Tổ chức và Cấu trúc máy tính

GIỚI THIỆU

I. LỊCH SỬ MÁY TÍNH SỐ :

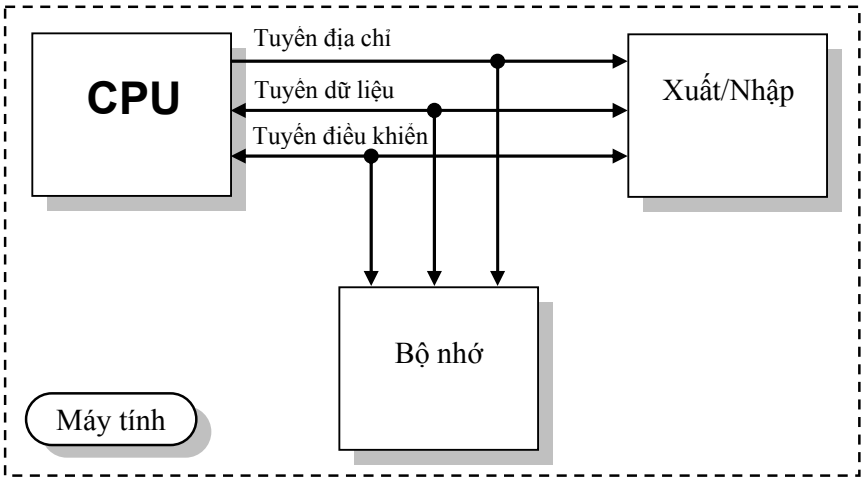
II. CẤU TRÚC CỦA MỘT HỆ THỐNG MÁY TÍNH :

1. Máy tính là gì ?

- Máy tính số là công cụ giúp con người giải quyết các công việc tính toán với tốc độ cao.
- Máy tính là một nhánh phát triển của ngành điện tử và hoạt động chủ yếu nhờ các linh kiện số.
- Nguyên tắc hoạt động chính của máy tính là thực hiện liên tục các lệnh. Các lệnh này do con người cung cấp ở nhiều dạng khác nhau trong đó dạng thấp nhất là số hệ 2. Các dạng khác như hợp ngữ, các ngôn ngữ lập trình, các ngôn ngữ cấp cao đều được dùng với mục đích làm giảm nhẹ việc lập trình bằng mã máy tức số hệ 2.
- Trong máy tính chia ra làm hai phần cứng và mềm.
- Phần cứng là phần vật chất cụ thể tạo nên máy tính như nguồn cung cấp, mạch chính máy tính, các thiết bị ngoại vi, ...
- Phần mềm là phần trừu tượng như các ý niệm, các giải thuật, các chương trình ...
- Vi xử lý là tên gọi của phần linh kiện số có chức năng điều hành mọi hoạt động của hệ thống máy tính. Tổng quát hơn, người ta gọi vi xử lý các linh kiện có khả năng giải quyết vấn đề bằng các chương trình.

2. Mô hình Von Neumann :

- Các máy tính từ lúc ra đời cho đến nay đều được chế tạo, cải tiến dựa trên mô hình Von Neumann như sau :

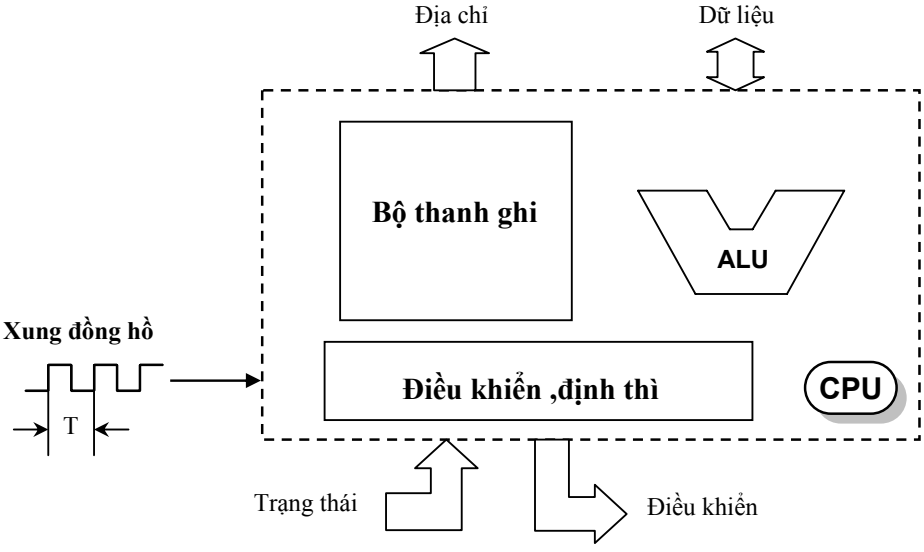


- CPU là khối xử lý trung tâm có khả năng điều hành mọi hoạt động của hệ thống.
- Khối bộ nhớ có chức năng lưu giữ các thông tin.

- Khối xuất nhập có chức năng điều khiển các thiết bị ngoại vi như màn hình, bàn phím, máy in ...
- Khối CPU điều khiển khối bộ nhớ và khối xuất nhập thông qua 3 tuyến : địa chỉ, dữ liệu và điều khiển/trạng thái.
- Tuyến địa chỉ dùng để lựa chọn, phân biệt vị trí các ô nhớ, các thiết bị ngoại vi.Tuyến địa chỉ là tuyến một chiều : ra CPU, vào các khối còn lại.
- Số lượng địa chỉ mà tuyến địa chỉ có thể quản lý được tùy thuộc vào số đường địa chỉ của tuyến. Chẳng hạn, với 1 đường địa chỉ, vi xử lý có thể phân biệt được 2 địa chỉ là 0 và 1; với 2 đường địa chỉ khả năng định địa chỉ lên đến 4 địa chỉ do sự tổ hợp của hai bit địa chỉ nói trên gồm địa chỉ 0 (00), 1 (01), 2 (10) và 3 (11); ...
- Trong trường hợp tổng quát, với tuyến địa chỉ có n đường, khả năng quản lý địa chỉ bộ nhớ lên đến 2n. (Với các CPU từ 386 trở lên, số đường địa chỉ là 32 nên dung lượng tối đa có thể quản lý được là 232 = 4 GB bộ nhớ = 4096 MB)
- Tuyến địa chỉ thường được ký hiệu bằng chữ **A** hay ***a*** **($a_{31}a_{30}...a_1a_0$)**.
- Tuyến dữ liệu là đường trao đổi thông tin giữa các khối với nhau. Tuyến dữ liệu là tuyến hai chiều. Với các CPU 386 trở lên, tuyến dữ liệu có 32 đường cho phép mỗi lần trao đổi được 4 byte dữ liệu.
- Tuyến dữ liệu thường được ký hiệu bằng chữ ***D*** hay ***d*** **($d_{31}d_{30}...d_1d_0$)**.
- Tuyến địa chỉ và tuyến dữ liệu theo sơ đồ Von Neumann là tuyến dùng chung cho cả hai khối bộ nhớ và xuất nhập với mục đích là tiết kiệm số đường trong mỗi tuyến. Chính vì vậy nên cần có thêm tuyến điều khiển để xác định rõ vi xử lý muốn làm việc với bộ nhớ hay với xuất nhập, hoặc chiều dữ liệu là chiều ra CPU hay vào CPU, ...
- Trên tuyến điều khiển, đường nào có chiều ra khỏi CPU thường được xem là dường điều khiển. Đường nào có chiều đi vào CPU được xem là các đường trạng thái.
- Mỗi đường trên tuyến điều khiển thường mang một tên riêng tùy theo ý nghĩa của mỗi đường. Chẳng hạn như MEMR là tín hiệu điều khiển việc đọc bộ nhớ, MEMW điều khiển ghi bộ nhớ, IORD điều khiển quá trình nhập, IOWR điều khiển việc xuất dữ liệu ...
- Các đường điều khiển/trạng thái có thể tác động ở mức 1 hoặc mức 0.

3. CPU - Khối xử lý trung tâm :

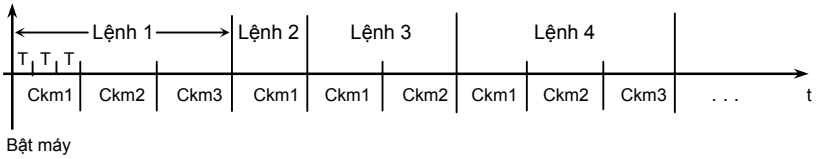
- Khối xử lý trung tâm điều hành các hoạt động của hệ thống bằng cách thực hiện liên tục và lặp đi lặp lại 2 bước : lấy lệnh và thi hành lệnh.



- Lệnh mà CPU thi hành được nạp trước đó vào trong bộ nhớ.
- Các lệnh nằm liên tục trong bộ nhớ tạo thành chương trình.
- CPU là một hệ thống số tuần tự, đồng bộ nên việc cung cấp xung đồng hồ clock là cần thiết.
- CPU hoạt động được với xung clock có tần số càng cao thì chạy càng nhanh.

3.1. Khối điều khiển, định thì (Control and timing unit) :

- Khối điều khiển định thì thể hiện chức năng điều khiển thông qua 3 bước :
  . Lấy lệnh : thực hiện quá trình đọc bộ nhớ, địa chỉ hay vị trí lệnh trong bộ nhớ được lưu giữ trong một thanh ghi đặc biệt của bộ thanh ghi. Nội dung của ô nhớ chính là mã lệnh.
  . Giải mã lệnh : xác định thao tác cần thực hiện từ mã lệnh đọc được.
  . Thi hành lệnh : thực hiện một trong các hoạt động với khối ALU, với bộ thanh ghi, với khối bộ nhớ và với khối xuất nhập. Các hoạt động với ALU và bộ thanh ghi được thực hiện trong nội bộ CPU. Các hoạt động với bộ nhớ hay xuất nhập được thực hiện bằng cách phát ra các tín hiệu điều khiển qua tuyến điều khiển ra ngoài.
- Chức năng định thì của khối điều khiển định thì được thực hiện thông qua các thời khoản gọi là **chu kỳ máy**. Chu kỳ máy là đơn vị thời gian nhỏ nhất trong các hoạt động của CPU.
- Một chu kỳ máy có thể kéo dài từ 3 đến 4 **chu kỳ xung clock** hoặc thay đổi tùy theo mỗi loại CPU.
- Các chu kỳ máy cơ bản là :
  - Chu kỳ lấy lệnh.
  - Chu kỳ đọc bộ nhớ.
  - Chu kỳ ghi bộ nhớ.
  - Chu kỳ xuất.
  - Chu kỳ nhập.
  - Chu kỳ đáp ứng ngắt quãng.
- Thời gian thực hiện hoàn tất một lệnh từ lúc lấy lệnh đến lúc thi hành xong lệnh được gọi là chu kỳ lệnh.
- **Chu kỳ lệnh** là một tổ hợp của một hoặc nhiều chu kỳ máy.
- Như vậy có thể xem hoạt động của một vi xử lý là một chuỗi nối tiếp các chu kỳ lệnh hay các chu kỳ máy cũng vậy trên trục thời gian.



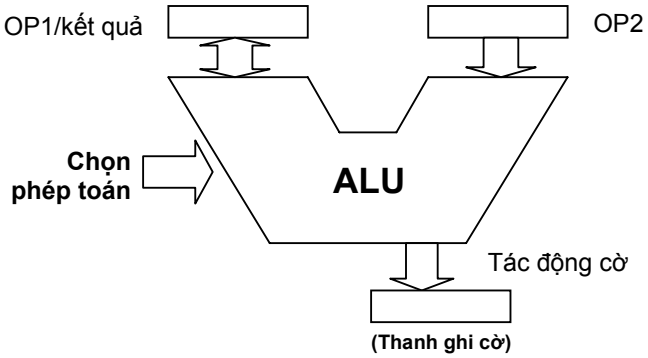
- Như vậy với cùng một lệnh của CPU, nếu tần số xung clock càng cao tức chu kỳ càng nhỏ thì thời gian thi hành lệnh càng ngắn.
- Các lệnh của CPU được mã hóa thành các mã lệnh. Các lệnh sử dụng thường xuyên được có 1 byte mã lệnh. Các lệnh sử dụng không thường xuyên được mã hóa thành 2 byte mã lệnh tức là thuộc các nhóm lệnh phụ . Nhờ vậy số lượng các lệnh có thể nhiều hơn 256 lệnh.

3.2. Bộ thanh ghi (Registers unit) :

- Thanh ghi là một dạng bộ nhớ hoạt động được ở tốc độ cao.
- Bộ thanh ghi chính là bộ nhớ trong của CPU.

- Số lượng các thanh ghi trong bộ thanh ghi thường bị hạn chế vì khó chế tạo, giá thành cao.
- CPU dùng các thanh ghi với những mục đích khác nhau thể hiện qua tên gọi của mỗi thanh ghi.
- Thanh ghi bộ đếm chương trình PC (program counter) dùng để giữ địa chỉ ô nhớ chứa mã lệnh sắp thi hành. Địa chỉ này sẽ được CPU đưa lên tuyến địa chỉ trong bước lấy lệnh.
- Thanh ghi bộ tích lũy ACC (accummulator) có vai trò quan trọng trong phần lớn các lệnh của CPU. Bộ tích lũy là thanh ghi được sử dụng nhiều hơn các thanh ghi khác trong bộ thanh ghi. ACC được dùng nhiều trong tính toán và giữ kết quả sau khi tính.
- Các CPU đủ mạnh cho phép thực hiện các phép tính trên cả các thanh ghi đa dụng khác.
- Thanh ghi đa dụng (general purpose register) là các thanh ghi được sử dụng với các mục đích khác nhau như dùng làm bộ đếm, chứa địa chỉ, chứa dữ liệu, có thể được dùng trong tính toán nhưng không linh động và mạnh như ACC ...
- Thanh ghi chỉ số (index register) dùng cho việc xử lý dãy, bảng hay chuỗi ký tự. Nội dung thanh ghi chỉ số được cộng vào địa chỉ bộ nhớ nền (base address) khi xác định vị trí ô nhớ.
- Thanh ghi con trỏ chồng SP (stack pointer) (con trỏ ngăn xếp) dùng cho phương pháp định địa chỉ theo chồng. Chồng là một vùng bộ nhớ được dành riêng để chứa các thông tin cần cất tạm trong một khoản thời gian ngắn, hoặc đôi khi có số lượng lớn mà không thể chứa trong các thanh ghi. SP giữ địa chỉ đỉnh chồng là nơi thông tin được cất vào. Chồng còn được gọi là dãy vào sau ra trước (Last In First Out).
- Thanh ghi cờ trạng thái (flags register) là thanh ghi được dùng theo từng bit để phản ánh trạng thái của CPU, của chương trình hoặc kết quả . Mỗi bit được gọi là một cờ và mang tên riêng như cờ nhớ (carry flag), cờ không (zero flag), cờ dấu (sign flag), cờ tràn (overflow flag), cờ kiểm tra chẵn/lẻ (parity flag), ...

3.3. Bộ số học luận lý ALU (Arithmetic logic unit) :



- ALU giữ vai trò tính toán trong CPU.
- ALU sử dụng hai thanh ghi toán hạng OP1 và OP2 để giữ các toán hạng và kết quả.
- ALU có thể thực hiện được các phép số học như cộng, trừ, tăng, giảm, nhân, chia, so sánh, các phép luận lý NOT, AND, OR, XOR, phép dịch (shift), quay (rotate).

3.4. Tập lệnh và các phương pháp định địa chỉ :
- Tập hợp các lệnh mà vi xử lý có thể thực hiện được gọi là tập lệnh.

- Tập hợp các lệnh nằm trong bộ nhớ mà vi xử lý phải thi hành theo một thứ tự nhất định để giải quyết một vấn đề nào đó gọi là chương trình.
- Mỗi lệnh thường mang một số thông tin như :
  . Hoạt động : cho biết lệnh làm gì, còn gọi là mã lệnh (opcode)
  . Các nguồn dữ liệu hay các toán hạng.
  . Nơi chứa kết quả.
  . Địa chỉ lệnh kế.
- Dạng tổng quát của một lệnh vi xử lý như sau :

Mã lệnh	Địa chỉ toán hạng 1	Địa chỉ toán hạng 2	Địa chỉ kết quả	Địa chỉ lệnh kế
8 bit	16 bit	16 bit	16 bit	16 bit

- Để giảm bớt chiều dài câu lệnh 72 bit=9 byte , người ta thường dùng một số thông tin ở dạng ẩn. Dùng thanh ghi PC để giảm bớt địa chỉ lệnh kế, dùng một trong hai toán hạng để chứa kết quả thì giảm được địa chỉ kết quả, dùng thanh ghi thay cho bộ nhớ sẽ giảm được địa chỉ toán hạng ...
- Các phương pháp định vị toán hạng là các cách thể hiện nơi chứa toán hạng cần xử lý.
- Các tiêu chuẩn để chọn phương pháp định vị toán hạng :
  . Địa chỉ dùng trong câu lệnh ngắn.
  . Truy xuất được bộ nhớ lớn.
  . Linh động.
  . Xác định địa chỉ nhanh.
  . Đơn giản.
- Các phương pháp thường dùng : trực tiếp, gián tiếp, tức thời, chỉ số, tương đối, thanh ghi, thanh ghi gián tiếp và chồng.

* *Trực tiếp* : địa chỉ trong lệnh là địa chỉ ô nhớ chứa toán hạng.

* *Gián tiếp* : địa chỉ trong lệnh là địa chỉ ô nhớ chứa một địa chỉ thứ hai. Địa chỉ thứ hai mới là địa chỉ ô nhớ chứa toán hạng.

* *Tức thời* : địa chỉ trong lệnh được thay bằng chính toán hạng.

* *Chỉ số* : địa chỉ ô nhớ chứa toán hạng được xác định bằng cách cộng địa chỉ trong lệnh với nội dung thanh ghi chỉ số.

* *Tương đối* : địa chỉ ô nhớ chứa toán hạng được xác định bằng cách cộng địa chỉ trong lệnh với nội dung thanh ghi PC.

* *Thanh ghi* : địa chỉ trong lệnh là địa chỉ thanh ghi và thường được ghép vào trong mã lệnh có nghĩa là sẽ không còn vùng địa chỉ toán hạng nữa).

* *Thanh ghi gián tiếp* : địa chỉ trong lệnh là địa chỉ thanh ghi. Nội dung thanh ghi là địa chỉ ô nhớ chứa toán hạng.

* *Chồng* : địa chỉ ô nhớ chứa toán hạng nằm trong thanh ghi SP nên trong lệnh không cần địa chỉ toán hạng nữa.

- Trong thực tế, các tập lệnh của các vi xử lý dùng nhiều phương pháp định vị toán hạng phối hợp với nhau trong cùng một lệnh, hoặc có thể không sử dụng một vài phương pháp và điều đó nói lên các điểm mạnh, yếu của từng vi xử lý trong vấn đề lập trình.

4. Tổ chức bộ nhớ :

4.1. Đặc điểm :

- Hiện nay, các bộ nhớ dùng trong máy tính đều là các bộ nhớ bán dẫn có khả năng lưu trữ thông tin theo dạng nhị phân.

- CPU muốn làm việc với bộ nhớ phải cung cấp địa chỉ, dữ liệu (trong trường hợp ghi thông tin vào bộ nhớ), và một số tín hiệu điều khiển như chọn bộ nhớ, cho phép đọc hoặc ghi.
- Đại lượng đặc trưng cho bộ nhớ là dung lượng và thời gian truy xuất bộ nhớ.
- Thời gian truy xuất bộ nhớ là khoản thời gian *từ lúc bộ nhớ nhận được địa chỉ và các tín hiệu điều khiển cho đến lúc đưa được dữ liệu ra tuyến dữ liệu*.
- Bộ nhớ có thời gian truy xuất càng nhỏ thì hoạt động càng nhanh.
- Bộ nhớ có dung lượng càng lớn thì càng chứa được nhiều thông tin.

