



Thuvien**PDF**.com

Kỹ thuật

**Vì xử lý**

Micropipette

UDS

son

UDS EBOOK  
[www.updatesofts.com](http://www.updatesofts.com)



Khoa  
DIỆN TỬ -  
VIỄN THÔNG



# Kỹ thuật vi xử lý **Microprocessors**

**Giảng viên: Phạm Ngọc Nam**



# Your instructor

- **Bộ môn kỹ thuật điện tử tin học**
  - Office: C9-401
  - Email: pnam-fet@mail.hut.edu.vn
- **Research:**
  - **FPGA, PSoC, hệ nhúng**
  - **Trí tuệ nhân tạo**
- **Education:**
  - **K37 điện tử-ĐHBK Hà nội (1997)**
  - **Master về trí tuệ nhân tạo 1999, Đại học K.U. Leuven, Vương quốc Bỉ**
    - ⇒ **Đề tài: Nhận dạng chữ viết tay**
  - **Tiến sĩ kỹ thuật chuyên ngành điện tử-tin học, 9/ 2004, Đại học K.U. Leuven, Vương Quốc Bỉ**
    - ⇒ **Đề tài: quản lý chất lượng dịch vụ trong các ứng dụng đa phương tiện tiên tiến**



# Nội dung môn học

- 1. Giới thiệu chung về bộ 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 ngắn**
- 6. Truy cập bộ nhớ trực tiếp DMA**
- 7. Các bộ vi xử lý trên thực tế**



# Tài liệu tham khảo

- Slides
- Văn Thế Minh, Kỹ thuật vi xử lý, Nhà xuất bản giáo dục, 1997.
- 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.
- 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: thiết kế một ứng dụng trên vi điều khiển: 20% tổng số điểm**
  - Làm theo nhóm 2-6 sinh viên**
  - Nộp danh sách các nhóm vào 3/1**
  - Các nhóm trình bày ý tưởng 17/1**
- **Kiểm tra: 10%**
  - 3 bài kiểm tra không báo trước**
  - dự đủ ít nhất 2 bài và kết quả của 2 bài > 5: 1 điểm**
  - thiếu 2 bài trở lên: không được thi lần 1**
- **Thi học kỳ:**
  - 1 câu lý thuyết, 2 câu bài tập (lập trình và thiết kế)**
  - 70% tổng số điểm**



# Chương 1

## Giới thiệu chung về hệ vi xử lý

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

# Chương 1

## Giới thiệu chung về hệ vi xử lý

- **Lịch sử phát triển của các bộ vi xử lý và máy tính**
  - **Thế hệ -1: The early days (...-1642)**
  - **Thế hệ 0: Mechanical (1642-1945)**
  - **Thế hệ 1: Vacuum tubes (1945-1955)**
  - **Thế hệ 2: Discrete transistors (1955-1965)**
  - **Thế hệ 3: Integrated circuits (1965-1980)**
  - **Thế hệ 4: VLSI (1980-?)**
- **Phân loại vi xử lý**
- **Các hệ đếm dùng trong máy tính ( nhắc lại)**
- **Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý**



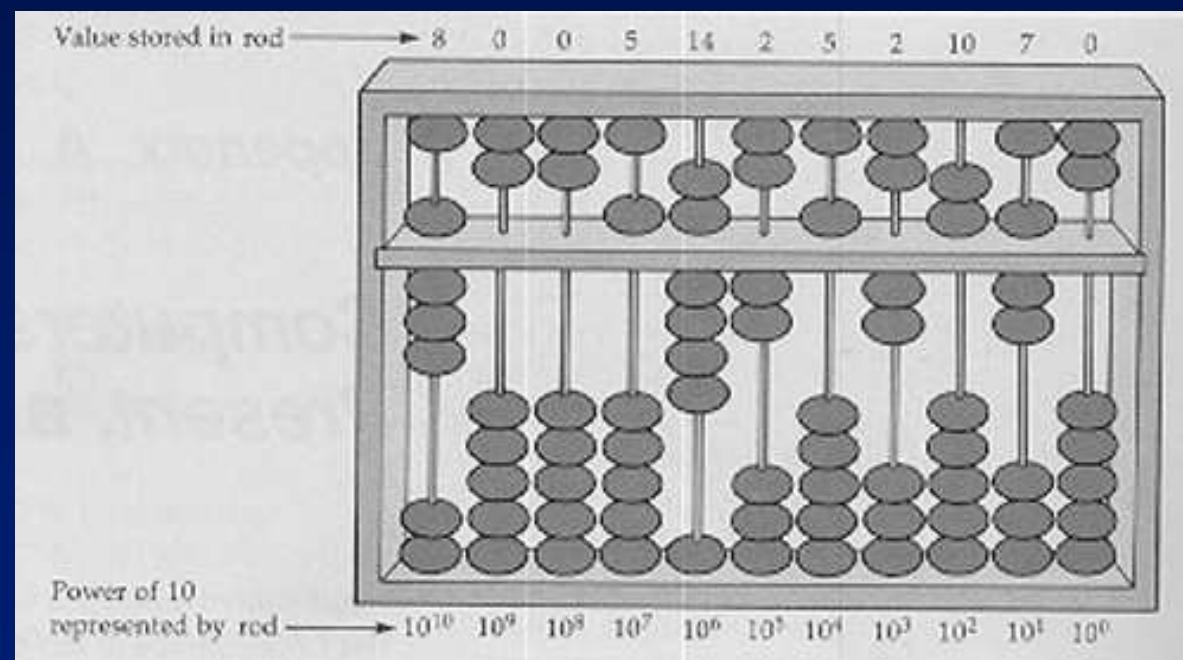
# Chương 1

## Giới thiệu chung về hệ vi xử lý

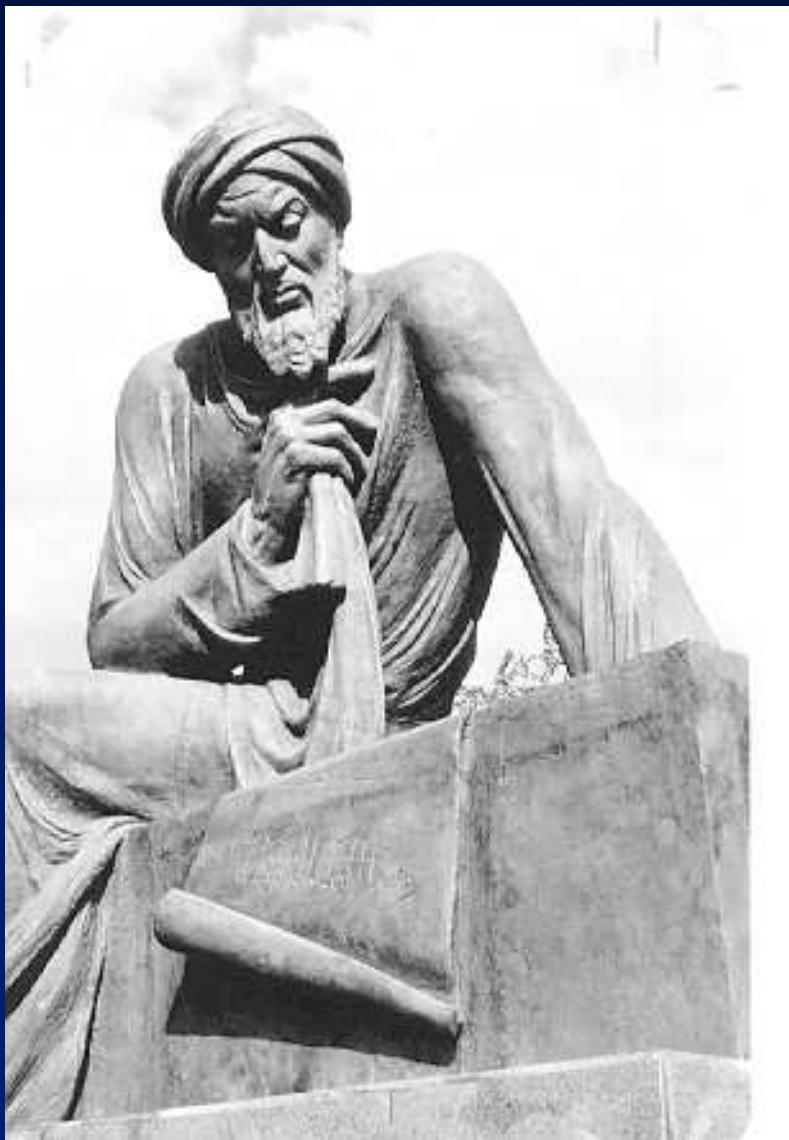
- Lịch sử phát triển của các bộ vi xử lý và máy tính
  - Thế hệ -1: The early days (...-1642)**
  - Thế hệ 0: Mechanical (1642-1945)**
  - Thế hệ 1: Vacuum tubes (1945-1955)**
  - Thế hệ 2: Discrete transistors (1955-1965)**
  - Thế hệ 3: Integrated circuits (1965-1980)**
  - Thế hệ 4: VLSI (1980-?)**
- Phân loại vi xử lý
- Các hệ đếm dùng trong máy tính ( nhắc lại)
- Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý

# Thế hệ -1: The early days (...-1642)

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



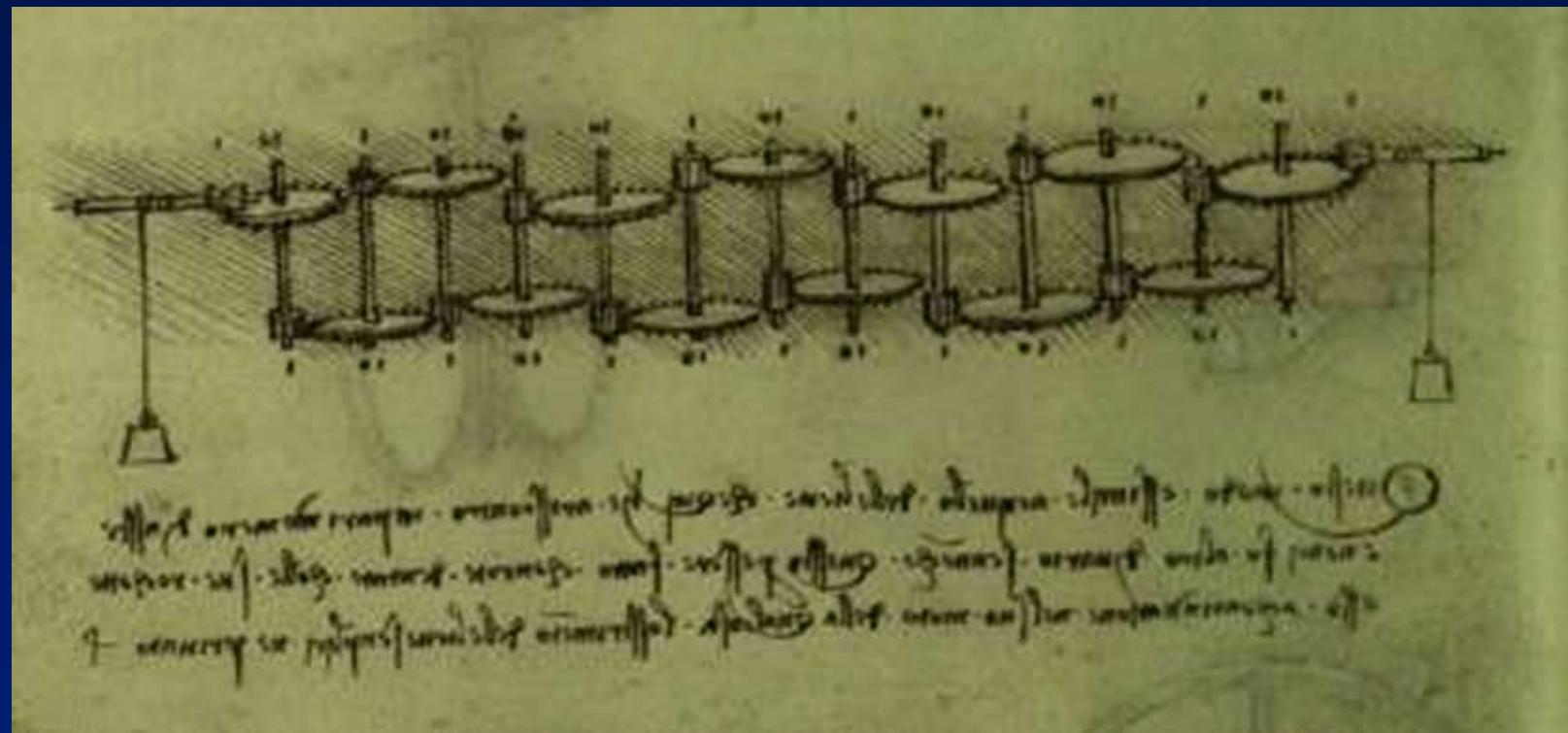
# Thế hệ -1: The early days (...-1642)



- **Thế kỷ 12: *Muhammad ibn Musa Al'Khowarizmi* đưa ra khái niệm về giải thuật *algorithm***

# Thế hệ -1: The early days (...-1642)

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





# Chương 1

## Giới thiệu chung về hệ vi xử lý

- **Lịch sử phát triển của các bộ vi xử lý và máy tính**
  - Thế hệ -1: The early days (...-1642)**
  - Thế hệ 0: Mechanical (1642-1945)**
  - Thế hệ 1: Vacuum tubes (1945-1955)**
  - Thế hệ 2: Discrete transistors (1955-1965)**
  - Thế hệ 3: Integrated circuits (1965-1980)**
  - Thế hệ 4: VLSI (1980-?)**
- **Phân loại vi xử lý**
- **Các hệ đếm dùng trong máy tính ( nhắc lại)**
- **Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

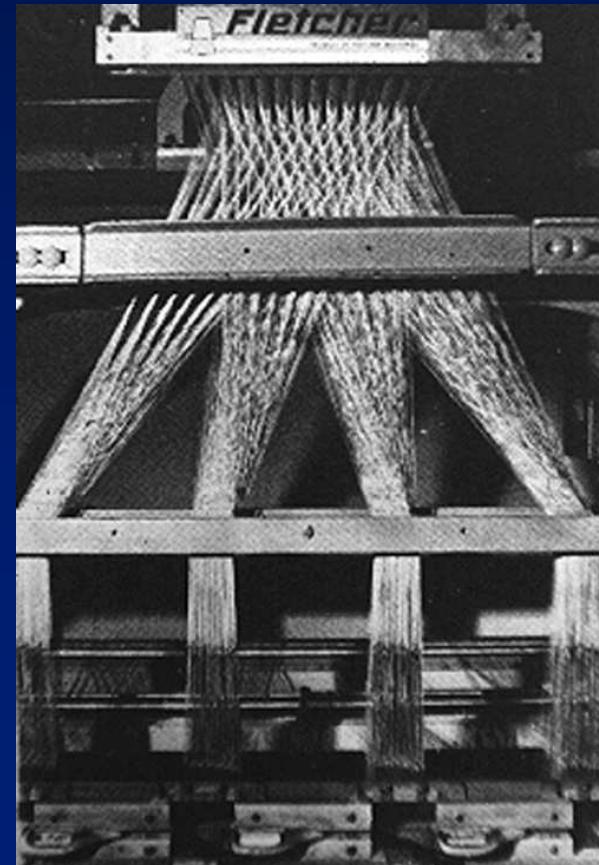
# Thế hệ 0: Mechanical (1642-1945)

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



# Thế 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 họa tiết dệt trên vải
- Bìa đục lỗ lưu trữ chương trình: máy đa năng đầu tiên



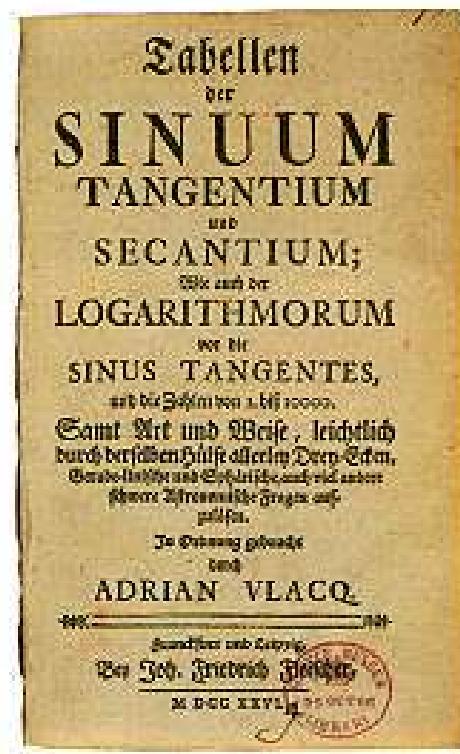
## Thế hệ 0: Mechanical (1642-1945)



- **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**



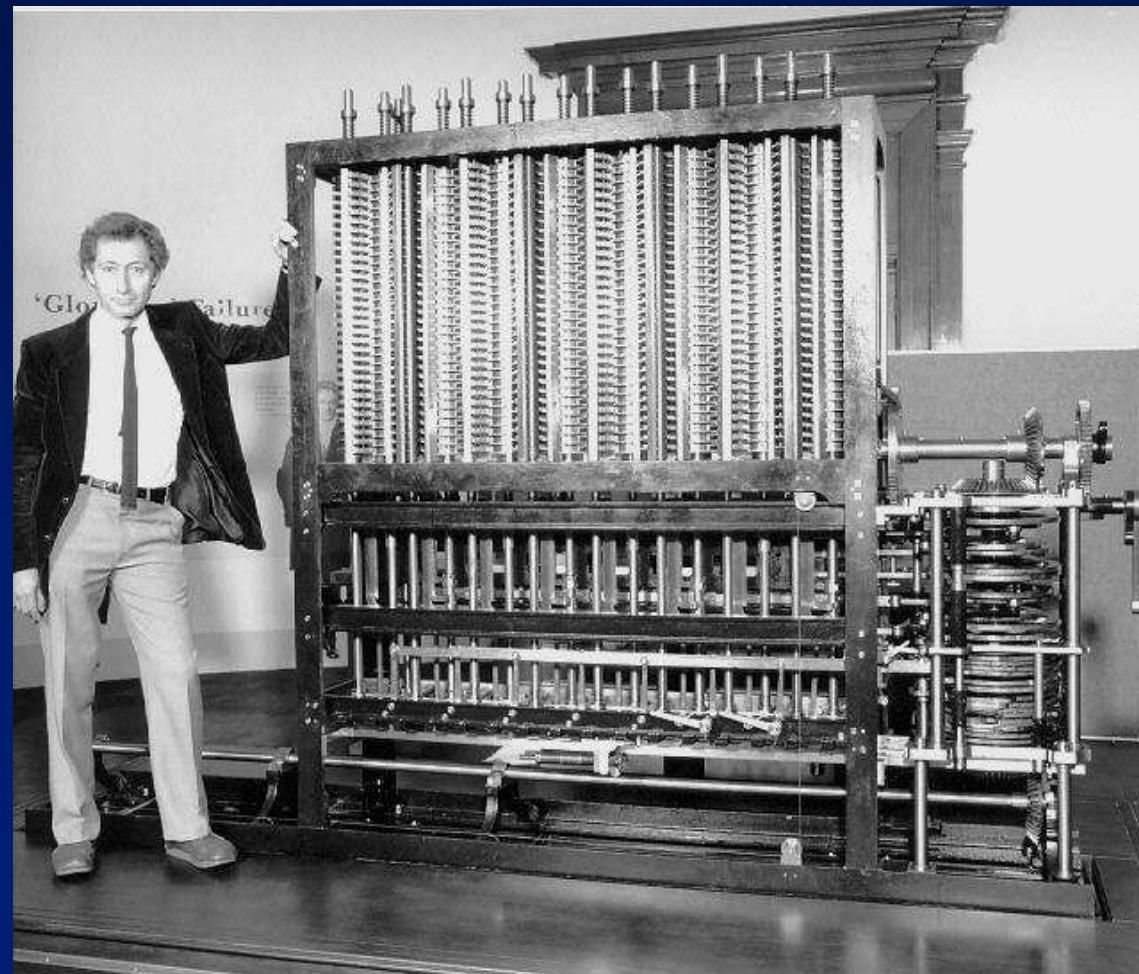
# Thế hệ 0: Mechanical (1642-1945)



3,700 errors

# Thế 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)**





## Thế 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





## Thế 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 (International Business Machine) vào năm 1924.**

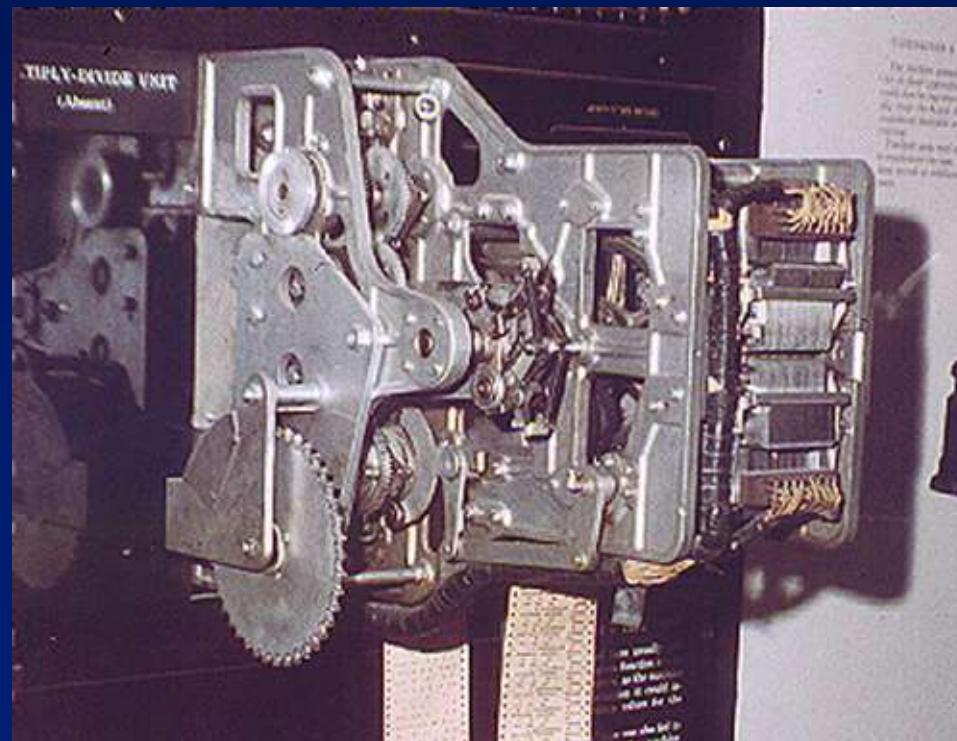
# Thế hệ 0: Mechanical (1642-1945)

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



## Thế hệ 0: Mechanical (1642-1945)

- **Máy tính cơ điện tự động lớn đa năng 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.**





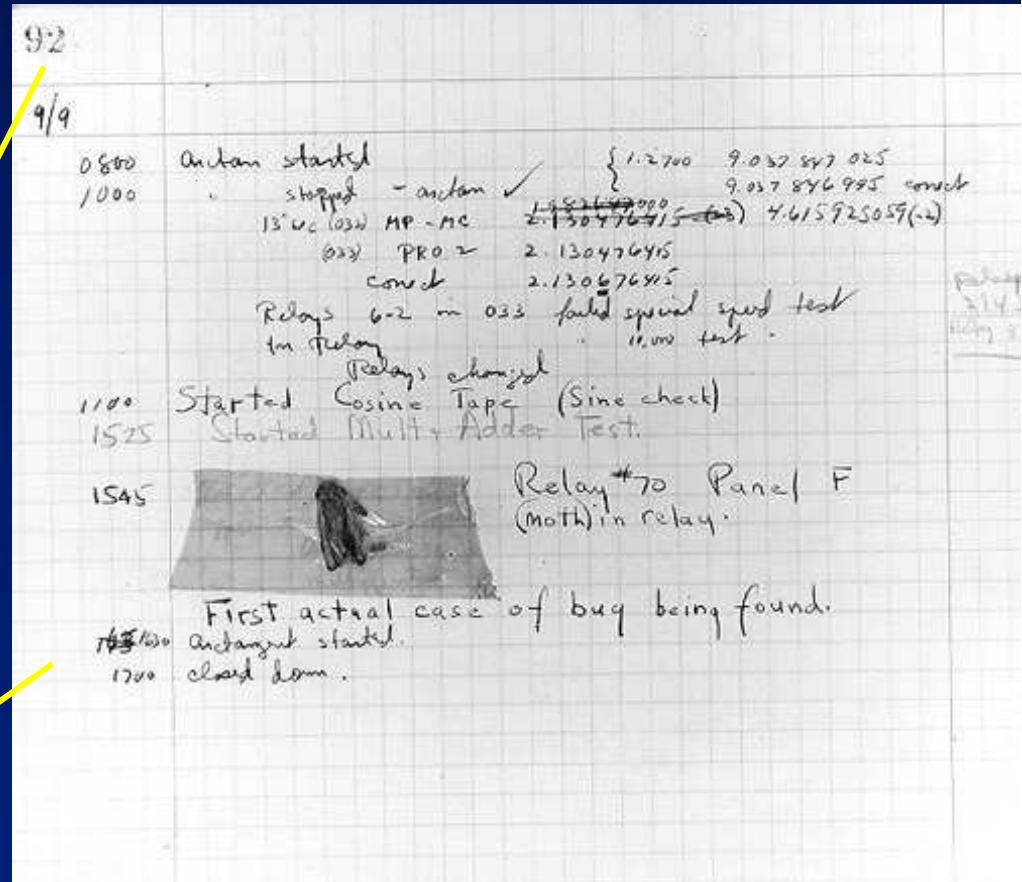
# Thế hệ 0: Mechanical (1642-1945)

- 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

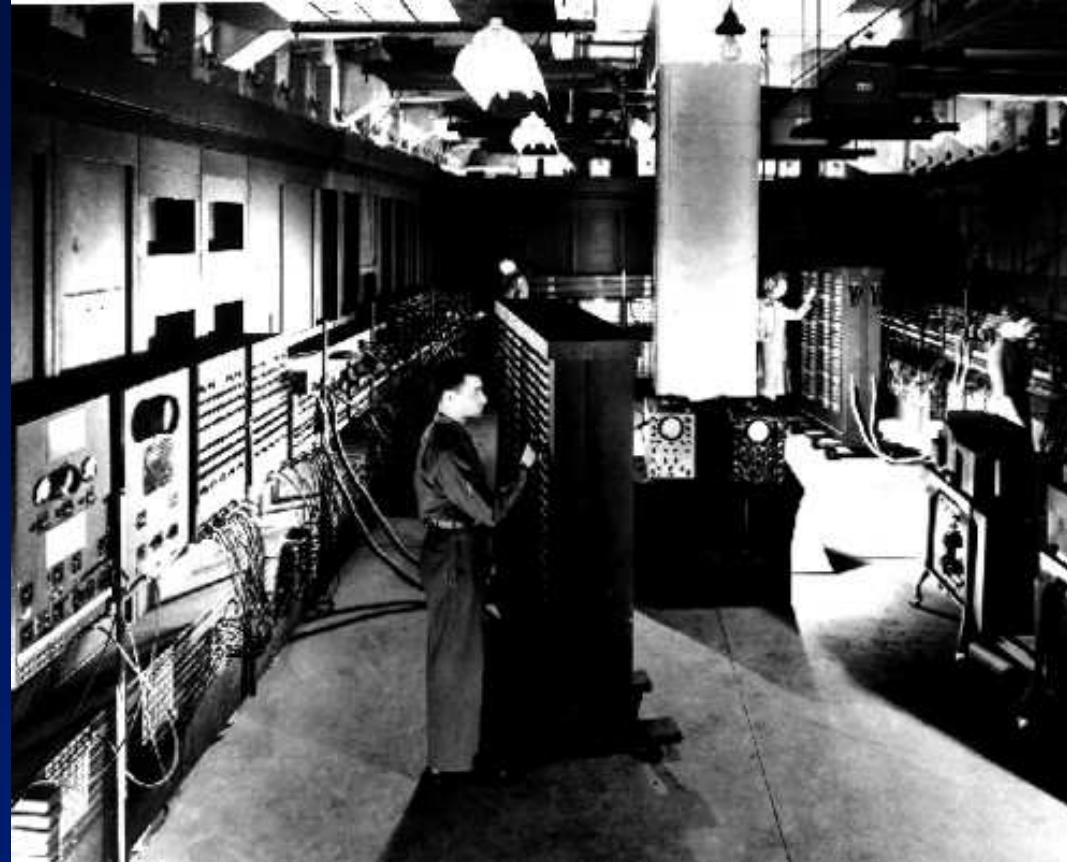
- **Lịch sử phát triển của các bộ vi xử lý và máy tính**
  - Thế hệ -1: The early days (...-1642)**
  - Thế hệ 0: Mechanical (1642-1945)**
  - Thế hệ 1: Vacuum tubes (1945-1955)**
  - Thế hệ 2: Discrete transistors (1955-1965)**
  - Thế hệ 3: Integrated circuits (1965-1980)**
  - Thế hệ 4: VLSI (1980-?)**
- **Phân loại vi xử lý**
- **Các hệ đếm dùng trong máy tính ( nhắc lại)**
- **Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

# Thế hệ 1: Vacuum tubes (1945-1955)



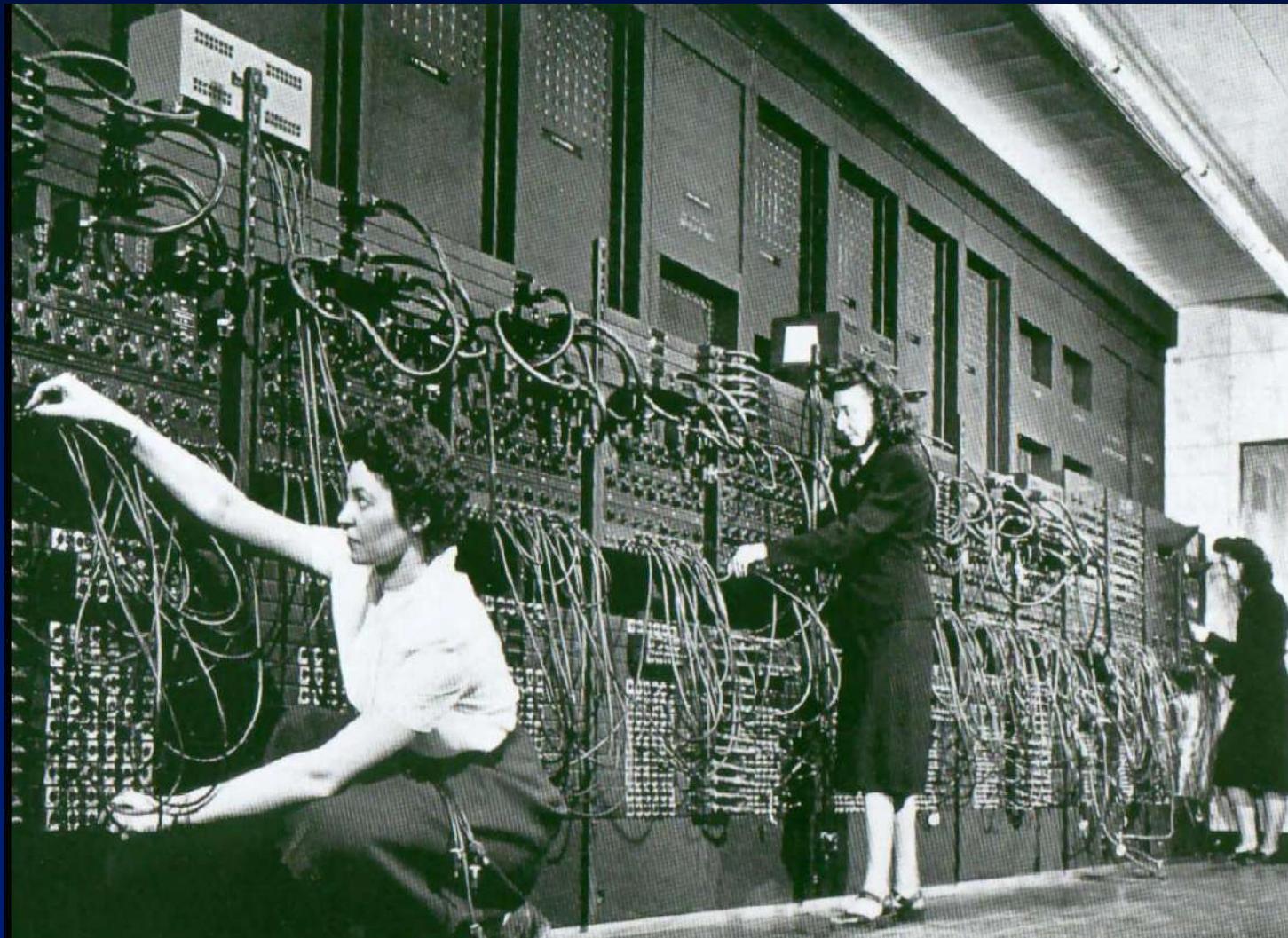
- Năm 1943, John Mauchly và J. Presper Eckert bắt đầu nghiên cứu về ENIAC

# Thế hệ 1: Vacuum tubes (1945-1955)



- **18000 vacuum tubes, 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)

# Thế hệ 1: Vacuum tubes (1945-1955)



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



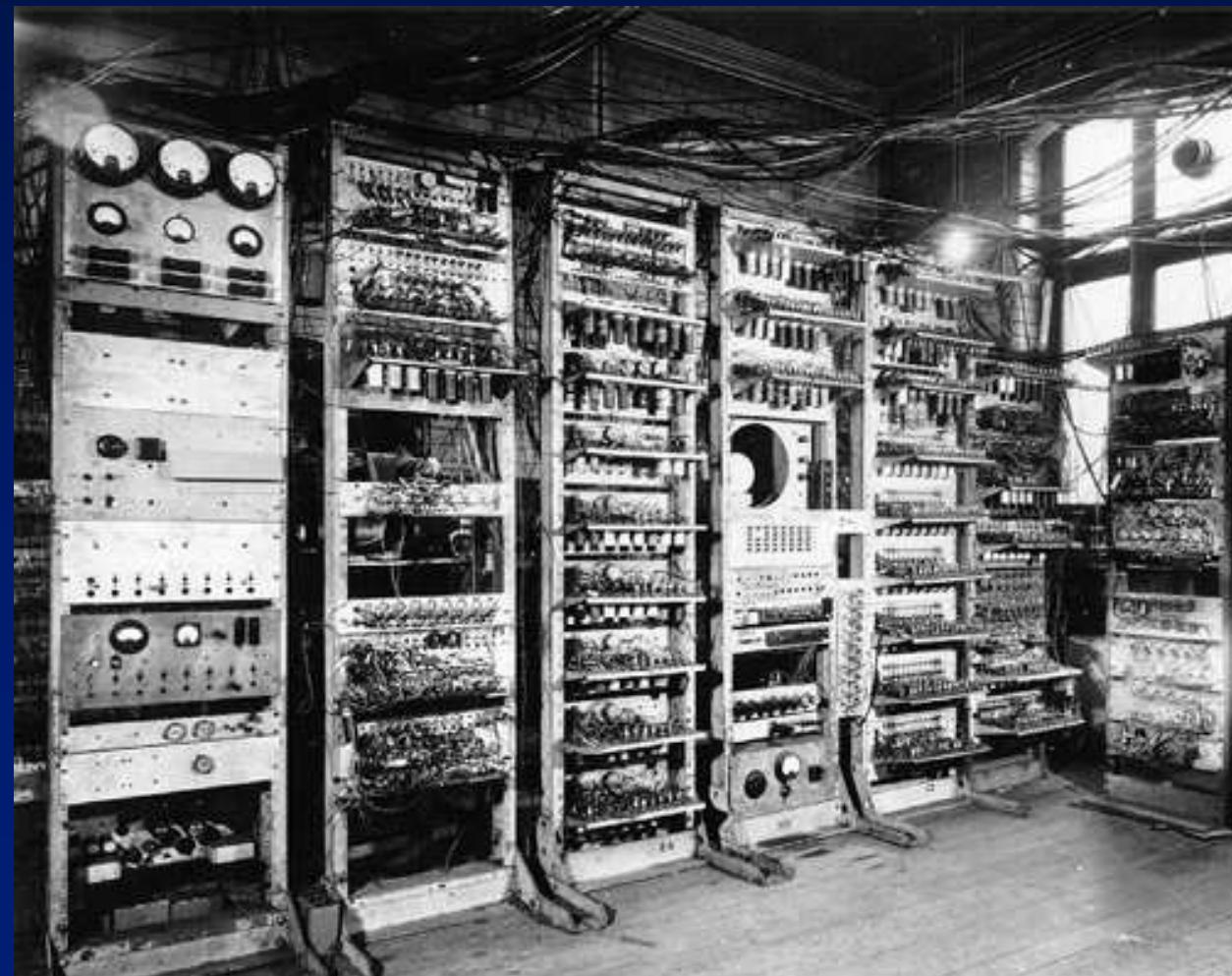
## Thế hệ 1: Vacuum tubes (1945-1955)

- 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 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: Vacuum tubes (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**





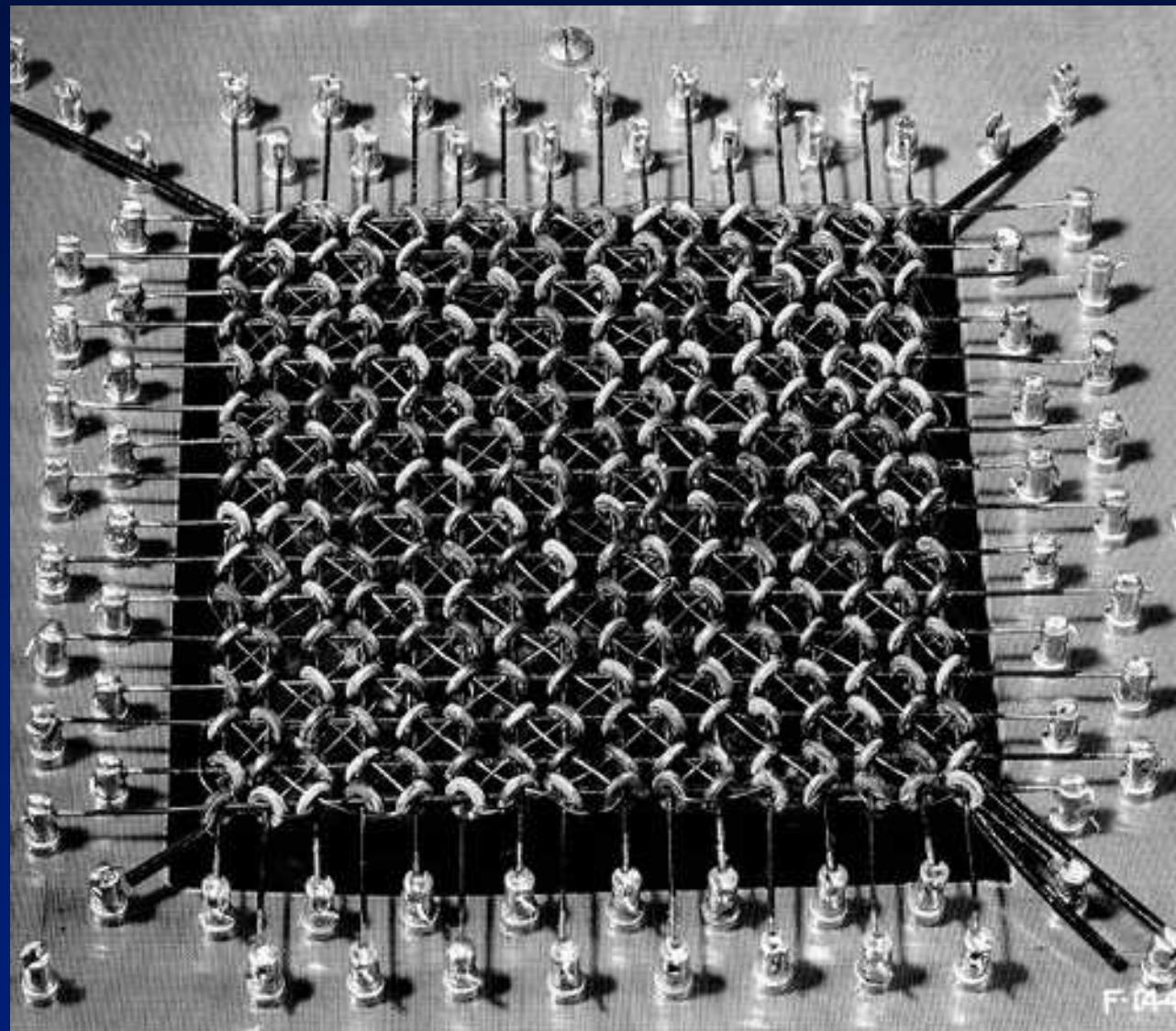
## Thế hệ 1: Vacuum tubes (1945-1955)

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



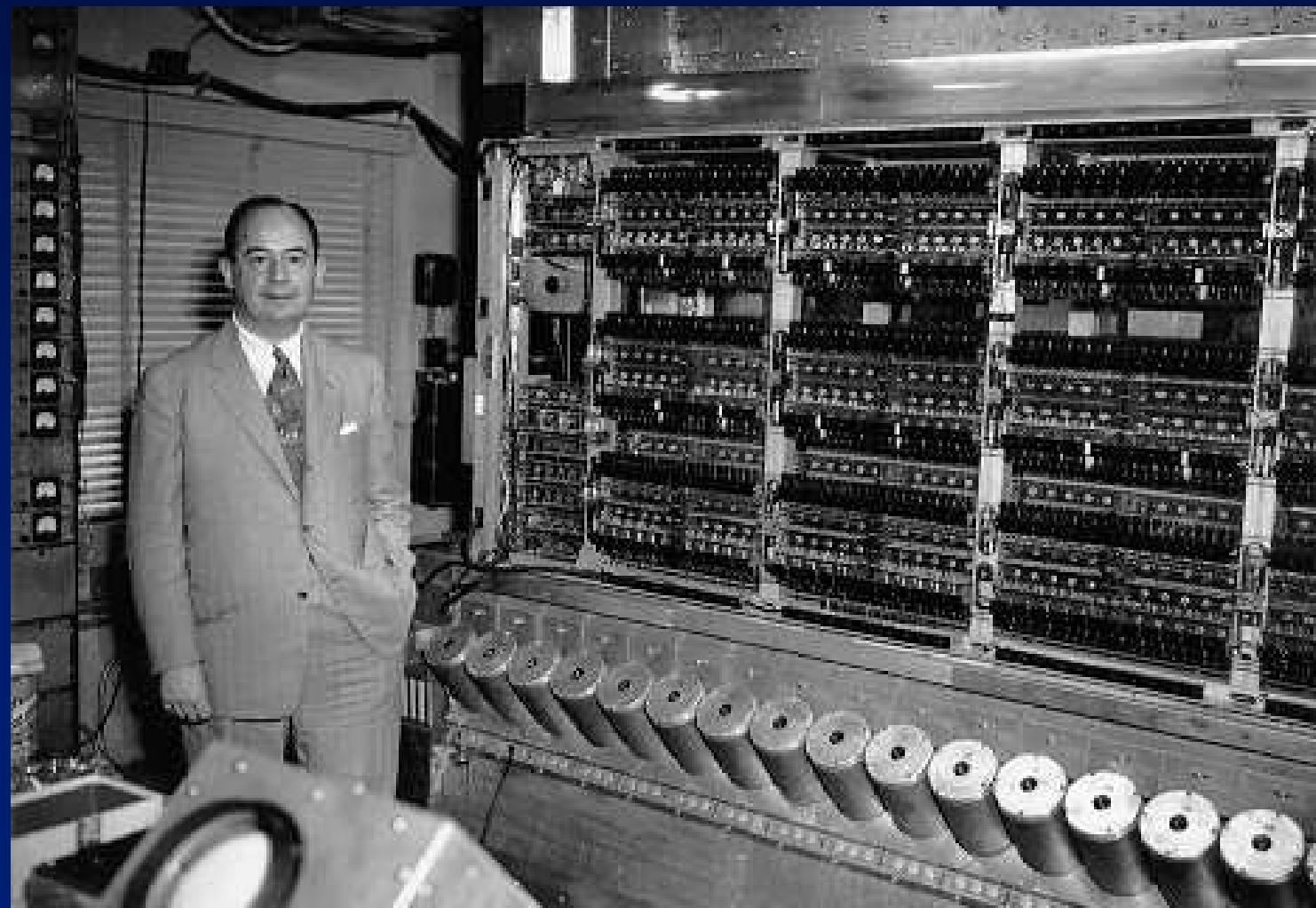
# Thế hệ 1: Vacuum tubes (1945-1955)

- Một magnetic core lưu trữ 256 bits



# Thế hệ 1: Vacuum tubes (1945-1955)

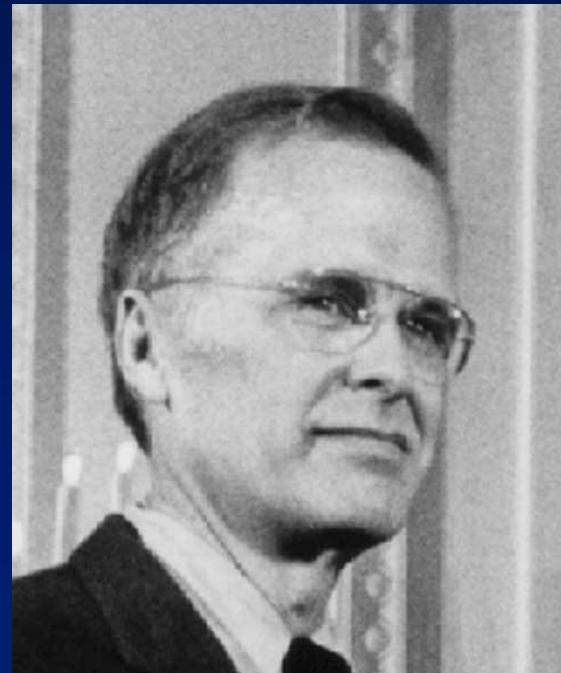
- John von Neumann năm 1952 với chiếc máy tính mới của ông





# Thế hệ 1: Vacuum tubes (1945-1955)

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





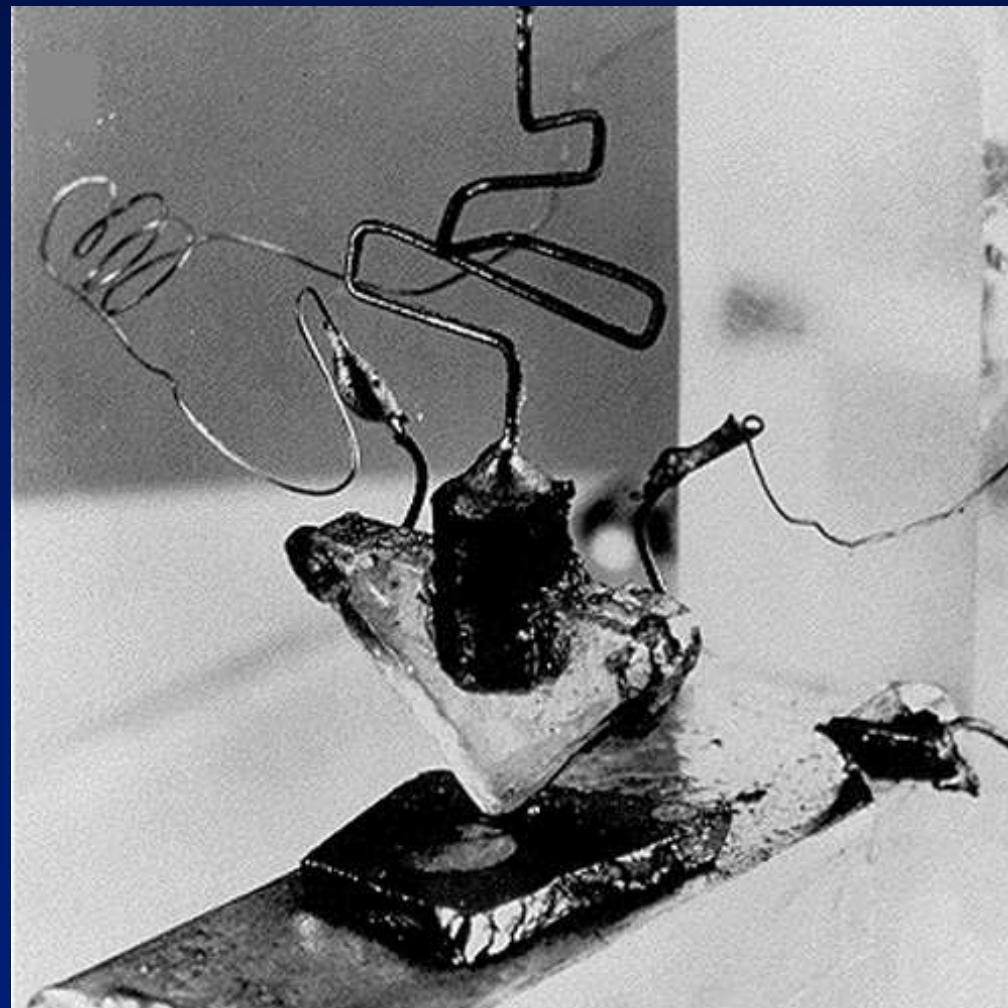
# Chương 1

## Giới thiệu chung về hệ vi xử lý

- Lịch sử phát triển của các bộ vi xử lý và máy tính
  - Thế hệ -1: The early days (...-1642)
  - Thế hệ 0: Mechanical (1642-1945)
  - Thế hệ 1: Vacuum tubes (1945-1955)
  - Thế hệ 2: Discrete transistors (1955-1965)
  - Thế hệ 3: Integrated circuits (1965-1980)
  - Thế hệ 4: VLSI (1980-?)
- Phân loại vi xử lý
- Các hệ đếm dùng trong máy tính ( nhắc lại)
- Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý

## Thế hệ 2: Discrete transistors (1955-1965)

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



## 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)





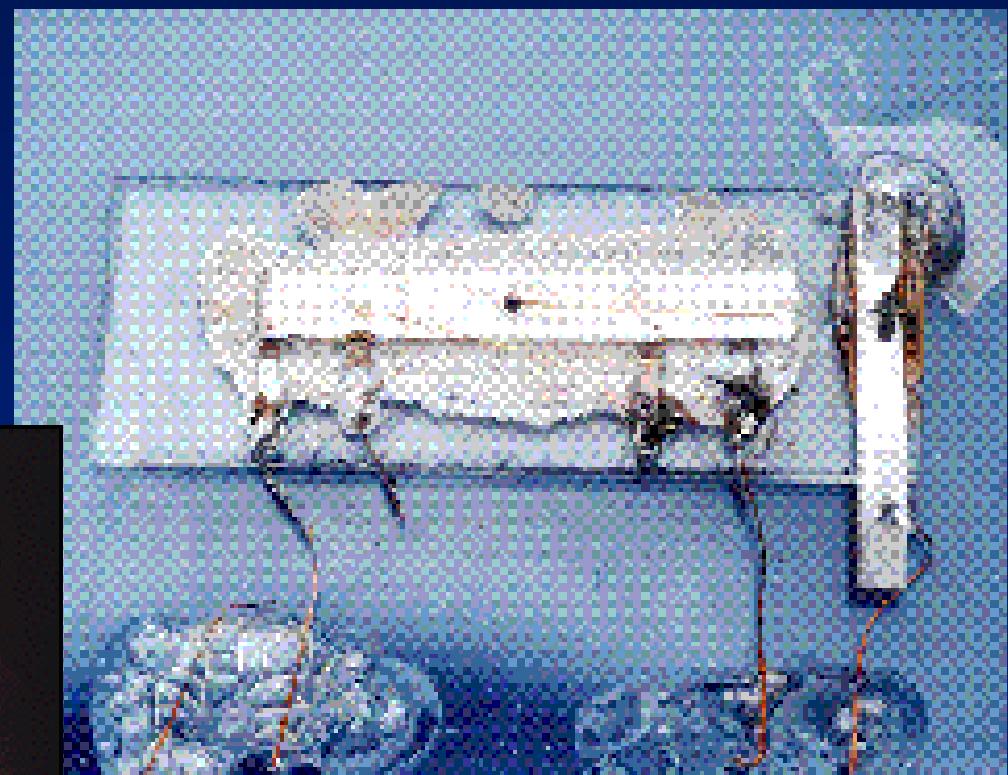
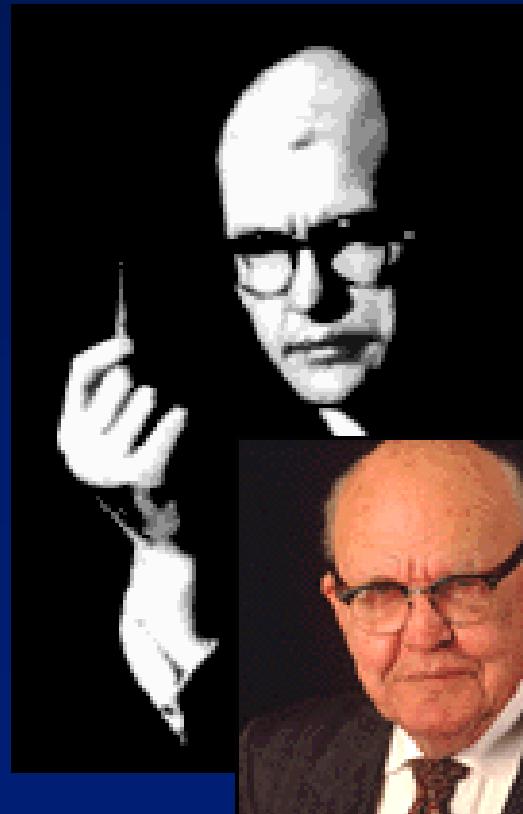
# Chương 1

## Giới thiệu chung về hệ vi xử lý

- **Lịch sử phát triển của các bộ vi xử lý và máy tính**
  - Thế hệ -1: The early days (...-1642)**
  - Thế hệ 0: Mechanical (1642-1945)**
  - Thế hệ 1: Vacuum tubes (1945-1955)**
  - Thế hệ 2: Discrete transistors (1955-1965)**
  - Thế hệ 3: Integrated circuits (1965-1980)**
  - Thế hệ 4: VLSI (1980-?)**
- **Phân loại vi xử lý**
- **Các hệ đếm dùng trong máy tính ( nhắc lại)**
- **Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

## 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.



## Thế hệ 3: Integrated circuits (1965-1980)

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



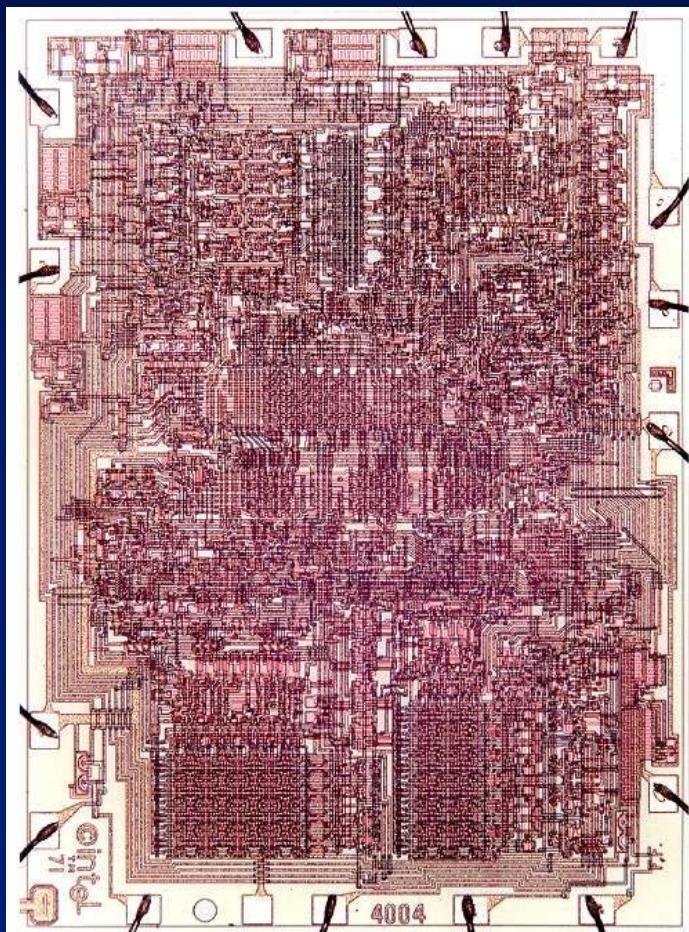
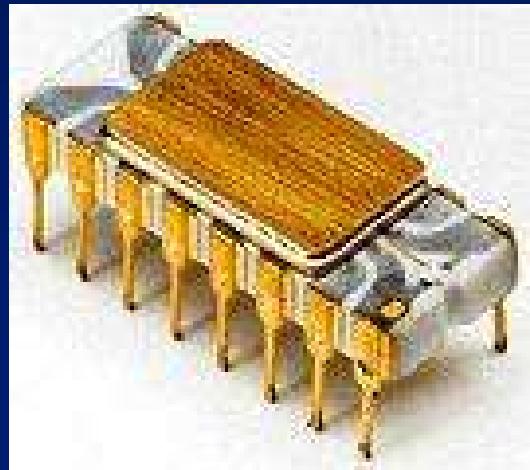
## Thế hệ 3: Integrated circuits (1965-1980)

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



## Thế hệ 3: Integrated circuits (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





## Thế hệ 3: Integrated circuits (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





# Chương 1

## Giới thiệu chung về hệ vi xử lý

- **Lịch sử phát triển của các bộ vi xử lý và máy tính**
  - Thế hệ -1: The early days (...-1642)**
  - Thế hệ 0: Mechanical (1642-1945)**
  - Thế hệ 1: Vacuum tubes (1945-1955)**
  - Thế hệ 2: Discrete transistors (1955-1965)**
  - Thế hệ 3: Integrated circuits (1965-1980)**
  - Thế hệ 4: VLSI (1980-?)**
- **Phân loại vi xử lý**
- **Các hệ đếm dùng trong máy tính ( nhắc lại)**
- **Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý**



## Thế hệ 4: VLSI (1980-?)

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





## 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

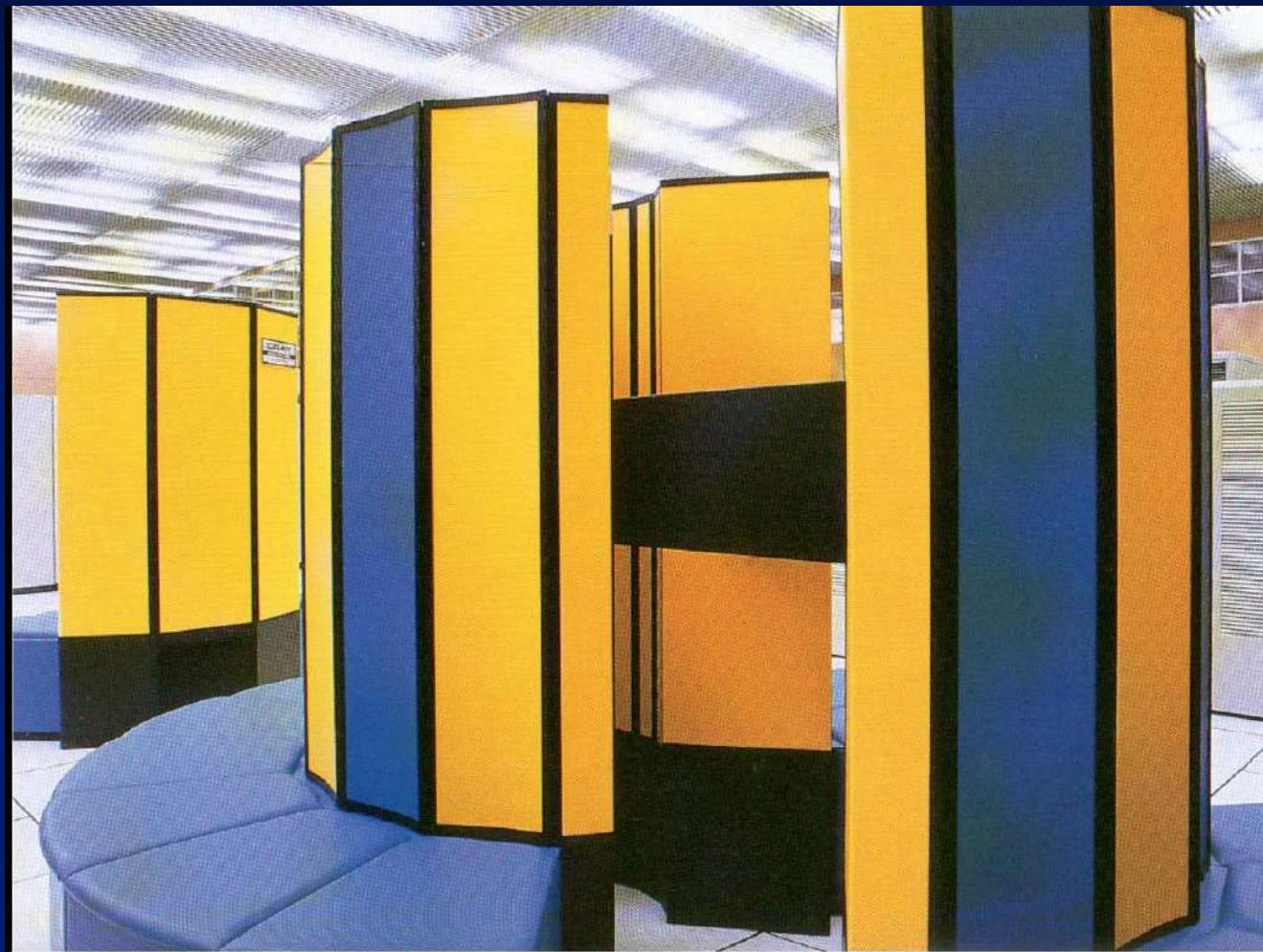


**Con chuột đầu tiên**



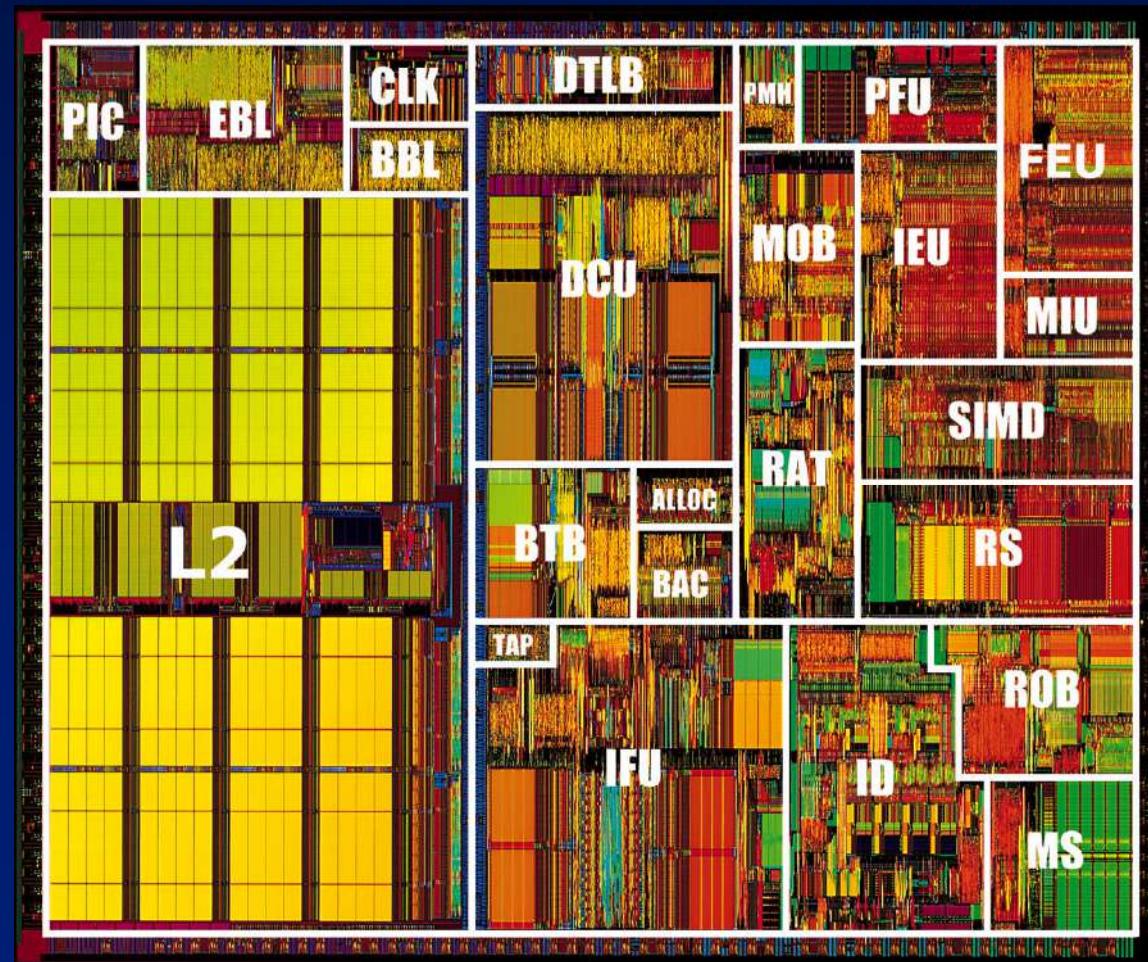
## Thế hệ 4: VLSI (1980-?)

- 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



## Thế 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





# Chương 1

## Giới thiệu chung về hệ vi xử lý

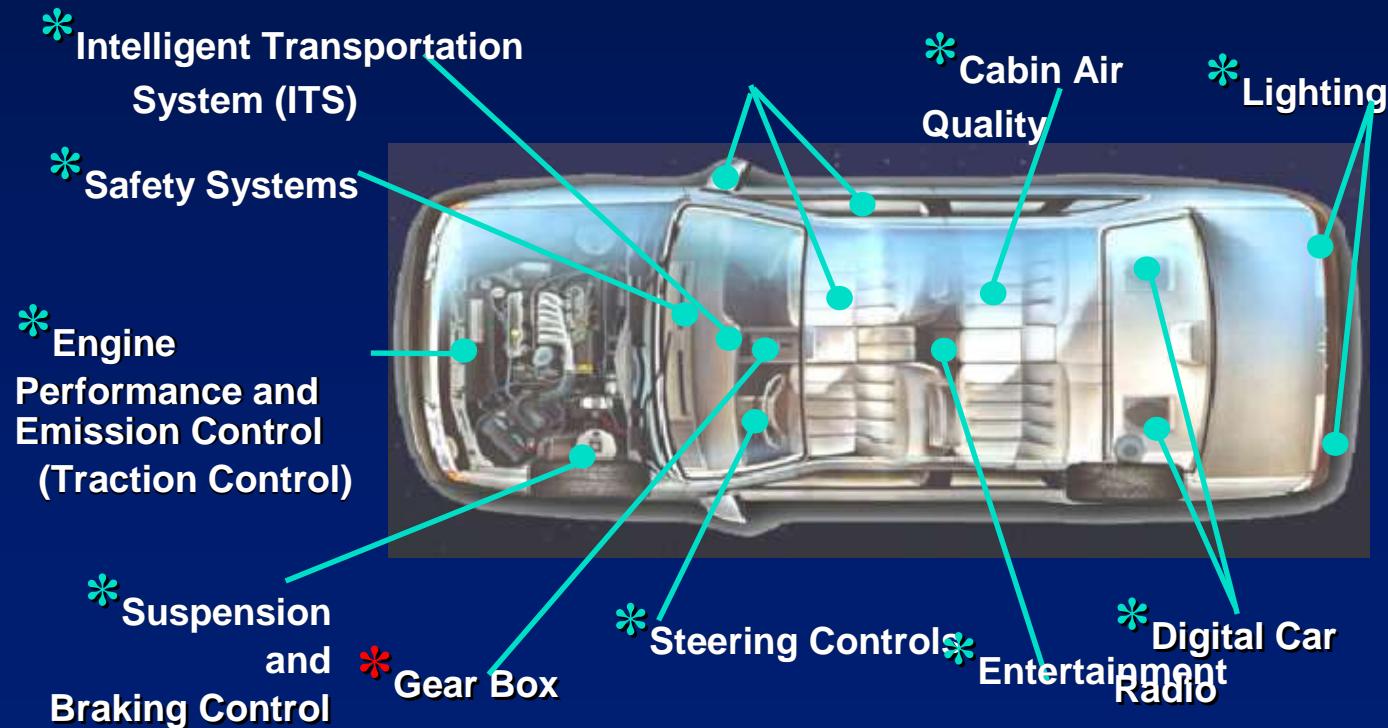
- **Lịch sử phát triển của các bộ vi xử lý và máy tính**
- **Phân loại vi xử lý**
- **Các hệ đếm dùng trong máy tính ( nhắc lại)**
- **Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý**

# Phân loại vi xử lý



# Phân loại vi xử lý

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





# Phân loại vi xử lý

**Phân loại theo giá thành:**

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



# Phân loại vi xử lý

- **Phân loại theo chức năng:**

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

- **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**
  - **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**



# Chương 1

## Giới thiệu chung về hệ vi xử lý

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



# Chương 1

## Giới thiệu chung về hệ vi xử lý

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



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

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



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

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



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

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



# Hệ 16 (Hexadecimal)

- **FEDC,76<sub>16</sub>=**
  - $15 \cdot 4096 + 14 \cdot 256 + 13 \cdot 16 + 12 \cdot 1 + 7 \cdot 1/16 + 6 \cdot 1/256$
  - $15 \cdot 16^3 + 14 \cdot 16^2 + 13 \cdot 16^1 + 12 \cdot 16^0 + 7 \cdot 16^{-1} + 6 \cdot 16^{-2}$
  - r = cơ số (r = 16), d=digit ( $0 \leq d \leq F$ ), m = số chữ số trước dấu phẩy, n = số chữ số sau dấu phẩy**

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



# 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

## Giới thiệu chung về hệ vi xử lý

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



# Cộng nhị phân

- Cộng thập phân

$$\begin{array}{r} \text{Nhớ} \quad 0 \ 1 \ 0 \\ \times \quad 8 \ 2 \ 7 \ 3 \\ \hline \text{y} \quad 5 \ 6 \ 2 \\ \text{Tổng} \quad 8 \ 8 \ 3 \ 5 \end{array}$$

- Cộng nhị phân

$$\begin{array}{r} \text{Nhớ} \quad 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \\ \times \quad 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \\ \hline \text{y} \quad 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \\ \text{Tổng} \quad 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \end{array}$$



# 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



# Nhân nhị phân

- **Nguyên tắc: cộng và dịch**

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



# Chia nhị phân

1 0 1 1 1 0 1 0

1 1 1 0

1 0 0 1 0 1 0

1 1 1 0

1 0 0 1 0

0 0 0 0

1 0 0 1 0

1 1 1 0

1 0 0

1 1 1 0

1 1 0 1

- **Nguyên tắc: trừ và dịch**



# Chương 1

## Giới thiệu chung về hệ vi xử lý

- Lịch sử phát triển của các bộ vi xử lý và máy tính
- Phân loại vi xử lý
- Các hệ đếm dùng trong máy tính ( nhắc lại)
  - Thập phân, Nhị phân, Hệ 8, Hệ 16
  - Cộng, trừ, nhân, chia
  - Các số âm
  - Số nguyên, số thực, BCD, ASCII
- Giới thiệu 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: bit dấu là bit MSB; ‘0’ = dương, ‘1’ = âm
- Ví dụ:  $01100_2 = +12_{10}$  và  $11100_2 = -12_{10}$
- Các số có dấu 8 bit 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**

$$13 = \quad \quad \quad 0000\ 1101$$

**Số bù 1 của 13 =**1111 0010

**Cộng thêm 1:**                            1

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



## 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**



# Số bù 2

Decimal	Số 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

- Lịch sử phát triển của các bộ vi xử lý và máy tính
- Phân loại vi xử lý
- Các hệ đếm dùng trong máy tính ( nhắc lại)
  - Thập phân, Nhị phân, Hệ 8, Hệ 16
  - Cộng, trừ, nhân, chia
  - Các số âm
  - Số nguyên, số thực, BCD, ASCII
- Giới thiệu 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



73/Chapter

# Số thực

(real number, floating point number)

- Ví dụ:  $1,234 = 1,234 \cdot 10^0 = 0,1234 \cdot 10^1 = \dots$
  - $11,01 \text{ B} = 1,101 \cdot 2^1 = 0,1101 \cdot 2^2 = \dots$

**mantissa**

**exponent**

- **Real number: (m, e)** , e.g. (0.1101, 2)
    - Single precision: 32 bit
    - Double precision: 64 bit



# Số thực (real number, floating point number)

- IEEE-754 format cho single-precision



**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**



# 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	$2^{-23}$ or $1.192 \times 10^{-7}$	$2^{-52}$ or $2.220 \times 10^{-16}$
Smallest positive	$2^{-126}$ or $1.175 \times 10^{-38}$	$2^{-1022}$ or $2.225 \times 10^{-308}$
Largest positive	$(2 - 2^{-23}) 2^{127}$ or $3.403 \times 10^{38}$	$(2 - 2^{-52}) 2^{1023}$ or $1.798 \times 10^{308}$
Decimal Precision	6 significant digits	15 significant digits



# BCD

- **Binary Coded Decimal number**

- **BCD chuẩn (BCD gói, packed BCD):**
  - ⇒ **1 byte biểu diễn 2 số BCD**
  - ⇒ **Ví dụ: 25: 0010 0101**

- **BCD không gói (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)**

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



# Chương 1

## Giới thiệu chung về hệ vi xử lý

- Lịch sử phát triển của các bộ vi xử lý và máy tính
- Phân loại vi xử lý
- Các hệ đếm dùng trong máy tính ( nhắc lại)
- Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý
  - Hệ vi xử lý

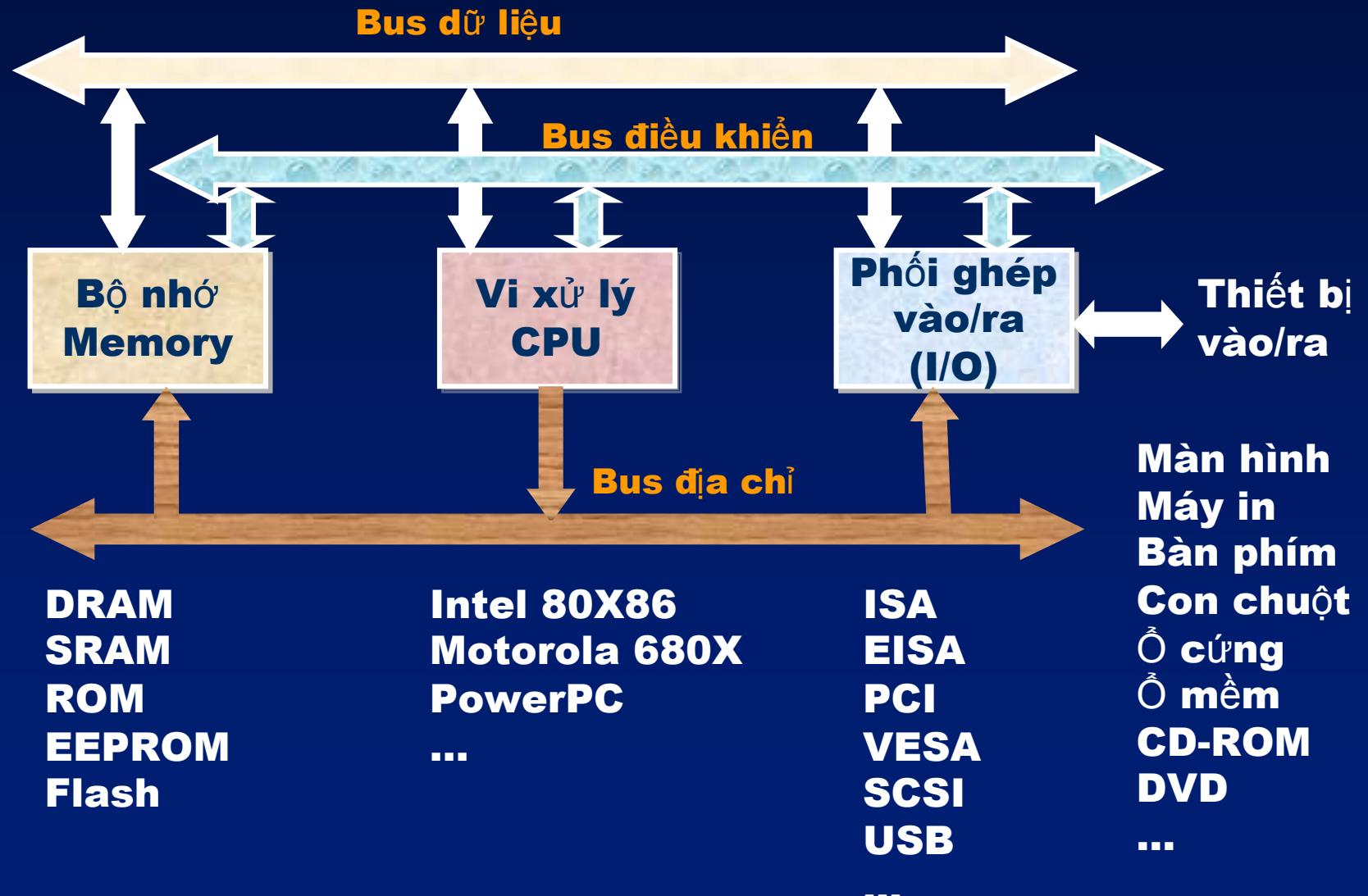


# Chương 1

## Giới thiệu chung về hệ vi xử lý

- Lịch sử phát triển của các bộ vi xử lý và máy tính
- Phân loại vi xử lý
- Các hệ đếm dùng trong máy tính ( nhắc lại)
- Giới thiệu sơ lược về cấu trúc và hoạt động của hệ vi xử lý
  - ✓ Hệ vi xử lý

# Hệ vi xử lý





# Hệ vi xử lý

- **CPU**

- **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**





# Hệ vi xử lý

- **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 => quản lý được  $2^{20}$  bytes=1Mbytes**



# Hệ vi xử lý

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



# 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 ngắn**
- 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

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Chương 2: Bộ vi xử lý Intel 8088/8086

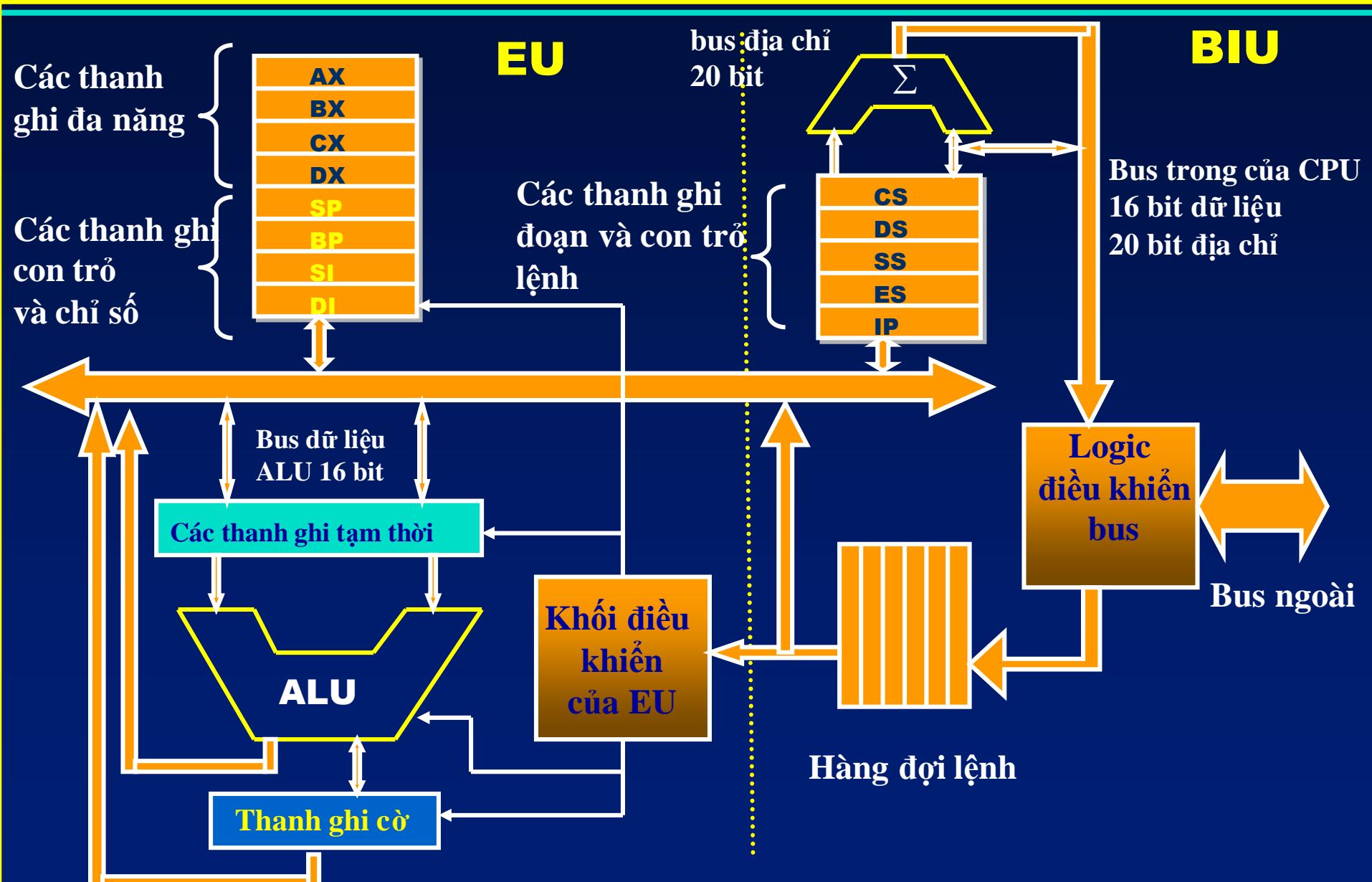
- **Cấu trúc bên trong**
  - **Sơ đồ khối**
  - **Các thanh ghi đa năng**
  - **Các thanh ghi đoạn**
  - **Các thanh ghi con trỏ và chỉ số**
  - **Thanh ghi cờ**
  - **Hàng đợi lệnh**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
  - Sơ đồ khối**
    - Các thanh ghi đa năng
    - Các thanh ghi đoạn
    - Các thanh ghi con trỏ và chỉ số
    - Thanh ghi cờ
    - Hàng đợi lệnh
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**

# Sơ đồ khối 8088/8086



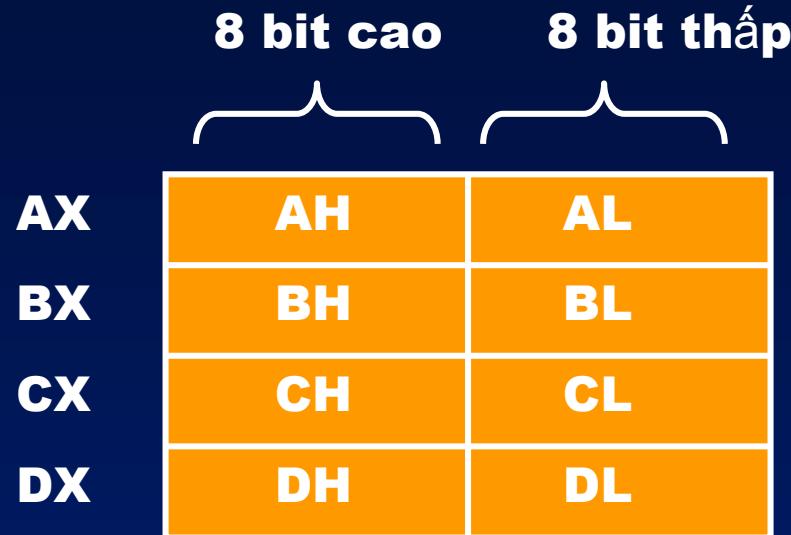


# Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
  - Sơ đồ khối**
  - Các thanh ghi đa năng**
  - Các thanh ghi đoạn**
  - Các thanh ghi con trỏ và chỉ số**
  - Thanh ghi cờ**
  - Hàng đợi lệnh**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Các thanh ghi đa năng của 8088/8086



- **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)



# Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
  - Sơ đồ khối**
  - Các thanh ghi đa năng**
  - Các thanh ghi đoạn**
  - Các thanh ghi con trỏ và chỉ số**
  - Thanh ghi cờ**
  - Hàng đợi lệnh**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**

# Các thanh ghi đoạn

- **Tổ chức của bộ nhớ 1 Mbytes**

- Đoạn bộ nhớ (segment) FFFFFFFH**

- ⇒  $2^{16}$  bytes = 64 KB

- ⇒ Đoạn 1: địa chỉ đầu 00000 H

- ⇒ Đoạn 2: địa chỉ đầu 00010 H

- ⇒ Đoạn cuối cùng: FFFF0 H

- 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)**





# 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**

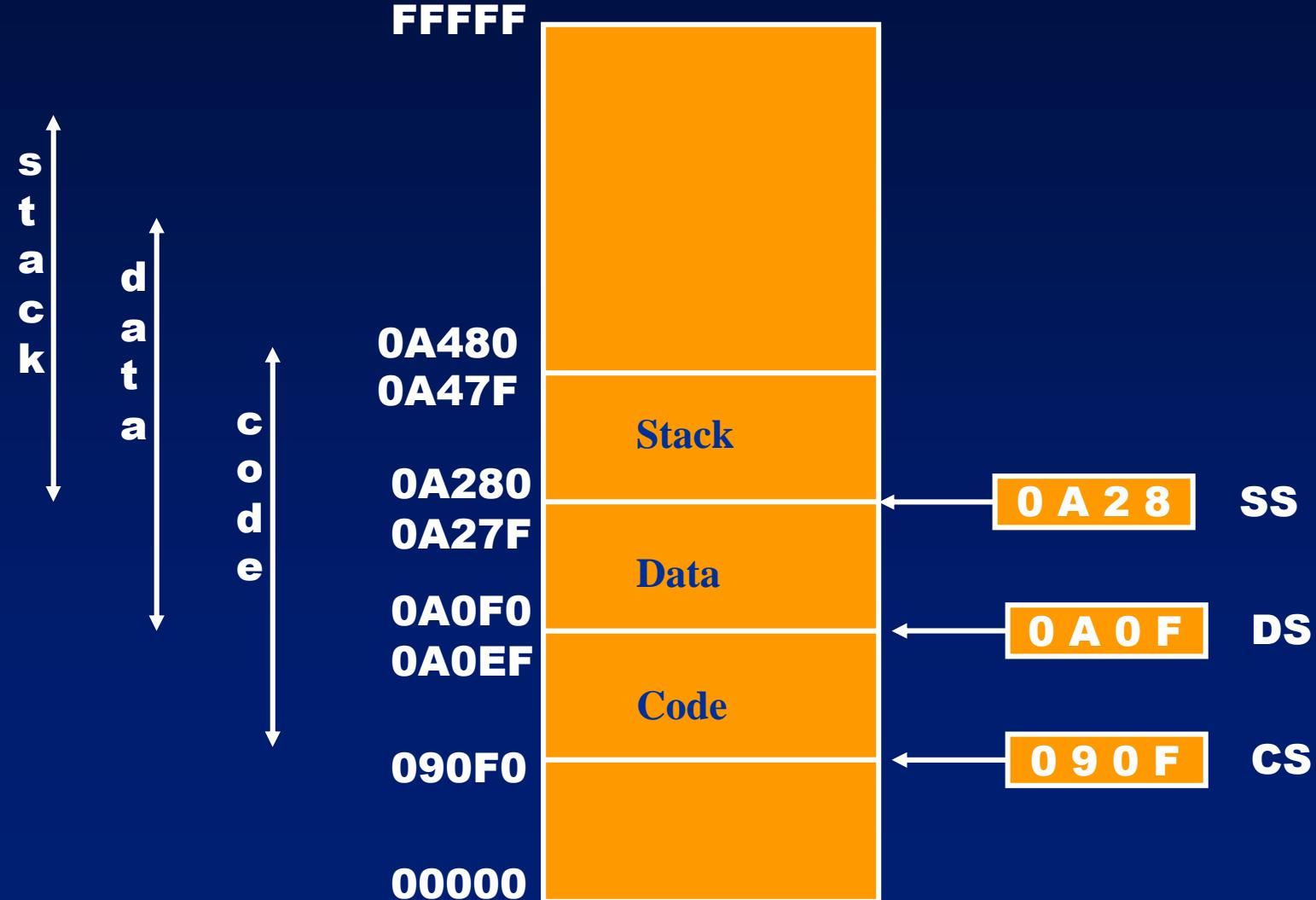
# Các thanh ghi đoạn

- Các thanh ghi đoạn: chứa địa chỉ đoạn



# Các thanh ghi đoạn

- Các đoạn chồng nhau





# Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
  - Sơ đồ khối**
  - Các thanh ghi đa năng**
  - Các thanh ghi đoạn**
  - Các thanh ghi con trỏ và chỉ số**
  - Thanh ghi cờ**
  - Hàng đợi lệnh**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Các thanh ghi con trỏ và chỉ số

- **Chứa địa chỉ lêch (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



# 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

- **Cấu trúc bên trong**
  - Sơ đồ khối**
  - Các thanh ghi đa năng**
  - Các thanh ghi đoạn**
  - Các thanh ghi con trỏ và chỉ số**
  - Thanh ghi cờ**
  - Hàng đợi lệnh**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Thanh ghi cờ (Flag Register)



- **9 bit được sử dụng, 6 cờ trạng thái:**
  - C hoặc CF (carry flag): CF=1 khi có nhó hoặc mượn từ MSB
  - P hoặc PF (parity flag): PF=1 (0) khi tổng số bít 1 trong kết quả là chẵn (lẻ)
  - A hoặc AF (auxiliary carry flag): cờ nhó phụ, AF=1 khi có nhó 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



# Thanh ghi cờ (Flag Register)



- **3 cờ điều khiển**

- **T hoặc TF (trap flag):** cờ bẫy, TF=1 khi 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)



# 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 bít của tổng bằng 1
- ZF=1 vì kết quả thu được là 0
- CF=1 vì có nhô từ bít msb trong phép cộng
- OF=1 vì có tràn trong phép cộng 2 số âm



## Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
  - Sơ đồ khối**
  - Các thanh ghi đa năng**
  - Các thanh ghi đoạn**
  - Các thanh ghi con trỏ và chỉ số**
  - Thanh ghi cờ**
  - Hàng đợi lệnh**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Hàng đợi lệnh

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

Không có  
pipelining



Có pipelining

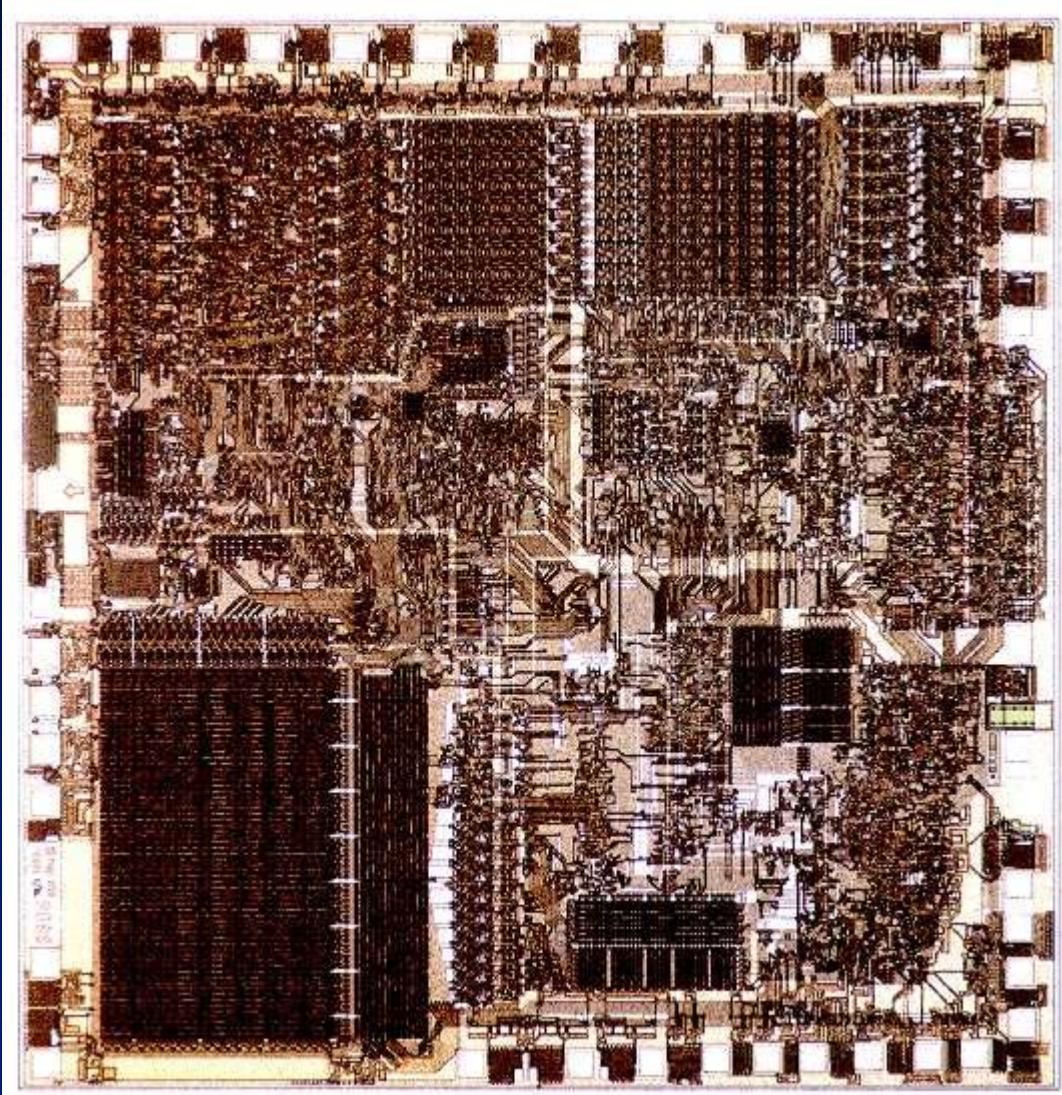




## Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**

# Intel 8088



- **16-bit processor**
- **introduced in 1979**
- **3  $\mu\text{m}$ , 5 to 8 MHz, 29 KTOR, 0.33 to 0.66 MIPS**

# Intel 8088

		MIN MODE	MAX MODE
GND	1	40	VCC
A14	2	39	A15
A13	3	38	A16/S3
A12	4	37	A17/S4
A11	5	36	A18/S5
A10	6	35	A19/S6
A9	7	34	SS0 (HIGH)
A8	8	33	MN/MX
AD7	9	32	RD
AD6	10	31	HOLD (RQ/GTO)
AD5	11	30	HLDA (RQ/GT1)
AD4	12	29	WR (LOCK)
AD3	13	28	I0/M (S2)
AD2	14	27	DT/R (S1)
AD1	15	26	DEN (S0)
AD0	16	25	ALE (QS0)
NMI	17	24	INTA (QS1)
INTR	18	23	TEST
CLK	19	22	READY
GND	20	21	RESET

- Chế độ Min và chế độ Max:

$\overline{MN/MX} = 1$  chế độ Min  
 $= 0$  chế độ Max với bus controller 8288

# Intel 8086



		MAX MODE	{ MIN MODE }
GND	1	40	V <sub>CC</sub>
AD14	2	39	AD15
AD13	3	38	A16/S3
AD12	4	37	A17/S4
AD11	5	36	A18/S5
AD10	6	35	A19/S6
AD9	7	34	BHE/S7
AD8	8	33	MN/MX
AD7	9	32	RD
AD6	CPU	31	RQ/GTO (HOLD)
AD5	11	30	RQ/GT1 (HLDA)
AD4	12	29	LOCK (WR)
AD3	13	28	S2 (M/I/O)
AD2	14	27	S1 (DT/R)
AD1	15	26	S0 (DEN)
AD0	16	25	QS0 (ALE)
NMI	17	24	QS1 (INTA)
INTR	18	23	TEST
CLK	19	22	READY
GND	20	21	RESET



## Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Bản đồ bộ nhớ của máy tính IBM PC



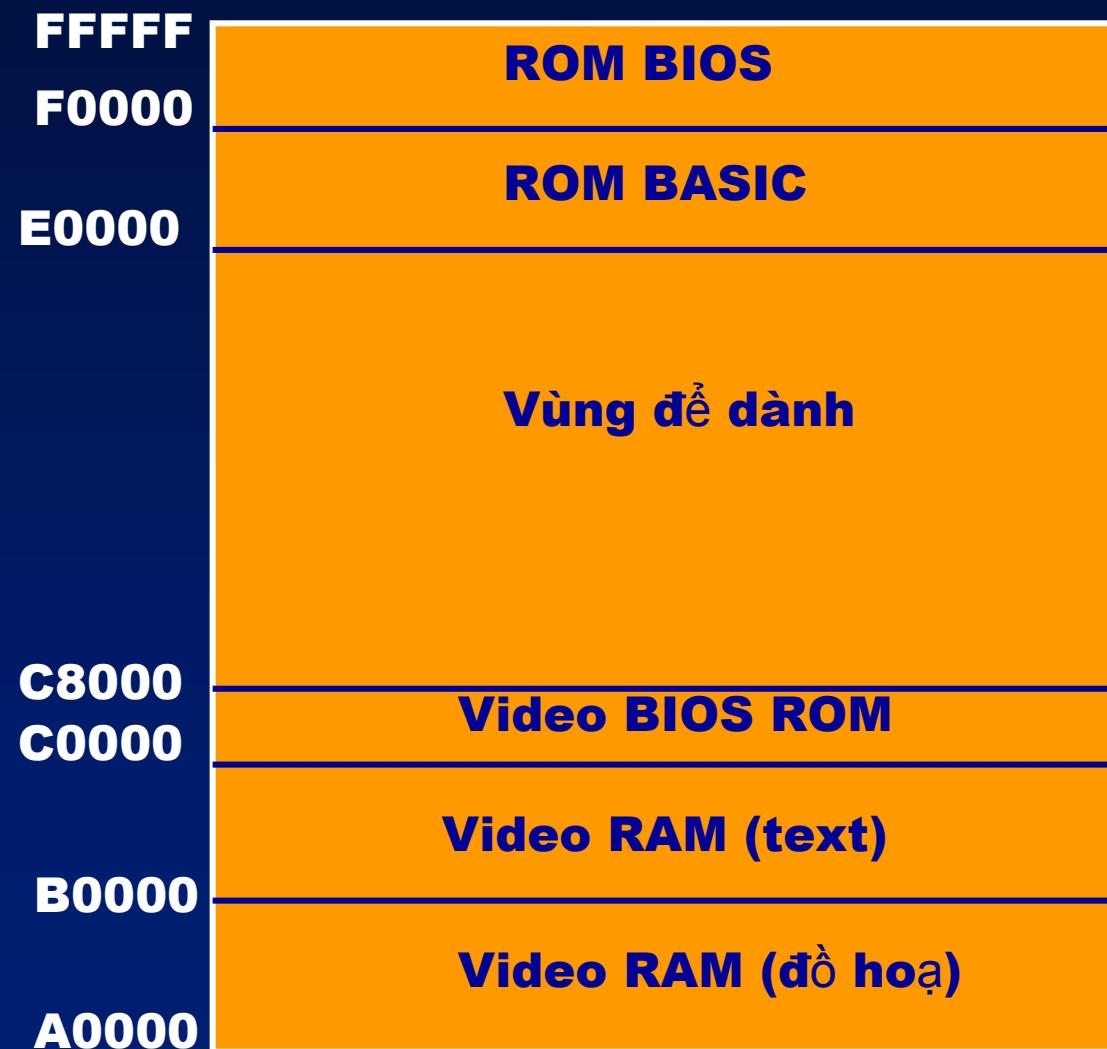


# Bản đồ vùng nhớ chương trình





# Bản đồ vùng nhớ hệ thống





# Các cổng vào ra

- Địa chỉ: 0000H – FFFFH, M/ $\overline{IO}$  = 0

FFFF	Vùng mở rộng
03F8	COM1
03F0	Điều khiển đĩa mềm
03D0	CGA adapter
0378	LPT1
0320	Điều khiển ổ cứng
02F8	COM2
0060	8255
0040	Định thời (8253)
0020	Điều khiển ngắn
0000	Điều khiển DMA

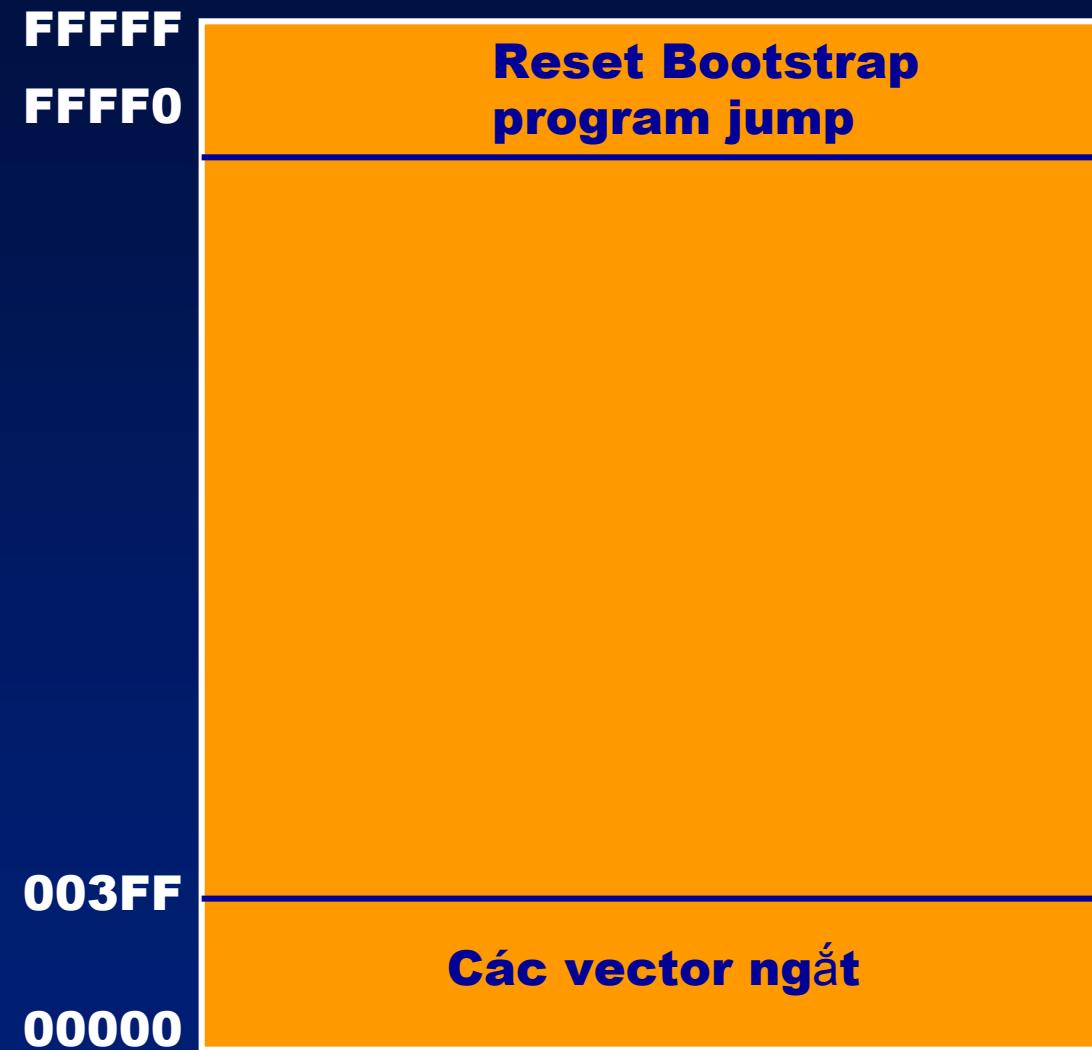


# Trình tự khởi động

- Khi bật nguồn hoặc nhấn Reset
  - CS=FFFFh và IP=0000 => địa chỉ FFFF0 chưa 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 ngắt 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



# Vùng nhớ dành riêng của 8088/8086





## Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
  - Chế độ địa chỉ thanh ghi**
  - Chế độ địa chỉ tức thì**
  - Chế độ địa chỉ trực tiếp**
  - Chế độ địa chỉ gián tiếp qua thanh ghi**
  - Chế độ địa chỉ tương đối cơ sở**
  - Chế độ địa chỉ tương đối chỉ số**
  - Chế độ địa chỉ tương đối chỉ số cơ sở**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# 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**
  - **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**



## 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 thi (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, OFF0H ; không hợp lệ**
  - MOV AX, OFF0H ;**
  - MOV DS, AX ;**
  - MOV [BX], 10 ; copy số thập phân 10 vào ô nhớ DS:BX**

## 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

# 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)



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

- Một toán hạng là thanh ghi cơ sở BX, BP và các hàng 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



## 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



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

- **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



# 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

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

- **Ví dụ:**

- 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



## Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# Cách mã hoá lệnh của 8086

**Opcode  
1-2 byte**

**MOD-REG-R/M  
0-1 byte**

**Dịch chuyển  
0-2 byte**

**Tức thì  
0-2 byte**

- Một lệnh có độ dài từ 1 đến 6 byte**





# Cách mã hoá lệnh của 8086



MOD

REG

R/M

MOD &lt;&gt; 11

- 00 không có dịch chuyển
- 01 dịch chuyển 8 bit
- 10 dịch chuyển 16 bit
- 11 R/M là thanh ghi

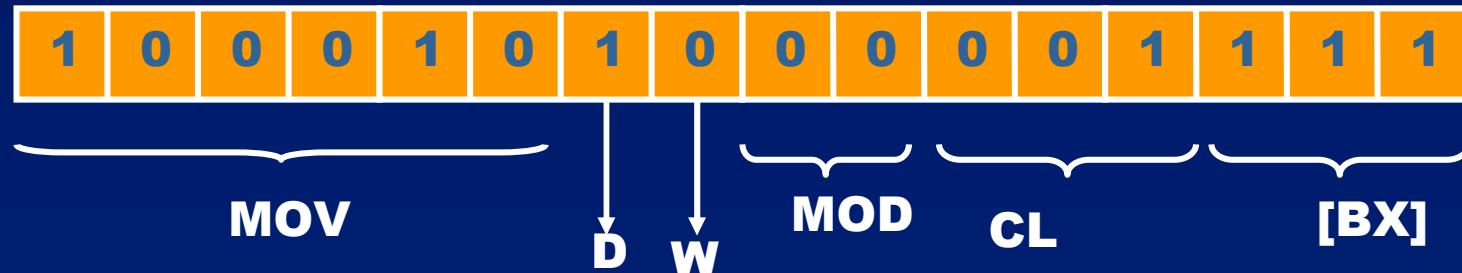
Thanh ghi	Mã
W=1	W=0
AX	AL
BX	BL
CX	CL
DX	DL
SP	AH
DI	BH
BP	CH
SI	DH

Mã	Chế độ địa chỉ
000	DS:[BX+SI]
001	DS:[BX+DI]
010	SS:[BP+SI]
011	SS:[BP+DI]
100	DS:[SI]
101	DS:[DI]
110	SS:[BP]
111	DS:[BX]



# 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+F3H], CL sang mã máy**



## Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
  - **Các lệnh di chuyển dữ liệu**
  - **Các lệnh số học và logic**
  - **Các lệnh điều khiển chương trình**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



## Chương 2: Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
- Sơ đồ chân
- Bản đồ bộ nhớ của máy tính IBM-PC
- Các chế độ địa chỉ của 8086
- Cách mã hóa lệnh của 8086
- Mô tả tập lệnh của 8086
  - Các lệnh di chuyển dữ liệu
  - Các lệnh số học và logic
  - Các lệnh điều khiển chương trình
- Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286



# Các lệnh di chuyển dữ liệu

- **MOV, XCHG, POP, PUSH, POPF, PUSHF, IN, OUT**
- **Các lệnh di chuyển chuỗi MOVS, MOVSB, MOVSW**
- **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]**



# Các lệnh di chuyển dữ liệu

- Khả năng kết hợp toán hạng của lệnh MOV**

Đích 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



# Các lệnh di chuyển dữ liệu

- **Lệnh XCHG**

- ❑ **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]**



# Các lệnh di chuyển dữ liệu

- **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]**

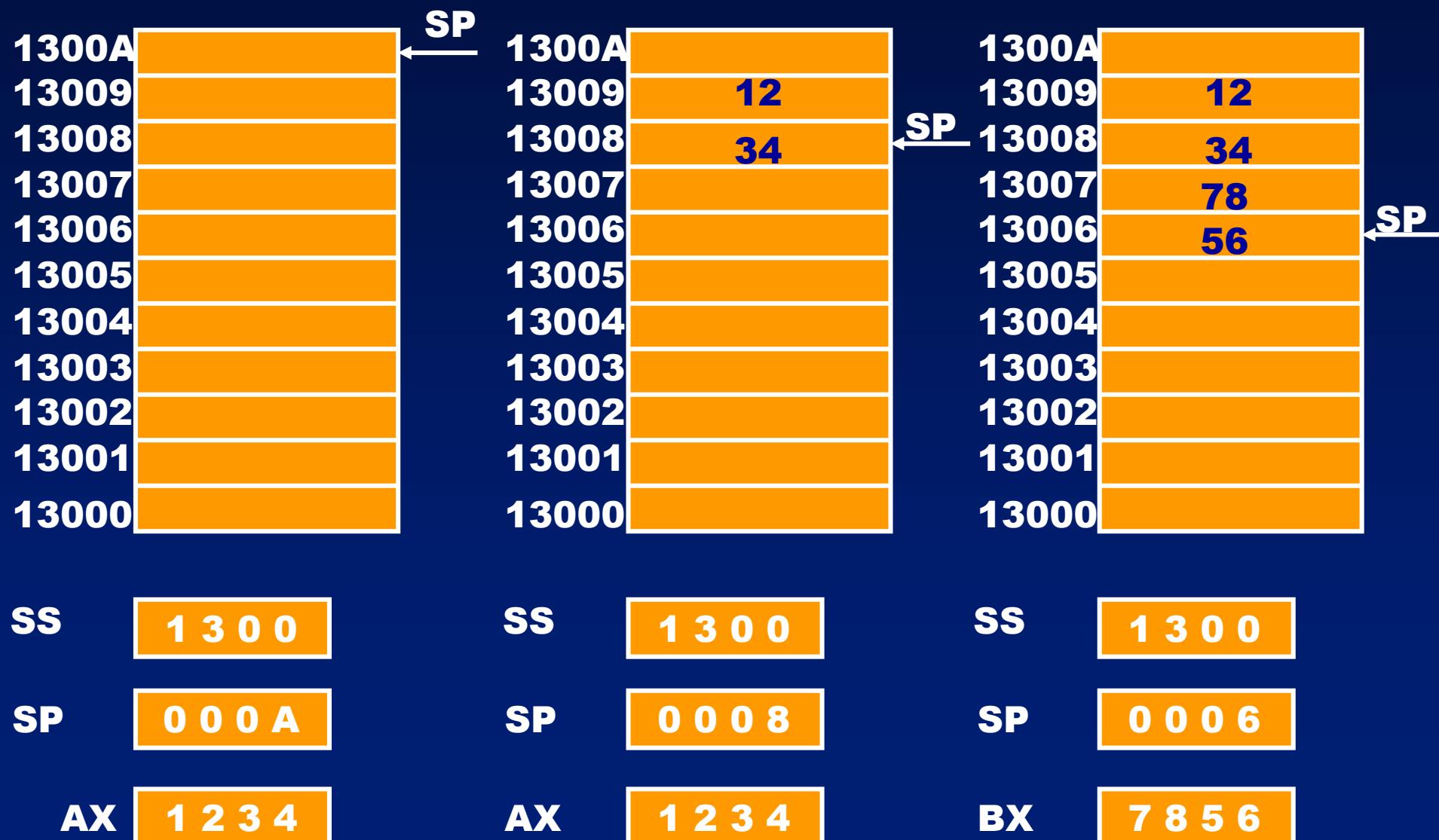
- **Lệnh PUSHF**

- ❑ **Cất nội dung của thanh ghi cờ vào ngăn xếp**



# Các lệnh di chuyển dữ liệu

- Ví dụ về lệnh PUSH





# Các lệnh di chuyển dữ liệu

- **Lệnh POP**

- **Dùng để lấy lại 1 từ vào thanh ghi hoặc ô nhớ từ đĩnh ngăn xếp**
  - **Cú pháp: POP Đích**
  - **Mô tả: {SP} => Đích, SP=SP+2**
  - **Giới hạn: thanh ghi 16 bit (trừ CS) 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 đĩnh ngăn xếp rồi đưa vào thanh ghi cờ**



# Các lệnh di chuyển dữ liệu

- Ví dụ lệnh POP

1300A	
13009	12
13008	34
13007	78
13006	56
13005	
13004	
13003	
13002	
13001	
13000	

SP

1300A	
13009	12
13008	34
13007	78
13006	56
13005	
13004	
13003	
13002	
13001	
13000	

SP

SS	1 3 0 0
SP	0 0 0 6
DX	3 2 5 4

SS	1 3 0 0
SP	0 0 0 8
DX	7 8 5 6



# Các lệnh di chuyển dữ liệu

- **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**
    - ⇒ **IN AL, F0H**
    - ⇒ **IN AX, 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**



# Các lệnh di chuyển dữ liệu

- **Các lệnh di chuyển chuỗi MOVS, MOVSB, 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**



## Chương 2: Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
- Sơ đồ chân
- Bản đồ bộ nhớ của máy tính IBM-PC
- Các chế độ địa chỉ của 8086
- Cách mã hóa lệnh của 8086
- Mô tả tập lệnh của 8086
  - Các lệnh di chuyển dữ liệu
  - Các lệnh số học và logic
  - Các lệnh điều khiển chương trình
- Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286



# 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**



# Các lệnh số học và logic

- **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:
    - ⇒ **ADC 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**



# Các lệnh số học và logic

- **Lệnh MUL**

- Lệnh nhân số không dấu

- Cú pháp: **MUL** nguồn

- Thực hiện:

- ⇒ **AX=AL\*** nguồn 8bit

- ⇒ **DXAX=AX\*** nguồn 16bit

- Lệnh này thay đổi cờ: CF, OF

- Ví dụ:

- ⇒ **MUL BL**

- **Lệnh IMUL**

- nhân số có dấu



# Các lệnh số học và logic

- **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



# Các lệnh số học và logic

- **Lệnh INC**

- Lệnh cộng 1 vào toán hạng là thanh ghi hoặc ô nhớ
  - Cú pháp: **INC Đích**
  - Thực hiện:  $\text{Đích} = \text{Đích} + 1$
  - Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
  - Ví dụ:
    - ⇒ **INC AX**

- **Lệnh DEC**

- Lệnh trừ 1 từ nội dung một thanh ghi hoặc ô nhớ
  - Cú pháp: **DEC Đích**
  - Thực hiện:  $\text{Đích} = \text{Đích} - 1$
  - Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
  - Ví dụ:
    - ⇒ **DEC [BX]**



# Các lệnh số học và logic

- **Lệnh AND**

- **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: xác định số bù 2 của toán hạng**



# Các lệnh số học và logic

## • Lệnh CMP

- Lệnh so sánh 2 byte hoặc 2 từ
- Cú pháp: **CMP Đích, nguồn**
- Thực hiện:
  - ⇒ Đích = nguồn : CF=0 ZF=1
  - ⇒ Đích > nguồn : CF=0 ZF=0
  - ⇒ Đích < nguồn : CF=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

# Các lệnh số học và logic

- **Lệnh RCL**

- Lệnh quay trái thông qua cờ nhó
- Cú pháp: **RCL Đích, CL** (với số lần quay lớn hơn 1)

**RCLĐích, 1**

**RCL Đích, Số lần quay (80286 trỏ lên)**

- Thực hiện: quay trái đích CL lần
- Đích là thanh ghi (trừ thanh ghi đoạn) hoặc ô nhớ
- Lệnh này thay đổi cờ: CF, OF



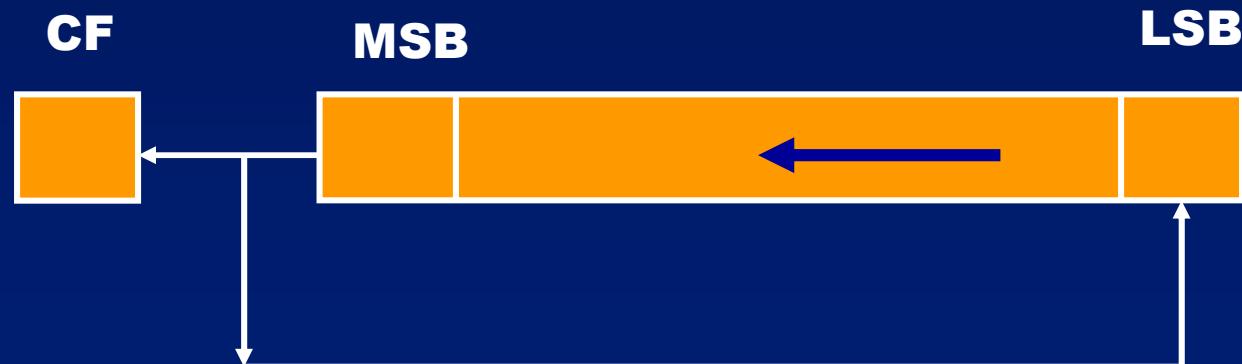
- **Lệnh RCR**

- Lệnh quay phải thông qua cờ nhó

# Các lệnh số học và logic

- **Lệnh ROL**

- Lệnh quay trái**
- Cú pháp:** **ROL** **Đích, CL** (với số lần quay lớn hơn 1)  
**ROL** **Đích, 1**  
**ROL** **Đích, Số lần quay (80286 trả lên)**
- Thực hiện:** quay trái đích CL lần
- Đích là thanh ghi (trừ thanh ghi đoạn) hoặc ô nhớ**
- Lệnh này thay đổi cờ: CF, OF**



- **Lệnh ROR**

- Lệnh quay phải**



# Các lệnh số học và logic

- **Lệnh SAL**

- Lệnh dịch trái số học

- Cú pháp: **SAL Đích, CL** (với số lần dịch lớn hơn 1)

**SAL Đích, 1**

**SAL Đích, số lần dịch (80286 trở lên)**

- Thực hiện: dịch trái đích CL bit tương đương với  
 $\text{Đích} = \text{Đích} * 2^{\text{CL}}$

- Lệnh này thay đổi cờ SF, ZF, PF



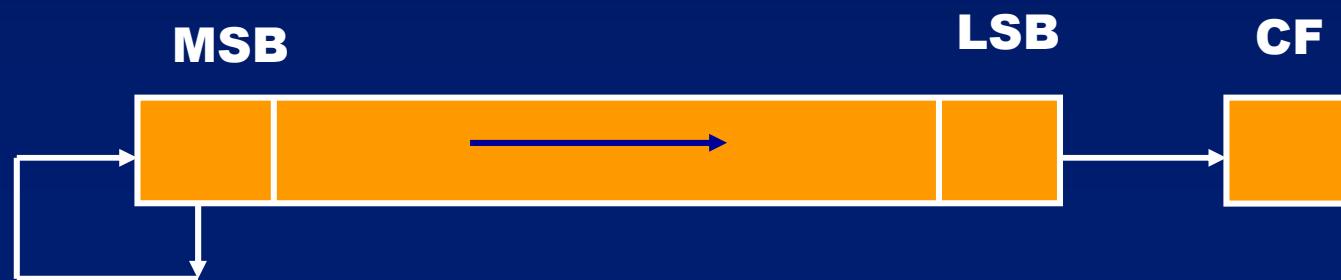
- **Lệnh SHL**

- Lệnh dịch trái logic tương tự như SAL

# Các lệnh số học và logic

- **Lệnh SAR**

- Lệnh dịch phải số học
- Cú pháp: **SAR Đích, CL** (với số lần dịch lớn hơn 1)  
**SAR Đích, 1**  
hoặc **SAR Đích, số lần dịch (80286 trở lên)**
- Thực hiện: dịch phải đích CL bit
- Lệnh này thay đổi cờ SF, ZF, PF, CF mang giá trị của MSB



# Các lệnh số học và logic

- **Lệnh SHR**

- Lệnh dịch phải logic

- Cú pháp: **SHR Đích, CL** (với số lần dịch lớn hơn 1)

**SHR Đích, 1**

hoặc **SHR Đích, số lần dịch** (80286 trả lên)

- Thực hiện: dịch phải đích CL bit

- Lệnh này thay đổi cờ SF, ZF, PF, CF mang giá trị của LSB



**Chú ý:**

**Trong các lệnh dịch và quay, toán hạng không được là thanh ghi đoạn**



## Chương 2: Bộ vi xử lý Intel 8088/8086

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
  - **Các lệnh di chuyển dữ liệu**
  - **Các lệnh số học và logic**
  - Các lệnh điều khiển chương trình**
    - ⇒ **Lệnh nhảy không điều kiện: JMP**
    - ⇒ **Lệnh nhảy có điều kiện JE, JG, JGE, JL, JLE...**
    - ⇒ **Lệnh lặp LOOP**
    - ⇒ **Lệnh gọi chương trình con CALL**
    - ⇒ **Lệnh gọi chương trình con phục vụ ngắt INT và IRET**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



# 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:

E B

Độ lệch

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

- ⇒ Thực hiện: IP=IP + độ lệch

- ⇒ Ví dụ:

XOR BX, BX

Nhan: MOV AX, 1

ADD AX, BX

JMP SHORT Nhan



# Lệnh nhảy không điều kiện JMP

## ☐ Lệnh nhảy gần (near jump)

- ⇒ Phạm vi nhảy:  $\pm 32$  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:  $IP=IP + \text{độ lệch}$

$IP=BX$

$IP=[BX+1] [BX]$

E 9

Dộ lệchLo

Dộ lệchHi

Nhảy gián tiếp



# Lệnh nhảy không điều kiện JMP

## ☐ Lệnh nhảy xa (far jump)

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



⇒ 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**

XOR CX, CX

MOV AX, 1

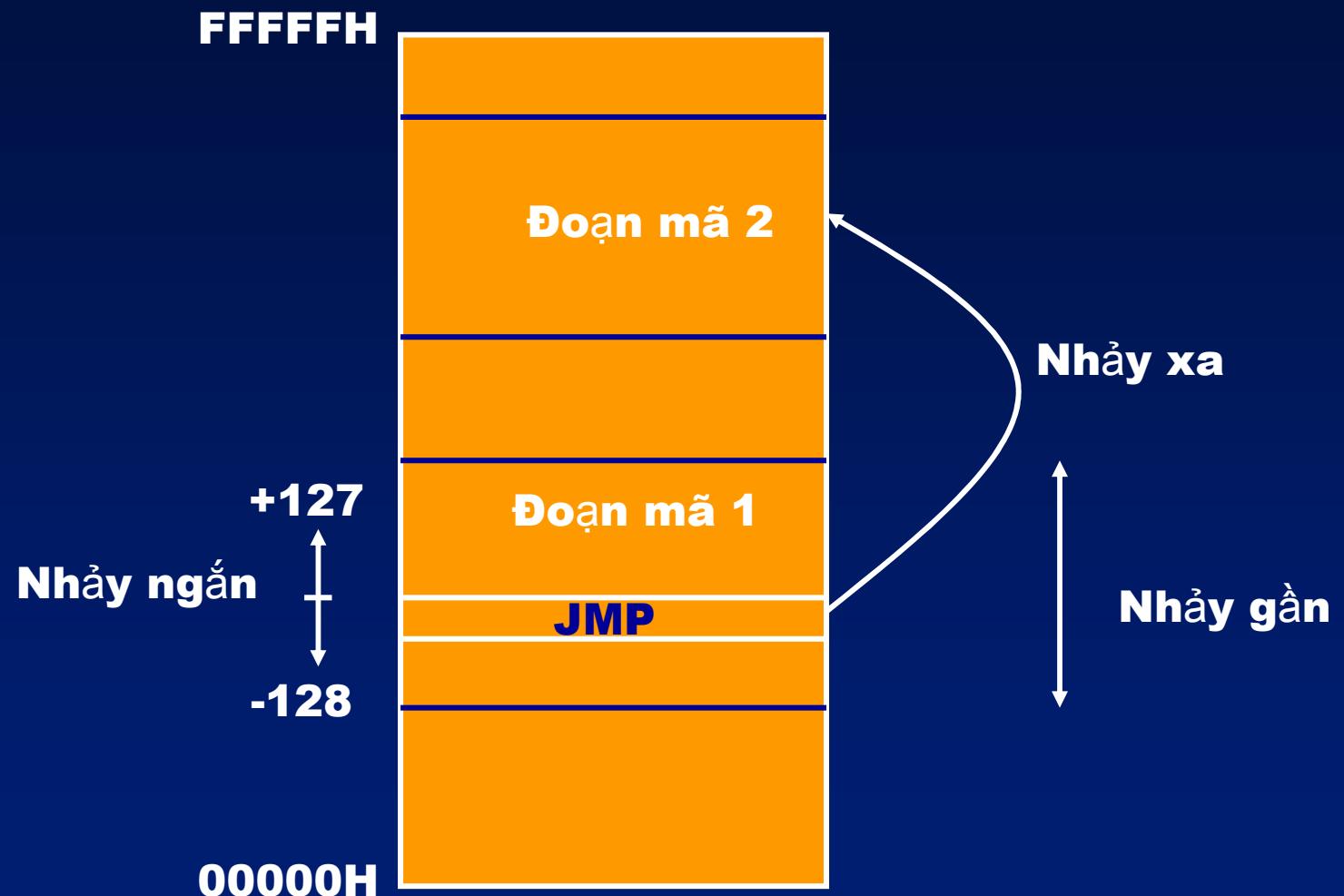
ADD AX, BX

JMP DWORD PTR **[BX]**

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

IP = [BX+1][BX]  
CS= [BX+3][BX+2]

# Tóm tắt lệnh JMP





## Lệnh nhảy có điều kiện

- JE or JZ, JNE or JNZ, JG, JGE, JL, JLE (dùng cho số có dấu) và JA, JB, JAE, JBE (dùng cho số không dấu) ...
- 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



# 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

XOR AL, AL

MOV CX, 16

Lap: INC AL

CMP AL, 10

LOOPE Lap

XOR AL, AL

MOV CX, 16

Lap: INC AL

CMP AL, 10

LOOPNE Lap

Lặp đến khi CX=0

Lặp đến khi CX=0  
hoặc AL<>10

Lặp đến khi CX=0  
hoặc AL=10



# Lệnh CALL

- **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

Tong PROC NEAR

ADD AX, BX

ADD AX, CX

RET

Tong ENDP

...

MOV BX, OFFSET Tong

CALL BX

**CALL WORD PTR [BX]**

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

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

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



# 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

**CALL DWORD PTR [BX]**

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

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



## Lệnh ngắt INT và IRET

- **INT** gọi chương trình con phục vụ ngắt (**CTCPVN**)
- **Bảng vector ngắt:** 1 Kbytes 00000H đến 003FF H
  - 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



# 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

- **Cấu trúc bên trong**
- **Sơ đồ chân**
- **Bản đồ bộ nhớ của máy tính IBM-PC**
- **Các chế độ địa chỉ của 8086**
- **Cách mã hóa lệnh của 8086**
- **Mô tả tập lệnh của 8086**
- **Cách đánh địa chỉ ở chế độ bảo vệ ở các máy tính từ 80286**



## Đánh địa chỉ bộ nhớ ở chế độ bảo vệ

- Cho phép truy cập dữ liệu và chương trình ở vùng nhớ trên 1M
- Thanh ghi lệnh **chứa địa chỉ lệnh**
- 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



# Đánh địa chỉ bộ nhớ ở chế độ bảo vệ

15

2 1 0



**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

7	Base(B31-B24)	G	P	O	A	Limit
5	Access rights			V	L19-L16	
3		Base(B23-B16)				
2		Base(B15-B0)				
0		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



# Đánh địa chỉ bộ nhớ ở chế độ bảo vệ

- **80286**
  - **Base 24 bit:** 000000H đến FFFFFFFH (16 MB)
  - **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:**  $2 * 8K * 64K = 1GB$  bộ nhớ => 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:**  $2 * 8K * 4 GB = 64$  Terabytes bộ nhớ



# Nội dung môn học

- 1. Giới thiệu chung về bộ 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 ngắn**
- 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

- **Giới thiệu khung của 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

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



# Chương 3 Lập trình hợp ngũ với 8086

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



# Cú pháp của chương trình hợp ngữ

```

1. .Model Small          khai báo kiểu kích thước bộ nhớ
2. .Stack 100            khai báo đoạn ngắn xếp
3. .Data                 khai báo đoạn dữ liệu
4. Tbaor 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
12. LEA      DX, MGB
13. DEC      BX
14. LAP:    MOV SI, DX
15.           MOV      CX, BX
16.           MOV      DI, SI
17.           MOV      AL, [DI]
18. TIMMAX: INC SI
19.           CMP      [SI], AL
20.           JNG      TIEP
21.           MOV      DI, SI
22.           MOV AL, [DI]
23.           MOV AL, [DI]
24. TIEP:   LOOP TIMMAX
25.           CALL DOICHO
26.           DEC      BX
27.           JNZ      LAP
28.           MOV AH, 9
29.           MOV      DX, Tbaor
30.           INT      21H
31.           MOV AH, 4CH
32.           INT      21H
33. MAIN Endp
34. DOICHO Proc          chú thích bắt đầu bằng dấu ;
35.           PUSH AX
36.           MOV      AL, [SI]
37.           XCHG
38.           MOV      [SI], AL
39.           POP      AX
40.           RET
41. DOICHO Endp          kết thúc chương trình chính
42. END MAIN              bắt đầu chương trình con
                           → kết thúc đoạn mã

```

Annotations from the original image:

- Line 1: 'khai báo kiểu kích thước bộ nhớ' (declaration of memory size)
- Line 2: 'khai báo đoạn ngắn xếp' (declaration of short stack)
- Line 3: 'khai báo đoạn dữ liệu' (declaration of data segment)
- Line 8: 'bắt đầu chương trình chính' (beginning of main program)
- Line 34: 'chú thích bắt đầu bằng dấu ;' (comment starting with a semicolon)
- Line 41: 'kết thúc chương trình chính' (end of main program)
- Line 42: 'bắt đầu chương trình con' (beginning of child program)
- Line 42: 'kết thúc đoạn mã' (end of code segment)



# 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 :**

**TWO\_WORD**

**?1**

**two-word**

**.@?**

**1word**

**Let's\_go**



# Chương 3 Lập trình hợp ngũ với 8086

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



# 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'**



# Chương 3 Lập trình hợp ngũ với 8086

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



## Biến và hằng

- **DB (define byte): định nghĩa biến kiểu byte**
- **DW (define word): định nghĩa biến kiểu từ**
- **DD (define double word): định nghĩa biến kiểu từ kép**
- **Biến byte:**
  - **Tên DB giá\_trị\_khởi đầu**
  - **Ví dụ:**

⇒ <b>B1 DB 4</b>	<b>MOV AL, B1</b>
⇒ <b>B1 DB ?</b>	<b>LEA BX, B1</b>
⇒ <b>C1 DB '\$'</b>	<b>MOV AL, [BX]</b>
⇒ <b>C1 DB 34</b>	



# Biến và hằng

- **Biến tự:**

**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



# Biến và hàng

- 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

**MOV AL, M1 ; copy 1 vào AL**  
**MOV AH, M1[2]**  
**MOV BX, 1**  
**MOV SI, 1**  
**MOV CL, M1[BX+SI]**  
**MOV AX, Word Ptr M1[BX+SI+2]**  
**MOV DL, M1[BX][SI]**



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



# Chương 3 Lập trình hợp ngũ với 8086

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



# Khung của chương trình hợp ngữ

- Khai báo quy mô sử dụng bộ nhớ
  - .MODEL Kiểu kích thước bộ nhớ**
  - Ví dụ: .Model Small**

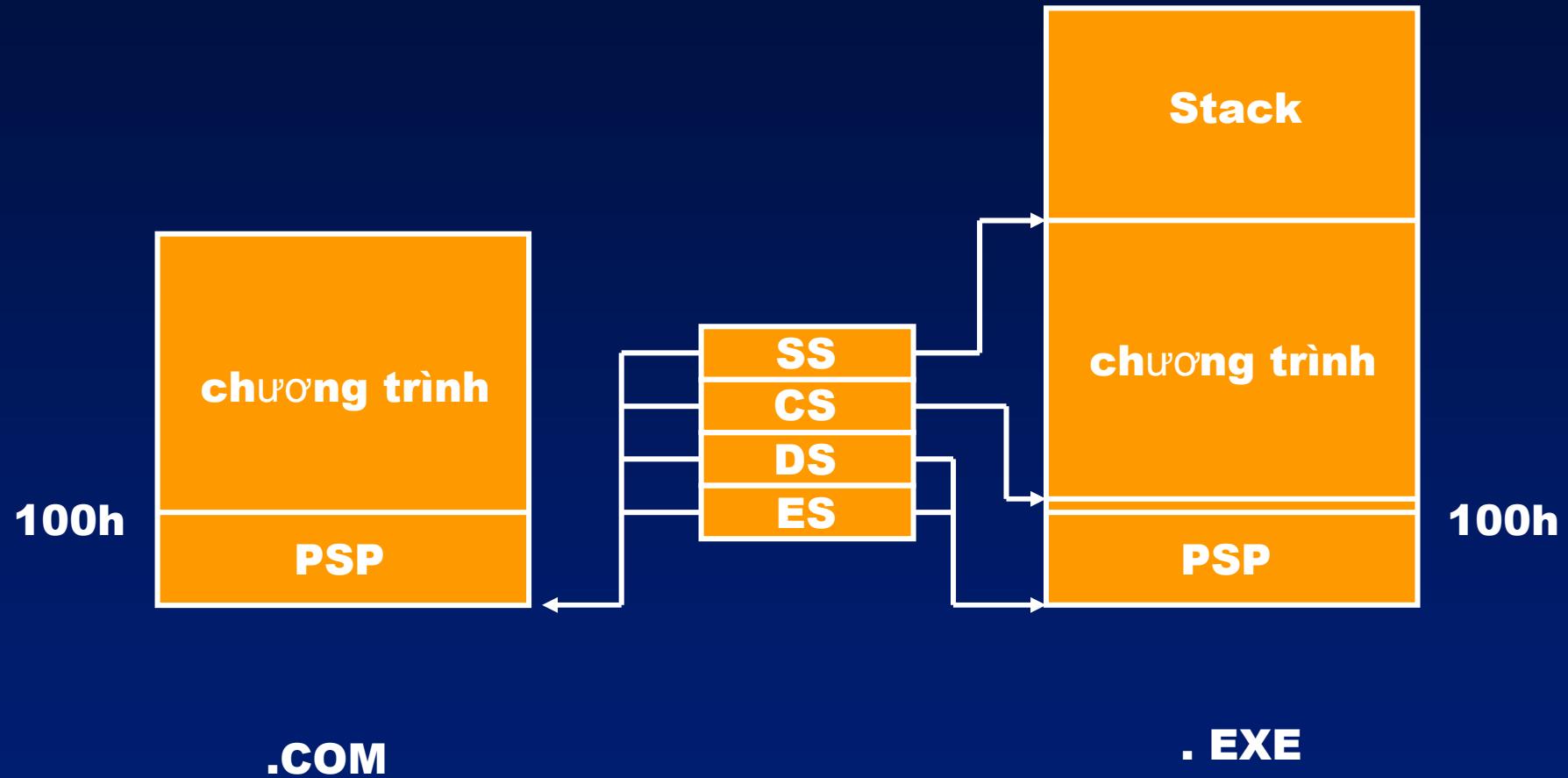
Kiểu	Mô tả
Tiny (hẹp)	mã lệnh và dữ liệu gói gọn trong một đoạn
Small (nhỏ)	mã lệnh nằm trong 1 đoạn, dữ liệu 1 đoạn
Medium (tB)	mã lệnh nằm trong nhiều đoạn, dữ liệu 1 đoạn
Compact (gọn)	mã lệnh nằm trong 1 đoạn, dữ liệu trong nhiều đoạn
Large (lớn)	mã lệnh nằm trong nhiều đoạn, dữ liệu trong nhiều 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



# 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 1KB**
- **Khai báo đoạn dữ liệu:**
  - ❑ **.Data**
  - ❑ **Khai báo các biến và hằng**
- **Khai báo đoạn mã**
  - ❑ **.Code**

# Khung của chương trình hợp ngữ





# 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
```



# 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
    ;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 dùng hàm 4CH của INT 21H
    MOV      AH, 4CH
    INT      21H
    Endp
END MAIN
```



# Khung của chương trình hợp ngữ

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

.Model Tiny

.Code

ORG 100h

START: JMP CONTINUE

;các định nghĩa cho biến và hằng

CONTINUE:

MAIN Proc

;các lệnh của chương trình

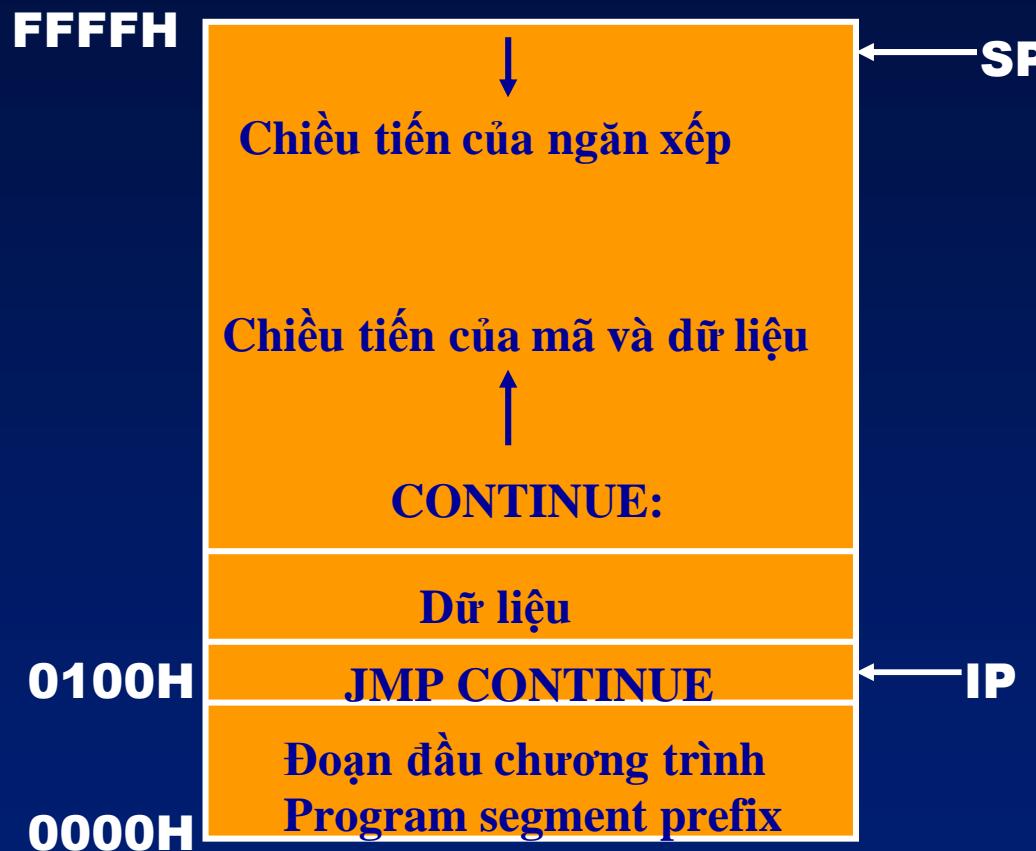
INT 20H ;trở về DOS

MAIN Endp

;các chương trình con nếu có

END START

# Khung của chương trình hợp ngữ





# 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
```



# Chương 3 Lập trình hợp ngữ với 8086

- **Giới thiệu khung của 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**



# Cách tạo một chương trình hợp ngữ

Tạo ra tệp văn bản của chương trình  
\*.asm

Dùng MASM để dịch ra mã máy  
\*.obj

Dùng LINK để nối tệp .obj thành  
\*.exe

Dịch được ra .com?

không

Dùng exe2bin để dịch \*.exe thành  
\*.com

chạy chương trình



# Chương 3 Lập trình hợp ngữ với 8086

- **Giới thiệu khung của 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**
  - **Cấu trúc lựa chọn**
  - **Cấu trúc lặp**
- **Một số chương trình cụ thể**



# 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
```



