Bùi Minh Thành
Hiệu đính từ bài giảng của
thầy Hồ Trung Mỹ (BMDT- DHBK)

Vi Xử Lý

Chương 1

Giới thiệu hệ VXL tổng quát

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

1.1 SỰ PHÁT TRIỂN CỦA CÁC HỆ VI XỬ LÝ

Họ vi mạch số và công nghệ

- **Integrated Circuits**
  - **Integrated Circuits → IC**
  - **Families of Integrated Circuits :**
    - **TTL** Transistor-Transistor Logic
    - **ECL** Emitter-Coupled Logic
    - **MOS** Metal-Oxide Semiconductor
    - **CMOS** Complementary Metal-Oxide Semiconductor

– Integrated Circuits classification :

Classification	Transistor	Typical IC
SSI	10 or less	54/74 logic gate
MSI	10 to 100	counter, adders
LSI	100 to 1000	small memory ICs, gate array
VLSI	1000 to 10^6	large memory ICs, microprocessor
ULSI	10^6 and up	Multifunction ICs

– Various series of the TTL logic family :

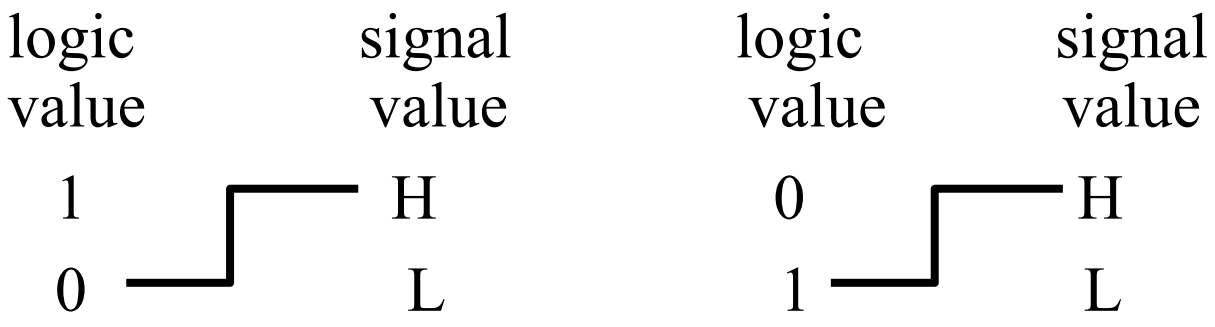
TTL Series	Prefix	Example
Standard TTL	74	7486
High-speed TTL	74H	74H86
Low-power TTL	74L	74L86
Schottky TTL	74S	74S86
Low-power Schottky TTL	74LS	74LS86
Advanced Schottky TTL	74AS	74AS86
Advanced Low-power Schottky TTL	74ALS	74ALS86

– Various series of the CMOS logic family :

CMOS Series	Prefix	Example
Original CMOS	40	4009
Pin compatible with TTL	74C	74C04
High-speed and Pin compatible with TTL	74HC	74HC04
High-speed and electrically compatible with TTL	74HCT	74HCT04

– Signal assignment and logic polarity :

Positive Logic		Negative Logic	
logic level	signal level	logic level	signal level
1	H	0	H
0	L	1	L



– Demonstration of positive and negative logic

x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

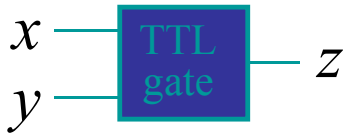
Truth table with H and L

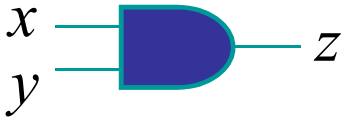
x	y	z
0(L)	0(L)	0(L)
0(L)	1(H)	0(L)
1(H)	0(L)	0(L)
1(H)	1(H)	1(H)

Truth table for positive logic

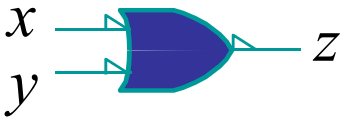
x	y	z
1(L)	1(L)	1(L)
1(L)	0(H)	1(L)
0(H)	1(L)	1(L)
0(H)	0(H)	0(H)

Truth table for Negative logic



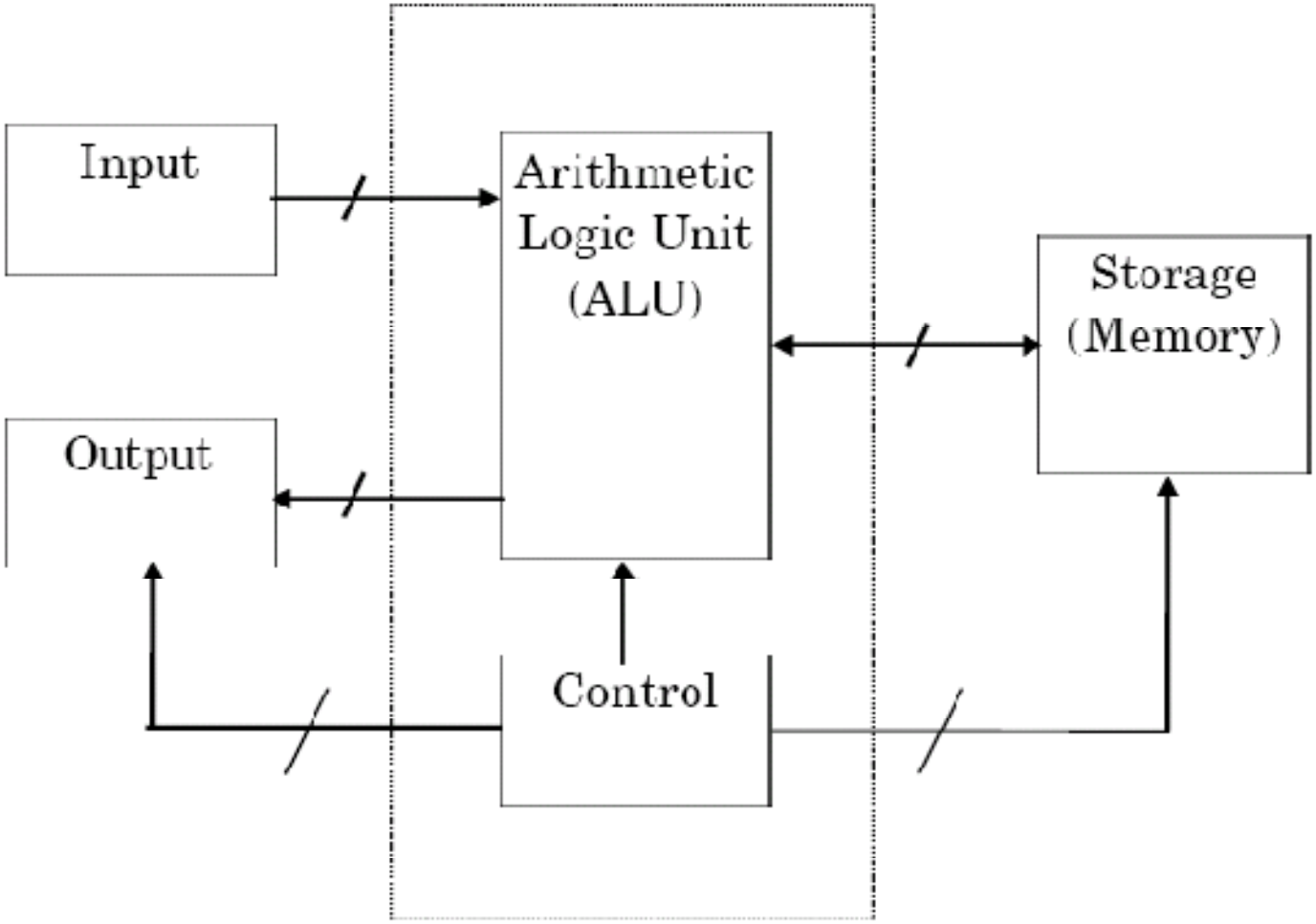


Positive logic AND gate



Negative logic OR gate

Sơ đồ khối một máy tính cổ điển



Hình 1.1 Máy tính cổ điển (Classical computer).

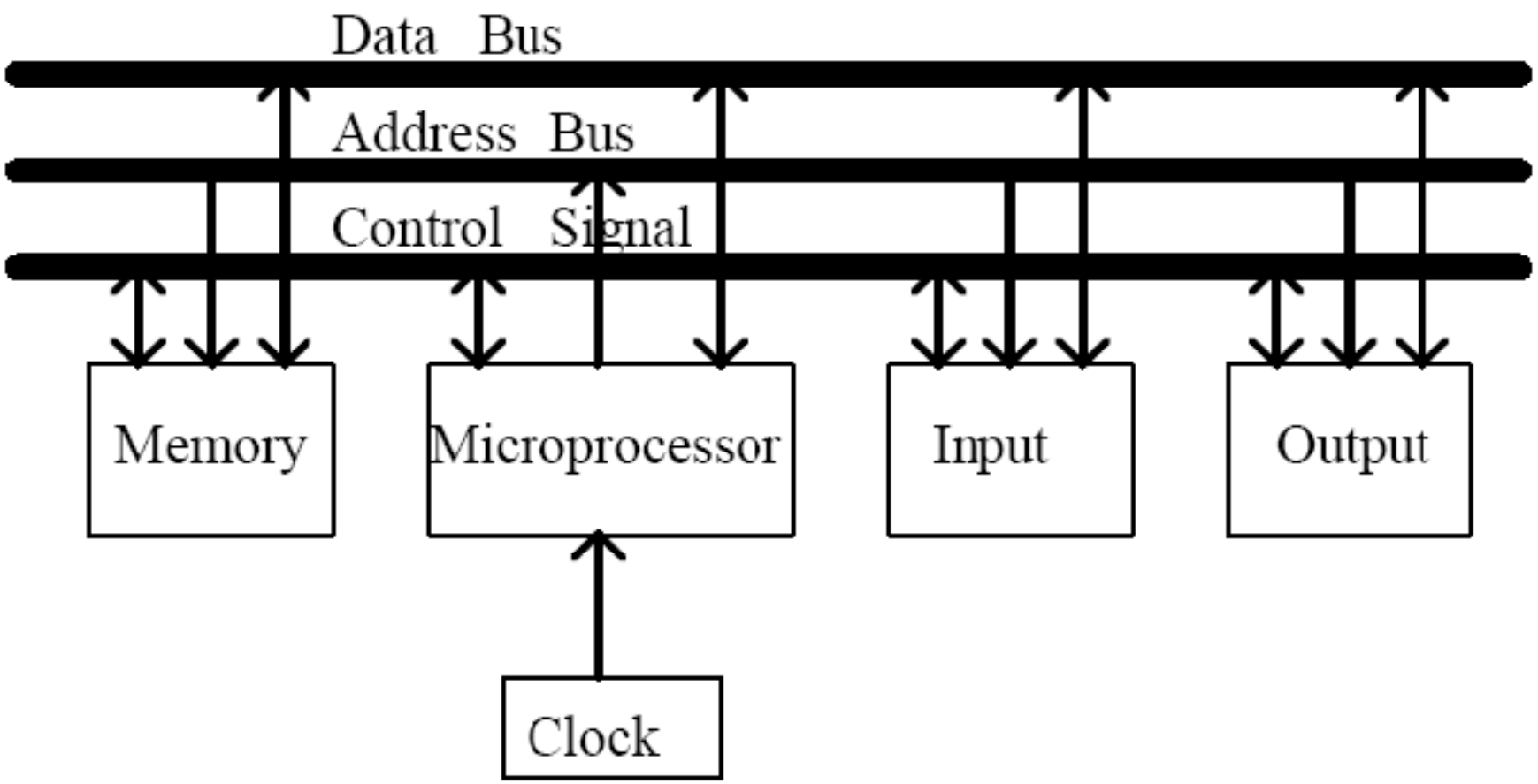
Phân loại CPU

Người ta có thể chia CPU làm 3 loại :

- **Multi-chip CPU** (CPU đa chip): Cần 2 hay nhiều chip LSI để cài đặt ALU và phần điều khiển của máy tính.
- **Microprocessor** (Vi xử lý): ta sẽ hạn chế từ microprocessor (mP/UP) cho một chip LSI/VLSI chứa ALU và phần điều khiển của một máy tính.
- **Single chip microprocessor** (Vi xử lý đơn chip): (còn gọi là microcomputer/microcontroller) là 1 chip LSI/VLSI chứa toàn bộ một máy tính như ở hình 1.1, và thường được gọi tắt là MCU (Micro-Controller Unit).