4.2. Tổ chức :

- Bộ nhớ của máy tính được tổ chức theo đơn vị truy xuất là byte.
- Trong thực tế, bộ nhớ bán dẫn được sản xuất theo dạng các linh kiện có dung lượng hạn chế (từ vài KB cho đến cỡ vài chục MB).
- Trong trường hợp dung lượng các linh kiện bộ nhớ không đủ đáp ứng dung lượng bộ nhớ của hệ thống khi thiết kế, nhà thiết kế phải ghép nhiều linh kiện bộ nhớ lại và phải giải quyết vấn đề giải mã địa chỉ.
- Giải mã địa chỉ bộ nhớ là bước không thể thiếu trong thiết kế bộ nhớ cho một hệ thống máy tính hoặc một hệ thống điều khiển tự động dùng vi xử lý. Đó chính là bước qui định tầm địa chỉ cho từng linh kiện bộ nhớ được ghép lại.
- Chẳng hạn, dùng 4 linh kiện bộ nhớ 4MB để tạo ra không gian bộ nhớ 16MB cho hệ thống thì tầm địa chỉ của các linh kiện bộ nhớ 4MB như sau :
  . 4MB đầu tiên : **000000H** **3FFFFFH**
  . 4MB thứ hai : **400000H** **7FFFFFH**
  . 4MB thứ ba : **800000H** **BFFFFFH** và
  . 4MB thứ tư : **C00000H** **FFFFFFH**.

4.3. Phân loại :

Bộ nhớ bán dẫn được chia thành hai loại chính là bộ nhớ chỉ đọc ROM (Read Only Memory) và bộ nhớ truy xuất bất kỳ RAM (Random Access Memory).

a) Bộ nhớ chỉ đọc ROM :

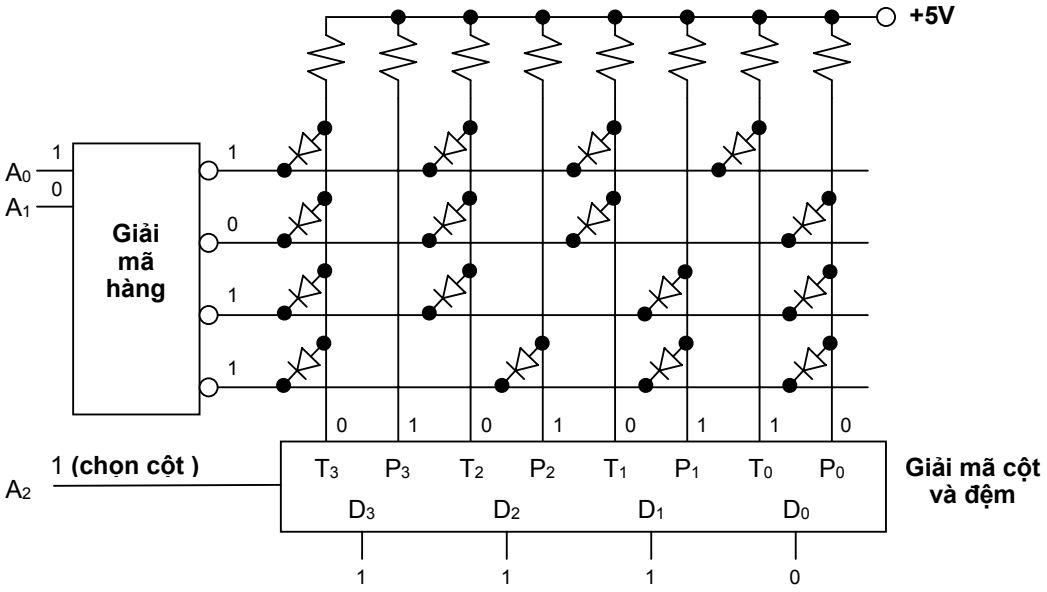
- ROM được dùng để giữ các thông tin không thay đổi như các chương trình khởi động máy tính (POST : Power On Self-Test), các hệ thống các chương trình con xuất nhập cơ bản của máy tính (BIOS : Basic Input Output System), các bảng thông số, ...

A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	1
0	1	1	0	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	0
1	1	0	1	1	0	0
1	1	1	1	0	0	0

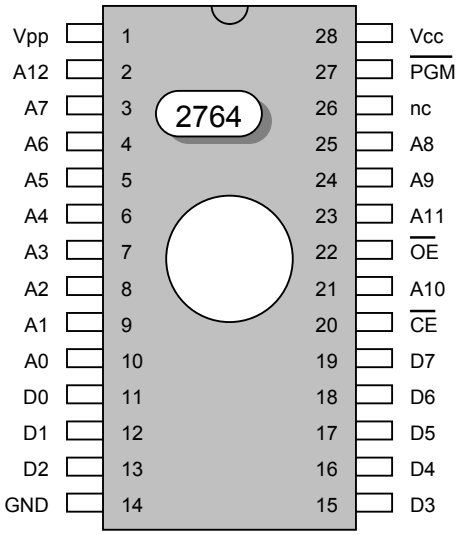
A_1 A_0 : giải mã hàng

A_2 : giải mã cột

- Thông tin được ghi vào ROM là thông tin chết, không thể sửa đổi nhưng không bị mất đi khi mắt nguồn cung cấp.
- Thông tin trong ROM được lưu trữ theo dạng ma trận. Mỗi thông tin được xác định bằng hai tọa độ là địa chỉ hàng và địa chỉ cột.
- Ví dụ ta có một ROM 8 x 4bit có nội dung và cấu tạo như sau :



- Các loại ROM :
  . ROM : dạng nguyên thủy ban đầu, thông tin được ghi lúc chế tạo.
  . PROM (Programmable ROM) : PROM được chế tạo ở dạng chưa có thông tin hay còn gọi là ROM trắng. Người sử dụng có thể ghi thông tin vào sau nhưng phải có thiết bị ghi chuyên dụng. PROM chỉ cho phép ghi thông tin một lần duy nhất.
  . EPROM (Erasable PROM) : EPROM cho phép xóa thông tin đã có trong EPROM và ghi thông tin mới vào EPROM nhiều lần. Việc xóa nội dung EPROM phải nhờ đến tia cực tím rọi trực tiếp vào cửa sổ thủy tinh trên linh kiện EPROM trong vòng 10 phút. Việc ghi thông tin vào EPROM cũng phải dùng thiết bị chuyên dụng.
  . EAROM (Electrically Alterable ROM) hay EEPROM (Electrically Erasable PROM) : gọi là ROM điện, cho phép ghi, xóa bằng xung điện mà không cần đến thiết bị đặc biệt.
  . Flash ROM là kết quả của sự kết hợp giữa ROM điện và RAM. ROM điện dùng để lưu trữ thông tin còn lúc sử dụng thì thông tin được đổ từ ROM điện sang RAM để chạy cho nhanh.
- Trong các loại ROM trên, EPROM được sử dụng rộng rãi trong thiết kế. Các EPROM thông dụng là 2716 (2K×8bit), 2732 (4K×8bit), 2764 (8K×8bit), 27128 (16K×8bit), ...
- Các tín hiệu cần thiết để cho ROM hoạt động gồm có các đường *địa chỉ* (A) từ tuyến địa chỉ, các đường *dữ liệu* (D) từ tuyến dữ liệu, tín hiệu *chọn chip* (CE hay CS) từ mạch giải mã địa chỉ bộ nhớ (là mạch vừa chọn bộ nhớ vừa chọn tầm địa chỉ cho linh kiện nhớ) và tín hiệu *đọc* (RD) từ tuyến điều khiển.



A12 ÷ A0 : địa chỉ
D7 D0 : dữ liệu
$\overline{OE}$: cho phép xuất
$\overline{CE}$: chọn chip
Vcc : nguồn
GND : đất
Vpp : nguồn lập trình
$\overline{PGM}$: chọn lập trình

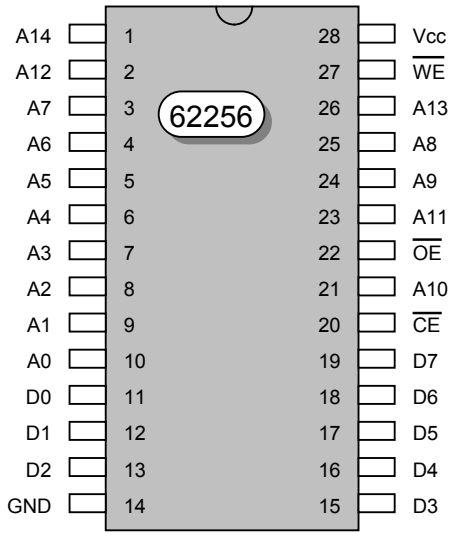
b) Bộ nhớ truy xuất bất kỳ RAM :

- RAM là bộ nhớ cho phép đọc hoặc ghi thông tin bất kỳ lúc nào trong quá trình làm việc mà không đòi hỏi thiết bị đặc biệt gì cả.
- Thông tin trong RAM chỉ tồn tại trong khi làm việc có nghĩa là khi mất nguồn cung cấp thì thông tin trong RAM sẽ bị mất hết.
- Dung lượng RAM chiếm đa số trong thiết kế bộ nhớ máy tính.
- RAM thường được dùng làm bộ nhớ làm việc như để chứa các thông số của hệ điều hành, để nạp các chương trình điều hành hệ thống, các chương trình ứng dụng. RAM còn được dùng làm các biến bộ nhớ, các vùng đệm dữ liệu cho đĩa.
- RAM cũng được chế tạo theo dạng ma trận nhớ nhưng mỗi phần tử nhớ khác với cấu tạo của ROM để có thể đọc ghi bất kỳ được.
- Người ta chia RAM ra làm hai loại chính : SRAM tức RAM tĩnh và DRAM tức RAM động.

* SRAM (Static RAM) :

. SRAM là loại RAM mà mỗi giao điểm của hàng và cột có cấu tạo như một flip-flop có khả năng giữ lại một bit thông tin.

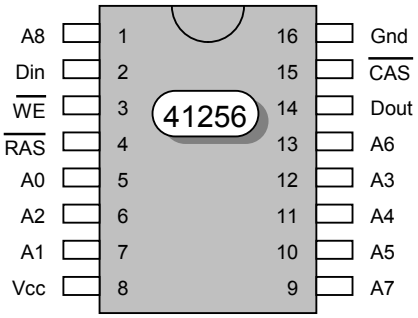
. Để truy xuất SRAM, cần có các tín hiệu địa chỉ, dữ liệu, chọn chip (CS) và tín hiệu chọn đọc hoặc ghi (RD, WR).



A14 A0 : địa chỉ
D7 D0 : dữ liệu
$\overline{OE}$: cho phép xuất
$\overline{CE}$: chọn chip
$\overline{WE}$: cho phép ghi
Vcc : nguồn
GND : đất

. SRAM được dùng nhiều trong thiết kế các mạch điều khiển bằng vi xử lý vì dễ thiết kế, dễ sử dụng nhưng ngược lại không thể chế tạo dung lượng lớn do số chân địa chỉ của vi mạch nhớ tăng lên quá nhiều.
. SRAM thông dụng là 6116 (2K×8bit), 6264 (8K×8bit), 62256 (32K×8bit).

* DRAM (Dynamic RAM) :
. DRAM có cấu tạo bit nhớ đơn giản hơn, ít linh kiện hơn SRAM nên có thể tăng dung lượng lên khá cao.
. Tuy nhiên, thông tin trong DRAM không tồn tại được như trong SRAM mà sẽ bị rò rỉ mất đi sau một khoản thời gian cỡ 2ms. Như vậy, muốn sử dụng DRAM để thiết kế bộ nhớ máy tính, người ta phải giải quyết vấn đề phục hồi nội dung của tất cả các ô nhớ trước khi nó mất đi. Thao tác đó gọi là làm mới bộ nhớ (refresh).
. Mặt khác, khi dung lượng của linh kiện nhớ tăng lên, số đường địa chỉ tăng theo nên muốn cho kích thước linh kiện không lớn quá người ta phải dùng phương pháp chọn địa chỉ hàng và địa chỉ cột (multiplexed address). Lúc đó, số chân linh kiện cho các đường địa chỉ giảm đi phân nữa (có nghĩa là địa chỉ hàng và địa chỉ cột dùng chung các chân địa chỉ) nhưng cần có thêm chân điều khiển chọn địa chỉ hàng RAS và chọn địa chỉ cột CAS.
. Các DRAM thường dùng : 4164 (64K×1bit), 41256 (256K×1bit), 44256 (256K×4bit) ...



A8 A0 : địa chỉ hàng / cột 18 bit
Din, Dout: dữ liệu nhập, xuất
$\overline{RAS}$: cho phép xuất
$\overline{CAS}$: chọn chip
$\overline{WE}$: cho phép ghi
Vcc : nguồn
Gnd : đất

5. Khối xuất nhập - Thiết bị ngoại vi :

5.1. Đặc điểm :
- Khối xuất nhập có chức năng điều khiển các thiết bị ngoại vi để giao tiếp với thế giới bên ngoài.
- Các thiết bị ngoại vi được điều khiển độc lập với nhau.
- Mỗi thiết bị ngoại vi thường được điều khiển bởi một mạch vi xử lý chuyên dụng riêng. Chẳng hạn như mạch điều khiển màn hình, mạch điều khiển đĩa, mạch điều khiển bàn phím, mạch điều khiển giao tiếp mạng ...
- Như vậy, trong khối xuất nhập có nhiều vi xử lý khác nhau.
- Mỗi vi xử lý điều khiển xuất nhập được thiết kế để giúp đỡ cho khối xử lý trung tâm trong việc giao tiếp với thiết bị ngoại vi, trong việc trao đổi dữ liệu với thiết bị ngoại vi.
- CPU giao tiếp với các vi xử lý xuất nhập thông qua các cổng xuất nhập.
- Mỗi cổng xuất nhập cũng có một địa chỉ riêng.
- Mỗi vi xử lý xuất nhập có thể có nhiều địa chỉ cổng xuất nhập.

- Việc lựa chọn cổng xuất nhập cũng được thực hiện thông qua mạch giải mã địa chỉ xuất nhập.

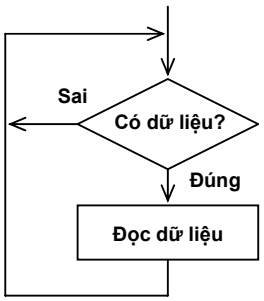
5.2. Phân loại :

- Thông thường người ta phân biệt các thiết bị ngoại vi theo chiều trao đổi dữ liệu. Màn hình, máy in, máy vẽ là các thiết bị xuất. Bàn phím, chuột, máy quét là các thiết bị nhập. Đĩa cứng, đĩa mềm là các thiết bị vừa xuất vừa nhập.
- Người ta còn phân biệt các thiết bị ngoại vi theo dạng dữ liệu giao tiếp với máy tính mà cụ thể là giữa *khối điều khiển xuất nhập* với *thiết bị ngoại vi*.
- Màn hình là thiết bị giao tiếp nối tiếp có điều chế thành tín hiệu RGB.
- Đĩa cứng, đĩa mềm, CD ROM là các thiết bị giao tiếp nối tiếp có mã hóa để tăng mật độ ghi dữ liệu.
- Bàn phím là thiết bị giao tiếp nối tiếp theo dạng đồng bộ.
- Chuột là các thiết bị giao tiếp nối tiếp theo dạng bất đồng bộ RS-232C.
- Máy in, máy vẽ là các thiết bị giao tiếp song song.

5.3. Các hoạt động xuất nhập đặc biệt :

a) Hoạt động kiểm tra trạng thái :

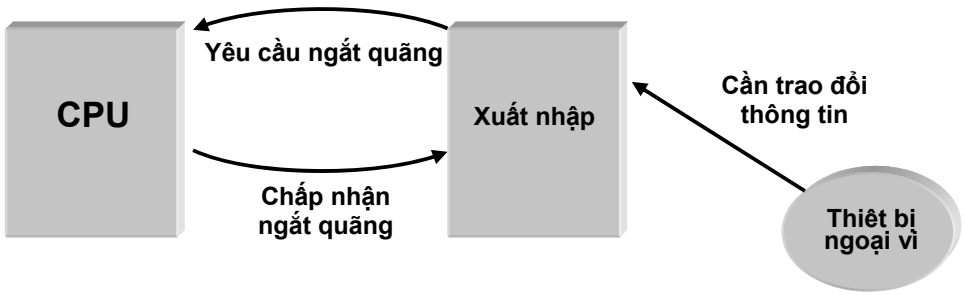
- Hoạt động kiểm tra trạng thái là hoạt động cần làm khi muốn nhập thông tin từ một cổng nhập.
- Mục đích của kiểm tra trạng thái là để biết khi nào thiết bị ngoại vi có dữ liệu sẵn sàng cho việc nhập.
- Thông thường các thiết bị ngoại vi đều là các thiết bị cơ nên tốc độ cung cấp thông tin không thể nhanh được. Do đó, khoảng thời gian mà CPU tiêu tốn cho các lần kiểm tra trạng thái mà không có dữ liệu là quá lớn. Chẳng hạn, có thể CPU ra kiểm tra trạng thái thiết bị ngoại vi cả trăm ngàn lần mới có thông tin một lần.
- Hoạt động này có thể tóm tắt như trong lưu đồ sau :



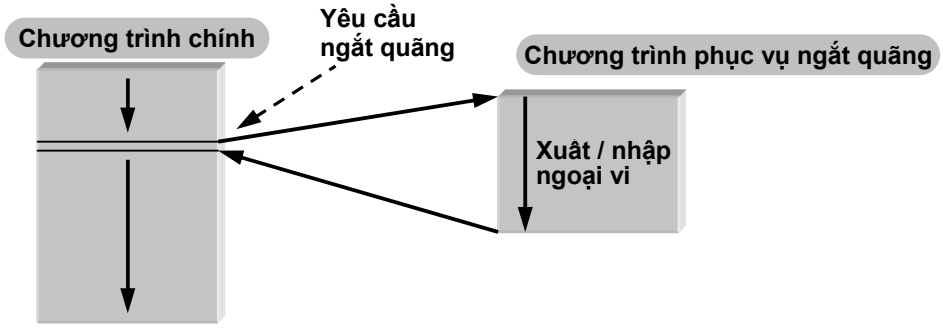
- CPU chạy trong chế độ kiểm tra trạng thái có hiệu suất rất thấp vì các khoảng thời gian không cần thiết do việc kiểm tra gây ra.
- Mặt khác người sử dụng cũng gặp khó khăn trong vấn đề lập trình vì phải luôn luôn canh chừng thiết bị nhập.

b) Cơ chế ngắt quãng (interrupt) :

- Cơ chế ngắt quãng được dùng với mục đích là tránh tối đa hoặc loại bỏ hẳn cơ chế kiểm tra trạng thái.
- Muốn thực hiện được điều đó, phần điều khiển xuất nhập của thiết bị nhập phải có khả năng thông báo cho CPU biết thời điểm cần trao đổi thông tin.
- Quá trình ngắt quãng bắt đầu khi thiết bị ngoại vi cần trao đổi thông tin với CPU. Thiết bị ngoại vi gởi tín hiệu báo cho phần điều khiển xuất nhập. Sau đó phần điều khiển xuất nhập gởi tiếp tín hiệu báo cho CPU biết thời điểm cần trao đổi thông tin. Tín hiệu đó được gọi là tín hiệu yêu cầu ngắt quãng INTR (interrupt request).



