CHƯƠNG I

GIỚI THIỆU CHUNG VỀ MÁY TÍNH ĐIỆN TỬ

1.1.Sự ra đời và phát triển của máy tính

Trước công nguyên đã xuất hiện các công cụ xử lý số để tính toán các vấn đề số học.

Giữa thế kỷ 17 nhà bác học Pháp B.Pascal (1823-1662) đã có nhiều đóng góp trong cơ giới hóa tính toán số học. Ông đã làm ra một máy tính mới với nguyên lý mới "bánh xe răng cưa". Các bánh xe của Pascal có 10 vị trí (từ 0-9) và xếp đặt kế tiếp nhau. Các máy này giúp cho việc tính tiền được nhanh chóng.

Tiếp theo đó nhà bác học Đức Leibniz (1646-1716) đã chế ra máy tính cơ học có thể nhân và khai căn bậc 2.

Thế kỷ 19: Nhà bác học Anh C.Babbage (1791-1871) đã nghĩ đến tự động hóa các máy tính cơ học, tự động thực hiện liên tiếp các phép tính. Máy của Babbage cần dùng băng đục lỗ để xác định phép tính thực hiện. Kiểu máy tính này được gọi là máy tính chương trình ngoài (ngược với máy tính hiện nay là máy tính chương trình trong) thực hiện luôn một loạt phép tính cố định trong một chương trình. Máy tính này đã có đủ CPU, bộ nhớ và thiết bị vào/ra.

Trong thế chiến lần thứ 2, nhiều hãng và trương đại học ở Mỹ đã xây dựng các máy tính bằng rơle dựa trên nguyên lý Babbage.

John Mauchly và học trò J. Presper Eckert ở trường Đại học Pennsylvania theo yêu cầu thiết kế máy tính để tính đường đạn. Mauchly gặp Atanasoff-1941 và sử dụng nguyên lý máy ABC (Atanasoff-Berry Computer, máy tính đưa ra 1930) để phát triển và đưa ra máy tính ENIAC (Electronic Numerical Integrator and Calculator, năm 1943 - 1946). Máy tính ENIAC được xây dựng từ các đèn điện tử. Máy nặng 30 tấn, trải rộng trên diện tích 170 m^2, công suất tiêu thụ 200 kW, thực hiện được 5000 phép tính/giây. ENIAC được coi là máy tính điện tử đầu tiên.

Nhà toán học Von Neumann là người tư vấn trong chế tạo ENIAC đã nghiên cứu máy tính ENIAC và đưa ra quan niệm mới.

1) Chương trình được ghi trước vào bộ nhớ: Máy tính có bộ nhớ để lưu trữ một chương trình trước khi thực hiện và ghi kết quả trung gian. Chương trình được thực hiện theo trình tự.
2) Ngắt rẽ nhánh: Máy tự động rẽ nhánh nhờ các quyết định logic

Phần lớn các máy tính ngày nay đều làm việc trên nguyên lý Von Neumann.

Từ khi ra đời đến nay, máy tính đã trải qua 5 thế hệ. Các thế hệ của máy tính điện tử bao gồm:

1) Thế hệ 1 (1951-1958): Đèn điện tử chân không
2) Thế hệ 2 (1959-1964): Bán dẫn, Assembller, Cobol (59)
3) Thế hệ 3 (1965-1970): Vi điện tử cỡ nhỏ và vừa (SSI, MSI)
4) Thế hệ 4 (1971 đến nay): Vi điện tử cỡ lớn và siêu lớn (LSI, VLSI, MSI, GSI)
5) Thế hệ 5 (1980-1990): Dự án xây dựng máy tính thế hệ 5 với tính năng xử lý song song trí tuệ nhân tạo và ngôn ngữ tự nhiên. Tuy nhiên dự án này không thành công như mục tiêu đã đề ra.

1.2.Phân loại các hệ thống máy tính

1.2.1. Phân loại các hệ thống máy tính theo hiệu năng (khả năng tính toán)

Dựa trên khả năng tính toán của MTĐT, người ta chia làm 4 loại: Máy tính cá nhân (Personal Computer), Máy tính Mini (Minicomputer), Máy tính lớn (Mainframe) và Máy tính siêu lớn (Supercomputer). Các thông số đặc trưng để đánh giá hiệu năng của máy tính là:

- Tốc độ (khả năng tính toán)
- Độ dài của từ xử lý
- Dung lượng bộ nhớ trong
- Dung lượng bộ nhớ ngoài

a. Máy tính cá nhân

Máy tính cá nhân thường là máy tính một người sử dụng.

Ví dụ:

Máy vi tính Pentium IV với

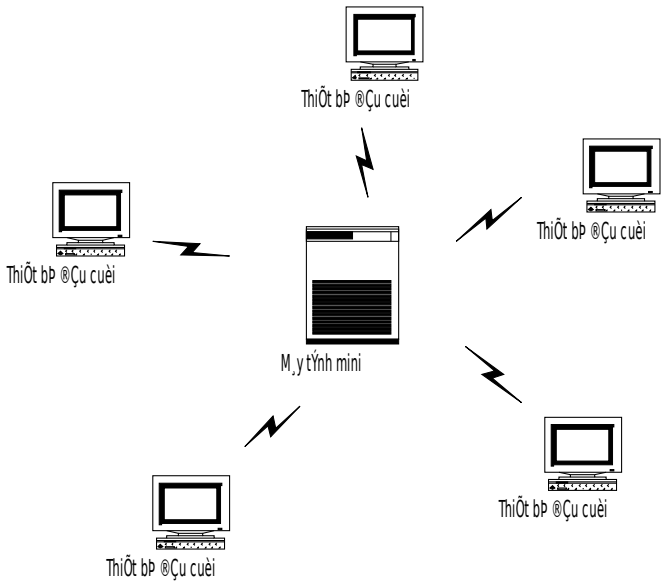
- Tốc độ được đặc trưng bằng tần số của đồng hồ 1,4GHz; 1,5GHz;...2,8GHz; 3,1GHz.
- Độ dài từ xử lý 32 bit.
- Dung lượng bộ nhớ trong: 64MB, 128MB, 256MB,...
- Dung lượng đĩa cứng: 20 GB, 40 GB, 80GB...

b. Máy tính cỡ vừa (Mini)

Một máy tính Mini thường có khả năng cho 20 – 100 người đồng thời sử dụng.

Máy tính Mini có hiệu năng sử dụng thường gấp 10 lần máy tính cá nhân.

Cấu hình của một máy tính mini có thể được thể hiện như trên *hình 1.1*.



H×nh 1.1: CÊu h×nh tiªu biÓu cña m¸y

c. Máy tính lớn

Máy tính lớn là máy tính đa người sử dụng. Thường một máy tính lớn có thể nối với hơn 100 thiết bị đầu cuối.

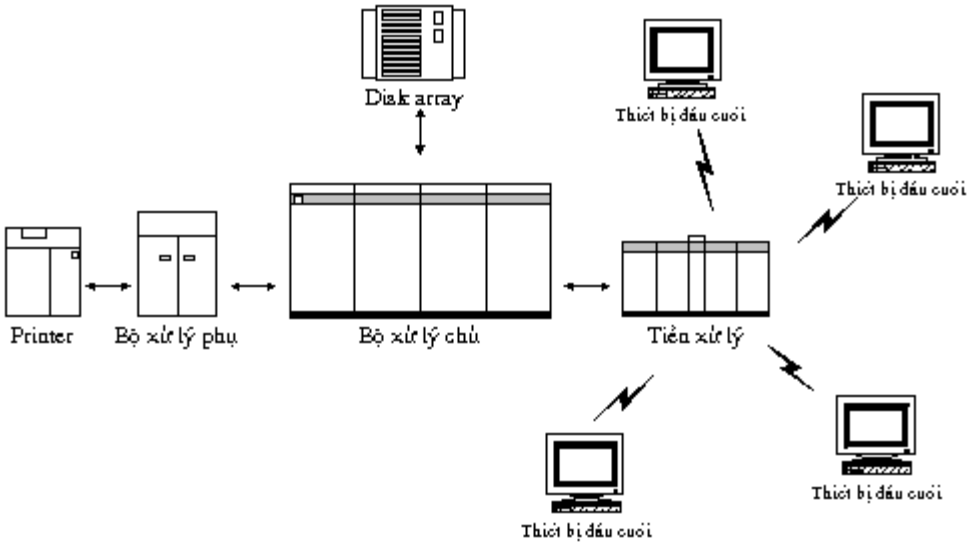
Thông thường máy tính lớn có cụm xử lý:

- *Bộ tiền xử lý (The-Front end-Processor)* làm nhiệm vụ thu thập và xử lý số bộ dữ liệu tới từ các thiết bị đầu cuối trước khi đưa vào cho bộ xử lý chủ.
- *Bộ xử lý chủ (Host processor)* làm nhiệm vụ xử lý chính và điều khiển toàn bộ hệ thống.
- *Bộ xử lý phụ trợ (The-back end-Processor)* làm nhiệm vụ quản lý tệp, cơ sở dữ liệu và máy in.

Máy tính lớn có khả năng tính toán gấp 10 lần máy Mini.

Ví dụ: Máy tính IBM 4381, IBM RS/400.

Cấu hình của một máy tính lớn được thể hiện như trên *hình 1.2*.



Hình 1.2: Cấu hình tiêu biểu của máy tính lớn

d. Máy tính siêu lớn

Máy tính siêu lớn các cấu trúc hướng vào tăng tốc độ xử lý bằng cách:

- Dùng linh kiện nhanh hơn.
- Rút ngắn khoảng cách giữa các linh kiện.
- Kiến trúc máy này khác với các loại máy mini.
- Máy tính siêu lớn phải làm việc trong phòng lạnh.

Ví dụ: Máy tính Cray 5,5 triệu USD; Compag 200 triệu USD – hàng ngàn bộ xử lý.

Máy tính siêu lớn có khả năng tính toán lớn hơn 10 lần máy tính lớn.

Ứng dụng:

- Phân tích số liệu động đất
- Mô phỏng dòng khí của máy bay.
- Nghiên cứu sức nổ của phản ứng hạt nhân.
- Nghiên cứu mô phỏng, tạo dạng cơn bão trong dự báo thời tiết.
- Kiến tạo những mô hình chuyển động trong vũ trụ.

1.2.2. Phân loại các hệ thống máy tính theo kiến trúc

Ngoài nguyên lý máy trình tự (theo Von Neumann), trong giai đoạn gần đây còn phát triển các máy tính theo nguyên lý song song. Theo Michael Flymn có thể phân loại máy tính như sau:

a) Các kiểu nguyên lí xử lí

Cơ sơ để phân loại dựa trên tính đơn xâu hay đa xâu của lệnh và dữ liệu do Michael Flynn đề xuất. Trong đó xâu lệnh (instruction stream) là một tập tuần tự các lệnh (instruction) được thực hiện bởi một CPU và xâu dữ liệu (data stream) là chuỗi các dữ liệu mà xâu lệnh cần có. Có bốn loại kiến trúc như sau:

SISD (**single instrucction stream, single data stream**).

Theo nguyên lý Von Neumann – máy tính tuần tự.

Một thời điểm chỉ có một lệnh được thực hiện, tương tự như khi sản xuất ô tô, một người làm tất cả các công việc.

Đây là mô hình máy tính von Neumann, còn gọi là thiết kế kiểu máy tính nối tiếp, trong đó chỉ một lệnh thực hiện ở một thời điểm. SISD qui chiếu loại máy serial scalar computer. Tất cả các máy SISD sử dụng một bộ đếm chương trình, PC (Program Counter), để tạo ra quá trình thực hiện liên tiấp các lệnh. Sau mỗi lần lệnh lấy được từ bộ nhớ, PC tự động cập nhật giá trị mới để có địa chỉ lệnh tiếp theo. (Tức thực hiện theo thứ tự liên tiếp (serial order)).

MISD (**multiple instruction stream, single data stream**).

Nhiều lệnh cùng thao tác trên một mảng dữ liệu.

Trong thực tế là kiến trúc mà số liệu được đưa qua một chuỗi các đơn vị xử lý.

- ✓ Ví dụ tương tự dây chuyền sản xuất mỗi người công nhân thực hiện một nhiệm vụ hoặc một tập hợp nhiệm vụ dựa trên kết quả của người trước đó.

Các kiến trúc đường ống như systolic array và vector processor thuộc loại MISD

Là loại kiến trúc với khả năng thực hiện vài lệnh thao tác trên một xâu dữ liệu. Cách tổ chức kiểu máy này như thế nào: có hai cách diễn đạt.

- Xem xét một lớp các máy tính, trong đó các đơn vị xử lí riêng biệt nhận các lệnh riêng biệt thao tác trên cùng một dữ liệu. Đây là loại máy thách thức các nhà kiến trúc máy tính, và chưa có máy nào loại này trong thực tế.

- Có thể xem xét khác như sau: đó là lớp các máy tính, trong đó dữ liệu đi qua liên tiếp các đơn vị xử lí. Các máy tính kiến trúc kiểu đường ống (pipelined), hay các bộ xử lí vector, thuộc lớp máy này: Xử lí vector là thực hiện qua các tầng, mỗi tầng thực hiện một chức năng xác định và tạo ra một kết quả trung gian. Lí do để nói kiểu kiến trúc này là MISD vì rằng các thành phần của một vector được nhìn nhận, thuộc vào cùng một dữ liệu, và tất cả các tầng biểu diễn đa lệnh áp dụng cho vector đó. (Ví dụ: dây chuyền lắp ráp…)

SIMD (**single instuction stream, multiple data stream**).

Chỉ một biểu lệnh giống nhau xử lý nhiều số liệu đồng thời.

Một đơn vị điều khiển khởi động nhiều đơn vị xử lý, tương tự như MISD hỗ trợ xử lý vector. Mỗi một phần tử của vector được đưa vào đơn vị xử lý riêng để thực hiện đồng thời.

Tương tự như trong dây chuyền sản xuất ô tô, nhiều công nhân cùng làm một số công việc. Mỗi công nhân tự xây dựng ô tô bằng cách cùng thực hiện một nhiệm vụ giống nhau tại một thời điểm.

Ví dụ tính lương cho 1.000 người thì máy SISD phải thực hiện 1.000 vòng lặp, còn trên máy SIMD có thể thực hiện song song 1.000 dòng số liệu .

Môt lệnh đơn được sử dụng để xử lí nhiều dữ liệu đồng thời. Trong loại máy này, một đơn vị điều khiển kích hoạt nhiều đợn vụ xử lí riêng biệt. SIMD hổ trợ xử lí vector như nói trên: gán các thành phần của vector cho từng đơn vị xử lí riêng biệt để các đơn vị này tính toán đồng thời. Ví dụ tính lương theo giờ cho 1000 người làm:

- trên máy SISD: cầm làm 1000 lần tuần tự ;
- trên máy SIMD: thực hiện song song đồng thời trên 1000 xâu xử liệu khác nhau (mỗi tính toán cho một lao động).

(ví dụ: Nhiều lao động làm cùng một việc như nhau đồng thời, sau khi xong việc, họ nhận công việc mới)

MIMD (**multiple instruction stream, multiple data stream**).

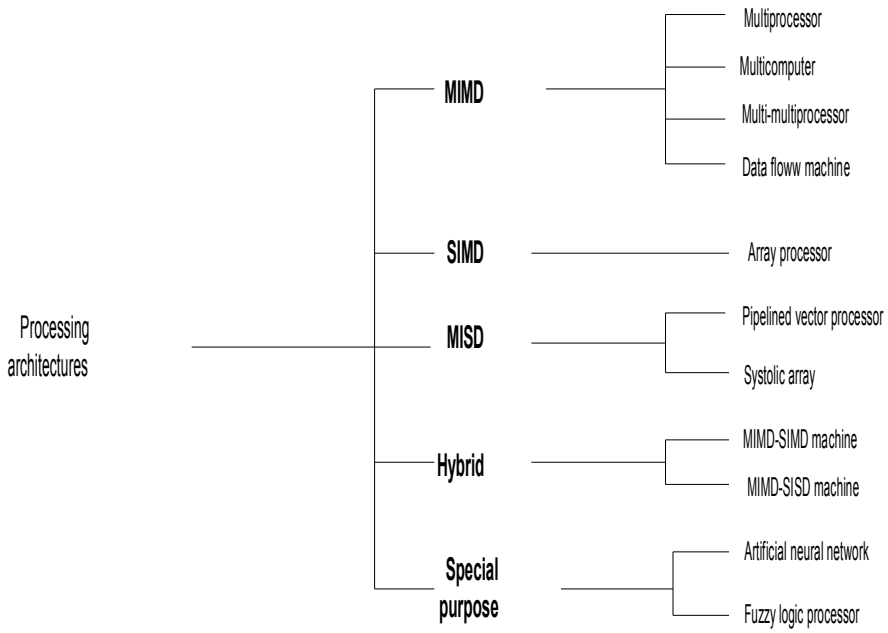
Máy có nhiều đơn vị xử lý – nhiều lệnh khác nhau được dùng để xử lý nhiều số liệu đồng thời. Đây là máy phức tạp nhất nhưng cũng hứa hẹn nhất để đạt được kết quả thông qua xử lý tương tranh.

Tương tự như trong dây chuyền sản xuất ô tô, mỗi người công nhân tự xây dựng ô tô một cách độc lập theo tập hợp các lệnh riêng của mình.

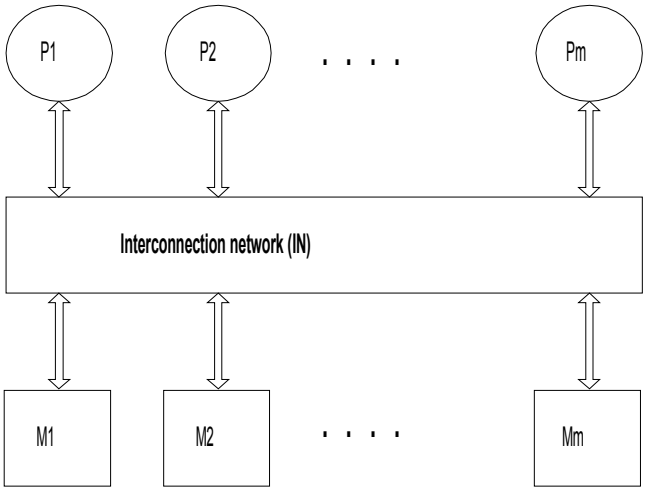
Là loại máy có nhiều đơn vị xử lí, nhiều lệnh dùng để xử lí nhiều dữ liệu đồng thời. Loại máy này phức tạp nhất, tính đồng thời ở đây là vì không chỉ các bộ xử lí hoạt động đồng thời, mà nhiều chương trình (tiến trình) được thực hiện trong cùng một khung thời gian. (Ví dụ: Mỗi người không làm cùng một việc đồng thời, mà thực hiện độc lập tập các chỉ thị cho riêng họ).

b. Các kiểu máy tính

Tuy nhiên cho tới ngày nay cách phân loại trên chỉ ở mức độ tương đối. Thực tế kiến trúc máy mang mầu sắc pha trộn. Hình sau cho thấy cách phân loại các kiến trúc xử lí thường thấy:

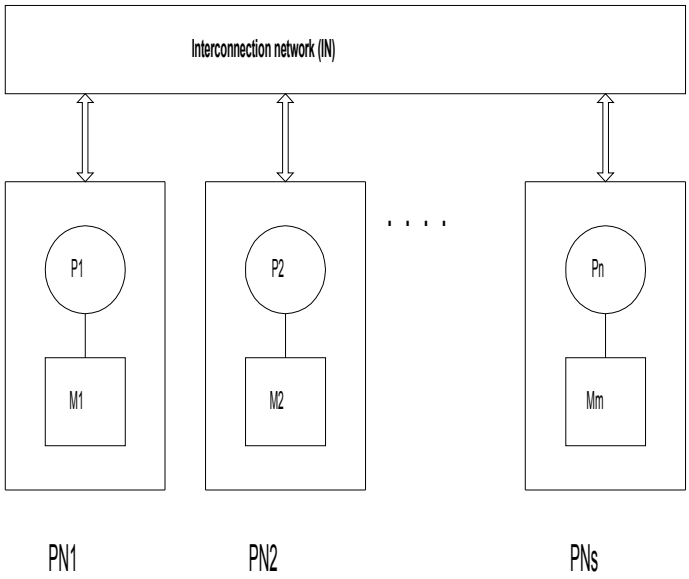


1. multiprocessor: Là một máy song song với nhiều CPU , chia sẻ bộ nhớ hệ thống. Mỗi CPU có thể lập để chạy một phần của một chương trình, hay chạy một chương trình khác với chương trình chạy trên CPU khác. Mô hình sau:



Mm modul bộ nhớ, P1,…Pn CPU. mạnh kết nối bên trong (IN) kết nối mỗi CPU với một tập các modul bộ nhớ. Để chuyển data giữa hai CPU, một trình tự chuyển data phải được lập trình để thực hiện.

2.mulicomputer: Là một máy song song, trong đó mỗi CPU có một bộ nhớ riêng biệt, trong khi đó bộ nhớ chính phân phối riêng phần cho từng CPU. Đặc điểm là các CPU không thể truy nhập vào bộ nhớ của nhau. Đây là sự phân biệt cơ bản với máy đa xử lí nói trước đó.

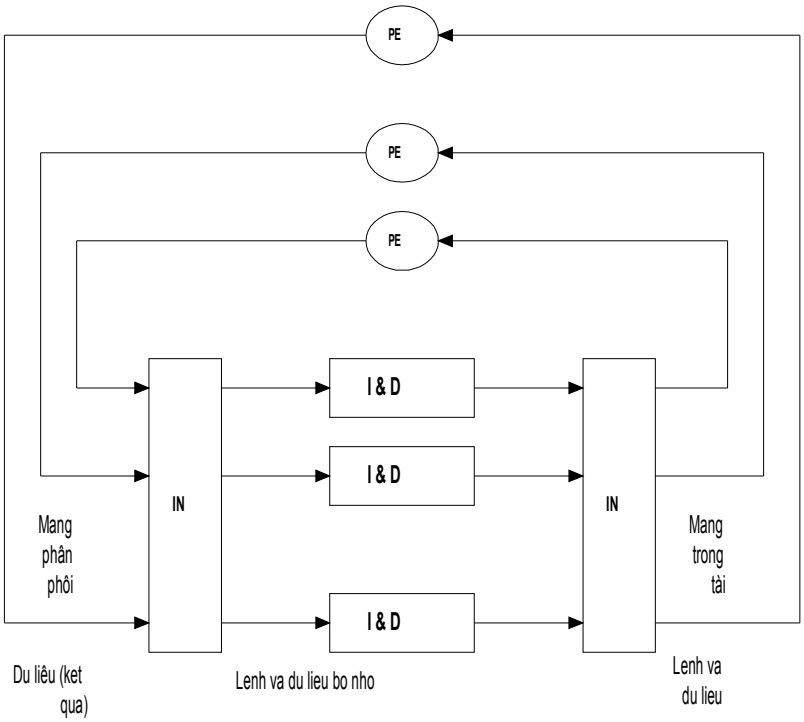


Trong hình, mỗi nút (PNs) có CPU Pn và Mn riêng biệt. NI là mạng kết nối. Data sẽ được các lệnh chuyển chuyển từ PN này đến PN kia có kết nối với nó. Nếu chuyển data tới một PN không kết nối, thì data phải đi qua một PN trung gian.

3. Multi-multiprocessor: Là phối hợp của hai kiểu cấu trúc trên (multiprocessor, và mulicompute), trong đó mỗi một nút là một đa xử lí (multiprocessor).

4.Data flow machine: Là loại kiến trúc, trong đó một lệnh sẵn sàng thực hiện khi data cho các toán hạng (operand) của lệnh đã sẵn sàng. Data gọi là sẵn có là do kết quả kết quả thực hiện lệnh trước đó và chuyển data đó cho các toán hạng của lệnh đang đợi. Với cách thức như vậy, kiến trúc tạo ra luồng dữ liệu. Ở đây ta thấy không cần có bộ đếm chương trình như trong mô hình von Neumann.

Các lệnh luồng dữ liệu là tự chứa đựng trong bản thân lệnh, có nghĩa là lệnh không qui chiếu vào bộ nhớ chính, mà mang theo các giá trị của biến trong bản thân lệnh. Trong kiến trúc này việc thực hiện một lệnh không tác động đến các lệnh khác đang sẵn sàng thực hiện. Như vậy sẽ có nhiều lệnh đang sẵn sàng có thể được thực hiện đồng thời và kết quả là có tiềm năng tính toán đồng thời rất cao.

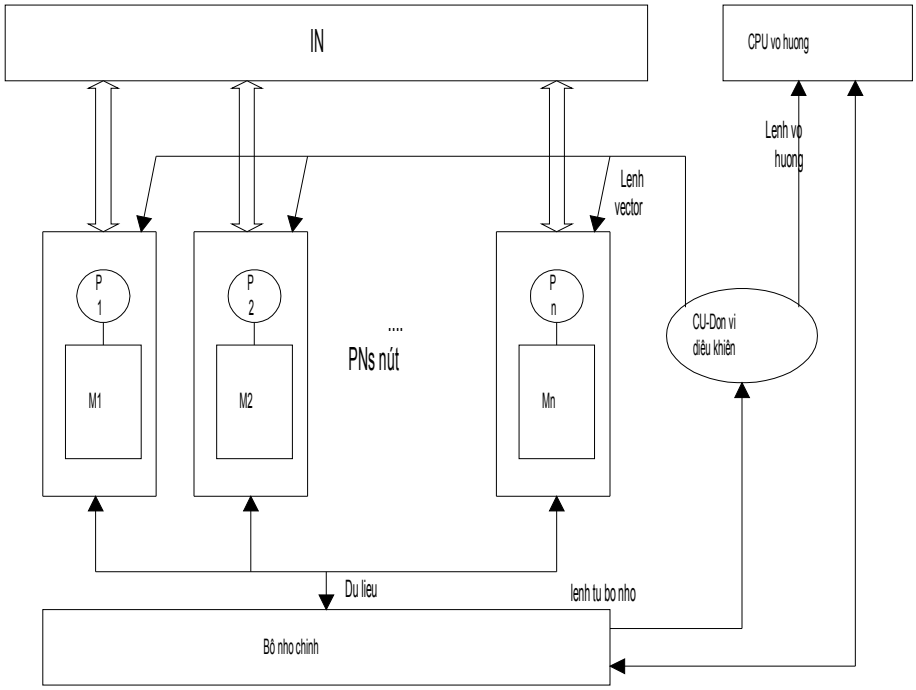


Lệnh và các toán hạng được giữ trong bộ nhớ lệnh và data (I&D). Một khi lệnh sẵn sàng thực hiện, lệnh được gởi đến một trong các thành phần xử lí (PE) qua mạng phân phôi IN. Mỗi PE là một CPU với bộ nhớ riêng hạn chế. Khi thực hiện lệnh , PE tính toán và gởi kết quả tới mạng đích (IN) đến đích nhận data..

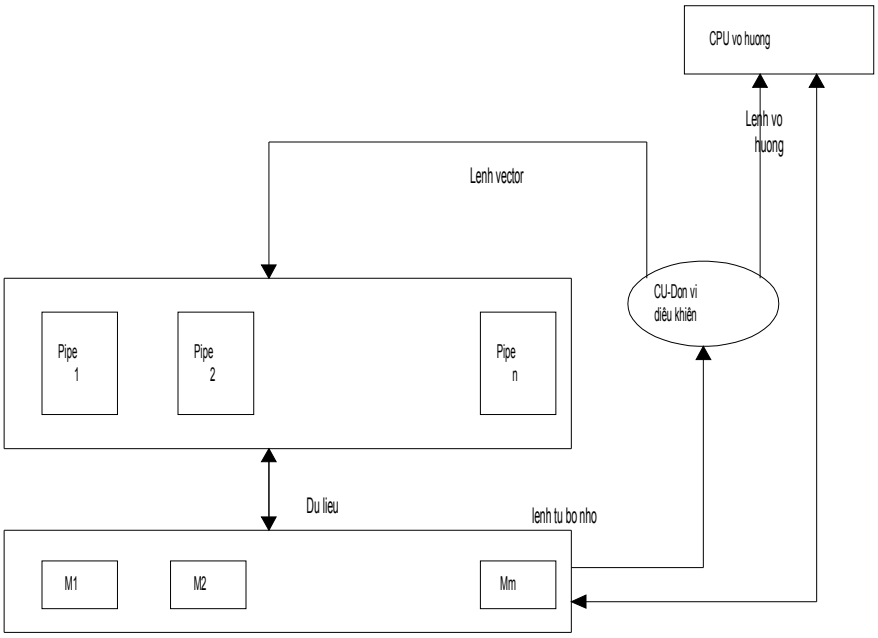
5. Array processor. (Xử lí đa chiều):

Bao gồm một tập các nút xử lí (PNs) và một bộ xử lí vô hướng (scalar CPU) làm việc dưới sự kiểm soát của đơn vị xử lí trung tâm (Control Unit CU). CU tìm các lệnh trong bộ nhớ, giải mã lệnh và gởi các lệnh đó cho CPU vô hướng hay cho các nút phụ thuộc vào kiểu lệnh. Nếu là lệnh vô hướng, thi chuyển cho CPU vô hướng, còn thì chuyển cho tất cả các nút PNs. Các PN thực hiện cùng một lệnh đồng thời trên các dữ liệu khác nhau đã có trong bộ nhớ của nó. Như vậy xử lí đa chiều chỉ cần một chương trình (một code) cho tất cả các nút , không cần nhân bản chương trình cho mỗi nút.

Ý tương đằng sau xử lí đa chiều là để khai thác tính song song trong tập dữ liệu đã cho, chứ không phải để thực hiện song song trình tự của thực hiện lệnh. Tính toán song song thực hiện bằng cách gán cho mỗi CPU tới một phần dữ liệu. Nếu data là vector, thì data đó đơn giản là một thành phần vector. Xử lí đa chiều tăng cường hiệu năng bằng cách thao tác tất cả các phần dữ liệu (đã gán cho mỗi CPU) đồng thời. Các phép toán số học, logic đều có thể thực hiện trên vector, nên xử lí đa chiều còn gọi là xử lí vector.

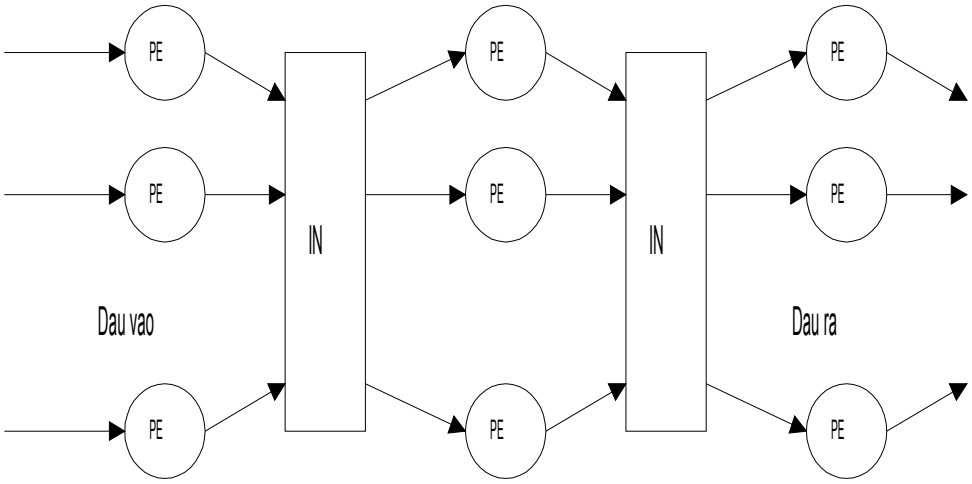


6. Pipelined vector procesor (Xử lí vector kiểu đường ống) Là khả năng xử lí các toán hạng vector (chuỗi các data liên tục) có hiệu quả cao. Đó là điểm khác biệt với xử lí đa chiều: đa chiều được điều khiển bằng lệnh (lệnh kiểu vector), còn xử lí vector kiểu đường ống lại được điều khiển bởi chuỗi dữ liệu liên tục.



Trong kiến trúc này có hai CPU chính: một CPU vô hướng, và một CPU vector. Cả hai nhận lệnh từ CU. CPU vector kiểm soát thực hiện các lệnh vector bằng các ống dẫn, còn CPU vô hướng thực hiện lệnh như các CPU thông thường. CU lấy lệnh từ bộ nhớ, giải mã lệnh và tuỳ loại lệnh sẽ chuyển cho các CPU.

7. Systolic array: Bao gồm số lượng lớn các thành phần xử lí giống như nhau (processing elemtne PE). Mỗi PE có bộ nhớ gioéi hạn, và để không giới hạn số PE, mỗi PE chie được nối đến các láng giềng của nó bởi mạng IN. Như vậy ta nhận thấy cách nối giống kiến trúc đường ống, ví dụ như ở trường tuyến tính hay trường hai chiều. Trong hệ thống này, các dữ liệu hay các kết quả từng phần đi qua các PE trong thời gian thực hiện của một vài chu kì xử lí. Ở mỗi chu kì, một số PE thực hiện một số các thao tác như sanhu (ví dụ như nhân hay chia) trên các dữ liệu của các PE đó, và gởi các kết quả từng phần tới các PE láng giềng.



8. Hybrid architecture: Là kết hợp các đặc thù của các kiến trúc khác nhau để tạo ra hiệu năng tốt nhất cho tính toán song song. Có hai kiểu tính toán song song:

- kiểm soát song song: hai hay nhiều tính toán (operatons) thực hiện đồng thời trên các CPU khác nhau;

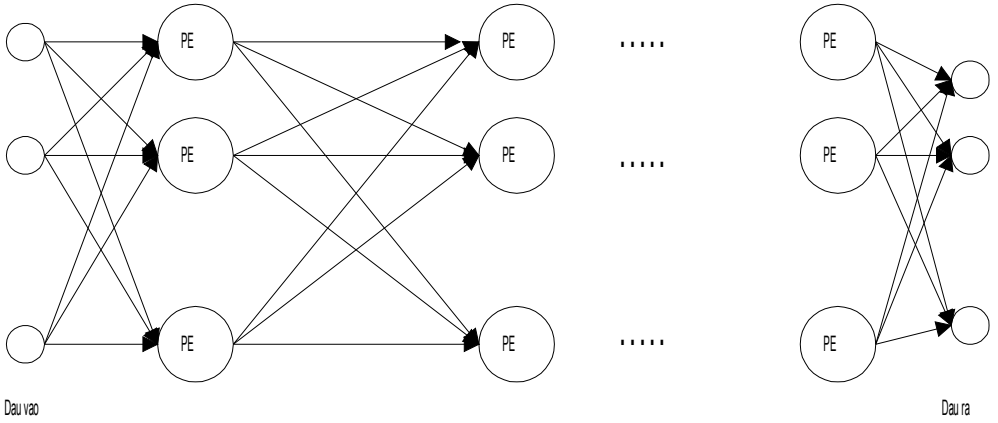
- dữ liệu song song: cùng một tính toán thực hiện trên các phần của dữ liệu bởi nhiều CPU đồng thời.

MIMD là mô hình lí tưởng của kiểm soát song song,nó thích hợp cho bài toán cần nhiều tính toán khác nhau thực hiện đồng thời trên một dữ liệu tách biệt. Máy SIMD thích hợp cho xuể lí xử lí số liệu song song, thích hợp cho bài toán kiểu cùng một thao tác xử lí đồng thời trên các phần khác nhau của một dữ liệu. SIMD hổ trợ xử lí vector qua thiết kế đường ống.

Trong thực tế xử lí dữ liệu song song là rất lớn, vì quá trình đó tỉ lệ với lượng dữ liệu đưa vào tính toán. Tuy nhiên không phải lúc nào giải pháp này cũng thành công, do đó cần sử dụng phối hợp cả hai cách nói trên. Ví dụ, một số ứng dụng chạy tốt khi chia chia nhỏ để mỗi phần của ứng dụng dùng xử lí dữ liệu song song, trong khi tất cả các phần lại chạy theo kiểu kiểm si\oát song song theo kiến trúc ống. Một nhóm các CPU lấy dữ liệu, thực hiện các tính toán ban đầu, sau đó chuyển kết quả cho nhosm thứ hai, và nhóm hai tính toán, chuyển tiếp … cho tới khi có kết quả cuối. Các máy kết hợp cả hai MIMD và SIMD, cho hiệu quả đáng ghi nhận.

9. Các thiết bị đặc biệt

9.1 **Mạng neuron nhân tạo (Artificial neural network (ANN))**: Được xây dựng từ vô số các thành phần tính toán hoạt động song song, có khả năng học và tự thích nghi với sự thay đổi của môi trường tính toán và đương đầu với hỗn loạn. Cấu trúc mạng neuron nhân tạo hứa hẹn giải quyết được các vấn đề mà máy von Neumann khó có thể thực hiện được (ví dụ như mô phỏng thông tin tự nhiên, nhận dạng mẫu gene, … những vấn đề cần có năng lực tính toán kiểu con người mới thực hiện được).



Mỗi một PE bắt chước một vài đặc tính của neuron sinh học, chúng có một tập các đầu vào, và một hay vài đầu ra. Mỗi đầu vào được gán một trọng lượng số. Trọng lượng này tương tự như nồng độ tiếp hợp (synaptic strength) của neuron sinh học. Tất cả các đầu vào của mỗi PE được bội lên bằng trọng lượng và sau đó cộng lại để xác định mức hoạt động của neuron. Một mức hoạt động là một chức năng, gọi là chức năng hoạt động, được dùng để tạo một đầu ra (tín hiệu ra). Đầu ra của một lớp là đầu vào của lớp tiếp theo, và tại đó chúng được cộng lại, đánh giá, tạo đầu ra. Quá trình này đi qua toàn mạng neuron để tìm được một quyết định cuối cùng nào đó.

Không giống như von Neumann, trong đó thành phần cơ bản là CPU, ANN là kiến trúc kết nối (mạng neuron) giữa các PE. Với một bài toán cho trước, ta cần xác định giá trị chính xác cho các trọng lượng để mạng có thể thực hiện các tính toán cần thiết. Thông thường việc xác định giá trị được tiến hành bằng phương pháp điều chỉnh tương tác của trọng lượng theo hướng cải thiện hiệu năng của mạng neuron. Luật điều chỉnh trọng lượng gọi là luật học (learning rule) và toàn bộ quá trình để có được giá trị chính xác của trọng lượng gọi là quá trình học (learning).

9.2 Logic mờ (Fuzzy logic processor): là các nguyên lí hình thức của lập luận gần đúng. Trong khi trước đây ta có lập luận hai giá trị (true và false). Logic mờ nổ lực giải quyết hiệu quả với tính phức tạp của quá trình nhận thức của con người, và nó vượt qua một số các phiền phức phối hợp với logic nhị nguyên cổ điển không thể phản ánh được quá trình nhận thức thực của con người. Tuy các phần mềm phát triển trên logic mờ mang lại một số kết quả tố cho vài ứng dụng, thì các ứng dụng hiệu năng cao đang cần các bộ xử lí logic mờ chuyên dụng.

1.2.2. Processor performance

The performance of the processor, which can be considered as the central nervous system of the units that compose the computer system, is measured using the number of instructions that can be executedd in a unit of time as an index. These indexes are indicated below.

(1) MIPS

MIPS is an acronym of Million Instructions Per Second, and indicates in million units the number of instructions that can be executed in one second. In other words, a 1 MIPS processor is a processor that can execute one million instructions per second. Basically, the larger the number of instructions that can be executed, the higher the value. The term MIPS is mainly used to indicate the performance of processors of high end mainframe computers. However, it is meaningless to use this index to compare processors of
different types of machines that execute different instruction contents.

(3) FLOP

Floating operation

(3) Clock

In order to set the pace in which the micro-instructions, which are basic operations, are executed, the processor has a click inside. A quartz crystal oscillator that pulses in regular intervals when electrical current passes through is sued in this clock. The time taken for this oscillator to pulse once (one cycle) is called click, The basic operations of the processor are performed according to this clock. The number of clocks varies according to the instruction.

The clock reciprocal number is called clock frequency. Clock frequency is used as an index to measure the performance of a personal computer.

Example Performance of a processor with a clock frequency of 500 MHz

500 MHz = 500×10^6 Hz = 500,000,000 Hz (times/second); 500 hundred million pulses per second

$\frac{1}{0.5 \times 10^9} = 2 \times 10^{-9} = 2$ nano (seconds/times); 1 pulse for every 2 nanoseconds

(4) CPI (Cycles Per Instruction)

A CPI is the number of clocks required to execute instruction This index indirectly indicates the execution time of one instruction

Literature Architecture:

John L. Hennessy and David A. Patterson, Computer Architecture: A Quantitative Approach, third edition, Morgan Kaufmann, New York, 2003. See www.mkp.com/CA3.

David A. Patterson and John L. Hennessy, Computer Organization and Design: The Hardware Interface. *Text for COEN 171.*

Gerrit A. Blaauw and Frederick P. Brooks, Jr., Computer Archtecture: Concepts and Evolution, *Addison Wesley, 1997.*

William Stallings, Computer Organization and Architecture, *Prentice Hall, 2000.*

Miles J. Murdocca and Vincent P. Heuring, Principles of Computer Architecture, *Prentice Hall, 2000.*

John D. Carpinelli, Computer Systems Organization and Architecture, *Addison Wesley, 2001.*

Davis A. Palterson & John L. Hannesy, Computer Organization and design: The hardware / Software interface, 1998.

