



ĐIỆN TỬ -
VIỄN THÔNG

Kỹ thuật vi xử lý Microprocessors

Người hướng dẫn:
PGS.TS. HOÀNG MẠNH THẮNG



Your instructor

- Họ tên: **HOÀNG MẠNH THẮNG**
- Bộ môn: Điện tử & Kỹ thuật máy tính
 - Office: C9-401
 - Email: hmtthang01@hust.edu.vn
- Member of Intel Higher Education Program, IEEE, IEICE
- Research:
 - Chaotic communications, nonlinear circuits, laser's dynamics
 - ...
- Education:
 - Đại học: K38 Điện tử-Viễn Thông, ĐHBK Hà nội (1998)
 - Master về Điện tử-Viễn thông ĐHBK Hà nội (2001),
 - ⇒ Đề tài: Phân tích các đặc tính của Laser buồng cộng hưởng đứng
 - Tiến sĩ kỹ thuật chuyên ngành Thông tin và điều khiển, 8/2007, Đại học Công Nghệ Nagaoka, Nhật bản
 - ⇒ Đề tài: Đồng bộ hỗn loạn và ứng dụng của nó trong truyền tin bảo mật



Nội dung môn học

- 1. Giới thiệu chung về hệ vi xử lý**
- 2. Bộ vi xử lý Intel 8088/8086**
- 3. Lập trình hợp ngữ cho 8086**
- 4. Tổ chức vào ra dữ liệu**
- 5. Ngắt và xử lý ngắt**
- 6. Truy cập bộ nhớ trực tiếp (Direct Memory Access)**
- 7. Các bộ vi xử lý trên thực tế**



Tài liệu tham khảo

- Slides
- Barry B. Brey, **The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium and Pentium Pro Processor: Architecture, Programming, and Interfacing, Fourth Edition**, Prentice Hall, 1997.
- Văn Thé Minh, **Kỹ thuật vi xử lý**, Nhà xuất bản giáo dục, 1997.
- Quách Tuấn Ngọc và cộng sự, **Ngôn ngữ lập trình Assembly và máy vi tính IBM-PC**, 2 tập, Nhà xuất bản giáo dục, 1995.
- Cảm ơn giáo sư Rudy Lauwereins đã cho phép sử dụng slides của ông



Mục đích của môn học

- Nắm được cấu trúc, nguyên lý hoạt động của bộ vi xử lý và hệ vi xử lý
- Có khả năng lập trình bằng hợp ngữ cho vi xử lý
- Có khả năng lựa chọn vi xử lý thích hợp cho các ứng dụng cụ thể
- Nắm được các bộ vi xử lý trên thực tế



Bài tập lớn và thi

- **Bài tập lớn (30% điểm)**
 - Thiết kế một hệ thống sử dụng vi xử lý (vi điều khiển, DSP...) hoặc
 - Thiết kế hệ thống card ngoại vi cho máy tính
 - Không được thi lần 1, 2 nếu không làm bài tập lớn
- **Điểm giữa kỳ (30%)**
 - Thi giữa kỳ, và
 - Điểm chuyên cần (tính vào điểm giữa kỳ, theo quy chế), thông qua, điểm danh
- **Thi cuối kỳ (40%)**
 1. Lý thuyết: Xem mục đích của môn học
 2. Lập trình hợp ngữ
 3. Thiết kế bộ nhớ và thiết bị ngoại vi cho hệ vi xử lý
 4. Điểm chuyên cần (tính vào điểm giữa kỳ, theo quy chế), thông qua, điểm danh

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính**
- 1.2 Phân loại vi xử lý**
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)**
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính**
 - 1.1.1 Thế hệ -1: Thời xa xưa (....-1642)
 - 1.1.2 Thế hệ 0: Máy tính cơ khí (1642-1945)
 - 1.1.3 Thế hệ 1: Đèn điện tử-Vacuum tubes (1945-1955)
 - 1.1.4 Thế hệ 2: Transistor rời rạc-Discrete transistors (1955-1965)
 - 1.1.5 Thế hệ 3: Mạch tích hợp-Integrated circuits (1965-1980)
 - 1.1.6 Thế hệ 4: Mạch tích hợp cỡ lớn-VLSI (1980-?)
- 1.2 Phân loại vi xử lý**
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)**
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

Chương 1

Giới thiệu chung về hệ vi xử lý

1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính

- 1.1.1 Thế hệ -1: Thời xa xưa (...-1642)
- 1.1.2 Thế hệ 0: Máy tính cơ khí (1642-1945)
- 1.1.3 Thế hệ 1: Đèn điện tử-Vacuum tubes (1945-1955)
- 1.1.4 Thế hệ 2: Transistor rời rạc-Discrete transistors (1955-1965)
- 1.1.5 Thế hệ 3: Mạch tích hợp-Integrated circuits (1965-1980)
- 1.1.6 Thế hệ 4: Mạch tích hợp cỡ lớn-VLSI (1980-?)

1.2 Phân loại vi xử lý

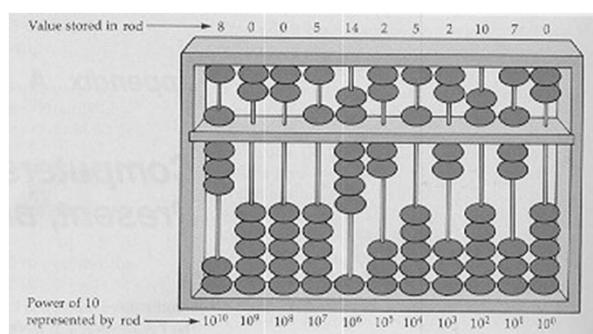
1.3 Các hệ đếm dùng trong máy tính (nhắc lại)

1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

9

Thế hệ -1: The early days (...-1642)

- **Bàn tính, abaci, đã được sử dụng để tính toán. Khái niệm về giá trị theo vị trí đã được sử dụng**



10

5

Thế hệ -1: The early days (...-1642)

11/Chapter1



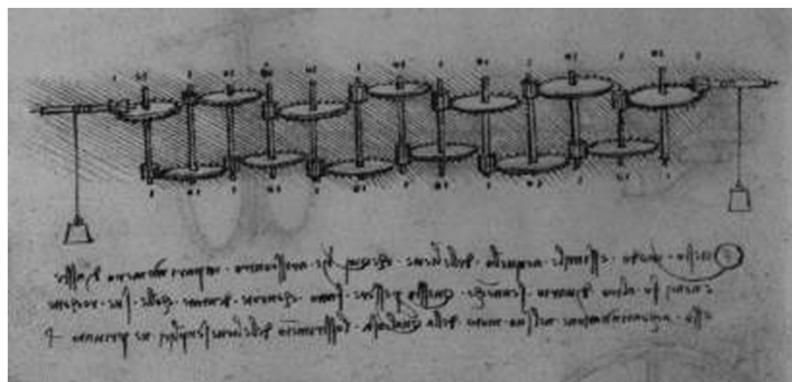
- **Thế kỷ 12: Muhammad ibn Musa Al'Khowarizmi đưa ra khái niệm về giải thuật algorithm**
- Một danh sách các chỉ dẫn mô tả một cách chính xác các bước của một quá trình mà đảm bảo là quá trình này sẽ phải kết thúc sau một số bước nhất định với câu trả lời đúng cho từng trường hợp cụ thể của một vấn đề cần giải quyết

11

Thế hệ -1: The early days (...-1642)

12/Chapter1

- Codex Madrid - Leonardo Da Vinci (1500)
 - Vẽ một cái máy tính cơ khí



12

6

Chương 1

Giới thiệu chung về hệ vi xử lý

1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính

- 1.1.1 Thế hệ -1: Thời xa xưa (...-1642)
- 1.1.2 Thế hệ 0: Máy tính cơ khí (1642-1945)
- 1.1.3 Thế hệ 1: Đèn điện tử-Vacuum tubes (1945-1955)
- 1.1.4 Thế hệ 2: Transistor rời rạc-Discrete transistors (1955-1965)
- 1.1.5 Thế hệ 3: Mạch tích hợp-Integrated circuits (1965-1980)
- 1.1.6 Thế hệ 4: Mạch tích hợp cỡ lớn-VLSI (1980-?)

1.2 Phân loại vi xử lý

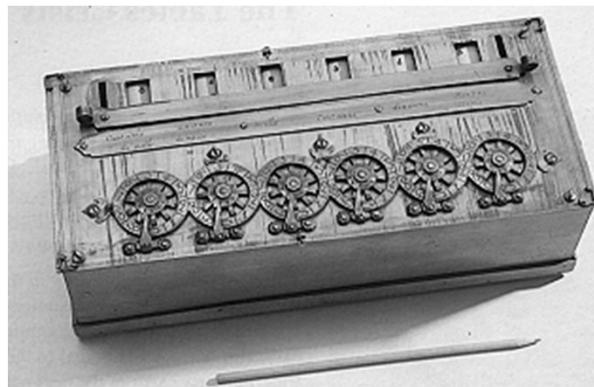
1.3 Các hệ đếm dùng trong máy tính (nhắc lại)

1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

13

Thế hệ 0: Mechanical (1642-1945)

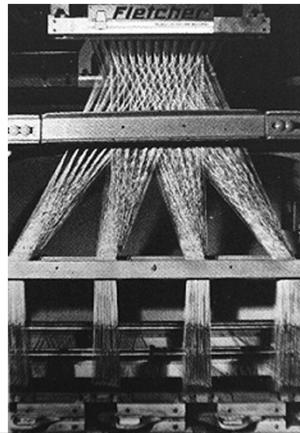
- Blaise Pascal, con trai của một người thu thuế, đã chế tạo một máy cộng có nhớ vào năm 1642



14

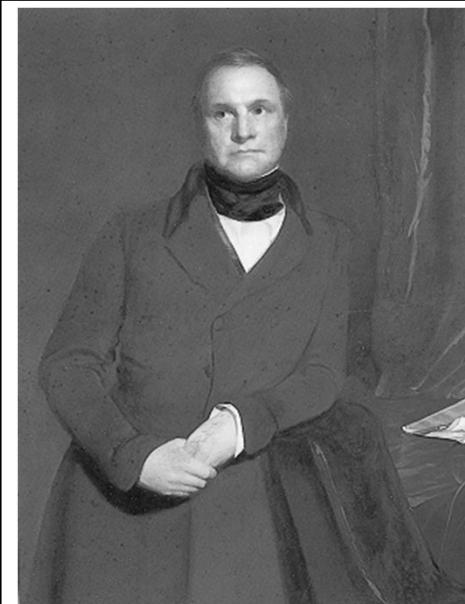
Thẻ hệ 0: Mechanical (1642-1945)

- Năm 1801, Joseph-Marie Jacquard đã phát minh ra máy dệt tự động sử dụng bìa đục lỗ để điều khiển hoạ tiết dệt trên vải
- Bìa đục lỗ lưu trữ chương trình: máy đa năng đầu tiên



15

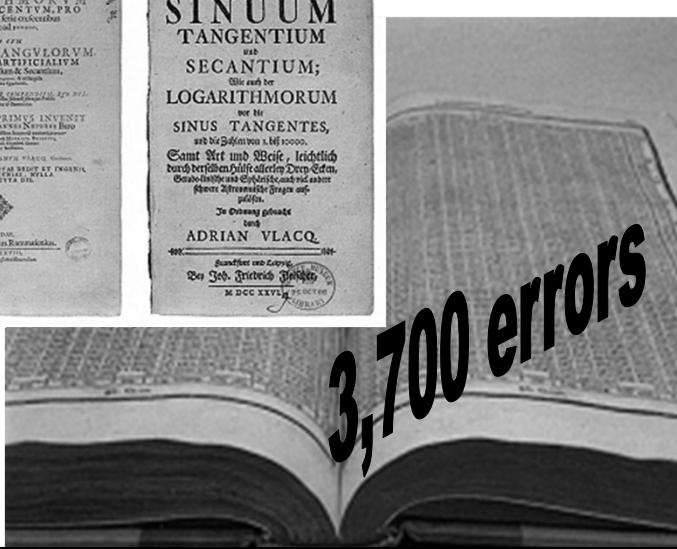
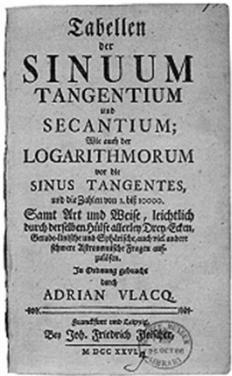
Thẻ hệ 0: Mechanical (1642-1945)



- 1822, Charles Babbage nhận ra rằng các bảng tính dùng trong hàng hải có quá nhiều lỗi dẫn tới việc rất nhiều tàu bị mất tích
- Ông đã xin chính phủ Anh hỗ trợ để nghiên cứu về máy tính

16

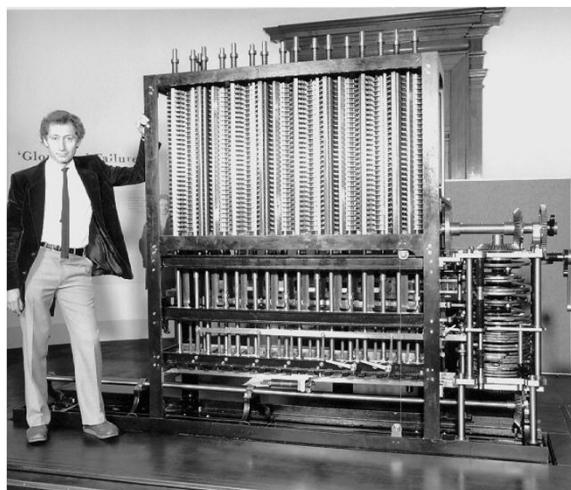
Thé hệ 0: Mechanical (1642-1945)



17

Thé hệ 0: Mechanical (1642-1945)

- Babbage đã thiết kế một cái máy vi phân Difference Engine để thay thế toàn bộ bảng tính: máy thực hiện một ứng dụng cụ thể đầu tiên (application specific hard-coded machine)



18

Thé hệ 0: Mechanical (1642-1945)

- Ada Augusta King, trở thành lập trình viên đầu tiên vào năm 1842 khi cô viết chương trình cho Analytical Engine, thiết bị thứ 2 của Babbage**



19

Thé hệ 0: Mechanical (1642-1945)

- Herman Hollerith, người Mỹ, thiết kế một máy tính để xử lý dữ liệu về dân số Mỹ 1890**
- Ông thành lập công ty, Hollerith Tabulating Company, sau đây là Calculating-Tabulating-Recording (C-T-R) company vào năm 1914 và sau này được đổi tên là IBM vào năm 1924.**

20

10

Thé hệ 0: Mechanical (1642-1945)

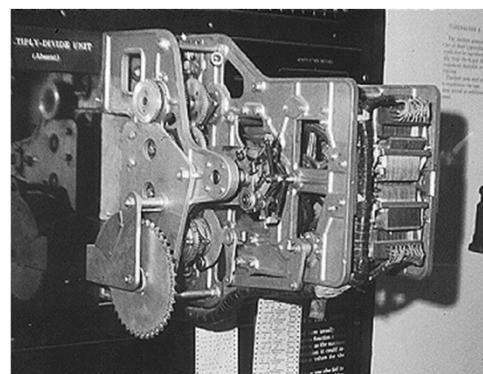
- Konrad Zuse, Berlin, Đức, phát triển vào năm 1935 máy tính Z-1 sử dụng rơ le và số nhị phân
- Chu kỳ lệnh: 6 giây (0.17 Hz)



21

Thé hệ 0: Mechanical (1642-1945)

- Máy tính cơ điện tự động lớn đầu tiên là máy Harvard Mark I (IBM Automatic Sequence Control Calculator), phát minh bởi Howard Aiken vào cuối 1930
- ASCC không phải là máy tính có chương trình lưu trữ sẵn mà các lệnh được ghi vào các băng giấy.



22

11



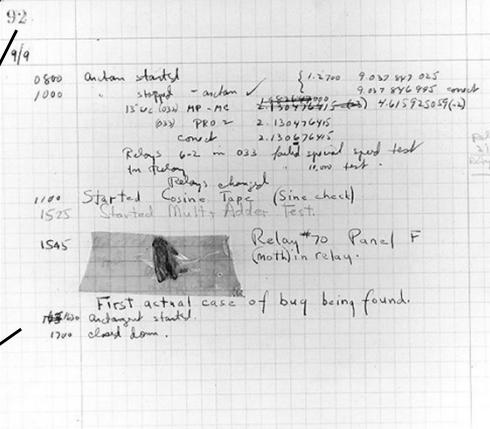
Thế hệ 0: Mechanical (1642-1945)

- Grace Murray Hopper found the first computer bug beaten to death in the jaws of a relay. She glued it into the logbook of the computer and thereafter when the machine stops (frequently) she told Howard Aiken that they are "debugging" the computer.

**Numbered pages
for USA patents**



Lab book!!



Chương 1

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính
 - 1.1.1 Thế hệ -1: Thời xa xưa (....-1642)
 - 1.1.2 Thế hệ 0: Máy tính cơ khí (1642-1945)
 - 1.1.3 Thế hệ 1: Đèn điện tử-Vacuum tubes (1945-1955)
 - 1.1.4 Thế hệ 2: Transistor rời rạc-Discrete transistors (1955-1965)
 - 1.1.5 Thế hệ 3: Mạch tích hợp-Integrated circuits (1965-1980)
 - 1.1.6 Thế hệ 4: Mạch tích hợp cỡ lớn-VLSI (1980-?)
- 1.2 Phân loại vi xử lý
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

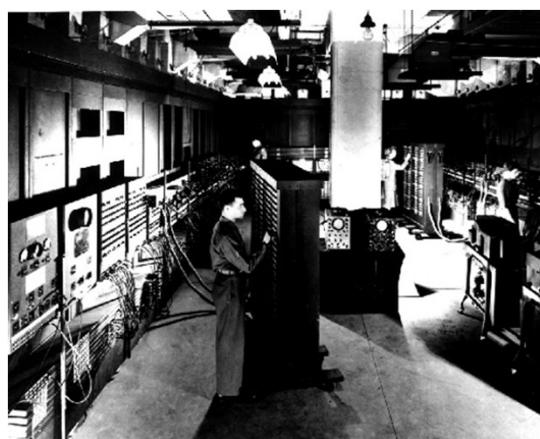
Thế hệ 1: Đèn điện tử (1945-1955)



- Năm 1943, John Mauchly và J. Presper Eckert bắt đầu nghiên cứu về ENIAC

25

Thế hệ 1: Đèn điện tử (1945-1955)

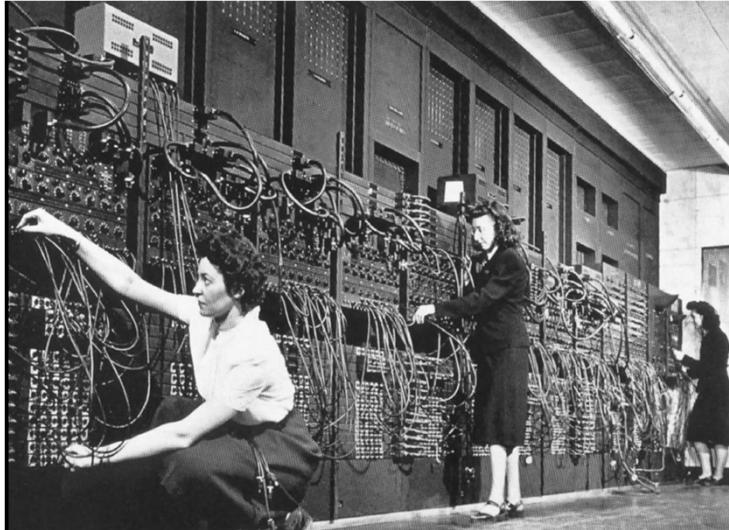


- 18000 đèn điện tử, 1500 rơ le, 30 tấn, 140 kW, 20 thanh ghi 10 chữ số thập phân, 100 nghìn phép tính/ giây
- “Trong tương lai máy tính sẽ nặng tối đa là 1.5 tấn” (Popular Mechanics, 1949)

26

13

Thế hệ 1: Đèn điện tử(1945-1955)



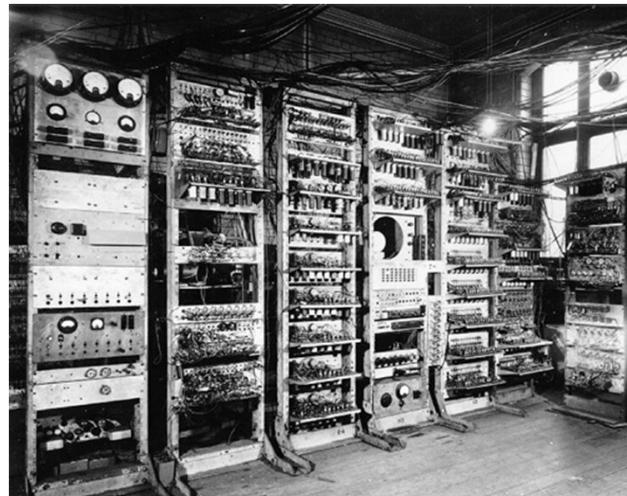
- Lập trình thông qua 6000 công tắc nhiều nấc và nặng hàng tấn

Thế hệ 1: Đèn điện tử (1945-1955)

- Năm 1946, John von Neumann phát minh ra máy tính có chương trình lưu trong bộ nhớ
- Máy tính của ông gồm có một đơn vị điều khiển, một đơn vị xử lý số học và logic (ALU), một bộ nhớ chương trình và dữ liệu và sử dụng số nhị phân thay vì số thập phân.
- Máy tính ngày nay đều có cấu trúc von Neumann
- ông đặt nền móng cho hiện tượng “von Neumann bottleneck”, sự không tương thích giữa tốc độ của bộ nhớ với đơn vị xử lý

Thế hệ 1: Đèn điện tử (1945-1955)

- Năm 1948, máy tính có chương trình lưu trữ trong bộ nhớ đầu tiên được vận hành tại trường đại học Manchester: Manchester Mark I



29

Thế hệ 1: Đèn điện tử (1945-1955)

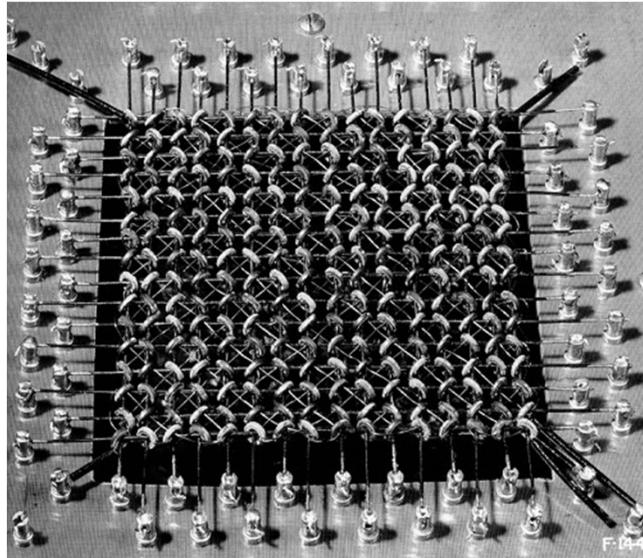
- Năm 1951, máy tính Whirlwind lần đầu tiên sử dụng bộ nhớ lõi từ (magnetic core memories). Gần đây nguyên lý này đã được sử dụng lại để chế tạo MRAM ở dạng tích hợp.



30

Thẻ hệ 1: Đèn điện tử (1945-1955)

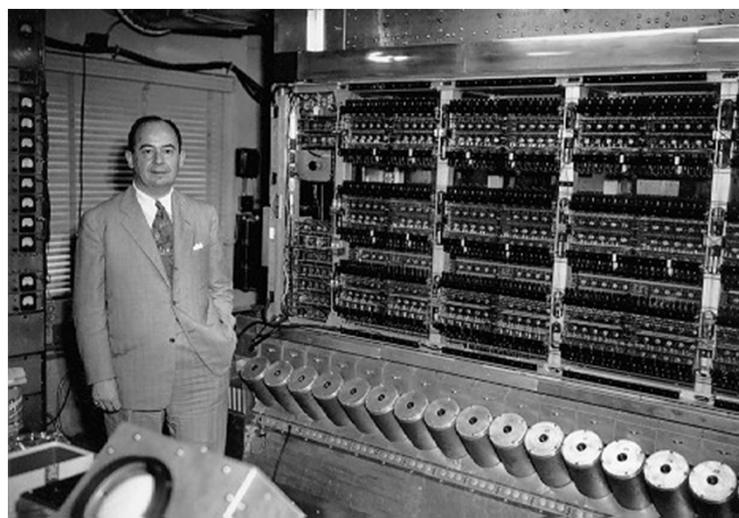
- Một magnetic core lưu trữ 256 bits



31

Thẻ hệ 1: Đèn điện tử (1945-1955)

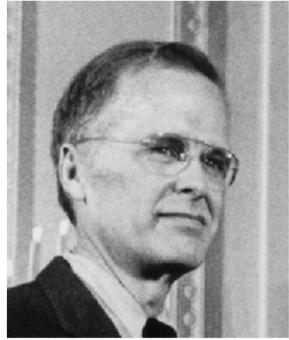
- John von Neumann năm 1952 với chiếc máy tính mới của ông



32

Thế hệ 1: Đèn điện tử (1945-1955)

- Năm 1954, John Backus, IBM phát minh ra FORTRAN



33

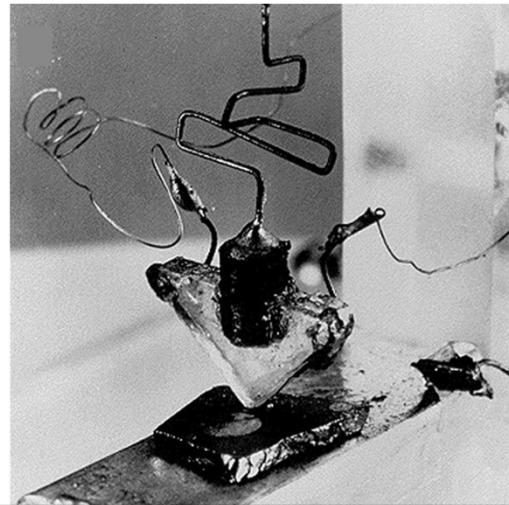
Chương 1 Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính
 - 1.1.1 Thế hệ -1: Thời xa xưa (....-1642)
 - 1.1.2 Thế hệ 0: Máy tính cơ khí (1642-1945)
 - 1.1.3 Thế hệ 1: Đèn điện tử-Vacuum tubes (1945-1955)
 - 1.1.4 Thế hệ 2: Transistor rời rạc-Discrete transistors (1955-1965)
 - 1.1.5 Thế hệ 3: Mạch tích hợp-Integrated circuits (1965-1980)
 - 1.1.6 Thế hệ 4: Mạch tích hợp cỡ lớn-VLSI (1980-?)
- 1.2 Phân loại vi xử lý
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

34

Thế hệ 2: Discrete transistors (1955-1965)

- Năm 1947, William Shockley, John Bardeen, and Walter Brattain phát minh ra transistor



35

Thế hệ 2: Discrete transistors (1955-1965)

- Năm 1955, IBM công bố IBM704, máy tính mainframe sử dụng tranzistor
- Đây là máy tính với phép toán dấu phẩy động đầu tiên (5 kFlops, clock: 300 kHz)



36

Chương 1

Giới thiệu chung về hệ vi xử lý

1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính

- 1.1.1 Thế hệ -1: Thời xa xưa (...-1642)
- 1.1.2 Thế hệ 0: Máy tính cơ khí (1642-1945)
- 1.1.3 Thế hệ 1: Đèn điện tử-Vacuum tubes (1945-1955)
- 1.1.4 Thế hệ 2: Transistor rời rạc-Discrete transistors (1955-1965)
- 1.1.5 Thế hệ 3: Mạch tích hợp-Integrated circuits (1965-1980)
- 1.1.6 Thế hệ 4: Mạch tích hợp cỡ lớn-VLSI (1980-?)

1.2 Phân loại vi xử lý

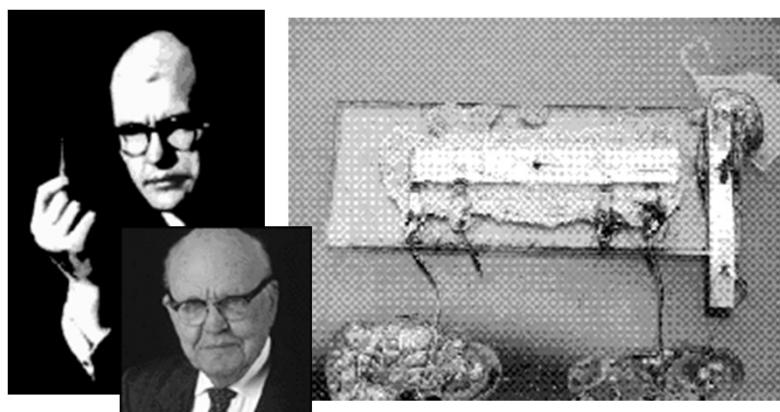
1.3 Các hệ đếm dùng trong máy tính (nhắc lại)

1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

37

Thế hệ 3: Integrated circuits (1965-1980)

- Năm 1958, Jack St. Clair Kilby of Texas Instruments (Nobel prize physics, 2000) đưa ra và chứng minh ý tưởng tích hợp 1 transistor với các điện trở và tụ điện trên một chip bán dẫn với kích thước 1 nửa cái kẹp giấy. Đây chính là IC.



38

19

Thé hệ 3: Mạch tích hợp (1965-1980)

- 7/4/1964 IBM đưa ra System/360, họ máy tính tương thích đầu tiên của IBM



39

Thé hệ 3: Mạch tích hợp (1965-1980)

- Năm 1965, Digital Equipment Corporation, đưa ra chiếc máy tính mini đầu tiên DP-8

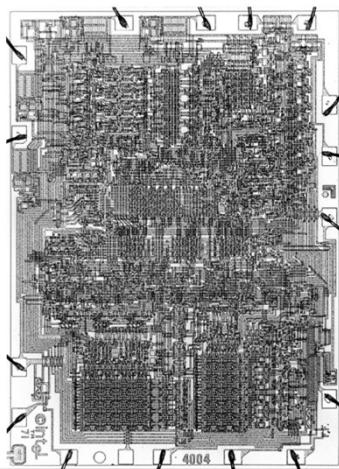
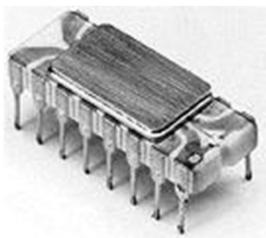


40

20

Thẻ hệ 3: Mạch tích hợp (1965-1980)

- Năm 1971, Ted Hoff chế tạo Intel 4004 theo đơn đặt hàng của một công ty Nhật bản để tạo chip sản xuất calculator. Đây là vi xử lý đầu tiên với 2400 transistor (microprocessor, processor-on-a-chip).
- 4 bit dữ liệu, 12 bit địa chỉ



41

Thẻ hệ 3: Mạch tích hợp (1965-1980)

- 1973-1974, Edward Roberts, William Yates and Jim Bybee chế tạo MITS Altair 8800, máy tính cá nhân đầu tiên
- Giá \$375, 256 bytes of memory, không keyboard, không màn hình và không bộ nhớ ngoài
- Sau đó, Bill Gate và Paul Allen viết chương trình dịch BASIC cho Altair



42

21

Chương 1

Giới thiệu chung về hệ vi xử lý

1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính

- 1.1.1 Thế hệ -1: Thời xa xưa (...-1642)
- 1.1.2 Thế hệ 0: Máy tính cơ khí (1642-1945)
- 1.1.3 Thế hệ 1: Đèn điện tử-Vacuum tubes (1945-1955)
- 1.1.4 Thế hệ 2: Transistor rời rạc-Discrete transistors (1955-1965)
- 1.1.5 Thế hệ 3: Mạch tích hợp-Integrated circuits (1965-1980)
- 1.1.6 Thế hệ 4: Mạch tích hợp cõi lớn-VLSI (1980-?)

1.2 Phân loại vi xử lý

1.3 Các hệ đếm dùng trong máy tính (nhắc lại)

1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

43

Thế hệ 4: VLSI (1980-?)

- Năm 1981, IBM bắt đầu với IBM "PC" sử dụng hệ điều hành DOS.



44

22

Thế hệ 4: VLSI (1980-?)

- Năm 1984, Xerox PARC (Palo Alto Research Center) đưa ra máy tính để bàn Alto với giao diện người và máy hoàn toàn mới: windows, biểu tượng, mouse



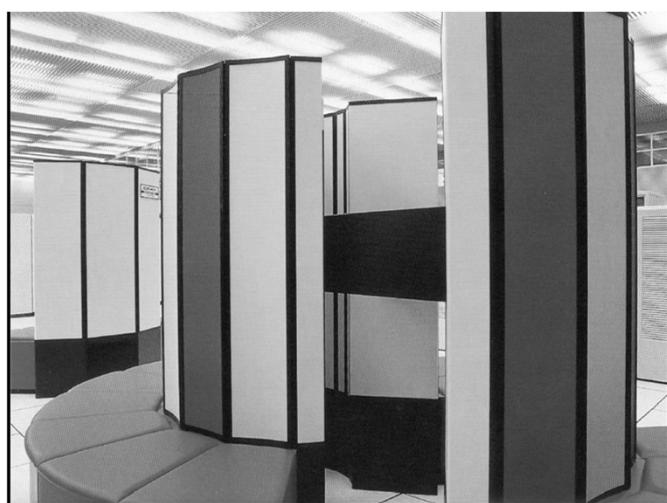
Con chuột đầu tiên



45

Thế hệ 4: VLSI (1980-?)

- Năm 1986, siêu máy tính Cray-XMP với 4 bộ xử lý đã đạt tốc độ tính toán là 840 MFlops. Nó được làm mát bằng nước

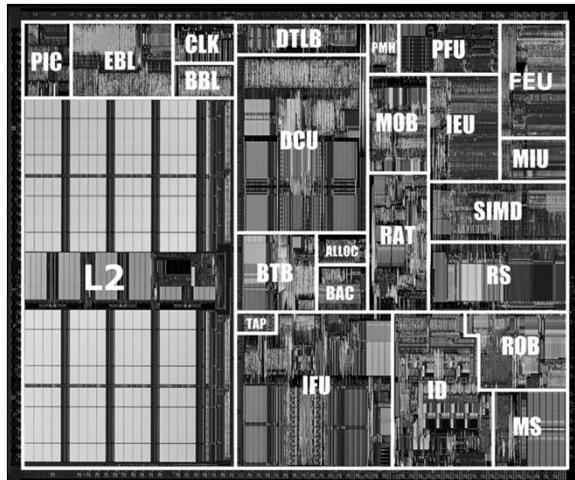


46

23

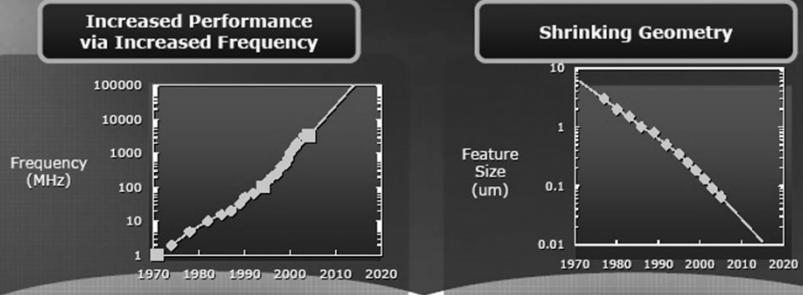
Thẻ hệ 4: VLSI (1980-?)

- Tốc độ tính toán này đã đạt được với máy tính cá nhân 1 vi xử lý, Pentium III, vào quý 1 năm 2000



47

Thẻ hệ 4: VLSI (1980-?)



1971 4004 Processor
2300 Transistors 1978 8008 Processor i386 Processor
IBM PC 32-bit 1985 Pentium Processor
3.1M transistors 1993 Montecito
1.7B Transistors

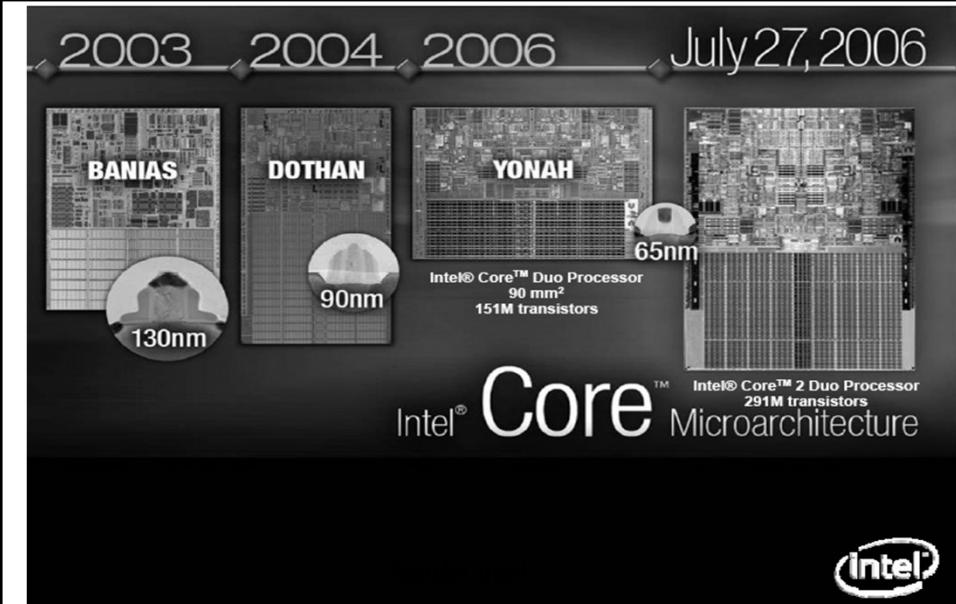
Nguồn Intel



48

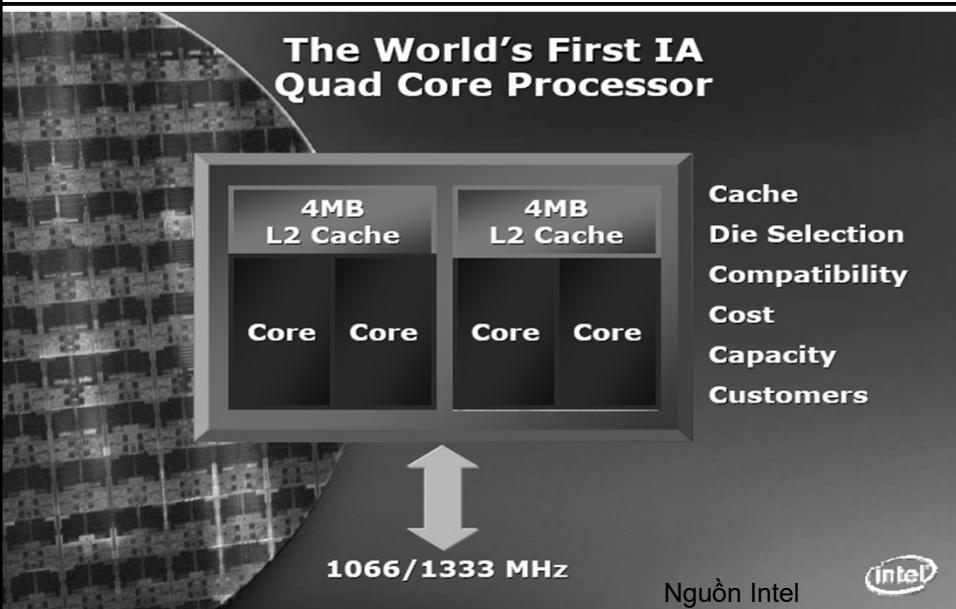
24

Thé hệ 4: VLSI (1980-?)



49

Thé hệ 4: VLSI (1980-?)

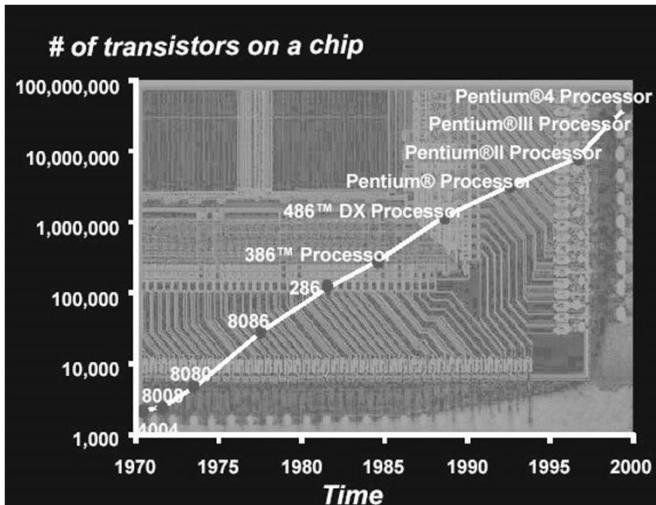
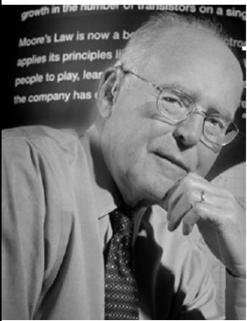


50

25



Thẻ hệ 4: VLSI (1980-?)



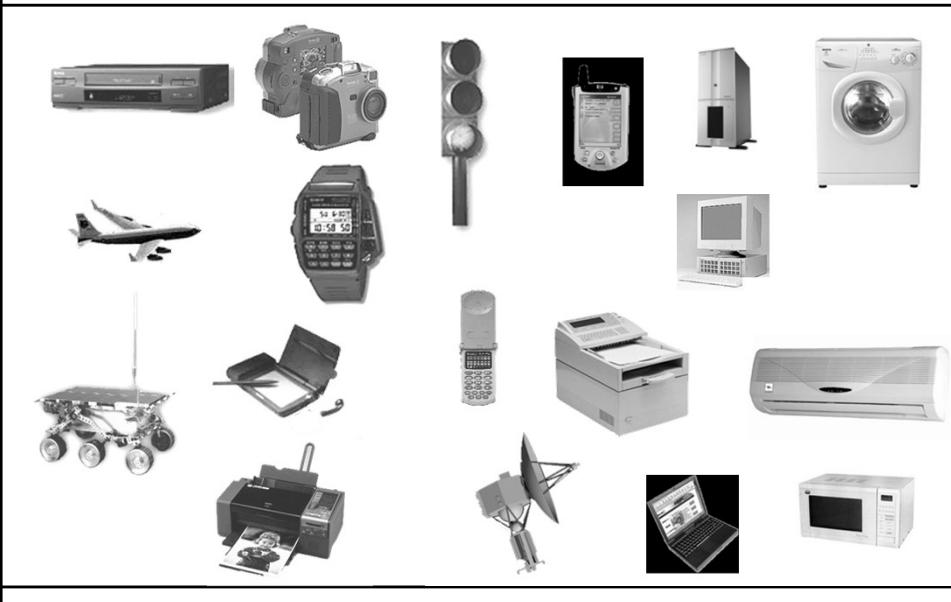
Moore's law: number of transistors doubles every 18 months
(Gordon Moore, founder Intel Corp.)

Chương 1

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính
- 1.2 Phân loại vi xử lý
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

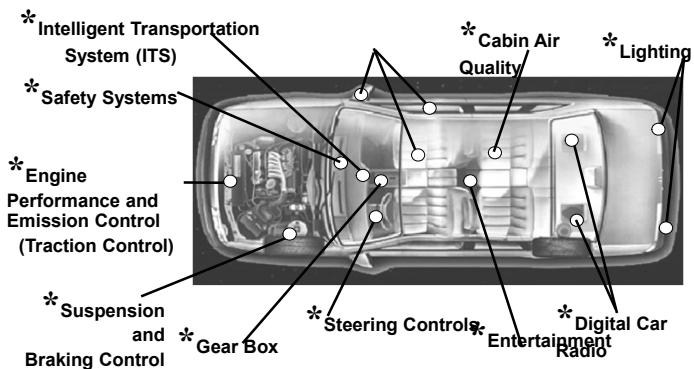
1.2 Phân loại vi xử lý



53

1.2 Phân loại vi xử lý

- **BMW > 100 processors**
- **Trung bình 1 công dân Mỹ ~ 75 processors**



54

27

1.2 Phân loại vi xử lý

Cách 1: Phân loại theo giá thành:

Type	Giá (USD)	Example application
Disposable system	1	Greeting cards
Embedded system	10	Watches, cars, appliances
Game computer	100	Home video games
Personal computer	1K	Desktop computer
Server	10K	Network server
Collection of workstations	100K	Departmental supercomputer
Mainframe	1M	Batch processing in bank
Supercomputer	10M	Weather forecasting

55

1.2 Phân loại vi xử lý

Cách 2: Phân loại theo chức năng

- Vi xử lý đa năng (General Purpose Microprocessor)
- DSP (Digital Signal Processor)
- Vi điều khiển (Microcontroller)
- Vi xử lý chuyên dụng ASIP (Application Specific Integrated Processor)

56

1.2 Phân loại vi xử lý

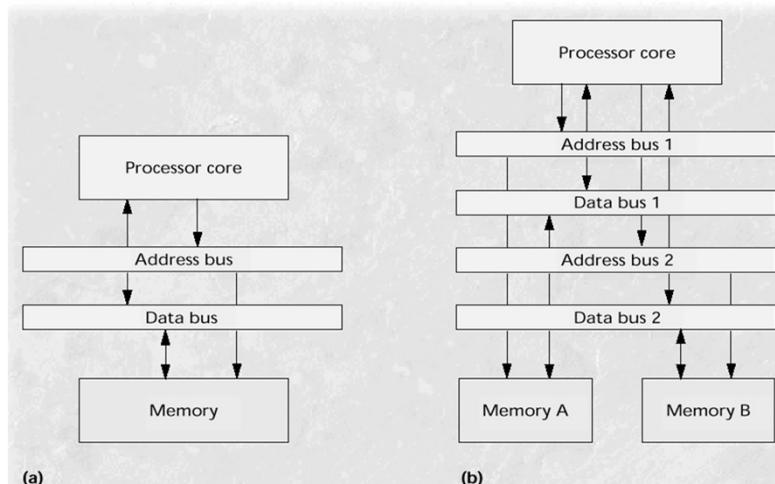
• Cách 3: Phân loại theo tập lệnh

- CISC (complex Instruction Set computer): máy tính có tập lệnh phức tạp
 - ⇒ nhiều lệnh
 - ⇒ cấu trúc phức tạp
 - ⇒ mỗi lệnh: có độ dài khác nhau và thực hiện trong 1 đến chục chu kỳ xung nhịp
 - ⇒ Ví dụ: Intel x86, AMD
- RISC (reduced instruction Set computer): máy tính có tập lệnh rút gọn
 - ⇒ ít lệnh
 - ⇒ mỗi lệnh có độ dài cố định và thực hiện trong 1 đến 2 chu kỳ xung nhịp
 - ⇒ cấu trúc vi xử lý đơn giản, có nhiều thanh ghi
 - ⇒ tốc độ xung nhịp lớn và tiêu thụ năng lượng thấp
 - ⇒ Ví dụ: ARM, PowerPC

57

1.2 Phân loại vi xử lý

Cách 4: Theo cấu trúc hệ thống Von Neumann vs. Harvard



58

29

Chương 1

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính**
- 1.2 Phân loại vi xử lý**
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)**
 - 1.3.1 Thập phân, Nhị phân, Hệ 8, Hệ 16
 - 1.3.2 Cộng, trừ, nhân, chia
 - 1.3.3 Các số âm
 - 1.3.4 Số nguyên, số thực, BCD, ASCII
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

Chương 1

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính**
- 1.2 Phân loại vi xử lý**
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)**
 - 1.3.1 Thập phân, Nhị phân, Hệ 8, Hệ 16
 - 1.3.2 Cộng, trừ, nhân, chia
 - 1.3.3 Các số âm
 - 1.3.4 Số nguyên, số thực, BCD, ASCII
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

1.3.1 Các hệ đếm Hệ thập phân

- $1234,567_{10} =$

$1 \cdot 1000 + 2 \cdot 100 + 3 \cdot 10 + 4 \cdot 1 + 5 \cdot 0.1 + 6 \cdot 0.01 + 7 \cdot 0.001$

$1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0 + 5 \cdot 10^{-1} + 6 \cdot 10^{-2} + 7 \cdot 10^{-3}$

$r = \text{cơ số}$ ($r = 10$), $d = \text{digit}$ ($0 \leq d \leq 9$), $m = \text{số chữ số trước dấu phẩy}$, $n = \text{số chữ số sau dấu phẩy}$

$$D = \sum_{i=-n}^{m-1} d_i \bullet r^i$$

1.3.1 Các hệ đếm Hệ nhị phân

- $1011,011_2 =$

$1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 + 0 \cdot 0.5 + 1 \cdot 0.25 + 1 \cdot 0.125$

$1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}$

$r = \text{cơ số}$ ($r = 2$), $d = \text{digit}$ ($0 \leq d \leq 1$), $m = \text{số chữ số trước dấu phẩy}$, $n = \text{số chữ số sau dấu phẩy}$

$$B = \sum_{i=-n}^{m-1} d_i \bullet 2^i$$

1.3.1 Các hệ đếm Hệ 8 (Octal)

- $7654,32_8 =$

$7 \cdot 512 + 6 \cdot 64 + 5 \cdot 8 + 4 \cdot 1 + 3 \cdot 0.125 + 2 \cdot 0.015625$

$7 \cdot 8^3 + 6 \cdot 8^2 + 5 \cdot 8^1 + 4 \cdot 8^0 + 3 \cdot 8^{-1} + 2 \cdot 8^{-2}$

$r = \text{cơ số } (r = 8)$, $d = \text{digit } (0 \leq d \leq 7)$, $m = \text{số chữ số trước dấu phẩy}$, $n = \text{số chữ số sau dấu phẩy}$

$$O = \sum_{i=-n}^{m-1} d_i \bullet 8^i$$

1.3.1 Các hệ đếm Hệ 16 (Hexadecimal)

- $FEDC,76_{16} =$

$15 \cdot 4096 + 14 \cdot 256 + 13 \cdot 16 + 12 \cdot 1 + 7 \cdot 1/16 + 6 \cdot 1/256$

$15 \cdot 16^3 + 14 \cdot 16^2 + 13 \cdot 16^1 + 12 \cdot 16^0 + 7 \cdot 16^{-1} + 6 \cdot 16^{-2}$

$r = \text{cơ số } (r = 16)$, $d = \text{digit } (0 \leq d \leq F)$, $m = \text{số chữ số trước dấu phẩy}$, $n = \text{số chữ số sau dấu phẩy}$

$$H = \sum_{i=-n}^{m-1} d_i \bullet 16^i$$

1.3.1 Các hệ đếm

65/Chapter1

Chuyển đổi giữa các hệ đếm

- **Chuyển từ hệ thập phân sang nhị phân**

- Quy tắc: lấy số cần đổi chia cho 2 và ghi nhớ phần dư, lấy thương chia tiếp cho 2 và ghi nhớ phần dư. Lặp lại khi thương bằng 0. Đảo ngược thứ tự dãy các số dư sẽ được chữ số của hệ nhị phân cần tìm

- Ví dụ: Đổi 34 sang hệ nhị phân: 100010

- **Chuyển từ hệ nhị phân sang hệ 16 và ngược lại**

- $1011\ 0111B = B7H$

Chương 1

66/Chapter1

Giới thiệu chung về hệ vi xử lý

1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính

1.2 Phân loại vi xử lý

1.3 Các hệ đếm dùng trong máy tính (nhắc lại)

- 1.3.1 Thập phân, Nhị phân, Hệ 8, Hệ 16

- 1.3.2 Cộng, trừ, nhân, chia

- 1.3.3 Các số âm

- 1.3.4 Số nguyên, số thực, BCD, ASCII

1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

1.3.2 Các phép toán Cộng nhị phân

- Cộng thập phân

Nhớ 0 1 0

x 8 2 7 3

y 5 6 2

Tổng 8 8 3 5

- Cộng nhị phân

Nhớ 0 0 1 1 1 1 1

x 1 0 0 1 1 0 1 1

y 1 0 1 0 1 1 1

Tổng 1 1 1 1 0 0 1 0

1.3.2 Các phép toán Trừ nhị phân

x 1 1 1 0 1

y 1 1 1 1

Mượn 1 1 1 0

Hiệu 0 1 1 1 0

1.3.2 Các phép toán Nhân nhị phân

- Nguyên tắc: cộng và dịch

$$\begin{array}{r}
 1110 \\
 1101 \\
 \hline
 1110 \\
 0000 \\
 1110 \\
 \hline
 10110110
 \end{array}$$

