

KỸ THUẬT VI XỬ LÝ

(MicroProcessor Technology)

CHƯƠNG 4: GHÉP 8088 VỚI BỘ NHỚ VÀ TỔ CHỨC VÀO/RA DỮ LIỆU

☺: Nguyễn Trung Kiên

✉: kiennt@neu.edu.vn

NỘI DUNG

4.1. Các vi mạch phụ trợ cho 8088

4.2. Phối ghép 8088 với bộ nhớ

4.3. Phối ghép 8088 với hệ thống vào ra

NỘI DUNG



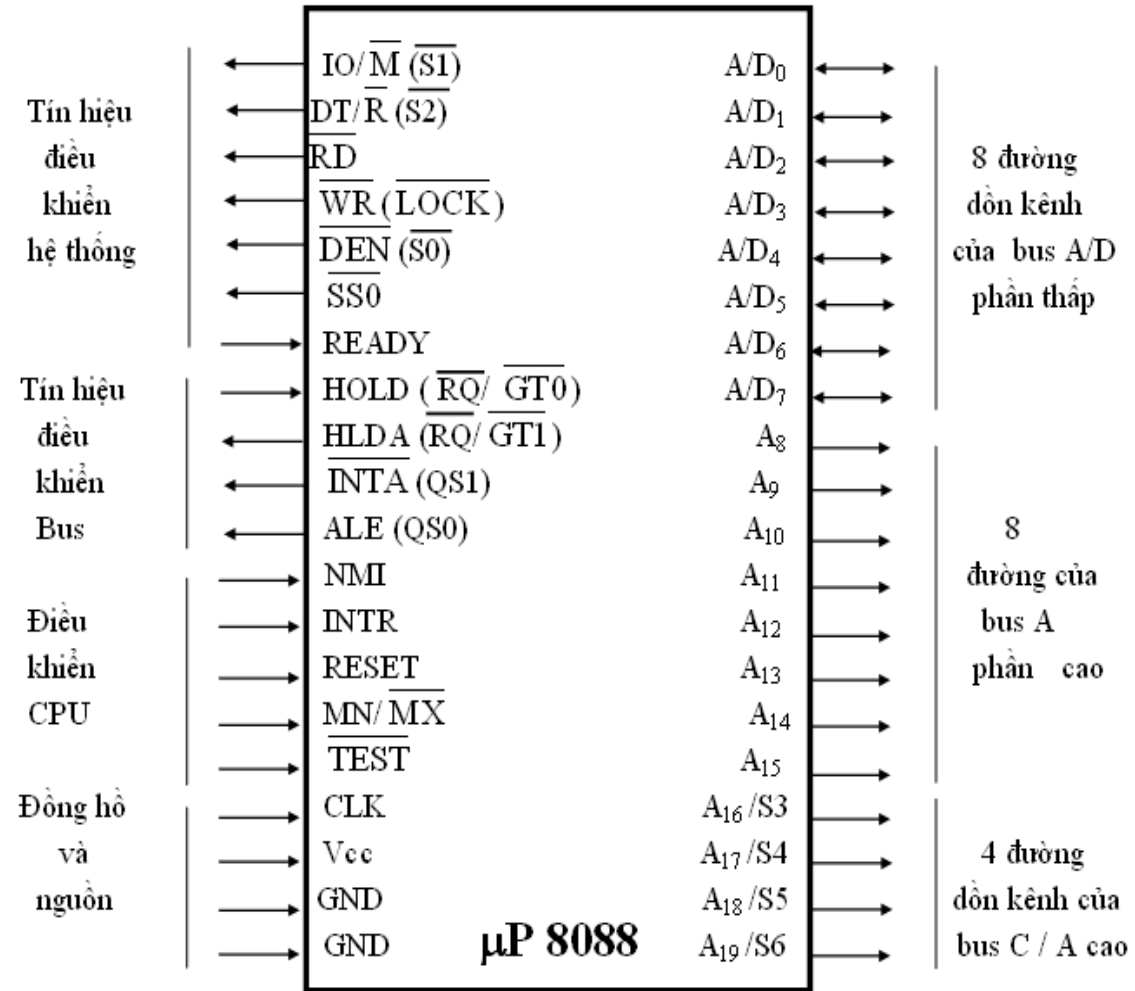
4.1. Các vi mạch phụ trợ cho 8088

4.2. Phối ghép 8088 với bộ nhớ

4.3. Phối ghép 8088 với hệ thống vào ra

4.1. Các vi mạch phụ trợ cho 8088

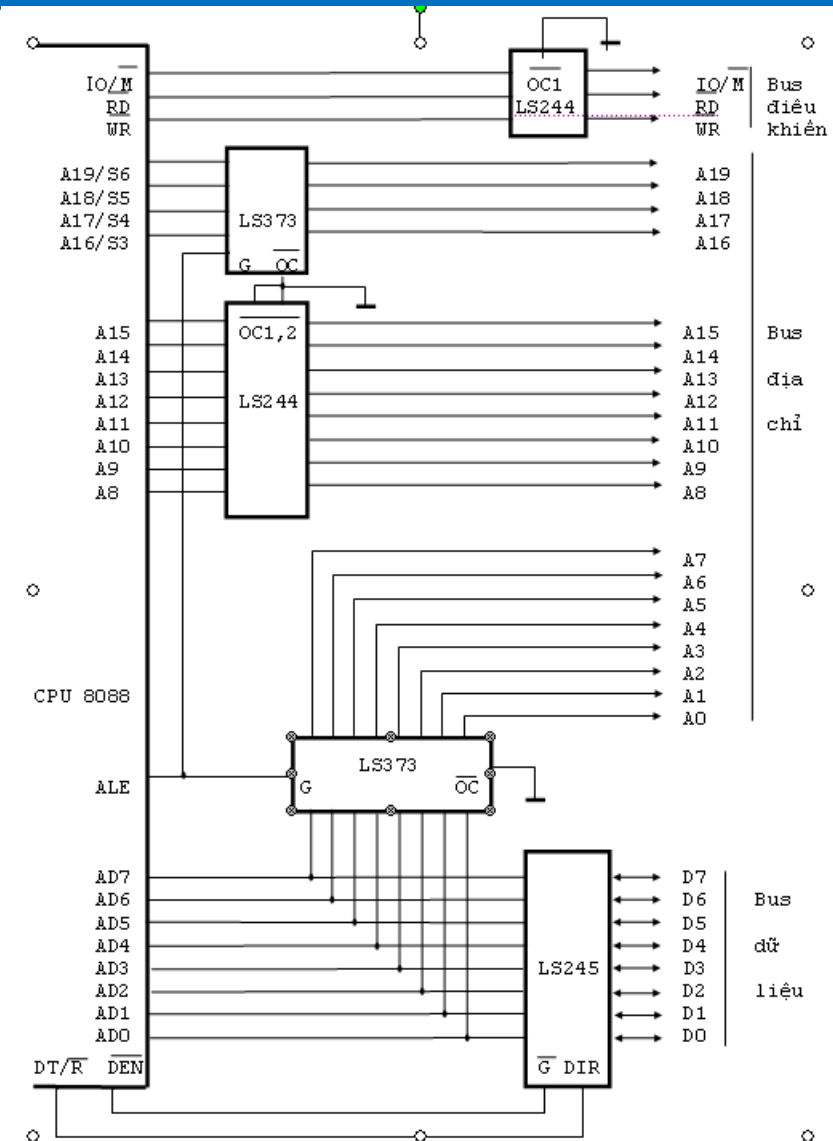
❖ Các tín hiệu của 8088 ở chế độ MIN và MAX



Hình 5. 1. Các tín hiệu của 8088 ở chế độ MIN và (MAX).

4.1. Các vi mạch phụ trợ cho 8088

- ❖ Phân kênh để tách thông tin và việc đệm cho các bus.



Hình 5.3. Bus hệ thống có khuếch đại đệm.

4.1. Các vi mạch phụ trợ cho 8088

❖ Mạch tạo xung nhịp 8284:

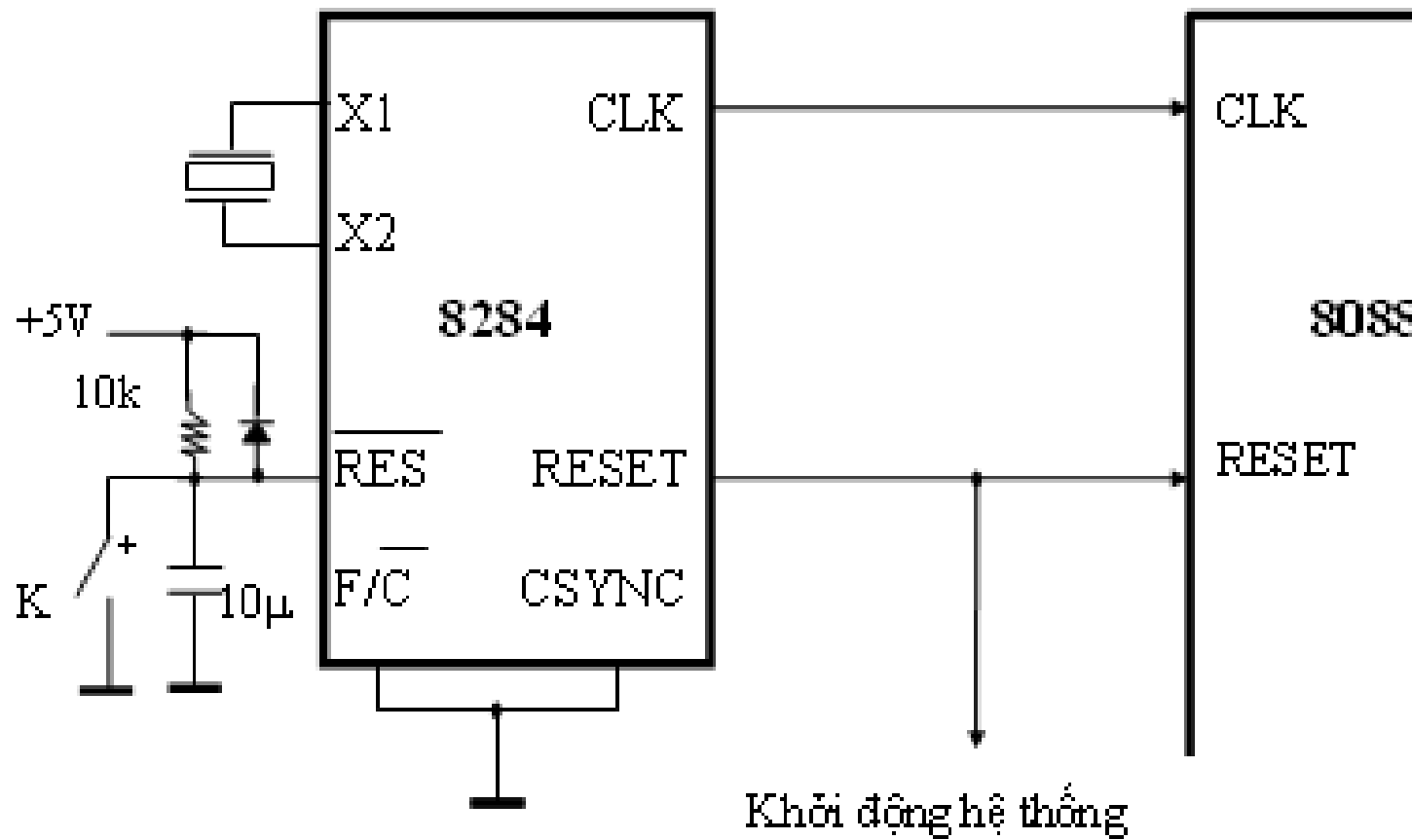
Tên của các chân tín hiệu:

CSYNC	1	18	Vcc	CSYNC[O] : Clock synchronisation
PCLK	2	17	X1	PCLK[O] : Peripheral clock
$\overline{\text{AEN1}}$	3	16	X2	$\overline{\text{AEN1}}, \overline{\text{AEN2}}$ [I] : Address enable
RDY1	4	15	$\overline{\text{ASYNC}}$	RDY1, RDY2 [I] : Bus ready
READY	5	14	EFI	READY[O] : Tới READY của 8088
RDY2	6	13	F/ $\overline{\text{C}}$	CLK[O] : Tới READY của 8088
$\overline{\text{AEN2}}$	7	12	OSC	X1, X2 [I] : Crystal
CLK	8	11	$\overline{\text{RES}}$	$\overline{\text{ASYNC}}$ [I] : Ready synchronisation
GND	9	10	RESET	select

EFI [I] : External frequency input
F/ $\overline{\text{C}}$ [I] : Frequency/Crystal; OSC [O] : Osc output;
RES [I] : Reset input; RESET [O] : Reset output.

Hình 5. 5. Mạch tạo xung đồng hồ 8284 cho CPU 8088

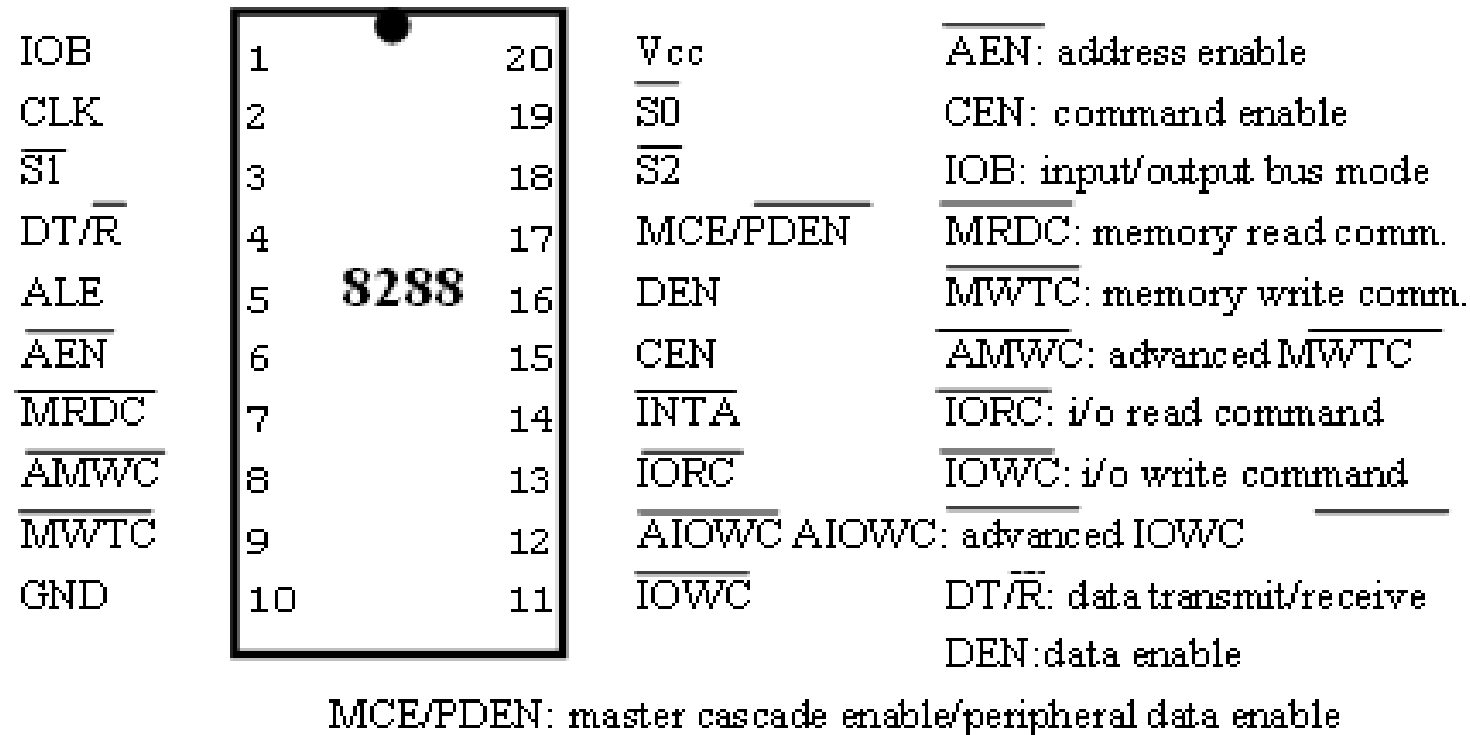
4.1. Các vi mạch phụ trợ cho 8088



Hình 5.6. Mạch 8284 nối với 8088.

4.1. Các vi mạch phụ trợ cho 8088

❖ Mạch điều khiển bus 8288



Hình 5. 7. Mạch tạo xung điều khiển 8288

4.1. Các vi mạch phụ trợ cho 8088

❖ Các tín hiệu chính của 8288 bao gồm:

- S2, S1, S0 [I, I, I] : là các tín hiệu trạng thái lấy thẳng từ CPU. Từ các tín hiệu này, 8288 sẽ tạo ra các tín hiệu điều khiển khác nhau tại các chân ra của nó để điều khiển hoạt động của các thiết bị nối với CPU.
- CLK [I]: đây là đầu vào nối với xung đồng hồ hệ thống (từ mạch 8284) và dùng để đồng bộ toàn bộ các xung điều khiển đi ra từ mạch 8288.
- AEN [I]: là tín hiệu đầu vào để sau một khoảng thời gian trễ cỡ 150 ns sẽ kích hoạt các tín hiệu điều khiển ở đầu ra của 8288.
- CEN [I]: là tín hiệu đầu vào để cho phép đưa ra tín hiệu DEN và các tín hiệu điều khiển khác của 8288.
- IOB [I]: tín hiệu để điều khiển mạch 8288 làm việc ở các chế độ bus khác nhau.
 - Khi IOB =1 mạch 8288 làm việc ở chế độ bus vào/ra, khi IOB = 0 mạch 8288 làm việc ở chế độ bus hệ thống (như trong các máy IBM PC). cho các bộ nhớ chậm có được thêm thời gian ghi.
- IORC [O]: tín hiệu điều khiển đọc thiết bị ngoại vi. Nó kích hoạt các thiết bị được chọn để các thiết bị này đưa dữ liệu ra bus.

4.1. Các vi mạch phụ trợ cho 8088

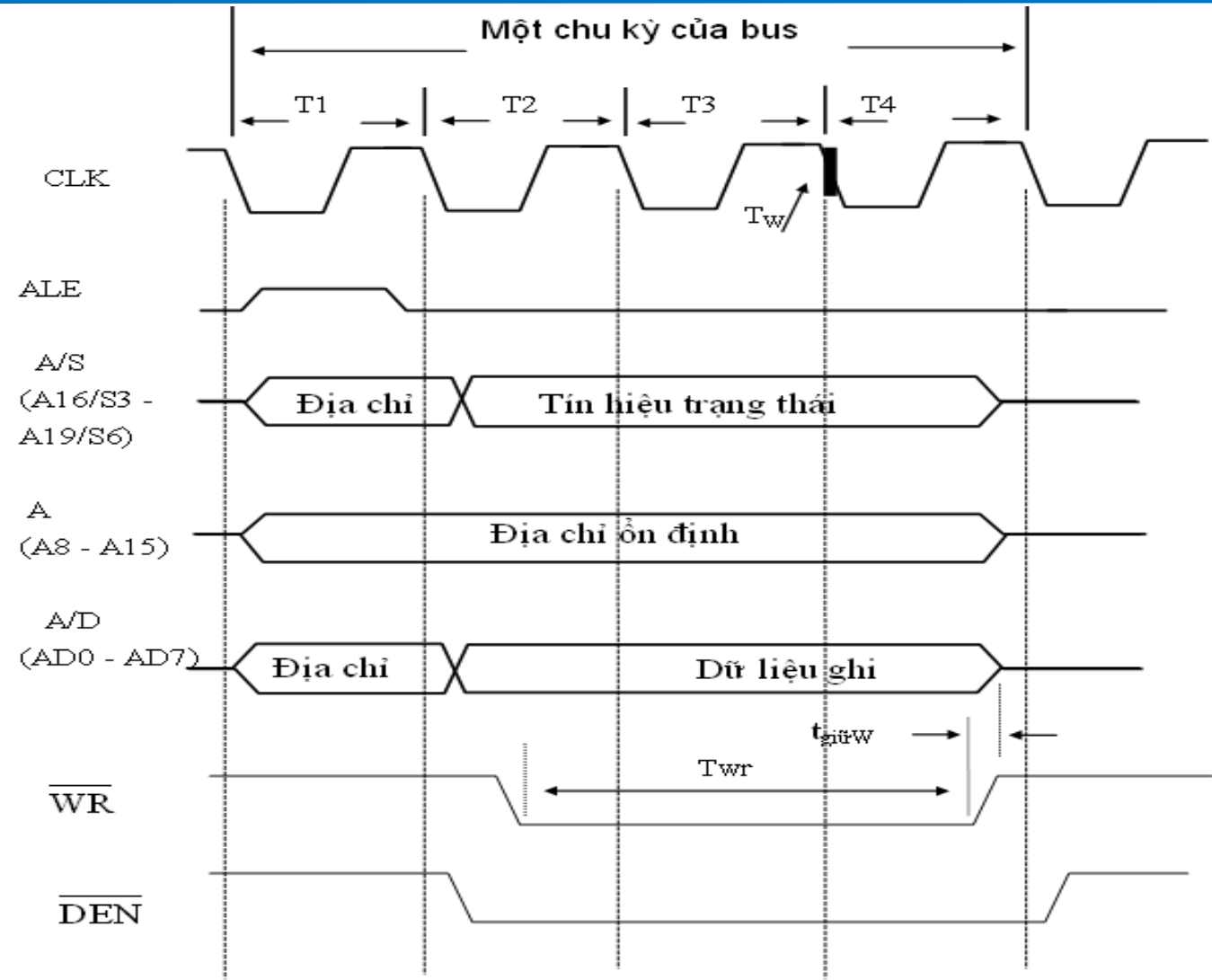
- ❖ MRDC [O]: tín hiệu điều khiển đọc bộ nhớ. Nó kích hoạt bộ nhớ đưa dữ liệu ra bus.
- ❖ MWTC [O], AMWC [O]: là các tín hiệu điều khiển ghi bộ nhớ hoặc ghi bộ nhớ kéo dài.
 - Đó thực chất là các tín hiệu giống như MEMW, nhưng AMWC (advanced memory write command) hoạt động sớm lên một chút để tạo ra khả năng cho các bộ nhớ chậm có được thêm thời gian ghi.
- ❖ IORC [O]: tín hiệu điều khiển đọc thiết bị ngoại vi. Nó kích hoạt các thiết bị được chọn để các thiết bị này đưa dữ liệu ra bus.
- ❖ IOWC [O], AIOWC [O]: là các tín hiệu điều khiển đọc thiết bị ngoại vi hoặc đọc thiết bị ngoại vi kéo dài.
 - Đó thực chất là các tín hiệu giống như IOW, nhưng AIOWC (advanced I/O write command) thì hoạt động sớm lên một chút để cho các thiết bị ngoại vi chậm được kéo dài thêm thời gian ghi.
- ❖ INTA [O]: là đầu ra để thông báo là CPU chấp nhận yêu cầu ngắt của thiết bị ngoại vi và lúc này các thiết bị ngoại vi sẽ phải đưa số hiệu ngắt ra bus để CPU đọc.
- ❖ DT/R [O]: là tín hiệu để điều khiển hướng đi của dữ liệu trong hệ vào hay ra so với CPU (DT/R = 0: CPU đọc dữ liệu, DT/R = 1 CPU ghi dữ liệu).

4.1. Các vi mạch phụ trợ cho 8088

- ❖ Trong các máy IBM PC thì tín hiệu này được nối đến các chân DIR của mạch đệm 2 chiều 74LS245 để điều khiển dữ liệu đi từ CPU đến bus hệ thống khi ghi hoặc ngược lại, từ bus hệ thống đến CPU khi đọc.
- ❖ DEN [O]: đây là tín hiệu để điều khiển bus dữ liệu trở thành bus cục bộ hay bus hệ thống.
- ❖ Trong các máy IBM PC thì tín hiệu này được sử dụng cùng với tín hiệu của mạch điều khiển ngắt PIC 8259 để tạo ra tín hiệu điều khiển cực G của mạch đệm 2 chiều 74LS245.
- ❖ MCE/PDEN [O]: đây là tín hiệu dùng định chế độ làm việc cho mạch điều khiển ngắt PIC 8259 để nó làm việc ở chế độ chủ.
- ❖ ALE [O]: đây là tín hiệu cho phép chốt địa chỉ có tại các chân dồn kênh địa chỉ - dữ liệu AD0-AD7, tín hiệu này thường được nối với chân G của mạch 74LS373 để điều khiển mạch này chốt lấy địa chỉ.

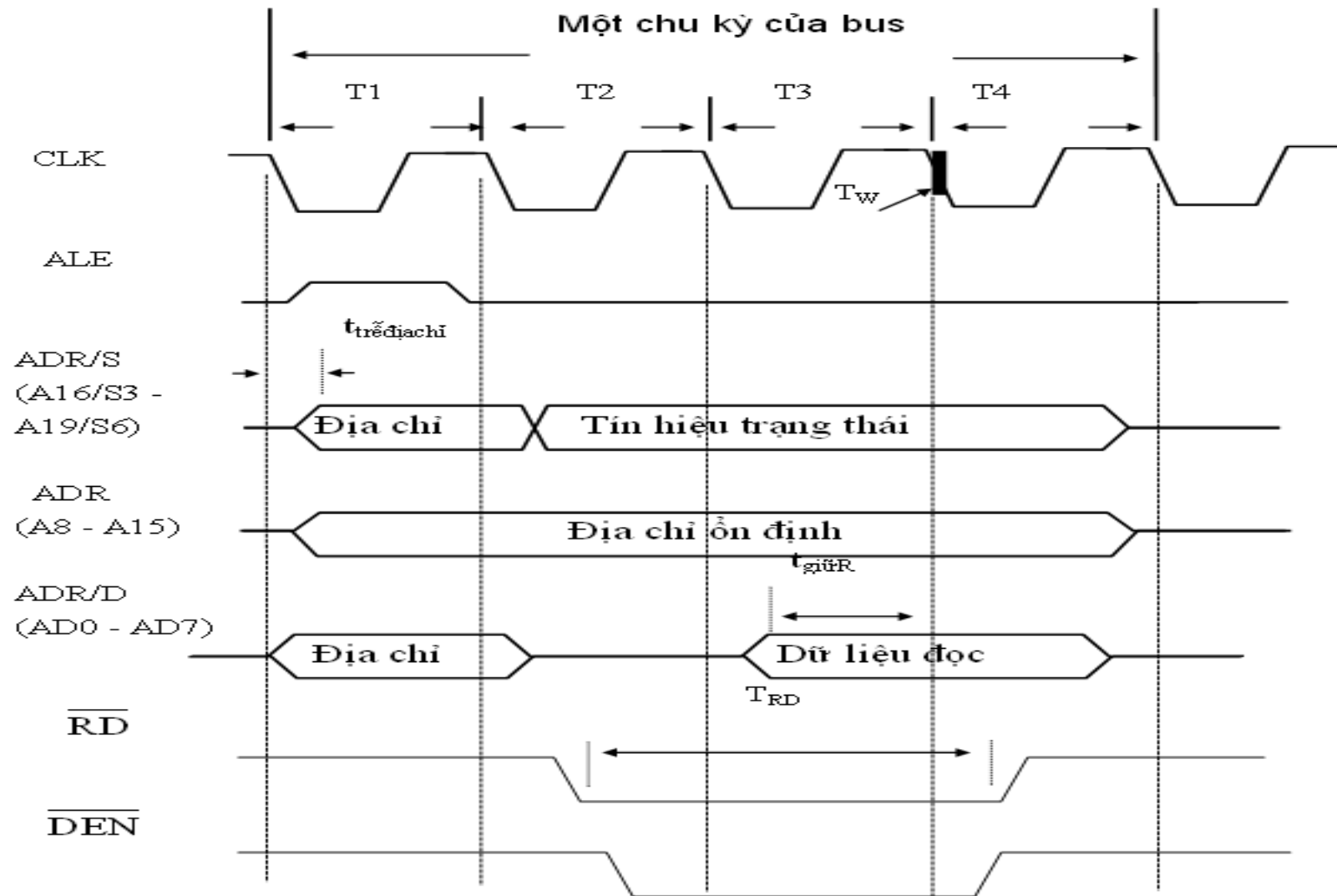
4.1. Các vi mạch phụ trợ cho 8088

❖ Biểu đồ thời gian của các lệnh ghi/đọc



Hình 5.8. Các tín hiệu của CPU 8088 trong chu kỳ ghi đơn giản hoá.

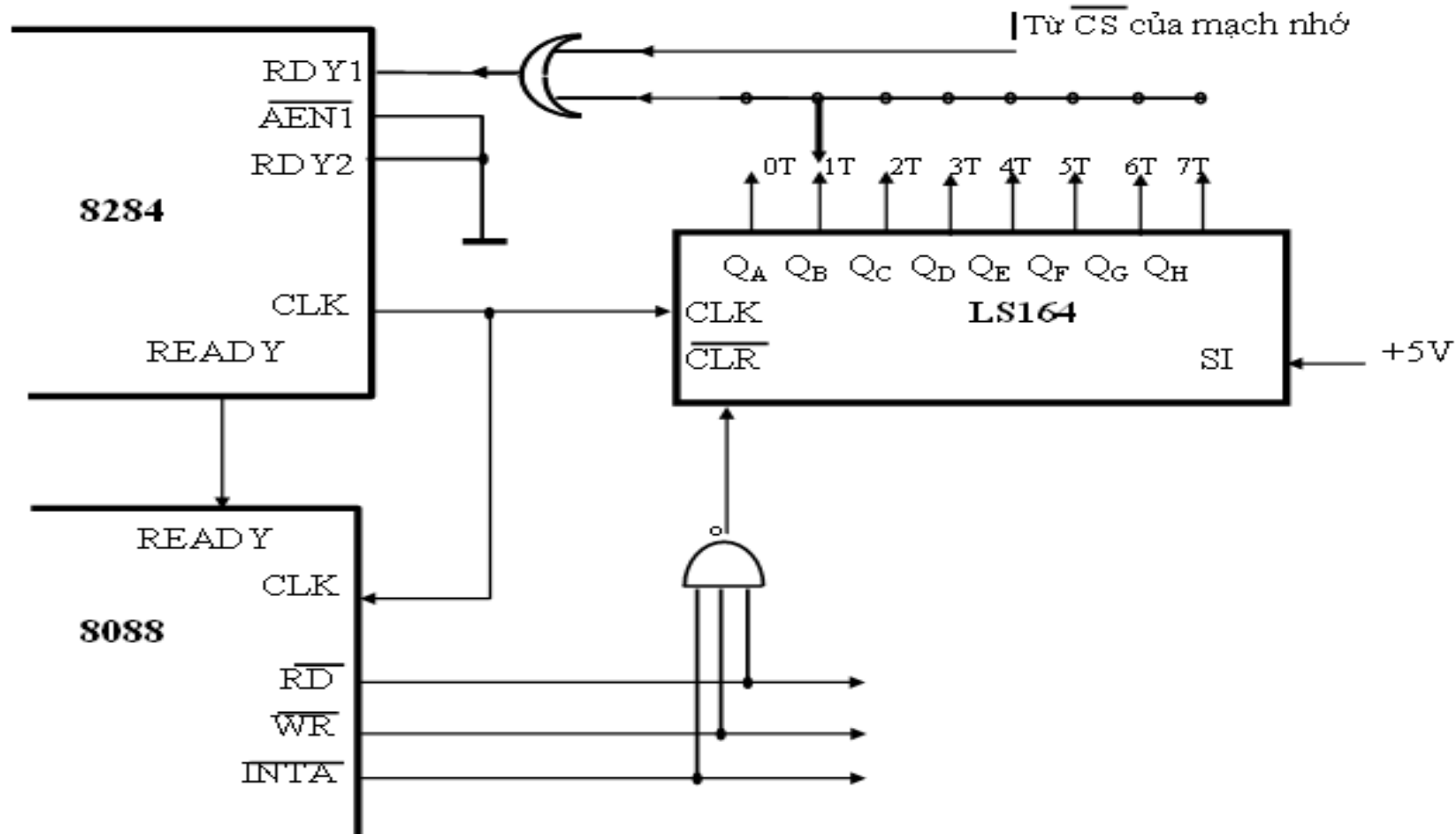
4.1. Các vi mạch phụ trợ cho 8088



Hình 5.9. Các tín hiệu của CPU 8088 trong chu kỳ đọc đơn giản hoá.

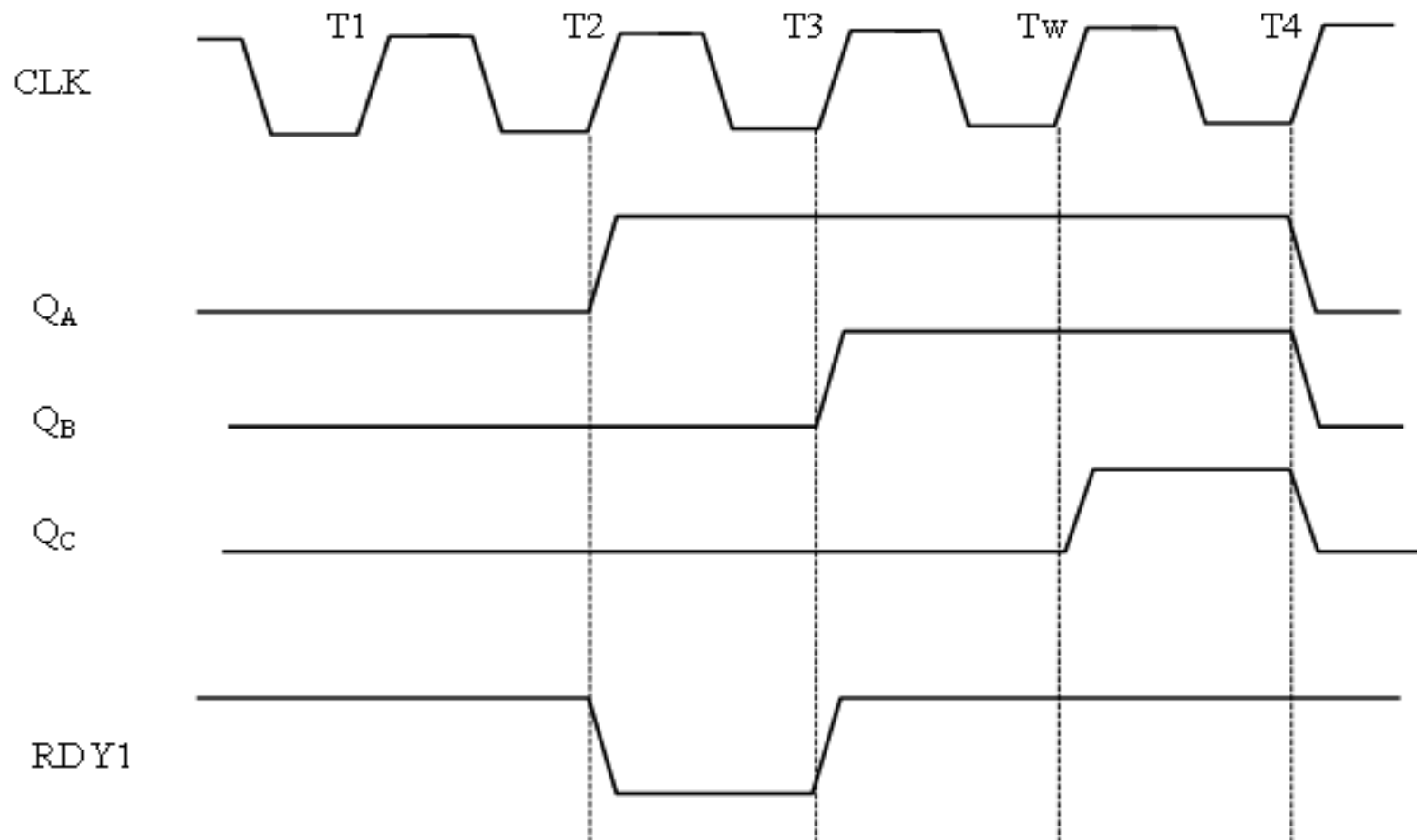
4.1. Các vi mạch phụ trợ cho 8088

❖ Mạch tạo 0-7 trạng thái chờ (đang để là 1)



4.1. Các vi mạch phụ trợ cho 8088

❖ và biểu đồ thời gian



NỘI DUNG

4.1. Các vi mạch phụ trợ cho 8088

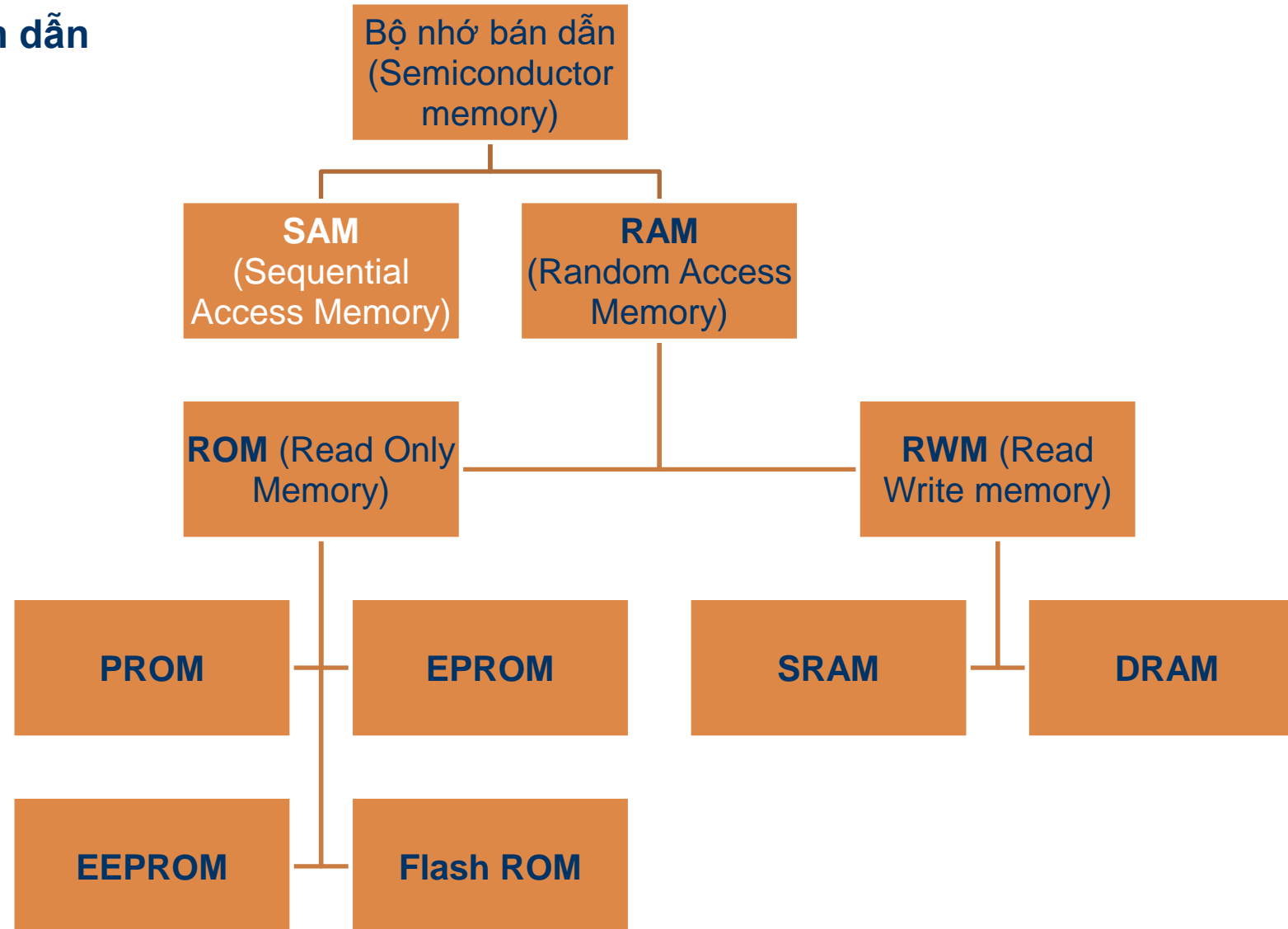


4.2. Phối ghép 8088 với bộ nhớ

4.3. Phối ghép 8088 với hệ thống vào ra

4.2. Phối ghép 8088 với bộ nhớ

Phân loại bộ nhớ bán dẫn



RAM (Random Access Memory)

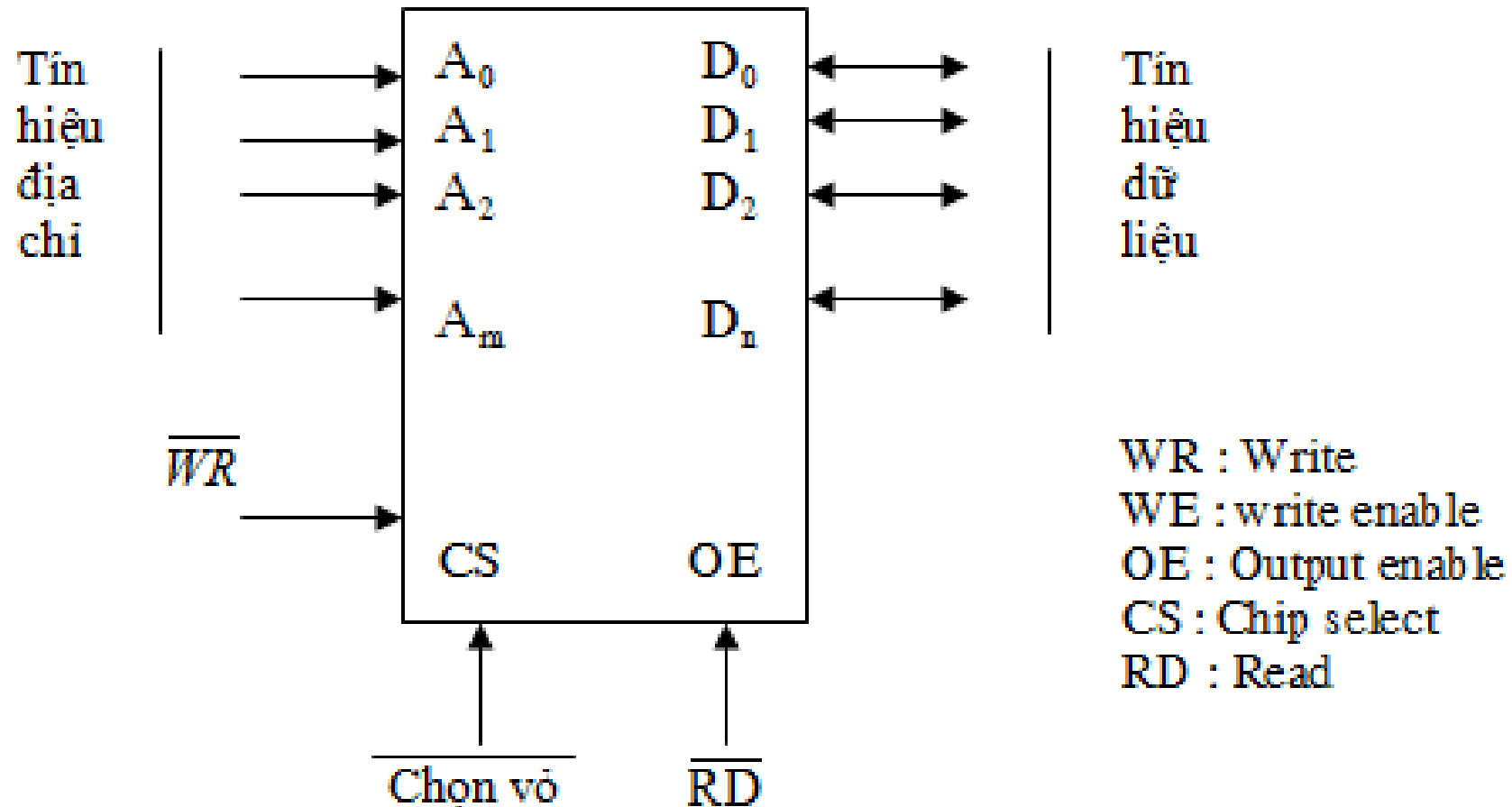
- ❖ Còn gọi là mạch nhớ truy cập ngẫu nhiên
- ❖ Là bộ nhớ chính trong máy dùng để chứa tạm thời các mã lệnh hay dữ liệu để CPU có thể truy cập nhanh chóng.
- ❖ Nội dung nhớ của RAM là không cố định có nghĩa là phải có nguồn nuôi để duy trì nội dung nhớ đó, thông tin ghi trong mạch bị mất khi mất nguồn điện nuôi cho mạch.
- ❖ RAM có hai loại chính:
 - **SRAM (Static Random Access Memory):**
 - Dữ liệu trong SRAM này luôn tồn tại trong khi được cấp nguồn
 - Gọi là SRAM (RAM tĩnh) vì không cần phải thực hiện bất kỳ thao tác nào trong khi lưu dữ liệu. Thực tế đó là các mạch Flip-Flop trong tình trạng Set hoặc Reset do đó luôn giữ trạng thái đến thao tác ghi tiếp theo hoặc mất điện.
 - Nhược điểm của SRAM này là dung lượng không lớn cỡ 32Kx8, tốn điện và đắt tiền.
 - **DRAM (Dynamic Random Access Memory) RAM động.**
 - DRAM có dung lượng lớn hơn tới 16Mx1.
 - DRAM chỉ có khả năng lưu trữ dữ liệu trong khoảng từ 2 đến 4ms (vì thực tế RAM động lưu giữ các bit dưới dạng điện tích chứa trong các tụ điện cực nhỏ). Sau 2-4ms này nội dung của DRAM phải được ghi lại (refresh).
 - DRAM yêu cầu quá nhiều chân địa chỉ nên các nhà sản xuất phải phân đường tín hiệu địa chỉ vào.

