TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP
KHOA ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH

BÀI GIẢNG HỌC PHẦN:KIẾN TRÚC MÁY TÍNH

Theo chương trình đào tạo 150 TC

Số tín chỉ: 03

Thái Nguyên, năm 2014

GIÁO ÁN HỌC PHẦN:KIẾN TRÚC MÁY TÍNH

Theo chương trình đào tạo 150 TC

Số tín chỉ: 03

Thái Nguyên, ngày 22 tháng 5 năm 2014

Trưởng bộ môn
(ký và ghi rõ họ tên)

Trưởng khoa
(ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

ĐỀ CƯƠNG CHI TIẾT HỌC PHẦN: KIẾN TRÚC MÁY TÍNH

(Học phần bắt buộc)

1. Tên học phần: TEE Kiến trúc máy tính .

2 . Số tín chỉ: 03;

3. Trình độ cho sinh viên năm thứ: 3 (KTMT, SPKT Điện, SPKT Tin) hoặc 4 (Cơ điện tử).

4. Phân bổ thời gian

- Lên lớp lý thuyết: 3 (tiết/tuần) x 12 (tuần) = 36 tiết.
- Thảo luận: 6 (tiết/tuần) x 3 (tuần) = 18 tiết.
- Hướng dẫn bài tập lớn (dài): Không.
- Khác: Không.
- Tổng số tiết thực dạy: = 54 tiết thực hiện.
- Tổng số tiết chuẩn: = 45 tiết chuẩn.

5. Các học phần học trước

6. Học phần thay thế, học phần tương đương

7. Mục tiêu của học phần

Sinh viên nắm được cơ sở kiến trúc của một hệ thống máy tính; Nguyên lý hoạt động của các thành phần như: Cơ chế tính toán của CPU; Giao tiếp giữa các thành phần và vào/ra dữ liệu; Tổ chức và lưu trữ thông tin của bộ nhớ, …

8. Mô tả vắn tắt nội dung học phần

Giới thiệu chung; Cơ sở kiến trúc máy tính; Tính toán luận lý; Tổ chức và kiến trúc bộ nhớ; Giao tiếp và truyền/nhận dữ liệu; Các hệ thống giao tiếp cấp thấp; Thiết kế hệ thống xử lý; Tổ chức của bộ xử lý trung tâm (CPU); Hiệu năng hệ thống; Đa xử lý.

9. Nhiệm vụ của sinh viên

9.1. Phần lý thuyết

1. Dự lớp ≥ 80 % tổng số thời lượng của học phần.
2. Chuẩn bị thảo luận.
3. Bài tập, Bài tập lớn (dài): Không
4. Khác: Tham quan, thực hành, … : Không

9.2. Phần thí nghiệm

Sinh viên phải hoàn thành các bài thí nghiệm sau:

10. Tài liệu học tập

- **Sách, giáo trình chính:**
- *Bài giảng* ***"Kiến trúc và tổ chức máy tính",*** BM Kỹ thuật máy tính

[1]. Nguyễn Đình Việt, Kiến trúc máy tính, NXB Giáo dục, 2000.
[2]. Tống Văn On, Cấu trúc máy tính cơ bản, NXB Thống kê, 2001.

[3]. Tống Văn On, Cấu trúc máy tính nâng cao, NXB Thống kê, 2001.
[4]. Trần Quang Vinh, Nguyên lý phần cứng và kỹ thuật ghép nối máy tính, NXB Giáo dục, 2002.
[5]. Tống Văn On, Hoàng Đức Hải, Giáo trình cấu trúc máy tính, NXB Giáo dục, 2000.
[6]. Nguyễn Nam Trung, *Cấu trúc máy vi tính và thiết bị ngoại vi*, NXB KHKT, 2000.

11. Tiêu chuẩn đánh giá sinh viên và thang điểm

11.1. Các học phần lý thuyết

- **Tiêu chuẩn đánh giá**

1. Chuyên cần;
2. Thảo luận, bài tập;
3. Bài tập lớn (dài);
4. Kiểm tra giữa học phần;
5. Thi kết thúc học phần;
6. Khác.

- **Thang điểm**

- Điểm đánh giá bộ phận chấm theo thang điểm 10 với trọng số như sau:
  + Chuyên cần: 5 %.
  + Thảo luận, bài tập: 10 %.
  + Bài tập lớn (dài): 15 %.
  + Kiểm tra giữa học phần: 20 %.
- Điểm thi kết thúc học phần: 50 %.
- Điểm học phần: Là điểm trung bình chung có trọng số của các điểm đánh giá bộ phận và điểm thi kết thúc học phần làm tròn đến một chữ số thập phân.

11.2. Các học phần thí nghiệm

Điểm học phần thí nghiệm bằng trung bình chung có trọng số điểm các bài thí nghiệm.

12. Nội dung chi tiết học phần

Người biên soạn: ThS. Tăng Cẩm Nhung

13. Lịch trình giảng dạy

- Số tuần dạy lý thuyết: 12 tuần
- Số tuần thảo luận, bài tập: 3 tuần
- Số tuần thực dạy: 15 tuần

Tuần thứ	Nội dung	Tài liệu học tập, tham khảo	Hình thức học

1	**Chương 1: Mở đầu** 1.1 Những khái niệm và nguyên lý cơ bản 1.2 Lịch sử phát triển của máy tính. 1.3 Phân loại máy tính 1.4 Các thành phần cơ bản trong hệ thống máy tính 1.5 Mô hình phân cấp của máy tính 5.1.1. Mức logic số 5.1.2. Mức vi chương trình 5.1.3. Mức máy tính thông thường 5.1.4. Mức máy hệ điều hành 5.1.5. Mức ngôn ngữ assembly 5.1.6. Mức ngôn ngữ bậc cao	[1] - [2]-[3]	Giảng
2	**Chương 2. Biểu diễn thông tin trong máy tính** 2.1.Thông tin và mã hoá thông tin 2.2.. Biểu diễn số 2.2.1. Khái niệm hệ đếm 2.2.2. Chuyển đổi giữa các hệ đếm 2.2.3. Biểu diễn số nguyên	[1] - [2]-[3]	Giảng
3	2.2.4. Biểu diễn số thực 2.1. Các phép toán số học trong hệ nhị phân	[1] - [2]-[3]	Giảng
4	**Chương 3. Mức logic số** 3.1. Giới thiệu về các chip logic 3.2.Các mạch tổ hợp 3.2.1. Bộ dồn kênh và bộ phân kênh 3.2.2 Bộ mã hoá và giải mã 3.2.3. Bộ cộng và bộ trừ 3.3.Bộ tạo xung Clock 3.4.Một số mạch logic cơ bản 3.4.1. Mạch chốt D 3.4.2Mạch lật Flip-Flop 3.4.3. Ứng dụng	[1] - [2]-[3]	Giảng
5	3.5. Bộ nhớ 3.5.1. Tổ chức bộ nhớ 3.5.2. Thiết kế modul nhớ	[1] - [2]-[3]-[4]	
6	**Chương 4 Mức vi chương trình** 4.1. Cách thức hoạt động của CPU 4.2. Chức năng và hoạt động của bộ xử lý 4.2.1. Đơn vị điều khiển 4.2.2. Đơn vị xử lý toán học và logic	[1] - [2]-[3]-[4]	

	4.2.3. Thanh ghi 4.2.4. Hệ thống BUS		
7	4.3. Vi kiến trúc 4.3.1. Đường dữ liệu 4.3.2. Vi chỉ thị 4.3.3. Định thời cho vi chỉ thị 4.3.4. Định trình tự cho các vi chỉ thị 4.4. Ví dụ về kiến trúc trong mức máy thông thường	[1] - [2]-[3]-[4]	
8	Thảo luận	[1] - [2]-[3]-[4]-[7]	Thảo luận
9	**Chương 5. Mức máy thông thường** 5.1. Khuôn dạng lệnh 5.1.1. Lệnh tham chiếu bộ nhớ 5.1.2. Lệnh tham chiếu thanh ghi 5.1.3. Lệnh tham chiếu vào ra 5.2. Các chế độ địa chỉ 5.3. Bộ nhớ chính	[1] - [2]-[3]-[4]	Giảng
10	5.4. Bộ nhớ đệm (Cache) 5.4.1. Hoạt động của bộ nhớ Cache 5.4.2. Ánh xạ Cache 5.4.3. Các giải thuật thay thế	[1] - [2]-[3]-[4]-[7]	Giảng
11	**Chương 6. Mức máy hệ điều hành** 6.1. Giới thiệu về mức máy hệ điều hành 6.2. Bộ nhớ ảo 6.3. Các chỉ thị vào/ra ảo 6.4. Bộ nhớ Cache	[1] - [2]-[3]-[4]-[7]	Giảng
12	**Chương 7. Các thiết bị ngoại vi** 7.1. Khái quát 7.2. Bàn phím – Keyboard 7.3. Màn hình – Monitor 7.4. Đĩa từ và đĩa quang		
13	**Chương 8. Máy tính IBM-PC** • Giới thiệu • Máy IBM PC nguyên thủy		

	• Máy IBM PC/XT, IBM PC/AT • Hệ điều hành DOS của máy IBM PC		
14	Thảo luận	[1] - [2]-[3]-[4]-[7]	Thảo luận
15	Thảo luận	[1] - [2]-[3]-[4]-[7]	Thảo luận

14. Ngày phê duyệt:

15. Cấp phê duyệt:

Đề cương chi tiết học phần đã được Hội đồng khối ngành **Điện – Điện tử và SPKT Điện – Tin học** phê duyệt.

CHƯƠNG 1 MỞ ĐẦU

I. Mục đích:

- Giới thiệu về các khái niệm, nguyên lý cơ bản của Kiến trúc máy tính
- Giúp sinh viên nắm được các thành phần của máy tính
- Giúp sinh viên nắm được mô hình phân cấp của máy tính

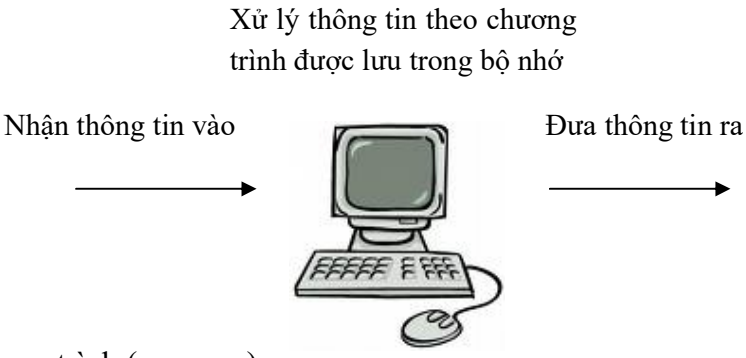
II. Yêu cầu: Sinh viên sau khi học cần phải phân biệt được các thành phần của máy tính, nắm được các mức của mô hình phân cấp.

III. Nội dung: Chương mở đầu được giảng dạy trong 3 tiết của tuần 1.

1.1 Những khái niệm và nguyên lý cơ bản

1.1.1 Khái niệm máy tính

Máy tính (computer) là một thiết bị có khả năng thao tác (lưu trữ, xử lý) trên dữ liệu (thông tin) theo một cách phức tạp và lập trình được. Việc tính toán của nó thực hiện theo một chương trình - một dãy các câu lệnh. Dữ liệu có thể được biểu diễn dưới rất nhiều hình thức của thông tin như: số, ký tự, hình ảnh, âm thanh, … Trước khi phát minh ra máy tính, thuật ngữ computer thường được dùng để ám chỉ một người chuyên làm nhiệm vụ tính toán (human computer)



- Khái niệm chương trình (program):

Chương trình là dãy các câu lệnh nằm trong bộ nhớ, nhằm mục đích hướng dẫn máy tính thực hiện một công việc cụ thể nào đấy. Máy tính thực hiện theo chương trình.

- Khái niệm phần mềm (software):

Bao gồm các thuật toán và các biểu diễn cho máy tính của chúng ta, đó chính là các chương trình. Chương trình có thể được biểu diễn (lưu trữ) trên bìa đục lỗ, băng từ, đĩa từ, … hay các môi trường khác, tuy nhiên cái cơ bản nhất của phần mềm chính là tập hợp các câu lệnh (chỉ thị) tạo nên chương trình chứ không phải là môi trường vật lý được sử dụng để ghi (lưu trữ) chương trình.

- Phần cứng (Hardware):

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

- Phần dẻo (Firmware):

Phần sụn (hay còn gọi là phần nhão) là dạng trung gian giữa phần cứng và phần mềm, nó là phần mềm được nhúng vào các mạch điện tử trong quá trình chế tạo ra các mạch điện tử này. Firmware được sử dụng khi các chương trình hiếm khi hoặc không bao giờ cần thay đổi.

Một ví dụ trực quan cho phần sụn này chính là ROM BIOS chứa các chương trình khởi động, các dịch vụ vào/ra cơ sở, dữ liệu về cấu hình của hệ thống, … mà chúng đã tối ưu, hoàn chỉnh mà không cần phải thay đổi nữa (ít thay đổi). Hay các phần mềm trong đồ chơi hoặc trong các dụng cụ máy móc, điện thoại di động, …

Firmware cũng được sử dụng khi các chương trình không được phép mất đi khi mất điện (nguồn nuôi). Trong nhiều máy tính các vi chương trình thuộc Firmware (chẳng hạn như các chương chình con phục vụ ngắt của BIOS).

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

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

Trong những máy tính đầu tiên thì ranh giới (sự tách biệt) giữa phần cứng và phần mềm là rõ rệt. Phần cứng chỉ thực hiện một số ít các chỉ thị đơn giản, chẳng hạn như lệnh cộng (ADD), lệnh so sánh (CMP), lệnh nhảy (JMP) hoặc một vài lệnh khác. Còn mọi chỉ thị (lệnh) khác được lập trình một cách rõ ràng (chẳng hạn như

tính sin, cos, nhân, …). Nếu chương trình cần nhân (hoặc tính sin của một số) 2 số thì người lập trình phải tự mình viết một chương trình con để thực hiện công việc đó (hoặc lấy nó từ thư viện chương trình mẫu, nếu ai đó đã lập sẵn và đưa vào thư viện chương trình mẫu). Dần dần theo thời gian, những người thiết kế phần cứng nhận thấy rõ ràng một số thao tác được thực hiện khá thường xuyên, đó là những công việc đơn giản (làm nhiều lần thì dễ bị lỗi hay sai sót (nên cần thiết phải xây dựng những mạch điện phần cứng đặc biệt để thực hiện chúng một cách trực tiếp, làm cho nó được thực hiện nhanh hơn.

Kết quả là có khuynh hướng chuyển các thao tác xuống phần cứng (mức 0). Những công việc trước đây được lập trình một cách rõ ràng ở mức máy thông thường thì nay thấy được ở mức dưới, trong phần cứng.

Với sự ra đời của kỷ nguyên vi chương trình và máy tính nhiều mức, một khuynh hướng ngược lại cũng xuất hiện. Trong các máy tính đầu tiên, lệnh cộng (ADD), hoặc trừ (SUB) được thực hiện bằng phần cứng (không ai nghi ngờ về điều đó). Trong một máy tính được lập vi chương trình, chỉ thị ADD của mức máy thông thường (hoặc mức ngôn ngữ assembly) thường được thông dịch bằng vi chương trình chạy ở mức dưới cùng và được thực hiện như một dãy các bước nhỏ hơn (dãy các vi chỉ thị ((vi) chương trình).

- Lấy lệnh (fetch).
- Xác định kiểu của lệnh.
- Định vị dữ liệu sẽ được cộng.
- Lấy dữ liệu (lấy toán hạng).
- Thực hiện cộng (lệnh).
- Cất kết quả.

Qua đó ta thấy nó "vận chuyển" theo khuynh hướng từ dưới lên trên, từ mức phần cứng tới mức vi chương trình.

Khi xây dựng một máy tính nhiều mức, người thiết kế phải quyết định sẽ đặt những gì tại mỗi mức. Cái gì đặt vào phần cứng, cái gì được thực hiện bằng phần mềm.

Trong các máy tính hiện đại ngày nay có rất nhiều chỉ thị được thực hiện bằng phần cứng hoặc bằng vi chương trình, nhưng ban đầu (khởi thuỷ – nguyên văn) chúng được lập trình rất rõ ràng tại mức máy thông thường.

Ví dụ:

- Các chỉ thị cho phép nhân và chia số nguyên.
- Các chỉ thị đối với số dấu phảy động.

- Các chỉ thị số học có độ chính các kép (các phép tính số học trên các con số có số chữ số có nghĩa nhiều gấp hai bình thường).
- Các chỉ thị gọi chương trình con và trở về chương trình con từ chương trình chính sau khi thực hiện xong chương trình con.
- Các chỉ thị để đếm (cộng 1 vào một biến: inc i <=> i:=i+1).
- Các chỉ thị xử lý xâu ký tự.

Tóm lại chúng ta thấy rằng ranh giới giữa phần cứng và phần mềm là tuỳ ý và chúng không ngừng thay đổi theo sự phát triển cũng như yêu cầu trong từng lĩnh vực cụ thể (chẳng hạn: đặt hàng một máy tính chuyên dụng nào đó (sự thiết kế sẽ được định hướng theo yêu cầu của lĩnh vực đó).

Các phần mềm ngày nay có thể là phần cứng trong nay mai hoặc ngược lại. Hơn thế nữa ranh giới giữa các mức khác nhau cũng dễ thay đổi. Theo quan điểm của từng lập trình viên thì việc chọn một chỉ thị (lệnh) thực tế được thi hành như thế nào không quan trọng (có thể chỉ quan tâm tới tốc độ thực hiện của chỉ thị đó). Một người lập trình ở mức máy (thông thường) có thể sử dụng chỉ thị nhân của mức này như thể nó là một chỉ thị cho phần cứng mà không phải quan tâm lo lắng gì cả, thậm chí cũng không cần biết nó có thực sự là một chỉ thị cho phần cứng hay không. Phần cứng của người này có thể gọi là phần mềm của ngeời khác.

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

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

1.1.2 Nguyên lý xây dựng máy tính điện tử

Có hai nguyên lý cơ bản để xây dựng máy tính điện tử là: nguyên lý số và nguyên lý tương tự.

- *Nguyên lý số*: sử dụng các trạng thái rời rạc của một đại lượng vật lý để biểu diễn số liệu, nguyên lý này còn được gọi là nguyên lý đếm. Ví dụ về tình trạng rời rạc của đại lượng vật lý theo nguyên lý số được thể hiện trong bảng 1-1.

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

Bảng 1-1 Trạng thái trong nguyên lý số

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

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

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

Bảng 1-2 Đại lượng biến thiên trong nguyên lý tương tự

1.1.3 Ngôn ngữ máy

Lệnh máy: Các mạch điện tử của máy tính có thể hiểu và thực hiện trực tiếp được một tập hợp hữu hạn các lệnh rất đơn giản thường được gọi là chỉ thị (instruction) máy hay lệnh máy, chẳng hạn: Cộng hai số với nhau; Kiểm tra xem một số có bằng không hay không; Vận chuyển một nhóm dữ liệu từ vùng này của bộ nhớ sang một vùng khác.

Ngôn ngữ máy: Tập các chỉ thị máy tạo nên ngôn ngữ này để giao tiếp với máy tính được gọi là ngôn ngữ máy (machine language).

Hầu hết các ngôn ngữ máy rất đơn giản, nên việc sử dụng chúng là khó và tẻ nhạt. Khắc phục vấn đề này bằng cách thiết kế một tập chỉ thị mới dễ sử dụng hơn tập chỉ thị máy đã được xây dựng ngay bên trong máy. Tập các chỉ thị này cũng tạo thành một ngôn ngữ mà chúng ta sẽ gọi là ngôn ngữ mức (Level) 2 - L2, còn tập chỉ thị được xây dựng ngay bên trong máy chúng ta sẽ gọi là ngôn ngữ L1.

Chương trình: Một dãy các chỉ thị mô tả việc thực hiện một nhiệm vụ cụ thể như thế nào được gọi là chương trình (program).

Hai cách để máy tính thi hành được các chương trình viết bằng L2:

Cách thứ nhất: đầu tiên thay thế mỗi chỉ thị trong chương trình này bằng một dãy tương đương gồm các chỉ thị trong ngôn ngữ L1. Kết quả thu được một chương

trình gồm toàn các chỉ thị thuộc ngôn ngữ L1. Sau đó máy tính thực hiện chương trình mới bằng ngôn ngữ L1 chứ không phải chương trình cũ bằng ngôn ngữ L2. Kỹ thuật này được gọi là dịch hay biên dịch - Compilation (hay translation), còn chương trình thực hiện việc này được gọi là Trình biên dịch - Compiler.

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

1.2 Lịch sử phát triển của máy tính

1.2.1 Thế hệ số không – máy tính cơ khí.

Nhà khoa học Pháp Blase Pascal (1623-1662) là người đầu tiên chế tạo được một chiếc máy tính hoạt động được (1642). Đây Hoàn toàn là một chiếc máy tính cơ khí, sử dụng các bánh răng, năng lượng cung cấp cho máy là sức người - quay tay. Máy tính của Pascal chỉ làm được phép tính cộng và trừ.

Ba mươi năm sau nhà bác học Đức Baron Gottfried von Leibniz (1646-1716) đã chế tạo thành công một chiếc máy tính cơ khí khác, ngoài hai phép tính cộng và trừ nó còn có thể thực hiện phép nhân và chia (sau Blase Pascal 30 năm).

Sau đó, giáo sư Charles Babbage đã thiết kế và xây dựng máy sai phân (difference engine). Nó được thiết kế để chạy một giải thuật đơn: phương pháp sai phân hữu hạn sử dụng các đa thức và cũng chỉ thực hiện các phép toán cộng và trừ. Nam 1834, Babbage thiết kế và xây dựng máy phân tích (analytical engine). Máy phân tích có 4 thành phần: bộ lưu trữ (bộ nhớ), bộ tính toán, thành phần nhập (đầu đọc thẻ đục lỗ) và thành phần xuất (in và đục lỗ). Bộ tính toán có thể nhận các toán hạng từ bộ lưu trữ, thực hiện phép toán cộng, trừ, nhân hay chia chúng và trả kết quả về bộ lưu trữ.

Phát triển tiếp theo của máy phân tích là máy đa năng. Máy đọc lệnh từ các thẻ đục lỗ và thực thi chúng. Bằng cách đục lỗ một chương trình khác trên thẻ nhập, máy phân tích có khả năng thực hiện các tính toán khác. Lập trình viên máy tính đầu tiên là Ada Lovelace đã tạo ra phần mềm cho máy phân tích.

Vào những năm 1930, Konrad Zuse xây dựng một chuỗi các máy tính toán tự động bằng cách sử dụng các relay từ. Sau đó, John Atanasoff và George Stibbitz đã thiết kế các máy tính (calculator). Máy của Atanasoff sử dụng số nhị phân và có các tụ điện làm cho bộ nhớ được làm tươi theo chu kỳ. Tuy nhiên, máy này bị thất bại do công nghệ phần cứng không tương xứng với ý tưởng thiết kế.

Năm 1944, Aiken hoàn tất máy tính Mark 1, có tất cả 72 từ, mỗi từ 23 số thập phân và có thời gian một chu kỳ là 6 giây. Việc nhập và xuất thực hiện bằng các bằng giấy đục lỗ.

1.2.2 Máy tính đèn điện tử - thế hệ thứ nhất

Năm 1943, máy tính số điện tử đầu tiên trên thế giới bắt đầu hoạt động, máy Colossus. Colossus do Alan Turing thiết kế nhằm thực hiện giải mã các thông điệp đã mã hóa trong chiến tranh thế giới thứ 2. Cung trong năm 1943, Mauchley và Presper Eckert bắt đầu tiến hành xây dựng máy tính ENIAC (Electronic Numerical Integrator And Computer). ENIAC gồm

1800 dèn diện tử và 1500 relay, cân nặng 30 tấn, công suất tiêu thụ 140 kWh. Nó có tất cả 20 thanh ghi, mỗi thanh ghi có thể lưu trữ một số thập phân 10 chữ số.

Sau đó, John von Neumann thiết kế máy IAS dựa cơ sở trên máy EDVAC, là một phiên bản nâng cao của ENIAC. Máy von Neumman có 5 phần co bản: bộ nhớ, đơn vị luận lý số học (ALU – Arithmetich Logic Unit), đơn vị điều khiển chương trình, thiết bị nhập và thiết bị xuất. Bộ nhớ có tất cả 4096 từ, mỗi từ lưu trữ 40 bit. Mỗi từ chứa 2 lệnh 20 bit hay một số nguyên có dấu 39 bit. Mỗi lệnh 20 bit gồm có 8 bit xác định loại lệnh và 12 bit xác định 1 trong 4096 từ nhớ

Vào cùng thời gian của máy IAS, các nhà nghiên cứu ở MIT cũng đang xây dựng một máy tính, máy Whirlwind 1. Nó có từ dài 16 bit và thiết kế để điều khiển thời gian thực.

1.2.3 Máy tính transistor – thế hệ thứ hai

Năm 1948, John Bardeen, Walter Brattain và William Shockley phát minh ra transistor đã làm cuộc cách mạng trong lĩnh vực máy tính. Máy tính transistor đầu tiên duợc xây dựng tại MIT, máy TX-0 (Transistorized experimental computer 0), có 16 bit tương tự như Whirlwind 1.

Năm 1961, máy tính PDP-1 xuất hiện có 4K từ 18 bit và khoảng thời gian một chu kỳ là 5 μs. Vài năm sau, PDP-8 ra dời có 12 bit nhưng giá thành rẻ hơn PDP-1 rất nhiều (16.000 USD so với 120.000 USD). PDP-8 có một đổi mới đó là hình thành một bus đơn gọi là omnibus trong đó bus là tập hợp các dây nối song song dùng để kết nối các thành phần của máy tính.

Trong khi dó, IBM xây dựng một phiên bản của 709 bằng transistor, đó là máy tính 7094 có thời gian một chu kỳ là 2 μs và bộ nhớ 32K từ 36 bit. Nam 1964, công ty CDC giới thiệu máy 6600 có tốc dộ nhanh hơn 7094 do bên trong CPU có một co chế song song. CPU có vài đơn vị thực hiện phép cộng, các đơn vị khác thực hiện phép nhân, chia và tất cả chúng đều hoạt động song song. Với một công việc, máy có khả nang thực thi 10 lệnh dồng thời.

1.2.4 Máy tính IC – thế hệ thứ ba

Vi mạch được phát minh cho phép đặt vài chục transistor trong một chip đơn. Việc này giúp cho các máy tính xây dựng trên IC nhỏ hơn, nhanh hơn và rẻ hơn so với các máy tính transistor. Lúc này, IBM giới thiệu một sản phẩm đơn, máy System 360, duợc thiết kế dựa trên các vi mạch. Đổi mới quan trọng trong 360 là khả năng đa lập trình (multiprogramming), có vài chương trình trong bộ nhớ đồng thời để khi một chương trình đang chờ xuất / nhập dữ liệu thì chương trình khác có

thể tính toán. Một đặc trưng khác của 360 là không gian địa chỉ lớn (thời điểm lúc đó), với 224 byte nhớ (16 MB).

1.2.5 Máy tính cá nhân và VLSI – thế hệ thứ tư

Vào thập niên 80, vi mạch VLSI (Very Large Scale Integrate) có khả năng chứa vài chục ngàn, vài trăm ngàn và vài triệu transistor trên một chip đơn đã được chế tạo. Sự phát triển này dẫn đến việc sản xuất các máy tính nhỏ hơn và nhanh hơn. Do đó, giá cả đã giảm xuống đến mức một cá nhân có thể sở hữu một máy tính. Các máy tính cá nhân thường dùng cho việc xử lý từ, các bảng tính và các ứng dụng tương hỗ khác. Các máy tính trong thế hệ này có thể chia thành 5 loại: máy tính cá nhân, máy tính mini, mainframe, siêu máy tính.

Máy tính mini sử dụng trong các ứng dụng thời gian thực như điều khiển không lưu hay tự động hóa. Siêu máy tính mini dùng trong các hệ thống chia sẻ thời gian, các máy chủ. Mainframe dùng trong các nhóm công việc lớn hay đòi hỏi cơ sở dữ liệu lớn, … Siêu máy tính dược thiết kế đặc biệt để cực đại hóa số các thao tác dấu chấm động trong 1s (FLOP – floating point operations per second). Máy tính nào có tốc độ dưới 1 GF/s thì không được xem là siêu máy tính.

1.3 Phân loại máy tính

Có nhiều phương pháp và cách phân loại khác nhau như: theo nguyên lý xây dựng, theo thế hệ, theo mục tiêu thiết kế v.v. Ở đây ta nêu lên một số phương pháp phân loại máy tính điện tử.

a) Phân loai theo phương pháp truyền thống.

- Máy vi tính (Microcomputer)

Một thiết bị hay hệ thống điện tử có khả năng xử lý dữ liệu, dùng để tính toán hay kiểm soát các hoạt động mà có thể biểu diễn dưới dạng số hay quy luật lôgic.

- Máy tính nhỏ (Minicomputer)

Là một dạng máy tính nhỏ cầm tay, với tốc độc trung bình, có khả năng xử lý và thực thi các chương trình cỡ nhỏ và chuyên biệt.

- Máy tính lớn (Mainframe Computer)

Máy tính cỡ lớn, thường là các máy tính chủ trong các hệ thống mạng của công ty hoặc nhà máy

- Siêu máy tính (Super Computer)

Cray-2; máy tính nhanh nhất thế giới trong thời gian 1985–1989.

Một siêu máy tính là một máy tính vượt trội trong khả năng và tốc độ xử lý. Thuật ngữ Siêu Tính Toán được dùng lần đầu trong báo New York World vào năm 1920 để nói đến những bảng tính (tabulators) lớn của IBM làm cho trường Đại học Columbia. Siêu máy tính hiện nay có tốc độ xử lý hàng trăm teraflop (một teraflop tương đương với hiệu suất một nghìn tỷ phép tính/giây) hay bằng tổng hiệu suất của 6.000 chiếc máy tính hiện đại nhất hiện nay gộp lại (một máy có tốc độ khoảng từ 3-3,8 gigaflop).

Có thể hiểu siêu máy tính là hệ thống những máy tính làm việc song song.

Siêu máy tính Roadrunner của IBM - 2008

Siêu máy tính IBM Blue Gene/L nhanh nhất thế giới - 2006.

b) Phân loại theo phương pháp hiện đại

- Máy tính để bàn (Desktop Computer)

Là máy tính cá nhân, hay máy tính đa năng, đáp ứng như cầu mọi người sử dụng chung trong các lĩnh vực Home, office, …

- Máy chủ (Servers)

Có nhiều loại máy chủ khác nhau, phục vụ các yêu cầu từ các máy khách trong hệ thống mạng. Như máy chủ WEB, máy chủ dữ liệu, máy chủ tên miền,…

- Máy tính nhúng (Embedded Computer)

Máy tính được đặt vào trong một thống lớn, làm nhiệm vụ xử lý thông tin và điều khiển hoạt động cho một phần hoặc toàn bộ hệ thống.

VD: Hệ thống điều khiển điện và điều khiển Oto.

c) Phân loại theo nguyên lý xây dựng máy tính

Theo phương pháp này máy tính được phân chia thành hai lớp là máy tính tương tự và máy tính số. Mỗi lớp lớn này lại có thể được chia thành các lớp con, ví dụ máy tính đa năng và máy tính chuyên dụng . . .

- Máy tính số (Digital Computer)

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

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

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

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

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

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

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

Các máy tính chuyên dụng được sử dụng rộng rãi hiện nay:

- ✓ Các máy tính điều khiển, nằm trong các hệ thống điều khiển những đối tượng và quá trình thực (Robot, điều khiển máy bay, vệ tinh, ...).
- ✓ Các máy tính được thiết kế để giải một loại (dạng) bài toán nhất định, chẳng hạn tính tổng các tích 2 số trong các bài toán về dự báo khí tượng, giải hệ phương trình vi phân thường, . . .
- ✓ Các máy tính tìm thông tin, dùng để xử lý những lượng lớn các thông tin chữ và số (tìm kiến thông tin trên mạng).

- **MTS đa năng:** Là loại MTS được chế tạo ra để giải một lớp lớn các bài toán mà thành phần của lớp bài toán này có thể còn chưa được nêu đầy để khi thiết kế máy. Những tính chất cơ bản của MTS đa năng là:

- Có tập lệnh tương đối lớn, bao gồm nhiều nhóm, mỗi nhóm phục vụ cho một loại thao tác nhất định, để có thể thi hành nhiều loại chương trình khác nhau một cách có hiệu quả.
- Kích thước các thanh ghi trong bộ xử lý và độ lớn của một đơn vị bộ nhớ (word - từ nhớ) phải không quá lớn cũng không quá nhỏ, thông thường bằng bội số của 8. Như vậy chúng có thể chứa vừa vặn một số ký tự ASCII 8 bit hoặc chứa được cón số đủ lơn cho hầu hết các chương trình ứng dụng chạy trên máy. Các chương trình sạn thảo văn bản, chương trình quản lý, . . . nói chung đều có dữ liệu kiểu ký tự.
- Các thiết bị nhớ có thể nhận vào, lưu trữ và đưa ra các số liệu, các kết quả và các chương trình. Còn trong MTS chuyên dụng các chương trình, hằng số, . . . thường được cứng hoá.

- Hệ thống trao đổi thông tin (giao diện) giữa máy và người sử dụng thuận tiện, giảm nhẹ các công việc mà con người phải thực hiện trong quá trình máy tính hoạt động.

- **Máy tính tương tự (Analog Computer)**

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

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

GTE Analog Computer EA22

Các MTTT vận hành rất thuận tiện, thường đưa ra kết quả dưới dạng đồ thị, đặc biệt với thời gian cực kỳ ngắn (tốc độ thi hành rất cao).

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

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

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

Trước chiến tranh thế giới thứ II, máy tính tương tự điện và cơ khí được coi là một thành tựu vĩ đại và rất nhiều người nghĩ rằng nó là tương lai của khoa học tính toán. Các máy tính tương tự được sử dụng các đại lượng vật lý biến thiên liên

tục như dòng điện và điện áp hay tốc độ quay của các trục để biểu diễn các giá trị đang được xử lý. Một ví dụ điển hình đó là chiếc máy tích phân bằng nước được xây dựng năm 1936. Không giống như các máy tính số hiện đại, máy tính analog không quá phức tạp và cần phải thiết lập cấu hình lại (lập trình lại) một cách thủ công để chuyển chúng từ công việc này sang giải quyết một vấn đề khác. Các máy tính tương tự đã đem lại nhiều lợi ích trong giai đoạn khai sinh của máy tính số do chúng có thể giải quyết nhiều bài toán phức tạp trong khi khả năng của các máy tính số ban đầu còn rất giới hạn. Nhưng cùng với sự phát triển của nhanh hơn và sử dụng bộ nhớ lớn hơn của máy tính số, máy tính tương tự dần bị thay thế, đặc biệt là sự phát triển của kỹ thuật lập trình và kỹ thuật mã hoá.

Máy tính tương tự được ứng dụng mạnh nhất trong các thiết bị ngắm bắn cho vũ khí, ví dụ như máy ngắm ném bom Norden và các máy tính toán cho pháo binh trên chiến trường. Một số máy tính loại này vẫn còn được sử dụng vài thập kỷ sau CTTG II.

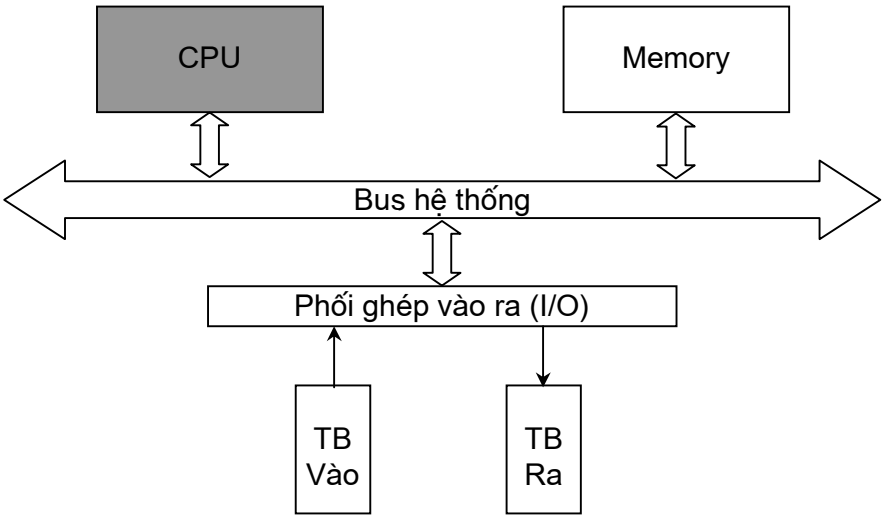
Máy tính tương tự dạng lai, được điều khiển bởi các thiết bị điện tử số, vẫn giữ vai trò quan trọng cho tới những năm 1950, 1960, và sau đó trong một số ứng dụng đặc biệt.

- **Máy tính lai (Hybrid Computer)**

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

1.4 Các thành phần cơ bản trong hệ thống máy tính

Mô hình chung của một máy tính



Hình 1-1 Mô hình chung của một hệ thống máy tính

1.4.1 CPU

CPU viết tắt của chữ Central Processing Unit (tiếng Anh), tạm dịch là đơn vị xử lí trung tâm. CPU có thể được xem như não bộ, một trong những phần tử cốt lõi nhất của máy vi tính. Nhiệm vụ chính của CPU là xử lý các chương trình vi tính và dữ kiện. CPU có nhiều kiểu dáng khác nhau. Ở hình thức đơn giản nhất, CPU là một con chip với vài chục chân. Phức tạp hơn, CPU được ráp sẵn trong các bộ mạch với hàng trăm con chip khác. CPU là một mạch xử lý dữ liệu theo chương trình được thiết lập trước. Nó là một mạch tích hợp phức tạp gồm hàng triệu transistor

Cấu trúc cơ bản CPU

- Đơn vị điều khiển (CU:Control Unit): Điều khiển hoạt động của máy tính theo chương trình đã định sẵn.
- Đơn vị số học và logic (ALU: Arithmetic And Logic Unit): thực hiện các phép toán số học và logic trên các dữ liệu cụ thể.
- Tập thanh ghi (RF: Register File): Lưu trữ các thông tin tạm thời phục vụ cho hoạt động của CPU.
- Đơn vị nối ghép BUS(BIU: Bus Interface Unit): kết nối và trao đổi thông tin giữa Bus bên trong và Bus bên ngoài CPU.

1.4.2 Bộ nhớ trong

Là loại bộ nhớ mà CPU có thể truy cập trực tiếp, có tốc độ cao và dung lượng thường nhỏ. Bộ nhớ trong chia làm 2 loại

- Bộ nhớ chính (main memmory): ROM và RAM
- Bộ nhớ đệm Cache

- RAM (Random access memory), hay Bộ nhớ truy cập ngẫu nhiên:

Tốc độ truy cập nhanh, lưu trữ giữ liệu tạm thời, dữ liệu sẽ bị mất vĩnh viễn khi không còn nguồn điện cung cấp.

- ROM (Read Only Memory), hay Bộ nhớ chỉ đọc:

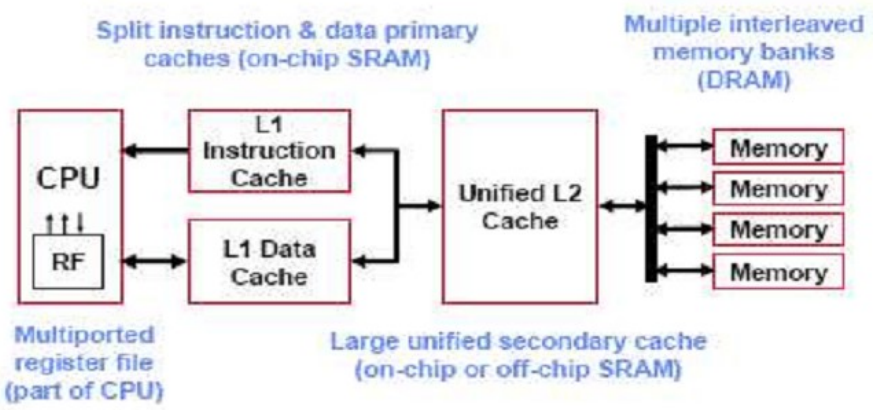
Lưu trữ các chương trình mà khi mất nguồn điện cung cấp sẽ không bị (xóa) mất. Ngày nay còn có công nghệ FlashROM tức bộ nhớ ROM không những chỉ đọc mà còn có thể ghi lại được, nhờ có công nghệ này BIOS được cải tiến thành FlashBIOS.