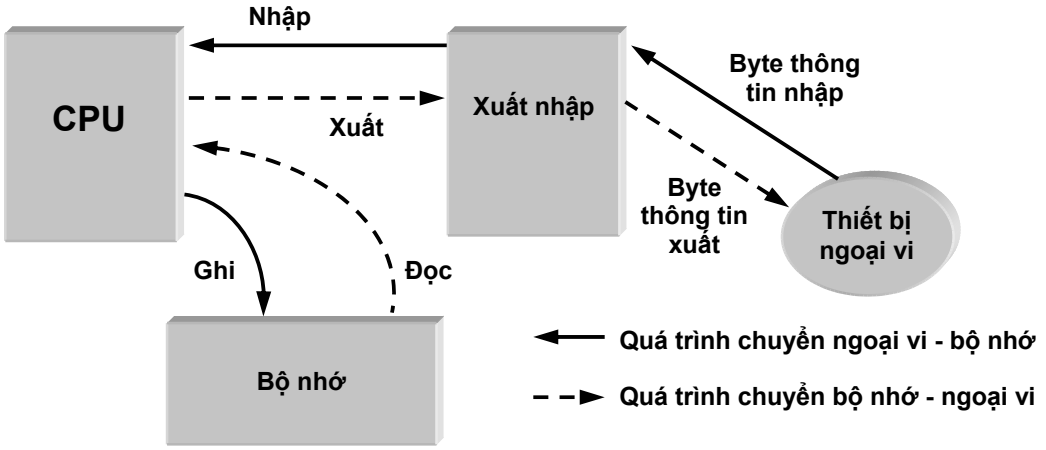
- Khi CPU đồng ý trao đổi thông tin với thiết bị ngoại vi, nó trả lời lại phần điều khiển xuất nhập bằng tín hiệu chấp nhận ngắt quãng INTA interrupt acknowledge đồng thời tạm ngưng việc chạy chương trình chính (chương trình chính xem như bị ngắt) và gọi sang một chương trình con đặc biệt gọi là chương trình phục vụ ngắt quãng.
- Trong chương trình phục vụ ngắt quãng, CPU phải giải quyết vấn đề trao đổi thông tin với thiết bị ngoại vi yêu cầu ngắt quãng. Và sau khi làm xong, chương trình phục vụ ngắt quãng sẽ trả điều khiển về cho chương trình chính. Lúc đó xem như chấm dứt quá trình ngắt quãng.



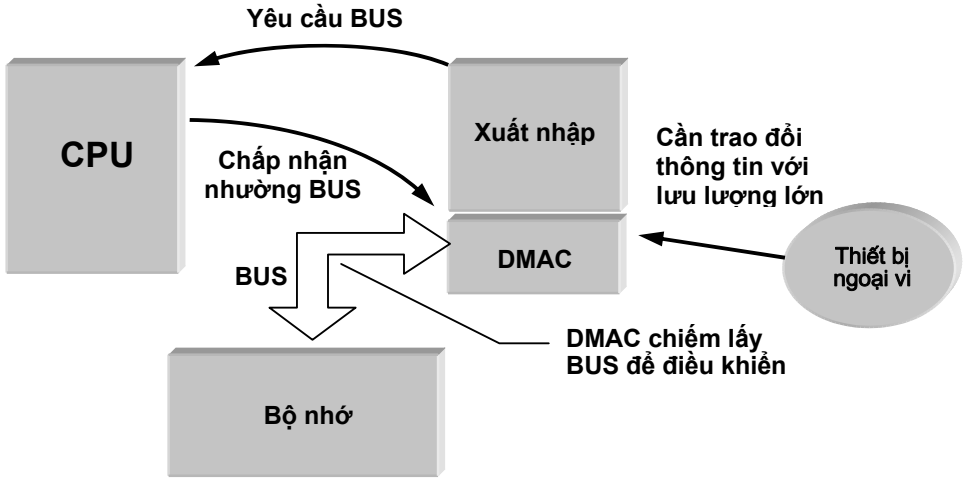
- Cơ chế ngắt quãng cho phép CPU trao đổi thông tin với thiết bị ngoại vi chỉ khi nào có yêu cầu mà thôi. Như vậy CPU sẽ không tiêu tốn thời gian cho các hoạt động kiểm tra trạng thái vô ích nữa (là lúc không có dữ liệu).
- Phần lớn các thiết bị nhập điều dùng cơ chế ngắt quãng như bàn phím, chuột, đồng hồ thời gian, . . .

c) Cơ chế DMA (Direct Memory Access) :

- Thông thường khi chương trình muốn chuyển một byte dữ liệu từ ngoại vi vào bộ nhớ, nó phải thực hiện thông qua CPU gồm một thao tác nhập và sau đó là thao tác ghi bộ nhớ. Hoặc ngược lại, khi muốn đưa thông tin từ bộ nhớ ra ngoại vi, CPU phải làm thao tác đọc bộ nhớ rồi xuất ra ngoại vi.



- Trong trường hợp đó, chức năng của CPU chỉ là chuyển dữ liệu mà không xử lý gì cả. Điều này nếu lặp đi lặp lại nhiều lần cũng sẽ làm lãng phí việc sử dụng CPU vì chức năng chủ yếu của CPU là xử lý dữ liệu.
- Để giải quyết vấn đề đó, người ta sử dụng một cơ chế đặc biệt gọi là DMA - truy xuất bộ nhớ trực tiếp.
- Trong hệ thống máy tính cơ chế DMA được thực hiện bởi một vi xử lý xuất nhập gọi là DMAC (DMA Controller).
- Cơ chế DMA giúp cho hệ thống rút ngắn thời gian trao đổi thông tin giữa thiết bị ngoại vi với bộ nhớ. Cơ chế này tiện dụng cho các thiết bị ngoại vi có lưu lượng thông tin trao đổi lớn (trao đổi một khối dữ liệu lớn trong một thời gian ngắn). Chẳng hạn như đĩa từ, CDROM, mạch lấy mẫu âm thanh.
- Ví dụ *Sound blaster card 16* có tần số lấy mẫu là 44100Hz, mỗi lần lấy mẫu được một giá trị số 16 bit (2 byte) và có 2 kênh nên cần nhập vào bộ nhớ 44100 × 2 × 2 = 150KB trong một giây.
- Quá trình DMA cũng bắt đầu khi có tín hiệu yêu cầu DMA từ thiết bị ngoại vi đến vi xử lý DMAC. DMAC tiếp tục gởi tín hiệu yêu cầu CPU nhường cho nó các tuyến địa chỉ, dữ liệu và điều khiển để nó thực hiện việc chuyển thông tin trực tiếp từ thiết bị ngoại vi vào bộ nhớ hay ngược lại.
- Khi CPU đồng ý nhường BUS, nó trả lời cho DMAC bằng tín hiệu chấp nhận nhường BUS và đồng thời tạm ngưng hoạt động (thực sự ngưng hoạt động lấy lệnh và thi hành lệnh), tự tách ra khỏi các tuyến hệ thống.
- Quá trình DMA kết thúc khi DMAC chuyển hết byte dữ liệu cuối cùng và trả các tuyến lại cho CPU. Lúc đó mọi hoạt động của hệ thống trở lại bình thường.



- Trong thực tế, việc sử dụng cơ chế DMA khá phức tạp và qua nhiều công đoạn vì vi xử lý xuất nhập DMAC không thi hành bất kỳ lệnh nào, chương trình nào để chuyển dữ liệu mà nó thực hiện hoàn toàn bằng phần cứng nên cần thiết phải ***có trước*** đầy đủ các thông tin về vị trí dữ liệu ngoài thiết bị ngoại vi, vị trí bộ nhớ chứa dữ liệu và số lượng byte cần chuyển.
- Các thông tin này phải được CPU nạp cho DMAC để nó giữ lại trong các thanh ghi bên trong DMAC trước khi có tín hiệu yêu cầu DMA từ ngoại vi.

5.4. Các thiết bị ngoại vi thông dụng :

a) Màn hình :

- Gồm hai phần : cạc điều khiển màn hình và màn hình.

- Điểm đặc trưng của cạc màn hình là độ phân giải và số màu thể hiện được.
- Độ phân giải được tính theo số điểm sáng (pixel) thể hiện một hình ảnh trên màn hình. Ví dụ độ phân giải 320 200 có nghĩa là hình ảnh sẽ được chia thành 320 điểm theo chiều ngang và 200 điểm theo chiều đứng.
- Độ phân giải càng lớn, hình ảnh thể hiện càng đẹp, càng mịn.
- Số màu thể hiện của cạc màn hình càng nhiều, cạc càng mạnh. Số màu của mỗi điểm sáng còn được tính theo số bit màu. Ví dụ 1 bit màu thể hiện được 2 màu, 2 bit màu thể hiện được 4 màu, 8 bit màu thể hiện được 356 màu, 16 bit màu thể hiện được 64K màu, và 24 bit màu thể hiện được 17.6M màu.
- Một số cạc màn hình cùng độ phân giải và số màu như sau :

Monochromes	:	320 200 2 màu
CGA	:	320 200 4 màu
EGA	:	640 480 16 màu
VGA 512K RAM	:	640 480 256 màu
	:	800×600×256 màu
	:	1024×768×16 màu
SVGA 1M RAM	:	640×480×17.6M màu
	:	800×600×64K màu
	:	1024×768×256 màu
SVGA 2M RAM	:	800×600×17.6M màu
	:	1024×768×64K màu
SVGA 4M RAM	:	1024×768×17.6M màu

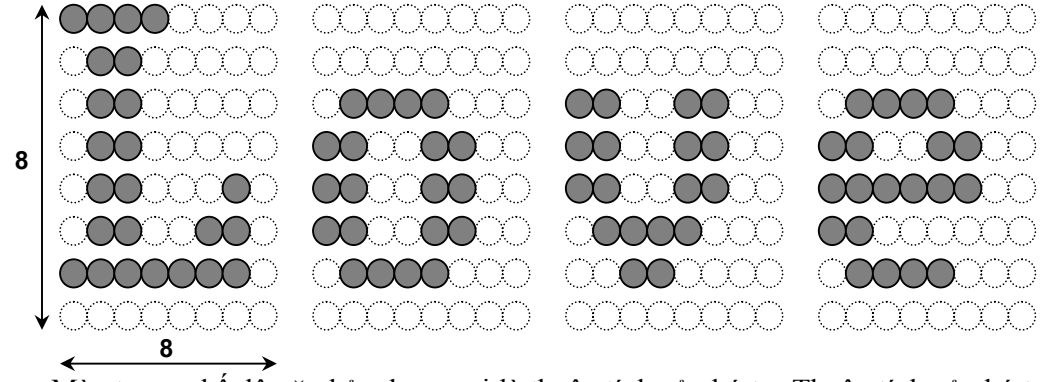
- Từ các thông số độ phân giải và số màu, ta tính ra được dung lượng bộ nhớ màn hình (Video RAM) cần thiết cho mỗi loại cạc. Ví dụ chế độ 800 600 64K cần :

  800*600*2 bytes = 960000 bytes nghĩa là cần 1M bytes AM màn hình.

- Màn hình thường được đặc trưng bằng các thông số sau :

  . Kich thước màn hình : 14" (Inches), 15", 17", 19", . . .
  . Kích thước điểm sáng : .31 mm, .29 mm, .22 mm, . . .
  . Các tần số quét :
  + Ngang (dòng) : 40 KHz, 70 KHz, 90 KHz, . . .
  + Đứng (mành) : 50 Hz ÷ 160 Hz

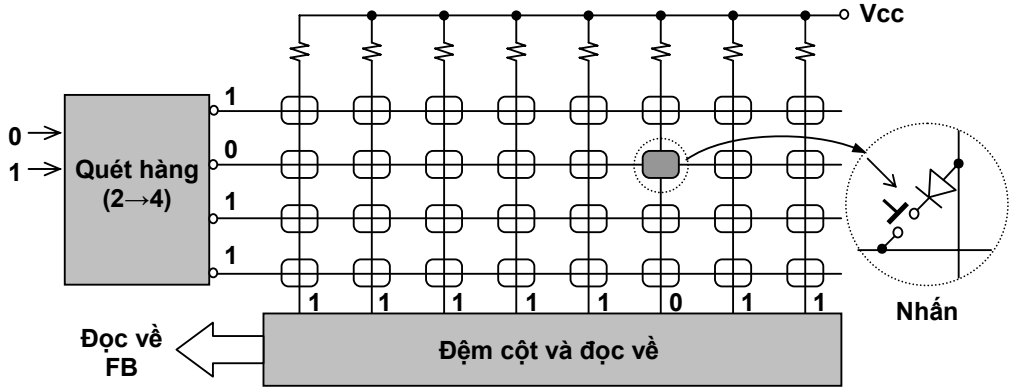
- Màn hình có kích thước màn hình càng lớn, kích thước điểm sáng càng nhỏ và các tần số quét càng lớn thì càng tốt, càng đắt.
- Việc thể hiện thông tin trên màn hình được cạc màn hình chia ra thành hai loại : văn bản (text) và đồ họa (graphics).
- Trong chế độ văn bản, các ký tự được thể hiện theo ma trận điểm sáng. Kích thước ma trận có thể là 8×8 , 14×8 hay 16×8 (16 điểm theo hàng dọc và 8 điểm theo hàng ngang). Sau đây là một ví dụ về ma trận điểm 8×8.



- Màu trong chế độ văn bản được gọi là thuộc tính của ký tự. Thuộc tính của ký tự gồm các thành phần : màu chữ (8 màu), màu nền (8 màu), thuộc tính sáng chói và thuộc tính nhấp nháy. Các thuộc tính của ký tự được lưu trữ trong 1 byte thuộc tính.
- Toàn bộ màn hình được chia thành 25 hàng, mỗi hàng có 80 cột, và mỗi cột tương ứng một ký tự (80×25).
- Trong chế độ đồ họa, đơn vị thể hiện là điểm sáng (pixel). Tọa độ một điểm sáng được xác định tùy theo độ phân giải màn hình đã chọn.

b) Bàn phím :

- Bàn phím là thiết bị dùng để nhập dữ liệu bằng tay nên số phím phải cho phép nhập vào các ký tự ASCII cơ bản như chữ, số và các ký hiệu khác.
- Các bàn phím thường có cấu tạo theo ma trận và các hàng được kiểm tra liên tục để phát hiện ra sự nhấn phím. Quá trình đó gọi là quét bàn phím.

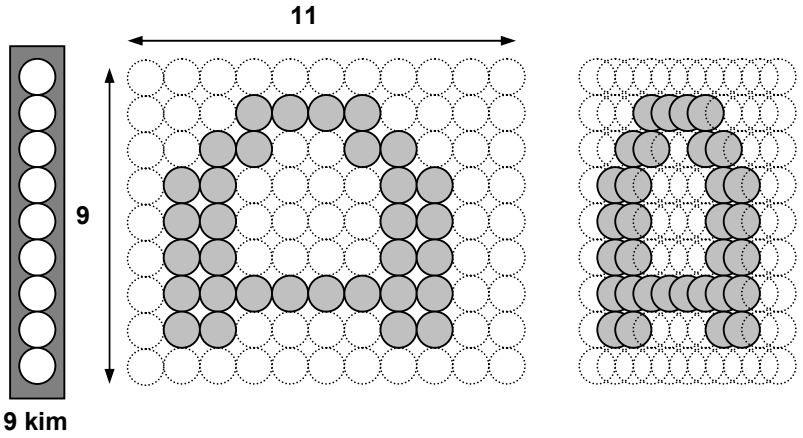


- Để quét bàn phím, người ta dùng một mạch quét hàng cho phép các hàng lần lượt được chọn (xuống 0). Thực chất mạch quét hàng chính là một mạch giải mã mà đầu ra chỉ cho một hàng xuống 0, còn các hàng còn lại lên 1.
- Dữ liệu đọc về từ mạch đệm cột cho biết phím nào trên hàng được chọn đã nhấn. Ví dụ dữ liệu FF cho biết không có phím nào nhấn, dữ liệu FB cho biết phím ở cột thứ 6 đang nhấn, ... Hoặc có thể xem mỗi bit của dữ liệu đọc về là trạng thái nhấn hoặc không nhấn của phím trên cột tương ứng với vị trí bit.
- Với cách tổ chức như trên, dữ liệu quét (là dữ liệu xuất ra mạch quét) và dữ liệu đọc về cho biết vị trí của phím được nhấn. Các dữ liệu đó không thể dùng để gởi về máy tính được mà phải qua một bước chuyển đổi thành mã quét (scan code) của riêng từng phím ở dạng nối tiếp.
- Việc chuyển đổi này do một vi xử lý riêng của bàn phím thực hiện.
- Mỗi phím có một mã quét riêng. Ví dụ phím ESC có mã quét là 01, phím (!-1) có mã quét là 02, phím A có mã quét là 1E, . . .

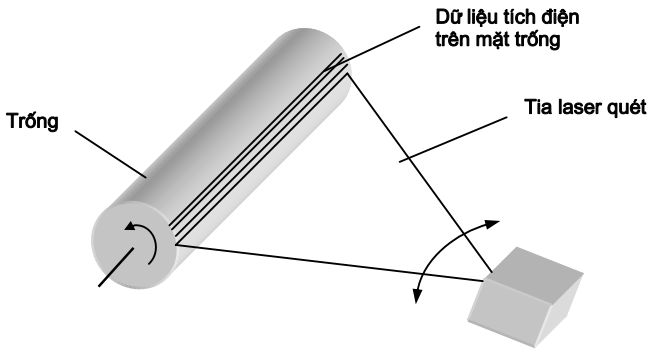
- Việc nhận mã quét và đổi ra mã ASCII tương ứng của phím là do bên máy tính xử lý.

c) Máy in :

- Có 3 loại thường dùng : máy in kim, máy in phun mực và máy in laser.
- Mỗi loại máy in trên có thể là máy in trắng đen hoặc màu.
- Điểm đặc trưng chung của các loại máy in trên là in theo dạng điểm và chất lượng in là do độ phân giải quyết định.
- Máy in kim có độ phân giải 72 dpi (dots per inch), máy in phun mực 150 dpi và máy in laser 300 dpi, 600 dpi hay 1200 dpi.
- Máy in kim có thể có 9 kim (các máy in Epson FX, LX) hay 24 kim (các máy in Epson FQ, LQ) được bố trí theo hàng đứng và ký tự được in theo dạng ma trận điểm với kích thước 9×11 (9 hàng, 11 cột).



- Đầu kim in ma trận điểm theo từng cột. Mỗi lần in xong một cột, đầu kim được kéo đi một khoảng bằng nửa kim để in cột kế tiếp. Do đó các điểm có thể chồng chập lên nhau phân nửa.
- Khi in mỗi cột, vị trí kim nào tương ứng với điểm màu đen sẽ được điều khiển để đóng lên một băng mực và in lên trên giấy thành một điểm. Các vị trí kim tương ứng với điểm trắng sẽ không hoạt động nên trên mặt giấy sẽ không có vết.

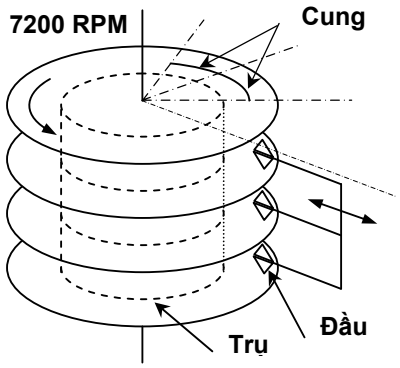


- Nguyên tắc hoạt động của máy in laser hoàn toàn khác. Phần chủ yếu của máy in laser là một trống hình trụ có khả năng tích điện theo từng điểm dưới tác dụng của tia laser.
- Các thông tin cần in được một vi xử lý điều khiển tia laser quét qua trống dọc theo trục của nó để tích điện. Độ mảnh của tia laser quyết định độ nét của máy in (chính là số điểm chia trên mỗi inch).

- Sau khi tích điện xong, trống sẽ lăn qua hộp mực để hút các hạt mực bột dính vào mặt trống. Điểm nào có điện tích thì hút mực, điểm nào không tích điện sẽ không hút.
- Mặt trống dính mực theo dạng của các thông tin cần in được lăn qua một mặt giấy để cho mực bám lên giấy.
- Cuối cùng giấy có mực được sấy nóng lên để cho mực bột chảy ra và thấm vào giấy.

d) Đĩa từ :

- Đĩa từ là thiết bị lưu trữ thông tin bằng cách từ hóa các hạt từ trên một mặt đĩa theo một qui luật đã được định trước.
- Đĩa từ được chia ra 2 loại : đĩa cứng có dung lượng lớn (từ vài chục MB cho đến hàng GB), cố định, truy xuất nhanh; đĩa mềm có dung lượng nhỏ (1.2MB, 1.44MB), dễ dàng mang đi, truy xuất chậm.
- Đĩa từ được phân chia thành các phần nhỏ để lưu trữ thông tin với các thông số vật lý như sau :
  - Trụ (Cylinder): các trụ đồng trục với nhau. Người ta có thể chia đĩa ra thành nhiều trụ. Chẳng hạn, đĩa mềm có 80 trụ, đĩa cứng có thể lên đến 1024 hoặc hơn nữa. Đường cắt giữa trụ với mặt đĩa còn được gọi là vết (track). Tuy vậy hai khái niệm trụ và vết thường được dùng thay cho nhau cũng được.