ROM (Read Only Memory)

- ❖ Là bộ nhớ chỉ có thể đọc thông tin ra.
- ❖ Thực chất ROM là tổ hợp các mạch điện thể hiện các trạng thái 1 hay 0.
- ❖ Thông tin trên ROM vẫn tồn tại thường xuyên ngay cả khi mất điện hoặc tắt máy.
- ❖ Bộ nhớ ROM có các loại PROM, EPROM, EEPROM, NOVROM, EAROM...:
 - ROM mặt nạ - thông tin trên loại ROM này được đưa vào khi sản xuất, không thể thay đổi được thông tin.
 - PROM (*Programmable ROM*): Người dùng chỉ nạp thông tin vào một lần bằng cách đốt các "cầu chì" (fuse) sau đó không thay đổi được nữa.
 - EPROM (*Erasable Programmable ROM*): Người dùng nạp thông tin bằng điện, thông tin có thể thay đổi được sau khi xóa sạch các thông tin bằng tia cực tím. EPROM là loại vi mạch nhớ có khả năng lập trình được.
 - EEPROM (*Electrically Erasable PROM*): Giống như EPROM nhưng nó có thể xóa thông tin được bằng điện.

Vi mạch nhớ

- ❖ Một bộ nhớ thường được tạo nên từ nhiều vi mạch nhớ.
- ❖ Một vi mạch nhớ thường có cấu trúc chung sau:



Các nhóm tín hiệu của 1 vi mạch nhớ

❖ Các chân địa chỉ

- Tất cả các vi mạch nhớ có các đầu vào địa chỉ để xác định **vị trí ô nhớ**.
- Số lượng các chân địa chỉ quyết định số ô nhớ bên trong nó.
 - Vi mạch nhớ có m bit địa chỉ thì dung lượng của mạch nhớ đó là 2^m từ nhớ.
- Ví dụ: vi mạch nhớ có 10 chân địa chỉ ($A_0 - A_9$) thì dung lượng của mạch nhớ đó là 1K ô nhớ $\rightarrow 10$ đầu vào địa chỉ xác định bất cứ một trong 1024 ô nhớ của nó.

❖ Các chân dữ liệu

- Tất cả các vi mạch nhớ đều có đầu ra hoặc vào/ra dữ liệu để đưa dữ liệu vào để lưu trữ trong bộ nhớ hoặc đọc dữ liệu từ bộ nhớ ra ngoài.
- Số các đầu ra xác định số bit lưu trữ trong mỗi ô nhớ của bộ nhớ.

Các nhóm tín hiệu của 1 vi mạch nhớ

❖ Tín hiệu chọn mạch (chọn vỏ)

- Mỗi vi mạch nhớ có một (hoặc nhiều hơn một) đầu vào để chọn ra vi mạch nhớ cụ thể để ghi/đọc → tín hiệu chọn mạch gọi là (*Chip Select*) \overline{CS} hoặc \overline{CE} (*Chip Enable*).
 - Vi mạch nhớ RAM thông thường có các đầu vào \overline{CS}
 - Vi mạch nhớ ROM có ít nhất một đầu vào \overline{CE}
- Các tín hiệu chọn vỏ thường được nối với đầu ra của bộ giải mã địa chỉ.
 - Khi một mạch nhớ không được chọn thì bus dữ liệu của nó bị treo (ở trạng thái trở kháng cao).

Các nhóm tín hiệu của 1 vi mạch nhớ

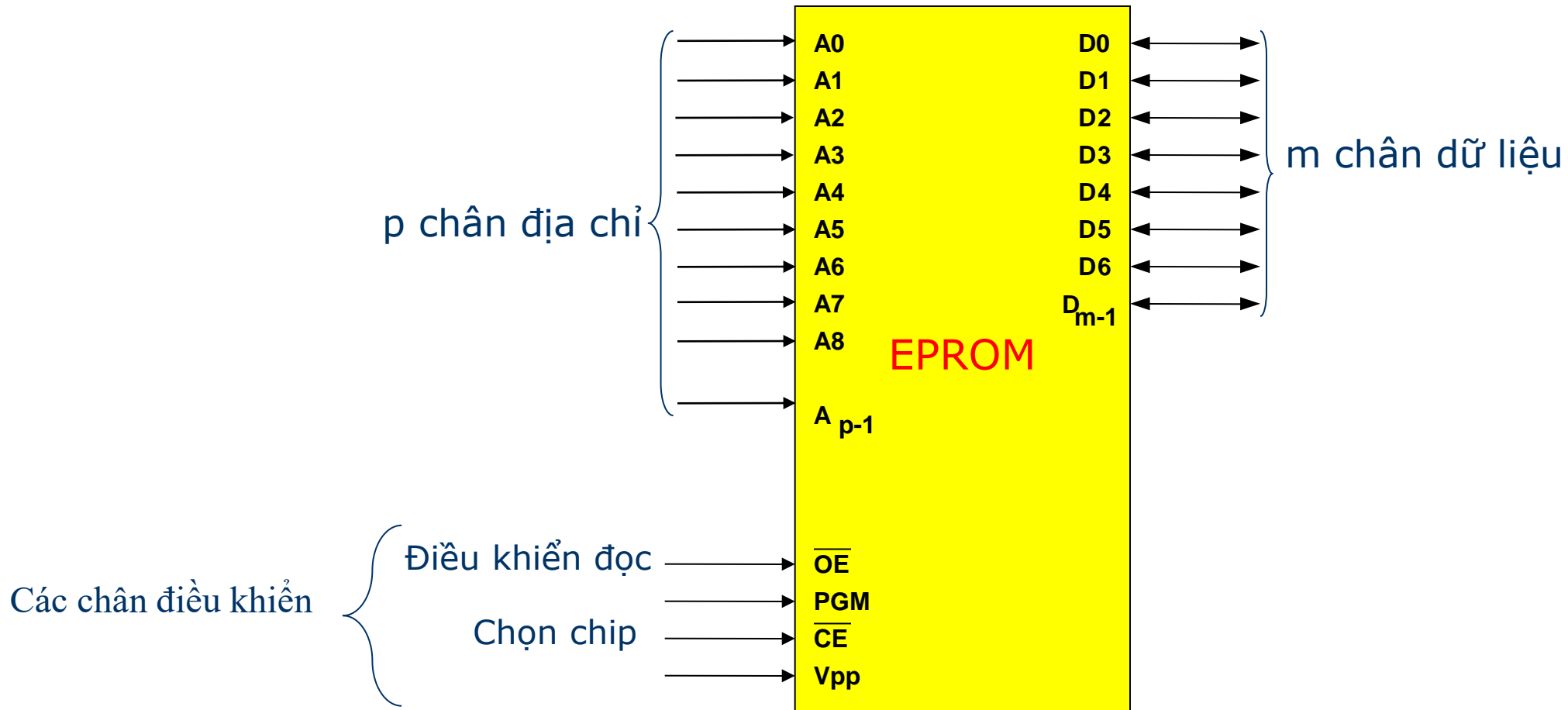
❖ Tín hiệu điều khiển:

- Cần có trong tất cả các mạch nhớ
- ROM:
 - Thường chỉ có 1 chân điều khiển là \overline{OE} (*Output Enable*) hoặc gate selection (G) để cho phép dữ liệu được đưa ra bus.
 - Một mạch nhớ không được mở bởi \overline{OE} thì bus dữ liệu của nó bị treo.
- Một mạch nhớ RAM:
 - Nếu chỉ có 1 tín hiệu điều khiển thì thường đó là \overline{WE} , tín hiệu này điều khiển quá trình đọc/ghi chỉ khi tín hiệu chọn mạch được kích hoạt.
 - Nếu mạch nhớ có 2 tín hiệu điều khiển thì, thường là R/\overline{W} (Write Enable) điều khiển ghi và \overline{OE} điều khiển đọc (hoặc G).
 - Khi có cả hai chân này thì chúng không bao giờ cùng hoạt động (hai tín hiệu này phải ngược pha nhau để điều khiển việc đọc/ghi mạch nhớ).
- Khi không hoạt động các chân này sẽ được đặt ở trạng thái trở kháng cao.

Thời gian thâm nhập t_{ac}

- ❖ Là 1 thông số đặc trưng khác của bộ nhớ.
- ❖ Là thời gian kể từ khi có xung địa chỉ trên bus địa chỉ cho đến khi có dữ liệu ra ổn định trên bus dữ liệu.
- ❖ t_{ac} phụ thuộc rất nhiều vào công nghệ chế tạo
 - Các bộ nhớ làm bằng công nghệ lưỡng cực có thời gian nhỏ (10-30ns)
 - Các bộ nhớ làm bằng công nghệ MOS có thời gian thâm nhập lớn hơn nhiều (cỡ 150 ns hoặc hơn nữa).

Các chip EPROM



PGM: xung lập trình

V_{pp} : điện áp ghi

Dung lượng của 1 chip nhớ

- ❖ Một chip nhớ được xem như một mảng gồm n ô nhớ.
- ❖ Mỗi ô nhớ lưu trữ được m -bit dữ liệu
- ❖ Dung lượng của chip thường được biểu diễn: $n \times m$
- ❖ Ví dụ: Một chip có dung lượng $2K \times 8$ nghĩa là:
 - chip đó có 2048 ô nhớ
 - Mỗi ô nhớ có thể lưu trữ được 1 byte dữ liệu
- ❖ m : số chân dữ liệu của chip
- ❖ $\log_2(n) = p$ là số chân địa chỉ của chip

Hoạt động ghi dữ liệu vào EPROM

- ❖ Việc ghi dữ liệu vào EPROM được gọi là lập trình cho EPROM
- ❖ Được thực hiện bằng thiết bị chuyên dụng (Bộ nạp EPROM)
 - Dữ liệu muốn ghi sẽ được đưa lên các chân dữ liệu
 - Địa chỉ của ô nhớ muốn ghi dữ liệu vào đưa lên các chân địa chỉ
 - Chân Vpp: được cấp điện áp tương ứng với từng loại chip gọi là điện áp lập trình.
 - Dữ liệu tại các chân dữ liệu sẽ được ghi vào một ô nhớ xác định nhờ các tín hiệu đưa vào ở các chân địa chỉ và một xung (thường gọi là xung lập trình) đưa vào chân PGM

Hoạt động đọc dữ liệu từ một chip EPROM

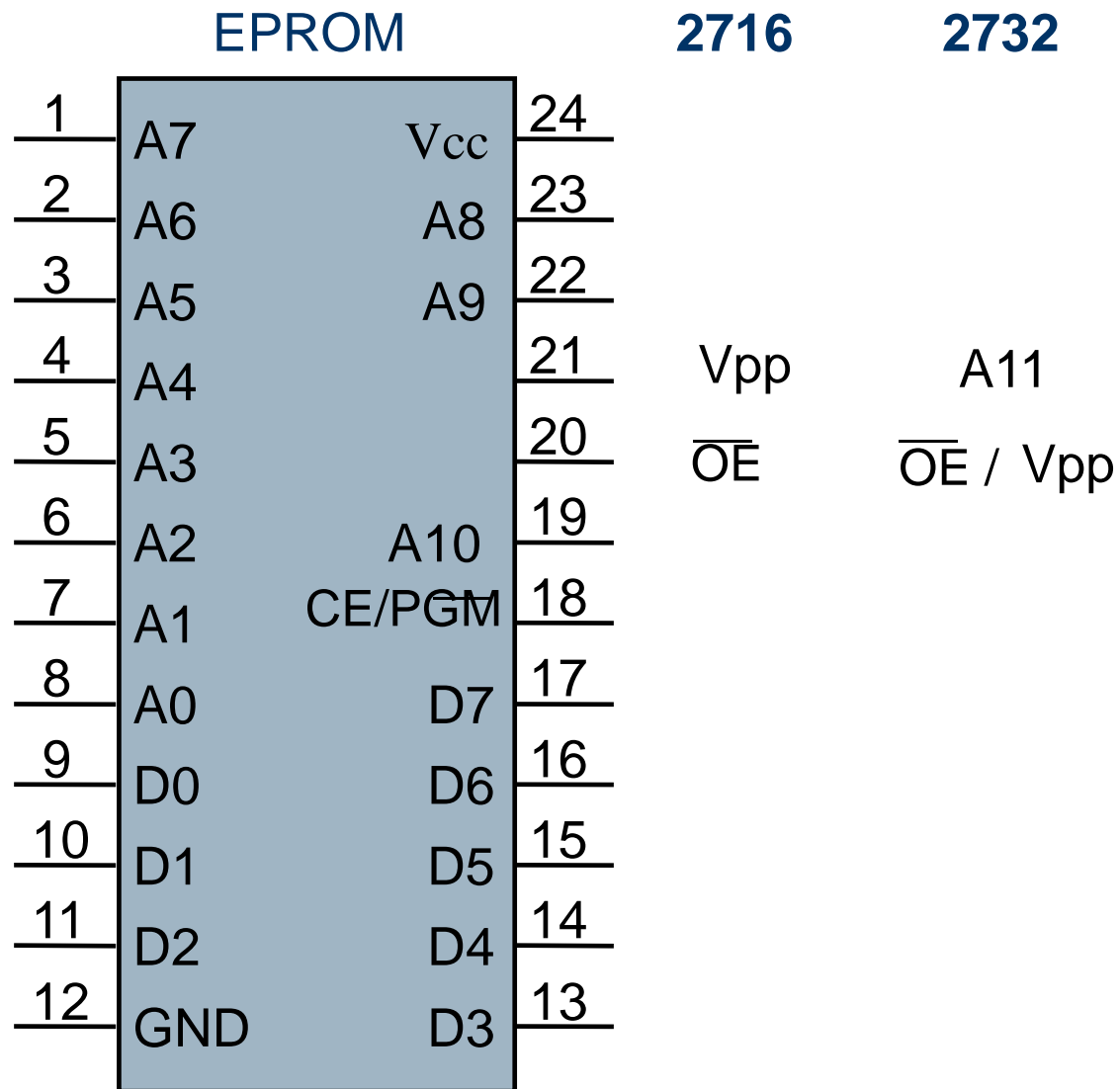
- ❖ Để đọc dữ liệu từ 1 ô nhớ của 1 chip EPROM, bộ vi xử lý cần phải:
 - Chọn chip đó: $0 \rightarrow \text{CE}$
 - Áp các tín hiệu địa chỉ của ô nhớ cần đọc vào các chân địa chỉ $A_{p-1} - A_0$
 - Đọc: $0 \rightarrow \text{OE}$
- ❖ Kết quả: m bit dữ liệu cần đọc xuất hiện ở các chân dữ liệu $D_{m-1} - D_0$

Họ EPROM thông dụng 27x

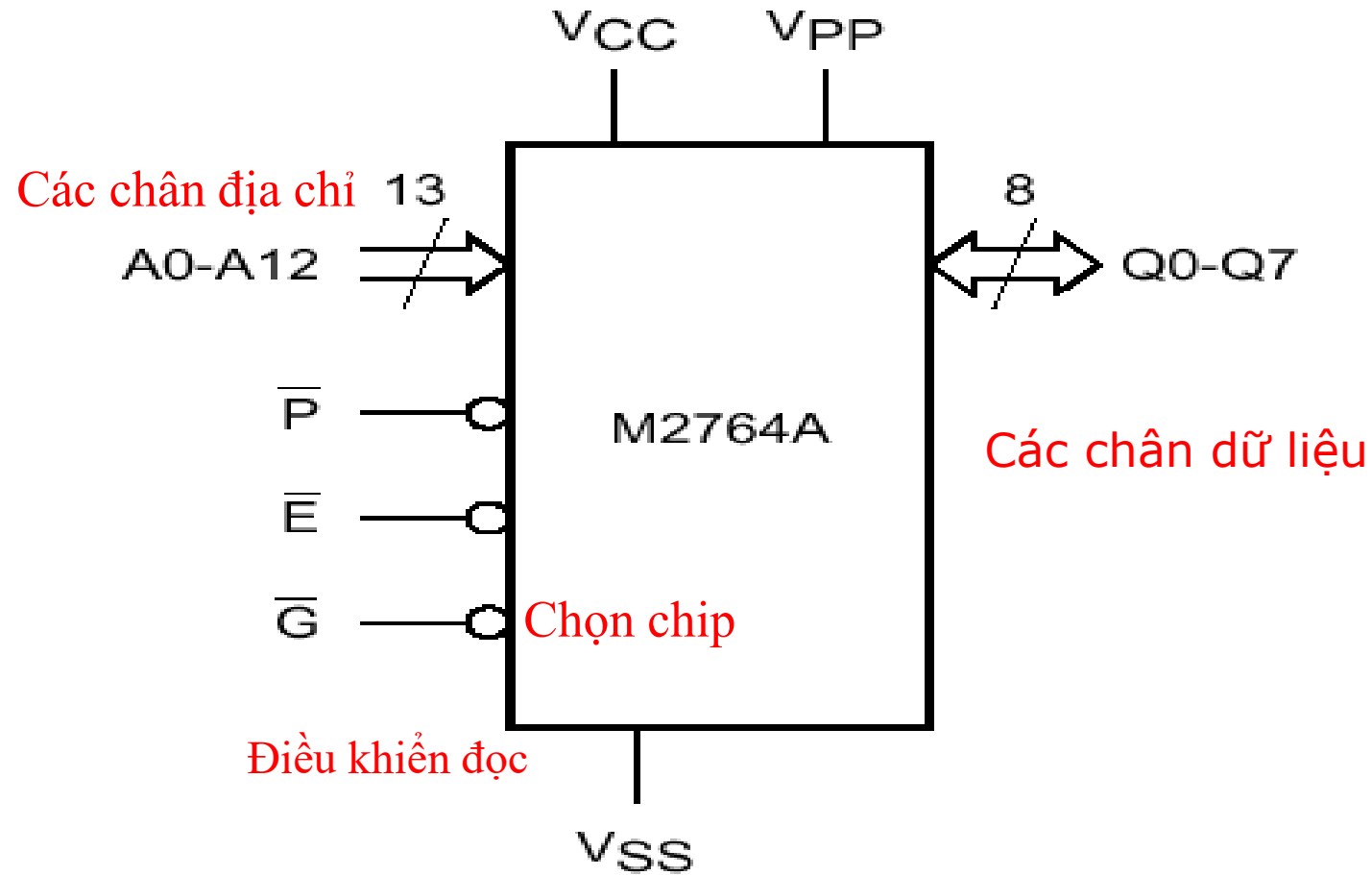
Số hiệu của chip	Dung lượng
2716	2Kx8
2732	4Kx8
2764	8Kx8
27128	16Kx8
27256	32Kx8
27512	64Kx8

Họ EPROM 27x

Sơ đồ chân của 2716 và 2732



EPROM 2764



AI00776B

EPROM 2764

Mode	\overline{E}	\overline{G}	\overline{P}	A9	V _{PP}	Q0 - Q7
Read	V _{IL}	V _{IL}	V _{IH}	X	V _{CC}	Data Out
Output Disable	V _{IL}	V _{IH}	V _{IH}	X	V _{CC}	Hi-Z
Program	V _{IL}	V _{IH}	V _{IL} Pulse	X	V _{PP}	Data In
Verify	V _{IL}	V _{IL}	V _{IH}	X	V _{PP}	Data Out
Program Inhibit	V _{IH}	X	X	X	V _{PP}	Hi-Z
Standby	V _{IH}	X	X	X	V _{CC}	Hi-Z
Electronic Signature	V _{IL}	V _{IL}	V _{IH}	V _{ID}	V _{CC}	Codes Out

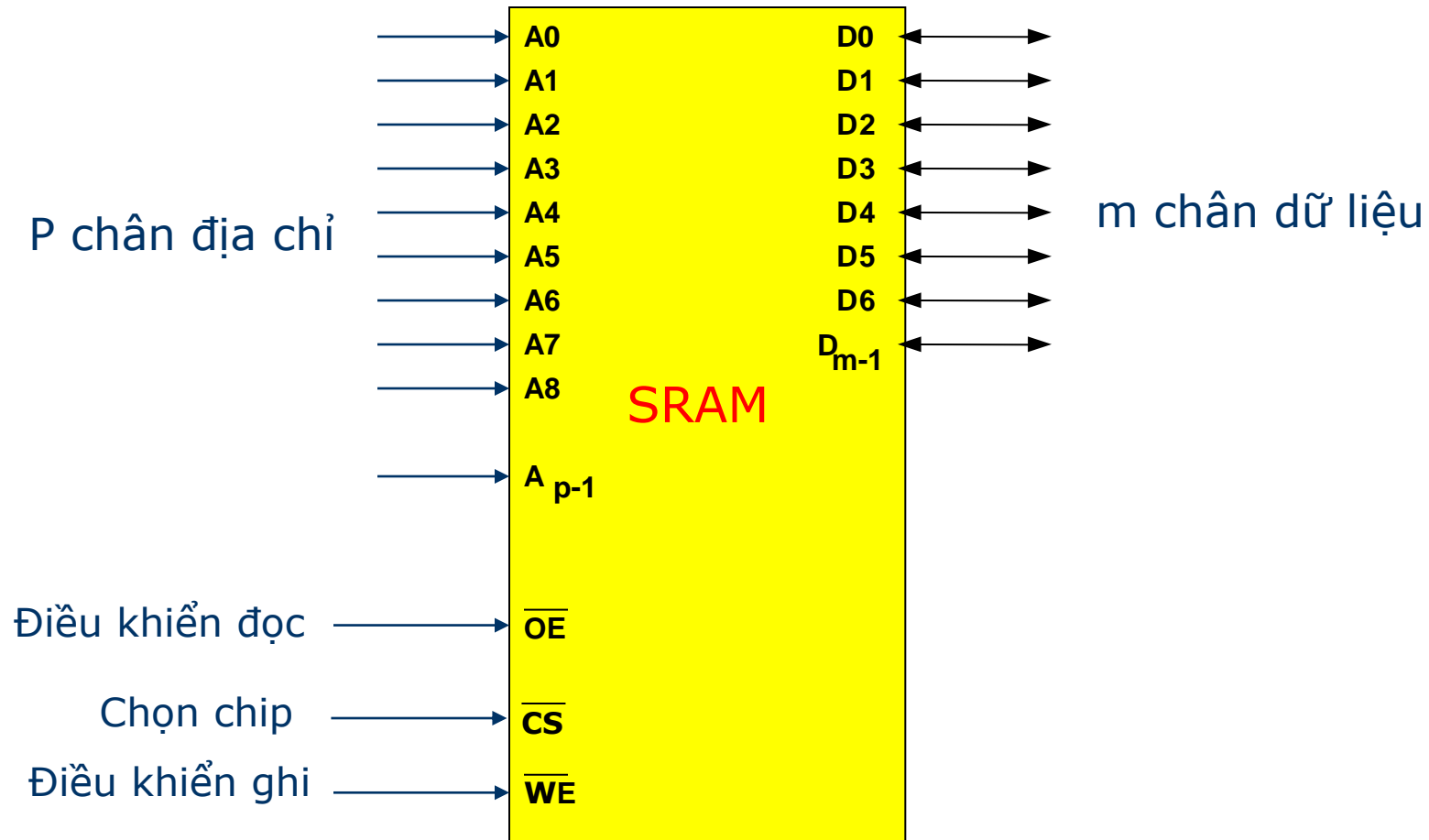
Note: X = V_{IH} or V_{IL}, V_{ID} = 12V ± 0.5%.

Lập trình cho 2764

- ❖ Trước hết cần phải xoá
 - Xoá một chip tức là làm cho tất cả các bit = 1
- ❖ Xoá một chip EPROM bằng tia cực tím
- ❖ Lập trình bằng cách:
 - VPP mắc ở mức 12.5V
 - E và P đều ở mức thấp TTL
- ❖ Các bit dữ liệu đưa vào các chân dữ liệu
- ❖ Các bit địa chỉ đưa vào các chân địa chỉ

Các chip SRAM

❖ Sơ đồ khối:



Đọc dữ liệu từ một chip SRAM

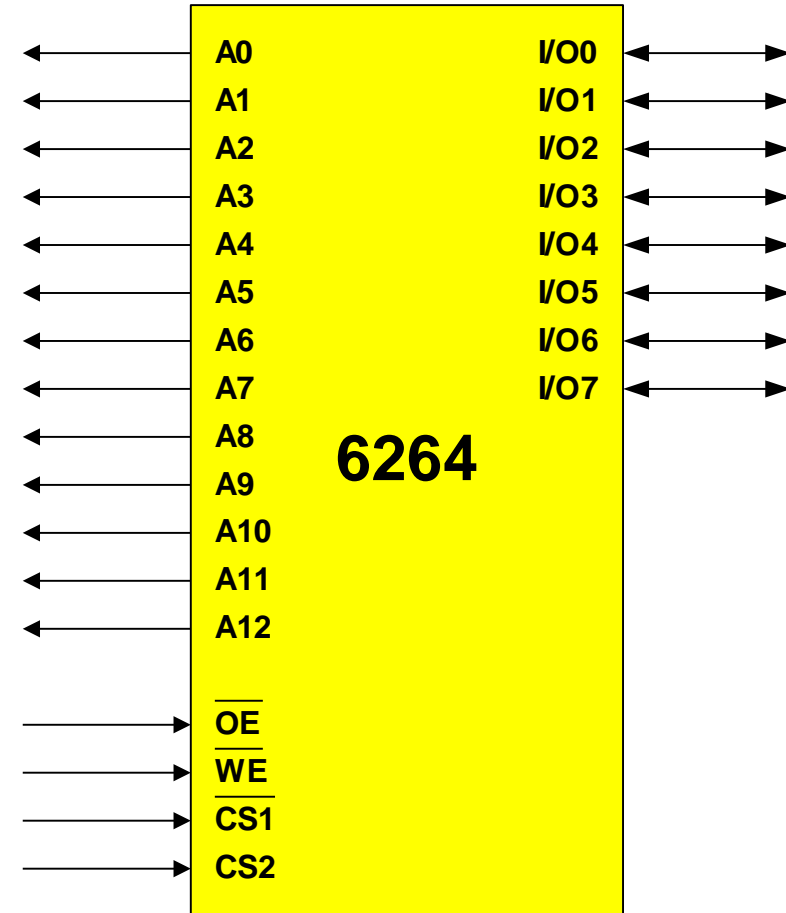
- ❖ Để đọc dữ liệu từ 1 ô nhớ của 1 chip SRAM, vi xử lý cần phải:
 - Chọn chip đó: $0 \rightarrow CS$
 - Áp các tín hiệu địa chỉ vào $A_{p-1} - A_0$
 - Đọc: $0 \rightarrow OE$
- ❖ Kết quả là m bit dữ liệu cần đọc xuất hiện ở các chân dữ liệu $D_{m-1} - D_0$

Ghi dữ liệu vào một chip SRAM

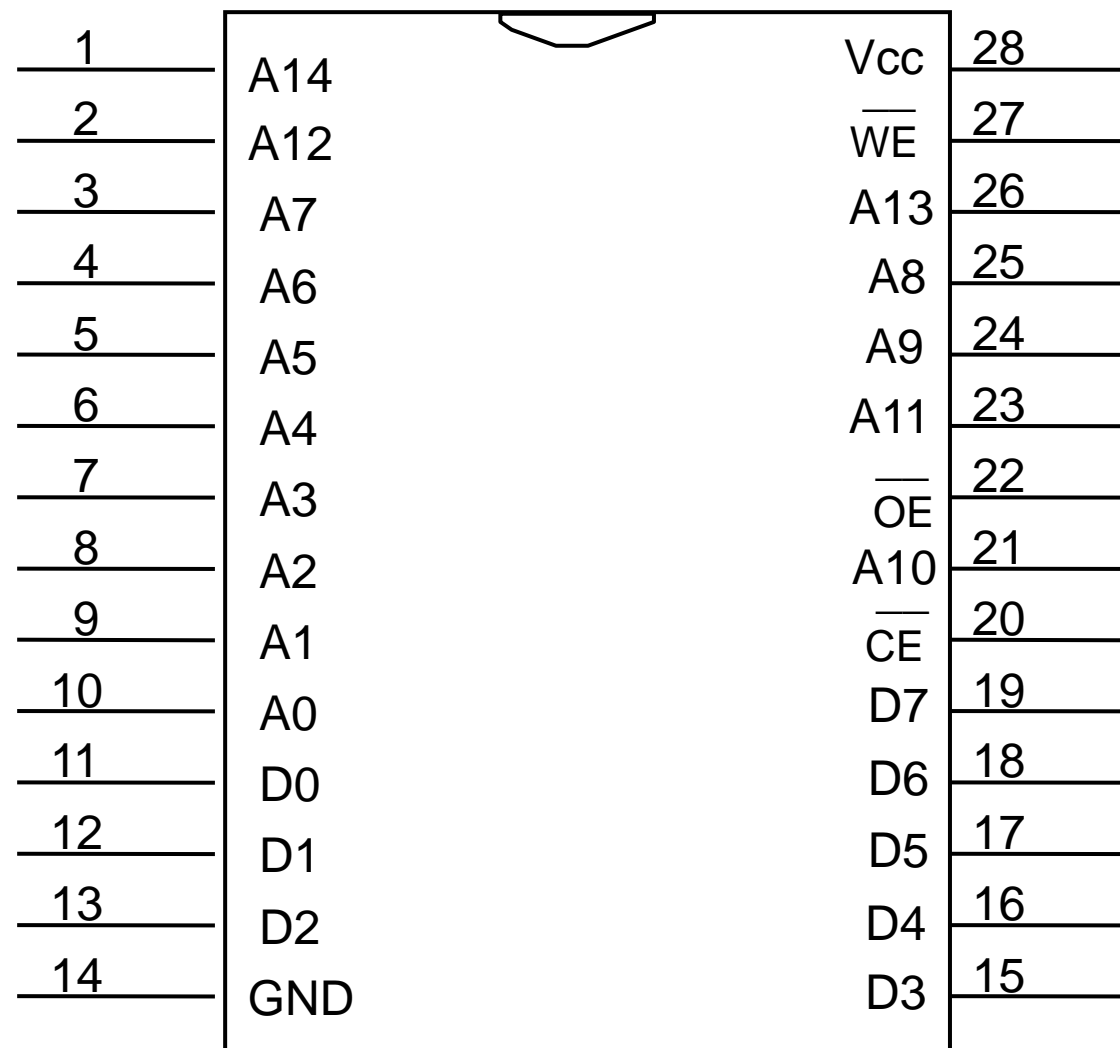
- ❖ Để ghi m bit dữ liệu vào 1 ô nhớ của 1 chip SRAM, vi xử lý cần phải:
 - Chọn chip đó: $0 \rightarrow CS$
 - Áp các tín hiệu địa chỉ vào $A_{p-1} - A_0$
 - Áp m bit dữ liệu cần ghi vào các chân dữ liệu $D_{m-1} - D_0$
 - Ghi: $0 \rightarrow WE$
- ❖ Kết quả là các bit dữ liệu ở các chân dữ liệu sẽ được ghi vào ô nhớ đã chọn

SRAM 6264

- ❖ Dung lượng 8Kx8
 - 8 chân dữ liệu
 - 13 chân địa chỉ
- ❖ Hai chân chọn chip
- ❖ Chân điều khiển đọc
- ❖ Chân điều khiển ghi

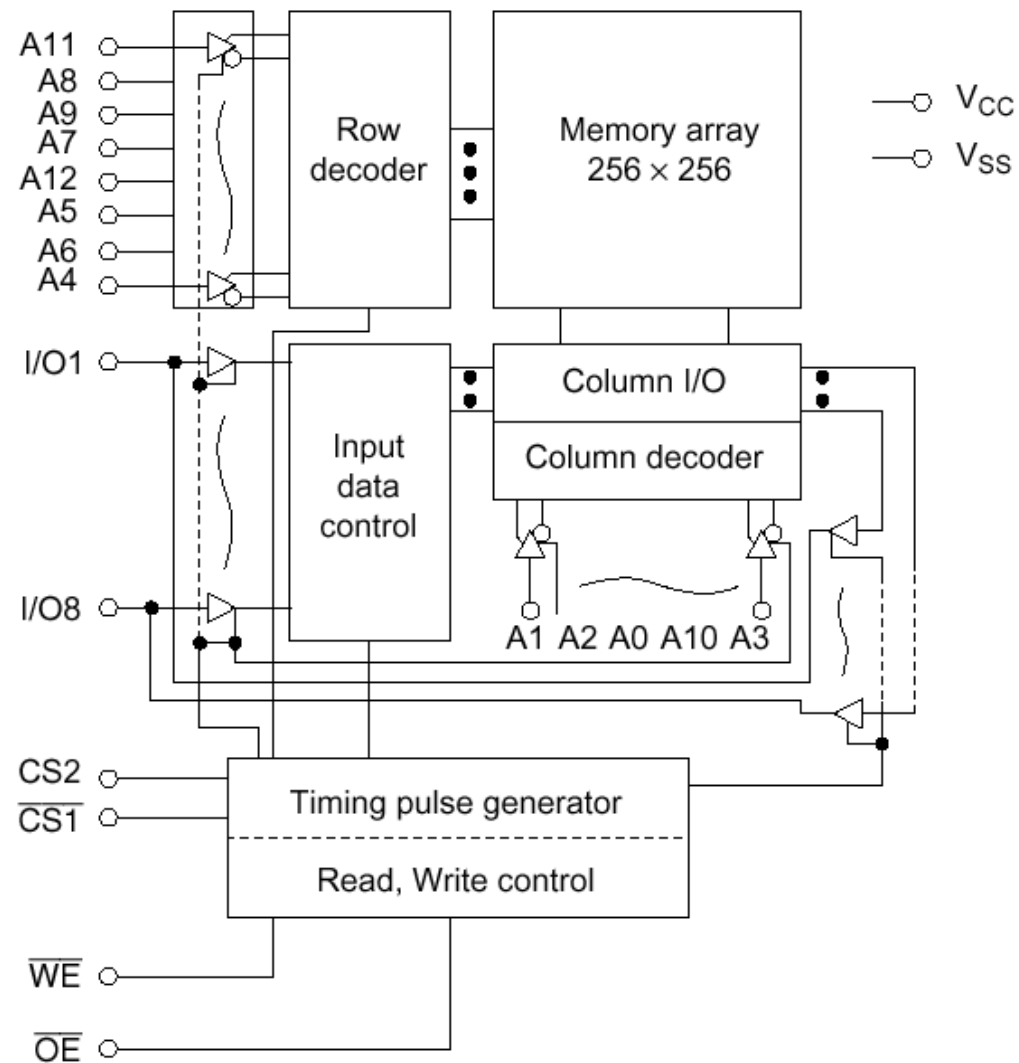


32K x 8 Static RAM



51256S

Sơ đồ khối 6264



Chức năng của 6264

\overline{WE}	$\overline{CS1}$	CS2	\overline{OE}	Mode	V_{CC} current	I/O pin	Ref. cycle
x	H	x	x	Not selected (power down)	I_{SB}, I_{SB1}	High-Z	—
x	x	L	x	Not selected (power down)	I_{SB}, I_{SB1}	High-Z	—
H	L	H	H	Output disable	I_{CC}	High-Z	—
H	L	H	L	Read	I_{CC}	Dout	Read cycle (1)–(3)
L	L	H	H	Write	I_{CC}	Din	Write cycle (1)
L	L	H	L	Write	I_{CC}	Din	Write cycle (2)

Note: x: H or L

Phối ghép bộ nhớ cho VXL 8088

❖ Bus hệ thống của 8088:

- Bus địa chỉ **20-bit**: các đường địa chỉ từ A_{19} đến A_0
- Bus dữ liệu **8-bit**: các đường dữ liệu từ D_7 đến D_0
- Bus điều khiển:
 - Gồm các đường điều khiển riêng lẻ phục vụ cho hoạt động truy cập bộ nhớ và các cổng I/O.
 - Mỗi đường thường được ký hiệu bằng tên của tín hiệu điều khiển.
- Bus hệ thống:
 - Không nối trực tiếp với các chân của 8088 thông qua các mạch đệm, chốt.