Andrew S Tanenbaun, Structred Computer Organization, 4th,1998.James M. feldman, Charles T. Retter, Computer Architecture Adesignned text based on generic RISC. 1994.

1.3. Máy tính mẫu

Máy tính điện tử đơn giản được mô tả dưới các mức độ khác nhau:

1 – Sơ đồ khối

2 – Sơ đồ kiến trúc

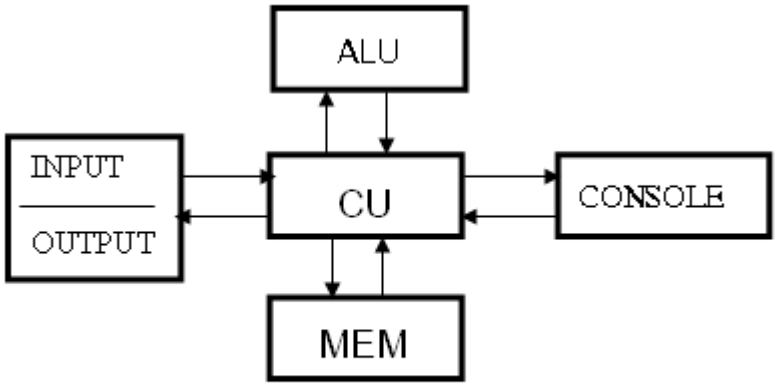
3 – Sơ đồ vận chuyển giữa các thanh ghi

4 – Sơ đồ mạch logic bậc thấp

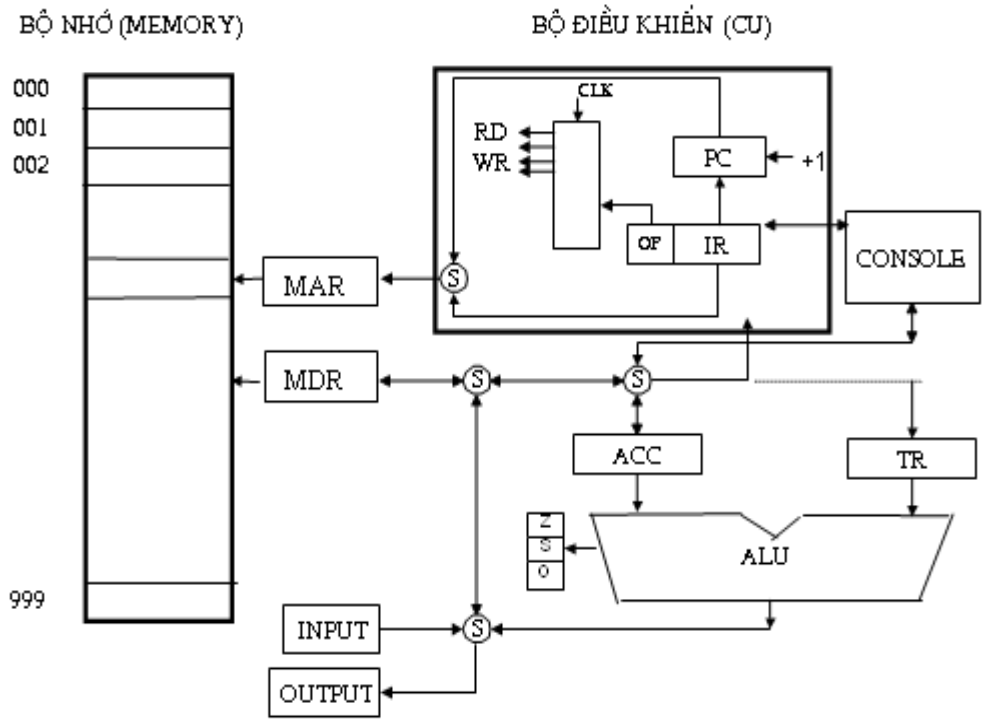
5 – Sơ đồ các mạch điện tử

1.3.1. Sơ đồ khối của máy tính mẫu

Máy tính mẫu *(hình 1.3 và hình 1.4)* này được mô phỏng theo một máy tính của trường đại học Virginia Technology nhằm mục đích trang bị những khái niệm cơ bản ban đầu để nhập môn vào cấu trúc máy tính.



Hình 1.3: Sơ đồ khối của máy tính mẫu



Hình 1.4: Sơ đồ khối của máy tính mẫu

Máy tính mẫu có các đặc điểm sau:

- Đơn giản
- Máy làm việc với hệ 10
- Dung lượng bộ nhớ nhỏ
- Tập lệnh hạn chế

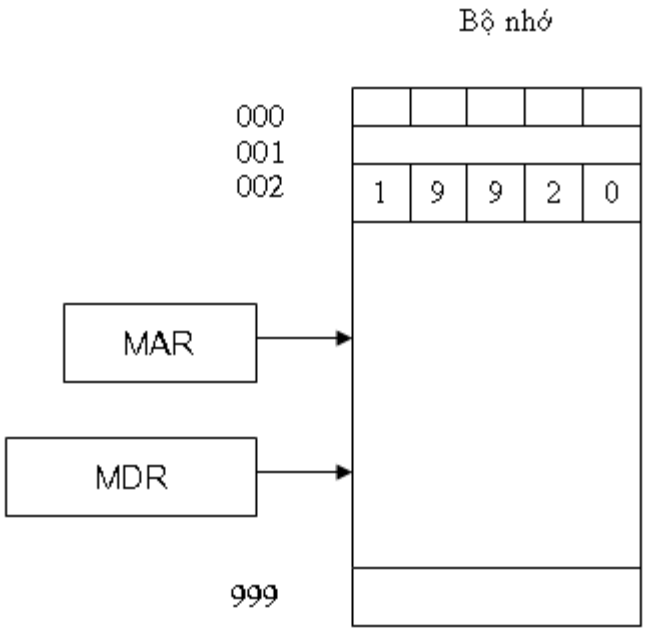
a. Bộ nhớ

Bộ nhớ của máy tính mẫu được thể hiện trên *hình 1.5*.

- Bộ nhớ có 1000 ô nhớ.
- Địa chỉ từ 000-999 (như vậy địa chỉ có 3 chữ số)
- Mỗi ô nhớ có khả năng lưu trữ được 5 chữ số
- Có hai thanh ghi liên quan tới bộ nhớ

MAR (Memory address register) có 3 chữ số chứa địa chỉ của ô nhớ cần truy nhập.

MDR (Memory date-register) có 5 chữ số chứa số liệu của ô nhớ có địa chỉ ở MAR trong phép truy nhập.



Hình 1.5: Bộ nhớ của máy tính mẫu

- Có 2 lệnh để truy nhập bộ nhớ

Lệnh đọc ô nhớ

- Để đọc một ô nhớ, máy CU đưa địa chỉ của ô nhớ cần đọc vào thanh ghi MAR (Ví dụ: 002 → MAR)

  CU ra lệnh đọc – READ (RD)

- Nội dung của ô nhớ có địa chỉ ở MAR được đưa ra thanh ghi MDR (Ví dụ: 19920 → MDR)

  Nội dung của ô nhớ không thay đổi.

Lệnh ghi vào ô nhớ

- Để ghi vào một ô nhớ, CU đưa địa chỉ của ô nhớ cần ghi vào thanh ghi MAR (Ví dụ: 002 → MAR)
- CU đưa số liệu vào thanh ghi MDR (Ví dụ: 20001 → MDR)

  CU ra lệnh ghi - WRITE (WR)

  Nội dung của thanh ghi MDR được chuyển vào ô nhớ có địa chỉ ở MDR.

b. Đơn vị điều khiển

Trong máy tính mẫu đơn vị điều khiển gồm:

- *Thanh đếm chương trình PC (Program counter)* chứa địa chỉ của lệnh sẽ được thực hiện. PC chứa được 3 chữ số 000-999.
- *Thanh ghi lệnh IR (Instruction register)* chứa lệnh đang thực hiện.

Độ dài của thanh ghi là 5 chữ số:

OP	operand

Trong đó, 2 chữ số đầu chứa mã lệnh, 3 chữ số sau chứa địa chỉ của số liệu hay số liệu tham gia phép tính.

Mỗi khi thực hiện xong 1 lệnh hay 1 phần của lệnh nội dung của PC thay đổi theo 2 cách theo trình tự hoặc rẽ nhánh.

Theo trình tự

PC ← (PC) + 1

Rẽ nhánh

PC ← operand (IR_{2-4})

- *Bộ giải mã lệnh ID (Instruction Decoder)* giải mã lệnh để nhận biết lệnh sẽ phải làm gì.
- *Bộ tạo tín hiệu điều khiển (RD, WR)* sinh ra các tín hiệu để điều khiển các bộ phận trong máy tính và các thiết bị ngoại vi nhằm thực hiện lệnh đã được giải mã.

c. Đơn vị số học

Đơn vị số học gồm:

- *ACC (Accumulator)* là thanh ghi chứa 1 toán hạng khi ALU thực hiện phép tính và chứa kết quả sau khi thực hiện
- *ALU* thực hiện các phép tính cộng trừ số học có dấu, Flag Register.
- *Thanh ghi cờ* có 3 cờ:

  + Cờ ZF = 1 nếu sau khi thực hiện phép tính (ACC) = 0, ngược lại ZF = 0

  + Cờ SF (negative) = 1 nếu sau khi thực hiện phép tính kết quả âm. Ngược lại SF = 0

  + Cờ OF (overflow): cờ tràn

d. Thiết bị vào ra

- Thiết bị vào: Máy vào bìa có địa chỉ 000
- Thiết bị ra: Máy đục bìa có địa chỉ 000

e. Bàn điều khiển Console

- Cho phép đưa số liệu vào ACC và bộ nhớ bằng tay (store)
- Khởi động chương trình (Start)
- Dừng chương trình (Stop)
- Hiển thị kết quả bằng đèn led

1.3.2. Hoạt động của máy tính mẫu

a. ***Tập lệnh của máy tính mẫu***

- Máy tính mẫu có 13 lệnh.
- Lệnh có độ dài 5 chữ số chia làm 2 phần:

+ Mã lệnh OP (operation code) (2 chữ số).

OP cho biết máy cần làm gì thực hiện phép tính nào.

+ Toán hạng operand (operation address) (3 chữ số).

Operand cho biết địa chỉ ô nhớ chứa số liệu hay số liệu trực tiếp tham gia phép tính.

- Sau đây là các lệnh chủ yếu của máy tính mẫu:

Mã ngữ	Mã máy	Các thao tác	Ý nghĩa
STOP	00	PC ← (M)	Dừng máy
LOAD	01	ACC ← (M)	Nạp nội dung của ô nhớ vào ACC
STORE	02	M ← (ACC)	Lưu nội dung ACC vào ô nhớ
ADD	05	ACC ← (ACC) + (M)	Cộng nội dung của ô nhớ với nội dung ACC và kết quả giữ ở ACC
SUB	06	ACC ← (ACC) - (M)	Trừ nội dung ACC đi nội dung của ô nhớ, kết quả giữ ở ACC
JMP	07	PC ← (IR_{2-4})	Nhảy không điều kiện
JZ	08	PC ←(IR_{2-4}) nếu ZF = 1 PC ←(PC)+1 nếu ZF = 0	Nhảy nếu kết quả phép tính là 0 (ACC) = 0
JS	09	PC ←(IR_{2-4}) nếu SF = 1	Nhảy nếu kết quả phép tính âm

		PC ←(PC)+1 nếu SF = 0	
JO	10	PC ←(IR_{2-4}) nếu OF = 1 PC ←(PC)+1 nếu OF= 0	Nhảy nếu kết quả phép tính tràn
IN	12	ACC ← bìa	Đọc một số có 5 chữ số vào ACC
OUT	13	đục bìa ←(ACC)	Đưa nội dung của ACC ra máy đục bìa

Ghi chú: Các lệnh 03,04,11 không dùng ở đây.

b. Chu kỳ lệnh

Quá trình nhận và thực hiện lệnh gọi là chu kỳ lệnh.

Mỗi chu kỳ lệnh bao gồm nhiều chu kỳ máy.

Một chu kỳ máy bao gồm nhiều nhịp.

Một chu kỳ lệnh máy tính mẫu thực hiện các công việc sau:

1. Nhận lệnh từ bộ nhớ
2. Thực hiện lệnh
3. Thay đổi nội dung của thanh ghi PC

Bước 1 là giống nhau cho tất cả các lệnh.

Mô tả hoạt động của các lệnh

- Các lệnh chuyển số liệu bao gồm lệnh nạp (LOAD) và lệnh lưu (STORE).
- Các lệnh số học gồm lệnh cộng (ADD) và lệnh trừ (SUB).

Ví dụ: Quá trình thực hiện lệnh cộng (ADD)

Giả sử (PC) = 050; (050) = 05063; (063) = 20000, (ACC)=01560

1. CU đọc ô nhớ 050

   050 → MAR

   ra lệnh READ

   05063 → MDR

2. CU chuyển nội dung của thanh ghi MDR sang IR

   05063 → IR

3. CU giải mã lệnh, nhận biết mã 05 là lệnh cộng nội dung ô nhớ 063 và ACC
4. CU đọc ô nhớ 063

063 → MAR (đưa nội dung của (IR_{2-4}) vào MAR)

ra lệnh READ

20000 → MDR

5. CU yêu cầu ALU thực hiện cộng nội dung của ACC với nội dung của MDR → TR. (ACC → TR , MDR → ACC)
6. Thực hiện phép cộng (ACC) với (TR)

   01560 + 20000 = 21560

   (ACC) = 21560
7. CU tăng nội dung của PC thêm 1

   (PC) = 050 + 1 = 051

- Các lệnh điều khiển chương trình

Một chương trình được đặt trong bộ nhớ dưới dạng một chuỗi các lệnh có địa chỉ liên tiếp nhau . Khi một lệnh đã được nhận vào CPU và thực hiện thì nội dung của thanh đếm chương trình PC sẽ thay đổi cho phép thực hiện tiếp chương trình. Có 2 trường hợp:

Trường hợp thực hiện lệnh kế tiếp theo thì:

PC ← (PC) + 1

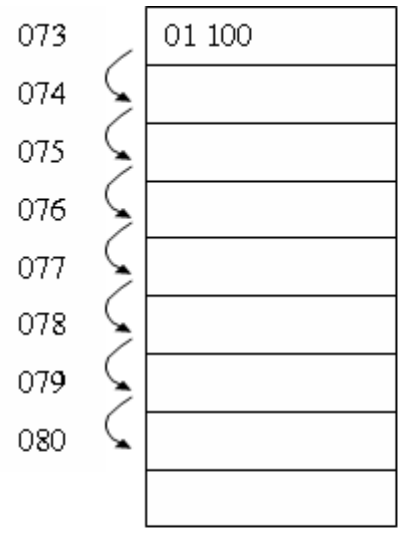
Trường hợp rẽ nhánh thì nội dung của PC được thay thế bằng toán hạng trong lệnh

PC ← operand .

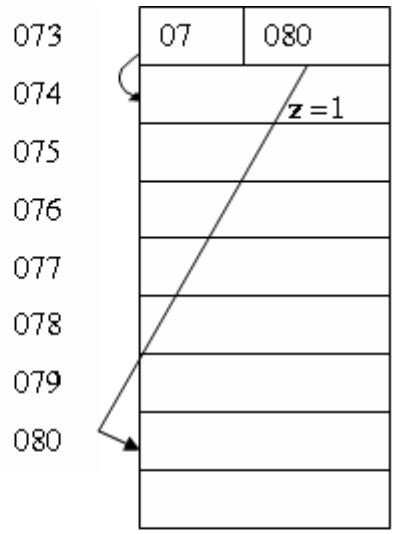
Các lệnh rẽ nhánh có khả năng thay đổi thứ tự từng lệnh.Trong máy tính mẫu

- Lệnh JMP là lệnh nhảy không điều kiện.
- Lệnh JZ, JS, JO là các lệnh nhảy có điều kiện.

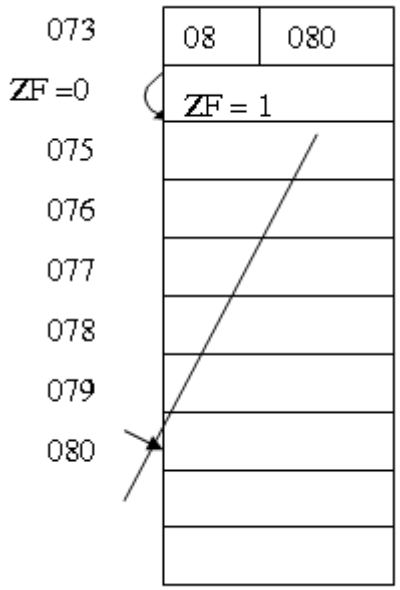
Ví dụ: lệnh JMP (73) = 07 080 *(Hình 1.6)*

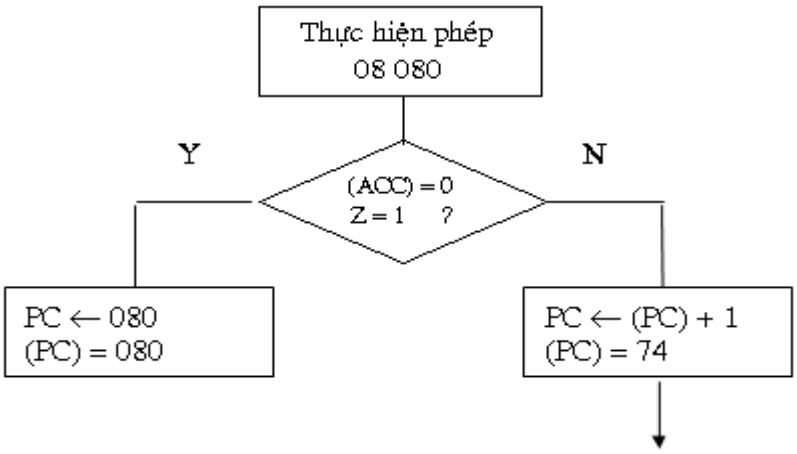


Các lệnh thực hiện liên tiếp nhau



Sau khi thực hiện lệnh ở ô nhớ 73 lệnh tiếp theo sẽ là ở ô nhớ 080 (nhảy không điều kiện)





Hình 1.6: Lệnh nhảy không điều kiện

- Các lệnh /vào ra

Các lệnh vào/ra thực hiện trao đổi số liệu giữa CPU và thiết bị ngoại vi.

Lệnh vào số liệu

12	X X X

mã lệnh địa chỉ thiết bị vào

Địa chỉ của thiết bị vào là 000

Chu kỳ lệnh của lệnh IN

Giả sử (PC) = 100; (100) = 12000; số liệu vào 10101

1. CU đọc ô nhớ 100

   100 → MAR

   ra lệnh READ

   12000 → MDR

2. (MDR) ← (IR)

   (IR) = 12000

3. CU giải mã lệnh, nhận biết 12 là lệnh nhận số liệu từ thiết bị vào.

4. CU đọc cửa vào 000

   10101 → ACC

5. (PC) ← (PC) + 1

Lệnh ra số liệu

13	XXX

Địa chỉ của thiết bị ra là 000

c. Lập trình bằng ngôn ngữ máy

Đối với một chương trình, trong bộ nhớ gồm 2 vùng lưu trữ:

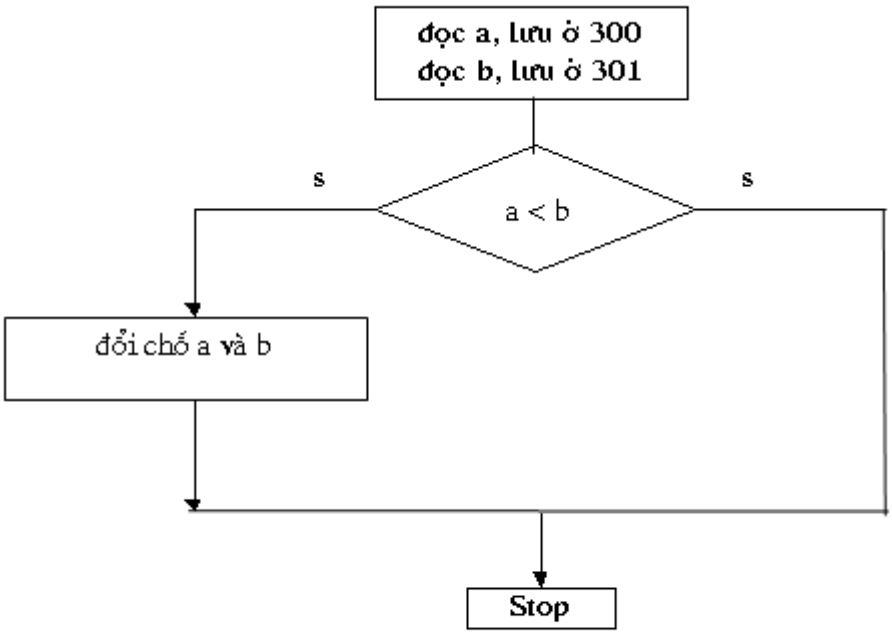
- Vùng lưu trữ chương trình
- Vùng lưu trữ dữ liệu

Ta tìm thấy sự hoạt động của máy qua ví dụ cụ thể sau *(hình 1.7):*

Đọc 2 số thập phân a và b từ thiết bị vào và xếp theo thứ tự từ bé đến lớn vào các ô nhớ 300 và 301

Chương trình lưu trữ trong bộ nhớ bắt đầu ở địa chỉ 100

Số liệu lưu trữ trong bộ nhớ bắt đầu ở địa chỉ 300



Hình 1.7: Lưu đồ thuật toán

Chương trình dưới dạng mã máy:

Địa chỉ	Mã máy	Mã ngữ	Ghi chú
100	12000	IN 000	đọc a
101	02300	STORE 300	lưu vào ô nhớ 300
102	12000	IN 000	đọc b
103	02301	STORE 301	lưu số b vào ô nhớ 301
104	06300	SUB 300	b - a
105	09107	JS	nhảy đến ô nhớ * nếu âm (tức là b<a)
106	00000	STOP	dừng máy nếu a < b
107	01300	LOAD 300	nạp a vào ACC
108	02400	STORE 400	lưu a vào 310
109	01301	LOAD 300	nạp b vào ACC
110	02300	STORE 300	lưu b vào 300
111	01400	LOAD 400	nạp a vào ACC
112	02301	STORE 301	lưu a vào 301
113	07106	JMP 106	nhảy về dừng

Một số khái niệm chung:

- Tất cả những gì mang tính chất vật liệu trong máy tính gọi là phần cứng (*Hardward*).
- Tất cả những gì mang tính chất phi vật liệu gọi là phần mềm (*Software*).
- Chương trình điều khiển phần cứng nạp trong mạch ROM gọi là *Firmware.*
- Ngôn ngữ máy: CPU chỉ hiểu các lệnh của ngôn ngữ máy. Đó là chuỗi các số nhị phân.
- Hợp ngữ dùng ký hiệu để biểu diễn lệnh, thanh ghi và ô nhớ.
- Ngôn ngữ máy và hợp ngữ là đặc trưng cho từng họ máy.
- Ngôn ngữ bậc cao cho phép người lập trình viết chương trình giống văn bản của ngôn ngữ tự nhiên, có thể chuyển đổi giữa các họ máy.
- Từ cách nhiền của ngýời sử dụng hệ điều hành là cách tổ chức, điều khiển các bộ phận (phần cứng và phần mềm) của máy tính hợp lý để tạo ra các dịch vụ có hiệu quả.

1.3.3. Cấu trúc Bus của MTĐT

- BUS địa chỉ
- BUS số liệu

- BUS điều khiển
- ROM (Read Only Memory)
- RAM (Random Acces Memory)

Qua phân tích hoạt động của máy tính đơn giản ta thấy các khối bên trong CPU và bên ngoài máy tính được kết nối bằng tập hợp tín các tín hiệu gọi là BUS hệ thống (System Bus). Tất cả các tín hiệu mang các bit địa chỉ gọi là BUS địa chỉ (BUS đ/c) (Address Bus). Tất cả các tín hiệu mang các bit số liệu gọi là BUS số liệu (BUS s/l) (Data Bus). Tất cả tín hiệu điều khiển gọi là BUS điều khiển (BUS đ/k) (Control Bus).

CHƯƠNG II

XỬ LÝ SỐ LIỆU, BIỂU DIỄN THÔNG TIN VÀ LỆNH TRONG MÁY TÍNH ĐIỆN TỬ

Chúng ta đã biết máy tính điện tử thực hiện các lệnh để xử lý các số liệu. Lệnh và số liệu được biểu diễn bởi chuỗi các bit nhị phân.

2.1. Hệ thống số

a. Dạng tổng quát của hệ nhị phân

$$N = a_{p-1}2^{p-1} + a_{p-2}2^{p-2} ... + a_1 2^1 + a_0$$

Trong đó a_0, a_1, a_2,...,a_{p-2} , a_{p-1} có giá trị 1 hay 0 là hệ số, p có giá trị là số nguyên, 2 là cơ số.

Ví dụ: số 1101b

$$1.2^3 + 1.2^2 + 0.2^1 + 1.2^0$$

$$8 + 4 + 0 + 1 = 13$$

Từ khái niệm bit ta có một số khái niệm sau:

- Tổ hợp của 4 bit gọi là nibble
- Tổ hợp của 8 bit gọi là byte
- Tổ hợp của 2 byte trở thành từ.
- Tổ hợp của 4 byte trở thành từ đúp
- Tổ hợp của 8 byte trở thành từ kép
- VXL (Bộ vi xử lý) xử lý số liệu có độ dài 1 byte gọi là VXL 8 bit.
- VXL xử lý số liệu có độ dài 2 bytes (Word) gọi là VXL 16 bit
- VXL xử lý số liệu có độ dài 4 bytes (Double Word) gọi là VXL 32 bit
- VXL xử lý số liệu có độ dài 8 bytes (Quad Word) gọi là VXL 64 bit.

b. Chuyển đổi số từ thập phân sang nhị phân và nhị phân sang thập phân

Muốn chuyển một số thập phân sang số nhị phân ta chuyển đổi phần nguyên và phần thập phân riêng. Thuật toán chuyển đổi rút ra từ dạng tổng quát của số nhị phân.

Ví dụ: chuyển đổi số thập phân 125,625

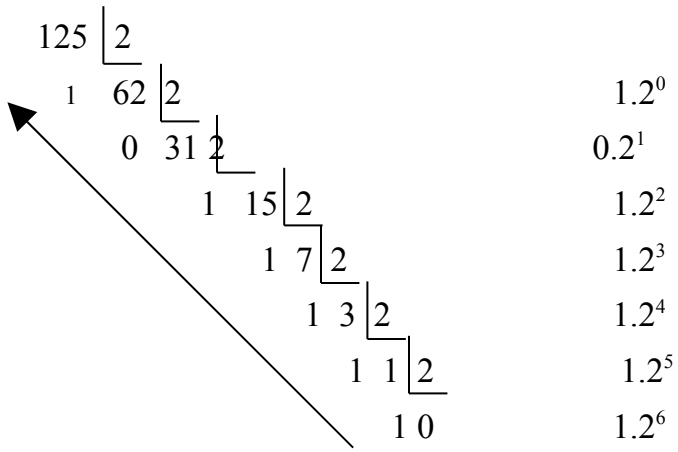
Ta thực hiện chuyển riêng phần nguyên và phần lẻ.

- **Phép chuyển đổi số thập phân sang số nhị phân**

Quy tắc chuyển phần nguyên:

- Lấy số cần chuyển chia cho 2 và ghi nhớ phần dư.
- Lấy thương của phép chia trên chia cho 2 và ghi nhớ phần dư.
- Làm như vậy cho tới khi được thương bằng 0.
- Lấy dãy các số dư theo trình tự đảo ngược ta được số nhị phân cần tìm.

Ví dụ:



Kết quả 125 = 1111101b

Quy tắc chuyển đổi phần thập phân:

- Lấy phần thập phân cần chuyển nhân 2, ghi phần nhớ phần nguyên
- Lấy phần thập phân của tích nhân 2, ghi nhớ phần nguyên của tích
- Làm như vậy cho tới khi được tích chẵn là 1
- Lấy dãy các phần nguyên của tích theo chiều thuận ta được số nhị phân của phần thập phân cần tìm

Ví dụ: 0,625 x 2 = 1, 250 1.2^{-1}

0,250 × 2 = 0,500 0.2^{-2}

0,500 × 2 = 1,000 1.2^{-3}

Kết quả 0,625 = 0,101b

Vậy 125,625 = 1111101,101b

- **Phép chuyển đổi hệ nhị phân sang thập phân**

Quy tắc rút ra từ dạng tổng quát của số nhị phân

Muốn chuyển số nhị phân sang thập phân ta tính các giá trị 2^i tương ứng với các chữ số khác 0 thứ i của số nhị phân.

Cộng các số tính được cho ta kết quả số thập phân cần tìm.

Ví dụ: $110101{,}1101 = 1.2^5 + 1.2^4 + 1.2^2 + 1.2^0 + 1.2^{-1} + 1.2^{-2} + 1.2^{-4}$

$= 32 + 16 + 4 + 1 + 0{,}5 + 0{,}25 + 0{,}0625$

= 53,8125

c. Hệ mười sáu (Haxa decimal)

Nếu dùng số nhị phân để biểu diễn số có giá trị lớn ta sẽ thu được số nhị phân quá dài. Trong thực tế người ta dùng nhóm 4 bit nhị phân thành 1 số hệ mười sáu. Để phân biệt với hệ khác thêm chữ H hay h vào sau số.

0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0100	3	1011	B
0011	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Ví dụ: 1110 1011 1100 0011 b = EBC3h

d. Biểu diễn số nguyên

Có 2 cách biểu diễn số nguyên:

- Biểu diễn số nguyên theo dấu và độ lớn
- Biểu diễn số nguyên theo số bù 2: cho phép biểu diễn số âm 1 cách thích hợp để thực hiện phép trừ

Số nhị phân	Mã bù 2	Mã hệ 2 có dấu và độ lớn
00000000	+0	+0
00000001	+1	+1
00000010	+2	+2
01111111	+127	+127
10000000	-128	-0
10000001	-127	-1
10000010	-126	-2
11111101	-3	-125
11111110	-2	-126
11111111	-1	-127

Nhận xét:

- Nếu biểu diễn số có dấu theo kiểu dấu và độ lớn (sign and magnitude) ta có 1 bit dùng làm dấu còn 7 bit biểu diễn giá trị của số. Một byte ta có thể biểu diễn số âm và số dương nằm trong khoảng -127...0, +0... +127
- Số bù 2 dùng cả 8 bit để biểu diễn giá trị của số được mã hóa. Một byte biểu diễn được số âm, số dương trong khoảng -128...0 ... + 127
- Một số nguyên biểu diễn theo hệ 2 có giá trị khác nhau nếu kiểu đó là mã để biểu diễn số theo kiểu hệ nhị phân có dấu và mã bù 2. Cả 2 cách biểu diễn này đều có bit D7 = 1 cho số âm và D7 = 0 cho số dương.

Ví dụ 1: 15 - 13 = 15 + (-13)

15	0000 1111
13	0000 1101
bù 1 của 13	1111 0010
+1	1
bù 2 của 13	1111 0011
Do đó:	
15	0000 1111
(-13)	1111 0011
2	1 0000 0010

Ví dụ 2: 12 - 13 = 12 + (-13)

12	0000 1100
(-13)	1111 0011
-1	1111 1111

Tìm giá trị của số bù 2 có bit dấu là 1: thực hiện phép đảo các bit rồi cộng với 1.

Ví dụ 3:

Số bù 2:	11111111
Số bù 1:	00000000
Giá trị của số:	00000001

2.2. Thuật toán các phép tính (cộng, trừ, nhân, chia)

a. Cộng 2 số nhị phân

0 + 0 = 0	1 + 1 = 0	nhớ 1
0 + 1 = 1	1 + 1 + 0 = 0	nhớ 1
1 + 0 = 1	1 + 1 + 1 = 1	nhớ 1

Ví dụ

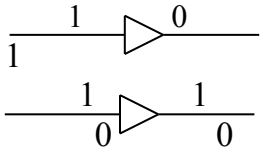
	nhị phân	thập phân
a =	01011101	93
b =	01001100	76
	10101001	169

b. Phép trừ nhị phân được thực hiện bằng phép cộng số bị trừ và số bù 2 của số trừ

Ví dụ: 93 - 76 = 93 + (-76)

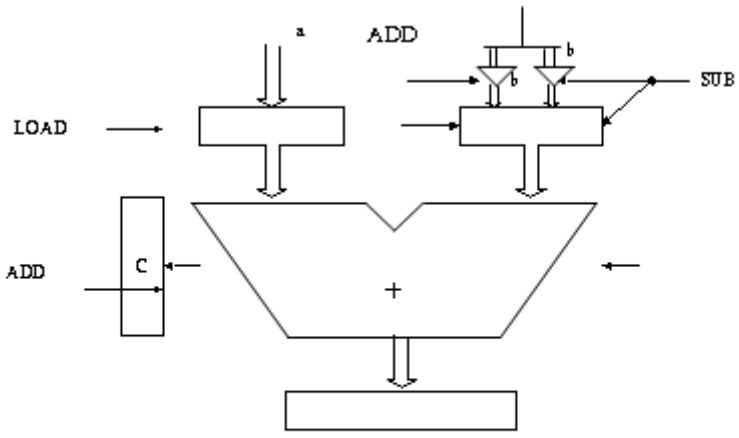
Trước hết tìm số bù 2 của 76

0100 1100 → 1011 0011 → +1 → 1011 0100



93	0101 1101	0101 1101
76	1011 0100	0100 1100
17	10001 0001	0001 0001

Nhận xét: Bộ cộng trừ 8 bit thể hiện trên hình *2.1*.



Hình 2.1: Bộ cộng trừ 8 bit

Cấu trúc của bộ cộng và bộ trừ về cơ bản giống nhau nên trong MTĐT người ta chỉ dùng một ALU thực hiện cả phép cộng và trừ.

Trong các bộ vi xử lý đơn giản (8 bit) người ta thường dùng ALU với cấu trúc trên.

c. Phép nhân

Quy tắc nhân cũng giống như hệ thập phân

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Ví dụ:

1001	Số bị nhân	
0110	Số nhân	
0000	Thành phần 1 của tổng tích lũy	
1001		2
1001		3
0000		4
0110110	Tổng tich lũy	

Nhận xét:

Mỗi lần nhân 1 bit khác 0 của số nhân với số bị nhân ta thu được chính số bị nhân.

Nếu dịch trái số lần tương đương với vị trí số khác 0 đó trong số nhân sẽ tạo ra một thành phần của tổng tích lũy.

Tổng các thành phần là kết quả của phép nhân.

Thuật toán cộng và dịch của phép nhân:

Nhân bit thấp nhất của số nhân với số bị nhân. Nếu LSB = 0 (Least Significant Bit) thì thành phần này bằng 0, nếu LSB khác 0 thì thành phần này chính bằng số bị nhân.

Mỗi thành phần thứ i tiếp theo của tổng tích lũy được tính tương tự nhưng phải dịch trái i bit.

số bị nhân	
0110	số nhân
1001	dịch trái số bị nhân 1 lần
1001	dịch trái số bị nhân 2 lần
110110	Tổng tích lũy

- Tính tổng của các thành phần tổng tích luỹ ta được tích cần tìm

d. Phép chia

Phép chia là phép ngược của phép nhân nên có thể thực hiện bằng phép trừ và dịch liên tiếp cho đến khi không thể trừ được nữa.

Để thực hiện được bằng ALU với các phần tử cộng và dịch ta thực hiện được thuật toán sau:

Đổi số chia thành số bù 2 của nó

Lấy số bị chia trừ đi số chia (cộng với số bù 2 của số chia)

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

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

Dịch trái kết quả thu được và làm lại bước 2 cho đến khi kết quả cuối cùng là 0 hoặc nhỏ hơn số chia

Ví dụ: 36 : 7 mã nhị phân của 7: 111b; mã bù 2 của 7: 1001b

36 : 7 = 5 dư 1

2.3. Cộng trừ số BCD (Binary Coded Decimal)

Người ta dùng 4 bit nhị phân để mã hóa 1 chữ số thập phân (BCD nén)

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

Cách biểu diễn số thập phân như vậy còn gọi là số thập phân nén (packed decimal number) . Như vậy hai chữ số có độ dài 1 byte, các bộ ALU tính số nhị phân có độ dài 1 byte, có thể xử lý 2 số thập phân mã BCD (BCD).

Muốn cộng 2 số BCD, trước hết cộng chúng như cộng 2 số nhị phân sau đó hiệu chỉnh kết quả cho đúng với biểu diễn số BCD.

Ví dụ 1:

09	0000 1001
05	0000 0101
14	0000 1110

Trường hợp này kết quả sai vì số 1110, không phải là số BCD

Ví dụ 2:

09	0000 1001
08	0000 1000
17	0001 0001

Trường hợp này kết quả sai vì có nhớ bit D3 sang bit D4 .

Trong máy vi tính họ Intel thì cờ nhớ phụ AF =1 (Auxiliary Flag). Cả 2 trường hợp trên phải thực hiện phép hiệu chỉnh thập phân (để kết quả là 2 số thập phân). Muốn vậy phải cộng kết quả với 6.

Ở *ví dụ 1*:

0000 1001	09
0000 0101	05
0000 1110	
0000 0110	
0001 0100	14

Ví dụ 2

0000 1001	09
0000 1000	08
0001 0001	
0000 0110	
0001 0111	17

Quy tắc cộng 2 số BCD

1, Nếu nửa thấp của byte kết quả biểu diễn số lớn hơn 9 hoặc có nhớ sang bit D4 thì phải cộng thêm 0110b vào nửa thấp và đưa AF = 1

2, Sau bước 1, nếu nửa cao của byte kết quả biểu diễn số lớn hơn 9 hoặc có nhớ thì phải cộng nửa cao với 0110b và đưa CF = 1.

Quy tắc trừ hai số BCD:

1, Nếu 4 bit thấp của byte kết quả biểu diễn số lớn hơn 9 hay cờ AF= 1 thì trừ đi (0110b) và lập cờ AF = 1.

2, Nếu 4 bit cao của kết quả biểu diễn số lớn hơn 9 hay CF = 1 thì trừ 4 bit cao đi 0110b và lập CF = 1.

2.4 Số dấu phẩy động

Nhược điểm của số dấu phảy cố định là khó biểu diễn các số quá lớn hay quá nhỏ.

Ví dụ: 6,02.1023

$8,76.10^{-24}$

Để khắc phục ta dùng số với dấu phẩy động

Dạng tổng quát

$$\delta.M.\alpha^E$$

δ : dấu của phần định trị

M: phần định trị

α: cơ số

E: số mũ

a. Số thập phân dấu phẩy động $\alpha = 10$

Ví dụ:

$125.000 = 125.000.10^0$ δ =+

M = 125.000

E = 0

$125.000 = 125.10^3$ δ =+

M = 125

E = 3

$125.000 = 0,125.10^6$ δ =+

M = 0,125

E = 6

$125.000 = 125.000.000.10^{-3}$ $\delta = +$

$M = 125.000.000$

$E = -3$

Như vậy số dấu phảy động được định nghĩa bởi 2 giá trị:

Phần định trị chứa những chữ số có nghĩa cần thiết để định nghĩa chính xác giá trị tương đối của số. Phần định trị là số có dấu thường dùng 1 chữ số để biểu diễn dấu

Phần số mũ cho biết độ lớn của những chữ số của phần định trị.

Phần số mũ, trong nhiều máy tính biểu diễn dấu như sau:

- Chọn 1 số để định nghĩa số mũ là 0
- Các số lớn hơn số đó là số mũ có dấu dương
- Các số nhỏ hơn là số mũ có dấu âm

b. Số nhị phân dấu phảy động $\alpha = 2$

Tương tự như số thập phân dấu phảy động:

$101.2^0 = 5.2^0 = 5$

$0,101.2^3 = 0,625.2^3 = 5/8 \,.\, 8 = 5$

$101\ 000.2^{-3} = 40.\ 1/8 = 40/8 = 5$

Một cách tương đối:

- Nếu phần định trị dịch về phía phải thì số mũ phải được tăng lên tương ứng.
- Nếu phần định trị dịch về bên trái thì số mũ giảm đi.

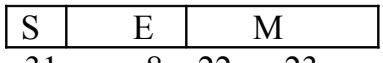
c. Chuẩn hóa

Chuẩn IEEE 754

$R = (-1)^s \,.\, (1 + M_1.2^{-1} + M_2.2^{-2} + ... + M_n.2^{-n}) \, 2^{E7.E6...E0 \, - \, 127}$

Các Format chuẩn:

Single Precision

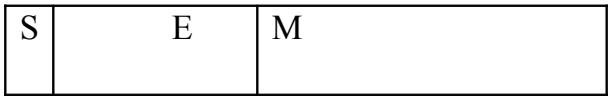


31 8 22 23 0

Double Precision

63 11 51 52

Extended Precision



79 15 63 64

2.5. Biểu diễn thông tin

2.5.1. Biểu diễn ký tự

Để biểu diễn thông tin mà con người dễ hiểu máy tính phải có giao diện thích hợp. Cách trao đổi thông tin truyền thống nhất là bộ chữ viết.