69

1.3.2 Các phép toán Chia nhị phân

$$\begin{array}{r}
 10111010 \\
 1110 \\
 \hline
 1001010 \\
 1110 \\
 \hline
 10010 \\
 0000 \\
 10010 \\
 1110 \\
 \hline
 100
 \end{array}$$

- Nguyên tắc: trừ và dịch

70

Chương 1

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính**
- 1.2 Phân loại vi xử lý**
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)**
 - 1.3.1 Thập phân, Nhị phân, Hệ 8, Hệ 16
 - 1.3.2 Cộng, trừ, nhân, chia
 - 1.3.3 Các số âm
 - 1.3.4 Số nguyên, số thực, BCD, ASCII
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

Biểu diễn bằng dấu và độ lớn (Sign-Magnitude)

- Một số có dấu bao gồm 2 phần: dấu và độ lớn
- Ví dụ hệ 10: $+123_{10}$ (thông thường ‘123’) và -123_{10}
- Hệ nhị phân: bít dấu là bít MSB; ‘0’ = dương, ‘1’ = âm
- Ví dụ: $01100_2 = +12_{10}$ và $11100_2 = -12_{10}$
- Các số có dấu 8 bít sẽ có giá trị từ -127 đến +127 với 2 số 0: $1000\ 0000 (-0)$ và $0000\ 0000 (+0)$



Số bù 2

- **Số bù 1 (bù lô gic): đảo bit**
 - $1001 \Rightarrow 0110$
 - $0100 \Rightarrow 1011$
- **Số bù 2 (bù số học): số bù 1 +1**
- **Ví dụ: Tìm số bù 2 của 13**

$$13 = 0000\ 1101$$

Số bù 1 của 13 =1111 0010

Cộng thêm 1: 1

Số bù 2 của 13= 1111 0011 (tức là -13)

73



Số bù 2

- **Ví dụ: Tìm số bù 2 của 0**

$$0 = 0000\ 0000$$

Số bù 1 của 0 =1111 1111

Cộng thêm 1: 1

Số bù 2 của 0= 0000 0000 (tức là -0)

- Như vậy với số bù 2, số 0 được biểu diễn 1 cách duy nhất
- Số có dấu 8 bit sẽ có giá trị từ -128 đến 127

74

37

Số bù 2

Decimal	S_ bù 2	Sign-magnitude
-8	1000	-
-7	1001	1111
-6	1010	1110
-5	1011	1101
-4	1100	1100
-3	1101	1011
-2	1110	1010
-1	1111	1001
0	0000	1000 & 0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111

Chương 1

Giới thiệu chung về hệ vi xử lý

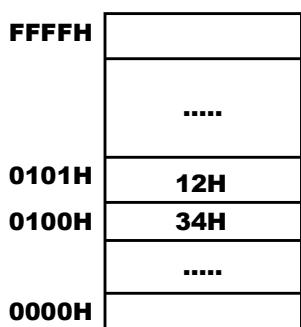
- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính
- 1.2 Phân loại vi xử lý
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)
 - 1.3.1 Thập phân, Nhị phân, Hệ 8, Hệ 16
 - 1.3.2 Cộng, trừ, nhân, chia
 - 1.3.3 Các số âm
 - 1.3.4 Số nguyên, số thực, BCD, ASCII
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý

Số nguyên (integer)

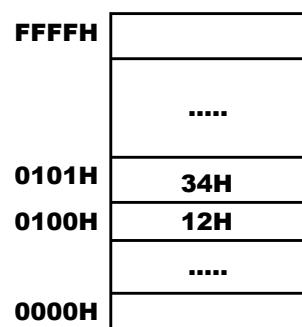
- 8 bit
 - unsigned: 0 đến 255
 - signed : -128 đến 127 (bù hai)
- 16 bit
 - unsigned: 0 đến 65535 ($2^{16}-1$)
 - signed : -32768 (2^{15}) đến 32767 ($2^{15}-1$)
- 32 bit
 - unsigned: 0 đến $2^{32}-1$
 - signed : -2^{31} đến $2^{31}-1$

Little endian và big endian

- Số 1234 H được lưu trữ thế nào trong bộ nhớ 8 bit?



little endian
Intel microprocessors



big endian
Motorola microprocessors

Số thực (real number, floating point number)

- Ví dụ: $1,234 = 1,234 \cdot 10^0 = 0,1234 \cdot 10^1 = \dots$
- $11,01_B = \underbrace{1,101}_{\text{mantissa}} \cdot \underbrace{2^1}_{\text{exponent}} = \underbrace{0,1101}_{\text{mantissa}} \cdot \underbrace{2^2}_{\text{exponent}} = \dots$
- Real number: (m, e), e.g. (0.1101, 2)
 - Single precision: 32 bit
 - Double precision: 64 bit

79

Số thực (real number, floating point number)

- IEEE-754 format cho single-precision

31	30	23	22	0
S	biased exponent e	fraction f of normalized mantissa		

1 sign bit: 0 dương, 1 âm

8 bit biased exponent = exponent + 127

24 bit mantissa chuẩn hóa = 1 bit ẩn + 23 bit fraction

Mantissa chuẩn hóa: có giá trị giữa 1 và 2 : 1.f

Ví dụ: biểu diễn 0.1011 dưới dạng IEEE-754

Sign bit s=0

chuẩn hóa mantissa: $0.1011 = 1.011 \cdot 2^{-1}$

Biased exponent: $-1 + 127 = 126 = 01111110$

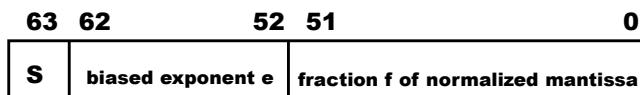
IEEE format: 0 01111110 01100000000000000000000000000000

80

40

Số thực (real number, floating point number)

- IEEE-754 format cho double-precision



1 sign bit: 0 dương, 1 âm

11 bit biased exponent = exponent + 1023

53 bit mantissa chuẩn hóa = 1 bit ẩn + 52 bit fraction

single precision: $(-1)^s \times 2^{e-127} \times (1.f)_2$

double precision: $(-1)^s \times 2^{e-1023} \times (1.f)_2$

Số thực (real number, floating point number)

	Single Precision	Double Precision
Machine epsilon Độ chính xác	2^{-23} or 1.192×10^{-7}	2^{-52} or 2.220×10^{-16}
Smallest positive Số dương nhỏ nhất	2^{-126} or 1.175×10^{-38}	2^{-1022} or 2.225×10^{-308}
Largest positive Số dương lớn nhất	$(2 - 2^{-23}) 2^{127}$ or 3.403×10^{38}	$(2 - 2^{-52}) 2^{1023}$ or 1.798×10^{308}
Decimal Precision Độ chính xác thập phân	6 significant digits 6 chữ số sau dấu phẩy	15 significant digits 15 chữ số sau dấu phẩy

BCD

- **Binary Coded Decimal number**

- **BCD chuẩn (BCD nén, packed BCD):**

- ⇒ 1 byte biểu diễn 2 số BCD

- ⇒ Ví dụ: 25: 0010 0101

- **BCD không nén (unpacked BCD) :**

- ⇒ 1 byte biểu diễn 1 số BCD

- ⇒ ví dụ: 25: 00000010 00000101

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

ASCII

- **American Standard Code for Information Interchange (7-bit code)**

b3b2b1b0	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	'	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	o	DEL	

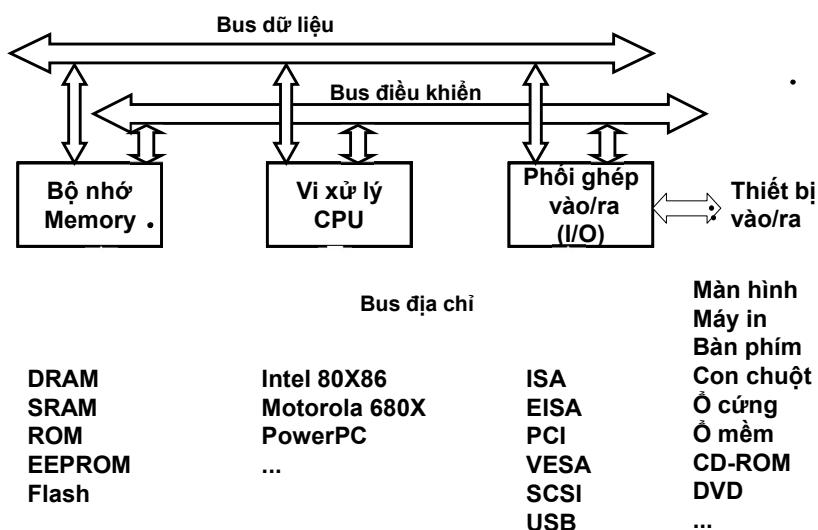
Chương 1

Giới thiệu chung về hệ vi xử lý

- 1.1 Lịch sử phát triển của các bộ vi xử lý và máy tính**
- 1.2 Phân loại vi xử lý**
- 1.3 Các hệ đếm dùng trong máy tính (nhắc lại)**
- 1.4 Sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

85

1.4.1 Hệ vi xử lý



86

43

1.4.1 Hệ vi xử lý

- CPU (Central Processing Unit)**

- Đơn vị số học và logic (Arithmetic Logical Unit)**

- Thực hiện các phép toán số học

- ✓ Cộng, trừ, nhân chia

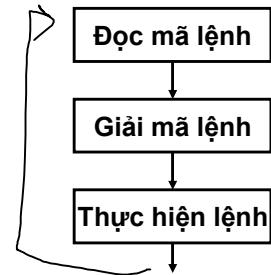
- Thực hiện các phép toán logic

- ✓ And, or, compare..

- Đơn vị điều khiển (Control Unit)**

- Các thanh ghi (Registers)**

- Lưu trữ dữ liệu và trạng thái của quá trình thực hiện lệnh



Interval bus

1.4.1 Hệ vi xử lý

- Đọc thông tin từ bộ nhớ vào CPU**

- Xác định xem lệnh đó là lệnh gì**

- Thực hiện lệnh**

- Nếu cần thì đọc thêm thông tin từ bộ nhớ/cổng**

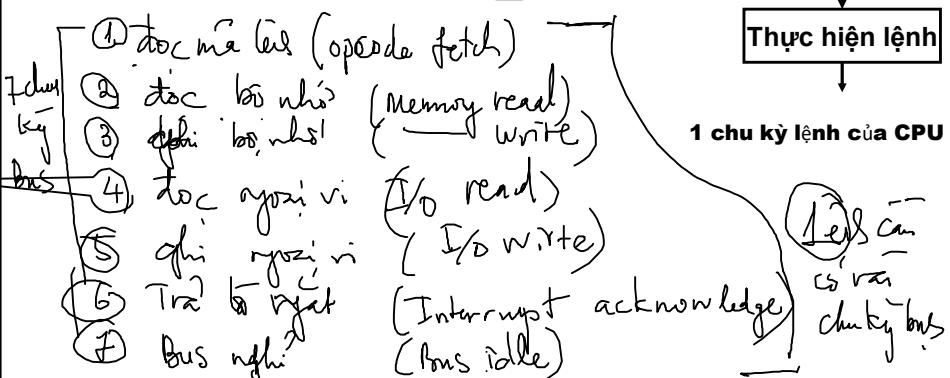
- Tính toán và ghi thông tin ra bộ nhớ/cổng**

Đọc mã lệnh

Giải mã lệnh

Thực hiện lệnh

1 chu kỳ lệnh của CPU



1.4.2 Bộ nhớ, BUS

- **Memory**

- ROM: không bị mất dữ liệu, chứa dữ liệu điều khiển hệ thống lúc khởi động
- RAM: mất dữ liệu khi mất nguồn, chứa chương trình và dữ liệu trong quá trình hoạt động của hệ thống

- **Bus dữ liệu**

- 8, 16, 32, 64 bit tùy thuộc vào vi xử lý

- **Bus địa chỉ:**

- 16, 20, 24, 32, 36 bit
- số ô nhớ có thể đánh địa chỉ: 2^N
- Ví dụ: 8088/8086 có 20 đường địa chỉ \Rightarrow quản lý được 2^{20} bytes=1Mbytes

1.4.2 Bộ nhớ, BUS

Nhà sản xuất	Tên vi xử lý	Bus dữ liệu	Bus địa chỉ	Khả năng địa chỉ
Intel	8088	8	20	1 M
	8086	16	20	1 M
	80186	16	20	1 M
	80286	16	24	16 M
	80386SX	16	24	16 M
	80386DX	32	32	4 G
	80486DX	32	32	4 G
	Pentium	64	32	4 G
	Pentium Pro	64	36	64 G
Motorola	Pentium I, II, III, IV	64	36	64 G
	68000	16	24	16 M
	68010	16	24	16 M
	68020	32	32	4 G
	68030	32	32	4 G
	68040	32	32	4 G
	68060	64	32	4 G
	PowerPC	64	32	4 G



Nội dung môn học

1. Giới thiệu chung về hệ vi xử lý
2. Bộ vi xử lý Intel 8088/8086
3. Lập trình hợp ngữ cho 8086
4. Tổ chức vào ra dữ liệu
5. Ngắt và xử lý ngắt
6. Truy cập bộ nhớ trực tiếp DMA
7. Các bộ vi xử lý trên thực tế



Chương 2: Bộ vi xử lý Intel 8088/8086

- 2.1 Cấu trúc bên trong
- 2.2 Sơ đồ chân
- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hoá lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

- 2.1.1 Sơ đồ khái
- 2.1.2 Các thanh ghi đa năng
- 2.1.3 Các thanh ghi đoạn
- 2.1.4 Các thanh ghi con trỏ và chỉ số
- 2.1.5 Thanh ghi cờ
- 2.1.6 Hàng đợi lệnh

2.2 Sơ đồ chân

- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hoá lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



Chương 2: Bộ vi xử lý Intel 8088/8086

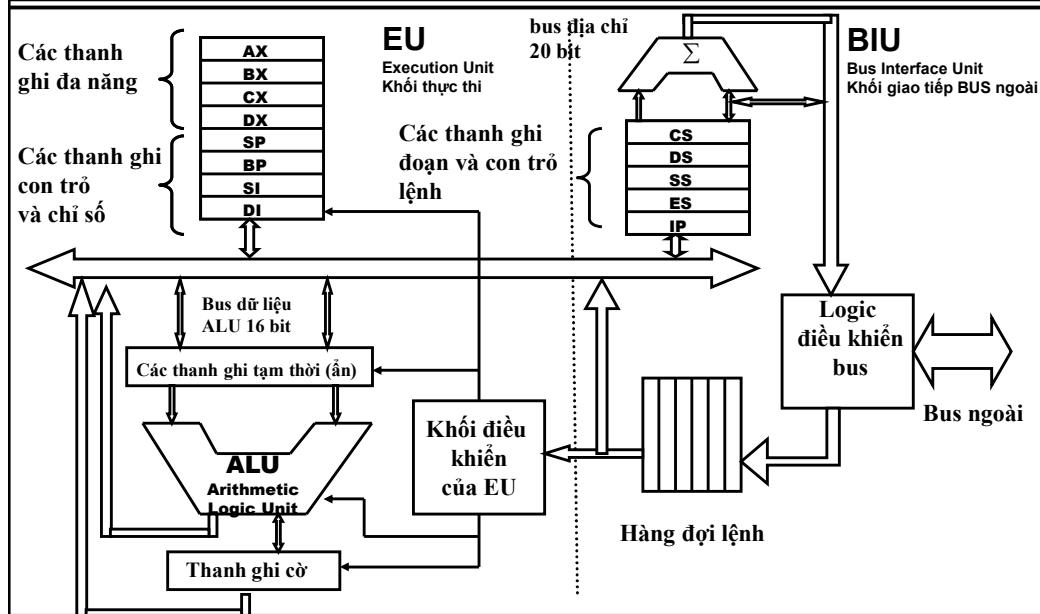
2.1 Cấu trúc bên trong

- 2.1.1 Sơ đồ khái
- 2.1.2 Các thanh ghi đa năng
- 2.1.3 Các thanh ghi đoạn
- 2.1.4 Các thanh ghi con trỏ và chỉ số
- 2.1.5 Thanh ghi cờ
- 2.1.6 Hàng đợi lệnh

2.2 Sơ đồ chân

- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hoá lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286

2.1.1 Sơ đồ khói 8088/8086



5

Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

- 2.1.1 Sơ đồ khói
- 2.1.2 Các thanh ghi đa năng
- 2.1.3 Các thanh ghi đoạn
- 2.1.4 Các thanh ghi con trỏ và chỉ số
- 2.1.5 Thanh ghi cờ
- 2.1.6 Hàng đợi lệnh

2.2 Sơ đồ chân

- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hóa lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286

6

3



2.1.2 Các thanh ghi đa năng của 8088/8086

	8 bit cao	8 bit thấp
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

- **8088/8086 đến 80286 : 16 bits**
- **80386 trở lên: 32 bits EAX, EBX, ECX, EDX**

- **Thanh ghi chứa AX (accumulator):** chứa kết quả của các phép tính. Kết quả 8 bit được chứa trong AL
- **Thanh ghi cơ sở BX (base):** chứa địa chỉ cơ sở, ví dụ của bảng dùng trong lệnh XLAT (Translate)
- **Thanh ghi đếm CX (count):** dùng để chứa số lần lặp trong các lệnh lặp (Loop). CL được dùng để chứa số lần dịch hoặc quay trong các lệnh dịch và quay thanh ghi
- **Thanh ghi dữ liệu DX (data):** cùng AX chứa dữ liệu trong các phép tính nhân chia số 16 bit. DX còn được dùng để chứa địa chỉ công trong các lệnh vào ra dữ liệu trực tiếp (IN/OUT)

7



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

- 2.1.1 Sơ đồ khái
- 2.1.2 Các thanh ghi đa năng
- 2.1.3 Các thanh ghi đoạn
- 2.1.4 Các thanh ghi con trỏ và chỉ số
- 2.1.5 Thanh ghi cờ
- 2.1.6 Hàng đợi lệnh

2.2 Sơ đồ chân

- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hóa lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286

8



2.1.3 Các thanh ghi đoạn

- Tổ chức của bộ nhớ 1 Mbytes**

Ván đề: Sử dụng 2 thanh ghi 16bit để xác định địa chỉ 20bit (1M)

Đoạn bộ nhớ (segment)

⇒ 2^{16} bytes = 64 KB

⇒ Đoạn 1: địa chỉ 0000

⇒ Đoạn 2: địa chỉ 0001

⇒ Đoạn cuối cùng: FFFF

Ô nhớ trong đoạn:

⇒ địa chỉ lệch: offset

⇒ Ô 1: offset: 0000

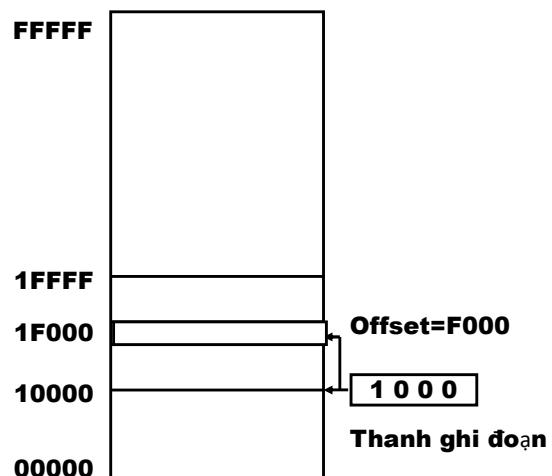
⇒ Ô cuối cùng: offset: FFFF

Địa chỉ vật lý:

⇒ Segment : offset

Địa chỉ vật lý=Segment*16 + offset

Chế độ thực (real mode)



9



2.1.3 Các thanh ghi đoạn

- Ví dụ: Địa chỉ vật lý 12345H**

Địa chỉ đoạn	Địa chỉ lệch
1000 H	2345H
1200 H	0345H
1004 H	?
0300 H	?

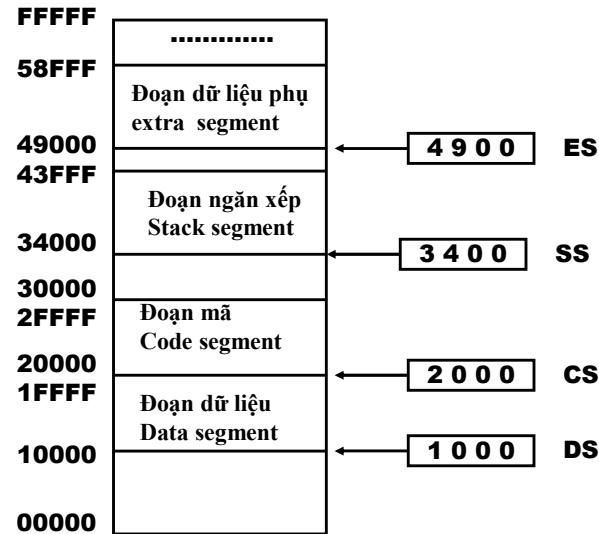
- Ví dụ: Cho địa chỉ đầu của đoạn: 49000 H, xác định địa chỉ cuối**

10



2.1.3 Các thanh ghi đoạn

- Các thanh ghi đoạn: chứa địa chỉ đoạn

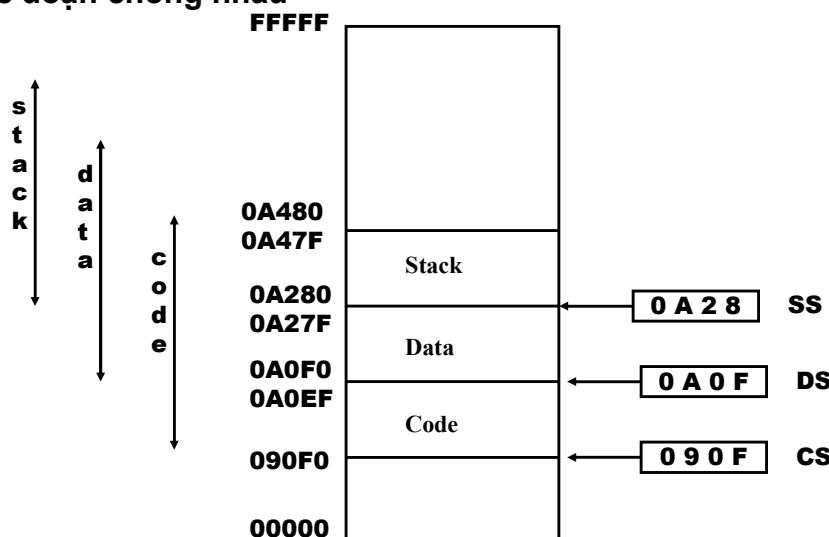


11



2.1.3 Các thanh ghi đoạn

- Các đoạn chồng nhau



12



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

- 2.1.1 Sơ đồ khái
- 2.1.2 Các thanh ghi đa năng
- 2.1.3 Các thanh ghi đoạn
- 2.1.4 Các thanh ghi con trỏ và chỉ số
- 2.1.5 Thanh ghi cờ
- 2.1.6 Hàng đợi lệnh

2.2 Sơ đồ chân

- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hóa lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



2.1.4 Các thanh ghi con trỏ và chỉ số

• Chứa địa chỉ lệnh (offset)

- Con trỏ lệnh IP (instruction pointer): chứa địa chỉ lệnh tiếp theo trong đoạn mã lệnh CS.
⇒ CS:IP
- Con trỏ cơ sở BP (Base Pointer): chứa địa chỉ của dữ liệu trong đoạn ngắn xếp SS hoặc các đoạn khác
⇒ SS:BP
- Con trỏ ngắn xếp SP (Stack Pointer): chứa địa chỉ hiện thời của đỉnh ngắn xếp
⇒ SS:SP
- Chỉ số nguồn SI (Source Index): chứa địa chỉ dữ liệu nguồn trong đoạn dữ liệu DS trong các lệnh chuỗi
⇒ DS:SI
- Chỉ số đích (Destination Index): chứa địa chỉ dữ liệu đích trong đoạn dữ liệu DS trong các lệnh chuỗi
⇒ DS:DI
- SI và DI có thể được sử dụng như thanh ghi đa năng
- 80386 trở lên 32 bit: EIP, EBP, ESP, EDI, ESI



2.1.4 Các thanh ghi con trỏ và chỉ số

- Thanh ghi đoạn và thanh ghi lệch ngầm định

Segment	Offset	Chú thích
CS	IP	Địa chỉ lệnh
SS	SP hoặc BP	Địa chỉ ngăn xếp
DS	BX, DI, SI, số 8 bit hoặc số 16 bit	Địa chỉ dữ liệu
ES	DI	Địa chỉ chuỗi đích



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

- 2.1.1 Sơ đồ khôi
- 2.1.2 Các thanh ghi đa năng
- 2.1.3 Các thanh ghi đoạn
- 2.1.4 Các thanh ghi con trỏ và chỉ số
- 2.1.5 Thanh ghi cờ
- 2.1.6 Hàng đợi lệnh

2.2 Sơ đồ chân

- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hóa lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



2.1.5 Thanh ghi cờ (Flag Register)

15 14

2 1 0



- **9 bit được sử dụng, 6 cờ trạng thái:**

- C hoặc CF (carry flag): CF=1 khi có nhớ hoặc mượn từ MSB
- P hoặc PF (parity flag): PF=1 (0) khi tổng số bit 1 trong kết quả là chẵn (lẻ)
- A hoặc AF (auxiliary carry flag): cờ nhớ phụ, AF=1 khi có nhớ hoặc mượn từ một số BCD thấp sang BCD cao
- Z hoặc ZF (zero flag): ZF=1 khi kết quả bằng 0
- S hoặc SF (Sign flag): SF=1 khi kết quả âm
- O hoặc OF (Overflow flag): cờ tràn OF=1 khi kết quả là một số vượt ra ngoài giới hạn biểu diễn của nó trong khi thực hiện phép toán cộng trừ số có dấu



2.1.5 Thanh ghi cờ (Flag Register)

15 14

2 1 0



- **3 cờ điều khiển**

- T hoặc TF (trap flag): cờ bẫy, TF=1 → CPU làm việc ở chế độ chạy từng lệnh
- I hoặc IF (Interrupt enable flag): cờ cho phép ngắt, IF=1 thì CPU sẽ cho phép các yêu cầu ngắt (ngắt che được) được tác động (Các lệnh: STI, CLI)
- D hoặc DF (direction flag): cờ hướng, DF=1 khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải sang trái (lệnh STD, CLD)



2.1.5 Thanh ghi cờ (Flag Register)

- Ví dụ:

$$\begin{array}{r}
 80h \\
 + \\
 80h \\
 \hline
 100h
 \end{array}$$

- SF=0 vì msb trong kết quả =0
- PF=1 vì có 0 bit của tổng bằng 1
- ZF=1 vì kết quả thu được là 0
- CF=1 vì có nhón từ bit msb trong phép cộng
- OF=1 vì có tràn trong phép cộng 2 số âm
 - ⇒ Cộng 2 số âm thu được kết quả dương



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

- 2.1.1 Sơ đồ khôi
- 2.1.2 Các thanh ghi đa năng
- 2.1.3 Các thanh ghi đoạn
- 2.1.4 Các thanh ghi con trỏ và chỉ số
- 2.1.5 Thanh ghi cờ
- 2.1.6 Hàng đợi lệnh

2.2 Sơ đồ chân

- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hóa lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



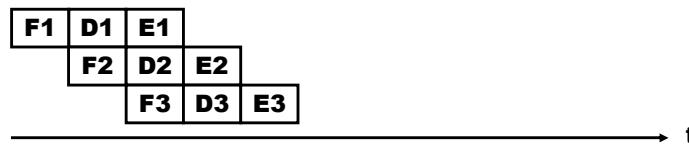
2.1.6 Hàng đợi lệnh

- 4 bytes đối với 8088 và 6 bytes đối với 8086
- Xử lý pipeline

Không có
pipelining



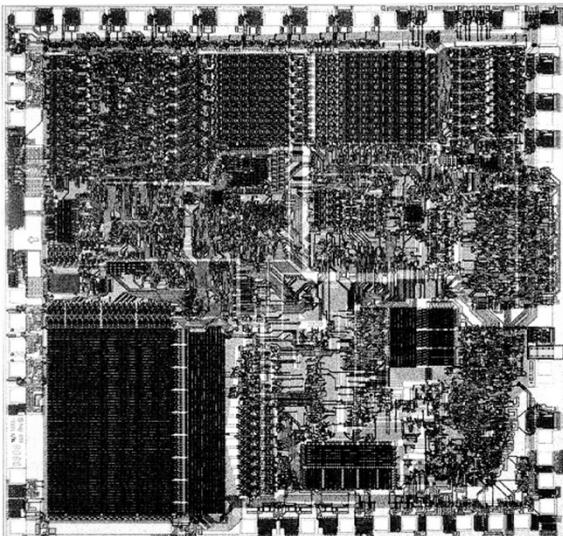
Có pipelining



Chương 2: Bộ vi xử lý Intel 8088/8086

- 2.1 Cấu trúc bên trong
- 2.2 Sơ đồ chân
- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hóa lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286

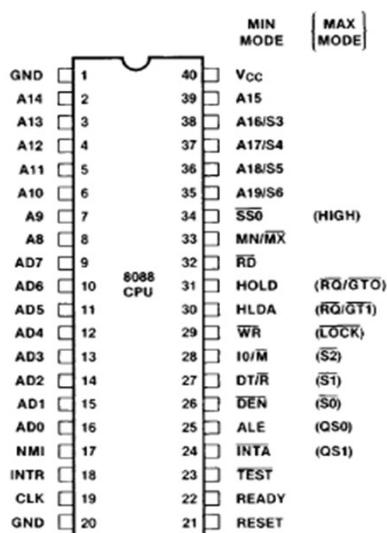
2.2 Sơ đồ chân Intel 8088



- 16-bit processor
- introduced in 1979
- 3 μm, 5 to 8 MHz, 29 KTOR, 0.33 to 0.66 MIPS

23

2.2 Sơ đồ chân Intel 8088



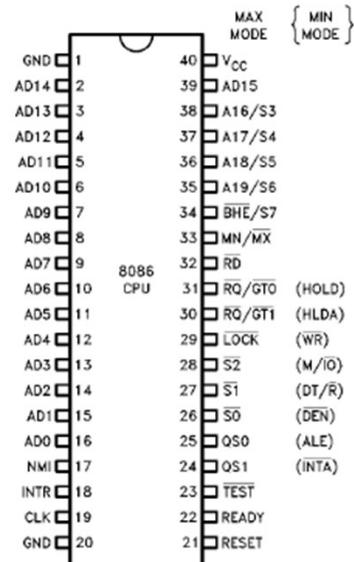
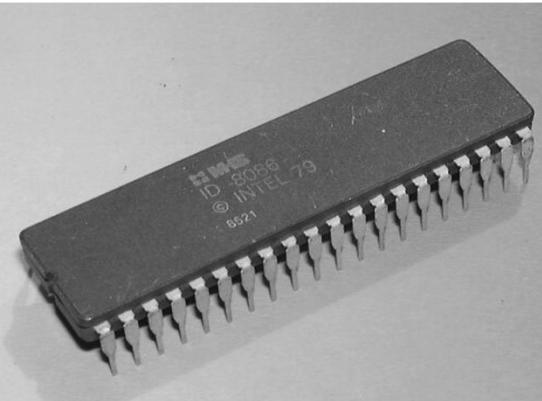
• Chế độ Min và chế độ Max:

MN/MX = 1 chế độ Min
= 0 chế độ Max với bus controller
8288

24

12

2.2 Sơ đồ chân Intel 8086



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

2.2 Sơ đồ chân

2.3 Bản đồ bộ nhớ của máy tính IBM-PC

2.4 Các chế độ địa chỉ của 8086

2.5 Cách mã hoá lệnh của 8086

2.6 Mô tả tập lệnh của 8086

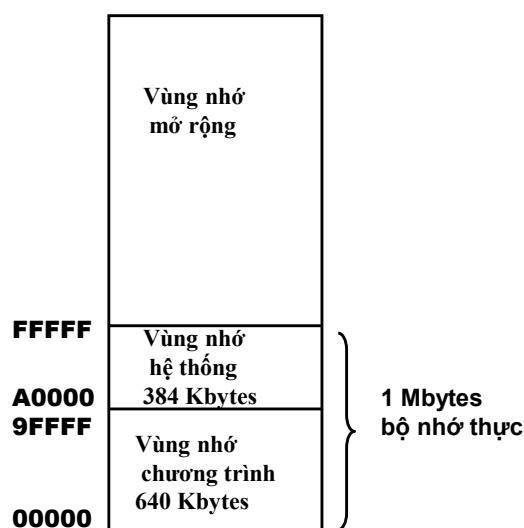
2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286

2.3.1 Trình tự khởi động

- Khi bật nguồn hoặc nhấn Reset

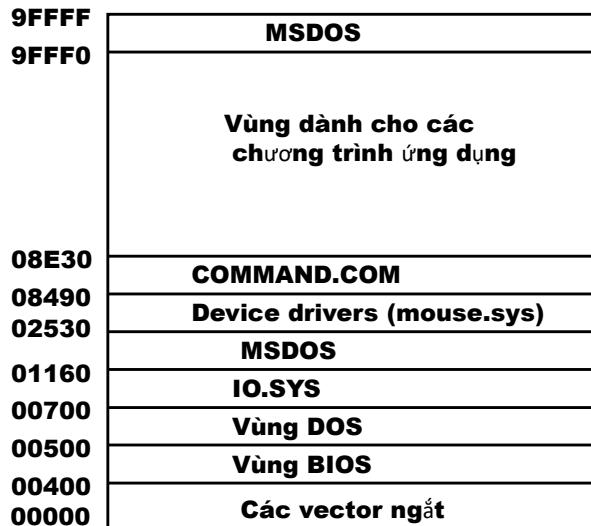
- CS=FFFFh và IP=0000 => địa chỉ FFFF0 chứa chỉ thị chuyển điều khiển đến điểm khởi đầu của các chương trình BIOS
- Các chương trình BIOS kiểm tra hệ thống và bộ nhớ
- Các chương trình BIOS khởi tạo bảng vector ngắn và vùng dữ liệu BIOS
- BIOS nạp chương trình khởi động (boot program) từ đĩa vào bộ nhớ
- Chương trình khởi động nạp hệ điều hành từ đĩa vào bộ nhớ
- Hệ điều hành nạp các chương trình ứng dụng

2.3.2 Bản đồ bộ nhớ của máy tính IBM PC



2.3.2 Bản đồ bộ nhớ máy tính IBM-PC Vùng nhớ chương trình

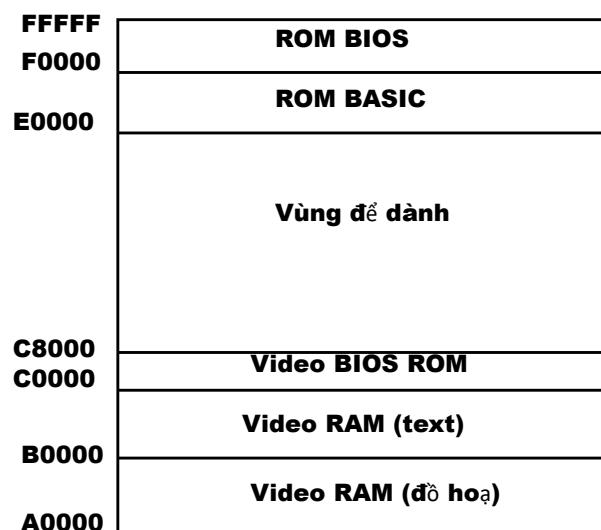
29/Chapter2



29

2.3.2 Bản đồ bộ nhớ máy tính IBM-PC Vùng nhớ hệ thống

30/Chapter2

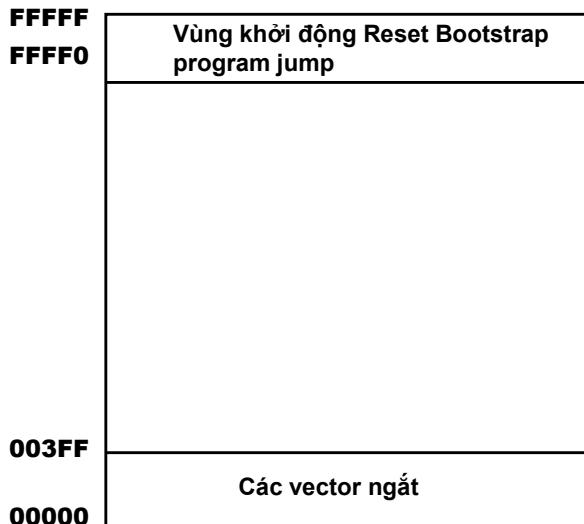


30

15

2.3.2 Bản đồ bộ nhớ máy tính IBM-PC Vùng nhớ dành riêng của 8088/8086

31/Chapter2

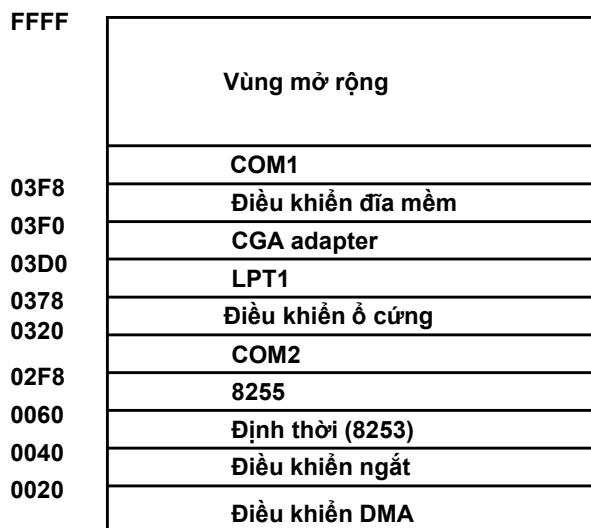


31

2.3.2 Bản đồ bộ nhớ máy tính IBM-PC Các cổng vào ra

32/Chapter2

- Địa chỉ: 0000H – FFFFH, M/I_O = 0



32

16



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

2.2 Sơ đồ chân

2.3 Bản đồ bộ nhớ của máy tính IBM-PC

2.4 Các chế độ địa chỉ của 8088/8086

2.4.1 Chế độ địa chỉ thanh ghi

2.4.2 Chế độ địa chỉ tức thì

2.4.3 Chế độ địa chỉ trực tiếp

2.4.4 Chế độ địa chỉ gián tiếp qua thanh ghi

2.4.5 Chế độ địa chỉ tương đối cơ sở

2.4.6 Chế độ địa chỉ tương đối chỉ số

2.4.7 Chế độ địa chỉ tương đối chỉ số cơ sở

2.5 Cách mã hoá lệnh của 8088/8086

2.6 Mô tả tập lệnh của 8088/8086

2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



2.4.1 Chế độ địa chỉ thanh ghi (Register Addressing Mode)

- Dùng các thanh ghi như là các toán hạng
- Tốc độ thực hiện lệnh cao
- **Ví dụ:**
 - MOV BX, DX ; Copy nội dung DX vào BX, nội dung DX được giữ nguyên
 - MOV AL, BL ; Copy nội dung BL vào AL
 - MOV AL, BX ; không hợp lệ vì các thanh ghi có kích thước khác nhau
 - MOV ES, DS ; không hợp lệ (segment to segment)
 - MOV CS, AX ; không hợp lệ vì CS không được dùng làm thanh ghi đích
 - ADD AL, DL ; Cộng nội dung AL và DL rồi đưa vào AL



2.4.2 Chế độ địa chỉ tức thì (Immediate Addressing Mode)

- Toán hạng đích là thanh ghi hoặc ô nhớ
- Toán hạng nguồn là hằng số
- Dùng để nạp hằng số vào thanh ghi (trừ thanh ghi đoạn và thanh cờ) hoặc vào ô nhớ trong đoạn dữ liệu DS
- Ví dụ:
 - MOV BL, 44 ; Copy số thập phân 44 vào thanh ghi BL
 - MOV AX, 44H ; Copy 0044H vào thanh ghi AX
 - MOV AL, 'A' ; Copy mã ASCII của A vào thanh ghi AL
 - MOV DS, 0FF0H ; không hợp lệ
 - MOV AX, 0FF0H ;
 - MOV DS, AX ;
 - MOV [BX], 10 ; copy số thập phân 10 vào ô nhớ DS:BX



2.4.3 Chế độ địa chỉ trực tiếp (Direct Addressing Mode)

- Một toán hạng là địa chỉ ô nhớ chứa dữ liệu
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
 - MOV AL, [1234H] ; Copy nội dung ô nhớ có địa chỉ DS:1234 vào AL
 - MOV [4320H], CX ; Copy nội dung của CX vào 2 ô nhớ liên tiếp DS: 4320 và DS: 4321



2.4.4 Chế độ địa chỉ gián tiếp qua thanh ghi (Register indirect Addressing Mode)

- Một toán hạng là thanh ghi chứa địa chỉ của 1 ô nhớ dữ liệu
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
 - MOV AL, [BX] ; Copy nội dung ô nhớ có địa chỉ DS:BX vào AL
 - MOV [SI], CL ; Copy nội dung của CL vào ô nhớ có địa chỉ DS:SI
 - MOV [DI], AX ; copy nội dung của AX vào 2 ô nhớ liên tiếp DS: DI và DS: (DI +1)



2.4.5 Chế độ địa chỉ tương đối cơ sở (Based relative Addressing Mode)

- Một toán hạng là thanh ghi cơ sở BX, BP và các hằng số biểu diễn giá trị dịch chuyển
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
 - MOV CX, [BX]+10 ; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+10 và DS:BX+11 vào CX
 - MOV CX, [BX+10] ; Cách viết khác của lệnh trên
 - MOV AL, [BP]+5 ; copy nội dung của ô nhớ SS:BP+5 vào thanh ghi AL

2.4.6 Chế độ địa chỉ tương đối chỉ số (Indexed relative Addressing Mode)

- Một toán hạng là thanh ghi chỉ số SI, DI và các hằng số biểu diễn giá trị dịch chuyển
- Toán hạng kia chỉ có thể là thanh ghi
- **Ví dụ:**
 - MOV AX, [SI]+10 ; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:SI+10 và DS:SI+11 vào AX
 - MOV AX, [SI+10] ; Cách viết khác của lệnh trên
 - MOV AL, [DI]+5 ; copy nội dung của ô nhớ DS:DI+5 vào thanh ghi AL

39

2.4.7 Chế độ địa chỉ tương đối chỉ số cơ sở (Based Indexed relative Addressing Mode)

- Kết hợp của 2 chế độ địa chỉ trước
- **Ví dụ:**
 - MOV AX, [BX] [SI]+8 ; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+SI+8 và DS:BX+SI+9 vào AX
 - MOV AX, [BX+SI+8] ; Cách viết khác của lệnh trên
 - MOV CL, [BP+DI+5] ; copy nội dung của ô nhớ SS:BP+DI+5 vào thanh ghi CL

40

20



Tóm tắt các chế độ địa chỉ

Chế độ địa chỉ	Toán hạng	Thanh ghi đoạn ngầm định
Thanh ghi	Thanh ghi	
Tức thì	Dữ liệu	
Trực tiếp	[offset]	DS
Gián tiếp qua thanh ghi	[BX] [SI] [DI]	DS DS DS
Tương đối cơ sở	[BX] + dịch chuyển [BP] + dịch chuyển	DS SS
Tương đối chỉ số	[DI] + dịch chuyển [SI] + dịch chuyển	DS DS
Tương đối chỉ số cơ sở	[BX] + [DI]+ dịch chuyển [BX] + [SI]+ dịch chuyển [BP] + [DI]+ dịch chuyển [BP] + [SI]+ dịch chuyển	DS DS SS SS

41



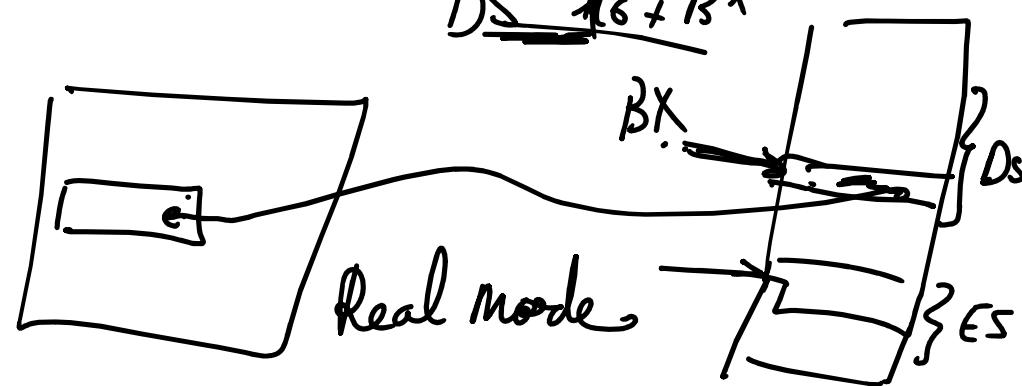
Bỏ chế độ ngầm định thanh ghi đoạn (Segment override)

- Ví dụ:

DS

- MOV AL, [BX]; Copy nội dung ô nhớ có địa chỉ DS:BX vào AL
- MOV AL, ES:[BX] ; Copy nội dung ô nhớ có địa chỉ ES:BX vào AL

*DS * 16 + BX*



42

21



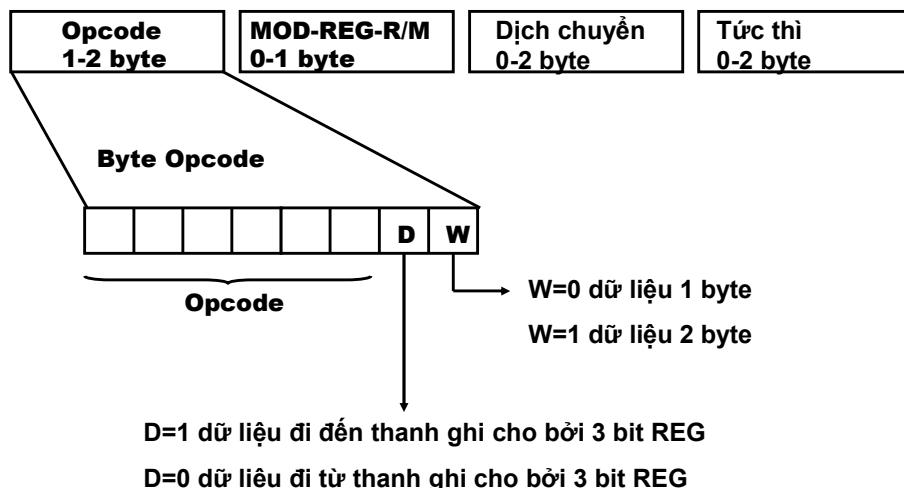
Chương 2: Bộ vi xử lý Intel 8088/8086

- 2.1 Cấu trúc bên trong
- 2.2 Sơ đồ chân
- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC
- 2.4 Các chế độ địa chỉ của 8086
- 2.5 Cách mã hóa lệnh của 8086
- 2.6 Mô tả tập lệnh của 8086
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



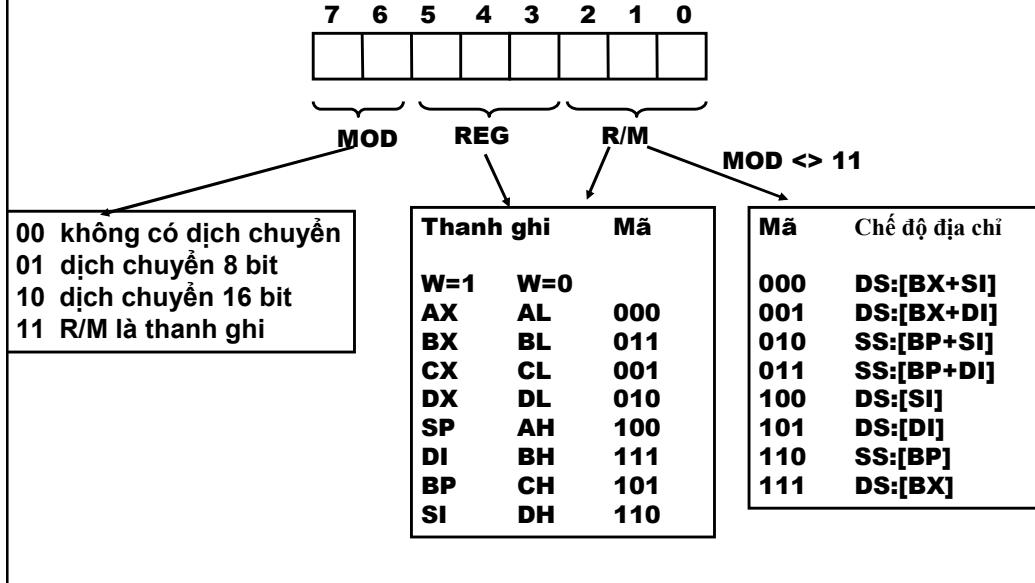
2.5 Cách mã hóa lệnh của 8086

- Một lệnh có độ dài từ 1 đến 6 byte





2.5 Cách mã hoá lệnh của 8086

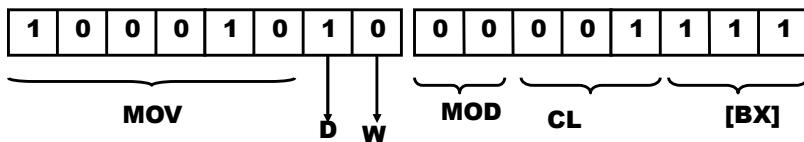


45



2.5 Cách mã hoá lệnh của 8086

- Ví dụ: chuyển lệnh MOV CL, [BX] sang mã máy
 - opcode MOV: 100010
 - Dữ liệu là 1 byte: W=0
 - Chuyển tới thanh ghi: D=1
 - Không có dịch chuyển: MOD=00
 - [BX] nên R/M=111
 - CL nên REG=001



Ví dụ 2: chuyển lệnh MOV [SI+0F3H], CL sang mã máy

46

23



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

2.2 Sơ đồ chân

2.3 Bản đồ bộ nhớ của máy tính IBM-PC

2.4 Các chế độ địa chỉ của 8086

2.5 Cách mã hoá lệnh của 8086

2.6 Mô tả tập lệnh của 8086

 2.6.1 Các lệnh di chuyển (thay đổi) dữ liệu

 2.6.2 Các lệnh số học và logic

 2.6.3 Các lệnh điều khiển chương trình

 2.6.4 Các lệnh khác

2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

2.2 Sơ đồ chân

2.3 Bản đồ bộ nhớ của máy tính IBM-PC

2.4 Các chế độ địa chỉ của 8086

2.5 Cách mã hoá lệnh của 8086

2.6 Mô tả tập lệnh của 8086

 2.6.1 Các lệnh di chuyển (thay đổi) dữ liệu

 2.6.2 Các lệnh số học và logic

 2.6.3 Các lệnh điều khiển chương trình

 2.6.4 Các lệnh khác

2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



2.6.1 Các lệnh di chuyển dữ liệu

Làm việc với bộ nhớ và thanh ghi

- MOV, XCHG, POP, PUSH, POPF, PUSHF, IN, OUT, LDS, LEA, LES
- Các lệnh di chuyển chuỗi MOVS, MOVSB, MOVSW, LODS(B/W)
- MOV
 - Dùng để chuyển giữa các thanh ghi, giữa 1 thanh ghi và 1 ô nhớ hoặc chuyển 1 số vào thanh ghi hoặc ô nhớ
 - Cú pháp: MOV Đích, nguồn
 - Lệnh này không tác động đến cờ
 - Ví dụ:
 - ⇒ MOV AX, BX
 - ⇒ MOV AH, 'A'
 - ⇒ MOV AL, [1234H]