Giải mã địa chỉ cho bộ nhớ

- ❖ Mỗi mạch nhớ nối ghép với CPU cần phải được CPU quy chiếu tới một cách chính xác khi thực hiện thao tác ghi/đọc.
 - → mỗi mạch nhớ phải được gán cho một vùng riêng biệt có địa chỉ xác định nằm trong không gian địa chỉ tổng thể của bộ nhớ.
- ❖ Việc phân định không gian địa chỉ tổng thể thành các vùng nhớ khác nhau để thực hiện những chức năng nhất định gọi là **phân vùng bộ nhớ**.
- ❖ Chú ý:
 - Làm việc với bộ nhớ $IO/\overline{M}=0$

Bản đồ địa chỉ bộ nhớ

❖ Bản đồ địa chỉ bộ nhớ:

- Là sơ đồ biểu diễn các vị trí đã sử dụng trong bộ nhớ
- Được dùng để phân biệt các vị trí, nhiệm vụ của chúng và phân chia vùng nhớ cho RAM, ROM và các cổng vào/ra.

❖ Vi mạch 8088 có 20 đường dây địa chỉ → quản lý được 2^{20} vùng nhớ khác nhau

- Bản đồ bộ nhớ hệ thống dùng 8088: 00000H → FFFFFH.

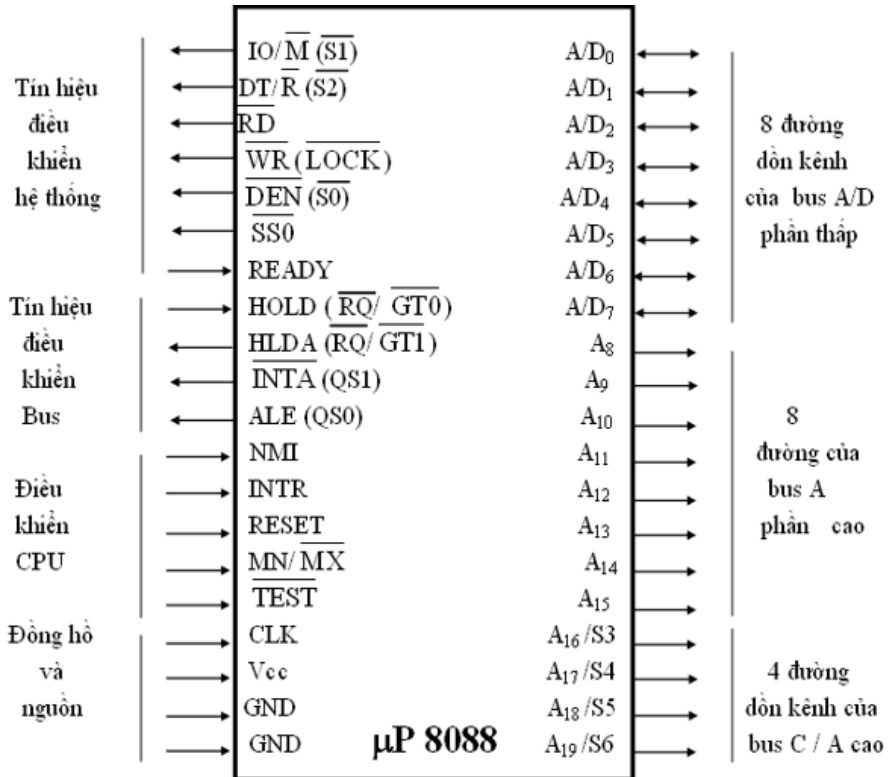
Không gian địa chỉ bộ nhớ 1M

A19 đến A0 (HEX)	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
00000	0000	0000	0000	0000	0000
FFFFFF	1111	1111	1111	1111	1111

Ví dụ: Một địa chỉ bất kỳ 34FD0h

0011 0100 11111 1101 0000

Bộ nhớ đầy đủ 1MB



AX	3F1C	A19
BX	0023	A0
CX	0000	:
DX	FCA1	:
CS	XXXX	:
SS	XXXX	:
DS	2000	:
ES	XXXX	:
BP	XXXX	D7
SP	XXXX	:
		D0
SI	XXXX	MEMR
DI	XXXX	
IP	XXXX	MEMW

	FFFFF	36
	FFFFE	25
	FFFFD	19
A19	:	:
:	:	:
A0	20023	13
	20022	7D
	20021	12
	20020	29
D7	:	:
:	:	:
D0	10008	8A
	10007	F4
	10006	07
	10005	88
	10004	42
RD	10003	39
	10002	27
	10001	98
	10000	45
	:	:
WR	:	:
	00001	95
CS	00000	23

Bộ nhớ < 1MB

Nếu chỉ cần bộ nhớ có dung lượng nhỏ hơn 1MB thì giải quyết như thế nào?

- ❖ Phụ thuộc vào các chip nhớ sẵn có
- ❖ Phụ thuộc yêu cầu phân bổ địa chỉ cho các loại bộ nhớ vật lý khác nhau
- ❖ ...

Bộ nhớ đầy đủ 1MB

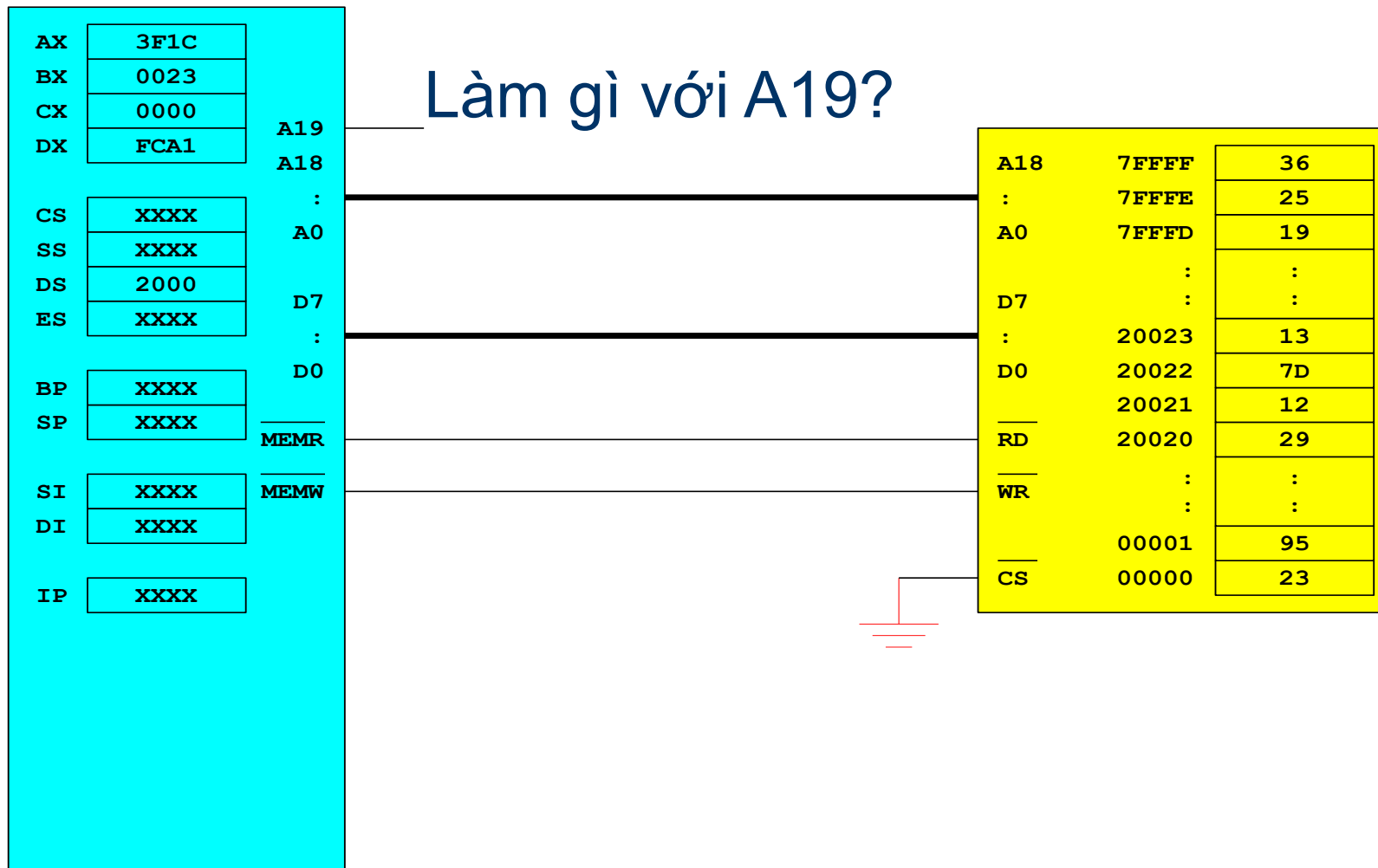
512K đầu tiên của không gian địa chỉ bộ nhớ
(Các địa chỉ có bit cao nhất $A_{19} = 0$)

A19 đến A0 (HEX)	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
00000	000	0000	0000	0000	0000
7FFFF	111	1111	1111	1111	1111

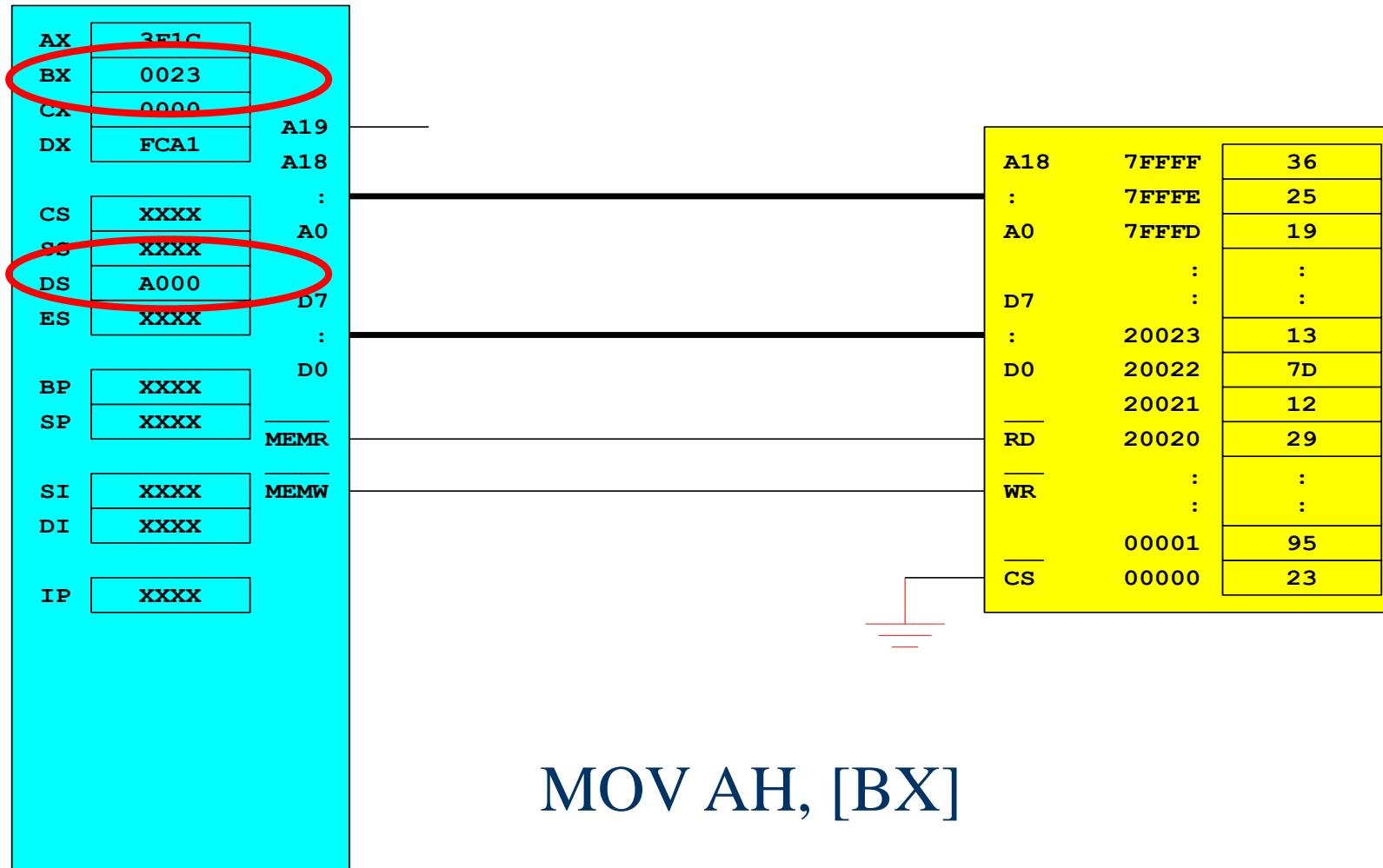
512K tiếp theo của không gian địa chỉ bộ nhớ
(Các địa chỉ có bit cao nhất $A_{19} = 1$)

A19 đến A0 (HEX)	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
80000	000	0000	0000	0000	0000
FFFFFF	111	1111	1111	1111	1111

Bộ nhớ 512KB




Điều gì xảy ra nếu 8088 đọc ô nhớ A0023h?



Điều gì xảy ra nếu 8088 đọc ô nhớ A0023h?

A19 đến A0 (HEX)	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
A0023	1010	0000	0000	0010	0011



A_{19} không được nối đến bộ nhớ nên nếu 8088 phát logic “1” trên A_{19} thì bộ nhớ cũng không nhận biết được.

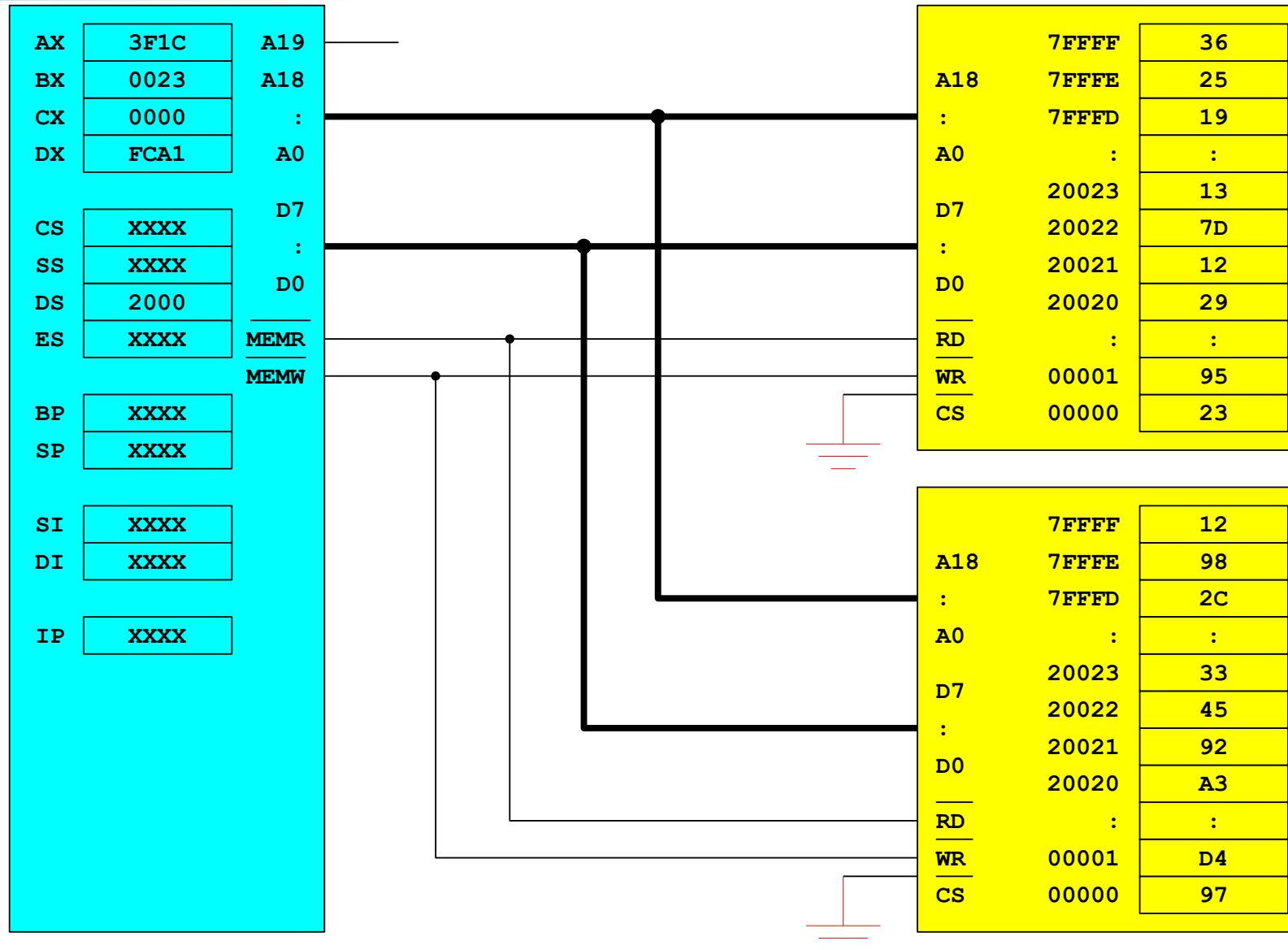
Điều gì xảy ra nếu 8088 đọc ô nhớ 20023h?

A19 đến A0 (HEX)	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
20023	0010	0000	0000	0010	0011



Với bộ nhớ tình hình không có gì khác!

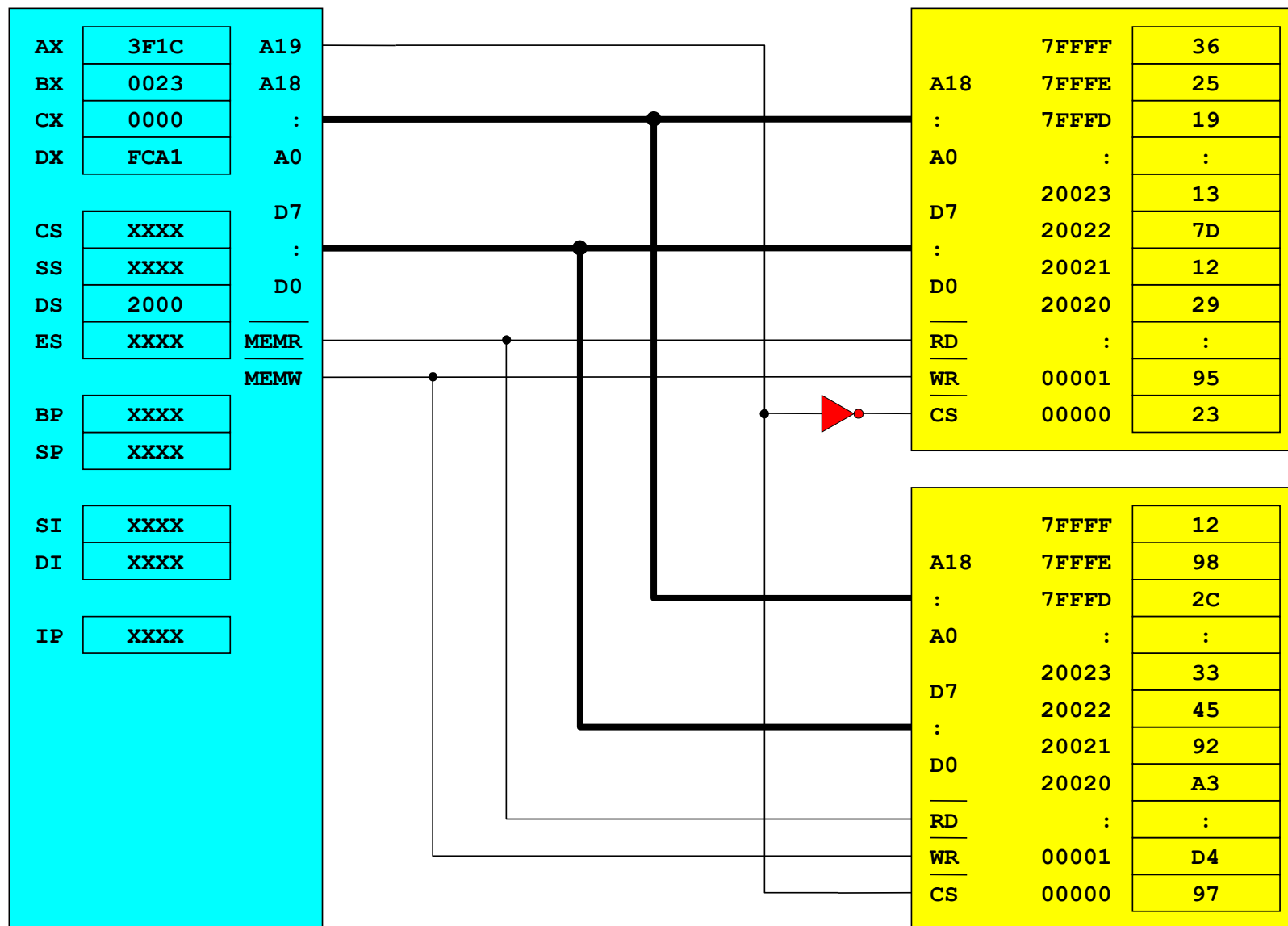
Nếu Bộ nhớ gồm 2 khối 512KB như thế này?



Có vấn đề !!!

- ❖ **Vấn đề là:** Xung đột Bus. Hai khối nhớ sẽ cung cấp dữ liệu cùng một lúc khi 8088 đọc bộ nhớ
- ❖ **Giải pháp:** Dùng A19 làm “người phân xử” để giải quyết xung đột trên bus.
 - Nếu A19 ở mức logic “1” thì khối nhớ trên hoạt động (khối nhớ dưới bị cấm) và ngược lại

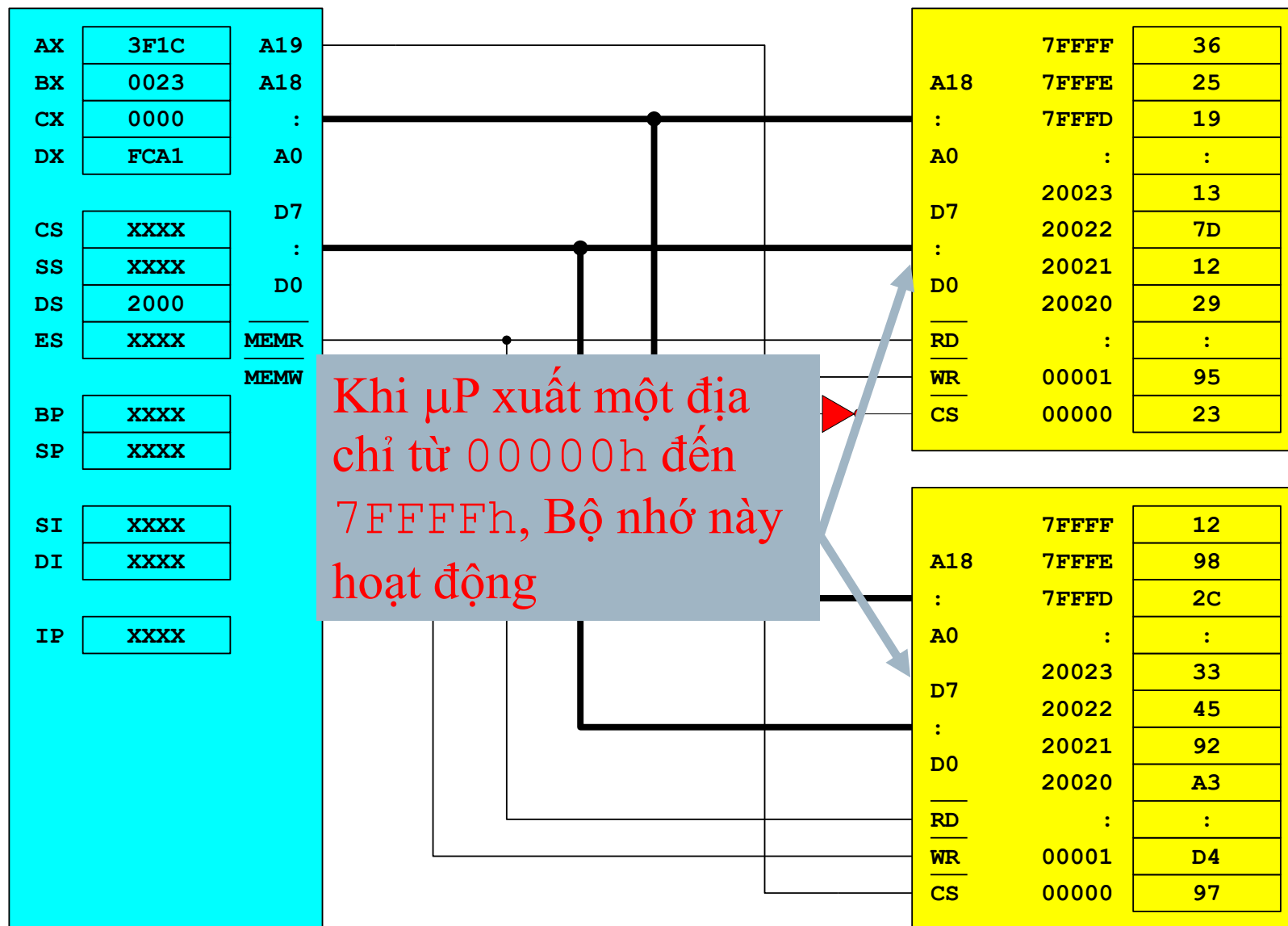
Bộ nhớ gồm hai khối nhớ 512KB



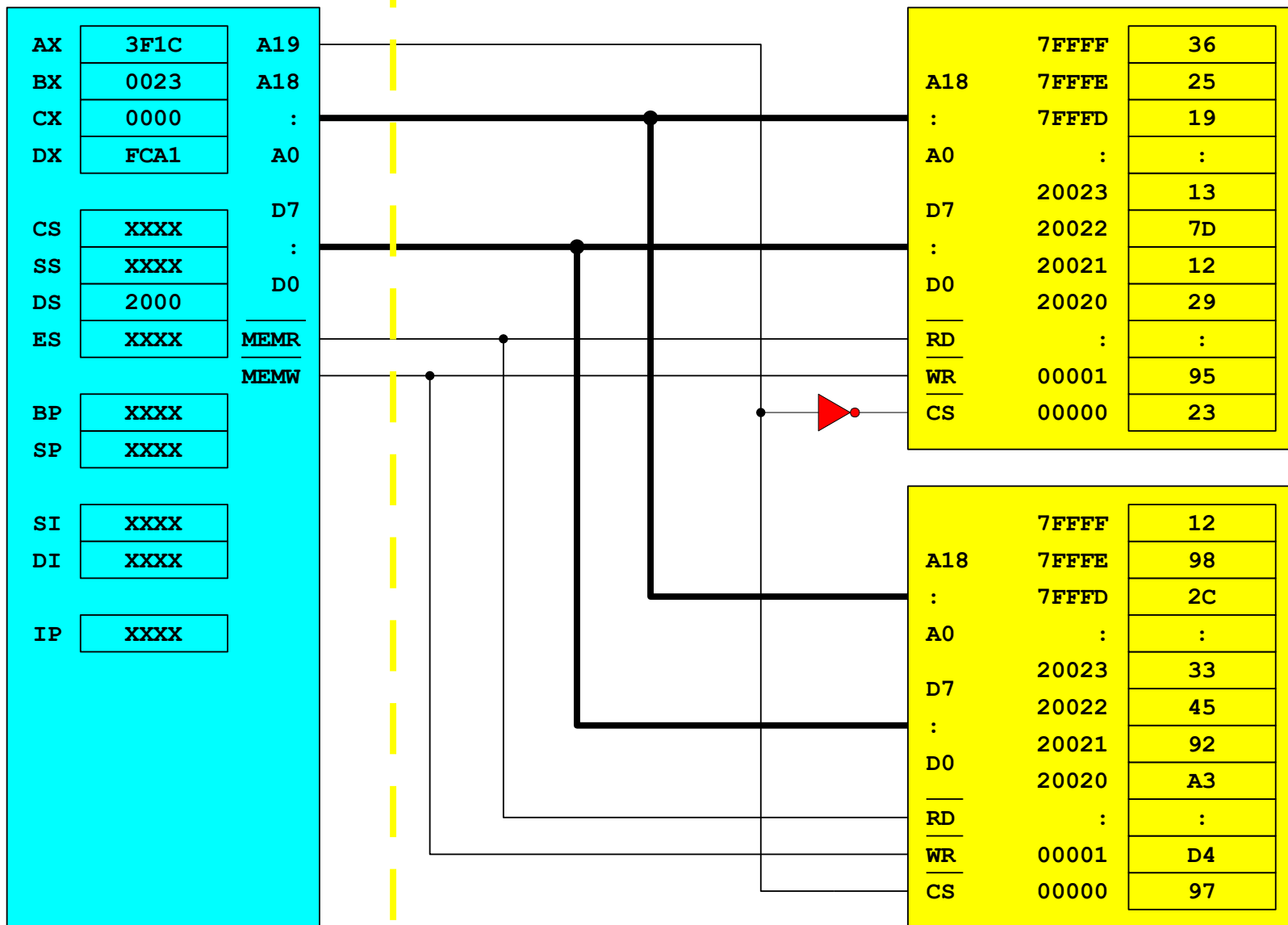
Không gian địa chỉ bộ nhớ 1M

A19 đến A0 (HEX)	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
00000	0000	0000	0000	0000	0000
7FFFF	0111	1111	1111	1111	1111
80000	1000	0000	0000	0000	0000
FFFFFF	1111	1111	1111	1111	1111

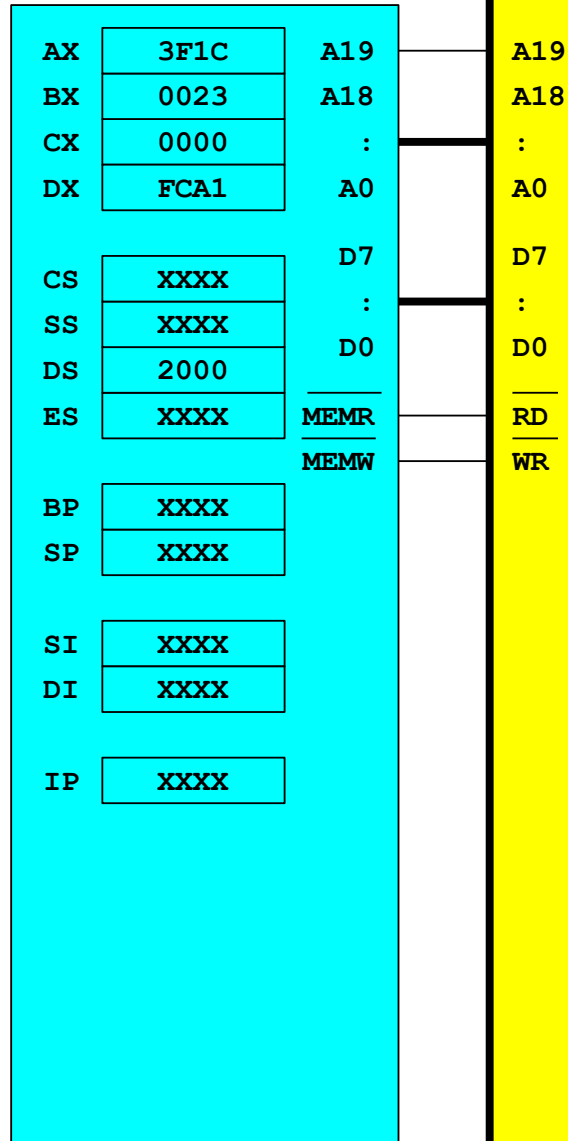
Bộ nhớ gồm hai khối nhớ 512KBa



Bộ nhớ gồm hai khối nhớ 512KB

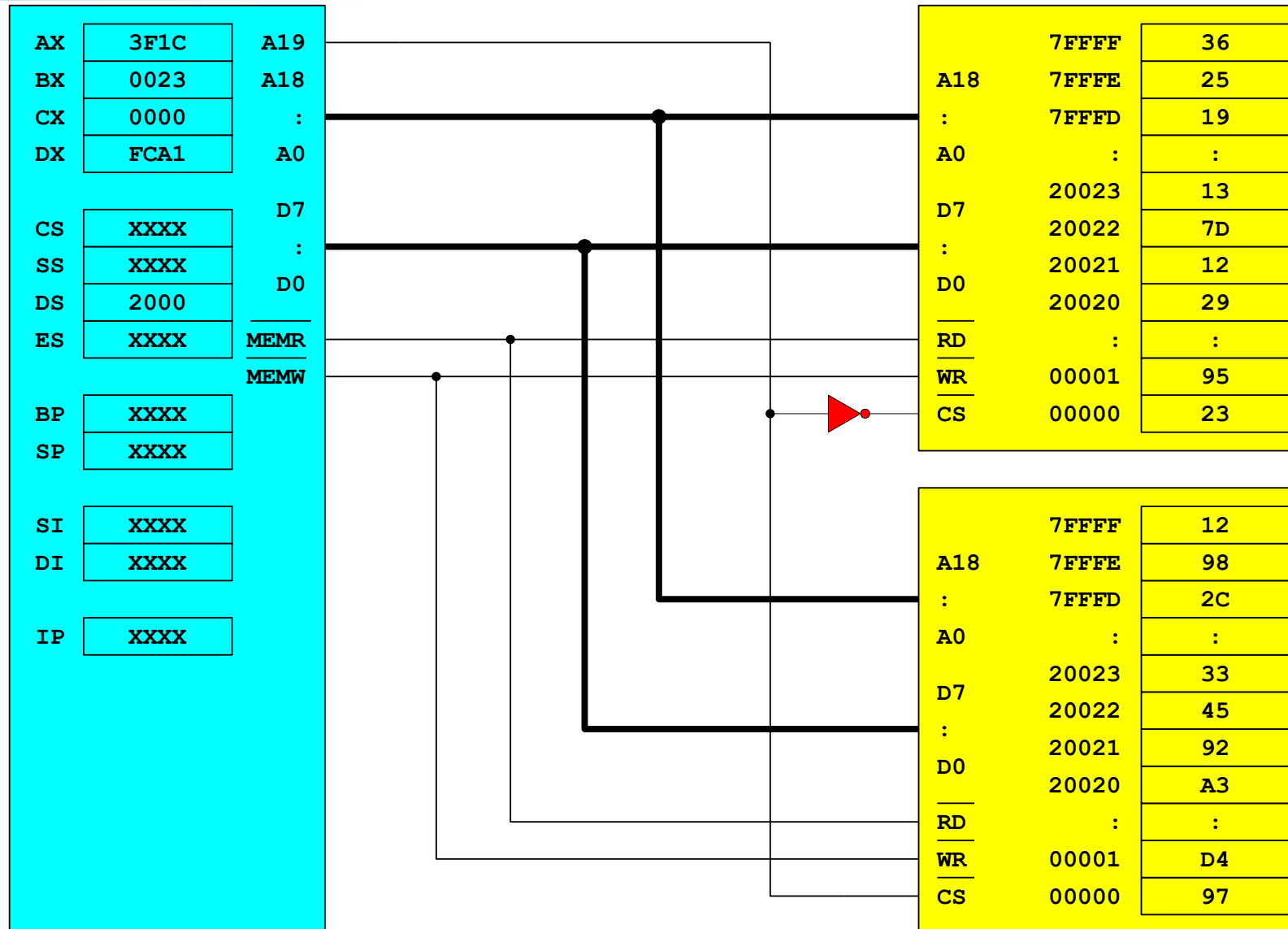


Thiết kế Bộ nhớ cho Hệ vi xử lý

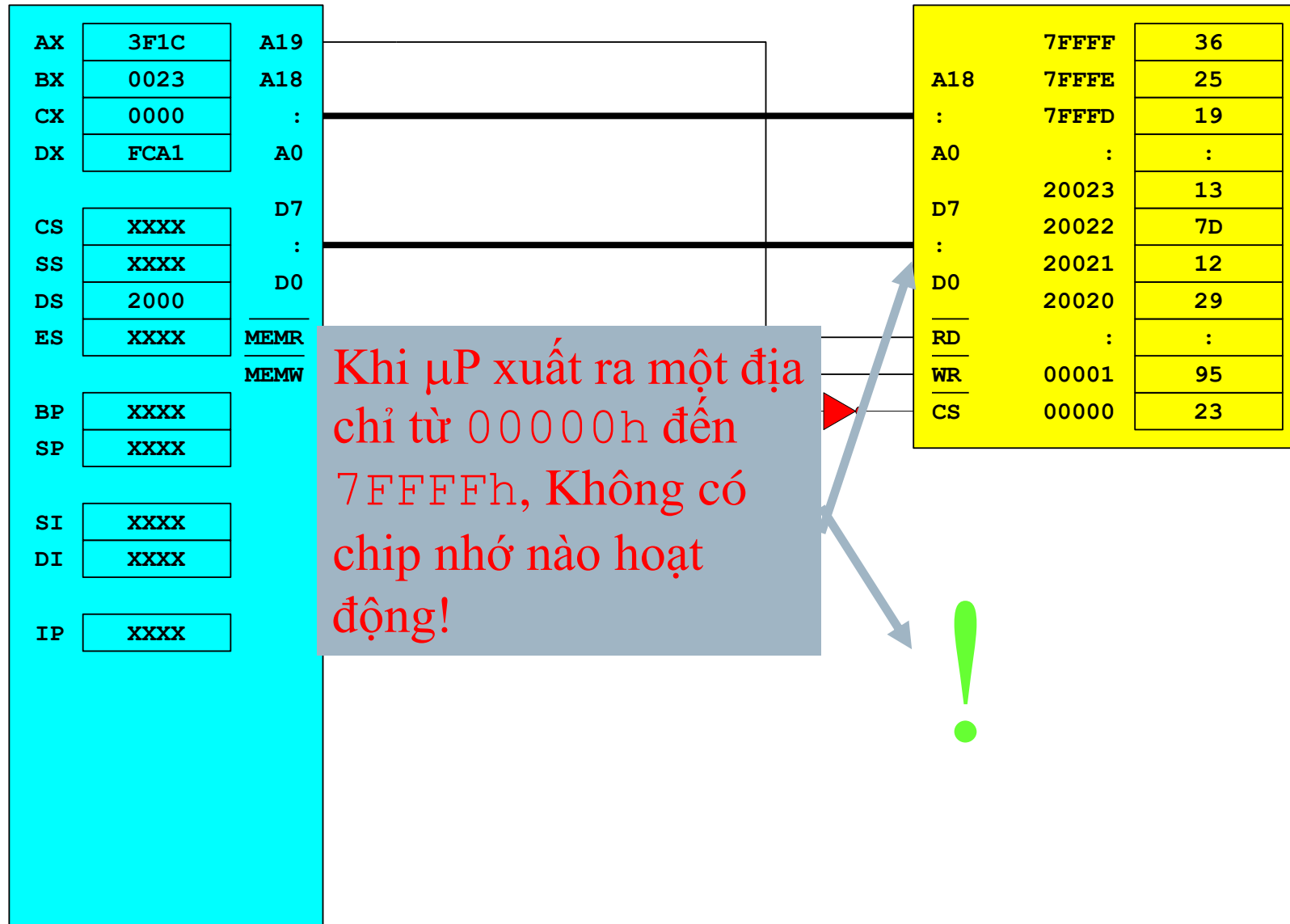


Ghép nối các chip nhớ riêng lẻ với Bus hệ thống sao cho không xảy ra xung đột nhờ mạch giải mã địa chỉ bộ nhớ

Nếu bỏ đi khối nhớ bên dưới?



