các máy đơn xử lý là chế tạo các máy tính đa xử lý.

Máy tính đa xử lý rất hiệu quả khi dùng cho đa chương trình. Đa chương trình được dùng chủ yếu cho các máy tính lớn và cho các máy phục vụ lớn. Các ví dụ về các siêu máy tính dùng kỹ thuật xử lý song song:

Máy điện toán Blue Gene/L của IBM đang được đặt tại Phòng thí nghiệm Lawrence Livermore, và đứng đầu trong số 500 siêu máy tính mạnh nhất thế giới. Siêu máy tính Blue Gene/L sẽ được sử dụng cho các công việc "phi truyền thống", chủ yếu là giả lập và mô phỏng các quá trình sinh học và nguyên tử. Máy điện toán Blue Gene/L đã đạt tốc độ hơn 70 teraflop (nghìn tỷ phép tính/giây). Kết quả này có thể sẽ đưa cỗ máy lên vị trí dẫn đầu trong danh sách các siêu máy tính nhanh nhất thế giới, được công bố ngày 8/11/2004. Theo đó, siêu máy tính do IBM lắp ráp đã đạt tốc độ 70,72 teraflop trong các cuộc thử nghiệm hồi tháng 10/2004. IBM nghiên cứu và phát triển Blue Gene với mục đích thử nghiệm nhằm tạo ra các hệ thống cực mạnh nhưng chiếm ít không gian và tiêu thụ ít năng lượng. IBM dự kiến, sẽ lắp đặt cho phòng thí nghiệm quốc gia Lawrence Livermore một siêu máy tính có tốc độ nhanh gấp 4 lần so với kỷ lục vừa đạt được. Khi đó, thiết bị sẽ được ứng dụng vào nhiều nghiên cứu khoa học. Hệ thống mới bao gồm 16,384 giao điểm điện toán kết nối *32.768 bộ xử lý*.

Thông tin mới nhất (02/2005) cho biết: siêu máy tính IBM Blue Gene/L vừa thiết lập kỷ lục mới đó là có khả năng xử lý 135,5 nghìn tỷ phép tính/giây (135,3 teraflop), vượt xa kỷ lục 70,72 teraflop do chính siêu máy tính này lập nên. Số bộ xử lý (BXL) của Blue Gene/L vừa được các nhà khoa học tăng lên gấp đôi (64.000 BXL) nhằm tăng cường khả năng tính toán cho siêu máy tính này. Cũng cần phải nhắc lại rằng thiết kế hoàn thiện của siêu máy tính Blue Gene/L, dự kiến sẽ hoàn tất vào khoảng tháng 6 tới, sẽ bao gồm 130.000 BXL với tốc độ tính toán được kỳ vọng vào khoảng 360 teraflop.

Blue Gene là tên gọi chung cho dự án nghiên cứu siêu máy tính được IBM khởi động từ năm 2000, với mục đích ban đầu là thiết kế một "cỗ máy" có khả năng xử lý 1 teraflop. Trong khi đó, siêu máy tính Blue Gene/L là một trong nhiều sản phẩm chủ lực của IBM nhằm cạnh tranh với các hãng đối thủ Silicon Graphics và NEC.

Hãng điện tử khổng lồ NEC phát hành một supercomputer dạng vector, máy SX-8 mới ra đời có tốc độ xử lý cực đại lên tới 65 teraflop (65 nghìn tỷ phép tính dấu phẩy động/giây) và khả năng hoạt động ổn định ở mức xấp xỉ 90% của tốc độ 58,5% teraflop. Máy SX-8 có kiến trúc khác hẳn Blue Gene/L của IBM. Nó dùng kiến trúc vector nên đem đến độ ổn định khi hoạt động cao hơn nhiều so với dạng máy tính vô hướng (scalar) như của IBM

Một hệ thống tại trung tâm nghiên cứu của Cơ quan hàng không vũ trụ Mỹ (NASA) tại California cũng đạt được tốc độ 42,7 teraflop. Với tên gọi Columbia, siêu máy tính này sẽ được sử dụng để nghiên cứu khí tượng và thiết kế máy bay. Hệ thống trị giá 50 triệu USD (thời điểm tháng 10/2004) này sử dụng phần mềm Linux và đã được SGI ký hợp đồng bán cho Cơ quan hàng không vũ trụ Mỹ NASA. Nó có thể thực hiện 42,7 nghìn tỷ phép tính/giây (42,7 teraflop). Tuy nhiên, tốc độ đó chưa phải là tất cả những gì nổi bật của siêu máy tính này: hệ thống mới chỉ khai thác có 4/5 công suất của *10.240 bộ xử lý Intel Itanium 2* trong toàn bộ cỗ máy đặt ở trung tâm nghiên cứu của NASA ở California (Mỹ). Siêu máy tính này không giống với hầu hết các siêu máy tính hiện nay thường được tạo nên theo kiểu cluster, với sự tham gia của nhiều cỗ máy giá rẻ. Columbia được thiết lập từ 20 máy tính mà mỗi chiếc có 512 bộ xử lý, kết nối bằng công nghệ mạng cao tốc và đều chạy một hệ điều hành độc lập. Cách xây dựng này rất hữu ích cho những công việc như giả lập các yếu tố khí động lực cho tàu không gian. Một ứng dụng khác của siêu máy tính Columbia là việc dự báo bão. Phần mềm cho tác vụ này đang được thiết kế và hứa hẹn khả năng dự báo chính xác đường đi của bão sớm 5 ngày. Toàn bộ máy Columbia chiếm dụng một diện tích bằng khoảng 3 sân bóng rổ.

4.2.11. Kiến trúc IA-64

Kiến trúc IA-64 là một kiến trúc mới được giới thiệu trong những năm gần đây. Kiến trúc này là sản phẩm của sự kết hợp nghiên cứu giữa hai công ty máy tính hàng đầu thế giới là Intel, HP (Hewlett Packard) và một số trường đại học. Kiến trúc mới dựa trên sự phát triển của công nghệ mạch tích hợp và kỹ thuật xử lý song song. Kiến trúc IA-64 giới thiệu một sự khởi đầu mới quan trọng của kỹ thuật siêu vô hướng - kỹ thuật xử lý lệnh song song (EPIC: Expicitly Parallel Intruction Computing) - kỹ thuật ảnh hưởng nhiều đến sự phát triển của bộ xử lý hiện nay. Sản phẩm đầu tiên thuộc kiến trúc này là bộ xử lý ***Itanium***.

4.2.11.1. Đặc trưng của kiến trúc IA-64:

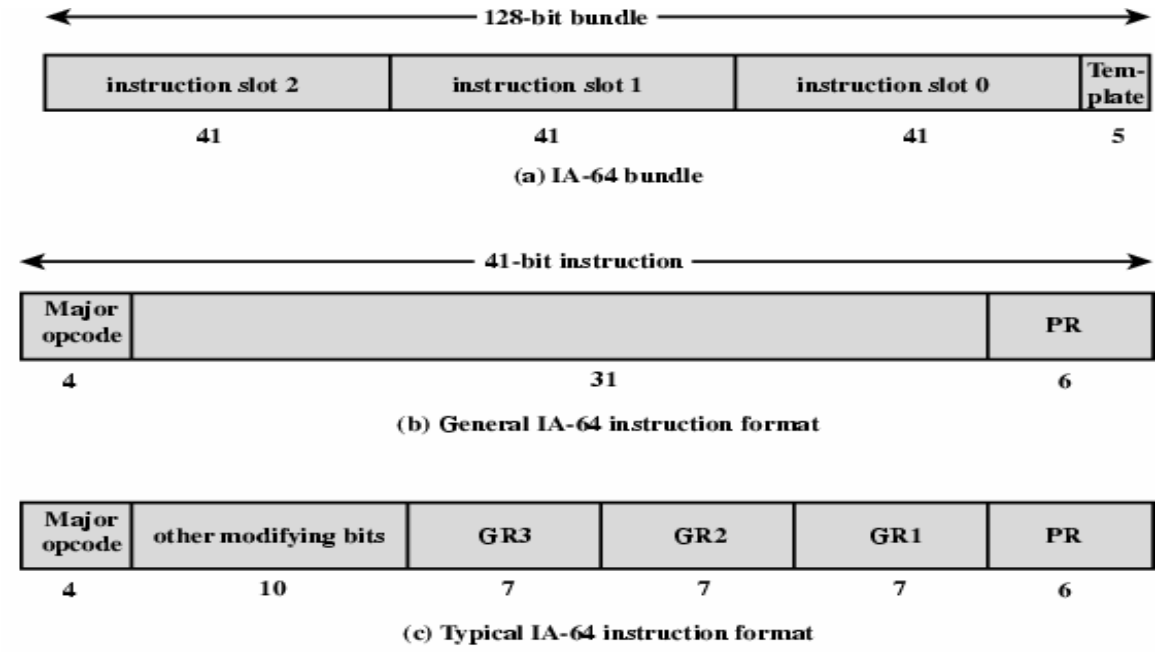
- Cơ chế xử lý song song là song song các lệnh mã máy (EPIC) thay vì các bộ xử lý song song như hệ thống đa bộ xử lý.
- Các lệnh dài hay rất dài (LIW hay VLIW).
- Các lệnh rẽ nhánh xác định (thay vì đoán các lệnh rẽ nhánh như các kiến trúc trước).
- Nạp trước các lệnh (theo sự suy đoán).
- Các đặc trưng của tổ chức của bộ xử lý theo kiến trúc IA-64:
- Có nhiều thanh ghi: số lượng thanh ghi các bộ xử lý kiến trúc IA-64 là 256 thanh ghi. Trong đó, 128 thanh ghi tổng quát (GR) 64 bit cho các tính toán số nguyên, luận lý; 128 thanh ghi 82 bit (FR) cho các phép tính dấu chấm động và dữ liệu đồ hoạ; ngoài ra, còn có 64 thanh ghi thuộc tính (PR)1 bit để chỉ ra các thuộc tính lệnh đang thi hành.
- Nhiều bộ thi hành lệnh: hiện nay, một máy tính có thể có tám hay nhiều hơn các bộ thi hành lệnh song song. Các bộ thi hành lệnh này được chia thành bốn kiểu:

+ Kiểu I (I-Unit): dùng xử lý các lệnh tính toán số nguyên, dịch, luận lý, so sánh, đa phương tiện.

+ Kiểu M (M-Unit): Nạp và lưu trữ giữa thanh ghi và bộ nhớ thêm vào một vài tác vụ ALU.

+ Kiểu B (B-Unit): Thực hiện các lệnh rẽ nhánh.
+ Kiểu F (F-Unit): Các lệnh tính toán số dấu chấm động

Định dạng lệnh trong kiến trúc IA-64



PR: Predicate register
GR: General hay Floating-point

Hình 4-14. Định dạng lệnh trong kiến trúc IA-64

Kiến trúc IA-64 định nghĩa một *gói* (buldle) 128 bit chứa ba lệnh (mỗi lệnh dài 41 bit) và một trường mẫu (template field) 5 bit. Bộ xử lý có thể lấy một hay nhiều *gói lệnh* thi hành cùng lúc. Trường mẫu (template field) này chứa các thông tin chỉ ra các lệnh có thể thực hiện song song (Bảng 4-1.). Các lệnh trong một bó có thể là các lệnh độc lập nhau. Bộ biên dịch sẽ sắp xếp lại các lệnh trong các gói lệnh kề nhau theo một thứ tự để các lệnh có thể được thực hiện song song. Hình 4-14a chỉ ra định dạng lệnh trong kiến trúc IA-64. Hình 4-14b mô tả dạng tổng quát của một lệnh trong gói lệnh. Trong một lệnh, mã lệnh chỉ có 4 bit chỉ ra 16 khả năng có thể để thi thi hành một lệnh và 6 bit chỉ ra thanh ghi thuộc tính được dùng với lệnh. Tuy nhiên, các mã tác vụ này còn tuỳ thuộc vào vị trí của lệnh bên trong gói lệnh, vì vậy khả năng thi hành của lệnh nhiều hơn số mã tác vụ được chỉ ra. Hình 4-14c mô tả chi tiết các trường trong một lệnh (41 bit). Trong bảng Bảng 4-1 , các kiểu L-Unit, X-Unit là các kiểu mở rộng, có thể thực hiện lệnh bởi I-Unit hay B-Unit.

Bảng 4-1. Bảng mã hoá tập hợp các ánh xạ trong trường mẫu.

Template	*Slot 0*	*Slot 1*	*Slot 2*
00	M-Unit	I-Unit	I-Unit
01	M-Unit	I-Unit	I-Unit
02	M-Unit	I-Unit	I-Unit
03	M-Unit	I-Unit	I-Unit
04	M-Unit	L-Unit	X-Unit
05	M-Unit	L-Unit	X-Unit
08	M-Unit	M-Unit	I-Unit
09	M-Unit	M-Unit	I-Unit
0A	M-Unit	M-Unit	I-Unit

0B	M-Unit	M-Unit	I-Unit
0C	M-Unit	F-Unit	I-Unit
0D	M-Unit	F-Unit	I-Unit
0E	M-Unit	M-Unit	F-Unit
0F	M-Unit	M-Unit	F-Unit
10	M-Unit	I-Unit	B-Unit
11	M-Unit	I-Unit	B-Unit
12	M-Unit	B-Unit	B-Unit
13	M-Unit	B-Unit	B-Unit
16	B-Unit	B-Unit	B-Unit
17	B-Unit	B-Unit	B-Unit
18	M-Unit	M-Unit	B-Unit
19	M-Unit	M-Unit	B-Unit
1C	M-Unit	F-Unit	B-Unit
1D	M-Unit	F-Unit	B-Unit

4.3. KIẾN TRÚC TẬP LỆNH

4.3.1. Các kiểu toán hạng

Kiểu của toán hạng thường được đưa vào trong mã tác vụ của lệnh. Có bốn kiểu toán hạng được dùng trong các hệ thống:

- Kiểu địa chỉ.
- Kiểu dạng số: số nguyên, dấu chấm động,...
- Kiểu dạng chuỗi ký tự: ASCII, EBIDEC,...
- Kiểu dữ liệu logic: các bit, cờ,...

Tuy nhiên một số ít máy tính dùng các nhãn để xác định kiểu toán hạng. Thông thường loại của toán hạng xác định luôn chiều dài của nó. Toán hạng thường có chiều dài là byte (8 bit), nửa từ máy tính (16 bit), từ máy tính (32 bit), từ đôi máy tính (64 bit). Đặc biệt, kiến trúc PA của hãng HP (Hewlet Packard) có khả năng tính toán với các số thập phân BCD. Một vài bộ xử lý có thể xử lý các chuỗi ký tự.

4.3.1.1. Số lượng địa chỉ toán hạng trong lệnh

- 0 địa chỉ toán hạng:

Các toán hạng đều được ngầm định.

Sử dụng toán hạng Stack.

Ví dụ 4.3.1*:* push a
push b
add
poc c

Có nghĩa là: c= a+b

Nhưng kiểu này không thông dụng.

- 1 địa chỉ toán hạng

Một toán hạng được chỉ ra trong lệnh.

Một toán hạng ngầm định, thường là thanh ghi (thanh chứa - accumulator)

Được sủ dụng trên các máy của thế hệ trước.

- 2 địa chỉ toán hạng

Một toán hạng vừa là toán hạng nguồn vừa là toán hạng đích, toán hạng còn lại là toán hạng nguồn.

a= a+b

Giá trị cũ của một toán hạng nguồn bị mất do phải chứa kết quả.

Ưu điểm: Rút gọn độ dài từ lệnh và phổ biến.

-3 địa chỉ toán hạng

Hai toán hạng nguồn và một toán hạng đích.

c= a+b

Từ lệnh dài vì phải mã hóa địa chỉ cho cả ba toán hạng.

Bảng 4-2. Số lượng địa chỉ toán hạng trong lệnh

Số toán hạng	Dạng lệnh	Chức năng	Ghi chú
3	PT A,B,C	A ←B PT C	
2	PT A,B	A←A PT B	
1	PT A	AC←AC PT A	
0	PT	S-1 ←(S-2) PT (S-1)	Làm việc với ngăn xếp

Được sử dụng trên các bộ vi xử lý tiên tiến.

Một số lệnh ví dụ

Bảng 4-3. Số lượng địa chỉ toán hạng trong lệnh

Không địa chỉ	Một địa chỉ	Hai địa chỉ	Ba địa chỉ
PUSH M	LOAD M	MOV A,B	ADD A,B,C
POP M	STOR M	ADD A,B	SUB A,B,C
ADD	ADD M	SUB A,B	MPY A,B,C
SUB	SUB M	MPY A,B	DIV A,B,C
MPY	MPY M	DIV A,B	
DIV	DIV M		

Chú ý: Trong một lệnh tối thiểu phải có một toán hạng là thanh ghi

***Ví dụ 4.3.2*:** Thực hiện phép toán sau.

f = (A + B/D) (C – D/E) + E/F

Dùng các lệnh 3, 2, 1, 0 địa chỉ

Với lệnh 3 địa chỉ

DIV R1,B,D	;	R1 ← B/D
ADD R1,R1,A	;	R1 ← A + B/D

DIV R2,D,E	;	R2 ← D/E
SUB R2,C,R2	;	R2 ← C-D/E
MPY R1,R1,R2	;	R1 ← (A+B/D)(C-D/E)
DIV R2,E,F	;	R2 ← E/F
ADD R1,R2,R1	;	R1 ← (A+B/D)(C-D/E)+E/F

Với lệnh 2 địa chỉ

MOV R1,B	;	R1 ← B
DIV R1,D	;	R1 ← B/D
ADD R1,A	;	R1 ← A+B/D
MOV R2,D	;	R2 ← D
DIV R2,E	;	R2 ← D/E
SUB C,R2	;	C ← C-D/E (ô nhớ chứa C)
MPY R1,C	;	R1 ← (A+B/D)(C-D/E)
MOV R2,E	;	R2← E
DIV R2,F	;	R2 ← E/F
ADD R1,R1	;	R1 ← (A+B/D)(C-D/E)+E/F

Với lệnh 1 địa chỉ

LOAD B	;	**AC ← B**
DIV D	;	AC ← B/D
ADD A	;	AC ← A+B/D
STOR M1	;	M1 ← A+B/D
LOAD D	;	AC ← D
DIV E	;	AC ←D/E
STOR M2	;	M2 ← D/E
LOAD C	;	AC ← C
SUB M2	;	AC ← C-D/E
MPY M1	;	AC ← (A+B/D)(C-D/E)
STOR M1	;	M1 ← (A+B/D)(C-D/E)
LOAD E	;	AC ← E

DIV F	;	AC ← E/F
ADD M1	;	AC ← (A+B/D)(C-D/E)+E/F

Với lệnh 0 địa chỉ

PUSH D	;	S – 1 ← D
PUSH B	;	S – 2 ← B
DIV	;	S – 1 ← B/D
PUSH A	;	S – 2 ← A
ADD	;	S – 1 ← A+B/D
PUSH E	;	S – 2 ← E
PUSH D	;	S – 3 ← D
DIV	;	S – 2 ← D/E
PUSH C	;	S – 3 ← C
SUB	;	S – 2 ← (C-D/E)
MPY	;	S – 1 ← (A+B/D)(C-D/E)
PUSH F	;	S – 2 ← F
PUSH E	;	S – 3 ← F
DIV	;	S – 2 ← E/F
ADD	;	S – 1 ← (A+B/D)(C-D/E)+E/F

4.3.1.2. Đánh giá về số địa chỉ toán hạng

a) Nhiều địa chỉ toán hạng: Các lệnh phức tạp hơn

- Cần nhiều thanh ghi hơn.
- Chương trình có ít lệnh hơn.
- Nhận lệnh và thực hiện lệnh chậm hơn.

b) Ít địa chỉ toán hạng: Các lệnh đơn giản hơn

- Cần ít thanh ghi hơn
- Chương trình có nhiều lệnh hơn
- Nhận lệnh và thực hiện lệnh nhanh hơn.

4.3.2. Tập lệnh

4.3.2.1. Các lệnh chuyển dữ liệu

MOVE: Copy dữ liệu từ nguồn tới đích.
LOAD: Nạp dữ liệu từ bộ nhớ tới bộ xử lý.
STORE: Cất dữ liệu từ bộ xử lý đến bộ nhớ.
EXTRANGE: Trao đổi nội dung của nguồn và đích.
CLEAR: Chuyển các bit 0 vào toán hạng đích.
SET: Chuyển các bít 1 vào toán hạng đích.

PUSH: Cất nội dung toán hạng nguồn vào ngăn sếp.

POP: Lấy nội dung đỉnh ngăn sếp đưa đến toán hạng đích

4.3.2.2. Các lệnh số học

ADD: Cộng hai toán hạng.

SUBTRACT: Trừ hai toán hạng.

MULTIPLY: Nhân hai toán hạng.

DIVIDE: Chia hai toán hạng.

ABSOLUTE: Lấy trị tuyệt đối hai toán hạng

NEGATE: Đổi dấu toán hạng (Lấy bù 2)

INCREMENT: Tăng toán hạng thêm 1

DECREMENT: Giảm toán hạng đi 1

COMPARE: Trừ hai toán hạng để lập cờ

4.3.2.3. Các lệnh logic

AND: Thực hiện phép AND hai toán hạng

OR: Thực hiện phép OR hai toán hạng

XOR: Thực hiện phép XOR hai toán hạng

NOT: Đảo bít hai toán hạng (lấy bù 1)

TEST: Thực hiện phép AND hai toán hạng để lập cờ

SHIFT: Dịch trái (phải) toán hạng

ROTATE: Quay trái (phải) toán hạng

Minh họa các lệnh AND, OR, XOR

Giả sử có hai thanh ghi chứa dữ liệu như sau:

(R1) = 1010 1010

(R2) = 0000 1111

R1← (R1) AND (R2) = 0000 1010

Phép toán AND dùng để xóa một số bit và giữ nguyên một số bit còn lại của toán hạng.

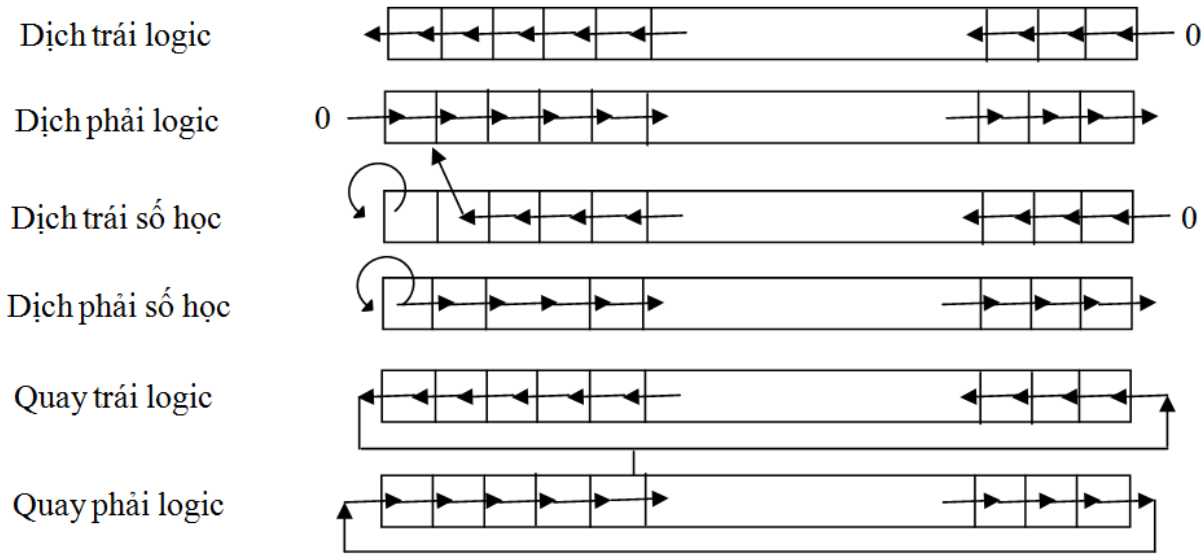
R1 ← (R1) OR (R2) = 1010 1111

Phép toán OR dùng để thiết lập một số bit và giữ nguyên một số bit của toán hạng.

R1 ← (R1) XOR (R2) = 1010 0101

Phép toán XOR dùng để đảo một số bít và giữ nguyên một số bít còn lại của toán hạng.

Các thao tác SHIFT và ROTATE:



Hình 4-15. Các thao tác SHIFT và ROTATE

4.3.2.4. Các lệnh vào ra chuyên dụng

INPUT: Copy dữ liệu từ một cổng xác định đưa tới đích.

OUTPUT: Copy dữ liệu từ nguồn tới một cổng xác định.

4.3.2.5. Các lệnh chuyển điều kiện

JUMP (BRANCH): Lệnh nhảy không điều kiện. Nạp vào PC một địa chỉ xác định.

JUMP CONDITIONAL: Lệnh nhảy có điều kiện. Điều kiện đúng -> nạp vào PC một địa chỉ xác định. Điều kiện sai -> Không làm gì cả.

CALL: Lệnh gọi chương trình con. Cất nội dung của PC (địa chỉ trở về) ra một vị trí xác định (thường ở Stack). Nạp vào PC địa chỉ của lệnh đầu tiên của chương trình con.

RETURN: Lệnh trở về từ chương trình con. Khôi phục địa chỉ trở về trả lại cho PC để trở về chương trình chính

4.3.2.6. Lệnh rẽ nhánh

- Lệnh rẽ nhánh không điều kiện.

Chuyển tới thực hiện lệnh ở vị trí có địa chỉ XXX. PC không trỏ sang lệnh kế tiếp mà nhảy xuống lệnh có địa chỉ XXX

PC ⟵ XXX

- Lệnh rẽ nhánh có điều kiện.

Trong lệnh có kèm theo điều kiện

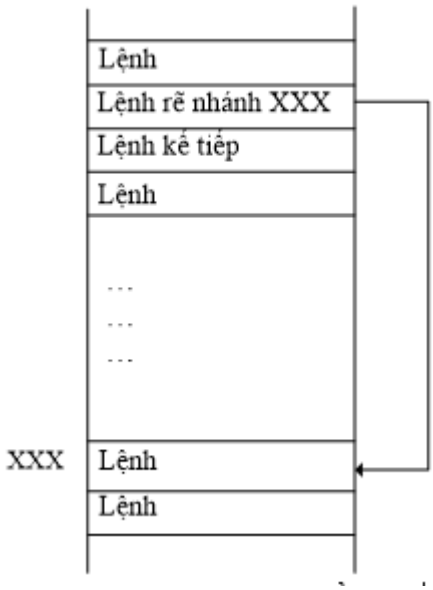
Kiểm tra điều kiện trong lệnh:

Nếu lệnh đúng -> thực hiện lệnh ở vị trí có địa chỉ XXX

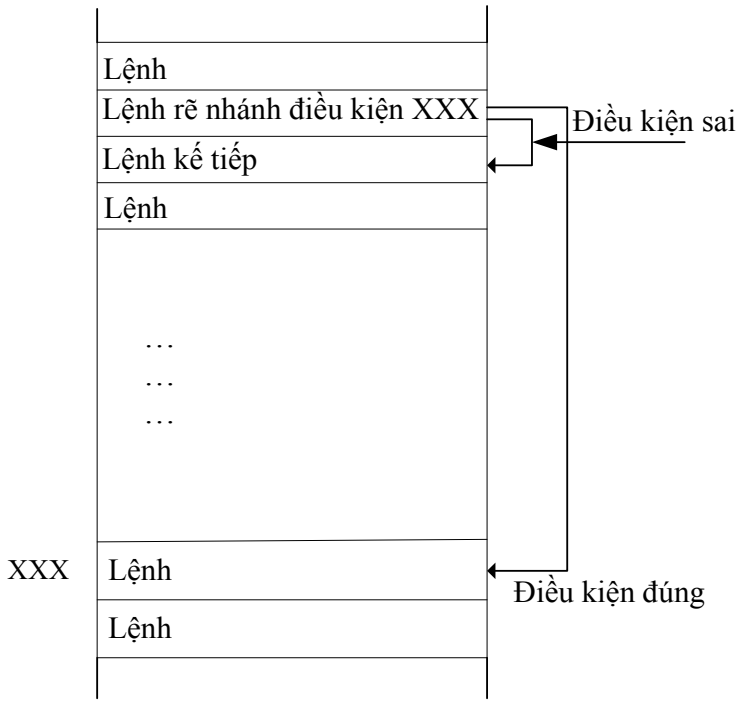
Nếu điều kiện sai -> thực hiện lệnh_kế_tiếp.

Điều kiện thường được kiểm tra thông qua các cờ.

Có nhiều lệnh rẽ nhánh có điều kiện.



Hình 4-16. Lệnh rẽ nhánh không



Hình 4-17. Lệnh rẽ nhánh có điều kiện

4.3.2.7. Lệnh CALL và RETURN

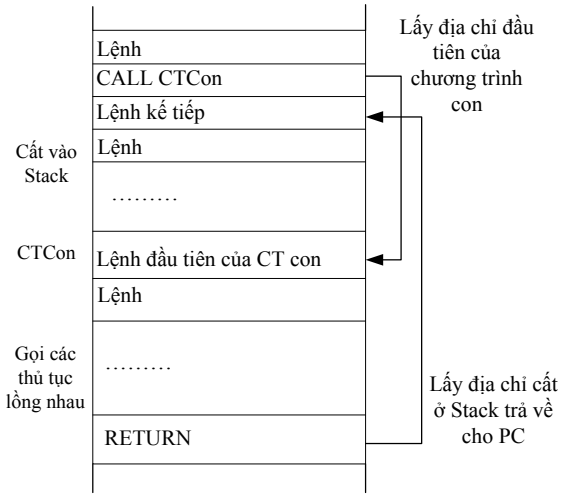
- Lệnh gọi chương trình con CALL.

Cất nội dung PC (chứa địa chỉ của lệnh kế tiếp) ra Stack.

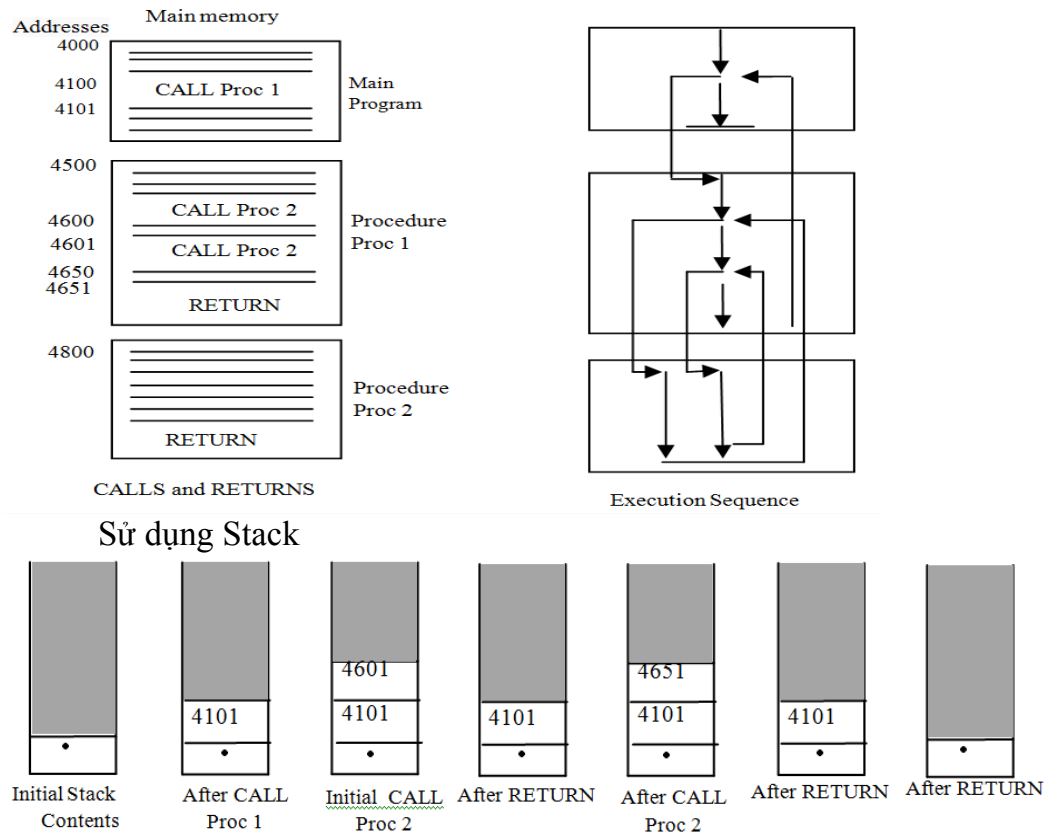
Nạp vào PC địa chỉ của lệnh đầu tiên của chương trình con được gọi -> Bộ xử lý được chuyển sang thực hiện chương trình con tương ứng.

Lệnh trở về chương trình con RETURN.

Lấy địa chỉ của lệnh kế tiếp cất ở Stack nạp trả lại cho PC -> Bộ xử lý được điều khiển quay trở về thực hiện lệnh nằm sau lệnh CALL.



Hình 4-18a. Lệnh CALL và RETURN



Hình 4-18b. Lệnh CALL và RETURN

4.3.2.8. Các lệnh điều khiển hệ thống

HALT: Dừng thực hiện chương trình

WAIT: Tạm dừng thực hiện chương trình, lặp kiểm tra điều kiện cho đến khi thỏa mãn thì tiếp tục thực hiện.

NO OPERATION: Không thực hiện gì cả.

LOCK: Cấm không cho xin chuyển nhượng BUS

UNLOCK: Cho phép xin chuyển nhượng BUS.

4.4. NGÔN NGỮ LẬP TRÌNH VÀ CHƯƠNG TRÌNH DỊCH

4.4.1. Khái niệm ngôn ngữ lập trình

Ngôn ngữ lập trình (Programming language) là một tập con của ngôn ngữ máy tính. Đây là một dạng ngôn ngữ được chuẩn hóa (đối lập với ngôn ngữ tự nhiên). Nó được dùng để miêu tả những quá trình, những ngữ cảnh một cách rất chi tiết. Nói cách khác, ngôn ngữ lập trình là một hệ thống được ký hiệu hóa để miêu tả những tính toán (qua máy tính) trong một dạng mà cả con người và máy đều có thể đọc và hiểu được.