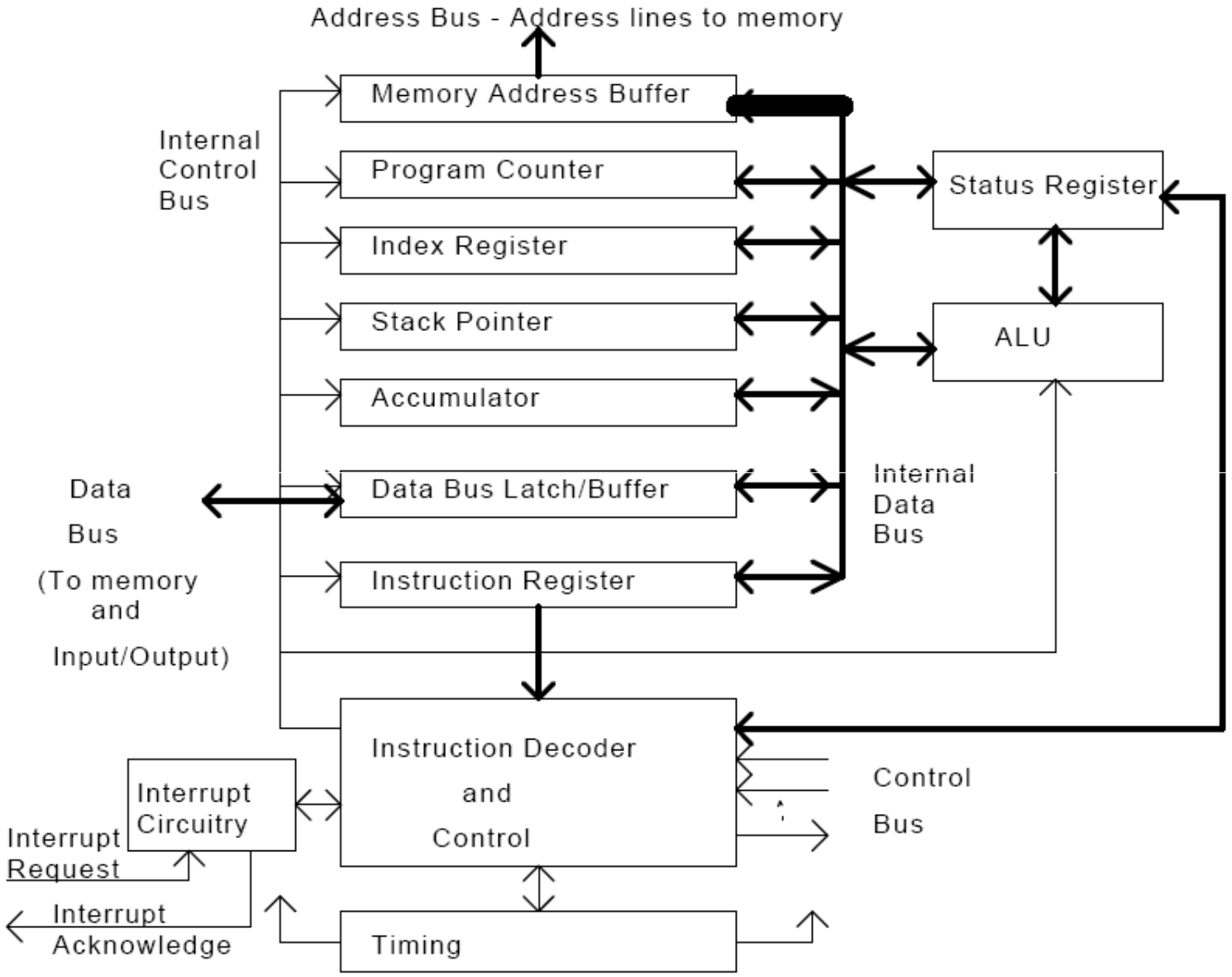
Sơ đồ khối máy vi tính

Một máy tính dựa trên vi xử lý thì được gọi là máy vi tính (microcomputer) và được gọi tắt là µC (uC)



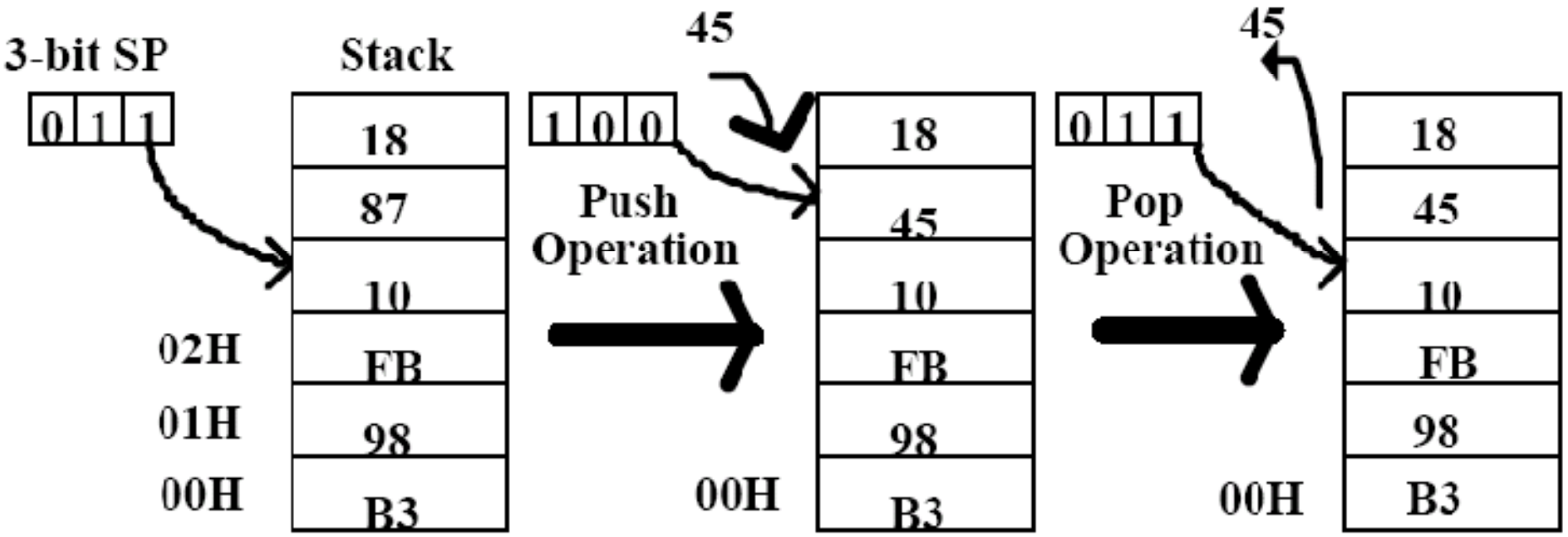
Hình 1.2 Sơ đồ khối của máy vi tính

Tổ chức bên trong của vi xử lý



Hình 1.3 Sơ đồ khối tổ chức bên trong của vi xử lý

Thí dụ cài đặt ngăn xếp trong bộ nhớ.



SP (stack pointer) trỏ đến dữ liệu đang được truy cập

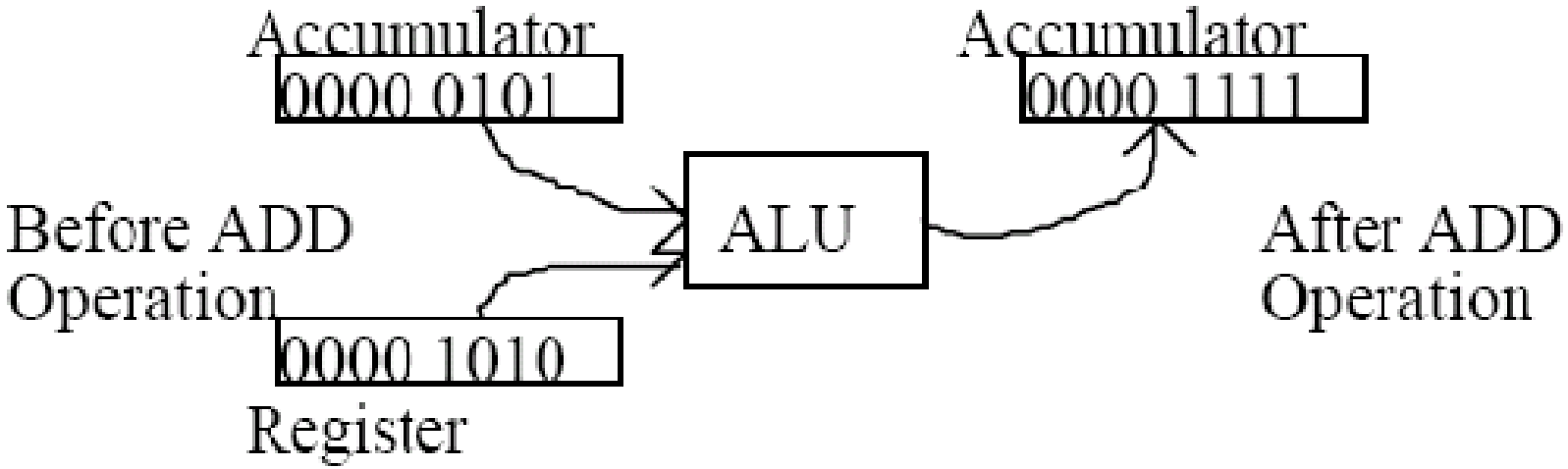
Ví dụ:

PUSH direct
(SP) ← (SP) + 1
((SP)) ← (direct)

POP direct
(direct) ← ((SP))
(SP) ← (SP) - 1

Thanh ghi tích lũy (Accumulator)

- Các kết quả của các phép toán của ALU thường được cất trong thanh ghi tích lũy (cũng được gọi là ACC). Thí dụ ALU thực thi lệnh ADD (cộng) như sau:



Thanh ghi trạng thái (Status Register)

- Trong khi thực hiện một số phép toán số học hoặc logic, một số điều kiện nhất định phát sinh mà ảnh hưởng đến trình tự thực thi chương trình.
- Người ta cần phải lưu trữ các điều kiện như vậy trong một nhóm các flipflop (hoặc thanh ghi) được gọi là **thanh ghi trạng thái (status register)** (cũng được gọi là **thanh ghi mã điều kiện) [code condition register]**) trong một khoảng thời gian để xác định trình tự thực thi chương trình.

Một số cờ trong thanh ghi trạng thái

- Cờ Z (Zero)
- Cờ S (Sign)
- Cờ C (Carry)
- Cờ HC (Half Carry)
- Cờ OV (Overflow)
- . . .

Lịch sử phát triển vi xử lý

Thời kỳ đầu

- 1969 - 70 **Intel 4004**, vi xử lý đầu tiên, 4-bit **Intel 4040**, nhanh hơn 4004
- 1971 **Intel 8008**, phiên bản 8 bit của 4004
- 1973 **Intel 8080**, 10 lần nhanh hơn 8008

  (Các sản phẩm tương tự: Motorola MC6800, Zilog Z80)
- 1974 **MITS Altair 8800**, máy vi tính đầu tiên được lập trình bằng **BASIC** được phát triển bởi Bill Gates và Paul Allen.
- 1977 **Apple II**, máy tính gia đình phổ cập đầu tiên **Intel 8085**, vi xử lý 8 bit sau cùng
- 1978 **Intel 8086**, vi xử lý 16 bit , nhanh hơn nhiều
- 1979 **Intel 8088**

Thập niên 1980

- 1980 **Motorola 68000**
- 1981 **IBM PC** với **Intel 8088**, chạy ở xung nhịp 4.77 MHz với một ổ đĩa mềm 160KB và hệ điều hành MS-DOS 1.0/1.1
- 1982 **Intel 80286**
- 1984 **Apple Macintosh**, với Motorola 68000
- 1985 **Intel 80386**
- 1987 **Macintosh II**
- 1989 **Intel 80486** với tốc độ xung nhịp 25 MHz cao hơn.