# Câu 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:
```



# 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**

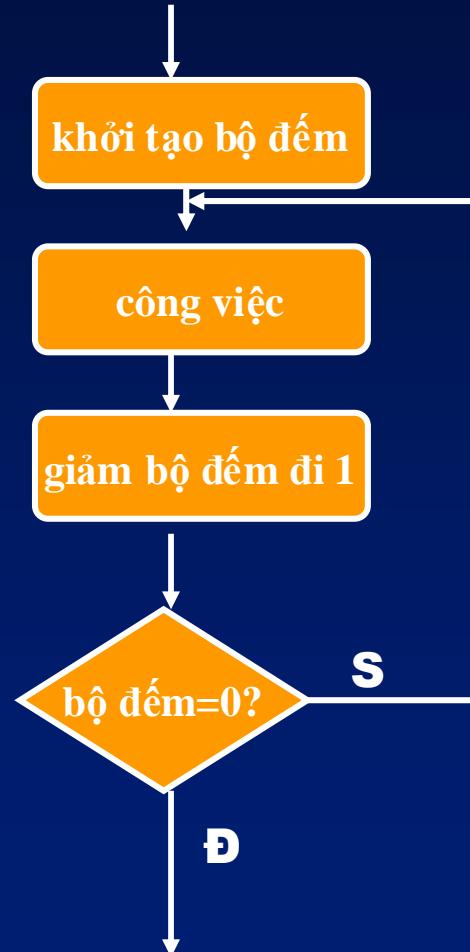
<b>CMP</b>	<b>AX, 0 ;</b>
<b>JL</b>	<b>AM ; AX&lt;0</b>
<b>JE</b>	<b>Khong ; AX=0</b>
<b>JG</b>	<b>DUONG; AX&gt;0</b>
<b>AM: MOV</b>	<b>CX, -1</b>
<b>JMP</b>	<b>End_case</b>
<b>Khong: MOV</b>	<b>CX, 0</b>
<b>JMP</b>	<b>End_case</b>

**DUONG: MOV CX, 1**

**End\_case:**

# Cấu trúc lặp FOR-DO

- For số lần lặp Do công việc

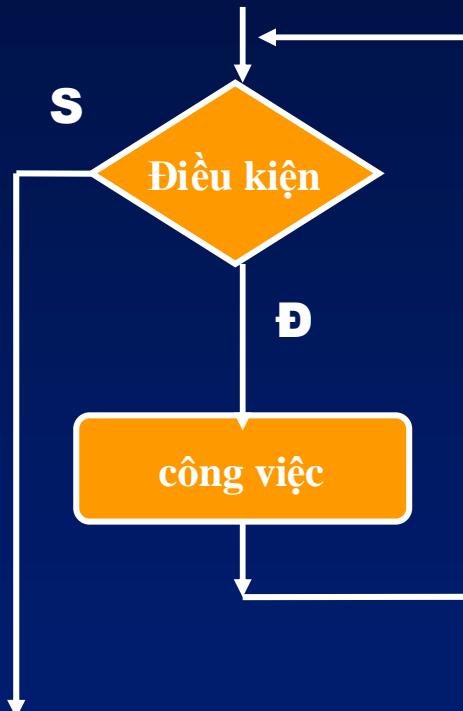


ví dụ: Hiển thị một dòng ký tự \$ trên màn hình

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	

# Cấu trúc lặp While-DO

- **While điều kiện Do công việc**

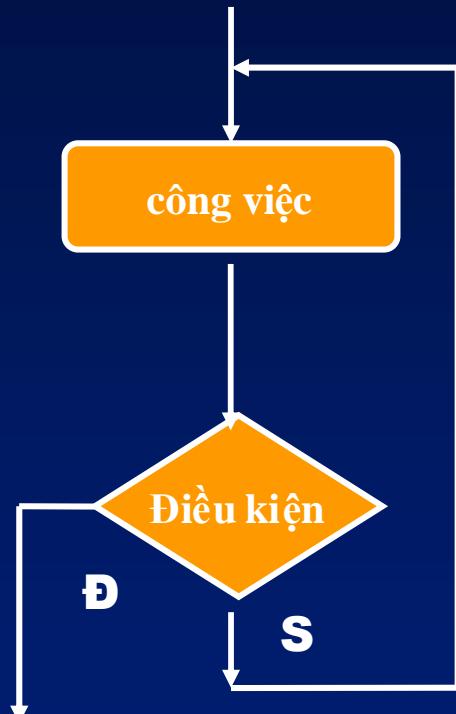


ví dụ: **đếm số ký tự đọc được c từ bàn phím, khi gặp ký tự CR thì thôi**

<pre> XOR CX, CX           ;CX=0 MOV AH,1             ;hàm đọc ký tự từ bàn phím TIEP: INT 21H         ;đọc một ký tự vào AL                   CMP AL, 13   ;đọc CR?                   JE End_while ;đúng, thoát                   INC CX        ;sai, thêm 1 ký tự vào tổng                   JMP TIEP      ;đọc tiếp End_while: </pre>	
--	--

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



## Chương 3 Lập trình hợp ngữ với 8086

- **Giới thiệu khung của 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**



# 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 lệnh của chuỗi ký tự cần hiển thị**
  - **Hàm 4CH: kết thúc chương trình loại .exe**
    - ⇒ **Vào: AH=4CH**



# Một số chương trình cụ thể

- **Ví dụ 1: Lập chương trình yêu cầu người sử dụng gõ vào một chuỗi cái thường và hiển thị dạng chữ hoa và mã ASCII dưới dạng nhị phân của chữ cái đó lên màn hình**

□ **Ví dụ:**

- ⇒ **Hay nhap vao mot chu cai thuong: a**
- ⇒ **Mã ASCII dưới dạng nhị phân của a là: 11000001**
- ⇒ **Dạng chu hoa của a là: A**

□ **Giải:**

- **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.**

□ **Giải:**

- **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.**

□ **Giải:**



# Một số chương trình cụ thể

- **Ví dụ 4: Viết chương trình cho phép nhập vào kích thước  $M*N$  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  $N*M$  và in mảng mới ra màn hình**

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

...

□ Giải:



# 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 4: Tổ chức vào ra dữ liệu

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



# Chương 4: Tổ chức vào ra dữ liệu

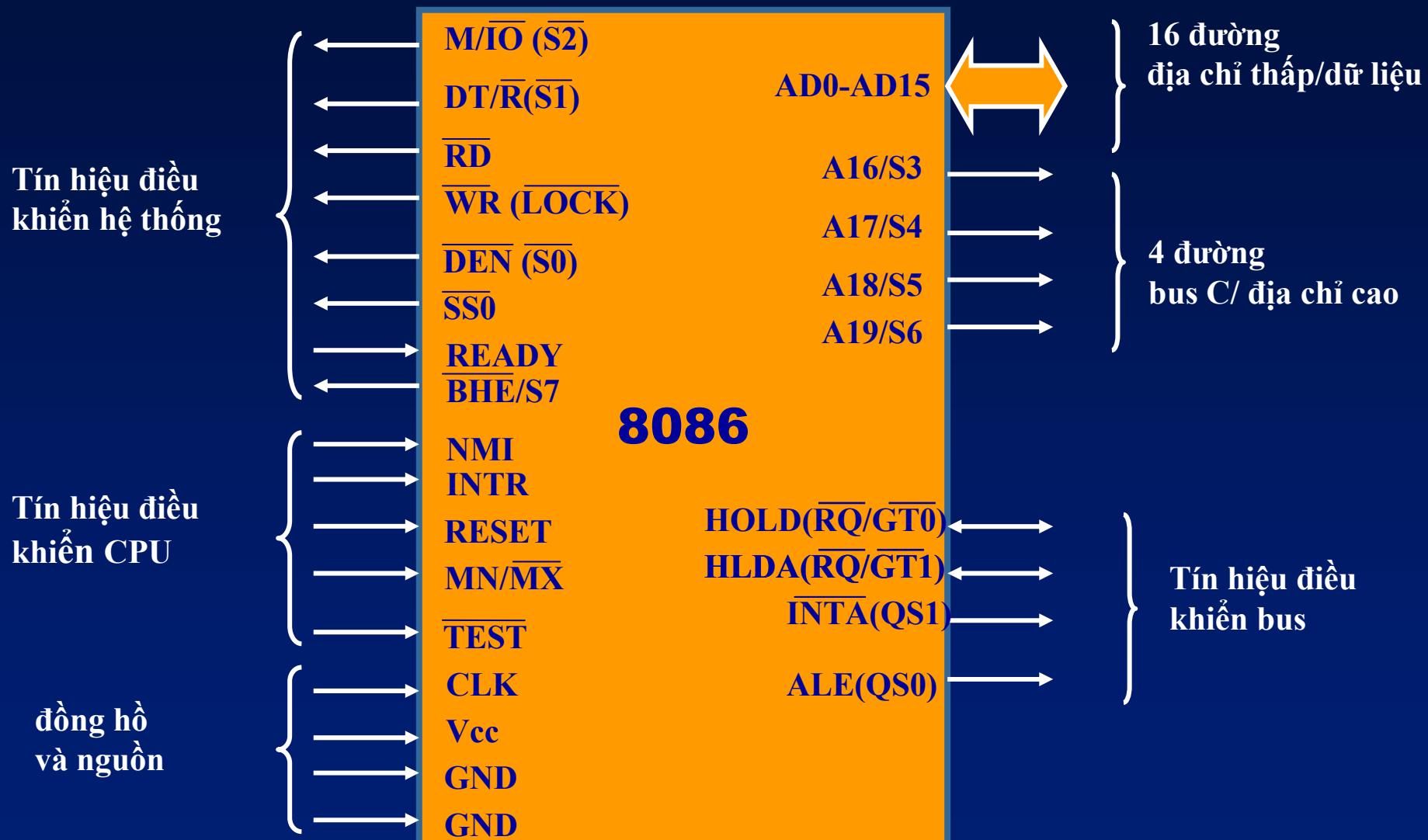
- Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
  - Các tín hiệu của 8086
  - Phân kênh và việc đệm cho các bus
  - Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288
  - Biểu đồ thời gian của các lệnh ghi/đọc
- Ghép nối 8088 với bộ nhớ
- Ghép nối 8086 với bộ nhớ
- Ghép nối với thiết bị ngoại vi



# Chương 4: Tổ chức vào ra dữ liệu

- **Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288**
  - Các tín hiệu của 8086**
  - Phân kênh và việc đệm cho các bus**
  - Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288**
  - Biểu đồ thời gian của các lệnh ghi/đọc**
- **Ghép nối 8088 với bộ nhớ**
- **Ghép nối 8086 với bộ nhớ**
- **Ghép nối với thiết bị ngoại vi**

# Các chân tín hiệu của 8086



# 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**

- **READY: input pin,**

- 0 => vi xử lý vào trạng thái đợi**
- 1: has no effect**

- **INTR: interrupt request**

- IF=1 và INTR=1=> cho phép 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



# 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<sup>O</sup>**
    - ⇒ **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**



# 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 trở kháng cao

- **HLDA (Hold Acknowledge)**

- ⇒ khi HOLD=1, HLDA=1

# Các chân tín hiệu của 8086

- Các chân ở chế độ Max**

- S2, S1, S0**

⇒ ghép nối với điều khiển bus 8288

S2	S1	S0	chu kỳ điều khiển của bus
0	0	0	chấp nhận yêu cầu ngắt
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

# Các chân tín hiệu của 8086

- **Các chân ở chế độ Max**

- $\overline{RQ}/\overline{GT0}$  và  $\overline{RQ}/\overline{GT1}$ : 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**

- $\overline{LOCK}$**

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

- $\overline{QS0}$  và  $\overline{QS1}$ :**

⇒ **trạng thái của hàng đợi lệnh**

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

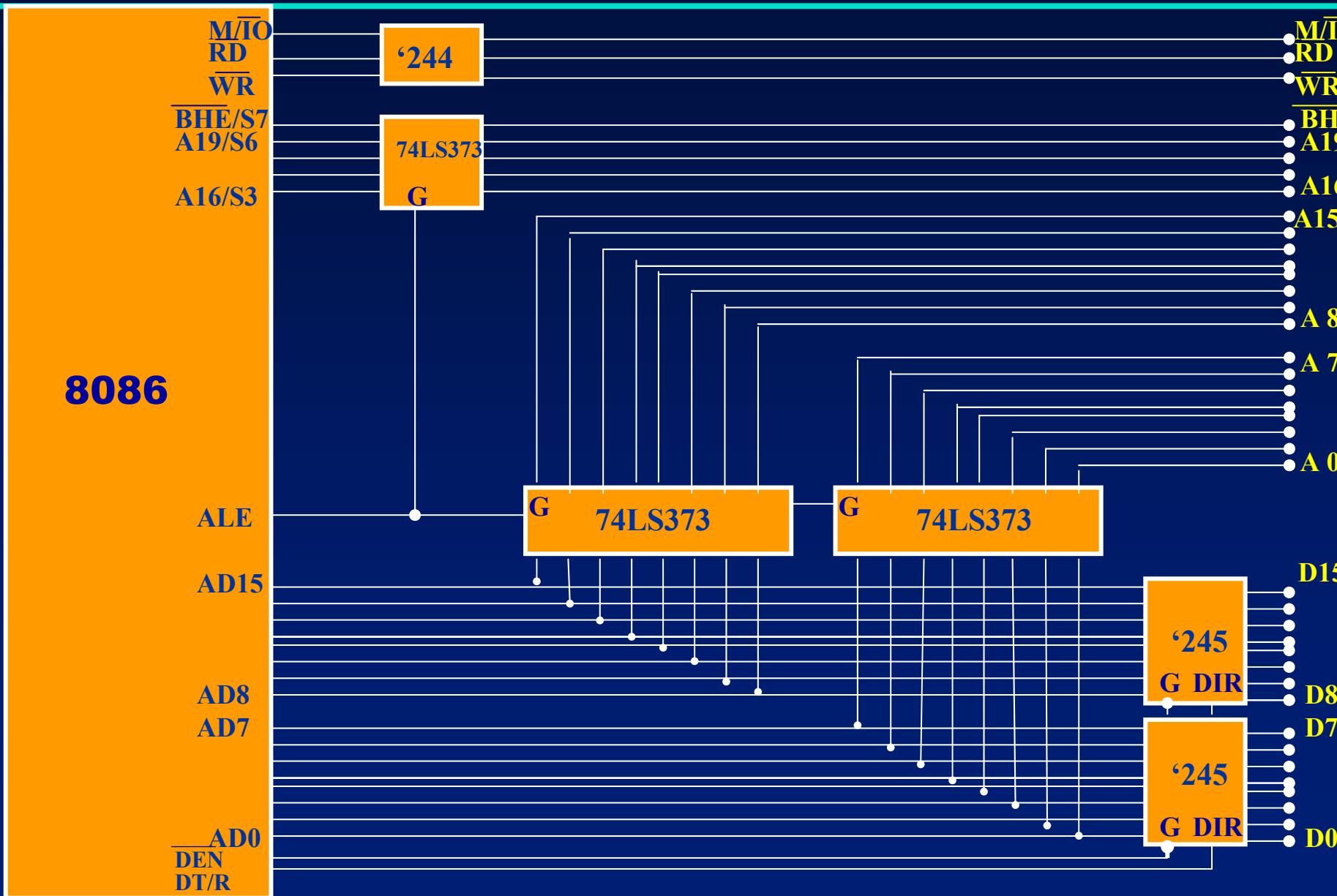
- Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
  - Các tín hiệu của 8086
  - Phân kênh và việc đệm cho các bus
  - Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288
  - Biểu đồ thời gian của các lệnh ghi/đọc
- Ghép nối 8088 với bộ nhớ
- Ghép nối 8086 với bộ nhớ
- Ghép nối với thiết bị ngoại vi



# 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

# Phân kênh và đếm cho các bus

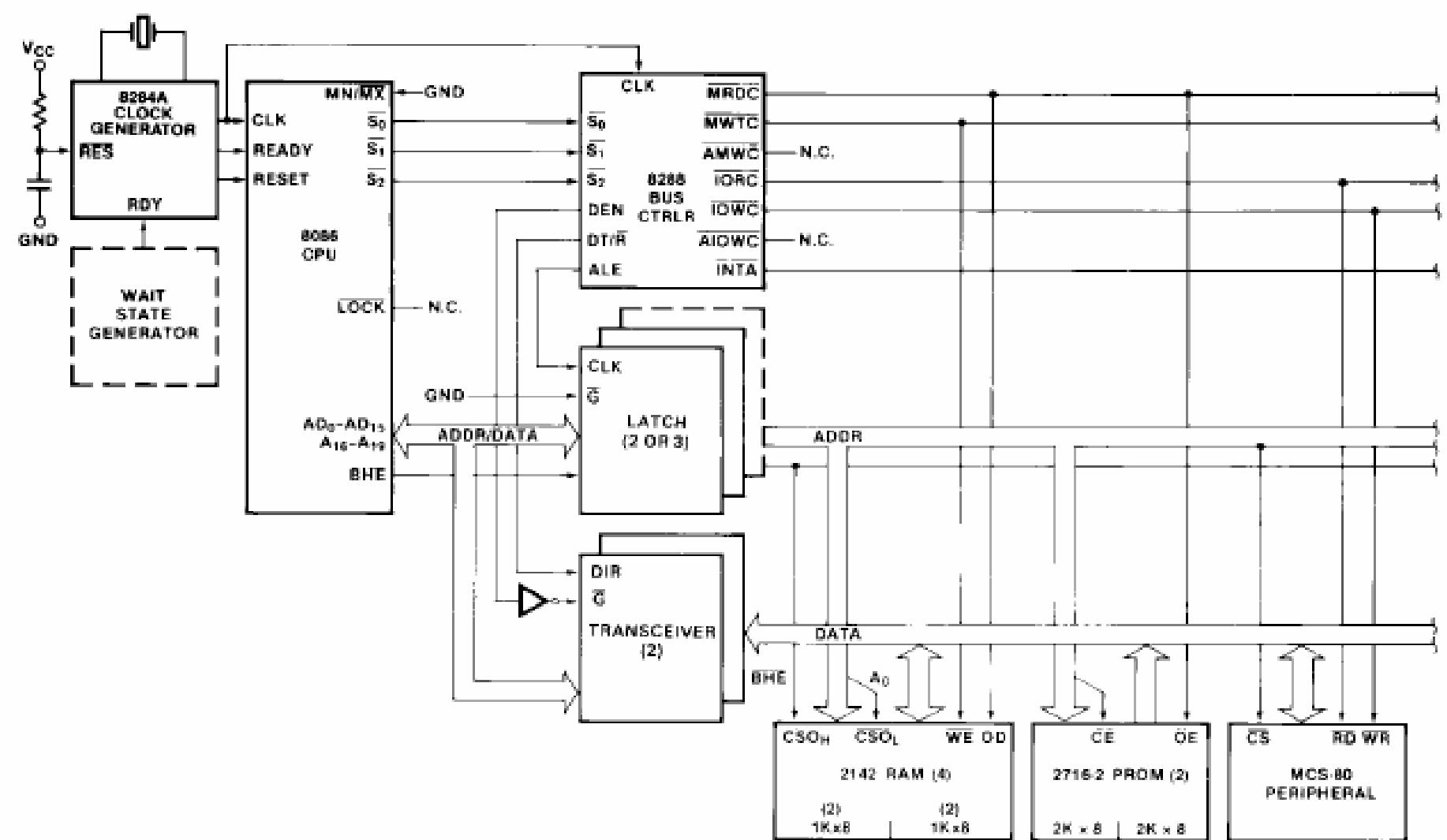




# Chương 4: Tổ chức vào ra dữ liệu

- Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
  - Các tín hiệu của 8086
  - Phân kênh và việc đệm cho các bus
  - Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288
  - Biểu đồ thời gian của các lệnh ghi/đọc
- Ghép nối 8088 với bộ nhớ
- Ghép nối 8086 với bộ nhớ
- Ghép nối với thiết bị ngoại vi

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

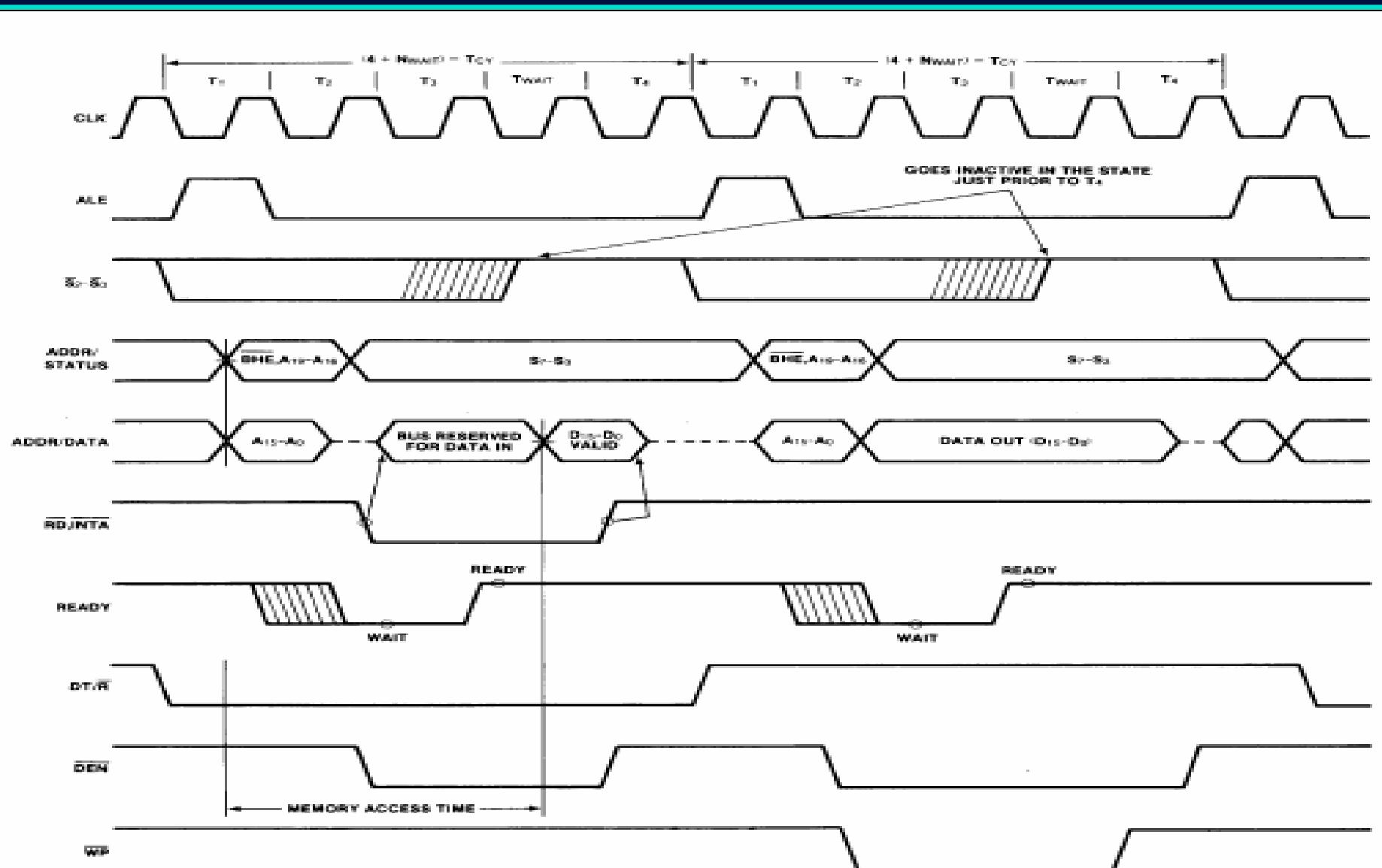




## Chương 4: Tổ chức vào ra dữ liệu

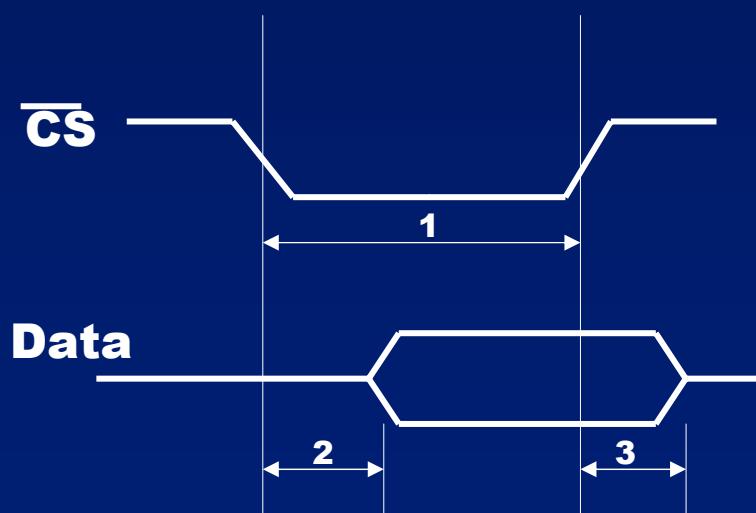
- Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
  - Các tín hiệu của 8086
  - Phân kênh và việc đệm cho các bus
  - Mạch tạo xung nhịp 8284 và mạch điều khiển bus 8288
  - Biểu đồ thời gian của các lệnh ghi/đọc
- Ghép nối 8088 với bộ nhớ
- Ghép nối 8086 với bộ nhớ
- Ghép nối với thiết bị ngoại vi

# Biểu đồ thời gian



# Biểu đồ thời gian

- Các ký hiệu trong biểu đồ thời gian:

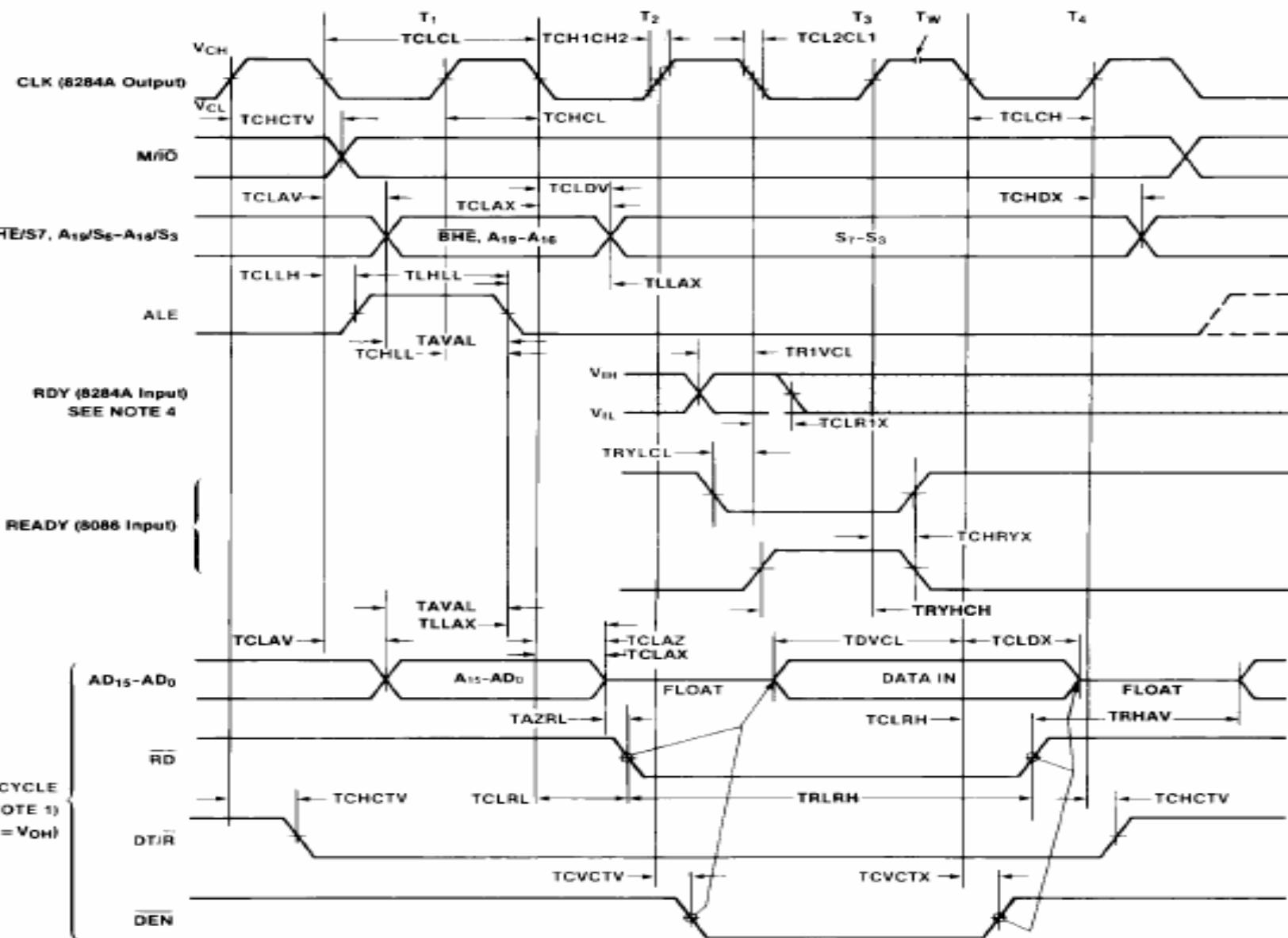


		Min	max	Units
1	CS hold time	60		ns
2	CS to data valid		30	ns
3	Data hold time	5	10	ns

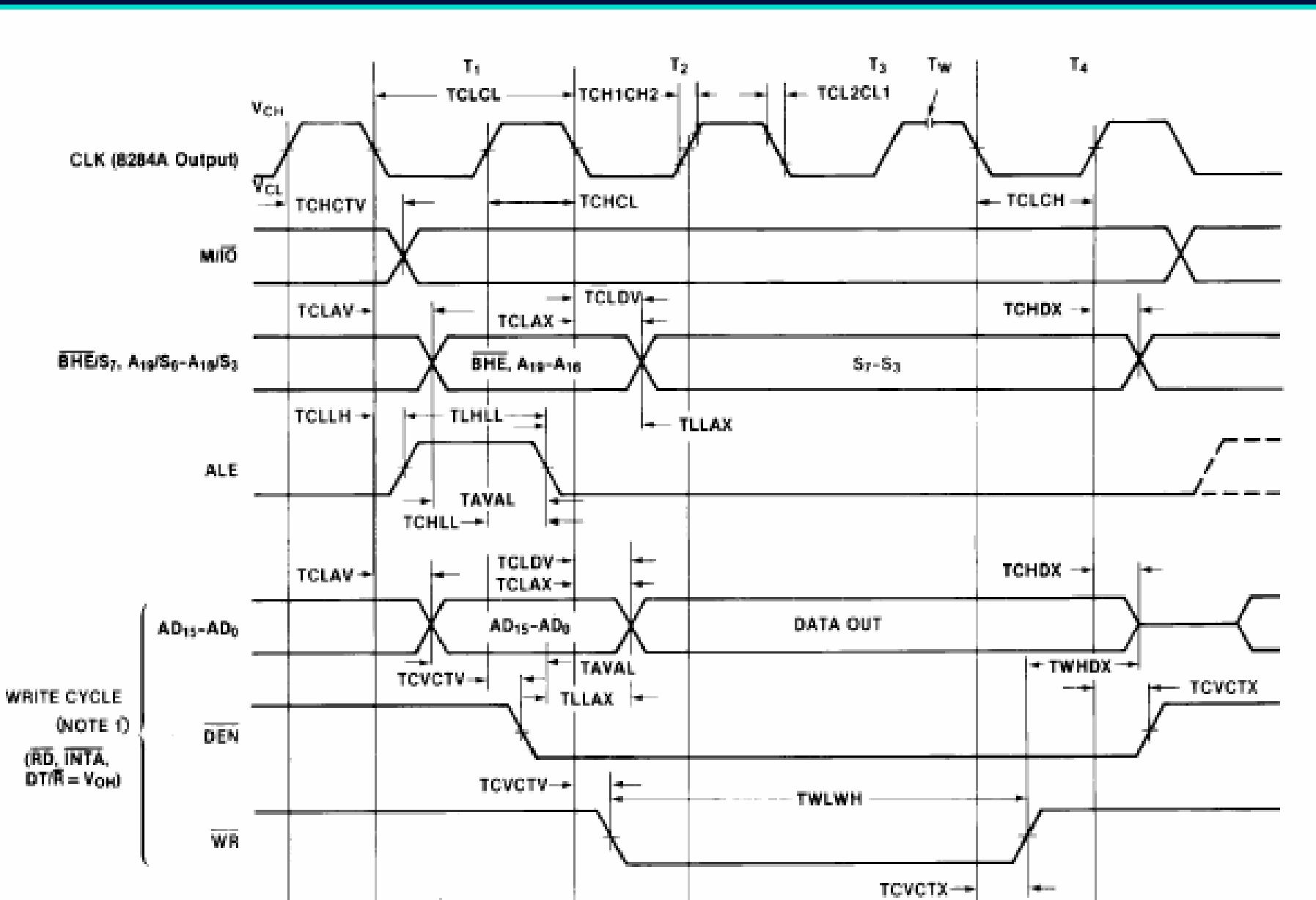
# Biểu đồ thời gian

- Một chu kỳ ghi/đọc của CPU (chu kỳ bus): 4 chu kỳ xung nhịp T
  - 5 MHz:  $4 * 200 \text{ ns} = 800 \text{ ns}$
  - 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:  $T_w = n * T$
    - ⇒ 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

# Biểu đồ thời gian



# Biểu đồ thời gian





## Chương 4: Tổ chức vào ra dữ liệu

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



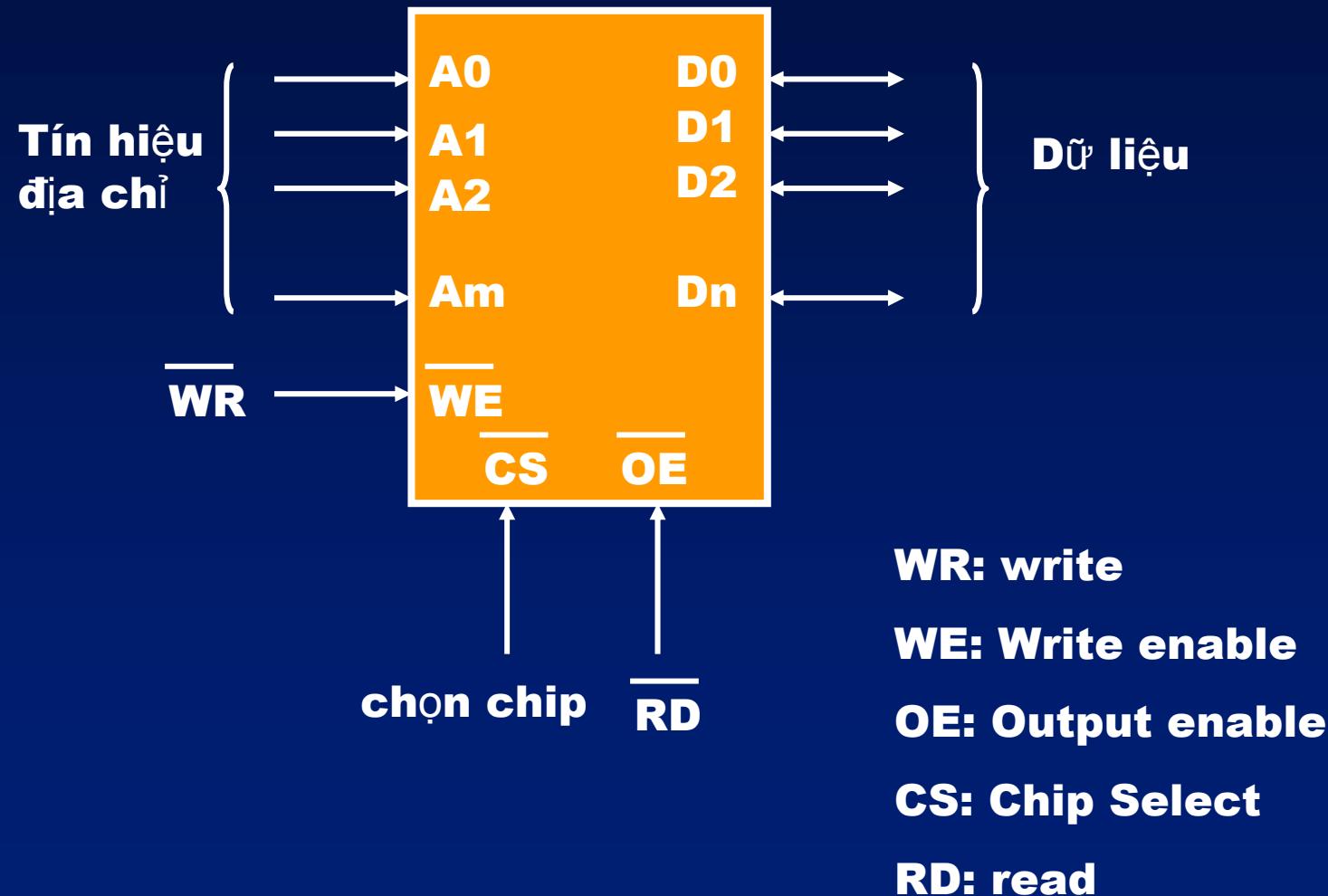
## Chương 4: Tổ chức vào ra dữ liệu

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

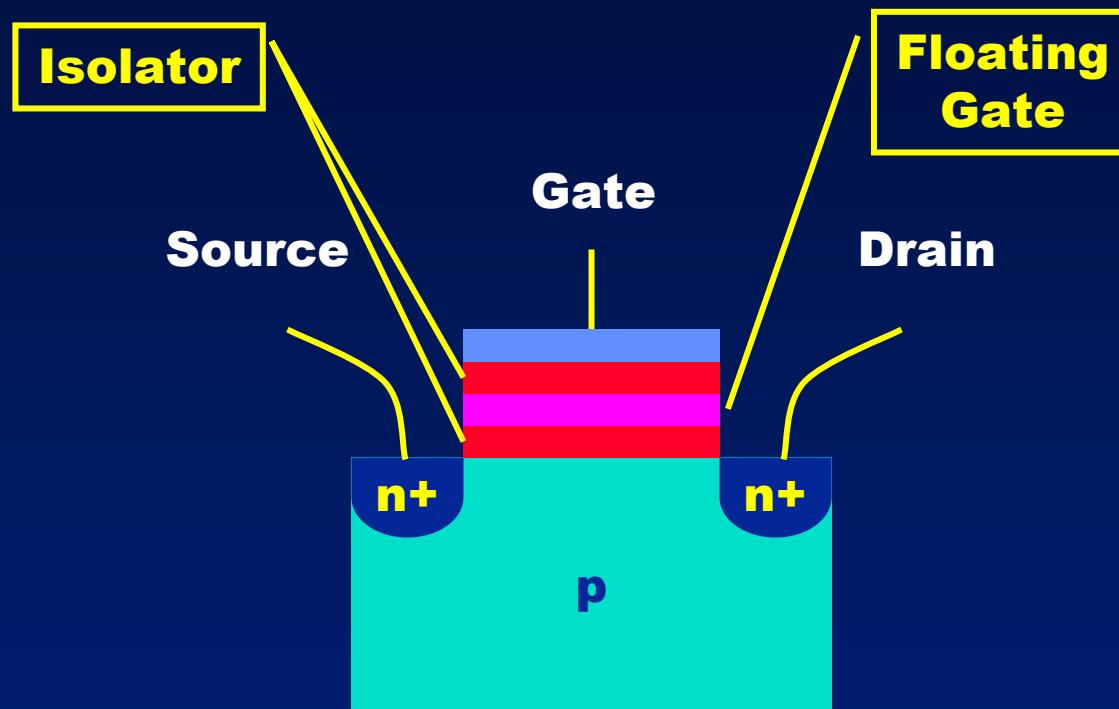
# Các loại bộ nhớ bán dẫn

- **Bộ nhớ không bị mất dữ liệu (non-volatile)**
  - ROM (Read Only Memory)**
  - PROM (Programmable ROM)**
  - EPROM (Electrically programmable ROM)**
  - Flash**
  - EEPROM (Electrically Erasable Programmable ROM)**
  - FeRAM (Ferroelectric Random Access Memory)**
  - MRAM (Magnetochemical Random Access Memory)**
- **Bộ nhớ bị mất dữ liệu (volatile)**
  - SRAM (Static RAM)**
  - SBSRAM (Synchronous Burst RAM)**
  - DRAM (Dynamic RAM)**
  - FPDRAM (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)**

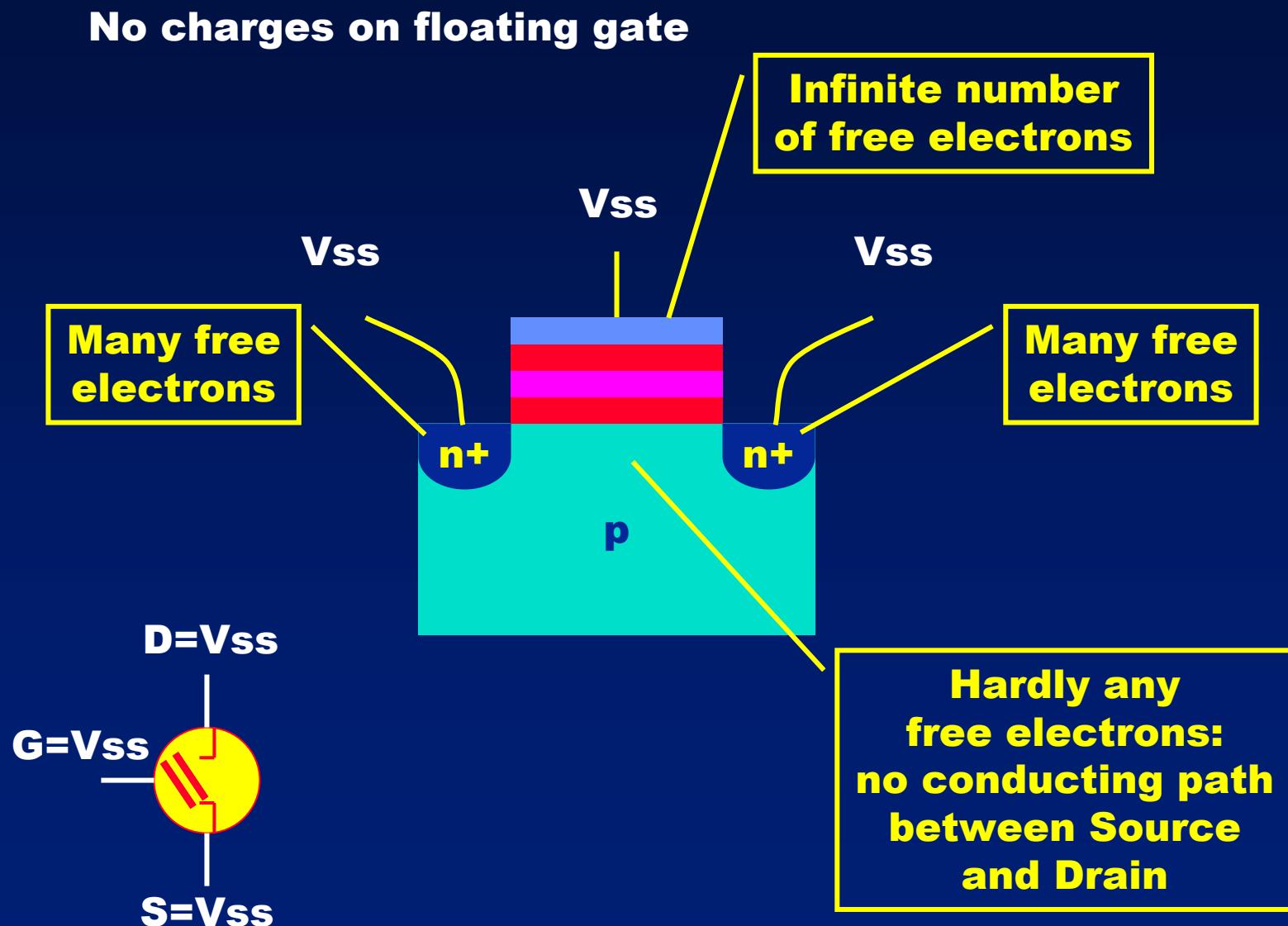
# Các loại bộ nhớ bán dẫn



# EPROM

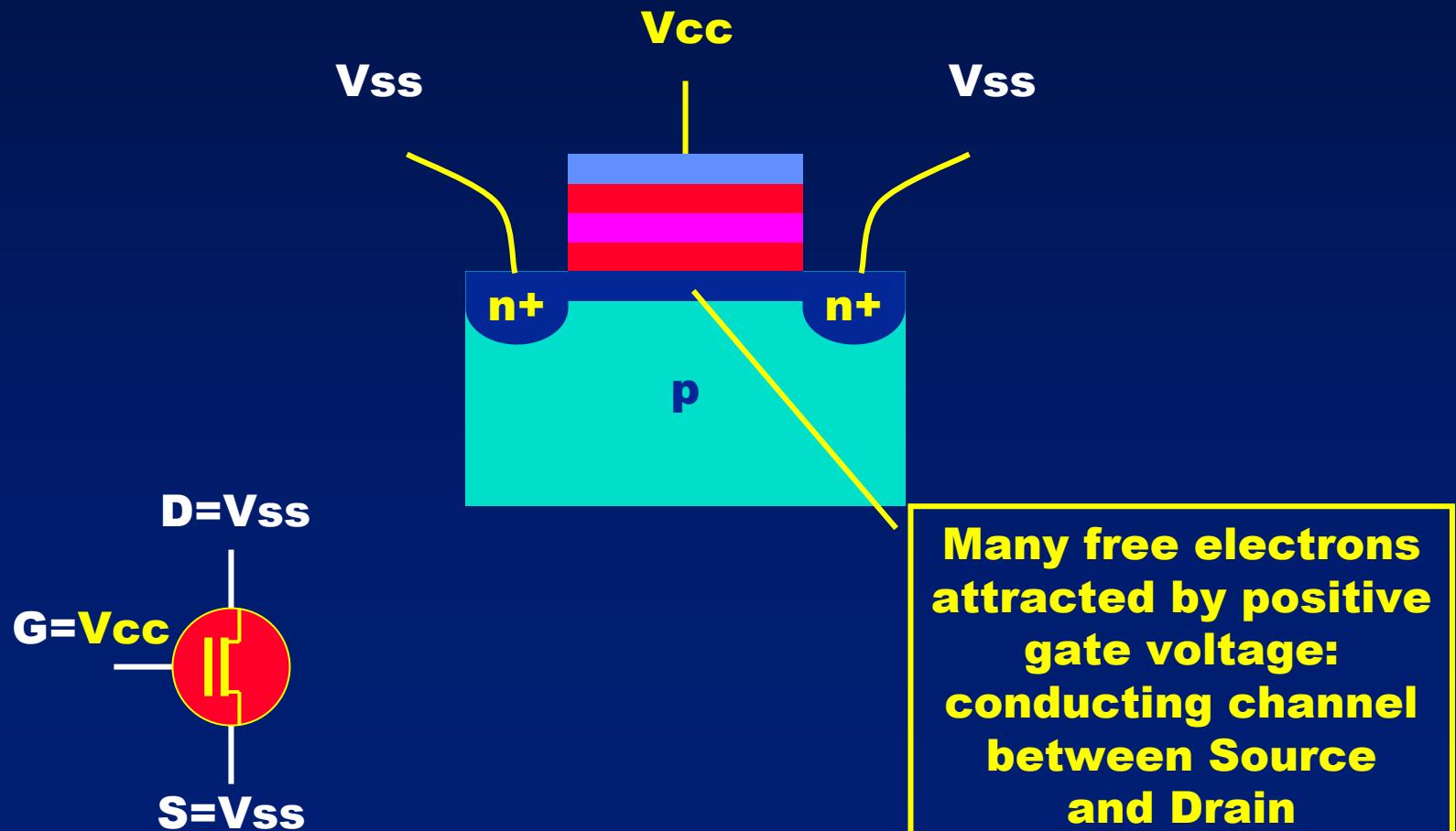


# EPROM



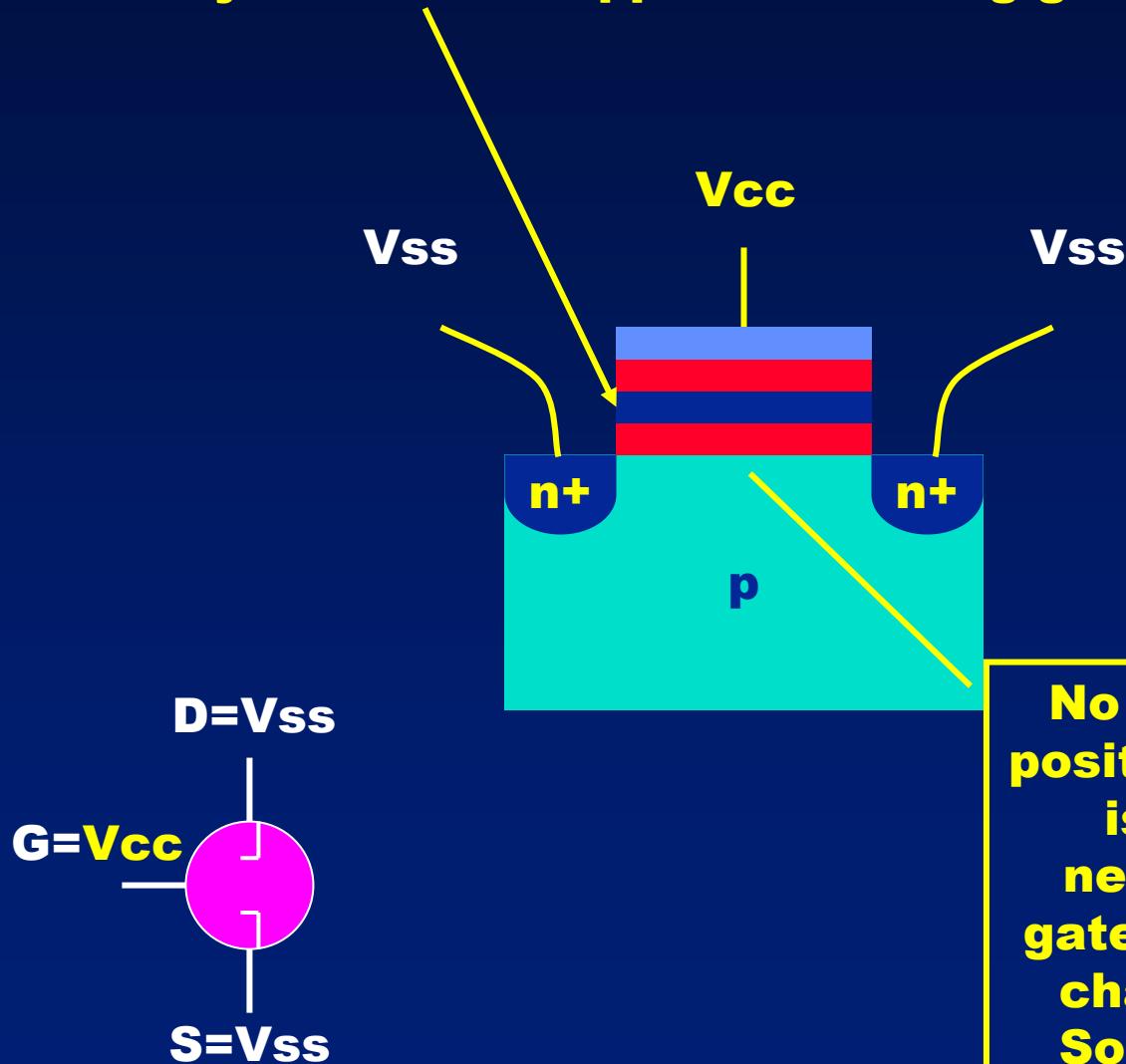
# EPROM

No charges on floating gate

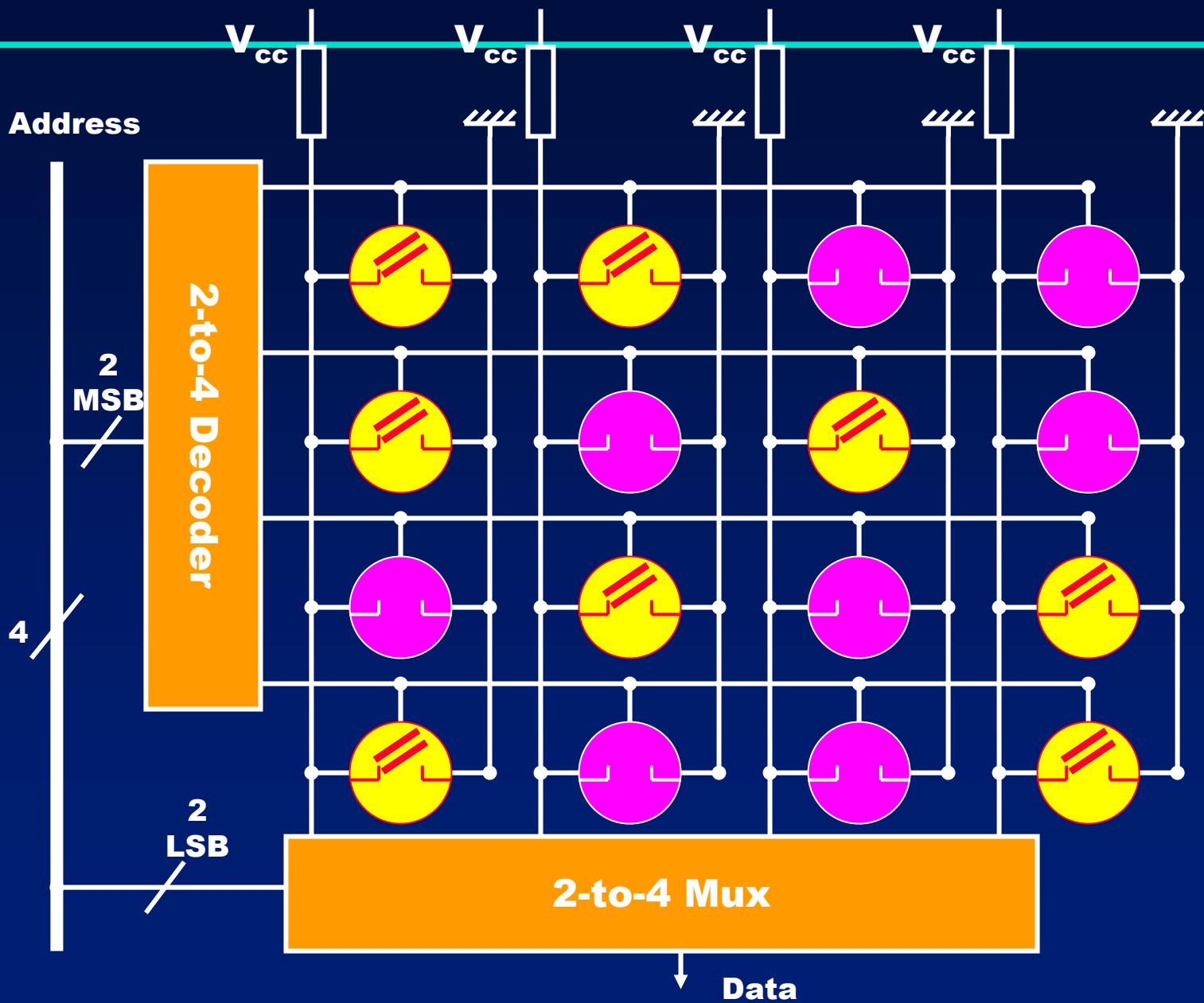


# EPROM

Many electrons trapped on floating gate

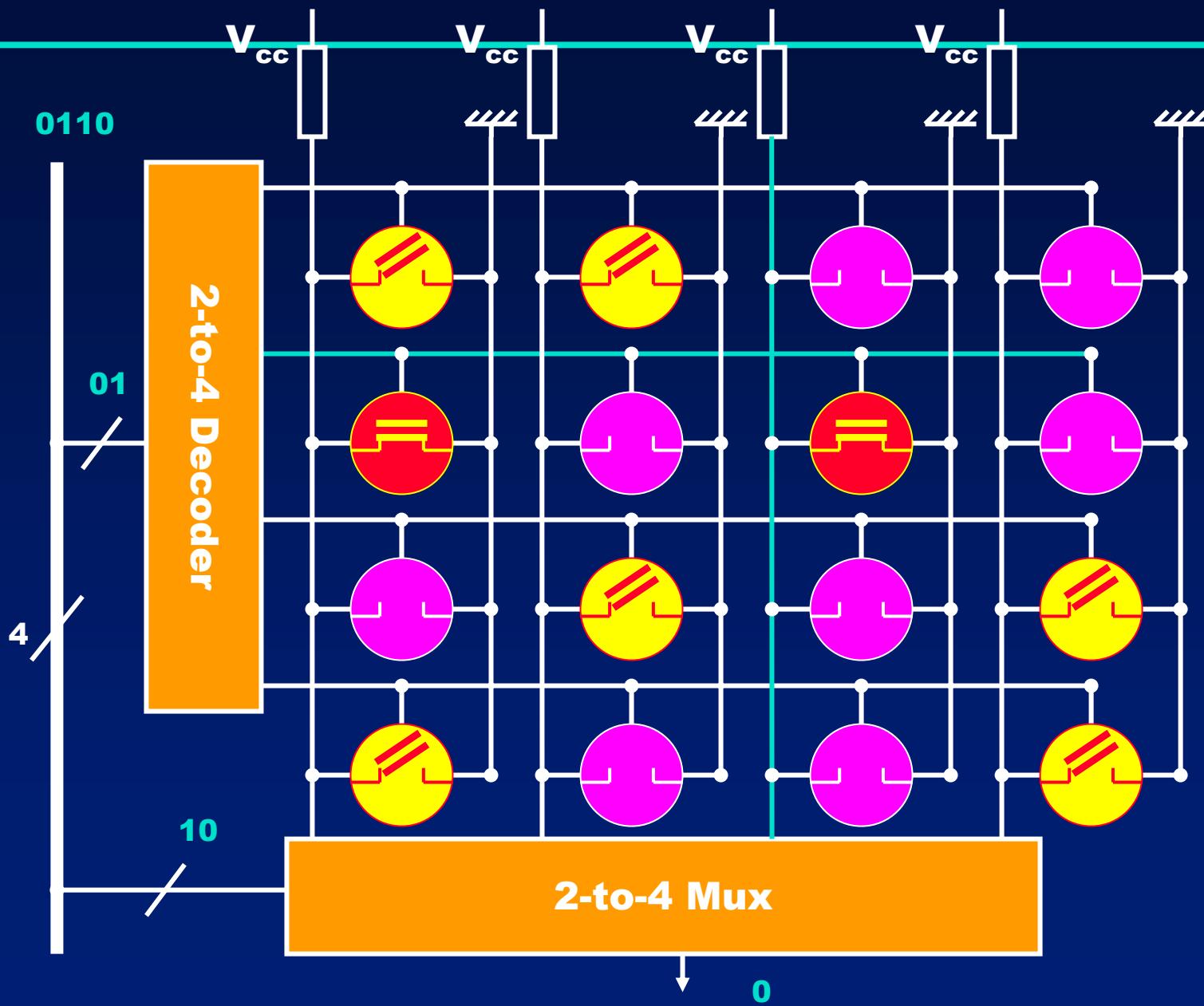


# EPROM: reading



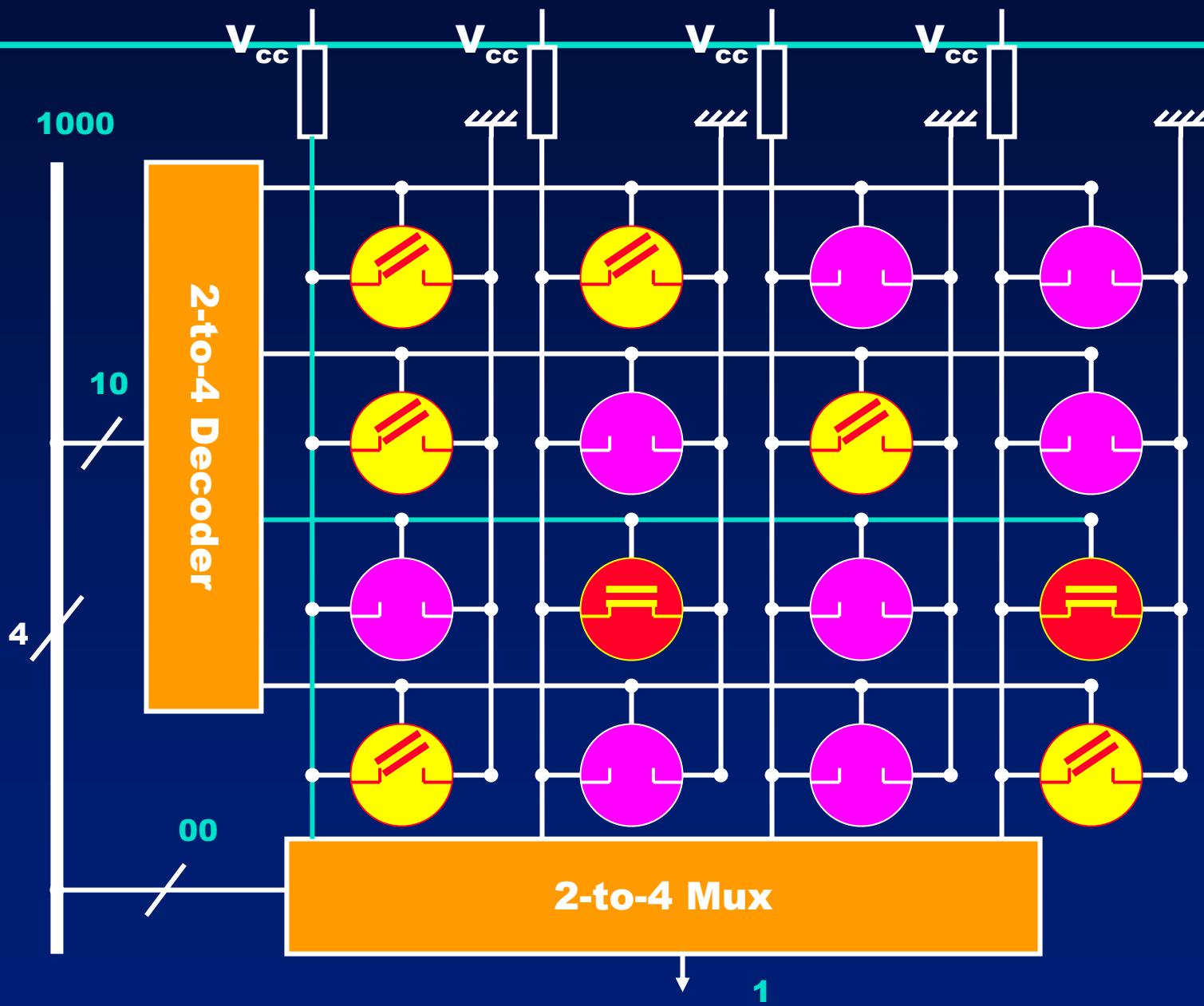
Read(0x6)

# EPROM: reading

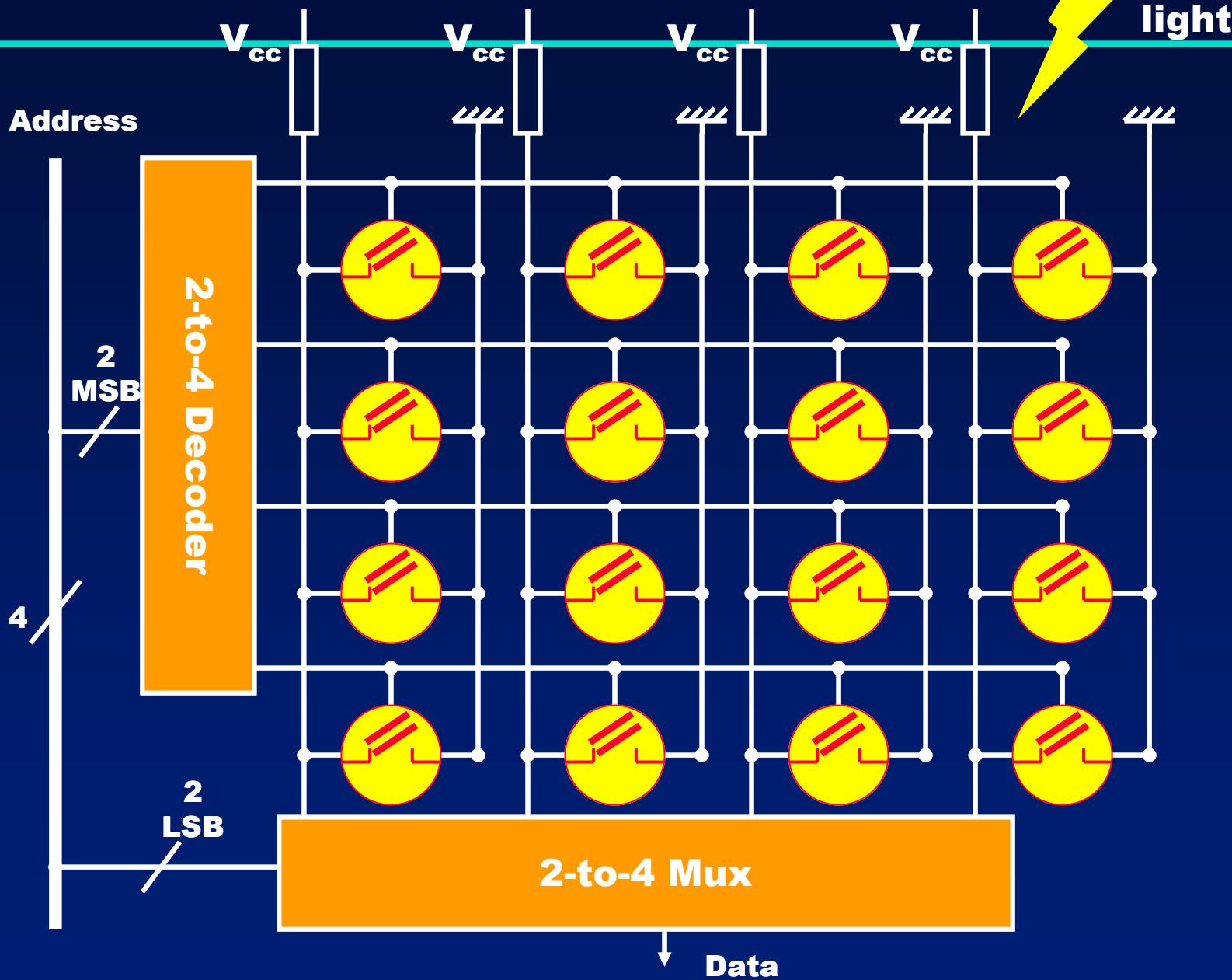


Read(0x8)

# EPROM: reading

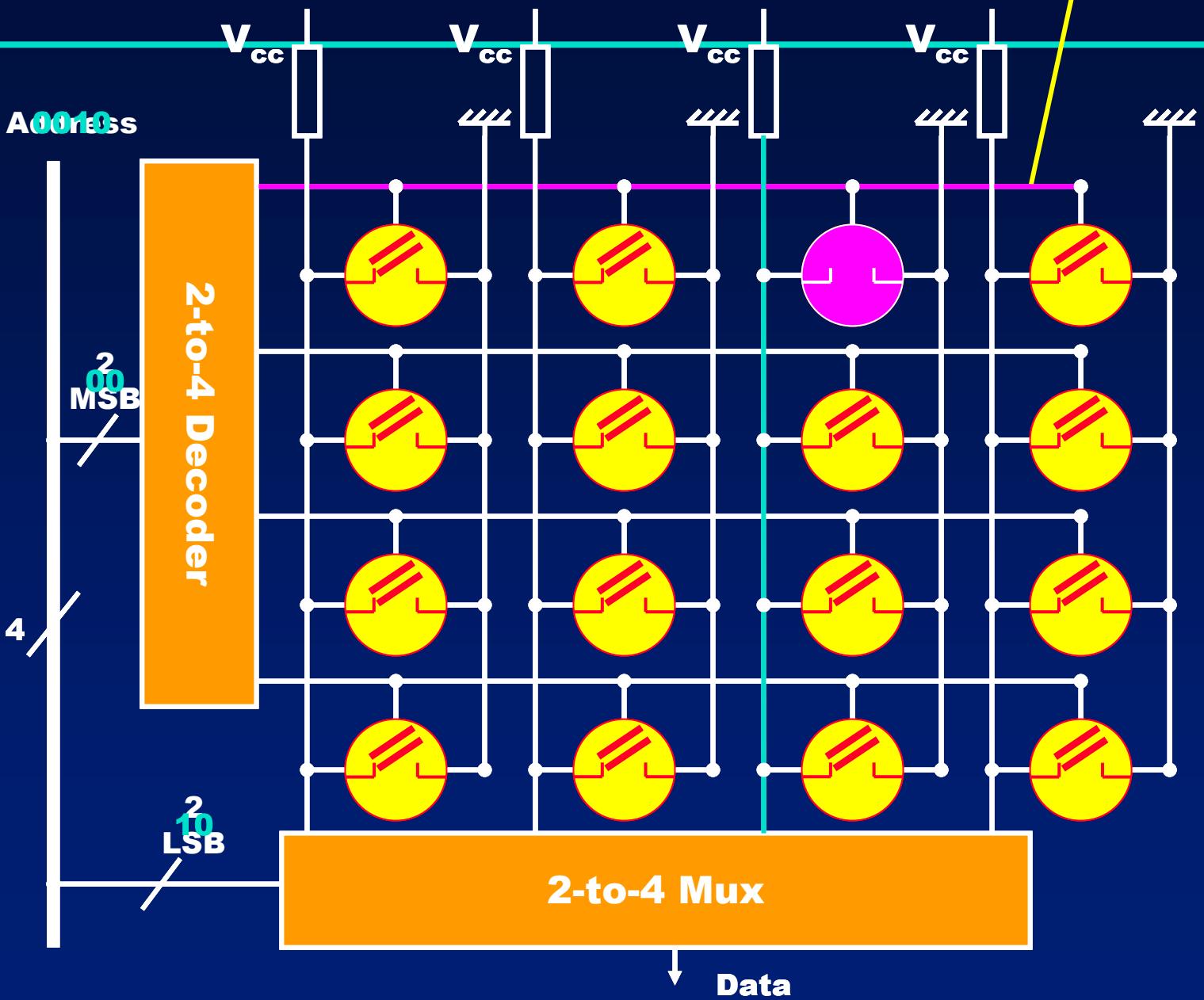


# EPROM: erasing



Write 1 at 0x2

# EPROM: writing



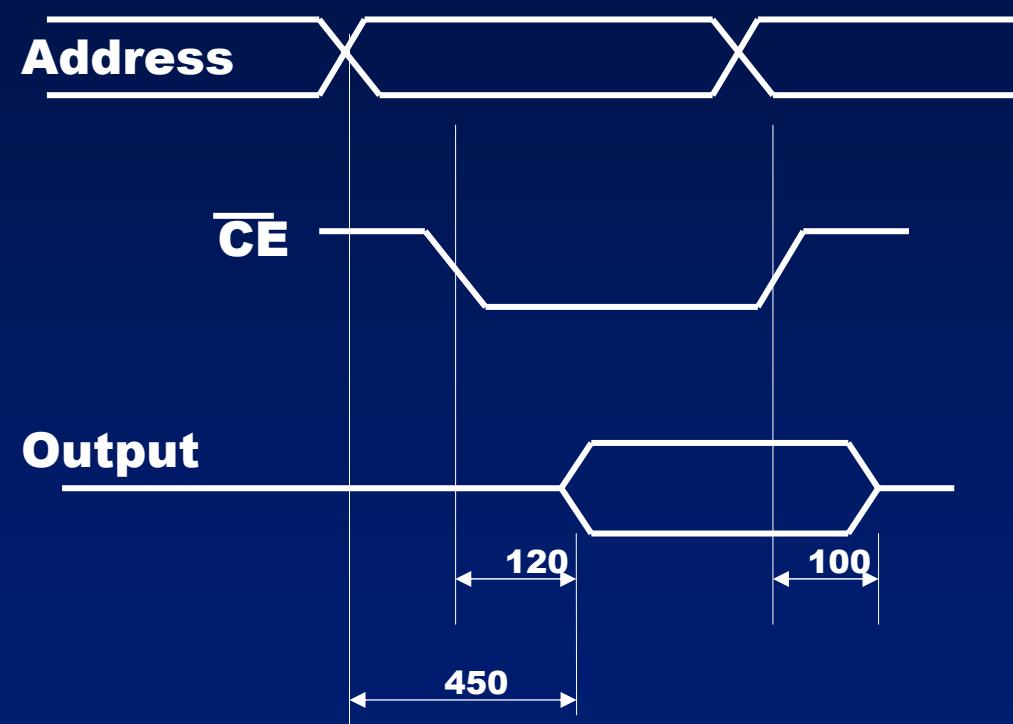
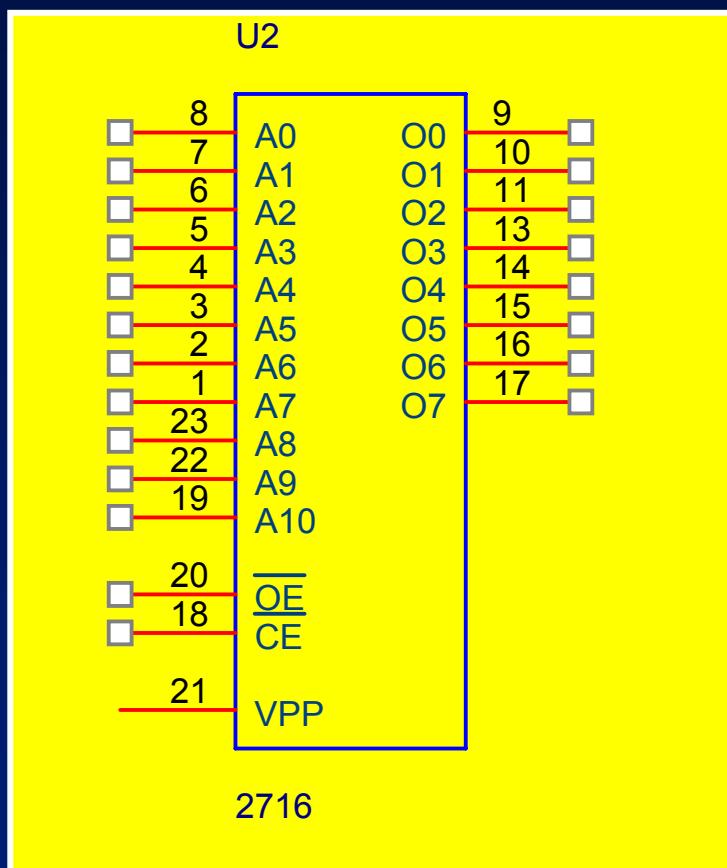


# EPROM

- **Ghi vào EPROM**
  - Dùng mạch nạp với điện áp 12 V
  - 1 ms một bit
- **Xoá EPROM**
  - 20 phút dưới tia tử ngoại
  - Số lần ghi 3 lần
- **Đọc EPROM**
  - 100 ns
- **EPROM họ 27xxx**
  - 2708 (1K\*8), 2716 (2K\*8), 2732 (4K\*8), 2764 (8K\*8)
  - 27128 (16K\*8), 27256 (32K\*8), 27512 (64K\*8)

# EPROM

- Ví dụ: 2716 EPROM



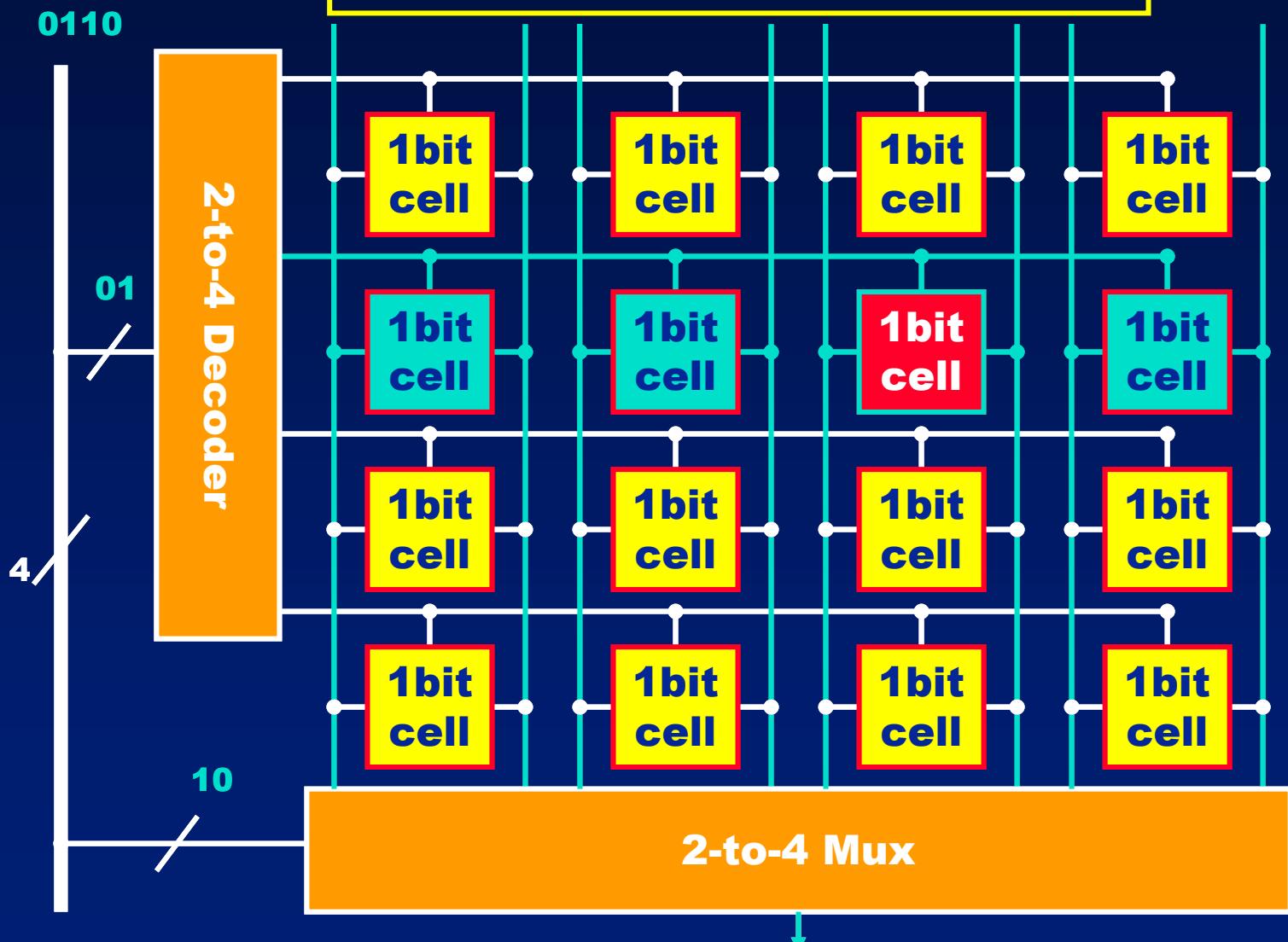
# So sánh các loại ROM

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

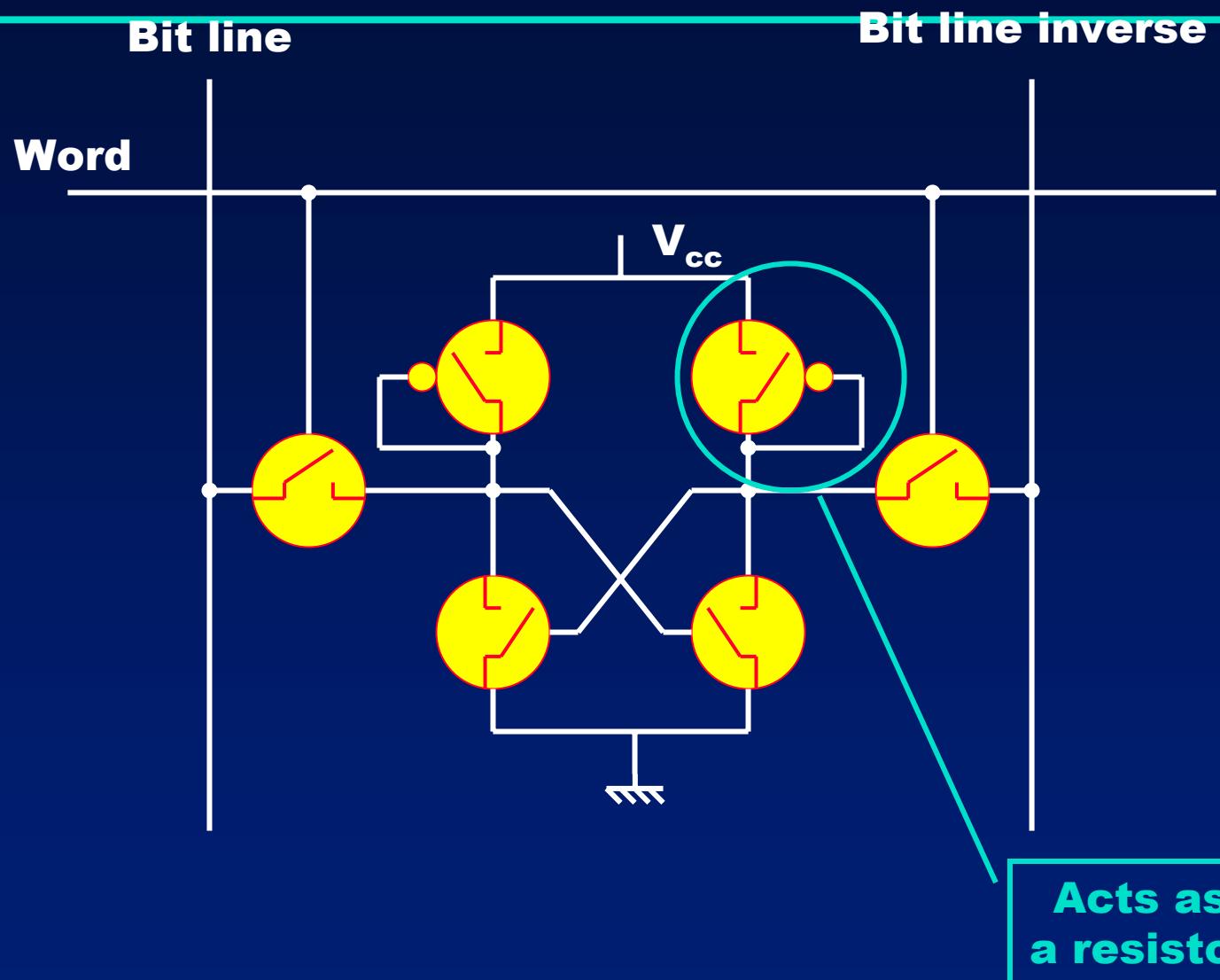
# SRAM

One row of cells is read out at once

MUX selects one out of these cells

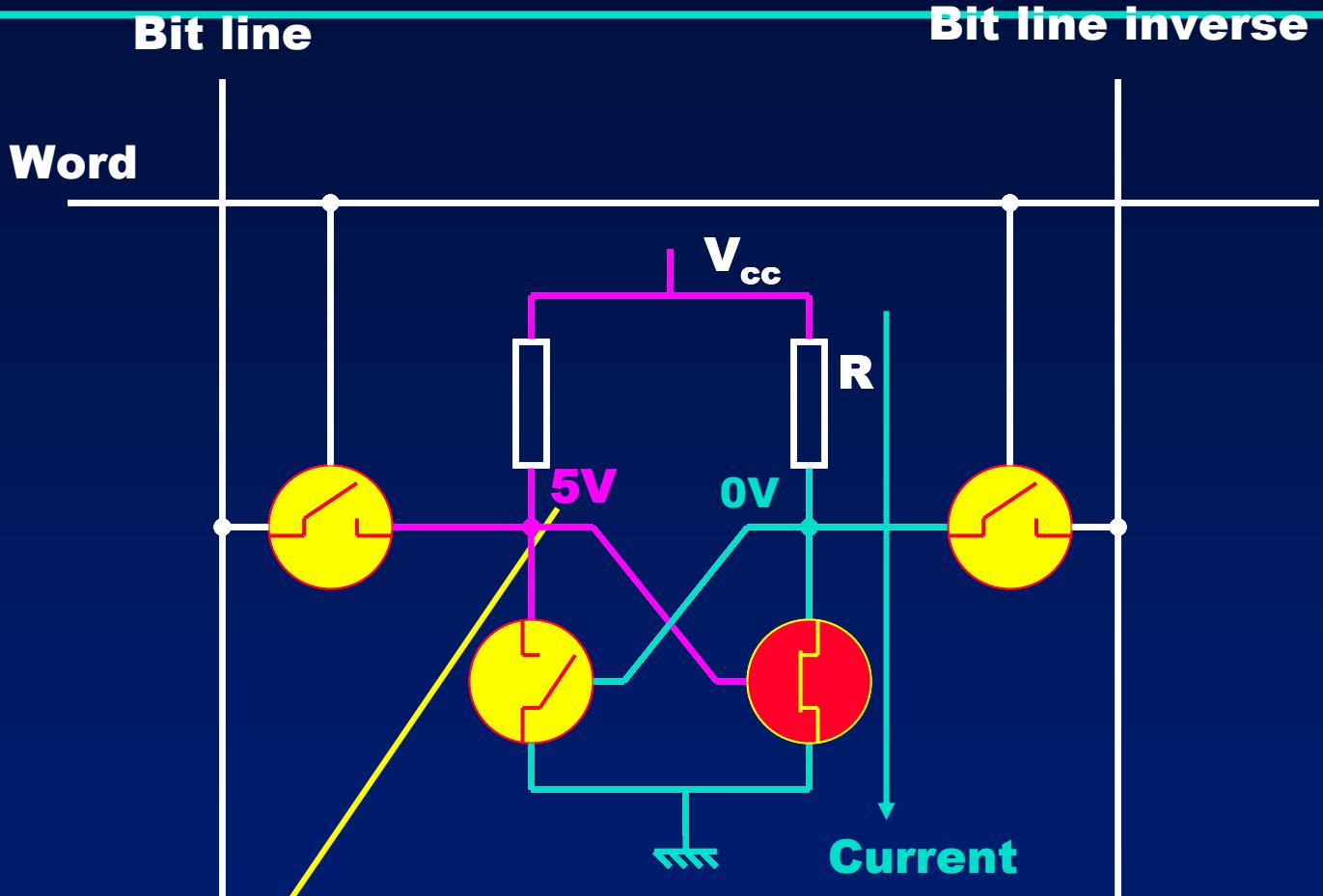


# SRAM bit cell



## Storage

# SRAM bit cell



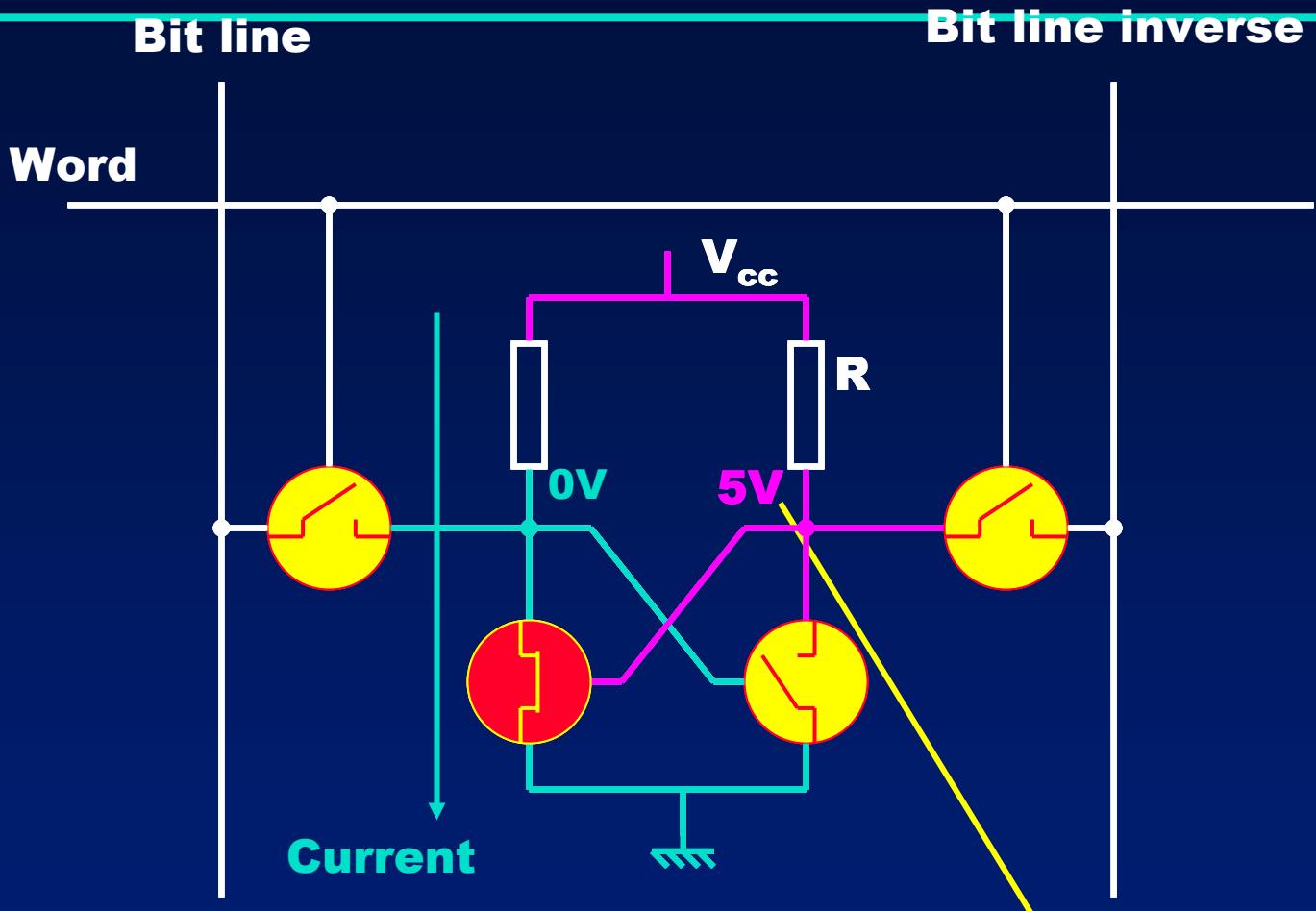
**Assumption**

Stable situation; stores a '1'

Dissipates continuously

## Storage

# SRAM bit cell

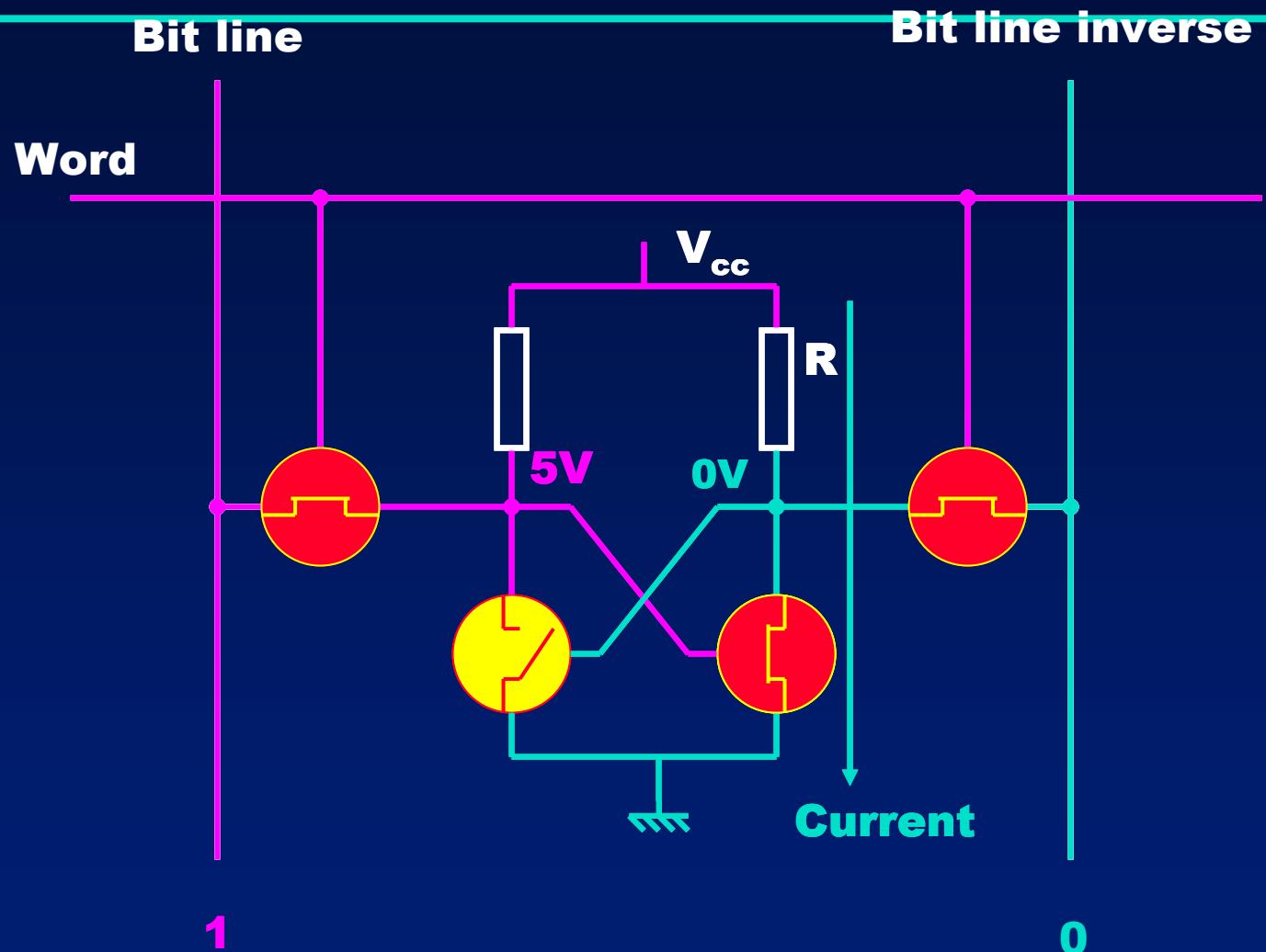


Stable situation; stores a '0'

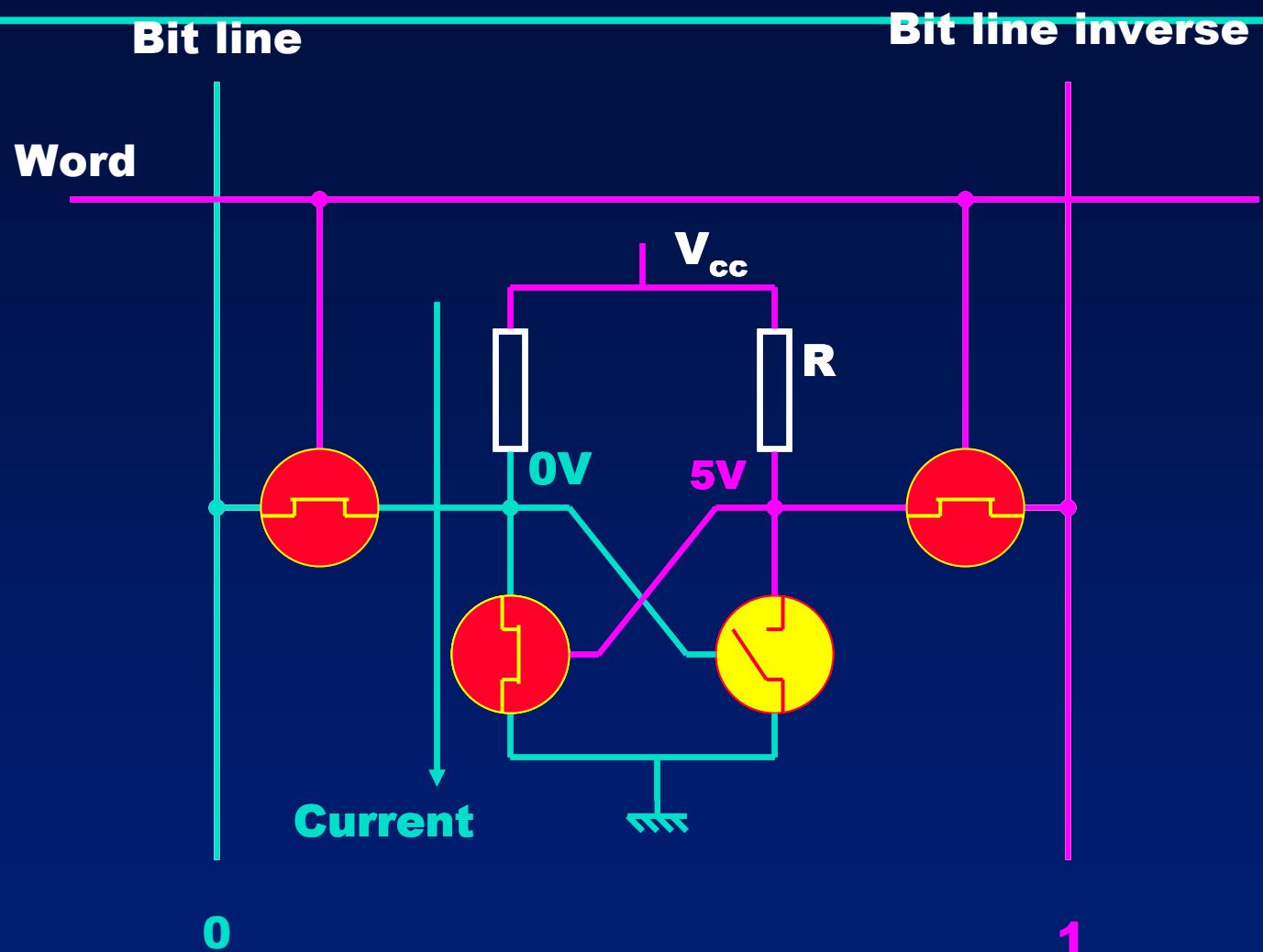
Dissipates continuously

Assumption

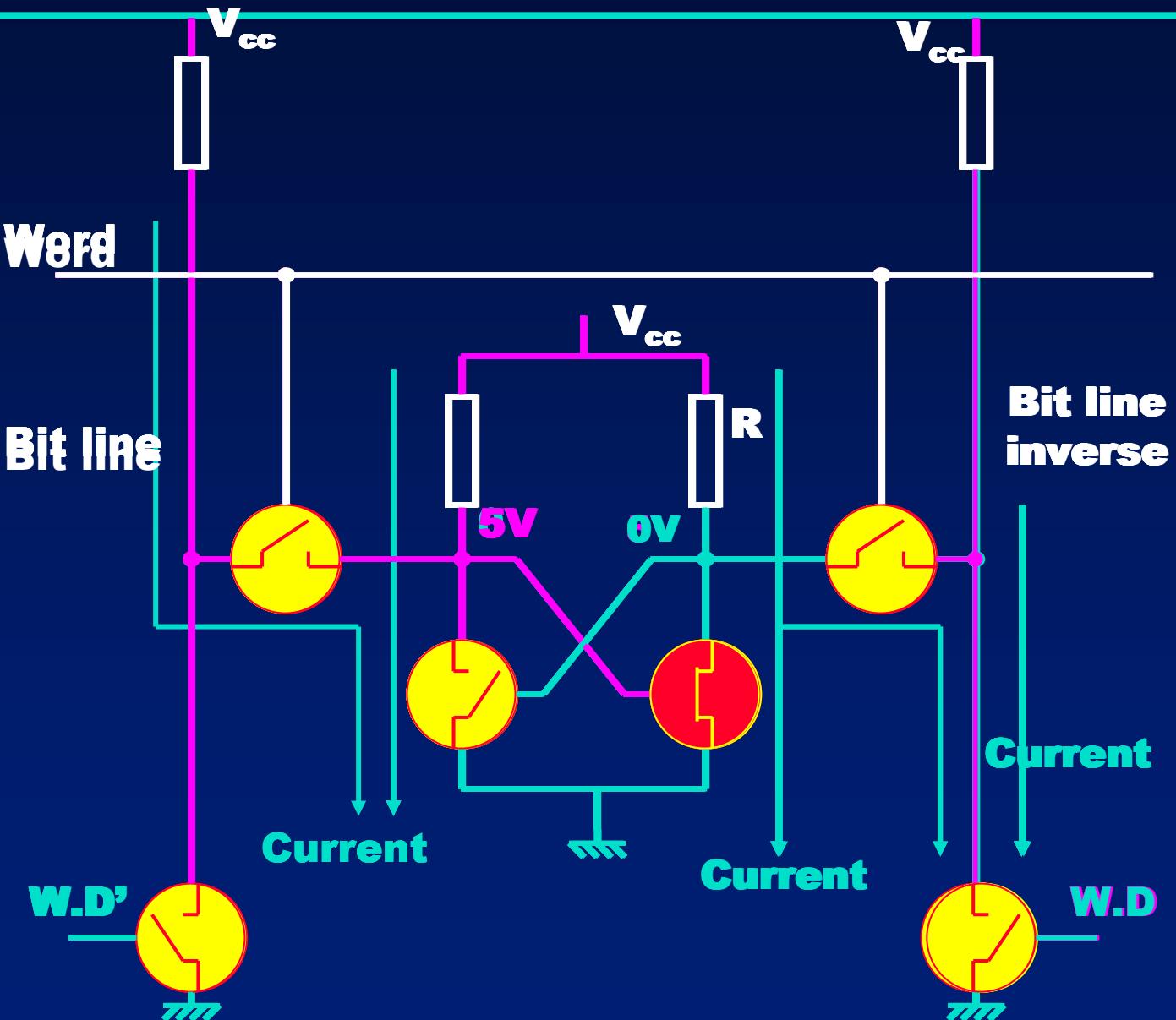
# Reading of a '1' SRAM bit cell



# Reading of a '0' SRAM bit cell

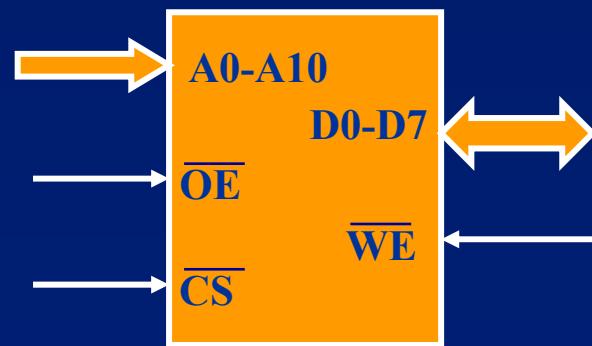


# Writing of a '1' SRAM bit cell

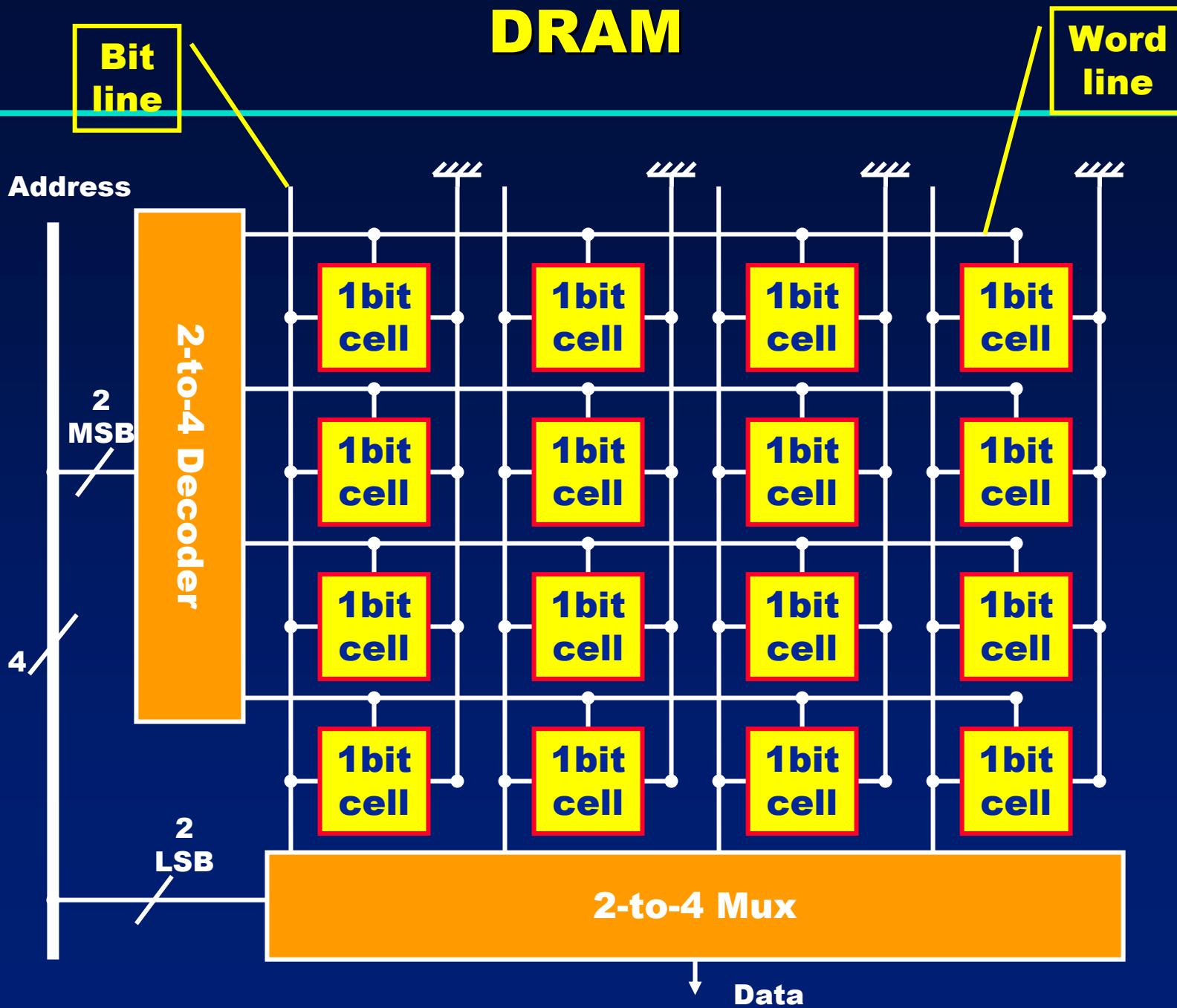


# SRAM

- **Đặc điểm:**
  - 6 transistors 1 bit: đắt!
  - **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**



# DRAM

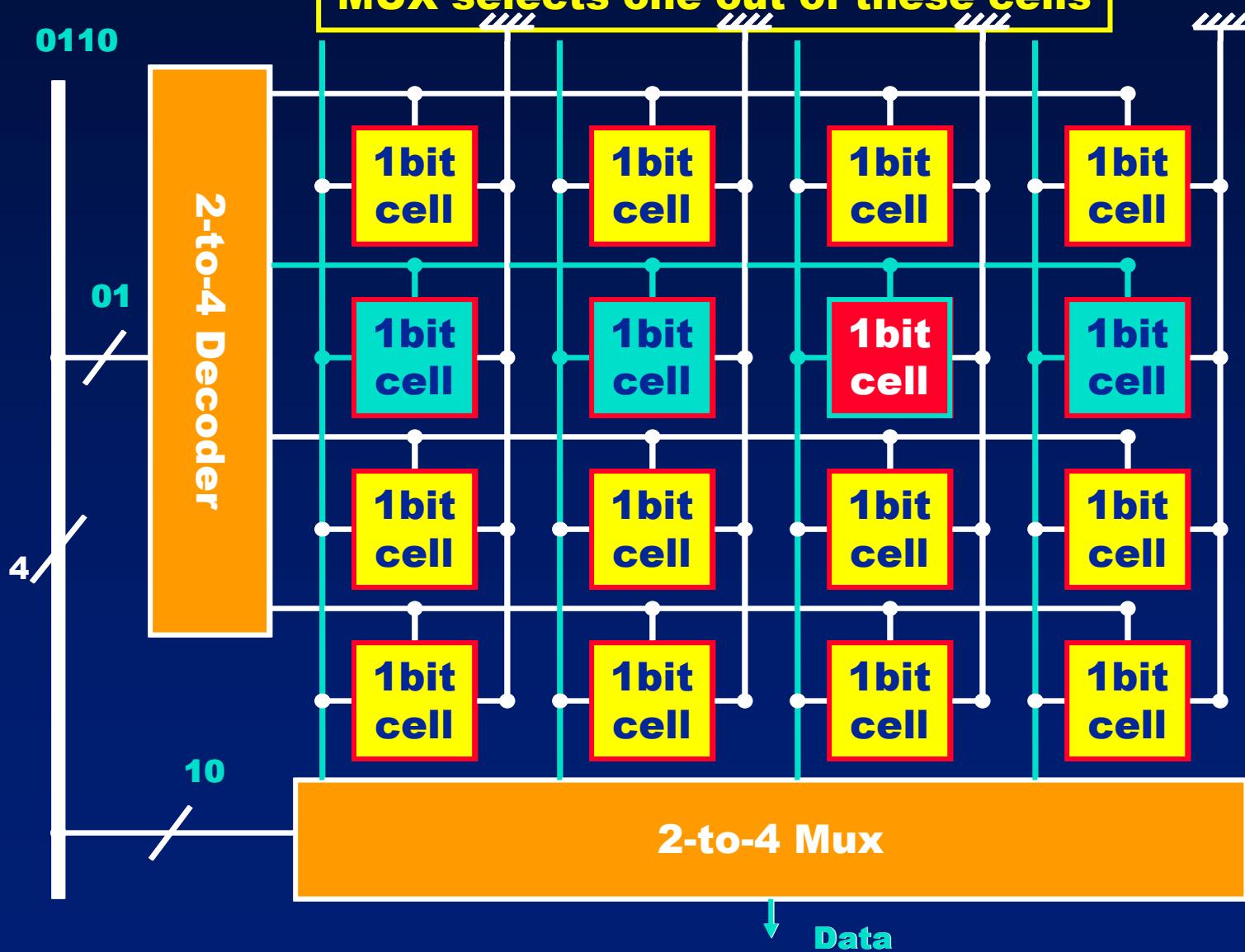




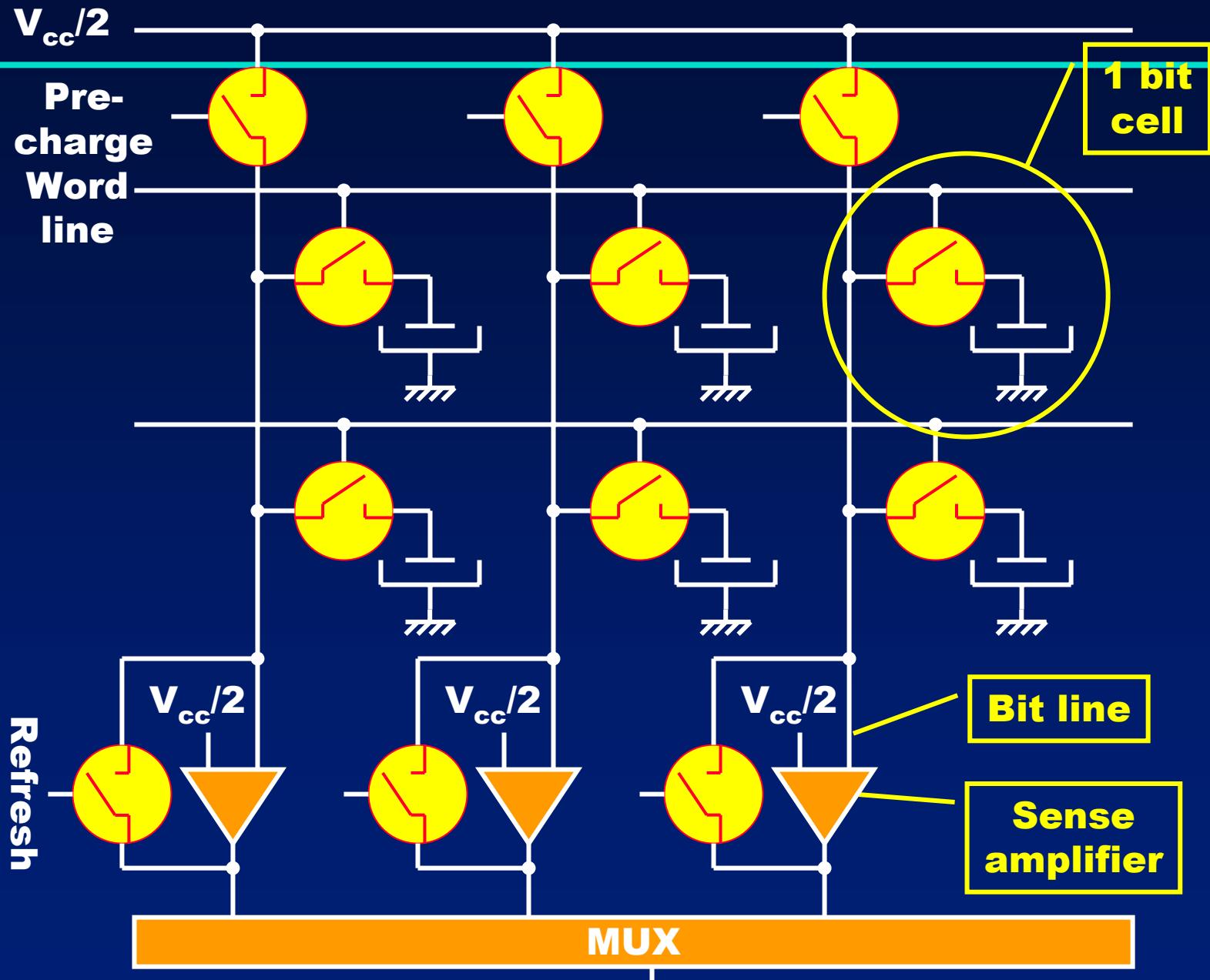
# DRAM

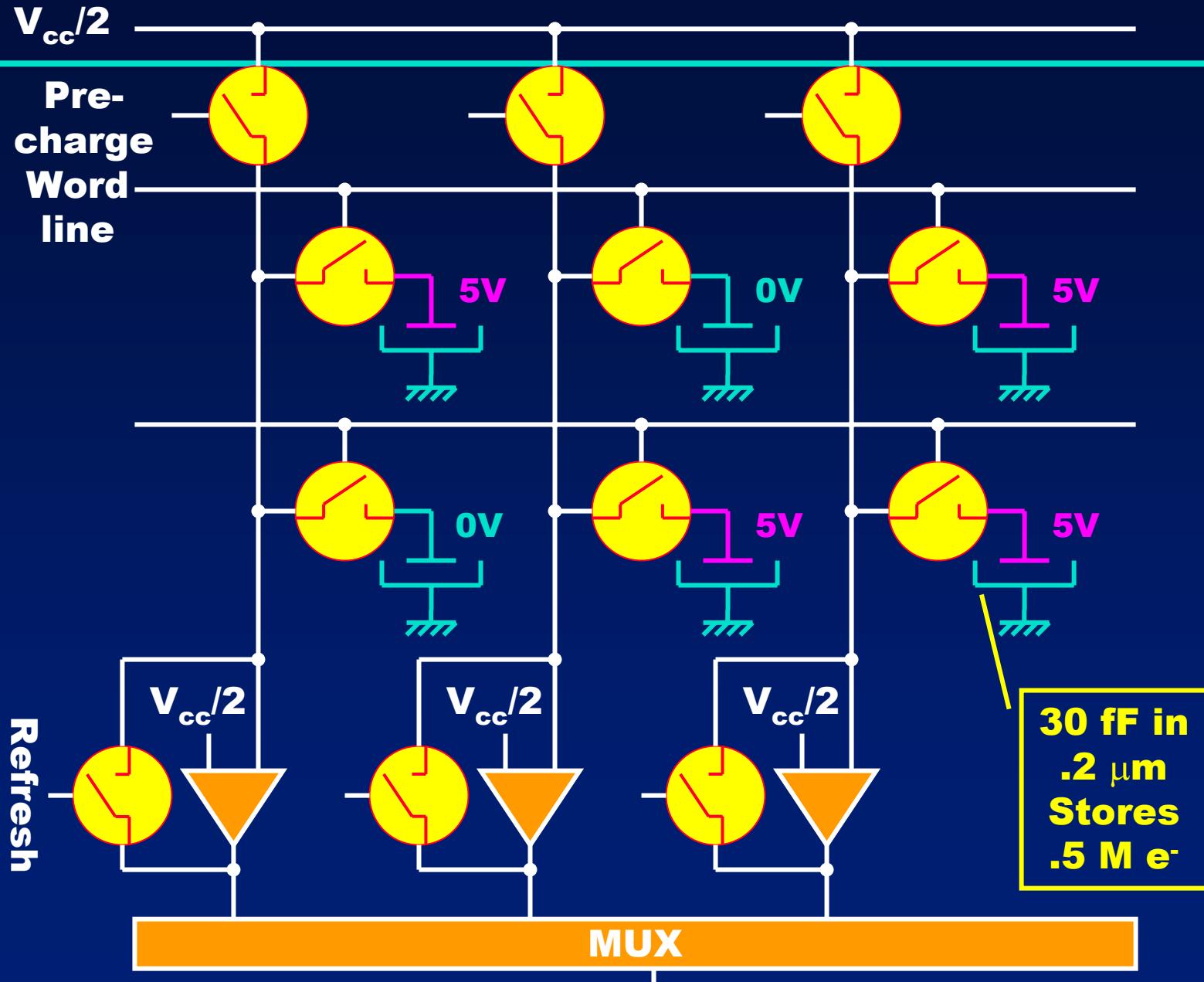
**One row of cells is read out at once**

**MUX selects one out of these cells**



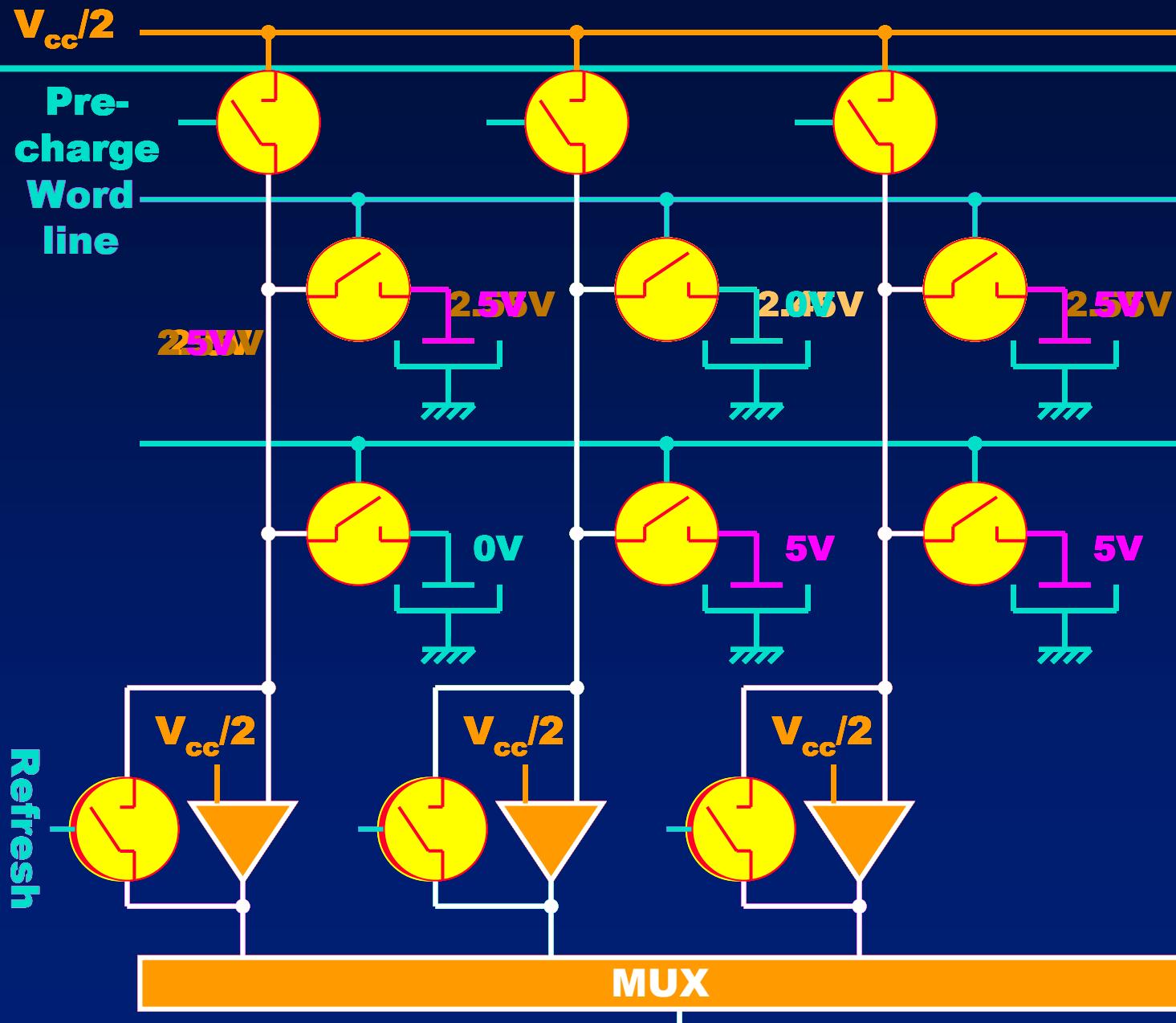
# DRAM bit cell



**Storage****DRAM bit cell**

Read

# DRAM bit cell

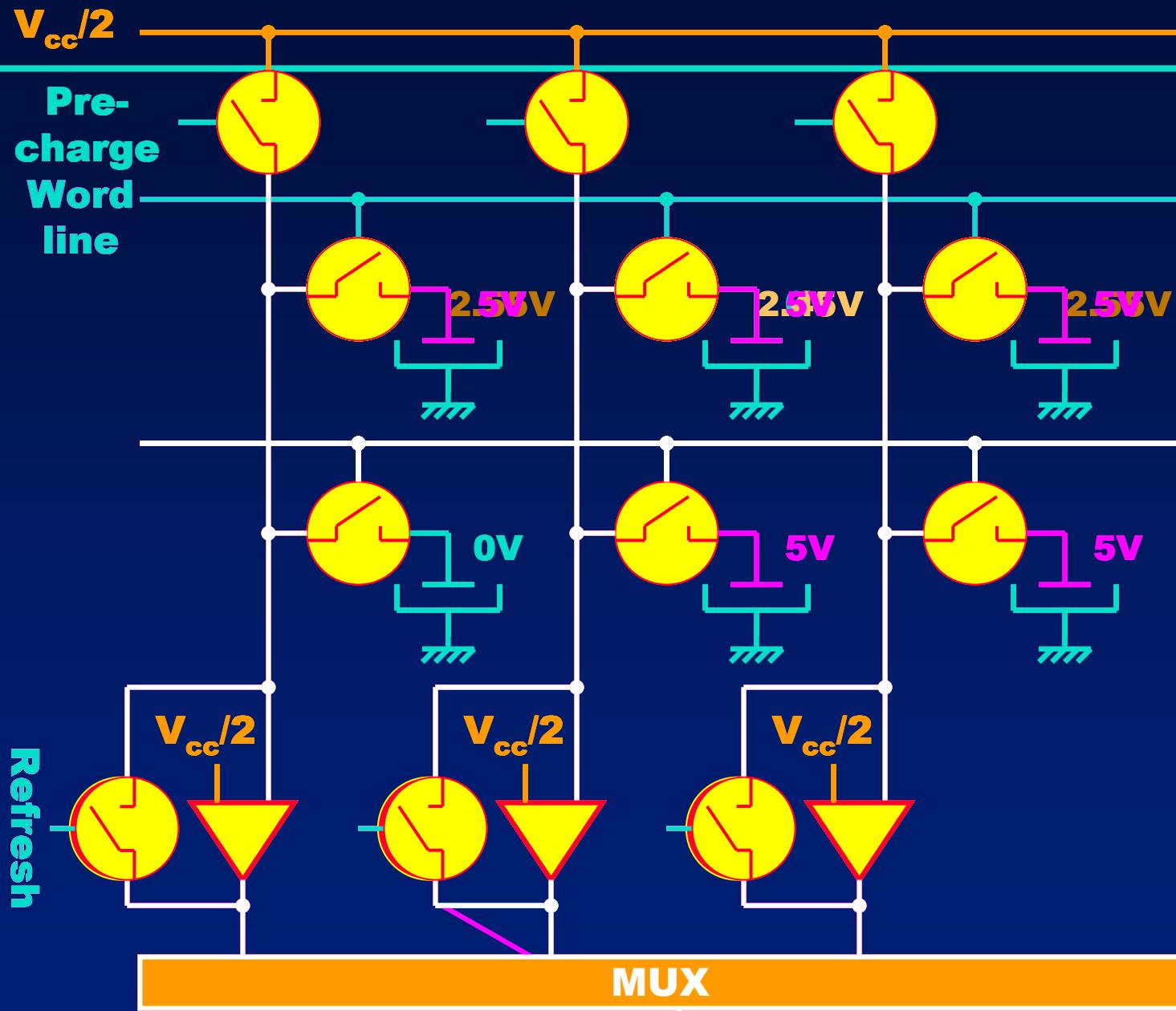


# DRAM bit cell

- 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.**

**Write**

# DRAM bit cell

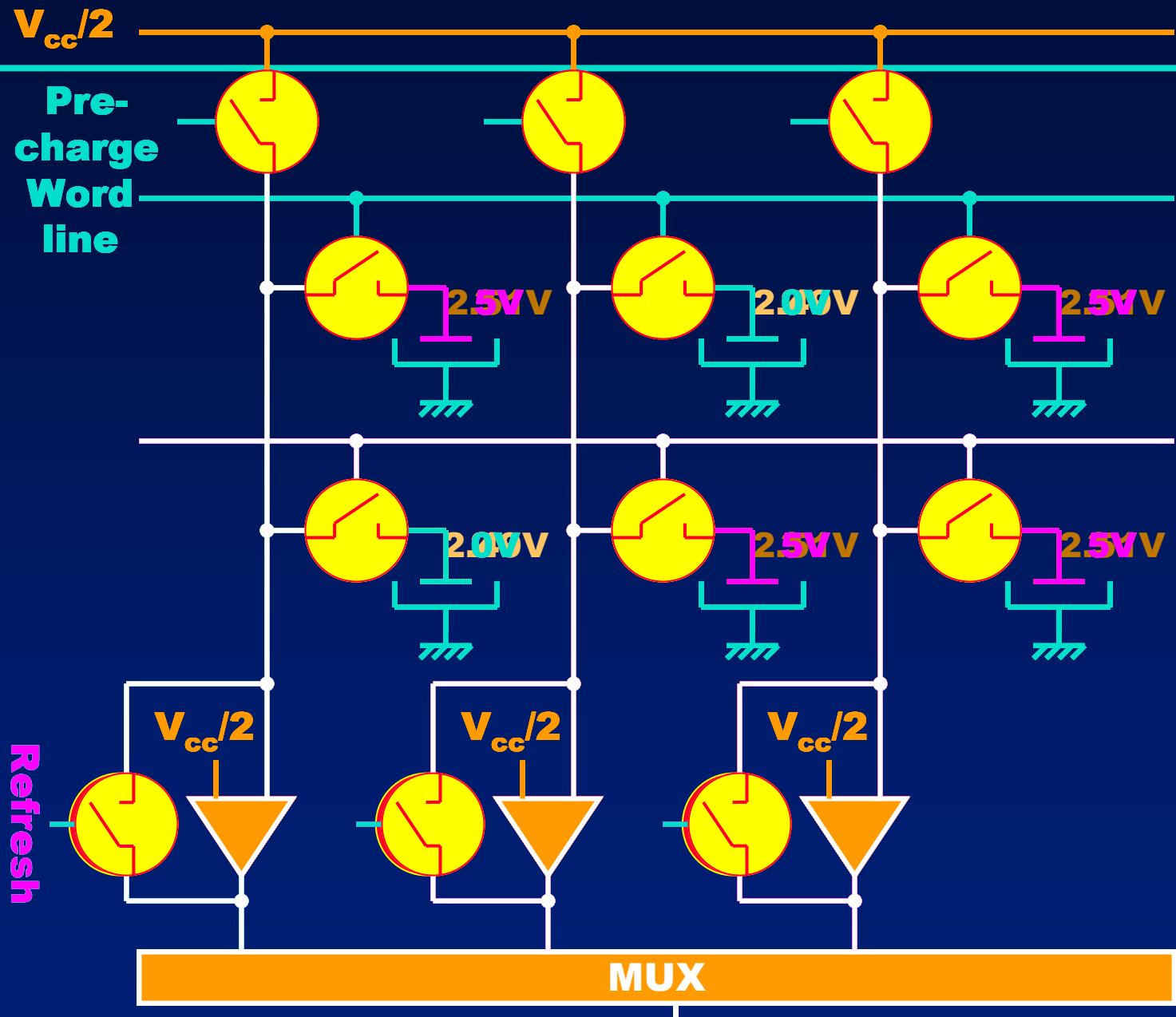


# DRAM bit cell

- 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.

## Refresh

## DRAM bit cell

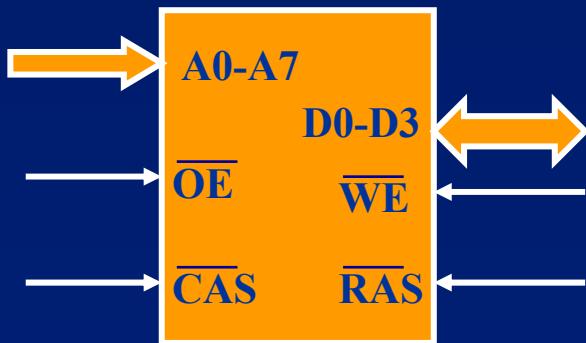


# DRAM bit cell

- 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.

# DRAM

- **Đặc điểm:**
  - 1 transistor 1 bit: rẻ, tuy nhiên việc điều khiển quá trình làm tươi làm tăng giá thành của DRAM
  - Chỉ 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

# DRAM

- **Đặc điểm:**
  - **1 transistor 1 bit: rẻ, tuy nhiên việc điều khiển quá trình làm tươi làm tăng giá thành của DRAM**
  - **Chỉ 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

# DRAM

- **Examples of DRAM:**

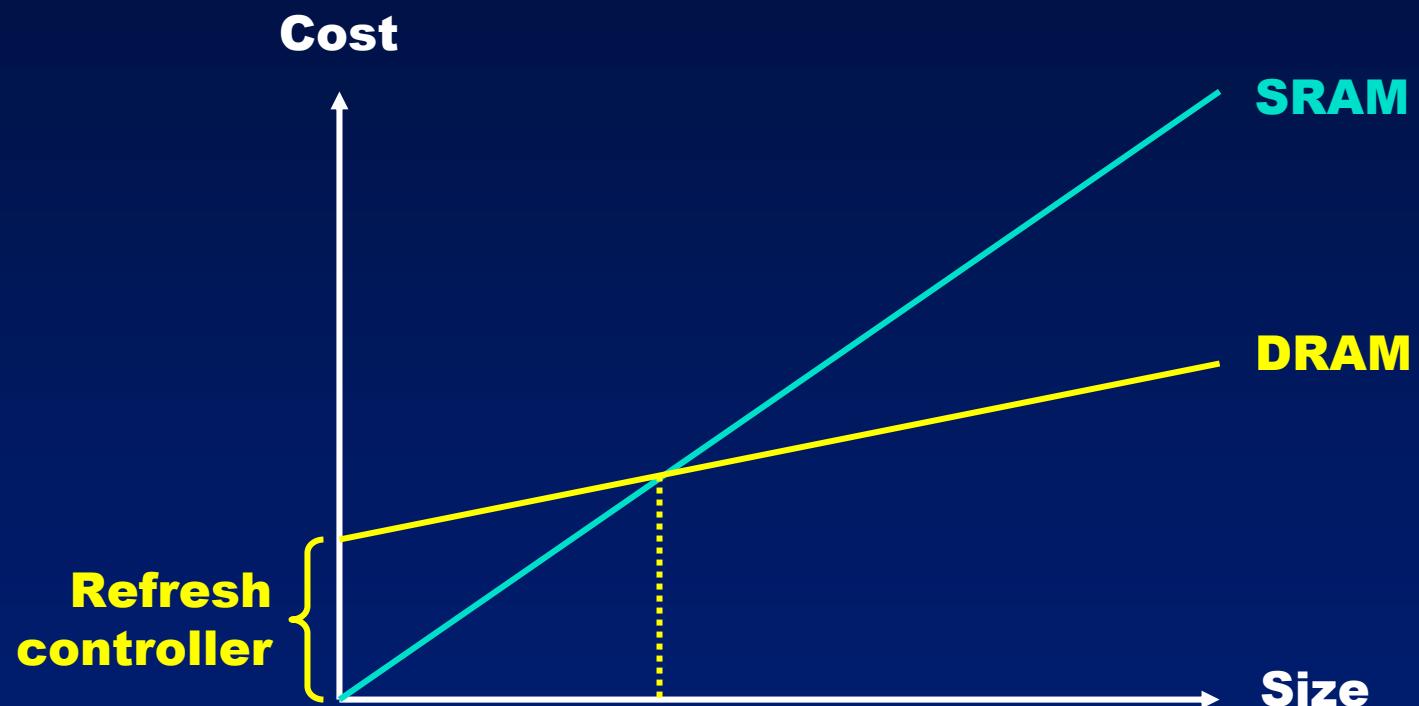
- **SIMM (Single Inline Memory Module): 72 pins**
- **DIMM (Dual Inline Memory Module): 168 pins**



**72-Pin SIMM**



# SRAM - DRAM





# Chương 4: Tổ chức vào ra dữ liệu

- Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
- Ghép nối 8088 với bộ nhớ
  - Các loại bộ nhớ bán dẫn
  - Giải mã địa chỉ cho bộ nhớ
    - Dùng công NAND
    - Dùng bộ giải mã 74LS138, 74LS139
    - Dùng PROM
    - Dùng PAL (Programmable Array Logic)
  - Ghép nối 8088 với bộ nhớ
- Ghép nối 8086 với bộ nhớ
- Ghép nối với thiết bị ngoại vi

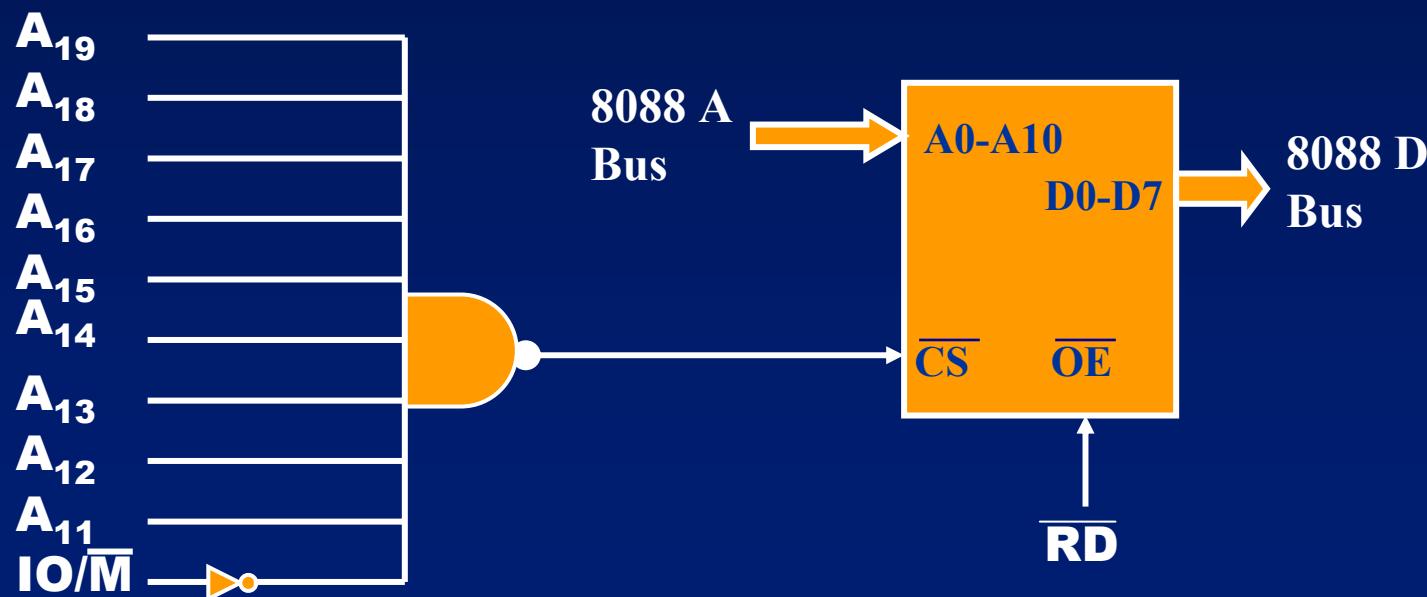
# Giải mã địa chỉ bộ nhớ dùng cỗng NAND

- **Ví dụ: Ghép EPROM 2716 (2K \* 8) với 8088**
- **Phân tích:**
  - 2716: 11 đường địa chỉ A10-A0
  - 8088: 20 đường địa chỉ A20-A0
  - **Chọn vùng nhớ 2K trong 1M?**
    - ⇒ EPROM: 0000H-003FFH: không được phép
    - ⇒ **chọn: FF800H-FFFFFH: chứa đoạn khởi động FFFF0H-FFFFFH**

A <sub>19</sub> A <sub>18</sub> A <sub>17</sub> A <sub>16</sub>	A <sub>15</sub> A <sub>14</sub> A <sub>13</sub> A <sub>12</sub>	A <sub>11</sub> A <sub>10</sub> A <sub>9</sub> A <sub>8</sub>	A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub>	A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>
• FF800: 1 1 1 1	1 1 1 1	1   0 0 0	0 0 0 0	0 0 0 0
• FFFFF: 1 1 1 1	1 1 1 1	1   1 1 1	1 1 1 1	1 1 1 1

# Giải mã địa chỉ bộ nhớ dùng cỗng NAND

- FF800: 1 1 1 1    1 1 1 1    1 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



# Giải mã địa chỉ bộ nhớ dùng bộ giải mã

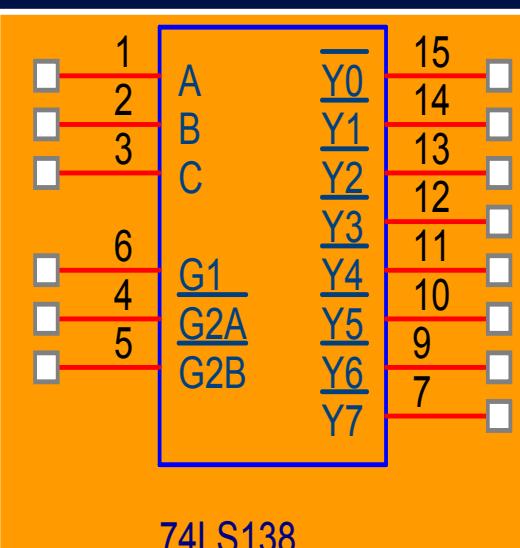
- Ví dụ:** Dùng EPROM 2764 (8K\*8) để ghép thành bộ nhớ 64 K cho 8088 bắt đầu từ địa chỉ F0000H
- Phân tích:**
  - Địa chỉ bắt đầu F0000H => địa chỉ kết thúc: FFFFFH
  - Cần ghép 8 EPROM 2764 vì  $64=8 \times 8K$

	A <sub>19</sub> A <sub>18</sub> A <sub>17</sub> A <sub>16</sub>	A <sub>15</sub> A <sub>14</sub> A <sub>13</sub> A <sub>12</sub>	A <sub>11</sub> A <sub>10</sub> A <sub>9</sub> A <sub>8</sub>	A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub>	A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub>	
• F0000:	1 1 1 1	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	
• F2000:	1 1 1 1	0 0 1	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	
• F4000:	1 1 1 1	0 1 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	
...						
...						
• FE000:	1 1 1 1	1 1 1	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	



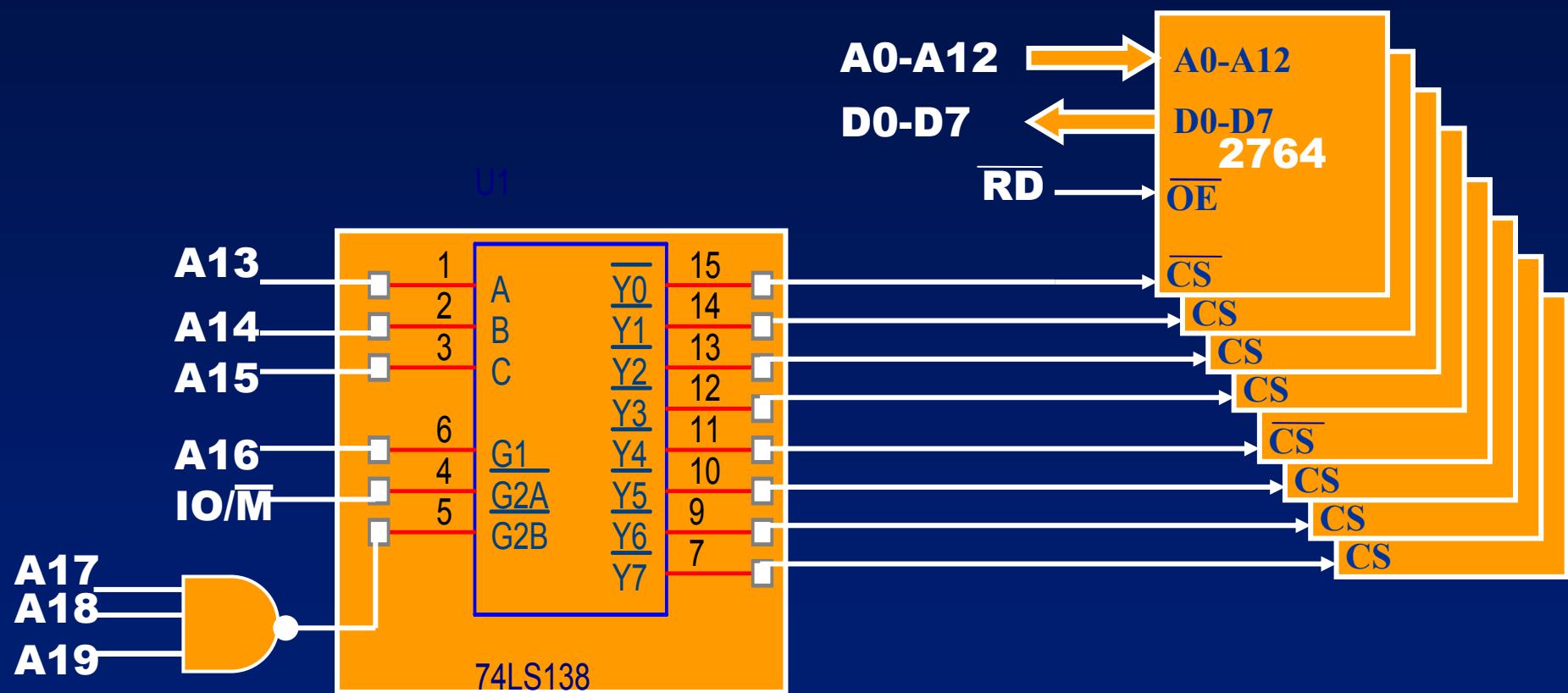
# **Giải mã địa chỉ bộ nhớ dùng bộ giải mã**

- Dùng bộ giải mã 3-8 74LS138



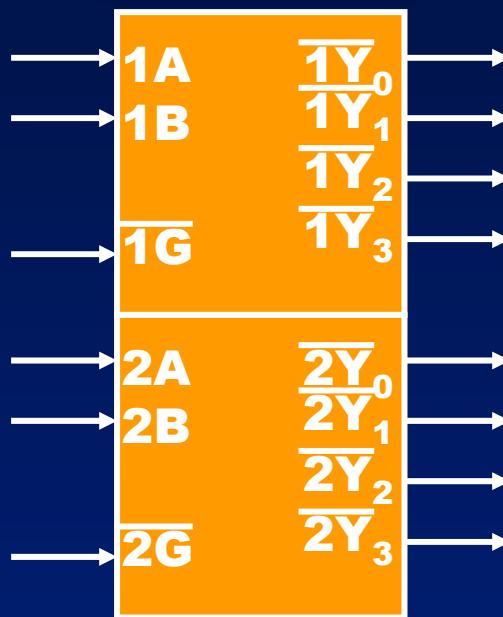
# Giải mã địa chỉ bộ nhớ dùng bộ giải mã

- Dùng bộ giải mã 3-8 74LS138



# Giải mã địa chỉ bộ nhớ dùng bộ giải mã

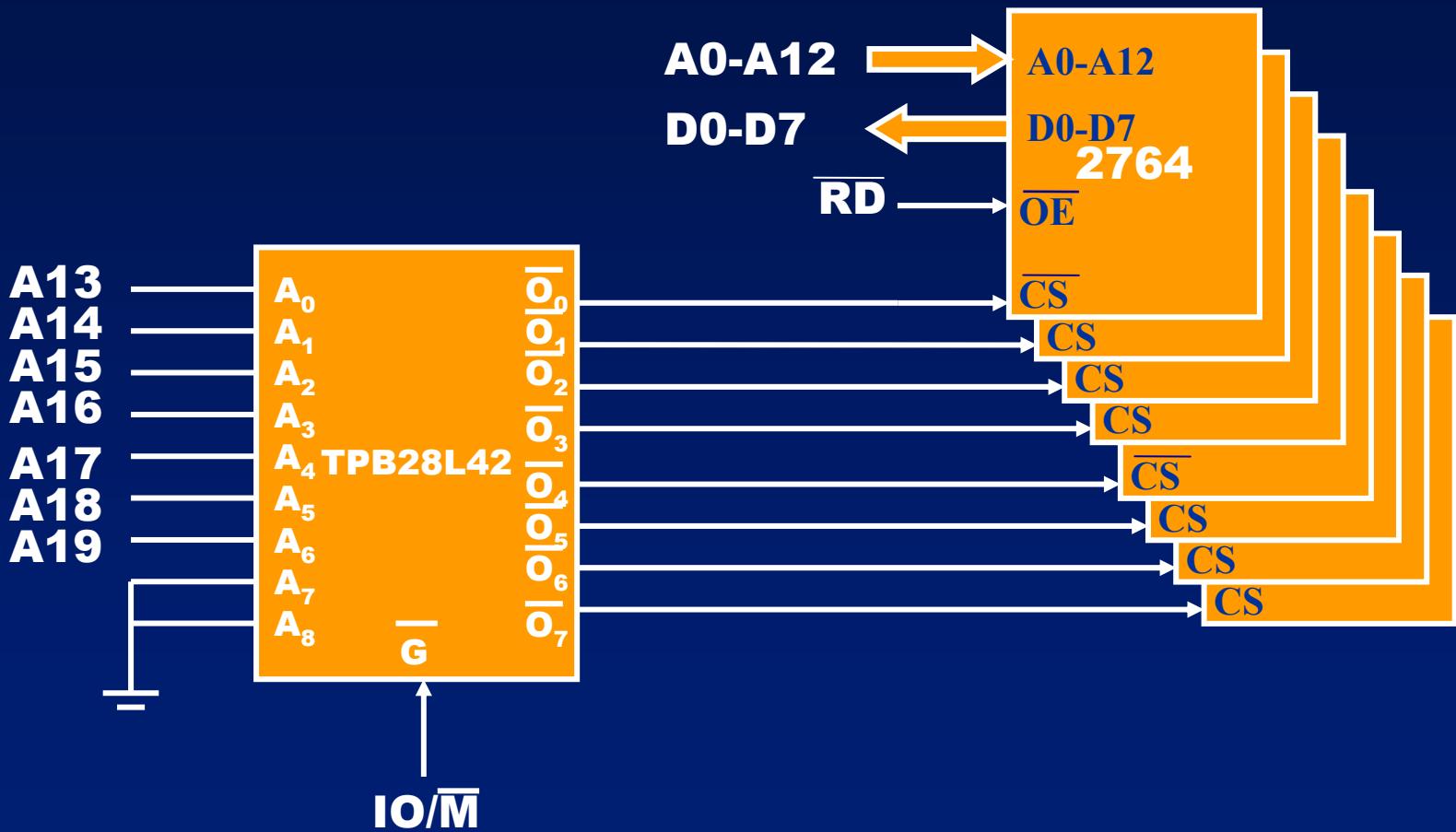
- Dùng bộ giải mã kép 2-4 74LS139



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

# Giải mã địa chỉ bộ nhớ dùng PROM

- Dùng PROM TPB28L42 (512\*8)



# 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<sub>19</sub> - A<sub>13</sub> onto 8 outputs.*

			<i>pins 1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>
			A19	A18	A17	A16	A15	A14	A13	NC	NC	GND

I1	1	20	V <sub>CC</sub>									
I2	2	19	O8									
I3	3	18	O7									
I4	4	17	O6									
I5	5	16	O5									
I6	6	15	O4									
I7	7	14	O3									
I8	8	13	O2									
I9	9	12	O1									
GND	10	11	I10									

*Equations:*

$$\begin{aligned}
 O1 &= A_{19} * A_{18} * A_{17} * A_{16} * /A_{15} * /A_{14} * /A_{13} \\
 O2 &= A_{19} * A_{18} * A_{17} * A_{16} * /A_{15} * /A_{14} * A_{13} \\
 O3 &= A_{19} * A_{18} * A_{17} * A_{16} * /A_{15} * A_{14} * /A_{13} \\
 O4 &= A_{19} * A_{18} * A_{17} * A_{16} * /A_{15} * A_{14} * A_{13} \\
 O5 &= A_{19} * A_{18} * A_{17} * A_{16} * A_{15} * /A_{14} * /A_{13} \\
 O6 &= A_{19} * A_{18} * A_{17} * A_{16} * A_{15} * /A_{14} * A_{13} \\
 O7 &= A_{19} * A_{18} * A_{17} * A_{16} * A_{15} * A_{14} * /A_{13} \\
 O8 &= A_{19} * A_{18} * A_{17} * A_{16} * A_{15} * A_{14} * A_{13}
 \end{aligned}$$



# Chương 4: Tổ chức vào ra dữ liệu

- Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
- Ghép nối 8088 với bộ nhớ
  - Các loại bộ nhớ bán dẫn
  - Giải mã địa chỉ cho bộ nhớ
  - Ghép nối 8088 với bộ nhớ
    - Ghép nối 8088 với ROM
    - Ghép nối 8088 với SRAM
    - Ghép nối 8088 với DRAM
- Ghép nối 8086 với bộ nhớ
- Ghép nối với thiết bị ngoại vi



# 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

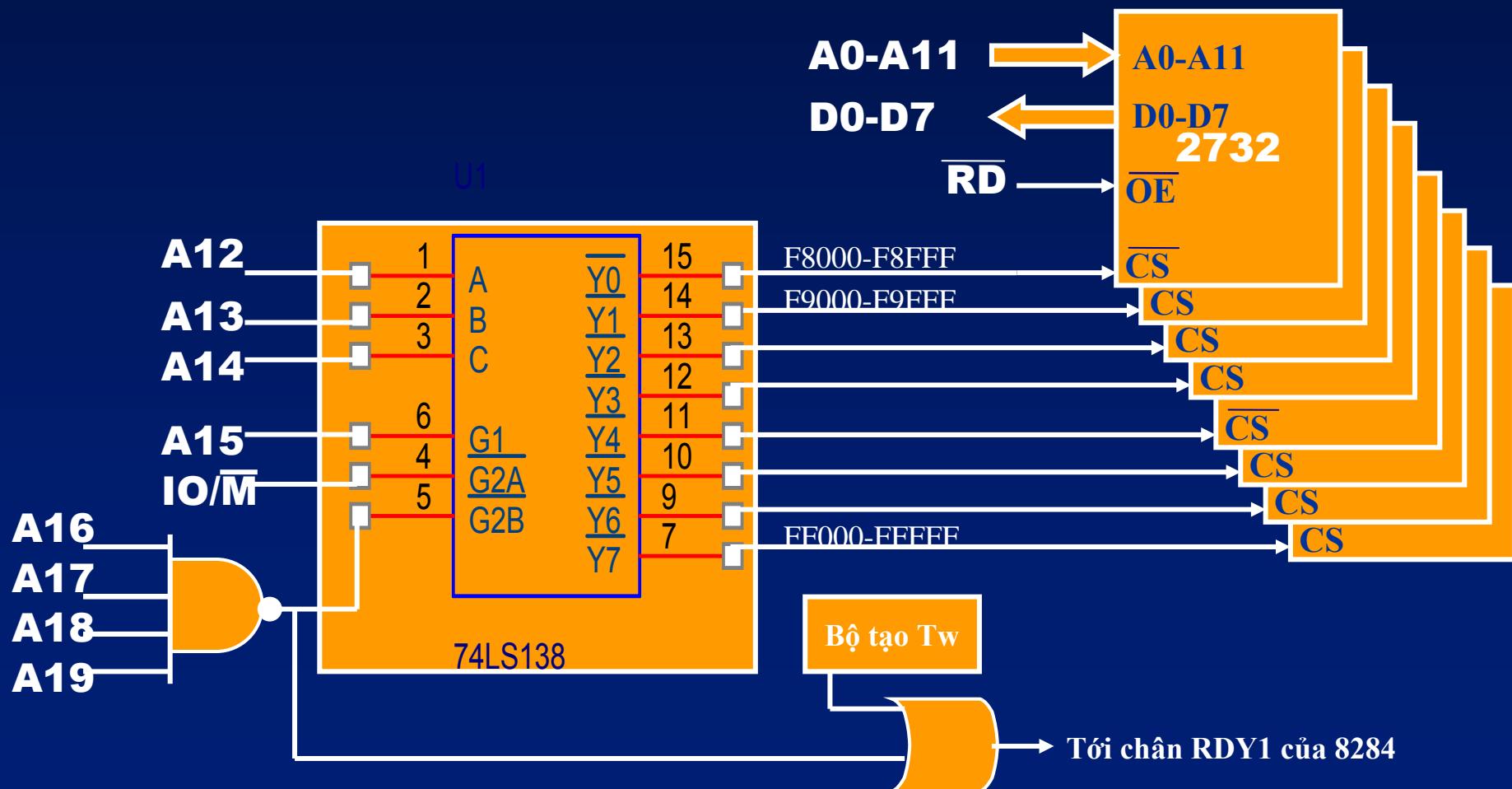
- **Ghép có chèn thêm thời gian đợi của CPU**

- ⇒ 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 420 ns**

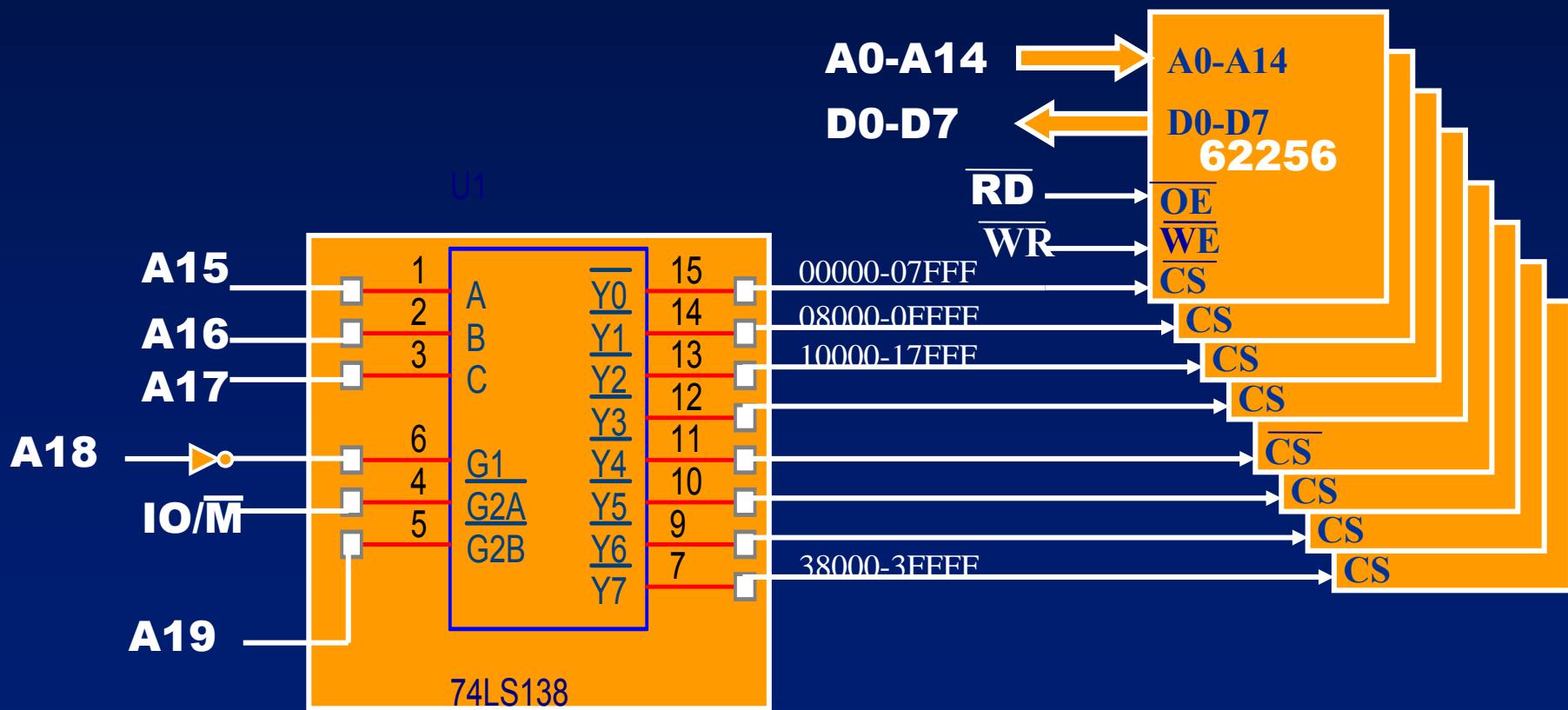
# Ghép nối 8088 với ROM

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



# Ghép nối 8088 với SRAM

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





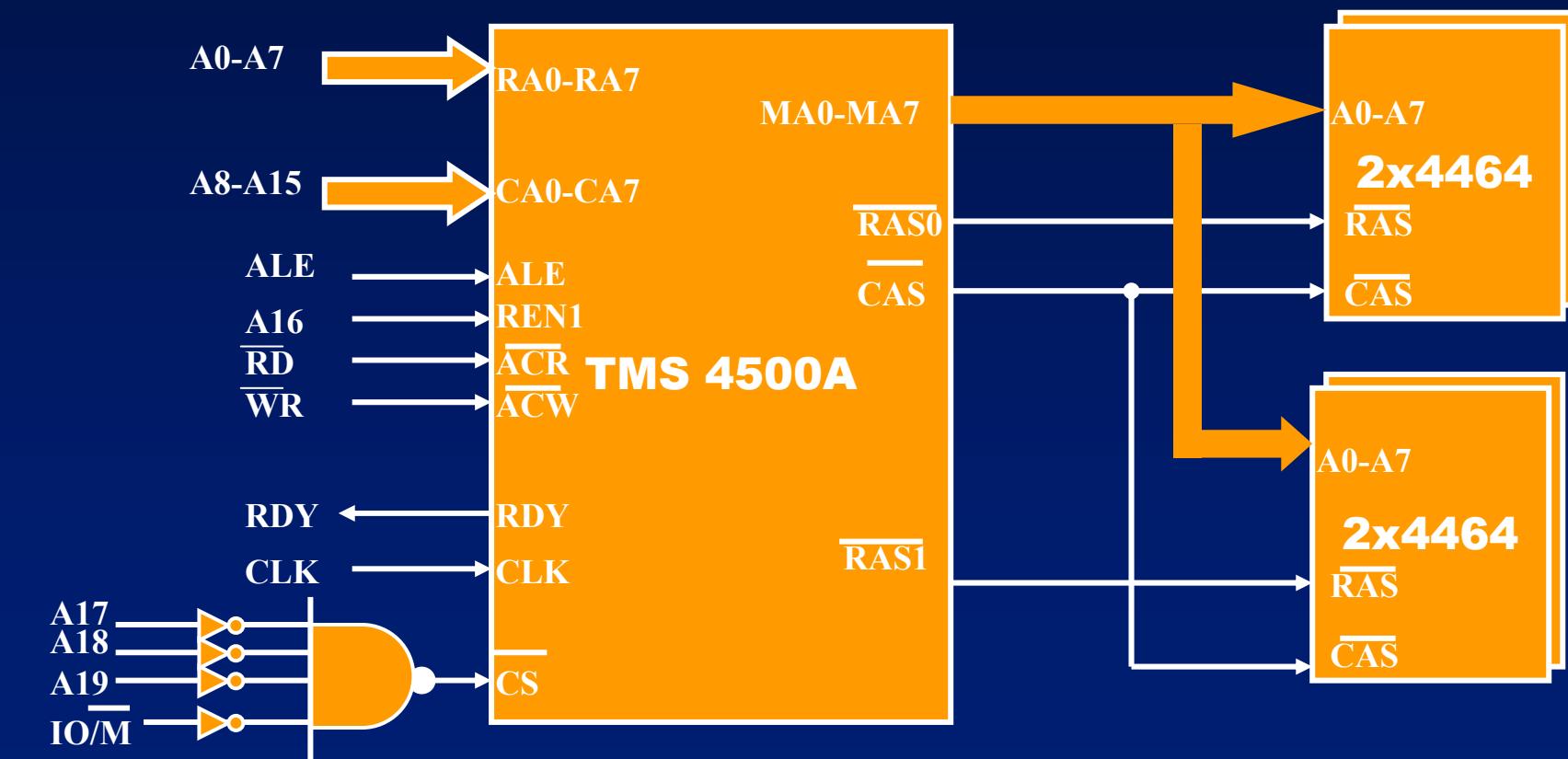
# Ghép nối 8088 với DRAM

- **Cần có DRAM controller:**

- **Dồn 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**

# Ghép nối 8088 với DRAM

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

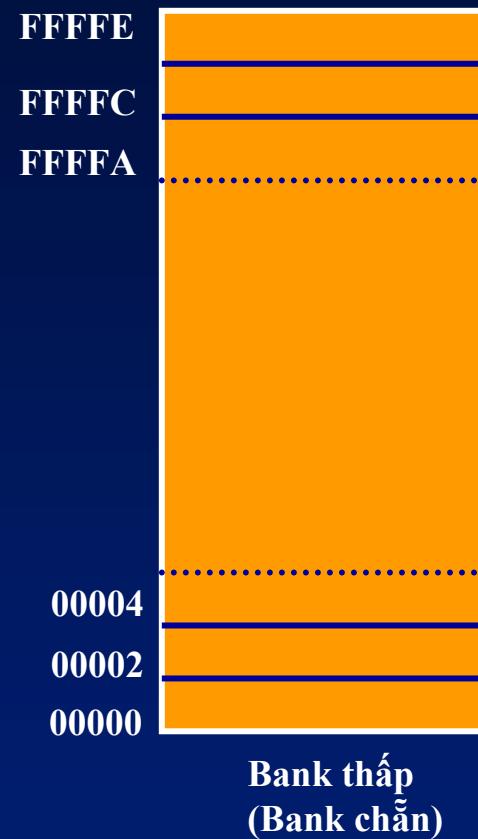
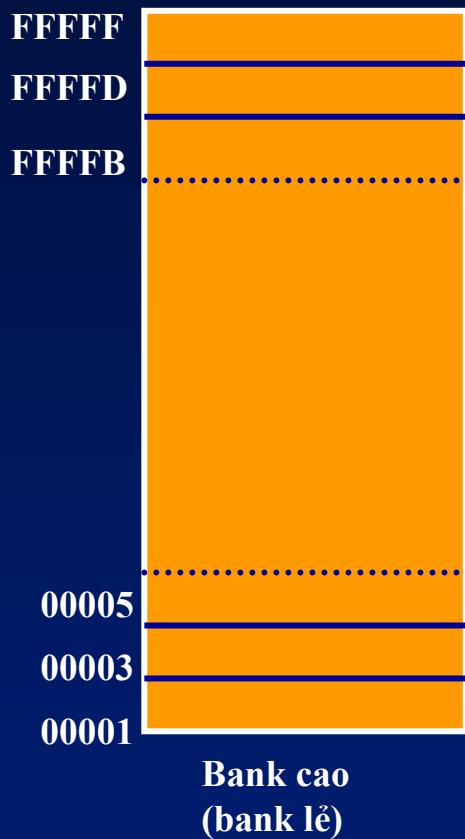




## Chương 4: Tổ chức vào ra dữ liệu

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

# Ghép nối 8086 với bộ nhớ



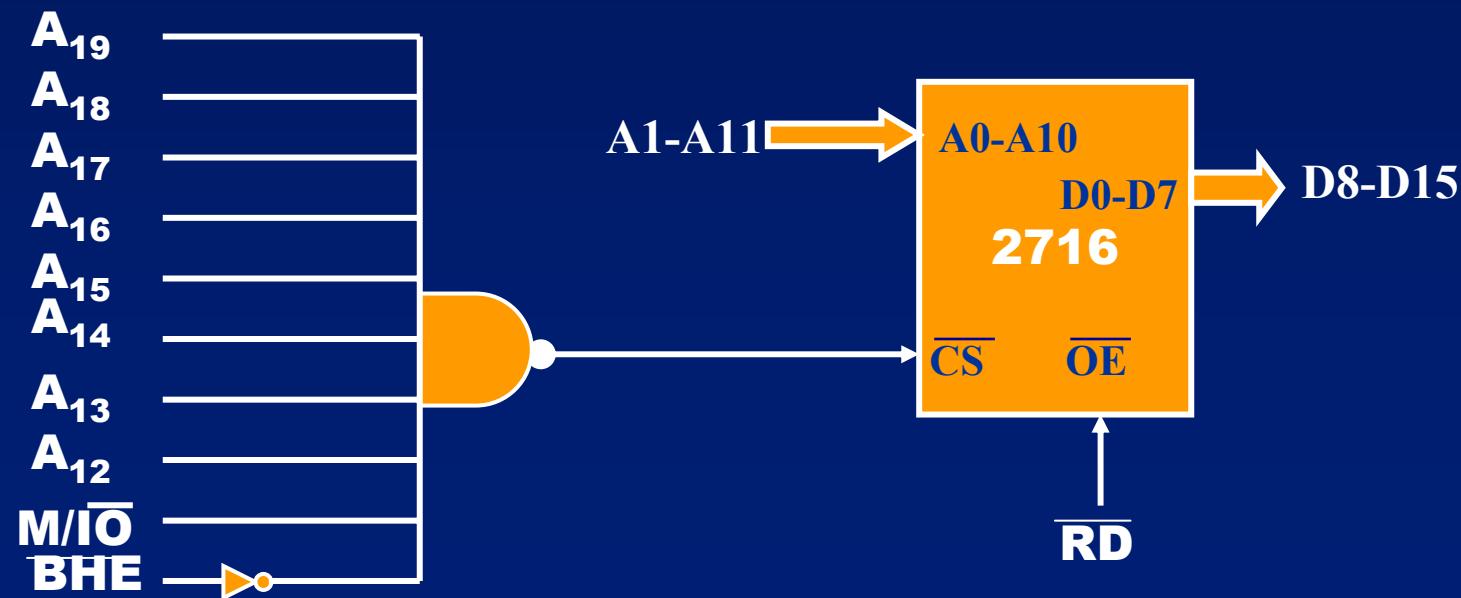
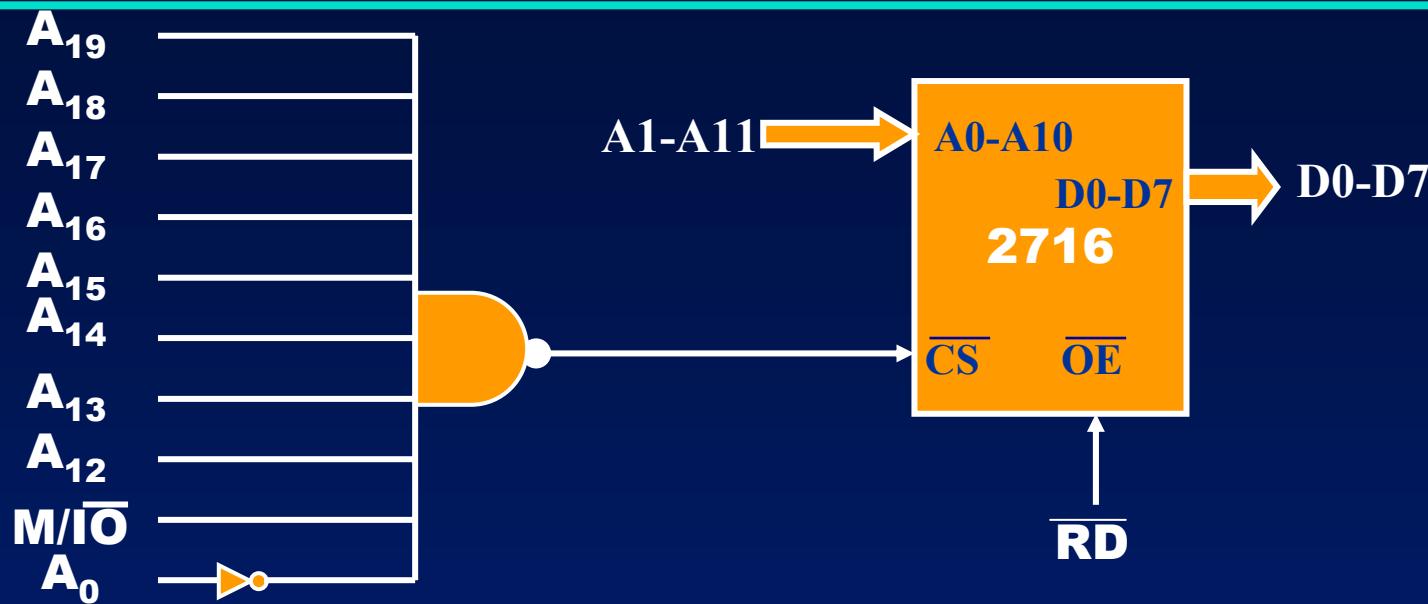
<b>BHE</b>	<b>A0</b>	<b>Chức năng</b>
0	0	chọn cả 2 bank
0	1	chọn bank cao
1	0	chọn bank thấp
1	1	không chọn bank nào



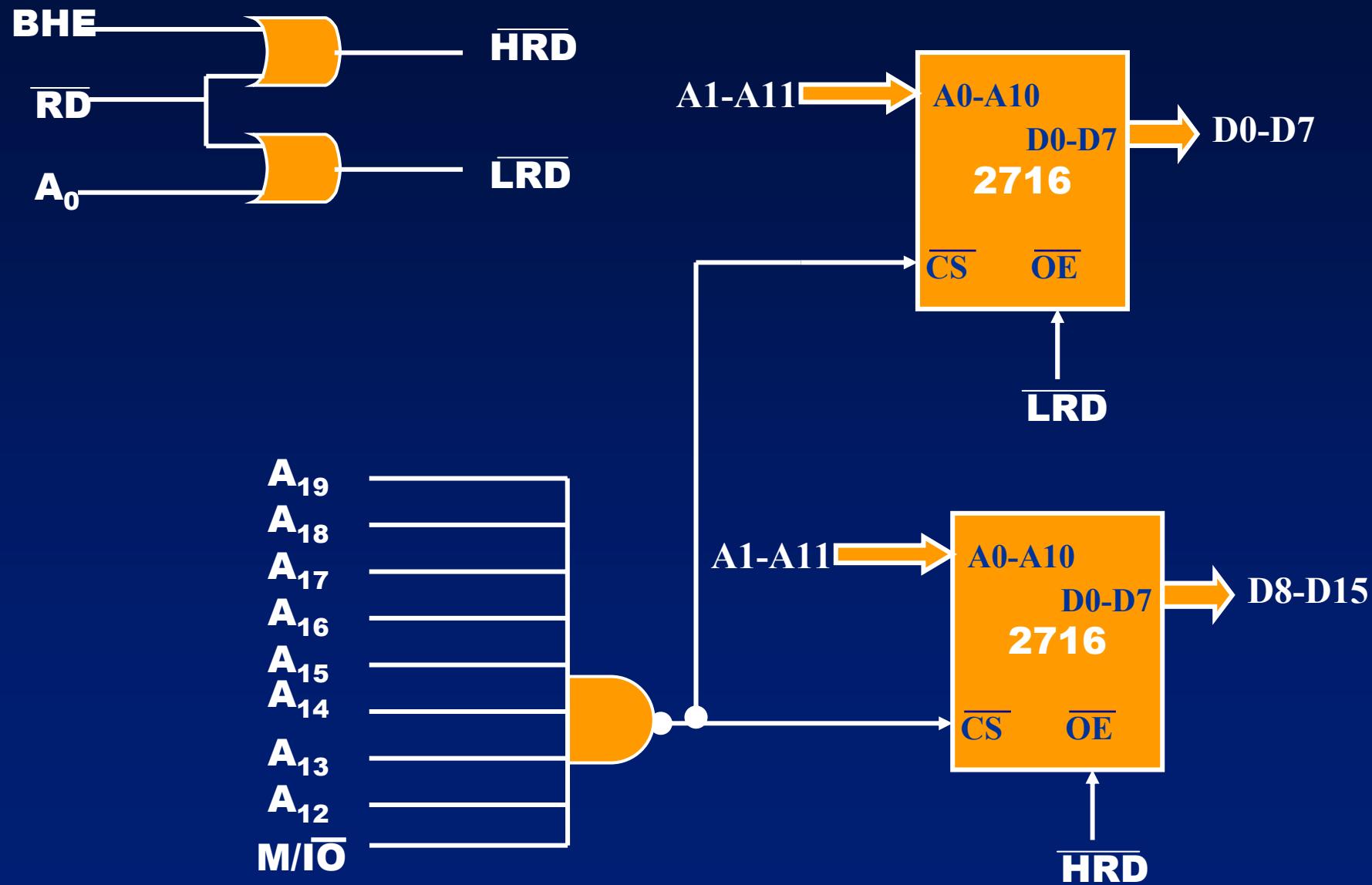
# Ghép nối 8086 với bộ nhớ

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

# Ghép nối 8086 với bộ nhớ

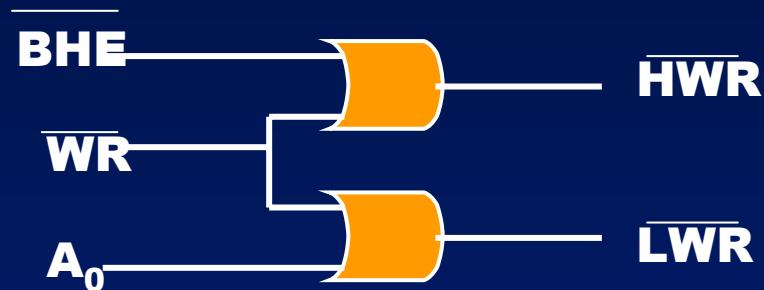


# Ghép nối 8086 với bộ nhớ



# 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





# 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)**

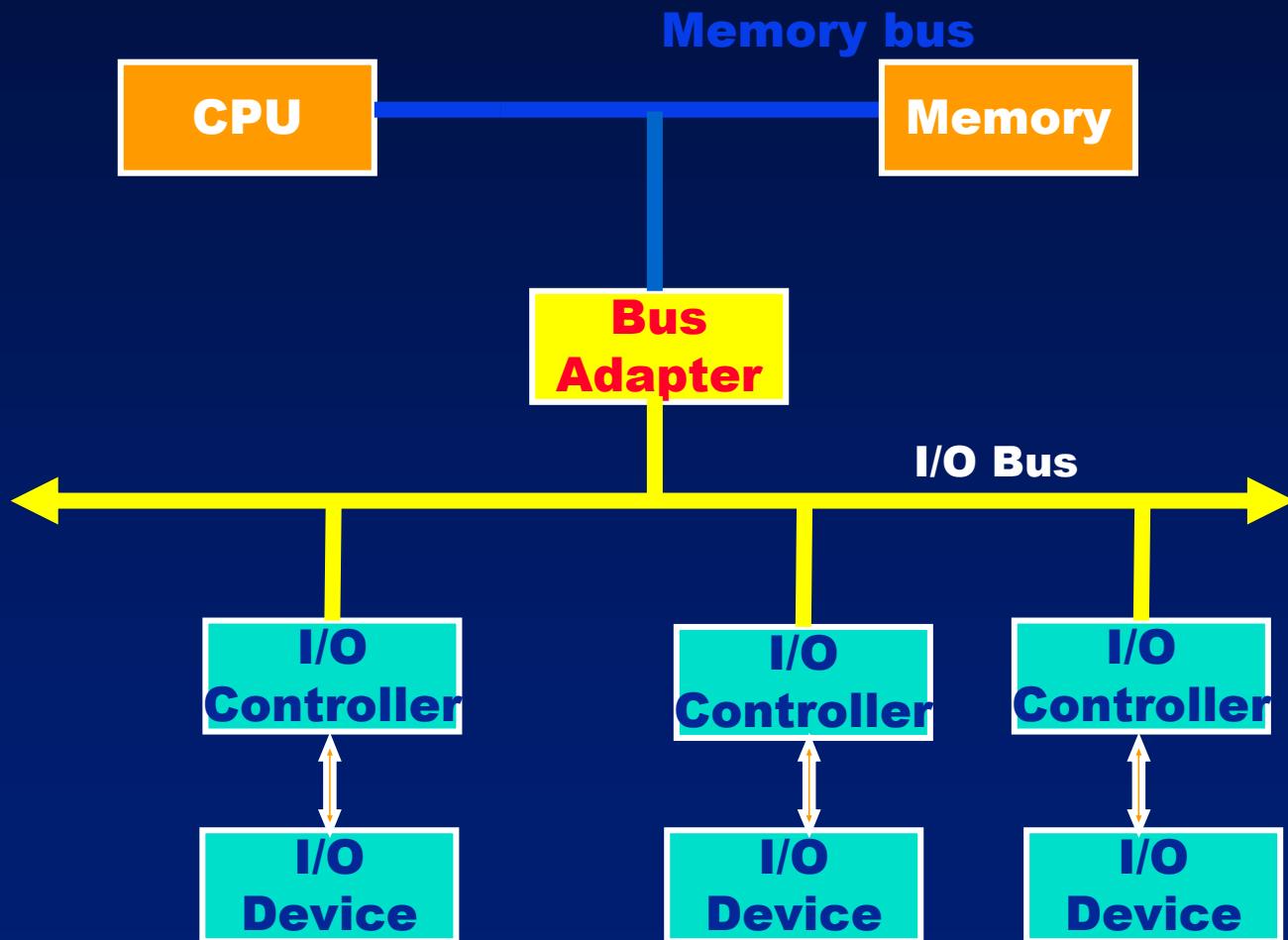
# Chương 4: Tổ chức vào ra dữ liệu

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

# Chương 4: Tổ chức vào ra dữ liệu

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

# Các kiểu giao tiếp giữa vi xử lý và thiết bị ngoại vi



# Các kiểu giao tiếp giữa vi xử lý và thiết bị ngoại vi

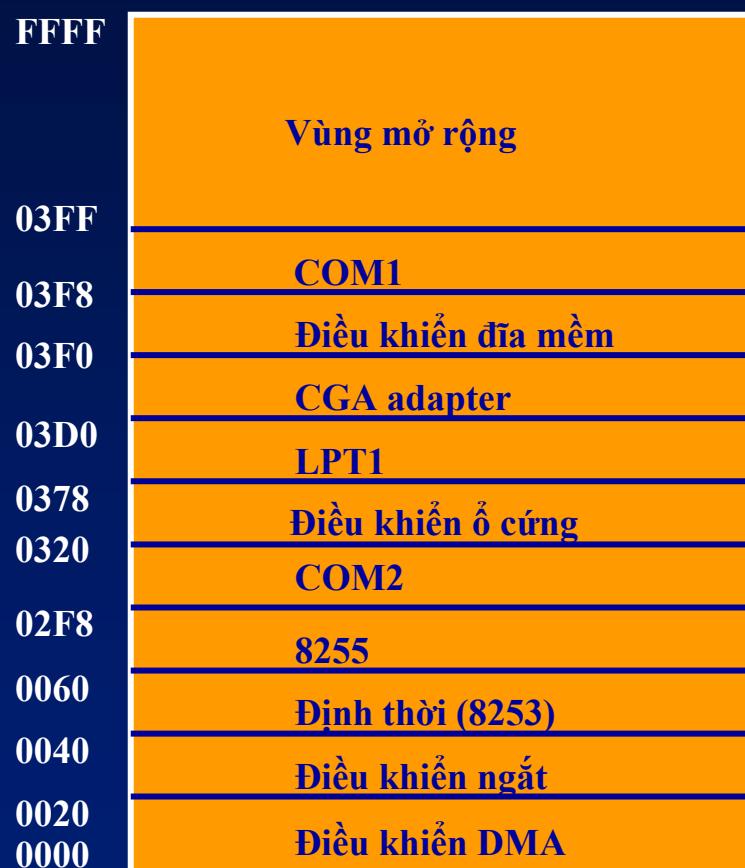
- **Giao tiếp kiểu thăm dò, móc nối (handshaking)**
  1. CPU kiểm tra trạng thái của thiết bị ngoại vi
  2. Nếu thiết bị ngoại vi sẵn sàng trao đổi dữ liệu việc trao đổi sẽ được thực hiện bởi tín hiệu móc nối
  3. Nếu thiết bị ngoại vi chưa sẵn sàng, CPU sẽ thực hiện công việc khác và quay lại bước 1
- **Giao tiếp bằng ngắt (Interrupt)**
  1. Thiết bị ngoại vi muốn trao đổi dữ liệu với CPU, nó sẽ gửi tín hiệu yêu cầu ngắt tới chân INTR của CPU
  2. CPU chấp nhận yêu cầu ngắt bằng cách gửi tín hiệu INTA tới thiết bị ngoại vi
  3. CPU thực hiện chương trình con phục vụ ngắt
- **Giao tiếp bằng truy cập bộ nhớ trực tiếp (DMA)**
  1. Thiết bị ngoại vi muốn truy cập trực tiếp bộ nhớ không thông qua CPU, nó đưa tín hiệu yêu cầu tới chân HOLD của CPU thông qua khối điều khiển DMA
  2. CPU chấp nhận và gửi tín hiệu HLDA tới khối điều khiển DMA và treo các bus
  3. Khối điều khiển DMA sẽ điều khiển việc trao đổi dữ liệu giữa thiết bị ngoại vi và bộ nhớ

# Chương 4: Tổ chức vào ra dữ liệu

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

# Các kiểu ghép nối vào ra

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



**Địa chỉ: 0000H-FFFFH**  
**M/I $\bar{O}$ =0**

**Vào ra dữ liệu bằng lệnh IN, OUT**

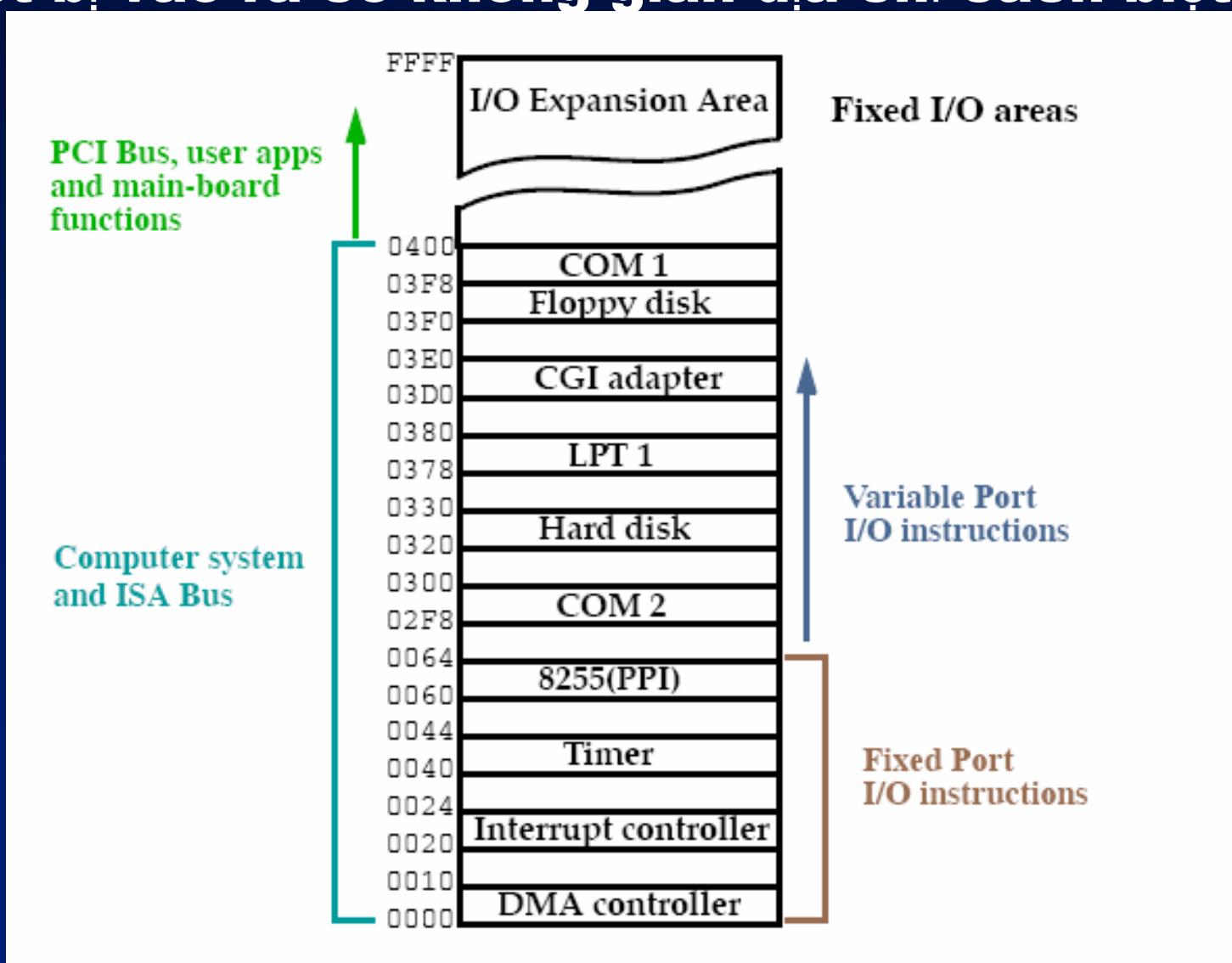
**Ví dụ:**

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

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

# Các kiểu ghép nối vào ra

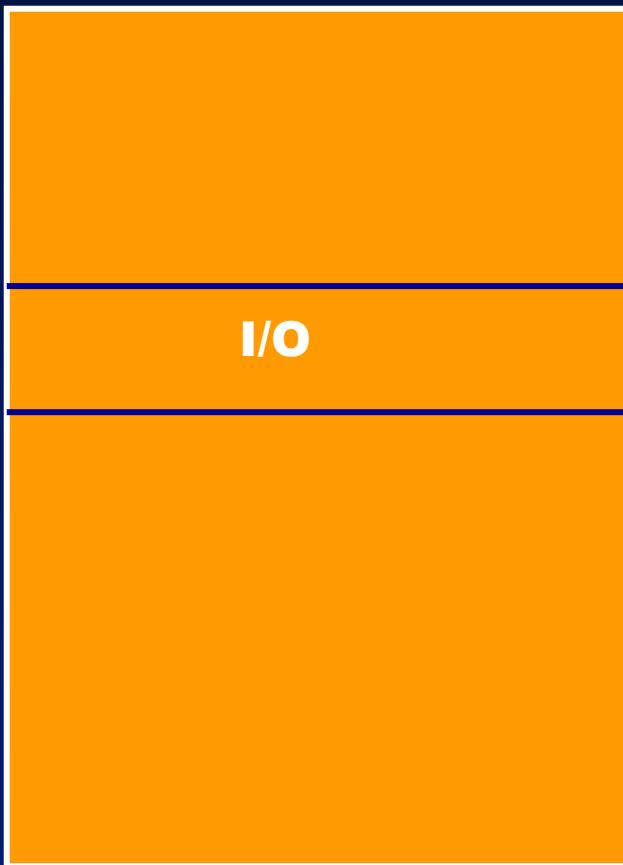
- Thiết bị vào ra có không gian địa chỉ cách biệt:



# Các kiểu ghép nối vào ra

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

FFFF



$$M/I\bar{O}=1$$

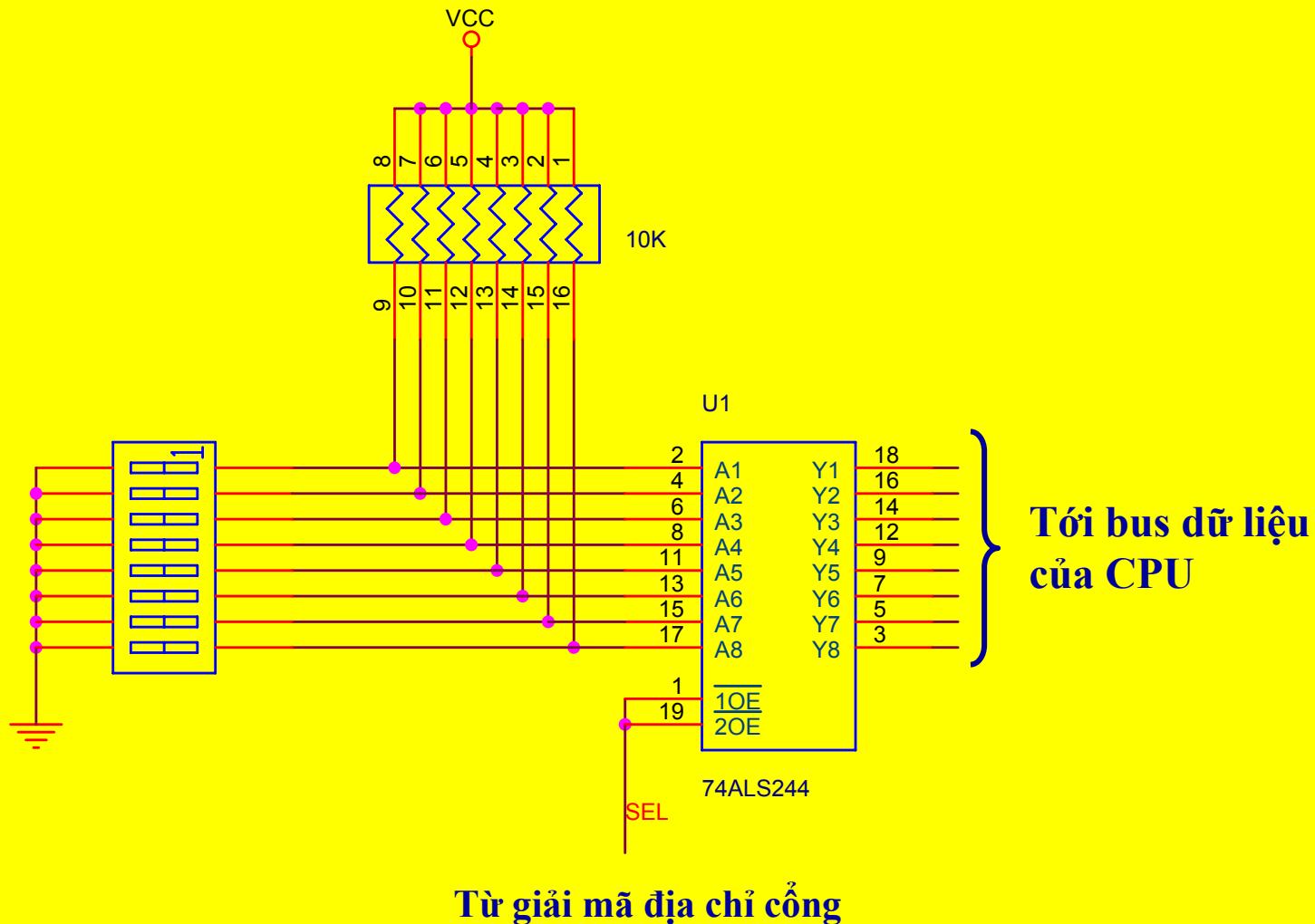
Vào ra dữ liệu bằng bất kỳ lệnh  
di chuyển dữ liệu nào giữa  
CPU và bộ nhớ

Ví dụ:

**MOV AX, [0FF3H]**

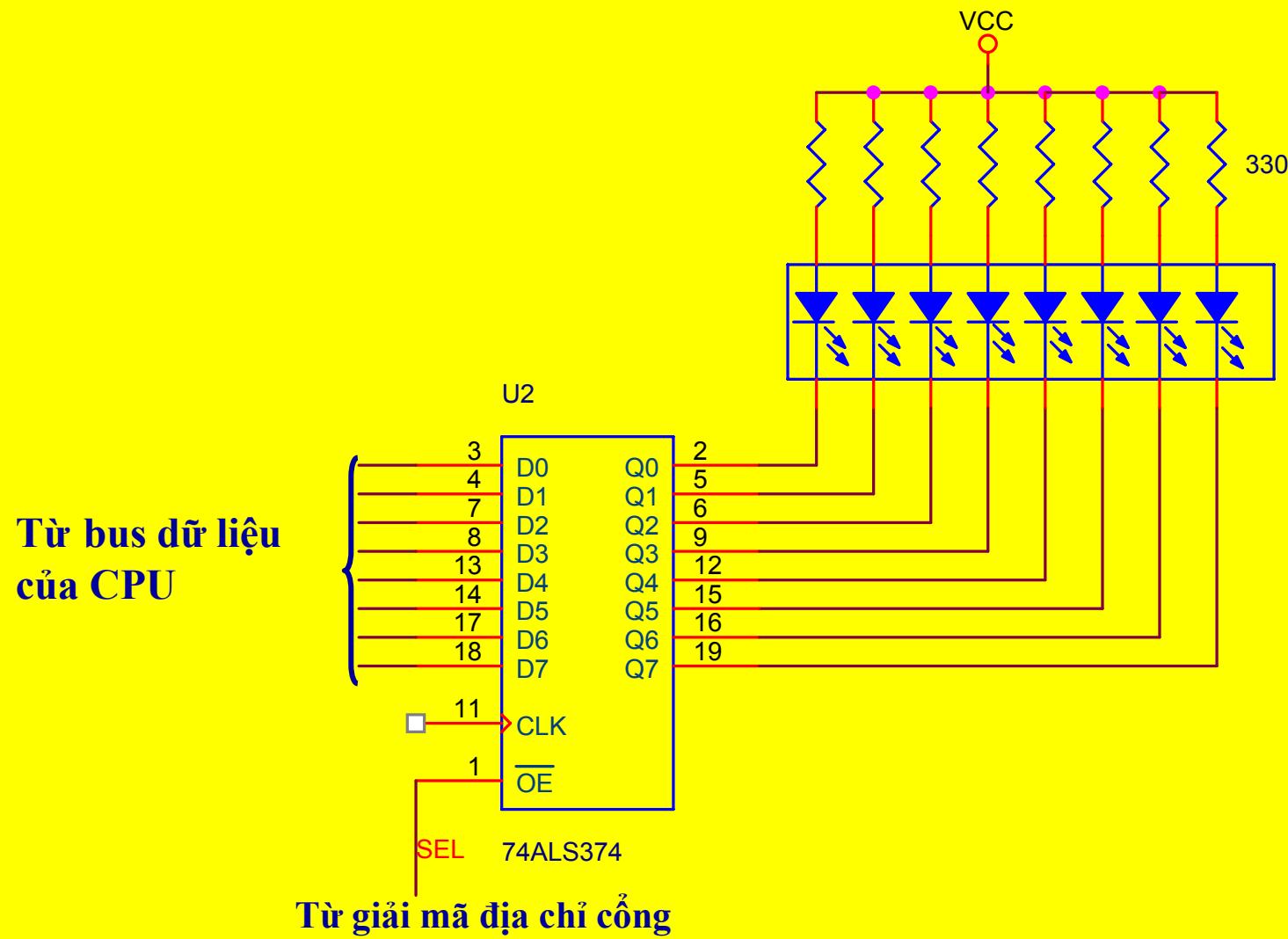
# Các kiểu ghép nối vào ra

- Ví dụ cỗng vào đơn giản:



# Các kiểu ghép nối vào ra

- Ví dụ cỗng ra đơn giản:



# Chương 4: Tổ chức vào ra dữ liệu

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

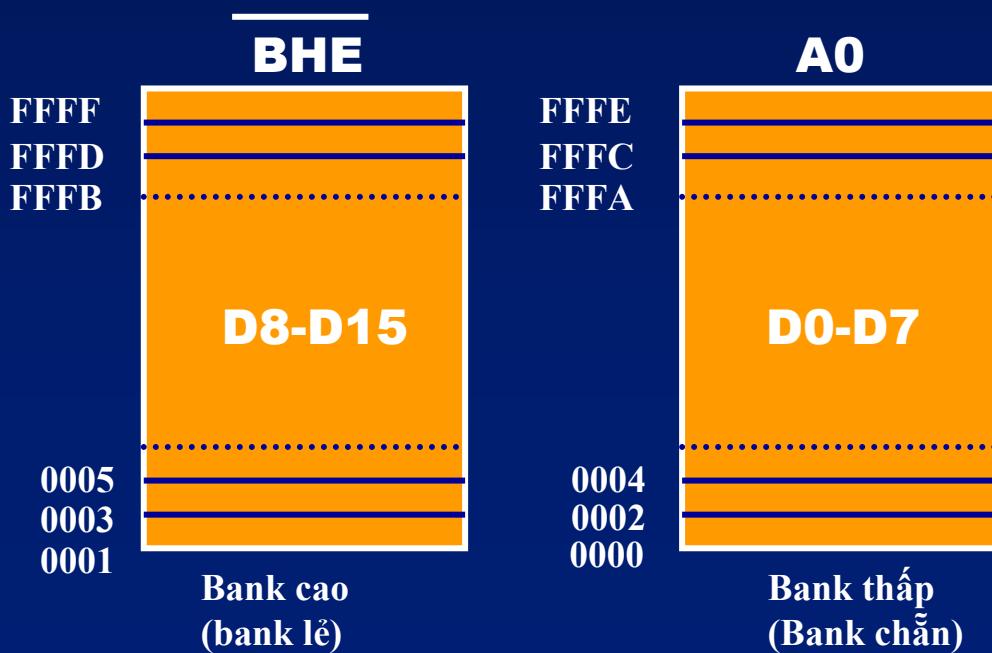
# Giải mã địa chỉ cho các thiết bị vào/ra

- **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

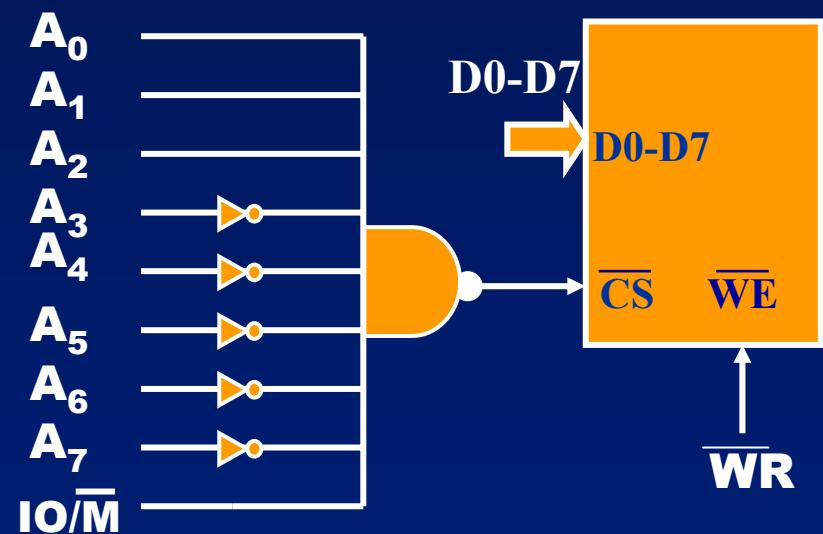
- **8 bit dữ liệu hay 16 bit?**

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

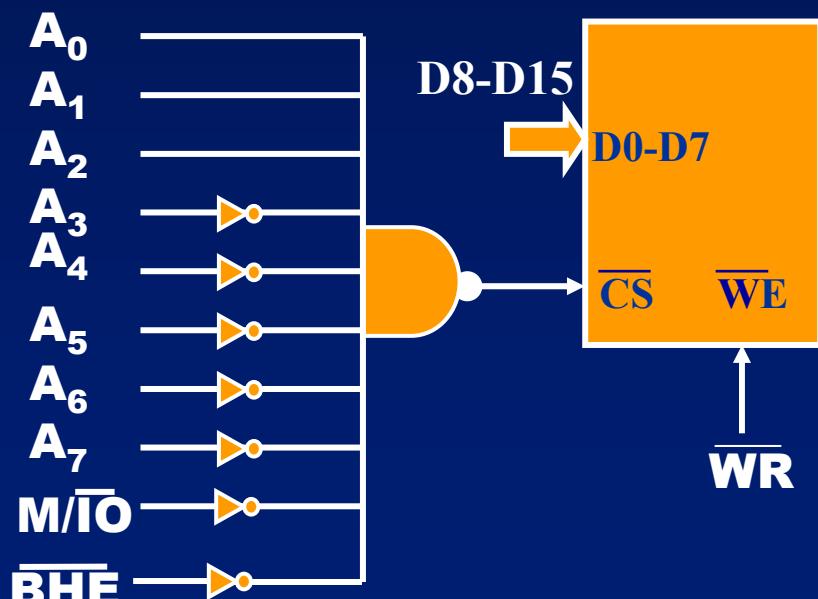


# Giải mã địa chỉ cho các thiết bị vào/ra

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



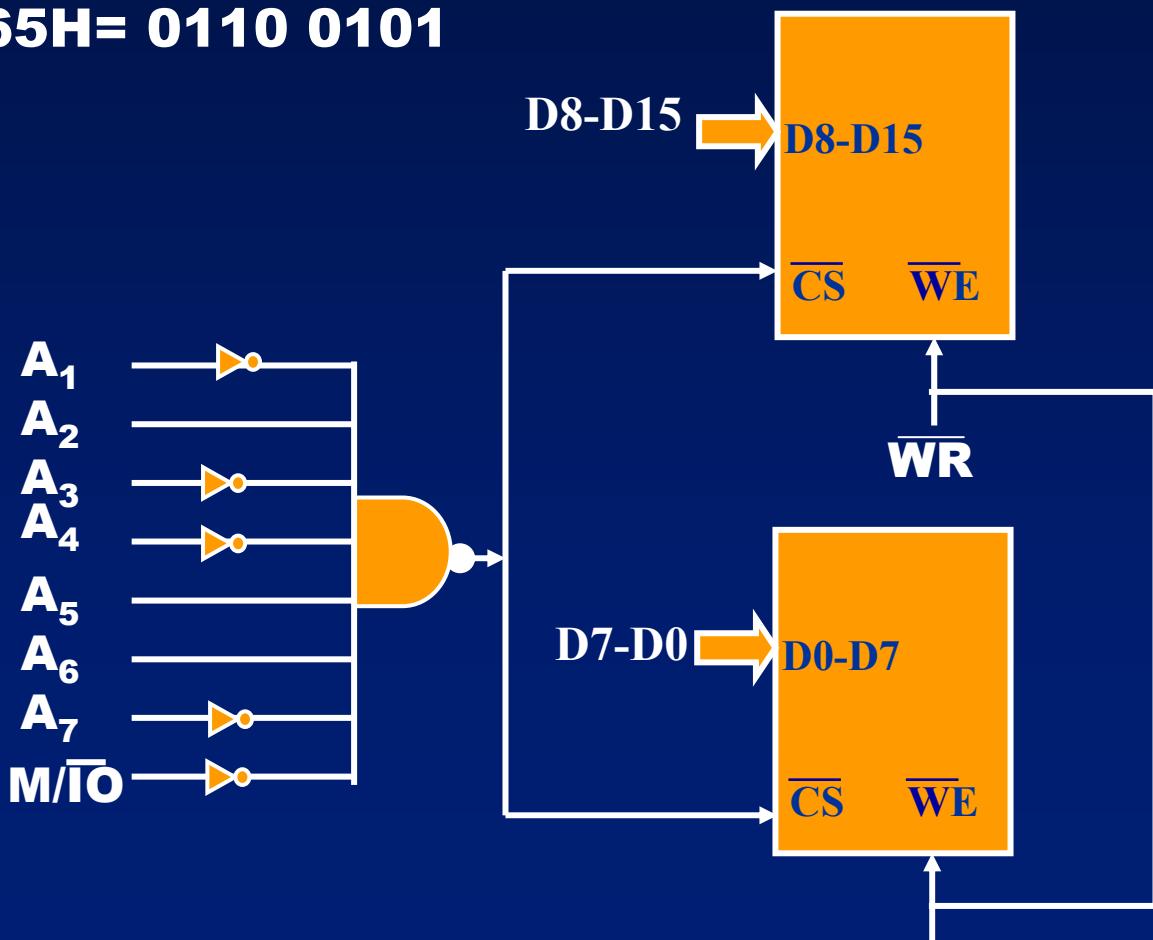
8088



8086

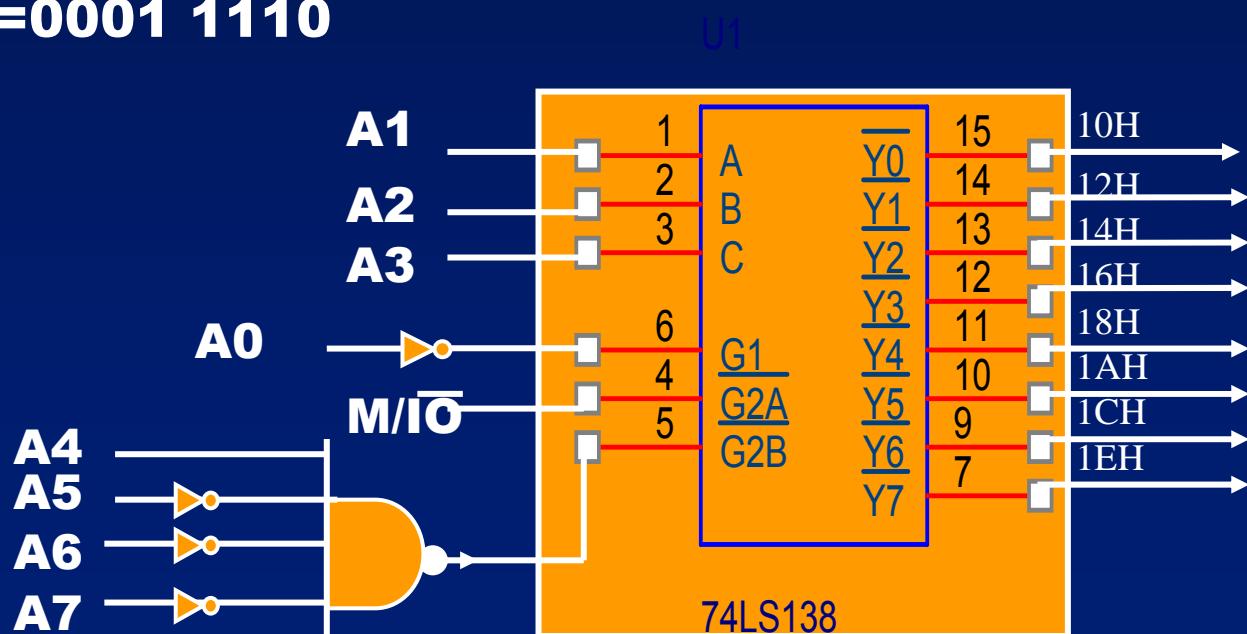
# Giải mã địa chỉ cho các thiết bị vào/ra

- Ví dụ: Giải mã địa chỉ cho thiết bị ra 16 bit với địa chỉ  
cỗng 64H và 65H
  - $64H = 0110\ 0100$
  - $65H = 0110\ 0101$



# 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





# Chương 4: Tổ chức vào ra dữ liệu

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

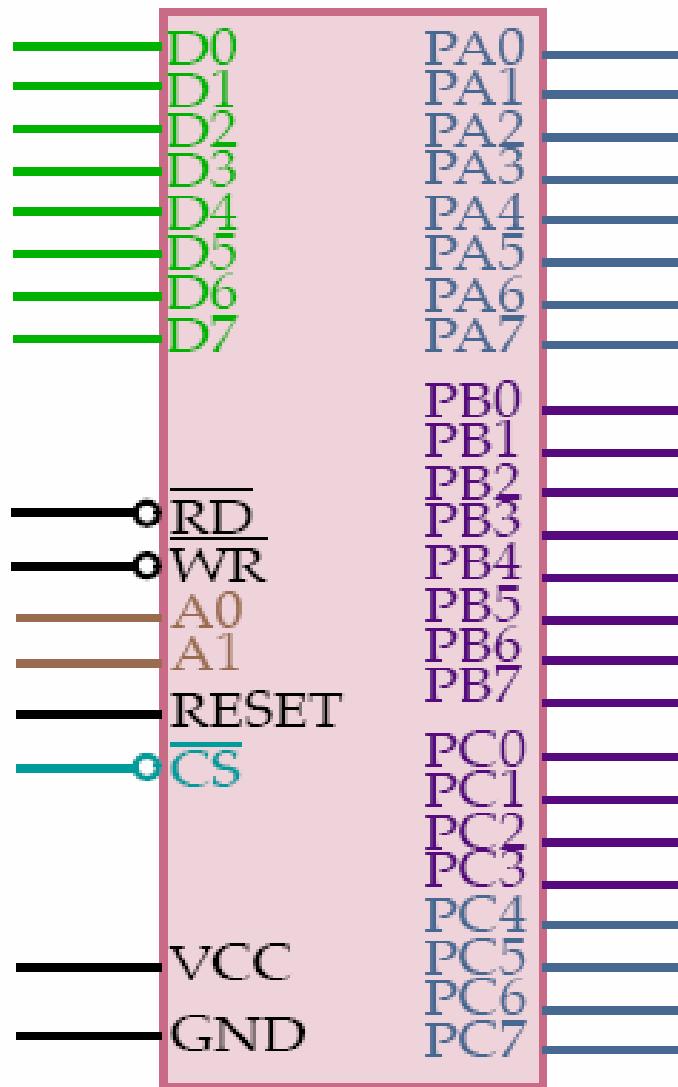


# Cấu trúc của 8255A

- **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 chip 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**

# Cấu trúc của 8255A

82C55



## Group A

Port A (PA7-PA0) and upper half of port C (PC7 - PC4)

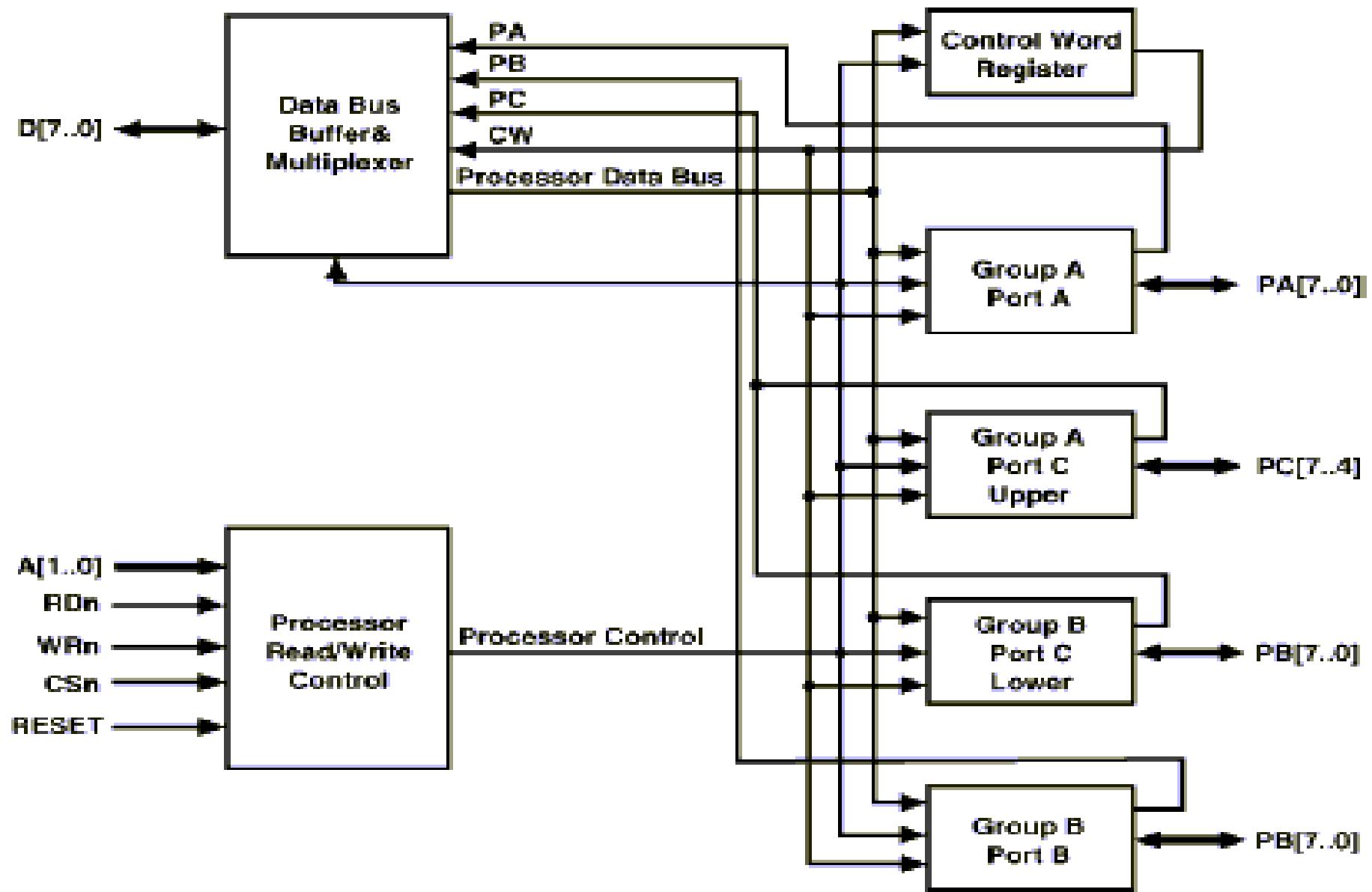
## Group B

Port B (PB7-PB0) and lower half of port C (PC3 - PC0)

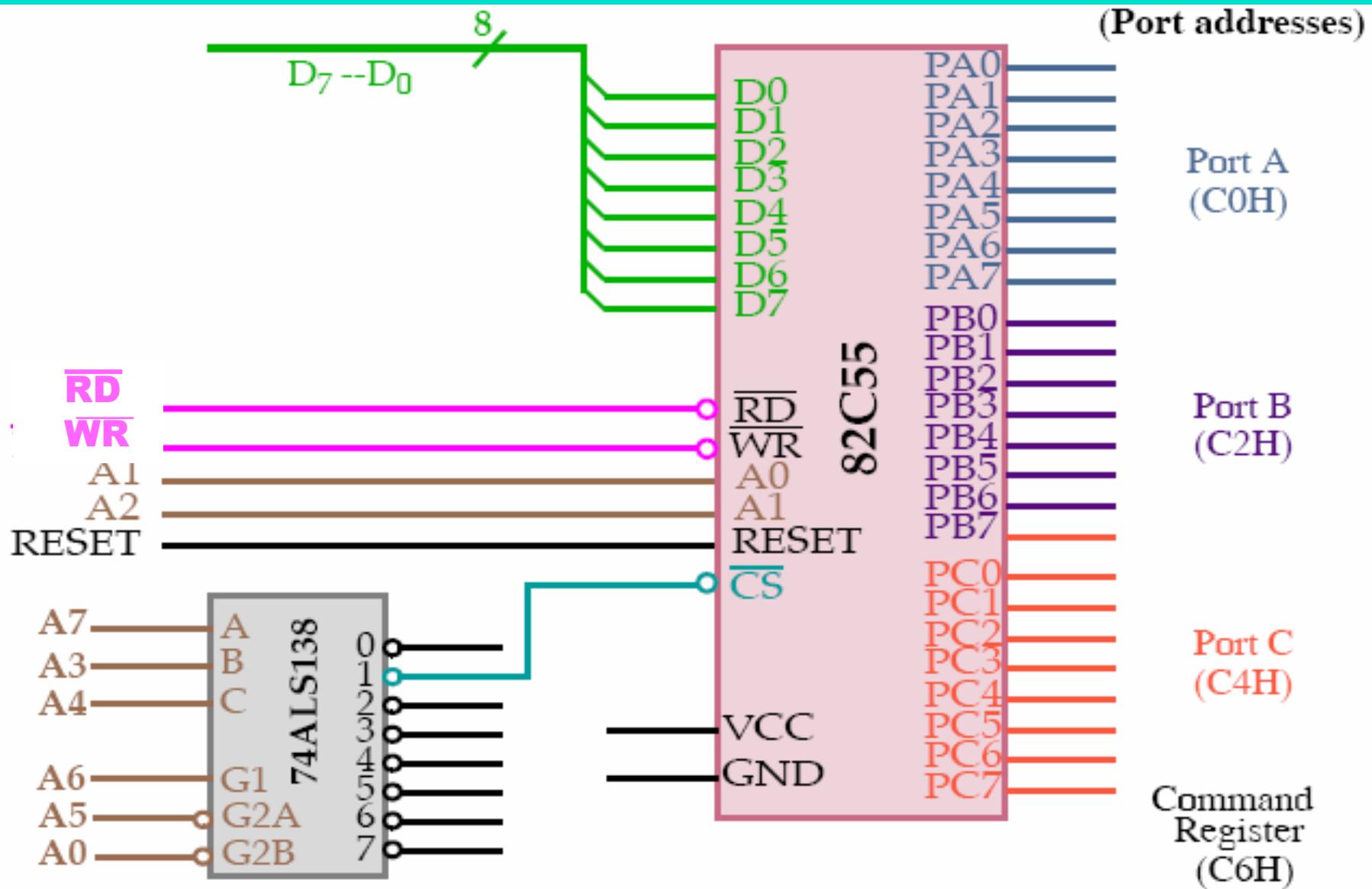
## I/O Port Assignments

$A_1$	$A_0$	Function
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Command Register

# Cấu trúc của 8255A

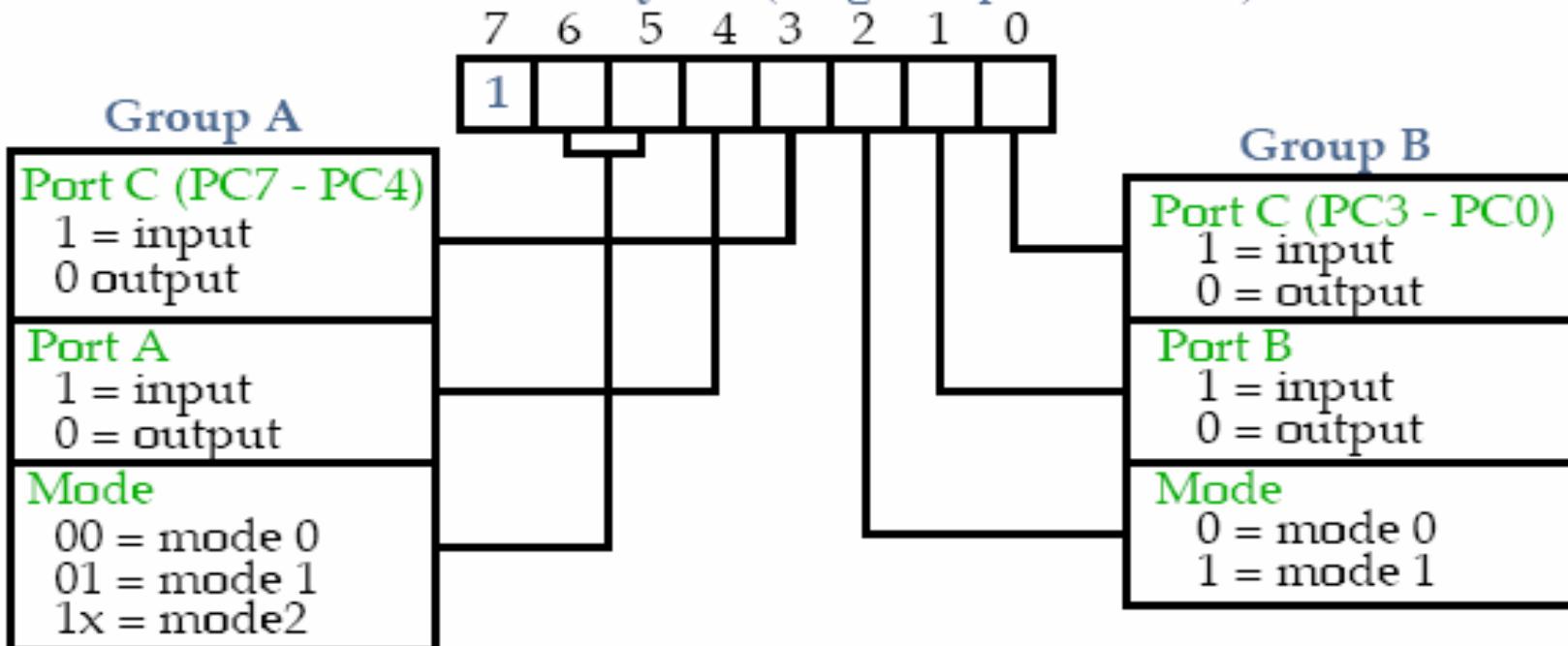


# Cấu trúc của 8255A

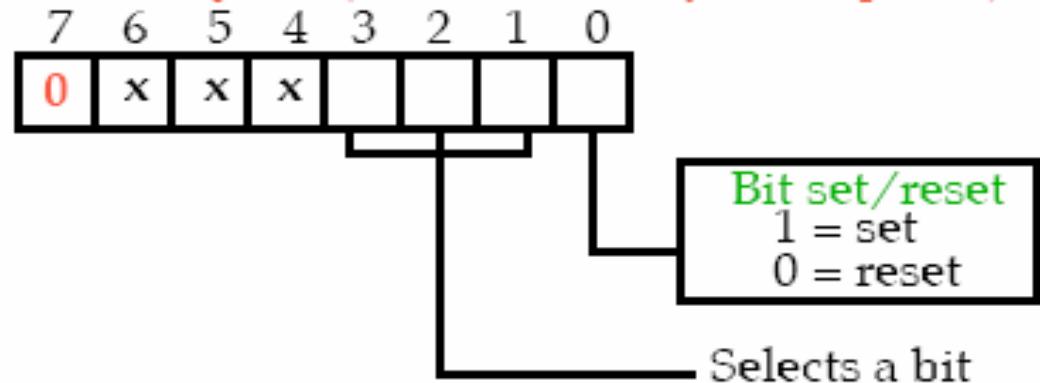


# Các chế độ làm việc của 8255A

Command Byte A (Programs ports A, B, C)



Command Byte B (Sets or resets any bits in port C)



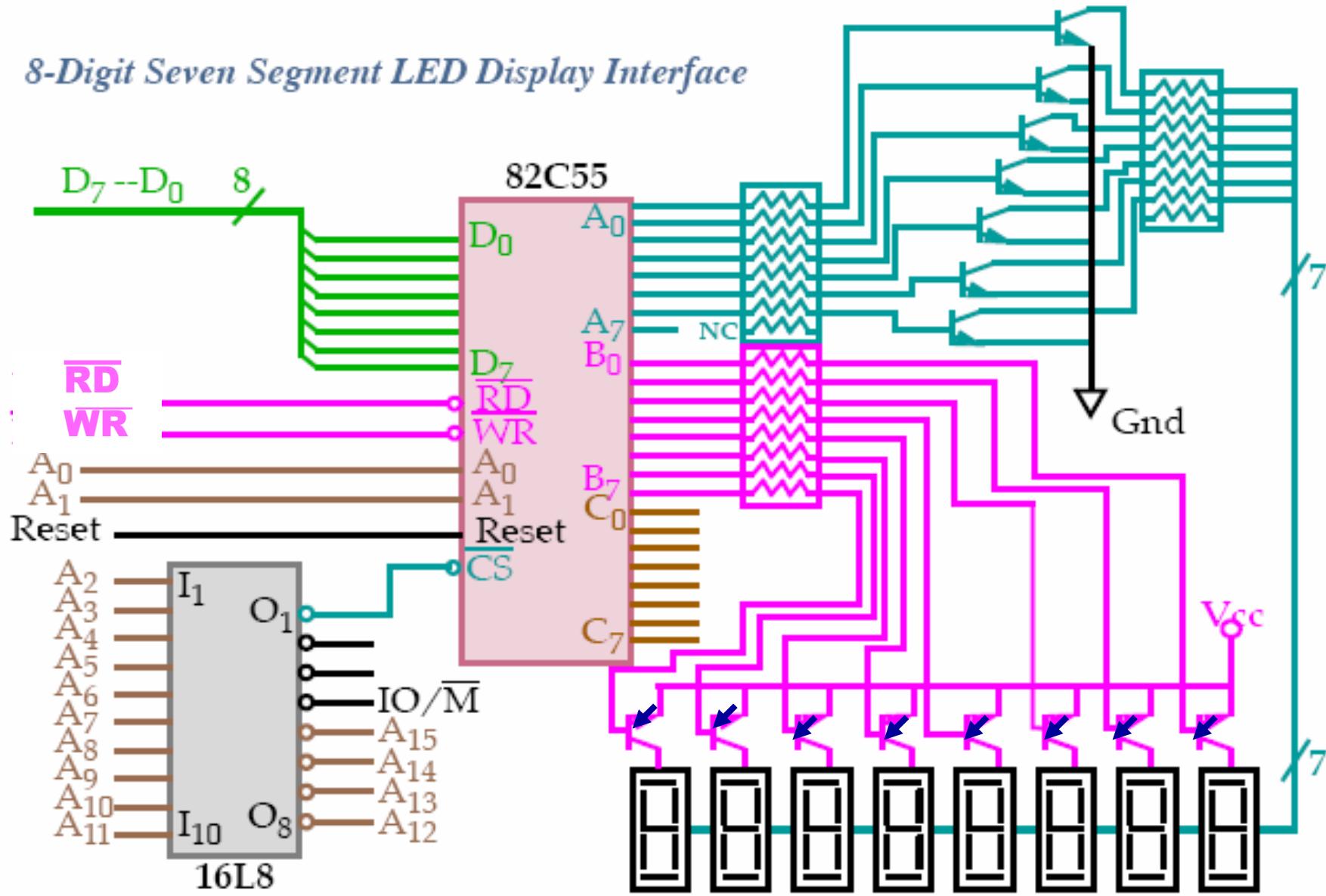


# Các chế độ làm việc của 8255A

- **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<sub>H</sub> đảm nhiệm. Cổng B có thể làm việc ở chế độ 1 hoặc 0**

# Chế độ 0

8-Digit Seven Segment LED Display Interface



• Giả th  
0703I

; Lập trình cho 8255