- Cache:

Cache là tên gọi của bộ nhớ đệm – nơi lưu trữ các dữ liệu nằm chờ các ứng dụng hay phần cứng xử lý. Mục đích của nó là để tăng tốc độ xử lý. Cache là một cơ chế lưu trữ tốc độ cao đặc biệt. Nó có thể là một vùng lưu trữ của bộ nhớ chính hay một thiết bị lưu trữ tốc độ cao độc lập.

Cache cấu tạo bằng bộ nhớ tĩnh (SRAM) có tốc độ cao nhưng đắt tiền thay vì bộ nhớ động (DRAM) có tốc độ thấp hơn và rẻ hơn, được dùng cho bộ nhớ chính. Cơ chế lưu trữ bộ nhớ cache này rất có hiệu quả. Bởi lẽ, hầu hết các chương trình thực tế truy xuất lặp đi lặp lại cùng một dữ liệu hay các lệnh giống nhau. Nhờ lưu trữ các thông tin này trong SRAM, máy tính sẽ khỏi phải truy xuất vào DRAM vốn chậm chạp hơn.

Một số bộ nhớ cache được tích hợp vào trong kiến trúc của các bộ vi xử lý. Chẳng hạn, CPU Intel đời 80486 có bộ nhớ cache 8 KB, trong khi lên đời Pentium là 16 KB. Các bộ nhớ cache nội (internal cache) như thế gọi là Level 1 (L1) Cache (bộ nhớ đệm cấp 1). Các máy tính hiện đại hơn thì có thêm bộ nhớ cache ngoại (external cache) gọi là Level 2 (L2) Cache (bộ nhớ đệm cấp 2). Các cache này nằm giữa CPU và bộ nhớ hệ thống DRAM. Sau này, do nhu cầu xử lý nặng hơn và với tốc độ nhanh hơn, các máy chủ (server), máy trạm (workstation) và mới đây là CPU Pentium 4 Extreme Edition được tăng cường thêm bộ nhớ đệm L3 Cache.



Hình 1-2. Bộ nhớ đệm Cache

1.4.3 Bộ nhớ ngoài

Có dung lượng lơn, để lưu các chương trình và dữ liệu lâu dài, như HDD, CDROM, Tape, ...

Các loại bộ nhớ dựa trên công nghệ FlashROM: Kết hợp với chuẩn giao tiếp máy tính USB (Universal Serial Bus) tạo ra các bộ nhớ máy tính di động thuận tiện và đa năng như: Các thiết bị giao tiếp USB lưu trữ dữ liệu, thiết bị giao tiếp USB chơi nhạc số, chơi video số; khóa bảo mật qua giao tiếp USB; thẻ nhớ... Dung lượng thiết bị lưu trữ FlashROM đã lên tới 32GB (Samsung công bố năm 2005), trong tương lai, có thể FlashROM sẽ dần thay thế các ổ đĩa cứng, các loại đĩa CD, DVD...

1.4.4 Hệ thống vào ra (Input/Output System)

Giúp máy tính trao đổi thông tin với thế giới bên ngoài, bao gồm hai hoạt động chính là nhận thông tin Input và gửi thông tin ra Output.

Nhập hay đầu vào (Input): Các bộ phận thu nhập dữ liệu hay mệnh lệnh như là bàn phím, chuột...

Xuất hay đầu ra (Output): Các bộ phận trả lời, phát tín hiệu, hay thực thi lệnh ra bên ngoài như là màn hình, máy in, loa, ...

Thông qua hệ thống vào ra máy tính có thể trao đổi thông tin với thiết bị ngoài vi.

Các thiết bị ngoại vi cơ bản:

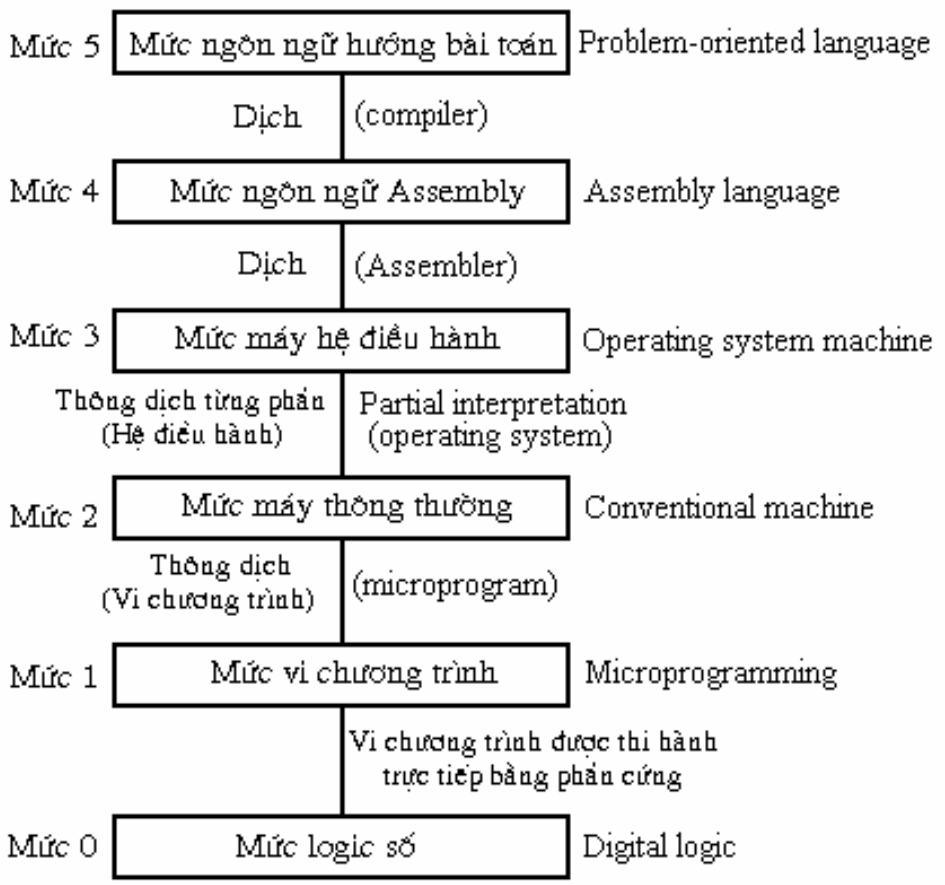
- Thiết bị vào: bàn phím, chuột, …
- Thiết bị ra: máy in, màn hình,…
- Thiết bị nhớ: đĩa từ, quang,….
- Thiết bị truyền thông: Modem,…

1.4.5 Hệ thống bus

Hệ thống bus (Bus system) làm nhiệm vụ vận chuyển thông tin giữa các phần khác nhau trong máy tính (Hệ thống Bus bao gồm bus dữ liệu dùng để vận chuyển dữ liệu từ bộ nhớ tới CPU hoặc ngược lại. Bus địa chỉ dùng để vận chuyển tín hiệu địa chỉ (ô nhớ hay cổng vào/ra do CPU phát ra. Bus điều khiển dùng để vận chuyển tín hiệu điều khiển do CPU phát ra để điều khiển các khối trong hệ thống hoặc do thiết bị ngoại vi gửi tới CPU yêu cầu thực hiện một công việc nào đó).

1.5 Mô hình phân cấp của máy tính

Một máy với n mức có thể được xem như n máy ảo khác nhau, mỗi máy ảo có một ngôn ngữ máy tương ứng. Chúng ta sẽ sử dụng thuật ngữ "mức" và "máy ảo" với cùng một ý nghĩa. Chỉ có các chương trình được viết bằng ngôn ngữ L1 là có thể được thực hiện trực tiếp bằng các mạch điện tử mà không cần sự can thiệp trung gian của việc dịch hoặc thông dịch. Các chương trình được viết bằng L2, L3,..., Ln hoặc là phải được thông dịch bằng một trình thông dịch chạy ở một mức thấp hơn hoặc là phải được dịch sang một ngôn ngữ khác tương ứng với một mức thấp hơn. Hình 1.03 dưới đây mô tả các máy tính nhiều mức hiện đại



Hình 1-3. Mô hình phân cấp máy tính

Cấp 0 chính là phần cứng của máy tính. Các mạch điện tử của cấp này sẽ thực thi các chương trình ngôn ngữ máy của cấp 1. Trong cấp logic số, đối tượng quan tâm là các cổng logic. Các cổng này được xây dựng từ một nhóm các transistor.

Cấp 1 là cấp ngôn ngữ máy thật sự. Cấp này có một chương trình gọi là vi chương trình (microprogram), vi chương trình có nhiệm vụ thông dịch các chỉ thị của cấp 2. Hầu hết các lệnh trong cấp này là di chuyển dữ liệu từ phần này đến phần khác của máy hay thực hiện việc một số kiểm tra đơn giản.

Mỗi máy cấp 1 có một hay nhiều vi chương trình chạy trên chúng. Mỗi vi chương trình xác định một ngôn ngữ cấp 2. Các máy cấp 2 đều có nhiều điểm chung ngay cả các máy cấp 2 của các hãng sản xuất khác nhau. Các lệnh trên máy cấp 2 được thực thi bằng cách thông dịch bởi vi chương trình mà không phải thực thi trực tiếp bằng phần cứng.

Cấp thứ 3 thường là cấp hỗn hợp. Hầu hết các lệnh trong ngôn ngữ của cấp máy này cũng có trong ngôn ngữ cấp 2 và đồng thời có thêm một tập lệnh mới, một tổ chức bộ nhớ khác và khả năng chạy 2 hay nhiều chương trình song song. Các

lệnh mới thêm vào sẽ được thực thi bằng một trình thông dịch chạy trên cấp 2, gọi là hệ điều hành. Nhiều lệnh cấp 3 được thực thi trực tiếp do vi chương trình và một số lệnh khác được thông dịch bằng hệ điều hành (do đó, cấp này là cấp hỗn hợp).

Cấp 4 thật sự là dạng tượng trưng cho một trong các ngôn ngữ. Cấp này cung cấp một phương pháp viết chương trình cho các cấp 1, 2, 3 dễ dàng hơn. Các chương trình viết bằng hợp ngữ được dịch sang các ngôn ngữ của cấp 1, 2, 3 và sau đó được thông dịch bằng các máy ảo hay thực tương ứng.

Cấp 5 bao gồm các ngôn ngữ được thiết kế cho người lập trình nhằm giải quyết một vấn đề cụ thể. Các ngôn ngữ này được gọi là cấp cao. Một số ngôn ngữ cấp cao như Basic, C, Cobol, Fortran, Lisp, Prolog, Pascal và các ngôn ngữ lập trình hướng đối tượng như C++, J++, … Các chương trình viết bằng các ngôn ngữ này thường được dịch sang cấp 3 hay 4 bằng các trình biên dịch (compiler).

IV. Bài tập củng cố kiến thức

1. Trình bày về các mức của mô hình phân cấp. Mức nào chưa có khái niệm về chương trình
2. Phân loại máy tính điện tử
3. Trình bày các thành phần cơ bản của máy tính điện tử
4. Thế nào là kiến trúc máy tính, thế nào là cấu trúc máy tính

CHƯƠNG 2 BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH

I. Mục đích : Giúp sinh viên nắm được :

- Phương pháp biểu diễn thông tin trong máy tính
- Phương pháp chuyển đổi giữa các hệ cơ số
- Phương pháp biểu diễn số âm trong các thế hệ máy tính
- Phương pháp biểu diễn số thực trong máy tính

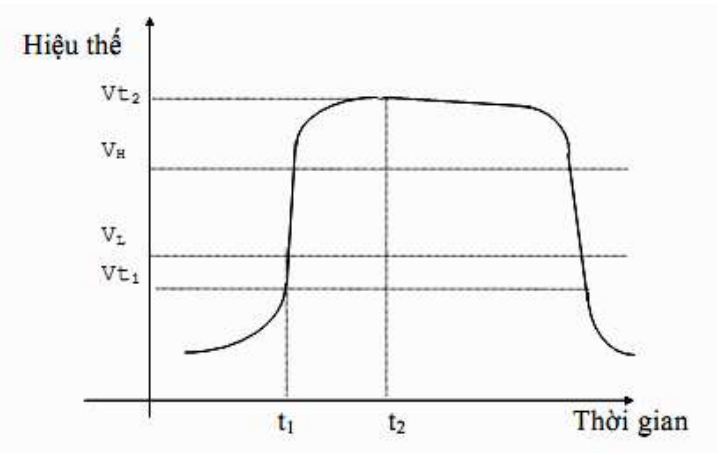
II. Yêu cầu : Sau khi học xong sinh viên cần phải làm bài tập, biết cách biểu diễn các loại thông tin trong máy tính. Với bài tập bằng tiếng anh sinh viên có thể trình bày bằng tiếng anh/tiếng việt

III. Nội dung : Chương này được giảng dạy trong 6 tiết lý thuyết và 3 tiết thảo luận làm bài tập

2.1 Thông tin và mã hoá thông tin

2.1.1 Khái niệm về thông tin

Khái niệm về thông tin gắn liền với sự hiểu biết một trạng thái cho sẵn trong nhiều trạng thái có thể có vào một thời điểm cho trước.



Hình 2-1 Trạng thái biểu diễn hiệu điện thế

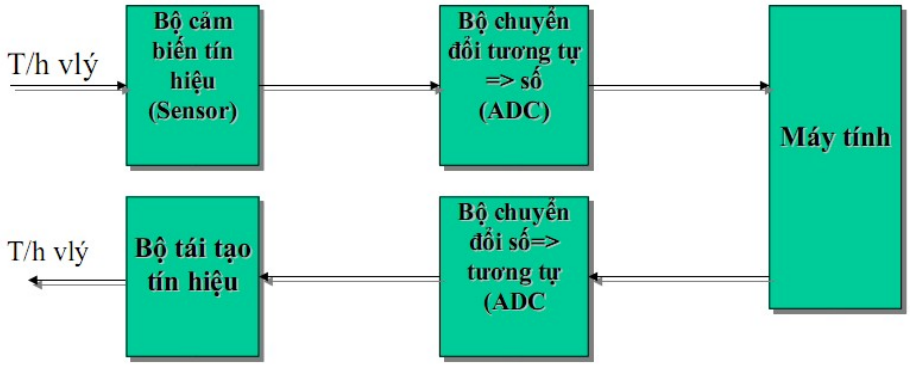
Trong hình này, chúng ta quy ước có hai trạng thái có ý nghĩa: trạng thái thấp khi hiệu điện thế thấp hơn VL và trạng thái cao khi hiệu điện thế lớn hơn VH. Để có thông tin, ta phải xác định thời điểm ta nhìn trạng thái của tín hiệu. Thí dụ, tại thời điểm t1 thì tín hiệu ở trạng thái thấp và tại thời điểm t2 thì tín hiệu ở trạng thái cao.

2.1.2 Mã hoá dữ liệu

Nguyên tắc chung về mã hoá dữ liệu

Mọi dữ liệu được đưa vào máy tính được mã hoá thành số nhị phân.

- Các loại dữ liệu:
  - Dữ liệu nhân tạo: do con người quy ước
  - Dữ liệu tự nhiên: tồn tại khách quan với con người
- Mã hoá dữ liệu nhân tạo
  - Dữ liệu số nguyên: mã hoá theo một số chuẩn đã qui ước
  - Dữ liệu số thực: mã hoá bằng số dấu chấm động
  - Dữ liệu phi số (ký tự): mã hoá theo các bộ mã ký tự hiện hành như : ASCII, Unicode,…



Hình 2-2 Qui trình biến đổi tín hiệu

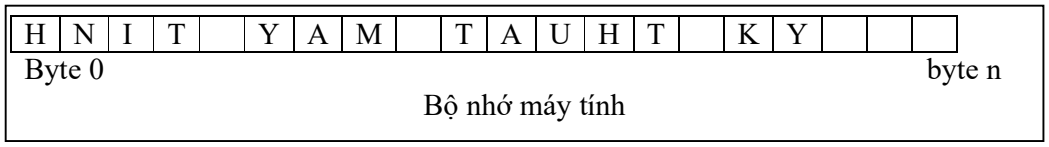
Nguyên tắc lưu trữ dữ liệu trên máy tính

Bộ nhớ chính tổ chức lưu trữ dữ liệu theo đơn vị byte. Độ dài từ dữ liệu có thể chiếm từ 1 đến 4 byte. Vì vậy cần phải biết thứ tự chúng lưu trữ trong bộ nhớ chính đối với các dữ liệu nhiều byte.

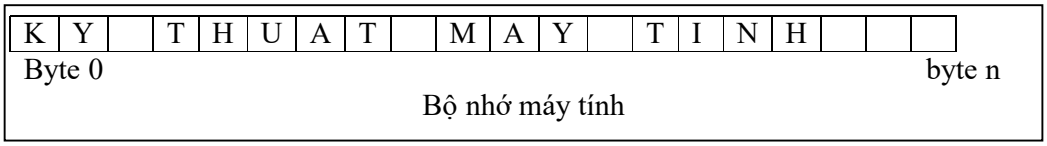
Có hai cách lưu trữ được đưa ra

- Little Endian (đầu nhỏ): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ nhỏ hơn.

Minh họa : Giả thiết cần lưu một một dòng chữ KY THUAT MAY TINH sử dụng mã ASCII 8 bit.



- Big Endian (đầu to): Byte có ý nghĩa thấp hơn được lưu trữ trong bộ nhớ ở vị trí có địa chỉ lớn hơn.



2.2 Biểu diễn số

2.2.1 Khái niệm hệ đếm

Các chữ số cơ bản của một hệ đếm là các chữ số tối thiểu để biểu diễn mọi số trong hệ đếm ấy.

- *Hệ thập phân có các chữ số cơ bản là 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.*
- *Hệ nhị phân có các chữ số cơ bản là 0, 1.*

- *Hệ thập lục phân có các chữ số cơ bản được ký hiệu là 0,.., 9, A, B, C, D, E, F.*

Nếu một số có giá trị lớn hơn các số cơ bản thì nó sẽ được biểu diễn bằng cách tổ hợp các chữ số cơ bản theo công thức sau :

$$\mathbf{X = a_n\ a_{n-1} \dots a_1\ a_0 = a_n b^n + a_{n-1} b^{n-1} + \dots + a_1 b + a_0 \quad (*)}$$

Với b là cơ số hệ đếm, a_0, a_1, a_2,...., a_n là các chữ số cơ bản

X là số ở hệ đếm cơ số b.

Ví dụ:

Hệ thập phân cho X = 123 thì $X = 1 * 10^2 + 2 * 10^1 + 3$ với b=10

Hệ nhị phân cho X = 110 thì $X = 1 * 2^2 + 1 * 2^1 + 0$ với b=2

2.2.2 Chuyển đổi giữa các hệ đếm

Chuyển đổi từ cơ số n sang cơ số 10 (với n = {2, 8, 16, ...}) theo công thức dạng tổng quát ở các phần trên.

Các chuyển đổi sô A từ cơ số 10 sang cơ số n theo thuật toán chung là :

Bước 1: Nếu A < n thì kết quả = A, trái lại đến bước 2

Bước 2: A = A chia nguyên cho n, B_i=A chia dư cho n (i=1,2,3,...)

Nếu A = 0 thì đến bước 3 trái lại làm lại bước 2 với i=i+1

Bước 3: Kết quả = $B_iB_{i-1}B_{i-2}...B_1$

Ví dụ : 118 chuyển sang cơ số 2 (n=2)

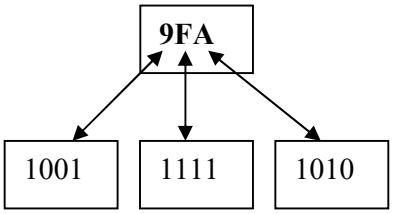
Phép tính	Số dư
118 ÷ 2 = 59	0
59 ÷ 2 = 29	1
29 ÷ 2 = 14	1
14 ÷ 2 = 7	0
7 ÷ 2 = 3	1
3 ÷ 2 = 1	1
1 ÷ 2 = 0	1

Lược trình các con số dư theo thứ tự từ dưới lên trên, cho chúng ta một số nhị phân 11101102.

- Chuyển đổi từ cơ số 2 sang cơ số 16 và ngược lại

  Mỗi một ký tự ở cơ số 16 chuyển đổi tương ứng sang 4 bit với cơ số 2 và ngược lại

VD : 9FA là số cơ số 16 chuyển sang cơ số 2 và ngược lại:



2.2.3 Biểu diễn số nguyên

a. Số nguyên không dấu

Nguyên tắc chung: Dùng n chữ số nhị phân thì biểu diễn được $\mathbf{2^n}$ số

Dải biểu diễn: $\mathbf{0 \div 2^n - 1}$

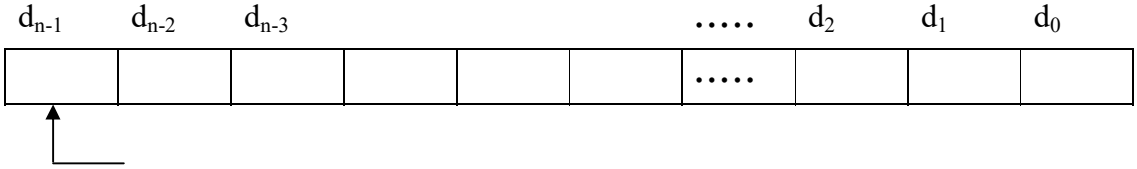
Ví dụ: n = 8 bit: dải biểu diễn: $0 \div 2^8-1$, hay từ $0 \div 255$

Cách biểu diễn: Biểu diễn ở dạng nhị phân một cách bình thường

b. Số nguyên có dấu

Có nhiều cách để biểu diễn một số n bit có dấu. Trong tất cả mọi cách thì bit cao nhất luôn tượng trưng cho dấu.

Khi đó, bit dấu có giá trị là 0 thì số nguyên dương, bit dấu có giá trị là 1 thì số nguyên âm. Tuy nhiên, cách biểu diễn dấu này không đúng trong trường hợp số được biểu diễn bằng số thừa K mà ta sẽ xét ở phần sau trong chương này (bit dấu có giá trị là 1 thì số nguyên dương, bit dấu có giá trị là 0 thì số nguyên âm).



Số nguyên có bit d_{n-1} là bit dấu và có trị số tượng trưng bởi các bit từ d_0 tới d_{n-2} .

- Biểu diễn bằng dấu và giá trị tuyệt đối

Trong cách này, bit d_{n-1} là bit dấu và các bit từ d_0 tới d_{n-2} cho giá trị tuyệt đối.

Một số n bit tương ứng với số nguyên thập phân có dấu sau :

$$\mathbf{N = (-1)^{d-1} \sum_{i=0}^{n-2} d_i 2^i}$$

Ví dụ: $+25_{10} = 00011001_2$ $-25_{10} = 10011001_2$

- Một Byte (8 bit) có thể biểu diễn các số có dấu từ -127 tới +127.
- Có hai cách biểu diễn số không là 0000 0000 (+0) và 1000 0000 (-0).

- Biểu diễn bằng số bù 1

Trong cách biểu diễn này, số âm -N được có bằng cách thay các số nhị phân d_i của số dương N bằng số bù của nó (nghĩa là nếu $d_i = 0$ thì người ta đổi nó thành 1 và ngược lại).

Ví dụ: $+25_{10} = 00011001_2$ $-25_{10} = 11100110_2$

- Một Byte cho phép biểu diễn tất cả các số có dấu từ -127 (1000 00002) đến 127 (0111 11112)
- Có hai cách biểu diễn cho 0 là 0000 0000 (+0) và 1111 1111 (-0).

- Biểu diễn bằng số bù 2

Để có số bù 2 của một số nào đó, người ta lấy số bù 1 rồi cộng thêm 1. Vậy một từ n bit (d_{n-1} ..... d_0) có trị thập phân như sau :

$$\mathbf{N = -d_{n-1}2^{n-1} + \sum_{i=0}^{n-2} d_i 2^i}$$

Một số n bit có thể biểu diễn các số có dấu từ - 2^{n-1} đến 2^{n-1}-1. Chỉ có một cách duy nhất để biểu diễn cho số không là tất cả các bit của số đó đều bằng không.

Ví dụ: $+25_{10} = 00011001_2$ $-25_{10} = 11100111_2$

- Dùng 1 Byte (8 bit) để biểu diễn một số có dấu lớn nhất là +127 và số nhỏ nhất là –128.
- Chỉ có một giá trị 0: $+0 = 00000000_2$, $-0 = 00000000_2$

- Biểu diễn bằng số thừa K

Trong cách này, số dương của một số N có được bằng cách "cộng thêm vào" số thừa K được chọn sao cho tổng của K và một số âm bất kỳ luôn luôn dương. Số âm -N của số N có được bằng cách lấy K-N (hay lấy bù hai của số vừa xác định).

Ví dụ: (số thừa K=128, số "cộng thêm vào" 128 là một số nguyên dương. Số âm là số lấy bù hai số vừa tính, bỏ qua số giữ của bit cao nhất) :

+2510 = 100110012 -2510 = 011001112

- Dùng 1 Byte (8 bit) để biểu diễn một số có dấu lớn nhất là +127 và số nhỏ nhất là –128.
- Chỉ có một giá trị 0: +0 = 100000002, -0 = 100000002

2.3 Các phép toán số học trong hệ nhị phân

Phần này chúng ta quan tâm đến các phép tính số học trên số nhị phân, với cơ bản các phép tính +,-,*,/ tương tự với cơ số 10 thông thường

2.3.1 Phép cộng nhị phân

Cộng nhị phân được thực hiện theo quy tắc ở Bảng 2.1

SỐ HẠNG	SỐ HẠNG	TỔNG	SỐ NHỚ	K ẾT QUẢ
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	10

Bảng 2-1 Quy tắc Cộng Nhị phân.

Chú ý:

Khi cộng, thực hiện từ bit có trọng số thấp đến bit có trọng số cao.

Ví dụ: Thực hiện các phép Cộng Nhị phân:

1011 + 1100 = 10111

Trong phép cộng nhị phân được chia thành 2 dạng là :

- Phép cộng số nguyên không dấu
- Phép cộng số nguyên có dấu

a. Phép cộng số nguyên không dấu

Khi cộng hai số nguyên không dấu n bit:

- Nếu không có nhớ ra khỏi bit cao nhất thì tổng luôn đúng (Cout = 0)
- Nếu có nhớ ra ngoài (Carry out) thì tổng là sai (Cout = 1), ta nói rằng phép cộng đã ***tràn nhớ***
- Tràn nhớ ra ngoài xảy ra khi tổng > 2^n - 1

b. Phép cộng số nguyên có dấu

Khi cộng hai số nguyên có dấu n bit, không quan tâm đến bit nhớ ra ngoài (Cout), kết quả nhận được là n bit:

- Nếu cộng hai số khác dấu, tổng thu được luôn luôn đúng
- Cộng hai số cùng dấu, nếu tổng cùng dấu với các số hạng thì tổng đó đúng
- Cộng hai số cùng dấu, nếu tổng ngược dấu với các số hạng thì tổng đó sai, ta nói tổng ***bị tràn*** (Overflow)

2.3.2 Phép trừ nhị phân

Phép trừ nhị phân được thực hiện theo quy tắc trình bày ở bảng 2.2

SỐ BỊ TRỪ	SỐ TRỪ	HIỆU	VAY
0	0	0	0
0	1	1	1

1	0	1	0
1	1	0	0

Bảng 2-2. Quy tắc Trừ Nhị phân :

Chú ý:

Phép tính được thực hiện từ Bit trọng số thấp đến Bit có trọng số thấp đến Bit có trọng số cao. Số vay sẽ được trừ vào Bit có trọng số cao hơn ở liền kề.

Ví dụ: Thực hiện các tính Trừ Nhị phân sau:

1011 – 0110 = 0101

Để thực hiện phép trừ được thuận lợi hơn, người ta chuyển đổi phép trừ thành phép cộng với số Bù 2 của nó.

2.3.3 Phép nhân nhị phân

Phép nhân nhị phân được thực hiện như nhân thập phân.

Ví dụ: Có phép tính:

1001 nhân với 1101 Ta thực hiện:

```
      1100     Số bị nhân (12)
    x 1011     Số nhân (11)
      1100
  +  1100
     0000
    1100
  --------
  100000100    Tích (132)
```

Trong phép cộng nhị phân được chia thành 2 dạng là :

- Phép nhân số nguyên không dấu
- Phép nhân số nguyên có dấu

a. Phép nhân số nguyên không dấu

Các tích thành phần được tính như sau:

- Nếu bit tương ứng của số nhân bằng 0 → tích thành phần bằng 0
- Nếu bit tương ứng của số nhân bằng 1 → tích thành phần bằng số bị nhân
- Tích thành phần tiếp theo được dịch trái 1 bit so với tích trước đó
- Tổng các tích thành phần là Tích cuối cùng
- Nhân hai số n bit, tích có độ dài 2n bit

b. Phép nhân số nguyên có dấu

Không thực hiện trực tiếp được trong máy tính

- Chuyển đổi thành số dương nếu cần
- Thực hiện nhân như đối với số không dấu
- Nếu hai số khác dấu → tích là số âm

2.3.4 Phép chia nhị phân

Phép chia nhị phân được thực hiện như chia thập phân.

Ví dụ: Có phép tính: 1110101 chia cho 1001

Ta thực hiện:

```
1110101  :  1001
1001        1101
-----
 01011
  1001
 -----
 001001
    1001
    ----
    0000
```

Kết quả: 1111010 : 1001 = 1101

2.4 Biểu diễn số dấu chấm động

2.4.1 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: Chuyển sang hệ Nhị phân số: 13,625

Thực hiện:

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

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

2.4.2 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

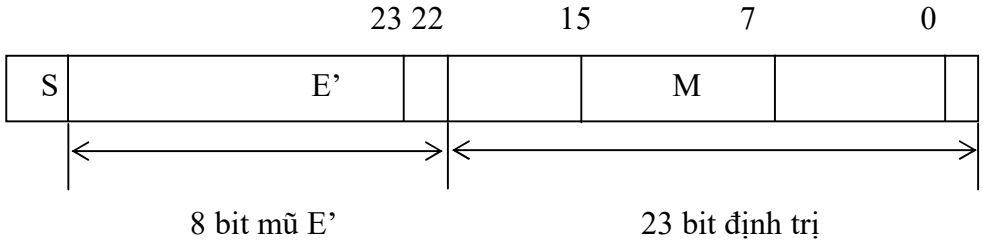
Chuẩn **IEEE 754-1985** phân định 3 dạng số dấu chấm động cơ bản (IEEE: Institute of Electrical and Electronics Engineers)

- 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)

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



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 :

Giả sử có một dãy 32 bit như sau:

1 0111 11111 100 0000 0000 0000 0000 0000

bit dấu = 1 (số âm)

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

Ví dụ :

Chuyển một số thực hệ 10 sang hệ 2 biểu diễn bằng số thực dấu phẩy động:

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

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

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

❖ **Chuẩn 80 bit:**

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

2.5 Biểu diễn ký tự

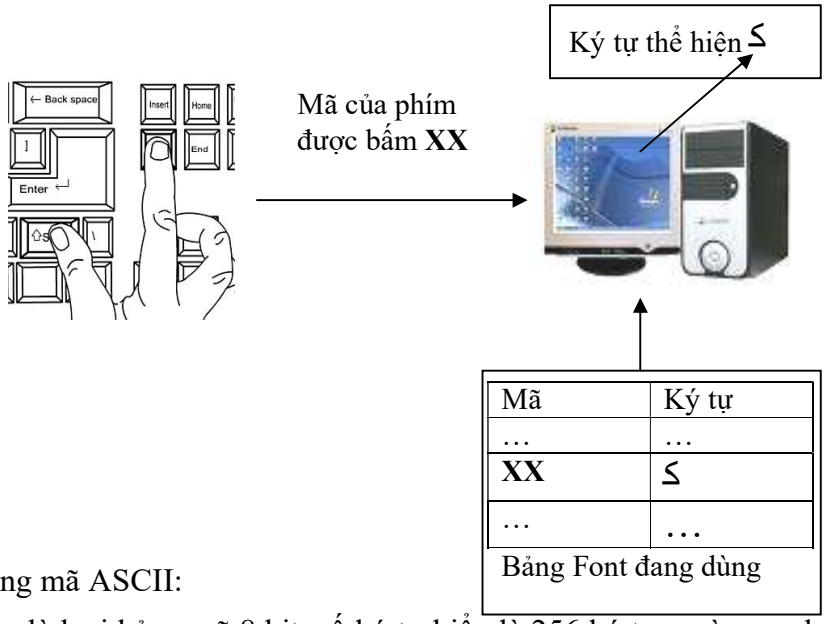
Ký tự thường được sử dụng trong các công việc hàng ngày trên máy tính, trong thực tế chúng ta thấy hầu hết các bàn phím của các máy tính đều là các ký tự la tinh, những tại sao lại có thể soạn được các ngôn ngữ khác nhau trên thế giới. Cụ thể đối với Việt Nam có các ký tự đặc biệt như "â", "ậ", … Tại sao máy tính lạ co thể hiểu và vẫn thể hiện được.

Sự thực là máy tính sử dụng một lọa bảng mã gọi là bộ Font, có thể xem bảng Font là một bảng gồm hai cột như sau:

Mã	Ký tự
0	A
1	B
2	C

…	…
66	Â
67	Ậ
…	…

Bằng cách này thì thực ra máy tính không hề hiểu các ký tự A, Â, Ậ, … Khi ta gõ vào một phím trên bàn phím thì mã số của phím đó sẽ được gửi lên máy tính, sau đó tùy thuộc môi trường ta đang sử dụng bộ Font là thì ký tự tương ứng sẽ được vẽ ra.



❖ Bảng mã ASCII:

Đây là loại bảng mã 8 bit, số ký tự biểu là 256 ký tự, ngày nay loại bảng mã này ít được sử dụng vì số ký tự tể hiện ít, chỉ đủ dùng cho một quốc gia, gây bất tiện khi chuyển văn bản

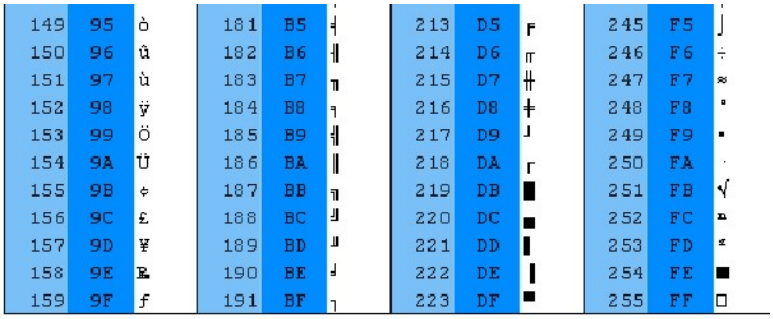
Below is the standard ASCII characters.

Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	_
48	0	64	@	80	P	96	`	112	p		

Bảng 2-3 Bảng mã ASCII chuẩn

Below is the extended ASCII characters.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	|	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠



149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	▉	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	□

Bảng 2-4 Bảng mã ASCII mở rộng

❖ Bảng mã Unicode

Do các hãng máy tính hàng đầu thế giới kết hợp thiết kế. Kích thước bộ mã này là 16 bit có thể xây dựng bộ mã toàn cầu với số ký tự có thể mã là 2^{16} ký tự với 128 ký tự đầu có mã trùng mã trong bảng mã ASCII.

Ví dụ một đoạn trên bảng mã Unicode

2013	2014	201C	201D	2018	2019	00F7	25CA	00FF	0178	2044	20AC	2039	203A	FB01	FB02
–	—	“	”	‘	’	÷	◊	ÿ	Ÿ	⁄	€	‹	›	fi	fl
2021	00B7	201A	201E	2030	00C2	00CA	00C1	00CB	00C8	00CD	00CE	00CF	00CC	00D3	00D4
‡	·	‚	„	‰	Â	Ê	Á	Ë	È	Í	Î	Ï	Ì	Ó	Ô
F8FF	00D2	00DA	00DB	00D9	0131	02C6	02DC	00AF	02D8	02D9	02DA	00B8	02DD	02DB	02C7
	Ò	Ú	Û	Ù	ı	ˆ	˜	¯	˘	˙	°	¸	˝	˛	ˇ
											00A4	00AD	0100	0101	0102
											¤	-	Ā	ā	Ă
0103	0104	0105	0106	0107	0108	0109	010A	010B	010C	010D	010E	010F	0110	0111	0112
ă	Ą	ą	Ć	ć	Ĉ	ĉ	Ċ	ċ	Č	č	Ď	ď	Đ	đ	Ē
0113	0114	0115	0116	0117	0118	0119	011A	011B	011C	011D	011E	011F	0120	0121	0122
ē	Ĕ	ĕ	Ė	ė	Ę	ę	Ě	ě	Ĝ	ĝ	Ğ	ğ	Ġ	ġ	Ģ
0123	0124	0125	0126	0127	0128	0129	012A	012B	012C	012D	012E	012F	0130	0132	0133
ģ	Ĥ	ĥ	Ħ	ħ	Ĩ	ĩ	Ī	ī	Ĭ	ĭ	Į	į	İ	IJ	ij
0134	0135	0136	0137	0138	0139	013A	013B	013C	013D	013E	013F	0140	0143	0144	0145
Ĵ	ĵ	Ķ	ķ	ĸ	Ĺ	Í	Ļ	ļ	Ľ	ľ	Ŀ	ŀ	Ń	ń	Ņ
0146	0147	0148	0149	014A	014B	014C	014D	014E	014F	0150	0151	0154	0155	0156	0157
ņ	Ň	ň	ʼn	Ŋ	ŋ	Ō	ō	Ŏ	ŏ	Ő	ő	Ŕ	ŕ	Ŗ	ŗ
0158	0159	015A	015B	015C	015D	015E	015F	0162	0163	0164	0165	0166	0167	0168	0169
Ř	ř	Ś	ś	Ŝ	ŝ	Ş	ş	Ţ	ţ	Ť	ť	Ŧ	ŧ	Ũ	ũ
016A	016B	016C	016D	016E	016F	0170	0171	0172	0173	0174	0175	0176	0177	0179	017A
Ū	ū	Ŭ	ŭ	Ů	ů	Ű	ű	Ų	ų	Ŵ	ŵ	Ŷ	ŷ	Ź	ź
017B	017C	017F	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	018A	018B	018C
Ż	ż	ſ	ƀ	Ɓ	Ƃ	ƃ	Ƅ	ƅ	Ɔ	Ƈ	ƈ	Đ	Ɗ	Ƌ	ƌ
018D	018E	018F	0190	0191	0193	0194	0195	0196	0197	0198	0199	019A	019B	019C	019D
ƍ	Ǝ	Ə	Ɛ	Ƒ	Ɠ	Ɣ	ƕ	Ɩ	Ɨ	Ƙ	ƙ	ƚ	ƛ	Ɯ	Ɲ

Bảng 2-5 Bảng mã Unicode

IV. Bài tập củng cố kiến thức

2.1 Đổi số $(01011010)_2$ ra số thập phân.

2.2 Đổi số $(-31)_{10}$ ra số nhị phân

2.3 Đổi số $(-123)_{10}$ ra số nhị phân bằng phương pháp mã bù hai.

2.4 Đổi số (- 34) ra số nhị phân bằng phương pháp "Mã dấu và độ lớn "

2.5 Đổi số (- 55) ra số nhị phân bằng phương pháp "Mã bù một "

2.6 Cho các dãy số nhị phân 8 bit sau đây, dùng phương pháp "Mã bù hai " hãy xác định giá trị của chúng:

- 0010 1101
- 1010 1011

2.7 Cho biết cách biểu diễn số thực theo dạng chuẩn IEEE 754/85 chính xác đơn 32bit. Áp dụng cho số thực -34.66

2.8 Calculate the binary representation of 2303?

2.9 How many Bytes do you need to represent 1023?

2.10. Which decimal number is coded with 10101010

2.11 Represent the following decimal numbers in both binary sign/magnitude and twos complement using 16 bits: 512; - 29

2.12: Represent the following twos complement values in decimal: 1101011; 0101101

2.13 Assume numbers are represented in 8-bit twos complement representtation. Show the calculation of the following:

a. 6+13 b. -6+13 c. 6-13 d. -6-13

2.14. Express the following numbers in IEEE 32bit floating – format :

a. -5 b. -6 c. -1.5 d. 384 e. 1/16 f. -1/32

2.15 Let 5BCA0000 be a floating point number in IBM format, expressed in hexadecimal. What is the decimal value of the number

CHƯƠNG 3 MỨC LOGIC SỐ

I. Mục đích

- Giúp sinh viên nắm được cơ bản về các cổng, các mạch cơ bản
- Giúp sinh viên nắm được mức logic số, mức thấp nhất trong mô hình phân cấp máy tính
- Biết cách xây dựng một số mạch cơ bản như mạch cộng đủ, mạch cộng nửa … trên modul thí nghiệm
- Biết cách thiết kế modul nhớ nhằm tăng dung lượng của bộ nhớ

II. Yêu cầu

- Sinh viên cần làm bài tập trong sách bài tập
- Thực hiện modul thí nghiệm
- Xây dựng modul nhớ

III. Nội dung: Chương 3 được học trong 6 tiết lý thuyết và 1 tiết thực hành, 3 tiết bài tập

3.1 Giới thiệu về cổng và đại số logic

3.1.1 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:

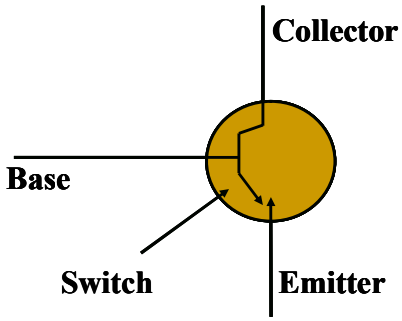
- Mạch logic bởi nó thực hiện các phép tính đại số logic.
- Phần tử ra quyết định (decision making element)

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).