2.6.1 Các lệnh di chuyển dữ liệu

Làm việc với bộ nhớ và thanh ghi

- Khả năng kết hợp toán hạng của lệnh MOV

Dịch Nguồn	Thanh ghi đa năng	Thanh ghi đoạn	ô nhớ	Hằng số
Thanh ghi đa năng	YES	YES	YES	NO
Thanh ghi đoạn	YES	NO	YES	NO
Ô nhớ	YES	YES	NO	NO
Hằng số	YES	NO	YES	NO



2.6.1 Các lệnh di chuyển dữ liệu Làm việc với bộ nhớ và thanh ghi

- **Lệnh XCHG (Exchange 2 operands)**
 - Dùng để hoán chuyển nội dung giữa hai thanh ghi, giữa 1 thanh ghi và 1 ô nhớ
 - Cú pháp: XCHG Đích, nguồn
 - Giới hạn: toán hạng không được là thanh ghi đoạn
 - Lệnh này không tác động đến cờ
 - Ví dụ:
 - ⇒ XCHG AX, BX
 - ⇒ XCHG AX, [BX]
- **Lệnh LEA- Load Effective Address**
 - Nạp địa chỉ hiệu dụng vào thanh ghi
 - Thực hiện: Đích=Địa chỉ lệnh (hoặc hiệu dụng) của nguồn
 - Ví dụ:
 - ⇒ LEA DX,Message
 - ⇒ LEA CX,[BX][DI]
- **LDS-Load Register and DS with Words from Memory**
- **LES-Load Register and ES with Words from Memory**



2.6.1 Các lệnh di chuyển dữ liệu Làm việc với ngăn xếp

- **Lệnh PUSH**
 - Dùng để cất 1 từ từ thanh ghi hoặc ô nhớ vào đỉnh ngăn xếp
 - Cú pháp: PUSH Nguồn
 - Mô tả: SP=SP-2, Nguồn => {SP}
 - Giới hạn: thanh ghi 16 bit hoặc là 1 từ nhớ
 - Lệnh này không tác động đến cờ
 - Ví dụ:
 - ⇒ PUSH BX
 - ⇒ PUSH PTR[BX]
- **Lệnh PUSHF**
 - Cất nội dung của thanh ghi cờ vào ngăn xếp



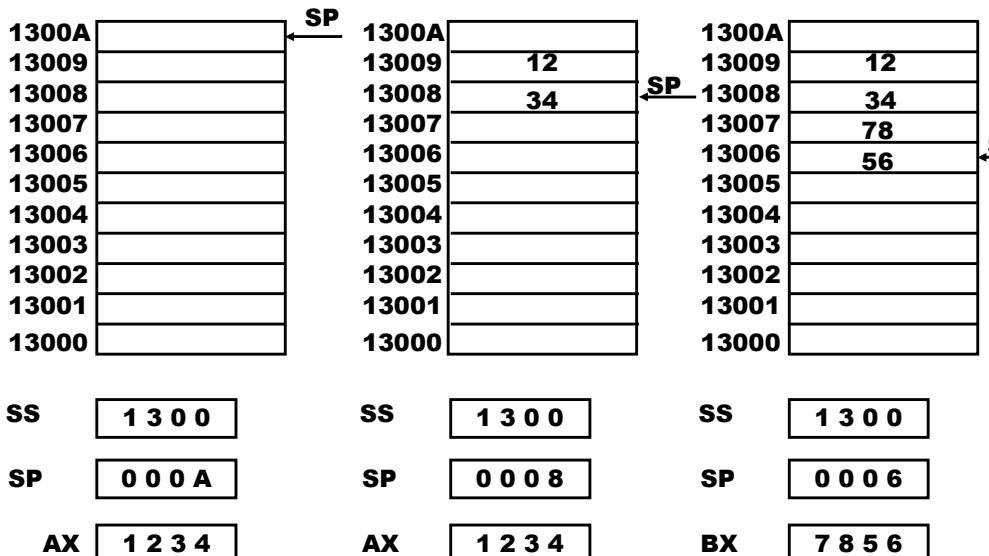
2.6.1 Các lệnh di chuyển dữ liệu

Làm việc với ngăn xếp

- Ví dụ về lệnh PUSH

PUSH AX

PUSH BX



53



2.6.1 Các lệnh di chuyển dữ liệu

Làm việc với ngăn xếp

- Lệnh POP

- Dùng để lấy lại 1 từ vào thanh ghi hoặc ô nhớ từ đỉnh ngăn xếp
- Cú pháp: POP Đích
- Mô tả: {SP} => Đích, SP=SP+2
- Giới hạn: thanh ghi 16 bit (trừ IP) hoặc là 1 từ nhớ
- Lệnh này không tác động đến cờ
- Ví dụ:
 - POP BX
 - POP PTR[BX]

- Lệnh POPF

- Lấy 1 từ từ đỉnh ngăn xếp rồi đưa vào thanh ghi cờ

54

2.6.1 Các lệnh di chuyển dữ liệu Làm việc với cổng

- **Lệnh IN**
 - Dùng để đọc 1 byte hoặc 2 byte dữ liệu từ cổng vào thanh ghi AL hoặc AX
 - Cú pháp: IN Acc, Port
 - Lệnh này không tác động đến cờ
 - Ví dụ:

⇒ IN AX, 00H	MOV DX,03F8h
⇒ IN AL, F0H	
⇒ IN AX, DX	IN AL,DX
- **Lệnh OUT**
 - Dùng để đưa 1 byte hoặc 2 byte dữ liệu từ thanh ghi AL hoặc AX ra cổng
 - Cú pháp: OUT Port, Acc
 - Lệnh này không tác động đến cờ
 - Ví dụ:

⇒ OUT 00H, AX	
⇒ OUT F0H, AL	
⇒ OUT DX, AX	
- **Chú ý:**
 - Nếu Port là hằng số: Port={0 . . . FFH}
 - Nếu muốn truy cập cổng có địa chỉ lớn hơn FFH thì phải dùng Port là DX

2.6.1 Các lệnh di chuyển dữ liệu Làm việc với mảng, chuỗi

- **Các lệnh di chuyển chuỗi MOVS, MOVS, MOVSW**
 - Dùng để chuyển một phần tử của chuỗi này sang một chuỗi khác
 - Cú pháp: MOVS chuỗi đích, chuỗi nguồn

MOVSB	
MOVSW	
 - Thực hiện:
 - ⇒ DS:SI là địa chỉ của phần tử trong chuỗi nguồn
 - ⇒ ES:DI là địa chỉ của phần tử trong chuỗi đích
 - ⇒ Sau mỗi lần chuyển SI=SI +/- 1, DI=DI +/- 1 hoặc SI=SI +/- 2, DI=DI +/- 2 tùy thuộc vào cờ hướng DF là 0/1
 - Lệnh này không tác động đến cờ
 - Ví dụ:
 - ⇒ MOVS byte1, byte2
- **LODS/LODSB/LODSW-Load String Byte/Word into AL/AX**



Lệnh di chuyển chuỗi

- Bài toán thường gặp:**

- Copy ChuoiNguon sang ChuoiDich, kích thước : N

Nạp địa chỉ ChuoiNguon vào 1 thanh ghi: SI

Nạp địa chỉ ChuoiDich vào 1 thanh ghi: DI

Lặp N lần

$AL \leftarrow [SI]$ $[DI] \leftarrow AL$ $SI = SI + 1$ $DI = DI + 1$	$\left. \right\}$ MOVSB
--	---------------------------------------



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

2.2 Sơ đồ chân

2.3 Bản đồ bộ nhớ của máy tính IBM-PC

2.4 Các chế độ địa chỉ của 8086

2.5 Cách mã hóa lệnh của 8086

2.6 Mô tả tập lệnh của 8086

- 2.6.1 Các lệnh di chuyển (thay đổi) dữ liệu

- 2.6.2 Các lệnh số học và logic

- 2.6.3 Các lệnh điều khiển chương trình

- 2.6.4 Các lệnh khác

2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



2.6.2 Các lệnh số học và logic

- ADD, ADC, SUB, MUL, IMUL, DIV, IDIV, INC, DEC
- AND, OR, NOT, NEG, XOR
- Lệnh quay và dịch: RCL, RCR, SAL, SAR, SHL, SHR
- Lệnh so sánh: CMP, CMPS

- **Lệnh ADD**
 - Lệnh cộng hai toán hạng
 - Cú pháp: ADD Đích, nguồn
 - Thực hiện: Đích=Đích + nguồn
 - Giới hạn: toán hạng không được là 2 ô nhớ và thanh ghi đoạn
 - Lệnh này thay đổi cờ: AF, CF, OF, PF, SF, ZF
 - Ví dụ:
 - ⇒ ADD AX, BX
 - ⇒ ADD AX, 40H

61



2.6.2 Các lệnh số học và logic

Các lệnh số học

- **Lệnh ADC**
 - Lệnh cộng có nhớ hai toán hạng
 - Cú pháp: ADC Đích, nguồn
 - Thực hiện: Đích=Đích + nguồn+CF
 - Giới hạn: toán hạng không được là 2 ô nhớ và thanh ghi đoạn
 - Lệnh này thay đổi cờ: AF, CF, OF, PF, SF, ZF
 - Ví dụ:
 - ⇒ ADD AL, 30H
- **Lệnh SUB**
 - Lệnh trừ
 - Cú pháp: SUB Đích, nguồn
 - Thực hiện: Đích=Đích - nguồn
 - Giới hạn: toán hạng không được là 2 ô nhớ và thanh ghi đoạn
 - Lệnh này thay đổi cờ: AF, CF, OF, PF, SF, ZF
 - Ví dụ:
 - ⇒ SUB AL, 30H

62



2.6.2 Các lệnh số học và logic

Các lệnh số học

- **Lệnh MUL**

- Lệnh nhân số không dấu
- Cú pháp: MUL nguồn (nguồn phải là thanh ghi)
- Thực hiện:
 - ⇒ AX=AL* nguồn 8bit
 - ⇒ DXAX=AX* nguồn 16bit
- Lệnh này thay đổi cờ: CF, OF
- Ví dụ:
 - ⇒ MUL BL ; AX=AL*BL

- **Lệnh IMUL**

- nhân số có dấu



2.6.2 Các lệnh số học và logic

Các lệnh số học

- **Lệnh DIV**

- Lệnh chia 2 số không dấu
- Cú pháp: DIV nguồn
- Thực hiện:
 - ⇒ AL = thương (AX / nguồn 8bit) ; AH=dư (AX / nguồn 8bit)
 - ⇒ AX = thương (DXAX / nguồn 16bit) ; DX=dư (DXAX / nguồn 16bit)
- Lệnh này không thay đổi cờ
- Ví dụ:
 - ⇒ DIV BL

- **Lệnh IDIV**

- chia 2 số có dấu



2.6.2 Các lệnh số học và logic

Các lệnh số học

- **Lệnh INC-Increase by 1**
 - Lệnh cộng 1 vào toán hạng là thanh ghi hoặc ô nhớ
 - Cú pháp: INC Đích
 - Thực hiện: Đích=Đích + 1
 - Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
 - Ví dụ:
⇒ INC AX
- **Lệnh DEC-Decrease by 1**
 - Lệnh trừ 1 từ nội dung một thanh ghi hoặc ô nhớ
 - Cú pháp: DEC Đích
 - Thực hiện: Đích=Đích - 1
 - Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
 - Ví dụ:
⇒ DEC [BX]



2.6.2 Các lệnh số học và logic

Các lệnh logic

- **Lệnh AND- VÀ logic**
 - Lệnh AND logic 2 toán hạng
 - Cú pháp: AND Đích, nguồn
 - Thực hiện: Đích=Đích And nguồn
 - Giới hạn: toán hạng không được là 2 ô nhớ hoặc thanh ghi đoạn
 - Lệnh này thay đổi cờ: PF, SF, ZF và xoá cờ CF, OF
 - Ví dụ:
⇒ AND BL, 0FH
- **Lệnh XOR, OR: tương tự như lệnh AND**
- **Lệnh NOT: đảo từng bit của toán hạng**
- **Lệnh NEG: đảo dấu của toán hạng, xác định số bù 2 của toán hạng**



2.6.2 Các lệnh số học và logic

Các lệnh so sánh

- **Lệnh CMP-Compare 2 operands to Update the Flags**
 - Lệnh so sánh 2 byte hoặc 2 từ
 - Cú pháp: CMP Đích, nguồn
 - Thực hiện:
 - ⇒ Đích = nguồn : SF=0 ZF=1
 - ⇒ Đích > nguồn : SF=0 ZF=0
 - ⇒ Đích < nguồn : SF=1 ZF=0
 - Giới hạn: toán hạng phải cùng độ dài và không được là 2 ô nhớ
- **Lệnh CMPS**
 - Dùng để so sánh từng phần tử của 2 chuỗi có các phần tử cùng loại
 - Cú pháp: CMPS chuỗi đích, chuỗi nguồn
 - CMPSB
 - CMPSW
 - Thực hiện:
 - ⇒ DS:SI là địa chỉ của phần tử trong chuỗi nguồn
 - ⇒ ES:DI là địa chỉ của phần tử trong chuỗi đích
 - ⇒ Sau mỗi lần so sánh SI=SI +/- 1, DI=DI +/- 1 hoặc SI=SI +/- 2, DI=DI +/- 2 tùy thuộc vào cờ hướng DF là 0/1
 - Cập nhật cờ AF, CF, OF, PF, SF, ZF



2.6.2 Các lệnh số học và logic

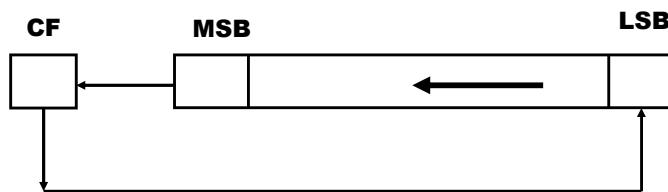
Các lệnh so sánh

- **TEST - AND 2 operands to Update the Flags**
 - Cú pháp: TEST Operand1,Operand2
 - Ví dụ: Kiểm tra bit 0 của AL
 - ⇒ TEST AL,01h ;ZF=1 nếu AL.0=0, ZF=0 nếu AL.0=1
- **Các lệnh thiết lập cờ**
 - STC-Set the Carry Flag CF=1
 - STD-Set the Direction Flag DF=1
 - STI-Set the Interrupt Flag IF=1
- **Các lệnh xóa cờ**
 - CLC-Clear the Carry Flag CF=0
 - CLD-Clear the Direction Flag DF=0
 - CLI-Clear the Interrupt Flag IF=0
- **CMC-Complement the Carry Flag CF=not(CF)**

2.6.2 Các lệnh số học và logic Dịch và quay

- **Lệnh RCL-Rotate through Carry flag to the Left**

- Lệnh quay trái thông qua cờ nhó
- Cú pháp: RCL Đích, CL
RCL Đích, Số lần quay
- Thực hiện: quay trái đích CL lần
- Lệnh này thay đổi cờ: CF, OF

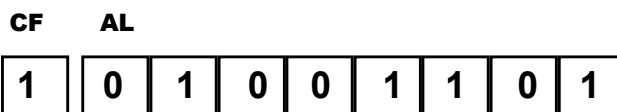


- **Lệnh RCR-Rotate through Carry flag to the Right**

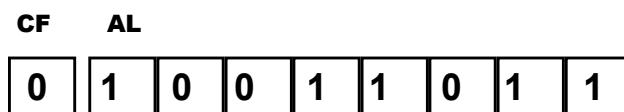
- Lệnh quay phải thông qua cờ nhó

2.6.2 Các lệnh số học và logic Dịch và quay

- Ví dụ: RCL AL,1



- Ví dụ: RCR AL,1





2.6.2 Các lệnh số học và logic Dịch và quay

- **Lệnh SAL-Shift Arithmetically Left**

- Lệnh dịch trái số học
- Cú pháp: SAL Đích, CL hoặc SAL Đích, số lần dịch
- Thực hiện: dịch trái đích CL bit tương đương với $\text{Đích}=\text{Đích} \times 2^{\text{CL}}$
- Lệnh này thay đổi cờ SF, ZF, PF



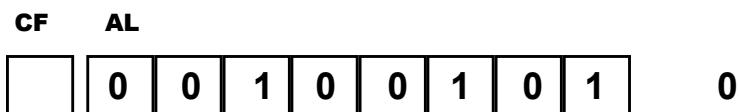
- **Lệnh SHL-SHift Left**

- Lệnh dịch trái logic tương tự như SAL



2.6.2 Các lệnh số học và logic Dịch và quay

- Ví dụ: **SAL AL,1**



- Trước: **AL=25H=37**

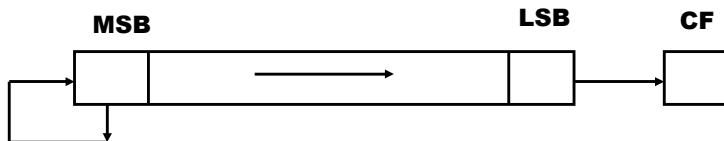
- Sau: **AL=4AH=74**



2.6.2 Các lệnh số học và logic Dịch và quay

- Lệnh SAR-Shift Arithmetically Right**

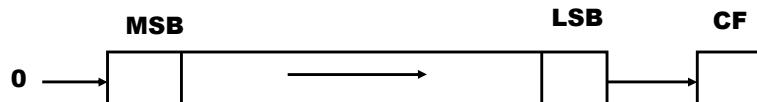
- Lệnh dịch phải số học
- Cú pháp: SAR Đích, CL hoặc SAR Đích, số lần dịch
- Thực hiện: dịch phải đích CL bit
- Lệnh này thay đổi cờ SF, ZF, PF, CF mang giá trị của LSB



2.6.2 Các lệnh số học và logic Dịch và quay

- Lệnh SHR-SHift Right**

- Lệnh dịch phải logic
- Cú pháp: SHR Đích, CL hoặc SHR Đích, số lần dịch
- Thực hiện: dịch phải đích CL bit
- Lệnh này thay đổi cờ SF, ZF, PF, CF mang giá trị của LSB





2.6.2 Các lệnh số học và logic Dịch và quay

- Ví dụ: SAR AL,1

AL									CF
1	0	1	0	0	1	1	0		

Trước: $AL=A6H=-90(166)$ Sau: $AL=D3H=-45(211)$

- Ví dụ: SHR AL,1

AL									CF
0	1	0	1	0	0	1	1	0	

Trước: $AL=A6H=-90 (166)$ Sau: $AL=53H=+83$



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

2.2 Sơ đồ chân

2.3 Bản đồ bộ nhớ của máy tính IBM-PC

2.4 Các chế độ địa chỉ của 8086

2.5 Cách mã hoá lệnh của 8086

2.6 Mô tả tập lệnh của 8086

2.6.1 Các lệnh di chuyển (thay đổi) dữ liệu

2.6.2 Các lệnh số học và logic

2.6.3 Các lệnh điều khiển chương trình

2.6.4 Các lệnh khác

2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



2.6.3 Các lệnh điều khiển chương trình Lệnh nhảy không điều kiện JMP

- Dùng để nhảy tới một địa chỉ trong bộ nhớ

- 3 loại: nhảy ngắn, gần và xa

- Lệnh nhảy ngắn (short jump)

⇒ Độ dài lệnh 2 bytes:



⇒ Phạm vi nhảy: -128 đến 127 bytes so với lệnh tiếp theo lệnh JMP

⇒ Thực hiện: IP=IP + độ lệch

⇒ Ví dụ:

```

XOR AX, AX
Nhan: MOV BX, 1
ADD AX, BX
JMP SHORT Nhan

```



2.6.3 Các lệnh điều khiển chương trình Lệnh nhảy không điều kiện JMP

- Lệnh nhảy gần (near jump)

⇒ Phạm vi nhảy: ± 32 Kbytes so với lệnh tiếp theo lệnh JMP

⇒ Ví dụ:

```

XOR BX, BX
Nhan: MOV AX, 1
ADD AX, BX
JMP NEAR Nhan

```

```

XOR CX, CX
MOV AX, 1
ADD AX, BX
JMP NEAR PTR BX

```

```

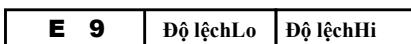
XOR CX, CX
MOV AX, 1
ADD AX, BX
JMP WORD PTR [BX]

```

Thực hiện: IP=IP+ độ lệch

IP=BX

IP=[BX+1] [BX]



Nhảy gián tiếp

2.6.3 Các lệnh điều khiển chương trình

Lệnh nhảy không điều kiện JMP

Lệnh nhảy xa (far jump)

⇒ Độ dài lệnh 5 bytes đối với nhảy tới nhãn:

E	A	d/c Offset Lo	d/c offset Hi	d/c segment Lo	d/c segment Hi
----------	----------	---------------	---------------	----------------	----------------

⇒ Phạm vi nhảy: nhảy trong 1 đoạn mã hoặc nhảy sang đoạn mã khác

⇒ Ví dụ:

EXTRN Nhan: FAR

```
Next: MOV AX, 1
      ADD AX, BX
      JMP FAR PTR Next
      .....
      JMP FAR Nhan
```

Thực hiện: IP=IP của nhãn
CS=CS của nhãn

XOR CX, CX

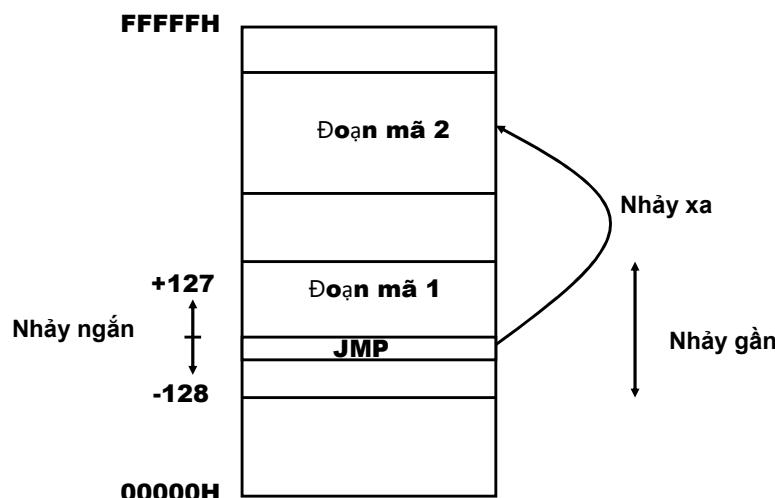
```
MOV AX, 1
ADD AX, BX
JMP DWORD PTR [BX]
```

$$\begin{aligned} \text{IP} &= [\text{BX}+1][\text{BX}] \\ \text{CS} &= [\text{BX}+3][\text{BX}+2] \end{aligned}$$

79

2.6.3 Các lệnh điều khiển chương trình

Tóm tắt lệnh JMP



80

40

2.6.3 Các lệnh điều khiển chương trình Lệnh nhảy có điều kiện

- JE/JZ, JNE/JNZ, JG/JNLE, JGE/JNL, JL/JNGE, JLE/JNG (dùng cho số có dấu) và
JA/JNBE, JB/JC/JNAE, JAE/JNB/JNC, JBE/JNA (dùng cho số không dấu) và
JNP/JPO, JP/JPE, JNS
- Nhảy được thực hiện phụ thuộc vào các cờ
- Là các lệnh nhảy ngắn
- Ví dụ:

```
Nhan1: XOR BX, BX
Nhan2: MOV AX, 1
        CMP AL, 10H
        JNE Nhan1
        JE Nhan2
```

Thực hiện: IP=IP + độ dịch

2.6.3 Các lệnh điều khiển chương trình Lệnh lặp LOOP

- LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ
- Là lệnh phối hợp giữa DEC CX và JNZ

```
XOR AL, AL
MOV CX, 16
Lap: INC AL
LOOP Lap
```

Lặp đến khi CX=0

```
XOR AL, AL
MOV CX, 16
Lap: INC AL
CMP AL, 10
LOOPE Lap
```

**Lặp đến khi CX=0
hoặc AL<>10 (ZF=0)**

```
XOR AL, AL
MOV CX, 16
Lap: INC AL
CMP AL, 10
LOOPNE Lap
```

**Lặp đến khi CX=0
hoặc AL=10**



2.6.3 Các lệnh điều khiển chương trình Lệnh CALL (và RET)

- Dùng để gọi chương trình con
- Có 2 loại: **CALL gần** và **CALL xa**
 - **CALL gần (near call):** tương tự như nhảy gần
 - ⇒ Gọi chương trình con ở trong cùng một đoạn mã

```
Tong PROC NEAR
    ADD AX, BX
    ADD AX, CX
    RET
Tong ENDP
...
CALL Tong
```

Cắt IP vào ngăn xếp
 IP=IP + dịch chuyển
 RET: lấy IP từ ngăn xếp

```
Tong PROC NEAR
    ADD AX, BX
    ADD AX, CX
    RET
Tong ENDP
...
MOV BX, OFFSET Tong
CALL BX
```

Cắt IP vào ngăn xếp
 IP= BX
 RET: lấy IP từ ngăn xếp

CALL WORD PTR [BX]

Cắt IP vào ngăn xếp
 IP= [BX+1] [BX]
 RET: lấy IP từ ngăn xếp



2.6.3 Các lệnh điều khiển chương trình Lệnh CALL

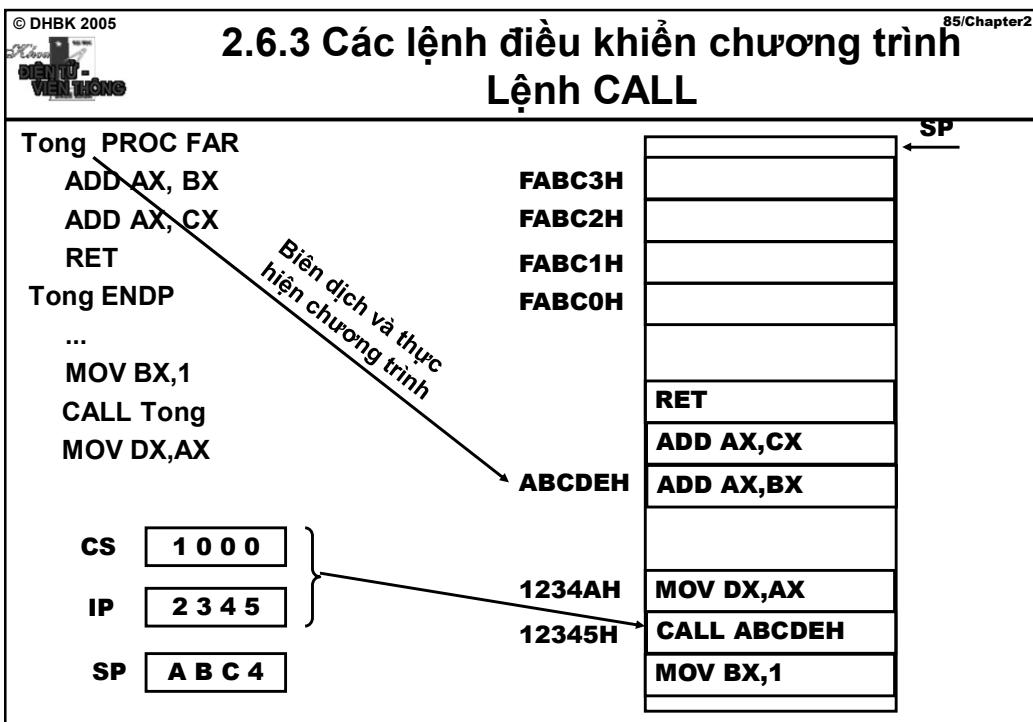
- **CALL xa (far call):** tương tự như nhảy xa
 - ⇒ Gọi chương trình con ở ngoài đoạn mã

```
Tong PROC FAR
    ADD AX, BX
    ADD AX, CX
    RET
Tong ENDP
...
CALL Tong
```

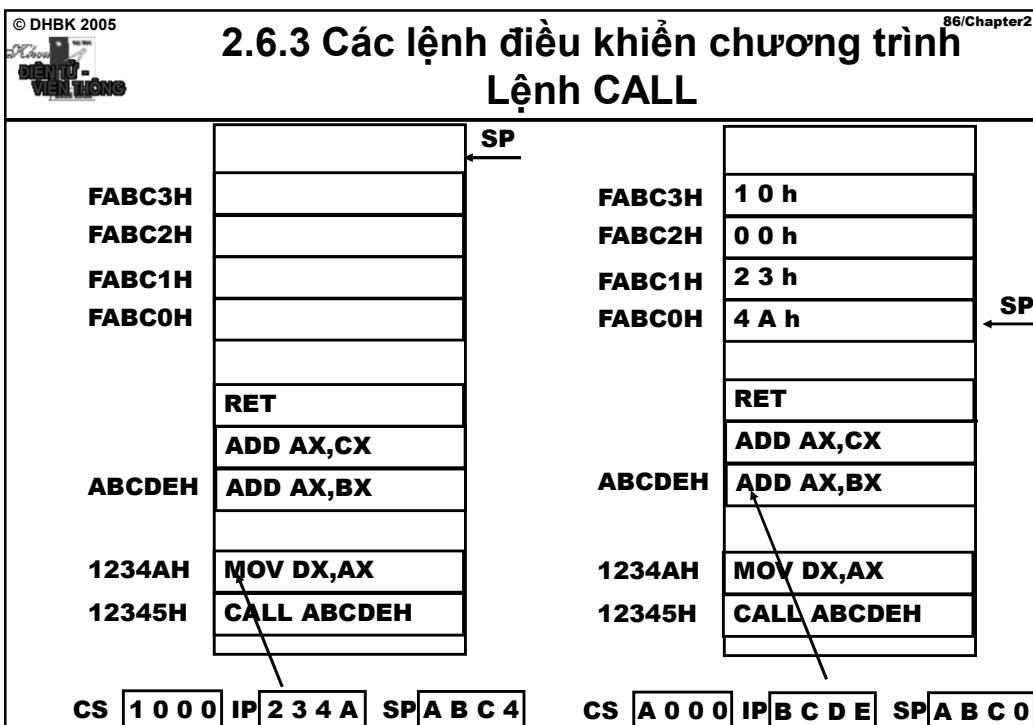
Cắt CS vào ngăn xếp
 Cắt IP vào ngăn xếp
 IP=IP của Tong
 CS=CS của Tong
 RET: lấy IP từ ngăn xếp
 lấy CS từ ngăn xếp

CALL DWORD PTR [BX]

Cắt CS vào ngăn xếp
 Cắt IP vào ngăn xếp
 IP = [BX+1][BX]
 CS= [BX+3][BX+2]
 RET: lấy IP từ ngăn xếp
 lấy CS từ ngăn xếp



85

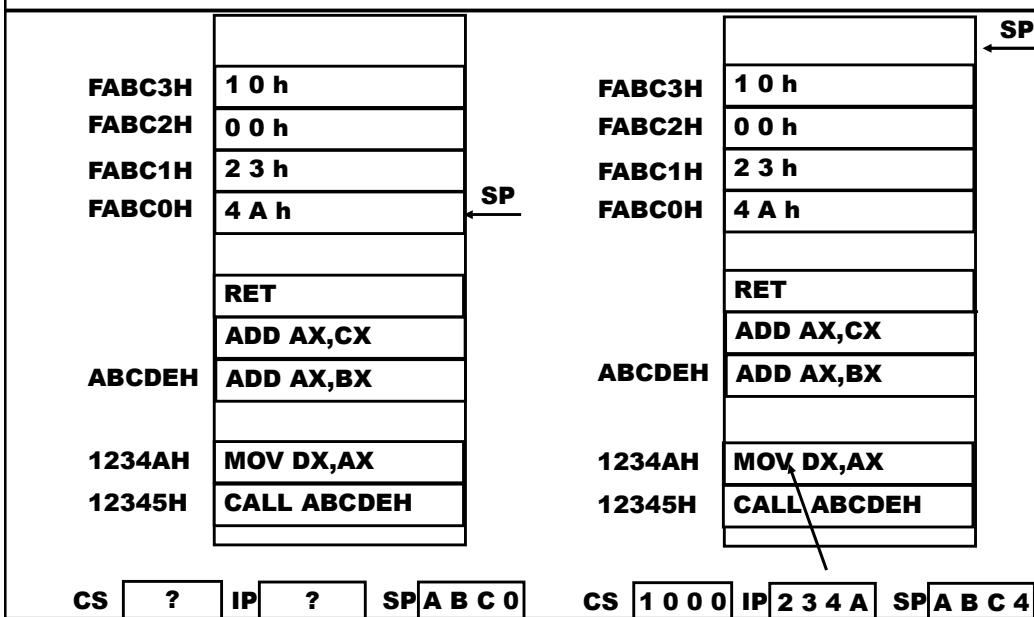


86

43

2.6.3 Các lệnh điều khiển chương trình RET

87/Chapter2



87

2.6.3 Các lệnh điều khiển chương trình Lệnh ngắt INT và IRET

88/Chapter2

- INT gọi chương trình con phục vụ ngắt (CTCPVN)
- Bảng vector ngắt: 1 Kbytes 00000H đến 003FFH
 - 256 vector ngắt
 - 1 vector 4 bytes, chứa IP và CS của CTCPVN
 - 32 vector đầu dành riêng cho Intel
 - 224 vector sau dành cho người dùng
- Cú pháp: INT Number
- Ví dụ: INT 21H gọi CTCPVN của DOS

88

44



2.6.3 Các lệnh điều khiển chương trình

Lệnh ngắt INT và IRET

- **Thực hiện INT:**

- Cắt thanh ghi cờ vào ngăn xếp
- IF=0 (cấm các ngắt khác tác động), TF=0 (chạy suốt)
- Cắt CS vào ngăn xếp
- Cắt IP vào ngăn xếp
- IP=[N*4], CS=[N*4+2]

- **Gặp IRET:**

- Lấy IP từ ngăn xếp
- Lấy CS từ ngăn xếp
- Lấy thanh ghi cờ từ ngăn xếp



Chương 2: Bộ vi xử lý Intel 8088/8086

2.1 Cấu trúc bên trong

2.2 Sơ đồ chân

2.3 Bản đồ bộ nhớ của máy tính IBM-PC

2.4 Các chế độ địa chỉ của 8086

2.5 Cách mã hóa lệnh của 8086

2.6 Mô tả tập lệnh của 8086

- 2.6.1 Các lệnh di chuyển (thay đổi) dữ liệu

- 2.6.2 Các lệnh số học và logic

- 2.6.3 Các lệnh điều khiển chương trình

- 2.6.4 Các lệnh khác

2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286



2.6.4 Các lệnh khác

- **NOP-No Operation** (tiêu tốn 3 chu kỳ đồng hồ)
- **WAIT-Wait for TEST or INTR Signal**
 - Chờ cho đến khi có tín hiệu mức thấp tác động vào chân TEST hoặc mức cao tác động vào chân INTR)
- **HALT-Halt Processing**
 - VXL dừng hoạt động. Để thoát khỏi trạng thái dừng phải tác động vào các chân INTR, NMI, RESET.
- **ESC-Escape**
 - Truyền dữ liệu cho bộ đồng xử lý toán học 8087



Chương 2: Bộ vi xử lý Intel 8088/8086

- 2.1 Cấu trúc bên trong**
- 2.2 Sơ đồ chân**
- 2.3 Bản đồ bộ nhớ của máy tính IBM-PC**
- 2.4 Các chế độ địa chỉ của 8086**
- 2.5 Cách mã hoá lệnh của 8086**
- 2.6 Mô tả tập lệnh của 8086**
- 2.7 Cách quản lý bộ nhớ ở chế độ bảo vệ ở các máy tính từ 80286**



2.7 Đánh địa chỉ bộ nhớ ở chế độ bảo vệ

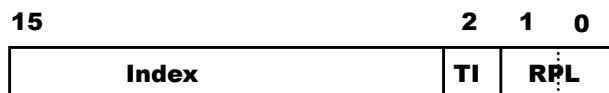
- **Lý do:**
 - Hỗ trợ đa nhiệm từ 80286.
 - Tương thích ngược với 8086.
 - Cho phép truy cập dữ liệu và chương trình ở vùng nhớ trên 1M
- **Thanh ghi lệnh chứa địa chỉ lệch**
- **Thanh ghi đoạn chứa từ chọn đoạn (segment selector)**
 - từ chọn đoạn chọn 1 phần tử trong 1 trong 2 bảng mô tả đoạn (Descriptor Table), mỗi bảng có kích thước 64 KB
 - ⇒ **Bảng mô tả đoạn toàn cục (Global DT):** chứa thông tin về các đoạn của bộ nhớ mà tất cả các chương trình có thể truy nhập
 - ⇒ **Bảng mô tả đoạn cục bộ (Local DT):** chứa thông tin về các đoạn của 1 chương trình
 - Mô tả đoạn chứa thông tin về địa chỉ bắt đầu của đoạn

93



2.7 Đánh địa chỉ bộ nhớ ở chế độ bảo vệ

segment selector



RPL: mức ưu tiên yêu cầu, 00 cao nhất, 11 thấp nhất

TI=0, sử dụng bảng toàn cục, TI=1 sử dụng bảng cục bộ

Index: 13 bit chỉ số để chọn 1 trong 8K mô tả đoạn trong bảng mô tả đoạn

7	0 0 0 0 0 0 0	0 0 0 0 0 0 0
5	Access rights	Base(B23-B16)
3	Base(B15-B0)	
1	Limit(L15-L0)	

mô tả đoạn của 80286

6	7	Base(B31-B24)	G	D	O	A	Limit
4	5	Access rights	Base(B23-B16)				V(L19-L16)
2	3	Base(B15-B0)					
0	1	Limit(L15-L0)					

mô tả đoạn từ 80386

Base: xác định địa chỉ bắt đầu của đoạn

Limit: giới hạn kích thước tối đa của đoạn

94



2.7 Đánh địa chỉ bộ nhớ ở chế độ bảo vệ

Figure 1(a) – Real Mode Addressing

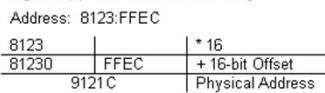
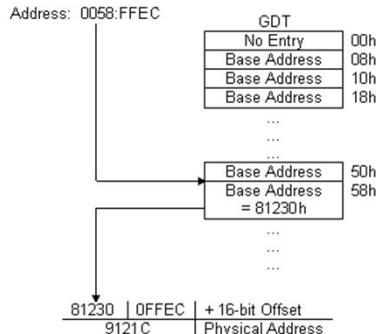


Figure 1(b) – Protected Mode Addressing



- Mỗi chương trình sử dụng một số vùng nhớ. CPU bảo vệ vùng nhớ đó, không cho phép chương trình khác truy cập vào.
- Không cho phép đọc mã lệnh ở đoạn dữ liệu và ngược lại.



2.7 Đánh địa chỉ bộ nhớ ở chế độ bảo vệ

- 80286**
 - Base 24 bit: 000000H đến FFFFFFFH (16 M đoạn)
 - Limit 16 bit: kích thước đoạn: từ 1 đến 64 KB
 - Địa chỉ vật lý= Base + độ lệch
 - 1 chương trình có thể sử dụng tối đa: $2^8 \times 64 \text{ KB} = 1\text{GB}$ bộ nhớ ảo (virtual memory)
- 80386/486/Pentium**
 - Base 32 bit: 00000000H đến FFFFFFFFH (4 GB)
 - Limit 20 bit:
 - ⇒ G=0: kích thước đoạn: từ 1 đến 1MB
 - ⇒ G=1: kích thước đoạn từ 4K đến 4 GB
 - Địa chỉ vật lý= Base + độ lệch
 - 1 chương trình có thể sử dụng tối đa: $2^8 \times 4 \text{ GB} = 64 \text{ Terabytes}$ bộ nhớ



Nội dung môn học

1. Giới thiệu chung về hệ vi xử lý
2. Bộ vi xử lý Intel 8088/8086
3. Lập trình hợp ngữ cho 8086
4. Tổ chức vào ra dữ liệu
5. Ngắt và xử lý ngắt
6. Truy cập bộ nhớ trực tiếp DMA
7. Các bộ vi xử lý trên thực tế



Chương 3 Lập trình hợp ngữ với 8086

- 3.1 Giới thiệu khung của chương trình hợp ngữ
- 3.2 Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- 3.3 Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- 3.4 Một số chương trình cụ thể



Chương 3 Lập trình hợp ngữ với 8086

- **3.1 Giới thiệu khung của chương trình hợp ngữ**
 - 3.1.1 Cú pháp của chương trình hợp ngữ
 - 3.1.2 Dữ liệu cho chương trình
 - 3.1.3 Biến và hằng
 - 3.1.4 Khung của một chương trình hợp ngữ
- **Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC**
- **Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ**
- **Một số chương trình cụ thể**



Chương 3 Lập trình hợp ngữ với 8086

- **3.1 Giới thiệu khung của chương trình hợp ngữ**
 - 3.1.1 Cú pháp của chương trình hợp ngữ
 - Dữ liệu cho chương trình
 - Biến và hằng
 - Khung của một chương trình hợp ngữ
- **Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC**
- **Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ**
- **Một số chương trình cụ thể**



3.1.1 Cú pháp của chương trình hợp ngữ

```

1. .Model Small               khai báo kiểu kích thước bộ nhớ
2. .Stack 100                khai báo đoạn ngăn xếp
3. .Data                      khai báo đoạn dữ liệu
4. Tbao DB 'Chuoi da sap xep:', 10, 13
5. MGB DB 'a', 'Y', 'G', 'T', 'y', 'Z', 'U', 'B', 'D', 'E',
6. DB '$'
7. .Code                     khai báo đoạn mã lệnh
8. MAIN Proc                 bắt đầu chương trình chính
9.     MOV AX, @Data          :khai dau DS
10.    MOV DS, AX
11.    MOV BX, 10             ;BX: so phan tu cua mang
12.    LEA      DX, MGB       ;DX chi vao dau mang byte
13.    DEC      BX            ;so vong so sanh phai lam
14.    LAP:   MOV SI, DX      ;SI chi vao dau mang
15.        MOV      CX, BX      ;CX so lan so cua vong so
16.        MOV      DI, SI      ;gia su ptu dau la max
17.        MOV      AL, [DI]     ;AL chua phan tu max
18.    TIMMAX: INC SI         ;chỉ vao phan tu ben canh
19.        CMP      [SI], AL      ;phan tu moi > max?
20.        JNG      TIETP       ;khong, tim max
21.        MOV      DI, SI      ;dung, DI chi vao max
22.        MOV      AL, [DI]     ;AL chua phan tu max
23.    TIETP: LOOP DOICHO     ;tim max cua mot vong so
24.        CALL DOICHO          ;doi cho max voi so moi
25.        DEC      BX            ;so vong so con lai
26.        JNE      LAF           ;lam tiep vong so moi
27.        MOV      AH, 9          ;hien thi chuoi da sap xep
28.        LEA      DX, Thao
29.        INT      21H
30.        MOV      AH, 4CH
31.        INT      21H
32.        INT      AX
33. MAIN Endp                  kết thúc chương trình chính
34. DOICHO Proc                bắt đầu chương trình con
35.     PUSH AX
36.     MOV      AL, [SI]
37.     XCHG   AL, [DI]
38.     MOV      [SI], AL
39.     POP      AX
40.     RET
41. DOICHO Endp
42. END MAIN                   kết thúc đoạn mã

```

5



3.1.1 Cú pháp của chương trình hợp ngữ

- **Tên Mã lệnh Các toán hạng ; chú giải**
- **Chương trình dịch không phân biệt chữ hoa, chữ thường**
- **Trường tên:**
 - chứa các nhãn, tên biến, tên thủ tục
 - độ dài: 1 đến 31 ký tự
 - tên không được có dấu cách, không bắt đầu bằng số
 - được dùng các ký tự đặc biệt: ? . @_ \$ %
 - dấu . phải được đặt ở vị trí đầu tiên nếu sử dụng
 - Nhãn kết thúc bằng dấu :
 - Ví dụ:

TWO_WORD
?1
two-word
.@?
1word
Let's_go

6



Chương 3 Lập trình hợp ngữ với 8086

- **3.1 Giới thiệu khung của chương trình hợp ngữ**
 - Cú pháp của chương trình hợp ngữ
 - 3.1.2 Dữ liệu cho chương trình**
 - Biến và hằng
 - Khung của một chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

7



3.1.2 Dữ liệu cho chương trình

- **Dữ liệu:**
 - các số hệ số 2: 0011B
 - hệ số 10: 1234
 - hệ số 16: 1EF1H, 0ABBAH
 - Ký tự, chuỗi ký tự: 'A', "abcd"

8



Chương 3 Lập trình hợp ngữ với 8086

- **3.1 Giới thiệu khung của chương trình hợp ngữ**
 - Cú pháp của chương trình hợp ngữ
 - Dữ liệu cho chương trình
 - 3.1.3 Biến và hằng
 - Khung của một chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể



3.1.3 Biến và hằng

- **DB (Define Byte):** định nghĩa biến kiểu byte
- **DW (Define Word):** định nghĩa biến kiểu từ - 2 byte
- **DD (Define Double word):** định nghĩa biến kiểu từ kép - 4 byte
- **Biến byte:**
 - Tên DB giá trị khởi đầu
 - Ví dụ:
 - ⇒ B1 DB 4
 - ⇒ B1 DB ?
 - ⇒ C1 DB '\$'
 - ⇒ C1 DB 34



3.1.3 Biến và hằng

- **Biến từ:**

- Tên DW `gia_trí_khởi đầu`
- Ví dụ:
 ⇒ W1 DW 4
 ⇒ W2 DW ?

