TỔNG QUAN VỀ CẤU TRÚC MÁY TÍNH VÀ HỢP NGỮ

1.1. Giới thiệu

Máy tính số (Digital computer) là máy giải quyết các vấn đề bằng cách thực hiện các chỉ thị do con người cung cấp. Chuỗi các chỉ thị này gọi là chương trình (program). Các mạch điện tử trong một máy tính số sẽ thực hiện một số giới hạn các chỉ thị đơn giản cho trước. Tập hợp các chỉ thị này gọi là tập lệnh của máy tính. Tất cả các chương trình muốn thực thi đều phải được biến đổi sang tập lệnh trước khi được thi hành. Các lệnh cơ bản là:

- Cộng 2 số.
- So sánh với 0.
- Di chuyển dữ liệu.

Tập lệnh của máy tính tạo thành một ngôn ngữ giúp con người có thể tác động lên máy tính, ngôn ngữ này gọi là ngôn ngữ máy (machine language). Tuy nhiên, hầu hết các ngôn ngữ máy đều đơn giản nên để thực hiện một yêu cầu nào đó, người thiết kế phải thực hiện một công việc phức tạp. Đó là chuyển các yêu cầu này thành các chỉ thị có chứa trong tập lệnh của máy. Vấn đề này có thể giải quyết bằng cách thiết kế một tập lệnh mới thích hợp cho con người hơn tập lệnh đã cài đặt sẵn trong máy (built-in). Ngôn ngữ máy sẽ được gọi là ngôn ngữ cấp 1 (L1) và ngôn ngữ vừa được hình thành gọi là ngôn ngữ cấp 2 (L2).

Một phương pháp thực thi chương trình L2 là chuyển một lệnh trong L2 bằng một chuỗi các lệnh tương đương trong L1. Kết quả là sẽ tạo thành một chương trình L1 và máy tính sẽ thực hiện chương trình tương đương L1 thay vì thực hiện chương trình L2. Kỹ thuật này gọi là biên dịch (compile). Cách khác là một lệnh trong chương trình L2 sẽ được xem như dữ liệu ngõ vào của chương trình L1 và toàn bộ chương trình L2 sẽ được thực thi tuần tự. Kỹ thuật này gọi là thông dịch (interprete), nó không yêu cầu tạo ra một chương trình mới trong L1.

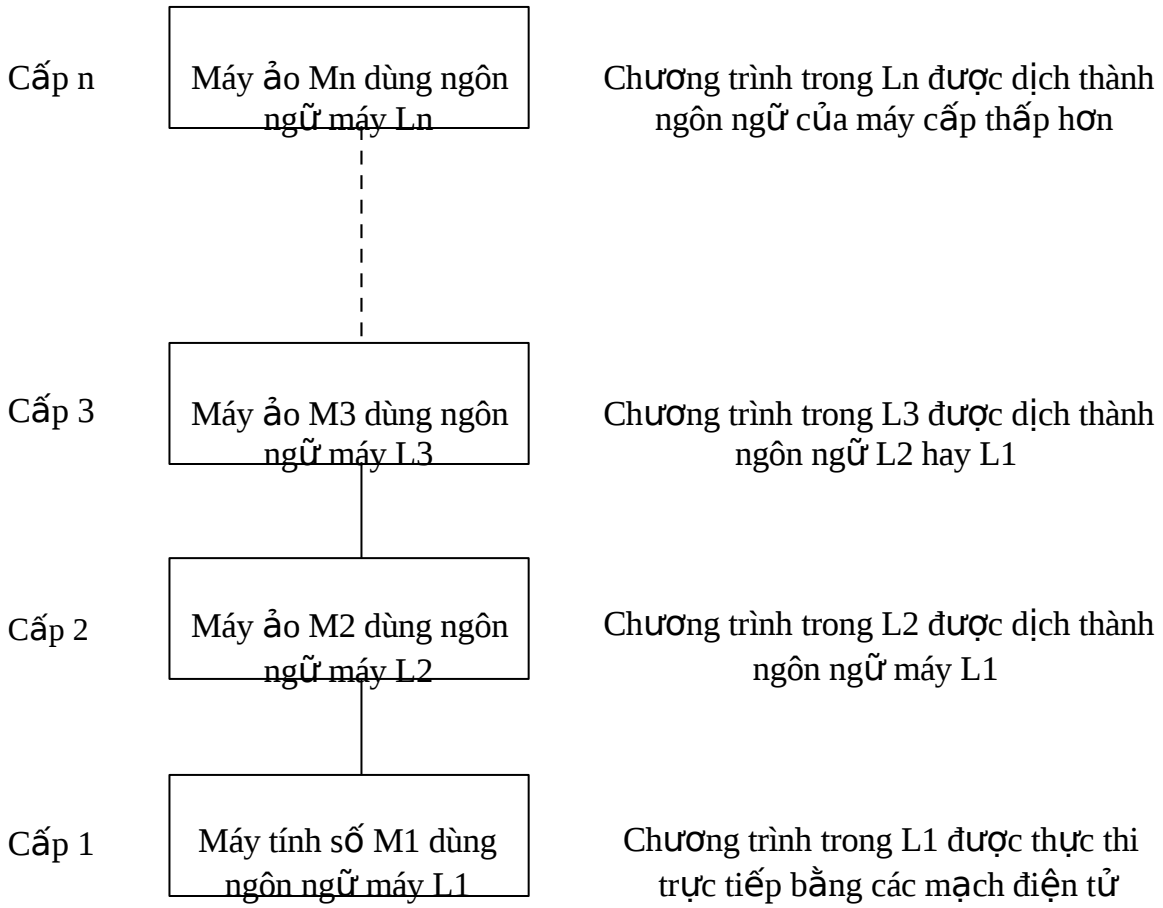
Biên dịch và thông dịch đều thực hiện chương trình L2 thông qua tập lệnh trong chương trình L1. Chúng khác nhau ở chỗ là khi biên dịch thì toàn bộ chương trình L2 sẽ được chuyển thành chuỗi lệnh L1 rồi sau đó mới được thực thi còn đối với phương pháp thông dịch thì sẽ thực thi từng lệnh trong L2. Để thuận tiện hơn, ta giả sử tồn tại một máy tính sử dụng ngôn ngữ máy là L2, ta gọi máy tính này là máy ảo (virtual machine).

Tuy nhiên, trong thực tế, để có thể thực hiện biên dịch và thông dịch , các ngôn

ngữ L1 và L2 không được khác nhau nhiều. Như vậy, ngôn ngữ L2 cũng không thật sự giúp ích nhiều cho người thiết kế. Do đó, một tập lệnh kế tiếp được hình thành sẽ hướng về con người nhiều hơn là máy tính, tập lệnh này sẽ tạo thành một ngôn ngữ và ta gọi là ngôn ngữ L3. Ta có thể viết các chương trình trong L3 như là đã tồn tại máy tính sử dụng

ngôn ngữ L3 (máy ảo L3). Các chương trình này sẽ được dịch sang ngôn ngữ L2 và được thực thi bằng một chương trình dịch L2.

Việc xây dựng toàn bộ chuỗi các ngôn ngữ, mỗi ngôn ngữ được tạo ra sẽ thích hợp hơn ngôn ngữ trước đó sẽ có thể tiếp tục cho đến khi nhận được ngôn ngữ thích hợp nhất. Sơ đồ một máy ảo n cấp có thể biểu diễn như sau:

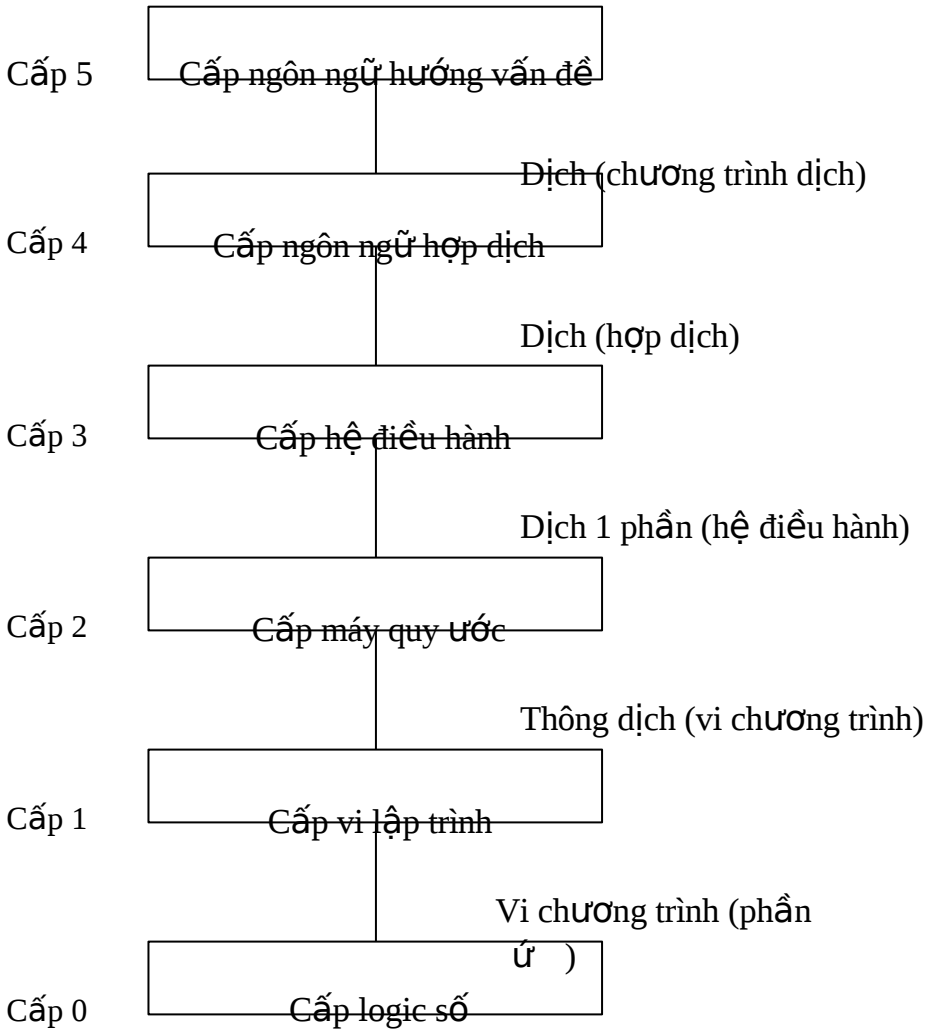


Hình 1.1. Máy ảo n cấp

Một máy tính số có n cấp có thể xem như có n-1 máy ảo khác nhau, mỗi máy ảo có một ngôn ngữ máy riêng. Các chương trình viết trên các máy ảo này không thể thực thi trực tiếp mà phải dịch thành các ngôn ngữ máy cấp thấp hơn. Chỉ có máy thật dùng ngôn ngữ máy L1 mới có thể thực thi trực tiếp bằng các mạch điện tử. Một lập trình viên sử dụng máy ảo cấp n không cần biết tất cả các trình dịch này. Chương trình trong máy ảo cấp n sẽ được thực thi bằng cách dịch thành ngôn ngữ máy cấp thấp hơn và ngôn ngữ máy này sẽ được dịch thành ngôn ngữ máy thấp hơn nữa hay dịch trực tiếp thành ngôn ngữ máy L1 và thực thi trực tiếp trên các mạch điện tử.

1.2. Máy nhiều cấp

Hầu hết các máy tính hiện nay gồm có 6 cấp:



Hình 1.2 – Các cấp trên máy tính số

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).

1.3. Quá trình phát triển của máy nhiều cấp

Các máy tính đầu tiên trong thập niên 40 chỉ có 2 cấp: cấp máy quy ước và cấp logic số. Các lập trình viên phải làm việc trên cấp máy quy ước và chương trình được thực thi trên cấp logic số. Trong thập niên 50, Wikes đề xuất ý tưởng thiết kế máy tính 3 cấp. Máy tính này có một trình thông dịch cài đặt sẵn, không thay đổi, có nhiệm vụ thực thi các chương trình trong cấp máy quy ước. Như vậy, phần cứng chỉ thực thi các vi chương trình với số lệnh giới hạn nên các mạch điện tử cũng đơn giản hơn.

Trình dịch hợp ngữ (assembler) và các trình biên dịch cho ngôn ngữ cấp cao (compiler) phát triển vào những năm 50 tạo điều kiện dễ dàng hơn cho lập trình viên. Tuy nhiên, vào lúc này, lập trình viên phải tự điều hành máy. Vào những năm 60, việc tự động hóa công việc điều hành bắt đầu được thực hiện. Một chương trình gọi là hệ điều hành (operating system) luôn được lưu trữ bên trong máy tính. Lập trình viên cung cấp các thẻ điều khiển và chương trình, chúng sẽ được đọc và thực thi bằng hệ điều hành.

Trong nhiều năm tiếp theo, hệ điều hành càng trở nên phức tạp. Các lệnh, tiện ích và đặc trưng mới được thêm vào cấp máy quy ước cho đến khi xuất hiện một cấp mới. Một số lệnh của cấp mới này giống như cấp máy quy ước nhưng một số lệnh lại hoàn toàn khác, nhất là các lệnh xuất nhập. Vào những năm đầu thập niên 60, các nghiên cứu ở đại học Dartmouth, MIT đã phát triển các hệ điều hành cho phép lập trình viên có thể tác động trực tiếp lên máy tính. Trong các hệ thống này, thiết bị đầu cuối từ xa được nối với máy tính trung tâm qua các đường điện thoại. Một lập trình viên có thể gõ chương trình và nhận kết quả trả về tức thời ở bất cứ nơi nào có thiết bị đầu cuối. Các hệ thống này gọi là hệ thống chia sẻ thời gian (time-sharing system).

2. Phần cứng và phần mềm (Hardware and software)

Các chương trình viết bằng ngôn ngữ máy (cấp 1) được thực thi trực tiếp bằng các mạch điện tử của máy tính, không có trình thông dịch và biên dịch nào can thiệp vào. Các mạch điện tử cùng với bộ nhớ và các thành phần xuất / nhập tạo nên phần cứng máy tính.

Phần cứng bao gồm các mạch tích hợp, các board mạch in, cable, nguồn cung cấp, bộ nhớ, thiết bị đầu cuối, …

Phần mềm bao gồm các giải thuật và các biểu diễn của các giải thuật này gọi là chương trình. Nó chính là tập hợp các lệnh tạo thành một chương trình, chứ không phải là các phương tiện vật lý lưu trữ chúng.

Một dạng trung gian giữa phần mềm và phần cứng gọi là phần dẻo (firmware). Nó chính là thành phần bao gồm phần mềm được đặt vào bên trong các mạch điện tử trong quá trình sản xuất. Phần dẻo được dùng khi chương trình không thay đổi hay hiếm khi phải thay đổi như chương trình điều khiển đặt trong ROM BIOS.

Một thao tác bất kỳ thực thi bằng phần mềm có thể được gắn trực tiếp vào phần cứng và một lệnh bất kỳ thực thi bằng phần cứng cũng có thể được mô phỏng bằng phần mềm. Quyết định đặt một số chức năng vào phần mềm và các chức năng khác vào phần cứng dựa trên các yếu tố giá thành, tốc độ, độ tin cậy. Trên nhiều máy tính đầu tiên, phần cứng và phần mềm được phân biệt rõ ràng. Phần cứng thực hiện vài lệnh đơn giản như cộng và nhảy, các thủ tục khác phải do lập trình viên tự thiết kế. Sau đó, một số thao tác thường xuyên thực thi đòi hỏi các nhà thiết kế hướng đến yêu cầu xây dựng các mạch điện từ thực thi các thao tác này. Kết quả là hình thành xu hướng di chuyển các thao tác theo hướng từ cấp cao xuống cấp thấp hơn. Một số thao tác trước đây được lập trình ở cấp máy quy ước, sau đó được chuyển xuống thực thi ở phần cứng.

Tuy nhiên, khi xuất hiện thế hệ máy tính dùng vi lập trình và thế hệ máy tính nhiều cấp, lại xuất hiện xu hướng ngược lại, nghĩa là di chuyển các thao tác từ cấp thấp lên cấp cao hơn. Ví dụ như lệnh cộng sẽ được thực hiện trực tiếp bằng phần cứng ở các máy trước kia. Đối với máy tính được vi lập trình hóa, lệnh cộng của cấp máy quy ước được thông dịch bằng một vi chương trình chạy trên cấp thấp nhất và được thực thi bằng một chuỗi các bước nhỏ: tìm lệnh, nạp lệnh, xác định lệnh, định vị dữ liệu, tìm và nạp dữ liệu từ bộ nhớ, thực thi phép cộng và lưu trữ kết quả.

Một số đặc trưng trước đây được lập trình ở cấp máy quy ước, sau đó được thực hiện bằng phần cứng hay vi chương trình:

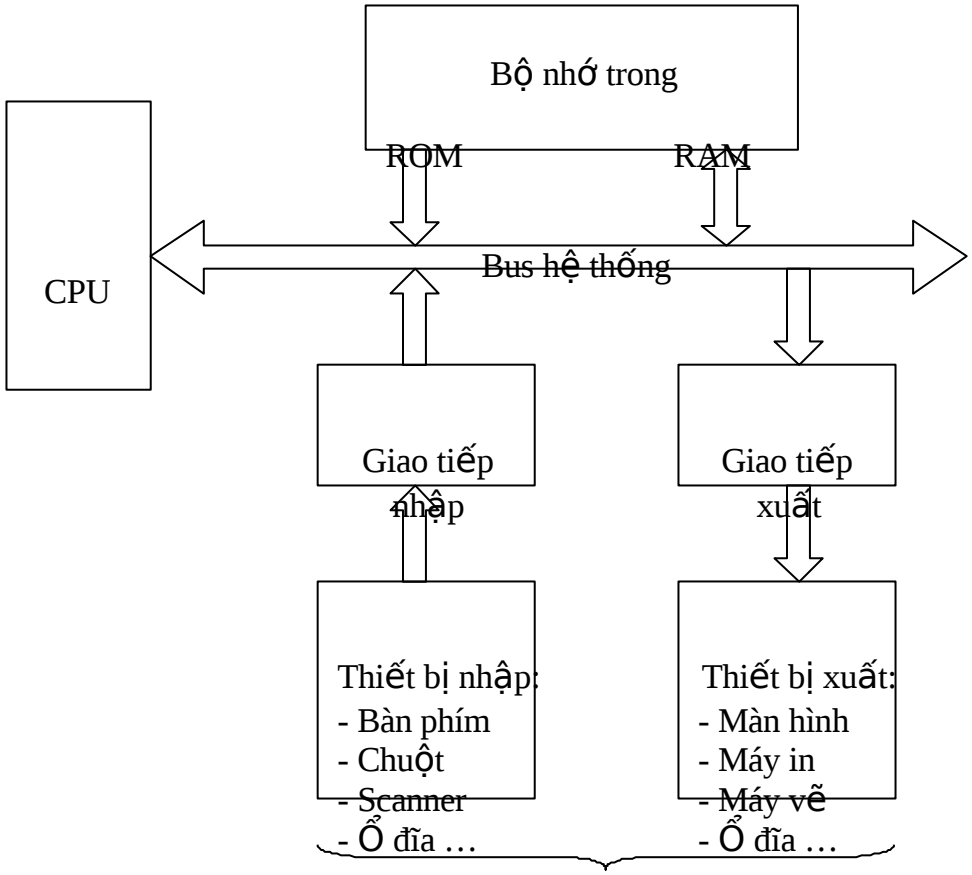
- Các lệnh nhân, chia số nguyên.
- Các lệnh xử lý dấu chấm động.
- Các lệnh gọi thủ tục và quay về từ lệnh gọi thủ tục.
- Các lệnh đếm.
- Các lệnh quản lý chuỗi ký tự.
- Các đặc trưng làm tăng tốc độ tính toán chuỗi: định địa chỉ chỉ số và định địa chỉ gián tiếp.
- Các đặc trưng cho phép chương trình di chuyển trong bộ nhớ sau khi đã thực thi (cấp phát lại bộ nhớ).
- Các xung clock cho thủ tục định thời.
- Các ngắt báo hiệu cho máy tính.

- Khả năng chuyển đổi quá trình.

Như vậy, ta thấy ranh giới giữa phần cứng và phần mềm là không nhất định và thường xuyên thay đổi. Theo quan điểm của lập trình viên, cách thức thực thi một lệnh là không quan trọng, ngoại trừ tốc độ thực thi. Như vậy, phần cứng của người này có thể là phần mềm của người kia.Từ đó dẫn đến ý tưởng thiết kế máy tính có cấu trúc (structured computer). Đó là cấu trúc một máy tính thành một chuỗi các cấp, lập trình viên làm việc trên cấp n không quan tâm đến các cấp khác.

3. Tổ chức hệ thống máy tính

3.1. Cấu trúc một hệ thống máy tính

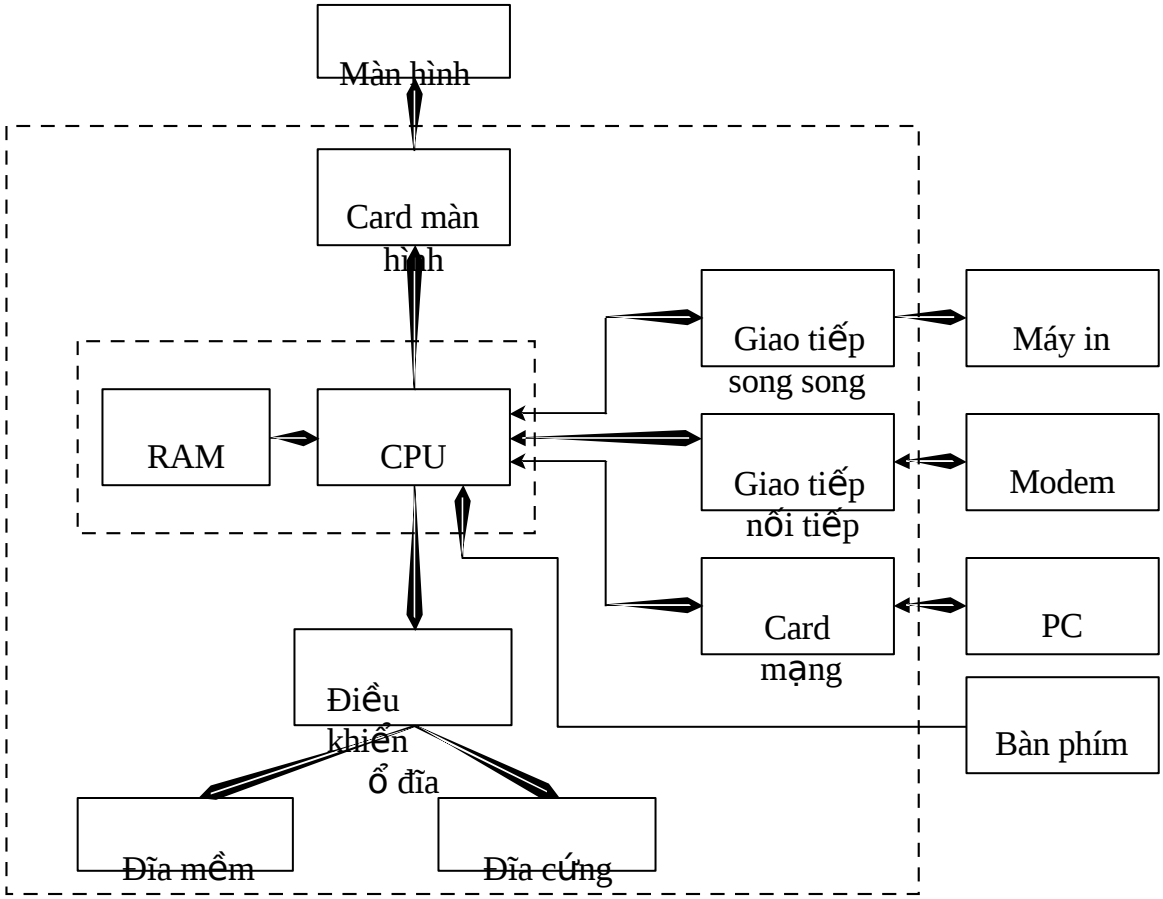


Hình 1.3 – Sơ đồ khối một hệ thống máy tính

Sơ đồ khối của một hệ thống máy vi tính có thể mô tả như hình vẽ. Nó bao gồm các khối:

- ***Khối xử lý trung tâm (CPU – Central Processing Unit)***: nhận và thực thi các lệnh. Bên trong CPU gồm các mạch điều khiển logic, mạch tính toán số học, …
- ***Bộ nhớ (Memory)***: lưu trữ các lệnh và dữ liệu. Nó bao gồm 2 loại: bộ nhớ trong và bộ nhớ ngoài. Bộ nhớ thường được chia thành các ô nhớ nhỏ. Mỗi ô nhớ được gán một địa chỉ để CPU có thể định vị khi cần đọc hay ghi dữ liệu.
- ***Thiết bị ngoại vi (Input / Output)***: dùng để nhập hay xuất dữ liệu. Bàn phím, chuột, scanner, … thuộc thiết bị nhập; màn hình, máy in, … thuộc thiết bị xuất. Các ổ đĩa thuộc bộ nhớ ngoài cũng có thể coi vừa là thiết bị xuất vừa là thiết bị nhập. Các thiết bị ngoại vi liên hệ với CPU qua các mạch giao tiếp I/O (I/O interface)/
- ***Bus hệ thống***: tập hợp các đường dây để CPU có thể liên kết với các bộ phận khác.

3.2. Hoạt động của máy tính



Hình 1.4 – Sơ đồ khối một PC với các thiết bị ngoại vi

CPU được nối với các thành phần khác bằng bus hệ thống nghĩa là sẽ có nhiều thiết bị cùng dùng chung một hệ thống dây dẫn để trao đổi dữ liệu. Do đó, để hệ thống không bị xung đột, CPU phải xử lý sao cho trong một thời điểm, chỉ có một thiết bị hay ô nhớ đã chỉ định mới có thể chiếm dụng bus hệ thống. Do mục đích này, bus hệ thống bao gồm 3 loại:

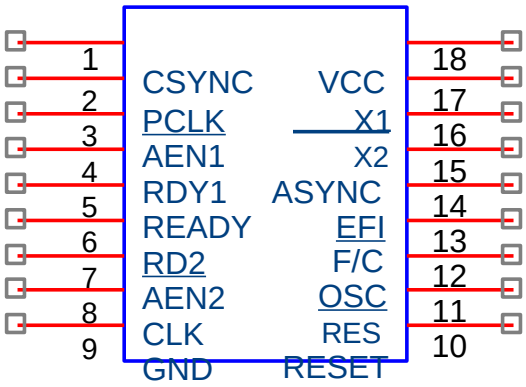
- Bus dữ liệu (data bus): truyền tải dữ liệu
- Bus địa chỉ (address bus): chọn ô nhớ hay thiết bị ngoại vi
- Bus điều khiển (control bus): hỗ trợ trao đổi thông tin trạng thái như phân biệt CPU phải truy xuất bộ nhớ hay ngoại vị, thao tác xử lý là đọc/ghi, …

CPU phát tín hiệu địa chỉ của thiết bị lên bus địa chỉ. Tín hiệu này được dưa vào mạch giải mã địa chỉ chọn thiết bị. Bộ giải mã sẽ phát ra chỉ một tín hiệu chọn chip đúng sẽ cho phép mở bộ đệm của thiết bị cần thiết, dữ liệu lúc này sẽ được trao đổi giữa CPU và thiết bị. Trong quá trình này, các tín hiệu điều khiển cũng được phát trên control bus để xác định mục đích của quá trình truy xuất.

3.3. Các chip hỗ trợ

3.3.1. Mạch tạo xung clock 8284

Mạch tạo xung clock dùng để cung cấp xung clock cho CPU.



Hình 1.5 – Mạch tạo xung clock 8284

CSYNC (Clock Synchronisation): ngõ vào xung đồng bộ chung khi hệ thống có các 8284 dùng dao động ngoài tại chân EFI. Khi dùng mạch dao động trong thì phải nối GND.

PCLK (Peripheral Clock): xung clock f = f_x/6 (f_x là tần số thạch anh) với chu kỳ bổn phận 50%.

AEN 1 , ***AEN 2*** (Address Enable): cho phép chọn các chân tương ứng RDY1, RDY2 báo hiệu trạng thái sẵn sàng của bộ nhớ hay thiết bị ngoại vi.

RDY1, RDY2 (Bus ready): kết hợp với AEN1, AEN2 tạo các chu kỳ đợi ở CPU

READY: nối đến chân READY của µP.

CLK (Clock): xung clock f = f_x/3, nối với chân CLK của CPU.

RESET: nối với chân RESET của CPU, là tín hiệu khởi động lại toàn hệ thống.

RES (Reset Input): chân khởi động cho 8284, được nối với mạch RC để tự khởi động khi bật nguồn.

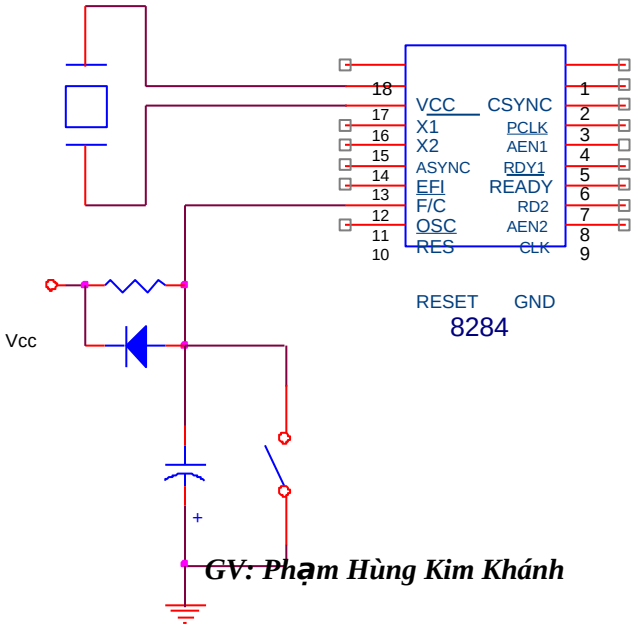
OSC: ngõ ra xung clock có tần số f_x.

F/C (Frequency / Crystal): chọn nguồn tín hiệu chuẩn cho 8284, nếu ở mức cao thì chọn tần số xung clock bên ngoài, ngược lại thì dùng xung clock từ thạch anh.

EFI (External Frequency Input): xung clock từ bộ dao động ngoài.

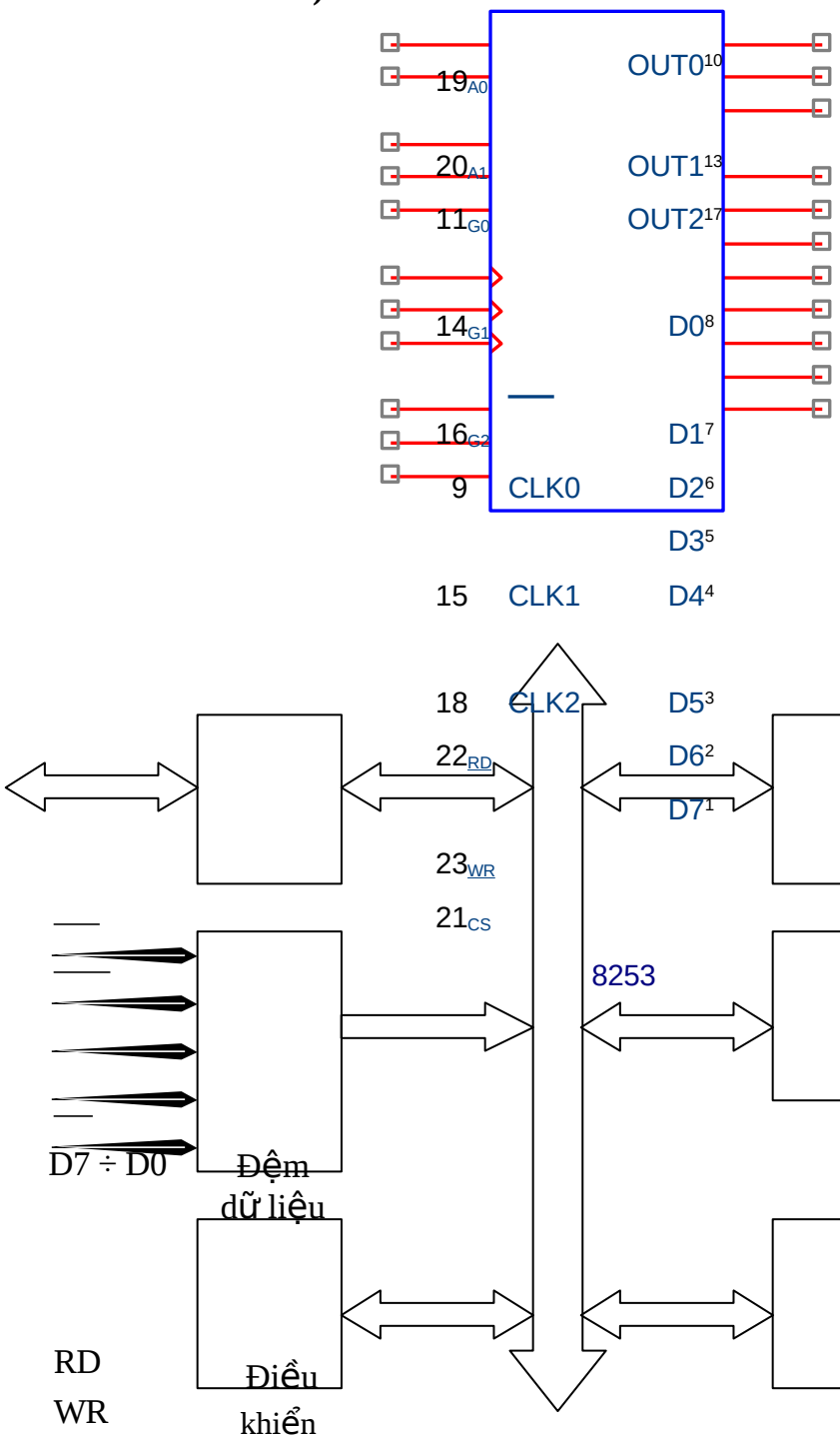
ASYNC : chọn chế độ làm việc cho tín hiệu RDY. Nếu ASYNC = 1, tín hiệu RDY có ảnh hưởng đến tín hiệu READY cho đến khi có xung âm của xung clock. Ngược lại thì RDY chỉ ảnh hưởng khi xuất hiện xung âm.

X1,X2: ngõ vào của thạch anh, dùng để tạo xung chuẩn cho hệ thống.

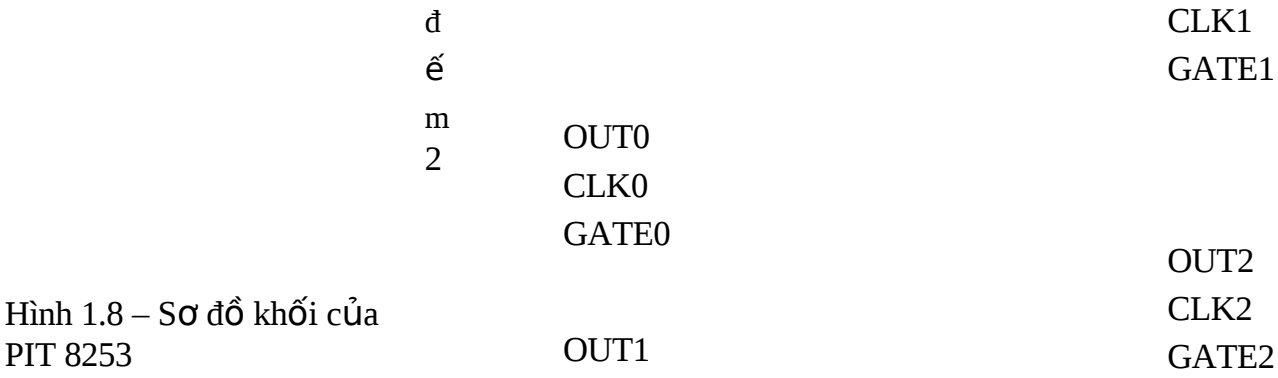


Hình 1.6 – Mạch khởi động cho 8284

3.3.2. Mạch định thời PIT – 8253 / 8254 (Programmable Interval Timer)



Hình 1.7 – Sơ đồ chân của PIT 8253



Hình 1.8 – Sơ đồ khối của PIT 8253

D7 ÷ D0: bus dữ liệu

CLK0 ÷ CLK2: ngõ vào xung clock cho các bộ đếm

OUT0 ÷ OUT2: ngõ ra bộ đếm

BUS NỘI

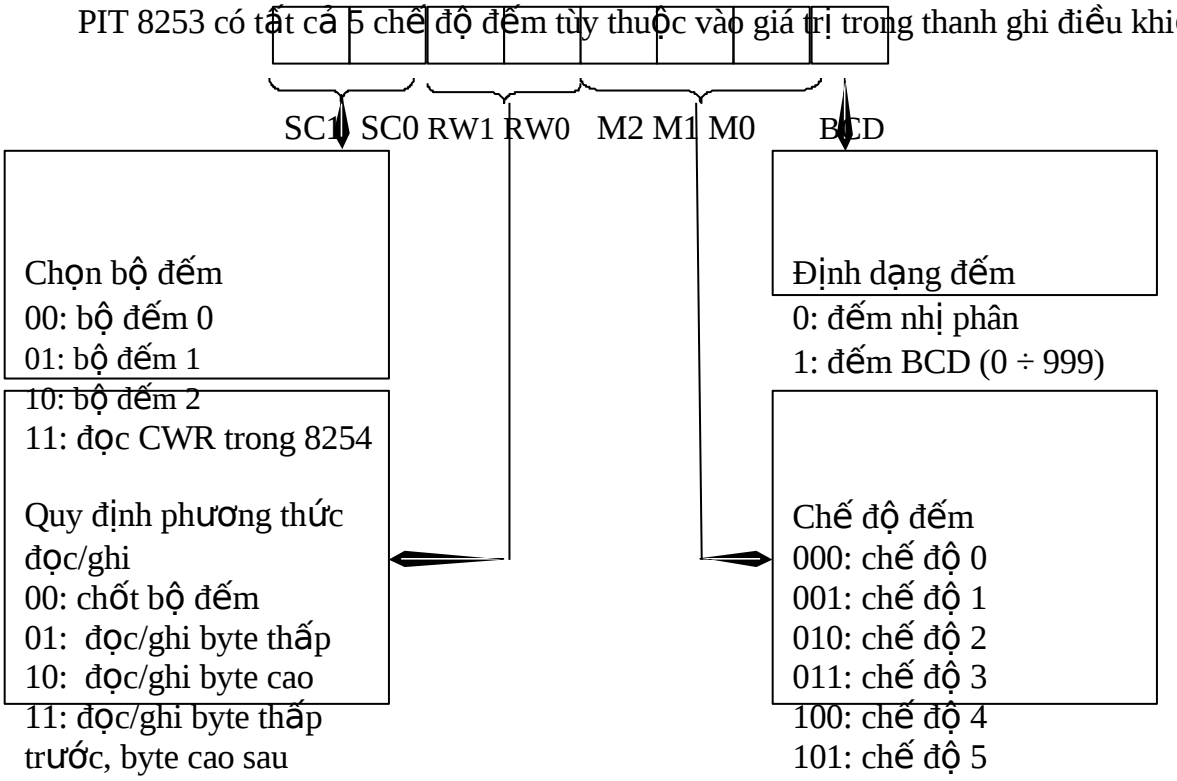
RD , ***WR*** : cho phép CPU đọc / ghi dữ liệu từ / đến các thanh ghi của 8253

A1, A0: giải mã chọn bộ đếm hay thanh ghi điều khiển, thường được nối với bus địa chỉ của CPU

A1	A0	Chọn
0	0	Bộ đếm 0
0	1	Bộ đếm 1
1	0	Bộ đếm 2
1	1	Thanh ghi từ điều khiển

G0 ÷ ***G2*** (Gate): cho phép hay cấm các bộ đếm hoạt động (=1: cho phép, =0: cấm).

PIT 8253 có tất cả 5 chế độ đếm tùy thuộc vào giá trị trong thanh ghi điều khiển.



Hình 1.9 – Dạng từ điều khiển của 8253

PIT 8253 có 3 bộ đếm lùi 16 bit có thể lập trình và độc lập với nhau. Mỗi bộ đếm có tín hiệu xung clock riêng (8254 tương tự như 8253 nhưng có thêm lệnh đọc thanh ghi từ điều khiển CWR). Địa chỉ các thanh ghi của PIT đối với PC là:

Port (1)	Port (2)	Thanh ghi
40h	48h	Bộ đếm 0
41h	49h	Bộ đếm 1
42h	4Ah	Bộ đếm 2
43h	4Bh	CWR

- **Các chế độ đếm:**

Chế độ 0 (Interrupt on Terminal Count): tín hiệu ngõ ra ở mức thấp cho tới khi bộ đếm tràn thì sẽ chuyển lên mức cao.

Chế độ 1 (Programmable Monoflop): tín hiệu ngõ ra chuyển xuống mức thấp tại cạnh âm của xung clock đầu tiên và sẽ chuyển lên mức cao khi bộ đếm kết thúc.

Chế độ 2 (Rate Generator): tín hiệu ngõ ra xuống mức thấp trong chu kỳ đầu tiên và sau đó chuyển lên mức cao trong các chu kỳ còn lại.

Chế độ 3 (Square-Wave Generator): tương tự như chế độ 2 nhưng xung ngõ ra là sóng vuông khi giá trị đếm chẵn và sẽ thêm một chu kỳ ở mức cao khi giá trị đếm lẻ.

Chế độ 4 (Software-triggered Pulse): giống như chế độ 2 nhưng xung Gate không khởi động quá trình đếm mà sẽ đếm ngay khi số đếm ban đầu được nạp. Ngõ ra ở mức cao để đếm và xuống mức thấp trong chu kỳ xung đếm. Sau đó, ngõ ra sẽ trở lại mức cao.

Chế độ 5 (Hardware-triggered Pulse): giống như chế độ 2 nhưng xung Gate không khởi động quá trình đếm mà được khởi động bằng cạnh dương của xung clock ngõ vào. Ngõ ra ở mức cao và xuống mức thấp sau một chu kỳ clock khi quá trình đếm kết thúc.

- **Ba chức năng của 8253 trong PC:**

Cập nhật đồng hồ hệ thống: bộ đếm 0 của PIT phát tuần hoàn một ngắt cứng qua IRQ0 của 8259 để CPU có thể thay đổi đồng hồ hệ thống. Bộ đếm hoạt động trong chế độ 2. Ngõ vào được cấp xung clock tần số 1.19318 MHz. G0 = 1 để bộ đếm luôn được phép đếm. Giá trị ban đầu được nạp là 0 cho phép PIT phát ra xung chính xác với tần số:1.19318/65536 = 18.206Hz. Cạnh dương của mỗi xung này sẽ tạo ra một ngắt cứng trong 8259. Yêu cầu này sẽ dẫn tới ngắt 08h để cập nhật đồng hổ hệ thống 18.206 lần trong 1 giây.

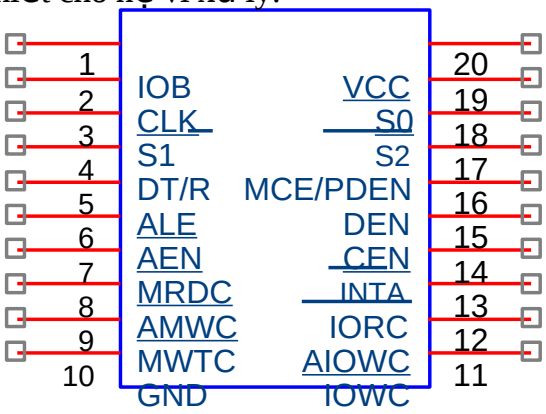
Làm tươi bộ nhớ: PIT nối với chip DMAC dùng làm tươi bộ nhớ DRAM. Bộ đếm 1 sẽ định kỳ kích hoạt kênh 0 của DMAC-8237A để tiến hành 1 chu trình đọc giả làm tươi bộ nhớ. Bộ nhớ 1 hoạt động trong chế độ 3 phát sóng vuông với giá trị nạp ban đầu là 18. Do đó sóng vuông được phát ra có tần số 1,19318 MHz/18 = 66288 Hz (chu kỳ bằng 0.015s). Như vậy cứ sau 15 ms cạnh dương của sóng vuông này sẽ tạo 1 chu kỳ đọc giả để làm tươi bộ nhớ.

Phát sóng âm với tần số biến đổi ra loa của PC: Bộ đếm 2 của PIT được dùng để

phát sóng âm ra loa của PC.

3.3.3. Mạch điều khiển bus 8288

Mạch điều khiển bus 8288 lấy một số tín hiệu điều khiển của CPU và cung cấp các tín hiệu điều khiển cần thiết cho hệ vi xử lý.



Hình 1.10 – Mạch điều khiển bus 8288

IOB (Input / Output Bus Mode): điều khiển để 8288 làm việc ở các chế độ bus khác nhau.

CLK (Clock): ngõ vào lấy từ xung clock hệ thống (từ 8284) và dùng để đồng bộ toàn bộ các xung điều khiển đi ra từ mạch 8288.

S 2 , ***S1*** , ***S0*** : các tín hiệu trạng thái lấy trực tiếp từ CPU. Tuỳ theo các giá trị nhận được mà 8288 sẽ đưa các tín hiệu theo bảng:

S2	S1	S0	Tạo tín hiệu
0	0	0	INTA
0	0	1	IORC
0	1	0	IOWC , AIOWC
0	1	1	Không
1	0	0	MRDC
1	0	1	MRDC
1	1	0	MWTC , AMWC
1	1	1	Không

DT/ R (Data Transmit/Receive): CPU truyền (1) hay nhận (0) dữ liệu.

ALE (Address Latch Enable): tín hiệu cho phép chốt địa chỉ, tín hiệu này thường được nối với chân G của 74573 để điều khiển chốt địa chỉ.

AEN (Address Enable): chờ thời gian trễ khoảng 150 ns sẽ tạo các tín hiệu điều

khiển ở đầu ra của 8288 để đảm bảo rằng địa chỉ sử dụng đã hợp lệ.

MRDC (Memory Read Command): điều khiển đọc bộ nhớ

MWTC (Memory Write Command): điều khiển ghi bộ nhớ

AMWC (Advanced MWTC),: giống như MWTC nhưng hoạt động sớm hơn một chút dùng cho các bộ nhớ chậm đáp ứng kịp tốc độ CPU.

IOWC (I/O Write Command): điều khiển ghi ngoại vi

AIOWC (Advanced IOWC),: giống như IOWC nhưng hoạt động sớm hơn một chút dùng cho các ngoại vi chậm đáp ứng kịp tốc độ CPU.

IORC (I/O Read Command): điều khiển đọc ngoại vi

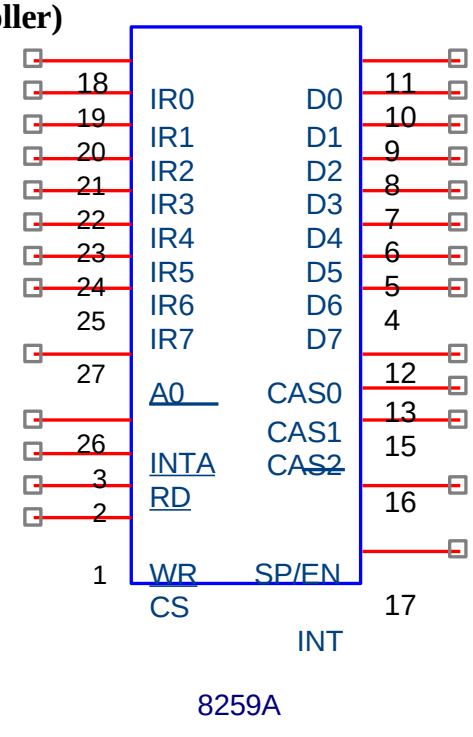
INTA (Interrupt Acknowledge): ngõ ra thông báo CPU chấp nhận yêu cầu ngắt của thiết bị ngoại vi

CEN (Command Enable): cho phép đưa ra tín hiệu DEN và các tín hiệu điều khiển khác của 8288.

DEN (Data Enable): điều khiển bus dữ liệu thành bus cục bộ hay bus hệ thống.

MCE / ***PDEN*** (Master Cascade Enable / Peripheral Data Enable): định chế độ làm việc cho mạch điều khiển ngắt PIC 8259 để nó làm việc ở chế độ master.

3.3.4. Chip điều khiển ngắt ưu tiên PIC 8259A (Priority Interrupt Controller)



Hình 1.11 – Sơ đồ chân của 8259A

Trong trường hợp nhiều yêu cầu ngắt cần phải phục vụ, ta thường dùng vi mạch 8259A để giải quyết vấn đề ưu tiên. 8259A có thể giải quyết được 8 yêu cầu ngắt với 8 mức ưu tiên khác nhau.

- **Các khối chức năng:**

IRR (thanh ghi yêu cầu ngắt): lưu trữ các yêu cầu ngắt tại ngõ vào

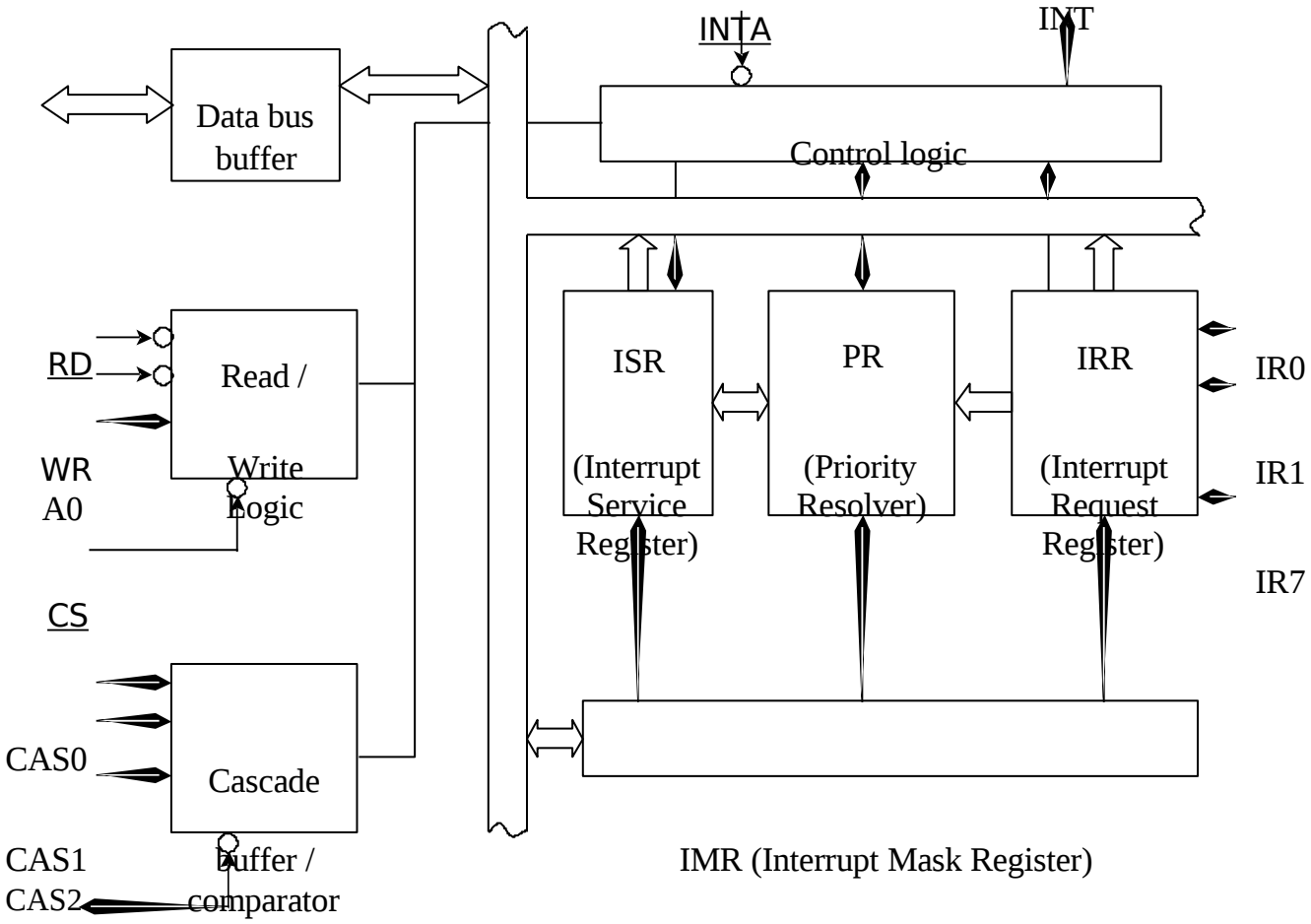
ISR (thanh ghi phục vụ ngắt): lưu trữ các yêu cầu ngắt đang phục vụ

IMR (thanh ghi mặt nạ ngắt): lưu trữ mặt nạ của các yêu cầu ngắt tại ngõ vào

Control logic (logic điều khiển): gởi yêu cầu ngắt tới chân INTR của CPU khi có tín hiệu ngắt tại ngõ vào của 8259A và nhận trả lời chấp nhận yêu cầu ngắt hay không INTA từ CPU để đưa kiểu ngắt vào CPU.