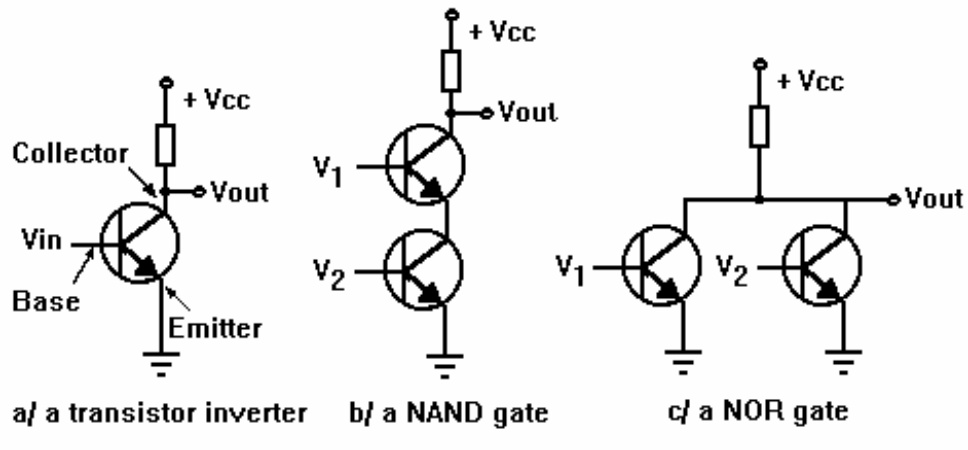


Hình 3-1 Cấu tạo Transistor

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 3-2 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.

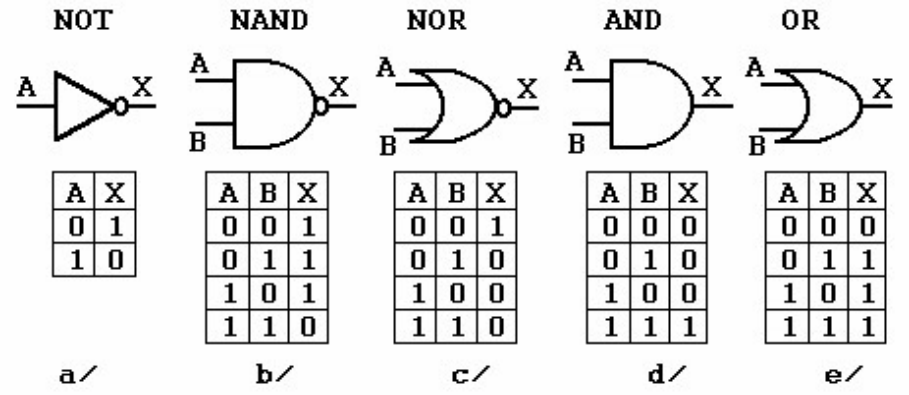
3.1.2 Đại số logic

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 3-03).



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

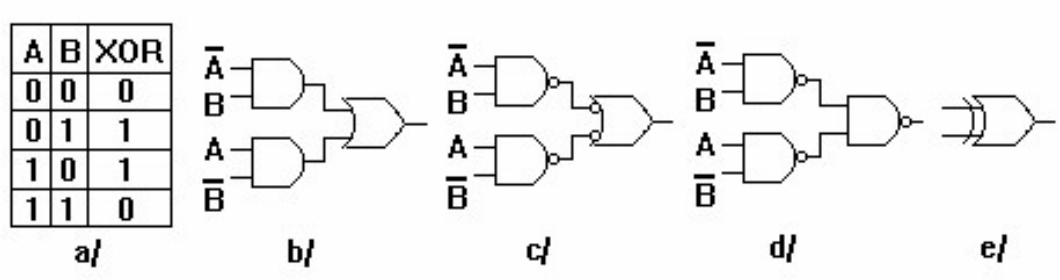
- Mô tả hàm logic bằng số nhị phân:
  - Liệt kê các hàng của bảng chân lý theo thứ tự số học (cơ số 2) của tổ hợp biến vào, nghĩa là cho 2 biến theo thứ tự 00, 01, 10 và 11.
  - Đọc kết quả (giá trị của hàm) từ bảng chân lý theo cột, từ trên xuống dưới. Như vậy NAND sẽ là 1110, NOR là 1000, AND là 0001 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.

3.1.3 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.



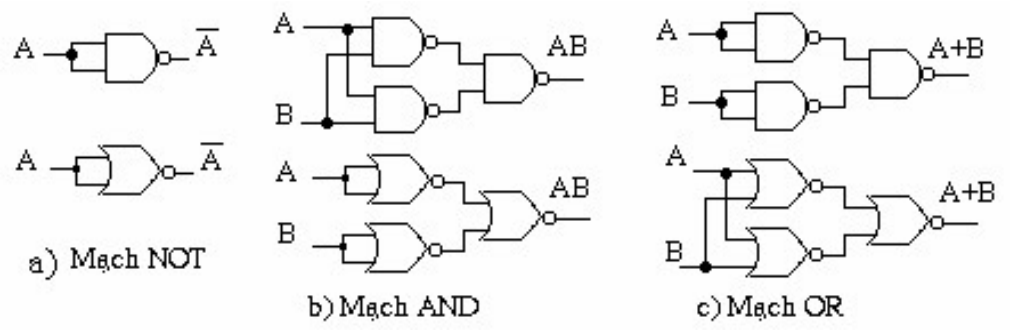
Hình 3-4 Xây dựng mạch điện bằng hàm logic

- 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.

3.1.4 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: sử dụng một loại cổng NAND hoặc NOR.



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

3.2 Các mạch logic số cơ bản

3.2.1 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.
- Thường 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.

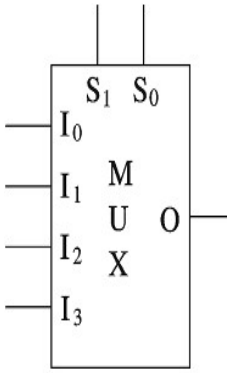
3.2.2 Mạch tổ hợp

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.

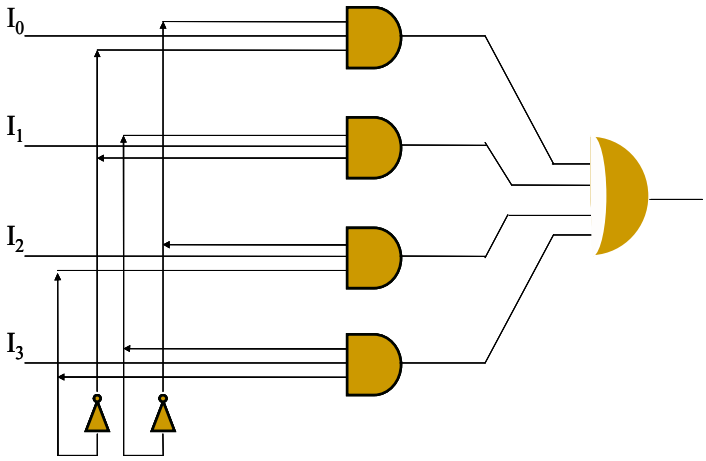
Một số mạch tổ hợp mạch tổ hợp điển hình như: mạch dồn kênh, mạch phân kênh, mạch mã hoá, mạch giải mã, mạch so sánh, v.v..

a. Mạch dồn kênh (**Multiplexer)**

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



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



Hình 3-5 Mạch dồn kênh cho 4 đường dữ liệu vào

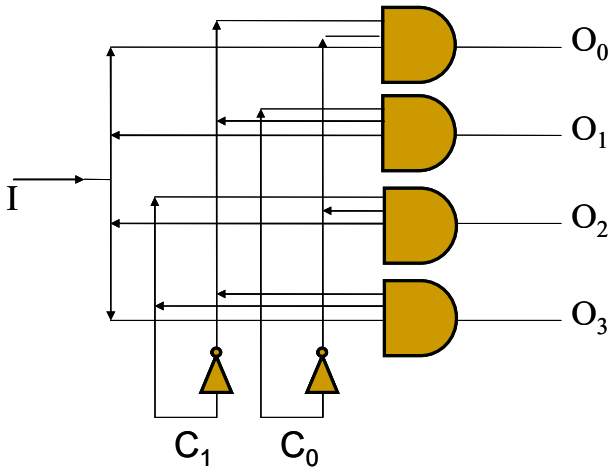
Nhiều ứng dụng logic số cần đến các mạch có nhiều đầu vào và nhiều đầu ra trong đó các giá trị ra hoàn toàn được xác định bởi các giá trị đầu vào ở thời điểm đang xét. Những mạch như vậy được gọi là Mạch tổ hợp *(Combinational Circuit),* các mạch này thường được thể hiện bởi bảng chân lý.

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..

b. Mạch phân kênh (**Demultiplexe**)

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

C_1	C_0	
0	0	O_0=I
0	1	O_1=I
1	0	O_2=I
1	1	O_3=I



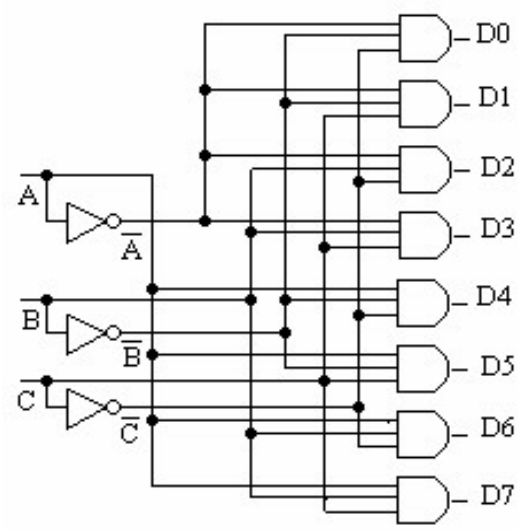
Hình 3-6 Mạch phân kênh 1 đầu vào 4 đầu ra

c. Mạch giải mã (decoder)

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



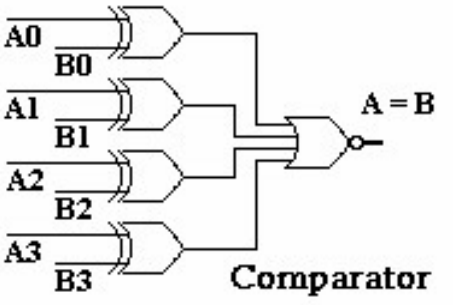
Hình 3-7 Mạch giải mã 3 đầu vào

d. 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.



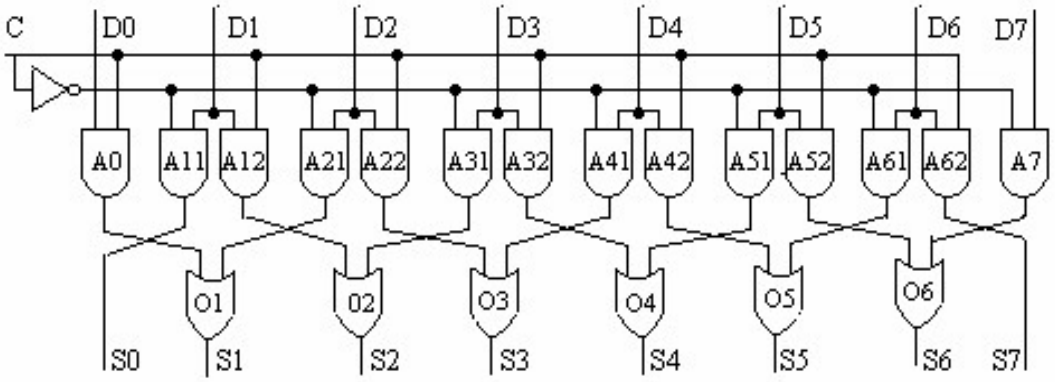
3.2.3 Các mạch số học

a. 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 3-8 Bộ dịch 8bit

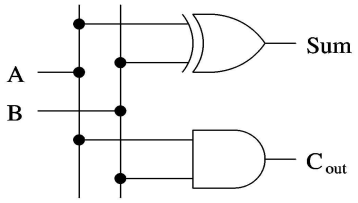
C xác định hướng dịch chuyển của các bit:

- 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.

b. Bộ cộng

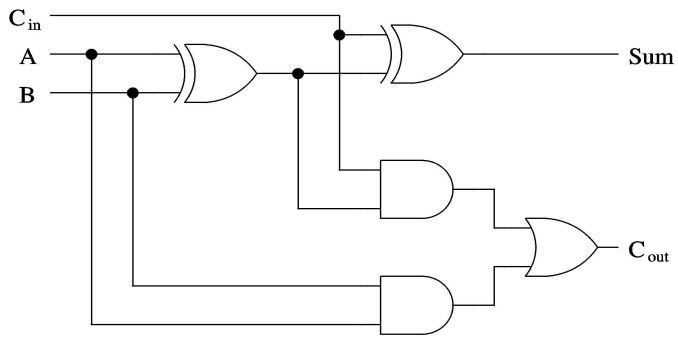
Bộ cộng là một phần rất căn bản của mọi CPU. Hình 3-9 a là mạch tính tổng 2 bit - Haft_Adder. Bộ cộng này không có lối vào cho số nhớ Carry-in cho nên không dùng để cộng các bit bậc cao hơn 0. Hình 3-9 b là mạch tính tổng 2 bit vào và bit nhớ C-in, trả lại tổng - sum và số nhớ C-out**.** Bộ cộng này có lối vào C-in, nó được xây dựng từ 2 bộ Half-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



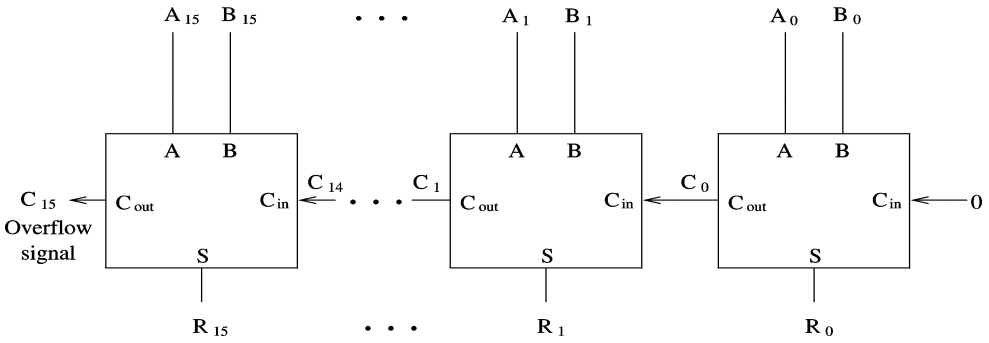
Hình 3-9 Bộ cộng

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

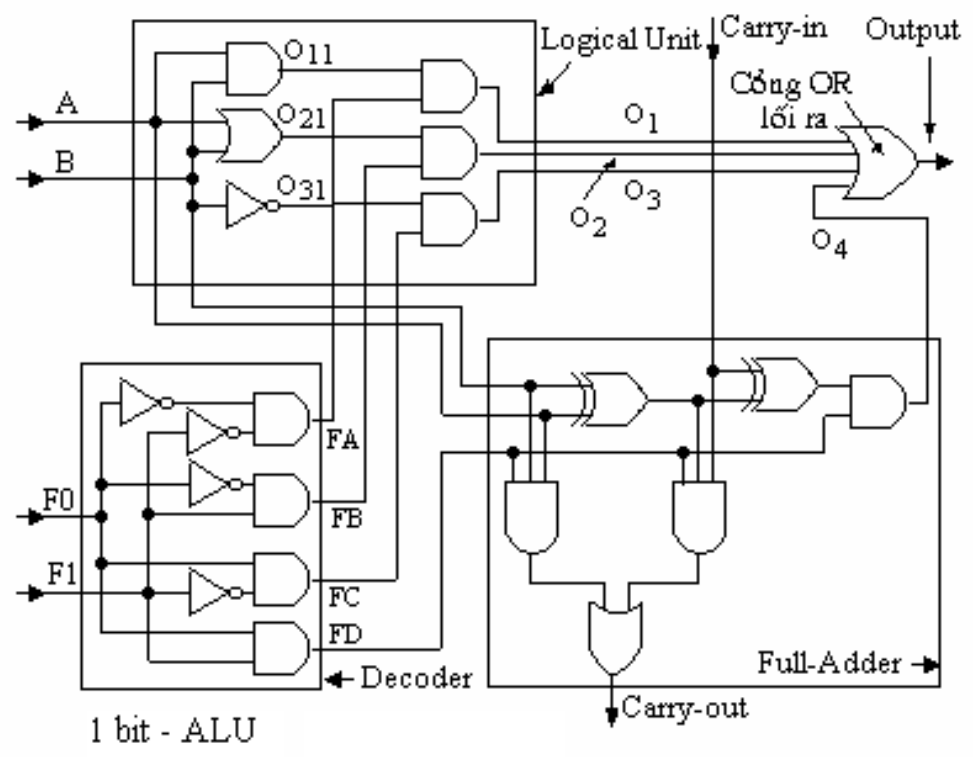


Hình 3-10 Bộ 16-bit ripple-carry adder

c. 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 3-11 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



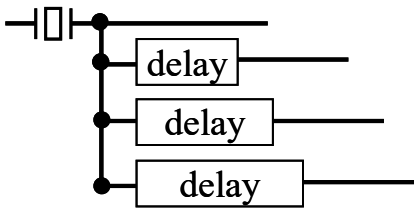
Hình 3-11 - Cấu tạo một ALU

d. 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.

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 Tổ chức bộ nhớ

3.3.1 Khái quát

Là một trong các thành phần quan trọng nhất của máy tính điện tử, được dùng để lưu trữ các lệnh và dữ liệu. Bộ nhớ được xây dựng từ các phần tử nhớ cơ bản:

- Mỗi phần tử nhớ cơ bản có thể nhớ được 1 bit thông tin
- Đó là các mạch điện có hai trạng thái cân bằng ổn định (mạch lưỡng bền, flip-flop), flip-flop sẽ nằm mãi ở một trong hai trạng thái cân bằng nếu không có tín hiệu điện phù hợp kích thích vào làm cho nó phải thay đổi trạng thái.

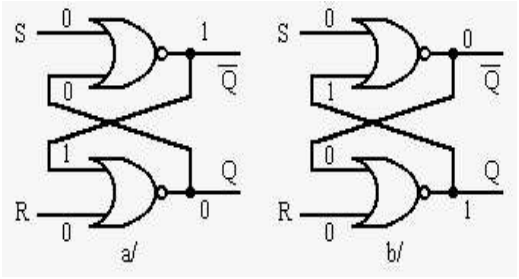
3.3.2 Phần tử nhớ 1 bit

Thanh ghi chốt (Latch) là dạng đơn giản của flip-flop, được xây dựng từ 2 cổng NAND hoặc 2 cổng NOR. Sự thay đổi trạng thái của thanh ghi chốt có thể xảy ra trong thời gian kéo dài của xung đồng hồ chứ không phải trong thời gian sườn xung đồng hồ, người ta gọi đó là sự chuyển mạch theo mức. Các thanh ghi chốt 1 bit có thể được sử dụng làm các phần tử nhớ cơ bản xây dựng nên bộ nhớ của máy tính.

a. 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 $\overline{Q}$



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.

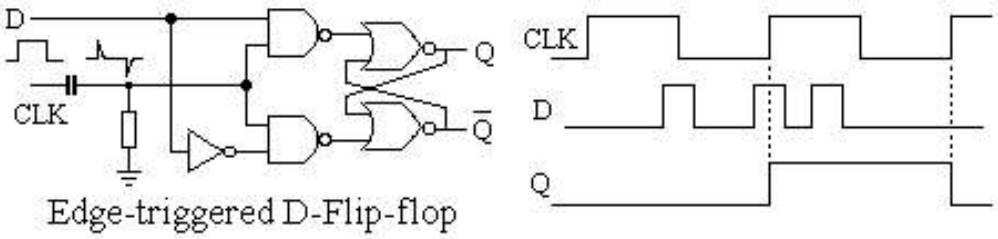
- R=S=0: Q sẽ không thay đổi. (trạng thái này tồn tại trong phần lớn thời gian hoạt động của thanh ghi)
- S=1, R=0, Q=1 không phụ thuộc vào trạng thái trước đó.
- R=1, S=0, Q=0 không phụ thuộc vào trạng thái trước đó.
- Tổ hợp R=S=1 bị cấm.

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.

b. Mạch Flip-Flop

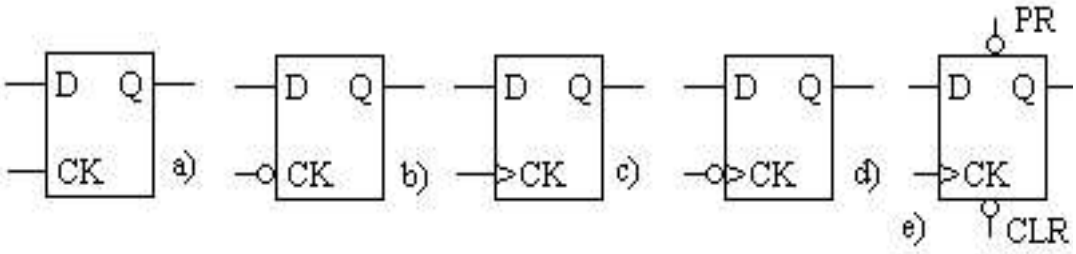
Trong nhiều mạch điện thường cần phải lấy mẫu (sample) giá trị tín hiệu trên một đường dây nào đó tại một thời điểm cụ thể và ghi nhớ giá trị đó.

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



Edge-triggered D-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



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:

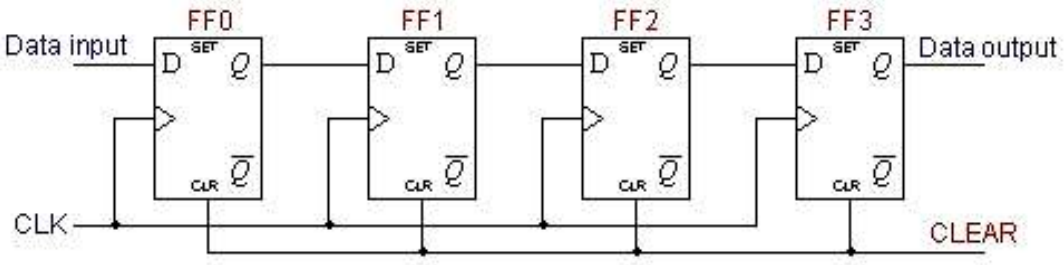
SISO (Serial In Serial Output - vào nối tiếp ra nối tiếp)

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

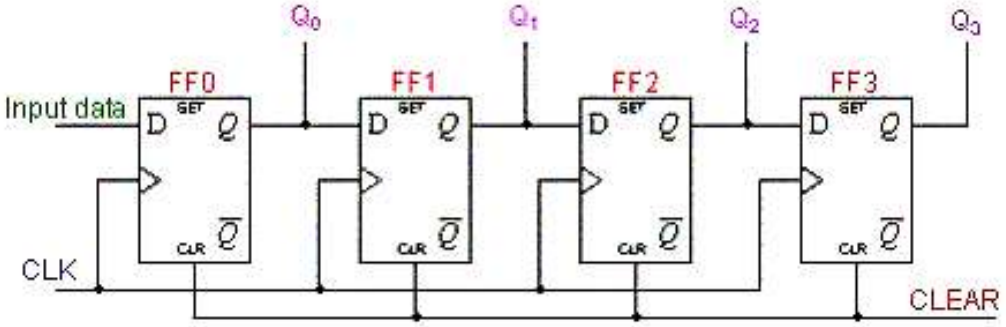
PISO (Parallel In Serial Output - vào song song ra nối tiếp)

PIPO (Parallel In Parallel Output - vào song song ra song song)

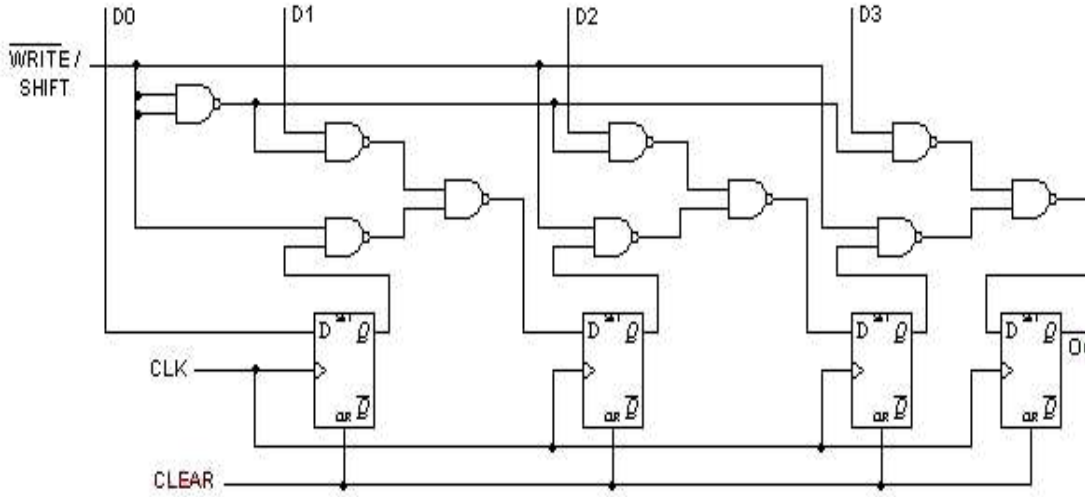
Cấu trúc các dạng kết nối được mô tả trong hình (a), (b), (c), (d) dưới đây:



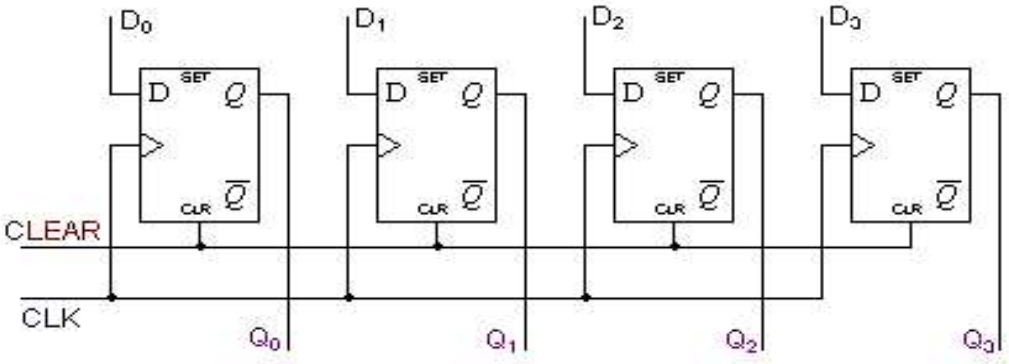
(a) Kết nối dạng SISO: Dữ liệu cần dịch chuyển được đưa vào ngõ D của mạch FF đầu tiên (FF0). Ở mỗi xung kích lên của đồng hồ ck, sẽ có 1 bit được dịch chuyển từ trái sang phải, nối tiếp từ tầng này qua tầng khác và đưa ra ở đầu Q của mạch sau cùng (FF3)



(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 $\overline{\mathbf{Q}}$), 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 3-12 Có một số dạng kết nối thanh ghi dịch

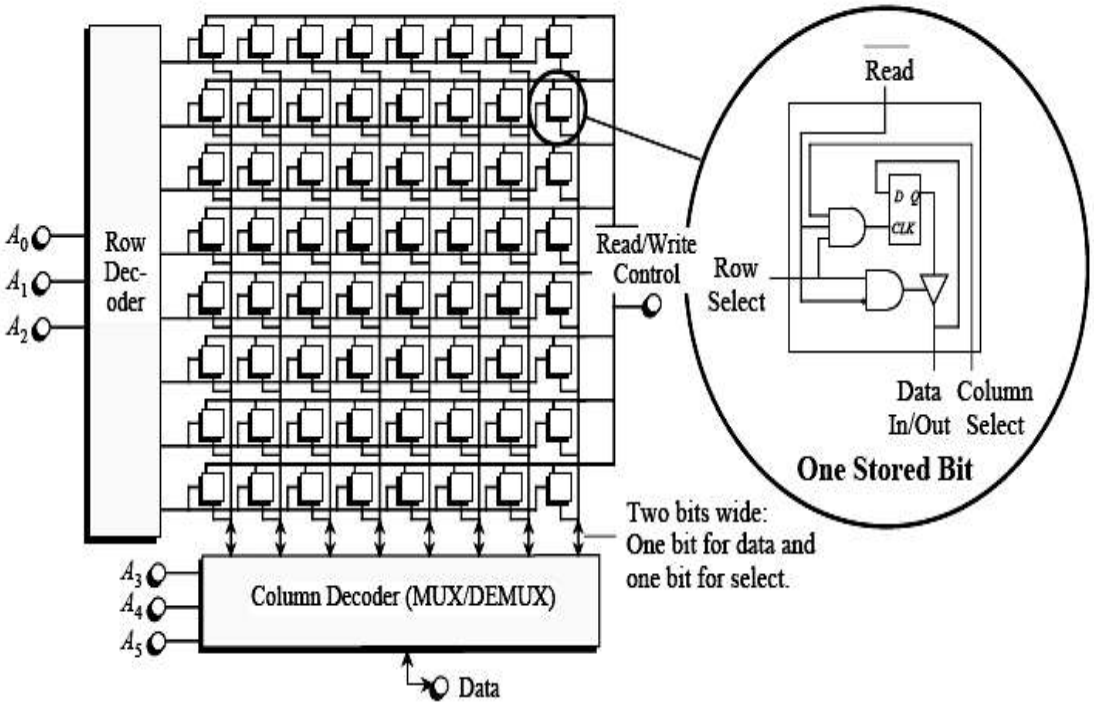
3.3.3 Tổ chức bộ nhớ

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 3-13 Sơ đồ mô tả cấu trúc một vi mạch 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.
- Các đường dữ liệu: D0 ÷ Dm – 1, nhơ vậy độ dài ngăn nhớ là m bit.
- Dung lượng chip nhớ: 2n x m bit
- 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

b. 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.
- **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

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:

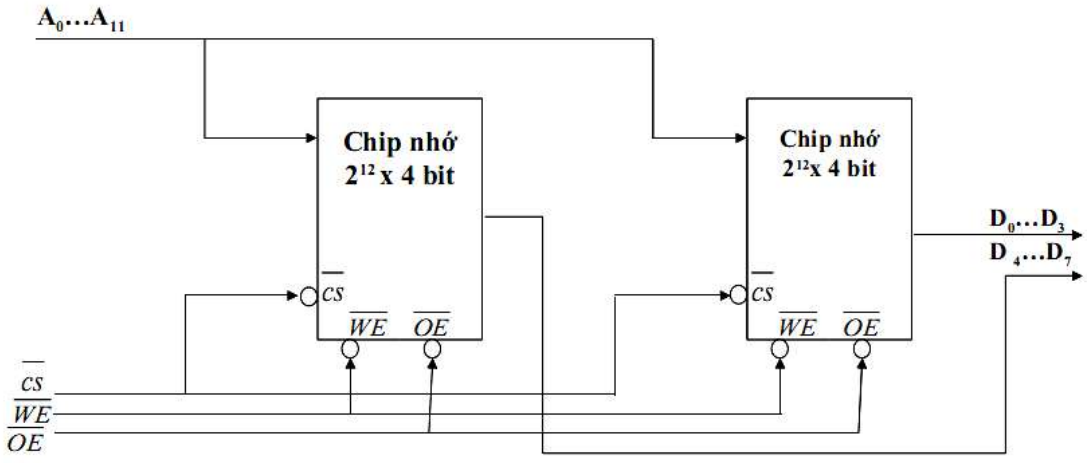
Cho các chip nhớ SDRAM dung lượng 4K x 4 bit. Hãy thiết kế module nhớ có kích thước 4K x 8 bit

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:



- **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 :

Cho các chip nhớ SDRAM dung lượng 4K x 8 bit. Hãy thiết kế module nhớ có kích thước 8K x 8 bit.

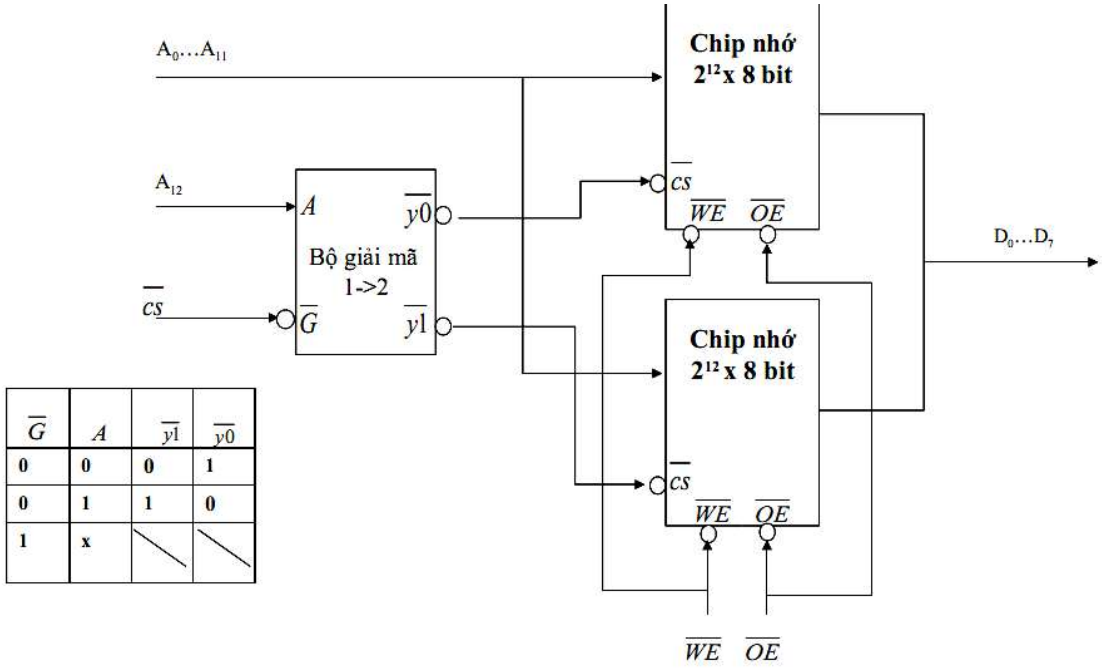
Giải:

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:

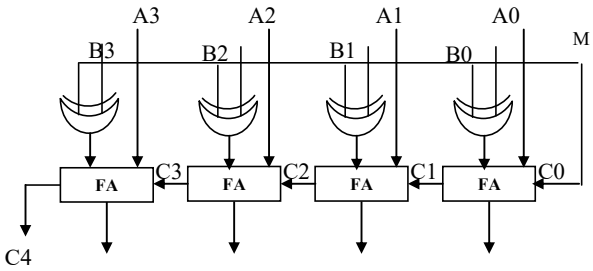


$\overline{G}$	A	$\overline{y1}$	$\overline{y0}$
0	0	0	1
0	1	1	0
1	x		

- **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

IV. Bài tập củng cố kiến thức:

3.1 Cho mạch cộng trừ như hình sau, với các giá trị đầu vào M = 0, A = 0111, B = 0101



Hãy xác định các giá trị đầu ra S0, S1, S2, S3 và C4

3.2 Có bao nhiêu khối kích thước 16 word trong một bộ nhớ có không gian nhớ là 256MB? Hãy thể hiện tổ chức logic đầy đủ của bộ nhớ bao gồm hai phần: phần xác định vị trí của khối và địa chỉ của Word trong khối

3.3 Để thiết kế một bộ nhớ có dung lượng 2KB, cần:

a. Sử dụng bao nhiêu chip nhớ RAM có dung lượng là 128bytex8bit.

b. Tính số bus địa chỉ cần thiết truy cập đến bộ nhớ 2KB và số đường dùng chung cho các chip.

c. Bộ giải cần bao nhiêu đường vào/ra?

Cho trước: sử dụng 8 đường dữ liệu

3.4 * Vẽ mô hình kết nối bộ nhớ với CPU, biết rằng hệ thống bộ nhớ gồm có 2KB các chip nhớ RAM có dung lượng 256x8 bit và 4KB các chip nhớ ROM có dung lượng là 1024x8 bit. (8 đường dữ liệu)

3.5* Cho 4 chip nhớ RAM có kích thước 128x8 bit và 2 chip nhớ ROM có kích thước 512x8 bit. Hãy cho biết từ các chíp nhớ trên có thể xây dựng được một hệ thống bộ nhớ có dung lượng bằng bao nhiêu. Vẽ sơ đồ kết nối giữa CPU và hệ thống bộ nhớ với kích thước của đường dữ liệu bằng 8bit.

3.6 * Vẽ mô hình kết nối bộ nhớ với CPU, biết rằng hệ thống bộ nhớ gồm có 16KB các chip nhớ RAM có dung lượng 256x8 bit và 4KB các chip nhớ ROM có dung lượng là 128x8 bit. (8 đường dữ liệu)

CHƯƠNG 4 MỨC VI CHƯƠNG TRÌNH

I. Mục đích

- Giúp sinh viên nắm được các vấn đề cơ bản về mức vi chương trình
- Giúp sinh viên nắm được các vấn đề về đường dữ liệu

II. Yêu cầu

- Hiểu lý thuyết
- Biết cách thiết kế một CPU trên phần mềm mô phỏng

III. Nội dung: Nội dung chương này được trình bày trên 6 tiết lý thuyết và 2 tiết thực hành

4.1 Chức năng và hoạt động của bộ xử lý

Trong hệ thống máy tính, Đơn vị xử lý trung tâm (CPU – Central Proccesing Unit)đóng vai trò quan trọng vì đây là đơn vị điều khiển và xử lý dữ liệu. CPU hoạt động theo chương trình nằm trong bộ nhớ chính.

Cấu trúc cơ bản của CPU gồm 4 thành phần chính:

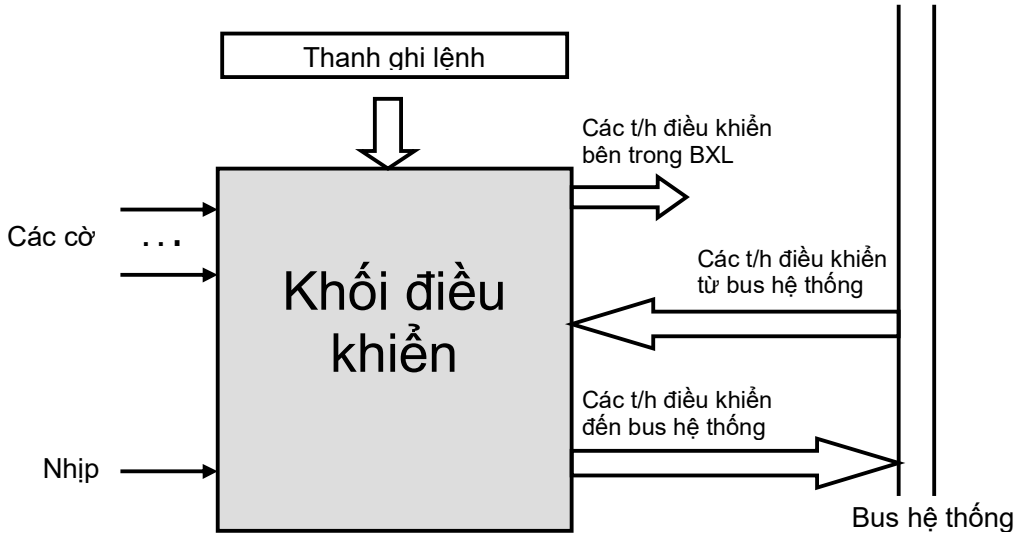
- Đơn vị điều khiển
- Đơn vị xử lý toán học và logic
- Thanh ghi
- Hệ thống BUS

4.1.1 Đơn vị điều khiển (Control Unit - CU)

Điều khiển hoạt động của máy tính theo chương trình đã định sẵn. Qui trình hoạt động diễn ra như sau

- Điều khiển việc nhận lệnh tiếp theo từ bộ nhớ, đưa vào thanh ghi lệnh.
- Tăng nội dung của PC để trỏ sang lệnh tiếp theo
- Giải mã lệnh nằm trong thanh ghi lệnh để xác định thao tác mà lệnh yêu cầu
- Phát ra các tín hiệu điều khiển thực hiện lệnh đó
- Nhận tín hiệu yêu cầu từ bên ngoài, xử lý các tín hiệu đó.