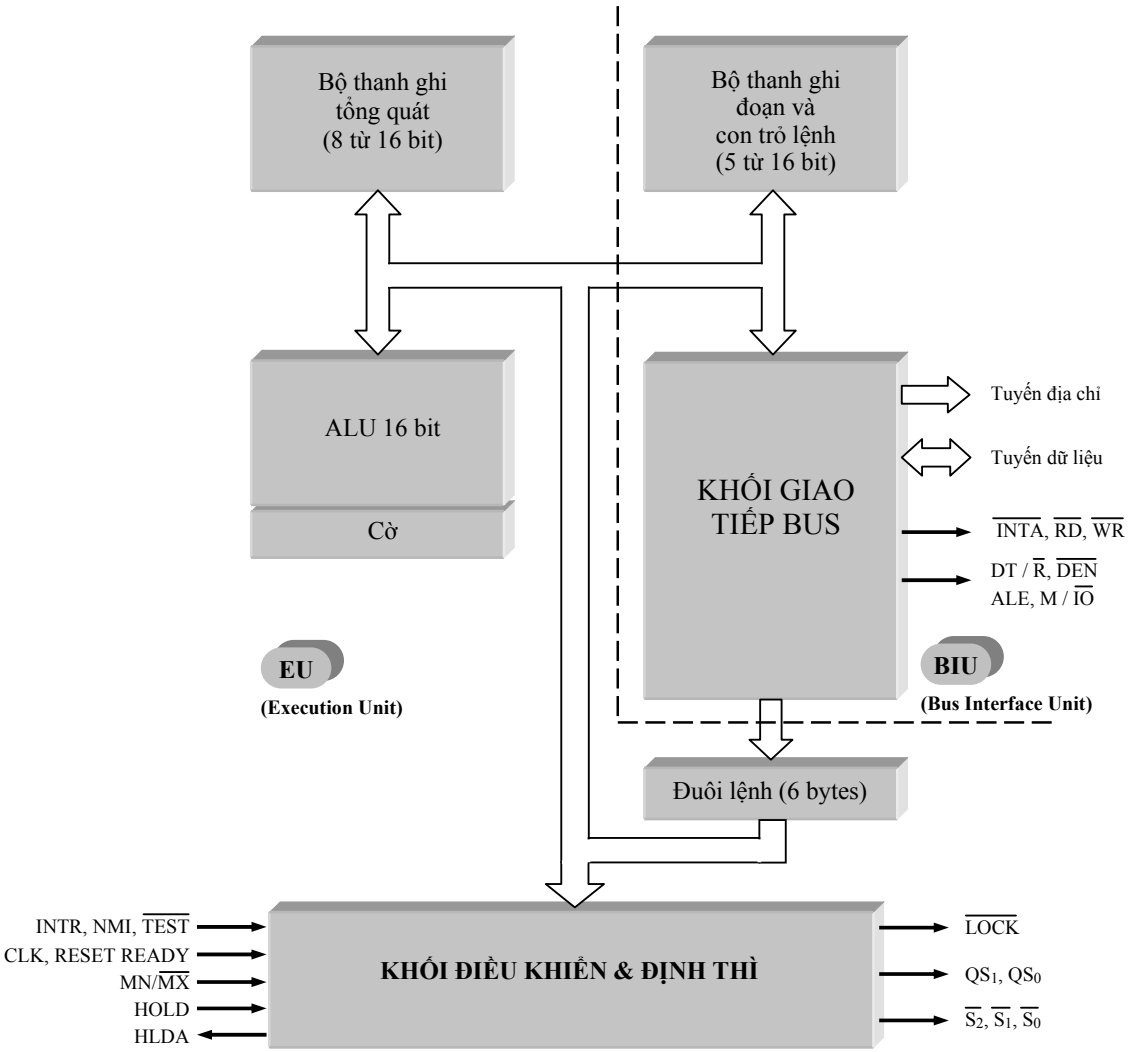
  - Đầu (Head): là đầu từ dùng để đọc và ghi. Thường mỗi đầu từ chuyên dùng để đọc một mặt đĩa duy nhất nên khái niệm số đầu cũng chính là số mặt đĩa. Các đầu từ được điều khiển bởi một động cơ bước và có thể dịch chuyển vào ra theo chiều hướng tâm. Khoảng cách giữa hai bước dịch chuyển đầu từ qui định khoảng cách giữa hai trụ hay nói khác đi, số trụ phân chia lệ thuộc vào số bước dịch chuyển của đầu từ.
  - Mẫu tin (**R**ecord) hay cung (sector): trên mỗi vết, người ta chia ra thành các cung tròn bằng nhau chính là mẫu tin. Thường mỗi mẫu tin có khả năng giữ được 512 bytes dữ liệu (0.5 KB).
- Như vậy có thể xem bộ ba thông số (C,H,R) như là tọa độ vật lý của mỗi cung trên đĩa.
- Trong thực tế, việc lưu trữ thông tin trên đĩa còn lệ thuộc vào cách quản lý đĩa của riêng từng hệ điều hành.

CẤU TRÚC VI XỬ LÝ 8086/8088

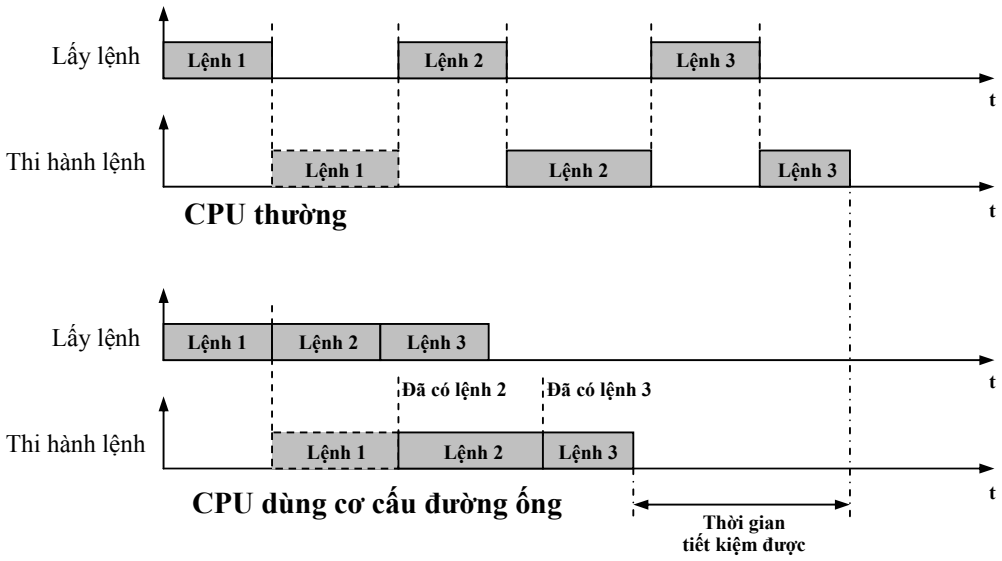
I. CẤU TRÚC BÊN TRONG CỦA VI XỬ LÝ 8086/8088 :

1. Sơ đồ khối :

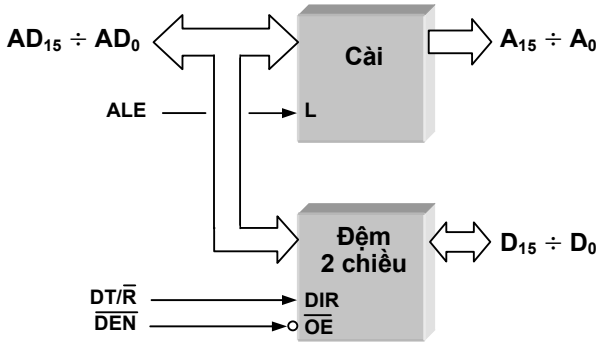
- Vi xử lý INTEL 8086 / 8088 là các vi xử lý 16 bit.
- Các phép toán xử lý bên trong CPU là phép toán thực hiện trên số nhị phân 8 và 16 bit.
- Tuyến địa chỉ có 20 đường nên có thể quản lý lên đến 1 MB bộ nhớ (tầm địa chỉ từ 00000h đến FFFFFh).
- Tuyến dữ liệu của 8086 có 16 bit.
- Tuyến dữ liệu của 8088 có 8 bit. Đây là một cải tiến về mặt thương mại so với 8086 để việc giao tiếp với bộ nhớ và xuất nhập đơn giản hơn, dễ dàng thiết kế hơn.



- Cho phép sử dụng hệ thống ngắt quãng và cơ chế DMA (Direct Memory Access).
- Sơ đồ khối vi xử lý 8086 gồm hai phần chính : khối giao tiếp Bus (BIU:Bus interface unit) và khối thực thi (EU:Execution unit).
- Khối BIU chịu trách nhiệm lấy lệnh và giao tiếp ra bên ngoài để điều khiển bộ nhớ và xuất nhập.
- Khối EU có nhiệm vụ thi hành lệnh, định thì, kiểm tra các tín hiệu trạng thái, các tín hiệu yêu cầu ngắt quãng, cơ chế DMA, tín hiệu RESET, tín hiệu READY.
- Các lệnh trong bộ nhớ được khối BIU lấy vào liên tục và cất trong đuôi lệnh (có chiều dài 6 byte đối với 8086 hoặc 4 byte đối với 8088). Sau đó khối EU lấy lệnh từ đuôi lệnh ra để giải mã và thi hành.
- Hoạt động của hai khối BIU và EU diễn ra độc lập với nhau nên quá trình lấy lệnh và thi hành lệnh được vi xử lý thực hiện đồng thời theo cơ cấu đường ống (pipeline). Điều này tuy không làm tăng tốc độ xử lý của CPU (giới hạn bởi tần số xung đồng bộ) nhưng làm giảm bớt thời gian thi hành của cả chương trình.
- Hình sau minh họa về sự phân phối thời gian cho hai quá trình lấy lệnh và thi hành lệnh của CPU bình thường và của CPU dùng cơ cấu đường ống.



- Tuyến địa chỉ dữ liệu dùng chung $AD_{15} \div AD_0$ cần có tín hiệu điều khiển ALE để phân biệt lúc nào là địa chỉ lúc nào là dữ liệu. Chỉ khi ALE = 1, tín hiệu trên tuyến chung được xem là địa chỉ. Trường hợp còn lại xem là tuyến dữ liệu.
- Thường thì phải nhờ một mạch cài địa chỉ và một mạch đệm 2 chiều để tách tuyến chung ra thành hai tuyến phân biệt.



2. Bộ thanh ghi :

2.1 Bộ thanh ghi đa dụng :

- Gồm 8 thanh ghi 16 bit.
- Các thanh ghi AX, BX, CX, DX có thể dùng phân nửa như các thanh ghi 8 bit AH, AL, BH, BL, CH, CL, DH, DL.
- Thanh ghi AH là nửa cao của thanh ghi AX. Thanh ghi AL là nửa thấp của thanh ghi AX. Chẳng hạn nếu AX= 1234h thì AH=12h và AL=34h.
- AX là thanh ghi bộ tích lũy 16 bit (ACC).
- AL là thanh ghi bộ tích lũy 8 bit.
- BX là thanh ghi nền (base register).
- CX là thanh ghi bộ đếm (counter).
- DX là thanh ghi dữ liệu (data).
- SI là thanh ghi chỉ số nguồn (source index).
- DI là thanh ghi chỉ số đích (destination index).
- BP là thanh ghi con trỏ nền (base pointer).
- SP là thanh ghi con trỏ chồng (stack pointer).

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

2.2 Bộ thanh đoạn và con trỏ lệnh :

- Gồm 4 thanh ghi đoạn 16 bit dùng để quản lý bộ nhớ theo phương pháp phân đoạn.
- CS là thanh ghi đoạn chương trình (code segment).
- DS là thanh ghi đoạn dữ liệu (data segment).
- SS là thanh ghi đoạn chồng (stack segment).
- ES là thanh ghi đoạn mở rộng (extra segment).
- IP là thanh ghi con trỏ lệnh (instruction pointer).

CS
DS
SS
ES
IP

2.3 Thanh ghi trạng thái :

×	×	×	×	OF	DF	IF	TF	SF	ZF	×	AF	×	PF	×	CF

- SF là cờ dấu (sign flag).
- CF là cờ nhớ (carry flag).
- ZF là cờ không (zero flag).
- OF là cờ tràn (overflow flag).
- AF là *cờ trung gian* (auxiliary flag) hay còn gọi là *cờ nhớ nửa* (half-carry flag).
  AF = 0 khi không có sự tràn về dung lượng 4 bit.
  AF = 1 khi có sự tràn về dung lượng 4 bit.
  Ví dụ : phép toán 00001001 + 00000111 = 00010000 sẽ lập cờ AF lên 1.
- Cờ AF thường được dùng trong các phép toán BCD (là các số dùng hệ nhị phân 4 bit để biểu diễn số thập phân từ 0 ÷ 9).
- PF là *cờ kiểm tra chẵn lẻ* (parity flag).
  PF = 1 nếu số bit 1 của kết quả là số chẵn.
  PF = 0 nếu số bit 1 của kết quả là số lẻ.
  Ví dụ : sau khi thực hiện (00000101 AND 00000101) thì PF = 1.
- DF là *cờ định hướng* (direction flag).
  DF = 0 : định hướng giảm địa chỉ cho các lệnh xử lý chuỗi.
  DF = 1 : định hướng tăng địa chỉ cho các lệnh xử lý chuỗi.
- IF là *cờ ngắt quãng* (interrupt enable flag).
  IF = 0 : cấm ngắt cứng INTR.
  IF = 1 : cho phép ngắt cứng INTR.

- TF là *cờ bẫy* (trap flag). Dùng để chạy từng bước khi cần kiểm tra hoạt động của CPU.

II. QUẢN LÝ BỘ NHỚ CỦA VI XỬ LÝ 8086/8088 :

1. Phân đoạn và phân loại địa chỉ :

1.1. Sự phân đoạn bộ nhớ :

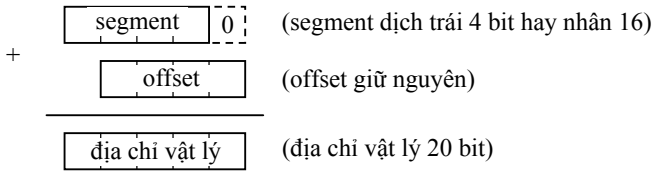
- CPU 8086 dùng phương pháp phân đoạn bộ nhớ để quản lý bộ nhớ 1MB của nó.
- Địa chỉ 20 bit của bộ nhớ 1MB không thể chứa đủ trong các thanh ghi 16 bit của CPU 8086 vì vậy bộ nhớ 1 MB được chia ra thành các đoạn (segment) 64KB.
- Địa chỉ trong các đoạn 64KB chỉ có 16 bit nên CPU 8086 dễ dàng xử lý bằng các thanh ghi của nó.
- Như vậy phương pháp phân đoạn bộ nhớ là cách dùng các thanh ghi 16 bit để biểu diễn cho địa chỉ 20 bit.

1.2. Địa chỉ vật lý và địa chỉ luận lý :

- Địa chỉ 20 bit được gọi là địa chỉ vật lý. Địa chỉ vật lý được dùng trong thiết kế các mạch giải mã địa chỉ cho bộ nhớ và xuất nhập.
- Ngược lại, trong lập trình, địa chỉ vật lý không thể dùng được mà nó được thay thế bằng địa chỉ luận lý.
- Địa chỉ luận lý là địa chỉ gồm có hai thành phần : địa chỉ đoạn (segment) và địa chỉ trong đoạn (offset).
- Mỗi địa chỉ thành phần chỉ có 16 bit và được viết theo cách sau :

SEGMENT:OFFSET

- Segment và offset là các số hệ 16.
- Cách tính địa chỉ vật lý từ địa chỉ luận lý như sau :



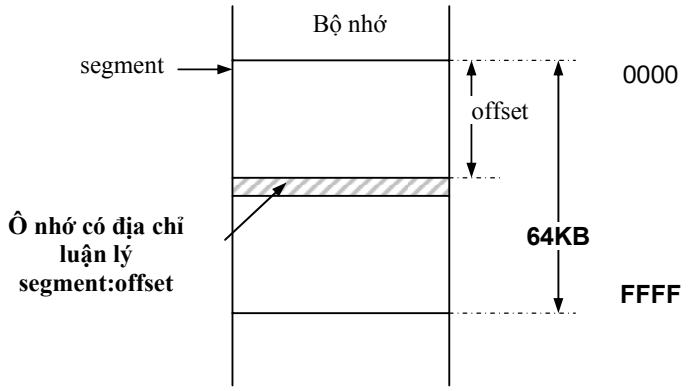
Hoặc theo công thức :

địa chỉ vật lý = (segment x 16) + offset

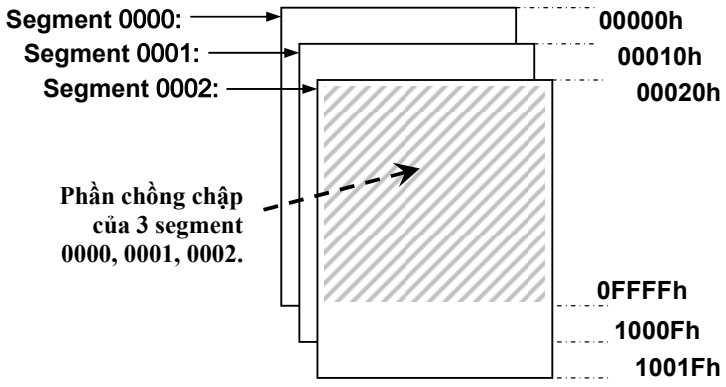
Ví dụ : tính địa chỉ vật lý tương ứng với địa chỉ luận lý B001:1234

địa chỉ vật lý = B0010h + 1234 = B1244h

- Địa chỉ segment còn được gọi là địa chỉ nền của đoạn. Nó cho biết điểm bắt đầu của đoạn trong bộ nhớ.
- Trong khi đó, địa chỉ offset thể hiện khoảng cách kể từ đầu đoạn của ô nhớ cần tham khảo. Do offset có 16 bit nên chiều dài tối đa của một đoạn là 64K. Trong mỗi đoạn, ô nhớ đầu tiên có offset là 0000h và ô nhớ cuối cùng có offset là FFFFh.



- Mỗi ô nhớ chỉ có duy nhất một địa chỉ vật lý nhưng có thể có nhiều địa chỉ luận lý. Chẳng hạn các địa chỉ luận lý 1234:1234, 1334:0234, 1304:0534, ... đều có chung địa chỉ vật lý 13574h.
- Để hiểu rõ tại sao, ta lần lượt xem quan hệ giữa địa chỉ vật lý với các thành phần segment và offset.
- Với địa chỉ luận lý 0000:0000 ta có địa chỉ vật lý là 00000h.
- Bây giờ ta giữ nguyên phần segment và tăng phần offset lên 1 thành ra địa chỉ luận lý 0000:0001. Địa chỉ vật lý tương ứng là 00001h.
- Tương tự với địa chỉ luận lý 0000:0002 ta có địa chỉ vật lý 00002h.
- Ta nhận thấy khi offset tăng 1 đơn vị thì địa chỉ vật lý tăng 1 địa chỉ hoặc là tăng 1 byte. Như vậy ta có thể xem đơn vị của offset là byte.
- Bây giờ ta làm lại quá trình trên nhưng không tăng offset nữa mà tăng phần segment. Ta có :
  địa chỉ luận lý 0001:0000 tương ứng với địa chỉ vật lý 00010h.
  địa chỉ luận lý 0002:0000 tương ứng với địa chỉ vật lý 00020h.
- Ta nhận thấy khi segment tăng 1 đơn vị thì địa chỉ vật lý tăng 10h địa chỉ hoặc là tăng 16 byte. Người ta gọi đơn vị của segment là paragraph.
  1 paragraph = 16 bytes
- Điều này cũng có thể được giải thích là do cách tính địa chỉ vật lý từ địa chỉ luận lý đã nêu ở trên.



