ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
The Univesity of Da Nang - University of Technology and Education

BÀI GIẢNG KỸ THUẬT VI XỬ LÝ
MICROPROCESSORS

Số TC : 2TC

GVC. TS. TRẦN HOÀNG VŨ
Khoa Điện – Điện tử
Email: thvu@ute.udn.vn

Tài liệu tham khảo

- 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

Đánh giá môn học

1. Chuyên cần, phát biểu trong lớp: 20%
2. KT Giữa kỳ : 30%
3. Thi kết thúc môn học: 50%

Chương1
Các hệ thống đếm, Mã hoá và Các linh kiện số cơ bản

Nhắc lại các nội dung sau :

- Biểu diễn dữ liệu
- Các loại mã
- Bộ đệm 3 trạng thái
- Bộ giải mã 3-8 (74LS138)

CÁC HỆ ĐẾM DÙNG TRONG MÁY TÍNH

TỔNG QUÁT

- **r = cơ số : số ký hiệu dùng trong hệ đếm**
- **d=digit : giá trị của ký hiệu**
- **m = số chữ số trước dấu phẩy**
- **n = số chữ số sau dấu phẩy**

$$d_m d_{m-1} \ldots . d_1 d_0 d_{-1} d_{-2} \ldots d_{-n}$$

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

CÁC HỆ ĐẾM DÙNG TRONG MÁY TÍNH

HỆ THẬP PHÂN (10):
r=10
{0,1,2,3,4,5,6,7,8,9}

$$D = \sum_{i=-n}^{m-1} d_i \bullet 10^i$$

HỆ NHỊ PHÂN (2):
r=2
{0,1}

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

HỆ BÁT PHÂN (8):
r=8
{0,1,2,3,4,5,6,7}

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

HỆ THẬP LỤC PHÂN (16):
r=16
{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

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

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 chứ số của hệ nhị phân cần tìm

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

Chyển từ hệ nhị phân sang hệ 16 và ngược lại

- 1011 0111B = B7H

CỘNG NHỊ PHÂN

Cộng thập phân

Nhớ	**0 1 0**
x	**8 2 7 3**
y	**5 6 2**
Tổng	**8 8 3 5**

Cộng nhị phân

Nhớ	**0 0 1 1 1 1 1**
x	**1 0 0 1 1 0 1 1**
y	**1 0 1 0 1 1 1**
Tổng	**1 1 1 1 0 0 1 0**

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 PHẬN

Nguyên tắc: cộng và dịch

```
      1 1 1 0
      1 1 0 1
    ---------
      1 1 1 0
    0 0 0 0
  1 1 1 0
1 1 1 0
-----------
1 0 1 1 0 1 1 0
```

CHIA NHỊ PHÂN

Nguyên tắc: trừ và dịch

```
1 0 1 1 1 0 1 0 | 1 1 1 0
  1 1 1 0       |
  ---------     | 1 1 0 1
  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 |
```

SỐ CÓ DẤU

Một số có dấu bao gồm 2 phần: dấu và độ lớn

Ví dụ hệ 10: $+123_{10}$ (thông thường '123') và -123_{10}

Hệ nhị phân: bít dấu là bít MSB; '0' = dương, '1' = âm

Ví dụ: $01100_2 = +12_{10}$ và $11100_2 = -12_{10}$

Các số có dấu 8 bít sẽ có giá trị từ -127 đến +127 với 2 số 0: 1000 0000 (-0) và 0000 0000 (+0)

SỐ BÙ 2

Số bù 1 (bù lô gic): đảo bit

- **1001 => 0110**
- **0100 => 1011**

Số bù 2 (bù số học): số bù 1 +1

Ví dụ: Tìm số bù 2 của 13

13 = 0000 1101

Số bù 1 của 13 =1111 0010

Cộng thêm 1: 1

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

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 bít sẽ có giá trị từ -128 đến 127

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)**

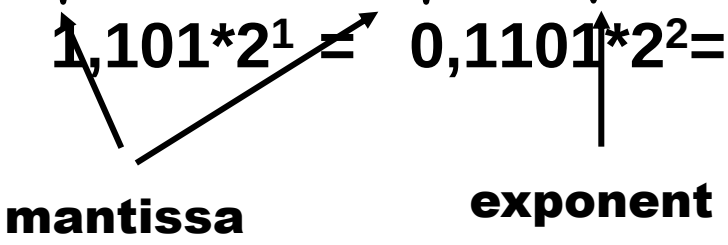
32 bit

- **unsigned: 0 đến 2^{32}-1**
- **signed : -2^{31} đến 2^{31}-1**

Số thực
(real number, floating point number)

Ví dụ: $1,234 = 1,234*10^0 = 0,1234*10^1 = ...$

$11,01_B = 1,101*2^1 = 0,1101*2^2 = ...$

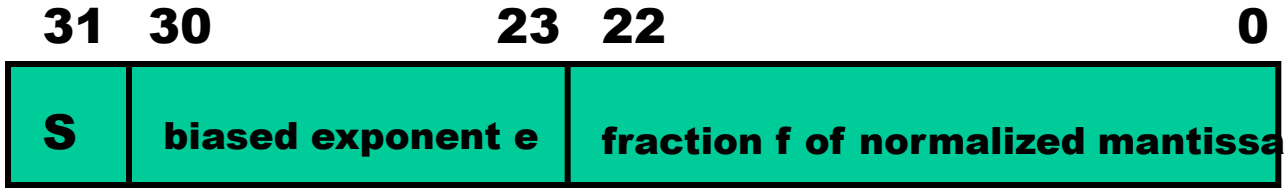


Real number: (m, e) , e.g. (0.1101, 2)

- **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 hoá = 1 bit ẩn + 23 bit fraction

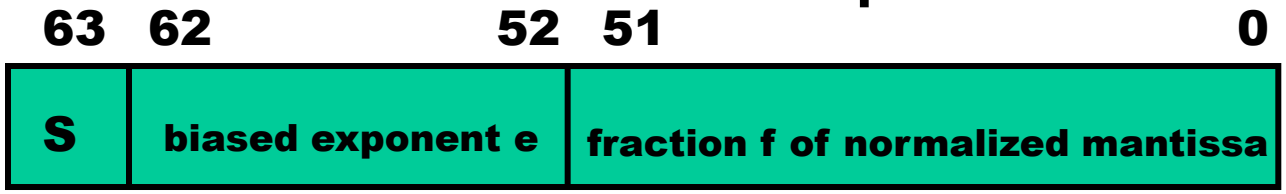
Mantissa chuẩn hoá: có giá trị giữa 1 và 2 : 1.f

Ví dụ: biểu diễn 0.1011 dưới dạng IEEE-754

Sign bit s=0
chuẩn hoá mantissa: $0.1011=1.011*2^{-1}$
Biased exponent: -1 + 127=126=01111110
IEEE format: 0 01111110 01100000000000000000000

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

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

b3b2b1b0	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	V
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

BCD

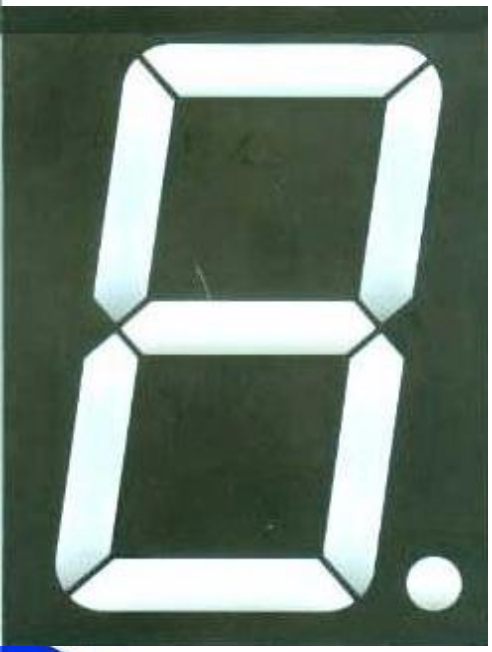
Binary Coded Decimal number

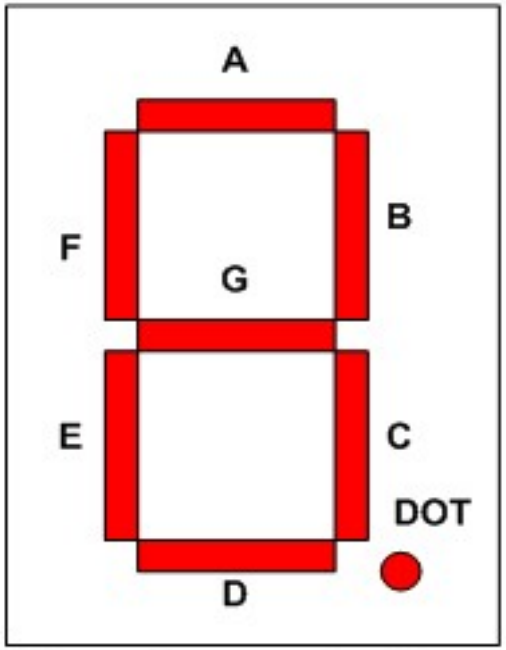
- **BCD chuẩn (BCD gói, packed BCD):**
  - **1 byte biểu diễn 2 số BCD**
  - **Ví dụ: 25: 0010 0101**
- **BCD không gói (unpacked BCD) :**
  - **1 byte biểu diễn 1 số BCD**
  - **ví dụ: 25: 00000010 00000101**
- **BCD 8421**
- **A → 1010**
- **F→ 1111**
- **1011 -> B**
- **1100 _> C**
- **1110 → D**
- **Chuyển đổi (B-D-H)**
- **BCD 7421**

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

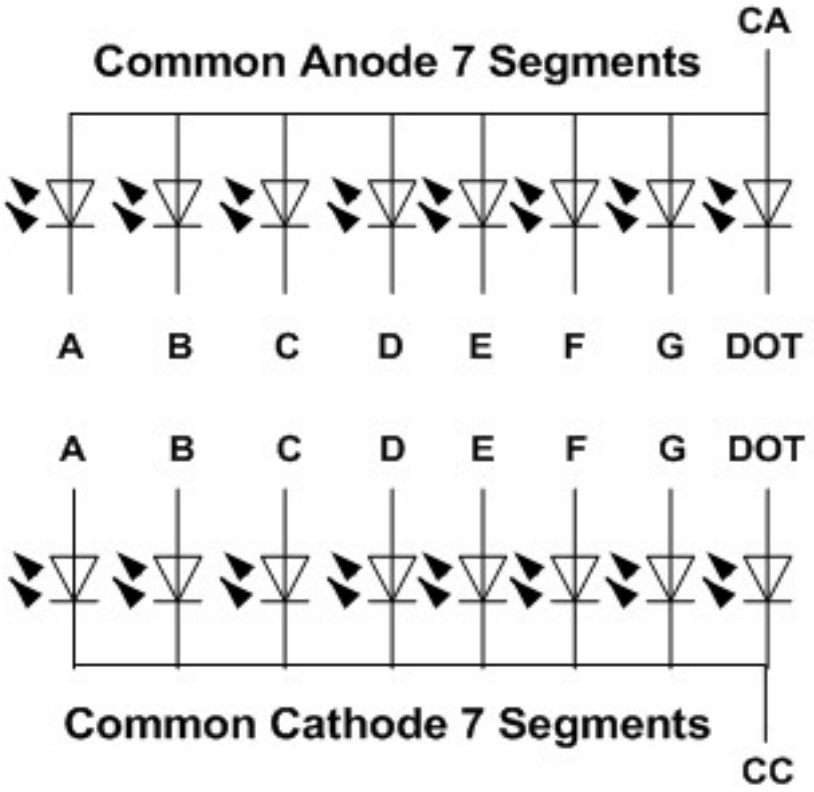
7 Segment Led

- Anod chung
- Cathod chung





The 7 Segment's Name and the DOT



7-Segment code

HIỂN THỊ	ANOD CHUNG	CATHOD CHUNG	HIỂN THỊ	ANOD CHUNG	CATHOD CHUNG
0	C0h	3Fh	9	98h	67h
1	F9h	06h	A	88h	77h
2	A4h	5B	B	C6h	39h
3	B0h	4Fh	C	86h	79h
4	99h	66h	D	E8h	71h
5	92h	6Dh	E	82h	70h
6	82h	7Dh	F	89h	76h
7	F8h	07h	.	7Fh	80h
8	80h	7Fh	trắng	FFh	00h

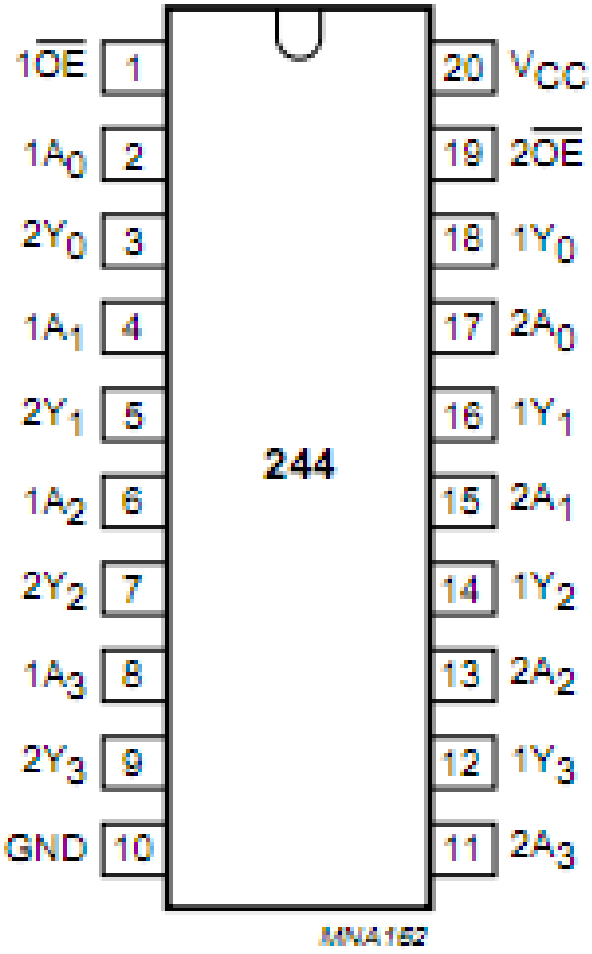
BỘ ĐỆM 3 TRẠNG THÁI

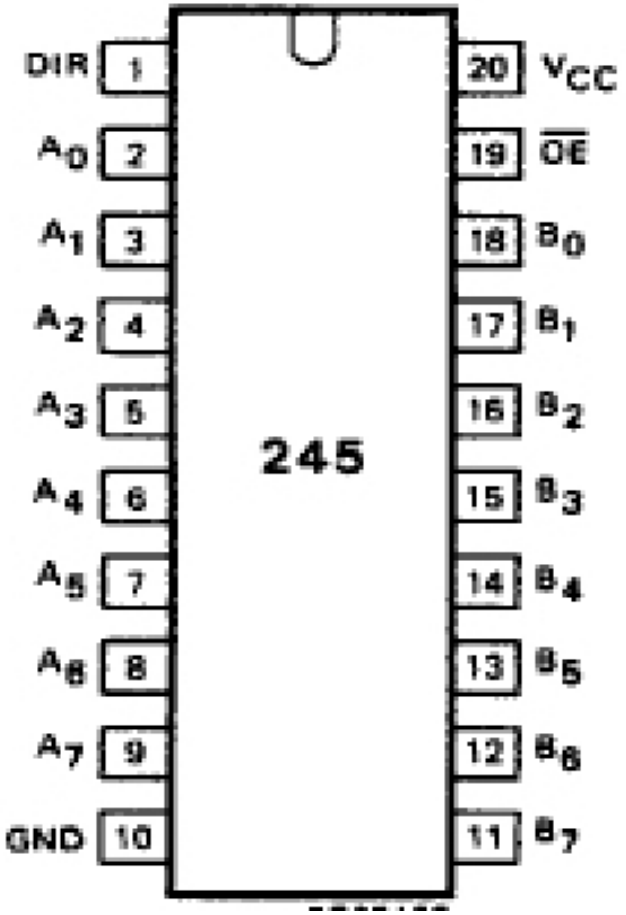
1OE=0

20E=0

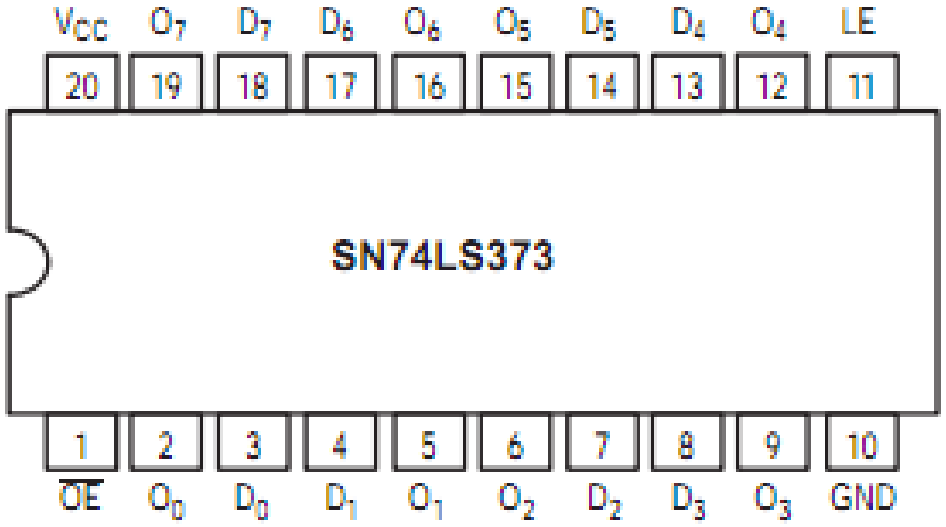
Ai→vào

Yi→ra





BỘ CHỐT 74LS373



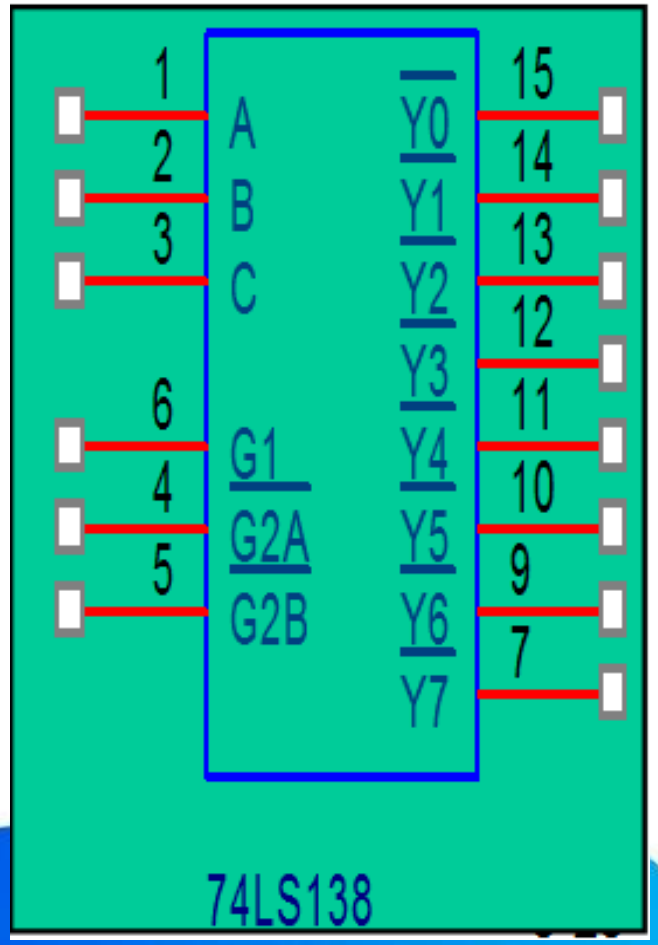
TRUTH TABLE

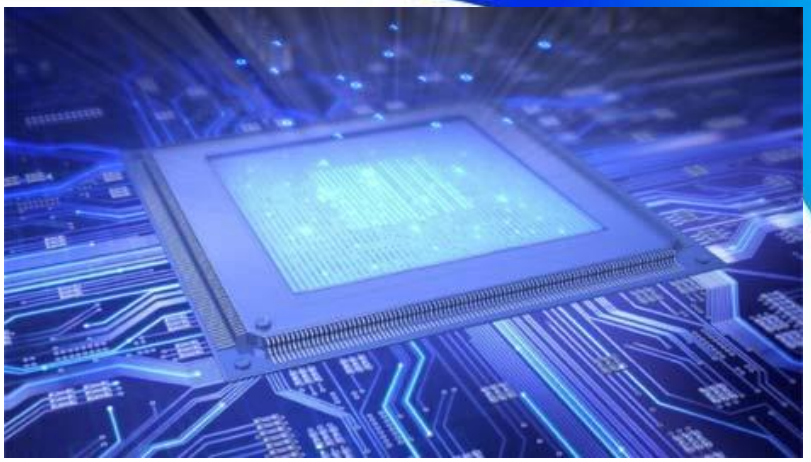
D_n	LE	$\overline{OE}$	O_n
H	H	L	H
L	H	L	L
X	L	L	Q_0
X	X	H	Z*

BỘ GIẢI MÃ 3-8 (IC 74138)

Để IC hoạt động ở chế độ giải mã, thiết lập như sau :
G1 =1; G2A=G2B =0
(Trường hợp khác : Y0-Y7=1)

INPUT			OUTPUT							
C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0





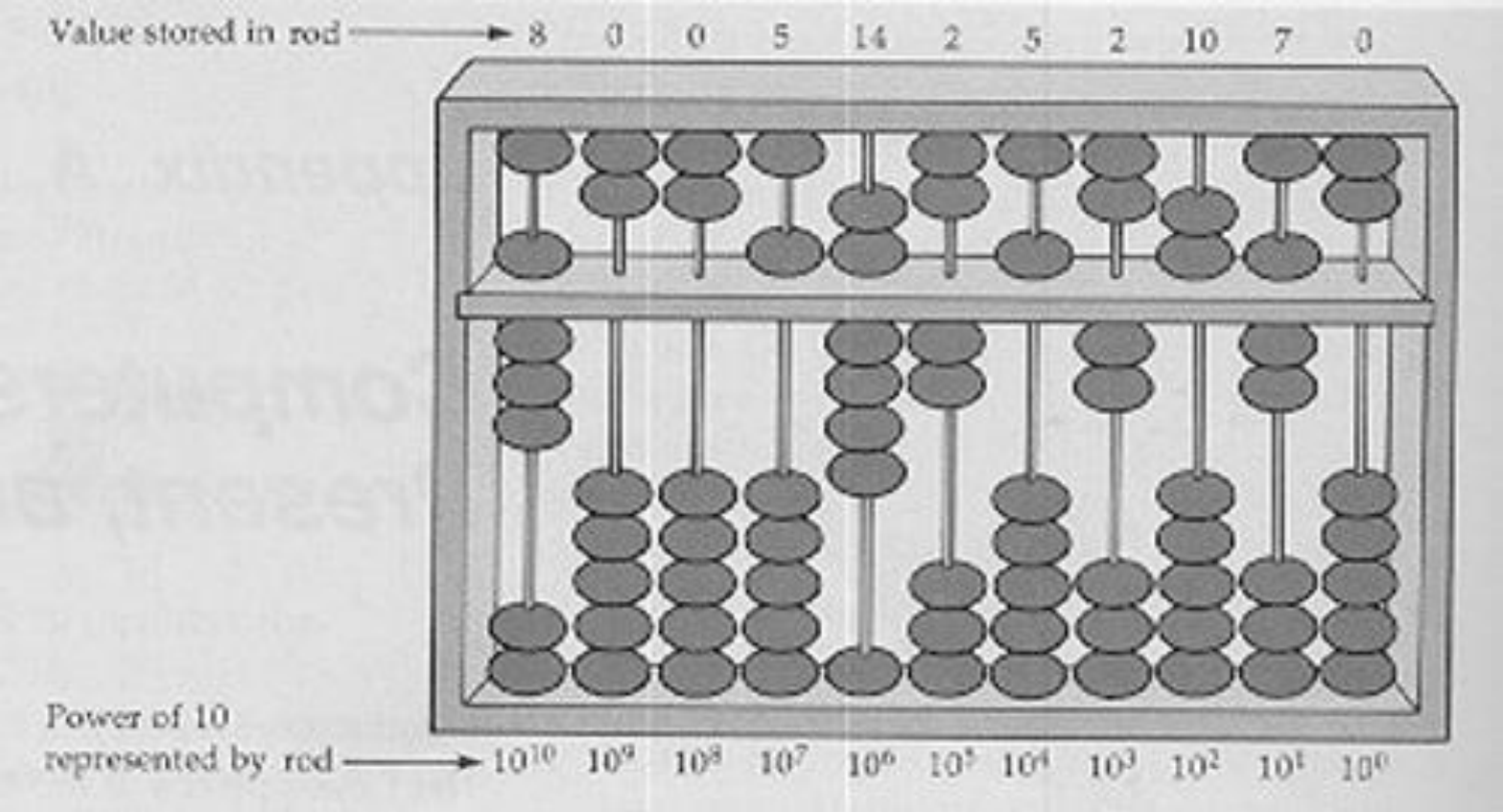
Chương2
Khái niệm về vi xử lý và hệ thống vi xử lý

LỊCH SỬ PHÁT TRIỂN 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-?)
- Thế hệ 5: ?

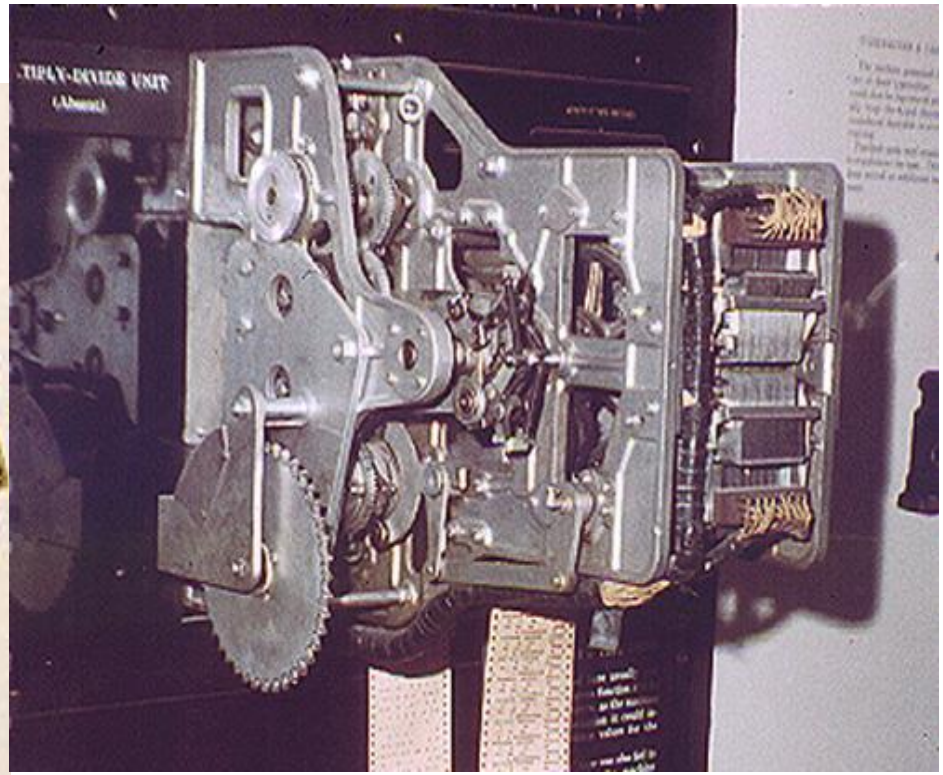
Thế hệ -1: The early days (…-1642)

Bàn tính, *abacus*, đã được sử dụng để tính toán.



Thế hệ 0: Mechanical (1642-1945)

Máy tính Pascal (1642)



Máy Harvard Mark I (IBM Automatic Sequence Control Calculator), phát minh bởi Howard Aiken (1930_

Thế hệ 1: Vacuum tubes (1945-1955)

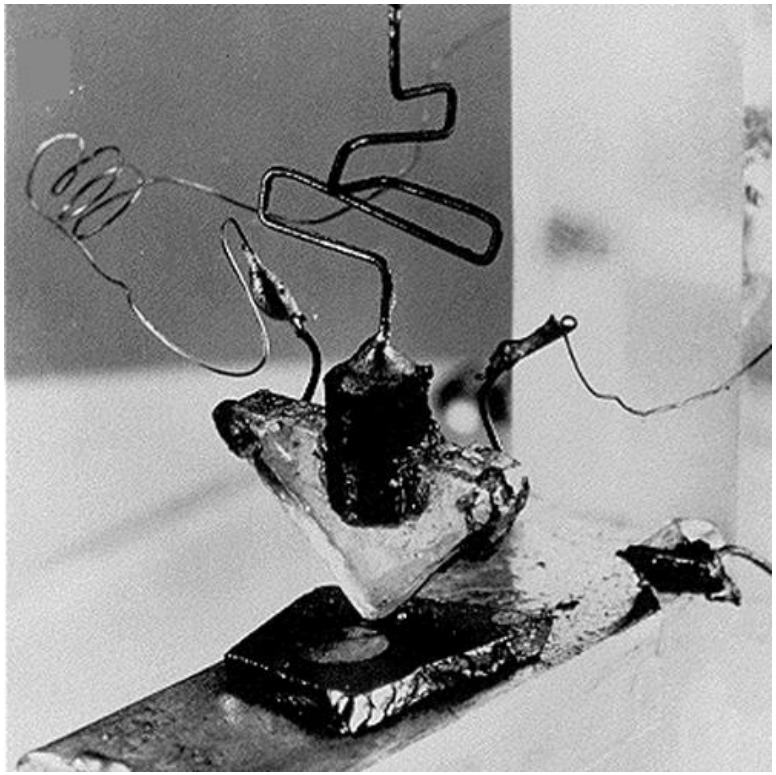
Năm 1946, John von Neumann phát minh ra máy tính có chương trình lưu trong bộ nhớ

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

Thế hệ 2: Discrete transistors (1955-1965)

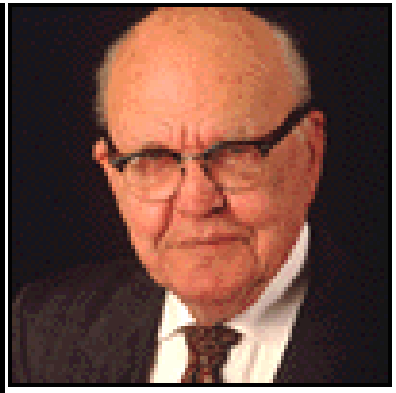
Năm 1947, William Shockley, John Bardeen, and Walter Brattain phát minh ra transistor

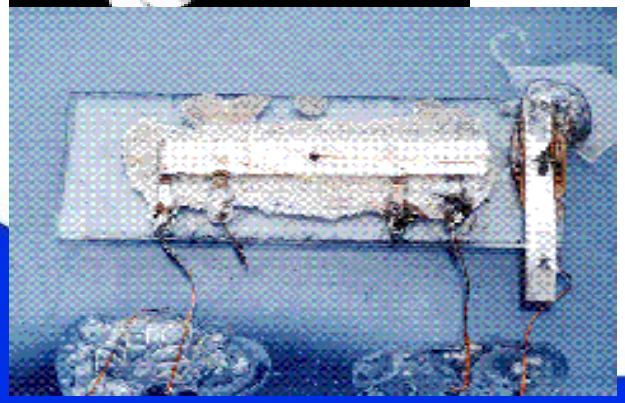


IBM704, máy tính mainframe sử dụng tranzistor(5 kFlops, clock: 300 kHz)

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.



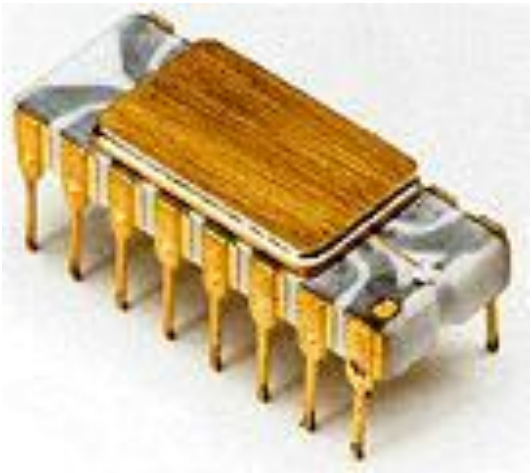


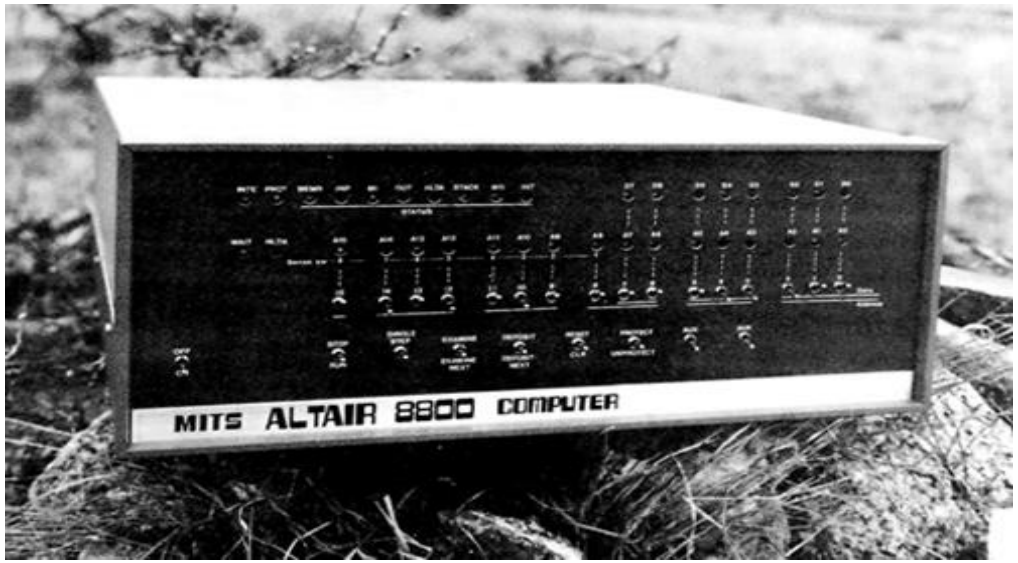
Digital Equipment Corporation, đưa ra chiếc máy tính mini đầu tiên DP-8 (1965)

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 bít dữ liệu, 12 bit địa chỉ



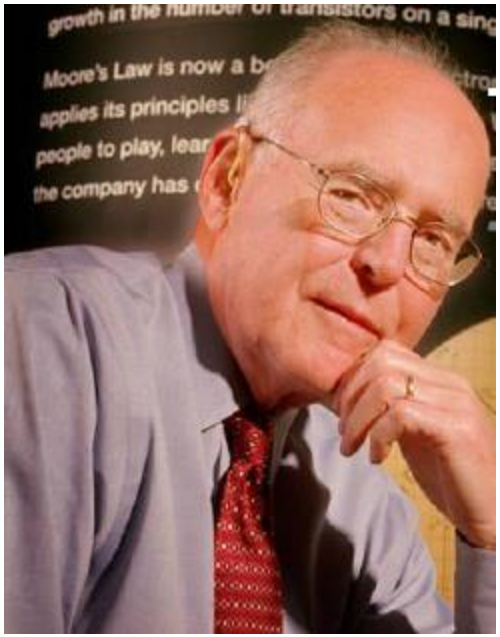


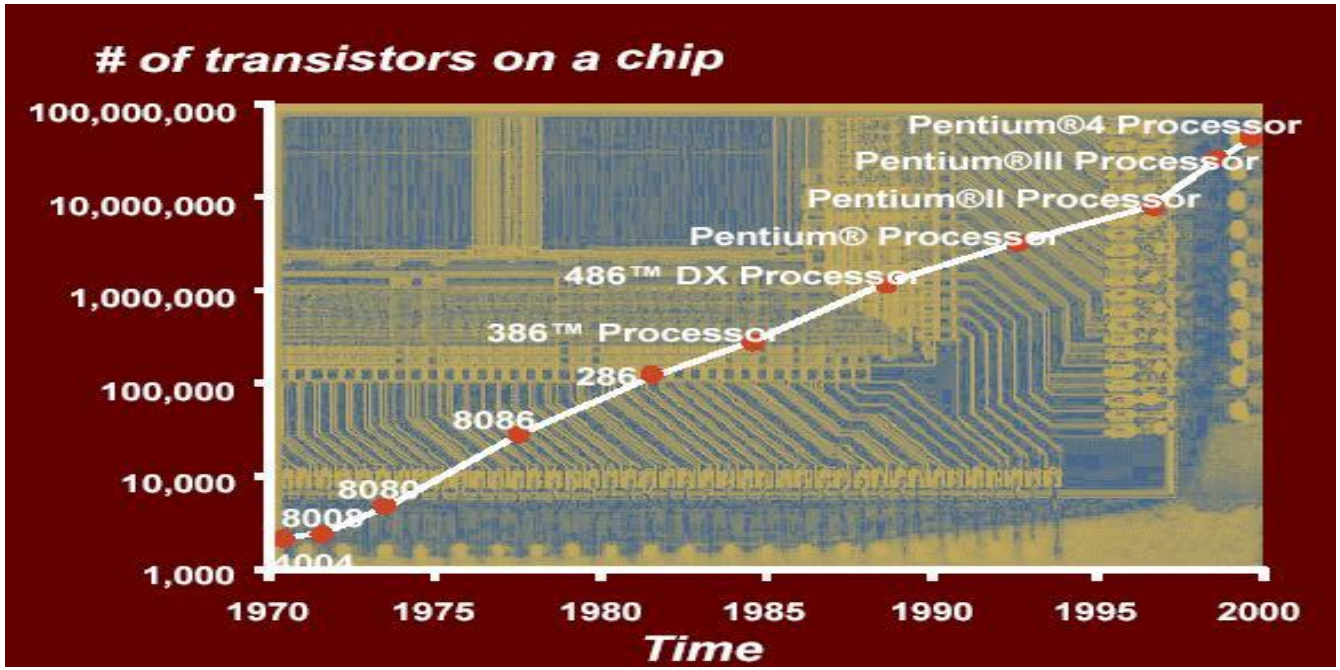
1973-1974, Edward Roberts, William Yates and Jim Bybee chế tạo MITS Altair 8800, máy tính cá nhân đầu tiên

Thế hệ 4: VLSI (1980-?)