Mô hình kết nối của CU



Hình 4-1 Mô hình kết nối CU

Các tín hiệu đến khối điều khiển

- Clock: tín hiệu xung nhịp từ mạch tạo dao động.
- Mã lệnh từ thanh ghi lệnh đưa đến CU giải mã

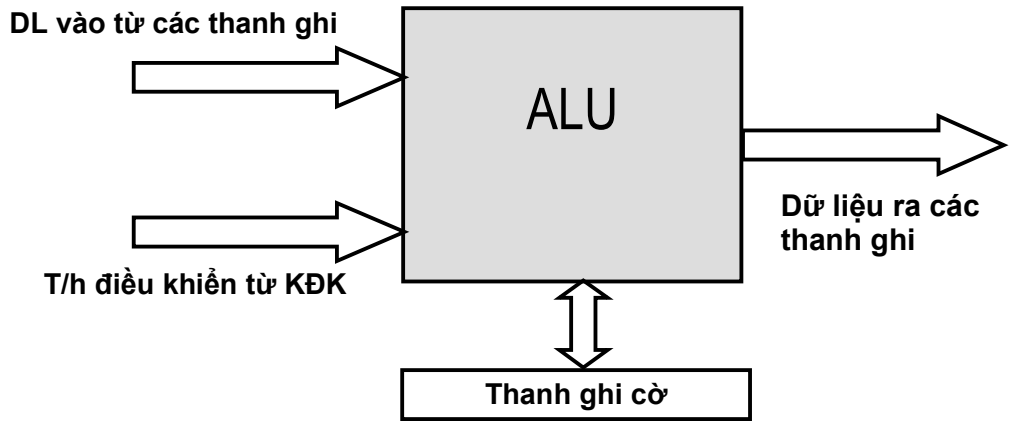
- Các trạng thái cờ đưa đến cho biết trạng thái của CPU cũng như trạng thái thực hiện các phép toán trong ALU.
- Các tín hiệu điều khiển từ BUS điều khiển.
- Các tín hiệu điều khiển bên trong CPU: điều khiển thanh ghi, ALU.
- Các tín hiệu điều khiển bên ngoài CPU đó là Bộ nhớ hay cổng vào ra

4.1.2 Đơn vị xử lý toán học và logic (Arithmetic and logical Unit - ALU)

Thực hiện phép toán số học và các phép toán logic trên các dữ liệu cụ thể. Đó là các phép toán:

- Số học: cộng, trừ, nhân, chia, tăng, giảm, đảo,..
- Logic: AND, OR,XOR, NOT, dịch bit,…

Mô hình kết nối của ALU



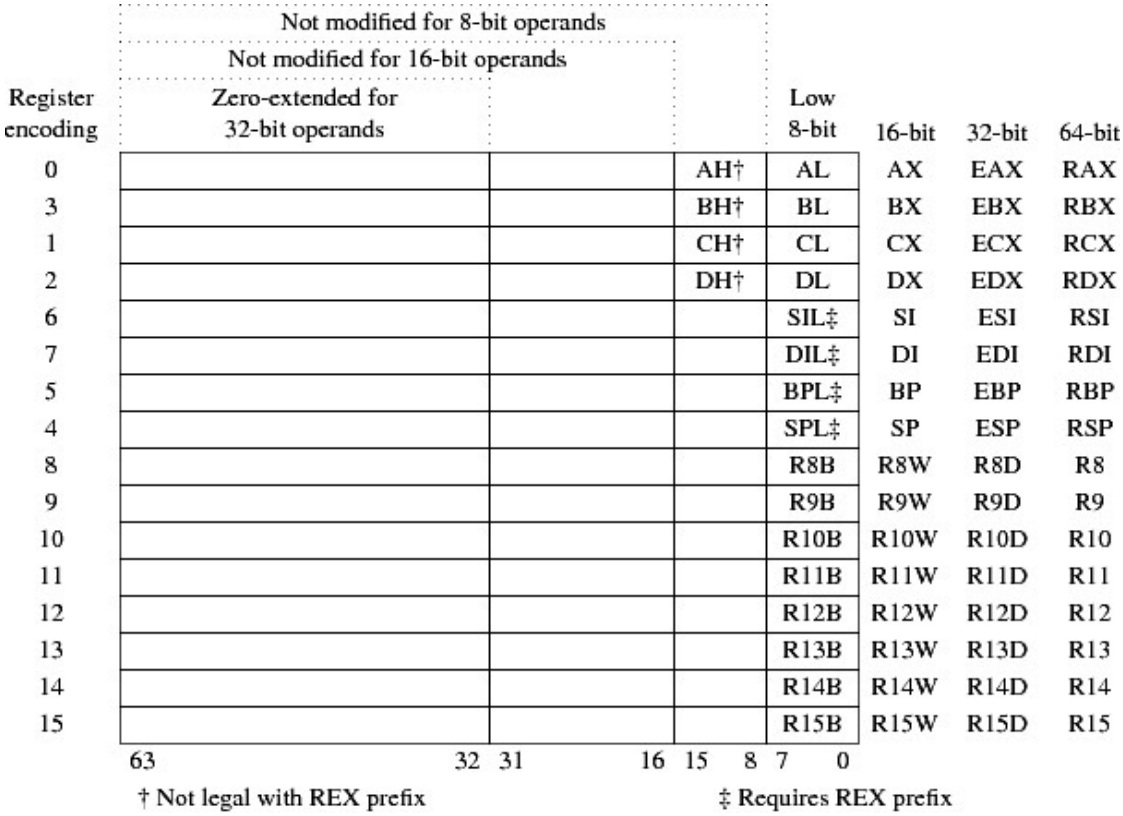
Hình 4-2 Mô hình kết nối ALU

4.1.3 Thanh ghi

Chức năng

- Thực chất là vùng nhớ được CPU nhận biết qua tên thanh ghi và có tốc độ truy xuất cực nhanh.
- Chứa thông tin tạm thời phục vụ cho hoạt động ở thời điểm hiện tại của CPU
- Số lượng thanh ghi tuỳ thuộc vào bộ vi xử lý cụ thể -> tăng hiệu năng CPU
- Thanh ghi chia 2 loại: Loại lập trình được và loại không lập trình được

Dưới đây là tập các thanh ghi đa năng ở các thế hệ vi xử lý từ 8bit – 64bit:



Register encoding	Not modified for 8-bit operands / Not modified for 16-bit operands / Zero-extended for 32-bit operands			Low 8-bit	16-bit	32-bit	64-bit
0			AH†	AL	AX	EAX	RAX
3			BH†	BL	BX	EBX	RBX
1			CH†	CL	CX	ECX	RCX
2			DH†	DL	DX	EDX	RDX
6				SIL‡	SI	ESI	RSI
7				DIL‡	DI	EDI	RDI
5				BPL‡	BP	EBP	RBP
4				SPL‡	SP	ESP	RSP
8				R8B	R8W	R8D	R8
9				R9B	R9W	R9D	R9
10				R10B	R10W	R10D	R10
11				R11B	R11W	R11D	R11
12				R12B	R12W	R12D	R12
13				R13B	R13W	R13D	R13
14				R14B	R14W	R14D	R14
15				R15B	R15W	R15D	R15

63 32 31 16 15 8 7 0

† Not legal with REX prefix ‡ Requires REX prefix

❖ Phân loại thanh ghi theo chức năng

- Thanh ghi địa chỉ: Thanh ghi được sử dụng để quản lý địa chỉ của ngăn nhớ hay cổng vào ra.
- Thanh ghi dữ liệu: Thanh ghi dùng để lưu trữ dữ liệu tạm thời
- Thanh ghi đa năng: Thanh ghi có thể chứa dữ liệu hoặc địa chỉ đều được.
- Thanh ghi điều khiển/trạng thái: Thanh ghi chứa thông tin về trạng thái CPU.
- Thanh ghi lệnh: thanh ghi chứa lệnh đang được thực hiện

4.1.4 Hệ thống BUS

a. Định nghĩa

Bus là đường truyền dữ liệu điện giữa các thiết bị khác nhau trong một hệ thống máy tính

Bus có thể là một hoặc hai chiều. Bus hai chiều có thể truyền dữ liệu đi theo cả hai chiều, nhưng không đồng thời.

- Bus một chiều thường được sử dụng để nối hai thanh ghi, trong đó một thanh ghi luôn luôn là nguồn còn thanh ghi kia luôn luôn là đích.

- Bus hai chiều hay được sử dụng khi một thanh ghi bất kỳ trong tập hợp các thanh ghi có thể là nguồn và một thanh ghi bất kỳ khác là đích.

Bus 3 trạng thái:

- Là bus mà các thiết bị nối với nó có khả năng nối và tách chính nó ra khỏi bus về mặt điện.
- Thời gian nối/tách: ns
- Tên gọi 3 trạng thái: mỗi đường dây có thể có giá trị 0 hay 1 hoặc là bị tách ra.
- Thường được sử dụng khi cần nối nhiều thiết bị vào bus, mỗi thiết bị này đều có thể đổ thông tin lên bus.

b. Phân loại BUS:

Người ta thường phân loại bus theo 3 cách sau:

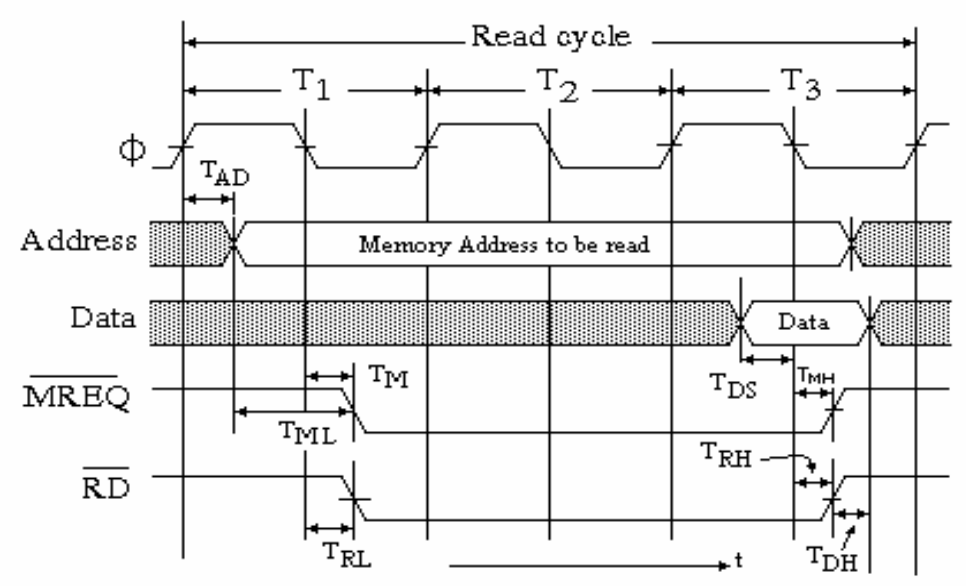
Theo tổ chức phần cứng:

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

Theo giao thức truyền thông:

- Bus đồng bộ:

Mọi hoạt động bus xảy ra trong một số nguyên lần chu kỳ đồng hồ - chu kỳ bus. Trên Hình là giản đồ thời gian của một bus đồng bộ với tần số đồng hồ là 4 MHz (T=250ns)



Việc đọc 1 byte từ bộ nhớ chiếm 3 chu kỳ bus (T1, T2 và T3).

Vì tất cả các tín hiệu điện thay đổi mức không phải là tức thời, nên trên hình vẽ có các sườn xung (giả sử các sườn xung kéo dài 10ns).

Wait state: Nếu bộ nhớ không có khả năng đáp ứng đủ nhanh, nó cần phát tín hiệu đòi chờ (wait state) trước sườn xuống của T2, yêu cầu đưa thêm vào một số chu kỳ bus; khi bộ nhớ đã đưa ra tín hiệu ổn định, nó sẽ đảo tín hiệu WAIT.

Block Transfer: Khi một thao tác đọc khối bắt đầu, master báo cho slave biết có bao nhiêu byte cần được truyền đi, thí dụ truyền đi con số này trong chu kỳ T1, sau đó đáng lẽ truyền đi một byte, slave đưa ra trong mỗi chu kỳ một byte cho tới khi đủ số byte được thông báo.

Bus skew: Có thể rút ngắn chu kỳ để làm cho bus truyền dữ liệu nhanh hơn. Trong thí dụ trên mỗi byte được truyền đi trong 750ns, vậy bus có dải thông là 1.33 Mbytes/sec. Nếu đồng hồ có tần số 8MHz, thời gian một chu kỳ chỉ còn một nửa, giải thông sẽ là 2.67MBytes/sec.Tuy vậy việc giảm chu kỳ bus dẫn đến các khó khăn về mặt kỹ thuật, dẫn đến một hiệu ứng gọi là bus skew (skew: nghiêng, xiên, ghềnh). Điều quan trọng là thời gian chu kỳ phải dài hơn so với skew để tránh việc những khoảng thời gian được số hoá lại trở thành các đại lượng biến thiên liên tục.

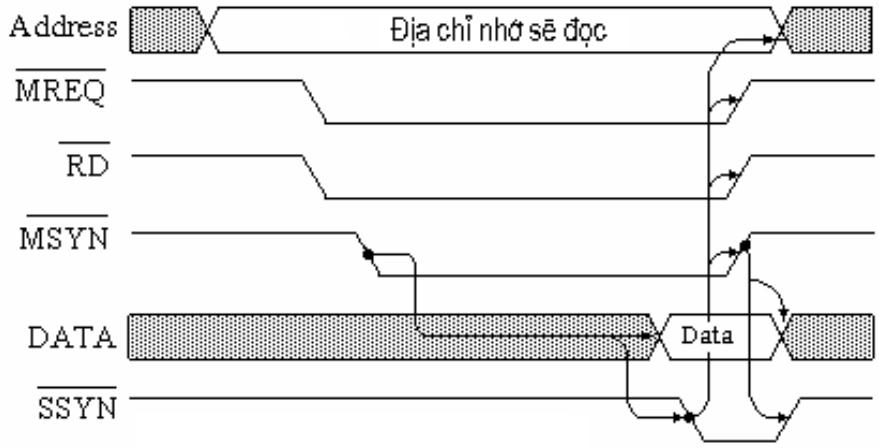
- Bus không đồng bộ:

Bên Master cần lần lượt phát ra: Address, $\overline{\textbf{MREQ}}$, $\overline{\textbf{RD}}$, $\overline{\textbf{MSYN}}$ (Master SYNchronization)

Bên Slave: Đưa dữ liệu ra, Phát tín hiệu $\overline{\textbf{SSYN}}$ (Slave SYNchronization) tích cực.

Bên Master: Chốt dữ liệu lại, Đảo $\overline{\textbf{MSYN}}$ thành không tích cực

Bên Slave: Ngừng đưa dữ liệu ra, Đảo $\overline{\textbf{SSYN}}$ thành không tích cực



Full handshake: Đó là tập các tín hiệu phối hợp với nhau, chủ yếu gồm có 4 tín hiệu (4 sự kiện):

$\overline{\textbf{MSYN}}$ được đặt tích cực

$\overline{\textbf{SSYN}}$ được đặt tích cực để đáp lại tín hiệu $\overline{\textbf{MSYN}}$ tích cực

$\overline{\textbf{MSYN}}$ được đặt tích cực để đáp lại tín hiệu $\overline{\textbf{SSYN}}$ tích cực

$\overline{\textbf{SSYN}}$ được đảo thành không tích cực để đáp lại tín hiệu $\overline{\textbf{MSYN}}$ đảo thành không tích cực

Nhận xét: Full handshake là độc lập thời gian, mỗi sự kiện được gây ra bởi một sự kiện trước đó chứ không phải bởi một xung đồng hồ. Nếu một cặp master-slave nào đó hoạt động chậm thì cặp kế tiếp không hề bị ảnh hưởng. Tuy ưu điểm của bus không đồng bộ rất rõ ràng, nhưng trong thực tế phần lớn các bus đang được sử dụng lại là loại đồng bộ. Lý do căn bản là các hệ thống sử dụng bus đồng bộ dễ thiết kế hơn. MPU chỉ cần chuyển các mức tín hiệu cần thiết sang trạng thái tích cực là các chip nhớ đáp lại ngay, không cần tín hiệu phản hồi. Chỉ cần các chip được chọn phù hợp thì mọi hoạt động đều trôi trảy, không cần phải "bắt tay" (handshake).

Theo loại tín hiệu truyền trên bus:

Bus địa chỉ (**Address bus**): Đường dẫn các thông tin về tín hiệu địa chỉ. Địa chỉ: gồm vị trí bộ nhớ và thiết bị

Bus dữ liệu (**Data bus**): Đường dẫn các thông tin dữ liệu.Dữ liệu: gồm nội dung bộ nhớ và số liệu thiết bị

Bus điều khiển (**Control bus**): Đường dẫn các thông tin tín hiệu điều khiển (xác định các thao tác).

c. Tần số bus

Có 2 kiểu hoạt động: Bus đồng bộ và Bus bất đồng bộ

Máy tính hoạt động theo bus đồng bộ, làm việc theo nhịp

Trên mainboard có 1 bộ dao động tạo các xung nhịp tuần hoàn cho máy tính (xung đồng hồ)

Tần số nhịp đo theo đơn vị Hz

Hz = số nhịp / s

Khz = 1000Hz

Mhz = 1000 Khz

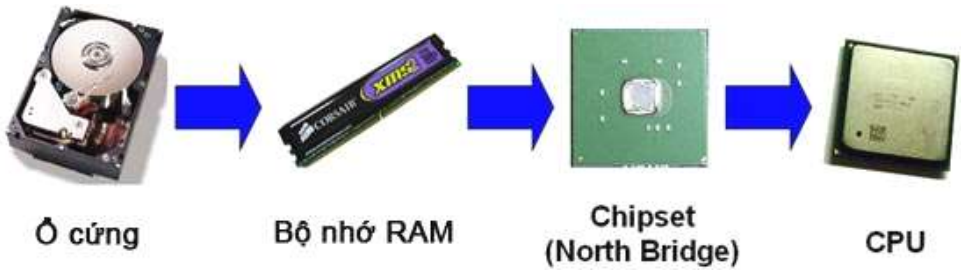
Ghz = 1000 Mhz

Tần số bus cục bộ, bus hệ thống, CPU, bus mở rộng, bộ nhớ.

4.2 Cách thức hoạt động của CPU

Khi kích đúp vào một biểu tượng nào đó để chạy chương trình thì những gì sẽ xảy ra là:

1. Chương trình đã lưu bên trong ổ đĩa cứng sẽ được đưa vào bộ nhớ RAM. Ở đây chương trình chính là một loạt các chỉ lệnh đối với CPU.
2. CPU sử dụng mạch phần cứng được gọi là memory controller để tải dữ liệu chương trình từ bộ nhớ RAM.
3. Lúc đó dữ liệu bên trong CPU sẽ được xử lý.
4. Những gì diễn ra tiếp theo sẽ phụ thuộc vào chương trình vừa được nạp. CPU có thể tiếp tục tải và thực thi chương trình hoặc có thể thực hiện một công việc nào đó với dữ liệu đã được xử lý, như việc hiển thị kết quả thực hiện nào đó lên màn hình.



Hình 4-3 Dữ liệu được lưu đưa vào CPU

Trước đây, CPU điều khiển sự truyền tải dữ liệu giữa ổ đĩa cứng và bộ nhớ RAM. Vì ổ đĩa cứng thường có tốc độ truy cập thấp hơn so với bộ nhớ RAM nên nó làm chậm chung cho cả hệ thống, chính vì vậy CPU sẽ rất bận cho tới khi dữ liệu đã được truyền tải từ ổ đĩa cứng vào bộ nhớ RAM. Phương pháp này được gọi là PIO, Processor I/O (hay Programmed I/O). Ngày nay, sự truyền tải dữ liệu giữa ổ đĩa cứng và bộ nhớ RAM được thực hiện mà không sử dụng đến CPU, như vậy nó sẽ

làm cho hệ thống hoạt động nhanh hơn. Phương pháp này được gọi là bus mastering hay DMA (Direct Memory Access). Để đơn giản hóa hơn cho hình vẽ, bài giảng không đưa vào chip cầu nối (được gọi là north bridge chip) giữa ổ đĩa cứng và bộ nhớ RAM trên hình 4-3, tuy nhiên là có một chip đó tại vị trí nối này.

Nhìn chung, tất cả các hoạt động trong máy tính đều trải qua 4 chu kỳ chính đó là: Lấy lệnh, Giải mã lệnh, Nhận dữ liệu, Xử lý dữ liệu

Lấy lệnh (Fetch Instructions - FI)

- Địa chỉ của lệnh cần thực hiện nằm trong bộ đếm chương trình (PC - Program Counter), được đưa qua bộ đệm địa chỉ, qua bus địa chỉ để tìm ra ngăn nhớ chứa lệnh.
- Tiếp theo, BXL phát ra tín hiệu đọc ngăn nhớ vừa tìm được
- Nội dung của ngăn nhớ được chuyển qua bus dữ liệu và đưa đến thanh ghi lệnh (Instruction Reg.)

Giải mã lệnh (Instructions Decode – ID)

- Lệnh từ thanh ghi lệnh được đưa đến khối điều khiển
- Tại đây, lệnh được giải mã để xác định thao tác mà lệnh yêu cầu
- Khi đó, khối điều khiển sẽ phát ra tín hiệu điều khiển tương ứng với lệnh đó.

Nhận dữ liệu (Fetch Data – FD)

- BXL phát ra địa chỉ của ngăn nhớ/cổng vào ra chứa dữ liệu cần nhận
- BXL phát ra tín hiệu điều khiển đọc ngăn nhớ/cổng vào ra tương ứng
- Dữ liệu được chuyển qua bus dữ liệu đưa vào tập thanh ghi bên trong

Xử lý dữ liệu (Process Data – PD)

- Dữ liệu được chuyển từ các thanh ghi vào ALU
- ALU sẽ thực hiện các phép toán dưới sự điều khiển của khối điều khiển
- Kết quả phép toán được cất tạm thời vào thanh ghi dữ liệu

4.3 Vi kiến trúc

4.3.1 Đường dữ liệu

Đường dữ liệu là một phần của CPU, có chức năng đọc các toán hạng từ các thanh ghi, thực hiện các phép tính trên toán hạng đó và lưu trữ kết quả vào trong các thanh ghi.

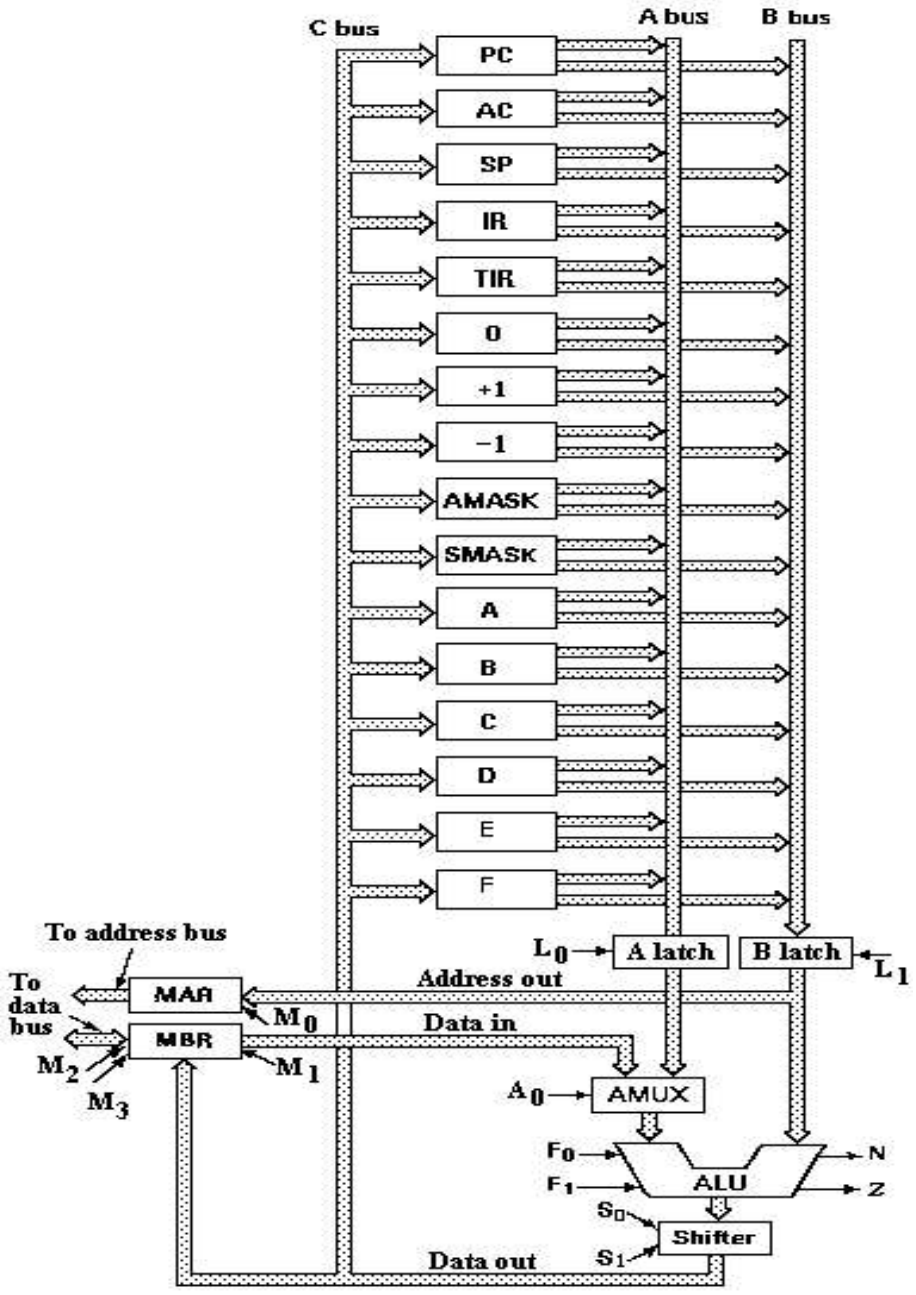
Các thành phần trong đường dữ liệu gồm có:

- 16 thanh ghi 16 bit giống nhau tạo nên một bộ nhớ tạm chỉ truy cập được ở mức vi chương trình.

- Thanh ghi đưa nội dung ra: bus A, bus B hoặc đồng thời cả 2 bus.
- Nạp vào thanh ghi: từ bus C.
- ALU (16 bit) có thể thực hiện 4 chức năng: A+B, A AND B, A, NOT A
- F0 và F1 định chức năng sẽ được ALU thực hiện.
- ALU sinh ra hai bit trạng thái dựa trên kết quả ra hiện thời của nó:
    - N có giá trị 1 khi kết quả ra là âm
    - Z có giá trị 1 khi kết quả ra bằng 0 (Zê-rô).
- Shifter có thể dịch: trái, phải, không dịch. Có thể thực hiện dịch trái hai bit một giá trị ghi trong thanh ghi R, bằng cách tính R+R trong ALU, sau đó dịch kết quả (tổng) một bit nữa sang trái bằng thanh ghi dịch.
- A latch và B latch: được nạp từ bus A và bus B để cho phép có thể thay đổi nội dung các thanh ghi đã nạp giá trị vào ALU.
- Các tín hiệu L0 và L1 điều khiển việc nạp giá trị trên bus A và bus B vào các thanh ghi chốt.
- MAR: thanh ghi địa chỉ ô nhớ cần thao tác
    - Có thể được nạp từ thanh ghi chốt B song song với một thao tác của ALU.
    - M0 điều khiển việc nạp của MAR.
- MBR: thanh ghi đệm (dữ liệu) đọc/ghi bộ nhớ.
    - Được nạp giá trị từ đầu ra của thanh ghi dịch, giá trị này cũng có thể đồng thời được chứa vào một trong các thanh ghi của bộ nhớ tạm.
    - M1 điều khiển việc nạp của MBR từ đầu ra của thanh ghi dịch
    - M2 và M3 điều khiển việc đọc và ghi bộ nhớ.
- AMUX: bộ dồn kênh để chọn dữ liệu đưa vào ALU từ A latch hay từ MBR
    - A0 điều khiển AMUX.

Các tín hiệu điều khiển đường dữ liệu gồm có 9 nhóm:

- 16 tín hiệu để điều khiển việc nạp cho bus A từ bộ nhớ tạm (có 16 thanh ghi).
- 16 tín hiệu để điều khiển việc nạp cho bus B từ bộ nhớ tạm.
- 16 tín hiệu để điều khiển việc nạp cho bộ nhớ tạm từ bus C.
- 2 tín hiệu để điều khiển 2 thanh ghi chốt A và B.
- 2 tín hiệu để điều khiển chức năng ALU.
- 2 tín hiệu để điều khiển bộ dịch.
- 4 tín hiệu để điều khiển MAR và MBR.
- 2 tín hiệu để chỉ rõ thao tác đối với bộ nhớ (R/W).
- 1 tín hiệu để điều khiển Amux.



Hình 4-4 Cấu trúc đường dữ liệu

4.3.2 Vi chỉ thị

Để điều khiển đường dữ liệu có thể thông qua 61 tín hiệu kể trên, các tín hiệu này có thể được thiết kế trong một thanh ghi điều khiển có kích thước bằng 61 bit. Mỗi bit dùng cho một tín hiệu điều khiển: bit bằng 1 có nghĩa là tín hiệu được đặt tích cực, còn bằng 0 nghĩa là không tích cực.

Một chu kỳ của đường dữ liệu bao gồm các chu kỳ:

- Một chu kỳ bao gồm việc mở cổng cho các giá trị trong bộ nhớ tạm đi vào bus A và bus B, chốt chúng lại trong hai thanh ghi chốt bus A latch và B latch
- Cho các giá trị từ các thanh ghi chốt chạy qua ALU và shifter
- Cuối cùng là việc chứa các kết quả vào trong bộ nhớ tạm hoặc vào MBR. Ngoài ra MAR cũng có thể được nạp, sau đó một chu kỳ bộ nhớ sẽ bắt đầu.

Có thể giảm số bít cần thiết để điều khiển đường dữ liệu nếu chấp nhận bổ sung thêm một số mạch điện:

- Giảm số bit điều khiển mỗi bus A, B và C từ 16 xuống còn 4: sử dụng bộ giải mã 4-to-16.Chấp nhận chỉ nạp dữ liệu trên bus C vào 1 trong 16 thanh ghi.
- Bỏ 2 bit điều khiển A latch và B latch: vì luôn cần chúng tích cực ở các thời điểm xác định nên có thể dùng luôn tín hiệu đồng hồ.
- Nếu bộ nhớ đã dùng tín hiệu RD và WR thì có thể dùng chúng thay thế cho M2 và M3
- Tuy nhiên nên bổ sung 1 tín hiệu là ENC: cho / không cho phép chứa giá trị trên bus C vào bộ nhớ tạm.
    - ENC=1: Có cất giá trị trên bus C
    - ENC=0: Không cất

Khuôn dạng của vi chỉ thị: bao gồm 13 trường, tên và ý nghĩa của các trường được liệt kê trong hình 4-5

Bits	1	2	2	2	1	1	1	1	1	4	4	4	8
	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR

AMUX	COND	ALU	SH	MBR,MAR,RD ,WR,ENC
0 = A latch 1 = MBR	0= No jump 1= Jump if N=1 2= Jump if Z=1 3= Jump always	0= A+B 1= A AND B 2= A 3 = $\overline{A}$	0=No shift 1=Shift right 1 bit 2=Shift left 1 bit 3=(not used)	0 = No 1 = Yes

AMUX	Điều khiển lối vào bên trái của ALU: 0 = A latch, 1 = MBR
ALU	Chức năng ALU: 0 = A+B, 1 = A AND B, 2 = A, 3 = $\overline{A}$
SH	Chức năng thanh ghi dịch: 0 = no shift, 1 = right, 2 = left
MBR	Nạp cho MBR từ thanh ghi dịch: 0 = không nạp, 1 = nạp
MAR	Nạp cho MAR từ thanh ghi chốt B: 0 = không nạp, 1 = nạp
RD	Yêu cầu đọc bộ nhớ: 0=không đọc, 1=nạp cho MBR từ bộ nhớ
WR	Yêu cầu ghi bộ nhớ: 0 = không ghi, 1 = ghi MBR vào bộ nhớ
ENC	Điều khiển việc chứa vào bộ nhớ tạm: 0=không chứa, 1=chứa
C	Chọn thanh ghi để chứa vào (nếu ENC = 1): 0 = PC, 1 = AC, .v.v.
B	Chọn nguồn cho bus B: 0 = PC, 1 = AC, .v.v.
A	Chọn nguồn cho bus A: 0 = PC, 1 = AC, .v.v.

Diễn giải một vi chỉ thị điều khiển đường dữ liệu

Hình 4-5 Diễn giải một vi chỉ thị điều khiển đường dữ liệu

4.3.3 Định thời cho vi chỉ thị

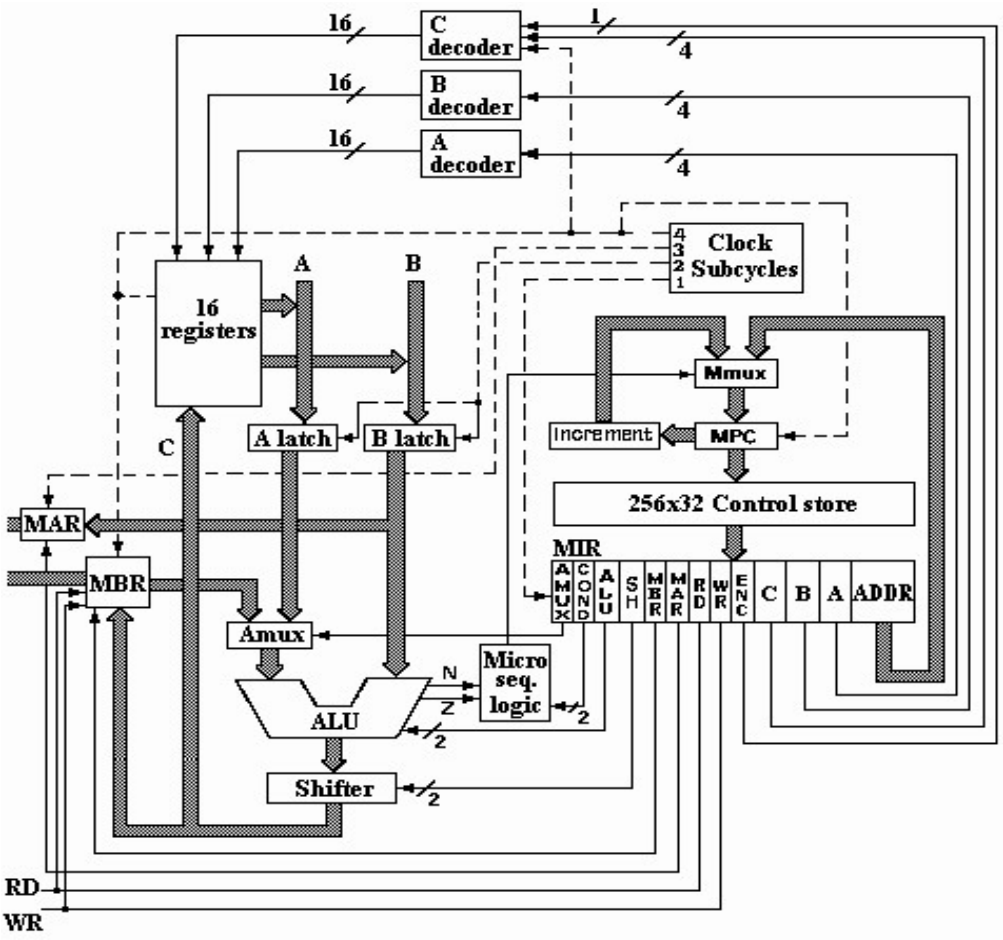
Các trường trong vi chỉ thị không nhất thiết, hoặc không được điều khiển đồng thời, nên cần phải thực hiện định thời (timing) cho vi chỉ thị. Định thời là việc định trình tự xảy ra các sự kiện trong quá trình thực hiện một vi chỉ thị.

Một chu kỳ ALU cơ sở bao gồm việc:

1. Nạp vi chỉ thị tiếp theo sẽ được thi hành vào MIR .

2. Mở cổng các thanh ghi tạm đi vào bus A và bus B và giữ chúng trong A latch và B latch.
3. Khi các giá trị đưa vào ALU ổn định, dành thời gian cho ALU và shifter để chúng sinh ra giá trị ra ổn định; nạp MAR nếu cần thiết.
4. Khi giá trị ra của shifter ổn định, chứa giá trị trên bus C vào bộ nhớ tạm và nạp cho MBR nếu việc này cũng được yêu cầu.

Để đạt được việc định đúng trình tự các sự kiện, chúng ta sử dụng một đồng hồ 4 pha (có 4 chu kỳ con), như trên hình vẽ 4-06. Trên hình là sơ đồ khối mô tả chi tiết về vi cấu trúc bao gồm 2 phần là đường dữ liệu và phần điều khiển. Trong phần điều khiển, đơn vị quan trọng nhất chính là bộ điều khiển (control store), đây là nơi chứa vi lệnh có dung lượng là 256x32. Bộ nhớ điều khiển cần 1 thanh ghi để trỏ đến vi lệnh tiếp theo sẽ được thực thi đó là bộ đếm vi chương trình (MPC – MicroProgram Counter).



Hình 4-6 Sơ đồ khối Vi kiến trúc

Việc định thời cho vi chỉ thị được thực hiện trong 4 chu kỳ con. Ta thấy bộ điều khiển liên tục sao chép vi lệnh được địa chỉ hoá bởi MPC vào thanh ghi MIR. Trong 3 chu kỳ con còn lại MIR không bị ảnh hưởng và không có gì xẩy ra đối với MPC. Chu kỳ tiếp theo, MIR ổn định và các trường khác bắt đầu điều khiển đường dữ liệu. Hai bộ giải mã A và B bắt đầu thực hiện giải mã chô việc chọn các thanh ghi để xuất nôi dung lên Bus A và B. Mạch tạo xung clock cũng tác động lên A latch và B latch chốt dữ liệu ở đầu vào và đưa dữ liệu ra. Bộ đếm vi chương trình tăng giá trị lên 1 (MPC=MPC+1) để chuẩn bị cho việc nạp vi lệnh kế tiếp. Tại chu kỳ con thứ ba, ALU và shifter được cung cấp thời gian để tạo các kết quả có giá trị. Trường Amux của vi lệnh xác định ngõ vào bên trái của ALU, còn ngõ vào bên phải luôn được lấy từ mạch chốt B. Trong khi ALU và shifter đang tính toán thì MAR được nạp từ bus B nếu trường MAR là 1. Chu kỳ con thứ 4, tuỳ thuộc vào hai trường ENC và MBR, kết quả đầu ra của ALU sẽ được đưa vào một trong hai vị trí là thanh ghi MBR hay bộ nhớ nháp. Mạch giải mã C có ngõ vào nối với ENC, đường xung clock 4 cà trường C của vi lệnh. Mạch này tạo ra 16 tín hiệu điều khiển. Ở bên trong mạch này thực hiện việc giải mã 4-to-16 với 4 ngõ vào từ trường C rồi AND mỗi ngõ ra với tín hiệu là kết quả của việc AND đường chu kỳ con 4 với ENC. Trong chu kỳ con thứ 4, thanh ghi MBR cũng được nạp nếu trường MBR=1.