4.4.2. Các loại ngôn ngữ lập trình thông dụng

Có hàng trăm loại ngôn ngữ lập trình khác nhau, mỗi loại ngôn ngữ đều có cú pháp riêng của nó. Một số ngôn ngữ thì được phát triển để dùng trên các loại máy tính chuyên biệt, một số ngôn ngữ khác thì - do sự thành công của nó - đã trở thành chuẩn và được áp dụng trên đa số các máy tính. Ngôn ngữ lập trình có thể được phân chia thành 3 loại chính: Ngôn ngữ máy, hợp ngữ và ngôn ngữ cấp cao.

4.4.2.1. Ngôn ngữ máy

Ngôn ngữ máy (mã máy) là ngôn ngữ nền tảng của bộ vi xử lý. Các chương trình được viết trong tất cả các loại ngôn ngữ khác cuối cùng đều được chuyển thành ngôn

ngữ máy trước khi chương trình đó được thi hành. Vì tập lệnh của ngôn ngữ máy phụ thuộc vào loại vi xử lý nên ngôn ngữ máy sẽ khác nhau trên những máy tính có sử dụng bộ vi xử lý khác nhau. Lợi điểm của viết chương trình bằng ngôn ngữ máy là lập trình viên có thể điều khiển máy tính trực tiếp và đạt được chính xác điều mình muốn làm. Do đó, các chương trình ngôn ngữ máy được viết tốt là những chương trình rất hiệu quả (tốc độ thi hành nhanh, kích thước nhỏ). Bất lợi của chương trình ngôn ngữ máy là thông thường sẽ mất rất nhiều thời gian để viết, rất khó đọc, theo dõi để tìm lỗi. Thêm vào đó, bởi vì chương trình được viết bằng tập lệnh phụ thuộc vào bộ vi xử lý nên chương trình chỉ chạy được trên những máy tính có cùng bộ vi xử lý mà thôi. Ngôn ngữ máy cũng được gọi là ngôn ngữ cấp thấp *(low-level language)*

4.4.2.2. Hợp ngữ

Hợp ngữ được phát triển nhằm giúp các lập trình viên dễ nhớ các chỉ thị của chương trình hơn. Hợp ngữ tương tự như ngôn ngữ máy nhưng lại sử dụng các ký hiệu gợi nhớ (mnemonics hay mã lệnh hình thức - symbolic operation code) để biểu diễn cho các mã lệnh của máy. Một đặc điểm khác nữa là hợp ngữ thông thường cho phép định địa chỉ hình thức (symbolic addressing), nghĩa là một vị trí bộ nhớ trong máy tính có thể được tham chiếu tới thông qua một cái tên hoặc ký hiệu, chẳng hạn như TOTAL thay vì phải sử dụng địa chỉ thực sự của nó (bằng con số nhị phân) trong ngôn ngữ máy. Các chương trình hợp ngữ còn bao gồm các chỉ thị vĩ mô *(macro instruction)* có thể tạo ra nhiều lệnh mã máy. Các chương trình hợp ngữ được chuyển sang mã máy thông qua một chương trình đặc biệt gọi là trình hợp dịch (assembler). Mặc dù hợp ngữ tương đối dễ dùng hơn mã máy nhưng hợp ngữ vẫn được xem là ngôn ngữ cấp thấp bởi vì nó vẫn còn rất gần với từng thiết kế của máy tính.

Nói chung một trình hợp ngữ được chia làm 4 đoạn:

- Đoạn mã
- Đoạn dữ liệu
- Đoạn dữ liệu mở rộng
- Đoạn ngăn xếp

Trong chương trình đòi hỏi phải có ít nhất một đoạn ngăn xếp. Đoạn ngăn xếp giúp lưu trữ các kết quả trung gian khi thực hiện chương trình.

4.4.2.3. Ngôn ngữ cấp cao

Cuộc cách mạng của ngôn ngữ máy tính bắt đầu với sự phát triển của ngôn ngữ cấp cao vào cuối thập kỷ 1950 và 1960. Ngôn ngữ cấp cao gần gũi hơn với ý niệm ngôn ngữ mà hầu hết mọi người đều biết, nó bao gồm các danh từ, động từ, ký hiệu toán học, liên hệ và các thao tác luận lý. Các yếu tố này có thể được phối hợp, liên kết với nhau tạo thành một hình thức của câu. Các "câu" này được gọi là các mệnh đề của chương trình (program statement). Chính vì những đặc điểm này, các lập trình viên dễ dàng đọc và dễ học ngôn ngữ cấp cao hơn so với ngôn ngữ máy hoặc hợp ngữ. Một lợi điểm quan trọng là ngôn ngữ cấp cao thông thường không phụ thuộc vào máy tính, nghĩa là các chương trình viết bằng ngôn ngữ cấp cao có thể chạy trên các loại máy tính khác nhau (sử dụng các bộ vi xử lý khác nhau).

Ngôn ngữ cấp cao như: C, C++, PASCAL, BASIC, COBOL, FORTRAN,...

4.4.3. Chương trình dịch

Mọi chương trình được viết bằng các ngôn ngữ không phải là ngôn ngữ máy cuối cùng đều phải được chuyển đổi sang ngôn ngữ máy trước khi được thi hành. Chương

trình ngôn ngữ cấp cao được dịch sang ngôn ngữ máy bằng một trong hai cách: Trình biên dịch (compiler) hoặc trình thông dịch (interpreter).

4.4.3.1. Trình biên dịch

Sẽ chuyển đổi toàn bộ chương trình sang mã máy, rồi chứa kết quả vào đĩa để có thể thi hành về sau. Chương trình ngôn ngữ cấp cao được chuyển đổi được gọi là chương trình nguồn (source program) và chương trình ngôn ngữ máy được tạo ra được gọi là chương trình đối tượng (object program) hoặc mã đối tượng (object code). Khi người dùng muốn chạy chương trình, chương trình đối tượng sẽ được nạp lên bộ nhớ chính của CPU và các chỉ thị của chương trình sẽ được thi hành. Khi được hướng dẫn bởi các chỉ thị của chương trình, CPU sẽ truy xuất dữ liệu và tạo ra các kết quả. Trình biên dịch sẽ kiểm tra cú pháp chương trình, thực hiện các phép kiểm tra logic và đảm bảo các dữ liệu sắp được sử dụng trong các phép so sánh, tính toán đã được định nghĩa một cách hợp lý ở một nơi nào đó trong chương trình. Một chức năng quan trọng của trình biên dịch là nó sẽ tạo ra một danh sách lỗi của tất cả mệnh đề trong chương trình vi phạm cú pháp của ngôn ngữ. Danh sách này giúp lập trình viên dễ dàng sửa đổi chương trình.

Do ngôn ngữ máy phụ thuộc vào bộ vi xử lý nên các máy tính khác nhau sẽ cần có các trình biên dịch khác nhau đối với cùng một ngôn ngữ cấp cao. Ví dụ, một máy mainframe, máy mini và máy tính cá nhân cần có các trình biên dịch khác nhau để biên dịch cùng một chương trình nguồn sang mã máy của từng loại máy này.

4.4.3.2. Trình thông dịch

Thay vì chuyển đổi toàn bộ chương trình nguồn như trình biên dịch, trình thông dịch chỉ chuyển đổi một mệnh đề của chương trình và thực hiện đoạn mã kết quả ngay, sau đó nó tiếp tục chuyển đổi mệnh đề thứ 2 rồi thi hành đoạn mã kết quả thứ 2 và cứ thế. Khi sử dụng trình thông dịch, mỗi lần chạy chương trình là mỗi lần chương trình nguồn được thông dịch sang ngôn ngữ máy. Không có chương trình đối tượng nào được tạo ra.

Các trình thông dịch thường được dùng trên các máy tính cá nhân không có đủ bộ nhớ hoặc sức mạnh tính toán cần thiết để dùng trình biên dịch. Lợi điểm của trình thông dịch là lập trình viên vẫn có thể chạy một chương trình vẫn còn lỗi cú pháp. Chỉ đến lúc thông dịch đến câu lệnh có lỗi cú pháp, quá trình thi hành chương trình mới bị ngừng lại và trình thông dịch sẽ thông báo lỗi. Điểm bất lợi là các chương trình thông dịch chạy không nhanh bằng các chương trình được biên dịch vì quá trình chuyển đổi sang ngôn ngữ máy được thực hiện cùng với quá trình thi hành chương trình. Vì lý do này, ngày nay, đa số các ngôn ngữ cấp cao đều dùng trình biên dịch.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 4

Câu 1. Trình bày chức năng và cấu trúc của một bộ vi xử lý?

Câu 2. Trình bày phương pháp tổ chức của CPU:

a) Nhiệm vụ của CPU?
b) Sơ đồ cấu trúc cơ bản của CPU?
c) Các thành phần cơ bản của CPU?
d) Đơn vị điều khiển có chức năng gì?

Câu 3. Trình bày chức năng và nhiệm vụ:

a) Các thanh ghi đoạn?
b) Các thanh ghi đa năng?
c) Các thanh ghi con trỏ và chỉ số?
d) Đơn vị số học và logic (ALU)?
e) Các thanh ghi cờ FR?
f) Đơn vị điều khiển?

Câu 4. Tìm hiểu trong hệ thống CPU:

a) Các thành phần chình trong CPU là gì?
b) Khái niệm Data path khi đề cập đến tổ chức của bộ xử lý?
c) Chức năng và đặc điểm của bộ đếm chương trình PC?
d) Viết sơ đồ thuật toán để mô tả cách thức đọc một địa chỉ từ bộ nhớ vào CPU để xử lý?

Câu 5. Ngôn ngữ lập trình:

a) Nêu các loại ngôn ngữ lập trình thông dụng
b) Nêu các chương trình dịch và ưu nhược điểm của mỗi loại.
c) Trình bày cấu trúc cơ bản về Ngôn ngữ Assembler

Câu 6. Thực hiện phép toán sau.

f = E - (A + B/C) / (D + B) Dùng các lệnh 3, 2, 1, 0 địa chỉ

Câu 7: Viết chương trình bằng ngôn ngữ lập trình Assembly

a) Hiện ra hai câu “Chao mung ban den voi Assembly” “Assembly that de!”. Mỗi câu trên một dòng.
b) Yêu cầu “nhập một ký tự và xuất ra màn hình ký tự vừa nhập”.
c) Yêu cầu “nhập vào một ký tự. Chuyển ký tự đó sang ký tự hoa”.
d) Yêu cầu “Chuyển đổi ký tự hoa thành ký tự thường”.
e) Yêu cầu “nhập vào một chuỗi. In ra màn hình chuỗi thường, chuỗi in. Dùng chương trình con”.
f) Yêu cầu “nhập vào một chuỗi. Đếm chiều dài của chuỗi nhập vào”.
g) Yêu cầu “nhập vào 2 số kiểu word,in ra màn hình tổng 2 số vừa nhập”.
h) Yêu cầu “cho một mảng M gồm 20 phần tử kiểu Word giá trị tuỳ ý (không phải nhập giá trị các phần tử). Tính tổng giá trị các phần tử có giá trị chia hết cho 7”.

i) Yêu cầu "nhập vào 1 số kiêu word in ra màn hình mã nhị phân tương ứng của số đó".

Câu 8. Giả sử máy tính có các thanh ghi R0 = 1800, R1 = 1600, R2 = 1400 và giá trị tại ô nhớ M(1900) = 100, M(100) = 130 (các số trong hệ thập phân). Máy tính sử dụng lệnh hai toán hạng có dạng:

LỆNH Toán_hạng_đích, Toán_hạng_nguồn

Hãy cho biết địa chỉ thực của bộ nhớ cần truy cập đến và giá trị các thanh ghi khi thực hiện các lệnh sau:

a) ADD R1, R0
b) MOVE 500(R0), R1
c) SUB R1, (R2)

STORE 1000, #1200

Câu 9. Trình bày đường đi của dữ liệu:

a) Bộ điều khiển mạch điện tử?
b) Diễn biến thi hành lệnh mã máy?
c) Ngắt quãng (INTERRUPT)?

Câu 10. Trình bày về các kỹ thuật và những khó khăn:

a) Kỹ thuật ống dẫn (PIPELINE)?
b) Khó khăn trong kỹ thuật ống dẫn?
c) Siêu ống dẫn?
d) Siêu vô hướng (SUPERSCALAR)?

Chương 5
HỆ THỐNG NHỚ

Trang bị cho sinh viên kiến thức về chức năng và nguyên lý hoạt động của các cấp bộ nhớ máy tính: bộ nhớ cache: nguyên lý vận hành, phân loại các mức, đánh giá hiệu quả hoạt động; và nguyên lý vận hành của bộ nhớ ảo.

Sinh viên cần hiểu được các cấp bộ nhớ và cách thức vận hành của các loại bộ nhớ được giới thiệu để có thể đánh giá được hiệu năng hoạt động của các loại bộ nhớ.

5.1. TỔNG QUAN VỀ HỆ THỐNG NHỚ

5.1.1. Phân loại hệ thống nhớ

5.1.1.1. Vị trí:

a) Bên trong CPU
- Tập các thanh ghi.
- Bộ nhớ vi chương trình trong đơn vị điều khiển.

b) Bộ nhớ trong
- Bộ nhớ chính.
- Bộ nhớ cache.

c) Bộ nhớ ngoài
- Các thiết bị nhớ ngoài như ổ cứng, đĩa từ, đĩa quang, USB...

5.1.1.2. Dung lượng

Độ dài từ nhớ: Tính bằng bit, thường là 8, 16, 32, 64 bit.

Số lượng từ nhớ.

5.1.1.3. Đơn vị trao đổi:

Trao đổi theo từ nhớ: Đơn vị tự nhiên ở tổ chức bộ nhớ. Kích thước từ nhớ thường là số bit dùng để biểu diễn số hoặc độ dài lệnh.

Trao đổi theo khối nhớ: Là đơn vị truyền dữ liệu lớn hơn từ nhớ, thường được dụng truyền dữ liệu với bộ nhớ ngoài.

5.1.1.4. Phương pháp truy nhập:

Truy nhập tuần tự: Băng từ.

Truy nhập trực tiếp: Giống như truy nhập tuần tự, truy nhập trực tiếp bao hàm việc chia sẻ đọc viết cơ khí.Những từ nhớ của bản ghi có địa chỉ cơ sở duy nhất trên vị trí vật lý. Việc truy nhập được hoàn thành bởi truy nhập trực tiếp là đi đến vùng lân cận chung cộng với tìm kiếm tuần tự, đếm hoặc đợi để đi đến vị trí cuối cùng.Thời gian truy nhập có thể thay đổi được. Các loại đĩa sử dụng phương pháp truy nhập trực tiếp.

Truy nhập ngẫu nhiên: Mỗi vị trí địa chỉ trong bộ nhớ là độc nhất. Thời gian truy nhập các vị trí đã cho là độc lập với dãy truy nhập ưu tiên và là hằng số.Như vậy, vị trí nào cũng có thể được chọn ngẫu nhiên, và địa chỉ trực tiếp.Bộ nhớ chính là truy nhập ngẫu nhiên.

Truy nhập liên kết: Đây là kiểu truy nhập ngẫu nhiên có thể làm sự so sánh vị trí bít trong từ cho một phép toán cụ thể và làm việc này cho tất cả các từ đồng thời. Vì vậy một từ được tìm lại được dựa vào chính nội dung của nó thay vì địa chỉ của nó.Với truy nhập ngẫu nhiên thông thường, mỗi vị trí có địa chỉ cơ khí của mình, và thời gian tìm là hằng số độc lập với vị trí hay mẫu hình truy nhập ưu tiên.Bộ nhớ cache dùng cách truy nhập này.

5.1.1.5. Hiệu năng:

Thời gian truy nhập: Đối với truy nhập ngẫu nhiên đó là thời gian để thực hiện hoạt động đọc ghi. Đó là thời gian từ khi địa chỉ đã sẵn sàng trong bộ nhớ đến khi dữ liệu được cất trữ hoặc được làm có thể sử dụng được. Đối vớ truy nhập không phải là ngẫu nhiên thời gian truy nhập là thời gian đưa vị trí đọc viết cơ khí đến vị trí mong muốn.

Chu kỳ nhớ: Khoảng cách giữa hai lần truy nhập.

Tốc độ truyền: Bao nhiêu byte trong một đơn vị thời gian.

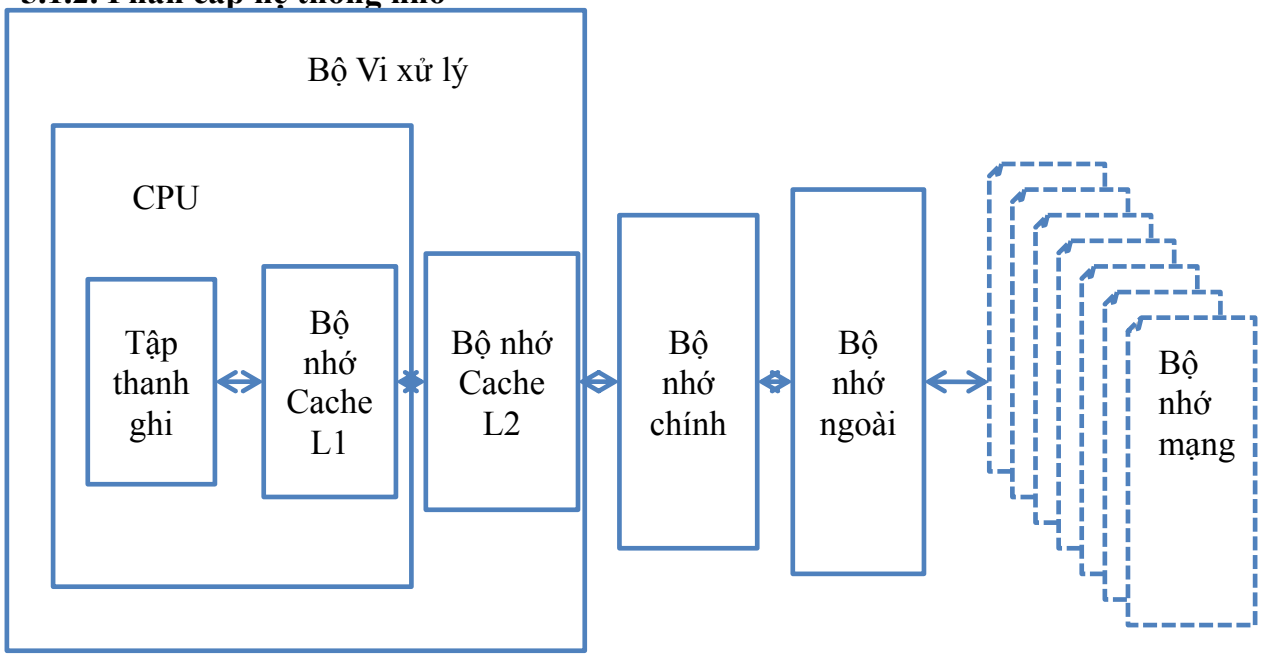
5.1.1.6. Kiểu vật lý:

- Bộ nhớ bán dẫn
- Bộ nhớ từ
- Bộ nhớ quang

5.1.1.7. Các đặc tính vật lý:

- Khả biến / không khả biến
- Xóa được / không xóa được

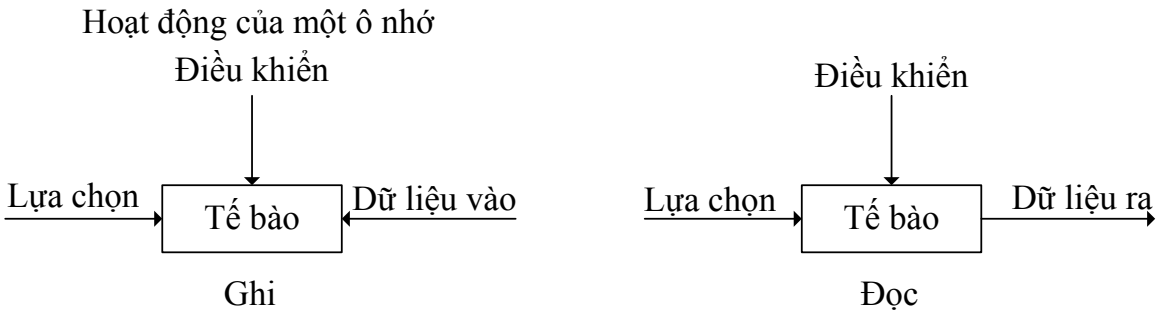
5.1.2. Phân cấp hệ thống nhớ



Hình 5-1. Phân cấp hệ thống nhớ

Kết luận: Dung lượng tăng dần, tốc độ giảm dần, giá thà.nh/ 1 bit giảm dần.

5.2. BỘ NHỚ BÁN DẪN



Hình 5-2. Hoạt động của ô nhớ

5.2.1. Phân loại bộ nhớ bán dẫn

Thực tế ROM và RAM đều là loại bộ nhớ truy xuất ngẫu nhiên, nhưng RAM được giữ tên gọi này. Để phân biệt chính xác ROM và RAM ta có thể gọi ROM là bộ nhớ chết (nonvolatile, vĩnh cữu) và RAM là bộ nhớ sống (volatile, không vĩnh cữu) hoặc nếu coi ROM là bộ nhớ chỉ đọc thì RAM là bộ nhớ đọc được - viết được (Read-Write Memory). Có 3 loại bộ nhớ bán dẫn:

- Bộ nhớ bán dẫn chỉ đọc: (Read Only Memory, ROM)
- Bộ nhớ truy xuất ngẫu nhiên: (Random Access Memory, RAM)
- Thiết bị logic khả trình: (Programmable Logic Devices, PLD) có thể nói điểm khác biệt giữa PLD với ROM và RAM là qui mô tích hợp của PLD thường không lớn như ROM và RAM và các tác vụ của PLD thì có phần hạn chế.

Bảng 5-1. Phân loại bộ nhớ bán dẫn

<table>
<tr><th>Kiểu bộ nhớ</th><th>Tiêu chuẩn</th><th>Khả năng xóa</th><th>Cơ chế ghi</th><th>Tính khả biến</th></tr>
<tr><td>Read Only Memory(ROM)</td><td rowspan="2">Bộ nhớ chỉ đọc</td><td rowspan="2">Không xóa được</td><td>Mặt nạ</td><td rowspan="6">Không khả biến</td></tr>
<tr><td>Programmable ROM(PROM)</td><td rowspan="5">Bằng điện</td></tr>
<tr><td>ERASABLE prom(eprom)</td><td rowspan="2">Bộ nhớ hầu như chỉ đọc</td><td>Bằng tia cực tím, cả chíp.</td></tr>
<tr><td>Electrically Erasable PROM(EEPROM)</td><td>Bằng điện mức từng byte</td></tr>
<tr><td>Flash memory</td><td rowspan="2">Bộ nhớ ghi-đọc</td><td>Bằng điện, từng khối</td></tr>
<tr><td>Random Access Memory(RAM)</td><td>Bằng điện mức từng byte</td><td>Bằng điện</td><td>Khả biến</td></tr>
</table>

5.2.1.1. ROM (Read Only Memory)

Mặc dù có tên gọi như thế nhưng chúng ta phải hiểu là khi sử dụng ROM, tác vụ đọc được thực hiện rất nhiều lần so với tác vụ ghi. Thậm chí có loại ROM chỉ ghi một lần khi xuất xưởng.

a) ROM lưu trữ các thông tin:

- Thư viện các chương trình con
- Các chương trình điều khiển hệ thống (BIOS)
- Các bảng chức năng
- Vi chương trình
- ROM là Bộ nhớ không khả biến

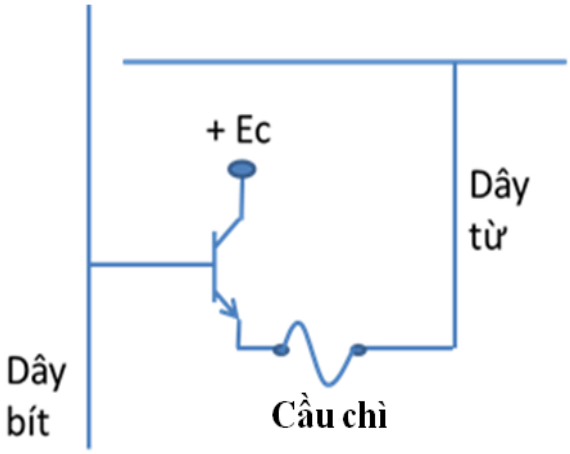
b) ROM mặt nạ (Mask Programmed ROM, MROM)

Đây là loại ROM được chế tạo để thực hiện một công việc cụ thể như các bảng tính, bảng lượng giác, bảng logarit,... ngay sau khi xuất xưởng và có giá thành rất đắt.

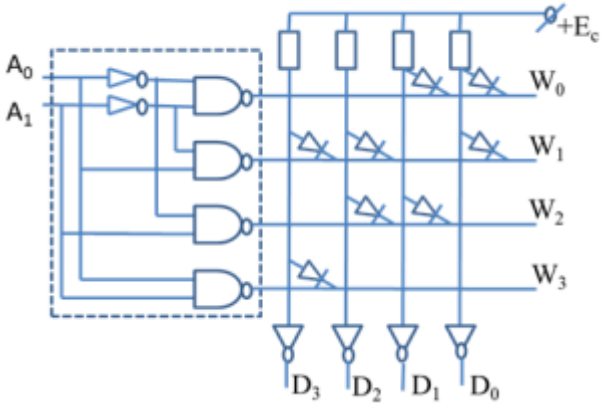
c) ROM khả trình (Programmable ROM, PROM)

Có cấu tạo giống MROM nhưng ở mỗi vị trí nhớ đều có linh kiện nối với cầu chì.

Một khi cầu chì đã bị phá vỡ thì không thể nối lại được do đó loại ROM này cho phép lập trình một lần duy nhất để sử dụng, nếu bị lỗi không thể sửa chữa được.



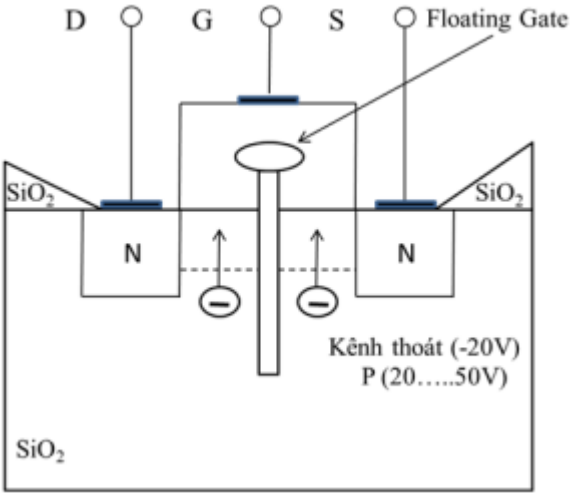
Hình 5-3. Sơ đồ PROM



Hình 5-4. Sơ đồ ROM Diode

ROM khả trình, xóa được bằng tia U.V. (Ultra Violet Erasable Programmable ROM, U.V. EPROM)

Đây là loại ROM rất tiện cho người sử dụng vì có thể dùng được nhiều lần bằng cách xóa và nạp lại.Trước khi ghi lại phải xóa bằng tia cực tím. Điểm bất tiện của U.V EPROM là cần thiết bị xóa đặc biệt phát tia U.V. và mỗi lần xóa tất cả tế bào nhớ trong một IC nhớ đều bị xóa. Như vậy người sử dụng phải nạp lại toàn bộ chương trình (UV chính là tia cực tím).



Hình 5-5. Sơ đồ EPROM

d) ROM khả trình và xóa được bằng xung điện (Electrically Erasable PROM, EEPROM hay Electrically Alterable PROM, EAPROM)

Đây là loại ROM khả trình và xóa được nhờ xung điện và đặc biệt là có thể xóa để sửa trên từng byte.

e) FLASH ROM

EPROM là loại nonvolatile, có tốc độ truy xuất nhanh (khoảng 120ns), mật độ tích hợp cao, giá thành rẻ tuy nhiên để xóa và nạp lại phải dùng thiết bị đặc biệt và lấy ra khỏi mạch.

5.2.1.2. RAM (Random Acess Memory)

Bộ nhớ đọc ghi, khả biến, lưu trữ thông tin tạm thời. Có hai loại RAM: RAM tĩnh và RAM động.

a) RAM tĩnh.

Các bít được lưu trữ bằng FF nên thông tin ổn định. Có cấu trúc phúc tạp. Dung lượng chíp nhỏ. Tốc độ nhanh. Đắt tiền. Dùng làm bộ nhớ cache

b) RAM động (Dynamic RAM, DRAM)

Các bít được lưu trữ trên tụ điện nên cần phải có mạch làm tươi, nó cấu trúc đơn giản, dung lượng lớn, nhưng tốc độ chậm hơn,rẻ tiền hơn,dùng làm bộ nhớ chính.

5.2.1.3. Các DRAM tiên tiến

- EDRAM (Enhanced DRAM)
- CDRAM (Cache DRAM)
- SDRAM (Synchronous DRAM): Được làm viêc đồng bộ bởi xung clock
- DDR – SDRAM (Double Data Rate SDRAM)
- RDRAM (Rambus DRAM)

5.2.1.4. Làm tươi bộ nhớ DRAM

Bộ nhớ DRAM có các hàng cần phải được làm tươi trong mỗi chu kỳ 2mS. Mạch làm tươi trong chip nhớ phải kiểm tra điện áp các ô nhớ, nếu nó lớn hơn $V_{cc}/2$ thì nạp nó tới V_{cc} , nếu bé hơn $V_{cc}/2$ thì xả hết về 0V.

Để đọc một từ từ BANK nhớ DRAM, trước hết DRAM Controller hoặc một mạch khác cấp tín hiệu WE# = 1. Sau đó gửi nửa thấp của địa chỉ, ứng với địa chỉ hàng, rồi tín hiệu RAS# = 0. Sau 1 thời gian, controller cấp nửa địa chỉ cao, ứng với địa chỉ cột,

rồi tín hiệu CAS# = 0. Sau thời gian nhất định, từ cần có sẽ xuất hiện trên Output Data của nhớ.

Để viết vào DRAM, các tín hiệu cũng tương tự, ngoại trừ sau tín hiệu CAS# = 0, controller cấp WE# = 0 để quy định viết vào RAM.

Controller làm tươi DRAM bằng cách gửi ra mỗi địa chỉ trong 512 địa chỉ hàng và cấp RAS# = 0 theo chu kỳ, khoảng 4mS. Việc làm tươi được tiến hành hoặc theo *burst mode* hoặc theo *distributed mode*. Trong burst mode toàn bộ 512 hàng được định địa chỉ và đánh nhịp lần lượt cách nhau 4mS. Còn ở distributed mode hàng được định địa chỉ và đánh nhịp sau 4/512 mS. Là mạch làm tươi DRAM với controller làm tươi 8208.

Những nhiệm vụ chính của việc điều khiển nhớ DRAM của máy tính là:

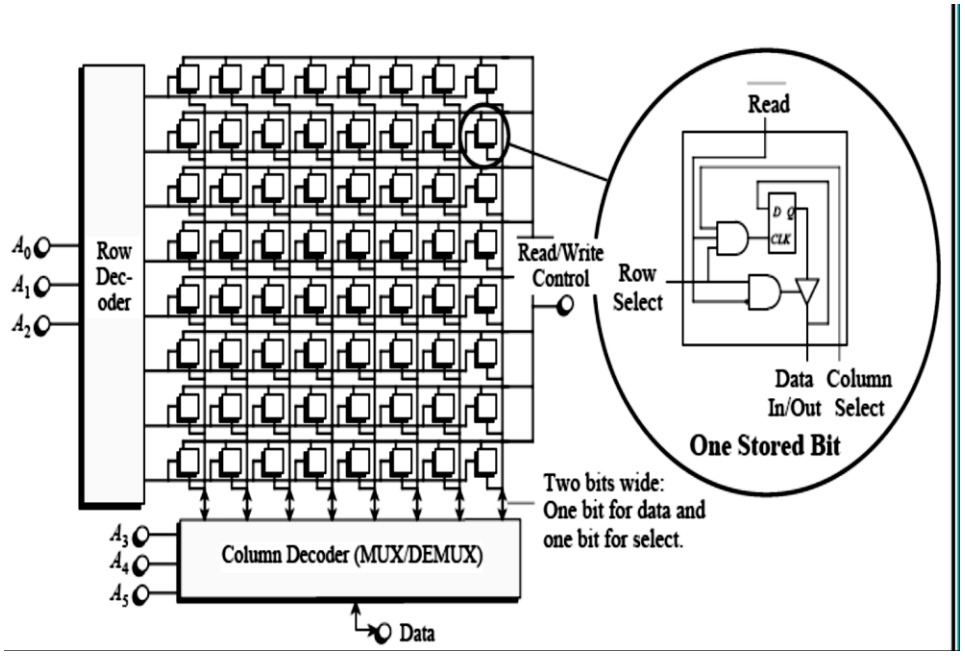
- Làm tươi mỗi ô nhớ sau một khoảng thời gian vài mS.
- Cấp hai nửa địa chỉ cùng các tín hiệu RAS#, CAS# thích hợp.
- Bảo đảm thao tác đọc/viết và làm tươi không xảy ra đồng thời.
- Cấp tín hiệu đọc/viết để điều khiển chiều số liệu.

5.2.2. Tổ chức bộ nhớ

Bộ nhớ thường được tổ chức từ nhiều vi mạch (chip) nhớ ghép lại để có độ rộng bus địa chỉ và dữ liệu cần thiết. Các chip nhớ có đầy đủ chức năng của một bộ nhớ bao gồm:

- Ma trận nhớ: gồm các ô nhớ, mỗi ô nhớ tương ứng với một bit nhớ.
- Mạch giải mã địa chỉ cho bộ nhớ.
- Mạch logic cho phép đọc.
- Mạch logic cho phép ghi.
- Các mạch đệm vào, ra.