Từ thập niên 1990 trở lại đây

- 1990 **Microsoft Windows 3.0** ra đời
  Motorola 68040 được triển khai.
- 1991 **Apple** và **IBM** hợp tác để khảo sát **RISC**
- 1992 **Microsoft Windows 3.1** đã trở thành chuẩn cho các PC.
- 1993 **Intel Pentium (80586)** ra đời, công nghệ **MMX** được cung cấp sau.
- 1995 **Microsoft Windows 95**
- 1995 **Intel Pentium Pro (P6)**
- 1997 **Intel Pentium II**
- 1998 **Intel Pentium II Xeon**
- 1999 **Intel Pentium III**
- 2001 **Intel Pentium IV**

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

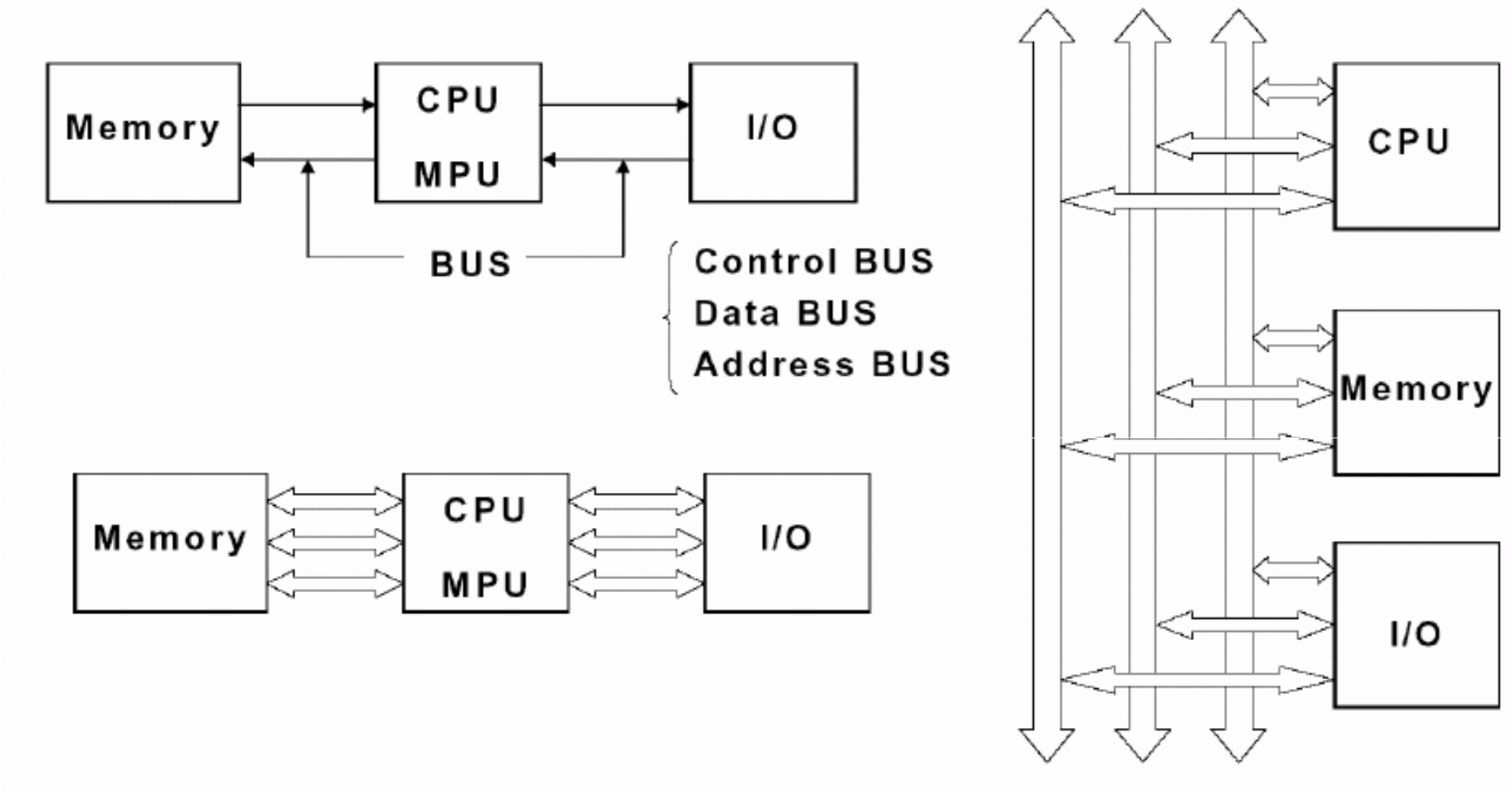
1.7 Giãi mã địa chỉ

1.8 Định thì

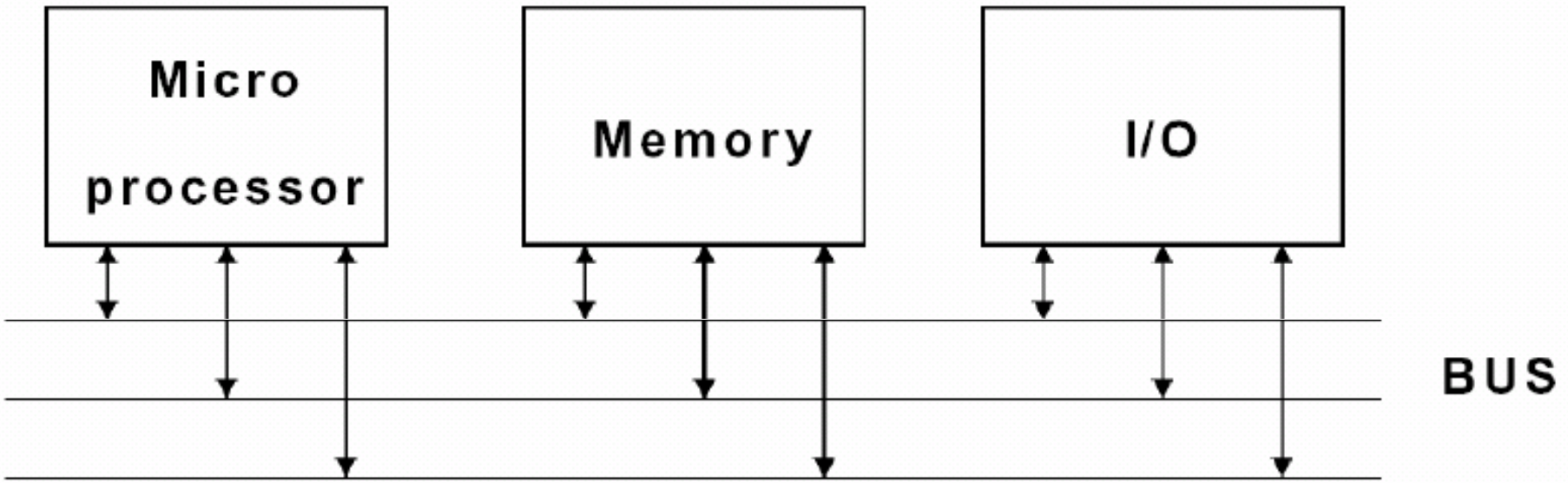
1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

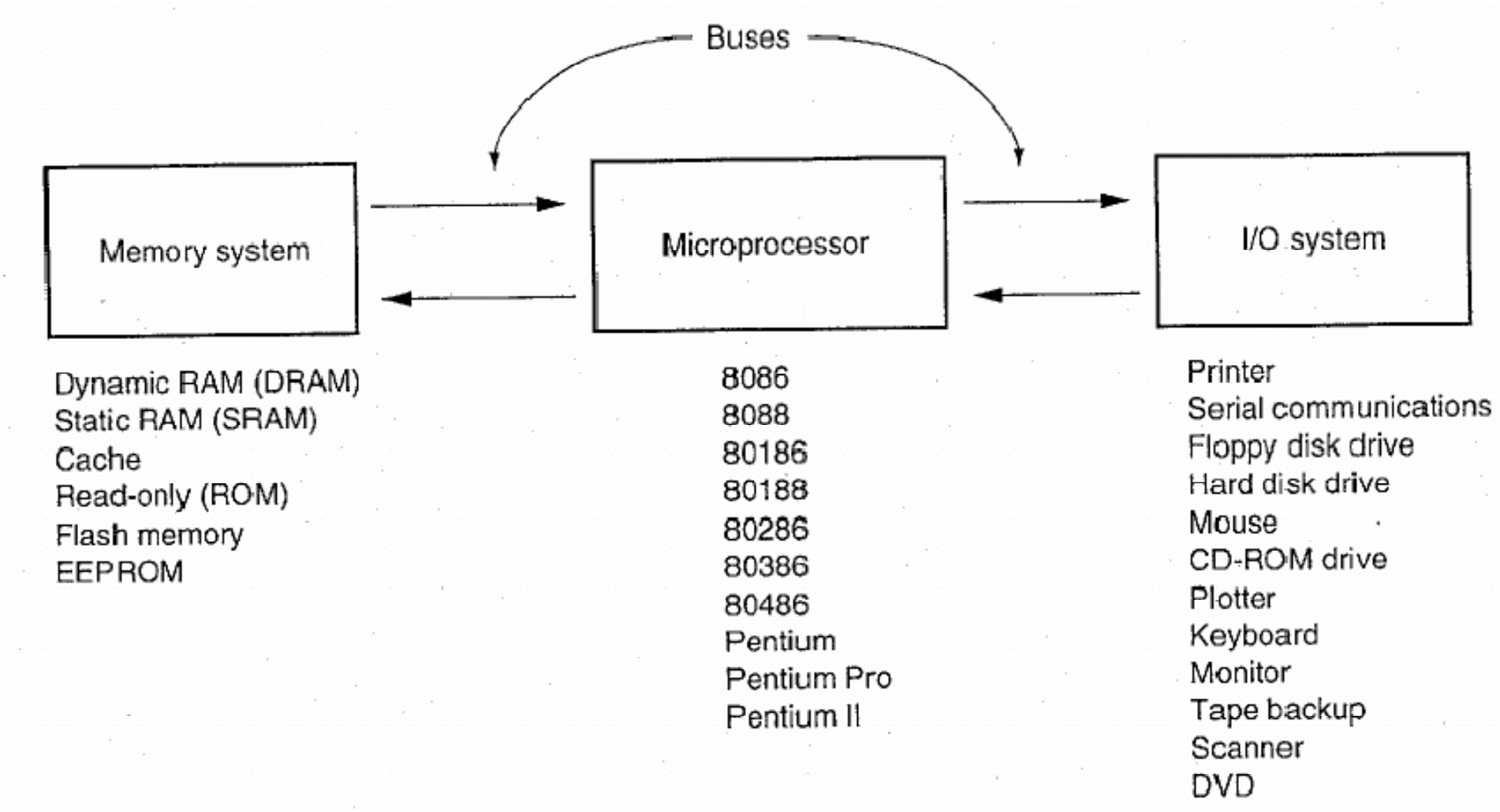
1.2 SƠ ĐỒ KHỐI MỘT HỆ VI XỬ LÝ CƠ BẢN



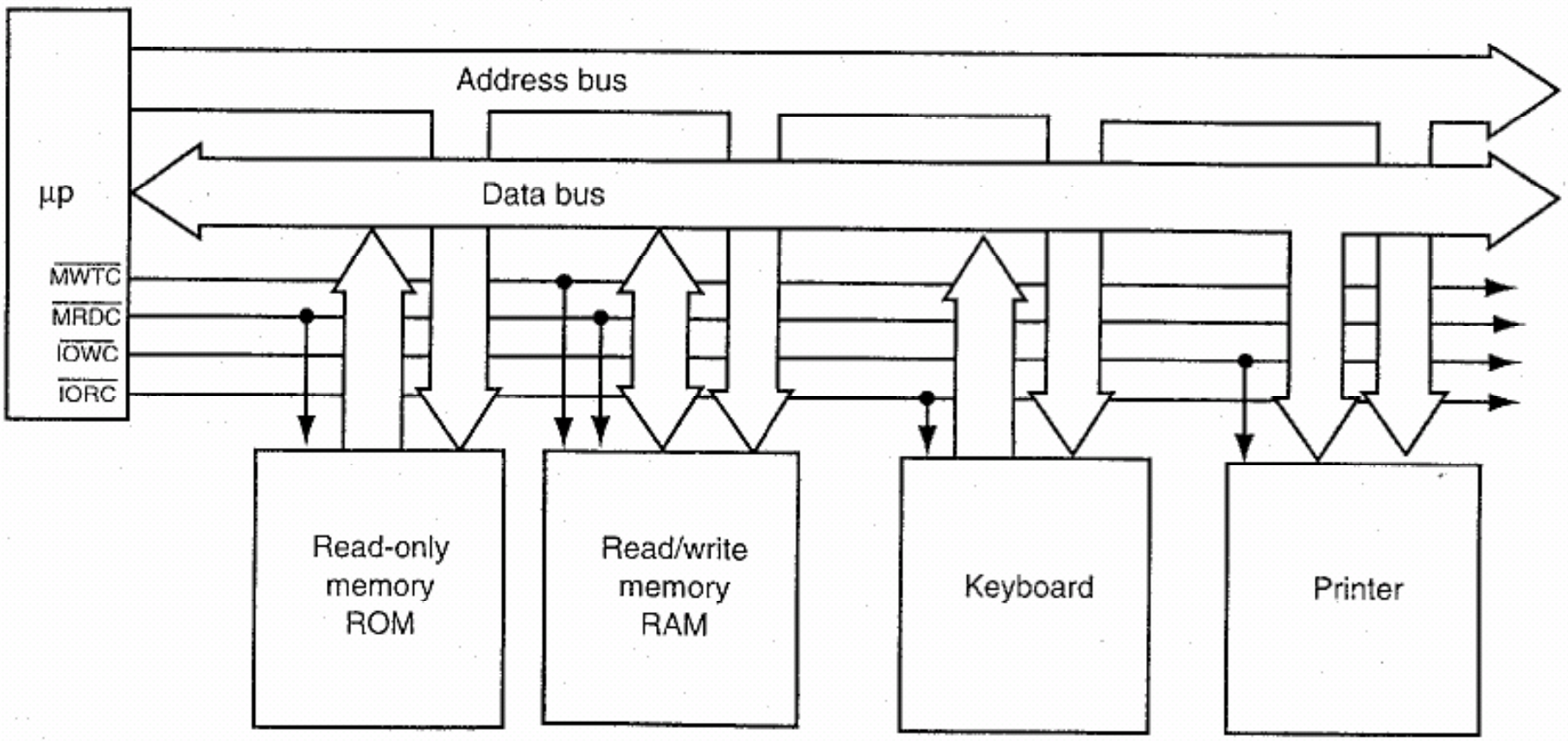
Hình 1.5 Các kết nối cơ bản với vi xử lý