```
MOV AL, 10000000B
MOV DX, 703H
OUT DX, AL
```

; Port A, Port B mode 0, output

; Thủ tục hiển thị LED từ dữ liệu chứa trong bộ nhớ

```
DISP PROC NEAR
    PUSHF
    PUSH AX
    PUSH BX
    PUSH DX
    PUSH SI
```

; cất các thanh ghi vào ngăn xếp

; Thiết lập các thanh ghi để hiển thị

```
MOV BX, 8
MOV AH, 7FH
MOV SI, OFFSET MEM-1
MOV DX, 701H
```

; số LED  
; chọn LED đầu tiên 0111 1111  
; địa chỉ chứa dữ liệu  
; địa chỉ cổng B

; 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
```

; chọn 1 số  
; địa chỉ cổng A  
; dữ liệu của 7 đoạn led

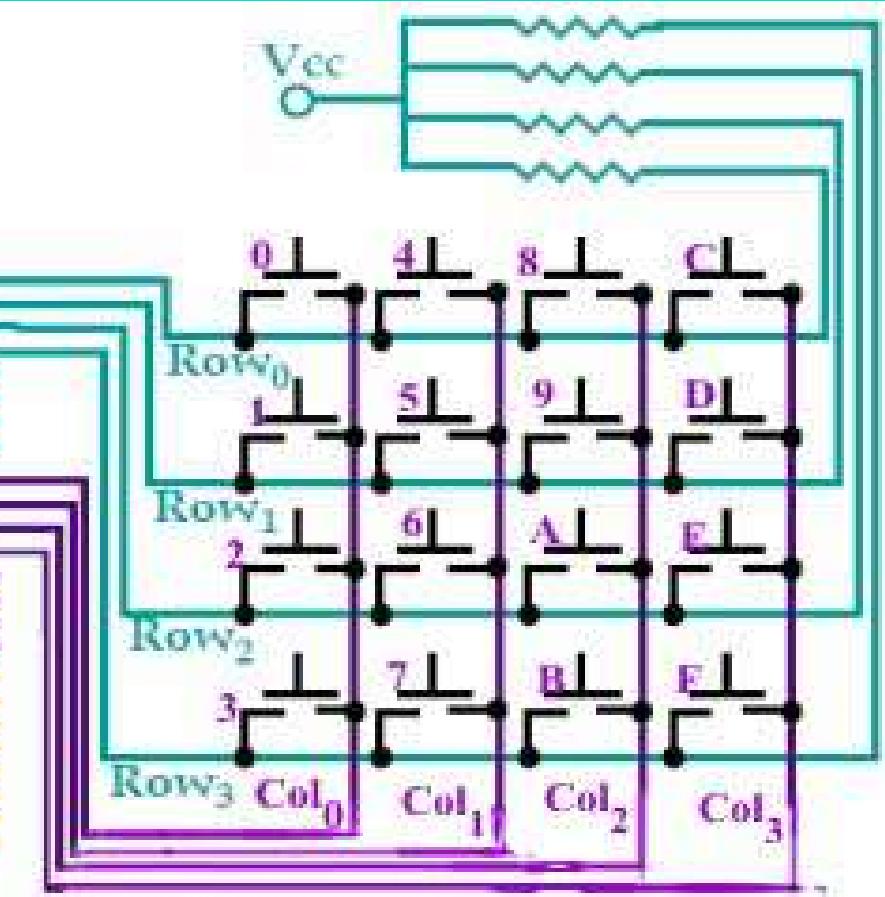
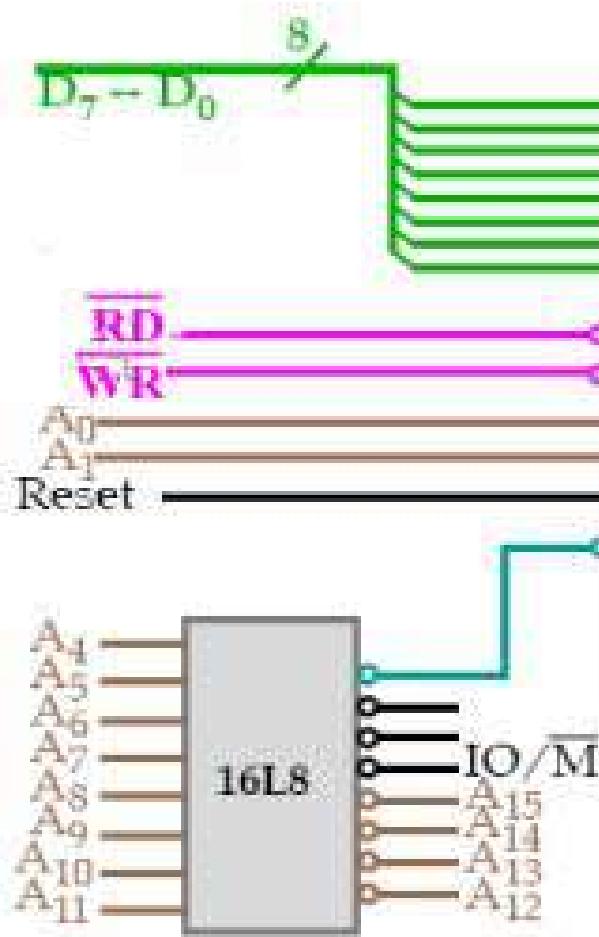
; trễ 1 ms  
; số tiếp theo  
; địa chỉ cổng B  
; giảm chỉ số  
; lặp lại 8 lần

; khôi phục lại các thanh ghi

```
POP SI
POP DX
POP BX
POP AX
POPF
RET
DISP ENDP
```

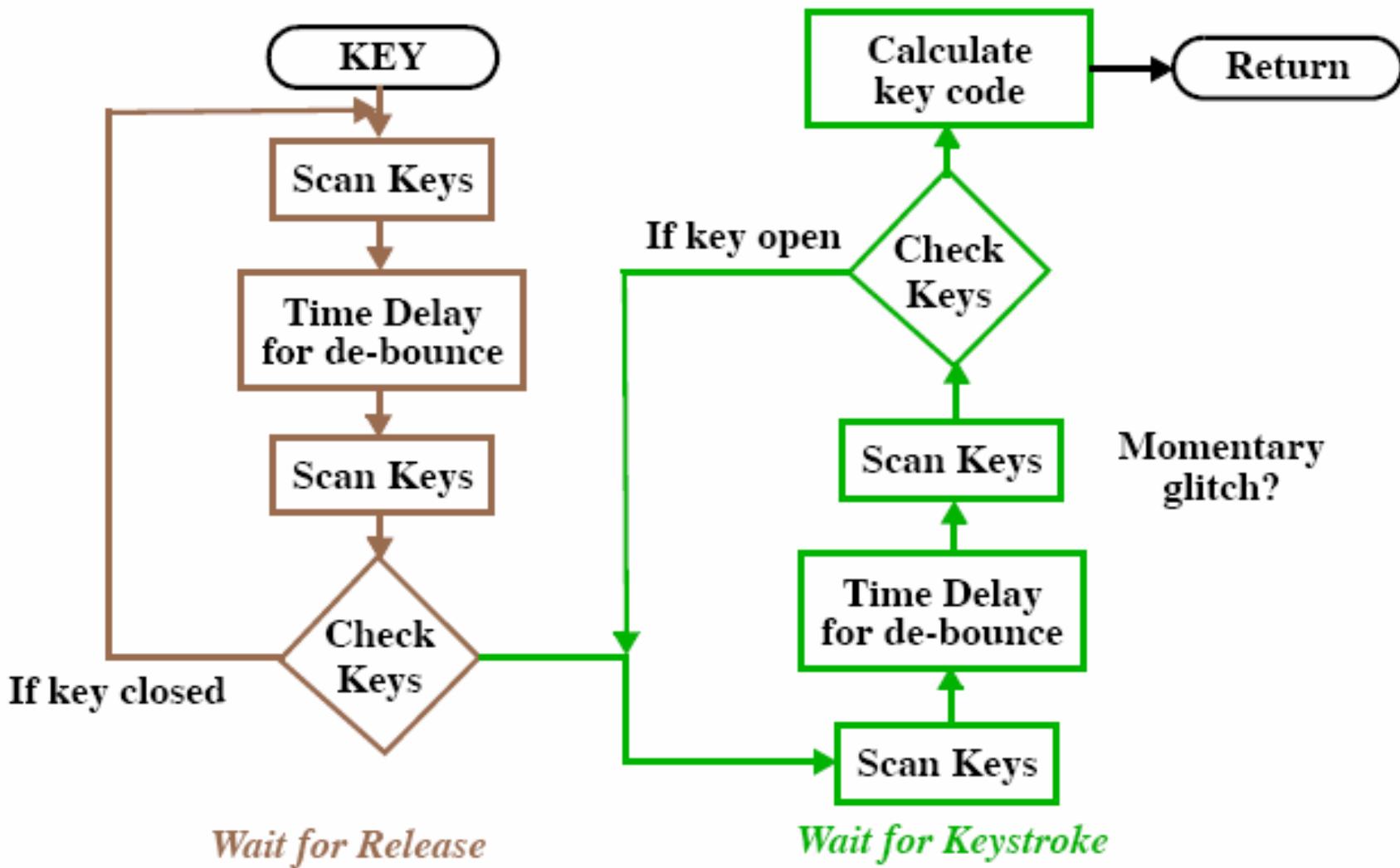
# Chế độ 0

*4x4 keyboard matrix interface*



# Chế độ 0

*Flow chart of a keyboard-scanning procedure*



# Chế độ 0

ROWS	EQU	4 ; 4 hàng
COLS	EQU	4 ; 4 cột
PORTA	EQU	50H
PORTB	EQU	51H
KEY	PROC NEAR	USES CX
	CALL SCAN	;test all keys
	JNZ KEY	; if key closed
	CALL DELAY	; đợi 10 ms
	CALL SCAN	
	JNZ KEY	
KEY1:	CALL SCAN	
	JZ KEY1	; if no key closed
	CALL DELAY	
	CALL SCAN	
	JZ KEY1	
	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	
	ENDP	

SCAN	PROC	NEAR	USES	BX
	MOV CL, ROWS	;form row mask		
	MOV BH, OFFH			
	SHL BH, CL			
	MOV CX, COLS	;load column count		
	MOV BL, OFEH	;get selection mode		
SCAN1:	MOV AL, BL	;select column		
	OUT PORTB, AL			
	ROL BL, 1			
	IN AL, PORTA;	read rows		
	OR AL, BH			
	CMP AL, 0FFH	;test for a key		
	JNZ SCAN2			
	LOOP SCAN1			
SCAN2:	RET			
	SCAN	ENDP		
DELAY:	PROC	NEAR	USES	CX
	MOV CX, 5000	;10ms (8MHZ)		
DELAY1:	LOOP	DELAY1		
	RET			
	ENDP			

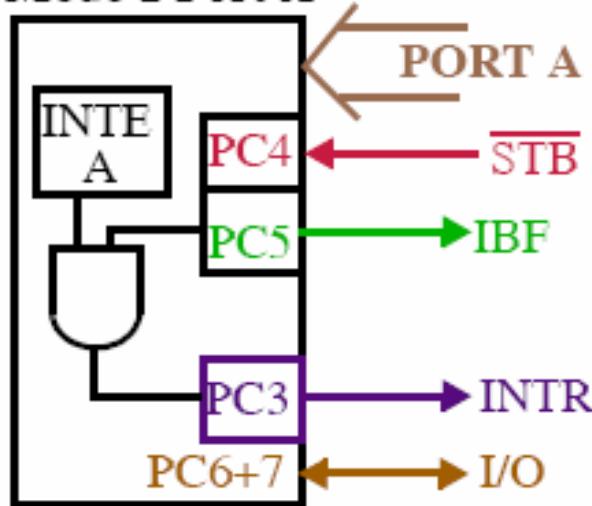
# Chế độ 1

- **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

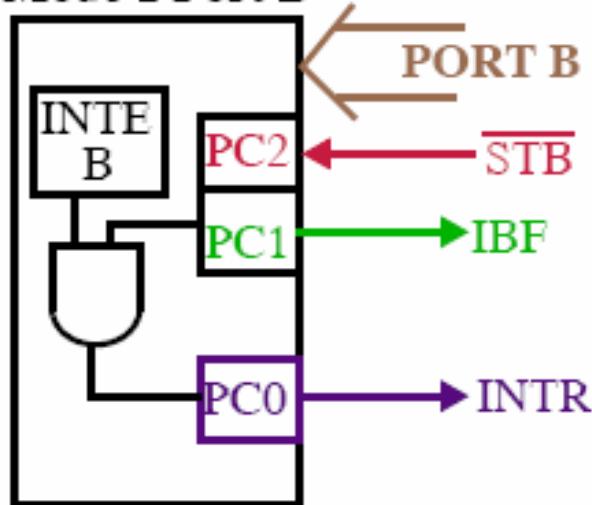
STB	The <b>strobe</b> input loads data into the port latch on a 0-to-1 transition
IFB	<b>Input buffer full</b> is an output indicating that the input latch contain information
INTR	<b>Interrupt request</b> is an output that requests an interrupt
INTE	The <b>interrupt enable</b> signal is neither an input 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.