Cách tổ chức đơn giản nhất là tổ chức theo word. Một ma trận nhớ có độ dài của cột bằng số lượng word (W)và độ dài hàng bằng số lượng bit (B) của một word. Phương pháp này có thời gian truy xuất ngắn nhưng đòi hỏi bộ giải mã lớn khi tổng số word lớn.



Hình 5-6. Tổ chức bộ nhớ

a. Tổ chức chip nhớ

Trong một chip bao gồm:

- Các đường địa chỉ: A0 ÷ An – 1, như vậy chip nhớ có 2^n ngăn nhớ.

a. Các đường dữ liệu: D0 ÷ Dm – 1, như vậy độ dài ngăn nhớ là m bit.

b. Dung lượng chip nhớ: 2n x m bit

c. Các đường điều khiển:

- Tín hiệu chọn chip: CS (Chip Select)
- Tín hiệu điều khiển đọc: RD / OE
- Tín hiệu điều khiển ghi: WR / WE

d. Thiết kế Module nhớ bán dẫn

- Đặt vấn đề: Cho chip nhớ 2^n x m bit. Yêu cầu sử dụng chip nhớ trên thiết kế module nhớ dung lượng là bội kích thước chip nhớ trên.
- Giải quyết vấn đề: Có hai cách
  - Thiết kế để tăng độ dài ngăn nhớ, số ngăn nhớ không thay đổi.
  - Thiết kế để tăng số lượng ngăn nhớ, độ dài ngăn nhớ không thay đổi.
  - Thiết kế để tăng cả độ dài từ nhớ và số ngăn nhớ.

e. Thiết kế tăng độ dài ngăn nhớ

- Giả thiết: Cho các chip nhớ có dung lượng 2^n x m bit. (n là số đường địa chỉ, m là số bít trong một ô nhớ)
- Yêu cầu: Thiết kế module nhớ có kích thước: 2^n x (k.m) bit

f. Giải quyết: Để thiết kế được yêu cầu ta xác định hai thông số n (số đường địa chỉ)và k(số chip nhớ cần để ghép vào module thiết kế.

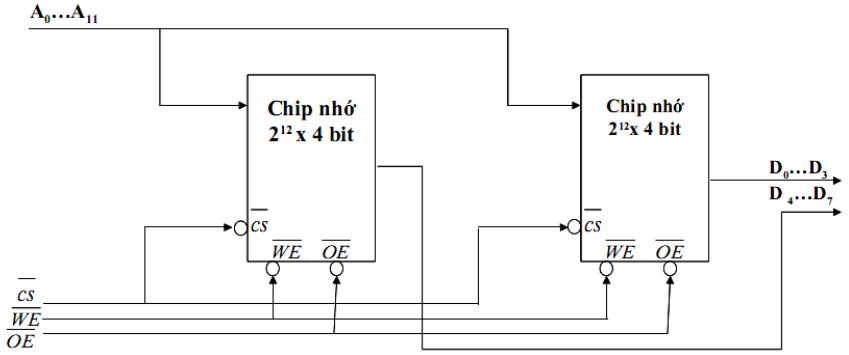
Ví dụ 5.2.1: Cho các chip nhớ SDRAM dung lượng 4K x 4 bit. Hãy thiết kế module nhớ có kích thước 4K x 8 bit

Giải:

Dung lượng chip nhớ 4K x 4 bit = 2^{12}K x 4 bit, => số đường địa chỉ n = 12, số đường dữ liệu m=4.

Nhận thấy với yêu cầu của đề bài thì số đường địa chỉ là 12 đường không đổi (số ngăn nhớ không thay đổi), số đường dữ liệu là 8 (tức kihcs thước một ô nhớ đang từ 4 bit tăng lên thành 8bit), vậy số chip sử dụng để thiết kế là 2(k=2).

Mạch thiết kế:



Hình 5-7. Thiết kế module nhớ có kích thước 4K x 8 bit

- **Thiết kế tăng số lượng ngăn nhớ**
  - Giả thiết: Cho các chip nhớ có dung lượng 2^n x m bit.
  - Yêu cầu: Thiết kế module nhớ có kích thước: $2^k.2^n$ x m bit
- Giải quyết:

Để thiết kế được ta xác định hai thông số n+k (số đường địa chỉ mới để mã hóa đủ số ô nhớ cần thiết kế) và 2^k (số chip nhớ cần để ghép vào module thiết kế).

Ví dụ 5.2.2: Cho các chip nhớ SDRAM dung lượng 4K x 8 bit. Hãy thiết kế module nhớ có kích thước 8K x 8 bit.

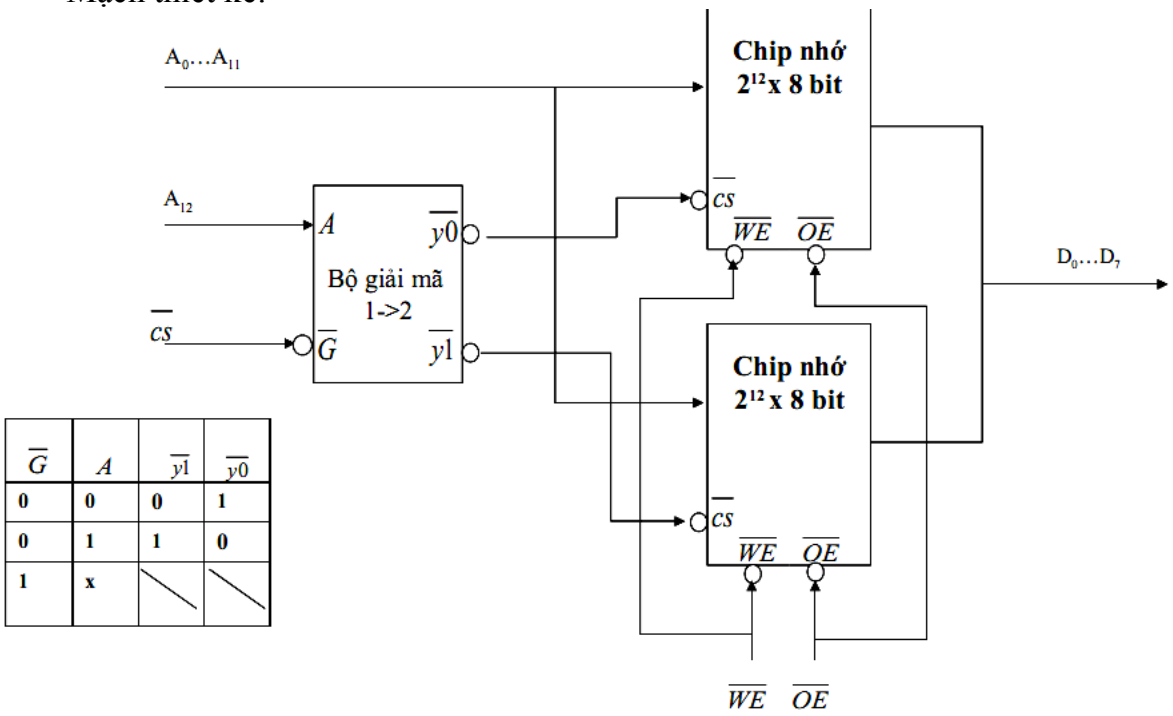
Giải:

Nhận thấy rằng đề yêu cầu tăng số lượng ô nhớ lên 2 lần tức từ 4K lên 8K, còn kích thước một ô nhớ vân là 8 bit.

Dung lượng 4K x 8bit = 2^{12}k x 8 bit, => số đường địa chỉ là n =12 và số đường dữ liệu m=8.

Yêu cầu mới là 8K x 8bit = 2^{13}K x 8bit = 2 x 2^{12}K x 8bit = 2 chíp nhớ 4Kx8bit.

Mạch thiết kế:

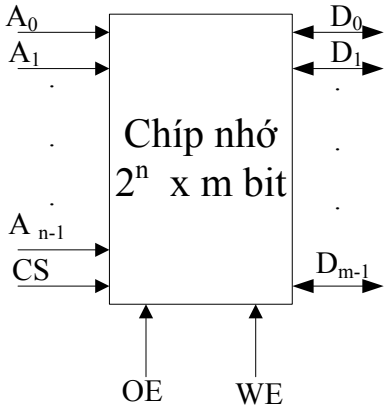


$\overline{G}$	A	$\overline{y1}$	$\overline{y0}$
0	0	0	1
0	1	1	0
1	x		

- Tăng cả số lượng và độ dài ngăn nhớ

- Giả thiết: Cho chip nhớ 2n x m bit
- Yêu cầu: Thiết kế module nhớ có kích thước: 2p+n x (q.m) bit
- Giải quyết: Cần ghép nối q.2p chip thành 2p bộ, mỗi bộ q chip và phải

5.2.2.1. Tổ chức của chip nhớ



Hình 5-8. Sơ đồ cơ bản của chip nhớ

**** Các tín hiệu của chip nhớ***

Các đường địa chỉ:

$A_{n-1} \div A_0$ có n từ nhớ

Các đường dữ liệu:

$D_{m-1} \div D_0$ có độ dài từ nhớ bằng m bít

Dung lượng chip nhớ = 2^n x m bit

Các đường điều khiển:

Tín hiệu chọn chip CS (Chip select)

Tín hiệu điều khiển đọc OE (Output Enable)

Tín hiệu điều khiển ghi WE (Write Enable)

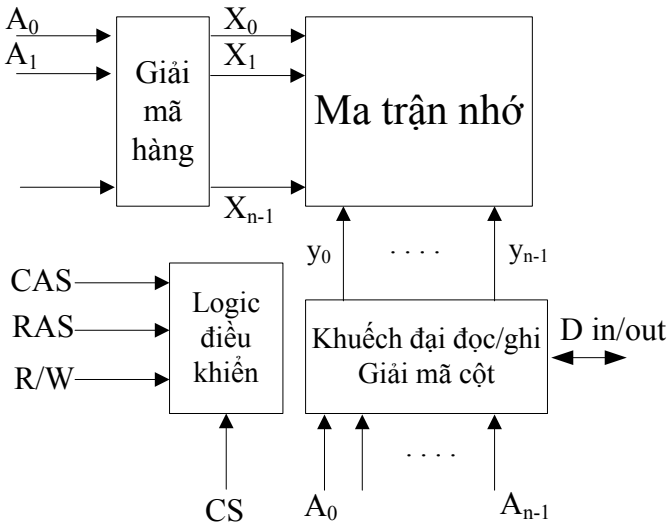
**** Tổ chức của DRAM***

Dùng n đường địa chỉ dồn kênh nên cho phép truyền 2n bit địa chỉ

Tín hiệu chọn địa chỉ hàng RAS (Row Address Select)

Tín hiệu chọn địa chỉ cột CAS (Column Address Select)

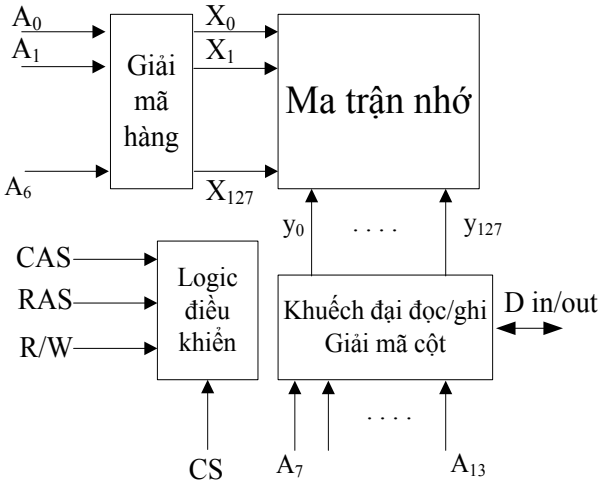
Dung lượng của DRAM = 2^{2n} x m bit



Hình 5-9. Cấu trúc RAM

***Ví dụ 5.2.3*:** Vẽ cấu trúc RAM (16K x 1 bit)

Ta có: 16k x 1bit = 2^4 x 2^{10} = 2^{14}



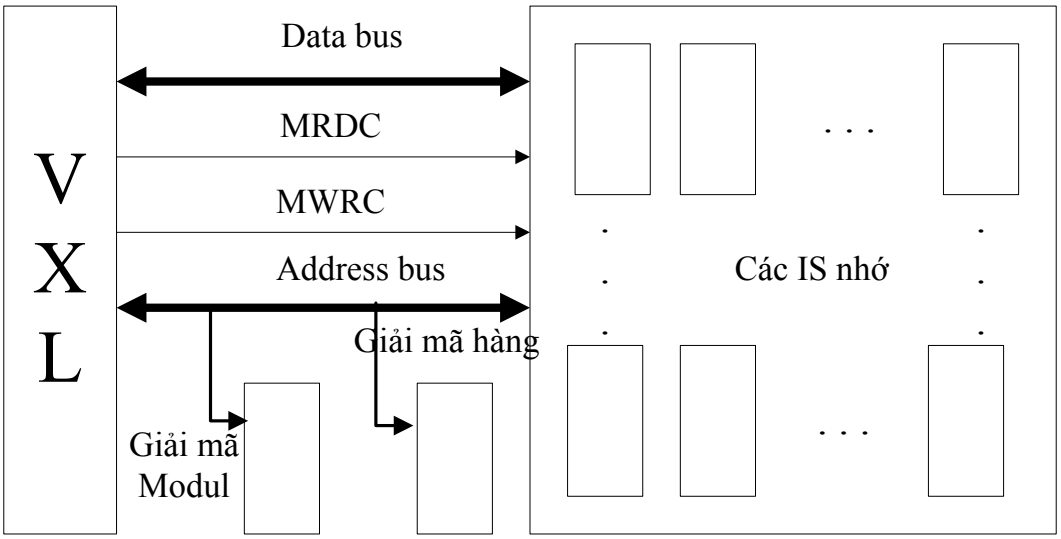
5.2.2.2. Thiết kế mô-đun nhớ bán dẫn

Dung lượng chip nhớ 2^n x m bit

Chính vì thế cần thiết kế để tăng dung lượng:

- Thiết kế tăng độ dài từ nhớ
- Thiết kế tăng số lượng từ nhớ
- Kết hợp cả 2 cách trên

Cấu trúc của một bộ nhớ



Hình 5-10. Cấu trúc của bộ nhớ

Xác định ma trận IC

- Số IC trên một hàng (Số cột) = Số bit dữ liệu bộ nhớ/ Số bit dữ liệu IC
- Số IC trên một cột (Số hàng) = Số từ nhớ của bộ nhớ/ Số từ nhớ của IC

Các IC trên cùng một hàng nối chung CE

Các IC trên cùng một cột được nối chung với các bit dữ liệu tương ứng

Tất cả các IC của ma trận được nối chung bit địa chỉ

Giải mã chọn hàng:

Đầu ra nối với các tín hiệu CE của từng hàng

Đầu vào là các bit được còn lại

Giải mã chọn module:

Đầu ra đưa vào CE của IC giải mã

Ví dụ 5.2.4: Xây dựng bộ nhớ 64KB cho một máy tính sử dụng CPU 8085 (16 bit địa chỉ, 8 bit dữ liệu) từ các RAM được cấu tạo từ các ma trận 128x128 bit với 4 bit dữ liệu: Địa chỉ bộ nhớ bắt đầu: C000H

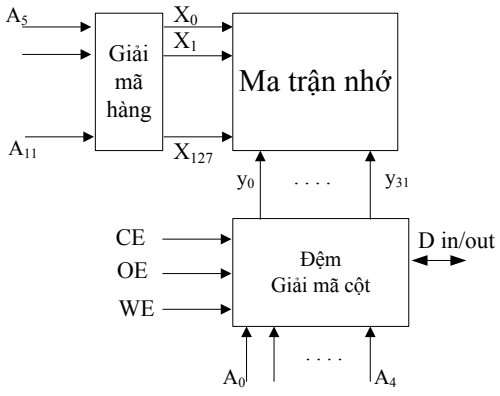
$a_H = \log_2 N = \log_2 128 = 7$

$a_C = \log_2(N/m) = \log_2 (128/4) = 5$

$a = a_C + a_H = 7+5 = 12$

Vậy dung lượng 1 IC là 2a x m = 4Kx4

$(128x128) = 2^{14} = 4Kx4$



Ma trận IC:

Số IC trên một hàng: 8/4 = 2

Số IC trên một cột : 16K/4K = 4

Số bit dùng cho giải mã: 16 – 12 = 4 bit

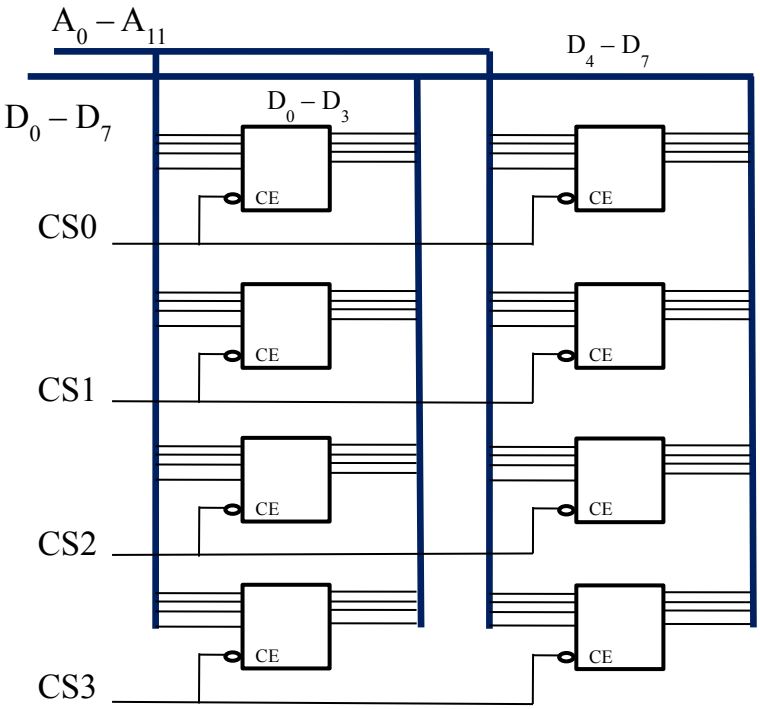
Số bit dùng cho giải mã chọn hàng : 2

Số bit dùng cho giải mã chọn module : 2

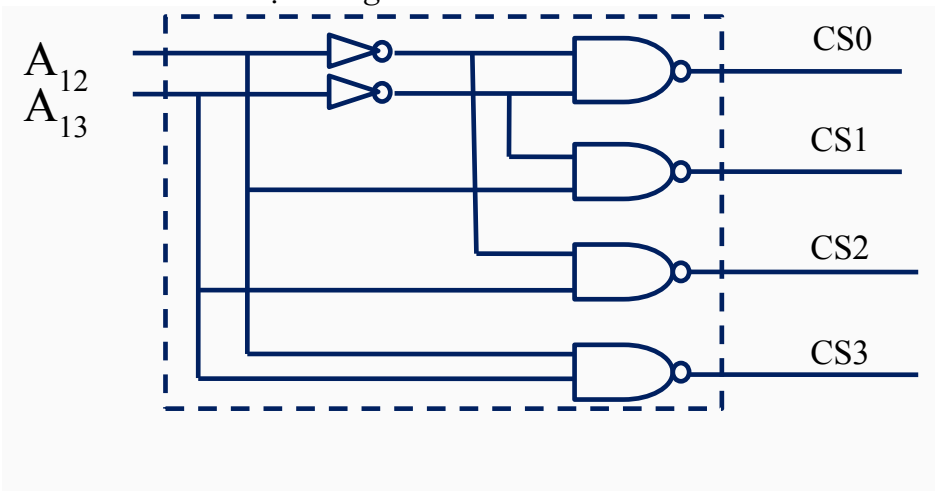
CS0: C0000H – CFFFFH : 1 1 0 0

CS1: D0000H – DFFFFH : 1 1 0 1

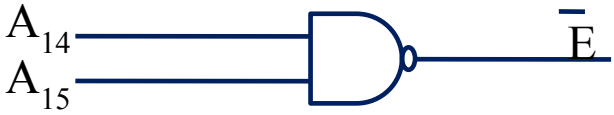
CS2: E0000H – EFFFFH : 1 1 1 0



Giải mã chọn hàng



Giải mã chọn module



Ví dụ 5.2.5. (Tăng độ dài từ nhớ)

Cho chip nhớ SRAM 4K x 4 bit

Thiết kế module nhớ 4K x 8 bit

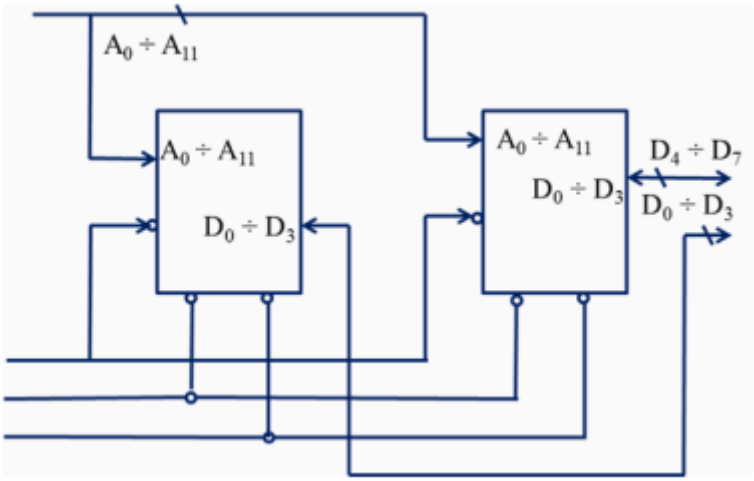
Hướng dẫn:

Dung lượng chip nhớ 4K x 4 bit = 2^{12} x 4 bit. Chip nhớ có: 12 chân địa chỉ và 4 chân dữ liệu.

Module nhớ 4K x 8 bit =2^{12} x 8 bit . Module nhớ có: 12 chân địa chỉ và 8 chân dữ liệu.

Số IC trên một hàng: 8/4 = 2

Số IC trên một cột : 4K/4K = 1



Ví dụ 5.2.6. (Tăng số lượng từ nhớ)

Cho chip nhớ SRAM 4K x 8 bit

Thiết kế module nhớ 8K x 8bit

Hướng dẫn:

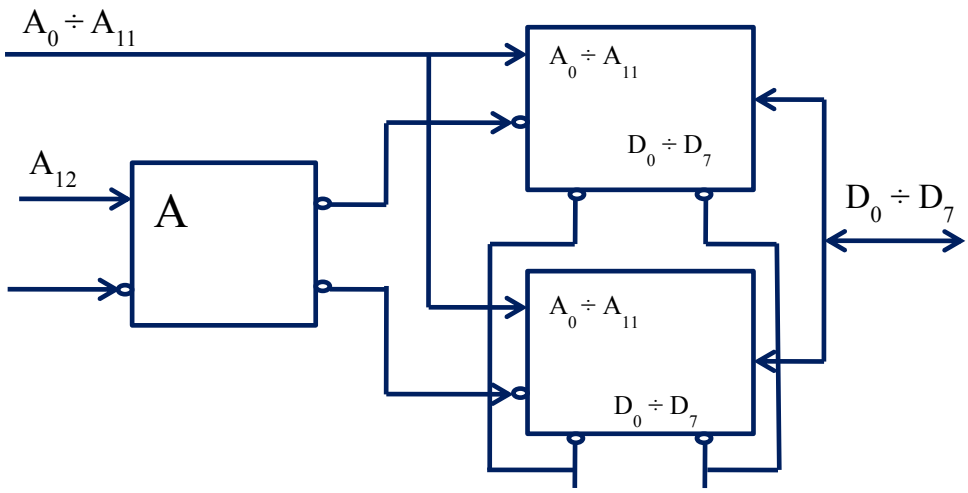
Dung lượng chip nhớ 4K x 8 bit = 2^{12} x 8 bit. Chip nhớ có: 12 chân địa chỉ, 8 chân dữ liệu

Dung lượng module nhớ 8K x 8 bit = 2^{13} x 8 bit. Module nhớ có: 13 chân địa chỉ, 8 chân dữ liệu

Số IC trên một hàng: 8/8 = 1

Số IC trên một cột : 8K/4K = 2

$\overline{G}$	A	Y0	Y1
0	0	0	1
0	1	1	0
1	X	1	1



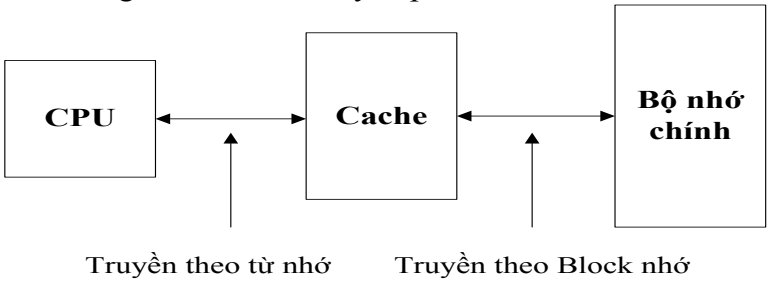
5.3. BỘ NHỚ CACHE, BỘ NHỚ TRUY CẬP NHANH

Khi tốc độ của bộ vi xử lý ngày càng vượt xa tốc độ truy nhập bộ nhớ chính (DRAM làm việc nhanh nhất chỉ 60ns), có nghĩa là bộ vi xử lý phải mất thêm vài chu kỳ đợi bộ nhớ hoàn thành quá trình đọc/ghi. Điều này làm giảm hiệu suất làm việc của bộ vi xử lý. Một giải pháp hữu hiệu là sử dụng bộ nhớ đệm cache với tốc độ truy nhập chỉ vài ns. Cache tiết kiệm thời gian truy xuất bộ nhớ của CPU bằng cách dự đoán trước lệnh mà CPU sẽ cần và nạp nó vào trong cache trước khi CPU thực sự cần đến nó. Nếu lệnh cần thiết đã có sẵn trong cache thì CPU sẽ truy xuất dữ liệu từ cache, nếu không, CPU mới truy xuất lên bộ nhớ chính.

5.3.1. Nguyên tắc chung của cache

Bộ nhớ cache dựa theo nguyên lý cục bộ hóa tham chiếu bộ nhớ. Tức là, trong một khoảng thời gian đủ nhỏ CPU thường chỉ tham chiếu các thông tin trong một khối nhớ cục bộ.

Cache có tốc độ nhanh hơn bộ nhớ chính. Cache được đặt giữa CPU và bộ nhớ chính nhằm tăng tốc độ CPU truy cập bộ nhớ. Cache có thể được đặt trên chíp CPU.



Hình 5-11. Vị trí đặt bộ nhớ cache

5.3.1.1. Các đặc điểm của bộ nhớ Cache

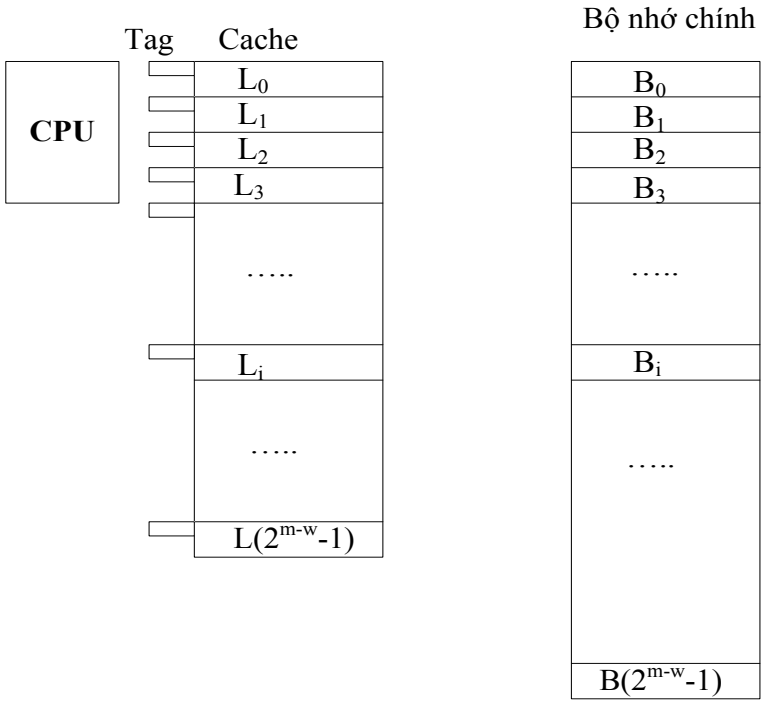
- Cache có tốc độ nhanh hơn bộ nhớ chính
- Bộ nhớ bán dẫn tốc độ cao
- Dung lượng rất nhỏ so với ổ cứng (256 – 512 KB)
- Cache HIT: Xác định từ nhớ nằm trong Cache
- Cache Miss: Xác định từ nhớ không nằm trong cache
- Tỷ lệ Hit = (Số lần tìm thấy từ nhớ/Tổng số lần tìm)

- H = (85 – 90%)
- Tỷ lệ Miss = 1 – H

5.3.1.2. Thao tác của bộ nhớ Cache:

- CPU yêu cầu nội dung của ngăn nhớ.
- CPU kiểm tra trên cache với dữ liệu này.
- Nếu có, CPU nhận dữ liệu từ cache.
- Nếu không có, đọc Block nhớ chứa dữ liệu từ bộ nhớ chính vào cache.
- Tiếp đó chuyển dữ liệu từ cache vào CPU.

5.3.1.3. Cấu trúc chung của cache/ bộ nhớ chính



Hình 5-12. Cấu trúc chung của cache

Sử dụng n bit địa chỉ với độ dài từ nhớ xác định. Số lượng từ nhớ xác định là 2^n.

Bộ nhớ chính được chia thành các Block, mỗi Block có chứa 2^w ô nhớ. Số Block = $2^{n-w} = 2^s$.

Trong n bit địa chỉ dùng w bit LSB để xác định vị trí của từ nhớ trong Block.

Kích thước của Block = 8, 16, 32, 64, 128 byte.

Một số Block của bộ nhớ chính được nạp vào các Line của cache.

Số Line của của Cache = $2^{m-w} = 2r$

Nội dung Tag (thẻ nhớ) cho biết Block nào của bộ nhớ chính hiện đang được chứa ở Line đó.

Khi CPU truy nhập (đọc/ghi) một từ nhớ, có hai khả năng sảy ra:

Từ nhớ đó có trong cache (cache hit)

Từ nhớ đó không có trong cache (cache miss)

Bộ nhớ cache:

Có m bit địa chỉ (m < n)

Cache cũng được chi thành các Block, được gọi là Line. Kích thước của 1 Line bằng với các kích thước của một Block

Số Line của của Cache = $2^{m-w} = 2^r$

Trong thực tế thì số Line << số Block

Trong trường hợp Cache miss cần có một thuật toán thay thế thích hợp

Để CPU phân biệt được các Block khi đưa vào trong Cache ta sử dụng địa chỉ TAG. Địa chỉ TAG cho biết Block nào đang nằm trong Cache. Mỗi Block đều có một địa chỉ TAG duy nhất.

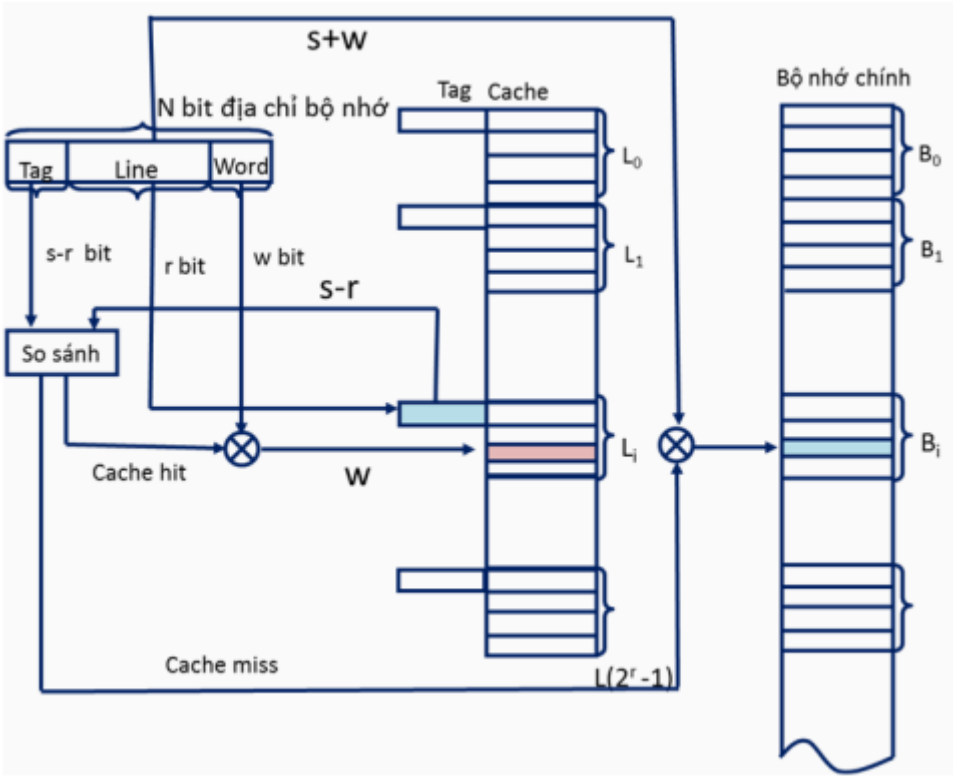
Khi xác định 1 từ nhớ nào đó có trong nằm trong cache hay không ta chỉ việc so sánh phần địa chỉ TAG của từ nhớ cần tìm với phần địa chỉ TAG của tất cả các Line có trong Cache.

5.3.2. Các phương pháp ánh xạ

Các phương pháp ánh xạ, chính là các phương pháp tổ chức bộ nhớ cache. Có ba phương pháp ánh xạ: Ánh xạ trực tiếp , ánh xạ liên kết toàn phần và loại thứ ba là phương pháp tổng quát của 2 phương pháp trên đó là ánh xạ liên kết tập hợp. (Trong khuôn khổ chương trình chỉ nghiên cứu ánh xạ trực tiếp và ánh xạ liên kết toàn phần).