Hình 1.6 Kết nối vi xử lý tiêu biểu

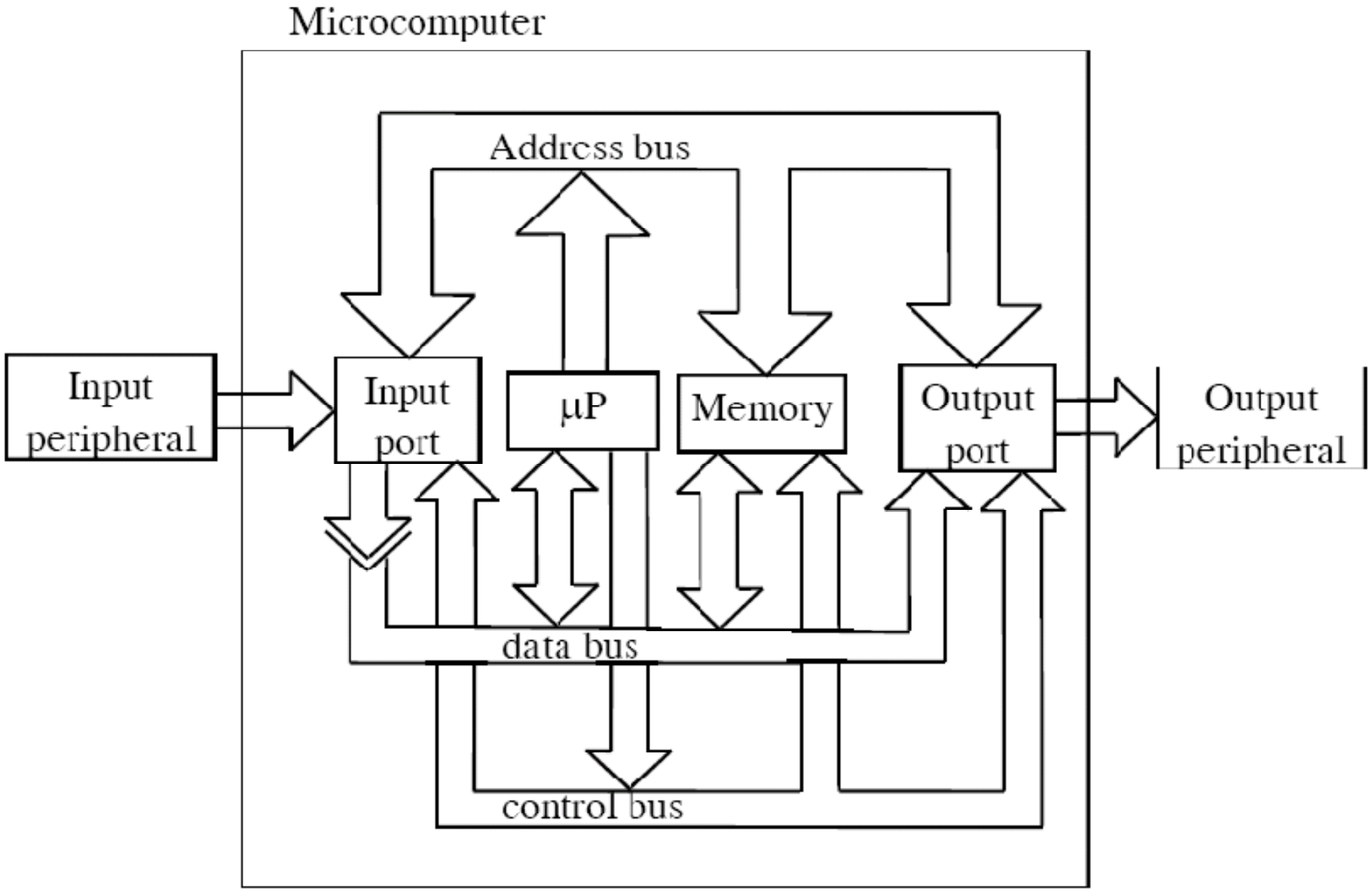


Hình 1.7 Các hệ thống máy tính dựa trên vi xử lý.



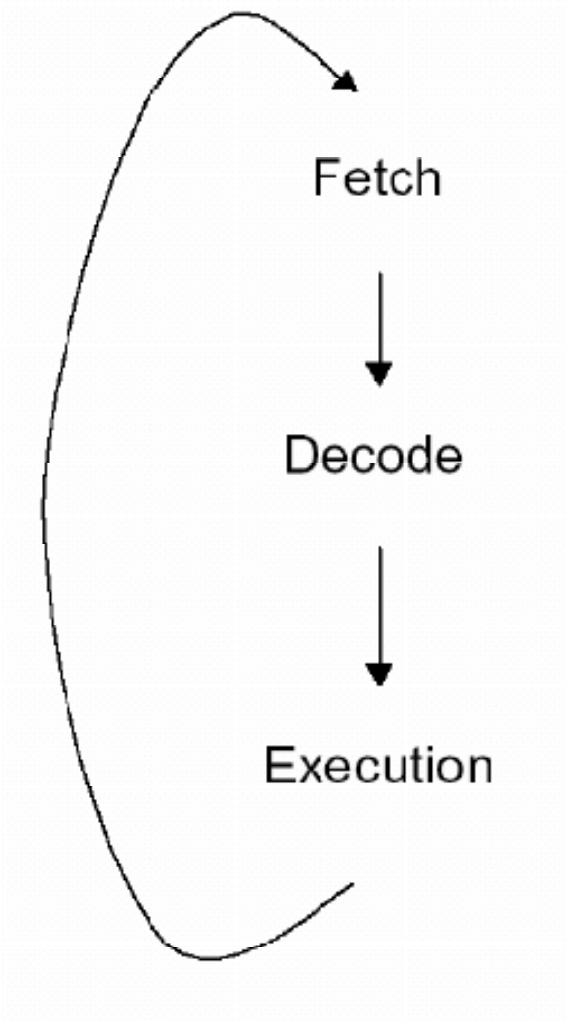
Hình 1.8 Sơ đồ khối hệ thống máy vi tính

Với MRDC=Memory Read Control=Điều khiển đọc bộ nhớ
MWTC=Memory Write Control=Điều khiển ghi bộ nhớ
IORC=I/O Read Control=Điều khiển đọc I/O
IOWC=I/O Write Control=Điều khiển ghi I/O



Hình 1.9 Sơ đồ khối một hệ thống vi xử lý cơ bản.

Chu kỳ nhận (Fetch)–giải mã (Decode)–thực thi (Excecution) của CPU

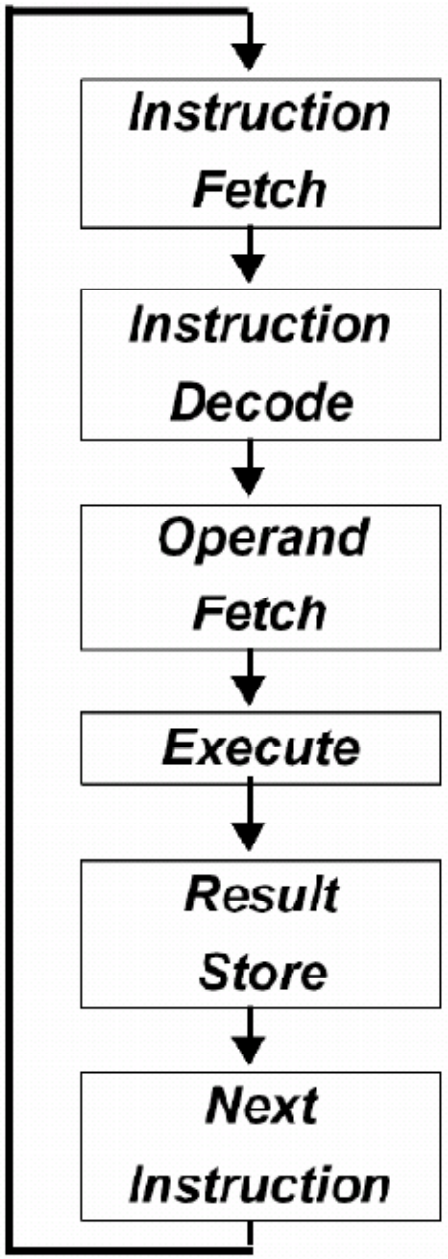


Nạp lệnh từ bộ nhớ vào thanh ghi nội.

Diễn dịch lệnh nhận được

Thực thi lệnh, thí dụ: đọc/ghi dữ liệu, tính toán số học trên dữ liệu nhận được.

Hình 1.10 Chu kỳ nhận–giải mã–thực thi



Nhận lệnh:
Nhận lệnh từ bộ nhớ chương trình

Giải mã lệnh:
Xác định các tác động cần có và kích thước lệnh.

Nhận toán hạng:
Định vị trí để lấy dữ liệu toán hạng.

Thực thi: Tính giá trị kết quả hoặc trạng thái

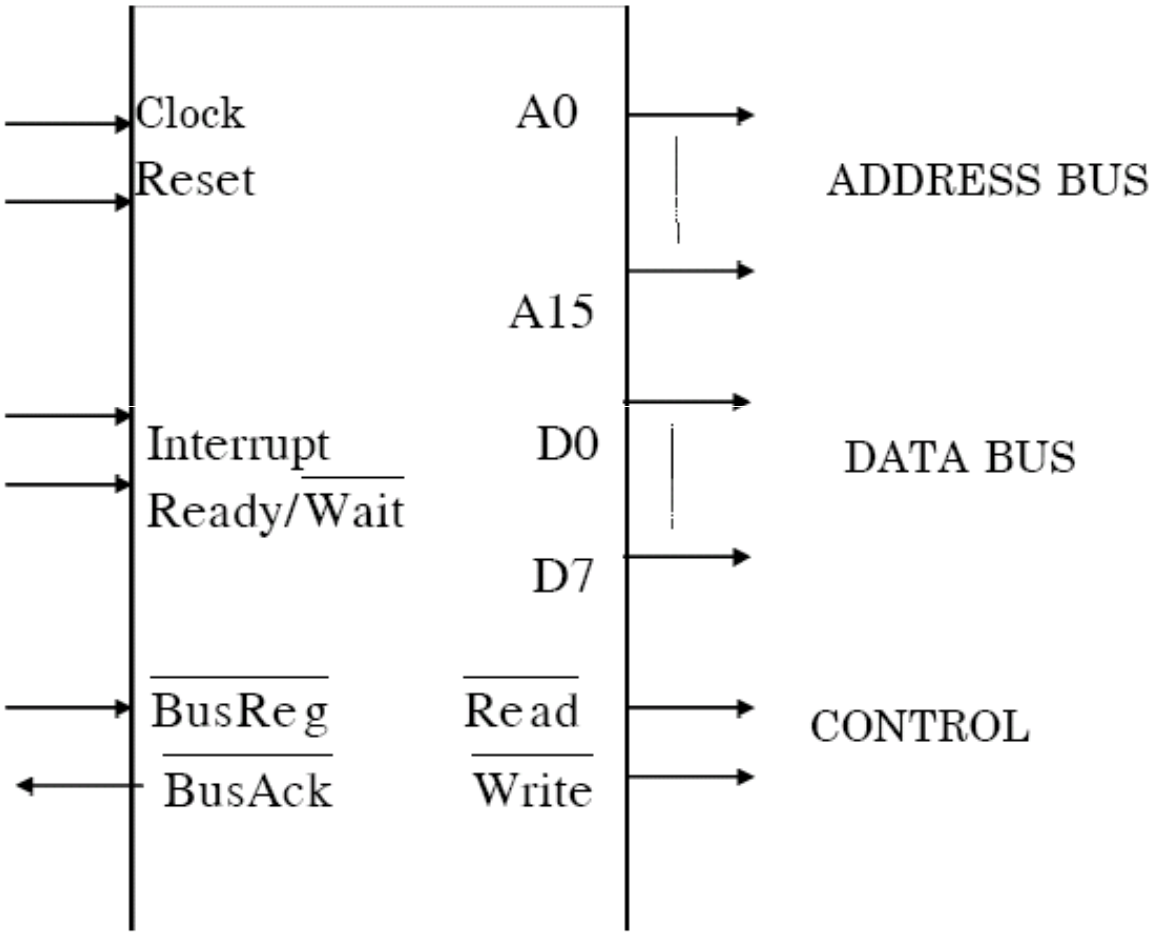
Lưu kết quả:
Cất kết quả vào bộ nhớ hay thanh ghi để sử dụng sau.