# Chế độ 1

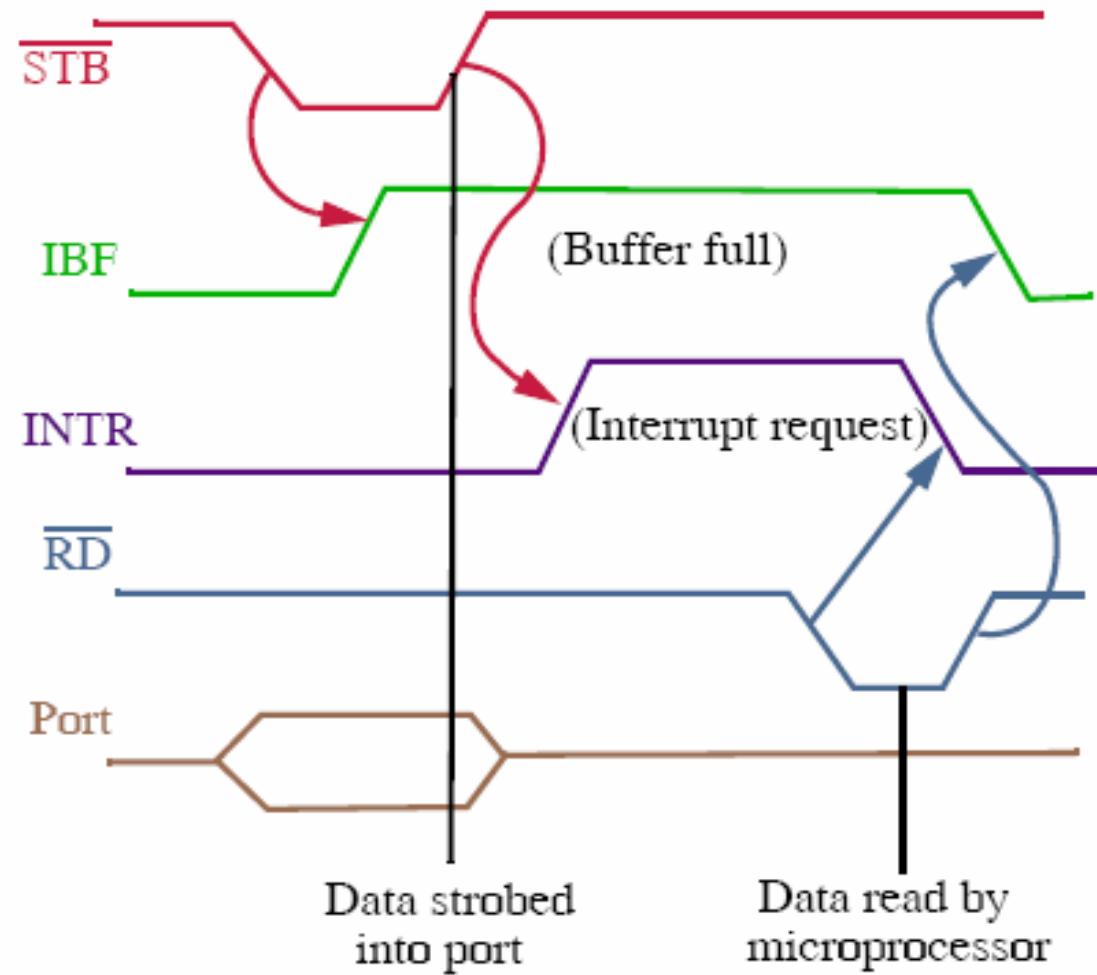
Mode 1 Port A



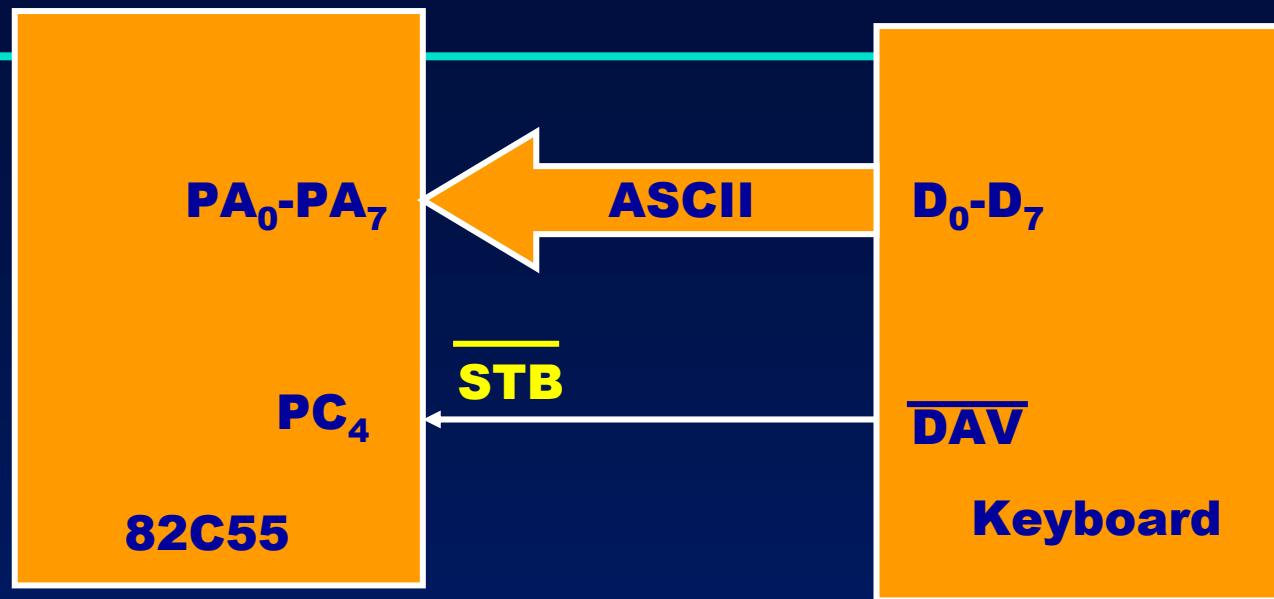
Mode 1 Port B



Timing Diagram



# Chế độ 1



<b>Bit5</b>	<b>EQU</b>	<b>20H</b>	
<b>PortC</b>	<b>EQU</b>	<b>22H</b>	
<b>PortA</b>	<b>EQU</b>	<b>20H</b>	
<b>Read</b>	<b>PROC NEAR</b>		
<b>IN</b>	<b>AL, PortC</b>	<b>; read PortC</b>	
<b>Test</b>	<b>AL, Bit5</b>	<b>; test IBF</b>	
<b>JZ</b>	<b>READ</b>	<b>; if IBF=0</b>	
<b>IN</b>	<b>AL, PortA</b>	<b>; read data</b>	
<b>RET</b>			
<b>Read Endp</b>			

# Chế độ 1

- 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**

**OBF**

**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**

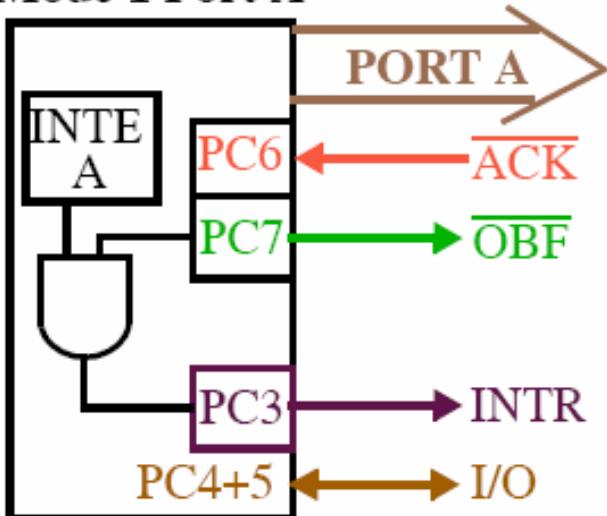
The **interrupt enable signal** is neither an input nor an output; it is an internal bit programmed via the PC6(port A) or PC2(port B) bits.

**PC5,PC4**

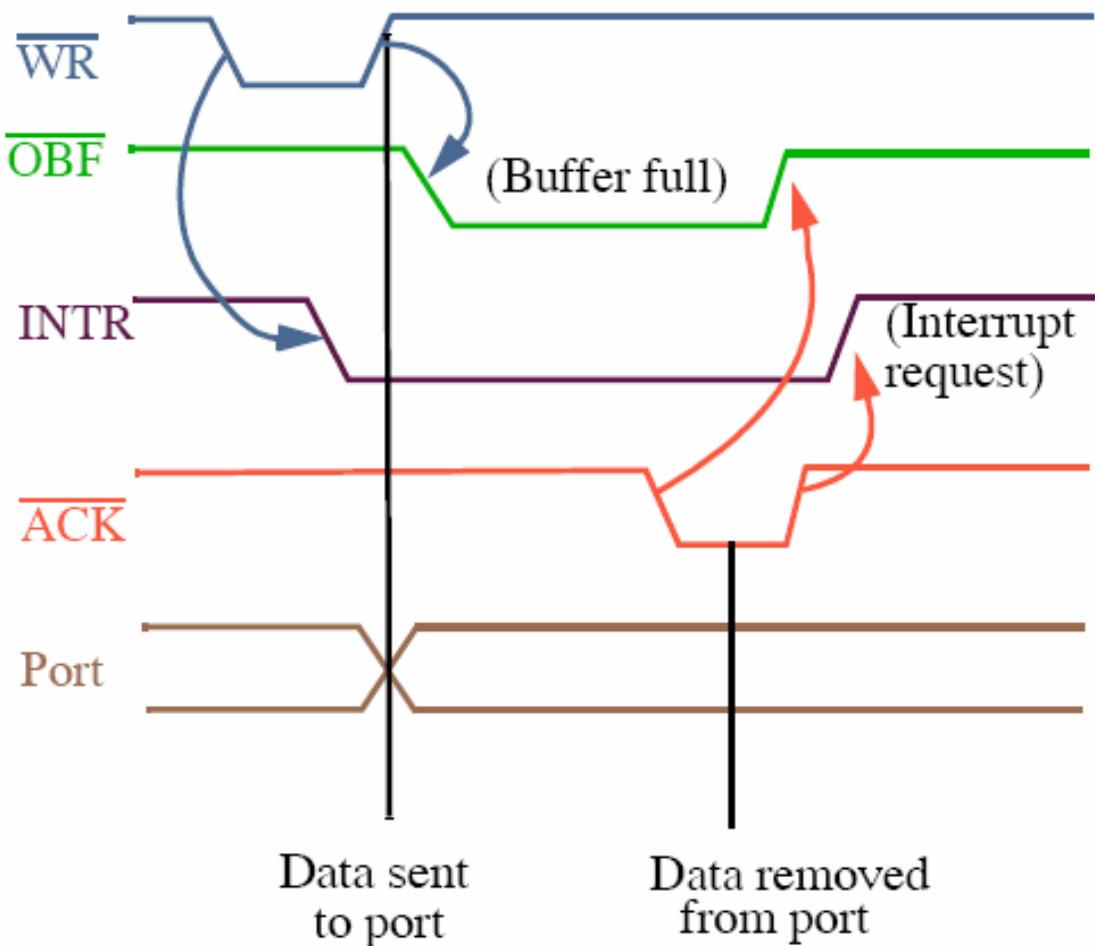
The port C pins 5 and 4 are general-purpose I/O pins that are available for any purpose.

# Chế độ 1

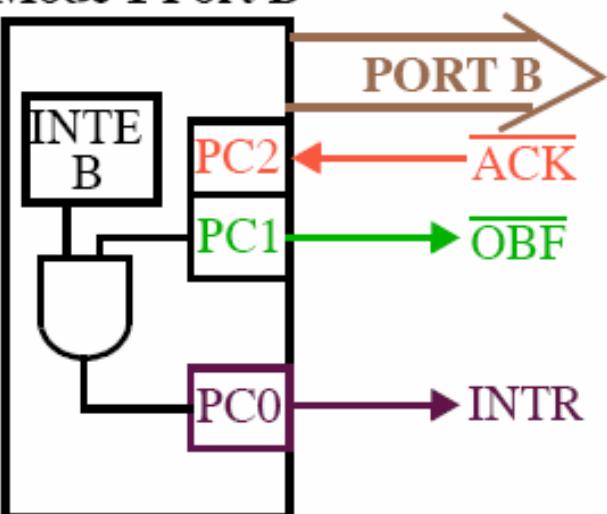
Mode 1 Port A



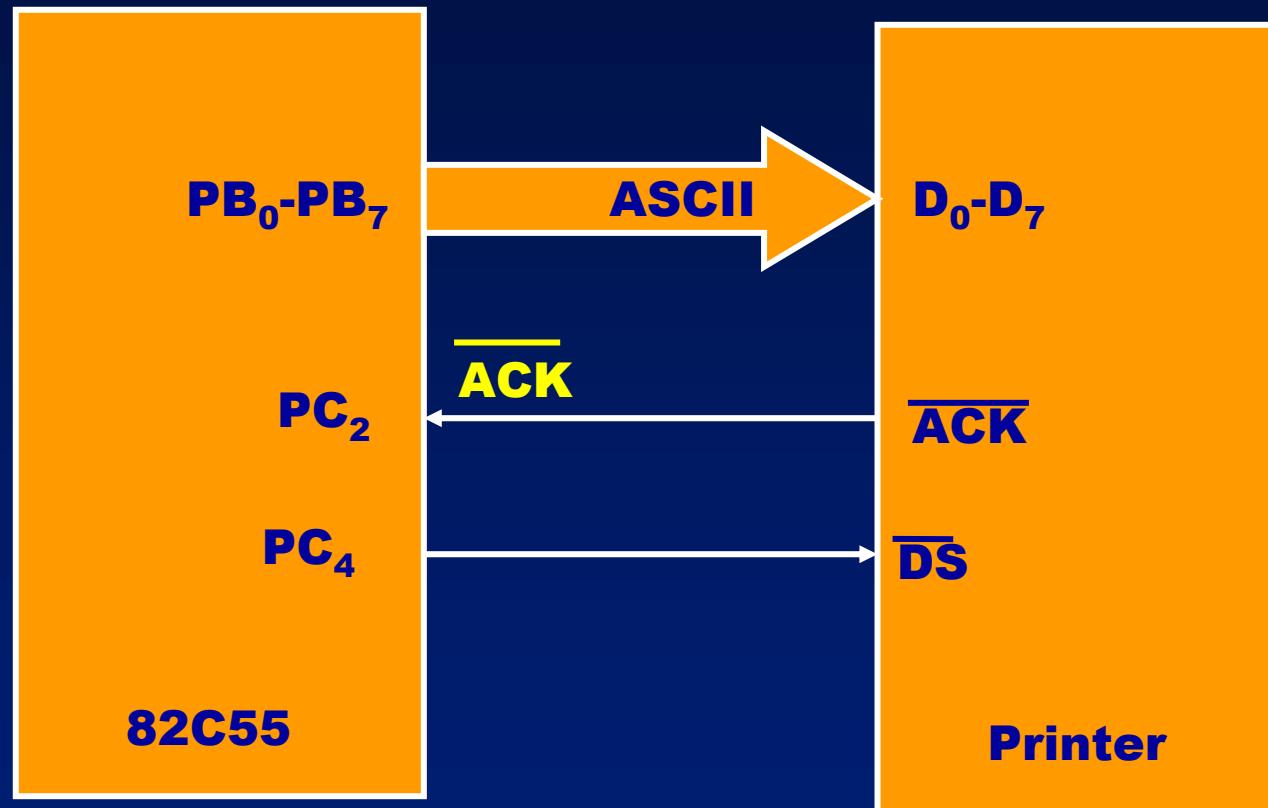
Timing Diagram



Mode 1 Port B



# Chế độ 1



DS: data strobe

# Chế độ 1

```
BIT1    EQU    2
PORTC   EQU    62H
PORTB   EQU    61H
CMD     EQU    63H
```

```
PRINT  PROC  NEAR
;check printer ready
    IN     AL,PORTC      ;get OBF
    TEST   AL, BIT1 ;test OBF
    JZ     PRINT         ;if OBF=0
```

```
;send character to printer
    MOV   AL, AH        ;get data
    OUT   PORTB, AL     ;print data
```

;send data strobe to printer

```
    MOV   AL, 8          ;clear DS
    OUT   CMD,AL
    MOV   AL, 9          ;set DS
    OUT   CMD, AL
    RET
PRINT ENDP
```

## Chế độ 2

- **Chỉ cho phép đổi 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

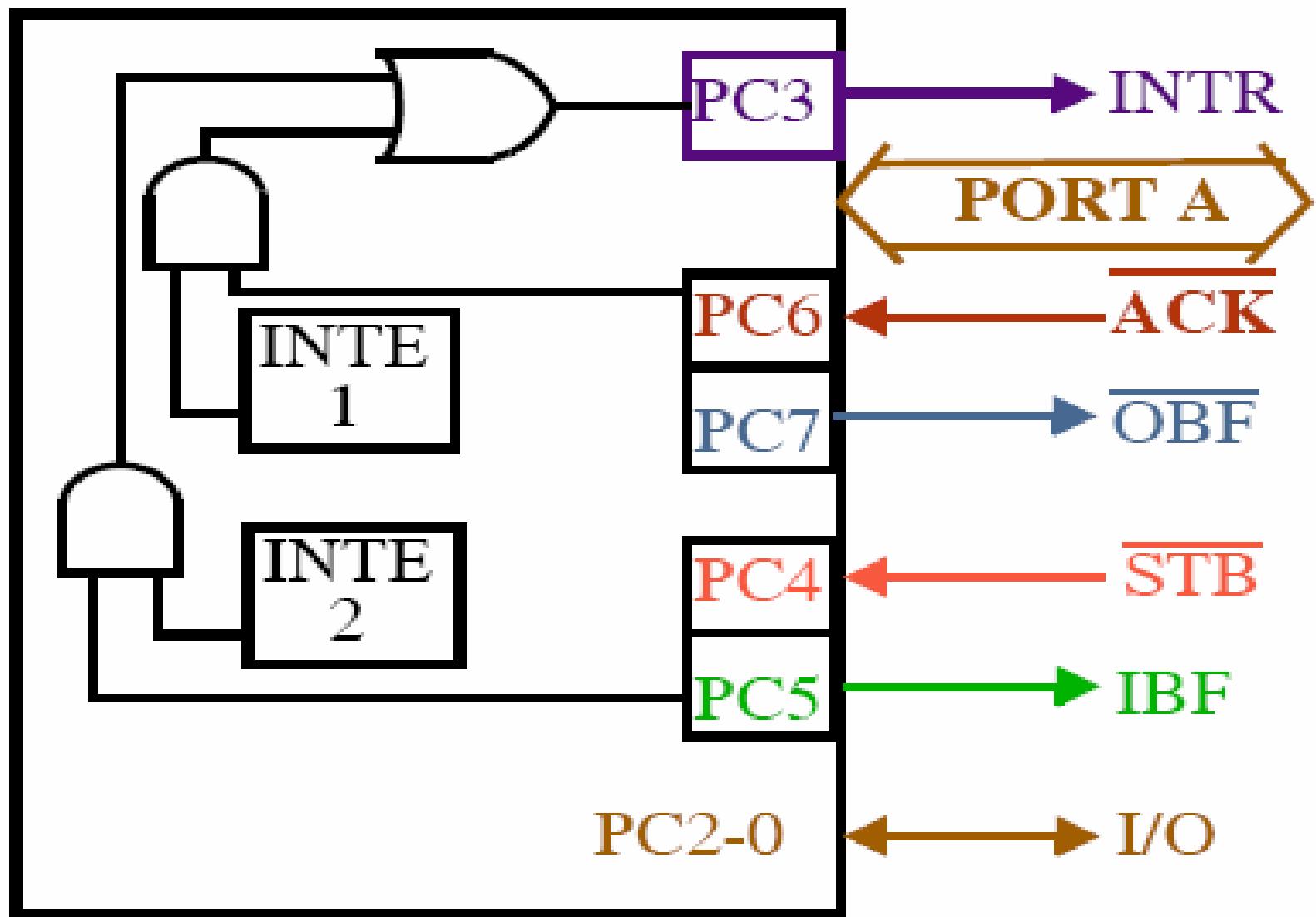
**INTE**

**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.

## Chế độ 2

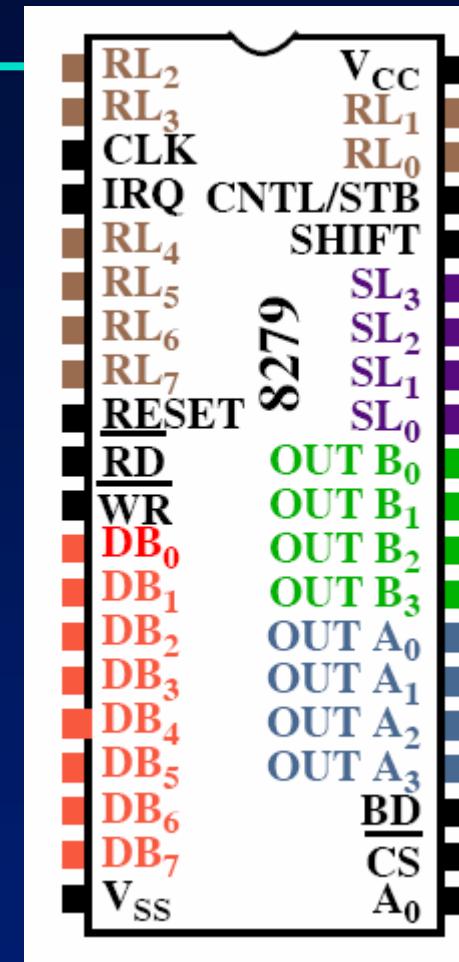


# Chương 4: Tổ chức vào ra dữ liệu

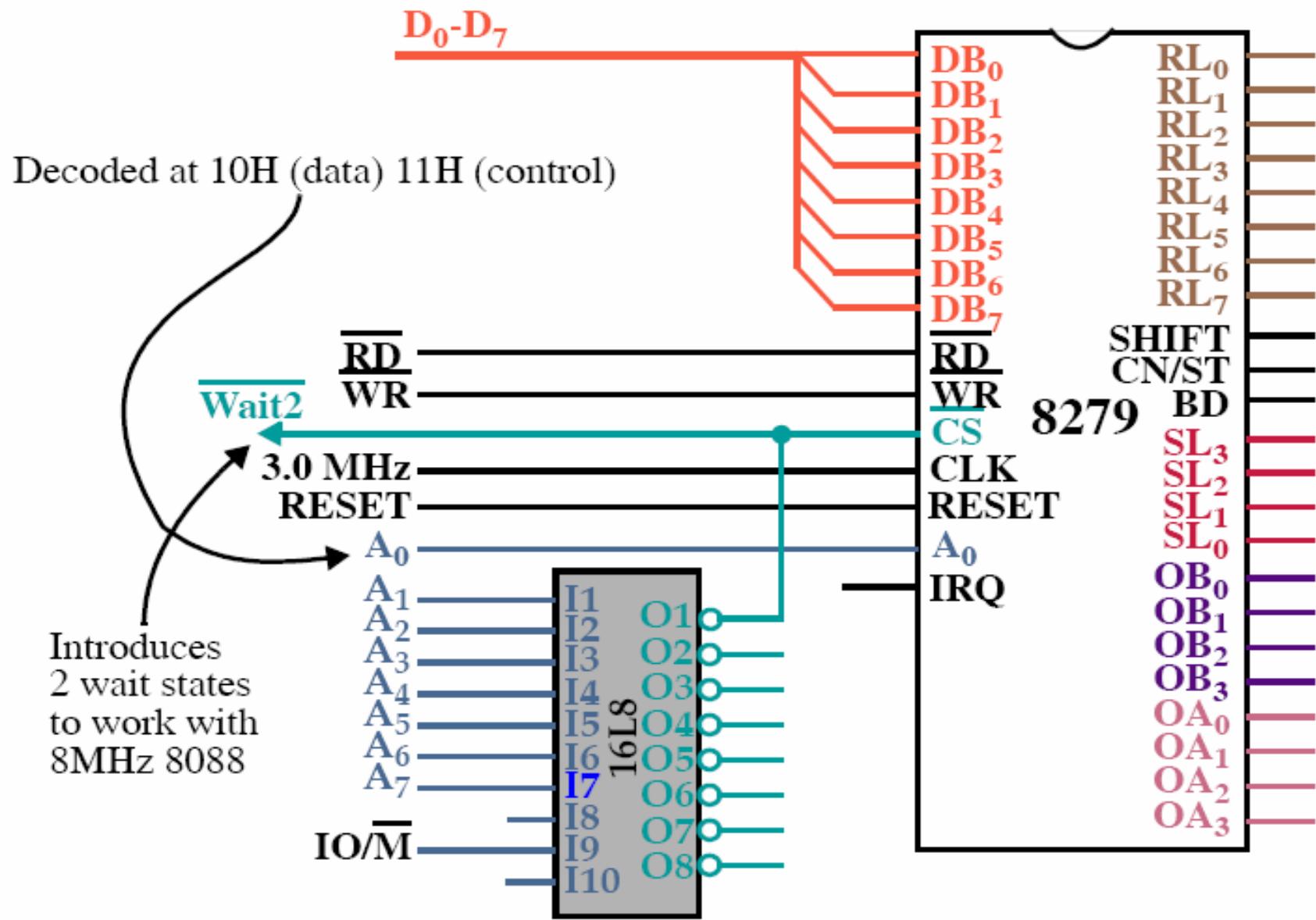
- Các tín hiệu của 8086 và các mạch phụ trợ 8284, 8288
- Ghép nối 8088 với bộ nhớ
- Ghép nối 8086 với bộ nhớ
- **Ghép nối với thiết bị ngoại vi**
  - Các kiểu ghép nối vào/ra
  - Giải mã địa chỉ cho các thiết bị vào/ra
  - Mạch ghép nối vào ra song song lập trình được 8255A
  - Mạch điều khiển bàn phím/màn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

# Mạch điều khiển 8279

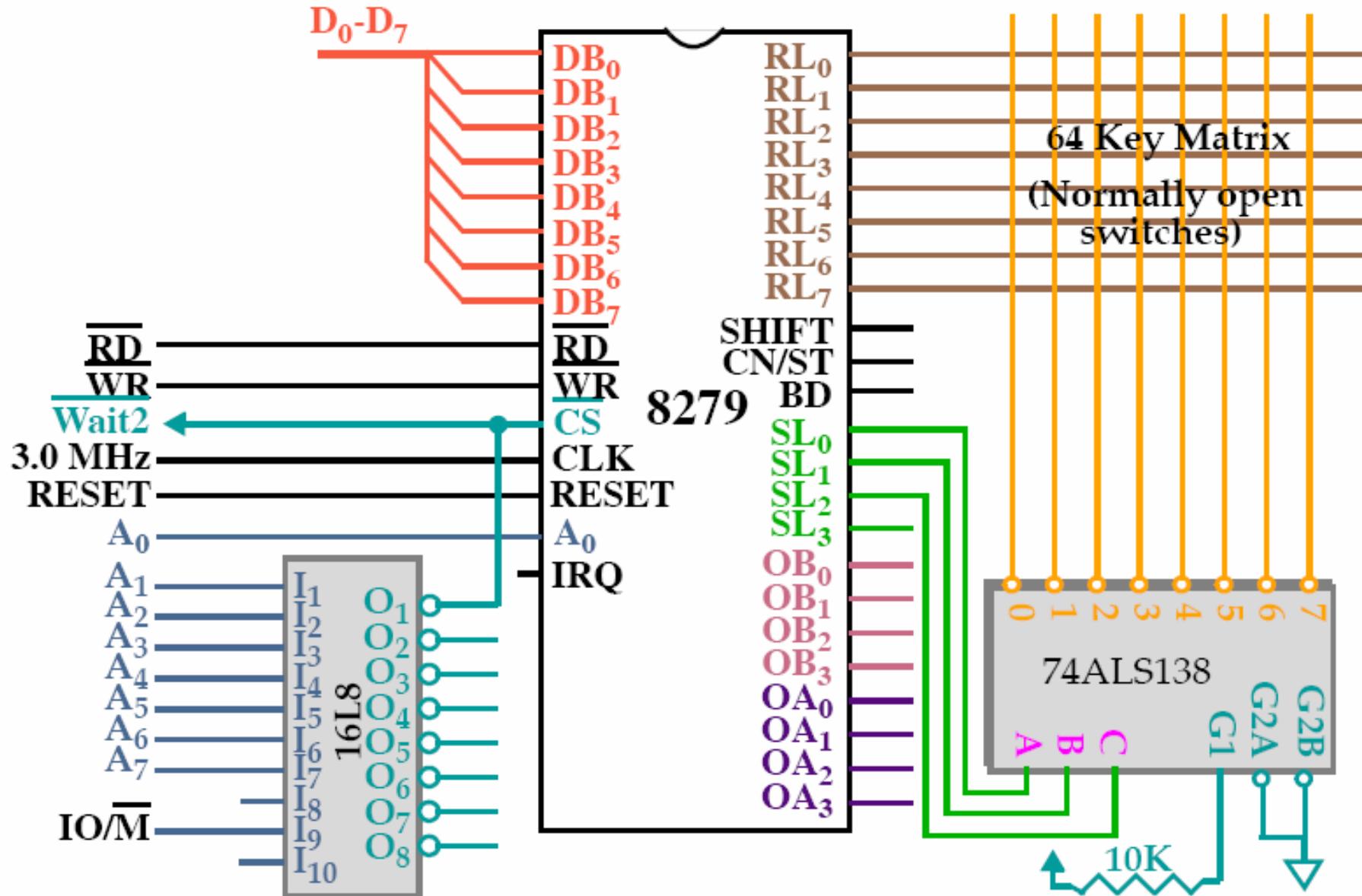
- **Điều khiển bàn phím và màn hiển thị 8279:**
  - **quét và mã hóa cho bàn phím** tới 64 phím
    - ⇒ bộ đệm FIFO có thể chứa 8 ký tự
  - **Điều khiển màn hiển thị** tới 16 số
    - ⇒ 16\*8 RAM để chứa thông tin về 16 số hiển thị
- **Các tín hiệu chính:**
  - **A<sub>0</sub>**: chọn giữa chế độ dữ liệu hoặc điều khiển
  - **RD**: xoá trắng 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
  - **DB<sub>7</sub>-DB<sub>0</sub>**: 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



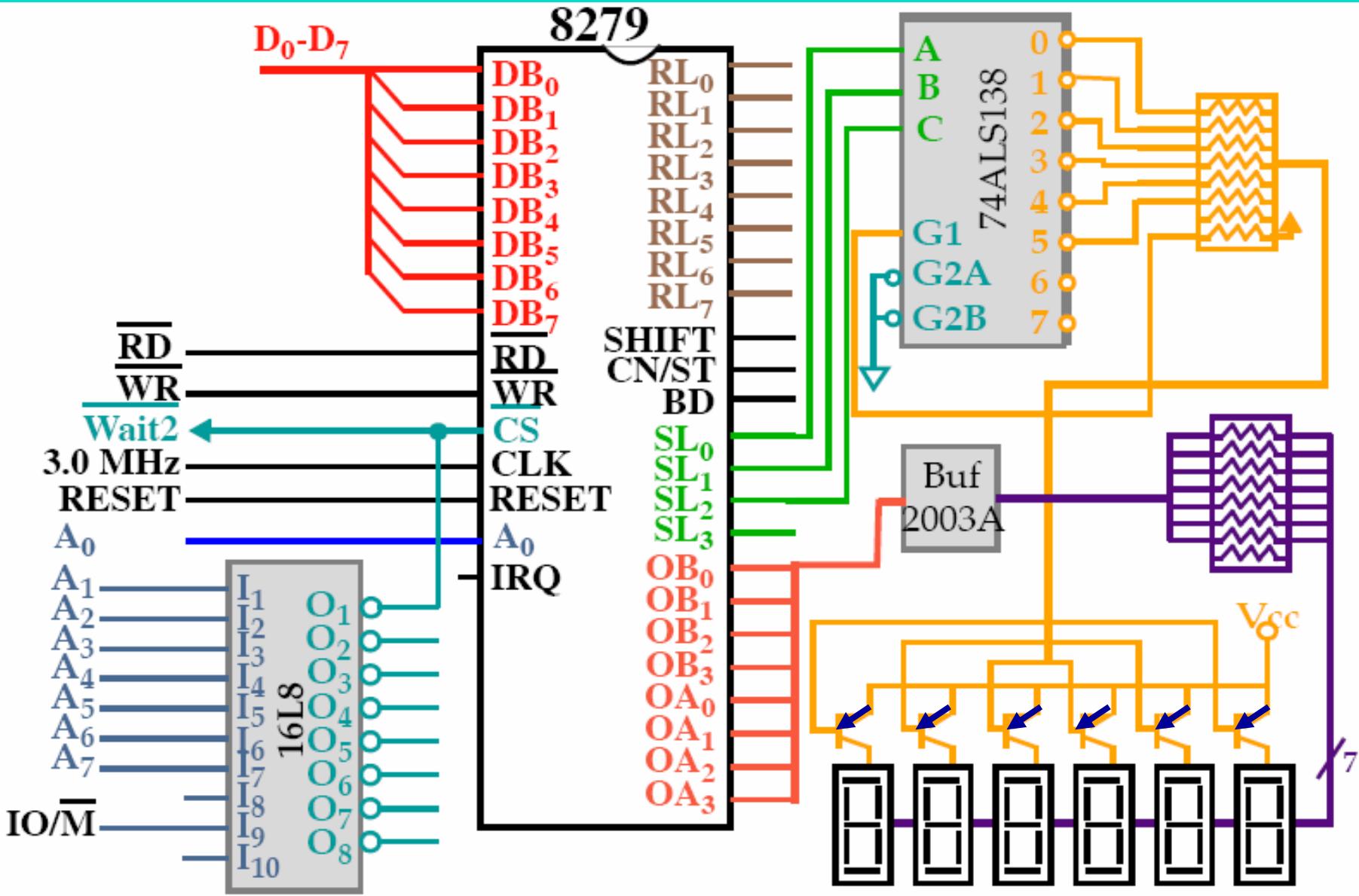
# Ghép nối 8279 với 8088



# Ghép nối 8279 với bàn phím



# Ghép nối 8279 với màn hiển thị



# Lập trình cho 8279

- Tùy điều khiển:  $D_7D_6D_5D_4D_3D_2D_1D_0$

<i>D<sub>7</sub></i>	<i>D<sub>6</sub></i>	<i>D<sub>5</sub></i>	<i>Function</i>	<i>Purpose</i>
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.

# Lập trình cho 8279

## ○ 000DD MMM

**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.

<i>DD</i>	<i>Function</i>
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

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



# Bộ định thời lập trình được 8254

Three independent 16-bit programmable counters (**timers**).

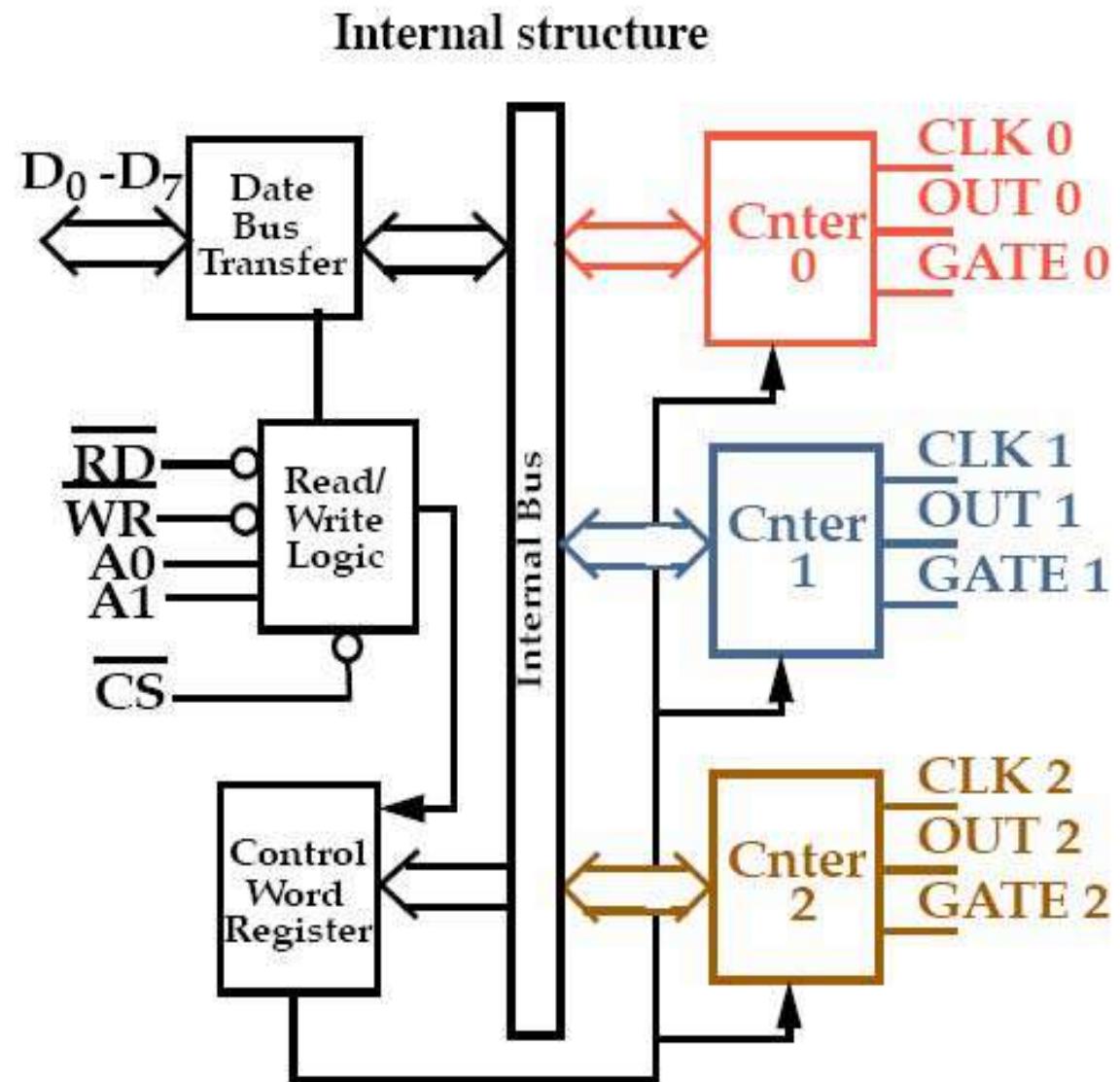
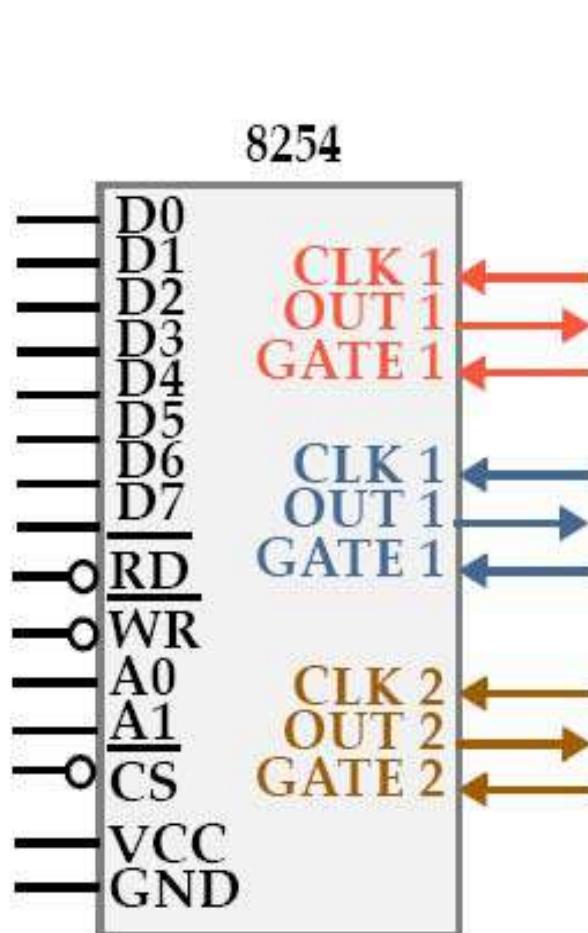
Each capable in of counting in binary or BCD with a maximum frequency of 10MHz.

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.

# Bộ định thời lập trình được 8254



# Bộ định thời lập trình được 8254

## 8254 Pin Definitions

**A<sub>1</sub>, A<sub>0</sub>**: 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.

**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.

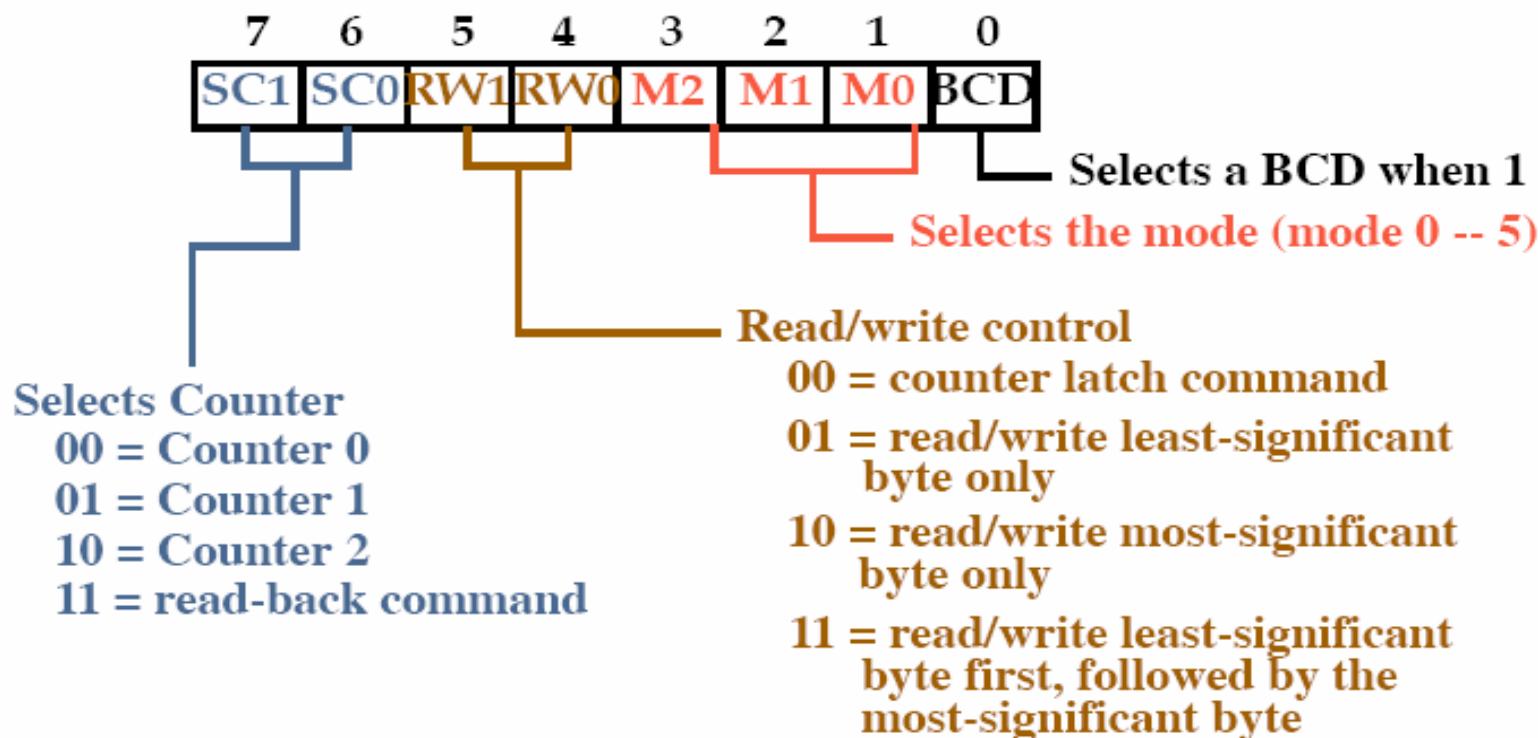
A <sub>1</sub>	A <sub>0</sub>	Function
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word

# 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).



# 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.

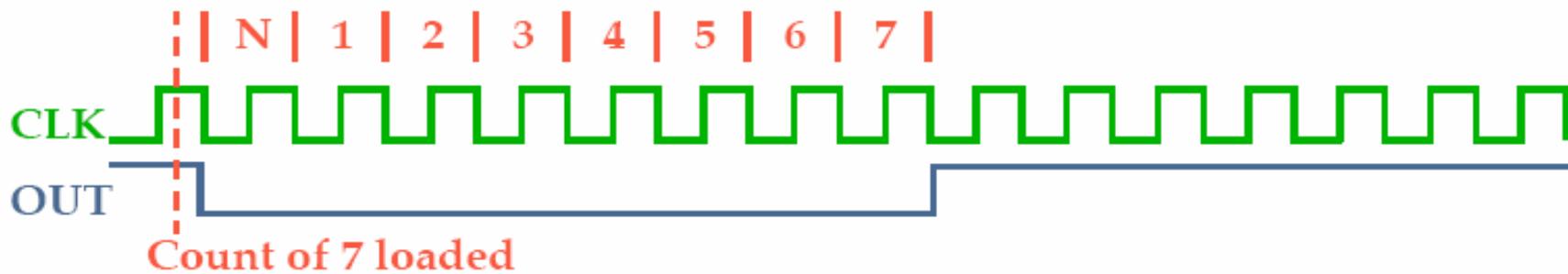
Each counter has a program control word used to select the way the counter operates.

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.



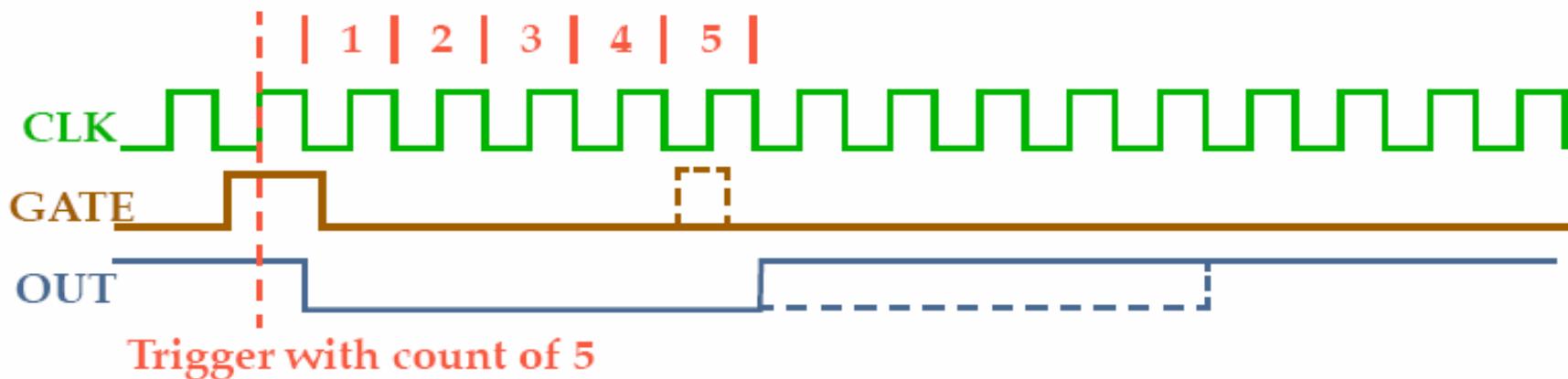
# Bộ định thời lập trình được 8254

## 8254 Modes

- Mode 1: One-shot mode.

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

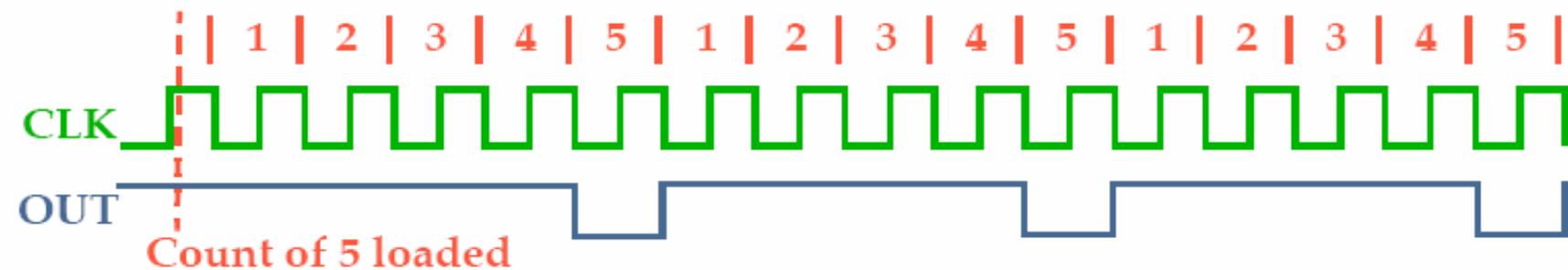
Counter reloaded if G is pulsed again.



- 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.

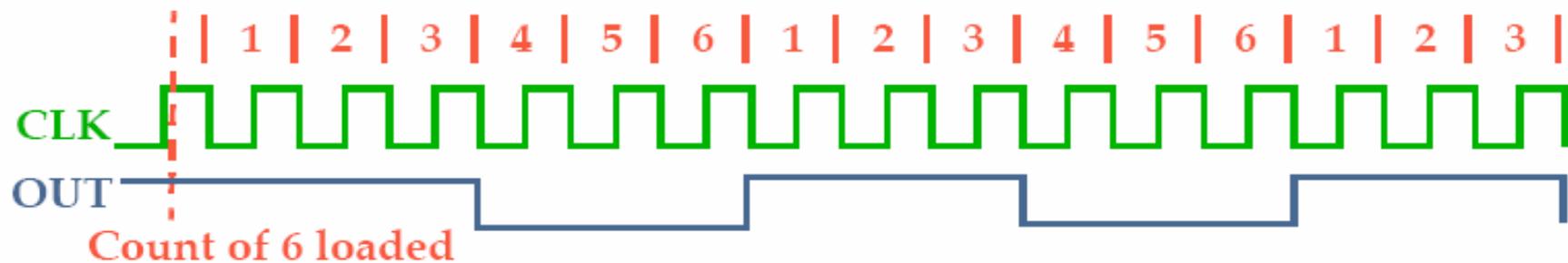


# Bộ định thời lập trình được 8254

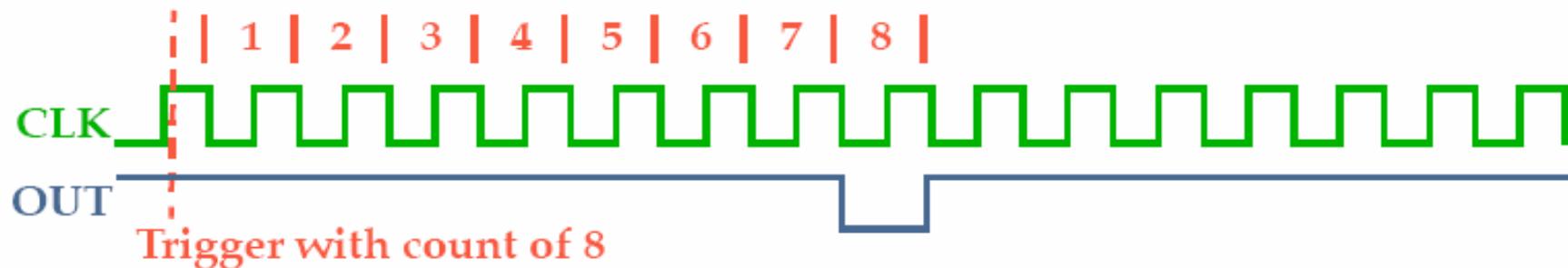
## 8254 Modes

- **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

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

# Giao tiếp truyền thông lập trình được

## 16550

*Programmable Communications Interface: 16550*

A universal asynchronous receiver/transmitter (UART).

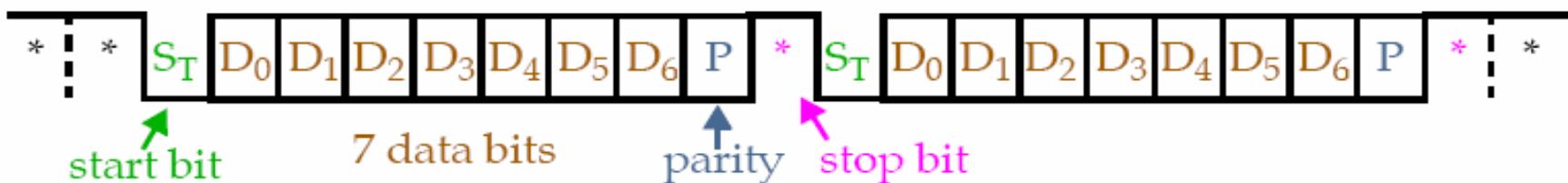
Operation speed: 0-1.5M Baud (Baud is # of bits transmitted/sec, including start, stop, data and parity).

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.

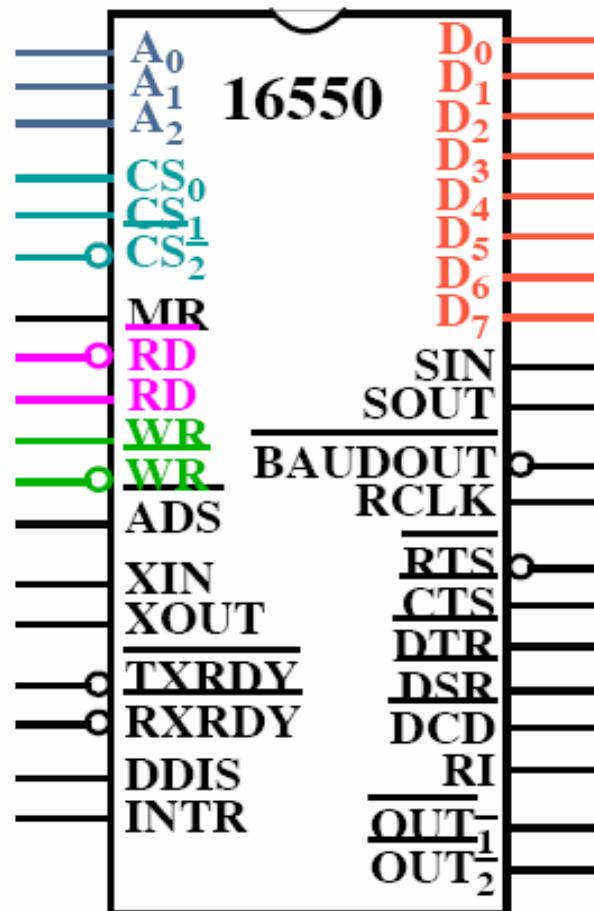


Two 10-bit frames of asynchronous data.

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

# Giao tiếp truyền thông lập trình được 16550

*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*.

# Giao tiếp truyền thông lập trình được

## 16550

### *Pinout of the 16550*

- A<sub>0</sub>, A<sub>1</sub> and A<sub>2</sub>: Select an internal register for programming and data transfer.

A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	Register
0	0	0	Receiver buffer (read) and transmitter holding (write)
0	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.
- CS<sub>0</sub>, CS<sub>1</sub>, CS<sub>2</sub>: 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).

# Giao tiếp truyền thông lập trình được

## 16550

### *Pinout of the 16550*

- **D<sub>7</sub>-D<sub>0</sub>**: 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.

# Giao tiếp truyền thông lập trình được 16550

## *Pinout of the 16550*

- RD,  $\overline{RD}$ : Read inputs (either can be used) -- cause data to be read from the register given by the address inputs.
- $\overline{RI}$ : Ring indicator input -- set to 0 by modem to indicate telephone is ringing.
- $\overline{RTS}$ : Request-to-send -- signal to modem, indicating UART wishes to send data.
- SIN, SOUT: Serial data pins, in and out.
- $\overline{RXRDY}$ : Receiver ready -- used to transfer received data via DMA techniques.
- $\overline{TXRDY}$ : Transmitter ready -- used to transfer transmitter data via DMA.
- WR,  $\overline{WR}$ : 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.

# Giao tiếp truyền thông lập trình được 16550

## Programming the 16550

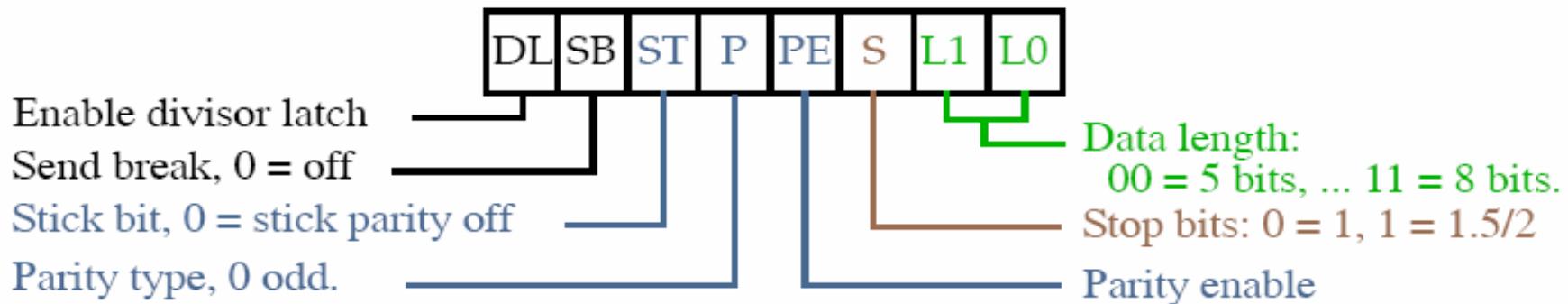
Two phases: Initialization, operation.

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.

# Giao tiếp truyền thông lập trình được

## 16550

### 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.

<i>ST</i>	<i>P</i>	<i>PE</i>	<i>Function</i>
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.

# Giao tiếp truyền thông lập trình được

## 16550

### 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.

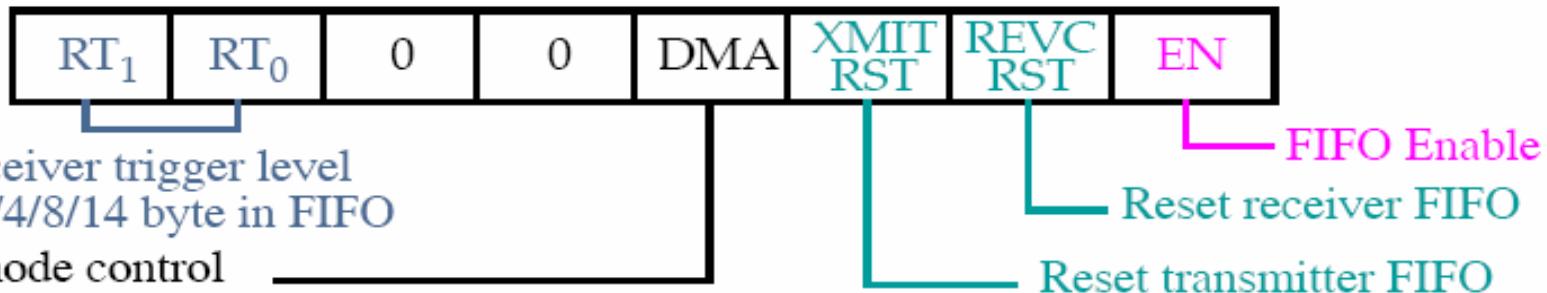
Port 000 used to hold least significant byte, 001 most significant.

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.



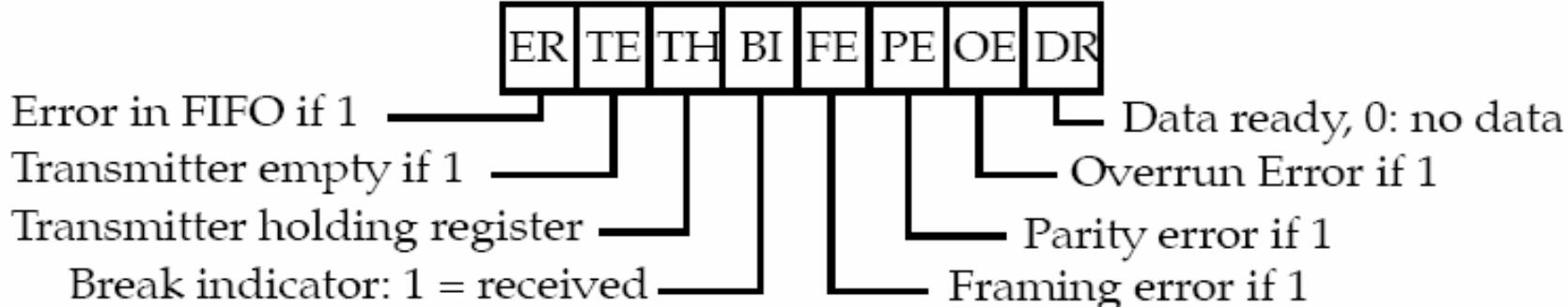
# Giao tiếp truyền thông lập trình được

## 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.

# 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.

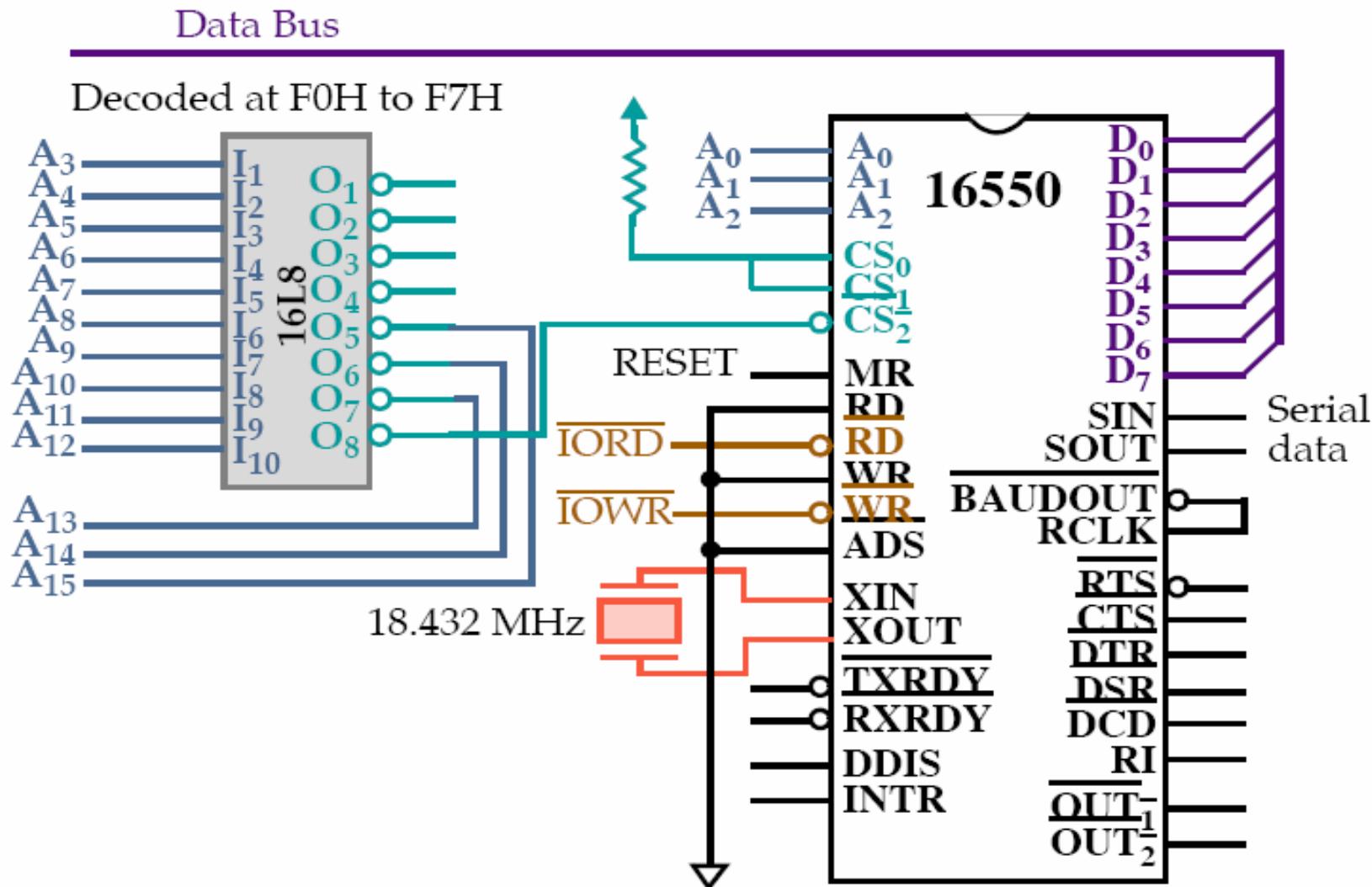
Software is failing to read the data from the FIFO.

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

# Giao tiếp truyền thông lập trình được

## 16550

*Example of 16550*



# Giao tiếp truyền thông lập trình được 16550

## 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.

# Giao tiếp truyền thông lập trình được

## 16550

### 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.

# Giao tiếp truyền thông lập trình được

## 16550

### 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

# 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
- $\overline{\text{BSY}}$ : When active, indicates the printer is busy and cannot accept more data
- $\overline{\text{PAP}}$ : High level shows that paper is about to run out
- $\overline{\text{OFON}}$ : 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.

# 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

# Chương 4: Tổ chức vào ra dữ liệu

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

# Bộ biến đổi số tương tự DAC

## Digital-to-Analog (DAC) Converters

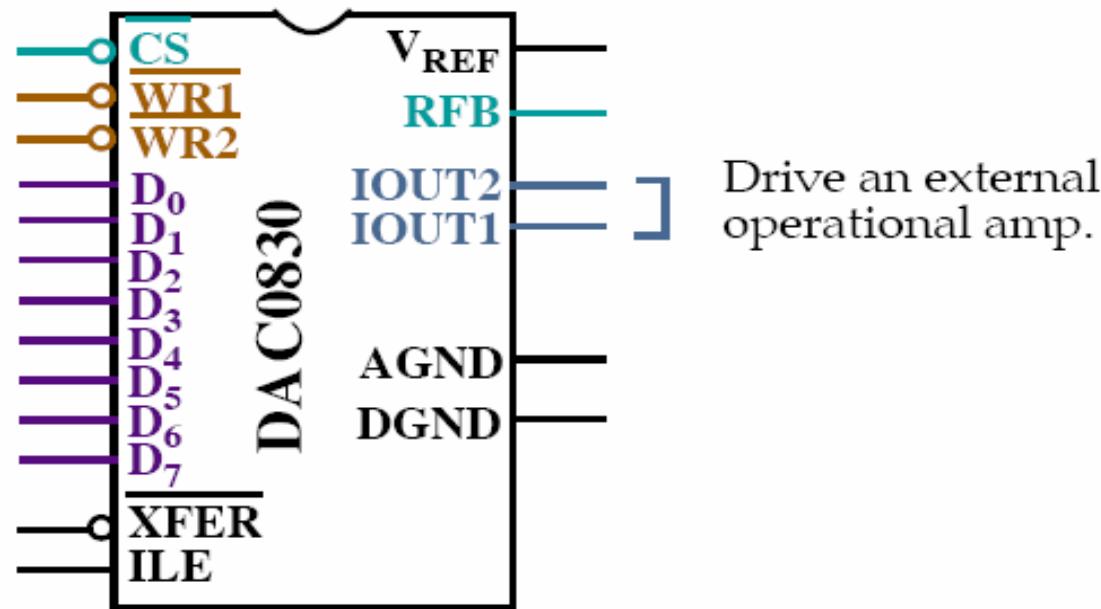
Used to convert between analog and digital data.

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.



# Bộ biến đổi số tương tự DAC

## Digital-to-Analog (DAC) Converters

8-bit digital value drives D<sub>0</sub> through D<sub>7</sub>.

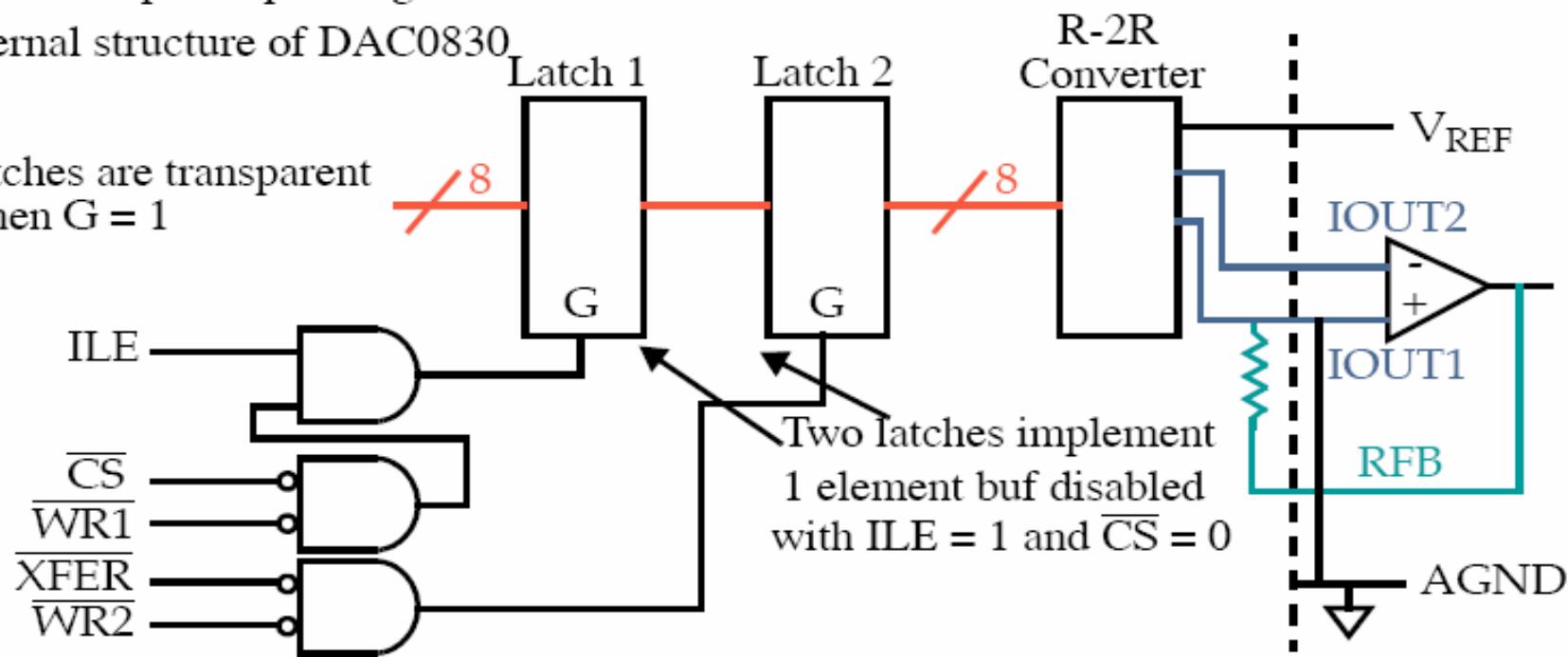
The outputs are IOUT1 and IOUT2.

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.

Internal structure of DAC0830

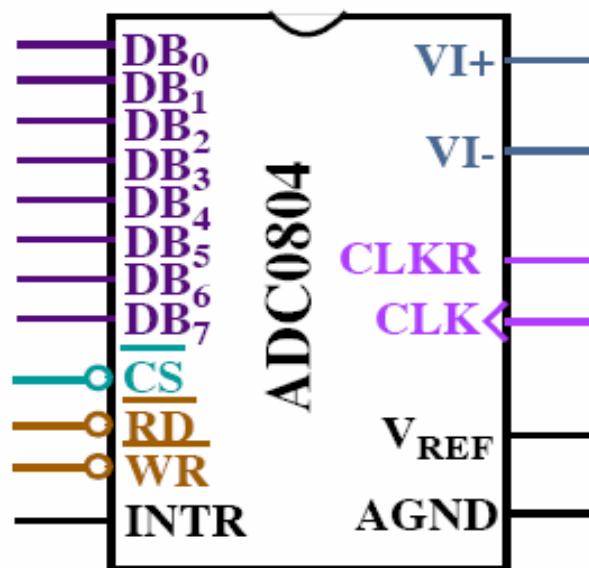
Latches are transparent  
when G = 1



# Bộ biến đổi tương tự số ADC

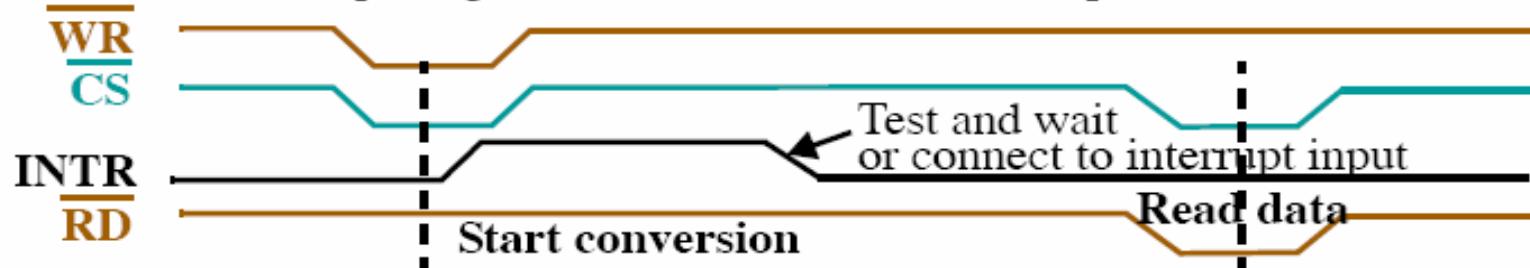
## Analog-to-Digital (ADC) Converters

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.



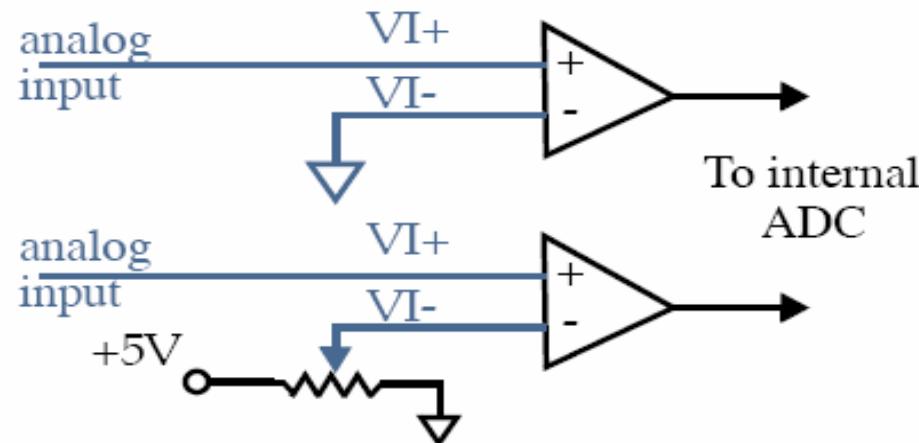
# Bộ biến đổi tương tự số ADC

## Analog-to-Digital (ADC) Converters

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

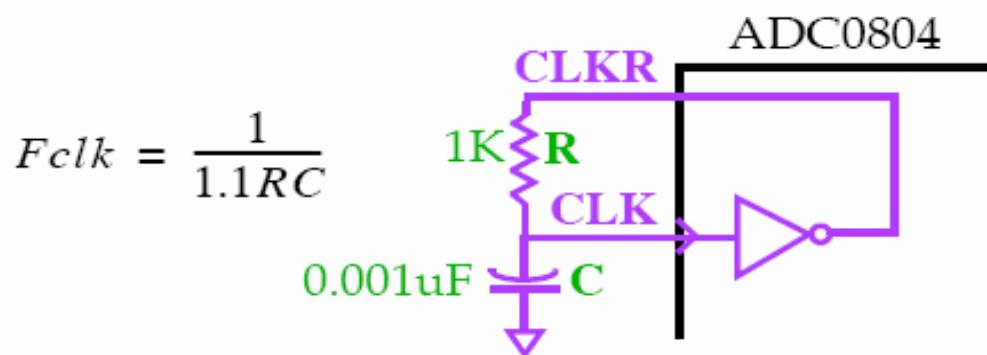
To sense a 0 to +5V input.

To sense an input offset from GND.



The ADC0804 requires a clock, generated either with:

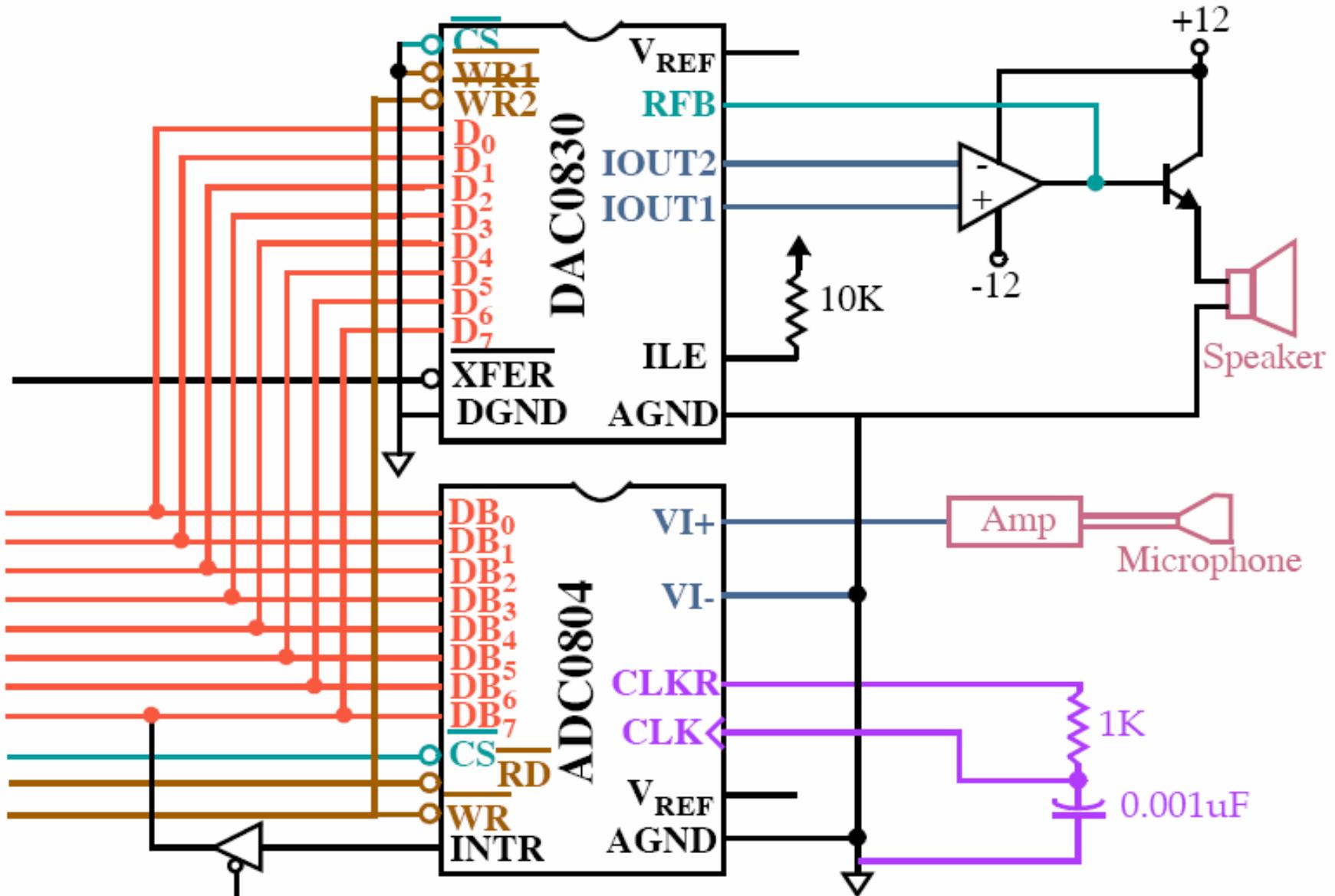
- An external clock applied to the CLK pin.
- Using an RC circuit.



Permissible clk frequencies  
are 100KHz to 1.46MHz.

Desirable to run at max.

# Bộ biến đổi tương tự số ADC





# Nội dung môn học

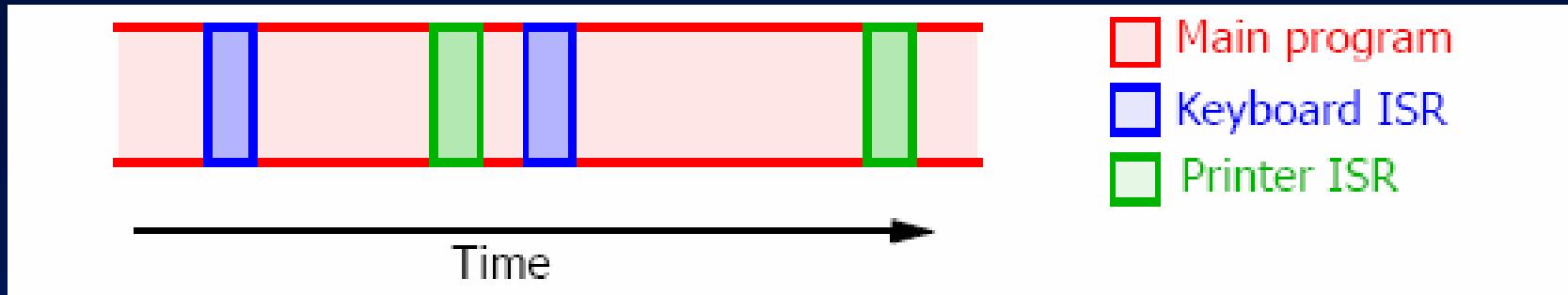
- 1. Giới thiệu chung về bộ vi xử lý**
- 2. Bộ vi xử lý Intel 8088/8086**
- 3. Lập trình hợp ngữ cho 8086**
- 4. Tổ chức vào ra dữ liệu**
- 5. Ngắt và xử lý ngắn**
- 6. Truy cập bộ nhớ trực tiếp DMA**
- 7. Các bộ vi xử lý trên thực tế**



# Chương 5: Ngắt và xử lý ngắt

- **Giới thiệu về ngắt**
- **Đáp ứng của CPU khi có yêu cầu ngắt**
- **Các thủ tục ngắt của người sử dụng**
- **Xử lý ưu tiên ngắt**
- **Mạch điều khiển ngắt ưu tiên 8259A**
- **Ngắt trong máy tính IBM PC**

# Giới thiệu về ngắt



- **2 loại ngắt:**

- **Ngắt cứng: tín hiệu yêu cầu ngắt tự**

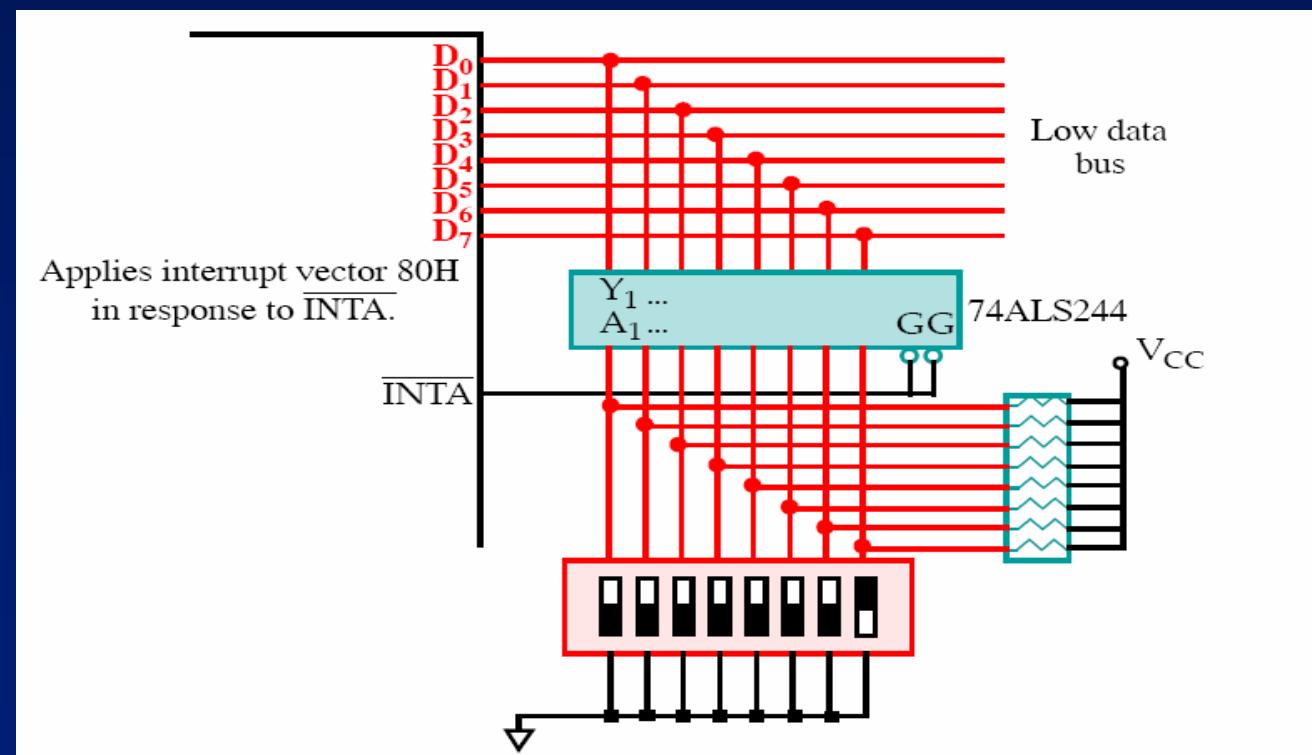
- ⇒ **NMI (ngắt không che được)**

- ✓ **Lỗi chấn lě và các lỗi hệ thống nghiêm trọng khác (ví dụ: mất nguồn)**

- ⇒ **và INTR (ngắt che được)**

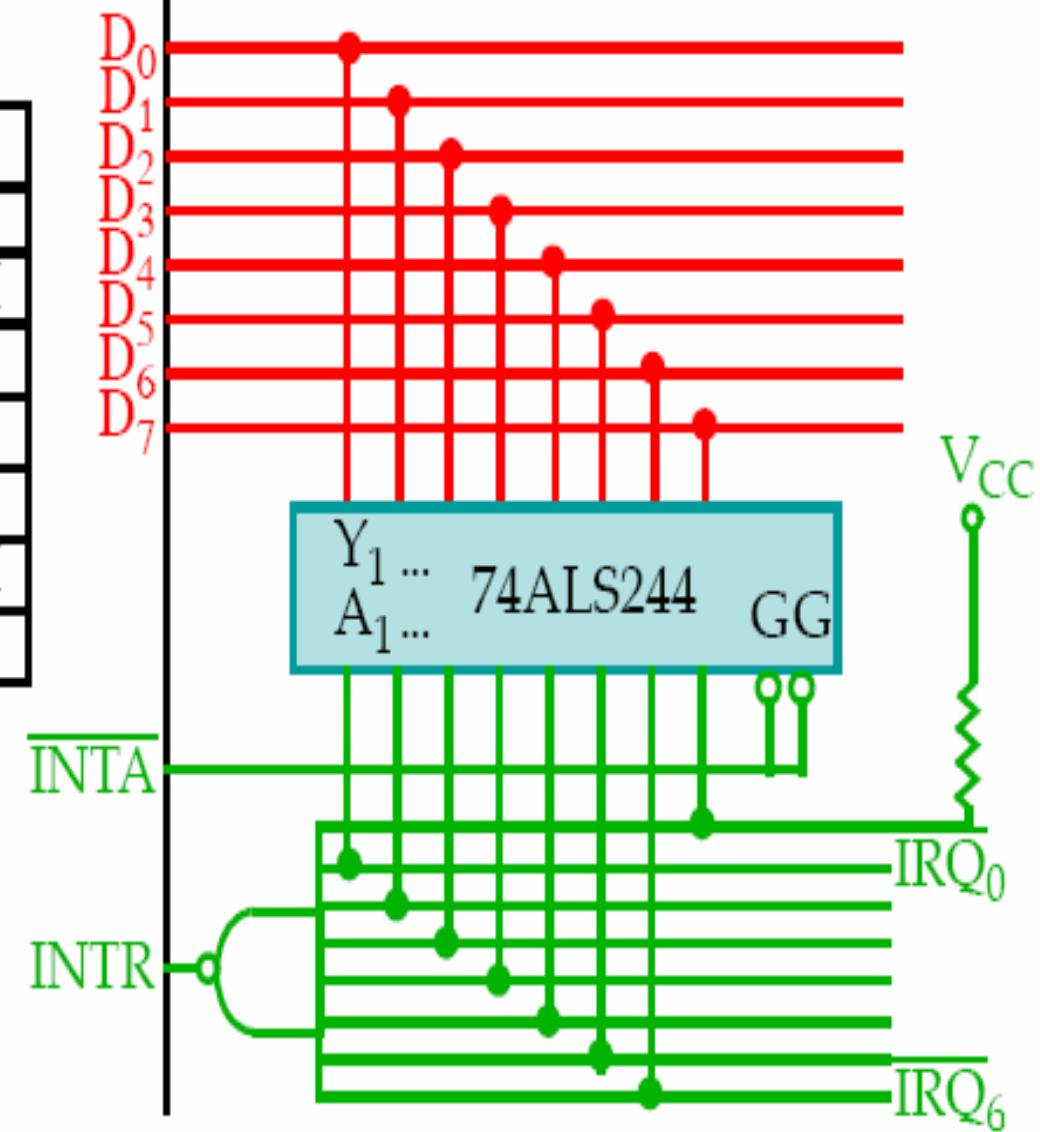
- **Ngắt mềm: CPU thực hiện các lệnh ngắt INT N, 0=< N <=255**

# Giới thiệu về ngắt



# Giới thiệu về ngắt

IRQs								
6	5	4	3	2	1	0	Vect	
1	1	1	1	1	1	0	FEH	
1	1	1	1	1	0	1	FDH	
1	1	1	1	0	1	1	FBH	
1	1	1	0	1	1	1	F7H	
1	1	0	1	1	1	1	EFH	
1	0	1	1	1	1	1	DFH	
0	1	1	1	1	1	1	BFH	





# Đáp ứng của CPU khi có yêu cầu ngắn

- **Bảng vector ngắn: 1 Kbytes 00000H đến 003FF H**
  - 256 vector ngắn
  - 1 vector 4 bytes, chứa IP và CS của CTCPVN
  - 32 vector đầu dành riêng cho Intel
  - 224 vector sau dành cho người dùng

# Đáp ứng của CPU khi có yêu cầu ngắt

Chương trình chính

CTCPVN

CPU:

- Cắt thanh ghi cờ F
- Xoá IF và TF
- Cắt CS và IP
- Lấy địa chỉ CTCPVN

lệnh cắt các  
thanh ghi

CPU:

- Lấy lại IP và CS
- Lấy lại thanh ghi cờ F

lệnh lấy các  
thanh ghi

IRET



# Các thủ tục ngắn của người sử dụng

- **Thiết lập vector ngắn:**

- **Cắt vector ngắn hiện tại:**

- ⇒ **Dùng hàm 35H của ngắn 21H của DOS**

- ✓ **Vào: AH=35h, AL= số hiệu ngắn**

- ✓ **Ra: ES:BX = địa chỉ đoạn : địa chỉ offset của CTCPVN**

- ⇒ **Cắt ES và BX vào thanh ghi hoặc ô nhớ**

- **Đưa vector của thủ tục ngắn của người sử dụng vào bảng vector ngắn:**

- ⇒ **Dùng hàm 25H của ngắn 21H**

- ✓ **Vào: AH=25H, AL= số hiệu ngắn, DS:DX= địa chỉ đoạn: địa chỉ offset của CTCPVN của người sử dụng**

- **Khôi phục lại vector cũ trước khi kết thúc CTCPVN của người sử dụng**



# Các thủ tục ngắn của người sử dụng

.Model Small

.Stack 100

.Data

OLD_IP	DW	?
OLD_CS	DW	?

.Code

Main Proc

;Lấy vector cũ của ngắn 40H

```
MOV AH, 35H  
MOV AL, 40H  
INT 21H  
MOV OLD_IP, BX  
MOV OLD_CS, ES
```

;Thiết lập vector ngắn 40H mới

```
MOV DX, offset New40  
MOV AX, CS  
PUSH DS  
MOV DS, AX  
MOV AH, 25H  
INT 21H  
POP DS
```

Main Endp

New40 Proc

;các lệnh của CTCPVN

New40 Endp

End Main



## Xử lý ưu tiên ngắt

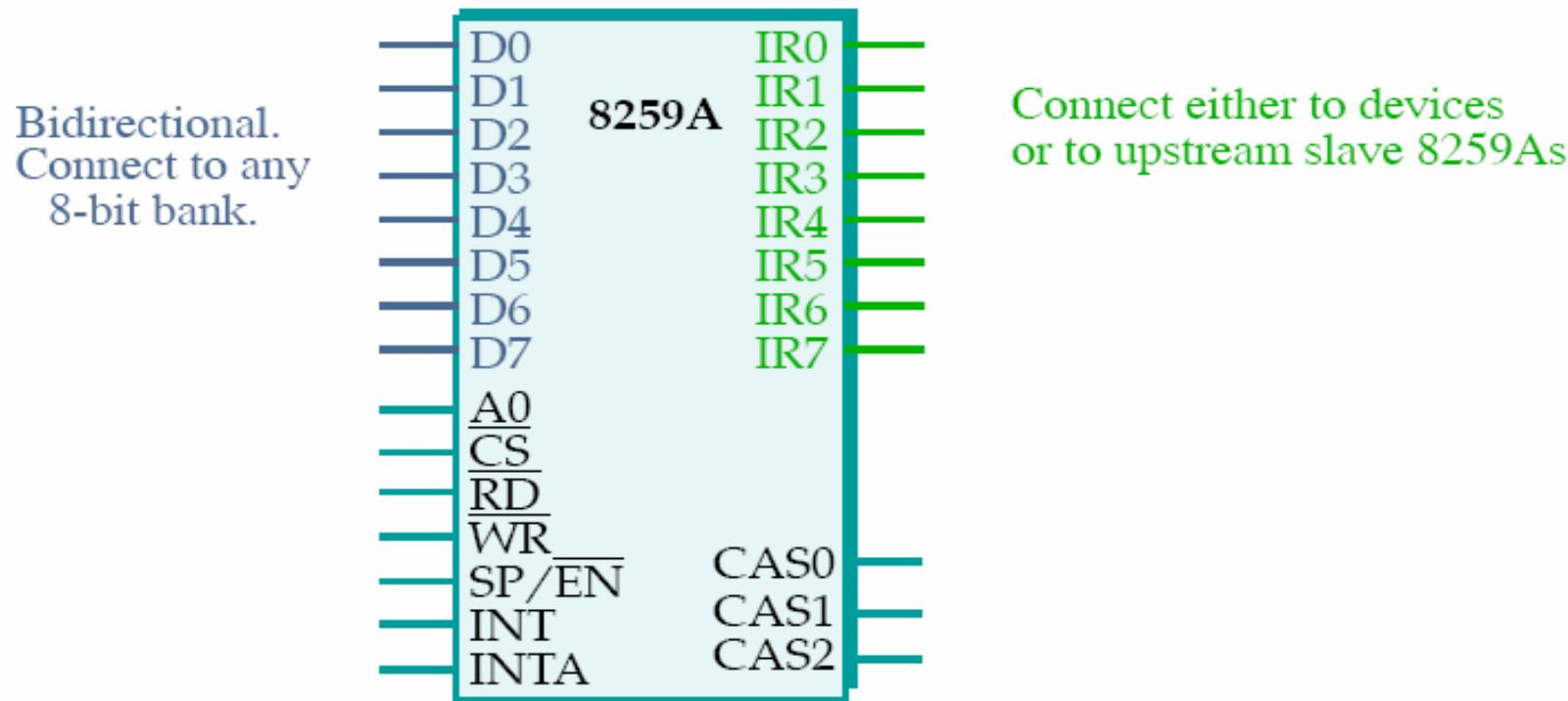
- **Ngắt có mức ưu tiên cao nhất sẽ được phục vụ trước**
- **Các mức ưu tiên:**
  - Ngắt nội bộ: INT 0, INT 1
  - Ngắt không che được: NMI
  - Ngắt che được INTR
  - Ngắt mềm INT N
- **CPU sẽ xử lý thế nào nếu CPU đang thực hiện phép chia và số chia bằng 0 đồng thời có yêu cầu ngắt từ chân INTR?**

# Mạch điều khiển ngắt 8259A

## 8259A Programmable Interrupt Controller

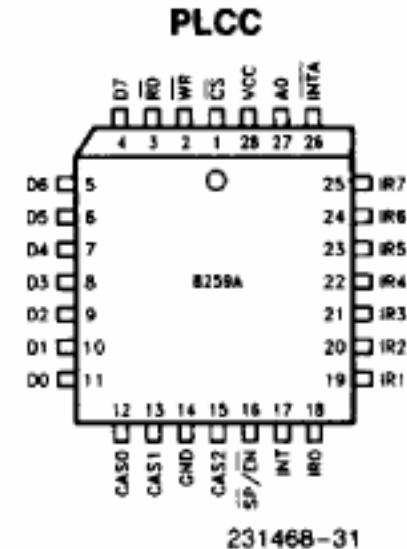
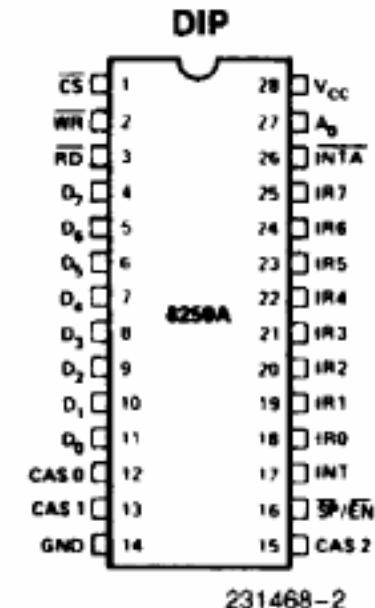
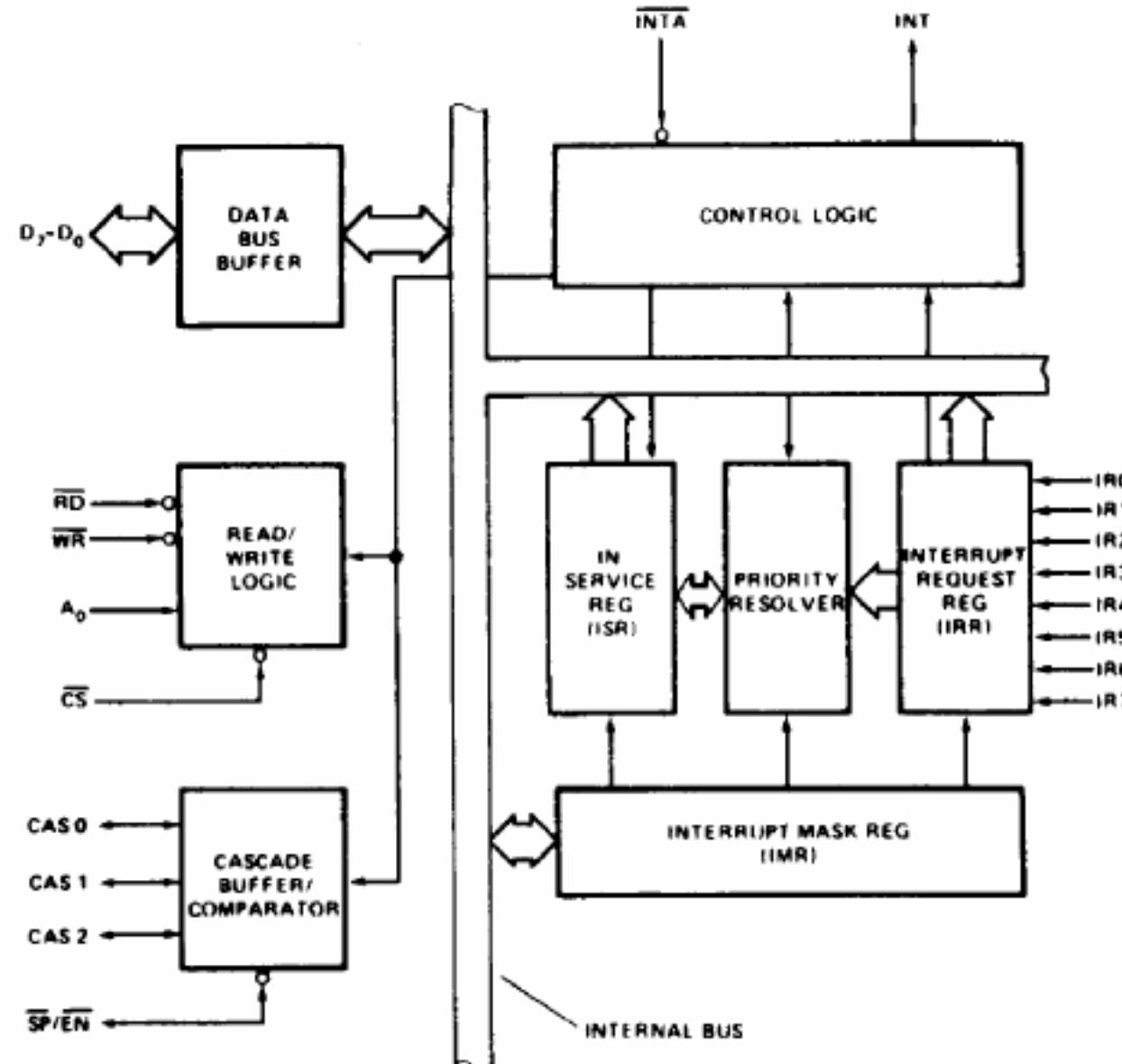
The 8259A adds 8 vectored priority encoded interrupts to the microprocessor.