Data bus buffer (đệm bus dữ liệu): giao tiếp giữa 8259A với bus dữ liệu của CPU.

Cascade buffer / comparator (đệm nối tầng và so sánh): lưu trữ và so sánh số hiệu của các kiểu ngắt trong trường hợp dùng nhiều mạch 8259A.



Hình 1.12 – Sơ đồ khối của PIC 8259A

- **Các tín hiệu điều khiển:**

CAS0 ÷ 2 (In, Out): các ngõ vào chọn mạch 8259A tớ (slave) từ mạch 8259A chủ (master) trong trường hợp dùng nhiều mạch 8259A để tăng yêu cầu ngắt.

INTERNAL BUS

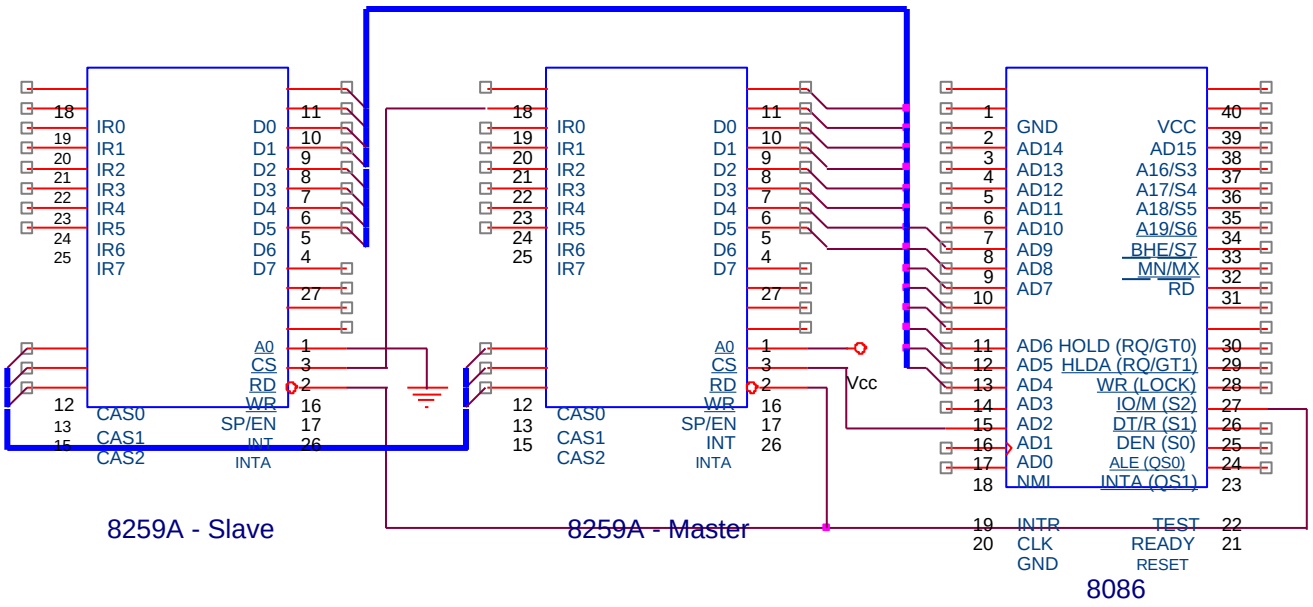
SP / EN (In, Out) (Slave Program / Enable Buffer): nếu 8259A hoạt động ở chế độ không dùng đệm dữ liệu thì tín hiệu này dùng để xác định mạch 8259A là mạch chủ (SP = 1) hay tớ (SP = 0). Nếu 8259A hoạt động ở chế độ có đệm dữ liệu thì tín hiệu này dùng để cho phép giao tiếp giữa 8259A và CPU, khi đó mạch 8259A là master hay slave phải dựa vào từ lệnh khởi động ICW4.

INT (Out): tín hiệu yêu cầu ngắt đưa đến CPU (chân INTR).

INTA (In): nhận trả lời chấp nhận ngắt hay không từ CPU (chân INTA)

A0: cho phép chọn các từ điều khiển của 8259A.

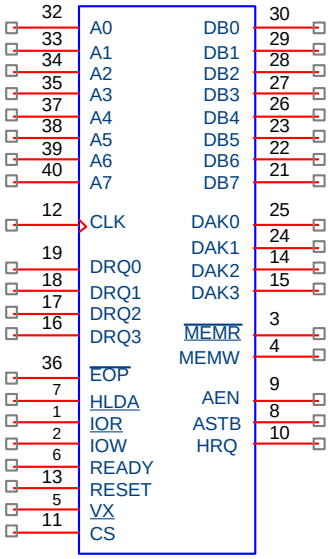
8259A cho phép xử lý 8 ngắt với 8 mức ưu tiên khác nhau. Trong trường hợp hệ thống có số lượng ngắt lớn hơn thì có thể mắc nhiều 8259A liên tầng.



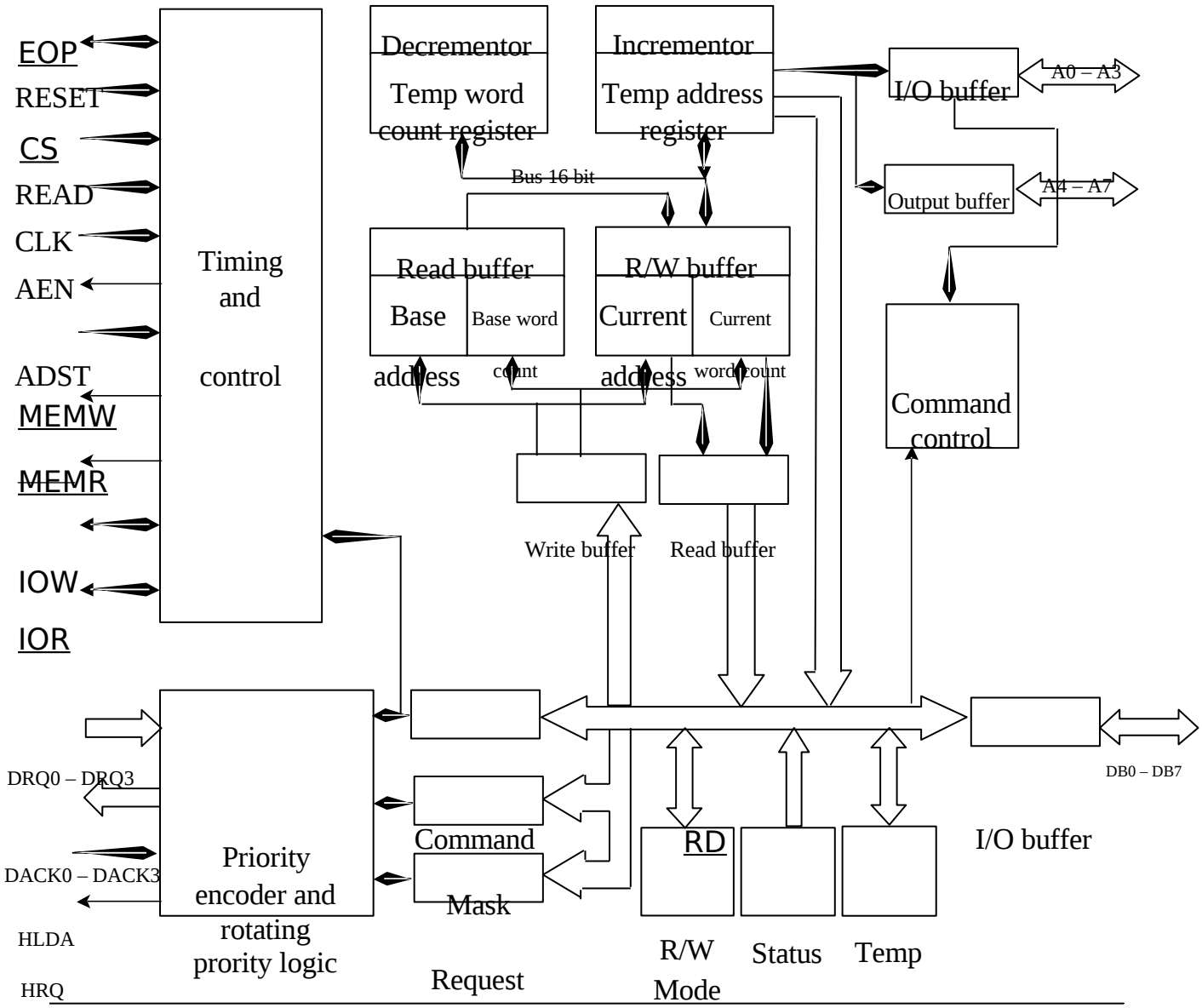
Hình 1.13 – 8259A mắc liên tầng

3.3.5. Chip điều khiển truy nhập bộ nhớ trực tiếp DMAC 8237 (Direct Memory Access Controller)

DMAC 8237 có thể thực hiện truyền dữ liệu theo 3 kiểu: kiểu đọc (từ bộ nhớ ra thiết bị ngoại vi), kiểu ghi (từ thiết bị ngoại vi đến bộ nhớ) và kiểu kiểm tra.


32 A0
33 A1
34 A2
35 A3
37 A4
38 A5
39 A6
40 A7
12 CLK
19 DRQ0
18 DRQ1
17 DRQ2
16 DRQ3
36 EOP
7 HLDA
1 IOR
2 IOW
6 READY
13 RESET
5 VX
11 CS
30 DB0
29 DB1
28 DB2
27 DB3
26 DB4
23 DB5
22 DB6
21 DB7
25 DAK0
24 DAK1
14 DAK2
15 DAK3
3 MEMR
4 MEMW
9 AEN
8 ASTB
10 HRQ

8237


EOP
RESET
CS
READ
CLK
AEN
ADST
MEMW
MEMR
IOW
IOR
Timing and control
Decrementor
Temp word count register
Incrementor
Temp address register
I/O buffer
A0 – A3
Output buffer
A4 – A7
Bus 16 bit
Read buffer
Base address
Base word count
R/W buffer
Current address
Current word count
Command control
Write buffer
Read buffer
DRQ0 – DRQ3
DACK0 – DACK3
HLDA
HRQ
Priority encoder and rotating prority logic
Command
Mask
Request
RD
R/W
Mode
Status
Temp
I/O buffer
DB0 – DB7

– Sơ đồ chân và sơ đồ khối của DMAC 8237A

A8 – A15

D0 – D1

- **Khối Timing and Control (định thời và điều khiển):**

Tạo các tín hiệu định thời và điều khiển cho bus ngoài (external bus). Các tín hiệu này được đồng bộ với xung clock đưa vào DMAC (tần số xung clock tối đa là 5 MHz).

- **Khối Priority encoder and rotating priority logic (mã hóa ưu tiên và quay mức ưu tiên):**

DMAC 8237A có 2 mô hình ưu tiên: mô hình ***ưu tiên cố định*** (***fixed priority***) và mô hình ***ưu tiên quay*** (***rotating priority***). Trong mô hình ưu tiên cố định, kênh 0 sẽ có mức ưu tiên cao nhất còn kênh 3 có mức ưu tiên thấp nhất. Còn đối với mô hình ưu tiên quay thì mức ưu tiên khi khởi động giống như mô hình ưu tiên cố định nhưng khi yêu cầu DMA tại một kênh nào đó được phục vụ thì sẽ được đặt xuống mức ưu tiên thấp nhất.

- **Khối Command Control (điều khiển lệnh):**

Giải mã các thanh ghi lệnh (xác định thanh ghi sẽ được truy xuất và loại hoạt động cần thực hiện).

- **Các thanh ghi:**

DMAC 8237A có tất cả 12 loại thanh ghi nội khác nhau:

Tên	Kích thước (bit)	Số lượng
Thanh ghi địa chỉ cơ sở (Base Address Register)	16	4
Thanh ghi đếm từ cơ sở (Base Word Count Register)	16	4
Thanh ghi địa chỉ hiện hành (Current Address Register)	16	4
Thanh ghi đếm từ hiện hành (Current Word Count Register)	16	4
Thanh ghi địa chỉ tạm (Temporary Address Register)	16	1
Thanh ghi đếm từ tạm (Temporary Word Count Register)	16	1
Thanh ghi trạng thái (Status Register)	8	1
Thanh ghi lệnh (Command Register)	8	1
Thanh ghi tạm (Temporary Register)	8	1
Thanh ghi chế độ (Mode Register)	6	4
Thanh ghi mặt nạ (Mask Register)	4	1
Thanh ghi yêu cầu (Request Register)	4	1

- **Chức năng các chân của 8237A:**

CLK (Input): tín hiệu xung clock của mạch. Tín hiệu này thường được lấy từ 8284 sau khi qua cổng đảo.

CS (Input): thường được nối với bộ giải mã địa chỉ.

RESET (Input): khởi động 8237A, được nối với ngõ RESET của 8284. Khi Reset thì thanh ghi mặt nạ được lập còn các phần sau bị xóa:

+ Thanh ghi lệnh

+ Thanh ghi trạng thái

+ Thanh ghi yêu cầu

+ Thanh ghi tạm

+ Flip-flop đầu/cuối (First/Last flip-flop)

READY (Input): nối với READY của CPU để tạo chu kỳ đợi khi truy xuất các thiết bị ngoại vi hay bộ nhớ chậm.

HLDA (Hold Acknowledge)(Input): tín hiệu chấp nhận yêu cầu treo từ CPU

DRQ_0 – DRQ_3 (DMA Request)(Input): các tín hiệu yêu cầu treo từ thiết bị ngoại vi.

DB0 – DB7 (Input, Output): nối đến bus địa chỉ và dữ liệu của CPU

$\overline{IOR}$, ***$\overline{IOW}$*** (Input, Output): sử dụng trong các chu kỳ đọc và ghi

$\overline{EOP}$ (End Of Process)(Input,Output): bắt buộc DMAC kết thúc quá trình DMA nếu là ngõ vào hay dùng để báo cho một kênh biết là dữ liệu đã chuyển xong (Terminal count – TC), thường dùng như yêu cầu ngắt để CPU kết thúc quá trình DMA.

A0 – A3 (Input, Output): chọn các thanh ghi trong 8237A khi lập trình hay dùng để chứa 4 bit địa chỉ thấp.

A4 – A7 (Output): chứa 4 bit địa chỉ

HRQ (Hold Request)(Output): tín hiệu yêu cầu treo đến CPU

$DACK_0$ – $DACK_3$ (DMA Acknowledge)(Output): tín hiệu trả lời yêu cầu DMA cho các kênh.

AEN (Output): cho phép lấy địa chỉ vùng nhớ cần trao đổi

ADSTB (Address Strobe)(Output): chốt các bit địa chỉ cao A8 – A15 chứa trong các chân DB0 – DB7

MEMR , ***MEMW*** (Output): dùng để đọc / ghi bộ nhớ.

- **Các thanh ghi nội:**

Các thanh ghi nội trong DMAC 8237A được truy xuất nhờ các bit địa chỉ thấp A0 – A3.

Bit địa chỉ				Địa chỉ	Chọn chức năng	R/W?
A3	A2	A1	A0			
0	0	0	0	X0	Thanh ghi địa chỉ bộ nhớ kênh 0	R/W
0	0	0	1	X1	Thanh ghi đếm từ kênh 0	R/W
0	0	1	0	X2	Thanh ghi địa chỉ bộ nhớ kênh 1	R/W
0	0	1	1	X3	Thanh ghi đếm từ kênh 1	R/W

0	1	0	0	X4	Thanh ghi địa chỉ bộ nhớ kênh 2	R/W
0	1	0	1	X5	Thanh ghi đếm từ kênh 2	R/W
0	1	1	0	X6	Thanh ghi địa chỉ bộ nhớ kênh 3	R/W
0	1	1	1	X7	Thanh ghi đếm từ kênh 3	R/W
1	0	0	0	X8	Thanh ghi trạng thái / lệnh	R/W
1	0	0	1	X9	Thanh ghi yêu cầu	W
1	0	1	0	XA	Thanh ghi mặt nạ cho một kênh	W
1	0	1	1	XB	Thanh ghi chế độ	W
1	1	0	0	XC	Xóa flip-flop đầu/cuối	W
1	1	0	1	XD	Xóa toàn bộ các thanh ghi / đọc thanh ghi tạm	W/R
1	1	1	0	XE	Xóa thanh ghi mặt nạ	W
1	1	1	1	XF	Thanh ghi mặt nạ	W

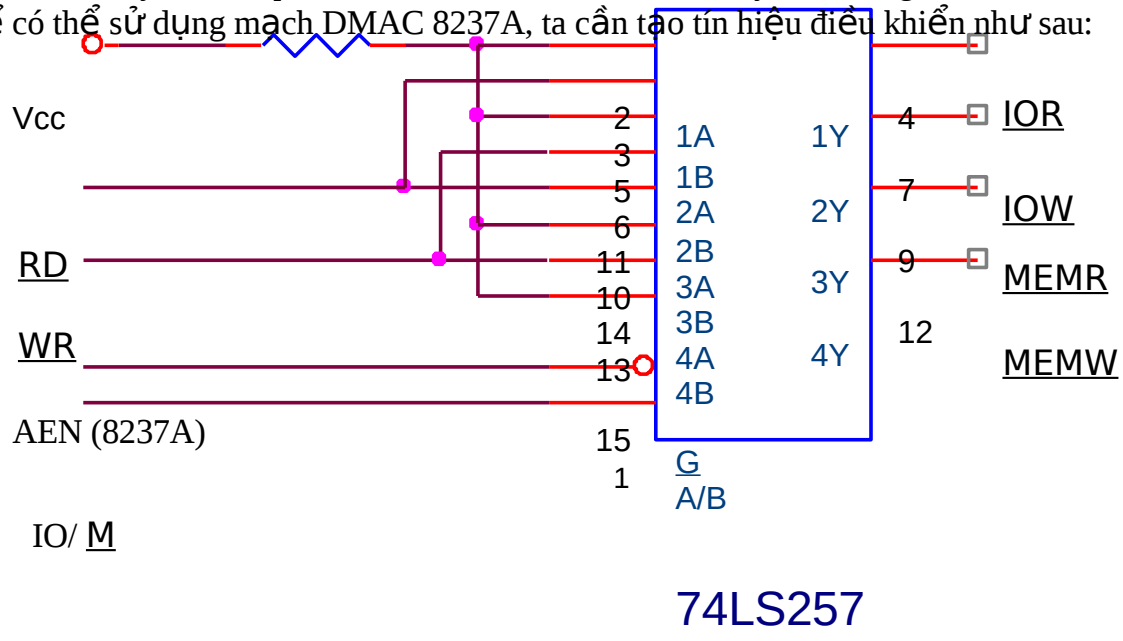
Địa chỉ các thanh ghi nội dùng ghi / đọc địa chỉ:

Kênh	IOR	IOW	**A3**	**A2**	**A1**	**A0**	**Thanh ghi**	**R/W?**
0	1	0	0	0	0	0	Địa chỉ cơ sở và địa chỉ hiện hành	W
	0	1	0	0	0	0	Địa chỉ hiện hành	R
	1	0	0	0	0	1	Bộ đếm cơ sở và bộ đếm hiện hành	W
	0	1	0	0	0	1	Bộ đếm hiện hành	R
1	1	0	0	0	1	0	Địa chỉ cơ sở và địa chỉ hiện hành	W
	0	1	0	0	1	0	Địa chỉ hiện hành	R
	1	0	0	0	1	1	Bộ đếm cơ sở và bộ đếm hiện hành	W
	0	1	0	0	1	1	Bộ đếm hiện hành	R
2	1	0	0	1	0	0	Địa chỉ cơ sở và địa chỉ hiện hành	W
	0	1	0	1	0	0	Địa chỉ hiện hành	R
	1	0	0	1	0	1	Bộ đếm cơ sở và bộ đếm hiện hành	W
	0	1	0	1	0	1	Bộ đếm hiện hành	R
3	1	0	0	1	1	0	Địa chỉ cơ sở và địa chỉ hiện hành	W
	0	1	0	1	1	0	Địa chỉ hiện hành	R
	1	0	0	1	1	1	Bộ đếm cơ sở và bộ đếm hiện hành	W
	0	1	0	1	1	1	Bộ đếm hiện hành	R

Địa chỉ các thanh ghi trạng thái và điều khiển:

IOR `	IOW	A3	A2	A1	A0		Thanh ghi
1	0	1	0	0	0		Ghi thanh ghi lệnh
0	1	1	0	0	0		Đọc thanh ghi trạng thái
1	0	1	0	0	1		Ghi thanh ghi yêu cầu
1	0	1	0	1	0		Ghi thanh ghi mặt nạ
1	0	1	0	1	1		Ghi thanh ghi chế độ
1	0	1	1	0	0		Xóa flip-flop đầu/cuối
1	0	1	1	0	1		Xóa tất cả các thanh ghi nội
0	1	1	1	0	1		
1	0	1	1	1	0		Địa chỉ cơ sở và địa chỉ hiện hành
0	1	1	1	1	0		Địa chỉ hiện hành
1	0	1	1	1	1		Bộ đếm cơ sở và bộ đếm hiện hành
0	1	1	1	1	1		Bộ đếm hiện hành

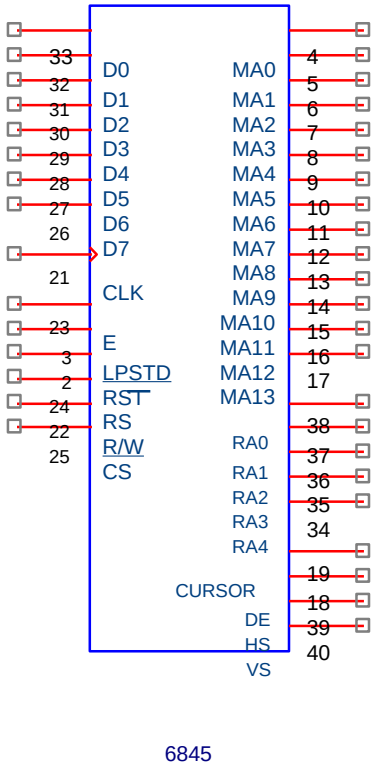
Mạch 8273A-5 chứa 4 kênh trao đổi dữ liệu DMA với mức ưu tiên lập trình được. 8237A-5 có tốc độ truyền 1 MBps cho mỗi kênh và 1 kênh có thể truyền 1 mảng có độ dài 64 KB. Để có thể sử dụng mạch DMAC 8237A, ta cần tạo tín hiệu điều khiển như sau:



Hình 1.15 – Tín hiệu điều khiển cho hệ thống làm việc với DMAC 8237A

Tín hiệu AEN từ 8237A dùng để cấm các tín hiệu điều khiển từ CPU khi DMAC đã nắm quyền điều khiển bus.

3.3.6. Chip điều khiển màn hình CRTC 6845 (Cathode Ray Tube Controller)



Hình 1.16 – Sơ đồ chân của 6845

RST (Reset): khởi động lại 6845.

LPSTD (Light Pen Strobe): lưu trữ địa chỉ hiện hành của RAM màn hình trong thanh ghi bút sáng. CPU đọc thanh ghi và xác định vị trí bút sáng trên màn hình.

MA0 ÷ ***MA13*** (Memory Address): 14 địa chỉ nhớ cho RAM màn hình.

DE (Display Enable): cho phép (=1) hay không (=0) các tín hiệu điều khiển và địa chỉ vùng hiện lên màn hình.

CURSOR: vị trí con trỏ đã quét (=1) hay chưa (=0).

VS (Vertical Synchronization): ngõ ra tín hiệu đồng bộ quét dọc

HS (Horizontal Synchronization): ngõ ra tín hiệu đồng bộ quét ngang

RA0 ÷***RA4*** (Row Address): phân định hàng quét của ký tự trong chế độ văn bản (32 hàng quét). Trong chế độ đồ họa, chúng kết hợp với MA0 ÷ MA13 tạo các địa chỉ cho các bank RAM màn hình.

$\overline{\textbf{\textit{D0}}}$ ÷ ***D7***: đường dữ liệu.

CS : chọn chip.

RS (Regigter Select): chọn thanh ghi địa chỉ (=0) hay thanh ghi dữ liệu (=1).

E: xung âm kích hoạt bus dữ liệu và dùng như xung clock cho 6845 đọc / ghi dữ liệu vào các thanh ghi bên trong.

R/W : đọc / ghi dữ liệu vào các thanh ghi.

CLK: dùng đồng bộ với tín hiệu của màn hình và thường bằng tốc độ hiện ký tự trên màn hình.

3.3.7. Chip đồng xử lý toán học 8087/80287/80387 (Mathematical co-processor)

Các bộ đồng xử lý toán 80x87 hỗ trợ CPU trong việc tính toán các biểu thức dùng dấu chấm động như cộng, trừ, nhân, chia các số dấu chấm động, căn thức, logarit, … Chúng cho phép xử lý các phép toán này nhanh hơn nhiều so với CPU. Thời gian xử lý giữa 8087 và 8086 như sau (dùng xung clock 8 MHz):

Phép toán	**8087 [μs]**	**8086 [μs]**
Cộng / trừ	10.6	1000
Nhân	11.9	1000
Chia	24.4	2000
Căn bậc hai	22.5	12250
Tang	56.3	8125
Lũy thừa	62.5	10680
Lưu trữ	13.1	750

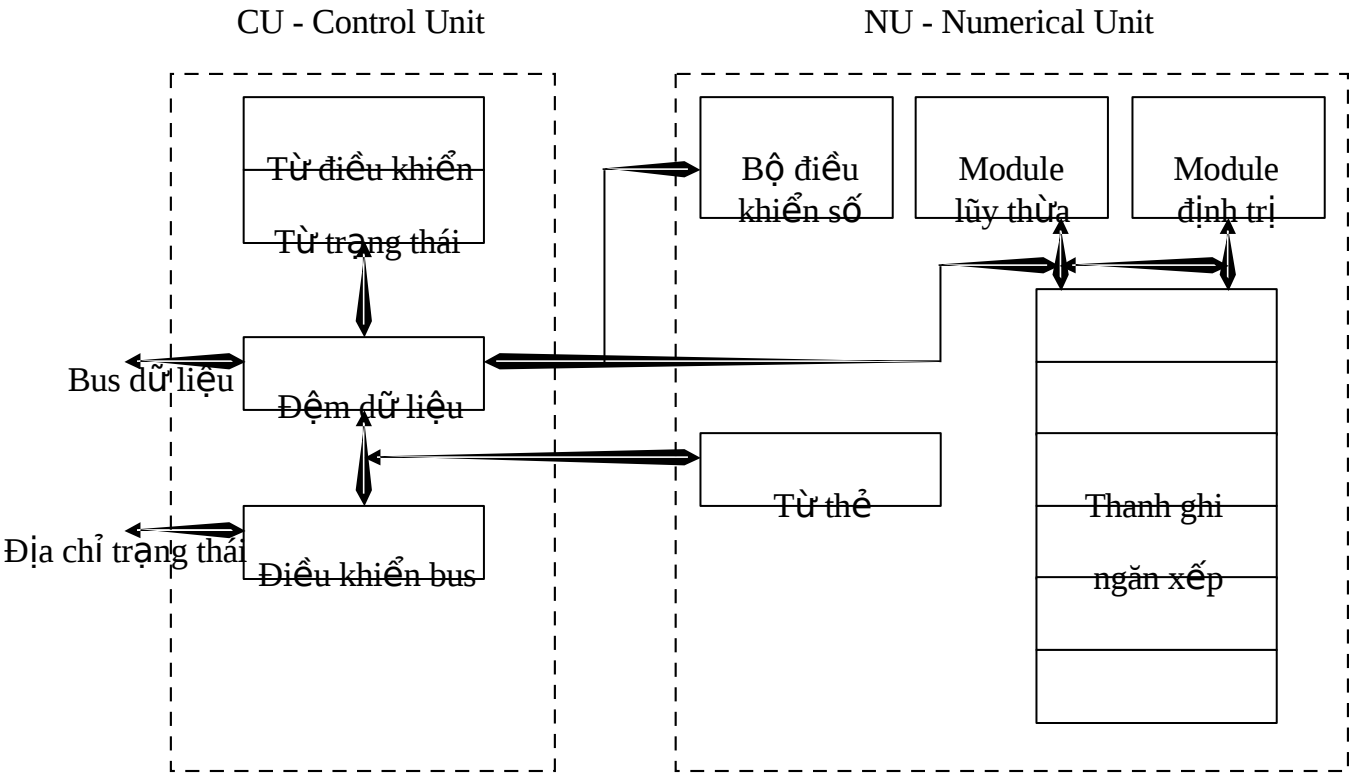
- **8087:**

8087 gồm một đơn vị điều khiển (CU – Control Unit) dùng để điều khiển bus và một đơn vị số học (NU – Numerical Unit) để thực hiện các phép toán dấu chấm động trong các mạch tính lũy thừa (exponent module) và mạch tính phần định trị (mantissa module). Khác với 8086, thay vì dùng các thanh ghi rời rạc là một ngăn xếp thanh ghi.

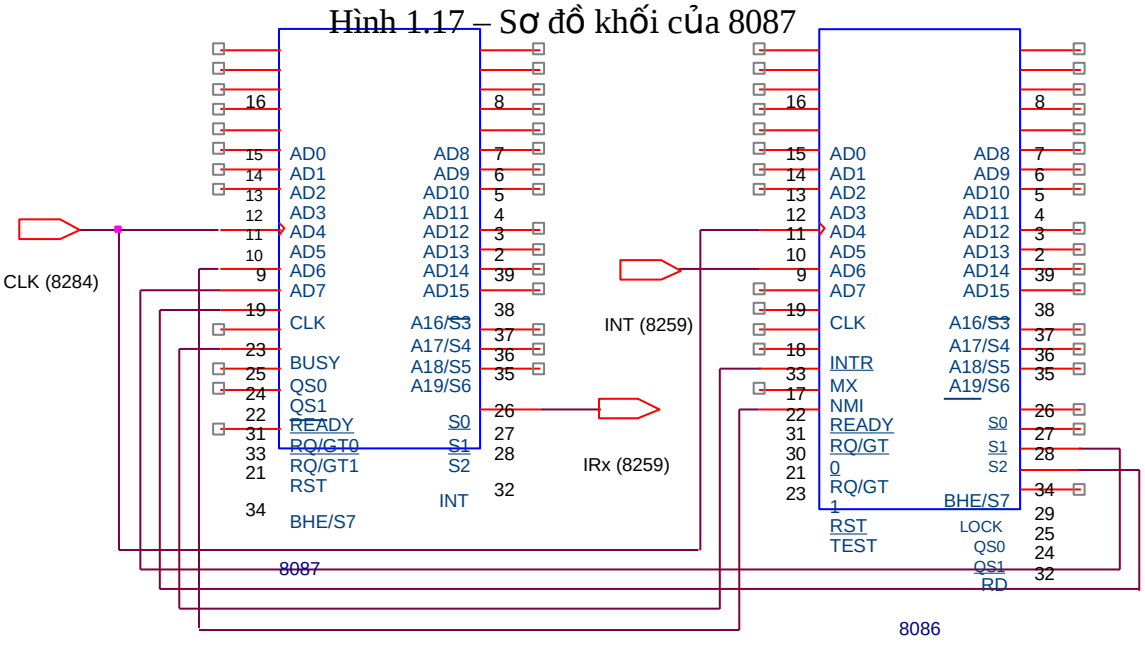
Đơn vị điều khiển nhận và giải mã lệnh, dọc và ghi các toán hạng, chạy các lệnh điều khiển riêng của 8087. Do đó, CU có thể đồng bộ với CPU trong khi NU đang thực hiện các công việc tính toán. CU bao gồm bộ điều khiển bus, bộ đệm dữ liệu và hàng lệnh.

Ngăn xếp thanh ghi có tất cả 8 thanh ghi từ R0 ÷ R7, mỗi thanh ghi dài 80 bit trong đó bit 79 là bit dấu, bit 64 ÷ 78 dùng cho số mũ và phần còn lại là phần định trị. Dữ liệu truyền giữa các thanh ghi này được thực hiện rất nhanh do 8087 có độ rộng bus dữ liệu là 84 bit và không cần phải biến đổi định dạng.

Ngay sau khi reset PC, bộ đồng xử lý kiểm tra xem nó có được nối với PC hay không bằng các đường BHE /S7. 8087 sẽ điều chỉnh độ dài của hàng lệnh cho phù hợp với CPU (nếu dùng 8086 thì độ dài là 6 byte).



Hình 1.17 – Sơ đồ khối của 8087



Hình 1.18 – Sơ đồ kết nối 8087

và CPU 8086

8087 có một thanh ghi trạng thái là thanh ghi từ thẻ (tag word) gồm các cặp bit Tag0 ÷ Tag7 để lưu trữ các thông tin liên quan đến nội dung của các thanh ghi R0 ÷ R7 để cho phép thực hiện một số tác vụ nhanh hơn. Mỗi thanh ghi từ thẻ có 2 bit xác định 4 giá trị khác nhau của các thanh ghi Ri.

Tag = 00: xác định

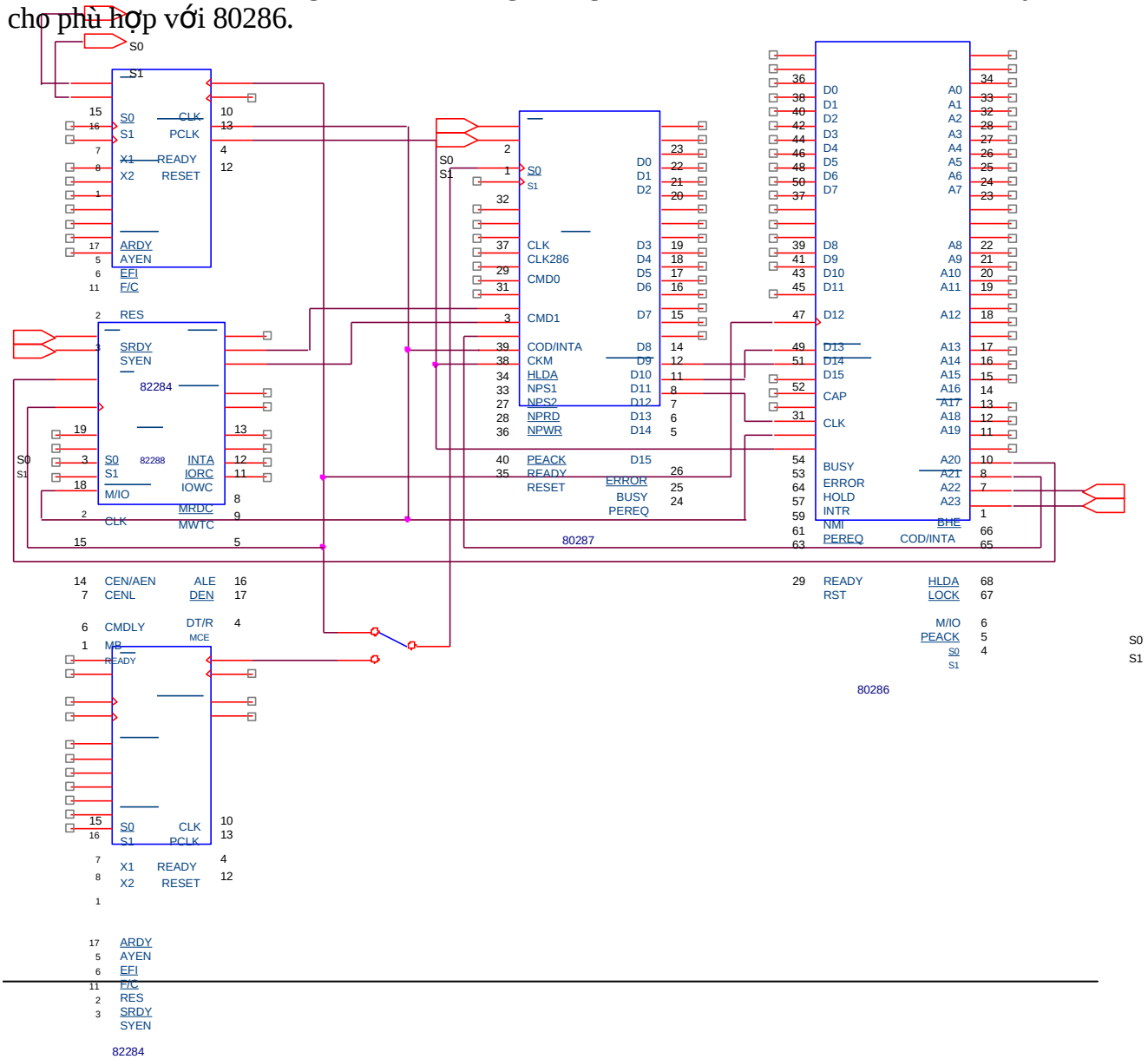
Tag = 01: zero

Tag = 10: NAN, giá trị bất thường

Tag = 11: rỗng

- **80287:**

Do 80286 có chế độ mạch bảo vệ nên mạch ghép nối giữa 80286 và 80287 được thiết kế khác 8087 ở đơn vị điều khiển CU. Bộ đồng xử lý ở đây không thực hiện truy xuất bộ nhớ trực tiếp. Để truy xuất được bộ nhớ, 80287 không những cần một đơc vị định địa chỉ đơn giản của nó mà còn phải được tăng cường thêm chức năng quản lý bộ nhớ của 80286. Cấu trúc bên trong của 80287 cũng tương tự như 8087, chỉ có đơn vị bus thay đổi cho phù hợp với 80286.



Hình 1.19 – Sơ đồ kết nối giữa 80286 và 80287

Khác với 8087, 80287 hoạt động không đồng bộ với CPU nên có thể dùng xung clock riêng.

- **80387:**

Ưu điểm của 80387 so với 80287 là có thể thực hiện các phép toán số học nhanh hơn. No có bus dữ liệu 32 bit như CPU và sử dụng công nghệ CMOS nên công suất tiêu thụ thấp hơn.

4. Các thế hệ máy tính

4.1. Máy tính cơ khí

Năm 1942, nhà khoa học Pháp Blaise Pascal xây dựng một máy đầu tiên thực hiện công việc tính toán. Đây là thiết bị hoàn toàn bằng cơ khí sử dụng các bánh răng và cung cấp lực bằng một cánh tay quay. Nó chỉ thực hiện được các phép toán cộng và trừ. 30 năm sau, nhà toán học Đức Baron Gottfried Wilherm von Leibniz xây dựng một máy cơ khí làm được phép nhân và chia.

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. Năm 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.

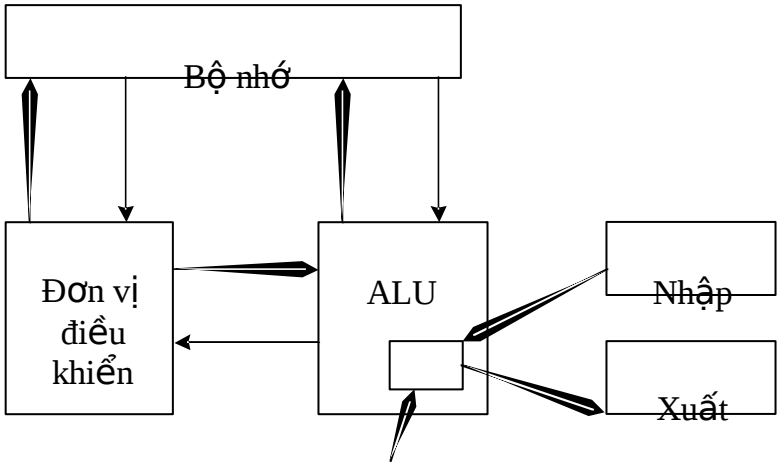
Năm 1944, Aiken hoàn tất máy tính Mark 1, có tất cả 72 từ, mỗi từ 23 số thập phân và có thời gian một chu kỳ là 6 giây. Việc nhập và xuất thực hiện bằng các băng giấy đục lỗ.

4.2. 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. Cũng trong năm 1943, Mauchley và Presper Eckert bắt đầu tiến hành xây dựng máy tính ENIAC (Electronic Numerical Integrator And Computer). ENIAC gồm 1800 đèn điện 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 cơ 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.



Thanh ghi tích lũy

Hình 1.20 – Máy von Neumann

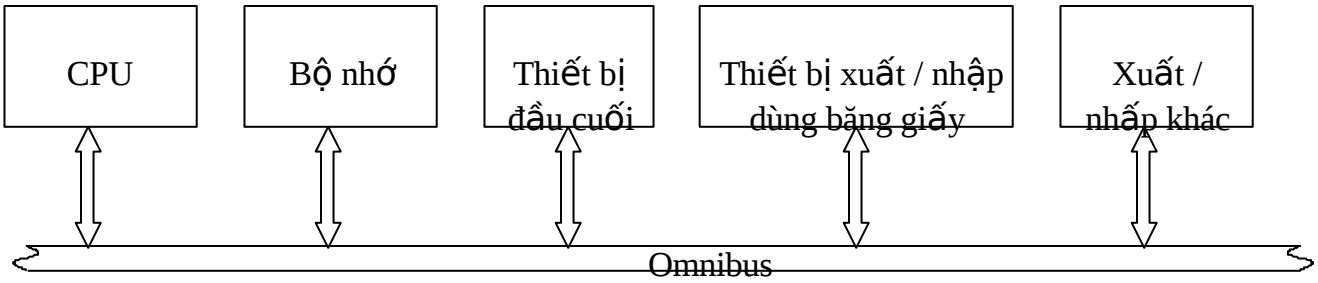
Vào cùng thời gian của máy IAS, các nhà nghiên cứu ở MIT cũng đang xây dựng một máy tính, máy Whirlwind 1. Nó có từ dài 16 bit và thiết kế để điều khiển thời gian thực.

4.3. 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 được 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 đời 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 đó, 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. Năm 1964, công ty CDC giới thiệu máy 6600 có tốc độ nhanh hơn 7094 do bên trong CPU có một cơ 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 năng thực thi 10 lệnh đồng thời.



Hình 1.21 – Omnibus của PDP-8

4.4. 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, được 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 2^{24} byte nhớ (16 MB).

4.5. 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, siuê 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 được thiết kế đặc biệt để cựa 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.

Chương 2

TỔ CHỨC CPU (8086/8088/80286)

1. Định thời chu kỳ bus

Mỗi chu kỳ bus bắt đầu bằng việc xuất địa chỉ bộ nhớ hoặc I/O port (chu kỳ xung nhịp T1). Với 8086 thì địa chỉ này có thể là địa chỉ bộ nhớ 20 bit, địa chỉ I/O gián tiếp 16 bit (thanh ghi DX) hay địa chỉ I/O trực tiếp 8 bit. Bus điều khiển có 4 tín hiệu tác động mức thấp là MEMR , MEMW , IOR và IOW .

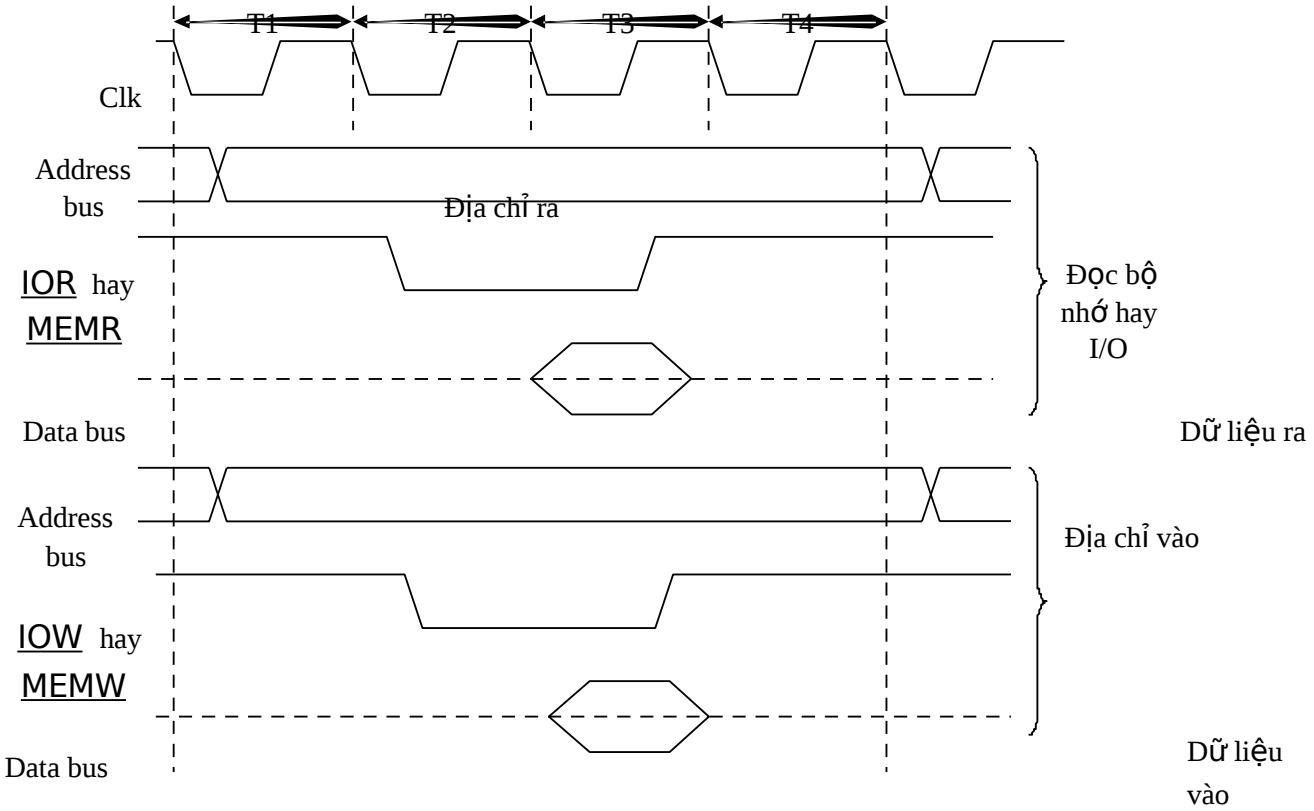
Các chuỗi sự kiện xảy ra trong một chu kỳ bus đọc bộ nhớ:

T1: CPU xuất địa chỉ bộ nhớ. Các đường dữ liệu không hoạt động và các đường điều khiển bị cấm

T2: Đường điều khiển MEMR xuống mức thấp. Đơn vị bộ nhớ ghi nhận chu kỳ bus này là quá trình đọc bộ nhớ và đặt byte hay word có địa chỉ đó lên bus dữ liệu.

T3: CPU đặt cấu hình để các đường bus dữ liệu là nhập. Trạng thái này chủ yếu để bộ nhớ có thời gian tìm kiếm byte hay word dữ liệu

T4: CPU đợi dữ liệu trên bus dữ liệu. Do đó, nó thực hiện chốt bus dữ liệu và giải phóng các đường điều khiển đọc bộ nhớ. Quá trình này sẽ kết thúc chu kỳ bus.



Hình 2.1 – Định thì chu kỳ bus

Ghi bộ
nhớ hay
I/O

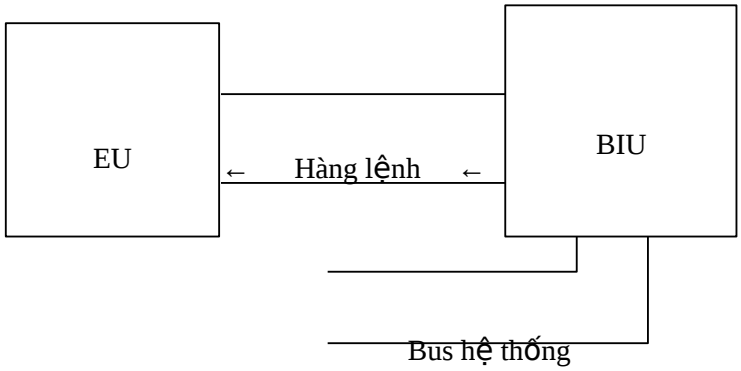
Trong một chu kỳ bus, CPU có thể thực hiện đọc I/O, ghi I/O, đọc bộ nhớ hay ghi bộ nhớ. Các đường bus địa chỉ và bus điều khiển dùng để xác định địa chỉ bộ nhớ hay I/O và hướng truyền dữ liệu trên bus dữ liệu.

Chú ý rằng CPU điều khiển tất cả các quá trình trên nên bộ nhớ bắt buộc phải cung cấp được dữ liệu vào lúc MEMR lên mức cao trong trạng thái T4. Nếu không, CPU sẽ đọc dữ liệu ngẫu nhiên không mong muốn trên bus dữ liệu. Để giải quyết vấn đề này, ta có thể dùng thêm các trạng thái chờ (wait state).

2. Kiến trúc nội

2.1. Kiến trúc nội

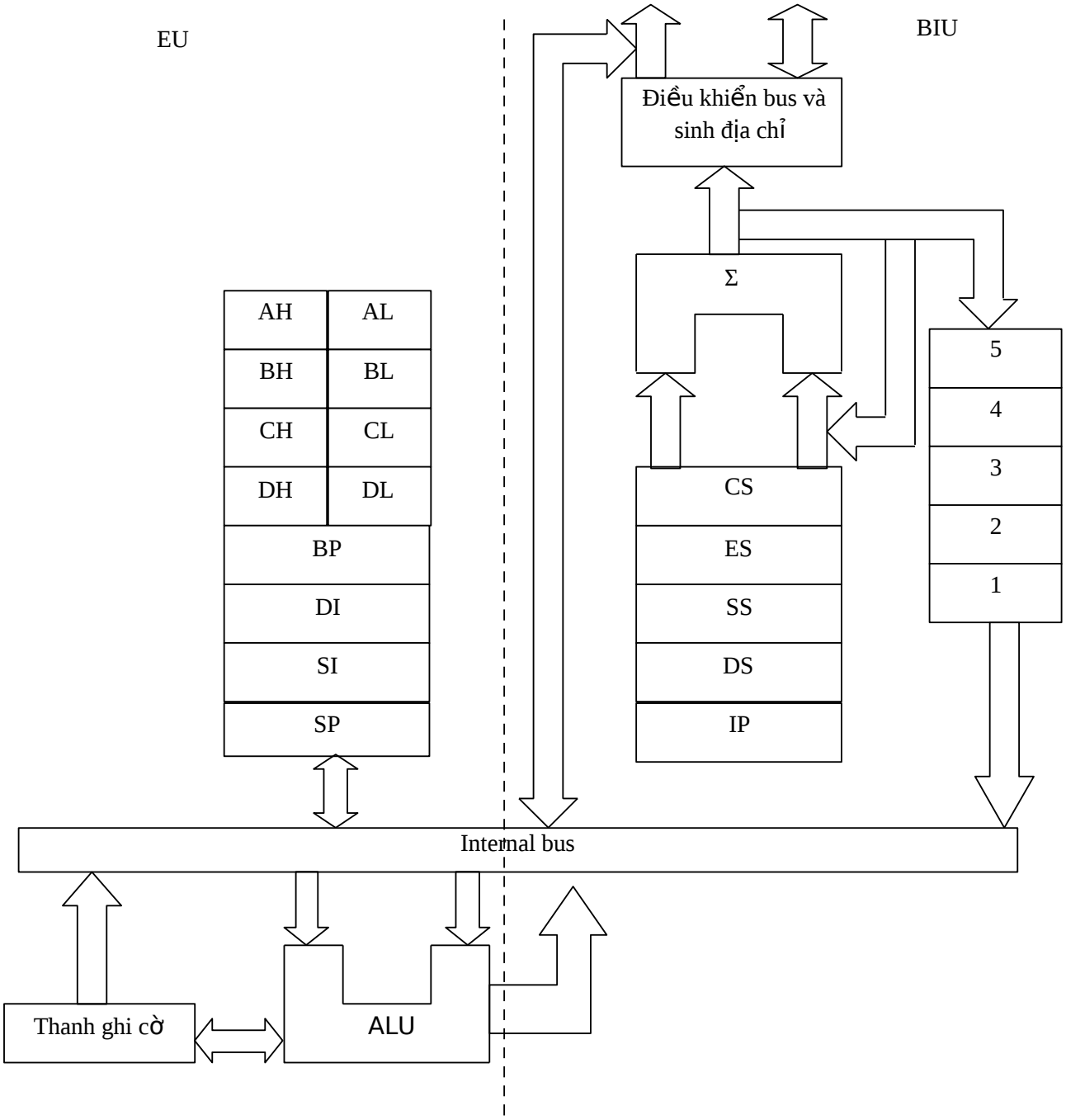
CPU có khả năng thực hiện các tác vụ dữ liệu theo tập lệnh bên trong. Một lệnh được ghi nhận bằng mã đã được định nghĩa trước, gọi là mã lệnh (opcode). Trước khi thực thi một lệnh, CPU phải nhận được mã lệnh từ bộ nhớ chương trình của nó. Quá trình xử lý này gọi là chu kỳ nhận lệnh (fetch cycle). Một khi các mã được nhận và được giải mã thì mạch bên trong CPU có thể tiến hành thực thi (execute) mã lệnh.