Lệnh kế:
Xác định địa chỉ lệnh kế.

Hình 1.11 Chu kỳ nhận giải mã thực thi chi tiết hơn

Các đường tín hiệu kết nối với một vi xử lý tiêu biểu

Hai tín hiệu điều khiển cơ bản là **READ &WRITE** thường được gọi là **read strobe (lấy mẫu đọc)** & **write strobe (lấy mẫu ghi)**.



Hình 1.12 Các tín hiệu vi xử lý cơ bản.

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

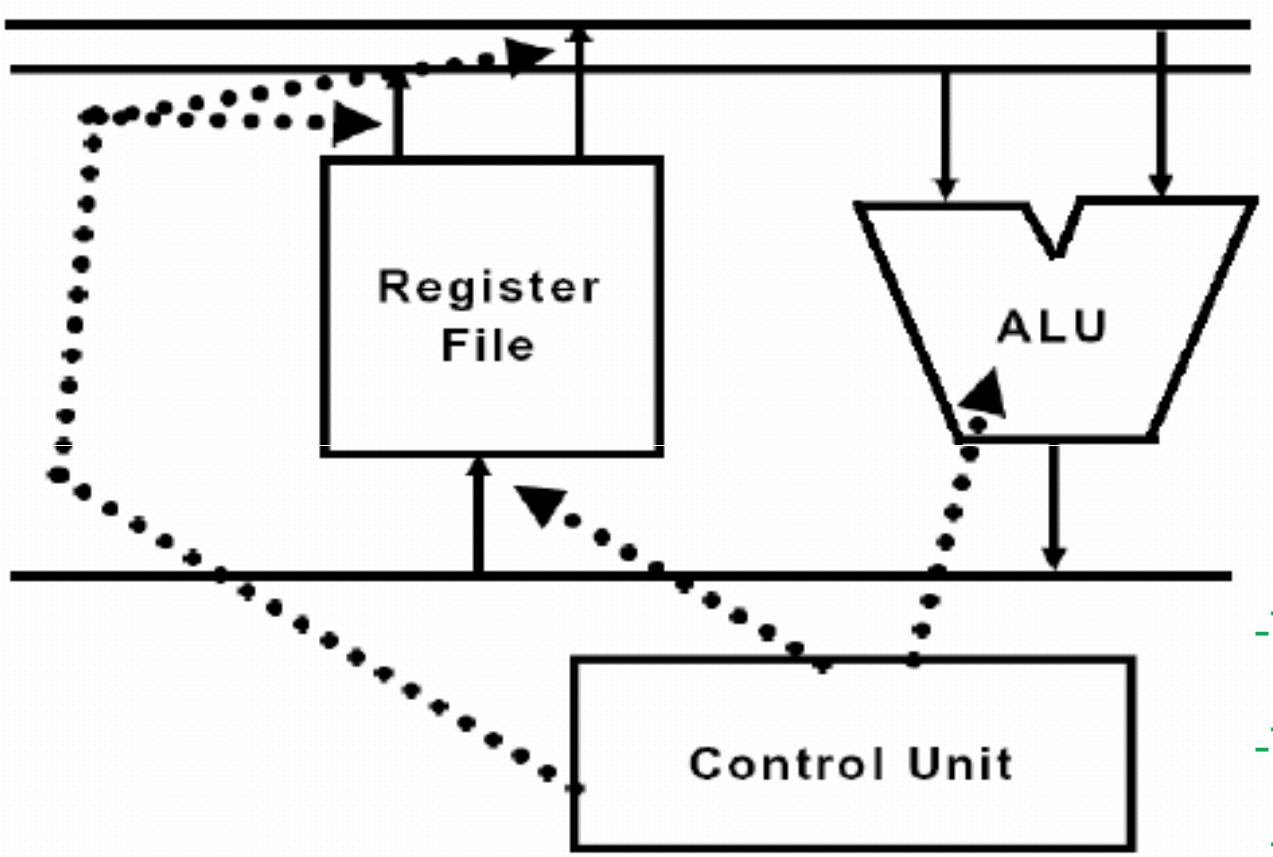
1.8 Định thì

1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

1.3 CPU

Các thành phần chính của CPU



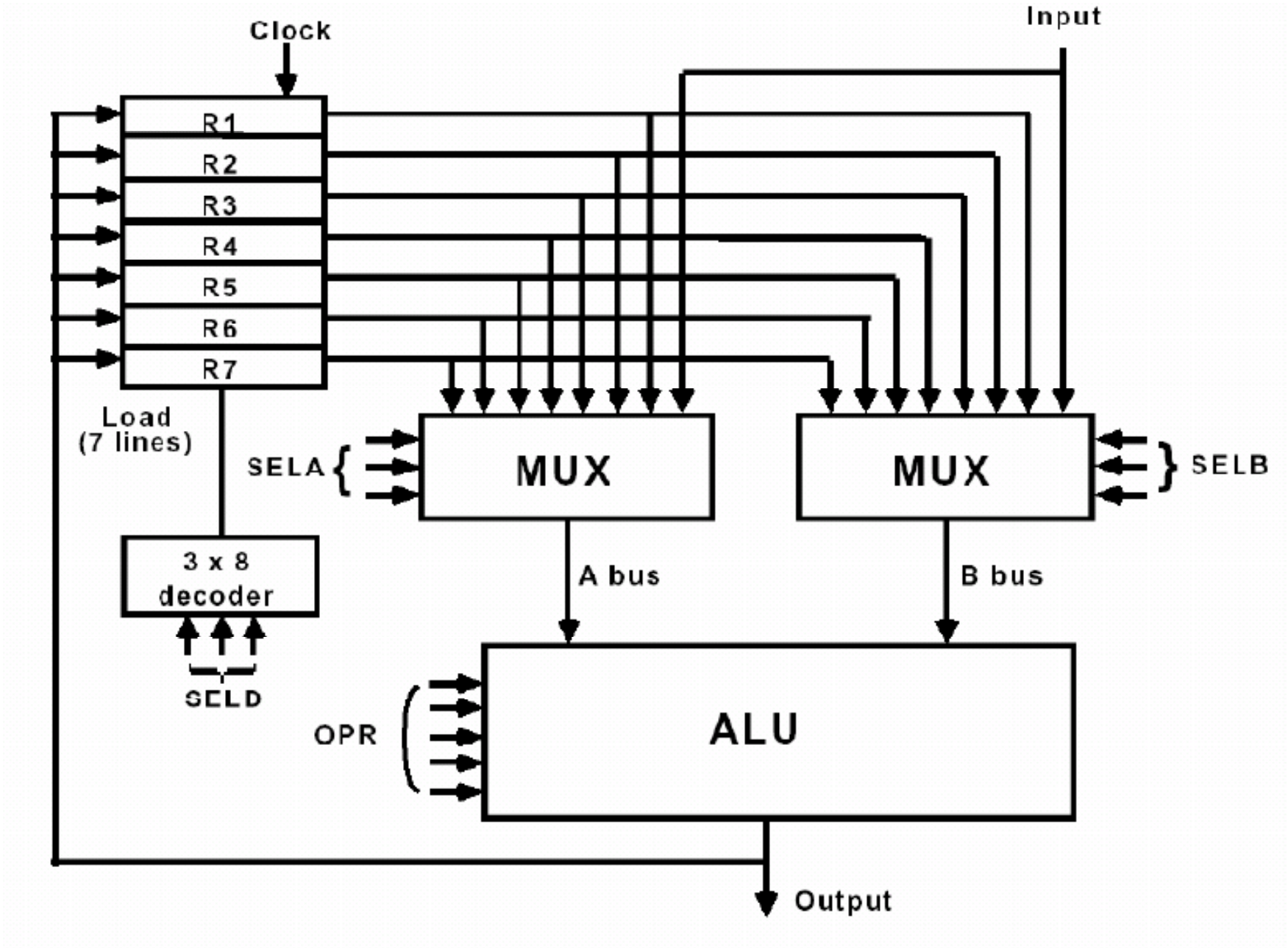
Hình 1.13 Các thành phần chính của CPU

-Thành phần lưu trữ

-Thành phần thực thi (xử lý)

-Thành phần chuyển [tín hiệu]: bus

- Thành phần điều khiển



Hình 1.14 Tổ chức các thanh ghi

Thí dụ: R1 ← R2 + R3

1. Chọn MUX A (SELA): BUS A ← R2
2. Chọn MUX B (SELB): BUS B ← R3
3. Chọn hoạt động ALU (OPR): ADD (cộng)
4. Giải mã chọn lựa đích đến (SELD): R1 ← Out Bus (kết quả từ ALU)

Như vậy "từ điều khiển" (control word) có dạng sau

Số bit	3	3	3	5
	SELA	SELB	SELD	OPR

Mã hóa các vùng chọn thanh ghi

Mã nhị phân	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Mã hóa các tác vụ (hoạt động) của ALU

Chọn	Tác vụ	Ký hiệu
00000	chuyển A	TSFA
00001	Tăng A thêm 1	INCA
00010	ADD A + B	ADD
00101	A – B	SUB
00110	Giảm A đi 1	DECA
01000	AND A và B	AND
01010	OR A và B	OR
01100	XOR A và B	XOR
01110	Bù A	COMA
10000	Dịch phải A	SHRA
11000	Dịch trái A	SHLA

Thí dụ: các vi tác vụ của ALU

Vi tác vụ	Tên ký hiệu				Từ điều khiển			
	SELA	SELB	SELD	OPR				
R1 ← R2 - R3	R2	R3	R1	SUB	010	011	001	00101
R4 ← R4 ∨ R5	R4	R5	R4	OR	100	101	100	01010
R6 ← R6 + 1	R6	-	R6	INCA	110	000	110	00001
R7 ← R1	R1	-	R7	TSFA	001	000	111	00000
Output ← R2	R2	-	None	TSFA	010	000	000	00000
Output ← Input	Input	-	None	TSFA	000	000	000	00000
R4 ← shl R4	R4	-	R4	SHLA	100	000	100	11000
R5 ← 0	R5	R5	R5	XOR	101	101	101	01100

Dạng lệnh

- Các vùng trong lệnh
  1. Vùng mã lệnh (Op-code): Cho biết tác vụ nào sẽ được thực hiện.
  2. Vùng địa chỉ: chỉ địa chỉ bộ nhớ hoặc thanh ghi của CPU.
  3. Vùng cách định địa chỉ (mode): chỉ cách xác định toán hạng hoặc địa chỉ thật.
- Số vùng địa chỉ trong lệnh phụ thuộc vào tổ chức nội của CPU.
- Có 3 tổ chức CPU thông dụng nhất:

– Tổ chức 1 thanh ghi tích lũy:

```
Thí dụ:     ADD X          /*    AC ← AC + M[X]          */
```

– Tổ chức thanh ghi tổng quát

```
Thí dụ:     ADD R1, R2, R3     /*    R1 ← R2 + R3        */
            ADD R1, R2         /*    R1 ← R2 + R3        */
            MOV R1, R2         /*    R1 ← R2             */
            ADD  R1, X         /*    R1 ← R1 + M[X]      */
```

– Tổ chức ngăn xếp

```
Thí dụ:     PUSH X             /*    TOS ← M[X]          */
            ADD
```

Các lệnh 3 địa chỉ

Chương trình tính: X = (A + B) * (C + D)

```
ADD   R1, A, B      /*    R1 ← M[A] + M[B]    */
ADD   R2, C, D      /*    R2 ← M[C] + M[D]    */
MUL   X, R1, R2     /*  M[X] ← R1 * R2        */
```

Kết quả cho chương trình ngắn, nhưng lệnh sẽ dài vì chiếm nhiều bit.

Các lệnh 2 địa chỉ

Chương trình tính: X = (A + B) * (C + D)

```
MOV   R1, A         /*    R1 ← M[A]           */
ADD   R1, B         /*    R1 ← R1 + M[B]      */
MOV   R2, C         /*    R2 ← M[C]           */
ADD   R2, D         /*    R2 ← R2 + M[D]      */
MUL   R1, R2        /*    R1 ← R1 * R2        */
MOV   X, R1         /*    M[X] ← R1           */
```

Các lệnh 1 địa chỉ

Sử dụng thanh ghi tích lũy AC hiểu ngầm cho tất cả các xử lý dữ liệu.