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.

# Mạch điều khiển ngắt 8259A





# Mạch điều khiển ngắt 8259A

WR

Connects to a write strobe signal (one of 8 for the Pentium).

RD

Connects to the  $\overline{IORC}$  signal.

INT

Connects to the INTR pin on the microprocessor.

INTA

Connects to the  $\overline{INTA}$  pin on the microprocessor.

A0

Selects different command words in the 8259A.

CS

Chip select - enables the 8259A for programming and control.

SP/ $\overline{EN}$

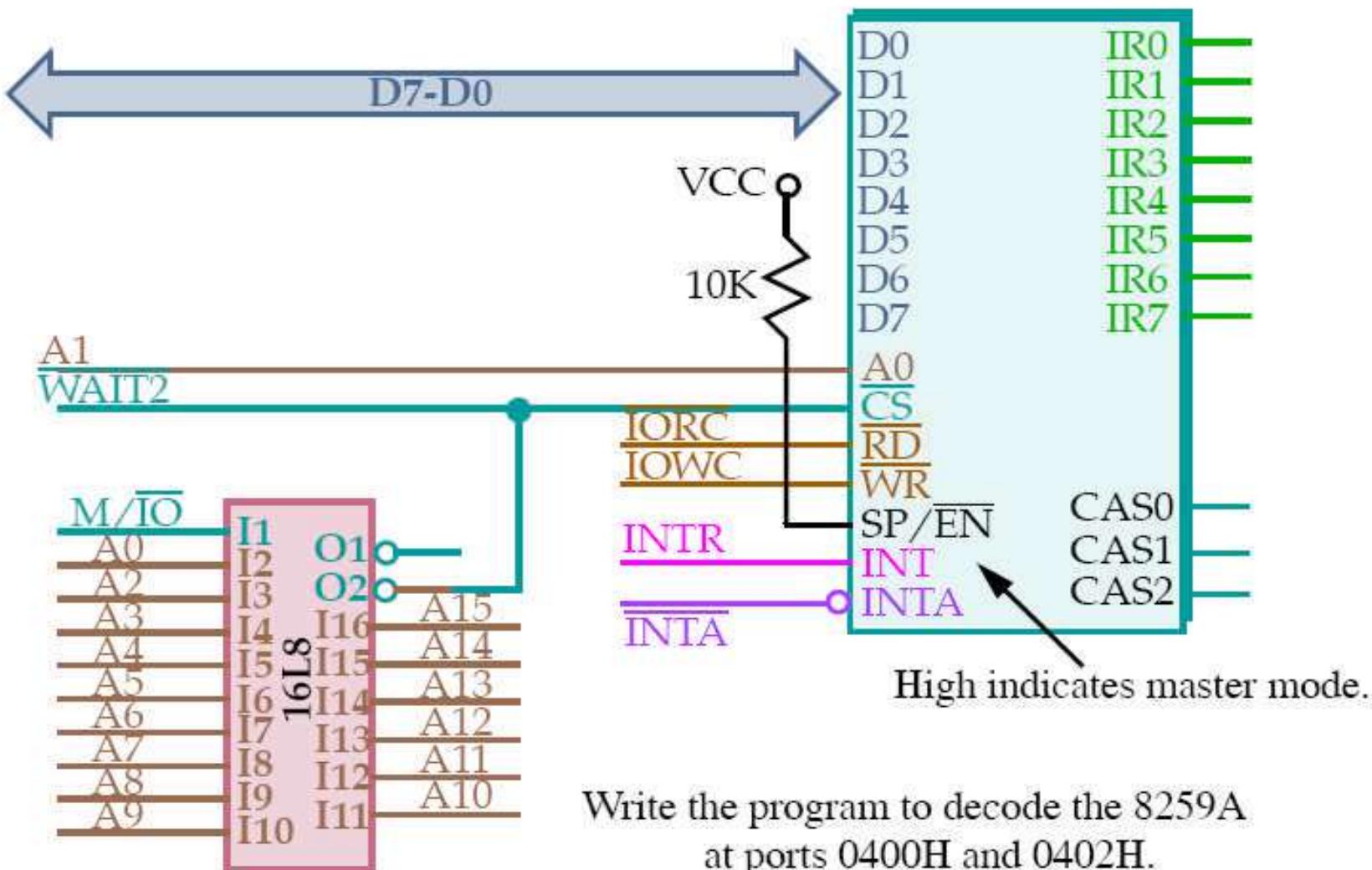
Slave Program (1 for master, 0 for slave)/Enable Buffer (controls the data bus transivers when in buffered mode).

CAS2-CAS0

Used as outputs from the master to the slaves in cascaded systems.

# Mạch điều khiển ngắt 8259A

A single 8259A connected in the 8086.



# Mạch điều khiển ngắt 8259A

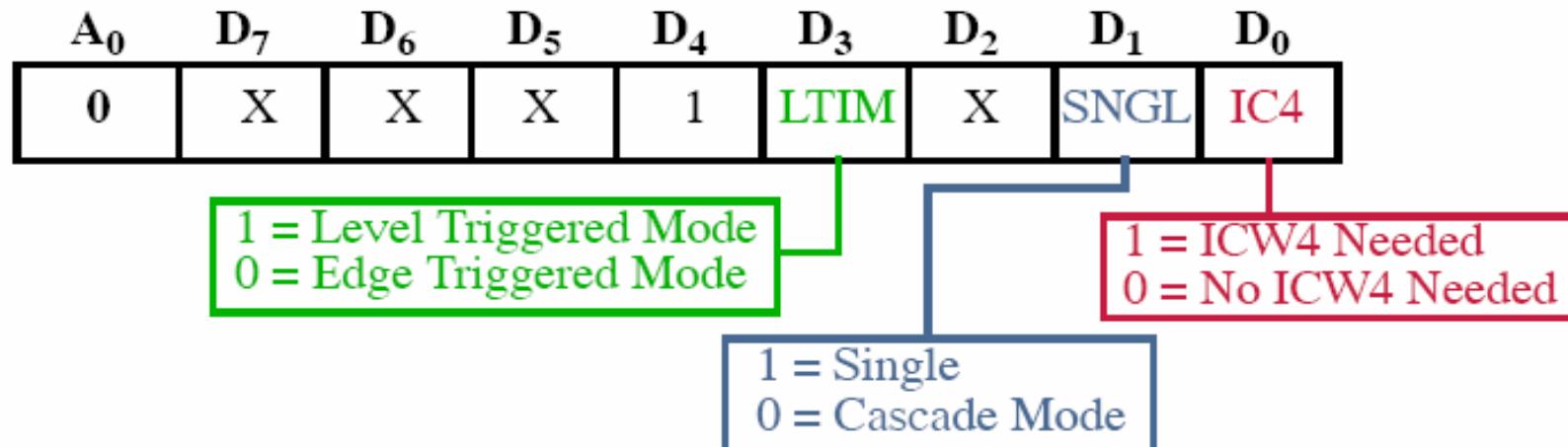
Programmed by *Initialization (ICWs)* and *Operation (OCWs)* Command Words.

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:

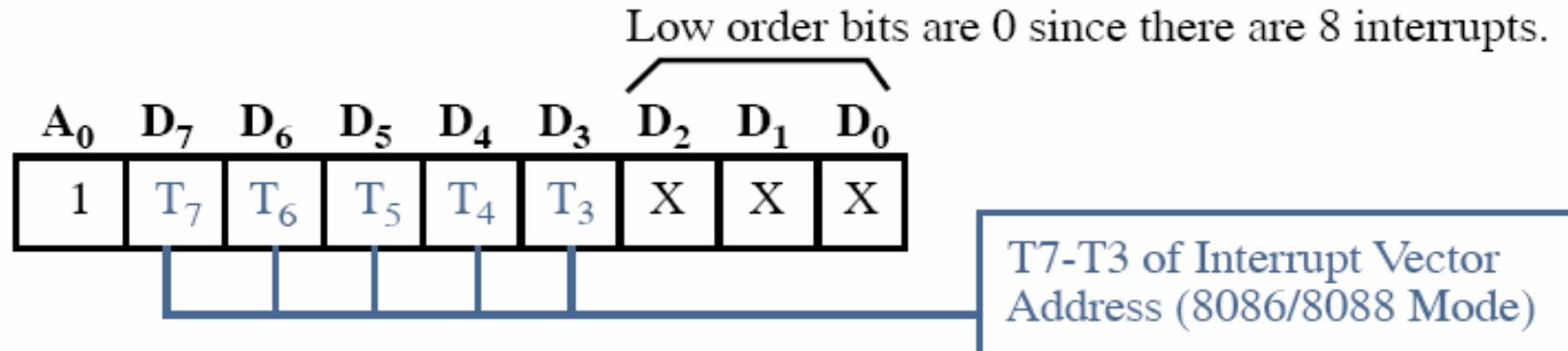


LTIM indicates if IRQ lines are positive edge-triggered or level-triggered.



# Mạch điều khiển ngắt 8259A

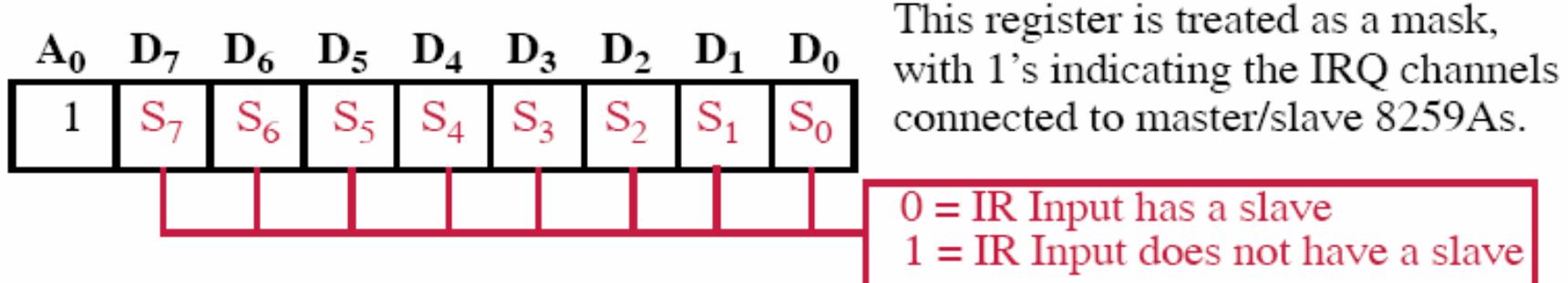
## ○ ICW2:



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

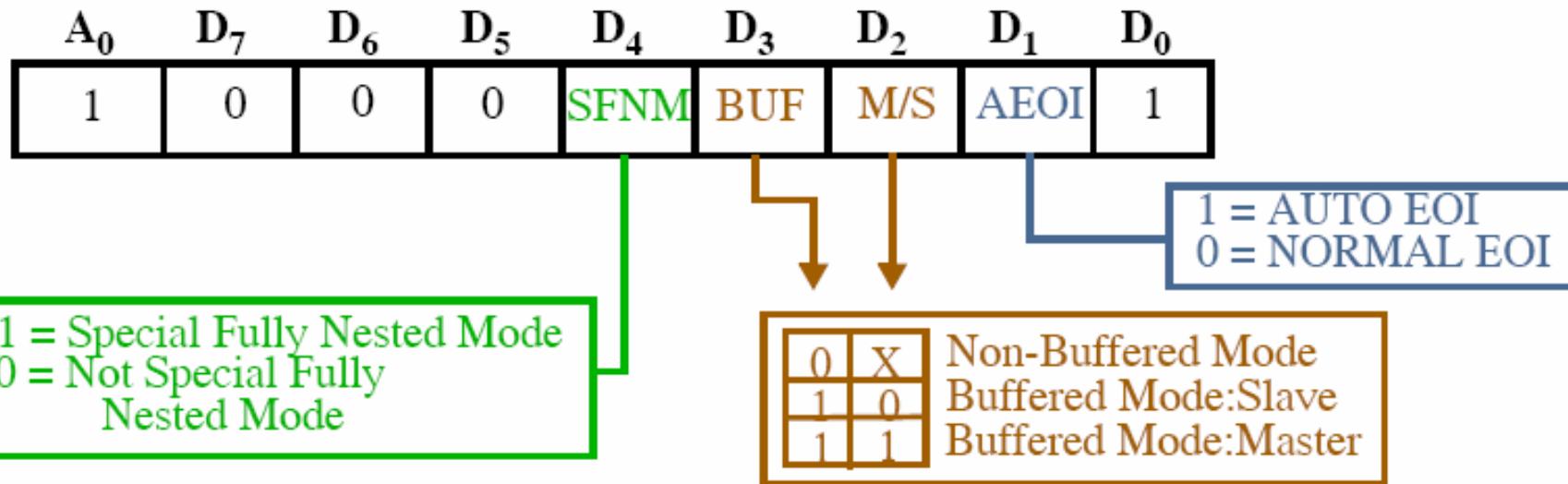
For example, if programmed to generate vectors 08H-0FH, 08H is placed into these bit positions.

## ○ ICW3:



# 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.

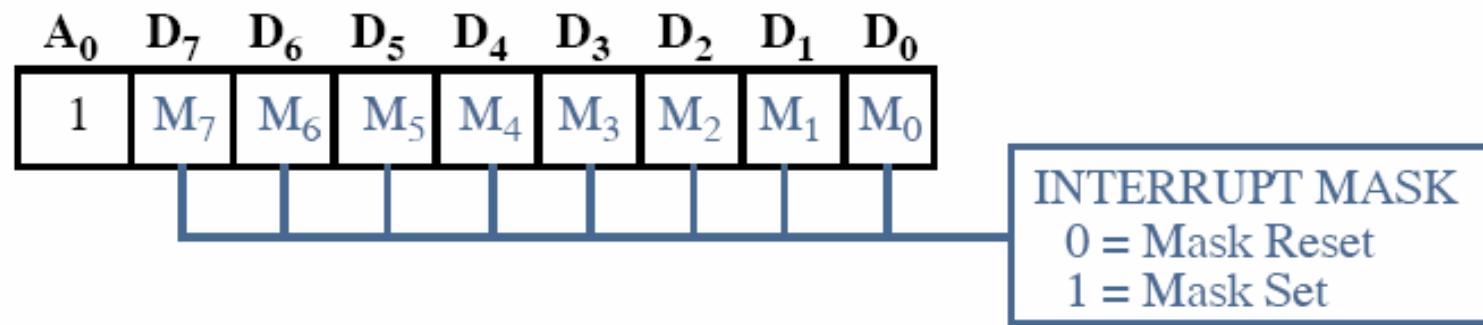
AEOI, if 1, indicates that an interrupt automatically resets the interrupt request bit, otherwise OCW2 is consulted for EOI processing.



# Mạch điều khiển ngắt 8259A

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

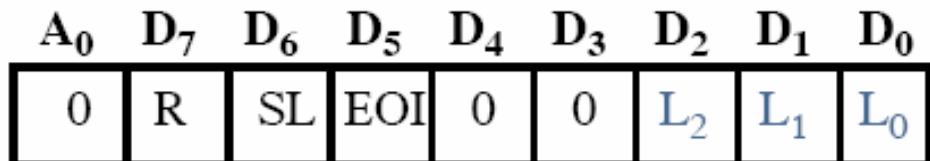
- OCW1:



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:

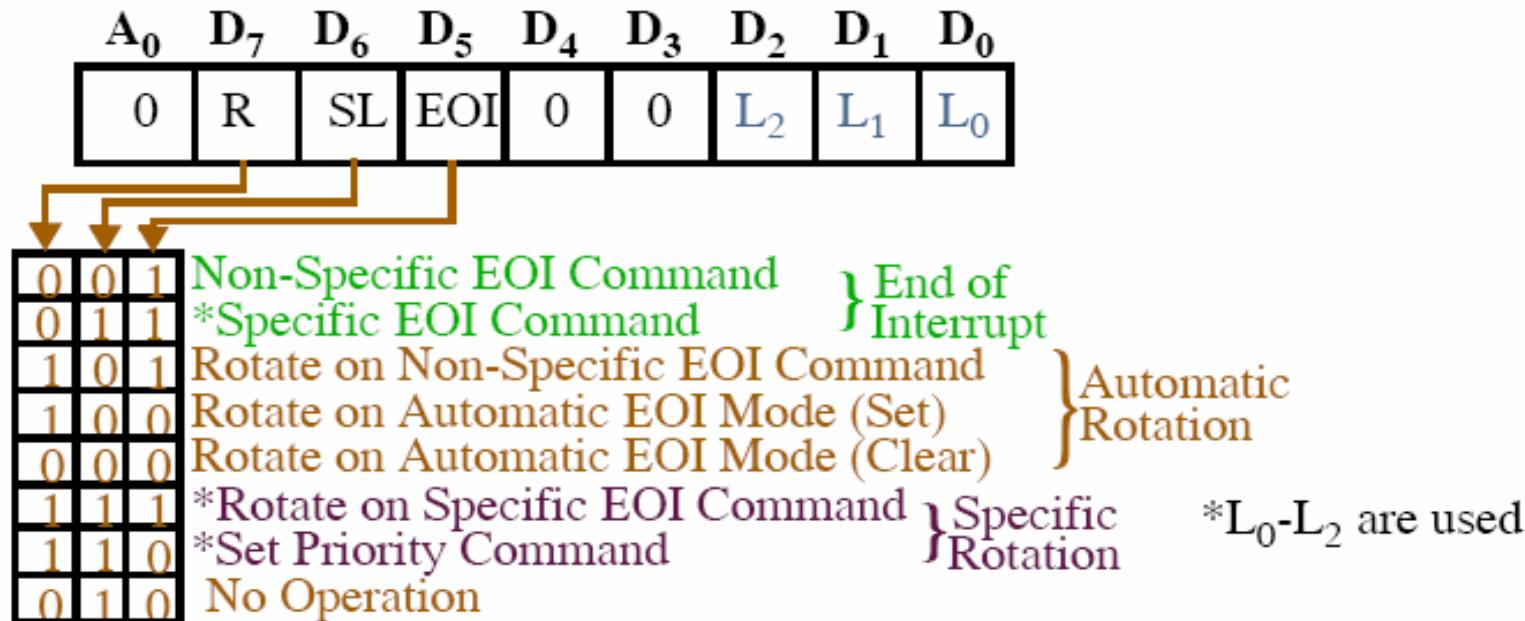


Only programmed when the AEOI mode in ICW4 is 0.

Allows you to control priorities after each interrupt is processed.

# 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.

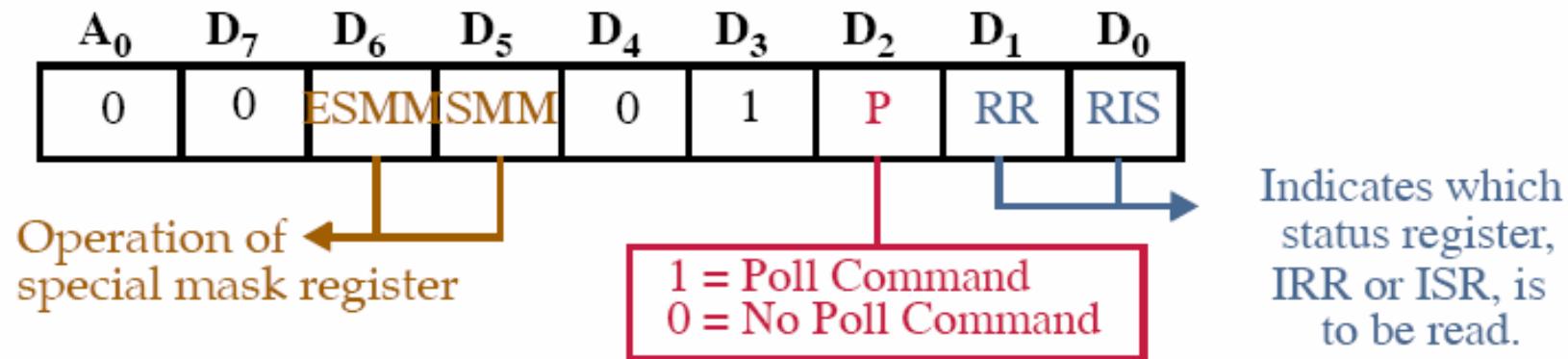
**Specific EOI:** ISR resets a specific interrupt request given by L<sub>2</sub>-L<sub>0</sub>.

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 (L<sub>2</sub>-L<sub>0</sub>).

# Mạch điều khiển ngắt 8259A

- OCW3:



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.

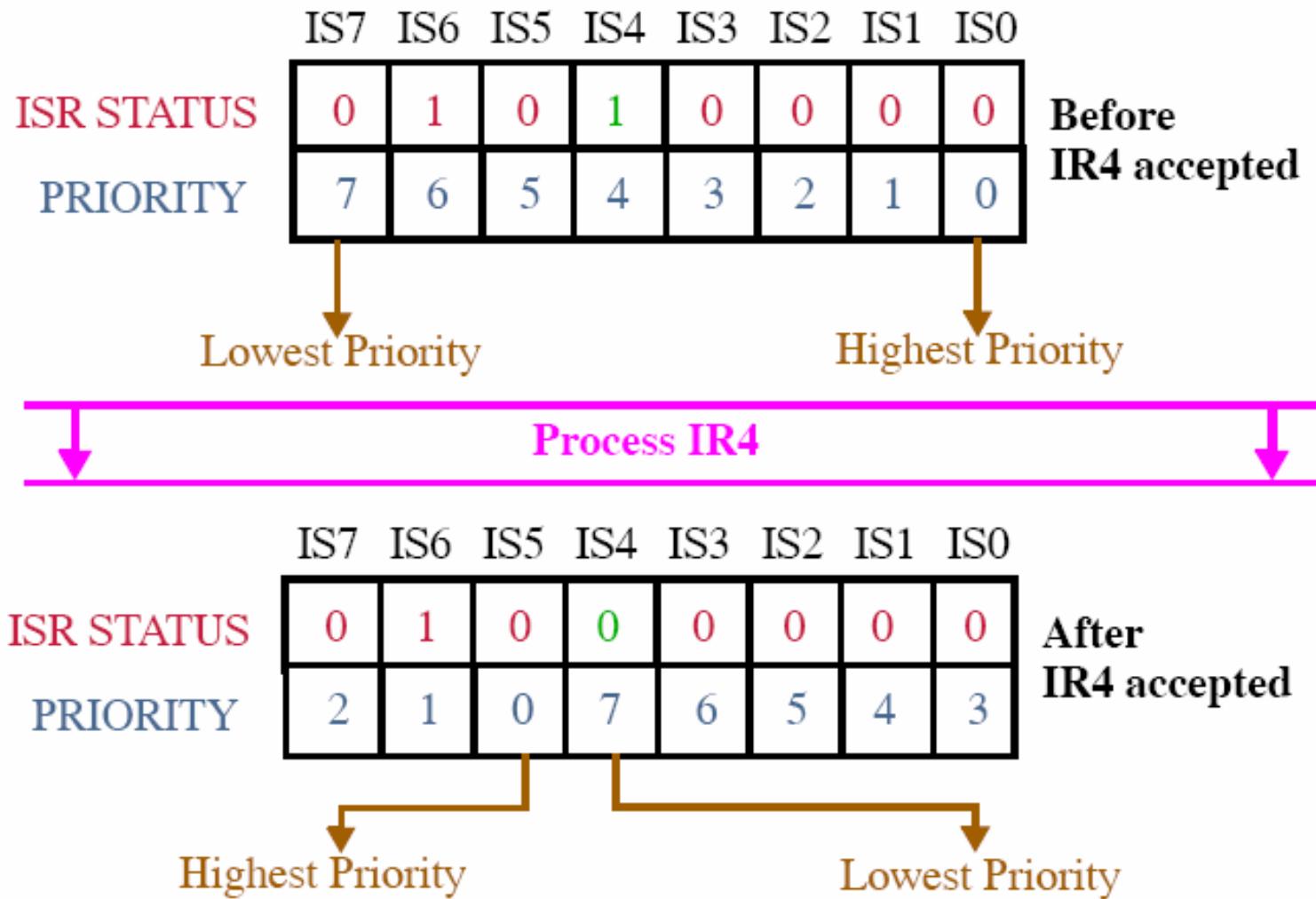
There are three status registers, Interrupt Request Register (IRR), In-Service Register (ISR) and Interrupt Mask Register (IMR).

- 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.



# Mạch điều khiển ngắt 8259A

ISR update procedure with rotating priority configured.





# 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:

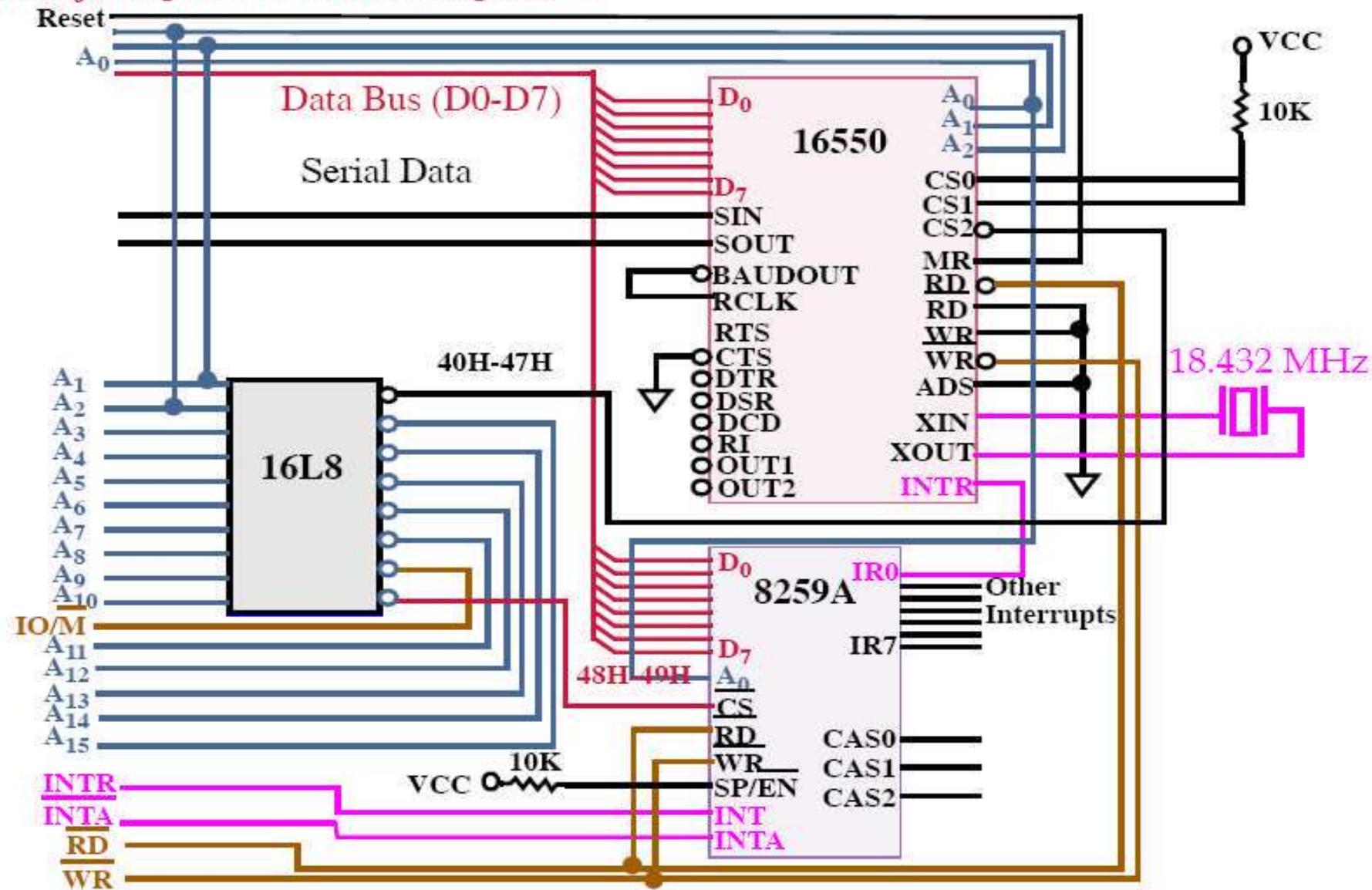
- The transmitter is ready to send another character.
- The receiver has received a character.
- An error is detected while receiving data.
- A modem interrupt occurs.

The 16550 is decoded at 40H and 47H.

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

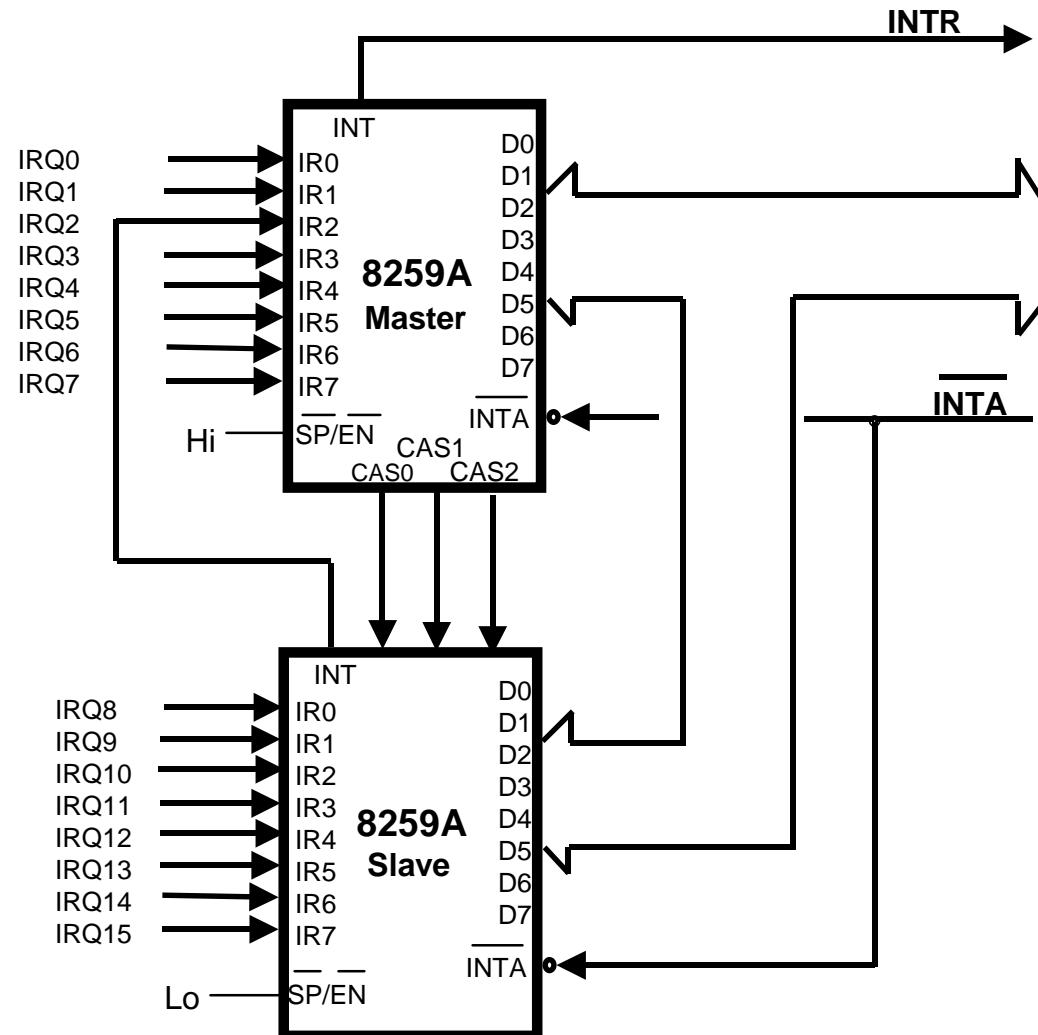
# Mạch điều khiển ngắt 8259A

*Interfacing 16550 UART using 8259A*



# Các ngắt trong máy tính IBM/PC

Cascaded i8259As





# Các ngắt trong máy tính IBM/PC

## Priority

Highest

IRQ0

IRQ1

IRQ2

IRQ8

IRQ9

IRQ10

IRQ11

IRQ12

IRQ13

IRQ14

IRQ15

IRQ3

IRQ4

IRQ5

IRQ6

IRQ7

Lowest

## Use of PC/AT Interrupt

Timer 0

Keyboard

From slave 8259

Real time clock

\*

C0-processor

Hard disk controller

COM2 port

COM1 port

LPT2

Floppydisk controller

LPT1

\* IRQ9 interrupt is redirected to IRQ2 vector

# Các ngắt trong máy tính IBM/PC

PC/AT VECTOR TABLE

0000	— — — —	00h
00020	— — — —	01h
00024	— — — —	02h
00028	— — — —	03h
		04h
		05h
		06h
		07h
		08h
		09h
		0Ah
		0Bh
		0Ch
		0Dh
		0Eh
		0Fh
		10h
		20h
		3Fh
		40h
		60h
		66h
		70h
		71h
		72h
		73h
		74h
		75h
		76h
		77h

IRQ15 Reserved  
 IRQ14 Fixed Disk Controller  
 IRQ13 80x87  
 IRQ12 Reserved  
 IRQ11...Reserved  
 IRQ10 ..Reserved  
 IRQ9 Directed to IRQ2  
 IRQ8 CMOS RTC

EMM  
User Interrupts

ROM BIOS and VIDEO

MSDOS SWI (30 -3F reserved)

ROM BIOS SWI

IRQ 7 LPT1  
 IRQ6 Floppy Disk  
 IRQ5 LPT2  
 IRQ4 COM1 Port  
 IRQ3 COM2 Port  
 IRQ2 Cascade from Slave 8259  
 IRQ1 Keyboard  
 IRQ0 Timer tick

80x87 not present  
 Invalid opcode  
 Print screen (BIOS)  
 Overflow  
 Break point instruction  
 NMI  
 Single step  
 Divide by zero



# 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 6: Truy cập bộ nhớ trực tiếp DMA

- **Giới thiệu về DMA**
- **Mạch DMAC 8237A của Intel**



# 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.

# 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).

<u>IOR</u>	A <sub>7</sub>
<u>IOW</u>	A <sub>6</sub>
<u>MEMR</u>	A <sub>5</sub>
<u>MEMW</u>	A <sub>4</sub>
x	EOP
<u>READY</u>	A <sub>3</sub>
<u>HLDA</u>	A <sub>2</sub>
<u>ADSTB</u>	A <sub>1</sub>
<u>AEN</u>	A <sub>0</sub>
<u>HRQ</u>	V <sub>CC</sub>
<u>CS</u>	DB <sub>0</sub>
<u>CLK</u>	DB <sub>1</sub>
<u>RESET</u>	DB <sub>2</sub>
<u>DACK<sub>2</sub></u>	DB <sub>3</sub>
<u>DACK<sub>3</sub></u>	DB <sub>4</sub>
<u>DREQ<sub>3</sub></u>	DACK <sub>0</sub>
<u>DREQ<sub>2</sub></u>	DACK <sub>1</sub>
<u>DREQ<sub>1</sub></u>	DB <sub>5</sub>
<u>DREQ<sub>0</sub></u>	DB <sub>6</sub>
V <sub>SS</sub>	DB <sub>7</sub>



## Mạch DMA 8237A của Intel

- Although i8237A may not appear as a discrete component in recent PCs, it's still there... (integrated in chipsets, ISPC)
- The i8237A has four independent DMA channels
- Original PC/XT design had one i8237A for four DMA channels
- PC/AT used two i8237As to provide 7 DMA channels
- i8237A is programmable device and can be configured for single transfers, block transfers, Reads, Writes or Memory-to-Memory transfers



## Mạch DMAC 8237A của Intel

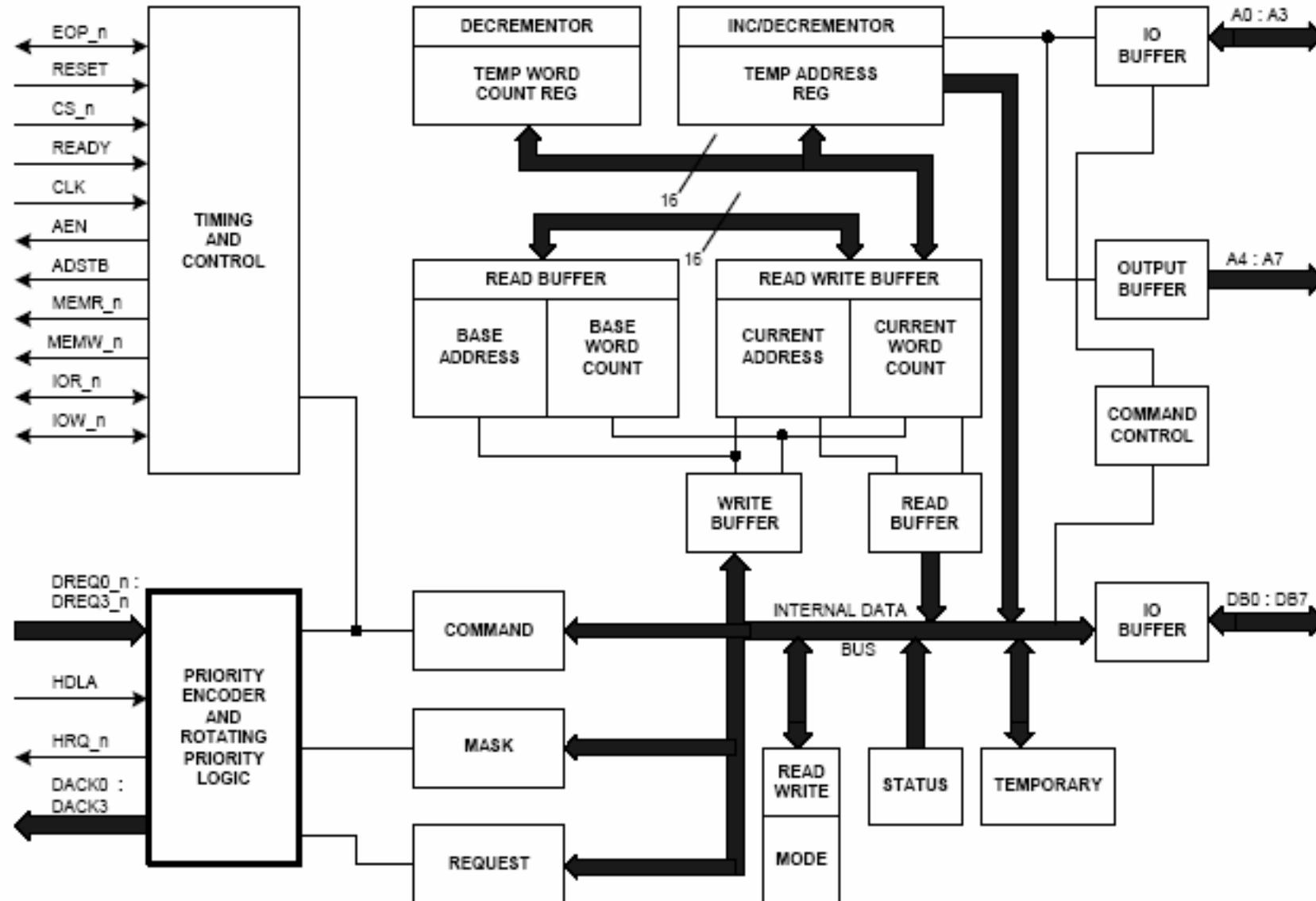
- i8237A allows byte addressing for 8-bit data transfers
- In the PC/AT design, a contrived 16-bit transfer design is implemented using the i8237A
- i8237A uses a multiplexed address and data bus to reduce the device pin count.
  - DB0..DB7 lines contain the data bus along with the high byte of the 16-bit address bus.
  - An external latch is required to demultiplex the address lines



# Mạch DMAC 8237A của Intel

- Clk: < 5MHz.
- $\overline{CS}$ : 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.
- DREQ<sub>3</sub>-DREQ<sub>0</sub>: DMA request inputs used to request a DMA transfer.
- DB<sub>7</sub>-DB<sub>0</sub>: Used to program the 8237 and output upper 8-bits of address.
- $\overline{IOR}$ ,  $\overline{IOW}$ ,  $\overline{MEMR}$ ,  $\overline{MEMW}$ : Outputs used to control memory and I/O.
- $\overline{EOP}$ : Bidirectional: as an input, used to terminate a DMA transfer, as an output, signals the end of the DMA transfer.
- A<sub>3</sub>-A<sub>0</sub>: Address pins select an internal register during programming and output part of the address for a transfer.
- A<sub>7</sub>-A<sub>4</sub>: Address outputs.
- HRQ: Output that connects to HOLD pin on micro to request a DMA.
- DACK<sub>3</sub> - DACK<sub>0</sub>: Used to select an I/O device (ack a DMA request).
- AEN/ADSTB: Enable latch (and strobe) to transfer DB<sub>x</sub> to upper 8 A bits.

# Mạch DMAC 8237A của Intel





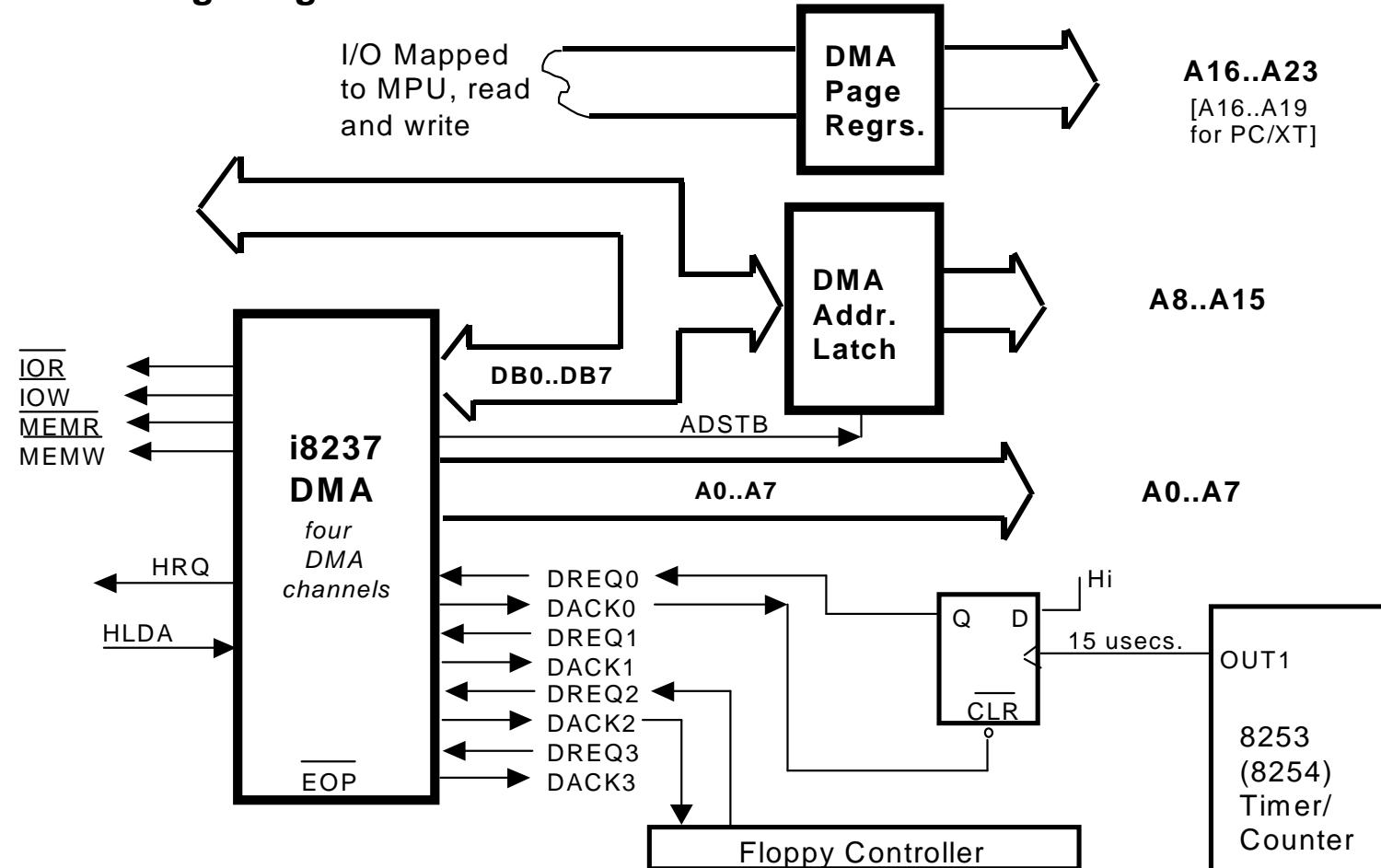
# Mạch DMAC 8237A của Intel

Some of the internal registers are:

- CAR<sub>3</sub> - CAR<sub>0</sub>: Used to hold the 16-bit memory address used for a DMA transfer. Either incremented or decremented after a byte is transferred.
- CWCR<sub>3</sub> - CWCR<sub>0</sub>: 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.

# How the PC uses the i8237A

**i8237A Address Latch and Page Registers**



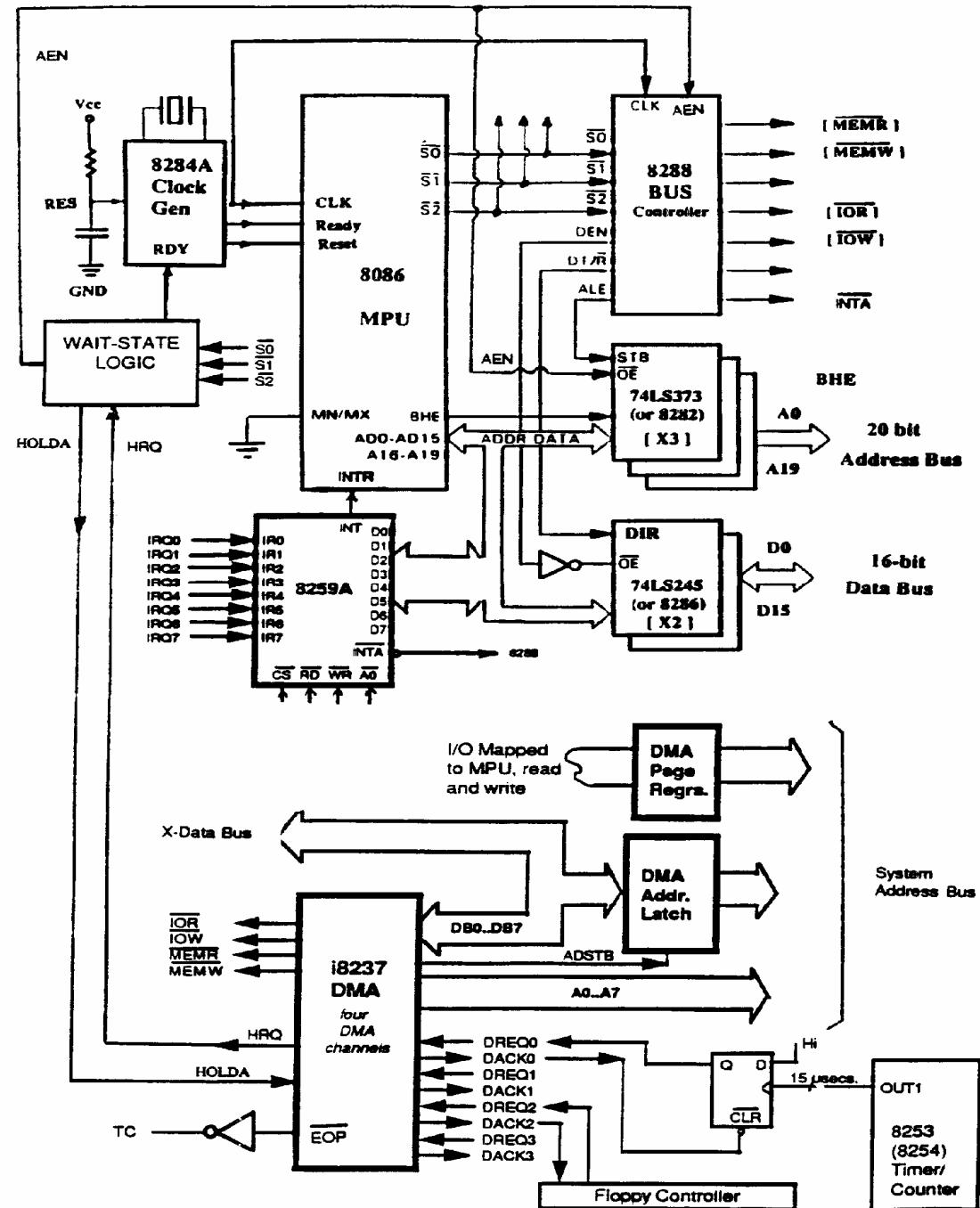


## DMA Address Tracking

- The i8237A has four registers for tracking memory addresses during a DMA block
  - BASE ADDRESS REGISTER
  - BASE WORD COUNT REGISTER
  - CURRENT ADDRESS REGISTER
  - CURRENT WORD COUNT REGISTER

# DMA in the PC/XT

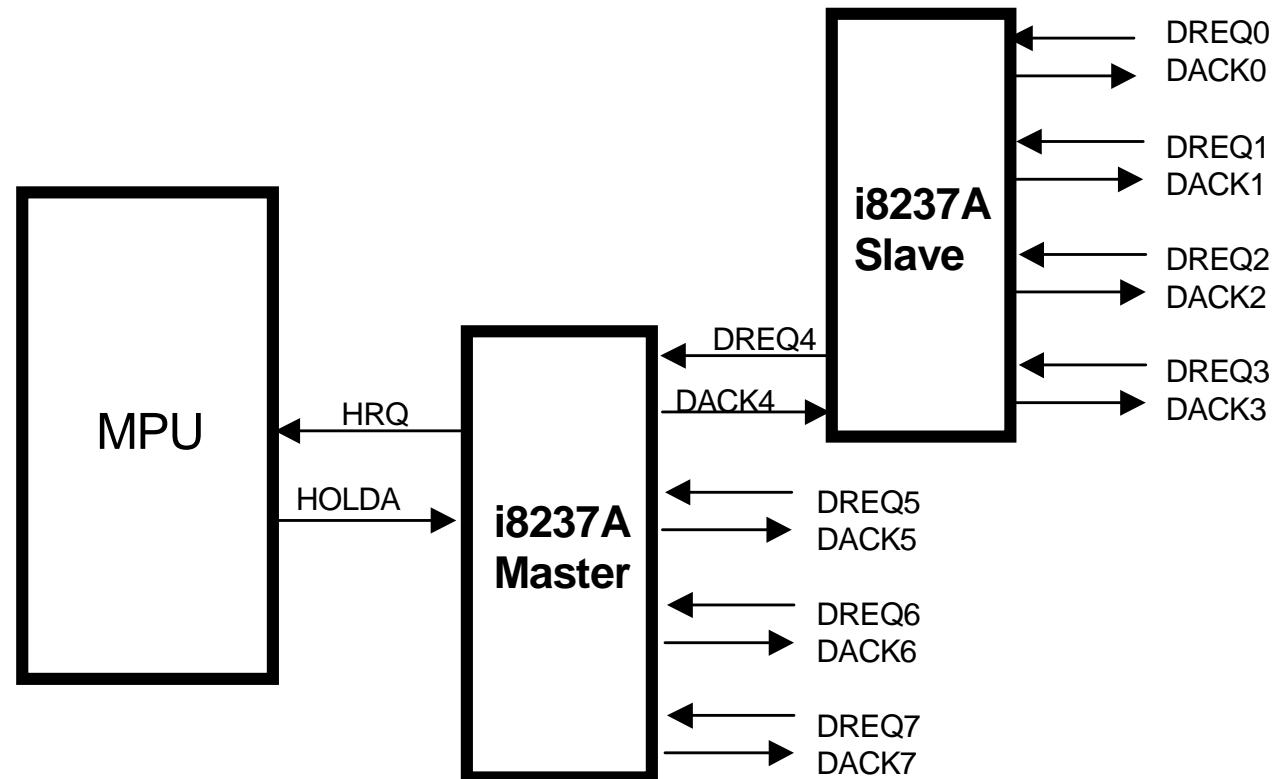
DMA CONTROLLER IN THE PC/XT CIRCUIT



# DMA Cascadation

## Cascaded i8237As in the PC/AT

Cascaded i8237A  
DMA Controllers





## PC/AT DMA Channel priorities

- DMA channel 0 (DREQ0) has the highest priority
- DMA channel 7 (DREQ7) has the lowest
- Note, when a DMA transfer is in session, it cannot be '*interrupted*' by another DMA request, even if the DMA request is made by a higher priority DMA channel.
- The current DMA transfer session will be completed before the pending DMA request is accepted



# DMA Channels in the PC/AT

DMA	Priority	Pre-defined Use in PC/AT	8-bit or 16-bit
DREQ0	Highest	Memory Refresh*	8-bits
DREQ1		Not defined	8-bits
DREQ2		Floppy Disk	8-bits
DREQ3		Not defined	8-bits
DREQ4		Cascade	not used
DREQ5		Not defined	16-bits
DREQ6		Not defined	16-bits
DREQ7	Lowest	Not defined	16-bits



# 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 6: Truy cập bộ nhớ trực tiếp DMA

- **Giới thiệu về DMA**
- **Mạch DMAC 8237A của Intel**



# 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.

# 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).

<u>IOR</u>	A <sub>7</sub>
<u>IOW</u>	A <sub>6</sub>
<u>MEMR</u>	A <sub>5</sub>
<u>MEMW</u>	A <sub>4</sub>
x	EOP
<u>READY</u>	A <sub>3</sub>
<u>HLDA</u>	A <sub>2</sub>
<u>ADSTB</u>	A <sub>1</sub>
<u>AEN</u>	A <sub>0</sub>
<u>HRQ</u>	V <sub>CC</sub>
<u>CS</u>	DB <sub>0</sub>
<u>CLK</u>	DB <sub>1</sub>
<u>RESET</u>	DB <sub>2</sub>
<u>DACK<sub>2</sub></u>	DB <sub>3</sub>
<u>DACK<sub>3</sub></u>	DB <sub>4</sub>
<u>DREQ<sub>3</sub></u>	DACK <sub>0</sub>
<u>DREQ<sub>2</sub></u>	DACK <sub>1</sub>
<u>DREQ<sub>1</sub></u>	DB <sub>5</sub>
<u>DREQ<sub>0</sub></u>	DB <sub>6</sub>
V <sub>SS</sub>	DB <sub>7</sub>



## Mạch DMA 8237A của Intel

- Although i8237A may not appear as a discrete component in recent PCs, it's still there... (integrated in chipsets, ISPC)
- The i8237A has four independent DMA channels
- Original PC/XT design had one i8237A for four DMA channels
- PC/AT used two i8237As to provide 7 DMA channels
- i8237A is programmable device and can be configured for single transfers, block transfers, Reads, Writes or Memory-to-Memory transfers



## Mạch DMAC 8237A của Intel

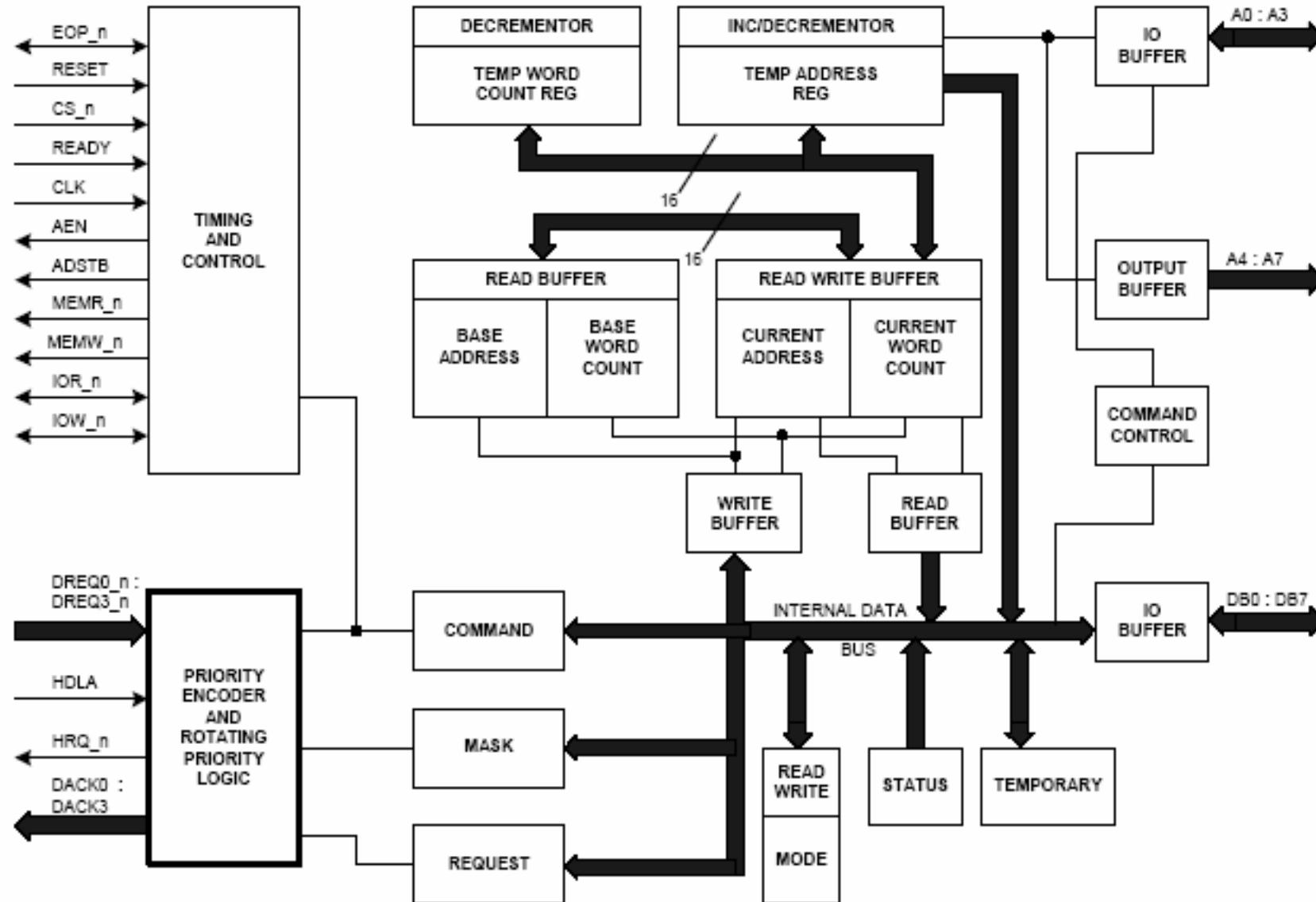
- i8237A allows byte addressing for 8-bit data transfers
- In the PC/AT design, a contrived 16-bit transfer design is implemented using the i8237A
- i8237A uses a multiplexed address and data bus to reduce the device pin count.
  - DB0..DB7 lines contain the data bus along with the high byte of the 16-bit address bus.
  - An external latch is required to demultiplex the address lines



# Mạch DMAC 8237A của Intel

- Clk: < 5MHz.
- $\overline{CS}$ : 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.
- DREQ<sub>3</sub>-DREQ<sub>0</sub>: DMA request inputs used to request a DMA transfer.
- DB<sub>7</sub>-DB<sub>0</sub>: Used to program the 8237 and output upper 8-bits of address.
- $\overline{IOR}$ ,  $\overline{IOW}$ ,  $\overline{MEMR}$ ,  $\overline{MEMW}$ : Outputs used to control memory and I/O.
- $\overline{EOP}$ : Bidirectional: as an input, used to terminate a DMA transfer, as an output, signals the end of the DMA transfer.
- A<sub>3</sub>-A<sub>0</sub>: Address pins select an internal register during programming and output part of the address for a transfer.
- A<sub>7</sub>-A<sub>4</sub>: Address outputs.
- HRQ: Output that connects to HOLD pin on micro to request a DMA.
- DACK<sub>3</sub> - DACK<sub>0</sub>: Used to select an I/O device (ack a DMA request).
- AEN/ADSTB: Enable latch (and strobe) to transfer DB<sub>x</sub> to upper 8 A bits.

# Mạch DMAC 8237A của Intel





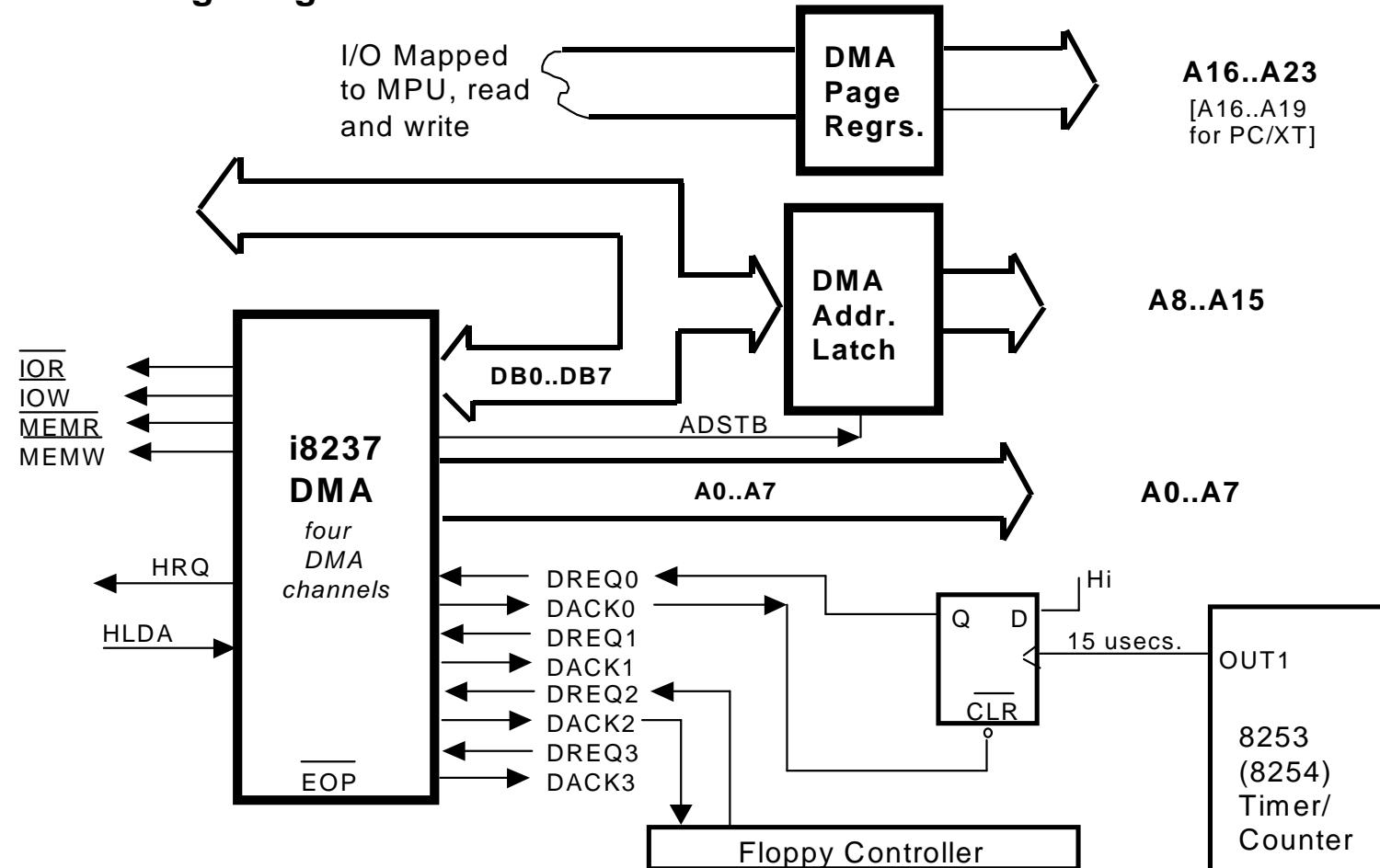
# Mạch DMAC 8237A của Intel

Some of the internal registers are:

- CAR<sub>3</sub> - CAR<sub>0</sub>: Used to hold the 16-bit memory address used for a DMA transfer. Either incremented or decremented after a byte is transferred.
- CWCR<sub>3</sub> - CWCR<sub>0</sub>: 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.

# How the PC uses the i8237A

**i8237A Address Latch and Page Registers**



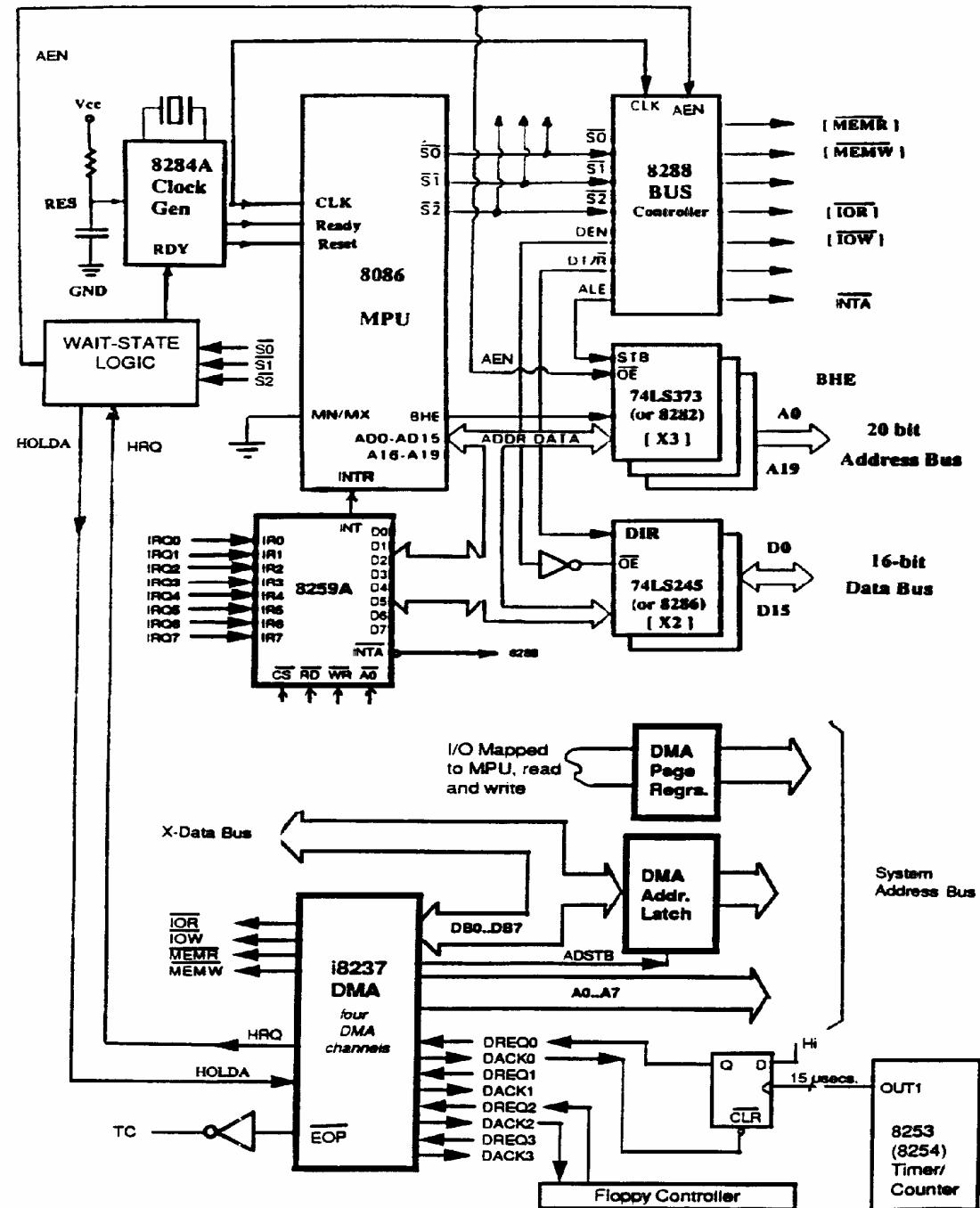


## DMA Address Tracking

- The i8237A has four registers for tracking memory addresses during a DMA block
  - BASE ADDRESS REGISTER
  - BASE WORD COUNT REGISTER
  - CURRENT ADDRESS REGISTER
  - CURRENT WORD COUNT REGISTER

# DMA in the PC/XT

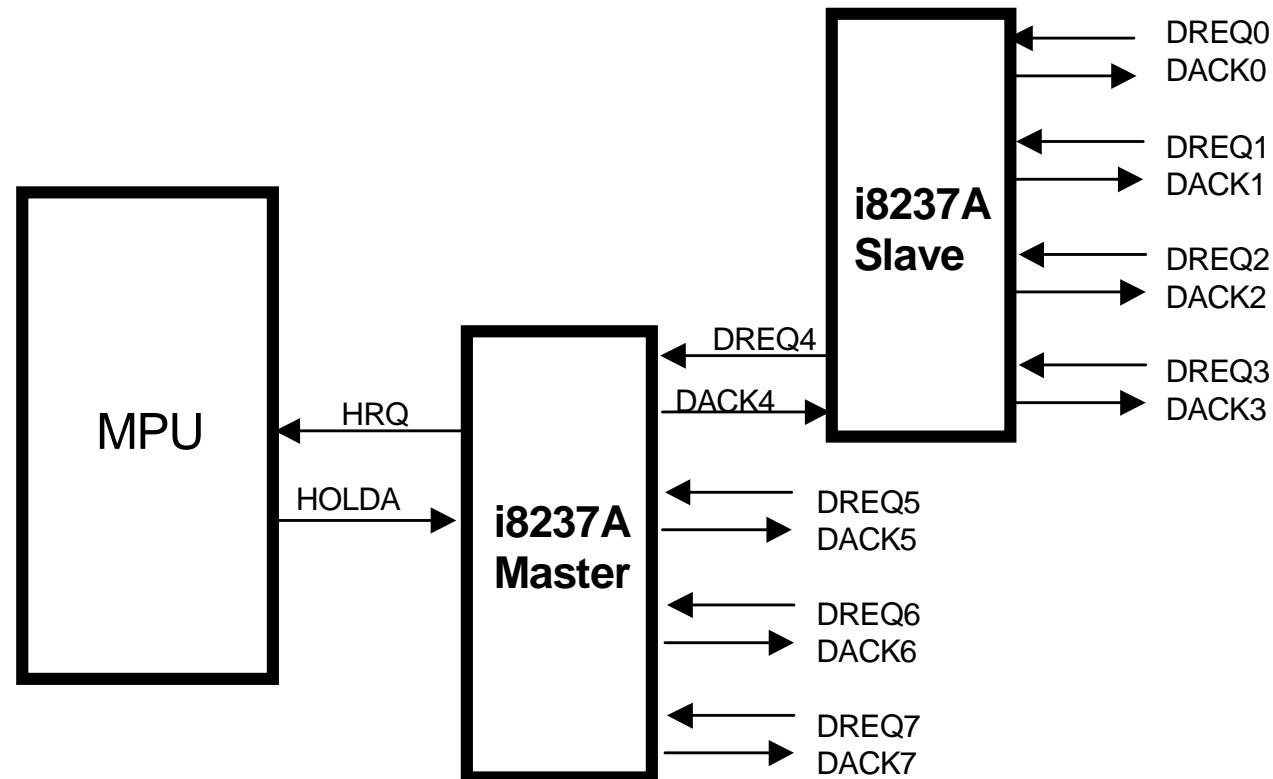
DMA CONTROLLER IN THE PC/XT CIRCUIT



# DMA Cascadation

## Cascaded i8237As in the PC/AT

Cascaded i8237A  
DMA Controllers





## PC/AT DMA Channel priorities

- DMA channel 0 (DREQ0) has the highest priority
- DMA channel 7 (DREQ7) has the lowest
- Note, when a DMA transfer is in session, it cannot be '*interrupted*' by another DMA request, even if the DMA request is made by a higher priority DMA channel.
- The current DMA transfer session will be completed before the pending DMA request is accepted



# DMA Channels in the PC/AT

DMA	Priority	Pre-defined Use in PC/AT	8-bit or 16-bit
DREQ0	Highest	Memory Refresh*	8-bits
<b>DREQ1</b>		<b>Not defined</b>	<b>8-bits</b>
DREQ2		Floppy Disk	8-bits
DREQ3		Not defined	8-bits
DREQ4		Cascade	not used
DREQ5		Not defined	16-bits
DREQ6		Not defined	16-bits
DREQ7	Lowest	Not defined	16-bits

# 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**

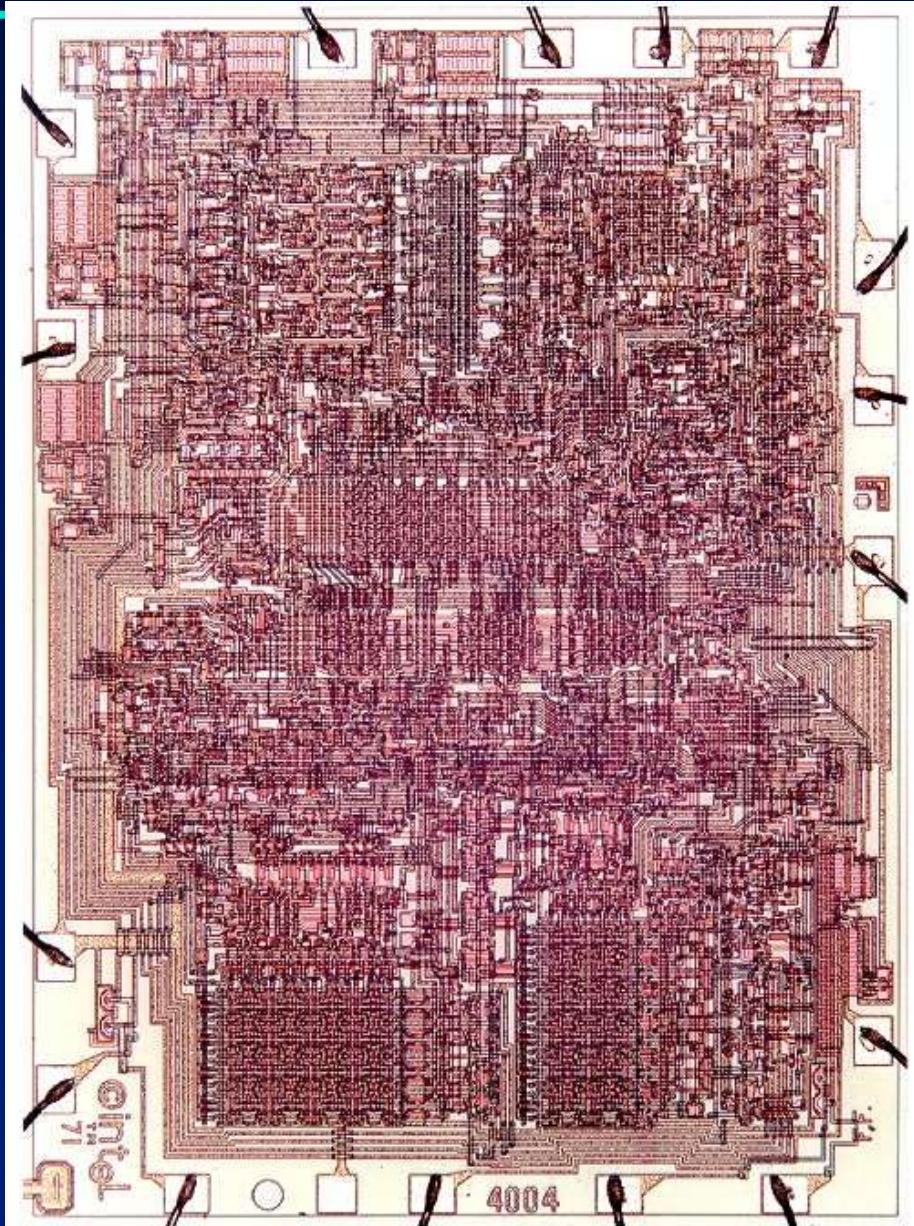
# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Hộ vi điều khiển 8051
  - Hộ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# Chương 7: Các bộ vi xử lý trên thực tế

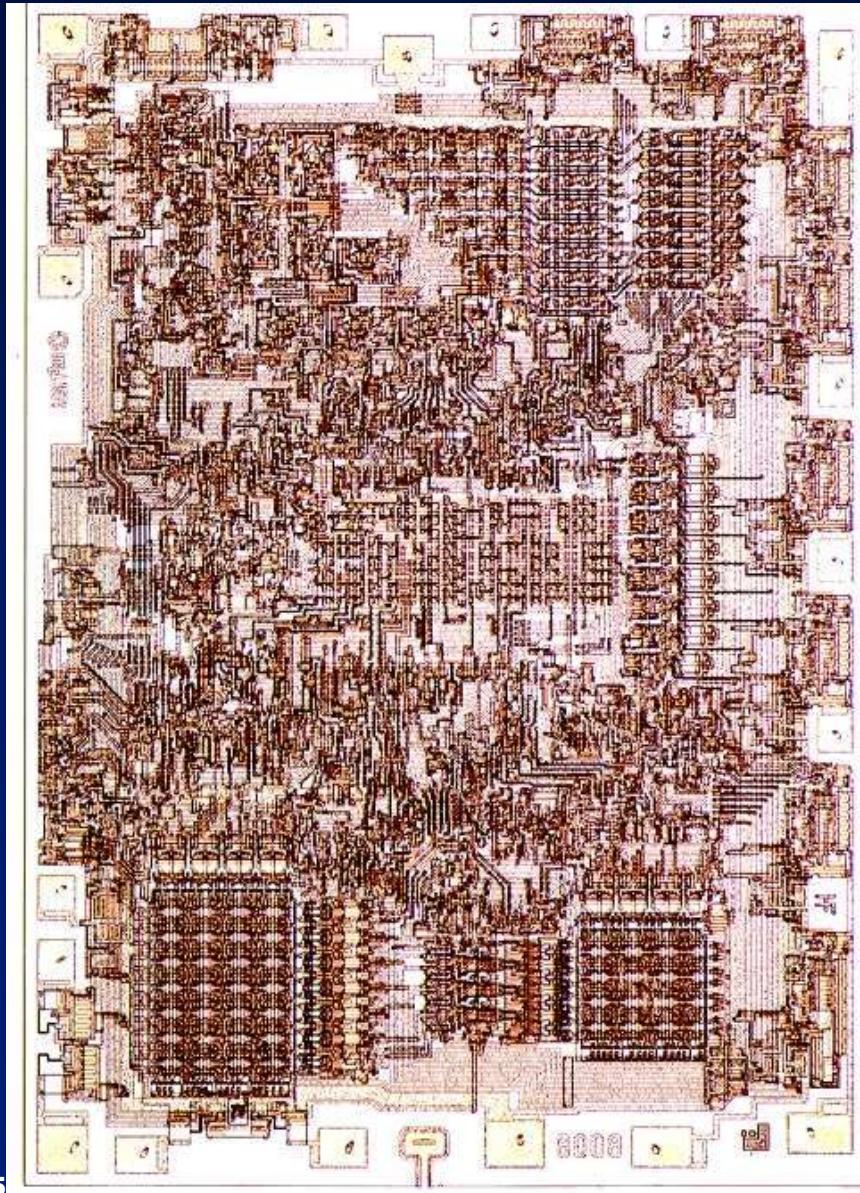
- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Họ vi điều khiển 8051
  - Họ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# Intel 4004



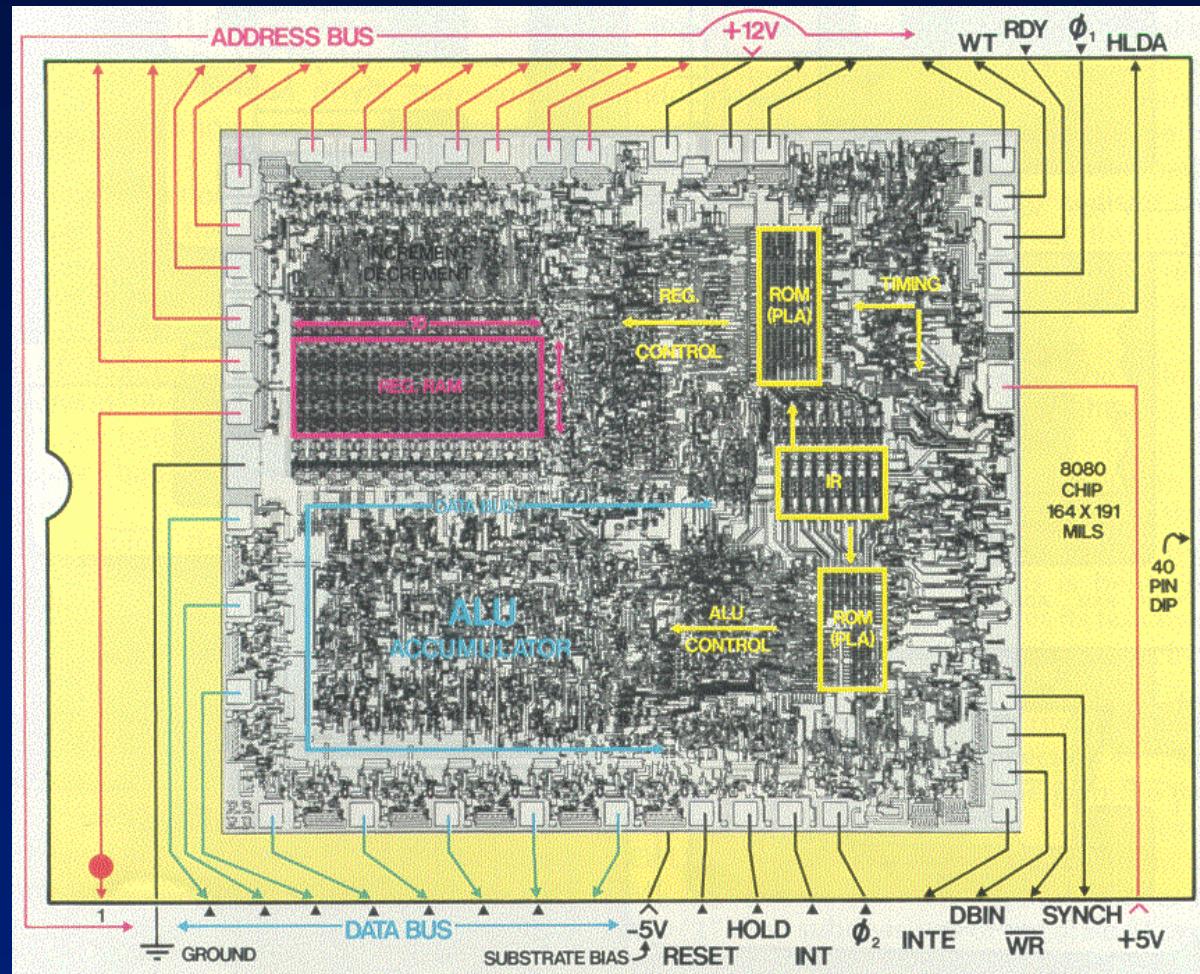
- First microprocessor (1971)
- 4-bit processor
- 2300 Transistors (P-MOS),  $10 \mu\text{m}$
- 0.06 MIPS, 108 KHz, 640 bytes addressable memory
- -15V power supply

# Intel 8008



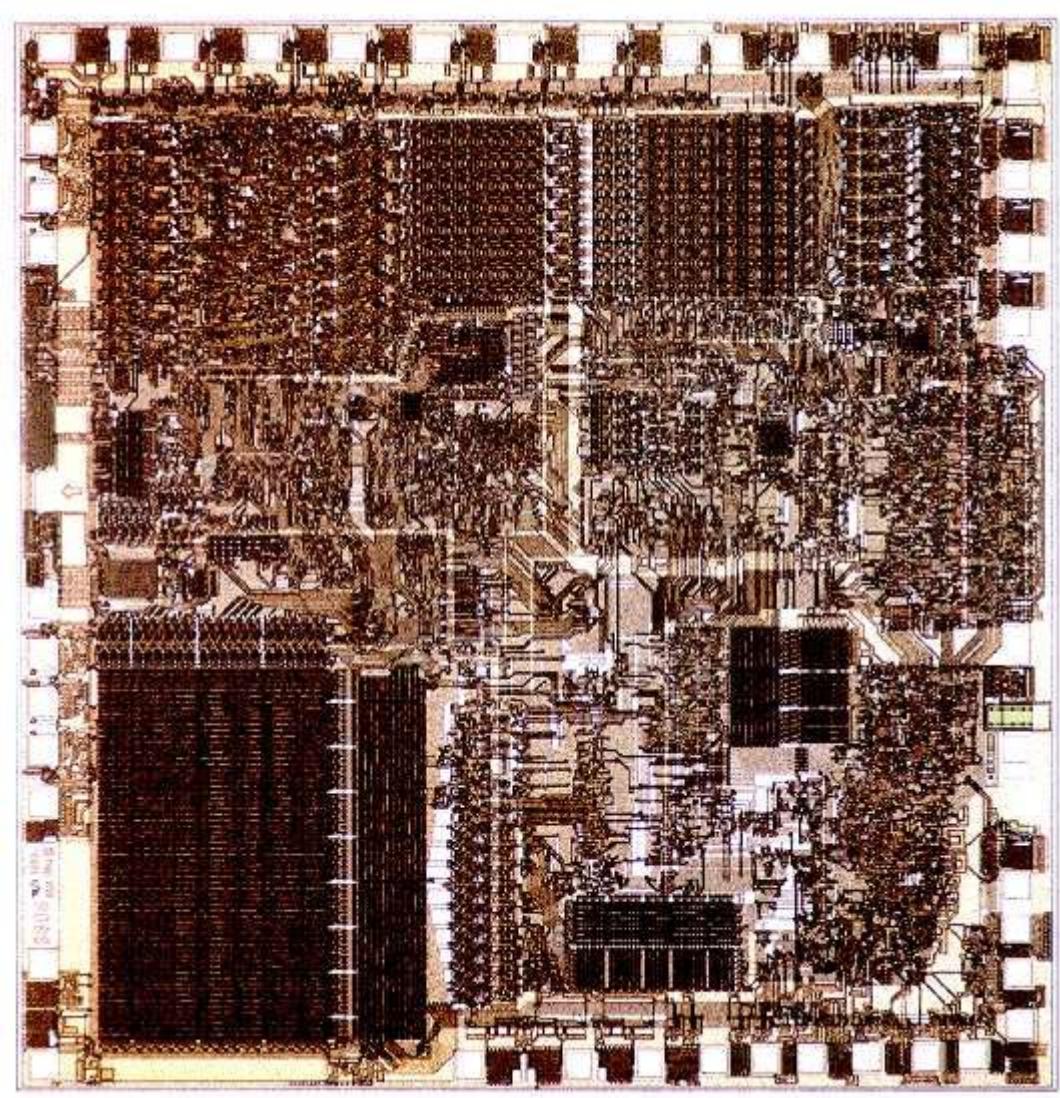
- **First 8-bit processor (1972)**
- **Cost \$500; at this time, a 4-bit processor costed \$50**
- **Complete system had 2 Kbyte RAM**
- **200 KHz clock frequency, 10 μm, 3500 TOR, 0.06 MIPS, 16 Kbyte addressable memory**
- **18 pin package, multiplexed address and data bus**

# Intel 8080



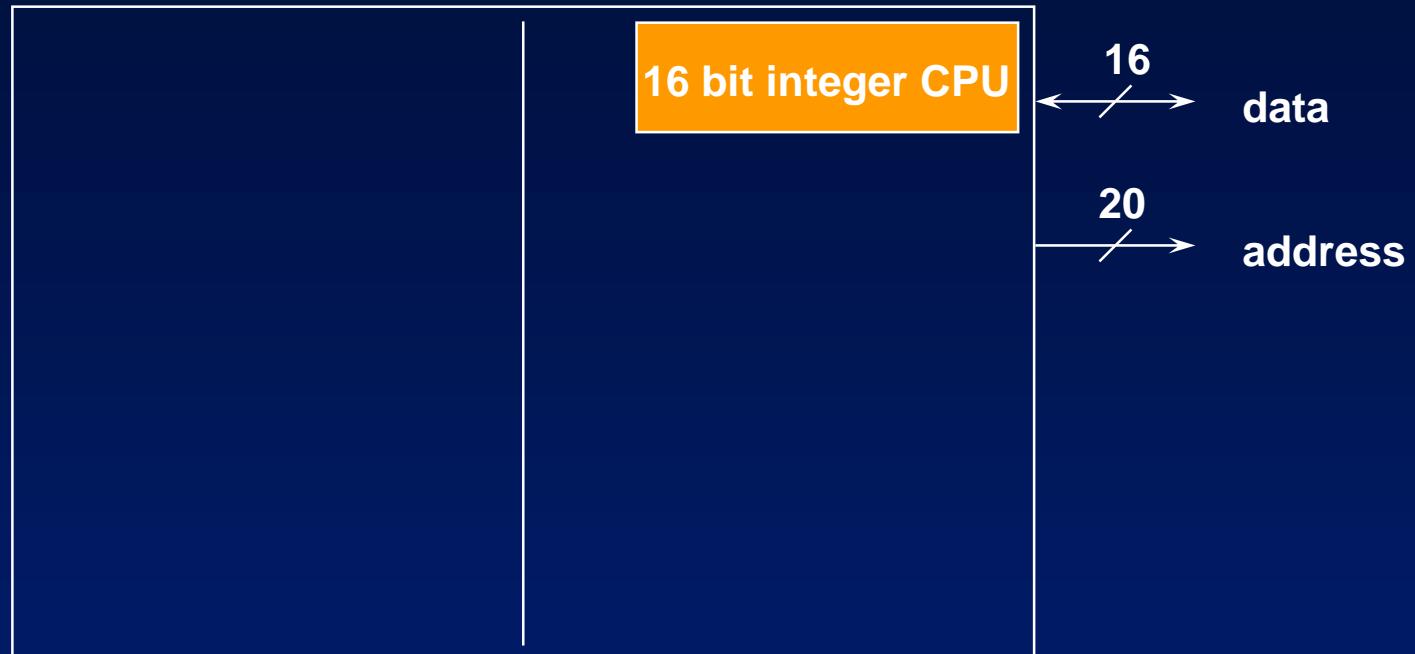
- **Second gen. 8-bit processor, introduced in 1974**
- **40 pin package, NMOS, 500K instructions/s, 6 µm, 2 MHz, ±5V & +12V power supply, 6 KTOR, 0.64 MIPS**
- **64 Kbyte address space (“as large as designers want”, EDN 1974)**

# Intel 8088



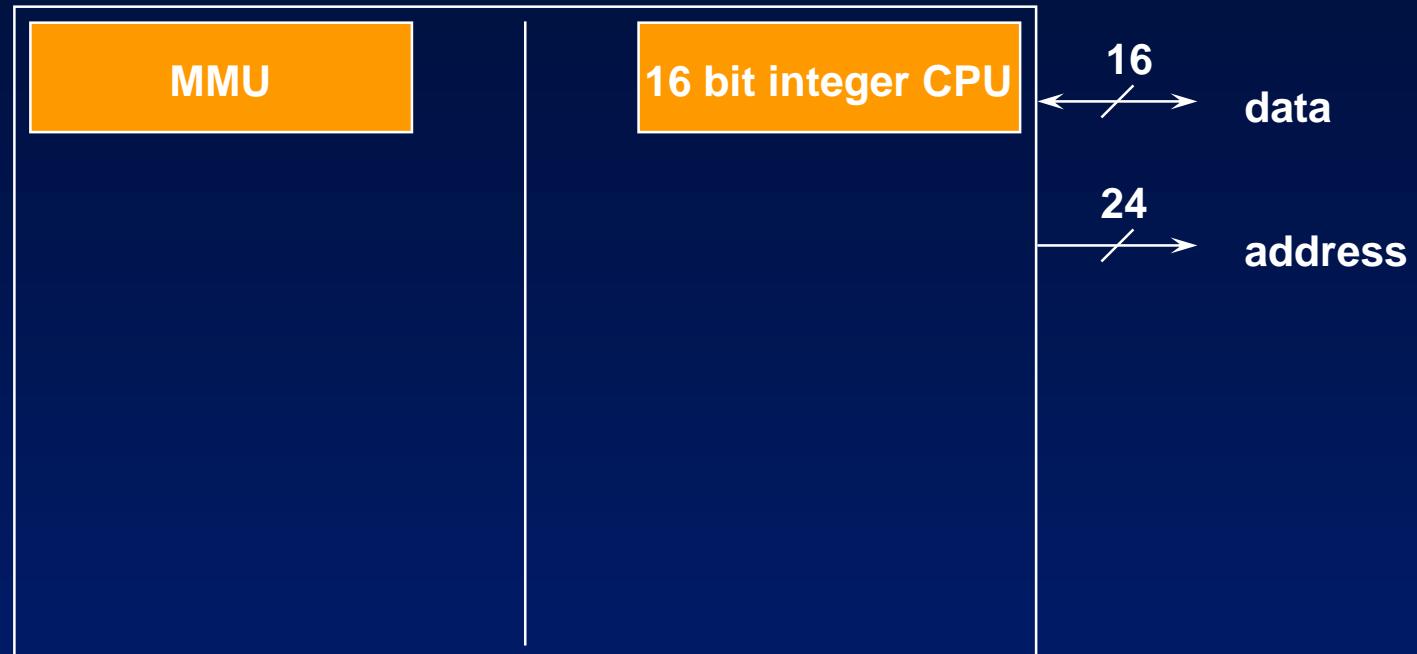
- **16-bit processor**
- **introduced in 1979**
- **3  $\mu$ m, 5 a 8 MHz, 29 KTOR, 0.33 a 0.66 MIPS, 1 Mbyte addressable memory**