Mã ASCII (American Standard Code for Information Interchange)

Mã ASCII dùng 7 bit để mã hóa và bit D7 để kiểm tra chẵn lẻ.

6. Mã ASCII được chuẩn hóa bởi ANSI (American National Standard Institute và ISO, ISO 646).
7. Phân bổ mã trong chuẩn ASCII

Ký tự	**Mã HEX**
NULL	00
31 ký tự điều khiển	01-1F
Các dấu	20-2F
Số 0-9	30-39
Các dấu khác	3A-40
A-Z	41-5A
Các dấu khác	5B-60
a-z	61-7A
Các dấu khác	7B-7F

Mã ASCII mở rộng 8 bit - 256 ký tự

+Đủ để biểu diễn chữ cái của tiếng Đức, Pháp...

+ Tạm đủ cho tiếng Việt, nhưng thiếu đối với tiếng Hán, Hàn, Nhật...

b. Mã Unicode

Mã Unicode do hãng Xerox đề nghị

8. Dùng 2 byte để mã hóa ký tự. Không chứa các ký tự điều khiển
9. Phân bổ mã trong chuẩn Unicode
   - (8192) Mã chữ cái chuẩn
   - (4096) Mã toán học, ký hiệu kỹ thuật
   - (4096) Chữ tượng hình, Hán, Hàn, Nhật
   - (05633) Dành cho người sử dụng
   - (00512) Vùng tương thích
10. Mã tiếng Việt: mã âm tiết, mã phụ âm nhấn mạnh.

+ Các chữ cái có âm tiết là ký tự tổng hợp (composite character). Ví dụ: â = a và ^

+ Các phụ âm nhấn mạnh đ chỉ dùng 1 mã.

11. Để biểu diễn tiếng Việt cần

+ 33 chữ cái thường a ă â b c d đ e ê f g h i j k l m n o ô ơ p q r s t u ư v w x y z.

+ 33 chữ cái hoa.

+ 5 dấu thanh ‘,`, ? , ~ , .

2.5.2. Biểu diễn hình ảnh, âm thanh và các đại lượng khác

a. Biểu diễn hình ảnh

Đồ họa điểm (Pixel graphics)

- Biểu diễn hình ảnh bằng ma trận điểm

- Cách ghi thông tin của điểm ảnh lên tệp để lưu trữ gọi là khuôn dạng ảnh khác nhau ta có:

BMP	Bit Map ảnh nhị phân
PCX	PC paintbrush X
GIF	Graphic Image File

Đồ hoạ điểm có nhược điểm là chiếm quá nhiều bộ nhớ.

Đồ họa vector (Vector gralphics)

Biểu diễn hình ảnh bằng phương pháp này sẽ khắc phục nhược điểm của phương pháp trên.

- Biểu diễn hình ảnh từ các đối tượng cơ bản như điểm, đường thẳng, đa giác mặt, khối...

- Máy tính sẽ dựa vào công thức toán để dựng lại hình ảnh từ các đối tượng cơ bản.

- Cho phép biểu diễn hiển thị đối tượng 3 chiều (3D- graphics)

Biểu diễn hình ảnh động (video)

Có thể biểu diễn bằng đồ hoạ điểm ảnh hay đồ họa vector nhưng hình ảnh chuyển động có yếu tố thời gian.

Do thị giác con người có độ trễ nhất định nên chỉ cần 30 ảnh trong một giây là người ta có cảm giác hình ảnh chuyển động trơn tru. Như vậy để hiển thị ảnh

chuyển động máy tính cần cung cấp bộ nhớ để lưu 30 ảnh tĩnh và các thiết bị ngoại vi cũng cần đủ nhanh để xử lý số lượng 30 ảnh tĩnh trong 1 giây.

Để khắc phục người ta nén video số hóa lại, có nhiều phương pháp nén nên có nhiều khuôn dạng video, ví dụ như:

MPEG

Các thuật toán nén tập trung 2 yếu tố:

- Nén từng ảnh tĩnh

- Nén những phần không thay đổi theo thời gian.

Phương pháp định hướng đối tượng (vector) dùng để biểu diễn hình ảnh chuyển động đặc biệt là ảnh không gian ba chiều.

b. Biểu diễn âm thanh

Thính giác con người chỉ phân biệt được tối đa tần số fmax = 20 KHz. Theo Nyquist thì tần số trích mẫu phải bằng hoặc lớn hơn 2 lần tần số sóng âm thanh.

Tần số trích mẫu ngày nay theo chuẩn của công nghệ CD (Compact Disk) là 44 KHz. Độ lớn của biên độ trích mẫu được mã hóa bằng 8 bit nhị phân. CD dùng 16 bit.

Khuôn dạng âm thanh thường gặp là WAVE, MPEG.

Âm thanh cũng có thể phân tích thành nhiều đối tượng và mã hóa. Kỹ thuật này dùng để nhận dạng tiếng nói và tổng hợp tiếng nói.

c. Biểu diễn các đại lượng vật lý khác

Nguyên tắc chung:

Chuyển các đại lượng vật lý sang tín hiệu điện.

Chuyển tín hiệu điện tương tự sang tín hiệu số qua ADC (Analog- Digital Converter)

2.6. Các dạng lệnh trong máy tính điện tử

Dạng tổng quát của lệnh:

OP	Add1	Add2	Add3	Add4

Mã lệnh: OP.

Địa chỉ 1: chứa toán hạng 1.

Địa chỉ 2: chứa toán hạng 2.

Địa chỉ 3: chứa kết quả.

Địa chỉ 4: chứa địa chỉ của lệnh tiếp theo.

a. MTĐT lệnh có 4 địa chỉ

Add3 (Add1) * (Add2)

PC (Add4)

b. MTĐT lệnh có 3 địa chỉ

Add3 (Add1) * (Add2)

PC (PC + 1)

c. MTĐT lệnh có 2 địa chỉ

Add2 (Add1) * (Add2)

PC (PC + 1)

d. MTĐT lệnh có 1 địa chỉ

ACC (Add1) * (ACC)

PC (PC + 1)

e. MTĐT lệnh có 0 địa chỉ

Tất cả các lệnh đều thao tác với số liệu trên ngăn xếp. Ngăn xếp là tập hợp các ô nhớ hay thanh ghi làm việc theo nguyên lý LIFO. Trong CPU có một thanh ghi luôn chỉ đỉnh của ngăn xếp SP (Stack Pointer).

(SP) ((SP)) * ((SP-1))

PC (PC + 1)

CHƯƠNG III

BỘ NHỚ

3.1. Giới thiệu chung về bộ nhớ

Bộ nhớ là phương tiện lưu trữ thông tin bao gồm chương trình và số liệu trong hệ thống tính toán. Qua phần giới thiệu chung này, người đọc sẽ có hình ảnh tổng quan và những khái niện cơ bản nhất về bộ nhớ.

3.1.1. Một số thông số chính của mạch nhớ

Độ dài của ô nhớ: Độ dài của ô nhớ cho biết số bit chứa trong ô nhớ, có thể tính bằng bit, byte (8 bit), từ (16 bit), từ đúp (32 bit) hay từ kép (64 bit).

Dung lượng (Capacity) của mạch nhớ xác định số bit hay byte hay từ cực đại mà mạch nhớ có thể chứa. Giả sử mạch nhớ có n bit địa chỉ và mỗi từ có độ dài là m, như vậy mạch nhớ có dung lượng 2^n (m bit được tổ chức như 2^n từ, mỗi từ m bit). N bit địa chỉ chỉ n đầu vào địa chỉ của mạch nhớ. Với n bit địa chỉ, một ô nhớ duy nhất trong 2^n ô được xác định. Tổng số ô nhớ là $L = 2^n$. Như vậy, số lượng ô nhớ trong mạch nhớ là lũy thừa cơ số 2. Với L cho trước, số lượng bit địa chỉ cần thiết để phân biệt L vị trí nhớ là n=log2L. Đơn vị đo dung lượng bộ nhớ thông thường nhất là: Byte(B), KiloByte ($1KB=2^{10}B$), MegaByte ($1MB=2^{20}B$), GigaByte ($1GB=2^{30}B$), TetraByte ($1TB=2^{40}B$)...

Thời gian thâm nhập (Acces Time) là thời gian từ thời điểm áp địa chỉ tới BUS địa chỉ khi nội dung của ô nhớ đó được đưa ra BUS số liệu, ký hiệu là t_A , thời gian này phụ thuộc vào công nghệ chế tạo và cấu trúc mạch nhớ.

Chu kỳ đọc (Read Cycle) là thời gian kể từ khi áp địa chỉ để đọc ô nhớ cho đến khi có thể áp địa để đọc ô nhớ tiếp theo, ký hiệu là t_{RC}. Đó là thời gian ngắn nhất giữa hai lần đọc mạch nhớ.

Chu kỳ ghi (Write Cycle) là thời gian kể từ khi áp địa chỉ để ghi ô nhớ cho đến khi có thể áp địa để ghi ô nhớ tiếp theo, ký hiệu là t_{WC}. Đó là thời gian ngắn nhất giữa hai lần ghi mạch nhớ.

Tần số của mạch nhớ là lượng thông tin lớn nhất có thể đọc hay ghi vào mạch nhớ trong thời gian 1 giây.

$$f = 1/t_M$$

Trong đó $t_M = \text{Max}\ (t_{RC}, t_{WC})$

3.1.2. Phân loại bộ nhớ

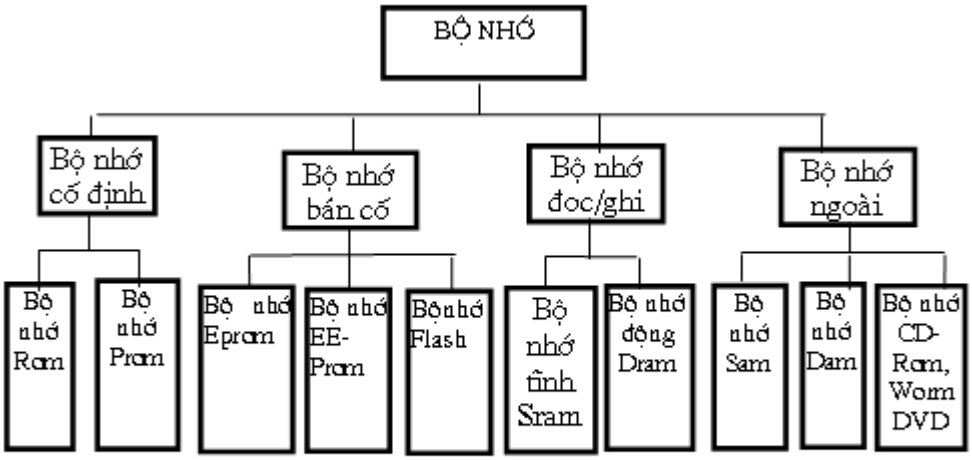
Nói chung, bộ nhớ được phân loại theo một vài thuộc tính *(Hình 3.1 biểu diễn một cách phân loại các bộ nhớ)*. Sau đây là một số cách phân loại bộ nhớ:

Theo chức năng bộ nhớ được chia thành hai loại:

- Bộ nhớ trong (bộ nhớ chính)
- Bộ nhớ ngoài. (bộ nhớ phụ)

Dựa trên thời gian ghi và cách ghi bộ nhớ trong có thể chia thành:

- Bộ nhớ cố định
- Bộ nhớ bán cố định
- Bộ nhớ đọc/ ghi



Hình 3.1: Các loại bộ nhớ

a. Bộ nhớ cố định

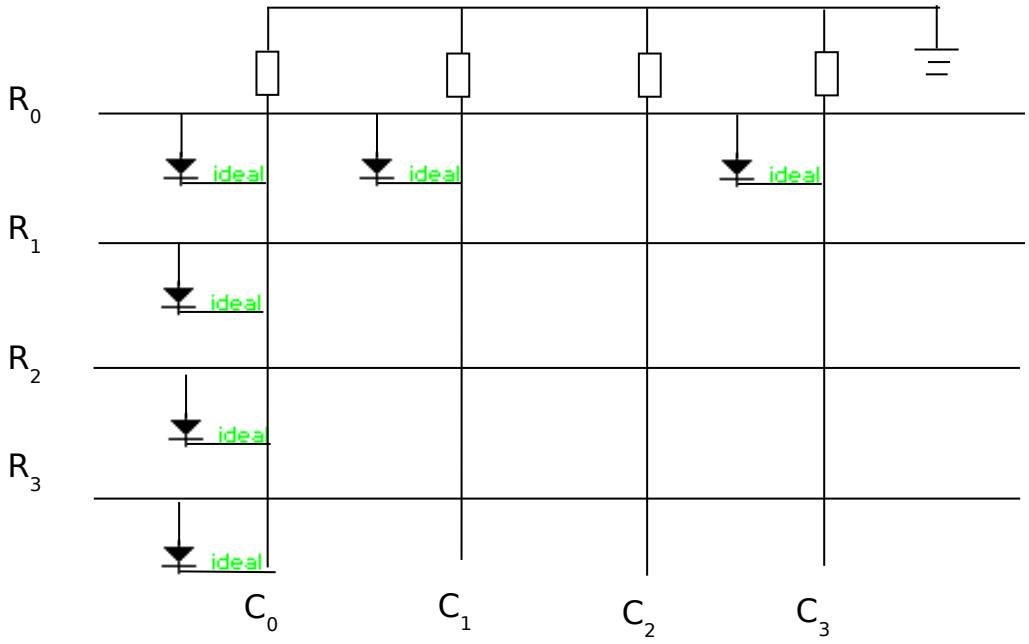
ROM (Read Only Memory)

Bộ nhớ có nội dung ghi sẵn một lần khi chế tạo được gọi là bộ nhớ cố định và được ký hiệu là ROM. Việc ghi được thực hiện bằng mặt nạ. Một phần tử nhớ trong ROM thường đơn giản hơn nhiều so với một mạch lật trong bộ nhớ đọc /ghi, vì trạng thái của nó cố định. Chương trình điều khiển của hầu hết các hệ vi tính được giữ trong ROM. Một phần tử nhớ (một bit nhớ) thường được thực hiện bởi một 1 Diode, 1 Transistor lưỡng cực hay 1 Transistor trường. *Hình 3.2* mô tả nguyên lý một ma trận nhớ đơn giản gồm 4 hàng và 4 cột (4 từ, mỗi từ 4 bit) . Các dây hàng

tương ứng với dậy chọn từ nhớ, các dây cột tương ứng với bit nhớ trong từ. Vị trí tương ứng giá trị logic 1 có nối diode, vị trí tương ứng với 0 logic để trống.

Khi R_0 được chọn các diod dẫn có dòng qua các cột tương ứng có logic 1

R_0 chọn $C_0C_1C_2C_3 = 1101$



H×nh 3.2: CÊu tróc m¹ch nhí ROM

PROM (Programmable Read Only Memory)

Một dạng của ROM là PROM. Giống như ROM, PROM chỉ có thể ghi một lần.

Tất cả các bit của PROM sau khi chế tạo cố định ở 0 hay 1, phụ thuộc vào loại mạch. Với việc sử dụng một thiết bị ghi (bộ ghi PROM) những bit mong muốn có thể ghi giá trị ngược lại. Giá trị của các bit một khi đã ghi, không thay đổi được nữa. Cấu trúc phần tử PROM cũng tương tự như ROM. Các diode nối tiếp với cầu chì điện tử được nối ở tất cả các nút. Khi chưa ghi tất cả các bit có giá trị 1 logic. Thiết bị ghi sẽ "đốt cháy" các cầu chì ở các vị trí muốn có giá trị logic 0.

- Diode ở tất cả các vị trí
- Diode nối với cầu chì
- Nếu có xung điện ghi cầu chì sẽ bị cháy

b. Bộ nhớ bán cố định

EPROM (Erasable Programmable Read Only Memory)

Đây là bộ nhớ xóa được bằng tia cực tím và ghi lại được. Số lần xóa và ghi lại không hạn chế. Chúng có thời gian ghi lớn hơn rất nhiều so với bộ nhớ đoc/ghi. Dưới tác dụng của tia cực tím tất cả các ô nhớ bị xóa cùng một lúc, mạch nhớ phải được đưa ra khỏi hệ vi tính để xóa. Điều thuận tiện ở bộ nhớ bán cố định cũng như ROM, là bộ nhớ tuy bất biến khi dùng nhưng vẫn có thể ghi lại được. Một bộ nhớ bất biến là bộ nhớ có nội dung không bị mất khi nguồn điện bị ngắt.

Nội dung của bộ nhớ không bất biến sẽ thay đổi khi mất nguồn nuôi. Thuật ngữ "ROM" thường được sử dụng ở đây để chỉ bộ nhớ cố định và bộ nhớ bán cố định vì chúng đều có tính chất bất biến. Cấu trúc của một bit nhớ là một Transistor MOS có thêm cửa thả nổi (Floating Gate) *(Hình 3.3)*.

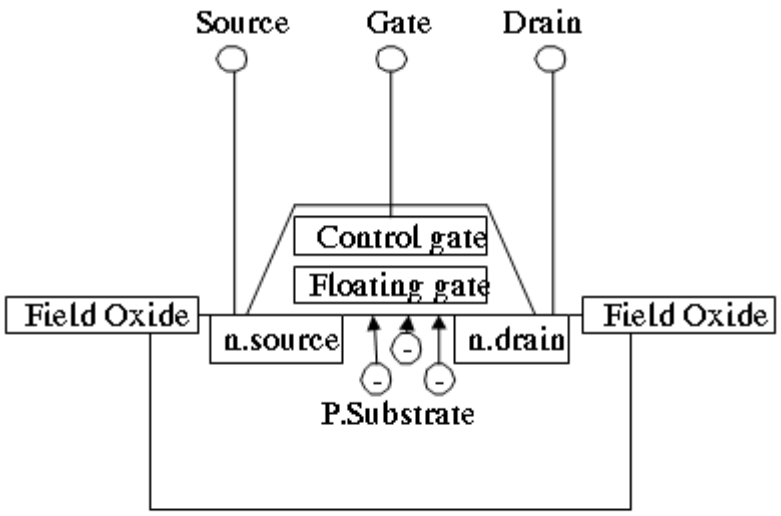
Cách nối các cực của Transistor:

- Cực nguồn nối vói mức logic 1
- Cực cửa nối với dây chọn từ
- Cực máng nối với dây chọn bit

Nếu cực thả nối không có điện tích thì Transistor hoạt động bình thường. Khi dây từ được kích thì cực cửa có điện tích dương làm cho cực nguồn và cực máng thông, dây bit có mức logic 1. Nếu cực cửa thả nổi có điện tích âm, Transistor đóng kể cả khi dây từ được chọn.

Muốn nạp giá trị logic 0 vào bit nhớ thì phải đưa điện tích âm vào cửa thả nổi bằng cách đưa xung điện có biên độ khoảng 20V giữa cực cửa và cực máng trong khoảng thời gian từ 5ms đến 50ms tuỳ theo loại EPROM. Dưới tác động của xung điện này các điện tử sẽ có năng lượng đủ lớn đi qua lớp cách điện giữa đế và cực thả nổi. Các điện tử sẽ được tích lại trong cửa thả nổi sau khi xung điện được cắt.

Muốn xoá toàn bộ nội dung của mạch EPROM, ta phải đưa mạch vào đèn tia cực tím, thời gian khoảng 20 phút. Dưới tác động của tia cực tím, các điện tử ở cực tả nổi hấp thụ năng lượng và nhảy lên mức năng lượng cao hơn và rời khỏi cực thả nổi. Như vậy, sau khi xoá tất cả các bit của mạch EPROM có giá trị logic 1.



Hình 3.3: Cấu tạo bit EPROM

Source : cực nguồn

Drain : cực máng

Field Oxide

n-Source: vùng nguồn điện tử

n-Drain : vùng máng điện tử

P-Substrate đế bán dẫn loại lỗ

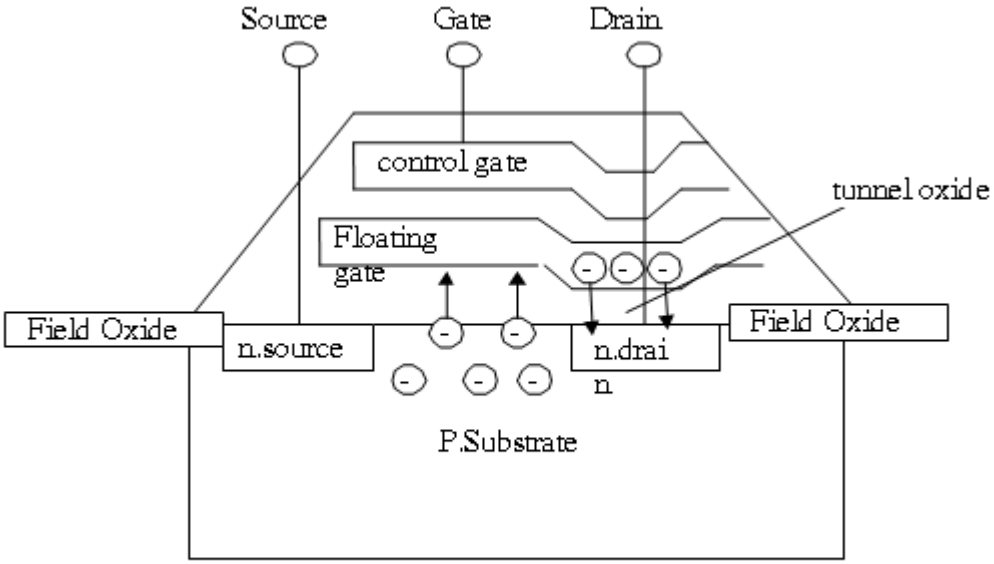
Có thêm cửa nổi (Floating gate) so với các transistor MOS thường

EEPROM (Electrical Erasable Programmable Read Only Memory

EEPROM cũng tương tự EPROM, có thể ghi được nhiều lần, có nghĩa là ghi lại và sử dụng lại, nhưng EEPROM không xoá bằng tia cực tím mà bằng xung điện nên khi xóa vẫn để trong mạch điện. Về nguyên lý cấu trúc bit nhớ của EEPROM cũng là Transistor với cực thả nổi nhưng có lớp oxyt mỏng giữa cực thả nổi và cực *máng (Hình 3.4)*.

Để nạp giá trị logic 0 vào bit nhớ cũng phải thực hiện tương tự như EPROM

Để xoá các bit về 1 cần đưa điện thế - 20V giữa cực cửa và cực máng làm cho điện tử từ cực thả nổi chảy về cực máng qua kênh màng mỏng oxyt. Dòng điện tử này không được quá lâu để cực thả nổi trở thành nơi tích điện dương.



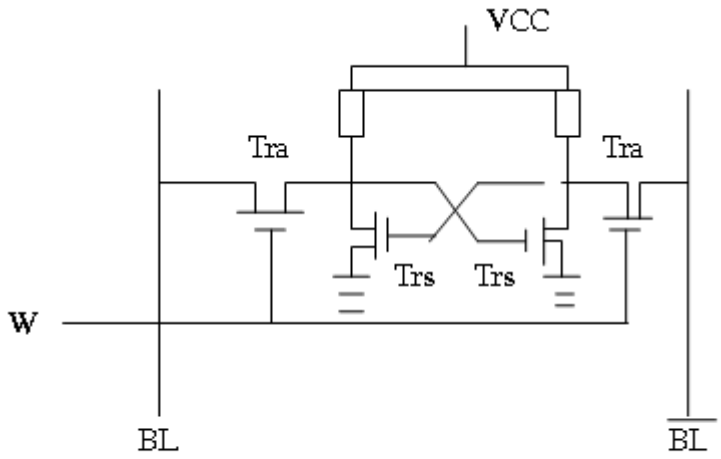
Hình 3.4: Cấu tạo của bit EEPROM

c. Bộ nhớ đọc/ ghi

RAM (Radom Access Memory)

Bộ nhớ có thể ghi và đọc nhiều lần, với thời gian ghi ngắn cỡ vài chục đến vài trăm nano giây. Trong các hệ vi tính, bộ nhớ đọc/ghi được sử dụng để cất giữ chương trình, kết quả trung gian...

SRAM (Static RAM)

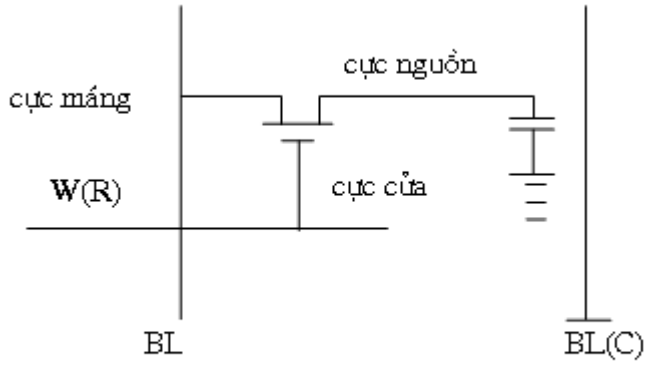


Hình 3.5: Cấu tạo của bit nhớ SRAM

SRAM là bộ nhớ đọc/ghi có nguyên lý hoạt động tĩnh. Bộ nhớ đọc/ghi tĩnh có các ô nhớ cấu tạo tương tự như mạch lật *(Hình 3.5)*. SRAM không cần điều khiển phức tạp như DRAM, tốc độ nhanh hơn (tA = 10 ns), tin cậy hơn nhưng giá thành tính theo bit đắt hơn DRAM, được sử dụng làm bộ đệm cache trong các bộ vi xử lý hay máy vi tính. Bit nhớ được tạo bởi Flip-Flop.

DRAM (Dynamic RAM)

Cấu trúc của bit DRAM không phải tạo từ mạch FLIP-FLOP mà từ một chiếc tụ bán dẫn được nối *(Hình 3.6)*. Do điện tích trên tụ điện có thể bị dò qua công tắc "không lý tưởng", DRAM cần được khôi phục nội dung đều đặn, nếu không nội dung sẽ mất. Mạch DRAM cần điều khiển phức tạp hơn SRAM, nhưng có dung lượng lớn, giá thành rẻ nên được dùng làm bộ nhớ chính trong các máy vi tính.



Hình 3.6: Cấu tạo của bit nhớ DRAM

d. Bộ nhớ ngoài

SAM (Sequencial Access Memory**)**

SAM là bộ nhớ ngoài thâm nhập trình tự. Thời gian thâm nhập phụ thuộc vào vị trí của thông tin trên phương tiện mang tin. Ví dụ Tap Cartridges dùng như thiết bị lưu trữ dữ liệu của máy vi tính.

DAM (Direct Access Memory)

DAM cho phép thâm nhập tới bất cứ vùng dữ liệu nào cần. Thời gian thâm nhập phụ thuộc vào vị trí của thông tin trên phương tiên mang tin. Ví dụ thiết bị đĩa cứng, đĩa mềm thuộc loại thiết bị ngoại vi kiểu DAM.

CD-ROM (Compact Disk Read Only Memory)

CD-ROM là loại thiết bị đĩa dựa trên nguyên lý quang laser. Dung lượng của đĩa CD-ROM rất lớn có khả năng chứa tới 650 MByte.

WORM (Write Once Read Many)

WORM cho phép người sử dụng ghi thông tin một lần để đọc nhiều lần. Kiểu đĩa này rất thuận tiện cho việc sao chép phần mềm hay dự liệu khi triển khai các ứng dụng tin học.

DVD (Digital Versail Disk)

DVD cho phép ghi nhiều lớp thông tin trên một đĩa làm cho dung lượng của đĩa tăng lên đáng kể so với CD-ROM. Một đĩa DVD có thể lưu trữ lượng thông tin gấp 17 lần một đĩa CD-ROM.

Những mạch nhớ bán dẫn hiện nay có sức chứa giới hạn. Trong một vài ứng dụng, một bộ nhớ không những cần có dung lượng đủ lớn, mà còn cần được tổ chức để có số lượng từ và số lượng bit trong một từ như mong muốn. Nói chung, trong bộ nhớ có nhiều vi mạch nhớ được nối ghép lại để có độ dài từ và tổng số từ cần thiết.

Những vi mạch nhớ bán dẫn được thiết kế sao cho có đầy đủ những chức năng của một bộ nhớ như:

- Một ma trận các phần tử nhớ, mỗi phần tử chứa một bit
- Phần mạch logic để giải mã địa chỉ cho ô nhớ
- Phần mạch lôgic cho phép đọc/ghi được nội dung của ô nhớ
- Bộ đệm vào, bộ kích ra
- Phần mạch mở rộng địa chỉ

3.1.3. Phân cấp bộ nhớ

Phân cấp bộ nhớ được thể hiện trên *hình 3.7*.

Quan sát hệ thống nhớ từ CPU ra ngoài ta có các thành phần nhớ sau:

1. Các thanh ghi đa năng chứa một toán hạng hay kết quả trung gian, được điều khiển bằng phần cứng

2. Bộ nhớ đệm Cache chứa mảng lệnh và số liệu được sử dụng trong thời gian gần nhất, được điều khiển bằng phần cứng và chương trình. Bộ nhớ cache đặt giữa CPU và bộ nhớ chính. Bộ nhớ cache chứa một phần bản sao của bộ nhớ chính. Khi CPU thâm nhập vào dữ liệu nó đưa địa chỉ tới bộ điều khiển Cache, sau đó một trong hai quá trình sẽ xảy ra.

- Trúng (cache hit): nếu địa chỉ tìm thấy trong Cache

- Trượt (cache miss): nếu địa chỉ không có trong Cache

Khi trượt một khối nhớ từ bộ nhớ chính sẽ được đưa vào thay thế cho một đường (khối) của Cache. Đường nào sẽ được chọn để thay dựa trên hai nguyên lý sau:

- Cục bộ theo thời gian: nếu CPU thâm nhập vào một ô nhớ thì có xác suất cao nó sẽ thâm nhập ô nhớ đó trong tương lai.

- Cục bộ theo không gian: nếu CPU thâm nhập vào một ô nhớ thì có xác suất cao nó sẽ thâm nhập các lệnh và dữ liệu đặt sát các vị trí đó trong tương lai.

Trường hợp ghi vào Cache dữ liệu sẽ được ghi vào bộ nhớ chính, ta phân biệt hai trường hợp sau:

- Khi ghi vào Cache thì đồng thời ghi vào bộ nhớ chính, phương pháp này gọi là ghi xuyên (Write through)

- Khi ghi chỉ ghi vào bộ nhớ Cache, dữ liệu từ Cache sẽ được chuyển vào bộ nhớ chính tại một thời điểm thích hợp sau đó (ví dụ khi chuyển dữ liệu từ bộ nhớ chính ra thiết bị ngoại vi).

Việc ánh xạ giữa bộ nhớ Cache và bộ nhớ chính có thể tổ chức theo phương pháp khác nhau:

- Cache ánh xạ trực tiếp (Direct mapping cache)

- Cache ánh xạ liên kết toàn phần (Full associative mapping cache)

- Cache ánh xạ liên kết cụm (Set associative mapping cache)

Nội dung về bộ nhớ Cache sẽ được nghiên cứu kỹ hơn trong cấu trúc máy II

3. Bộ nhớ trong (bộ nhớ chính) chứa chương trình và số liệu đang thực hiện

4. Bộ nhớ ngoài lưu trữ chương trình và số liệu với khối lượng lớn. Nó cũng chứa phần nhớ ảo, khi máy tính chạy trong chế độ địa chỉ ảo.

Nếu đánh số phân cấp theo giá trị tăng dần từ trong CPU ra ngoài, ta có nhận xét sau:

- Thời gian thâm nhập của bộ nhớ có mức phân cấp càng thấp thì càng nhỏ

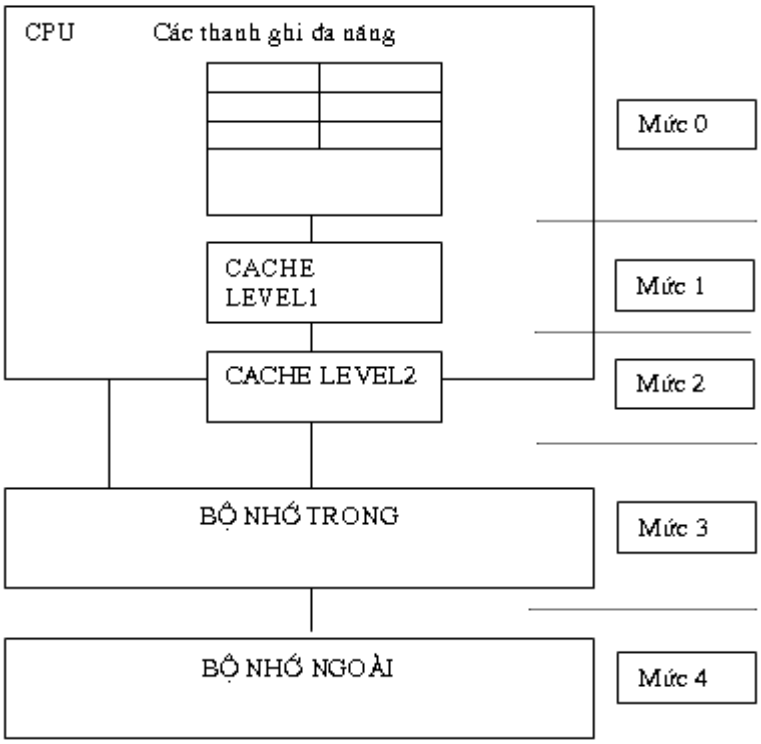
$$t_{Ai} < t_{Ai+1}$$

- Giá thành tính theo bit của bộ nhớ có mức phân cấp càng thấp thì càng cao

$$c_i > c_{i+1}$$

- Dung lương của bộ nhớ có mức phân cấp càng thấp thì càng nhỏ

$$S_i < S_{i+1}$$



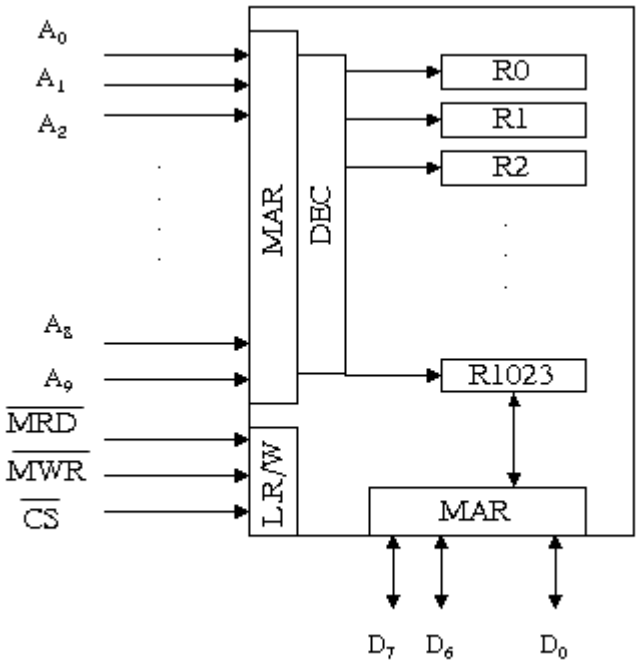
Hình 3.7 Phân cấp bộ nhớ

3.2. Cấu trúc của bộ nhớ bán dẫn

3.2.1. Cấu trúc của một mạch nhớ đọc/ghi tĩnh (SRAM)

Tập hợp của nhiều thanh ghi trong một mạch vi điện tử gọi là mạch nhớ.

Ví dụ: mạch nhớ 1KB



a. Các thành phần cơ bản của một mạch nhớ

Tập hợp các ô nhớ

Các ô nhớ được tổ chức theo 2 phương pháp

Phương pháp chọn tuyến tính (linear Selection)

- Tổ chức theo từ
- Số hàng bằng số từ
- Phải có bộ giải mã 1 từ n để có 1 đầu ra duy nhất từ một tổ hợp n đầu vào
- Mạch nhớ kiểu chọn tuyến tính chỉ có 1 bộ giải mã hàng nhưng bộ giải mã phức tạp, giá thành cao.
- Thời gian thâm nhập nhanh
- Phương pháp này chỉ dùng với bộ nhớ dung lượng nhỏ.

Phương pháp chọn trùng hợp (Coincident Selection)

Ô nhớ chỉ được chọn nếu đồng thời được chọn hàng và cột.

- Các bộ giải mã đơn giản giá thành rẻ

- Thời gian thâm nhập dài hơn (chậm). Giải mã gồm hai bước giải mã hàng trước, giải mã cột sau

Phương pháp này dùng cho các mạch nhớ dung lượng lớn

Bộ giải mã địa chỉ cho phép chọn tới từng ô nhớ (DEC)

Phần logic đọc ghi (L.R/W) cho phép đọc/ghi nội dung của từng ô nhớ

Bộ đệm địa chỉ (MAR) và đệm số liệu (MDR) cho phép ghép nối mạch nhớ với các mạch logic khác hay ghép các mạch nhớ (với nhau trong hệ thống máy vi tính).

Phần chọn mạch cho phép mở rộng dung lượng bộ nhớ (CS) bằng cách ghép nhiều mạch nhớ với nhau.

b. Các tín hiệu của mạch nhớ 1kB

- Địa chỉ A0...A9 (Số ô nhớ có thể địa chỉ $2^{10} = 1024$)
- Tín hiệu đọc $\overline{MRD}$ đưa nội dung của ô nhớ ra $\overline{MDR}$
- Tín hiệu ghi $\overline{MWR}$ ghi số liệu vào một ô nhớ
- $\overline{CS}$ tín hiệu chọn mạch
- Các tín hiệu số liệu D0...D7 (2 chiều, 3 trạng thái)

c. Quá trình đọc ghi mạch nhớ

Quá trình đọc mạch nhớ

CPU đưa địa chỉ vào mạch nhớ (A0...A9)

Đưa tín hiệu CS $\overline{\text{về}}$ tích cực ($\overline{CS} = 0$)

Đưa tín hiệu đọc về tích cực ($\overline{MRD} = 0$)

Sau thời gian thâm nhập t_A, nội dung của ô nhớ cần đọc sẽ ở đầu ra D0... D7.

Quá trình ghi mạch nhớ

- CPU đưa địa chỉ của ô nhớ cần ghi vào mạch nhớ (A0...A9)
- Đưa tín hiệu $\overline{CS}$ về tích cực ($\overline{CS} = 0$)
- Đưa số liệu vào D0... D7
- Ra lệnh ghi đưa tín hiệu $\overline{MWR}$ về tích cực, số liệu từ D0...D7 được ghi vào ô nhớ có địa chỉ ở A0...A9

3.2.2. Nguyên lý cấu trúc của một mạch DRAM

Cấu trúc của mạch nhớ DRAM so với mạch nhớ SRAM có các đặc tính như thường tổ chức theo bit (hay cụũm bit) và địa chỉ tổ chức theo kiểu dồn kênh.

Ví dụ mạch DRAM 1 Mbit, cần 20 đường địa chỉ A0...A19

Để tiết kiệm chân nối người ta dùng phương pháp dồn kênh (Multiplex)

A0... A9 địa chỉ hàng chọn bởi $\overline{RAS}$ (Row Address Strob)

A10...A19 địa chỉ cột chọn bởi $\overline{CAS}$ (Column Address Strob)

Mạch nhớ DRAM được làm tươi khi đọc dữ liệu, ghi dữ liệu, hay bằng cách làm tươi cả hàng...

Trong ví dụ trên mạch nhớ có dung lượng 1Mbit, để được bộ nhớ có dung lượng 1MB phải ghép 8 mạch nhớ 1Mbit với nhau.

Để xây dựng bộ DRAM đòi hỏi điều khiên phức tạp hơn nên trong công nghiệp, người ta đã sản xuất ra các mạch điều khiển DRAM. Ví dụ mạch điều khiển bộ nhớ DRAM 82C212 có khả năng quản lý 4 băng nhớ DRAM, độ rộng 16 bit và 2 bit kiểm tra chẵn lẻ.