Chương trình tính: X = (A + B) * (C + D)

```
LOAD  A        /*   AC ← M[A]              */
ADD   B        /*   AC ← AC + M[B]         */
STORE T        /*   M[T] ← AC              */
LOAD  C        /*   AC ← M[C]              */
ADD   D        /*   AC ← AC + M[D]         */
MUL   T        /*   AC ← AC * M[T]         */
STORE X        /*   M[X] ← AC              */
```

Các lệnh 0 địa chỉ

Ta có thể tìm thấy các lệnh này trong các CPU tổ chức theo ngăn xếp.

Chương trình tính: X = (A + B) * (C + D)

```
PUSH  A        /*   TOS ← M[A]             */
PUSH  B        /*   TOS ← M[B]             */
ADD            /*   TOS ← M[A] + M[B]      */
PUSH  C        /*   TOS ← M[C]             */
PUSH  D        /*   TOS ← M[D]             */
ADD            /*   TOS ← M[C] + M[D]      */
MUL            /* TOS ← (M[A] + M[B]) * (M[C] + M[D])   */
POP   X        /*   M[X] ← TOS             */
```

Các cách định địa chỉ (Addressing modes)

Cách định địa chỉ (còn gọi là cách định vị địa chỉ)

- Cách định địa chỉ cho biết các quy tắc để diễn dịch hay sửa đổi vùng địa chỉ của lệnh (trước khi toán hạng được tham chiếu thật sự).
- Có nhiều cách định địa chỉ để:
  - cho người sử dụng lập trình linh hoạt.
  - sử dụng các bit trong vùng địa chỉ một cách hữu hiệu.
- Sau đây chúng ta sẽ khảo sát các cách định địa chỉ thông dụng trong các CPU.

Cách định địa chỉ hiểu ngầm (Implied addressing mode) hay hàm ý

- Địa chỉ của các toán hạng được hiểu ngầm trong lệnh, như vậy không cần địa chỉ cho toán hạng đó trong lệnh.
- Nếu gọi tắt địa chỉ thật là EA (Effective Address) thì với các lệnh đó thường

  EA = AC hay EA = Stack[SP] (nghĩa là đỉnh ngăn xếp)

Cách định địa chỉ tức thời (Immediate addressing mode)

- Thay vì chỉ ra địa chỉ của toán hạng, với cách này thì toán hạng có sẵn trong lệnh. Do đó:
  - Không cần có địa chỉ của toán hạng đó trong lệnh.
  - Tuy nhiên phải chỉ rõ giá trị toán hạng trong lệnh.
  - Đôi khi cần nhiều bit hơn số bit dành cho địa chỉ.
  - Nhanh chóng có được giá trị toán hạng.

Cách định địa chỉ thanh ghi (Register addressing mode)

Vùng địa chỉ trong lệnh chứa địa chỉ của thanh ghi của CPU.

- Toán hạng cần lấy phải là thanh ghi.
- Địa chỉ ngắn hơn địa chỉ bộ nhớ.
- Tiết kiệm vùng địa chỉ trong lệnh.
- Nhận được toán hạng nhanh hơn định địa chỉ bộ nhớ.
- Địa chỉ thật EA = IR(R) (IR=thanh ghi lệnh; IR(R): vùng thanh ghi của IR)

Cách định địa chỉ gián tiếp qua thanh ghi (Register Indirect addressing mode)

Trong lệnh chỉ ra thanh ghi chứa địa chỉ bộ nhớ của toán hạng.

- Tiết kiệm số bit trong lệnh vì địa chỉ thanh ghi ngắn hơn địa chỉ bộ nhớ.
- Nhận toán hạng chậm hơn định địa chỉ thanh ghi hay định địa chỉ bộ nhớ.
- Địa chỉ thật EA = [IR(R)] (với [x] là nội dung của x).

Thanh ghi được sử dụng trong cách này có thể có thêm đặc tính tự động tăng (thêm 1) hoặc tự động giảm (bớt 1), đặc tính này đặc biệt có lợi khi dùng thanh ghi để truy cập bộ nhớ, giá trị trong thanh ghi được tăng (thêm 1) hoặc giảm (bớt 1) một cách tự động.

Cách định địa chỉ trực tiếp (Direct addressing mode)

Trong lệnh chứa địa chỉ bộ nhớ mà có thể được sử dụng trực tiếp với bộ nhớ thật.

- Nhanh hơn các cách định địa chỉ bộ nhớ khác.
- Với vùng nhớ thật lớn thì cần quá nhiều bit cho địa chỉ.
- Địa chỉ thật EA = IR(addr) (IR(addr)=vùng địa chỉ của IR)

Cách định địa chỉ gián tiếp (Indirect addressing mode)

Vùng địa chỉ của lệnh chỉ ra địa chỉ của ô nhớ mà chứa địa chỉ của toán hạng.

- Khi sử dụng địa chỉ viết gọn có thể định địa chỉ cho vùng bộ nhớ lớn với số bit tương đối nhỏ.

- Làm chậm việc nhận được toán hạng vì phải truy cập thêm bộ nhớ.

- Địa chỉ thật EA = M[IR(addr)]

Các cách định địa chỉ tương đối (Relative addressing modes)

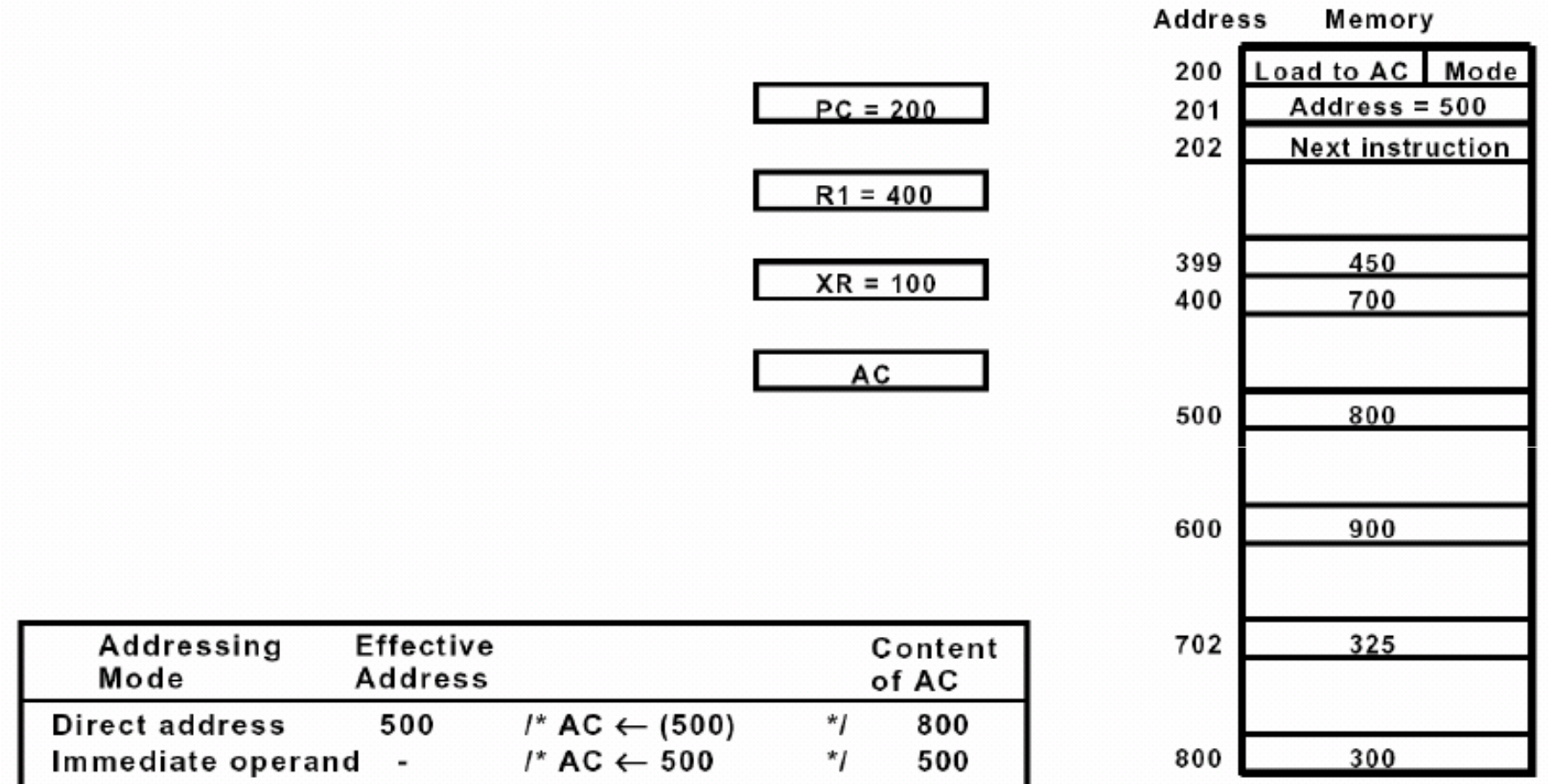
Vùng địa chỉ trong lệnh chỉ ra phần địa chỉ (địa chỉ viết gọn) mà có thể được sử dụng cùng với thanh ghi có đề cập trong lệnh để tính ra địa chỉ thật của toán hạng.

- Vùng địa chỉ trong lệnh ngắn.
- Có thể truy cập vùng bộ nhớ lớn với số bit cho địa chỉ nhỏ.
- Địa chỉ thật EA = f(IR(addr), R) (với R đôi khi được hiểu ngầm)

Có 3 cách định địa chỉ tương đối khác nhau tùy theo R:

1. ***Cách định địa chỉ tương đối PC*** (R = PC): EA = PC + IR(addr)
2. ***Cách định địa chỉ theo chỉ số*** (Indexed Addressing mode) (R=IX, với IX là thanh ghi chỉ số): EA = IX + IR(addr)
3. ***Cách định địa chỉ với thanh ghi nền*** (Base Register Addressing mode) (R=BAR, với BAR là thanh ghi địa chỉ nền [Base Address Register]):
   EA = BAR + IR(addr)

Các thí dụ về các cách định địa chỉ



Addressing Mode	Effective Address		Content of AC
Direct address	500	/* AC ← (500) */	800
Immediate operand	-	/* AC ← 500 */	500
Indirect address	800	/* AC ← ((500)) */	300
Relative address	702	/* AC ← (PC+500) */	325
Indexed address	600	/* AC ← (RX+500) */	900
Register	-	/* AC ← R1 */	400
Register indirect	400	/* AC ← (R1) */	700
Autoincrement	400	/* AC ← (R1)+ */	700
Autodecrement	399	/* AC ← -(R) */	450

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

1.4 BỘ NHỚ

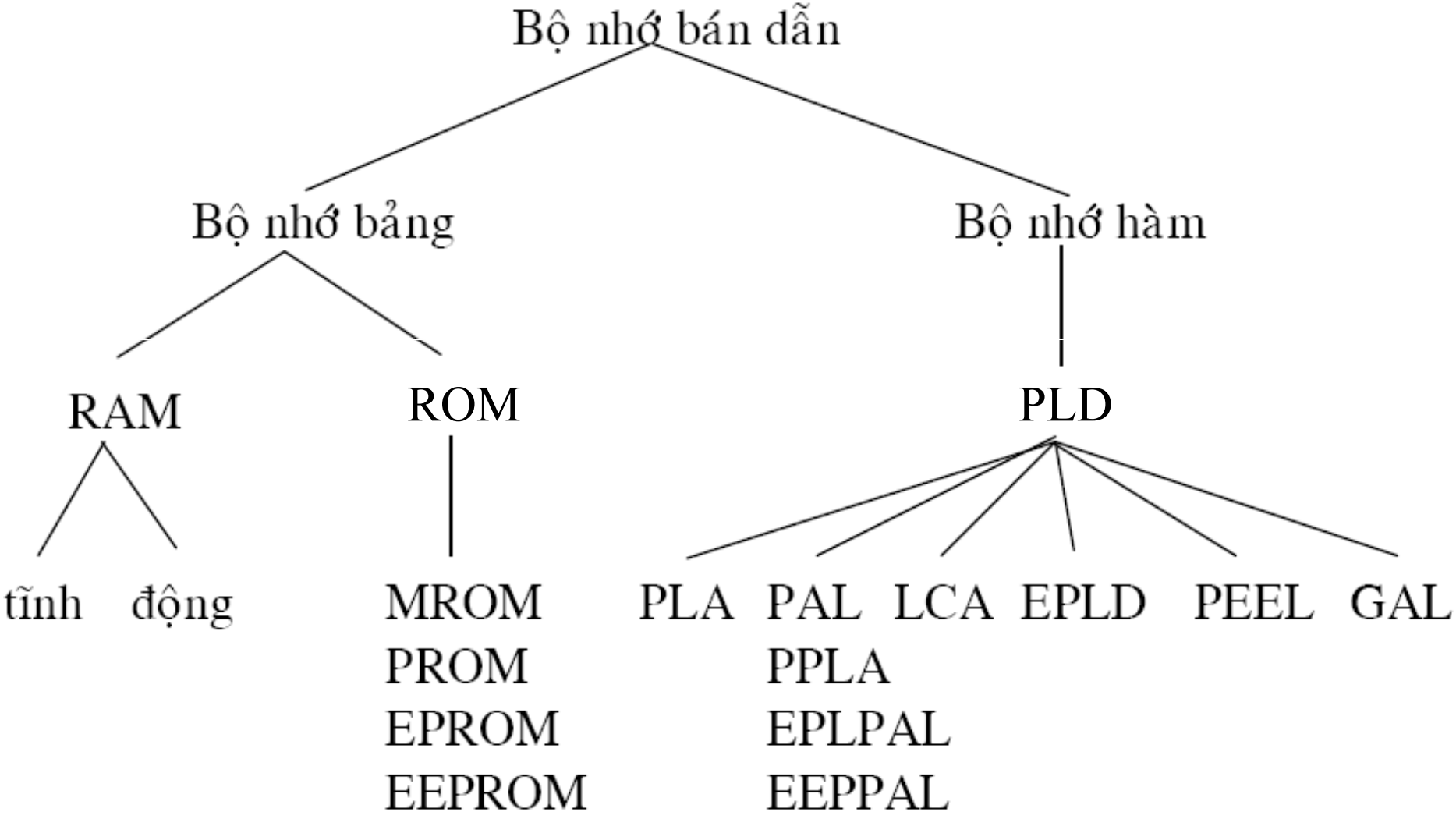
Bit, Byte và word

Với thời đại số hiện nay thì thông tin được sinh ra, truyền đi và lưu trữ dưới dạng nhị phân.