4.3.4 Định trình tự cho các vi chỉ thị

Đó là việc chọn vi chỉ thị tiếp theo: tuần tự hay nhảy (Jump). Trong mỗi vi lệnh sẽ có hai trường: ADDR là địa chỉ của vi lệnh tiếp theo và COND quyết định vi lệnh kế tiếp được tìm nạp từ MPC+1 hay từ ADDR. Mọi vi lệnh đều có khả năng chứa một lệnh nhảy có điều kiện. Quyết định này có được do lệnh nhảy có điều kiện được dùng rất phổ biến trong các vi chương trình và cho phép mọi vi lệnh có thể có 2 vi lệnh kế tiếp nhằm cho chương trình chạy nhanh hơn so với việc thiết lập một điều kiện nào đó trong vi lệnh và sau đó kiểm tra điều kiện này trong vi lệnh kế tiếp.

Việc chọn vi lệnh kế tiếp được đưa ra bởi khối Logic trình tự vi lệnh - Micro sequencing logic trong chu kỳ con thứ tư bằng cách dùng các tín hiệu N và Z của ALU. Đầu ra của khối này sẽ điều khiển mạch chọn kênh M(Mmux) để đưa MPC+1 hoặc ADDR tới. Có 4 cách lựa chọn vi chỉ thị tiếp theo, chỉ ra bằng trường COND như sau:

0 = Không nhảy

1 = Nhảy tới ADDR nếu N =1

2 = Nhảy tới ADDR nếu Z =1

3 = Nhảy tới ADDR vô điều kiện

IV. Bài tập củng cố kiến thức

4.1 Bộ xử lý trung tâm của máy tính bao gồm các thành phần nào? Chức năng của mỗi thành phần đó?

4.2 Trong bộ VXL 8086 có bao nhiêu nhóm thanh ghi chính? Hãy cho biết chức năng của các nhóm thanh ghi đó. Giả sử bộ nhớ có địa chỉ logic là 10FFh: 0001h, hãy tính địa chỉ vật lý tương ứng với địa chỉ logic đó.

4.3 Trong kiến trúc đường dữ liệu, thanh ghi MBR có nhiệm vụ gì? Có những tín hiệu nào tác động đến thanh ghi này?

4.4 Trong kiến trúc đường dữ liệu, thanh ghi MAR có nhiệm vụ gì? Có những tín hiệu nào tác động đến thanh ghi này?

4.5 Các thành phần và nhiệm vụ của đường đi dữ liệu?

4.6* Hãy mô tả đường đi dữ liệu cho các lệnh sau đây

ADD A, 1001_2

NEG A

CHƯƠNG 5 MỨC MÁY THÔNG THƯỜNG

I. Mục đích

- Giúp sinh viên nắm được về khuôn dạng lệnh
- Nắm được về phân loại, cấu tạo và nguyên lý làm việc của bộ nhớ Caches
- Nắm được về các loại bộ nhớ

II. Yêu cầu

- Sinh viên làm bài tập, bài tập tiếng anh có thể làm bằng tiếng việt hoặc tiếng anh
- Sinh viên biết phân biệt các loại bộ nhớ

III. Nội dung: Chương 5 được giảng dạy trong 6 tiết lý thuyết và 3 tiết bài tập

5.1 Khuôn dạng lệnh

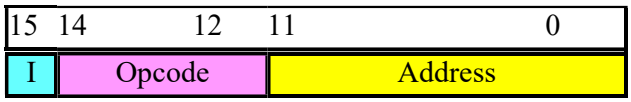
Chương trình bao gồm một chuỗi các chỉ thị, mỗi chỉ thị xác định một tác động cụ thể nào đó. Một phần trong chỉ thị bao gồm các trường như sau

- Mã phép toán (opcode) (trường này luôn phải có): cho biết thao tác gì được thực hiện
- Địa chỉ bộ nhớ của toán hạng hay chứa một giá trị hằng số.

Khuôn dạng lệnh thường có 3 dạng: lệnh không chứa toán hạng, lệnh chứa một toán hạng, lệnh chứa hai toán hạng.

Nhiều chỉ thị chứa hoặc chỉ rõ vị trí của dữ liệu được chỉ thị sử dụng. Vấn đề chính là việc xác định xem toán hạng được ở đâu bộ nhớ, thanh ghi hay các cổng vào/ra thông qua cách thức định địa chỉ. Trong chỉ thị thường có ba dạng tham chiếu: Lệnh tham chiếu bộ nhớ, Lệnh tham chiếu thanh ghi, Lệnh vào/ra

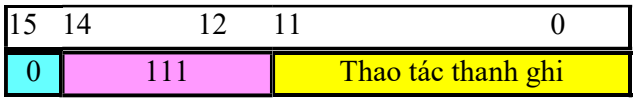
5.1.1 Lệnh tham chiếu bộ nhớ



Trong đó:

- 12bit để xác định địa chỉ
- 3 bit xác định mã lệnh
- 1 bit để xác định kiểu định địa chỉ
  - I=0: định địa chỉ trực tiếp
  - I=1 định địa chỉ gián tiếp

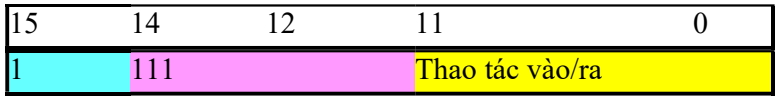
5.1.2 Lệnh tham chiếu thanh ghi



Trong đó:

- Mã lệnh: 111
- Bit 15 =0

5.1.3 Lệnh tham chiếu vào ra



Trong đó:

- Mã lệnh: 111
- Bit 15 =1

Tập các chỉ thị cơ bản được thi hành cho cấp máy 2 và 3 (ngôn ngữ sử dụng là hợp ngữ) được mô tả trong bảng 5.1

xxxxxxxxxxxx là địa chỉ máy 12 bit; ở cột 4 nó được ghi là x

YYYYYYYY là một hằng số 8 bit; ở cột 4 nó được ghi là y

Mã nhị phân (Binary)	Ký hiệu (Mnemonic)	Thực hiện	Giải thích
0000xxxxxxxxxxxx	LODD	Load direct	ac:=m[x]
0001xxxxxxxxxxxx	STOD	Store direct	m[x]:=ac
0010xxxxxxxxxxxx	ADDD	Add direct	ac:=ac+m[x]
0011xxxxxxxxxxxx	SUBD	Subtract direct	ac:=ac-m[x]
0100xxxxxxxxxxxx	JPOS	Jump positive	if ac ≥ 0 then pc:=x
0101xxxxxxxxxxxx	JZER	Jump zero	if ac=0 then pc:=x
0110xxxxxxxxxxxx	JUMP	Jump	pc:=x
0111xxxxxxxxxxxx	LOCO	Load constant	ac:=x (0 ≤ x ≤ 4095)
1000xxxxxxxxxxxx	LODL	Load local	ac:=m[sp+x]
1001xxxxxxxxxxxx	STOL	Store local	m[x+sp]:=ac
1010xxxxxxxxxxxx	ADDL	Add local	ac:=ac+m[sp+x]
1011xxxxxxxxxxxx	SUBL	Subtract local	ac:=ac-m[sp+x]
1100xxxxxxxxxxxx	JNEG	Jump negative	if ac < 0 then pc:=x
1101xxxxxxxxxxxx	JNZE	Jump nonzero	if ac <> 0 then pc:=x

Bảng 5-1 Tập chỉ thị của Vi kiến trúc

5.2 Mô hình phân cấp bộ nhớ

Chúng ta cần xem xét bộ nhớ ở 2 góc độ là logic (hoạt động) và vật lý (cấu tạo). Trước hết chúng ta cần biết bộ nhớ là gì, chúng nằm ở đâu trong hệ thống và hoạt động như thế nào. Sau đó chúng ta sẽ xem xét một số loại bộ nhớ và tốc độ, hình dạng và các modul nhớ mà ta có thể cài đặt.

Bộ nhớ là không gian làm việc của bộ vi xử lý. Đó là nơi cất giữ tạm thời các chương trình và dữ liệu đang được thao tác với bộ vi xử lý. Lưu trữ bộ nhớ là được xem tạm thời vì dữ liệu chỉ tồn tại trong thời gian máy đang hoạt động và không bị khởi động lại. Trước khi khởi động lại hay tắt máy, dữ liệu đã thay đổi cần được ghi

vào các thiết bị lưu trữ lâu dài (thường là ổ đĩa cứng) để sau này có thể nạp lại vào bộ nhớ.

Chúng ta hay gọi bộ nhớ là RAM tức là Random Access Memory (bộ nhớ truy cập ngẫu nhiên). Bộ nhớ chính được gọi là RAM vì chúng ta có thể truy cập một cách ngẫu nhiên và nhanh chóng bất kỳ vị trí nào trong bộ nhớ. Tên gọi này đôi lúc có thể gây nhầm lẫn vì bộ nhớ ROM cũng truy nhập ngẫu nhiên nhưng khác với RAM là nó không thể ghi được. Tương tự, bộ nhớ đĩa cũng được truy cập ngẫu nhiên nhưng rõ ràng đó không phải là RAM.

Qua quá trình phát triển nhiều năm của máy tính, khái niệm RAM không còn là một chữ viết tắt nữa mà nó chỉ một không gian nhớ chính để bộ vi xử lý chạy được các chương trình. Nó thường được cấu tạo bởi các chip có tên là DRAM (Dynamic RAM – RAM động). Một đặc điểm của bộ nhớ DRAM là nó lưu trữ dữ liệu theo dạng động, nghĩa là thông tin có thể ghi đi ghi lại trên RAM bất kỳ lúc nào. Một tính chất quan trong nữa của bộ nhớ DRAM là dữ liệu tồn tại cho đến khi mất điện.

Do đó, khi nói đến bộ nhớ của máy tính, chúng ta muốn nói đến RAM hay bộ nhớ vật lý của hệ thống, là những modul hay các chip nhớ chính để lưu trữ các chương trình và dữ liệu mà bộ vi xử lý đang hoạt động. Không nên nhầm lẫn giữa bộ nhớ với thuật ngữ dung lượng lưu trữ (storage) được dùng để chỉ các ổ đĩa từ và băng từ (mặc dù chúng có thể được sử dụng như một dạng RAM – Bộ nhớ ảo: Virtual Memory).

RAM vừa chỉ các chip tạo nên bộ nhớ của hệ thống, vừa chỉ sơ đồ và cách bố trí logic của bộ nhớ. Sơ đồ và cách bố trí logic cho biết cách các địa chỉ bộ nhớ được sắp xếp tương ứng với các chip và vùng địa chỉ nào chứa loại thông tin nào.

Nhiều người mới làm quen với máy tính thường nhầm lẫn giữa bộ nhớ và đĩa lưu trữ(storage disk) bởi vì cả hai cùng sử dụng đơn vị đo lường là byte, Kilobyte, Megabyte, Gigabyte hoặc đơn vị cao hơn.

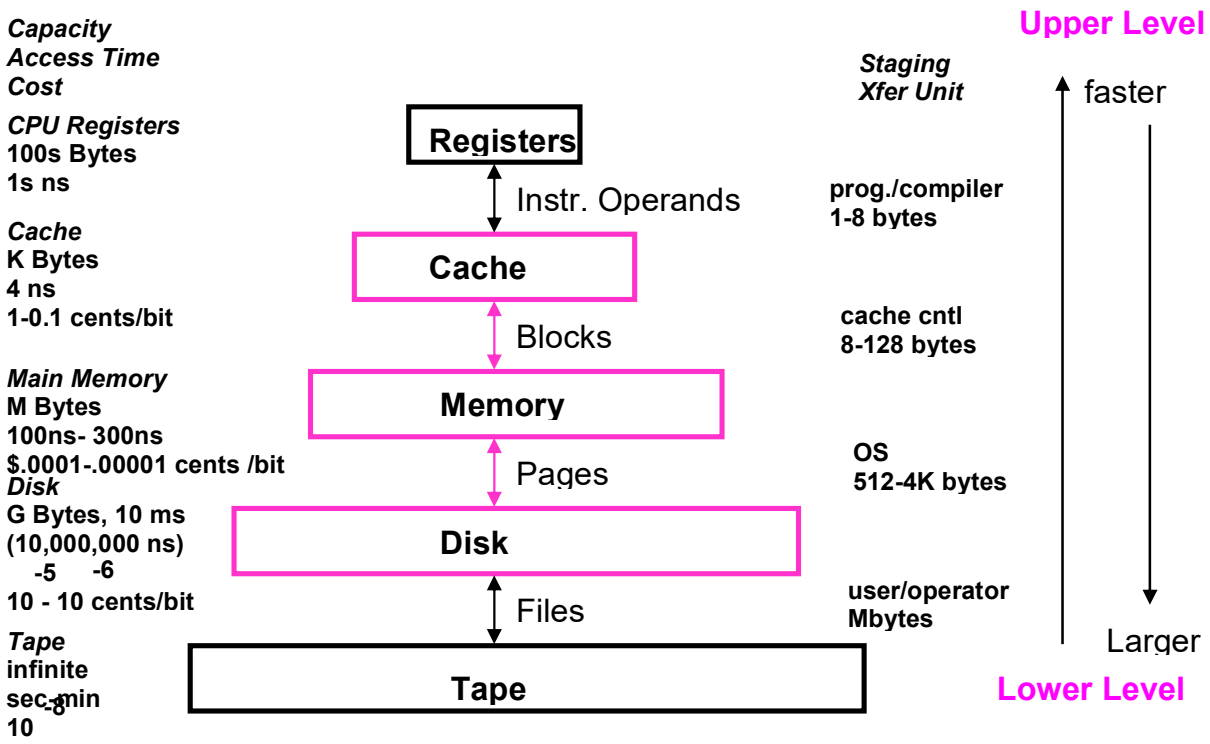
Một điểm quan trọng mà chúng ta cần ghi nhớ đó là: một file khi được nạp vào trong bộ nhớ, nó chỉ là bản sao của file đó, còn thực chất file đó vẫn tồn tại trong ổ đĩa. Vì tính chất tạm thời của bộ nhớ cho nên file đã bị thay đổi cần được ghi lại vào đĩa cứng trước khi tắt máy, vì khi tắt máy dữ liệu trong bộ nhớ sẽ bị xoá. Bộ nhớ lưu trữ các chương trình đang chạy và dữ liệu của chúng sử dụng. Các chip nhớ có lúc được gọi là "lưu trữ không ổn định" (volatile storage) bởi vì khi tắt máy hoặc có sự cố về nguồn điện, lưu trữ trong RAM sẽ bị mất. Vì bản chất bất ổn định

như vậy cho nên nhiều người dùng có thói quen ghi lại thường xuyên (một số chương trình ứng dụng có thể ghi một cách tự động theo thời gian đã định).

Các đặc trưng của bộ nhớ

- Ví trí:
  - Bên trong CPU: Tập các thanh ghi (register), cache
  - Bộ nhớ trong: Bộ nhớ chính (Main memmory – RAM, ROM)
  - Bộ nhớ ngoài: Các thiết bị nhớ, RAID (HDD, CD-Rom, …)
- Dung lượng:
  - Độ dài từ nhớ (tính bằng bit) – Kích thước trên một đơn vị lưu trữ
  - Số lượng từ nhớ - Dung lượng bộ nhớ
- Đơn vị truyền:
  - Từ nhớ - Truyền tuần tự từng Word
  - Khối nhớ - Truyền một khối gồn n Word
- Phương pháp truy nhập:
  - Truy nhập tuần tự (băng từ) – Để đến được điểm n đầu từ phải duyệt qua n-1 vị trí trước.
  - Truy nhập trực tiếp (các loại đĩa) – Đầu từ di chuyển trực tiếp đến vị trí cần đọc.
  - Truy nhập ngẫu nhiên (bộ nhớ bán dẫn) – ô nhớ cần đọc sẻ được giả mã để lấy thông tin ngay lập tức.
  - Truy nhập liên kết (cache) – Truy cập thông qua ban sao của ô nhớ cần đọc
- Hiệu năng:
  - Thời gian truy nhập
  - Chu kỳ truy xuất bộ nhớ
  - Tốc độ truyền
- Kiểu bộ nhớ vật lý:
  - Bộ nhớ bán dẫn – Lưu trữ bằng điện
  - Bộ nhớ từ - Lưu trữ dùng từ tính
  - Bộ nhớ quang – Lưu trữ sử dụng công nghệ Laze
- Các đặc tính vật lý:
  - Khả biến/không khả biến
  - Xoá được/không xoá được

Việc trao đổi dữ liệu giữa CPU và bộ vi xử lý và bộ nhớ chính là một thao tác quan trọng, chiếm đa số trong các lệnh xử lý dữ liệu nên nó quyết định hiệu suất của hệ thống vi xử lý nói chung và máy tính nói riêng. Bộ nhớ chính và bộ nhớ ngoài thường có tốc độ trao đổi dữ liệu chậm (chênh lệch) hơn so với tốc độ làm việc của CPU (kể cả việc vận chuyển dữ liệu trong bộ vi xử lý). Để nâng cao tốc độ xử lý dữ liệu chung của toàn hệ thống, người ta tìm cách nâng cao tốc độ trao đổi dữ liệu (kể cả lệnh) giữa bộ vi xử lý và bộ nhớ ngoài phạm vi bộ vi xử lý. Dựa trên nguyên lý cục bộ về không gian và thời gian mà người ta xây dựng hệ thống nhớ 5 cấp như sau:

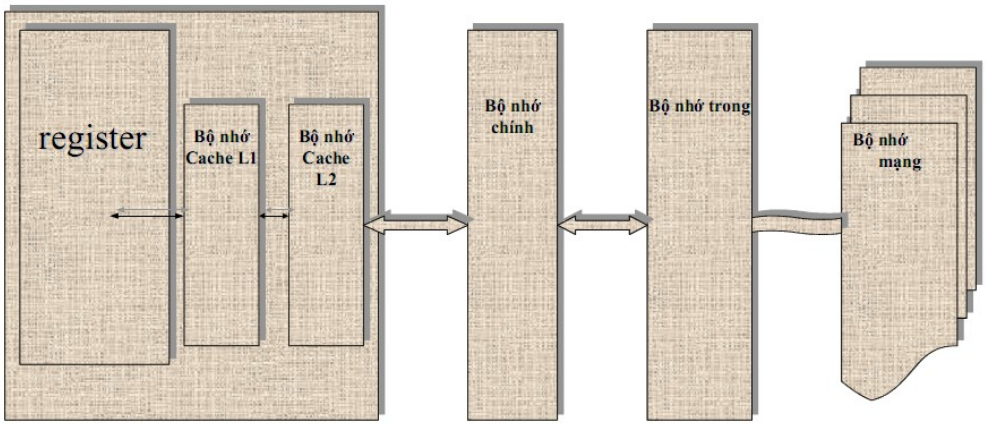


Hình 5-1 Mô hình phân cấp bộ nhớ

- ***Cấp 0 : Tập các thanh ghi nằm trong bộ vi xử lý. Thanh ghi là bộ nhớ kiểu*** SRAM nên tác động nhanh và thông tin ổn định. Đây là thành phần nhớ có tốc độ trao đổi dữ liệu nhanh nhất trong hệ thống vì nó gần ALU và CU. Tuy nhiên nó có dung lượng nhỏ.
- ***Cấp 1 : Primary cache (cache sơ cấp) : Là bộ nhớ có tốc độ trao đổi dữ liệu rất*** nhanh (nhỏ hơn thanh ghi), có dung lượng nhỏ và được đặt trong bộ vi xử lý, nhưng cũng có thể nằm ngoài bộ vi xử lý. Trong các bộ vi xử lý tiên tiến, bộ nhớ cache thường được tách (chia) làm 2 với mục đích tránh xung

đột trong xử lý song song (đại diện là pipline) là Icache : dành cho lệnh và Dcache : dành cho dữ liệu.

- ***Cấp 2 : Secondary cache (cache thứ cấp) : Cũng giống như Primary cache, nhưng*** loại này nằm ngoài bộ vi xử lý. Nó chỉ có khi có Primary cache (ngược lại, nó chính là Primary cache). Dung lượng của Secondary cache thường lớn hơn Primary cache và nhỏ hơn bộ nhớ.
- ***Cấp 3 : Main Memory (Bộ nhớ chính): Chứa chương trình và dữ liệu đang hoạt động.*** Bộ nhớ này được bộ vi xử lý đánh địa chỉ trực tiếp và quản lý thông qua địa chỉ đó. Một phần của chương trình đang được thi hành có thể nằm trong cache (lệnh và dữ liệu) nhằm tăng tốc độ hoạt động của hệ thống. Dung lương của bộ nhớ chính thường lớn hơn rất nhiều lần dung lượng bộ nhớ cache. Như đã biết, trong các hệ thống máy tính hiện đại ngày nay thì bộ nhớ chính thường là DRAM.
- ***Cập 4 : Secondary memory (Bộ nhớ thứ cấp – bộ nhớ ngoài***): Bộ nhớ này có dung lượng rất lớn nhưng tốc độ trao đổi dữ liệu chậm. Bộ nhớ này để lưu trữ chương trình và dữ liệu một cách lâu dài, cho nhiều người sử dụng (ghi/đọc, mất nguồn điện vẫn còn thông tin). Đại diện cho các bộ nhớ loại này đó chính là các ổ đĩa cứng, mềm CD ROM, CD – WOM, CD WR, băng từ, …



Từ trái qua phải: dung lượng tăng dần, tốc độ giảm dần, giá thành tính theo đơn vị byte hoặc bit giảm dần.

5.3 Bộ nhớ đệm (Cache)

5.3.1 Tổng quan và ý nghĩa của cache

Bộ nhớ Cache là kiểu bộ nhớ tốc độ cao có bên trong CPU để tăng tốc độ truy cập cho dữ liệu và các chỉ lệnh được lưu trong bộ nhớ RAM.

Một máy tính sẽ hoàn toàn vô dụng nếu bạn không bắt bộ vi xử lý (CPU) thực hiện một nhiệm vụ nào đó. Công việc sẽ được thực hiện thông qua một chương trình, chương trình này lại gồm rất nhiều các chỉ lệnh để ra lệnh cho CPU làm việc.

CPU lấy các chương trình từ bộ nhớ RAM. Tuy nhiên có một vấn đề với bộ nhớ RAM đó là khi nguồn nuôi của nó bị cắt thì các thành phần dữ liệu được lưu trong RAM cũng sẽ bị mất – chính điều này nên một số người nói rằng bộ nhớ RAM là một môi trường "dễ bay hơi". Các chương trình và dữ liệu như vậy phải được lưu trên môi trường không "dễ bay hơi" sau khi tắt máy tính (giống như các ổ đĩa cứng hay các thiết bị quang như đĩa CD và DVD).

Khi kích đúp vào một biểu tượng trong Windows để chạy một chương trình nào đó. Các chương trình thông thường được lưu trên ổ đĩa cứng của máy tính, khi được gọi nó sẽ được nạp vào bộ nhớ RAM sau đó từ bộ nhớ RAM, CPU nạp chương trình thông qua một mạch có tên gọi là memory controller, thành phần này được đặt bên trong chipset (north bridge chip- chíp cực bắc) trên các bộ vi xử lý Intel hoặc bên trong CPU trên các bộ vi xử lý AMD.

Vấn đề là CPU không thể tìm nạp dữ liệu trực tiếp từ các ổ đĩa cứng vì tốc độ truy suất dữ liệu của ổ đĩa cứng là quá thấp với nó, thậm chí nếu nếu bạn có cả ổ đĩa cứng với tốc độ truy suất lớn nhất. Hãy lấy một số ví dụ làm dẫn chứng cho điều này, ổ cứng SATA-300 – một loại ổ đĩa cứng có tốc độ nhanh nhất hiện đang được cung cấp ngày nay đến phần lớn người dùng – có tốc độ truyền tải theo lý thuyết là 300 MB/s. Một CPU chạy với tốc độ 2GHz với đường dữ liệu* 64-bit sẽ truyền tải dữ liệu bên trong với tốc độ 16GB/s – như vậy là lớn gấp 50 lần.

Sự khác nhau trong tốc độ cũng bắt nguồn từ một thực tế đó là các ổ đĩa cứng còn bao gồm cả hệ thống cơ khí, các hệ thống cơ khí này bao giờ cũng chậm hơn hệ thống điện tử thuần túy, các thành phần cơ khí phải chuyển động để dữ liệu mới có thể được đọc ra (điều này chậm hơn rất nhiều so với việc chuyển động của điện tử). Hay nói cách khác, bộ nhớ RAM là 100% điện tử, có nghĩa là nó sẽ nhanh hơn tốc độ của ổ đĩa cứng và quang.

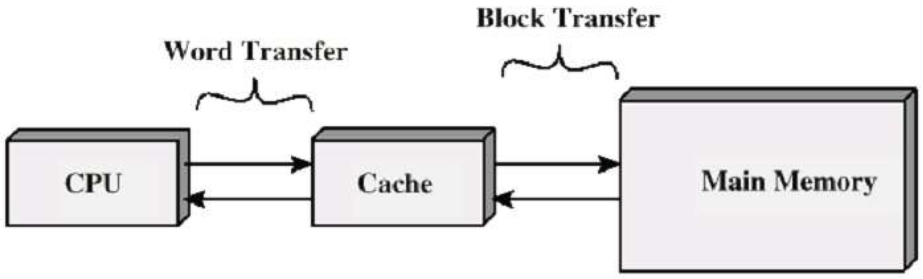
Tuy nhiên đây chính là vấn đề, thậm chí bộ nhớ RAM nhanh nhất cũng không nhanh bằng CPU. Nếu bạn sử dụng các bộ nhớ DDR2-800, chúng truyền tải dữ liệu ở tốc độ 6.400 MB/s – 12.800 MB/s nếu sử dụng chế độ hai kênh. Thậm chí con số này còn có thể lên đến 16GB/s trong ví dụ trước, vì các CPU hiện nay còn có thể tìm nạp dữ liệu từ L2 memory cache ở tốc độ 128- bit hay 256-bit, chúng ta đang nói về 32 GB/s hoặc 64 GB/s nếu CPU làm việc bên trong với tốc độ 2GHz.

Giải pháp đã tìm thấy để giảm sự ảnh hưởng trong việc sử dụng bộ nhớ RAM chậm hơn CPU là sử dụng một số lượng nhỏ các RAM tĩnh giữa CPU và bộ nhớ RAM. Công nghệ này được gọi là bộ nhớ Cache và ngày nay có một số lượng nhỏ bộ nhớ tĩnh này được đặt bên trong CPU.

Bộ nhớ Cache copy hầu hết các dữ liệu đã được truy cập gần đây từ bộ nhớ RAM vào bộ nhớ tĩnh và đoán dữ liệu gì CPU sẽ hỏi tiếp theo, tải chúng đến bộ nhớ tĩnh trước khi CPU yêu cầu thực sự. Mục đích là làm cho CPU có thể truy cập vào bộ nhớ Cache thay vì truy cập trực tiếp vào bộ nhớ RAM, vì nó có thể truy vấn dữ liệu từ bộ nhớ Cache một cách tức thời hoặc cũng hầu như ngay lập tức thay vì phải đợi khi truy cập vào dữ liệu được đặt trong RAM. CPU càng truy cập vào Cache nhớ thay cho RAM nhiều hơn thì hệ thống sẽ càng hoạt động nhanh hơn. Cũng theo đó, chúng ta sẽ sử dụng hoán đổi hai thuật ngữ "dữ liệu" và "chỉ lệnh" cho nhau vì những gì được lưu bên trong mỗi địa chỉ nhớ không có gì khác biệt đối với bộ nhớ.

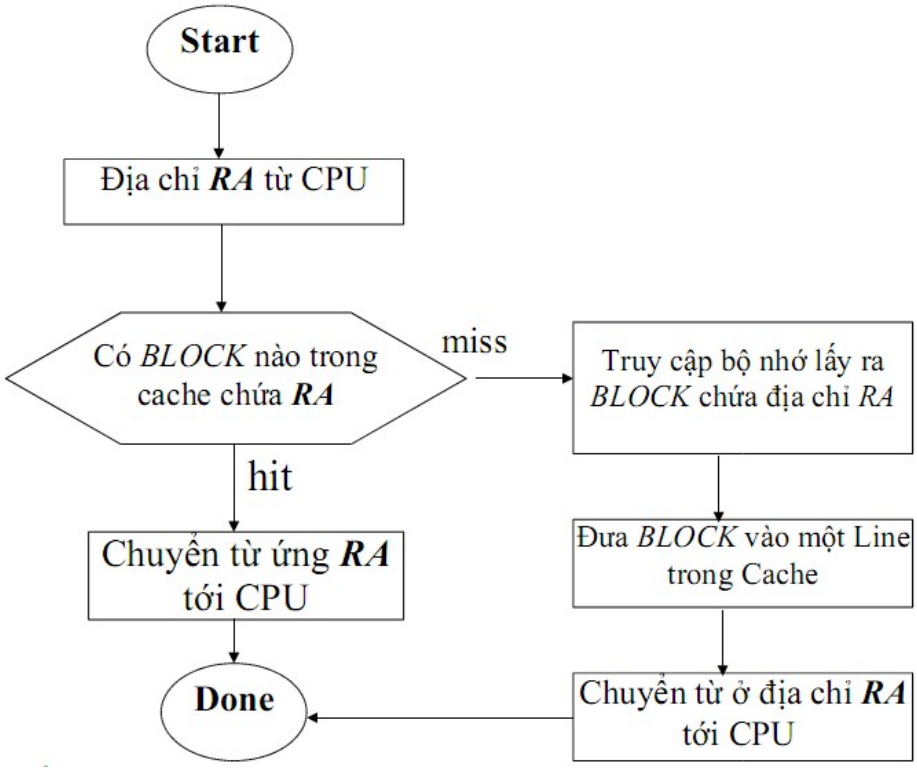
Nguyên tắc:

- Cache có tốc độ truy xuất nhanh hơn rất nhiều bộ nhớ chính
- Cache được đặt giữa CPU và bộ nhớ chính nhằm tăng tốc độ trao đổi thông tin giữa CPU và bộ nhớ chính.
- Cache thường được đặt trong chip vi xử lý

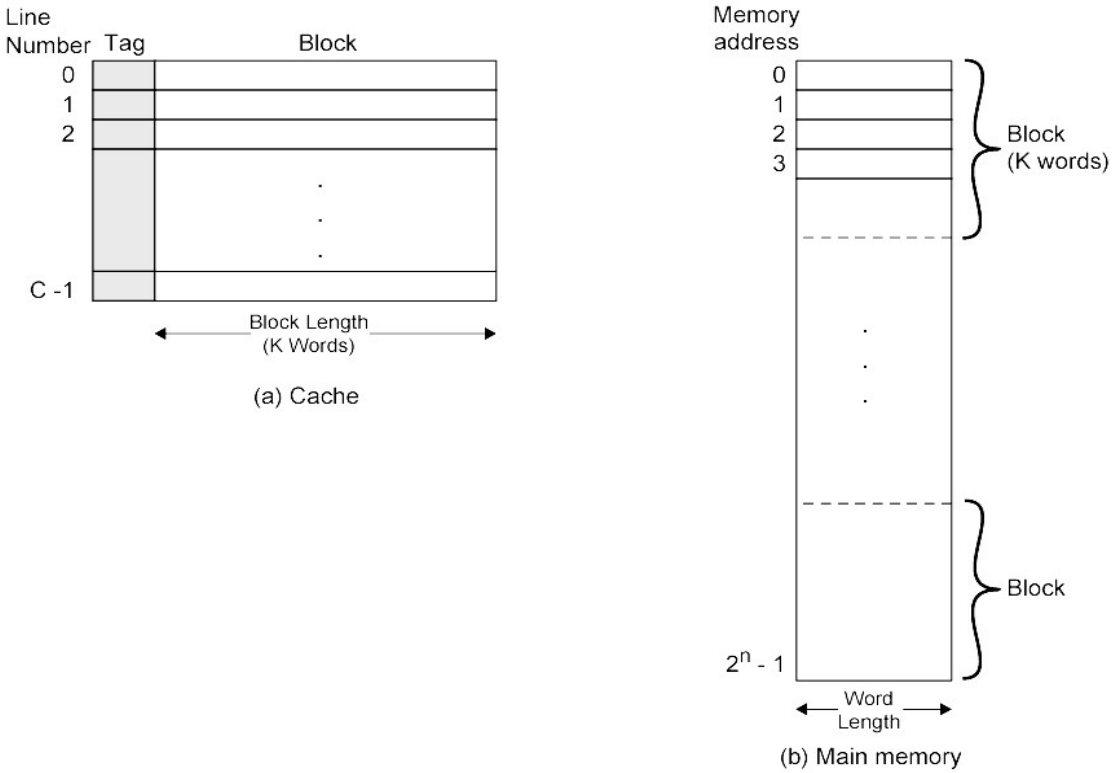


Các thao tác chính của Cache:

- CPU yêu cầu lấy nội dung của một ngăn nhớ bằng việc đưa ra một địa chỉ xác định ô nhớ.
- CPU kiểm tra xem có nội dung cần tìm trong Cache
  - Nếu có: CPU nhận dữ liệu từ bộ nhớ Cache
  - Nếu không có: Bộ điều khiển Cache đọc Block nhớ chứa dữ liệu CPU cần vào Cache. Tiếp đó chuyển dữ liệu từ Cache đến CPU
- Sơ đồ thao tác cache, bộ nhớ chính và CPU



5.3.2 Cấu trúc của bộ nhớ cache



- Bộ nhớ cache được chia thành các line (C line), mỗi line chia thành 2 trường, trường Tag và Block. Trường Tag lưu địa chỉ của ô nhớ được chuyển vào cache

từ main memmory (RAM). Trường Block lưu dữ liệu của block chuyển từ MainMem vào.

- Trên MainMem cũng được chia thành các block có kích thước bằng kích thước block trên Cache.
- Trong mỗi block lại được chia thành k word.

Vậy:

- Trên MainMem có 2^n word => số block = $2^n/k$
- Kích thước của cache = C * k word

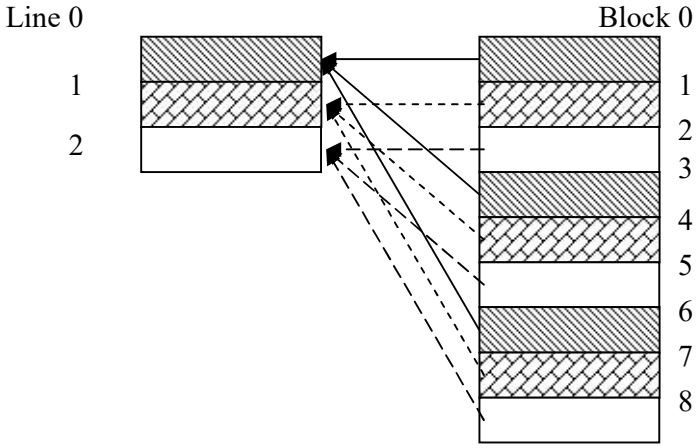
5.3.3 Các phương pháp ánh xạ cache

a) Ánh xạ trực tiếp

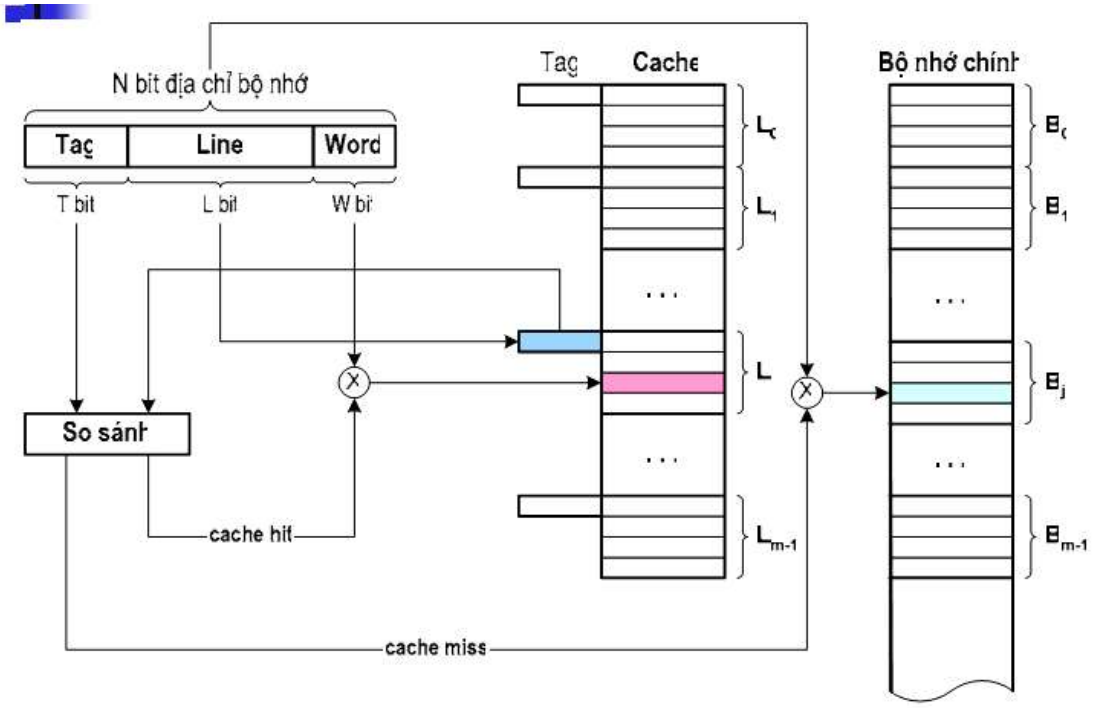
Nguyên tắc: Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một vị trí Line duy nhất của cache:

- 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 mod m) (với m là số Line của cache).



Mô tả:



❖ **Đặc điểm của ánh xạ trực tiếp**

Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường:

- Trường Word gồm W bit xác định một từ nhớ trong Block hay Line: 2^W = kích thước của Block hay Line
- Trường Line gồm L bit xác định một trong số các Line trong cache: 2^L = số Line trong cache = m
- Trường Tag gồm T bit: T = N - (W+L), 2^T = số block của bộ nhớ chính

Khi CPU phát ra một địa chỉ A có N bit với 3 trường trên, đầu tiên CPU sẽ truy cập vào line (L mod m) trên cache, so sánh trường tag của line này với trường tag trong địa chỉ A, nếu giống nhau (Cache hit) thì word thứ W trong line này sẽ được tải vào CPU. Nếu khác nhau (Cache miss) tức trong cache chưa có ô nhớ cần truy xuất, hệ thống sẽ truy xuất tới block thứ T (trường tag trong A) để tải block đó vào cache.

Ưu điểm:

- Bộ so sánh đơn giản vì chỉ thực hiện một lần với 2 toán hạng vào

Nhược điểm:

- Xác suất cache hit thấp và hiệu xuất của cache không cao vì mỗi một block chỉ có thể đưa vào một vị trí xác định trong khi các vị trí khác có thể đang trống.

Ví dụ:

Hãy xác định giá trị của Line(block), Tag và Word trong địa chỉ có kích thước 32 bit là 3FE9704Ah, biết rằng:

- Bộ nhớ sử dụng cơ chế ánh xạ trực tiếp.
- 1 word=2 byte:
- Kích thước cache 16K line
- Kích thước của 1 bock = 1 line = 16 word

Giải:

- 3FE9704Ah = 0011 1111 1110 1001 0111 0000 0100 1010 b
- Kích thước cache 16K line = 2^{14} line => L=14 bit
- Kích thước 1 block = 16 word = 2^4 => k = 4 bit
- => Số bít cho trường Tag = 32 – 4 – 14 = 14
- Kết quả là:

	Tag	**Line**	**Word**
Size	14 bit	14 bit	4 bit
Hệ nhị phân	0011 1111 1110 10	01 0111 0000 0100	1010
Hệ 10	4090	5892	10

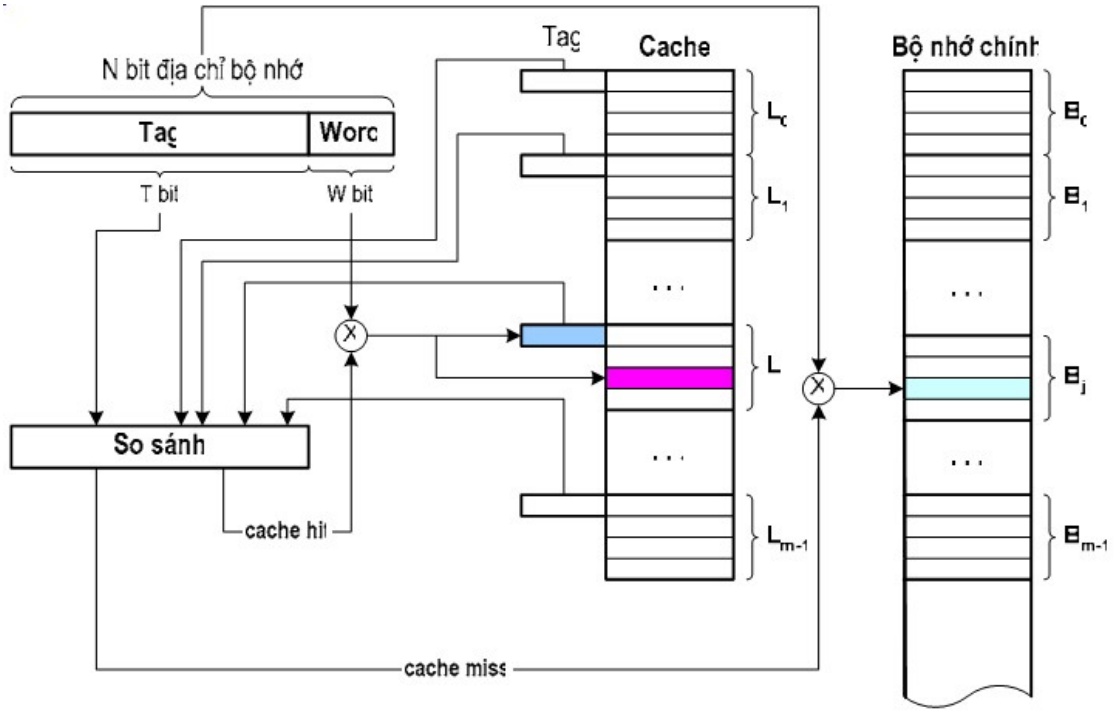
b) Ánh xạ toàn phần

Để khắc phục nhược điểm của phương pháp trực tiếp là, có thể rất nhiều block tranh chấp 1 line trên cache trong kh các vị trí khác bỏ trống, thì phương pháp này cho phép một block trên MainMem được tải vào một vị tri line bất kỳ còn trống trên cache.