5.3.2.1. Ánh xạ trực tiếp (Direct mapping)

Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một Line của cache:



Hình 5-13. Minh họa ánh xạ trực tiếp

B_0 -> L_0

B_1 -> L_1

.....

B_{m-1} -> L_{m-1}

B_m -> L_0

B_{m+1} -> L_1

.....

Tổng quát

B_j chỉ có thể được nạp vào $L_{j \bmod M}$

M là số Line của cache.
Số lượng Block: 2^s; (s= n - w)
Trường Word : Kích thước Block hay Line : 2^w (w là độ dài từ nhớ)
Trường Line : Số Line của cache 2^r (có r bit)
Trường TAG: Số bit = s - r

Bộ nhớ chính được tổ chức như sau:

TAG	**LINE**	**WORD**
s- r	r	w
N bít địa chỉ		

Ví dụ 5.3.1: Cho bộ nhớ chính có 16 từ nhớ chia thành các Block, mỗi Block 2 từ nhớ. Cache có dung lượng 8 từ nhớ. Xác định từ địa chỉ để quản lý bộ nhớ cache theo phương pháp ánh xạ trực tiếp.

Hướng dẫn:

Bộ nhớ chính có 16 từ nhớ = 2^4 , vậy n = 4
Mỗi Block có 2 từ nhớ = 2^1 , vậy w = 1
s = n – w = 4 – 1 = 3
Cache có dung lượng 8 từ nhớ = 2^3, vậy m = 3
r = m – w = 3 – 1 = 2
s – r = 1
Tổ chức của một Block:

TAG	**LINE**	**WORD**
1	2	1
N bít địa chỉ		

Tổ chức của cache:

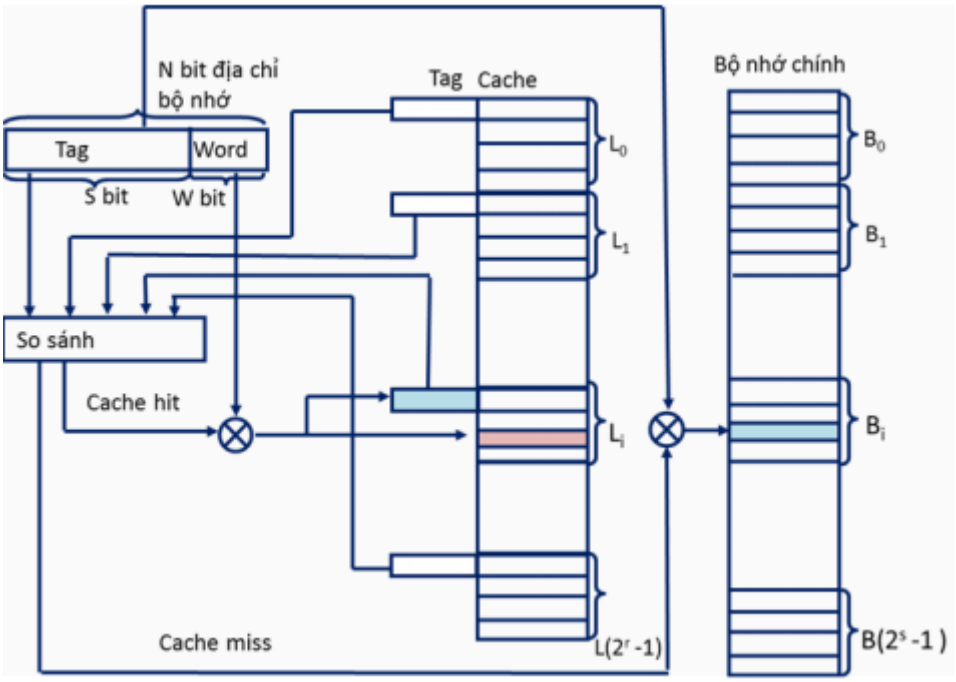
TAG	**W**	**Cache**	**Line**
00	0	DL1	L0
	1	DL2	
01	0	DL3	L1
	1	DL4	
10	0	DL5	L2
	1	DL6	
11	0	DL7	L3
	1	DL8	

Tổ chức bộ nhớ chính

TAG	LINE	Word	BNC	Block
0	00	0	DL1	B0
		1	DL2	
	01	0	DL3	B1
		1	DL4	
	10	0	DL5	B2
		1	DL6	
	11	0	DL7	B3
		1	DL8	
1	00	0	DL9	B4
		1	DL10	
	01	0	DL11	B5
		1	DL12	
	10	0	DL13	B6
		1	DL14	
	11	0	DL15	B7
		1	DL16	

5.3.2.2. Ánh xạ liên kết toàn phần (Fully associative mapping)

Mỗi Block có thể nạp vào bất kỳ Line nào của cache, miễn là Line đó còn rỗng.



Hình 5-14. Minh họa ánh xạ liên kết toàn phần

Địa chỉ của bộ nhớ chính bao gồm hai trường:

TAG	**WORD**
s	w
n bit địa chỉ	

Trường Word giống như trường hợp ánh xạ trực tiếp.
Trường Tag dùng để xác định Block của bộ nhớ chính.
Tag xác định Block đang nằm ở Line đó.
Đặc điểm của ánh xạ toàn phần:
- Kích thước bộ so sánh lớn nên tốc độ giảm
- Xác suất cache hit cao
- Bộ so sánh phức tạp
- Cần một thuật toán thay thế khi cache đầy
- Các Block có thể nằm ở mọi vị trí trong cache

Ví dụ 5.3.2:

Cho n = 4, w = 1, s = 3, r = 2
Tổ chức của Block:

TAG	**WORD**
3	1
n bit địa chỉ	

Tổ chức của cache:

r=2	**TAG**	**WORD**	**Line**
00	111	DL7	L0
		DL8	
01	011	DL3	L1
		DL4	
10	001	DL1	L2
		DL2	
11	101	DL5	L3
		DL6	

Tổ chức bộ nhớ chính

TAG	W	BNC	Block
000	0	DL1	B0
	1	DL2	
001	0	DL3	B1
	1	DL4	
010	0	DL5	B2
	1	DL6	
011	0	DL7	B3
	1	DL8	
100	0	DL9	B4
	1	DL10	
101	0	DL11	B5
	1	DL12	
110	0	DL13	B6
	1	DL14	
111	0	DL15	B7
	1	DL16	

Ví dụ 5.3.3: Ánh xạ địa chỉ

Không gian địa chỉ bộ nhớ chính = 64 MB.
Dung lượng bộ nhớ cache là 64 KB.
Kích thước Line (Block) = 64 byte.
Xác định số bit của các trường địa chỉ cho hai trường hợp :
- Ánh xạ trực tiếp
- Ánh xạ liên kết toàn phần

Với ánh xạ trực tiếp

Bộ nhớ chính = 64 MB = 2^{26} byte -> n = 26 bit
Dung lượng bộ nhớ Cache = 64 KB = 2^{16} byte -> m= 16 bit
Kích thước Block = 64 byte = 2^6 byte -> w = 6 bit
r = m-w= 16-6=10 bit
s= n – w = 26 – 6 = 20 bit
-> s – r = 20 – 10 = 10 bít

Tag	Line	Word
s-r	r	w
10	10	6

Ánh xạ liên kết toàn phần

Bộ nhớ chính = 64Mb = 2^{26} byte -> n = 26 bit
Line = 64 byte = 2^6 byte -> w = 6 bit
Số bit của trường tag sẽ là: s = n – w = 26-6= 20 bit

Tag	Word
s	w
20	5

5.3.2.3. Ánh xạ liên kết tập hợp

- Đây là phương pháp tổ hợp từ 2 phương pháp trên

B_0 -> S_0

B_1 -> S_1

.....

- Cache chia thành các tập (set), mỗi set chưa 1 số line

Số set= $\frac{2^r}{k} = 2^d$ (k là số line trong set)

Nếu k=1 -> r=d: phương pháp ánh xạ trực tiếp

Nếu k= 2^r -> d=0: phương pháp anh xạ liên kết toàn phần (k=2,4,6,8...)

Tổ chức của Block:

TAG	**SET**	**WORD**
s-d	d	w

Ví dụ 5.3.4:

Cho không gian địa chỉ bộ nhớ 4GB, dung lượng bộ nhớ Cache là 256KB, kích thước line là 32 byte.

Xác định số bit của trường địa chỉ theo phương pháp:

a) Ánh xạ trực tiếp

b) Ánh xạ liên kết toàn phần

c) Ánh xạ liên kết tập hợp.

Hướng dẫn:

Bộ nhớ chính= 4GB= 2^{32} byte => n=32 bit

Cache = 256 KB= 2^{18} byte => m= 18 bit

Line = 32 byte= 2^5 byte => w= 5 bit

Có: r= m-w= 18-5=13 bit

s= n-w= 32-5=27 bit

a) Ánh xạ trực tiếp

TAG	**LINE**	**WORD**
s-r	r	w
14	13	5

b) Ánh xạ liên kết toàn phần

TAG	WORD
s	w
27	5

c) Ánh xạ liên kết tập hợp

Số set= $\frac{2^r}{k} = \frac{2^{13}}{4} = 2^{11}$ => d=11

Có s-d= 27-11= 16 bit

TAG	SET	WORD
s-d	d	w
16	11	5

Ví dụ 5.3.5:

Giả thiết rằng máy tính có 128KB Cache tổ chức theo kiểu ánh xạ liên kết tập hợp 4 line, Cache có tất cả 1024 set tính từ s0-s1023. Địa chỉ bộ nhớ chính là 32 bit và được đánh địa chỉ cho từng byte

a) Tính số bit cho từng trường địa chỉ cho từng bộ nhớ Cache

b) Hãy xác định byte nhớ địa chỉ 003D024AF (H) được xác định vào set nào trong bộ nhớ Cache?

Hướng dẫn:

a) Bộ nhớ chính= 32 bit => n=32 bit

Cache = 128 KB= 2^{17} byte => m= 17 bit

Theo giả thiết bài cho có k=4

=> $\frac{2^r}{4} = 1024$ => r= 12

r= m-w => w= 17-12=5 bit

s= n-w= 32-5=27 bit

*) Ánh xạ trực tiếp:

TAG	LINE	WORD
s-r	r	w
15	12	5

*) Ánh xạ liên kết toàn phần

TAG	WORD
s	w
27	5

*) Ánh xạ liên kết tập hợp:

Số set= $2^d = 2^{10}$ => d=10

s-d= 27-10=17 bit

TAG	SET	WORD
s-d	d	w
17	10	5

b) Địa chỉ 003D02AF= 0000 0000 0011 1101 0000 0010 1010 1111

Các bit lần lượt được sắp xếp:

TAG	SET	WORD
17 bit	10 bit	5 bit
0000 0000 0011 1101 0	000 0010 101	0 1111

Số set = 000 0010 101= $1.2^4 + 1.2^2 + 1.2^0 = 21$.

Vậy địa chỉ 003D02AF được xác định vào S_{21}.

5.3.3. Thuật giải thay thế

Không phải lựa chọn

Mỗi Block chỉ ánh xạ vào một Line xác định

Thay thế Block ở Line đó.

Được thực hiện bằng phần cứng.

Random: Thay thế ngẫu nhiên

FIFO (First In First Out): Thay thế Block nào nằm lâu nhất trong Set đó.

LFU (Least Frequently Used): Thay thế Block nào trong Set có số lần truy nhập ít nhất trong cùng một khoảng thời gian.

LRU (Least Recently Used): Thay thế Block ở trong Set tương ứng có thời gian lâu nhất không được tham chiếu tới.

Tối ưu nhất trong thuật giải thay thế là LRU

5.3.4. Phương pháp ghi dữ liệu cache hit

Ghi xuyên qua (Write - throught): Ghi cả cache và cả bộ nhớ chính

Tốc độ chậm

Ghi trả sau (Write - back): Chỉ ghi ra cache

Tốc độ nhanh

Khi Block trong cache bị thay thế cần phải ghi trả cả Block về bộ nhớ chính

5.3.5. Cache trên các bộ xử lý Intel

- 80486: 8KB cache L1 trên chip
- Pentium: Có 2 cache L1 trên chip.

Cache lệnh = 8Kb và Cache dữ liệu = 8Kb

- Pentium 4 (2000): Có hai mức cache L1 và L2 trên chip

Cache L1: Mỗi cache 8 Kb

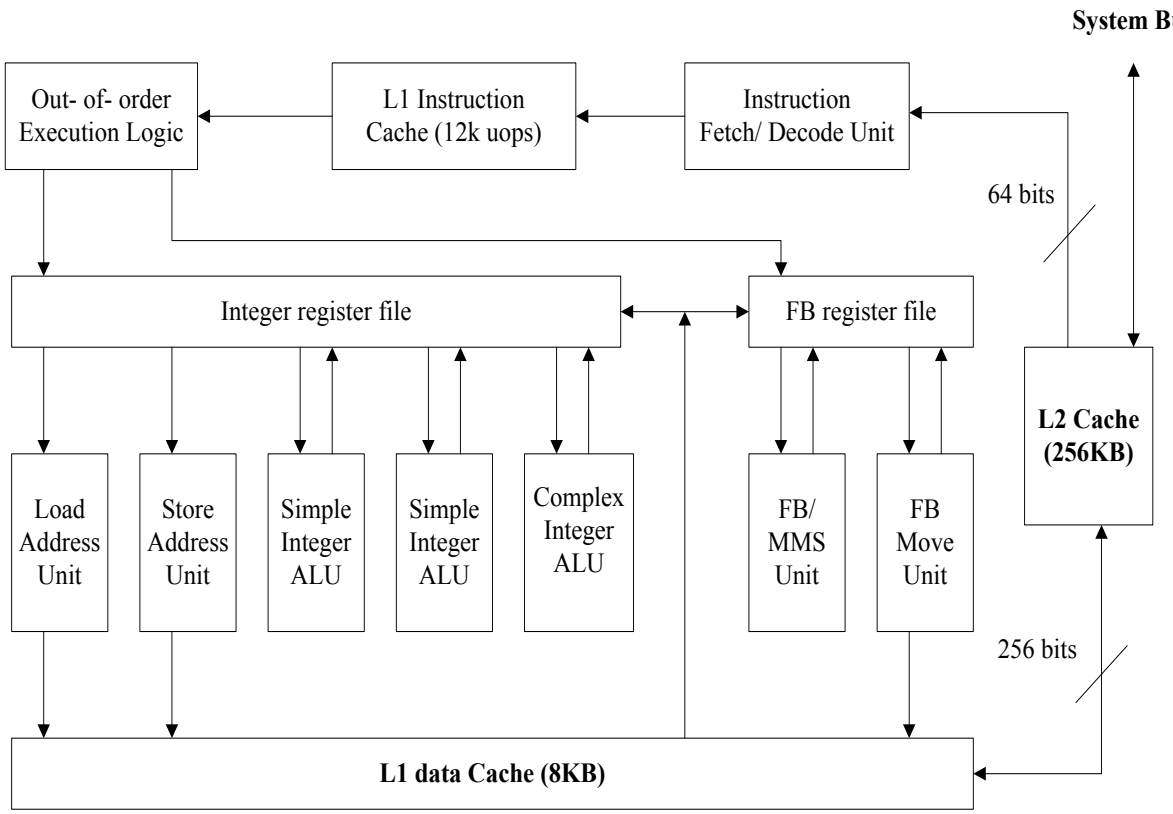
Kích thước Line = 64 byte

Ánh xạ liên kết tập hợp 4 đường

Cache L2: Mỗi cache 256 Kb

Kích thước Line = 128 byte

Ánh xạ liên kết tập hợp 8 đường.



Hình 5-15. Sơ đồ Pentium 4

5.4. BỘ NHỚ NGOÀI

5.4.1. Đĩa từ

Dù rằng công nghệ mới không ngừng phát minh nhiều loại bộ phận lưu trữ một lượng thông tin lớn thì đĩa từ vẫn giữ vị trí quan trọng từ năm 1965. Đĩa từ có hai nhiệm vụ trong máy tính.

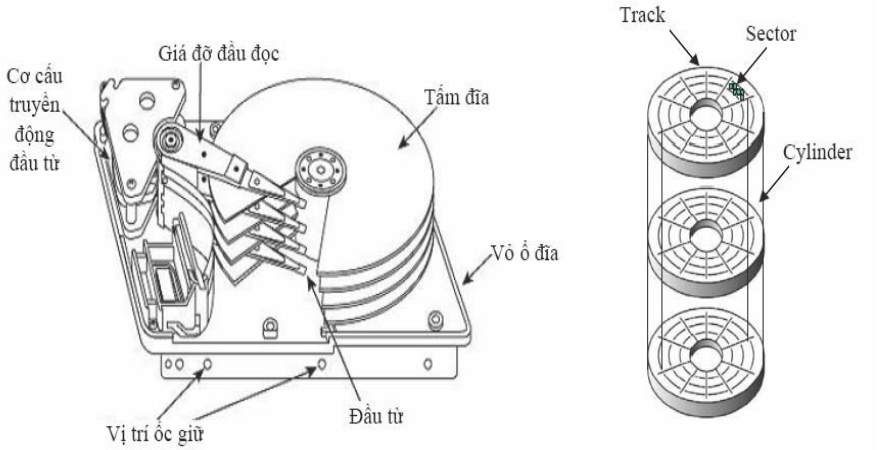
- Lưu trữ dài hạn các tập tin.
- Thiết lập một cấp bộ nhớ bên dưới bộ nhớ trong để làm bộ nhớ ảo lúc chạy chương trình.

Do đĩa mềm dần được các thiết bị lưu trữ khác có các tính năng ưu việt hơn nên chúng ta không xét đến thiết bị này trong chương trình mà chỉ nói đến đĩa cứng. Trong tài liệu này mô tả một cách khái quát cấu tạo, cách vận hành cũng như đề cập đến các tính chất quan trọng của đĩa cứng.

Một đĩa cứng chứa nhiều lớp đĩa (từ 1 đến 4) quay quanh một trục khoảng 3.600 - 15.000 vòng mỗi phút. Các lớp đĩa này được làm bằng kim loại với hai mặt được phủ một chất từ tính. Đường kính của đĩa thay đổi từ 1,3 inch đến 8 inch. Mỗi mặt của một lớp đĩa được chia thành nhiều đường tròn đồng trục gọi là *rãnh*. Thông thường mỗi mặt của một lớp đĩa có từ 10.000 đến gần 30.000 rãnh. Mỗi rãnh được chia thành nhiều *cung* (sector) dùng chứa thông tin. Một rãnh có thể chứa từ 64 đến 800 cung. Cung là đơn vị nhỏ nhất mà máy tính có thể đọc hoặc viết (thông thường khoảng 512 bytes). Chuỗi thông tin ghi trên mỗi cung gồm có: số thứ tự của cung, một khoảng trống, số liệu của cung đó bao gồm cả các mã sửa lỗi, một khoảng trống, số thứ tự của cung tiếp theo [4].

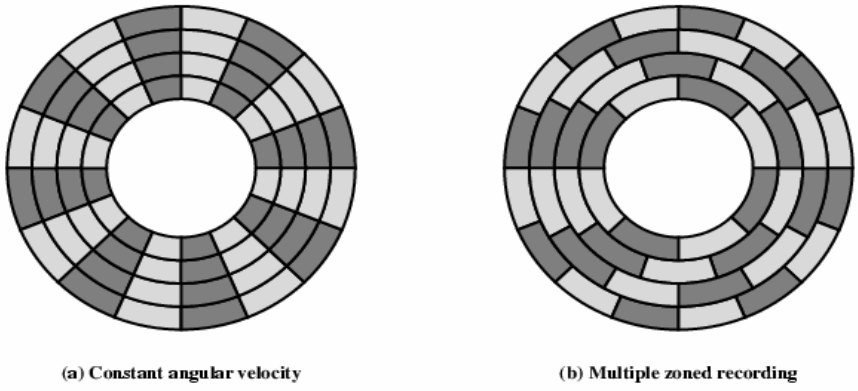
Với kỹ thuật ghi mật độ không đều, tất cả các rãnh đều có cùng một số cung,

điều này làm cho các cung dài hơn ở các rãnh xa trục quay có mật độ ghi thông tin thấp hơn mật độ ghi trên các cung nằm gần trục quay.



Hình 5-16. Cấu tạo của một đĩa cứng

Với công nghệ ghi với mật độ đều, người ta cho ghi nhiều thông tin hơn ở các rãnh xa trục quay. Công nghệ ghi này ngày càng được dùng nhiều với sự ra đời của các chuẩn giao diện thông minh như chuẩn SCSI.



Mật độ ghi không đều **Mật độ ghi đều**

Hình 5-17. Mật độ ghi đĩa

Để đọc hoặc ghi thông tin vào một cung, ta dùng một đầu đọc ghi di động áp vào mỗi mặt của mỗi lớp đĩa. Các đầu đọc/ghi này được gắn chặt vào một thanh làm cho chúng cùng di chuyển trên một đường bán kính của mỗi lớp đĩa và như thế tất cả các đầu này đều ở trên những rãnh có cùng bán kính của các lớp đĩa. Từ "*trụ*" (cylinder) được dùng để gọi tất cả các rãnh của các lớp đĩa có cùng bán kính và nằm trên một hình trụ. Người ta luôn muốn đọc nhanh đĩa từ nên thông thường ổ đĩa đọc nhiều hơn số dữ liệu cần đọc; người ta nói đây là cách đọc trước. Để quản lý các phức tạp khi kết nối (hoặc ngưng kết nối) lúc đọc (hoặc ghi) thông tin, và việc đọc trước, ổ đĩa cần có bộ điều khiển đĩa. Công nghiệp chế tạo đĩa từ tập trung vào việc nâng cao dung lượng của đĩa mà đơn vị đo lường là mật độ trên một đơn vị bề mặt.

Bảng 5-2. Bảng thông số kỹ thuật đĩa cứng

Dung lượng tối đa	có thể đạt 500GB
Số lượng đầu đọc	1-8
Số tấm ghi (đĩa)	1-4
Cache (bộ đệm)	2-16 MB
Số cung (Sector- 512 bytes/ sector)	xxx,xxx,xxx
Tốc độ quay đĩa (RPM)	3600-15000
Mật độ	có thể đạt 95 Gb/in^2
Mật độ rãnh (TP1- Max Tracks/ Inch)	có thể đạt 120,000
Mật độ ghi BP1 (Max Bits/Inch)	có thể đạt 702,000
Tốc độ dữ liệu tối đa (Internal)	có thể đạt 900 Mb/s
Tốc độ truyền dữ liệu với ngoại vi	có thể đạt 320 Mb/s
Thời gian chuyển track R/W	có thể đạt 15ms
Thời gian quay nửa vùng	có thể đạt 6 ms

5.4.2. Đĩa quang

Các thiết bị lưu trữ quang rất thích hợp cho việc phát hành các sản phẩm văn hoá, sao lưu dữ liệu trên các hệ thống máy tính hiện nay. Ra đời vào năm 1978, đây là sản phẩm của sự hợp tác nghiên cứu giữa hai công ty Sony và Philips trong công nghiệp giải trí. Từ năm 1980 đến nay, công nghiệp đĩa quang phát triển mạnh trong cả hai lĩnh vực giải trí và lưu trữ dữ liệu máy tính. Quá trình đọc thông tin dựa trên sự phản chiếu của các tia laser năng lượng thấp từ lớp lưu trữ dữ liệu. Bộ phận tiếp nhận ánh sáng sẽ nhận biết được những điểm mà tại đó tia laser bị phản xạ mạnh hay biến mất do các vết khắc (pit) trên bề mặt đĩa. Các tia phản xạ mạnh chỉ ra rằng tại điểm đó không có lỗ khắc và điểm này được gọi là điểm nền (land). Bộ nhận ánh sáng trong ổ đĩa thu nhận các tia phản xạ và khuếch tán được khúc xạ từ bề mặt đĩa. Khi các nguồn sáng được thu nhận, bộ vi xử lý sẽ dịch các mẫu sáng thành các bit dữ liệu hay âm thanh. Các lỗ trên CD sâu 0,12micron và rộng 0,6 micron (1 micron bằng một phần ngàn mm). Các lỗ này được khắc theo một track hình xoắn ốc với khoảng cách 1,6 micron giữa các vòng, khoảng 16.000 track/inch. Các lỗ (pit) và nền (land) kéo dài khoảng 0,9 đến 3,3 micron. Track bắt đầu từ phía trong và kết thúc ở phía ngoài theo một đường khép kín các rìa đĩa 5mm. Dữ liệu lưu trên CD thành từng khối, mỗi khối chứa 2.352 byte. Trong đó, 304 byte chứa các thông tin về bit đồng bộ, bit nhận dạng (ID), mã sửa lỗi (ECC), mã phát hiện lỗi (EDC) [4].

Còn lại 2.048 byte chứa dữ liệu. Tốc độ đọc chuẩn của CD-ROM là 75 khối/s hay 153.600 byte/s hay 150KB/s (1X).

Dưới đây là một số loại đĩa quang thông dụng:

+ CD (Compact Disk): Đĩa quang không thể xoá được, dùng trong công nghiệp giải trí (các đĩa âm thanh được số hoá). Chuẩn đĩa có đường kính 12 cm, âm thanh phát từ đĩa khoảng 60 phút (không dừng).

+ CD-ROM (Compact Disk Read Only Memory): Đĩa không xoá dùng để chứa các dữ liệu máy tính. Chuẩn đĩa có đường kính 12 cm, lưu trữ dữ liệu hơn 650 MB.

+ CD-R (CD-Recordable): Giống như đĩa CD, đĩa mới chưa có thông tin, người dùng có thể ghi dữ liệu lên đĩa một lần và đọc được nhiều lần, không xoá được.

+ CD-RW (CD-Rewritable): có thể ghi và đọc dữ liệu nhiều lần.

+ DVD (Digital Video Disk – Digital Versatile Disk): Đĩa chứa các hình ảnh video

được số hóa. Kích thước đĩa có hai loại: 8cm và 12cm. Đĩa DVD có thể chứa dữ liệu trên cả hai mặt đĩa, dung lượng tối đa lên đến 17GB.

Tốc độ đọc chuẩn của DVD – ROM (1X) là 1.3 MB/S (1X của DVD tương đương khoảng 9X của CD ROM).

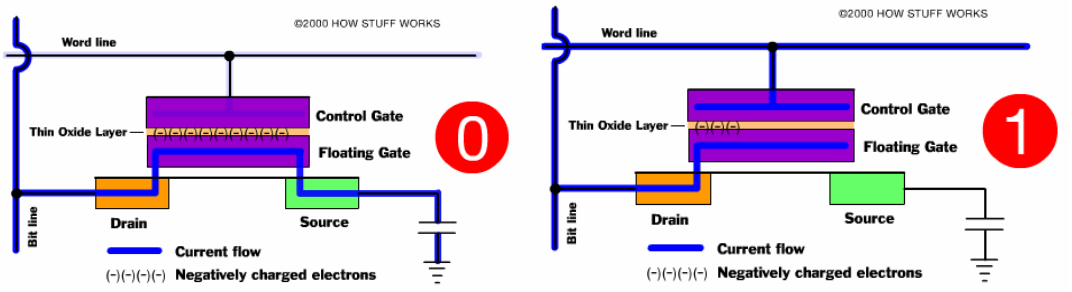
Bảng 5-3. So sánh một số thông số của hai loại đĩa CDROM và DVDROM

Đặc trưng	CDROM	DVDROM
Kích thước Pit	0.834 micron	0.4 micron
Khoảng cách rãnh	1.6 micron	0.74 micron
Số lớp dữ liệu trên đĩa	1 lớp	2 lớp
Số mặt đĩa	1 mặt	1 - 2 mặt
Dung lượng	640-700MB	1.36 - 17 GB
Độ phân giải phim	VCD= 320x200	720x640

Với các đặc tính của đĩa quang, giá thành ngày càng thấp, được xem như một phương tiện thích hợp để phân phối các phần mềm cho máy vi tính. Ngoài ra, đĩa quang còn được dùng để lưu trữ lâu dài các dữ liệu thay thế cho băng từ.

5.4.3. Các loại thẻ nhớ

Hiện nay, thẻ nhớ là một trong những công nghệ mới nhất được dùng làm thiết bị lưu trữ. Thẻ nhớ flash là một dạng bộ nhớ bán dẫn EEPROM(công nghệ dùng để chế tạo các chip BIOS trên các vỉ mạch chính), được cấu tạo bởi các hàng và các cột. Mỗi vị trí giao nhau là một ô nhớ gồm có hai transistor, hai transistor này cách nhau bởi một lớp ô-xít mỏng. Một transistor được gọi là *floating gate* và transistor còn lại được gọi là *control gate*. Floating gate chỉ có thể nối kết với hàng (word line) thông qua control gate. Khi đường kết nối được thiết lập, bit có giá trị 1. Để chuyển sang giá trị 0 theo một qui trình có tên *Fowler-Nordheim tunneling*. Tốc độ, yêu cầu về dòng điện cung cấp thấp và đặc biệt với kích thước nhỏ gọn của các loại thẻ nhớ làm cho kiểu bộ nhớ này được dùng rộng rãi trong công nghệ lưu trữ và giải trí hiện nay [4,7].



Hình 5-18. Minh hoạ hai trạng thái của một bit nhớ trong thẻ nhớ

5.4.4. Băng từ

Băng từ có cùng công nghệ với các đĩa từ nhưng khác đĩa từ hai điểm:

- Việc thâm nhập vào đĩa từ là ngẫu nhiên còn việc thâm nhập vào băng từ là tuần tự. Như vậy việc tìm thông tin trên băng từ mất nhiều thời gian hơn việc tìm thông tin trên đĩa từ.

- Đĩa từ có dung lượng hạn chế còn băng từ gồm có nhiều cuộn băng có thể lấy ra khỏi máy đọc băng nên dung lượng của băng từ là rất lớn (hàng trăm GB). Với chi phí thấp, băng từ vẫn còn được dùng rộng rãi trong việc lưu trữ dữ liệu dự phòng.

Các băng từ có chiều rộng thay đổi từ 0,38cm đến 1,27 cm được đóng thành cuộn và được chứa trong một hộp bảo vệ. Dữ liệu ghi trên băng từ có cấu trúc gồm một số các rãnh song song theo chiều dọc của băng.

Có hai cách ghi dữ liệu lên băng từ:

Ghi nối tiếp: với kỹ thuật ghi xoắn ốc, dữ liệu ghi nối tiếp trên một rãnh của băng từ, khi kết thúc một rãnh, băng từ sẽ quay ngược lại, đầu từ sẽ ghi dữ liệu trên rãnh mới tiếp theo nhưng với hướng ngược lại. Quá trình ghi cứ tiếp diễn cho đến khi đầy băng từ.

Ghi song song: để tăng tốc độ đọc-ghi dữ liệu trên băng từ, đầu đọc - ghi có thể đọc-ghi một số rãnh kề nhau đồng thời. Dữ liệu vẫn được ghi theo chiều dọc băng từ nhưng các khối dữ liệu được xem như ghi trên các rãnh kề nhau. Số rãnh ghi đồng thời trên băng từ thông thường là 9 rãnh (8 rãnh dữ liệu - 1byte và một rãnh kiểm tra lỗi).

5.4.5. Biện pháp an toàn dữ liệu khi lưu trữ thông tin trong đĩa từ

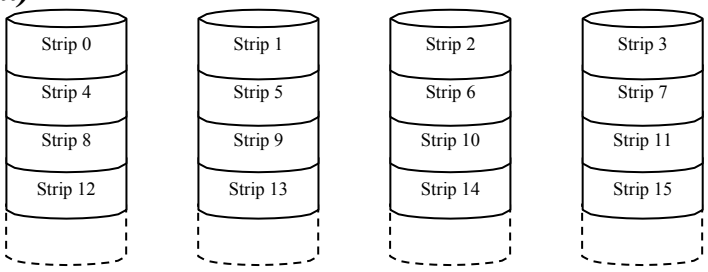
Người ta thường chú trọng đến sự an toàn trong lưu giữ thông tin ở đĩa từ hơn là sự an toàn của thông tin trong bộ xử lý. Bộ xử lý có thể hư mà không làm tổn hại đến thông tin. Ổ đĩa của máy tính bị hư có thể gây ra các thiệt hại rất to lớn.

Một phương pháp giúp tăng cường độ an toàn của thông tin trên đĩa từ là dùng một mảng đĩa từ. Mảng đĩa từ này được gọi là *Hệ thống đĩa dự phòng* (*RAID - Redundant Array of Independent Disks*). Cách lưu trữ dư thông tin làm tăng giá tiền và sự an toàn (ngoại trừ RAID 0). Cơ chế RAID có các đặc tính sau:

- RAID là một tập hợp các ổ đĩa cứng (vật lý) được thiết lập theo một kỹ thuật mà hệ điều hành chỉ "nhìn thấy" chỉ là một ổ đĩa (logic) duy nhất.
- Với cơ chế đọc/ghi thông tin diễn ra trên nhiều đĩa (ghi đan chéo hay soi gương).
- Trong mảng đĩa có lưu các thông tin kiểm tra lỗi dữ liệu; do đó, dữ liệu có thể được phục hồi nếu có một đĩa trong mảng đĩa bị hư hỏng.