Nếu bỏ đi khối nhớ bên dưới thì ...



Giải mã đầy đủ và không đầy đủ

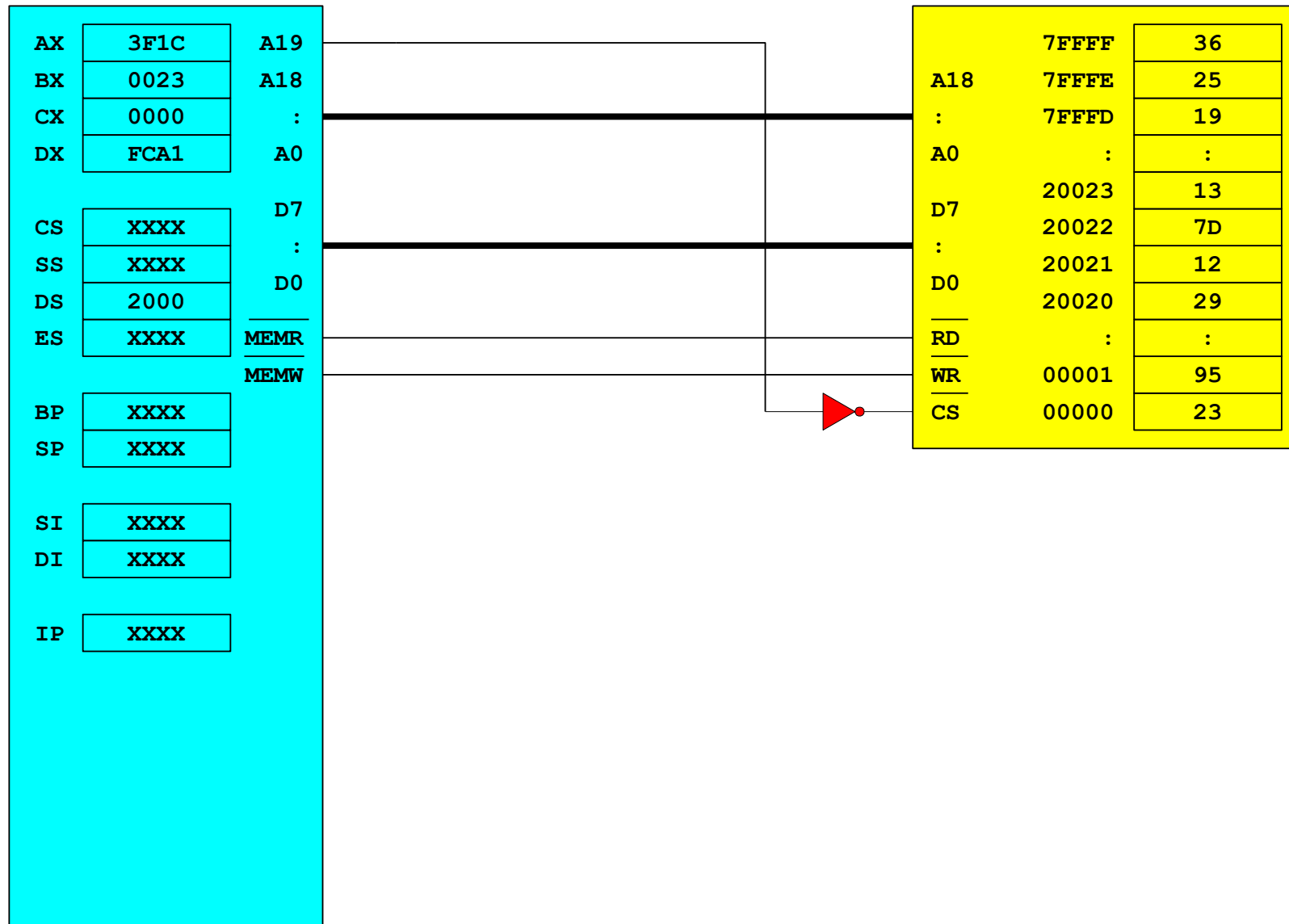
❖ Giải mã đầy đủ (Full Decoding)

- Tất cả các đường địa chỉ có nghĩa đều được sử dụng vào mạch giải mã
- Mỗi ô nhớ chỉ có một địa chỉ vật lý duy nhất

❖ Giải mã không đầy đủ (Partial Decoding)

- Không phải tất cả các đường địa chỉ có nghĩa đều được sử dụng vào mạch giải mã
- Một ô nhớ có hơn một địa chỉ vật lý

Giải mã đầy đủ



Giải mã đầy đủ

A19 đến A0 (HEX)	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
80000	1000	0000	0000	0000	0000
FFFFFF	1111	1111	1111	1111	1111

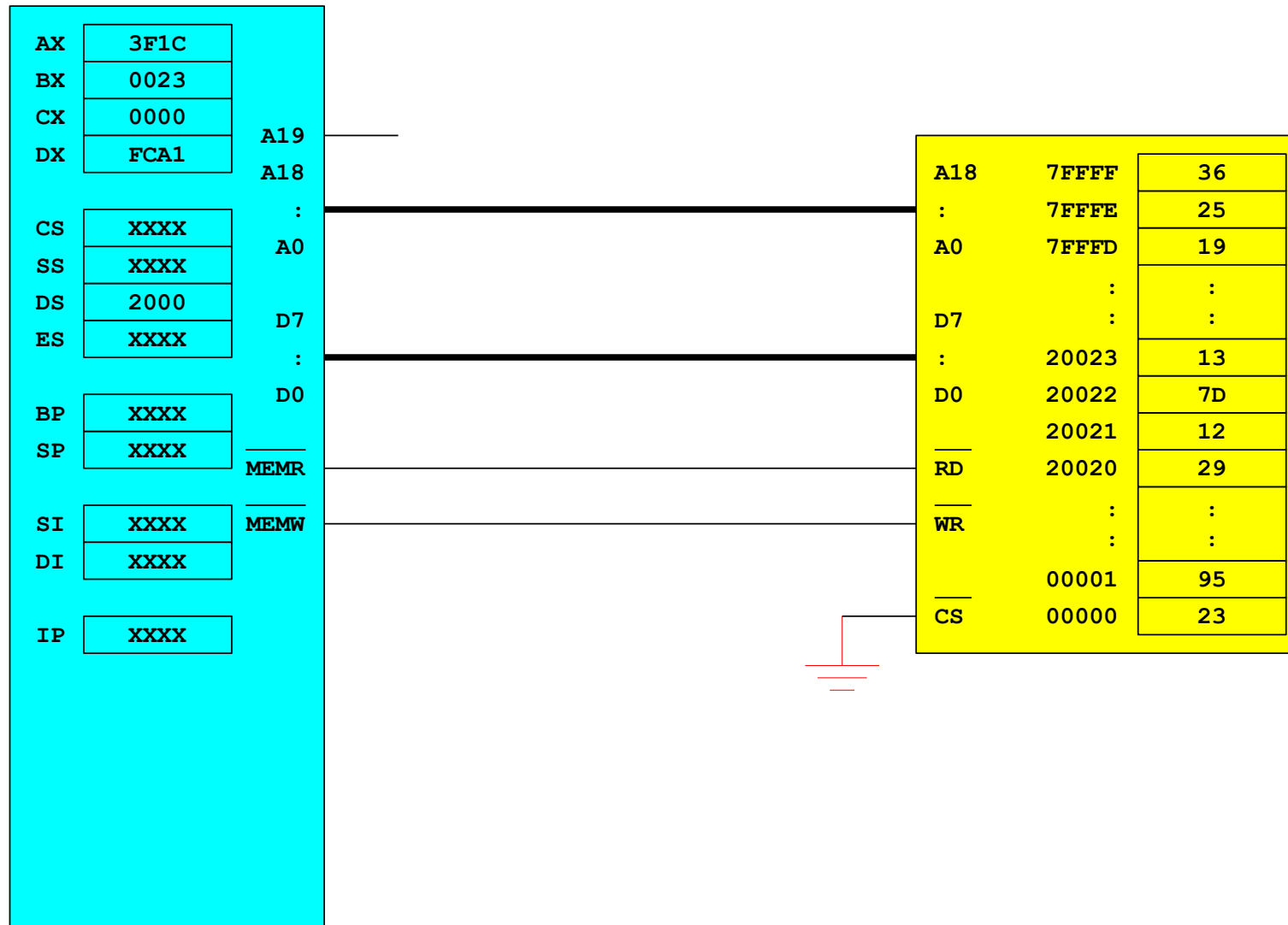
A₁₉ phải ở mức “1” thì chip nhớ mới hoạt động (được chọn)

Giải mã đầy đủ

A19 đến A0 (HEX)	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
00000	0000	0000	0000	0000	0000
7FFFF	0111	1111	1111	1111	1111

Do đó, nếu vi xử lý phát ra một địa chỉ từ 00000h đến 7FFFFh (A_{19} ở mức logic "0") chip nhớ sẽ không được chọn.

Giải mã không đầy đủ



Giải mã không đầy đủ

A19 đến A0 (HEX)	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
00000	0000	0000	0000	0000	0000
7FFFF	0111	1111	1111	1111	1111
80000	1000	0000	0000	0000	0000
FFFFFF	1111	1111	1111	1111	1111

A19 không có ý nghĩa với chip nhớ

Giải mã không đầy đủ

A19 đến A0 (HEX)	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
00000	0000	0000	0000	0000	0000
7FFFF	0111	1111	1111	1111	1111
80000	1000	0000	0000	0000	0000
FFFFFF	1111	1111	1111	1111	1111

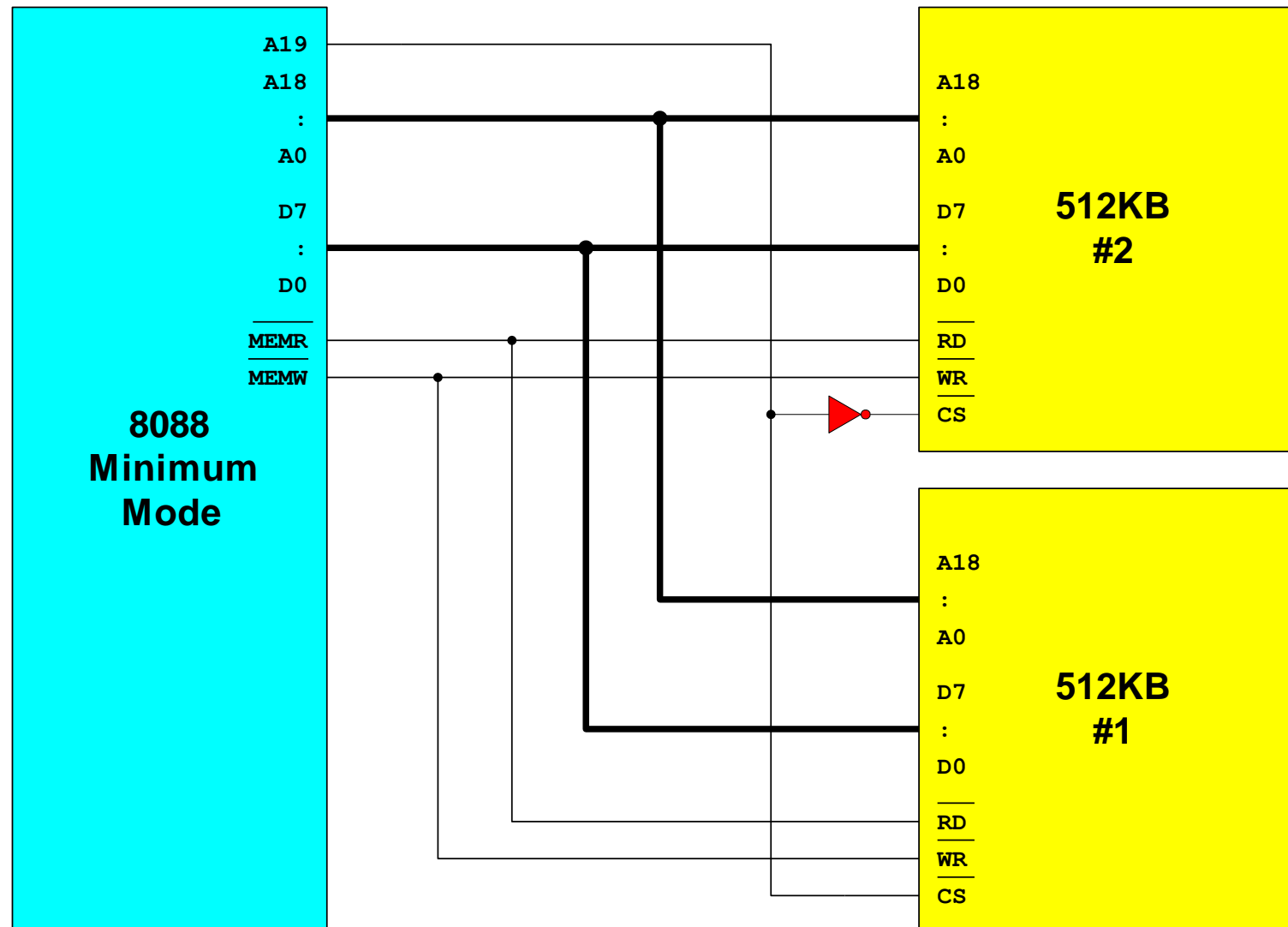
Địa chỉ thực tế

Giải mã không đầy đủ

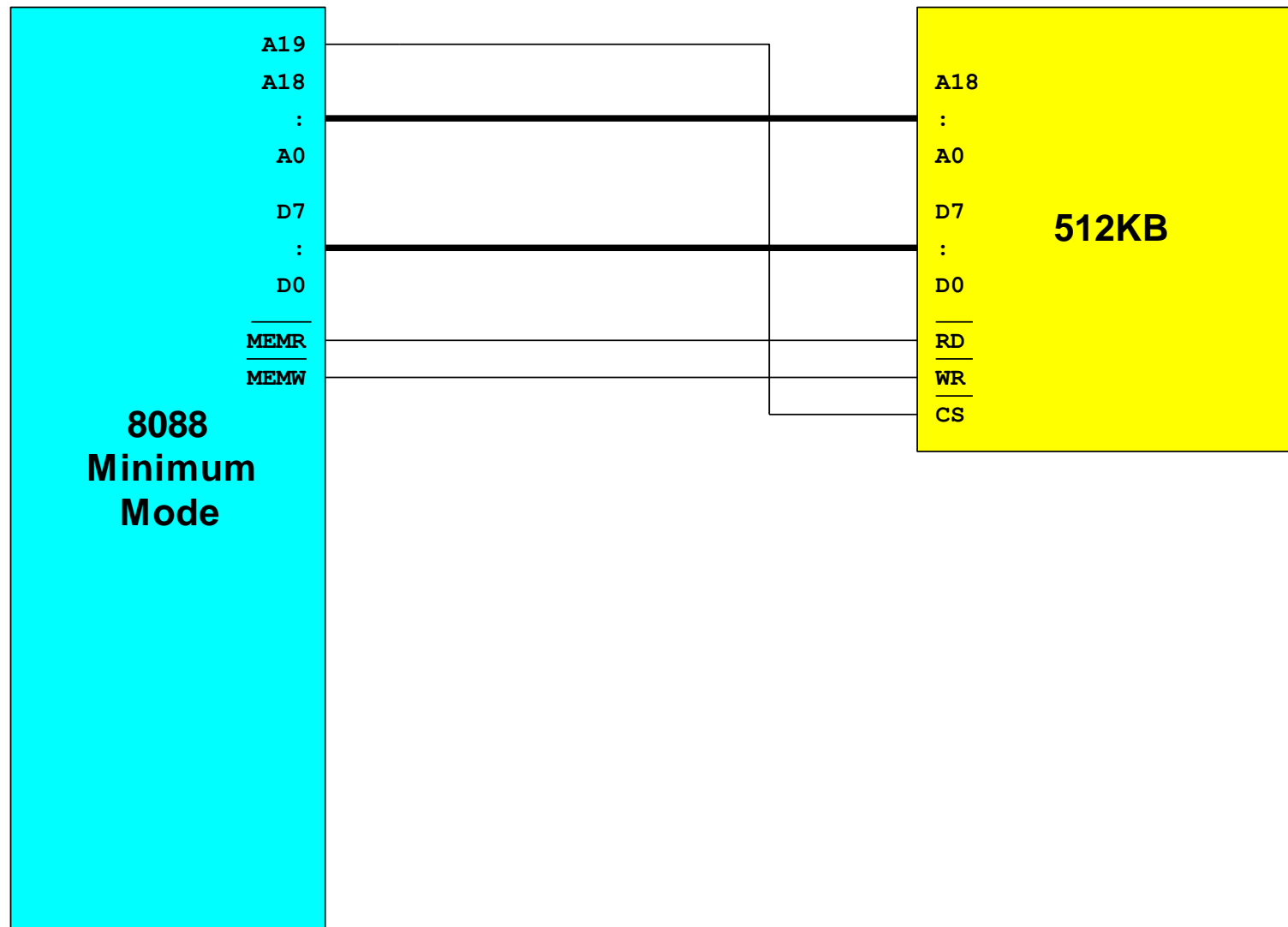
A19 đến A0 (HEX)	A ₁₉ A ₁₈ A ₁₇ A ₁₆	A ₁₅ A ₁₄ A ₁₃ A ₁₂	A ₁₁ A ₁₀ A ₉ A ₈	A ₇ A ₆ A ₅ A ₄	A ₃ A ₂ A ₁ A ₀
00000	0000	0000	0000	0000	0000
7FFFF	0111	1111	1111	1111	1111
80000	1000	0000	0000	0000	0000
FFFFF	1111	1111	1111	1111	1111

Địa chỉ thực tế

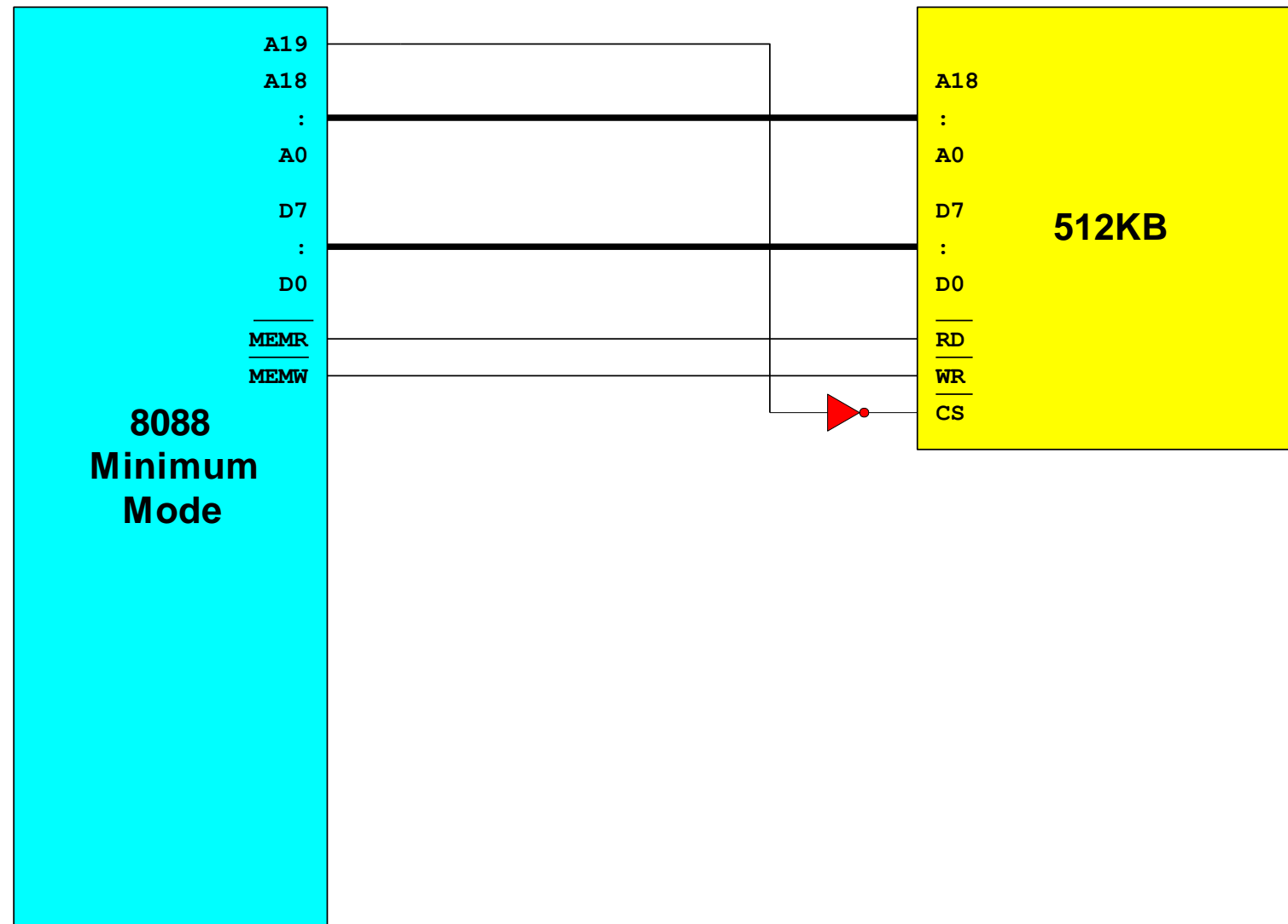
Bộ nhớ gồm 2 chip 512Kx8



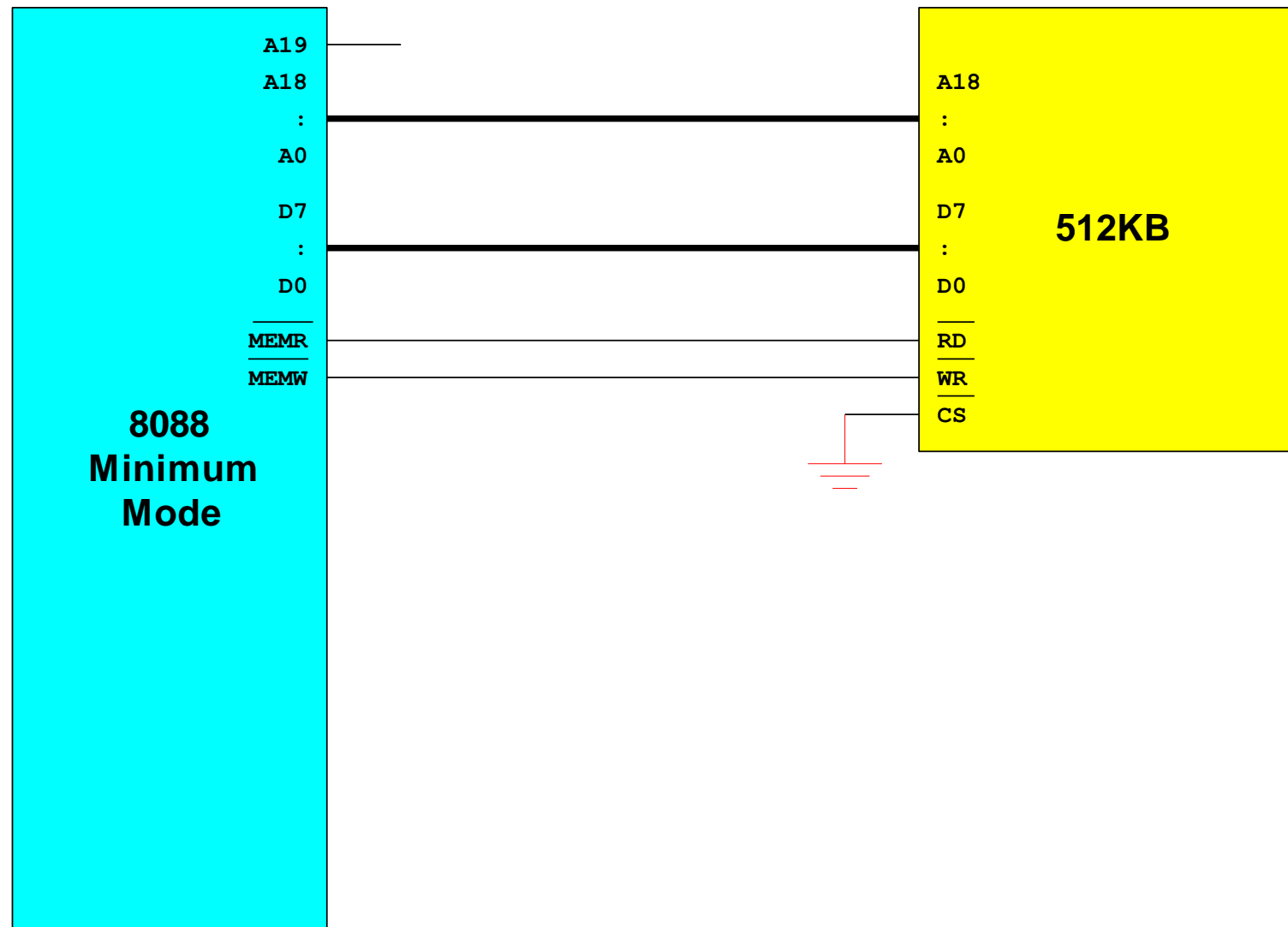
Bộ nhớ chỉ có một chip 512Kx8 (Ver. 1)



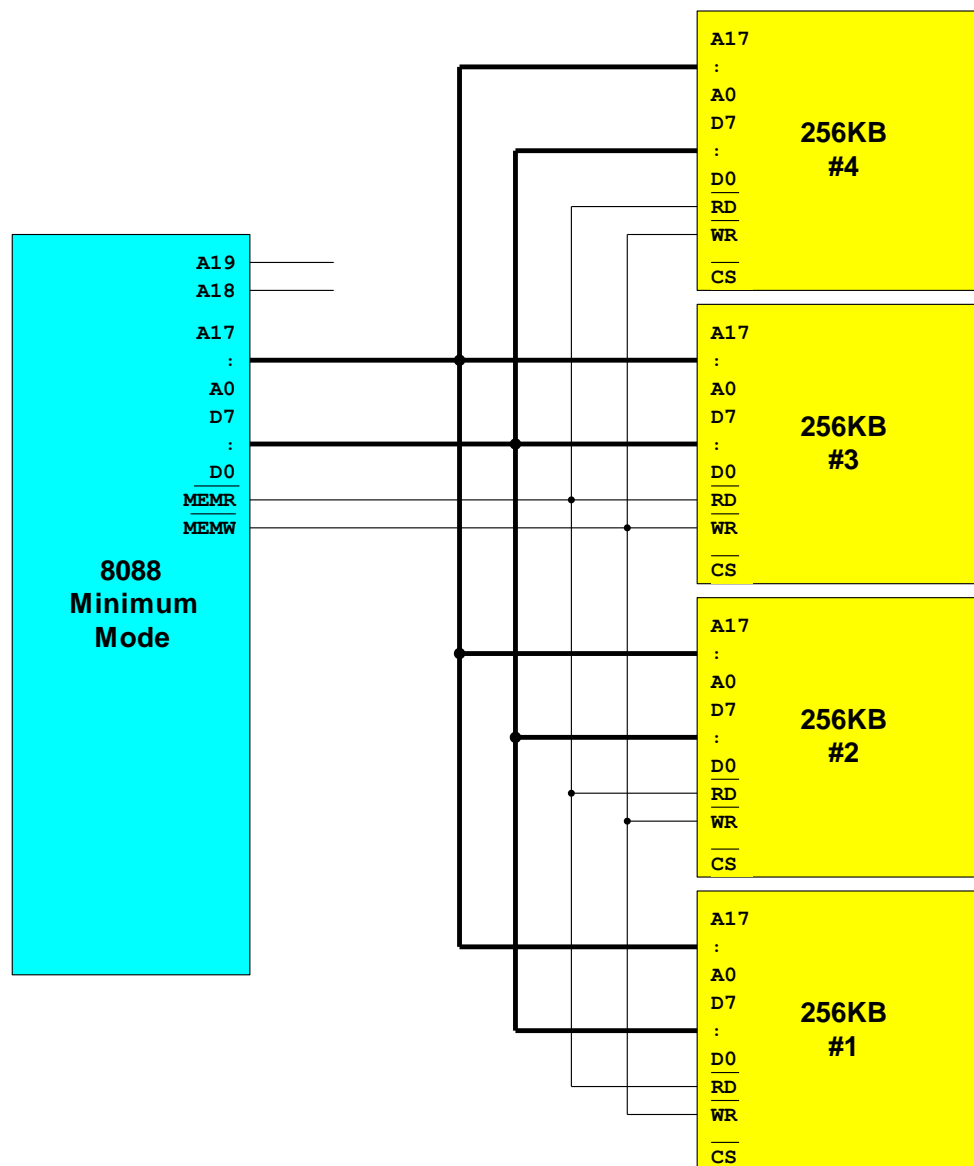
Bộ nhớ chỉ có một chip 512Kx8 (Ver. 2)



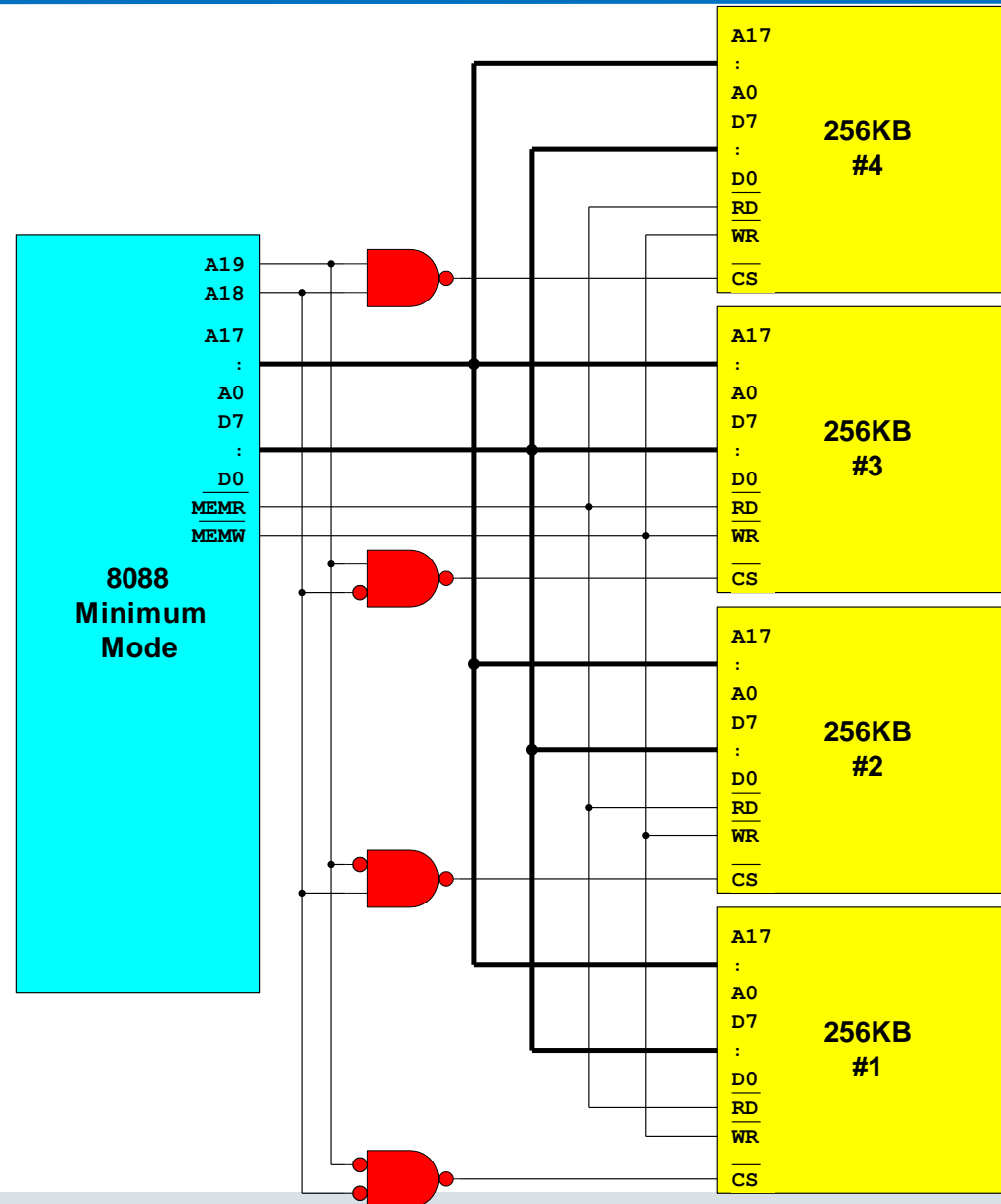
Bộ nhớ chỉ có một chip 512Kx8 (Ver. 3)



Bộ nhớ gồm 4 chip 256Kx8



Bộ nhớ gồm 4 chip 256Kx8



Dải địa chỉ của chip #1

A19 đến A0 (HEX)	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
00000 -----	0000 -----	0000 -----	0000 -----	0000 -----	0000 -----
3FFFF -----	0011 -----	1111 -----	1111 -----	1111 -----	1111 -----

Dải địa chỉ của chip #2

A19 đến A0 (HEX)	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
40000 -----	01000 -----	0000 -----	0000 -----	0000 -----	0000 -----
7FFFF -----	0111 -----	1111 -----	1111 -----	1111 -----	1111 -----

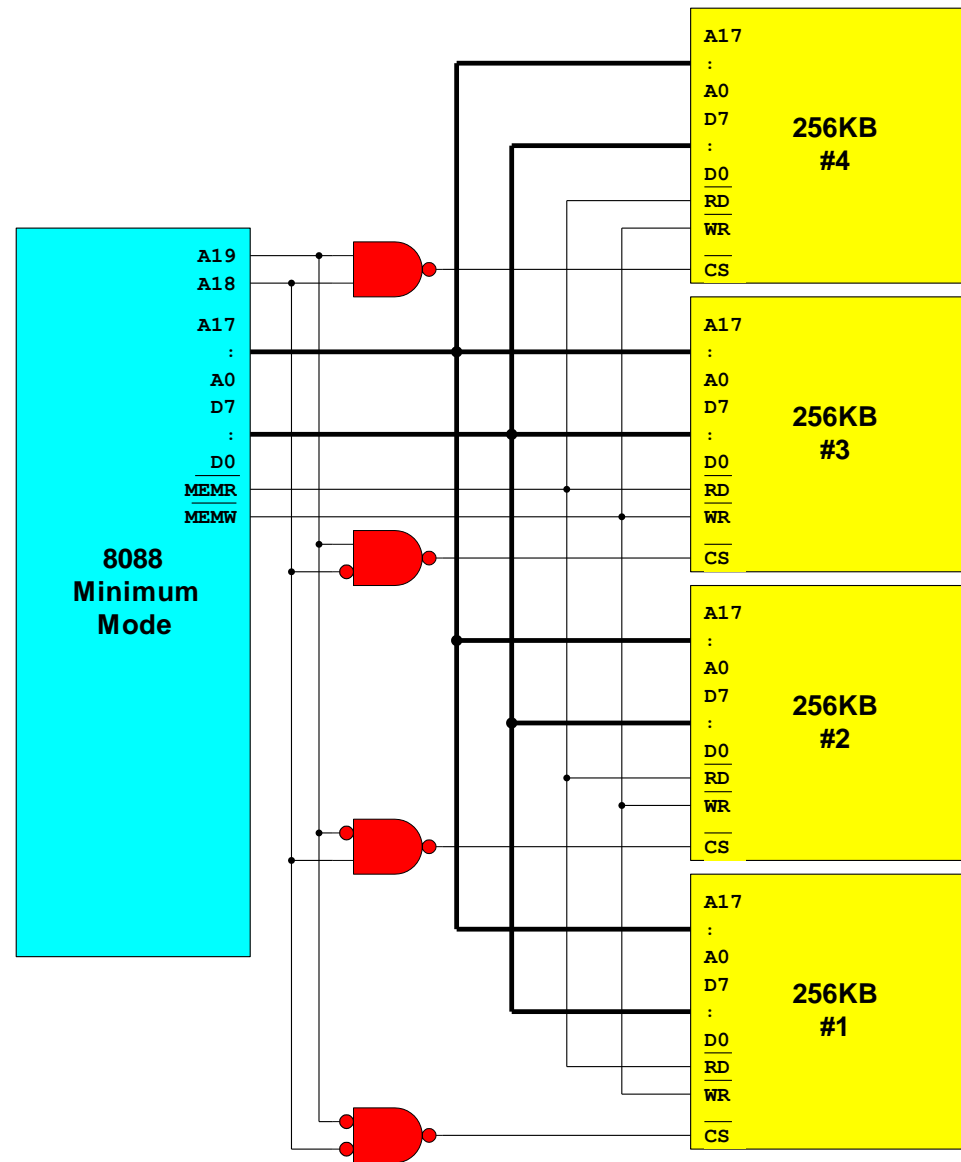
Dải địa chỉ của chip #3

A19 đến A0 (HEX)	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
80000 -----	1000 -----	0000 -----	0000 -----	0000 -----	0000 -----
BFFFF -----	1011 -----	1111 -----	1111 -----	1111 -----	1111 -----

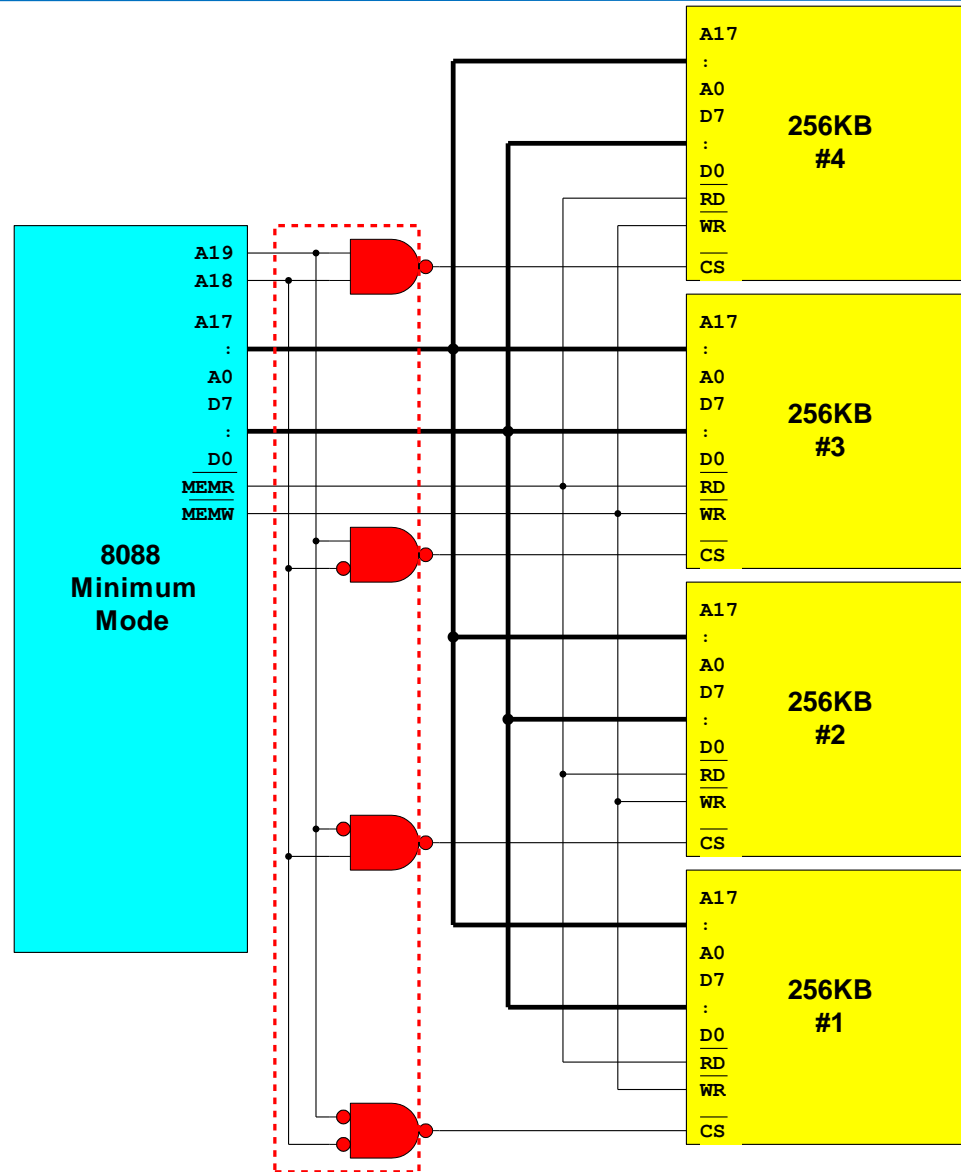
Dải địa chỉ của chip #4

A19 đến A0 (HEX)	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
C0000 -----	1100 -----	0000 -----	0000 -----	0000 -----	0000 -----
FFFFFF -----	1111 -----	1111 -----	1111 -----	1111 -----	1111 -----