- **Biến mảng:**

- M1 DB 4, 5, 6, 7, 8, 9
- M2 DB 100 DUP(0)
- M3 DB 100 DUP(?)
- M4 DB 4, 3, 2, 2 DUP (1, 2 DUP(5), 6)
- M4 DB 4, 3, 2, 1, 5, 5, 6, 1, 5, 5, 6

1300A	
13009	
13008	9
13007	8
13006	7
13005	6
13004	5
13003	4
13002	
13001	
13000	

M1



3.1.3 Biến và hằng

- **Biến mảng 2 chiều:**

$$\begin{pmatrix} 1 & 6 & 3 \\ 4 & 2 & 5 \end{pmatrix}$$

- M1 DB 1, 6, 3
DB 4, 2, 5
- M2 DB 1, 4
DB 6, 2
DB 3, 5

1300A	
13009	
13008	5
13007	2
13006	4
13005	3
13004	6
13003	1
13002	
13001	
13000	

M1



3.1.3 Biến và hằng

- Biến kiểu xâu ký tự**

- STR1 DB 'string'
- STR2 DB 73h, 74h, 72h, 69h, 6Eh, 67h
- STR3 DB 73h, 74h, 'r', 'i', 6Eh, 67h

- Hằng có tên**

- Có thể khai báo hằng ở trong chương trình
- Thường được khai báo ở đoạn dữ liệu
- Ví dụ:**
 - ⇒ CR EQU 0Dh ; là carriage return
 - ⇒ LF EQU 0Ah ; LF là line feed
 - ⇒ CHAO EQU 'CR Hello'
 - ⇒ MSG DB CHAO, '\$'



Chương 3 Lập trình hợp ngữ với 8086

- 3.1 Giới thiệu khung của chương trình hợp ngữ**

- Cú pháp của chương trình hợp ngữ
- Dữ liệu cho chương trình
- Biến và hằng
- 3.1.4 Khung của một chương trình hợp ngữ

- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC**
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ**
- Một số chương trình cụ thể**



3.1.4 Khung của chương trình hợp ngữ

- Khai báo quy mô sử dụng bộ nhớ**

- .MODEL** Kiểu kích thước bộ nhớ
- Ví dụ:** .Model Small

Kiểu	Mô tả
Tiny (hẹp)	mã lệnh và dữ liệu gói gọn trong một đoạn
Small (nhỏ)	mã lệnh nằm trong 1 đoạn, dữ liệu 1 đoạn
Medium (trung bình)	mã lệnh nằm trong nhiều đoạn, dữ liệu 1 đoạn
Compact (gọn)	mã lệnh nằm trong 1 đoạn, dữ liệu trong nhiều đoạn
Large (lớn)	mã lệnh nằm trong nhiều đoạn, dữ liệu trong nhiều đoạn, không có mảng nào lớn hơn 64 K
Huge (đồ sộ)	mã lệnh nằm trong nhiều đoạn, dữ liệu trong nhiều đoạn, các mảng có thể lớn hơn 64 K

15



3.1.4 Khung của chương trình hợp ngữ

- Khai báo đoạn ngắn xếp**

- .Stack** Kích thước (bytes)
- Ví dụ:**
 - ⇒ .Stack 100 ; khai báo stack có kích thước 100 bytes
- Giá trị ngầm định 1 KB**

- Khai báo đoạn dữ liệu:**

- .Data**
- Khai báo các biến và hằng**

- Khai báo đoạn mã**

- .Code**

16



3.1.4 Khung của chương trình hợp ngữ

- Khung của chương trình hợp ngữ để dịch ra file .EXE

```
.Model Small
.Stack 100
.Data
    ;các định nghĩa cho biến và hằng
.Code
MAIN Proc
    ;khởi đầu cho DS
    MOV AX, @data
    MOV DS, AX
    ;các lệnh của chương trình

    ;trở về DOS dùng hàm 4CH của INT 21H
    MOV AH, 4CH
    INT 21H
MAIN Endp
    ;các chương trình con nếu có
END MAIN
```

17



3.1.4 Khung của chương trình hợp ngữ

- Chương trình Hello.EXE

```
.Model Small
.Stack 100
.Data
    CRLF DB 13,10,'$'
    MSG DB 'Hello! $'

.Code
MAIN Proc
    ;khởi đầu cho DS
    MOV AX, @data
    MOV DS, AX
    ;vẽ dấu dòng mới dùng hàm 9 của INT 21H
    MOV AH,9
    LEA DX, CRLF
    INT 21H
    ;Hiển thị lời chào dùng hàm 9 của INT 21H
    MOV AH,9
    LEA DX, MSG
    INT 21H
    ;vẽ dấu dòng mới dùng hàm 9 của INT 21H
    MOV AH,9
    LEA DX, CRLF
    INT 21H
    ;trở về DOS dùng hàm 4CH của INT 21H
    MOV AH, 4CH
    INT 21H
MAIN Endp
END MAIN
```

18



3.1.4 Khung của chương trình hợp ngữ

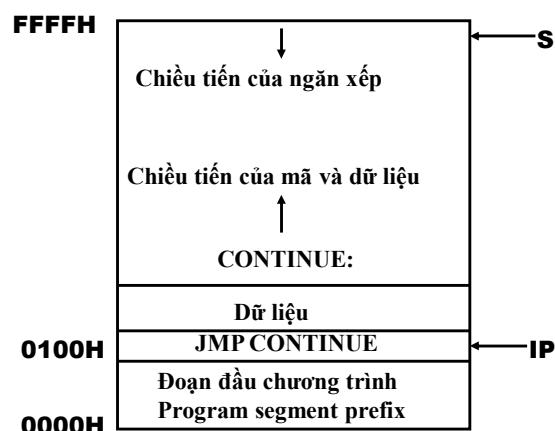
- Khung của chương trình hợp ngữ để dịch ra file .COM

- Chỉ có 1 đoạn cho Code, Data, Stack
- Trở về DOS bằng INT 20H

```
.Model Tiny
.Code
    ORG 100h
START: JMP CONTINUE
; các định nghĩa cho biến và hằng
CONTINUE:
MAIN Proc
; các lệnh của chương trình
INT 20H ; trở về DOS
MAIN Endp
; các chương trình con nếu có
END START
```



3.1.4 Khung của chương trình hợp ngữ





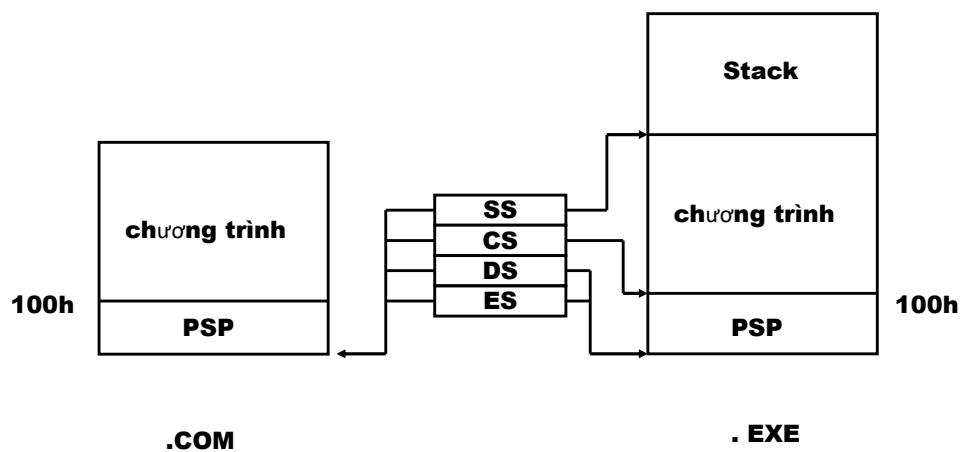
3.1.4 Khung của chương trình hợp ngữ

- Chương trình Hello.COM

```
.Model Tiny
.Code
    ORG 100H
START: JMP CONTINUE
        CRLF DB 13,10,'$'
        MSG  DB 'Hello! $'
CONTINUE:
MAIN Proc
;về đầu dòng mới dùng hàm 9 của INT 21H
    MOV AH,9
    LEA DX, CRLF
    INT 21H
;Hiển thị lời chào dùng hàm 9 của INT 21H
    MOV AH,9
    LEA DX, MSG
    INT 21H
;về đầu dòng mới dùng hàm 9 của INT 21H
    MOV AH,9
    LEA DX, CRLF
    INT 21H
;trở về DOS
    INT 20H
MAIN Endp
END START
```



3.1.4 Khung của chương trình hợp ngữ





Chương 3 Lập trình hợp ngữ với 8086

Giới thiệu khung của chương trình hợp ngữ

3.2 Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC

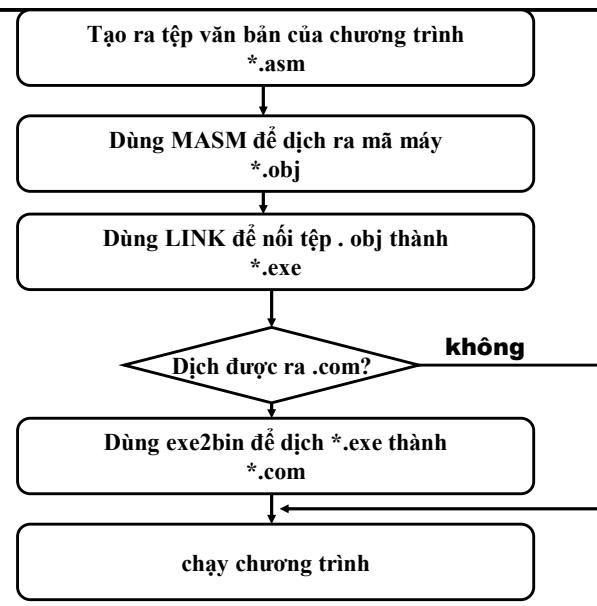
Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ

Một số chương trình cụ thể

23



3.2 Cách tạo một chương trình hợp ngữ



24



Chương 3 Lập trình hợp ngữ với 8086

- 3.1 Giới thiệu khung của chương trình hợp ngữ
- 3.2 Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- 3.3 Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
 - 3.3.1 Cấu trúc lựa chọn
 - 3.3.2 Cấu trúc lặp
- 3.4 Một số chương trình cụ thể



3.3.1 Cấu trúc lựa chọn If-then

- If điều_kiện then công_việc
- Ví dụ: Gán cho BX giá trị tuyệt đối của AX

```

; If AX<0
CMP AX, 0 ; AX<0 ?
JNL End_if ; không, thoát ra
; then
NEG AX ; đúng, đảo dấu
End_if: MOV BX, AX ;gán

```



3.3.1 Cấu trúc lựa chọn If-then-else

- If điều_kiện then công_việc1 else công_việc2
- Ví dụ: if AX<BX then CX=0 else CX=1

```

; if AX<BX
CMP    AX, BX      ; AX<BX ?
JL     Then_ ; đúng, CX=0
;else
MOV    CX, 1   ; sai, CX=1
JMP    End_if
Then_ : MOV  CX, 0;
End_if:

```



3.3.1 Cấu trúc lựa chọn Case

- Case Biểu thức
 - Giá trị 1: công việc 1
 - Giá trị 2: công việc 2
 - ...
 - Giá trị N: công việc N
- END CASE
- Ví dụ:
 - Nếu AX<0 thì CX=-1
 - Nếu AX=0 thì CX=0
 - Nếu AX>0 thì CX=1

```

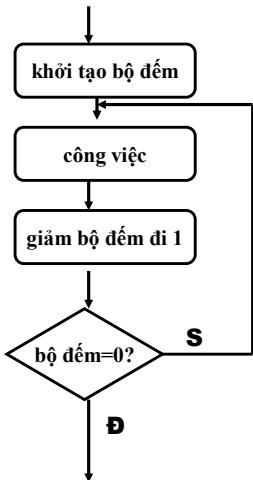
        CMP    AX, 0  ;
        JL     AM   ; AX<0
        JE     Khong ; AX=0
        JG     DUONG; AX>1
AM:  MOV  CX, -1
      JMP  End_case
Khong:  MOV  CX, 0
      JMP  End_case

DUONG:  MOV CX, 1
      End_case:

```

3.3.2 Cấu trúc lặp For-Do

- For số lần lặp Do công việc



ví dụ: Hiển thị một dòng ký tự \$ trên màn hình

```

MOV CX, 80      ;số lần lặp
MOV AH,2        ;hàm hiển thị
MOV DL,'$';DL chứa ký tự cần hiển thị
HIEN: INT 21H   ; Hiển thị
LOOP HIEN
;End_for
  
```

3.3.2 Cấu trúc lặp While-Do

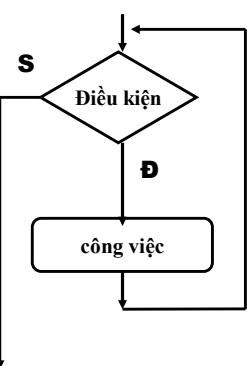
- While điều kiện Do công việc

ví dụ:

Khởi tạo AX=0, BX=0

Trong khi AX<>10 thì

 BX=BX+1 và AX=AX+2



XOR AX, AX ;AX=0
XOR BX, BX ;BX=0
TIEP:

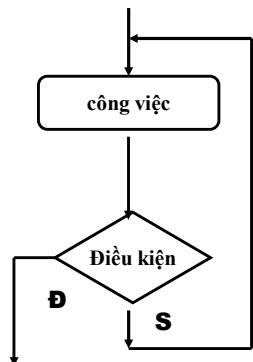
```

CMP AX,10      ;so sánh AX với 10
JE End_while   ;kết thúc nếu AX=10
INC BX         ;BX=BX+1
ADD AX,2       ;AX=AX+2
jmp tiep
  
```

End_while:

3.3.2 Cấu trúc lặp Repeat-until

- Repeat công việc until điều kiện



ví dụ: đọc từ bàn phím cho tới khi gặp ký tự CR thì thôi

MOV AH,1 TIEP: INT 21H CMP AL, 13 JNE TIEP End_:	; hàm đọc ký tự từ bàn phím ; đọc một ký tự vào AL ; đọc CR? ; chưa, đọc tiếp
--	--

Chương 3 Lập trình hợp ngữ với 8086

- 3.1 Giới thiệu khung của chương trình hợp ngữ
- 3.2 Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- 3.3 Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- 3.4 Một số chương trình cụ thể



3.4.1 Xuất nhập dữ liệu

- **2 cách:**
 - Dùng lệnh IN, OUT để trao đổi với các thiết bị ngoại vi**
 - ⇒ phức tạp vì phải biết địa chỉ cổng ghép nối thiết bị
 - ⇒ Các hệ thống khác nhau có địa chỉ khác nhau
 - Dùng các chương trình con phục vụ ngắt của DOS và BIOS**
 - ⇒ đơn giản, dễ sử dụng
 - ⇒ không phụ thuộc vào hệ thống
- **Ngắt 21h của DOS:**
 - Hàm 1: đọc 1 ký tự từ bàn phím**
 - ⇒ Vào: AH=1
 - ⇒ Ra: AL=mã ASCII của ký tự, AL=0 khi ký tự là phím chức năng
 - Hàm 2: hiện 1 ký tự lên màn hình**
 - ⇒ Vào: AH=2
 - DL=mã ASCII của ký tự cần hiển thị
 - Hàm 9: hiện chuỗi ký tự với \$ ở cuối lên màn hình**
 - ⇒ Vào: AH=9
 - DX=địa chỉ lệch của chuỗi ký tự cần hiển thị
 - Hàm 4CH: kết thúc chương trình loại .exe**
 - ⇒ Vào: AH=4CH

33



3.4.2 Một số chương trình cụ thể

- **Ví dụ 1: Lập chương trình yêu cầu người sử dụng gõ vào một chữ cái thường và hiển thị dạng chữ hoa của chữ cái đó lên màn hình**
 - Ví dụ:**
 - ⇒ Hay nhap vao mot chu cai thuong: a
 - ⇒ A
- **Ví dụ 2: Đọc từ bàn phím một số hệ hai (dài nhất là 16 bit), kết quả đọc được để tại thanh ghi BX. Sau đó hiện nội dung thanh ghi BX ra màn hình.**
- **Ví dụ 3: Nhập một dãy số 8 bit ở dạng thập phân, các số cách nhau bằng 1 dấu cách và kết thúc bằng phím Enter. Sắp xếp dãy số theo thứ tự tăng dần và in dãy số đã sắp xếp ra màn hình.**

34



3.4.2 Một số chương trình cụ thể

- Ví dụ 4:** Viết chương trình cho phép nhập vào kích thước $M \times N$ và các phần tử của một mảng 2 chiều gồm các số thập phân 8 bit.

- ⇒ Tìm số lớn nhất và nhỏ nhất của mảng, in ra màn hình
- ⇒ Tính tổng các phần tử của mảng và in ra màn hình
- ⇒ Chuyển thành mảng $N \times M$ và in mảng mới ra màn hình

□ Giải:

```

Hãy nhập giá trị M=
Hãy nhập giá trị N=
Nhập phần tử [1,1]=
Nhập phần tử [1,2]
.....
Số lớn nhất là phần tử [3,4]=15
Số nhỏ nhất là phần tử [1,2]=2
Tổng=256
...

```



Nội dung môn học

1. Giới thiệu chung về hệ vi xử lý
2. Bộ vi xử lý Intel 8088/8086
3. Lập trình hợp ngữ cho 8086
4. Tổ chức vào ra dữ liệu
5. Ngắt và xử lý ngắt
6. Truy cập bộ nhớ trực tiếp DMA
7. Các bộ vi xử lý trên thực tế

1



Chương 4: Tổ chức vào ra dữ liệu

- 4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
- 4.2 Ghép nối 8088 với bộ nhớ
- 4.3 Ghép nối 8086 với bộ nhớ
- 4.4 Ghép nối với thiết bị ngoại vi

2

1



Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.1.1 Các tín hiệu của 8086

4.1.2 Phân kênh và việc đệm cho các bus

4.1.3 Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288

4.1.4 Biểu đồ thời gian của các lệnh ghi/đọc

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi



Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.1.1 Các tín hiệu của 8086

4.1.2 Phân kênh và việc đệm cho các bus

4.1.3 Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288

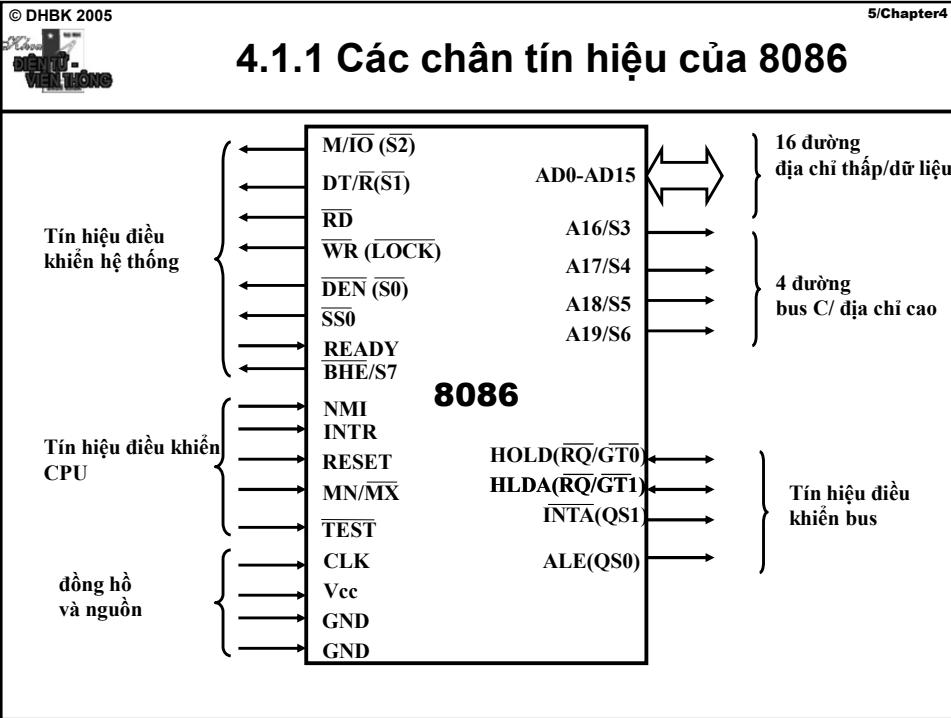
4.1.4 Biểu đồ thời gian của các lệnh ghi/đọc

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.1.1 Các chân tín hiệu của 8086



5

4.1.1 Các chân tín hiệu của 8086

- AD0-AD15:**
 - ☐ ALE =1: 16 chân địa chỉ cho bộ nhớ hoặc I/O
 - ☐ ALE=0: 16 đường dữ liệu
- A19/S6-A16/S3**
 - ☐ 4 bit địa chỉ cao
 - ☐ 4 bit trạng thái:
 - ⇒ S6 luôn bằng 1
 - ⇒ S5: trạng thái của IF
 - ⇒ S4, S3: bit trạng thái về thanh ghi đoạn đang truy cập
- INTR: interrupt request**
 - ☐ IF=1 và INTR=1 => xảy ra ngắt
- TEST**
 - ☐ nếu =0, CPU ở trạng thái đợi và thực hiện lệnh NOP
 - ☐ =1, lệnh WAIT đợi đến khi TEST=0

S4	S3	
0	0	ES
0	1	SS
1	0	CS or No
1	1	DS

6

3

4.1.1 Các chân tín hiệu của 8086

- **NMI (Non-maskable interrupt)**
 - NMI=1 => thực hiện INT 2
- **RESET**
 - 1: khởi động lại hệ thống và thực hiện lệnh tại ô nhớ FFFF0H
- **MN/MX**
 - 1: chế độ min
 - 0: chế độ max
- **BHE/S7:**
 - 0: cho phép truy cập byte cao dữ liệu
 - Trạng thái S7 luôn bằng 1
- **RD**
 - 0: CPU đọc dữ liệu từ bộ nhớ hoặc thiết bị ngoại vi
- **Các chân ở chế độ min**
 - M/I/O**
 - ⇒ 1: truy cập bộ nhớ
 - ⇒ 0: truy cập thiết bị ngoại vi I/O
 - WR**
 - ⇒ 0: dữ liệu hợp lệ tại bus dữ liệu để đưa ra bộ nhớ hoặc thiết bị ngoại vi

7

4.1.1 Các chân tín hiệu của 8086

- **Các chân ở chế độ min**
 - INTA**: interrupt acknowledge
 - ⇒ 0: khi INTR=1 và IF=1
 - ALE**: address latch enable
 - DT/R**: data transmit/receive
 - ⇒ 1: bus dữ liệu đang truyền dữ liệu đi
 - ⇒ 0: bus dữ liệu đang nhận dữ liệu
 - DEN**: Data enable
 - ⇒ 0: kích hoạt đệm dữ liệu ngoài
 - HOLD**
 - ⇒ 1: CPU tạm dừng hoạt động để nhường quyền điều khiển cho DMA, các bus được đặt ở trạng thái trống cao
 - HLDA (Hold Acknowledge)**
 - ⇒ khi HOLD=1, HLDA=1

8

4

4.1.1 Các chân tín hiệu của 8086

- Các chân ở chế độ Max**

- S₂, S₁, S₀**

⇒ ghép nối với điều khiển bus 8288

S ₂	S ₁	S ₀	chu kỳ điều khiển của bus
0	0	0	chấp nhận yêu cầu ngắn
0	0	1	đọc thiết bị ngoại vi
0	1	0	Ghi thiết bị ngoại vi
0	1	1	Dừng
1	0	0	đọc mã lệnh
1	0	1	đọc bộ nhớ
1	1	0	ghi bộ nhớ
1	1	1	bus rỗi

4.1.1 Các chân tín hiệu của 8086

- Các chân ở chế độ Max**

- RQ/GT0 và RQ/GT1: Request/Grant**

⇒ Tín hiệu yêu cầu dùng bus của các bộ vi xử lý khác/chấp nhận treo bus của CPU
⇒ GT0 có mức ưu tiên cao hơn GT1

- LOCK**

⇒ 0: cấm các bộ vi xử lý khác dùng bus

- QS0 và QS1:**

⇒ trạng thái của hàng đợi lệnh

RDY: chân tín hiệu báo rằng bộ nhớ or cổng vào/ra sẵn sàng trao đổi dữ liệu với vxl
RDY = 1: trao đổi bth, RDY = 0: vxl chèm thêm các trạng thái đợi

QS1	QS0	Trạng thái hàng đợi lệnh
0	0	không hoạt động
0	1	đọc byte mã lệnh đầu tiên
1	0	hàng đợi rỗng
1	1	đọc byte tiếp theo

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.1.1 Các tín hiệu của 8086

4.1.2 Phân kênh và việc đệm cho các bus

4.1.3 Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288

4.1.4 Biểu đồ thời gian của các lệnh ghi/đọc

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

11

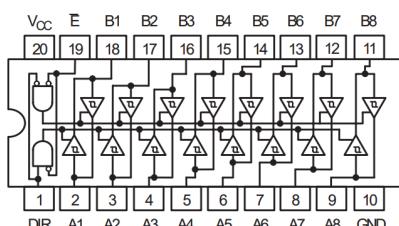
4.1.2 Phân kênh và đệm cho các bus

- Vì sao phải phân kênh và khuyếch đại đệm:**

- Các bus địa chỉ và dữ liệu dùng chung chân
- Nâng cao khả năng tải của bus

- Các vi mạch phân kênh và đệm:**

- 74LS373: phân kênh
- 74LS245: đệm dữ liệu 2 chiều
- 74LS244: đệm 3 trạng thái theo 1 chiều



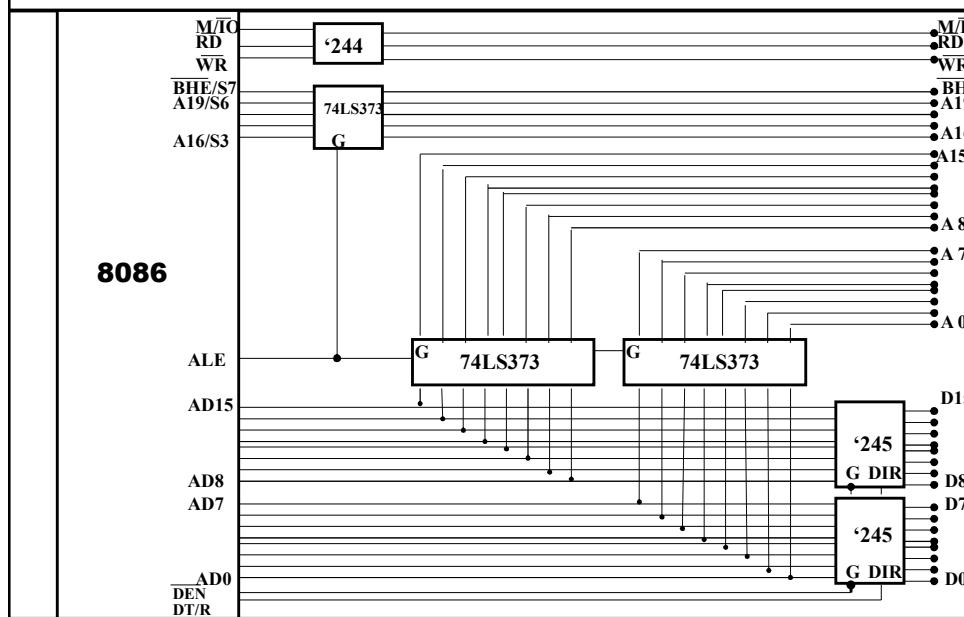
DIR: Direction

74LS245 pin-out

12

6

4.1.2 Phân kênh và đệm cho các bus



13

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.1.1 Các tín hiệu của 8086

4.1.2 Phân kênh và việc đệm cho các bus

4.1.3 Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288

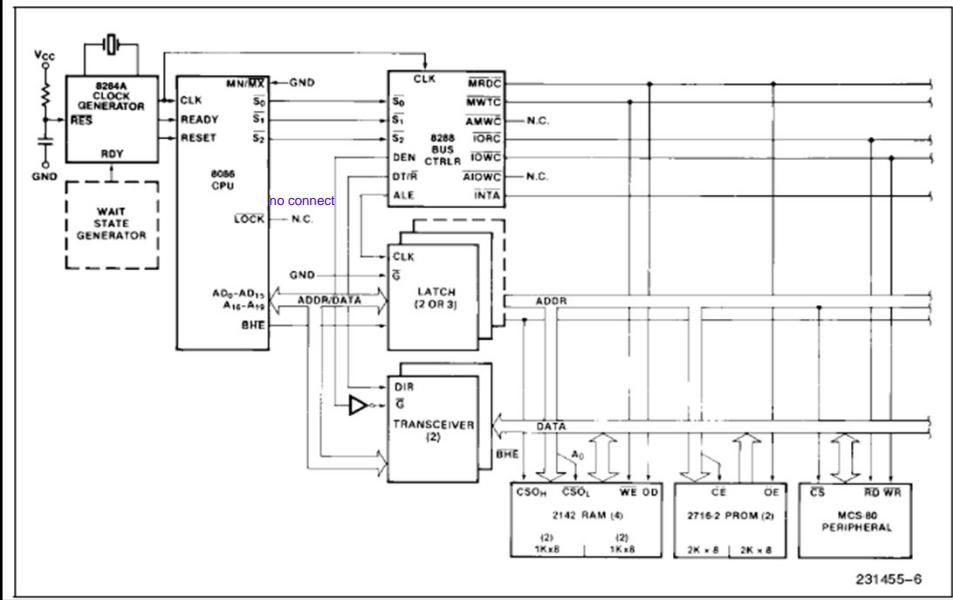
4.1.4 Biểu đồ thời gian của các lệnh ghi/đọc

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.1.3 Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288



15

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.1.1 Các tín hiệu của 8086

4.1.2 Phân kênh và việc đệm cho các bus

4.1.3 Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288

4.1.4 Biểu đồ thời gian của các lệnh ghi/đọc

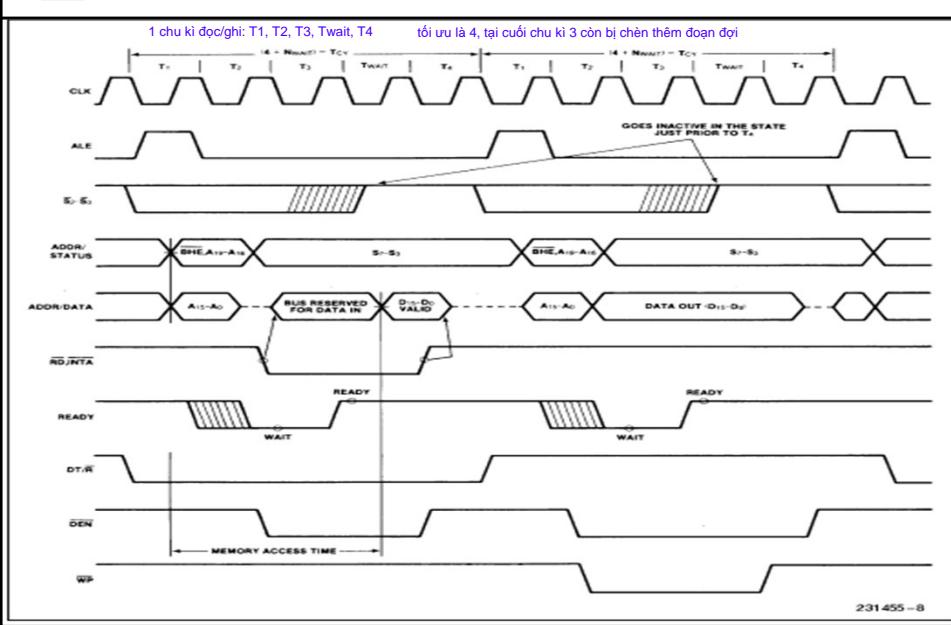
4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

16

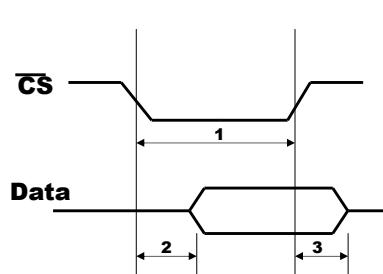
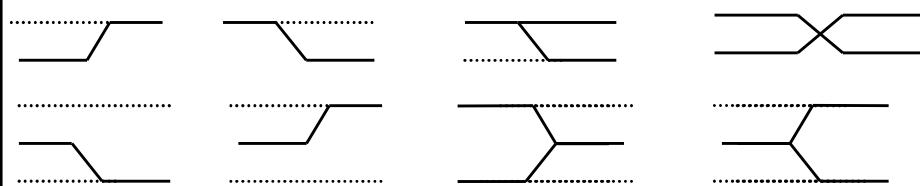
4.1.4 Biểu đồ thời gian



17

4.1.4 Biểu đồ thời gian

- Các ký hiệu trong biểu đồ thời gian:



		Min	max	Units
1	CS hold time	60		ns
2	CS to data valid		30	ns
3	Data hold time	5	10	ns

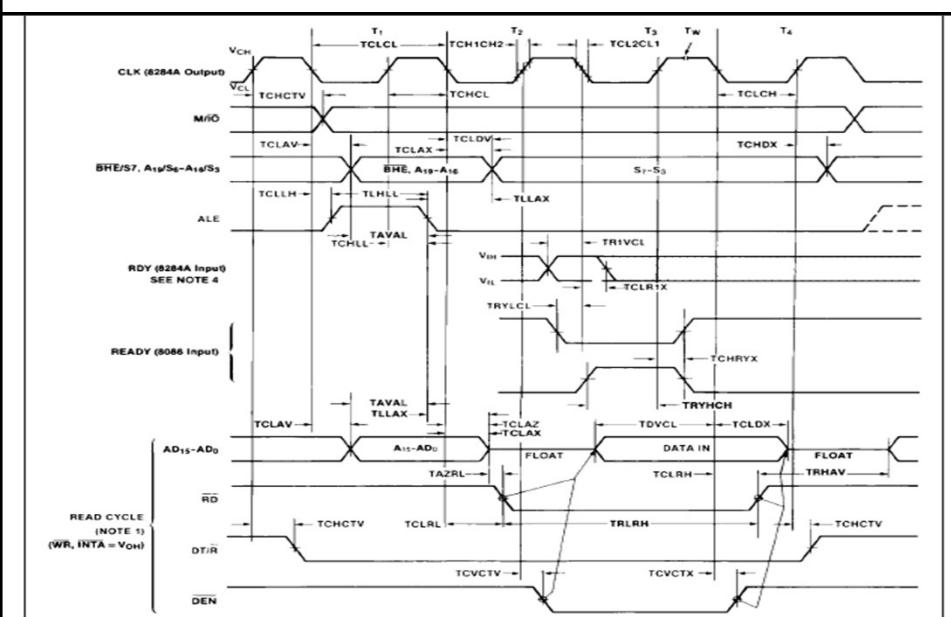
18

4.1.4 Biểu đồ thời gian

- Một chu kỳ ghi/đọc của CPU (chu kỳ bus): 4 chu kỳ xung nhịp T
 - 5 MHz: $4 \times 200 \text{ ns} = 800 \text{ ns}$
 - T1:
 - ⇒ CPU đưa ra địa chỉ của bộ nhớ hoặc I/O, DT/R, M/I/O, ALE
 - T2:
 - ⇒ CPU đưa ra RD hoặc WR, DEN và dữ liệu trên D0-D15 nếu là lệnh ghi
 - ⇒ CPU đọc tín hiệu READY tại cuối chu kỳ của T2 để xử lý trong chu kỳ tiếp theo khi nó làm việc với bộ nhớ hay I/O chậm
 - T3:
 - ⇒ Nếu READY=0 => T3 trở thành chu kỳ đợi: Tw=n*T
 - ⇒ Tại cuối T3, CPU sẽ đọc dữ liệu nếu là lệnh đọc dữ liệu
 - T4:
 - ⇒ Các tín hiệu trên bus được giải phóng
 - ⇒ WR chuyển từ 0 lên 1 kích hoạt quá trình ghi của bộ nhớ

19

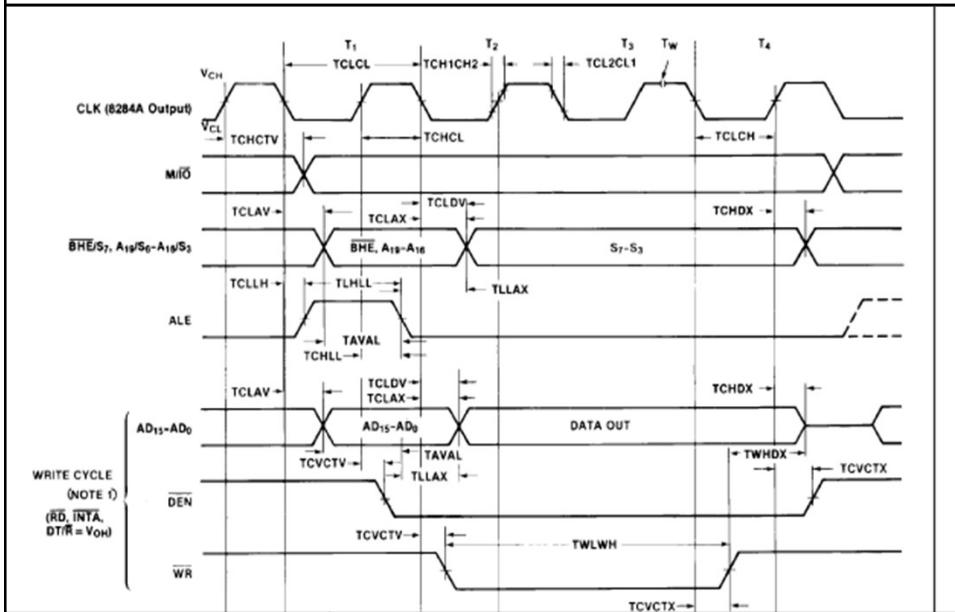
4.1.4 Biểu đồ thời gian



20

10

4.1.4 Biểu đồ thời gian



21

Chương 4: Tổ chức vào ra dữ liệu

- 4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288**
 - 4.2 Ghép nối 8088 với bộ nhớ**
 - 4.2.1 Các loại bộ nhớ bán dẫn
 - 4.2.2 Giải mã địa chỉ cho bộ nhớ
 - 4.2.3 Ghép nối 8088 với bộ nhớ
 - 4.3 Ghép nối 8086 với bộ nhớ**
 - 4.4 Ghép nối với thiết bị ngoại vi**



Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

 4.2.1 Các loại bộ nhớ bán dẫn

 4.2.2 Giải mã địa chỉ cho bộ nhớ

 4.2.3 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi



4.2.1 Các loại bộ nhớ bán dẫn

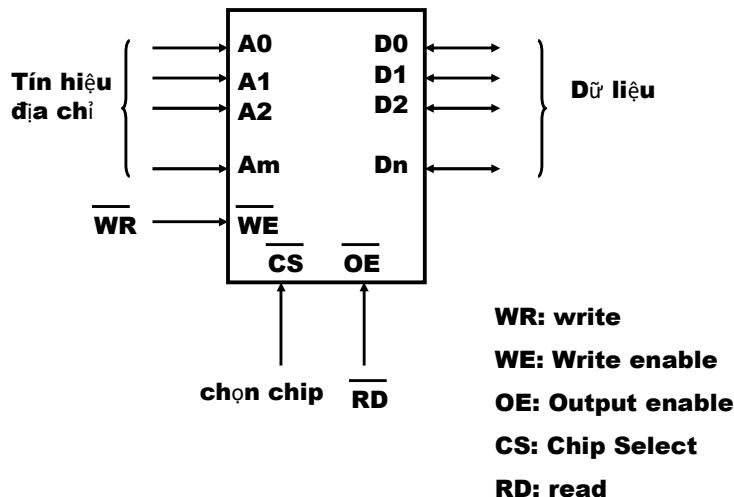
- **Bộ nhớ không bị mất dữ liệu (non-volatile)**

- ROM (Read Only Memory)
- PROM (Programmable ROM)
- EPROM (Erasable programmable ROM)
- Flash
- EEPROM (Electrically Erasable Programmable ROM)
- FeRAM (Ferroelectric Random Access Memory)
- MRAM (Magnetoelectronic Random Access Memory)

- **Bộ nhớ bị mất dữ liệu (volatile)**

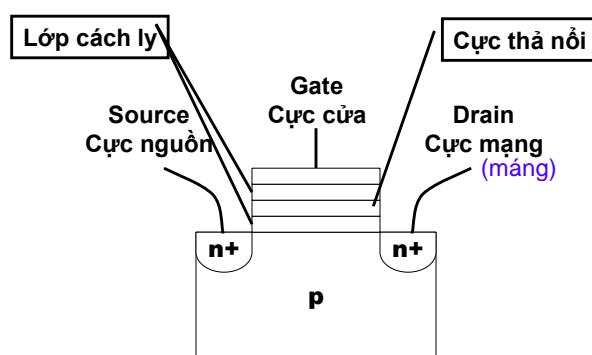
- SRAM (Static RAM)
- SBSRAM (Synchronous Burst RAM)
- DRAM (Dynamic RAM)
- FPDGRAM (Fast Page mode Dynamic RAM)
- EDO DRAM (Extended Data Out Dynamic RAM)
- SDRAM (Synchronous Dynamic RAM)
- DDR-SDRAM (Double Data Rate SDRAM)
- RDRAM (Rambus Dynamic RAM)

4.2.1 Các loại bộ nhớ bán dẫn



25

EPROM

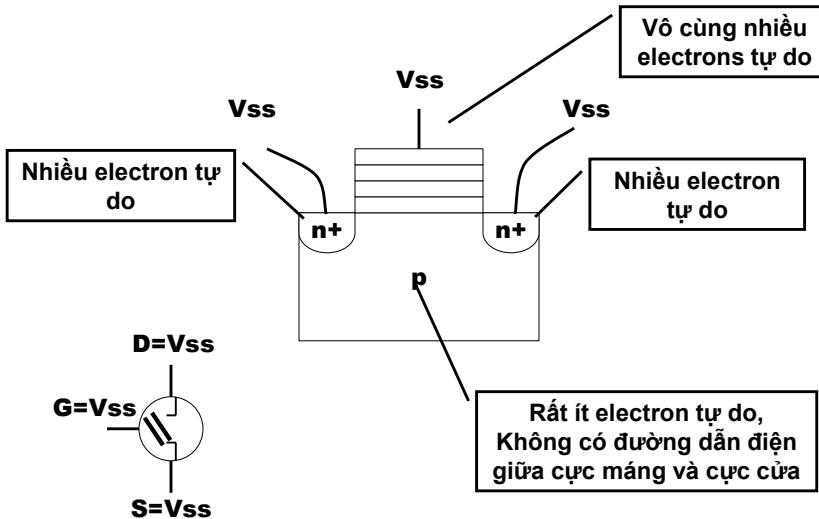


26

13

EPROM

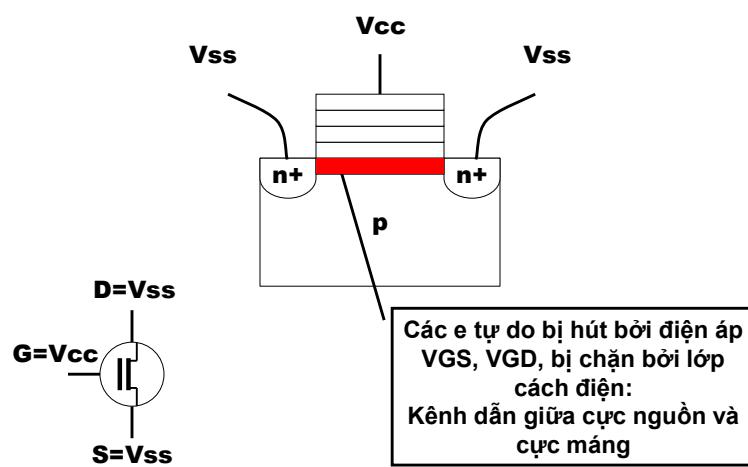
Khi không có e tự do ở cực thà nỗi



27

EPROM

Khi không có e tự do ở cực thà nỗi

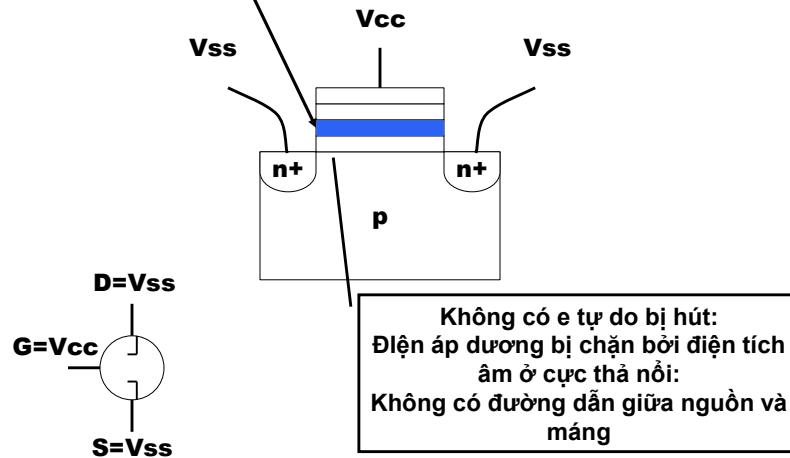


28

14

EPROM

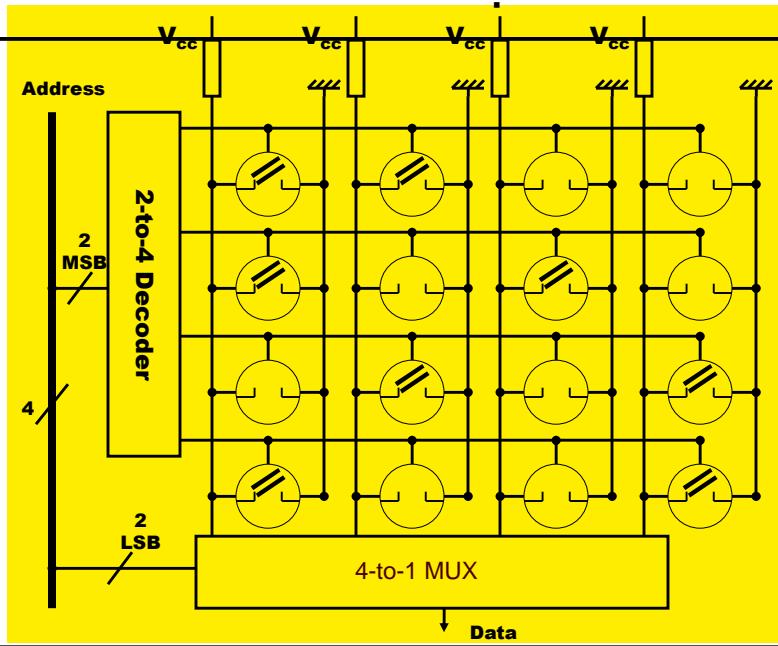
Nhiều electron tự do bị nhốt tại cực thỏi nỗi



29

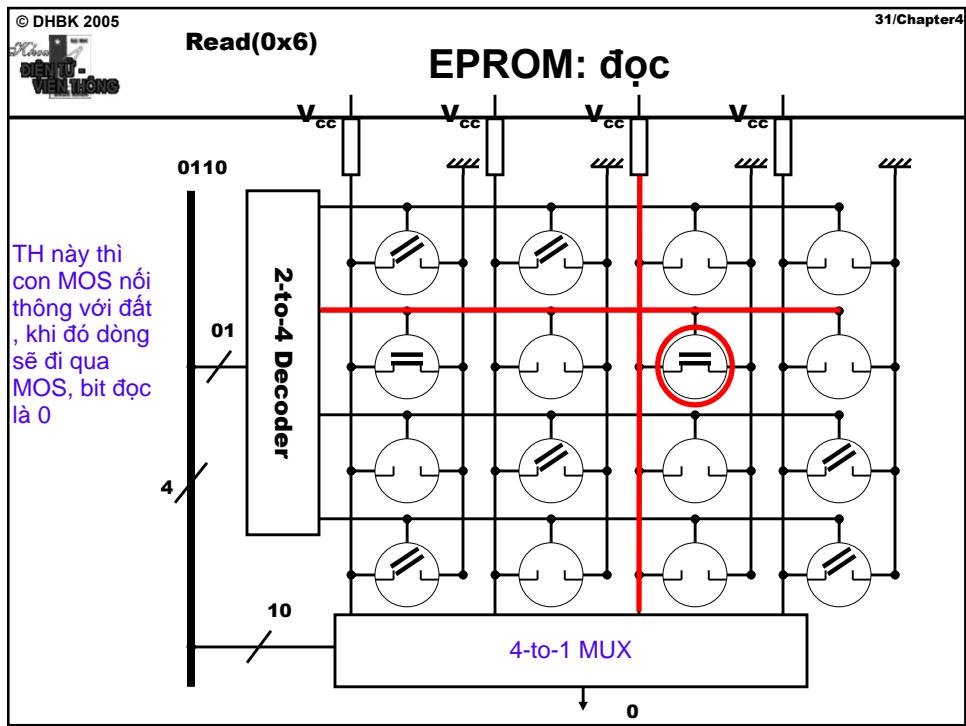
EPROM: đọc

Chia 4 bit
địa chỉ
thành 2bit
MSB và 2
bit LSB

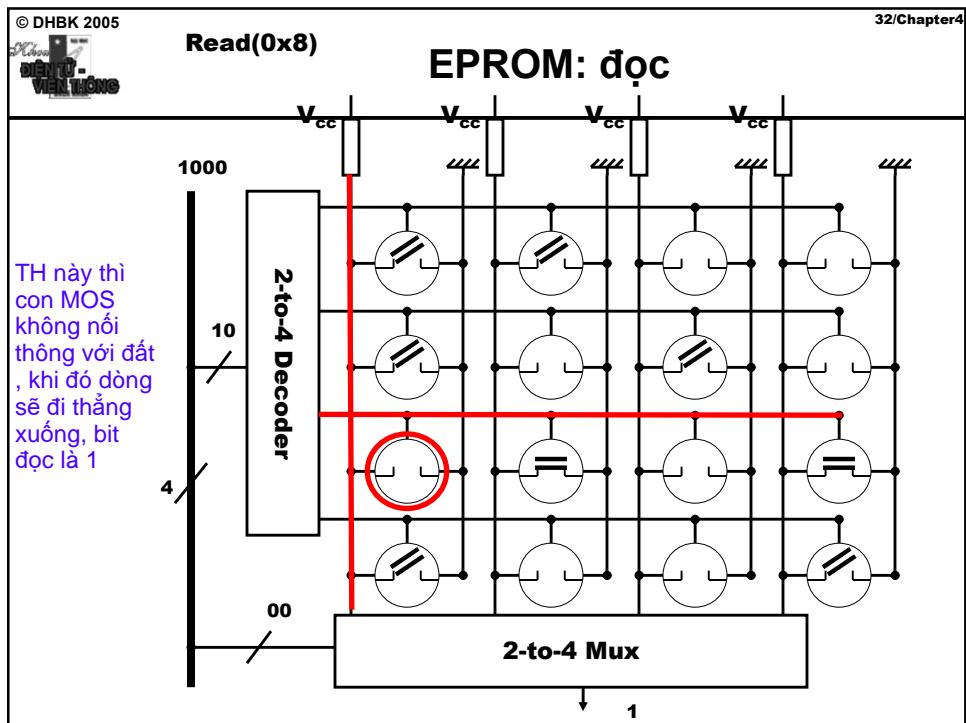


30

15

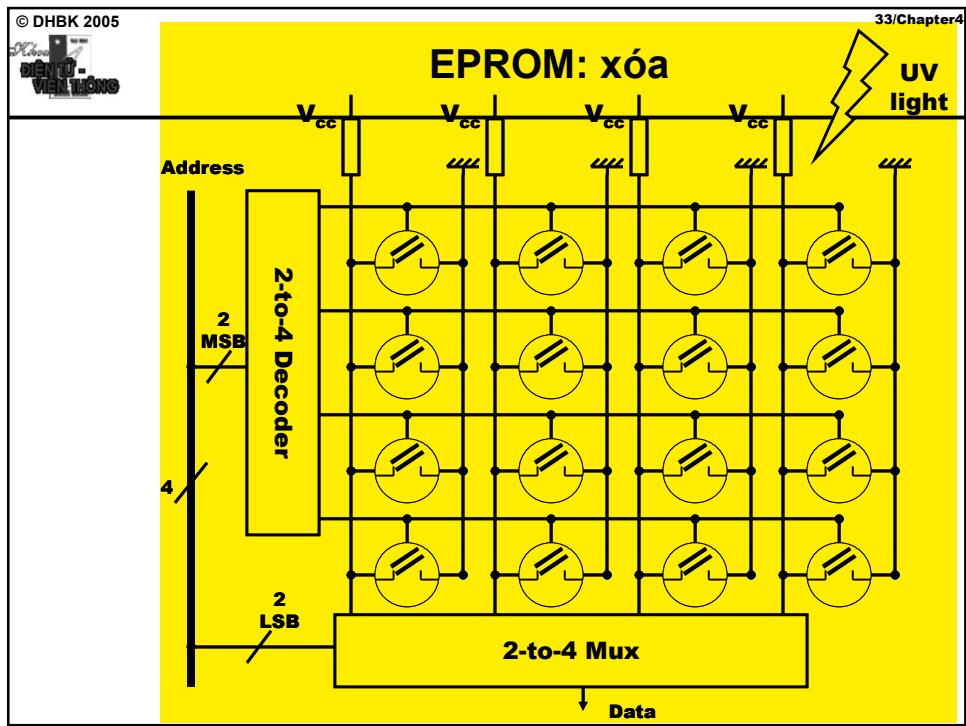


31

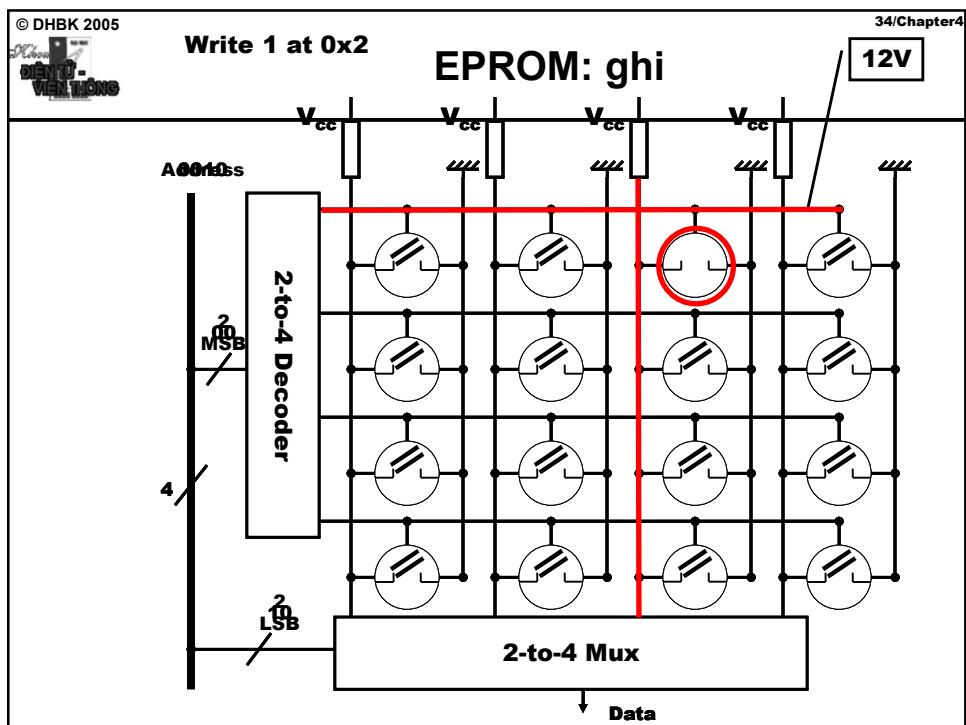


32

16



33



34

17

EPROM

- **Ghi vào EPROM**

- Dùng mạch nạp với điện áp 12 V
- 1 ms một bit

1ms/bit (CHẬM) dung lượng
lớn dẫn đến khả năng ghi lâu
(1MB = 2,33 giờ)

- **Xoá EPROM**

- 20 phút dưới tia tử ngoại
- Số lần ghi 3 lần

- **Đọc EPROM**

- 100 ns đọc nhanh hơn ghi

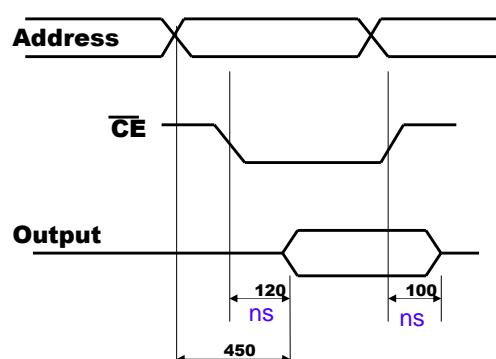
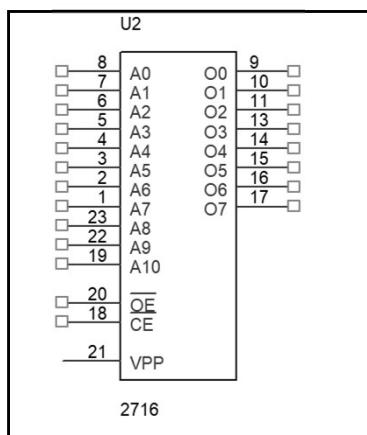
- **EPROM họ 27xxx**

- 2708 (1K*8), 2716 (2K*8), 2732 (4K*8), 2764 (8K*8)
- 27128 (16K*8), 27256 (32K*8), 27512 (64K*8)