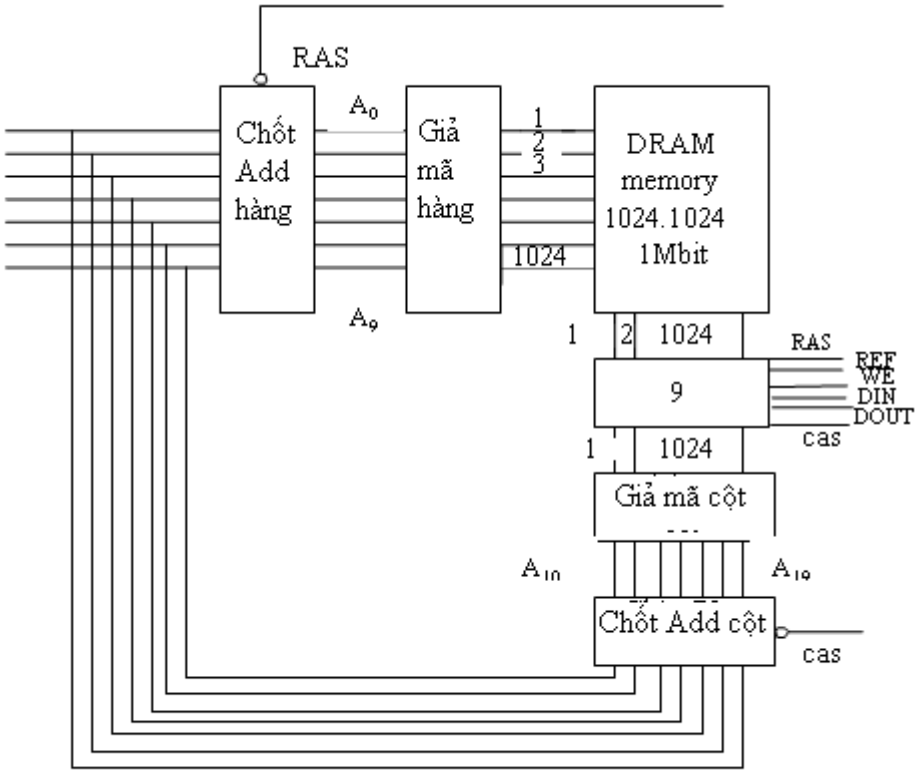
Các thông số cơ bản của mạch DRAM được mô tả trên *hình 3.9*

T_{ras} là thời gian từ khi tín hiệu RAS trở thành tích cực cho tới khi số liệu ổn định ở đầu ra

T_{cas} là thời gian từ khi tín hiệu CAS trở thành tích cực cho tới khi số liệu ổn định ở đầu ra

T_{pr} là thời gian từ khi số liệu ổn định cho tới khi có thể thay đổi tín hiệu RAS để đọc số liệu tiếp theo

T_{rasc} là cho kỳ đọc mạch DRAM



Hình 3.9 DRAM 1Mbits

Addr sử dụng cho 1 Mbits : A0-A19, chia theo ROW và COLUM

3.2.3. Tổ chức các mạch DRAM thành các Modul

Trong máy vi tính các mạch DRAM được ghép thành các Modul nhớ như:

SIMM (Single In-line Memory Modul)

DIMM (Double In-line Memory Modul).

RIMM (RamBus In-line Memory Modul)

Các mạch DRAM được sử dụng là:

SDRAM (Synchronous DRAM)

DDR (Double Data Rate)

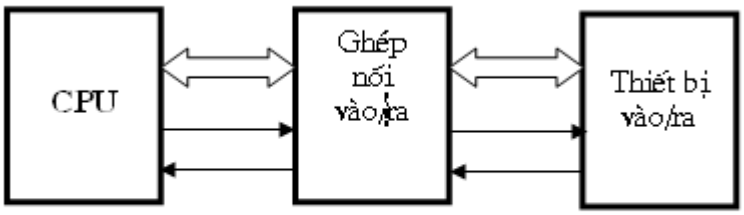
DDR2 (Double Data Rate 2).

CHƯƠNG IV

CÁC PHƯƠNG PHÁP VÀO/RA SỐ LIỆU

Hệ thống vào/ra *(hình 4.1)* gồm:

- Các thiết bị vào/ra
- Các bộ ghép nối vào/ra



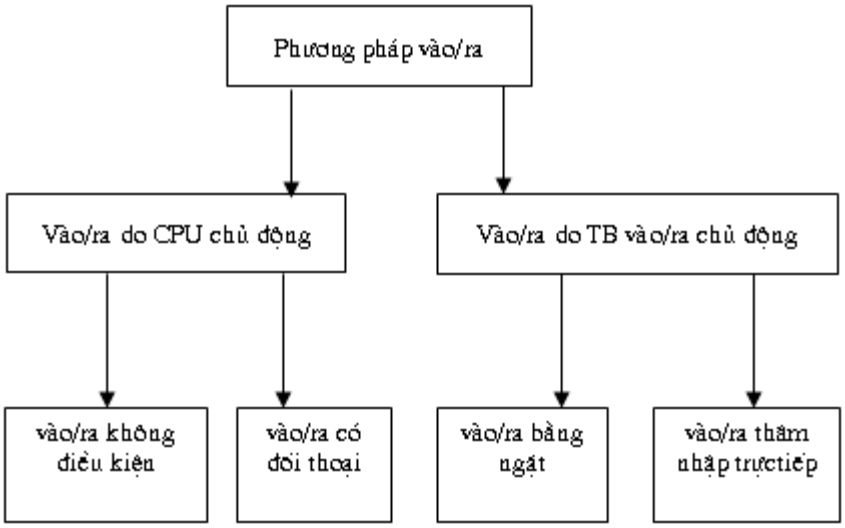
Hình 4.1: Sơ đồ hệ thống vào/ra

Trong bộ ghép nối vào/ra có các thanh ghi dữ liệu và các thanh ghi điều khiển và trạng thái. Các thanh ghi này còn gọi là cổng vào/ra. Các cổng này được địa chỉ bởi các địa chỉ vào/ra.

4.1. Phương pháp vào/ra số liệu do CPU chủ động

Vào/ra do CPU chủ động còn gọi là vào/ra điều khiển bằng chương trình được chia thành hai nhóm:

- Vào/ra số liệu bằng chương trình không điều kiện
- Vào/ra số liệu bằng chương trình có đối thoại.



Hình 4.2 các phương pháp Vào/Ra

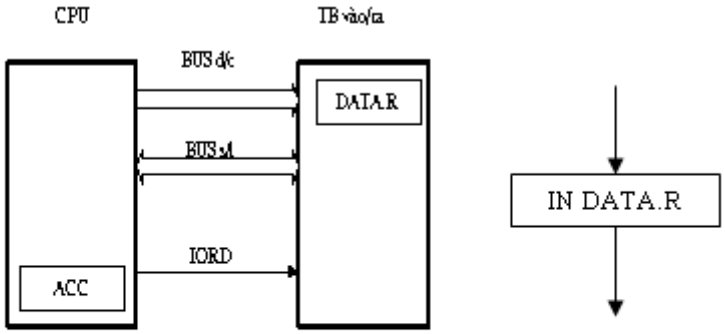
4.1.1. Vào/ra số liệu không điều kiện

Vào/ra số liệu không điều kiện có các đặc điểm như:

- CPU chuyển số liệu thông qua chương trình
- CPU giả thiết TB vào/ra luôn sẵn sàng chuyển số liệu
- Việc chuyển số liệu được thực hiện giữa các thanh ghi của CPU (ACC) và thanh ghi (cổng) của TB vào/ra

a.Vào số liệu *(hình 4.3)*

Trong thiết bị vào/ra có thanh ghi số liệu số liệu ký hiệu DATA.R.

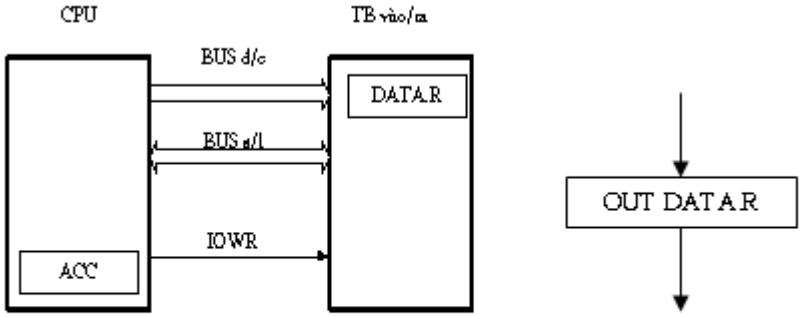


Hình 4.3a: Vào dữ liệu
Sơ đồ khối phần cứng

Các bước thực hiện:

- CPU đưa ra BUS đ/c địa chỉ của thanh ghi DATA.R
- CPU đưa ra BUS đ/k tín hiệu $\overline{\text{IORD}}$
- Số liệu từ DATA.R được đưa vào BUS số liệu và dưới tác động của tín hiệu $\overline{\text{IORD}}$ và được đưa vào ACC của CPU

b. Ra số liệu *(hình 4.4)*



Hình 4.4a: Ra dữ liệu
(Sơ đồ khối phần cứng)

Các bước thực hiện:

- CPU đưa địa chỉ của TB vào/ra ra BUS đ/c
- CPU đưa số liệu cần ghi ra BUS s/l
- CPU đưa tín hiệu $\overline{\text{IOWR}}$ ra BUS đ/k
- Dưới tác động của tín hiệu $\overline{\text{IOWR}}$ số liệu được ghi vào thanh ghi DATA.R của thiết bị ra.

Cơ chế này có hạn chế vì CPU ra lệnh đọc ($\overline{\text{IORD}}$) hay ghi ($\overline{\text{IOWR}}$) mà không kiểm tra xem thiết bị vào/ra có sẵn sàng gửi hay nhận số liệu hay chưa.

Trong hầu hết các máy tính việc chuyển số liệu được thể hiện qua lệnh vào ra riêng:

IN ⌊XXX⌋

địa chỉ thiết bị vào

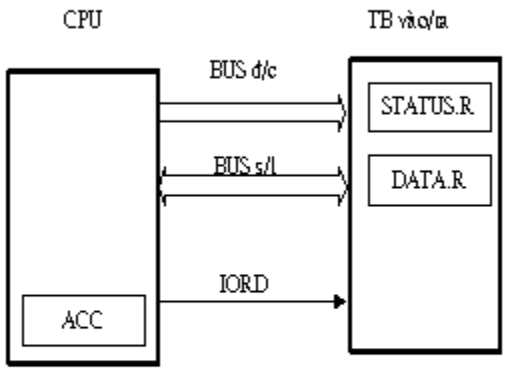
OUT ⌊XXX⌋

địa chỉ TB ra

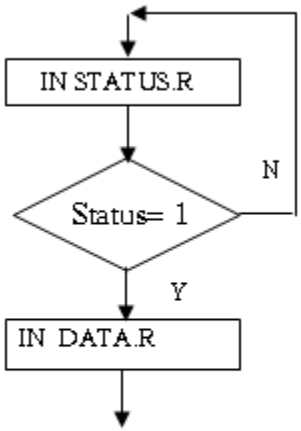
4.1.2. Vào/ra số liệu có đối thoại

Trong thiết bị vào/ra ngoài thanh ghi số liệu còn có thanh ghi trạng thái (Status.R) để thông báo TB vào/ra đã sẵn sàng chuyển số liệu chưa.

a. Vào số liệu có đối thoại *(hình 4.5)*



Hình 4.5a: Vào số liệu có đối thoại
(Sơ đồ khối phần cứng)

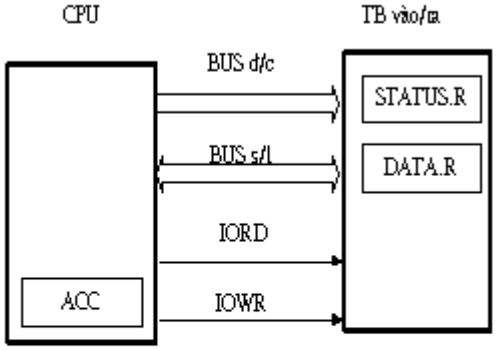


Hình 4.5b:Vào số liệu có đối thoại
(Lưu đồ phần mềm)

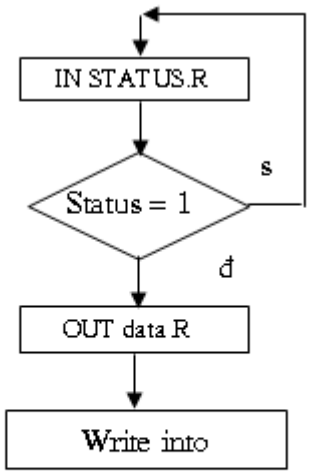
Các bước thực hiện:

- CPU đưa địa chỉ của thanh ghi trạng thái STATUS.R ra BUS đ/c
- CPU đưa tín hiệu đọc IORD ra BUS đ/k
- Nội dung của STATUS.R chuyển tới BUS s/l và được đưa vào ACC của CPU
- CPU phân tích xem thiết bị vào/ra đã sẵn sàng chưa. Nếu sẵn sàng thì CPU ra lệnh đọc số liệu, nếu chưa thì quay về chờ.

b. Ra số liệu có đối thoại *(hình 4.6)*



Hình 4.6a: Ra số liệu có đối thoại. (Sơ đồ khối phần cứng)



Hình 4.6b: Ra số liệu có đối thoại (Lưu đồ phần mềm)

Các bước thực hiện:

- CPU đưa địa chỉ của thanh ghi STATUS.R ra BUS đ/c
- CPU đưa $\overline{\text{IORD}}$ ra BUS đ/k
- Nội dung của thanh ghi STATUS.R đưa vào BUS s/l và được đưa vào ACC của CPU
- CPU phân tích nội dung của ACC, có hai trường hợp xảy ra:

. Nếu TB vào/ra sẵn sàng , CPU đưa số liệu ra TB vào/ra

. Nếu TB vào/ra chưa sẵn sàng CPU quay về đọc thanh ghi trạng thái.

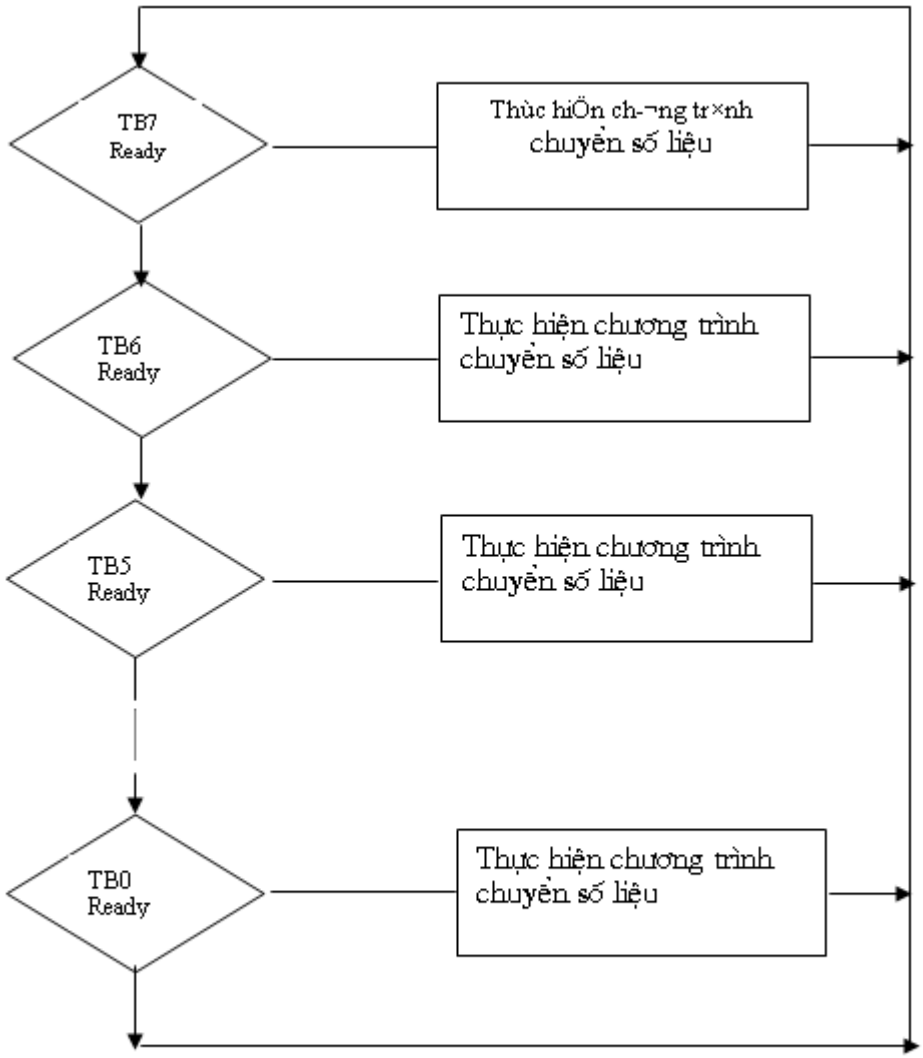
Trong phương pháp vào/ra có đối thoại, nếu thiết bị vào/ra chưa sẵn sàng nhận hay gửi số liệu thì CPU phải chờ. Nói các khác phương pháp này gây lãng phí thời gian của CPU. Để giảm bớt thời gian chờ, CPU có thể dùng phương pháp hỏi vòng các TB vào/ra

c. Vào/ra số liệu có đối thoại bằng phương pháp hỏi vòng *(hình 4.7)*

Ví dụ: CPU quản lý 8 TB vào/ra TB0...TB7

Phương pháp này có ưu điểm là giảm bớt thời gian chờ của CPU nhưng việc đáp ứng yêu cầu chuyển số liệu của các thiết bị là không đều nhau, không có

khả năng đáp ứng yêu cầu tức thời của thiết bị. Nhược điểm này có thể được khắc phục bằng phương pháp vào/ra do thiết bị vào/ra chủ động.



Hình 4.7: Vào/ra số liệu có đối thoại bằng phương pháp hỏi vòng

4.2. Phương pháp vào/ra số liệu do thiết bị vào/ra chủ động

4.2.1. Nguyên lý vào/ra bằng ngắt (hình 4.8 và hình 4.9)

Những hạn chế của phương pháp vào/ra bằng chương trình được khắc phục bằng phương pháp vào/ra bằng ngắt. Trong phương pháp này TB vào/ra chủ động khởi động quá trình vào/ra số liệu.

Quá trình thực hiện ngắt:

- CPU hoạt động bình thường

- Khi TB vào/ra sẵn sàng chuyển số liệu sẽ gửi yêu ngắt tới CPU bằng tín hiệu $\overline{\text{INTR}}$ (Interrupt Request)

- CPU thực hiện nốt lệnh đang thực hiện trước khi trả lời

- Xác định ngắt và trả lời thiết bị vào/ra bằng tín hiệu $\overline{\text{INTA}}$ (Interrupt Acknowledgement)

- Đẩy PSW (Program State Word) và PC (Program Counter) vào ngăn xếp

TB vào/ra thông qua bộ điều khiển ngắt cho biết địa chỉ của chương trình con phục vụ ngắt. CPU nạp địa chỉ này vào PC.

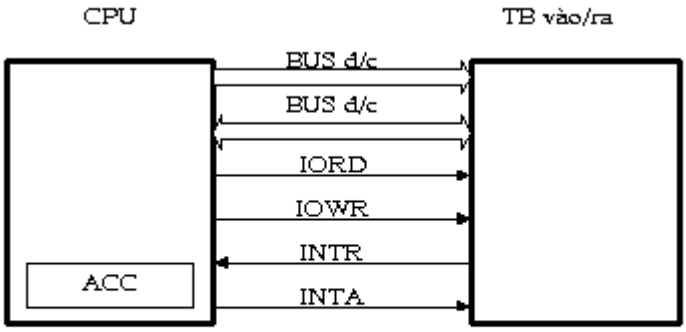
- CPU nhảy đến chương trình con phục vụ ngắt (Interrupt Service Routine) ISR

- Chương trình ISR sẽ đẩy các thanh sẽ bị thay đổi trong chương trình con vào ngăn xếp.

- Chương trình ISR sẽ thực hiện việc chuyển số liệu giữa TB vào/ra và bộ nhớ qua CPU (ACC).

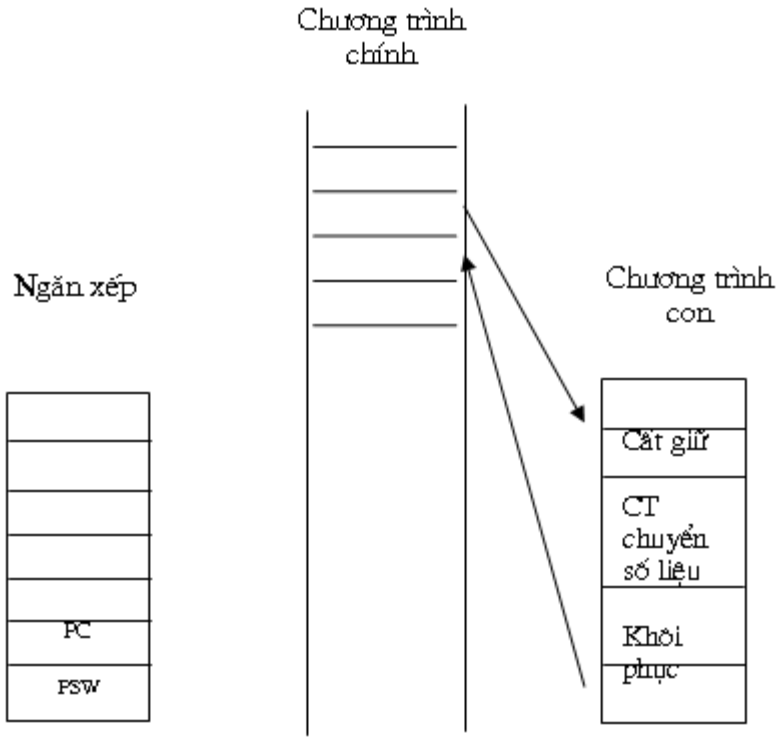
- Sau khi chuyển số liệu xong, CPU khôi phục các thanh ghi

- Khôi phục PC và PSW từ ngăn xếp, trở về chương trình chính thực hiện tiếp nhiệm vụ trước khi có ngắt.



Hình 4.8: Sơ đồ khối phần cứng

(Vẽ kiểu khác với Interrupt Controler I-8259 !)



Hình 4.9: Lưu đồ phần mềm

Bằng phương pháp ngắt thiết bị vào ra được phục vụ theo yêu cầu của chúng nên thường sử dụng trong các ứng dụng thời gian thực đòi hỏi thời gian khởi động phục vụ ngắn.

Ví dụ:

Hệ thống báo động (từ đầu cảm biến điều khiển lò hơi khi nhiệt độ vượt ngưỡng)

Đồng hồ thời gian thực

Báo lỗi phần cứng

Báo mất nguồn nuôi

Báo lỗi trong truyền tin

Một số khái niệm khác

- Ưu tiên: Trong cùng một thời điểm có thể có nhiều yêu cầu ngắt nên máy tính (CPU) phải có phương pháp để lập thứ tự phục vụ ngắt đảm bảo cho việc vào/ra số liệu của toàn hệ không bị xáo trộn. Thông thường mỗi thiệt bị vào/ra được

gắn một mức ưu tiên. Tuỳ theo mức ưu tiên mà CPU xác định thứ tự phục vụ cho các thiết bị vào/ra.

- NMI (Non Maskable Interrupt) là yêu cầu ngắt tức thời, không cấm và cho phép được bằng phần mềm được.

- Ngắt mềm dùng lệnh (INT) để gọi các chương trình phục vụ ngắt của hệ thống.

4.2.2. Nguyên lý thâm nhập bộ nhớ trực tiếp (DMA: Direct Memory Access)

Trong các phương pháp vào/ra trình bày trên có các nhược điểm sau:

- Vào/ra bằng chương trình phải chuyển số liệu giữa thiết bị vào/ra và bộ nhớ thông qua ACC *(hình 4.10)*

Sử dụng phương pháp vào/ra điều khiển bằng chương trình ta thấy:

- Việc vào/ra số liệu đều qua ACC của CPU
- Muốn đọc số liệu từ TB vào/ra vào bộ nhớ phải qua 2 bước

  (DATA.R) $\rightarrow$ ACC

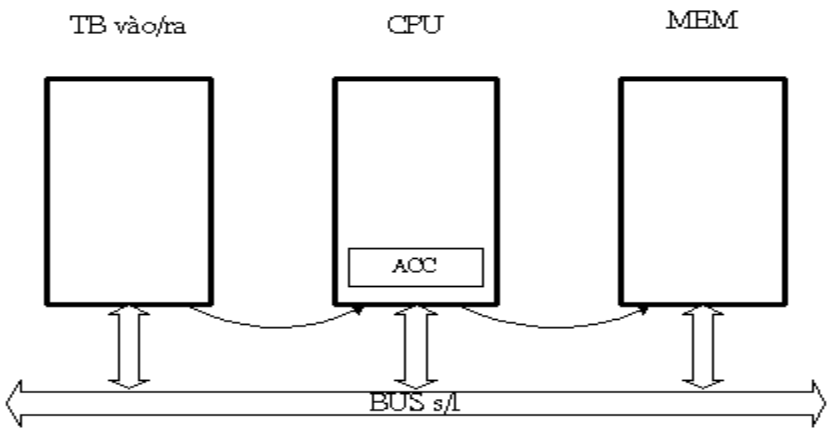
  (ACC) $\rightarrow$ MEM
- Muốn đưa nội dung của ô nhớ tới TB vào/ra cũng phải qua 2 bước

  (MEM) $\rightarrow$ ACC

  (ACC) $\rightarrow$ DATA.R của TB vào/ra

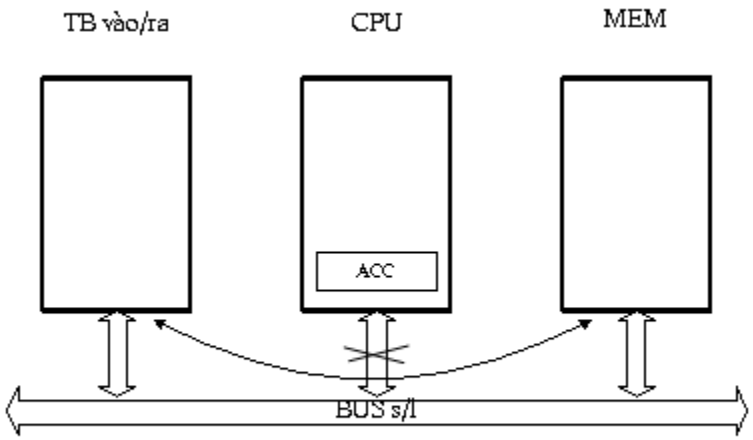
Như vậy việc chuyển số liệu giữa thiết bị ngoại vi và bộ nhớ cần hai bước, tốc độ chậm.

Phương pháp vào/ra bằng ngắt bảo đảm thiết bị vào/ra được phục vụ gần như tức thời (trong thời gian ngắn). Nhưng trong chương trình con phục vụ ngắt, quá trình chuyển số liệu được thực hiện bằng chương trình nên tốc độ chậm.



Hình 4.10: Nguyên lý vào/ra bằng chương trình chuyển số liệu

Phương pháp vào/ra thâm nhập bộ nhớ trực tiếp DMA (Direct Memory Access) *(hình 4.11)* khắc phục các nhược điểm trên. Phương pháp này chuyển số liệu với bộ nhớ không qua ACC của CPU



Hình 4.11: Vào/ra không qua ACC

Chương trình con để chuyển số liệu giữa TB vào/ra và bộ nhớ được thay đổi bằng đơn vị điều khiển đặc biệt bằng phần cứng DMAC (Direct Memory Access Controller).

Quá trình thực hiện DMA:

CPU làm việc bình thường

Khi thiết bị ngoại vi muốn chuyển số liệu trực tiếp với bộ nhớ thì gửi yêu cầu tới DMAC qua tín hiệu DRQ (DMA Reqest)

Bộ điều khiển DMAC chuyển yêu cầu này tới CPU qua tín hiệu HOLD

CPU thực hiện nốt chu kỳ máy đang thực hiện, treo BUS và trả lời DMAC bằng tín hiệu HLDA.

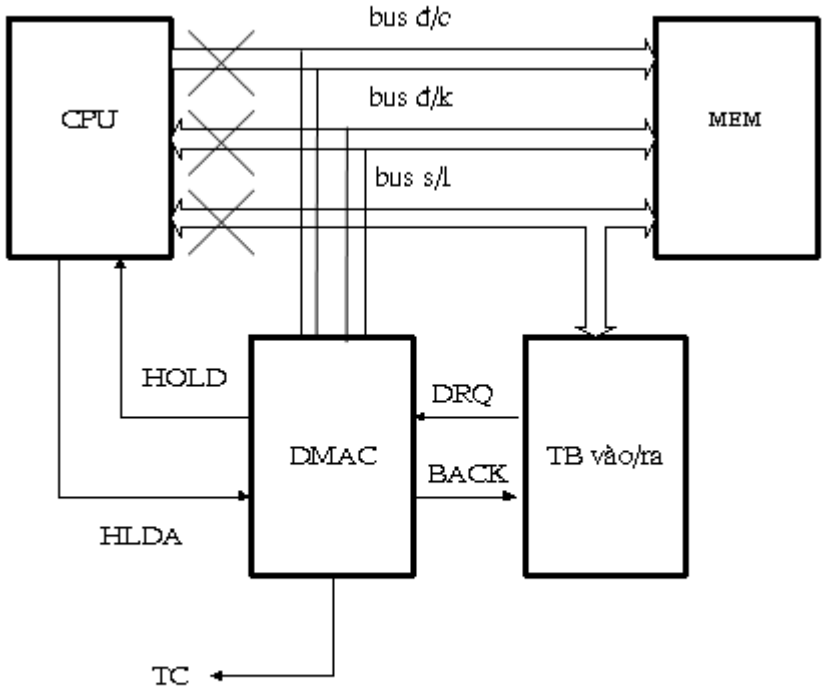
DMAC trả lời thiết bị vào/ra bằng tín hiệu DACK, làm chủ BUS sinh ra

- BUS đ/c (Các tín hiệu địa chỉ)
- Các tín hiệu điều khiển
- Điều khiển chuyển số liệu giữa bộ nhớ và TB vào/ra

Số liệu chuyển giữa bộ nhớ và thiết bị vào/ra thường là cả một khối.

Khi chuyển xong số liệu DMAC đưa tín hiệu TC (Terminal Count) thành tích cực để báo một quá trình DMA kết thúc

Có thể minh họa hoạt động của phương pháp vào/ra số liệu bằng DMA như *hình 4.12*.



Hình 4.12: Vào/ra số liệu bằng DMA

4.3. Địa chỉ thiết bị vào/ra

Bộ nhớ là tập hợp các thanh ghi, quá trình chuyển số liệu giữa CPU và bộ nhớ thực chất là quá trình chuyển số liệu giữa các thanh ghi ACC của CPU và các ô nhớ.

Trong các thiết bị vào/ra đều có thanh ghi số liệu, việc chuyển số liệu giữa CPU và TB vào/ra thực chất cũng là chuyển số liệu giữa các thanh ghi.

Trong nhiều hệ máy tính có khả năng có

+ địa chỉ riêng cho TB vào/ra và

+ địa chỉ riêng cho bộ nhớ

Các hệ thống như vậy gọi là hệ thống có vào/ra tách biệt.

Các hệ thống này thường có lệnh vào/ ra riêng.

IN XXX

địa chỉ thiết bị vào

OUT XXX

địa chỉ TB ra

Trong một số trường hợp (đặc biệt là các hệ vi xử lý họ MOTOROLA) không có không gian địa chỉ riêng cho TB vào/ra mà phải dùng không gian địa chỉ của bộ nhớ thay cho không gian địa chỉ TB vào/ra.

Trong các hệ thống này một số ô nhớ đặc biệt không dùng như ô nhớ mà dùng như các thanh ghi cho TB vào/ra. Các ô nhớ này còn gọi là các ô nhớ giả (Pseudo). Quá trình chuyển số liệu giữa CPU và TB vào/ra được thực hiện nhờ các lệnh chuyển số liệu giữa CPU và bộ nhớ.

CHƯƠNG V

CẤU TRÚC CỦA ĐƠN VỊ XỬ LÝ TRUNG TÂM

5.1. Họ Vi xử lý Intel 80x86

Với sự tiến bộ của công nghệ vi điện tử, toàn bộ CPU được cấy bên trong một mạch vi điện tử. Mạch vi điện tử như vậy còn gọi là Microprocessor (MPU), (MP). Họ vi xử lý Intel 80x86 được sử dụng để xây dựng dòng máy tính cá nhân IBM-PC (Personal Computer), được sử dụng rộng rãi nhất hiện nay. Bảng sau đây sẽ giới thiệu quá trình phát triển và tính năng cơ bản của họ vi xử lý Intel 80x86.

PC/AT Compatibility Standard | * Protected Mode | **Quad pumped 100 | #Integrated L2

CPU	8088	80286	80386	80486	Pentium®	Pentium® Pro	Pentium® II	Pentium® 4
Addr-BUS	20	24	32	32	32	36	36	36
VIRTUAL SIZE	1 MEG	16 MEG*	4 GIG*	4 GIG*	4 GIG*	64 GIG*	64 GIG*	64 GIG*
MEM SIZE	NA	1 GIG	64 TERA	64 TERA	64 TERA	64 TERA	64 TERA	64 TERA
D-BUS	8	16	32	32	64	64	64	64
REG SIZE	16	16	32	32	32	32	32	32
#SEG.REGS	4	4	6	6	6	6	6	6
MATH	8087	80287	80387	On Chip	On Chip	On Chip	On Chip	On Chip
BUS SPEED (MHz)	4.77,8	8,12	16,25, 33	33, 50	50, 60, 66	60, 66	66, 100	100/400*
PAGING	NO	NO	YES	YES	YES	YES	YES	YES
ON CHIP CACHE	NO	NO	NO	YES	YES	#YES	#YES	#YES

5.2. Giới thiệu về Bộ vi xử lý 80x86

Chúng ta bắt đầu bằng việc tìm hiểu tính năng và kiến trúc bộ vi xử lý 16-bit 80286 để dễ dàng tiến tới làm chủ các bộ vi xử lý 32-bit phức tạp hơn như Pentium.

5.2.1. Tính năng cơ bản của bộ vi xử lý 80286

Bộ vi xử lý 80286 là một bộ vi xử lý thuộc họ vi xử lý 80x86. Họ 80x86 gồm các bộ vi xử lý 8086, 8088, 80186 80286 và 80386, 80486 và Pentium.

Một số đặc tính kỹ thuật chính của MP 80286 là:

Tần số của đồng hồ thời gian: 10MHz, 8MHz, 16MHz, 20MHz.

Khả năng địa chỉ hóa: 16 mega byte (MB) nhớ vật lý, 1 giga byte (GB) nhớ ảo.

Hai chế độ hoạt động: Chế độ địa chỉ thực, chế độ địa chỉ ảo hay chế độ bảo vệ.

Có đơn vị quản lý bộ nhớ bên trong bộ vi xử lý.

Có bốn mức bảo vệ bộ nhớ.

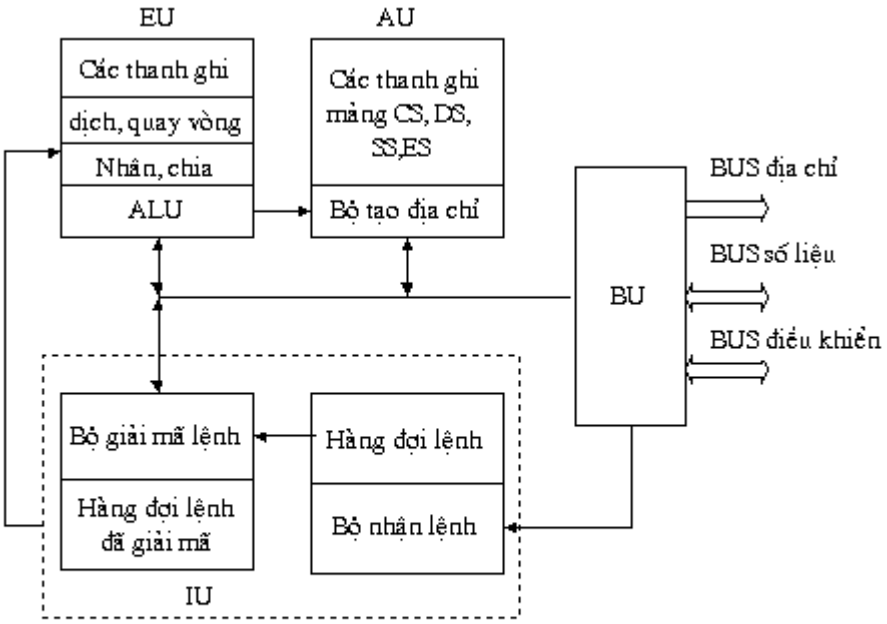
Có khả năng đối thoại với các bộ đồng xử lý.

Trong chế độ địa chỉ thực, MP 80286 có mã lệnh tương thích với MP 8086 và 8088, nhưng quá trình thực hiện nhanh hơn.

Khả năng địa chỉ hóa bộ nhớ vật lý lớn nhất là 16 MB vì có 24 bit địa chỉ. Trong chế độ thực, chỉ sử dụng các đường địa chỉ A0-A19 và như vậy địa chỉ hóa được một mega byte.

5.2.2. Sơ đồ khối chức năng của bộ vi xử lý (Hình 5.1)

Bộ vi xử lý MP được xây dựng từ 4 đơn vị có thể làm việc song song.



Hình 5.1: Sơ đồ chức năng của MP80286

a. Đơn vị bus BU (Bus Unit)

BU thực hiện các chức năng chính sau:

Sinh ra các tín hiệu địa chỉ, số liệu và điều khiển để thâm nhập vào bộ nhớ hay thiết bị vào/ra.

Thiết lập quan hệ với bộ đồng xử lý EP hay các bộ xử lý khác đang làm chủ BUS.

Cho phép quá trình nhập lệnh song song với các quá trình khác vì có hàng đợi lệnh 6 byte. Vì vậy loại trừ được thời gian chết trong khi tìm lệnh trong bộ nhớ.

b. Đơn vị lệnh IU (Instruction Unit)

IU thực hiện các chức năng chính sau:

Nhận lệnh từ bộ nhớ vào hàng đợi lệnh.

Giải mã lệnh thành con trỏ tới vi chương trình thực hiện lệnh.

Quá trình đọc lệnh được thực hiện khi không có đơn vị khác yêu cầu sử dụng BUS.

Quá trình nhận lệnh và giải mã lệnh được thực hiện song song với quá trình thực hiện các lệnh trước đó.

c. Đơn vị thực hiện lệnh EU (Execution Unit)

EU thực hiện các phép tính sau:

Cộng, trừ (8,16 bit).

Nhân, chia bằng phần cứng.

Các phép tính xử lý bit, dịch chuyển, quay vòng ở các thanh ghi và ô nhớ.

Về phương kiến trúc, tập thanh ghi đa năng đã tương đối phong phú.

d. Đơn vị địa chỉ AU (Address Unit)

Bao gồm các thanh ghi mảng (CS. DS, SS, ES) và bộ chuyển đổi địa chỉ.

Trong họ vi xử lý 80x86, bộ nhớ được tổ chức theo mảng. Địa chỉ ô nhớ gồm hai thành phần: địa chỉ mảng và địa chỉ offset. Địa chỉ mảng cho biết mảng nhớ nào trong bộ nhớ, địa chỉ offset cho biết ô nhớ cụ thể trong bộ nhớ. Cặp địa chỉ mảng: địa chỉ offset còn gọi là địa chỉ logic. Từ địa chỉ logic sẽ được tính thành địa chỉ vật lý để truy nhập bộ nhớ vật lý.

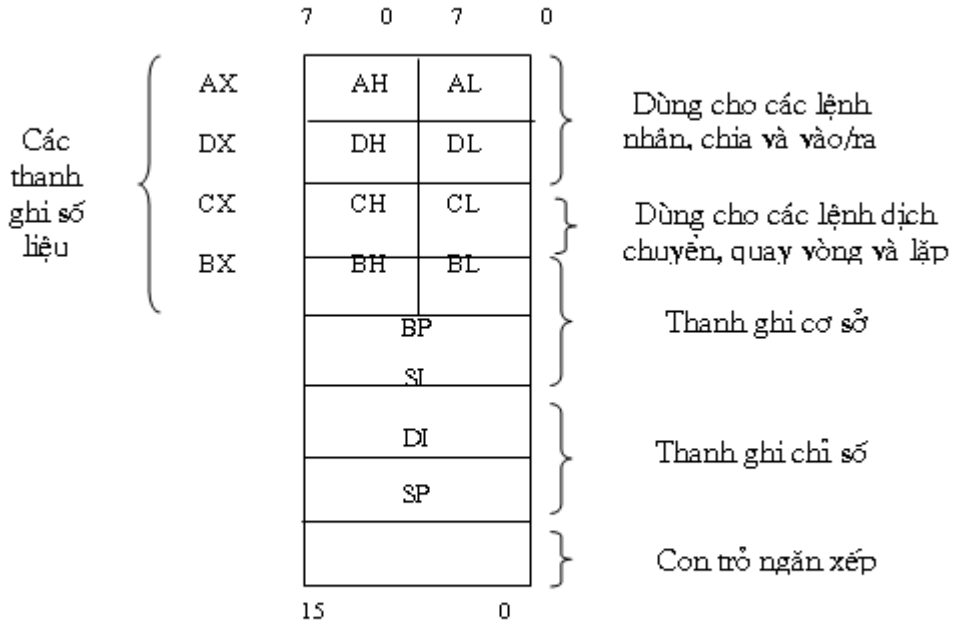
5.3. Các thanh ghi của 80x86

MP 80286 gồm 15 thanh ghi 16 bit, chia làm 3 nhóm:

Thanh ghi đa năng *(hình 5.2a)*

Các thanh ghi này chứa các toán hạng của các phép tính số học hay logic. Bốn thanh ghi 16 bit AX, BX, CX, DX có thể sử dụng như tám thanh ghi 8 bit. Bốn thanh ghi chứa byte thấp (AL, BL, CL, DL) và bốn thanh ghi chứa byte cao (AH, BH, CH, DH) .