Hình 2.2 – Kiến trúc tổng quát của CPU 8086

BIU (Bus Interface Unit – đơn vị giao tiếp bus) nhận các mã lệnh từ bộ nhớ và đặt chúng vào hàng chờ lệnh. EU (Execute Unit – đơn vị thực thi) sẽ giải mã và thực hiện các lệnh trong hàng. Chú ý rằng các đơn vị EU và BIU làm việc độc lập với nhau nên BIU có khả năng đang nhận một lệnh mới trong khi EU dang thực thi lệnh trước đó. Khi EU đã thực hiện xong lệnh, nó sẽ lấy mã lệnh kế tiếp trong hàng lệnh (instruction queue).

Kiến trúc nội của CPU 8086 ở hình 2.3. Nó có 2 bộ xử lý riêng: BIU và EU. BIU cung cấp các chức năng phần cứng, bao gồm tạo các địa chỉ bộ nhớ và I/O để chuyển dữ liệu giữa EU và bên ngoài CPU. EU nhận các mã lệnh chương trình và dữ liệu từ BIU, thực thi các lệnh này và chứa các kết quả trong các thanh ghi. Ngoài ra, dữ liệu cũng có thể chứa trong một vị trí bộ nhớ hay được ghi vào thiết bị xuất. Chú ý rằng EU không có bus hệ thống nên phải thực hiện nhận và xuất tất cả các dữ liệu của nó thông qua BIU. Sự khác biệt giữa CPU 8086 và 8088 là BIU. Trong 8088, đường bus dữ liệu là 8 bit trong khi của 8086 là 16 bit. Ngoài ra hàng lệnh của 8088 dài 4 byte trong khi của 8086 là 6 byte. Tuy nhiên do EU giữa hai loại µP này giống nhau nên ***các chương trình viết cho 8086 có thể chạy được trên 8088 mà không cần thay đổi gì cả***.



Hình 2.3 – Kiến trúc nội của 8086

2.2. Cơ chế đường ống (pipeline)

- **Quá trình nhận lệnh và thực thi lệnh:**

1/ BIU xuất nội dung của thanh ghi con trỏ lệnh IP (Instruction Pointer) ra bus địa chỉ để chọn byte hay word đọc vào BIU.

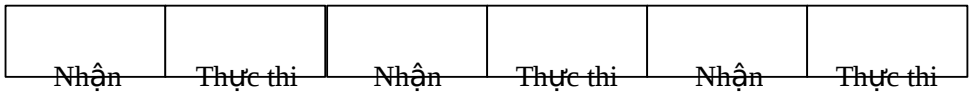
2/ Thanh ghi IP được tăng lên để chuẩn bị nhận lệnh kế (số byte tăng lên của IP tùy thuộc vào kích thước lệnh trước đó).

3/ Khi lệnh ở trong BIU, nó được đưa sang hàng lệnh (queue). Đây là một thanh ghi lưu trữ dạng FIFO (First In First Out – Vào trước ra trước), dùng cơ chế xử lý xen kẽ liên tục các dòng mã lệnh (kỹ thuật đường ống – pipelining).

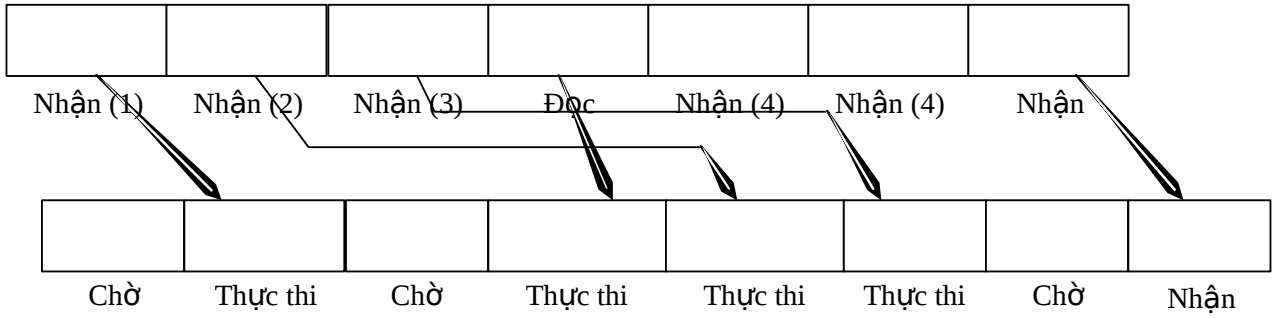
4/ Giả sử ban đầu hàng lệnh trống, EU sẽ không làm gì cả cho đến khi bắt đầu xuất hiện một lệnh trong hàng, EU sẽ lấy lệnh ra khỏi hàng và bắt đầu thực thi lệnh đó.

5/ Trong khi EU đang thực thi lệnh, BIU tiến hành nhận lệnh mới. Tuỳ theo thời gian thực thi lệnh mà BIU có thể đưa vào hàng lệnh nhiều lệnh mới trước khi EU thực hiện lệnh xong và tiếp tục lấy lệnh mới.

BIU được lập trình để có thể nhận một lệnh mới bất kỳ lúc nào hàng lệnh có chỗ cho 1 byte (8088) hay 2 byte (8086). Lợi ích của phương pháp xử lý theo cơ chế pipeline là EU có thể thực thi các lệnh gần như liên tục thay vì phải đợi BIU nhận thêm lệnh mới.



(a)



(b)

(1): lệnh thực thi không cần dữ liệu trong hàng
(2): lệnh thực thi cần dữ liệu trong hàng
(3): lệnh nhảy
(4): các lệnh bị bỏ qua do lệnh nhảy

Hình 2.4

(a) CPU thông thường dùng chu kỳ nhận và thực thi lệnh tuần tự
(b) Kiến trúc dạng pipeline của 8086/8088 cho phép thực thi các lệnh mà không bị trễ do quá trình nhận lệnh

Có 3 điều kiện làm cho EU ở chế độ chờ:

- Điều kiện thứ nhất xảy ra khi lệnh cần truy xuất đến một vị trí bộ nhớ không ở trong hàng. BIU phải treo quá trình nhận lệnh và xuất ra địa chỉ của ô nhớ này. Sau khi truy xuất bộ nhớ, EU có thể tiếp tục quá trình thực thi lệnh từ hàng lệnh và BIU có thể tiếp tục đưa các lệnh vào hàng.
- Điều kiện thứ hai xảy ra khi lệnh được thực thi là lệnh nhảy (jump). Trong trường hợp này, thay vì dùng địa chỉ lệnh kế tiếp, ta phải chuyển đến địa chỉ mới (không tuần tự). Tuy nhiên, BIU vẫn luôn đặt các lệnh theo tuần tự và do đó sẽ lưu các lệnh không sử dụng. Trong khi nhận lệnh kế tiếp tại địa chỉ do lệnh jump chỉ đến, EU phải đợi và tất cả các byte trong hàng phải bỏ.

- Điều kiện thứ ba có thể làm BIU treo quá trình nhận lệnh đó là khi thực thi các lệnh có thời gian thực thi lớn. Giả sử như lệnh AAM (ASCII Adjust for Multiplication) cần 83 chu kỳ xung nhịp để hoàn tất trong khi đó với 4 chu kỳ xung nhịp cho quá trình nhận lệnh thì hàng sẽ bị đầy. Như vậy BIU phải đợi cho đến khi lệnh được thực hiện xong và EU nhận mã lệnh từ hàng thì mới có thể tiếp tục quá trình nhận lệnh.

2.3. Cơ chế siêu phân luồng (hyper-threading)

Internet, thương mại điện tử và phần mềm ứng dụng doanh nghiệp đang ngày càng đòi hỏi nhiều năng lực tính toán của các máy chủ hơn. Để nâng cao tốc độ, phần mềm cần phải được phân luồng - các chỉ thị sẽ được chia thành nhiều dòng lệnh để có thể xử lý đồng thời trên nhiều bộ xử lý. Intel đã đưa ra kỹ thuật phân luồng cho phép nâng cao tốc độ và khả năng tính toán song song cho những ứng dụng đa luồng. Công nghệ mới của Intel mô phỏng mỗi bộ vi xử lý vật lý như là hai bộ vi xử lý luận lý (logic), tài nguyên vật lý được chia sẻ và có cấu trúc chung giống hệt nhau cho cả hai bộ xử lý logic. Hệ điều hành và phần mềm ứng dụng sẽ xem như như đang chạy trên hai hay nhiều bộ xử lý, kết quả là tốc độ xử lý trung bình có thể tăng lên xấp xỉ 40% đối với một bộ xử lý vật lý, Intel gọi kỹ thuật này là siêu phân luồng.

Kỹ thuật siêu phân luồng cho phép các phần mềm ứng dụng được viết cho những máy chủ đa luồng có thể thực hiện các chỉ thị song song đồng thời trên mỗi bộ xử lý riêng, bằng cách này sẽ cải thiện tức thì tốc độ giao dịch cũng như thời gian đáp ứng và các yêu cầu đặc thù khác của phần mềm nghiệp vụ và thương mại điện tử. Kỹ thuật này tương thích với các phần mềm ứng dụng và hệ điều hành sẵn có trên các máy chủ (server), nó cho phép hỗ trợ nhiều người dùng hơn và tăng khối lượng công việc được xử lý trên một máy chủ. Với các máy trạm (workstation) cao cấp, kỹ thuật siêu phân luồng cũng sẽ tăng đáng kể tốc độ các phần mềm ứng dụng đòi hỏi năng lực tính toán cao, ví dụ như phần mềm thiết kế 3 chiều, xử lý ảnh hay video… Trong thời gian tới sẽ xuất hiện ngày càng nhiều phần mềm được thiết kế đặc biệt và tối ưu hoá cho Kỹ thuật này.

Từ tháng 01/2002, kỹ thuật siêu phân luồng đã được Intel đưa vào các bộ vi xử lý Xeon đời mới, khởi đầu với các bộ xử lý có tốc độ 1.8GHz và 2.0GHz với 512KB cache thứ cấp, sản xuất bằng công nghệ 0.13 micron (Xeon 1.7GHz, 1.8GHz, 2.0GHz với 256KB cache thứ cấp được sản xuất bằng công nghệ 0.18 không hỗ trợ siêu phân luồng). Tại thời điểm đầu tiên khi Intel giới thiệu bộ xử lý Xeon cùng với chipset 860, chỉ có một số rất ít các nhà sản xuất hàng đầu như IBM, Compaq, Dell, SuperMicro, Tyan… hỗ trợ bộ vi xử lý này, số lượng sản phẩm cũng rất ít. Tuy nhiên, khi có thêm các chipset hỗ trợ bộ xử lý Xeon như E7500 và Serverworks GC, nhiều nhà sản xuất khác đã có sản phẩm hỗ trợ bộ xử lý Xeon. Tuy nhiên đối với đa số người dùng, nhất là người dùng máy tính để bàn (desktop) thì kỹ thuật siêu phân luồng còn khá xa lạ. Intel chỉ chuẩn bị đưa ra bộ xử lý Pentium IV dành cho desktop áp dụng kỹ thuật siêu luồng (tốc độ khởi điểm là 3.06GHz).

Kỹ thuật siêu phân luồng (hyper-threading) cho phép các ứng dụng đa luồng thực hiện các luồng song song. Trong các kỹ thuật trước, sự phân luồng thực hiện bằng cách cắt các lệnh thành nhiều dòng (stream) khác nhau, mỗi dòng sẽ do một vi xử lý thực hiện (trong hệ thống đa xử lý). Với kỹ thuật siêu phân luồng, sự phân luồng sử dụng các tài nguyên của vi xử lý hiệu quả hơn do quá trình song song là tốt hơn.

Kỹ thuật siêu phân luồng cung cấp trạng thái song song ở cấp độ luồng (TLP – thread level parallelism) cho mỗi vi xử lý, kết quả là gia tăng khả năng tận dụng tài nguyên của vi xử lý. Siêu phân luồng là một dạng của kỹ thuật đa luồng song song (SMT – Simultaneous Multi Threading) trong đó nhiều luồng có thể được thực thi tại cùng một thời điểm trên một vi xử lý. Vấn đề này thực hiện bằng cách kết hợp 2 AS (Architectural State) trong mỗi vi xử lý, các AS sẽ dùng chung tài nguyên của vi xử lý. Kỹ thuật này làm đáp ứng thời gian của vi xử lý sẽ nhanh hơn trong môi trường đa nhiệm và cho phép thực hiện nhanh các hoạt động đa luồng và đa nhiệm bằng cách sử dụng các tài nguyên nhàn rỗi.

- **Kỹ thuật siêu phân luồng và đa luồng song song (SMT - Simultaneous Multi-Threading)**

Intel phát triển SMT từ một công nghệ gốc có tên mã là Jackson với cái tên khác là Hyper-Threading – kỹ thuật siêu phân luồng. Trước khi có thể hiểu về cách thức hoạt động của kỹ thuật này, chúng ta cần phải tìm hiểu cơ bản về nó, đặc biệt là về chuỗi lệnh và cách chúng hoạt động.

Cái gì làm cho một ứng dụng có thể chạy? Làm thế nào CPU biết các chỉ dẫn để thực hiện và thực hiện với dữ liệu nào? Tất cả những thông tin này có chứa trong mã biên dịch của ứng dụng đang chạy mỗi khi nạp ứng dụng đó vào. Ứng dụng lần lượt gửi các chuỗi lệnh báo cho CPU biết phải làm gì để đáp ứng, và đối với CPU chuỗi lệnh sẽ là một tập các chỉ thị cần phải thực thi. CPU biết chính xác các chỉ thị này nằm ở đâu nhờ thanh ghi bộ đếm chương trình (PC – Program Counter). PC luôn chỉ đến vị trí trong bộ nhớ nơi mà các chỉ thị cần thực hiện tiếp theo đã được lưu giữ, như vậy một khi chuỗi lệnh được gửi đến CPU thì địa chỉ trong bộ nhớ của chuỗi lệnh này đã được nạp sẵn vào PC, vì vậy CPU biết bắt đầu thực hiện từ đâu. Sau mỗi chỉ thị, PC sẽ tăng lên và quá trình tiếp tục đến hết chuỗi lệnh. Khi chuỗi lệnh được thực hiện xong, PC sẽ bị ghi đè bởi chỉ thị tiếp theo. Chuỗi lệnh có thể bị ngắt bởi một yêu cầu khác, khi đó CPU sẽ lưu giá trị hiện tại của PC trong ngăn xếp (stack) và nạp giá trị mới vào PC, tuy nhiên hạn chế là tại mỗi thời điểm chỉ có thể có duy nhất một chuỗi lệnh được thực thi. Một hướng giải quyết chung cho vấn đề này là sử dụng hai hay nhiều CPU, nếu tại mỗi thời điểm một CPU chỉ có thể thực thi một chuỗi lệnh thì hai hay nhiều CPU sẽ thực thi được hai hay nhiều chuỗi lệnh. Tuy vậy, lại có nhiều vấn đề nảy sinh với cách giải quyết này, trước hết là nhiều CPU sẽ tốn nhiều tiền, quan trọng hơn nữa là việc quản lý hai hay nhiều CPU để chúng chia sẻ tốt tài nguyên chung. Ví dụ, cho tới trước khi chipset AMD 760MP được đưa ra, tất cả các nền tảng x86 đa xử lý chỉ hỗ trợ việc chia băng thông sẵn có giữa các CPU, điều quan trọng nhất là các ứng dụng và hệ điều hành cần phải có khả năng hỗ trợ tính năng này. Hiện nay, để giải quyết nhanh các chuỗi lệnh phức tạp, phần cứng nói chung phải nhờ vào phương án xử lý đa luồng, hệ điều hành phải hỗ trợ xử lý đa luồng, và phải tăng tốc độ một cách thật sự, giống như có nhiều bộ xử lý (trong hầu hết các trường hợp). Kỹ thuật siêu phân luồng của Intel giải quyết vấn đề bằng cách thực hiện nhiều hơn một chuỗi lệnh tại cùng một thời điểm.

- **Hiệu quả của các bộ vi xử lý**

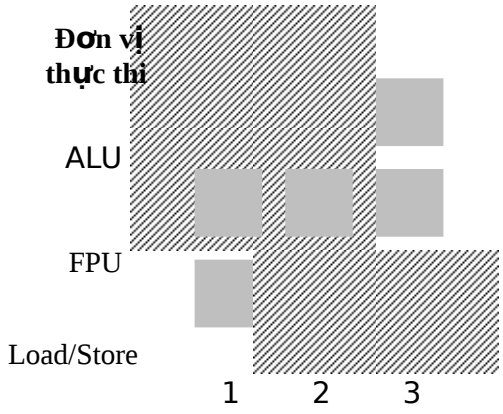
Lấy P4 làm ví dụ, CPU này có tổng cộng 7 đơn vị thực thi, hai trong số đó có thể thực hiện hai lệnh mỗi xung clock (gọi là double pumped ALUs). Nhưng ngay cả như vậy thì cũng không thể tìm được phần mềm nào tận dụng hết các đơn vị thực thi đó. Hầu hết các phần mềm cho máy tính cá nhân đang sử dụng chỉ làm việc với một ít

phép tính số nguyên như nạp và lưu trữ mà không hề động đến đơn vị thực thi dấu chấm động. Còn một số phần mềm chỉ tập trung vào mỗi đơn vị xử lý dấu chấm động mà không sử dụng đến đơn vị xử lý số nguyên. Ngay cả ứng dụng chủ yếu sử dụng phép tính số nguyên cũng không tận dụng tất cả các đơn vị xử lý số nguyên, đặc biệt là một thành phần trong CPU chuyên dùng cho phép dịch hay quay.

Giả sử một CPU với 3 đơn vị thực thi: một đơn vị số nguyên (ALU – Arithmetic Logic Unit), một đơn vị dấu chấm động (FPU – Floating Point Unit) và một đơn vị nạp/lưu trữ (đơn vị dùng để đọc/ghi bộ nhớ). Giả sử CPU có thể thực hiện mọi lệnh trong vòng một chu kỳ xung clock và đồng thời giải quyết nhiều lệnh tới cả ba đơn vị thực thi. Ta cần CPU thực thi chuỗi lệnh sau:

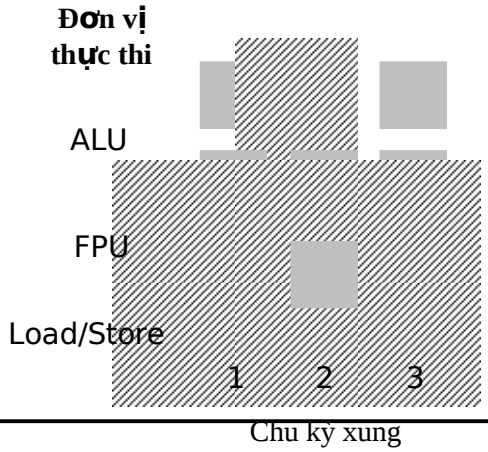
1+1
10+1
Lưu trữ kết quả

Biểu đồ dưới đây sẽ giúp minh họa mức độ của các đơn vị thực thi, màu xám biểu thị đơn vị thực thi không sử dụng, gạch chéo cho biết đơn vị thực thi hoạt động.



Có thể thấy rằng trong mỗi xung clock sẽ chỉ có 33% trong số các đơn vị được sử dụng, và trong các phép toán này hoàn toàn không sử dụng FPU.

Giả sử gửi một chuỗi lệnh khác đến các đơn vị thực thi của CPU, lần này là các lệnh tải, cộng và lưu trữ:

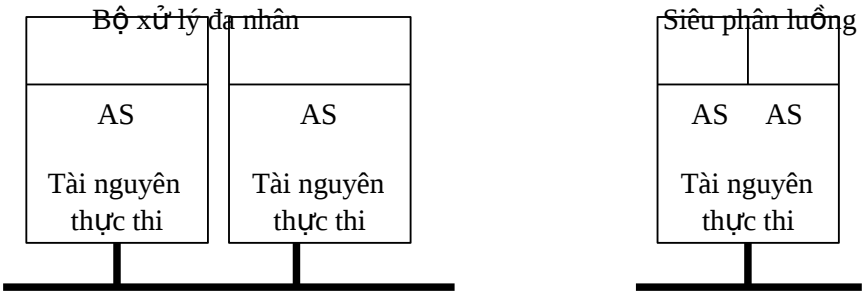


Ta thấy rằng cũng chỉ sử dụng có 33% số các đơn vị thực thi. Thuật toán xử lý song song được gọi là ILP (instruction level parallelism), ở đó các chỉ dẫn phức tạp được thực hiện đồng thời bởi vì CPU có khả năng tận dụng các đơn vị xử lý song song, tức là có nhiều hơn 33% số đơn vị xử lý được sử dụng. Tuy nhiên trên thực tế hầu hết các mã lệnh x86 không phải là ILP, vì vậy ta phải tìm những cách khác để tăng hiệu quả. Ví dụ, hệ thống có 2 CPU và chúng có thể thực hiện các chuỗi lệnh đồng thời, cách này được biết đến như là xử lý song song theo luồng để tăng cường hiệu năng, tuy nhiên lại rất tốn kém.

- **Kỹ thuật siêu phân luồng**

Các đơn vị thực thi không được sử dụng thường xuyên là do CPU không thể lấy dữ liệu nhanh như nó mong muốn do tắc nghẽn đường truyền (memory bus và front-side-bus), dẫn đến sự giảm sút hoạt động của các đơn vị thực thi. Ngoài ra, một nguyên nhân khác đã được đề cập là có quá ít ILP trong hầu hết các chuỗi lệnh thực thi.

Kỹ thuật siêu phân luồng



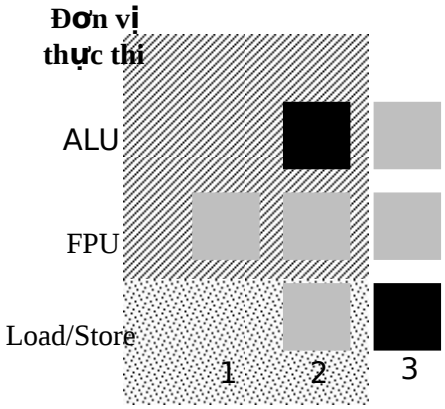
Hình 2.5 – So sánh bộ xử lý đa nhân và siêu phân luồng

Hiện thời đa số các phương pháp dùng để cải thiện hiệu năng trong các thế hệ CPU là tăng tốc độ xung clock và tăng độ lớn của bộ nhớ đệm (cache). Nhưng cho dù cả hai cách này cùng được sử dụng thì vẫn không thực sự sử dụng hết được tài nguyên sẵn có của CPU. Nếu có cách nào đó cho phép thực thi được nhiều chuỗi lệnh đồng thời mới có thể tăng hiệu quả sử dụng tài nguyên của CPU. Đó chính là cách mà kỹ thuật siêu phân luồng của Intel đã làm được, bản chất của nó là chia sẻ tài nguyên để sử dụng hiệu quả hơn các đơn vị thực thi lệnh đã có sẵn trên CPU.

Siêu phân luồng là một kỹ thuật nằm ngoài x86, là một phần nhỏ của SMT. Ý tưởng của SMT rất đơn giản: một CPU vật lý sẽ xuất hiện trên hệ điều hành như là hai CPU logic và hệ điều hành không thể phân biệt được. Nhiệm vụ của hệ điều hành là gửi 2 chuỗi lệnh tới 2 CPU và phần cứng sẽ đảm nhiệm những công việc còn lại.

Trong các CPU sử dụng kỹ thuật siêu phân luồng, mỗi CPU logic sở hữu một tập các thanh ghi, kể cả thanh ghi bộ đếm chương trình riêng (separate program counter), CPU vật lý sẽ luân phiên các giai đoạn tìm/giải mã lệnh giữa hai CPU logic và thực thi những thao tác từ hai chuỗi lệnh đồng thời theo cách hướng tới những đơn vị thực thi ít được sử dụng.

- **Hạn chế của siêu phân luồng**



Chu kỳ xung

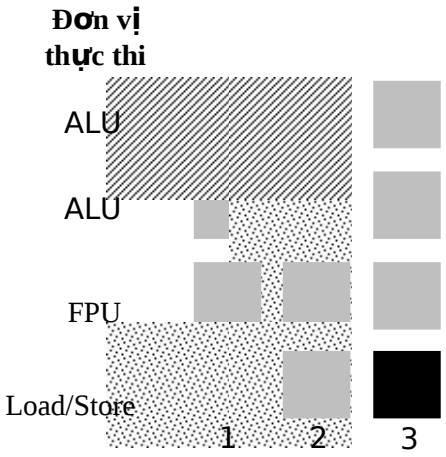
Giả sử rằng CPU đơn giản trước đây cũng có các đặc tính của siêu phân luồng:

Các ô gạch chéo hiển thị một chỉ dẫn từ chuỗi lệnh thứ nhất đang được thực hiện, trong khi những ô chấm chấm hiển thị một chỉ dẫn từ chuỗi lệnh thứ hai đang được thực hiện. Các ô màu xám hiển thị những đơn vị thực hiện không được sử dụng, trong khi các ô màu đen hiển thị xung đột khi mà cả hai chỉ dẫn đều sử dụng cùng một đơn vị thực thi. Rõ ràng là việc thực thi song song hai chuỗi lệnh với kỹ thuật siêu phân luồng lại thực hiện chậm hơn so với một CPU thông thường. Nguyên nhân thật ra rất đơn giản: CPU đồng thời thực hiện hai chuỗi lệnh quá đơn giản, tất cả đều là trùng lặp với lệnh add, load, store. Nếu thực thi các ứng dụng đòi hỏi nhiều phép toán động cùng với các ứng dụng số nguyên thì kết quả sẽ khác đi. Hiện tại các ứng dụng văn phòng trên máy tính để bàn hầu như chỉ sử dụng số nguyên (và trong tương lai chắc cũng vẫn chỉ sử dụng số nguyên). Vì vậy lợi ích mà công nghệ siêu phân luồng đem lại thấp (và đôi khi còn kém hơn không dùng công nghệ siêu phân luồng). Trên thực tế, nếu kích hoạt tính năng siêu phân luồng trên desktop, có thể giảm tốc độ tới 10%. Tuy nhiên người dùng các ứng dụng tính toán phức tạp thì sẽ được hưởng lợi rất nhiều từ kỹ thuật này. Ngoài ra kỹ thuật này cũng tăng tốc đáng kể cho các máy chủ, nhất là các máy chủ web server.

- **Lợi ích của siêu phân luồng**

Intel đã tạo ra siêu phân luồng không chỉ để cho các CPU máy chủ. Thực ra kiến trúc NetBurst của P4 và Xeon hiện nay hoàn chỉnh với lõi SMT. Xét ví dụ ở trên, ta cho thêm một ALU thứ 2 và thực hiện hai chuỗi lệnh trên.

Với một ALU thứ 2, xung đột duy nhất gặp phải là lần lưu trữ cuối cùng. Ta biết rằng CPU P4 được thiết kế với ba đơn vị số nguyên (hai ALU và một đơn vị xử lý số nguyên khác chậm hơn cho phép dịch/quay). Quan trọng hơn nữa là mỗi ALU của P4 có thể thực hiện hai vi lệnh trong cùng một xung clock, nghĩa là trong hai chỉ dẫn add (phép cộng) mỗi chỉ dẫn có thể từ hai chuỗi lệnh khác nhau, được thực hiện đồng thời trong một xung clock duy nhất trên P4/Xeon.



Chu kỳ xung

Nhưng điều đó vẫn chưa giải quyết được vấn đề, do việc tăng thêm các đơn vị xử lý để tăng hiệu quả với kỹ thuật siêu phân luồng lại tốn kém đứng từ quan điểm vật lý (làm cho CPU có nhiều transistor hơn, tiêu tốn nhiều điện năng hơn; hoặc phải giảm kích thước CPU với các công nghệ chế tạo mới). Thay vào đó, Intel đang khuyến khích các nhà phát triển tối ưu hoá kỹ thuật siêu phân luồng. Chẳng hạn sử dụng lệnh dừng (HALT) một trong các bộ xử lý logic sẽ tối đa được tốc độ cho các ứng dụng không sử dụng được kỹ thuật siêu phân luồng, CPU còn lại chỉ hoạt động như là hệ thống một CPU. Khi một ứng dụng có thể sử dụng lợi ích từ siêu phân luồng, bộ xử lý logic thứ hai lại tiếp tục được hoạt động.

3. Các thanh ghi

CPU 8086/8088 có tất cả 14 thanh ghi nội. Các thanh ghi này có thể phân loại như sau:

- Thanh ghi dữ liệu (data register)
- Thanh ghi chỉ số và con trỏ (index & pointer register)
- Thanh ghi đoạn (segment register)
- Thanh ghi trạng thái và điều khiển (status & control register)

3.1. Các thanh ghi dữ liệu

Các thanh ghi dữ liệu gồm có các thanh ghi 16 bit AX, BX, CX và DX trong đó nửa cao và nửa thấp của mỗi thanh ghi có thể định địa chỉ một cách độc lập. Các nửa thanh ghi này (8 bit) có tên là AH và AL, BH và BL, CH và CL, DH và DL.

Các thanh ghi này được sử dụng trong các phép toán số học và logic hay trong quá trình chuyển dữ liệu.

Thanh ghi	**Sử dụng trong**
AX	MUL, IMUL (toán hạng nguồn kích thước word) DIV, IDIV (toán hạng nguồn kích thước word) IN (nhập word) OUT (xuất word)

	CWD Các phép toán xử lý chuỗi (string)
AL	MUL, IMUL (toán hạng nguồn kích thước byte) DIV, IDIV (toán hạng nguồn kích thước byte) IN (nhập byte) OUT (xuất byte) XLAT AAA, AAD, AAM, AAS (các phép toán ASCII) CBW (đổi sang word) DAA, DAS (số thập phân) Các phép toán xử lý chuỗi (string)
AH	MUL, IMUL (toán hạng nguồn kích thước byte) DIV, IDIV (toán hạng nguồn kích thước byte) CBW (đổi sang word)
BX	XLAT
CX	LOOP, LOOPE, LOOPNE Các phép toán string với tiếp dầu ngữ REP
CL	RCR, RCL, ROR, ROL (quay với số đếm byte) SHR, SAR, SAL (dịch với số đếm byte)
DX	MUL, IMUL (toán hạng nguồn kích thước word) DIV, IDIV (toán hạng nguồn kích thước word)

AX (ACC – Accumulator): thanh ghi tích luỹ

BX (Base): thanh ghi cơ sở

CX (Count): đếm

DX (Data): thanh ghi dữ liệu

3.2. Các thanh ghi chỉ số và con trỏ

Bao gồm các thanh ghi 16 bit SP, BP, SI và DI, thường chứa các giá trị offset (độ lệch) cho các phần tử định địa chỉ trong một phân đoạn (segment). Chúng có thể được sử dụng trong các phép toán số học và logic. Hai thanh ghi con trỏ (SP – Stack Pointer và BP – Base Pointer) cho phép truy xuất dễ dàng đến các phần tử đang ở trong ngăn xếp (stack) hiện hành. Các thanh ghi chỉ số (SI – Source Index và DI – Destination Index) được dùng để truy xuất các phần tử trong các đoạn dữ liệu và doạn thêm (extra segment). Thông thường, các thanh ghi con trỏ liên hệ đến đoạn stack hiện hành và các thanh ghi chỉ số liên hệ đến doạn dữ liệu hiện hành. SI và DI dùng trong các phép toán chuỗi.

3.3. Các thanh ghi đoạn

Bao gồm các thanh ghi 16 bit CS (Code segment), DS (Data segment), SS (stack segment) và ES (extra segment), dùng để định địa chỉ vùng nhớ 1 MB bằng cách chia thành 16 đoạn 64 KB.

Tất cả các lệnh phải ở trong đoạn mã hiện hành, được định địa chỉ thông qua thanh ghi CS. Offset (độ lệch) của mã được xác định bằng thanh ghi IP. Dữ liệu chương trình thường được đặt ở đoạn dữ liệu, định vị thông qua thanh ghi DS. Stack

định vị thông qua thanh ghi SS. Thanh ghi đoạn thêm có thể sử dụng để định địa chỉ các toán hạng, dữ liệu, bộ nhớ và các phần tử khác ngoài đoạn dữ liệu và stack hiện hành.

3.4. Các thanh ghi điều khiển và trạng thái

Thanh ghi con trỏ lệnh IP (Instruction Pointer) giống như bộ đếm chương trình (Program Counter). Thanh ghi điều khiển này do BIU quản lý nhằm lưu trữ offset từ bắt đầu đoạn mã đến lệnh thực thi kế tiếp và không thể xử lý trực tiếp thanh ghi IP.

Thanh ghi cờ (Flag register) dài 16 bit chứa 3 bit điều khiển (TF, IF và DF) và 6 bit trạng thái (OF, SF, ZF, AF, PF và CF) còn các bit còn lại không sử dụng.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

- OF (Overflow - tràn): OF = 1 xác định tràn số học, xảy ra khi kết quả vượt ra ngoài phạm vi biểu diễn
- DF (Direction- hướng): xác định hướng chuyển chuỗi, DF = 1 khi CPU làm việc với chuỗi theo thứ tự từ phải sang trái và ngược lại.
- IF (Interrupt - ngắt): cho phép hay cấm các ngắt có mặt nạ.
- TF (Trap - bẫy): đặt CPU vào chế độ từng bước, dùng cho các chương trình gỡ rối (debugger).
- SF (Sign - dấu): dùng để chỉ các kết quả số học là số dương (SF = 0) hay âm (SF = 1).
- ZF (Zero): = 1 nếu kết quả của phép toán trước là 0.
- AF (Auxiliary – nhớ phụ): dùng trong các số thập phân để chỉ nhớ từ nửa byte thấp hay mượn từ nửa byte cao.
- PF (Parity): PF = 1 nếu kết quả của phép toán là có tổng số bit 1 là chẵn (dùng để kiểm tra lỗi truyền dữ liệu)
- CF (Carry): CF = 1 nếu có nhớ hay mượn từ bit cao nhất của kết quả. Cờ này cũng dùng cho các lệnh quay.

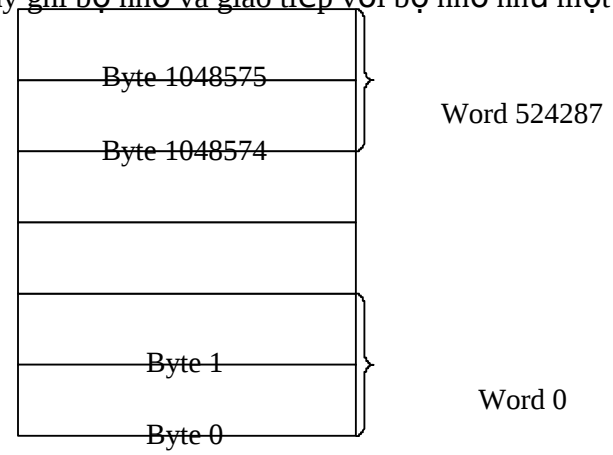
4. Phân đoạn bộ nhớ

Ta biết rằng dù 8086 là CPU 16 bit (có bus dữ liệu 16 bit) nhưng vẫn dùng bộ nhớ theo các byte. Điều này cho phép CPU làm việc với byte cũng như word, nó rất quan trọng trong giao tiếp với các thiết bị I/O như máy in, thiết bị đầu cuối và modem (chúng được thiết kế để chuyển dữ liệu mã hoá ASCII 7 hay 8 bit). Ngoài ra, nhiều mã lệnh của 8086/8088 có chiều dài 1 byte nên cần phải truy xuất được các byte riêng biệt để có thể xử lý các lệnh này.

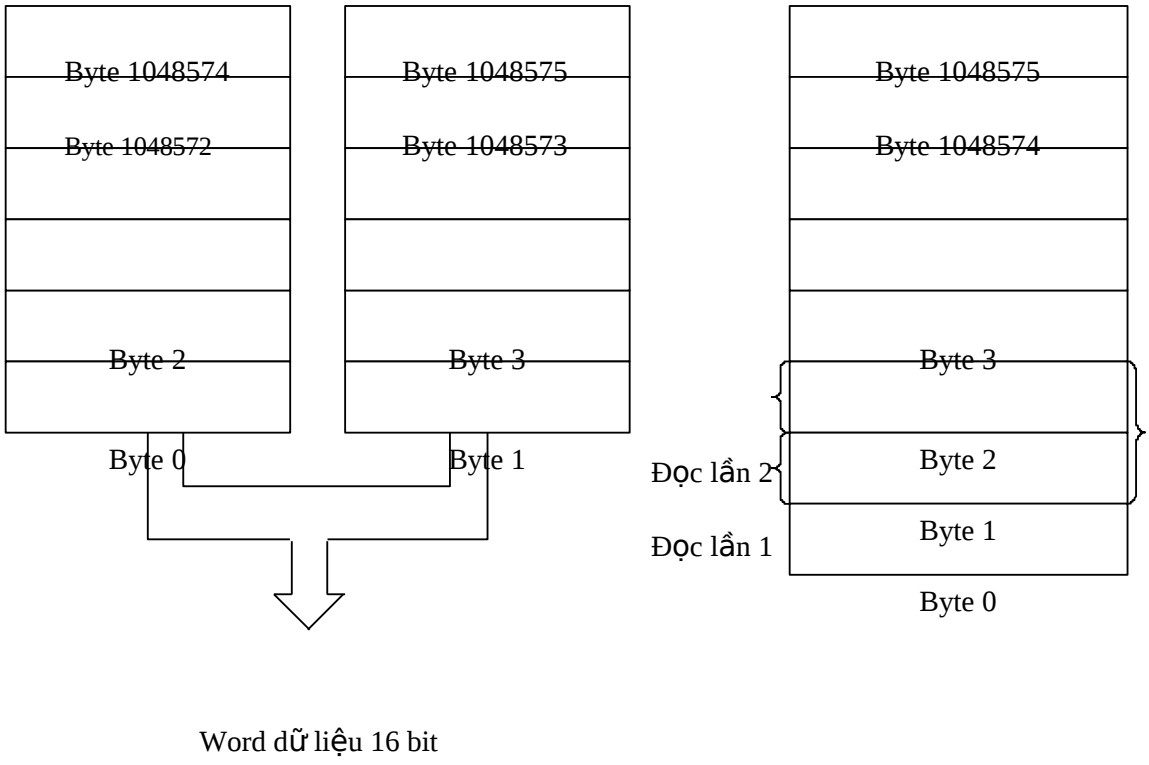
8086/8088 có bus địa chỉ 20 bit nên có thể cho phép truy xuất 2^{20} = 1048576 địa chỉ bộ nhớ khác nhau.

Để thực hiện đọc 16 bit từ bộ nhớ, 8086 sẽ thực hiện đọc đồng thời byte có địa chỉ lẻ và byte có địa chỉ chẵn. Do đó, 8086 tổ chức bộ nhớ thành các bank chẵn và lẻ.

Theo hình 2.6, ta có thể thấy rằng các word luôn bắt đầu tại địa chỉ chẵn nhưng ta vẫn có thể đọc word có địa chỉ lẻ bằng cách thực hiện 2 chu kỳ đọc bộ nhớ: một chu kỳ đọc byte thấp và một chu kỳ đọc byte cao nhưng điều này làm chậm tốc độ xử lý. Đối với 8088 thì do bus dữ liệu 8 bit nên dù word có địa chỉ chẵn hay lẻ, nó cũng cần phải thực hiện 2 chu kỳ đọc hay ghi bộ nhớ và giao tiếp với bộ nhớ như một bank.

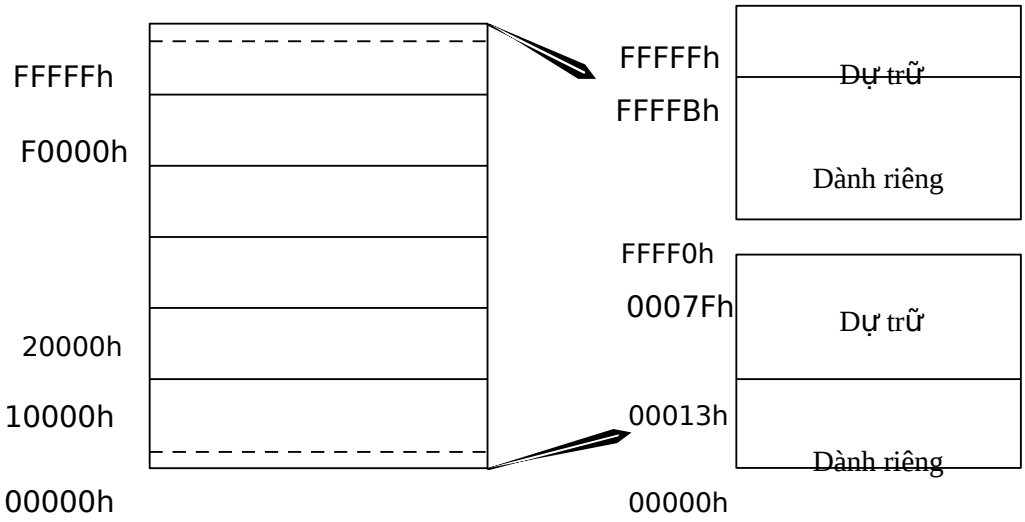


Hình 2.6 – Vùng nhớ của 8086/8088 có 1048576 byte hay 524288 word



Hình 2.7 – Đọc word địa chỉ chẵn và địa chỉ lẻ

Ngoài ra bộ nhớ cũng chia thành 16 khối, mỗi khối có kích thước 64 KB, bắt đầu ở địa chỉ 00000h và kết thúc ở FFFFFh. Địa chỉ bắt đầu mỗi khối sẽ tăng lên 1 ở số hex có ý nghĩa nhiều nhất khi thay đổi từ khối này sang khối kia. Ví dụ như khối 00000h → 10000h → 20000h …



Hình 2.8 – Bảng bộ nhớ cho 8086/8088

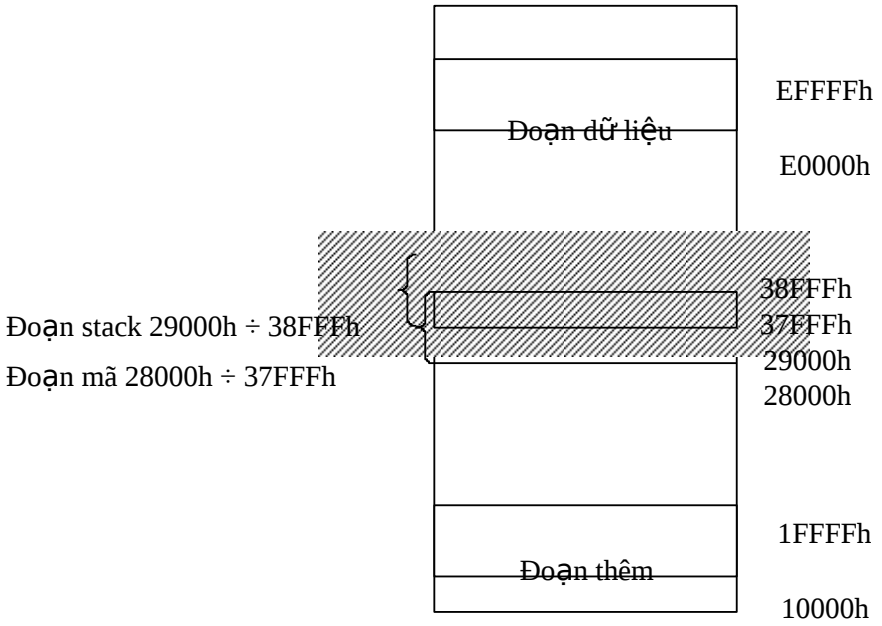
8086/8088 định nghĩa 4 khối bộ nhớ 64KB: đoạn mã (code segment) giữ các mã lệnh chương trình, đoạn ngăn xếp (stack segment) lưu các địa chỉ sẽ trả về từ các chương trình con (subroutine) hay trình phục vụ ngắt (interrupt subroutine), đoạn dữ liệu (data segment) lưu trữ dữ liệu cho chương trình và đoạn thêm (extra segment) thường dùng cho các dữ liệu dùng chung.

Các thanh ghi đoạn (CS, DS, SS và ES) dùng để chỉ vị trí nền của mỗi đoạn. Các thanh ghi này có 16 bit trong khi địa chỉ bộ nhớ là 20 bit nên để xác dịnh vị trí bộ nhớ, ta sẽ thêm 4 bit 0 vào các bit thấp của thanh ghi đoạn. Giả sử như thanh ghi CS chứa giá trị 1111h thì nó sẽ chỉ tới địa chỉ nền là 11110h. Chú ý rằng địa chỉ bắt đầu một đoạn không thể tuỳ ý mà phải bắt đầu tại một địa chỉ chia hết cho 16. Nghĩa là 4 bit thấp phải là 0. Ta cũng chú ý rằng 4 đoạn có thể không tách rời nhau mà chồng lấp lên nhau và ta cũng có thể cho 4 giá trị của các thanh ghi đoạn bằng nhau nghĩa là 4 đoạn này trùng nhau.

VD: Thanh ghi DS có giá trị là 1000h thì địa chỉ nền là 10000h. Địa chỉ kết thúc tìm được bằng cách cộng địa chỉ nền với giá trị FFFFh (64K) → địa chỉ kết thúc là 10000h + FFFFh = 1FFFFh. Như vậy đoạn dữ liệu có địa chỉ từ 10000h ÷ 1FFFFh.

Các vị trí bộ nhớ không được định nghĩa trong các đoạn hiện hành không thể truy xuất được. Muốn truy xuất đến các vị trí đó, ta phải định nghĩa lại một trong các thanh ghi đoạn sau cho đoạn phải chứa vị trí đó. Như vậy, tại một thời điểm bất kỳ ta chỉ có thể truy xuất tối đa 4 × 64 KB = 256 KB bộ nhớ. Nội dung của các thanh ghi đoạn chỉ có thể xác định thông qua phần mềm.

VD: Giả sử các thanh ghi đoạn có các giá trị CS = 2800h, DS = E000h, SS = 2900h và ES = 1000h. Ta có vị trí các đoạn trong bảng bộ nhớ như sau:



Hình 2.9 – Vị trí các phân đoạn theo giá trị các thanh ghi đoạn

- **Địa chỉ logic và địa chỉ vật lý:**

Các địa chỉ trong một đoạn thay đổi từ 0000h ÷ FFFFh, tương ứng với chiều dài đoạn là 64 KB. Một địa chỉ trong một đoạn được gọi là ***địa chỉ logic hay offset***. Ví dụ như địa chỉ logic 0010h của đoạn mã trong hình 2.9 sẽ có địa chỉ thật sự là 28000h + 0010h = 28010h. Địa chỉ này gọi là ***địa chỉ vật lý***. Địa chỉ vật lý chính là địa chỉ thật sự xuất hiện ở bus địa chỉ, nó có chiều dài 20 bit còn địa chỉ logic là độ lệch (offset) từ vị trí 0 của một đoạn cho trước.

VD: Giả sử xét các đoạn như hình 2.9. Địa chỉ vật lý tương ứng với địa chỉ logic 1000h trong đoạn stack là:

29000h + 1000h = 2A000h

Địa chỉ vật lý tương ứng với địa chỉ logic 2000h trong đoạn mã là:

28000h + 2000h = 2A000h

Ta thấy rằng có thể địa chỉ vật lý trùng nhau khi địa chỉ logic khác nhau nghĩa là một địa chỉ vật lý có thể có nhiều địa chỉ logic khác nhau.

Để chỉ địa chỉ logic 1000h trong đoạn mã, ta dùng ký hiệu CS:1000h. Tương tự như vậy cho các đoạn khác, nghĩa là địa chỉ logic 1111h trong đoạn dữ liệu sẽ là DS:1111h.

Mọi lệnh tham chiếu bộ nhớ sẽ có một thanh ghi đoạn mặc nhiên. Thanh ghi IP cung cấp địa chỉ offset khi truy xuất đến đoạn mã và BP cho đoạn stack. Ví dụ như IP = 1000h và CS = 2000h thì BIU sẽ truy xuất đến địa chỉ 20000h + 1000h = 21000h và nhận byte tại vị trí này.

Tham chiếu bộ nhớ	Đoạn mặc nhiên	Đoạn khác	Offset
Nhận lệnh	CS	Không	IP
Tác vụ stack	SS	Không	SP
Dữ liệu tổng quát	DS	CS,ES,SS	Địa chỉ hiệu dụng
Nguồn của string	DS	CS,ES,SS	SI
Đích của string	ES	Không	DI
BX dùng làm con trỏ	DS	CS,ES,SS	Địa chỉ hiệu dụng
BP dùng làm con trỏ	SS	CS,ES,SS	Địa chỉ hiệu dụng

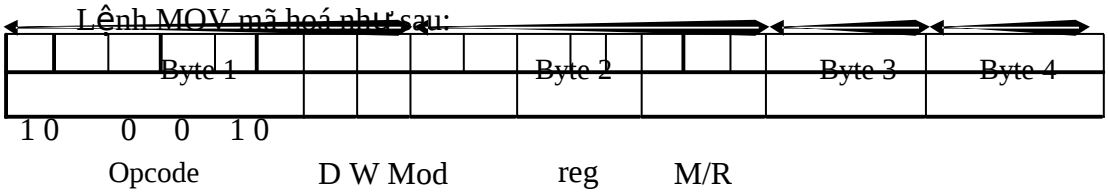
VD: Ta sử dụng lệnh MOV [BP],AL với BP = 2C00h. Ở đây BP dùng làm con trỏ nên dùng đoạn stack. Giả sử các phân đoạn như hình 2.9 thì địa chỉ vật lý sẽ là 29000h + 2C00h = 2BC00h

5. Cách mã hoá lệnh

Lệnh của CPU sẽ biểu diễn bằng các ký tự dưới dạng gợi nhớ (mnemonic) để có thể dễ dàng sử dụng. Đối với CPU thì các lệnh được biểu diễn bằng các mã lệnh (opcode) nên sau khi nhận lệnh CPU phải thực hiện giải mã lệnh rồi mới thực thi nó. Một lệnh CPU có thể dài 1 byte hay nhiều byte. Nếu ta dùng 1 byte để mã hoá thì sẽ mã hoá được 256 lệnh khác nhau. Tuy nhiên do một lệnh không phải chỉ có một cách thực hiện nên ta không thể thực hiện đơn giản như trên.

Để tìm hiểu cách mã hoá lệnh, ta xét lệnh **MOV des,src** dùng để chuyển dữ liệu giữa hai thanh ghi hay một ô nhớ và một thanh ghi.

Lệnh MOV mã hoá như sau:



Để mã hóa lệnh MOV, ta cần dùng ít nhất là 2 byte trong đó 6 bit dùng cho mã lệnh.

Bit D xác định hướng truyền của dữ liệu, D = 0 xác định dữ liệu sẽ đi từ thanh ghi cho bởi 3 bit Reg, D = 1 xác định dữ liệu sẽ đi đến thanh ghi cho bởi 3 bit Reg.

Bit W xác định sẽ truyền 1 byte (W = 0) hay 1 word (W = 1).

3 bit Reg dùng để chọn thanh ghi sử dụng:

Mã	Thanh ghi W = 1	Thanh ghi W = 0
000		
001		
010		
011		
100		
101		
110		
111		

AX	AL
CX	CL
DX	DL
BX	BL
SP	AH
BP	CH
SI	DH
DI	BH

2 bit mod và 3 bit R/M (Register / Memory) dùng để xác định chế độ địa chỉ cho các toán hạng của lệnh.