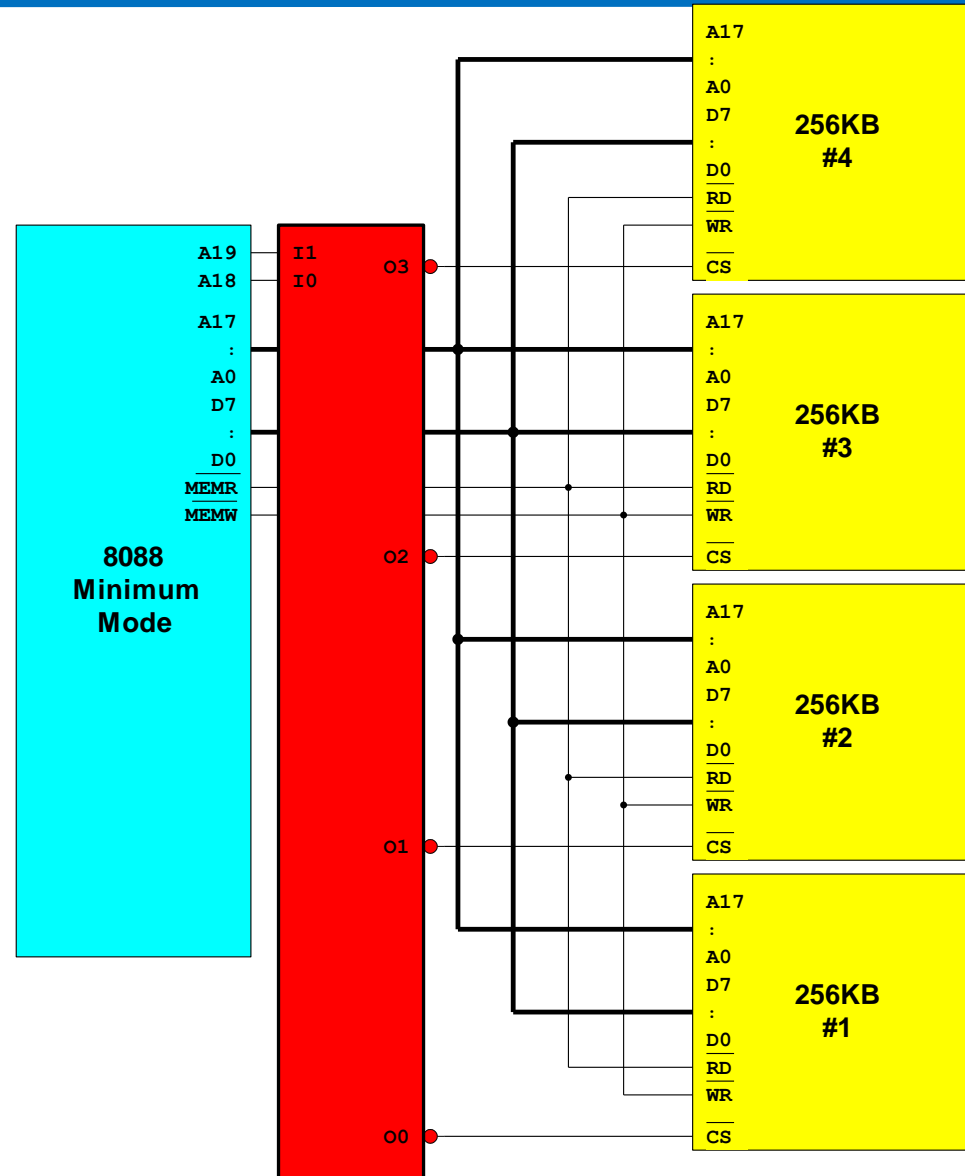
Bộ nhớ gồm 4 chip 256Kx8 dùng các cổng logic



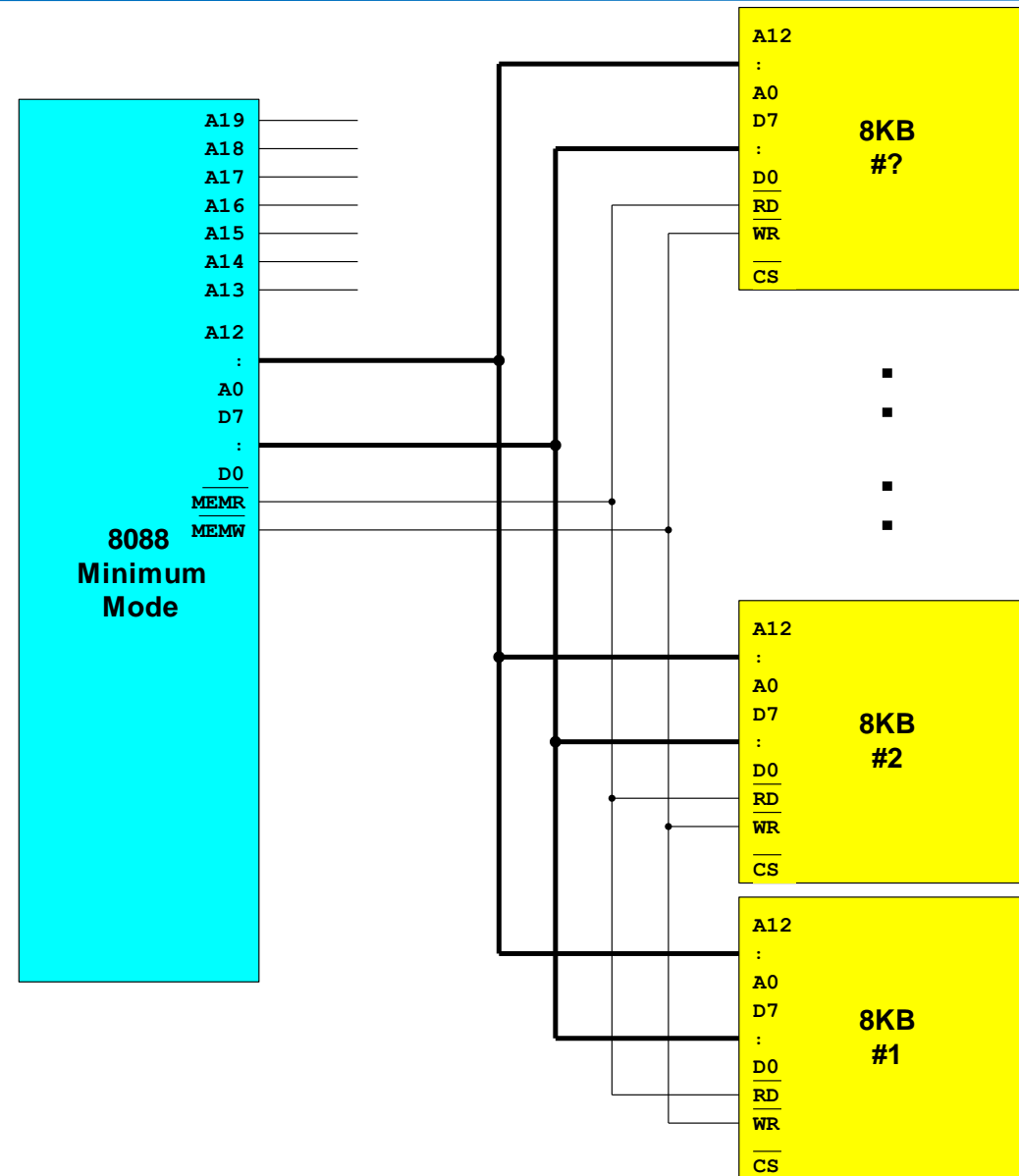
Bộ nhớ gồm 4 chip 256Kx8 dùng các cổng logic



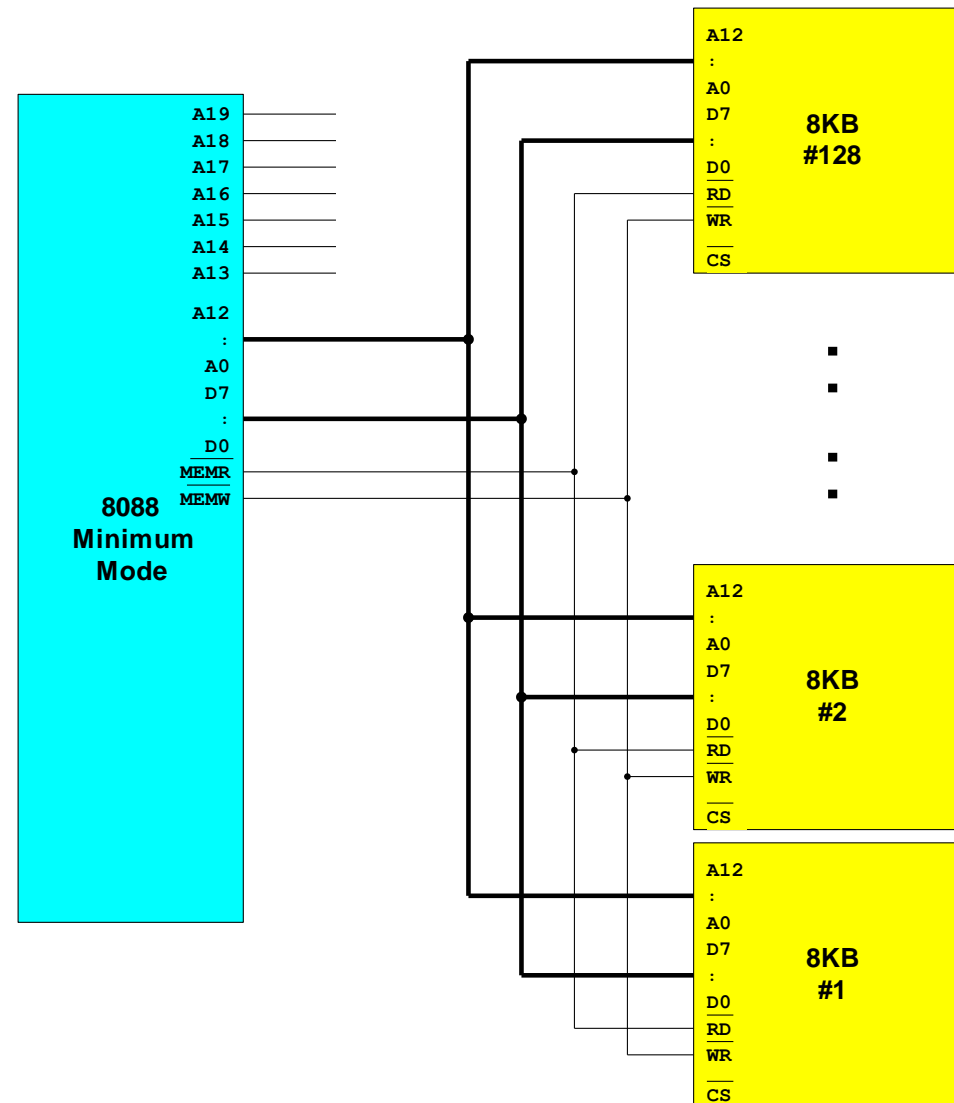
Bộ nhớ gồm 4 chip 256Kx8 dùng một chip giải mã 2-4



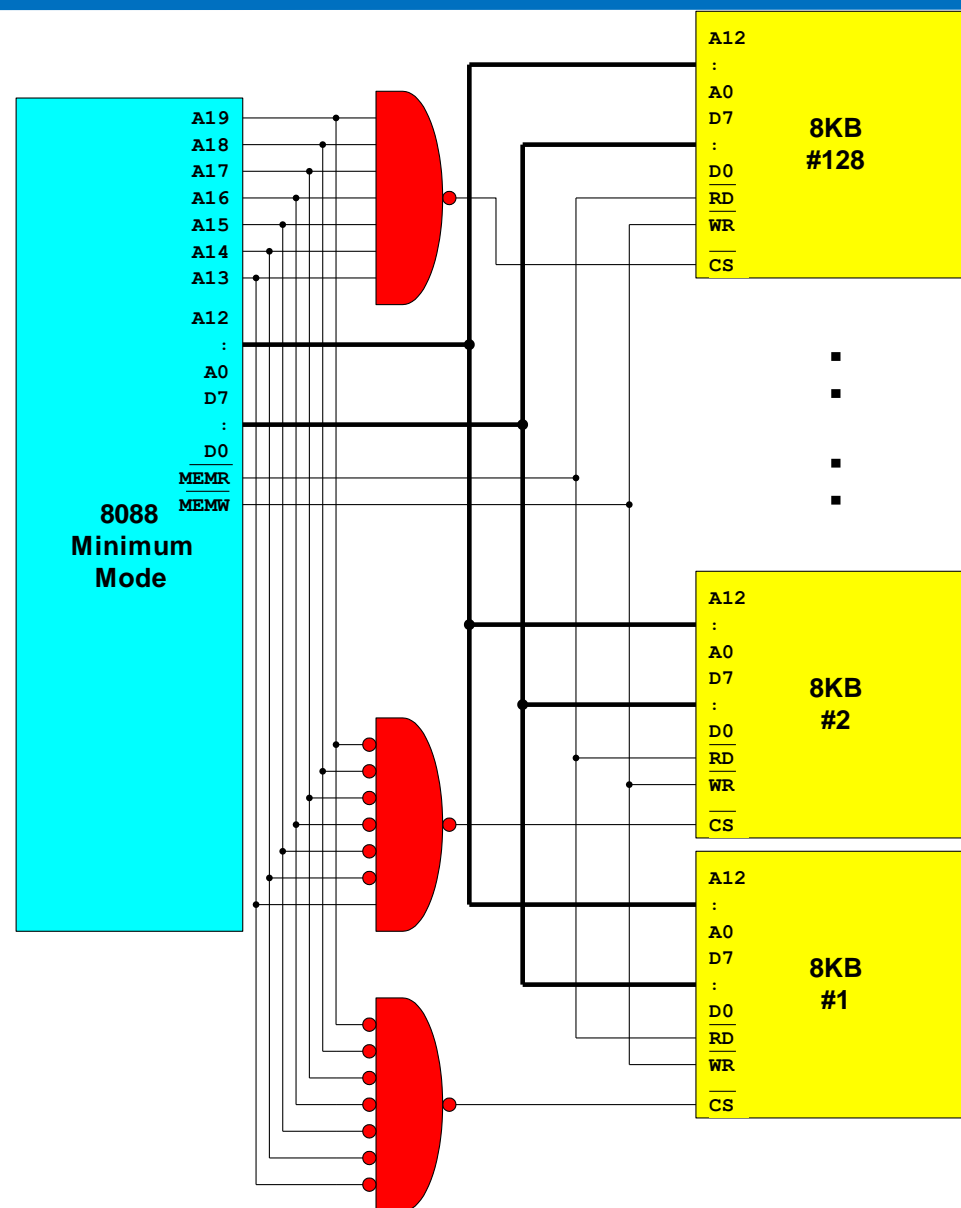
Ghép nối các chip nhớ 8Kx8 với μ P 8088



Bộ nhớ gồm 128 chip 8Kx8



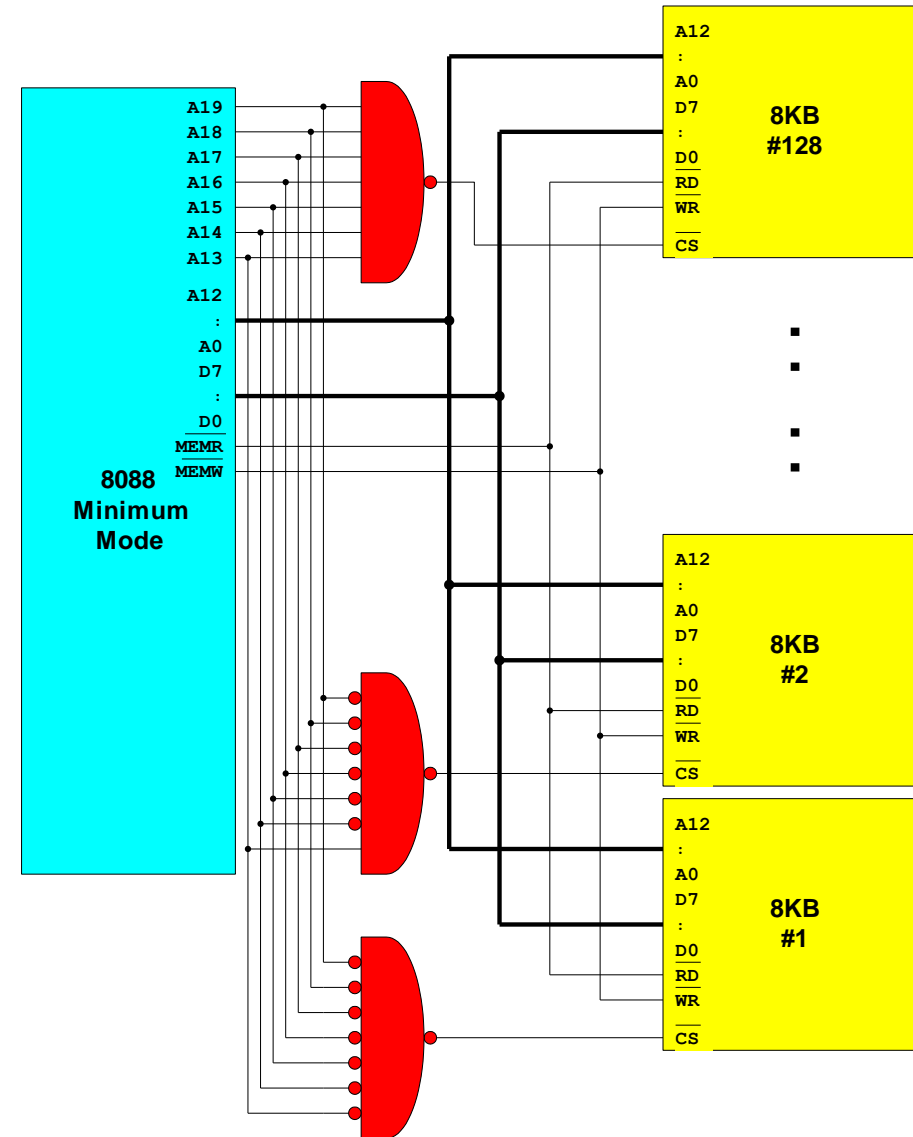
Bộ nhớ gồm 128 chip 8Kx8



Dải địa chỉ của Chip #__

A19 đến A0 (HEX)	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
-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----

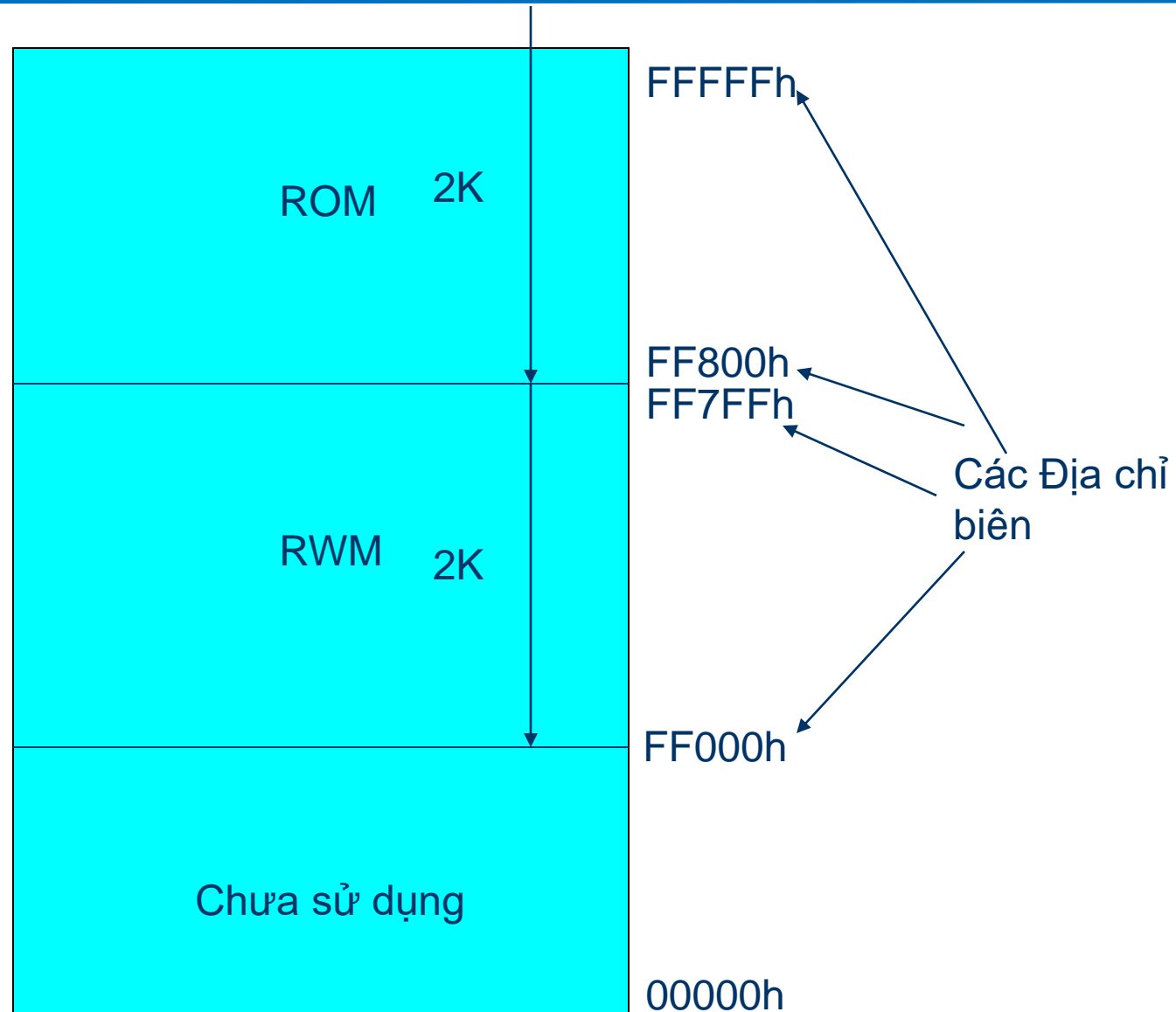
Bộ nhớ gồm 128 chip 8Kx8



Phát biểu bài toán

- ❖ Thiết kế bộ nhớ cho hệ vi xử lý 8088 thoả mãn các yêu cầu:
 - ROM có dung lượng 2Kx8 chiếm dụng các địa chỉ từ FFFFFh trở xuống
 - RWM có dung lượng 2Kx8 chiếm dụng các địa chỉ tiếp theo ngay sau ROM
 - Chỉ được phép sử dụng:
 - EPROM 2716 2Kx8
 - SRAM 4016 2Kx8
 - Chip giải mã 74LS138
 - Các cổng logic

Bước 1: Vẽ bản đồ bộ nhớ cần thiết kế



Bước 2: Chuyển các địa chỉ biên từ H sang B

A19 đến A0 (HEX)	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
FF800	1111	1111	1000	0000	0000
FFFFFF	1111	1111	1111	1111	1111

Bước 2: Chuyển các địa chỉ biên từ H sang B

A19 đến A0 (HEX)	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
FF000	1111	1111	0000	0000	0000
FF7FF	1111	1111	0111	1111	1111

Nhận xét

- ❖ Khi các địa chỉ dành cho ROM được đưa lên Address Bus: $A_{19} - A_{12} = 1$ và $A_{11} = 1$
- ❖ Khi các địa chỉ dành cho RWM được đưa lên Address Bus: $A_{19} - A_{12} = 1$ và $A_{11} = 0$

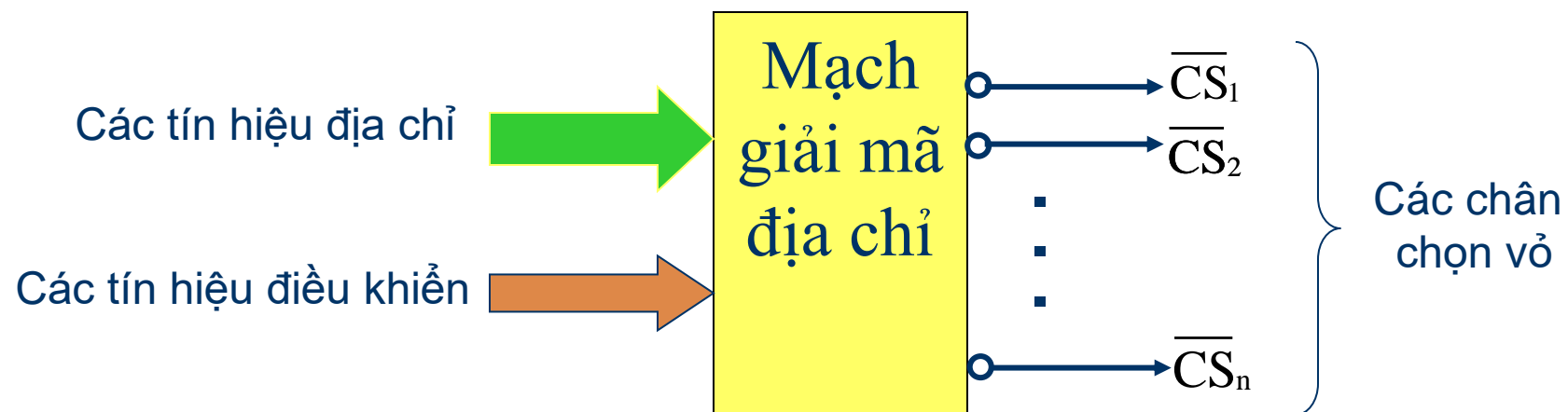
Bước 3: Vẽ mạch giải mã địa chỉ bộ nhớ

- ❖ Ghép các chân dữ liệu của các chip nhớ với D-Bus
- ❖ Ghép các chân địa chỉ và các chân điều khiển:
 - Khi vi xử lý truy cập các cổng I/O thì các chip nhớ bị cấm (Khi $IO/M = 1$)
 - Khi vi xử lý truy cập bộ nhớ ($IO/M = 0$) thì chỉ có một chip nhớ làm việc.
- ❖ Có thể có nhiều lời giải khác nhau

Kỹ thuật giải mã

- ❖ Để CPU có thể thâm nhập vào từng ô nhớ của hệ thống → phải giải mã địa chỉ từ CPU để kích hoạt lên một vùng duy nhất trong bản đồ bộ nhớ.
- ❖ Nếu không có hệ thống giải mã thì chỉ có thể có một thiết bị nhớ duy nhất có thể ghép nối với CPU.
- ❖ Một số trong kỹ thuật giải mã:
 - Phương pháp giả mã NAND đơn giản
 - Phương pháp dùng các vi mạch giải mã (74LS138, 74LS139 ...)
- ❖ Ví dụ: ghép nối EPROM 2716 với CPU.
 - EPROM 2716 có 11 đường địa chỉ, trong khi CPU có 20 (CPU 8088)
 - CPU gửi ra địa chỉ 20 bit của bộ nhớ mỗi khi nó đọc hay ghi dữ liệu.

Mạch giải mã địa chỉ tổng quát



Mạch giải mã địa chỉ tổng quát

❖ Đầu vào của bộ giải mã:

■ Các tín hiệu địa chỉ:

- Gồm các bit địa chỉ có quan hệ nhất định với các tín hiệu chọn vỏ ở đầu ra.

■ Tín hiệu điều khiển:

- Thường là tín hiệu IO/M dùng để phân biệt đối tượng mà CPU chọn làm việc là bộ nhớ hay thiết bị vào ra.

❖ Đầu ra của bộ giải mã là các tín hiệu chọn vỏ

- Được nối với các tín hiệu chọn vỏ của các chip nhớ.

Phương pháp giải mã NAND đơn giản

- ❖ Ví dụ 1: Ghép nối EPROM 2716 (2Kx8) với CPU. Thiết kế mạch giải mã đơn giản dùng mạch NAND cho EPROM này với địa chỉ đầu là FF800h.

Phương pháp giả mã NAND đơn giản

EPROM 2716 (2Kx8)

Dải địa chỉ:

Địa chỉ đầu: FF800h

Địa chỉ cuối: FFFFFh

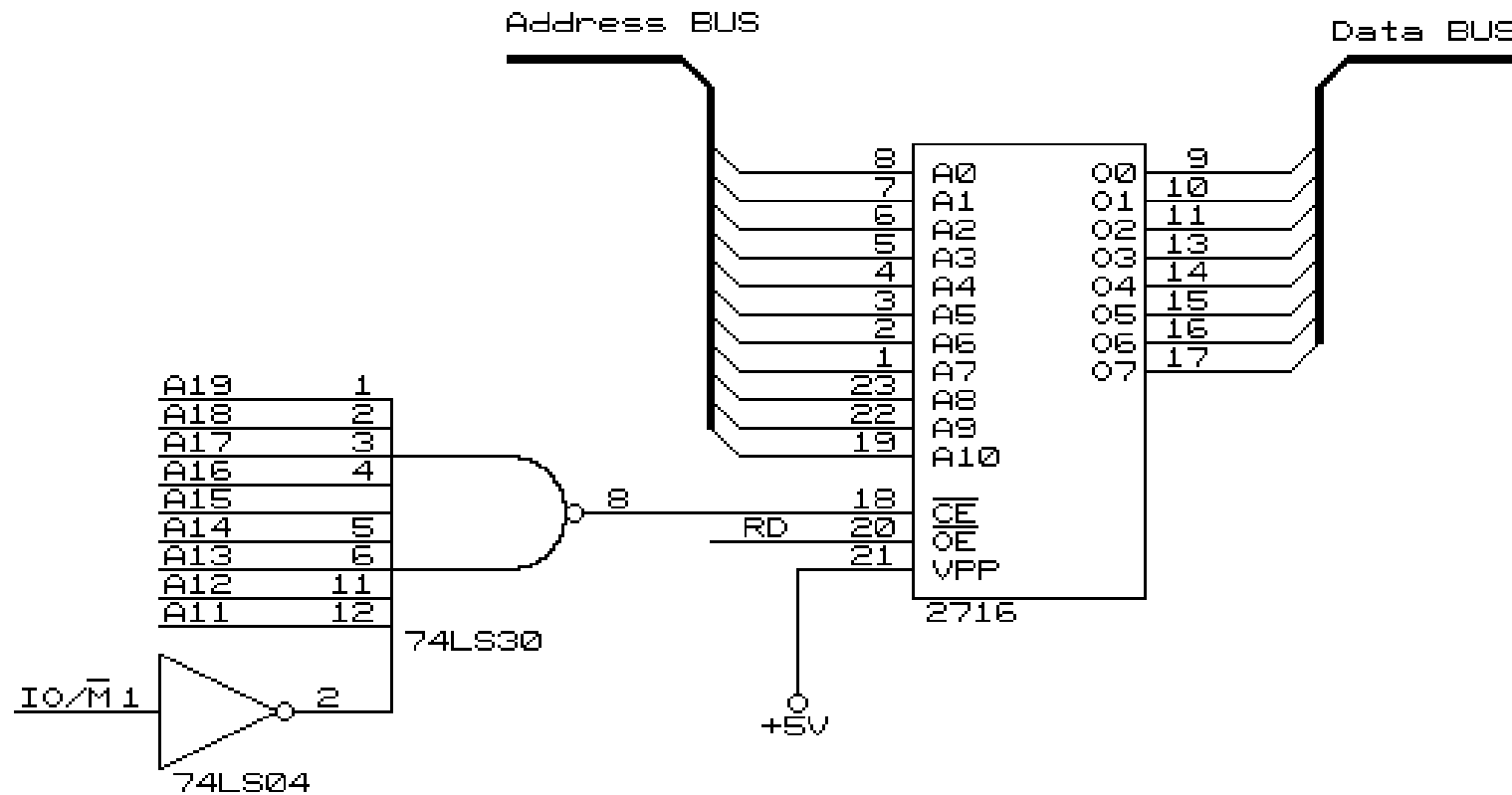
A ₁₉ đến A ₀ (HEX)	A ₁₉ A ₁₈ A ₁₇ A ₁₆	A ₁₅ A ₁₄ A ₁₃ A ₁₂	A ₁₁ A ₁₀ A ₉ A ₈	A ₇ A ₆ A ₅ A ₄	A ₃ A ₂ A ₁ A ₀
FF800	1111	1111	1000	0000	0000
FFFFFF	1111	1111	1111	1111	1111

Phần địa chỉ không đổi

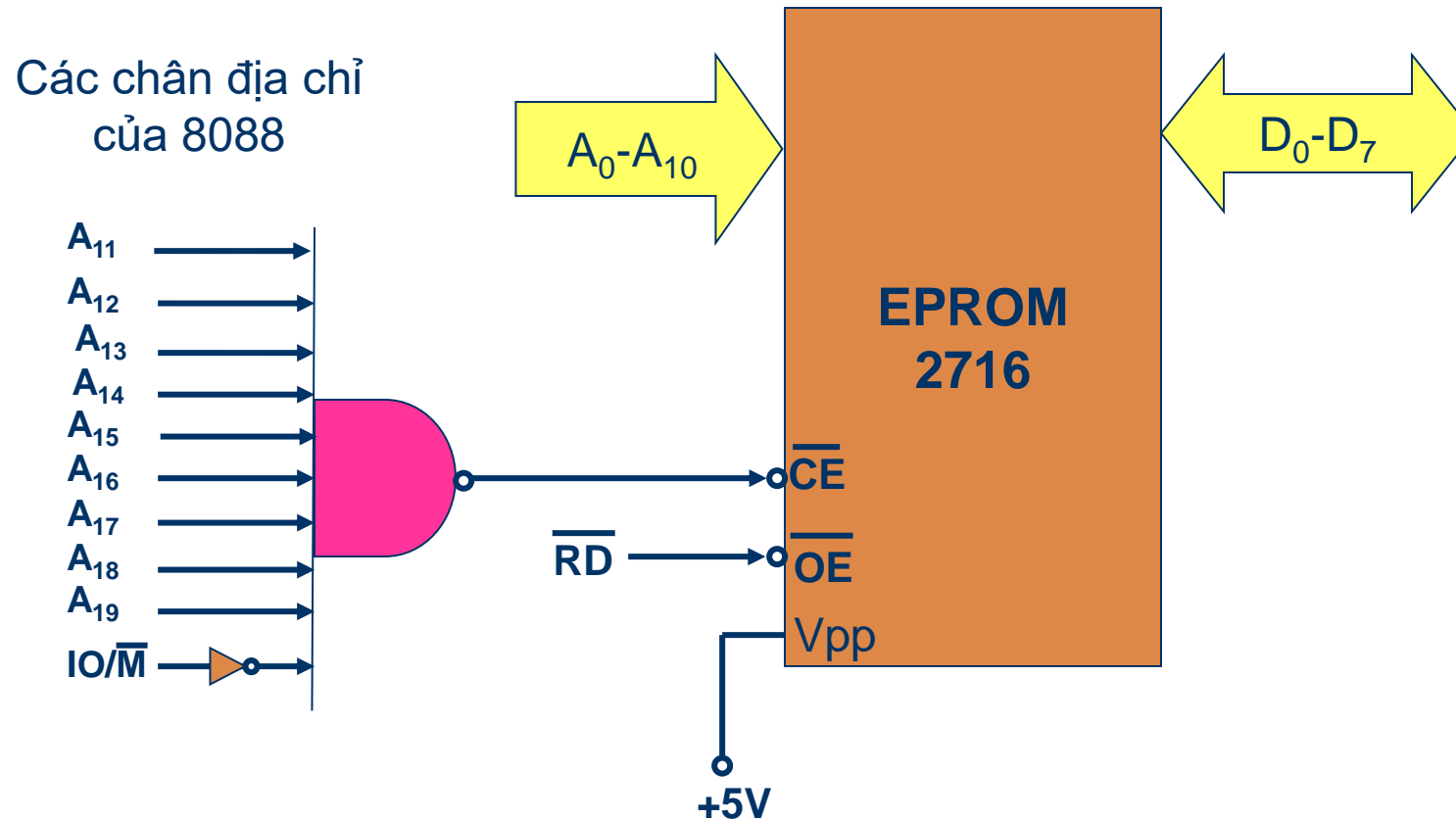
Phần địa chỉ thay đổi

Phương pháp giả mã NAND đơn giản

- ❖ Ví dụ 1: Mạch giải mã NAND đơn giản cho EPROM 2716 (2Kx8) với không gian địa chỉ FF800H-FFFFFFH



Ví dụ: ghép nối EPROM 2716 với CPU



Phương pháp giải mã NAND đơn giản

- ❖ Các đường địa chỉ $A_{10}-A_0$ của 8088 nối với các chân địa chỉ $A_{10}-A_0$ của EPROM.
- ❖ 9 chân địa chỉ ($A_{19}-A_{11}$) ở mức 1 kết hợp cùng với xung IO/M (đã được đảo) để tạo ra xung chọn vùng nhớ 2KB đặt tại CE nối với bộ giải mã để thực hiện việc chọn một trong nhiều phần 2KB trong bản đồ bộ nhớ 1MB của 8088.
- ❖ Mỗi ô nhớ cụ thể trong 2KB của mạch nhớ EPROM do các bit A_0-A_{10} chọn ra.
- ❖ Trong mạch giải mã dùng phương pháp cổng NAND đơn giản này, một cổng NAND được dùng trong việc giải mã.
 - Đầu ra của NAND sẽ ở mức logic "0" nếu tất cả các đầu vào của nó có mức logic "1"
 - Mức logic "0" ở đầu ra được đưa tới đầu vào (*Chip Enable*) để kích hoạt EPROM. Dữ liệu được đọc ra khỏi EPROM khi cũng ở logic 0, điều này được thực hiện thông qua việc đưa tín hiệu vào.