Nguyên tắc:

- Mỗi Block có thể nạp vào bất kỳ Line nào của cache.
- Địa chỉ của bộ nhớ chính bao gồm hai trường:
  - Trường Word giống như trường hợp ở trên.
  - Trường Tag dùng để xác định Block của bộ nhớ chính. Tag xác định Block đang nằm ở Line đó

Mô tả:



❖ **Đặc điểm của ánh xạ toàn phần**

Khi CPU phát ra một địa chỉ A gồm N bit trong đó có 2 trường tag + word. Đầu tiên CPU sẽ duyệt từ line 0 cho đến hết, lần lượt kiêm tra trường Tag cua line hiện thời nếu trùng với trường Tag của A tức cache hit thì ô nhớ Word sẽ được tải vào CPU. Trái lại không tìm thấy line nào có tag trùng với Tag của A tức cache miss thì CPU phải truy xuất vào block thứ Tag trên MainMem.

Ưu điểm:

- Hiệu xuất cache tối đa, cache hit cao.

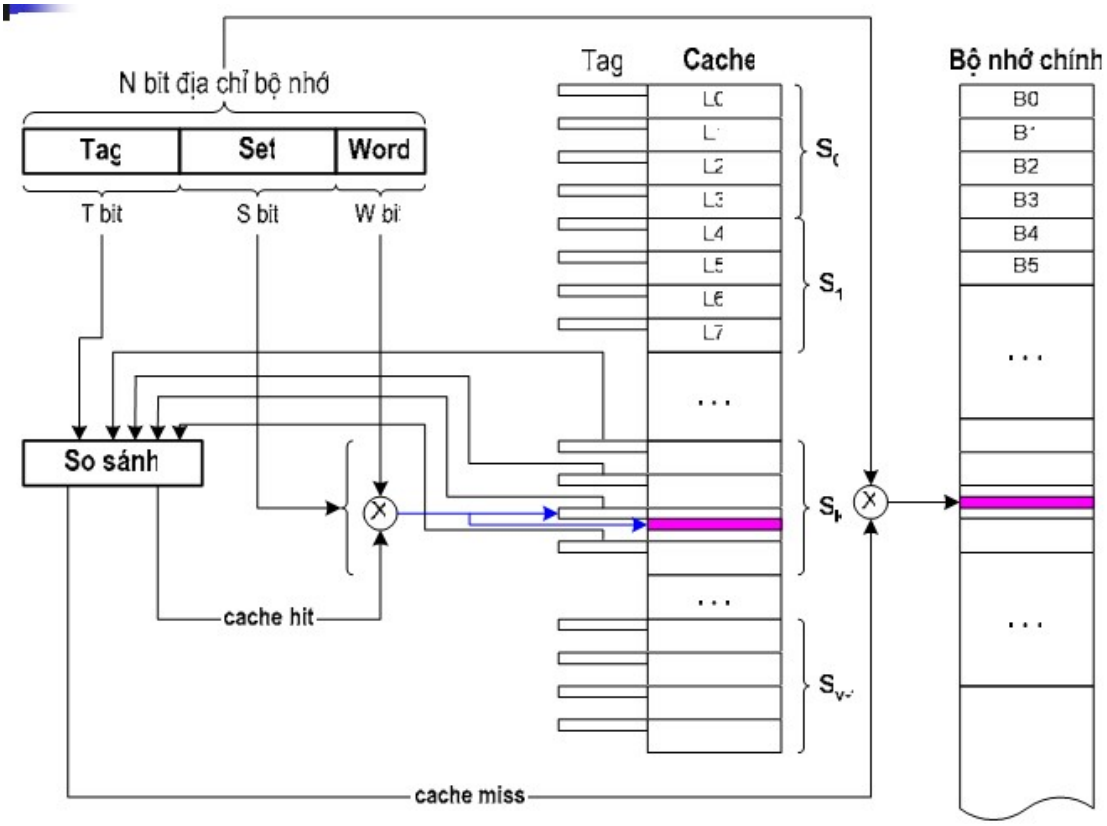
Nhược điểm:

- Tốc độ tìm kiếm chậm, bộ so sánh phức tạp vì phải lấy tất các các tag của cache để so sánh.

c) Ánh xạ liên kết thành bộ

Đây là phương pháp án dụng cacr hai phương pháp trên, trên cache và MainMem nhóm các line hay block lại thành các SET, khi đó một block thuộc một SET thứ S trên MainMem chỉ được nạp vào SET thứ (S mod x) với x là số SET trên cache, còn trong phạm vi một SET thì block đó có thể dặt vào vị trí bất kỳ nào có line đang trông.

Mô tả:



❖ **Đặc điểm của ánh xạ thành bộ**

CPU phát ra một địa chỉ A n bit gồm 3 trường Tag + Set + Word, đầu tiên CPU sẽ truy xuất vào SET thứ (Set mod x) với x là số SET của cache, sau dó lấy trường tag của A so sánh lần lượt với các Tag của các line trong cache, nếu trùng thì Word đượct tải vào CPU, trái lại CPU phải tỉ từ bock thứ Tag trên MainMem.

- Kích thước Block = 2W Word
- Trường Set có S bit dùng để xác định một trong số V = 2S Set
- Trường Tag có T bit: T = N - (W+S)
- Thông thường 2,4,8,16Lines/Set

Ưu điểm:

- Tăng tốc độ tìm kiếm nhớ ứng dụng cơ ánh xạ trực tiếp đối với các SET.
- Giảm được cache Miss do áp dụng ánh xạ toàn phần trong một SET

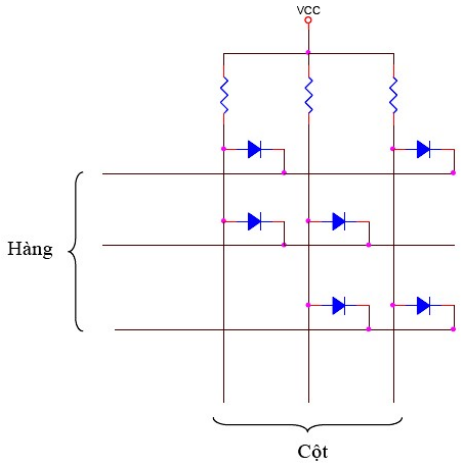
5.4 Bộ nhớ trong

5.4.1 Bộ nhớ ROM

Đây là loại bộ nhớ dùng trong các hãng sãn xuất là chủ yếu. Nó có đặc tính là thông tin lưu trữ trong ROM không thể xoá được và không sửa được, thông tin sẽ

được lưu trữ mãi mãi. Nhưng ngược lại ROM có bất lợi là một khi đã cài đặt thông tin vào rồi thì ROM sẽ không còn tính đa dụng. Ví dụ điển hình là các con "chip" trên motherboard hay là BIOS ROM để vận hành khi máy tính vừa khởi động.

ROM được chế tạo trên một miếng Silic theo một số bước xử lý như quang khắc và khuếch tán để tạo các tiếp xúc bán dẫn dẫn điện một chiều (diode, FET, …). Người thiết kế xác định chương trình muốn ghi vào ROM và thông tin này dùng để điều khiển trong quá trình tạo ROM. Sơ đồ đơn giản về ROM có thể biểu diễn như sau:



ROM đơn giản dùng diode

Giao giữa một hàng và một cột là một vị trí ô nhớ. Nếu vị trí này tồn tại một diode thì sẽ lưu trữ dữ liệu 0, ngược lại thì lưu trữ dữ liệu 1. Khi đọc một ô nhớ nào đó trên một hàng thì bộ giải mã sẽ đặt hàng đó xuống mức 0, các hàng còn lại ở mức logic 1. Nếu giao giữa hàng và cột không có diode (nghĩa là lưu trữ dữ liệu 1) thì giá trị tương ứng trên cột là 1 (do cột nối với Vcc thông qua điện trở R). Nếu có diode (lưu trữ dữ liệu 0) thì diode sẽ phân cực thuận nên điện áp rơi trên diode khoảng 0.7V điện áp trên cột cũng là 0.7V tương ứng với mức logic 0.

Công nghệ chế tạo ROM được sử dụng là lưỡng cực và MOS. Thời gian truy xuất của bộ nhớ lưỡng cực khoảng từ 50 ÷ 90 ns còn của MOS lớn hơn khoảng 10 lần nhưng bộ nhớ MOS có kích thước nhỏ hơn và tiêu thụ năng lượng ít hơn.

❖ **Các công nghệ ROM:**

- EPROM:

Đối với EPROM, dữ liệu có thể ghi vào bằng điện và có thể xóa được. EPROM sử dụng một transistor có cấu trúc FAMOST (Floating gate avalanche injection MOS transistor). Đối với transistor loại này, cấu tạo cũng giống như dạng FET thông thường nhưng trong cực gate tồn tại thêm một cực, gọi là cực nổi

(floating gate). Nếu cực nổi không có điện tích thì transistor này hoạt động như một FET thông thường, nghĩa là nếu cực Gate có điện áp dương thì FET dẫn, cực drain nối với cực source cực drain có mức logic 1. Nếu cực nổi có chứa điện tích thì nó sẽ tạo ra trường điện từ đủ sức ngăn chặn không cho FET dẫn cực drain có mức logic 0.

Quá trình nạp điện tử vào cửa nổi được thực hiện bằng các xung điện có độ rộng khoảng 50 ms và biên độ 20V đặt vào cực gate và cực drain. Điện tử được lưu trữ tại vùng cửa nổi khi xung điện tắt (thời gian lưu trữ ít nhất là 10 năm). Để xóa dữ liệu trên EPROM, ta phải chiếu ánh sáng tử ngoại vào chip nhớ. Các điện tử sẽ hấp thu năng lượng đủ để có thể vượt ra khỏi cực nổi làm cho cực nổi không chứa điện tử toàn bộ EPROM chứa giá trị 1. Do đó, trên các chip EPROM sẽ có một cửa số bằng thạch anh cho phép ánh sáng tử ngoại đi qua.

- EEPROM

Do việc sử dụng cửa sổ thạch anh không tiện lợi nên những năm gần đây đã xuất hiện them chip ROM có thể xóa bằng điện. Cấu tạo của EEPROM cũng giống như EPROM nhưng lúc này có thêm một lớp màng mỏng oxide giữa vùng cực nổi và cực drain cho phép các điện tử di chuyển từ vùng cực nổi sang cực drain khi đặt một điện áp âm. Như vậy, quá trình lập trình cho EEPROM tương tự như EPROM trong khi đó để xóa dữ liệu thì chỉ cần đặt một điện áp -20V vào cực gate và cực drain với thời gian thích hợp (vì nếu thời gian quá dài thì các điện tử sẽ di chuyển thêm sang cực drain làm cho cực nổi sẽ có điện tích dương FET sẽ không hoạt động như bình thường).

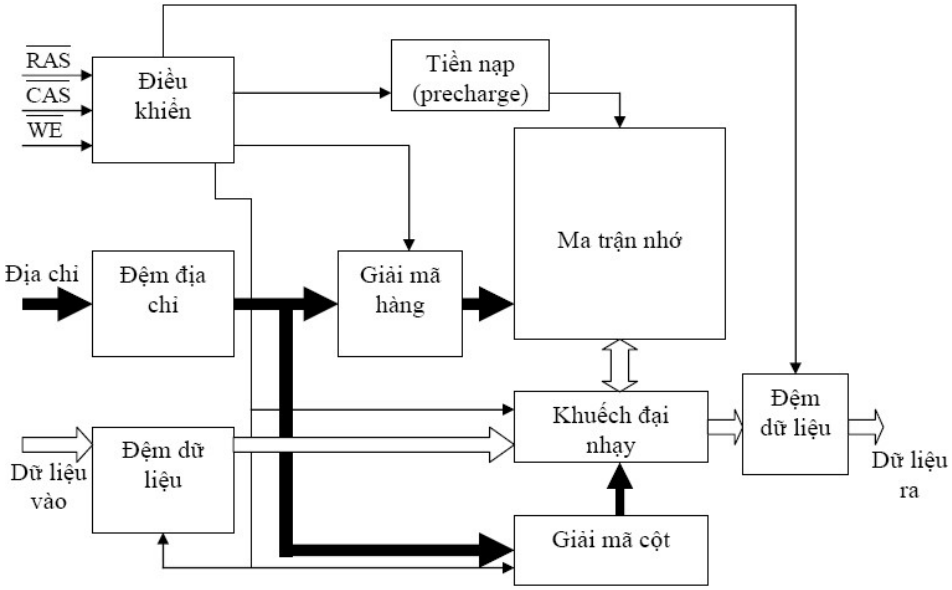
5.4.2 Bộ nhớ RAM

RAM là thế hệ kế tiếp của ROM, cả RAM và ROM đều là bộ nhớ truy xuất ngẫu nhiên, tức là dữ liệu được truy xuất không cần theo thứ tự. Tuy nhiên ROM chạy chậm hơn RAM rất nhiều. Thông thường ROM cần trên 50ns để xử lý dữ liệu trong khi đó RAM cần dưới 10ns.

a) DRAM

❖ **Cấu tạo của DRAM**

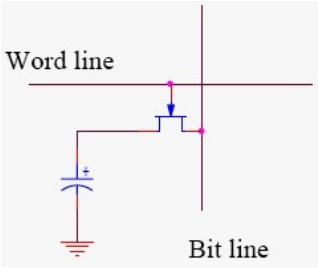
Địa chỉ xác định ô nhớ chia thành 2 phần: địa chỉ hàng và cột. Hai địa chỉ này được đưa lần lượt vào bộ đệm. Quá trình dồn kênh địa chỉ điều khiển bằng các tín hiệu RAS (Row Access Strobe) và CAS (Column Access Strobe). Bộ điều khiển nhớ của CPU phải thực hiện 3 công việc sau: chia địa chỉ từ CPU thành các địa chỉ hàng và cột; tích cực các chân RAS , CAS và WE một cách chính xác; truyền và



nhận các dữ liệu đọc, ghi.

Sơ đồ tổ chức của một DRAM:

Một ô nhớ của DRAM có thể biểu diễn như hình vẽ:



❖ **Làm tươi DRAM**

Điện tích trên tụ điện bị giảm theo thời gian do chúng phóng qua FET và lớp điện môi oxide làm cho dữ liệu có thể bị mất. Do đó, tụ điện phải được nạp lại một cách tuần hoàn (refresh), thông thường khoảng từ 1 ÷ 16 ms tùy theo loại RAM. Có 3 phương pháp refresh:

- **Chỉ tác động RAS:**

Tạo chu kỳ đọc giả (dummy read) để làm tươi ô nhớ. Trong chu kỳ giả này, tín hiệu RAS tích cực và địa chỉ hàng được đưa vào DRAM nhưng CAS bị cấm nên không thể truyền dữ liệu ra ngoài được. Để làm tươi toàn bộ bộ nhớ thì tất cả các

địa chỉ phải được cấp lần lượt. Nhược điểm của phương pháp này là cần phải dùng mạch logic hay một chương trình để làm tươi. Trong máy tính, điều này thực hiện bằng kênh 0 của DMAC 8237, tác động định kỳ bằng bộ đếm 1 của PIT 8253.

- **Tác động CAS trước RAS:**

DRAM có một mạch logic làm tươi của riêng nó với một bộ đếm địa chỉ. Tín hiệu CAS sẽ tích cực trong một khoảng thời gian trước khi RAS tích cực. Địa chỉ làm tươi được phát ra bên trong bằng bộ đếm địa chỉ mà không cần mạch logic bên ngoài. Sau mỗi chu kỳ làm tươi, bộ đếm địa chỉ sẽ tự động tăng lên 1 để chỉ địa chỉ kế tiếp.

- ***Ẩn*:**

Chu kỳ làm tươi được chứa sau chu kỳ đọc bộ nhớ. Tín hiệu CAS giữ nguyên mức thấp trong khi chỉ có RAS lên mức cao. Ở đây, bộ đếm địa chỉ cũng tự phát ra địa chỉ làm tươi.Việc đọc dữ liệu trong chu kỳ đọc cũng có thể thực hiện ngay cả khi đang thực hiện chu kỳ làm tươi. Phương pháp này sẽ tiết kiệm được thời gian do thời gian làm tươi thường ngắn hơn so với thời gian đọc.

b) SRAM

Đối với SRAM, nội dung ô nhớ vẫn giữ nguyên khi chưa mất nguồn cung cấp mà không cần phải tốn thời gian làm tươi ô nhớ. Do điện áp chênh lệch lớn nên thời gian xử lý khuếch đại sẽ nhỏ hơn trong DRAM (thời gian truy xuất của DRAM là khoảng 1ns trong khi của DRAM khoảng 40ns). Từ đó, SRAM không thực hiện phân kênh địa chỉ hàng và cột (điều này sẽ làm giảm thời gian truy xuất). Sau khi dữ liệu ổn định, bộ giải mã cột chọn cột phù hợp và đưa dữ liệu đến bộ đệm ngõ ra.

Các khác biệt của SRAM so với DRAM:

- Tốc độ của SRAM lớn hơn DRAM do không phải tốn thời gian refresh..
- Chế tạo SRAM tốn kém hơn DRAM nên thông thường sử dụng DRAM để hạ giá thành sản phẩm.

<u>Một số loại RAM</u>

- **FPM - DRAM (Fast Page Mode DRAM)**

DRAM chuẩn được truy cập qua một kỹ thuật được gọi là kỹ thuật phân trang (paging). Việc truy cập thông thường bằng cách chọn địa chỉ hàng và địa chỉ cột mất rất nhiều thời gian (giải mã hai bước). Phân trang cho phép truy nhập tất cả dữ liệu trong một hàng nhanh hơn bằng cách giữ nguyên địa chỉ các hàng và chỉ thay đổi giá trị cột. Bộ nhớ sử dụng kỹ thuật này gọi là bộ nhớ đánh số trang Page

mode hay Fast page Mode (FPM). Ngoài ra còn một số cách gọi khác là: Column hay Nibble Mode Memory.

- **EDO - DRAM (Extended Data Out DRAM)**

Một loại bộ nhớ khác gọi là EDO RAM được dùng trong các máy Pentium từ năm 1995. EDO là bộ nhớ kiểu FPM đã được sửa lỗi hay còn được gọi là Hyper Page Mode. Hãng Micron Technology đã phát minh ra và lấy bằng sáng chế EDO RAM. Bộ nhớ EDO gồm những Chip được sản xuất đặc biệt cho phép chồng các lần truy nhập liên tiếp. Cái tên Extended Data Out xuất phát từ thực tế là không giống như FPM, các trình điều khiển dữ liệu ra (data output driver) trên chip không bị tắt khi mạch điều khiển bộ nhớ xoá địa chỉ cột để bắt đầu chu kỳ tiếp theo. Điều này cho phép chu kỳ tiếp theo chồng lên chu kỳ trước và tiết kiệm thời gian được khoảng 10ns một chu kỳ.

Để sử dụng bộ nhớ EDO, chipset của mainboard phải hỗ trợ bộ nhớ đó. Hầu hết các mainboard về sau kể từ loại Intel 430 FX (Trion) công bố năm 1995 đều hỗ trợ EDO RAM. Do chi phí cho chip bộ nhớ EDO tương đương chip chuẩn và các mainboard Intel đều hỗ trợ EDO nên thị trường máy tính tập trung vào bộ nhớ EDO.

Rất lý tưởng khi sử dụng EDO RAM cho các máy có tốc độ bus dưới 66 MHz, tức là các máy trên thị trường trước năm 1998. Từ năm 1998, thị phần của EDO RAM đã giảm do bộ nhớ chính nhanh hơn và mới hơn là SDRAM (Synchronuos DRAM) đã trở thành bộ nhớ chuẩn của các máy PC mới.

- **BDEO-DRAM (Burst Extended Data Out DRAM)**

Là thế hệ sau của EDO DRAM, dùng kỹ thuật đường ống (pipeline) để rút ngắn thời gian dò địa chỉ.

- **SDRAM (Synchronous DRAM)**

Đây là một loại RAM có nguyên lý chế tạo khác hẳn với các loại RAM trước. Đồng bộ (synchronous) là một khái niệm rất quan trọng trong lĩnh vực số. RAM hoạt động do một bộ điều khiển xung nhịp (clock memory), dữ liệu sẽ được truy xuất hay cập nhật mỗi khi clock chuyển từ logic 0 sang 1, đồng bộ có nghĩa là ngay lúc clock nhảy từ logic 0 sang 1 chứ không hẳn là chuyển sang logic 1 hoàn toàn (tác động bằng cạnh xung). Do kỹ thuật này, SDRAM và các thế hệ sau có tốc độ cao hơn hẳn các loại DRAM trước, đạt tốc độ 66, 100, 133 MHz.

- **DDR SDRAM (Double Data Rate SDRAM)**

Đây là loại bộ nhớ cải tiến từ SDRAM. Nó nhân đôi tốc độ truy cập của SDRAM bằng cách dùng cả hai quá trình đồng bộ khi clock chuyển từ logic 0 sang

1 và từ logic 1 sang 0 (dùng cả cạnh âm và cạnh dương). Loại RAM này được CPU Intel và AMD hỗ trợ, tốc độ vào khoảng 266 MHz. (DDR-SDRAM đã ra đời trong năm 2000)

- **DRDRAM (Direct Rambus DRAM)**

Hệ thống Rambus (tên hãng chế tạo) có nguyên lý và cấu trúc chế tạo hoàn toàn khác loại SDRAM truyền thống. Bộ nhớ sẽ được vận hành bởi một hệ thống phụ gọi là kênh truyền Rambus trực tiếp (direct Rambus channel) có độ rộng bus 16 bit và một xung clock 400MHz (có thể lên tới 800MHz). Theo lý thuyết thì cấu trúc mới nầy sẽ có thể trao đổi dữ liệu với tốc độ 400MHz x 16 bit = 400MHz x 2 bytes = 800 MBps. Hệ thống Rambus DRAM cần một chip serial presence detect (SPD) để trao đổi với motherboard.

Ta thấy kỹ thuật mới nầy dùng giao tiếp 16 bit, khác hẳn với cách chế tạo truyền thống là dùng 64 bit cho bộ nhớ nên kỹ thuật Rambus cho ra đời loại chân RIMM (Rambus Inline Memory Module), khác so với bộ nhớ truyền thống. Loại RAM này chỉ được hỗ trợ bởi CPU Intel Pentum IV, tốc độ vào khoảng 400 – 800 MHz

- **SLDRAM (Synchronous - Link DRAM)**

Là thế hệ sau của DRDRAM, thay vì dùng kênh Rambus trực tiếp 16 bit và tốc độ 400MHz, SLDRAM dùng bus 64 bit chạy với tốc độ 200MHz. Theo lý thuyết thì hệ thống mới có thể đạt được tốc độ 200MHz x 64 bit = 200MHz x 8 bytes = 1600 MBps, tức là gấp đôi DRDRAM. Điều thuận tiện là là SLDRAM được phát triển bởi một nhóm 20 công ty hàng đầu về vi tính cho nên nó rất da dụng và phù hợp nhiều hệ thống khác nhau.

- **VRAM (Video RAM)**

Khác với bộ nhớ trong hệ thống, do nhu cầu về đồ hoạ ngày càng cao, các hãng chế tạo card đồ họa đã chế tạo VRAM riêng cho video card của họ mà không cần dùng bộ nhớ của hệ thống chính. VRAM chạy nhanh hơn vì ứng dụng kỹ thuật Dual Port nhưng đồng thời cũng đắt hơn rất nhiều.

- **SGRAM (Synchronous Graphic RAM)**

Là sản phẩm cải tiến của VRAM, nó sẽ đọc và viết từng block thay vì từng mảng nhỏ.

IV. Bài tập củng cố kiến thức

5.1 Dựa vào chức năng, cách tổ chức và truy nhập có thể phân chia bộ nhớ thành những loại chính nào? Đặc trưng cơ bản của từng loại bộ nhớ đó

5.2 Mục đích của việc làm tươi cho các RAM là gì?

5.3 Mô tả quy trình đọc bộ nhớ của bộ xử lý.

5.4 Bộ nhớ trong VXL 8086 được chia ra thành bao nhiêu vùng nhớ, và đó là các vùng nhớ nào? Những thanh ghi nào được sử dụng để quản lý địa chỉ cho các vùng nhớ đó?

5.5 Một hệ thống Cache ánh xạ trực tiếp có kích thước là 512 Line được thiết kế gồm các block, mỗi block có kích thước bằng 8word. Vậy một word có địa chỉ: A4573 sẽ được lưu vào Line nào, số Tag là bao nhiêu?

5.6 Hãy xác giá trị của dữ liệu tại địa chỉ 76359 được cho dưới đây là bao nhiêu (biểu diễn dưới dạng số Hexa).

Biết rằng bộ nhớ sử dụng cơ chế ánh xạ liên kết đầy đủ, bảng dưới đây biểu diễn 5 Line trong một Cache với kích thước của một khối bằng 8.

	Địa chỉ Word trong khối							
Tag	000	001	010	011	100	101	110	111
10011011011000101	10	65	BA	0F	C4	19	6E	C3
00111000011010101	21	75	CB	80	D5	2A	7F	B5
10111100010111001	32	87	DC	91	E6	3B	F0	A6
01110110001101011	43	98	ED	A2	F7	4C	E1	97
00111100100111000	54	9A	FE	B3	08	5D	D2	88

5.7 A set- associative cache consists of 64 lines, or slots, divided into four-line sets. Main memory contains 4K blocks of 128 words each. Show the format of main memory addresses.

5.8 A two - way set - associative cache has lines of 16 bytes and a total size of 8 kbytes. The 64 - Mbyte main memory is byte addressable. Show the format of main memory addresses

5.9 Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

a. How is a 16 - bit memory address divided into tag, line number, and byte number?

b. Into what line would bytes with each of the following addresses be stored?

0001 0001 0001 1011

1100 0011 0011 0100

1101 0000 0001 1101

1010 1010 1010 1010

c. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?

d. How many total bytes of memory can be stored in the cache?

e. Why is the tag also stored in the cache?

5.10 Consider a memory system that uses a 32 - bit address to address at the byte level, plus a cache that uses a 64 -byte line size.

a. Assume a direct mapped cache with a tag field in the address of 20 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

b. Assume an associative cache. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in cache, size of tag.

c. Assume a four - way set-associative cache with a tag field in the address of 9 bits. Show the address format and determine the following parameters: number of addressable units, number of blocks in main memory, number of lines in set, number of sets in cache, number of lines in cache, size of tag.

CHƯƠNG 6 CẤP HỆ ĐIỀU HÀNH

I. Mục đích:

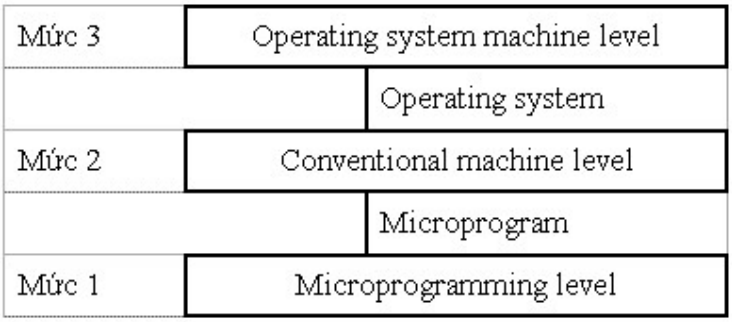
- Giúp sinh viên nắm được các vấn đề cơ bản của mức máy hệ điều hành
- Năm được về bộ nhớ ảo
- Nắm được cơ chế và cách thực hiện việc phân trang

II. Yêu cầu: Yêu cầu sinh viên học lý thuyết và làm bài tập trong sách bài tập

III. Nội dung: Nội dung chương 6 được trình bày trong 3 tiết lý thuyết

6.1 Giới thiệu mức máy hệ điều hành

Trình thông dịch chạy trên máy mức 1 có thể thông dịch các chương trình được viết bằng ngôn ngữ máy mức 2. Trình thông dịch chạy trên máy mức 2 có thể thông dịch các chương trình được viết bằng ngôn ngữ máy mức 3. Vì lý do lịch sử trình thông dịch chạy trên máy mức 2 để hỗ trợ cho máy mức 3 được người ta gọi là hệ điều hành - operating system. Chúng ta gọi mức 3 là mức máy hệ điều hành.



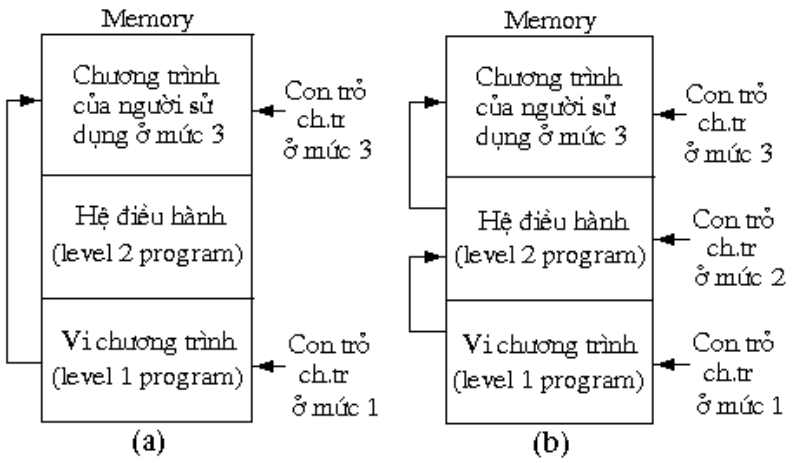
Hình 6-1 Mức 2 và 3 được hỗ trợ bởi phần mềm

Mức máy HĐH đã tiến hoá dần lên từ mức máy thông thường:

- Hầu hết các chỉ thị của mức máy HĐH cũng có ở mức máy thông thường
- Các chỉ thị mức 3 khác, được gọi là các chỉ thị Mức máy HĐH - OSML (Operating System Machine Level).

Nếu bắt HĐH thông dịch tất cả các chỉ thị của mức 3: không hiệu quả & không cần thiết

- Các chỉ thị mức 3 thông thường: thông dịch trực tiếp bằng vi chương trình (Hình a)
- Các chỉ thị OSML: HĐH thông dịch (Hình b).



Hình 6-2 Chỉ thị được thông dịch bởi vi chương trình

Hầu hết các HĐH cho các máy tính lớn là các hệ thống đa chương trình (multiprogramming systems): HĐH không phải chỉ hỗ trợ cho một máy ảo mức 3 mà là một vài máy ảo mức 3 một cách song song. Nhiệm vụ quan trọng hàng đầu của HĐH là giải quyết việc quản lý mọi máy ảo chứ không phải là vấn đề thông dịch các chỉ thị OSML.

Phân loại hệ thống:

- Hệ thống chia sẻ thời gian (time-sharing): Mỗi máy ảo này được nối với một terminal ở xa
- Hệ thống đa chương trình theo lô (batch multiprogramming): Nếu không có một terminal ở xa
- Các dạng lai

Khi nghiên cứu HĐH, cần tập trung vào 3 chủ đề chính, quan trọng nhất sau đây:

1. Bộ nhớ ảo (virtual memory), đó là kỹ thuật làm cho máy thể hiện như có một bộ nhớ lớn hơn bộ nhớ mà nó thực sự có.
2. Vào/ra file (file I/O), đây là một khái niệm có mức độ trừu tượng cao hơn khái niệm về các chỉ thị vào/ra.
3. Xử lý song song - làm thế nào để nhiều quá trình có thể được đồng thời thi hành ở mức 3.

6.2 Bộ nhớ ảo

Thời kỳ đầu của kỷ nguyên máy tính, bộ nhớ của máy tính thường nhỏ, các nhà lập trình thường phải dành phần lớn thời gian nhằm làm sao cho chương trình càng bé càng tốt.

Một giải pháp truyền thống, được sử dụng rộng rãi trong nhiều năm: sử dụng bộ nhớ phụ + kỹ thuật overlay (mỗi overlay có thể nằm lọt trong bộ nhớ). Kỹ thuật này đòi hỏi nhiều công sức của người lập trình cho việc quản lý overlay.

Bộ nhớ ảo (Virtual Memory):

- Được một nhóm nhà khoa học ở Manchester, nước Anh, đề xuất năm 1961.
- Phương pháp thực hiện quá trình overlay một cách tự động.
- Đầu những năm 1970 bộ nhớ ảo đã trở nên thông dụng trong các hầu hết các máy tính.

Ngày nay thậm chí cả các bộ vi xử lý, trong đó có 80386 của Intel và 68030 Motorola cũng có các hệ thống bộ nhớ ảo rất tinh vi.

6.2.1 Việc phân trang – Paging

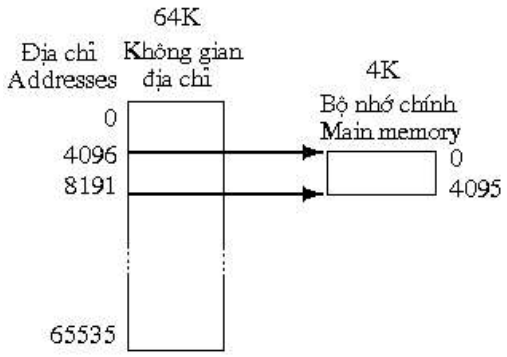
Xét một thí dụ: một máy tính trong đó các chỉ thị có trường địa chỉ 16 bit và có bộ nhớ 4096 word.

- Một chương trình chạy trên máy tính này có thể đánh địa chỉ 65536 word bộ nhớ, chứ không phải 4096.
- Khi còn chưa phát minh ra bộ nhớ ảo, phải phân biệt 2 miền địa chỉ:

0..4095: khả dụng

4096.. 65535: không khả dụng

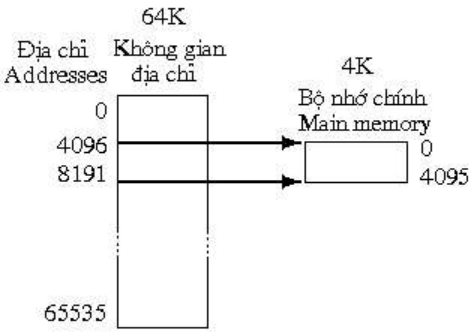
Ý tưởng: làm tách biệt các khái niệm về không gian địa chỉ và các vị trí nhớ:



- Vào bất cứ thời điểm nào 4096 word bộ nhớ có thể được truy cập trực tiếp, nhưng chúng không nhất thiết phải tương ứng với các địa chỉ từ 0..4095.
- Thí dụ chúng ta có thể "bảo" ho máy tính rằng từ lúc này trở đi, mỗi khi truy cập địa chỉ:
  - 4096: sử dụng word 0
  - 4097: sử dụng word 1
  - 8191: sử dụng word 4095 v.v.

Điều gì sẽ xảy ra nếu chương trình nhảy tới một địa chỉ nằm trong khoảng 8192..12287?

- Nếu máy tính không có bộ nhớ ảo: chương trình này sẽ gây ra một bẫy lỗi, chẳng hạn "Nonexistent memory referenced" rồi bị dừng.
- Nếu máy tính có bộ nhớ ảo: một dãy các bước sau có thể sẽ lần lượt diễn ra:



1. Nội dung của bộ nhớ chính được cất vào bộ nhớ phụ.
2. Các word 8192..12287 đang nằm trong bộ nhớ phụ được nạp vào bộ nhớ chính.
3. Ánh xạ địa chỉ sẽ được thay đổi để ánh xạ các địa chỉ 8192..12287 vào các vị trí nhớ 0..4095.
4. Chương trình tiếp tục như không có điều gì bất thường xảy ra.

Các khái niệm cần phân biệt:

0..4095: ***không gian địa chỉ vật lý***

4096..: ***không gian địa chỉ ảo*** (virtual address space)

Kỹ thuật thực hiện overlay tự động được gọi là phân trang - paging, các đoạn chương trình được đọc vào bộ nhớ chính từ bộ nhớ phụ được gọi là các trang.

Cơ chế phân trang có thể coi là trong suốt đối với người lập trình.

- Khi lập trình không cần phải biết đến sự tồn tại của bộ nhớ ảo.
- Có thể coi như đang lập trình cho một máy tính có một bộ nhớ chính rất lớn (ảo giác)

Paging so với segmentation: trong phương pháp Segmentation, người lập trình phải ý thức về sự tồn tại của các phân đoạn.

6.2.2 Thực hiện việc phân trang

Một đòi hỏi cốt yếu của bộ nhớ ảo là phải có bộ nhớ phụ để chứa toàn bộ chương trình.

Một số quy ước:

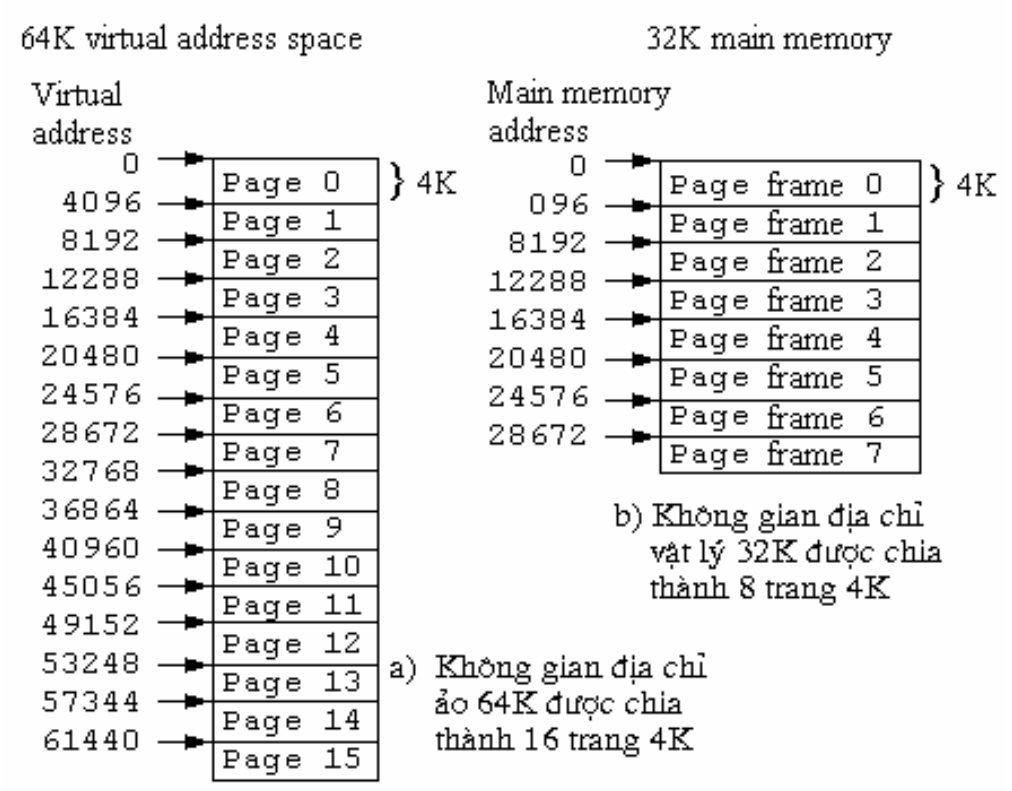
- Bản copy của chương trình trong bộ nhớ phụ là bản gốc
- Phần của chương trình thỉnh thoảng được đưa vào bộ nhớ chính là các bản copy

Một công việc tự nhiên và quan trọng là cập nhật cho bản gốc, mọi thay đổi xảy ra với bản copy trong bộ nhớ chính, cuối cùng cần phải được phản ánh vào

trong bản gốc. Không gian địa chỉ ảo được chia thành các trang có kích thước bằng nhau, nằm trong khoảng 512..4096. Không gian địa chỉ vật lý cũng được chia thành các mảnh, được gọi là khung trang (page frame), có cùng kích thước với trang. Trong các thiết kế thực, bộ nhớ chính của các máy tính lớn có thể có hàng chục, hàng trăm hay thậm chí hàng nghìn khung trang.