Ngoài chức năng chung, các thanh ghi trên còn có các chức năng ngầm định:
- AX giữ vai trò như thanh ghi ACC của các máy tính tổng quát.
- AX và DX dùng trong các lệnh nhân, chia và vào/ra.
- CX giữ chức năng thanh đếm, dùng cho tất cả các lệnh quay vòng, lệnh dịch chuyển và các lệnh lặp.
- BX và BP là những thanh ghi cơ sở, chứa địa chỉ cơ sở trong phép tính địa offset.
- SI và DI gọi là những thanh ghi chỉ số, chứa địa chỉ ofsset có thể tăng dần khi thâm nhập vào một cấu trúc dữ liệu.
- SP là con trỏ ngăn xếp, chứa địa chỉ offset của đỉnh ngăn xếp. BP còn gọi con trỏ cơ sở, được dùng ngầm định để truy nhập vào ngăn xếp. BP cũng tham gia để tính địa chỉ offset trong các phép tính địa chỉ trong lệnh.



Hình 5.2a: Các thanh ghi đa năng

Các thanh ghi mảng (segment)*(hình 5.2b)*

Các chương trình thường được cấu tạo từ nhiều modul lệnh và số liệu. MP 80286 cho phép một chương trình đang thực hiện có thể thâm nhập tức thời đến bốn mảng nhớ. Bốn thanh ghi mảng nhận biết bốn mảng nhớ đang sử dụng. Các thanh ghi mảng đó là:

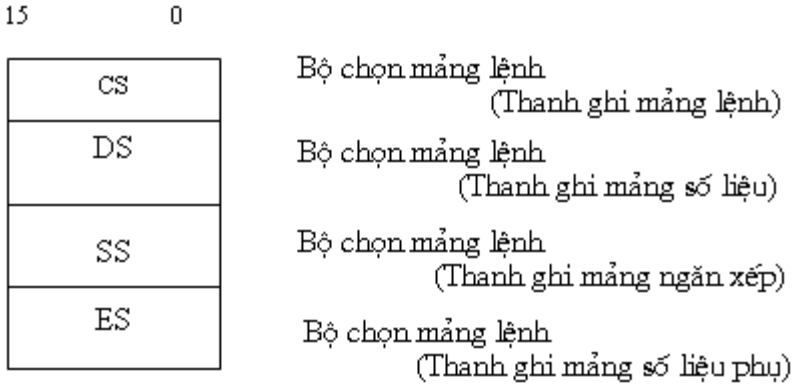
CS: thanh ghi mảng lệnh

DS: thanh ghi mảng số liệu

SS thanh ghi mảng ngăn xếp

ES thanh ghi mảng số liệu phụ.

Nội dung của thanh ghi mảng gọi là bộ chọn mảng nên có nhiều khi còn gọi thanh ghi mảng là thanh ghi chọn mảng.



Hình 5.2b: Các thanh ghi mảng

c. Các thanh ghi điều khiển và trạng thái

Các thanh ghi này gồm:

Thanh ghi cờ FR (Flag Register)

Con trỏ lệnh IP (Instruction Pointer)

Thanh ghi trạng thái máy MSW (Machine Status Word)

Con trỏ lệnh IP

Con trỏ lệnh luôn chứa địa chỉ tương đối (so với đầu của mảng nhớ đang sử dụng) của lệnh sẽ thực hiện tiếp theo. Như vậy con trỏ lệnh cùng với thanh ghi mảng lệnh (CS: IP) định nghĩa địa chỉ logic của lệnh tiếp theo, có chức năng như thanh đếm chương trình (PC) trong máy tính tổng quát.

Con trỏ lệnh được điều khiển bởi cơ chế ngắt, bẫy và các phép chuyển điều khiển.

Thanh ghi cờ FR

Thanh ghi cờ FR có 16 bit được biểu diễn trên *hình 5.3*:

CF (Carry Flag), cờ nhớ.

CF=1 chỉ rằng có hiện tượng nhớ từ bit cao nhất của một phép toán 8 bit hay 16 bit, sau khi thực hiện các lệnh số học. Tất cả các lệnh dịch chuyển và quay vòng đều ảnh hưởng đến cờ CF.

PF (Parity Flag), cờ kiểm tra chẵn lẻ.

PF=1, nếu 8 bit thấp của kết quả có số các chữ số 1 là chẵn, ngược lại PF=0.

AF (auxiliary carry Flag), cờ nhớ phụ.

AF được sử dụng trong các phép tính số học với mã BCD (Binary Coded Decimal). Nó cho biết có nhớ từ số BCD thấp sang số BCD cao hay bit số liệu D3 sang bit D4 trong thanh ghi AL.

ZF (Zero Flag), cờ 0

ZF=1, khi kết quả phép tính bằng 0.

SF (Sign Flag), cờ dấu.

SF cho biết kết quả phép tính là dương hay âm, nếu âm, SF=1 ngược lại, SF=0

OF (Overflow Flag), cờ tràn.

OF dùng trong phép tính số học có dấu, nó chỉ rằng kết quả tràn sang bit dấu.

- TF (Trap Flag), cờ TF chỉ chế độ chạy từng lệnh. Nếu cờ TF=1, MP 80286 ở chế độ chạy từng bước để giúp cho việc hiệu chỉnh chương trình được thuận lợi.

Mỗi lệnh gây ra một ngắt để MP 80286 nhảy vào chương trình con phục vụ hiệu chỉnh chương trình.

Cờ TF chỉ có thể thay đổi thông qua ngăn xếp: cất thanh ghi cờ RF vào ngăn xếp, thiết lập bit TF trong ngăn xếp và đưa trở lại thanh ghi cờ.

Trước khi nhảy vào chương trình con phục vụ hiệu chỉnh, thanh ghi cờ FR được tự động lưu giữ. Trong chương trình con này bit TF được xoá, để khi thực hiện các lệnh của chương trình con không xẩy ra chế độ chạy từng bước. Khi trở về từ chương trình con phục vụ hiệu chỉnh, thanh ghi cờ được phục hồi và bit TF vẫn giữ giá trị cũ. (TF=1).

- IF (Interrupt enable Flag), cờ cho phép ngắt. Nếu cờ IF=1 cho phép ghi nhận ngắt ở chân INTR của bộ vi xử lý.

Bit IF được thiết lập bằng lệnh STI và được xóa bằng lệnh CLI hoặc thông qua ngăn xếp.

- DF (Direction Flag), bit hướng.

DF cho biết hướng tiến triển trong một chuỗi số liệu.

Nếu DF=1, nội dung của thanh ghi SI hoặc DI hay cả hai thanh SI và DI sẽ tự động giảm đi 1. Nếu DF=0, nội dung của các thanh ghi này sẽ tự động tăng 1 trong các lệnh xử lý chuỗi.

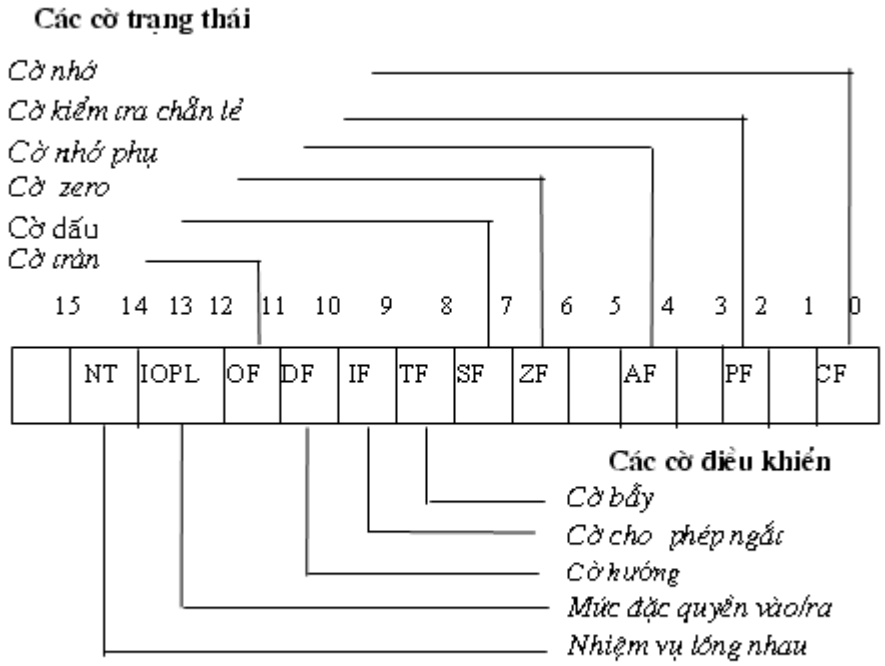
Bit DF có thể thiết lập bằng lệnh STD, xóa bằng lệnh CLD hay có thể điều khiển thông qua ngăn xếp.

NT (Nested Task), nhiệm vụ lồng nhau.

NT cho biết lệnh đang thực hiện đang tiến triển bên trong cùng một nhiệm vụ hay sẽ gây ra việc chuyển nhiệm vụ. NT chỉ dùng trong chế độ bảo vệ của MP 80286.

IOPL (I/O Privilege Level), mức đặc quyền vào/ra

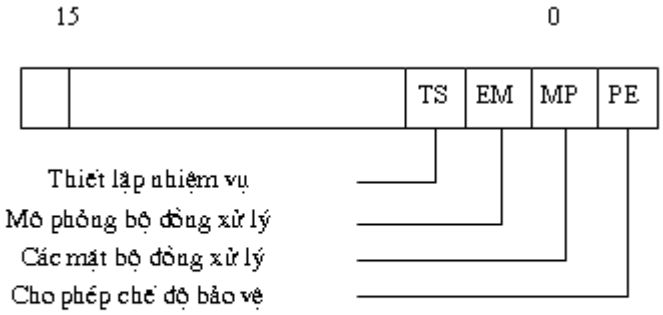
IOPL chỉ ra mức đặc quyền thấp nhất mà nhiệm vụ đang thực hiện cần có, để được phép thực hiện các lệnh vào/ra. Cũng như NT, IOPL chỉ có ở chế độ bảo vệ.



Hình 5.3: Thanh ghi cờ

Thanh ghi trạng thái máy MSW

Thanh ghi MSW có 16 bit nhưng chỉ có 4 bit được dùng, còn các bit khác dự trữ cho các bộ vi xử lý 32 bit *(hình 5.4)*.



Hình 5.4: Thanh ghi MSW

- PE (Protected mode Enable), cho phép bộ vi xử lý chuyển sang chế độ bảo vệ.

Bit PE=1 chỉ rằng MP 80286 đang làm việc ở chế độ bảo vệ. Một khi đã được thiết lập thì chỉ có RESET mới xóa PE về 0 được.

- MP (Monitor Processor Extension).

Bit MP cho biết đang có một bộ đồng xử lý cùng làm việc.

- EM (Emulate Processor Extension).

EM sẽ cho phép mô phỏng một bộ đồng xử lý.

- TS (Task Switch)

Bit này sẽ được thiết lập mỗi khi chuyển nhiệm vụ và thường được dùng trong trường hợp có bộ đồng xử lý. Khi chuyển nhiệm vụ không thay đổi phạm vi (context) của bộ đồng xử lý. Như vậy, khi MP 80286 thực hiện lệnh đồng xử lý đầu tiên (sau chuyển nhiệm vụ), sẽ gây ra ngoại lệ số 7. Xử lý ngoại lệ này cho biết có cần thay đổi phạm vi của bộ đồng xử lý hay không?

5.4. Các chế độ địa chỉ hóa

Địa chỉ trực tiếp

Địa chỉ offset của toán hạng chứa trong 16 bit của lệnh.

OFFSET=16 bit dịch chuyển của lệnh

Ví dụ: MOV AX, [34FFh]

Địa chỉ gián tiếp qua thanh ghi

Địa chỉ offset của toán hạng chứa trong các thanh ghi SI, DI, BX

OFFSET = (SI , DI, BX)

Ví dụ: MOV AX, [SI]

Địa chỉ tương đối theo thanh ghi cơ sở BX, BP

Địa chỉ offset của toán hạng bằng tổng của dịch chuyển chứa trong lệnh và nội dung của các thanh ghi cơ sở BX và BP

OFFSET = (BX hay BP) + dịch chuyển

Ví dụ: MOV AX, [BX] + 3FFh

Có thể sử dụng chế độ địa chỉ này để xây dựng mảng một chiều.

Địa chỉ tương đối theo chỉ số SI, DI

Địa chỉ offset của toán hạng bằng tổng của dịch chuyển chứa trong lệnh và nội dung của thanh ghi SI hay DI

OFFSET= (SI hay DI) + dịch chuyển

Ví dụ: MOV AX, [SI] + 154Eh

Có thể sử dụng chế độ địa chỉ này để xây dựng mảng một chiều.

Địa chỉ tương đối có qua thanh ghi cơ sở và chỉ số

Địa chỉ offset của toán hạng bằng tổng của nội dung thanh ghi cơ sở và nội dung thanh ghi chỉ số.

OFFSET= (BX hay BP) + (SI hay DI)

Ví dụ: MOV AX, [BX] + [SI]

Có thể sử dụng chế độ địa chỉ này để xây dựng mảng hai chiều.

Khi ta có một vùng số liệu động và muốn làm việc với các phần tử của vùng thì chế độ địa chỉ hóa này là thích hợp. Thanh ghi cơ sở phục vụ chọn vùng, thanh ghi chỉ số trỏ đến bên trong của vùng xác định.

Địa chỉ tương đối, có thanh ghi cơ sở, thanh ghi chỉ số và dịch chuyển

Địa chỉ offset của toán hạng là tổng nội dung của thanh ghi cơ sở, nội dung của thanh ghi chỉ số và dịch chuyển chứa trong lệnh.

OFFSET = (BX hay BP)+ (SI hay DI) + dịch chuyển

Ví dụ: MOV AX, [BX] + [SI] + 154Eh

Có thể sử dụng chế độ địa chỉ này để xây dựng mảng hai chiều.

Nhờ có thanh ghi cơ sở, chế độ địa chỉ này cho phép lặp lại một cấu trúc mà ở trong đó có các vùng số liệu cần tìm. Chỉ số cho phép chọn phân tử xác định bên trong vùng dữ liệu này.

g. Số liệu tức thời trong lệnh

Ví dụ: MOV AX, 3000h

5.5. Cấu trúc lệnh cơ bản của 80x86

Cấu trúc lệnh cơ bản của MP80286 được mô tả trong *hình 5.5*.

Lệnh của MP 80286 có độ dài từ 1 đến 6 byte.

1. OP – 6 bit cao của byte thứ nhất là mã lệnh.

2. D là hướng chuyển số liệu.

D = 1 Có mã ở phần REG là thanh ghi đích

D = 0 Có mã ở phần REG là thanh ghi nguồn

3. W = 0 chỉ lệnh làm việc với toán hạng 8 bit

W = 1 chỉ lệnh làm việc với toán hạng 16 bit

4. MOD cho biết 1 toán hạng là ở ô nhớ hay ở thanh ghi.

Nếu toán hạng ở trong bộ nhớ thì có 3 trường hợp xảy ra

không có dịch chuyển

có 8 bit dịch chuyển

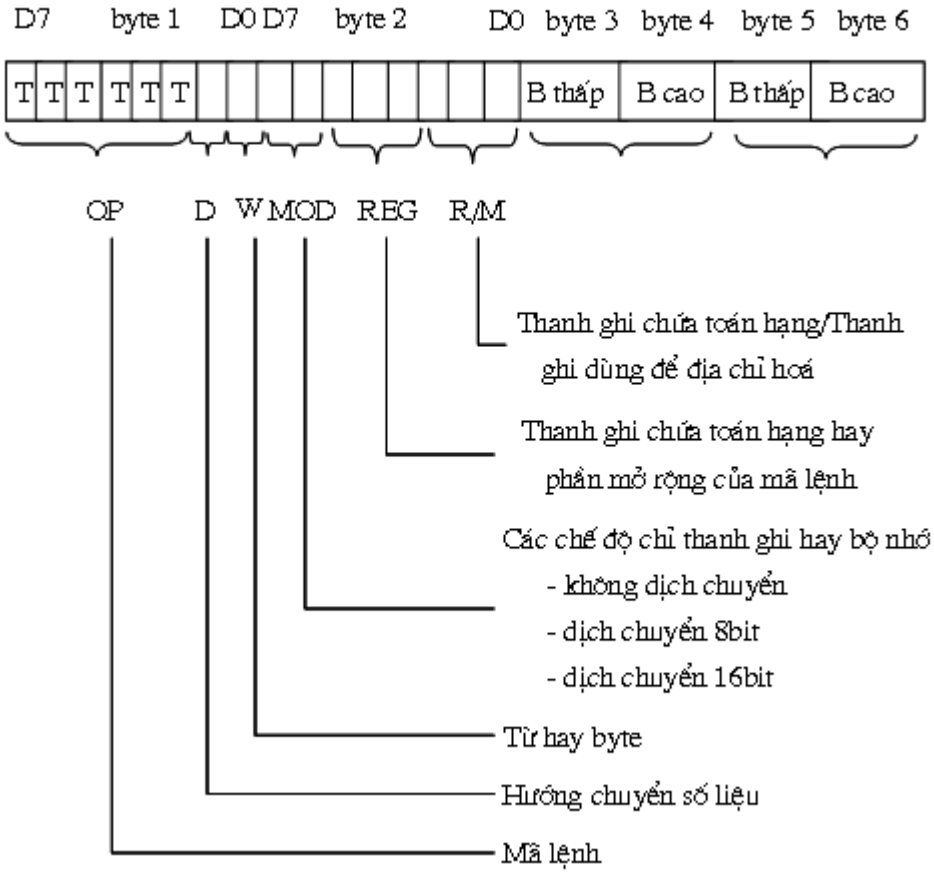
có 16 bit dịch chuyển

toán hạng ở trong thanh ghi

5. REG - Định nghĩa các thanh ghi:

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI

111	BH	DI



Hình 5.5: Cấu trúc lệnh của MP 80286

6. Nếu MOD = 11 thì 3 bit R/M có ý nghĩa sau

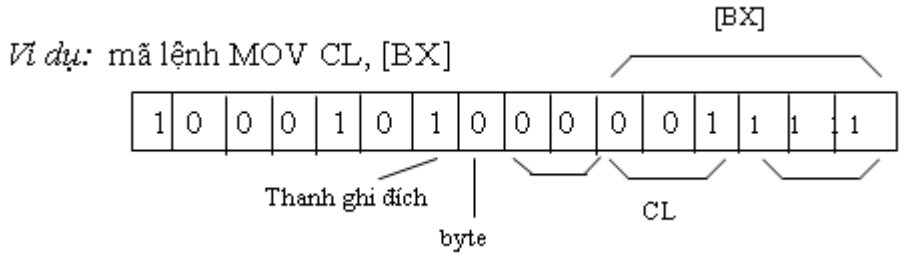
R/M	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Nếu MOD khác 11 thì 3 bit R/M có ý nghĩa sau:

R/M	MOD = 00	MOD = 01	MOD = 10
000	(BX) + (SI)	(BX) + (SI) + disp8	(BX) + (SI) + disp16
001	(BX) + (DI)	(BX) + (DI) + disp8	(BX) + (DI) + disp16
010	(BP) + (SI)	(BP) + (SI) + disp8	(BP) + (SI) + disp16
011	(BP) +(DI)	(BP) +(DI) + disp8	(BP) + (DI) + disp16
100	(SI)	(SI) + disp8	(SI) + disp16
101	(DI)	(DI) + disp8	(DI) + disp16
110	địa chỉ trực tiếp	(BP) + disp8	(BP) + disp16
111	BX	(BX) + disp8	(BX) + disp16

Bảng sau cho biết thanh ghi mảng ngầm định hay có thể viết tường minh trong lệnh cùng với địa chỉ offset tương ứng:

Các thao tác quy chiếu bộ nhớ	Thanh ghi mảng ngầm định	Thanh ghi mảng thay thế	Địa chỉ offset
1. Nhận lệnh	CS	N	IP
2. Thao tác ngăn xếp	SS	N	SP
3. Đia chỉ nguồn trong thao tác chuỗi	DS	CS,ES, SS	SI
4, Địa chỉ đích trong thao tác chuỗi	ES	N	DI
5. BP dùng như thanh ghi cơ sở	SS	CS, DS, ES	OFFSET
6. Các phép đọc ghi số liệu thông thường	DS	CS, ES, SS	OFFSET



Tập lệnh của MP 80286 có thể chia làm bảy nhóm như sau:

Nhóm lệnh chuyển số liệu

Nhóm lệnh xử lý chuỗi số liệu

Nhóm lệnh số học

Nhóm lệnh logic

Nhóm lệnh thay đổi trình tự thực hiện chương trình

Nhóm lệnh điều khiển bộ vi xử lý

Nhóm lệnh hỗ trợ ngôn ngữ bậc cao.

5.6. Quản lý bộ nhớ và chế độ địa chỉ ảo của 80x86

5.6.1. Các khái niệm mảng nhớ, không gian nhớ và nhiệm vụ

Đơn vị quản lý bộ nhớ cho phép chuyển các địa chỉ logic thành những địa chỉ vật lý của bộ nhớ.

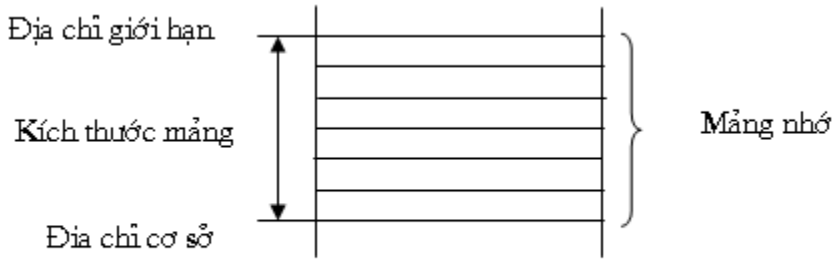
Một mảng nhớ được định nghĩa như là một tập hợp các ô nhớ liên tục không quá 64KB và có thể trao đổi giữa bộ nhớ trung tâm và bộ nhớ ngoài.

Mỗi mảng nhớ được định nghĩa bởi ba thông số:

Địa chỉ cơ sở (24 bit)

Kích thước mảng (độ dài mảng, 16 bit)

Quyền thâm nhập (8 bit) *(hình 5.6)*



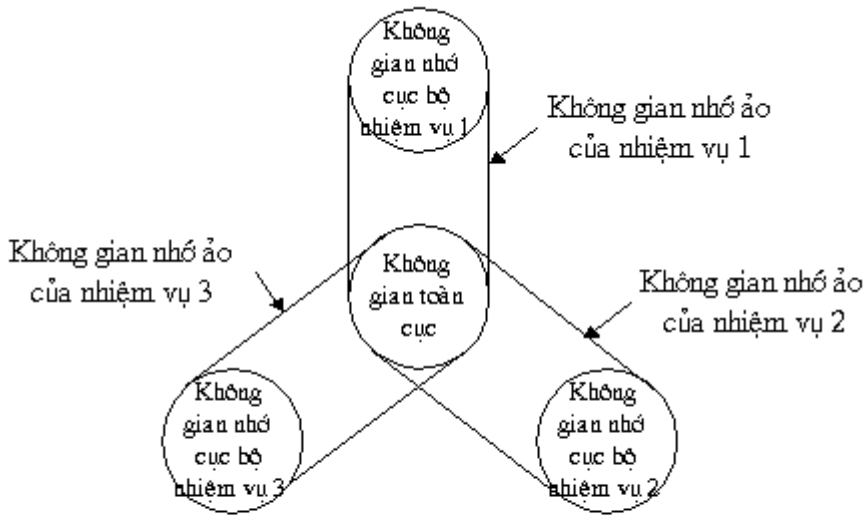
Hình 5.6: Định nghĩa một mảng

Ba thông số này tạo thành bộ mô tả mảng (Segment Descriptor)

Nhiệm vụ ở đây được định nghĩa ở mức thực hiện, khác với khái niệm nhiệm vụ trong hệ điều hành. Nhiệm vụ là việc thực hiện một tập hợp các quy trình gắn với một trạng thái xác định của bộ vi xử lý.

Không gian nhớ được định nghĩa gắn với nhiệm vụ.

Không gian nhớ được dành riêng cho một nhiệm vụ gọi là không gian nhớ cục bộ. Không gian nhớ mà tất cả các nhiệm vụ đều có thể thâm nhập tới gọi là không gian nhớ toàn cục. Nguyên lý này được làm sáng tỏ ở *hình 5.7*.



Hình 5.7: Cách ly các nhiệm vụ bằng cách chia không gian nhớ

5.6.2. Địa chỉ ảo

Một địa chỉ của ô nhớ được định nghĩa bởi hai thành phần

Bộ chọn mảng 16 bit

Offset 16 bit

Bộ chọn mảng này có ý nghĩa khác nhau, khi MP 80286 làm việc ở chế độ địa chỉ thực và chế độ bảo vệ *(hình 5.9)*.

Trong chế độ thực, bộ chọn mảng biểu diễn các bit cao của địa chỉ cơ sở của mảng nhớ.

Trong chế độ bảo vệ, bộ chọn mảng có ý nghĩa sau:

Hai bit thấp là mức đặc quyền yêu cầu RPL đối với mảng nhớ (Requested Privilege Level).

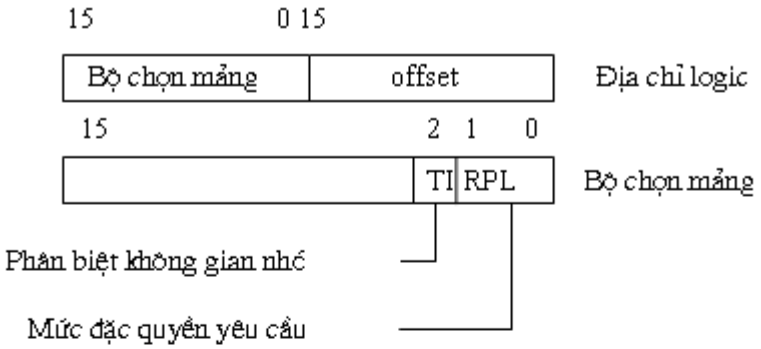
Bit TI (Table Indicator) xác định không gian nhớ.

TI = 0, trỏ tới bảng các bộ mô tả mảng của không gian nhớ toàn cục GDT (Global Descriptor Table).

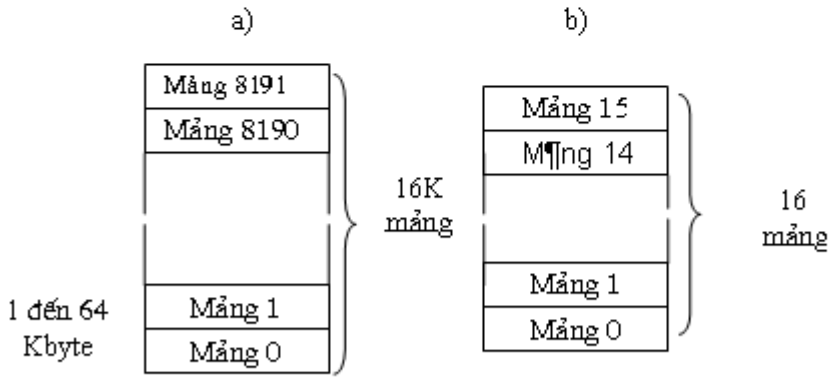
TI = 1, trỏ tới bảng các bộ mô tả mảng của không gian nhớ cục bộ LDT (Local Descriptor Table).

- 13 bit cao dành cho chỉ số INDEX, nó được sử dụng để trỏ tới 2^{13} = 8192 bộ mô tả mảng nhớ trong GDT và 8192 bộ mô tả mảng nhớ trong LDT. Tổng số mảng nhớ có thể được định nghĩa là 16384.

Dung lượng lớn nhất của mảng nhớ là 64kB. Vậy không gian nhớ ảo dành cho một nhiệm vụ có thể là một giga byte ($2*2^{13}$. $*2^{16} = 2^{30}$ = 1 GB). Dung lượng bộ nhớ vật lý lớn nhất trong hệ vi xử lý 80286 là 2^{24} = 16 MB.



Hình 5.8: Địa chỉ logic trong chế độ bảo vệ



Hình 5.9: Dung lượng địa chỉ hóa được cho một nhiệm vụ:
a) Chế độ bảo vệ b) Chế độ thực

Cách tính địa chỉ vật lý từ địa chỉ logic

Địa chỉ logic có 32 bit, gồm bộ chọn 16 bit và offset 16 bit. Bộ chọn có ba thành phần: chỉ số, TI và RPL.

TI cho biết bộ mô tả thuộc GDT hay LDT. Vì mỗi bộ mô tả mảng có 8 byte nên địa chỉ của bộ mô tả trong bảng sẽ là địa chỉ cơ sở của bảng cộng với chỉ số

nhân 8. MP 80286 sẽ tìm thấy trong bộ mô tả địa chỉ cơ sở của mảng nhớ vật lý và và giới hạn của nó. Cộng 24 bit địa chỉ cơ sở của mảng với 16 bit offset trong địa chỉ logic sẽ cho 24 bit địa chỉ vật lý của ô nhớ.

5.6.3. Bảng các bộ mô tả

Trong 80286 tồn tại hai loại bảng các bộ mô tả mảng.

GDT (Global Descriptor Table): bảng các bộ mô tả toàn cục (chung).

LDT (Local Descriptor Table): bảng các bộ mô tả mảng cục bộ

Đối với MP 80286 có bốn loại bộ mô tả mảng:

- Bộ mô tả mảng số liệu

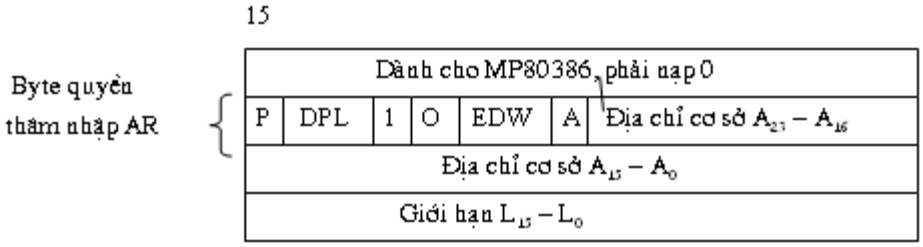
Bộ mô tả mảng lệnh

Bộ mô tả mảng hệ thống

Bộ mô tả cửa giao dịch

a. Bộ mô tả mảng số liệu

Bộ mô tả này quy chiếu tới mảng số liệu và mảng ngăn xếp. Dạng của bộ mô tả số liệu ở *hình 5.10*.



Hình 5.10: Bộ mô tả mảng số liệu

Tám byte của bộ mô tả chứa các thông tin mô tả về mảng như: địa chỉ cơ sở, giới hạn (độ dài) của mảng và quyền thâm nhập vào mảng. Hai byte đầu dành cho các bộ vi xử lý trong tương lai của INTEL, khi khởi động phải nạp giá trị 0.

Byte quyền thâm nhập có các bit sau:

P (Bit present)

Nếu mảng số liệu mà bộ mô tả quy chiếu tới đã được nạp trong bộ nhớ, thì bit P=1, ngược lại P=0.

Khi chương trình thâm nhập vào mảng số liệu, chưa được nạp vào bộ nhớ, sẽ gây ra ngoại lệ 11 hay 12. Chương trình xử lý ngoại lệ này sẽ nạp mảng số liệu cần thiết vào bộ nhớ từ thiết bị nhớ ngoài (đĩa từ).

DPL (Descriptor Privilege Level)

DPL cho biết mức đặc quyền của mảng số liệu mà bộ mô tả quy chiếu.

E (Executable)

E = 0 chỉ bộ mô tả là mảng số liệu,

ED (Expansion Direction)

ED chỉ chiều tiến triển của mảng số liệu. Nếu ED=1, thì mảng số liệu sẽ thuộc loại ngăn xếp. Địa chỉ bắt đầu của mảng sẽ là tổng của địa chỉ cơ sở và độ dài cực đại của mảng. Địa chỉ sẽ giảm dần về phía giới hạn của mảng. Nếu ED=0 thì chiều phát triển của mảng đi từ địa chỉ cơ sở tăng dần tới giới hạn.

W (Writable)

Nếu W = 1, thì mảng số liệu ký hiệu là RW (Read-Write) và có thể đọc và ghi được. Nếu W = 0, thì mảng số liệu được bảo vệ, cấm ghi. Mảng số liệu này có ký hiệu là RO (Read-Only).

A (Accesed)

Nếu mảng nhớ mà bộ mô tả quy chiếu đã được sử dụng, thì bit A=1. Một khi A đã được thiết lập thì chỉ có thể xóa bằng chương trình. Bit A giúp cho việc thống kê tần suất thâm nhập vào mảng số liệu của một chương trình.

b. Bộ mô tả mảng lệnh

Bộ mô tả này quy chiếu tới mảng nhớ chứa chương trình (lệnh).

Bộ mô tả mảng lệnh có dạng tương tự như bộ mô tả mảng số liệu, riêng byte quyền thâm nhập có một số bit thay đổi

E = 1 để chỉ bộ mô tả quy chiếu tới mảng lệnh

Bit ED thay bằng C

Bit W thay bằng R

Nếu R= 0 thì mảng lệnh chỉ có thể thực hiện được và ký hiệu là EO (Executable Only). Nếu R = 1, thì mảng lệnh không những thực hiện mà còn đọc được, nên ký hiệu là ER (Executale and Read).

Nếu C = 0, thì chương trình con được gọi sẽ thực hiện với mức đặc quyền bằng DPL, trong bộ mô tả của mảng chứa chương trình con.

Nếu C = 1, thì chương trình con được gọi sẽ thực hiện với mức đặc quyền bằng DPL trong bộ mô tả quy chiếu mảng chứa chương trình gọi chương trình con đó.

c. Bộ mô tả mảng hệ thống

Bộ mô tả mảng này quy chiếu đến các mảng chứa thông tin cần cho hệ thống

Nếu kiểu =1, thì mô tả quy chiếu tới mảng chứa trạng thái của nhiệm vụ TSS (Task State Segment). Nhiệm vụ này không ở trạng thái đang thực hiện.

Nếu kiểu = 3, bộ mô tả quy chiếu mảng TSS của một nhiệm vụ tích cực.

Nếu kiểu = 2, bộ mô tả quy chiếu mảng chứa bảng các bộ mô tả cục bộ LDT.

Chúng ta hãy xem các thông tin chứa trong bảng các bộ mô tả.

Trong GDT chứa các bộ mô tả mảng tương ứng với tất cả các mảng nhớ trong không gian nhớ toàn cục. Trong LDT chứa các bộ mô tả mảng tương ứng với tất cả các mảng nhớ trong không gian nhớ cục bộ của một nhiệm vụ.

Mỗi bảng các bộ mô tả, cũng chính là một mảng nhớ được định nghĩa bằng một bộ mô tả mảng đặc biệt, thuộc nhóm bộ mô tả mảng hệ thống.

GDT là một bảng duy nhất nên không cần xác định bằng một bộ mô tả riêng. Địa chỉ cơ sở và kích thước của mảng GDT được chứa trong một thanh ghi đặc biệt gọi là thanh ghi bảng các bộ mô tả toàn cục GDTR (Global Descriptor Table Register).

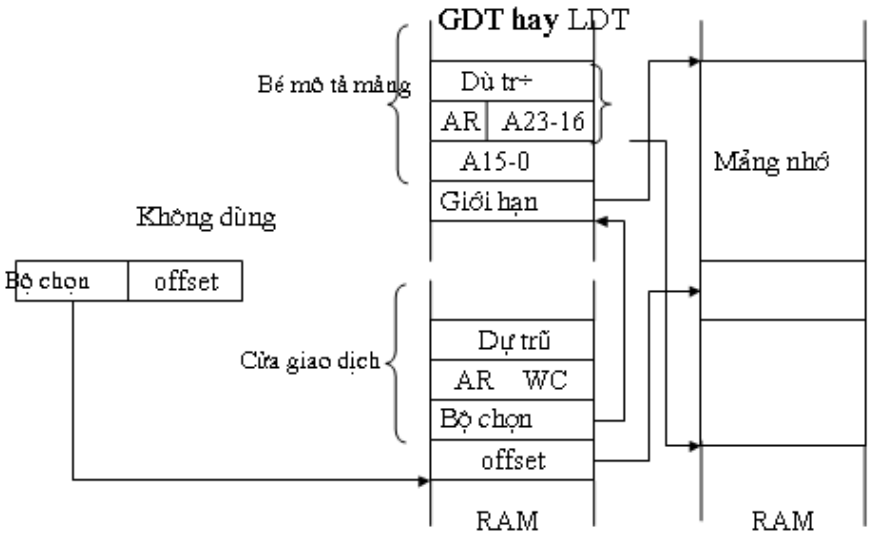
LDT được xác định bằng các bộ mô tả ở trong bảng GDT. Thông tin về địa chỉ cơ sở và kích thước của mảng chứa bảng các bộ mô tả cục bộ tương ứng với nhiệm vụ đang thực hiện được chứa trong thanh ghi bảng các bộ mô tả cục bộ LDTR (Local Deseriptor Table Register).

d. Bộ mô tả kiểu cửa giao dịch

Các lệnh CALL và JMP chỉ có thể thâm nhập vào mảng lệnh có mức đặc quyền cao hơn thông qua một cổng ghép nối gọi là cửa giao dịch. Bộ mô tả cửa giao dịch có dạng ở *hình 5.11*.

Có tất cả 4 loại cửa giao dịch:

- Cửa giao dịch kiểu gọi (CALL gate)
- Cửa giao dịch kiểu bẫy (TRAP gate)
- Cửa giao dịch kiểu ngắt (Interrupt gate)
- Cửa giao dịch nhiệm vụ (Task gate)



Hình 5.11: Cơ chế thâm nhập mảng nhớ thông qua cửa giao dịch CALL

5.6.4. Bảo vệ bộ nhớ

Bảo vệ bộ nhớ có ba mục đích:

Cách ly chương trình hệ thống và chương trình ứng dụng.

Cách ly giữa các nhiệm vụ.

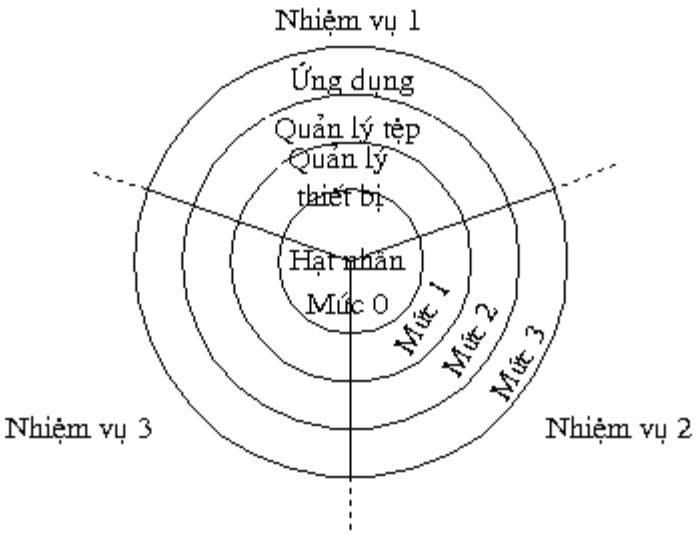
Kiểm chứng thời điểm thâm nhập vào đối tượng nhớ.

Các mức đặc quyền

MP 80286 cho phép sinh ra bốn mức đặc quyền. Mức 0 là mức đặc quyền cao nhất. Mức 3 là mức đặc quyền thấp nhất. Mỗi một mảng được phân bổ một mức đặc quyền. Chương trình cấu tạo từ các mảng số liệu và mảng lệnh. Mức đặc quyền phân bổ cho một chương trình cho biết chương trình có thẩm quyền làm những gì khi nó được thực hiện bởi một nhiệm vụ.

Mức đặc quyền của một nhiệm vụ thay đổi theo thời gian và phụ thuộc vào mức đặc quyền của chương trình đang chạy.

Sự phân cấp mức đặc quyền được thể hiện ở *hình 5.12*.



Hình 5.12: Sự phân cấp mức đặc quyền

Trong đó:

Nhân của Hệ điều hành gồm các chương trình sơ đẳng quản lý các tài nguyên của MP 80286 và bộ nhớ. Nhân phải gọn, có khả năng vận hành tốt, không bị hỏng do phần mềm của các lớp có mức đặc quyền thấp hơn.

Mức 1 chứa tất cả phần mềm liên quan tới quản lý hệ điều hành như: thiết lập mức ưu tiên giữa các nhiệm vụ, nạp thuật toán trao đổi (swapping) và quản lý các cổng vào/ra.

Mức 2 bao gồm các chức năng quản lý tệp, quản lý thư viện. Đó là các bộ ghép nối mềm, bậc cao cho các chương trình ứng dụng.

Mức 3 dành cho các chương trình ứng dụng.

Nguyên lý bảo vệ bộ nhớ đòi hỏi mỗi đối tượng nhớ phải có bộ mô tả cho biết mức đặc quyền của đối tượng đó.

5.7. Hệ lệnh cơ bản của họ vi xử lý 80x86

5.7.1. Nhóm lệnh chuyển số liệu

Đích Nguồn

MOV M/R$_1$, M/R$_2$

Chuyển 1 byte hay từ nguồn tới đích

$R_1 \leftarrow (R_2)$

$M \leftarrow (R_2)$

$R_1 \leftarrow (M)$

Ví dụ: MOV AX, CX

MOV [BX] + 20h, CX

MOV DX, [300h]

MOV M/R, DATA

Chuyển 1 byte (8bit) hay từ (16 bit) số liệu vào thanh ghi. Hay ô nhớ

Ví dụ: MOV DX, 03F8h

Ghi chú: Không chuyển trực tiếp số liệu vào các thanh ghi mảng được mà phải chuyển gián tiếp qua thanh ghi khác.