35

EPROM

- **Ví dụ: 2716 EPROM**



36

18

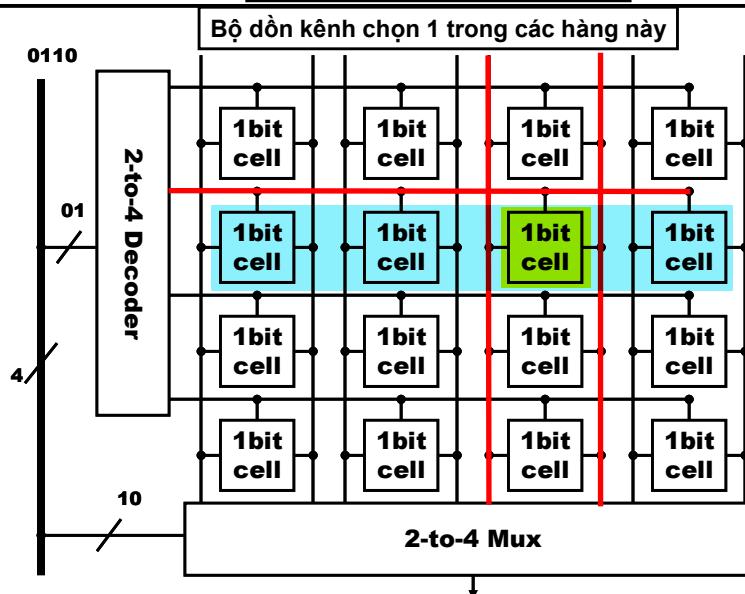
So sánh các loại ROM

Loại ROM	Thời gian ghi	Thời gian đọc	số lần ghi	Kích thước
ROM	NA	35 ns	0	Mbits
PROM	$1\mu\text{s}/\text{bit}$	35 ns	1	128 Kbits
EPROM	$1\text{ms}/\text{bit}$	45 ns	3	16 Mbits
Flash	$1\mu\text{s}/2\text{ KB}$	35 ns	1 triệu	GBits
EEPROM	10 ms/page	200 ns	10000	Mbit
FeRAM	60 ns	50 ns	1000 tì	32 Mbits
MRAM	5ns	5ns	10^{15}	4 Mbits

37

SRAM

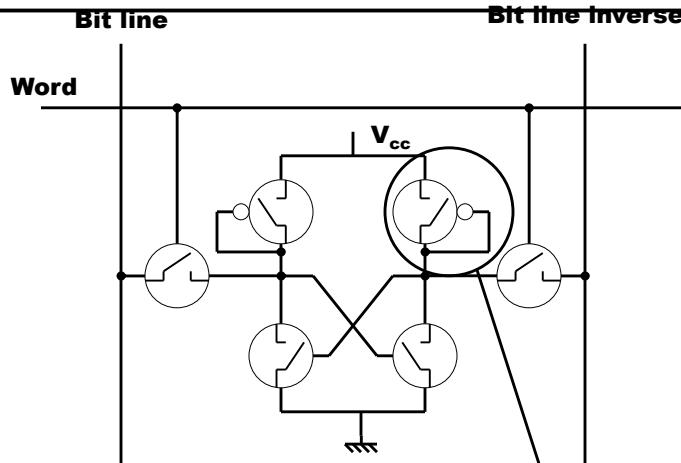
Bộ giải mã: 1 hàng được đọc ra



38

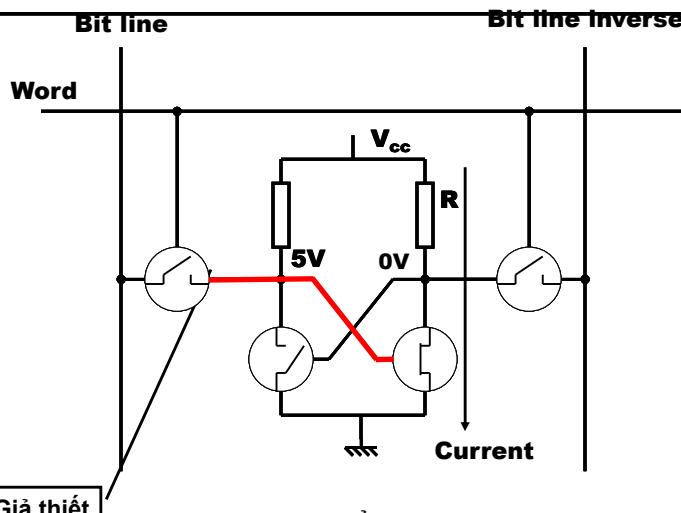
19

SRAM bit cell



39

SRAM bit cell



40

20

SRAM bit cell**Bit line****Bit line inverse****Word****Current****Giả thiết**

Trạng thái ổn định; lưu trữ bit '0'

Liên tục tiêu thụ điện

41

SRAM bit cell**Bit line****Bit line inverse****Word**

0

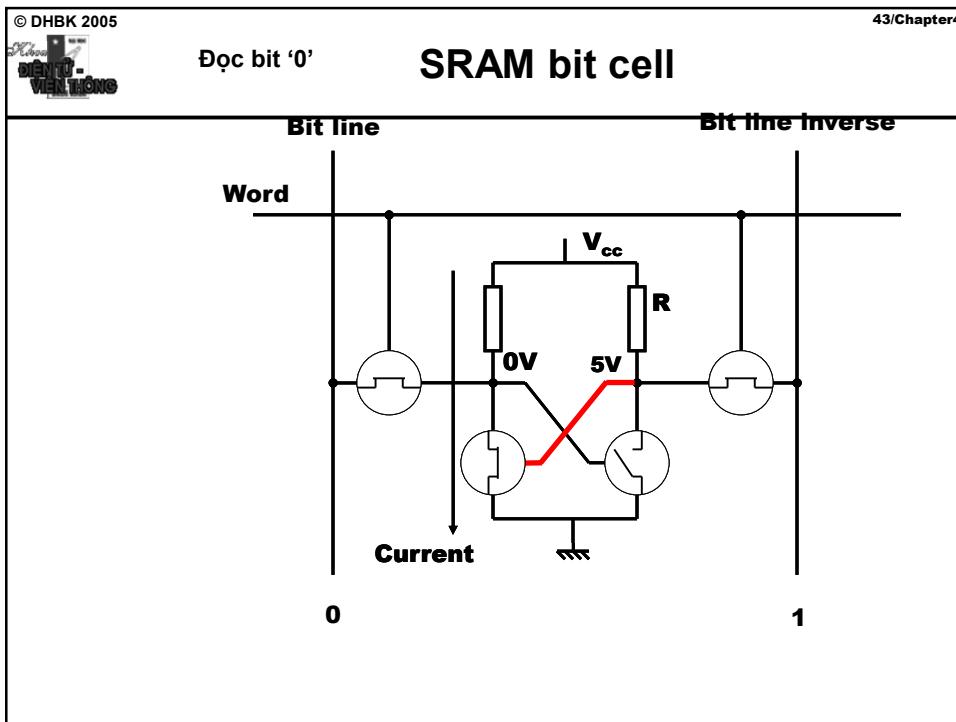
1

0

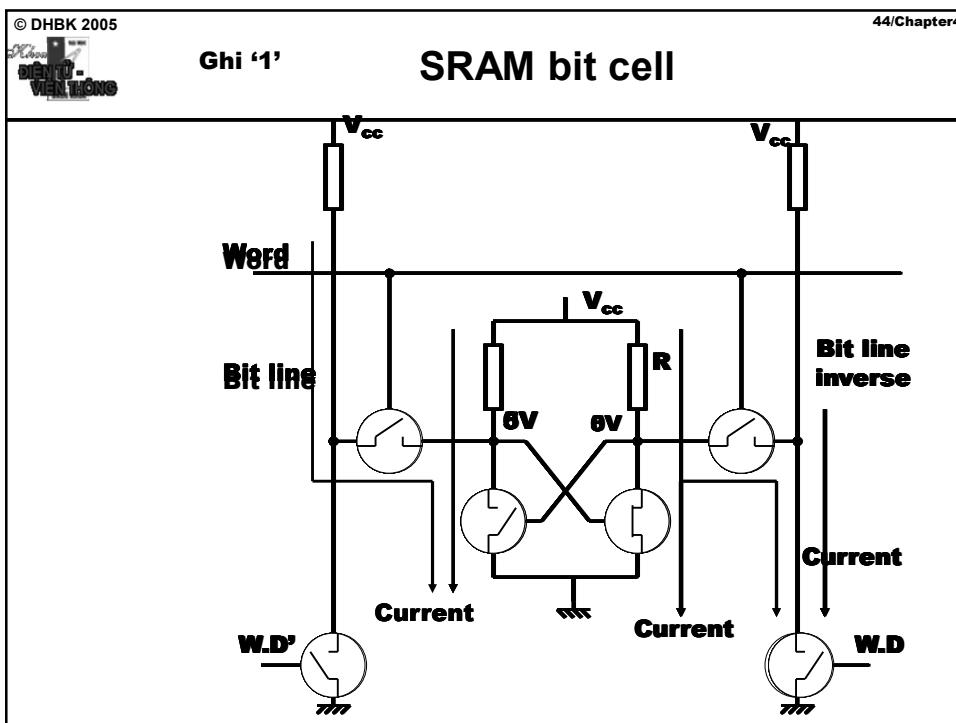
1

42

21



43



44

SRAM

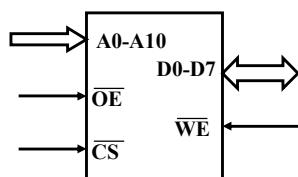
- **Đặc điểm:**

- 6 transistors 1 bit: cao!
- Bị mất dữ liệu khi mất nguồn
- nhanh: thời gian đọc và ghi 5 ns
- Liên tục tiêu thụ năng lượng
- Kích thước: 16 Mbit

- **Ứng dụng:**

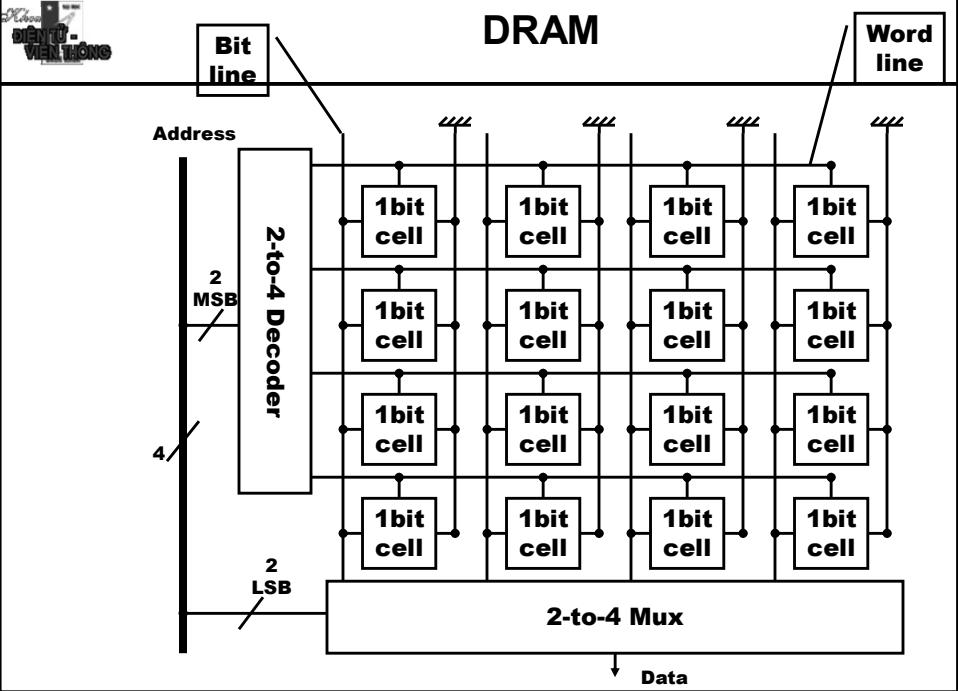
- Bộ nhớ nhỏ và nhanh (cache)
- Không dùng cho các thiết bị chạy pin

- **Ví dụ: 4016 (2K*8), 250 ns, 6264(8Kx8), 62128(16Kx8)**



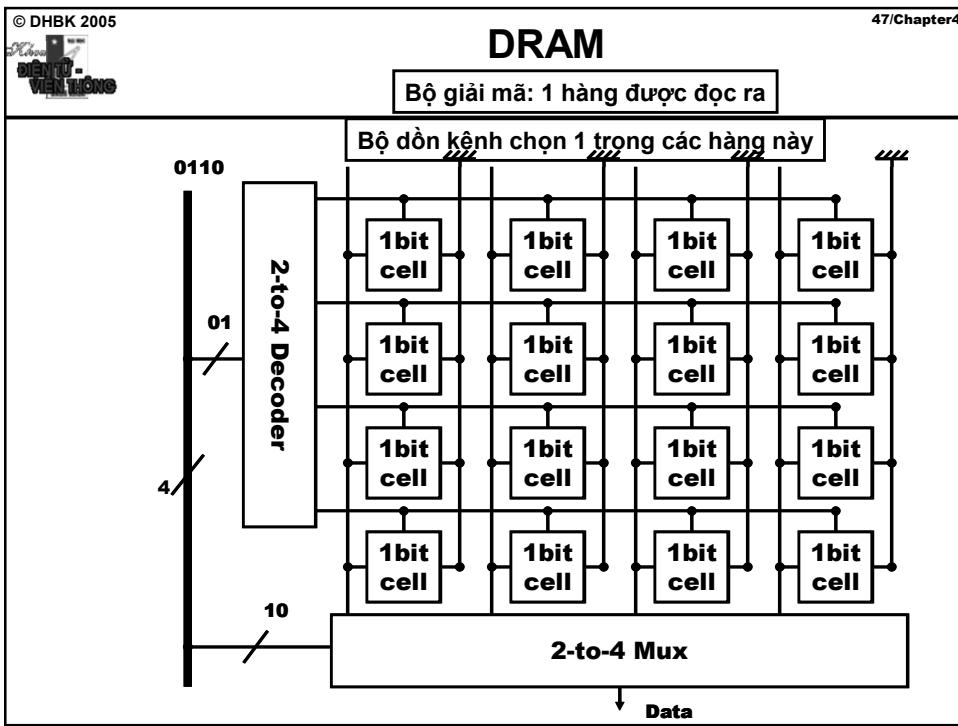
45

DRAM

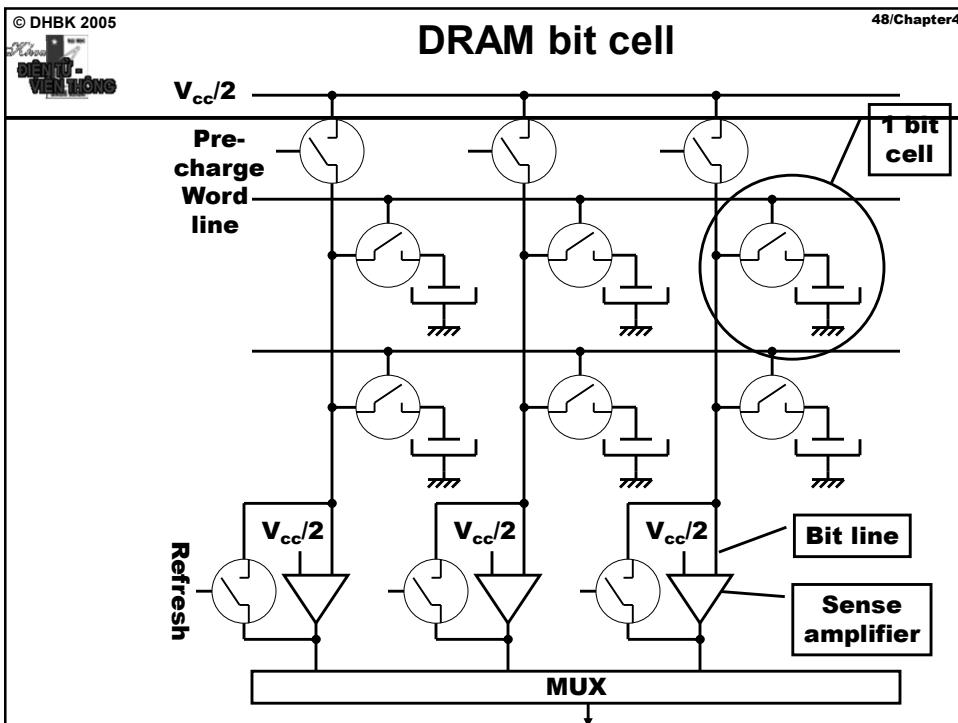


46

23

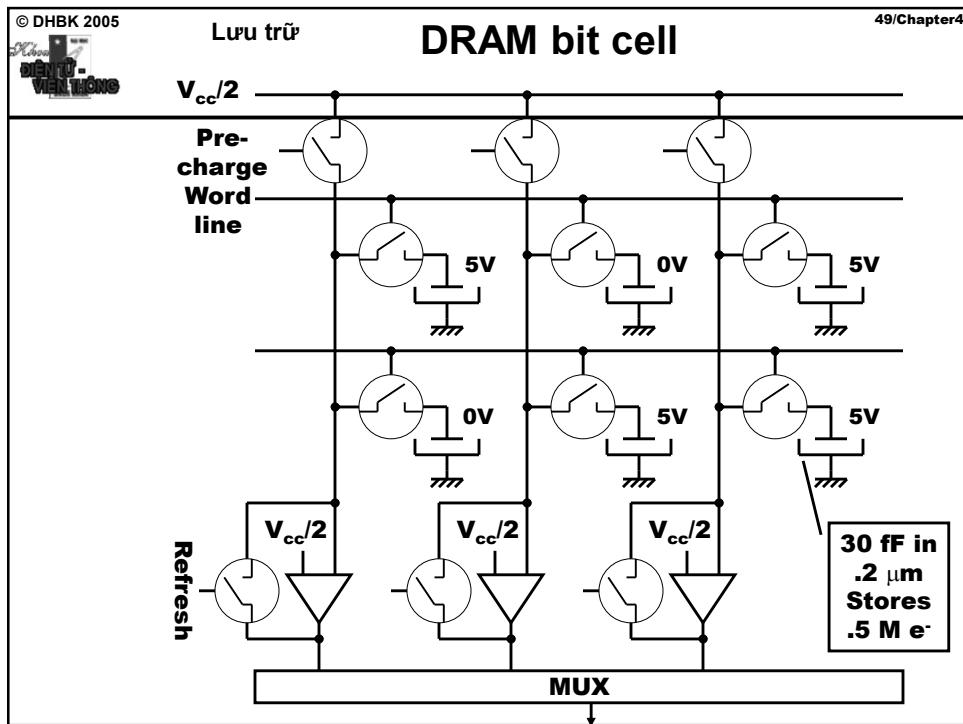


47

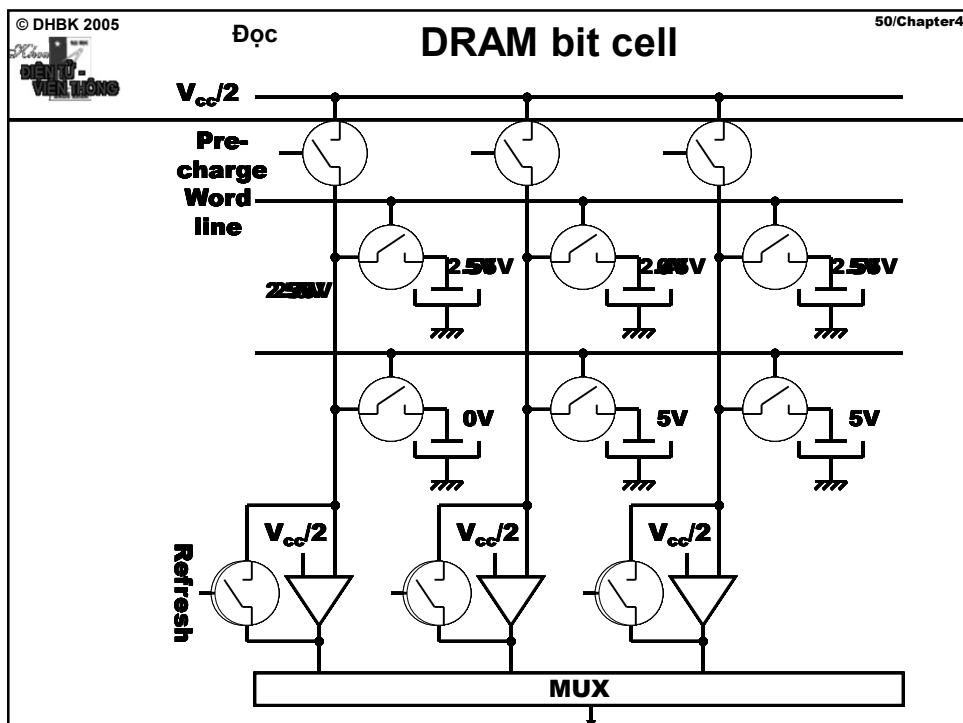


48

24



49



50

25

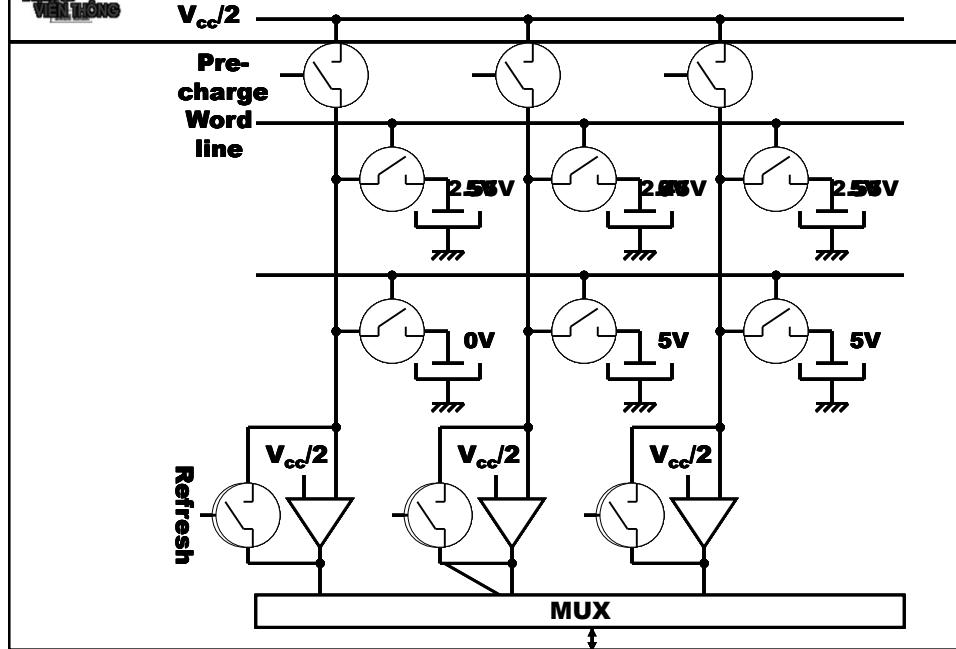
DRAM bit cell

- Chu kỳ đọc

- 1. Precharge
- 2. RAS (Row Address Select): đọc tất cả các bit trong hàng được chọn. Việc đọc này làm giá trị điện áp trên tụ điện bị thay đổi
- 3. Khuếch đại tín hiệu trên các cột tương ứng
- 4.a CAS (Column Address Select): chọn 1 cột và đưa dữ liệu ra ngoài
- 4.b Refresh: khôi phục lại dữ liệu ban đầu của hàng đã được chọn ở bước 2.

51

DRAM bit cell



52

26

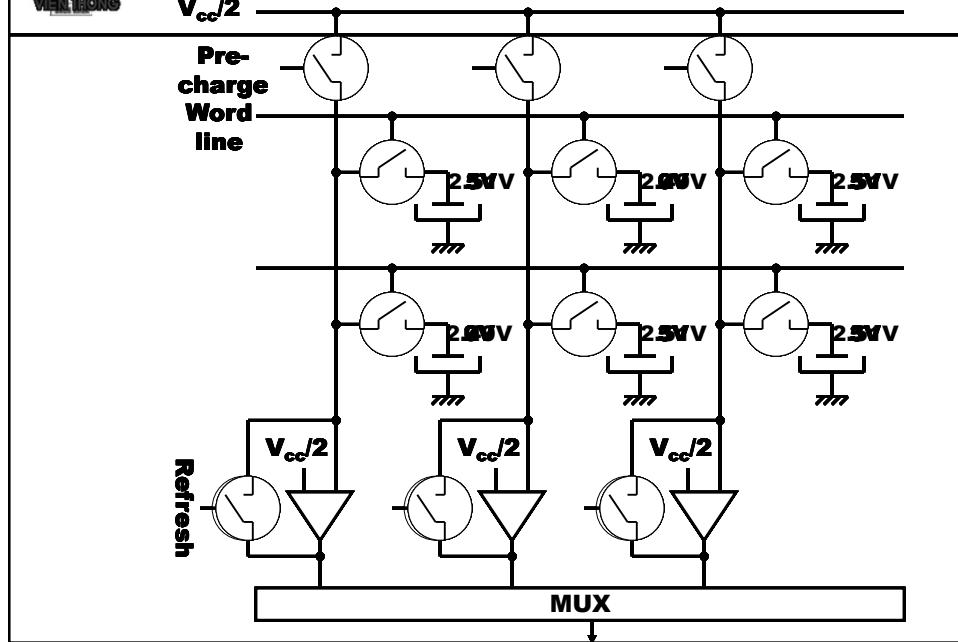
DRAM bit cell

- Chu kỳ ghi

- 1. Precharge
- 2. RAS (Row Address Select): đọc tất cả các bit trong hàng được chọn. Việc đọc này làm giá trị điện áp trên tụ điện bị thay đổi
- 3. Khuếch đại tín hiệu trên các cột tương ứng
- 4.a CAS (Column Address Select): chọn 1 cột và đưa giá trị cần ghi vào cột đó
- 4.b Refresh: khôi phục lại dữ liệu ban đầu của hàng đã được chọn ở bước 2 trừ bit vừa mới được ghi vào.

53

DRAM bit cell



54

27

DRAM bit cell

- Chu kỳ làm tươi

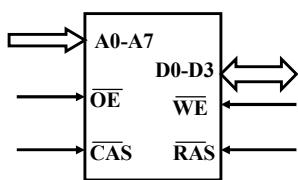
- 1. Precharge
- 2. RAS (Row Address Select): đọc tất cả các bit trong hàng được chọn. Việc đọc này làm giá trị điện áp trên tụ điện bị thay đổi
- 3. Khuếch đại tín hiệu trên các cột tương ứng
- 4. Refresh: khôi phục lại dữ liệu ban đầu của hàng đã được chọn ở bước 2.

55

DRAM

- Đặc điểm:

- 1 transistor 1 bit: rẻ, tuy nhiên việc điều khiển quá trình làm tươi làm tăng giá thành của DRAM
- Chi tiêu thụ năng lượng trong quá trình làm tươi và truy nhập
- Tương đối nhanh: thời gian đọc và ghi 50 ns
- Mỗi một hàng phải được làm tươi sau 4 ms
 - ⇒ Nếu có 1024 hàng, chu kỳ làm tươi sẽ là 4 µs
- Kích thước: 4 Gbits
- Được dùng làm bộ nhớ chính trong các hệ vi xử lý
- Ví dụ: TMS 4464 (64K*4)

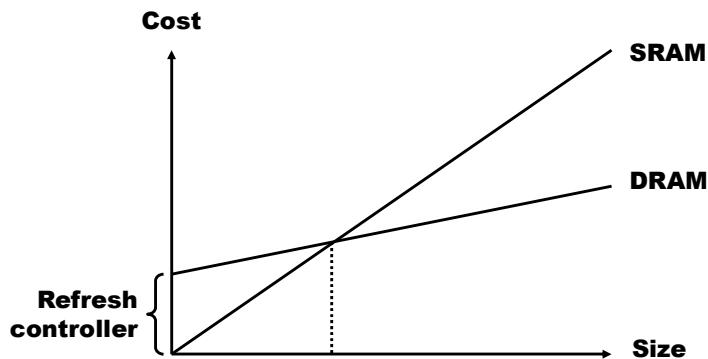


CAS: cho phép chốt địa chỉ cột
RAS: cho phép chốt địa chỉ hàng

56

28

SRAM vs DRAM



57

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.2.1 Các loại bộ nhớ bán dẫn

4.2.2 Giải mã địa chỉ cho bộ nhớ

- Dùng cổng NAND
- Dùng bộ giải mã 74LS138, 74LS139
- Dùng PROM
- Dùng PAL (Programmable Array Logic)

4.2.3 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

58

29

4.2.2 Giải mã địa chỉ bộ nhớ dùng cổng NAND

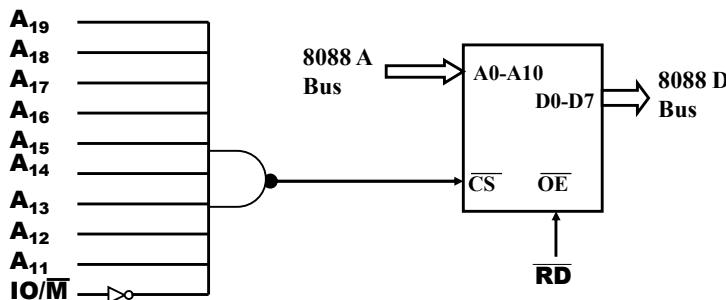
- Ví dụ: Ghép EPROM 2716 (2K * 8) với 8088
- Phân tích:
 - 2716: 11 đường địa chỉ A10-A0
 - 8088: 20 đường địa chỉ A19-A0
 - Chọn vùng nhớ 2K trong 1M?
 - ⇒ EPROM: 00000H-003FFH: không được phép → ko dc động vào
 - ⇒ chọn: FF800H-FFFFFH: chứa đoạn khởi động FFFF0H-FFFFFH

$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11} A_{10}A_9A_8$	$A_7\ A_6\ A_5\ A_4$	$A_3\ A_2\ A_1\ A_0$
----------------------------	----------------------------	-----------------------	----------------------	----------------------

- FF80: 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0
- 1 1 1 1 1 1 1 1 1 | x x x x x x x x x x x
- FFFF: 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 1 1 1

4.2.2 Giải mã địa chỉ bộ nhớ dùng cổng NAND

- FF80: 1 1 1 1 1 1 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0
- 1 1 1 1 1 1 1 1 1 | x x x x x x x x x x x
- FFFF: 1 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 1 1 1



4.2.2 Giải mã địa chỉ bộ nhớ dùng bộ giải mã

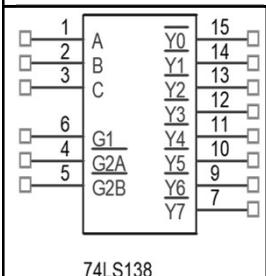
- Ví dụ: Dùng EPROM 2764 (8K*8) để ghép thành bộ nhớ 64 K cho 8088 bắt đầu từ địa chỉ F0000H
 - Phân tích:
 - Địa chỉ bắt đầu F0000H => địa chỉ kết thúc: FFFFFH
 - Cần ghép 8 EPROM 2764 vì $64 = 8 \times 8K$

	A ₁₉ A ₁₈ A ₁₇ A ₁₆	A ₁₅ A ₁₄ A ₁₃	A ₁₂	A ₁₁ A ₁₀ A ₉ A ₈	A ₇ A ₆	A ₅	A ₄	A ₃ A ₂	A ₁ A ₀		
• F0000:	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0					} IC 1
• F1FFF:	1 1 1 1	0 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1					
• F2000:	1 1 1 1	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0				} IC 2	
• F3FFF:	1 1 1 1	0 0 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1					
• F4000:	1 1 1 1	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0				} IC 3	
• F5FFF:	1 1 1 1	0 1 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1					
...											
• FE000:	1 1 1 1	1 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0					} IC 8	
• FFFFF:	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1					

61

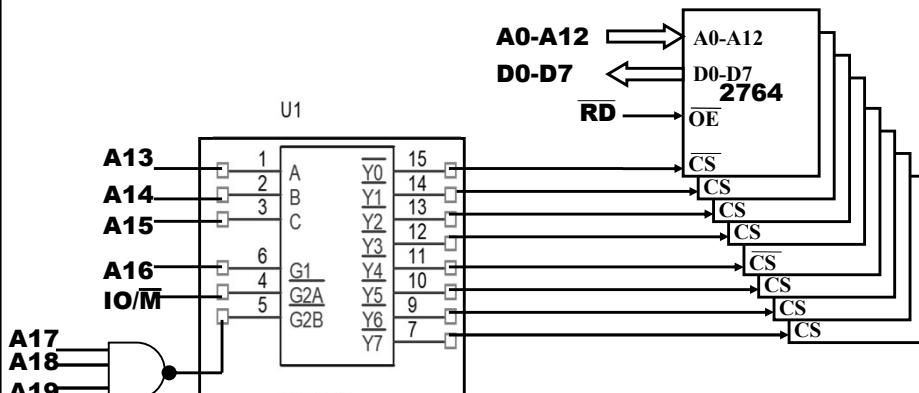
4.2.2 Giải mã địa chỉ bộ nhớ dùng bộ giải mã

- Dùng bộ giải mã 3-8 74LS138



4.2.2 Giải mã địa chỉ bộ nhớ dùng bộ giải mã

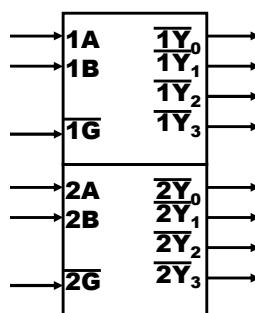
- Dùng bộ giải mã 3-8 74LS138



63

4.2.2 Giải mã địa chỉ bộ nhớ dùng bộ giải mã

- Dùng bộ giải mã kép 2-4 74LS139

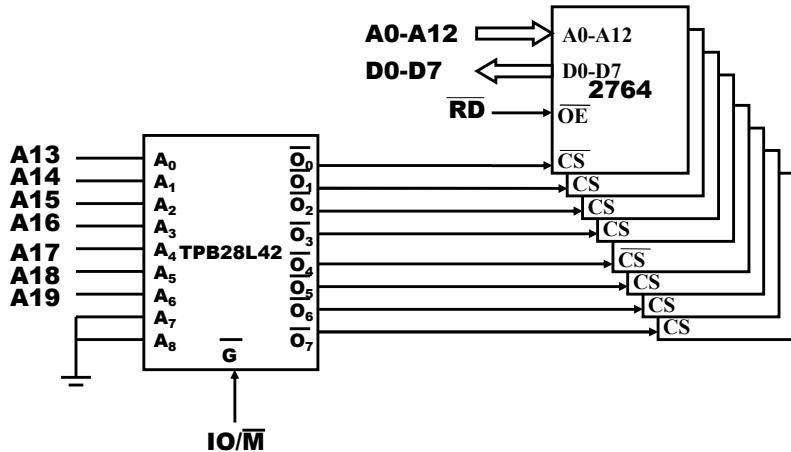


- Ví dụ: Dùng EPROM 27128 (16K*8) để ghép thành bộ nhớ 64 K cho 8088 bắt đầu từ địa chỉ F0000H

64

4.2.2 Giải mã địa chỉ bộ nhớ dùng PROM

- Dùng PROM TPB28L42 (512*8)



65

4.2.2 Giải mã địa chỉ bộ nhớ dùng PAL

AMD 16L8 PAL decoder.

It has 10 fixed inputs (Pins 1-9, 11), two fixed outputs (Pins 12 and 19) and 6 pins that can be either (Pins 13-18).

Programmed to decode address lines $A_{19} - A_{13}$ onto 8 outputs.



Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.2.1 Các loại bộ nhớ bán dẫn

4.2.2 Giải mã địa chỉ cho bộ nhớ

4.2.3 Ghép nối 8088 với bộ nhớ

Ghép nối 8088 với ROM

Ghép nối 8088 với SRAM

Ghép nối 8088 với DRAM

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi



4.2.3 Ghép nối 8088 với bộ nhớ

• Nguyên tắc:

Ghép trực tiếp:

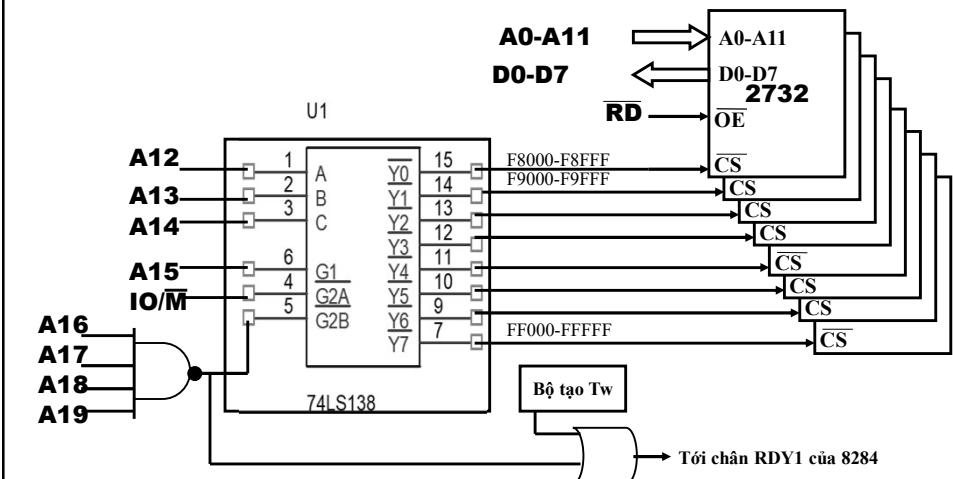
⇒ Thời gian truy cập bộ nhớ của CPU > thời gian truy cập của bộ nhớ + thời gian giải mã địa chỉ

Ghép có chèn thêm thời gian đợi của CPU

⇒ Thời gian truy cập bộ nhớ của CPU < thời gian truy cập của bộ nhớ + thời gian giải mã địa chỉ

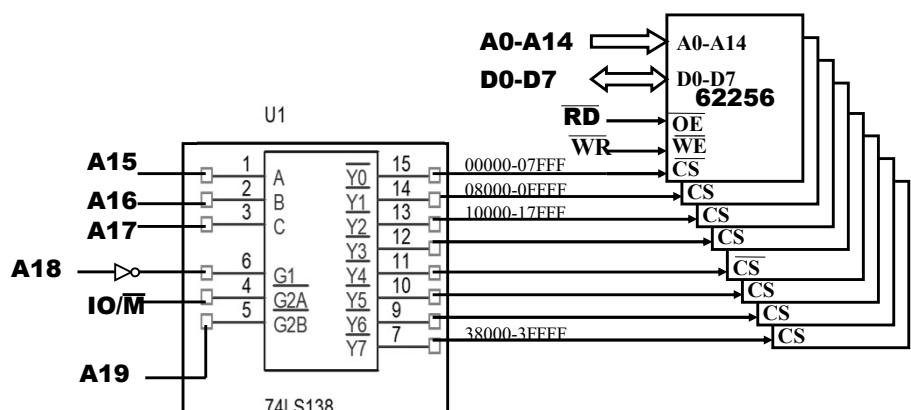
8088 hoạt động ở 5 MHz có thời gian truy cập bộ nhớ 460 ns

- Ví dụ: ghép nối 8088 với EPROM 2732-450 ns



69

- Ví dụ: ghép nối 8088 với SRAM 62256 (32K*8) để được bộ nhớ 256 KB, bắt đầu từ địa chỉ 00000H



70

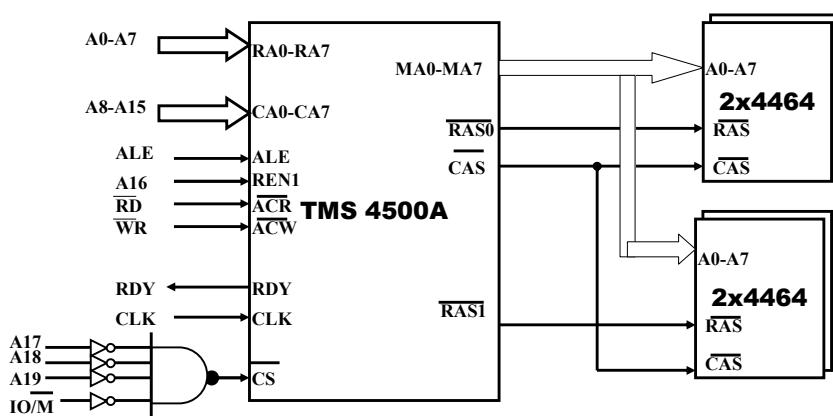
4.2.3 Ghép nối 8088 với bộ nhớ DRAM

- **Cần có DRAM controller:**

- Dòn kênh 2 loại tín hiệu địa chỉ cho mỗi mạch nhớ và cung cấp xung cho phép chốt địa chỉ RAS và CAS
- Cung cấp tín hiệu việc ghi đọc bộ nhớ
- Làm tươi bộ nhớ trong thời gian thích hợp
- Đảm bảo không có xung đột trong hoạt động ghi đọc với công việc làm tươi

4.2.3 Ghép nối 8088 với bộ nhớ DRAM

- **Ví dụ: ghép 8088 với TMS 4464 (64K*4) DRAM để được bộ nhớ 128 KB, bắt đầu tại địa chỉ 00000H**

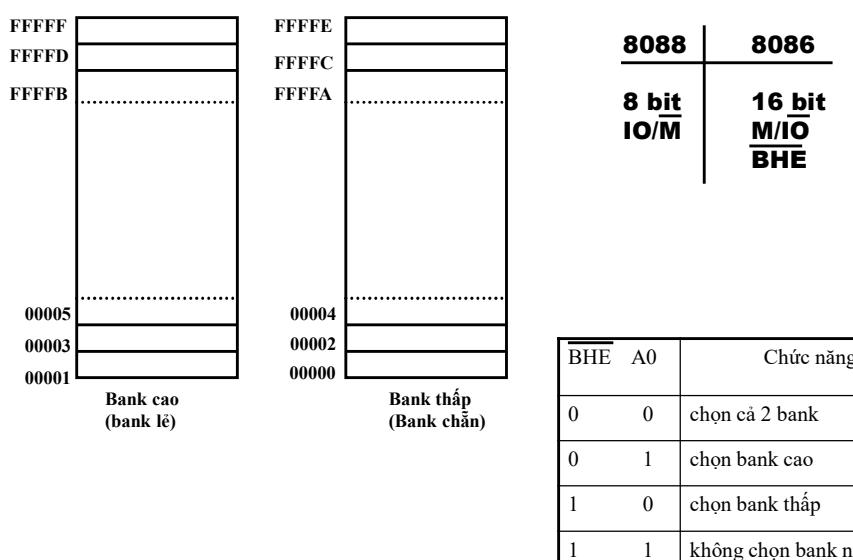


Chương 4: Tổ chức vào ra dữ liệu

- 4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
- 4.2 Ghép nối 8086 với bộ nhớ
- 4.3 Ghép nối 8086 với bộ nhớ
- 4.4 Ghép nối với thiết bị ngoại vi

73

4.3 Ghép nối 8086 với bộ nhớ



74

37

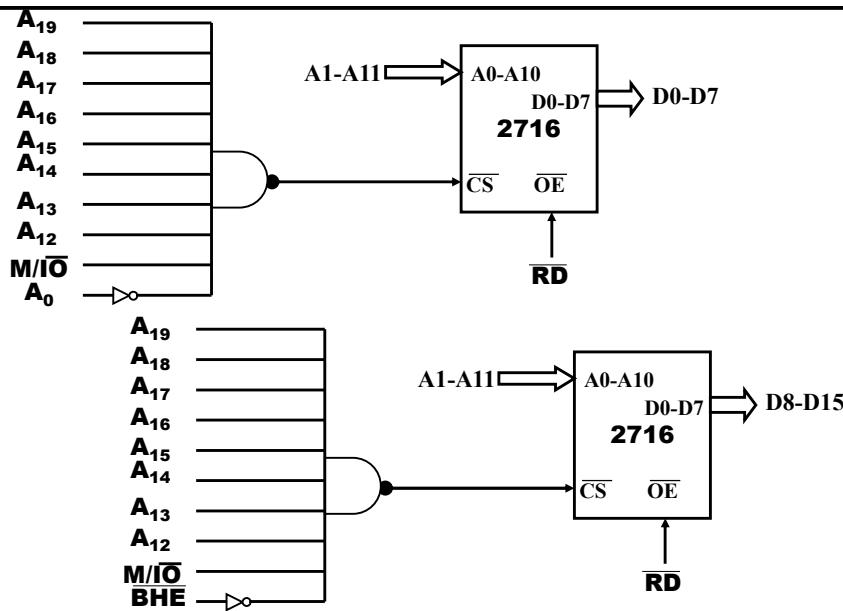
4.3 Ghép nối 8086 với bộ nhớ

- Ví dụ: Ghép EPROM 2716 (2K * 8) với 8086 để được vùng bộ nhớ FF000H-FFFFFH
 - Cần 2 IC vì $4KB = 2^2 \cdot 2KB$

$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$	
• FF000: 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0	
• FFFF: 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0	Bank thấp
• FF001: 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0		1 1 1 1 1 1 1 1	Bank cao
• FFFF: 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	

75

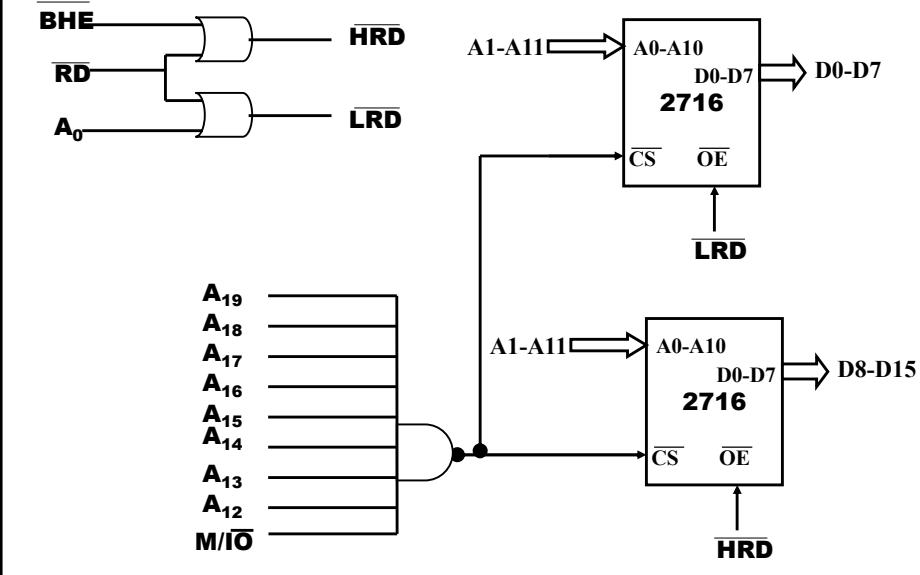
4.3 Ghép nối 8086 với bộ nhớ (Dùng bộ giải mã độc lập cho từng bank)



76

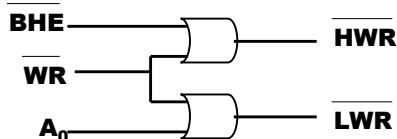
38

4.3 Ghép nối 8086 với bộ nhớ (Dùng chung bộ giải mã 2 bank)



4.3 Ghép nối 8086 với bộ nhớ

- Ví dụ: ghép nối 8086 với SRAM 62256 (32K*8) để được bộ nhớ 256 KB, bắt đầu từ địa chỉ 00000H





4.3 Ghép nối 8086 với bộ nhớ

- Ví dụ: thiết kế hệ thống nhớ cho 8086 với 64 KB EPROM và 128 KB SRAM sử dụng SRAM 62256 (32K*8) và EPROM 27128 (16K*8)



Chương 4: Tổ chức vào ra dữ liệu

- 4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
- 4.2 Ghép nối 8088 với bộ nhớ
- 4.3 Ghép nối 8086 với bộ nhớ
- 4.4 Ghép nối với thiết bị ngoại vi
 - 4.4.1 Các kiểu ghép nối vào/ra
 - 4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra
 - 4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A
 - 4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279
 - 4.4.5 Bộ định thời lập trình được 8254
 - 4.4.6 Giao tiếp truyền thông lập trình được 16550
 - 4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.4.1 Các kiểu ghép nối vào/ra

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A

4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279

4.4.5 Bộ định thời lập trình được 8254

4.4.6 Giao tiếp truyền thông lập trình được 16550

4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804

4.4.1 Các kiểu ghép nối vào ra

- Thiết bị vào ra có không gian địa chỉ cách biệt:

Vùng mở rộng
03F8
03F0
03D0
0378
0320
02F8
0060
0040
0020...
0000

Địa chỉ: 0000H-FFFFH

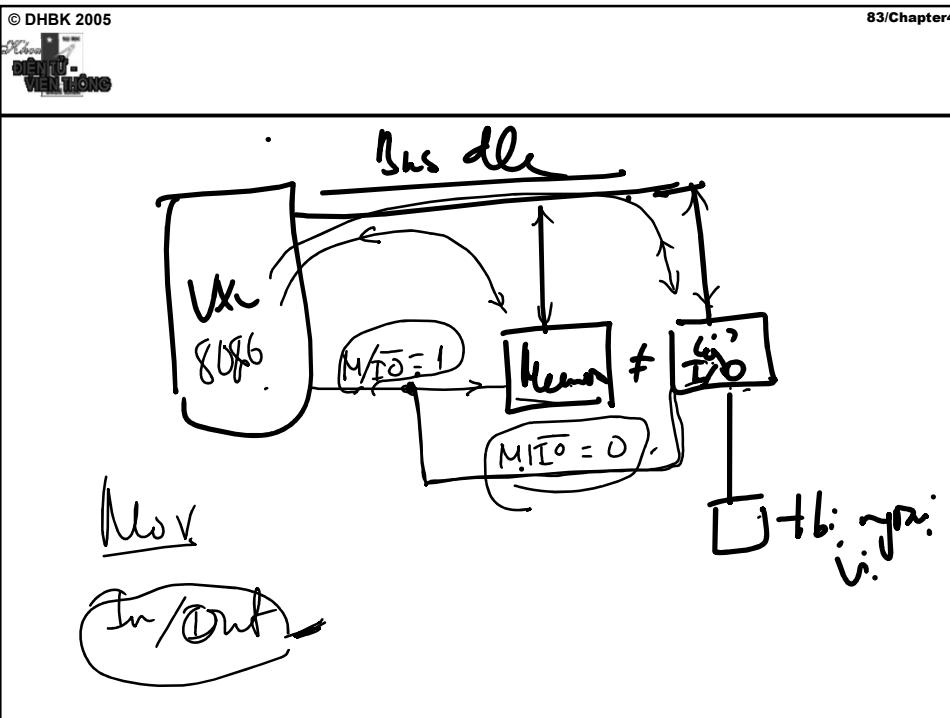
M/I/O=0

Vào ra dữ liệu bằng lệnh IN, OUT

Ví dụ:

IN AX, 00H
IN AL, F0H
IN AX, DX
IN AL, DX

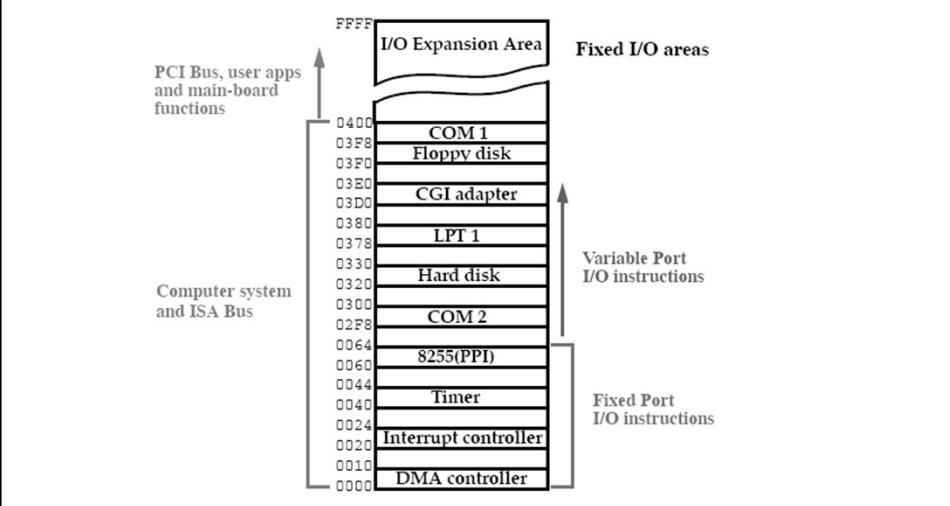
OUT 00H, AX
OUT F0H, AL
OUT DX, AX
OUT DX, AL