MOD / R/M	00	01	10	11	
				W = 1	W = 0
000	[BX]+[SI]	[BX]+[SI]+addr8	[BX]+[SI]+addr16	AX	AL
001	[BX]+[DI]	[BX]+[DI]+addr8	[BX]+[DI]+addr16	CX	CL
010	[BP]+[SI]	[BP]+[SI] +addr8	[BP]+[SI] +addr16	DX	DL
011	[BP]+[DI]	[BP]+[DI] +addr8	[BP]+[DI] +addr16	BX	BL
100	[SI]	[SI] +addr8	[SI] +addr16	SP	AH
101	[DI]	[DI] +addr8	[DI] +addr16	BP	CH
110	addr16	[BP] +addr8	[BP] +addr16	SI	DH
111	[BX]	[BX] +addr8	[BX] +addr16	DI	BH

Tổng quát, 8086/8088 có khoảng 300 tác vụ có thể có trong tập lệnh của nó. Mỗi lệnh kéo dài từ 1 đến 6 byte. Từ ví dụ trên, ta thấy mã lệnh có các vùng:

- Vùng mã lệnh (opcode): chứa mã lệnh của lệnh sẽ thực thi
- Vùng thanh ghi (reg): chứa các thanh ghi sẽ thực hiện
- Vùng chế độ (mod)
- Vùng thanh ghi / bộ nhớ R/M (Reg/Mem)

6. Các cách định địa chỉ

Cách định địa chỉ	Mã đối tượng	Từ gợi nhớ	Ví dụ Đoạn truy xuất	Hoạt động	Mô tả
Tức thời	B80010	MOV AX,1000h	Mã	AH ← 10h AL ← 00h	(1)
Thanh ghi	8BD1	MOV DX,CX	Trong µP	DX ← CX	(2)
Trực tiếp	8A260010	MOV AH,[1000h]	Dõõ liệu	AH ← [1000h]	(3)
Gián tiếp thanh ghi	8B04 FF25 FE4600 FF0F	MOV AX,[SI] JMP [DI] INC BYTE PTR [BP] DEC WORD PTR [BX]	Dữ liệu Dữ liệu Stack Dữ liệu	AL ← [SI]; AH ← [SI+1] IP ← [DI+1:DI] [BP] ← [BP]+1 [BX+1:BX] ← [BX+1:BX]-1	(4)
Có chỉ số	8B4406 FF6506	MOV AX,[SI+6] JMP [DI+6]	Dữ liệu Dữ liệu	AL ← [SI+6]; AH ← [SI+7] IP ← [DI+7:DI+6]	(5)

Có nền	8B4602 FF6702	MOV AX,[BP+2] JMP [BP+2]	Stack Dữ liệu	AL ← [BP+2]; AH ← [BP+3] IP ← [BX+3:BX+6]	(6)
Có nền và có chỉ số	8B00 FF21 FE02 FF0B	MOV AX,[BX+SI] JMP [BX+DI] INC BYTE PTR [BP+SI] DEC WORD PTR [BP+DI]	Dữ liệu Dữ liệu Stack Stack	AL ← [BX+SI];AH ← [BX+SI+1] IP ← [BX+DI+1:BX+DI] [BP+SI] ← [BP+SI]+1 [BP+DI+1:BP+DI] ← [BP+DI+1:BP+DI]-1	(7)

Có nền và có chỉ số với độ dời	8B4005 FF6105 FE4205 FF4B05	MOV AX,[BX+SI+5] JMP [BX+DI+5] INC BYTE PTR [BP+SI+5] DEC WORD PTR [BP+DI+5]	Dữ liệu Dữ liệu Stack Stack	AL ← [BX+SI+5] AH ← [BX+SI+1] IP ← [BX+DI+6:BX+DI+5] [BP+SI+5] ← [BP+SI+5]+1 [BP+DI+6:BP+DI+5] ← [BP+DI+6:BP+DI+5]-1	(8)
String	A4	MOVSB	Thêm, dữ liệu	[ES:DI] ← [DS:DI] Nếu DF = 0 thì SI ← SI + 1; DI ← DI + 1 Nếu DF = 1 thì SI ← SI - 1; DI ← DI - 1	(9)

- BYTE PTR và WORD PTR tránh lầm giữa truy xuất byte và word.
- Độ dời được cộng vào thanh ghi con trỏ hay nền là số nhị phân dạng bù 2.
- (1): nguồn dữ liệu trong lệnh
- (2): đích và nguồn là các thanh ghi của µP
- (3): địa chỉ bộ nhớ cung cấp trong lệnh
- (4): địa chỉ bộ nhớ cung cấp trong thanh ghi con trỏ hay chỉ số
- (5): địa chỉ bộ nhớ là tổng của thanh ghi chỉ số cộng với độ dời trong lệnh
- (6): địa chỉ bộ nhớ là tổng của thanh ghi BX hay BP cộng với độ dời trong lệnh
- (7): địa chỉ bộ nhớ là tổng của thanh ghi chỉ số và thanh ghi nền
- (8): địa chỉ bộ nhớ là tổng của thanh ghi chỉ số, thanh ghi nền và độ dời trong lệnh
- (9): địa chỉ nguồn bộ nhớ là thanh ghi SI trong đoạn dữ liệu và địa chỉ đích bộ nhớ là thanh ghi DI trong đoạn thêm

6.1. Định địa chỉ tức thời

Các lệnh dùng cách định địa chỉ tức thời lấy dữ liệu trong lệnh làm một phần của lệnh. Trong cách này, dữ liệu sẽ được chứa trong đoạn mã thay vì trong đoạn dữ liệu. Dữ liệu cho lệnh ***MOV AX,1000h*** được cung cấp tức thời sau mã lệnh B8. Chú ý rằng trong mã đối tượng byte dữ liệu cao đi sau byte dữ liệu thấp.

Cách định địa chỉ tức thời thường dùng để nạp một thanh ghi hay vị trí bộ nhớ với các dữ liệu ban đầu. Sau đó, các lệnh kế tiếp sẽ làm việc với các dữ liệu này. Tuy nhiên, cách định địa chỉ này không sử dụng được cho các thanh ghi đoạn.

6.2. Định địa chỉ thanh ghi

Một số lệnh chỉ làm công việc chuyển dữ liệu giữa các thanh ghi của CPU. Ví dụ như ***MOV DX,CX*** sẽ chuyển dữ liệu từ thanh ghi CX vào thanh ghi DX. Ở đây ta không cần thực hiện tham chiếu bộ nhớ.

Ta có thể kết hợp cách định địa chỉ tức thời và định địa chỉ thanh ghi để nạp dữ liệu cho các thanh ghi đoạn.

6.3. Định địa chỉ trực tiếp

Ngoài 2 cách định địa chỉ trên, tất cả các cách định địa chỉ còn lại đều cần phải truy xuất đến bộ nhớ với ít nhất một toán hạng. Trong cách định địa chỉ trực tiếp, địa

chỉ bộ nhớ được cung cấp trực tiếp như là một phần của lệnh. Ví dụ như lệnh ***MOV***

AH,[1000h] sẽ đưa nội dung chứa trong ô nhớ DS:1000h vào thanh ghi AH hay lệnh ***MOV [2000h],AX*** sẽ đưa nội dung chứa trong AX vào 2 ô nhớ liên tiếp DS:2000h và DS:2001h

6.4. Định địa chỉ truy xuất bộ nhớ gián tiếp

Các cách định địa chỉ trực tiếp sẽ thuận lợi cho các truy xuất bộ nhớ không thường xuyên. Tuy nhiên, nếu một ô nhớ cần phải truy xuất nhiều lần trong một chương trình thì quá trình nhận địa chỉ (2 byte) sẽ phải thực hiện nhiều lần. Điều này sẽ không hiệu quả. Để giải quyết vấn đề này, ta thực hiện lưu trữ địa chỉ của ô nhớ cần truy xuất trong một thanh ghi con trỏ, chỉ số hay thanh ghi cơ sở (BX, BP, SI hay DI). Ngoài ra, ta có thể sử dụng độ dời bù 2 bằng cách cộng vào các thanh ghi để dời đi so với vị trí được các thanh ghi chỉ đến.

Cách định địa chỉ	**Địa chỉ hiệu dụng (EA – Effective Address)**		
	Độ dời	**Thanh ghi nền**	**Thanh ghi chỉ số**
Gián tiếp thanh ghi	Không	BX hay BP	Không
	Không	Không	SI hay DI
Có chỉ số	-128 ÷ 127	Không	SI hay DI
Có nền	-128 ÷ 127	BX hay BP	Không
Có nền và chỉ số	Không	BX hay BP	SI hay DI
Có nền và chỉ số với độ dời	-128 ÷ 127	BX hay BP	SI hay DI

Như vậy, một độ dời có thể được cộng vào thanh ghi nền và kết quả này được cộng tiếp vào thanh ghi chỉ số. Địa chỉ thu được gọi là địa chỉ hiệu dụng EA.

Ngoài ra ta cũng có thể viết cách định địa chỉ gián tiếp như sau:

MOV AX,table[SI]

Trong đó ***table*** là nhãn gán cho một vị trí ô nhớ nào đó. Lệnh này sẽ truy xuất phần tử thứ SI trong dãy ***table*** (giả sử SI = 2 thì sẽ truy xuất phần tử thứ 2).

Chú ý rằng các đoạn mặc định cho các cách định địa chỉ gián tiếp là đoạn stack khi dùng BP, là đoạn dữ liệu khi dùng BX, SI hay DI.

VD: Lệnh:

MOV AH,10h	thực hiện định địa chỉ tức thời
MOV AX,[BP + 10]	thực hiện định địa chỉ có nền
MOV AH,[BP + SI]	thực hiện định địa chỉ có nền và có chỉ số

6.5. Định địa chỉ chuỗi

Chuỗi là một dãy liên tục các byte hay word lưu trữ trong bộ nhớ dưới dạng các ký tự ASCII. 8086/8088 có các lệnh dùng để xử lý chuỗi, các lệnh này sử dụng cặp thanh ghi DS:SI để chỉ nguồn chuỗi ký tự và ES:DI để chỉ đích chuỗi. Lệnh MOVSB sẽ chuyển byte dữ liệu nguồn đến vị trí đích trong đó SI và DI sẽ tăng hay giảm tuỳ theo giá trị của DF.

Chương 3
BỘ NHỚ

1. Một số khái niệm

1.1. Bộ nhớ (memory)

Là thiết bị nhớ có thể ghi và chứa thông tin. ROM, RAM, cache, đĩa cứng, đĩa mềm, CD.... đều có thể gọi là bộ nhớ (vì chúng đều lưu trữ thông tin). Các tính chất:

- *Dung lượng*: khả năng lưu trữ dữ liệu của thiết bị. Ví dụ: CD chứa được 700MB, đĩa mềm chứa được 1.44MB, đĩa cứng chứa được 40 GB, 60GB, cache L1 chứa được 16KB, cache L2 chứa được 256 KB ...
- *Tốc độ truy nhập*: liên quan đến tốc độ truyền dữ liệu của thiết bị. Tính về tốc độ thì CPU là lớn nhất, kế tiếp là Cache, sau nữa là các loại RAM.
- *Giao tiếp*: cấu trúc bên ngoài của bộ nhớ. Ví dụ, các RAM có số chân cắm và đặc tính khác nhau.

1.2. Phân loại bộ nhớ

1.2.1. ROM (Read Only Memory)

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.

1.2.2. PROM (Programmable ROM)

Mặc dù ROM nguyên thủy là không ghi hay xóa được, nhưng các thế hệ sau của ROM đã đa dụng hơn như PROM. Các hãng sản xuất có thể cài đặt lại ROM bằng cách dùng các loại dụng cụ đặc biệt và đắt tiền. Thông tin có thể cài đặt vào chip và nó sẽ lưu lại mãi trong chip. Một đặc điểm lớn nhất của loại PROM là thông tin chỉ cài đặt một lần mà thôi. CD cũng có thể được gọi là PROM vì chúng ta có thể lưu trữ thông tin vào nó chỉ một lần duy nhất và không thể xoá được.

1.2.3. EPROM (Erasable Programmable ROM)

Một dạng cao hơn PROM là EPROM, tức là ROM có thể xoá và ghi lại được. EPROM khác PROM ở chỗ là thông tin có thể được viết và xoá nhiều lần theo ý người sử dụng, và phương pháp xoá là phần cứng (dùng tia hồng ngoại).

1.2.4. EEPROM (Electrically Erasable Programmable ROM)

Đây là một dạng cao hơn EPROM, đặt điểm khác biệt duy nhất so với EPROM là có thể ghi và xoá thông tin lại nhiều lần bằng phần mềm.

1.2.5. RAM (Random Access Memory)

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.

1.2.6. SRAM (Static RAM) và DRAM (Dynamic RAM)

SRAM (RAM tĩnh) là loại RAM lưu trữ dữ liệu không cần cập nhật thường xuyên trong khi DRAM là loại RAM cần cập nhật dữ liệu thường xuyên. Thông thường dữ liệu trong DRAM sẽ được làm tươi (refresh) nhiều lần trong một giây để giữ lại những thông tin đang lưu trữ, nếu không thì dữ liệu trong DRAM cũng sẽ bị mất do hiện tượng rò rỉ điện tích của các tụ điện. 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.

1.2.7. FPM - DRAM (Fast Page Mode DRAM)

Là một dạng cải tiến của DRAM, về nguyên lý thì FPM - DRAM sẽ chạy nhanh hơn DRAM do cải tiến cách dò địa chỉ trước khi truy xuất dữ liệu. FPM - DRAM hầu như không còn sản xuất trên thị trường hiện nay nữa.

1.2.8. EDO - DRAM (Extended Data Out DRAM)

Là một dạng cải tiến của FPM - DRAM, nó truy xuất nhanh hơn FPM - DRAM nhờ một số cải tiến cách dò địa chỉ trước khi truy cập dữ liệu. Tuy nhiên, EDO - DRAM là cần hỗ của chipset hệ thống. Loại bộ nhớ nầy chạy với máy 486 trở lên (tốc độ dưới 75MHz). EDO DRAM cũng đã quá cũ so với kỹ thuật hiện nay, tốc độ của EDO-DRAM nhanh hơn FPM-DRAM từ 10 - 15%.

1.2.9. 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.

1.2.10. 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.

1.2.11. 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)

1.2.12. 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

1.2.13. 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.

1.2.14. 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.

1.2.15. SGRAM (Synchronous Graphic RAM)

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

1.2.16. Flash Memory

Là sản phẩm kết hợp giữa RAM và đĩa cứng, bộ nhớ flash có thể chạy nhanh như SDRAM mà và vẫn lưu trữ được dữ liệu khi không có nguồn cung cấp.

1.2.17. Một số thuật ngữ liên quan

- PC66, PC100, PC133, PC1600, PC2100, PC2400:

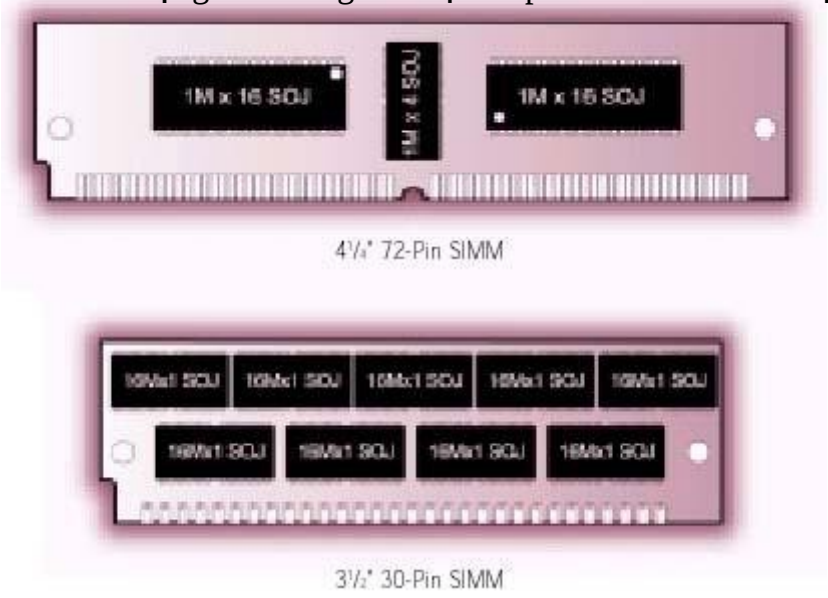
PC66, 100, 133MHz là tốc độ của hệ thống chipset của motherboard. PC1600, PC2100, PC2400: ra đời khi kỹ thuật Rambus phát triển. Đặc điểm của loại motherboard này là dùng loại DDR SDRAM (Double Data Rate Synchronous Dynamic RAM). DDR SDRAM sẽ chạy gấp đôi (trên lý thuyết) loại RAM bình thường vì nó dùng cả cạnh

dương và âm của xung clock. Do đó PC100 sẽ thành PC200 và nhân lên 8 bytes độ rộng

bus của DDR SDRAM: PC200 * 8 = PC1600. Tương tự PC133 sẽ là PC133 * 2 * 8bytes = PC2100 và PC150 sẽ là PC150 * 2 * 8 = PC2400.

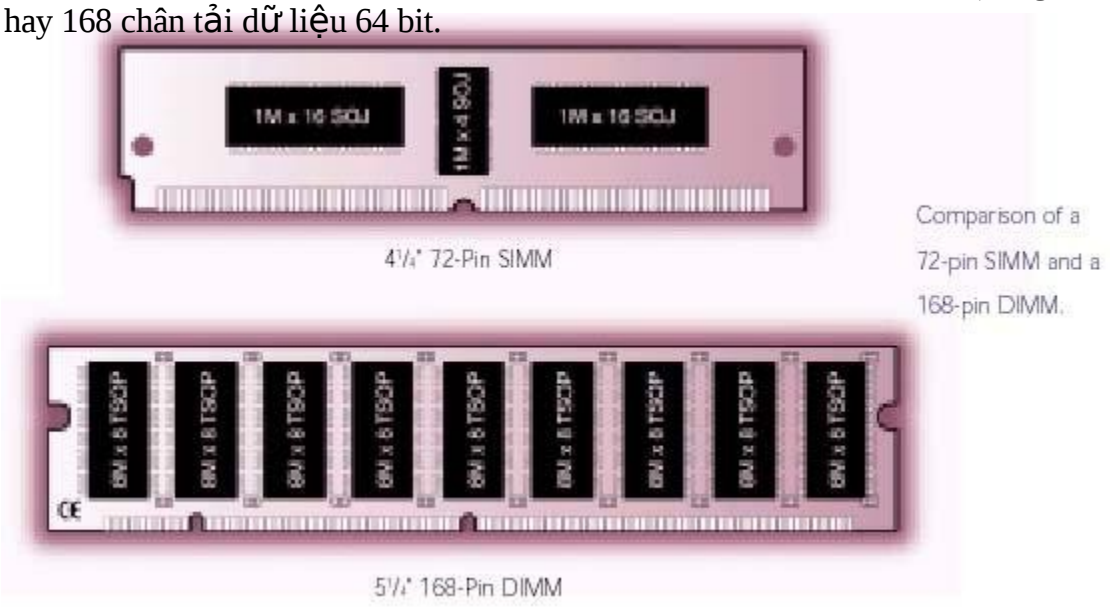
- BUS: gồm nhiều dây dẫn điện nhỏ gộp lại, là hệ thống truyền dữ liệu giữa các bộ phận trong máy tính.
- FSB (Front Side Bus): bus từ CPU tới bộ nhớ chính.
- BSB (Back Side Bus): bus từ bộ điều khiển bộ nhớ tới Cache level 2.
- Cache memory: Là loại bộ nhớ có dung lượng rất nhỏ (thường nhỏ hơn 1MB) và chạy rất nhanh (gần bằng tốc độ của CPU). Thông thường thì Cache nằm gần CPU và có nhiệm vụ cung cấp những dữ liệu thường hay đang sử dụng cho CPU. Sự hình thành của Cache là một cách nâng cao hiệu quả truy xuất của máy tính mà thôi. Những dữ liệu thường sử dụng (hoặc đang) được chứa trong Cache, mỗi khi xử lý hay thay đổi dữ liệu, CPU sẽ dò trong Cache trước xem có tồn tại hay không, nếu có nó sẽ lấy ra dùng lại còn không thì sẽ tìm tiếp vào RAM hoặc các bộ phận khác. Lấy một ví dụ đơn giản là nếu mở Microsoft Word lên lần đầu tiên sẽ thấy hơi lâu nhưng mở lên lần thứ hai thì nhanh hơn rất nhiều vì trong lần mở thứ nhất các lệnh để mở Microsoft Word đã được lưu giữ trong Cache, CPU chỉ việc tìm nó và dùng lại. Cache rất đắt tiền và chế tạo rất khó khăn bởi nó gần như là CPU (về cấu thành và tốc độ). Thông thường Cache nằm gần CPU, trong nhiều trường hợp Cache nằm bên trong CPU. Người ta gọi Cache Level 1 (L1), Cache level 2 (L2)...là do vị trí của nó gần hay xa CPU. Cache L1 gần CPU nhất, sau đó là Cache L2...
- Xen kẽ (interleave): là một kỹ thuật làm tăng tốc độ truy xuất bằng cách giảm bớt thời gian nhàn rỗi của CPU. Ví dụ, CPU cần đọc thông tin thông từ hai nơi A và B khác nhau, vì CPU chạy quá nhanh nên A chưa kịp lấy dữ liệu ra, CPU phải chờ. Khi đó CPU có thể lấy dữ liệu từ B rồi sau đó trở lại A. Do đó, CPU có thể rút bớt thời gian mà lấy được dữ liệu ở cả A và B.
- Bursting: là một kỹ thuật khác để giảm thời gian truyền tải dữ liệu trong máy tính. Thay vì CPU lấy từng byte một, bursting sẽ giúp CPU lấy thông tin mỗi lần là một block.
- ECC (Error Correction Code): là một kỹ thuật để kiểm tra và sửa lỗi trong trường hợp 1 bit nào đó của bộ nhớ bị sai giá trị trong khi lưu chuyển dữ liệu. Những loại RAM có ECC thường dùng cho server. Tuy nhiên không có ECC cũng không phải là mối lo lớn vì theo thống kê 1 bit trong bộ nhớ có thể bị sai giá trị khi chạy trong gần 750 giờ (tức là khoảng 1 tháng).
- CAS (Column Address Strobe) latency: là diễn tả thời gian trễ trong việc truy xuất dữ liệu của bộ nhớ và được tính bằng chu kỳ xung clock. Ví dụ, CAS3 là trễ 3 chu kỳ xung clock. Các nhà sản xuất cố gắng hạ thấp chỉ số trễ xuống nhưng nó sẽ tỷ lệ nghịch với giá thành sản phẩm.
- Số chân của RAM: thông thường là 30, 72, 144, 160, 168, 184.
- Cách tính dung lượng: Thông thường RAM có hai chỉ số, ví dụ, 16Mx8. Thông số đầu biểu thị số hàng của RAM trên đơn vị bit, thông số thứ nhì biểu thị số cột của RAM. 16Mx8 = 16 MegaBit x 8 cột = 128 Mega Bit = 16MB.

- SIMM (Single In-Line Memory Module): là loại ra đời sớm và có hai loại 30 hay 72 chân. Loại RAM thường tải dữ liệu mỗi lần 8 bit, sau đó phát triển lên 32 bit. Loại 72-pin SIMM có chiều rộng 4½" trong khi loại 30-pin SIMM có chiều rộng 3½".



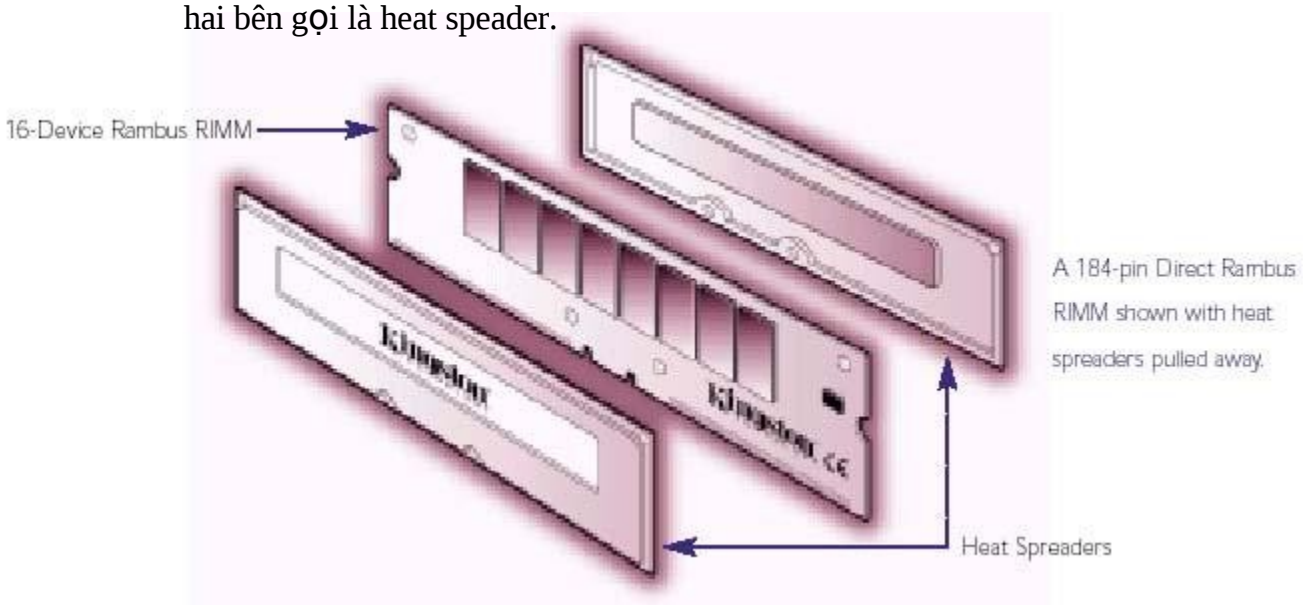
Hình 3.1 – Dạng chân của SIMM

- DIMM (Dual In-line Memory Modules): cũng gần giống như loại SIMM nhưng có số chân là 72 hoặc 168. Một đặc điểm khác để phân biệt DIMM với SIMM là các chân của SIMM dính lại với nhau tạo thành một mảng để tiếp xúc với memory slot trong khi DIMM có các chân hoàn toàn cách rời độc lập với nhau. Một đặc điểm phụ nữa là DIMM được cài đặt thẳng đứng trong khi SIMM thì ấn vào nghiêng khoảng 45^0. Thông thường loại 30 chân tải dữ liệu 16 bit, loại 72 chân tải dữ liệu 32 bit, loại 144 (dùng cho notebook) hay 168 chân tải dữ liệu 64 bit.



Hình 3.2 – Dạng chân của DIMM

- SO DIMM (Small Outline DIMM): là loại bộ nhớ dùng cho notebook, có hai loại chân là 72 hoặc 144. Loại 72 chân dùng bus 32 bit, loại 144 chân dùng bus 64 bit.
- RIMM (Rambus In-line Memory Modules) và SO RIMM (RIMM dùng cho notebook): là kỹ thuật của hãng Rambus, có 184 chân (RIMM) và 160 chân (SO RIMM) và truyền dữ liệu mỗi lần 16 bit (thế hệ cũ chỉ có 8 bit) nên chạy nhanh hơn các loại cũ. Tuy nhiên do chạy với tốc độ cao, RIMM tụ nhiệt rất cao nên cách chế tạo cũng phải khác so với các loại RAM truyền thống. Như hình vẽ bên dưới bạn sẽ thấy RAM có hai thanh giải nhiệt kẹp hai bên gọi là heat speader.



Hình 3.3 – Dạng chân của RIMM

2. Bộ nhớ trong

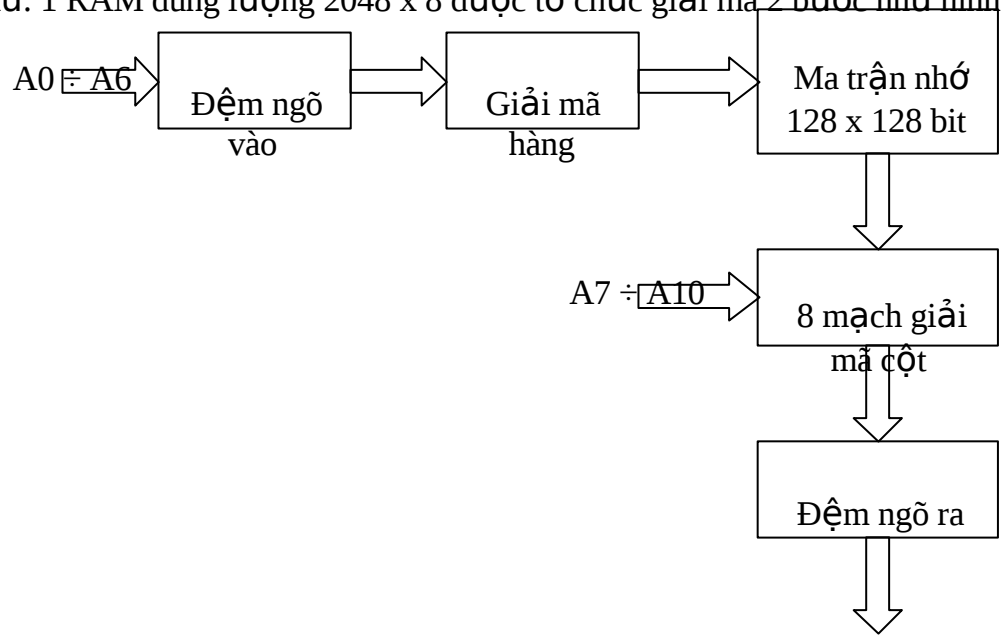
2.1. Tổ chức bộ nhớ

Bộ nhớ thường được tổ chức từ nhiều vi mạch 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.

Phương pháp giải mã hai bước cho phép giảm kích thước của phần giải mã địa chỉ bằng cách sử dụng khái niệm word logic và word vật lý. Word vật lý bao gồm tất cả các bit trong một hàng của ma trận và word logic là số bit tương ứng được gởi ra đồng thời. Lúc này, bộ nhớ cần hai mạch giải mã: giải mã hàng để chọn word vật lý và mạch giải mã cột có các mạch dồn kênh (multiplexer) chọn một word logic từ một word vật lý. Ví dụ như: 1 RAM dung lượng 2048 x 8 được tổ chức giải mã 2 bước như hình vẽ:



Hình 3.4 – Giải mã hai bước cho bộ nhớ

Ma trận nhớ là 128 x 128 bit, có 128 = 2^7 word vật lý. Một word vật lý được chọn bởi 7 đường địa chỉ từ A0 ÷A6. Độ giải mã hàng chọn 1 hàng từ 128 hàng.

Một word vật lý được chia làm 16 nhóm 8 bit. Nhóm thứ nhất chứa bit cao nhất của 16 word logic. Nhóm thứ 2 chứa các bit tiếp theo và nhóm cuối cùng chứa các bit thấp nhất. Như vậy mạch giải mã cột gồm 8 bộ dồn kênh 1 → 16 để cung cấp 1 word logic 8 bit. Các bit địa chỉ từ A7÷A10 đều khiển mạch giải mã cột. Trong trường hợp đặc biệt khi số phần tử trong 1 word vật lý bằng số bit trong 1word vật lý thì đó là bộ nhớ tổ chức theo bit nghĩa là mỗi word logic có độ dài 1 bit.

Các mạch điệm ngõ ra đảm bảo không những mức logic mong muốn và cung cấp đủ dòng mà còn có ngõ ra cực thu hở hay ba trạng thái cho phép kết nối chung với một vài bộ nhớ khác. Mạch đệm ngõ ra được điều khiển bằng các tín hiệu chọn mạch $\overline{CS}$, cho phép bộ nhớ $\overline{CE}$, cho phép ngõ ra $\overline{OE}$.

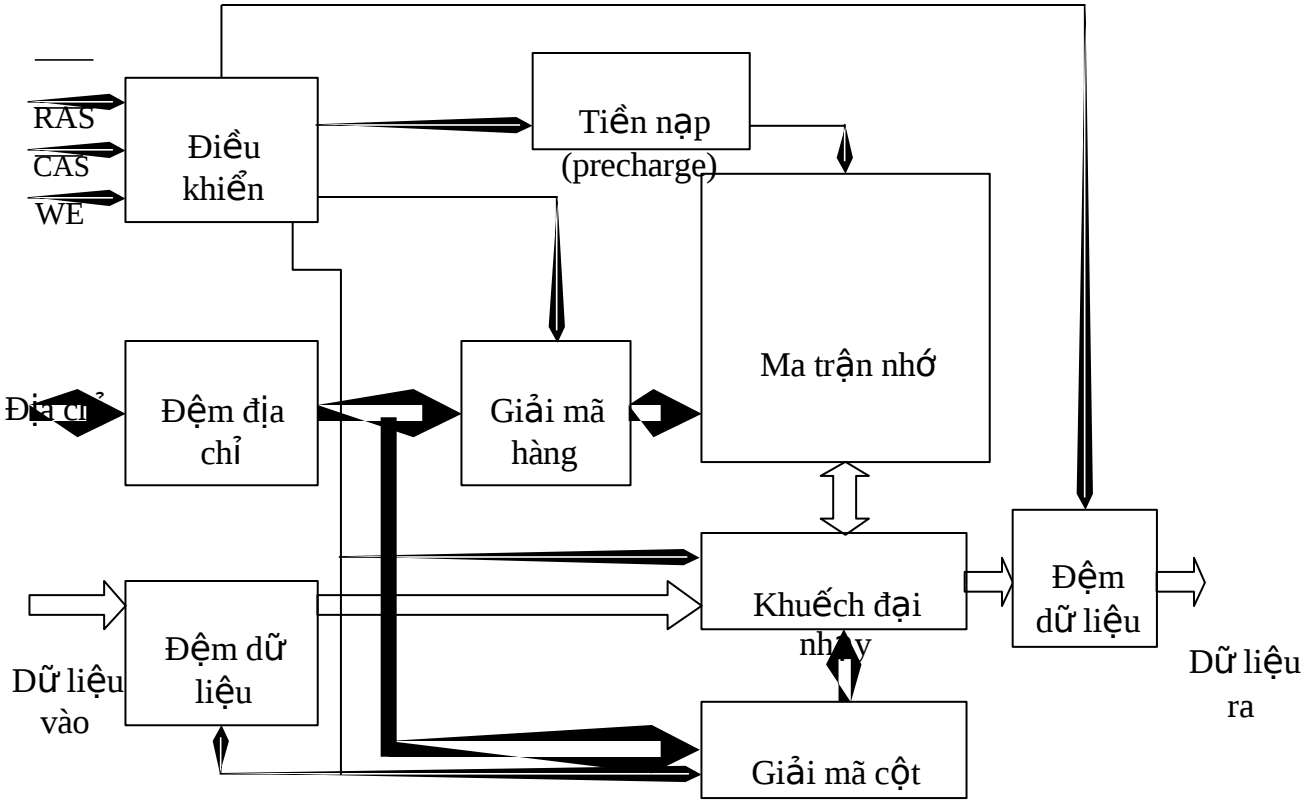
2.2. DRAM

2.2.1. 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 $\overline{RAS}$ (Row Access Strobe) và $\overline{CAS}$ (Column Access Strobe). Bộ điều khiển nhớ của CPU phải

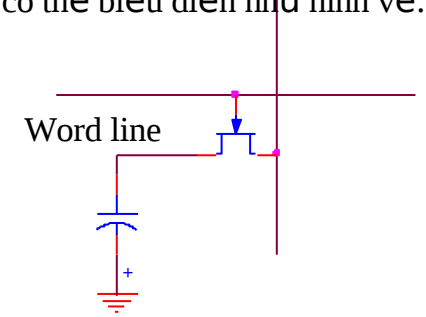
thực hiện 3 công việc sau: chia địa chỉ từ CPU thành các địa chỉ hàng và cột; tích cực các chân RAS , CAS và WE một cách chính xác; truyền và nhận các dữ liệu đọc, ghi.

Sơ đồ tổ chức của một DRAM:



Hình 3.5 – Sơ đồ cấu tạo DRAM

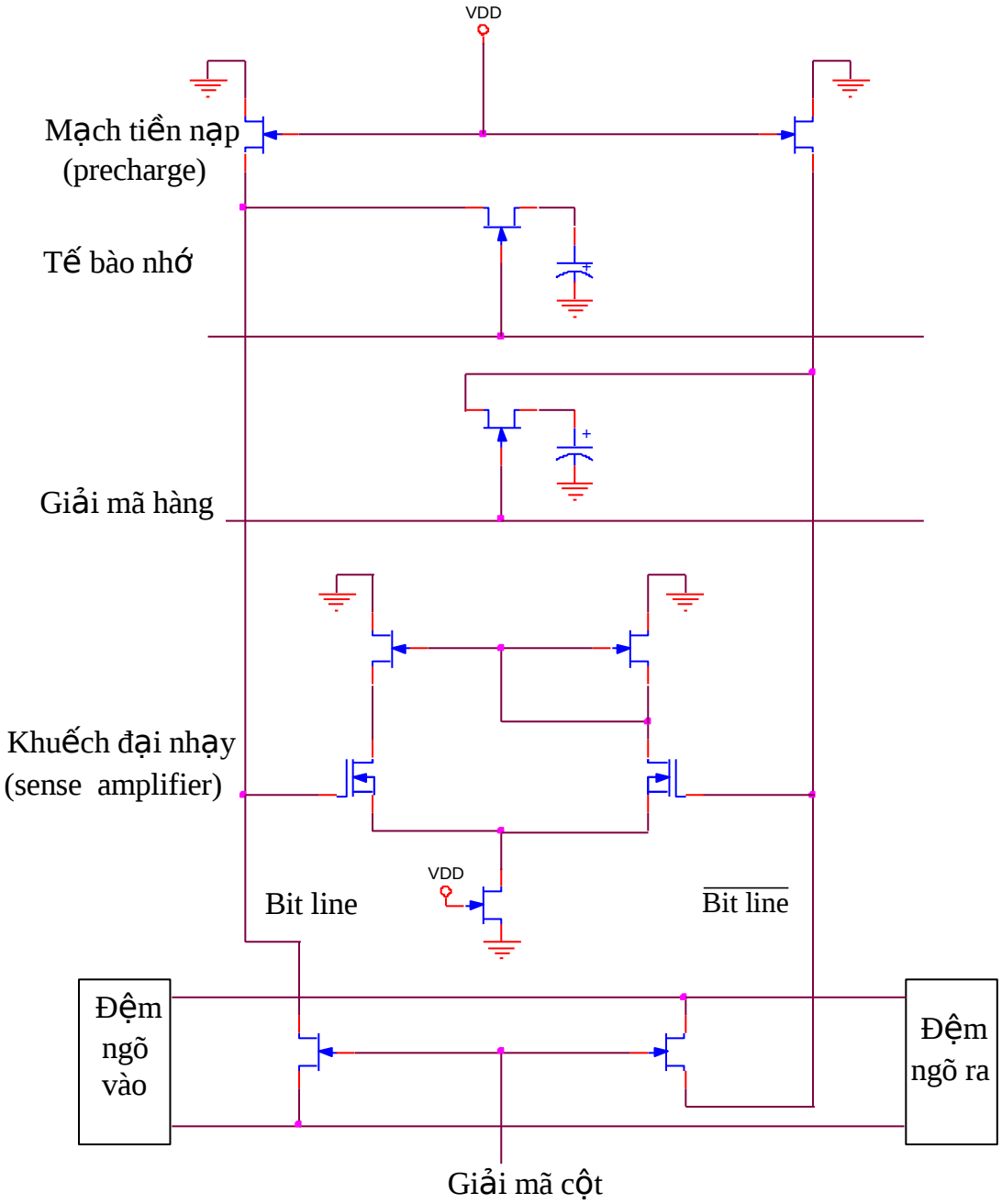
Một ô nhớ của DRAM có thể biểu diễn như hình vẽ:



Hình 3.6 – Cấu tạo một tế bào nhớ DRAM

FET hoạt động như một công tắc, khi đường word tích cực thì cho phép mở FET. Do hiện tượng rò rỉ trên FET, nên sau một thời gian, điện áp trên tụ sẽ giảm dần.

2.2.2. Quá trình đọc / ghi bộ nhớ



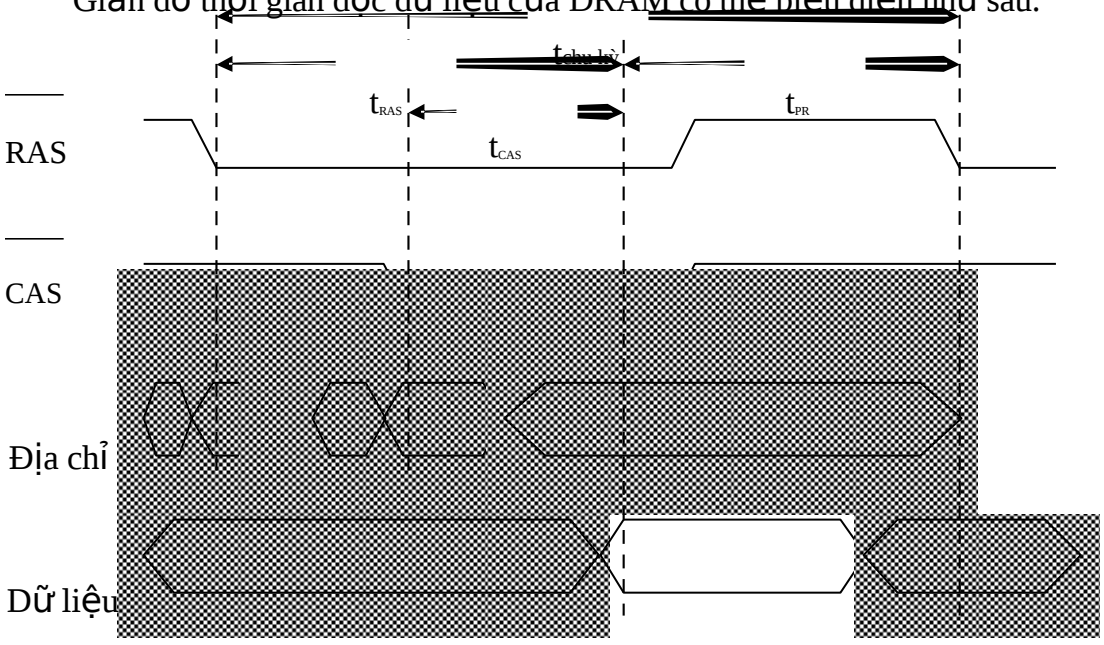
Hình 3.7 – Sơ đồ nguyên lý của DRAM

Mạch tiền nạp cho phép xâm nhập và kích hoạt word line khi nạp tất cả các bit line bằng $V_{cc}/2$. Quá trình tiền nạp sẽ dừng khi các cặp dây này có giá trị bằng nhau về điện áp. Thời gian đòi hỏi cho quá trình này gọi là thời gian tiền nạp RAS. Khi quá trình này kết thúc thì mới có thể thực hiện truy xuất ô nhớ. Mạch tiền nạp làm tăng tính ổn định của điện áp ra.

Do điện dung của tụ điện rất nhỏ so với điện dung ghép giữa các bit line nên điện áp trên các bit line thay đổi nhỏ, khoảng 100 mV. Nếu điện tích của tụ điện ban đầu bằng

0 thì điện thế trên bit line sẽ giảm xuống và kéo điện áp này xuống mức 0. Ngược lại, nếu điện tích khác 0 thì điện thế trên bit line sẽ tăng lên và nâng điện áp này lên V_{cc}. Tín hiệu giải mã cột được cấp sau tín hiệu giải mã hàng để cho phép chọn chính xác cột.

Giản đồ thời gian đọc dữ liệu của DRAM có thể biểu diễn như sau:



Hình 3.7 – Giản đồ thời gian đọc DRAM

t_{RAS}: thời gian thâm nhập RAS – là thời gian từ lúc đặt địa chỉ hàng cho tới khi dữ liệu ra khỏi bộ đệm.

t_{CAS}: thời gian thâm nhập CAS – là thời gian từ lúc đặt địa chỉ cột cho tới khi dữ liệu ra khỏi bộ đệm ($t_{CAS} < t_{RAS}$).

t_{PR}: thời gian hồi phục (thời gian tiền nạp RAS) – thời gian cần thiết từ lúc ngõ ra ổn định cho đến khi có thể cung cấp một địa chỉ mới.

2.2.3. 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.

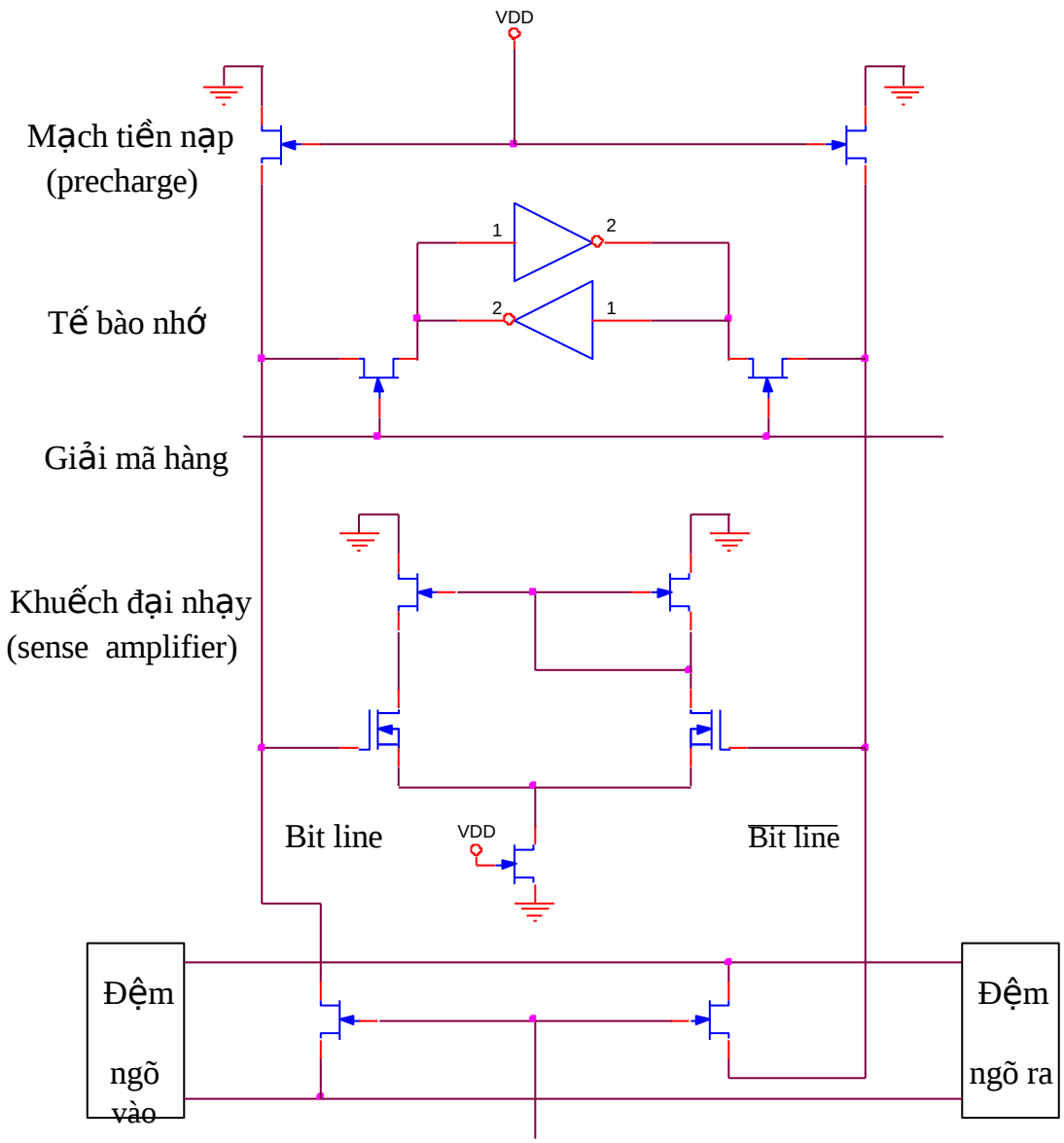
2.2.4. Các chế độ hoạt động nhanh của DRAM

DRAM có khả năng thực hiện một hay nhiều chế độ cột để giảm thời gian t_{CAS}.

- ***Chế độ trang***: quá trình truy xuất ô nhớ chỉ thay đổi địa chỉ cột. Như vậy, một trang sẽ tương ứng với một hàng trong ma trận nhớ. Để khởi động quá trình đọc, mạch điều khiển nhớ tác động tín hiệu RAS như thông thường nhưng lại bỏ qua địa chỉ hàng. Trong chế độ này, nếu ô nhớ tiếp theo trong cùng một trang thì tín hiệu RAS vẫn giữ liên tục ở mức tích cực. Do đó, thời gian truy xuất có thể giảm xuống 50%.
- ***Chế độ cột tĩnh***: tương tự như chế độ trang nhưng tín hiệu CAS giữ nguyên không đổi. DRAM sẽ tự phát hiện sự thay đổi địa chỉ sau một khoảng thời gian CAS không đổi.
- ***Chế độ nibble***: thay đổi tín hiệu CAS 4 lần để chuyển 1 nibble dữ liệu.
- ***Chế độ nối tiếp***: tương tự như chế độ nibble nhưng không phải chỉ chuyển 4 lần trạng thái của tín hiệu CAS. Về nguyên tắc, toàn bộ hàng có thể đưa ra tuần tự.
- ***Chế độ xen kẽ***: chế độ này là phương pháp hạn chế trễ do thời gian tiền nạp RAS. Bộ nhớ chia thành vài bank xen kẽ nhau theo một tỷ số xác định. Dữ liệu có địa chỉ chẵn đặt ở bank 0 và địa chỉ lẻ đặt ở bank 1. Khi đó, thời gian tiền nạp của bank 0 là thời gian truy xuất của bank 1 và ngược lại.

2.3. 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.



Hình 3.8 – Sơ đồ nguyên lý của SRAM

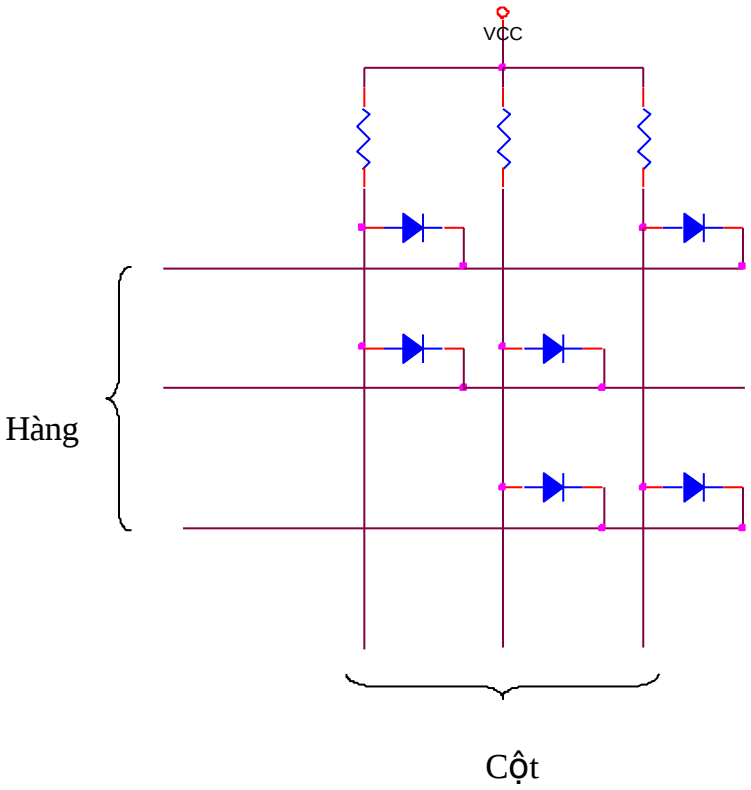
2.4. ROM

RAM không thích hợp cho các chương trình khởi động do dữ liệu trên RAM sẽ mất khi không cung cấp điện. Do đó, phải dùng đến ROM, là chip nhớ trong đó dữ liệu sẽ được lưu trữ mãi không cần duy trì nguồn cung cấp.

2.4.1. ROM

ROM được chế tạo trên một phiến 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:



Hình 3.9 – 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 V_{cc} 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.

2.4.2. PROM

Cũng tượng tự như ROM nhưng tất cả các điểm giao giữa hàng và cột đều có một diode mắc nối tiếp với một cầu chì. Khi chưa lập trình, các cầu chì vẫn còn nguyên thì nội dung trên ROM là 0. Khi lập trình, nếu cần bit nào bằng 1 thì ta đặt một xung điện ứng với bit đó, cầu chì sẽ bị đứt và xem như tại vị trí này không có diode (ứng với mức logic 1).

2.4.3. 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.

2.4.4. 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 vói 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).