Ví dụ 2: ghép nối EPROM 2732 với CPU

❖ Địa chỉ đầu là A3000h

Ví dụ: ghép nối EPROM 2732 với CPU

❖ Địa chỉ đầu là A3000h

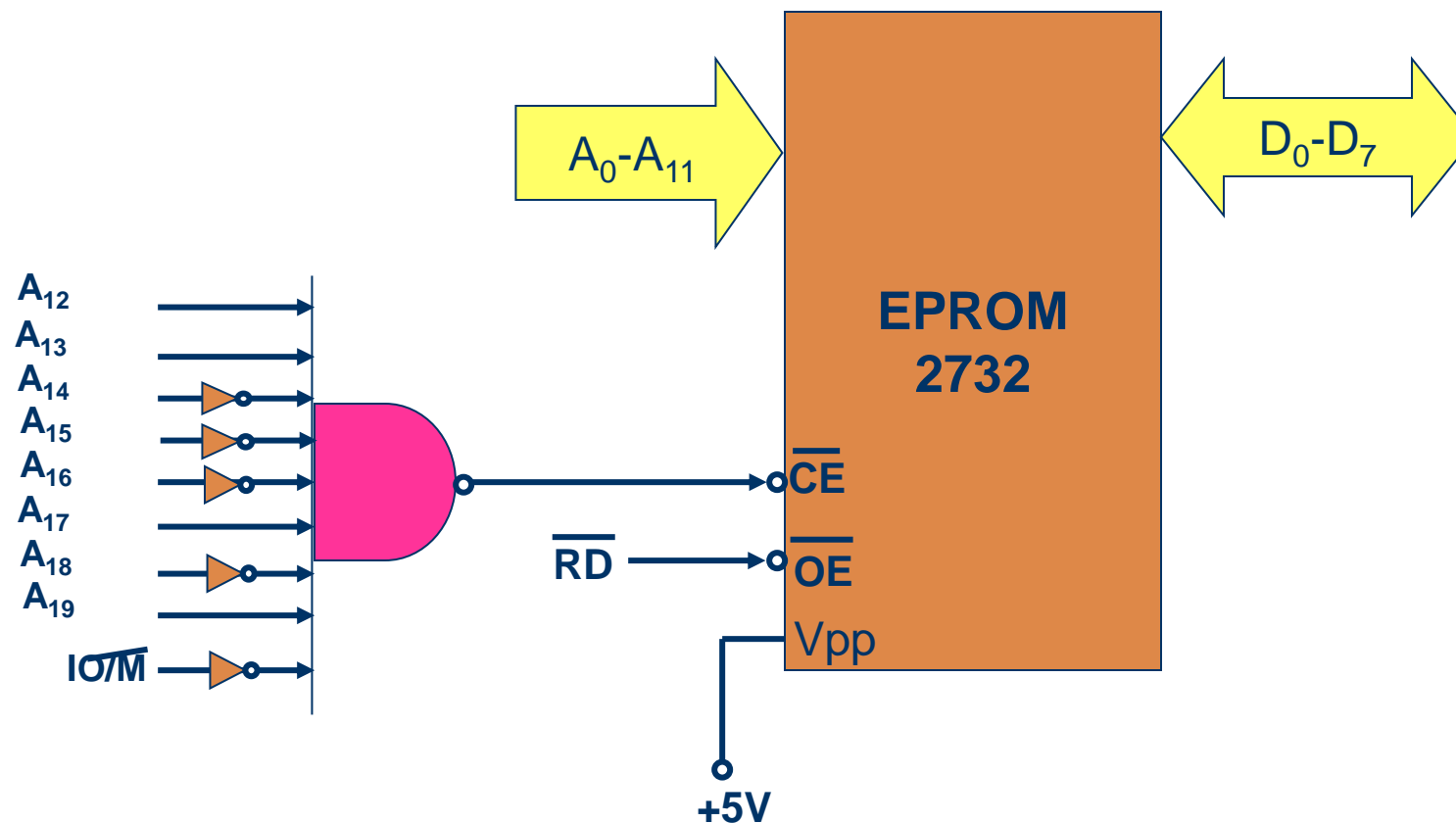
❖ → Dải địa chỉ: A3000 h → A3FFF h

A_{19} đến A_0 (HEX)	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
A3000	1010	0011	0000	0000	0000
A3FFF	1010	0011	1111	1111	1111

Phần địa chỉ không đổi

Phần địa chỉ thay đổi

Ví dụ: ghép nối EPROM 2732 với CPU



Bài tập

- ❖ Có các vi mạch SRAM 8KB cùng các mạch phụ cần thiết. Hãy thiết kế bộ giải mã địa chỉ để ghép nối với bộ vi xử lý 8088 (chế độ MIN) tạo ra bộ nhớ dung lượng 16KB đặt từ địa chỉ đầu là 3C000h.

Bài tập

❖ Bản đồ địa chỉ:

$A_{19} \rightarrow A_0$ (HEX)		$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$
SRAM #1	Địa chỉ Đầu	3C000	0011	1100	0000	0000
	Địa chỉ cuối	3DFFF	0011	1101	1111	1111
SRAM #2	Địa chỉ Đầu	3E000	0011	1110	0000	0000
	Địa chỉ cuối	3FFFF	0011	1111	1111	1111

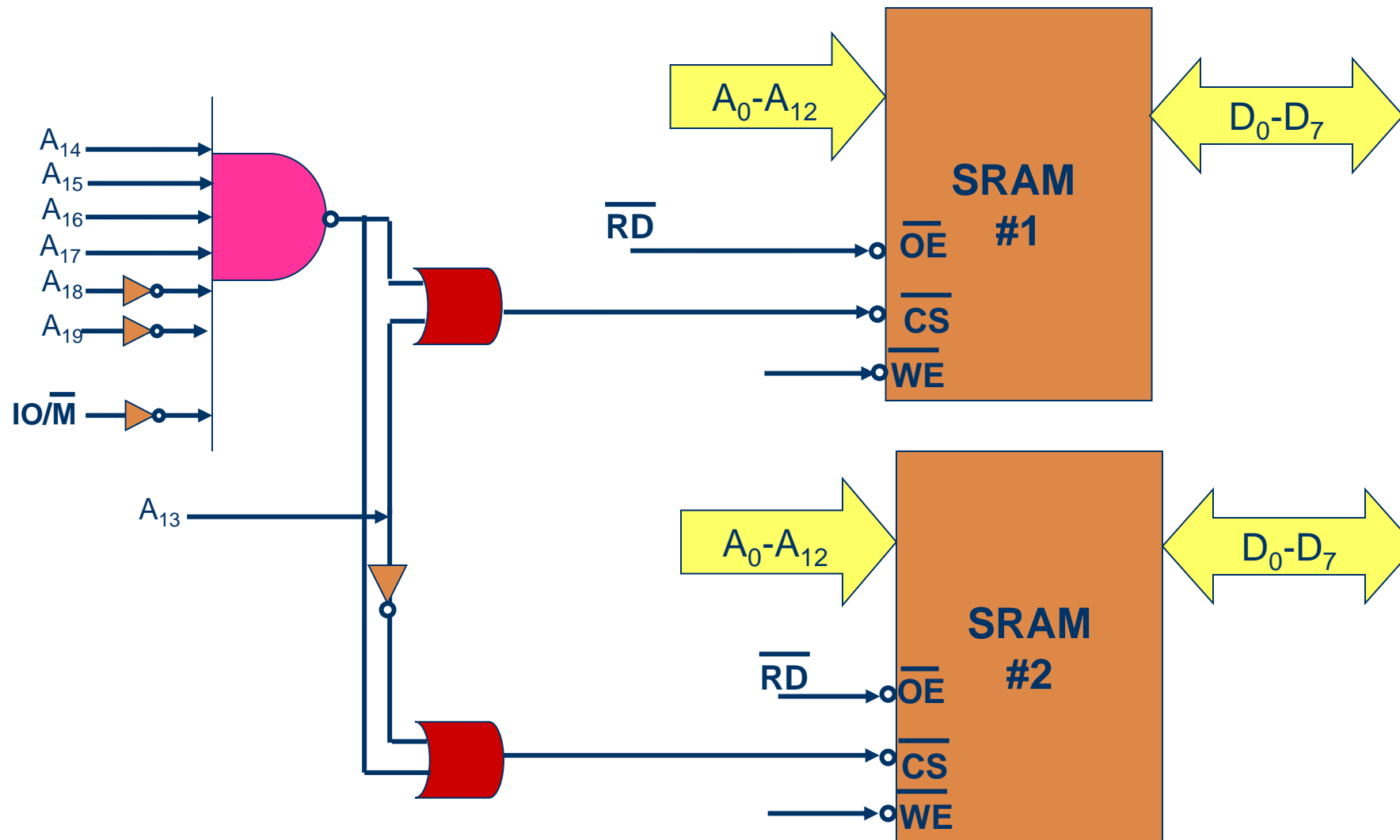
Phần địa chỉ không đổi

Phần địa chỉ thay đổi: $A_{12}-A_0$

$A_{13}=0$: SRAM#1

$A_{13}=1$: SRAM#2

Bài tập



Phương pháp dùng các vi mạch giải mã có sẵn

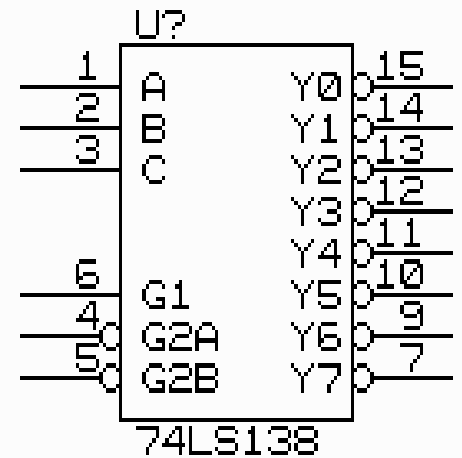
(74LS138, 74LS139 ...)

❖ Một số vi mạch được dùng nhiều trong các hệ thống giải mã của VXL:

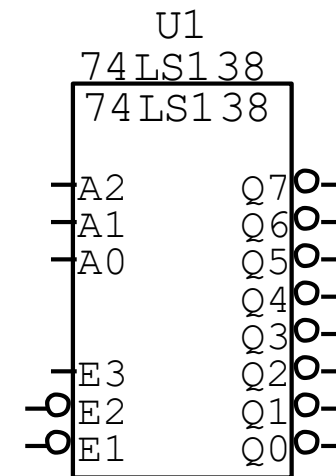
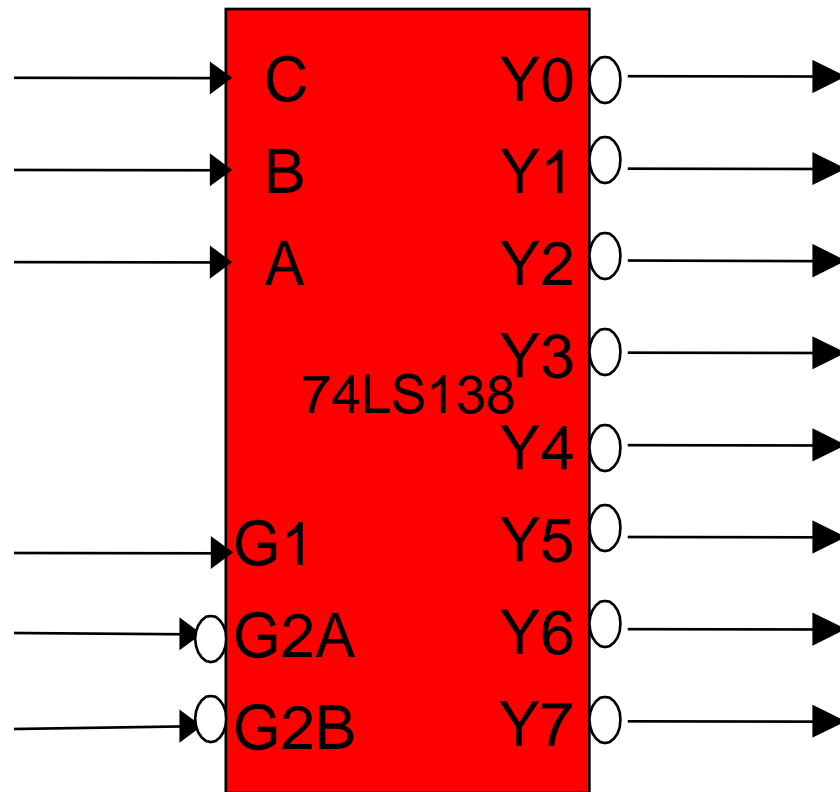
- 74LS138 (mạch giải mã 3-8)
- 74LS139 (mạch giải mã 2-4).

❖ Vi mạch 74LS138.

- Bảng sự thật của vi mạch cho biết tại một thời điểm chỉ có một trong tám đầu ra có mức logic 0.
- Điều kiện cần để có một trong các mức thấp ở đầu ra là các đầu vào G2B, G2A, G1 phải được kích hoạt:
 - G2B, G2A ở mức thấp và
 - G1 ở mức cao.
- Khi 74LS138 đã được kích hoạt, các đầu vào địa chỉ A, B và C sẽ lựa chọn đầu ra → cho phép có thể chọn được 1 trong 8 thiết bị nhớ khác nhau tại 1 thời điểm.



74LS138: Một chip giải mã 3-8



74LS138

❖ Bảng chân lý của 74LS138

Đầu vào						Các đầu ra							
Cho phép			Chọn										
G2B	G2A	G1	C	B	A	0	1	2	3	4	5	6	7
1	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	0	x	x	x	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

Ví dụ

- ❖ Cho các mạch nhớ EPROM 8KB (2764). Hãy dùng mạch giải mã 74LS138 để thực hiện dành riêng vùng nhớ 64KB có địa chỉ bắt đầu F0000h cho các EPROM 8KB.

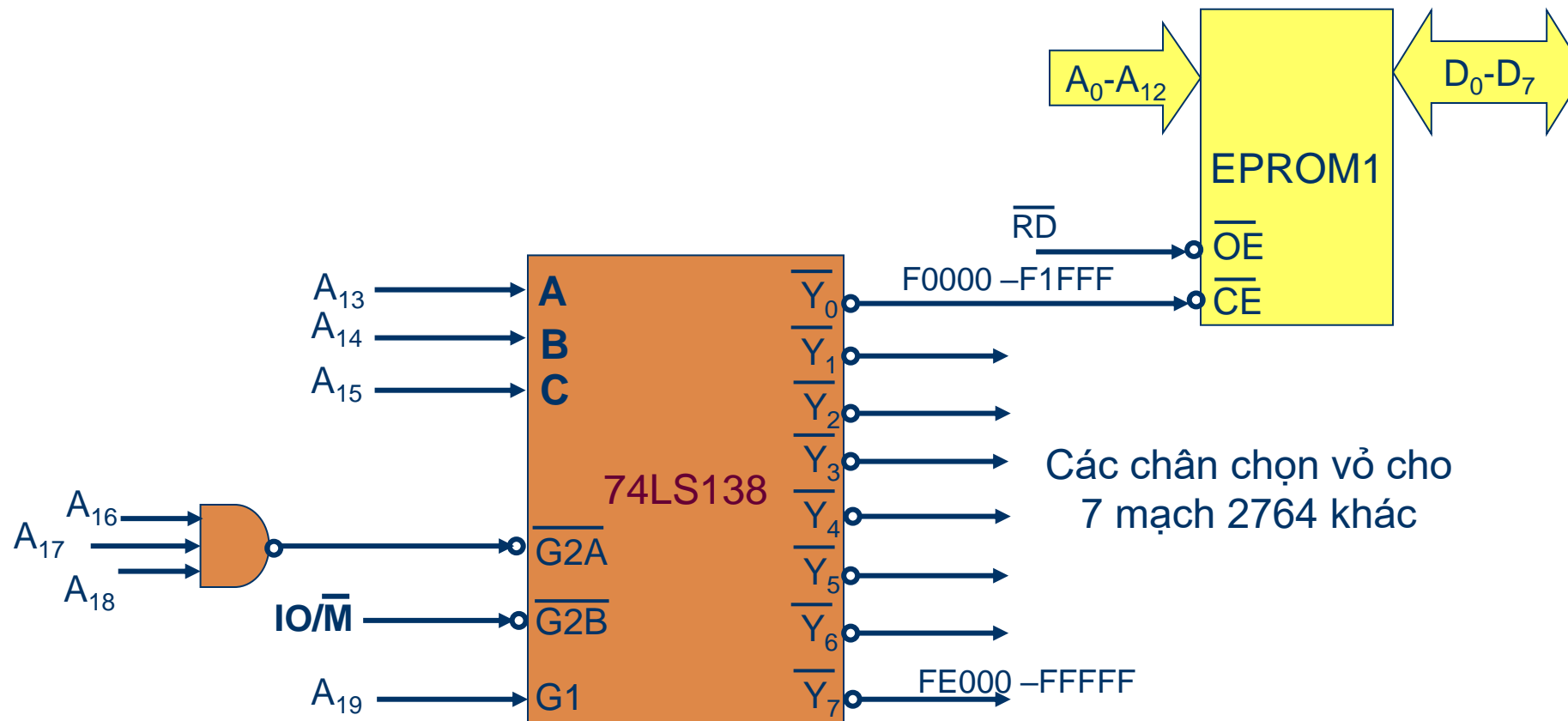
Ví dụ

❖ Cho các mạch nhớ EPROM 8K (2764). Hãy dùng mạch giải mã 74LS138 để thực hiện dành riêng vùng nhớ 64KB có địa chỉ bắt đầu F0000h.

❖ Giải:

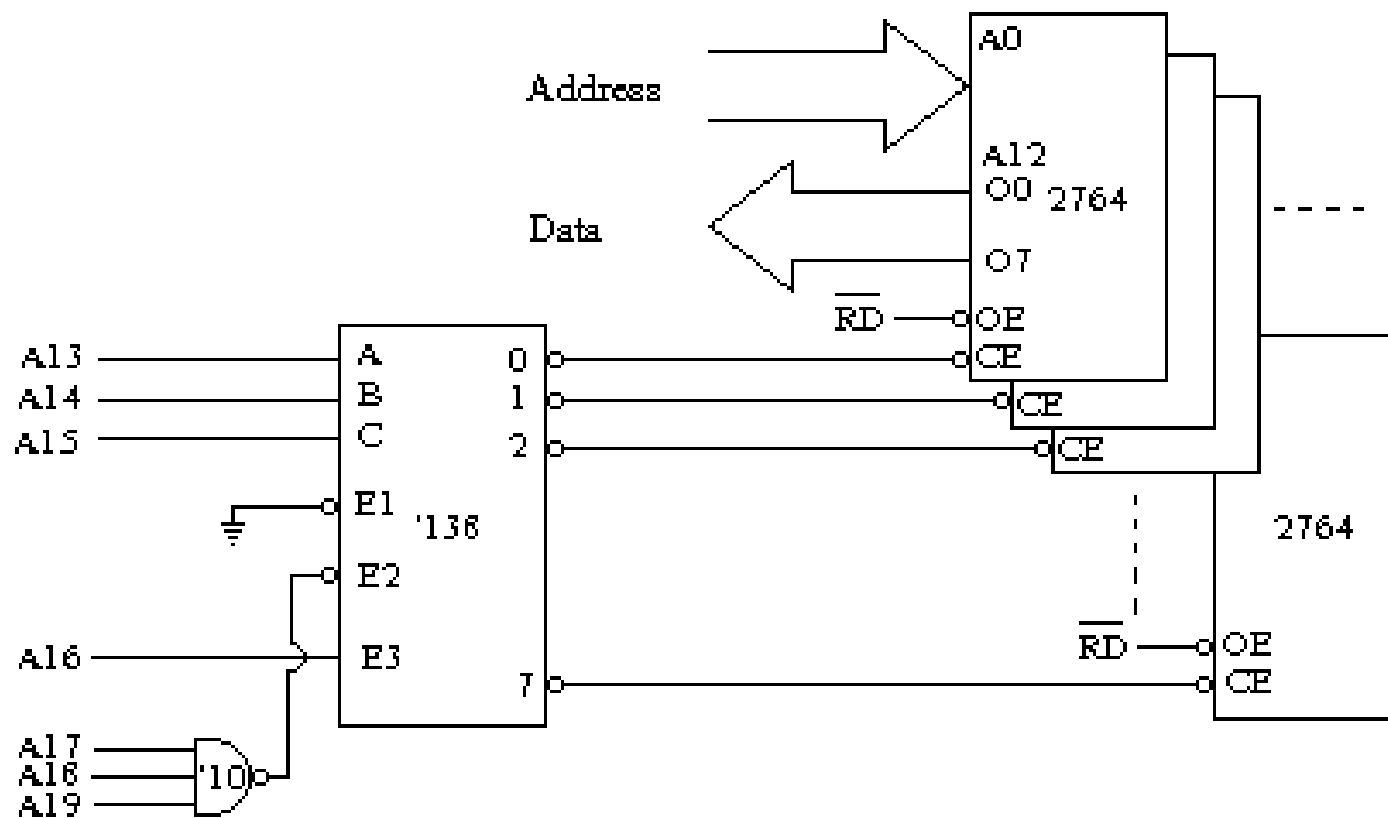
- EPROM 8K → 13 đường địa chỉ (A_0 - A_{12})
- Dùng 8 EPROM 8K
- → Dải địa chỉ F0000h – FFFFFh
- Bản đồ địa chỉ:
 - EPROM#1: F0000h → F1FFF h
 - EPROM#2: F2000h → F3FFF h
 - EPROM#3: F4000h → F5FFF h
 - EPROM#4: F6000h → F7FFF h
 - EPROM#5: F8000h → F9FFF h
 - EPROM#6: FA000h → FBFFF h
 - EPROM#7: FC000h → FDFFF h
 - EPROM#8: FE000h → FFFFF h

Ví dụ

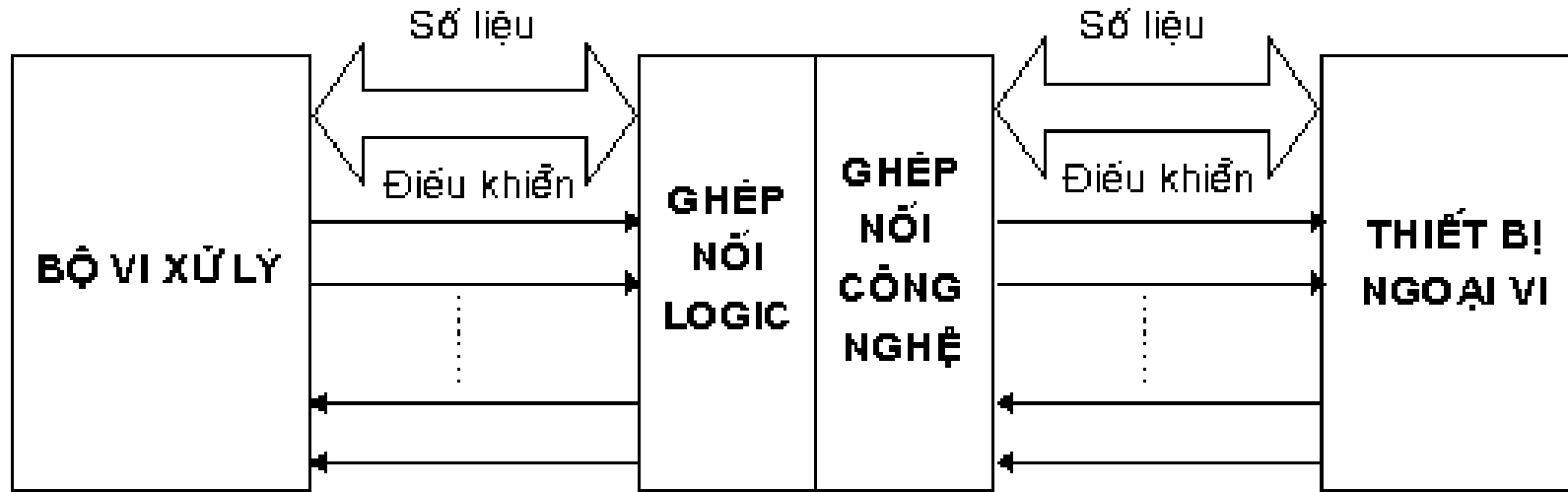


Ví dụ

❖ Ví dụ mạch giải mã chọn ROM dùng 74LS138



Phối ghép 8088 với thiết bị ngoại vi



- ❖ Thiết bị ngoại vi được ghép nối với BUS của hệ VXL thông qua các phần:
 - **Thích ứng công nghệ:**
 - Điều chỉnh mức tín hiệu giữa công nghệ sản xuất thiết bị ngoại vi và công nghệ sản xuất các mạch cấu trúc nên BUS của bộ VXL.
 - **Thích ứng về logic:**
 - Tạo các tín hiệu điều khiển thiết bị ngoại vi từ những tín hiệu trên BUS hệ thống.
 - Cấu trúc các mạch logic này gọi là mạch ghép nối (Interface).

Các kiểu ghép nối vào/ra

❖ Vào ra tách biệt:

- Thiết bị vào ra và bộ nhớ có địa chỉ tách biệt

❖ Vào ra theo địa chỉ bộ nhớ:

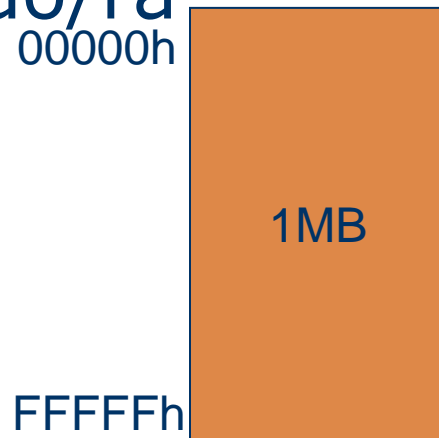
- Thiết bị vào ra và bộ nhớ có chung không gian địa chỉ

Vào ra tách biệt

- ❖ Hệ thống sẽ quản lý các thiết bị vào ra hoàn toàn độc lập với bộ nhớ.
 - Việc quản lý địa chỉ của các thiết bị vào/ra thông qua các địa chỉ cổng được giải mã hoàn toàn tách biệt với bộ nhớ.
- ❖ Người dùng có thể mở rộng không gian dành cho bộ nhớ lên đến 1MB (đối với vi mạch 8086/8088).
 - Tín hiệu IO/M được dùng để chỉ định BUS địa chỉ đang chứa **địa chỉ bộ nhớ** hay **địa chỉ cổng vào/ra**.
- ❖ Việc trao đổi dữ liệu giữa VXL với các thiết bị vào/ra thông qua các lệnh IN, OUT.

Vào ra tách biệt

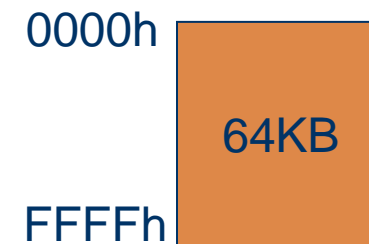
❖ Không gian riêng biệt cho toàn bộ nhớ và thiết bị vào/ra



Bộ nhớ



Thiết bị vào



Thiết bị ra

Lệnh kiểu:

MOV

IN

OUT

Tín hiệu điều khiển:

$IO/\overline{M} = 0$

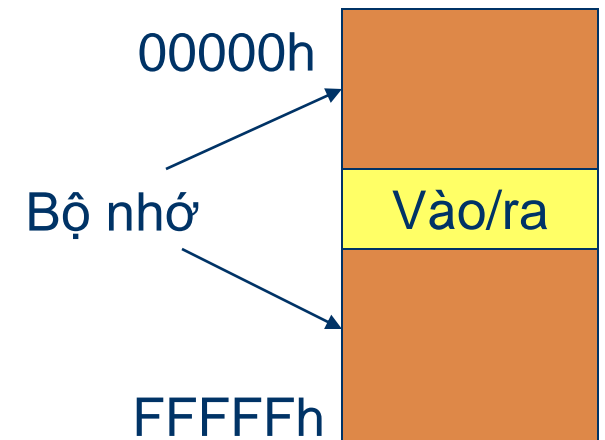
$IO/\overline{M} = 1$

$IO/\overline{M} = 1$

Vào ra theo địa chỉ bộ nhớ

- ❖ Có thể dùng bất kỳ một lệnh trao đổi dữ liệu nào giữa CPU với bộ nhớ.
- ❖ Thiết bị vào/ra theo địa chỉ bộ nhớ được xem như một vùng nhớ trong bản đồ bộ nhớ → làm giảm không gian nhớ của hệ thống dùng kỹ thuật vào/ra theo địa chỉ bộ nhớ.
- ❖ Điểm mạnh của kỹ thuật này:
 - không cần dùng đến tín hiệu IO/M để giải mã địa chỉ vì nó không dùng các lệnh IN, OUT.

Không gian chung cho bộ nhớ và thiết bị ngoại vi:



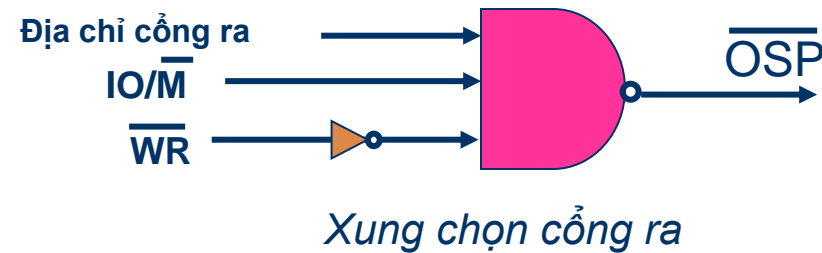
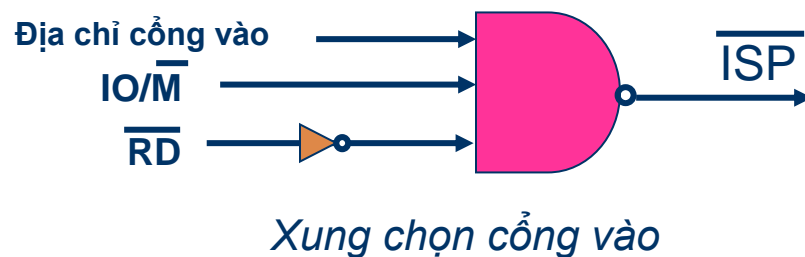
Kiểu lệnh và tín hiệu điều khiển chung cho cả 2:

MOV

$\text{IO}/\overline{\text{M}} = 0$

Giải mã địa chỉ cho thiết bị vào/ra

- ❖ Việc giải mã địa chỉ cho thiết bị ngoại vi cũng tương tự với việc giải mã địa chỉ cho bộ nhớ.
- ❖ Thông thường các cổng có địa chỉ 8 bit $A_0 - A_7$.
 - Trong một số hệ VXL, các cổng sẽ có địa chỉ 16 bit ($A_0 - A_{15}$).
- ❖ Giải mã địa chỉ cho thiết bị vào/ra:
 - Thực hiện bộ giải mã dùng mạch NAND
 - Thực hiện bộ giải mã dùng mạch giải mã 74LS138
- ❖ Ví dụ: Mạch giải mã đơn giản cho các thiết bị ngoại vi



Thực hiện bộ giải mã dùng mạch giải mã 74LS138

❖ Các bộ giải mã địa chỉ 00-07h cho:

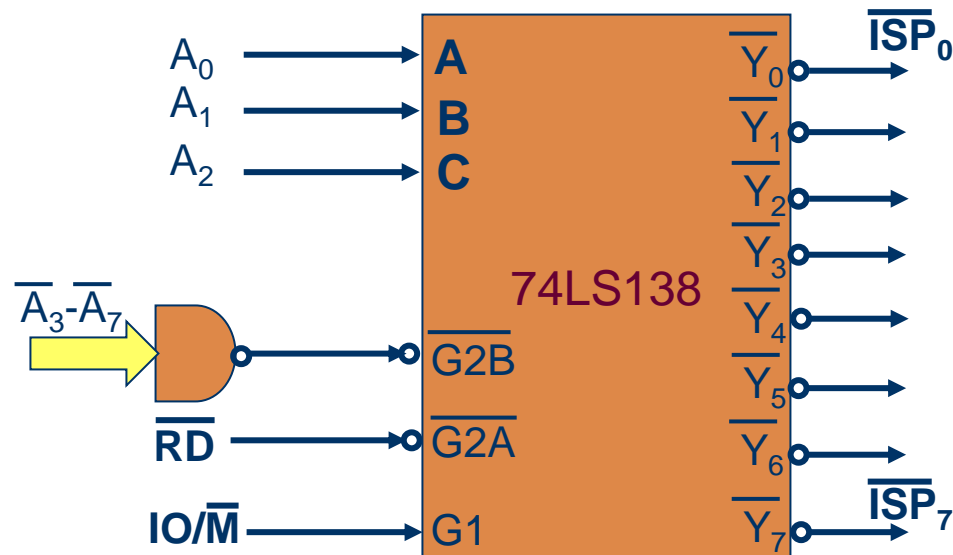
0000 0000

A_7-A_3 : các bit không đổi (luôn =0)

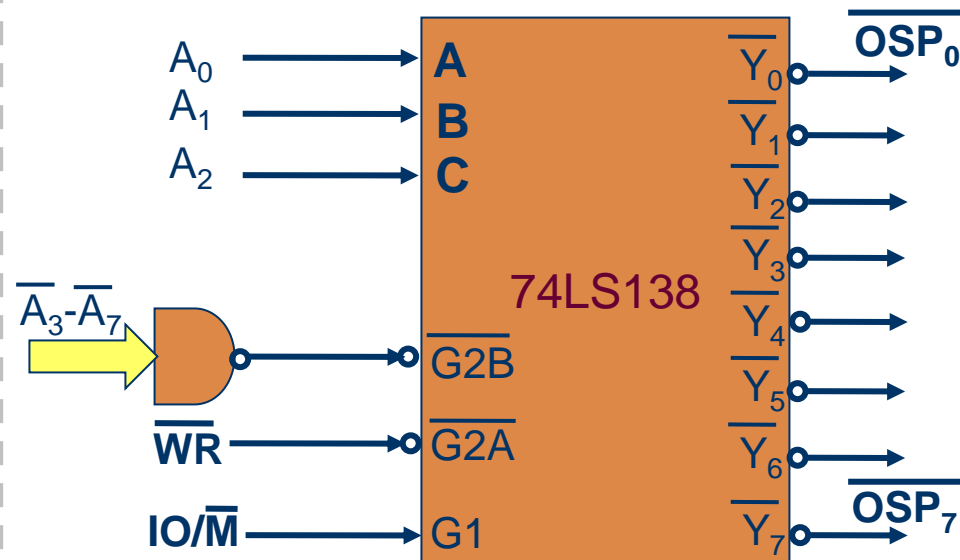
0000 0111

A_2-A_0 : các bit thay đổi

▪ a) Cổng vào



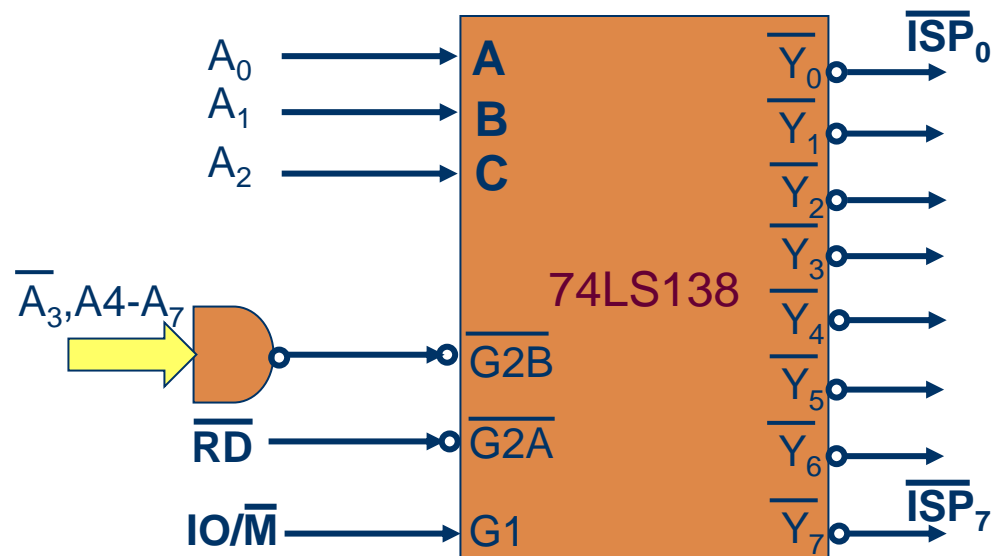
□ b) cổng ra.



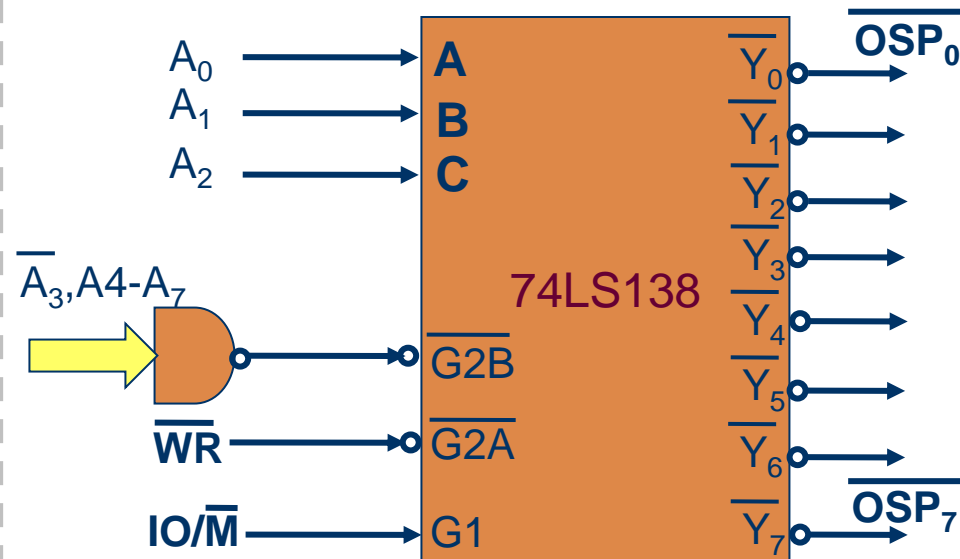
Thực hiện bộ giải mã dùng mạch giải mã 74LS138

❖ Các bộ giải mã địa chỉ F0-F7h cho:

▪ a) 8 cổng vào



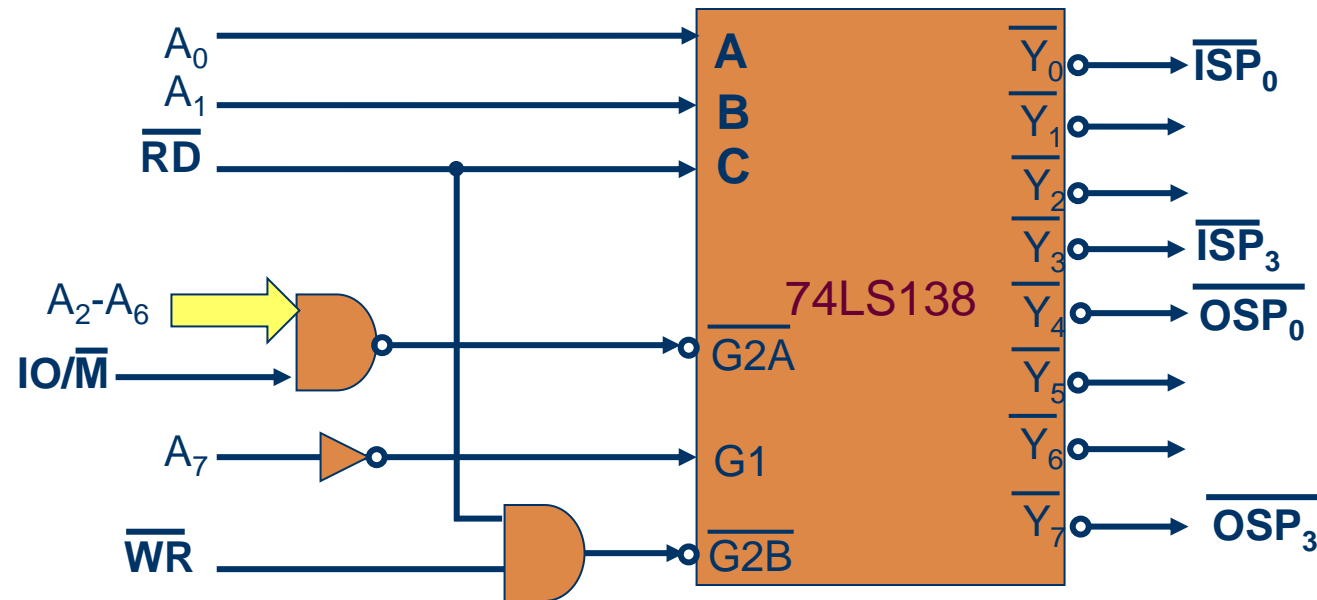
□ b) 8 cổng ra.