83

4.4.1 Các kiểu ghép nối vào ra

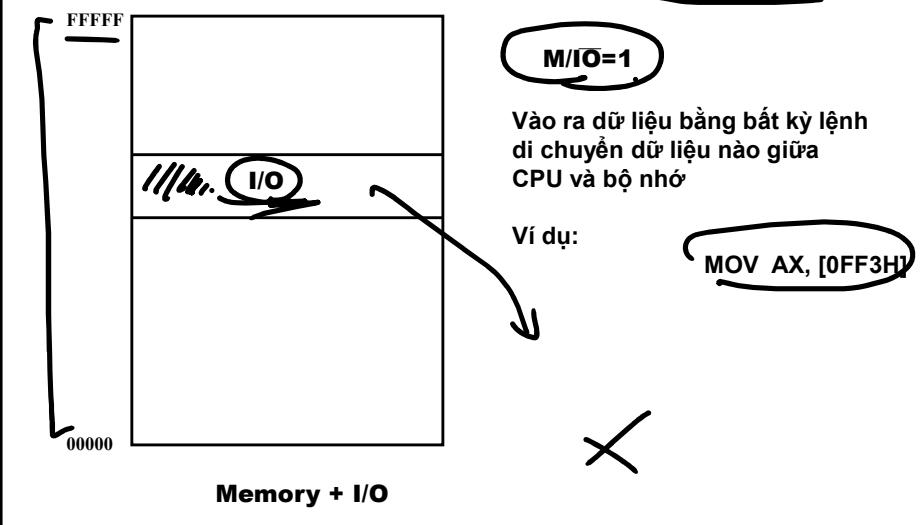
- Thiết bị vào ra có không gian địa chỉ tách biệt với không gian địa chỉ của bộ nhớ:



84

4.4.1 Các kiểu ghép nối vào ra

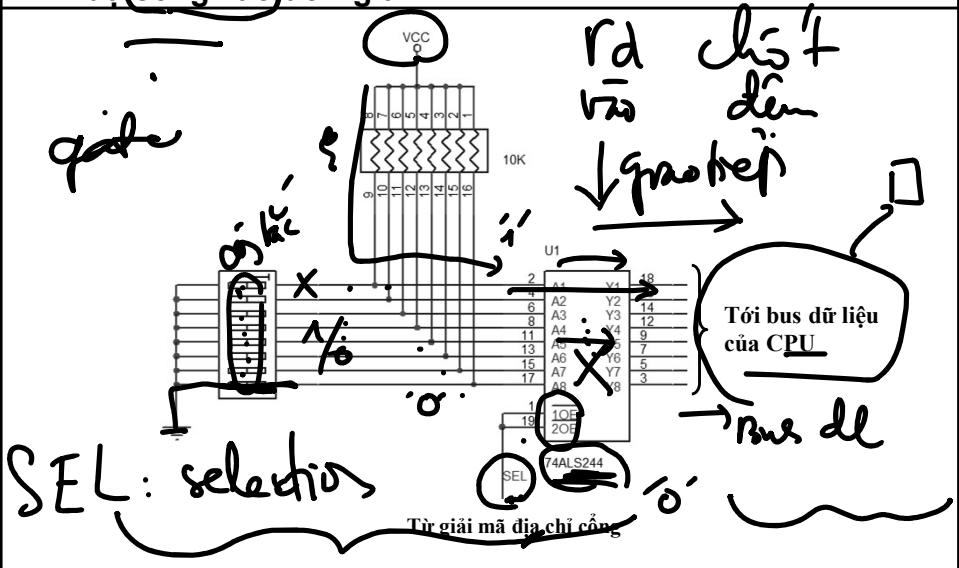
- Thiết bị vào/ra có cùng không gian địa chỉ với bộ nhớ



85

4.4.1 Các kiểu ghép nối vào ra

- Ví dụ cổng vào đơn giản:



86

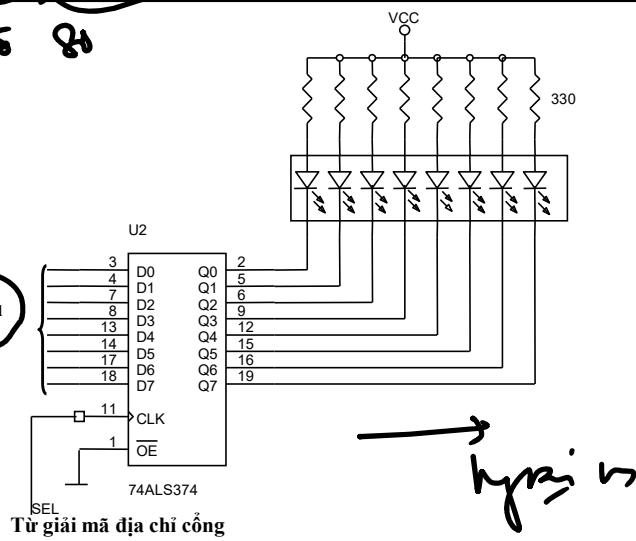
43

4.4.1 Các kiểu ghép nối vào ra

- Ví dụ công nghệ đơn giản:

thì sao

Từ bus dữ liệu
của CPU



hybrid in

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.4.1 Các kiểu ghép nối vào/ra

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A

4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279

4.4.5 Bộ định thời lập trình được 8254

4.4.6 Giao tiếp truyền thông lập trình được 16550

4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804

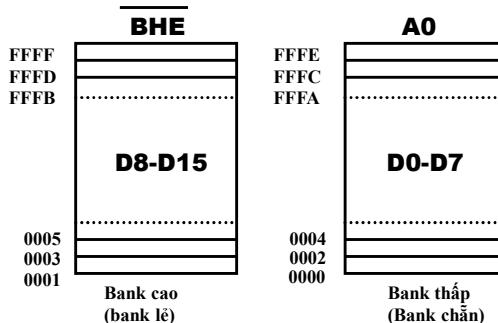
4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

- **8 bit địa chỉ hay 16 bit?**

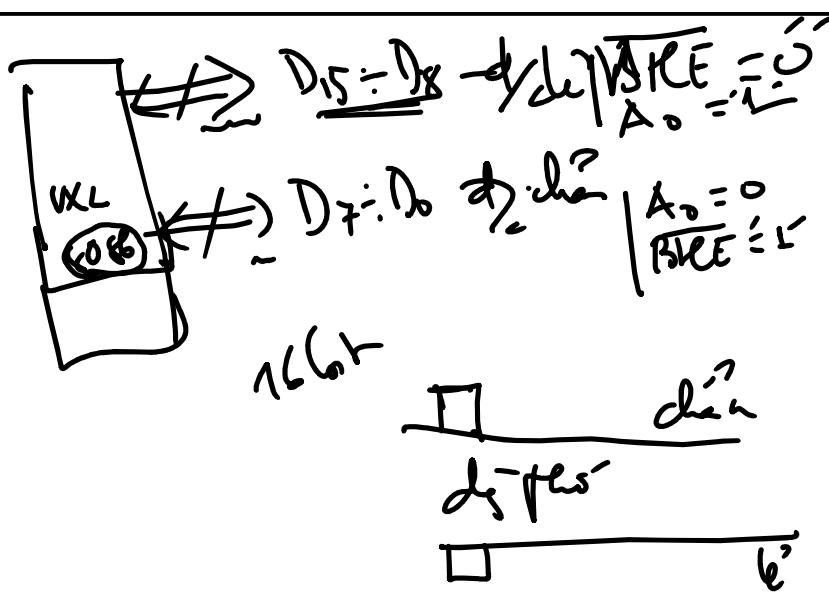
- Tổng số thiết bị < 256: 8 bit A0-A7: 00H-FFH
- Tổng số thiết bị >256: 16 bit A0-A15: 0000H-FFFFH

- **8 bit dữ liệu hay 16 bit?**

- Nếu cổng là 8 bit: chọn 1 trong 2 bank
- Nếu cổng là 16 bit: chọn cả 2 bank

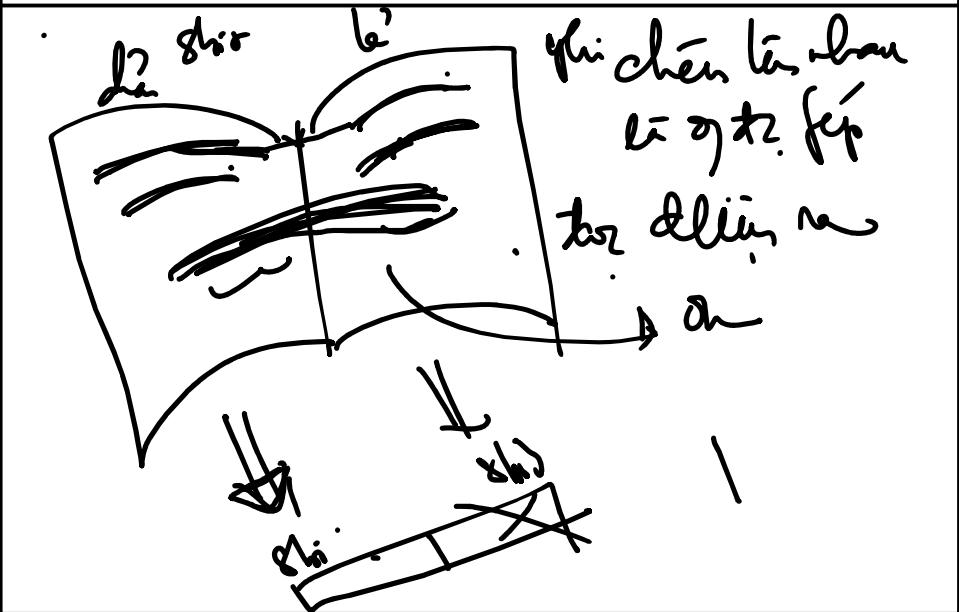


89

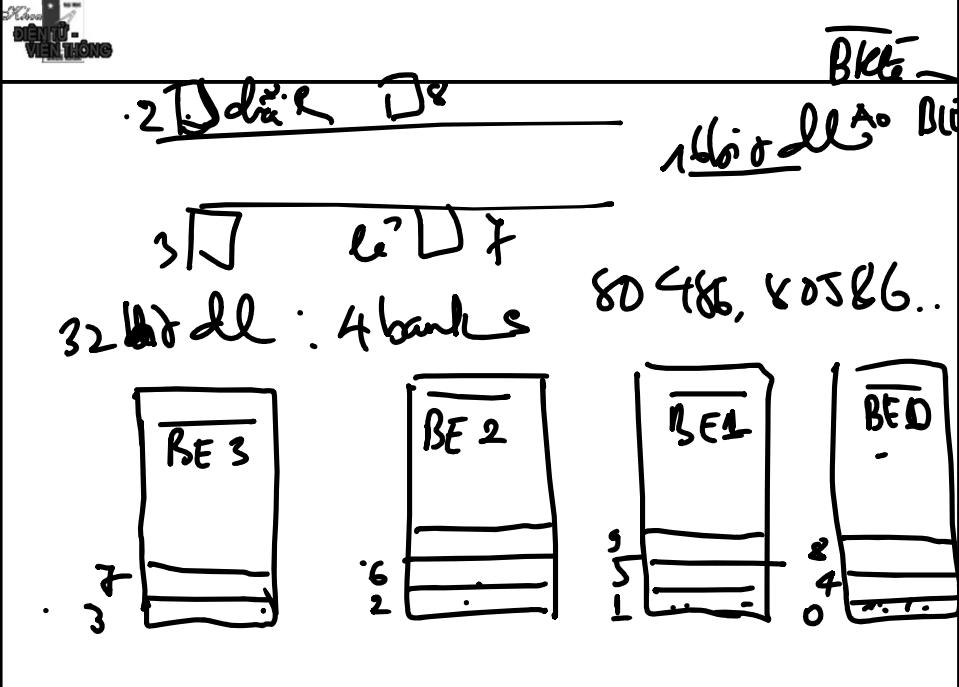


90

45

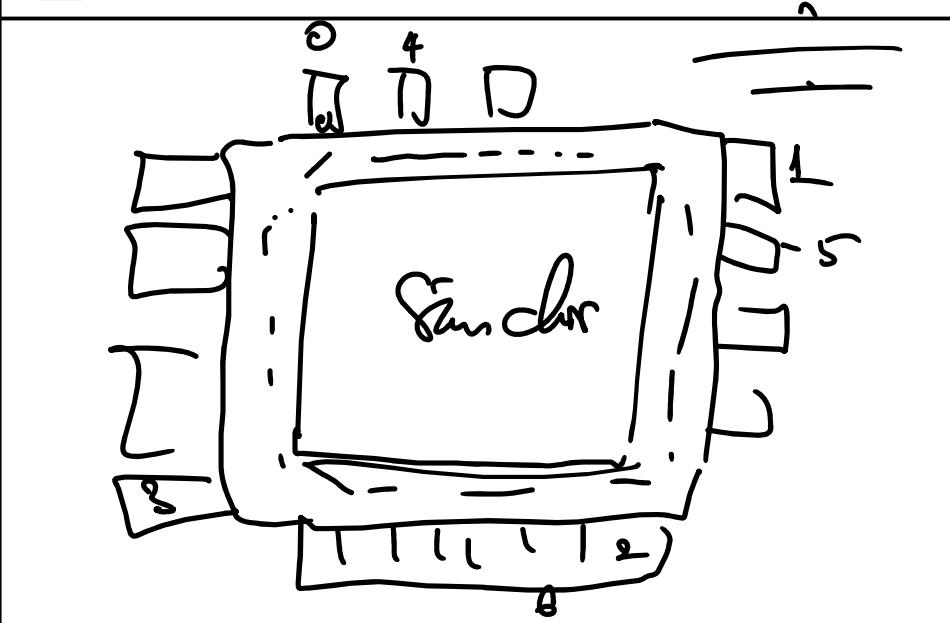


91



92

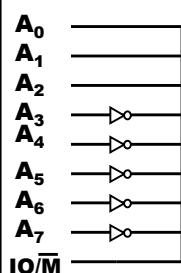
46



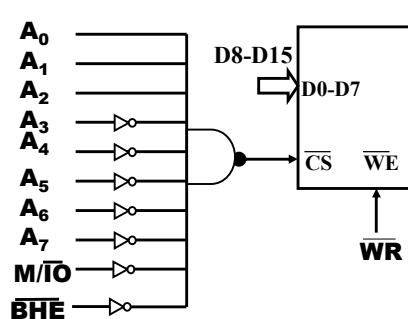
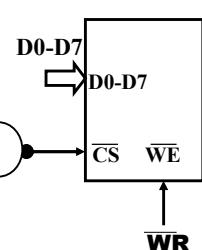
93

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

- Ví dụ: Giải mã địa chỉ cho thiết bị ra 8 bit với địa chỉ 07H**
- 07H = 0000 0111



8088



8086

94

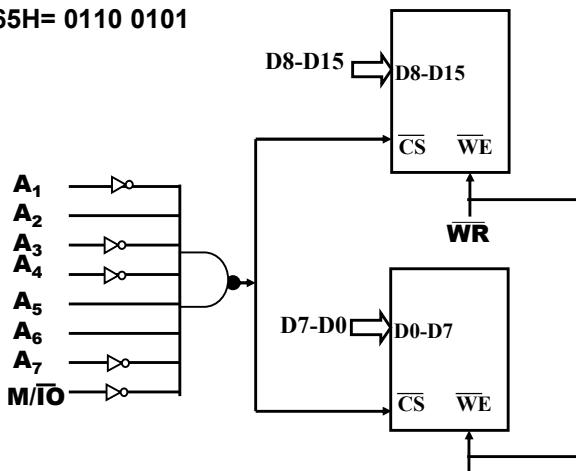
47

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

- Ví dụ: Giải mã địa chỉ cho thiết bị ra 16 bit đi với địa chỉ cổng 64H và 65H

64H= 0110 0100

65H= 0110 0101



95

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

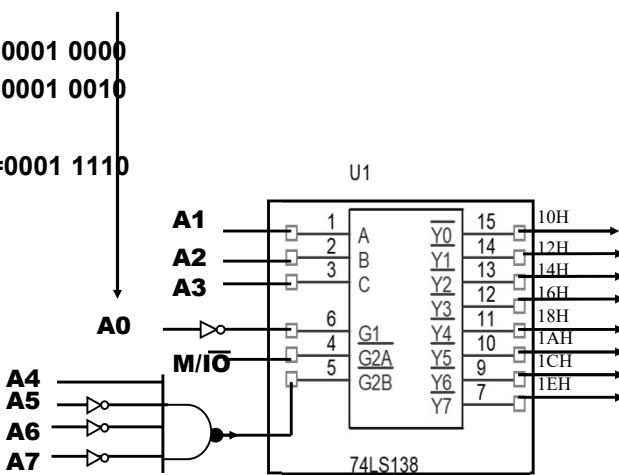
- Ví dụ: Giải mã địa chỉ cho các cổng vào ra 8 bit ở bank thấp với các địa chỉ 10H, 12H, 14H, 16H, 18H, 1AH, 1CH, 1EH

10H=0001 0000

12H=0001 0010

....

1EH=0001 1110



96

48



Chương 4: Tổ chức vào ra dữ liệu

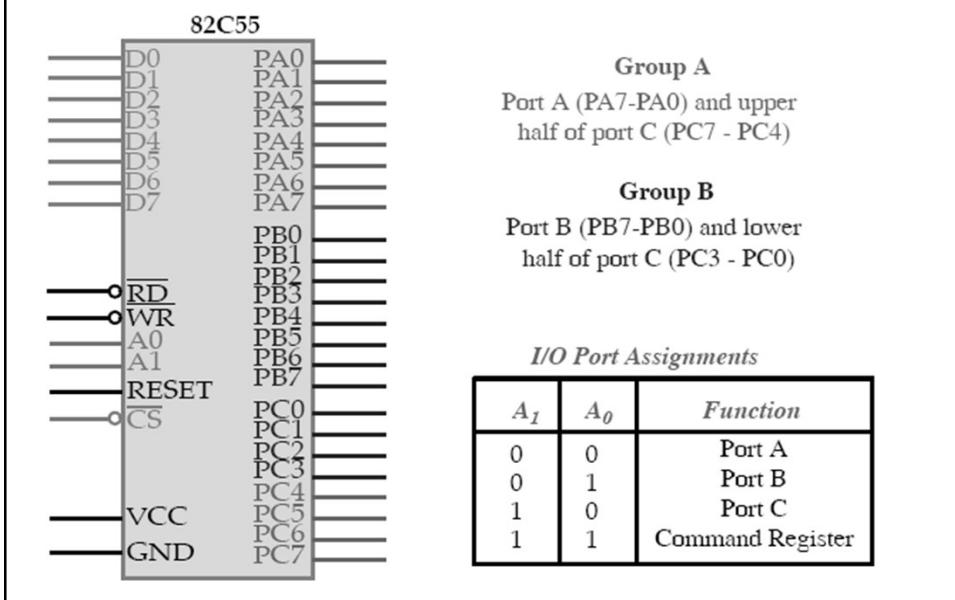
- 4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288**
- 4.2 Ghép nối 8088 với bộ nhớ**
- 4.3 Ghép nối 8086 với bộ nhớ**
- 4.4 Ghép nối với thiết bị ngoại vi**
 - 4.4.1 Các kiểu ghép nối vào/ra**
 - 4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra**
 - 4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A**
 - 4.4.3.1 Cấu trúc của 8255A**
 - 4.4.3.2 Các chế độ làm việc của 8255A**
 - 4.4.3.3 Lập trình cho 8255A**
 - 4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279**
 - 4.4.5 Bộ định thời lập trình được 8254**
 - 4.4.6 Giao tiếp truyền thông lập trình được 16550**
 - 4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804**



4.4.3.1 Cấu trúc của 8255A

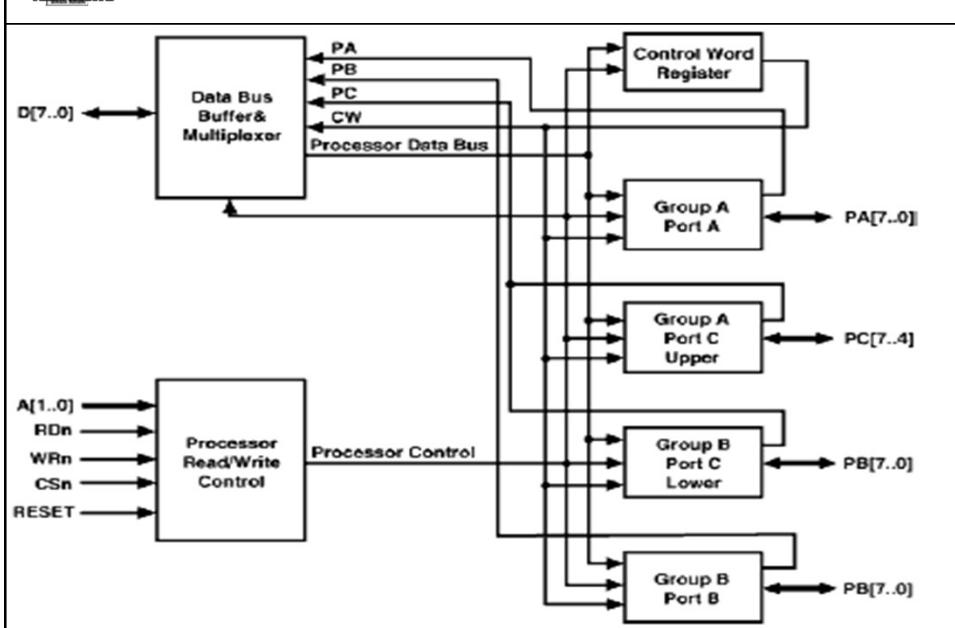
- **Giao tiếp các thiết bị tương thích TTL với vi xử lý**
- **Thường được dùng để giao tiếp bàn phím và máy in trong các máy tính PC (dưới dạng là một khối trong chíp tích hợp)**
- **Cần chèn trạng thái đợi khi làm việc với vi xử lý >8 MHz**
- **Có 24 đường vào ra và có 3 chế độ làm việc**
- **Trong các máy PC, địa chỉ cổng của 8255 là 60H-63H**

4.4.3.1 Cấu trúc của 8255A



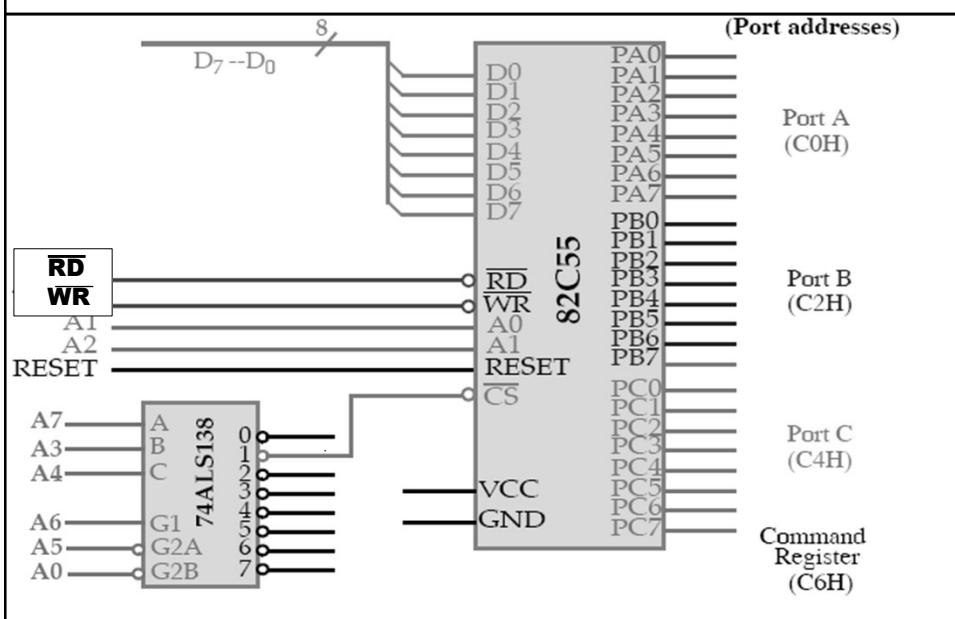
99

4.4.3.1 Cấu trúc của 8255A



100

4.4.3.1 Cấu trúc của 8255A



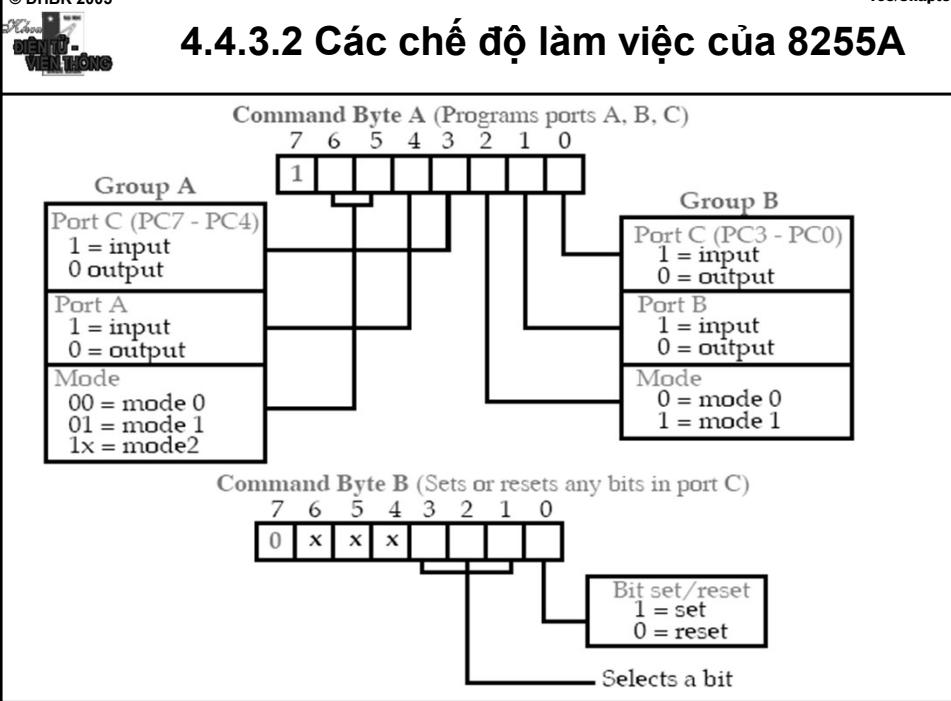
101

- Hãy ghép nối 8255 với VXL 8086 để trao đổi dữ liệu với thiết bị ngoại vi 16 bit dữ liệu (ví dụ như vxl nhận dữ liệu 16 bit từ IC ADC, thông qua 8255) ?

102



4.4.3.2 Các chế độ làm việc của 8255A



103



4.4.3.2 Các chế độ làm việc của 8255A

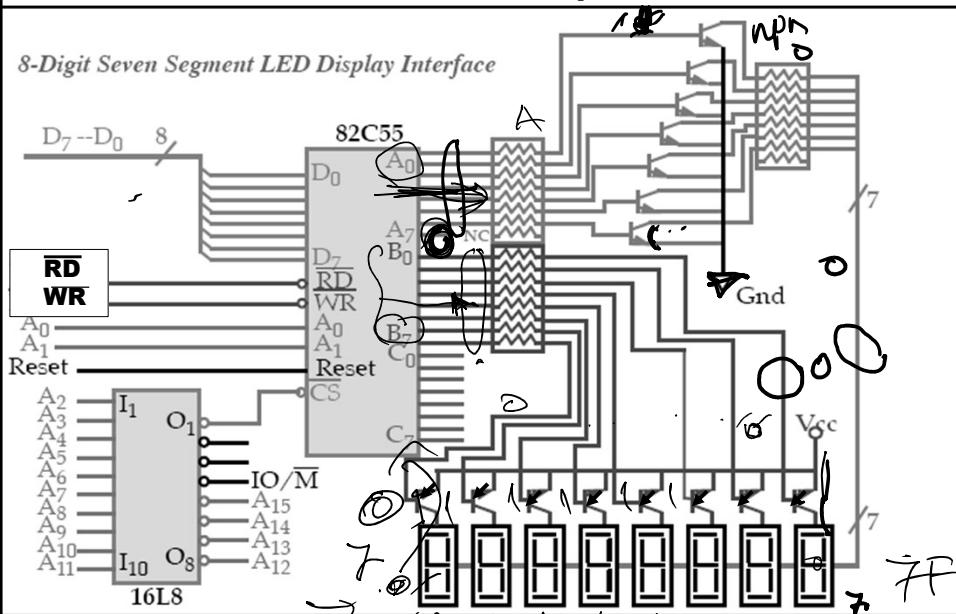
- **Chế độ 0:** Chế độ vào/ra đơn giản: các cổng có thể làm việc như là cổng vào có đệm hoặc cổng ra có chốt đệm.
- **Chế độ 1:** Chế độ này cho phép cổng A và B làm việc như các thiết bị vào hoặc ra có tín hiệu móc nối (handshaking) do các bit tương ứng của cổng C trong cùng nhóm đảm nhiệm
- **Chế độ 2:** chế độ này cho phép cổng A làm việc 2 chiều với các tín hiệu móc nối do cổng PC_H đảm nhiệm. Cổng B có thể làm việc ở chế độ 0

104

4.4.3.2 Các chế độ làm việc của 8255A

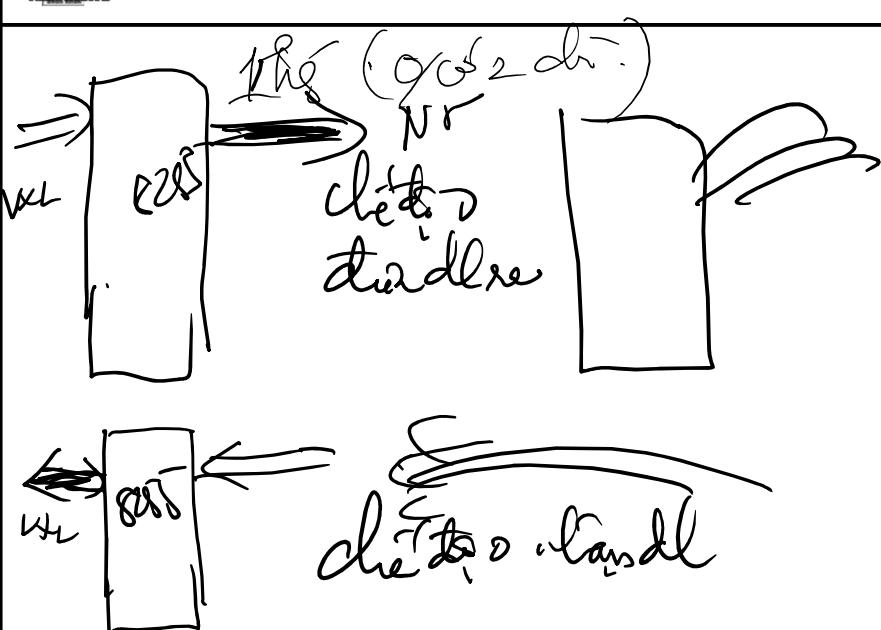
Chế độ 0

105/Chapter



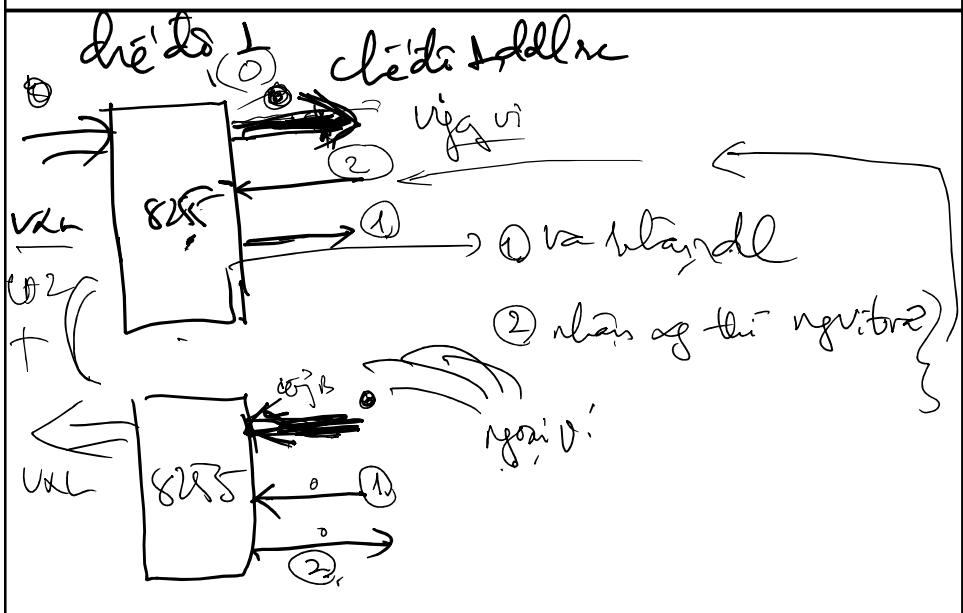
105

106/Chapter

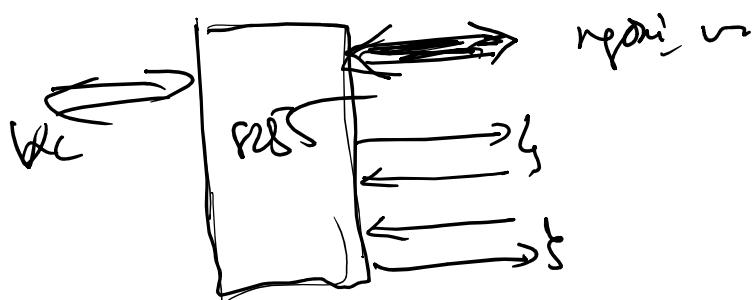


106

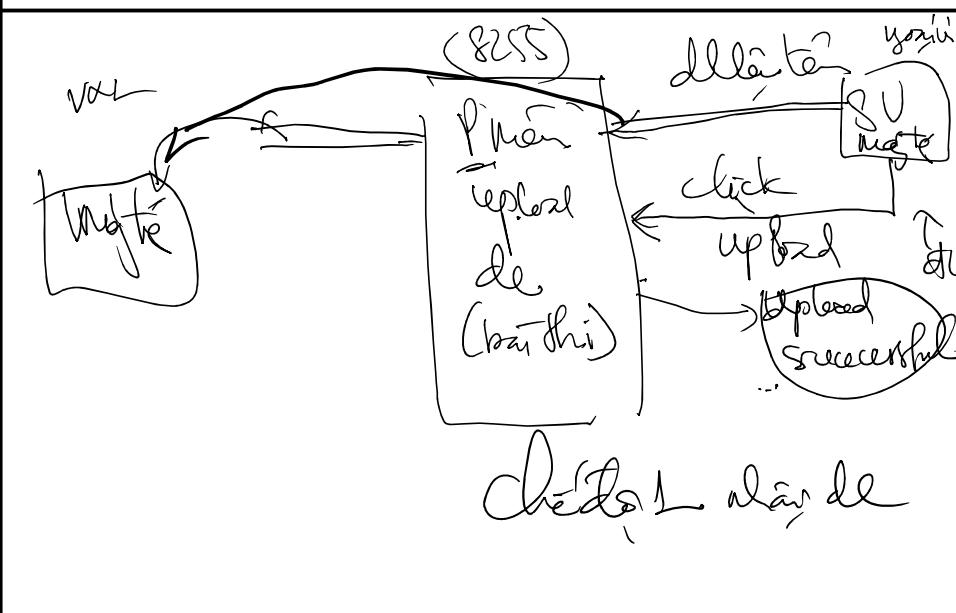
53



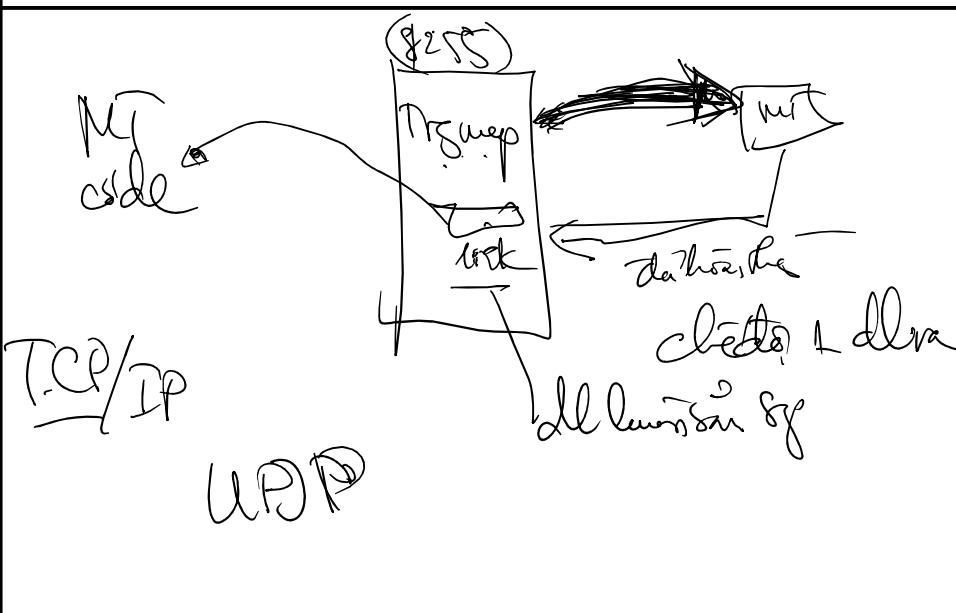
107



108



109



110

© DHBK 2005

Giả thiết
địa chỉ
của các
cổng
của
8255 là
0700H-
0703H

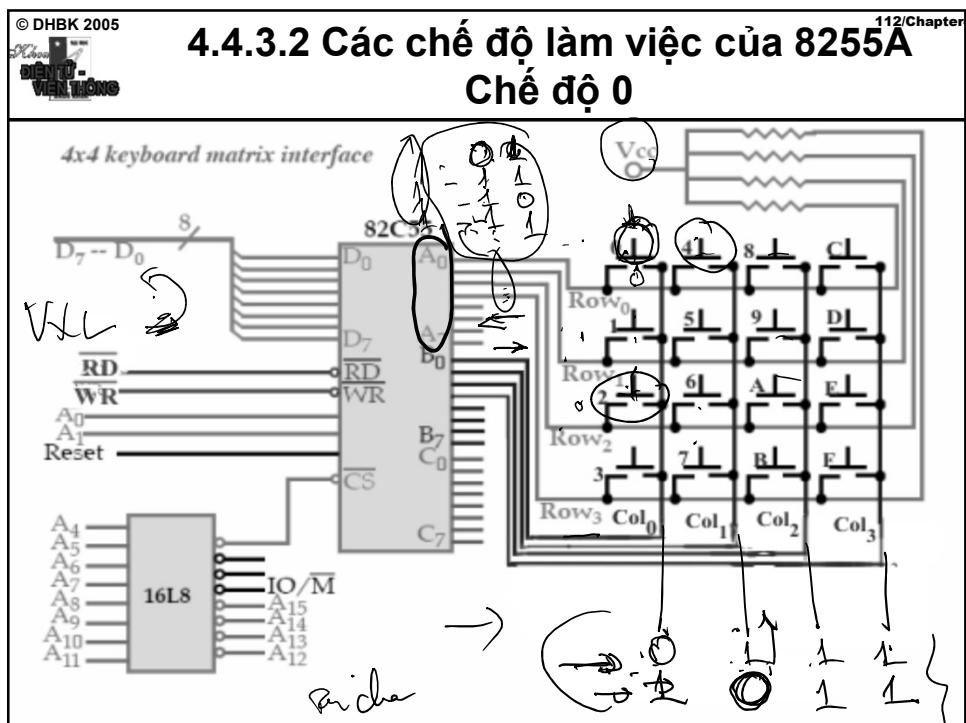
```

; Lập trình cho 8255
MOV AL, 10000000B           ; Port A, Port B mode 0, output
MOV DX, 703H
OUT DX, AL

; Thủ tục hiển thị LED từ dữ liệu chứa trong bộ nhớ
DISP PROC NEAR
    PUSHF
    PUSH AX
    PUSH BX
    PUSH DX
    PUSH SI
    ; Thiết lập các thanh ghi để hiển thị
    MOV BX, 8
    MOV AH, 7FH
    MOV SI, OFFSET MEM-1
    MOV DX, 701H
    ; Hiển thị 8 số
DISP1: MOV AL, AH
        OUT DX, AL
        DEC DX
        MOV AL, [BX+SI]
        OUT DX, AL
        CALL Delay
        ROR AH, 1
        INC DX
        DEC BX
        JNZ DISP1
    ; Khôi phục lại các thanh ghi
    POP SI
    POP DX
    POP BX
    POP AX
    POPF
    RET
DISP ENDP

```

111



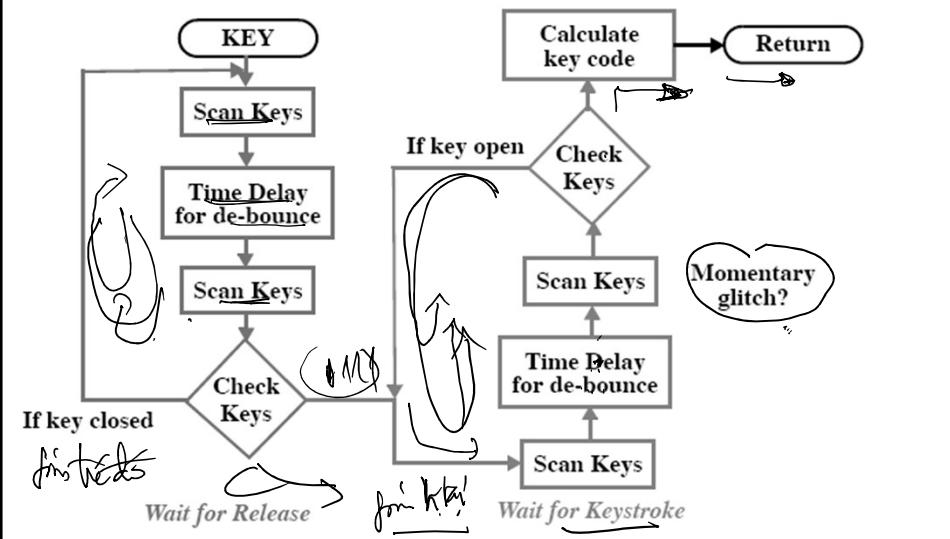
112

4.4.3.2 Các chế độ làm việc của 8255A

Chế độ 0

113/Chapter

Flow chart of a keyboard-scanning procedure



113

114/Chapter				
ROWS EQU 4 ; 4 hàng	SCAN PROC NEAR USES CL, ROWS ;form row mask			
COLS EQU 4 ; 4 cột	MOV MOV BH, OFFH			
PORTA EQU 50H	SHL SHL BH, CL			
PORTB EQU 51H	MOV MOV CX, COLS ;load column count			
KEY PROC NEAR USES CX	MOV MOV BL, OFEH ;get selection mode			
CALL SCAN ;test all keys	SCAN1: MOV AL, BL ;select column			
JNZ KEY ; if key closed	OUT OUT PORTB, AL			
CALL DELAY ; đợi 10 ms	ROL ROL BL, 1			
CALL SCAN	IN IN AL, PORTA; read rows			
JNZ KEY	OR OR AL,BH			
	CMP CMP AL, 0FFH ;test for a key			
	JNZ JNZ SCAN2			
	LOOP LOOP SCAN1			
KEY1:	SCAN2: RET			
CALL SCAN	SCAN ENDP			
JZ KEY1 ; if no key closed	DELAY PROC NEAR USES CX			
CALL DELAY	MOV MOV CX, 5000 ;10ms (8MHZ)			
CALL SCAN	DELAY1: LOOP RET			
JZ KEY1	DELAY ENDP			
PUSH AX ;cắt mã hàng				
MOV AL, COLS ;cal starting row key				
SUB AL, CL				
MOV CH, ROWS				
MUL CH				
MOV CL, AL				
DEC CL				
POP AX				
KEY2:				
ROR AL,1 ;find row position				
INC CL				
JC KEY2				
MOV AL,CL ;move code to AL				
RET				
KEY ENDP				

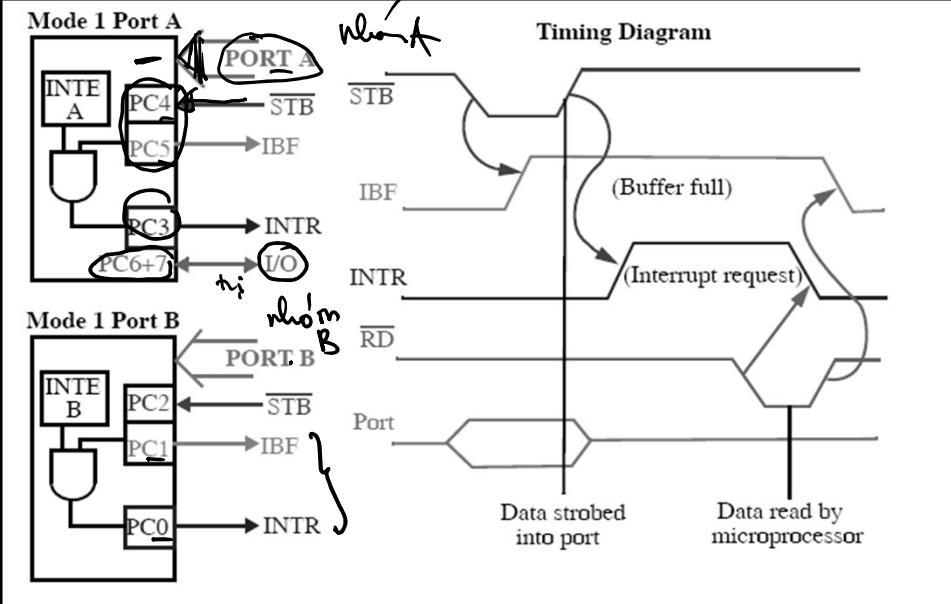
57

114

4.4.3.2 Các chế độ làm việc của 8255A

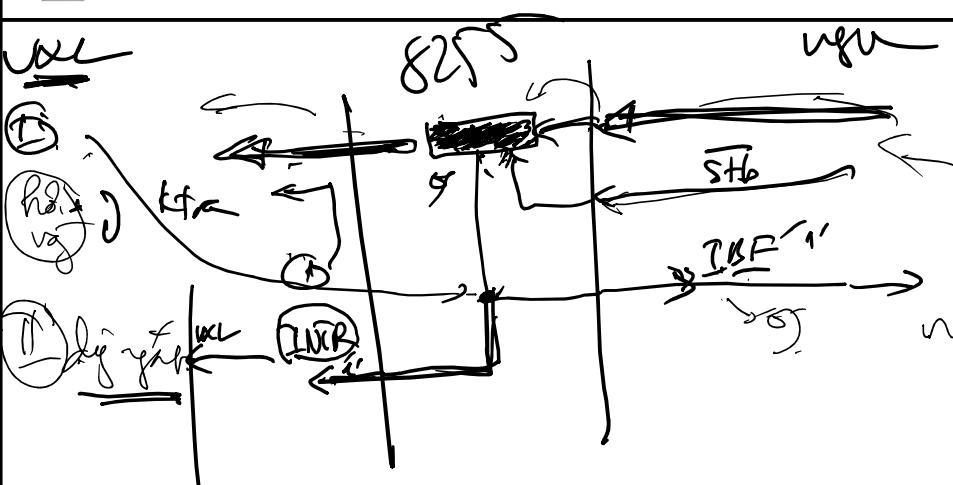
Chế độ 1 nhận dữ liệu

115/Chapter



115

116/Chapter



116

58

4.4.3.2 Các chế độ làm việc của 8255A Chế độ 1

117/Chapter

- Port A và B làm việc ở chế độ cỗng vào có chốt:

- dữ liệu sẽ được giữ tại cỗng A, B cho đến khi CPU sẵn sàng
- cỗng C làm cỗng điều khiển và cấp tín hiệu móc nối

STB The strobe input loads data into the port latch on a 0-to-1 transition

IFB Input buffer full is an output indicating that the input latch contain information

INTR Interrupt request is an output that requests an interrupt

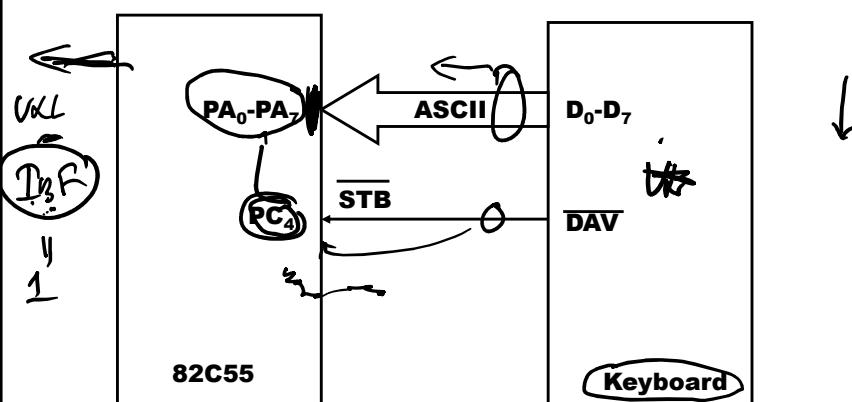
INTE The interrupt enable signal is neither an input nor an output; it is an internal bit programmed via the PC4(port A) or PC2(port B) bits.

PC7,PC6 The port C pins 7 and 6 are general-purpose I/O pins that are available for any purpose.

117

4.4.3.2 Các chế độ làm việc của 8255A Chế độ 1

118/Chapter

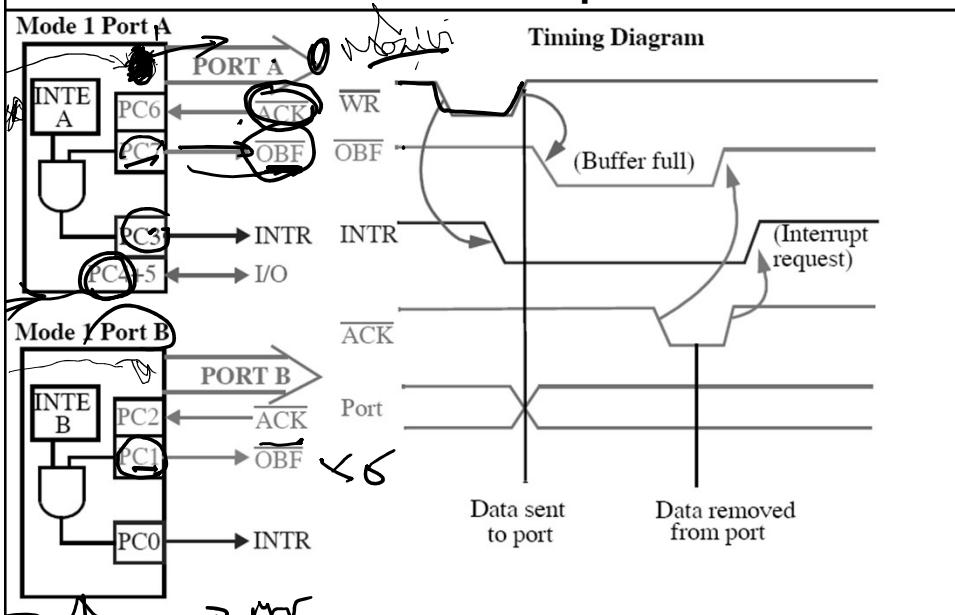


118

59

4.4.3.2 Các chế độ làm việc của 8255A Chế độ 1

119/Chapter



119

4.4.3.2 Các chế độ làm việc của 8255A Chế độ 1

120/Chapter

- Port A và B làm việc ở chế độ cổng ra có chốt:
 - tương tự như cổng ra ở chế độ 0
 - cổng C làm cổng điều khiển và cấp tín hiệu móc nối

QBF

Output buffer full is an output that goes low when data is latched in either port A or port B. Goes low on ACK.

ACK

The acknowledge signal causes the OBF pin to return to **1**. This is a response from an external device.

INTR

Interrupt request is an output that requests an interrupt

INTE

The interrupt enable signal is neither an input nor an output; it is an internal bit programmed via the PC6(port A) or PC2(port B) bits.

PC5,PC4

The port C pins 5 and 4 are general-purpose I/O pins that are available for any purpose.