2.5. Bộ nhớ Flash và bộ nhớ Cache

Bộ nhớ flash có cấu trúc giống như EEPROM nhưng lớp oxide mỏng hơn nên chỉ cần điện áp 12V là có thể xóa được dữ liệu. Bộ nhớ flash có thể hoạt động như RAM nhưng có thể lưu trữ được dữ liệu khi mất nguồn cung cấp. Thành phần chính của flash là ma trận nhớ gồm các ô nhớ FAMOST và không thực hiện phân kênh địa chỉ. Quá trình đọc dữ liệu thực hiện bằng điện áp 5V và lập trình dùng điện thế 12V. Thời gian lưu trữ của flash ít nhất là 10 năm.

Do sự ra đời của các CPU tốc độ nhanh, khi sử dụng các DRAM thì tốc độ đáp ứng của DRAM không theo kịp tốc độ của CPU nên hoạt động của hệ thống sẽ bị chậm lại làm giảm hiệu suất của máy tính (do phải thêm vào các khoảng thời gian chờ). Một giải pháp có thể thực hiện là thay bằng các SRAM có tốc độ cao hơn nhưng lại đòi hỏi giá thành cao. Bộ nhớ cache ra đời bằng cách kết hợp tốc độ cao của SRAM và giá thành rẻ của DRAM. Cache sẽ được đặt giữa CPU và bộ nhớ chính DRAM, đó là bộ nhớ SRAM có dung lượng nhỏ, bộ nhớ này sẽ lưu trữ các dữ liệu tạm thời mà CPU thường sử dụng nhằm làm giảm thời gian chờ đợi của CPU. Khi CPU cần dữ liệu, nó sẽ đọc dữ liệu trước từ cache.

3. Bộ nhớ ngoài

Bộ nhớ chính bằng vật liệu bán dẫn không thể lưu trữ một khối lượng rất lớn dữ liệu nên cần phải có thêm các thiết bị nhớ bên ngoài như băng giấy đục lỗ, băng cassette, trống từ, đĩa từ, đĩa quang, … Các thiết bị lưu trữ này còn được gọi là bộ nhớ khối (mass storage). Thiết bị nhớ khối thông dụng nhất là đĩa từ. Đĩa từ là một tấm đĩa tròn, mỏng làm bằng chất dẻo, thủy tinh cứng hay kim loại cứng, trên đó có phủ một lớp bột từ tính oxide sắt từ. Đĩa từ sử dụng kỹ thuật ghi từ để lưu trữ dữ liệu. Khi đã ghi dữ liệu trên đĩa, dữ liệu có thể tồn tại khi không còn nguồn cung cấp và cũng có khả năng xóa đi, thay thế bằng dữ liệu mới.

3.1. Đĩa mềm và ổ đĩa mềm (Floppy disk and floppy disk drive)

3.1.1. Đĩa mềm

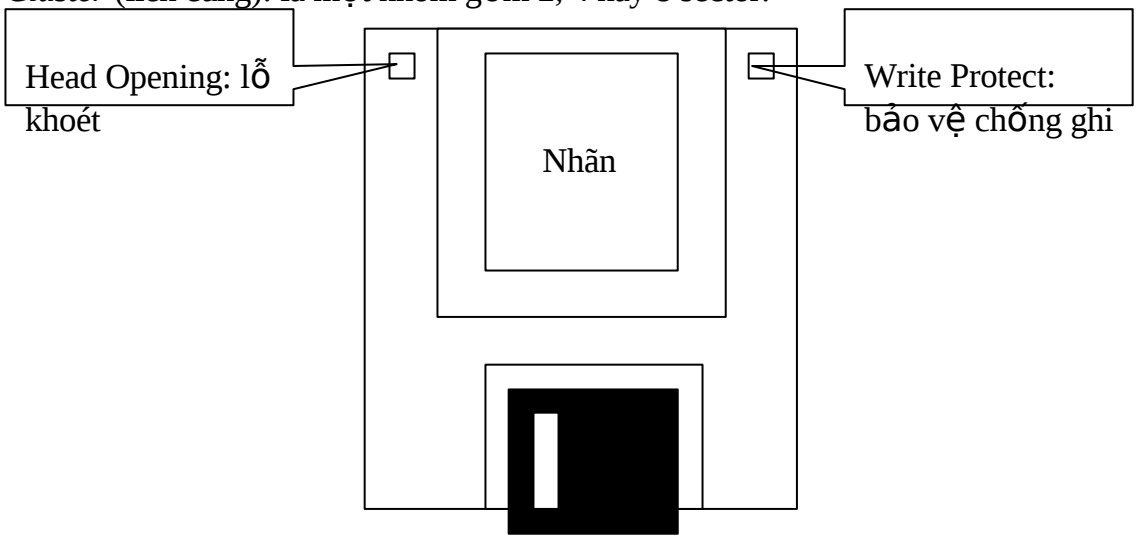
Đĩa mềm gồm một đĩa từ bằng nhựa dẻo được bảo vệ bằng một bao giấy hay nhựa cứng. Trên bao có khoét một lỗ dài cho phép đầu đọc của ổ đĩa có thể tiếp xúc với mặt đĩa để đọc / ghi dữ liệu. Có 2 loại đĩa mềm: đường kính 5.25 inch (hầu như không còn sử dụng) và đường kính 3.5 inch (chỉ dùng dung lượng 1.44 MB).

Mỗi đĩa mềm được tổ chức thành các đơn vị sau:

Track (rãnh từ): là vùng đường tròn đồng tâm lưu trữ dữ liệu. Mật độ ghi dữ liệu tính bằng đơn vị track/inch. Track được đánh số bắt đầu từ 0 kể từ vòng ngoài vào.

Sector (cung từ): mỗt track sẽ được chia thành nhiều sector, mỗi sector chứa 512 bytedữ liệu. Số sector/track tùy thuộc vào từng loại đĩa (từ 8 ÷ 36). Sector được đánh số từ 1.

Cluster (liên cung): là một nhóm gồm 2, 4 hay 8 sector.



Hình 3.10 – Đĩa mềm 3.5 inch

Dung lượng đĩa mềm = số track× số sector/track × số mặt× 512 byte

Loại đĩa	Dung lượng	Số track	Số sector/track	Tổng số sector	Track/inch
5.25 SS/SD	160KB	40	8	320	48
5.25 SS/DD	180KB	40	9	360	48
5.25 DS/DD	320KB	40	8	640	48
5.25 DS/DD	360KB	40	9	720	48
5.25 DS/HD	1.2MB	80	15	2400	48
3.5 DS/DD	720KB	80	9	1440	135
3.5 DS/HD	1.44MB	80	18	2880	270
3.5 DS/ED	2.88MB	80	36	5760	540

SS: Single Side DS: Double Side SD: Single Density DD: Double Density

HD: High Density ED: Extra High Density

Chương trình định dạng đĩa mềm (format) cho phép tạo ra các track và sector trên đĩa. Ngoài 512 byte dữ liệu, các track và sector còn chứa các byte lưu trữ thông tin dùng cho mục đích định vị và đồng bộ.

3.1.2. Ổ đĩa mềm

Ổ đĩa mềm cho phép CPU đọc / ghi dữ liệu lên đĩa mềm. Khi đó, đĩa được quay bằng một động cơ điều khiển với tốc độ 300 vòng/phút đối với đĩa 300 KB hay 360 vòng/phút đối với các loại đĩa khác. Đĩa mềm có hai mặt thì phải cần hai đầu từ đọc / ghi dữ liệu. Đầu từ được gắn ở đầu cần truy xuất (arm access). Chuyển động quay của một động cơ bước sẽ biến thành chuyển động tịnh tiến theo phương bán kính của cần truy xuất qua cơ cấu bánh răng. Đầu từ có một cuộn dây cảm ứng. Khi đọc, sự biến thiên từ thông của phần tử lưu trữ tạo thành điện thế cảm ứng ở hai đầu ra cuộn dây tạo nên tín hiệu dữ liệu. Khi ghi, cuộn dây sẽ phát ra từ trường qua khe từ để từ hóa bột oxide sắt trên mặt đĩa thành các trạng thái tương ứng với mức dữ liệu 0 và 1.

- **Mạch điều khiển ổ đĩa mềm:**

Mạch điều khiển ổ đĩa mềm thường được cắm trên một khe cắm mở rộng. Bộ điều khiển có một vi xử lý riêng với chương trình trong ROM của nó, thông thường là NEC µPD765 hay Intel 8207A. Việc ghi dữ liệu được thực hiện qua các bước sau:

- Dữ liệu truyền từ bus vào bộ giao tiếp bus.
- Bộ điều khiển xác định byte CRC (Cycle Redundancy Check), đổi dữ liệu song song thành nối tiếp và định dạng thích hợp
- Bộ tách dữ liệu đổi chuỗi dữ liệu thành chuỗi mã FM hay MFM (Modified FM) và phát ra các xung đánh dấu.
- Mạch giao tiếp SA-450 truyền chuỗi dữ liệu đã mã hóa tới ổ đĩa.

- Đầu từ ghi số liệu đã mã hóa lên đĩa.

Địa chỉ các thanh ghi của mạch điều khiển ổ đĩa mềm:

	Sơ cấp	Thứ cấp	R/W
Địa chỉ cơ sở	3F0h	370h	
Thanh ghi trạng thái A	3F1h	371h	R
Thanh ghi trạng thái B	3F1h	371h	R
Thanh ghi ngõ ra số DOR	3F2h	372h	R/W
Thanh ghi trạng thái chính	3F4h	374h	R
Thanh ghi chọn tốc độ truyền dữ liệu	3F4h	374h	W
Thanh ghi dữ liệu	3F5h	375h	R/W
Thanh ghi ngõ vào số	3F7h	377h	R
Thanh ghi điều khiển cấu hình	3F7h	377h	W
Kênh DMA	2	2	
Yêu cầu ngắt IRQ	6	6	
Ngắt INTR	0Eh	0Eh	

- ***Thanh ghi ngõ ra số (DOR - Digital Output Register)***: điều khiển động cơ, chọn ổ đĩa và khởi tạo bộ điều khiển

D7	D6	D5	D4	$\overline{D3}$	$\overline{D2}$	D1	D0

MOTD MOTC MOTB MOTA DMA $\overline{RESET}$ DR1 DR0

$\overline{MOTD}$, MOTC, MOTB, MOTA: điều khiển động cơ (motor) cho các ổ đĩa

DMA : cho phép (=1) hay cấm (=0) kênh DMA và IRQ

$\overline{RESET}$: cho phép (=1) hay cấm (=0) reset bộ điều khiển

DR1, DR0: chọn ổ đĩa 00 (A), 01 (B), 10 (C), 11 (D)

- ***Thanh ghi trạng thái chính (main status register)***: là thanh ghi chỉ đọc, chứa thông tin về bộ điều khiển đĩa mềm

D7	D6	D5	D4	D3	D2	D1	D0
MRQ	DIO	NDMA	BUSY	ACTD	ACTC	ACTB	ACTA

RQ: sẵn sàng (=1) hay không sẵn sàng (=0) ghi dữ liệu.

DIO: chiều truyền từ bộ điều khiển tới CPU (=1) hay ngược lại (=0).

NMDA: có chế độ DMA (=0) hay không (=1).

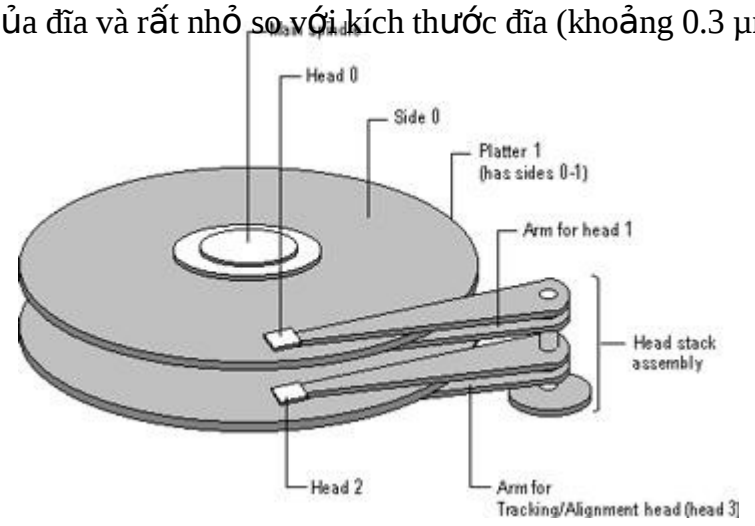
BUSY: kích hoạt về lệnh (=1) hay không (=0).

ACTA, ACTB, ACTC, ACTD: chọn ổ đĩa tương ứng (=1).

3.2. Đĩa cứng và ổ đĩa cứng

3.2.1. Cấu tạo

Đĩa cứng gồm một hay nhiều đĩa từ bằng kim loại hay nhựa cứng được xếp thành một chồng theo một trục đứng và được đặt trong một hộp kín. Dung lượng đĩa cứng lớn hơn nhiều so với đĩa mềm. Ổ đĩa cứng có nhiều đầu từ, các đầu từ này gắn trên một cần truy xuất và di chuyển thành một khối. Khi đĩa quay, đầu từ không chạm vào mặt đĩa mà cách một lớp đệm không khí. Khoảng cách giữa mặt đĩa và đầu từ tùy theo tốc độ quay và mật độ ghi dữ liệu của đĩa và rất nhỏ so với kích thước đĩa (khoảng 0.3 µm).



Hình 3.11 – Cấu tạo đĩa cứng

Đĩa cứng cũng được phân thành các đơn vị vật lý như đĩa mềm. Ngoài ra, nó còn một khái niệm nữa là *cylinder*. Cylinder là vị trí của đầu từ khi di chuyển trên các mặt tạo thành một hình trụ, đó là một chồng các track xếp nằm lên nhau đối với một vị trí đầu từ.

Dung lượng đĩa cứng = số head× số cylinder× số sector/track × số mặt× 512 byte

Tốc độ quay của đĩa cứng là 3600 vòng/phút nên thời gian truy xuất của đĩa cứng nhanh hơn đĩa mềm nhiều. *Thời gian truy xuất dữ liệu* (data access time) là một thông số quan trọng của đĩa cứng, bao gồm *thời gian tìm kiếm* (seek time), *thời gian chuyển đầu từ* (head switch time) và *thời gian quay trễ* (rotational latency). Thời gian tìm kiếm là thời gian chuyển đầu từ từ một track này sang track khác. Thời gian chuyển đầu từ là thời gian chuyển giữa hai trong số các đầu từ khi đọc hay ghi dữ liệu. Thời gian quay trễ là thời gian tính từ khi đầu từ được đặt trên một track cho đến khi tới được sector mong muốn.

3.2.2. Định dạng cấp thấp (low – level format)

Đĩa cứng phải được định dạng cấp thấp trước khi sử dụng. Đó là tạo ra các track và sector trên đĩa bằng cách ghi lên đĩa các thông tin liên quan đến chúng. Với các sector, thông tin này được ghi vào vùng *tiêu đề nhận dạng sector*, được đặt ở đầu mỗi sector. Vùng tiêu đề này chứa các thông tin như số thứ tự đầu từ, số sector, số cylinder, khai báo nhận dạng ID và mã CRC để phát hiện lỗi dữ liệu.

Trong đĩa cứng có thêm một khái niệm là ***hệ số xen kẽ*** (interleave factor) của các sector nhằm làm khớp tốc độ quay của đĩa từ với tốc độ mà đầu từ có thể xử lý dữ liệu khi chúng qua hết một sector. Ví dụ như với đĩa có 17 sector/track, đầu từ sẽ đọc được 512 x 16 x 60 = 522,240 byte/s. Do đó, vùng đệm sẽ đầy lên rất nhanh. Đồng thời, CPU cần phải xử lý dữ liệu nên sẽ mất thêm một khoảng thời gian nữa. Như vậy, nếu dữ liệu được ghi lên các sector liên tiếp thì CPU không thể xử lý kịp sector kế tiếp sẽ đặt cách đó n sector. Khi đó, đĩa cứng sẽ có hệ số xen kẽ n các sector dữ liệu không liên tiếp nhau về mặt vật lý. Chương trình định dạng cấp thấp sẽ đánh số thứ tự các sector liên tiếp theo một trật tự định trước phụ thuộc vào hệ số xen kẽ. Nếu đĩa cứng không được định dạng với hệ số xen kẽ tối ưu thì hiệu suất sử dụng sẽ thấp.

Sector xấu (bad sector): trong quá trình định dạng cấp thấp, các sector xấu không thể lưu trữ dữ liệu sẽ được đánh dấu để không dùng nữa. Quá trình này gọi là *định bản đồ các sector xấu*.

3.2.3. Bộ điều khiển và giao tiếp đĩa cứng

Khác với bộ điều khiển đĩa mềm chỉ dùng các byte CRC, bộ điều khiển đĩa cứng còn dùng thêm các byte ECC (Error Correcting Code) cho phép sửa lỗi và dùng mạch giao tiếp ST412/506.

3.2.3.1. Chuẩn giao tiếp IDE (Integrated Drive Electronics)

Với cách giao tiếp thông thường, ổ đĩa chỉ chứa các linh kiện điện tử tối thiểu đòi hỏi cho việc điều khiển động cơ và các cổng logic còn quá trình điều khiển lệnh (đọc sector, đọc tín hiệu mã hóa, …) được thực hiện trên mạch điều khiển đĩa cứng. Như vậy, các dữ liệu được mã hóa phải đi từ ổ đĩa qua cable truyền, tới bộ điều khiển để giải mã có thể làm sai lạc dữ liệu. Giao tiếp IDE giải quyết vấn đề này bằng cách tích hợp cả ổ đĩa và bộ điều khiển vào cùng một khối. Việc kết nối giữa bus và mạch giao tiếp IDE được thực hiện bằng mạch host-adapter, mạch này cung cấp một số bộ đệm và giải mã dùng cho kết nối ổ đĩa. Giao tiếp IDE cho phép phục vụ nhiều nhất là hai ổ đĩa (master gán địa chỉ 0 và slave gán địa chỉ 1). Một số ổ đĩa IDE trang bị thêm bộ nhớ cache cho ít nhất 2 track nhằm giảm thời gian truy xuất trung bình của đĩa.

CPU truy xuất bộ điều khiển IDE qua một số thanh ghi dữ liệu và điều khiển. Chúng được phân thành hai nhóm với địa chỉ cơ sở 1F0h và 3F0h.

Thanh ghi	**Địa chỉ**	**Độ rộng thanh ghi**	**R/W**
Thanh ghi dữ liệu	1F0h	16	R/W

Thanh ghi lỗi	1F1h	8	R
Bù trước	1F1h	8	W
Số các sector	1F2h	8	R/W
Số sector	1F3h	8	R/W
Cylinder LSB	1F4h	8	R/W
Cylinder MSB	1F5h	8	R/W
Ổ đĩa / đầu từ	1F6h	8	R/W
Thanh ghi trạng thái	1F7h	8	R
Thanh ghi lệnh	1F7h	8	W
Thanh ghi trạng thái biến đổi	3F6h	8	R
Thanh ghi ngõ ra số	3F6h	8	W
Địa chỉ ổ đĩa	3F7h	8	R

Chương trình giao tiếp CPU với IDE gồm 3 bước:

- Giai đoạn lệnh: CPU chuẩn bị các thanh ghi và chuyển mã lệnh để khởi động việc thực thi lệnh.
- Giai đoạn dữ liệu: ổ đĩa định vị đầu từ và truyền dữ liệu với bộ nhớ chính.
- Giai đoạn kết quả: bộ điều khiển cung cấp thông tin trạng thái cho lệnh đã chạy trong các thanh ghi tương ứng và phát một ngắt tại IRQ14 (INT 76h).

3.2.3.2. Chuẩn giao tiếp SCSI (Small Computer System Interface)

SCSI là một chuẩn giao tiếp giữa ổ đĩa và PC rất mềm dẻo và mạnh. Chuẩn này xây dựng một bus giao tiếp gồm 8 đơn vị SCSI trong đó một host-adapter nối 7 đơn vị còn lại với bus hệ thống. Khác với IDE, host-adapter này phức tạp hơn vì nó phải thực hiện các chức năng của bus SCSI và không bị ràng buộc bởi các hạn chế của bus hệ thống. Bus SCSI chỉ phục vụ cho quá trình trao đổi dữ liệu giữa các đơn vị được nối với nó. Trong một thời điểm chỉ có hai đơn vị có thể hoạt động. Việc trao đổi dữ liệu có thể thực hiện giữa host-adapter và ổ đĩa hay giữa hai đơn vị SCSI mà không có sự tham gia của CPU.

3.3. Tổ chức logic của đĩa mềm và đĩa cứng

3.3.1. Sector logic

BIOS dùng các sector vật lý để quản lý dữ liệu trong khi hệ điều hành dùng một sơ đồ khác gọi là sector logic. Đó là đánh số các sector liên tục từ 0. Ví dụ như một đĩa mềm 3 ¼ 2 mặt với 80 track và 18 sector/track, dung lượng 1.44 MB như sau:

Head 0, track 0, sector 1	0
……	
Head 0, track 0, sector 18	17
Head 1, track 0, sector 1	18
……	
Head 1, track 0, sector 18	35
Head 0, track 1, sector 1	36
……	
Head 0, track 1, sector 18	53
Head 1, track 79, sector 1	2862
……	
Head 1, track 79, sector 18	2879

3.3.2. Phân vùng (Partition)

Một đĩa cứng có thể chia thành nhiều ổ đĩa logic và được xem như những ổ đĩa vật lý riêng biệt. Về mặt logic, một phân vùng được xem như một đĩa cứng và có thể cài đặt một hệ điều hành tùy ý lên đó. Có 3 loại phân vùng trên đĩa cứng: DOS chính (Primary DOS), DOS mở rộng (Extended DOS) và phi DOS (non-DOS).

Để lưu trữ thông tin về các phân vùng, DOS lưu trữ trong một vùng cố định: head 0, track 0, sector 1 (sector vật lý đầu tiên của đĩa cứng), sector này được gọi là sector phân vùng (partition sector). Thông tin về từng phân vùng được lưu trữ bởi các điểm vào phân vùng (partition entries) trong bảng phân vùng (partition table).

- **Cấu trúc của sector phân vùng**:

MBR (chương trình kiểm tra bảng phân vùng và gọi boot sector): 446 byte
Bảng phân vùng: 64 byte
Nhận dạng (thường là AA55h): 2 byte

Một ví dụ của sector phân vùng như sau:

- MBR chiếm 664 byte đầu tiên của sector và kết thúc bằng ký hiệu nhận dạng đĩa (in đậm: FD 4E F2 14).
- Phần còn lại là bảng phân vùng và nhận dạng (in nghiêng)

```
Physical Sector:Cyl 0,Side 0,Sector 1

  00000000:00 33 C0 8E D0 BC 00 7C -8B F4 50 07 50 1F FB FC .3.....|..P.P..
  00000010:BF 00 06 B9 00 01 F2 A5 -EA 1D 06 00 00 BE BE 07 ................
  00000020:B3 04 80 3C 80 74 0E 80 -3C 00 75 1C 83 C6 10 FE ...<.t..<.u.....
```

```
00000030:CB 75 EF CD 18 8B 14 8B -4C 02 8B EE 83 C6 10 FE .u......L.......
00000040:CB 74 1A 80 3C 00 74 F4 -BE 8B 06 AC 3C 00 74 0B .t..<.t.....<.t.
00000050:56 BB 07 00 B4 0E CD 10 -5E EB F0 EB FE BF 05 00 V.......^.......
00000060:BB 00 7C B8 01 02 57 CD -13 5F 73 0C 33 C0 CD 13 ..|...W.._s.3...
00000070:4F 75 ED BE A3 06 EB D3 -BE C2 06 BF FE 7D 81 3D Ou..........}.=
00000080:55 AA 75 C7 8B F5 EA 00 -7C 00 00 49 6E 76 61 6C U.u.....|..Inval
00000090:69 64 20 70 61 72 74 69 -74 69 6F 6E 20 74 61 62 id partition tab
000000A0:6C 65 00 45 72 72 6F 72 -20 6C 6F 61 64 69 6E 67 le.Error loading
000000B0:20 6F 70 65 72 61 74 69 -6E 67 20 73 79 73 74 65 operating syste
000000C0:6D 00 4D 69 73 73 69 6E -67 20 6F 70 65 72 61 74 m.Missing operat
000000D0:69 6E 67 20 73 79 73 74 -65 6D 00 00 80 45 14 15 ing system...E..
000000E0:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
000000F0:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000100:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000110:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000120:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000130:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000140:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000150:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000160:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000170:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000180:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
00000190:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
000001A0:00 00 00 00 00 00 00 00 -00 00 00 00 00 00 00 00 ................
000001B0:00 00 00 00 00 00 00 00 -FD 4E F2 14 00 00      .........N......
                                                 80 01              ..
000001C0:01 00 06 0F 7F 96 3F 00 -00 00 51 42 06 00 00 00 .....?...QB....
000001D0:41 97 07 0F FF 2C 90 42 -06 00 A0 3E 06 00 00 00 A....,.B...>....
000001E0:C1 2D 05 0F FF 92 30 81 -0C 00 A0 91 01 00 00 00 .-....0.........
000001F0:C1 93 01 0F FF A6 D0 12 -0E 00 C0 4E 00 00 55 AA ...........N..U.
```

- **Cấu trúc của bảng phân vùng:**

Bảng phân vùng có 64 byte chia thành 4 điểm vào, mỗi điểm vào có vị trí bắt đầu tại byte thứ 446 (phân vùng 1), 462 (phân vùng 2), 478 (phân vùng 3) và 494 (phân vùng 4)

Byte offset	Kích thước	Ý nghĩa
00	1 byte	Phân vùng có khởi động được (=80h) hay không (=00h)
01	1 byte	Head bắt đầu
02	6 bit	Sector bắt đầu (bit 0-5, bit 6-7 dùng cho cylinder)
03	10 bit	Cylinder bắt đầu
04	1 byte	Loại phân vùng
05	1 byte	Head kết thúc
06	6 bit	Sector kết thúc
07	10 bit	Cylinder kết thúc
08	4 byte	Sector tương đối
12	4 byte	Tổng số sector

Loại phân vùng:

Giá trị	Ý nghĩa
01h	FAT-12 hay ổ đĩa logic với số sector < 32680
04h	FAT-16hay ổ đĩa logic với số sector ≥ 2680 và ≤ 65535
05h	Phân vùng mở rộng
06h	BIGDOS FAT (> 32MB)
07h	NTFS chính
0Bh	FAT32 chính, dùng INT 13 mở rộng
0Ch	FAT32 mở rộng, dùng INT 13 mở rộng
0Eh	FAT16 mở rộng, dùng INT 13 mở rộng
0Fh	FAT16 chính, dùng INT 13 mở rộng

4 loại phân vùng sau cho phép dùng hệ thống file FAT mà Windows NT không thể nhận dạng được.

Sector tương đối: là độ dời của sector bắt đầu của phân vùng với sector bắt đầu của đĩa.

```
                                                    80 01                 . .
000001C0:01 00 06 0F 7F 96 3F 00 -00 00 51 42 06 00 00 00 .....?...QB....
000001D0:41 97 07 0F FF 2C 90 42 -06 00 A0 3E 06 00 00 00 A....,.B...>....
000001E0:C1 2D 05 0F FF 92 30 81 -0C 00 A0 91 01 00 00 00 .-....0.........
000001F0:C1 93 01 0F FF A6 D0 12 -0E 00 C0 4E 00 00 55 AA ...........N..U.
```

Xét phân vùng đầu tiên: 80 01 01 00 06 0F 7F 96 3F 00 00 00 51 42 06 00

Byte 0 = 80h: phân vùng khởi động được

Byte 1 = 01h: head 1

Byte 2 = 01h = 0000 0001b: sector 1 (chỉ dùng 6 bit 000001b)

Byte 3 = 00h = 0000 0000b: cylinder 0 (kết hợp với 2 bit cao của byte 2)

 bắt đầu tại Head 1, Sector 1, Cylinder 0

Byte 4 = 06: phân vùng BIGDOS

Byte 5 = 0Fh:head 15

Byte 6 = 7Fh = 0111 1111b: sector 63 (dùng 6 bit thấp 111111b)

Byte 7 = 96h = 1001 0110b: cylinder 406 (= 01 1001 0110b)

 kết thúc tại Head 15, Sector 63, Cylinder 406

Byte 8-11= 0000003Fh: độ dời so với sector đầu tiên là 63 sector.

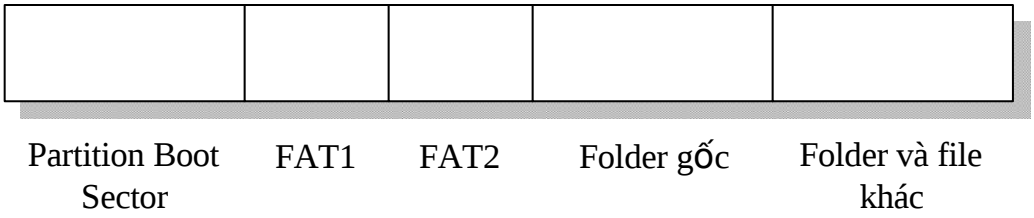
Byte 9-12 = 00064251: tổng cộng có 410.193 sector trong phân vùng

3.3.3. Hệ thống file FAT (File Allocation Table)

Hệ thống file FAT là hệ thống file đơn giản thiết kế cho đĩa dung lượng nhỏ và cấu trúc folder đơn giản. Đối với hệ thống file này, hệ điều hành lưu trữ file lên đĩa cứng theo đơn vị cluster. Mỗi cluster gồm một hay nhiều sector. Để theo dõi những cluster này, hệ điều hành sử dụng một cấu trúc là bảng định vị file (FAT). Để bảo vệ đĩa, một bản sao của FAT sẽ được lưu trữ thêm để tránh trường hợp FAT hỏng, đồng thời FAT và folder gốc sẽ được lưu trữ tại một vị trí cố định để có thể lấy được các file cần thiết khi khởi động hệ thống.

Sự khác nhau giữa các hệ thống FAT:

Hệ thống file	Số byte/cluster	Số cluster tối đa
FAT12	1.5	< 4087
FAT16	2	≤ 65,526 và ≥ 4087
FAT32	4	≤ 268,435,456 và ≥ 65,526



Hình 3.12 – Cấu trúc của FAT

3.3.3.1. Sector khởi động phân vùng PBS

Sector khởi động dài 512 byte đặt tại sector logic 0 của ổ đĩa logic. Nó chứa một chương trình cho phép nạp nhân (kernel) của hệ điều hành để khởi động hệ thống.

Cấu trúc của PBS:

Byte Offset	Kích thước [byte]	Ý nghĩa
00h	3	Lệnh nhảy
03h	8	Tên nhà sản xuất
0Bh	25	Khối thông số BIOS (BPB)
24h	26	
3Eh	448	Khối thông số BIOS mở rộng
1FEh	2	Mã khởi động

(bootstrap code)

Đánh dấu kết thúc sector

Cấu trúc của BPB (BIOS Parameter Block) và BPB mở rộng (Extended BPB):

Byte Offset	Kích thước [byte]	Ý nghĩa
0Bh	2	Số byte/sector (thường là 512)
0Dh	1	Số sector/cluster
0Eh	2	Số sector dành cho boot sector
10h	1	Số lượng FAT (thường là 2)
11h	2	Số điểm vào gốc (root entry)
13h	2	Số sector/đĩa (nếu < 16 bit). Ngược lại thì = 0
15h	1	Môi trường lưu trữ, F8h để chỉ đĩa cứng
16h	2	Số sector/FAT
18h	2	Số sector/track
1Ah	2	Số head
1Ch	4	Số sector ẩn (giống như sector tương đối)
20h	4	Số sector/đĩa (nếu > 16 bit). Ngược lài thì = 0
24h	1	Số thứ tự của đĩa vật lý (thường là 80h)
25h	1	Head hiện hành, không dùng cho FAT
26h	1	Nhận dạng (=28h hay 29h để WinNT có thể nhận ra)
27h	4	Số serial của đĩa
2Bh	11	Nhãn đĩa
36h	8	Nhận dạng hệ thống file

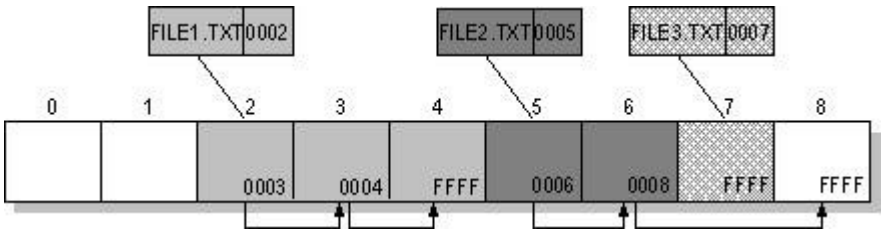
3.3.3.2. Bảng định vị file FAT

FAT là một bảng gồm một số điểm vào ứng với số cluster trên đĩa. Số cluster bằng với tổng số sector trên đĩa chia cho số sector trên một cluster. Kích thước của mỗi điểm vào tùy thuộc vào hệ điều hành, có thể là 12, 16 hay 32 bit (ứng với FAT12, FAT16 và FAT32). Hai điểm vào đầu tiên dùng để nhận dạng loại đĩa (đĩa cứng, đĩa mềm, bao nhiêu mặt, bao nhiêu sector/track, …). Nội dung của một điểm vào:

- 0000h: cluster trống
- FFF7h: cluster xấu
- FFF8h – FFFFh: cluster cuối cùng của file
- Cluster đang sử dụng bởi file

Mỗi cluster chứa một con trỏ chỉ tới cluster kế tiếp trong file hay giá trị FFFFh để xác định cluster kết thúc file.

File1.txt chứa trong 3 cluster liên tục, file2.txt cũng chứa trong 3 cluster nhưng bị phân mảnh còn file3.txt chỉ chứa trong 1 cluster.



Hình 3.13 – Ví dụ của FAT

Một ví dụ của bảng FAT16 như sau (16 byte):

```
F8 FF FF FF 03 00 00 04 00 05 00 FF FF B1 05 01 A9
```

Byte 1,2 (F8 FF): nhận dạng đĩa

Byte 3,4 (FF FF): cluster kết thúc của một file nào đó.

Btye 5,6 (03 00): chứa giá trị 0003h cluster kế tiếp là cluster 3.

Byte 7,8 (04 00): chứa giá trị 0004h cluster kế tiếp là cluster 4.

Byte 9,10 (05 00): chứa giá trị 0005h cluster kế tiếp là cluster 5.

Byte 11,12 (FF FF): cluster kết thúc file file chứa trong 4cluster.

Các byte tiếp theo: dùng cho các file khác.

3.3.3.3. Folder gốc

Folder gốc chứa các điểm vào cho các file và folder nằm trên gốc của ổ đĩa. Folder gốc khác với các folder khác là nó có vị trí và kích thước cố định (512 điểm vào đối với đĩa cứng). Một điểm vào folder gốc có nội dung như sau:

Byte offset	Kích thước [byte]	Ý nghĩa
00h	8	Tên file
08h	3	Phần mở rộng
0Bh	1	Thuộc tính file
0Ch	10	Dự trữ
16h	2	Giờ thay đổi thông tin cuối cùng
18h	2	Ngày thay đổi thông tin cuối cùng
1Ah	2	Cluster đầu tiên của file
1Ch	4	Kích thước file

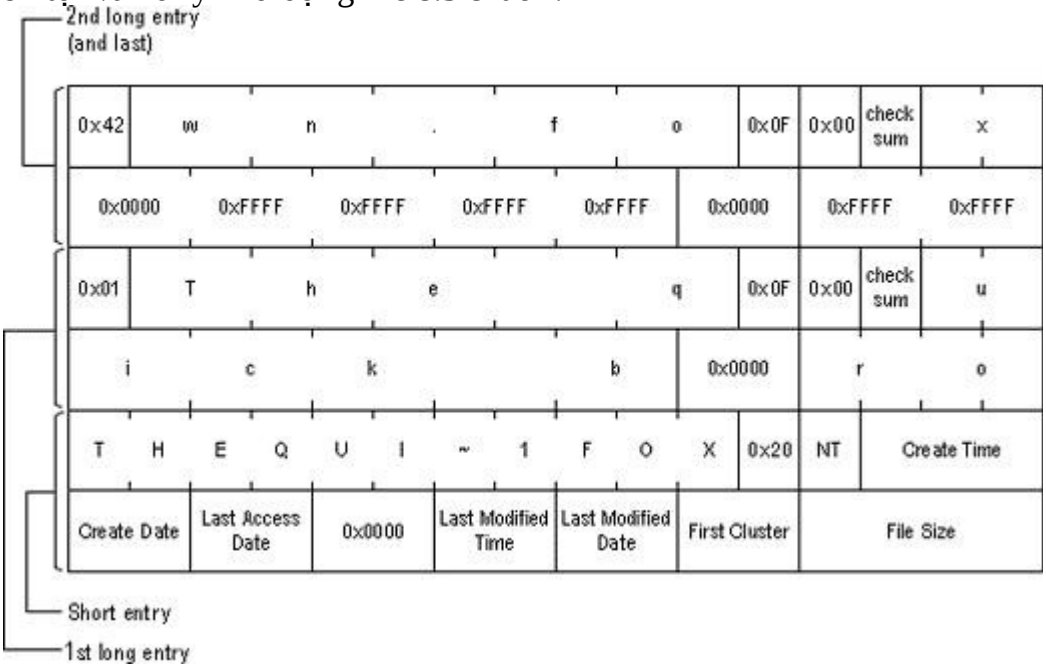
Folder có một tập các điểm vào folder 32 byte (folder entry) cho mỗi file và folder con chứa trong nó. Điểm vào folder bao gồm:

- Tên: 8.3 ký tự

- Byte thuộc tính: 1 byte bao gồm thuộc tính lưu trữ (archive), ẩn (hidden), hệ thống (system) và chỉ đọc (read-only). User có thể bật hay tắt các thuộc tính này.
- Thời gian tạo: 3 byte
- Ngày tạo: 2 byte
- Ngày truy xuất cuối cùng: 2 byte
- Thời gian thay đổi thông tin cuối cùng: 2 byte
- Ngày thay đổi thông tin cuối cùng: 2 byte
- Số thứ tự của cluster bắt đầu file: 2 byte
- Kích thước: 2 byte

Tên file:

Bắt đầu từ WinNT 3.5, file được tạo trên đĩa FAT dùng các bit thuộc tính để hỗ trợ tên file dài mà không ảnh hưởng đến các hệ điều hành trước (DOS). Khi tạo một file, nếu tên file dài thì Windows sẽ tạo một tên dạng 8.3 cho file và sẽ thêm các điểm vào thứ cấp của file, mỗi điểm vào chứa 13 ký tự. DOS sẽ bỏ qua các điểm vào này, xem như chúng không tồn tại và xử lý như dạng file 8.3 chuẩn.



Hình 3.14 – Ví dụ về tên file dài

3.3.4. FAT32

FAT32 là một dạng mở rộng của hệ thống file FAT để hỗ trợ lưu trữ lớn hơn 2 GB. Ổ đĩa FAT32 có thể chứa nhiều hơn 65256 cluster và mỗi cluster nhỏ hơn so với FAT16 nên hiệu suất sử dụng sẽ cao hơn. File lớn nhất có thể lưu trữ trên ổ đĩa FAT32 là 4GB-2.

- **Thay đổi trong boot sector và bootstrap:**

Thay đổi trong boot sector:

Thay đổi	Mô tả
Các sector dành riêng	FAT32 chứa nhiều sector dành riêng hơn FAT16 và FAT12, thường là 32
Thay đổi của boot sector	Do FAT32 BPB lớn hơn BPB chuẩn, MBR trên ổ đĩa FAT32 lớn hơn 1 sector đồng thời 1 sector trong khu vực dành riêng chứa số lượng cluster chưa sử dụng và số lượng cluster mới sử dụng gần nhất. Các giá trị này là thành phần của cấu trúc BIGFATBOOTFSINFO cho phép hệ thống khởi tạo các giá trị mà không cần đọc toàn bộ FAT.
Folder gốc	Folder gốc trên ổ đĩa FAT32 không chứa tại một vị trí cố định như FAT16 hay FAT12. Folder gốc là một chuỗi cluster thông thường. Thành phần *A_BF_BPB_RootDirStrtClus* trong cấu trúc BPB chứa số thứ tự của cluster đầu tiên của folder gốc và *BPB_RootEntries* sẽ bị bỏ qua.
Số sector/FAT	Thành phần *A_BF_BPB_SectorsPerFAT* của BPB luôn bằng 0 trên ổ đĩa FAT32 mà thay vào đó là 2 thành phần *A_BF_BPB_BigSectorsPerFat* và *A_BF_BPB_BigSectorsPerFatHi*.

- **BPB (BIOS Parameter Block):**

BPB trong FAT32 là phiên bản mở rộng của BPB FAT16/FAT12. Nó cũng cung cấp thông tin như trước nhưng đồng thời thêm vào một số trường:

Tên	Kích thước [byte]	Mô tả
A_BF_BPB_BytesPerSector	2	Số byte/sector
A_BF_BPB_SectorsPerCluster	1	Số sector/cluster
A_BF_BPB_ReservedSectors	2	
A_BF_BPB_NumberOfFATs	1	
A_BF_BPB_RootEntries	2	
A_BF_BPB_TotalSectors	2	
A_BF_BPB_MediaDescriptor	1	
A_BF_BPB_SectorsPerFAT	2	
A_BF_BPB_SectorsPerTrack	2	
A_BF_BPB_Heads	2	
A_BF_BPB_HiddenSectors	2	
A_BF_BPB_HiddenSectorsHigh	2	

Số thứ tự của sector dành riêng, bắt đầu bằng sector 0

Số lượng FAT

Bỏ qua

Tổng số sector

Tương tự như FAT16

= 0 head trên đĩa

Số Số sector ẩn trên đĩa

sector/t Số sector ẩn trên đĩa (2 byte cao)

rack

Số

A_BF_BPB_BigTotalSectors	2	Tổng số sector trên FAT32
A_BF_BPB_BigTotalSectorsHigh	2	Tổng số sector trên FAT32 (2 byte cao)
A_BF_BPB_BigSectorsPerFat	2	Số sector/FAT
A_BF_BPB_BigSectorsPerFatHi	2	Số sector/FAT (2 byte cao)
A_BF_BPBExtFlags	2	Bit 7 xác định thông tin trên FAT hiện hành có cập nhật có tất cả các FAT khác hay không.
A_BF_BPB_FS_Version	2	Phiên bản của hệ thống file
A_BF_BPB_RootDirStrtClus	2	Số thứ tự cùa cluster đầu tiên của folder gốc
A_BF_BPB_RootDirStrtClusHi	2	Số thứ tự cùa cluster đầu tiên của folder gốc (2 byte cao)
A_BF_BPB_FSInfoSec	2	Số thứ tự của sector chứa thông tin hệ thống file.
A_BF_BPB_BkUpBootSec	2	Số thứ tự của bản sao boot sector.
A_BF_BPB_Reserved	12	Dành riêng

Cấu trúc BIGFATBOOTFSINFO:		
Cấu trúc này chứa các thông tin về hệ thống file trên ổ đĩa FAT32.		
Tên	**Kích thước [byte]**	**Mô tả**
bfFSInf_Sig	4	Nhận dạng của sector thông tin hệ thống file. Giá trị này là FSINFOSIG (0x61417272L).
bfFSInf_free_clus_cnt	4	Số cluster không sử dụng (=-1 nếu không xác định)
bfFSInf_next_free_clus	4	Số thứ tự cluster định vị gần nhất
bfFSInf_resvd	12	Dành riêng

Phản chiếu FAT (mirroring):	
Trên FAT16/FAT12, FAT đầu tiên luôn là bản sơ cấp và bất kỳ thay đổi nào đầu được tự động cập nhật trên các bản sao. Đối với FAT32, phản chiếu FAT có thể bỏ qua và một bản sao khác với bản đầu tiên có thể là bản sơ cấp. Phản chiếu được cho phép bằng cách xóa bit 0080h trong thành phần *extddpb_flags* của cấu trúc DPB.	
Phản chiếu	Không
Cho phép (xóa bit 0080h)	

Mô tả

Bất kỳ khi nào một sector FAT thay đổi thì nó sẽ cập nhật cho các FAT khác.

Chỉ một FAT tích cực, thường dùng khi FAT có bad sector.

- **DPB (Drive Parameter Block):**

Cấu trúc của DPB:

```
DPB STRUC
    dpb_drive            DB    ?
    dpb_unit             DB    ?
    dpb_sector_size      DW    ?
    dpb_cluster_mask     DB    ?
    dpb_cluster_shift    DB    ?
    dpb_first_fat        DW    ?
    dpb_fat_count        DB    ?
    dpb_root_entries     DW    ?
    dpb_first_sector     DW    ?
    dpb_max_cluster      DW    ?
    dpb_fat_size         DW    ?
    dpb_dir_sector       DW    ?
    dpb_reserved2        DD    ?
    dpb_media            DB    ?
ifdef NOTFAT32
    dpb_first_access     DB    ?
else
    dpb_reserved         DB    ?
endif
    dpb_reserved3        DD    ?
    dpb_next_free        DW    ?
    dpb_free_cnt         DW    ?
ifndef NOTFAT32
    extdpb_free_cnt_hi   DW    ?
    extdpb_flags         DW    ?
    extdpb_FSInfoSec     DW    ?
    extdpb_BkUpBootSec   DW    ?
    extdpb_first_sector  DD    ?
    extdpb_max_cluster   DD    ?
    extdpb_fat_size      DD    ?
    extdpb_root_clus     DD    ?
    extdpb_next_free     DD    ?
endif
DPB ENDS
```

Tên	Mô tả
dpb_drive	Số thứ tự đĩa (0 = A, 1 = B, …)
dpb_unit	Số thứ tự của đơn vị đĩa (unit number), cho phép trình điều khiển thiết bị phân biệt đĩa.
dpb_sector_size	Số byte/sector
dpb_cluster_mask	Số sector/cluster – 1
dpb_cluster_shift	Số sector/cluster biểu diễn theo số mũ của 2
dpb_first_fat	Số thứ tự của sector đầu tiên chứa FAT
dpb_fat_count	Số FAT/đĩa
dpb_root_entries	Số điểm vào/folder gốc
dpb_first_sector	Số thứ tự sector đầu tiên của cluster đầu tiên
dpb_max_cluster	Số cluster/đĩa -1 (không dùng trong FAT32)

dpb_fat_size	Số sector đang sử dụng /FAT (= 0 đối với FAT32, thay thế bằng *extdpb_fat_size*)
dpb_dir_sector	Số thứ tự của sector đầu tiên chứa folder gốc (không dùng trong FAT32)
dpb_reserved2	Dành riêng
dpb_media	Môi trường trữ tin
reserved	Dành riêng
dpb_first_access	Xác định đĩa có dùng được hay không
dpb_reserved3	Dành riêng
dpb_next_free	Số thứ tự cluster định vị gần nhất
dpb_free_cnt	Số lượng cluster trống (=FFFFh nếu không xác định)
extdpb_free_cnt_hi	2 byte cao xác định số cluster trống
extdpb_flags	Mô tả đĩa
extdpb_FSInfoSec	Số thứ tự của sector chứa thông tin hệ thống file
extdpb_BkUpBootSec	Số thứ tự của bản sao boot sector
extdpb_first_sector	Sector đầu tiên của cluster đầu tiên
extdpb_max_cluster	Số sector/đĩa + 1
extdpb_fat_size	Số sector sử dụng bởi FAT
extdpb_root_clus	Số thứ tự của cluster đầu tiên trong folder gốc
extdpb_next_free	Số thứ tự cluster định vị gần nhất

- **Loại phân vùng:**

Loại phân vùng hợp lệ cho trong bảng sau (giá trị tương ứng chứa trong thành phần *Part_FileSystem* của cấu trúc S_PARTITION.

Giá trị	Mô tả
PART_UNKNOWN (00h)	Unknown
PART_DOS2_FAT (01h)	FAT12
PART_DOS3_FAT (04h)	FAT16 (< 32 MB)
PART_EXTENDED (05h)	Phân vùng DOS mơ rộng
PART_DOS4_FAT (06h)	FAT16 (> 32 MB)
PART_DOS32 (0Bh)	FAT32 (có thể tới2047GB)
PART_DOS32X (0Ch)	FAT32 dùng Int 13h mở rộng

PART_DOSX13 (0Eh) Giống PART_DOS4_FAT (06h) dùng Int 13h mở rộng
PART_DOSX13X (0Fh) Giống PART_EXTENDED (05h) dùng Int 13h mở rộng

- **Cấu trúc S_PARTITION:**

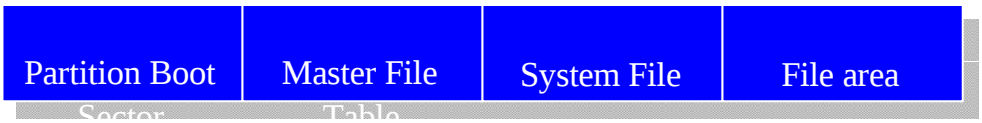
```
s_partition    STRUC
    Part_BootInd        DB    ?
    Part_FirstHead      DB    ?
    Part_FirstSector    DB    ?
    Part_FirstTrack     DB    ?
    Part_FileSystem     DB    ?
    Part_LastHead       DB    ?
    Part_LastSector     DB    ?
    Part_LastTrack      DB    ?
    Part_StartSector    DD    ?
    Part_NumSectors     DD    ?
s_partition    ENDS
```

Tên	Mô tả
Part_BootInd	Phân vùng có khởi động được (=80h) hay không (=00h)
Part_FirstHead	Head đầu tiên của phân vùng
Part_FirstSector	Sector đầu tiên của phân vùng (bit 0-5; 6-7 dùng cho Part_FirstTrack)
Part_FirstTrack	Track đầu tiên của phân vùng
PartFileSystem	Hệ thống file của phân vùng
Part_LastHead	The last head of the partition. This is a 0-based number that represents the offset from the beginning of the disk. The partition includes the head specified by this member.
Part_LastSector	Sector đầu tiên của phân vùng (bit 0-5; 6-7 dùng cho Part_LastTrack)
Part_LastTrack	Track đầu tiên của phân vùng
Part_StartSector	Số thứ tự của sector đầu tiên trên đĩa
Part_NumSectors	Số sector/phân vùng

3.3.5. Hệ thống file NTFS (New Technology File System)

NTFS là hệ thống file có hiệu suất cao và có thể tự sửa lỗi sử dụng cho Windows XP, 2000, NT. Hệ thống này hỗ trợ vấn đề bảo mật ở cấp độ file, nén file và kiểm định. Nó cũng hỗ trợ đĩa dung lượng lớn và giải pháp lưu trữ cao như RAID. NTFS cung cấp khả năng kết hợp giữa sự thực thi, độ tin cậy và tính tương thích không có trong hệ thống FAT đồng thời làm giảm thời gian thực thi các hoạt động trên file như đọc, ghi, tìm kiếm và cả các hoạt động nâng cao như khôi phục hệ thống file ở các đĩa cứng dung lượng rất lớn.

NTFS bao gồm các đặc trưng bảo mật cần thiết cho các máy dịch vụ file (file server) và các máy tính cá nhân đầu trên (high-end) trong môi trường làm việc theo nhóm. NFTS cũng hỗ trợ điều khiển xử lý dữ liệu và cấp quyền quản lý dữ liệu nhằm bảo đảm tính chính xác của dữ liệu. NTFS có thể cho phép cấp quyền xử lý cho file hay folder theo từng user riêng lẻ.



Hình 3.15 – Cấu trúc của NTFS

PBS (Partition Boot Sector) bắt đầu tại sector 0, dài 16 sector. File đầu tiên trên đĩa NTFS là MFT (Master File Table). MFT chứa các thông tin về tất cả các file và folder trên đĩa.

Các đặc trưng mới trong NTFS5 (Windows 2000):

- Mã hóa: hệ thống file mã hóa (EFS – Encrypting File System) cung cấp kỹ thuật mã hóa lõi file để lưu trữ tên đĩa NTFS.
- Chỉ tiêu đĩa (disk quota): quản lý và giới hạn dung lượng đĩa có thể sử dụng.
- Điểm phân tích lại (Reparse point): dùng cho nhiều đặc trưng lưu trữ của Windows 2000.
- Điểm cài đặt đĩa (Volume mount point): dựa trên các điểm phân tích lại để cho phép người quản trị xử lý kết hợp thư mục gốc (root) của một đĩa như là cấu trúc folder của một đĩa khác.
- File phân mảnh (Sparse file): cho phép tạo file rất lớn (dùng tất cả dung lượng đĩa).
- Hiệu chỉnh liên kết đã sắp xếp (Distributed link tracking): cung cấp dịch vụ hiệu chỉnh liến kết để bảo đảm tính toàn vẹn của shortcut.