- Từ các địa chỉ vật lý tính ra ở trên, ta thấy segment 0000 nằm ở đầu vùng nhớ nhưng segment 0001 bắt đầu cách đầu vùng nhớ chỉ có 16 bytes, segment 0002 bắt đầu cách đầu vùng nhớ 32 bytes . . .

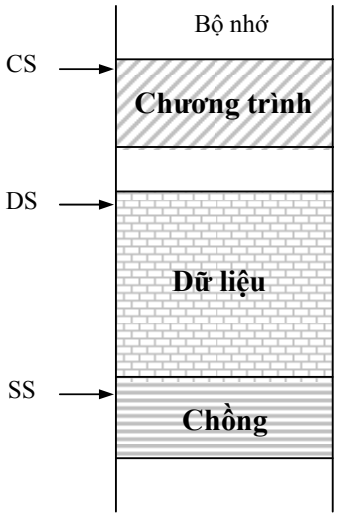
- Phần chồng chập của cả ba segment 0000, 0001 và 0002 trên hình vẽ là vùng bộ nhớ mà bất kỳ ô nhớ nào nằm trong đó (địa chỉ vật lý từ 00020h đến 0FFFFh) đều có thể có địa chỉ luận lý tương ứng trong cả 3 segment. Chẳng hạn ô nhớ có địa chỉ vật lý 0002Dh sẽ có địa chỉ luận lý trong segment 0000 là 0000:002D, trong segment 0001 là 0001:001D và trong segment 0002 là 0002:000D.
- Như vậy nếu vùng bộ nhớ nào càng có nhiều segment chồng chập lên nhau thì các ô nhớ trong đó càng có nhiều địa chỉ luận lý (một ô nhớ có ít nhất 1 địa chỉ luận lý và nhiều nhất là 65536/16=4096 địa chỉ luận lý).

2. Địa chỉ luận lý và các thanh ghi :

- Để tham khảo đến bộ nhớ trong chương trình, vi xử lý 8086 cho phép sử dụng các địa chỉ luận lý một cách trực tiếp hoặc thông qua các thanh ghi của nó.
- Các thanh ghi đoạn dùng để chứa địa chỉ đoạn segment.
- Các thanh ghi tổng quát dùng để chứa địa chỉ trong đoạn offset.
- Để tham khảo đến địa chỉ luận lý có segment trong thanh ghi DS, offset trong thanh ghi BX, ta viết DS:BX. Nếu lúc tham khảo, DS=2000h, BX=12A9h thì địa chỉ luận lý DS:BX chính là tham khảo đến ô nhớ 2000:12A9.
- Trong cách sử dụng địa chỉ luận lý thông qua các thanh ghi của vi xử lý 8086, có một số cặp thanh ghi luôn luôn phải dùng chung với nhau một cách bắt buộc như sau :

  CS:IP : *lấy lệnh (địa chỉ lệnh sắp thi hành).*
  SS:SP : *địa chỉ đỉnh chồng.*
  SS:BP : *thông số trong chồng (dùng cho chương trình con).*
  DS:SI : *địa chỉ chuỗi nguồn (chỉ có ý nghĩa trong các lệnh xử lý chuỗi).*
  ES:DI : *địa chỉ chuỗi đích (chỉ có ý nghĩa trong các lệnh xử lý chuỗi).*

- Chương trình mà vi xử lý 8086 thi hành thường có 3 đoạn : đoạn chương trình có địa chỉ trong thanh ghi CS, đoạn dữ liệu có địa chỉ trong thanh ghi DS và đoạn chồng có địa chỉ trong thanh ghi SS.

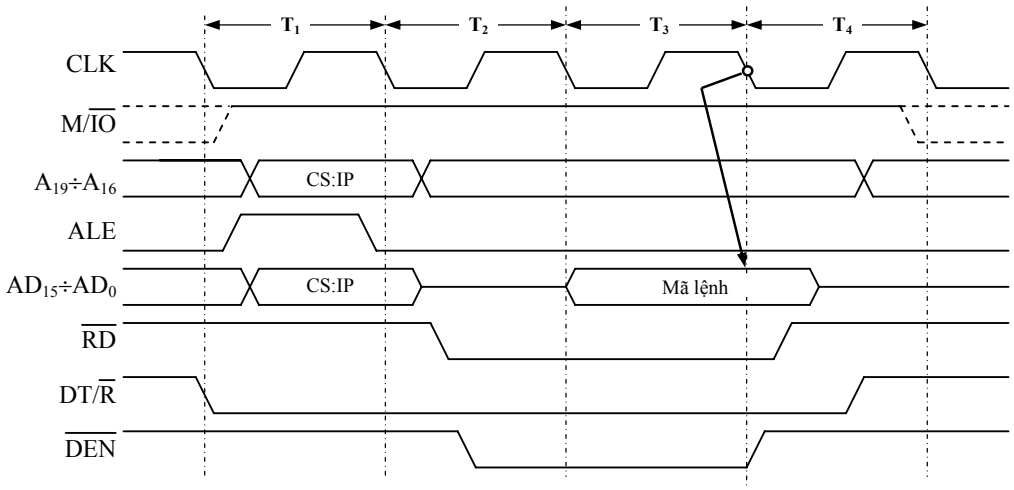


II. CÁC HOẠT ĐỘNG CHÍNH CỦA VI XỬ LÝ 8086/8088 :

1. Lấy lệnh :

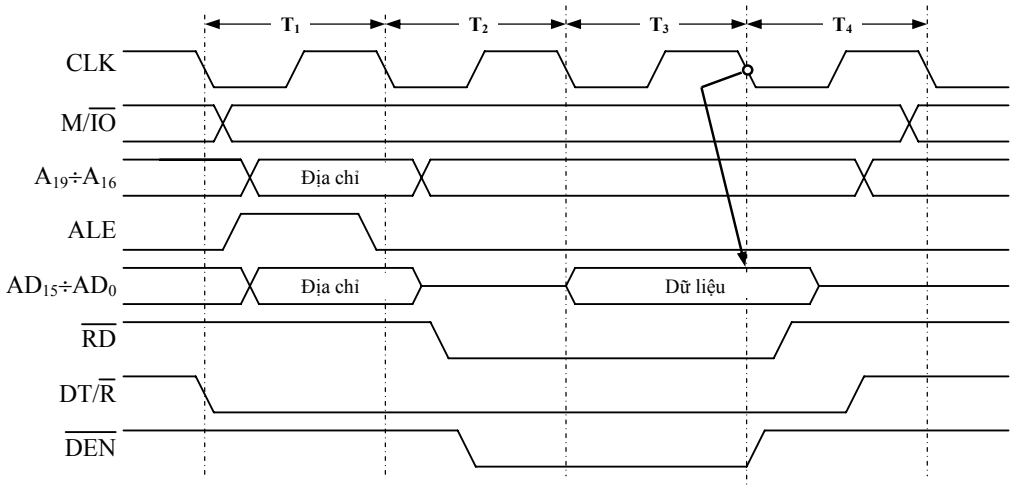
- Thực hiện bằng chu kỳ máy lấy lệnh kéo dài trong 4T (chu kỳ xung clock).
- Thực chất của hoạt động lấy lệnh là hoạt động đọc bộ nhớ.

- Để mở đầu cho hoạt động lấy lệnh, địa chỉ lệnh CS:IP được đổi thành địa chỉ vật lý và được đưa lên tuyến địa chỉ 20 bit kèm theo tín hiệu cho phép cài địa chỉ ALE (Address latch enable) trong T1 .
- Các tín hiệu điều khiển được đưa ra trong T2 và T3 gồm có :
  M/$\overline{IO}$=1 $\overline{RD}$=0 $\overline{DEN}$=0 DT/$\overline{R}$=0
- Sau đó, CPU sẽ đọc mã lệnh từ tuyến dữ liệu vào đầu T4.
- Thường các chu kỳ máy của vi xử lý được cung cấp trong các sổ tay tra cứu dưới dạng các giản đồ xung như sau :



2. <u>Đọc bộ nhớ</u> :

- Thực hiện bằng chu kỳ máy đọc bộ nhớ kéo dài trong 4T.
- Trong T1, địa chỉ bộ nhớ được đổi thành địa chỉ vật lý và được đưa lên tuyến địa chỉ 20 bit kèm theo tín hiệu cho phép cài địa chỉ ALE.
- Các tín hiệu điều khiển được đưa ra trong T2 và T3 gồm có :
  M/$\overline{IO}$=1 $\overline{RD}$=0 $\overline{DEN}$=0 DT/$\overline{R}$=0
- Sau đó, CPU sẽ đọc mã lệnh từ tuyến dữ liệu vào đầu T_4.

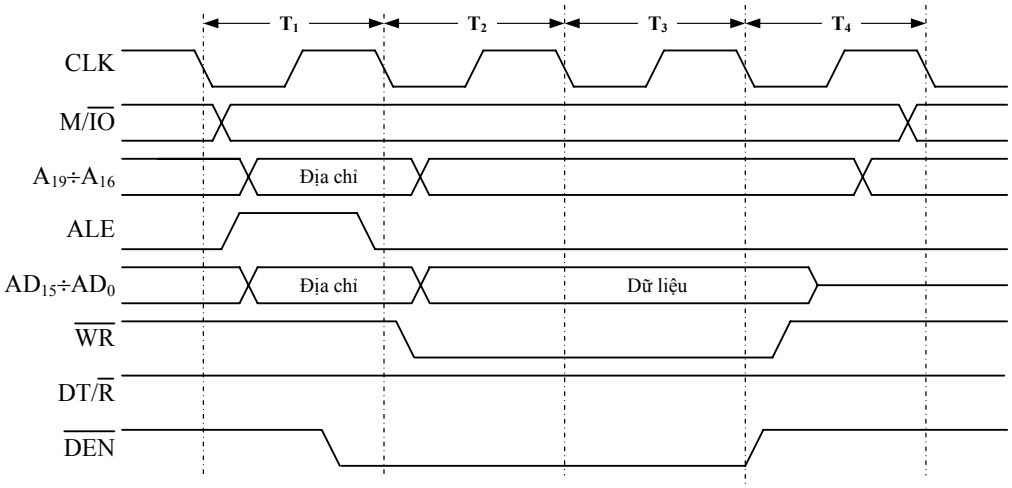


- Giản đồ xung của chu kỳ máy đọc bộ nhớ ở trên cũng có thể dùng chung cho chu kỳ máy nhập.

- Trong trường hợp bộ nhớ có thời gian truy xuất bộ nhớ quá lớn, không thể đưa dữ liệu ra đúng vào thời điểm CPU lấy dữ liệu, người ta phải thiết kế một mạch điều khiển để phát ra tín hiệu READY=0 nhằm kéo dài chu kỳ đọc ra thêm 1 hay nhiều chu kỳ xung clock (các chu kỳ kéo dài thêm gọi là chu kỳ đợi TW) trước khi chuyển sang T_4.

3. <u>Ghi bộ nhớ</u> :

- Thực hiện bằng chu kỳ máy ghi bộ nhớ kéo dài trong 4T.
- Trong T1, địa chỉ bộ nhớ được đổi thành địa chỉ vật lý và được đưa lên tuyến địa chỉ 20 bit kèm theo tín hiệu cho phép cài địa chỉ ALE.
- Dữ liệu cần ghi vào bộ nhớ được đưa ra tuyến dữ liệu trong T2 và T3.
- Các tín hiệu điều khiển được đưa ra trong T2 và T3 gồm có :
  M/$\overline{IO}$=1 $\overline{WR}$=0 $\overline{DEN}$=0 DT/$\overline{R}$=1
- Sau đó, CPU sẽ chờ cho đến hết T4 và kết thúc chu kỳ ghi.
- Giản đồ xung của chu kỳ máy ghi bộ nhớ và chu kỳ máy xuất như sau :



4. <u>Nhập</u> :

- Thực hiện bằng chu kỳ máy nhập kéo dài trong 4T.
- Trong T1, địa chỉ cổng xuất nhập 16 bit được đưa lên tuyến địa chỉ kèm theo tín hiệu cho phép cài địa chỉ ALE.
- Các tín hiệu điều khiển được đưa ra trong T2 và T3 gồm có :
  M/$\overline{IO}$=0 $\overline{RD}$=0 $\overline{DEN}$=0 DT/$\overline{R}$=0
- Sau đó, CPU sẽ đọc mã lệnh từ tuyến dữ liệu vào đầu T4.
- Giản đồ xung giống như của chu kỳ máy đọc bộ nhớ, chỉ có khác chỗ tín hiệu M/$\overline{IO}$ ở mức 0.

5. <u>Xuất</u> :

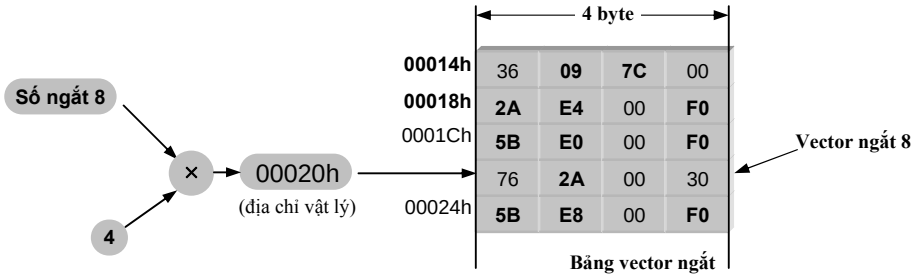
- Thực hiện bằng chu kỳ máy xuất kéo dài trong 4T.
- Trong T1, địa chỉ cổng xuất nhập 16 bit được đưa lên tuyến địa chỉ kèm theo tín hiệu cho phép cài địa chỉ ALE.
- Dữ liệu cần xuất được đưa ra tuyến dữ liệu trong T2 và T3.
- Các tín hiệu điều khiển được đưa ra trong T2 và T3 gồm có :
  M/$\overline{IO}$=0 $\overline{WR}$=0 $\overline{DEN}$=0 DT/$\overline{R}$=1
- Sau đó, CPU sẽ chờ cho hết T4 và kết thúc chu kỳ xuất.
- Giản đồ xung tương tự như của chu kỳ máy ghi bộ nhớ, chỉ khác tín hiệu M/$\overline{IO}$ ở mức 0.

6. Đáp ứng ngắt quãng :

- Vi xử lý 8086 sử dụng 3 loại ngắt quãng :
  . Ngắt hệ thống : do CPU phát ra khi có một lỗi nghiêm trọng xảy ra trong quá trình hoạt động của nó. Chẳng hạn như chia cho số 0, điện áp nguồn cung cấp giảm thấp, chia tràn . . .
  . Ngắt cứng : do thiết bị ngoại vi gây ra khi cần trao đổi thông tin với CPU. Đặc trưng của ngắt cứng là tín hiệu yêu cầu ngắt quãng INTR.
  . Ngắt mềm : do thi hành lệnh INT trong chương trình. Thực chất của ngắt mềm chính là một dạng gọi đến chương trình con.
- Mục đích của việc phục vụ ngắt quãng là bằng cách nào đó chuyển điều khiển sang cho một chương trình con đặc biệt gọi là chương trình phục vụ ngắt quãng của riêng ngắt quãng được phục vụ.
- Đối với vi xử lý 86, việc phục vụ ngắt quãng được thực hiện thông qua số ngắt của từng ngắt quãng. Mỗi ngắt quãng có một số ngắt riêng. Số ngắt là một số 1 byte nên vi xử lý 86 chỉ có thể phục vụ cho tối đa 256 ngắt quãng.
- Vi xử lý 86 sử dụng phương pháp vector ngắt để chuyển điều khiển đến các chương trình phục vụ ngắt quãng.
- Vector ngắt là các biến bộ nhớ dài 4 bytes mà có khả năng chứa được một địa chỉ luận lý đầy đủ gồm 2 byte segment và 2 byte offset. Người ta dùng vector ngắt để chứa địa chỉ bắt đầu của chương trình phục vụ ngắt quãng.
- Các vector ngắt được xếp nối tiếp nhau kể từ đầu của vùng bộ nhớ tạo thành bảng vector ngắt. Chiều dài của bảng vector ngắt là 256 4=1024 hay 400h. Như vậy bảng vector ngắt sẽ nằm trong vùng bộ nhớ có địa chỉ vật lý từ 00000h đến 003FFh.
- Số thứ tự của các vector ngắt được qui định chính là số ngắt tương ứng nên vị trí của vector ngắt được xác định theo cách sau :

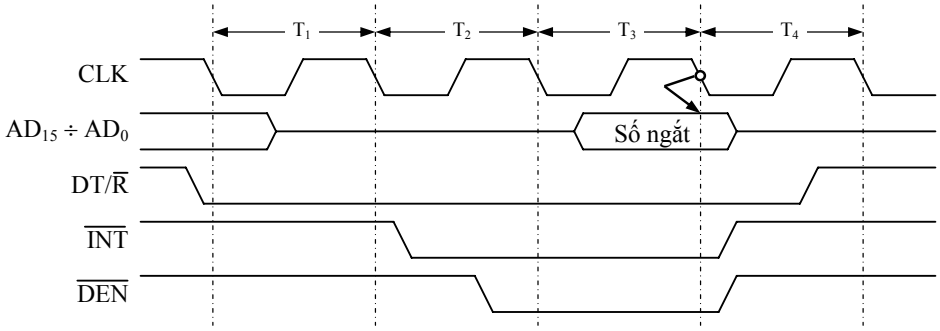
địa chỉ vật lý của vector ngắt = số ngắt × 4

- Sau khi xác định được vị trí của vector ngắt rồi, CPU sẽ lấy địa chỉ chương trình phục vụ ngắt trong vector ngắt ra và chuyển điều khiển đến đó. Tức là nó sẽ thực hiện một lệnh gọi đến chương trình phục vụ ngắt quãng.
- Ví dụ để phục vụ cho ngắt quãng số 8 theo hình sau, CPU sẽ chạy chương trình con có địa chỉ trong vector ngắt 8 (ở địa chỉ vật lý 8 4=32=20h) mà cụ thể là địa chỉ luận lý 3000:2A76.



- Hoạt động đáp ứng ngắt quãng của vi xử lý 86 chỉ dùng cho ngắt cứng.
- Vi xử lý 86 dùng hoạt động này để đọc số ngắt tương ứng từ khối xuất nhập.
- Hoạt động đáp ứng ngắt quãng được thực hiện bằng chu kỳ máy đáp ứng ngắt quãng kéo dài trong 4T.
- Tuyến địa chỉ không được dùng trong chu kỳ đáp ứng ngắt quãng.

- Các tín hiệu điều khiển gồm có :
  $\overline{DEN}$=0 DT/$\overline{R}$=0 $\overline{INTA}$=0
- Tín hiệu $\overline{INTA}$ là tín hiệu đặc trưng cho chu kỳ máy đáp ứng ngắt quãng.
- Tín hiệu yêu cầu ngắt quãng INTR được kiểm tra ở cuối mỗi chu kỳ lệnh nghĩa là CPU phải thi hành xong lệnh hiện tại rồi mới chuyển sang hoạt động đáp ứng ngắt quãng. Khi đó nó phát ra 2 chu kỳ máy đáp ứng ngắt quãng liên tiếp mà còn gọi là chu kỳ INTA.
- Sau chu kỳ INTA thứ 2, sự thi hành lệnh được chuyển sang chương trình con phục vụ ngắt quãng.
- Giản đồ xung chu kỳ máy đáp ứng ngắt quãng như sau :



TẬP LỆNH VI XỬ LÝ 8086/8088

I. TỔ CHỨC LỆNH CỦA VI XỬ LÝ 8086/8088 :

1. Dạng lệnh :

- Một lệnh của vi xử lý 86 có dạng tổng quát như sau :

  <Mã gợi nhớ> <Toán hạng đích>,<Toán hạng nguồn>

- **Mã gợi nhớ** giúp cho người sử dụng biết hoạt động của lệnh. Mã gợi nhớ thường là các chữ tiếng anh viết tắt như : MOV là lệnh chuyển, ADD là lệnh cộng, AND là lệnh và luận lý, JMP là lệnh nhảy . . .
- **Toán hạng đích** giữ kết quả (nếu có yêu cầu) sau khi thi hành lệnh. Toán hạng đích có thể là thanh ghi hay bộ nhớ.
- **Toán hạng nguồn** có thể là thanh ghi, bộ nhớ hay một số tức thời.
- Toán hạng thanh ghi là các thanh ghi của vi xử lý 86 gồm các thanh ghi tổng quát (8 bit lẫn 16 bit) và các thanh ghi đoạn đã biết.
- Toán hạng số tức thời có thể là số trong các hệ đếm khác nhau và được viết theo qui định như sau :

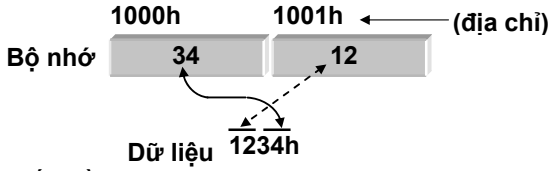
  . Số hệ 2 : ××××××××B (× là 1 bit nhị phân).
  Ví dụ : 01101101B, 11111111B
  . Số hệ 10 : ××××× , hay ×××××D (× là một số thuộc hệ 10).
  Ví dụ : 65535, 1000
  . Số hệ 16 : ××××H và bắt đầu bằng số (là một số thuộc hệ 16).
  Ví dụ : 1A59H, 0E05BH

- Toán hạng bộ nhớ dùng trong tập lệnh vi xử lý 86 sử dụng phương pháp định địa chỉ tổng hợp được gọi là *địa chỉ hiệu dụng*.