Tuỳ theo kỹ thuật thiết lập, RAID có thể có các mức sau:

5.4.5.1. RAID 0. (Strip – Tạo lát)



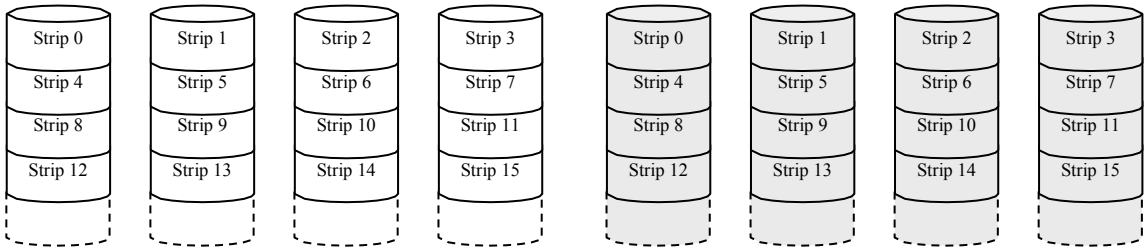
Hình 5-19. RAID 0

Thực ra, kỹ thuật này không nằm trong số các kỹ thuật có cơ chế an toàn dữ liệu. Khi mảng được thiết lập theo RAID 0, ổ đĩa logic có được (mà hệ điều hành nhận biết) có dung dượng bằng tổng dung lượng của các ổ đĩa thành viên. Điều này giúp cho người dùng có thể có một ổ đĩa logic có dung lượng lớn hơn rất nhiều so với dung lượng thật của ổ đĩa vật lý cùng thời điểm. Dữ liệu được ghi phân tán trên tất cả các đĩa trong mảng. Đây chính là sự khác biệt so với việc ghi dữ liệu trên các đĩa riêng lẻ bình thường bởi vì thời gian đọc-ghi dữ liệu trên đĩa tỉ lệ nghịch với số đĩa có trong tập hợp (số đĩa trong tập hợp càng nhiều, thời gian đọc – ghi dữ liệu càng nhanh). Tính chất này của RAID 0 thật sự hữu ích trong các ứng dụng yêu cầu nhiều thâm nhập đĩa với dung lượng lớn, tốc độ cao (đa phương tiện, đồ hoạ,…). Tuy nhiên, như đã nói ở trên, kỹ thuật này không có cơ chế an toàn dữ liệu, nên khi có bất kỳ một hư hỏng nào trên một đĩa thành viên trong mảng cũng sẽ dẫn đến việc mất dữ liệu toàn bộ trong mảng đĩa. Xác suất hư hỏng đĩa tỉ lệ thuận với số lượng đĩa được thiết lập trong RAID 0. RIAD 0 có thể được thiết lập bằng phần cứng (RAID controller) hay phần mềm

(Stripped Applications)

5.4.5.2. RAID 1 (Mirror - Đĩa gương)

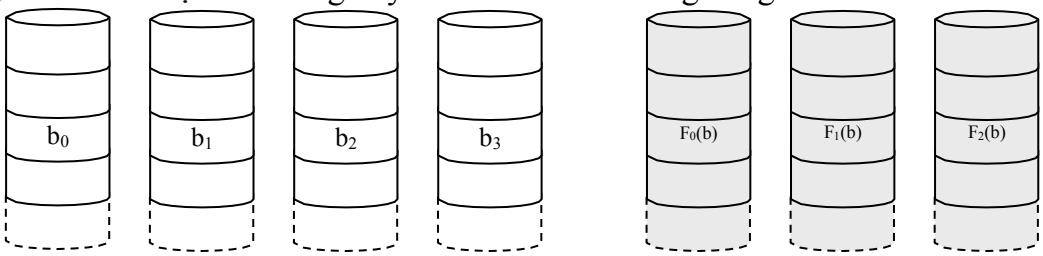
Phương cách thông thường tránh mất thông tin khi ổ đĩa bị hư là dùng đĩa gương, tức là dùng 2 đĩa. Khi thông tin được viết vào một đĩa, thì nó cũng được viết vào đĩa gương và như vậy luôn có một bản sao của thông tin. Trong cơ chế này, nếu một trong hai đĩa bị hư thì đĩa còn lại được dùng bình thường. Việc thay thế một đĩa mới (cung thông số kỹ thuật với đĩa hư hỏng) và phục hồi dữ liệu trên đĩa đơn giản. Căn cứ vào dữ liệu trên đĩa còn lại, sau một khoảng thời gian, dữ liệu sẽ được tái tạo trên đĩa mới (rebuild). RAID 1 cũng có thể được thiết lập bằng phần cứng (RAID controller) hay phần mềm (Mirror Applications) với chi phí khá lớn, hiệu suất sử dụng đĩa không cao (50%).



Hình 5-20. RAID 1

5.4.5.3. RAID 2

Dùng kỹ thuật truy cập đĩa song song, tất cả các đĩa thành viên trong RAID đều được đọc khi có một yêu cầu từ ngoại vi. Một mã sửa lỗi (ECC) được tính toán dựa vào các dữ liệu được ghi trên đĩa lưu dữ liệu, các bit được mã hoá được lưu trong các đĩa dùng làm đĩa kiểm tra. Khi có một yêu cầu dữ liệu, tất cả các đĩa được truy cập đồng thời. Khi phát hiện có lỗi, bộ điều khiển nhận dạng và sửa lỗi ngay mà không làm giảm thời gian truy cập đĩa. Với một thao tác ghi dữ liệu lên một đĩa, tất cả các đĩa dữ liệu và đĩa sửa lỗi đều được truy cập để tiến hành thao tác ghi. Thông thường, RAID 2 dùng mã Hamming để thiết lập cơ chế mã hoá, theo đó, để mã hoá dữ liệu được ghi, người ta dùng một bit sửa lỗi và hai bit phát hiện lỗi. RAID 2 thích hợp cho hệ thống yêu cầu giảm thiểu được khả năng xảy ra nhiều đĩa hư hỏng cùng lúc.

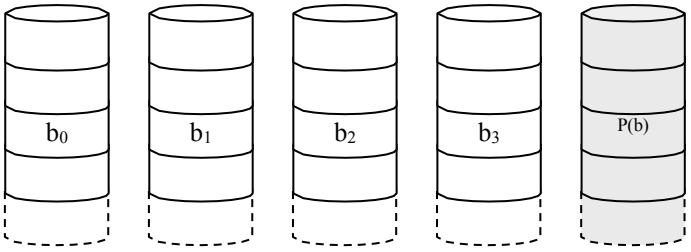


Hình 5-21. RAID 2

5.4.5.4. RAID 3

Dùng kỹ thuật ghi song song, trong kỹ thuật này, mảng được thiết lập với yêu cầu tối thiểu là 3 đĩa có các thông số kỹ thuật giống nhau, chỉ một đĩa trong mảng được dùng để lưu các thông tin kiểm tra lỗi (parity bit). Như vậy, khi thiết lập RAID 3, hệ điều hành nhận biết được một đĩa logic có dung lượng n-1/n (n: số đĩa trong mảng). Dữ liệu được chia nhỏ và ghi đồng thời trên n-1 đĩa và bit kiểm tra chẵn lẻ được ghi trên đĩa dùng làm đĩa chứa bit parity – chẵn lẻ đan chéo ở mức độ bít. Bít chẵn lẻ là một bít mà người ta thêm vào một tập hợp các bít làm cho số bít có trị số 1 (hoặc 0) là chẵn

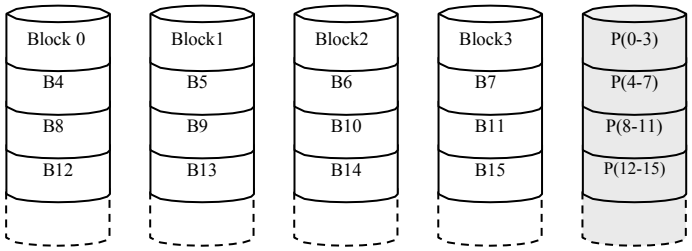
(hay lẻ). Thay vì có một bản sao hoàn chỉnh của thông tin gốc trên mỗi đĩa, người ta chỉ cần có đủ thông tin để phục hồi thông tin đã mất trong trường hợp có hỏng ổ đĩa. Khi một đĩa bất kỳ trong mảng bị hư, hệ thống vẫn hoạt động bình thường. Khi thay thế một đĩa mới vào mảng, căn cứ vào dữ liệu trên các đĩa còn lại, hệ thống tái tạo thông tin. Hiệu suất sử dụng đĩa cho cách thiết lập này là n-1/n. RAID 3 chỉ có thể được thiết lập bằng phần cứng (RAID controller).



Hình 5-22. RAID 3

5.4.5.5. RAID 4

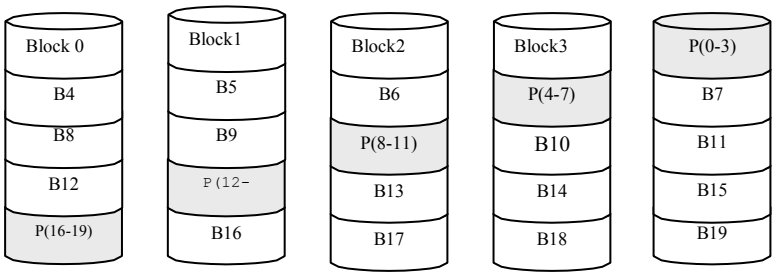
Từ RAID 4 đến RAID 6 dùng kỹ thuật truy cập các đĩa trong mảng độc lập. Trong một mảng truy cập độc lập, mỗi đĩa thành viên được truy xuất độc lập, do đó mảng có thể đáp ứng được các yêu cầu song song của ngoại vi. Kỹ thuật này thích hợp với các ứng dụng yêu cầu nhiều ngoại vi là các ứng dụng yêu cầu tốc độ truyền dữ liệu cao. Trong RAID 4, một đĩa dùng để chứa các bit kiểm tra được tính toán từ dữ liệu được lưu trên các đĩa dữ liệu. Khuyết điểm lớn nhất của RAID 4 là bị nghẽn cổ chai tại đĩa kiểm tra khi có nhiều yêu cầu đồng thời từ các ngoại vi.



Hình 5-23. RAID 4

5.4.5.6. RAID 5

Yêu cầu thiết lập giống như RAID 4, dữ liệu được ghi từng khối trên các đĩa thành viên, các bit chẵn lẻ được tính toán mức độ khối được ghi trải đều lên trên tất cả các ổ đĩa trong mảng. Tương tự RAID 4, khi một đĩa bất kỳ trong mảng bị hư hỏng, hệ thống vẫn hoạt động bình thường. Khi thay thế một đĩa mới vào mảng, căn cứ vào dữ liệu trên các đĩa còn lại, hệ thống tái tạo thông tin. Hiệu suất sử dụng đĩa cho cách thiết lập này là n-1/n. RAID 5 chỉ có thể được thiết lập bằng phần cứng (RAID controller). Cơ chế này khắc phục được khuyết điểm đã nêu trong cơ chế RAID 4.

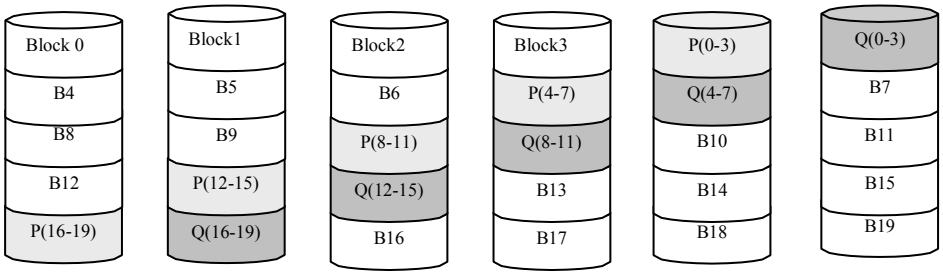


Hình 5-24. RAID 5

5.4.5.7. RAID 6

Trong kỹ thuật này, cần có n+2 đĩa trong mảng. Trong đó, n đĩa dữ liệu và 2 đĩa riêng biệt để lưu các khối kiểm tra. Một trong hai đĩa kiểm tra dùng cơ chế kiểm tra như trong RAID 4&5, đĩa còn lại kiểm tra độc lập theo một giải thuật kiểm tra. Qua đó, nó có thể phục hồi được dữ liệu ngay cả khi có hai đĩa dữ liệu trong mảng bị hư hỏng.

Hiện nay, RAID 0,1,5 được dùng nhiều trong các hệ thống. Các giải pháp RAID trên đây (trừ RAID 6) chỉ đảm bảo an toàn dữ liệu khi có một đĩa trong mảng bị hư hỏng. Ngoài ra, các hư hỏng dữ liệu do phần mềm hay chủ quan của con người không được đề cập trong chương trình. Người dùng cần phải có kiến thức đầy đủ về hệ thống để các hệ thống thông tin hoạt động hiệu quả và an toàn.



Hình 5-25. RAID 6

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 5

Câu 1. Hãy trình bày hệ thống phân cấp trong bộ nhớ:

a) Vẽ sơ đồ và giải thích hệ thống phân cấp bộ nhớ?

b) Thành phần nào trong cấu trúc phân cấp bộ nhớ giúp làm tăng hiệu năng hệ thống và làm giảm giá thành sản xuất của máy tính?

c) Giải thích vì sao bộ nhớ RAM được coi là khả biến còn ổ cứng là bất biến?

Câu 2. Hệ thống nhớ

a) Ý nghĩa các kỹ thuật tổ chức Cache là gì?

b) So sánh 3 phương pháp ánh xạ cache: ánh xạ trực tiếp, ánh xạ kết hợp đầy đủ và ánh xạ tập kết hợp?

c) Phương pháp ánh xạ nào trong các phương pháp trên được sử dụng nhiều nhất trong thực tế? Tại sao?

Câu 3. Hệ thống nhớ

a) Trình bày ba phương pháp thay thế dòng cache: Ngẫu nhiên, FIFO và LRU.

b) So sánh ba phương pháp trên.

c) Phương pháp nào cho hệ số trúng (hit) cao nhất? Giải thích?

Câu 4. Trình bày quá trình ghép nhiều ổ đĩa cứng vật lý thành một hệ thống ổ đĩa cứng:

a) RAID là gì?

b) Tại sao RAID có thể nâng cao được tính tin cậy và tốc độ truy nhập hệ thống lưu trữ?

c) So sánh ba loại RAID 0, RAID 1 và RAID 5

Câu 5. Trình bày các phương pháp ánh xạ bộ nhớ cache.

a) Phương pháp ánh xạ trực tiếp

b) Phương pháp ánh xạ liên kết toàn phần

c) Phương pháp ánh xạ liên kết tập hợp

Câu 6. Xây dựng Kỹ thuật phát hiện lỗi và sửa lỗi Hamming code

a) Trình bày các kỹ phát hiện và hiệu chỉnh lỗi trong bộ nhớ

b) Vẽ và giải thích sơ đồ phát hiện lỗi và hiệu chỉnh lỗi

c) Cho ví dụ minh họa mã sửa lỗi Hamming (m=4, k=3)?

Câu 7. Cho chip nhớ SRAM 4K x 8 bit

Thiết kế mô-đun nhớ 16K x 8 bit

Câu 8. Cho chip nhớ SRAM 4K x 8 bit

Thiết kế mô-đun nhớ 32K x 8 bit

Câu 9. Cho chip nhớ SRAM 4K x 4 bit

Thiết kế mô-đun nhớ 8K x 8 bit

Câu 10. Cho chip nhớ SRAM 8K x 8 bit

Thiết kế mô-đun nhớ 16K x 16

Câu 11. Chỉ dùng các chip nhớ SRAM có dung lượng 16K x 8 bit và các bộ giải mã 2→4, hãy thiết kế module nhớ có dung lượng 256KB.

Câu 12. Cho bộ nhớ chính có 16 từ nhớ chia thành các Block, mỗi Block 2 từ nhớ. Cache có dung lượng 8 từ nhớ. Xác định từ địa chỉ để quản lý bộ nhớ cache theo phương pháp ánh xạ trực tiếp.

Câu 13. Một máy tính có dung lượng bộ nhớ chính là 64MByte được tổ chức theo các block nhớ. Bộ nhớ cache có dung lượng là 64Kbyte, kích thước đường là 64 byte. Tính dạng địa chỉ truy cập cache trong trường hợp ánh xạ liên kết hoàn toàn.

Câu 14. Giả thiết rằng máy tính có 128KB cache tổ chức theo kiểu ánh xạ liên kết tập hợp 4 line. Cache có tất cả là 1024 Set từ S0->S1023. Địa chỉ bộ nhớ chính là 32 bít và đánh địa chỉ cho từng byte.

Xác định byte nhớ có địa chỉ 115B14CF (H) được ánh xạ vào Set nào của Cache?

Câu 15. Cho không gian địa chỉ bộ nhớ 32GB, dung lượng bộ nhớ Cache là 6MB, kích thước line là 32 byte. Xác định số bit của trường địa chỉ theo phương pháp:

a) Ánh xạ trực tiếp
b) Ánh xạ liên kết toàn phần
c) Ánh xạ liên kết tập hợp.

Câu 16. Cho không gian địa chỉ bộ nhớ 64GB, dung lượng bộ nhớ Cache là 8MB, kích thước line là 32 byte. Xác định số bit của trường địa chỉ theo phương pháp:

a) Ánh xạ trực tiếp
b) Ánh xạ liên kết toàn phần
c) Ánh xạ liên kết tập hợp.

Câu 17. Cho không gian địa chỉ bộ nhớ 128GB, dung lượng bộ nhớ Cache là 20MB, kích thước line là 32 byte. Xác định số bit của trường địa chỉ theo phương pháp:

a) Ánh xạ trực tiếp
b) Ánh xạ liên kết toàn phần
c) Ánh xạ liên kết tập hợp.

Chương 6
HỆ THỐNG VÀO RA

Trang bị cho sinh viên kiến thức về hệ thống chuyển dữ liệu giữa máy tính và thiết bị ngoại vi, các phương pháp trao đổi dữ liệu và các cách ghép nối với thiết bị ngoại vi.

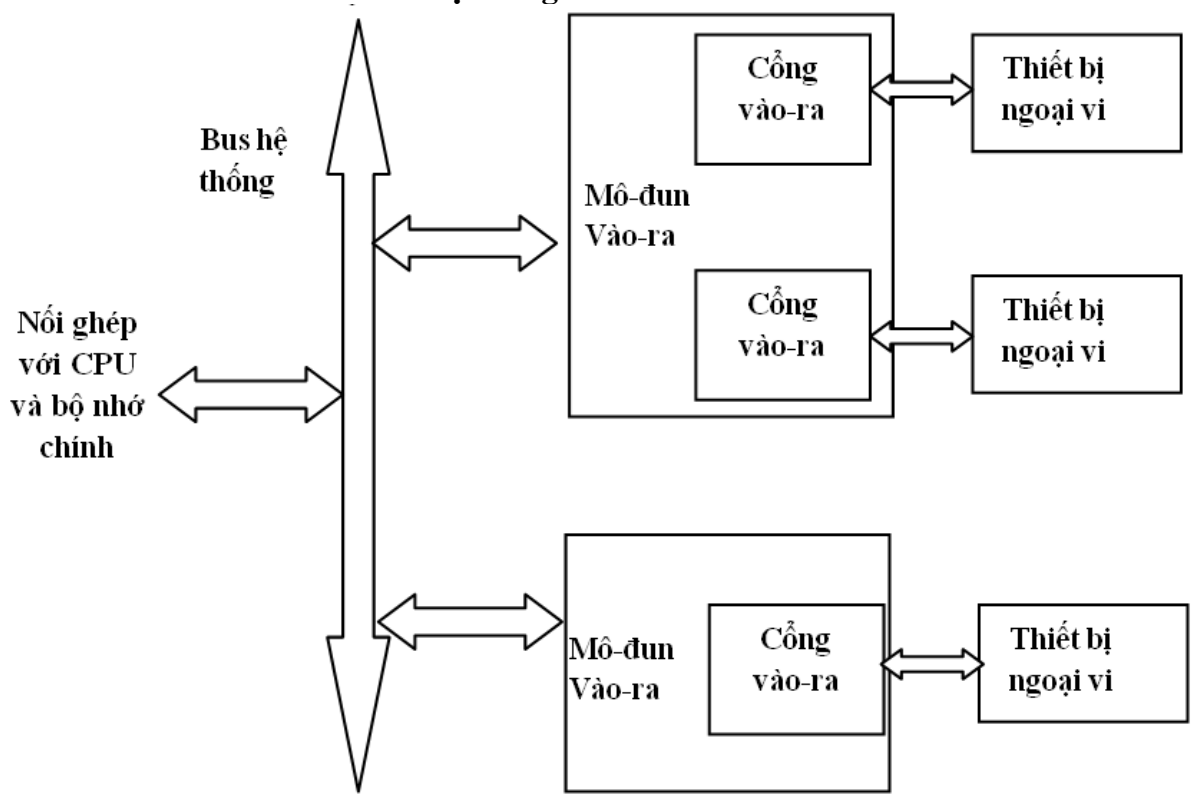
Sinh viên cần nắm vững các kiến thức về hệ thống kết nối cơ bản các bộ *phận bên trong máy tính, cách giao tiếp giữa các ngoại vi và bộ xử lý. Thông qua hệ thống BUS truyền dữ liệu.*

Trang bị cho sinh viên kiến thức về cấu trúc của hệ thống truyền dữ liệu, các phương pháp trao đổi dữ liệu song song, nối tiếp, giữa máy tính và thiết bị ngoại vi.

Sinh viên cần nắm vững các kiến thức về hệ thống kết nối cơ bản các bộ phận bên trong máy tính, các phương pháp trao đổi dữ liệu song song, nối tiếp, giao diện đa năng, giao diện cao tốc.

6.1. CẤU TRÚC CHUNG CỦA HỆ THỐNG VÀO RA

6.1.1. Cấu trúc cơ bản của hệ thống vào ra



Hình 6-1. Cấu trúc cơ bản của hệ thống vào ra

✓ Chức năng của hệ thống vào ra: Trao đổi thông tin giữa máy tính với thế giới bên ngoài.

✓ Các thao tác cơ bản: Vào dữ liệu (Input), ra dữ liệu (Ouput)

✓ Các thành phần chính: Các thiết bị ngoại vi, các mô-đun vào-ra

✓ Đặc điểm của vào-ra:

+ Tồn tại đa dạng các thiết bị ngoại vi khác nhau về: Nguyên tắc hoạt động, tốc độ, khuôn dạng dữ liệu

+ Tất cả các thiết bị ngoại vi đều chậm hơn CPU và RAM, nên cần có các mô-đun vào-ra để nối ghép các thiết bị ngoại vi với CPU và bộ nhớ chính

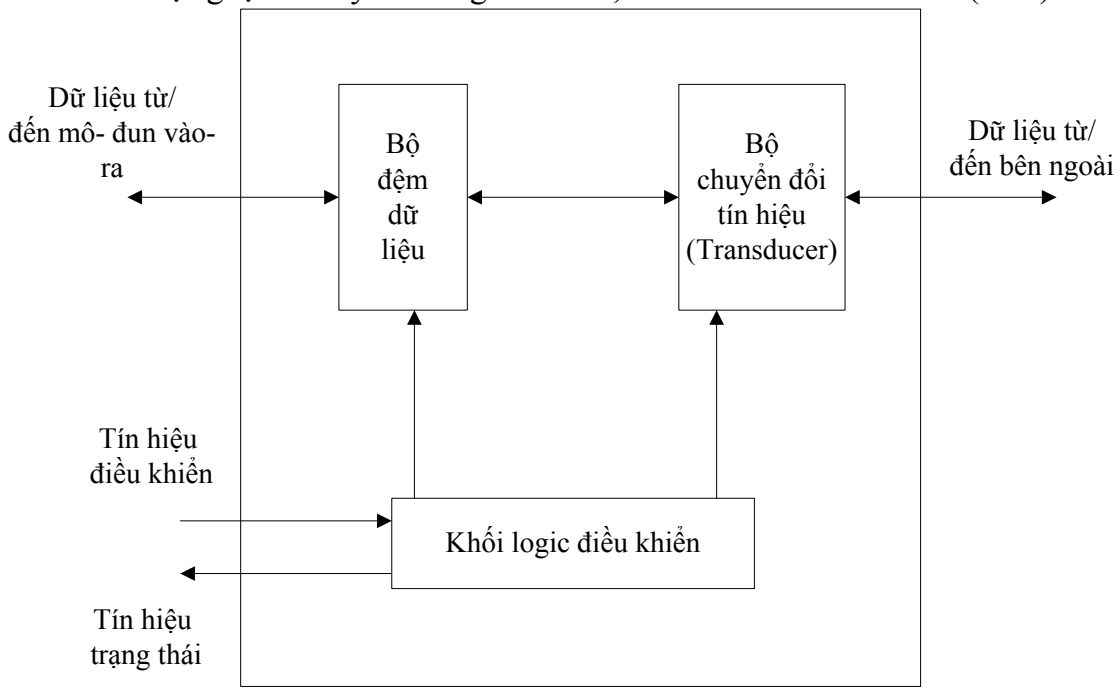
6.1.2. Các thiết bị ngoại vi

- Chức năng: Chuyển đổi dữ liệu giữa bên trong và bên ngoài máy tính
- Phân loại:

Thiết bị ngoại vi giao tiếp người với máy. Ví dụ: Bàn phím, màn hình, máy in....

Thiết bị ngoại vi giao tiếp máy với máy. Gồm các thiết bị theo dõi và kiểm tra

Thiết bị ngoại vi truyền thông. Modem, Network Interface Card (NIC).



Hình 6-2. Cấu trúc chung của thiết bị ngoại vi

- Các thành phần của thiết bị ngoại vi:

Bộ chuyển đổi tín hiệu: Chuyển đổi dữ liệu giữa bên ngoài với bên trong máy tính

Bộ đệm dữ liệu: Đệm dữ liệu khi truyền giữa mô-đun vào-ra với thiết bị ngoại vi.

Khối logic điều khiển: Điều khiển hoạt động của thiết bị ngoại vi đáp ứng theo yêu cầu từ mô-đun vào-ra.

6.1.3. Mô-đun vào-ra

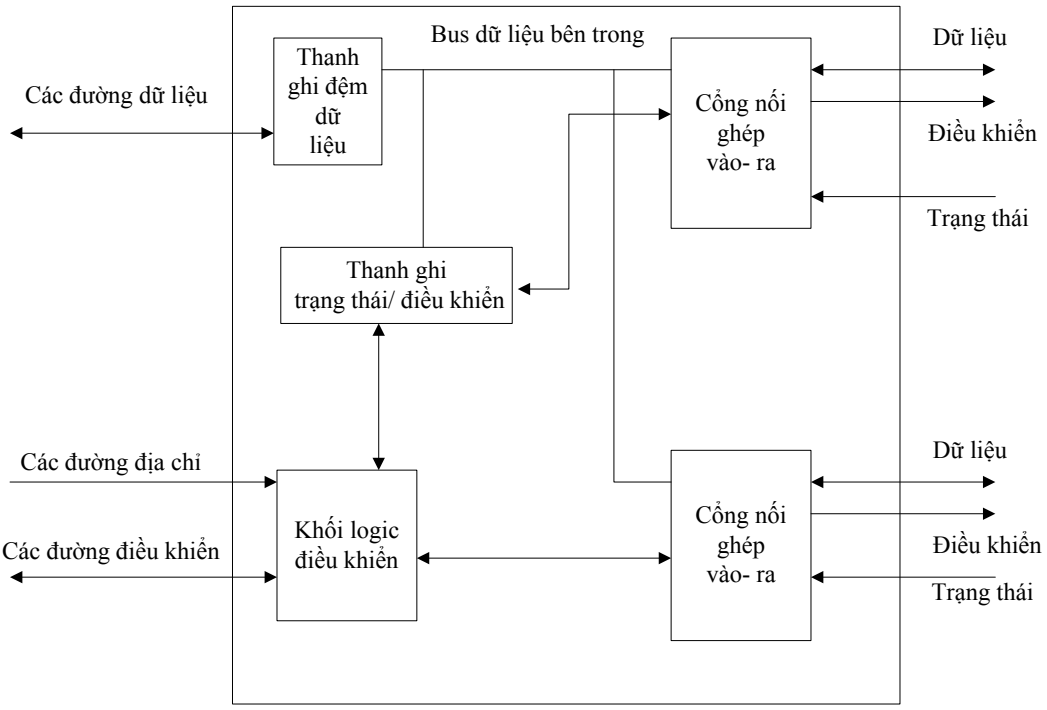
Điều khiển và định thời

Trao đổi thông tin với CPU hoặc bộ nhớ chính

Trao đổi thông tin với thiết bị ngoại vi

Đệm giữa bên trong máy tính với thiết bị ngoại vi

Phát hiện lỗi của thiết bị ngoại vi



Hình 6-3. Cấu trúc chung của mô-đun vào-ra

- Các thành phần của mô-đun vào-ra:

Thanh ghi đệm dữ liệu: Đệm dữ liệu trong quá trình trao đổi

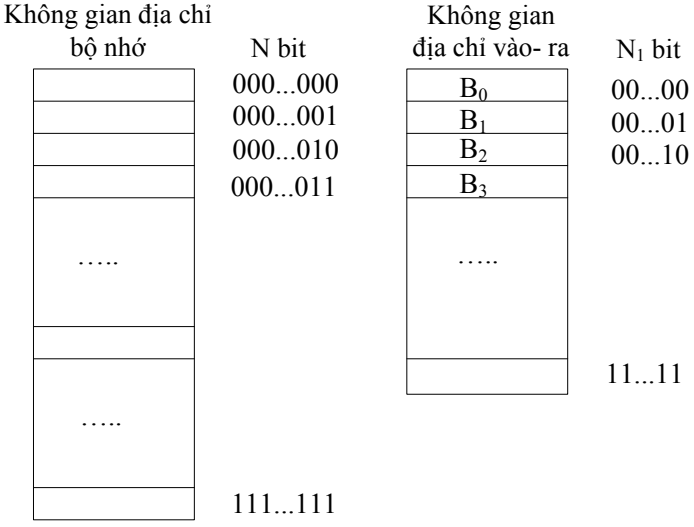
Các cổng vào ra (I/O port): Kết nối với thiết bị ngoại vi, mỗi cổng có một địa chỉ xác định

Thanh ghi trạng thái/ điều khiển: Lưu giữ thông tin trạng thái/điều khiển cho các cổng vào ra

Khối logic điều khiển: Điều khiển mô-đun vào-ra

6.1.4. Địa chỉ hóa cổng vào ra

6.1.4.1. Không gian địa chỉ của bộ xử lý



Hình 6-4. Không gian địa chỉ của bộ xử lý

- Một số bộ xử lý chỉ quản lý duy nhất một không gian địa chỉ. Không gian địa chỉ của bộ nhớ là 2^N địa chỉ.