3.3.5.1. PBS – Partition Boot Sector

Boot sector của đĩa dùng định dạng NTFS mô tả như sau:

Byte Offset	Kích thước [byte]	Mô tả
00h	3	Lệnh nhảy
03h	8	Tên nhà sản xuất
0Bh	25	BPB
24h	48	Extended BPB
54h	426	Bootstrap code
01FEh	2	Kết thúc sector

Trong đĩa NTFS, dữ liệu trong trường BPB và trường BPB mở rộng cho phép chương trình nạp (Ntldr – NT loader program) tìm kiếm MFT trong suốt quá trình khởi động. Trên đĩa NTFS, không giống như trong FAT, MFT không được chỉ rõ trên một

sector xác định trước. Do đó, MFT có thể di chuyển sang vị trí khác nếu như tồn tại sector lỗi tại vị trí mặc định. Tuy nhiên, khi dữ liệu bị sai, Windows NT/2000 sẽ xem như là đĩa chưa định dạng.

Trường BPB và BPB mở rộng trên đĩa NTFS mô tả như sau:

Byte Offset	Kích thước [byte]	Mô tả
0Bh	2	Số byte/sector
0Dh	1	Số sector/cluster
0Eh	2	Sector dự trữ
10h	3	Luôn bằng 0
13h	2	Không dùng
15h	1	Media Descriptor
16h	2	Luôn bằng 0
18h	2	Số sector/track
1Ah	2	Số lượng head
1Ch	4	Các sector ẩn
20h	4	Không dùng
24h	4	Không dùng
28h	8	Tổng số sector
30h	8	Số cluster vật lý cho file $MFT
38h	8	Số cluster vật lý cho file $MFTMirr
40h	4	Số Cluster/File Record Segment
44h	4	Số Cluster/Index Block
48h	8	Số serial của đĩa
50h	4	Checksum

Một ví dụ của đĩa NTFS chạy Windows 2000 như sau:

- Byte 00h ÷ 0Ah: chứa lệnh nhảy và OEM ID (Original Equipment Manufacturer Identification) (in đậm)
- Byte 0Bh ÷ 53h: BPB và BPB mở rộng.
- Phần còn lại: mã tự khởi động và đánh dấu kết thúc sector (in đậm).

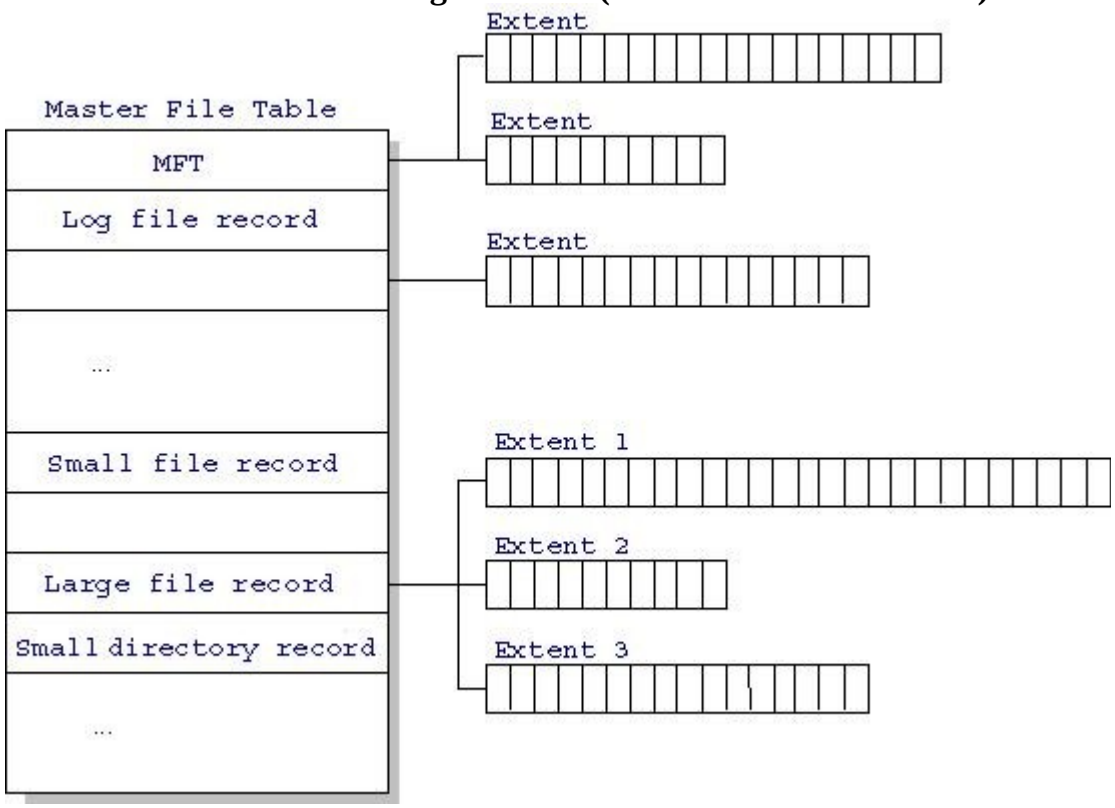
```
Physical Sector:Cyl 0, Side 1, Sector 1

00000000:EB 52 90 4E 54 46 53 20 -20 20 20 00 02 08 00 00          .R.NTFS ........
00000010:00 00 00 00 00 F8 00 00 -3F 00 FF 00 3F 00 00 00 ........?...?...
00000020:00 00 00 00 80 00 80 00 -4A F5 7F 00 00 00 00 00 ........J.......
00000030:04 00 00 00 00 00 00 00 -54 FF 07 00 00 00 00 00 ........T.......
00000040:F6 00 00 00 01 00 00 00 -14 A5 1B 74 C9 1B 74 1C ...........t..t.
00000050:00 00 00 00 FA 33 C0 8E -D0 BC 00 7C FB B8 C0 07 .....3.....|....
00000060:8E D8 E8 16 00 B8 00 0D -8E C0 33 DB C6 06 0E 00 ..........3.....
00000070:10 E8 53 00 68 00 0D 68 -6A 02 CB 8A 16 24 00 B4 ..S.h..hj....$..
00000080:08 CD 13 73 05 B9 FF FF -8A F1 66 0F B6 C6 40 66 ...s......f...@f
```

00000090:0F B6 D1 80 E2 3F F7 E2 -86 CD C0 ED 06 41 66 0F .....?.....Af.

```
000000A0:B7 C9 66 F7 E1 66 A3 20 -00 C3 B4 41 BB AA 55 8A ..f..f....A..U.
000000B0:16 24 00 CD 13 72 0F 81 -FB 55 AA 75 09 F6 C1 01 .$...r...U.u....
000000C0:74 04 FE 06 14 00 C3 66 -60 1E 06 66 A1 10 00 66 t......f`..f...f
000000D0:03 06 1C 00 66 3B 06 20 -00 0F 82 3A 00 1E 66 6A ....f;....:..fj
000000E0:00 66 50 06 53 66 68 10 -00 01 00 80 3E 14 00 00 .fP.Sfh.....>...
000000F0:0F 85 0C 00 E8 B3 FF 80 -3E 14 00 00 0F 84 61 00 ........>.....a.
00000100:B4 42 8A 16 24 00 16 1F -8B F4 CD 13 66 58 5B 07 .B..$.......fX [.
00000110:66 58 66 58 1F EB 2D 66 -33 D2 66 0F B7 0E 18 00 fXfX.-f3.f......
00000120:66 F7 F1 FE C2 8A CA 66 -8B D0 66 C1 EA 10 F7 36 f......f..f....6
00000130:1A 00 86 D6 8A 16 24 00 -8A E8 C0 E4 06 0A CC B8 ......$.........
00000140:01 02 CD 13 0F 82 19 00 -8C C0 05 20 00 8E C0 66 ..............f
00000150:FF 06 10 00 FF 0E 0E 00 -0F 85 6F FF 07 1F 66 61 ..........o...fa
00000160:C3 A0 F8 01 E8 09 00 A0 -FB 01 E8 03 00 FB EB FE ................
00000170:B4 01 8B F0 AC 3C 00 74 -09 B4 0E BB 07 00 CD 10 .....<.t........
00000180:EB F2 C3 0D 0A 41 20 64 -69 73 6B 20 72 65 61 64 .....A disk read
00000190:20 65 72 72 6F 72 20 6F -63 63 75 72 72 65 64 00 error occurred.
000001A0:0D 0A 4E 54 4C 44 52 20 -69 73 20 6D 69 73 73 69 ..NTLDR is missi
000001B0:6E 67 00 0D 0A 4E 54 4C -44 52 20 69 73 20 63 6F ng...NTLDR is co
000001C0:6D 70 72 65 73 73 65 64 -00 0D 0A 50 72 65 73 73 mpressed...Press
000001D0:20 43 74 72 6C 2B 41 6C -74 2B 44 65 6C 20 74 6F Ctrl+Alt+Del to
000001E0:20 72 65 73 74 61 72 74 -0D 0A 00 00 00 00 00 00 restart........
000001F0:00 00 00 00 00 00 00 00 -83 A0 B3 C9 00 00 55 AA ..............U.
```

3.3.5.2. Bảng file chính (MFT – Master File Table)



Hình 3.16 - Cấu trúc của MFT

Mỗi file trên đĩa NTFS đại diện bằng một bản ghi (record) trong một file đặc biệt gọi là MFT. NTFS dành riêng 16 sector đầu tiên cho các thông tin đặc biệt. Bản ghi đầu tiên mô tả chính MFT, sau đó là bản ghi MFT ảnh (MFT mirror record). Nếu bản ghi đầu

tiên bị sai thì NTFS sẽ đọc bản ghi thứ hai để tìm file MFT ảnh trong đó bản ghi đầu tiên của MFT ảnh giống hệt như trong MFT. Vị trí của đoạn dữ liệu của cả hai file MFT và MFT ảnh được ghi lại trong boot sector. Một bản sao của boot sector được lưu trữ tại vị trí giữa (vật lý) của đĩa.

Bản ghi thứ ba là log file dùng cho mục đích khôi phục file. Bản ghi thứ 17 và các bản ghi ở phía sau dùng cho mỗi file và thư mục trong đĩa. MFT cấp một không gian nào đó cho mỗi bản ghi. File và thư mục nhỏ (1500 byte hay nhỏ hơn) có thể chứa hoàn toàn bên trong MFT.

Thiết kế này làm cho tốc độ xử lý file nhanh hơn. Đối với hệ thống file FAT (dùng FAT để liệt kê tên và địa chỉ của mỗi file), mỗi điểm vào (entry) FAT chứa một chỉ số trong bảng. Như vậy, để tìm một file trong hệ thống FAT, đầu tiên phải đọc bảng định vị file để đảm bảo file tồn tại. Sau đó, FAT lấy file bằng cách tìm kiếm chuỗi các vị trí được gán cho file. Đối với NTFS, file sẽ được lấy ra ngay lập tức.

Hình 3.17 - Bản ghi MFT cho file nhỏ và thư mục

Bản ghi thư mục được cấp chỗ trong MFT giống như bản ghi file (thư mục không chứa dữ liệu mà chứa các thông tin chỉ số). Bản ghi thư mục nhỏ có thể chứa hoàn toàn bên trong cấu trúc MFT trong đó các thư mục lớn hơn sẽ được chứa dưới dạng cây nhị phân (B-tree) và dùng con trỏ để xác định các cluster chứa các mục không chứa trong cấu trúc MFT.

3.3.5.3. Phân loại file NTFS

- **Thuộc tính file NTFS:**

NTFS quan niệm mỗi file (hay folder) như tập hợp các thuộc tính file. Các phần tử của tập hợp này là tên file, các thông tin bảo mật file và cả dữ liệu trong file. Mỗi thuộc tính được xác định bằng mã và tên thuộc tính. Khi các thuộc tính chứa trong bản ghi file MFT, chúng được gọi là các thuộc tính nội trú (resident attribute). Ví dụ như tên file và đặc tính thời gian luôn chứa trong bản ghi file MFT. Nếu các thông tin của file quá lớn so với bản ghi MFT, một số thuộc tính của chúng phải là ngoại trú (nonresident). Các thuộc tính ngoại trú được định vị trên một hay nhiều cluster bất kỳ nào đó trên đĩa. NTFS tạo thuộc tính Attribute List (danh sách thuộc tính) để mô tả vị trí của tất cả các bản ghi thuộc tính.

Các thuộc tính định nghĩa bằng hệ thống file NTFS:

Loại thuộc tính	Mô tả
Thông tin chuẩn (Standard Information)	Đặc trưng thời gian và số lượng liên kết
Danh sách thuộc tính (Attribute List)	Danh sách vị trí các bản ghi thuộc tính không chứa trong MFT
Tên file (File Name)	Tên file dài có thể đến 255 ký tự Unicode, tên file ngắn có dạng 8.3, không phân biệt chữa hoa và thường
Mô tả bảo mật (Security Descriptor)	Xác định tác giả và các user được phép xử lý file
Dữ liệu (Data)	Chứa dữ liệu file. NTFS cho phép nhiều thuộc tính dữ liệu trên mỗi file. Về cơ bản, mỗi file có một thuộc tính dữ liệu không tên và có thể có thêm các thuộc tính được đặt tên, mỗi thuộc tính có cú pháp riêng
Nhận dạng đối tượng (Object ID)	Duy nhất cho mỗi file, dùng cho dịch vụ hiệu chỉnh liên kết và không phải file nào cũng có thuộc tính này
Logged Tool Stream	Tương tự luồng dữ liệu nhưng dùng cho hoạt động lưu trữ log file
Điểm phân tích lại (Reparse Point)	Các điểm cài đặt đĩa, dùng cho IFS (Installable File System)
Chỉ số gốc (Index root)	Xử lý folder và các chỉ số khác
Chỉ số định vị (Index allocation)	Xử lý folder và các chỉ số khác
Bitmap	Xử lý folder và các chỉ số khác
Thông tin đĩa (Volume information)	Chứa phiên bản đĩa
Tên đĩa (Volume name)	Nhãn đĩa

File hệ thống NTFS:

File hệ thống là file lưu trữ dữ liệu (metadata) của hệ thống file và bổ sung thêm cho hệ thống file. Các file hệ thống được đặt trên đĩa bằng ứng dụng Format.

File hệ thống	Tên file	Bản ghi MFT	Mô tả
MFT		Log file	
MFT2			

$Mft	0	Chứa một bản ghi file cơ sở cho mỗi file và folder trên đĩa NTFS. Nếu thông tin lớn hơn một bản ghi đơn thì sẽ dùng thêm các bản ghi khác.
$MftMirr	1	Ảnh của 4 bản ghi đầu tiên của MFT.
$LogFile	2	Chứa danh sách các bước dùng cho khả năng khôi phục NTFS. Kích thước log file phụ thuộc vào kích thước đĩa

			(có thể tới 4 MB).
Đĩa	$Volume	3	Chứa thông tin đĩa như nhãn đĩa và phiên bản.
Định nghĩa thuộc tính	$AttrDef	4	Tên thuộc tính, số thứ tự và mô tả.
Chỉ số tên file gốc	$	5	Folder gốc
Cluster bitmap	$Bitmap	6	Mô tả cluster nào sử dụng
Boot sector	$Boot	7	Bao gồm BPB và mã khởi động (bootstrap loader).
File cluster xấu	$BadClus	8	Bad cluster của đĩa
Bảo mật file	$Secure	9	Chứa các mô tả bảo mật của file
Bảng chữ hoa	$Upcase	10	Chuyển chữ thường thành chữ hoa để thích hợp với ký tự Unicode.
File mở rộng NTFS	$Extend	11	Dùng cho các mục đích mở rộng như chỉ tiêu đĩa, điểm phân tích lại và nhận dạng đối tượng
		12–15	Dành riêng

- **Đa luồng dữ liệu NTFS:**

NTFS hỗ trợ đa luồng dữ liệu trong đó tên luồng đồng nhất với thuộc tính dữ liệu mới của file. Mỗi luồng dữ liệu, tập hợp của thuộc tính file, đại diện bằng một handle. Đặc trưng này cho phép quản lý dữ liệu như các đơn vị riêng lẻ. Khi sao chép một file trên đĩa NTFS sang đĩa FAT, các luồng dữ liệu và các thuộc tính khác không hỗ trợ bởi FAT sẽ bị mất.

- **File nén NTFS:**

WinNT/2000 hỗ trợ nén file, folder riêng lẻ và toàn bộ đĩa NTFS. File nén trên đĩa NTFS có thể đọc / ghi bằng các ứng dụng trên nền Windows mà không cần giải nén bằng các chương trình khác. Khi đọc file, file sẽ được tự động giải nén và sau đó nén lại một lần nữa khi đóng hay lưu file. Thuật toán nén trên NTFS hỗ trợ kích thước cluster lên tới 4 KB. Nếu kích thước cluster lớn hơn 4 KB thì không sử dụng chức năng nén. Mỗi luồng dữ liệu NTFS chứa các thông tin xác định có phần nào của luồng được nén hay không.

Hệ thống file mã hóa (EFS - Encrypting File System):

- ***Files and Folders (chỉ trong NTFS5):***

EFS cung cấp kỹ thuật mã hóa file để lưu trữ file trên đĩa NTFS. EFS cung cấp khả năng lưu trữ an toàn, chống lại sự xâm nhập từ bên ngoài. User làm việc với file và folder đã mã hóa giống như với các file thông thường. Hệ thống sẽ tự động giải mã khi xử lý file

hay folder và sau đó sẽ mã hóa lại. Nếu user không được cấp quyền thì sẽ khnôg thể xử lý được file. EFS có các lợi ích sau cho các ứng dụng mã hóa của hãng thứ 3:

+ Trong suốt đối với user và ứng dụng: user không cần nhớ mật mã để giải mã file.

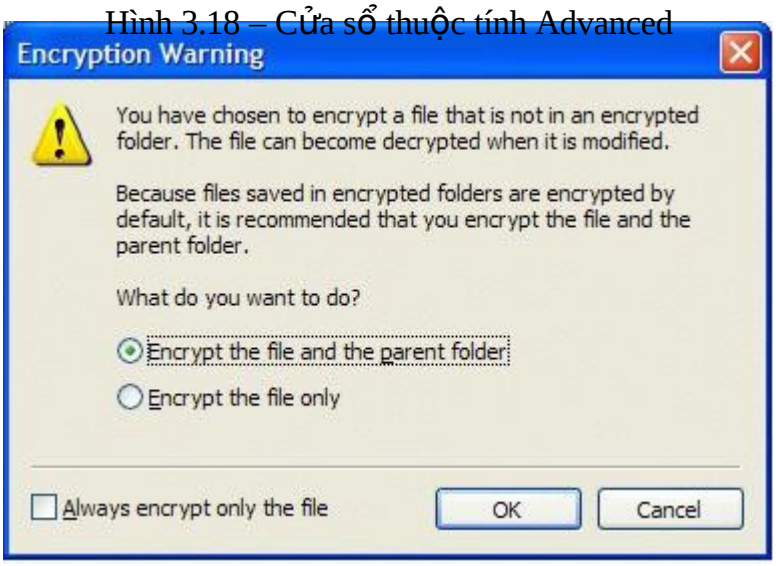
+ Độ bảo mật từ khóa cao.

+ Các quá trình mã hóa / giải mã thực hiện ở chế độ nhân (kernel), tránh rủi ro mất từ khóa.

+ Cung cấp cơ chế khôi phục dữ liệu rất có giá trị trong môi trường thương mại, cho phép khôi phục dữ liệu ngay cả khi nhân viên mã hóa đã rời công ty.

User có thể hiển thị các đặc trưng EFS bằng cách dùng lệnh **cipher.exe** hay dùng Windows Explorer (right-click trên file, chọn thẻ *General*, click nút *Advanced*).

Hình 3.18 – Cửa sổ thuộc tính Advanced

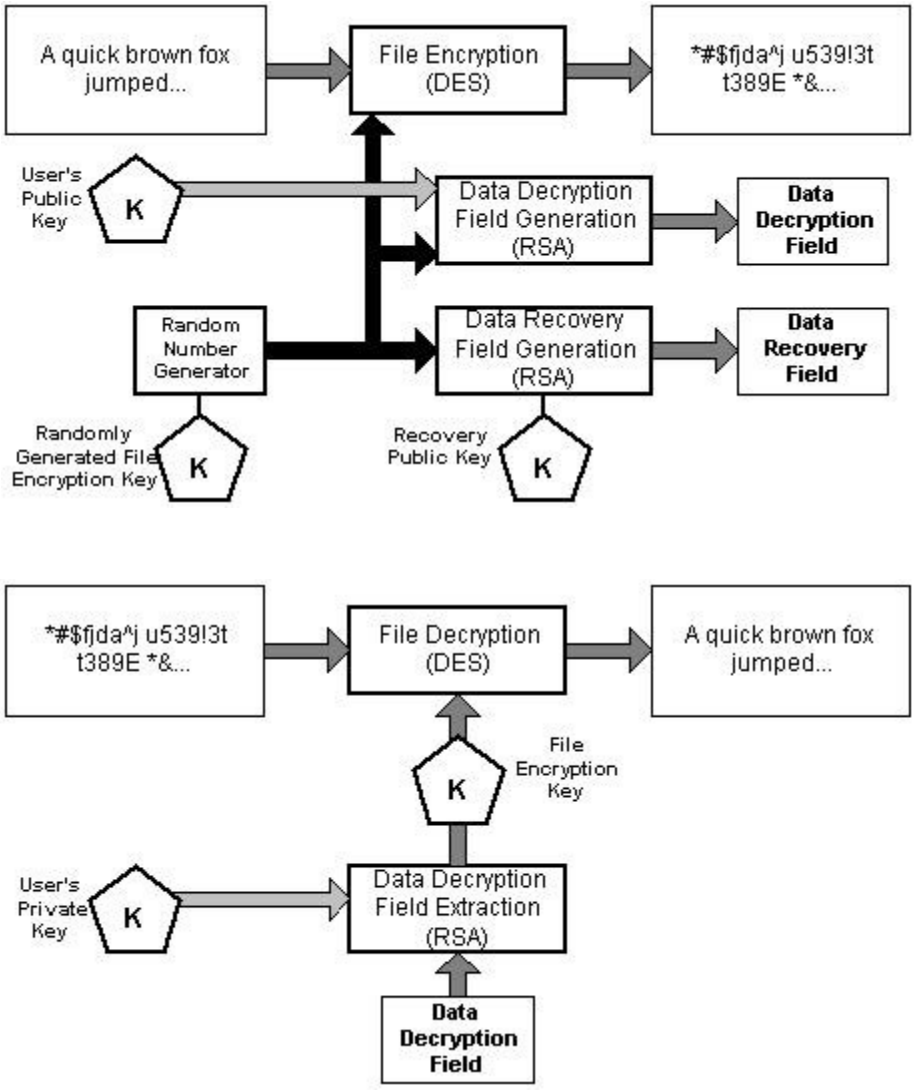


Hình 3.19 – Cửa sổ Encryption Warning

Sau đó, Windows yêu cầu user xác định chỉ mã hóa một file hay toàn folder. Khi xử lý file, Windows sẽ tạo ra một bản sao ẩn không mã hóa và làm việc trên bản sao này. Sau khi xử lý xong, Windows sẽ cập nhật file và xóa bản sao. Bản sao này chính là một lỗ hổng bảo mật do nó được lưu trữ ở dạng không mã hóa. Quá trình mã hóa toàn folder sẽ giải quyết được vấn đề này.

- ***EFS:***

EFS kết hợp kỹ thuật khóa công cộng và mã hóa khóa đối xứng để bảo vệ file. Dữ liệu trên file được mã hóa dùng thuật toán đối xứng. Khóa mã gọi là khóa mã hóa file (FEK – File Encryption Key). FEK được mã hóa dùng thuật toán công cộng RSA (1024 bit) và lưu trữ trên file. Sau khi file đã mã hóa, chỉ có các user có DDF và DRF phù hợp mới có thể xử lý file.



Hình 3.20 – Sơ đồ mã hóa và giải mã

3.3.6. So sánh NTFS và FAT

Tiêu chuẩn	NTFS5	NTFS	FAT32	FAT16
Hệ điều hành	Win2000, XP	WinNT, 2000, XP	Win98, ME, 2000, XP	DOS, Windows
Giới hạn				
Kích thước đĩa tối đa	2TB	2TB	2TB	2GB
Số file tối đa	Không giới hạn	Không giới hạn	Không giới hạn ~	65000
Kích thước file tối đa	Theo kích thước đĩa	Theo kích thước đĩa	4GB	2GB
Số cluster tối đa	Không giới hạn	Không giới hạn	268,435,456	65,535
Chiều dài tên file tối đa	255	255	255	Chuẩn: 8.3 Mở rộng: 255
Đặc trưng của hệ thống file				
Tên file Unicode	Ký tự Unicode	Ký tự Unicode	Ký tự hệ thống	Ký tự hệ thống
Ảnh bản ghi hệ thống	File MFT Mirror	File MFT Mirror	Bản sao của FAT	Bản sao của FAT
Vị trí boot sector	Sector đầu tiên và cuốicùng	Sector đầu tiên và cuốicùng	Sector đầu tiên	Sector đầu tiên
Thuộc tính file	Chuẩn và tùy ý	Chuẩn và tùy ý	Chuẩn	Chuẩn
Luồng liên tục	Có	Có	Không	Không
Nén	Có	Có	Không	Không
Mã hóa	Có	Không	Không	Không
Cấp quyền	Có	Có	Không	Không
Chỉ tiêu đĩa	Có	Không	Không	Không
File rời rạc	Có	Không	Không	Không
Điểm phân tích lại	Có	Không	Không	Không
Hiệu suất toàn đĩa				

Tự bảo mật	Có	Có	Không	Không
Khả năng khôi phục	Có	Có	Không	Không
Hiệu suất	Thấp trên đĩa nhỏ Cao trên đĩa lớn	Thấp trên đĩa nhỏ Cao trên đĩa lớn	Thấp trên đĩa lớn Cao trên đĩa nhỏ	Thấp trên đĩa lớn Cao trên đĩa nhỏ
Tiết kiệm không gian đĩa	Lớn	Lớn	Trung bình	Nhỏ trên đĩa lớn

3.4. Truy xuất ổ đĩa qua DOS và BIOS

3.4.1. Đĩa mềm

3.4.1.1. DOS

DOS cung cấp 3 ngắt cho quá trình truy xuất đĩa mềm và cứng là 25h, 26h, 21h.

Ngắt 25h: đọc sector

Thanh ghi	Giá trị gọi	Trả về
AL	Số thứ tự ổ đĩa	
CX	Số sector	
DX	Sector đầu	Mã lỗi
BX	Offset của vùng đệm đọc	
DS	Đoạn của vùng đệm đọc	
CF		

Ngắt 26h: ghi sector

Thanh ghi	Giá trị gọi	Trả về
AL	Số thứ tự ổ đĩa	
CX	Số sector	
DX	Sector đầu	Mã lỗi
BX	Offset của vùng đệm ghi	
DS	Đoạn của vùng đệm ghi	
CF		Lỗi nếu > 0

Mã lỗi trả về như sau:

Mã	Lỗi
01h	Lệnh không hợp lệ
02h	
04h	
08h	
10h	
20h	
40h	
80h	

Che địa chỉ không chính xác
Không tìm thấy sector
Tràn DMA
Lỗi CRC hay ECC
Lỗi bộ điều khiển
Lỗi tìm kiếm

Ổ đĩa không sẵn sàng

3.4.1.2. BIOS

Việc truy xuất ổ đĩa dùng ngắt 13h:

Hàm	Chức năng
00h	Khởi động ổ đĩa
01h	Đọc trạng thái và tác vụ cuối cùng
02h	Đọc sector
03h	Ghi sector
04h	Kiểm tra sector
05h	Định dạng track

3.4.2. Đĩa cứng

Giống như đĩa mềm, các sector logic có thể truy xuất bằng ngắt 25h, 26h của DOS. Các sector vật lý truy xuất bằng ngắt 13h của BIOS. Các hàm dùng cho đĩa cứng như sau:

Hàm	Chức năng
05h	Định dạng track và cylinder
06h	Định dạng và đánh dấu track xấu
07h	Định dạng và đánh dấu ổ đĩa
08h	Xác định các thông số ổ đĩa
09h	Cài đặt thông số ổ đĩa
0Ah	Đọc sector mở rộng
0Bh	Ghi sector mở rộng
0Ch	
0Dh	
0Eh	
0Fh	
10h	
11h	
19h	

Tìm kiếm

Khởi tạo đĩa cứng

Đọc bộ đệm sector

Ghi bộ đệm sector

Kiểm tra ổ đĩa xem đã sẵn sàng chưa

Chuẩn lại ổ đĩa

Nâng đầu từ đọc / ghi

Mã lỗi điều khiển:

Mã	Lỗi
00h	Không lỗi
02h	Không có tín hiệu tìm kiếm
03h	Lỗi ghi
04h	Ổ đĩa không sẵn sàng
06h	Không tìm tấy track 0
10h	Lỗi ECC trong trường ID
11h	Lỗi ECC trong trường dữ liệu
12h	Không có che địa chỉ ID
13h	Không có che địa chỉ dữ liệu
14h	Không có trường ID
15h	Lỗi tìm kiếm
16h	Lỗi bộ điều khiển bên trong
17h	Lỗi DMA
18h	Lỗi dữ liệu có thể sửa được

3.5. Đĩa quang

Ngày này, đĩa quang đã được sử dụng phổ biến, chúng có mật độ ghi thông tin cao hơn đĩa từ thông thường. Các đĩa quang dựa trên cùng một cộng nghệ được sử dụng trong Compact Disc để ghi âm nên được gọi tên là CD ROM.

- ***Nguyên lý chế tạo:***

Các đĩa CD ROM được tạo ra bằng cách dùng một tia laser mạnh đốt chảy các hốc đường kính 1 µm trên một đĩa chủ, từ đĩa chủ này sẽ tạo ra các khuôn để tạo các bản copy trên đĩa chất dẻo. Sau đó, phủ một lớp nhôm mỏng lên trên mặt đĩa và lại phủ một lớp chất dẻo trong suốt lên lớp nhôm để bảo vệ. Lớp nhôm có tác dụng phản xạ tia laser. Các hốc nhỏ được gọi là pit, diện tích không bị dốt nóng gọi là land. Chúng có độ phản xạ khác nhau nên có thể phân biệt được pit và land.

- ***Tổ chức dữ liệu:***

Thông tin trên CD ROM được ghi theo một đường xoắn ốc duy nhất và ghi thành từng nhóm 24 byte, mỗi byte được mở rộng thành 14 bit bằng cách dùng mã sửa sai Reed – Solomon. Ba bit được thêm vào giữa các nhóm và một byte đồng bộ được bổ sung để tạo thành 1 frame. 98 frame tạo thành một block chứa 2 KB dữ liệu. CD ROM có thể chứa 270,000 block tương ứng với dung lượng 553 MB.

Các đĩa CD ROM được đọc bằng một đầu dò đo năng lượng phản xạ từ bề mặt đĩa khi chiếu lên đó một tia laser công suất nhỏ. Dữ liệu được đọc với tốc độ 75 inches/s, cho tốc độ đọc dữ liệu là 153.6 KBps.

- ***WORM (Write Once Read Many):***

Các CD ROM mô tả như trên không thể ghi được, do đó thế hệ đĩa quang thứ hai ra đời, đó là WORM. Thiết bị này cho phép người sử dụng tự mình ghi thông tin nhưng sau khi đã tạo ra các pit thì không thể thay đổi được nữa. Cơ chế ghi của WORM có 2 loại tùy vào cấu trúc bề mặt:

+ Lớp phủ đĩa ở vùng bị đốt nóng sẽ bay hơi và làm lộ ra bề mặt của vùng đĩa dưới không còn lớp phủ. Hai vùng này có hệ số phản xạ khác nhau nên lưu trữ bit khác nhau.

+ Ở cuối quá trình ghi bằng xung laser, lớp phủ đĩa bị chảy ra sẽ được làm lạnh nhanh và kết rắn lại ở dạng vô định hình. Lớp này sẽ có hệ số phản xạ khác với lớp phủ cũ.

- ***Ổ đĩa quang từ (Magneto Optical):***

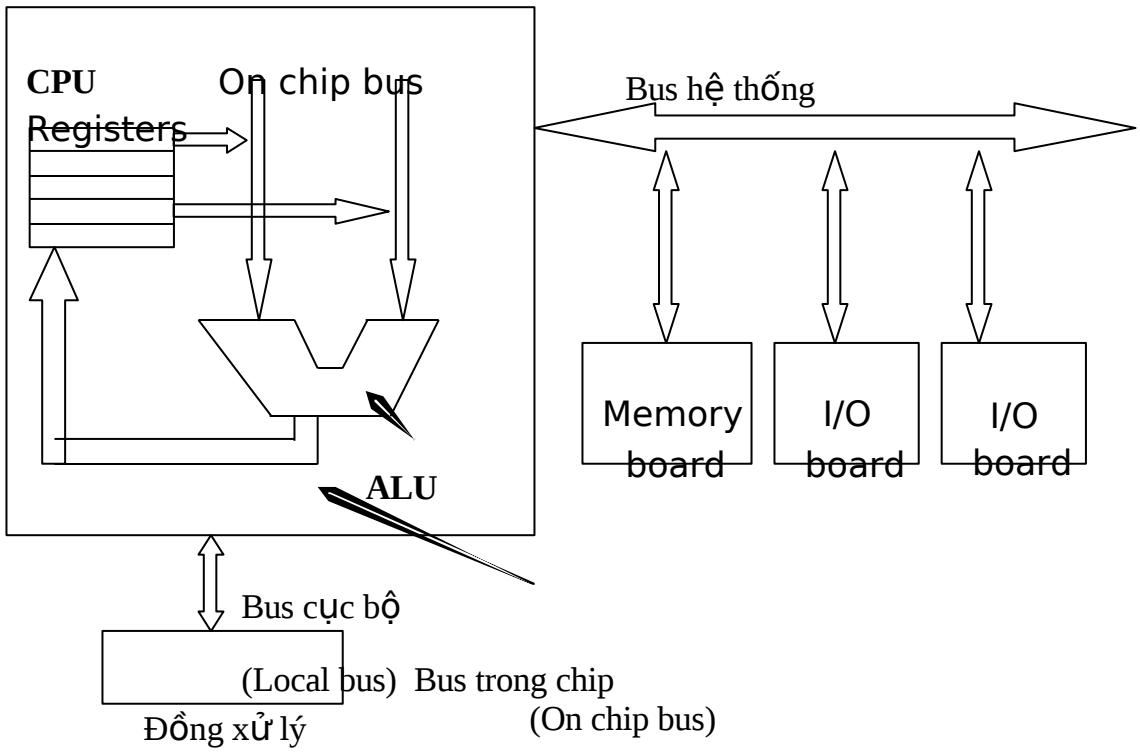
Các đĩa quang thế hệ thứ 3 là những môi trường quang học có thể xóa được. Đĩa có một lớp phủ đồng nhất làm bằng vật liệu hợp kim latan sắt từ. Khi tia laser phân cực đốt bề mặt đĩa, hướng phân cực của tia phản xạ được quay phụ thuộc vào mức nhiễm từ của bề mặt.

Khi cần ghi một bit lên đĩa, một xung laser ngắn và mạnh sẽ đốt bề mặt đĩa ở vùng cần thiết làm mức nhiễm từ của vùng này bằng 0. Cùng lúc đó, một nam châm phát ra từ trường có hướng phụ thuộc vào bit cần ghi là 0 hay 1.Hướng của từ trường sẽ xác định hướng của các domain từ trong vùng bị đốt nóng khi được làm lạnh xuống.

Khi đọc một bit, tia laser sẽ quét bề mặt đĩa và hệ phân cực sẽ nhạy với hướng phân cực của tia phản xạ. Nếu chiếu tia laser trên vùng đã ghi dữ liệu, hướng của mặt phẳng phân cực tia phản xạ phụ thuộc vào hướng của các domain từ. Từ đó, ánh sáng sẽ có thể tới bộ lọc phân cực (ứng với mức 1) hay không (ứng với mức 0). Để xóa dữ liệu, ta phải làm nóng các điểm xác định với hướng phù hợp của từ trường.

Chương 4
BUS

Bus là đường truyền tín hiệu điện nối các thiết bị khác nhau trong một hệ thống máy tính. Bus thường có từ 50 đến 100 dây dẫn được gắn trên mainboard, trên các dây này có các đầu nối đưa ra, các đầu này được sắp xếp và cách nhau những khoảng quy định để có thể cắm vào đó những I/O board hay board bộ nhớ (bus hệ thống – system bus).



Hình 4.1 - Các bus trong một hệ thống máy tính

Cũng có những bus dùng cho mục đích chuyên biệt, thí dụ nối 1 vi xử lý với 1 hay nhiều vi xử lý khác hoặc nối với bộ nhớ cục bộ (local bus).

Trong vi xử lý cũng có một số bus để nối các thành phần bên trong của bộ vi xử lý với nhau. Người thiết kế chip vi xử lý có thể tuỳ ý lựa chọn loại bus bên trong nó, còn với các bus liên hệ bên ngoài cần phải xác định rõ các quy tắc làm việc cũng như các đặc điểm kỹ thuật về điện và cơ khí của bus để người thiết kế mainboard có thể ghép nối chip vi xử lý với các thiết bị khác. Nói cách khác, các bus này phải tuân theo 1 chuẩn nào đó. Tập các quy tắc của chuẩn còn được gọi là giao thức bus (bus protocol)

Ngoài ra, có rất nhiều loại bus khác nhau được sử dụng, các bus này nói chung là không tương thích với nhau. Một số bus được sử dụng phổ biến:

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

Bus thường phân loại theo 3 cách sau:

- Theo tổ chức phần cứng (như trên).
- Theo giao thức truyền thông (bus đồng bộ và không đồng bộ).
- Theo loại tín hiệu truyền trên bus (bus địa chỉ, bus dữ liệu,…).

1. Bus hệ thống

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

Ví dụ: Khi CPU ra lệnh cho bộ điều khiển đĩa đọc/ghi một khối dữ liệu thì CPU là master còn bộ điều khiển đĩa là slave. Tuy nhiên, bộ điều khiển đĩa ra lệnh cho bộ nhớ nhận dữ liệu thì nó lại giữ vai trò master.

1.1. Bus Driver và Bus Receiver

Tín hiệu điện trong máy tính phát ra thường không đủ để điều khiển bus, nhất là khi bus khá dài và có nhiều thiết bị nối với nó. Chính vì thế mà hầu hết các bus master được nối với bus thông qua 1 chip gọi là ***bus driver***, về cơ bản nó là một bộ khuếch đại tín hiệu số. Tương tự như vậy, hầu hết các slave được nối với bus thông qua ***bus receiver***. Đối với các thiết bị khi thì đóng vai trò master, khi thì đóng vai trò slave, người ta sử dụng 1 chip kết hợp gọi là ***transceiver***. Các chip này đóng vai trò ghép nối và là các thiết bị 3 trạng thái, cho phép nó có thể ở trạng thái thứ 3 – hở mạch (thả nổi).

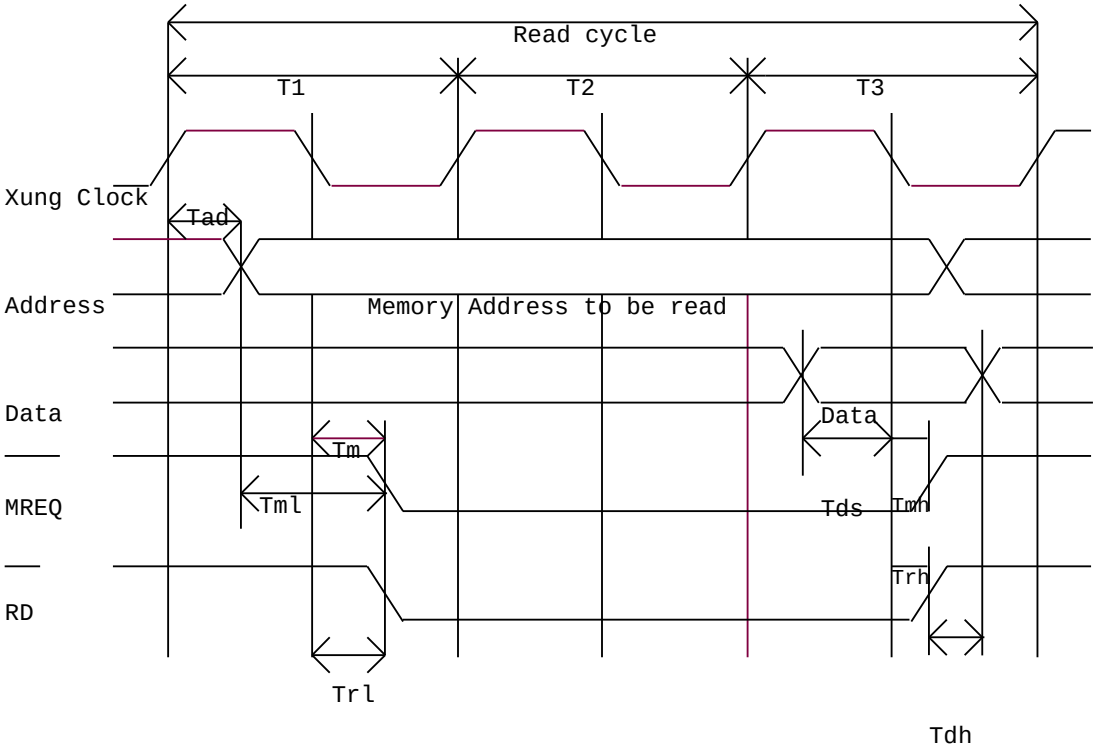
Giống như vi xử lý, bus có các đường địa chỉ, đường số liệu và đường điều khiển. Tuy nhiên, không nhất thiết có ánh xạ 1 – 1 giữa các tín hiệu ở các chân ra của vi xử lý và các đường dây của bus. Thí dụ: một số chíp vi xử lý có 3 chân ra, truyền ra các tín hiệu báo chíp vi xử lý đang thực hiện các thao tác MEMR , MEMW , IOR , IOW hay thao tác khác. Một bus điển hình thường có 4 đường trên.

Các vấn đề quan trọng nhất liên quan đến thiết kế bus là: xung clock bus (sự phân chia thời gian, hay còn gọi là **bus blocking**), cơ chế phân xử bus (bus arbitration), xử lý ngắt và xử lý lỗi.

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

1.2. Bus đồng bộ (Synchronous bus)

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



Hình 4.2 - Chu kỳ đọc trong bus đồng bộ

Hình trên là giản đồ thời gian của một bus đồng bộ với tần số xung clock là 4MHz, như vậy chu kỳ bus là 250ns.

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

- T1 bắt đầu bằng cạnh dương của ~~a xung~~ clock~~, tr~~ong một phần thời gian của T1, ~~vi xử lý~~ đặt địa chỉ byte cần đọc lên bus địa chỉ. Sau khi tín hiệu địa chỉ được xác lập, vi xử lý đặt các tín hiệu MREQ và RD tích cực (mức thấp). Tín hiệu MREQ (Memory Request) - xác định truy xuất bộ nhớ chứ không phải thiết bị I/O, còn tín hiệu RD - chọn đọc chứ không phải ghi dữ liệu.
- T2: thời gian cần thiết để bộ nhớ giải mã địa chỉ và đưa dữ liệu lên bus dữ liệu.

- T3: tại cạnh âm của T3, vi xử lý nhận dữ liệu trên bus dữ liệu, chứa vào thanh ghi bên trong vi xử lý và chốt dữ liệu. Sau đó vi xử lý đảo các tín hiệu $\overline{MREQ}$ và $\overline{RD}$.

Như vậy thao tác đọc đã hoàn thành, tại chu kỳ máy tiếp theo vi xử lý có thể thực hiện thao tác khác. Các giá trị cụ thể về thời gian của hình vẽ trên có thể được giải thích chi tiết như sau:

- T_{AD}: $T_{AD} \leq$ 110ns, nghĩa là nhà sản xuất vi xử lý đảm bảo rằng trong mọi chu kỳ đọc toán hạng từ bộ nhớ, vi xử lý sẽ đưa ra tín hiệu địa chỉ không nhiều hơn 110 ns tính từ thời điểm cạnh dương của T1.
- T_{DS}: giá trị nhỏ nhất là 50ns, có nghĩa là nhà sản xuất bộ nhớ phải đảm bảo rằng dữ liệu đã ổn định trên bus dữ liệu ít nhất là 50ns trước điểm giữa cạnh âm của T3. Yêu cầu này đảm bảo cho vi xử lý đọc dữ liệu tin cậy.

Khoảng thời gian bắt buộc đối với T_{AD} và T_{DS} xác định rằng trong trường hợp xấu nhất, bộ nhớ chỉ có 250 + 250 + 125 – 110 – 50 = 465 ns tính từ thời điểm có tín hiệu địa chỉ cho tới khi tạo ra dữ liệu trên bus dữ liệu. Nếu bộ nhớ không có khả năng đáp ứng đủ nhanh, nó phát tín hiệu $\overline{WAIT}$ trước cạnh âm của T2. Thao tác này đưa thêm các trạng thái chờ – wait state (tức là đưa thêm vào 1 chu kỳ bus), khi bộ nhớ đã đưa ra tín hiệu ổn định, nó sẽ đảo $\overline{WAIT}$ thành $WAIT$.

- T_{ML}: đảm bảo tín hiệu địa chỉ sẽ được xác lập trước tín hiệu $\overline{MREQ}$ ít nhất 60ns. Khoảng thời gian này sẽ quan trọng nếu tín hiệu $\overline{MREQ}$ điều khiển quá trình tạo tín hiệu chọn chip ($\overline{CS}$ hay $\overline{CE}$) do một số chip nhớ đòi hỏi phải nhận được tín hiệu địa chỉ trước tín hiệu chọn chip. Như vậy, không thể chọn chip nhớ với thời gian thiết lập 75ns.
- T_M, T_{RL}: cho phép 2 tín hiệu $\overline{MREQ}$ và $\overline{RD}$ tích cực trong khoảng thời gian 85ns tính từ thời điểm xuống của xung clock T1. Trong trường hợp xấu nhất, chíp nhớ chỉ có 250 + 250 – 85 – 50 = 365ns sau khi 2 tín hiệu trên tích cực để đưa dữ liệu ra bus dữ liệu. Sự bắt buộc về thời gian này bổ sung thêm sự bắt buộc thời gian với tín hiệu clock.
- T_{MH}, T_{RH} : thời gian để các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ được đảo sau khi dữ liệu đã được vi xử lý nhận vào.
- T_{DH}: Thời gian bộ nhớ cần giữ data trên bus sau khi tín hiệu $\overline{RD}$ đã đảo

Giản đồ thời gian một chu kỳ đọc trên bus đồng bộ đã được đơn giản hoá so với thực tế, trong đó các tín hiệu cần sử dụng lớn hơn nhiều. Giá trị tới hạn của các thông số cho trong bảng sau:

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

- **Truyền theo khối:**

Ngoài các chu kỳ đọc/ghi, một số bus truyền dữ liệu đồng bộ còn hỗ trợ truyền dữ liệu theo khối. Khi bắt đầu thao tác đọc khối, bus master báo cho slave biết số byte cần được truyền đi, thí dụ truyền con số này đi trong chu kỳ T1, sau đó đáng lẽ truyền đi 1 byte, slave đưa ra trong mỗi chu kỳ 1 byte cho tới khi đủ số byte được thông báo. Như vậy, khi đọc dữ liệu theo khối, n byte dữ liệu cần n+2 chu kỳ clock chứ không phải 3n chu kỳ.

Một cách khác để cho truyền dữ liệu nhanh hơn là giảm chu kỳ. Ở ví dụ trên: 1 byte được truyền đi trong 750ns, vậy bus có tốc độ truyền 1.33MBps. Nếu xung clock có tần số 8MHz, thời gian 1 chu kỳ chỉ còn một nửa, tốc độ sẽ là 2.67MBps. Tuy nhiên, giảm chu kỳ bus dẫn đến khó khăn về mặt kỹ thuật, các tín hiệu truyền trên các đường khác nhau không phải luôn có cùng tốc độ, dẫn đến hiệu ứng *bus skew*. 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.

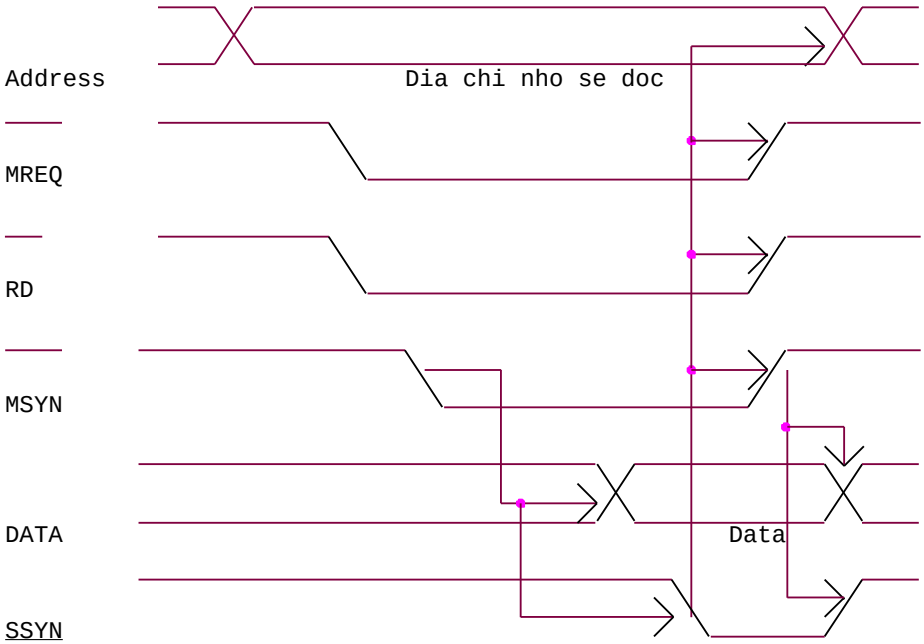
1.3. Bus bất đồng bộ(Asynchronous bus)

Bus bất đồng bộ không sử dụng xung clock đồng bộ, chu kỳ của nó có thể kéo dài tuỳ ý và có thể khác nhau đối với các cặp thiết bị khác nhau. Làm việc với các bus đồng bộ dễ dàng hơn do nó được định thời một cách gián đoạn , tuy vậy chính đặc điểm này cũng dẫn đên nhược điểm. Mọi công việc được tiến hành trong khoảng thời gian là bội số của xung clock, nếu 1 thao tác nào đó của vi xử lý hay bộ nhớ hoàn thành trong 3.1 chu kỳ thì nó cũng sẽ phải kéo dài trong 4 chu kỳ. Khi đã chọn chu kỳ bus và đã xây dựng bộ nhớ, I/O card cho bus này thì khó có thể tận dụng những tiến bộ của công nghệ. Chẳng hạn sau khi đã xây bus với sự định thời như trên, công nghệ mới đưa ra các vi xử lý và bộ nhớ có thời gian chu kỳ là 100ns chứ không còn là 750ns như cũ, thì chúng vẫn chạy với tốc đô thấp như các vi xử lý, bô nhớ loai cũ, bởi vì giao thức bus đòi hỏi bộ nhớ phải đưa được dữ liệu ra và ổn định trước thời điểm cạnh âm của T3. Nếu có nhiều thiết bị khác

nhau cùng nối với 1 bus, trong đó có thể có một số thiết bị hoạt động nhanh hơn hơn các thiết bị khác thì cần phải đặt bus hoạt động phù hợp với thiết bị có tốc độ thấp nhất.

Bus bất đồng bộ ra đời nhằm khắc phục những nhược điểm của bus đồng bộ. Trước hết master phát ra địa chỉ nhớ mà nó muốn truy cập, sau đó phát tín hiệu MREQ tích cực để xác định cần truy xuất bộ nhớ. Tín hiệu này cần thiết khi bộ nhớ và các cổng I/O sử dụng chung miền địa chỉ. Sau khi phát địa chỉ, bên master cũng phải phát tín hiệu RD tích cực để bên slave biết rằng master sẽ thực hiện thao tác đọc chứ không phải ghi.

Các tín hiệu MREQ và RD được đưa ra sau tín hiệu địa chỉ một khoảng thời gian phụ thuộc tốc độ hoạt động của master. Sau khi 2 tín hiệu này đã ổn định, master sẽ phát ra tín hiệu MSYN (master synchrization) ở mức tích cực để báo cho slave biết rằng các tín hiệu cần thiết đã sẵn sàng trên bus, slave có thể nhận lấy. Khi slave nhận được tín hiệu này, nó sẽ thực hiện công việc với tốc độ nhanh nhất có thể được, đưa dữ liệu của ô nhớ được yêu cầu lên bus dữ liệu. Khi hoàn thành slave sẽ phát tín hiệu SSYN (slave synchronization) tích cực.