2. Địa chỉ hiệu dụng :

- Địa chỉ hiệu dụng là tổ hợp của 3 nhóm sau được đặt trong dấu ngoặc vuông [ ]:
  . Nhóm thanh ghi chỉ số : SI, DI
  . Nhóm thanh ghi nền : BX, BP
  . Địa chỉ trực tiếp : số 16 bit
- Các thanh ghi trong cùng một nhóm không được xuất hiện trong cùng một địa chỉ hiệu dụng.
- Ví dụ :
  . Địa chỉ hiệu dụng hợp lệ :
  [1000h], [SI], [DI], [BX], [BP]
  [SI+BX], [SI+BP], [DI+BX], [DI+BP], [SI+1000h], [DI+100h], [BX+1], [BP+1]
  [SI][BX][1000h], [SI+ BP+1000h], [DI+BX][1000h], [DI+1000h][BP]
  . Địa chỉ hiệu dụng *không hợp lệ* :
  [70000], [AX], [SI+DI+1000h], [BX][BP]
- Địa chỉ hiệu dụng chính là thành phần offset của địa chỉ luận lý bộ nhớ.
- Segment của địa chỉ hiệu dụng được mặc định như sau :
  . Nếu không sử dụng BP trong địa chỉ hiệu dụng thì mặc định theo thanh ghi DS.
  . Nếu có BP trong địa chỉ hiệu dụng thì mặc định theo thanh ghi SS.

- Các hoạt động thực hiện trên bộ nhớ thông qua địa chỉ hiệu dụng chia ra làm 2 trường hợp : hoạt động 8 bit và hoạt động 16 bit.
- Hoạt động bộ nhớ 8 bit làm việc trên 1 byte bộ nhớ ngay vị trí chỉ ra bởi địa chỉ hiệu dụng.
- Hoạt động bộ nhớ 16 bit sẽ làm việc trên 2 byte bộ nhớ có địa chỉ kế tiếp nhau và nội dung của chúng được ghép lại thành dữ liệu 16 bit theo qui tắc "***byte cao địa chỉ cao, byte thấp địa chỉ thấp***" như trong hình sau :



- Để thuận tiện trong vấn đề giải thích lệnh, ta qui ước thêm cách diễn tả sau :
  . Dữ liệu 8 bit của bộ nhớ : **[địa chỉ]**
  . Dữ liệu 16 bit của bộ nhớ : **[địa chỉ +1,địa chỉ]**
- Để xác định rõ hoạt động của bộ nhớ, ta phải dùng thêm toán tử PTR như sau :
  . Hoạt động 8 bit : **BYTE PTR** [1000h] là tham khảo 1 byte bộ nhớ có địa chỉ 1000h
  . Hoạt động 16 bit : **WORD PTR** [1000h] là tham khảo đến 2 byte bộ nhớ liên tiếp 1000h và 1001h

II. CÁC NHÓM LỆNH CỦA VI XỬ LÝ 8086/8088 :

1. Ký hiệu qui ước :

- Các chữ viết tắt dùng trong các nhóm lệnh :

reg	: thanh ghi tổng quát.
reg16	: thanh ghi 16 bit.
segreg	: thanh ghi đoạn.
accum	: thanh ghi bộ tích lũy AX hoặc AL.
mem	: bộ nhớ (địa chỉ hiệu dụng).
mem16	: bộ nhớ 2 byte liên tiếp (địa chỉ hiệu dụng).
mem32	: bộ nhớ 4 byte liên tiếp (địa chỉ hiệu dụng).
immed	: số tức thời.
immed8	: số tức thời 8 bit.
shortlabel	: nhãn ngắn (-128 byte +127 byte).
nearlabel	: nhãn trong đoạn (2 byte offset).
farlabel	: nhãn ngoài đoạn (4 byte : 2 byte segment và 2 byte offset).

2. Nhóm lệnh chuyển dữ liệu :

2.1 Lệnh MOV :

- Dạng lệnh :

MOV *reg,reg*	MOV *reg,immed*
MOV *mem,reg*	MOV *mem,immed*
MOV *reg,mem*	MOV *mem16,segreg*
MOV *reg16,segreg*	MOV *segreg,mem16*
MOV *segreg,reg16*	

- Giải thích : ***thđ ← thn***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Chép toán hạng nguồn vào toán hạng đích.
- *Ví dụ* : MOV AX,CX ; *AX ← CX*

```
MOV  DL,BH              ; DL ← BH
MOV  [SI+1000h],BP      ; [SI+1001h, SI+1000h] ← BP
MOV  DX,[1000h]         ; DX ← [1001h,1000h]
MOV  DX,DS              ; DX ← DS
MOV  ES,BX              ; ES ← BX
MOV  DI,12h             ; DI ← 12h
MOV  AL,12h             ; AL ← 12h
MOV  BYTE PTR [1000h],12h     ; [1000h] ← 12h
MOV  WORD PTR [2000h],1200h ; [2001h,2000h] ← 1200h
MOV  [BX],DS            ; [BX+1,BX] ← DS
MOV  SS,[2000h]         ; SS ← [2001h,2000h]
```

2.2 Lệnh PUSH :

- Dạng lệnh : PUSH *reg16* PUSH *segreg*
  PUSH *mem16*
- Giải thích : ***SP ← SP-2***
  [SS:SP+1,SS:SP] ← thn
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Đẩy toán hạng nguồn 16 bit vào chồng (địa chỉ đỉnh chồng là SS:SP).
- *Ví dụ* :

```
PUSH  DI          ; [SS:SP+1,SS:SP] ← DI
PUSH  CS          ; [SS:SP+1,SS:SP] ← CS
PUSH  [SI]        ; [SS:SP+1,SS:SP] ← [SI+1,SI]
```

2.3 Lệnh POP :

- Dạng lệnh : POP *reg16* POP *segreg*
  POP *mem16*
- Giải thích : ***thđ ← [SS:SP+1,SS:SP]***
  SP ← SP+2
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lấy dữ liệu từ đỉnh chồng vào toán hạng đích.
- *Ví dụ* :

```
POP   AX          ; AX← [SS:SP+1,SS:SP]
POP   ES          ; ES ← [SS:SP+1,SS:SP]
POP   [BX+1]      ; [BX+2,BX+1]← [SS:SP+1,SS:SP]
```

2.4 Lệnh XCHG :

- Dạng lệnh : XCHG *reg,reg* XCHG *mem,reg*
  XCHG *accum,reg16* XCHG *reg,mem*
- Giải thích : ***thđ ↔ thn***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Trao đổi nội dung hai toán hạng cho nhau.
- *Ví dụ* :

```
XCHG   AX,CX      ; AX ↔ CX
XCHG   AH,AL      ; AH↔ AL
XCHG   [1000h],DX ; [1001h,1000h] ↔ DX
```

2.5 Lệnh IN :

- Dạng lệnh : IN *accum,immed8*
  IN *accum,DX*
- Giải thích : ***btl ← [cổng IO]***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nhập dữ liệu từ cổng xuất nhập vào thanh ghi bộ tích lũy AL hay AX. Trường hợp AX sẽ nhập byte thấp trước, byte cao sau.
- Dạng lệnh có *immed8* dùng trong trường hợp địa chỉ cổng xuất nhập 8 bit.
- *Ví dụ* : IN AL,61h
  IN AX,40h
- Dạng lệnh có thanh ghi DX dùng cho trường hợp địa chỉ cổng 16 bit. Tuy nhiên dạng này vẫn có thể dùng cho cổng xuất nhập có địa chỉ 8 bit và có lợi khi sử dụng địa chỉ cổng để nhập nhiều lần.
- *Ví dụ* : MOV DX,378h
  IN AL,DX

2.6 Lệnh OUT :

- Dạng lệnh : OUT *immed8,accum*
  OUT *DX,accum*
- Giải thích : ***[cổng IO] ← btl***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Xuất dữ liệu từ thanh ghi bộ tích lũy AL hoặc AX ra cổng xuất nhập có địa chỉ 8 bit là số tức thời *immed8* hay có địa chỉ 16 bit trong thanh ghi DX.
- *Ví dụ* : OUT 20h,AL
  MOV DX,2F8h
  OUT DX,AL

2.7 Lệnh XLAT :

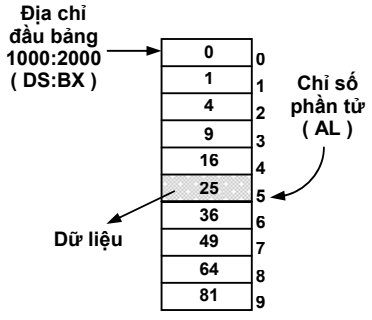
- Dạng lệnh : XLAT
- Giải thích : ***AL ← [DS:BX+AL]***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Tra bảng. Thanh ghi BX giữ địa chỉ đầu bảng. Thanh ghi AL giữ chỉ số của phần tử cần lấy ra.
- *Ví dụ* : bài toán tính bình phương một số nguyên có thể thực hiện bằng cách tra bảng như sau

```
MOV     CX,1000h
MOV     DS,CX
MOV     BX,2000h    ; địa chỉ đầu bảng
MOV     AL,5        ; chỉ số
XLAT                ; tra bảng
-----------------------
```

Sau khi làm xong lệnh XLAT :
AL = 25 = 5^2



- Lệnh XLAT có ứng dụng trong mã hóa dữ liệu.

2.8 Lệnh LEA :

- Dạng lệnh : LEA *reg16,mem*

- Giải thích : ***thđ ← địa chỉ***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nạp địa chỉ hiệu dụng vào thanh ghi 16 bit.
- *Ví dụ* : LEA BX,[1000h] ; *BX ← 1000h*
  LEA SI,[DI][BX][2000h] ; *SI ← DI+BX+2000h*

2.9 Lệnh LDS :
- Dạng lệnh : LDS *reg16,mem32*
- Giải thích : ***DS ← [địa chỉ+3,địa chỉ+2]***
  thđ ← [địa chỉ+1,địa chỉ]
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nạp 4 byte bộ nhớ (con trỏ) vào thanh ghi DS và một thanh ghi tổng quát.
- *Ví dụ* : LDS BX,[1000h] ; *DS ← [1003h, 1002h]*
  ; *BX ← [1001h, 1000h]*

2.10 Lệnh LES :
- Dạng lệnh : LES *reg16,mem32*
- Giải thích : ***ES ← [địa chỉ+3,địa chỉ+2]***
  thđ ← [địa chỉ+1,địa chỉ]
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nạp 4 byte bộ nhớ (con trỏ) vào thanh ghi ES và một thanh ghi tổng quát.
- *Ví dụ* : LES DI,[1000h] ; *ES ← [1003h, 1002h]*
  ; *SI ← [1001h, 1000h]*

2.11 Lệnh LAHF :
- Dạng lệnh : LAHF
- Giải thích : ***AH ← $Flags_L$***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nạp 8 bit thấp của thanh ghi cờ vào thanh ghi AH.

2.12 Lệnh SAHF :
- Dạng lệnh : SAHF
- Giải thích : ***$Flags_L$ ← AH***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Cất thanh ghi AH vào 8 bit thấp của thanh ghi cờ.

2.13 Lệnh PUSHF :
- Dạng lệnh : PUSHF
- Giải thích : ***SP ← SP - 2***
  [SS:SP+1,SS:SP] ← Flags
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Đẩy thanh ghi cờ vào chồng.

2.14 Lệnh POPF :

- Dạng lệnh : POPF
- Giải thích : ***Flags ← [SS:SP+1,SS:SP]***
  SP ← SP + 2
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×	×	×	×	×	×	×	×

- Lấy thanh ghi cờ từ chồng ra.

3. Nhóm lệnh số học :

3.1 Lệnh ADD :

- Dạng lệnh : ADD *reg,reg* ADD *reg,immed*
  ADD *mem,reg* ADD *mem,immed*
  ADD *reg,mem* ADD *accum,immed*
- Giải thích : ***thđ ← thđ + thn***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- Cộng toán hạng nguồn vào toán hạng đích. Kết quả cất vào toán hạng đích.
- *Ví dụ* :

```
ADD CX,SI          ; CX ← CX + SI
ADD DH,BL          ; DH ← DH + BL
ADD [1000h],BX     ; [1001h,1000h] ← [1001h,1000h] + BX
ADD [2000h],CL     ; [2000h] ← [2000h] + CL
ADD AL,[0000h]     ; AL ← AL + [0000h]
ADD BYTE PTR [SI+8],5   ; [SI+8] ← [SI+8] + 05h
```

3.2 Lệnh ADC :

- Dạng lệnh : ADC *reg,reg* ADC *reg,immed*
  ADC *mem,reg* ADC *mem,immed*
  ADC *reg,mem* ADC *accum,immed*
- Giải thích : ***thđ ← thđ + thn + CF***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- Cộng toán hạng đích với toán hạng nguồn với cờ nhớ. Kết quả cất vào toán hạng đích. ADC dùng cho phép cộng 2 số có chiều dài nhiều byte.
- *Ví dụ* :

```
ADC BX,AX                  ; BX ← BX + AX + CF
ADC BYTE PTR [1000h],7Ah   ; [1000h] ← [1000h]+7Ah+CF
```

3.3 Lệnh INC :

- Dạng lệnh : INC *reg* INC *mem*
- Giải thích : ***thđ ← thđ + 1***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	

- Tăng tức là cộng 1 vào toán hạng đích nhưng không ảnh hưởng cờ nhớ.
- *Ví dụ :*

```
INC CH
INC WORD PTR [1000h]
```

3.4 Lệnh AAA :

- Dạng lệnh : AAA
- Giải thích : ***Nếu ($b_3b_2b_1b_0$ của AL) > 9 hoặc AF=1 thì***
  AL ← (AL+6) and 0Fh, AH ← AH+1, CF ← 1, AF ← 1

- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	×	?	×

- Chỉnh ASCII sau phép cộng. Chỉnh kết quả trong AL thành 2 số BCD không nén trong AH và AL.
- *Ví dụ :* kết quả : AH=00, AL= 0Dh, AF=0, CF=0
  sau khi chỉnh: AH=01h, AL=03h, AF=1, CF=1

3.4 Lệnh DAA :

- Dạng lệnh : DAA
- Giải thích : ***Nếu ($b_3b_2b_1b_0$ của AL) > 9 hoặc AF=1 thì***
  AL ← (AL+6), AF ← 1
  Nếu AL > 9Fh hoặc CF=1 thì
  AL ← AL+60h, CF ← 1
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	×	?	×

- Chỉnh thập phân sau phép cộng. Chỉnh kết quả trong AL thành số BCD nén trong AL.
- *Ví dụ :* kết quả : AL= 0Dh, AF=0, CF=0
  sau khi chỉnh: AL=13h, AF=1, CF=0
  kết quả : AL= 9Dh, AF=0, CF=0
  sau khi chỉnh: AL=03h, AF=1, CF=1

3.5 Lệnh SUB :

- Dạng lệnh : SUB *reg,reg* SUB *reg,immed*
  SUB *mem,reg* SUB *mem,immed*
  SUB *reg,mem* SUB *accum,immed*
- Giải thích : ***thđ ← thđ - thn***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- Trừ toán hạng đích cho toán hạng nguồn. Kết quả cất vào toán hạng đích.
- *Ví dụ* :
```
SUB  DL,AL      ; DL ← DL - AL
SUB  CX,[DI]    ; CX ← CX - [DI+1,DI]
SUB  BP,4       ; BP ← BP - 4
```

3.6 Lệnh SBB :

- Dạng lệnh : SBB *reg,reg* SBB *reg,immed*
  SBB *mem,reg* SBB *mem,immed*
  SBB *reg,mem* SBB *accum,immed*
- Giải thích : ***thđ ← thđ - thn - CF***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- Trừ toán hạng đích cho toán hạng nguồn và cờ nhớ. Kết quả cất vào toán hạng đích.
- *Ví dụ* :
```
SBB  SI,BX              ; SI ← SI - BX -CF
SBB  BYTE PTR [BX],2    ; [BX+1,BX] ← [BX+1,BX] - 2 - CF
```

3.7 Lệnh DEC :

- Dạng lệnh : DEC *reg* DEC *mem*
- Giải thích : ***thđ ← thđ - 1***

- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	

- Giảm tức là trừ 1 vào toán hạng đích nhưng không ảnh hưởng cờ nhớ.
- *Ví dụ :* DEC AX
  DEC BYTE PTR [SI][2000h]

3.8 Lệnh NEG :

- Dạng lệnh : NEG *reg* NEG *mem*
- Giải thích : ***thđ ← bù 2(thđ)***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- Lấy bù 2 toán hạng đích.

3.9 Lệnh CMP :

- Dạng lệnh : CMP *reg,reg* CMP *reg,immed*
  CMP *mem,reg* CMP *mem,immed*
  CMP *reg,mem* CMP *accum,immed*
- Giải thích : ***thđ - thn***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- So sánh. Thực hiện trừ toán hạng đích cho toán hạng nguồn, không lưu lại kết quả mà chỉ giữ lại tác động của phép trừ lên các cờ.
- *Ví dụ* : CMP AL,8 ; *AL - 8*
  CMP WORD PTR [1000h], 3 ; *[1001h,1000h] - 3*

3.10 Lệnh AAS :

- Dạng lệnh : AAS
- Giải thích : ***Nếu ($D_3D_2D_1D_0$ của AL) > 9 hoặc AF=1 thì***
  AL ← (AL - 6) and 0Fh, AH ← AH - 1, CF ← 1, AF ← 1
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	×	?	×

- Chỉnh ASCII sau phép cộng. Chỉnh kết quả trong AL thành 2 số BCD không nén trong AH và AL.
- *Ví dụ :* kết quả : AH=00h, AL= 0Dh, AF=0, CF=0
  sau khi chỉnh: AH=01h, AL=03h, AF=1, CF=1

3.11 Lệnh DAS :

- Dạng lệnh : DAS
- Giải thích : ***Nếu ($D_3D_2D_1D_0$ của AL) > 9 hoặc AF=1 thì***
  AL ← (AL - 6), AF ← 1
  Nếu AL > 9Fh hoặc CF=1 thì
  AL ← AL - 60h, CF ← 1
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			×	×	×	×	×

- Chỉnh thập phân sau phép trừ. Chỉnh kết quả trong AL thành số BCD nén trong AL.
- *Ví dụ :* kết quả của (4 - 8) : AL= 0FCh, AF=1, CF=1
  sau khi chỉnh : AL=96h, AF=1, CF=1

3.12 Lệnh MUL :

- Dạng lệnh : MUL *reg* MUL *mem*
- Giải thích : ***Toán hạng nguồn 8 bit thì : AX ← AL * thn8***
  Toán hạng nguồn 16 bit thì : DX AX ← AX * thn16
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			?	?	?	?	×

- Nhân hai số không dấu 8 bit hay 16 bit. Số bit thực hiện được xác định bằng chiều dài của toán hạng nguồn.
  - ♣ Phép nhân 8 bit : thực hiện nhân AL với toán hạng nguồn, kết quả 16 bit cất trong thanh ghi AX.
  - ♣ Phép nhân 16 bit : thực hiện nhân AX với toán hạng nguồn, kết quả 32 bit cất trong 2 thanh ghi DX và AX. DX giữ 16 bit cao, AX giữ 16 bit thấp.
- *Ví dụ :* Nếu AL=5, CH=4, sau khi thực hiện lệnh
  MUL CH
  ta có AX = AL*CH = 0014h.
  Nếu AX=500h, [1001h,1000h]=401h, sau khi thực hiện lệnh
  MUL WORD PTR [1000h]
  ta có DXAX = AX * [1001h,1000h] = 500h * 401h = 00140500h
  Nghĩa là DX=0014h và AX=0500h.