- **Bit** là đơn vị cơ bản của thông tin nhị phân (bit=**b**inary dig**it**). Nó lấy giá trị 0 hoặc 1. Trong các máy tính số thì bit được truyền đi qua kết nối điện và được lưu trữ trong tế bào nhớ.
- **Byte** là đơn vị lớn hơn gồm 8 bit.
- **Word** là nhóm gồm nhiều byte (tùy theo quy ước có thể số byte là 1, 2, 4, 8,...).

Theo quy ước thông thường thì word gồm 2 byte (hay 16 bit) và word dài gồm 4 byte (32 bit).

Các loại bộ nhớ



Các loại bộ nhớ

- RAM= Random Access Memory (bộ nhớ truy cập ngẫu nhiên)
- SRAM (S=Static), DRAM (D=Dynamic)
- ROM= Read Only Memory
- M= Mask Programmed (được lập trình bằng che mặt nạ)
- P = Programmable (lập trình được, khả lập trình)
- EP = Erasable and Programmable
- EEP = Electrically Erasable and Programmable (xóa và lập trình bằng điện) (E^2ROM)
- PLD = Programmable Logic Device
- PLA = Programmable Logic Array (mảng logic lập trình được)
- PAL = Programmable Array Logic (logic mảng lập trình được)
- LCA = Logic Cell Array (Mảng tế bào logic)

Dung lượng bộ nhớ

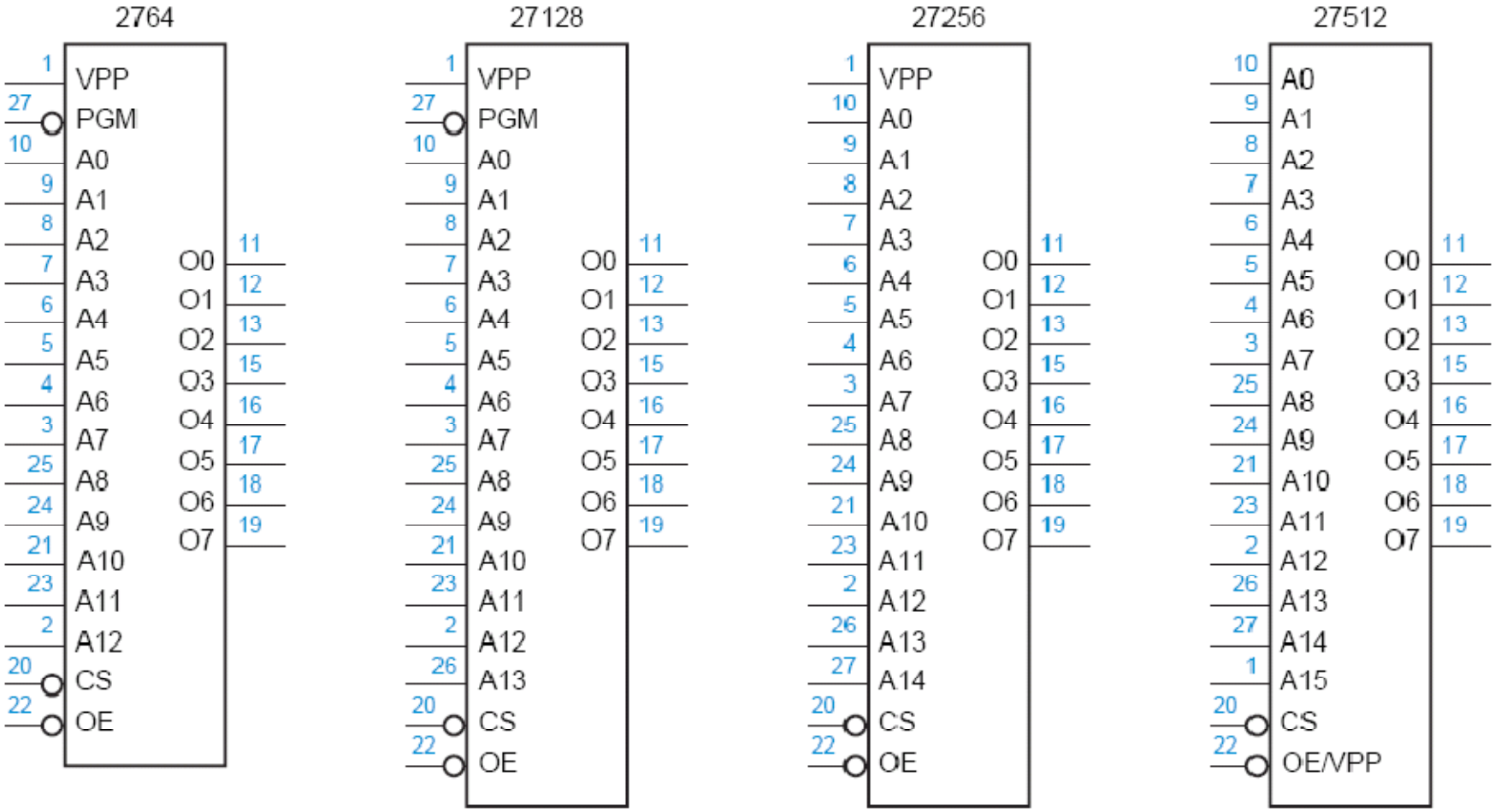
- Một bộ nhớ có độ rộng dữ liệu m bit với N đường địa chỉ thì sẽ có dung lượng (tính theo bit) là 2^N x m.
- Dung lượng nhớ cũng được tính theo kilobyte (KB), megabyte (MB) và gigabyte (GB) (với m=8)