Thực hiện bộ giải mã dùng mạch giải mã 74LS138

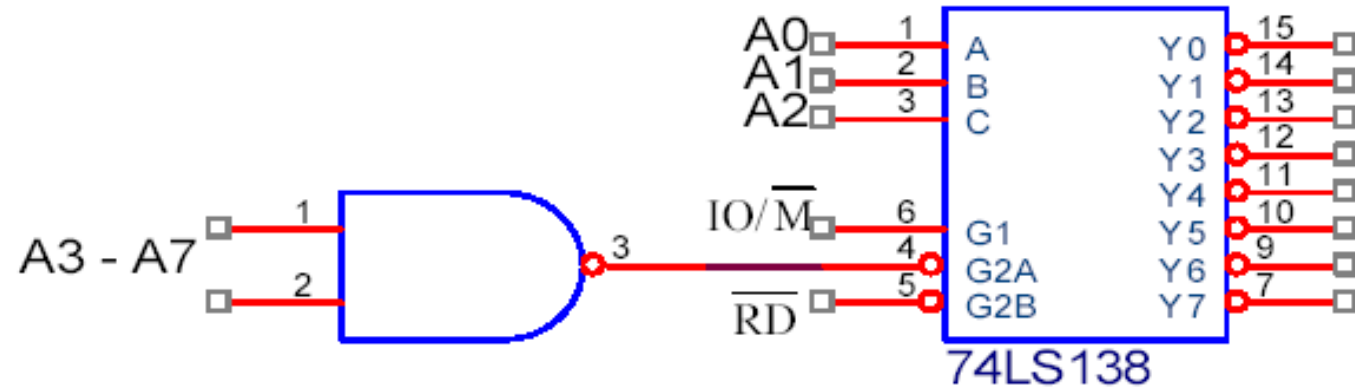
❖ Một bộ giải mã cho 4 cổng vào và cổng ra với địa chỉ 7Ch-7Fh cho:

- 7Ch: 0111 1100
- 7Fh: 0111 1111

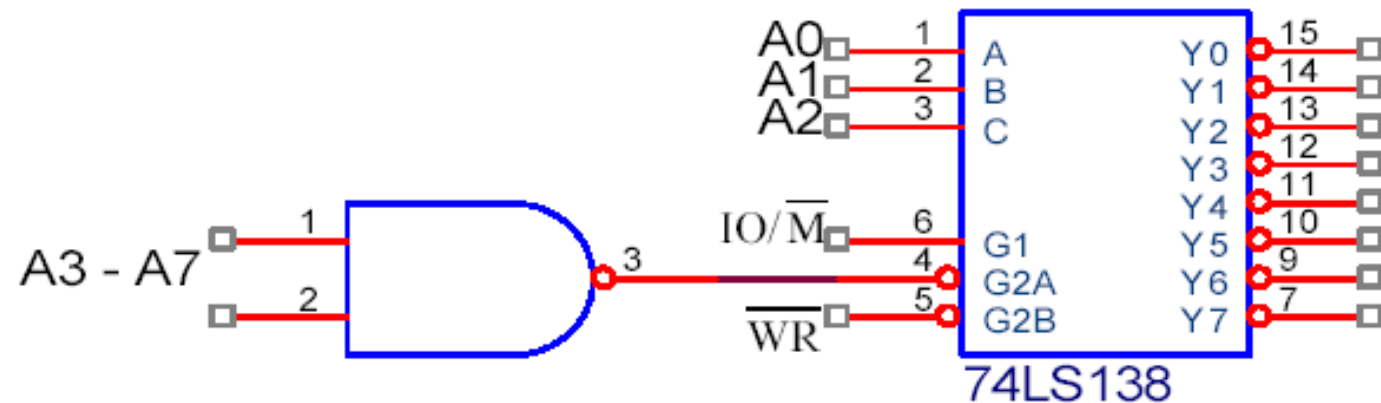


- 2 bit chọn cổng A,B
- Bit chọn cổng vào/ra

Giải mã cho các cổng



(a) Giải mã cho cổng vào



(b) Giải mã cho cổng ra

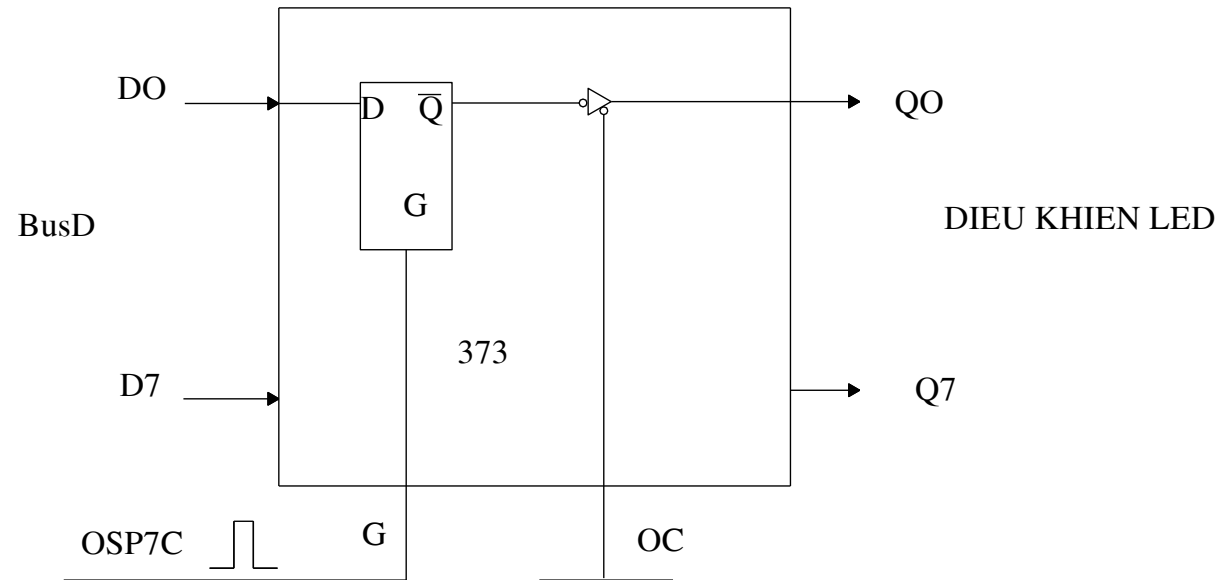
Các mạch cổng đơn giản

- ❖ Các mạch cổng có thể được xây dựng từ:
 - Các mạch chốt 8 bit
 - 74LS373: kích theo mức
 - 74LS374: kích theo cạnh
 - Các mạch đếm 8 bit (74LS245).
- ❖ Được dùng trong các giao tiếp đơn giản để μP và ngoại vi hoạt động tương thích với nhau
 - Ví dụ: để làm đếm bus hoặc các mạch cổng để tạo ra các tín hiệu móc nối.

Các mạch cổng đơn giản

❖ Ví dụ: Một mạch phối ghép đơn giản dùng mạch 74LS373 để đưa tín hiệu ra điều khiển các đèn LED bằng lệnh:

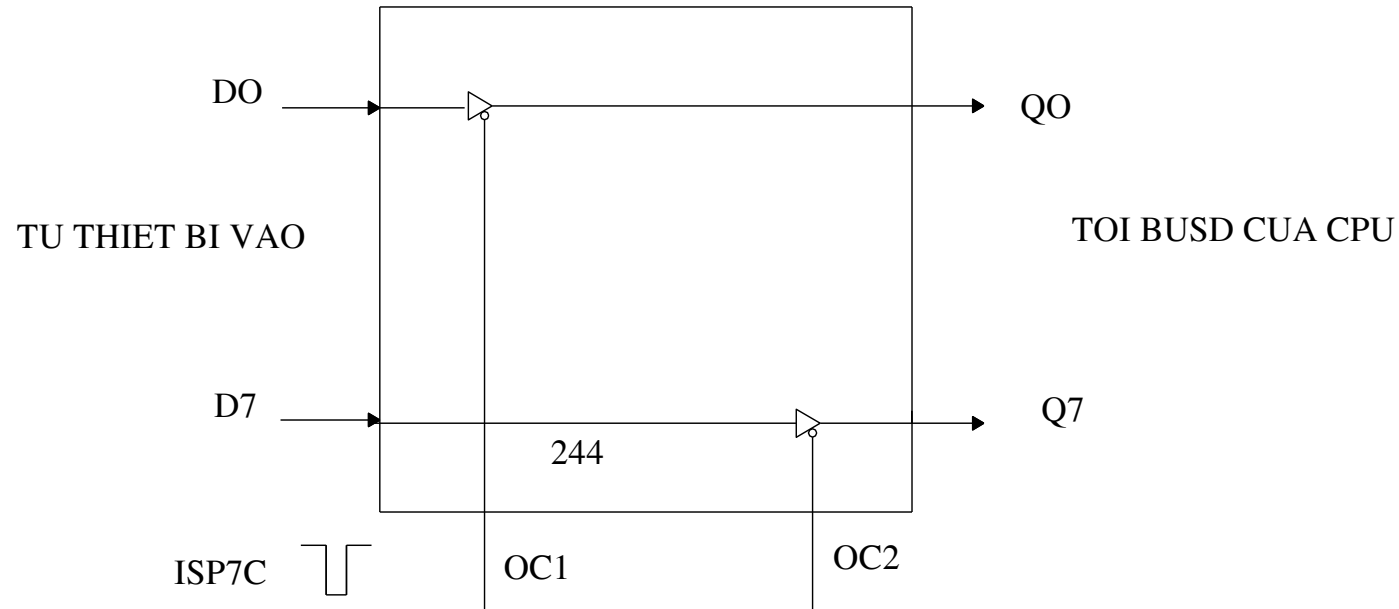
■ OUT 7Ch,AL



Các mạch cổng đơn giản

- ❖ Ví dụ: Một mạch phối ghép đơn giản dùng mạch 74LS244 để đọc tín hiệu từ thiết bị ngoại vi vào CPU bằng lệnh:

IN AL, 7Ch

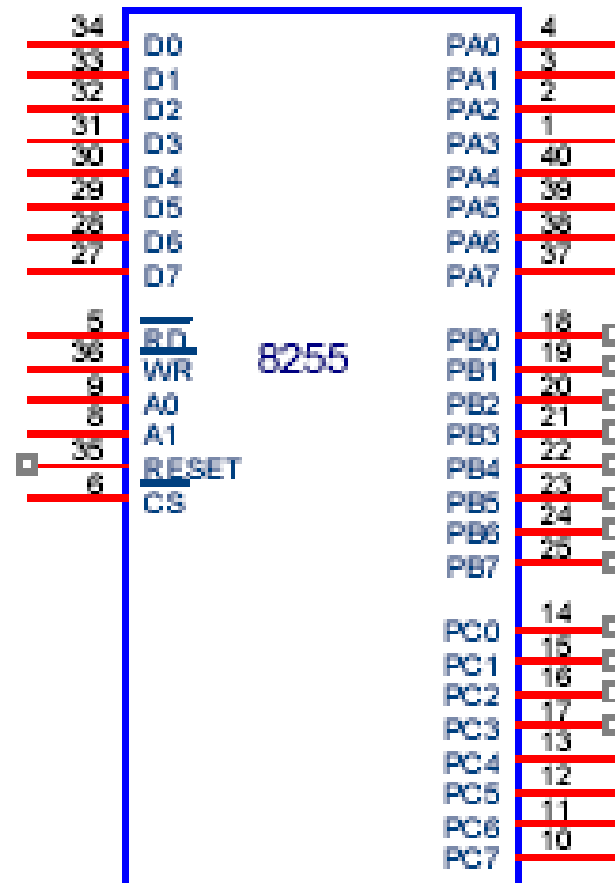


Mạch phối ghép vào/ra song song lập trình được 8255A

- ❖ 8255A là thiết bị vào ra song song lập trình được (**PPI-Programmable Peripheral Interface**).
 - Là một thiết bị I/O đa dụng có thể sử dụng với bất cứ μP nào
 - Có thể lập trình để truyền dữ liệu, từ I/O thông thường đến I/O interrupt.
- ❖ 8255A có thể chia thành 3 Port: (A, B, C)
 - Mỗi port A,B,C: 8 bit
 - Port C có thể sử dụng như 8 bit riêng hay chia thành 2 nhóm, mỗi nhóm 4 bit: PC_H ($PC_7 - PC_4$) và PC_L ($PC_3 - PC_0$).

Mạch phối ghép vào/ra song song lập trình được 8255A

❖ Sơ đồ chân của 8255A



D7 – D0: bus dữ liệu

PA7 – PA0: Port A

PB7 – PB0: Port B

PC7 – PC0: Port C

A1, A0: giải mã

\overline{RESET} : ngõ vào Reset

CS: Chip Select

\overline{RD} : Read

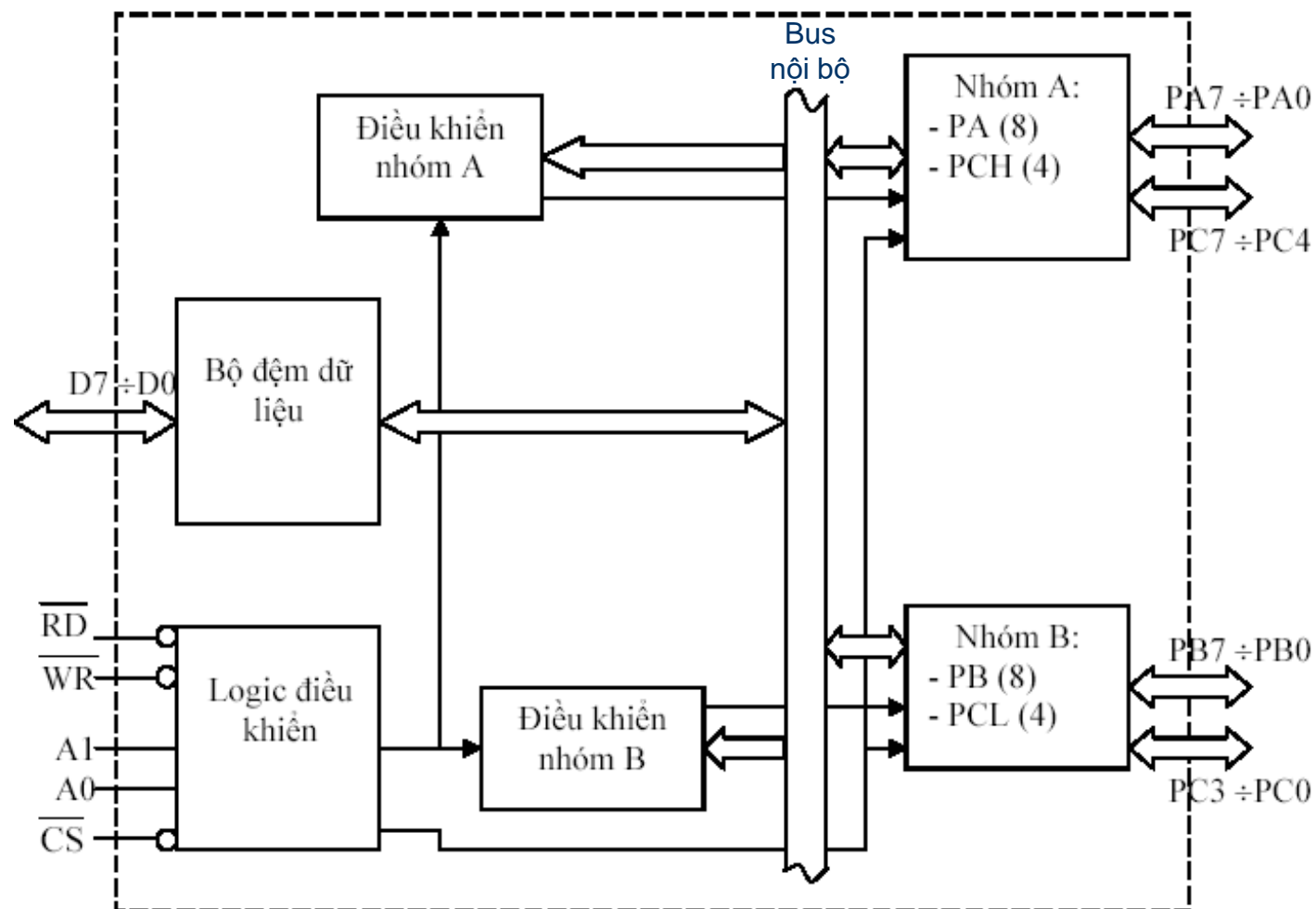
WR: Write

VCC: +5V

GND: 0V

Mạch phối ghép vào/ra song song lập trình được 8255A

❖ Sơ đồ khối của 8255



Mạch phối ghép vào/ra song song lập trình được 8255A

❖ Sơ đồ khối của 8255

- Khối logic điều khiển đọc ghi của 8255A gồm có 6 đường:
 - \overline{RD} (Read): cho phép đọc.
 - Khi chân này ở mức thấp thì cho phép đọc dữ liệu từ Port I/O đã chọn.
 - \overline{WR} (Write): cho phép ghi.
 - Khi chân này ở mức thấp thì cho phép ghi dữ liệu ra Port I/O đã chọn.
 - RESET:
 - Được nối với tín hiệu RESET chung của toàn hệ.
 - Khi chân này ở mức cao thì sẽ xoá thanh ghi điều khiển và đặt các Port ở chế độ vào.
 - \overline{CS} (Chip Select): chân chọn vi mạch
 - Được nối với mạch tạo xung chọn thiết bị
 - Đặt mạch 8255A vào một địa chỉ cơ sở nào đó.
 - A_1, A_0 : các tín hiệu địa chỉ kết hợp với tín hiệu \overline{CS} chọn ra 4 thanh ghi (cổng) bên trong 8255A.

Mạch phối ghép vào/ra song song lập trình được 8255A

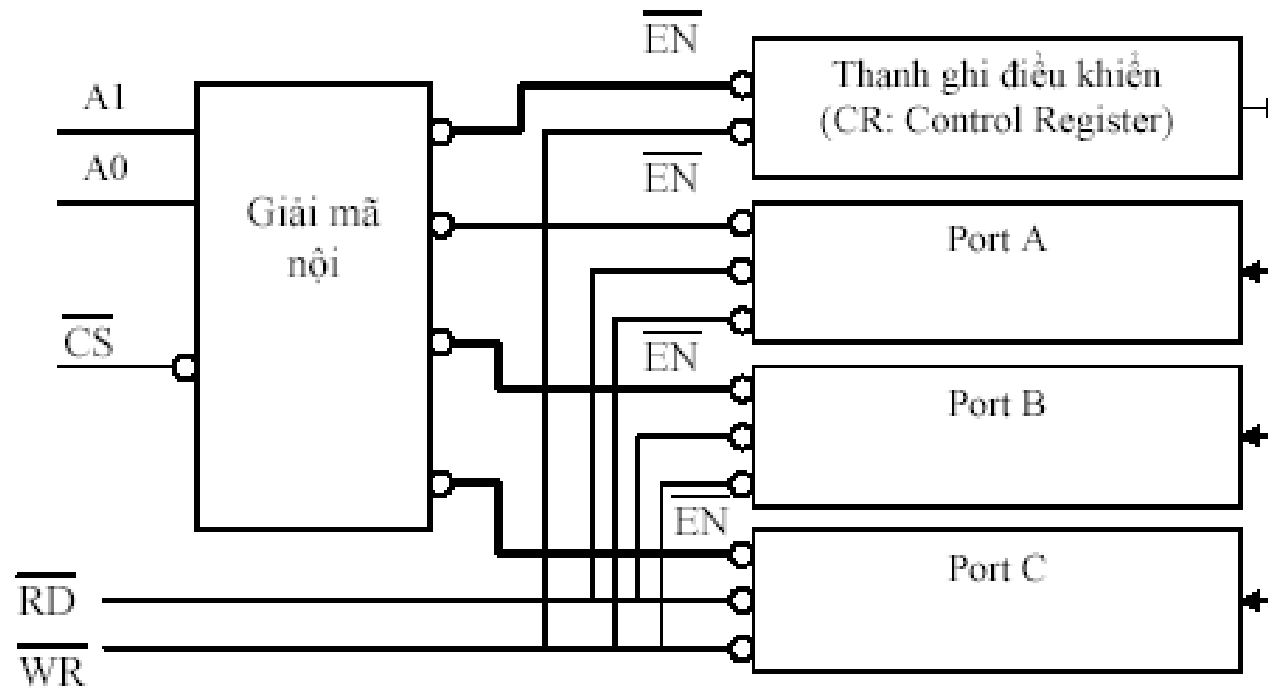
❖ Sơ đồ khối của 8255

- Khối logic điều khiển của 8255A gồm có 6 đường:
 - \overline{RD}
 - \overline{WR}
 - RESET
 - \overline{CS}
 - A₁, A₀: các tín hiệu địa chỉ kết hợp với tín hiệu \overline{CS} chọn ra 4 thanh ghi (cổng) bên trong 8255A.

\overline{CS}	A ₁	A ₀	Chọn ra
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Thanh ghi điều khiển (CR)
1	x	x	8255A không hoạt động (không chọn)

Mạch phối ghép vào/ra song song lập trình được 8255A

- ❖ A1, A0: các tín hiệu địa chỉ kết hợp với tín hiệu chọn ra 4 thanh ghi (cổng) bên trong 8255A.



Các chế độ hoạt động của 8255A

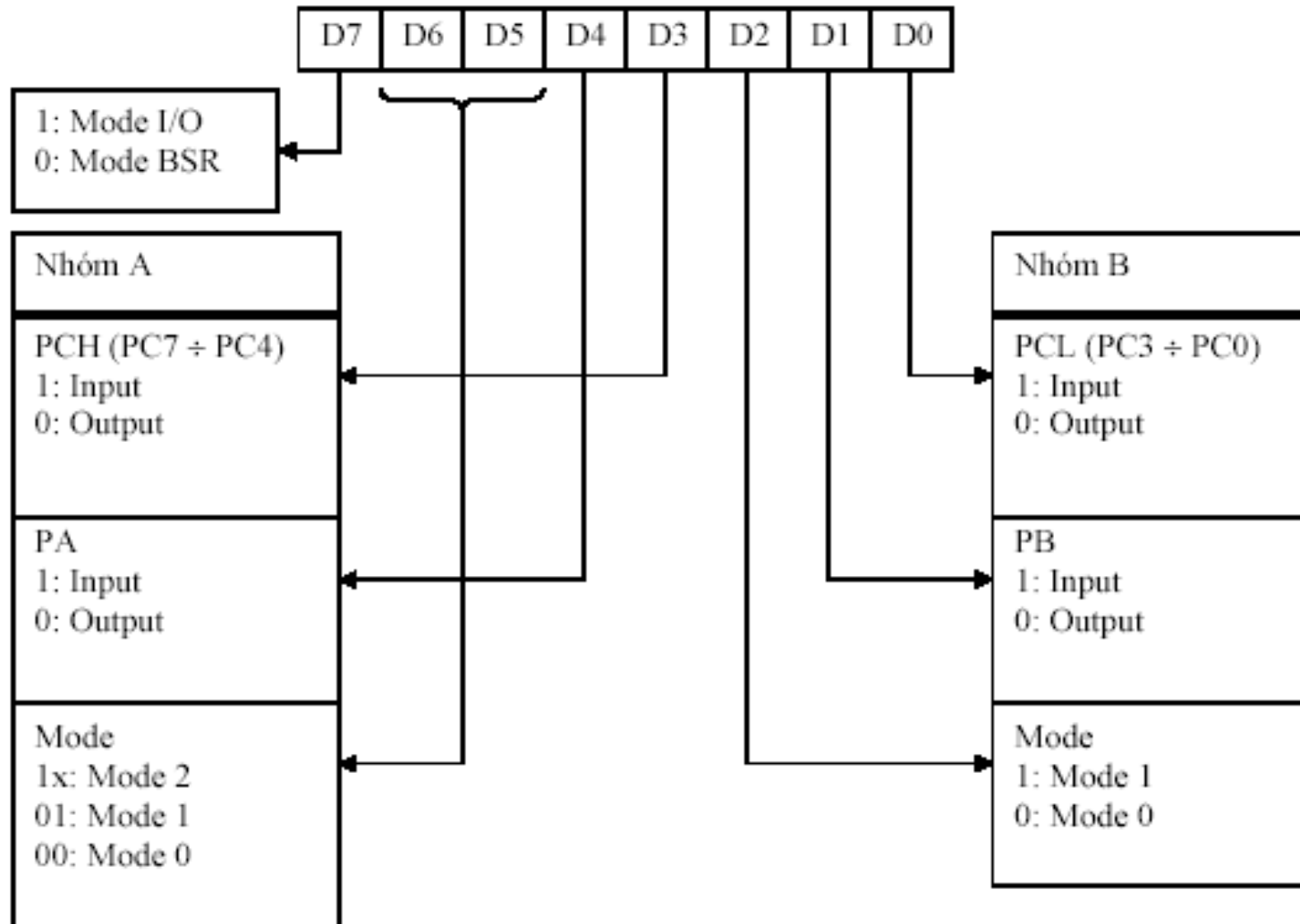
❖ 8255A có thể hoạt động ở 2 chế độ (mode):

- **Chế độ BSR:** dùng để lập hay xóa các bit của Port C.
- **Chế độ I/O:** gồm có 3 chế độ:
 - Chế độ 0: (vào/ra đơn giản)
 - Tất cả các Port làm việc như các Port I/O đơn giản: PA, PB, PC_H, PC_L đều có thể được định nghĩa là cổng vào hoặc ra.
 - Chế độ 1: (vào/ra có xung cho phép)
 - Các Port A, B dùng các bit của Port C làm tín hiệu bắt tay (handshake)
 - Các kiểu truyền dữ liệu I/O có thể được cài đặt, kiểm tra trạng thái và ngắt.
 - Chế độ 2: (vào/ra 2 chiều)
 - Port A có thể dùng để truyền dữ liệu song hướng dùng các tín hiệu bắt tay từ Port C còn Port B được thiết lập ở chế độ 0 hay 1.

Thanh ghi điều khiển

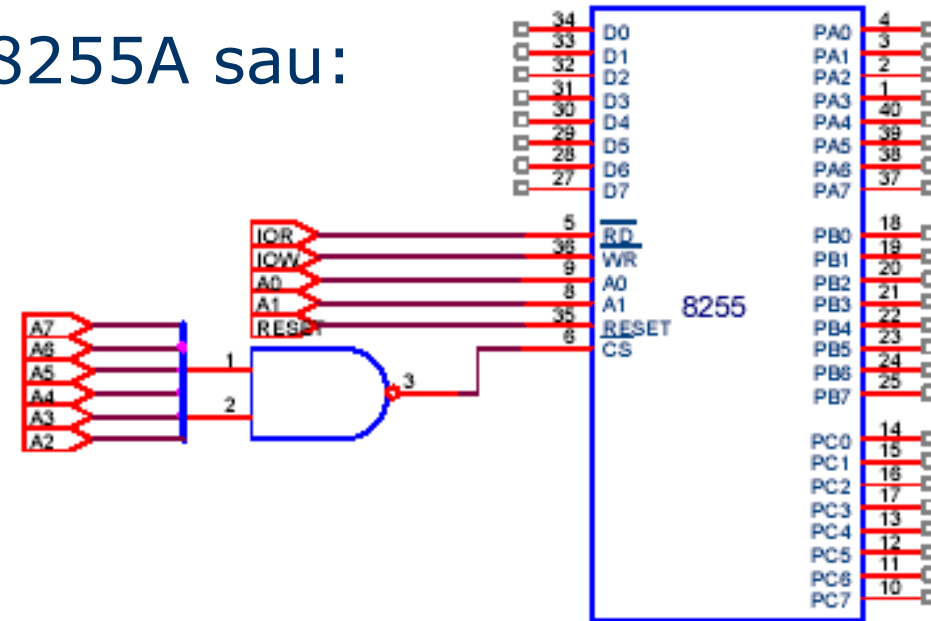
- ❖ 8255A có các chế độ hoạt động khác nhau và các Port của nó có thể có các chức năng I/O khác nhau.
- ❖ Để xác định chức năng của các Port, 8255A có 1 thanh ghi điều khiển (CR: Control Register).
 - Nội dung của thanh ghi CR ghi từ điều khiển (CW: Control Word) cho hoạt động của 8255A.
 - $D_7=1$: chế độ I/O
 - $D_7=0$: chế độ BSR
 - Từ điều khiển BSR không ảnh hưởng đến chức năng các Port A, B.
- ❖ Thanh ghi điều khiển sẽ được truy xuất khi $A_1 = A_0 = 1$.
 - Không thể thực hiện tác vụ đọc đối với thanh ghi này

Dạng từ điều khiển cho 8255A ở chế độ I/O



8255A

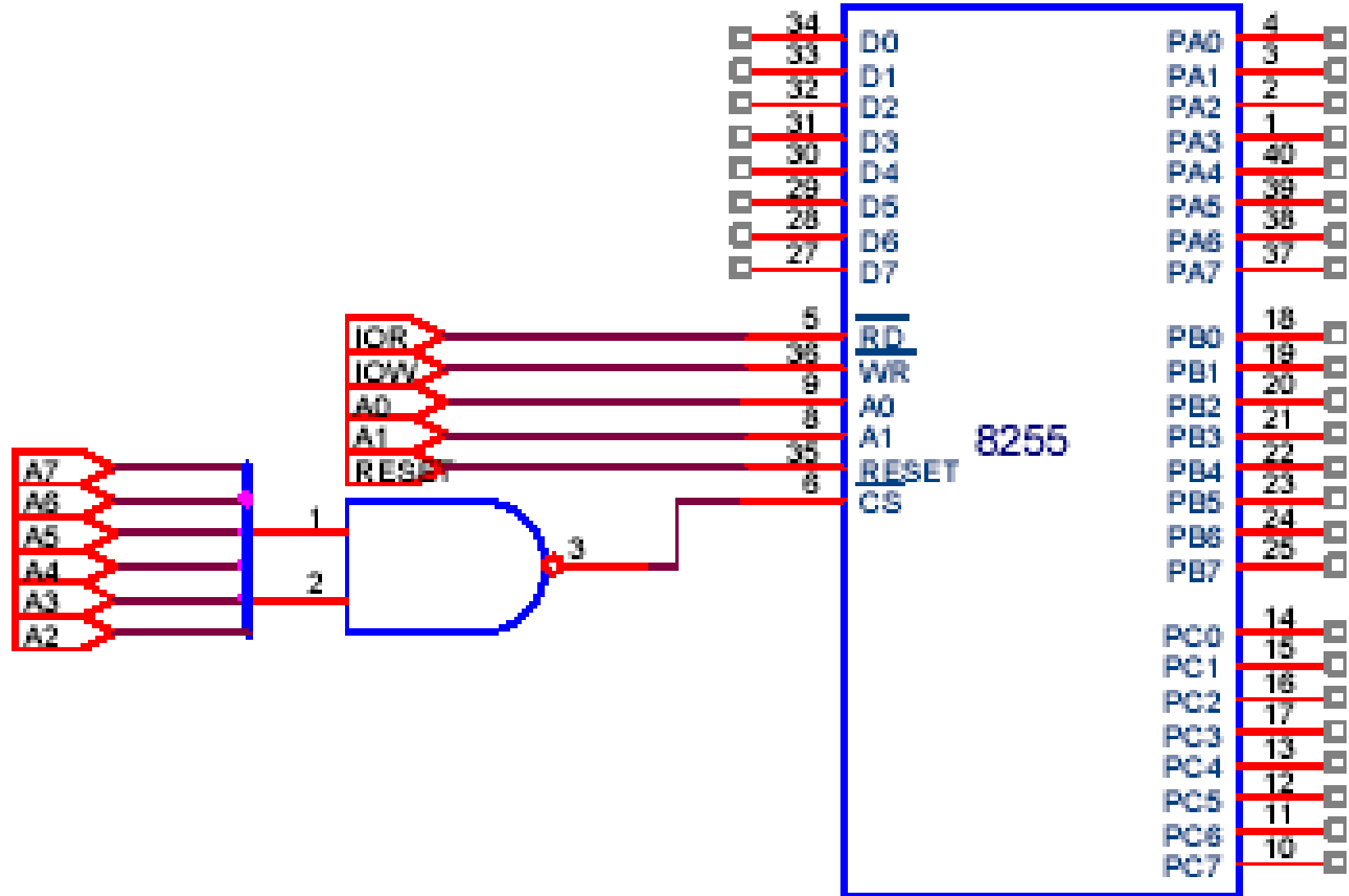
❖ **Ví dụ:** Xét sơ đồ kết nối 8255A sau:



- Để chọn Port A, ta phải có: $\begin{cases} CS=0 \\ A_1=0 \\ A_0=0 \end{cases}$
- Mà $CS = 0$ khi $A_7 = A_6 = A_5 = A_4 = A_3 = A_2 = 1 \rightarrow$ được địa chỉ Port I/O:

CS						A1 A0		Port	Địa chỉ hex
A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A1	A0		
1	1	1	1	1	1	0	0	A	FCh
						0	1	B	FDh
						1	0	C	FEh
						1	1	CR	FFh

8255A

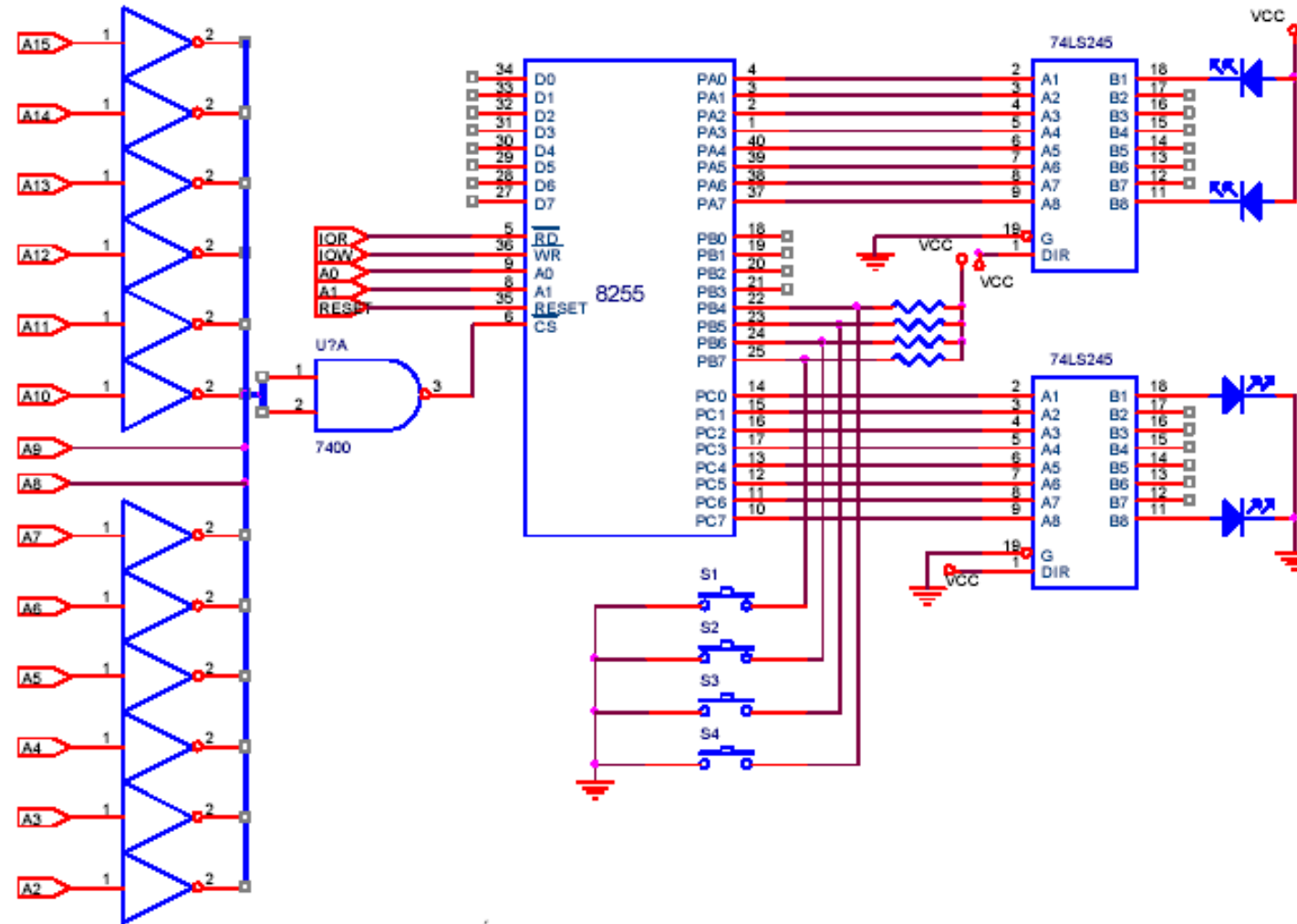


Mode 0: vào/ra đơn giản

- ❖ Trong chế độ này, mỗi port (hay nửa port của Port C) làm việc như các port vào hay ra với các tính chất:
 - Các cổng ra được chốt.
 - Các cổng vào không được chốt.
 - Các port không có khả năng bắt tay và ngắt.
- ❖ Để giao tiếp với ngoại vi thông qua 8255A cần phải:
 - Xác định địa chỉ của các port A, B, C và CR thông qua các chân chọn chip CS và giải mã A_1, A_0 .
 - Ghi từ điều khiển vào thanh ghi điều khiển.
 - Ghi các lệnh I/O để giao tiếp với ngoại vi qua các port A, B, C.

Mode 0: vào/ra đơn giản

❖ Ví dụ: Xét sơ đồ kết nối 8255A như sau:



Ví dụ (tiếp)

❖ Giao tiếp các port 8255A ở mode 0

■ Xác định địa chỉ port:

CS														A1 A0	Port	Địa chỉ hex
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	A1 A0		
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0 0	A	300h
														0 1	B	301h
														1 0	C	302h
														1 1	CR	303h

■ Từ điều khiển

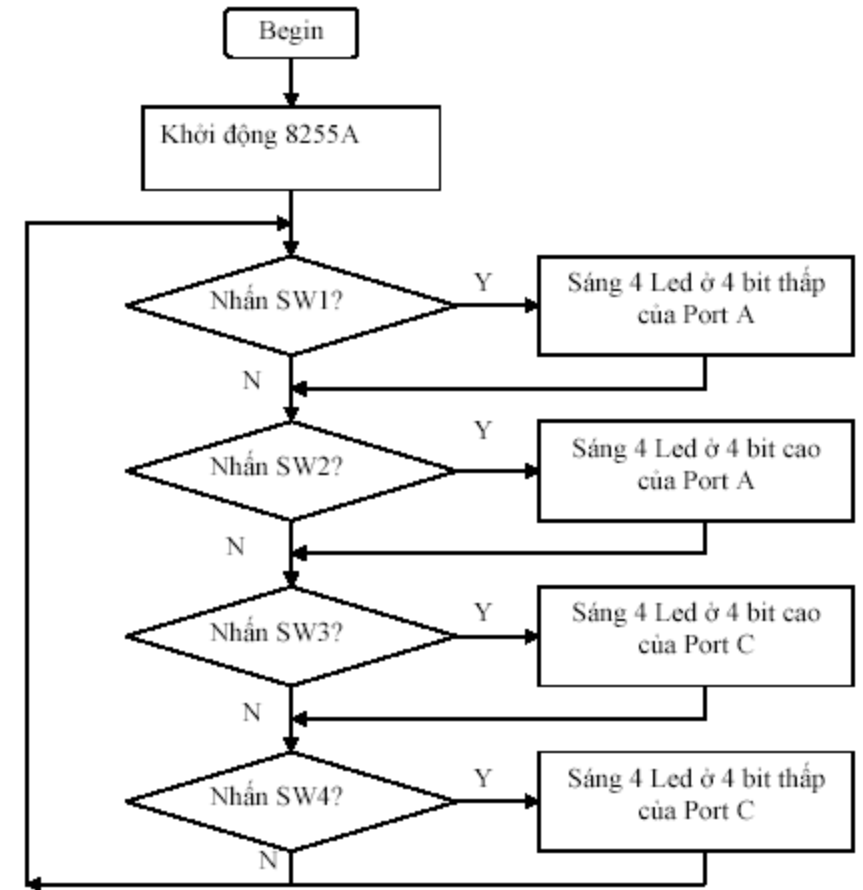
D7	D6 D5	D4	D3	D2	D1	D0
1	0 0	0	0	0	1	0
I/O mode	Nhóm A ở mode 0	PA: Output	PC _H : Output	Nhóm B ở mode 0	PB: Input	PC _L : Output

= 82h

Ví dụ (tiếp)

- ❖ Các Port của 8255A được khởi động bằng cách đặt từ điều khiển 82h vào thanh ghi điều khiển.
- ❖ Trong sơ đồ kết nối này:
 - 4 bit cao của Port B dùng làm Port vào
 - Port A và Port C làm Port ra
 - Các tác vụ Đọc và Ghi được phân biệt bằng các tín hiệu điều khiển IOR và IOW.