IBM-PC sử dụng hệ điều hành DOS (1981)

Thế hệ 4: VLSI (1980-?)





Luật Moore : Cứ sau 18 tháng, số lượng transistors tích hợp trong chip tăng gấp đôi (Gordon Moore, đồng sáng lập công ty Intel)

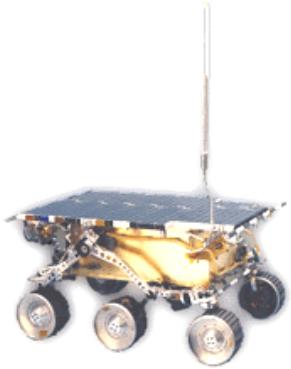
Phân loại vi xử lý





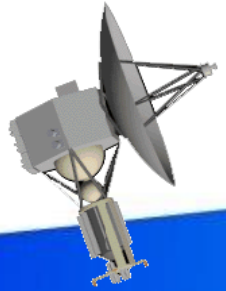
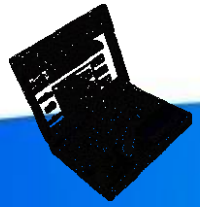




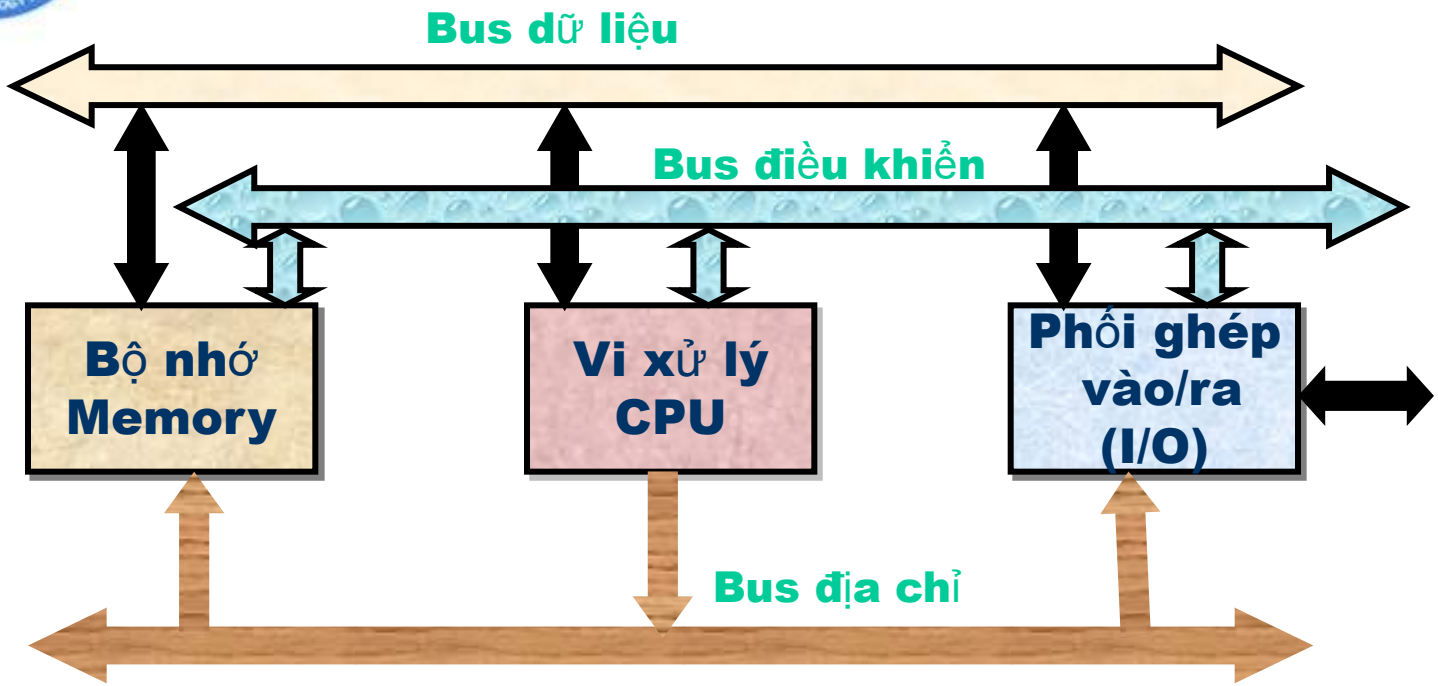








HỆ VI XỬ LÝ



DRAM
SRAM
ROM
EEPROM
Flash

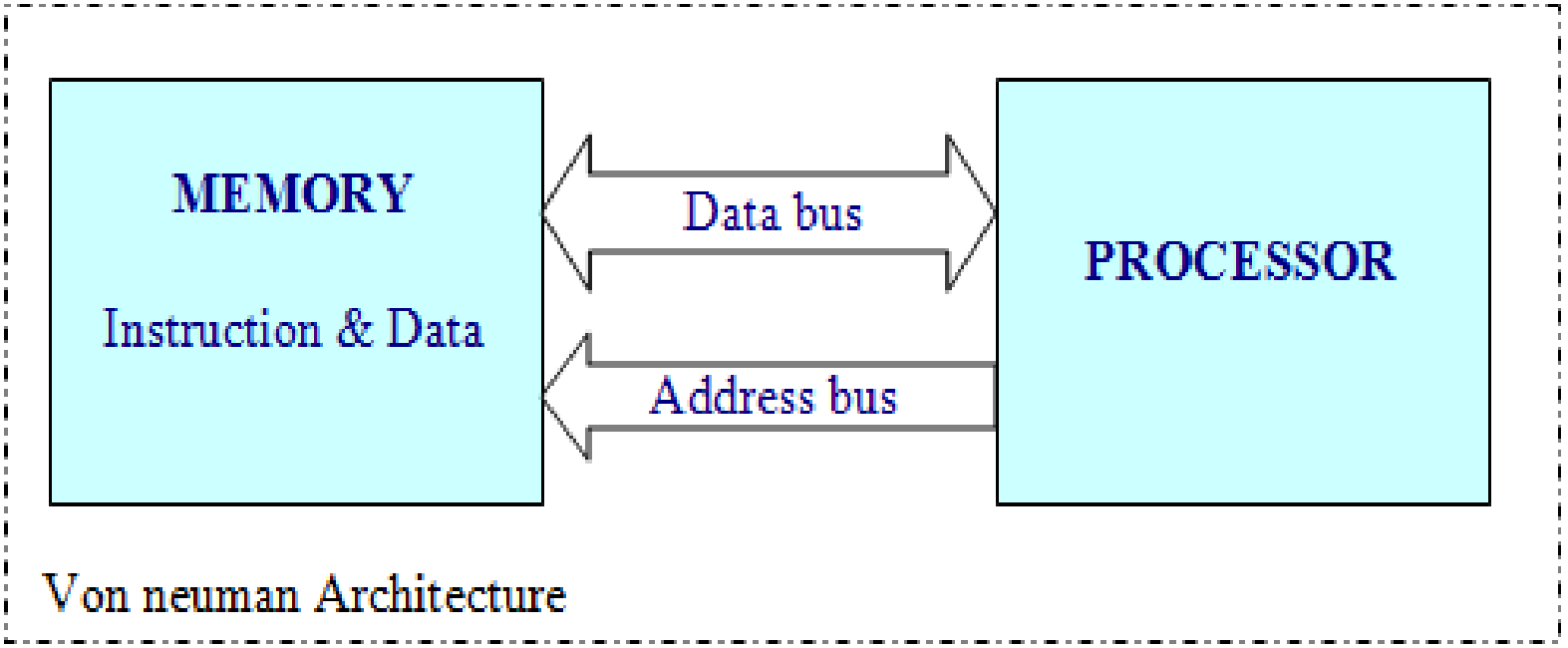
Intel 80X86
ARM
Motorola 680X
PowerPC
...

ISA
EISA
PCI
VESA
SCSI
USB
...

Màn hình
Máy in
Bàn phím
Con chuột
Ổ cứng
Ổ mềm
CD-ROM
DVD
...

Von Neumann Architecture

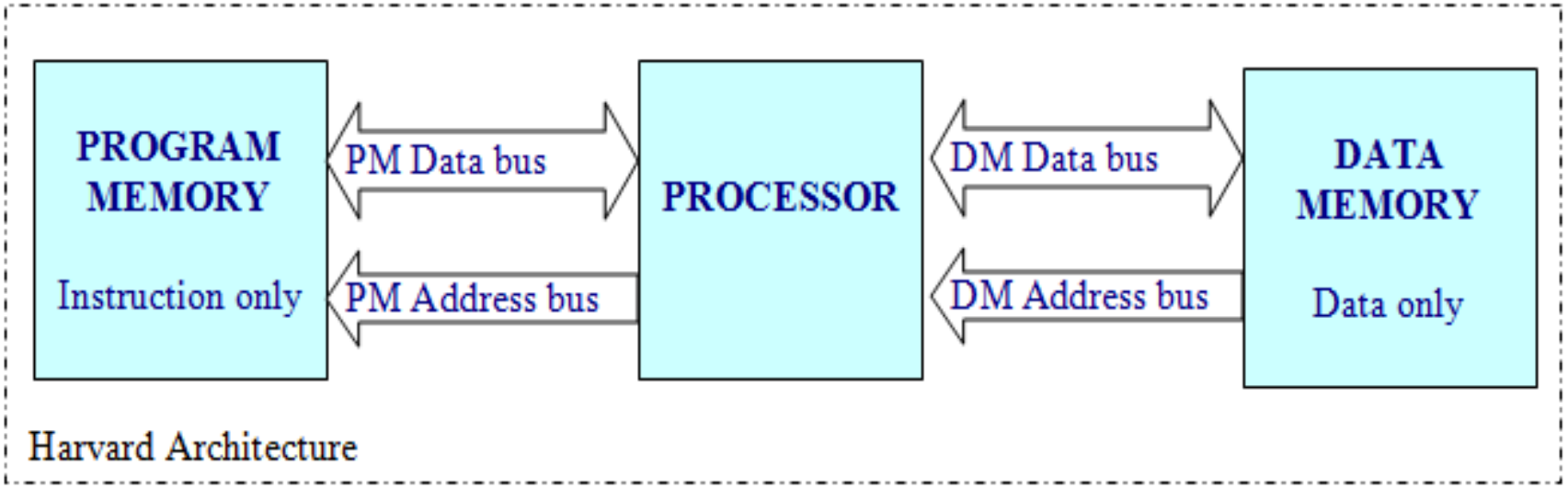
- ✓Do Von Neumann đề xuất
- ✓01 bộ nhớ chung cho chương trình và dữ liệu
- ✓01 hệ thống bus
- ✓Không thể vừa đọc dữ liệu vừa đọc lệnh cùng 1 lúc



Von neuman Architecture

Harvard Architecture

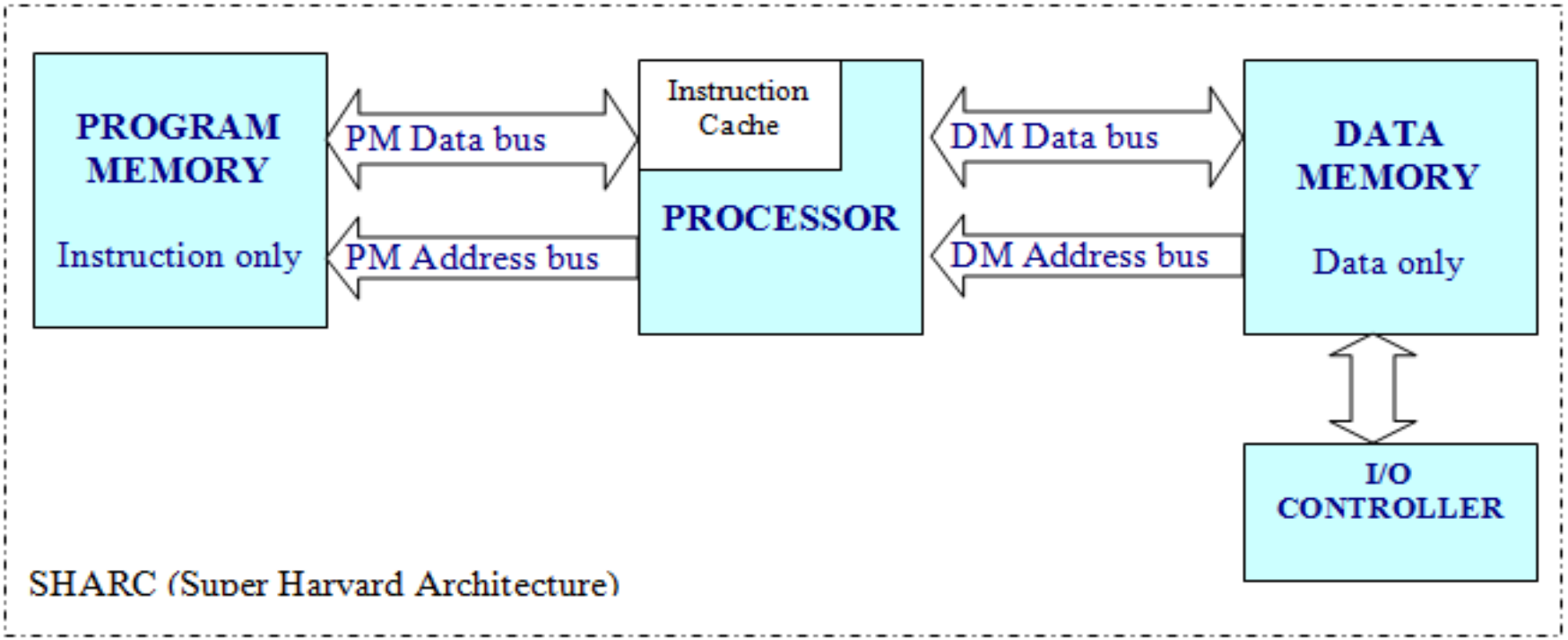
- ✓ Howard Aiken đại học Harvard đề xuất
- ✓ Bộ nhớ chương trình và bộ nhớ dữ liệu riêng
- ✓ 02 hệ thống bus
- ✓ Có thể đồng thời truy xuất dữ liệu và lệnh.
- ✓ Hầu hết các bộ DSP hiện nay sử dụng kiến trúc này



Harvard Architecture

SHARC (Super Harvard Architecture)

- ✓Thêm vào 1 số điểm đặc trưng để cải thiện thông lượng dữ liệu
- ✓Điển hình là dòng ADSP-2106x và ADSP-211x của Analog Device



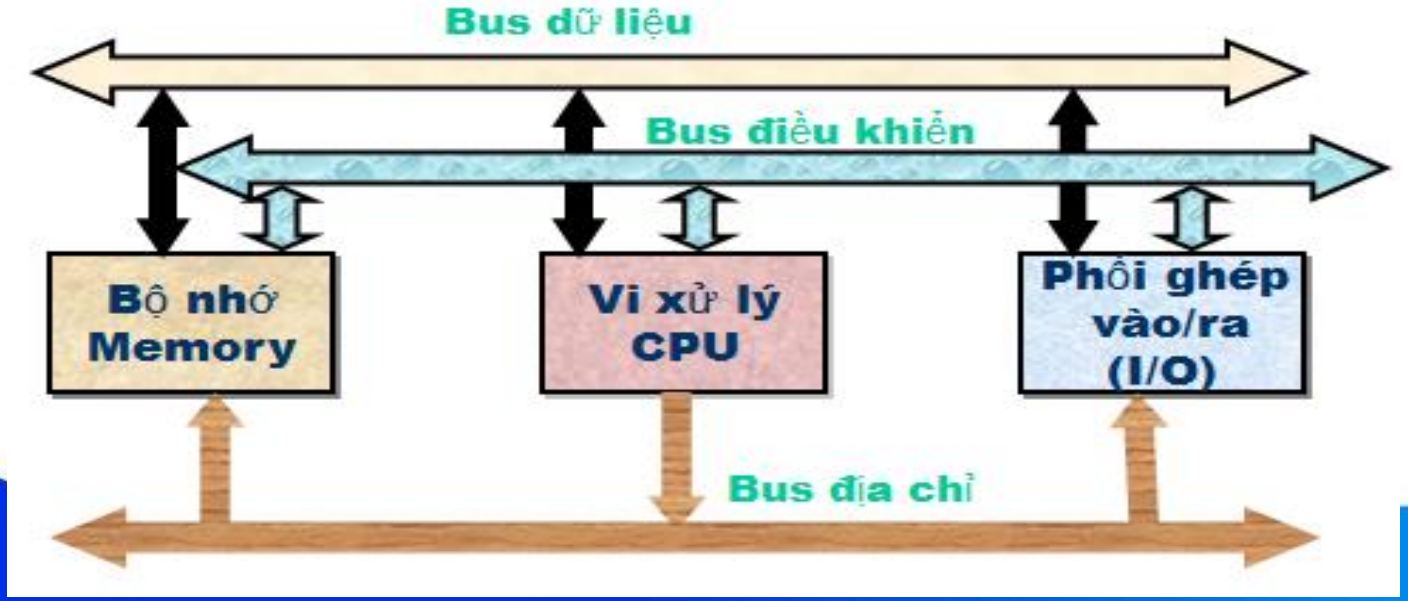
SHARC (Super Harvard Architecture)

BUS

Bus là tập hợp các đường vật lý cho phép liên kết các thành phần của máy tính với nhau

Gồm :

- **Bus dữ liệu (Data Bus)**
- **Bus địa chỉ: (Address Bus)**
- **Bus điều khiển (Control Bus)**



BUS

- **Bus dữ liệu (Data Bus)**
  - **Tập hợp các đường vật lý để chuyển tín hiệu mang dữ liệu (data)**
  - **Độ rộng : 8, 16, 32, 64 bit tùy thuộc vào vi xử lý**
  - **Tốc độ : (Hz)**

BUS

➢Bus địa chỉ: (Address Bus)

- Tập hợp các đường vật lý cho phép chuyển tín hiệu mang thông tin địa chỉ (address)
- Độ rộng : 16, 20, 24, 32, 36 bit
- Quyết định dung lượng bộ nhớ tối đa CPU có thể quản lý
- Abus=N→ số ô nhớ có thể đánh địa chỉ: 2^N
- Ví dụ: 8088/8086 có 20bit địa chỉ => quản lý được 2^{20} bytes=1Mbytes

BUS

- **Bus điều khiển (Control Bus)**
  - **Đường truyền vật lý mang thông tin điều khiển và trạng thái**
  - **Điều khiển các hoạt động đọc/ghi bộ nhớ hoặc I/O**

MEMORY

- ROM Vs. RAM
- SRAM Vs. DRAM
- Bộ nhớ chương trình Vs. Bộ nhớ dữ liệu

MEMORY

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 (Magnetoelectronic 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)

MEMORY

- **Bộ nhớ chương trình :**
  - **chứa chương trình (các mã lệnh) hướng dẫn CPU thực hiện một nhiệm vụ nào đó,.**
  - **Thường là bộ nhớ ROM nhưng đôi khi là bộ nhớ RAM**

- **Bộ nhớ dữ liệu**
  - **Chứa dữ liệu tạm thời (các tham số, các biến) tạo ra trong lúc thực hiện chương trình**
  - **Bộ nhớ dữ liệu thường là RAM (SRAM, DRAM ..)**

KHÔNG GIAN ĐỊA CHỈ

N đường địa chỉ có thể quản lý 2^n ô nhớ

- **vi xử lý có bus địa chỉ là 20 thì có thể quản lý tối đa 2^{20} ô nhớ**

  $2^{20} = 1MB$

- **Một máy tính có 4GB không gian bộ nhớ nghĩa là :**

  Vi xử lý của nó có đủ số đường địa chỉ (bus địa chỉ) để đánh địa chỉ cho 4GB bộ nhớ nghĩa là nó có 32 đường địa chỉ ($2^{32} = 4$ GB)

***Kilo*Byte = 2^{10} or 1024 bytes**

***Mega*Byte = 2^{20} or 1024^2 bytes**

***Giga*Byte = 2^{30} or 1024^3 bytes**

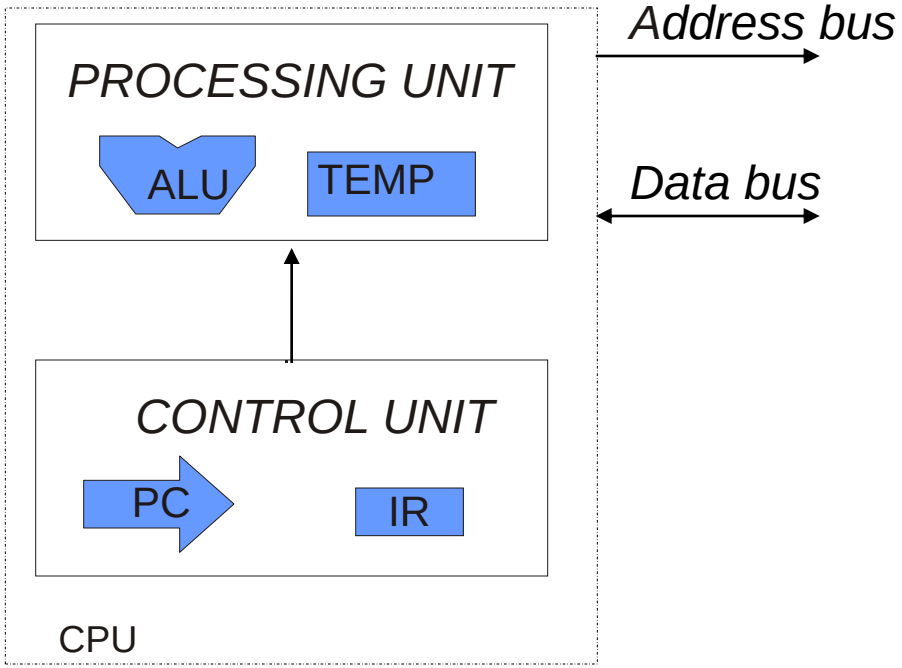
CPU - Tổ chức

ALU (Arithmetic Logic Unit)

Đơn vị luận lý số học

IR (Instructon Register) : Thanh ghi chỉ thị

PC (Program Counter) : Bộ đếm chương trình



- ALU thực hiện các phép toán số học và logic
- PC chứa địa chỉ của lệnh kế tiếp trong bộ nhớ. Mỗi lần nhận lệnh PC tự động tăng lên
- IR chứa lệnh đang thực hiện
- TEMP : Các thanh ghi tạm chứa toán hạng để ALU thực hiện

CPU - Hoạt động

Nạp mã lệnh
Fetch

↓

Giải mã lệnh
Decode

↓

Thực hiện lệnh
Execution

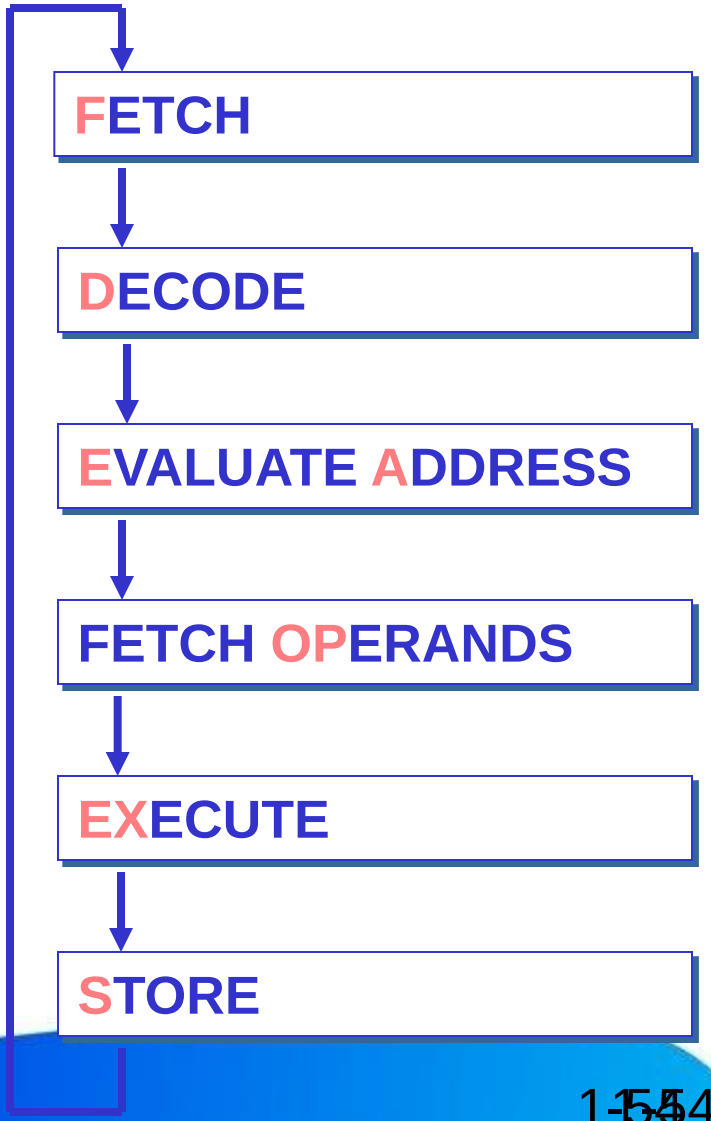
↓

Instruction Processing

Hoạt động của vi xử lý là 1 vòng lặp vô tận các quá trình nhận lệnh- giải mã lệnh- thực hiện lệnh

Instruction : lệnh (chỉ thị)

Instruction Processing (chi tiết hơn)



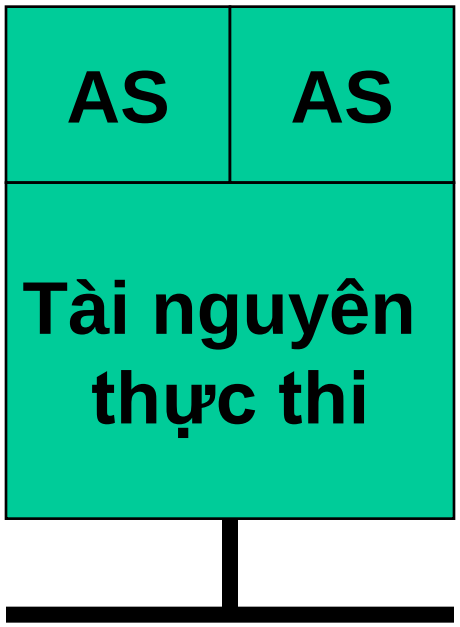
CISC & RISC

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

MỨC ĐỘ XỬ LÝ SONG SONG

- **Multithreading (Intel hyperthreading)**
  - ➢ **Fetch from each thread in alternating cycles**
  - ➢ **Share processor pipeline between two threads**
- **Multiple processor cores per chip**
- **Multiple processor chips per system**

Multithreading (Intel hyperthreading)

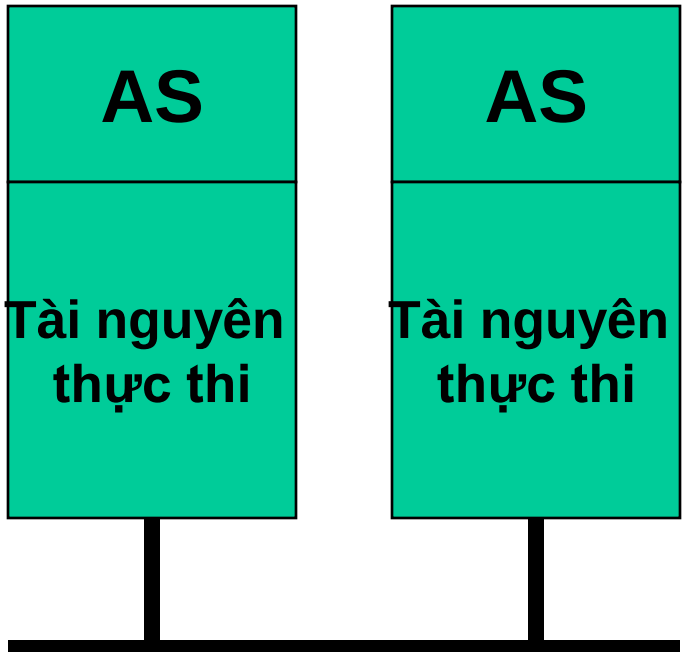


Mô phỏng mỗi bộ xử lý vật lý như là 2 bộ xử lý logic, tài nguyên vật lý sẽ được chia sẻ và có cấu trúc chung giống hệt nhau cho cả 2 bộ xử lý logic.
Hệ điều hành và phần mềm coi nó như 2 bộ xử lý song song

Kết quả là tốc độ trung bình tăng lên khoảng 20-30%.

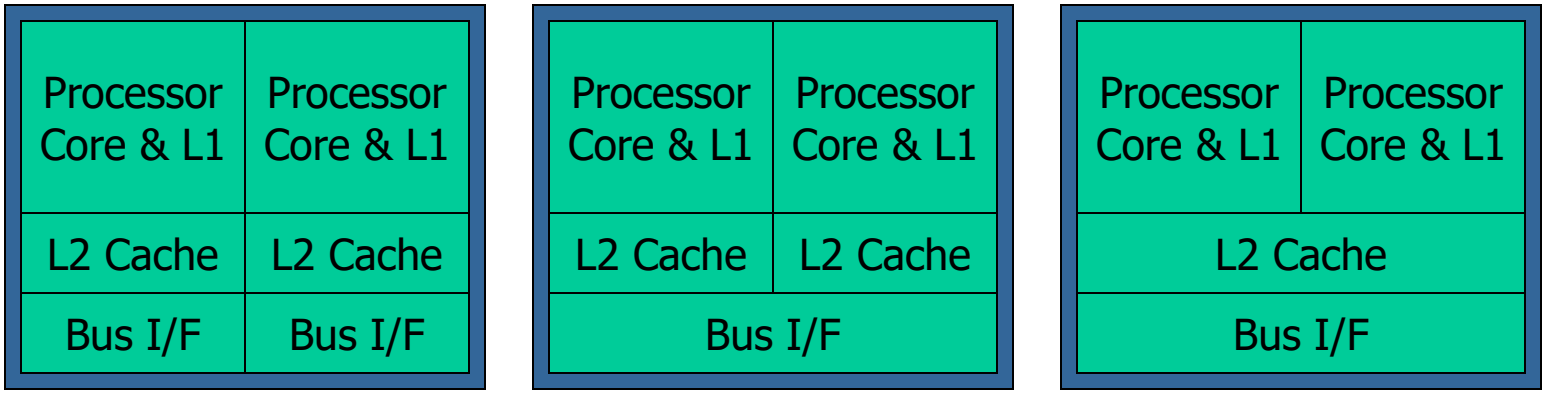
Hạn chế ?

BỘ XỬ LÝ ĐA NHÂN (Multiple Processor Cores per Chip)



Bộ xử lý đa nhân được kiến trúc bởi 2 hay nhiều CPU vật lý thực sự, có khả năng thực hiện đồng thời 2 chuỗi lệnh song song thực sự .

Multiple Processor Cores per Chip



Intel Pentium D

AMD Athlon X2
Intel Dual Core

IBM Power5
Intel Core2 Duo

Increased level of integration per package/chip

Perception of 2x performance (not always reality)

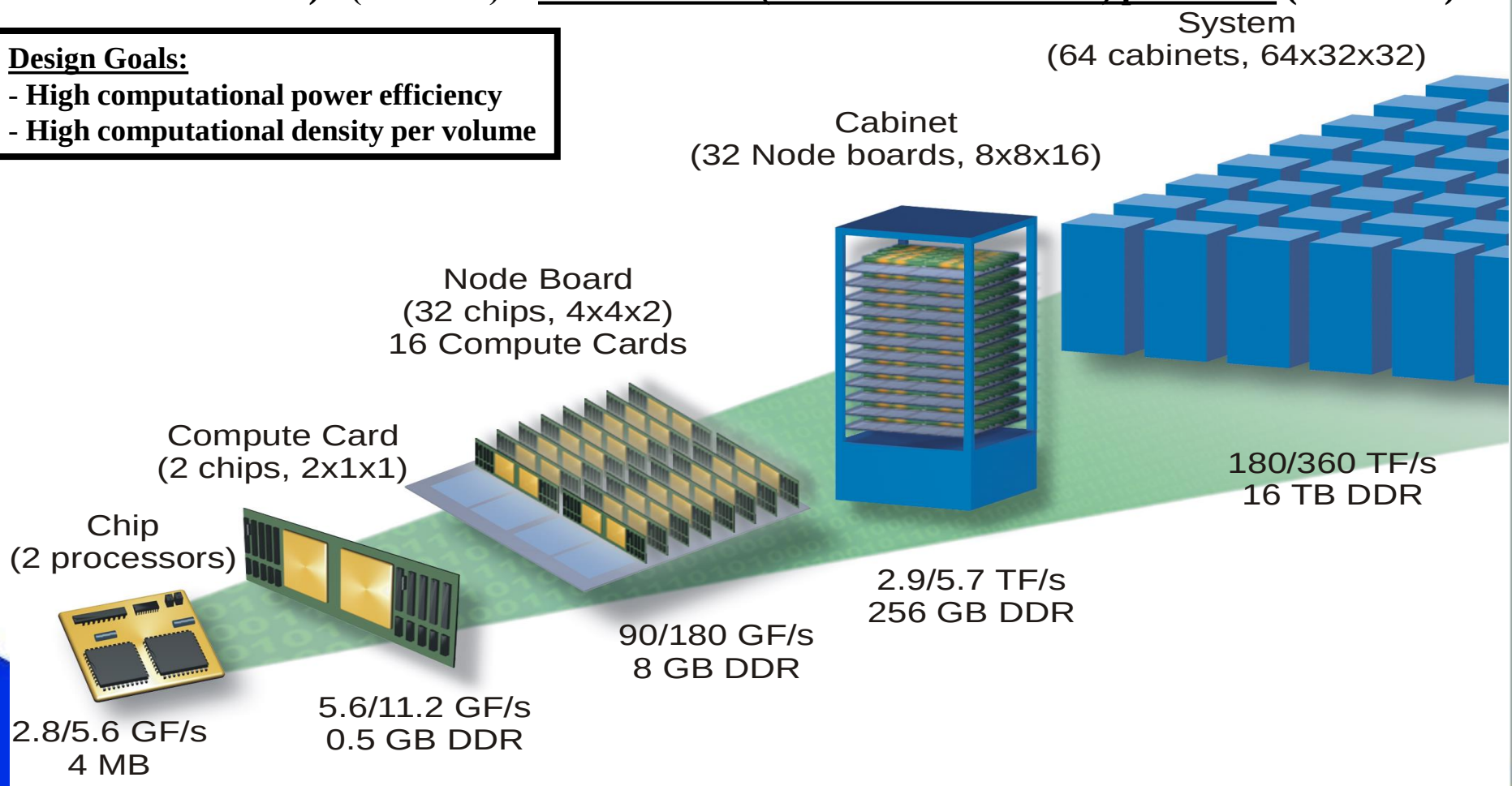
Can share nothing (Intel), Bus interface (AMD), L2 (IBM)

Multiple processor chips per system

IBM Blue Gene/L

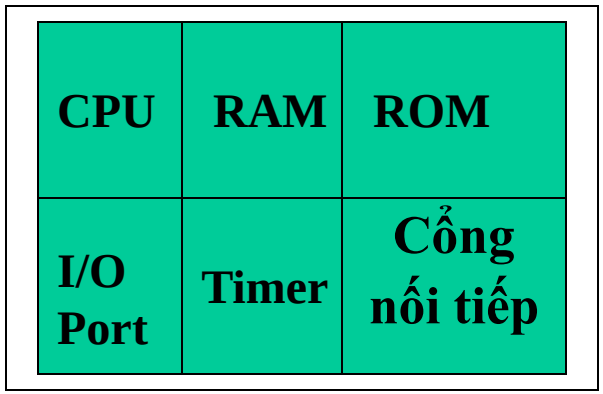
(2 processors/chip) • (2 chips/compute card) • (16 compute cards/node board) • (32 node boards/tower) • (64 tower) = <u>128k = 131072 (0.7 GHz PowerPC 440) processors</u> (64k nodes)

<u>Design Goals:</u>
- High computational power efficiency
- High computational density per volume



VI ĐIỀU KHIỂN - MICROCONTROLLER

Tích hợp nhiều chức năng trong 1 chip : CPU,RAM, ROM, I/O port …
Motorola 6811, Intel 4048, Intel8051, PIC 16x, Atmel AVR, Zilog Z80 …



tất cả bên trong 1 chip

Khái niệm vi xử lý được sử dụng bao gồm cả vi điều khiển hay nói cách khác vi điều khiển là 1 hệ vi xử lý đặc biệt