3.13 Lệnh IMUL :

- Dạng lệnh : IMUL *reg* IMUL *mem*
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			?	?	?	?	×

- Nhân hai số có dấu. Thực hiện giống hệt như lệnh MUL, chỉ có kết quả được xem là số có dấu.

3.14 Lệnh AAM :

- Dạng lệnh : AAM
- Giải thích : ***AH ← (AL / 0Ah)***
  AL ← số dư của (AL / 0Ah)
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			×	×	?	×	?

- Chỉnh ASCII sau phép nhân. Có thể dùng để đổi số hex ra số BCD không nén.
- *Ví dụ :* kết quả : AH = 00, AL = 41h.
  sau khi chỉnh : AH = 06, AL = 05.

3.15 Lệnh DIV :

- Dạng lệnh : DIV *reg* DIV *mem*
- Giải thích : ***Toán hạng nguồn 8 bit thì : AL ← (AX / thn8)***
  AH ← số dư của (AX / thn8)
  Toán hạng nguồn 16 bit thì : AX ← (DXAX / thn16)
  DX ← số dư của (DXAX / thn16)
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	?	?	?

- Chia hai số không dấu.
- Nếu toán hạng nguồn là thanh ghi hay bộ nhớ 8 bit, thực hiện chia số 16 bit trong thanh ghi AX cho toán hạng nguồn 8 bit. Kết quả 8 bit cất trong thanh ghi AL. Số dư 8 bit cất trong thanh ghi AH.

- Nếu toán hạng nguồn là thanh ghi hay bộ nhớ 16 bit, thực hiện chia số 32 bit trong 2 thanh ghi DXAX cho toán hạng nguồn 16 bit. Kết quả 16 bit cất trong thanh ghi AX. Số dư 16 bit cất trong thanh ghi DX.
- *Ví dụ :* Nếu AX=0024h, [2000h]=05 thì sau khi thực hiện lệnh
  DIV BYTE PTR [2000h]
  ta có *AL=07 và AH=01*.
  Nếu DX=0001h, AX=0024h, BX=0200h thì sau khi thực hiện lệnh
  DIV BX
  ta có *AX=0008 và DX=0024h.*

3.16 Lệnh IDIV :
- Dạng lệnh : IDIV *reg* IDIV *mem*
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			?	?	?	?	?

- Chia hai số có dấu. Thực hiện giống như lệnh DIV nhưng kết quả coi là số có dấu.

3.17 Lệnh AAD :
- Dạng lệnh : AAD
- Giải thích : ***AL ← ((AH * 0Ah) + AL)***
  AH ← 0
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
?			×	×	?	×	?

- Chỉnh ASCII trước phép chia IDIV. Có thể dùng lệnh này để đổi số BCD không nén trong AX ra thành giá trị nhị phân trong AL.
- *Ví dụ :* nếu có : AL=03h, AH=05h
  sau khi chỉnh : AL=35h, AH=00h

3.18 Lệnh CBW :
- Dạng lệnh : CBW
- Giải thích : ***Nếu AL < 80h thì AH ← 00h***
  Nếu AL >= 80h thì AH ← 0FFh
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Mở rộng dấu trước khi dùng lệnh chia. Đổi số 1 byte có dấu trong AL thành số 2 byte có dấu trong AX.

3.19 Lệnh CWD :
- Dạng lệnh : CWD
- Giải thích : ***Nếu AX < 8000h thì DX ← 0000h***
  Nếu AX >= 8000h thì DX ← 0FFFFh
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Mở rộng dấu trước khi dùng lệnh chia. Đổi số 2 byte có dấu trong AX thành số 4 byte có dấu trong DXAX.

4. Nhóm lệnh luận lý :

4.1 Lệnh NOT :
- Dạng lệnh : NOT *reg* NOT *mem*
- Giải thích : ***thđ ← bÓ 1 của thđ***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Đảo hay lấy bù 1.
- *Ví dụ :*
```
NOT   AL
NOT   WORD PTR [BX+1000h]
```

4.2 Lệnh SHL/SAL :
- Dạng lệnh :
```
SHL   reg,1          SHL   mem,1
SHL   reg,CL         SHL   mem,CL
```
- Giải thích : ***thđ ← (thđ) dịch trái 1 hay nhiều bit.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	×

- Dịch trái. Dạng SHL reg,1 dùng để dịch trái 1 bit. Dạng SHL reg,CL dùng để dịch trái nhiều bit. Lúc đó thanh ghi CL chứa số bit cần dịch.
- *Ví dụ :*
```
SHL   DH,1
SAL   CX,1
MOV   CL,3
SHL   WORD PTR [1000h],CL      ; dịch trái 3 bit.
```

4.3 Lệnh SHR :
- Dạng lệnh :
```
SHR   reg,1          SHR   mem,1
SHR   reg,CL         SHR   mem,CL
```
- Giải thích : ***thđ ← (thđ) dịch phải luận lý 1 hay nhiều bit.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	×

- Dịch phải luận lý. Dạng có thanh ghi CL dùng để dịch nhiều bit.
- *Ví dụ :*
```
SHR   AX,1
MOV   CL,2
SHR   BYTE PTR [1000h],CL      ; dịch phải luận lý 2 bit.
```

4.4 Lệnh SAR :
- Dạng lệnh :
```
SAR   reg,1          SAR   mem,1
SAR   reg,CL         SAR   mem,CL
```
- Giải thích : ***thđ ← (thđ) dịch phải số học 1 hay nhiều bit.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	×

- Dịch phải số học. Dạng có thanh ghi CL dùng để dịch nhiều bit.
- *Ví dụ :*
```
SAR   DX,1
MOV   CL,7
SAR   WORD PTR [2000h],CL      ; dịch phải số học 7 bit.
```

4.5 Lệnh ROL :
- Dạng lệnh :
```
ROL   reg,1          ROL   mem,1
ROL   reg,CL         ROL   mem,CL
```
- Giải thích : ***thđ ← (thđ) quay trái không qua cờ nhớ 1 hay nhiều bit.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	×

- Quay trái không qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.
- *Ví dụ :*
```
ROL   DL,1
MOV   CL,6
ROL   WORD PTR [1000h],CL  ; quay trái không qua cờ nhớ 6 bit.
```

4.6 Lệnh ROR :

- Dạng lệnh : ROR *reg,1* ROR *mem,1*
  ROR *reg,CL* ROR *mem,CL*
- Giải thích : ***thđ ← (thđ) quay phải không qua cờ nhớ 1 hay nhiều bit.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	×

- Quay phải không qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.
- *Ví dụ :*

```
ROR   SI,1
MOV   CL,3
ROR   WORD PTR [3000h],CL  ; quay phải không qua cờ nhớ 3 bit
```

4.7 Lệnh RCL :

- Dạng lệnh : RCL *reg,1* RCL *mem,1*
  RCL *reg,CL* RCL *mem,CL*
- Giải thích : ***thđ ← (thđ) quay trái qua cờ nhớ 1 hay nhiều bit.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	×

- Quay trái qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.
- *Ví dụ :*

```
RCL   BX,1
MOV   CL,4
RCL   BYTE PTR [1000h],CL      ; quay trái qua cờ nhớ 4 bit.
```

4.8 Lệnh RCR :

- Dạng lệnh : RCR *reg,1* RCR *mem,1*
  RCR *reg,CL* RCR *mem,CL*
- Giải thích : ***thđ ← (thđ) quay phải qua cờ nhớ 1 hay nhiều bit.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	×

- Quay phải qua cờ nhớ. Dạng có thanh ghi CL dùng để quay nhiều bit.
- *Ví dụ :*

```
RCR   CH,1
MOV   CL,2
RCR   BYTE PTR [1800h],CL      ; quay phải  qua cờ nhớ 2 bit.
```

4.9 Lệnh AND :

- Dạng lệnh : AND *reg,reg* AND *reg,immed*
  AND *mem,reg* AND *mem,immed*
  AND *reg,mem* AND *accum,immed*
- Giải thích : ***thđ ← thđ AND thn.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	0

- Và luận lý. Xóa cờ nhớ về 0.
- *Ví dụ :*

```
AND   CH,AH
AND   [SI],DX
AND   BYTE PTR [1000h],10000000b
AND   AX,0FFF0h
```

4.10 Lệnh TEST :

- Dạng lệnh : TEST *reg,reg* TEST *reg,immed*
  TEST *mem,reg* TEST *mem,immed*
  TEST *reg,mem* TEST *accum,immed*
- Giải thích : ***thđ AND thn.***

- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
0			×	×	?	×	0

- Và luận lý hai toán hạng nhưng không giữ lại kết quả mà chỉ lập các cờ. Xóa cờ nhớ và cờ tràn về 0. Thường dùng để kiểm tra bit. Lúc đó toán hạng nguồn là một mặt nạ bit cần thiết.
- *Ví dụ :*
```
TEST  DX,1                          ; kiểm tra bit 0
TEST  BYTE PTR [2000h],10000000b    ; kiểm tra bit 7
```

4.11 Lệnh OR :
- Dạng lệnh :
```
OR   reg,reg          OR   reg,immed
OR   mem,reg          OR   mem,immed
OR   reg,mem          OR   accum,immed
```
- Giải thích : ***thđ ← thđ OR thn.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	0

- Hay luận lý. Xóa cờ nhớ về 0.
- *Ví dụ :*
```
OR   DL,CH
OR   BP,[2000h]
OR   WORD PTR [1000h],000Fh
```

4.12 Lệnh XOR :
- Dạng lệnh :
```
XOR  reg,reg          XOR  reg,immed
XOR  mem,reg          XOR  mem,immed
XOR  reg,mem          XOR  accum,immed
```
- Giải thích : ***thđ ← thđ XOR thn.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	?	×	0

- Hay ngoại luận lý. Xóa cờ nhớ về 0.
- *Ví dụ :*
```
XOR  DL,80h          ; đảo bit 7
XOR  [2000h],AL      ; [2000h] ← [2000h] XOR AL
```

5. Xử lý chuỗi :

5.1 Tiếp đầu lệnh REP :

♣ Dạng 1 : REP *lệnh xử lý chuỗi*
- Giải thích : ***giảm CX, lặp lại lệnh theo sau Nếu CX ≠ 0***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lặp lại không điều kiện. Thanh ghi CX giữ số lần lặp. Thường dùng với lệnh chép chuỗi MOVS.
- *Ví dụ :*
```
MOV   CX,10
REP   MOVSB   ; thực hiện lệnh MOVSB 10 lần.
```

♣ Dạng 2 : REPE / REPZ *lệnh xử lý chuỗi*
- Giải thích : ***giảm CX, lặp lại lệnh theo sau Nếu CX ≠ 0 và ZF = 1***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lặp lại nếu bằng / nếu không. Hai điều kiện CX ≠ 0 và ZF = 1 phải thỏa đồng thời thì lệnh theo sau mới được lặp lại. Nếu không, làm qua lệnh kế. Thường dùng với lệnh so sánh chuỗi CMPS để tìm chuỗi con trong chuỗi lớn.
- *Ví dụ :*
```
MOV   CX,10
```

```
REPE  CMPSB          ; thực hiện lệnh CMPSB nếu chưa đủ 10
                     ; lần và hai chuỗi vẫn còn bằng nhau.
```

♣ Dạng 3 : REPNE / REPNZ *lệnh xử lý chuỗi*

- Giải thích : ***giảm CX, lặp lại lệnh theo sau Nếu CX ≠ 0 và ZF = 0***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lặp lại nếu bằng / nếu không. Hai điều kiện CX ≠ 0 và ZF = 1 phải thỏa đồng thời thì lệnh theo sau mới được lặp lại. Nếu không, làm qua lệnh kế. Thường dùng với lệnh quét chuỗi SCAS để dò tìm ký tự tong chuỗi.
- *Ví dụ :*

```
MOV   CX,20
REPNE SCASB          ; thực hiện lệnh CMPSB nếu chưa đủ 20
                     ; lần và chưa tìm ra ký tự trong chuỗi.
```

5.2 Lệnh MOVS :

- Dạng lệnh : MOVSB
  MOVSW
- Giải thích :
  - ♣ MOVSB : ***[ES:DI] ← [DS:SI]***
    Nếu DF=0 thì : SI ← SI + 1, DI← DI + 1
    ngược lại thì : SI ← SI - 1, DI← DI - 1
  - ♣ MOVSW : ***[ES:DI+1,ES:DI] ← [DS:SI+1,DS:SI]***
    Nếu DF=0 thì : SI ← SI + 2, DI← DI + 2
    ngược lại thì : SI ← SI - 2, DI← DI - 2
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Chép byte hay word từ nguồn sang chuỗi đích. Cặp thanh ghi DS:SI giữ địa chỉ chuỗi nguồn. Cặp thanh ghi ES:DI giữ địa chỉ chuỗi đích. Các địa chỉ chuỗi nguồn trong thanh ghi SI và địa chỉ chuỗi đích trong thanh ghi DI được tự động tăng hay giảm sau mỗi lần chép. Chiều tăng giảm địa chỉ tùy thuộc cờ định hướng DF. DF=0 xử lý tăng địa chỉ. DF=1 xử lý giảm địa chỉ.
- Lệnh này thường dùng kết hợp với tiếp đầu lệnh REP để thực hiện việc chép một chuỗi hay dãy. Lúc đó thanh ghi CX giữ chiều dài chuỗi nguồn.
- *Ví dụ 1 chép byte* : chép 80h byte từ địa chỉ 3000:1000 sang địa chỉ 4800:C200

```
MOV   AX,3000h
MOV   DS,AX
MOV   SI,1000h       ; địa chỉ chuỗi nguồn.
MOV   AX,4800h
MOV   ES,AX
MOV   DI,0C200h      ; địa chỉ chuỗi đích.
MOV   CX,80h         ; số lần chép.
CLD                  ; xóa cờ DF, xử lý tăng địa chỉ.
REP   MOVSB
```

- *Ví dụ 2 chép word* : yêu cầu như ví dụ 1

```
MOV   AX,3000h
MOV   DS,AX
MOV   SI,1000h       ; địa chỉ chuỗi nguồn.
MOV   AX,4800h
MOV   ES,AX
```

```
MOV  DI,0C200h    ; địa chỉ chuỗi đích.
MOV  CX,40h       ; số lần chép.
CLD               ; xóa cờ DF, xử lý tăng địa chỉ.
REP  MOVSW
```

5.3 Lệnh CMPS :

- Dạng lệnh : CMPSB
  CMPSW
- Giải thích :
  - ♣ CMPSB : ***[DS:SI] - [ES:DI]***
    Nếu DF=0 thì : SI ← SI + 1, DI← DI + 1
    ngược lại thì : SI ← SI - 1, DI← DI - 1
  - ♣ CMPSW : ***[DS:SI+1,DS:SI] - [ES:DI+1,ES:DI]***
    Nếu DF=0 thì : SI ← SI + 2, DI← DI + 2
    ngược lại thì : SI ← SI - 2, DI← DI - 2
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- So sánh byte hay word của chuỗi nguồn với chuỗi đích. Cặp thanh ghi DS:SI giữ địa chỉ chuỗi nguồn. Cặp thanh ghi ES:DI giữ địa chỉ chuỗi đích. Các thanh ghi giữ địa chỉ offset SI, DI được tự động tăng hay giảm sau mỗi lần so sánh. Chiều tăng giảm địa chỉ tùy thuộc cờ định hướng DF. DF=0 xử lý tăng địa chỉ. DF=1 xử lý giảm địa chỉ.
- Lệnh này thường dùng kết hợp với tiếp đầu lệnh REPE để thực hiện việc so sánh hai chuỗi hay hai dãy với nhau để tìm kiếm một chuỗi con trong một chuỗi lớn. Lúc đó thanh ghi CX giữ chiều dài chuỗi.
- Có thể có hai nguyên nhân làm ngừng lệnh so sánh chuỗi : hoặc hai chuỗi có byte hay word khác nhau (ZF = 0), hoặc hai chuỗi giống nhau (ZF = 1).
- *Ví dụ :* REPE CMPSB

5.4 Lệnh SCAS :

- Dạng lệnh : SCASB
  SCASW
- Giải thích :
  - ♣ SCASB : ***AL - [ES:DI]***
    Nếu DF=0 thì : DI← DI + 1
    ngược lại thì : DI← DI - 1
  - ♣ SCASW : ***AX - [ES:DI+1,ES:DI]***
    Nếu DF=0 thì : DI← DI + 2
    ngược lại thì : DI← DI - 2
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×			×	×	×	×	×

- Quét chuỗi nghĩa là so sánh byte trong thanh ghi AL hay word trong thanh ghi AX với chuỗi đích. Cặp thanh ghi ES:DI giữ địa chỉ chuỗi đích. Địa chỉ chuỗi đích được tự động tăng hay giảm sau mỗi lần so sánh. Chiều tăng giảm địa chỉ tùy thuộc cờ định hướng DF. DF=0 xử lý tăng địa chỉ. DF=1 xử lý giảm địa chỉ.
- Lệnh này thường dùng kết hợp với tiếp đầu lệnh REPNE để thực hiện việc tìm kiếm một dữ liệu trong một chuỗi. Lúc đó thanh ghi CX giữ chiều dài chuỗi.

- Có thể có hai nguyên nhân làm ngừng lệnh quét chuỗi : hoặc tìm thấy dữ liệu trong chuỗi (ZF=1 hay CX 0), hoặc hết chuỗi mà vẫn chưa tìm thấy dữ liệu (ZF=0 hay CX=0).
- *Ví dụ :* REPNE SCASW

5.5 Lệnh LODS :

- Dạng lệnh : LODSB
  LODSW
- Giải thích :
  - ♣ LODSB : ***AL ← [DS:SI]***
    Nếu DF=0 thì : SI ← SI + 1
    ngược lại thì : SI ← SI - 1
  - ♣ LODSW : ***AX ← [DS:SI+1,DS:SI]***
    Nếu DF=0 thì : SI ← SI + 2
    ngược lại thì : SI ← SI - 2
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nạp chuỗi nguồn byte vào thanh ghi AL hay chuỗi nguồn word vào thanh ghi AX. Cặp thanh ghi DS:SI giữ địa chỉ chuỗi nguồn. Địa chỉ chuỗi nguồn được tự động tăng hay giảm sau mỗi lần nạp. Chiều tăng giảm địa chỉ tùy thuộc cờ định hướng DF. DF=0 xử lý tăng địa chỉ. DF=1 xử lý giảm địa chỉ.

5.6 Lệnh STOS :

- Dạng lệnh : STOSB
  STOSW
- Giải thích :
  - ♣ STOSB : ***[ES:DI] ← AL***
    Nếu DF=0 thì : DI← DI + 1
    ngược lại thì : DI← DI - 1
  - ♣ STOSW : ***[ES:DI+1,ES:DI] ← AX***
    Nếu DF=0 thì : DI← DI + 2
    ngược lại thì : DI← DI - 2
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Cất byte trong thanh ghi AL hay word trong thanh ghi AX vào chuỗi đích. Cặp thanh ghi ES:DI giữ địa chỉ chuỗi đích. Địa chỉ chuỗi đích được tự động tăng hay giảm sau mỗi lần cất. Chiều tăng giảm địa chỉ tùy thuộc cờ định hướng DF. DF=0 xử lý tăng địa chỉ. DF=1 xử lý giảm địa chỉ.