Lưu đồ giải thuật:



Ví dụ (tiếp): Chương trình

```
.MODEL SMALL
.STACK 100h
.CODE
    main PROC
        ; Định cấu hình cho 8255
        MOV AL,82h ; Từ điều khiển (CW) là 82h
        MOV DX,303h ; Địa chỉ thanh ghi điều
        khiển (CR)
        OUT DX,AL
        ; Ghi CW vào CR

    cont:
        MOV DX,301h ; Địa chỉ Port B
        IN AL,DX
        ; Đọc dữ liệu từ Port B (công tắc)
        AND AL,0F0h ; Che 4 bit thấp
        MOV AH,AL
        CMP AH,01110000b
        ; Kiểm tra công tắc 1
        JNE notSW1 ; Nếu không nhấn
        MOV AL,0Fh ; Nếu nhấn công tắc 1 thì
        MOV DX,300h ; xuất ra Port A
        OUT DX,AL
        ; để sáng 4 Led ở 4 bit thấp (Port A)
```

```
notSW1:
    CMP AH,10110000b ; Kiểm tra công tắc 2
    JNE notSW2 ; Nếu không nhấn
    MOV AL,0F0h ; Nếu nhấn công tắc 2 thì
    MOV DX,300h ; xuất ra Port A
    OUT DX,AL ; để sáng 4 Led ở 4 bit cao (Port A)

notSW2:
    CMP AH,11010000b ; Kiểm tra công tắc 3
    JNE notSW3 ; Nếu không nhấn
    MOV AL,0Fh ; Nếu nhấn công tắc 3 thì
    MOV DX,302h ; xuất ra Port C
    OUT DX,AL ; để sáng 4 Led ở 4 bit cao (Port C)

notSW3:
    CMP AH,11100000b ; Kiểm tra công tắc 4
    JNE notSW4 ; Nếu không nhấn
    MOV AL,F0h ; Nếu nhấn công tắc 4 thì
    MOV DX,302h ; xuất ra Port C
    OUT DX,AL ; để sáng 4 Led ở 4 bit thấp (Port C)

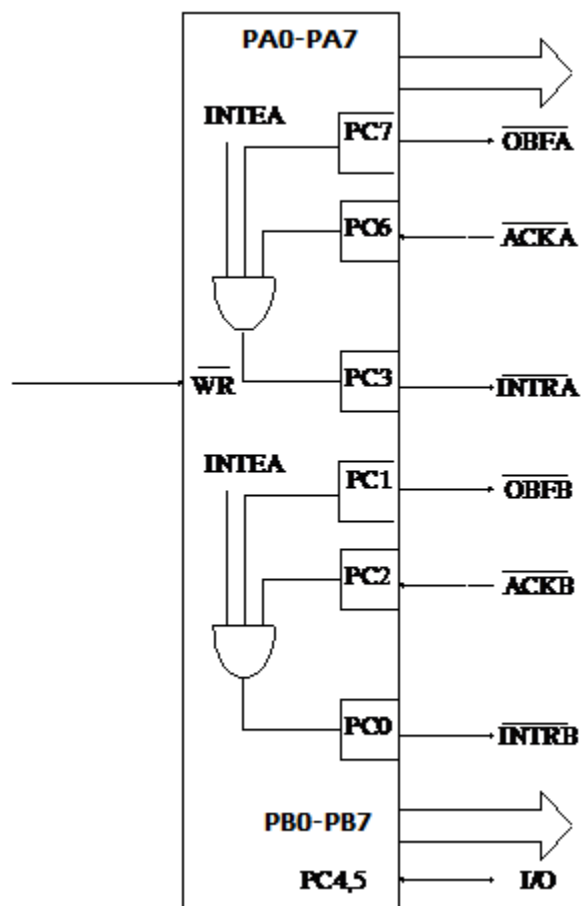
notSW4:
    cont
main ENDP
END main
```

Mode 1: vào/ra với xung cho phép

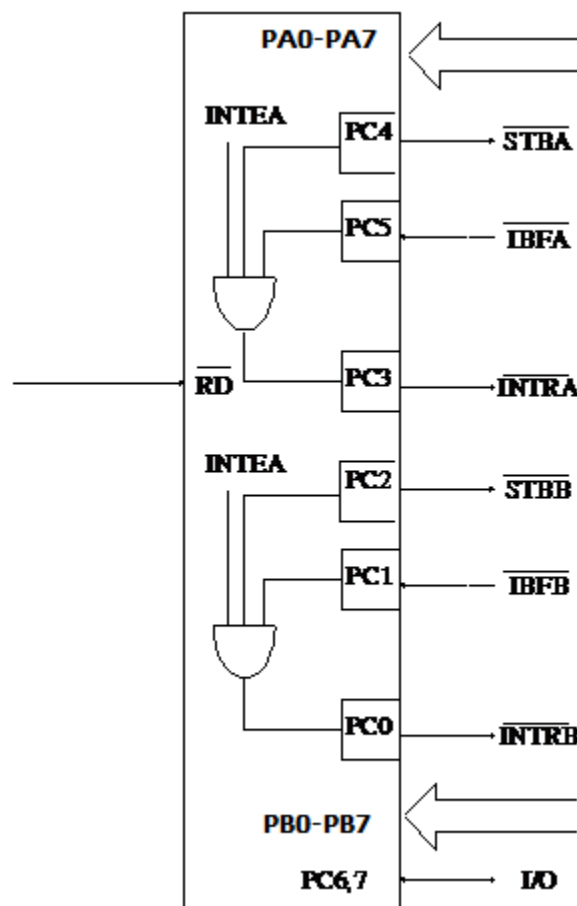
- ❖ Các tín hiệu bắt tay được trao đổi giữa μP và thiết bị ngoại vi trước khi truyền dữ liệu.
- ❖ Các đặc tính ở chế độ 1:
 - Hai Port A, B làm việc như các Port I/O 8 bit.
 - Mỗi Port sử dụng 3 đường từ Port C làm các tín hiệu bắt tay.
 - 2 đường còn lại của Port C có thể dùng cho các chức năng I/O đơn giản.
 - Dữ liệu đọc/ghi được chốt.
 - Hỗ trợ ngắt.

Mode 1: vào/ra với xung cho phép

❖ Mạch 8255A ở chế độ 1



Ra



Vào

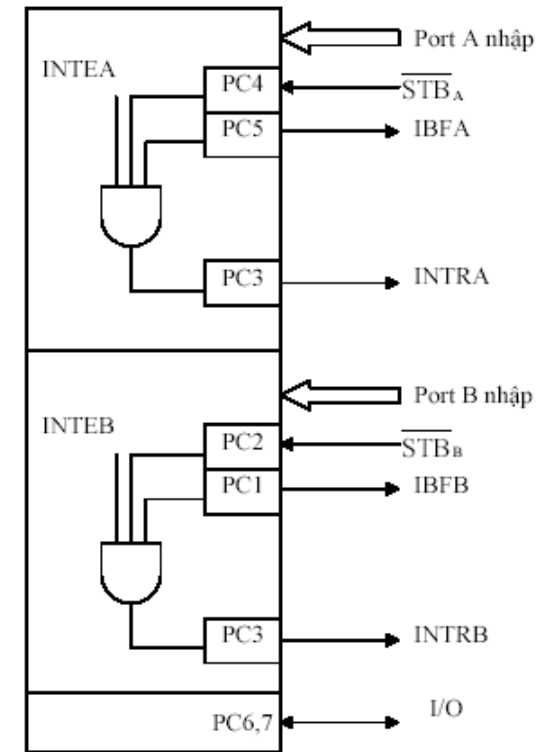
Vào dữ liệu trong chế độ 1

❖ Tín hiệu bắt tay:

- PC3, PC4 và PC5 → Port A
- PC0, PC1 và PC2 → Port B

❖ Các tín hiệu móc nối:

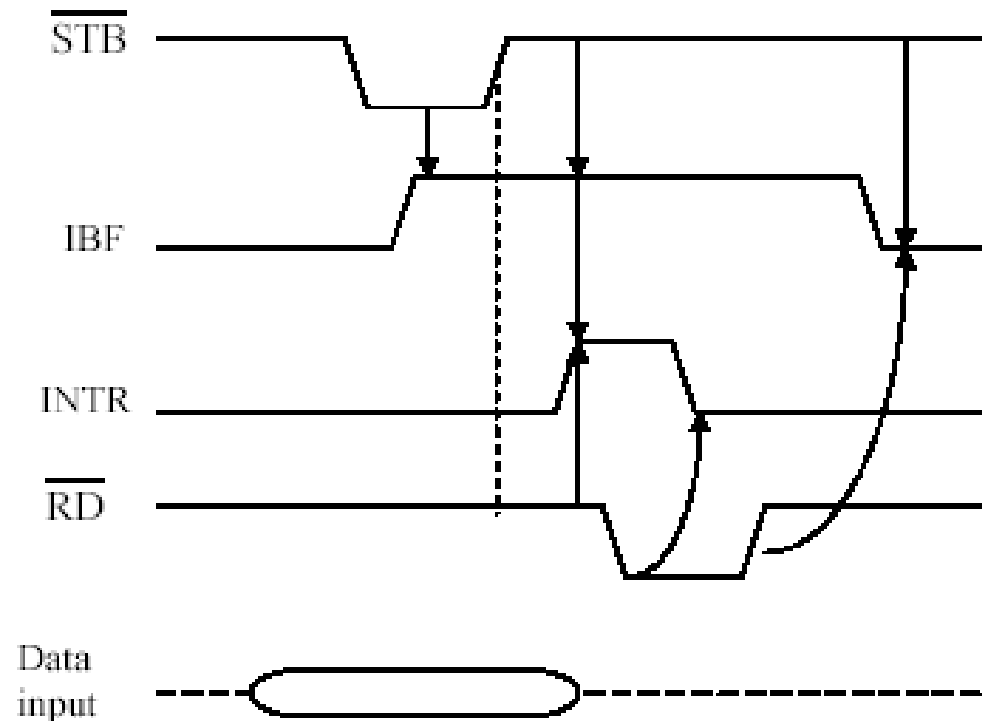
- **STB (Strobe Input):** cho phép chốt dữ liệu
 - Tích cực mức thấp
 - Được tạo bởi thiết bị ngoại vi để xác định rằng ngoại vi đã truyền 1 byte dữ liệu.
 - Khi 8255A đáp ứng STB, nó sẽ tạo ra IBF và INTR
- **IBF (Input Buffer Full):** đệm vào đầy
 - Dùng để xác nhận 8255A đã nhận byte dữ liệu.
 - Bị xoá khi μP đọc dữ liệu.
- **INTR (Interrupt Request):** Yêu cầu ngắt
 - Tín hiệu ra dùng để ngắt μP .
 - Được tạo ra nếu STB, IBF và INTE (flipflop bên trong) đều ở mức logic 1 và bị xoá bởi cạnh xuống của tín hiệu RD.
- **INTE (Interrupt Enable):**
 - Là một flipflop dùng để cho phép hay cấm quá trình tạo ra tín hiệu INTR.
 - Hai flipflop INTEA và INTEB được đặt/xoá dùng BSR mode thông qua PC4 và PC2.



Cấu hình vào của 8255A ở mode 1

Vào dữ liệu trong chế độ 1

❖ Dạng sóng định thì cho cổng vào có strobe



Vào dữ liệu trong chế độ 1

❖ Các từ điều khiển và trạng thái

- Từ điều khiển: để xác định từ điều khiển

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	1/0	1	1	X
I/O mode	PA: Mode 1		PA: nhập	PC6,7 1: nhập 0: xuất	PB: Mode 1	PB: nhập	

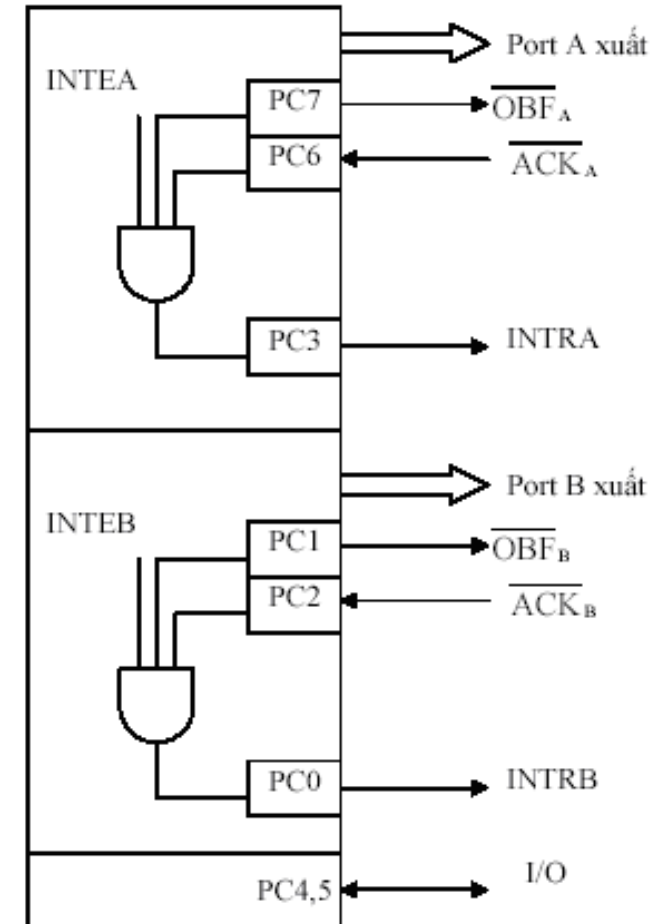
- Từ trạng thái: sẽ được đặt trong thanh ghi tích lũy nếu đọc Port C

D7	D6	D5	D4	D3	D2	D1	D0
I/O	I/O	IBFA	INTEA	INTRA	INTEB	IBFB	INTRB

Ra dữ liệu trong chế độ 1

❖ Chức năng các đường tín hiệu :

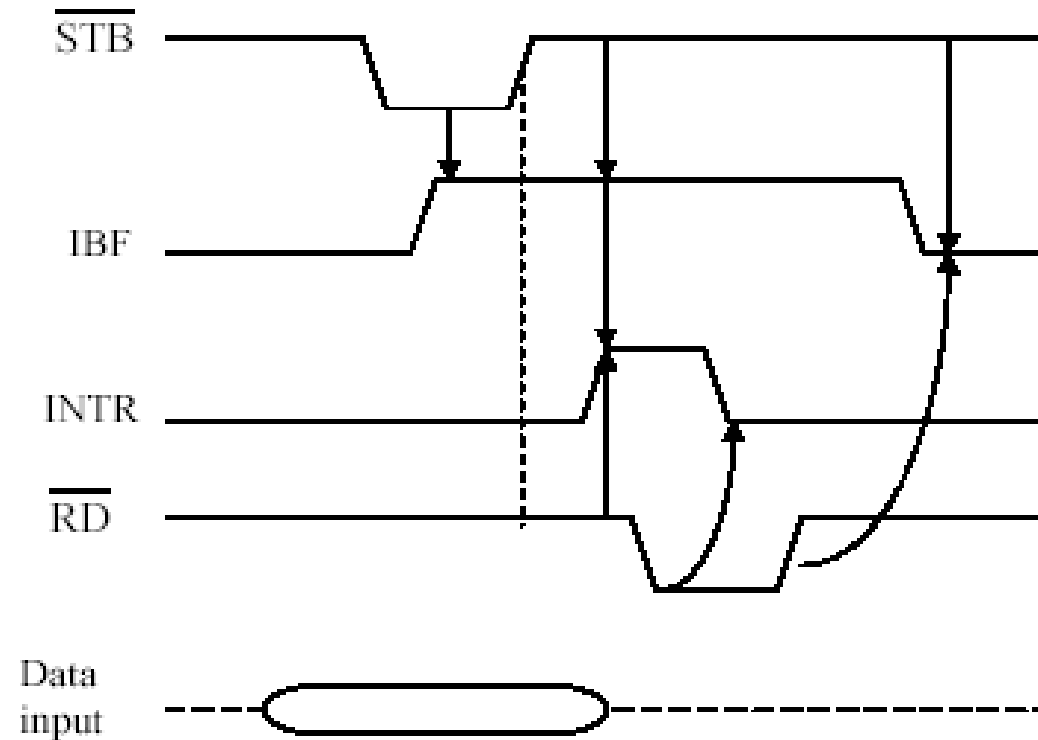
- OBF (Output Buffer Full): tín hiệu này sẽ xuống mức thấp khi mP ghi dữ liệu vào Port xuất của 8225A. Tín hiệu này đưa đến thiết bị ngoại vi để xác định dữ liệu sẵn sàng đưa vào ngoại vi (Hình 4.14). Nó sẽ lên mức cao khi 8255A nhận ACK từ ngoại vi.
- ACK (Acknowledge): đây là tín hiệu nhập từ ngoại vi (tích cực mức thấp) xác nhận dữ liệu đã nhập vào ngoại vi.
- INTR (Interrupt Request): đây là tín hiệu xuất, đặt bằng cạnh lên của tín hiệu ACK. Tín hiệu này có thể dùng để ngắt mP yêu cầu byte dữ liệu kế tiếp để xuất. INTR được đặt khi OBF, ACK và INTE ở mức logic 1 và được xóa bởi cạnh xuống của tín hiệu WR
- INTE (Interrupt Enable): đây là flipflop nội dùng để tạo tín hiệu INTR. Hai flipflop INTEA và INTEB điều khiển bằng các bit PC6 và PC2 thông qua BSR mode.



Cấu hình xuất của 8255A ở mode 1

Ra dữ liệu trong chế độ 1

❖ Dạng sóng định thì cho ngõ vào có strobe



Ra dữ liệu trong chế độ 1

❖ Các từ điều khiển và trạng thái

■ Từ điều khiển:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	1/0	1	0	X
I/O mode	PA: Mode 1		PA: xuất	PC4,5 1: nhập 0: xuất	PB: mode 1	PB: xuất	

■ Từ trạng thái:

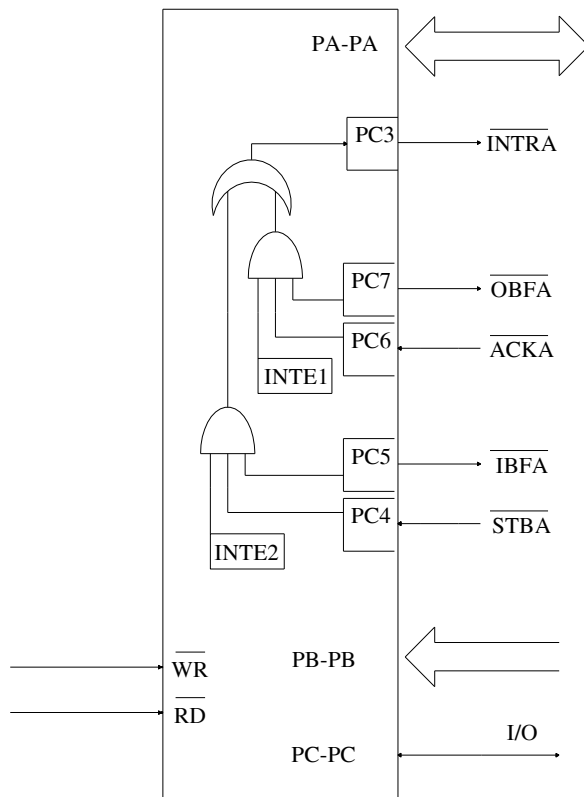
D7	D6	D5	D4	D3	D2	D1	D0
OBF _A	INTEA	I/O	I/O	INTRA	INTEB	OBF _B	INTRB

Mode 2: Vào ra 2 chiều

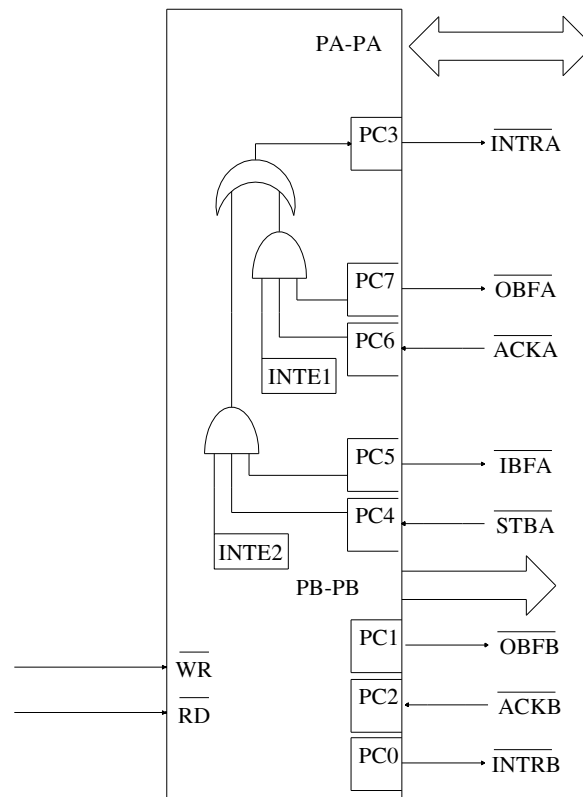
- ❖ Mode này dùng chủ yếu trong các ứng dụng như truyền dữ liệu giữa hai máy tính hay giao tiếp bộ điều khiển đĩa mềm.
- ❖ Trong mode này:
 - Port A dùng làm Port I/O 2 chiều
 - Port B làm việc ở Mode 0 hay 1.
 - Port A sử dụng 5 tín hiệu tại Port C (PC3→PC7) làm các tín hiệu điều khiển để truyền dữ liệu.
 - 3 tín hiệu còn lại của Port C (PC0→PC2) được dùng làm I/O đơn giản hay bắt tay cho Port B.

Mode 2: Vào ra 2 chiều

❖ 8255A ở chế độ bus 2 chiều:

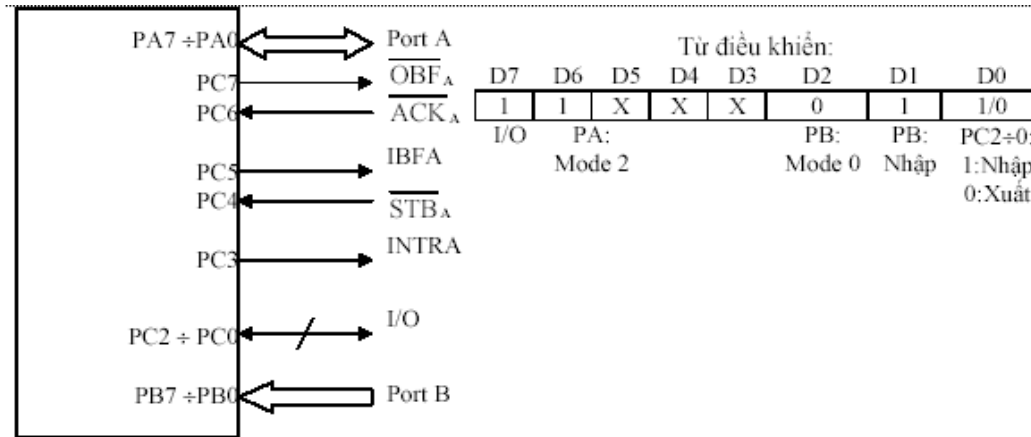


PA chế độ 2
PB: chế độ 0, vào

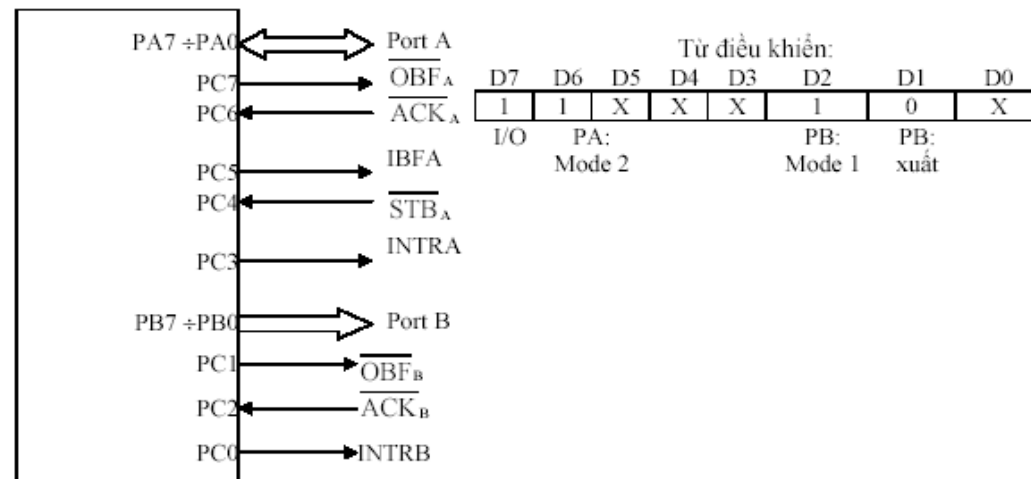


PA chế độ 2
PB: chế độ 1, ra

Mode 2: Vào ra 2 chiều



(a) 8255A ở mode 2 và mode 0 (nhập)



(a) 8255A ở mode 2 và mode 1 (xuất)

Mode BSR

- ❖ Mode BSR chỉ liên quan đến 8 bit của Port C
- ❖ Có thể đặt hay xóa các bit bằng cách ghi một từ điều khiển thích hợp vào thanh ghi điều khiển.
- ❖ Một từ điều khiển với $D_7 = 0$ gọi là từ điều khiển BSR, từ điều khiển này không làm thay đổi bất cứ từ điều khiển nào được truyền trước đó với $D_7 = 1$,
 - Các hoạt động I/O của Port A và B không bị ảnh hưởng bởi từ điều khiển BSR.

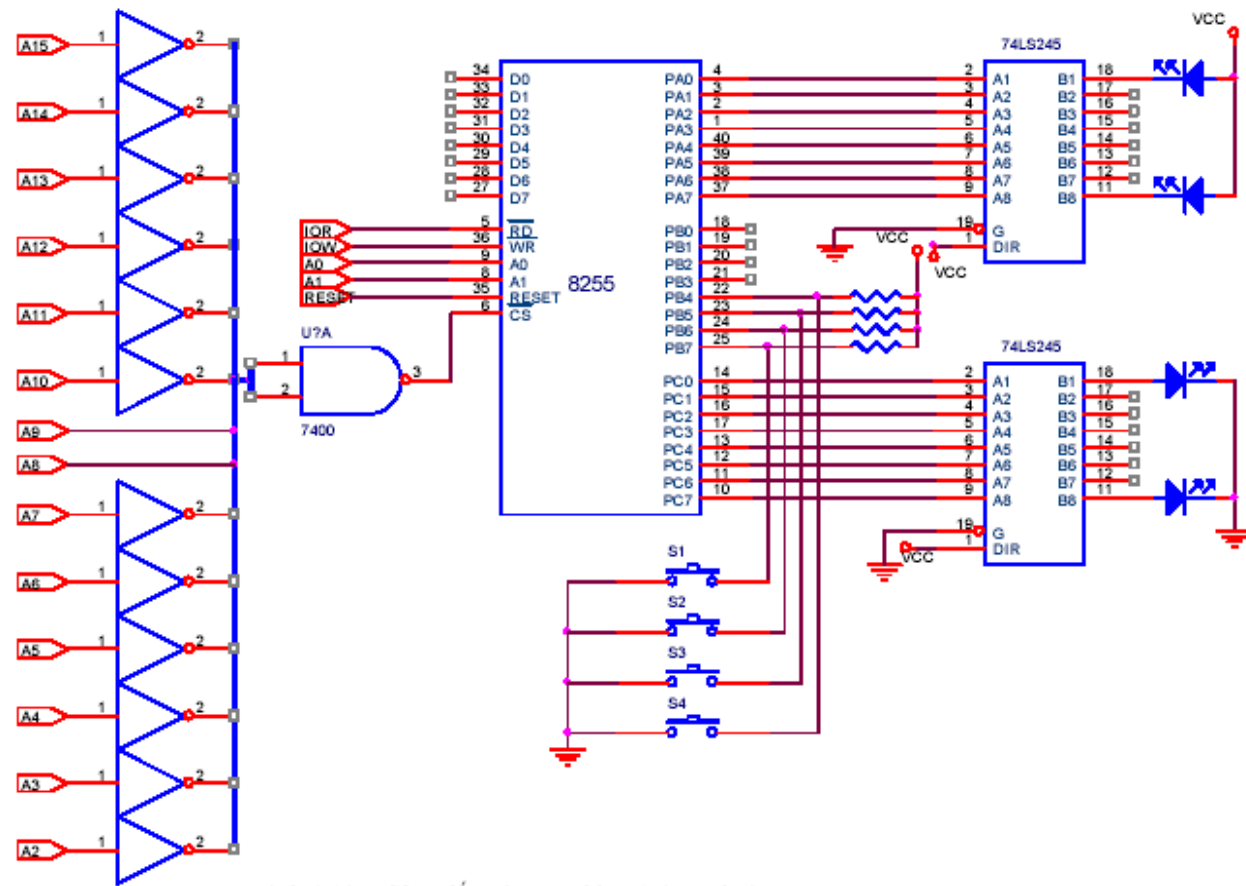
Từ điều khiển BSR

- ❖ Từ điều khiển BSR khi được ghi vào thanh ghi điều khiển sẽ đặt hay xoá mỗi lần 1 bit.

D7	D6	D5	D4	D3	D2	D1	D0
0	x	x	X				S/R
Mode BSR	Không sử dụng			Chọn bit			D0 = 0: Xoá (Reset) 1: Đặt (Set)
				000: PC0			
				001: PC1			
				010: PC2			
				011: PC3			
				100: PC4			
				101: PC5			
				110: PC6			
				111: PC7			

Từ điều khiển BSR

❖ **Ví dụ:** Xét sơ đồ kết nối 8255A như hình dưới. Hãy tạo một sóng chữ nhật tại bit PC0.



Ví dụ

- ❖ Để tạo một sóng chữ nhật tại PC0, ta cần 2 mức logic là 0 và 1 tại PC0.

	D7	D6	D5	D4	D3	D2	D1	D0	
Đặt bit PC0 = 1	0	0	0	0	0	0	0	1	= 01h
Xoá bit PC0 = 0	0	0	0	0	0	0	0	0	= 00h

- ❖ Địa chỉ thanh ghi điều khiển: 303h

- ❖ Chương trình con:

BSR:

```
MOV    AL,01h        ; Từ điều khiển BSR
MOV    DX,303h        ; Địa chỉ thanh ghi điều khiển (CR)
OUT    DX,AL          ; Đặt PC0 = 1
CALL   DELAY1         ; Chờ
MOV    AL,00h        ; Từ điều khiển BSR
OUT    DX,AL          ; Xóa PC0 = 0
CALL   DELAY2         ; Chờ
JMP    bsr
```

Chú ý

❖ Khi sử dụng ở mode BSR, cần chú ý:

- Để đặt hay xoá các bit ở Port C, từ điều khiển được ghi vào thanh ghi điều khiển chứ không ghi vào Port C.
- Một từ điều khiển BSR chỉ ảnh hưởng đến một bit của Port C.
- Từ điều khiển BSR không ảnh hưởng đến chế độ I/O.

Ví dụ 1

❖ **Ví dụ 1:** Giả thiết mạch 8255A có các địa chỉ sau:

Địa chỉ	Thanh ghi (Cổng)
7Ch	PA
7Dh	PB
7Eh	PC
7Fh	CR

- Lập trình để định nghĩa chế độ 0 cho 8255 với cấu hình các cổng:
 - PA: Ra
 - PB: Vào
 - PC_H: Ra
 - PC_L: Vào
- Sau đó đọc các:
 - dữ liệu có tại PB rồi đưa ra PA
 - dữ liệu có tại PC_L rồi đưa ra PC_H.

Ví dụ 1

; Định nghĩa các hằng cho cổng, thanh ghi điều khiển và
; cho từ điều khiển bằng lệnh giả EQU

PA EQU 7CH

PB EQU 7DH

PC EQU 7EH

CR EQU 7FH

CW EQU 83H ; từ điều khiển cho yêu cầu trên 83H=10000011

; dùng các hằng đó để định nghĩa cấu hình cho 8255A

MOV AL, CW ; từ điều khiển trong AL

OUT CR, AL ; đưa CW vào CWR

IN AL, PB ; đọc cổng PB

OUT PA, AL ; đưa dữ liệu đã đọc ra cổng PA

IN AL, PC ; đọc cổng PC_L

MOV CL, 4 ; sau đó quay AL

ROL AL, CL ; chuyển 4 bit thấp thành 4 bit cao

OUT PC, AL ; đưa dữ liệu đã đọc ra cổng PC_H

Ví dụ 2

- ❖ **Ví dụ 2:** Lập trình để bit PC4 của 8255A ở ví dụ 1 tạo ra 256 xung với $T=50\text{ms}$, độ rộng 50%. Giả thiết đã có sẵn chương trình con trễ 25ms là TRE25MS.

Ví dụ 2

; Định nghĩa các hằng cho cổng, thanh ghi từ điều khiển và cho từ điều khiển bằng lệnh giả EQU

PA	EQU	7CH	
PB	EQU	7DH	
PC	EQU	7EH	
CR	EQU	7FH	
CW	EQU	83H	; tu dieu khien cho yeu cau tren 83H=10000011
PC4ON	EQU	09H	; tu dieu khien de PC4=1 (On): 00001001B
PC4OFF	EQU	08H	; tu dieu khien de PC4=0 (Off): 00001000B

; đoạn chương trình còn lại

MOV	CX, 256	; so xung phai tao ra
CLI		; cam ngat de yen tam tao xung
MOV	AL, CW	; tu dieu khien trong AL
OUT	CR, AL	; dua CW vào CWR

Lap:

MOV	AL, PC4ON	; tu dieu khien cho PC4=1.
OUT	CR, AL	; PC4=1
CALL	TRE25MS	; keo dai 25ms.
MOV	AL, PC4OFF	; PC4=0
OUT	CR, AL	
CALL	TRE25MS	; keo dai 25ms
LOOP	Lap	; lap cho du so xung
STI		; cho phep ngat tro lai
...		

Ví dụ

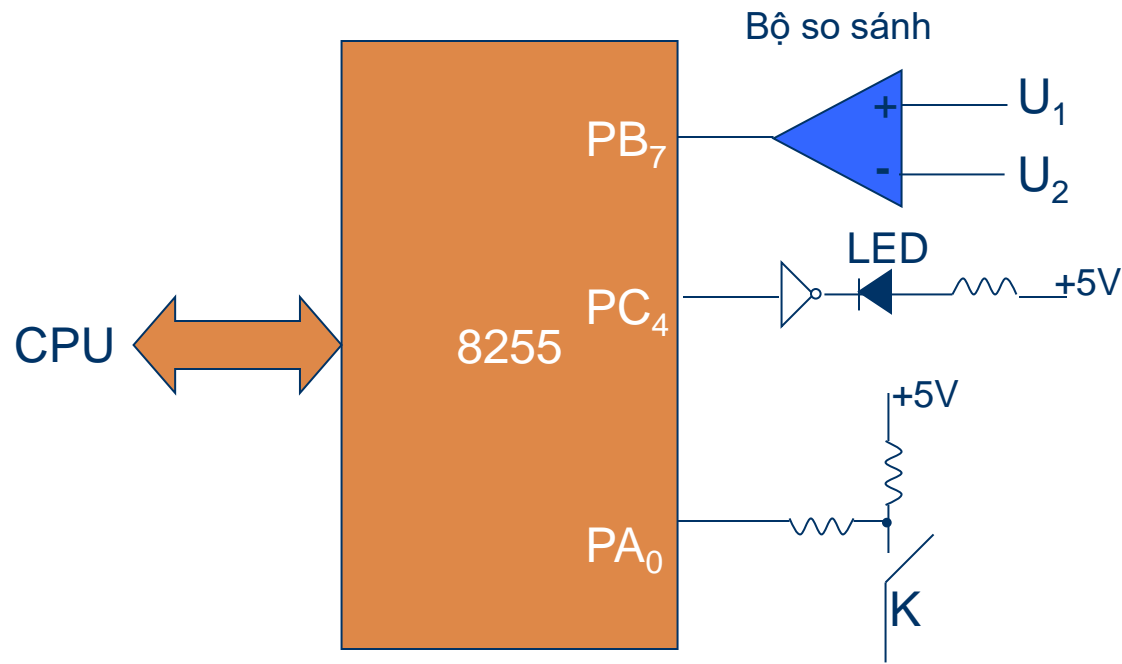
❖ Trong ví dụ trên, chú ý đến:

- Lệnh CLI (để cấm ngắt)
- Lệnh STI (để cho phép ngắt)
- ở đầu và cuối đoạn chương trình tính thời gian tạo ra dãy xung. Điều này là cần thiết vì ta dùng chương trình để tạo ra các khoảng thời gian và vì thế ta muốn các yêu cầu ngắt (nếu có) không ảnh hưởng tới việc tạo ra các khoảng thời gian đó.

Ví dụ

❖ **Ví dụ 3:** Có mạch 8255A và địa chỉ cơ sở là 30H nối với các phần tử ngoại vi như hình vẽ. Lập trình để có khi $U_1 > U_2$, thì đọc trạng thái công tắc K.

- Nếu K đóng → cho đèn LED tắt
- Nếu K mở → cho đèn LED sáng



Ví dụ

❖ Giải

- Phải định nghĩa các hằng và từ điều khiển để định nghĩa cấu hình:
 - PA, PB là cổng vào, PC là cổng ra \rightarrow mode 0.
 - \rightarrow Từ điều khiển: 10010010 (92h)
 - Khi $U_1 > U_2 \rightarrow$ đọc được PB7=1 \rightarrow phải đọc trạng thái công tắc K
 - Nếu khóa K mở ($PA_0=1$): đưa ra PC4=1 để đèn LED sáng
 - Nếu khóa K đóng ($PA_0=0$): đưa ra PC4=0 để đèn LED tắt
 - Khi $U_1 \leq U_2 \rightarrow$ đọc được PB7=0 \rightarrow đọc lại PB

Ví dụ 3

; Định nghĩa các hằng

PA	EQU	30H	
PB	EQU	31H	
PC	EQU	32H	
CR	EQU	33H	
CW	EQU	92H	; tu điều khiển cho yêu cầu trên 92H=10010010
PC4ON	EQU	09H	; tu điều khiển lặp bit PC4
PC4OFF	EQU	08h	; tu điều khiển xóa bit PC4

; cấu hình cho 8255A, đọc dữ liệu, đưa ra công

```
MOV AL, CW      ; tu điều khiển trong AL
OUT CR, AL       ; đưa CW vào CR
DocPB:
IN AL, PB        ; đọc công PB
AND AL, 80H      ; PB7=1?
JZ DocPB         ; đọc lại
IN AL, PA        ; đọc công PA
And AL, 01H      ; khóa K đóng (PA0=0)?
JZ Tat           ; dừng, tắt đèn
MOV AL, PC4ON
OUT CR, AL
JMP Ra
Tat:
MOV AL, PC4OFF
OUT CR, AL
Ra:...
```

Bài tập

- ❖ Lập chương trình hợp ngữ cho mạch PPI 8255 (có địa chỉ cơ sở là 44h) để thực hiện các công việc:
 - Định nghĩa PB và PC là các cổng vào, PA là cổng ra.
 - Đọc cổng PB, mỗi khi thấy bit $PB0=0$ và $PB6=1$ thì đọc một byte dữ liệu từ cổng PC, nhân đôi kết quả rồi đưa ra cổng PA.

;Dinh nghia hang

PA	EQU	44h
PB	EQU	45h
PC	EQU	46h
CR	EQU	47h
CW	EQU	8Bh

; cau hinh

MOV	AL,CW
OUT	CR,AL

DocPB:

IN	AL, PB
MOV	BL,AL
AND	AL, 01h
JNZ	DocPB
AND	BL,40h
JZ	DocPB
IN	AL,PC
SAL	AL,1
OUT	PA,AL

Thank You!