Ví dụ: Các bộ xử lý 680x0 (Motorola): Một số bộ xử lý quản lý hai không gian địa chỉ tách biệt:

Không gian địa chỉ bộ nhớ: 2^N địa chỉ

Không gian địa chỉ vào ra: 2^{N1} địa chỉ

Có tín hiệu điều khiển phân biệt truy nhập không gian địa chỉ

Tập lệnh có các lệnh vào-ra chuyên dụng

Ví dụ: pentium (Intel)

Không gian địa chỉ bộ nhớ= 2^{32} byte = 4 GB

Không gian địa chỉ vào-ra= 2^{16} byte = 64 KB

Tín hiệu điều khiển M/$\overline{IO}$

Lệnh vào-ra chuyên dụng: In , Out

6.1.4.2. Các phương pháp địa chỉ hóa cổng vào-ra

a) Vào ra riêng biệt:

Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ vào-ra

CPU trao đổi dữ liệu với cổng vào ra thông qua các lệnh vào-ra chuyên dụng (In, OUT)

Chỉ có thể thực hiện trên các hệ thống có quản lý không gian địa chỉ vào-ra riêng biệt

b) Vào ra theo bản đồ bộ nhớ:

Cổng vào-ra được đánh địa chỉ theo không gian địa chỉ bộ nhớ

Vào-ra giống như đọc/ghi bộ nhớ

CPU trao đổi dữ liệu với cổng vào-ra thông qua các lệnh truy nhập dữ liệu bộ nhớ

Có thể thực hiện trên mọi hệ thống

6.2. CÁC PHƯƠNG PHÁP TRAO ĐỔI DỮ LIỆU

6.2.1. Vào-ra bằng chương trình

6.2.1.1. Nguyên tắc chung

CPU điều khiển trực tiếp vào-ra bằng chương trình nên cần phải lập trình vào-ra

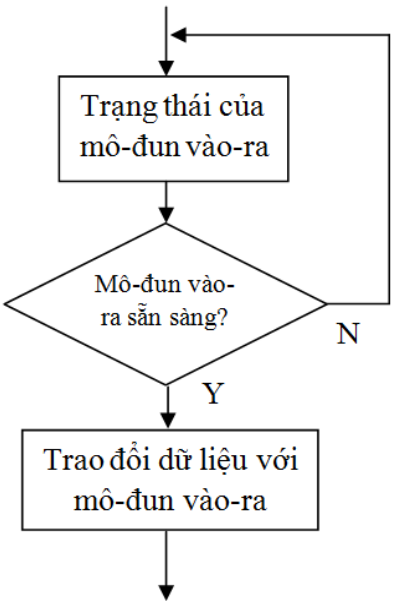
6.2.1.2. Các tín hiệu điều khiển vào-ra

- Tín hiệu điều khiển (control): Kích hoạt thiết bị ngoại vi
- Tín hiệu kiểm tra (test): Kiểm tra trạng thái của mô-đun vào-ra và thiết bị ngoại vi
- Tín hiệu điều khiển (read): Yêu cầu mô-đun vào-ra nhận dữ liệu bằng thiết bị ngoại vi và đưa vào thanh ghi đệm dữ liệu, rồi CPU nhận dữ liệu đó
- Tín hiệu điều khiển ghi (Write): Yêu cầu mô-đun vào-ra lấy dữ liệu trên bus dữ liệu đưa đến thanh ghi đệm dữ liệu và chuyển ra thiết bị ngoại vi

6.2.1.3. Các lệnh vào ra

- Vào- ra riêng biệt: Sử dụng lệnh vào-ra chuyên dụng (In, Out)
- Vào-ra theo bản đồ bộ nhớ: Sử dụng các lệnh trao đổi dữ liệu với bộ nhớ để trao đổi dữ liệu với cổng vào ra

6.2.1.4. Lưu đồ đoạn chương trình vào-ra



Hình 6-5. Lưu đồ đoạn chương trình vào-ra

6.2.1.5. Hoạt động của vào-ra bằng chương trình

- CPU yêu cầu thao tác vào-ra
- Mô-đun vào-ra thực hiện thao tác
- Mô-đun vào-ra thiết lập các bit trạng thái
- CPU kiểm tra các bit trạng thái: Nếu chưa sẵn sàng thì quay lại kiểm tra. Nếu sẵn sàn thì chuyển sang trao đổi dữ liệu với mô-đun vào-ra

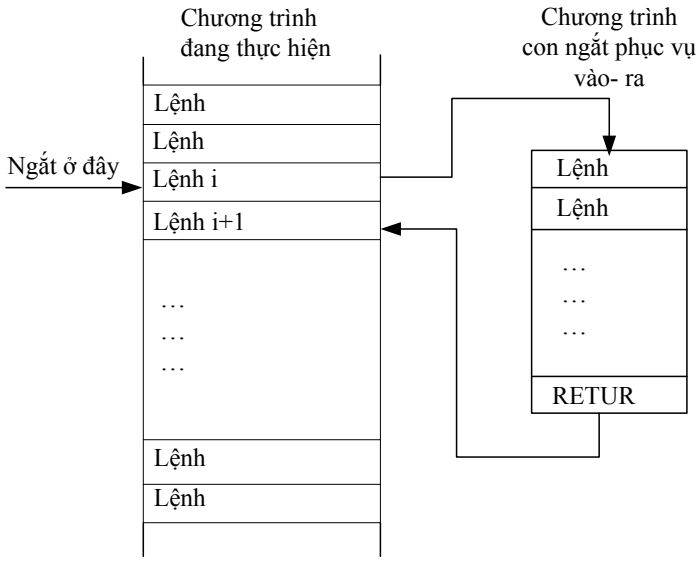
6.2.1.6. Đặc điểm của phương pháp vào-ra bằng chương trình

- Vào-ra do ý muốn của người lập trình
- CPU trực tiếp điều khiển vào-ra
- CPU đợi mô-đun vào-ra, nên tiêu tốn thời gian của CPU

6.2.2. Vào-ra điều khiển bằng ngắt

a) Nguyên tắc chung: CPU không phải đợi trạng thái sẵn sàng của mô-đun vào-ra, CPU thực hiện một chương trình nào đó

- Khi mô-đun vào-ra sẵn sàng thì nó phát tín hiệu ngắt CPU
- CPU thực hiện chương trình con vào-ra tương ứng để trao đổi dữ liệu
- CPU trở lại tiếp tục thực hiện chương trình đang bị ngắt



Hình 6-6. Vào-ra điều khiển bằng ngắt

b) Hoạt động vào dữ liệu:

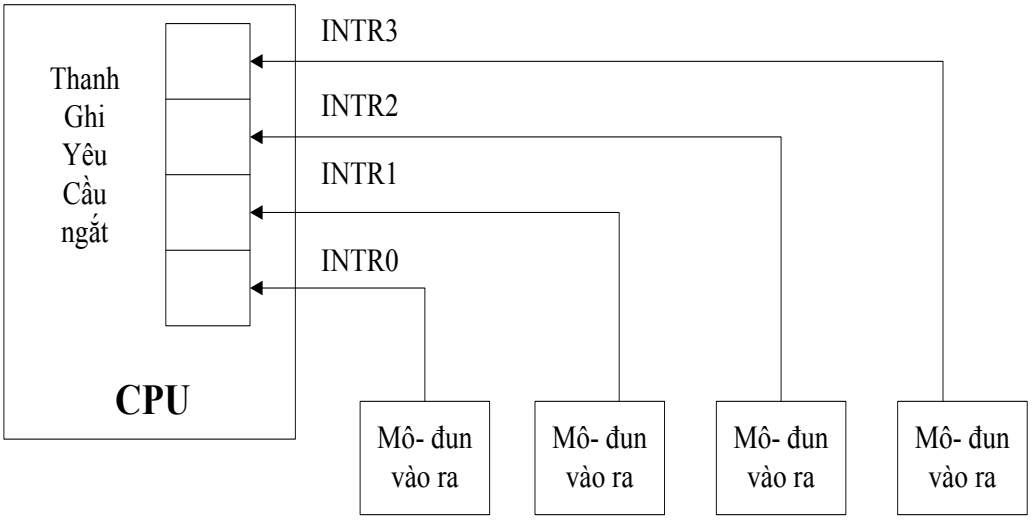
* Nhìn từ mô-đun vào-ra:
  - Mô-đun vào-ra nhận tín hiệu điều khiển đọc từ CPU
  - Mô-đun vào-ra nhận tín hiệu từ thiết bị ngoại vi, trong khi đó CPU làm việc khác
  - Khi đã có dữ liệu -> mô-đun vào-ra phát tín hiệu ngắt CPU
  - CPU yêu cầu dữ liệu
  - Mô-đun vào-ra chuyển dữ liệu đến CPU

* Nhìn từ CPU:
  - Phát tín hiệu điều khiển đọc
  - Làm việc khác

Cuối mỗi chu kỳ lệnh, kiểm tra tín hiệu ngắt. Nếu bị ngắt thì cất ngữ cảnh (nội dung các thanh ghi), thực hiện chương trình con ngắt để vào dữ liệu, khôi phục ngữ cảnh của chương trình đang thực hiện.

c) Các phương pháp nối ghép ngắt:

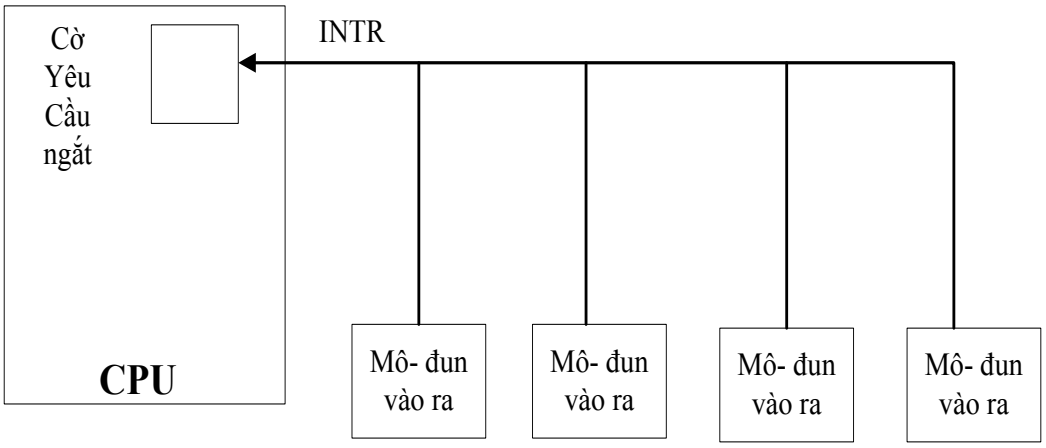
- Sử dụng nhiều đường yêu cầu ngắt
  + Mỗi mô-đun vào-ra được nối với đường yêu cầu ngắt
  + CPU phải có nhiều đường tín hiệu yêu cầu ngắt
  + Hạn chế số lượng mô-đun vào-ra
  + Các đường ngắt được quy định mức ưu tiên



Hình 6-7. Phương pháp nối ghép ngắt sử dụng nhiều đường yêu cầu ngắt

- Hỏi vòng bằng phần mềm:

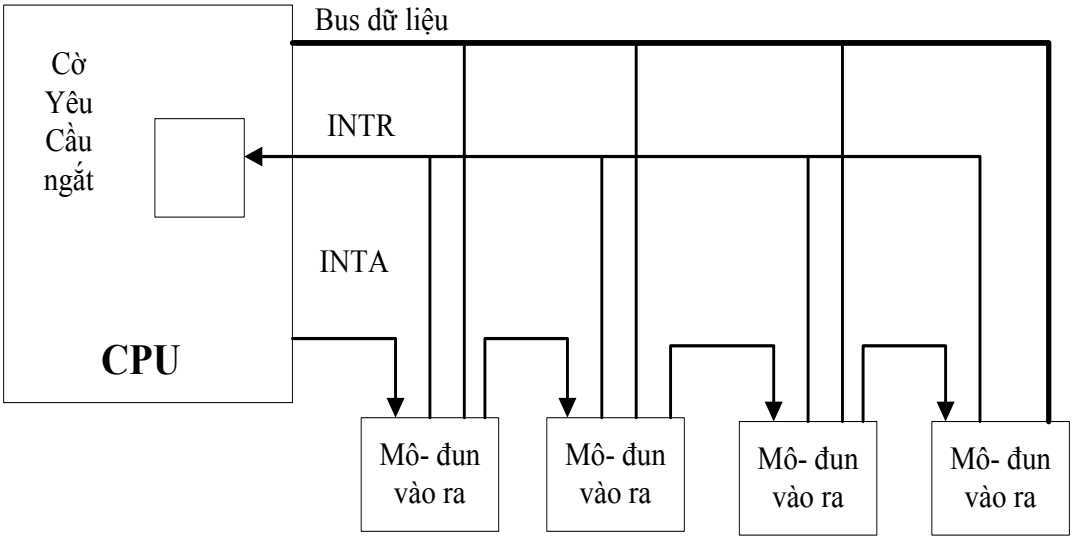
+ CPU sử dụng phần mềm thực hiện hỏi lần lượt từng mô-đun vào-ra
+ Hỏi vòng bằng phần mềm chậm
+ Thứ tự các mô-đun được hỏi vòng chính là thứ tự ưu tiên



Hình 6-8. Phương pháp nối ghép ngắt hỏi vòng bằng phần mềm

- Hỏi vòng bằng phần cứng:

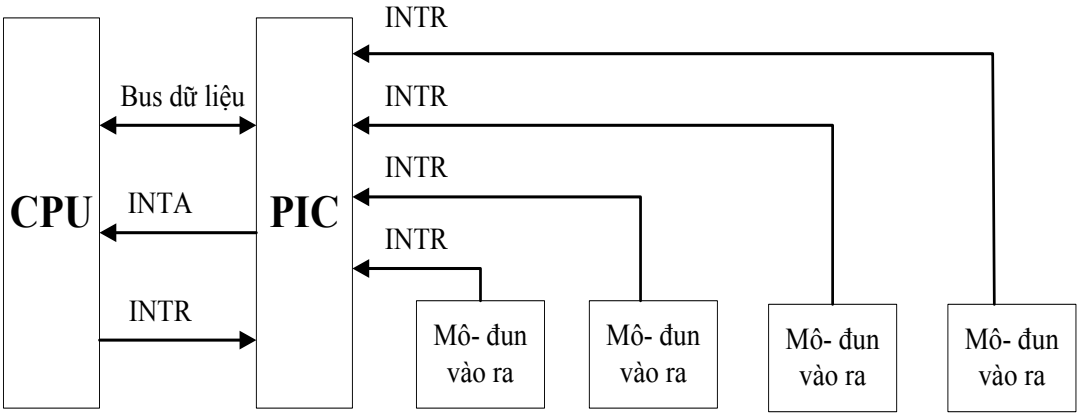
+ CPU phát tín hiệu chấp nhận ngắt (INTA) đến mô-đun vào-ra đầu tiên
+ Nếu mô-đun vào ra đó không gây ra ngắt thì nó gửi tín hiệu đến mô-đun kế tiếp cho đến khi xác định được mô-đun gây ngắt
+ Thứ tự các mô-đun vào-ra trong chuỗi xác định theo thứ tự ưu tiên



Hình 6-9. Phương pháp nối ghép ngắt hỏi vòng bằng phần cứng

- Sử dụng bộ điều khiển ngắt (PIC)

+ PIC: Programmable Interrupt Controller
+ PIC có nhiều đường vào yêu cầu ngắt có quy định mức ưu tiên
+ PIC chọn một yêu cầu ngắt không bị cấm có mức ưu tiên cao nhất gửi đến CPU



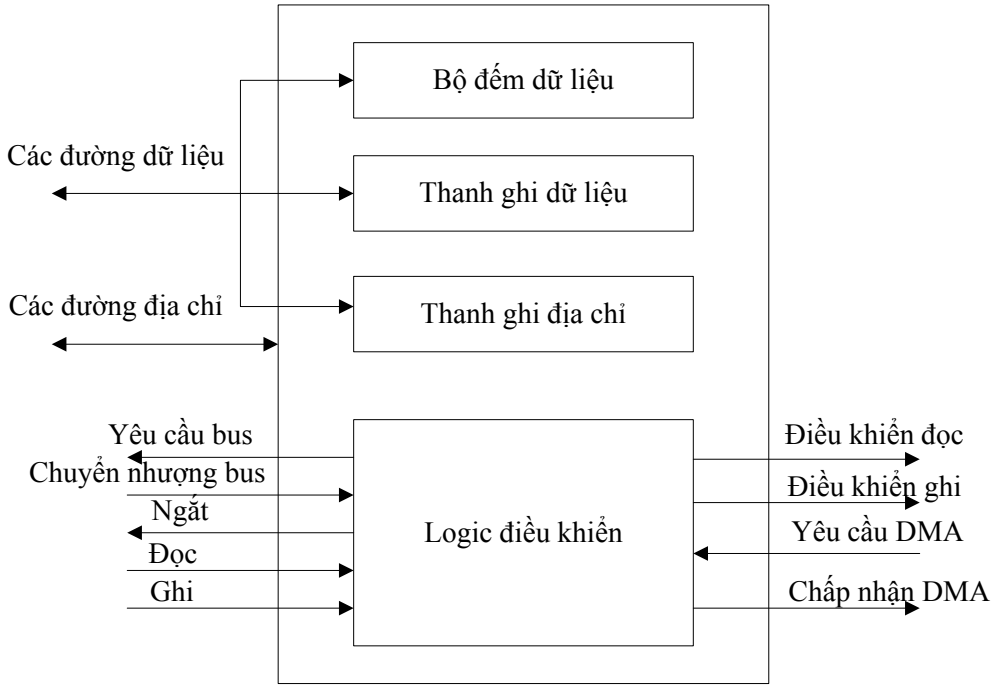
Hình 6-10. Phương pháp nối ghép ngắt sử dụng bộ điều khiển ngắt PIC

- Đặc điểm của vào ra điều khiển bằng ngắt:

+ Có sự kết hợp giữa phần cứng và phần mềm. Phần cứng gây ngắt CPU, phần mềm trao đổi dữ liệu
+ CPU trực tiếp điều khiển vào-ra
+ CPU không phải đợi mô-đun vào-ra nên hiệu quả sử dụng CPU tốt hơn

6.2.3. Truy nhập bộ nhớ trực tiếp – DMA (Direct memory access)

Vào ra bằng chương trình và ngắt CPU trực tiếp điều khiển nên chiếm thời gian của CPU, tốc độ truyền bị hạn chế vì phải chuyển qua CPU. Để khắc phục dùng DMA, thêm mô-đun phần cứng trên bus (DMAC). DMAC điều khiển trao đổi dữ liệu giữa mô-đun vào-ra với bộ nhớ chính [6].



Hình 6-11. Sơ đồ cấu trúc của DMAC

- *Các thành phần của DMAC:*
  - + Thanh ghi dữ liệu: Chứa dữ liệu trao đổi
  - + Thanh ghi địa chỉ: Chứa địa chỉ ngăn nhớ dữ liệu
  - + Bộ đếm dữ liệu: Chứa số từ dữ liệu cần trao đổi
  - + Logic điều khiển: Điều khiển hoạt động của DMAC
- *Hoạt động của DMA:*

+ CPU "nói" cho DMAC: Vào hay ra dữ liệu
  - + Địa chỉ thiết bị vào-ra (Cổng Vào-Ra tương ứng)
  - + Địa chỉ đầu của mảng nhớ chứa dữ liệu, sau đó nạp vào thanh ghi địa chỉ
  - + Số từ dữ liệu cần truyền, sau đó nạp vào bộ đếm dữ liệu

+ CPU làm việc khác
+ DMAC điều khiển trao đổi dữ liệu
+ Sau khi truyền được một từ dữ liệu thì:
  - + Nội dung thanh ghi địa chỉ tăng
  - + Nội dung bộ đếm dữ liệu giảm

+ Khi bộ đếm dữ liệu = 0, DMAC gửi tín hiệu ngắt CPU để báo kết thúc DMA

- *Các kiểu thực hiện DMA*
  - + DMA truyền theo khối: DMAC sử dụng bus để truyền xong cả khối dữ liệu
  - + DMA lấy chu kỳ: DMAC cưỡng bức CPU treo tạm thời từng chu kỳ bus, DMAC chiếm bus thực hiện truyền một từ dữ liệu
  - + DMA trong suốt: DMAC nhận biết những chu kỳ nào CPU không sử dụng bus thì chiếm bus để trao đổi một từ dữ liệu
- Đặc điểm của DMA: CPU không tham gia trong quá trình trao đổi dữ liệu.

+ DMAC điều khiển trao đổi dữ liệu giữa bộ nhớ chính và mô-đun vào-ra (hoàn toàn bằng phần cứng) nên tốc độ nhanh, phù hợp với những yêu cầu trao đổi mảng dữ liệu lớn

6.2.4. Kênh vào-ra hay bộ xử lý vào-ra

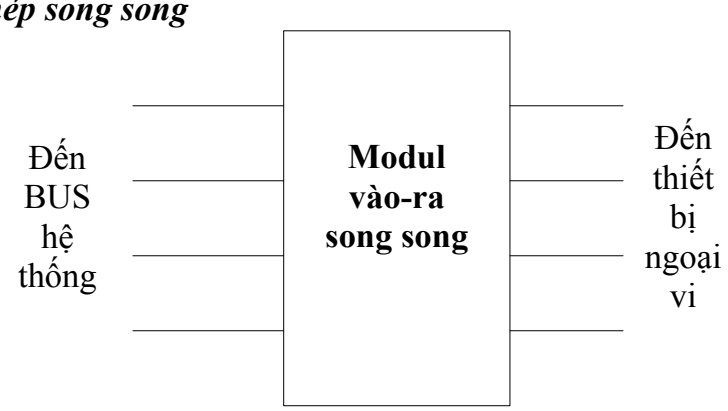
- Việc điều khiển vào-ra được thực hiện bởi một bộ xử lý vào-ra chuyên dụng
- Bộ xử lý vào-ra hoạt động theo chương trình riêng của nó
- Chương trình của bộ xử lý vào-ra có thể nằm trong bộ nhớ chính hoặc nằm trong bộ nhớ riêng
- Hoạt động theo kiến trúc đa xử lý

6.3. GHÉP NỐI VỚI THIẾT BỊ NGOẠI VI

6.3.1. Các kiểu nối ghép vào ra

Có hai kiểu nối ghép vào ra đó là: Nối ghép song song và nối ghép nối tiếp [7].

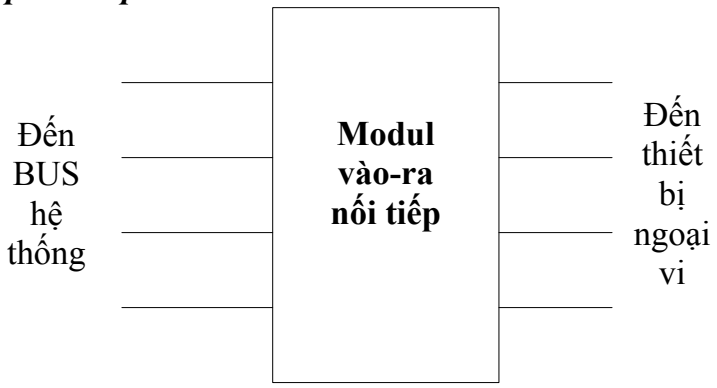
6.3.1.1. Nối ghép song song



Hình 6-12. Nối ghép vào- ra song song

- Truyền nhiều bit song song.
- Tốc độ nhanh.
- Cần nhiều đường truyền dữ liệu.

6.3.1.2. Nối ghép nối tiếp



Hình 6-13. Nối ghép vào- ra nối tiếp

- Cần truyền lần lượt từng bit.
- Cần có bộ truyền đổi dữ liệu từ song song sang nối tiếp và ngược lại.
- Tốc độ chậm hơn.
- Cần ít đường truyền dữ liệu.

6.3.2. Các cấu hình ghép nối

- Điểm tới điểm (Point to point). Thông qua một cổng vào – ra nối ghép với một thiết bị ngoại vi.
- Điểm tới đa điểm (Point to Multipoint). Thông qua một cổng vào – ra cho phép nối ghép được với nhiều thiết bị ngoại vi.

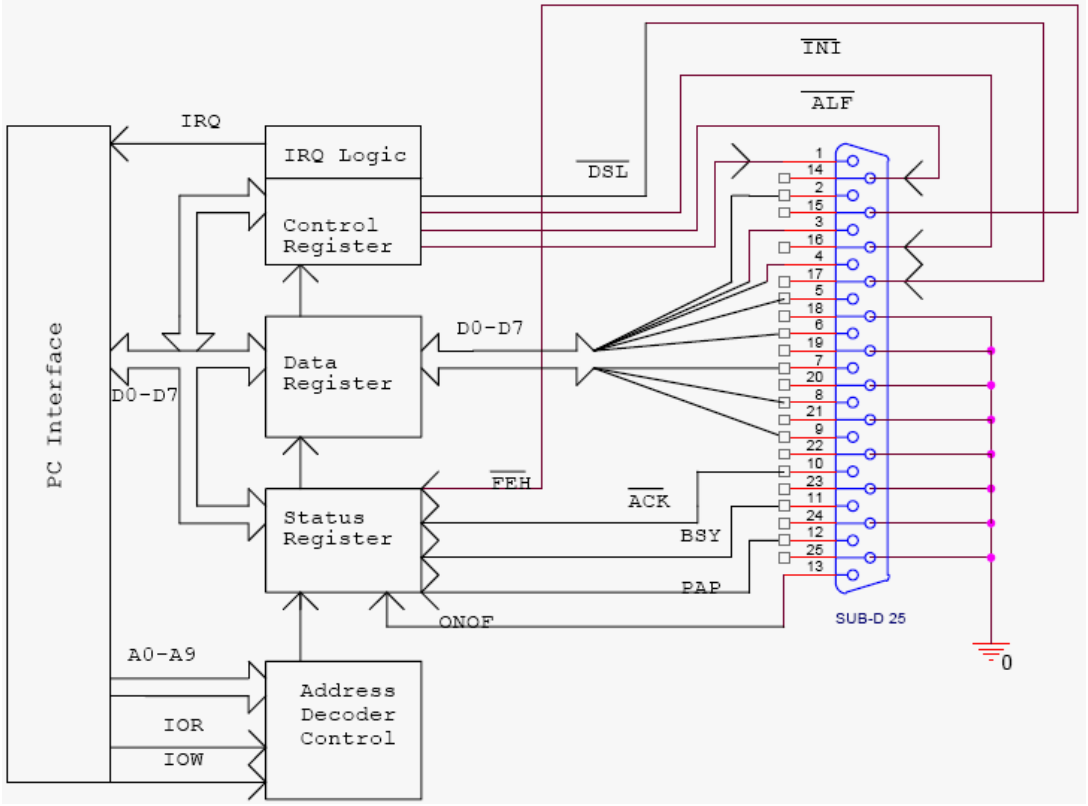
Ví dụ 6.3.1:

- SCSI (Small Computer System Interface): 7 hoặc 15 thiết bị.
- USB (Universal Serial Bus): 127 thiết bị.
- IEEE 1394 (Firewire): 63 thiết bị.

6.3.3. Các cổng vào ra thông dụng

6.3.3.1. Cổng song song LPT

Các máy tính PC được trang bị ít nhất là 1 cổng song song và 1 cổng nối tiếp. Khác với ghép nối nối tiếp có nhiều ứng dụng, ghép nối song song thường chỉ phục vụ cho máy in. Sơ đồ ghép nối song song như hình sau:



Hình 6-14. Ghép nối song song ra cổng LPT

Có ba thanh ghi có thể truyền số liệu và điều khiển máy in cũng như khối ghép nối. Địa chỉ cơ sở của các thanh ghi cho tất cả cổng LPT (line printer) từ LPT1 đến LPT4 được lưu trữ trong vùng số liệu BIOS. Thanh ghi số liệu được định vị ở offset 00h, thanh ghi trang thái ở 01h, và thanh ghi điều khiển ở 02h. Thông thường, địa chỉ cơ sở của LPT1 là 378h, LPT2 là 278h, do đó địa chỉ của thanh ghi trạng thái là 379h hoặc 279h và địa chỉ thanh ghi điều khiển là 37Ah hoặc 27Ah. Định dạng các thanh ghi như sau:

Bảng 6-1. Bảng định dạng cho các thanh ghi dữ liệu, trạng thái và điều khiển

(a) Thanh ghi dữ liệu (hai chiều)

	7	6	5	4	3	2	1	0
Tín hiệu máy in	D7	D6	D5	D4	D3	D2	D1	D0
Chân số	9	8	7	6	5	4	3	2

(b) Thanh ghi điều khiển máy in

	7	6	5	4	3	2	1	0
Tín hiệu máy in	BSY	/ACK	PAP	OFON	/FEH	x	x	x
Sốchân cắm	11	10	12	13	15	-	-	-

(c) Thanh ghi trạng thái máy in (chỉ đọc)

	7	6	5	4	3	2	1	0
Tín hiệu máy in	x	x	x	IRQ	/DSL	/INI	/ALF	STR
Sốchân cắm	-	-	-	-	17	16	14	1

x: không sử dụng

IRQ: yêu cầu ngắt cứng; 1 = cho phép; 0 = không cho phép

Bản mạch ghép nối chỉ có bus dữ liệu 8 bit do dữ liệu luôn đi qua máy in thành từng khối 8 bit. Các chân tín hiệu của đầu cắm 25 chân của cổng song song LPT như sau:

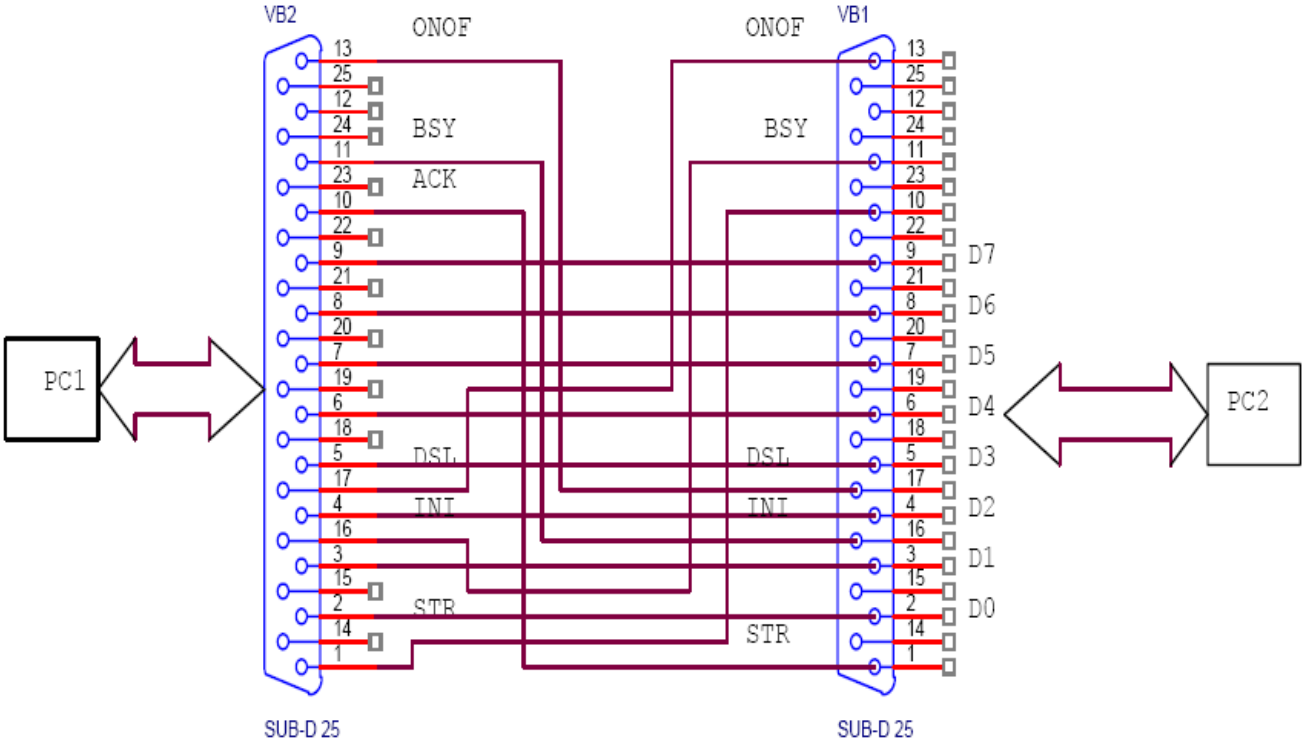
Bảng 6-2. Tín hiệu chân của cổng LPT

Chân	**Tín hiệu**	**Mô tả**
1	STR	Mức tín hiệu thấp, truyền dữliệu tới máy in
2	D0	Bit dữ liệu 0
3	D1	Bit dữ liệu 1
4	D2	Bit dữ liệu 2
5	D3	Bit dữ liệu 3
6	D4	Bit dữ liệu 4
7	D5	Bit dữ liệu 5
8	D6	Bit dữ liệu 6
9	D7	Bit dữ liệu 7
10	ACK	Mức thấp: máy in đã nhận 1 ký tự và có khả năng nhận nữa
11	BSY	Mức cao: ký tự đã được nhận; bộ đệm máy in đầy; khởi động máy in; máy in ở trạng thái off-line.
12	PAP	Mức cao: hết giấy
13	OFON	Mức cao: máy in ở trạng thái online
14	ALF	Tự động xuống dòng; mức thấp: máy in xuống dòng tự động
15	FEH	Mức thấp: hết giấy; máy in ở offline; lỗi máy in
16	INI	Mức thấp: khởi động máy in

17	DSL	Mức thấp: chọn máy in
18-25	GROUND	0V

Thường tốc độ xử lý dữ liệu của các thiết bị ngoại vi như máy in chậm hơn PC nhiều nên các đường ACK, BSY và STR được sử dụng cho kỹ thuật bắt tay. Khởi đầu, PC đặt dữ liệu lên bus sau đó kích hoạt đường STR xuống mức thấp để thông tin cho máy in biết rằng số liệu đã ổn định trên bus. Khi máy in xử lý xong dữ liệu, nó sẽ trả lại tín hiệu *ACK* xuống mức thấp để ghi nhận. PC đợi cho đến khi đường BSY từ máy in xuống thấp (máy in không bận) thì sẽ đưa tiếp dữ liệu lên bus.

Dữ liệu có thể trao đổi trực tiếp giữa 2 PC qua các cổng song song với nhau. Muốn vậy, các đường điều khiển bên này phải được kết nối với các đường trạng thái bên kia.



Hình 6-15. Trao đổi dữ liệu qua cổng song song giữa 2 PC

6.3.3.2. Nối tiếp (Serial)

Các ghép nối của PC cho trao đổi nối tiếp đều theo tiêu chuẩn RS-232 của EIA (Electronic Industries Association) hoặc của CCITT ở Châu Âu. Chuẩn này quy định ghép nối về cơ khí, điện, và logic giữa một thiết bị đầu cuối số liệu DTE (Data Terminal Equipment) và thiết bị thông tin số liệu DCE (Data Communication Equipment). Thí dụ, DTE là PC và DCE là MODEM. Có 25 đường với đầu cắm 25 chân D25 giữa DTE và DCE. Hầu hết việc truyền số liệu là bất đồng bộ. Có 11 tín hiệu trong chuẩn RS232C dùng cho PC, IBM còn quy định thêm đầu cắm 9 chân D9. Các chân tín hiệu và mối quan hệ giữa các đầu cắm 25 chân và 9 chân:

Bảng 6-3. Tín hiệu chân của cổng nối tiếp

D25	D9	Tín hiệu	Hướng truyền	Mô tả
1	-	-	-	Protected ground: nối đất bảo vệ
2	3	TxD	DTEDCE	Transmitted data: dữliệu phát
3	2	RxD	DCEDTE	Received data: dữliệu thu
4	7	RTS	DTEDCE	Request to send: DTE yêu cầu truyền dữliệu
5	8	CTS	DCEDTE	Clear to send: DCE sẵn sàng nhận dữliệu
6	6	DSR	DCEDTE	Data set ready: DCE sẵn sàng làm việc
7	5	GND	-	Ground: nối đất (0V)
8	1	DCD	DCEDTE	Data carier detect: DCE phát hiện sóng mang
20	4	DTR	DTEDCE	Data terminal ready: DTE sẵn sàng làm việc
22	9	RI	DCEDTE	Ring indicator: báo chuông
23	-	DSRD	DCEDTE	Data signal rate detector: dò tốc độtruyền

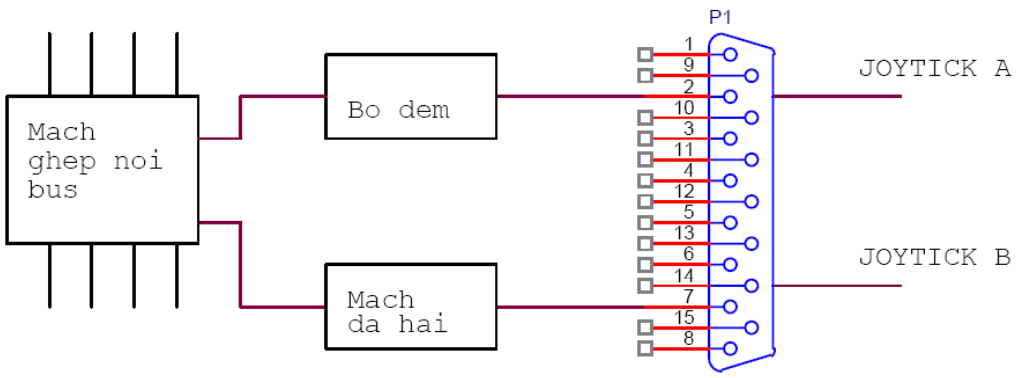
Chuẩn RS-232C cho phép truyền tín hiệu với tốc độ đến 20.000 bps nhưng nếu cáp truyền đủ ngắn có thể lên đến 115.200 bps. Chiều dài cáp cực đại là 17-20m.