Hình 4.3 - Chu kỳ đọc của bus bất đồng bộ

Master nhận được tín hiệu SSYN tích cực thì xác định được dữ liệu của slave đã sẵn sàng nên thực hiện việc chốt dữ liệu, sau đó đảo các đường địa chỉ cũng như các tín hiệu MREQ , RD và MSYN . Khi slave nhận được tín hiệu MSYN không tích cực, nó xác định kết thúc chu kỳ và đảo tín hiệu SSYN làm bus trở lại trạng thái ban đầu, mọi tín hiệu đều không tích cực, chờ bus master mới.

Trên giản đồ thời gian của bus bất đồng bộ, ta sử dụng mũi tên để thể hiện nguyên nhân và kết quả. MSYN tích cực dẫn đến việc truyền dữ liệu ra bus dữ liệu và đồng thời

cũng dẫn đến việc slave phát ra tín hiệu SSYN tích cực, đến lượt mình tín hiệu SSYN lại gây ra sự đảo mức của các đường địa chỉ, MREQ, RD và MSYN. Cuối cùng sự đảo mức của MSYN lại gây ra sự đảo mức tín hiệu SSYN và kết thúc chu kỳ.

Tập các tín hiệu phối hợp với nhau như vậy được gọi là bắt tay toàn phần (full handshake), chủ yếu gồm 4 tín hiệu sau:

- MSYN tích cực.
- SSYN tích cực để đáp lại tín hiệu MSYN.
- MSYN được đảo để đáp lại tín hiệu SSYN (tích cực).
- SSYN được đảo để đáp lại tín hiệu MSYN không tích cực.

Ta có thể nhận thấy bắt tay toàn phần là độc lập thời gian, mỗi sự kiện được gây ra bởi 1 sự kiện trước đó chứ không phải bởi xung clock. Nếu 1 cặp master-slave nào đó hoạt động chậm thì cặp master-slave kế tiếp không hề bị ảnh hưởng.

Tuy ưu điểm của bus bất đồng bộ rất rõ ràng, nhưng trong thực tế phần lớn các bus đang sử dụng là loại đồng bộ. Nguyên nhân là các hệ thống sử dụng bus đồng bộ dễ thiết kế hơn. Vi xử lý chỉ cần chuyển các mức tín hiệu cần thiết sang trạng thái tích cực là bộ nhớ đáp ứng ngay, không cần tín hiệu phản hồi. Chỉ cần các chọn phù hợp thì mọi hoạt động đều trôi chảy, không cần phải bắt tay.

1.4. Phân xử bus (bus arbitration)

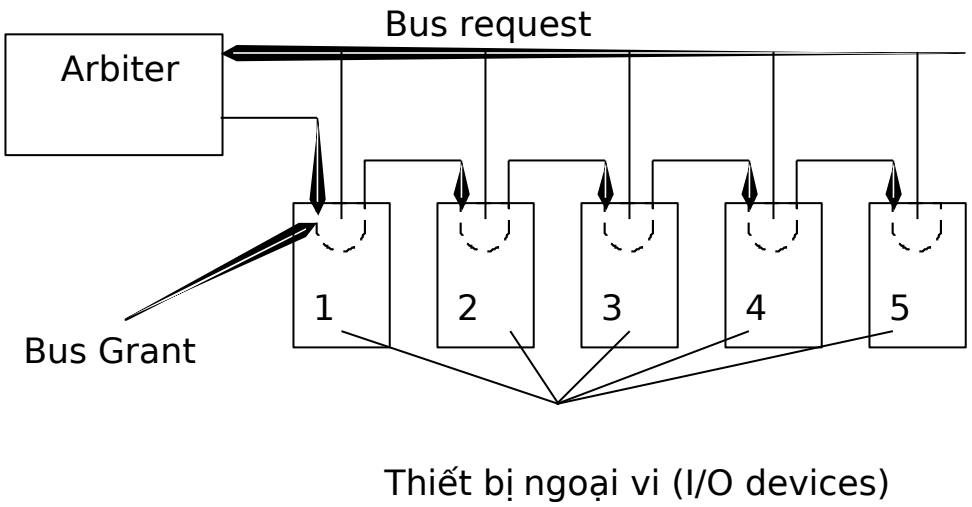
Trong hệ thống máy tính không phải chỉ có CPU làm bus master, các chip I/O cũng có lúc làm bus master để có thể đọc hay ghi bộ nhớ và gọi ngắt. Các bộ đồng xử lý cũng có thể làm bus master. Như vậy nảy sinh ra vấn đề: điều gì sẽ xảy ra khi 2 thiết bị trở lên đồng thời cần làm bus master? Từ đó cần có một cơ chế phân xử để tránh sự hỗn loạn của hệ thống. Cơ chế phân xử có thể là tập trung hay không tập trung.

1.4.1. Phân xử bus tập trung

Nhiều vi xử lý có đơn vị phân xử được chế tạo nằm ngay trong chip CPU, trong một số máy tính mini, đơn vị này nằm ngoài chíp CPU. Theo cơ chế này thì bộ phân xử (arbiter) chỉ có thể biết có yêu cầu chiếm dụng bus hay không mà không biết có bao nhiêu đơn vị muốn chiếm dụng bus. Khi arbiter nhận được yêu cầu, nó sẽ phát ra 1 tín hiệu cho phép trên đường dây (bus grant: cho phép sử dụng bus). Đường dây này nối qua tất cả các thiết bị I/O theo kiểu nối tiếp.

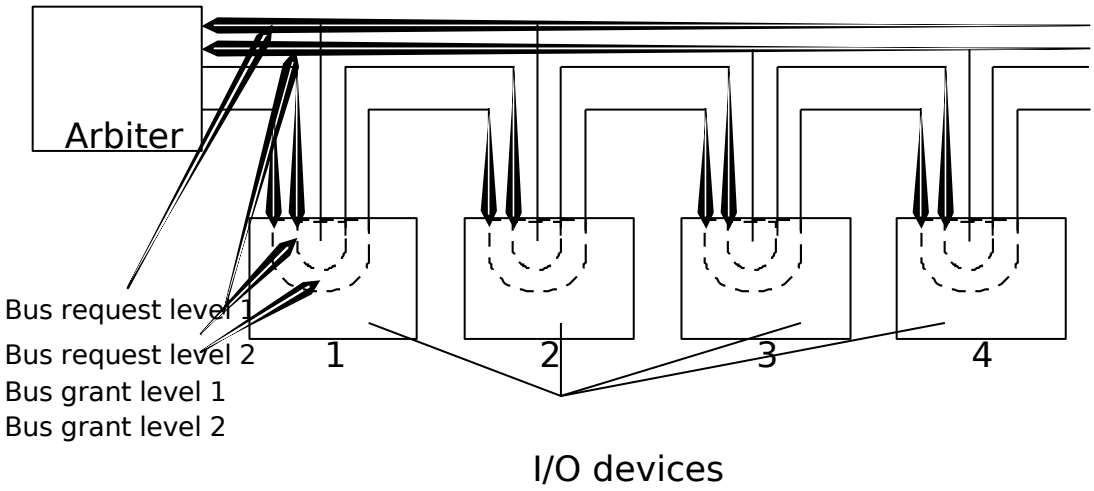
Khi thiết bị nằm gần arbiter nhất nhận được tín hiệu cho phép, nó kiểm tra xem có phải chính nó đã phát ra yêu cầu hay không. Nếu có thì nó sẽ chiếm lấy bus và không truyền tiếp tín hiệu cho phép trên đường dây. Nếu không thì nó sẽ truyền tín hiệu cho phép tới thiết bị kế tiếp trên đường dây, với thiết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

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



Hình 4.4 – Phân xử bus tập trung 1 mức nối tiếp

Một số loại bus có nhiều mức độ ưu tiên, với mỗi mức độ ưu tiên có đường yêu cầu bus (bus request) và đường dây cho phép bus (bus grant). Ví dụ: giả sử 1 bus có 2 mức ưu tiên 1 và 2 (các bus thực tế có 4, 8 hay 16 mức). Mỗi thiết bị trong hệ thống máy tính nối với 1 trong các mức yêu cầu bus, các đường thường được sử dụng nhiều hơn được gắn với đường dây có mức ưu tiên cao hơn. Ở ví du, các thiết bị 1, 2 sử dụng mức ưu tiên 1, còn các thiết bị 3, 4 sử dụng mức ưu tiên 2.



Hình 4.5 – Phân xử bus tập trung 2 mức

Nếu có một số thiết bị ở các mức ưu tiên khác nhau cùng yêu cầu, arbiter chỉ phát ra tín hiệu grant đối với yêu cầu có mức ưu tiên cao nhất. Trong số các thiết bị có cùng mức ưu tiên, thiết bị nào gần arbiter hơn sẽ ưu tiên hơn.

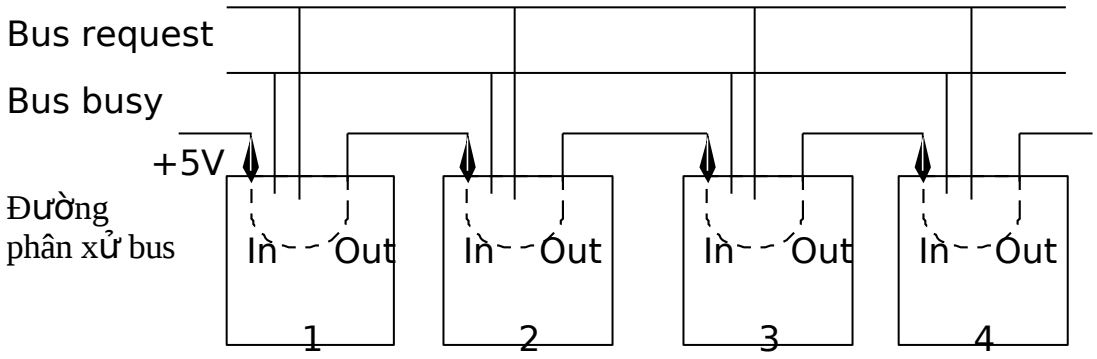
Về mặt kỹ thuật, không cần nối đường grant level 2 giữa các thiết bị vì chúng không bao giờ đòi hỏi bus ở mức 2. Tuy nhiên, trong thực tế để thuận tiện cho việc lắp đặt ~~người ta hay làm như sau: nối tất cả các đường grant thông qua tất cả các thiết bị, như vậy~~

sẽ dể dàng hơn là nối các đường grant một cách riêng biệt, và từ đó căn cứ vào thiết bị nào có quyền ưu tiên cao hơn.

Một arbiter có đường dây thứ 3 nối tới các thiết bị để các thiết bị xác nhận đã nhận được tín hiệu grant và chiếm dụng bus – đường ACK (acknowledgement). Ngay sau 1 thiết bị phát tín hiệu tích cực trên đường dây ACK, có thể đảo tín hiệu trên các đường dây request và grant xuống mức không tích cực. Các thiết bị khác có thể yêu cầu bus khi thiết bị đầu tiên đang dùng bus. Khi sự truyền thông kết thúc, bus master kế tiếp sẽ được lựa chọn. Cách làm việc như vậy làm tăng hiệu quả sử dụng bus, nhưng cần thêm 1 đường truyền tín hiệu và cấu trúc thiết bị cũng phức tạp hơn. Các chip trong máy tính PDP-11 và các chip Motorola làm việc với các bus như vậy.

1.4.2. Phân xử bus không tập trung:

Trong Multibus, người ta cho phép có thể lựa chọn sử dụng phân xử bus tập trung hay không tập trung, cơ chế phân xử bus không tập trung được thực hiện như sau:



Hình 4.6 – Phân xử bus không tập trung trong multibus

Cơ chế sử dụng 3 đường dây, không phụ thuộc vào số lượng thiết bị nối với bus:

- Bus request: yêu cầu chiếm dụng bus.
- Bus busy: đường báo bận, được bus master đặt ở mức tích cực khi có thiết bị đang chiếm dụng bus
- Bus arbitration: được mắc nối tiếp thành 1 chuỗi xích qua tất cả các thiết bị ngoại vi. Đầu của chuỗi này được gắn với mức điện áp 5V của nguồn nuôi.

Khi không có đơn vị nào yêu cầu chiếm dụng bus, đường dây phân xử bus truyền mức tích cực tới tất cả các thiết bị trong chuỗi xích. Khi 1 đơn vị nào đó muốn chiếm dụng bus, đầu tiên nó kiểm tra xem bus có rảnh không và đầu vào In của đường trọng tài bus có mức tích cực hay không. Nếu không (not active) thì nó không trở thành bus master. Ngược lại, nó sẽ đảo đầu Out thành không tích cực, làm cho các thiết bị đứng sau nó trong chuỗi xích có đầu In không tích cực.

Khi trạng thái có thể hiểu lầm (khoảng thời gian tín hiệu trên đầu In và Out đang thay đổi) qua đi, chỉ còn lại duy nhất 1 thiết bị có đầu In tích cực và Out không tích cực.

Thiết bị này trở thành bus master, nó sẽ đặt bus busy tích cực và bắt đầu truyền thông tin trên bus.

1.5. Xử lý ngắt

Ở trên, ta chỉ khảo sát các chu kỳ bus thông thường, trong đó master nhận hay gởi thông tin từ / đến slave. Một ứng dụng quan trọng nữa của bus là dùng để xử lý ngắt. Khi CPU ra lệnh cho thiết bị I/O làm một việc gì đó, nó thường chờ đợi tín hiệu ngắt do thiết bị I/O phát ra khi hoàn thành công việc được CPU yêu cầu. Khi nhận được tín hiệu ngắt, CPU sẽ đáp ứng ngay, có thể nhận dữ liệu do thiết bị I/O truyền về, hay gởi tiếp dữ liệu ra thiết bị I/O, hay CPU sẽ sử dụng bus cho một thao tác khác…. Như vậy chính ngắt phát ra tín hiệu yêu cầu sử dụng bus.

Vì có thể nhiều thiết bị ngoại vi cùng phát ra ngắt, cho nên cần có 1 cơ chế phân xử giống như đối với các bus thông thường. Giải pháp thường dùng là gán các mức độ ưu tiên cho các thiết bị và sử dụng 1 arbiter tập trung để trao quyền ưu tiên cho các thiết bị quan trọng thường xuyên được sử dụng. Hiện trên thị trường có những chip điều khiển ngắt được tiêu chuẩn hóa và được sử dụng rộng rãi. Ví dụ: Các máy IBM PC, PC/AT, PS/2 và cả dòng máy tương thích với IBM PC đều sử dụng chip 8259A.

Có thể nối 8 chip điều khiển I/O tới các đầu IRx (Interrupt request) của 8259A. Khi có 1 thiết bị nào đó muốn ngắt, nó đặt mức tích cực lên chân Irx, 8259A nhận được tín hiệu tích cực ở 1 hay một số đầu vào Irx thì sẽ đặt mức tích cực lên đầu dây INT. Tín hiệu INT sẽ truyền trực tiếp đến chân Interrupt của CPU. Khi CPU có thể xử lý được ngắt, nó gởi lại 1 tín hiệu chấp nhận ngắt cho 8259A. Lúc này, CPU chờ 8259A chỉ ra I/O nào yêu cầu ngắt, bằng cách gởi số hiệu của I/O đó lên bus dữ liệu (D0-D7) để đi đến CPU. Sau đó, phần cứng CPU sẽ sử dụng con số đó để tính chỉ số trong 1 bảng con trỏ - bảng vector ngắt (interrupt vector) để tìm địa chỉ chương trình con, cho chạy chương trình này để phục vụ ngắt. Các chương trình con này gọi là chương trình con xử lý ngắt.

2. Bus mở rộng (Expansion bus)

Bus mở rộng cho phép PC liên lạc được với các thiết bị ngoại vi, các thiết bị này được cài đặt qua các khe cắm mở rộng (expansion slot). Các thông số chính của bus mở rộng: tốc độ truyền tối đa giữa các thiết bị với nhau và giữa các thiết bị với bộ nhớ chính, số đường địa chỉ (số lượng ô nhớ có thể được truy xuất bởi 1 thiết bị), số đường ngắt cứng, ….

2.1. Bus ISA (Industry Standard Architecture)

Bus ISA dùng cho hệ thống chỉ được điều khiển bởi 1 CPU trên bảng mạch chính, tức là tất cả các chương trình và thiết bị đều chỉ được điều khiển bởi CPU đó. Tần số làm việc cực đại là 8.33 MHz (tốc độ chuyển tải cực đại là 16.66 MBps với số liệu 2 bytes). Bề rộng dữ liệu là 8 hay 16 bits. ISA có 24 đường địa chỉ nên quản lý được 16 MB bộ nhớ. Bus ISA tương thích 90% với bus AT.

2.2. Bus EISA và MCA

Sử dụng cho các CPU 32 bits (số liệu và đường địa chỉ) từ 80386 trở đi.

2.2.1. Bus EISA (Extended ISA)

Đây là chuẩn mở rộng của ISA để bố trí các dữ liệu 32 bits nhưng vẫn giữ được sự tương thích với mạch nối ghép cũ. Bus EISA có 2 nấc, các tín hiệu ISA được gửi qua nấc trên và các tín hiệu phụ trợ EISA qua nấc dưới. Các đặc trưng của EISA như sau:

- Về mặt cơ khí: có nhiều chân cắm hơn nhưng vẫn tương thích với ISA.
- Độ rộng dữ liệu: có thể truy xuất 2 đường 8 bits (tương thích với ISA), hay 2 đường 16 bits. Do đó, đơn vị quản lý bus 32 bits có thể chuyển tải 4 byte với bộ nhớ hoặc thiết bị ngoại vi. Điều này góp phần tăng tốc độ truyền tải lên khoảng 33 MBps so với 16.66 MBps của ISA.
- Độ rộng địa chỉ: ngoài 24 đường như ISA còn thêm 8 đường bổ sung nữa, do đó có thể định địa chỉ trong 4 GB bộ nhớ.
- Phần cứng được thiết kế theo hệ thống EISA phức tạp hơn ISA vì nó cũng phải thực hiện các chu kỳ bus tương thích với ISA. EISA có thể thực hiện phân xử bus, nó cho phép vi xử lý nằm ngoài bảng mạch chính có thể điều khiển toàn bộ bus. Điều này rất hiệu quả trong các hệ thống đa xử lý (multiprocessor). Hãng Intel đã phát triển 4 chip điện tử phục vụ cho bus EISA như sau:
    - ISP (Intergrated system peripheral)
    - BMIC (Bus master interface controller)
    - EBC (EISA bus controller)
    - EBB (EISA bus buffer)

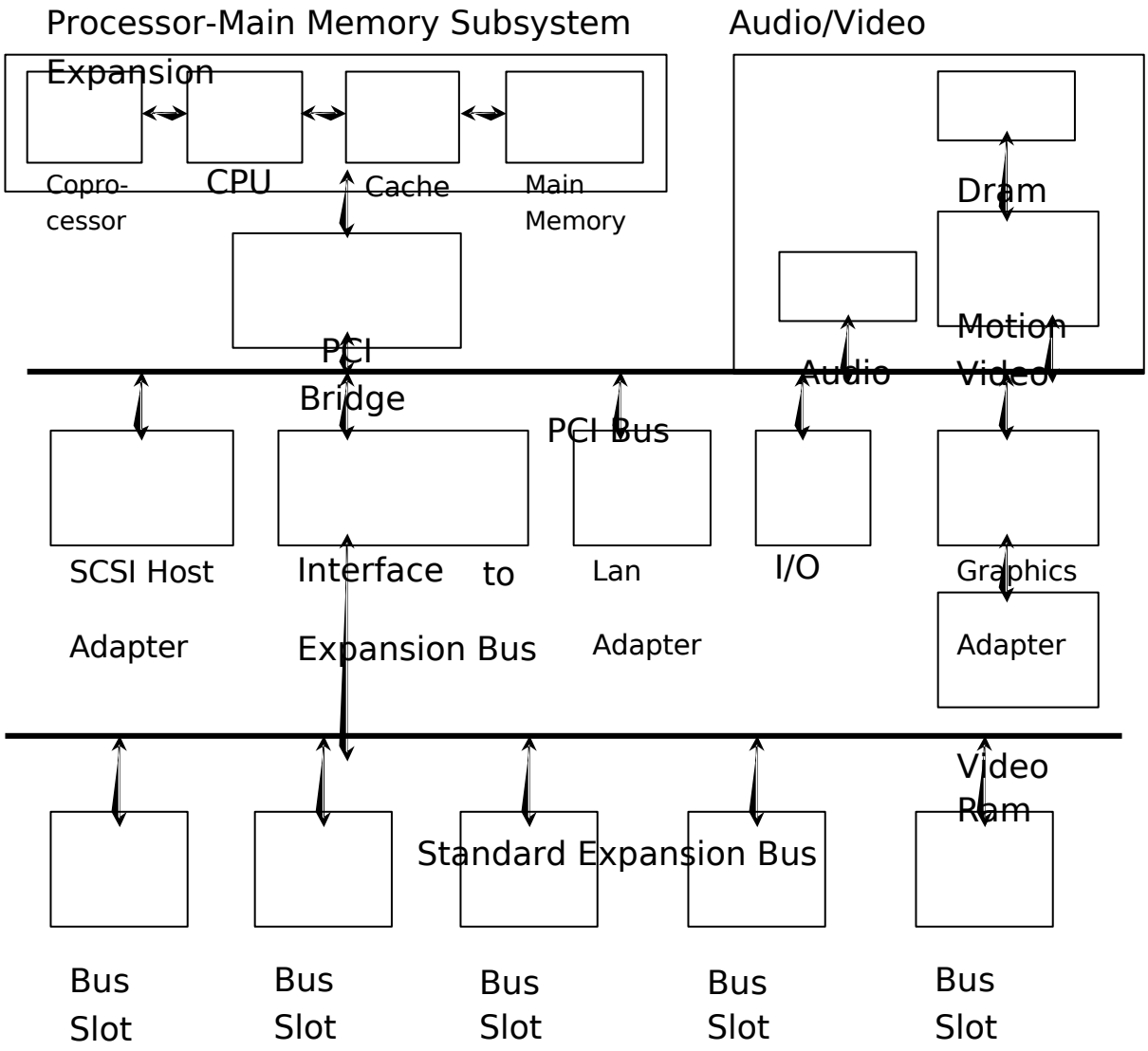
2.2.2. Bus MCA (Micro Channel Architecture)

Phục vụ cho hệ thống IBM PS/2 không tương thích với bus ISA, có thể hoạt động với 16 hay 32 bits dữ liệu. Nó có nhiều đường dẫn hơn ISA, thiết kế phức tạp cho phép giảm bớt các nhiễu cao tần của PC tới các thiết bị xung quanh. Tốc độ truyền dữ liệu có thể lên tới 160 MBps.

2.3. Bus cục bộ (Local Bus)

Nhược điểm của các bus chuẩn trên là mặc dù xung clock của CPU rất cao nhưng cũng chỉ làm việc với các ngoại vi với tốc độ truyền tải không quá 33MBps. Điều này không thể đáp ứng được tốc độ của các card đồ hoạ cắm vào khe cắm của bus mở rộng trong chế độ đồ họa. Chuẩn các *bus cục bộ* tạo thêm *các khe cắm mở rộng* nối trực tiếp vào bus cục bộ (bus nối giữa CPU và các bộ đệm). Do vậy, bus mở rộng loại này cho phép truy xuất lên trên 32 bit cũng như tận dụng được tốc độ xung clock của chính CPU, tránh được rào cản 8.33MHz của bus hệ thống. Theo hướng giải quyết này, Intel đã phát triển bus PCI và Uỷ ban VESA (Video Electronics Standards Association) đã phát triển bus VL.

2.3.1. Bus PCI (Peripheral Component Interconnect)



Hình 4.7 - Sơ đồ bus PCI

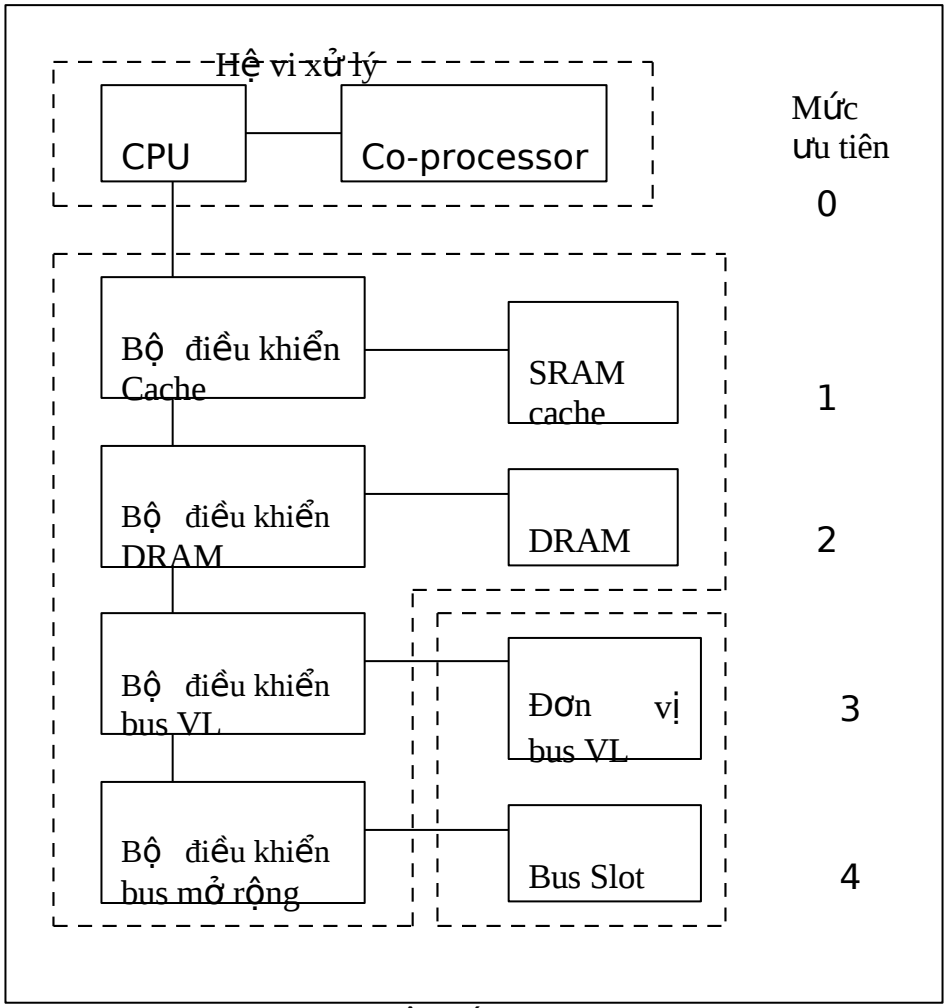
Bus PCI là bus của i486 trong đó dữ liệu và địa chỉ được gởi đi theo cách thức dồn kênh (multiplexing), các đường địa chỉ và dữ liệu được dồn chung trên các đường của PCI. Cách này tiết kiệm được số chân PCI nhưng lại hạn chế tốc độ vì phải cần 2 xung clock cho một quá trình truyền dữ liệu (1 cho địa chỉ và 1 cho dữ liệu). Việc nối giữa CPU, bộ nhớ chính, và bus PCI được thực hiện bằng *cầu PCI* (PCI bridge), qua đó bus PC sẽ phục vụ cho tất cả các đơn vị của bus PCI. Tối đa là 10 thiết bị có thể được nối tới bus PCI, trong đó cầu PCI được coi là một. Chu kỳ bus của PCI đạt gần bằng tốc độ chu kỳ bus của i486. Nó có thể hoạt động với độ rộng 32 bits dữ liệu và tốc độ 33MHz (có thể đạt 64 bits với tốc độ 66 MHz). Một điểm mạnh của PCI là dữ liệu được truyền tải theo kiểu cụm (burst), trong đó địa chỉ chỉ truyền đi 1 lần, sau đó nó sẽ được hiểu ngầm bằng cách cho các đơn vị phát hoặc thu đếm lên trong mỗi xung clock. Do đó, bus PCI hầu như được lấp đầy bởi dữ liệu. Tốc độ truyền tối đa trong kiểu burst có thể lên đến 120MBps.

2.3.2. Bus VL (VESA local bus)

Giống như PCI, bus VL cũng phân cách giữa hệ CPU, bộ nhớ chính, và bus mở rộng chuẩn. Thông qua bus cục bộ trên board mạch chính, nó có thể để điều khiển tối đa 3

thiết bị ngoại vi. Khe cắm VL có 116 tiếp điểm. Bus VL chạy với xung clock bên ngoài CPU, như vậy trong các máy DX2 thì tần số này chỉ bằng một nửa clock CPU. Về mặt logic, mỗi một thiết bị có thể ở một trong hai vị trí: LMB (Local bus master) hoặc LBT (Local bus target). Bộ phận điều khiển bus cục bộ LBC (local bus controller) trên main board sẽ quyết định thiết bị nào sẽ trở thành LMB , tức là được nắm quyền điều khiển bus và cho phép nhường quyền đó cho thiết bị có quyền ưu tiên cao hơn. Thường có 3 cấp ưu tiên được sắp xếp theo thứ tự giảm dần như sau: DMA/làm tươi, CPU/đơn vị làm chủ bus (bus master) và các đơn vị làm chủ bus khác. Thiết bị nào ở vị trí LBT thì không có khả năng làm các việc liên quan đến chuyển tải dữ liệu.

Bus VL chỉ làm việc với 32 bits, trong tương lai sẽ được mở rộng đến 64 bits.



Hình 4.8 – Bus VL

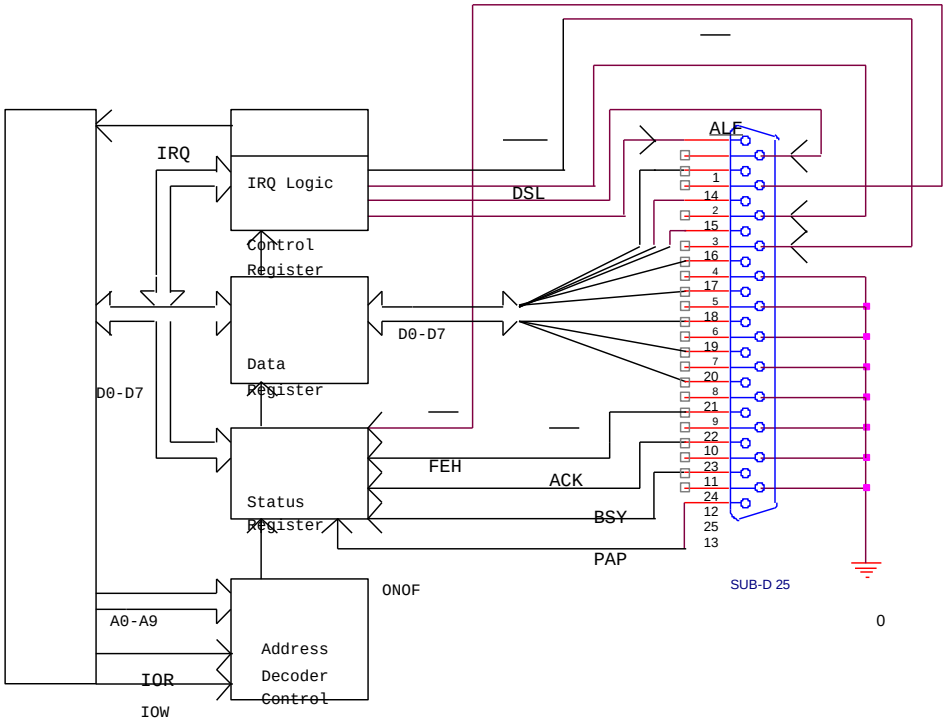
Chương 5

TỔ CHỨC VÀO/RA

1. Các kiểu giao tiếp

1.1. Song song (Parallel)

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 5.1 - 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:

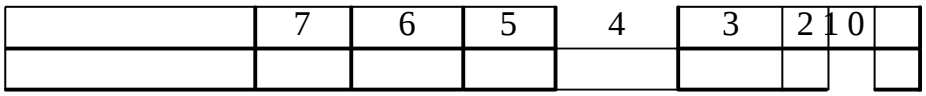
Thanh ghi dữ liệu (hai chiều):

	7	6	5	4	3	2	1	0
Tín hiệu máy in	D7	D6	D5	D4	D3	D2	D1	D0
Chân số	9	8	7	6	5	4	3	2

INI

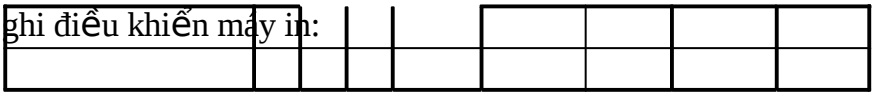
PC Interface

Thanh ghi trạng thái máy in (chỉ đọc):



Tín hiệu máy in BSY /ACK PAP OFON /FEH x x x
Số chân cắm 11 10 12 13 15 - - -

Thanh ghi điều khiển máy in:



7 6 5 4 3 2 1 0

Tín hiệu máy in x x x IRQ /DSL /INI /ALF STR
Số chân cắm - - - - 17 16 14 1

x: không sử dụng

IRQ: yêu cầu ngắt cứng; 1 = cho phép; 0 = không cho phép

Bản mạch ghép nối chỉ có bus dữ liệu 8 bit do dữ liệu luôn đi qua máy in thành từng khối 8 bit. Các chân tín hiệu của đầu cắm 25 chân của cổng song song LPT như sau:

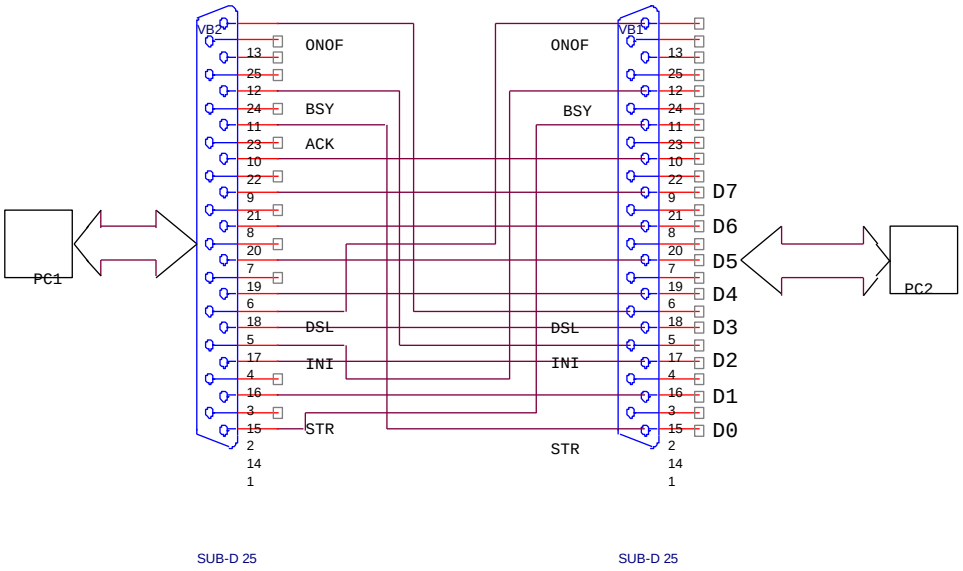
Chân	Tín hiệu	Mô tả
1	STR	Mức tín hiệu thấp, truyền dữ liệu tới máy in
2	D0	Bit dữ liệu 0
3	D1	Bit dữ liệu 1
4	D2	Bit dữ liệu 2
5	D3	Bit dữ liệu 3
6	D4	Bit dữ liệu 4
7	D5	Bit dữ liệu 5
8	D6	Bit dữ liệu 6
9	D7	Bit dữ liệu 7
10	ACK	Mức thấp: máy in đã nhận 1 ký tự và có khả năng nhận nữa
11	BSY	Mức cao: ký tự đã được nhận; bộ đệm máy in đầy; khởi động máy in; máy in ở trạng thái off-line.
12	PAP	Mức cao: hết giấy
13	OFON	Mức cao: máy in ở trạng thái online
14	ALF	Tự động xuống dòng; mức thấp: máy in xuống dòng tự động
15	FEH	Mức thấp: hết giấy; máy in ở offline; lỗi máy in
16	INI	Mức thấp: khởi động máy in
17	DSL	Mức thấp: chọn máy in
18-25		GROUND

0V

Thường tốc độ xử lý dữ liệu của các thiết bị ngoại vi như máy in chậm hơn PC nhiều nên các đường ACK , BSY và STR được sử dụng cho kỹ thuật bắt tay. Khởi đầu,

PC đặt dữ liệu lên bus sau đó kích hoạt đường STR xuống mức thấp để thông tin cho máy in biết rằng số liệu đã ổn định trên bus. Khi máy in xử lý xong dữ liệu, nó sẽ trả lại tín hiệu *ACK* xuống mức thấp để ghi nhận. PC đợi cho đến khi đường BSY từ máy in xuống thấp (máy in không bận) thì sẽ đưa tiếp dữ liệu lên bus.

Dữ liệu có thể trao đổi trực tiếp giữa 2 PC qua các cổng song song với nhau. Muốn vậy, các đường điều khiển bên này phải được kết nối với các đường trạng thái bên kia.



Hình 5.2 - Trao đổi dữ liệu qua cổng song song giữa 2 PC

Máy in có thể được truy xuất bằng chương trình qua DOS, BIOS hay trực tiếp qua các cổng. Các lệnh như "copy tên_file << PRN" trong DOS cho phép in 1 file ra máy in. Ngắt 17h với hàm 01h khởi động máy in, 00h in 1 ký tự ra máy in, 02h trả về trạng thái của máy in,… có sẵn trong BIOS.

1.2. Nối tiếp (Serial)

1.2.1. Truyền nối tiếp đồng bộ và bất đồng bộ

Ghép nối tiếp cho phép trao đổi giữa các thiết bị từng bit một. Dữ liệu thường được gửi theo các nhóm bit SDU (serial data unit) mà mỗi nhóm tạo thành 1 byte hay 1 word. Các thiết bị ngoại vi như: máy vẽ, modem, chuột có thể được nối với PC qua cổng nối tiếp COM.

Sự khác nhau giữa truyền nối tiếp đồng bộ và bất đồng bộ là: trong kỹ thuật truyền đồng bộ, ngoài đường dây dữ liệu phải đưa thêm vào một hoặc vài đường tín hiệu đồng bộ để cho biết rằng khi nào bit tiếp theo ổn định trên đường truyền. Ngược lại trong truyền bất đồng bộ, các bit dữ liệu tự nó chứa các thông tin để đồng bộ; phần phát và phần thu phải hoạt động với cùng 1 tần số xung clock. Thông tin đồng bộ (trong truyền bất đồng bộ) gồm có các bit start (cho biết bắt đầu của khối dữ liệu được truyền) và một bit stop (cho biết kết thúc khối dữ liệu).

1.2.2. Kiểm tra chẵn lẻ và tốc độ truyền

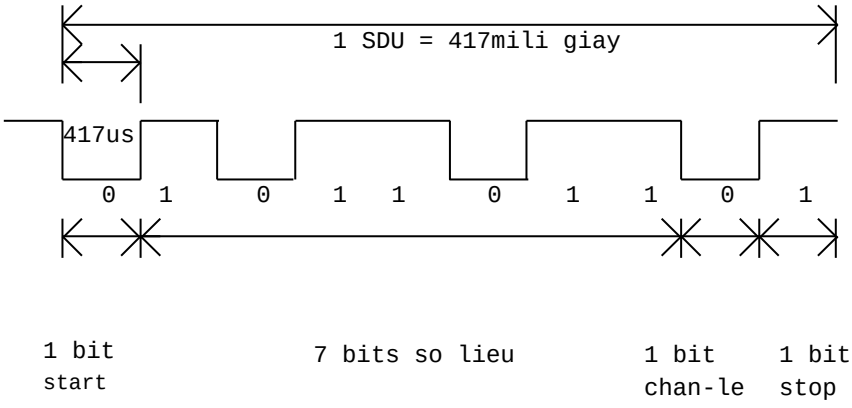
Bit chẵn lẻ (parity bit) được đưa vào khung SDU dùng để phát hiện lỗi trên đường truyền. Việc truyền bit chẵn lẻ chỉ kiểm soát được các lỗi trên đường truyền ngắn và các

lỗi bit đơn nên trong một số ứng dụng đặc biệt người ta phải dùng mã CRC mặc dù phức tạp hơn. Tuy nhiên, gần như tất cả các chip hỗ trợ cho ghép nối nối tiếp ngày nay đều được thiết kế phần cứng kiểm tra chẵn lẻ.

Một thông số khác liên quan tới truyền dữ liệu nối tiếp là tốc độ truyền dữ liệu được gọi là tốc độ baud. Trong việc truyền mã nhị phân, đó là số bit được truyền trong một giây (bps - bit per second).

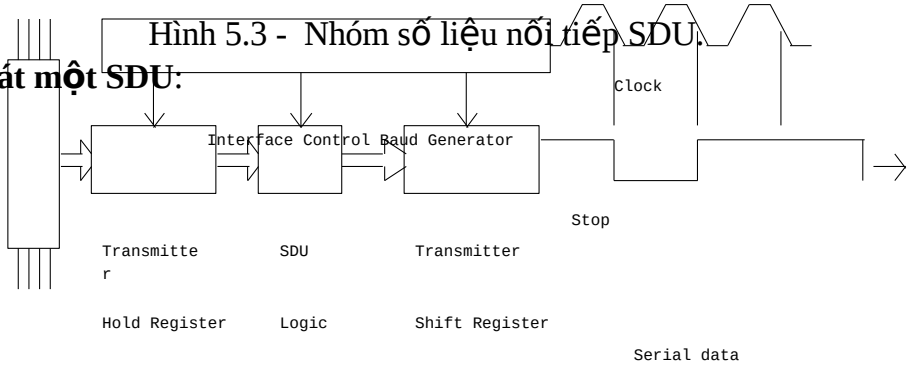
1.2.3. Nhóm dữ liệu nối tiếp SDU và nối tiếp hóa

Trước khi truyền chuỗi số liệu nối tiếp, máy phát và máy thu phải được khởi tạo để hoạt động với cùng một định dạng dữ liệu, cùng một tốc độ truyền. Một SDU với 1 bit start, 7 bits số liệu, 1 bit chẵn lẻ và 1 bit stop mô tả như hình vẽ. Lưu ý rằng: bit start luôn bằng 0 (space) và bit stop luôn bằng 1 (mark).

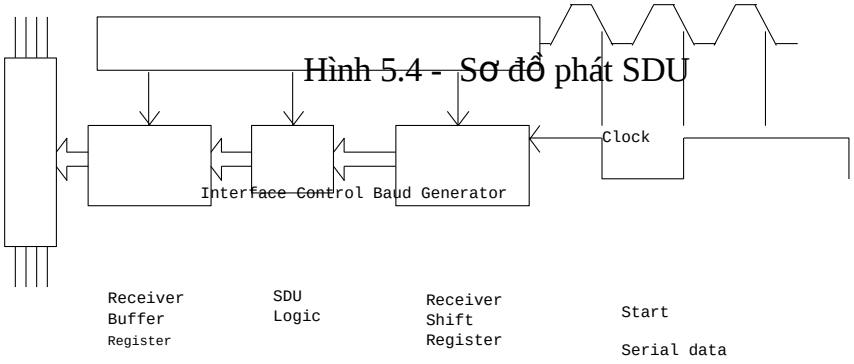


Hình 5.3 - Nhóm số liệu nối tiếp SDU.

Thu, phát một SDU:



Hình 5.4 - Sơ đồ phát SDU



Hình 5.5 - Sơ đồ thu SDU

Bus Interface

Bus Interface

Bus interface: ghép nối bus;
Serial data: dữ liệu nối tiếp;
Transmitter holder register: thanh ghi đệm giữ dữ liệu phát;
Transmitter shift register: thanh ghi dịch dữ liệu phát;
Receiver buffer register: thanh ghi đệm dữ liệu thu;
Receiver shift register: thanh ghi dịch dữ liệu thu;
SDU logic: mạch logic SDU;

Interface control baud generator: máy phát điều khiển tốc độ truyền dữ liệu baud;
Clock: xung clock;

1.2.4. Chuẩn ghép nối RS-232

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: dữ liệu phát
3	2	RxD	DCE→DTE	Received data: dữ liệu thu
4	7	RTS	DTE→DCE	Request to send: DTE yêu cầu truyền dữ liệu
5	8	CTS	DCE→DTE	Clear to send: DCE sẵn sàng nhận dữ liệu
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 độ truyền

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.

1.2.5. Truy xuất cổng nối tiếp dùng DOS và BIOS

Lệnh ngoại trú MODE của DOS có thể đặt các thông số cho cổng nối tiếp RS232.

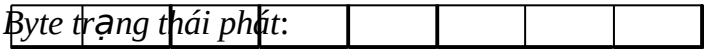
Thí dụ: *MODE COM2:2400, E,8 ,1* chọn cổng COM2, tốc độ 2400 baud, parity chẵn, 8 bit dữ liệu và 1 bit stop.

Cũng có thể dùng ngắt 21h của DOS để phát hoặc thu dữ liệu qua cổng nối tiếp bằng 4 hàm sau:

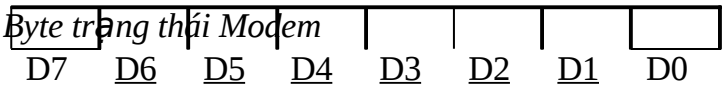
- Hàm 03h: đọc 1 ký tự
- Hàm 04h: phát 1 ký tự
- Hàm 3Fh: đọc 1 file
- Hàm 40h: ghi 1 file

BIOS cho phép truy xuất khối ghép nối nối tiếp qua ngắt 14h.

- Hàm 00h: khởi động khối ghép nối, định dạng dữ liệu, tốc độ truyền,….
- Hàm 01h, 02h: phát và thu 1 ký tự
- Hàm 03h: trạng thái của cổng nối tiếp
- Hàm 04h,05h: mở rộng các điều kiện khởi động khối ghép nối, cho phép truy xuất các thanh ghi điều khiển MODEM.

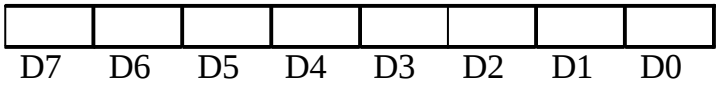


D7: lỗi quá thời gian (time-out); 1 = có lỗi; 0 = không lỗi;
D6: thanh ghi dịch phát; 1 = rỗng ; 0 = không rỗng
D5: thanh ghi đệm phát; 1 = rỗng; 0 = không rỗng
D4: ngắt đường truyền; 1= có ; 0 = không
D3: lỗi khung truyền SDU; 1 = có ; 0 = không
D2: lỗi chẵn lẻ; 1 = có ; 0 = không
D1: lỗi tràn; 1 = có ; 0 = không
D0: số liệu thu; 1 = có ; 0 = không



D7: phát hiện sóng mang; 1= phát hiện, 0 = không
D6: chỉ báo tín hiệu chuông; 1= có ; 0= không
D5: tín hiệu DTR; 1 = có , 0 = không
D4: tín hiệu CTS; 1 = có , 0 = không
D3: tín hiệu DDC, 1 = có , 0 = không
D2: tín hiệu delta RI; 1 = có, 0 = không
D1: tín hiệu delta DTR; 1 = có , 0 = không
D0: tín hiệu delta CTS; 1 = có , 0 = không

Thanh ghi DX chứa giá trị tương ứng với các cổng cần truy xuất (00h cho COM1, 01h cho COM2, 10h cho COM3, 11h cho COM4). Các thông số định dạng khung truyền SDU được nạp vào thanh ghi AL theo nội dung như sau:



D7, D6, D5: tốc độ baud
- 000 = 110 baud
- 001 = 150 baud
- 010 = 300 baud
- 011 = 600 baud
- 100 =1200 baud
- 101 = 2400 baud
- 110 = 4800 baud
- 111 = 9600 baud

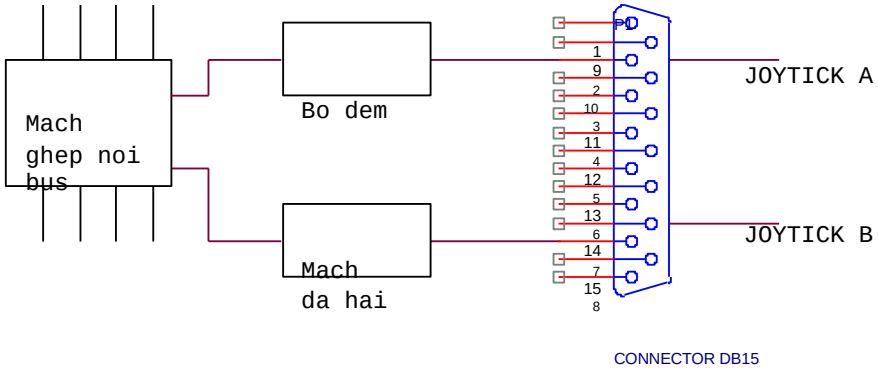
D4-D3: bit parity
- 00= không có
- 01= lẻ
- 10 = không có
- 11= chẵn

D2: số bit stop
- 0 = 1 bit
- 1 = 2 bits

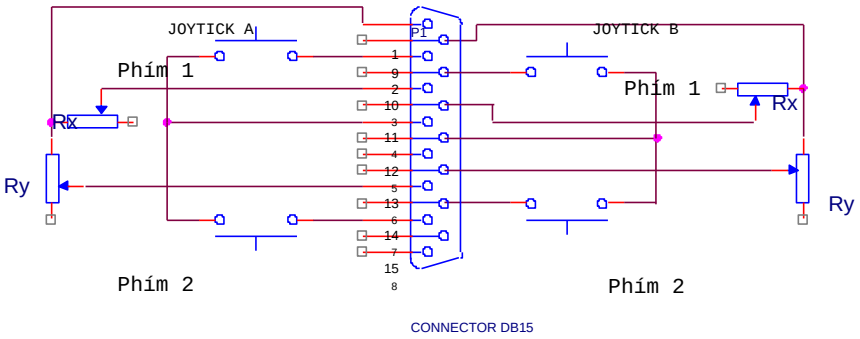
D1-D0: số bit số liệu
- 10 = 7 bits
- 11= 8 bits

2. Giao tiếp PC Game

Cấu trúc và chức năng của board ghép nối trò chơi (PC game) như hình bên dưới. Bằng lệnh IN và OUT có thể truy xuất qua địa chỉ 201h.



Hình 5.6 - Cấu trúc và chức năng của board ghép nối trò chơi



Hình 5.7 - Cấu tạo của đầu nối 15 chân và 2 joystick A và B

Chân của đầu nối 15 chân	Sử dụng cho
2	Phím 1 của Joystick A (BA1)
3	Biến trở X của Joystick A
6	Biến trở Y của Joystick A
7	Phím 2 của Joystick A (BA2)
10	Phím 1 của Joystick A (BB1)
11	Biến trở X của Joystick B
13	Biến trở Y của Joystick B
14	Phím 2 của Joystick A (BB2)
1, 8, 9, 15	Vcc (+5V)
4, 5, 12	GND (0V)

Board mạch được nối với bus hệ thống của PC chỉ qua 8 bits thấp của bus dữ liệu, 10 bits thấp của bus địa chỉ và các đường điều khiển IOR và IOW . Một đầu nối 15 chân được nối với board mạch cho phép nối cực đại hai thiết bị cho PC game gọi là joystick. Mỗi joystick có 2 biến trở có giá trị biến đổi từ 0 đến 100kΩ được đặt vuông góc với nhau đại diện cho vị trí x và y của joystick. Thêm nữa chúng có 2 phím bấm, thường là các công tắc thường hở phù hợp với các mức logic cao của các dây trên mạch.