6. Chuyển điều khiển :

6.1 Lệnh CALL :

- Dạng lệnh : CALL *nearlabel* CALL *mem16*
  CALL *farlabel* CALL *mem32*
  CALL *reg16*
- Giải thích :
  - ♣ *nearlabel* : ***PUSH IP***
    IP ← địa chỉ lệnh kế + độ dời 2 byte
  - ♣ *farlabel* : ***PUSH CS***
    PUSH IP

CS ← địa chỉ segment
IP ← địa chỉ offset

♣ *reg16* : ***PUSH IP***
IP ← reg16

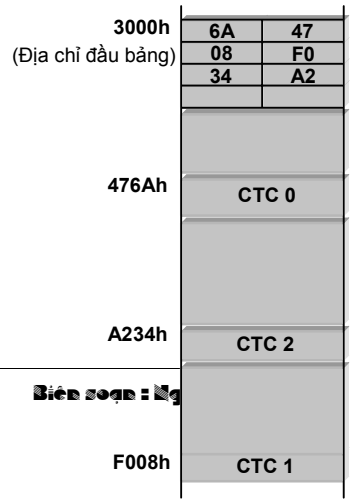
♣ *mem16* : ***PUSH IP***
IP ← [địa chỉ +1,địa chỉ]

♣ *mem32* : ***PUSH CS***
PUSH IP
CS ← [địa chỉ +3,địa chỉ+2]
IP ← [địa chỉ +1,địa chỉ]

- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Gọi chương trình con. Quá trình gọi chương trình con được thực hiện qua 2 bước :
  - ♣ Cất địa chỉ trở về - chính là địa chỉ lệnh ngay sau lệnh CALL - vào chồng.
  - ♣ Chuyển sự thi hành chương trình đến địa chỉ lệnh đầu tiên của chương trình con.
- Địa chỉ trở về chính là nội dung hiện tại của cặp thanh ghi CS:IP.
- Lệnh gọi trực tiếp đến nhãn nearlabel chỉ cất nội dung IP, và nạp giá trị offset mới vào IP (nội dung CS không đổi) nên chỉ có thể dùng để gọi bên trong một segment. Lệnh này còn được gọi là lệnh gọi gần hay gọi trong segment. Nhãn nearlabel còn được gọi là nhãn gần và có kích thước 2 byte.
  Ví dụ : CALL 0F008h
- Lệnh gọi trực tiếp đến nhãn farlabel cất nội dung IP lẫn CS, sau đó nạp giá trị offset mới vào IP, nạp giá trị segment mới vào CS nên có thể dùng để gọi đến bất kỳ vị trí bộ nhớ nào cũng được. Lệnh này còn được gọi là lệnh gọi xa hay gọi ngoài segment. Nhãn farlabel còn được gọi là nhãn xa và có kích thước 4 byte.
  Ví dụ : CALL 3000:F008
- Lệnh gọi gián tiếp qua thanh ghi reg16 cũng là một lệnh gọi gần. Lúc đó địa chỉ chương trình con được nạp vào thanh ghi trước khi thực hiện lệnh gọi.
  Ví dụ : MOV DX,0F008h
  CALL DX
- Lệnh gọi gián tiếp qua bộ nhớ mem16 cũng là một lệnh gọi gần. Lúc đó địa chỉ chương trình con được đặt tại ô nhớ có địa chỉ hiệu dụng trong lệnh.
  Ví dụ : CALL [BX+3000h]
- Lệnh gọi gián tiếp qua bộ nhớ mem32 là một lệnh gọi xa. Lúc đó địa chỉ chương trình con đặt tại ô nhớ có địa chỉ hiệu dụng trong lệnh phải là địa chỉ 4 byte. Lúc đó cần phải chỉ rõ ra hoạt động bộ nhớ 32 bit bằng cách dùng toán tử DWORD PTR.
  Ví dụ : CALL DWORD PTR [SI+2000h]
- Với lệnh gọi gián tiếp qua bộ nhớ ta có thể tổ chức sắp xếp các địa chỉ chương trình con thành một bảng trong bộ nhớ gọi là bảng nhảy. Lúc đó mỗi chương trình con sẽ được gọi theo số thứ tự của nó trong bảng nhảy.
- *Ví dụ bảng nhảy gần :*



Chương trình con 0 ở địa chỉ 476Ah.
Chương trình con 1 ở địa chỉ 0F008h.
Chương trình con 2 ở địa chỉ 0A234h.

Để gọi chương trình con 2 ta thực hiện :

```
MOV    BX,2      ; số thứ tự chương trình con.
ADD    BX,BX     ; nhân 2.
CALL   [BX+3000h]  ; gọi chương trình con.
```

6.2 Lệnh JMP :

- Dạng lệnh :

JMP	*shortlabel*	JMP	*mem16*
JMP	*nearlabel*	JMP	*mem32*
JMP	*farlabel*	JMP	*reg16*

- Giải thích :
  - ♣ *shortlabel* : ***IP ← IP + độ dời (mở rộng dấu 16 bit)***
  - ♣ *nearlabel* : ***IP ← địa chỉ***
  - ♣ *farlabel* : ***CS ← địa chỉ segment***
    IP ← địa chỉ offset
  - ♣ *reg16 :* ***IP ← reg16***
  - ♣ *mem16* : ***IP ← [địa chỉ +1,địa chỉ]***
  - ♣ *mem32* : ***CS ← [địa chỉ +3,địa chỉ+2]***
    IP ← [địa chỉ +1,địa chỉ]
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nhảy không điều kiện. Lệnh nhảy không điều kiện thực hiện giống như lệnh gọi nhưng không có bước lưu lại địa chỉ trở về.
- Lệnh nhảy đến nhãn ngắn shortlabel là lệnh nhảy tương đối. Nơi đến phải nằm trong phạm vi từ -128 đến +127 so với vị trí của lệnh nhảy. Toán hạng nguồn trong lệnh chỉ là byte độ dời để cộng thêm vào thanh ghi IP. Byte độ dời này được mở rộng dấu trước khi cộng vào thanh ghi IP.
- *Ví dụ :*

```
JMP   SHORT   18h
JMP   0F008h
JMP   DWORD   PTR   [3000h]
```

6.3 Lệnh RET :

- Dạng lệnh :

RET		RETF	
RET	*immed8*	RETF	*immed8*

- Giải thích :
  - ♣ *RET* : ***POP IP***
  - ♣ *RETF* : ***POP IP***
    POP CS
  - ♣ *RET immed8* : ***POP IP***
    SP ← SP + immed8
  - ♣ *RETF immed8* : ***POP IP***
    POP CS
    SP ← SP + immed8
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Trở về từ chương trình con. Lệnh trở về là lệnh dùng để kết thúc một chương trình con.
- Lệnh RET để kết thúc một chương trình con gần.
- Lệnh RETF để kết thúc một chương trình con xa.

- Dạng lệnh trở về có toán hạng *immed8* dùng cho các chương trình con có sử dụng thông số trong chồng. Khi đó, toán hạng nguồn *immed8* sẽ được cộng vào thanh ghi SP để chỉnh lại vị trí đỉnh chồng sau khi gọi chương trình con, tránh thất thoát bộ nhớ dùng cho chồng.

6.4 Lệnh nhảy có điều kiện :

- Dạng lệnh : J*cond* *shortlabel*
- Giải thích : ***Nếu thỏa điều kiện thì nhảy tương đối***
  IP ← địa chỉ lệnh kế + độ dời (mở rộng dấu 16 bit)
  ngược lại không làm gì cả (qua lệnh kế).
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lệnh nhảy có điều kiện dùng trạng thái các cờ để làm điều kiện.
- Sau đây là bảng mã lệnh nhảy có điều kiện cùng với điều kiện nhảy.

Mã lệnh	Giải thích	Điều kiện
JE/JZ	**Nhảy nếu bằng/không**	**ZF = 1**
JL/JNGE	**Nhảy nếu nhỏ hơn/không lớn hơn hoặc bằng**	**(SF xor OF) = 1**
JLE/JNG	**Nhảy nếu nhỏ hơn hoặc bằng /không lớn hơn**	**((SF xor OF) or ZF) = 1**
JB/JNAE/JC	**Nhảy nếu dưới /không trên hoặc bằng/nhớ**	**CF = 1**
JBE/JNA	**Nhảy nếu dưới hoặc bằng /không trên**	**(CF or ZF) = 1**
JP/JPE	**Nhảy nếu kiểm tra / kiểm tra chẵn**	**PF = 1**
JO	**Nhảy nếu tràn**	**OF = 1**
JS	**Nhảy nếu dấu**	**SF = 1**
JNE/JNZ	**Nhảy nếu không bằng/khác không**	**ZF = 0**
JNL/JGE	**Nhảy nếu không nhỏ hơn/lớn hơn hoặc bằng**	**(SF xor OF) = 0**
JNLE/JG	**Nhảy nếu không nhỏ hơn hoặc bằng /lớn hơn**	**((SF xor OF) or ZF) = 0**
JNB/JAE/JNC	**Nhảy nếu không dưới /trên hoặc bằng/không nhớ**	**CF = 0**
JNBE/JA	**Nhảy nếu không dưới hoặc bằng /trên**	**(CF or ZF) = 0**
JNP/JPO	**Nhảy nếu không kiểm tra / kiểm tra lẻ**	**PF = 0**
JNO	**Nhảy nếu không tràn**	**OF = 0**
JNS	**Nhảy nếu không dấu**	**SF = 0**

- *Ví dụ :*

```
        MOV   CX,3          ; thực hiện một vòng lặp làm 3 lần.
        MOV   AX,0
Nhan:   ADD   AX,12
        DEC   CX
        JNZ   Nhan          ; nhảy đến lệnh tại vị trí "Nhan" nếu CX ≠ 0.
        MOV   [3000h],AX
```

6.5 Lệnh LOOP :

♣Dạng lệnh 1 : LOOP *shortlabel*

- Giải thích : ***giảm CX, lặp vòng (nhảy) nếu CX ≠ 0***
  IP ← địa chỉ lệnh kế + độ dời (mở rộng dấu 16 bit)
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lập vòng không điều kiện. CX giữ số lần lặp. Rất tiện dụng trong việc tạo ra các vòng lặp. Chẳng hạn như ví dụ trong phần lệnh nhảy có điều kiện có thể viết lại gọn hơn như sau :

```
        MOV   CX,3
        MOV   AX,0
Nhan:   ADD   AX,12
```

```
            LOOP   Nhan
            MOV    [3000h],AX
```

- Một trong những ứng dụng phổ biến của lệnh lặp vòng là tạo ra các vòng làm trễ.
- *Ví dụ :*

```
      MOV    CX,10
Nhan: LOOP   Nhan     ; vòng lặp làm 10 lần lệnh LOOP

      MOV    CX,0
Nhan: LOOP   Nhan     ; vòng lặp làm 65536 lần lệnh LOOP
```

♣Dạng lệnh 2 : LOOPE/LOOPZ *shortlabel*

- Giải thích : ***giảm CX, lặp vòng (nhảy) nếu CX ≠ 0 và ZF = 1***
  IP ← địa chỉ lệnh kế + độ dời (mở rộng dấu 16 bit)
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lập vòng nếu bằng / nếu không. Hai điều kiện CX ≠ 0 *và* ZF = 1 phải thỏa đồng thời thì lệnh mới lặp vòng. Nếu không, không làm gì cả (qua lệnh kế).
- Đôi khi người ta xét các điều kiện để không lặp còn gọi là điều kiện thoát khỏi vòng lặp : CX = 0 hay ZF = 0. Có thể xem đó là các nguyên nhân gây ra kết thúc vòng lặp.

♣Dạng lệnh 3 : LOOPNE/LOOPNZ *shortlabel*

- Giải thích : ***giảm CX, lặp vòng (nhảy) nếu CX ≠ 0 và ZF = 0***
  IP← địa chỉ lệnh kế + độ dời (mở rộng dấu 16 bit)
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Lập vÎng näu khéng bÙng / näu kh¾c khéng. Hai ½iảu kiàn CX ≠ 0 và ZF = 0 phải thỏa đồng thời thì lệnh mới lặp vòng. Nếu không, không làm gì cả (qua lệnh kế).
- Đôi khi người ta xét các điều kiện để không lặp còn gọi là điều kiện thoát khỏi vòng lặp : CX = 0 hay ZF = 1. Có thể xem đó là các nguyên nhân gây ra kết thúc vòng lặp.

6.6 Lệnh JCXZ :

- Dạng lệnh : JCXZ *shortlabel*
- Giải thích : ***Nếu CX = 0 thì***
  IP ← địa chỉ lệnh kế + độ dời (mở rộng dấu 16 bit)
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Nhảy nếu CX=0. Thường dùng sau LOOPE, LOOPNE, REPE, REPNE để xác định nguyên nhân kết thúc vòng lặp.
- *Ví dụ :*

```
      REPE  CMPSB       ; so sánh 2 chuỗi
      JCXZ  nhan
      ----------------------------------------------------------
      ( đoạn chương trình xử lý cho trường hợp chuỗi khác nhau)
      ----------------------------------------------------------
nhan: ( đoạn chương trình xử lý cho trường hợp chuỗi giống nhau)
```

6.7 Lệnh INT :

- Dạng lệnh : INT *immed8*

INT *3*

- Giải thích : ***PUSHF***
  PUSH CS
  PUSH IP
  CS ← [(số ngắt * 4)+3, (số ngắt * 4)+2]
  IP ← [(số ngắt * 4)+1, (số ngắt * 4)]
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
		0					

- Ngắt quãng mềm. Thực chất của lệnh ngắt quãng là gọi đến một chương trình con đặc biệt gọi là chương trình phục vụ ngắt quãng.
- Cách thực hiện lệnh ngắt quãng chính là cách gọi xa gián tiếp qua bộ nhớ 32 bit.
- Số ngắt 1 byte *immed8* cung cấp trong lệnh chính là số thứ tự của chương trình con phục vụ ngắt quãng. Nhờ vậy nên mặc dù lệnh ngắt quãng là lệnh gọi xa nhưng lại rất ngắn.
- Bảng nhảy trong trường hợp này được gọi là bảng vector ngắt quãng. Vị trí của vector ngắt quãng được xác định bằng cách lấy số ngắt nhân 4. Kết quả này có thể xem là địa chỉ vật lý cũng được hoặc là địa chỉ offset lấy theo segment 0000 cũng được.
- Điểm khác biệt giữa lệnh ngắt quãng và lệnh gọi xa là thao tác cất thanh ghi trạng thái (cờ) vào chồng PUSHF. Chính vì thế nên chương trình con phục vụ ngắt quãng phải được kết thúc bằng một lệnh trở về khác là IRET.
- Các chương trình con phục vụ ngắt quãng thường được dùng cho các chương trình hệ thống (hệ điều hành, chương trình giao tiếp với các thiết bị, các chương trình con sử dụng thường xuyên, ...) hơn là dùng cho chương trình của người sử dụng.
- Số ngắt cũng theo qui ước của hệ thống như sau :
  00h ÷ 07h : ngắt hệ thống.
  08h ÷ 0Fh, 70h ÷ 77h : ngắt cứng.
  Còn lại : ngắt mềm.
- Một số ngắt thông dụng :
  INT 10h : màn hình.
  INT 13h : đĩa.
  INT 14h : thông tin liên lạc.
  INT 16h : bàn phím.
  INT 17h : máy in
  INT 21h : các phục vụ của MS-DOS.
  INT 20h : kết thúc chương trình, trở về DOS.

- Mỗi chương trình con phục vụ ngắt quãng có thể thực hiện nhiều chức năng bên trong nghĩa là các phục vụ được chia nhỏ ra nữa. Ví dụ ngắt phục vụ màn hình có chức năng chọn chế độ màn hình, chức năng định vị điểm nháy (cursor), chức năng xuất ký tự ra màn hình, chức năng đồ họa, . . .
- Thông số của chương trình phục vụ ngắt quãng thường được truyền thông qua các thanh ghi đầu vào (input) và k?t qu? thi hành chương trình con sẽ giữ trong các thanh ghi đầu ra (output).
- *Ví dụ :* dùng ngắt 17h, chức năng 0 để xuất ký tự ra máy in.
  Input : AH = số chức năng (trong trường hợp này là 0).
  AL = ký tự cần in (mã ASCII).
  DX = số thứ tự máy in (0=LPT1, 1=LPT2, . . .)
  Output : AL = trạng thái
  Bit Ý nghĩa (nếu = 1)

0	Quá thời gian đợi máy in.
1,2	Không dùng.
3	Lỗi xuất nhập.
4	Máy in đang được chọn.
5	Hết giấy.
6	Máy in đã nhận ký tự.
7	Máy in không bận.

- Như vậy để in ký tự 'A' ra máy in ta viết đoạn chương trình sau :

```
MOV  AH,0          ; nạp số chức năng.
MOV  AL,041h       ; mã ASCII của ký tự 'A'.
MOV  DX,0          ; nạp số thứ tự máy in LPT1.
INT  17h           ; gọi ngắt 17h để in.
MOV  [3000h],AL    ; cất trạng thái máy in.
```

6.8 Lệnh INTO :

- Dạng lệnh : INTO
- Giải thích : ***PUSHF***
  PUSH CS
  PUSH IP
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
		×	×				

- Ngắt quãng nếu tràn (OF = 1).

6.9 Lệnh IRET :

- Dạng lệnh : IRET
- Giải thích : ***POP IP***
  POP CS
  POPF
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
×	×	×	×	×	×	×	×

- Trở về từ chương trình phục vụ ngắt quãng.

7. Điều khiển bộ xử lý :

7.1 Lệnh CLC :

- Dạng lệnh : CLC
- Giải thích : ***CF ← 0***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
							0

- Xóa cờ nhớ về 0.

7.2 Lệnh STC :

- Dạng lệnh : STC
- Giải thích : ***CF ← 1***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
							1

- Lập cờ nhớ lên 1.

7.3 Lệnh CMC :

- Dạng lệnh : CMC
- Giải thích : ***CF ← bù 1 của CF***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
							×

- Lấy bù cờ nhớ.

7.4 Lệnh NOP :
- Dạng lệnh : NOP
- Giải thích : ***không làm gì cả***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
							0

- Không làm gì cả. Dùng để tạo ra các khoảng làm trễ ngắn.

7.5 Lệnh CLD :
- Dạng lệnh : CLD
- Giải thích : ***DF ← 0***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
	0						

- Xóa cờ định hướng về 0. Xử lý tăng địa chỉ trong các lệnh xử lý chuỗi.

7.6 Lệnh STD :
- Dạng lệnh : STD
- Giải thích : ***DF ← 1***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
	1						

- Lập cờ định hướng lên 1. Xử lý giảm địa chỉ trong các lệnh xử lý chuỗi.

7.7 Lệnh CLI :
- Dạng lệnh : CLI
- Giải thích : ***IF ← 0***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
		0					

- Xóa cờ ngắt quãng về 0. Cấm ngắt quãng cứng.

7.8 Lệnh STI :
- Dạng lệnh : STI
- Giải thích : ***IF ← 1***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF
		1					

- Lập cờ ngắt quãng lên 1. Cho phép ngắt quãng cứng.

7.9 Lệnh HLT :
- Dạng lệnh : HLT
- Giải thích : ***CPU vào trạng thái dừng.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Dừng CPU, chờ một ngắt quãng cứng xảy ra (INTR hay NMI).

7.10 Lệnh WAIT :
- Dạng lệnh : WAIT
- Giải thích : ***CPU vào trạng thái đợi.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- CPU vào trạng thái đợi cho đến khi ngỏ TEST tác động.

7.11 Tiếp đầu lệnh LOCK :

- Dạng lệnh : LOCK *lệnh*
- Giải thích : ***Khóa các tuyến trong khi thi hành lệnh theo sau.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Khóa các tuyến khi thi hành lệnh theo sau. Không cho phép các vi xử lý khác yêu cầu tuyến (chẳng hạn DMA).

7.12 Lệnh ESC :

- Dạng lệnh : ESC *immed,reg*
  ESC *immed,mem*
- Giải thích : ***đưa lệnh ra tuyến dữ liệu.***
- Tác động cờ :

OF	DF	IF	SF	ZF	AF	PF	CF

- Phát ra một lệnh cho vi mạch đồng xử lý 8087.
- *Ví dụ :* ESC 6,AL
  ESC 4,[2000h]