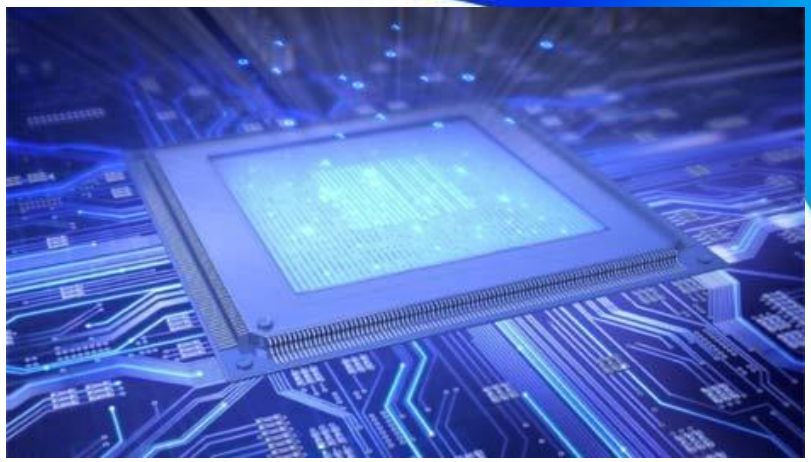
VI XỬ LÝ & VI ĐIỀU KHIỂN

Vi xử lý

- CPU chip riêng biệt. RAM, ROM, I/O, Timer bên ngoài
- Lượng ROM, RAM, I/O Ports tùy ý
- Giá thành cao
- Đa năng
- Đa mục đích

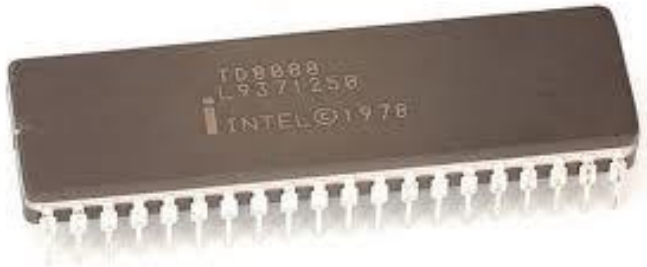
Vi điều khiển

- CPU, RAM, ROM, I/O & Timer nằm trên cùng 1 chip
- Cố định lượng ROM, RAM, I/O Ports trên chip
- Thích hợp cho các ứng dụng:
  - ✓giá cả thấp
  - ✓năng lượng tiêu thụ thấp
  - ✓không gian hạn chế
- Đơn mục đích



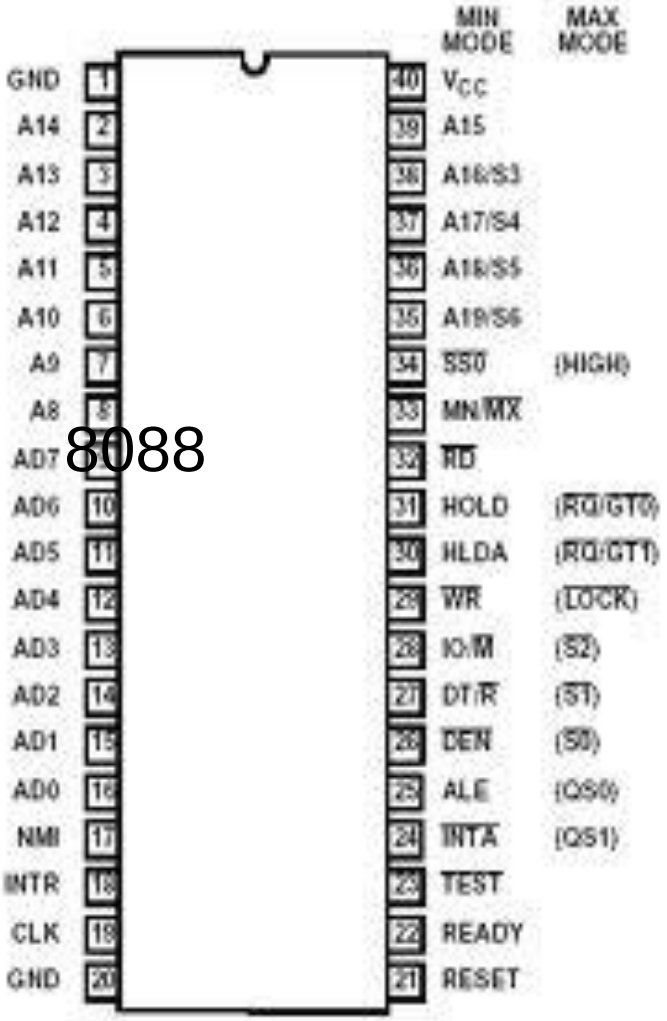
Chương 3
Kiến trúc phần cứng của hệ thống vi xử lý

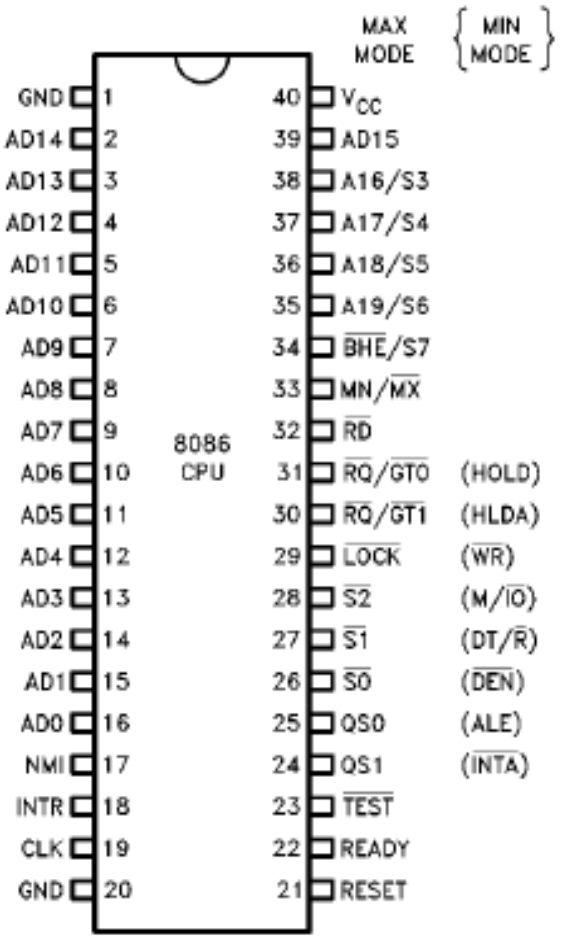
3.1 Bộ vi xử lý 8088



8088 White Box XT
4.77 MHz
640KB Memory
20MB HDD

INTEL 8088/8086

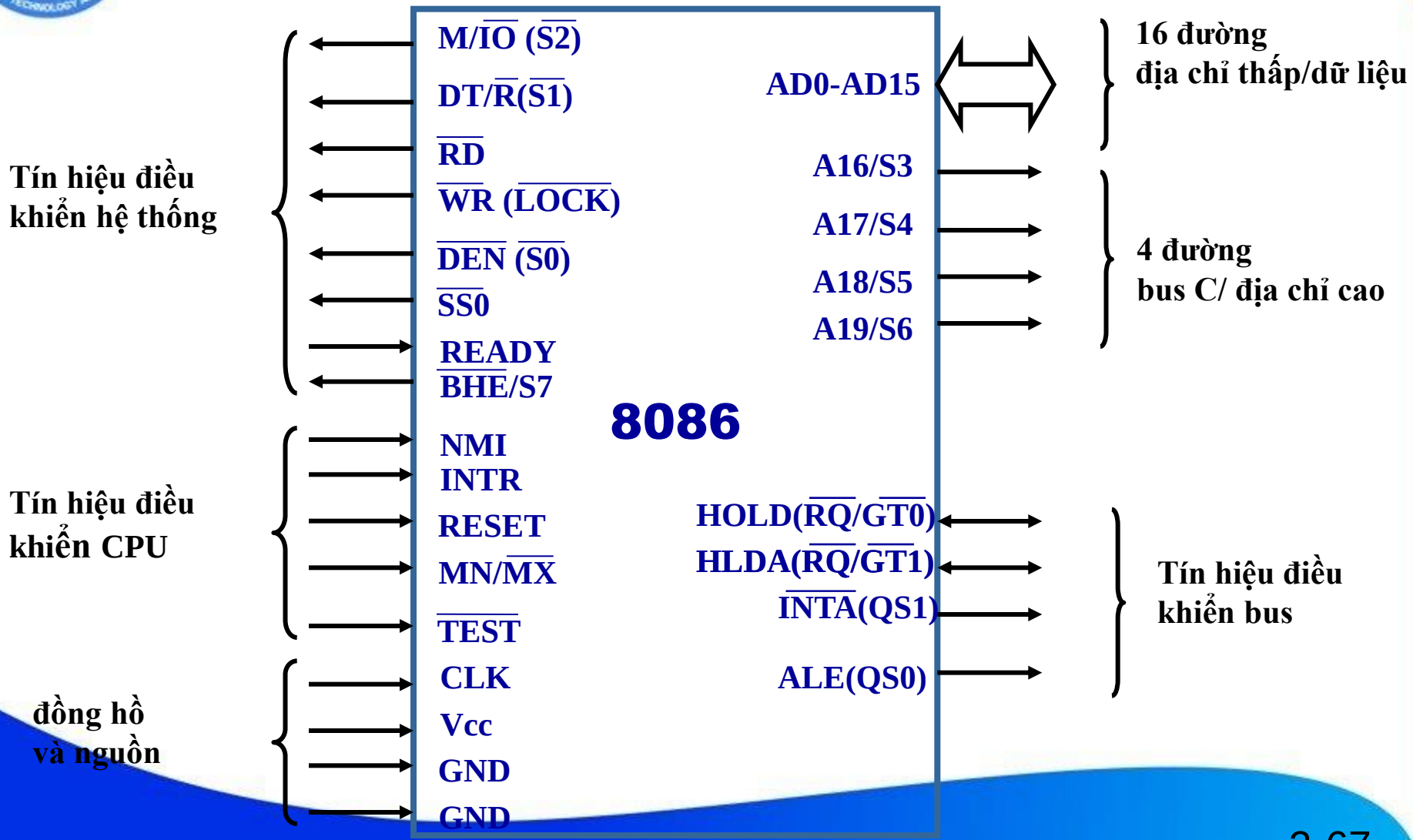




8088 vs. 8086

Hàng đợi lệnh	**:**	**4 byte**	**6 byte**
Data Bus	**:**	**8 bit**	**16 bit**
Truy cập memory	**:**	$IO/\overline{M}=1$	
Truy cập I/O	**:**	$IO/\overline{M}=0$	

CÁC CHÂN TÍN HIỆU 8086



Phân kênh và đệm cho các bus

Vì sao phải phân kênh ?

- Các bus địa chỉ và dữ liệu dùng chung chân → phải tách phần địa chỉ và phần dữ liệu ra

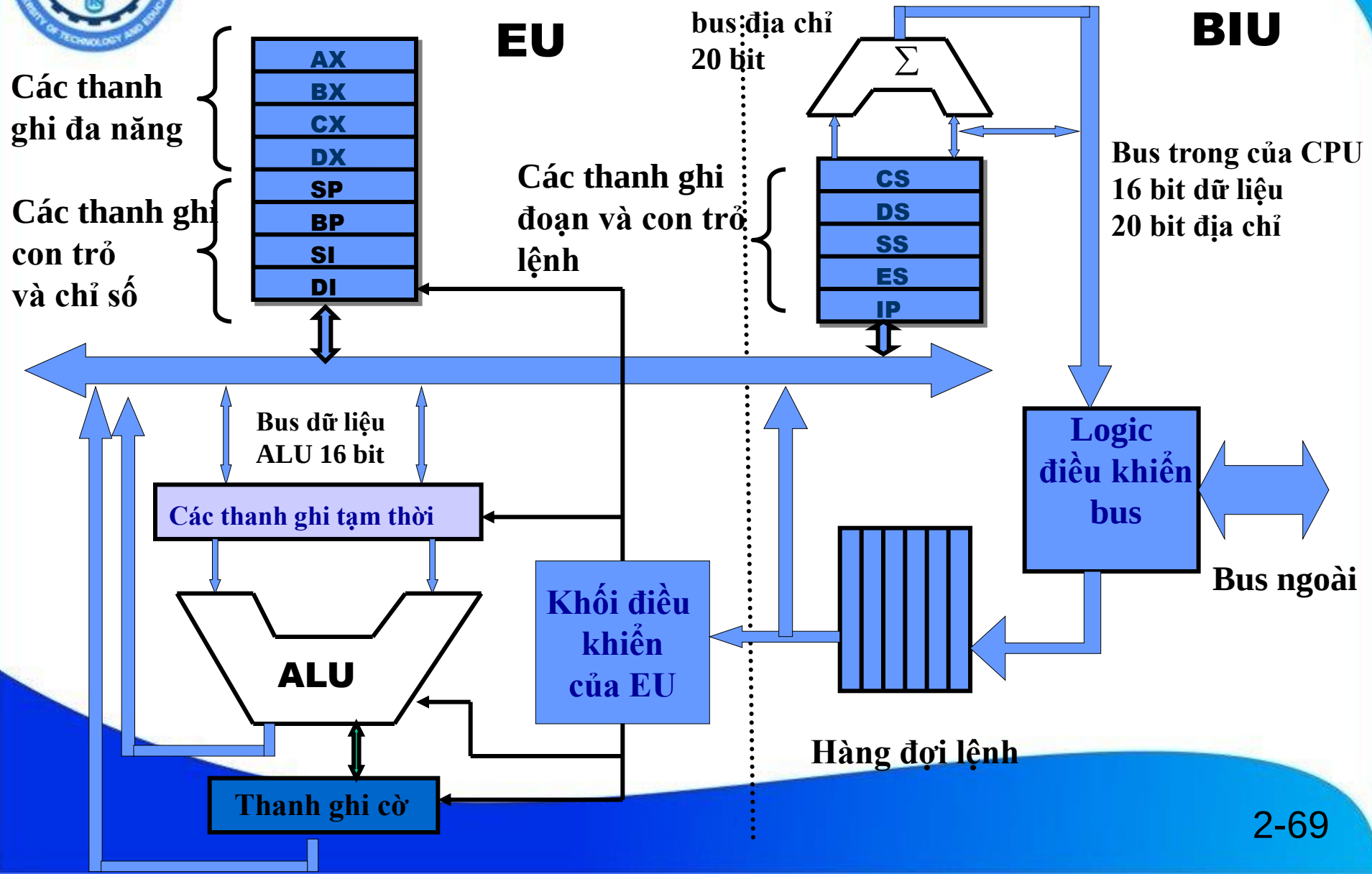
Vì sao phải đệm bus ?

- Dòng dữ liệu đi vào tất cả các thiết bị ghép trên bus → chia nhỏ dòng điện
- Nâng cao khả năng tải của bus

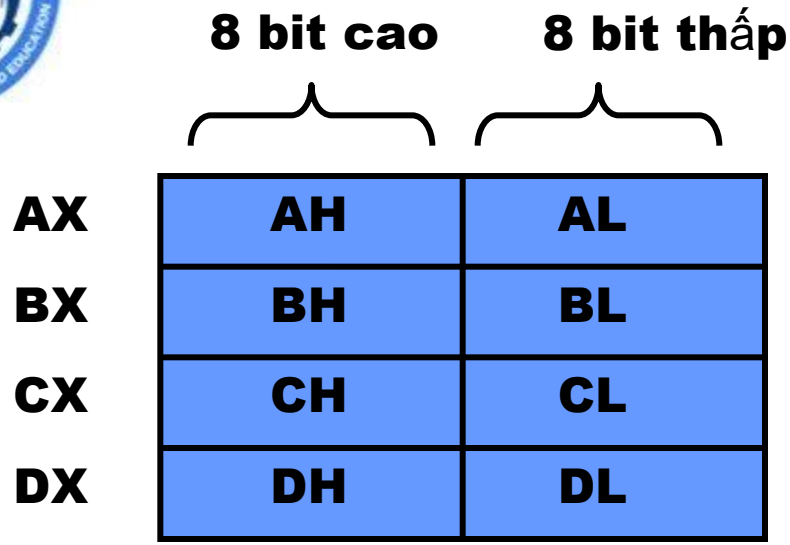
Các vi mạch phân kênh và đệm:

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

SƠ ĐỒ KHỐI


EU
BIU
bus địa chỉ
20 bit
Các thanh
ghi đa năng
AX
BX
CX
DX
SP
BP
SI
DI
Các thanh ghi
con trỏ
và chỉ số
Các thanh ghi
đoạn và con trỏ
lệnh
CS
DS
SS
ES
IP
Σ
Bus trong của CPU
16 bit dữ liệu
20 bit địa chỉ
Bus dữ liệu
ALU 16 bit
Các thanh ghi tạm thời
ALU
Khối điều
khiển
của EU
Logic
điều khiển
bus
Bus ngoài
Hàng đợi lệnh
Thanh ghi cờ

CÁC THANH GHI



- **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
- **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 cổng trong các lệnh vào ra dữ liệu trực tiếp (IN/OUT)

CÁC THANH GHI

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

- Đảm bảo tính tương thích ngược : các chương trình viết cho 8088/8086 vẫn chạy được trên các bộ xử lý 80386 hoặc cao hơn
- Các chương trình viết cho 80386 có thể không chạy được trên máy tính có bộ xử lý thấp hơn.

CÁC THANH GHI

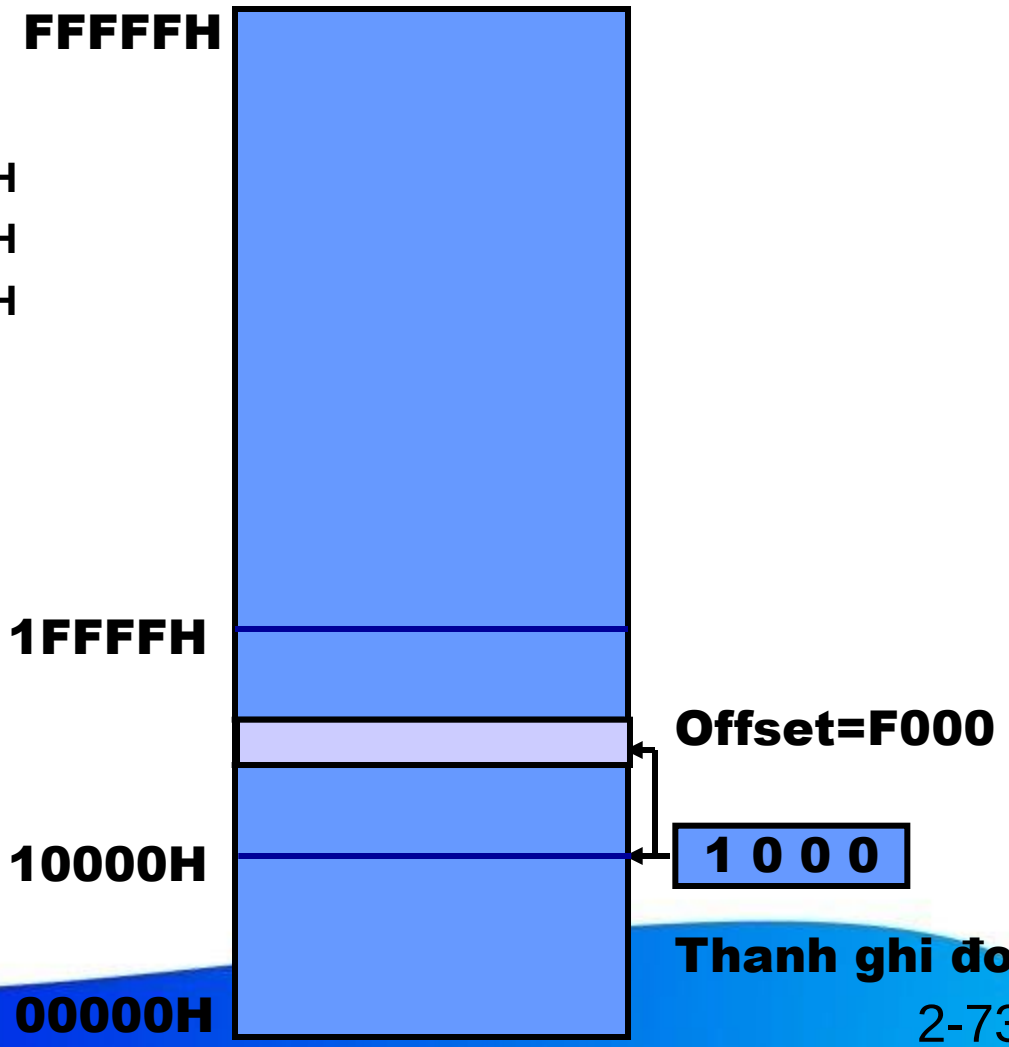
Thanh ghi đoạn : các cặp thanh ghi ngầm định

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

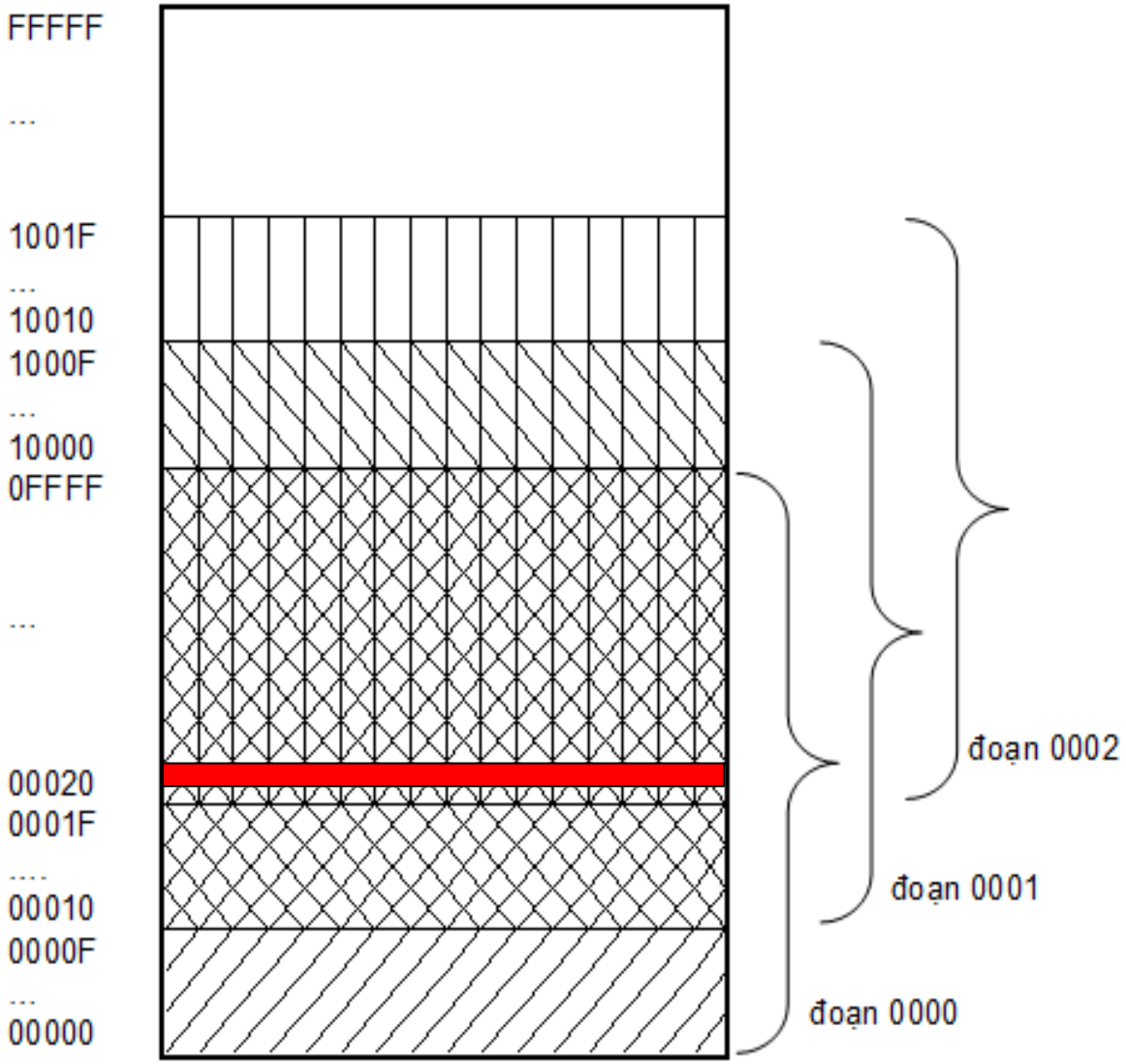
CÁC THANH GHI ĐOẠN

Tổ chức của bộ nhớ 1 Mbytes

- **Đoạn bộ nhớ (segment)**
  - **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**



CÁC ĐOẠN CHỒNG LÊN NHAU



Ô nhớ 00020h (địa chỉ vật lý)

Có các địa chỉ logic

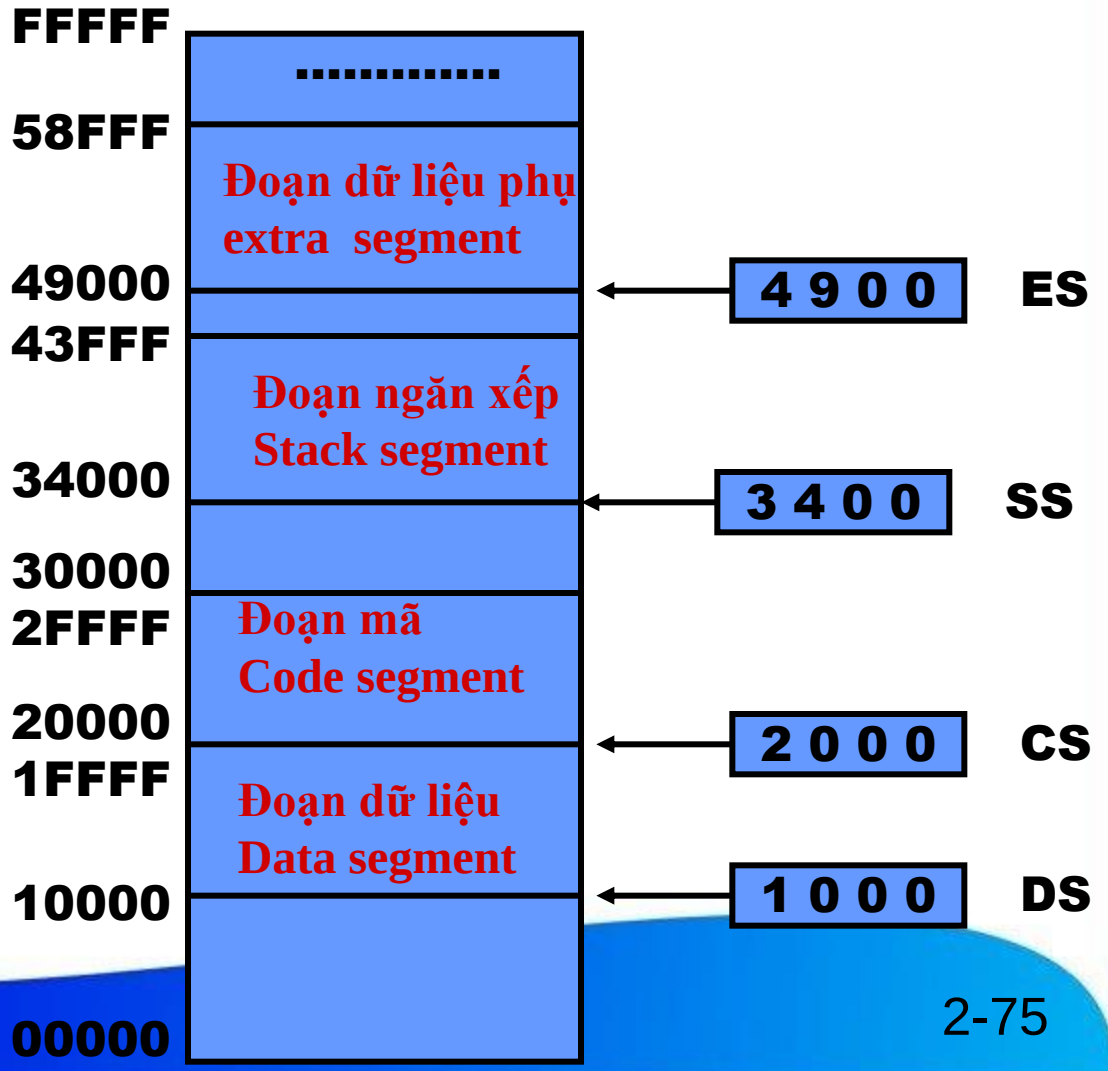
0000h : 0020h

0001h : 0010h

0002h : 0000h

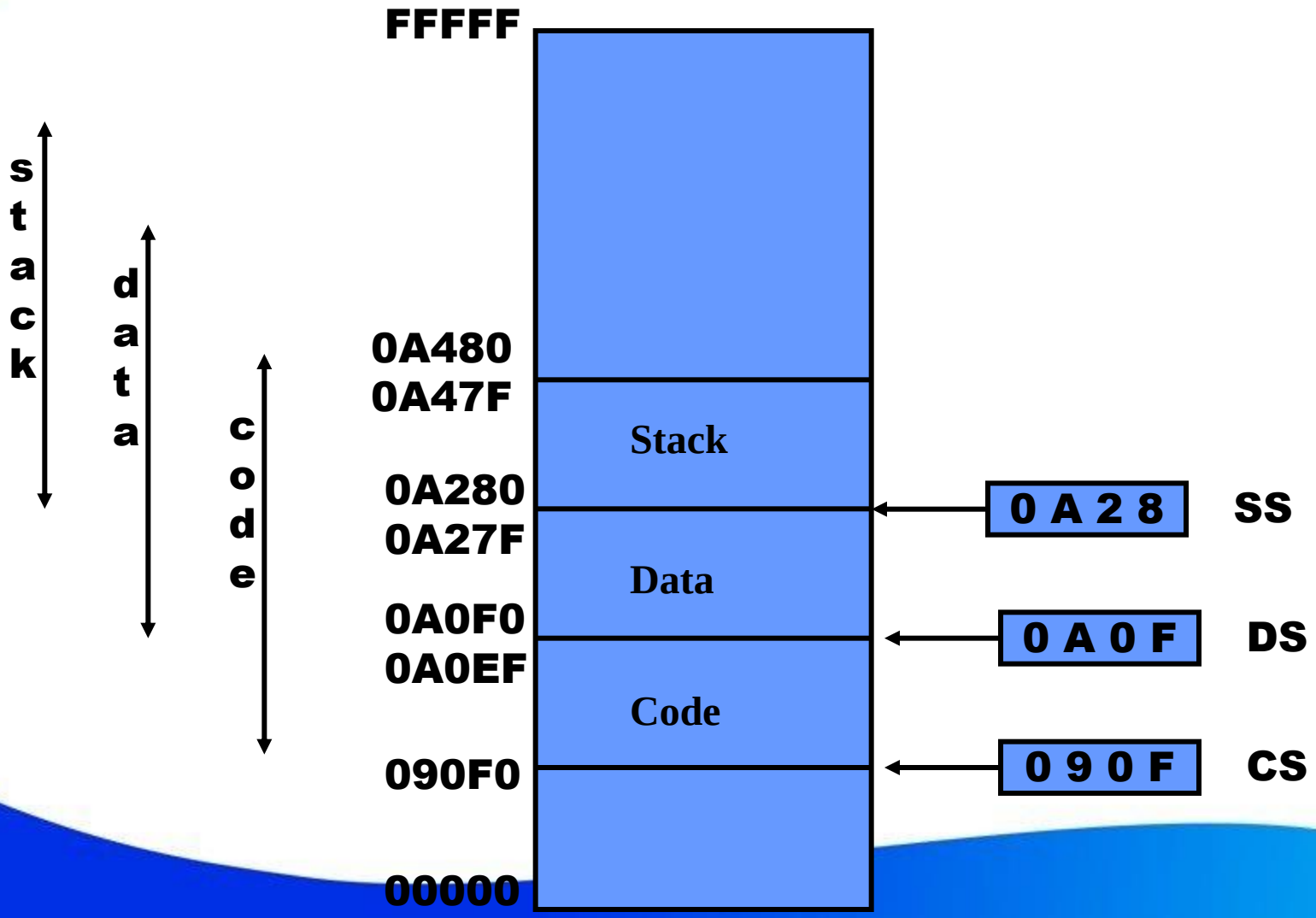
CÁC THANH GHI ĐOẠN

Các thanh ghi đoạn: chứa địa chỉ đoạn



CÁC THANH GHI ĐOẠN

Các đoạn có thể chồng lên nhau



THANH GHI CON TRỎ & CHỈ SỐ

Chứa địa chỉ lệch (offset)

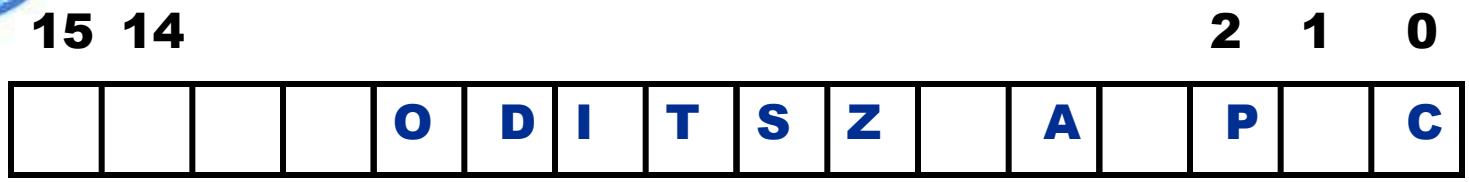
- Con trỏ lệnh IP (instruction pointer): chứa địa chỉ lệnh tiếp theo trong đoạn mã lệnh CS.
  - CS:IP
- Con trỏ cơ sở BP (Base Pointer): chứa địa chỉ của dữ liệu trong đoạn ngăn xếp SS hoặc các đoạn khác
  - SS:BP
- Con trỏ ngăn xếp SP (Stack Pointer): chứa địa chỉ hiện thời của đỉnh ngăn xếp
  - SS:SP
- 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

THANH GHI ĐOẠN

Các cặp thanh ghi 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

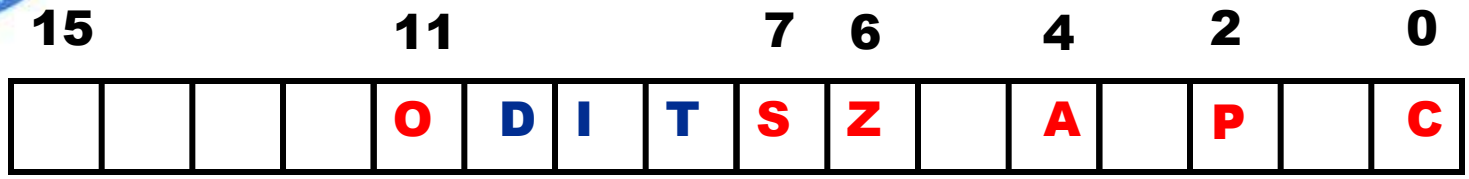
THANH GHI CỜ



9 bit được sử dụng

- 6 cờ trạng thái : CF, PF, AF, ZF, SF, OF
- 3 cờ điều khiển :DF, IF, TF

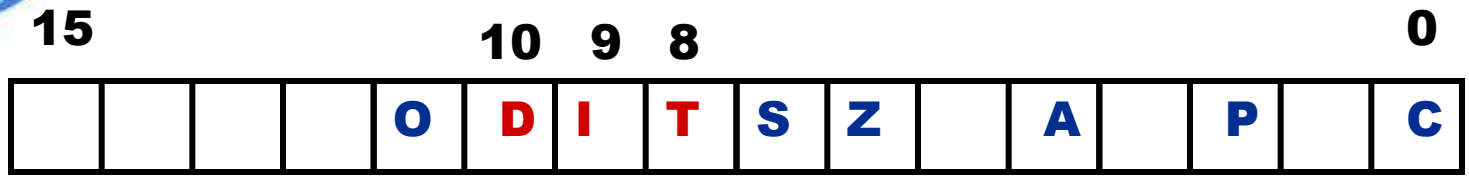
THANH GHI CỜ



6 cờ trạng thái:

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

THANH GHI CỜ

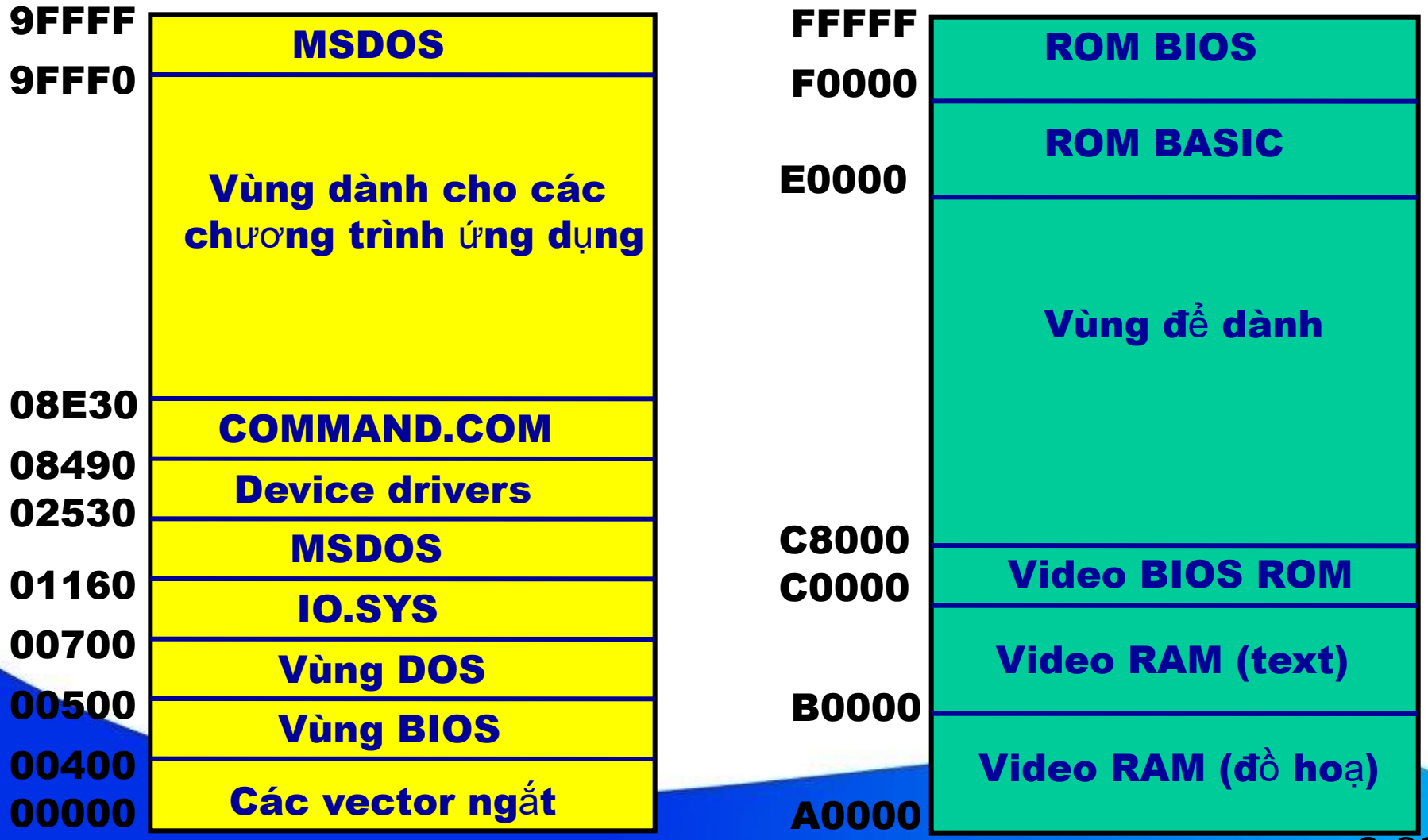


3 cờ điều khiển

- 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)

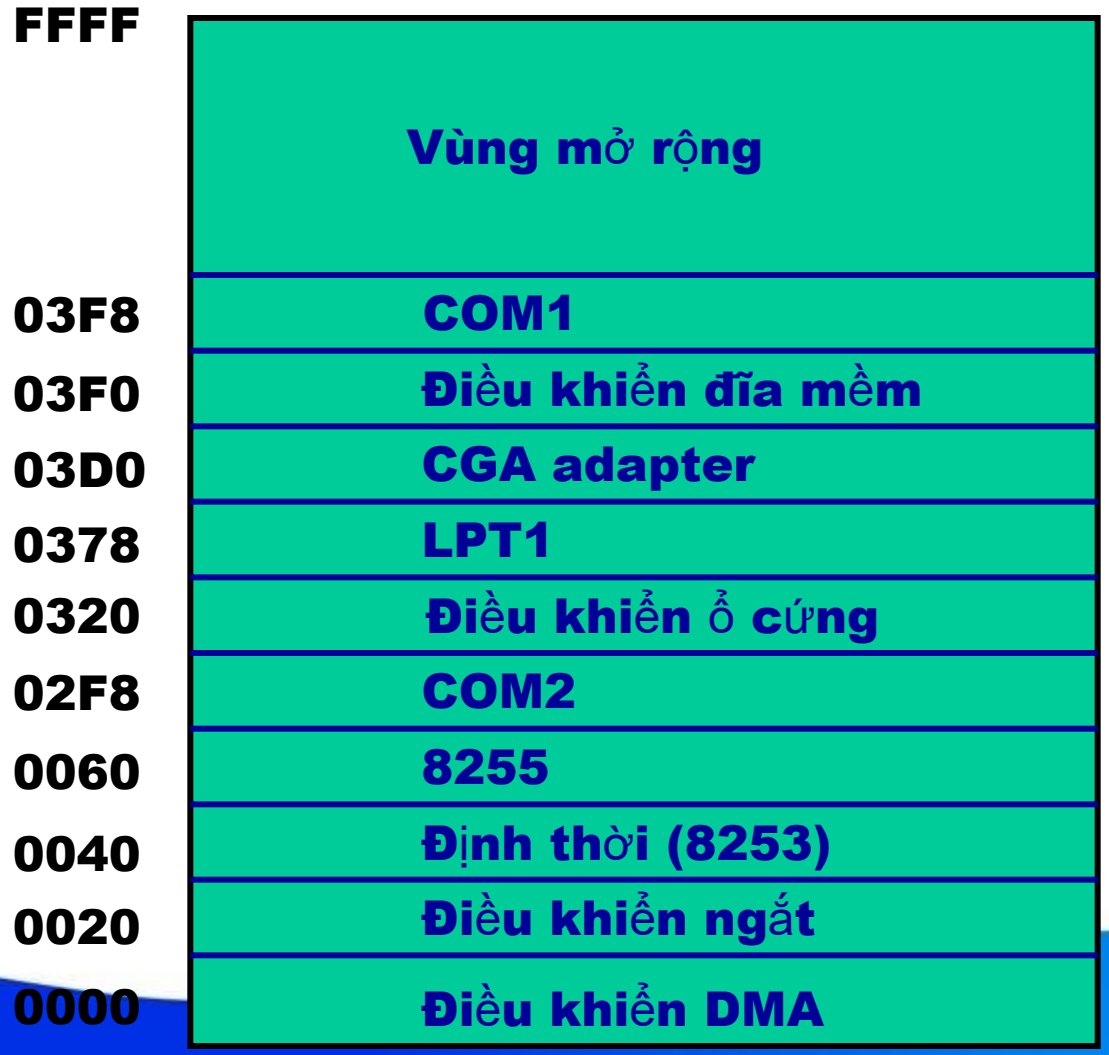
BẢN ĐỒ BỘ NHỚ CỦA MÁY TÍNH PC-IBM

BẢN ĐỒ BỘ NHỚ CỦA MÁY TÍNH PC-IBM



BẢN ĐỒ ĐỊA CHỈ I/O CỦA MÁY TÍNH IBM/PC

Địa chỉ: 0000H –FFFFH, M/IO = 0



3.2 THIẾT KẾ BỘ NHỚ CHO HỆ VI XỬ LÝ

CÁC LOẠI BỘ NHỚ

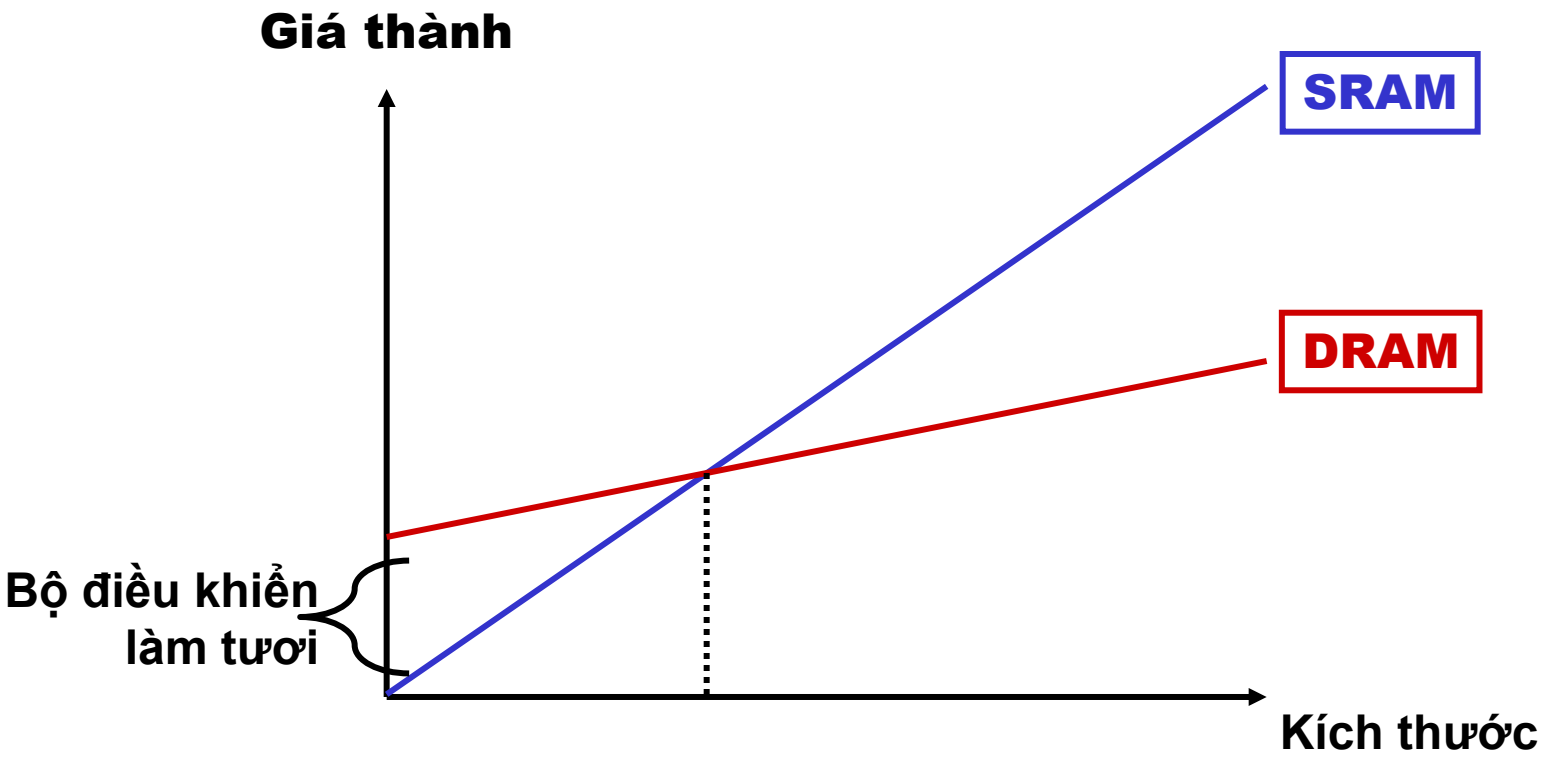
Bộ nhớ không bị mất dữ liệu (non-volatile)

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

Bộ nhớ bị mất dữ liệu (volatile)

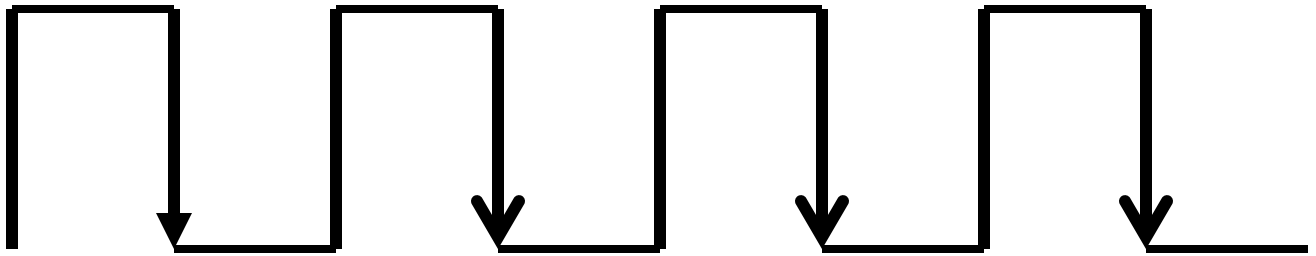
- SRAM (Static RAM)
- SBSRAM (Synchronous Burst RAM)
- DRAM (Dynamic RAM)
- FPDRAM (Fast Page mode Dynamic RAM)
- EDO DRAM (Extended Data Out Dynamic RAM)
- SDRAM (Synchronous Dynamic RAM)
- DDR-SDRAM (Double Data Rate SDRAM)
- RDRAM (Rambus Dynamic RAM)

SRAM & DRAM

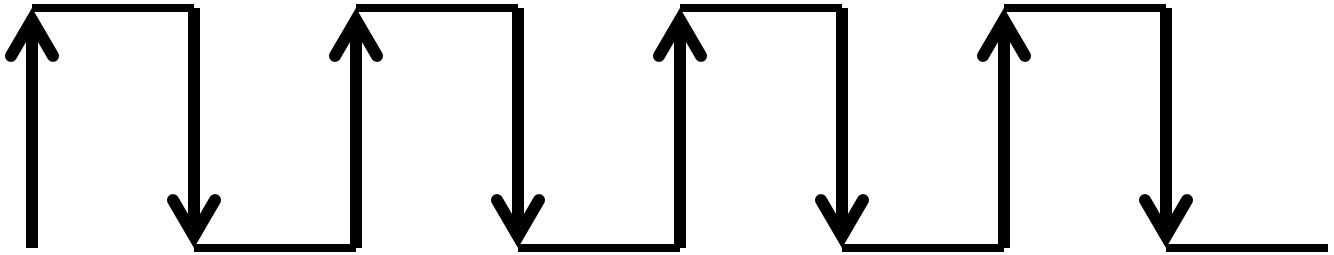


SDR & DDR

SDR SDRAM



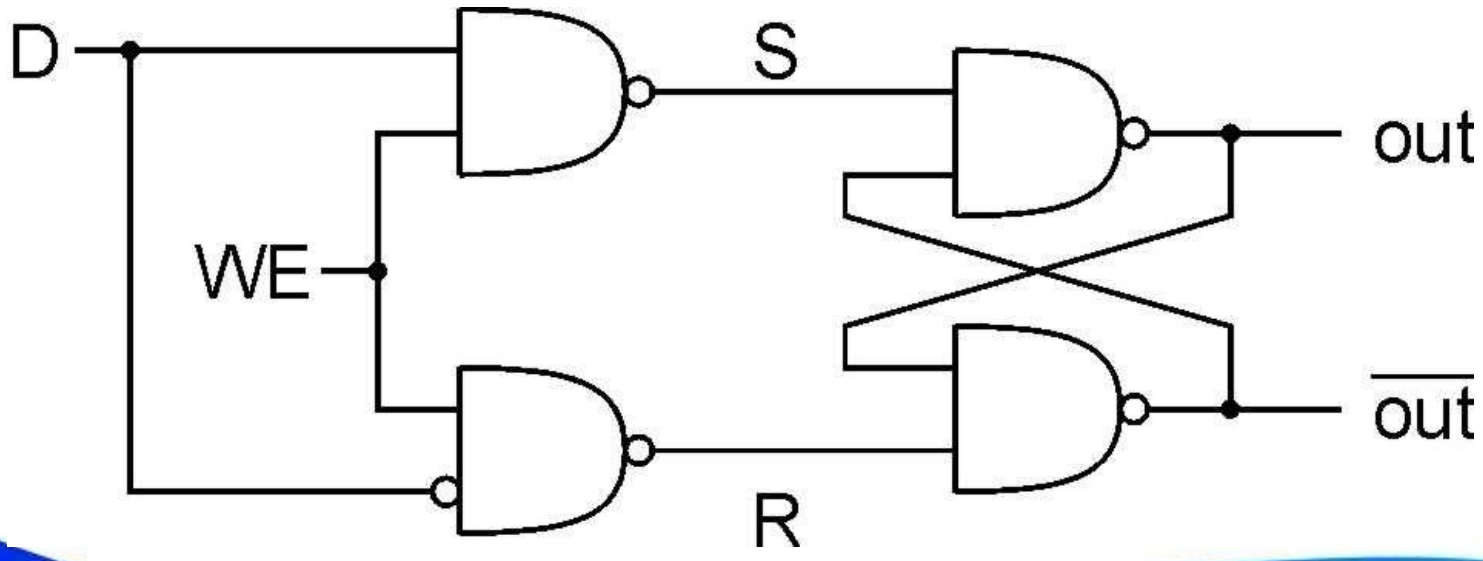
DDR SDRAM



Gated D-Latch - Bộ nhớ 1 bit

Hai ngõ vào : D (data) & WE (write enable)

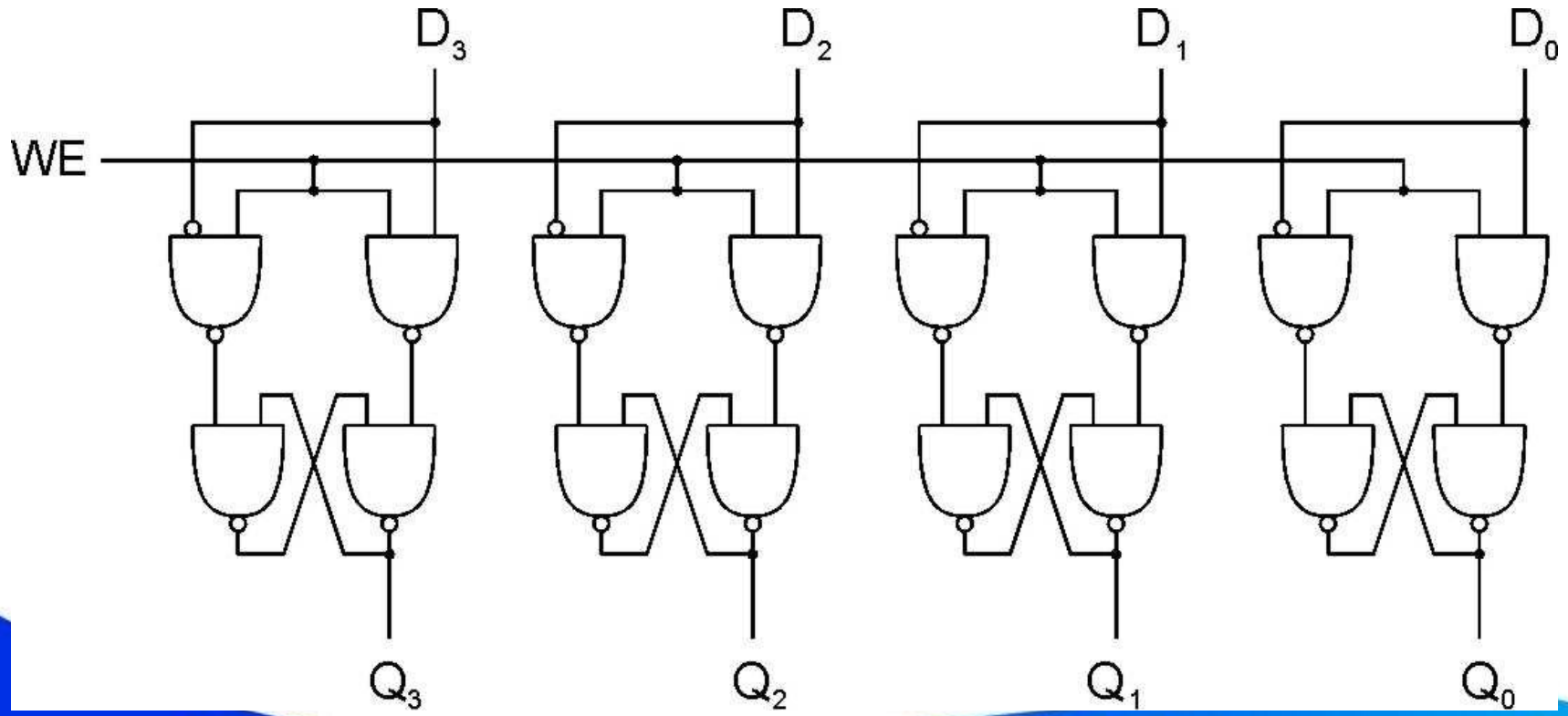
- **WE = 1, bộ chốt thiết lập giá trị của D**
  - **S = NOT(D), R = D**
- **WE = 0, bộ chốt chứa giá trị đã lưu trữ trước đó**
  - **S = R = 1**



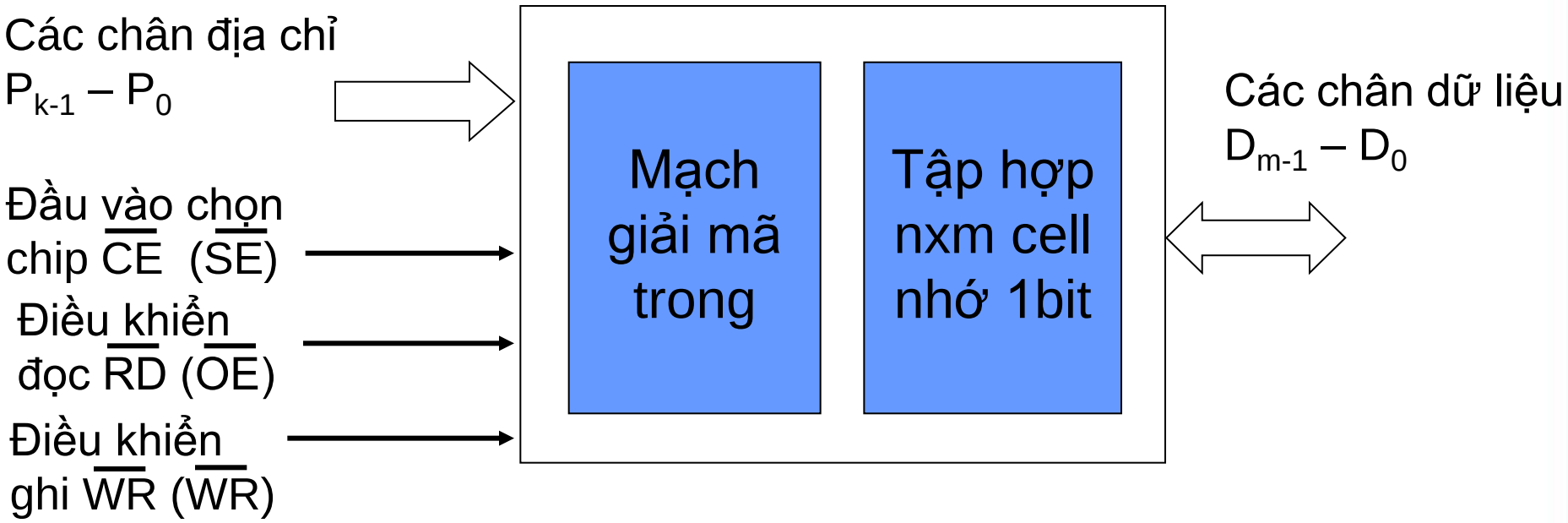
Register – Thanh ghi

Thanh ghi lưu trữ nhiều bit.

- **Sử dụng nhiều D-latches, tất cả được điều khiển bởi WE.**
- **WE=1, n-bit tại D lưu trữ vào thanh ghi.**



CẤU TRÚC & HOẠT ĐỘNG



m thường bằng 8,16,32

Tuỳ thuộc vào dung lượng bộ nhớ (tức là n) sẽ có số đường địa chỉ tương ứng . VD : n= 8 → k=3 ; n=256 → k=8

CẤU TRÚC & HOẠT ĐỘNG

DUNG LƯỢNG 1 CHIP NHỚ

- Một chíp nhớ được xem như 1 mảng gồm n ô nhớ
- Dung lượng 1 chip nhớ thường biểu diễn nxm
- m chính là số chân dữ liệu của chip nhớ (m=4,8,16 ..)
- $\log_2(n) = p$ là số chân địa chỉ của chip nhớ

HOẠT ĐỘNG ĐỌC

- Đưa các tín hiệu địa chỉ thích hợp vào các chân địa chỉ
- Đưa tín hiệu 0 vào chân điều khiển đọc (RD)
- Đưa tin hiệu 0 vào chân chọn chíp

➢ Khi đó m bit số liệu của thanh ghi có địa chỉ tương ứng xuất hiện ở các chân dữ liệu

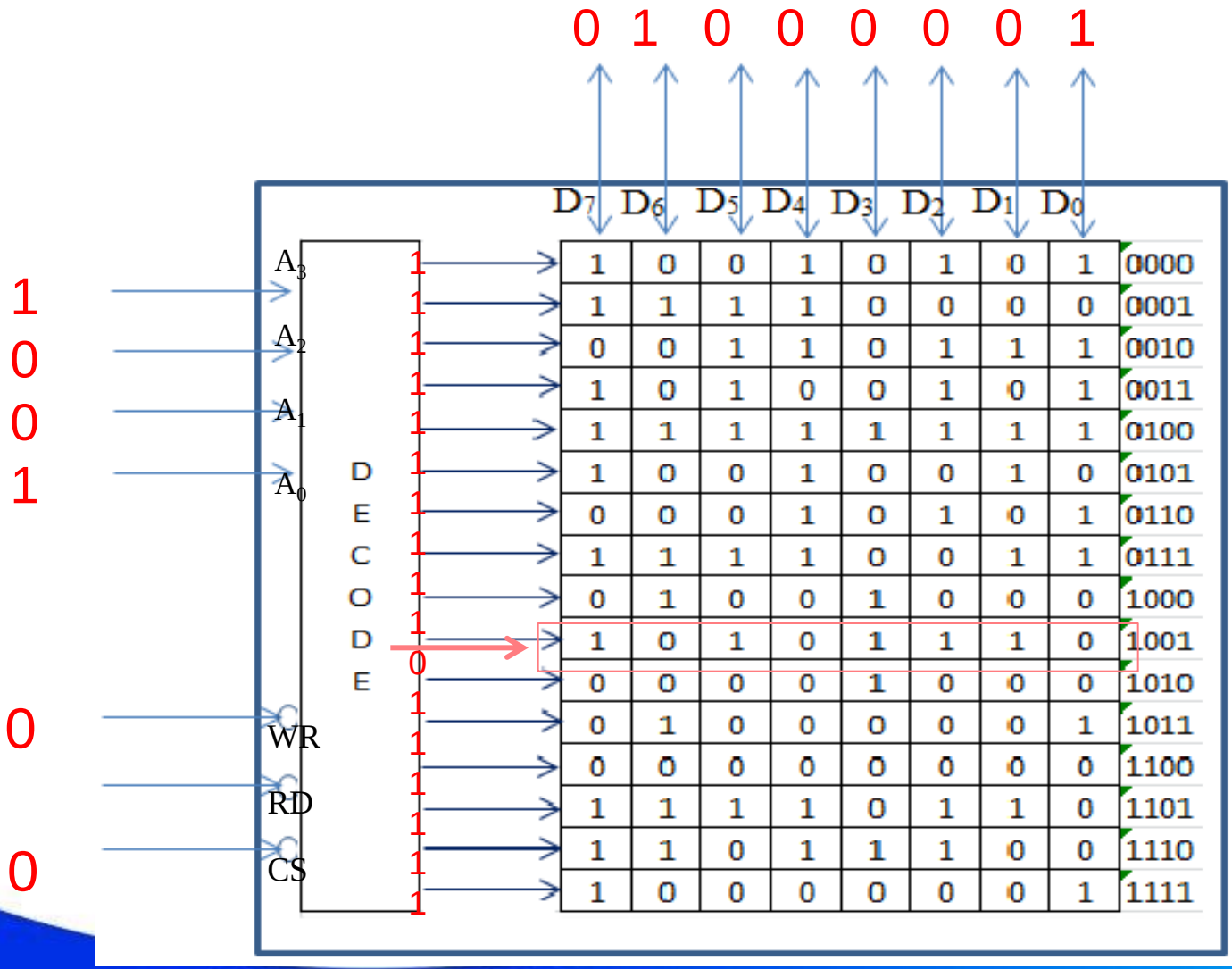
HOẠT ĐỘNG GHI

- Đưa các tín hiệu địa chỉ thích hợp vào các chân địa chỉ
- Đưa các số liệu cần ghi vào các chân dữ liệu
- Đưa tín hiệu 0 vào chân điều khiển ghi (RW)
- Đưa tin hiệu 0 vào chân chọn chíp

➢ Khi đó m bit số liệu được lưu ở thanh ghi có địa chỉ tương ứng

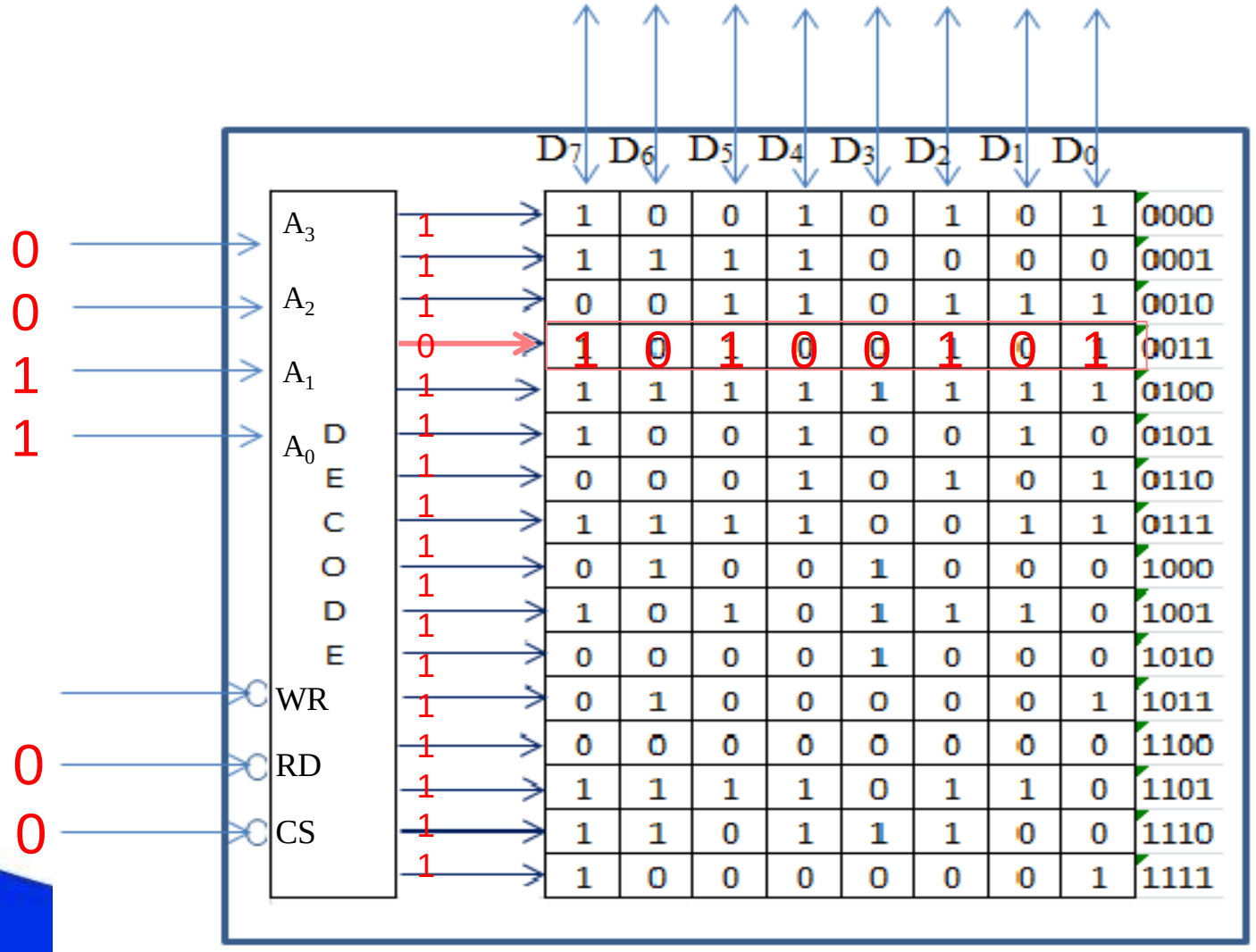
GHI BỘ NHỚ

Ghi giá trị 01000001 vào bộ nhớ tại địa chỉ 1001



ĐỌC BỘ NHỚ

Đọc bộ nhớ tại địa chỉ 0011



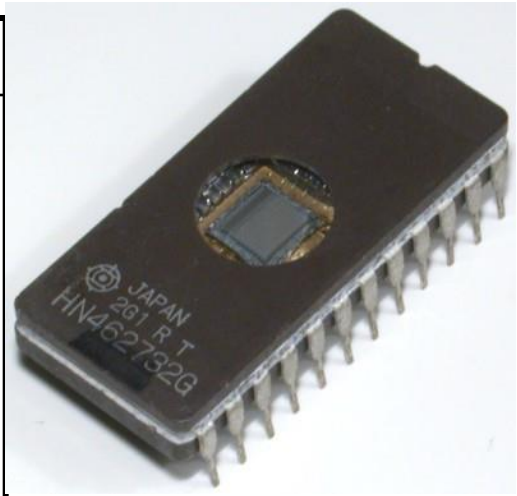
EPROM

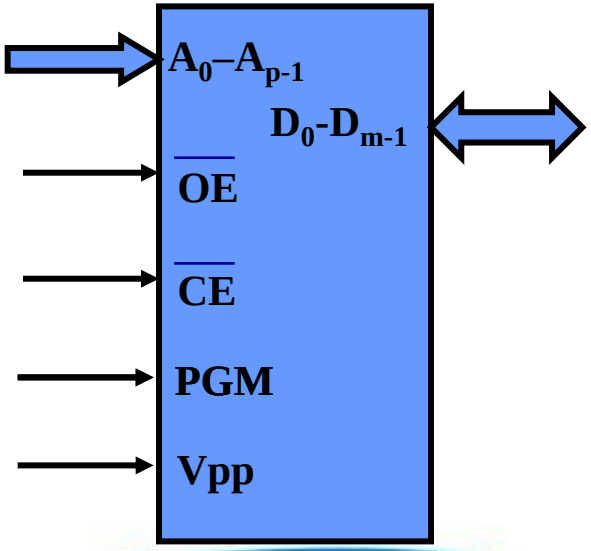
Ghi vào EPROM

- Ghi vào Eprom được gọi là lập trình cho Eprom
- Sử dụng thiết bị chuyên dụng gọi là bộ nạp EPROM (Program)
- Chân Vpp được cấp điện áp gọi là điện áp lập trình (thường là 12 V)
- Dữ liệu tại các chân dữ liệu sẽ được ghi vào ô nhớ xác định bởi các chân địa chỉ khi có xung đưa vào chân PGM

EPROM họ 27x

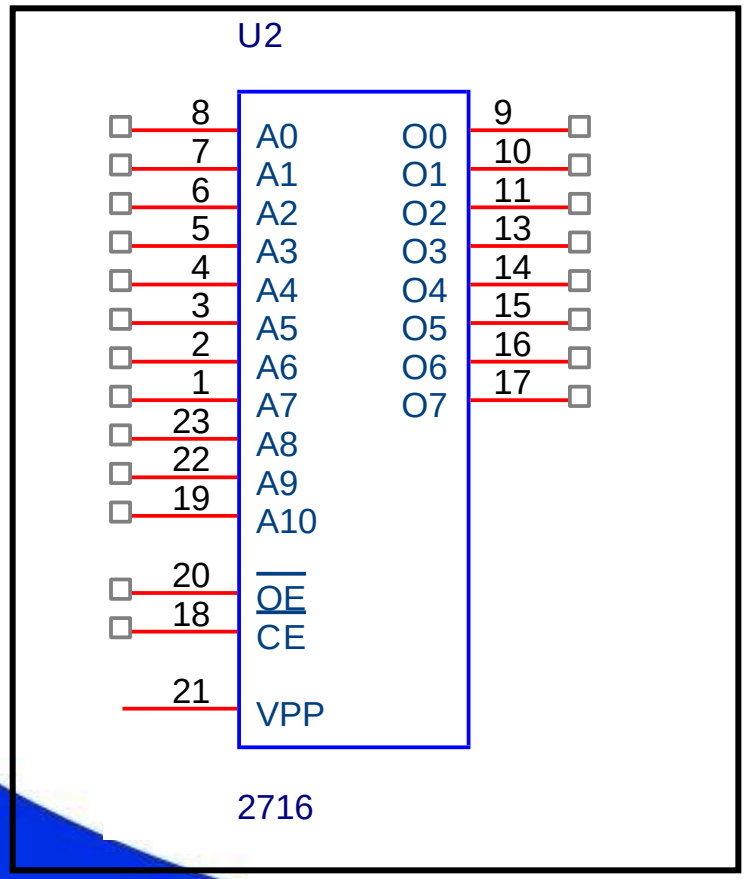
Số hiệu	Dung lượng
2708	1Kx8
2716	2Kx8
2732	4Kx8
2764	8Kx8
27128	16Kx8
27256	32Kx8
27512	64Kx8



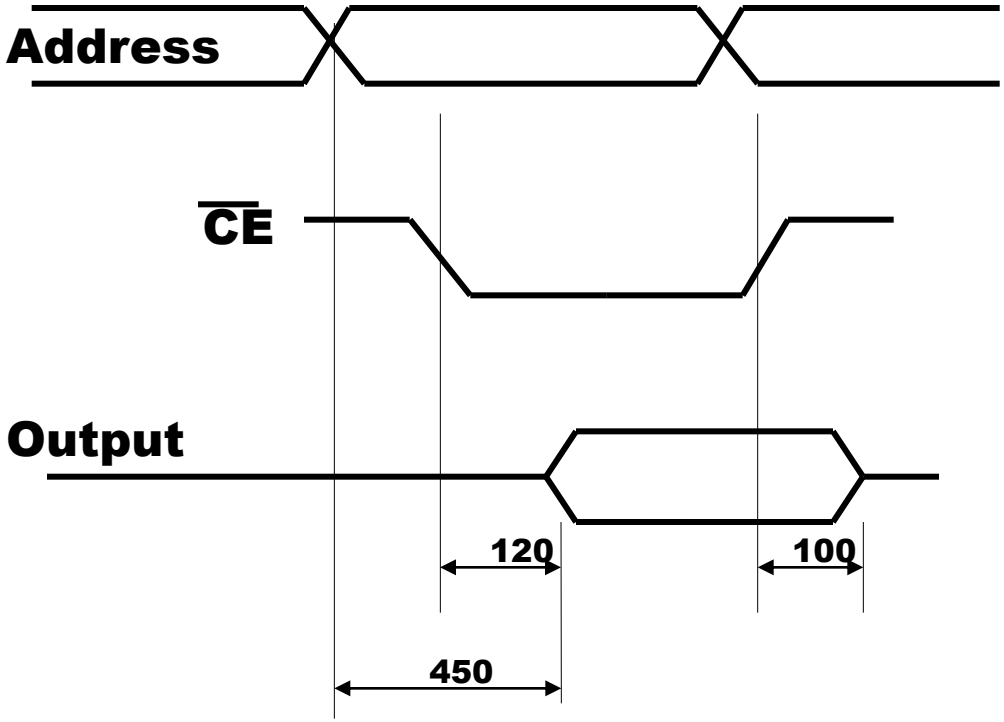


EPROM 2716

Sơ đồ chân



Chu kỳ đọc



SO SÁNH 1 SỐ 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

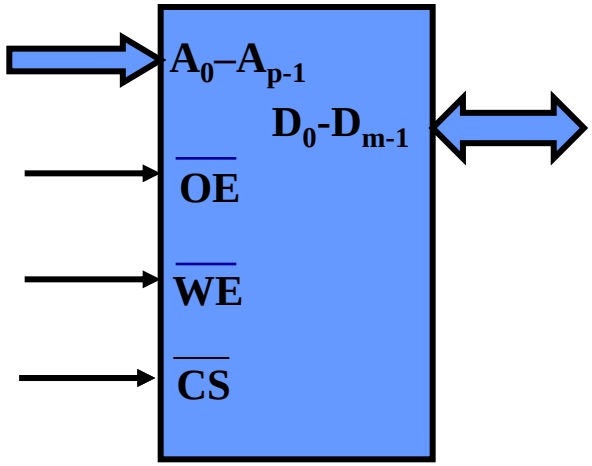
SRAM

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

Ứng dụng:

- Bộ nhớ nhỏ và nhanh (cache)



SRAM HỌ 62X

Số hiệu	Dung lượng
6208	1Kx8
6216	2Kx8
6232	4Kx8
6264	8Kx8
62128	16Kx8
62256	32Kx8
62512	64Kx8

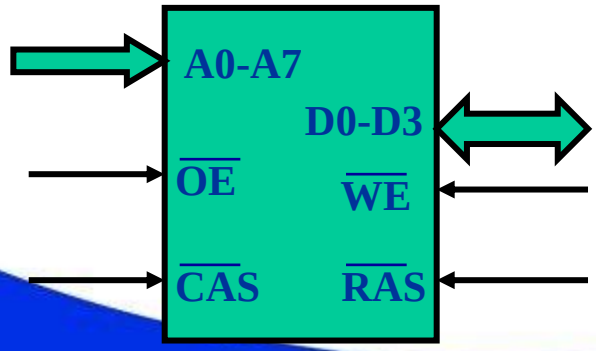
DRAM

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**

Được dùng làm bộ nhớ chính trong các hệ vi xử lý

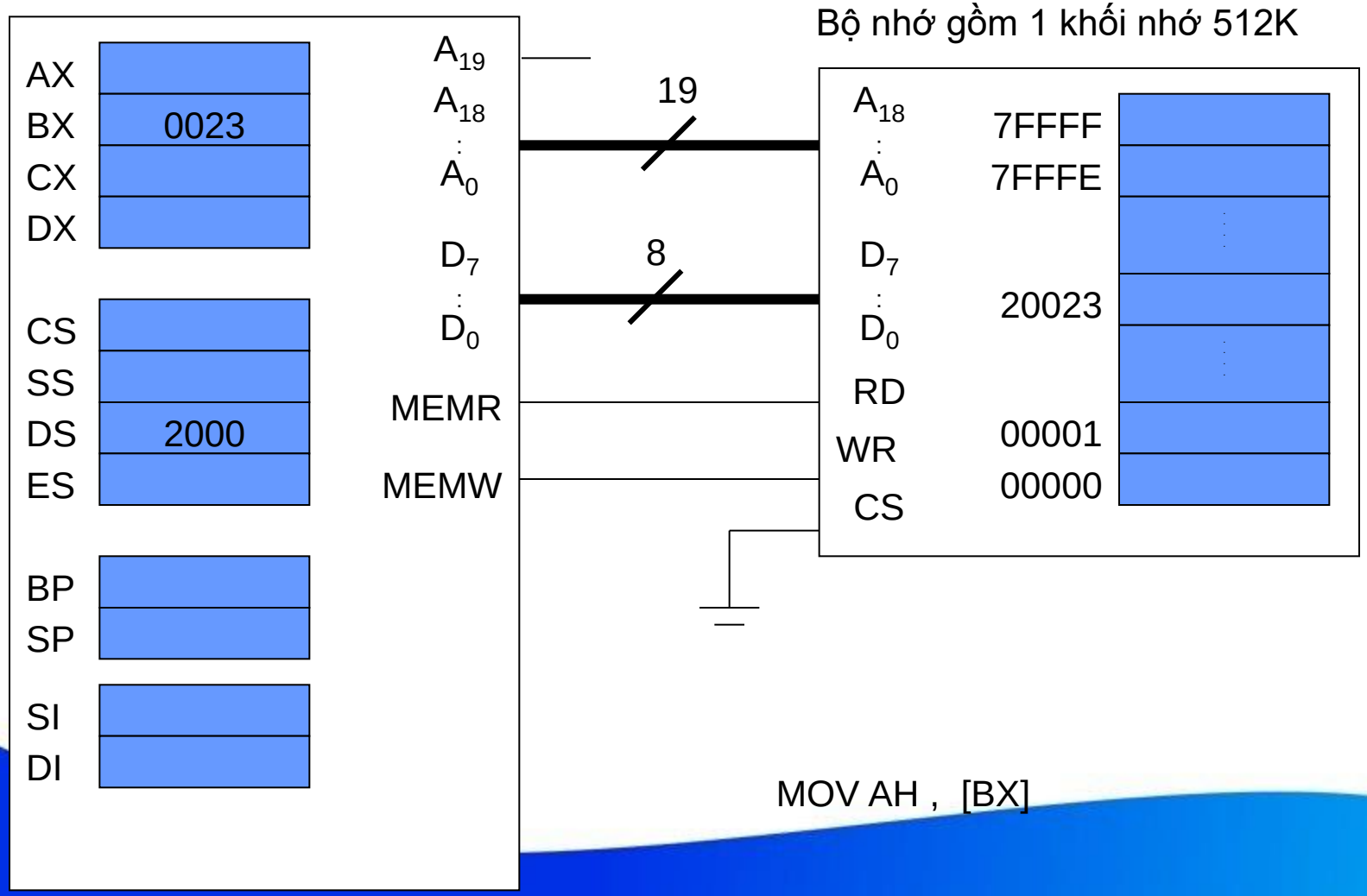
Ví dụ: TMS 4464 (64K*4)



$\overline{CAS}$: cho phép chốt địa chỉ cột
$\overline{RAS}$: cho phép chốt địa chỉ hàng

THIẾT KẾ BỘ NHỚ

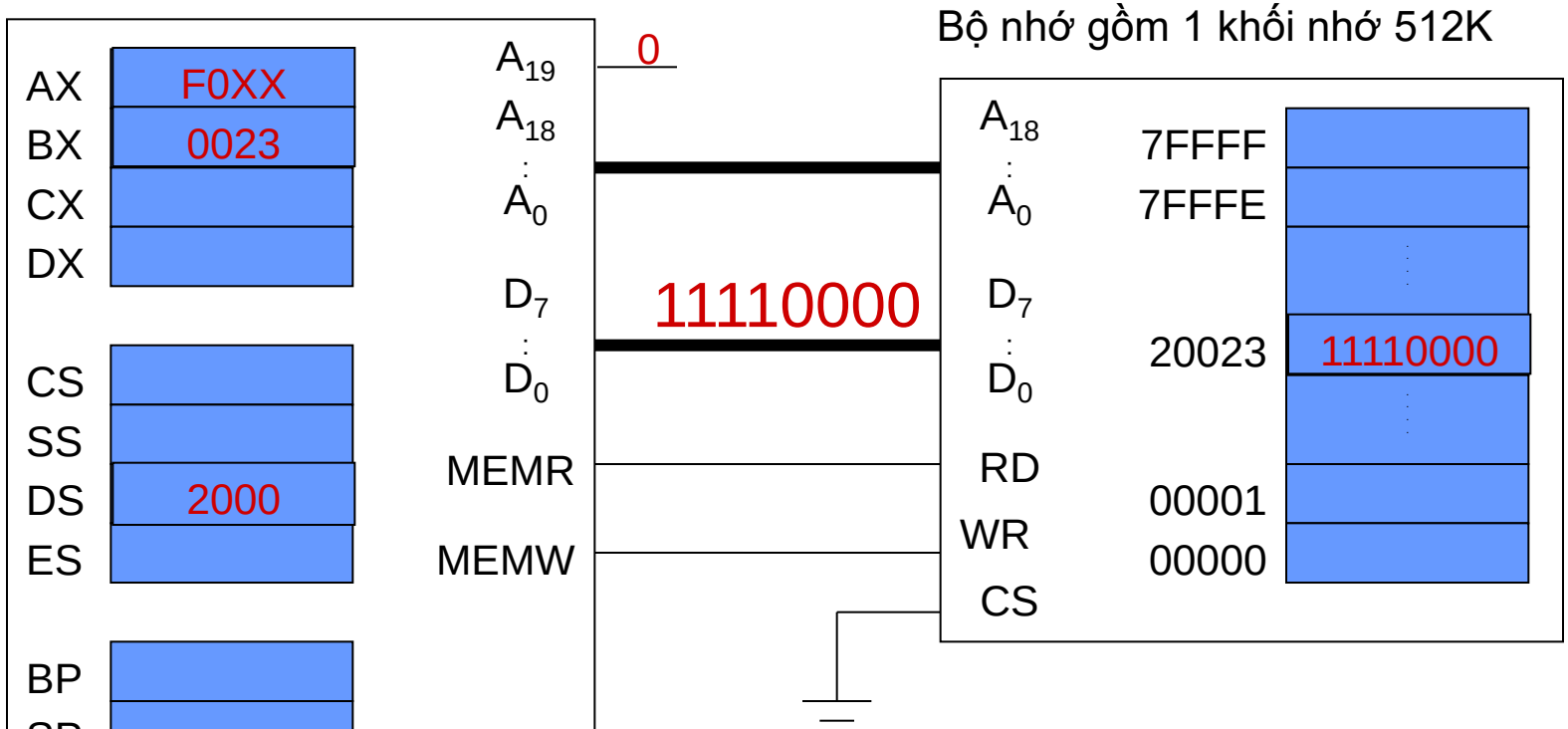
Ví dụ 1 : Xét hệ vi xử lý có bộ nhớ 512KB



MOV AH , [BX]

THIẾT KẾ BỘ NHỚ

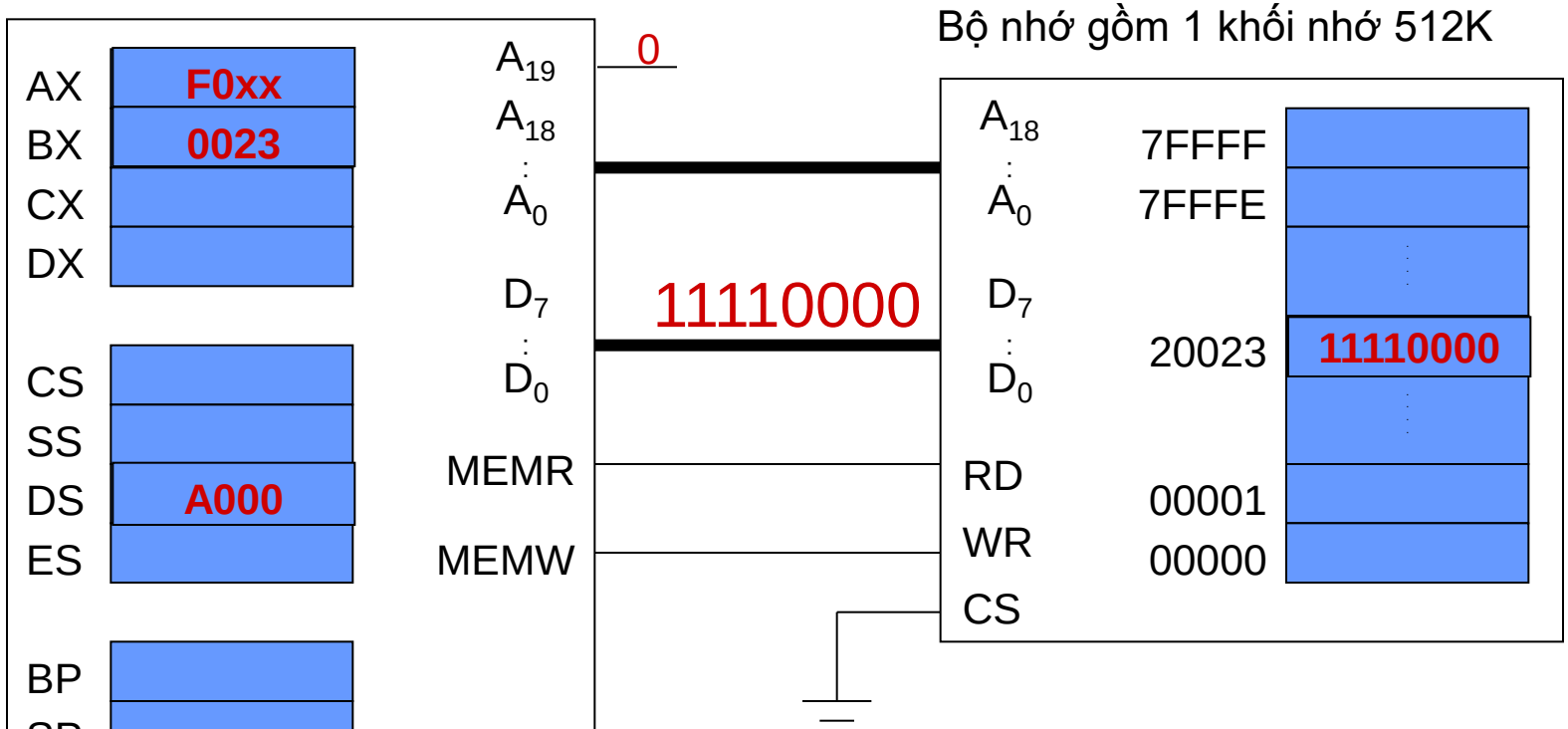
Điều gì xảy ra khi vi xử lý đọc ô nhớ 20023 ?




```
MOV AX, 2000H
MOV DS, AX
MOV BX, 0023H
MOV AH , [BX]
```

THIẾT KẾ BỘ NHỚ

Điều gì xảy ra khi vi xử lý đọc ô nhớ A0023 ?




```
MOV AX, A000H
MOV DS, AX
MOV BX, 0023H
MOV AH , [BX]
```

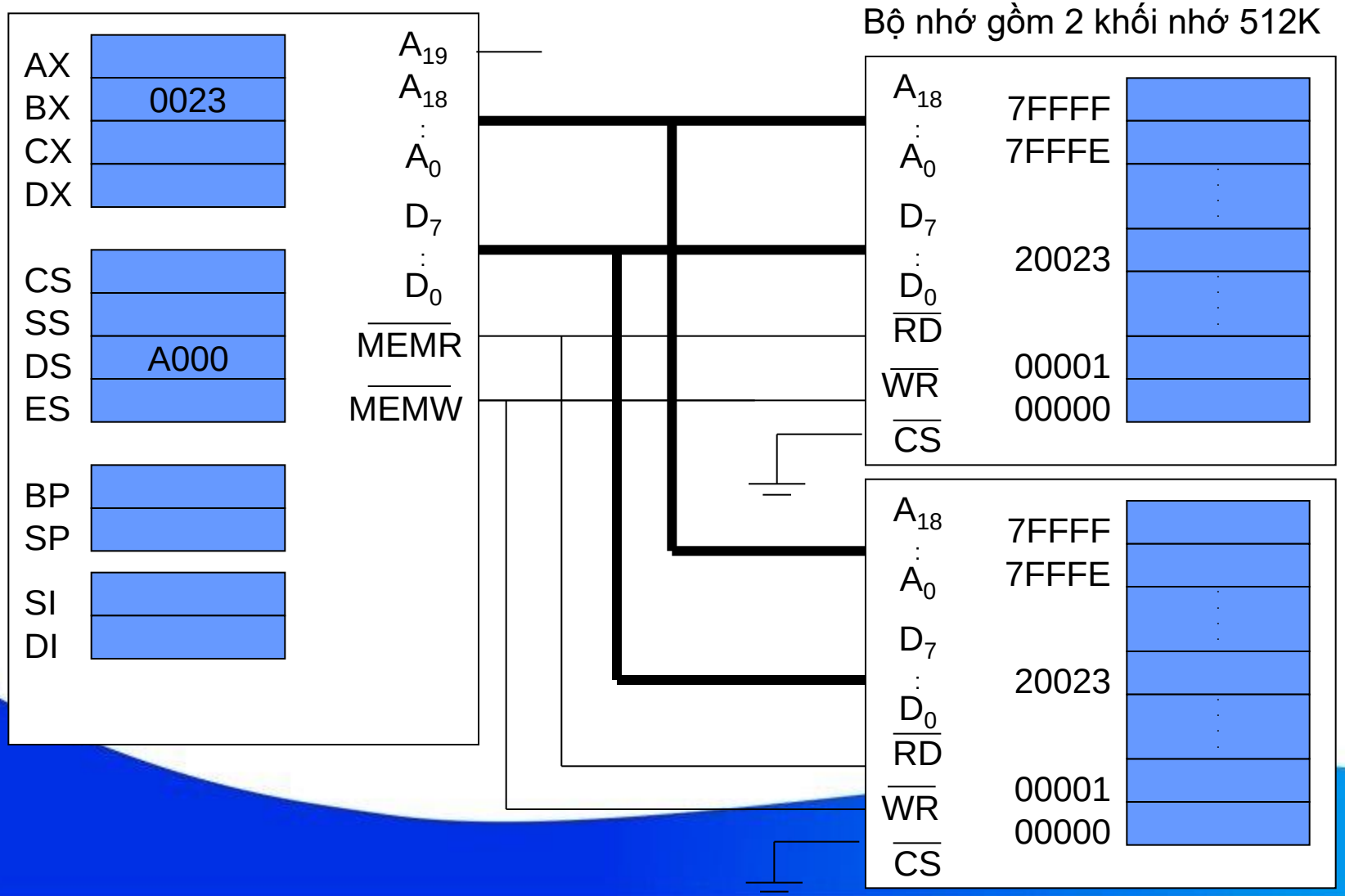
THIẾT KẾ BỘ NHỚ

KẾT LUẬN

- **Nếu chân A19 để trống thì việc đọc ô nhớ 20023h không khác gì đọc ô nhớ A0023h**
- **Một ô nhớ chiếm 2 địa chỉ vật lý**

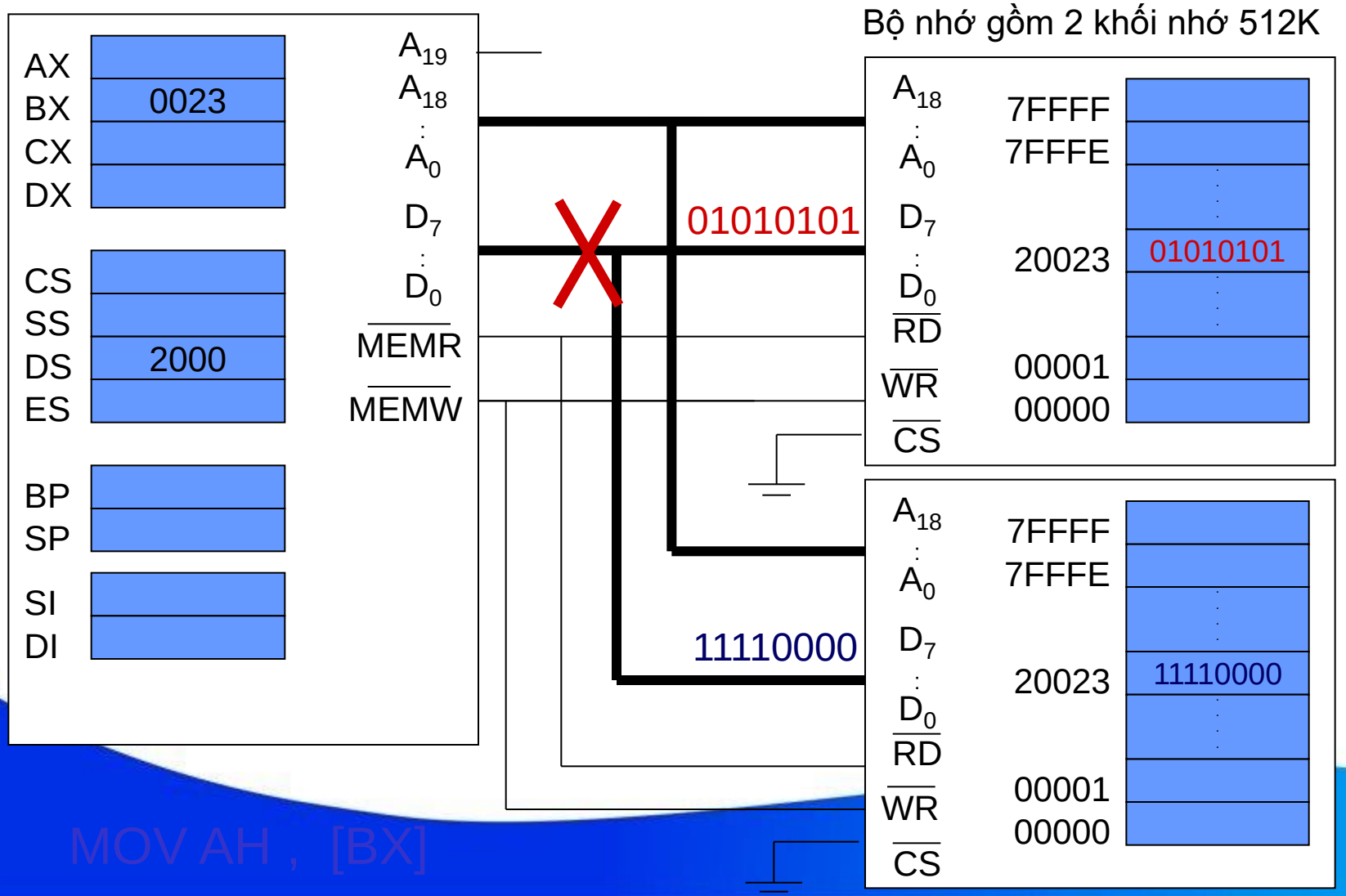
THIẾT KẾ BỘ NHỚ

Ví dụ 2 : Hệ vi xử lý gồm 2 khối nhớ 512KB



THIẾT KẾ BỘ NHỚ

Điều gì xảy ra khi hệ vi xử lý đọc ô nhớ 20023h



MOV AH , [BX]

THIẾT KẾ BỘ NHỚ

KẾT LUẬN :

- **Nếu chân A19 để trống thì 2 chíp nhớ cùng hoạt động.**
- **Nếu 8086 xuất 1 địa chỉ để đọc ô nhớ thì cả 2 chíp nhớ đều xuất số liệu ra cùng 1 lượt**

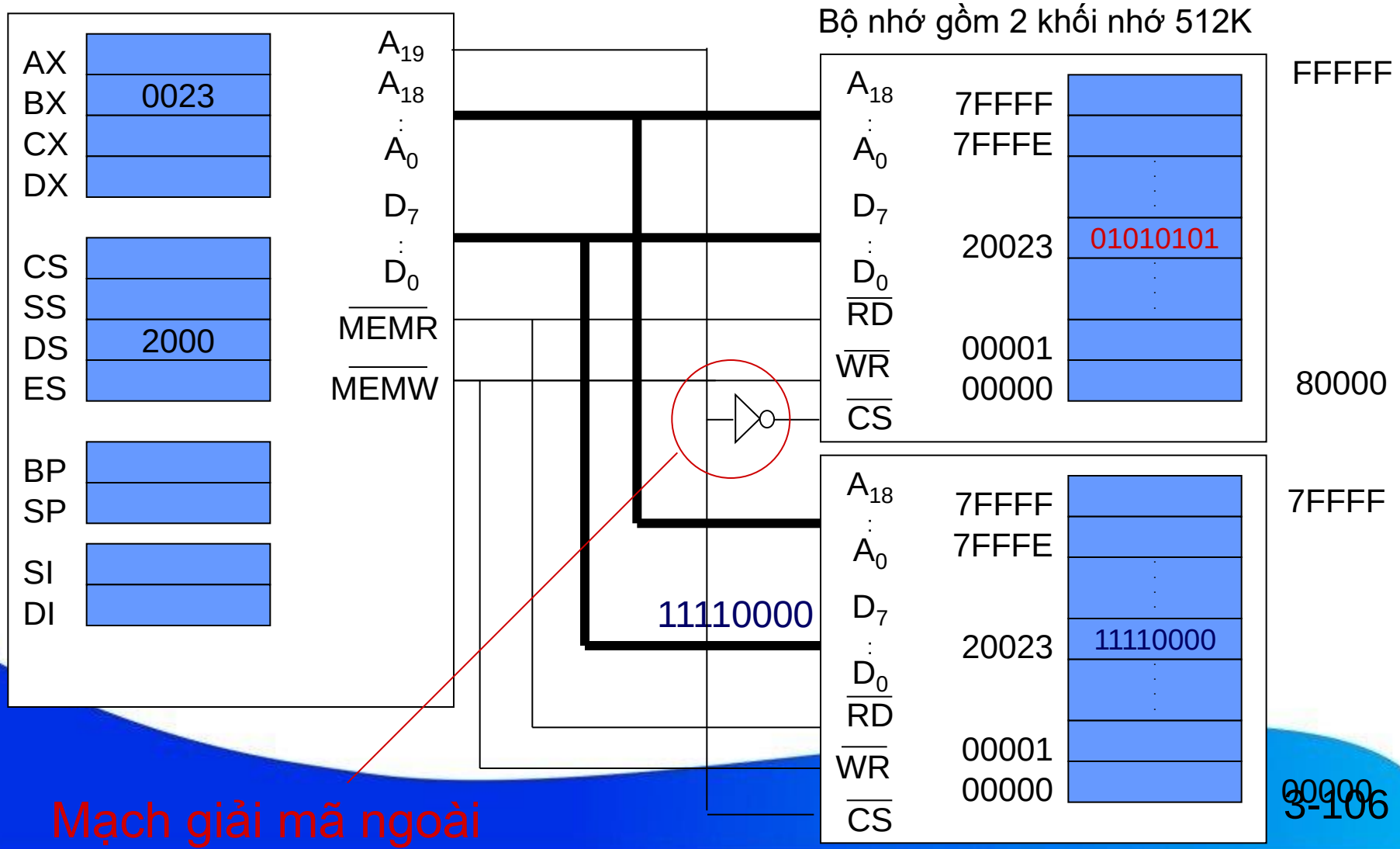
→ xung đột bus

Giải pháp :

Nếu A19 ở mức logic 1 thì chip trên hoạt động
Nếu A19 ở mức logic 0 thì chíp dưới hoạt động
→ mạch giải mã

THIẾT KẾ BỘ NHỚ

Điều gì xảy ra khi hệ vi xử lý đọc ô nhớ 20023h



THIẾT KẾ BỘ NHỚ

Không gian địa chỉ bộ nhớ

	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
00000h	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
7FFFFh	0 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
80000h	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
FFFFFh	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

THIẾT KẾ BỘ NHỚ

Thiết kế bộ nhớ cho hệ vi xử lý là ghép nối một số chíp nhớ có sẵn vào Bus hệ thống của hệ vi xử lý đó sao cho đảm bảo được dung lượng cần thiết và không có đụng độ xảy ra

Thiết kế bộ nhớ thực chất là thiết kế mạch giải mã ngoài

THIẾT KẾ BỘ NHỚ

Giới thiệu IC giải mã 74138

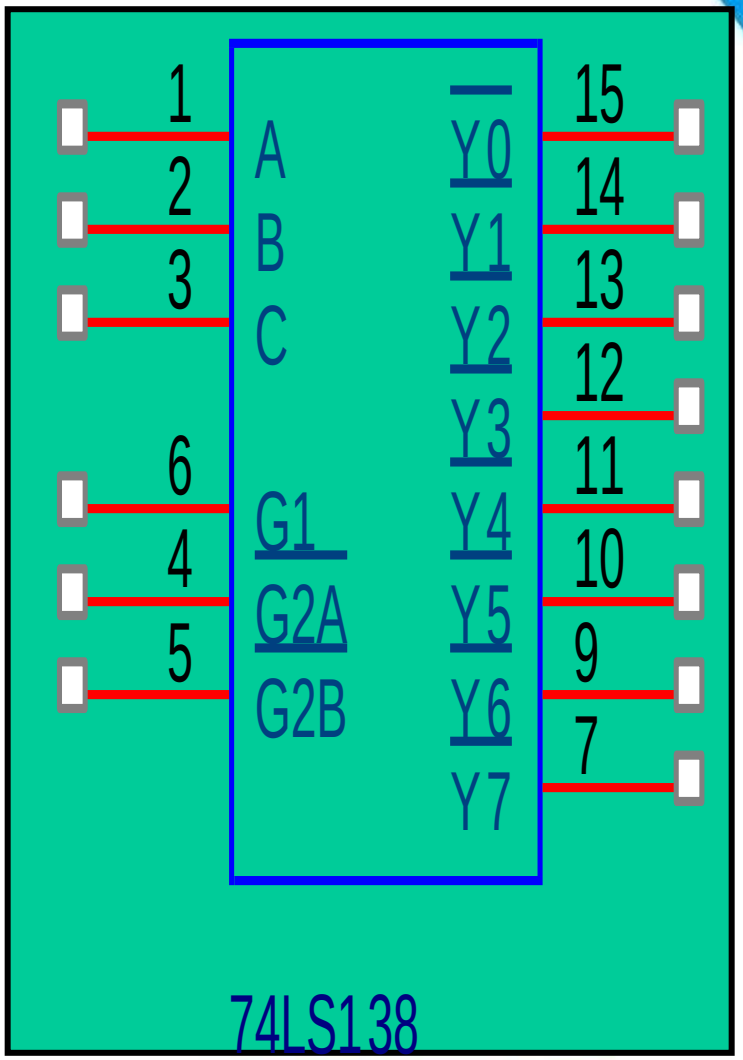
Để IC hoạt động thiết lập như sau :

1→ G1

0→$\overline{\text{G2A}}$

0→ $\overline{\text{G2B}}$

INPUT			OUTPUT							
C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0



THIẾT KẾ BỘ NHỚ

Bài toán : Hãy thiết kế bộ nhớ cho 8088 với các yêu cầu sau :

Bộ nhớ gồm có SRAM và EPROM

SRAM có dung lượng 2kX8 và có địa chỉ từ 00000h

EPROM có dung lượng 2kX8 và có địa chỉ tiếp theo ngay sau phần SRAM

Sử dụng các chip nhớ EPROM 2716 , SRAM 6216 ; các cổng logic và chip giải mã 74138

THIẾT KẾ BỘ NHỚ

Bước 1 : Vẽ bản đồ bộ nhớ

2KB= 2048 byte = 2^{11} byte

→cần 11 bit địa chỉ

00000000000b → 11111111111b

(000h → 7FFh)

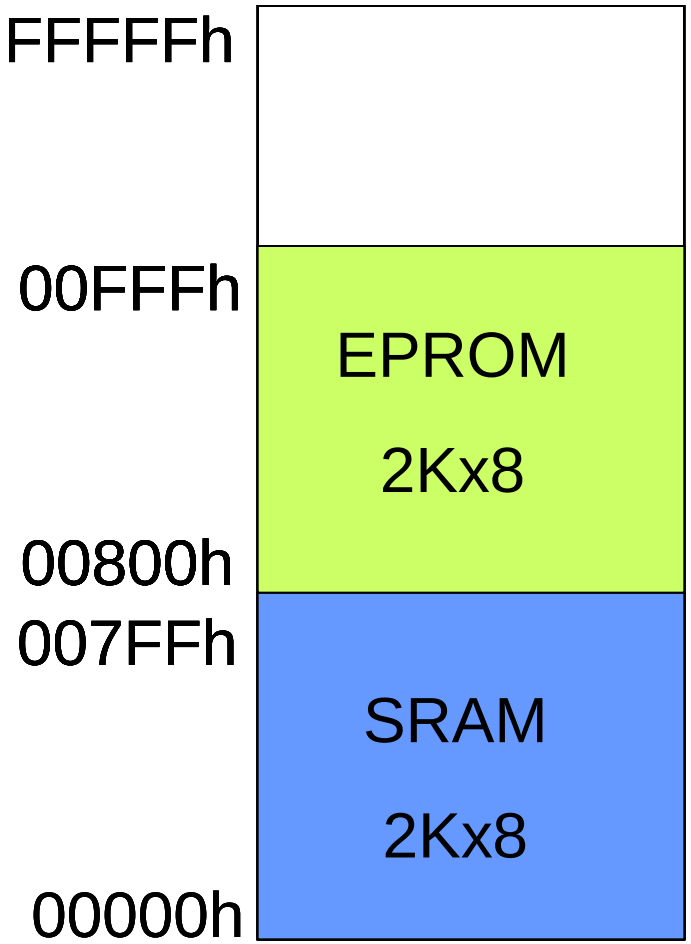
FFFFF

- 7FF

FF800

-1

FF7FF



THIẾT KẾ BỘ NHỚ

Bước 2 : Triển khai các địa chỉ biên từ HEX sang BIN

	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
00000h	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
…					
007FFh	0 0 0 0	0 0 0 0	0 1 1 1	1 1 1 1	1 1 1 1
00800h	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0 0 0 0
…					
00FFFh	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1	1 1 1 1
…					
FFFFFh	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

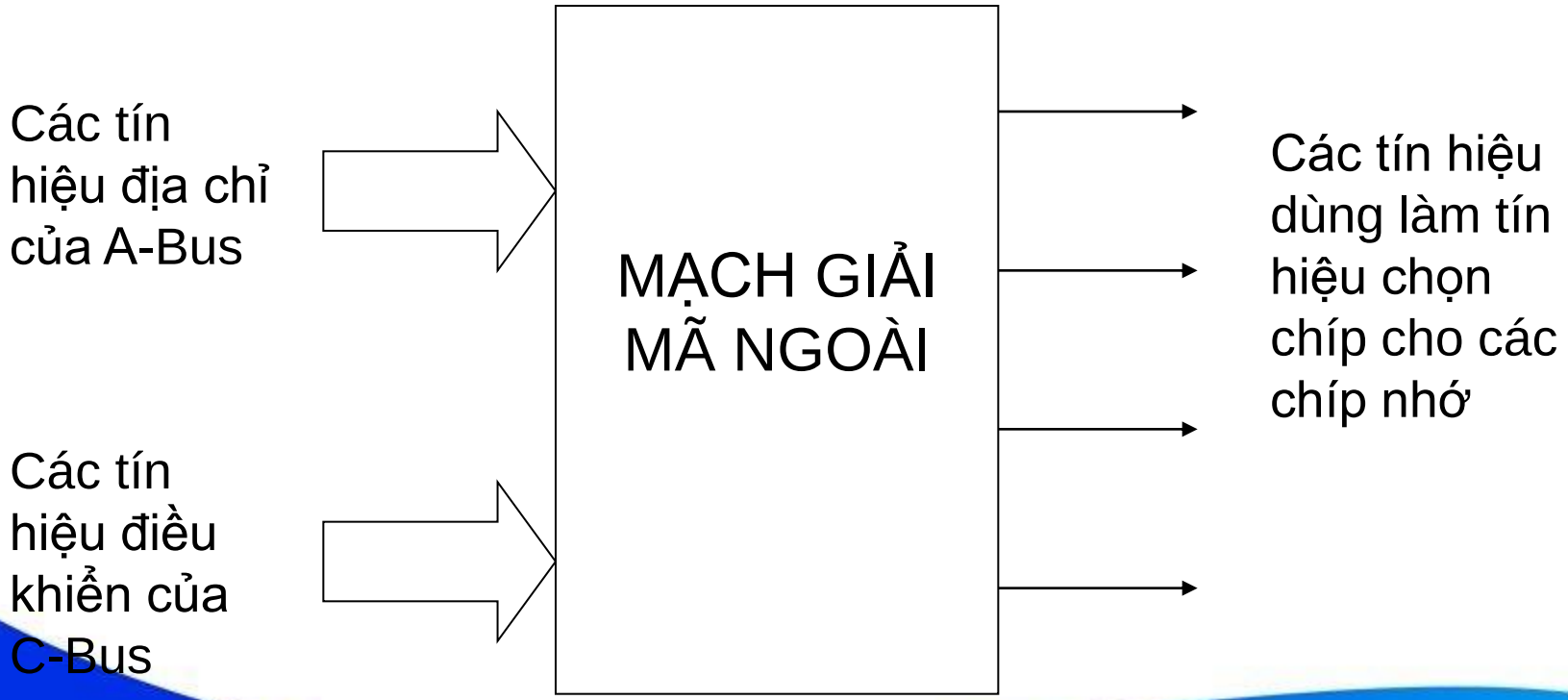
Nhận xét : SRAM : $A_{11} \rightarrow A_{19} = 0$