# Intel 8086



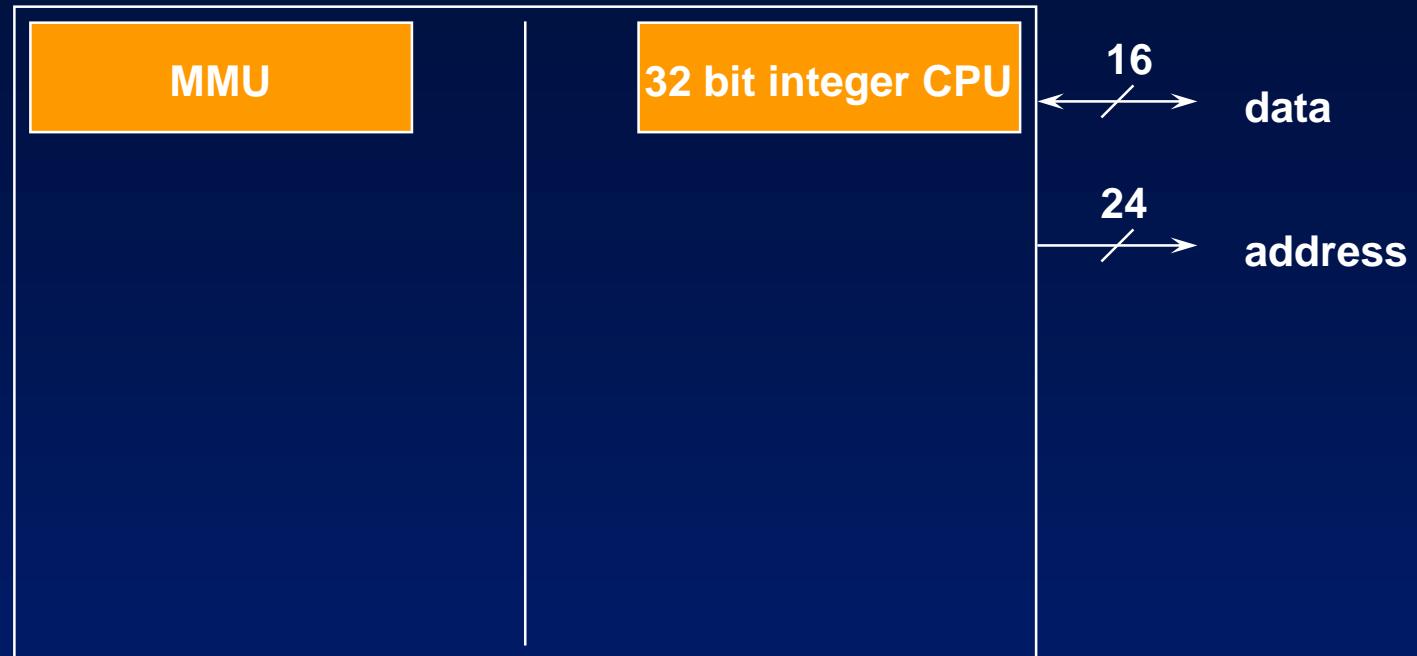
- **Introduced: 1978**
- **Clock frequency: 8 - 10 MHz**

# Intel 80286



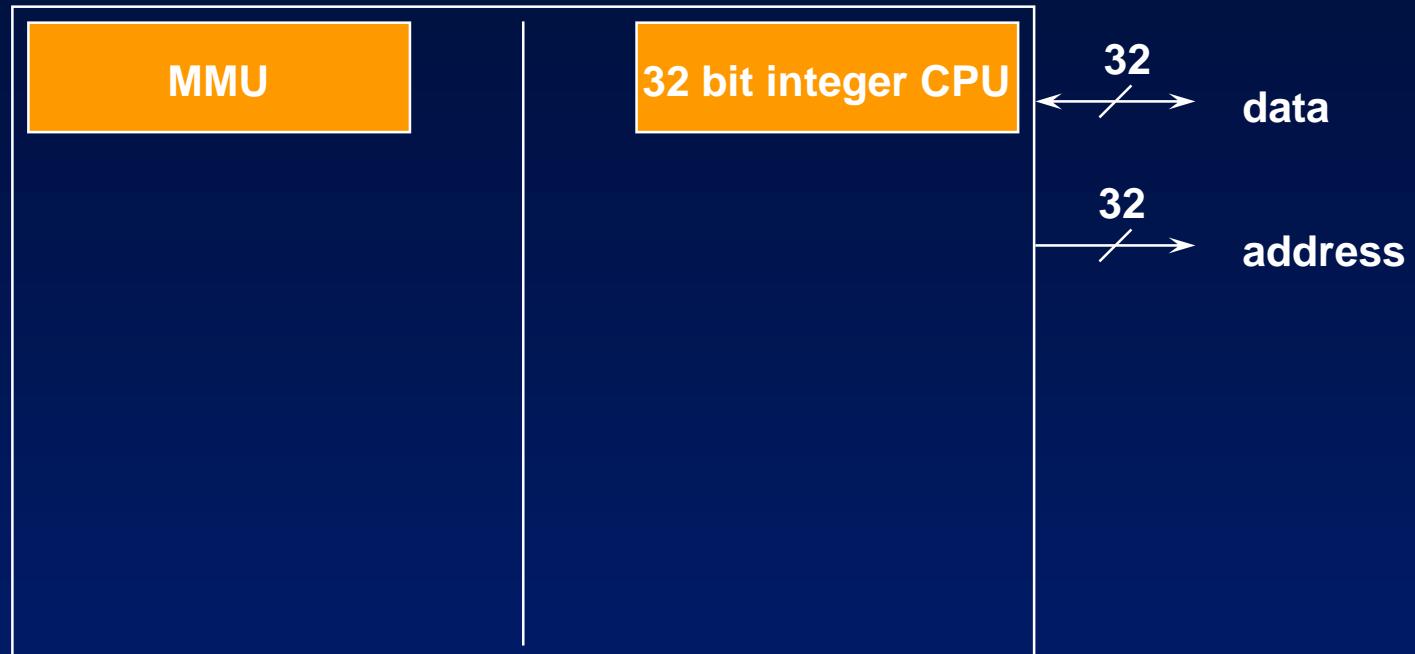
- **Introduced: 1983**
- **1.5  $\mu$ m, 134 KTOR, 0.9 to 2.6 MIPS**
- **Clock frequency: 6 - 25 MHz**

# Intel 80386sx



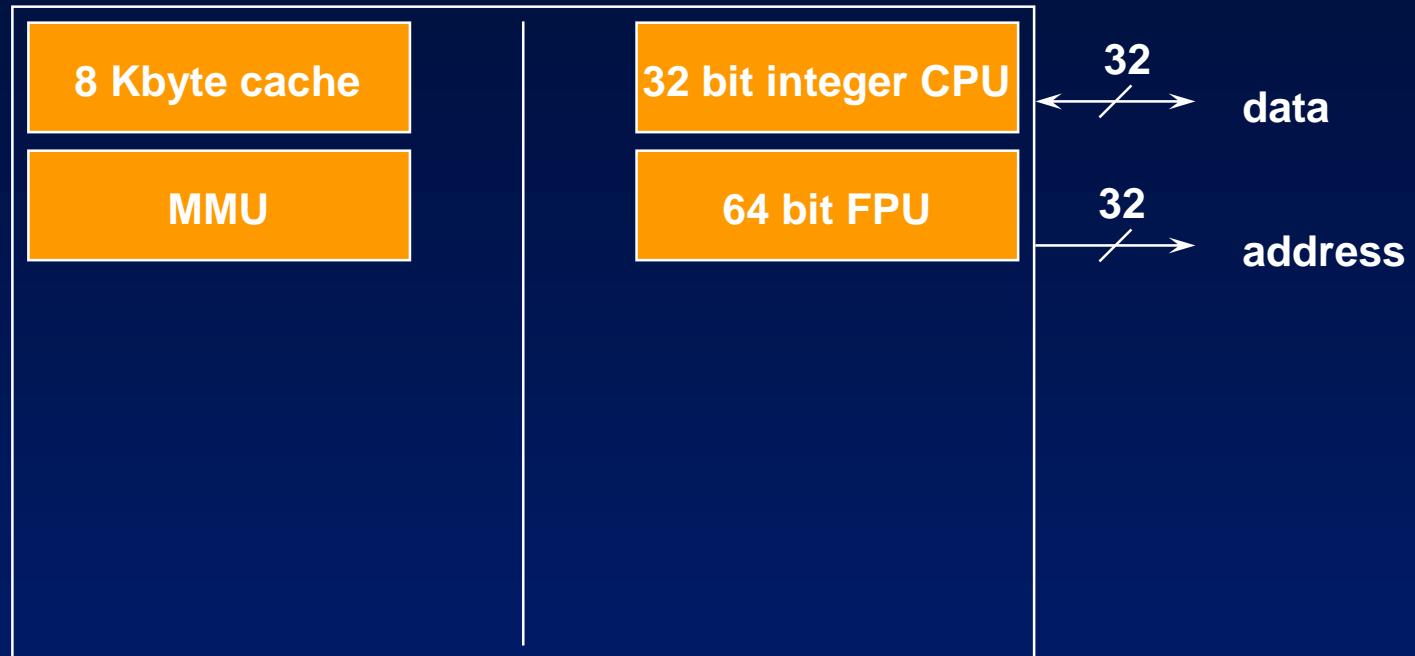
- **Introduced: 1986**
- **1 μm, 275 KTOR, 16 to 33 MHz, 5 to 11 MIPS**
- **Clock frequency: 16 - 25 MHz**
- **Software support and hardware protection for multitasking**

# Intel 80386dx



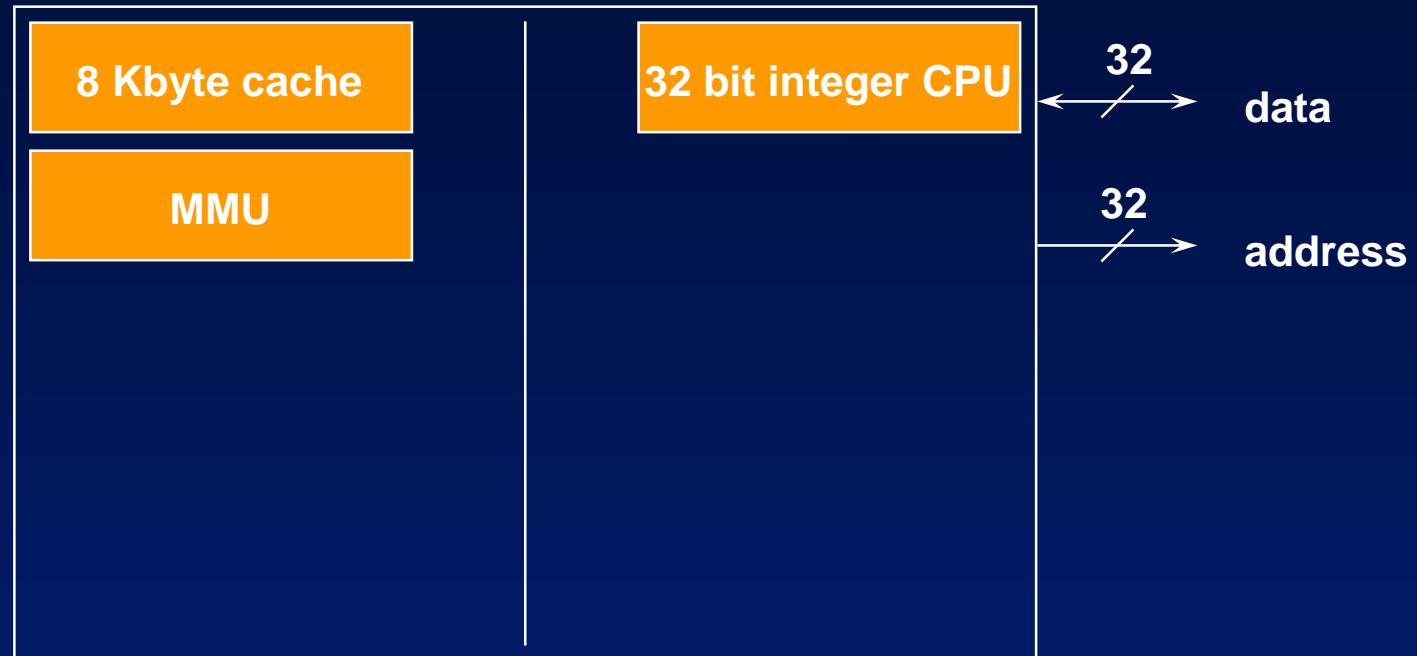
- **Introduced: 1988**
- **Clock frequency: 16 - 40 MHz**
- **Software support and hardware protection for multitasking**

# Intel 80486dx



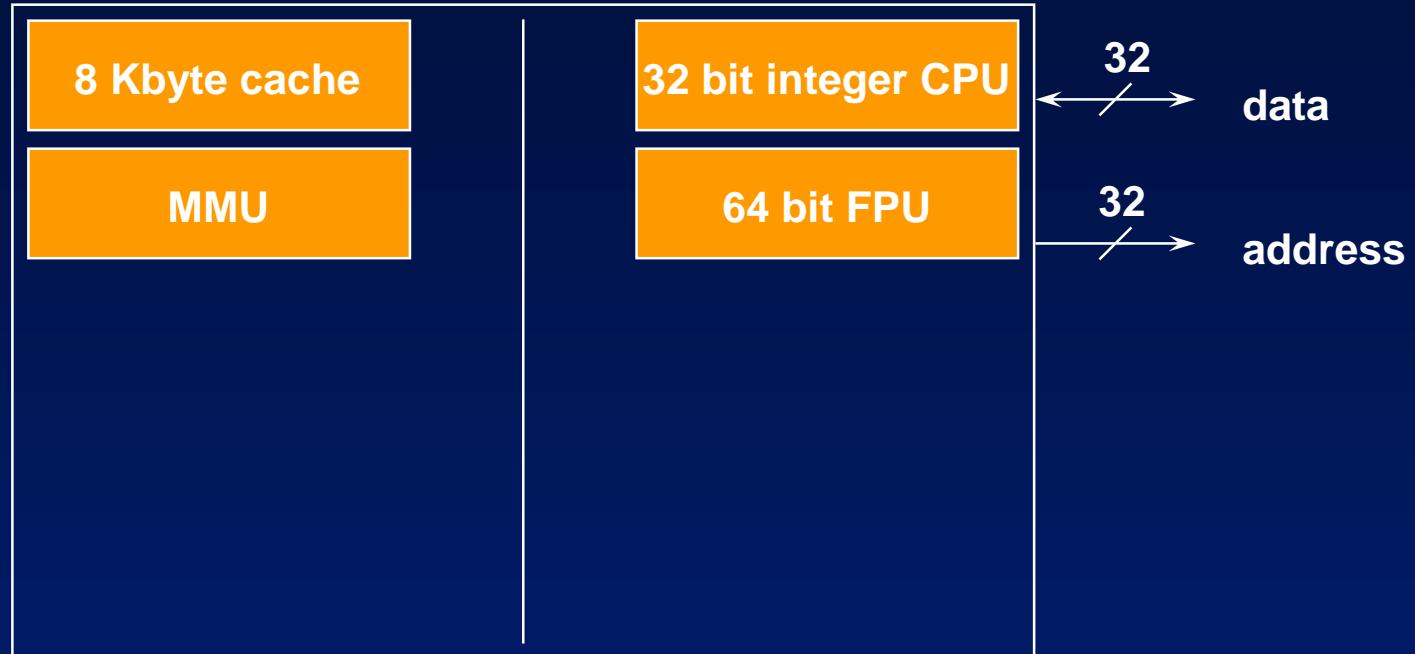
- **Introduced: 1989**
- **Clock frequency: 25 - 50 MHz**
- **Software support and hardware protection for multitasking**
- **Support for parallel processing**
- **Cache required: external memory is not fast enough**

# Intel 80486sx



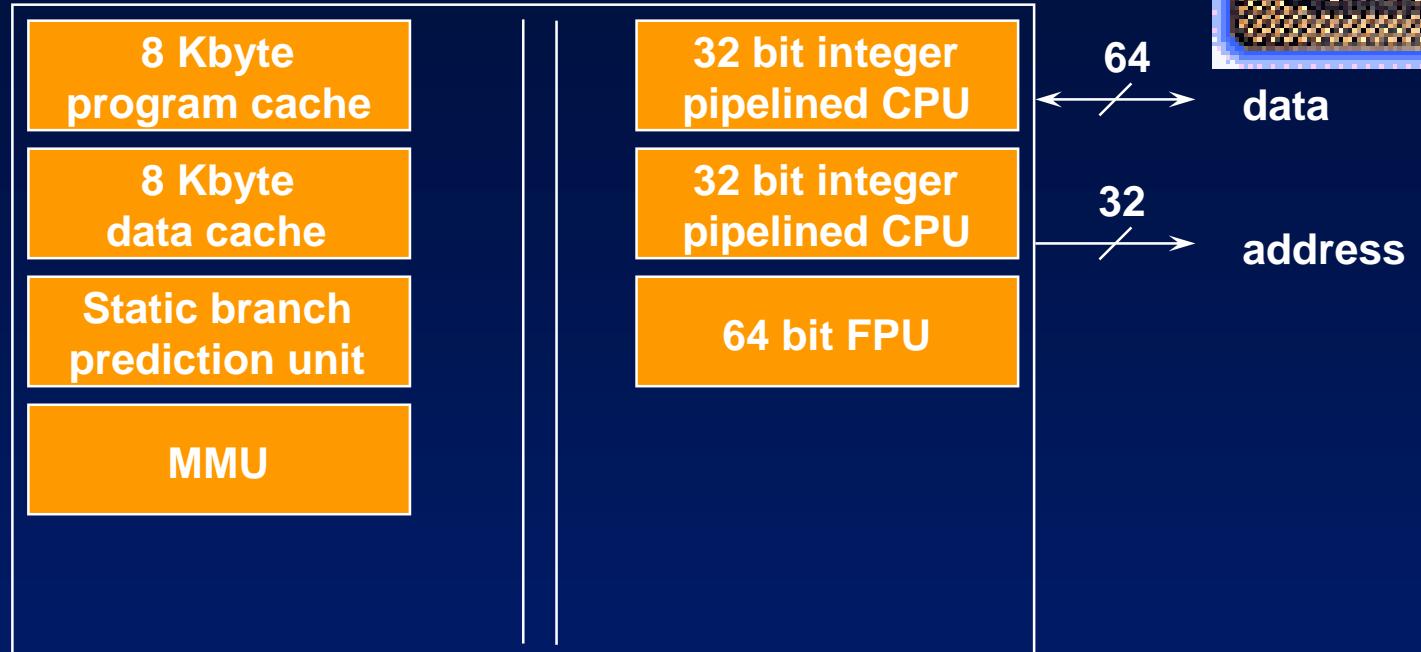
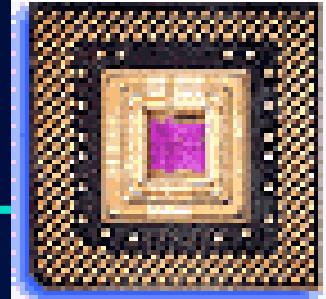
- **Introduced: 1989**
- **0.8  $\mu$ m, 1.2 MTOR, 20 to 41 MIPS**
- **Clock frequency: 25 - 50 MHz**
- **Software support and hardware protection for multitasking**
- **Support for parallel processing**
- **Cache required: external memory is not fast enough**

# Intel 80486dx2



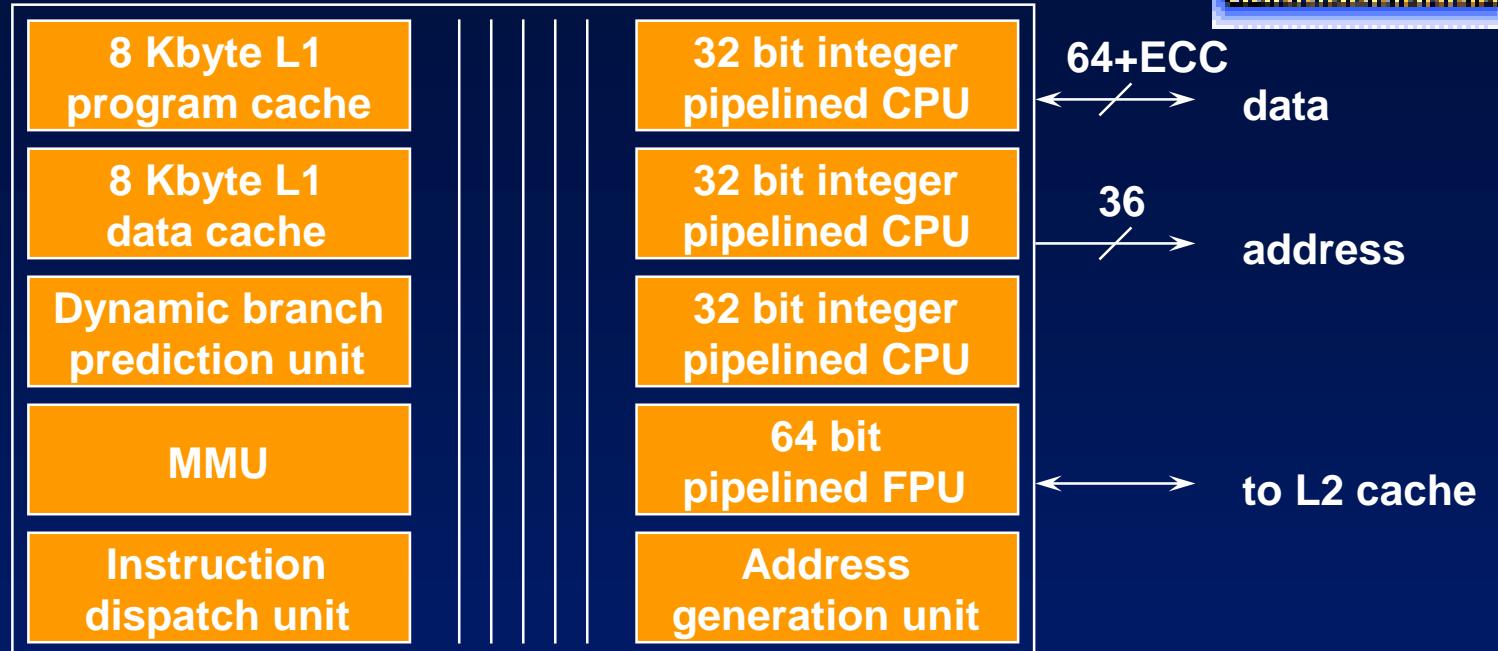
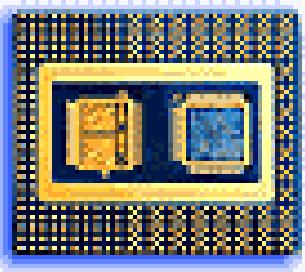
- **Introduced: 1992**
- **Clock frequency: internal: 50 - 66 MHz, external: 25 - 33 MHz**
- **Software support and hardware protection for multitasking**
- **Support for parallel processing**
- **Cache required: external memory is not fast enough**

# Intel Pentium



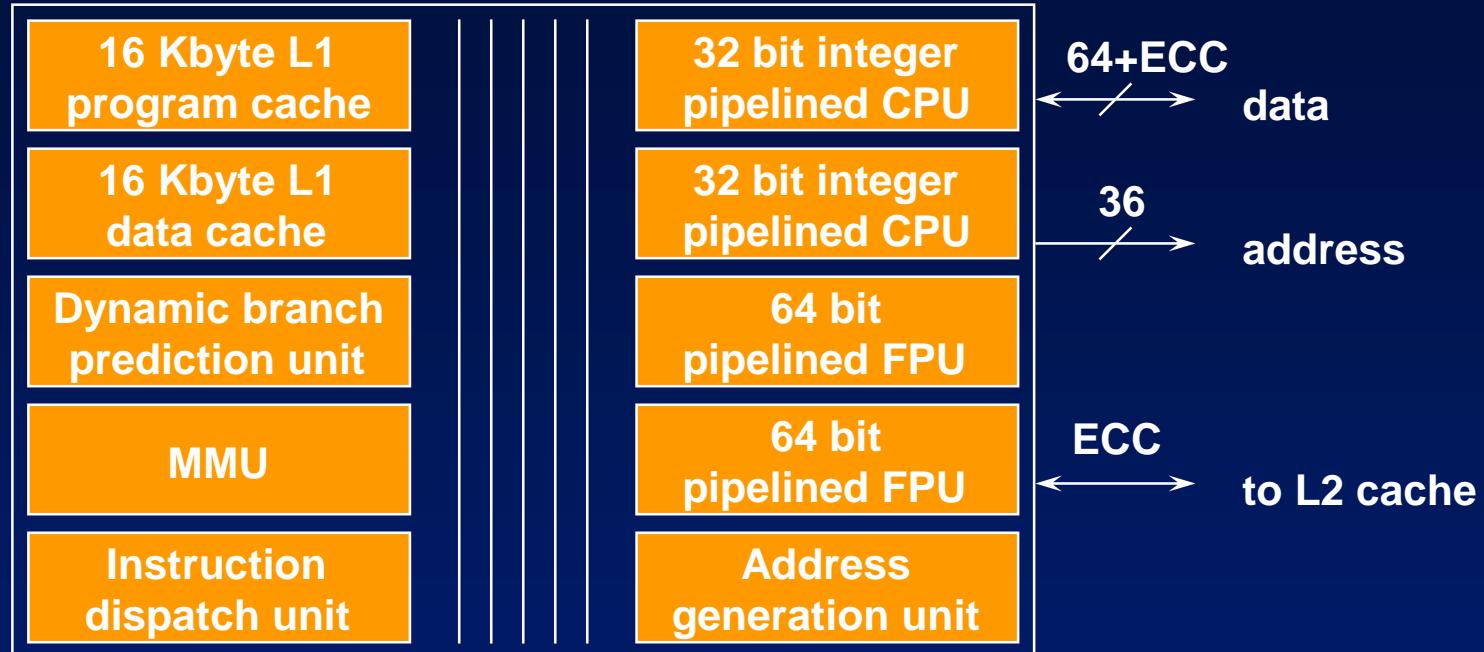
- **Introduced: 1993**
- **(.8 μm, 3.1 MTOR) up to (.35 mm, 4.5 MTOR incl. MMX)**
- **Clock frequency: internal: 60 - 166 MHz, external: 66 MHz**
- **Support for parallel processing: cache coherence protocol**
- **Super scalar**

# Intel Pentium Pro



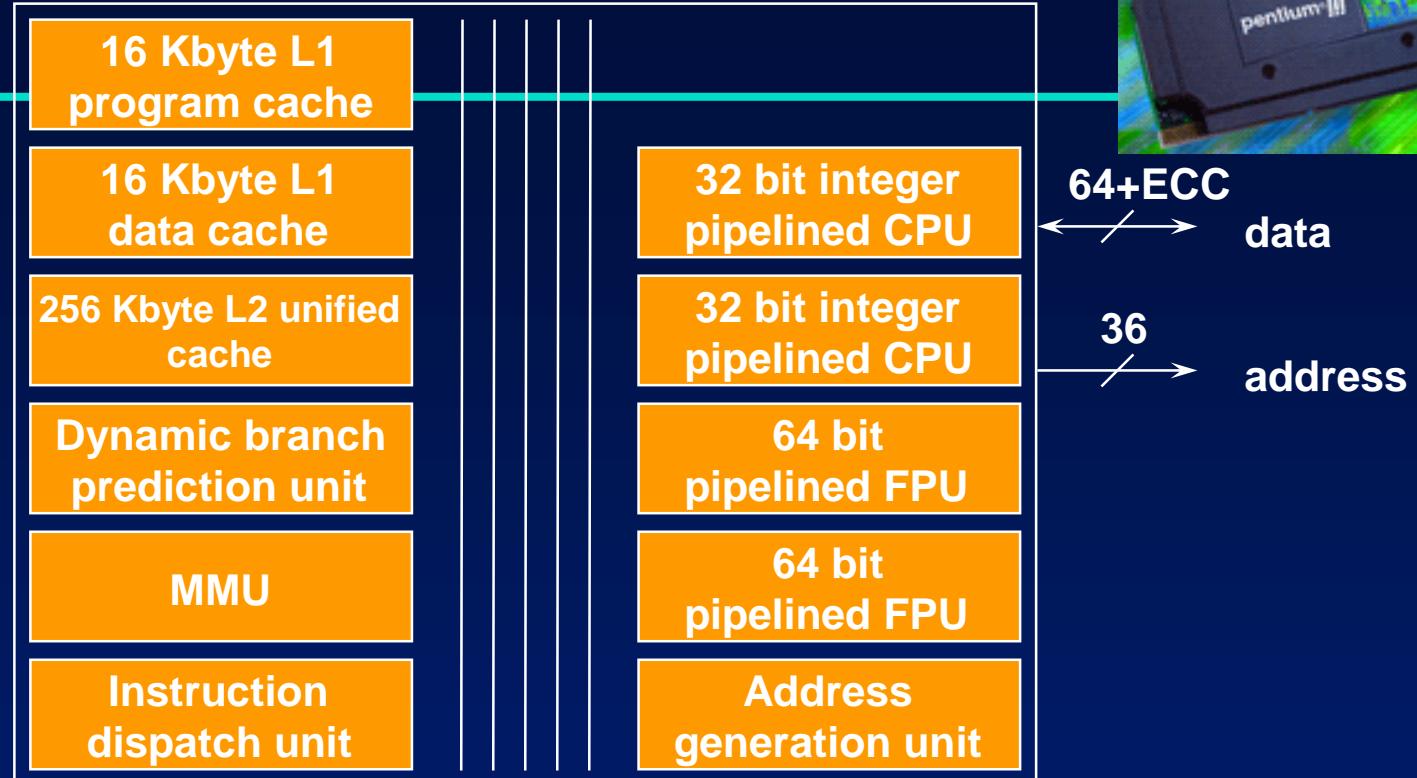
- **Introduced:** 1995, 0.35  $\mu\text{m}$ , 3.3 V, 5.5 MTOR, 35W, 387 pin
- **Clock frequency:** 150 - 200 MHz Internal, 60 - >100 MHz External
- **Super scalar (4 Instr./cycle), super pipelined (12 stages)**
- **Support for symmetrical multiprocessing ( $\leq 4$  CPU)**
- **MCM: 256-1024 Kbyte L2 4-way set associative cache**

# Intel Pentium II



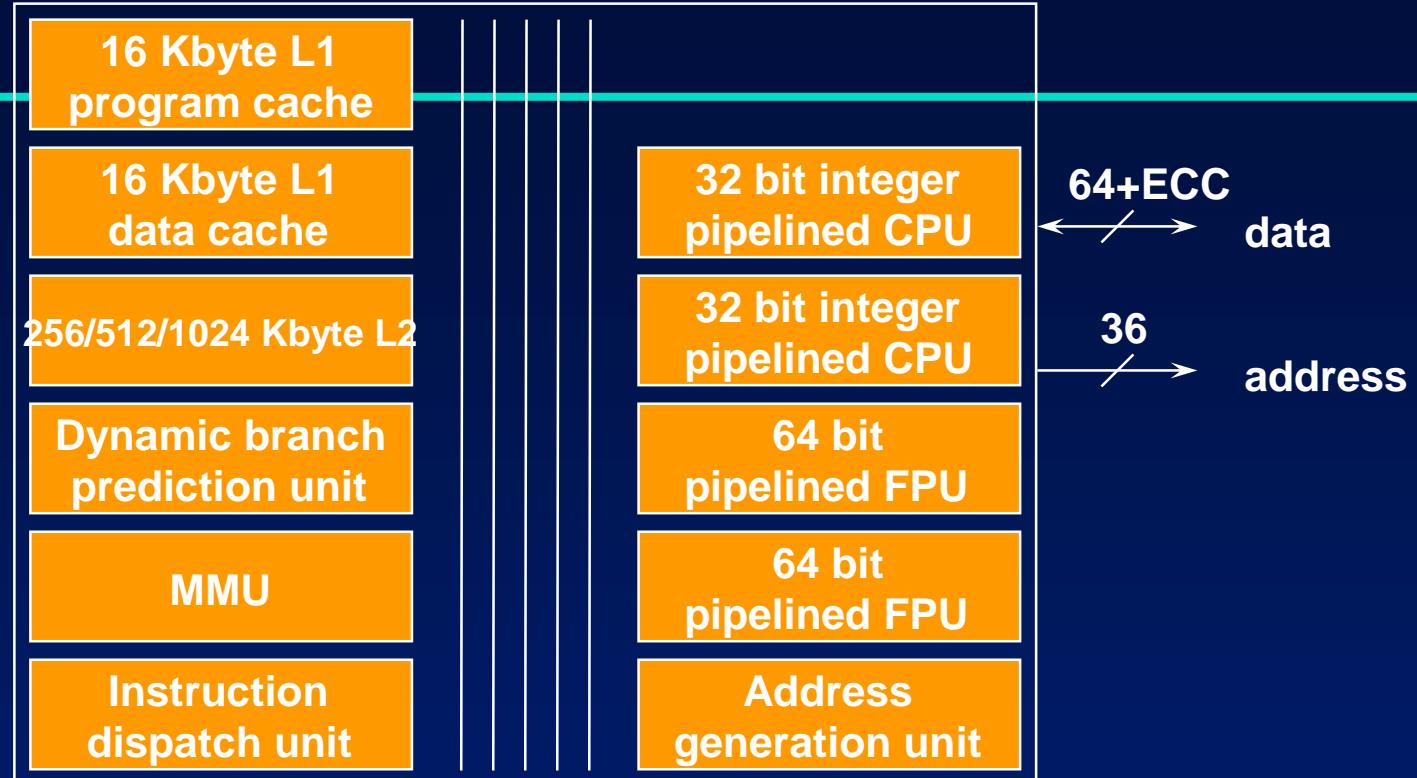
- **Introduced:** 1997, 0.25  $\mu\text{m}$ , 2.0 V, 9 MTOR, 43 W, 242 pin
- **Clock frequency:** 200 - 550 MHz Internal, 100 - 225 MHz L2 cache, 66 - 100 MHz External
- **Super scalar (4 Instr./cycle), super pipelined (12 stages)**
- **Support for symmetrical multiprocessing ( $\leq 8$  CPU)**
- **Single Edge Contact Cartridge with Thermal Sensor: 256-1024 Kbyte L2 4-way set associative cache**

# Intel Pentium III



- **Introduced: 1999, 0.18 μm , 6LM, 1.8 V, 28 MTOR, 370 pin**
- **Clock frequency: 450 - 1130 MHz Internal, 100-133 MHz External**
- **Super scalar (4 Instr./cycle), super pipelined (12 stages)**
- **Support for symmetrical multiprocessing (<2 CPU)**

# Intel Pentium IV



- **Introduced:** 2002, 0.13  $\mu\text{m}$  or 90nm , 1.8 V, 55 MTOR
- **Clock frequency:** 1.4 to 3.8 GHz Internal, 400 to 800 MHz External
- **Super scalar (4 Instr./cycle), super pipelined (12 stages)**
- **Newer versions:** Hyper threading, 3.8 MHz

# Intel Pentium IV

- Available at 3.80F GHz, 3.60F GHz, 3.40F GHz and 3.20F GHz
- • Supports Hyper-Threading Technology1 (HT Technology) for all frequencies with 800 MHz front side bus (FSB)
- • Supports Intel® Extended Memory 64Technology2 (Intel® EM64T)
- Supports Execute Disable Bit capability
- Binary compatible with applications running on previous members of the Intel microprocessor line
- Intel NetBurst® microarchitecture
- FSB frequency at 800 MHz
- Hyper-Pipelined Technology
- Advance Dynamic Execution
- Very deep out-of-order execution
- Enhanced branch prediction
- 775-land Package

# Intel Pentium IV

- **16-KB Level 1 data cache**
- **1-MB Advanced Transfer Cache (on-die, fullspeed Level 2 (L2) cache) with 8-way associativity and Error Correcting Code (ECC)**
- **144 Streaming SIMD Extensions 2 (SSE2) instructions**
- **13 Streaming SIMD Extensions 3 (SSE3) instructions**
- **Enhanced floating point and multimedia unit for enhanced video, audio, encryption, and 3D performance**
- **Power Management capabilities**
- **System Management mode**
- **Multiple low-power states**
- **8-way cache associativity provides improved cache hit rate on load/store operations**

# IA-64 (Itanium)

- Design started in 1994; first samples on the market in 2001
- 64-bit address space ( $4 \times 10^9$  Gbyte; we will never need that much...)
- 256 64-bit integer and 128 82-bit floating point registers; 64 branch target registers; 64 1-bit predicate registers
- 41 bit instruction word length
- 10-stage pipeline
- separate L1 data and program, 96 Kbyte L2 unified on-chip, 4 Mbyte L3 unified off-chip

# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Họ vi điều khiển 8051
  - Họ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# Trends for general purpose processors

- Higher clock frequencies: 4.7 → 30 GHz
- Faster memory: 120 ns → 50 ns
  - not proportional to clock frequency increase => use of caches and special DRAM memories (e.g. SDRAM)
- Limited by power dissipation => decreasing power supply voltage
- Parallel processing
- Memory with processor instead of processor with memory

# The future: general characteristics

	<b>Roadmap 2001</b>				<b>2002</b>	<b>2004</b>	<b>2007</b>	<b>2010</b>	<b>2013</b>	<b>2016</b>
<b>Roadmap 1998</b>		<b>1997</b>	<b>1999</b>		<b>2002</b>	<b>2005</b>	<b>2008</b>	<b>2011</b>	<b>2014</b>	
<b>Roadmap 1995</b>	<b>1995</b>	<b>1998</b>	<b>2001</b>		<b>2004</b>	<b>2007</b>	<b>2010</b>			
<b>Line width (nm)</b>	<b>350</b>	<b>250</b>	<b>180</b>		<b>130</b>	<b>90</b>	<b>65</b>	<b>45</b>	<b>32</b>	<b>22</b>
<b>Number of masks</b>	<b>18</b>	<b>22</b>	<b>22-24</b>		<b>24</b>	<b>24-26</b>	<b>26-28</b>	<b>28</b>	<b>29-30</b>	
<b>Wafer size (mm)</b>	<b>200</b>	<b>200</b>	<b>300</b>		<b>300</b>	<b>300</b>	<b>300</b>			
<b>Number of wiring levels</b>	<b>4-5</b>	<b>6</b>	<b>6-7</b>		<b>7</b>	<b>7-8</b>	<b>8-9</b>	<b>9</b>	<b>10</b>	
<b>Power supply V: desktop</b>	<b>3.3</b>	<b>1.8-2.5</b>	<b>1.5-1.8</b>		<b>1.1-1.5</b>	<b>1.0-1.2</b>	<b>0.7-0.9</b>	<b>0.6</b>	<b>0.5</b>	<b>0.4</b>
<b>Max. power dissipation/chip</b>	<b>80</b>	<b>70</b>	<b>90</b>		<b>130</b>	<b>160</b>	<b>170</b>	<b>175</b>	<b>183</b>	

**Will 22 nm be the end of the scaling race for CMOS?**

**Some believe 10 nm will be the end...**

**...thereafter, semiconductor drive will be scattered  
(MEMS, sensors, magnetic, optic, polymer, bio, ...)**

**Depending on application domain: besides and beyond silicon**

## Besides and beyond silicon (e.g. polymer electronics)



## Besides and beyond silicon: applied to future ambient intelligent environments



## Besides and beyond silicon: applied to future ambient intelligent environments



## Besides and beyond silicon: applied to ambient intelligent HomeLab (2002)



# The future: high performance ( $\mu$ P)

	Roadmap 2001				2001	2004	2007	2012	2016
<b>Roadmap 1998</b>		<b>1997</b>	<b>1999</b>	<b>2002</b>	<b>2005</b>	<b>2008</b>	<b>2011</b>	<b>2014</b>	
<b>Roadmap 1995</b>	<b>1995</b>	<b>1998</b>	<b>2001</b>	<b>2004</b>	<b>2007</b>	<b>2010</b>			
<b>Number of TOR</b>	<b>6M</b>	<b>11M</b>	<b>21M</b>	<b>76M</b>	<b>200M</b>	<b>520M</b>	<b>1.4G</b>	<b>3.6G</b>	
<b>On chip local clock freq. (MHz)</b>		<b>750</b>	<b>1250</b>	<b>2100</b>	<b>3500</b>	<b>6000</b>	<b>10000</b>	<b>16903</b>	
<b>On chip global clock freq. (MHz)</b>	<b>300</b>	<b>375</b>	<b>1200</b>	<b>1600</b>	<b>2000</b>	<b>2500</b>	<b>3000</b>	<b>3674</b>	
<b>Chip size (mm<sup>2</sup>)</b>	<b>250</b>	<b>300</b>	<b>340</b>	<b>430</b>	<b>520</b>	<b>620</b>	<b>750</b>	<b>901</b>	

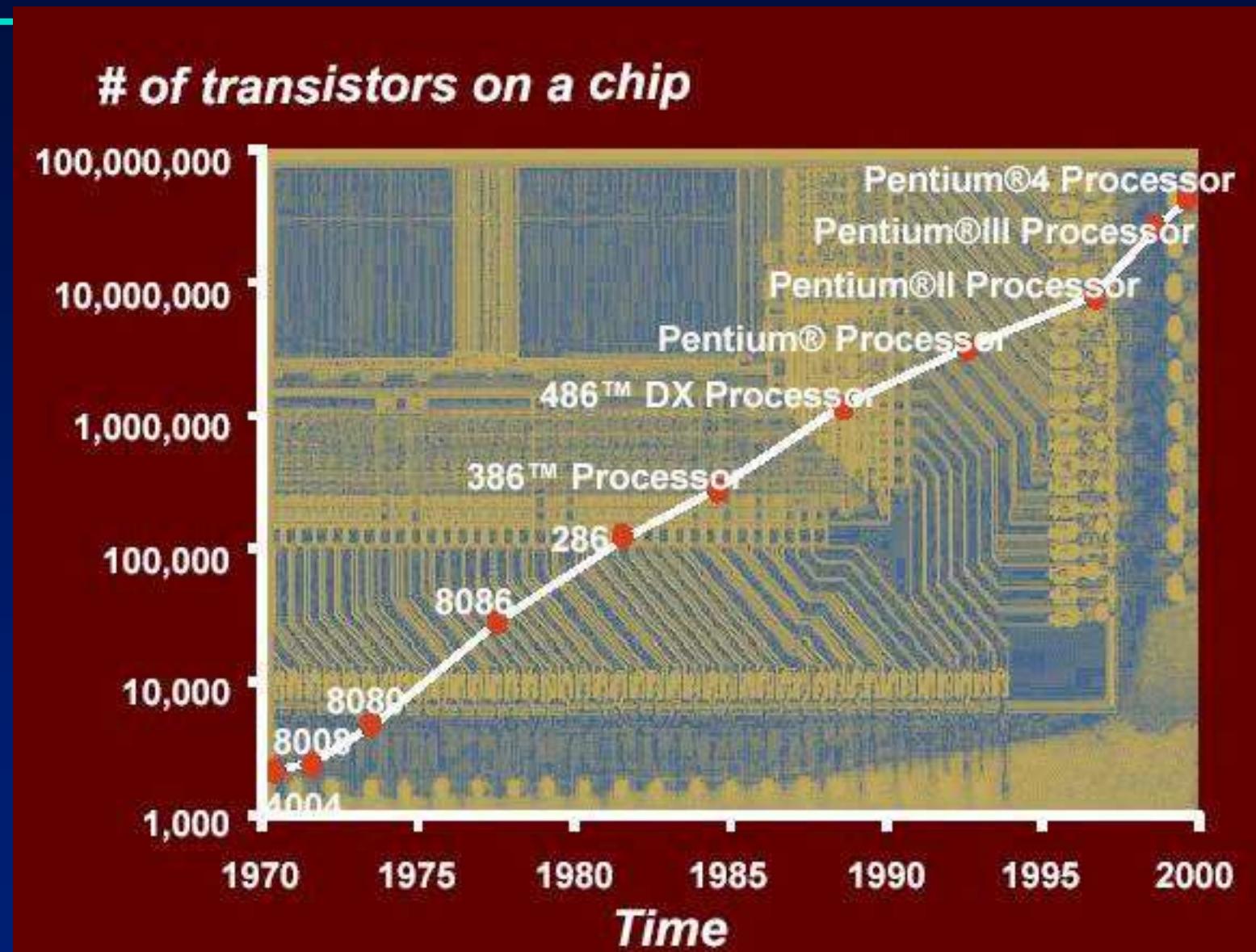
- CTO Intel says in 2001

- 2005

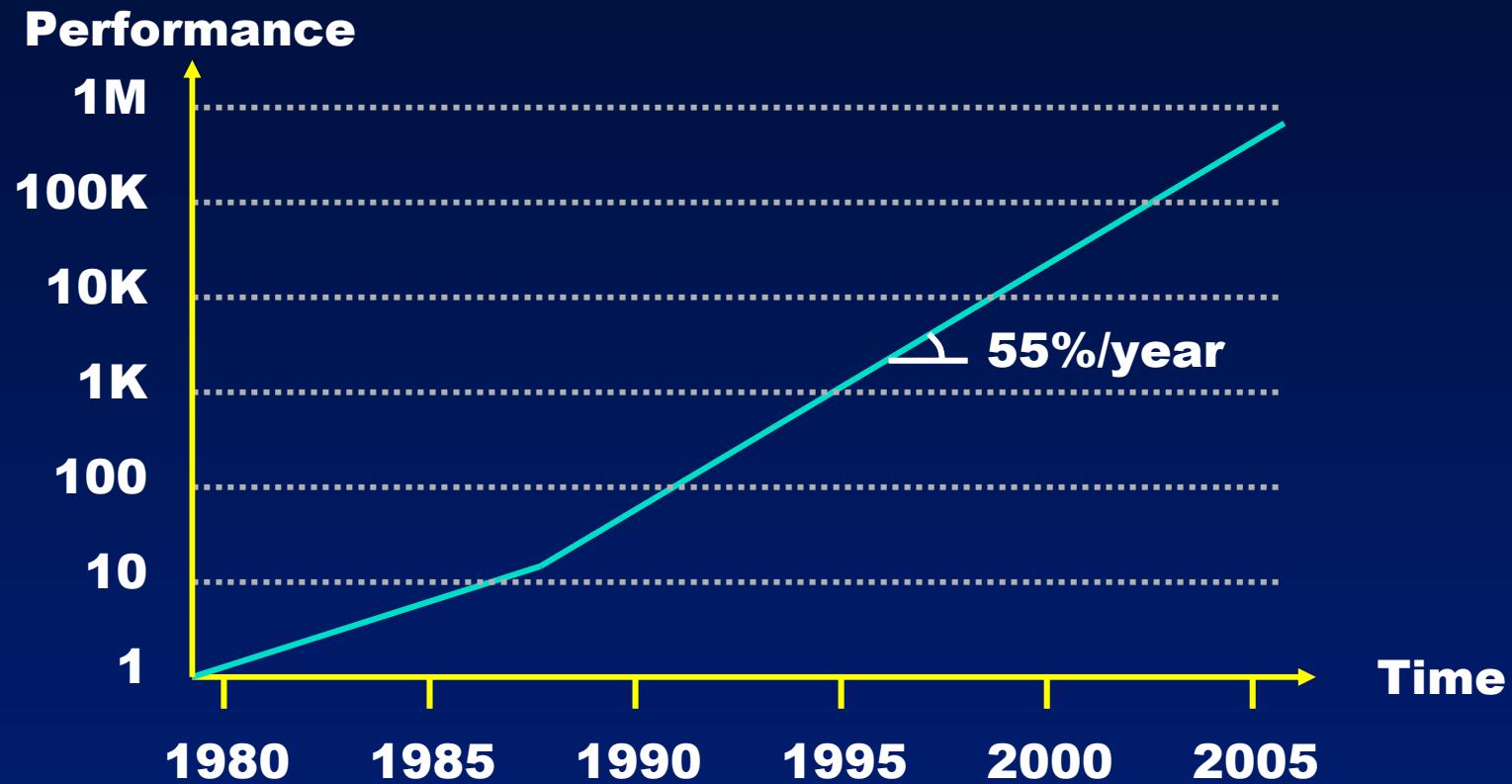
- ⇒ **425 MTOR**
    - ⇒ **100 nm**
    - ⇒ **1600 mm<sup>2</sup>**
    - ⇒ **30 GHz on chip**
    - ⇒ **without specific measures like individual transistor power-down: 3000 W, i.e. 3000 amps...**

- **1.8 GTOR in 2010**

# The future: high performance ( $\mu$ P)



# Processor performance



**Exponential growth for 3 decades!**

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

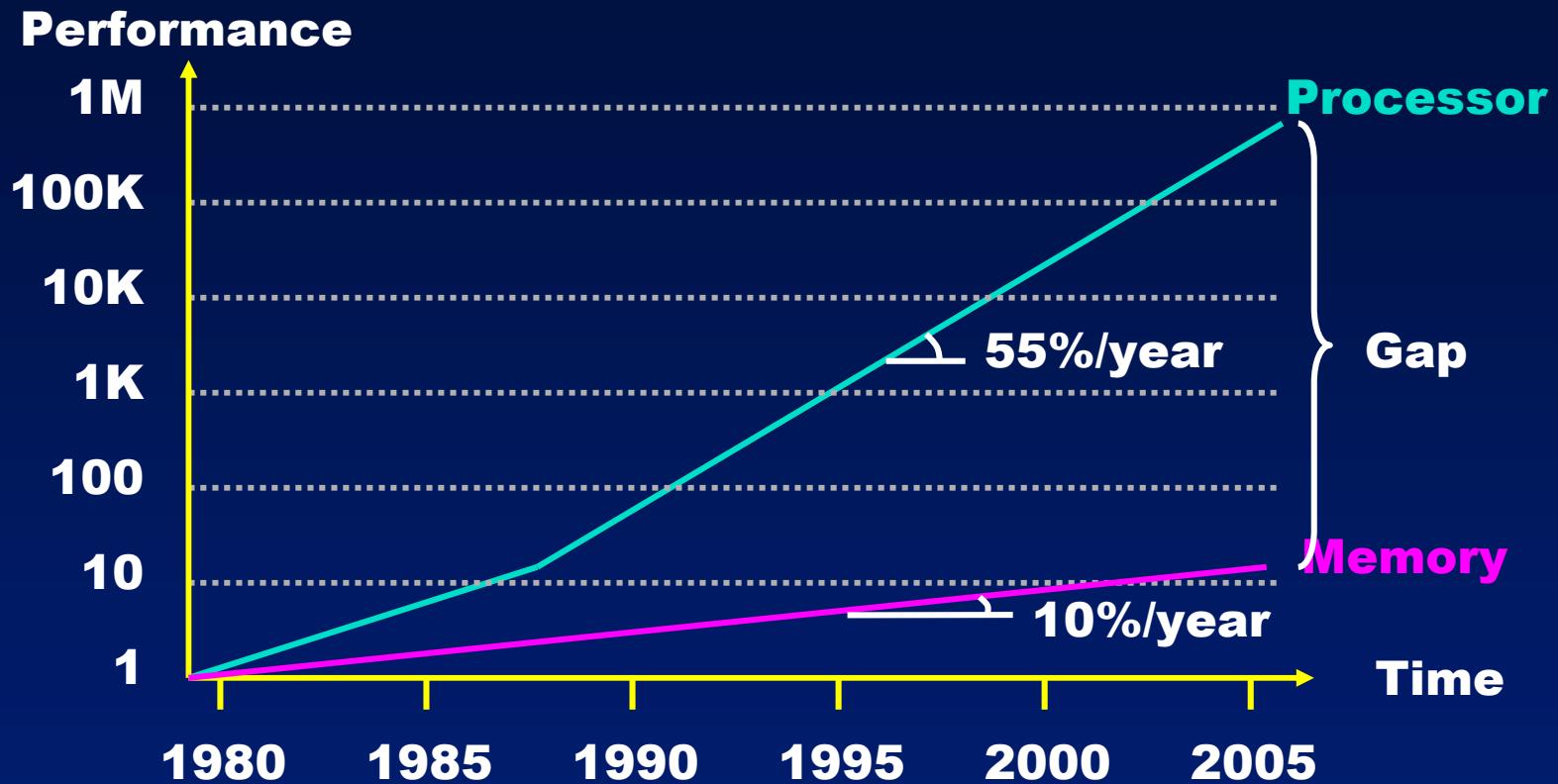
# Processor performance

- **Smaller line size**
  - More transistors => parallelism
    - ⇒ 1983: 1 instruction per 4 clock cycles
    - ⇒ 2002: 8 instructions per clock cycle
  - Smaller capacitors => faster
    - ⇒ 1983: 4 MHz
    - ⇒ 2002: 2800 MHz
  - Speed-up: 25000
- **Enables new applications**
  - UMTS with large rolled-up OLED screen enabling web downloadable services (e.g. virtual meetings)
- **Do we find applications that are demanding enough for next decade's processors?**

# The future: DRAM

Roadmap 2001			2003	2007	2011	2016	?	?
Roadmap 1998		1997	1999	2002	2005	2008	2011	2014
Roadmap 1995	1995	1998	2001	2004	2007	2010		
Number of bits per chip	64M	256M	1G	4G	16G	64G	256G	1T
Chip size (mm <sup>2</sup> )	190	280	400	560	790	1120	1580	2240

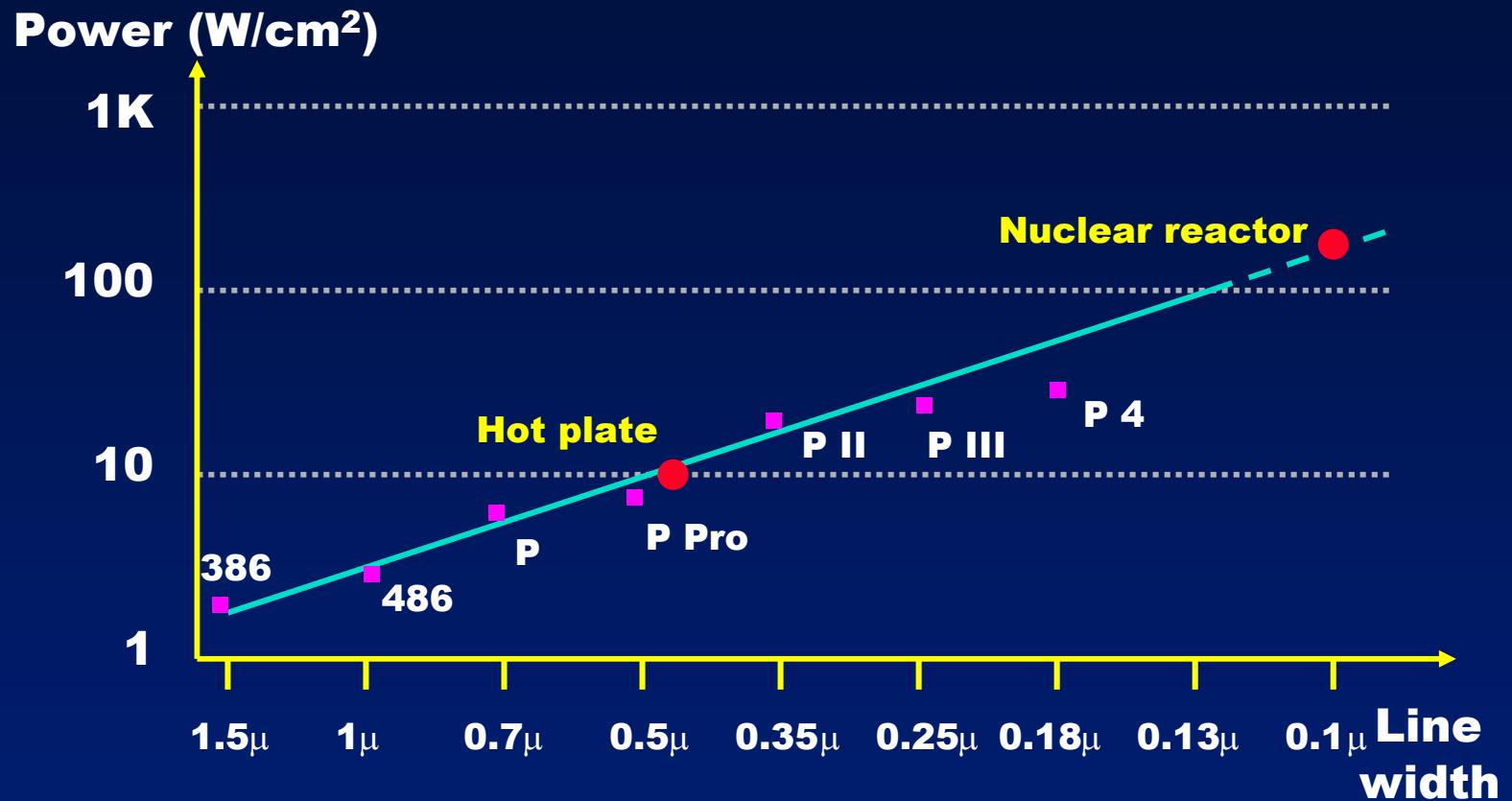
# Memory density



# Memory density

- Skills: center of gravity
  - USA: processors (Intel, Motorola, TI, ...)
  - Japan: memory (NEC, Toshiba, ...)
  - Future: IC = processor + memory  
Where???
- Memory density grows faster than needs
  - 1983: 512 Kbyte @ 64 Kbit/chip = 64 chips/PC
  - 2001: 256 Mbyte @ 512 Mbit/chip = 4 chips/PC
  - Compensated if you sell at least 16 times more PCs...
  - ... or if you find new applications (UMTS, car,...)
  - 2010: 4 Gbyte @ 64 Gbit/chip = 0.5 chip/PC
  - No need for such a large memory chip...
  - ... unless you find new applications (3D video...)

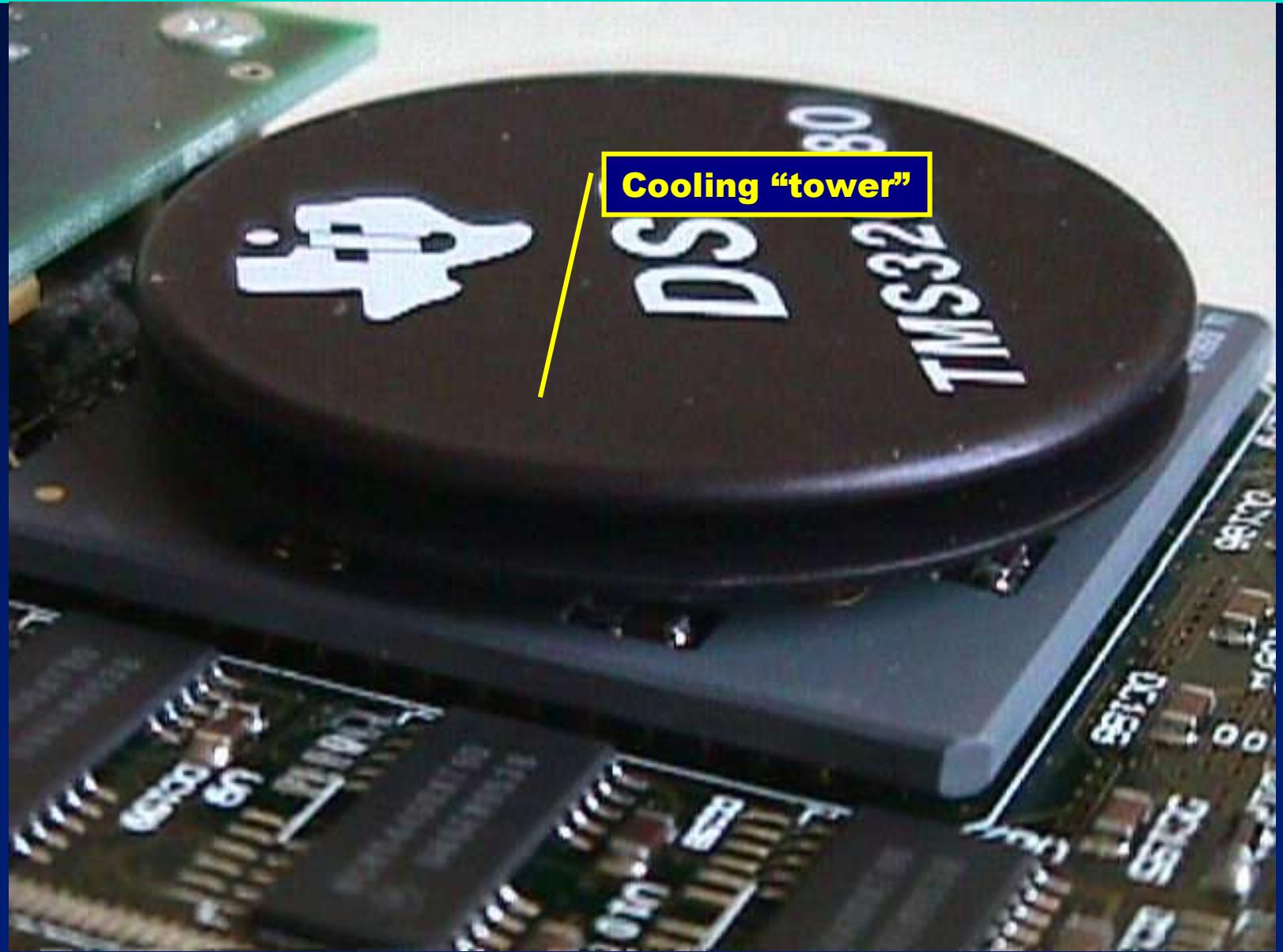
# Power consumption



**Processor architecture design driven by memory bottleneck & power problem!**

**Nevertheless, ‘cooling tower’ is necessary!**

# Power consumption



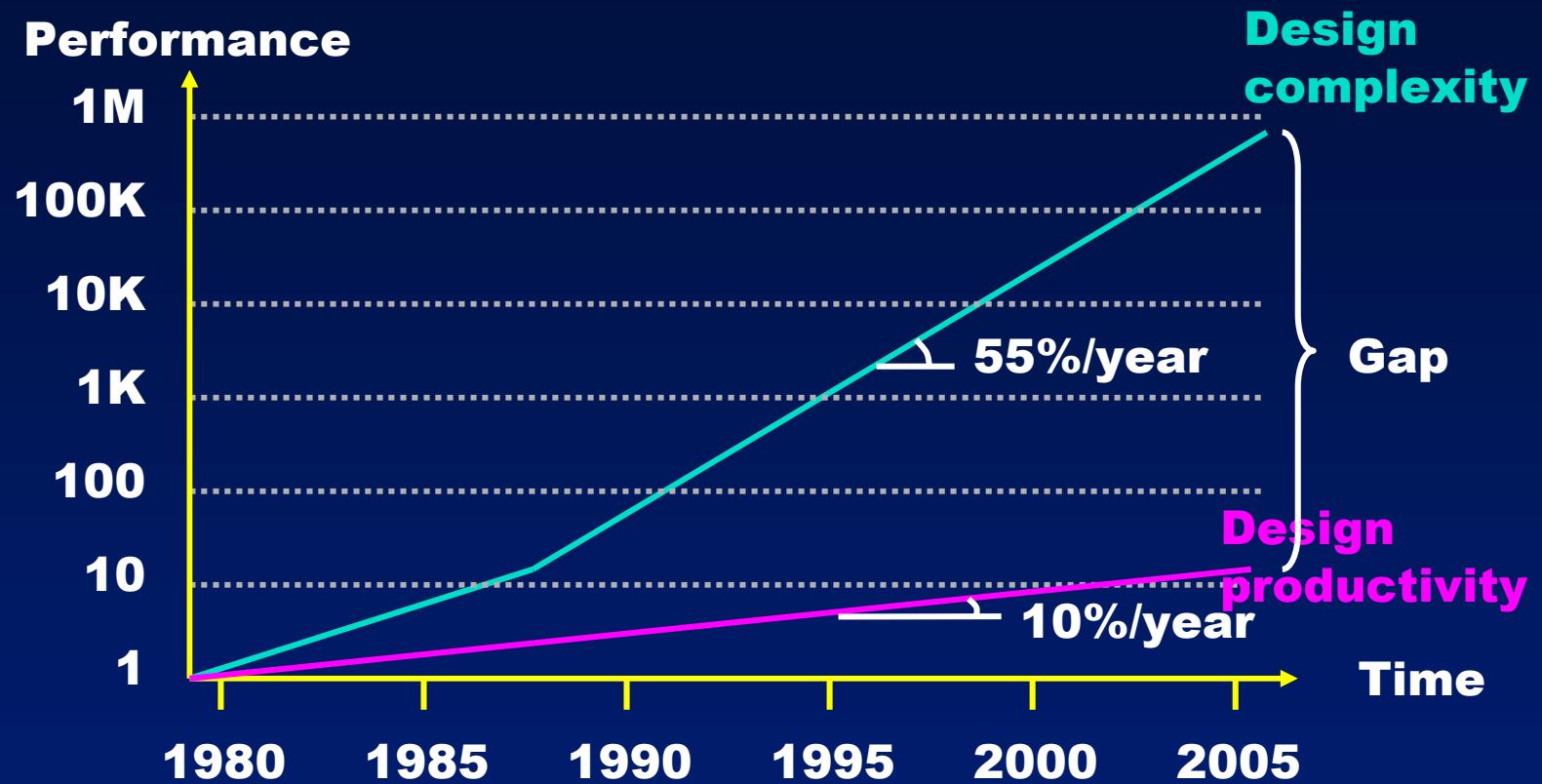
# Power consumption

- Let us do a calculation:
  - How long could a GSM using a Pentium 3 (hardly powerful enough...) last on a single battery charge?
  - Capacity of a battery:  
 $600 \text{ mAh} @ 4V = 2400 \text{ mWh}$
  - Power consumption Pentium 3: 45 W
  - One charge lasts for ... 3 minutes!!!
- Let us turn the computation upside down:
  - We want a GSM to last for 240 hours on a single charge.  
How much power may be consumed by the processor?
  - Capacity of a battery:  
 $600 \text{ mAh} @ 4V = 2400 \text{ mWh}$
  - Power consumption processor: 10 mW
  - Possible via specialization to the application:  
dedicated hardware...

## Summary on technological trends

- Technologically speaking, we can have the same exponential evolution for another decade
- This gives us at least 4 decades of exponential evolution, never seen in history
- End-user price stayed the same or even decreased
  - Since 30 years, the price for a brand new processor is 1000 USD
- So far for the good news...

# Design issues



**Unfortunately, Gordon Moore's law is also valid for the design complexity, which doubles every 18 months...**

**... and worse, design productivity doubles only every 10 years**

## Design issues

- We can build exponentially complex circuits, but we cannot design them
  - Design of Pentium 4: 8 years, during last 2 years with a team of 1000 persons
  - Who can afford this???

# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Hộ vi điều khiển 8051
  - Hộ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

## Giới thiệu về vi điều khiển

- **Vi điều khiển = CPU + Bộ nhớ + các khối ghép nối ngoại vi + các khối chức năng**
  - EEPROM**
  - RAM**
  - ADC/DAC**
  - Timer**
  - Bộ tạo xung nhịp**
  - PWM**
  - UART**
  - USB**
  - ...**

# Vi xử lý vs. Vi điều khiển

- **Vi xử lý là một CPU được sử dụng trong các máy tính.**
- **Các bộ điều khiển và các bộ vi xử lý có ba điểm khác nhau chính sau: kiến trúc phần cứng, phạm vi ứng dụng, và đặc điểm tập lệnh.**
- **Kiến trúc phần cứng:** Một bộ vi xử lý chỉ là một CPU đơn lẻ trong khi một bộ vi điều khiển là một IC chứa CPU và các mạch ngoại vi chính của một máy tính hoàn chỉnh (như RAM, ROM, giao diện nối tiếp, giao diện song song, bộ định thời, mạch xử lý ngắn)
- **Ứng dụng:** Vi xử lý thường được dùng như một CPU trong các máy tính trong khi các bộ điều khiển được sử dụng trong các thiết bị nhỏ để điều khiển các hoạt động.

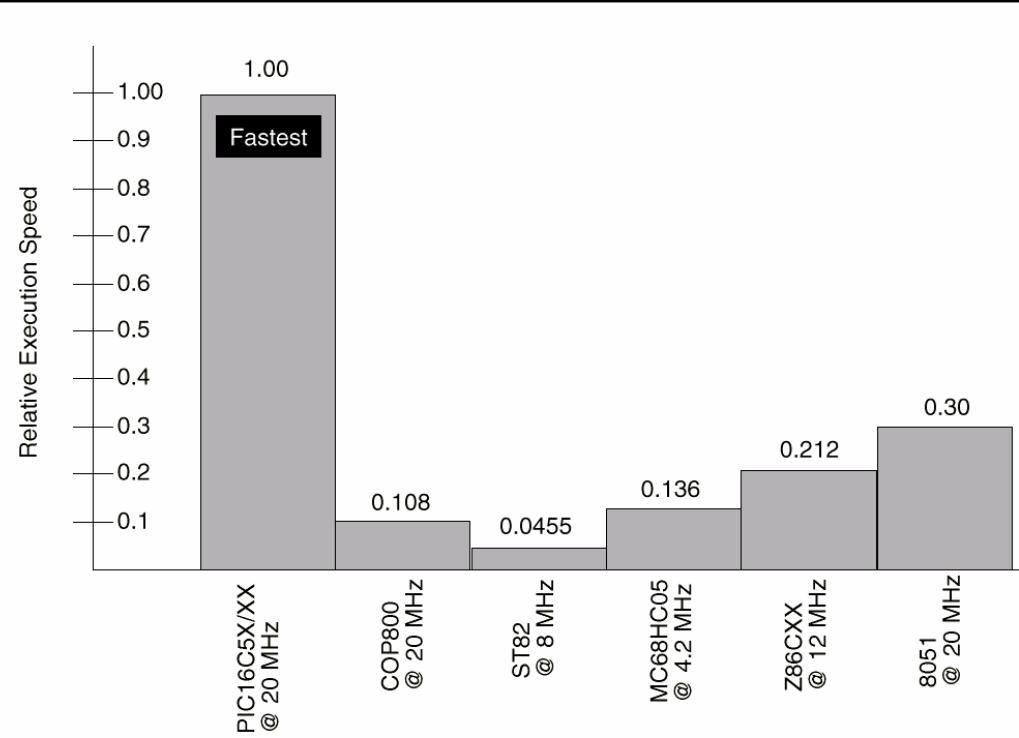
# Vì xử lý vs. Vì điều khiển

- **Tập lệnh:**
  - **Tập lệnh của vi xử lý thiên về xử lý dữ liệu.**
    - ⇒ **Chúng có thể làm việc với 4 bit, byte, word, thậm chí double word.**
    - ⇒ **Chế độ địa chỉ cho phép truy nhập các mảng dữ liệu lớn bằng cách sử dụng con trỏ và địa chỉ offset.**
  - **Tập lệnh của vi điều khiển dùng để điều khiển các truy nhập vào và ra.**
    - ⇒ **Chúng có các lệnh cho phép thiết lập và xoá các bit riêng rẽ và thực hiện các thao tác với bit.**
    - ⇒ **Có các lệnh cho các hoạt động vào/ra, đo thời gian các sự kiện, cho phép và thiết lập các mức độ ưu tiên cho các ngắt ngoài.**
- **Khả năng xử lý của vi điều khiển thấp hơn rất nhiều so với vi xử lý.**

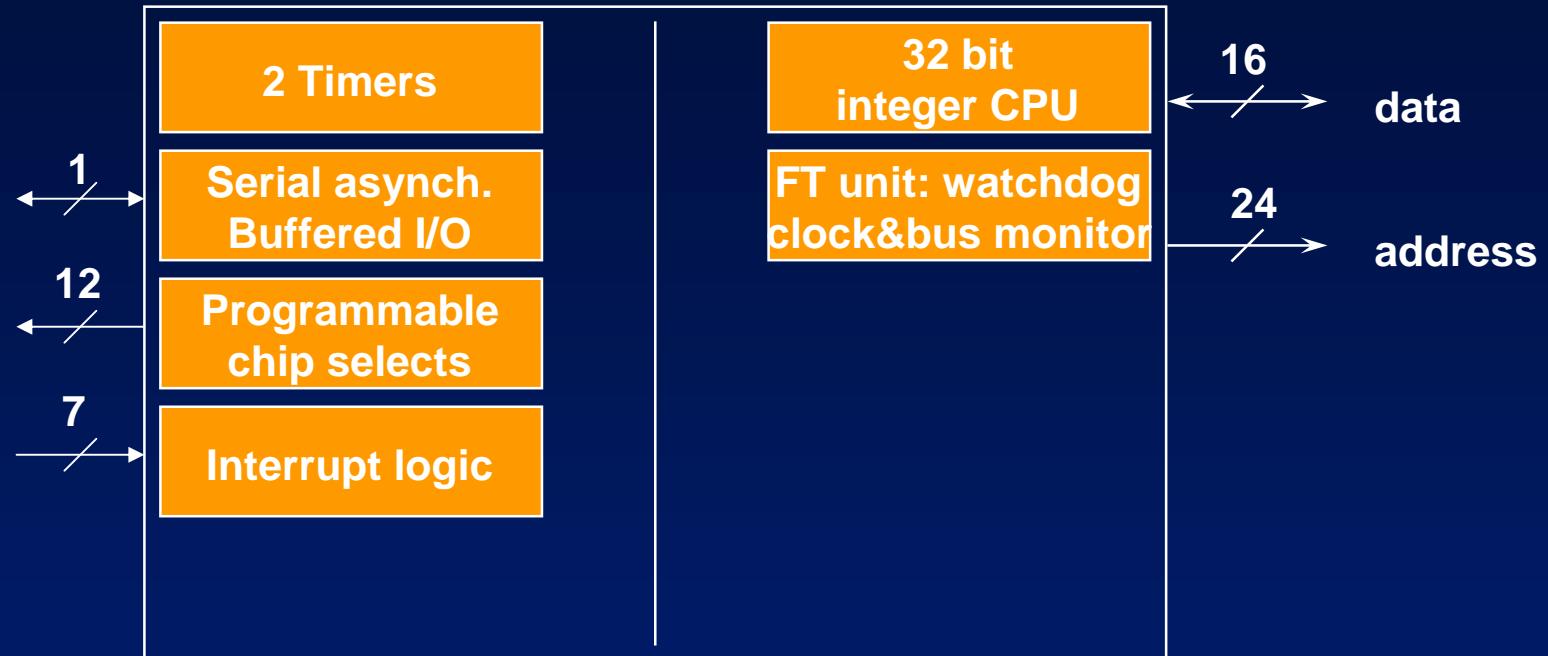
# Microcontroller facts

- **99% processor market**
- **Shipments- > 16 Billion in 2000, 8 bit > 1/2 market**
- **Major Players: Microchip 16Fxx, Intel 8051, Motorola MC68HC05, National COP800, SGS/Thomson ST62, Zilog Z86Cxx**

FIGURE 2: EXECUTION SPEED COMPARISON

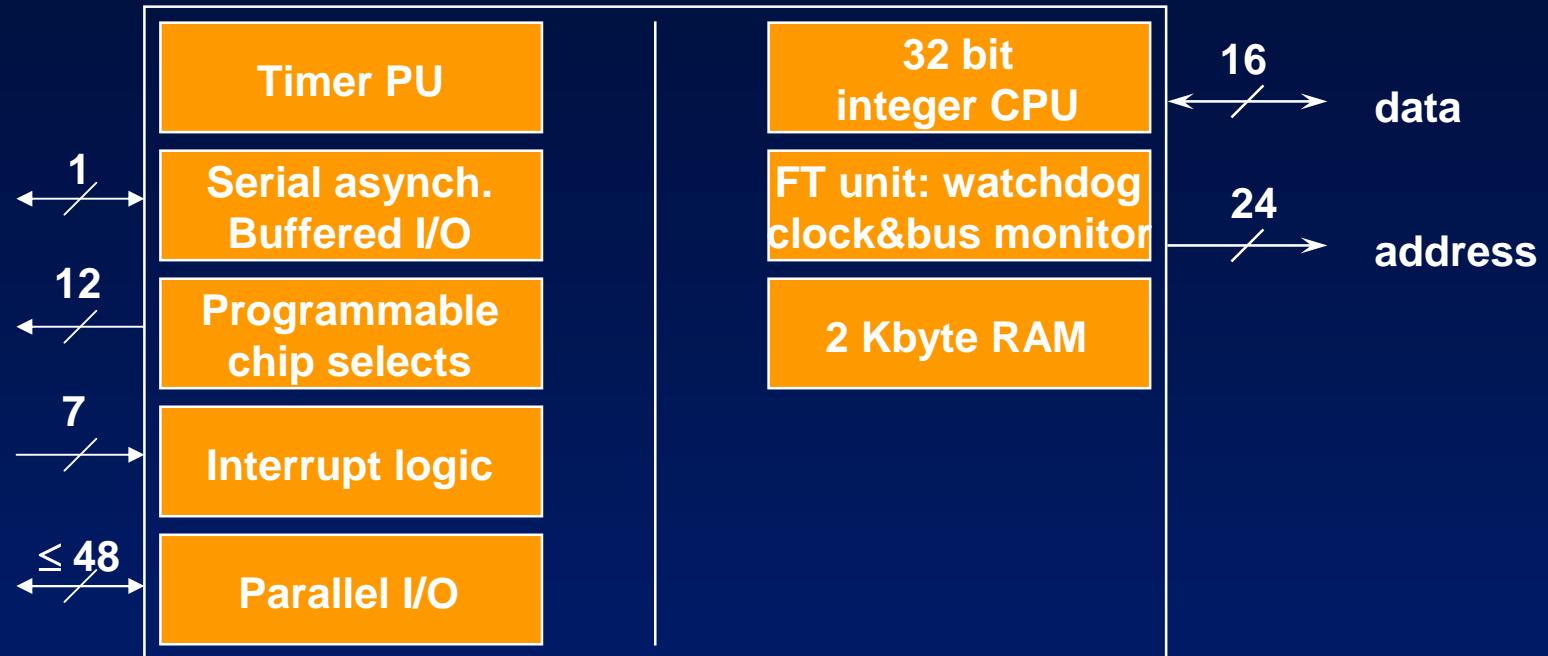


# Motorola MC68331



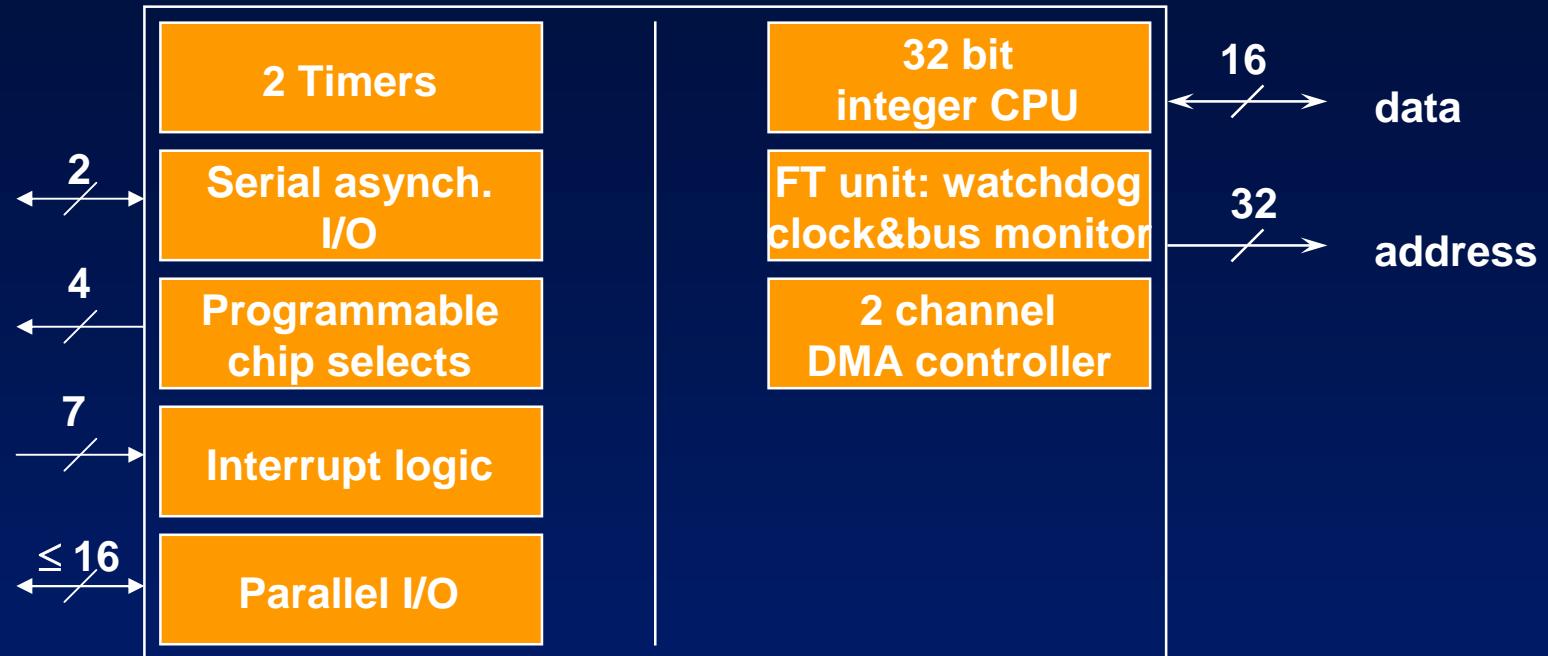
- **Clock frequency: > 16 MHz**
- **MC683xx: modular microcontroller unit: MC68000 core plus customized peripherals**

# Motorola MC68332



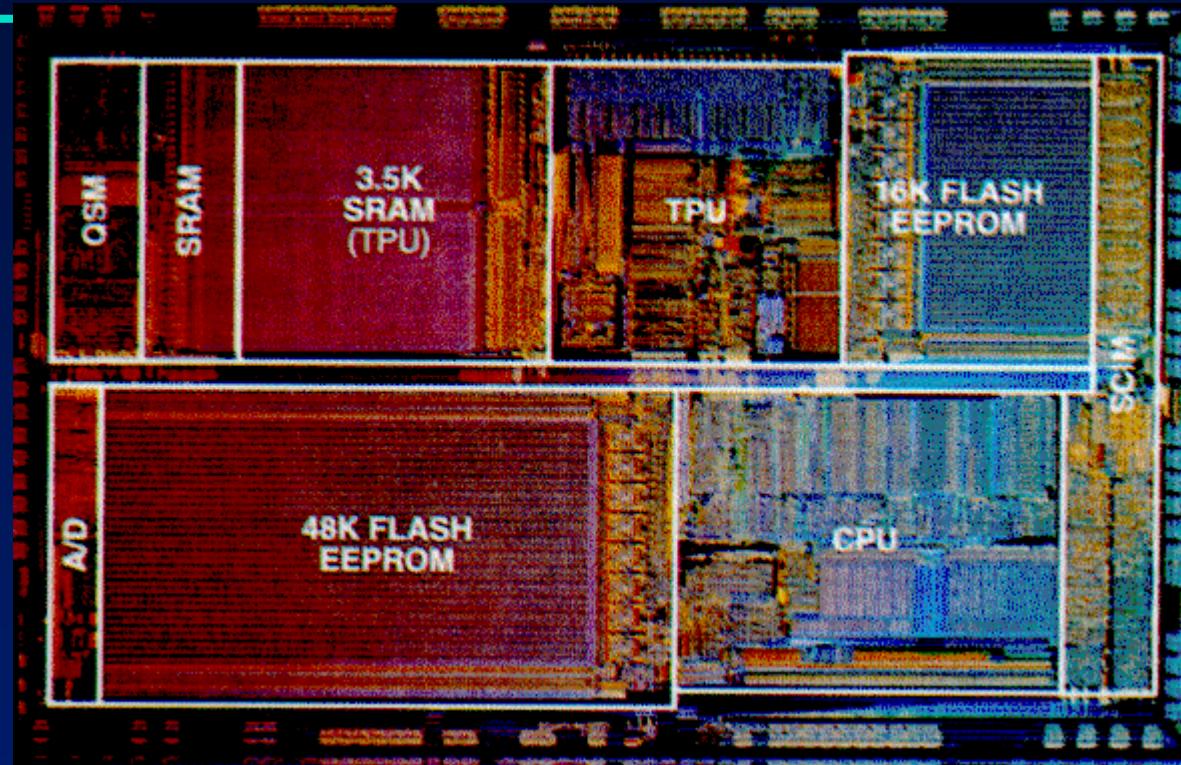
- **Clock frequency:** > 16 MHz
- **MC683xx:** modular microcontroller unit: MC68000 core plus customized peripherals

# Motorola MC68340



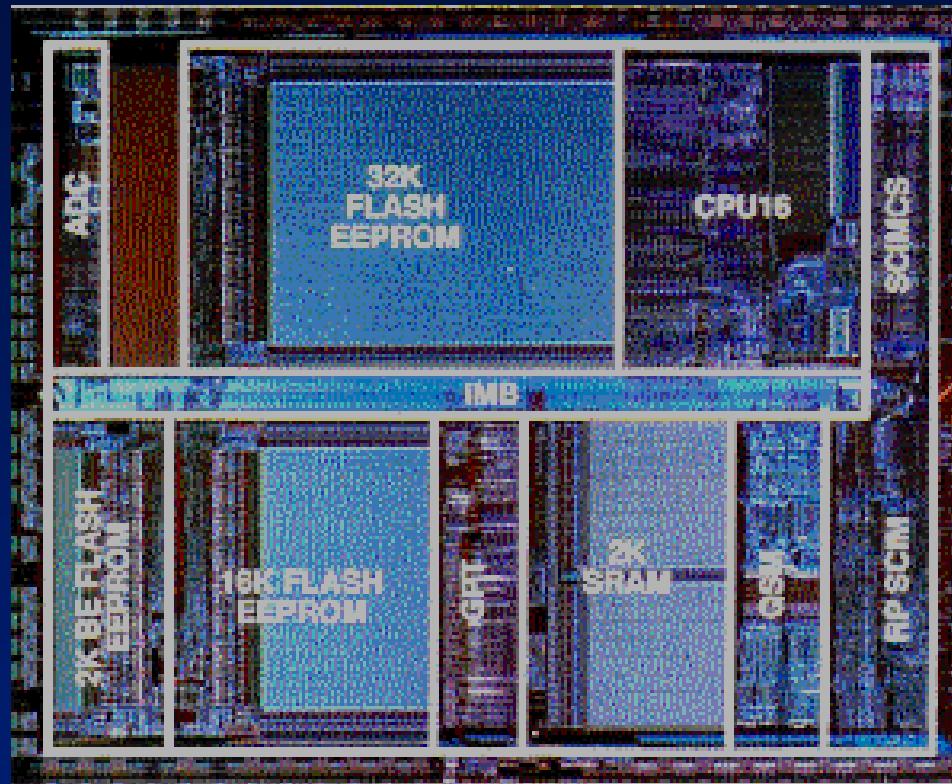
- **Clock frequency:** > 16 MHz
- **MC683xx:** modular microcontroller unit: MC68000 core plus customized peripherals

# Motorola MC68F333



- **64 Kbyte on chip flash EEPROM**
- **68020 processor**
- **4 Kbyte SRAM**
- **8 channel 10-bit ADC**
- **16 channel 16-bit timer**
- **several interfaces**
- **Introduced in 1994**

# Motorola MC68HC16



- **16-bit microprocessor**
- **Introduced in 1994**

[www.freescale.com](http://www.freescale.com)

# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Hộ vi điều khiển 8051**
  - Hộ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

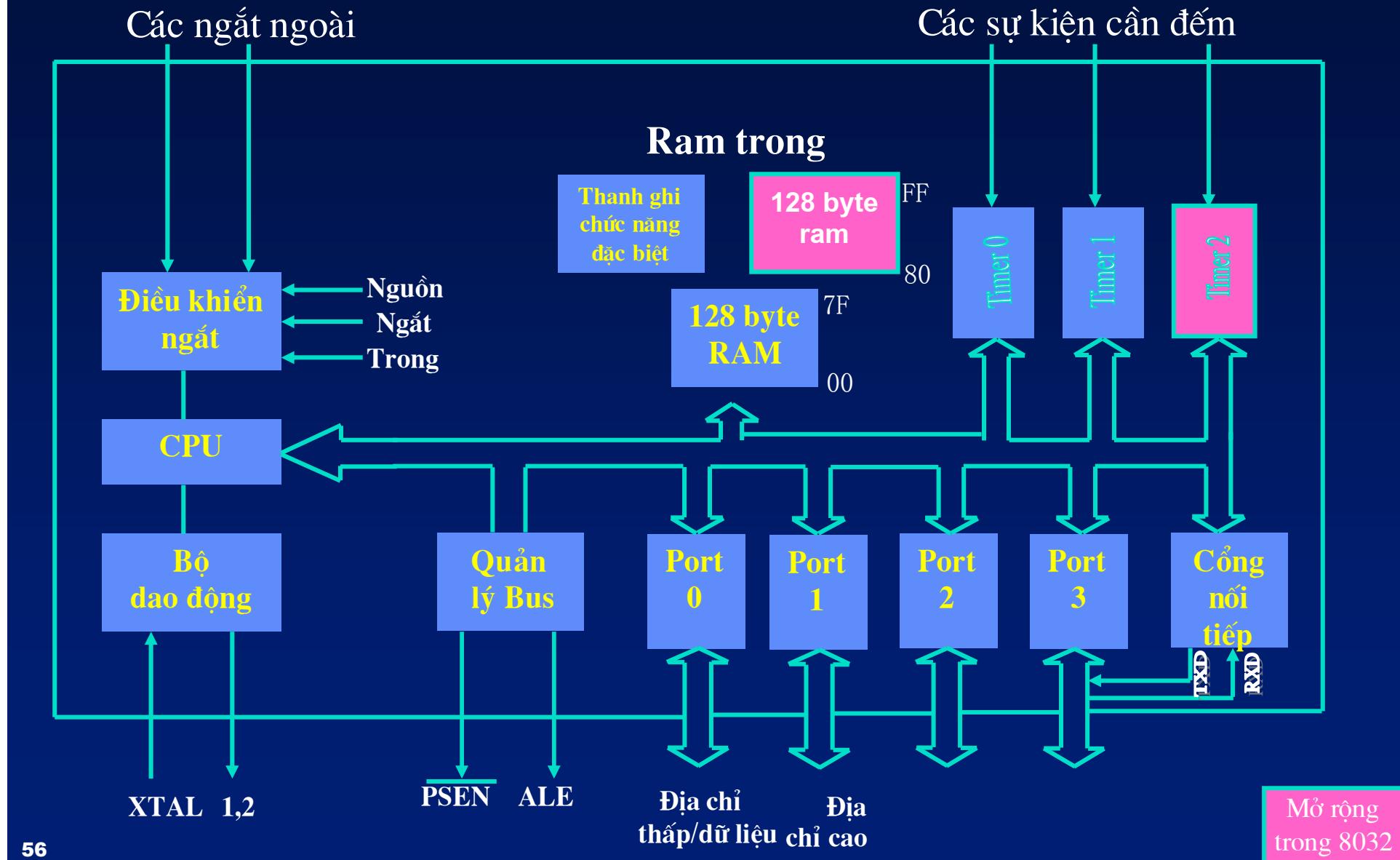
# Họ vi điều khiển 8051

- **Hiện nay có hơn 40 công ty sản xuất các loại vi điều khiển khác nhau của họ 8051.**
- **Một số công ty có trên 40 version 8051.**
- **Các CORE 8051 có thể được tổ hợp trong các FPGA hay ASIC.**
- **Trên 100 triệu vi điều khiển 8051 được bán ra mỗi năm.**
- **Họ 8051 gặt hái được rất nhiều thành công và nó cũng trực tiếp ảnh hưởng đến cấu trúc của các họ vi điều khiển hiện nay.**

# MCS-51

- 8051 thuộc họ vi điều khiển MCS-51.
- MCS-51 được phát triển bởi Intel và các nhà sản xuất khác (như Siemens, Philips) là các nhà cung cấp đúng thứ hai của họ này.
- **Tóm tắt một số đặc điểm chính của họ 8051:**
  - 4K bytes ROM trong
  - 128 bytes RAM trong
  - 4 cổng I/O 8-bit
  - 2 bộ định thời 16 bit
  - Giao diện nối tiếp
  - Quản lý được 64K bộ nhớ code bên ngoài
  - Quản lý được 64K bộ nhớ dữ liệu bên ngoài

# Hệ vi điều khiển 8051



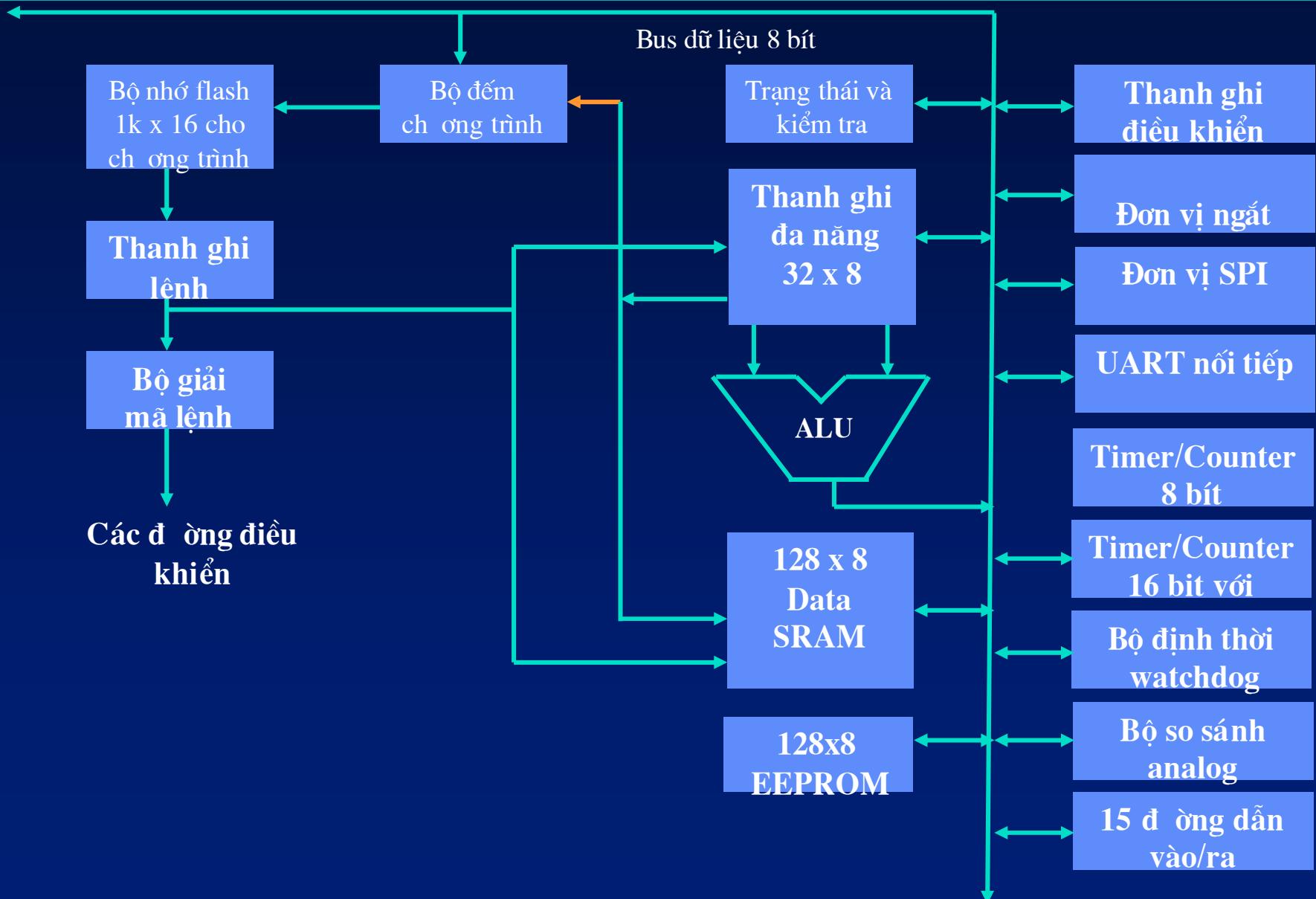
## Tham khảo

- **Hộ vi điều khiển 8051**
- [www.atmel.com/products/8051/](http://www.atmel.com/products/8051/)
- <http://chaokhun.kmitl.ac.th/~kswichit/>
- <http://www.sunrom.com/>

# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Hộ vi điều khiển 8051
  - Hộ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# Hệ vi điều khiển AVR



# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Hộ vi điều khiển 8051
  - Hộ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

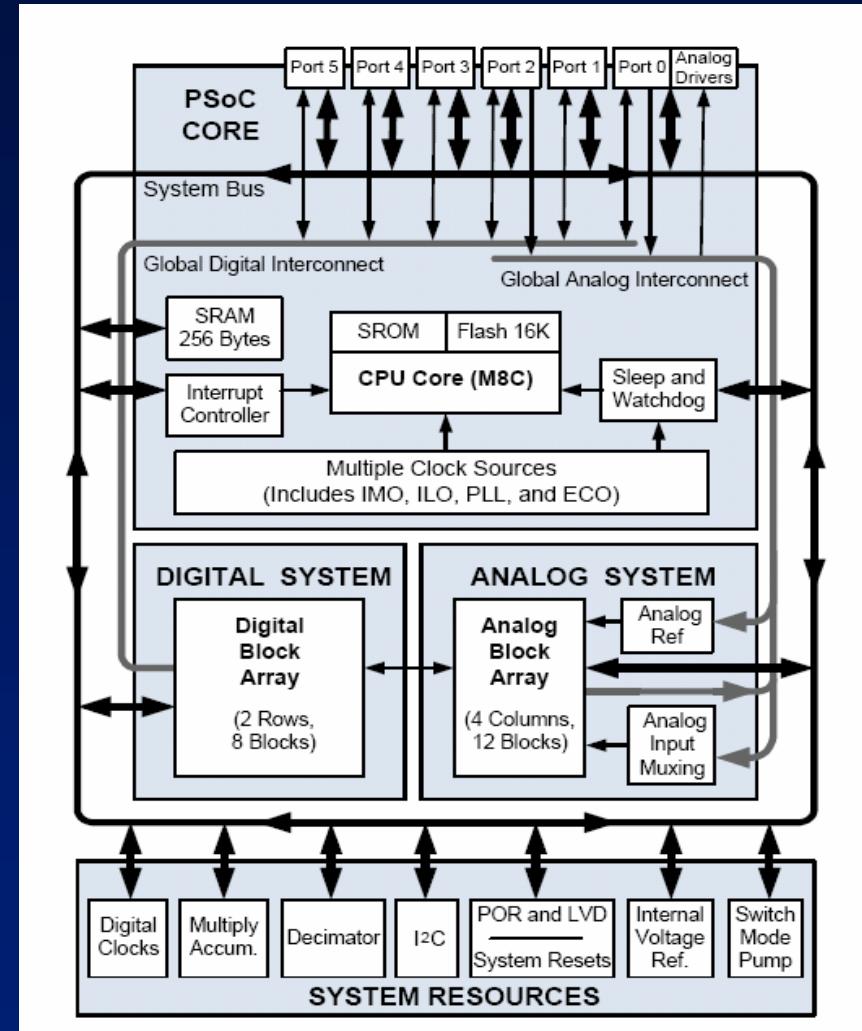
# Giới thiệu chung về PSoC

- **PSoC là gì?**

- Do hãng Cypress sản xuất**
- Một loại công nghệ IC mới phát triển trong vài năm gần đây.**
- Khả năng tích hợp đồng các loại linh kiện số và tương tự để tạo ra các khối số hoặc tương tự với chức năng tùy thuộc người dùng.**
- Kết hợp với một vi điều khiển trung tâm.**

# Cấu trúc PSoC

- **Lõi PSoC**
  - **Vi xử lý 8bit, 24MHz, 4MIPS**
  - **Flash ROM (từ 16K)**
  - **RAM (128b-2kb)**
  - **Bộ điều khiển ngắt**
  - **Bus**
- **Các khối số**
  - **Flip-Flop, cỗng logic**
- **Các khối tương tự**
  - **Các bộ khuếch đại thuật toán**
  - **Điện trở**
  - **Tụ điện điều khiển đóng ngắt**
- **Tài nguyên hệ thống**
- **Đặc biệt: Debugger core**



# Ứng dụng

## Xử lý tín hiệu

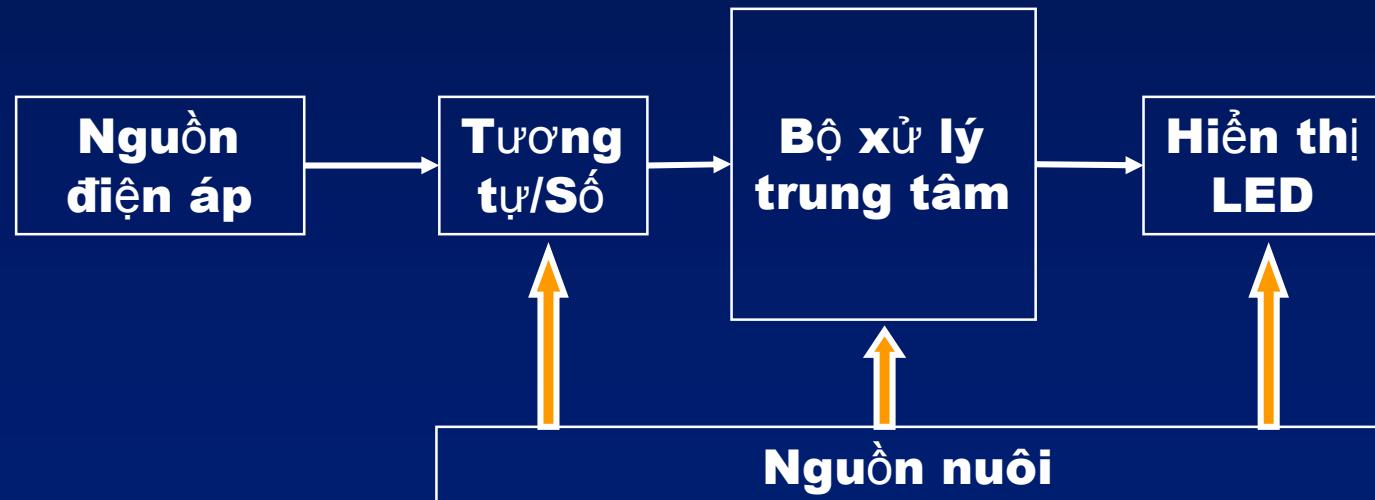
- Phát hiện sóng Sin
- Đo điện áp
- Đo tần số
- Điều chế/giải điều chế
- Lọc
- Nhận tương tự
- FSK

## Giao tiếp với cảm biến

- Điện trở nhiệt
- Cặp nhiệt
- Đo tín hiệu hồng ngoại thụ động
- Thu siêu âm
- Đo áp suất

# So sánh thiết kế giữa VXL, VĐK, PSoC

- **Ứng dụng:** Mạch đo điện áp 0-5V, hiển thị giá trị ra LED
  - Thiết kế phần cứng
  - Thiết kế phần mềm



# So sánh

	<b>Vi xử lý</b>	<b>Vi điều khiển</b>	<b>PSoC</b>
<b>Độ phức tạp phần cứng</b>	<b>Cao</b>	<b>Trung bình</b>	<b>Thấp</b>
<b>Giá thành</b>	<b>Cao</b>	<b>Trung bình</b>	<b>Trung bình</b>
<b>Độ ổn định</b>	<b>Thấp</b>	<b>Trung bình</b>	<b>Cao</b>
<b>Khả năng phát triển hệ thống</b>	<b>Khó</b>	<b>Trung bình</b>	<b>Dễ</b>

# Thiết kế hệ thống sử dụng PSoC

- Chuẩn bị
  - IC
  - Chương trình thiết kế: PSoC Designer
  - Chương trình nạp và mạch nạp
- Các bước thiết kế PSoC sử dụng PSoC designer
  - Thiết kế “cứng”
    - ⇒ Lựa chọn “linh kiện” để đưa vào các khối số, tương tự
    - ⇒ Đưa các linh kiện vào các khối và kết nối
    - ⇒ Thiết lập các thông số
  - Lập trình
    - ⇒ Kích hoạt các linh kiện: Component\_Start()
    - ⇒ Viết các thuật toán sử dụng linh kiện
    - ⇒ Vòng lặp vô tận

## Một số linh kiện

- **ADC (6-14 bit)**
- **DAC (6-9 bit)**
- **Khuếch đại**
- **DTMF**
- **Bộ lọc (thông thấp, thông dải)**
- **Bộ đếm, bộ định thời (8-24 bit)**
- **Đòn kẽm: số+tương tự**
- **Các linh kiện số: I2C, UART, giao tiếp LCD, mã CRC, mã giả ngẫu nhiên, giao tiếp chuẩn thu /phát hòng ngoại**

# Chọn linh kiện

**Linh kiện trong hệ thống**

**Tài nguyên hệ thống**

**Chọn linh kiện**

**Hướng dẫn sử dụng linh kiện**

**Features and Overview**

- 8-, 16-, 24- or 32-bit general purpose counters use one, two, three or four PSoC blocks, respectively
- Source clock rates up to 48 MHz
- Automatic reload of period on terminal count
- Programmable pulse width
- Input enables/disables continuous counter operation
- Interrupt option on compare output or terminal count

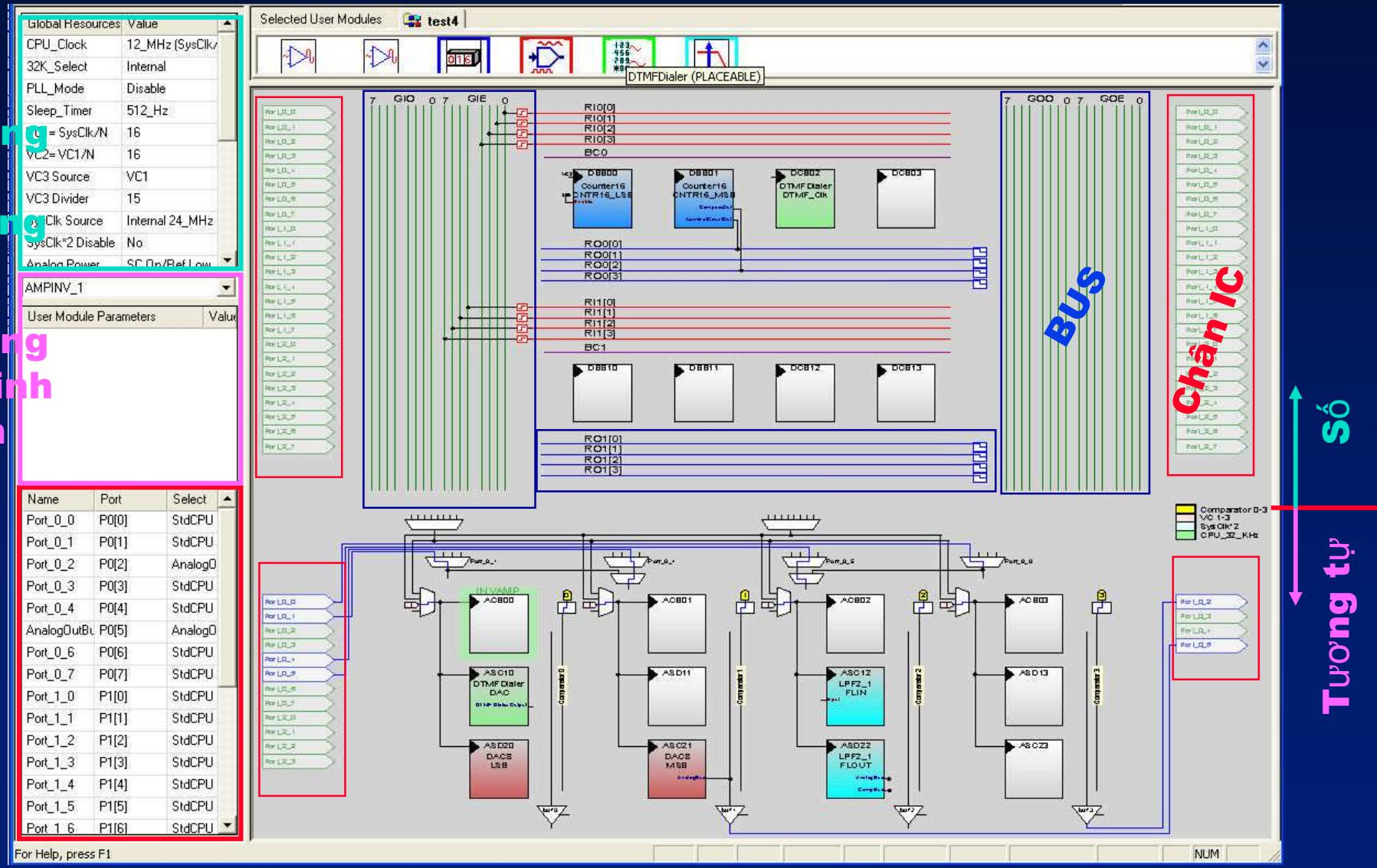
The 8-, 16-, 24- and 32-bit Counter User Modules provide a down counter with a programmable period and pulse width. The clock and enable signals can be selected from any system time base or external source. Once started, the counter operates continuously and reloads its internal value from the period register upon reaching terminal count. During each clock cycle, the counter compares the current count to the value stored in the compare register. Each clock cycle, the Counter tests the count against the value of the compare register for either a "less than" or "less than or equal to" condition. The comparator output provides a logic level that may be routed to pins and to other user modules. User modules designed for the CY8C27443B device family also permit the terminal count output to be used as an interrupt signal.