Ví dụ MOV AX, 2000h

MOV DS, AX

PUSH M/R

Chuyển nội dung của ô nhớ hay thanh ghi vào đỉnh của ngăn xếp.

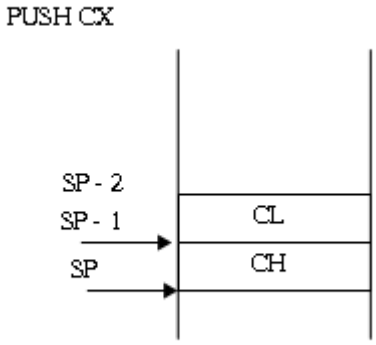
Các bước thực hiện lệnh PUSH:

Giảm SP đi 1

Nạp byte cao vào đỉnh ngăn xếp

Giảm SP di 1

Nạp byte thấp vào đỉnh ngăn xếp



Hình 5.13: Lệnh Push

Ghi chú – lệnh PUSH đối với thanh ghi cờ được viết PUSHF

PUSH làm việc với thanh ghi hay ô nhớ 16 bit

Không có PUSH số liệu trực tiếp, muốn lưu số liệu vào ngăn xếp phải chuyển vào thanh ghi hay ô nhớ rồi mới dùng lệnh PUSH.

POP M/R

Lấy 2 byte từ đỉnh ngăn xếp nạp vào ô nhớ (16 bit) hay thanh ghi (16 bit).

Các bước thực hiên lệnh POP:

Đọc byte thấp từ đỉnh ngăn xếp nạp vào phần thấp của thanh ghi hay ô nhớ

Tăng SP thêm 1

Đọc byte cào từ đỉnh ngăn xếp nạp vào phần cao của thanh ghi hay vào ô nhớ

Tăng SP thêm 1

Ghi chú: Lệnh POP đối với thanh ghi cờ được viết POPF

XCHG M/R1, M/R$_2$

Trao đổi nội dung của thanh ghi với thanh ghi hay ô nhớ với thanh ghi.

Các thanh ghi có thể có độ dài 8 bit hay 16 bit.

Ví dụ: XCHG BX, [BP + SI]

Ghi chú: Nếu R1 là AX thì nó được ngầm định không cần viết ở lệnh.

Ví dụ: - XCHG DX

- XCHG AX, AX dùng như lệnh NOP (3 chu kỳ)

IN ACC, PORT

Nạp vào ACC số liệu từ cổng vào/ra

ACC = AL 8 bit

ACC = AX 16 bit

PORT là địa chỉ của cổng vào/ra

Nếu địa chỉ của cổng lớn hơn FFh thì địa chỉ của cổng phải chứa trong thanh ghi DX. Để vào số liệu từ cổng vào/ra phải thực hiện 2 lệnh:

MOV DX, PORT

IN ACC, DX

Ví dụ: Địa chỉ cổng vào/ra số liệu của cổng COM1 là 3F8h, muốn nhập một byte dữ liệu từ cổng COM1 thực hiện hai lệnh:

Ví du: MOV DX, 3F8h

IN AL, DX

OUT PORT, ACC

Đưa số liệu từ ACC ra cửa vào/ra

Nếu ACC = AL 8 bit

ACC = AX 16 bit

PORT = là địa của cổng ra

Nếu địa của cửa ra lớn hơn FFh thì địa chỉ của cửa phải chứa vào DX.

Ví du: MOV DX, PORT

OUT DX, AL

XLAT

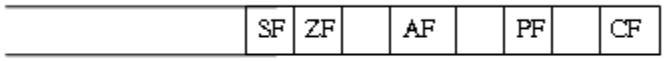
Nạp nội dung của ô nhớ các địa chỉ là

DS: [BX] + [AL] vào AL

Lệnh này dùng để tạo bảng và tra bảng.

LAHF

Chuyển (nội dung) byte thấp của thanh ghi cờ vào thanh ghi AH.



Hình 5.14: Lệnh LAHF

SAHF

Lưu nội dung của thanh ghi AH vào byte thấp của thanh ghi cờ.

LEA R, M

Nạp vào thanh ghi R, OFFSET của ô nhớ M

R ← OFFSET(M)

LES R, M

Nạp vào thanh ghi R nội dung của 2 byte nhớ trỏ bởi M và vào thanh ghi ES nội dung của 2 byte ô nhớ tiếp theo.

Ví dụ:

LES DI, [BX]

Nếu

(DS) = 1000h

(BX) = 080Ah

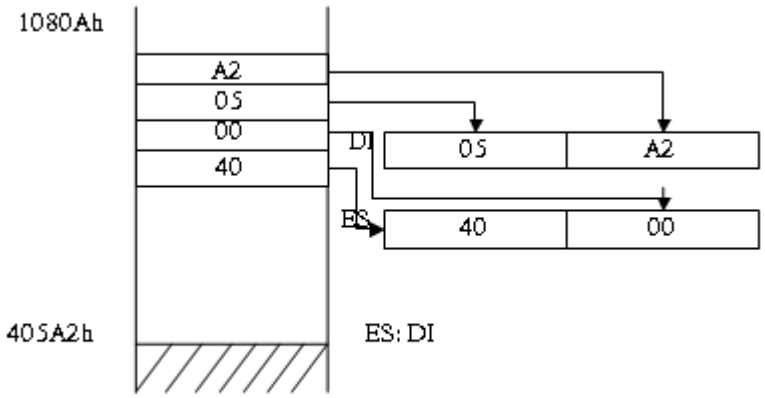
(1080Ah) = 05A2h

(1080CH) = 4000h

thì

(DI) = 05A2H

(ES) = 4000H



Hình 5.15: Lệnh LES

LDS R,M

Nạp vào thanh ghi R nội dung của 2 byte nhớ trỏ bởi M và và thanh ghi mảng DS, 2 byte nhớ tiếp theo.

Ví dụ: LDS [SI], 10h

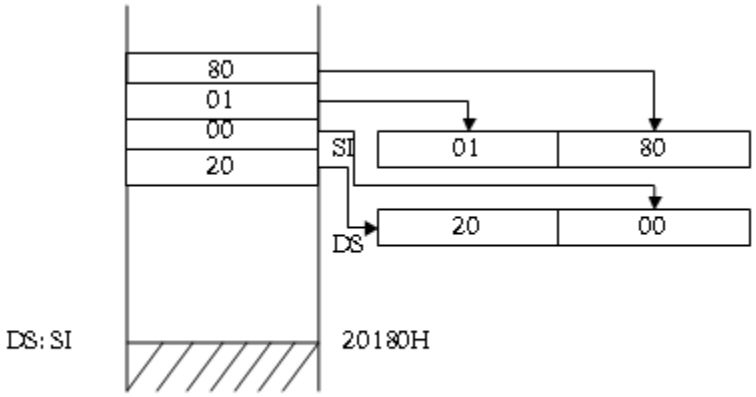
Nếu (DS) = C000h

(C0010h) = 0180h

(C0012h) = 2000h

thì (SI) = 0180h

(DS) = 2000h



Hình 5.16: Lệnh LDS

5.7.2. Nhóm lệnh số học

a. Phép cộng

ADD M/R, DATA

Cộng nội dung của M/R với số liệu tức thời chứa trong lệnh, kết quả chuyển vào thanh ghi hay ô nhớ đích.

M/R ← (M/R) + DATA

Ví dụ ADD AX, 0F0FH

Nếu

(AX) = 4064H

thì (AX) = 4F73H

ADD M/R_1 M/R_2

Cộng nội dung 2 thanh ghi hay nội dung của thanh ghi với nội dung 1 ô nhớ, kết quả chuyển vào thanh ghi hay ô nhớ đích.

$R_1 \leftarrow (R_1) + (R_2)$ Ví dụ: ADD BX, DX

$M \leftarrow (M) + (R_2)$ Ví dụ: ADD [BX] + 30h, CX

$R_1 \leftarrow (R_1) + (M)$ Ví dụ: ADD DX, [SI] + 3h

Không được phép cộng nội dung của 2 ô nhớ.

ADC M/R, DATA

Cộng nội dung của M/R với số liệu tức thời trong lệnh và với nội dung của cờ CF, kết quả chuyển vào thanh ghi hay ô nhớ đích.

ADC M/R$_1$, M/R$_2$

$R_1 \leftarrow (R_1) + (R_2) + CF$

$M \leftarrow (M) + (R_2) + CF$

$R_1 \leftarrow (R_1) + (M) + CF$

INC M/R

Tăng nội dung của thanh ghi hay ô nhớ thêm 1

M/R ← (M/R) + 1

Ghi chú: Lệnh này không dùng cho các thanh ghi mảng.

DAA

Hiệu chỉnh thập phân sau phép cộng cho thanh ghi AL.

Ví dụ: ADD AL, BL

DAA

Quy tắc:

Nếu 4 bit thấp là số lớn hơn 9 hay cờ AF = 1 (có nhớ từ bit D3 sang bit D4) thì cộng 6 vào nửa thấp của AL và đưa AF = 1.

Sau bước 1, nếu 4 bit cao của AL là số lớn hơn 9 hay cờ CF = 1 thì cộng vào thêm 6 và 4 bit cao và lập CF = 1.

AAA

Hiệu chỉnh kết quả trong thanh ghi AL khi cộng 2 số ASCII

Quy tắc:

Nếu 4 bit thấp của thanh ghi AL là số nhỏ hơn hoặc bằng 9 và cờ AF = 0 thì bỏ qua bước 2

Nếu 4 bit thấp của thanh ghi cờ AL là số lớn hơn hoặc cờ AF = 1 thì cộng 6 vào 4 bit thấp của thanh ghi AL và cộng 1 vào AH rồi thiết lập AF = 1

Xóa 4 bit cao của AL

Lập cờ CF = giá trị của AF

b. Phép trừ

SUB M/R, DATA

Lấy nội dung của ô nhớ hay thanh ghi đích M/R trừ đi số liệu trong lệnh rồi nạp kết quả vào ô nhớ hay thanh ghi đích M/R.

Phép trừ thực hiện theo số bù 2.

$M/R \leftarrow (M/R) - DATA$

SUB M/R$_1$, M/R$_2$

Trừ nội dung của thanh ghi hay ô nhớ đích (M/R1) đi nội dung thanh ghi hay ô nhớ nguồn (M/R2), kết quả đặt ở thanh ghi hay ô nhớ đích.

$R_1 \leftarrow (R_1) - (R_2)$

$M \leftarrow (M) - (R_2)$

$R_1 \leftarrow (R_1) - (M)$

Không có lệnh trừ giữa 2 ô nhớ.

Ví dụ: SUB CX, BX

SUB [SI], DX

SUB CX, [BX] [DI]

SBB M/R, DATA

Lấy nội dung của ô nhớ hay thanh ghi đích M/R trừ đi số liệu trong lệnh và cờ nhớ CF rồi nạp kết quả vào ô nhớ hay thanh ghi đích M/R.

$M/R \leftarrow (M/R) – DATA – (CF)$

SBB M/R$_1$, M/R$_2$

Trừ nội dung của thanh ghi hay ô nhớ đích (M/R1) đi nội dung của thanh ghi hay ô nhớ nguồn (M/R2) và cờ (CF), kết quả đặt ở (M/R_1).

$R_1 \leftarrow (R_1) - (R_2) - CF$

$M \leftarrow (M) - (R_2) - CF$

$R_1 \leftarrow (R_1) - (M) - CF$

DEC M/R

Giảm nội dung của thanh ghi R hay ô nhớ đi 1.

$M/R \leftarrow (M/R) – 1$

NEG M/R

Chuyển thành số bù 2 nội dung của ô nhớ hay thanh ghi.

$M/R \leftarrow (M/R) +1$

CMP M/R, DATA

So sánh nội dung của (M/R) với số liệu DATA.

Thực tế là thực hiện phép trừ nội dung của (M/R) đi số liệu nhưng kết quả không đặt lại (M/R) mà chỉ thay đổi các cờ.

CF, OF, PF, SF, ZF

CMP M/R$_1$, M/R$_2$

Lệnh này so sánh nội dung của thanh ghi hay ô nhớ, tuy nhiên không được so sánh giữa 2 ô nhớ. Thực tế, lệnh này làm phép trừ nhưng không ghi kết quả vào đích mà chỉ thay đổi cờ.

CMP R_1 , R_2

CMP M, R_2

CMP R_1, M

DAS

Hiệu chỉnh thập phân ở AL sau khi trừ.

Quy tắc:

Nếu 4 bit thấp của AL là số lớn hơn 9 hoặc cờ AF = 1 thì trừ (AL) đi 6 và lập AF = 1

Sau bước 1, nếu 4 bit cao của AL là số lớn hơn 9 hay CF = 1 thì trừ (AL) đi 60H và lập CF = 1

AAS

Hiệu chỉnh kết quả sau khi trừ ở AL.

Thực tế, lệnh này chuyển mã ASCII ở AL sang hai số BCD không nén (unpacked)

Quy tắc:

Nếu 4 bit thấp của AL là số nhỏ hơn hoặc bằng 9 và AF = 0 thì chuyển sang bước 3

Nếu 4 bit thấp của AL là số lớn hơn 9 hoặc AF = 1 thì lấy nội dung (AL) trừ đi 6, và (AH) trừ đi 1 và lập AF = 1

Xóa 4 bit cao trong AL

Lập CF bằng giá trị của AF

MUL M/R

Nhân số không dấu

Trường hợp thừa số là 8 bit: Nhân nội dung của thanh ghi hay ô nhớ M/R với nội dung của thanh ghi AL và kết quả là 16 bit đặt ở thanh ghi AX.

Biểu diễn

AX ← (AL) * (M/R)

Trường hợp thừa số là 16 bit: Nhận nội dung của thanh ghi hay ô nhớ (16 bit) với nội dung của thanh ghi AX và kết quả là 32 bit đặt ở đôi thanh ghi DX: AX.

Nếu nửa cao là 0 thì cờ OF và CF = 0.

Nếu nửa cao là khác 0 thì cờ OF và CF = 1.

Số không dấu thì 8 bit sẽ có giới hạn số là 0 – 255

còn 16 bit giới hạn số sẽ là 0 – 65.535.

IMUL M/R

Nhân số có dấu.

Tương tự như lệnh MUL.

Số có dấu thì 8 bit chỉ có phạm vi số

-128 đến +127

16 bit

- 32768 đến + 32767

Ví dụ: Nhân 3A62h với 2B14h

MOV AX, 3A62h

MOV CX, 2B14h

MUL CX

MOV M_1, AX

MOV M_2, DX

AAM

Hiệu chỉnh ASCII kết quả sau khi nhân.

Thực hiện chuyển kết quả là số nhị phân ở AL thành 2 số BCD ở AH và AL.

Quy tắc:

Chia (AL) cho 0Ah thương ở AH và số dư ở AL

Thiết lập các cờ

PF trên cơ sở kết quả ở AL

SF trên cơ sở kết quả D7 của AL

ZF trên cơ sở kết quả ở AL

Biểu diễn AH <- (AL) :0Ah

AL ← (AL) % 0Ah

DIV M/R

Chia số không dấu

Trường hợp thừa số là 8 bit:

Chia nội dung của (AX) cho nội dung của thanh ghi hay ô nhớ kết quả thương ở AL và số dư ở AH.

AL ← (AX) : (M/R)

Trường hợp thừa số là 16 bit

Chia nội dung của cặp thanh ghi DX:AX cho nội dung thanh ghi hay ô nhớ. Kết quả thương ở AX và số dư ở DX.

Nếu kết quả thương số lớn hơn FFh (Trường hợp thừa số là 8 bit) hay FFFFH (Trường hợp thừa số là 16 bit) thì gây ra ngắt số 0 (Ngắt xẩy ra khi chia cho số quá nhỏ).

IDIV M/R

Chia số có dấu

Tương tự như chia số không dấu, phạm vi số nhỏ hơn.

AAD

Hiệu chỉnh 2 số BCD không nén ở AH và AL thành số nhị phân ở AL trước khi thực hiện phép chia.

Nhân nội dung của AH với 0Ah

Cộng nội dung của AH vào AL

Nạp 00 vào AH

Các cờ sẽ thiết lập

- CF, OF, AF không xác định
- PF, ZF, SF xác định theo kết quả ở AL

CBW

Chuyển byte thành từ

Mở rộng dấu của thanh ghi AL sang AH.

Nếu bit D7 của AL là 1 thì nạp FFh vào AH

Nếu bit D7 của AL là 0 thì nạp 00h vào AH.

Lệnh này được sử dụng khi thực hiện phép chia có dấu (số chia 8 bit), nhưng số bị chia nhỏ chỉ chứa ở thanh ghi AL cần mở rông bit dấu sang thanh ghi AH

CWD

Chuyển từ thành từ đúp.

Mở rộng dấu của thanh ghi AX sang DX.

Nếu bit D15 của AX là 1 thì nạp FFFFh vào DX

Nếu bit D15 của AX là 0 thì nạp 0000h vào DX.

Lệnh này được sử dụng khi thực hiện phép chia có dấu (số chia 16 bit), nhưng số bị chia nhỏ chỉ chứa ở thanh ghi AX cần mở rộng bit dấu sang thanh ghi DX.

5.7.3. Nhóm lệnh logic và dịch chuyển

a. Các lệnh logic

NOT R/M

Đảo nội dung của thanh ghi hay ô nhớ

$R/M \leftarrow \overline{R/M}$

Ví dụ: NOT BL

Nếu trước phép tính (BL) = FBh thì sau phép tính (BL) = 04h

AND M/R, DATA

AND M/R$_1$, M/R$_2$

$R_1 \leftarrow (R_1)$ AND (R_2)

$M \leftarrow (M)$ AND (R_2)

$R_1 \leftarrow (R_1)$ AND (M)

Phép AND có thể ứng dụng để xóa một số bit nào đó và giữ nguyên giá trị các bit khác của thanh ghi hay ô nhớ đích.

Ví dụ (AL) = 0110 1101

(BL) = 1101 1011

AND AL, BL

0110 1101

AND 1101 1011

0100 1001

TEST M/R, DATA

TEST M/R$_1$, M/R$_2$

Tương tự phép AND logic nhưng kết quả không đưa vào thanh ghi hay ô nhớ đích.

Phép TEST có thể ứng dụng để kiểm tra xem một bit nào của thanh ghi hay ô nhớ đích là 0 hay 1 logic .

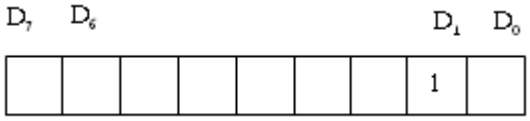
Ví dụ:

Một TB vào/ra có

thanh ghi trạng thái với địa chỉ là 3A0h

thanh ghi số liệu với địa chỉ là 3A1h

Thanh ghi trạng thái có nội dung như sau:



Giả thiết nếu bit D1 = 1 thì cho biết TB sẵn sàng, nếu D1 = 0 thì TB chưa sẵn sàng chuyển số liệu. Hãy viết đoạn mã ngữ để kiểm tra, nếu TB sẵn sàng thì gửi ký tự 'A' ra TB.

```
        MOV DX, 3A0h
Wait:   IN AL, DX
        TEST AL, 02h
        JZ Wait
        INC DX
        MOV AL, 41h
        OUT DX, AL
```

OR M/R, DATA

OR M/R$_1$, M/R$_2$

$R_1 \leftarrow (R_1)$ OR (R_2)

$M \leftarrow (M)$ OR (R_2)

$R_1 \leftarrow (R_1)$ OR (M)

Phép OR có thể ứng dụng để thiết lập lên 1 một số bit nào đó và giữ nguyên giá trị các bit khác của thanh ghi hay ô nhớ đích.

Ví dụ: (AL) = 0110 1001

(BL) = 1101 1011

OR AL, BL

0110 1001

OR 1101 1011

1111 1011

XOR M/R, DATA

XOR M/R$_1$, M/R$_2$

$R_1 \leftarrow (R_1)$ XOR (R_2)

$M \leftarrow (M)$ XOR (R_2)

$R_1 \leftarrow (R_1)$ XOR (M)

Phép XOR có thể ứng dụng để đảo một số bit nào đó và giữ nguyên giá trị các bit khác của thanh ghi hay ô nhớ đích.

Ví dụ: (AL) = 0110 1001

(BL) = 1101 1011

XOR AL, BL

0110 1001

XOR 1101 1011

1011 0010

Ghi chú:

Các phép tính logic AND, OR, XOR, TEST chỉ ảnh hưởng đến các cờ SF, ZF, PF còn các cờ OF và CF thì xóa về 0.

b. Các lệnh dịch chuyển

SHL M/R, Count

SAL M/R , Count

Dịch trái nội dung của thanh ghi hay ô nhớ số lần là count.

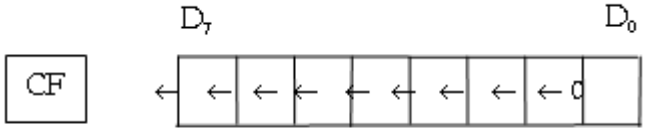
Nếu 1 lần thì count viết trực tiếp là 1.

Nếu count > 1 thì phải nạp count vào CL và thực hiện

Ví dụ: Muốn dịch trái thanh ghi SI 3 lần thì

MOV CL, 03

SHL SI, CL



Khi dịch trái các bit, bit thấp nhất sẽ được đưa giá trị 0 vào

SHR M/R, count

Dịch phải logic thanh ghi hay ô nhớ count lần

Nếu count = 1 thì lệnh có dạng SHR R/M, 1

Nếu count > 1 thì lệnh có dạng:

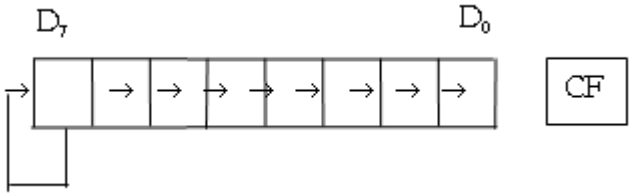
MOV CL, count

SAR M/R , CL

SAR M/R, count

Dịch phải số học nội dung của ô nhớ hay thanh ghi count lần.

Chú ý: bit dấu được giữ nguyên.



Nếu count = 1 thì lệnh có dạng SAR R/M, 1

Nếu count > 1 thì lệnh có dạng:

MOV CL, count

SAR R/M, CL

Ghi chú:

Các phép SHL, SAL, SAR, SHR

thay đổi cờ CF

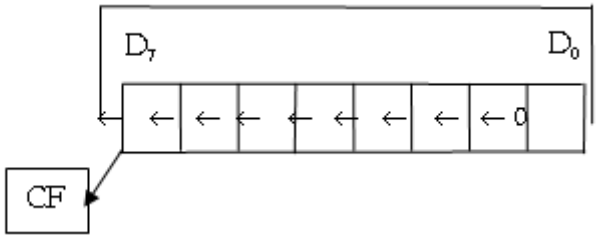
Nếu dịch 1 và có thay đổi cờ dấu thì - OF thay đổi

SHR thay đổi cả các cờ SF, ZF, PF

Các lệnh quay vòng

ROL M/R, count

Quay trái thanh ghi hay ô nhớ



Nếu count = 1 thì lệnh có dạng: ROL R/M, 1

Nếu count >1 thì lệnh có dạng: MOV CL , count

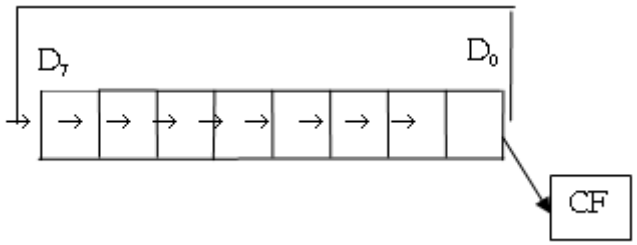
ROL R/M, CL

Ví dụ: MOV CL, 3

ROL BX, CL

ROR M/R, count

Quay phải nội dung của thanh ghi hay ô nhớ count lần.



Nếu count = 1 thì lệnh có dạng: ROL R/M, 1

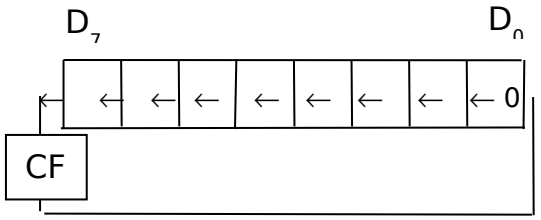
Nếu count >1 thì lệnh có dạng: MOV CL , count

ROL R/ M, CL

RCL M/R, count

Quay trái nội dung của ô nhớ hay thanh ghi qua cờ CF count lần.

Ví dụ: MOV



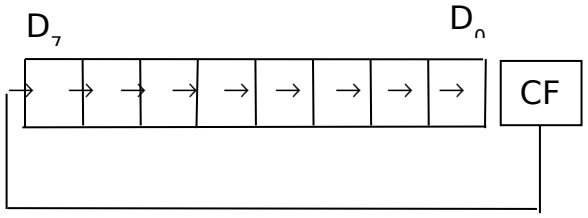
Nếu count = 1 thì lệnh có dạng: RCL M/R, 1

Nếu count khác 1 thì lệnh có dạng: MOV CL, count

RCL M/R, CL

RCR M/R, count

Quay phải nội dung của thanh ghi hay ô nhớ count lần qua cờ nhớ CF.



Nếu count = 1 thì lệnh có dạng: RCR M/R, 1

Nếu count khác 1 thì lệnh có dạng: MOV CL, count

RCR M/R, count

5.7.4. Nhóm lệnh thao tác chuỗi

REP

Là tiền lệnh. Các lệnh thao tác chuỗi sau lệnh REP sẽ được thực hiện số lần bằng nội dung của thanh ghi CX. Mỗi lần thực hiện thanh ghi CX giảm đi 1 và kết thúc khi (CX) = 0.

REPE (REPZ)

REPE và REPZ là đồng nghĩa.

Các lệnh xử lý chuỗi sau REPE sẽ được lặp lại cho đến khi (CX)=0 hay cờ ZF=0.

Lệnh này dùng để so sánh hai dãy kí tự cho đến khi có kí tự khác nhau thì dừng hay tìm kiếm kí tự có giá trị khác kí tự cho trước.

REPNE (REPNZ)

REPNE và REPNZ là hai lệnh đồng nghĩa.

Các lệnh xử lý chuỗi sau REPNE được lặp lại cho đến khi (CX)=0 hay cờ ZF=1.

Dùng để so sánh 2 dãy kí tự cho đến khi có ký tự bằng nhau hay tìm kiếm trong dãy ký tự một ký tự có giá trị bằng giá trị cho trước thì dừng.

Lệnh MOVS (MOVSB, MOVSW)

MOVSB dùng cho chuyển từng byte.

MOVSW dùng cho chuyển từng từ.

Chuyển nội dung của ô nhớ có địa chỉ offset trỏ bởi SI vào ô nhớ có địa chỉ offset trỏ bởi DI.

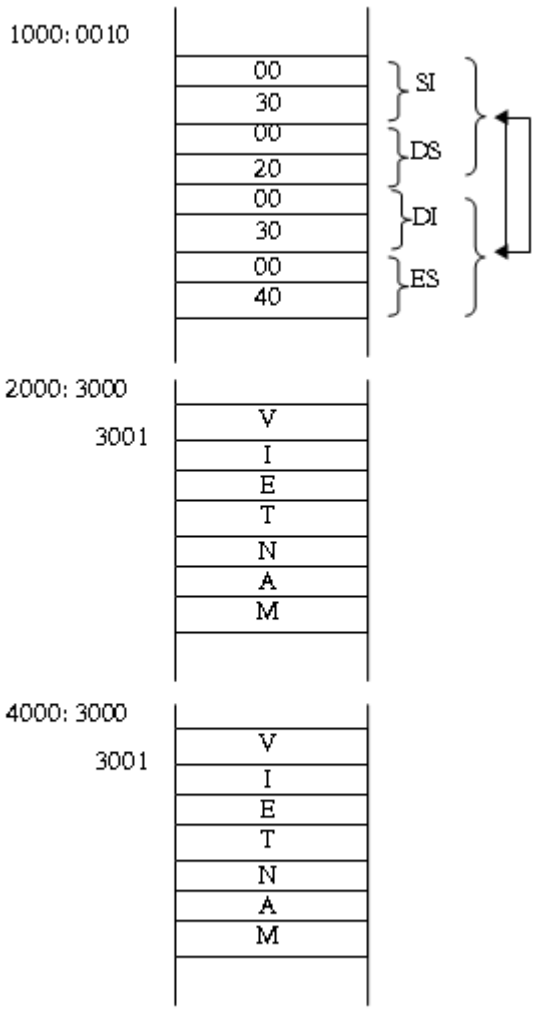
Sau mỗi lần chuyển địa chỉ của ô nhớ nguồn và đích (nội dung của SI và DI) tăng hay giảm đi 1 phụ thuộc vào giá trị của cờ hướng DF.

DF = 0 thì (DI) và (DI) tăng 1

DF = 1 thì (SI) và (DI) giảm 1

Ví dụ:

```
CLD
MOV AX, 1000
MOV DS, AX
MOV CX, 7
LES DI, [0014h]
LDS SI, [0010h]
REP MOVSB
```



Hình 5.17 : Lệnh MOVS

CMPS (CMPSP, CMPSW)

CMPSB dùng cho so sánh từng byte

CMPSW dùng cho so sánh từng từ

So sánh nội dung ô nhớ trỏ bởi SI với nội dung ô nhớ trỏ bởi DI.

Sau mỗi lần so sánh nội dung của SI hay DI tăng hay giảm 1 phụ thuộc vào cờ hướng DF

DF = 0 thì (SI) và (DI) tăng 1

DF = 1 thì (SI) và (DI) giảm 1

Lệnh này để sau tiền lệnh REPE hay REPNE

SCAS (SCASB, SCASW)

SCASB dùng cho so sánh AL với từng byte

SCASW dùng cho so sánh AX với từng từ từ

So sánh nội dung của AL (AX) với nội dung của ô nhớ trỏ bởi DI.

Sau mỗi lần so sánh DI tăng hay giảm phụ thuộc vào DF.

DF = 0 thì (DI) tăng 1

DF = 1 thì (DI) giảm 1

LODS (LODSB, LODSW)

LODSB dùng cho nạp từng byte byte vào AL

LODSW dùng cho nạp từng từ vào AX

Chuyển nội dung ô nhớ trỏ bởi SI vào AL (AX)

Chiều tăng hay giảm của SI phụ thuộc vào cờ DF.

Thường không dùng sau REP mà dùng trong vòng LOOP.

Ví dụ: Xây dựng một đoạn chương trình để đưa từng ký tự của một chuỗi ký tự chứa trong bộ đệm vào thanh ghi AL rôi đưa ra thiết bị ra.

STOS (STOSB, STOSW)

STOSB dùng cho lưu từng byte

STOSW dùng cho lưu từng từ

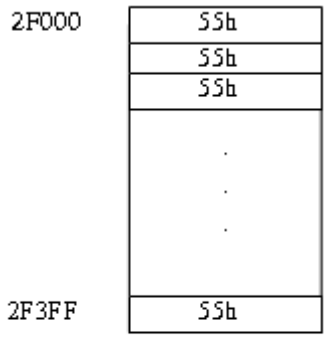
Chuyển nội dung của AL (AX) tới ô nhớ trỏ bởi DI.

Chiều tăng hay giảm của DI phụ thuộc vào cờ DF.

Ví dụ: Xây dựng một đoạn chương trình để đọc từng ký tự từ thiết bị vào/ra vào thanh ghi AL rồi đưa vào chứa trong bộ đệm .

Ví dụ: Lưu vào vùng nhớ có địa chỉ bắt đầu là 2F000H, 1KB có giá trị 55h

```
MOV DX, 2F00h
MOV ES, DX
XOR DI, DI
MOV CX, 1024
CLD
MOV AL, 55h
REP STOSB
```



Hình 5.19: Lệnh STOS

5.7.5. Nhóm lệnh điều khiển chương trình

Lệnh gọi chương trình con (CALL)

Lệnh CALL về phương diện mã lệnh được chia làm 4 loại

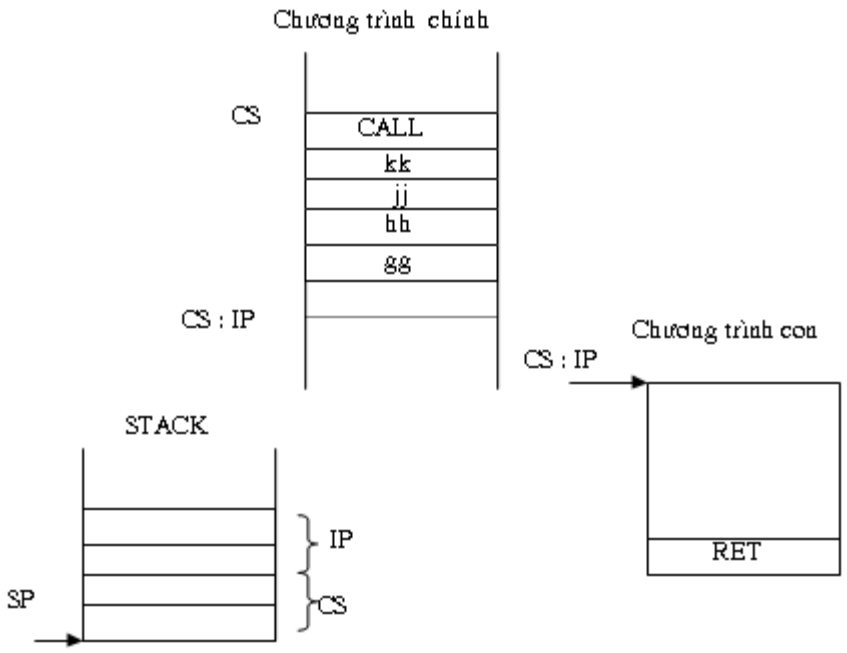
CALL address (gọi xa trực tiếp xa)

Lệnh này có nghĩa là địa chỉ của chương trình con trực tiếp trong lệnh và chương trinh con và chương trình chính ở hai mảng nhớ khác nhau. Mã lệnh gồm 5 byte, có dạng tổng quát:

op	kk	jj	hh	gg
	thấp	cao	thấp	cao
	IP		CS	

Các thao tác để thực hiện lệnh:

- Lưu CS hiện hành (của chương trình gọi vào ngăn xếp).
- Lưu địa chỉ offset của lệnh sau lệnh CALL vào ngăn xếp.
- Nạp vào IP nội dung 2 byte sau mã lệnh.
- Nạp vào CS nội dung 2 byte tiếp theo.
- Thực hiện lệnh ở CS : IP mới.



Hình 5.20: Lệnh CALL Address

CALL M (gọi xa gián tiếp xa)

Lệnh này có nghĩa là địa chỉ của chương trình con được địa chỉ gián tiếp qua ô nhớ và chương trinh con và chương trình chính ở hai mảng nhớ khác nhau

Các thao tác thực hiện lệnh:

- Lưu CS vào ngăn xếp.
- Lưu địa chỉ offset sau CALL vào ngăn xếp.
- Nạp 2 byte đầu của ô nhớ có địa chỉ là nội dung M vào IP.
- Nạp 2 byte tiếp theo vào CS.
- Thực hiện lệnh ở CS: IP.

CALL Disp (gọi gần trực tiếp)

Lệnh này có nghĩa là địa chỉ của chương trình con trực tiếp trong lệnh và chương trình con và chương trình chính ở cùng trong một mảng nhớ.

CALL M/R (gọi gần gián tiếp)

Lệnh này có nghĩa là địa chỉ của chương trình con được địa chỉ gián tiếp qua ô nhớ hay thanh ghi và chương trình con và chương trình chính ở ở cùng trong một mảng nhớ.

Các thao tác thực hiện lệnh:

- Lưu địa chỉ sau CALL vào ngăn xếp.
- Nếu M/R là thanh ghi R thì nạp nội dung của R vào IP

  Nếu M/R là ô nhớ thì nạp nội dung của ô nhớ 2 byte vào IP.
- Thực hiện lệnh CS : IP.

Lệnh RET (trở về từ chương trình con)

Tùy theo kiểu lệnh CALL lệnh RET sẽ tương ứng.

RET (xa)

Các thao tác thực hiện lệnh:

- Lấy 2 byte từ đỉnh ngăn xếp nạp vào IP.
- Lấy 2 byte tiếp theo vào CS.
- Thực hiện lệnh ở CS : IP.

RET (gần)

Các thao tác thực hiện lệnh:

- Lấy 2 byte từ đỉnh ngăn xếp nạp vào IP.
- Thực hiện lệnh CS : IP

RET disp 16

Ngoài các thao tác như RET (xa), RET (gần) còn cộng nội dung của SP với dịch chuyển 16 bit trong lệnh (disp16) trong lệnh.

Lệnh này thường được dùng để chuyển số liệu giữa chương trình chính và chương trình con.

Lệnh JMP nhảy không điều kiện (JMP)

Lệnh JMP về phương diện mã lệnh cũng được chia làm 4 loại tương tự lệnh CALL

JMP address (Nhảy xa trực tiếp)

JMP M (Nhảy xa gián tiếp xa)

JMP disp16 (Nhảy không điều kiện gần trực tiếp gần)

JMP R/M (Nhảy không điều kiện gần gián tiếp qua thanh ghi hay ô nhớ)

So với lệnh CALL thì JMP không có quá trình lưu giữ địa chỉ để quay trở về vào ngăn xếp.

Các lệnh nhảy có điều kiện

Ví dụ: lênh JZ Addr

Nếu thoả mãn điều kiện thì thực hiện nhảy tới địa chỉ Addr. Nếu không thoả mãn điều kiện thì thực hiện lệnh tiếp theo.

Các lệnh nhảy có điều kiện chỉ thực hiện được trong vòng 126 byte.

Các lệnh nhảy có điều kiện có thể chia làm 3 nhóm:

Các lệnh nhảy trên cơ sở các cờ

JO Addr	***nếu cờ OF = 1***
JNO Addr	***OF = 0***
JC Addr	***nếu cờ CF = 1***
JNC Addr	***CF = 0***
JZ Addr	***ZF = 1***
JNZ Addr	***ZF = 0***
JS Addr	***SF = 1***
JNS Addr	***SF = 0***
JP Addr	***PF = 1***
JNP Addr	***PF = 0***
JPE Addr	***PF = 1***
JPO Addr	***PF = 0***

JCXZ Addr nhảy nếu nội dung (CX) = 0

Các lệnh nhảy trên cơ sở so sánh 2 số có dấu, kiểm tra cả SF=0F

JE Addr	=
JNE Addr	≠
JG, JNLE Addr	>
JGE, JNL Addr	≥
JL, JNGE Addr	<
JLE, JNG Addr	≤

Các lệnh nhảy trên cơ sở các phép tính không dấu

JE Addr	=
JNE Addr	≠
JA, JNBE Addr	>
JAE, JNB Addr	≥
JB, JNAE Addr	<

JBE, JNA Addr ***≤***

LOOP Addr Lặp lại một khối lệnh số lần là nội dung của thanh ghi CX.

Sau một lần lặp nội dung của CX giảm đi 1.

Lệnh ngắt INT

Cú pháp ***INT số ngắt***

Có 256 loại ngắt đánh số từ 00h đến FFh, chứa ở bảng ngắt từ địa chỉ 0 - 3FFh. Mỗi vector ngắt có 4 byte, 2 byte đầu là địa chỉ offset, 2 byte sau là địa chỉ mảng của chương trình con phục vụ ngắt.

Quá trình hoạt động của MP khi thực hiện lệnh ngắt:

Lưu thanh ghi cờ vào ngăn xếp

Xóa các cờ IF và TF

Lưu nội dung của CS vào ngăn xếp

Lưu nội dung của IP vào ngăn xếp

Nạp 2 byte thấp của vector ngắt vào IP

Nạp 2 byte cao của vector ngắt vào CS

Thực hiện lệnh ở CS : IP

Lệnh IRET

Lệnh này đặt ổ cuối chương trình con phục vụ ngắt.

Quá trình thực hiện:

Lấy 2 byte từ đỉnh ngăn xếp đưa vào IP

Lấy 2 byte tiếp theo từ đỉnh ngăn xếp đưa vào CS

Lấy 2 byte tiếp từ đỉnh ngăn xếp đưa vào thanh ghi cờ FR

Trong bảng các vector ngắt gồm có các vector ngắt của bộ vi xử lý, ngắt cứng từ điều khiển ngắt PIC 8259 và các vector ngắt mềm của BIOS và DOS.

Ví dụ một số ngắt của bộ vi xử lý

Divide by Zero

Single Step

NMI

Break point

Overflow

5.7.6. Nhóm lệnh điều khiển bộ vi xử lý

STC ***Thiết lập cờ CF***
CLC ***Xóa cờ CF***
CMC ***Đảo cờ CF***

STI *Thiết lập cờ ngắt IF*
CLI *Xóa cờ ngắt IF*

STD *Thiết lập cờ hướng DF*
CLD *Xóa cờ hướng DF*

NOP *Không làm gì chỉ tính thời gian*
HLT *Dừng MP*

WAIT

Đưa MP về trạng thái chờ.