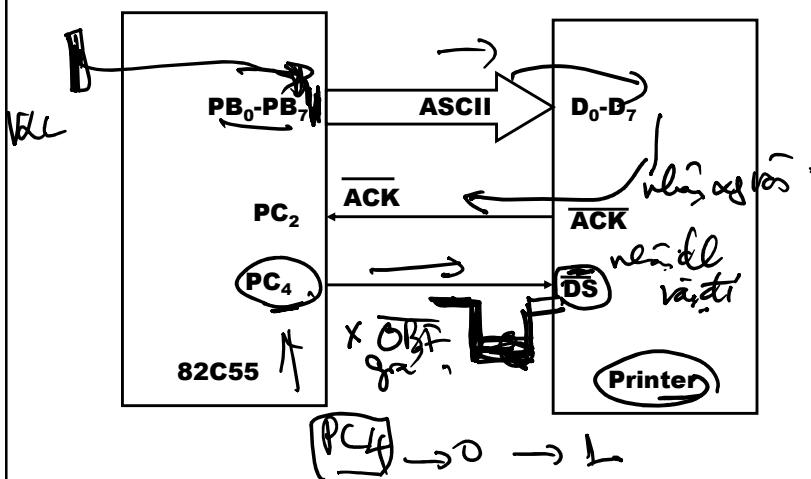
120

60

4.4.3.2 Các chế độ làm việc của 8255A

Chế độ 1

121/Chapter

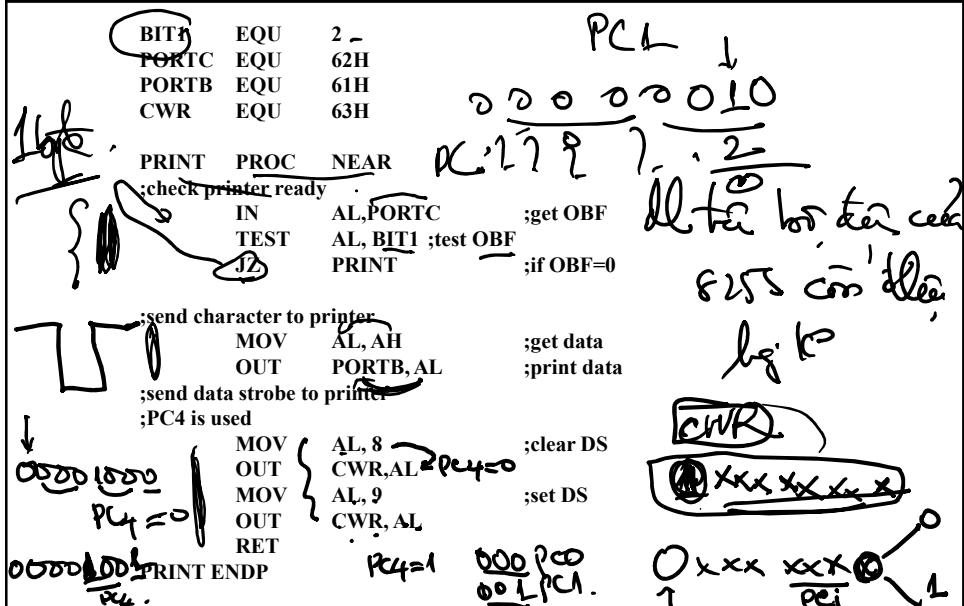


121

4.4.3.2 Các chế độ làm việc của 8255A

Chế độ 1

122/Chapter



122

61

4.4.3.2 Các chế độ làm việc của 8255A

123/Chapter

Chế độ 2

- Chỉ cho phép đối với cổng A
- Cổng A là cổng 2 chiều, dùng để giao tiếp giữa 2 máy tính hoặc dùng trong chuẩn giao tiếp IEEE-488 GPIB...

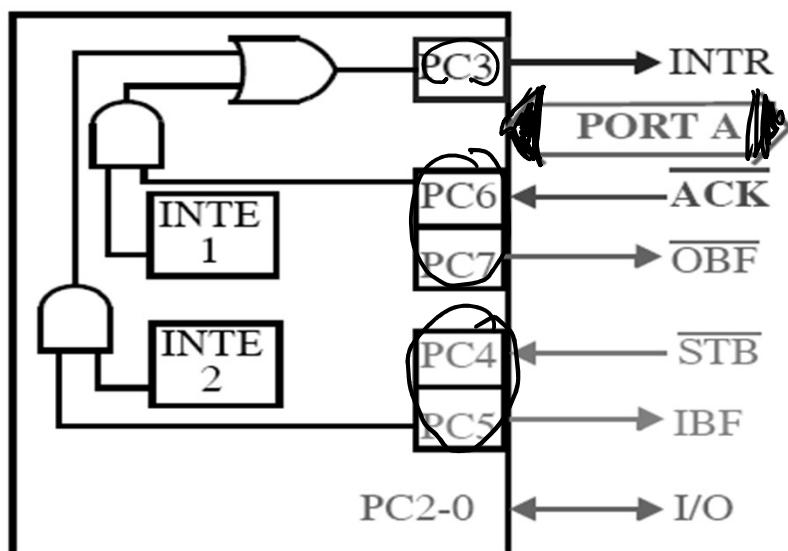
INTR	Interrupt request is an output that requests an interrupt
OBF	Output buffer full is an output indicating that the output buffer contains data for the bi-directional bus
ACK	Acknowledge is an input that enables tri-state buffers which are otherwise in their high-impedance state
STB	The strobe input loads data into the port A latch
IFB	Input buffer full is an output indicating that the input latch contains information for the external bi-directional bus
INTE	Interrupt enable are internal bits that enable the INTR pin. Bit PC6(INTE1) and PC4(INTE2)
PC2,PC1 and PC0	These port C pins are general-purpose I/O pins that are available for any purpose.

123

4.4.3.2 Các chế độ làm việc của 8255A

124/Chapter

Chế độ 2



124

62



Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.4.1 Các kiểu ghép nối vào/ra

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A

4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279

4.4.5 Bộ định thời lập trình được 8254

4.4.6 Giao tiếp truyền thông lập trình được 16550

4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804



4.4.4 Mạch điều khiển 8279

• Điều khiển bàn phím và màn hiển thị 8279:

- quét và mã hóa cho bàn phím tới 64 phím
 - ⇒ bộ đệm FIFO có thể chứa 8 ký tự
- Điều khiển màn hiển thị tới 16 số
 - ⇒ 16*8 RAM để chứa thông tin về 16 số hiển thị

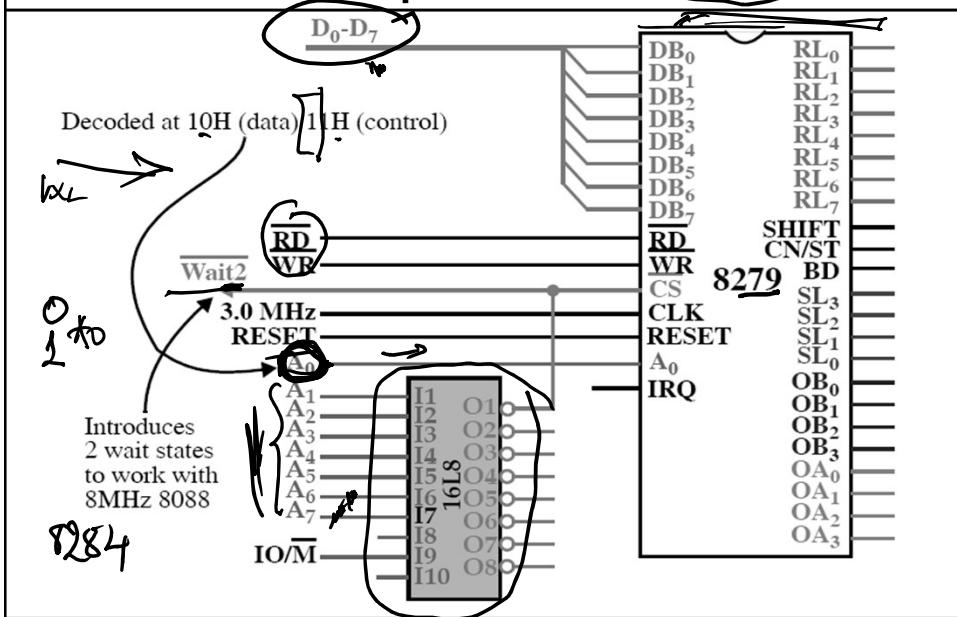
• Các tín hiệu chính:

- A₀: chọn giữa chế độ dữ liệu hoặc điều khiển
- BD: xoá trống màn hiển thị
- CLK: tín hiệu xung nhịp vào
- CN/ST (control/Strobe): cổng vào nối với phím điều khiển của bàn phím
- CS: chip select
- DB₇-DB₀: bus dữ liệu 2 chiều
- IRQ: =1 khi có phím bấm
- OUTA3-OUTA0: dữ liệu tới màn hiển thị (bit cao)
- OUTB3-OUTB0: dữ liệu tới màn hiển thị (bit thấp)
- RD: cho phép đọc dữ liệu từ thanh ghi điều khiển hoặc trạng thái
- RL7-RL0: xác định phím được nhấn
- SHIFT: nối với phím shift của bàn phím
- SL3-SL0: tín hiệu quét màn hình và màn hiển thị
- WR: viết dữ liệu vào thanh ghi điều khiển hoặc thanh ghi dữ liệu

RL ₂	V _{CC}
RL ₃	RL ₁
CLK	RL ₀
IRQ	CNTL/STB
RL ₄	SHIFT
RL ₅	SL ₃
RL ₆	SL ₂
RL ₇	SL ₁
RESET	SL ₀
RD	OUT B ₀
WR	OUT B ₁
DB ₀	OUT B ₂
DB ₁	OUT B ₃
DB ₂	OUT A ₀
DB ₃	OUT A ₁
DB ₄	OUT A ₂
DB ₅	OUT A ₃
DB ₆	BD
DB ₇	CS
V _{SS}	A ₀

4.4.4 Mạch điều khiển 8279 Ghép nối 8279 với 8088

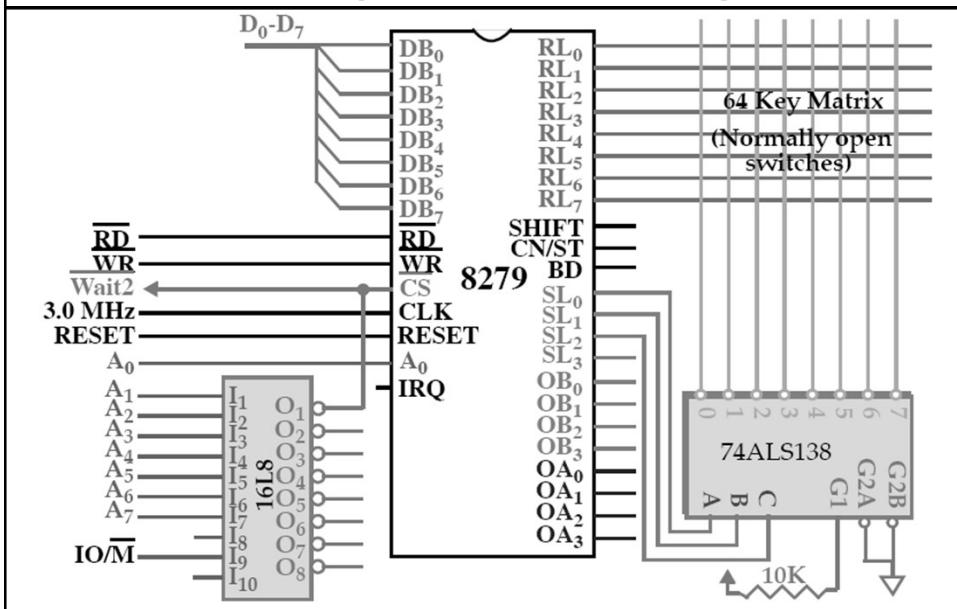
127/Chapter



127

4.4.4 Mạch điều khiển 8279 Ghép nối 8279 với bàn phím

128/Chapter

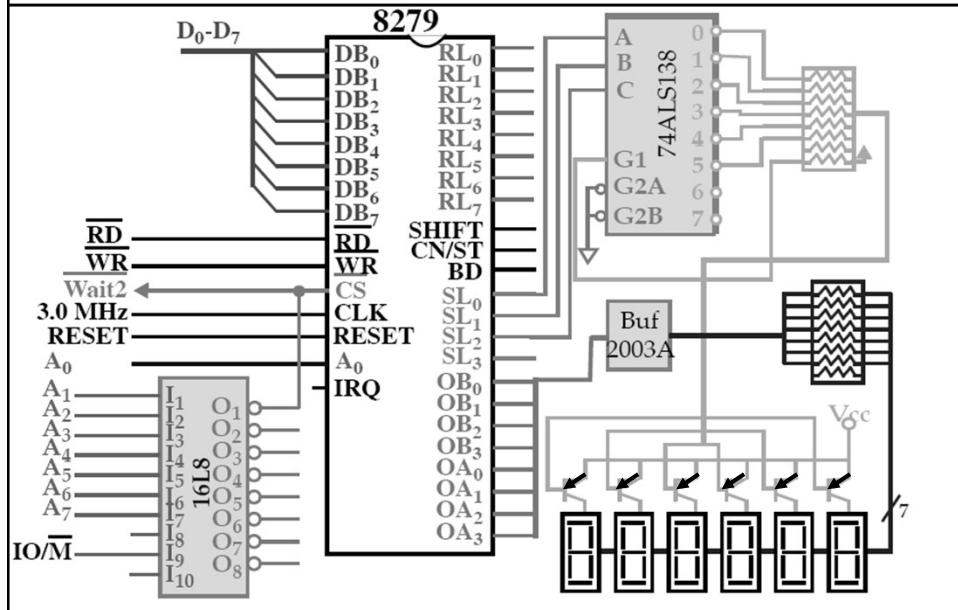


128

64

4.4.4 Mạch điều khiển 8279 Ghép nối 8279 với màn hiển thị

129/Chapter



129

4.4.4 Mạch điều khiển 8279 Lập trình cho 8279

130/Chapter

- Tùy điều khiển: D₇D₆D₅D₄D₃D₂D₁D₀ 8

D ₇	D ₆	D ₅	Function	Purpose
0	0	0	Mode set	Selects the number of display positions, type of key scan...
0	0	1	Clock	Programs internal clk, sets scan and debounce times.
0	1	0	Read FIFO	Selects type of FIFO read and address of the read.
0	1	1	Read Display	Selects type of display read and address of the read.
1	0	0	Write Display	Selects type of write and the address of the write.
1	0	1	Display write inhibit	Allows half-bytes to be blanked.
1	1	0	Clear	Clears the display or FIFO
1	1	1	End interrupt	Clears the IRQ signal to the microprocessor.

130

65

4.4.4 Mạch điều khiển 8279

Lập trình cho 8279

○ 000DDMM

Mode set: Opcode 000.

DD sets displays mode.

MMM sets keyboard mode.

DD field selects either:

- 8- or 16-digit display
- Whether new data are entered to the rightmost or leftmost display position.

DD	Function
00	8-digit display with left entry
01	16-digit display with left entry
10	8-digit display with right entry
11	16-digit display with right entry

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.4.1 Các kiểu ghép nối vào/ra

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A

4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279

4.4.5 Bộ định thời lập trình được 8254

4.4.6 Giao tiếp truyền thông lập trình được 16550

4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804

4.4.5 Bộ định thời lập trình được 8254

Three independent 16-bit programmable counters (timers).

Each capable in of counting in binary or BCD with a maximum frequency of 10MHz.

Used for controlling real-time events such as real-time clock, events counter, and motor speed and direction control.

Usually decoded at port address 40H-43H and has following functions:

- Generates a basic timer interrupt that occurs at approximately 18.2Hz.

Interrupts the micro at interrupt vector 8 for a clock tick.

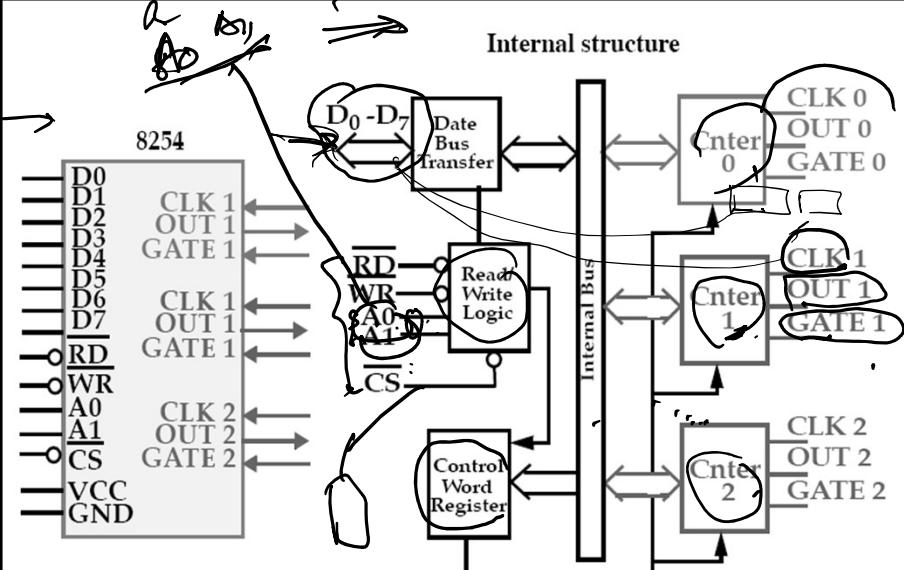
- Causes DRAM memory system to be refreshed.

Programmed with 15us on the PC/XT.

- Provides a timing source to the internal speaker and other devices.

133

4.4.5 Bộ định thời lập trình được 8254



134

67



4.4.5 Bộ định thời lập trình được 8254

8254 Pin Definitions

A_1, A_0 : The address inputs select one of the four internal registers with the 8254 as follows:

CLK : The clock input is the timing source for each of the internal counters.

It is often connected to the PCLK signal from the bus controller.

A_1	A_0	Function
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word

CS : Chip Select enables the 8254 for programming, and reading and writing.

G : The gate input controls the operation of the counter in some modes.

OUT : A counter output is where the wave-form generated by the timer is available.

RD/WR : Read/Write causes data to be read/written from the 8254 and often connects to the IORC/IOWC.

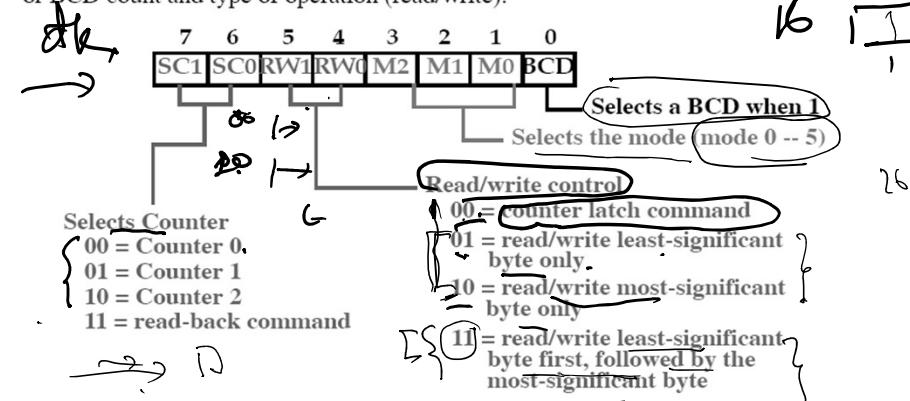


4.4.5 Bộ định thời lập trình được 8254

8254 Programming

Each counter is individually programmed by writing a control word, followed by the initial count.

The control word allows the programmer to select the counter, model of operation, binary or BCD count and type of operation (read/write).





4.4.5 Bộ định thời lập trình được 8254

8254 Programming

Each counter may be programmed with a count of 1 to FFFFH.

Minimum count is 1 all modes except 2 and 3 with minimum count of 2.

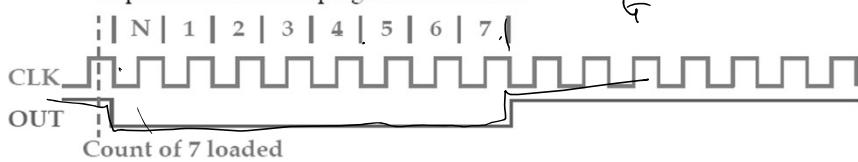
Each counter has a program control word used to select the way the counter operates.

If two bytes are programmed, then the first byte (LSB) stops the count, and the second byte (MSB) starts the counter with the new count.

There are 6 modes of operation for each counter:

Mode 0: An events counter enabled with G

The output becomes a logic 0 when the control word is written and remains there until N plus the number of programmed counts.



137



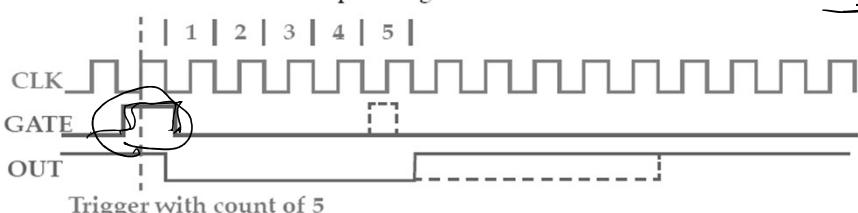
4.4.5 Bộ định thời lập trình được 8254

8254 Modes

Mode 1: One-shot mode

The G input triggers the counter to output a 0 pulse for $\text{Ocount} \text{ clocks}$.

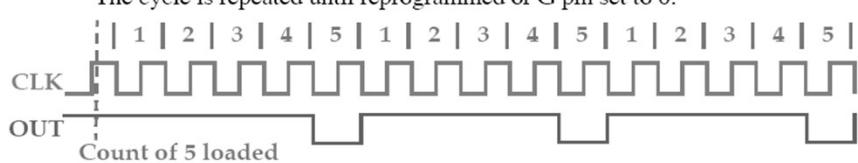
Counter reloaded if G is pulsed again.



Mode 2: Counter generates a series of pulses 1 clock pulse wide.

The separation between pulses is determined by the count.

The cycle is repeated until reprogrammed or G pin set to 0.



138

69

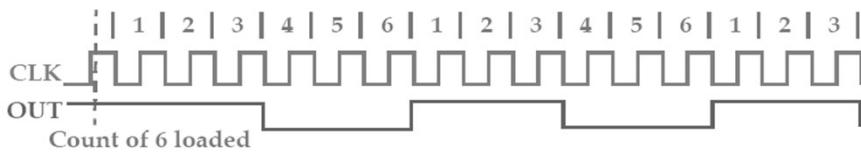


4.4.5 Bộ định thời lập trình được 8254

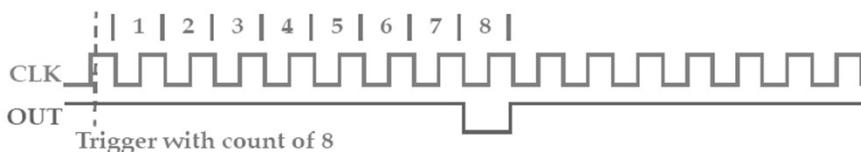
8254 Modes

○ Mode 3: Generates a continuous square-wave with G set to 1.

If count is even, 50% duty cycle otherwise OUT is high 1 cycle longer.



○ Mode 4: Software triggered one-shot (G must be 1).



○ Mode 5: Hardware triggered one-shot. G controls similar to Mode 1.



Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.4.1 Các kiểu ghép nối vào/ra

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A

4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279

4.4.5 Bộ định thời lập trình được 8254

4.4.6 Giao tiếp truyền thông lập trình được 16550

4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804

4.4.6 Giao tiếp truyền thông lập trình được 16550

141/Chapter

Programmable Communications Interface: 16550

A universal asynchronous receiver/transmitter (UART).

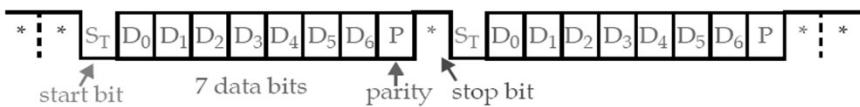
Operation speed: 0-1.5M Baud (Baud is # of bits transmitted/sec, including start, stop, data and parity).

Includes:

- A programmable Baud rate generator.
- Separate FIFO buffers for input and output data (16 bytes each).

Asynchronous serial data:

Transmitted and received without a clock or timing signal.



Two 10-bit frames of asynchronous data.

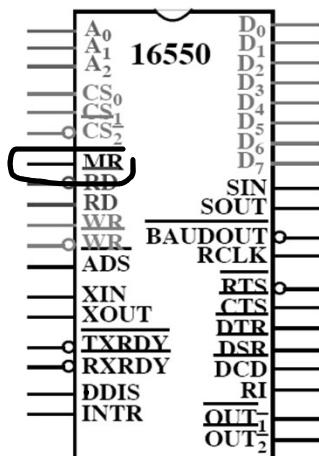
7- or 8- bit ASCII, e.g. w or w/o parity, is possible.

141

4.4.6 Giao tiếp truyền thông lập trình được 16550

142/Chapter

Programmable Communications Interface: 16550



Two separate sections are responsible for data communications:

Receiver

Transmitter

Can function in:

simplex: transmit only

half-duplex: transmit and receive but not simultaneously

full-duplex: transmit and receive simultaneously

The 16550 can control a modem through DSR, DTR, CTS, RTS, RI and DCD.

In this context, the modem is called the *data set* while the 16550 is called the *data terminal*.

71

142

4.4.6 Giao tiếp truyền thông lập trình được 16550

143/Chapter

Pinout of the 16550

- A₀, A₁ and A₂: Select an internal register for programming and data transfer.

A ₂	A ₁	A ₀	Register
1	0	0	Receiver buffer (read) and transmitter holding (write)
1	0	1	Interrupt enable
0	1	0	Interrupt identification (read) and FIFO control (write)
0	1	1	Line control
1	0	0	Modem control
1	0	1	Line status
1	1	0	Modem status
1	1	1	Scratch

- ADS: Address strobe used to latch address and chip select. Not needed on Intel systems - connected to ground.
- BAUDOUT: Clock signal from Baud rate generator in transmitter
- CS₀, CS₁, CS₂: Chip selects
- CTS: Clear to send -- indicates that the modem or data set is ready to exchange information. (Used in half-duplex to turn the line around).

143

4.4.6 Giao tiếp truyền thông lập trình được 16550

144/Chapter

Pinout of the 16550

- D₇-D₀: The data bus pins are connected to the microprocessor data bus.
- DCD: The data carrier detect -- used by the modem to signal the 16550 that a carrier is present.
- DDIS: Disable driver output -- set to 0 to indicate that the microprocessor is reading data from the UART. Used to change direction of data flow through a buffer.
- DSR: Data set ready is an input to 16550 -- indicates that the modem (data set) is ready to operate.
- DTR: Data terminal ready is an output -- indicates that the data terminal (16550) is ready to function.
- INTR: Interrupt request is an output to the micro -- used to request an interrupt.
 - Receiver error
 - Data received
 - Transmit buffer empty
- MR: Master reset -- connect to system RESET
- OUT1, OUT2: User defined output pins for modem or other device.
- RCLK: Receiver clock -- clock input to the receiver section of the UART.
Always 16X the desired receiver Baud rate.

144

72

4.4.6 Giao tiếp truyền thông lập trình được 16550

145/Chapter

Pinout of the 16550

- RD, RD: Read inputs (either can be used) -- cause data to be read from the register given by the address inputs.
- RI: Ring indicator input -- set to 0 by modem to indicate telephone is ringing.
- RTS: Request-to-send -- signal to modem, indicating UART wishes to send data.
- SIN, SOUT: Serial data pins, in and out.
- RXRDY: Receiver ready -- used to transfer received data via DMA techniques.
- TXRDY: Transmitter ready -- used to transfer transmitter data via DMA.
- WR, WR: Write (either can be used) -- connects to micro write signal to transfer commands and data to 16550.
- XIN, XOUT: Main clock connections - a crystal oscillator can be used.

145

4.4.6 Giao tiếp truyền thông lập trình được 16550

146/Chapter

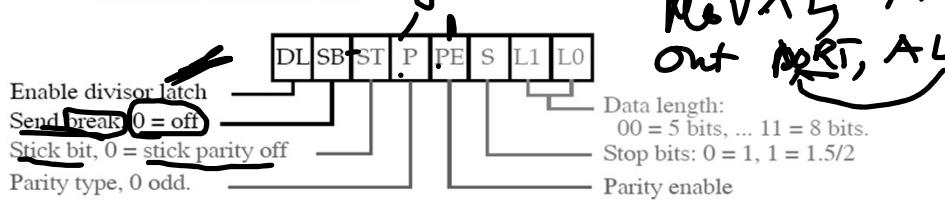
Programming the 16550

Two phases: Initialization, operation.

Initialization:

After RESET, the *line control register* and *baud rate generator* need to be programmed.

Line control register sets the # of data bits, # of stop bits and the parity.
Addressed at location 011.



■ Stop bits: S = 1, 1.5 stop bits used for 5 data bits, 2 used for 6, 7 or 8.

146

73

4.4.6 Giao tiếp truyền thông lập trình được 16550

147/Chapter

Programming the 16550

Initialization (cont.)

- ST, P and PE used to send even or odd parity, to send no parity or to send a 1 or a 0 in the parity bit position for all data.

ST	P	PE	Function
0	0	0	No parity
0	0	1	Odd parity
0	1	0	No parity
0	1	1	Even parity
1	0	0	Undefined
1	0	1	Send/receive 1
1	1	0	Undefined
1	1	1	Send/receive 0

No parity, both 0 -- used for internet connections.

- SB = 1 causes a break to be transmitted on SOUT.

A break is at least two frame of 0 data.

- DL = 1 enables programming of the baud rate divisor.

4.4.6 Giao tiếp truyền thông lập trình được 16550

148/Chapter

Programming the 16550

Initialization (cont.)

Baud rate generator is programmed with a divisor that sets baud rate of transmitter.

Baud rate generator is programmed at 000 and 001.

Port 000 used to hold least significant byte, 001 most significant.

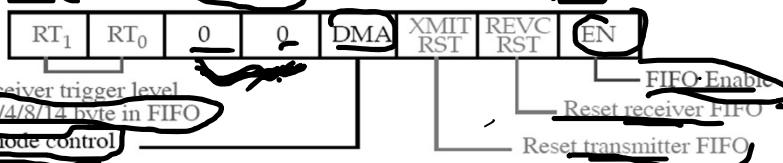
16

Value used depends on external clock/crystal frequency.

For 18.432MHz crystal, 10,473 gives 110 baud rate, 30 gives 38,400 baud.

Note, number programmed generates a clock 16X the desired Baud rate.

Last, the FIFO control register must be programmed at 010.

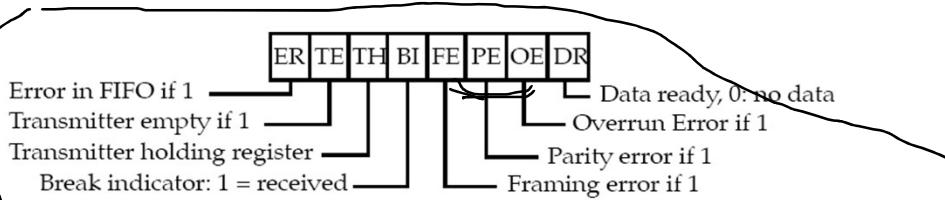


4.4.6 Giao tiếp truyền thông lập trình được 16550

Programming the 16550

Operation:

Status line register gives information about error conditions and state of the transmitter and receiver.



This register needs to be tested in software routines designed to use the 16550 to transmit/receive data.

Suppose a program wants to send data out SOUT.

It needs to poll the **TH** bit to determine if transmitter is ready to receive data.

To receive information, the **DR** bit is tested.

4.4.6 Giao tiếp truyền thông lập trình được 16550

Programming the 16550

Operation:

It is also a good idea to check for errors.

Parity error: Received data has wrong error -- transmission bit flip due to noise.

Framing error: Start and stop bits not in their proper places.

This usually results if the receiver is receiving data at the incorrect baud rate.

Overrun error: Data has overrun the internal receiver FIFO buffer.

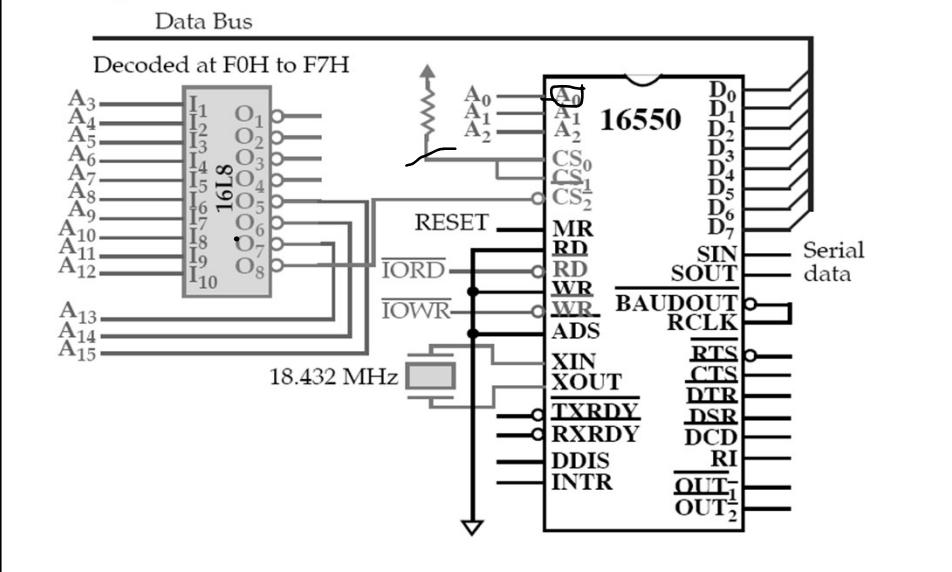
Software is failing to read the data from the FIFO.

Break indicator bit: Software should check for this as well, i.e. two consecutive frames of 0s.

4.4.6 Giao tiếp truyền thông lập trình được 16550

151/Chapter

Example of 16550



151

4.4.6 Giao tiếp truyền thông lập trình được 16550

152/Chapter

Serial Port

Most PC interfaces for serial data exchange comply with the RS-232C standard.

This standard defines the mechanical, electrical and logical interface for asynchronous data transfer between the data terminal equipment (DTE: Computer) and the data carrier equipment (DCE: modem, other computer etc.)

The 16550 or similar UART devices are used to perform the complex handshaking defined by the standard

The RS-232C standard defines 25 lines between DTE and DCE, but most are reserved for synchronous data transfer

For serial, asynchronous data exchange only 11 RS-232C signals are required

IBM defined a 9-pin connection for its serial port, which is the standard serial port found on most PCs today.

152

76



4.4.6 Giao tiếp truyền thông lập trình được 16550

153/Chapter

Serial Port

The RS-232C signals are similar to the UART signals discussed before

- RTS (Request to send)
- CTS (Clear to send)
- DCD (Data carrier detect)
- DSR (Data set ready)
- DTR (Data terminal ready)
- RI (Ring indicator)
- TD (Transmitted data)
- RD (Received data)

Can operate in simplex, half-duplex and full-duplex modes

Mostly used for connections to modems

Also used for serial printers

Null-modem connection can be used to transfer data between two DTEs.

Identified as the COM port in PCs.

153

4.4.6 Giao tiếp truyền thông lập trình được 16550

154/Chapter

Parallel Port

In a PC usually known as the LPT (line printer) port

The connection between the port and the printer is created by a 'Centronics' cable.

Named after the company that created the first printer interface standard

The centronics cable uses 36 wires, 18 of which are ground

As only 18 are required to communicate with the printer, IBM defined a 25 pin connector

Data is transferred using a 8-bit data register, other pins are used for handshaking and detecting errors

The status register is updated by the printer using dedicated signals on the connector and read by the PC to determine the printer status

The control register can be read or written by the PC and controls the operation of the printer

154

77



4.4.6 Giao tiếp truyền thông lập trình được 16550

Parallel Port

The parallel port signals are given below:

- $\overline{\text{STR}}$: A logic low transfers data to the printer
- D0-D7: Data bits 0 through 7
- $\overline{\text{ALF}}$: Logic low signals an auto line feed after every line
- $\overline{\text{INI}}$: Logic low initializes the printer
- $\overline{\text{ACK}}$: Acknowledge signal from the printer when data is transferred
- $\overline{\text{DSL}}$: Logic low selects the printer
- BSY: When active, indicates the printer is busy and cannot accept more data
- PAP: High level shows that paper is about to run out
- OFON: High level shows that the printer is on-line
- $\overline{\text{ERR}}$: Signals printer errors

An improved parallel port standard was defined by the IEEE: IEEE-1248

This is the port found in most modern PCs

Uses the same old centronics interface, has both 25 and 36 pin interfaces defined, but signal names are assigned according to the mode of operation.

155



4.4.6 Giao tiếp truyền thông lập trình được 16550

Parallel Port

Five modes of operation are defined:

- *Compatible Mode*: defined for backward compatibility with the old unidirectional model, also known as *SPP* (standard parallel port)
- *Byte Mode*: bidirectional centronics mode, 8 bits wide
- *Nibble Mode*: defines the minimum characteristics for a parallel port, data is transferred in nibbles (4-bits)
- *Extended Parallel Port (EPP)*: bidirectional data transfer, and also addresses, for a maximum of 256 units.
- *Enhanced Capability Mode (ECP)*: same as EPP, but uses data compression, FIFO with DMA and interrupt capability and command cycles, 128 maximum units

156

78

Chương 4: Tổ chức vào ra dữ liệu

4.1 Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288

4.2 Ghép nối 8088 với bộ nhớ

4.3 Ghép nối 8086 với bộ nhớ

4.4 Ghép nối với thiết bị ngoại vi

4.4.1 Các kiểu ghép nối vào/ra

4.4.2 Giải mã địa chỉ cho các thiết bị vào/ra

4.4.3 Mạch ghép nối vào ra song song lập trình được 8255A

4.4.4 Mạch điều khiển bàn phím/màn hình lập trình được 8279

4.4.5 Bộ định thời lập trình được 8254

4.4.6 Giao tiếp truyền thông lập trình được 16550

4.4.7 Bộ biến đổi số tương tự DAC0830 và bộ biến đổi tương tự số ADC0804

157

4.4.7.1 Bộ biến đổi số tương tự DAC

Digital-to-Analog (DAC) Converters

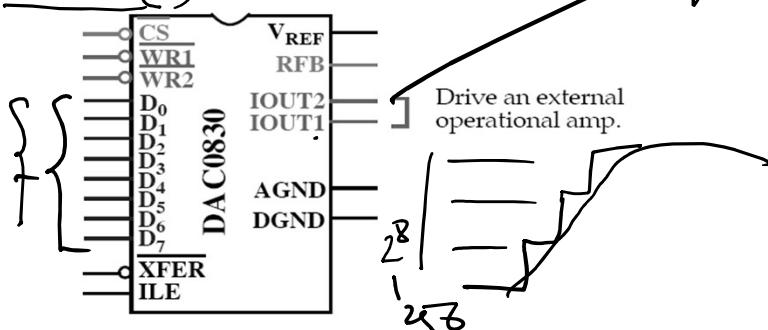
Used to convert between analog and digital data.

For example, the DAC 0830 (National Semi Corp.) is an 8-bit DAC that transforms an 8-bit binary number to an analog voltage.

8-bit yields 256 different analog voltages.

10-bit, 12-bit and 16-bit are also available.

Conversion time is 1μs.



158

79

4.4.7.1 Bộ biến đổi số tương tự DAC

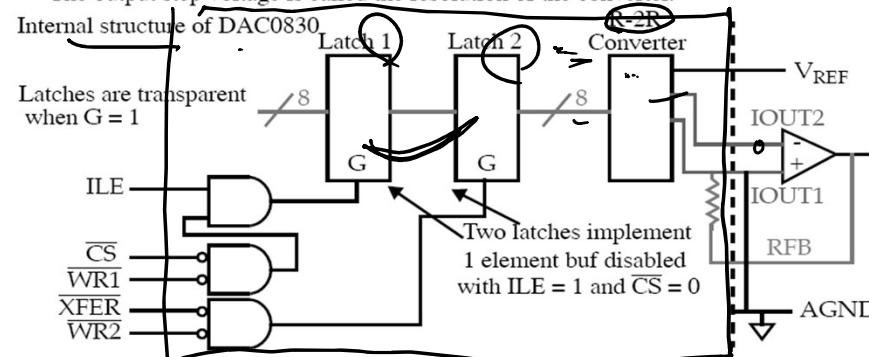
Digital-to-Analog (DAC) Converters

8-bit digital value drives D_0 through D_7 .

The outputs are I_{OUT1} and I_{OUT2} .

The output step voltage is defined by $-V_{REF}$ (reference voltage), divided by 255, e.g. if $V_{REF} = -5.0V$, then the output step voltage is $+0.0196$.

The output step voltage is called the resolution of the converter.

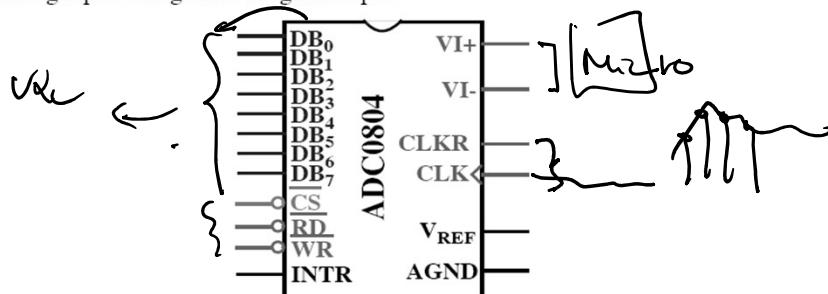


159

4.4.7.2 Bộ biến đổi tương tự số ADC

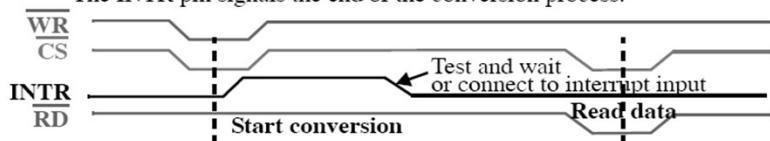
Analog-to-Digital (ADC) Converters

The ADC0804 is an 8-bit analog-to-digital converter that requires up to 100us to convert an analog input voltage into a digital output.



To start conversion, \overline{WR} is pulsed with \overline{CS} at GND.

The INTR pin signals the end of the conversion process.



160

80

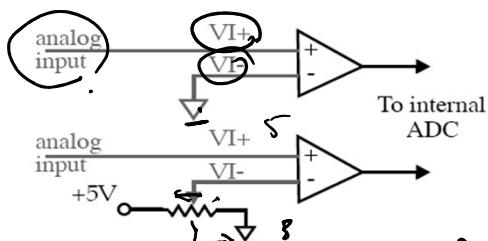


4.4.7.2 Bộ biến đổi tương tự số ADC

Analog-to-Digital (ADC) Converters

VI- and VI+ are connected to an internal operational amplifier.

To sense a 0 to +5V input.

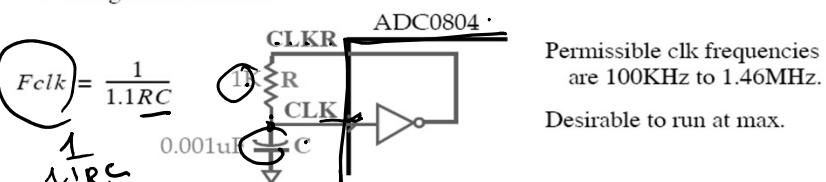


To sense an input offset from GND.

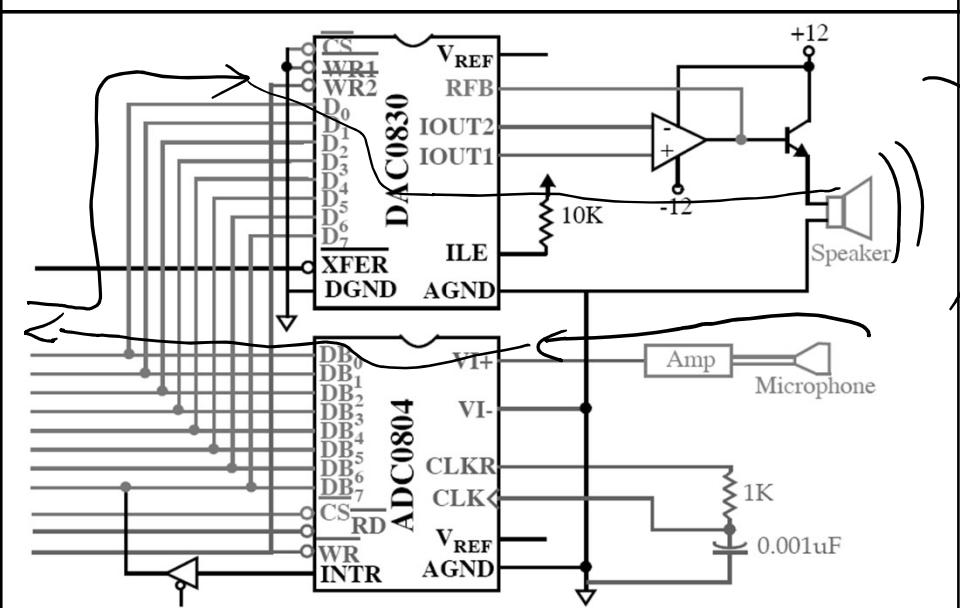
The ADC0804 requires a clock, generated either with:

An external clock applied to the CLK pin.

Using an RC circuit.



4.4.7.2 Bộ biến đổi tương tự số ADC





Nội dung môn học

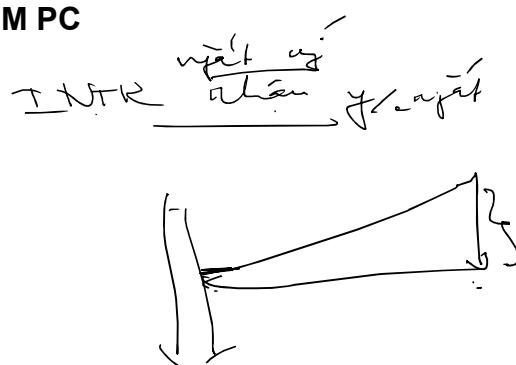
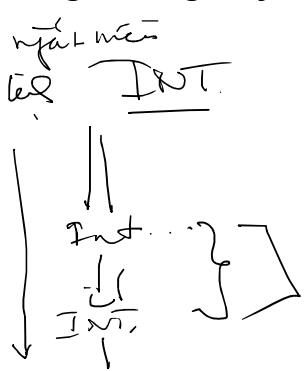
1. Giới thiệu chung về hệ vi xử lý
2. Bộ vi xử lý Intel 8088/8086
3. Lập trình hợp ngữ cho 8086
4. Tổ chức vào ra dữ liệu
5. Ngắt và xử lý ngắt
6. Truy cập bộ nhớ trực tiếp DMA
7. Các bộ vi xử lý trên thực tế

1



Chương 5: Ngắt và xử lý ngắt

- 5.1 Giới thiệu về ngắt
- 5.2 Đáp ứng của CPU khi có yêu cầu ngắt
- 5.3 Xử lý ưu tiên ngắt
- 5.4 Mạch điều khiển ngắt ưu tiên 8259A
- 5.5 Ngắt trong máy tính IBM PC



2

1



5.1 Giới thiệu về ngắt



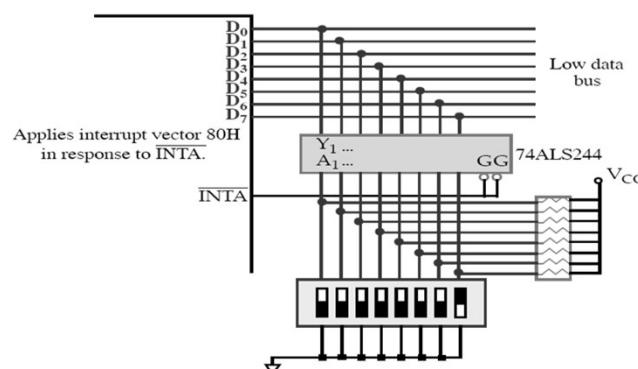
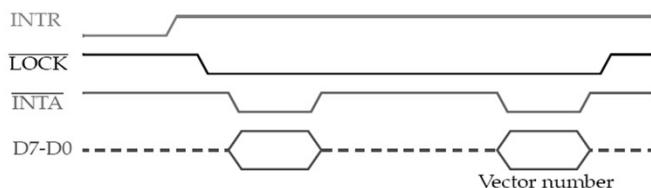
- 2 loại ngắt:**

- Ngắt cứng:** tín hiệu yêu cầu ngắt từ
 - ⇒ NMI (ngắt không che được)
 - ✓ Lỗi chẩn lẻ và các lỗi hệ thống nghiêm trọng khác (ví dụ: mất nguồn)
 - ⇒ và INTR (ngắt che được)
- Ngắt mềm:** CPU thực hiện các lệnh ngắt INT N, $0 \leq N \leq 255$
 - ⇒ ngắt 0 đến 31 dành riêng cho Intel
 - ⇒ ngắt 32 đến 255 dành cho người sử dụng

3

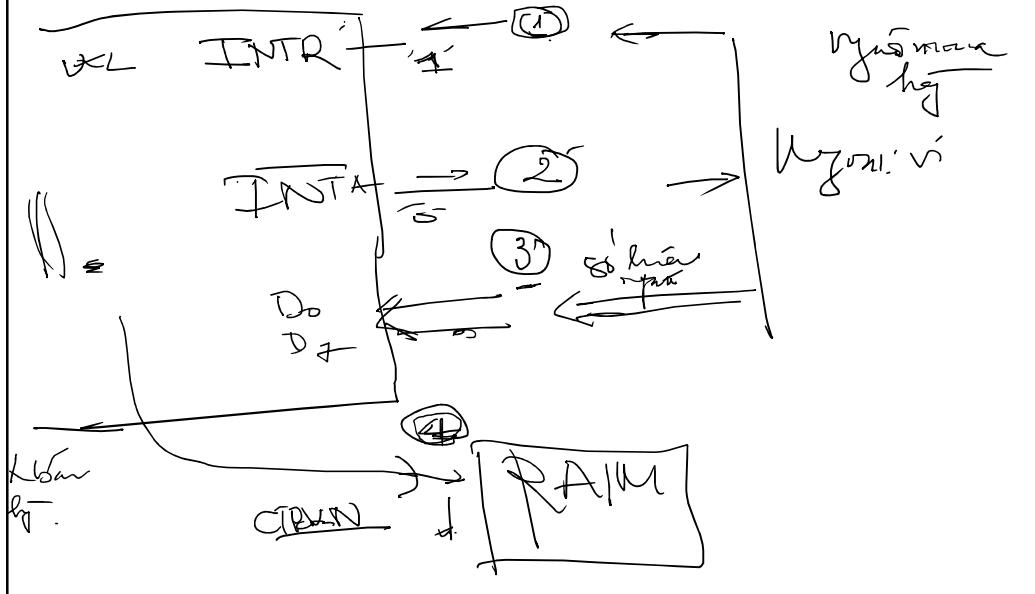


5.1 Giới thiệu về ngắt



4

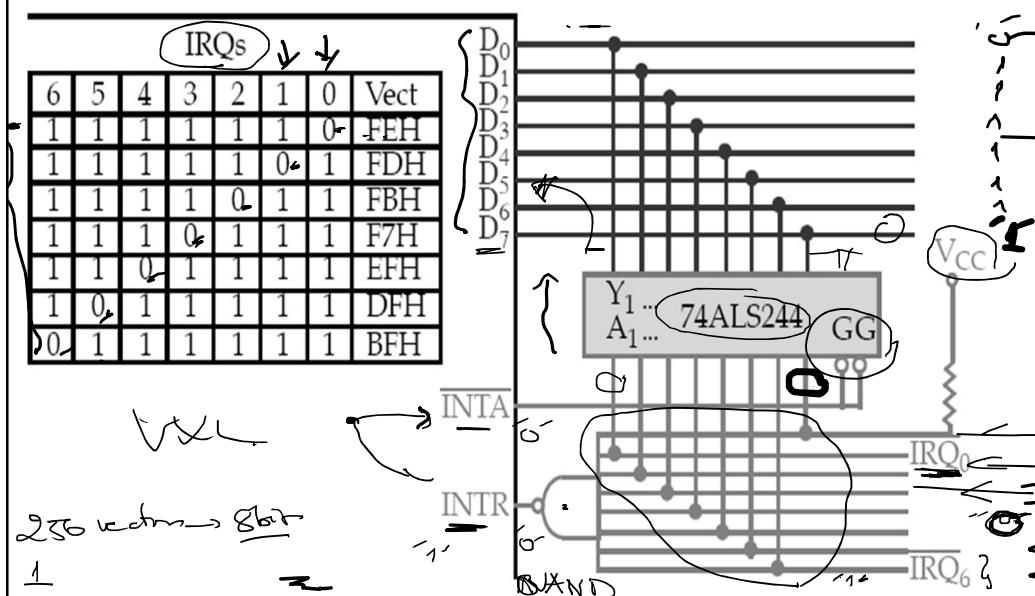
2



5



5.1 Giới thiệu về ngắt

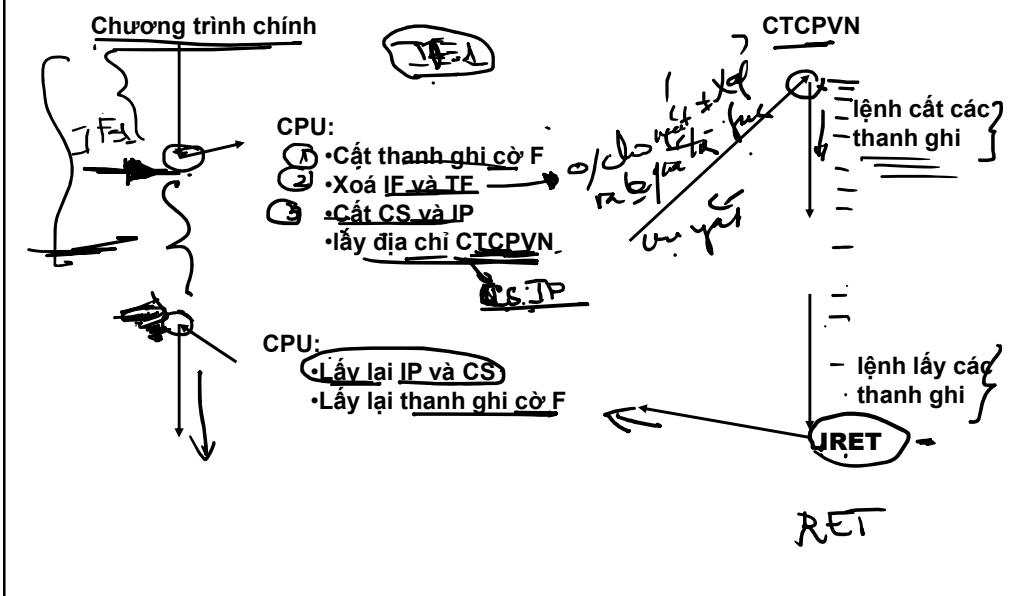


6

3



5.2 Đáp ứng của CPU khi có yêu cầu ngắt



7



IF = 0, TF = 0.