Có thể xác định được trạng thái nhấn hoặc nhả phím một cách dễ dàng bằng lệnh IN tới địa chỉ 201h. Nibble cao chỉ thị trạng thái của phím. Vì board không dùng đường IRQ do đó không có khả năng phát ra 1 ngắt, do vậy board chỉ hoạt động trong chế độ hỏi vòng (polling). Byte trạng thái của board game như sau:

D7	D6	D5	D4	D3	D2	D1	D0
BB_2	BB_1	BA_2	BA_1	BY	BX	AY	AX

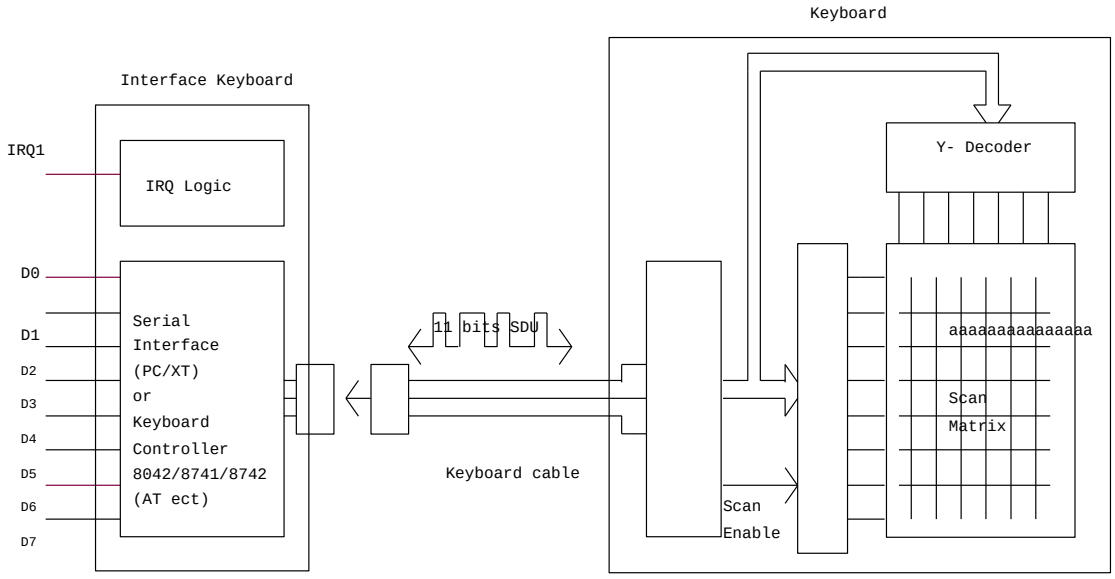
BB_2, BB_1, BA_2, BA_1: Trạng thái của các phím B_2, B_1, A_2, A_1; 1 = nhả; 0 = nhấn

BY, BX, AY, AX: Trạng thái của mạch đa hài tuỳ thuộc vào biến trở tương ứng.

3. Giao tiếp với bàn phím và mouse

3.1. Bàn phím

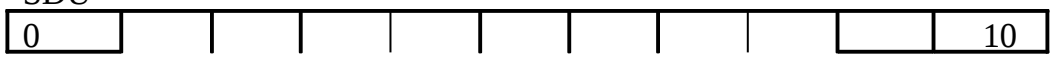
3.1.1. Cấu trúc và chức năng:



Hình 5.8 - Sơ đồ nguyên lý và các ghép nối của 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



STRT $DB_0DB_1DB_2DB_3DB_4DB_5DB_6DB_7$PAR STOP

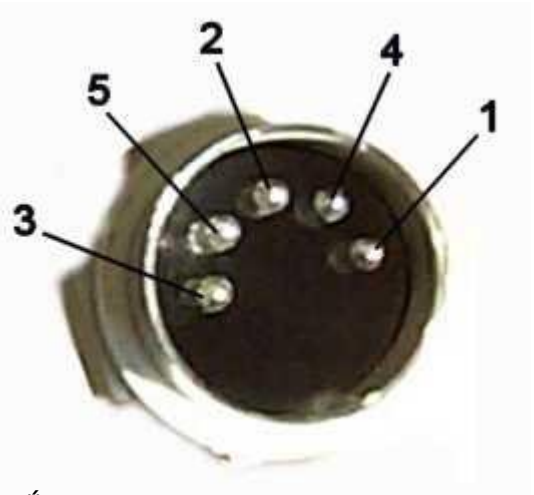
STRT: bit start (luôn bằng 0)
DB_0 - DB_7: bit số liệu từ 0 đến 7.
PAR: bit parity (luôn lẻ)
STOP: bit stop (luôn bằng 1).

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,….

Keyboard chip

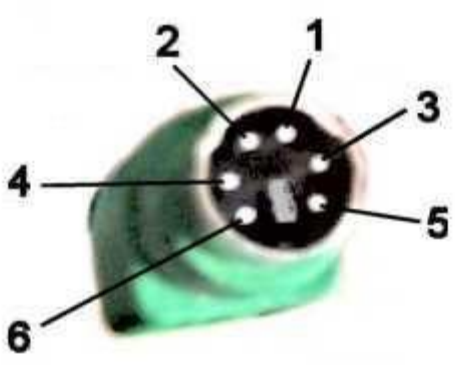
X - Decoder

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



Hình 5.9 – 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 5.10 – Đầu cắm bàn phím PS/2

3.1.2. Mã quét bàn phím:

Mỗi phím nhấn sẽ được gán cho 1 mã quét (scan code) gồm 1 byte. Nếu 1 phím được nhấn thì bàn phím phát ra 1 mã make code tương ứng với mã quét truyền tới mạch ghép nối bàn phím của PC. Ngắt cứng INT 09h được phát ra qua IRQ1.

Chương trình xử lý ngắt sẽ xử lý mã này tuỳ theo phím SHIFT có được nhấn hay không. Ví dụ: nhấn phím SHIFT trước, không rời tay và sau đó nhấn ‘C’:

make code được truyền - 42(SHIFT) - 46 (‘C’).

Nếu rời tay nhấn phím SHIFT thì bàn phím sẽ phát ra break code và mã này được truyền như make code. Mã này giống như mã quét nhưng bit 7 được đặt lên 1, do vậy nó tương đương với make code công với 128. Tuỳ theo break code, chương trình con xử lý ngắt sẽ xác định trạng thái nhấn hay rời của các phím. Thí dụ, phím SHIFT và ‘C’ được rời theo thứ tự ngược lại với thí dụ trên:

break code được truyền 174 (bằng 46 cộng 128 tương ứng với ‘C’) và 170 (bằng 42 cộng 128 tương ứng với SHIFT).

Phần cứng và phần mềm xử lý bàn phím còn giải quyết các vấn đề vật lý sau:

- Nhấn và nhả phím nhưng không được phát hiện.

- Khử nhiễu rung cơ khí và phân biệt 1 phím được nhấn nhiều lần hay được nhấn chỉ 1 lần nhưng được giữ trong một khoảng thời gian dài.

3.1.3. Truy xuất bàn phím qua BIOS

BIOS ghi các ký tự do việc nhấn các phím vào bộ đệm tạm thời được gọi là bộ đệm bàn phím (keyboard buffer), có địa chỉ 40:1E, gồm 32 byte và do vậy kết thúc ở địa chỉ 40:3D. Mỗi ký tự được lưu trữ bằng 2 bytes, byte cao là mã quét, và byte thấp là mã ASCII. Như vậy, bộ đệm có thể lưu trữ tạm thời 16 ký tự. Chương trình xử lý ngắt sẽ xác định mã ASCII từ mã quét bằng bảng biến đổi và ghi cả 2 mã vào bộ đệm bàn phím. Bộ đệm bàn phím được tổ chức như bộ đệm vòng (ring buffer) và được quản lý bởi 2 con trỏ. Các giá trị con trỏ được lưu trữ trong vùng số liệu của BIOS ở địa chỉ 40:1A và 40:1C.

Ngắt INT 16h trong BIOS cung cấp 8 hàm cho bàn phím. Thường các hàm BIOS trả về một giá trị 0 của ASCII nếu phím điều khiển hoặc chức năng được nhấn..

Các thí dụ:

- Giả sử phím ‘a’ đã được nhấn.

```
MOV AH,00h          ; chạy hàm 00h, đọc ký tự
INT 16h             ; phát một interrupt
```

Kết quả: AH = 30 (mã quét cho phím ‘a’); AL = 97 (ASSCII cho ‘a’)

- Giả sử phím ‘.HOME’ đã được nhấn.

```
MOV AH,00h          ; chạy hàm 00h, đọc ký tự
INT 16h             ; phát một interrupt
```

Kết quả: AH = 71 (mã quét cho phím ‘HOME’)
AL = 00 (các phím chức năng và điều khiển không có mã ASCII)

- Giả sử phím ‘HOME’ đã được nhấn.

```
MOV AH,10h          ; chạy hàm 10h, đọc ký tự
INT 16h             ; phát một interrupt
```

Kết quả: AH = 71 (mã quét cho phím ‘HOME’)
AL = E0h

3.1.4. Chương trình với bàn phím qua các cổng:

Bàn phím cũng là một thiết bị ngoại vi nên về nguyên tắc có thể truy xuất nó qua các cổng vào ra.

- ***Các thanh ghi và các port:***

Sử dụng 2 địa chỉ port 60h và 64h có thể truy xuất bộ đệm vào, bộ đệm ra và thanh ghi điều khiển của bàn phím.

Port	Thanh ghi
60h	Đệm ngõ ra
60h	Đệm ngõ vào
64h	Thanh

R/W
R
W
W
R

Thanh ghi trạng thái xác định trạng thái hiện tại của bộ điều khiển bàn phím. Thanh ghi này chỉ đọc (read only). Có thể đọc nó bằng lệnh IN tại port 64h.

7							0
PARE	TIM	AUXB	KEYL	C/D	SYSF	INPB	OUTB

PARE: Lỗi chẵn lẻ của byte cuối cùng được vào từ bàn phím; 1 = có lỗi chẵn lẻ, 0 = không có.

TIM: Lỗi quá thời gian (time-out); 1 = có lỗi, 0 = không có.

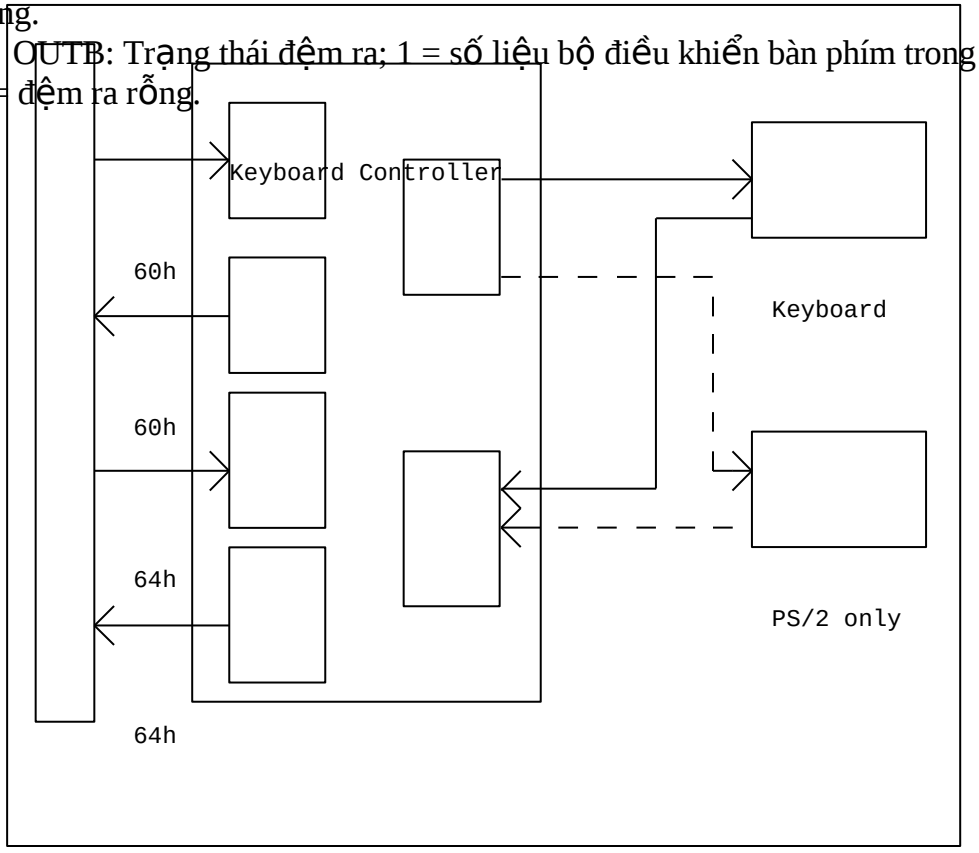
AUXB: Đệm ra cho thiết bị phụ (chỉ có ở máy PS/2); 1 = giữ số liệu cho thiết bị, 0 = giữ số liệu cho bàn phím.

KEYL: Trạng thái khóa bàn phím; 1 = không khóa, 0 = khóa.

C/D: Lệnh/số liệu; 1 = Ghi qua port 64h, 0 = Ghi qua port 60h.

INPB: Trạng thái đệm vào; 1 = số liệu CPU trong bộ đệm vào, 0 = đệm vào rỗng.

OUTB: Trạng thái đệm ra; 1 = số liệu bộ điều khiển bàn phím trong bộ đệm ra, 0 = đệm ra rỗng.



Hình 5.11 - Bộ điều khiển bàn phím

Thanh ghi điều khiển (64h)

7 0

C_7 C_6 C_5 C_4 C_3 C_2 C_1 C_0

Các lệnh cho bộ điều khiển bàn phím:

Mã	**Lệnh**
A7h	Cấm thiết bị phụ
A8h	Cho phép thiết bị phụ
A9h	Kiểm tra ghép nối tới thiết bị phụ
AAh	Tự kiểm tra
ABh	Kiểm tra ghép nối bàn phím
ADh	Cấm bàn phím
AEh	Cho phép bàn phím
C0h	Đọc cổng vào
C1h	Đọc cổng vào ra (byte thấp)
C2h	Đọc cổng vào ra (byte cao)
D0h	Đọc cổng ra
D1h	Ghi cổng ra
D2h	Ghi đệm ra bàn phím
D3h	Ghi đệm ra thiết bị phụ
D4h	Ghi thiết bị phụ
E0h	Kiểm tra đọc cổng vào
F0h	Gửi 1 xung tới lối ra
FFh	Cổng

Khóa bàn phím:

```
Start:
        IN AL, 64h  ; đọc byte trạng thái
TEST AL, 02h        ; kiểm tra bộ đệm có đầy hay không
JNZ start           ; một vài byte vẫn còn trong bộ đệm vào
OUT 64h, 0ADh       ; khóa bàn phím
```

- ***Các lệnh cho bàn phím:***

Tóm tắt các lệnh bàn phím:

Mã	**Lệnh**	**Mô tả**
EDh	Bật ON/OFF LED	Bật/tắt các đèn led của bàn phím
EEh	Echo	Trả về byte eeh
F0h	Đặt/nhận diện	Đặt 1 trong 3 mã quét và nhận diện các mã quét tập mã quét hiện tại.
F2h	Nhận diện bàn phím	Nhận diện ACK = AT, ACK+abh+41h=MF II.
F3h	Đặt tốc độ lặp lại/trễ	Đặt tốc độ lặp lại và thời gian trễ của bàn phím
F4h	Enable	Cho phép bàn phím hoạt động
F5h	Chuẩn/không cho phép	Đặt giá trị chuẩn và cấm bàn phím.
F6h	Chuẩn/cho phép	
FEh	Resend	
FFh	Reset	

Đặt giá trị chuẩn và cho phép bàn phím.
Bàn phím truyền ký tự cuối cùng một lần nữa tới bộ điều khiển bàn phím
Chạy reset bên trong bàn phím

Thí dụ: lệnh bật đèn led cho phím NUMCLOCK, tắt tất cả các đèn khác.

```
OUT  60H, EDH        ; ra lệnh cho bật tắt các đèn led
WAIT:
IN  AL, 64H          ; đọc thanh ghi trạng thái
JNZ  WAIT            ; bộ đệm vào đầy
OUT  60H, 02H        ; bật đèn cho numclock
```

Cấu trúc của byte chỉ thị như sau:

7					2	1	0
0	0	0	0	0	CPL	NUM	SCR

CPL: 1 = bật đèn Caps Lock; 0 = tắt

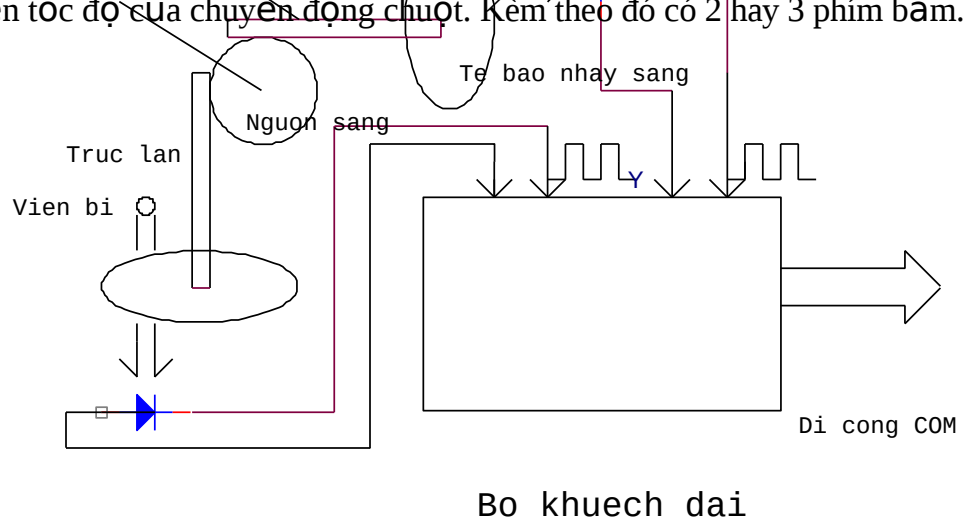
NUM: 1 = bật đèn Num Lock; 0 = tắt

SCR: 1 = bật đèn Scr Lock; 0 = tắt

3.2. Chuột

3.2.1. Cấu tạo

Cấu tạo của chuột rất đơn giản, phần trung tâm là 1 viên bi thép được phủ keo hoặc nhựa được quay khi dịch chuyển chuột. Chuyển động này được truyền tới 2 thanh nhỏ được đặt vuông góc với nhau. Các thanh này sẽ biến chuyển động của chuột theo 2 hướng X,Y thành sự quay tương ứng của 2 đĩa gắn với chúng. Trên 2 đĩa có những lỗ nhỏ liên tục đóng và ngắt 2 chùm sáng tới các sensor nhạy sáng để tạo ra các xung điện. Số các xung điện tỷ lệ với lượng chuyển động của chuột theo các hướng X,Y và số xung trên 1 sec biểu hiện tốc độ của chuyển động chuột. Kèm theo đó có 2 hay 3 phím bấm.



Hình 5.12 - Sơ đồ cấu tạo của chuột

3.2.2. Mạch ghép nối và chương trình điều khiển chuột

Hầu hết chuột được nối với PC qua cổng nối tiếp, qua đó chuột cũng được cấp nguồn nuôi từ PC. Khi dịch chuyển hoặc nhấn, nhả các phím chuột, nó sẽ phát ra một gói các số liệu tới mạch giao tiếp và mạch sẽ phát ra 1 ngắt. Phần mềm điều khiển chuột làm các nhiệm vụ: chuyển ngắt tới mạch giao tiếp nối tiếp xác định, đọc gói số liệu và cập nhật các giá trị bên trong liên quan tới trạng thái của bàn phím cũng như vị trí của chuột. Hơn nữa, nó còn cung cấp 1 giao tiếp mềm qua ngắt của chuột là 33h để định các giá trị bên trong này cũng như làm dịch chuyển con trỏ chuột trên màn hình tương ứng với vị trí của chuột.

Có thể chọn kiểu con trỏ chuột cứng hoặc mềm trong chế độ văn bản hay con trỏ chuột đồ hoạ trong chế độ đồ họa. Các hàm 09h và 0Ah trong ngắt 33h cho phép định nghĩa loại và dạng con trỏ chuột.

3.2.3. Chương trình với con trỏ

Ngắt 33h cho phép xác định vị trí, số lần bấm nháy (click) phím con trỏ và hình dạng con trỏ. Để hiện con trỏ trên màn hình phải dùng hàm 01h. Hàm 09h định nghĩa con trỏ chuột trong chế độ đồ họa . Hình bên dưới là thí dụ của mặt nạ con trỏ hình mũi tên trong trong kiểu hiện VGA phân dải cao 256 màu. Ở đây một điểm ảnh (pixel) được biểu diễn bằng 1 byte.

```
f . . . . . . . . . . . . . . . .
f  f . . . . . . . . . . . . . .
f x f . . . . . . . . . . . . . .
f x x f . . . . . . . . . . . . .
f x x x f . . . . . . . . . . . .
f x x x x f . . . . . . . . . . .
f x x x x x f . . . . . . . . . .
f x x x x x x f . . . . . . . . .
f x x x x f f f f . . . . . . . .
.f . f f x f . . . . . . . . . .
f f x f x f . . . . . . . . . . .
f . . f x x f . . . . . . . . . .
f . . . f x f . . . . . . . . . .
. . . . . f x x f . . . . . . . .
. . . . . . f x f . . . . . . . .
. . . . . . . f . . . . . . . . .
```

Hình 5.13 - Mặt nạ con trỏ chuột

Đoạn chương trình sau cho phép hiện con trỏ mềm với màu số 3 và sáng nhấp nháy:

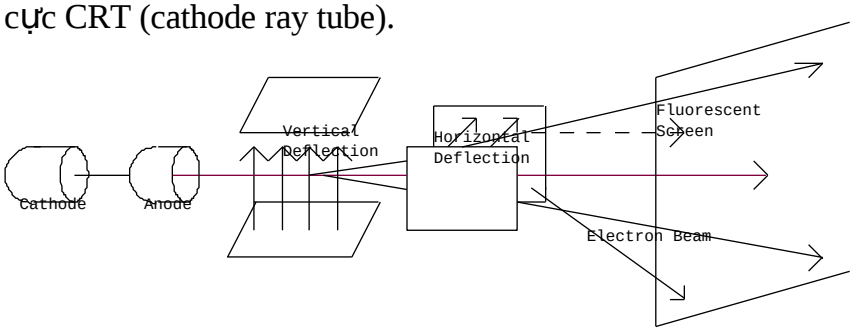
```
MOV AX, 0AH      ; chọn hàm
MOV BX, 00H      ; con trỏ chuột mềm
MOV CX, 00H      ; xoá ký tự trên màn hình

MOV DX, 8B02H ;BLNK=1b,BAK = 000b, INT = 1b, CHRx = 00000010b
INT 33H          ; gọi ngắt.
```

4. Monitor và card giao diện đồ hoạ

4.1. Nguyên lý hiện ảnh trên monitor

Phương pháp hiện ảnh trên màn hình của monitor máy tính cũng giống như trong máy thu hình thông thường. Hình bên dưới minh họa việc hiện ảnh trên màn hình kiểu ống phóng tia âm cực CRT (cathode ray tube).



Hình 5.14 – Cấu tạo ống hình CRT

Các điện tử phát xạ từ cathode trong ống được hội tụ thành 1 chùm tia, sau đó được tăng tốc và được làm lệch hướng chuyển động bởi các bộ phận lái tia. Tia này sẽ đập vào màn hình có phủ chất huỳnh quang để tạo thành 1 điểm sáng gọi là 1 điểm ảnh.

Do hiện tượng lưu ảnh trong võng mạc của mắt người nên khi tia điện tử được quét rất nhanh theo chiều ngang từ trái sang phải sẽ tạo nên 1 vệt sáng ngang được gọi là dòng quét. Đến cuối 1 dòng, nó được quét ngược trở về bên trái để quét tiếp dòng thứ 2 bên dưới ..v..v.. Quá trình quét các dòng được dịch dần từ trên xuống dưới cho suốt chiều dọc của màn hình được gọi là quét dọc.

Độ chói (sáng tối) được quyết định bởi cường độ chùm tia đập vào màn hình huỳnh quang và 1 điểm màu tự nhiên được hiện nhờ sự trộn lẫn của 3 màu: đỏ, xanh dương, xanh lá cây theo 1 tỉ lệ nào đó. Ba màu này được hiện nhờ 3 tia điện tử cùng bắn vào 3 điểm trên màn hình kề cận nhau, mỗi điểm được phủ chất huỳnh quang phát ra các màu tương ứng. 3 chùm tia điện tử đó được phát ra bởi 3 súng điện tử là 3 cathode được xếp đặt bên trong CRT một cách cẩn thận. Có 2 kiểu quét tia điện tử:

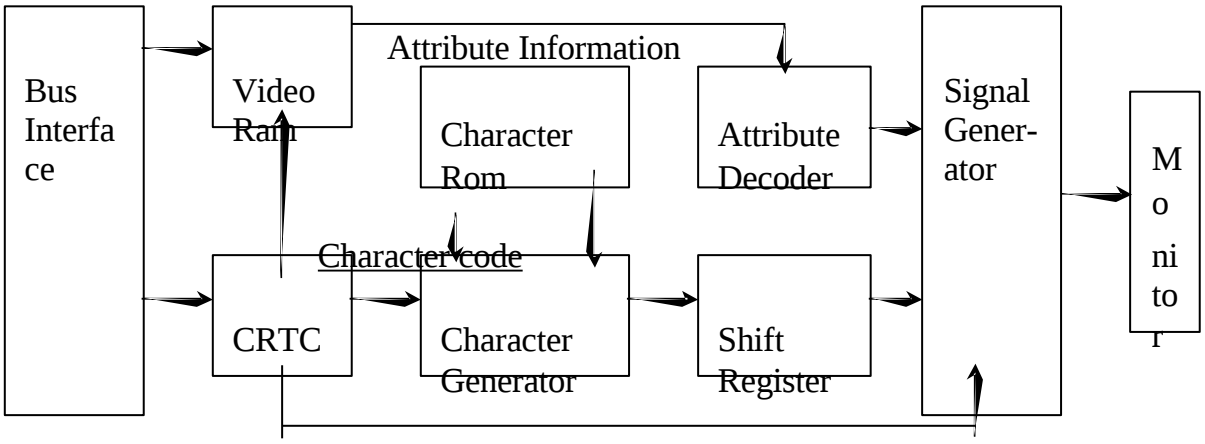
- Quét xen kẽ (interlaced): các dòng lẻ được quét trước cho đến hết màn hình theo chiều dọc, gọi là mành lẻ; sau đó các dòng chẵn tạo nên mành chẵn được quét sau. Phương pháp này có ưu điểm là thu hẹp được dải tần số làm việc của thiết bị nhưng có nhược điểm là hình ảnh bị nhấp nháy.

- Quét không xen kẽ (non-interlaced): các dòng quét được thực hiện tuần tự. Ưu điểm là hình ảnh được điều chỉnh chính xác và ổn định nhưng thiết kế mạch điện sẽ khó hơn vì phải giải quyết vấn đề tăng dải tần làm việc.

Hiện nay còn có các monitor dùng màn hình tinh thể lỏng LCD hoặc ống chứa khí được hoạt động theo nguyên lý tương tự như trên nhưng không có tia điện tử quét nên thay vì các điểm ảnh riêng biệt là các phần tử phát sáng được định địa chỉ một cách tuần tự. Do vậy, trên các monitor này hình ảnh cũng được phát ra từng dòng một. Quá trình quét ngược cũng không còn nữa vì ở đây đơn giản chỉ việc thay đổi địa chỉ về phần tử đầu dòng tiếp theo.

4.2. Card giao tiếp đồ họa

Để hiện các hình ảnh, ký tự, hay hình vẽ trên màn hình, PC phải thông qua mạch ghép nối màn hình (graphics adapter). Board mạch này thường được cắm trên khe cắm mở rộng của PC. Sơ đồ khối như hình sau:



Hình 5.15 - Sơ đồ khối của bản mạch ghép nối màn hình

Bus Interface: ghép nối bus; Video Ram: Ram Video
Signal generator: máy phát tín hiệu; Character code: mã ký tự

Attribute information: thông tin thuộc tính; Character rom: rom ký tự
Attribute decoder: bộ giải mã thuộc tính; Shift register: thanh ghi dịch

Character generator: máy phát ký tự; Synchronization information: thông tin đồng bộ.

Phần trung tâm là chip điều khiển ống hình CRTC (cathode ray tube controller). CPU thâm nhập RAM Video qua mạch ghép nối bus để ghi thông tin xác định ký tự hay hình vẽ cần hiển thị. CRTC liên tục phát ra các địa chỉ để Ram video đọc các ký tự trong đó và truyền chúng tới máy phát ký tự (character generator).

Trong chế độ văn bản (text mode), các ký tự được xác định bởi mã ASCII, trong

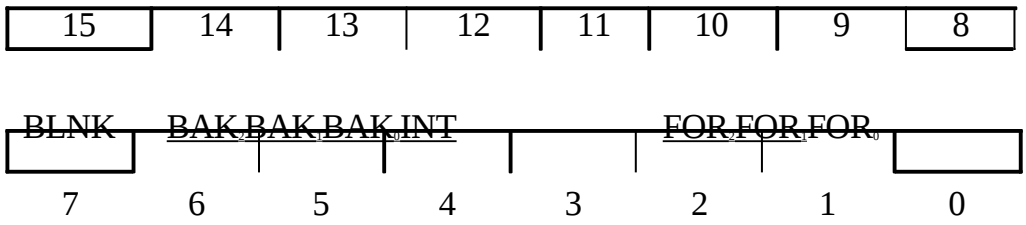
đó có cả các thông tin về thuộc tính của ký tự, thí dụ ký tự được hiện theo cách nhấp nháy hay đảo màu đen trắng ….ROM ký tự (character rom) lưu trữ các hình mẫu điểm ảnh của các ký tự tương ứng để máy phát ký tự biến đổi các mã ký tự đó thành 1 chuỗi các bit

điểm ảnh (pixel bit) và chuyển chúng tới thanh ghi dịch (shift register). Máy phát tín hiệu sẽ sử dụng các bít điểm ảnh này cùng với các thông tin thuộc tính từ Ram video và các tín hiệu đồng bộ từ CRTC để phát ra các tín hiệu cần thiết cho monitor.

Trong chế độ đồ họa (graphics mode), thông tin trong RAM video được sử dụng trực tiếp cho việc phát ra các ký tự. Lúc này các thông tin về thuộc tính cũng không cần nữa. Chỉ từ các giá trị bit trong thanh ghi dịch, máy phát tín hiệu sẽ phát các tín hiệu về độ sáng và màu cho monitor.

4.2.1. Máy phát ký tự trong các chế độ văn bản và đồ họa:

Mỗi ký tự được biểu diễn bởi 1 từ 2 byte trong RAM video. Byte thấp chứa mã ký tự, byte cao chứa thuộc tính. Cấu trúc của một từ nhớ video như sau:



$CHR_7 CHR_6 CHR_5 CHR_4 CHR_3 CHR_2 CHR_1 CHR_0$

BLNK: Nhấp nháy; 1 = bật, 0 = tắt

BAK_2 … BAK_0: Màu nền; (từ bảng màu hiện tại)
INT: Cường độ sáng ; 1 = cao, 0 = bình thường

FOR_2… FOR_0: Màu nền trước (từ bảng màu hiện tại)
CHR_7…CHR_0: Mã ký tự.

Trong chế độ văn bản, 6845 liên tục xuất các địa chỉ cho RAM video qua MA0-MA13. Ký tự ở góc tận cùng phía trên bên trái màn hình có địa chỉ thấp nhất mà 6845 sẽ cung cấp ngay sau khi quét dọc ngược. Logic ghép nối định địa chỉ cho RAM video bằng việc lấy ra mã ký tự cùng với thuộc tính. Mã ký tự dùng cho máy phát ký tự như là chỉ số thứ nhất trong ROM ký tự. Lúc này, 6845 định địa chỉ hàng quét đầu tiên của ma trận ký tự, địa chỉ hàng bằng 0. Các bit của ma trận điểm ảnh bây giờ sẽ được truyền đồng bộ với tần số video từ thanh ghi dịch tới máy phát tín hiệu. Nếu máy phát tín hiệu nhận được giá trị 1 từ thanh ghi dịch, nó sẽ phát tín hiệu video tương ứng với màu của ký tự. Nếu nhận được 0 nó sẽ cấp tín hiệu tương ứng với màu nền. Vậy dòng quét thứ nhất được hiện phù hợp với các ma trận điểm ảnh của các ký tự trong hàng ký tự thứ nhất. Khi tia điện tử đạt tới cuối dòng quét, 6845 kích hoạt lối ra HS để tạo ra quá trình quét ngược và đồng bộ ngang. Tia điện tử quay trở về bắt đầu quét dòng tiếp. Sau mỗi dòng quét, 6845 tăng giá trị RA0-RA4 lên 1. Địa chỉ dòng này hình thành một giá trị offset bên trong ma trận điểm ảnh cho ký tự được hiện. Dựa trên mỗi dòng quét như vậy, một dòng các điểm ảnh của ký tự trong hàng ký tự được hiện ra. Điều này có nghĩa là với ma trận 9x14 điểm ảnh cho 1 ký tự, hàng ký tự thứ nhất đã được hiện sau 14 dòng quét. Khi địa chỉ RA0-RA4 trở về giá trị 0, 6845 sẽ cấp 1 địa chỉ MA0-MA13 mới và hàng ký tự thứ hai sẽ được hiện ra cũng như vậy. Ở cuối dòng quét cuối cùng, 6845 sẽ reset địa chỉ MA0-MA13 và RA0-

RA4 và cho phép lối ra VS phát ra tín hiệu quét ngược cùng tín hiệu đồng bộ dọc.

Mỗi ký tự có chiều cao cực đại ứng với 32 dòng vì có 5 đường địa chỉ RA0-RA4, còn bộ nhớ video trong trường hợp này được tới 16K từ vì có địa chỉ MA0-MA13 là 14 bit. Trong chế độ đồ họa, chúng kết hợp với nhau để tạo thành địa chỉ 19 bit, lúc đó 6845 có thể định địa chỉ cho bộ nhớ video lên tới 512k từ. Trong trường hợp này, các byte trong RAM video không được dịch thành mã ký tự và thuộc tính nữa mà trực tiếp xác định cường độ sáng và màu của điểm ảnh. Đa số các RAM video được chia thành vài băng được định địa chỉ bởi RA0-RA4. Các đường MA0-MA13 sẽ định địa chỉ offset bên trong mỗi băng. Số liệu trong RAM video lúc này được trực tiếp truyền tới thanh ghi dịch và máy phát tín hiệu. ROM ký tự và máy phát ký tự không làm việc.

4.2.2. Tổ chức của RAM video

RAM video được tổ chức khác nhau tuỳ theo chế độ hoạt động và bản mạch ghép nối. Thí dụ, với RAM video 128 KB, có thể địa chỉ hóa toàn bộ bộ nhớ màn hình qua CPU như bộ nhớ chính. Nhưng nếu kích thước RAM video lớn hơn thì làm như vậy sẽ đè lên vùng ROM mở rộng ở địa chỉ C0000h. Do đó, card EGA và VGA với trên 128 KB nhớ được tăng cường thêm 1 chuyển mạch mềm (soft-switch) cho phép thâm nhập các cửa sổ 128 KB khác nhau vào RAM video lớn hơn nhiều. Các chuyển mạch này được quy định bởi riêng các nhà sản xuất board mạch.

4.2.2.1. Tổ chức trong chế độ văn bản

RAM video được coi như một dãy từ tuyến tính, từ đầu tiên được gán cho ký tự góc trên tận cùng bên trái màn hình gọi là hàng 1 cột 1. Từ thứ 2 là hàng 1, cột 2, …. Số từ tuỳ thuộc vào độ phân giải của kiểu hiện ký tự.

Thí dụ: độ phân giải chuẩn 25 hàng, 80 ký tự đòi hỏi 2000 từ nhớ 2 byte. Như vậy, tổng cộng cần 4 KB bộ nhớ RAM video. Trong khi đó với card có độ phân giải cao SVGA 60 hàng, 132 ký tự cần đến 15840 byte. Do đó RAM video thường được chia thành vài trang. Kích thước của mỗi trang tuỳ thuộc vào chế độ hiện của màn hình và số trang cực đại, phụ thuộc cả vào kích thước của RAM video. 6845 có thể được chương trình hóa sao cho địa chỉ khởi phát của MA0-MA13 sau quét ngược dọc là khác 00h. Nếu địa chỉ khởi phát là bắt đầu của 1 trang thì có thể quản lý RAM video theo vài trang tách biệt nhau, nếu CPU thay đổi nội dung của 1 trang mà trang đó hiện đang không hiện thì màn hình cũng không thay đổi. Do đó, cần phân biệt trang nhớ đang được kích hoạt (đang hiện) và trang đang được xử lý.

Đoạn chương trình ghi ký tự 'A' có cường độ sáng cao vào góc trên bên trái với màu số 7 và màu nền số 0. Trang thứ nhất và là duy nhất bắt đầu ở địa chỉ B0000h.

```
MOV AX, 0B000h;     nạp thanh ghi ax với địa chỉ đoạn của Ram video
MOV ES, AX;         truyền địa chỉ đoạn vào ES
MOV AH, 0F8h;       nạp byte thuộc tính 1111 1000 vào AH
MOV AL, 41h;        nạp mã ký tự của ‘A’ vào AL

MOV ES:[00H],AX;    ghi byte thuộc tính và mã ký tự vào RAM video.
```

4.2.2.2. Tổ chức trong chế độ đồ họa:

Tổ chức trong chế độ này phức tạp hơn. Ví dụ: với bản mạch Hercules, RAM video được chia thành 4 băng trên 1 trang . Băng thứ nhất: đảm bảo các điểm ảnh cho các dòng 0, 4, 8, …, 344; băng thứ hai cho các dòng 1, 5, 9, …, 345; băng thứ 3 cho các dòng 2, 6, 10, …., 346; và băng thứ 4 cho các dòng 3, 7, 11, …, 347. 64 KB được chia thành 2 trang 32 KB. Độ phân giải trong chế độ đồ họa là 720 x 348 điểm ảnh, mỗi điểm ảnh được biểu diễn bởi 1 bit. Do vậy, một dòng cần 90 byte (720 điểm ảnh / 8 điểm ảnh trên 1 byte). Địa chỉ của byte chứa điểm ảnh thuộc đường i và cột j trong trang k là:

B0000h+8000h*k+2000h*(i mod 4)+ 90*int(i/4)+int(j/8)

B0000h là đoạn video, 8000h là kích thước của trang, 2000h*(i mod 4) là offset của băng chứa byte đó, 90*int(i/4) là offset của dòng i trong băng và int(j/8) là offset của cột j trong băng.

Trong bản mạch CGA bộ nhớ video được chia thành 2 băng còn với EGA và VGA thì phức tạp hơn.

4.2.3. Truy xuất màn hình qua DOS và BIOS

4.2.3.1. Truy xuất qua DOS

Các hàm của int 21h có thể hiện các ký tự trên màn hình nhưng không can thiệp được vào màu:

- Hàm 02h: ra màn hình.
- Hàm 06h: ra một ký tự.
- Hàm 09h: ra một chuỗi.
- Hàm 40h: ghi file/ thiết bị

Từ DOS 4.0 trở đi có thể dùng lệnh **mode** để điều chỉnh số cột văn bản từ 40 đến 80 hay số dòng từ 25 đến 50.

Các lệnh **copy, type** và **print** trong command.com cho phép hiện text trên màn hình. DOS gộp chung bàn phím và monitor thành 1 thiết bị mang tên CON (console). Ghi CON là truyền số liệu tới monitor, còn đọc CON là nhận ký tự từ bàn phím. Ví dụ: để hiện nội dung của file output.txt lên màn hình của monitor sẽ có các cách sau:

- copy output.txt con
- type output.txt > con
- print output.txt /D:con

4.2.3.2. Truy xuất qua BIOS

Bios thâm nhập monitor bằng int 10h với nhiều chức năng hơn DOS, như đặt chế độ hiện hình, quản lý tự động các trang, phân biệt các điểm trên màn hình nhờ các tọa độ,…

- **Những thường trình đồ họa:**

BIOS trên main board có sẵn những hàm dùng cho thâm nhập MDA và CGA. BIOS của riêng EGA và VGA có những hàm mở rộng tương ứng trong khi vẫn giữ nguyên định dạng gọi.

Một trong những hàm quan trọng nhất của int 10h là hàm 00h dùng để đặt chế độ hiện hình. Để thay đổi chế độ hiện hình cần phải làm rất nhiều bước chương trình phức tạp để nạp các thanh ghi của chip 6845. Trong khi đó, hàm 00h làm cho ta tất cả các công việc này.

Thí dụ: tạo kiểu 6 với độ phân giải 640*200 trên CGA.

```
Mov ah, 00h      ; hàm 00h
Mov al, 06h      ; chế độ 6
Int 10h          ; gọi ngắt
```

Các board EGA/VGA có riêng BIOS của chúng. Trong quá trình khởi động PC, nó sẽ chặn int 10h lại và chạy chương trình BIOS của riêng board mạch. Thường trình cũ (của BIOS trên board mach chính) được thay địa chỉ tới int 42h. Tất cả các lệnh gọi int 10h sẽ được BIOS của EGA/VGA thay địa chỉ tới int 42h nếu board mạch EGA/VGA đang chạy các kiểu hiện tương thích với MDA hay CGA. Có các kiểu hoạt động từ 0 đến 7.

BIOS của EGA/VGA dùng vùng 40:84h tới 40:88h để lưu số liệu BIOS và các thông số của EGA/VGA. Nó có các hàm mới với các hàm phụ sau:

- Hàm 10h: truy xuất các thanh ghi màu và bảng màu
- Hàm 11h: cài đặt các bảng định nghĩa ký tự mới
- Hàm 12h: đặt cấu hình hệ con video
- Hàm 1Bh: thông tin về trạng thái và chức năng của BIOS video (chỉ có ở VGA)
- Hàm 1Ch: trạng thái save/restore của video (chỉ có ở VGA)

Sau đây là chức năng của các hàm và thí dụ sử dụng chúng:

- Hàm 10h, hàm phụ 03h – xoá/đặt thuộc tính

Ví dụ: Xoá thuộc tính nhấp nháy:

```
Mov ah, 10h      ; dùng hàm 10h
Mov al, 03h      ; dùng hàm phụ 03h
Mov bl, 00h      ; xoá thuộc tính nhấp nháy
Int 10h          ; gọi ngắt
```

- Hàm 11h – ghép nối với máy phát ký tự

Ví dụ: Nạp bảng định nghĩa ký tự 8*14 không cần chương trình CRTC:

```
Mov ah, 11h
Mov al, 01h
Mov bl, 03h
Int 10h
```

; dùng hàm 11h
; nạp bảng ký tự từ Rom Bios vào Ram
máy phát ký tự.
; gán số 3 cho bảng
; gọi ngắt

- Hàm 12h, hàm phụ 20h – chọn thường trình in màn hình. Dùng hàm phụ này có thể thay thế thường trình chuẩn cho INT 05h bằng thường trình có thể dùng cho các độ phân giải mới của EGA/VGA.

Ví dụ: Cho phép thường trình mới in màn hình:

```
Mov ah, 12h     ; dùng hàm 12h
Mov bl, 20h     ; dùng hàm phụ 20h
```

Ấn PRINT hoặc SHIFT+PRINT để gọi thường trình in đã được lắp đặt.

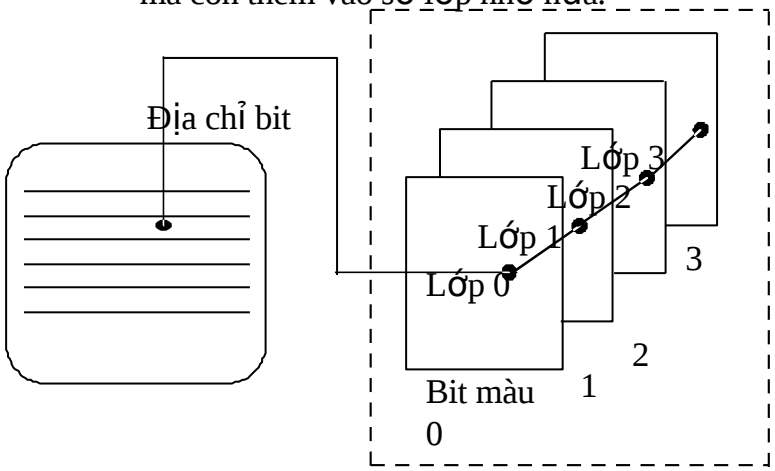
- **Truy xuất trực tiếp bộ nhớ video:**

Để vẽ 1 điểm trên màn hình, BIOS phải làm nhiều thủ tục nhưng nếu muốn vẽ toàn bộ 1 cửa sổ hình hay lưu trữ thì phải truy xuất trực tiếp RAM video.

- Với board đơn sắc MDA trong kiểu hiện văn bản số 7, 4 KB RAM được tổ chức như 1 dãy (array) gồm 2000 từ nhớ kề nhau (mỗi từ là mã thuộc tính: ký tự) tạo nên 25 dòng, 80 cột. RAM video bắt đẩu ở đọan B0000h, trong đó ký tự góc trên cùng bên trái là từ thứ nhất trong RAM video. Như vậy mỗi dòng có 160 byte (A0h). Địa chỉ của từ nhớ ứng với ký tự ở dòng i, cột j (i = 0-24, j = 0-79) được tính theo công thức sau:

  Address (i,j) = B0000h +A0h*i +02h*j.

- Với board EGA, ở kiểu hiện văn bản từ 0 đến 3 mã ký tự được lưu trữ trong lớp nhớ 0 cùng với thuộc tính trong lớp 1 của RAM video. Mạch logic chuyển địa chỉ trên board thực hiện sự kết hợp nhất định nào đó sao cho tổ chức và cấu trúc của RAM video cũng như cách tính địa chỉ vẫn tương đồng với cách của CPU. Trong chế độ đồ họa từ 13 đến 16, RAM video bắt đầu từ địa chỉ đoạn A000h. Các điểm ảnh được xếp kề cận nhau trong bộ nhớ và mỗi điểm ảnh đòi hỏi 4 bit, các bit này được phân ra ở 4 lớp nhớ. Như vậy địa chỉ của 1 trong 4 bit này trên 1 điểm ảnh không chỉ gồm đoạn video và offset mà còn thêm vào số lớp nhớ nữa.



Hình 5.16 - Các lớp nhỏ của RAM Video

Để hiện 1 điểm ảnh với 1 trong 16 màu, không phải chỉ tính địa chỉ bit mà còn phải thâm nhập 4 lớp nhớ. Muốn vậy, phải dùng thanh ghi mặt nạ bản đồ (map mask register). Thanh ghi này được định địa chỉ qua cổng chỉ số 3C4h với địa chỉ 02h và có thể được ghi qua cổng số liệu 3C5h. Cấu trúc của thanh ghi mặt nạ bản đố như sau:

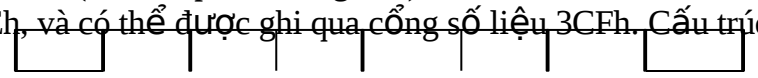
7	6	5	4	3	2	1	0
Res	Res	Res	Res	LY_3	LY_2	LY_1	LY_0

LY_3-LY_0: Thâm nhập ghi tới các lớp từ 0□3;
1 = cho phép; 0 = không cho phép
Res : Dự trữ

Ví dụ: Đặt bit 0 của byte ở địa chỉ A000:0000h cho lớ 0, 1, 3.

```
Mov AX, 0A000h      ; nạp đọan video vào AX
Mov ES, AX          ; truyền đọan video vào ES
Mov BX, 0000h       ; nạp offset 0000h vào BX
Out 3C4h, 02h       ; chỉ số 2  thanh ghi mặt nạ bản đồ
Out  3C5h, 0Bh      ; ghi 0000 1011b vào thanh ghi mặt nạ bản đồ
                          (cho phép lớp 0, 1, 3)
Mov 3C5h, 0Bh       ; đặt bit 0 trong các lớp 0, 1 và 3
```

Để lưu trữ nội dung màn hình cần phải đọc các giá trị bit của 4 lớp khi dùng thanh ghi chọn bản đồ đọc (read map select register). Nó được định địa chỉ với chỉ số 04h qua cổng chỉ số 3CEh, và có thể được ghi qua cổng số liệu 3CFh. Cấu trúc của thanh ghi này:



7	6	5	4	3	2	1	0
res	res	res	res	res	res	LY_1	LY_0

LY_1-LY_0: cho phép thâm nhập đọc với:
00 = lớp 0
01 = lớp 1
10 = lớp 2
11 = lớp 3
res : dự trữ

Ví dụ: đọc byte ở địa chỉ A000:0000h cho lớp 2:

```
Mov AX, A000h                Out 3CFh, 02h
Mov ES, AX
Mov BX, 0000h
Out 3Ceh, 04h
```

(cho phép lớp 2)

```
; nạp đọan video vào AX
; truyền đọan video vào ES
; nạp offset vào BX
; chỉ số 4  thanh ghi chọn bản đồ
đọc
; ghi 0000 0010b vào thanh ghi chọn
bản đồ đọc
```

```
Mov  AL, [ES:BX]      ; nạp byte trong lớp 2 vào AL.
```

Chú ý rằng 4 bit tại 4 lớp đại diện cho 1 điểm ảnh nên trong kiểu hiện 16 EGA có độ phân giải cao nhất mỗi dòng cần 80byte (640 điểm ảnh / 8 điểm ảnh trên 1 byte); mỗi trang màn hình gồm 32 KB. Địa chỉ byte của điểm ảnh ở dòng i, cột j trang k (i=0-349, j=0-639, k=0-1) là:

Address (i,j,k) = A0000h + 8000h*k + 50h*j + int (i/8).

Với board VGA, các chế độ hiện văn bản từ 0 đến 3 và 7 cũng như các chế độ đồ họa từ 4 đến 6 và 13 đến 16 của CGA. EGA và MDA đều chạy được trên nó.

Trong chế độ văn bản, mã ký tự được lưu trữ trong lớp nhớ 0 cùng với thuộc tính trong lớp 1 của RAM video VGA. Quá trình chuyển hóa địa chỉ cũng giống như EGA nhưng khác ở chổ nó vẫn đảm bảo chế độ văn bản 7 với độ phân giải 720x400, ma trận điểm ảnh 9x16. Trong chế độ đồ họa4 6 và 13 19 , mọi tổ chức, cấu trúc cũng như cách tính địa chỉ tương tự như CGA và EGA. VGA được tăng cường 3 kiểu hiện hình mới từ 17 đến 19.

Kiểu 17 tương thích với board đồ họa của máy PS/2 kiểu 30 là MCGA (multi colour graphics array). Các điểm ảnh chỉ gồm 1 bit (2 màu) được định vị chỉ trên lớp 0. Thí dụ, trong VGA kiểu 17 với 80 byte trên 1 dòng (640 điểm ảnh / 8 điểm ảnh trên 1 byte). Mỗi trang màn hình gồm 40 KB. Địa chỉ của byte ở dòng i, cột j (i= 0-479), j=0-639) như sau:

Address (i,j) = A0000h+50h*j+int (i/8)

Kiểu 18, 4 bit của điểm ảnh được phân trong 4 lớp nhớ như ở EGA. Trong kiểu VGA phân giải cao với 16 màu khác nhau, 80 byte trên 1 dòng (640 điểm ảnh / 8 điểm ảnh trên 1 byte), mỗi trang màn hình gồm 40 KB (A0000h byte); địa chỉ của mỗi byte ở dòng i, cột j (i=0-479; j = 0-639) bằng:

Address (i,j) = A0000h + 50h*j + int (i/8).

Kiểu 19 với 256 màu cho 1 điểm ảnh thì RAM video lại được tổ chức rất đơn giản như 1 dãy tuyến tính, trong đó 1 byte tương ứng với 1 điểm ảnh. Giá trị của byte phân định màu của điểm ảnh. Kiểu này đòi hỏi 320 byte (140h) trên 1 dòng (320 điểm ảnh / 1 điểm ảnh trên 1 byte). Một trang màn hình gồm 64 KB (10000h) nhưng chỉ có 64000 byte được sử dụng. Địa chỉ của điểm ảnh trong dòng i, cột j (i = 0-199, j=0-319) là:

Address (i,j) = A0000h + 140h*j + i

4.2.4. Bus cục bộ và chip xử lý đồ họa

Để tăng tốc độ hiện đồ họa có 2 giải pháp:

- Dùng bus cục bộ 32 bit để tránh hiện tượng nghẽn cổ chai (bottleneck) do bus ISA chỉ có 16 bit và tốc độ hạn chế (8.33Mhz); điều này cho phép 1 lượng thông tin nhiều hơn được trao đổi giữa CPU và board mạch trong 1 đơn vị thời gian.

- Dùng chip xử lý đồ họa với BIOS riêng trên board mạch điều khiển monitor. Chip này sẽ làm hầu hết các công việc trừ một ít lệnh và thông số mô tả nội dung phần màn hình cần hiện là được cấp từ CPU. Thí dụ cần vẽ 1 hình chữ nhật với màu nào đó, board chỉ cần vài thông số ban đầu từ CPU như tọa độ của 2 góc và giá tri màu là đủ. Cách giải quyết như vậy rất có lợi khi PC chạy trong chế độ đa nhiệm.