Các phương thức nối giữa DTE và DCE:

- Đơn công (simplex connection): dữ liệu chỉ được truyền theo 1 hướng.
- Bán song công (half-duplex): dữ liệu truyền theo 2 hướng, nhưng mỗi thời điểm chỉ được truyền theo 1 hướng.
- Song công (full-duplex): số liệu được truyền đồng thời theo 2 hướng.

6.3.3.3. Cổng PC-Game

Cấu trúc và chức năng của board ghép nối trò chơi (PC game) như hình bên dưới.



Hình 6-16. Cấu trúc của board ghép nối cổng PC-game

Bảng 6-4. Tín hiệu chân của cổng PC-game

Chân của đầu nối 15 chân	Sửdụng cho
2	Phím 1 của Joystick A (BA1)
3	Biến trở X của Joystick A
6	Biến trở Y của Joystick A
7	Phím 2 của Joystick A (BA2)
10	Phím 1 của Joystick A (BB1)
11	Biến trở X của Joystick B
13	Biến trở Y của Joystick B
14	Phím 2 của Joystick A (BB2)
1, 8, 9, 15	Vcc (+5V)
4, 5, 12	GND (0V)

Board mạch được nối với bus hệ thống của PC chỉ qua 8 bits thấp của bus dữ liệu, 10 bits thấp của bus địa chỉ và các đường điều khiển IOR và IOW. Một đầu nối 15 chân được nối với board mạch cho phép nối cực đại hai thiết bị cho PC game gọi là joystick.

Mỗi joystick có 2 biến trở có giá trị biến đổi từ 0 đến 100kΩ được đặt vuông góc với nhau đại diện cho vị trí x và y của joystick. Thêm nữa chúng có 2 phím bấm, thường là các công tắc thường hở phù hợp với các mức logic cao của các dây trên mạch.

Có thể xác định được trạng thái nhấn hoặc nhả phím một cách dễ dàng bằng lệnh IN tới địa chỉ 201h. Nibble cao chỉ thị trạng thái của phím. Vì board không dùng đường IRQ do đó không có khả năng phát ra 1 ngắt, do vậy board chỉ hoạt động trong chế độ hỏi vòng (polling). Byte trạng thái của board game như sau:

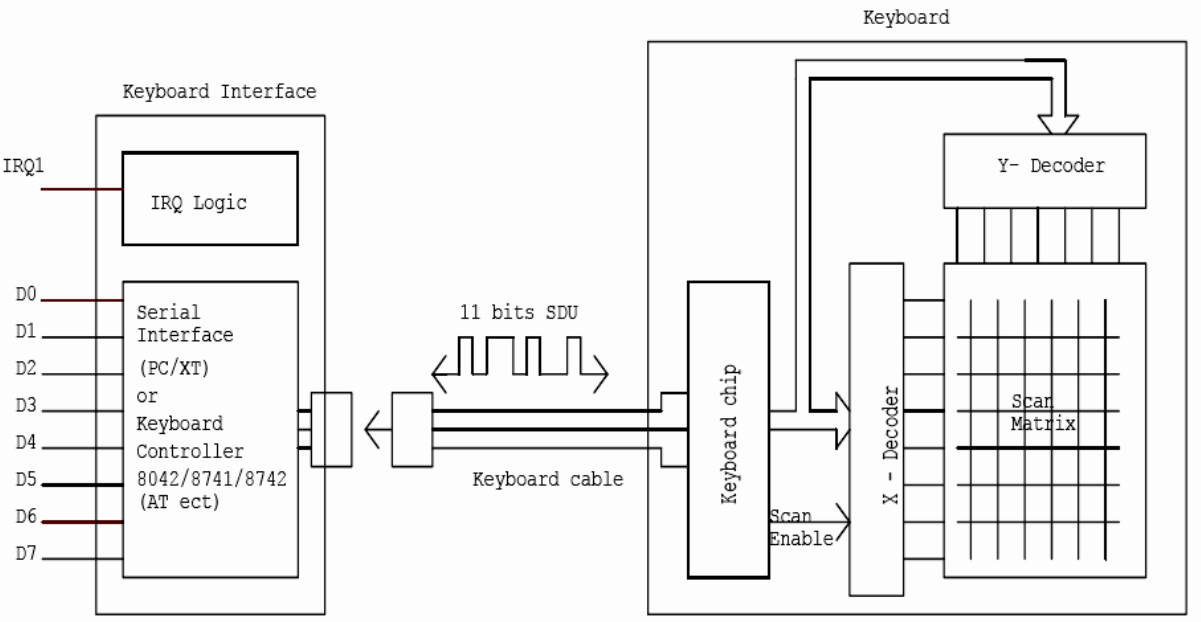
Bảng 6-5. Byte trạng thái của board game

D7	D6	D5	D4	D3	D2	D1	D0
BB2	BB1	BA2	BA1	BY	BX	AY	AX

BB2, BB1, BA2, BA1: Trạng thái của các phím B2, B1, A2, A1; 1 = nhả; 0 = nhấn

BY, BX, AY, AX: Trạng thái của mạch đa hài tuỳ thuộc vào biến trở tương ứng.

6.3.3.4. Cổng bàn phím



Hình 6-17. Sơ đồ kết nối cổng bàn phím

Chip xử lý bàn phím liên tục kiểm tra trạng thái của ma trận quét (scan matrix) để xác định công tắc tại các tọa độ X,Y đang được đóng hay mở và ghi một mã tương ứng vào bộ đệm bên trong bàn phím. Sau đó mã này sẽ được truyền nối tiếp tới mạch ghép nối bàn phím trong PC. Cấu trúc của SDU cho việc truyền số liệu này và các chân cắm của đầu nối bàn phím.

SDU

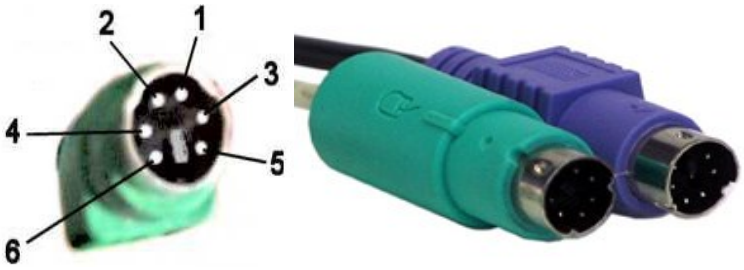
0										10
STRT	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	PAR	STOP

STRT: bit start (luôn bằng 0)
DB0 - DB7: bit số liệu từ 0 đến 7.
PAR: bit parity (luôn lẻ)
STOP: bit stop (luôn bằng 1).

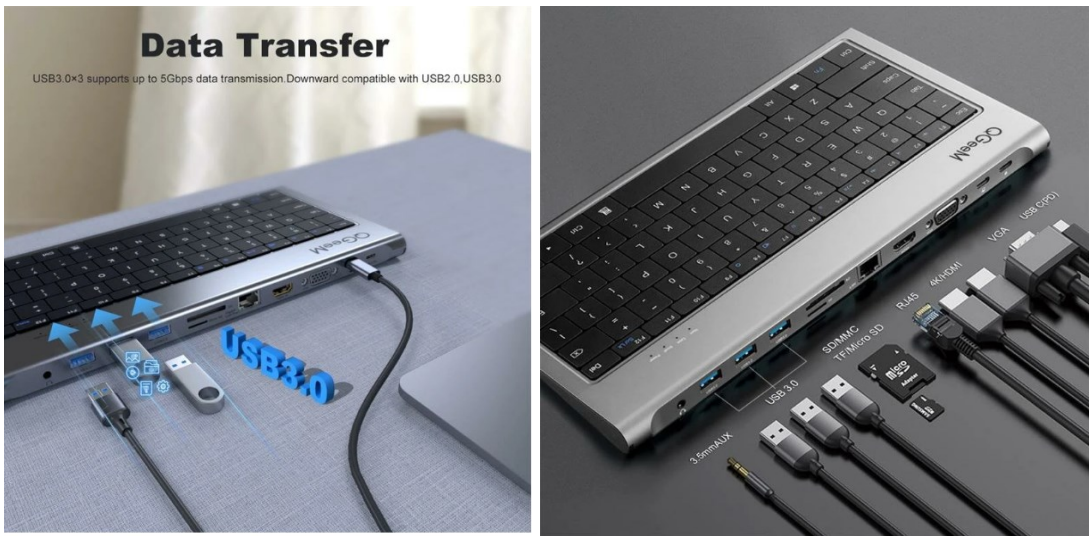
Tín hiệu xung nhịp dùng cho việc trao đổi dữ liệu thông tin nối tiếp đồng bộ với mạch ghép nối bàn phím (keyboard interface) trên main board được truyền qua chân số 1.

Một bộ điều khiển bàn phím đã được lắp đặt trên cơ sở các chíp 8042, hoặc 8742,8741. Nó có thể được chương trình hóa (thí dụ khóa bàn phím) hơn nữa số liệu có thể truyền theo 2 hướng từ bàn phím và mạch ghép nối, do vậy vi mã của chíp bàn phím có thể giúp cho việc nhận lệnh điều khiển từ PC, thí dụ như đặt tốc độ lặp lại của nhấn bàn phím,….

Chân 1: dữ liệu
Chân 2: không dùng
Chân 3: GND
Chân 4: Vcc
Chân 5: clock
Chân 6: không dùng



Hình 6-18. Đầu cắm bàn phím PS/2



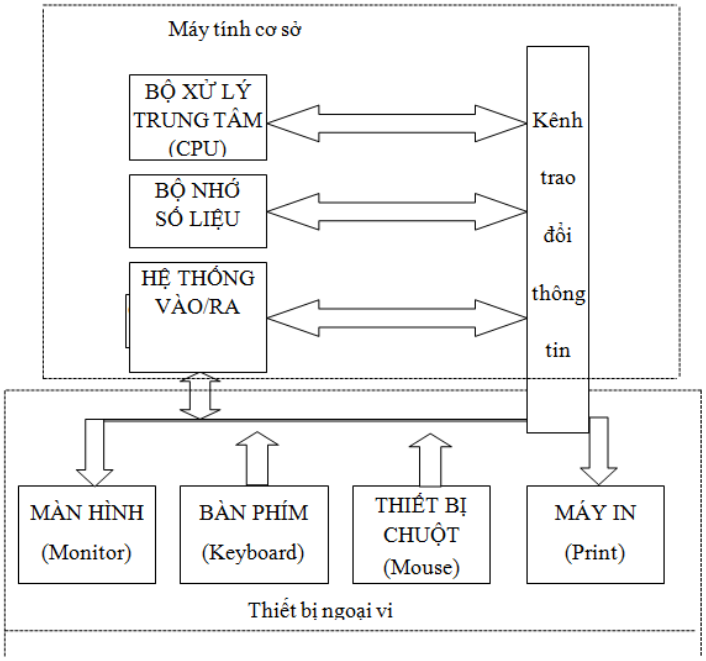
Hình 6-19. Đầu cắm cổng bàn phím bằng USB

6.4. GIAO DIỆN TRUYỀN DỮ LIỆU

6.4.1. Giao diện song song

Hệ thống ghép nối vào/ra đáp ứng việc truyền dữ liệu giữa hệ máy tính cơ sở với thiết bị ngoại vi được nối ghép với chúng.

Cấu trúc ghép nối của hệ thống được chỉ ra trên hình 7.1.



Hình 6-18. Cấu trúc ghép nối máy tính cơ sở với thiết bị ngoại vi

Hệ thống vào /ra trao đổi số liệu với các thiết bị ngoại vi thông qua mạch tương thích gọi là cổng (PORT). Thiết bị ngoại vi nối trực tiếp với cổng, cổng được nối trực tiếp với mạch kiểm soát kênh.

Khi thiết bị ngoại vi nối với cổng thì nó yêu cầu phải có một phương pháp thâm nhập vào bộ xử lý trung tâm, phương pháp này gọi là thao tác ngắt (INTERRUPT). Thao tác ngắt sẽ tạm thời dừng chương trình mà bộ xử lý trung tâm đang thực hiện, CPU sẽ đáp ứng cho thao tác ngắt bằng một chương trình con phục vụ ngắt (Interrupt Service Routine). Có hai loại thao tác ngắt cơ bản là: Ngắt có che mặt nạ (Maskable) và ngắt không che mặt nạ (Nonmaskable).

Thao tác ngắt không che mặt nạ đòi hỏi CPU phải đáp ứng ngay và được dùng trong những trường hợp khẩn cấp. Ví dụ như mất điện nguồn cung cấp.

Ngắt che mặt nạ là thao tác ngắt mà CPU có thể bỏ qua, tuỳ thuộc vào trạng thái của thanh ghi cờ trạng thái.

Ngắt có thể là có các mức ưu tiên khác nhau, tuỳ thuộc vào từng yêu cầu và độ cấp bách. Ví dụ như ngắt do yêu cầu phục vụ nhận dữ liệu vào của thiết bị thu thông tin ưu tiên cao hơn ngắt khi có yêu cầu đưa thông tin ra từ bộ vi xử lý.

Các lệnh ngắt có thể phát ra từ cả hai phần cứng và phần mềm.

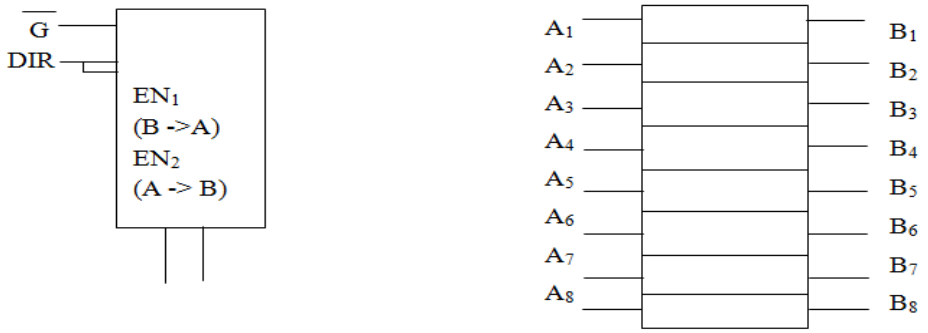
6.4.1.1. Mạch thu/phát đệm dữ liệu SN74LS245

Mạch thu/phát đệm kênh dữ liệu SN74LS245 là vi mạch thu/phát kênh hai chiều 8 bít. Vi mạch ba trạng thái này được dùng để đệm và điều khiển chiều chuyển động của số liệu trên kênh dữ liệu. Sơ đồ cấu trúc và bảng chân lý của vi mạch SN74LS245 được chỉ ra trên hình 7.2 [3].

Khi chân Enable $\overline{G}$ có mức lôgic cao thì vi mạch ở trạng thái trở kháng cao. Khi $\overline{G}$ xuống mức lôgic thấp thì số liệu sẽ di chuyển theo chiều do chân DIR (Direction) quyết định.

Nếu chân DIR ở mức lôgic thấp, thì số liệu đi từ lối vào B đến lối ra A. Ngược lại, nếu chân DIR ở mức lôgic cao, thì số liệu đi từ lối vào A đến lối ra B.

Trong hệ vi xử lý, vi mạch SN74LS245 thường dùng để kiểm soát chiều di chuyển của số liệu trên kênh dữ liệu. Khi chân DIR được nối với đường dây tín hiệu DT/$\overline{R}$ thì nó dùng để quy định dữ liệu sẽ được CPU thực hiện phát đi hay thu về. Lối vào $\overline{G}$ nối với đường tín hiệu DEN để kiểm soát việc nối ghép thông tin lối ra với kênh dữ liệu.



a. Sơ đồ cấu trúc vi mạch SN74LS245

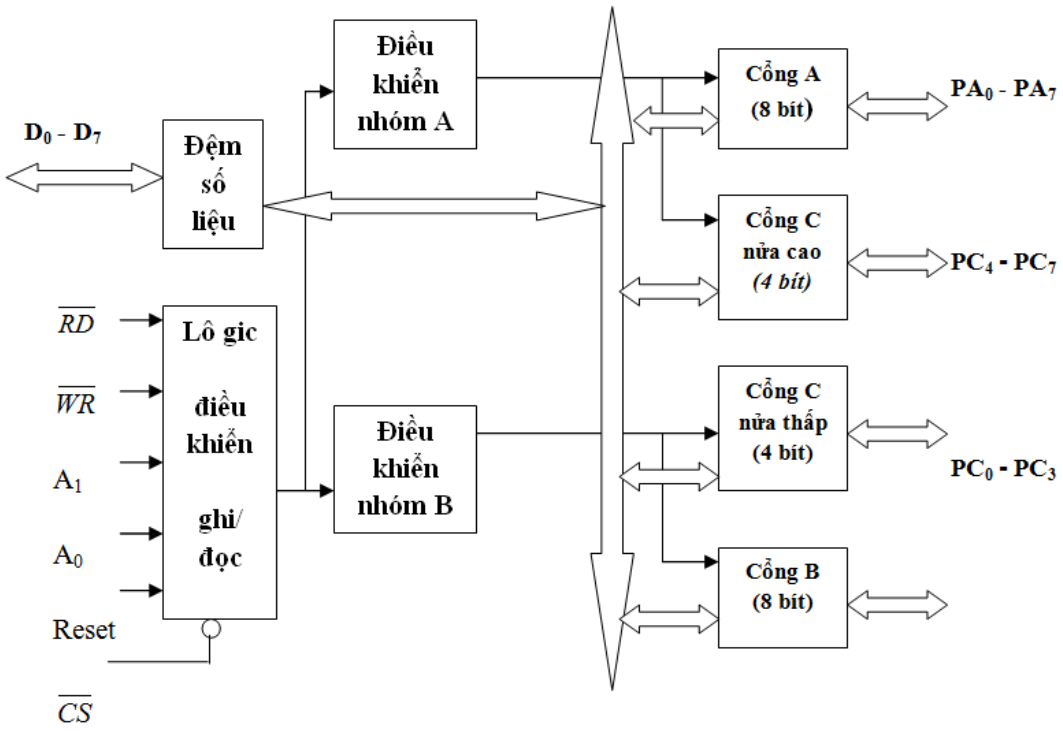
$\overline{G}$	DIR	OUT
L	L	B -> A
L	H	A -> B
H	X	Trở kháng cao

b. Bảng chân lý của vi mạch

Hình 6-19. Sơ đồ cấu trúc và bảng chân lý của vi mạch SN74LS245

6.4.1.2. Mạch tương thích với ngoại vi khả trình 8255A

Mạch tương thích với ngoại vi khả trình 8255A là vi mạch tương thích với các ngoại vi của hệ vi xử lý (PPI- Programmable Peripheral Interface) dùng để nối các hệ ngoại vi với máy vi tính. Vi mạch 8255A thực hiện chức năng tương hợp song song rất linh hoạt và được điều khiển bằng phần mềm.



Hình 6-20. Sơ đồ khối của mạch tương thích với ngoại vi khả trình 8255A

Về hướng CPU, vi mạch 8255A có các mạch đệm kênh dữ liệu hai chiều ($D_0 \div D_7$), các tín hiệu kiểm soát ghi / đọc ($\overline{RD}$, $\overline{WR}$, A_0, A_1, RESET, $\overline{CS}$).

Về phía đầu ra của vi mạch 8255A có các cổng:

- Cổng A có các tín hiệu I/0: $PA_0 \div PA_7$.
- Cổng B có các tín hiệu I/0: $PB_0 \div PB_7$.
- Cổng C có các tín hiệu I/0: $PC_0 \div PC_7$.

Các cổng A, B là các cổng 8 bít 2 chiều .

Cổng C được chia thành 2 cổng, mỗi cổng 4 bít. Các bít cao từ $PC_7 \div PC_4$ các bít thấp từ $PC_3 \div PC_0$. Các đường tín hiệu này dùng để di chuyển số liệu, các lệnh và thông tin trạng thái giữa CPU, vi mạch 8255A với các thiết bị ngoại vi.

Sự ấn định thời gian cho vi mạch 8255A trong việc di chuyển dữ liệu được điều khiển bằng tín hiệu kiểm soát đọc ($\overline{RD}$) và ghi ($\overline{WR}$). Các tín hiệu này cho phép CPU đọc thông tin ra từ vi mạch 8255A hay ghi thông tin vào vi mạch 8255A. Khi tín hiệu đọc có mức hiệu lực thấp thì CPU đọc dữ liệu hoặc thông tin trạng thái từ vi mạch 8255A ra qua kênh dữ liệu.

Khi tín hiệu ghi có mức hiệu lực thấp thì CPU ghi dữ liệu hoặc lời điều khiển vào vi mạch 8255A cũng thông qua kênh dữ liệu.

Việc lựa chọn các cổng được thực hiện bằng các đường tín hiệu A0 và A1 như chỉ ra trong bảng 6-6.

Bảng 6-6. Bảng xác định việc lựa chọn các cổng của vi mạch 8255A

A_1	A_0	Cổng
0	0	A
0	1	B
1	0	C
1	1	Kiểm soát

Vi mạch 8255A có 3 chế độ hoạt động cơ bản là 0, 1 và 2.

- Chế độ 0: Chế độ vào/ra cơ sở.
- Chế độ 1: Chế độ vào/ra chốt (Strobe).
- Chế độ 2: Chế độ hoạt động 2 chiều.

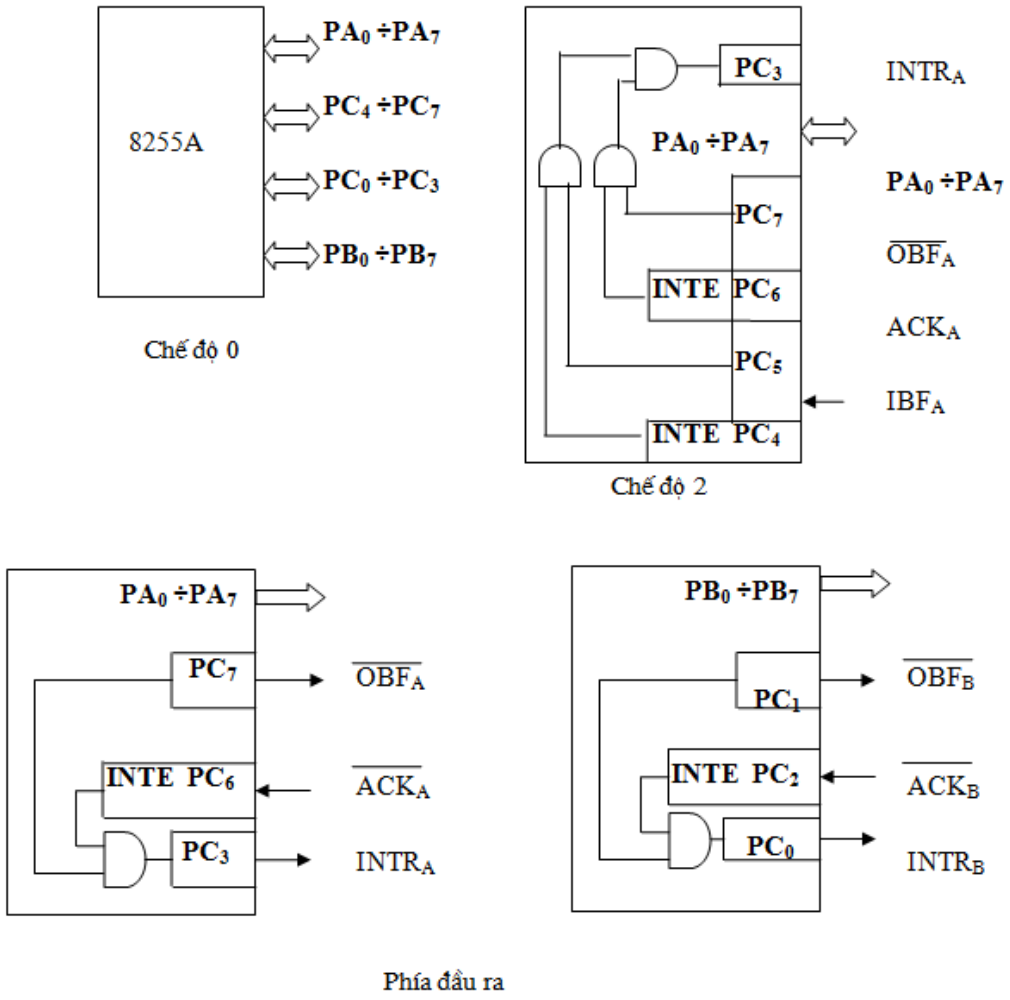
Trong chế độ 0: Vi mạch 8255A làm chức năng chốt, tức là dữ liệu được CPU đưa ra sẽ được giữ lại trong vi mạch 8255A cho đến khi nó chuyển trạng thái.

Chế độ 1: Dữ liệu chỉ được giữ lại trong vi mạch 8255A một khoảng thời gian ngắn, sau đó phải được chốt vào một mạch chốt ngoài nếu cần phải lưu lại số liệu.

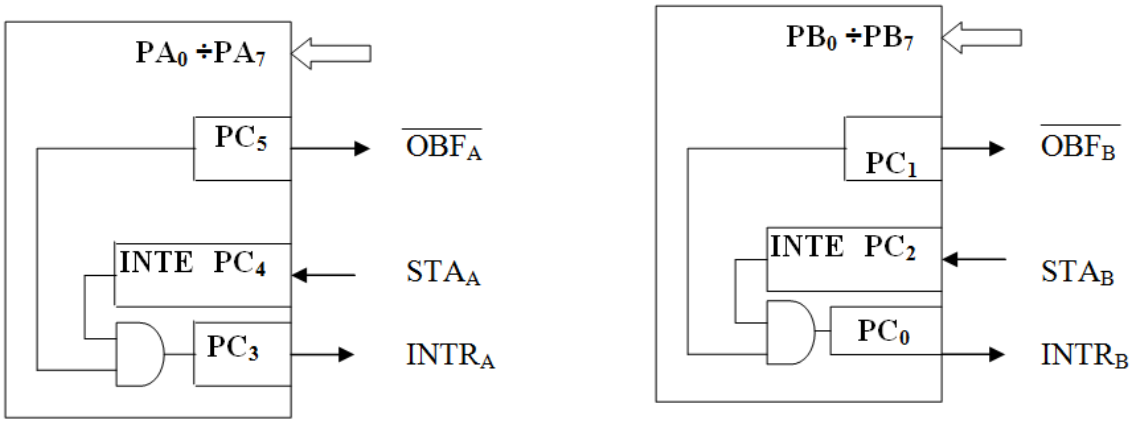
Chế độ 2: Là chế độ dùng cho hoạt động 2 chiều để trao đổi thông tin của hệ vi xử lý .

Các tín hiệu đối thoại và trạng thái của PPI 8255A ở các chế độ 1, 2, 3 được chỉ ra trên hình 6.17 và 6.18.

Chú ý rằng: Cổng B và cổng C các bít thấp từ $PC_3 \div PC_0$ chỉ có thể hoạt động trong chế độ 0 hoặc chế độ 1, bởi vì chỉ có 1 bít dữ liệu để xác định chế độ hoạt động. Bít điều khiển cuối cùng D_7 là cờ thiết lập chế độ. Nó phải được đặt lên mức lôgic 1 khi chế độ hoạt động thay đổi.



Hình 6-21. Các mạch logic bên trong và các tín hiệu ở các chế độ 0 và 2 của PPI 8255A

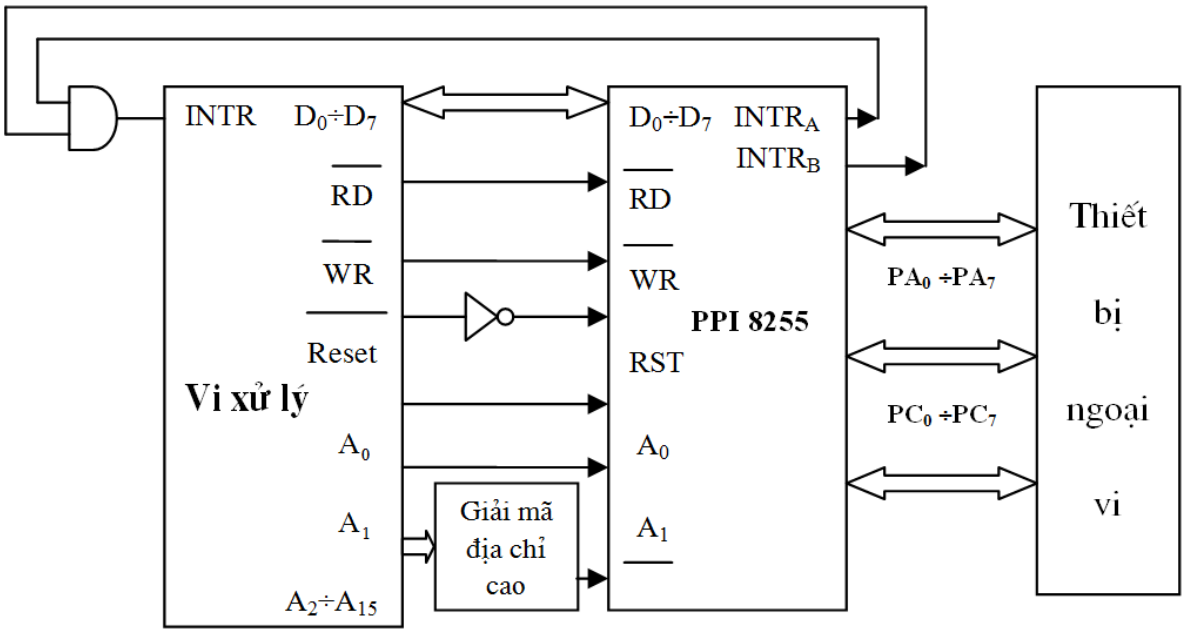


Hình 6-22. Các mạch logic bên trong và các tín hiệu ở các chế độ 1 của PPI 8255A

Sơ đồ ghép nối cổng vào/ra theo chương trình với bộ vi xử lý và thiết bị ngoại vi được chỉ ra trên hình 7.5.

PPI 8255A được đặt giữa bộ vi xử lý và thiết bị ngoại vi, nó đóng vai trò trung chuyển thông tin giữa bộ vi xử lý với thiết bị ngoại vi qua các kênh thông tin của máy vi tính với thiết bị ngoại vi.

Thông thường khi ghép với ngoại vi thì các cổng A và B dùng để ghép nối trao đổi dữ liệu, còn cổng C dùng cho các thông tin đối thoại về trạng thái và chốt số liệu cho thiết bị.



Hình 6-23. Ghép nối giữa PPI 8255A với máy vi tính và thiết bị ngoại vi

6.4.2. Giao diện tuần tự

Mạch kiểm soát ngắt khả trình 8259A (PIC - Programmable Interrupt Controller)

Trong hầu hết các hệ thống máy vi tính đều có dùng một phương pháp cho phép mạch vào/ra tăng cường sự chú ý của CPU. Khi CPU nhận được một yêu cầu ngắt (INTR hoặc NMI) thì nó tiến hành tuần tự một loạt các bước để đáp ứng yêu cầu đó. Đầu tiên nó hoàn thành lệnh đang thực hiện, rồi quyết định có báo nhận cho yêu cầu ngắt hay không. Trường hợp yêu cầu ngắt không có mặt nạ che (NMI) thì nó bắt buộc phải báo nhận và phục vụ cho yêu cầu đó. Trong trường hợp yêu cầu ngắt bình thường (INTR)

thì trước hết CPU tiến hành kiểm tra thanh ghi cờ để biết lệnh ngắt có bị che mặt nạ hay không. Sau đó nó sẽ phục vụ ngắt nếu cần thiết, nếu chưa cần thiết thì nó có thể bỏ qua thực hiện tiếp chương trình nó đương làm, sau khi hoàn thành chương trình mới chuyển sang phục vụ ngắt [5].

Khi CPU phục vụ ngắt, đầu tiên nó gửi cất nội dung của con trỏ lệnh và các thanh ghi đang được sử dụng có hiệu lực vào ngăn xếp (STACK). Tiếp theo là nó phục vụ ngắt bằng cách tìm địa chỉ cư trú của chương trình con phục vụ ngắt tương ứng được lưu trữ trong ROM. Chương trình con phục vụ ngắt là chương trình quy định các bước phải theo để phục vụ cho từng loại ngắt riêng biệt.