MP ra khỏi trạng thái này do:

Ngắt ngoài

Ngắt của bộ đồng xử lý.

LOCK

Đưa tín hiệu Lock về trạng thái tích cực và giữ ở trạng thái này trong quá trình của lệnh tiếp theo.

Chúng ta đã tìm hiểu các lệnh cơ bản của họ vi xử lý 80x86. Trong họ vi xử lý này, các bộ vi xử lý ra đời sau thường giữ nguyên tập lệnh của bộ vi xử lý trước và bổ sung một số lệnh mới.

5.8. DEBUG

DEBUG là một chương trình trợ giúp cho hiệu chỉnh chương trình. Nó cung cấp các điều kiện để kiểm tra chương trình ở dạng mà máy và các chương trình thực hiện được. DEBUG là công cụ cho phép tìm hiểu kiến trúc lệnh của máy vi tính.

5.8.1. Khởi động chương trình DEBUG

Các cách khởi động DEBUG

Có 2 cách khởi động DEBUG

Khởi động không có đối

> Debug (CR)

Chương tr7ình DEBUG được nạp vào bộ nhớ

Khi khởi động DEBUG không có đối thì

Các thanh ghi mảng CS, DS, SS, ES thiết lập ở đáy vùng nhớ còn tự do.

(IP) = 0100h

(SP) = FFEE

(AX) , (BX), (CX), (DX), (BP), (SI), (DI) = 0

Các cờ bị xóa

Khởi động có đối

>Debug filename (CR)

DEBUG được nạp vào vòng nhớ

Tệp có tên filename cũng được nạp vào vòng nhớ

Khi khởi động DBUG có đối

Các thanh ghi CS, DS, SS, ES, IP, SP nạp giá trị theo tệp filename

CX chứa độ dài của tệp nhỏ hơn 64kB

Nếu tệp dài hơn 64kB- Độ dài của tệp ở trong đôi thanh ghi DX: CX

Một số quy ước

Địa chỉ :

Address Ký hiệu địa chỉ

Có 3 dạng

CS : 0100 (DS, SS, ES) thanh ghi mảng : offset

1D5S : 0300 địa chỉ mảng : offset

Offset, thanh ghi mảng là ngầm định

Phạm vi vùng nhớ

Rangeký hiệu một vùng nhớ

Có 2 dạng

Address L value

Ví dụ DS : 0100 L 10

Address 1, address2

Ví dụ DS : 0100 010A

Một số tính chất chung của lệnh

Lệnh gồm 1 chữ in hay thường.

Lệnh và các tham số có thể cách nhau bằng dấu cách.

Hủy lệnh bằng Ctrl-break.

Bắt đầu thực hiện lệnh bằng CR.

Dừng tạm màn hình bằng Ctrl-numlock, để tiếp tục ấn phím bất kỳ.

5.8.2. Các lệnh của DEBUG

a. Lệnh hiển thị một vùng nhớ (DUMP)

Chức năng (CN): Hiển thị nội dung một vùng nhớ

Cú pháp (CP): D range

D address

Ví dụ: D 7000 : 0100 04FF

D 7000 : 0100 L400

Ví dụ: D100

Lênh trên sẽ hiển thị từ địa chỉ từ 100 cho đến hết 128 byte tiếp theo .

D

Sẽ hiển thị 128 byte tiếp theo kể từ địa chỉ hiện tại.

Ghi chú:

Thanh ghi mảng ngầm định là DS

Trên màn hình sẽ hiển thị theo 3 trường

1, địa chỉ

2, giá trị HEX của 16 byte

3, giá trị mã ASCII của 16 byte

(Nếu là mã ASCII điều khiển thì hiện dấu .).

b. Lệnh đưa số liệu vào ô nhớ (ENTER)

CN1:

Thay đổi nội dung của một hay nhiều byte bắt đầu từ một địa chỉ.

CP1: E address list

Ví dụ E DS : 100 F3 ‘xyz’ 8D

CN2: Hiển thị và thay đổi từng byte bắt đầu từ một địa chỉ.

CP2: E address

Tóm lại

Vào giá trị mới nếu muốn thay đổi giá trị cũ.

Ấn dấu cách nếu muốn giữ giá trị cũ và hiển thị byte tiếp theo.

Ấn dấu - để trở lại địa chỉ trước.

Ấn (CR) để kết thúc lệnh.

c. Lệnh lưu vào vùng nhớ một giá trị nhất định (FILL)

CN1: Lưu vùng nhớ có giá trị trong danh sách.

CP1: F range list

Ví dụ: F 1D03:100 L50 FF FE DD

thì các ô nhớ 1DOS . 100 đến 104 sẽ chứa các giá trị

FF FE DD FF FE

Ví dụ: F1D03: 100 4FFF 55

Ghi chú:

Nếu list ngắn hơn độ dài vùng nhớ thì list sẽ được lặp lại tới khi hết vùng nhớ.

Nếu list dài hơn vùng nhớ thì những byte dài hơn sẽ bị bỏ không được nạp.

Thanh ghi mảng ngầm định là DS.

d. Lệnh hiển thị và sửa đổi giá trị các thanh ghi (REGISTER)

CN: Hiển thị nội dung các thanh ghi và các cờ

Sửa đổi nội dung các thanh ghi và các cờ.

CP: R [Registor/F]

Có 3 dạng

R

Lệnh này hiển thị nội dung tất cả các thanh ghi

RAX

Lệnh này hiển thị nội dung một thanh ghi cụ thể (Ví dụ AX)

AX 14FF

: _ nếu thay nội dung của AX (Ví dụ bằng 55AA thì phải nhập vào)

nếu không thay đổi thì ấn CR để kết thúc lệnh.

RF

Lệnh này hiển thị giá trị các cờ ở dạng chữ:

NV UP DI NG NZ AC PE NC

Nếu muốn thay đổi một số cờ thì vào liên tiếp các cờ đó rồi ấn CR.

Bảng các ký hiệu giá trị logic của các cờ như sau:

Cờ	(1)	(0)	
OF	OV	NV	Cờ tràn
DF	DN	UP	Cờ hướng
IF	EI	DI	Cờ ngắt
SF	NG	PL	Cờ dấu
ZF	ZR	NZ	Cờ zero
AF	AC	NA	Cờ nhớ phụ
PF	PE	PO	Cờ KT chẵn lẻ
CF	CY	NC	Cờ nhớ

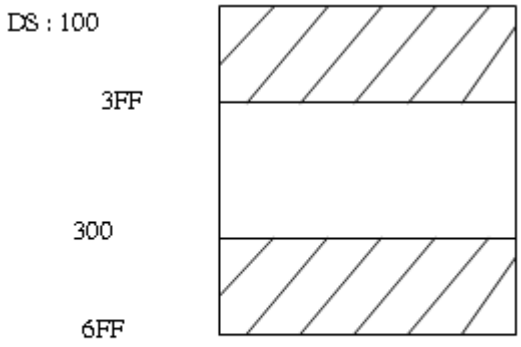
e. Lệnh so sánh các vùng nhớ (COMPARE)

CN: So sánh từng byte nội dung 2 vùng nhớ

CP: C range, address

Ví dụ: C100 4FF 300

C100 L 400 300



Hình 5.21: Lệnh COMPARE

Nếu phát hiện các byte khác nhau thì trên mà hình sẽ thông báo.

địa chỉ 1 byte 1 byte2 địa chỉ 2

f. Đọc giá trị từ một cổng vào/ra

CN: Đọc và hiển thị giá trị một byte từ cửa

CP: I address địa chỉ cửa

Ví dụ I 2F8

Trên màn hình sẽ hiển thị giá trị đọc vào từ cổng có địa chỉ là 2F8h, ví dụ 68

g. Đưa một byte số liệu ra cổng vào/ra

CN: Đưa 1 byte số liệu ra cổng

CP: O address value

Ví dụ O 2F8 41

(Đưa số 41h ra cổng có địa chỉ là 2F8h)

h. Chuyển số liệu (MOVE)

CN: Chuyển vùng nhớ xác định bởi giới hạn range đến vùng nhớ khác có địa chỉ address.

CP: M range address

Nếu trong range và address không xác định segment thì ngầm định là DS.

Ví dụ

- M CS : 100 110 CS : 500
- M CS : 100 L11 CS : 500

Nếu

- M CS : 100 110 500

thì 17 bytes từ địa chỉ CS : 100 sẽ chuyển đến DS : 500

i. Lệnh tìm kiếm (SEARCH)

CN : Tìm trong vùng nhớ xác định bởi range có ký tự trong list.

CP: S range, list

Nếu trong range không xác định segment thì ngầm định là DS.

Ví dụ: S CS: 100 110 41

S CS: 110 L11 41

Tìm ô nhớ có giá trị 41h trong vùng nhớ CS :100 đến CS : 10. Giả thiết trong vùng nhớ trên có 2 ô nhớ chứa giá trị 41h thì trên màn hình sẽ xuất hiện

1A44 : 0104

1A44 : 010D

k. Lệnh cộng trừ 2 số HEX

CN: Cộng và trừ 2 giá trị HEX và hiển thị kết quả lên màn hình.

CP : H value 1 value2

Ví dụ H1F8

Trên màn hình hiển thị kết quả:

27 17

(tổng) (hiệu)

l. Đặt tên tệp (Name)

CN: Đặt tên tệp sử dụng trong lệnh Nạp (Load) và Write.

CP: N Tentêp

Ví dụ: N Thu.com

m. Nạp từ đĩa vào vùng nhớ (Load)

CN: Nạp 1 tệp hay các cung số liệu trên đĩa vào vùng nhớ.

CP có hai dạng:

L [address số ổ đĩa cung đầu số lượng cung]

Ví dụ: L 1A44:100 1 0F 0D

ổ đĩa A=0, B=1, C=2

N Tep1

L

Tệp có tên là Tep1 sẽ được nạp vào vùng nhớ bắt đầu từ địa chỉ CS:100

Ví dụ: N Tep1

L 300

Tệp có tên là Tep1sẽ được nạp vào vùng nhớ bắt đầu từ địa chỉ CS:300

Chú ý: Nếu khởi động

DEBUG A:\Tep1.exe

thì chương trình Tep1.exe sẽ được đưa vào bộ nhớ địa chỉ bắt đầu CS:100.

n. Lệnh ghi đĩa (Write)

CN: ghi lên đĩa một vùng số liệu từ bộ nhớ

CP Có 2 dạng:

Cú pháp: W [address ổ đĩa cung đầu số cung)]

Ví dụ: W 200 1 1F 20

Ghi nội dung vùng nhớ có địa chỉ offset là 200h vào ổ đĩa A tại cung đầu tiên có số là 1Fh và số cung là 20h.

Ghi vào tệp

Ví dụ:

N Tep2

RCX

CX = 000F

: 3FF

W 200

Ghi vào tệp Tep2, 1kB bắt đầu từ vùng nhớ có địa chỉ đầu 200h.

o. Lệnh nhập chương trình hợp ngữ A (Absolut assembler)

CN: Soạn thảo và dịch trực tiếp các lệnh bằng hợp ngữ**.**

CP: A [address]

Ví dụ: A 200

thì trên màn hình

yyyy: xxxx MOV DX, 300 (CR)

yyyy: xxxx MOV AH, 9 (CR)

yyyy: xxxx ..... (CR)

Nếu phát hiện một lỗi trong lệnh debug đưa ra thông báo ERROR và hiện lại địa chỉ cuối cùng.

Khi soạn và dịch xong toàn bộ CT thì ấn (CR) để quay về DEBUG.

Ghi chú: Trong chương trình có thể sử dụng tất cả các lệnh cơ bản của bộ vi xử lý họ 80x86 , ngoài ra còn có một số lệnh giả (Pseudo Instruction).

p. Lệnh thực hiện chương trình (GO)

CN: Thực hiện 1 hoặc nhiều lệnh bắt đầu ở CS : IP hay xác định bởi địa chỉ sau dấu =.

CP: G [= address [address...]

q. Lệnh theo dõi thực hiện lệnh (TRACE)

CN: Thực hiện một hay một nhóm lệnh và hiển thị nội dung các thanh ghi, các cờ và lệnh sẽ thực hiện tiếp theo.

CP: T [= address] [,value]

(Địa chỉ bắt đầu , số lệnh cần thực hiện)

r. Lệnh theo dõi thực hiện 1 lệnh - P

CN: Tương tự lệnh T nhưng nhảy qua tất cả các lệnh kể cả lệnh CALL, và INT Như vậy bỏ qua việc theo dõi chi tiết từng lệnh trong chương trình con.

CP: P [= address] [,value]

s. Lệnh dịch ngược từ mã máy ra hợp ngữ U (Unassembler)

CN : Dịch ngược các lệnh dưới dạng mã máy ở trong vùng nhớ sang dạng hợp ngữ và hiển thị địa chỉ, mã máy và mã lệnh hợp ngữ.

CP: U [range][,address]

t. Lệnh ra khỏi debug (Quit)

Q

5.9. Các bộ vi xử lý tiên tiến

5.9.1. Các đặc điểm về kiến trúc của các bộ vi xử lý tiên tiến

a. Bộ đệm Cache

Bộ nhớ đệm Cache chứa mảng lệnh và số liệu được sử dụng trong thời gian gần nhất, được điều khiển bằng phần cứng và chương trình. Bộ nhớ cache đặt giữa CPU và bộ nhớ chính. Bộ nhớ cache chứa một phần bản sao của bộ nhớ chính. Khi CPU thâm nhập vào dữ liệu nó đưa địa chỉ tới bộ điều khiển Cache, sau đó một trong hai quá trình sẽ xảy ra.

- Trúng (cache hit): nếu địa chỉ tìm thấy trong Cache

- Trượt (cache miss): nếu địa chỉ không có trong Cache

Khi trượt một khối nhớ từ bộ nhớ chính sẽ được đưa vào thay thế cho một đường (khối) của Cache. Đường nào sẽ được chọn để thay dựa trên hai nguyên lý sau:

- Cục bộ theo thời gian: nếu CPU thâm nhập vào một ô nhớ thì có xác suất cao nó sẽ thâm nhập ô nhớ đó trong tương lai.

- Cục bộ theo không gian: nếu CPU thâm nhập vào một ô nhớ thì có xác suất cao nó sẽ thâm nhập các lệnh và dữ liệu đặt sát các vị trí đó trong tương lai.

Trường hợp ghi vào Cache dữ liệu sẽ được ghi vào bộ nhớ chính, ta phân biệt hai trường hợp sau:

- Khi ghi vào Cache thì đồng thời ghi vào bộ nhớ chính, phương pháp này gọi là ghi xuyên (Write through)

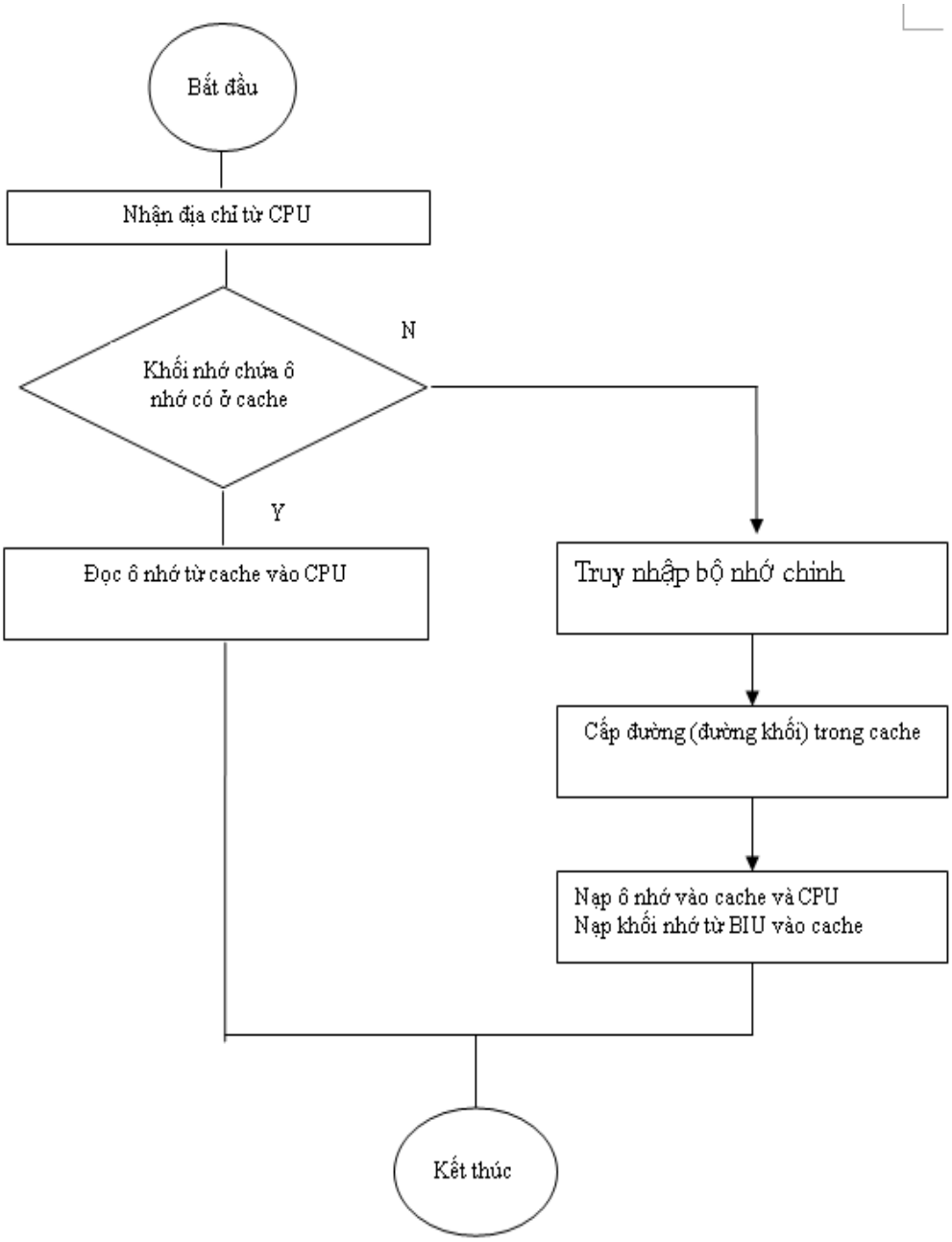
- Khi ghi chỉ ghi vào bộ nhớ Cache, dữ liệu từ Cache sẽ được chuyển vào bộ nhớ chính tại một thời điểm thích hợp sau đó (ví dụ khi chuyển dữ liệu từ bộ nhớ chính ra thiết bị ngoại vi).

Việc ánh xạ giữa bộ nhớ Cache và bộ nhớ chính có thể tổ chức theo phương pháp khác nhau:

- Cache ánh xạ trực tiếp (Direct mapping cache)
- Cache ánh xạ liên kết toàn phần (Full associative mapping cache)
- Cache ánh xạ liên kết cụm (Set associative mapping cache)

Nội dung về bộ nhớ Cache sẽ được nghiên cứu kỹ hơn trong cấu trúc máy II

Hình 5.22: Ví dụ về CPU đọc một ô nhớ.



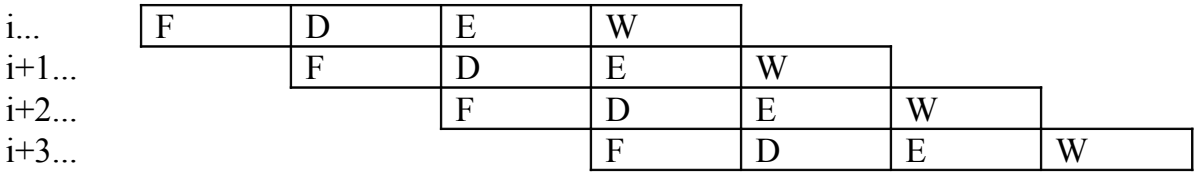
b. Đường ống

Đường ống k bước: chia một thao tác thành k thao tác cơ sở và thực hiện mỗi thao tác cơ sở trong một bước, bước này sau bước kia.

Tại bất cứ thời điểm nào đường ống k bước cũng xử lý k tập dữ liệu đồng thời.

Có 2 loại đường ống chính trong máy tính điện tử: đường ống lệnh và đường ống số học.

Ví dụ đường ống lệnh



c. Công nghệ RISC (Reduced Instruction Set Computer)

Các tính chất cơ bản của RISC:

Lệnh thực hiện trong một chu kỳ

Độ dài lệnh (1 từ) bằng độ rộng của BUS số liệu

Số lệnh ít

Số dạng lệnh khác nhau (không quá 4 dạng lệnh)

Số chế độ địa chỉ hóa không quá 4

Chỉ truy nhập bộ nhớ qua lệnh LOAD (nạp) và STORE (lưu)

Tất cả các lệnh là từ thanh ghi đến thanh ghi

Tập thanh ghi đa năng lớn

Điều khiển logic cứng

Hỗ trợ HLL (High Level Language)

5.9.2. Cấu trúc tổng quát của bộ vi xử lý tiên tiến

Các kí hiệu viết tắt

BIU	Bus Interface – Ghép nối bus
PFIU	Prefetch instruction Unit - Nhận lệnh trước vào hàng đợi
MMU	Memory Menagement Unit - Đơn vị quản lý bộ nhớ
ID	Instruction decoder - Bộ giải mã lệnh
Icache	Instruction Cache - Bộ đệm lệnh
Dcache	Data cache -Bộ đệm số liệu
BTCache	Branch Target Cache - Bộ đệm địa chỉ đích của các lệnh rẽ nhánh
CU	Control Unit - Đơn vị điều khiển
FPU	Floating Point Unit – Đơn vị xử lý số dấu phẩy động
FRF	Float Register File - Tập các thanh ghi của đơn vị xử lý số dấu phẩy động
IU	Integer Unit - Đơn vị xử lý số nguyên

IRF Integer Register File - Tập các thanh ghi của đơn vị xử lý số nguyên

SFU Special Function Unit – Đơn vị chức năng riêng

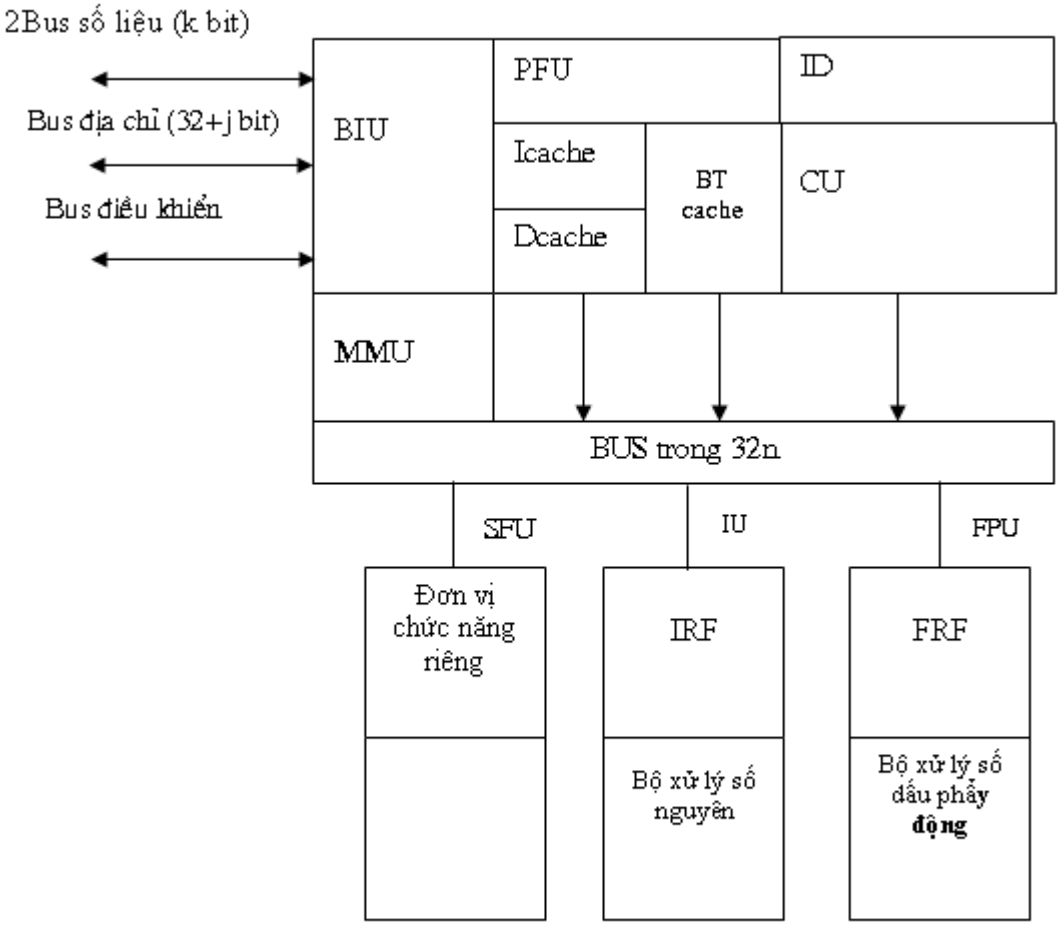
Ví dụ:

Đơn vị xử lý đồ họa

Đơn vị xử lý tín hiệu số

Đơn vị xử lý ảnh

Đơn vị xử lý vector và ma trận



Hình 5.23: Cấu trúc tổng quát của bộ vi xử lý tiên tiến

Cấu trúc tổng quát của bộ vi xử lý tiên tiến

Chúng ta mô tả cấu trúc tổng quát các bộ vi xử lý được sản xuất vào giữa những năm 1990. Cấu trúc này không biểu diễn một bộ vi xử lý cụ thể nào. Tuy vậy nó bao gồm các tính chất cơ bản của các bộ vi xử lý của giai đoạn này. Phần lớn các bộ vi xử lý giai đoạn này là 32 bit, các hệ thống 64 bit cũng đang tăng dần.

Trong hệ thống 32 bit, IU và các thanh ghi trong IRF là 32 bit, các Bus số liệu (Bus s/l) có thể là 32, 2x32, (3x32) hay 4x32.

Trong hệ thống 64 bit, IU và các thanh ghi trong IRF là 64bit, các Bus số liệu là 64 bit nhưng cũng có thể là 2x64 hay 4x64 bit.

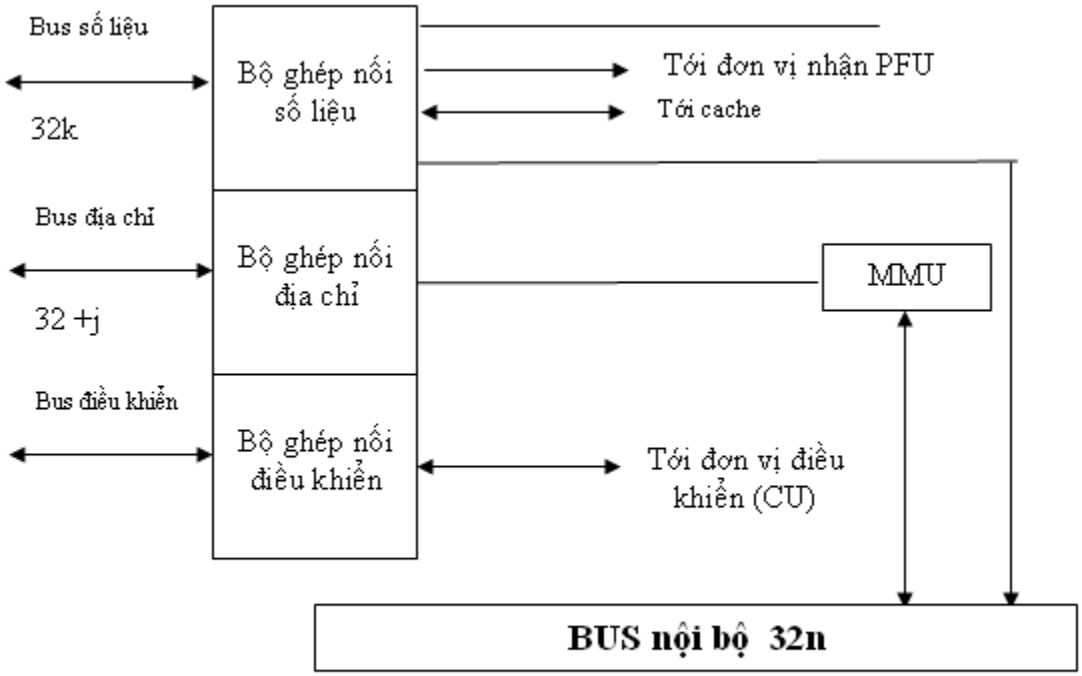
Cấu trúc tổng quát của một bộ vi xử lý tiên tiến bao gồm những bộ phận sau:

a. Hệ thống các đơn vị giao diện Bus (BIU)

BIU là hệ thống các bộ đệm giữa các đơn vị bên trong VXL và hệ thống bên ngoài, được nối với Bus hệ thống.

BIU có các đặc điểm sau:

BIU có 3 phần chính ghép nối BUS số liệu, BUS địa chỉ, Bus điều khiển



Hình 5.24: Các bộ ghép nối BUS

Bộ ghép nối số liệu kết nối Bus số liệu hệ thống và các đơn vị bên trong bộ VXL, số liệu được truyền giữa bộ ghép nối số liệu và các đơn vị khác của bộ VXL thông qua BUS nội bộ.

Có thể có một số kết nối trực tiếp từ bộ ghép nối số liệu với đơn vị nhận lệnh và hàng đợi lệnh hoặc với bộ nhớ cache.

Bộ ghép địa chỉ gửi các địa chỉ của lệnh hay của số liệu (toán hạng) tới Bus địa chỉ hệ thống

Địa chỉ được tạo ra bởi MMU.

Bộ ghép nối điều khiển gửi và nhận một số tín hiệu điều khiển và trạng thái từ bộ xử lý ra hệ thống bên ngoài và ngược lại từ thiết bị bên ngoài tới bộ xử lý. Phần lớn các tín hiệu của bộ ghép nối điều khiển được nối với CU, một số tín hiệu nối trực tiếp với các đơn vị khác .

b. Đơn vị điều khiển (CU)

Đơn vị điều khiển CU được thiết kế bằng mạch logic cứng hay bằng vi chương trình. Trong các bộ VXL truyền thống sử dụng công nghệ CISC, CU thường được xây dựng từ vi chương trình. Xu hướng gần đây trong công nghệ RISC, CU được thiết kế bằng logic cứng.

Đơn vị nhận lệnh bao gồm mạch logic lấy lệnh từ cache, rồi nối với hàng đợi lệnh FIFO, hàng đợi lệnh từ 8 đến 32 bytes. Các lệnh được lined-up sẽ được truyền đến đơn vị giải mã lệnh. Đơn vị giải mã giải mã lệnh và truyền các tín hiệu điều khiển trực tiếp tới CU - Phần lớn các Bộ VXL hiện nay là superscalar, có nghĩa là có nhiều hơn 1 lệnh tại một thời điểm được chuyển tiếp để giải mã.

Một số bộ VXL có SFU bổ sung cho IU và FPU. SFU có thể là

Đơn vị xử lý đồ họa

Đơn vị xử lý tín hiệu số

Đơn vị xử lý ảnh

Bộ xử lý vector và ma trận.

Bộ VXL như pentium (loại thế hệ đầu) chứa nhiều hơn 3 triệu transistor, Năm 2000, con số này có thể vượt 50 triệu. Một bộ VXL có thể có nhiều SFU.

c. Bộ đệm Cache

Cache là bộ nhớ truy nhập nhanh đặt giữa CPU và bộ nhớ chính. Cache với kích thước thích hợp có thể cải thiện đáng kể hiệu năng toàn bộ vì nó cho phép CPU thâm nhập thông tin nhanh hơn nhiều từ bộ nhớ chính. Các bộ VXL cải tiến có tới trên 32kB cache. Các bộ VXL hiện đại có đường ống (pipeline), có nghĩa là với đường ống của lệnh n giai đoạn, bộ xử lý các giai đoạn khác nhau của n lệnh đồng thời . Như vậy, trong khi CPU đang thâm nhập bộ nhớ để đọc lệnh, nó có thể thâm nhập bộ nhớ để đọc phần tử dữ liệu cho lệnh khác. Để phục vụ cho mục đích này cần có 2 cache riêng: Icache và Dcache

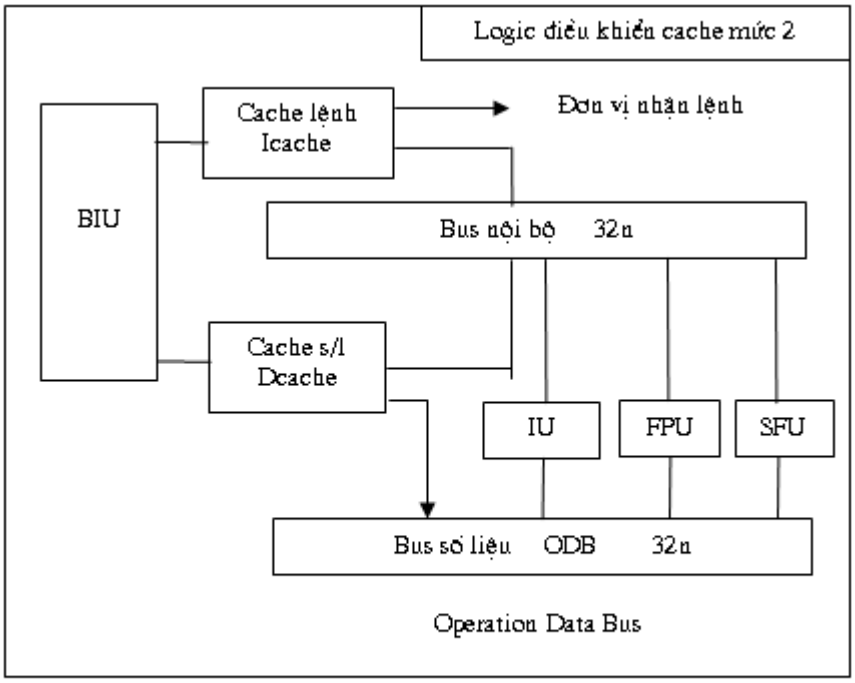
Dcache nhận thông tin từ bộ nhớ chính qua Bus số liệu và BIU từng khối trong các bộ vi xử lý hiện đại khối 16 hay 32 bytes.

Icache lệnh nối trực tiếp với đơn vị nhận lệnh và truyền 1 hay nhiều lệnh trong 1 nhịp.

Cả 2 cache đều nối với bus nội bộ

Trong một số bộ vi xử lý, Dcache còn nối với đơn vị xử lý qua ODB (Operation Data Bus). ODB đủ rộng để mang một vài toán hạng cùng một lúc (128 đến 156 bits).

Nhiều bộ VXL hiện đại thiết kế để làm việc với cache mức 2 (trong hay ngoài chip). Cache mức 2 đặt giữa cache mức 1 và bộ nhớ chính. Cache mức 2 có dung lượng lớn hơn. Nhiều bộ VXL hiện đại có BTC (branche target cache). Trong hệ thống có đường ống, khi gặp lệnh rẽ nhánh thì lệnh sau lệnh rẽ nhánh sẽ được đưa ra khỏi đường ống và thay vào đó là lệnh đích (lệnh đầu tiên của nhánh phải thực hiện. Nếu lệnh đích lấy từ bộ nhớ chính sẽ rất chậm; cho nên lệnh đích đầu tiên sẽ chứa ở BTC và lấy ra rồi đưa vào đường ống nhanh hơn. Rõ ràng sự có mặt của BTC hướng tới cải thiện toàn bộ hiệu năng của bộ VXL. Có một số hệ thống chỉ chứa địa chỉ đích, không chứa lệnh đích.

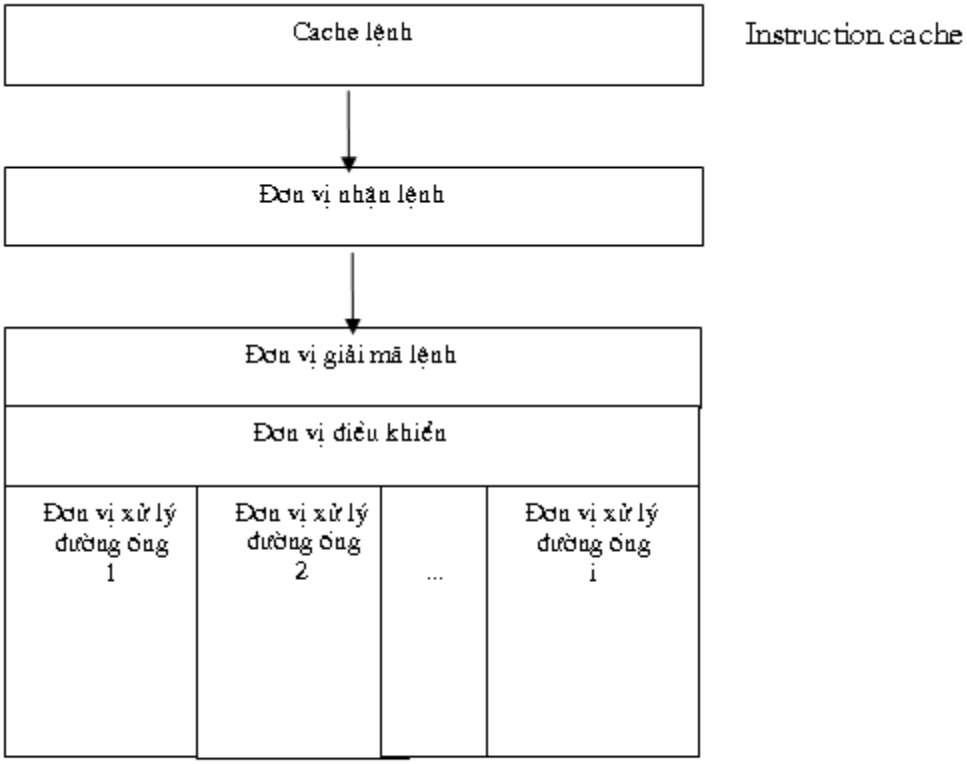


Hình 5.2.5: Các bộ nhớ Cache trong VXL

d. Xử lý song song ở mức lệnh

Các bộ VXL hiện đại nhất đã có xử lý song song ở mức lệnh. Trong nhiều trường hợp superscalar được sử dụng. Trong hệ thống superscalar đơn vị nhận lệnh sẽ đưa ra i lệnh đồng thời và chuyển tới bộ giải mã. Sau đó đơn vị điều khiển phát sinh các lệnh kích hoạt một số các đơn vị thao tác đường ống. Số đường ống có thể

khác số lệnh được đưa ra, tuy nhiên khả năng xử lý được đồng thời các lệnh đưa ra sẽ tốt hơn.



Hình 5.26: Xử lý song song trong VXL

e. Đơn vị số nguyên IU

Đơn vị số nguyên IU có những đặc điểm sau:

Có nhiều đơn vị xử lý giống nhau làm việc song song trong hệ thống superscalar.

Có nhiều đơn vị cộng/ trừ , nhân/chia trong IU.

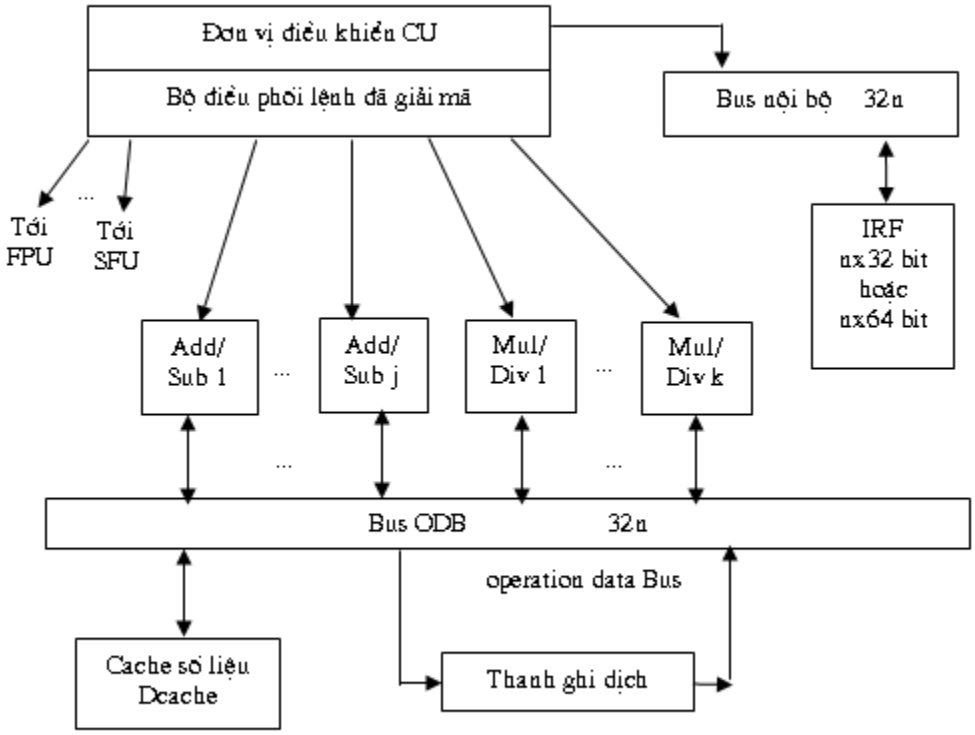
Có tập các thanh ghi IRF, 32 hay 64 bit.