	Total	Used
Digital Blocks	8	1
Analog Blocks	12	0
RAM	256	0
ROM	16384	67
Decimator	1	0
I2C Controller	1	0

# Thiết kế

Thông  
số  
chung

Thông  
số linh  
kiện



# Lập trình

Quản lý  
theo dự  
án

**test4 [CY8C27443] - PSoC Designer - [main]**

File Edit View Project Config Build Debug Program Tools Window Help

test4 files

- Source Files
  - boot.asm
  - main.c
  - speech.c
- Headers
  - memory.inc
  - speech.h
- Library Source
  - counter16.asm
  - counter16int.asm
  - dac8.asm
  - psocconfig.asm
  - psocconfigtbl.asm
- Library Headers
  - counter16.h
  - counter16.inc
  - dac8.h
  - dac8.inc
  - globalparams.h
  - globalparams.inc
  - psocapi.h
  - psocapi.inc
  - psocgpoint.h
  - psocgpoint.inc
- External Headers
  - m8c.h
  - m8c.inc
  - m8ssc.inc
- Flashsecurity.txt

Files

```

// C main line
//-----
#include <m8c.h>          // part specific constants and macros
#include "PSoCAPI.h"        // PSoC API definitions for all User Modules
#include "speech.h"
#define PERIOD (unsigned int)25
#define SetLED() PRTODR |= 0x70    // PO[6]=1
#define ClrLED() PRTODR &= ~0x70   // PO[6]=0

unsigned int index=0;
char LEDcounter=0;
signed char cnt=0;
#pragma interrupt_handler Counter16_ISR
void Counter16_ISR(void)
{
    DAC8_WriteBlind(*(Speech+index));
//    DAC8_WriteStall(cnt);
    index++;
    if(index>=NSamples)
        index=0;
    cnt++;
    if (cnt>=127)
        cnt=-128;
    LEDcounter++;
    if (LEDcounter%2)
        SetLED();
    else
        ClrLED();
}
void main()
{
    Counter16_WritePeriod(PERIOD);           /* set the period to 1000 */
    Counter16_WriteCompareValue(PERIOD/2);    /* generate a 50 duty cycle */
}

```

Build Debug Find in Files 1 Find in Files 2 Results

For Help, press F1 Line 11, Column 1 NUI

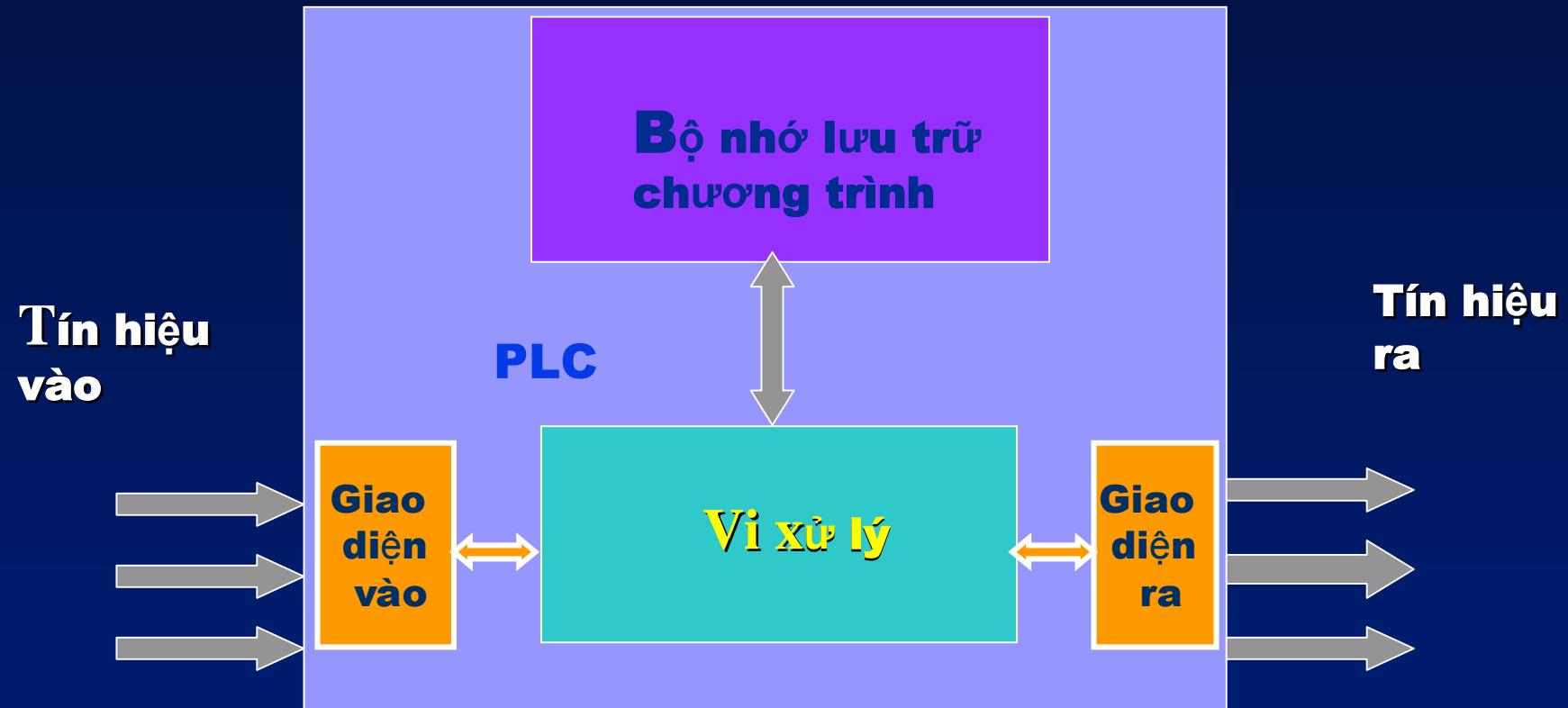
# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Hộ vi điều khiển 8051
  - Hộ vi điều khiển AVR
  - PSOC
  - Điều khiển trong công nghiệp PLC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# PLC (Programmable Logic Controller)

- **PLC = CPU + giao diện vào ra**
- **Đặc điểm của PLC:**
  - Giá thành hợp lý cho các ứng dụng điều khiển phức tạp
  - Chịu được rung động, nhiệt, ẩm, tiếng ồn và có độ bền cao
  - Có sẵn giao diện cho các thiết bị vào và thiết bị ra
  - Lập trình dễ dàng với ngôn ngữ lập trình đơn giản, chủ yếu giải quyết các phép toán logic và chuyển mạch
- **PLC được dùng chủ yếu để điều khiển trong công nghiệp:**
  - Điều khiển băng chuyền
  - Điều khiển thang máy
  - Điều khiển máy tự động: máy khoan, máy sấy ...
  - Điều khiển đèn giao thông
  - ...

# Cấu trúc của hệ thống PLC



# Cấu trúc bên ngoài



# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Hộ vi điều khiển 8051
  - Hộ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

## Trends for microcontrollers

- Standard CPU core surrounded by peripherals taken from a vast library
- Single architecture line is whole family
  - different memory & on-chip peripherals
  - for embedded applications
  - Deterministic behavior
    - ⇒ no caches, no virtual memory, but on-chip RAM
    - ⇒ no out-of-order execution
    - ⇒ delayed branch prediction

## Trends for microcontrollers

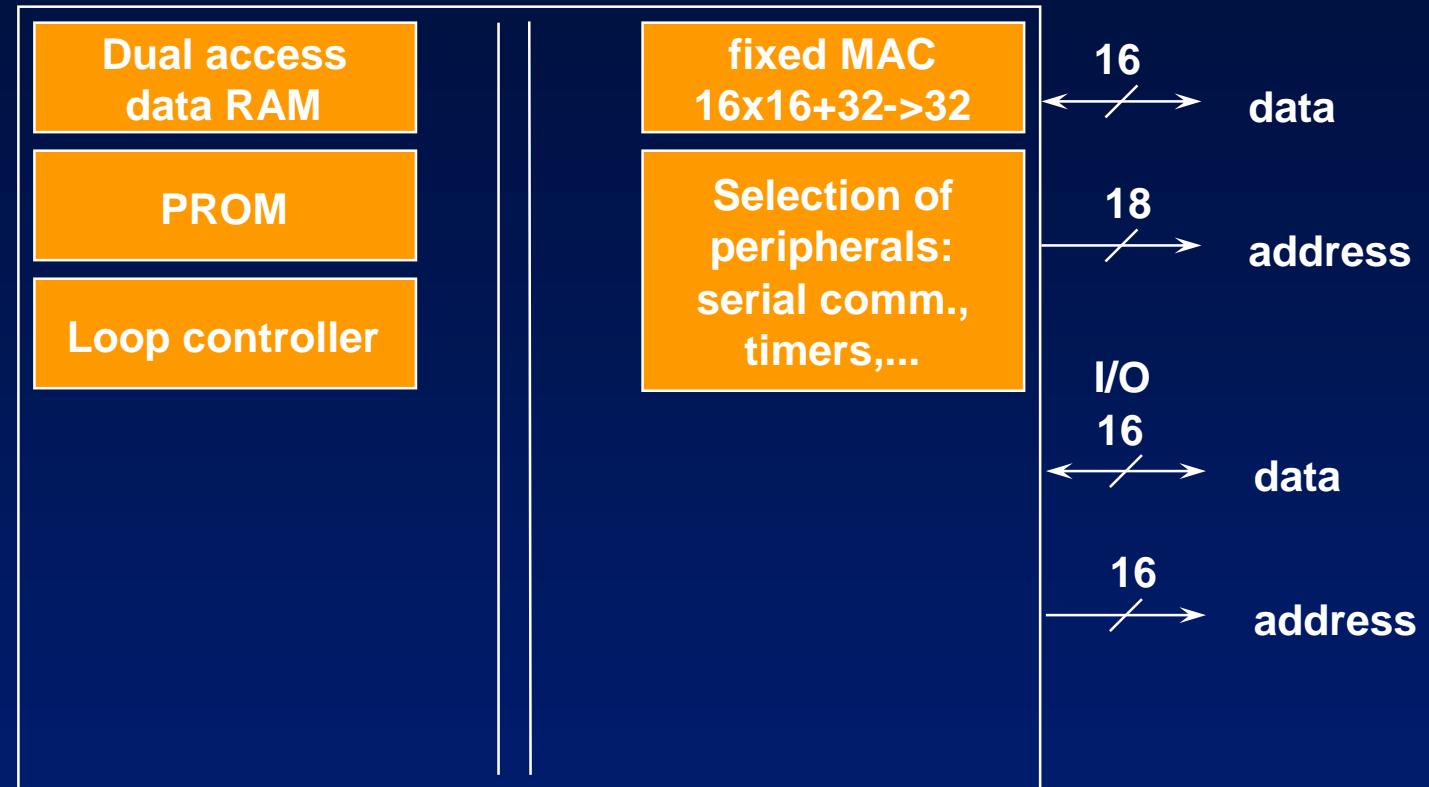
- Word length as small as possible
  - 4 bit: 2%
  - 8 bit: 36%
  - 16 bit: 25%
  - 32 bit: 34%
  - 64 bit: 3%
- Not pushing the limits of performance for cost reasons

# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Họ vi điều khiển 8051
  - Họ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# Texas Instruments TMS320C20x

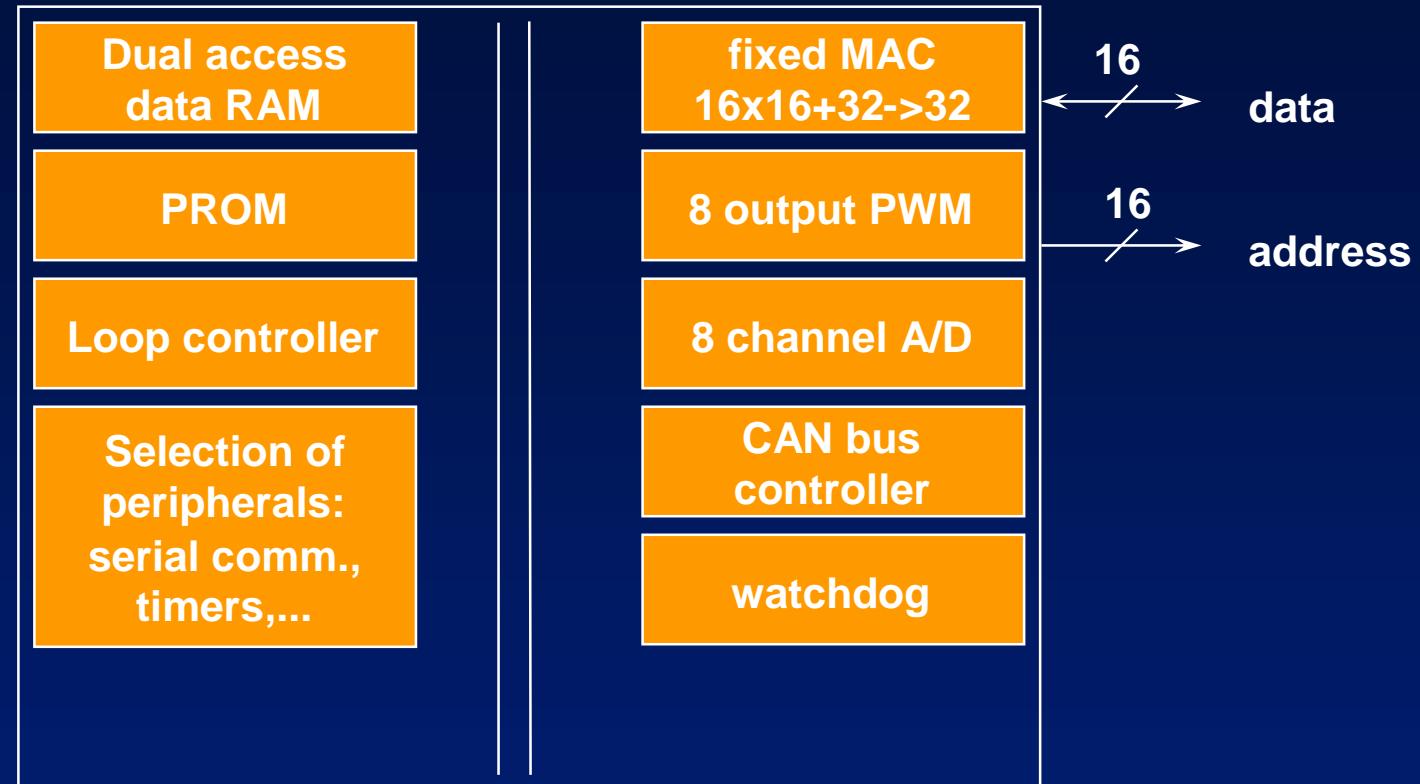
## Low end consumer Fixed Point



- Series continued; typical app.: Digital camera, feature-phones, disk drives, Point-of-Sales Terminal
- 40 MHz, 3.3-5V, 3LM
- Available as core

# Texas Instruments TMS320C24x

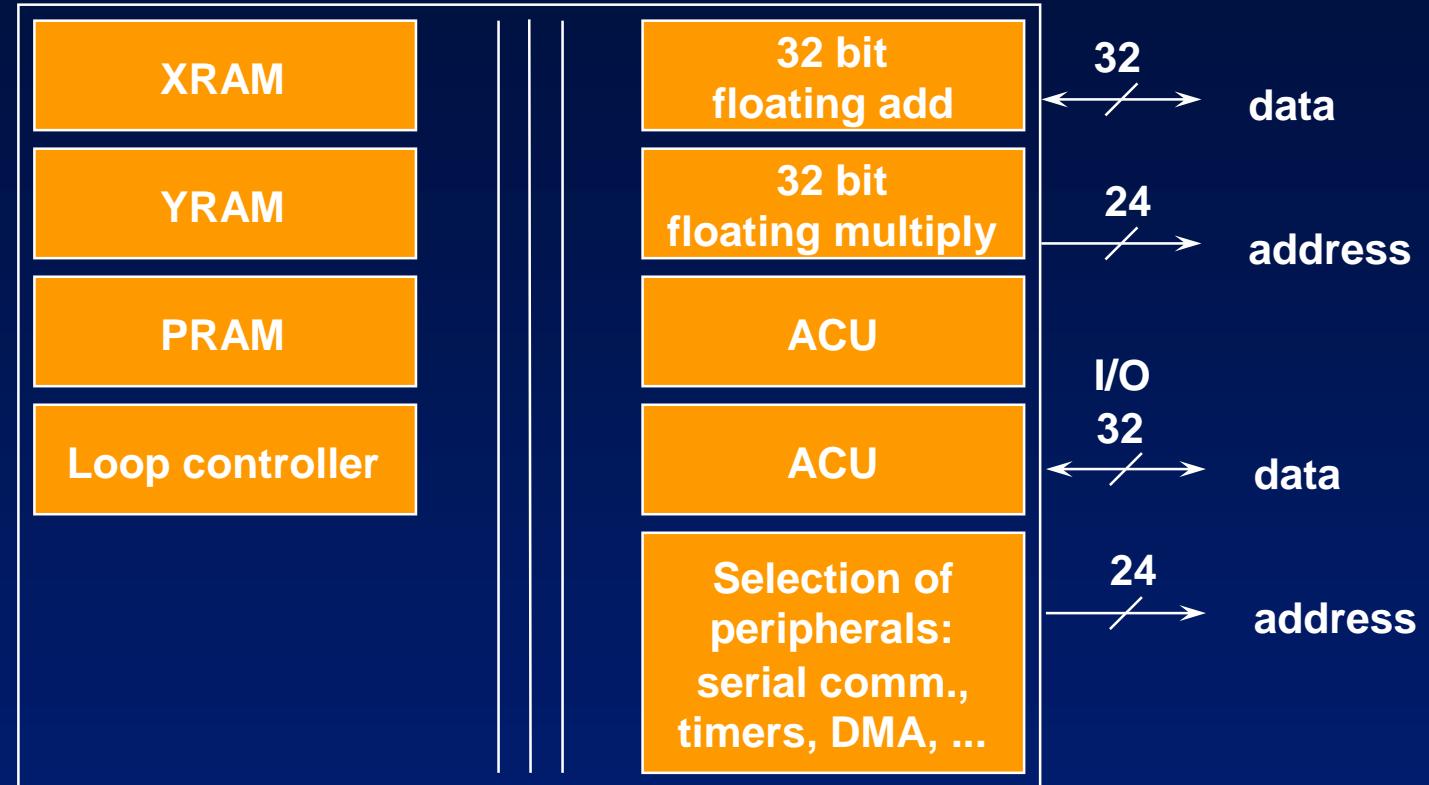
## Low end consumer Fixed Point



- Series continued; typical app.: electrical motor control
- 50 MHz, 5V

# Texas Instruments TMS320C3x

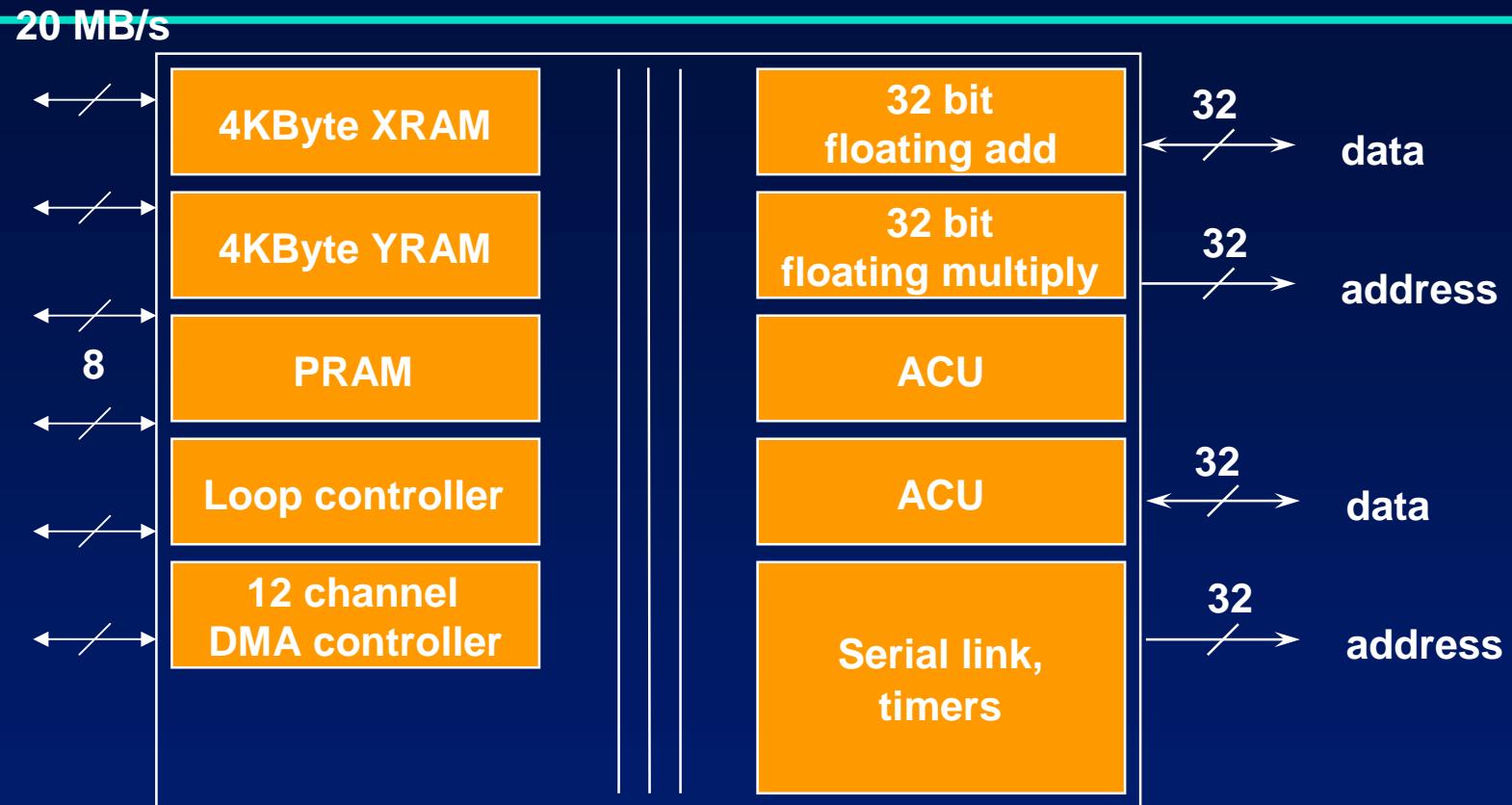
## Floating Point



- **Series discontinued; typical app.: speech, audio**
- **60 MHz, 3.3-5V, 144 pin**
- **Super scalar**

# Texas Instruments TMS320C4x

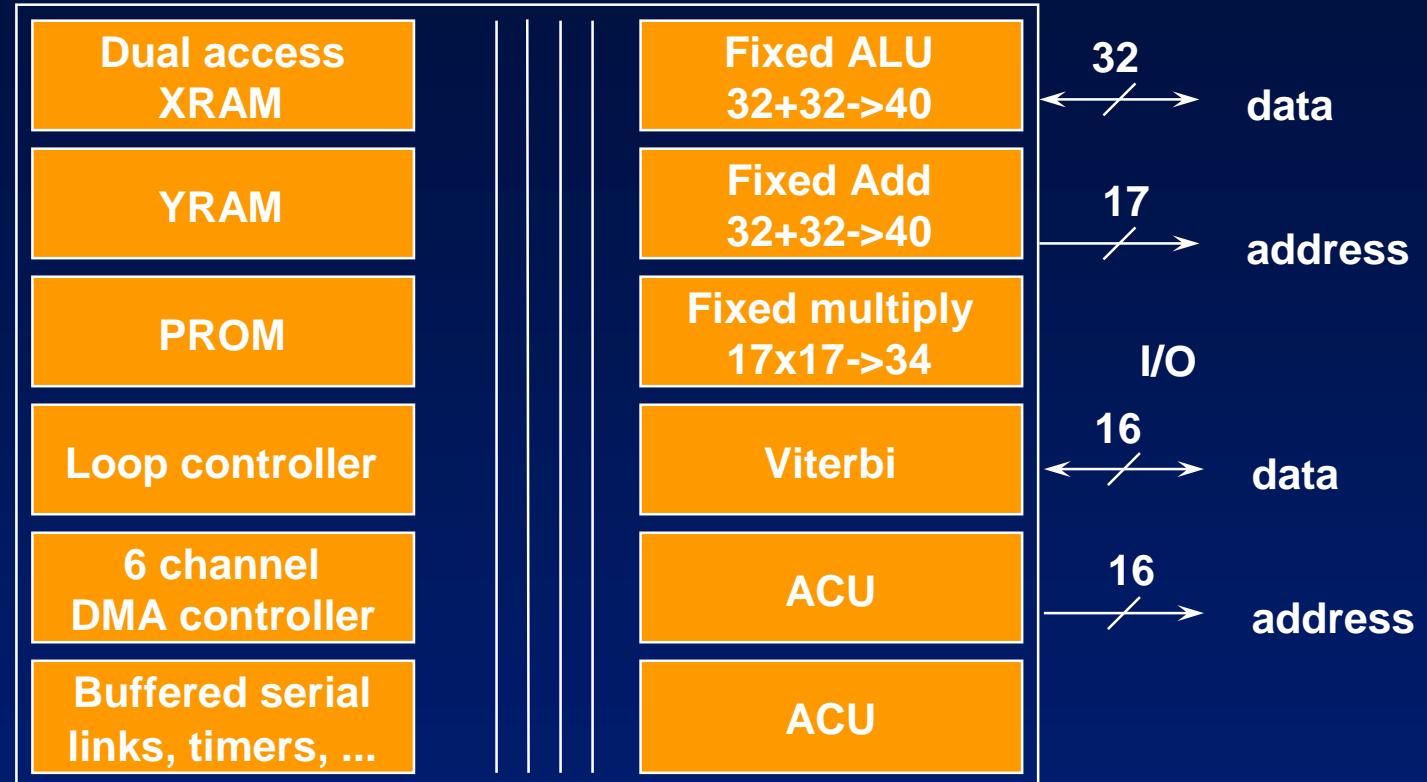
## Floating Point Message Passing



- Series discontinued; typical app.: prototyping, radar
- 60 MHz, 5V, 325 pin
- Super scalar; message passing multiprocessor

# Texas Instruments TMS320C54xx

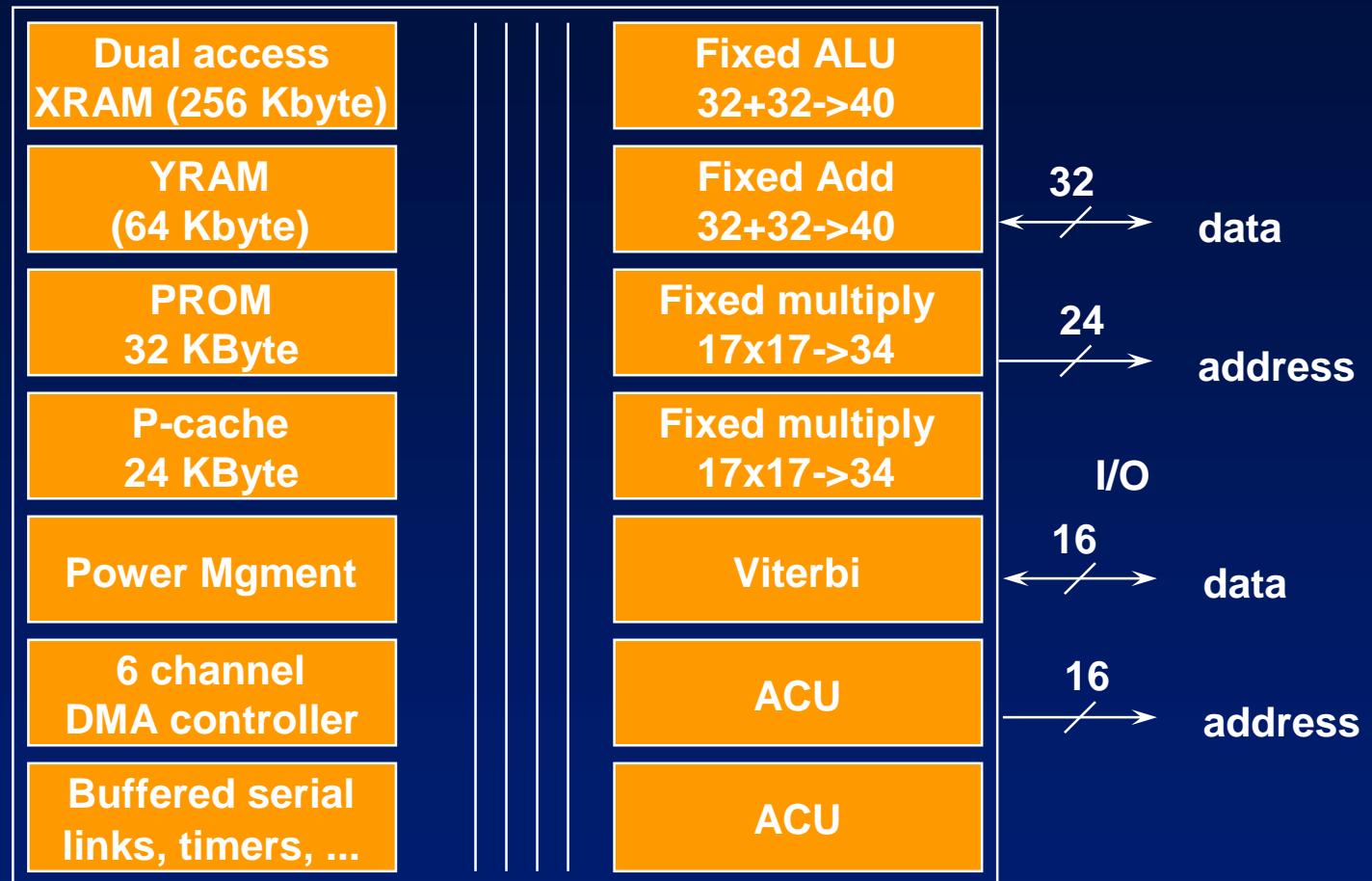
## High end consumer Fixed Point



- Series continued; typical app.: GSM, set-top box, audio
- 1.8-5V, max. 160 MHz, 144 pin, .15 $\mu$ m (1999), 0.32mW/MIPS for the core
- Specialized on-chip unit: will occur more often in future
- e.g. C5420: dual core + 2x100 MW on-chip SRAM
- 83    e.g. C5402: 5\$ for 100 MIPS

# Texas Instruments TMS320C5510

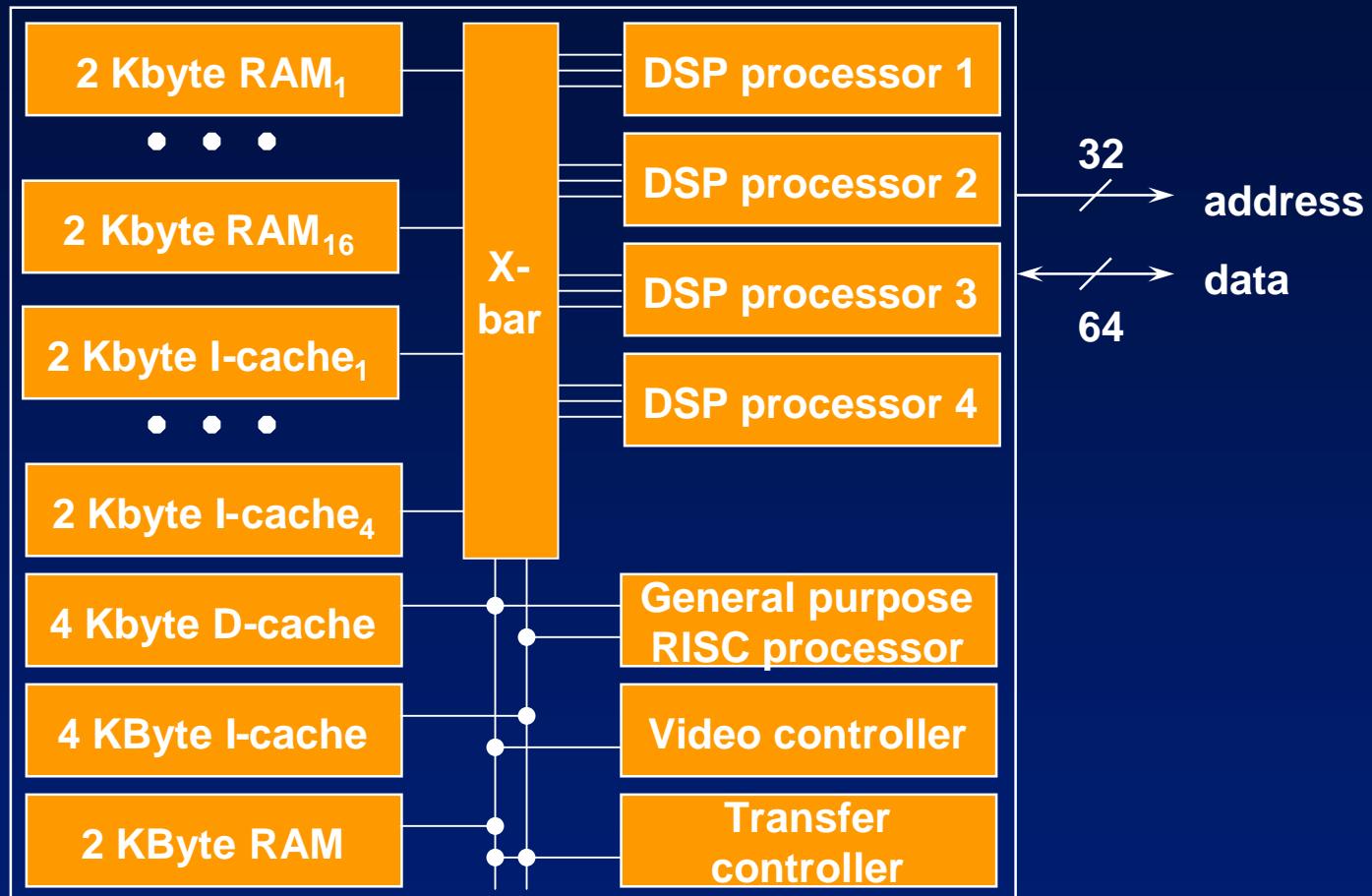
## High end consumer Fixed Point



- Series continued; typical app.: UMTS handheld
- 1.6V, 200 MHz, .15 $\mu$ m (2000), 400 MIPS, 0.05mW/MIPS (core), power management per unit and per cycle
- 84     Specialized on-chip unit: will occur more often in future

# Texas Instruments TMS320C8x

## Fixed Point Video



- **Series discontinued; typical app.: video phone, video conferencing, multimedia workstations**
- **Introduced: 1995, 50 MHz, 305 pin**
- **Multiprocessor-on-a-chip; sub-word SIMD for each DSP**

# Texas Instruments TMS320C6201

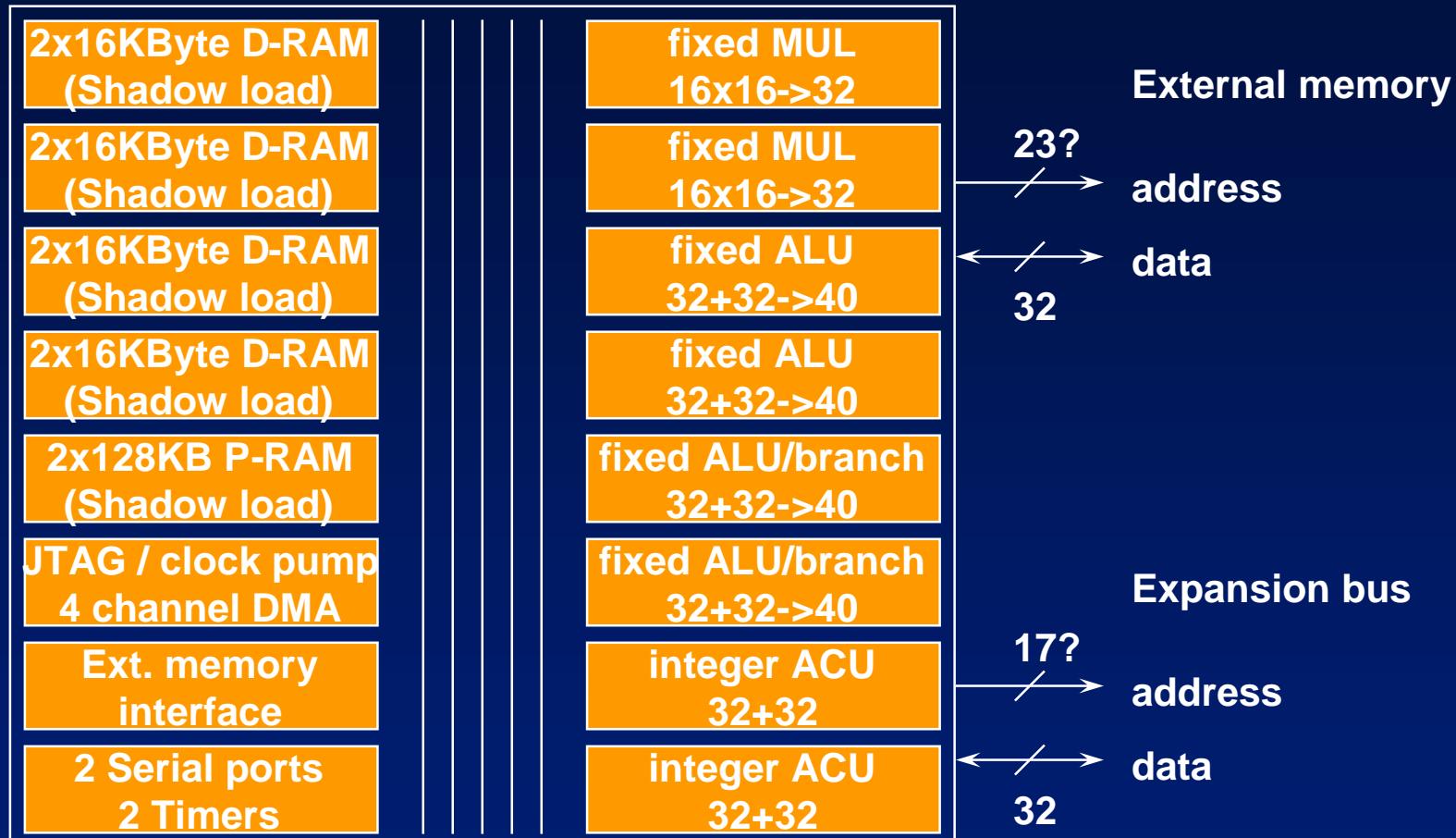
## High end Fixed Point



- Series continued; typical app.: modems, multimedia
- 1997, 0.25  $\mu$ m, 5ML, 352 pin, 200 MHz, 2.5V, 1.9W, \$85
- Super scalar (8 Instr./cycle), 1600 MIPS
- VLIW: 256 bit instruction word

# Texas Instruments TMS320C6202

## High end Fixed Point



- Series continued; typical app.: modems, multimedia
- 1999, 0.18  $\mu$ m, 5ML, 352 pin, 250 MHz, 1.8V, 1.9W, \$130
- Super scalar (8 Instr./cycle), 2000 MIPS, scales well till 700 MHz (6000 MIPS)
- Optimum choice when all data fits in on-chip memory

# Texas Instruments TMS320C6203

## High end Fixed Point



- Series continued; typical app.: base stations
- 2000, 0.15  $\mu$ m, 5ML, 18 mm<sup>2</sup> package size, 300 MHz, 1.5V, 1.5W
- Super scalar (8 Instr./cycle), 2400 MIPS
- Optimum choice when all data fits in on-chip memory

# Texas Instruments TMS320C6211

## High end Fixed Point



- Series continued; typical app.: modems, multimedia
- 1999, 0.18  $\mu$ m, 5ML, 256 pin, 150 MHz, 1.8V, 1.5W, \$25
- VLIW, 1.2 GIPS; cheap (25\$ in '99, 5\$ in '01)
- Optimum for random access to large memory space
- 89 80% of performance of C6x with infinite on-chip memory

# Texas Instruments TMS320C6416

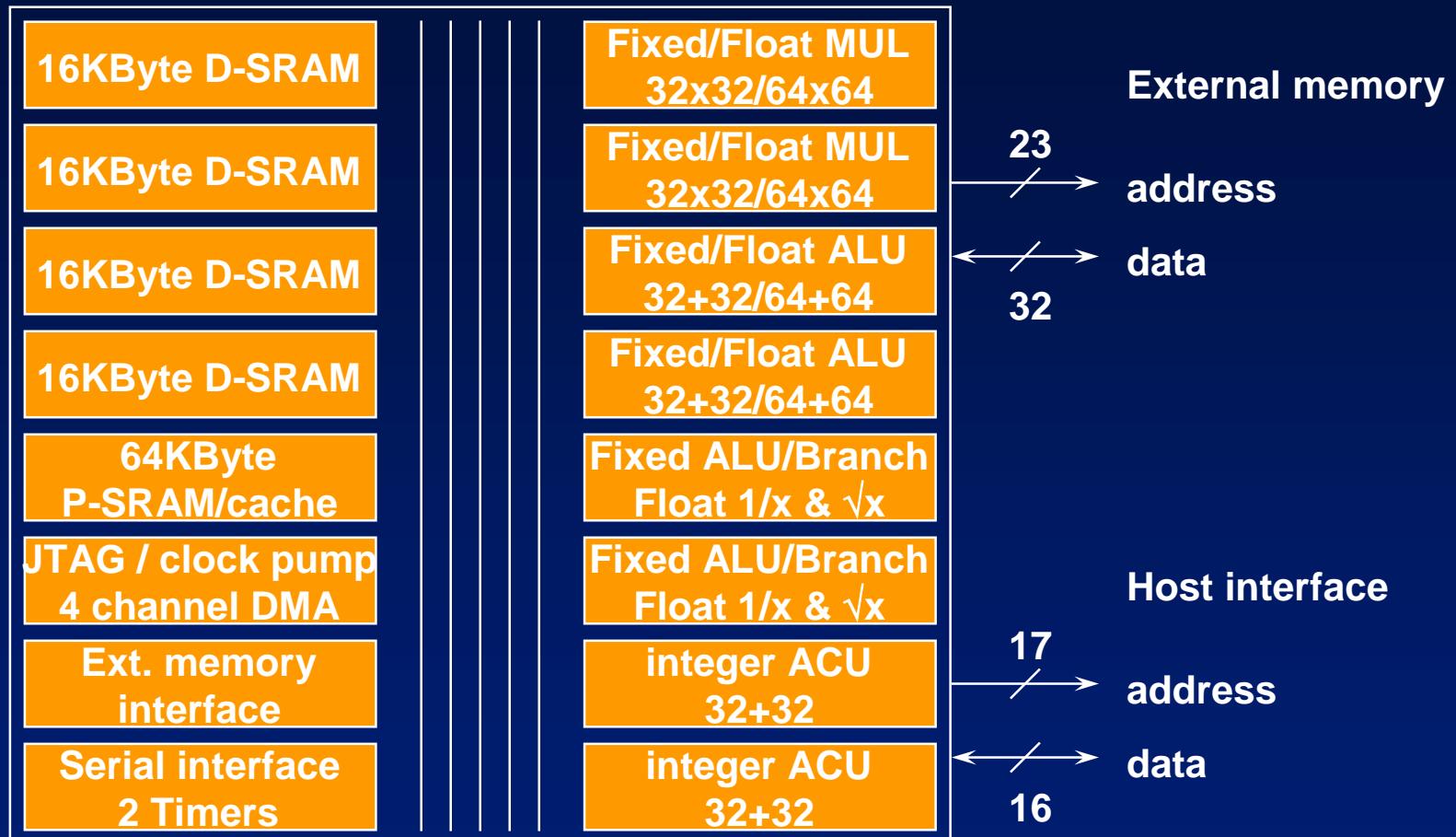
## High end Fixed Point



- Samples June 2001, 0.12  $\mu\text{m}$ , 6 LM, 532 pin, 400 MHz-600 MHz, 1.2V, starts at 95\$ in volume
- Super scalar (8 Instr./cycle), 3200-4800 MIPS
- Sub-word (8bit or 16bit) parallelism
- Specialized instr.: Galois Field Mult, bit manipulation

# Texas Instruments TMS320C6701

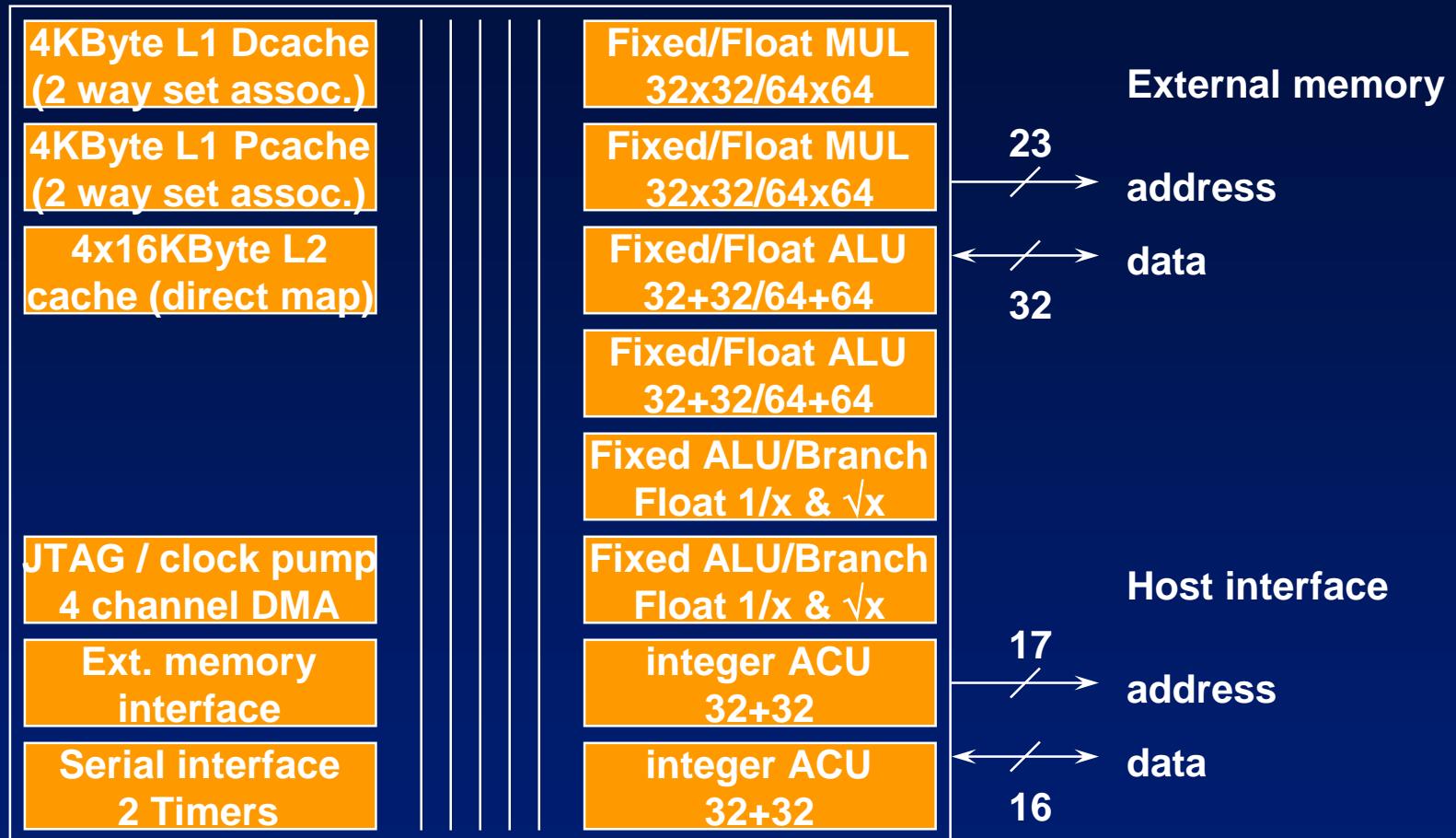
## High end Floating Point



- Series continued; typical app.: **video compression**
- Introduced: 1998, 0.18  $\mu\text{m}$ , 5ML, 352 pin, 167 MHz, 1.8V
- Super scalar (8 Instr./cycle); VLIW; 1 GFLOP
- Foreseen for '00: 50\$ (cf. C6211) & 3 GFLOP (cf. C6202)

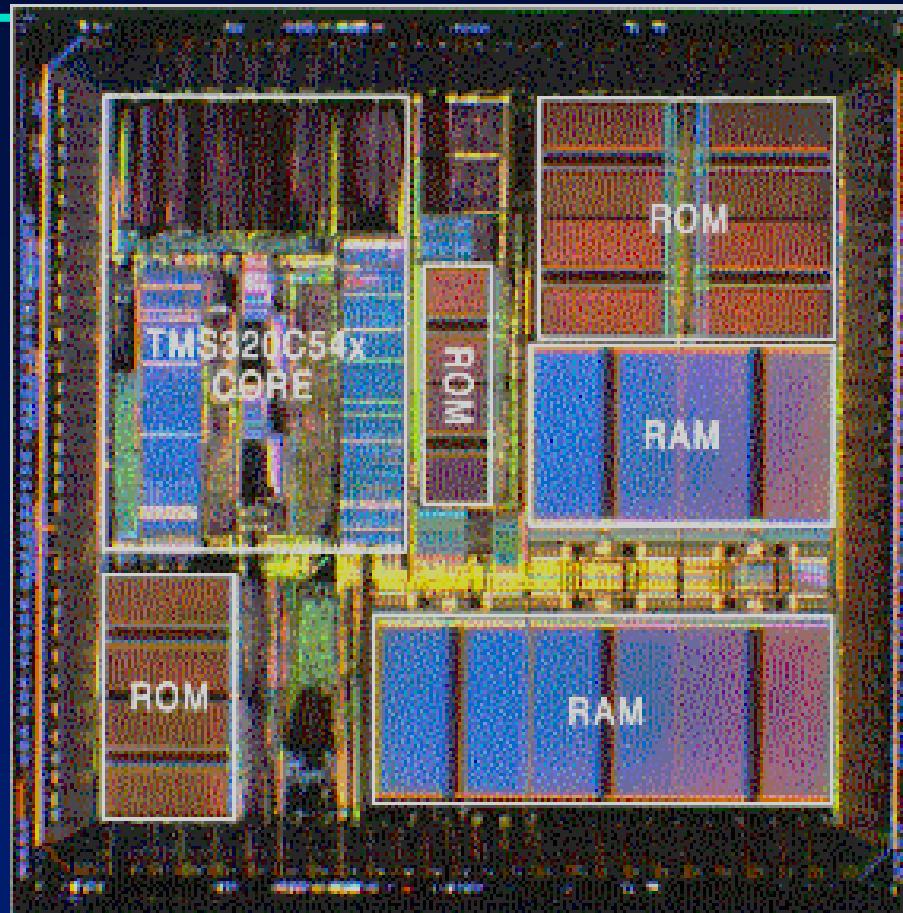
# Texas Instruments TMS320C6711

## High end Floating Point



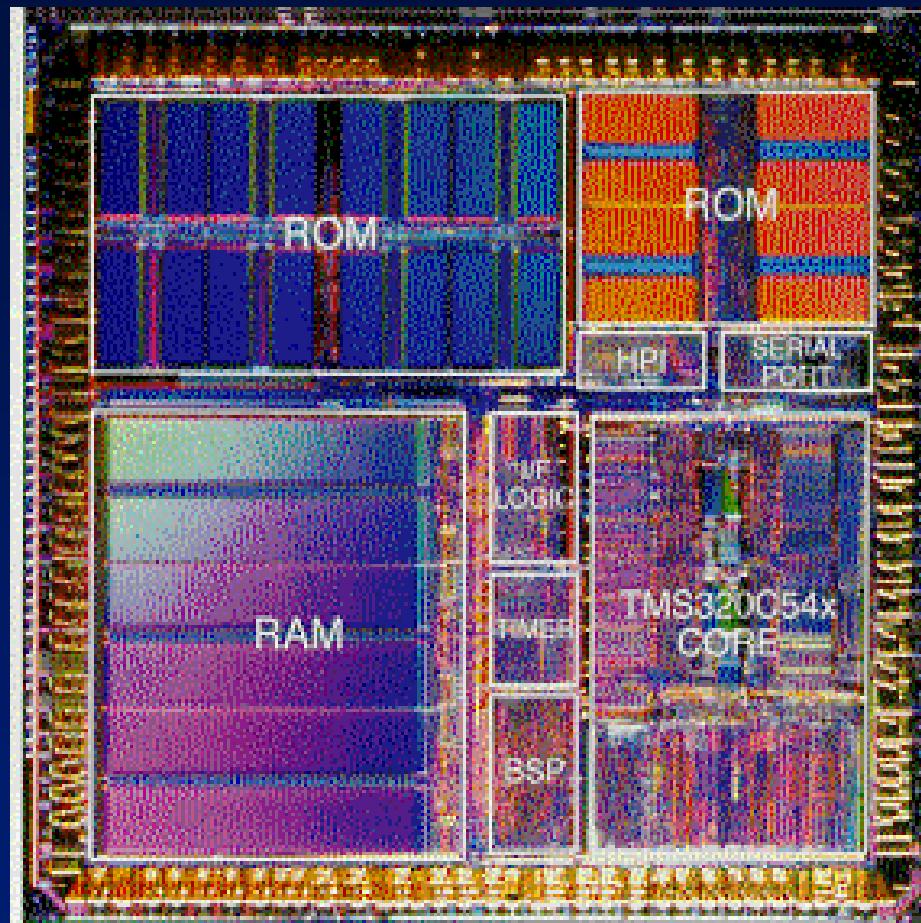
- Series continued; typical app.: video compression
- 2000, 0.18  $\mu$ m, 5ML, 256 pin, 100 MHz, 1.8V, 2W, \$20
- VLIW, 600 MFlops
- Optimum for random access to large memory space
- 80% of performance of C6x with infinite on-chip memory

# Texas Instruments



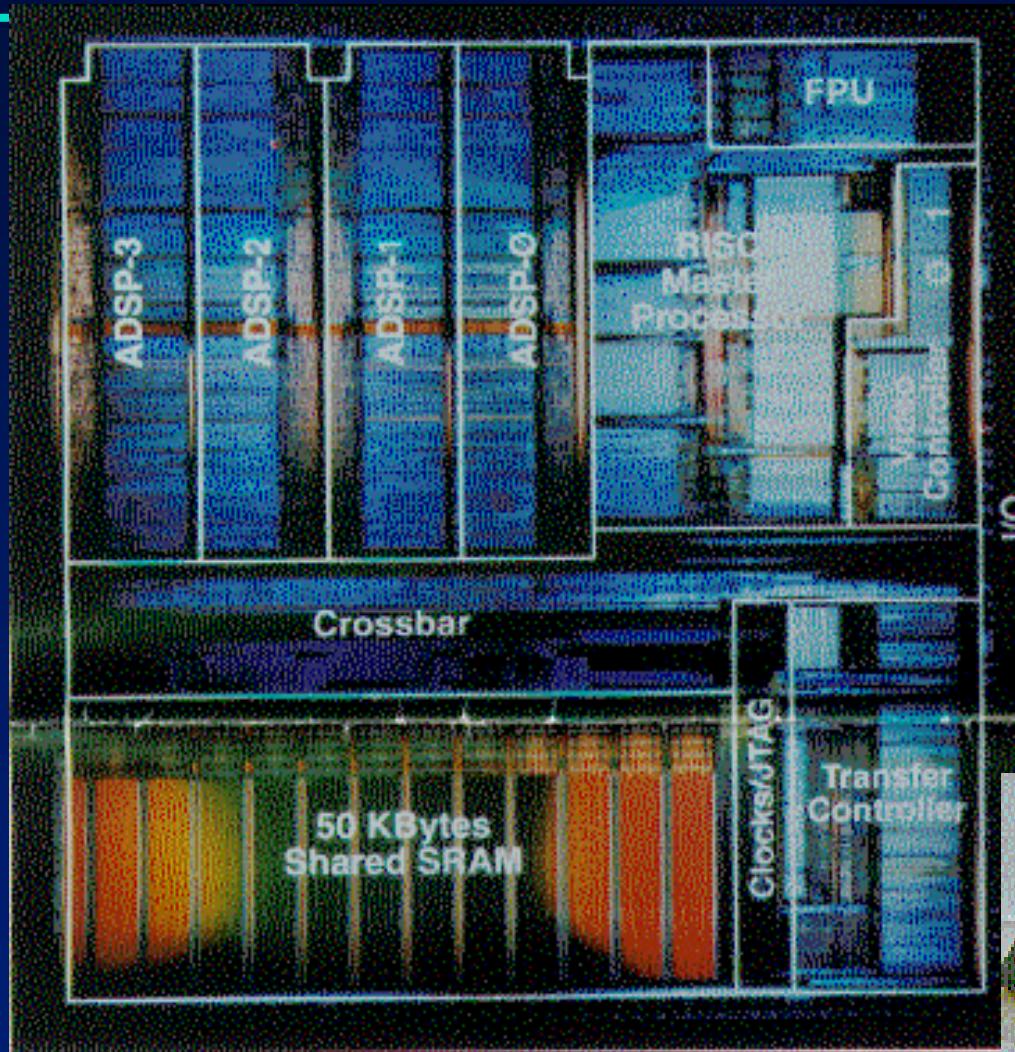
TMS320C541 (1995)

# Texas Instruments



TMS320C545 (1995)

# Texas Instruments



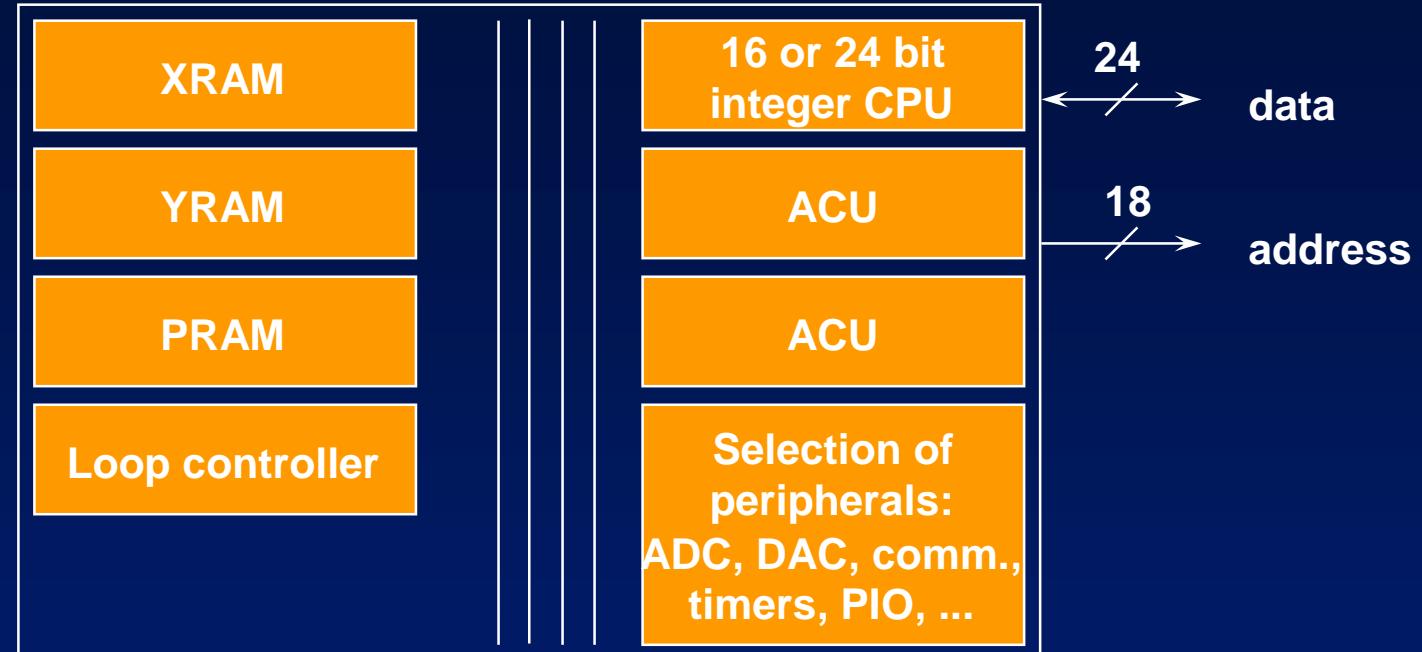
TMS320C80 (1994)



# Chương 7: Các bộ vi xử lý trên thực tế

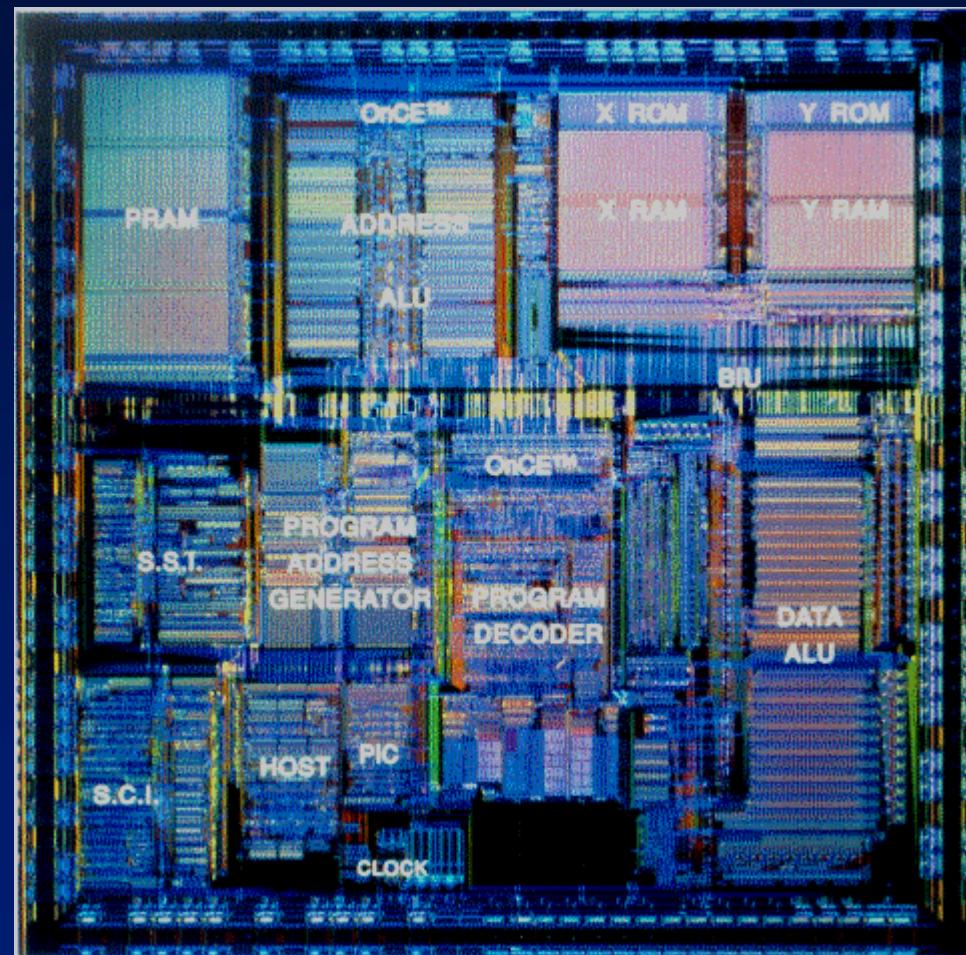
- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Họ vi điều khiển 8051
  - Họ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# Motorola MC56xxx Audio Fixed Point

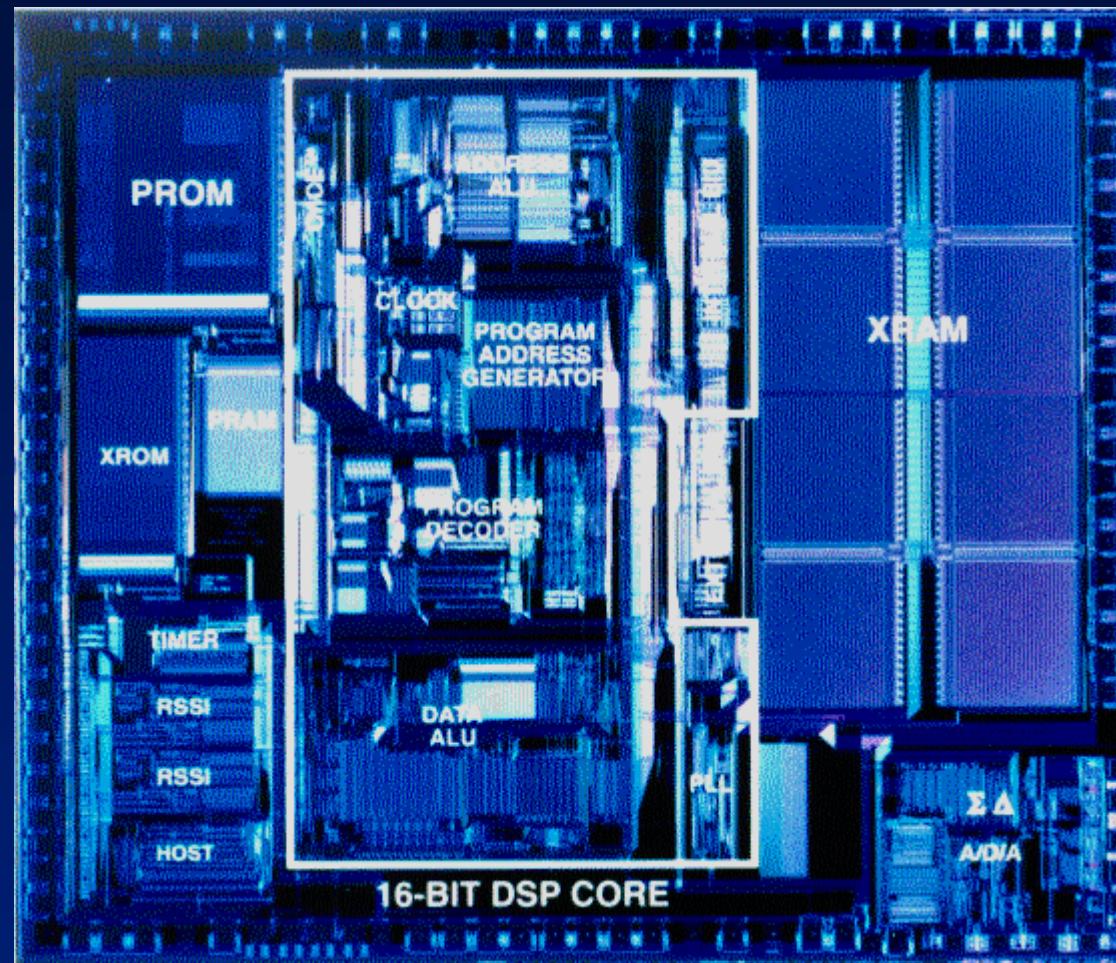


- **24 bit for audio: 16 bit data + overflow**

# Motorola MC56002



# Motorola MC56166

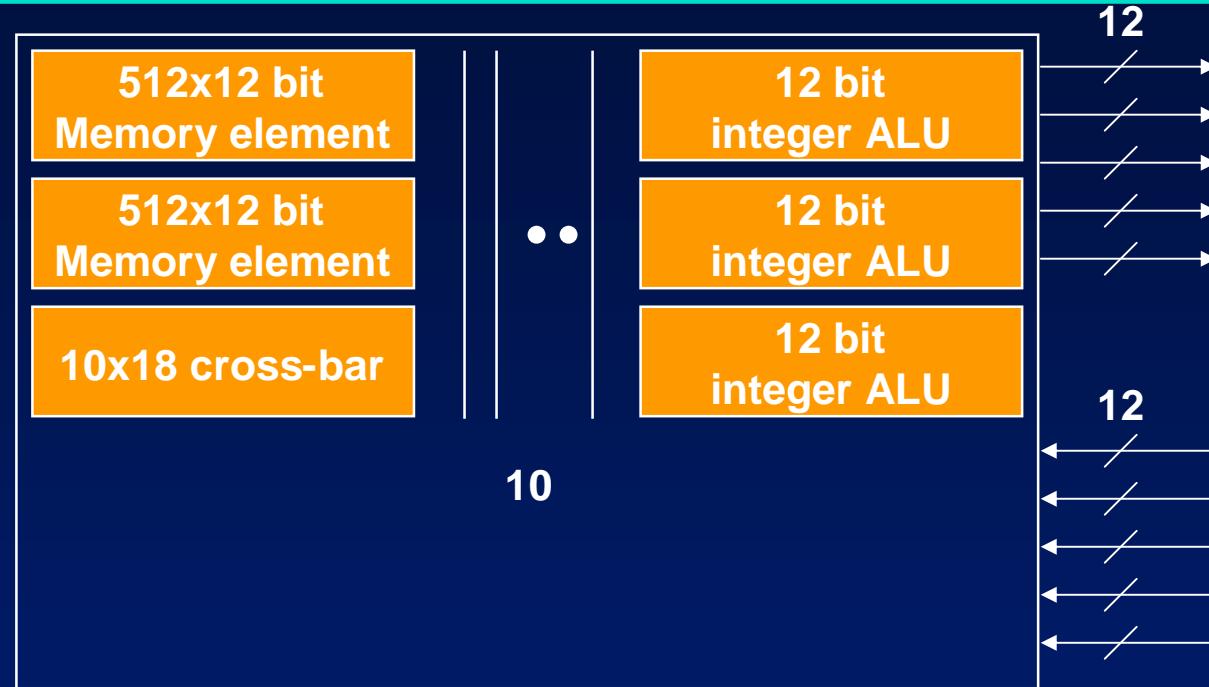


# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Họ vi điều khiển 8051
  - Họ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

# Philips VSP-1

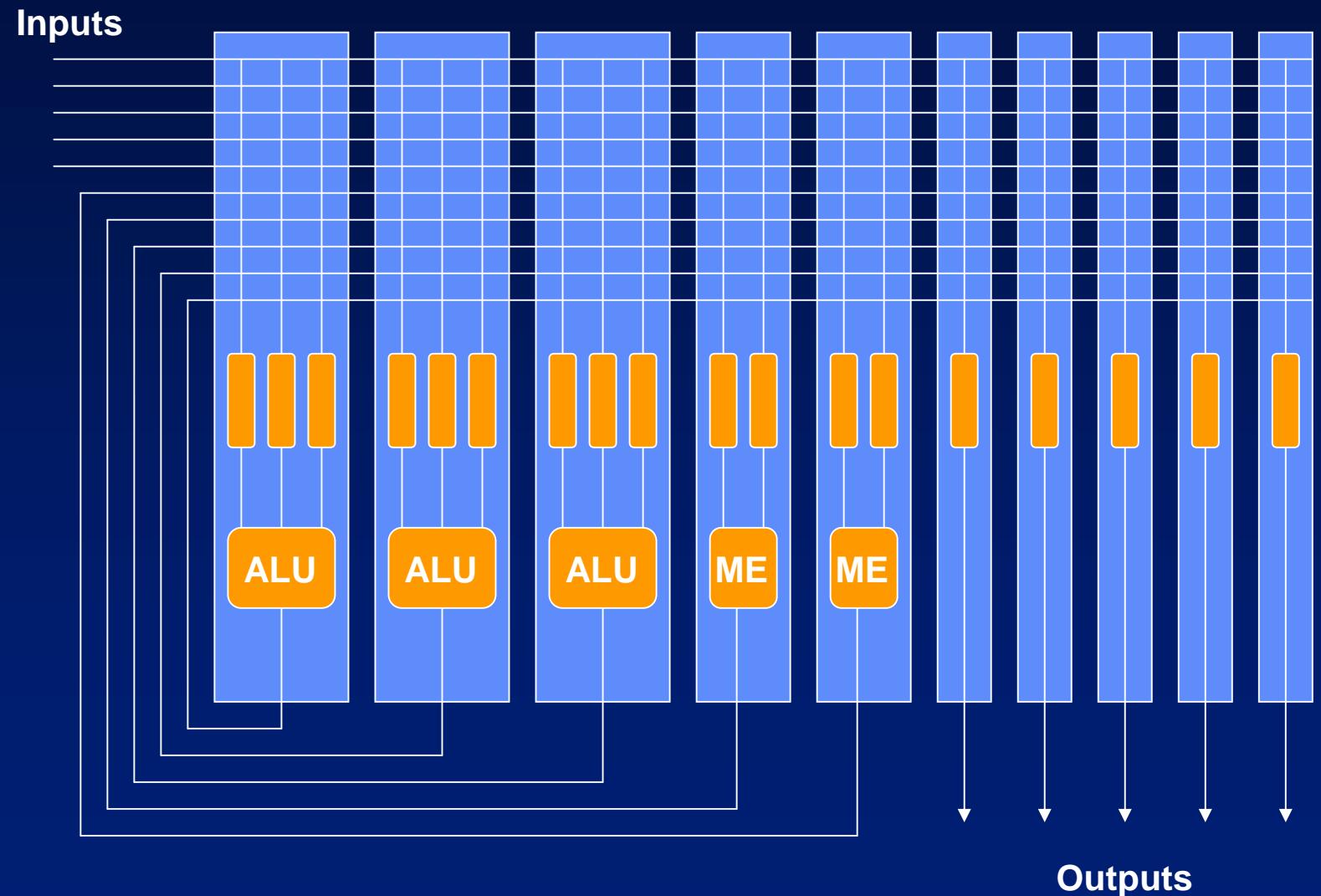
## Fixed Point Video



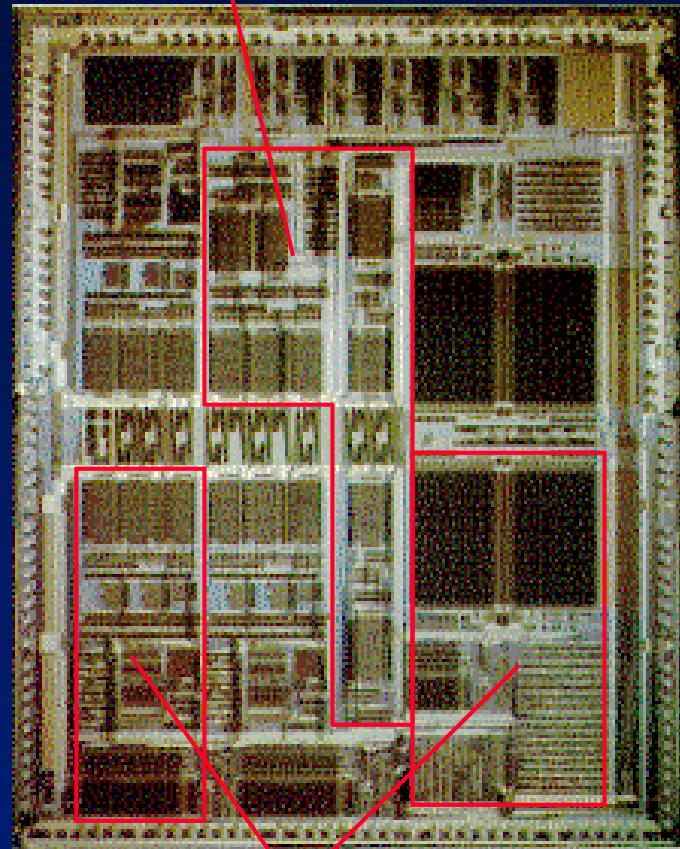
- **12 bit for video: 8 bit data + overflow**
- **Clock Frequency: 27 MHz**
- **1 instruction per sample period for HDTV,  
2 instructions per sample period for TV**

# Philips VSP-1

## Fixed Point Video



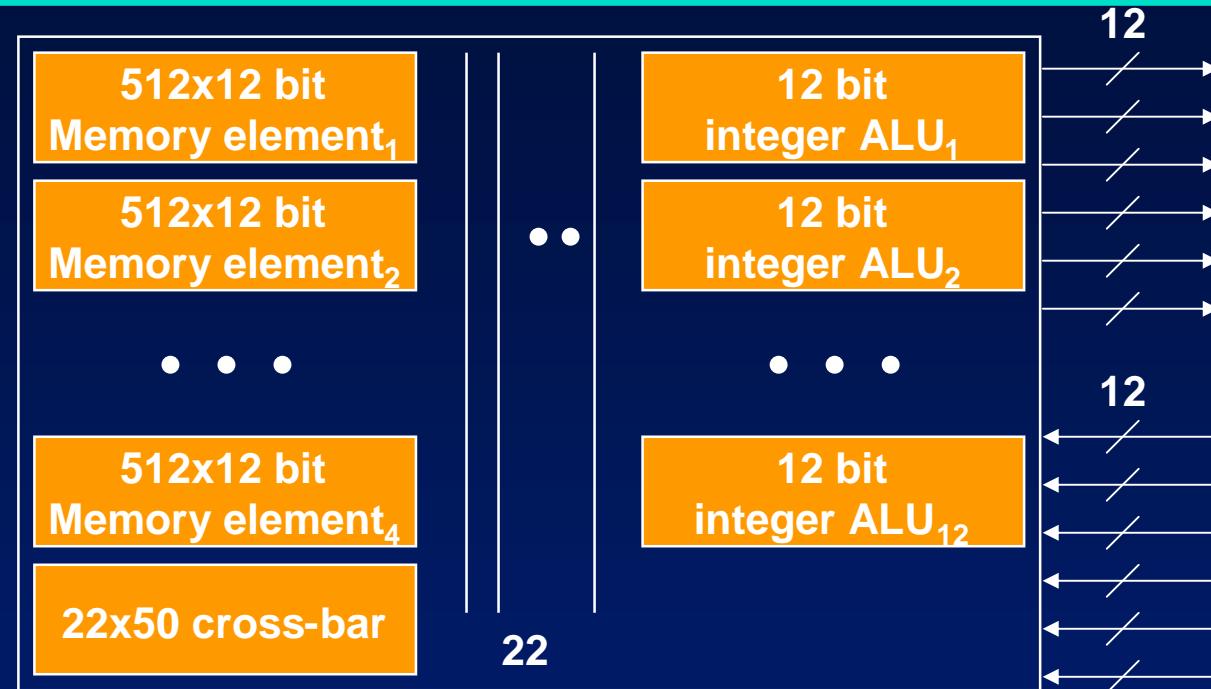
# Philips VSP-1 Fixed Point Video



- 206K Transistors
- 1.1W dissipation
- 27 MHz clock
- 176 pin
- Introduced in 1991

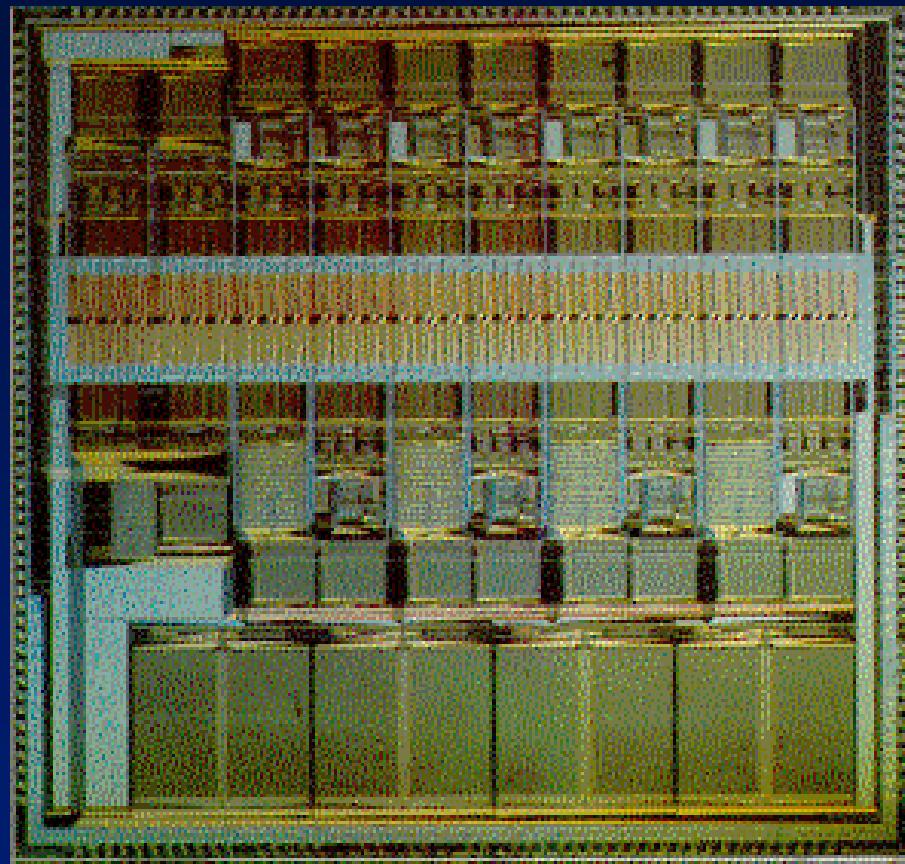
# Philips VSP-2

## Fixed Point Video



- **12 bit for video: 8 bit data + overflow**
- **Clock Frequency: 54 MHz**
- **2 instructions per sample period for HDTV,  
4 instructions per sample period for TV**

# Philips VSP-2 Fixed Point Video



- **1.15 M Transistors**
- **5W dissipation**
- **54 MHz clock frequency**
- **208 pin**
- **Introduced in 1994**

# Sony Graphics Engine

- Playstation 3
  - Status: prototype in 2001
  - 287.5 MTOR
  - 256 Mbit on-chip embedded DRAM
  - 2000-bit wide internal bus
  - 462 mm<sup>2</sup>
  - 180 nm CMOS

# Chương 7: Các bộ vi xử lý trên thực tế

- General purpose microprocessors
  - Intel 80x86
  - Xu hướng phát triển
- Microcontrollers
  - Vi điều khiển của Motorola
  - Họ vi điều khiển 8051
  - Họ vi điều khiển AVR
  - PSOC
  - Xu hướng phát triển
- Digital signal processors
  - Texas Instruments
  - Motorola
  - Philips
  - Xu hướng phát triển

## Trends for DSP processors

- No new generations that replace old generations, but multiple co-existing architecture lines
- Word length application dependent
  - Automotive: 16-bit fixed point (e.g. C2x)
  - Speech: 32-bit floating point (e.g. C30)
  - Audio: 24-bit fixed point (e.g. MC56K)
  - Telecommunications: 16-32 bit fixed point (e.g. C5x, C6x)
  - Video: 12-32 bit fixed point (e.g. C8x)
- Single architecture line is whole family
  - different memory & on-chip peripherals
  - for embedded applications (cf. microcontrollers)

## Trends for DSP processors

- **Deterministic behavior**
  - no caches, no virtual memory, but on-chip RAM banks
  - no out-of-order execution
  - delayed branch prediction
- **Increasing address space: 12 -> 32**
- **Multiple functions on single chip: CPU, FPU, multiple RAM banks, ACUs, loop controller, ADC, DAC, PWM, serial interfaces, ...**
- **Often provisions for parallel processing**