Hình 6-3 một thí dụ minh hoạ:

- Không gian địa chỉ ảo 64K được chia thành các frame kích thước 4K.
- Không gian địa chỉ vật lý 32K được chia thành các khung trang kích thước 4K.
- Để triển khai thực hiện được bộ nhớ ảo ở mức 2, cần phải sử dụng page table có 16 word, để theo dõi trạng thái sử dụng 16 trang của bộ nhớ ảo.

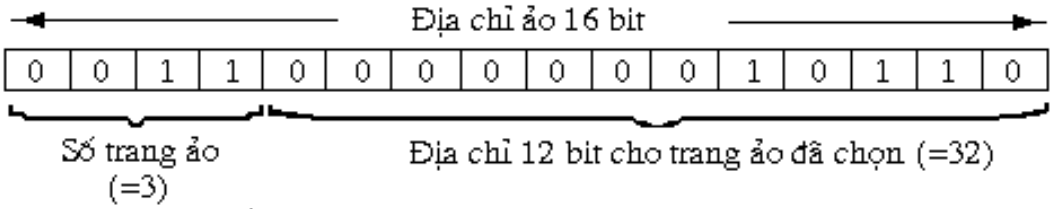


Hình 6-3 Cách thức chia không gian địa chỉ

Trong hình Hình 6-4(a): Khi một chương trình truy cập bộ nhớ, đầu tiên nó sẽ sinh ra một địa chỉ 16 bit (= 3016H = 12310):

- 4 bit cao làm số trang ảo
- 12 bit thấp làm địa chỉ trong trang ảo đã được chọn

12310 là địa chỉ 22 của trang 3.



a) Một địa chỉ ảo bao gồm số trang ảo 4-bit và địa chỉ tương đối (offset) 12 bit

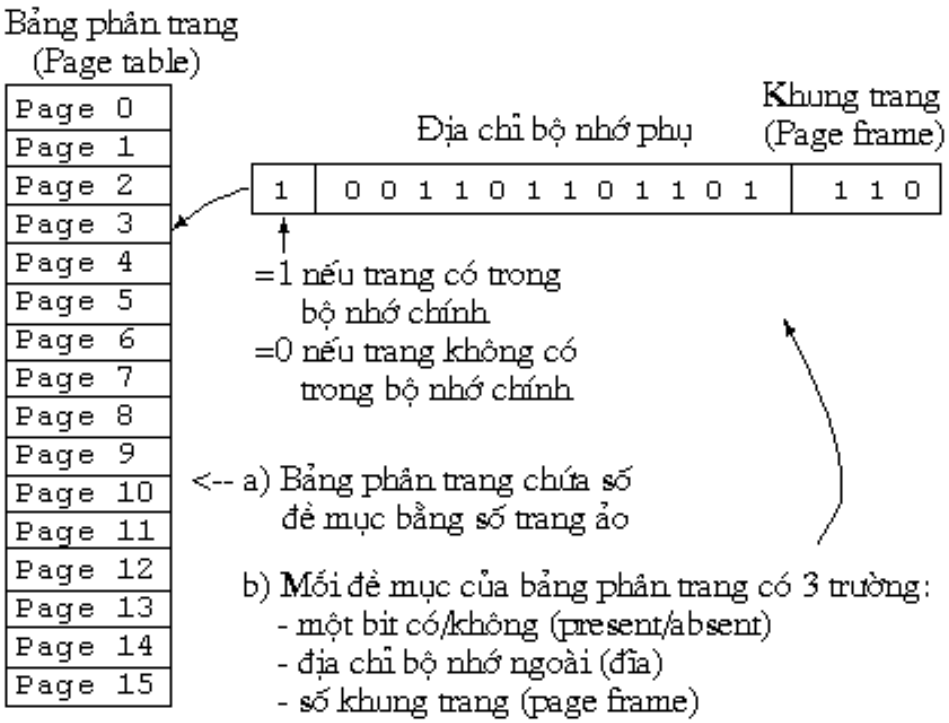
b) Số trang và địa chỉ ảo của chúng --->

Trang	Địa chỉ ảo
0	0 — 4095
1	4096 — 8191
2	8192 — 12287
3	12288 — 16383
4	16384 — 20479
5	20480 — 24575
6	24576 — 28671
7	28672 — 32767
8	32768 — 36863
9	36864 — 40959
10	40960 — 45055
11	45056 — 49151
12	49152 — 53247
13	53248 — 57343
14	57344 — 61439
15	61440 — 65535

Hình 6-4 Ví dụ về địa chỉ ảo

Trong hình Hình 6-4(**b):** Mối liên hệ giữa trang và địa chỉ ảo. Nếu địa chỉ ảo 0 của trang 3 là tại địa chỉ vật lý 12288, thì địa chỉ ảo 22 của trang 3 phải là tại địa chỉ vật lý 12310.

- Với một địa chỉ ảo cần truy cập, sẽ tính được trang ảo, sau đó HĐH phải tìm ra được trang ảo đó nằm ở đâu:
- Có 9 khả năng xảy ra:
    - 8: nằm ở một trong số 8 khung trang trong bộ nhớ chính
    - 9th: không có trong bộ nhớ chính mà nằm ở đâu đó trong bộ nhớ phụ.



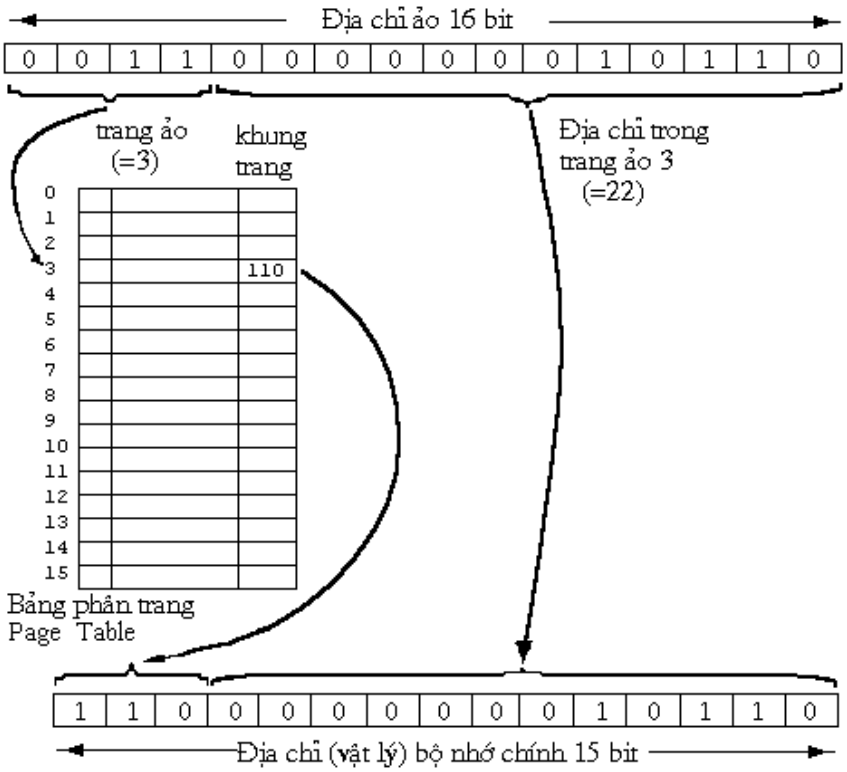
Hình 6-5 Ví dụ về một bảng phân trang

Cách tạo địa chỉ bộ nhớ chính (bộ nhớ vật lý) từ địa chỉ ảo:

Từ đ/c ảo tính được trang ảo (+ địa chỉ trong trang ảo). Từ trang ảo, kiểm tra page table sẽ biết trang ảo có trong bộ nhớ chính không. Giả sử có: trường khung trang 3 bit sẽ chỉ ra trang nằm ở đâu.

- 3 bit này sẽ được nạp vào 3 bit bên trái nhất của MAR
- 12 bit thấp của MAR được nạp địa chỉ trong trang ảo.

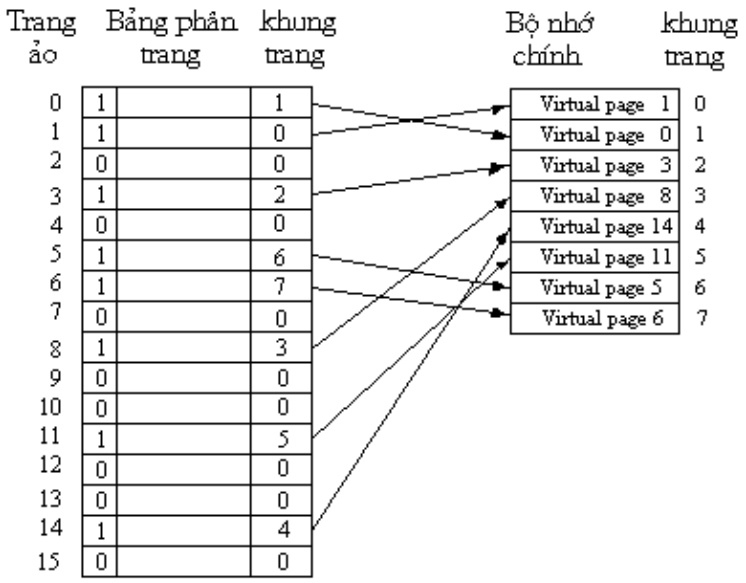
Do đó, ta nhận được 15 bit, chính là địa chỉ cần cho bộ nhớ chính dung lượng 32K.



Hình 6-6 Cách tạo ra địa chỉ bộ nhớ chính từ địa chỉ ảo

Minh hoạ ánh xạ từ không gian chỉ ảo lên không gian bộ nhớ chính (vật lý):

- Trang ảo số 0 nằm ở khung trang số 1.
- Trang ảo số 1 nằm ở khung trang số 0.
- Trang ảo số 2 không nằm trong bộ nhớ chính v.v.



Hình 6-7 Ánh xạ từ không gian địa chỉ ảo lên khung trang bộ nhớ chính có 8 khung trang

Nếu HĐH phải chuyển đổi mỗi địa chỉ ảo của chỉ thị máy mức 3 thành địa chỉ thực thì một máy mức 3 có bộ nhớ ảo có thể chạy chậm hơn một máy mức 3 không có bộ nhớ ảo rất nhiều lần và như vậy toàn bộ ý tưởng sẽ là không có ý nghĩa thực tế. Để tăng tốc độ việc chuyển đổi địa chỉ ảo thành địa chỉ vật lý, có 2 cách:

- Bảng phân trang thường được giữ trong các thanh ghi phần cứng đặc biệt, việc chuyển đổi được thực hiện trực tiếp bằng phần cứng. Cách này đòi hỏi các chi phí phần cứng.
- Bảng phân trang trong các thanh ghi tốc độ cao và dùng vi chương trình thực hiện chuyển đổi bằng lập trình trực tiếp đối với các thanh ghi. Tuỳ thuộc vào kiến trúc của mức vi chương trình, chuyển đổi bằng cách này có thể gần nhanh bằng cách chuyển đổi trực tiếp bằng phần cứng mà không đòi hỏi phải có các mạch điện đặc biệt.

6.2.3 Phương pháp cấp trang khi có yêu cầu và Mô hình tập làm việc

Trong thực tế bộ nhớ chính nói chung không đủ lớn để chứa tất cả các trang. Sự truy cập tới một địa chỉ thuộc một trang mà trang đó không có trong bộ nhớ chính được gọi là lỗi trang - page fault.

Khi lỗi trang xảy ra, hệ điều hành cần phải đọc trang được yêu cầu từ bộ nhớ phụ vào bộ nhớ chính, nhập vị trí nhớ vật lý mới của nó vào bảng phân trang và sau đó lặp lại chỉ thị đã gây ra lỗi trang.

Phương pháp Cấp trang khi có yêu cầu

Với máy có bộ nhớ ảo, có thể khởi động một chương trình ngay cả khi không có phần nào của chương trình này nằm trong bộ nhớ chính, chỉ cần thiết lập cho bảng phân trang để chỉ ra là mọi trang ảo nằm trong bộ nhớ phụ chứ không phải trong bộ nhớ chính.

Khi CPU lấy chỉ thị đầu tiên, lập tức nó gặp một lỗi trang, điều này làm cho trang có chứa chỉ thị đầu tiên được nạp vào và bảng phân trang được cập nhập thông tin. Sau đó chỉ thị đầu tiên có thể bắt đầu được thực hiện.

Nếu chỉ thị đầu tiên có hai địa chỉ mà chúng lại nằm ở các trang khác với trang chứa chính chỉ thị, thì lại xảy ra hai lỗi trang nữa và sẽ lại có thêm hai trang được nạp vào bộ nhớ chính trước khi chỉ thị này được thi hành.

Các chỉ thị tiếp theo của chương trình cũng có thể gây ra các lỗi trang nữa và quá trình trên lại tiếp diễn.v.v.

Phương pháp vận hành bộ nhớ ảo như thế được gọi là Cấp trang khi có yêu cầu - demand paging, các trang chỉ được nạp vào bộ nhớ chính khi xuất hiện một yêu cầu thực sự về trang, chứ không phải là được nạp từ trước.

Việc xem xét có sử dụng phương pháp cấp trang khi có yêu cầu hay không chỉ liên quan tới lúc bắt đầu chạy một chương trình lần đầu tiên. Khi chương trình này đã chạy được một lúc rồi, thì các trang cần đến đã được tập hợp trong bộ nhớ chính.

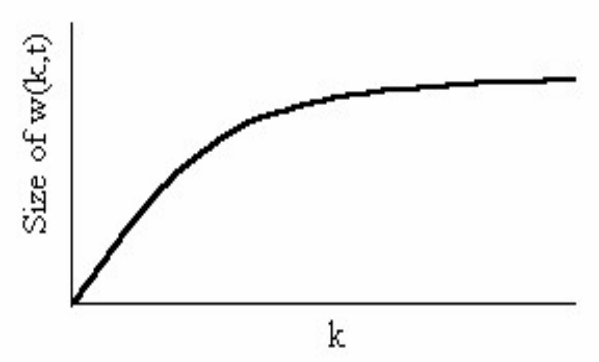
Nhược điểm: Có thể làm giảm hiệu suất của hệ thống rất trầm trọng:

- Nếu máy tính hoạt động trong chế độ chia sẻ thời gian và NSD được chuyển đổi kiểu quay vòng (thí dụ mỗi chương trình của NSD được cho chạy 100ms rồi tạm dừng để cho chạy một chương trình khác v.v.) thì từng chương trình sẽ được khởi động lại nhiều lần trong quá trình chạy của nó.
- Bởi vì mỗi chương trình có một bản đồ bộ nhớ duy nhất và bản đồ này sẽ bị thay đổi khi các chương trình được chuyển đổi ◊ page fault liên tục.

Phương pháp Mô hình tập làm việc

Cơ sở của phương pháp: Quan sát cho thấy hầu hết các chương trình không truy cập không gian địa chỉ của chúng đồng đều như nhau mà có khuynh hướng tập trung vào một số ít các trang.

Tập làm việc - working set: tập bao gồm tất cả các trang được sử dụng bởi k lần truy cập bộ nhớ gần đó nhất; w(k,t):



- w(k,t) là một hàm đơn điệu tăng của k.
- Tuy nhiên w(k,t) có cận trên (khi k cứ tăng dần) vì một chương trình không thể tham khảo nhiều trang hơn số trang của không gian địa chỉ của nó.
- Tồn tại một miền rộng các giá trị của k trong đó working set là không thay đổi.

Phương pháp Mô hình tập làm việc: trong khi còn đang thực hiện một chương trình, nạp sẵn tậplàm việc của một chương trình sắp được gọi thực hiện, dựa trên tập làm việc của lần chạy trước đó.

Ưu điểm:

- Khi chương trình được cho chạy trở lại, nó sẽ không sinh ra ngay lập tức một số lớn các lỗi trang làm lãng phí thời gian.
- Người ta đã biết rằng thời gian cần để đọc một trang từ đĩa vào bộ nhớ chính thường bằng hoặc lớn hơn thời gian thực hiện 20.000 chỉ thị.

Nhược điểm:

- Nếu tập làm việc được nạp "nhầm", chương trình vẫn phải đợi cho đến khi các trang mà nó cần được nạp vào.
- Với phương pháp cấp trang khi có yêu cầu: thì tình huống này không thể xảy ra.

Hiện nay người ta vẫn còn tranh luận về phẩm chất của cả 2 chiến lược này.

6.3 Chỉ thị vào/ra ảo

Nói chung các thỉ thị mức 1 (Vi chương trình) và các chỉ thị mức 2 (Máy thông thường) khác nhau nhiều. Trái lại, tập chỉ thị mức 3 chứa:

- Chứa hầu hết các chỉ thị của mức 2, riêng các chỉ thị dễ gây hỏng cho các máy mức dưới được loại bỏ.

- Được bổ sung thêm một số ít chỉ thị mới nhưng quan trọng: Input/Output, v.v.

Có hai nguyên nhân chính dẫn đến sự khác nhau này:

- Đảm bảo an ninh cho hệ thống
- Tạo thuận lợi cho người lập trình khi thực hiện các thao tác I/O.

6.3.1 Các chỉ thị vào/ra ảo đối với các file tuần tự

File kiểu tuần tự không đánh địa chỉ được.

Một cách để tổ chức vào/ra ảo là quan niệm dữ liệu ở bộ nhớ phụ sẽ được đọc hoặc ghi là một dãy các bản ghi logic, các bản ghi có thể có chiều dài cố định hoặc chiều dài thay đổi được. Một dãy các bản ghi logic được gọi là một file.

Thao tác với file tuần tự: tại vị trí hiện thời ("cửa sổ"), tại cuối, tại đầu file v.v. Một số chỉ thị I/O cơ sở:

- READ: đọc bản ghi tiếp theo:
  - Từ một file đã được chỉ ra (không cần chỉ ra địa chỉ bên trong file)
  - Đặt nó vào các ô nhớ liên tiếp trong bộ nhớ chính, bắt đầu từ một địa chỉ được chỉ rõ.
- WRITE: viết 1 bản ghi vào file.
- REWIND để định vị đầu file V.v.
- OPEN: Nhiều HĐH đòi hỏi phải mở file trước khi sử dụng nó, khi đó cần có chỉ thị OPEN.

6.3.2 Các chỉ thị vào/ra ảo đối với các file truy cập ngẫu nhiên

Trong nhiều ứng dụng các chương trình cần đọc/ghi các bản ghi logic một cách ngẫu nhiên. Hầu hết các hệ điều hành cung cấp các chỉ thị ảo để đọc hoặc ghi bản ghi logic thứ n. Có một dạng tổ chức file khác trong đó các bản ghi logic được đánh địa chỉ không phải bằng vị trí của nó trong file mà bằng nội dung của

6.3.3 Việc cài đặt các chỉ thị vào/ra ảo

Để có thể hiểu được các chỉ thị I/O ảo được cài đặt như thế nào ở máy mức 2, cần phải tìm hiểu việc các file được tổ chức và lưu trữ như thế nào. Vấn đề cơ bản mà hệ thống file phải giải quyết là việc sắp đặt bộ nhớ ngoài. Trong đó có hai vấn đề chính là:

- Kích thước của đơn vị nhớ ngoài sẽ được dùng để sắp đặt chỗ cho file.
- File sẽ được chứa trong các đơn vị nhớ ngoài liên tiếp hay không?

IV. Bài tập củng cố kiến thức

6.1 Nhiệm vụ quan trọng nhất của hệ điều hành là gì?

6.2 Bộ nhớ ảo là gì ? Tại sao phải sử dụng bộ nhớ ảo ?

6.3 Làm thế nào để tăng tốc độ việc chuyển đổi địa chỉ ảo thành địa chỉ vật lý ?

6.4 Phương pháp cấp trang khi có yêu cầu là gì ? Nêu ưu nhược điểm của phương pháp này.

6.5 Phương pháp tập làm việc là gì ? Nêu ưu nhược điểm của phương pháp này.

CHƯƠNG 7 CẤP HỢP NGỮ

I. Mục đích: Giúp sinh viên nắm được về

- Mức hợp ngữ
- Ngôn ngữ assembly

II. Yêu cầu: Yêu cầu sinh viên học lý thuyết và làm bài tập

III. Nội dung: Chương 7 được giảng dạy trong 3 tiết lý thuyết

7.1Vi hợp ngữ

Trong cấp vi chương trình , ngôn ngữ được sử dụng để thể hiện nội dung của vi chương trình là các ký hiệu, gọi là Micro Assembly Language (MAL). Vi chương trình phải được dịch sang dạng nhị phân, từng bit của vi chỉ thị 32 bit được nối với các đường dây để điều khiển toàn bộ đường dữ liệu.

Cách viết vi chỉ thị: Trên mỗi dòng ghi một vi chỉ thị, trên dòng đó ghi tên các trường khác 0 và giá trị của chúng.

Ví dụ để cộng AC với A và chứa kết quả vào AC, chúng ta có thể viết:

ENC = 1, C = 1, B = 1, A = 10

Nhiều ngôn ngữ vi chương trình trông giống như vậy, tuy nhiên việc viết chương trình như thế không gần gũi và dễ gây phức tạp. Chúng ta sẽ cố gắng làm cho nó trông giống với ngôn ngữ lập trình Pascal

Ví dụ, chỉ thị trên được viết như sau: ac:=a+ac.

Để chỉ rõ việc sử dụng các chức năng 0, 1, 2 và 3 của ALU, chúng ta viết:

```
ac:=a+ac
a:=band(ir,smask)        ;# band = "Boolean and“
ac:=a
a:=inv(a)            ;# inv = “invert”
```

Các thao tác dịch trái và dịch phải có thể được ký hiệu là lshift và rshift, như:

tir:=lshift(tir+tir)

Statement (Chỉ thị ngôn ngữ MAL)	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR
Mar := pc; rd	0	0	2	0	0	1	1	0	0	0	0	0	00
Rd	0	0	2	0	0	0	1	0	0	0	0	0	00
ir := mbr	1	0	2	0	0	0	0	0	1	3	0	0	00

pc := pc+1	0	0	0	0	0	0	0	0	1	0	6	0	00
Mar := ir; mbr := ac; wr	0	0	2	0	1	1	0	1	0	0	3	1	00
alu := tir; if n then goto 15	0	1	2	0	0	0	0	0	0	0	0	4	15
ac := inv (mbr)	1	0	3	0	0	0	0	0	1	1	0	0	00
tir := lshift (tir); if n then goto 25	0	1	2	2	0	0	0	0	1	4	0	4	25
alu := ac; if z then goto 22	0	2	2	0	0	0	0	0	0	0	0	1	22
ac:=band(ir, amask); goto 0	0	3	1	0	0	0	0	0	1	1	8	3	00
sp := sp + (-1); rd	0	0	0	0	0	0	1	0	1	2	2	7	00
tir := shift(ir+ir); if n then goto 69	0	1	0	2	0	0	0	0	1	4	3	3	69

Bảng 7-1 Nội dung thanh ghi vi chỉ thị

7.2 Giới thiệu về hợp ngữ

Cấp hợp ngữ là cấp thuộc mức trên của cấp vi chương trình. Cấp hợp ngữ khác với cấp vi chương trình, cấp quy ước, và cấp hệ điều hành do cấp này được thực hiện bởi chương trình dịch, không phải là trình phiên dịch.

Các chương trình biến đổi chương trình của người sử dụng được viết bằng ngôn ngữ này thành một ngôn ngữ khác được gọi là trình dịch. Ngôn ngữ dùng để viết chương trình ban đầu gọi là ngôn ngữ nguồn và ngôn ngữ mà chương trình ban đầu được biến đổi thành gọi là ngôn ngữ đích. Cả hai ngôn ngữ nguồn và ngôn ngữ đích đều xác định các cấp. Nếu bộ xử lý có thể thực thi trực tiếp các chương trình viết bằng ngôn ngữ nguồn, ta không cần dịch chương trình nguồn thành chương trình trong ngôn ngữ đích.

Việc dịch được sử dụng khi bộ xử lý chỉ có thể được dùng cho ngôn ngữ đích mà không dùng được cho ngôn ngữ nguồn. Nếu việc dịch được thực hiện đúng, việc chạy chương trình đã được dịch sẽ cho cùng kết quả như khi thực hiện chương trình nguồn đã cho bằng cách dùng bộ xử lý nếu có thể. Kết quả là ta có thể hiện thực một cấp mới không có bộ xử lý bằng cách trước tiên dịch các chương trình viết cho cấp này thành các chương trình trong cấp đích và sau đó thực thi chương trình ở cấp đích.

Điều quan trọng cần lưu ý là sự khác nhau giữa dịch và và phiên, chương trình ban đầu được viết bằng ngôn ngữ nguồn không được thực thi trực tiếp. Thay vào đó, chương trình này được đổi thành một chương trình tương đương gọi là chương trình đối tượng hay module đối tượng. Việc thực thi chương trình đối tượng

chỉ được tiến hành sau khi việc dịch đã hoàn tất. Trong dịch, ta có hai bước phân biệt:

1. Tạo ra chương trình tương đương trong ngôn ngữ đích
2. Thực hiện chương trình vừa mới tạo ra

7.2.1 Ngôn ngữ assembly là gì ?

Các chương trình thực hiện chuyển đổi chương trình của người sử dụng được viết bằng ngôn ngữ nào đó sang một ngôn ngữ khác được gọi là chương trình dịch(translator). Ngôn ngữ được sử dụng để viết chương trình nguồn được gọi là ngôn ngữ nguồn còn ngôn ngữ chương trình nguồn chuyển đổi sang gọi là ngôn ngữ đích. Cả ngôn ngữ nguồn và ngôn ngữ đích đều xác định các mức máy.

Dựa vào mối quan hệ giữa ngôn ngữ nguồn và ngôn ngữ đích có thể chia các chương trình dịch thành hai nhóm. Khi ngôn ngữ nguồn về căn bản là một sự biểu diễn bằng ký hiệu cho một ngôn ngữ máy bằng số thì chương trình dịch được gọi là assmbler và ngôn ngữ nguồn được gọi là ngôn ngữ assembly. Khi ngôn ngữ nguồn là một ngôn ngữ bậc cao như là C hoặc Pascal và ngôn ngữ đích là một ngôn ngữ máy bằng số hoặc là một biểu diễn bằng ký hiệu cho một ngôn ngữ như vậy thì chương trình dịch được gọi là compiler.

Ngôn ngữ assembly thuần khiết là ngôn ngữ trong đó mỗi chỉ thị của nó khi được dịch sinh ra đúng một chỉ thị máy. Nói cách khác, có tương ứng một – một giữa các chỉ thị máy và các chỉ thị trong ngôn ngữ assemblt. Nếu mỗi dòng trong chương trình assembly chứa một chỉ thị assembly và mỗi word máy chứa một chỉ thị máy, thì chương trình assembly dài n dòng sẽ sinh ra một chương trình ngôn ngữ máy dài n word.

Sử dụng ngôn ngữ assembly để lập trình dễ hơn sử dụng ngôn ngữ máy rất nhiều. Việc sử dụng tên và địa chỉ bằng ký hiệu thay cho số nhị phân(hoặc số hệ 8, hệ 16) tạo nên một sự khác biệt rất lớn. Mọi người đều có thể nhớ được các ký hiệu viết tắt cho cộng(add), trừ(substract), nhân(multiply) và chia(divide) là ADD, SUB, MUL và DIV; nhưng ít người có thể nhớ được các chỉ thị máy cho các phép toán đó là 24576, 57344, 28672 và 29184(cho máy PDP- 11). Người lập trình bằng ngôn ngữ assembly chỉ cần nhớ các tên bằng ký hiệu gợi nhớ ADD, SUB, MUL và DIV, bởi vì assembler sẽ chỉ dịch chúng sang các chỉ thị máy. Tuy nhiên muốn có ai đó lập trình bằng ngôn ngữ máy thì họ cần phải nhớ, nếu không thì sẽ phải liên tục tra cứu các giá trị này.

Đối với địa chỉ cũng có thể rút ra các nhận xét tương tự. Người lập trình bằng ngôn ngữ assembly có thể đặt tên bằng ký hiệu gợi nhớ cho các vị trí nhớ và

giao phó cho assembler lo cung cấp đúng các gía trị bằng số; trong khi đó người lập trình bằng ngôn ngữ máy luôn luôn phải làm việc với các giá trị bằng số của các địa chỉ.

Hệ quả là từ khi assembler ra đời cho đến nay không còn một ai viết chương trình bằng ngôn ngữ máy nữa.

Ngoài ánh xạ môt – một của các chỉ thị assembly vào các chỉ thị máy, ngôn ngữ assembly còn có một tính chất khác nữa làm cho nó khác hẳn các ngôn ngữ lập trình bậc cao, đó là người lập trình bằng ngôn ngữ assembly có thể truy cập tới tất cả các đặc điểm và các chỉ thị có trong máy đích. Người lập trình bằng ngôn ngữ bậc cao không làm được như vậy. Thí dụ nếu máy đích có một bit báo tràn số(overflow bit), thì chương trình bằng ngôn ngữ assembly có thể kiểm tra trực tiếp bit này, trong khi đó chương trình bằng ngôn ngữ Pascal không thể kiểm tra trực tiếp được. Nếu ở bàn điều khiển của một máy tính có các công tắc thì chương trình bằng ngôn ngữ assembly có thể đọc trạng thái của chúng. Một chương trình như vậy có thể thi hành từng chỉ thị trong tập chỉ thị của máy đích, còn chương trình bằng ngôn ngữ bậc cao không thể làm như vậy.

Một sự khác nhau quan trọng nữa giữa chương trình assembly và chương trình bằng ngôn ngữ bậc cao, là chương trình bằng ngôn ngữ assembly chỉ có thể chạy được trên một họ máy, trong khi đó chương trình được viết bằng ngôn ngữ bậc cao nói chung có thể chạy được trên nhiều họ máy, đây chính là một ưu điểm lớn của nó so với ngôn ngữ assembly.

Tóm lại tất cả các việc có thể thực hiện bằng ngôn ngữ máy đều có thể thực hiện được bằng ngôn ngữ assembly, tuy nhiên ngôn ngữ bậc cao không làm được như vạy một cách có hiệu quả. Các ngôn ngữ dùng cho lập trình hệ thống thường là giao của ngôn ngữ bậc cao và ngôn ngữ assembly, trong đó cú pháp là của ngôn ngữ lập trình bậc cao nhưng lại truy cập được máy bằng ngôn ngữ assembly.

7.2.2 Khuôn dạng chỉ thị ngôn ngữ assembly

Mặc dù cấu trúc của một chỉ thị ngôn ngữ assembly(trong chương trình này chúng ta sẽ trường gọi ngắn gọn là chỉ thị hay instruction) rất giống cấu trúc của một chỉ thị máy mà nó biểu diễn, các ngôn ngữ assembly cho các máy khác nhau và cho các mức khác nhau lại khá giống nhau, điều đó cho phép chúng ta có thể thảo luận chung về ngôn ngữ assembly. Trên hình 7-01 là đoạn chương trình ngôn ngữ assembly cho các máy dùng bộ vi xử lý 80386, thực hiện việc tính N = I + J + K. Trong đoạn chương trình lấy làm ví dụ, phần chương trình nằm bên trên dòng có

dấu chấm thực hiện việc tính toán, còn các dòng nằm bên dưới là các lệnh cho assembler dành ra các ô nhớ cho biến I, J, K và N chứ không phải là các biểu diễn ký hiệu cho các chỉ thị máy. Các chỉ thị là mệnh lệnh cho assembler thường được gọi là chỉ thị assembler.

Trường nhãn	Trường mã phép toán	Trường toán hạng	Trường chú thích
FORMUL	MOV	EAX, I	;Nạp giá trị biến I vào thanh ghi EAX
	ADD	EAX, J	;Cộng giá trị biến J vào thanh ghi EAX
	ADD	EAX, K	;Cộng giá trị biến K vào thanh ghi EAX
	MOV	N, EAX	;Chứa kết quả I+J+K vào biến N
	...		
I:	DD	2	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 2
J:	DD	3	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 3
K:	DD	4	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 4
N:	DD	0	;Dành sẵn 4 byte bộ nhớ, khởi tạo giá trị 0

Hình 7-1 Tính N = I + J + K

Chỉ thị - Instruction

Instruction là chỉ thị ngôn ngữ assembly, nó có 4 trường: trường nhãn, trường mã phép toán, trường toán hạng và trường chú thích. Có những chỉ thị chỉ có một trường mã phép toán, có chỉ thị có thêm trường các toán hạng, còn trường chú thích có hay không tùy thuộc người viết chương trình.

Nhãn	Mã phép toán	Toán hạng	; Chú thích

Nhãn (Label):

Nhãn là một dãy các ký hiệu chữ cái, chữ số...được sử dụng như một tên, gán cho các địa chỉ bộ nhớ, các lệnh trong chương trình có thể dùng tên này thay cho địa chỉ bộ nhớ mà nhãn đại diện. Nếu nhãn đứng trước các chỉ thị của ngôn ngữ assembly thì các lệnh nhảy có thể dùng nhãn cày thay cho địa chỉ đích nếu chúng cần nhảy tới lệnh được gán nhãn. Nhãn cũng cần cho các **chỉ thị assembler** có nhiệm vụ sắp đặt bộ nhớ, thí dụ DD (Define Double Word) – lệnh cho assembler dành sẵn 1 wỏd kép (4 byte), để có thể truy cập được dữ liệu cất ở đó thông qua tên nhãn.

Các ngôn ngữ assembly cụ thể có các yêu cầu khác nhau về chiều dài của dãy ký tự dùng làm nhãn, các ký tự được phép sử dụng trong nhãn. v. v.

Trên hình 7-1 chúng ta thấy có 5 nhãn: FORMUL, I, J, K và N, các nhãn kết thúc bằng ký tự " : ", đây chỉ là một quy ước cụ thể trong một số ngôn ngữ assembly,không phải là điều gì đặc biệt quan trọng.

Mã phép toán (Opcode – Operation code)

Mã phép toán là chữ viết tắt gợi trí nhớ của ngôn ngữ assembly đại diện cho một chỉ thị máy bằng số. Thí dụ ADD, MOV trên hình 7-01.

Trường toán hạng (operand)

Báo cho bộ xử lý biết phải tìm các toán hạng ở đâu, hay nói cách khác trường này chỉ ra địa chỉ của các toán hạng nếu phép toán có toán hạng. Tất nhiên cũng có những chỉ thị assembly không có toán hạng.

Trường chú thích

Chú thích (comment) thường được quy ước phải viết sau ký tự ‘ ; ‘ và kết thúc bởi dãy xuống dòng. Chú thích dùng để diênc giải dòng lệnh trong chương trình để việc đọc được dễ hiểu hơn. Trong một chương trình dài các chú thích rất quan trọng và cần thiết.

Giả chỉ thị - Pseudoinstruction

Trong chương trình viết bằng ngôn ngữ assembly, các **chỉ thị cho assembler** trông cũng giống như các chỉ thị cho bộ vi xử lý thực hiện, vì thế nó còn được gọi là giả chỉ thị - **Pseudoinstruction**. Chỉ thị assembler gồm có 4 trường hợp như sau:

Tên	Chỉ thị assembler	Đối số	;Chú thích

Tên

Cũng giống như trường nhãn, chỉ thị assembler có thể có hoặc không có trường tên. Tên là một dãy các ký hiệu chữ cái, chữ số... Có thể dùng tên để gán cho các địa chỉ bộ nhớ và sử dụng tên này như các biến trong các chương trình ngôn ngữ lập trình bậc cao. Cũng có thể gán tên cho các hằng số mà chương trình sẽ sử dụng.

Chỉ thị assembler

Cũng tương tự như các chỉ thị (instruction), đó là chữ viết tắt gợi trí nhớ của ngôn ngữ assembly, nhưng đó là lệnh cho chính chương trình dịch – assembler, báo cho nó biết phải làm gì hoặc làm như thế nào. Nói chung không có quy tắc nào giúp ta phân biệt chỉ thị (instruction) và vhỉ thị assembler (Pseudoinstruction) cả, chúng đều là các từ viết tắt gợi nhớ, người lập trình bằng ngôn ngữ assembly chỉ có cách tra cứu và dần dần thược “mặt” chúng.

Trường đối số (argument)

Tùy thuộc vào chỉ thị assembler cụ thể, đối số có thể là một địa chỉ nhớ hoặc một số để sử dụng cùng với chỉ thị và do chỉ thị xác định.

Trường chú thích

Chú thích thường viết sau ký hiệu ‘ ; ‘ và kết thúc bởi dãy xuống dòng (Enter). Các chú thích dùng để diễn giả dòng lệnh trong chương trình, để việc đọc lại được dễ hiểu hơn.

7.2.3 So sánh ngôn ngữ assembly và các ngôn ngữ bậc cao

Trước khi cần tạo ra các chương trình sẽ chạy thường xuyên, thí dụ các chương trình của hệ điều hành, nói chung người ta đều viết bằng ngôn ngữ assembly nhằm đạt tốc độ chạy cao và tiết kiệm bộ nhớ. Ngày nay người ta không làm như vậy nữa. Có thể kể ra một số nguyên nhân chính sau đây.

Thứ nhất, giá bộ nhớ giảm không ngừng, việc cố gắng viết chương trình sao cho càng nhỏ càng tốt để tiết kiệm bộ nhớ không còn mấy ý nghĩa nữa. Ngoài ra tốc độ bộ nhớ và CPU cũng ngày càng tăng, việc sử dụng ngôn ngữ assembly nhằm tạo ra các chương trình chạy nhanh hơn trong phần lớn các trường hợp cũng không đem lại ý nghĩa thực tế gì.

Nguyên nhân thứ hai là các chương trình ngày càng lớn và phức tạp, được nhiều người cùng tham gia viết trong thời gian dài, vấn đề dễ đọc, dễ bảo trì, dễ sửa đổi và sử dụng lại có ý nghĩa thực sự quan trọng.

Nguyên nhân thứ ba liên quan tới các chương trình dịch (compiler), việc viết các chương trình dịch đã trở thành một công nghệ phát triển, các chương trình dịch ngày nay có thể nhận vào các chương trình nguồn viết bằng ngôn ngữ bậc cao và sinh ra các chương trình mã máy rất hiệu quả, chúng vừa nhỏ vừa chạy nhanh.

Chúng ta sẽ rút ra được các kết luận hữu ích khi so sánh hai hệ điều hành là MULTICS và TSS/67 cho máy IBM 360/67. Cả hai hệ điều hành này đều được bắt đầu viết vào cùng một khoảng thời gian và chúng có kích thước xấp xỉ nhau. Một phần lớn, khoảng 95% hệ điều hành MULTICS được viết bằng ngôn ngữ bậc cao PL/I, còn hệ điều hành TSS/67 của IBM được viết hoàn toàn bằng ngôn ngữ assembly.

Các hệ điều hành lớn như MULTICS và TSS/67 phải thực hiện nhiều nhiệm vụ khác nhau, chẳng hạn phải điều khiển tất cả các thiết bị I/O, giải quyết tất cả các tình huống được định thời chặt chẽ, phải quản lý các cơ sở dữ liệu lớn. v. v. Hiệu suất cao là một vấn đề có tính cốt yếu đối với hệ điều hành, chính vì vậy nó có thể được sử dụng để viết nên hệ điều hành.

Năng suất lập trình tính trung bình theo số dòng lệnh không phụ thuộc ngôn ngữ lập trình được sử dụng.