Đang

8

4



5.3 Xử lý ưu tiên ngắt

- Ngắt có mức ưu tiên cao nhất sẽ được phục vụ trước
- Các mức ưu tiên:
 - Ngắt nội bộ: INT 0, INT N
 - Ngắt không che được: NMI
 - Ngắt che được INTR
 - Ngắt để chạy từng lệnh INT 1
- CPU sẽ xử lý thế nào nếu CPU đang thực hiện phép chia và số chia bằng 0 đồng thời có yêu cầu ngắt từ chân INTR?

9

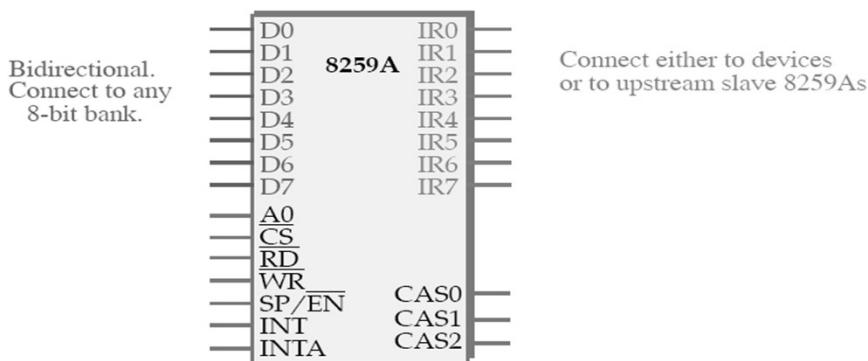


5.4 Mạch điều khiển ngắt 8259A

8259A Programmable Interrupt Controller

The 8259A adds 8 vectored priority encoded interrupts to the microprocessor.

It can be expanded to 64 interrupt requests by using one master 8259A and 8 slave units.

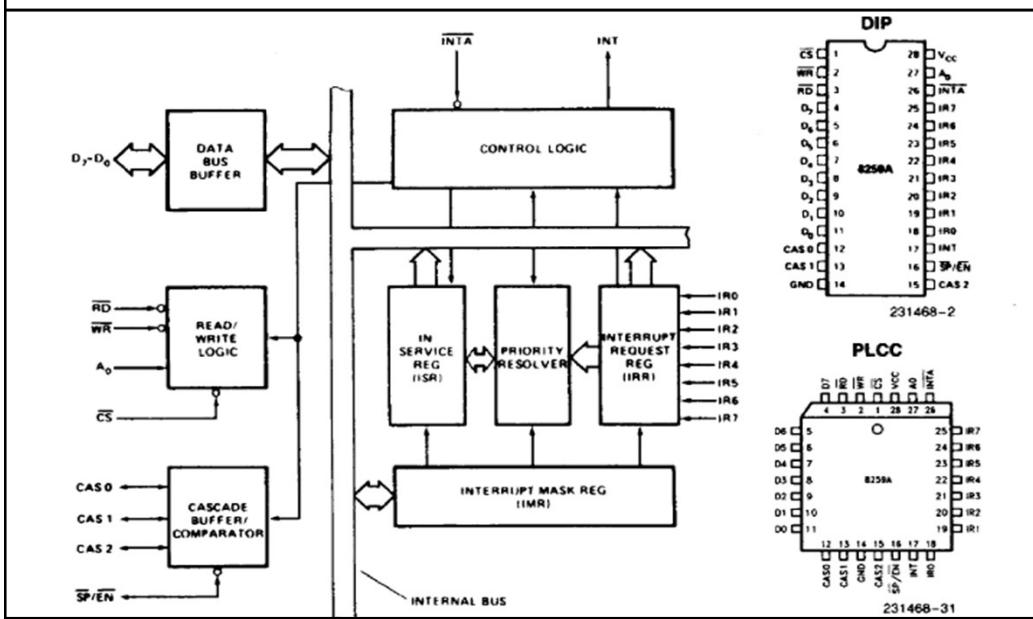


\overline{CS} and \overline{WR} must be decoded. Other connections are direct to microprocessor.

10



5.4 Mạch điều khiển ngắt 8259A



11



5.4 Mạch điều khiển ngắt 8259A

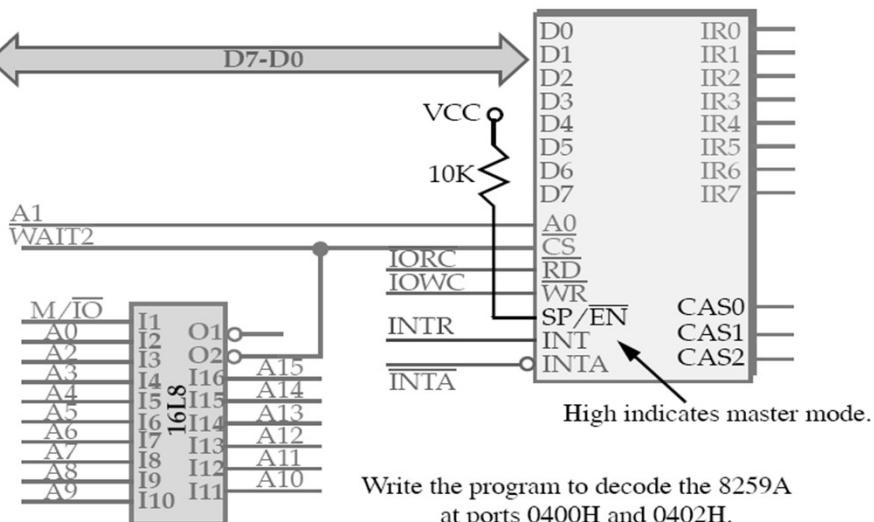
- WR
Connects to a write strobe signal (one of 8 for the Pentium).
- RD
Connects to the IORC signal.
- INT
Connects to the INTR pin on the microprocessor.
- INTA
Connects to the INTA pin on the microprocessor.
- A₀
Selects different command words in the 8259A.
- CS
Chip select - enables the 8259A for programming and control.
- SP/EN
Slave Program (1 for master, 0 for slave)/Enable Buffer (controls the data bus transivers when in buffered mode).
- CAS2-CAS0
Used as outputs from the master to the slaves in cascaded systems.

12



5.4 Mạch điều khiển ngắt 8259A

A single 8259A connected in the 8086.



Write the program to decode the 8259A
at ports 0400H and 0402H.

13



5.4 Mạch điều khiển ngắt 8259A

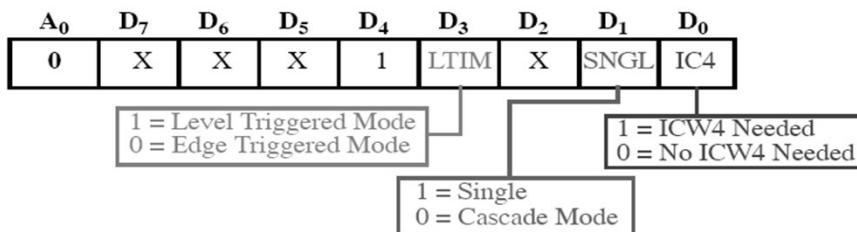
Programmed by *Initialization (ICWs)* and *Operation (OCWs)* Command Words.

There are 4 ICWs.

At power-up, ICW1, ICW2 and ICW4 must be sent.

If ICW1 indicates cascade mode, then ICW3 must also be sent.

○ ICW1:



LTIM indicates if IRQ lines are positive edge-triggered or level-triggered.

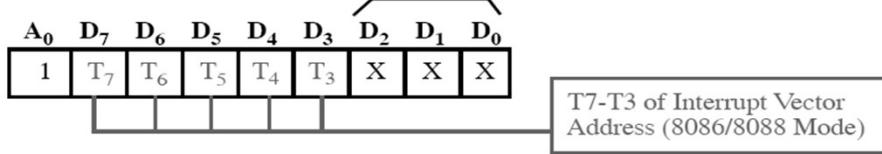
14



5.4 Mạch điều khiển ngắt 8259A

○ ICW2:

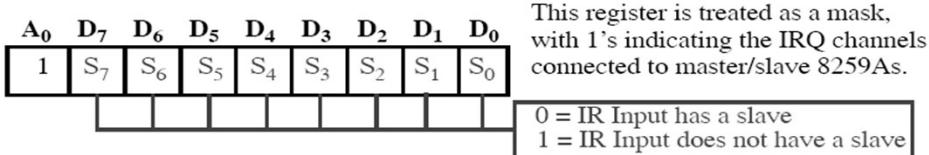
Low order bits are 0 since there are 8 interrupts.



These bits determine the vector numbers used with the IRQ inputs.

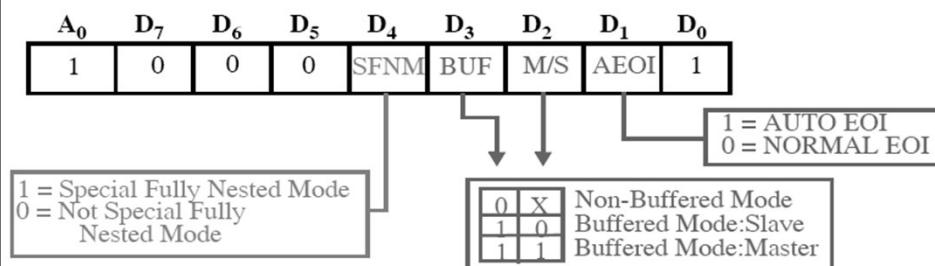
For example, if programmed to generate vectors 08H-0FH, 08H is placed into these bit positions.

○ ICW3:



5.4 Mạch điều khiển ngắt 8259A

○ ICW4:



Fully nested mode allows the highest-priority interrupt request from a slave to be recognized by the master while it is processing another interrupt from a slave.

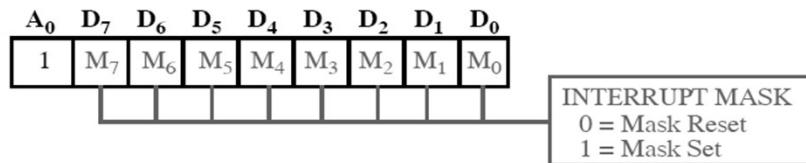
AEOI, if 1, indicates that an interrupt automatically resets the interrupt request bit, otherwise OCW2 is consulted for EOI processing.



5.4 Mạch điều khiển ngắt 8259A

The Operation Command Words (OCWs) are used to direct the operation of the 8259A.

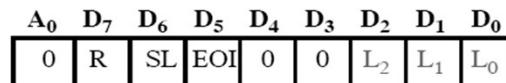
○ OCW1:



OCW1 is used to read or set the interrupt mask register.

If a bit is set, it will turn off (mask) the corresponding interrupt input.

○ OCW2:



Only programmed when the AEOI mode in ICW4 is 0.

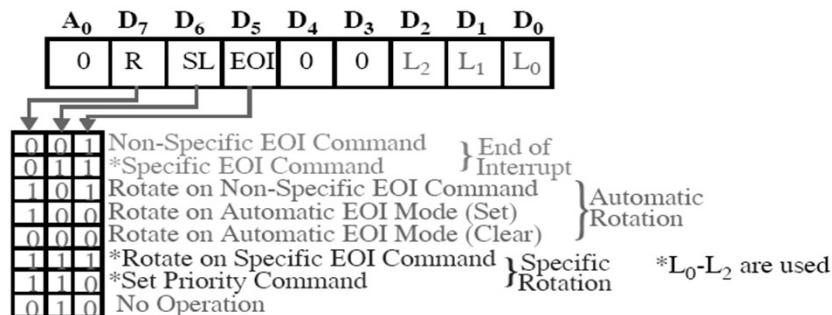
Allows you to control priorities after each interrupt is processed.

17



5.4 Mạch điều khiển ngắt 8259A

○ OCW2:



Non-specific EOI: Here, the ISR sets this bit to indicate EOI. The 8259A automatically determines which interrupt was active and re-enables it and lower priority interrupts.

Specific EOI: ISR resets a specific interrupt request given by L₂-L₀.

Rotate commands cause priority to be rotated w.r.t. the current one being processed.

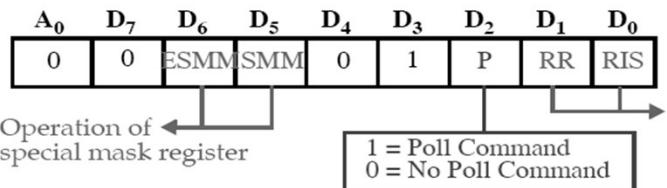
Set priority: allows the setting of the lowest priority interrupt (L₂-L₀).

9

18

5.4 Mạch điều khiển ngắt 8259A

- OCW3:



Indicates which status register, IRR or ISR, is to be read.

If polling is set, the next read operation will read the poll word.

If the leftmost bit is set in the poll word, the rightmost 3 bits indicate the active interrupt request with highest priority.

Allows ISR to service highest priority interrupt.

There are three status registers, Interrupt Request Register (IRR), In-Service Register (ISR) and Interrupt Mask Register (IMR).

- IRR: Indicates which interrupt request lines are active.
 - ISR: Level of the interrupt being serviced.
 - IMR: A mask that indicates which interrupts are on/off.

5.4 Mạch điều khiển ngắt 8259A

ISR update procedure with rotating priority configured.



5.4 Mạch điều khiển ngắt 8259A

Interfacing 16550 UART using 8259A

In the following configuration the 16550 is connected to the 8259A through IR0.

An interrupt is generated, if enabled through the interrupt control register, when either:

- The transmitter is ready to send another character.
- The receiver has received a character.
- An error is detected while receiving data.
- A modem interrupt occurs.

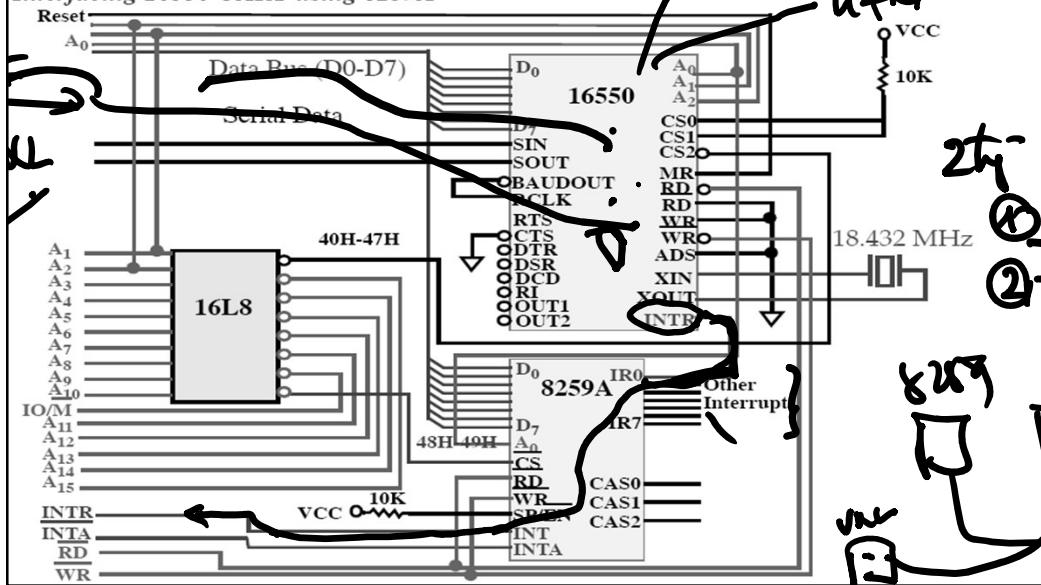
The 16550 is decoded at 40H and 47H.

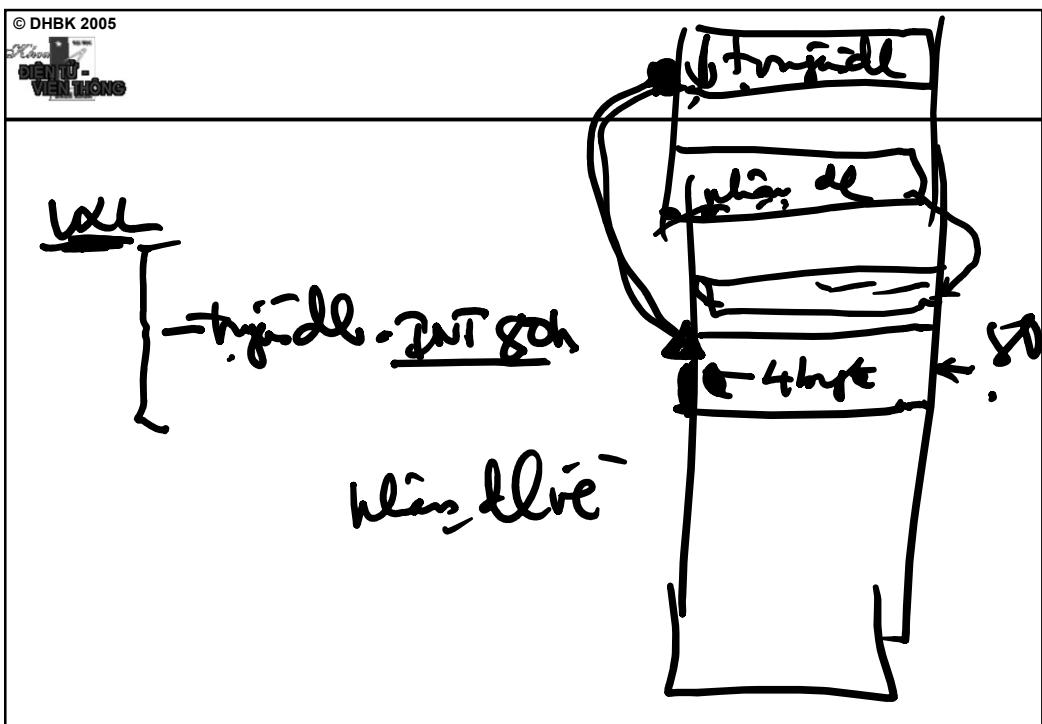
The 8259A is decoded at 48H and 49H.



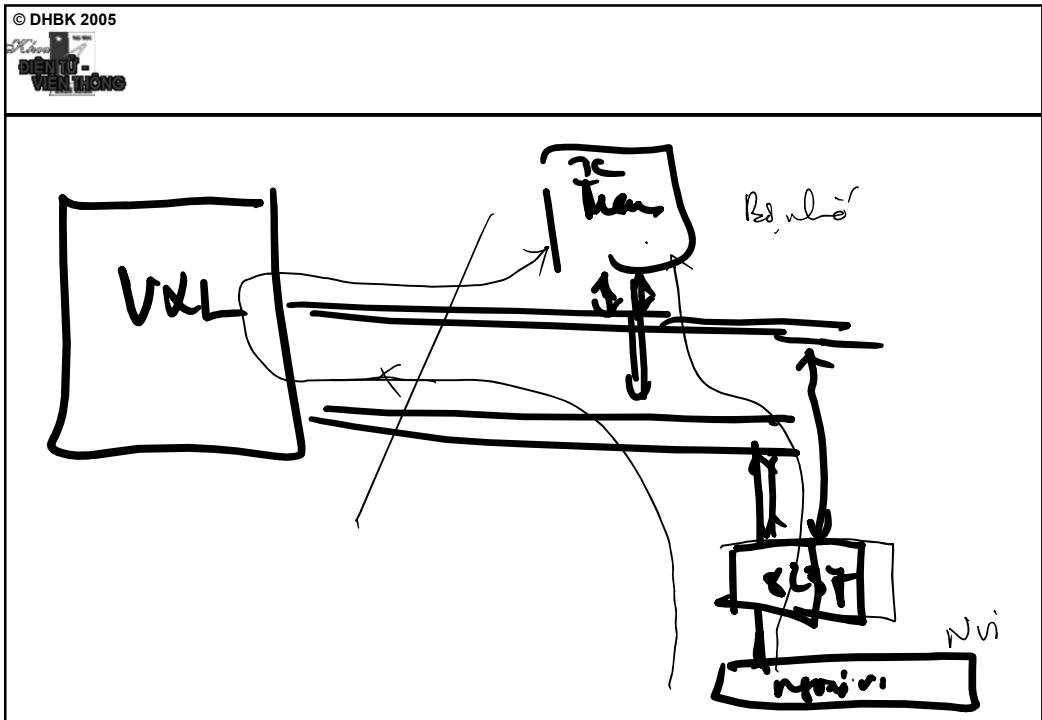
5.4 Mạch điều khiển ngắt 8259A

Interfacing 16550 UART using 8259A





23



24

12

© DHBK 2005
 Khoa Công nghệ
 Điện tử -
 Viễn thông

bằng chiaⁿ
 (2)

IC RAM duy nhất 16k x 8bit
 PCRAM & Exram

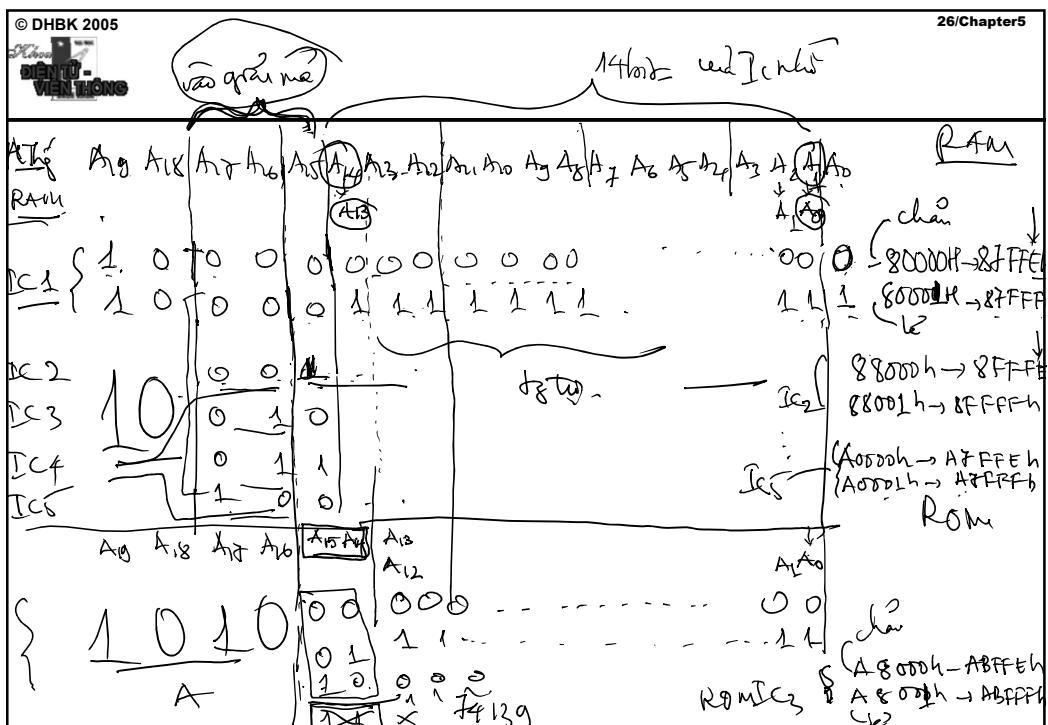
Ghép chia và 8086 160 kB SRAM + 48 kB ROM;
 - RAM: cần 8F IC = $\frac{160\text{ k}}{16\text{ k}} = 10\text{ IC}$ → 5 bộ chiaⁿ - $\frac{80\text{ bit}}{8\text{ bit}} \rightarrow$ giả mạo
 chia IC
 $\log_2 5 = 2.30$
 (3) kbit

- Số ROM cần thiết là $\frac{64\text{ k}}{8\text{ k}} = 8\text{ IC}$ → 3 bộ chiaⁿ → 2 kbyte
 chia IC
 RAM

RAM bắt đầu từ 80000h → $\frac{1A0000h}{2^4} = 14$ bit
 $\Rightarrow A_{13}$

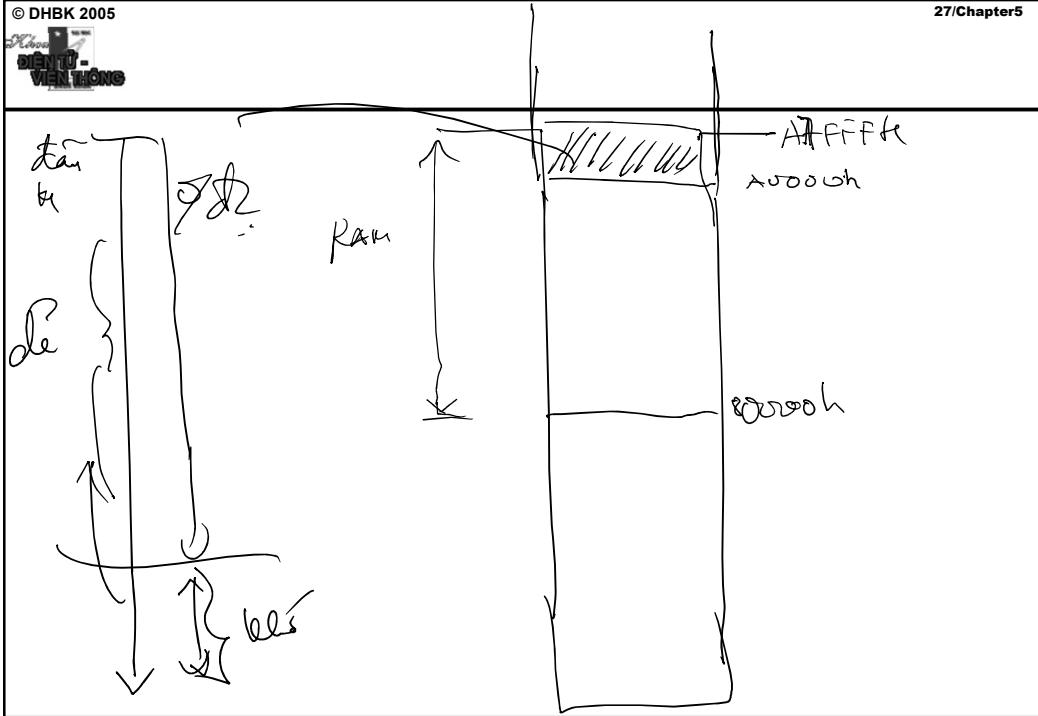
ROM → $\frac{16.2^{10}}{\log_2 8.2^{10}} = 14$ bit
 $\log_2 8.2^{10} = 13$ bit
 $\Rightarrow A_0 - A_2$

25

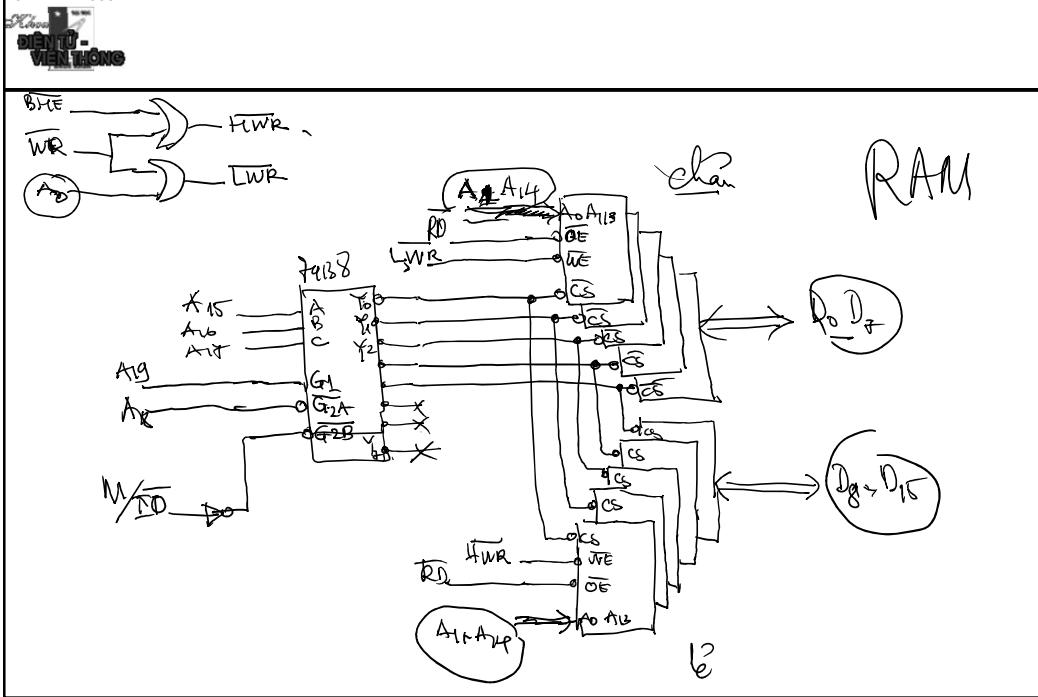


26

13

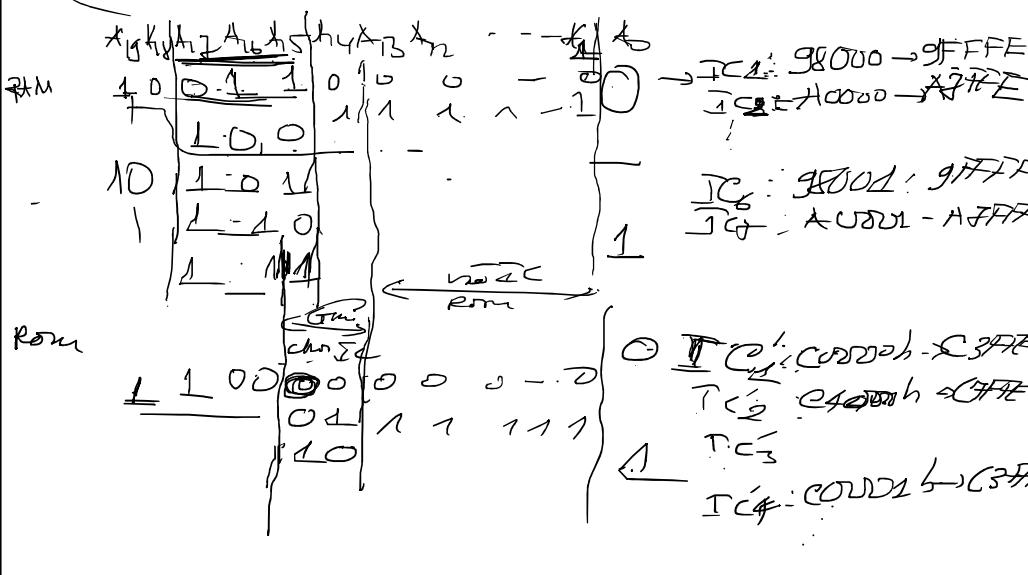
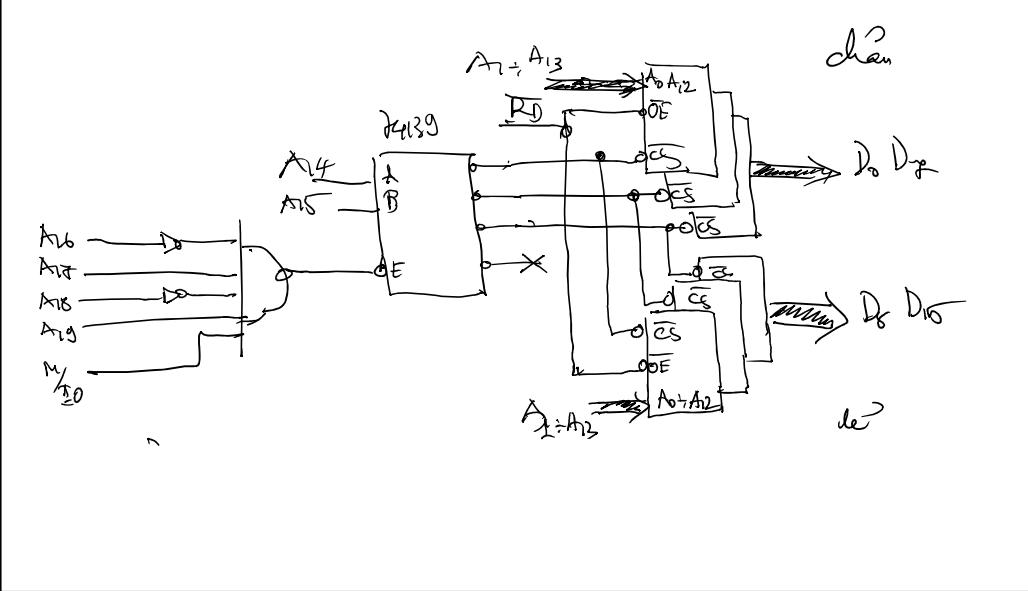


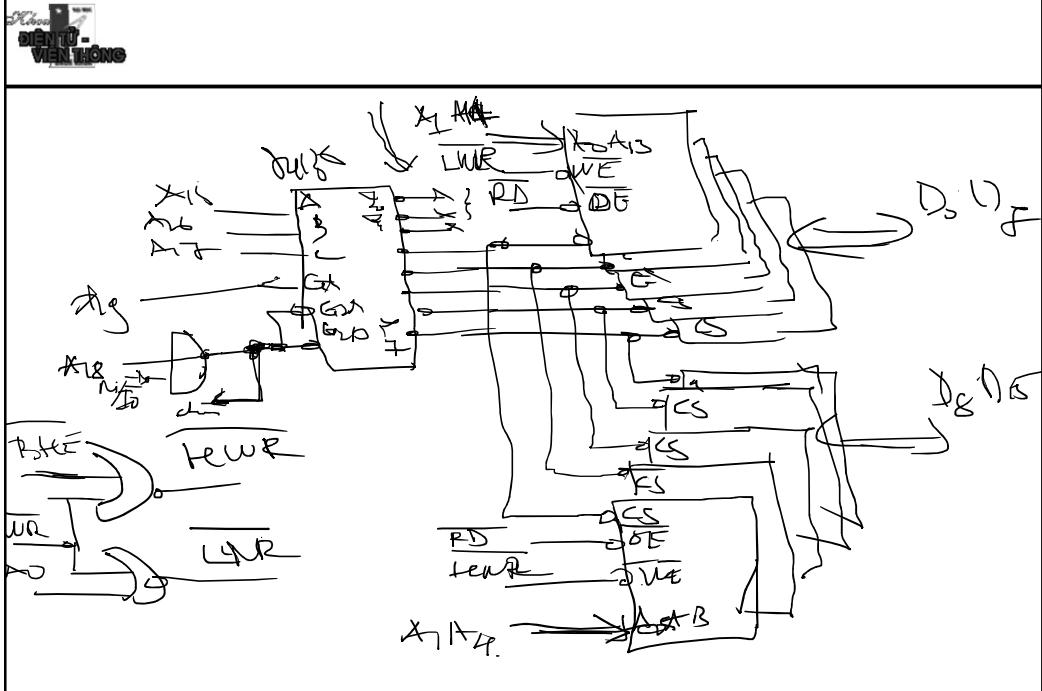
27



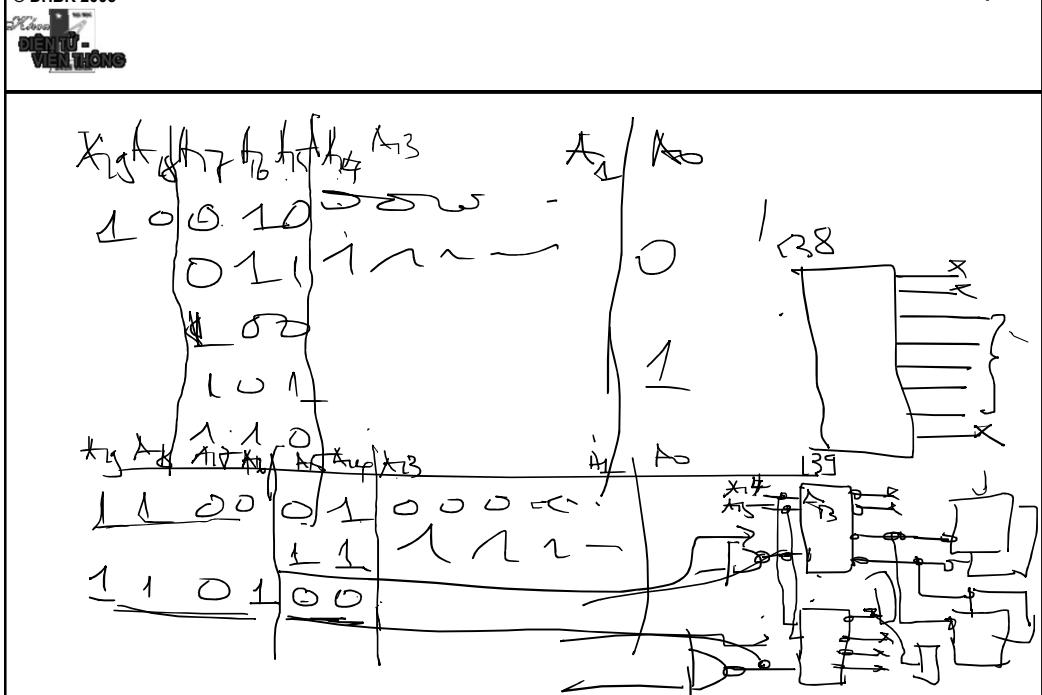
28

14

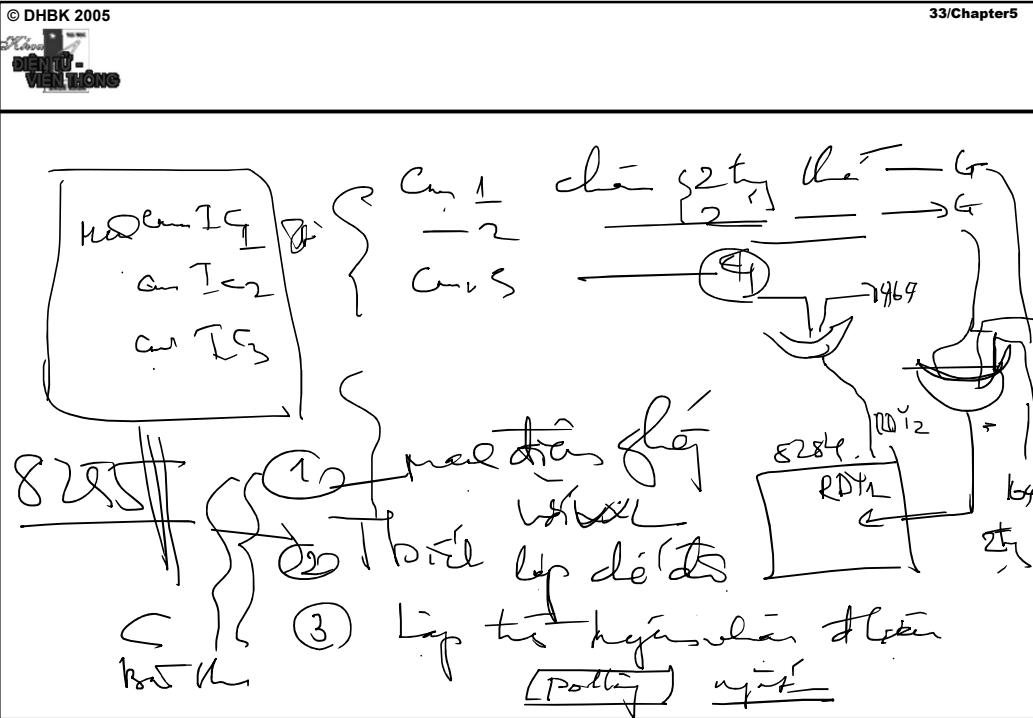




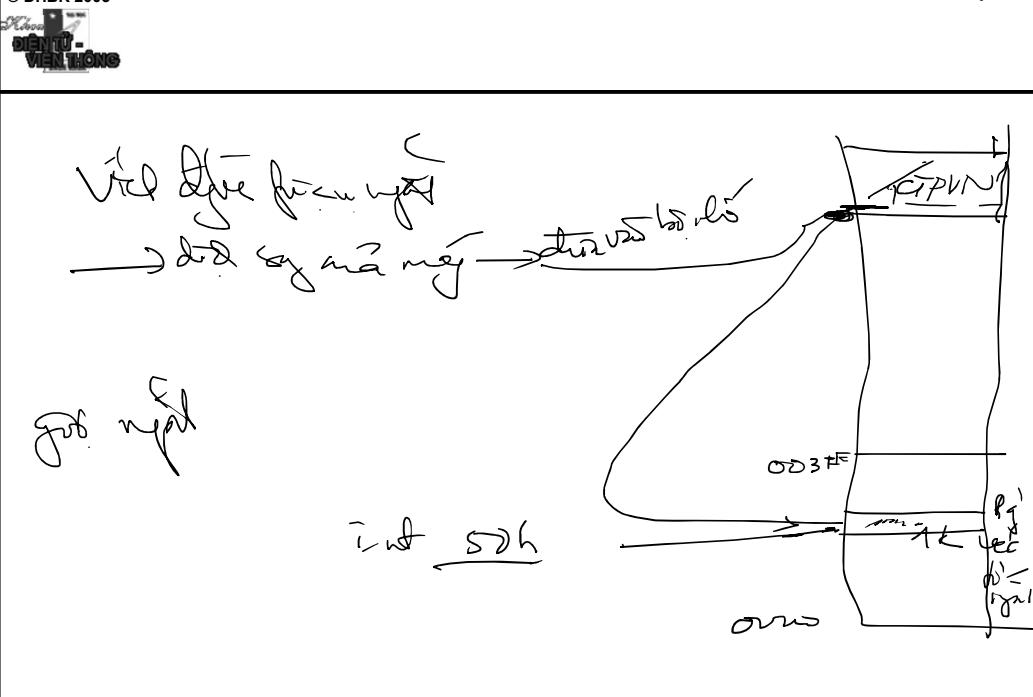
31



32

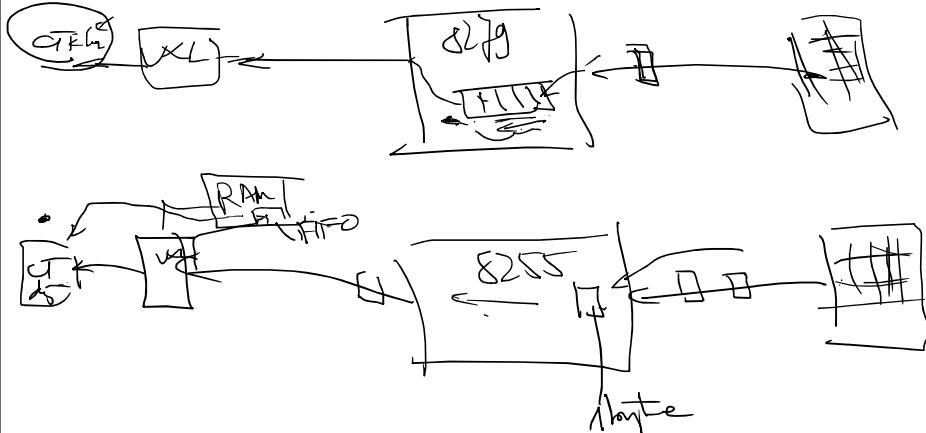


33



34

17



Bảng 1: Độ sâu IC,
 - Tổng số bit truyền vào IC
 - Tổng số bit được gom ra
 BC 2: vết dấu của dữ liệu và độ sâu IC, Văn
 b) gom mảng
 BC 3: vẽ mảng đến
Lap Sau thì gõ 8255 và VGA
thì sau để đọc lần, Lap thì hết

© DHBK 2005



Nội dung môn học

1. Giới thiệu chung về hệ vi xử lý
2. Bộ vi xử lý Intel 8088/8086
3. Lập trình hợp ngữ cho 8086
4. Tổ chức vào ra dữ liệu
5. Ngắt và xử lý ngắt
6. Truy cập bộ nhớ trực tiếp DMA
7. Các bộ vi xử lý trên thực tế

1

1



Chương 6: Truy cập bộ nhớ trực tiếp DMA

- 6.1 Giới thiệu về DMA
- 6.2 Mạch DMAC 8237A của Intel

2

2

© DHBK 2005



6.1 Giới thiệu về DMA

Direct Memory Access (DMA)

An alternative to the basic and interrupt-driven I/O discussed previously.

DMA allows data to be transferred between memory and the I/O device without processor intervention.

Speed of transfer limited to speed of memory components or DMA controller (up to 32-40 Mbytes/sec).

Common DMA operations:

- DRAM refresh
- Video refresh
- Disk-memory system reads and writes.

Two signals are used to request/ack a DMA transfer:

- HOLD is an input to the micro that requests a DMA action.
- HLDA is an output from the micro granting the DMA action.

The microprocessor responds by suspending the execution of the program and by placing its address, data and control bus in high-impedance states.

3

© DHBK 2005



6.2 Mạch DMAC 8237A của Intel

Direct Memory Access (DMA)

DMA 'reads' refer to transfers from memory to an I/O device and involves the use of MRDC and IOWC.

DMA 'writes' refer to transfers from an I/O device to memory and involves the use of MWTC and IORC.

The data transfer rate is determined by the speed of memory or the DMA controller (usually the latter).

The DMA controller provides memory with the address and select the appropriate I/O device (via DACK).

IOR	A ₇
IOW	A ₆
MEMR	A ₅
MEMW	A ₄
x	EOP
READY	A ₃
HLDA	A ₂
ADSTB	A ₁
AEN	A ₀
HRQ	V _{CC}
CS	DB ₀
CLK	DB ₁
RESET	DB ₂
DACK ₂	DB ₃
DACK ₃	DB ₄
DREQ ₃	DACK ₀
DREQ ₂	DACK ₁
DREQ ₁	DB ₅
DREQ ₀	DB ₆
V _{SS}	DB ₇

4

© DHBK 2005



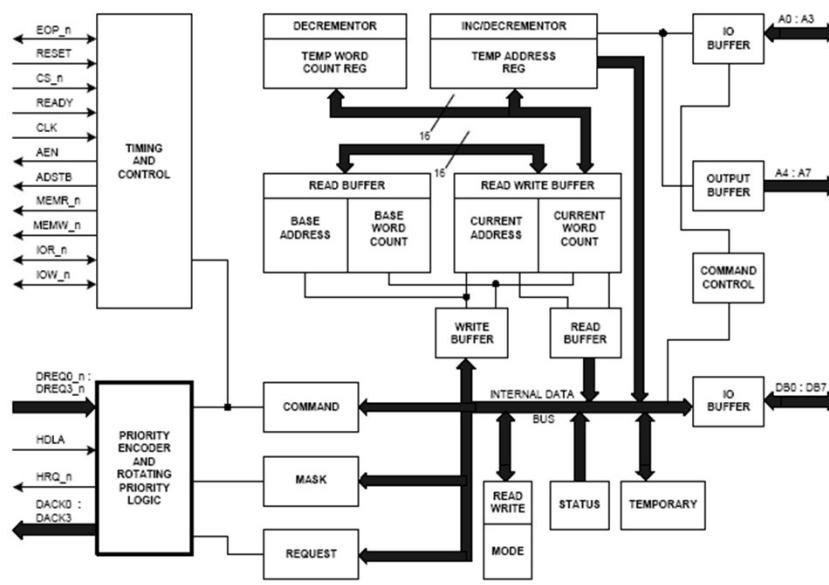
6.2 Mạch DMAC 8237A của Intel

- Clk: < 5MHz.
- CS: Output of a decoder.
- RESET: Clears all internal registers (command, status, request, etc.).
- READY: Allows memory and I/O to insert wait states into the 8237.
- HLDA: Input that tells 8237 that micro has released address, data and control buses.
- DREQ₃-DREQ₀: DMA request inputs used to request a DMA transfer.
- DB₇-DB₀: Used to program the 8237 and output upper 8-bits of address.
- IOR, IOW, MEMR, MEMW: Outputs used to control memory and I/O.
- EOP: Bidirectional: as an input, used to terminate a DMA transfer; as an output, signals the end of the DMA transfer.
- A₃-A₀: Address pins select an internal register during programming and output part of the address for a transfer.
- A₇-A₄: Address outputs.
- HRQ: Output that connects to HOLD pin on micro to request a DMA.
- DACK₃ - DACK₀: Used to select an I/O device (ack a DMA request).
- AEN/ADSTB: Enable latch (and strobe) to transfer DB_x to upper 8 A bits.

5

© DHBK 2005

6.2 Mạch DMAC 8237A của Intel



6

6.2 Mạch DMA 8237A của Intel

Some of the internal registers are:

- CAR₃ - CAR₀: Used to hold the 16-bit memory address used for a DMA transfer. Either incremented or decremented after a byte is transferred.
- CWCR₃ - CWCR₀: Current word count register programs a channel for the # of bytes (up to 64KB) transferred during a DMA action.
- CR: Command register programs the operation of the 8237. Bits in this register allow:
 - Memory-to-memory transfers (like MOVSB) where DMA channel 0 holds the source address and DMA channel 1 holds the dest address.
 - Memory-to-memory transfers in which DMA channel 0 holds a constant address -- used to fill a memory regions with a constant.
 - Fixed or rotating DMA channel priority, plus misc other options.
- MR: 'Mode of operation' register -- one for each channel. For example, block mode is used for memory-to-memory transfers.
- RR: Request register is used to request a DMA transfer via software -- essential for processor initiated memory-to-memory transfers.
- SR: Status register indicates when a DMA has completed.