EPROM : $A_{12} \rightarrow A_{19} = 0$; $A_{11} = 1$

THIẾT KẾ BỘ NHỚ

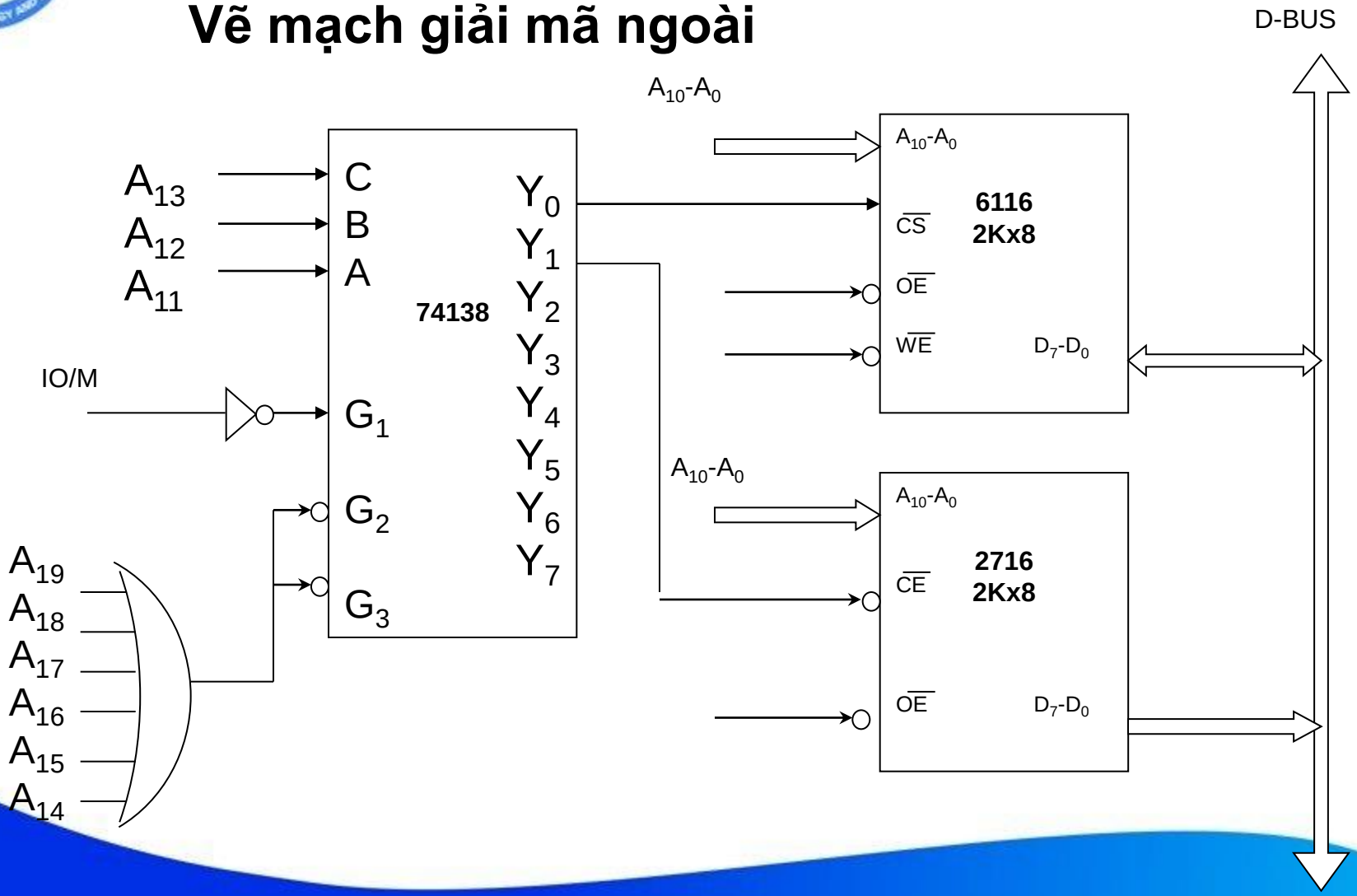
Bước 3 : Vẽ mạch giải mã ngoài

Yêu cầu của mạch giải mã ngoài:



THIẾT KẾ BỘ NHỚ

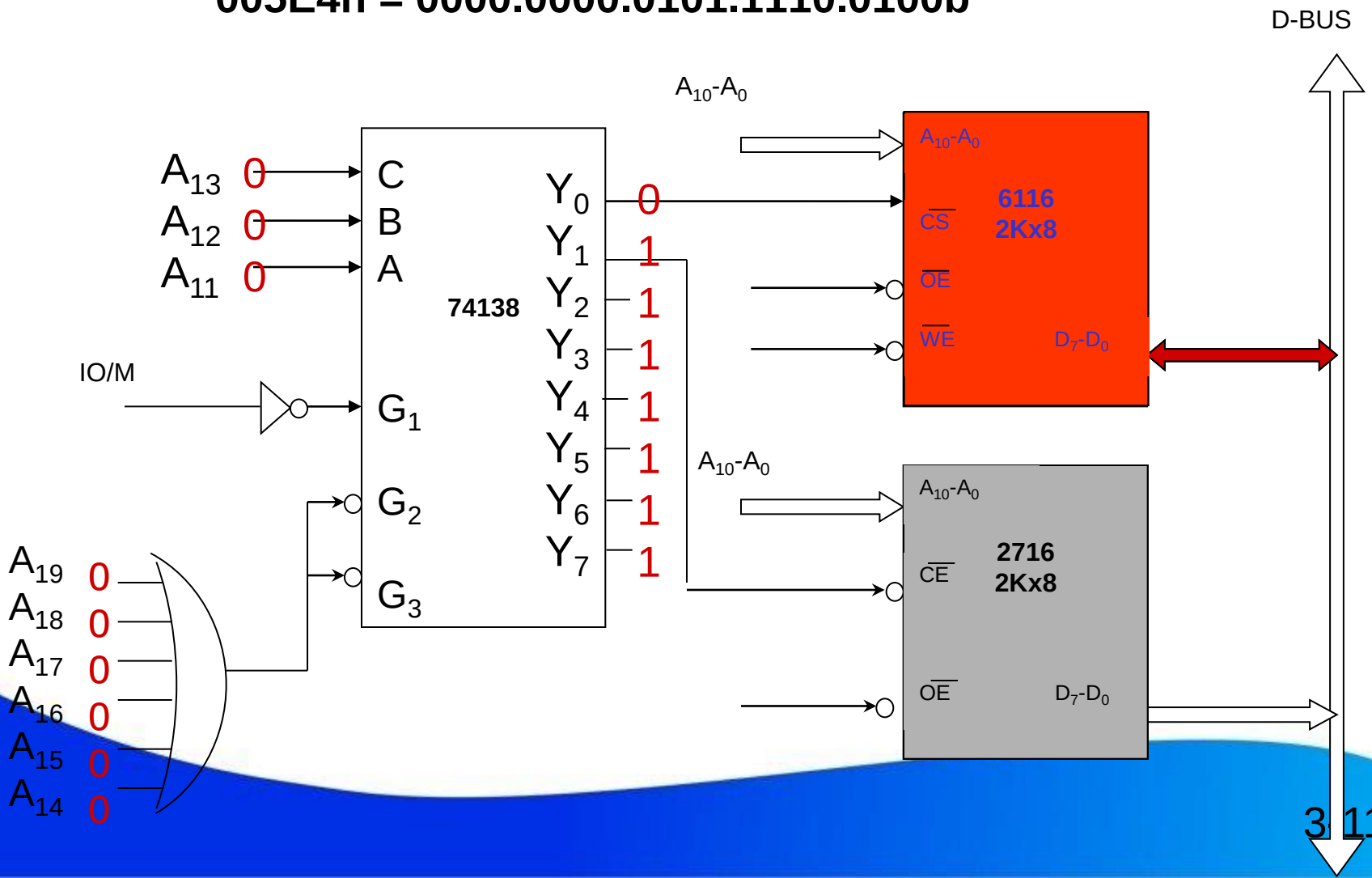
Vẽ mạch giải mã ngoài



THIẾT KẾ BỘ NHỚ

Bước 4 : Thử lại Đọc bộ nhớ SRAM tại địa chỉ 005E4h

005E4h = 0000.0000.0101.1110.0100b



VÍ DỤ 2

Hãy thiết kế bộ nhớ cho 8088 với các yêu cầu sau :

Bộ nhớ 16KB gồm có SRAM và ROM

SRAM có dung lượng 8kX8 và có địa chỉ từ 00000h

ROM có dung lượng 8kX8 và có địa chỉ tiếp theo ngay sau phần SRAM

Sử dụng các chip nhớ EPROM 2764 , SRAM 6264 ; các cổng logic và chip giải mã 74138

GIẢI

Bước 1 : Vẽ bản đồ bộ nhớ

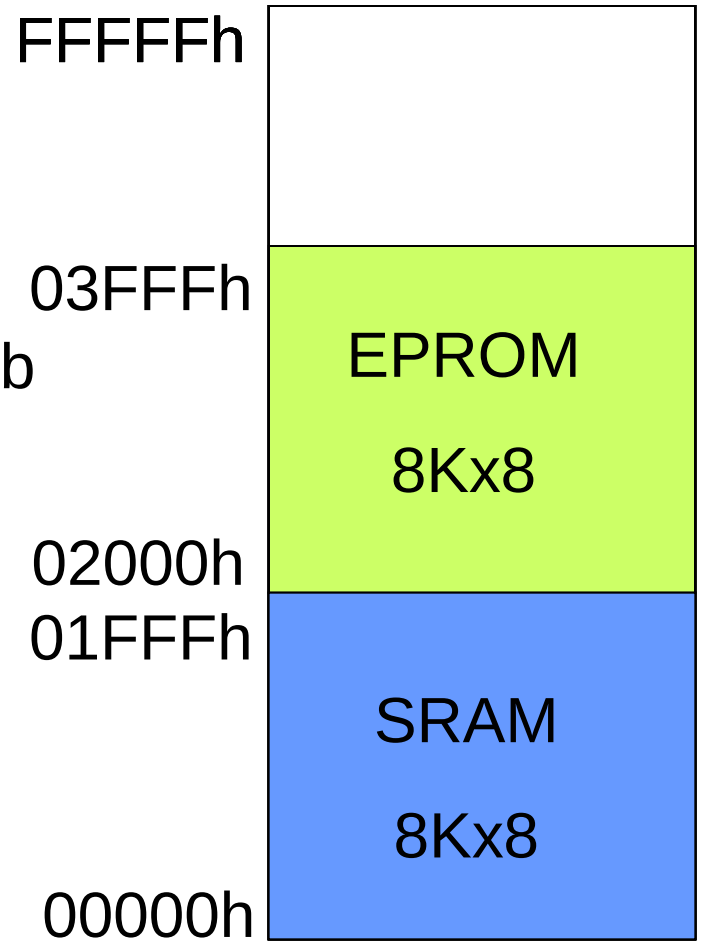
8KB = 2^{13} byte

→cần 13 bit địa chỉ

0000000000000b → 1111111111111b

Chiếm dụng không gian bộ nhớ:

2^{13} = 1FFF h



GIẢI

Bước 2 : Triển khai các địa chỉ biên từ HEX sang BIN

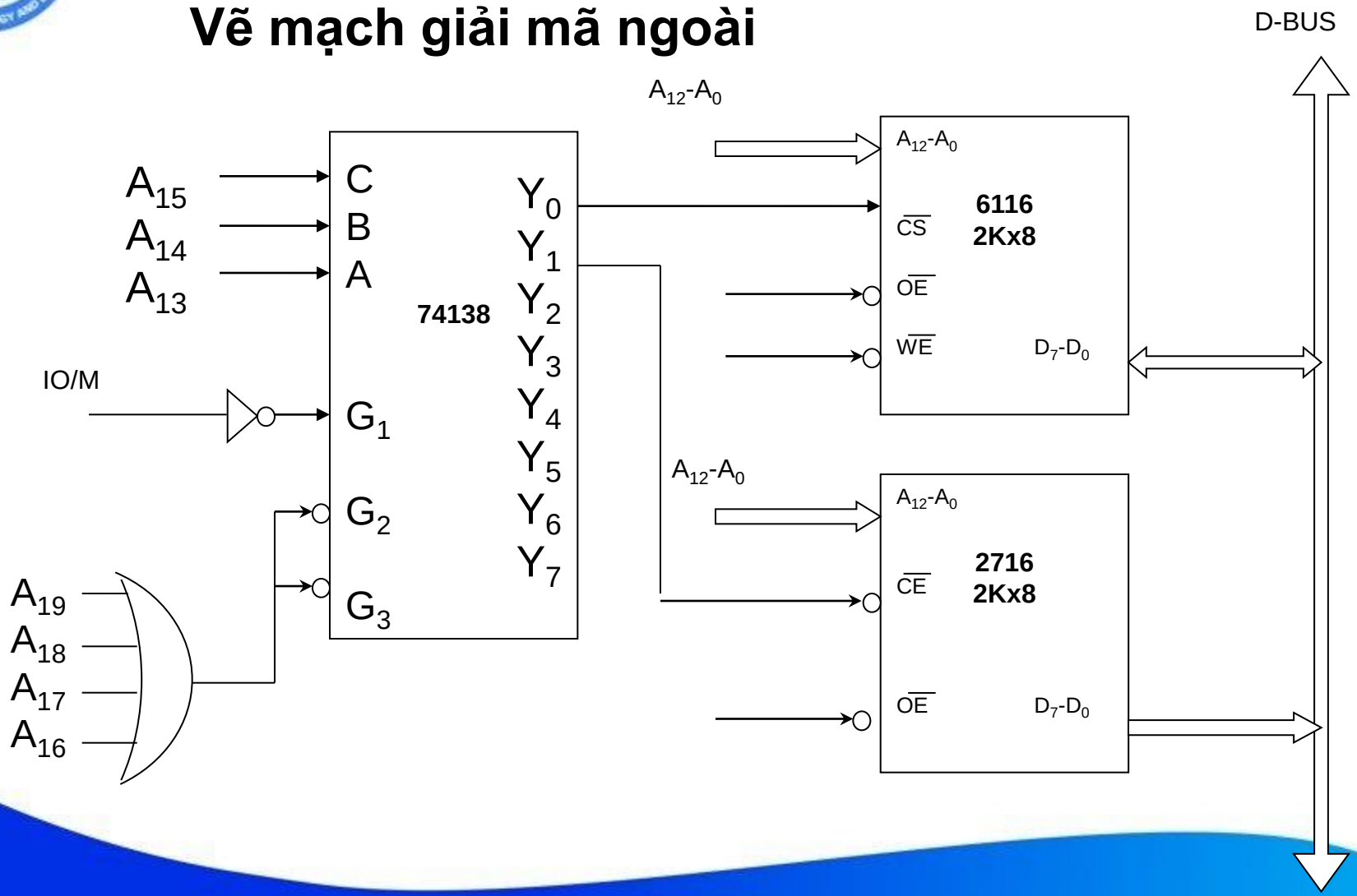
	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_{9}A_{8}$	$A_{7}A_{6}A_{5}A_{4}$	$A_{3}A_{2}A_{1}A_{0}$
00000h	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
…					
01FFFh	0 0 0 0	0 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1
02000h	0 0 0 0	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0
…					
03FFFh	0 0 0 0	0 0 1 1	1 1 1 1	1 1 1 1	1 1 1 1
…					
FFFFFh	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

Nhận xét : SRAM : $A_{13} \rightarrow A_{19} = 0$

EPROM : $A_{14} \rightarrow A_{19} = 0$; $A_{13} = 1$

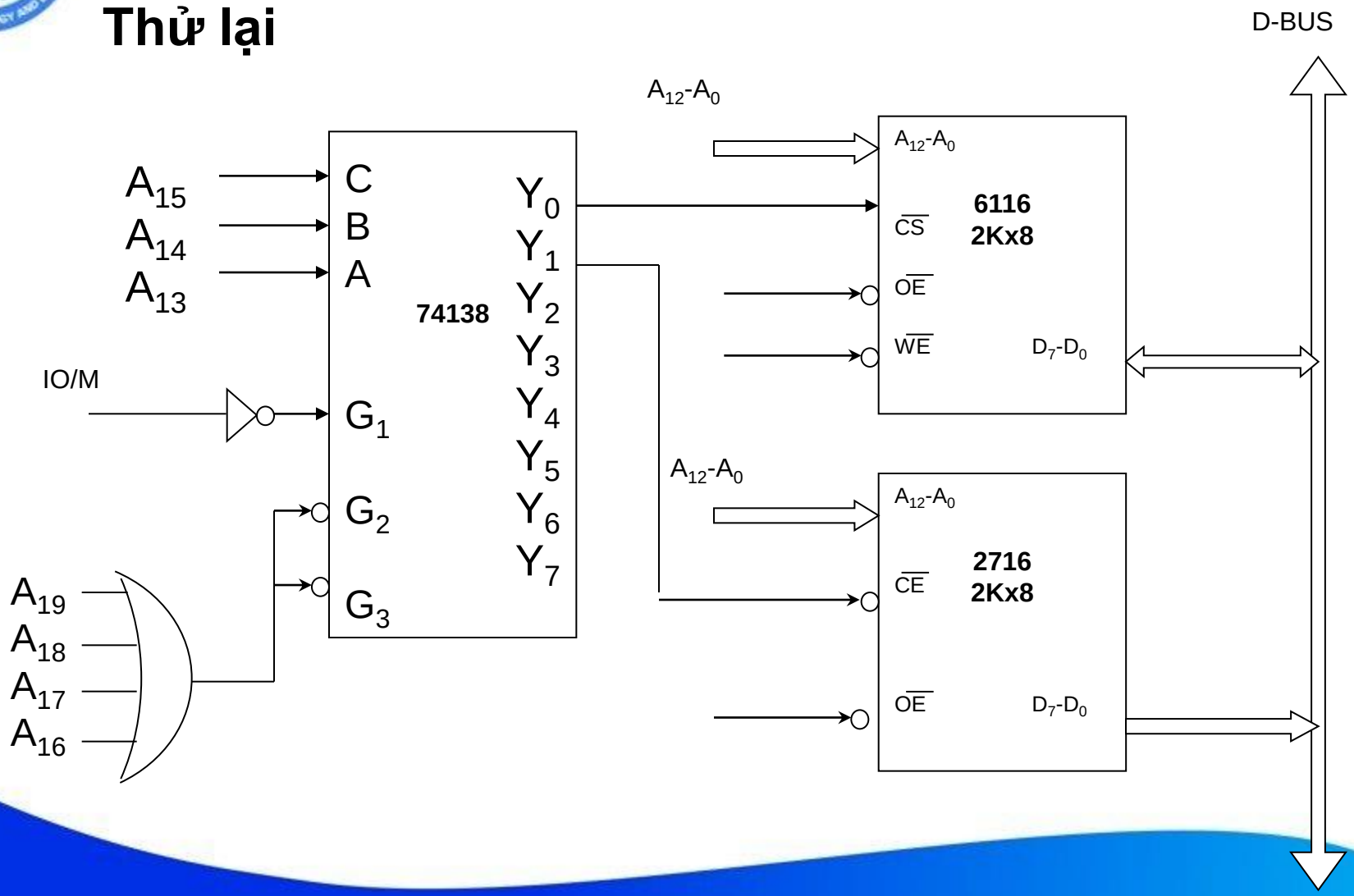
GIẢI

Vẽ mạch giải mã ngoài



THIẾT KẾ BỘ NHỚ

Thử lại



VÍ DỤ 3

Hãy thiết kế bộ nhớ cho 8088 với các yêu cầu sau :

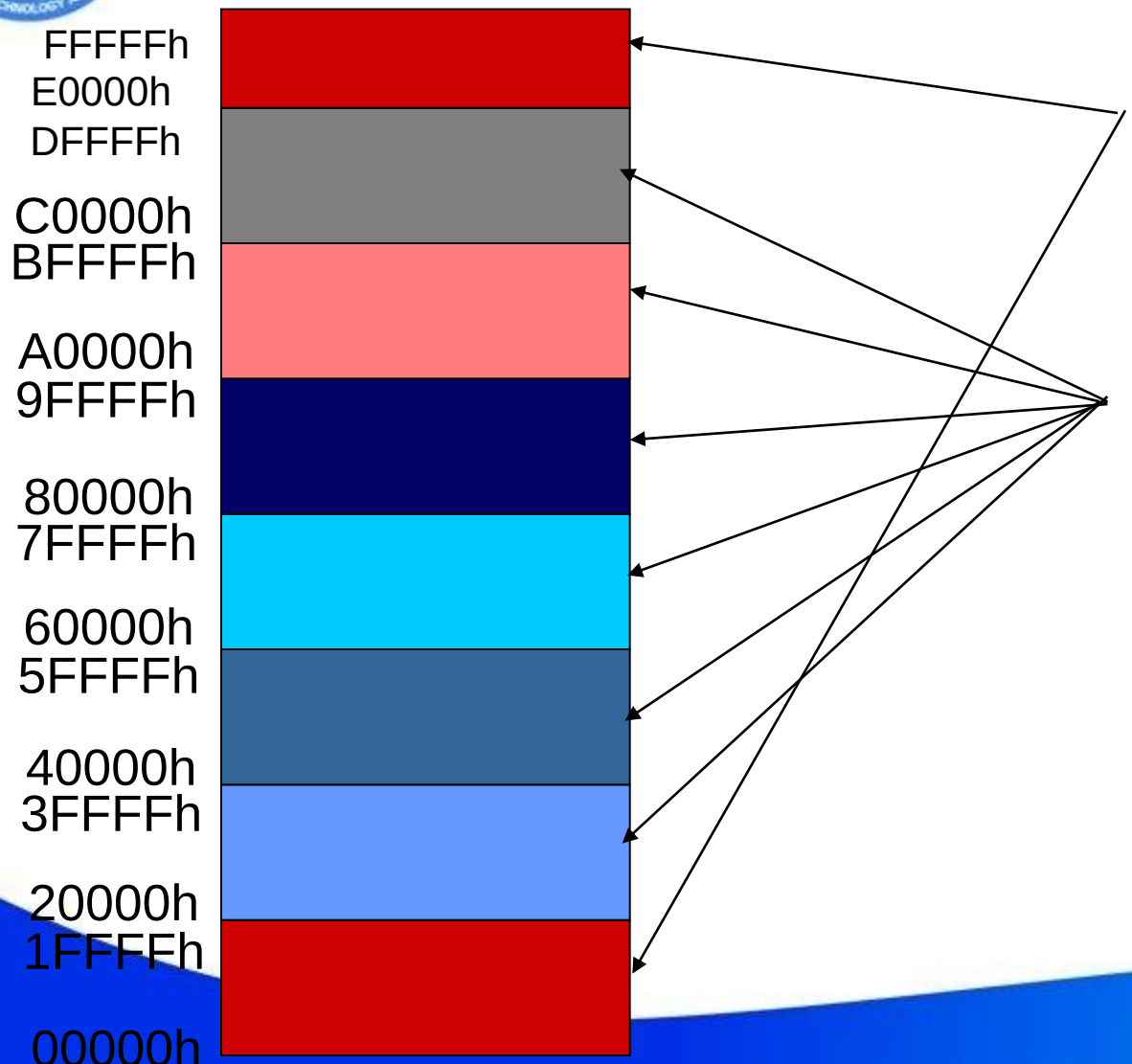
Bộ nhớ 1MB gồm có SRAM và ROM

ROM gồm có 2 phần, mỗi phần có dung lượng 128kX8 và nằm ở phần đầu tiên và phần cuối cùng của bộ nhớ

SRAM có dung lượng 768kX8 và có địa chỉ nằm ở giữa 2 phần ROM

Sử dụng các chip nhớ EPROM 271024 (128Kx8) , SRAM 621024 ; các cổng logic và chip giải mã 74138

BẢN ĐỒ BỘ NHỚ

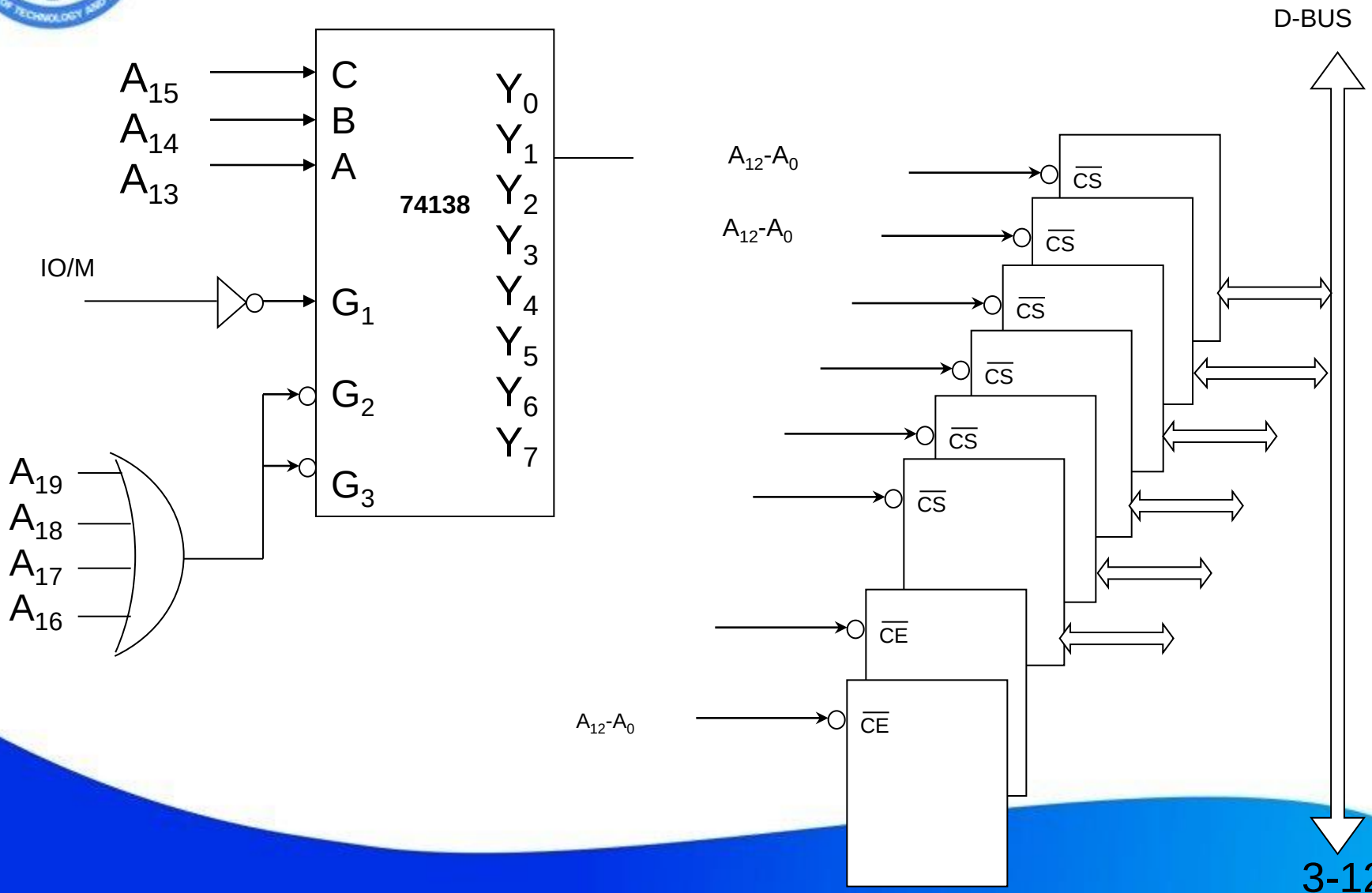


2 khối ROM mỗi khối 128K

6 khối RAM mỗi khối 128K

128Kx6 =768KB

	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
00000h	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
1FFFFh	0 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
20000h	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
3FFFFh	0 0 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
40000h	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
5FFFFh	0 1 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
60000h	0 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
7FFFFh	0 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
80000h	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
9FFFFh	1 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
A0000h	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
BFFFFh	1 0 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
C0000h	1 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
DFFFFh	1 1 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1
E0000h	1 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
FFFFFh	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1


D-BUS
A15
A14
A13
C
B
A
74138
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
IO/M
G1
G2
G3
A19
A18
A17
A16
A12-A0
A12-A0
A12-A0
CS
CS
CS
CS
CS
CS
CE
CE

THIẾT KẾ BỘ NHỚ CHO 8051

Hãy thiết kế bộ nhớ cho 8051 với các yêu cầu sau :

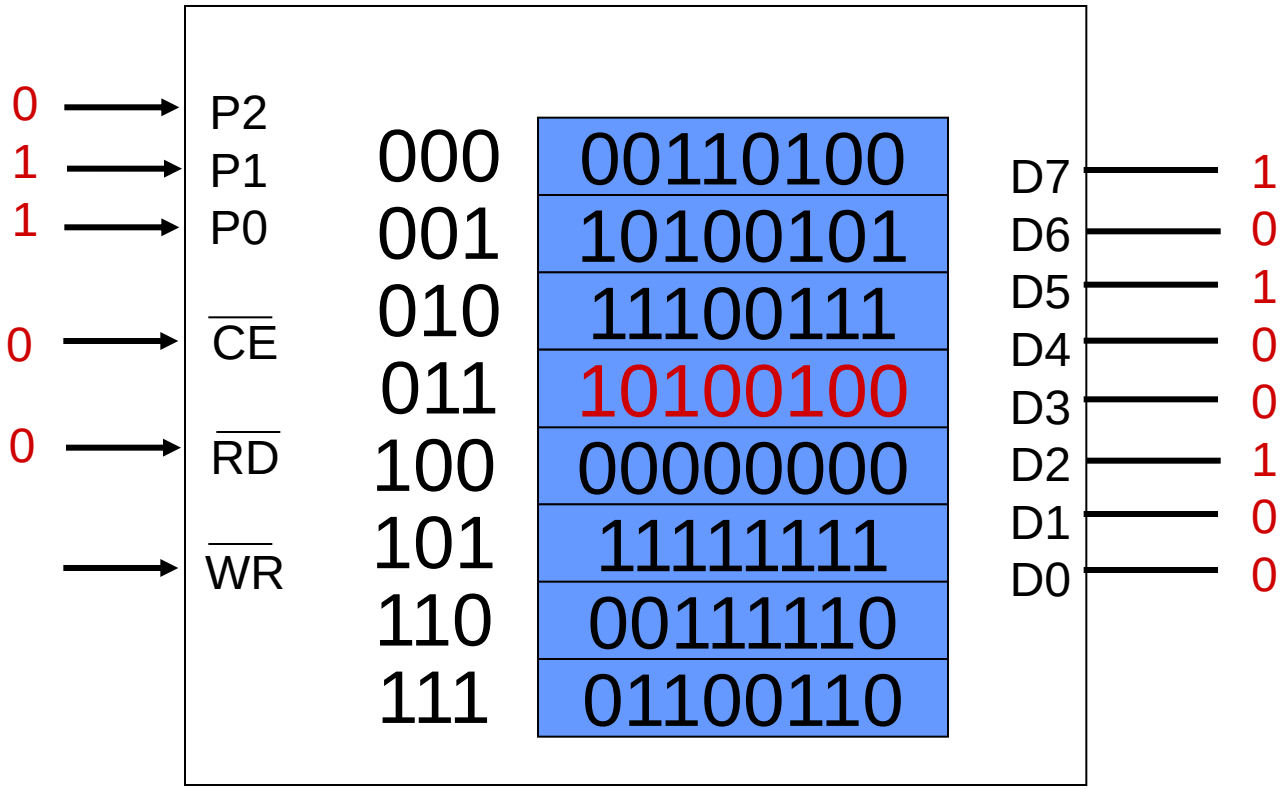
EPROM có dung lượng 64KB và nằm ở phần đầu tiên và phần cuối cùng của bộ nhớ

SRAM có dung lượng 768kX8 và có địa chỉ nằm ở giữa 2 phần ROM

Sử dụng các chip nhớ EPROM 271024 (128Kx8) , SRAM 621024 ; các cổng logic và chip giải mã 74138