Kết quả thực hiện hai dự án hệ điều hành trên cho thấy, chúng được hoàn thành trong các khoảng thời gian ngang nhau. Tuy nhiên điều đặc biệt đáng chú ý là

nhóm viết MULTICS chỉ có 50 người và chi phí ước tính 10 triệu đôla, trong khi nhóm viết TSS/67 gồm 300 người và chi phí ước tính là 50 triệu đôla (theo Grâhm,1970). Việc sử dụng ngôn ngữ lập trình bậc cao PL/I đã tiết kiệm cho dự án MULTICS hàng chục triệu đôla, chi phí cho việc viết chương trình chỉ bằng 20% so với dự án TSS/67.

Các nghiên cứu đã chỉ ra rằng, nếu một người lập trình làm việc cho một dự án trong khoảng thời gian vài năm, thì tính trung bình hàng tháng anh ta viết được từ 100 đến 200 dòng lệnh, không phụ thuộc vào ngôn ngữ lập trình được sử dụng (theo Corbató, 1969). Người lập trình chỉ có thể hy vọng đạt được năng suất lao động lập trình cao so với các chương trình nhỏ. Bởi vì một chỉ thị ngôn ngữ PL/I tương đương với từ 5 đến 10 lần năng suất của người lập trình assembly. Đối với các ngôn ngữ lập trình bậc cao khác, kết quả cũng tương tự như vậy.

Chương trình bằng ngôn ngữ bậc cao dễ đọc, dễ hiểu hơn.

Có một lý do quan trọng khác nữa khiến người ta tránh sử dụng ngôn ngữ assembly; Đó là ngay cả một người lập trình assembly chuyên nghiệp nói chung cũng khó mà hiểu được chương trình assembly do người khác viết, nhất là khi chương trình đó dài.

Chương trình nguồn đầy đủ của hệ điều hành MULTICS bằng ngôn ngữ PL/I dài khoảng 300 trang, với chiều dài đó hiểu và nắm vững đã là một việc không hề dễ dàng. Tuy nhiên việc đó cũng chỉ là tầm thường nếu đem so cới ciệc phải đọc 30.000 chương trình viết bằng assembly. Mặc dầu không có ai thử cố gắng đọc toàn bộ chương trình nguồn của MULTICS, nhưng cũng có những người cố gắng tìm hiểu các chương trình con cụ thể, chúng có chiều dài trung bình 4 trang bằng ngôn ngữ PL/I.

Trong các dự án lớn sự thay đổi nhân sự hàng năm tính trung bình là 15%, như vạy sau 5 năm trong nhóm lập trình chỉ còn lại rất ít người đã tham gia từ đầu dự án. Nếu như những người lập trình mới ra nhập nhóm không hiểu được chương trình của những người trước họ để lại thì các dự án sẽ gặp một trở ngại rất lớn.

IV. Bài tập củng cố kiến thức:

7.1 Trong cấp vi chương trình ngôn ngữ được sử dụng để thể hiện nội dung của vi chương trình là ngôn ngữ gì?

7.2 Cấp hợp ngữ là gì? So sánh cấp hợp ngữ với cấp vi chương trình và cấp hệ điều hành

7.3 Ngôn ngữ assembly là ngôn ngữ gì? So sánh ngôn ngữ assembly với ngôn ngữ bậc cao

7.4 Giải thích đoạn lệnh sau:

```
CLEAR R0;
MOVE R1, #100;
CLEAR R2;
LOOP: ADD R0 ,1000(R2);
INCREMENT R2;
DECREMENT R1;
BRANCH-IF > 0 LOOP; // nếu R1 >0 thì thực hiện LOOP
STORE (2000), R0;
```

7.5 Giải thích đoạn lệnh sau:

```
MOV R4, #15
ADD R4, (R1)
```

CHƯƠNG 8 HỆ THỐNG VÀO RA

I. Mục đích: Giúp sinh viên nắm được

- Cơ bản về hệ thống vào ra
- Phương pháp ghép nối thiết bị ngoại vi

II. Yêu cầu: sinh viên cần học lý thuyết và làm bài tập trong sách ra bài tập

III. Nội dung: Chương 8 được giảng dạy trong 2 tiết lý thuyết

8.1 Tổng quan về hệ thống vào ra

Chức năng: Trao đổi thông tin giữa Máy tính với môi trường bên ngoài.

Máy tính muốn trao đổi với con người cần có bàn phím, chuột, loa, mic, … Máy tính muốn trao đổi thông tin với không gian cần có máy ảnhthẻ nhớ, máy Scan, …

Các thao tác cơ bản:

- Vào dữ liệu
- Ra dữ liệu

Các thành phần chính:

- Thiết bị ngoại vi
- Module ghép nối vào ra

❖ **Thiết bị ngoại vi**

Chức năng: phương tiện chuyển đổi thông tin giữa bên trong và bên ngoài máy tính

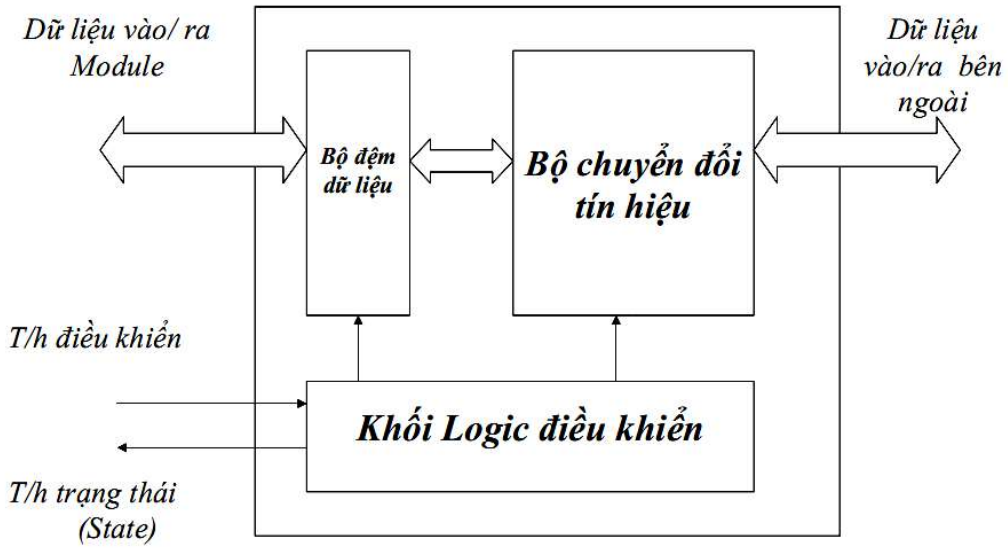
Đặc điểm các thiết bị:

Trên thị trường tồn tại rất nhiều các thiết bị ngoại vi khác nhau về: Nguyên tắc hoạt động, tốc độ, định dạng dữ liệu truyền, v.v. Đồng thời các thiết bị này có tốc độ làm việc chậm hơn CPU và RAM rất nhiều. Chính vì lý do trên cần có Module vào ra để ghép nối các thiết bị ngoại vi vào hệ thống BUS máy tính.

❖ **Phân loại:**

- Thiết bị nhập: Keyboard, Mouse, Scan, Micro,…
- Thiết bị xuất: Monitor, Printer,
- Thiết bị xuất nhập: Modem, NIC, Driver,…

❖ **Cấu trúc tổng quát của thiết bị ngoại vi:**



Hình 8-1 Sơ đồ cấu trúc tổng quan của thiết bị ngoại vi

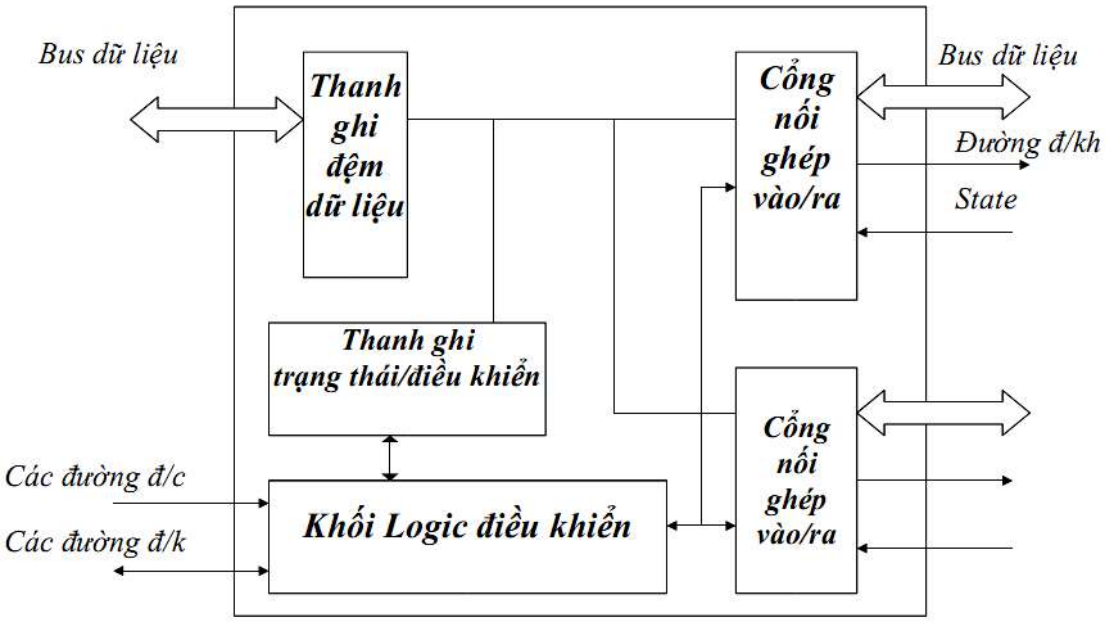
- Bộ chuyển đổi tín hiệu: chuyển đổi dữ liệu giữa bên trong và bên ngoài Máy tính
- Bộ đệm dữ liệu: nơi lưu trữ dữ liệu trung gian giữa Máy tính và thiết bị ngoại vi, đặt bên trong thiết bị ngoại vi.
- Khối logic điều khiển: điều khiển hoạt động của thiết bị ngoại vi theo tín hiệu từ Module I/O gởi tới thiết bị.

❖ **Module I/O**

Chức năng: Nối ghép thiết bị ngoại vi với bus của máy tính.

- Điều khiển và định thời
- Trao đổi thông tin với CPU
- Trao đổi thông tin với thiết bị ngoại vi
- Đệm giữa máy tính với thiết bị ngoại vi
- Phát hiện lỗi của các thiết bị ngoại vi.

Cấu trúc chung:



Hình 8-2 Sơ đồ cấu trúc chung của module vào ra

- Thanh ghi đệm dữ liệu: đệm dữ liệu trong quá trình trao đổi
- Cổng nối ghép vào ra: kết nối thiết bị ngoại vi, mỗi cổng có địa chỉ xác định và chuẩn kết nối riêng phụ thuộc sơ đồ chân.
- Thanh ghi trạng thái/điều khiển: lưu trữ thông tin trạng thái cho các cổng vào ra
- Khối logic điều khiển: điều khiển Module vào ra

8.2 Các phương pháp điều khiển vào ra

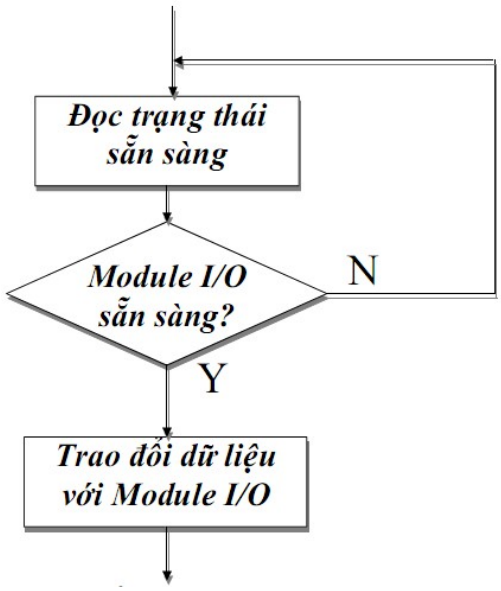
Có thể phân loại các phương pháp điều khiển vào ra theo 3 cách sau:

- Vào ra bằng chương trình
- Vào ra bằng ngắt
- Truy cập bộ nhớ trực tiếp DMA

8.2.1 Vào ra bằng chương trình – polling

Nguyên tắc chung:

- Sử dụng lệnh vào ra trong chương trình để trao đổi dữ liệu với cổng vào ra.
- Khi CPU thực hiện chương trình gặp lệnh vào ra thì CPU điều khiển trao đổi dữ liệu với cổng vào ra.



- CPU và thiết bị ngoại vi chỉ trao đổi dữ liệu khi có tín hiệu móc nối báo sẵn sàng (Ready/Akc) của các phía. Sau khi máy tính khởi động thiết bị ngoại vi (khối ghép nối), máy tính luôn chờ và kiểm tra trạng thái sẵn sàng của thiết bị ngoại vi gồm các bước:

  (1) Đọc thông tin về trạng thái sẵn sàng của thiết bi ngoại vi

  (2) Kiểm tra: Nếu thiết bị ngoại vi sẵn sàng thì trao đổi dữ liệu, ngược lại thì về bước (1) để kiểm tra lại.

Phương pháp này được dùng khi tốc độ trao đổi dữ liệu của các bên (CPU và thiết bị ngoại vi) rất không bằng nhau.

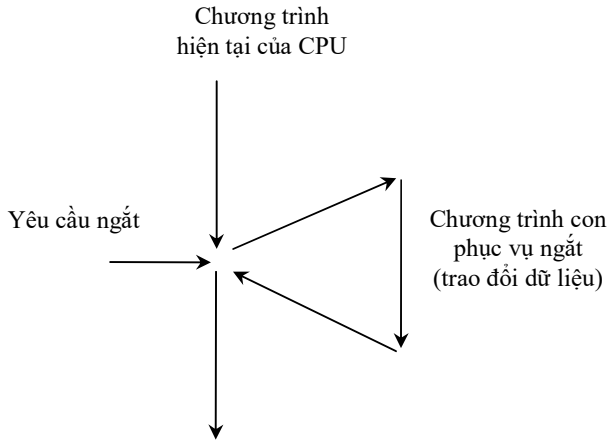
Nhận xét:

- Việc trao đổi thông tin là tin cậy vì chỉ trao đổi khi thiết bị ngoại vi sẵn sàng.
- Tốn thời gian CPU vì phải kiểm tra trạng thái sẵn sàng của thiết bị ngoại vi (việc kiểm tra này co CPU đảm nhiệm) nên giảm hiệu suất của hệ thống.
- Phù hợp với những hệ thống không đòi hỏi cao về tốc độ trao đổi dữ liệu, hệ thống có ít thiết bị ngoại vi.

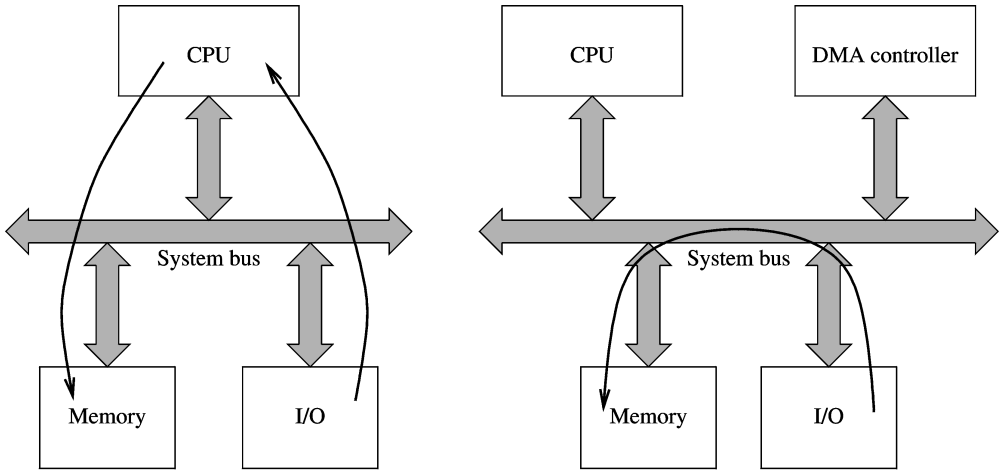
8.2.2 Vào ra bằng phương pháp ngắt

Bình thường máy tính thực hiện một chương trình (công việc) nào đó. Khi thiết bị ngoại vi có yêu cầu trao đổi dữ liệu, nó sẽ gửi tín hiệu yêu cầu ngắt CPU dừng công việc hiện tại, phục vụ cho trao đổi dữ liệu thông qua tín hiệu yêu cầu ngắt IRQ (Interrupt Request) tác động vào chân INTR (chân tiếp nhận yêu cầu ngắt) của CPU. CPU nhận được yêu cầu ngắt, nếu chấp nhận nó sẽ đưa ra xung INTA xác nhận tới thiết bị ngoại vi, sau đó CPU tìm chương trình con phục vụ ngắt tương ứng

số hiệu ngắt và thực hiện nó. Đó chính là chương trình con thực hiện trao đổi (vào/ra) dữ liệu do thiết bị ngoại vi yêu cầu. Khi trao đổi xong (ISR – Interrupt Service Routine) kết thúc thì CPU tiếp tục công việc (chương trình) đã bị gián đoạn.



8.2.3 Vào ra sử dụng DMA



(a) Programmed I/O transfer (b) DMA transfer

Trong các phương pháp vào/ra dữ liệu bằng chương trình kể trên, dữ liệu phải được chuyển qua lại từ bộ nhớ đến CPU rồi đến thiết bị ngoại vi hoặc ngược lại bằng việc thực hiện từng lệnh (MOV, IN hoặc OUT) của CPU với sự tham gia của các thanh ghi. Dữ liệu của mỗi lần vận chuyển là byte hoặc word (2 byte), tốc độ trao đổi dữ liệu phụ thuộc rất nhiều vào tốc độ thực hiện các lệnh trao đổi dữ liệu kể trên. Nói chung, tốc độ trao đổi dữ liệu là không thể nhanh được. Với các thiết bị làm việc với bộ nhớ khối như màn hình, ổ đĩa, ... yêu cầu trao đổi cả mảng dữ liệu thì phương pháp vào/ra dữ liệu bằng chương trình là không phù hợp. Khi đó người ta nghĩ đến việc điều khiển dữ liệu vào/ra trực tiếp từ bộ nhớ đến thiết bị ngoại vi

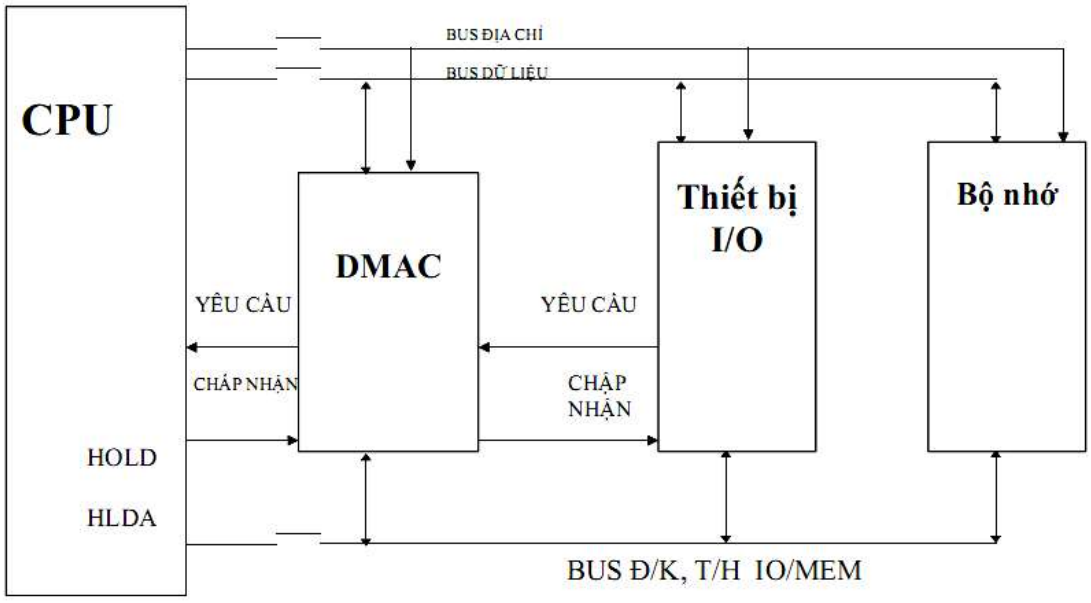
hoặc ngược lại mà không thông qua CPU bằng những lệnh trao đổi dữ liệu như MOV, IN hoặc OUT. Đó là phương pháp vào/ra dữ liệu bằng cách truy nhập trực tiếp bộ nhớ (DMA – Direcr Memory Access). Trong trường hợp này CPU trao quyền điều khiển cho một mạch phần cứng phụ điều khiển việc vào/ra dữ liệu, đó là DMAC – DMA Controller

❖ **Các thành phần của DMAC**

- Thanh ghi dữ liệu: chứa dữ liệu trao đổi.
- Thanh ghi địa chỉ: chứa địa chỉ của ngăn nhớ dữ liệu
- Bộ đếm dữ liệu: chứa số từ dữ liệu cần trao đổi
- Khối logic điều khiển: điều khiển hoạt động của DMAC

❖ **Hoạt động của DMA**

- Khi cần vào ra dữ liệu thì CPU nhờ DMAC tiến hành vào ra dữ liệu với thông tin cho biết như sau:
  - Địa chỉ thiết bị vào ra
  - Địa chỉ đầu của mảng nhớ chứa dữ liệu và DMAC nạp thanh ghi địa chỉ
  - Số từ dữ liệu cần truyền và DMAC nạp vào bộ đếm dữ liệu
  - CPU sẽ đi thực hiện việc khác
  - DMAC điều khiển việc trao đổi dữ liệu sau khi truyền một từ dữ liệu thì nội dung thanh ghi địa chỉ tăng lên và nội dung bộ đếm dữ liệu giảm xuống một đơn vị.
  - Khi bộ đếm bằng dữ liệu bằng 0, DMAC gởi tín hiệu ngắt CPU để báo kết thúc DMA



Hình 8-3 Sơ đồ kết nối của DMAC

❖ **Các kiểu thực hiện DMA**

- DMA truyền theo khối: DMAC sử dụng BUS để truyền cả khối dữ liệu (CPU chuyển nhượng BUS cho DMAC)
- DMA lấy chu kỳ: DMAC cưỡng bức CPU treo tạm thời từng chu kỳ BUS để 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 dữ liệu (DMAC lấy lén chu kỳ)

❖ **Đặc điểm DMA**

- CPU không tham gia trong quá trình trao đổi dữ liệu
- DMAC điều khiển trao đổi dữ liệu giữa bộ nhớ chính và Module vào ra với tốc độ nhanh.
- Phù hợp với yêu cầu trao đổi mảng dữ liệu có kích thước lớn.

8.3 Ghép nối thiết bị ngoại vi

❖ **Các kiểu nối ghép vào ra**

- Nối ghép song song
- Nối ghép nối tiếp

❖ **Nối ghép song song**

- Truyền các bit dữ liệu được truyền song song trong cùng một thời điểm trên nhiều đường dây.
- Tốc độ truyền cao

- Cần đường truyền song song để tải các bit dữ liệu cùng đi, điều đó khiến phương pháp này tốn kém về dây dẫn.

  Điển hình của phương pháp này công máy in 25 chân LPT.

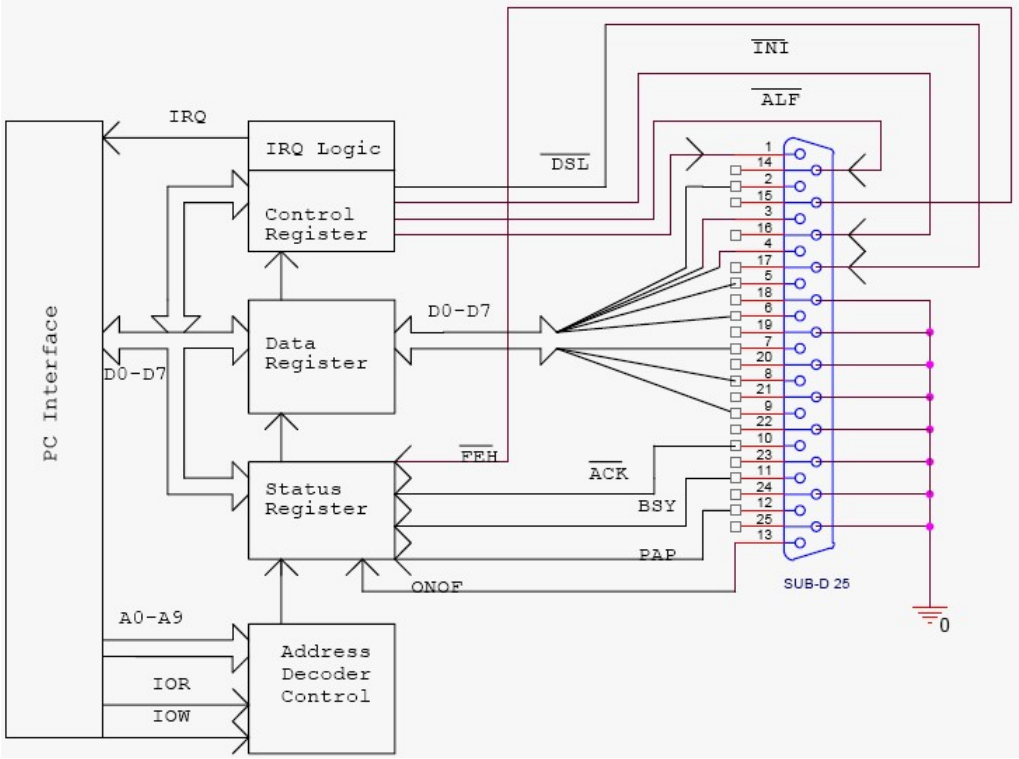
❖ **Nối ghép nối tiếp**

- Từng Bit của dữ liệu lần lượt được gửi đi trên một đường truyền duy nhất
- Dữ liệu trong máy tính thương ở dạng 8bit, 16b,… vì thế cần có bộ chuyển đổi từ song song sang nối tiếp.
- Tốc độ truyền của phương pháp này chậm vì truyền tường bit trên một đường dây
- Ưu điểm là kinh tế, không tốn dây, có thể dùng để truyền đi xa.
- Các cấu hình ghép nối ghép
  - Điểm - điểm (point to point): Qua một cổng vào ra chỉ có thể ghép một thiết bị ngoại vi (PS/2, COM, LPT,…)
  - Điểm - đa điểm (Point to multipoint): Thông qua một cổng vào ra ghép nhiều thiết bị vào ra. Ví dụ: SCSI(7,15), USB (127),…

8.4 Các cổng vào ra thông dụng

8.4.1 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 8-4 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:

	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

(a)

	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	-	-	-

(b)

	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

(c)

Bảng 8-1 Bảng định dạng cho các thanh ghi dữ liệu, trạng thái và điều khiển

(b) Thanh ghi dữ liệu (hai chiều)

(c) Thanh ghi trạng thái máy in (chỉ đọc)

(d) Thanh ghi điều khiển máy in

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:

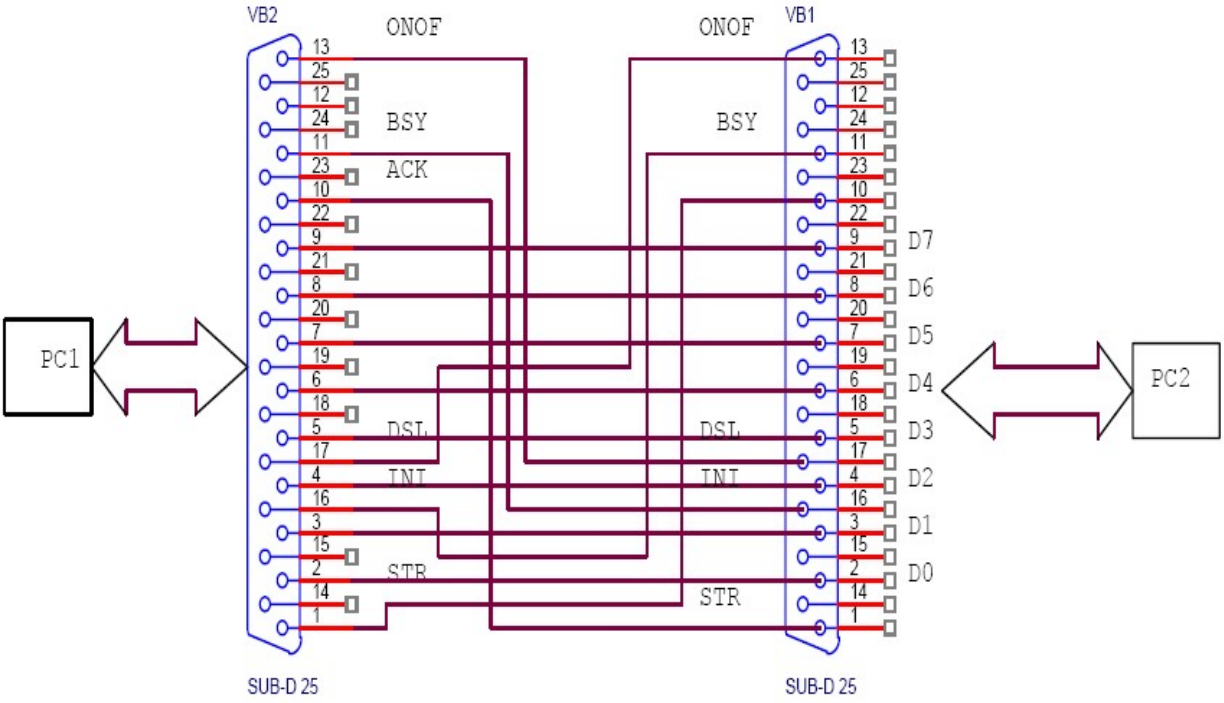
Chân	Tín hiệu	Mô tả
1	STR	Mức tín hiệu thấp, truyền dữliệu tới máy in
2	D0	Bit dữ liệu 0
3	D1	Bit dữ liệu 1
4	D2	Bit dữ liệu 2
5	D3	Bit dữ liệu 3
6	D4	Bit dữ liệu 4
7	D5	Bit dữ liệu 5
8	D6	Bit dữ liệu 6
9	D7	Bit dữ liệu 7
10	ACK	Mức thấp: máy in đã nhận 1 ký tự và có khả năng nhận nữa
11	BSY	Mức cao: 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

Bảng 8-2 Tín hiệu chân của cổng LPT

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 8-5 Trao đổi dữ liệu qua cổng song song giữa 2 PC

8.4.2 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:

D25	D9	Tín hiệu	Hướng truyền	Mô tả
1	-	-	-	Protected ground: nối đất bảo vệ
2	3	TxD	DTE□DCE	Transmitted data: dữliệu phát
3	2	RxD	DCE□DTE	Received data: dữliệu thu
4	7	RTS	DTE□DCE	Request to send: DTE yêu cầu truyền dữliệu
5	8	CTS	DCE□DTE	Clear to send: DCE sẵn sàng nhận dữliệu
6	6	DSR	DCE□DTE	Data set ready: DCE sẵn sàng làm việc
7	5	GND	-	Ground: nối đất (0V)
8	1	DCD	DCE□DTE	Data carier detect: DCE phát hiện sóng mang
20	4	DTR	DTE□DCE	Data terminal ready: DTE sẵn sàng làm việc
22	9	RI	DCE□DTE	Ring indicator: báo chuông
23	-	DSRD	DCE□DTE	Data signal rate detector: dò tốc độtruyền

Bảng 8-3 Tín hiệu chân của cổng nối tiếp

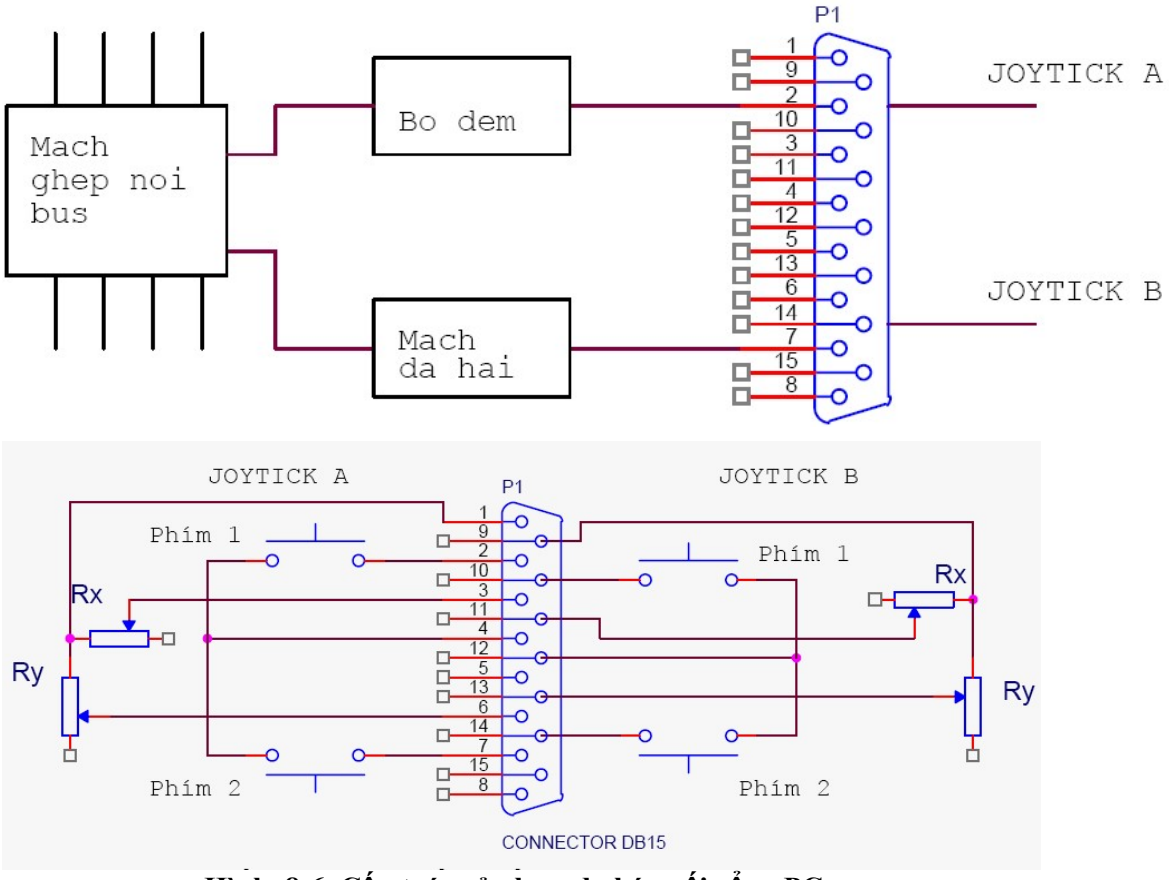
Chuẩn RS-232C cho phép truyền tín hiệu với tốc độ đến 20.000 bps nhưng nếu cáp truyền đủ ngắn có thể lên đến 115.200 bps. Chiều dài cáp cực đại là 17-20m.

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.

8.4.3 Cổng PC-Game

Cấu trúc và chức năng của board ghép nối trò chơi (PC game) như hình bên dưới.



Hình 8-6 Cấu trúc của board ghép nối 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)

Bảng 8-4 Tín hiệu chân của cổng PC-game

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:

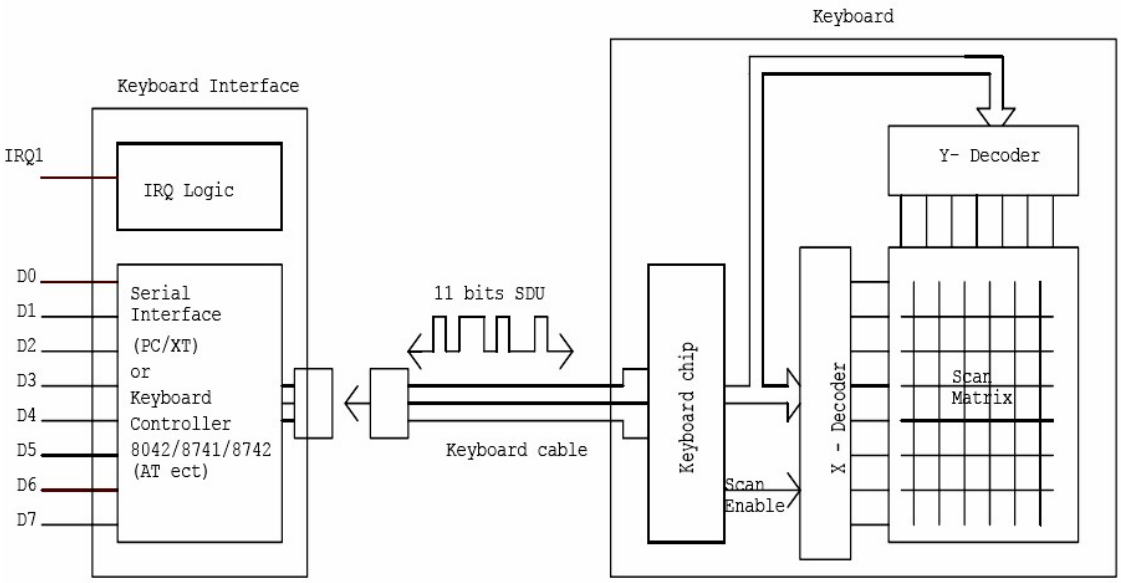
D7	D6	D5	D4	D3	D2	D1	D0
BB2	BB1	BA2	BA1	BY	BX	AY	AX

Bảng 8-5 Byte trạng thái của board game

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.

8.4.4 Cổng bàn phím



Hình 8-7 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)

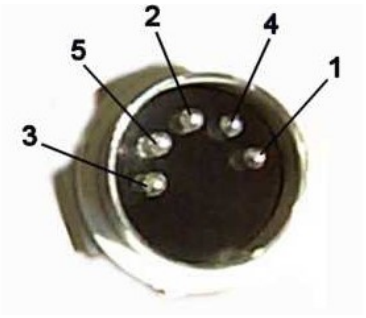
STOP: bit stop (luôn bằng 1).

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

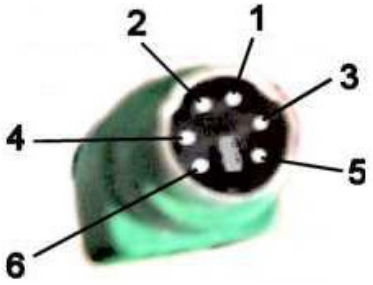
chíp bàn phím có thể giúp cho việc nhận lệnh điều khiển từ PC, thí dụ như đặt tốc độ lặp lại của nhấn bàn phím,….

Chân 1: clock
Chân 2: dữ liệu
Chân 3: Reset
Chân 4: GND
Chân 5: Vcc



Hình 8-8 Đầu cắm bàn phím AT

Chân 1: dữ liệu
Chân 2: không dùng
Chân 3: GND
Chân 4: Vcc
Chân 5: clock
Chân 6: không dùng



Hình 8-9 Đầu cắm bàn phím PS/2

IV. Bài tập củng cố kiến thức

8.1 Trong hệ thống máy tính, tập hợp các thiết bị vào và thiết bị ra có tên gọi chung là gì? Nêu một số đặc điểm chính của các thiết bị này?

8.2 Trong hệ thống máy tính thiết bị vào/ra được chia thành những nhóm cơ bản nào? Hãy cho biết một số thiết bị thuộc các nhóm trên.

8.3 Mô tả quy trình đọc dữ liệu từ thiết bị ngoại vi của bộ xử lý.

8.4 Trình bày chức năng và cấu trúc của Modul I/O

8.5 Vào ra bằng phương pháp Polling là gì? Trình bày ưu nhược điểm của phương pháp này

TÀI LIỆU THAM KHẢO

[1]. Bài giảng ***"Kiến trúc và tổ chức máy tính"*** , bộ môn Kỹ thuật máy tính – Khoa Điện Tử.

[2]. Bài giảng ***"Cấu trúc máy tính"*** , bộ môn Kỹ thuật máy tính – Khoa Điện Tử.

- **Sách tham khảo:**

[3]. Nguyễn Đình Việt, Kiến trúc máy tính, NXB Giáo dục, 2000.

[4]. Tống Văn On, Cấu trúc máy tính cơ bản, NXB Thống kê, 2001.

[5]. Tống Văn On, Cấu trúc máy tính nâng cao, NXB Thống kê, 2001.

[6]. Trần Quang Vinh, Nguyên lý phần cứng và kỹ thuật ghép nối máy tính, NXB Giáo dục, 2002.

[7]. Tống Văn On, Hoàng Đức Hải, Giáo trình cấu trúc máy tính, NXB Giáo dục, 2000.

[8]. Nguyễn Nam Trung, *Cấu trúc máy vi tính và thiết bị ngoại vi*, NXB KHKT, 2000.