Hệ thống công nghệ CISC thường có 8 đến 16 thanh ghi.

Hệ thống RISC có ít nhất 32 thanh ghi, một số hệ có tới trên 100 thanh ghi.

Bộ điều phối nhận chỉ thị được giải mã từ đơn vị điều khiển chuyển các chỉ thị này đến các đơn vị xử lý tương ứng. Các lệnh số nguyên tới IU, dấu phảy động tới FPU, các lệnh chức năng tới SFU- số liệu được chuyển đến các đơn vị xử lý từ Dcache qua ODB.0

Phần lớn các hệ thống có các thanh ghi dịch cho phép thực hiện một cách hiệu quả các lệnh dịch nhiều bit trong 1 nhịp



Hình 5.27 : Đơn vị số nguyên IU

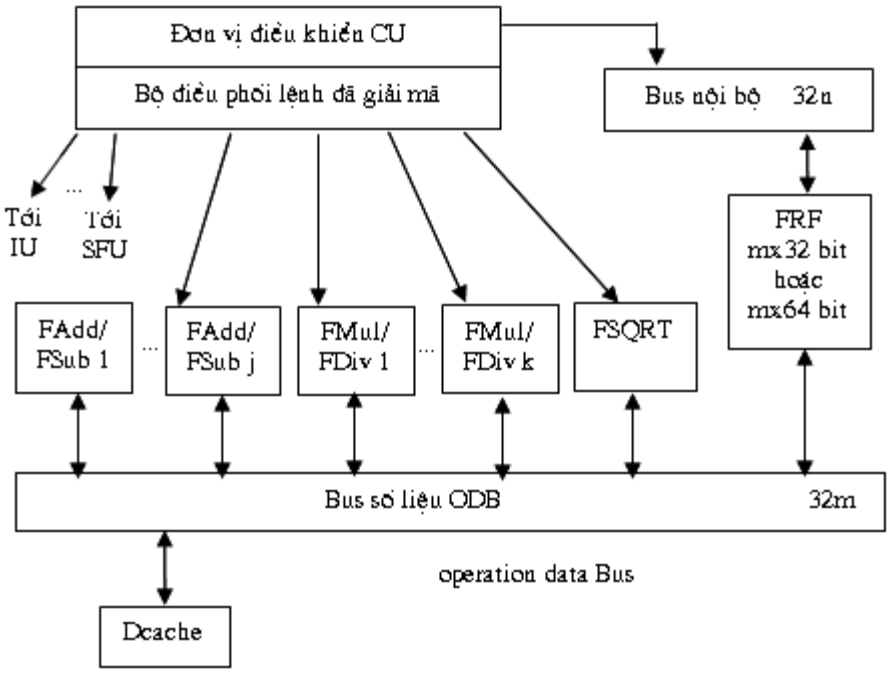
f. Đơn vị FPU

Các dòng thông tin cũng tương tự như IU.

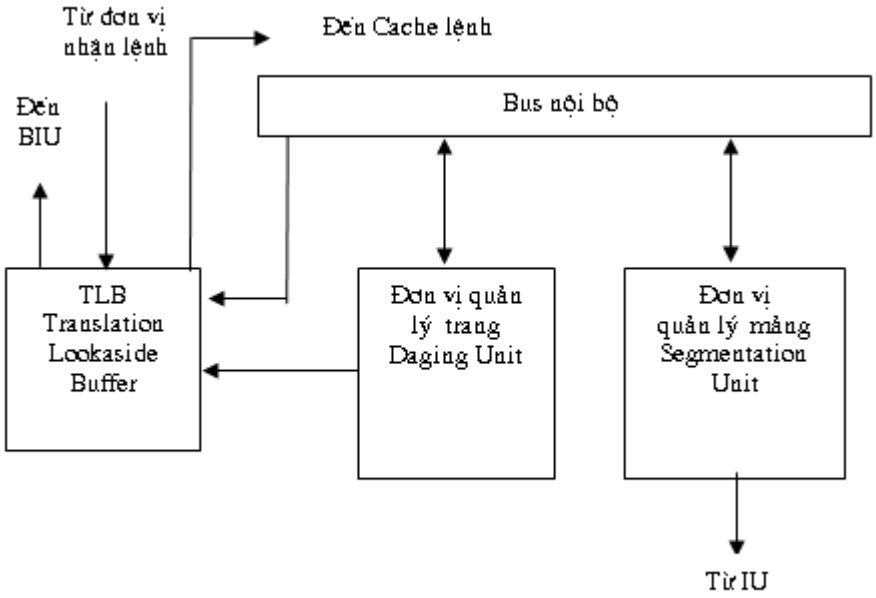
Có các tập thanh ghi riêng FRF, nếu CISC thì 8 nếu RISC thì 32

Hầu hết các thiết kế hiện đại dùng chuẩn IEEE 754 - 1985 cho số dầu phảy động

+ 32 bit độ chính xác đơn
+ 64 bit độ chính xác kép
+ 80 bit số dấu phảy mở rộng
+128 bit - SIMD



g. Đơn vị điều khiển (quản lý) bộ nhớ MMU



Hình 5.29: Đơn vị quản lý bộ nhớ MMU

Các chức năng chính của đơn vị quản lý bộ nhớ MMU:

Chuyển địa chỉ logic thành địa chỉ vật lý (thực) và gửi địa vật lý tới cache hay qua BIU và BUS đ/c tới thiết bị ngoài.

Cung cấp cơ chế phân trang trong cách tổ chức bộ nhớ ảo - (paging Unit).

Cung cấp cơ chế phân mảng bởi Segmentation Unit.

Cung cấp cơ chế bảo vệ bộ nhớ. Thường được thực hiện bên trong đơn vị quản lý trang hay mảng.

Bao gồm và quản lý TLB (Translation Lookaside Buffer) hay ATC (Address Translation Cache) cho việc chuyển đổi trang ảo sang trang vật lý.

Hầu hết các bộ VXL cải tiến đều có cơ chế phân trang và TLB (hay ATC).

Hầu hết các bộ VXL hiện đại có:

Bus số liệu dồn kênh

Bus số liệu và Bus địa chỉ riêng

Bộ ghép nối với Bus hệ thống với cache mức 2 riêng

Có các tín hiệu điều khiển và trạng thái phong phú

Có nhiều tín hiệu nguồn và đất như vậy đường nối nguồn và đất ngắn hơn làm đơn giản hóa việc thiết kế mạch và giá thành.

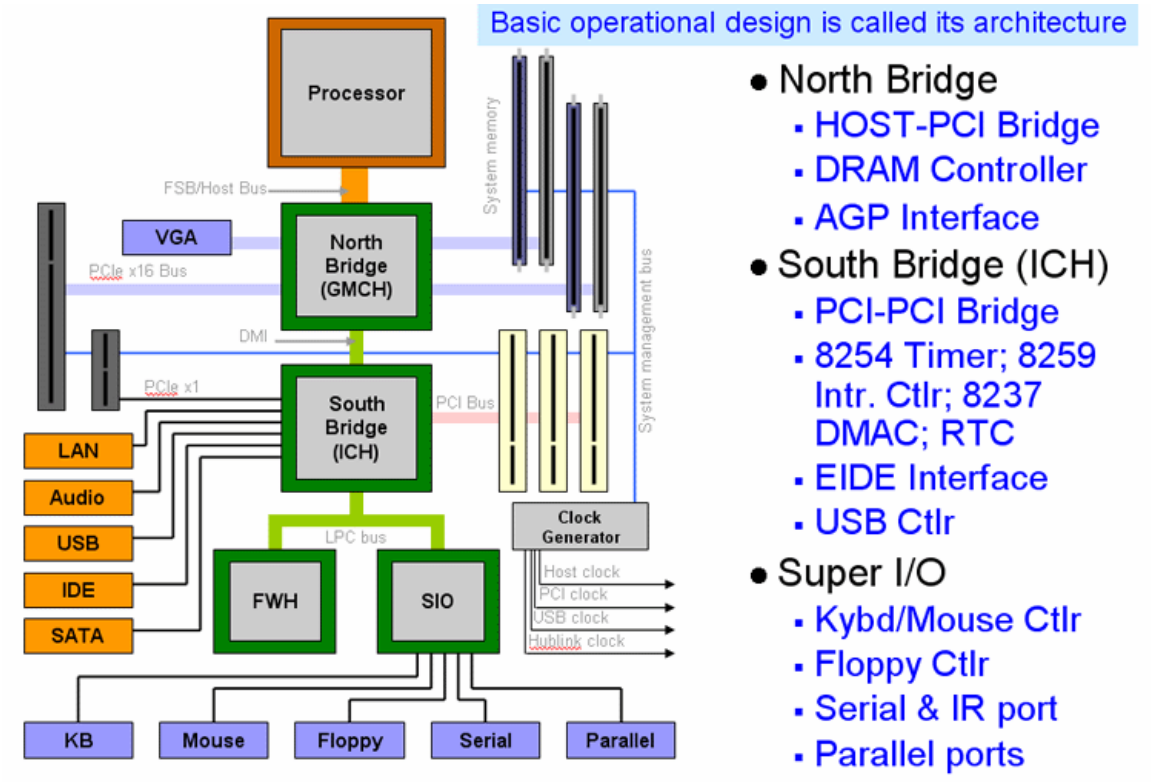
CHƯƠNG VI

GIỚI THIỆU VỀ MÁY VI TÍNH

Trong phạm vi nội dung môn học "Cấu trúc máy I", chương VI chỉ giới thiệu sơ đồ khối chung nhất về máy vi tính. Cấu trúc chi tiết của máy vi tính sẽ là nội dung của môn học "Cấu trúc máy II". Nội dung chương VI sẽ được cập nhật, thay đổi theo thời gian cho kịp với sự phát triển nhanh chóng của công nghệ thông tin nói chung và kỹ thuật máy tính nói riêng.

6.1. Sơ đồ khối của máy vi tính PC

Trên cơ sở tìm hiểu sơ đồ khối MainBoard của hãng Intel sẽ cho phép chúng ta dễ dàng tìm hiểu sang các họ vi tính khác.



Hình 6.1 Sơ dồ khối MainBoard của hãng Intel

Bộ vi xử lý (Procesor) trong máy vi tính này đã được giới thiệu trong mục 5.1 của Chương 5. Cầu phía bắc (North Bridge) kết nối bộ xử lý với BUS PCI (Peri pheral Component Interconnect), điều khiển bộ nhớ và bộ ghép nối điều khiển màn hình. Cầu phía nam (South Bridge) bao gồm các bộ đếm thời gian 8254, các bộ điều

khiển ngắt PIC 8259, các bộ điều khiển thâm nhập bộ nhớ trực tiếp DMAC 8237, các BUS ghép nối EIDE và BUS USB…Bộ điều khiển vào/ ra tích hợp (Super I/O) bao gồm các bộ điều khiển và ghép nối các thiết bị ngoại vi chậm như máy in, bàn phím, ổ đĩa mềm, cổng ghép nối song song và nối tiếp…Chúng ta giới thiệu một số khối cơ bản theo chức năng của máy vi tính này.

1. ROM – BIOS

Linh kiện được sử dung là mạch EEPROM hay Flash chứa một số chương trình và chương trình con như:

Chương trình kiểm tra hệ thống máy tính sau khi bật nguồn điện POST (Power On Self Test). Nếu là "plug and play" thì giai đoạn này truy nhập tham số trên card.PCI

Chương trình BIOS setup để người sử dụng thiết đặt tham số cấu hình của các thiết bị của máy vi tính hay thời gian.

Chương trình mồi (Boot) để khởi động hệ thống máy tính từ đĩa cứng hoặc CD-ROM hoặc đĩa mềm.

Các chương trình con chứa trong ROM-BIOS, cung cấp các hàm phục vụ giao diện phần cứng.

2. RAM

Trong họ máy vi tính này các mạch nhớ DRAM được tổ chức thành các Modul:

SIMM (Single In-line Memory Modul)

DIMM (Double In-line Memory Modul)

RIMM (Rambus in-line memory modul)

Các mạch DRAM được sử dụng là:

- SDRAM (Synchronous DRAM)

DDR (Double Data Rate) - Dữ liệu được truyền dữ liệu cả ở sườn lên và sườn xuống của tín hiệu nhịp (Ví dụ: 133 SDRAM thành 266 DDR

DDR2 tốc độ 266 MHz hay 533 MHz.

Bố trí địa chỉ bộ nhớ:

Địa chỉ	Kích thước	Chức năng
00000000-0009FFFF	640KB	Bộ nhớ RAM quy ước
000A0000-000BFFFF	128KB	RAM hiển thị
000C0000-000DFFFF	128KB	Bộ nhớ ROM mở rộng
000E0000-000FFFFF	128KB	ROM-BIOS
00100000-FFFDFFFF	4094.875	Không gian bộ nhớ RAM
FFFE0000-FFFFFFFF	128KB	ROM-BIOS

3. CMOS (Complement Metal Oxyde Semiconductor)

Trong máy vi tính truyền thống:

- Dung lượng là 64 byte: 14 byte chứa thời gian và 50 byte chứa thông số cấu hình

Nguồn nuôi bằng pin riêng

Truy nhập nôi dung của CMOS thông qua 2 cổng vào/ ra (Cổng chứa địa chỉ của từng byte có địa chỉ I/O là 70h và cổng chứa số liệu để ghi đọc vào ô nhớ có địa chỉ là 71h)

Đối với các máy vi tính mới, CMOS chứa tới 256 byte.

4. Bộ điều khiển ngắt PIC (Programmable Interrupt Controller)

Máy vi tính PC-XT có 1 mạch PIC
Máy vi tính PC-AT có 2 mạch PIC
Máy vi tính PC-AT mới có APIC (Advanced PIC)

Các mạch điều khiển ngắt này cung cấp các đầu vào ngắt cứng (IRQ) cho máy vi tính (Mục 6.2).

5. Bộ điều khiển thâm nhập bộ nhớ trực tiếp DMAC (Direct Memory Access Controller)

Máy vi tính PC-XT có 1 mạch DMAC
Máy vi tính PC-AT có 2 mạch DMAC
Máy vi tính PC-AT mới có thể sử dụng DMA qua PCI..

6. Bộ đếm thời gian lập trình được (Programmable Interval Timer- PIT)

Có 3 bộ đếm 16 bit với các chức năng:

- CT0: Đồng hồ hệ thống
  Đầu vào 1.190MHz
  Đầu ra IRQ_o (cứ 55ms có 1 xung, 18.2 lần/giây)
- CT1: Sinh tín hiệu để tạo chu kỳ làm tươi cho DRAM
- CT2: Điều khiển âm thanh ra loa
  Đầu vào 1.190MHz
  Tần số thay đổi do lập trình.

7. Ghép nối ổ đĩa cứng

Trong máy vi tính sử dụng hai chuẩn ghép nối chính:

IDE (Integrated Drive Electronics)

Mỗi máy vi tính PC có 2 ổ nối IDE, mỗi ổ cho phép ghép nối 2 ổ đĩa cứng từ hay quang –laser (CD-ROM)

SCSI (Small Computer System Interface)

SCSI-1	5 MHz
SCSI-2	10 MHz
SCSI-3	40 MHz
SCSI-4	80 MHz

8. Điều khiển màn hình

Điều khiển màn hình truyền thống sử dụng mạch MC6845

Tùy thuộc và độ phân giải và số màu có thể hiển thị người ta chia ra các bộ điều khiển màn hình chính như sau:

Bộ điều khiển:	***Độ phân giải:***
CGA (Color Graphics Adapter)	640×200
EGA (Enhanced Graphic Adapter)	640×350
VGA (Video Graphic Array)	640×480
SVGA (Super VGA)	1024×768
XGA (eXtended Graphics Array)	1024×768
SXGA (Super XGA)	1280×800
UXGA (Ultra XGA)	1600×1200

9. Bộ ghép nối âm thanh

Chức năng:
Nhận tín hiệu âm thanh
Lưu trữ tín hiệu âm thanh
Xử lý tín hiệu âm thanh
Phát âm thanh
MIDI (Musical Instrument Digital Interface)
Cấu tạo gồm:
Các bộ chuyển đổi ADC, DAC
Bộ xử lý tín hiệu số, xử lý âm thanh (DSP Digital Signal Processor)
Bộ tổng hợp âm thanh
Giao diện MIDI.
Giao diện CD-ROM

10. Bộ ghép nối mạng

NIC (Network Interface Controller, ví dụ Intel 82558)
Cấu trúc 32 bit với 3KB bộ đệm phát và thu
Hai tốc độ 10 BASE-T và 100 BASE-TX

Quản lý tự động (ngắt)
Có tín hiệu báo trạng thái.

11. Bộ ghép nối nối tiếp đa năng USB (Universal Serial Bus)

Giao diện đơn giản, linh hoạt, dễ sử dụng
Ghép nối máy vi tính với bàn phím, đĩa mềm, con chuột, điện thoại..
Có 3 chế độ: tốc độ 1.5Mbps, tốc độ 12Mbps và tốc độ 480Mbps (USB 2.0)

12. Bộ ghép nối nối tiếp UART 8250 (Universal Asynchronous Receiver Transmitter) hay Intel 82450

Trong máy vi tính PC-XT có 1 mạch UART
Trong máy vi tính PC-AT có 2 mạch UART

13. Bộ ghép nối song song PPI (Programmable Parallel Interface)

- PPI cho phép nối nhiều kiểu thiết bị truyền dữ liệu song song nhiều bit

Ví dụ: Ghép nối máy in.

14. Ghép nối chuột

Chuột nối tiếp qua cổng: COM1, COM2
Qua cổng dùng tia hồng ngoại
USB
Điều khiển qua vi mạch 8042 (PS/2).

Điều khiển đĩa mềm FDC (Floppy Disk Controller)

Điều khiển 4 ổ đĩa
Hai chuẩn FM (Frequency Modulation) và MFM (Modified FM)

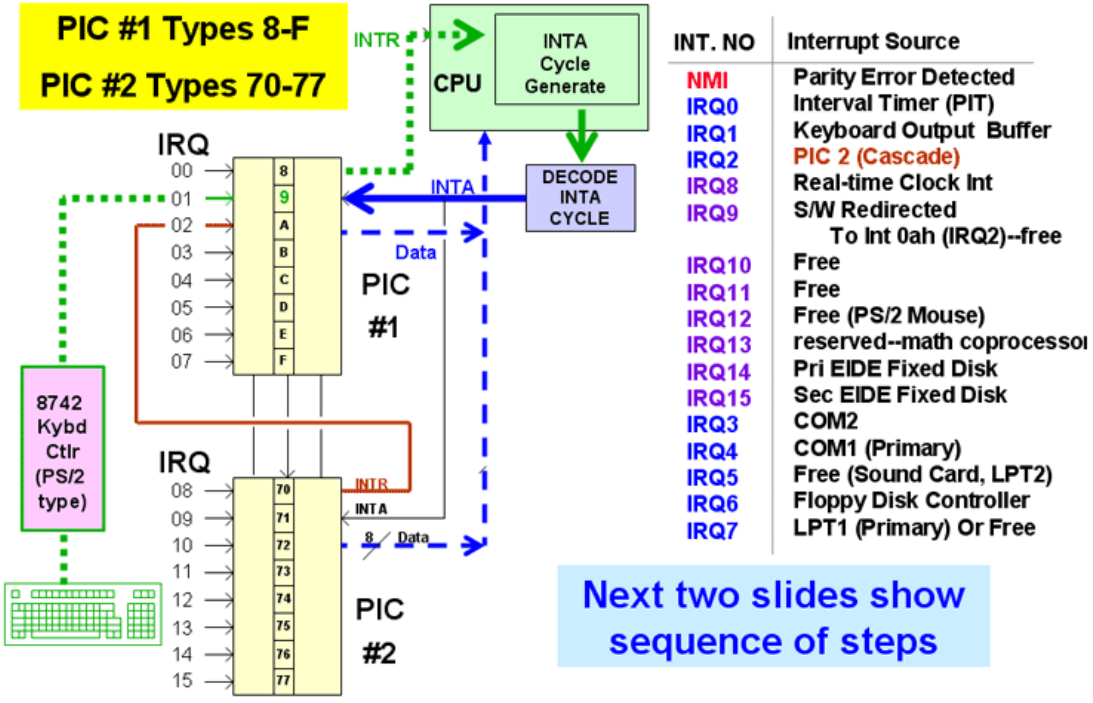
16. Điều khiển bàn phím KB(Keyboard)

- Sử dụng vi điều khiển Intel 8748
- Ghép nối nối tiếp với CPU

6.2 Các ngắt trong máy vi tính

Trong chế độ **địa chỉ thực** của họ vi xử lý của hãng Intel có 256 **số ngắt**. Mỗi số ngắt quy chiếu tới một vector ngắt. Mỗi vector ngắt gồm 4 byte: 2 byte đầu chứa địa chỉ offset, 2 byte tiếp theo chứa địa chỉ mảng của chương trình con phục vụ ngắt. Các vector ngắt đặt trong **Bảng vector ngắt** ở 1 KB đầu tiên của vùng nhớ RAM quy ước. Để truy nhập tới các vector ngắt, các ngắt cứng cung cấp số ngắt khi

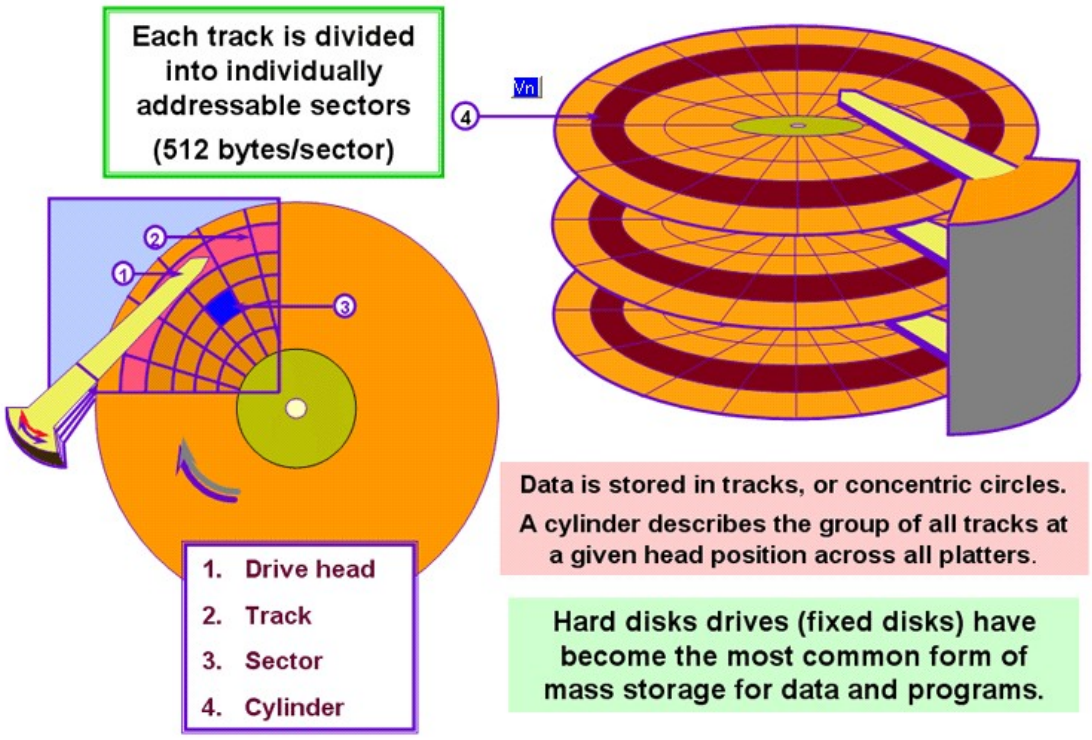
có yêu cầu ngắt (tín hiệu **IRQ), các** ngắt mềm cung cấp số ngắt khi thực hiện lệnh **INT** . Hình 6.2 định nghĩa các ngắt cứng.



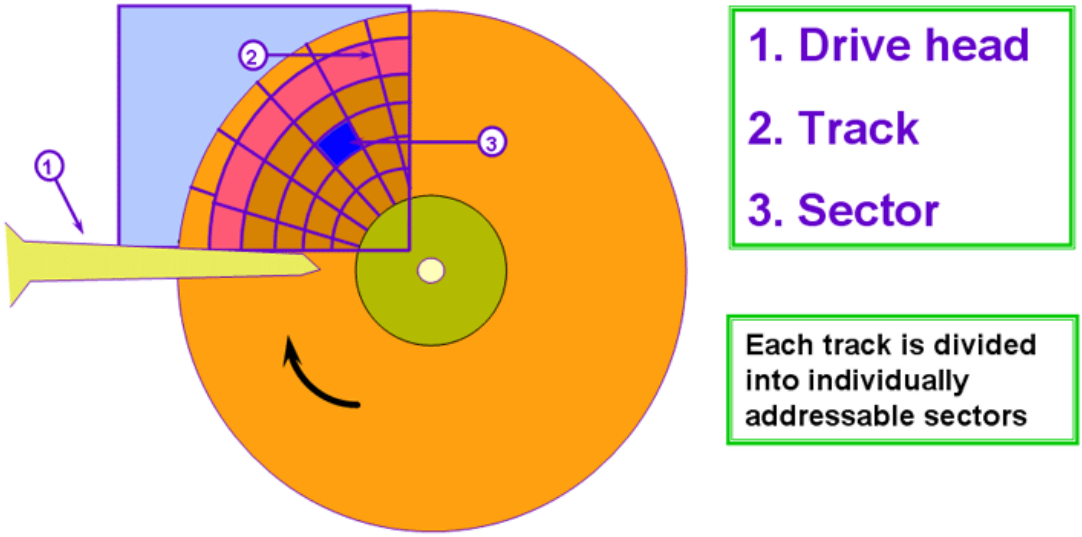
Hình 6.2 Các ngắt cứng trong máy vi tính

6.3. Nguyên lý cấu trúc và hoạt động của ổ đĩa

Tổ chức thông tin của đĩa từ được minh họa qua sơ đồ khối của chồng đĩa cứng (*Hình 6.3*) hay trên đĩa mềm (*Hình 6.4*)



Hình 6.3 Sơ đồ khối của chồng đĩa cứng



Hình 6.4 Sơ đồ bố trí thông tin trên đĩa mềm

Ví dụ về hoạt động của ổ đĩa mềm

Thông tin ghi trên đĩa mềm

Dữ liệu được ghi nối tiếp từng bit một trên các rãnh đồng tâm (track). Rãnh được đánh số từ ngoài vào tâm. Ví dụ: Đĩa 1.2MB có 80 rãnh được đánh số từ 00-79.

Rãnh được chia thành cung (sector). Một rãnh có 15 cung hay 18 cung (cho đĩa 1.44MB)

Việc phân chia cung có thể thực hiện bằng phần mềm.

Phân chia các cung được thực hiện theo hai chuẩn chính:

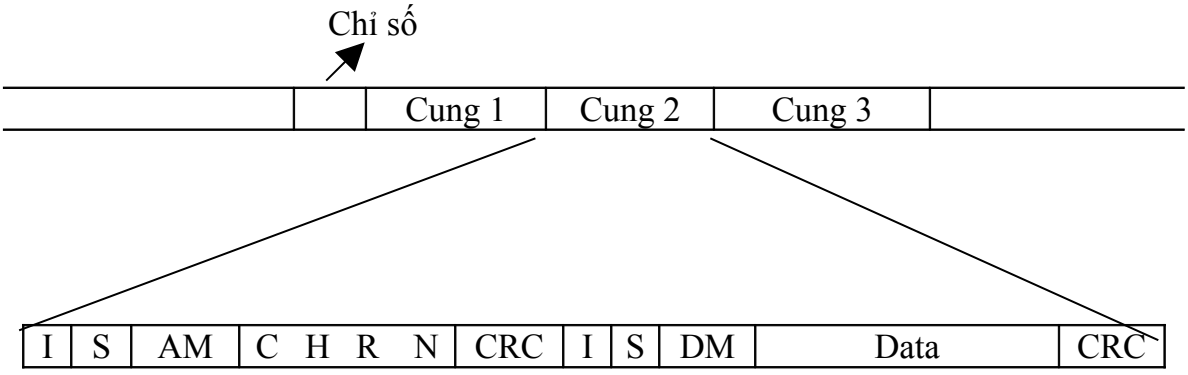
IBM 3740 cho các đĩa có mật độ đơn (FM)

IBM System 34 cho các đĩa có mật độ kép (MFM)

Cung đầu trên của một rãnh được nhận biết nhờ một cảm biến chỉ số.

Mỗi cung gồm 2 trường: trường địa chỉ và trường số liệu.

Ta minh họa bằng chuẩn IBM System 34



Hình 6.5 Cấu trúc của một cung trên đĩa mềm

Trong đó:

Trường địa chỉ:

I Khoảng trống bắt đầu một rãnh

S Khoảng trống đồng bộ hóa hoạt động của bộ điều khiển với tốc độ quay của đĩa

AM Đánh dấu bắt đầu vùng địa chỉ của cung

C H R N Địa chỉ của cung (Số trụ, Số đầu từ, Số cung, Mã độ dài của cung)

CRC Kiểm tra lỗi vùng địa chỉ cung

Hình 6.3 Sơ đồ khối của chồng đĩa cứng

Trường dữ liệu:

DM	Đánh dấu bắt đầu vùng số liệu
DATA	Vùng chứa số liệu (128, 256, 512 hay 1024 Byte)
CRC	Kiểm tra lỗi vùng dữ liệu của cung.

Hoạt động của ổ đĩa mềm được thực hiện qua hai thao tác cơ bản:

Thao tác tìm rãnh

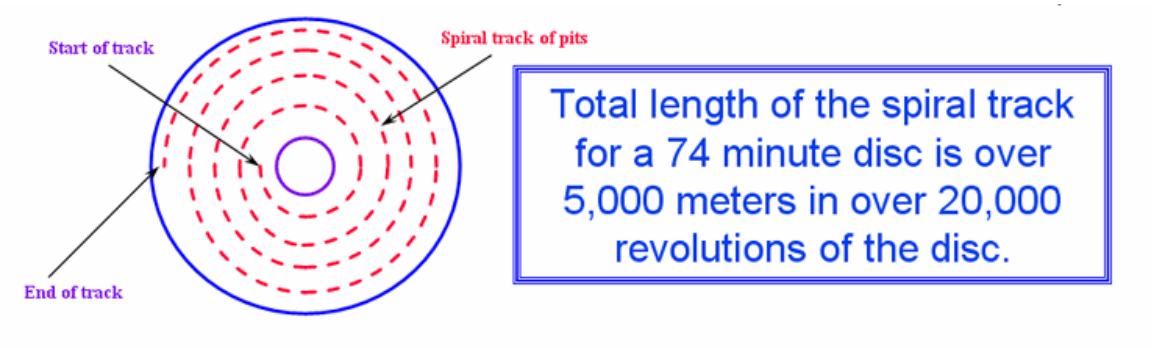
Chọn ổ đĩa trên đó phép tìm rãnh được thực hiện.
Xác định hướng dịch đầu từ (từ ngoài vào tâm hay ngược lại).
Cấm ghi
Tạo số bước cần thiết để dịch đầu từ đến rãnh cần tìm.

Kết quả phép tìm rãnh được kiểm tra bằng cách đọc trường địa chỉ tại vị trí hiện tại của đầu từ và so sánh với địa chỉ rãnh cần tìm. Nếu địa chỉ đọc được bằng địa chỉ rãnh cần tìm thì phép tìm rãnh đạt kết quả. Nếu chưa đúng thì có thể dịch đầu từ về rãnh 0 và phép tìm rãnh lặp lại (Nếu thử 10 lần mà kết quả không đạt thì ổ đĩa có lỗi).

Thao tác đọc/ghi số liệu thực hiện sau phép tìm rãnh

Chọn ổ đĩa cần đọc/ghi
Hạ đầu từ và ổn định đầu từ
Cho phép đọc để tìm địa chỉ cung mong muốn đọc hay ghi
Thực hiện phép đọc/ghi số liệu khi tìm đúng cung.

Ghi chú: Tổ chức thông tin trên đĩa CD-ROM theo đường xoáy trôn ốc từ trong ra ngoài chứ không phải là các rãnh đồng tâm từ ngoài vào tâm. Như Hình 6.5

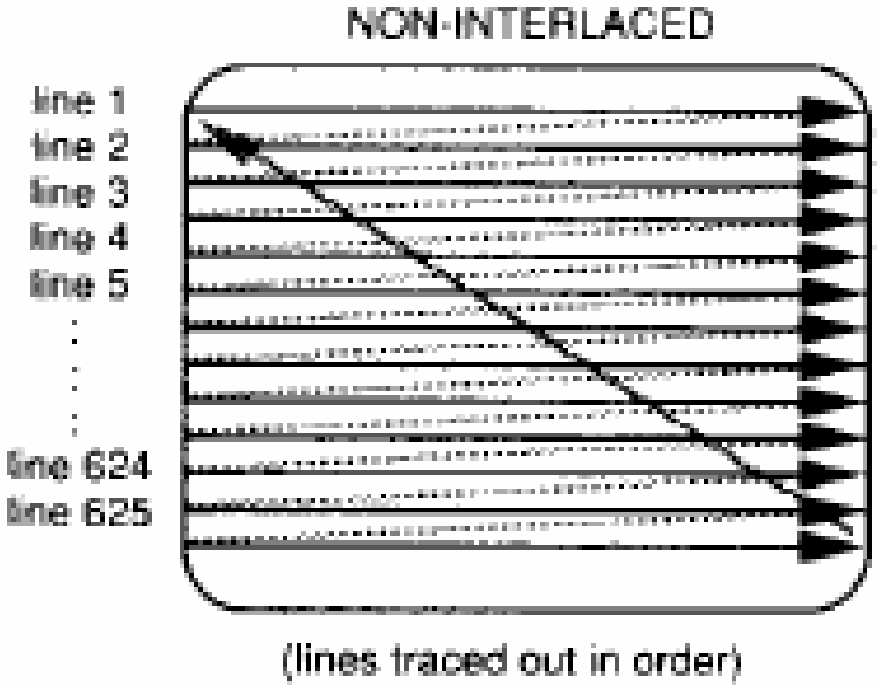


Hình 6.6 Cách ghi số liệu trên đĩa CD-ROM

6.4. Thiết bị màn hình

Trong phạm vi của môn học, chúng ta chỉ tìm hiểu phương pháp tạo ảnh và ký tự trên màn hình.

Phương pháp tạo ảnh trên màn hình thông dụng nhất là phương pháp quét mành (Raster Scan) như Hình 6.6



Hình 6.7 Nguyên lý quét màn hình.

Nguyên lý:

Điểm bắt đầu quét của tia điện tử ở phía trên bên trái màn hình, tọa độ (0,0).

Tia điện tử quét ngang từ trái qua phải. Khi hết dòng, tia điện tử hồi nhanh về đầu dòng (tia hồi ngang được dập tắt).

Trong khi chuyển động ngang, tia điện tử cũng được lái theo chiều dọc. Như vậy dần dần tia điện tử sẽ quét hết cả màn hình.

Mỗi điểm trên màn hình sẽ được chiếu sáng (1) hay tối (0). Tập hợp các điểm sáng tối sẽ tạo ra ảnh đen trắng trên màn hình.

Nếu là màn hình màu thì mỗi điểm ảnh phải được tạo ra bởi 3 màu cơ bản.

Ví dụ ta chọn 3 màu cơ bản là đỏ, lục, lam thì tổ hợp của 3 màu đó sẽ cho các màu theo bảng sau:

Đỏ	Lục	Lam	Màu
0	0	0	Đen
1	0	0	Đỏ
0	1	0	Lục
0	0	1	Lam
1	1	0	Vàng
1	0	1	Tím thẫm (mageta)
0	1	1	Lam lục (Cyan)
1	1	1	Trắng

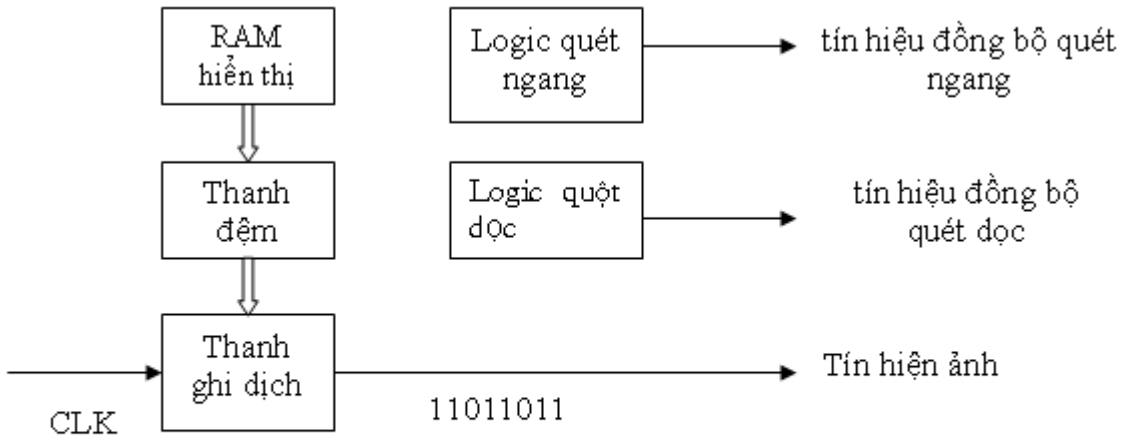
Trong màn hình: cứ 3 điểm huỳnh quang kế cận ứng với 3 màu cơ sở tạo thành một tam giác màu.

Màn hình phủ lớp phốt pho không lưu được ảnh lâu, để ảnh khỏi nhấp nháy, ảnh phải được thực hiện lại khoảng 50 lần /giây. Nếu quét xen kẽ thì có thể hiện 25 lần /giây. Quá trình này người ta gọi là làm tươi màn hình REFRESH.

Ảnh được lưu giữ trong bộ nhớ RAM, bộ nhớ này là RAM hiển thị (video RAM). Bộ nhớ RAM hiển thị phải được đọc để đưa lên màn hình 50 lần (25 lần) /giây.

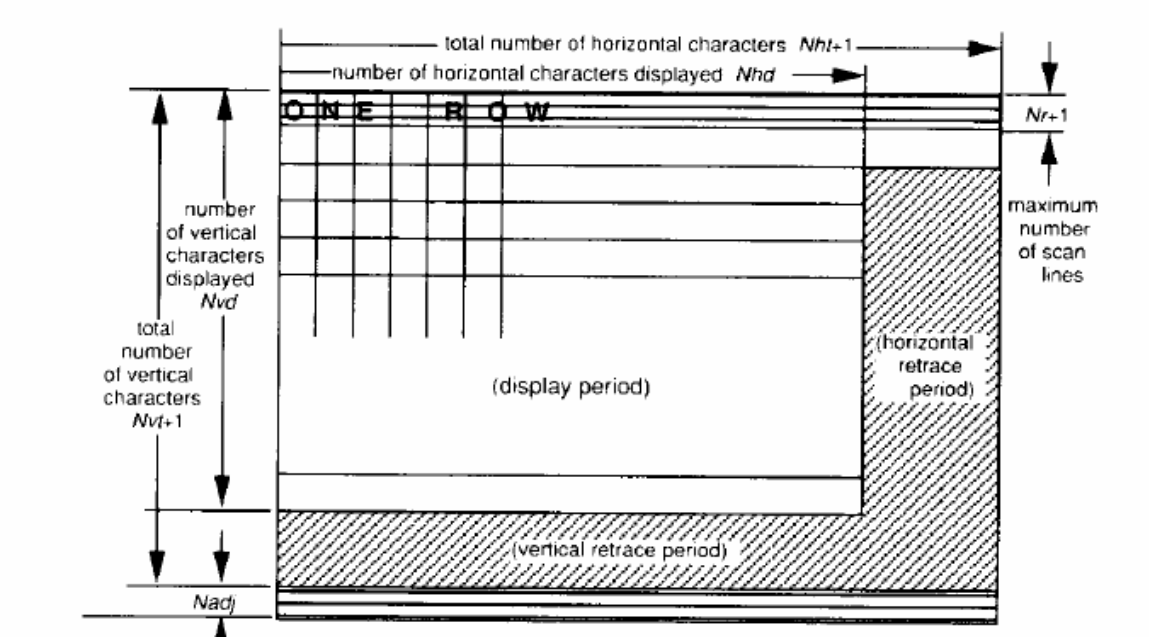
Việc tạo ảnh trên màn hình thực chất là thay đổi số liệu trong RAM hiển thị.

Để ảnh không bị trôi phải có tín hiệu đồng bộ ngang (HSYNC) và đồng bộ dọc (VSYNC)



Hình 6.8 Nguyên lý đưa tín hiệu ảnh lên màn hình

Nguyên lý hiển thị các ký tự trên màn hình được thể hiện qua Hình 6.8



Hình 6.9 Nguyên lý hiển thị các ký tự trên màn hình

MỤC LỤC

Tài liệu tham khảo chính:

[1] William Stallings, "Computer Organization and Architecture", International Edition, 1997
[2] INTEL System Board Technology, 2004
[3] Văn Thế Minh. Kỹ thuật vi xử lý, NXB Giáo dục, 1997
[4] Nguyễn Nam Trung, "Cấu trúc máy vi tính & Thiết bị ngoại vi", NXB Khoa học và kỹ thuật, 2000