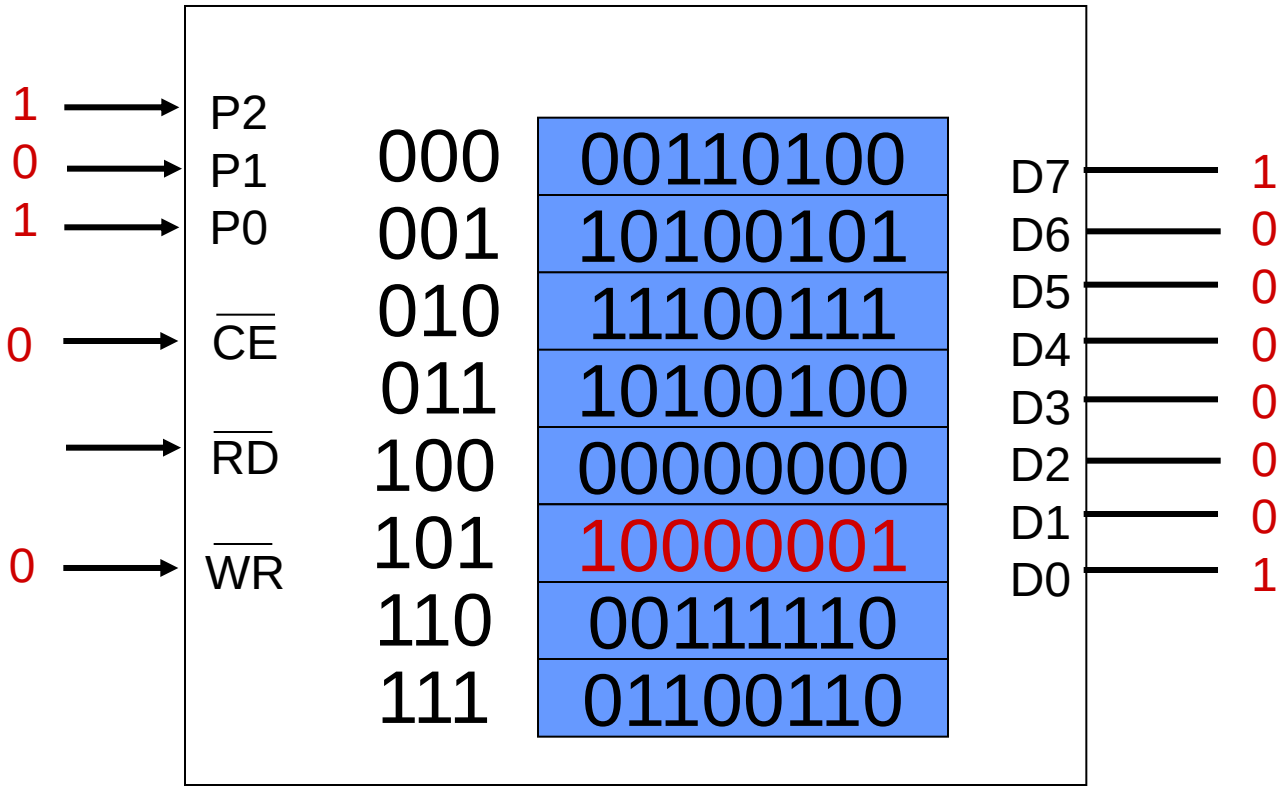
HOẠT ĐỘNG ĐỌC

Ví dụ bộ nhớ 8 byte. Đọc byte tại địa chỉ 011



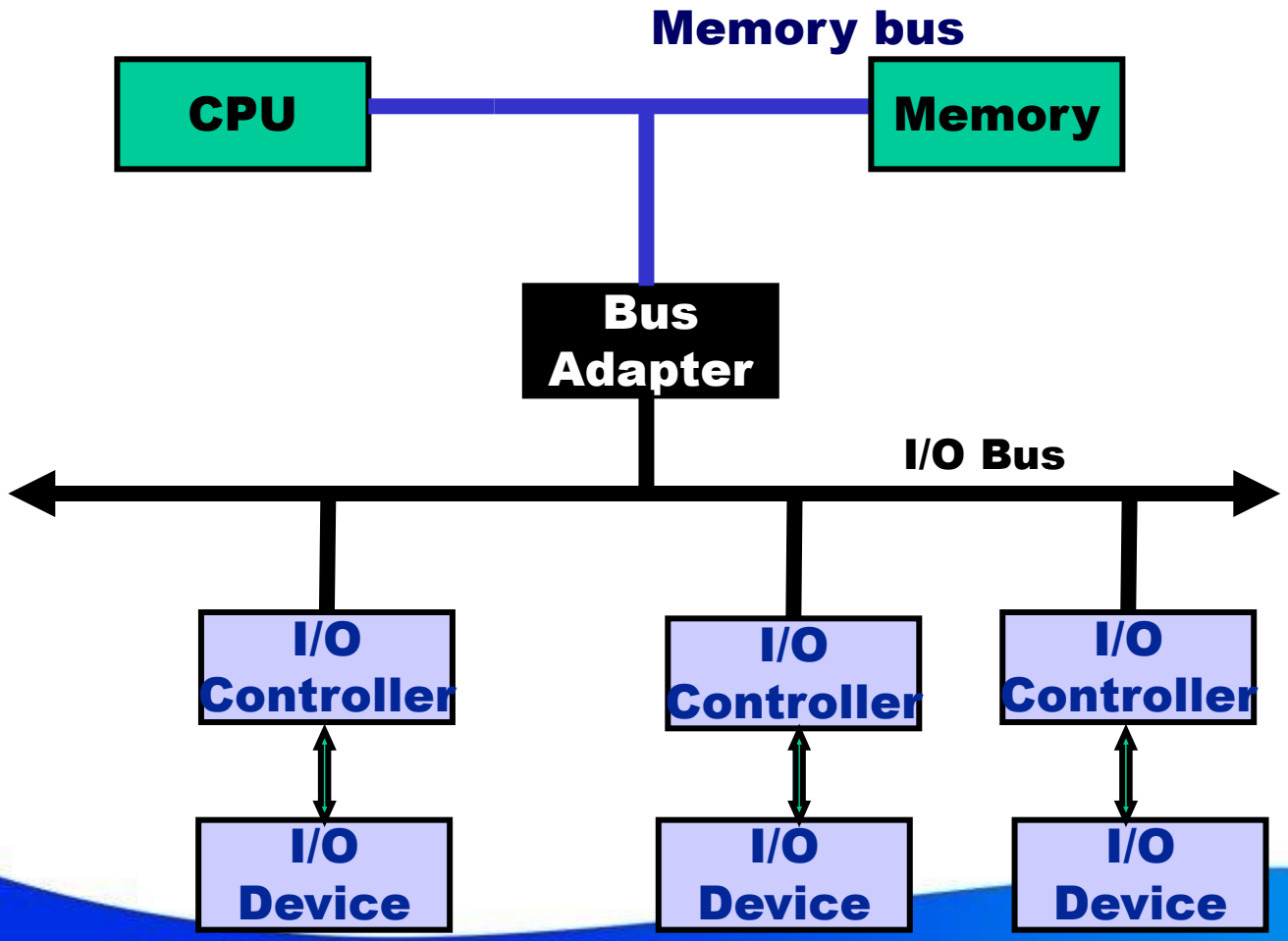
HOẠT ĐỘNG GHI

Ví dụ: Ghi vào bộ nhớ tại địa chỉ 101 giá trị 10000001



3.3 Thiết kế hệ thống các cổng I/O

CÁC KIỂU GIAO TIẾP VỚI NGOẠI VI



CÁC KIỂU GIAO TIẾP VỚI 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

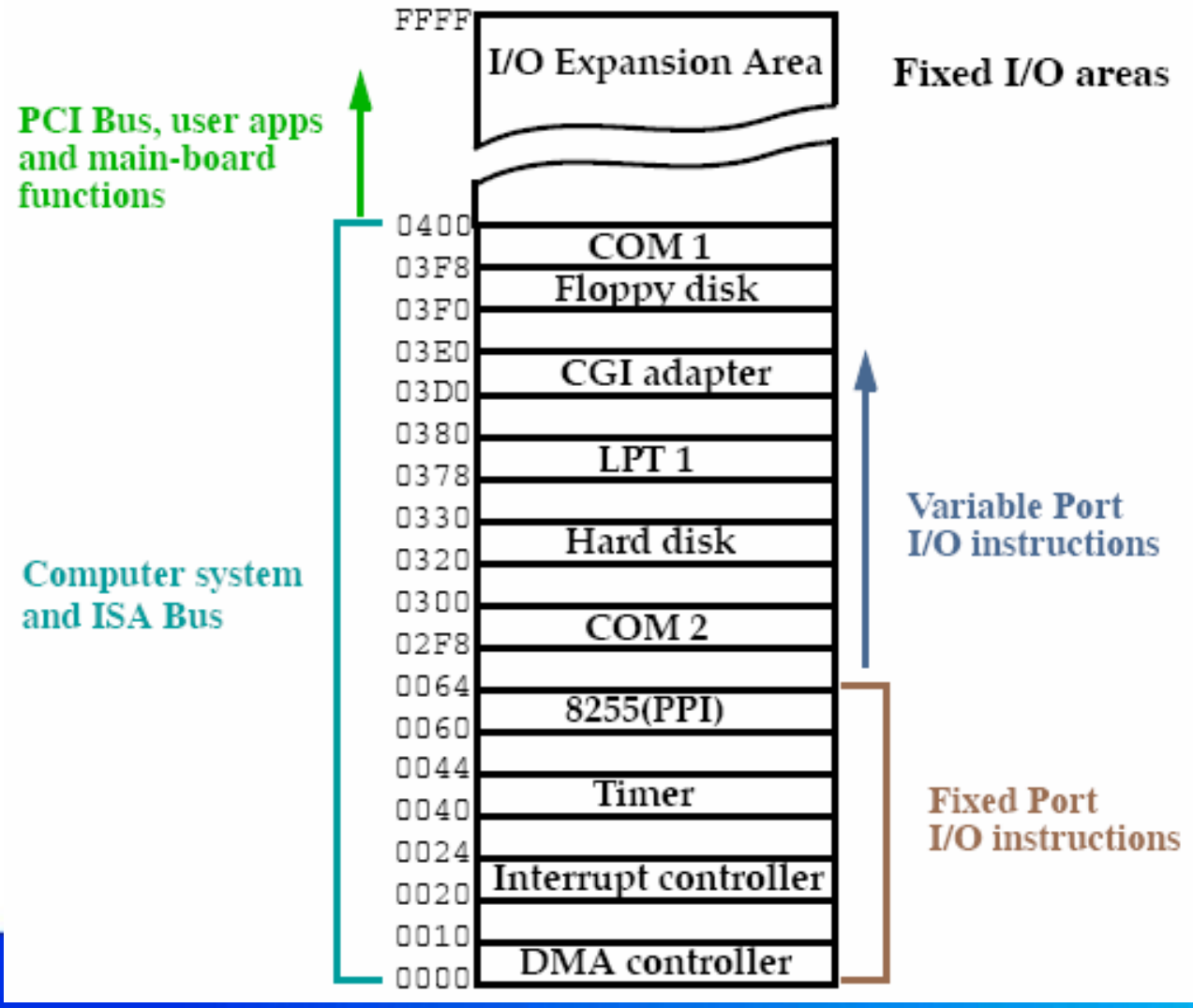
CÁC KIỂU GHÉP NỐI VÀO/RA

Thiết bị vào/ra có không gian địa chỉ cách biệt

Địa chỉ: 0000H-FFFFH
M/IO=0

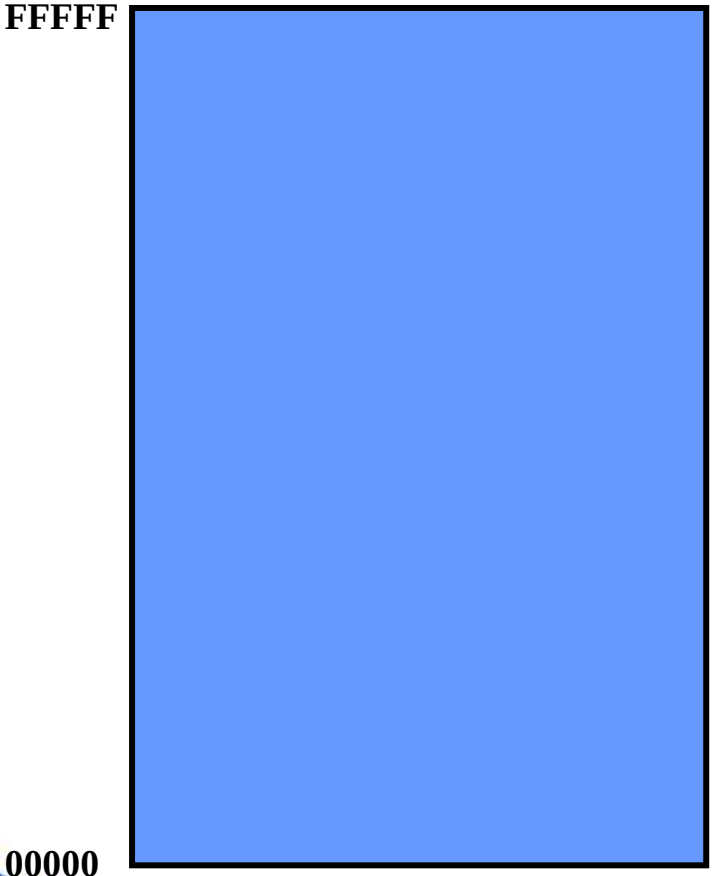
Vào/ra dữ liệu bằng lệnh IN, OUT

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 cùng không gian địa chỉ với bộ nhớ



Memory + I/O

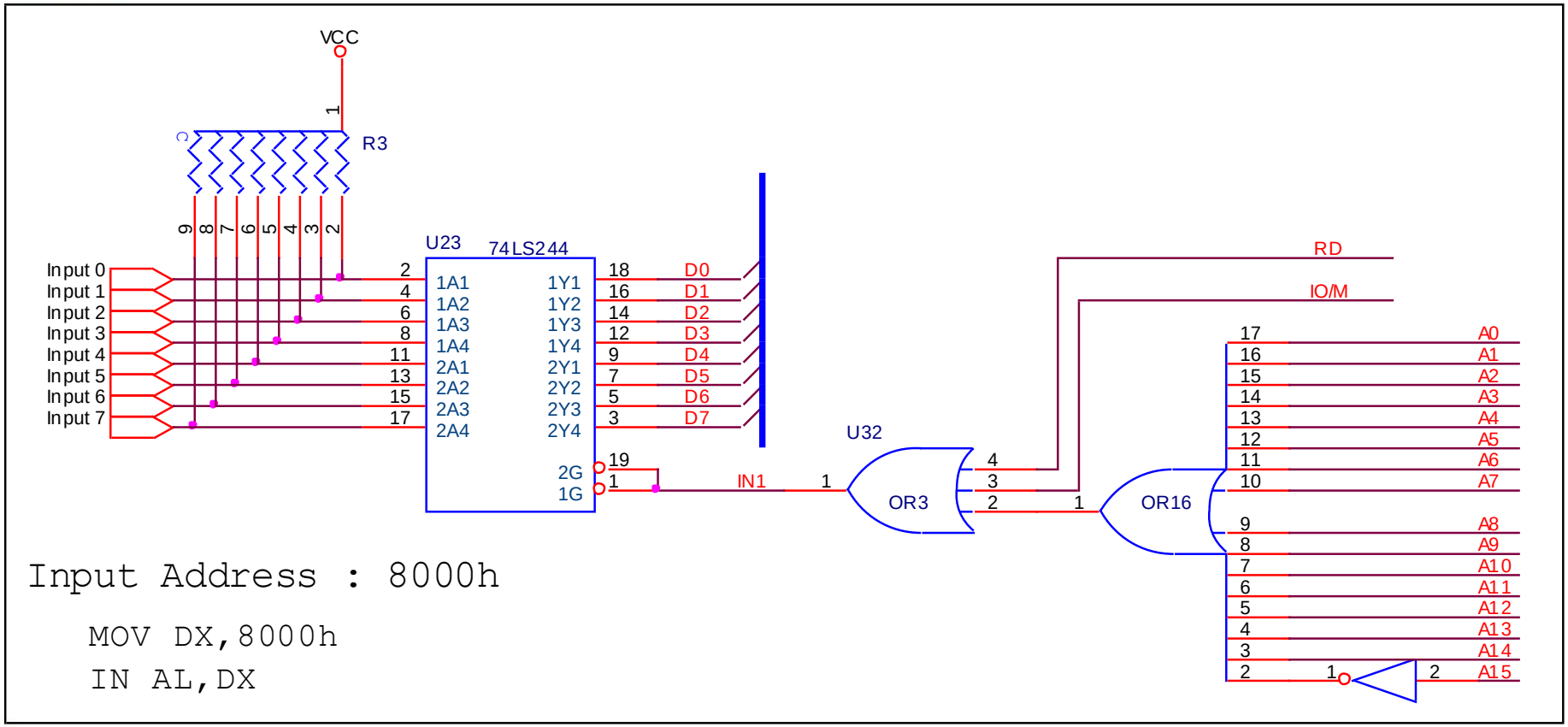
$\overline{IO}$/M=1

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

Ví dụ:

MOV AX, [0FF3H]

THIẾT KẾ CỔNG NHẬP 8 BIT CHO 8088




```
Input Address : 8000h

    MOV DX,8000h
    IN AL,DX
```

Sử dụng bộ đệm 74LS244 và điện trở kéo lên ở đầu vào

Mạch giải mã địa chỉ 8000h ghép nối với 8088

Không gian địa chỉ tách biệt với bộ nhớ

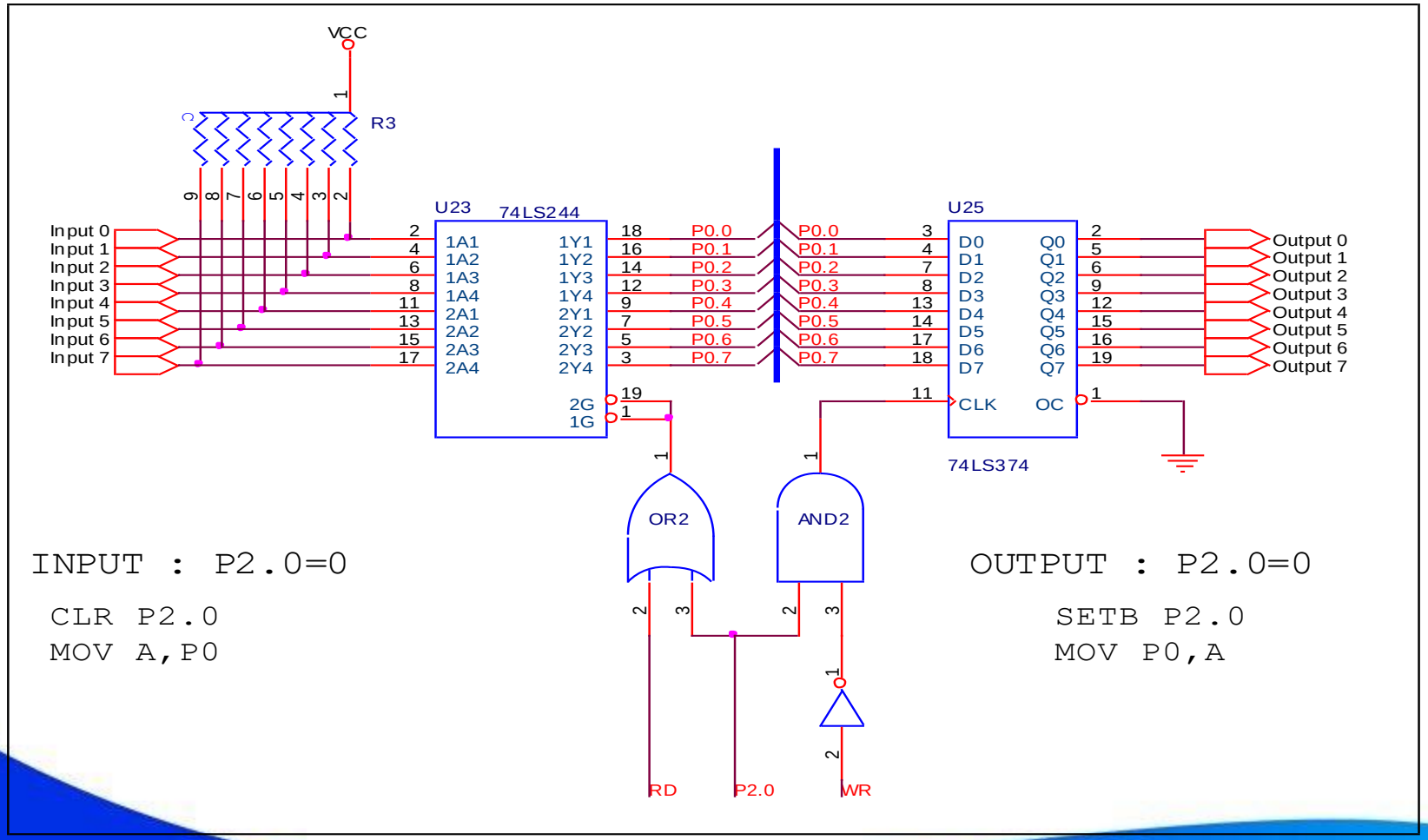
THIẾT KẾ CỔNG XUẤT 8 BIT CHO 8088

Sử dụng bộ chốt 74LS374

Mạch giải mã địa chỉ 0080h ghép nối với 8088

Không gian địa chỉ tách biệt với bộ nhớ

MỞ RỘNG CỔNG I/O CHO 8051



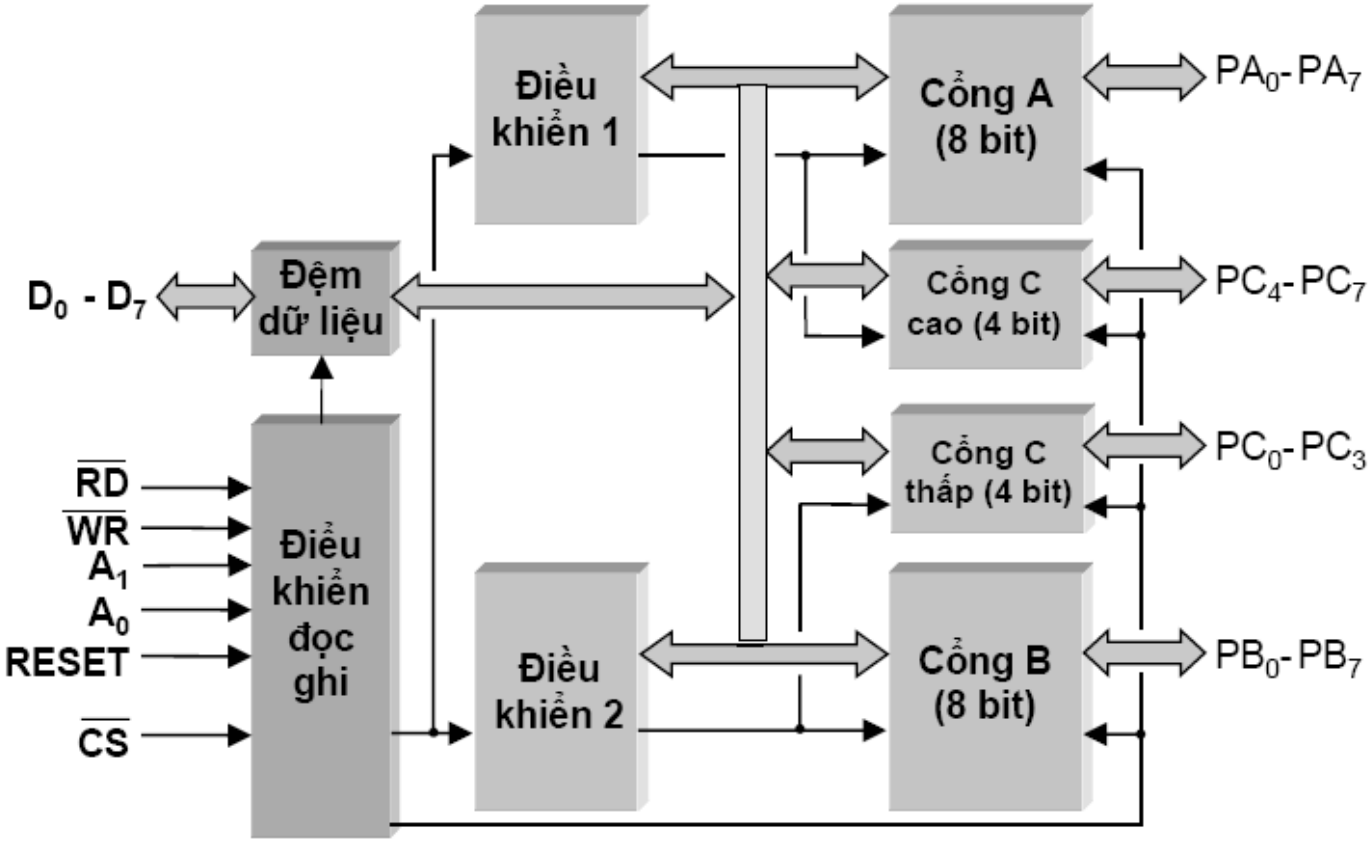

```
INPUT : P2.0=0
  CLR P2.0
  MOV A,P0
```

```
OUTPUT : P2.0=0
  SETB P2.0
  MOV P0,A
```

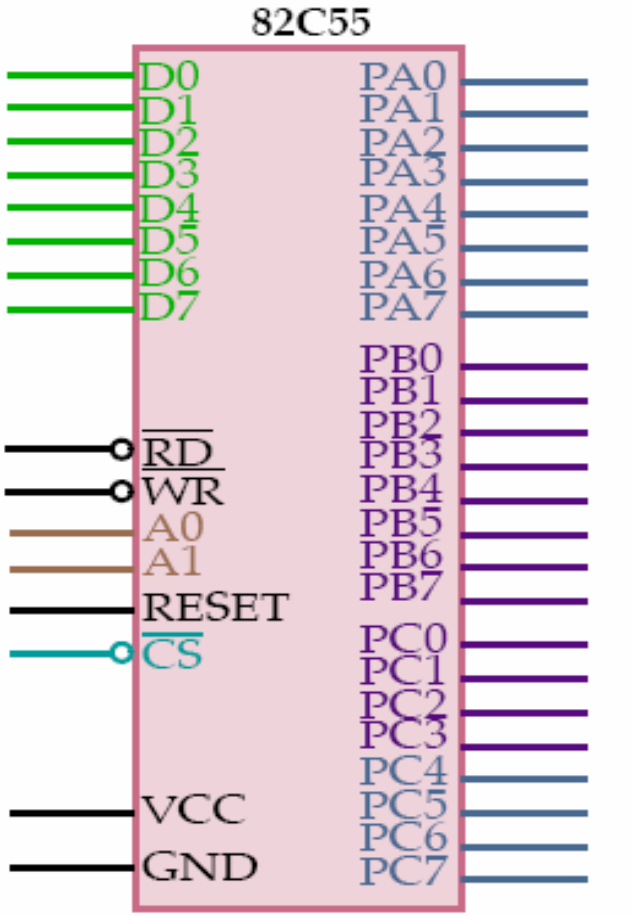
VI MẠCH GHÉP NỐI VÀO RA SONG SONG 8255

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

SƠ ĐỒ KHỐI CỦA 8255



CẤU TRÚC CỦA 8255 (có địa chỉ cổng 60H)



A_1	A_0	Thanh ghi	Chức năng
0	0	PA	Port A (có địa chỉ 60H)
0	1	PB	Port B
1	0	PC	Port C
1	1	CWR	Từ điều khiển (địa chỉ là 63H)

CÁC CHẾ ĐỘ (MODE) LÀM VIỆC CỦA 8255

❖ **MODE 0 : Chế độ vào/ra cơ sở**

Các cổng PA, PB, PC có thể được định nghĩa là các cổng vào hoặc ra

❖ **MODE 1 : Chế độ vào/ra có xung cho phép**

Các cổng PA, PB có thể định nghĩa là các cổng vào hoặc ra với các tín hiệu móc nối (handshaking) do các chân của cổng PC đảm nhận

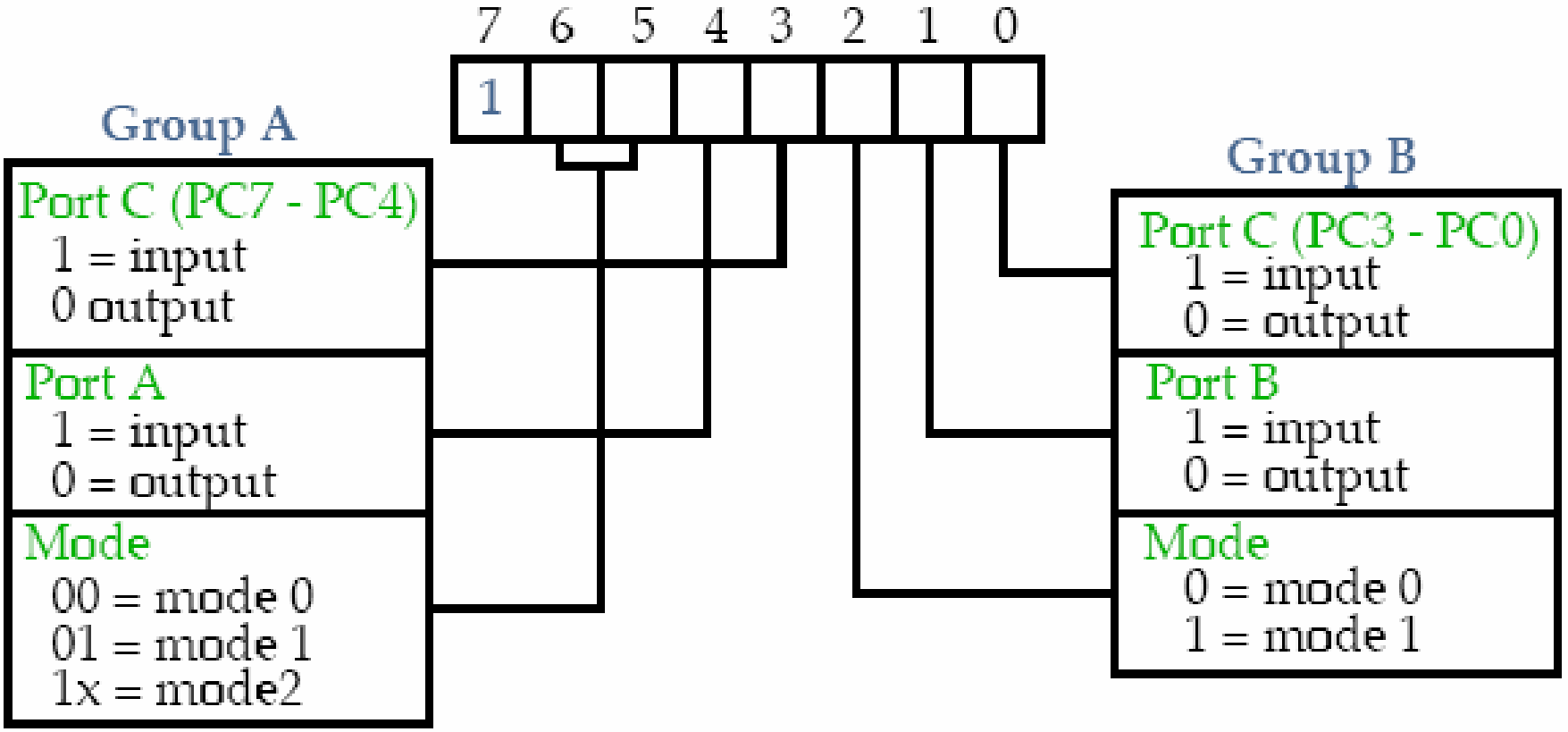
❖ **MODE 2 : Chế độ vào/ra 2 chiều**

-Cổng PA có thể được định nghĩa là cổng vào/ra 2 chiều với các tín hiệu móc nối (handshaking) do các chân của cổng PC đảm nhận

- Lúc này cổng PB có thể làm việc ở chế độ 0 hoặc 1

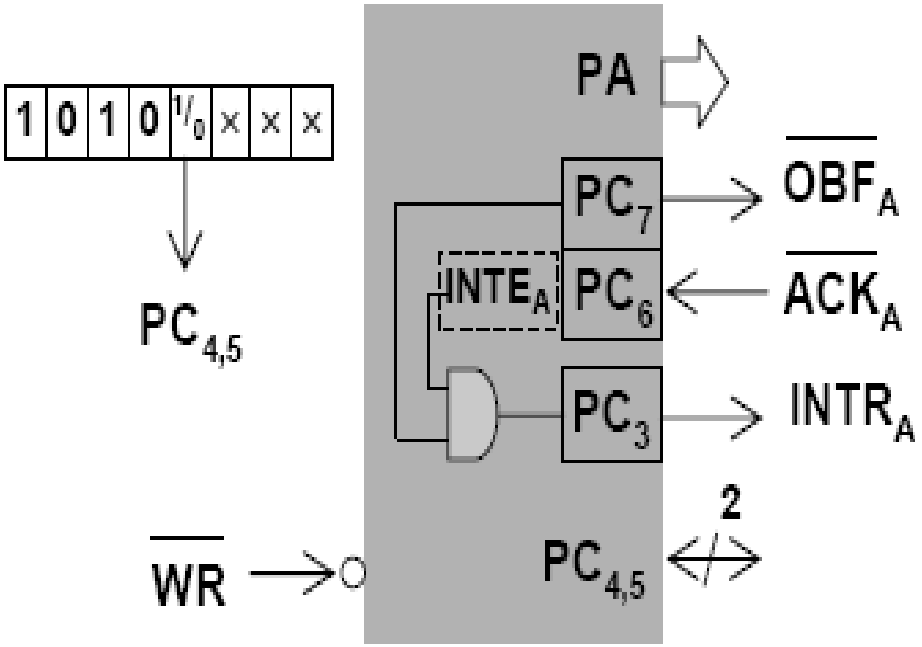
ĐỊNH NGHĨA CHẾ ĐỘ LÀM VIỆC CHO CÁC CỔNG

Để thiết lập chế độ làm việc cho các cổng PA,PB,PC ta thiết lập thanh ghi từ điều khiển (CWR) thích hợp

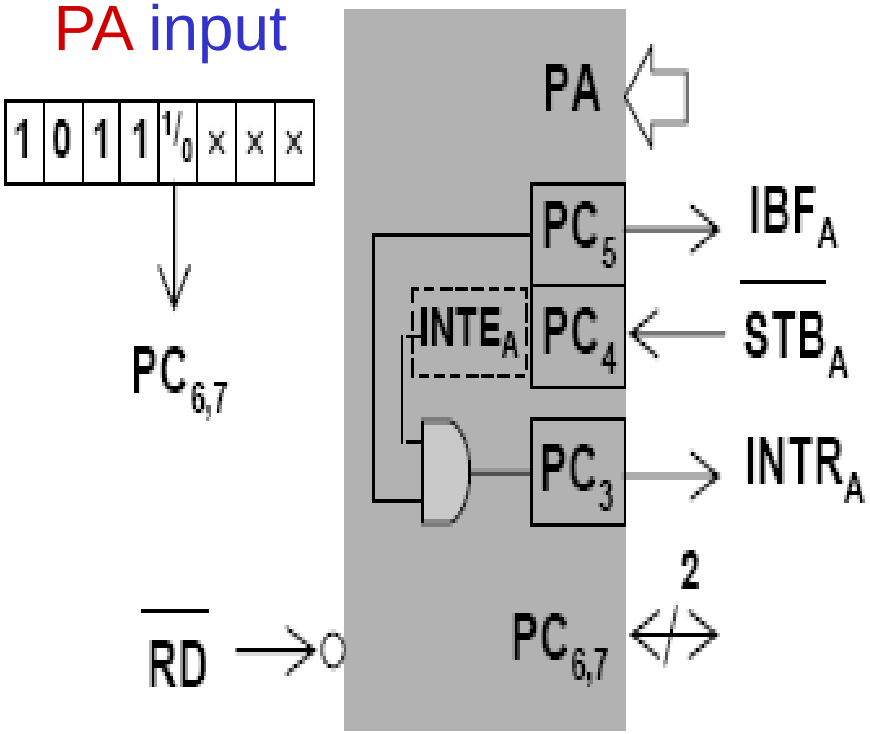


TÍN HIỆU HANDSHARING Ở MODE 1

PA output



PA input



$\overline{OBF}$: bộ đệm phát đầy

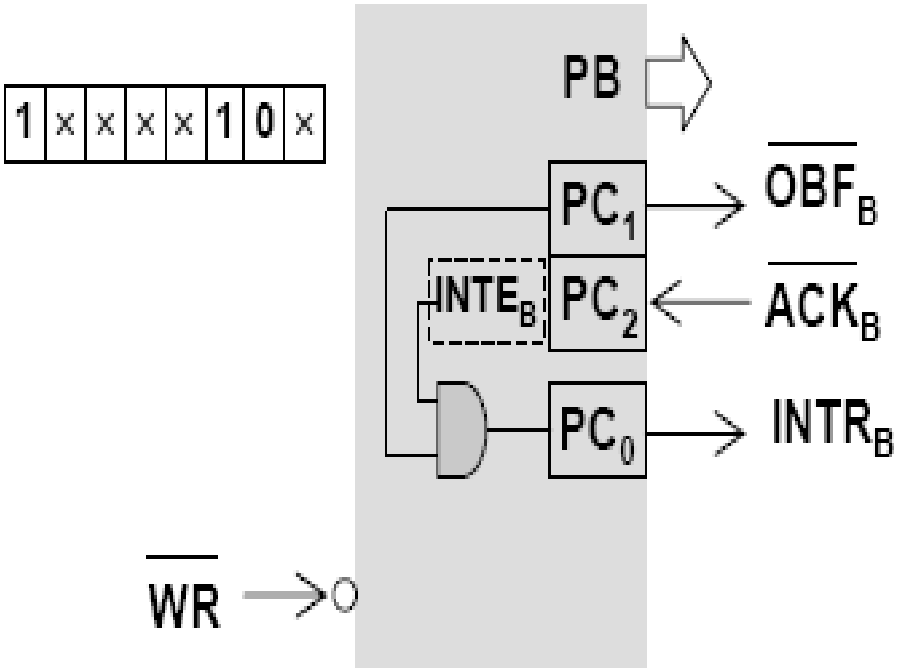
$\overline{ACK}$: đã nhận được dữ liệu

$\overline{INTR}$: tín hiệu yêu cầu ngắt của 8255

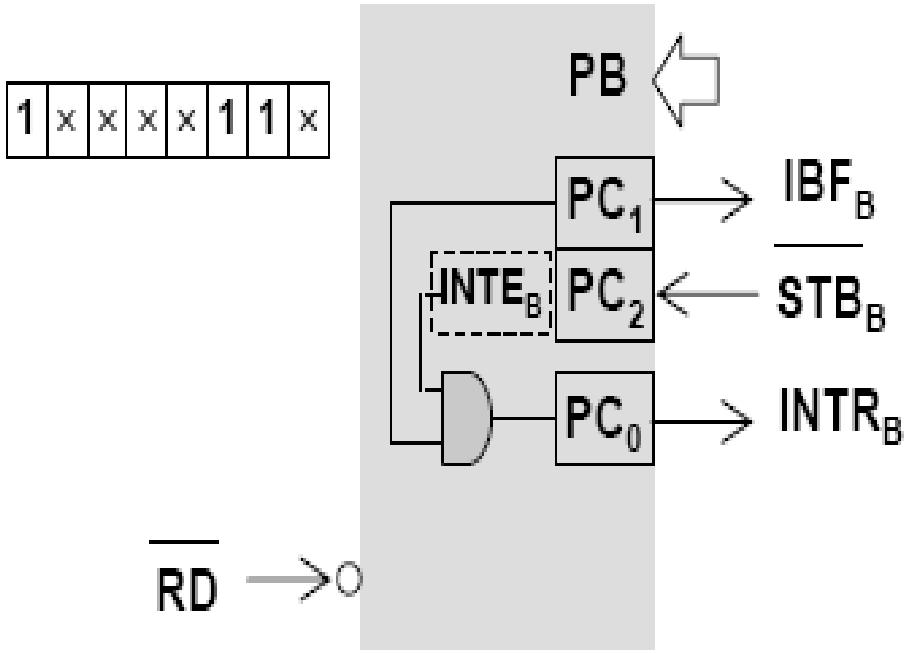
$\overline{IBF}$: bộ đệm thu đầy

$\overline{STB}$: cho phép chốt dữ liệu

TÍN HIỆU HANDSHARING Ở MODE 1

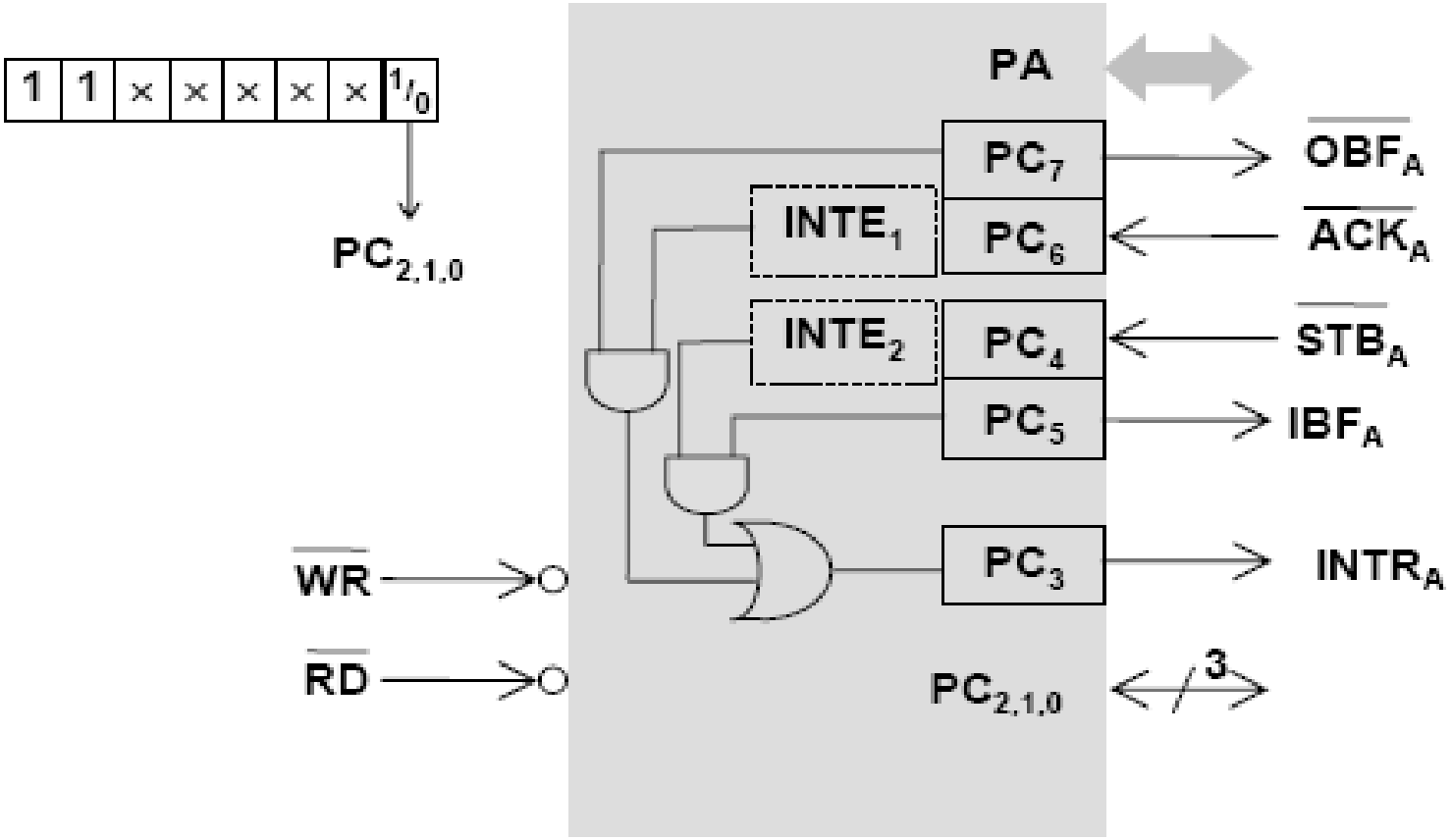


PB output

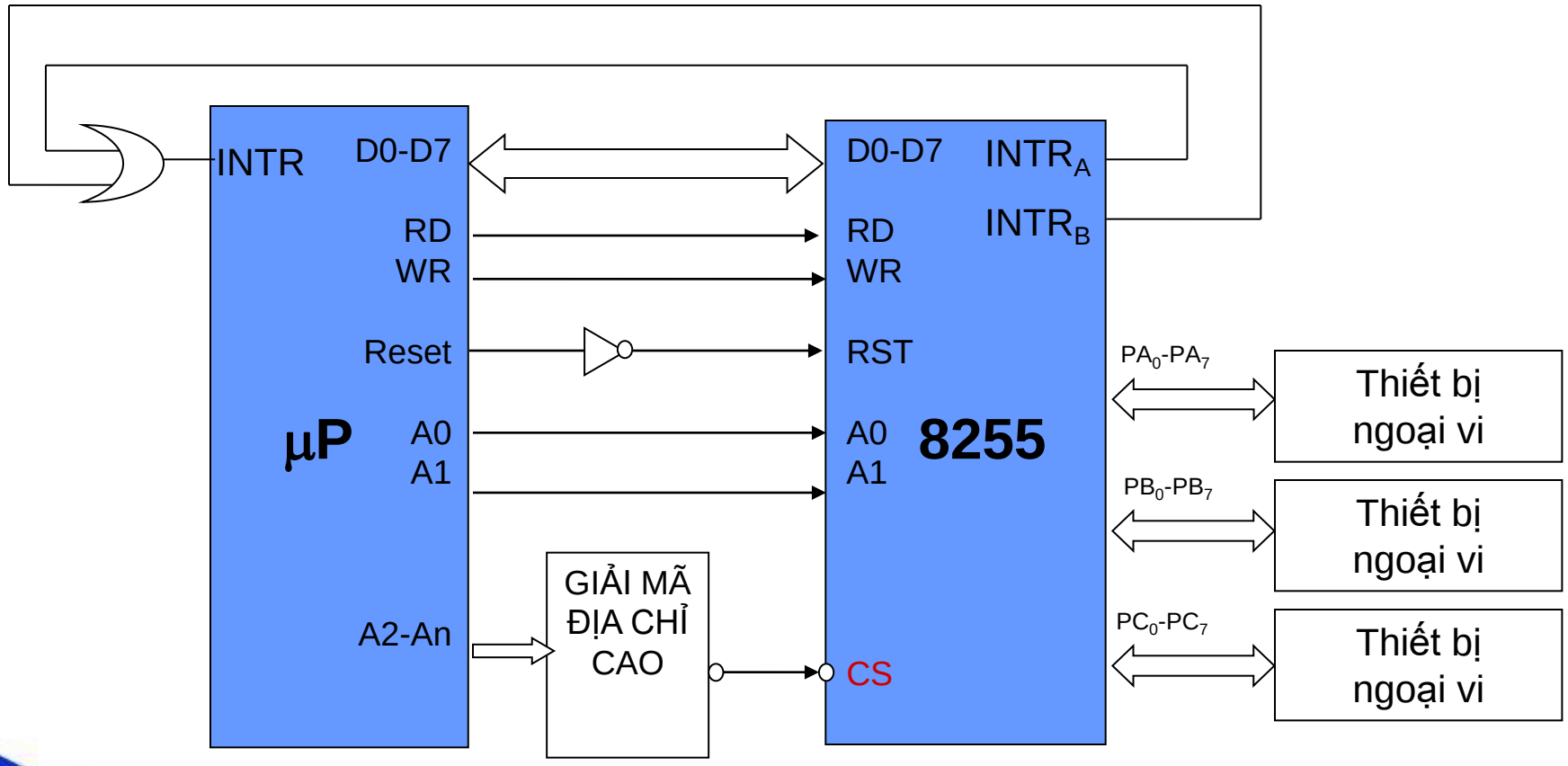


PB input

TÍN HIỆU HANDSHARING Ở MODE 2



GHÉP NỐI 8255 VỚI VI XỬ LÝ



LẬP TRÌNH CHO 8255

Ví dụ 1 : Giả sử chip 8255 có địa chỉ 60h, thiết lập để 8255 làm việc ở chế độ sau :

Port A : mode 0 , vào (I)

Port B : mode 0 , ra (O)

Port C (PCH) cao : mode 0 , vào

Port C thấp (PCL): mode 0 , ra

Xác định từ điều khiển= CWR?