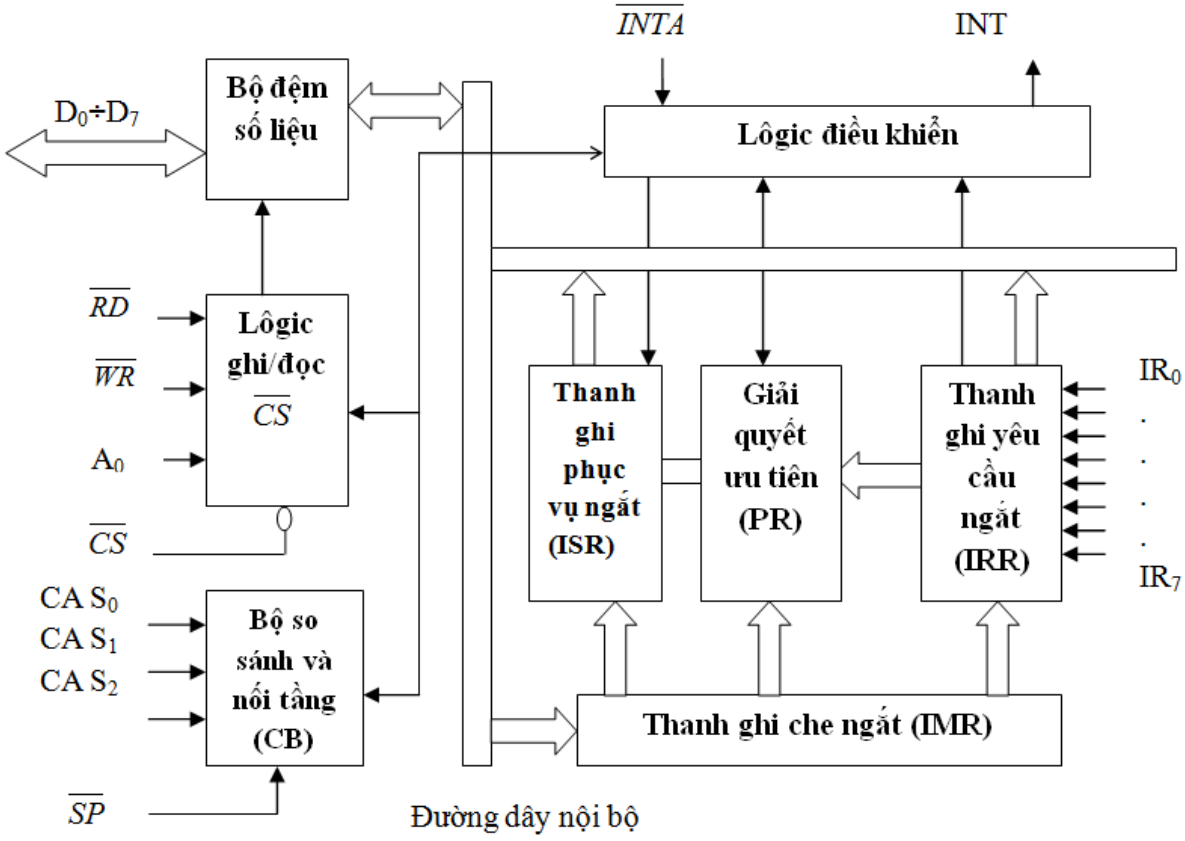
Cấu trúc của PIC 8259A được chỉ ra trên hình 6.19.

Cấu trúc bên trong của mạch kiểm soát ngắt khả trình 8259A gồm 8 khối chức năng:

- Mạch đệm kênh dữ liệu để tương thích kênh dữ liệu với kênh bên trong của PIC 8259A. Mạch đệm 3 trạng thái 8 bít này là mạch trao đổi thông tin 2 chiều, nó được mở bằng khối lôgic ghi/đọc.
- Khối lôgic ghi/đọc sẽ cung cấp chiều, thời gian, nguồn hoặc nơi đến cho dữ liệu di chuyển qua khối mạch đệm của kênh dữ liệu.

Các tín hiệu điều khiển lối vào $\overline{RD}$, $\overline{WR}$, A_0 và $\overline{CS}$ sẽ kiểm soát khối lôgic ghi/đọc.

- Thanh ghi yêu cầu ngắt (IRR) lưu trữ trạng thái của những tín hiệu vào yêu cầu ngắt.
- Thanh ghi phục vụ ngắt (ISR) dùng để lưu trữ mức ngắt sẽ được phục vụ ngay.



Hình 6-24. Sơ đồ khối của PIC 8259A

- Mạch giải quyết ưu tiên (PR) dùng để quyết định tín hiệu ngắt nào có mức ưu tiên cao nhất.

- Khối so sánh và nối tầng (Cascade Buffer) dùng để tương thích giữa PIC 8259A chính và phụ trong các thao tác nối tiếp.

- Thanh ghi mặt nạ ngắt (IMR) dùng để che hoặc không che mặt nạ đối với các tín hiệu yêu cầu ngắt khác nhau.

Khối lôgic kiểm soát (điều khiển) dùng để sử dụng các thông tin do IRR, ISR và PR đưa vào để kiểm soát tín hiệu ngắt INT ở lối ra. Tín hiệu ngắt INT này sẽ yêu cầu ngắt đối với CPU. Khối lôgic kiểm soát này đồng thời cũng quản lý tín hiệu $\overline{INTA}$ từ CPU đưa sang.

Các tín hiệu của PIC 8259A thực hiện các chức năng sau:

1. Vcc (Chân 28) - Nguồn cung cấp điện áp $+5^{v}$.

2. GND (Chân14) - Đất của nguồn nuôi.

3. $\overline{CS}$ (Chân1) - Tín hiệu chọn chíp, là tín hiệu vào hiệu lực thấp để mở PIC 8259A.

4. $\overline{WR}$ (Chân 2) - Tín hiệu ghi là tín hiệu vào hiệu lực thấp. Tín hiệu này liên kết với $\overline{CS}$ sẽ mở PIC 8259A để nhận các lời lệnh từ CPU đưa sang.

5. $\overline{RD}$ (Chân 3) - Tín hiệu đọc, là tín hiệu vào hiệu lực thấp tín hiệu này cùng với tín hiệu $\overline{CS}$ sẽ mở mạch PIC 8259A để đưa thông tin trạng thái lên kênh dữ liệu của CPU.

6. $CAS_0 \div CAS_2$ (Các chân 12, 13, 15), là các đường dây mắc nối tiếp (Cascade) này sẽ hình thành sự kiểm soát kênh, dùng với hệ sử dụng nhiều mạch PIC 8259A ghép với nhau để tăng số lượng các ngoại vi yêu cầu ngắt.

7. $\overline{SP}/\ \overline{EN}$ (Chân 16). Đây là chân 2 chức năng: Chương trình phụ (Slave Program) và mở mạch đệm (Enable Buffer). Chân này được dùng làm đầu ra để kiểm soát các mạch thu/phát đệm trong chế độ mạch đệm (EN) nó được làm đầu vào để mắc nối tiếp các mạch PIC 8259A trong chế độ SP.

8. ITR (Chân17) - Tín hiệu ngắt, là tín hiệu ra hiệu lực cao, dùng để ngắt CPU.

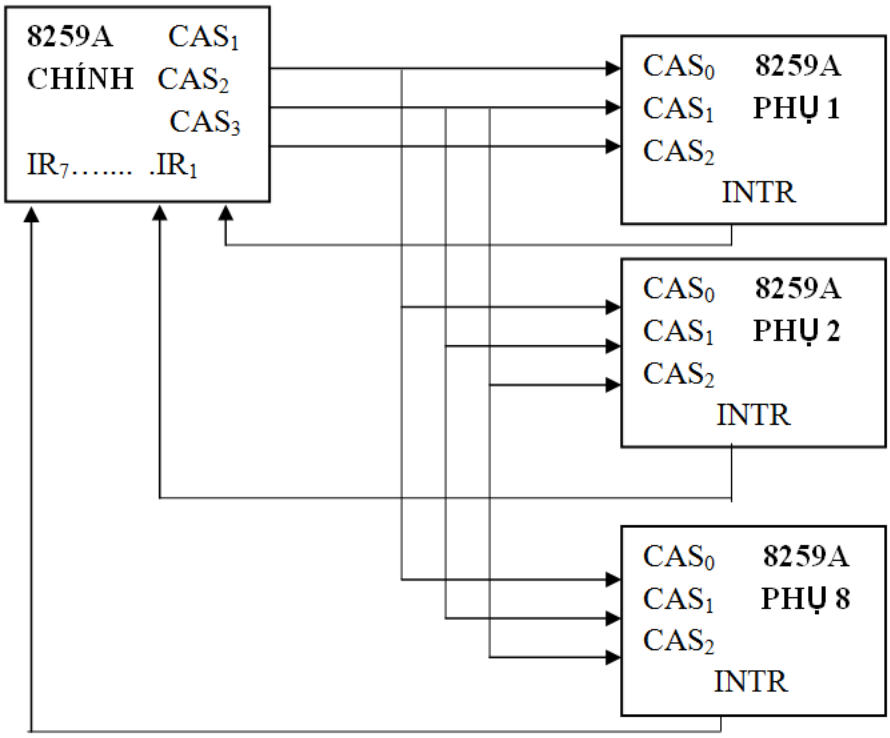
9. $IR_0 \div IR_7$ (Chân18 ÷ 25) - Các tín hiệu yêu cầu ngắt (Interrupt Request), là các tín hiệu vào hiệu lực cao, không đồng bộ, dùng để thông báo yêu cầu ngắt từ một ngoại vi.

10. $\overline{INTA}$ (chân 26) - Tín hiệu báo nhận ngắt (Interrupt Acknowledge), là tín hiệu vào hiệu lực thấp từ CPU, dùng để báo rằng CPU sắp tiến hành phục vụ ngắt.

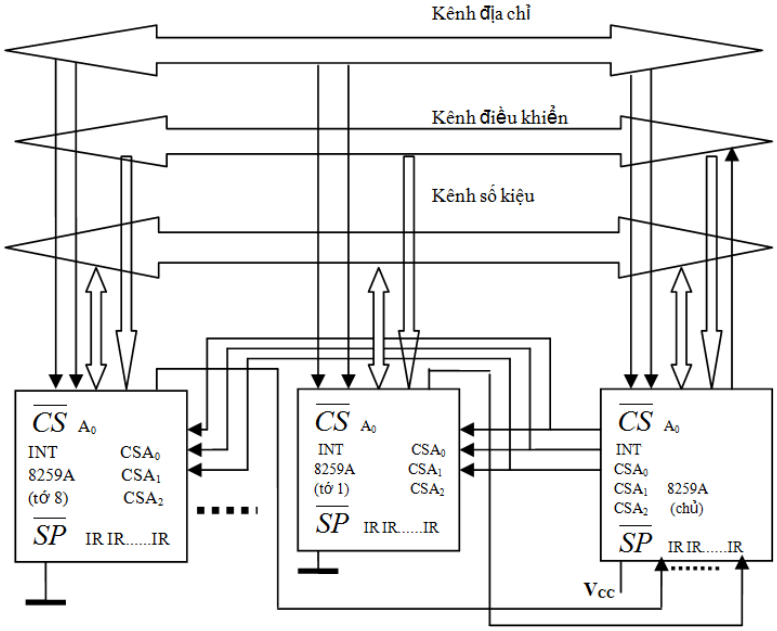
11. A_0 (Chân 27) - Đường địa chỉ A_0 cùng với các tín hiệu chọn chíp, ghi và đọc, dùng để chọn các lời lệnh khác nhau cho vi mạch 8259A.

12. $D_7 \div D_0$ (Các chân 4 ÷ 11) - Các tín hiệu kênh dữ liệu hai chiều này dùng cho việc di chuyển, kiểm soát, trạng thái và thông tin vector ngắt.

Cấu trúc ghép nối các vi mạch 8259A với nhau theo cơ chế chính-phụ (chủ - tớ) khi dùng nhiều vi mạch nối ghép với nhau để dùng cho nhiều ngoại vi được chỉ ra trên hình 7.6 b và c.



Hình 6-25. Sơ đồ ghép nối các vi mạch 8259A với nhau trong máy vi tính để tăng số lượng thiết bị ngoại vi (Phương pháp nối các chân điều khiển)



Hình 6-26. Sơ đồ ghép nối các vi mạch 8259A với nhau trong máy vi tính để tăng số lượng thiết bị ngoại vi (Phương pháp nối tín hiệu và các kênh thông tin)

Vi mạch 8259A chủ chịu trách nhiệm kiểm soát các vi mạch 8259A tớ khác thông qua tín hiệu INT của vi mạch tớ đưa về các chân IR_i tương ứng. Chân $\overline{SP}$ của vi mạch chủ được nối với Vcc, còn chân $\overline{SP}$ của các vi mạch tớ được nối với đất (GND). Các chân $CAS_0 \div CAS_2$ cũng được nối song song với nhau.

Các yêu cầu ngắt của các thiết bị ngoại vi khác nhau, có các mức ưu tiên khác nhau được chỉ ra

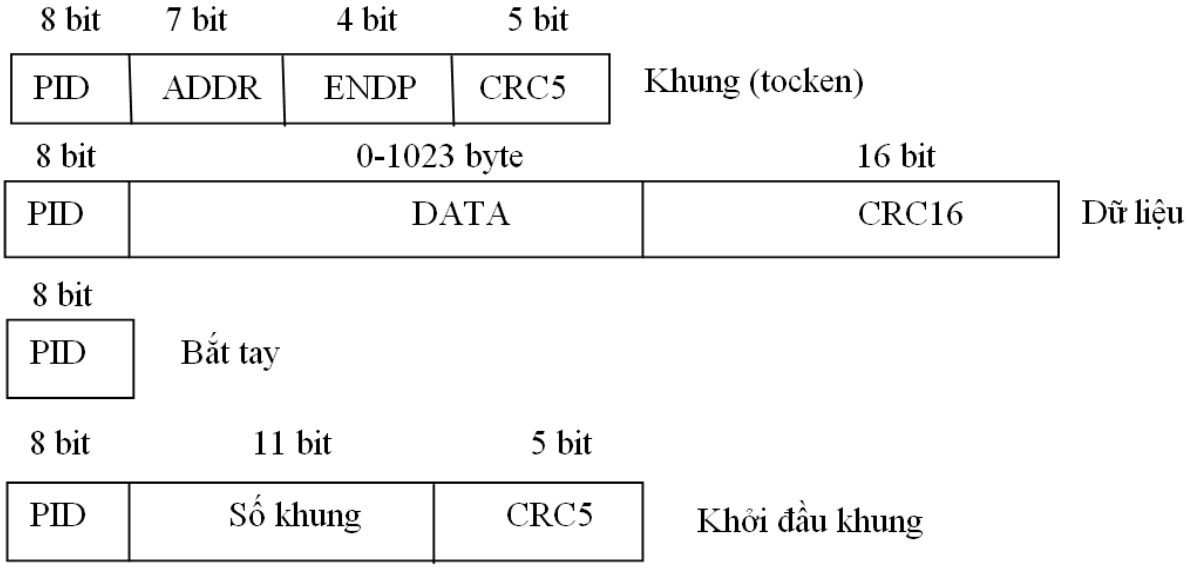
Bảng 6-7. Các giá trị đọc của các mức ưu tiên

Yêu cầu ưu tiên		RST	D7 1	D6 1	D5 A_2	D4 A_1	D3 A_0	D2 1	D1 1	D0 1
Thấp nhất	0	7	1	1	1	1	1	1	1	1
	1	6	1	1	1	1	0	1	1	1
	2	5	1	1	1	0	1	1	1	1
	3	4	1	1	1	0	0	1	1	1
	4	3	1	1	0	1	1	1	1	1
	5	2	1	1	0	1	0	1	1	1
	6	1	1	1	0	0	1	1	1	1
Cao nhất	7	0	1	1	0	0	0	1	1	1

6.4.3. Giao diện đa năng USB

Công nghệ USB (Universal Serial Bus) trong máy vi tính cá nhân đáp ứng nhu cầu một giao diện đơn giản, linh hoạt, dễ sử dụng và rẻ tiền. USB là một giao thức truyền dữ liệu tuần tự giữa máy vi tính (hay chủ USB) với các thiết bị ngoại vi. USB là một tiêu chuẩn và được xác định dựa vào " tín hiệu tốc độ" để giao tiếp. Theo lí thuyết, tốc độ chuẩn của USB 2.0 tối đa là 480 Mbps, tức 60MB/s. Trong khi chuẩn USB 3.0 (mới nhất hiện nay) được xác định với tốc độ tối đa là 4.8 - 5 Gbps, tức 600 - 625MB/s. Như vậy, về mặt lí thuyết USB 3.0 nhanh hơn USB 2.0 hơn 10 lần. Một ưu điểm của USB là tính năng cắm là chạy (người sử dụng có thể cắm thêm hoặc tháo ra một thiết bị ngoại vi mà không cần tắt máy chủ hay cài đặt lại hệ thống).

Giao thức USB: Một lần truyền tin USB cần đến 3 gói: Gói khung (token packet), gói dữ liệu (data packet), gói bắt tay (handshack packet).



Hình 6-27. Cấu trúc giao thức USB

- *PID: loại gói (packet identification);*
- *ADDR: địa chỉ (address);*
- *ENP: điểm cuối (endpoint);*
- *CRC: mã kiểm tra quay vòng dư (cyclic redudancy code);*
- *DATA: dữ liệu*

Mạch định thời gian khả trình 8253 (Programmable Interral Time / Counter)

Mạch định thời gian khả trình 8253 có công dụng chính là thực hiện chức năng định thời gian, đếm, cũng như là phát các trị số biến, phát tín hiệu đồng bộ, kiểm soát one-shot, đếm sự kiện, phát tần số và điều khiển mô tơ.

Mạch định thời gian khả trình 8253 gồm có 3 bộ đếm 16 bít, có khả năng định trước chương trình hoạt động.

Các bộ đếm 16 bít đều có khả năng đặt trước chế độ đếm ngược (giảm). Chúng có thể hoạt động trong chế độ đếm nhị phân hoặc BCD.

Sơ đồ khối của mạch định thời gian khả trình 8253 được chỉ ra trên hình 7.8.

Mạch định thời gian lập trình hoá 8253 có thể chia ra làm 3 phần:

- Phần lối vào chứa các mạch đệm ra số liệu, mạch kiểm soát lôgic đọc/ghi và thanh ghi lời điều khiển.

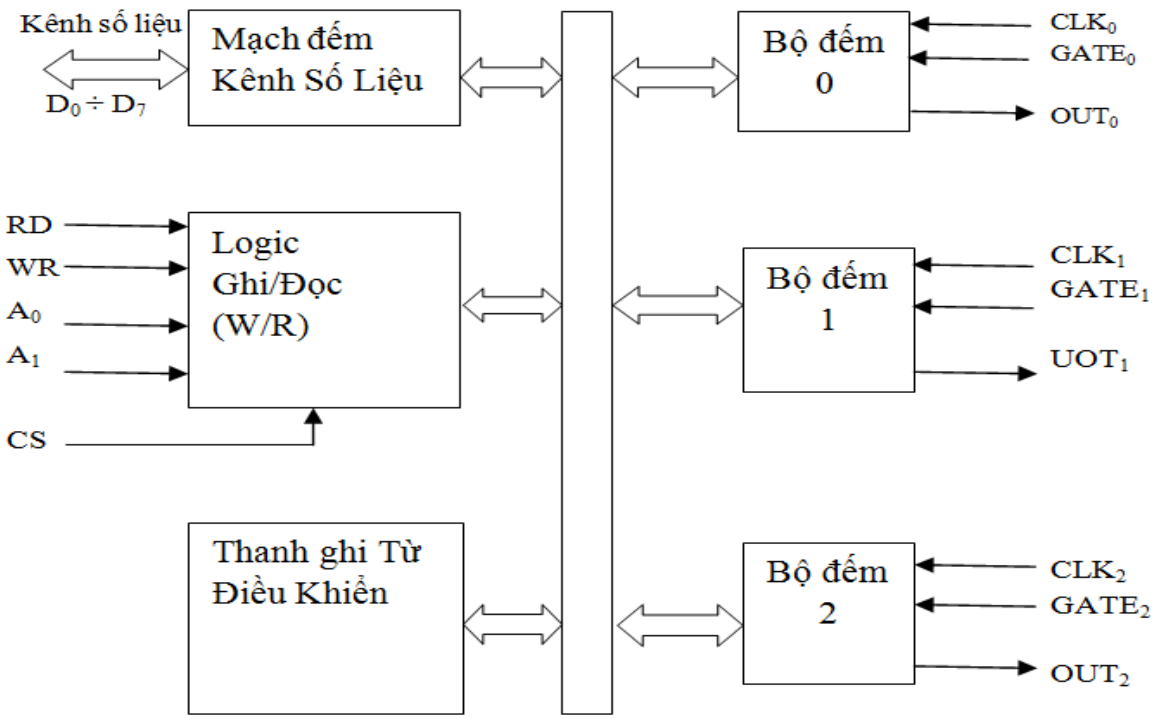
- Phần kênh nội bộ dùng để di chuyển dữ liệu và các tín hiệu điều khiển giữa lối vào và lối ra.

- Phần lối ra có 3 bộ đếm ngược 16 bít riêng biệt nhau.

- Các chân số liệu: $D_0 \div D_7$ là các tín hiệu hai chiều, ba trạng thái nối với các mạch đệm kênh dữ liệu. Các mạch đệm có thể phát, hoặc thu dữ liệu bằng lệnh OUT, hoặc IN của CPU. Các đường tín hiệu này dùng để đặt chế độ hoạt động cho mạch định thời gian khả trình 8253, đặt giá trị đầu tiên cho các bộ đếm và đọc dữ liệu ở bộ đếm đầu ra (các chân 1 ÷ 8).

- Chân $\overline{RD}$ (Chân 22)- Chân đọc, là tín hiệu vào hiệu lực thấp dùng để báo cho mạch định thời gian khả trình 8253 biết rằng CPU đang đưa dữ liệu ra.

- Các chân A_0/A_1 (Các chân 19 và 20) - Các tín hiệu của các chân địa chỉ này dùng để chọn một trong ba bộ đếm hoặc thanh ghi lời điều khiển.



Hình 6-28. Sơ đồ cấu trúc của mạch định thời gian khả trình 8253

Việc chọn các bộ đếm và thanh ghi lời điều khiển được chỉ ra trên bảng 7.3.

Thanh ghi từ điều khiển dùng để định chương trình và khởi đầu cho mạch định thời gian khả trình 8253. Khi cả hai chân A_0 và A_1 đều có mức lôgic 1 thì thanh ghi này được chọn. Sau đó có khả năng nhận thông tin từ các mạch đệm kênh dữ liệu. Thông tin này được lưu trữ trong thanh ghi điều khiển và được dùng để chọn lựa chế độ hoạt động của từng bộ đếm. Mỗi bộ đếm của vi mạch định thời gian khả trình 8253 được định chương trình riêng bằng cách ghi một từ điều khiển vào thanh ghi này.

Bảng 6-8. Chọn các bộ đếm hoặc thanh ghi lời điều khiển

A_1	A_0	Chức năng
0	0	Chọn bộ đếm 0
0	1	Chọn bộ đếm 1
1	0	Chọn bộ đếm 2
1	1	Chọn thanh ghi lời điều khiển

Dạng của lời điều khiển (kiểm soát) như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RL_1	RL_0	M_2	M_1	M_0	BCD

Các bít lời điều khiển này có các chức năng được miêu tả trên bảng 7.4, 7.5, 7.6. Các bít SC (Select Counter) là các bít chọn bộ đếm. Các bit RL (Read/Load) là các bít đọc/nạp số liệu. Các bít M (Mode) là các bít xác định chế độ.

Bảng 6-9. Chức năng của các bít chọn bộ đếm

$D_7(SC_1)$	$D_6(SC_0)$	Chức năng
0	0	Chọn bộ đếm 0
0	1	Chọn bộ đếm 1
1	0	Chọn bộ đếm 2
1	1	Trái luật

Bảng 6-10. Chức năng của các bít đọc/nạp số liệu

$D_5(RL_1)$	$D_4(RL_0)$	Chức năng
0	0	Thao tác chốt bộ đếm
0	1	Đọc/nạp byte thấp
1	0	Đọc/nạp byte cao
1	1	Đọc/nạp byte thấp trước, rồi đến byte cao

Bảng 6-11. Chức năng của các bít xác định chế độ

$D_3(M_2)$	$D_2(M_1)$	$D_1(M_0)$	Chức năng
0	0	0	Chế độ 0
0	0	1	Chế độ 1
x	1	0	Chế độ 2
x	1	1	Chế độ 3
1	0	0	Chế độ 4
1	0	1	Chế độ 5

Ghi chú

- ✓ Giá trị x là trạng thái không xác định (bít đó không cần quan tâm).
- ✓ Chế độ 0: Ngừng ở số đếm cuối; Lối ra có mức cao khi kết thúc đếm.
- ✓ Chế độ 1: Đếm từng bít theo các chương trình.
- ✓ Chế độ 2: Mạch phát các trị số.
- ✓ Chế độ 3: Mạch phát trị số xung vuông.
- ✓ Chế độ 4: Trigơ chốt (Strobe trigger) phần mềm.
- ✓ Chế độ 5: Trigơ chốt (Strobe trigger) phần cứng.

- Chân $\overline{CS}$ (Chân 21) - Tín hiệu chọn chíp, là tín hiệu vào hiệu lực thấp, dùng để mở mạch lôgic điều khiển. Do vậy tín hiệu này sẽ hiệu lực hoá cho các tín hiệu $\overline{RD}$, $\overline{WR}$, A_0 và A_1.

- CLK_0, CLK_1, CLK_2 (Các chân 9, 14, 18). Đây là các tín hiệu đồng bộ lối vào, ứng với đầu vào thời gian của các mạch đếm.

- OUT_0, OUT_1, OUT_2 (Các chân 10, 13, 17). Đây là các tín hiệu ra riêng biệt ứng với các mạch đếm.

- $GATE_0$, $GATE_1$, $GATE_2$ (Các chân 11, 14, 16). Đây là các tín hiệu cổng lối vào, hiệu lực cao, dùng để mở các mạch đếm.

6.4.4. Giao diện cao tốc IEEE 1394

IEEE 1394 là một chuẩn giao tiếp với băng thông cao do IEEE (Institute of Electrical and Electronic Engineers) công bố vào cuối năm 1995 (theo thứ tự công bố chuẩn thứ 1394 như một sự tình cờ hoặc là lý do để chuẩn này được đặt tên như vậy).

IEEE 1394 cũng còn được biết đến với tên khác như: FireWire (hãng Apple) i.LINK (hãng Sony) bởi bản thân IEEE 1394 không phải là một loại cổng, chúng chỉ là một chuẩn giao tiếp để các hãng phần cứng khác phát triển ra các cổng giao tiếp dựa trên chuẩn này nếu được chấp nhận rộng rãi [6].

Tiêu chuẩn chung nhất của IEEE 1394 là IEEE 1394a hoặc IEEE 1394a-2000 với con số 2000 là năm mà chuẩn được giới thiệu. Chuẩn IEEE 1394b được giới thiệu vào đầu năm 2003, chuẩn này hỗ trợ băng thông lên đến 800 Mbps và còn có khả năng mở rộng lên 3.200 Mbps trong tương lai. IEEE 1394b có tốc độ cao hơn các chuẩn IEEE

1394/IEEE 1394a bởi vì chúng hỗ trợ các công nghệ mạng bằng cáp quang và các cáp theo Category 5 UTP. IEEE 1394b hoàn toàn tương thích ngược với các thiết bị theo chuẩn IEEE 1394a.

IEEE 1394a hiện tại hỗ trợ các mức băng thông 100 Mbps, 200 Mbps, và 400 Mbps (tương ứng 12,5 MBps, 25 MBps, và 50 MBps). IEEE 1394 cho phép kết nối đồng thời đến 63 thiết bị bằng các hình thức phân nhánh. IEEE 1394a dùng cáp 6 sợi, trong đó 4 sợi cho truyền tín hiệu, 2 sợi cho cung cấp nguồn điện. Tuy nhiên một loại đầu cắm nhỏ hơn dùng cho các thiết bị tự cung cấp năng lượng chỉ có 4 sợi, trong đó không bố trí 2 sợi cung cấp điện năng. Các DV camcorder thường sử dụng loại giao tiếp IEEE 1394 có 4 sợi bởi chúng tự cung cấp năng lượng qua pin hoặc có nguồn điện riêng.

IEEE 1394b là thế hệ thứ 2 của chuẩn IEEE 1394 với những ứng dụng đầu tiên vào năm 2003. IEEE 1394b có 9 chân, hỗ trợ tốc độ truyền 800/3200 Mbps nên cao hơn, nó có các cải tiến sau so với thế hệ trước nó (IEEE 1394a):

- Tự sửa chữa lỗi (Self-healing loops)
- Hỗ trợ các cáp dài hơn.
- Hỗ trợ cáp CAT5 cũng như cáp quang.

IEEE 1394b có thể giao tiếp với nhiều loại thiết bị có sử dụng các chuẩn giao tiếp theo chuẩn này thông qua các loại cáp chuyển đổi số chân cắm: 9 chân -> 6 chân hoặc 4 chân để phù hợp với các thiết bị sử dụng các cổng giao tiếp theo chuẩn IEEE 1394a.

Các loại cáp sử dụng với IEEE 1394b bao gồm:

Beta: Chỉ dùng riêng với IEEE 1394b (không tương thích với IEEE 1394a)

Bilingual: Cáp loại này dùng đồng thời cho các thiết bị IEEE 1394a/b, giúp các cổng theo IEEE 1394b sử dụng tương thích ngược với các thiết bị theo chuẩn thế hệ trước nó. Có hai loại: 9 chân sang 6 chân và 9 chân ra 4 chân giúp việc tương thích giữa hai chuẩn a/b.

Ở các máy tính cá nhân phổ thông có giá thành thấp, chuẩn này chưa được đưa vào sử dụng bởi chúng làm tăng giá thành sản phẩm. Ở các hệ máy tính tầm trung và cao cấp chúng đã được tích hợp sẵn vào bo mạch chủ.

Nhiều bo mạch chủ bán rời cho các người dùng tự lắp ráp máy tính cũng được tích hợp các cổng IEEE 1394 ở dòng sản phẩm trung và cao cấp (đa số chúng có giá lớn hơn 100 USD ở thời điểm cuối năm 2007).

Một số bo mạch âm thanh thuộc dòng cao cấp cũng được tích hợp sẵn các cổng IEEE 1394, ví dụ các bo mạch âm thanh của Creative cao cấp thường tích hợp sẵn IEEE 1394.

Cũng giống như các giao tiếp khác có nhu cầu sử dụng trên các máy tính cá nhân nhưng không được tích hợp sẵn, một số nhà sản xuất phần cứng đã sản xuất các bo mạch cung cấp các cổng I/O như IEEE 1394. Chúng thường được gắn vào các khe PCI trong máy tính. Không những phục vụ cho các máy tính cá nhân để bàn, nếu các máy tính xách tay chưa được tích hợp sẵn các cổng giao tiếp theo chuẩn IEEE 1394 thì có thể sử dụng các PCMCIA card để mở rộng ra các cổng theo chuẩn IEEE 1394.

Do sự các thiết bị cần băng thông cao chưa phổ biến trong người tiêu dùng nên các cổng IEEE 1394 chưa được thôi thúc tích hợp sẵn trên tất cả các máy tính cá nhân. Đa số các thiết bị ngoại vi hiện nay chỉ khai thác đến bus USB như: ổ usb flash, bàn phím, chuột..., giao tiếp với các máy ảnh số do tốc độ truy cập dữ liệu còn chậm nên cũng chỉ sử dụng USB. Mặt khác USB 2.0 với băng thông 480 Mbps cũng đáp ứng khá tốt cho các thiết bị kể trên.

IEEE 1394 mới chỉ được khai thác ở các ứng dụng cần băng thông lớn, đặc biệt là (một số trong chúng vẫn chưa phổ biến với chuẩn này):

DV cameras: Hầu hết các DV camera (digital camera) ngày nay được tích hợp sẵn một cổng IEEE 1394 (nên IEEE 1394 còn quen được gọi là cổng DV hay DV port). Sự lựa chọn giao tiếp với chuẩn IEEE 1394 bởi chúng đáp ứng với băng thông cho video. Nhiều phần mềm, tiện ích kèm theo các DV camera này được phát triển trên nền chuẩn IEEE 1394 như: chuyển video từ DV cam sang PC, convert các định dạng video trực tiếp theo thời gian thực khi chuyển...

- Các ổ đĩa cứng gắn ngoài và các thiết bị lưu trữ khác gắn ngoài khác yêu cầu băng thông cao, như: ổ đĩa quang (ổ CD, ổ DVD).
- Các máy quét cao cấp với độ phân giải cao.
- Kết nối các máy tính với nhau với băng thông cao.
- Trong tương lai, khi mà nhiều thiết bị sử dụng các chuẩn IEEE 1394 thì chúng có thể thay thế các giao tiếp USB đang phổ biến hiện nay cũng như USB đã thay thế các cổng tuần tự và song song thế hệ trước nó.

CÂU HỎI ÔN TẬP VÀ BÀI TẬP CHƯƠNG 6

Câu 1. Hệ thống vào ra:

a) Nêu các chức năng của module điều khiển vào ra?
b) Các thành phần của module I/O?
c) Vì sao hệ thống PC phải cần đến module I/O?
d) Vẽ sơ đồ trình bày trên cho module I/O.

Câu 2. Hệ thống vào ra:

a) Nêu cấu trúc cơ bản của hệ thống vào-ra ?
b) Nêu các phương pháp địa chỉ hóa cổng vào-ra ?
c) Nêu các phương pháp trao đổi dữ liệu và đặc điểm của mỗi phương pháp ?

Câu 3. Trình bày phương pháp tổ chức I/O được lập:

a) Vào ra bằng chương trình
b) Vào ra điều khiển bằng ngắt
c) Vào ra điều khiển bằng DMA.
d) Vẽ sơ đồ module I/O.

Câu 4. Hệ thống truyền dữ liệu

a) Các cấu hình ghép nối ?
b) Các đặc điểm của mạch thu/phát đệm dữ liệu SN74LS245 ?
c) Vẽ sơ đồ, nêu chức năng và nhiệm vụ của mạch kiểm soát ngắt sử dụng chương trình hoá 8259A ?

TÀI LIỆU THAM KHẢO

1. J.L. Hennesy, D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 4th Edition, 2006.
2. Hồ Khánh Lâm, *Kỹ thuật vi xử lý (tập 1 và tập 2)*, NXB Thông tin và truyền thông, 2008.
3. Li-Shiuan Peh, Abhinav Agarwal, Elliott Fleming, Sang Woo Jun, Asif Khan, Myron King (MIT); Derek Chiou and Jihong Kim, *Computer architecture*, Seoul National University, 2012.
4. Mostafa Abd-El-Barr and Hesham El-Rewini, *Fundamentals of Computer Organization and Architecture*, John Wiley & Sons, Inc, 2005.
5. Nguyễn Đình Việt, *Kiến trúc máy tính,* Nhà xuất bản Đại học Quốc gia Hà Nội, 2006.
6. Trần Quang Vinh, *Cấu trúc máy vi tính*, Nhà xuất bản Đại học Quốc gia Hà Nội, 2007.
7. William Stallings, *Computer Organization and Architecture, Designing for Performance,* 8th Edition, Prentice Hall, 2009.