BÀI GIẢI :

Từ điều khiển : CWR = 10011000b = 98h

Địa chỉ từ điều khiển = 63h?

Chương trình :

```
MOV   AL, 98h
OUT   63h, AL
```

LẬP TRÌNH CHO 8255

Ví dụ 2 : Giả sử chip 8255 có địa chỉ 60h

a- Thiết lập để 8255 làm việc ở chế độ sau :

Port A : mode 2, ra

Port C cao : mode 2 , vào

Port B : mode 1 , ra

Port C thấp : mode 1 , ra

b- Vẽ các chân tín hiệu của 8255

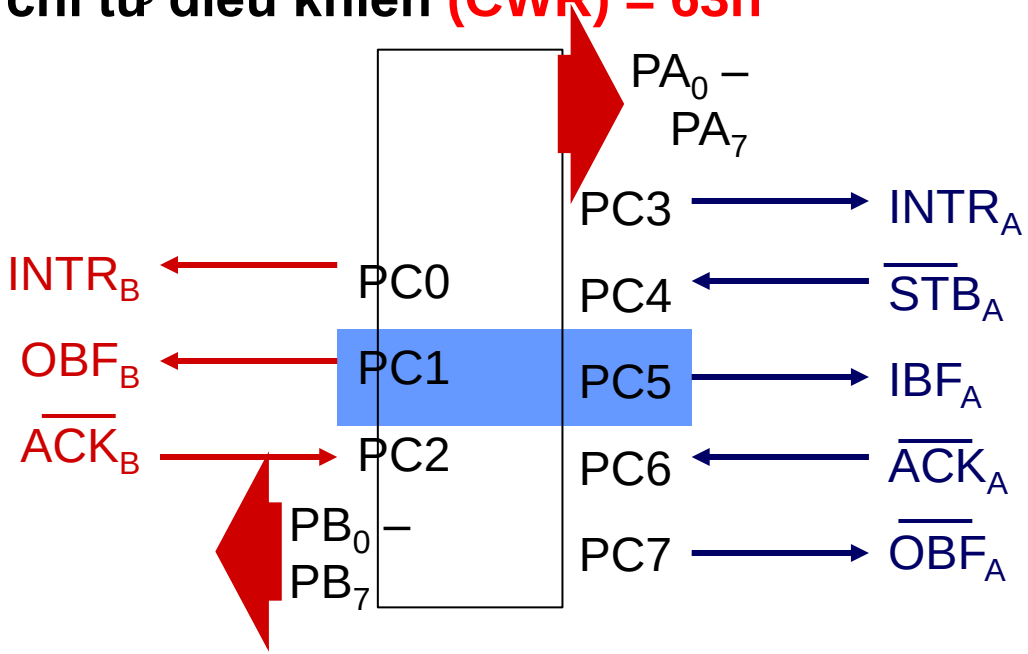
c- Port C còn bao nhiêu bit để làm cổng I/O?

LẬP TRÌNH CHO 8255

BÀI GIẢI :

a/ Từ điều khiển : CWR = 11001100B = CCh

b/ Địa chỉ từ điều khiển (CWR) = 63h




```
MOV   AL, CCh
OUT   63h, AL
```

c- Các chân của PC đã dùng hết để làm các tín hiệu Hanshaking cho PA và PB nên PC không còn chân nào để làm cổng I/O cả

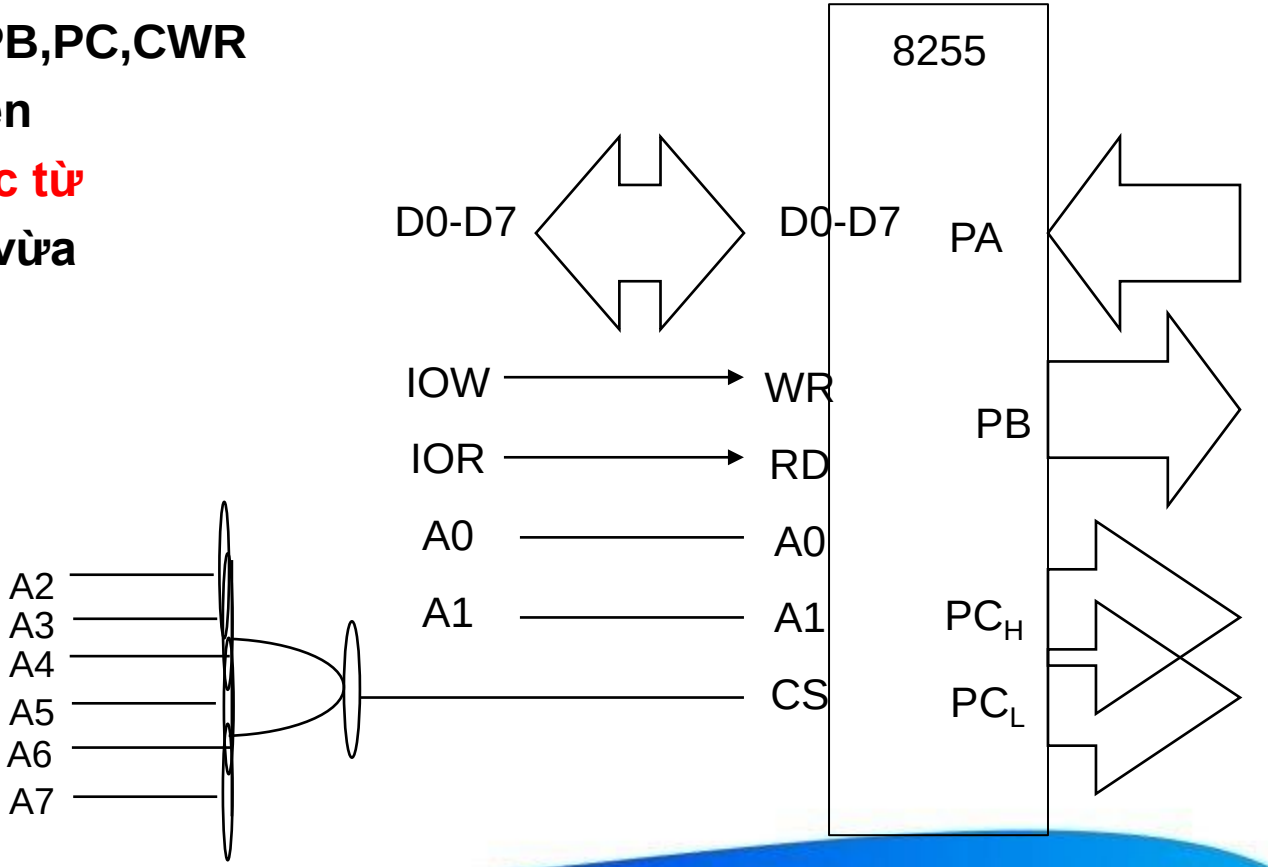
LẬP TRÌNH CHO 8255

Ví dụ 3 : Chip 8255 có phối ghép như hình bên. Nó được cấu hình để làm việc như sau :PA (mode0, input); PB(mode0 , output); PC(mode0 , output)

a- Xác định địa chỉ PA,PB,PC,CWR

b- Xác định từ điều khiển

c- Viết chương trình đọc từ cổng PA và xuất giá trị vừa đọc ra cổng PB và PC



LẬP TRÌNH CHO 8255

a- 8255 hoạt động khi tín hiệu đưa vào CS=0

→ A7-A2 = 010100 (A7-A2)

A7-A2	A1A0	Address	Port
010100	00	50h	PA
010100	01	51h	PB
010100	10	52h	PC
010100	11	53h	CWR

b- Từ điều khiển (CWR) = 10010000b=90h

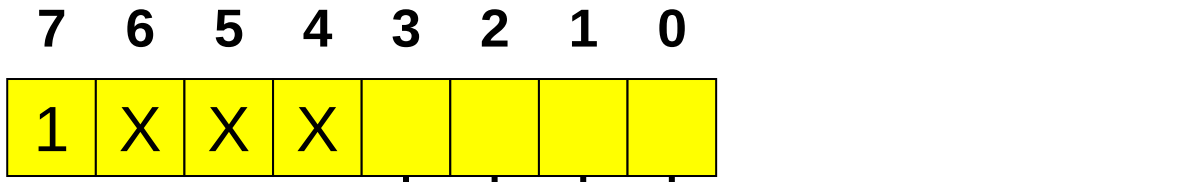
c- Chương trình

```
MOV AL,90H      ; đưa từ điều khiển vào AL
OUT 53H,AL      ; gửi từ điều khiển đến CWR
IN AL,50H       ; đọc từ port A vào thanh ghi AL
OUT 51H,AL      ; xuất AL ra port B
OUT 52H,AL      ; xuất AL ra port C
```

LẬP/XÓA BIT

Khi các bit của port C dùng làm các tín hiệu Hanshacking cho portA và portB trong mode1 hoặc mode2. Các bit này có thể được set/clear (lập/xóa)

Để lập hoặc xóa 1 bit của PC ta dùng từ điều khiển sau:



1=SET
0=CLEAR

SELEC BIT

BIT		PC							
		0	1	2	3	4	5	6	7
CWR	3	0	0	0	0	1	1	1	1
	2	0	0	1	1	0	0	1	1
	1	0	1	0	1	0	1	0	1

LẬP/XÓA BIT

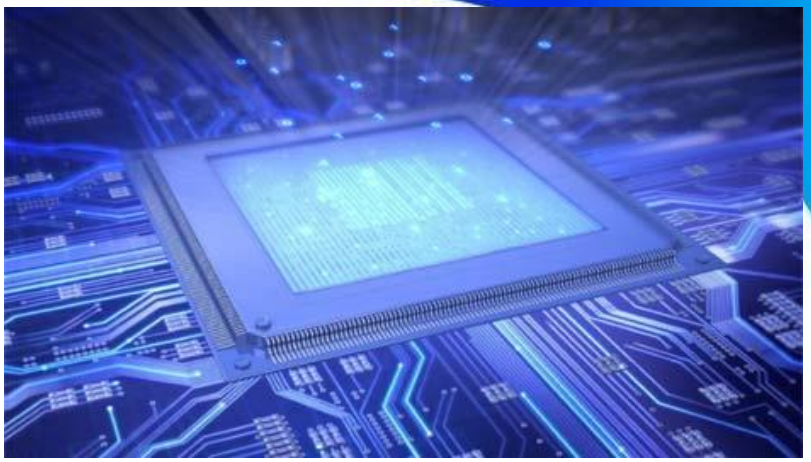
Ví dụ : Giả sử địa chỉ của 8255 là 60h

Clear bit 6 của PC

```
OUT 63H, 0CH          ; 00001100B =0CH
```

Set bit 7 của PC

```
OUT 63H, 0FH          ; 00001111B =0FH
```



Chương 4
Kiến trúc phần mềm của hệ thống vi xử lý

TẬP LỆNH CỦA 8086

- **Nhóm lệnh di chuyển dữ liệu**
  MOV, XCHG, POP, PUSH, POPF, PUSHF, IN, OUT

- **Các lệnh số học :**
  ADD, ADC, SUB, MUL, IMUL, DIV, IDIV, INC, DEC

- **Các lệnh logic :**
  AND, OR, NOT, NEG, XOR

- **Lệnh quay và dịch:**
  RCL, RCR, ROL, ROR, SAL, SAR, SHL, SHR

- **Các lệnh thao tác chuỗi**
  MOVS, MOVSB, MOVSW, CMPS

- **Các lệnh điều khiển chương trình :**
  JMP, JE, JZ, JNE or JNZ, JG, JGE, JL, JLE, JA, JB, JAE, JBE, JC, JNC, CALL, INT, IRET

NHÓM LỆNH DI CHUYỂN DỮ LIỆU

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]

NHÓM 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]

NHÓM 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

NHÓM LỆNH THAO TÁC CHUỖI

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 tuỳ thuộc vào cờ hướng DF là 0/1
- Cập nhật cờ AF, CF, OF, PF, SF, ZF

NHÓM LỆNH SỐ HỌC

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**

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**

NHÓM LỆNH SỐ HỌC

Lệnh INC

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

Lệnh DEC

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

NHÓM LỆNH SỐ HỌC

Lệnh MUL

- Lệnh nhân số không dấu
- Cú pháp: MUL nguồn
- Thực hiện:
  - AX=AL* nguồn8bit
  - DXAX=AX*nguồn16bit
- Lệnh này thay đổi cờ: CF, OF
- Ví dụ:
  - MUL BL

Lệnh IMUL

- nhân số có dấu

NHÓM LỆNH SỐ HỌC

Lệnh DIV

- **Lệnh chia 2 số không dấu**
- **Cú pháp: DIV nguồn**
- **Thực hiện:**
  - **AL = thương (AX / nguồn8bit) ; AH=dư (AX / nguồn8bit)**
  - **AX = thương (DXAX / nguồn16bit) ; DX=dư (DXAX / nguồn16bit)**
- **Lệnh này không thay đổi cờ**
- **Ví dụ:**
  - **DIV BL**

Lệnh IDIV

- **chia 2 số có dấu**

NHÓM LỆNH SỐ HỌC

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

- Chú ý : Lệnh CMP thường đứng trước các lệnh nhảy có điều kiện.

NHÓM LỆNH LOGIC, DỊCH & QUAY

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

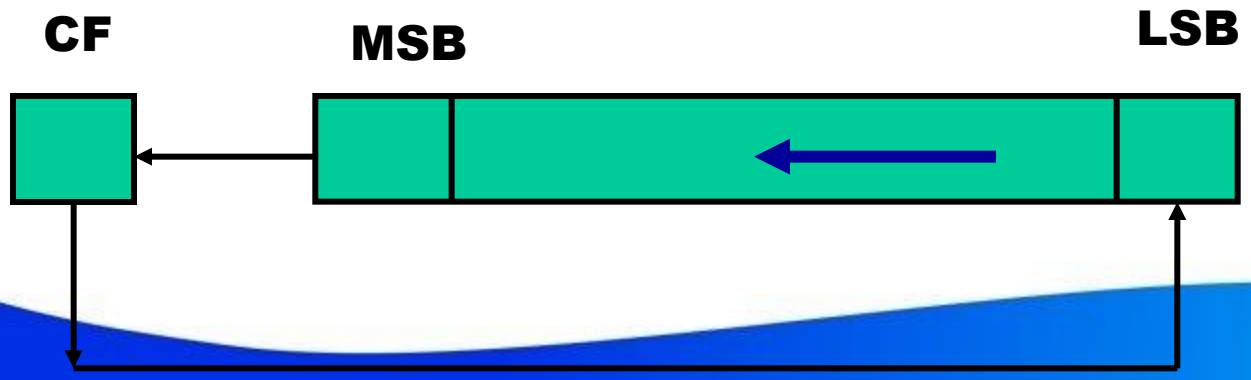
NHÓM LỆNH LOGIC, DỊCH & QUAY

Lệnh RCL

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

  RCL Đích, 1

  RCL Đích, Số lần quay (80286 trở 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**



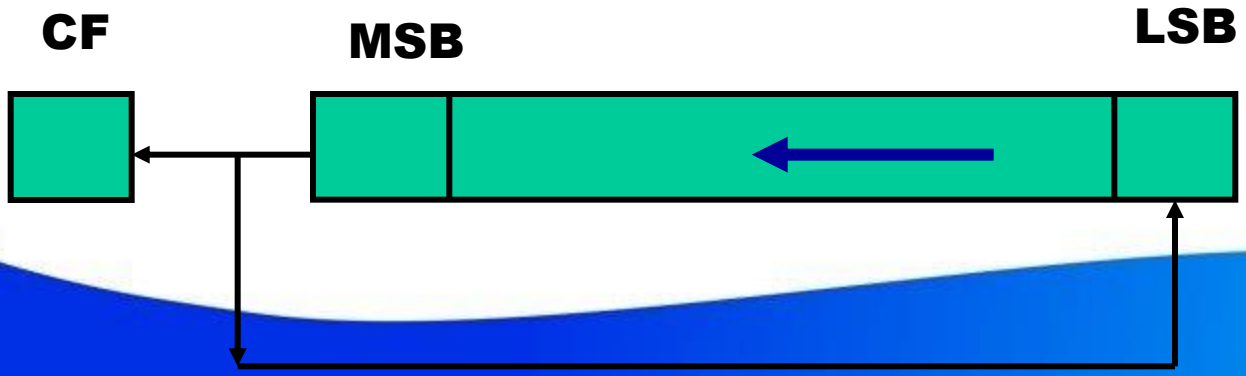
NHÓM LỆNH LOGIC, DỊCH & QUAY

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 trở 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**



NHÓM LỆNH LOGIC, DỊCH & QUAY

Lệnh SHL

- **Lệnh dịch trái số học**
- **Cú pháp: SHL Đích, CL (với số lần dịch lớn hơn 1)**

  SHL Đích, 1

  SHL Đích, số lần dịch (80286 trở lên)

- **Thực hiện: dịch trái đích CL bit tương đương với Đích=Đích*2^{CL}**
- **Lệnh này thay đổi cờ SF, ZF, PF**

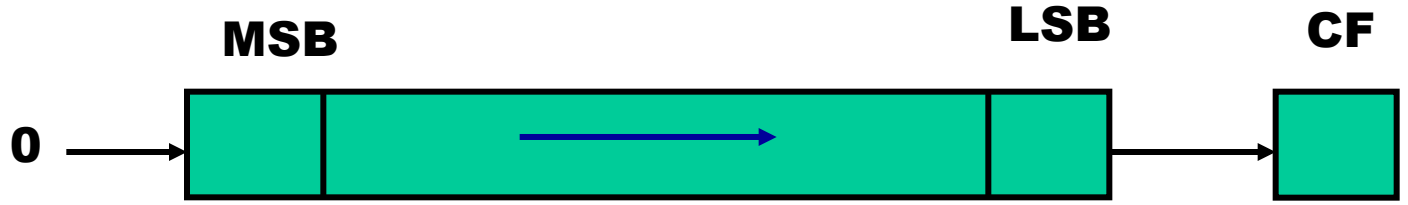
NHÓM LỆNH LOGIC, DỊCH & QUAY

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 trở 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

NHÓM LỆNH XUẤT/ NHẬP CỔNG

Lệnh IN

- Dùng để đọc 1 byte hoặc 2 byte dữ liệu từ cổng vào thanh ghi AL hoặc AX
- Cú pháp: IN Acc, Port
- Lệnh này không tác động đến cờ
- Ví dụ:
  - IN AX, 00H
  - IN AL, 0F0H
  - 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

NHÓM LỆNH THAO TÁC CHUỖI

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 tuỳ thuộc vào cờ hướng DF là 0/1
- Lệnh này không tác động đến cờ
- Ví dụ: MOVS byte1, byte2

NHÓM LỆNH ĐIỀU KHIỂN

- Lệnh nhảy không điều kiện : JMP
- Lệnh nhảy có điều kiện :
- Lệnh vòng lặp : LOOP
- Lệnh gọi chương trình con : CALL
- Lệnh xử lý ngắt : INT, IRET

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)**

  - **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: ± 32 Kbytes so với lệnh tiếp theo lệnh JMP
  - Ví dụ:

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

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

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

Thực hiện: IP=IP+ độ lệch

IP=BX **IP=[BX+1] [BX]**

Nhảy gián tiếp

E 9	Độ lệchLo	Độ lệchHi

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
```

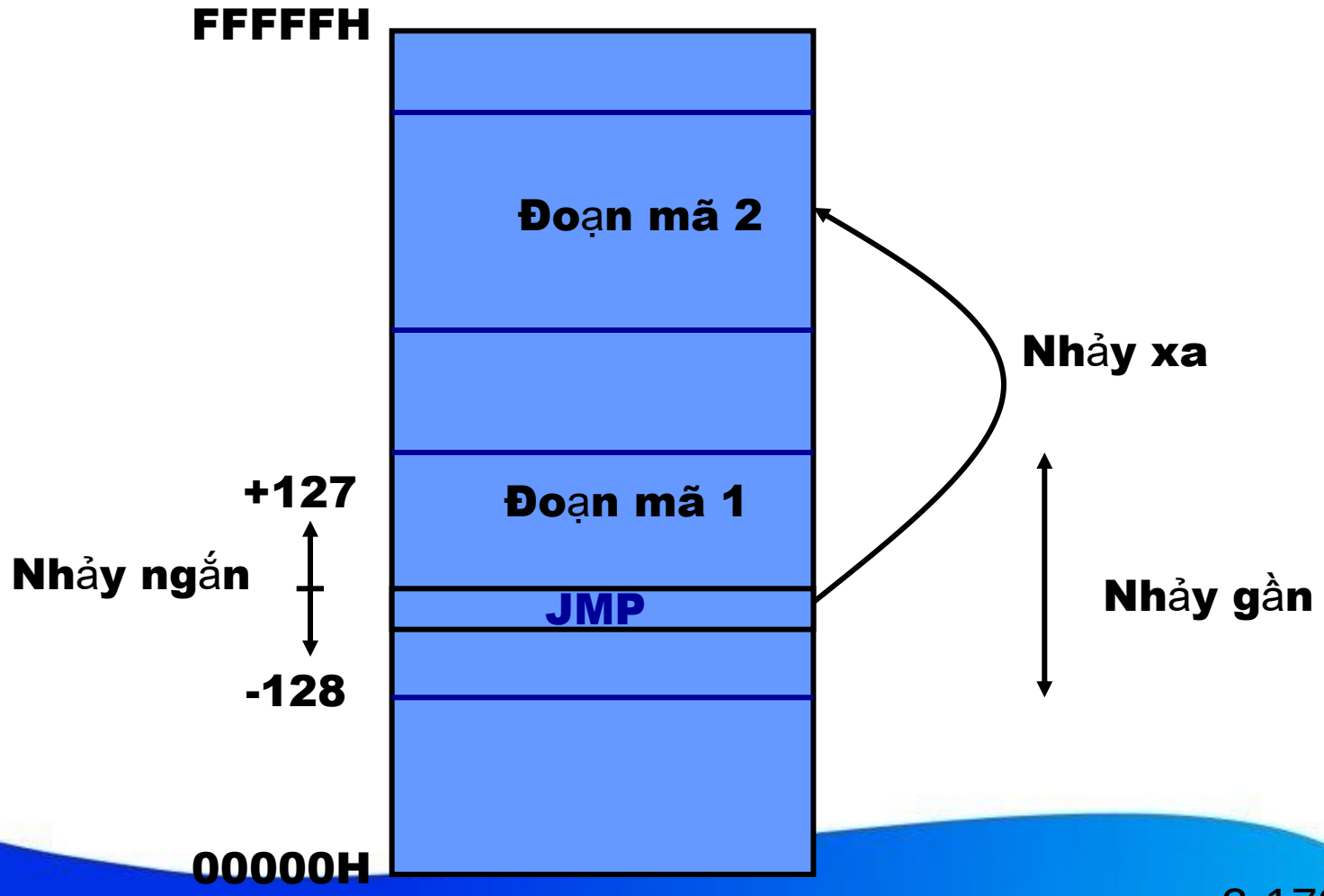
Thực hiện: IP=IP của nhãn
CS=CS của nhãn

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

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

LỆNH NHẢY KHÔNG ĐIỀU KIỆN JMP

Tóm tắt :



LỆNH NHẢY CÓ ĐIỀU KIỆN

Đối với số có dấu :

JE or JZ, JNE or JNZ, JG, JGE, JL, JLE , JC, JNC ...

Đối với số không dấu :

JA, JB, JAE, JBE , JC, JNC ...

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
```

LỆNH VÒNG LẶP

LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ

Là lệnh phối hợp giữa DEC CX và JNZ

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

Lặp đến khí CX=0

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

Lặp đến khí CX=0 hoặc AL<>10

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

Lặp đến khí CX=0 hoặc AL=10

LỆNH GỌI CHƯƠNG TRÌNH CON 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
```

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

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

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

```
CALL WORD PTR [BX]
```

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

LỆNH GỌI CHƯƠNG TRÌNH CON 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
```

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

```
CALL DWORD PTR [BX]
```

Cất CS vào ngăn xếp
Cất 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 & 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 & IRET

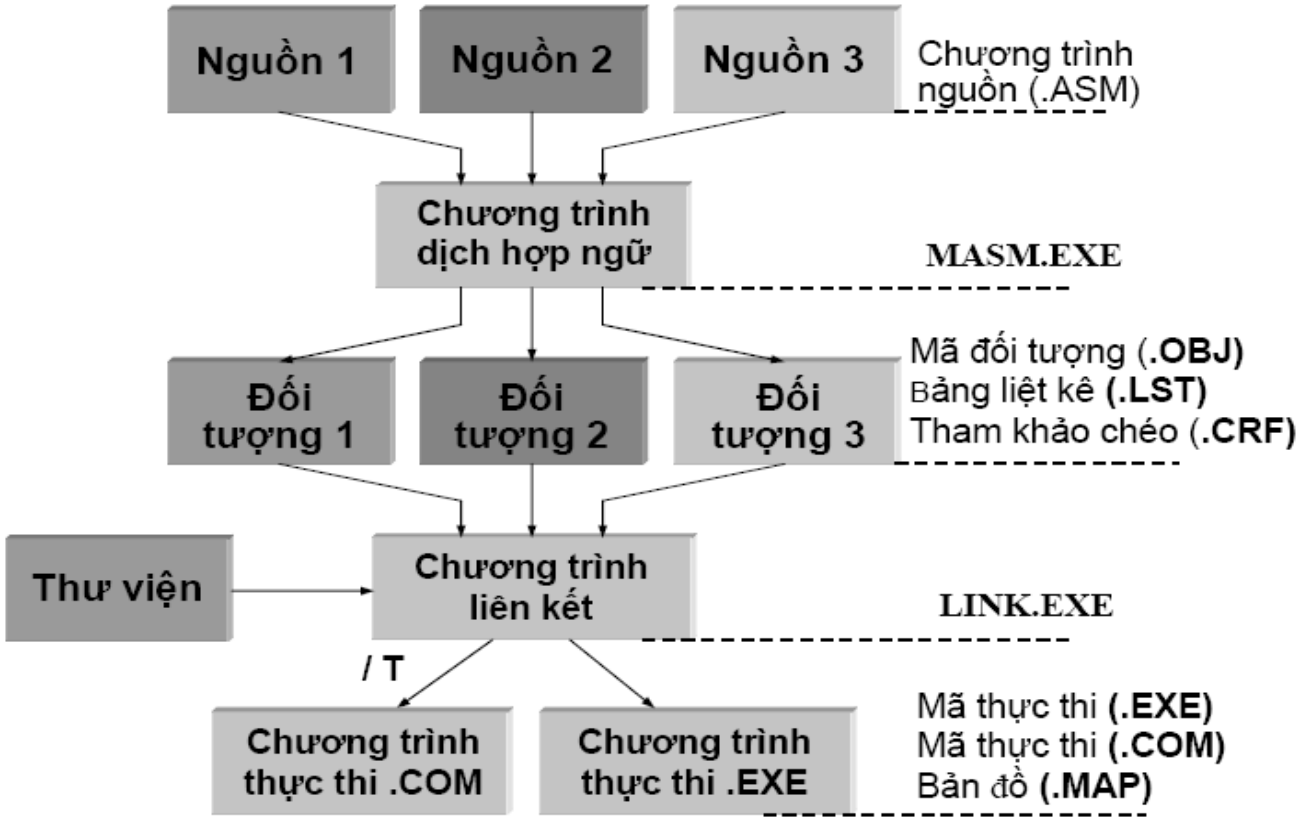
Thực hiện INT:

- Cất thanh ghi cờ vào ngăn xếp
- IF=0 (cấm các ngắt khác tác động), TF=0 (chạy suốt)
- Cất CS vào ngăn xếp
- Cất 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

DỊCH CHƯƠNG TRÌNH



KHUNG CHƯƠNG TRÌNH

```
.MODEL SMALL
.STACK   <số byte>
.DATA
< Các khai báo biến >
.CODE
Main PROC
< Thân chương trình chính>
<Kết thúc, trở về DOS>
Main ENDP
< Khai báo chương trình con>
END
```

```
CODE  segment para public ‘code’
        assume cs:code,ds:data,ss:stack
        <Các chương trình con>
Main PROC
        <Thân chương trình chính>
        <Kết thúc, trở về DOS>
Main ENDP
CODE ENDS
DATA segment para public 'data‘
        <Các khai báo biến và hằng>
DATA ENDS
STACK segment stack 'stack‘
        <Khai báo vùng đệm>
STACK ENDS
END Main
```

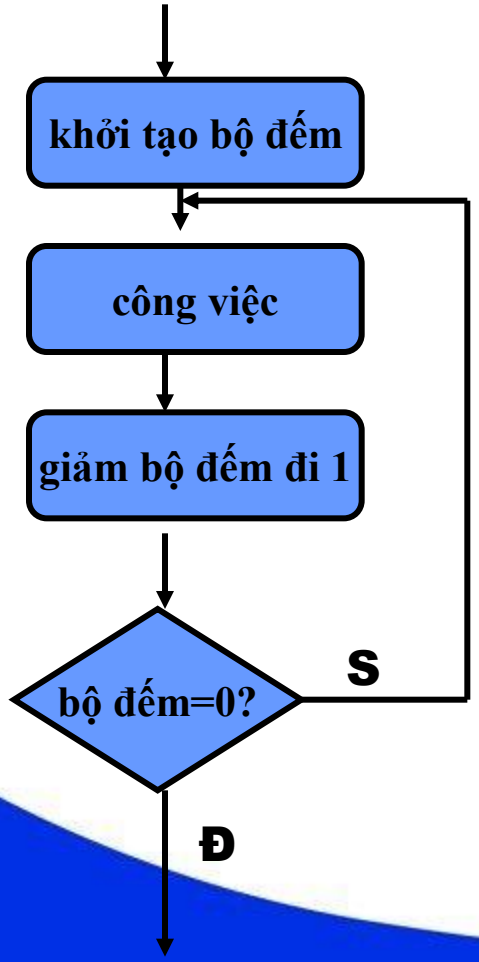
LẬP TRÌNH CHO 8086

Ví dụ 1 : Hiện chữ “Hello” ra màn hình

```
.Model   Small
.Stack   100
.Data
         MSG     DB      ‘Hello! $’
.Code
MAIN     Proc
         ;khới đầu cho DS
         MOV     AX, @data
         MOV     DS, AX
         ;Hiển thị lời chào dùng hàm 9 ngắt 21h
         MOV     AH,9
         LEA     DX, MSG
         INT     21H
         ;trở về DOS dùng hàm 4Ch,ngắt 21h
         MOV     AH, 4CH
         INT     21H
MAIN     Endp
         END MAIN
```

LẬP TRÌNH CHO 8086

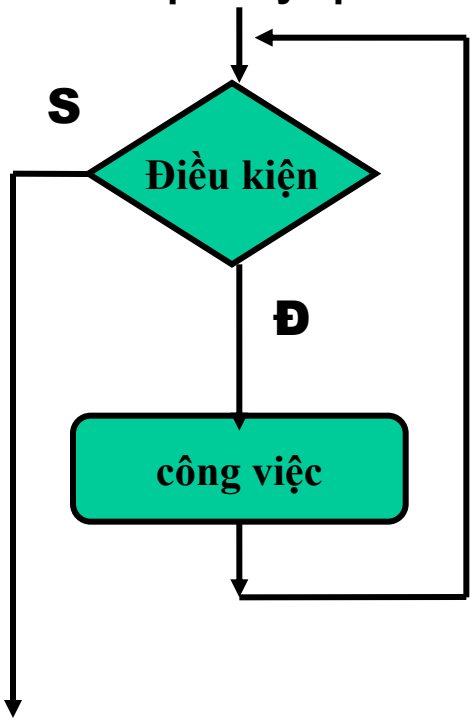
Ví dụ 2 : Hiển thị 50 kí tự '$' trên màn hình




```
.Model  Small
.Stack  100
.Data
        MSG     DB      'Hello! $'
.Code
MAIN    Proc
        MOV CX, 50      ;số lần lặp
        MOV  AH,2       ;hàm hiển thị
        MOV   DL,'$'    ;DL chứa ký tự cần hiển thị
HIEN:   INT 21H         ; Hiển thị
        LOOP   HIEN
        ;trở về DOS dùng hàm 4Ch,ngắt 21h
        MOV    AH, 4CH
        INT      21H
MAIN    Endp
        END MAIN
```

LẬP TRÌNH CHO 8086

Ví dụ 3 : Nhập chuỗi ký tự từ bàn phím cho đến khi người dùng ấn phím Enter. Đếm và hiển thị số ký tự đã nhập (< 10 ký tự).




```
            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:
            MOV AH,2        ; hàm hiển thị 1 ký tự
            MOV DX,CX
            ADD DL,30H      ; đổi sang mã ASCII
            INT 21H         ; hiển thị
```

➢ Cải tiến chương trình để hiển thị số ký tự >10 ký tự