  $1K=2^{10}=1024$

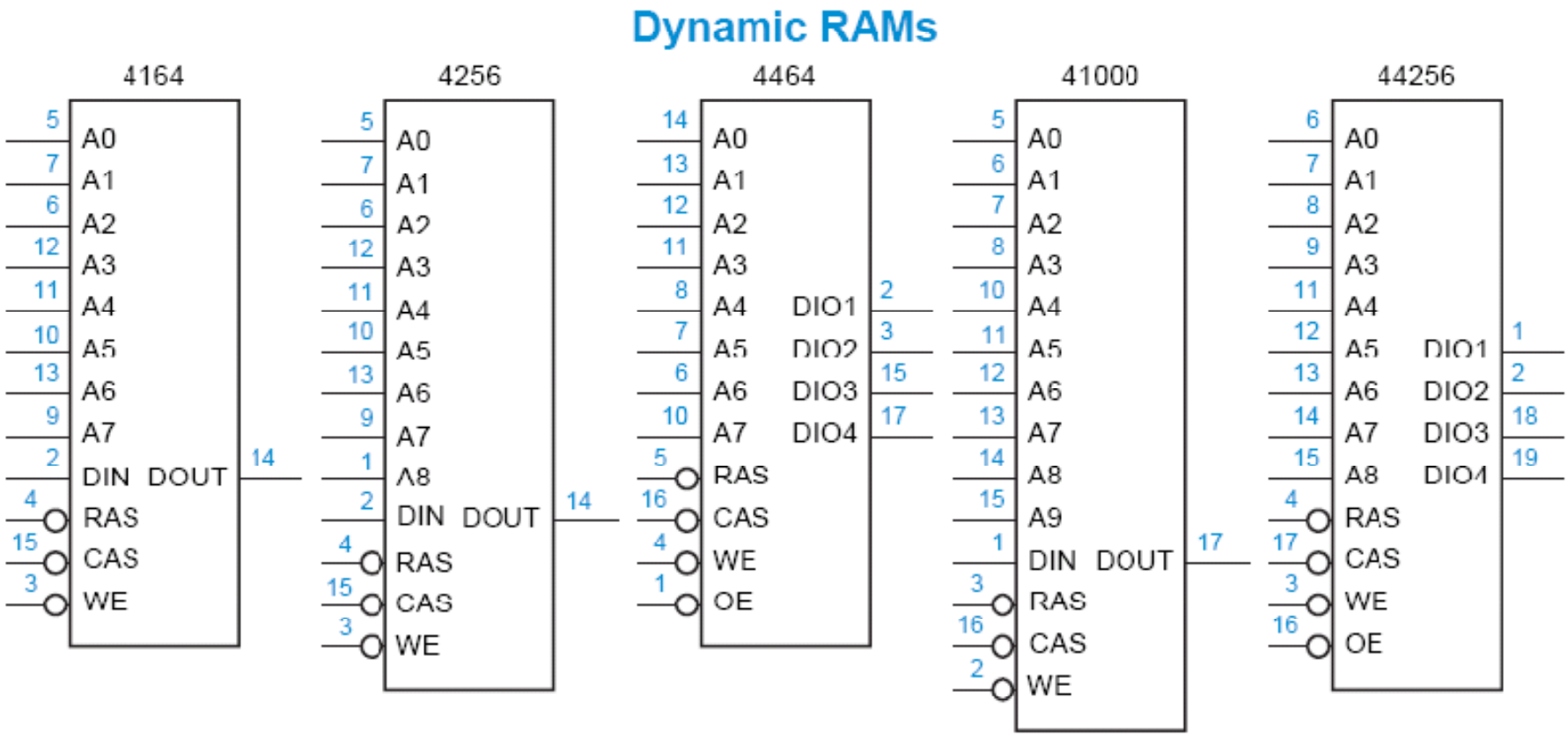
  $1M=2^{20}=1024K$

  $1G=2^{30}=1024M=2^{20}K$

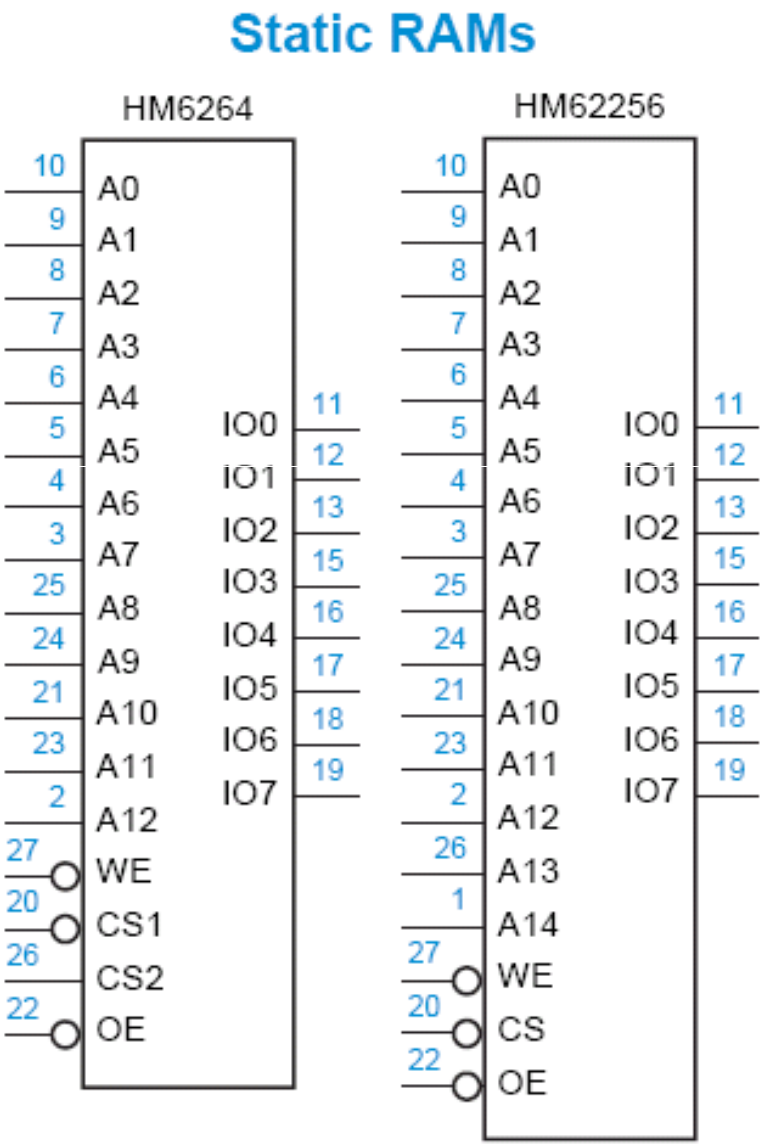
Các IC ROM thông dụng



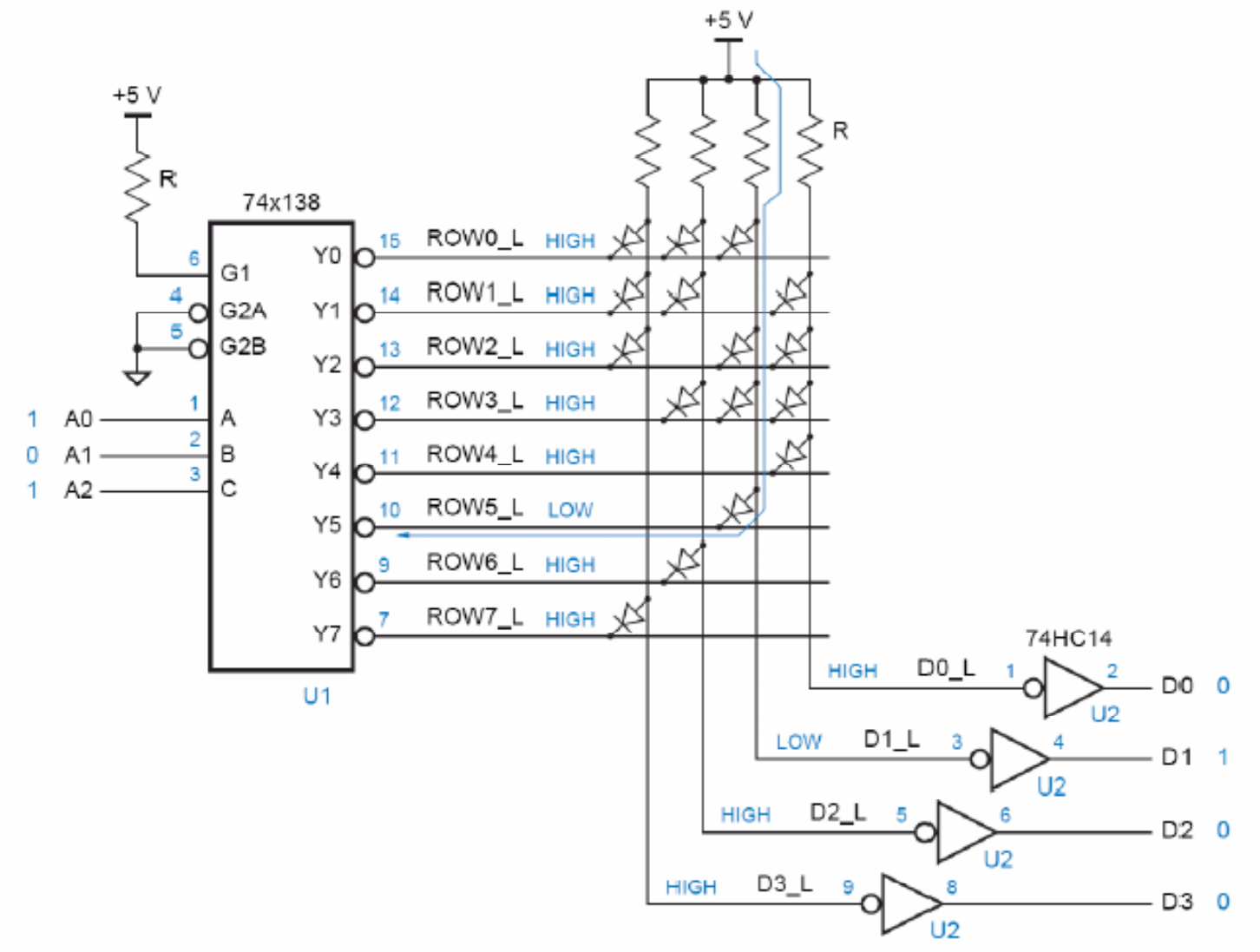
Một số IC DRAM



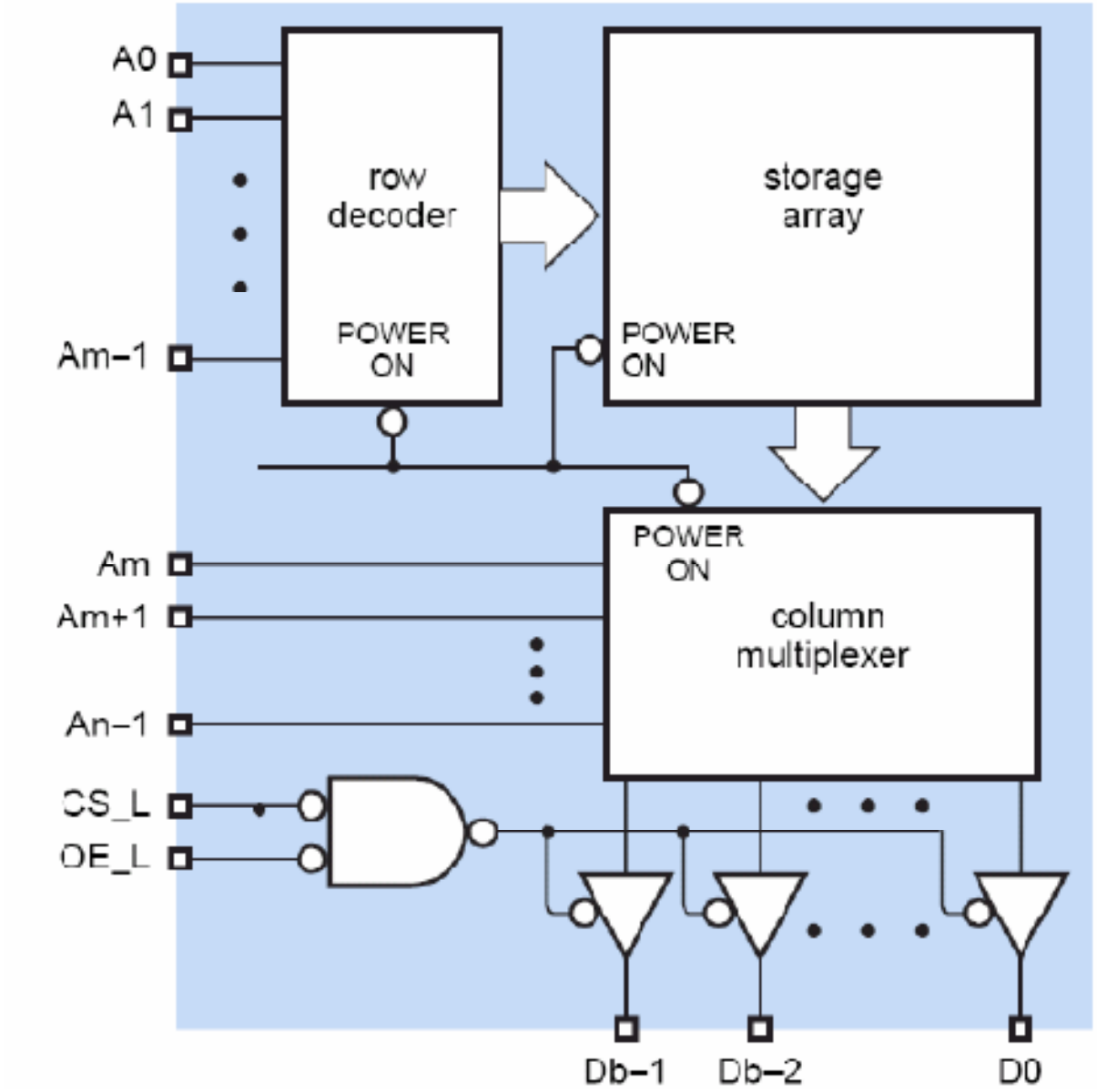
Một số IC SRAM



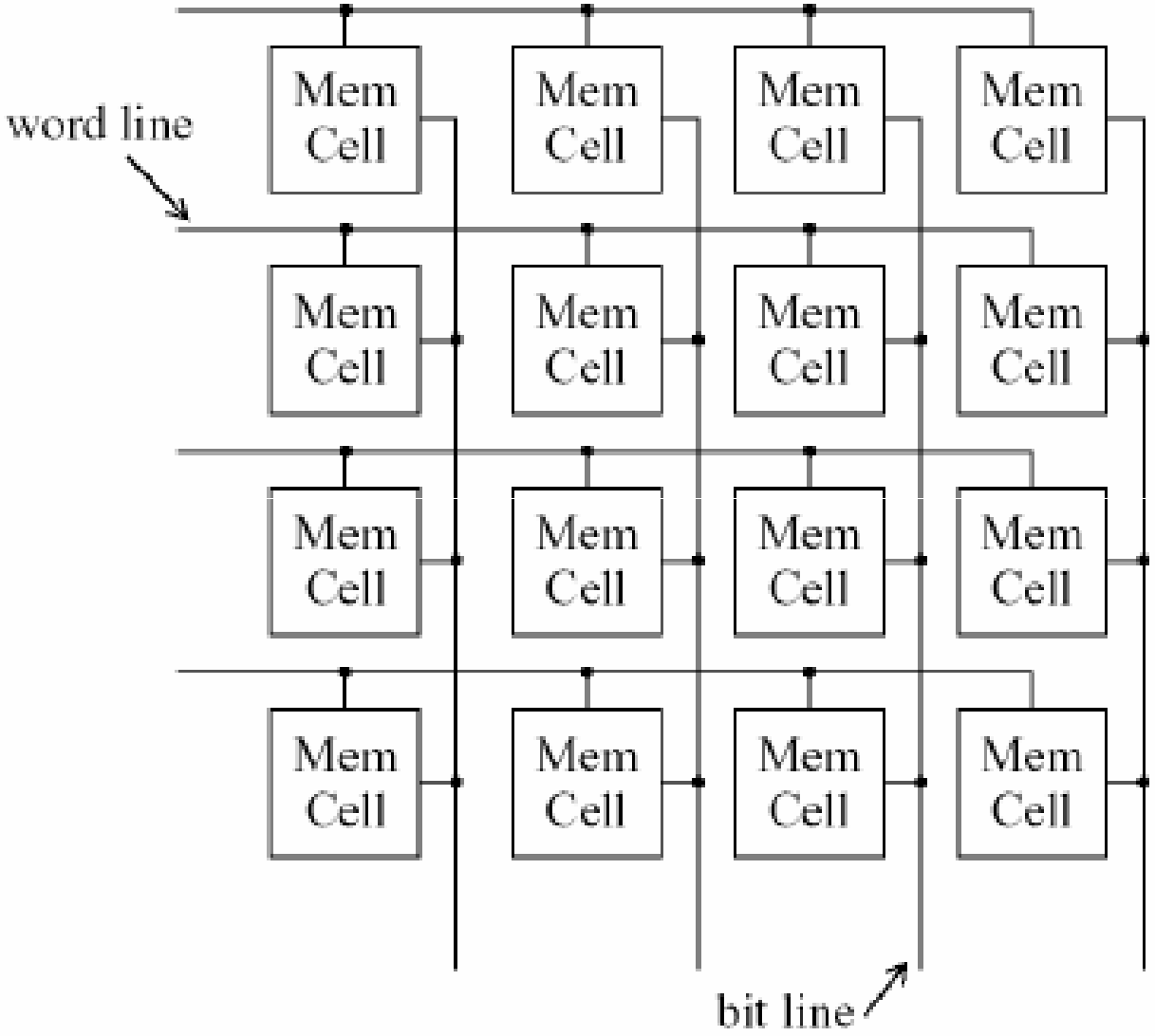
ROM 8 x 4 đơn giản



Cấu trúc ROM nội và tác dụng của các ngõ vào điều khiển

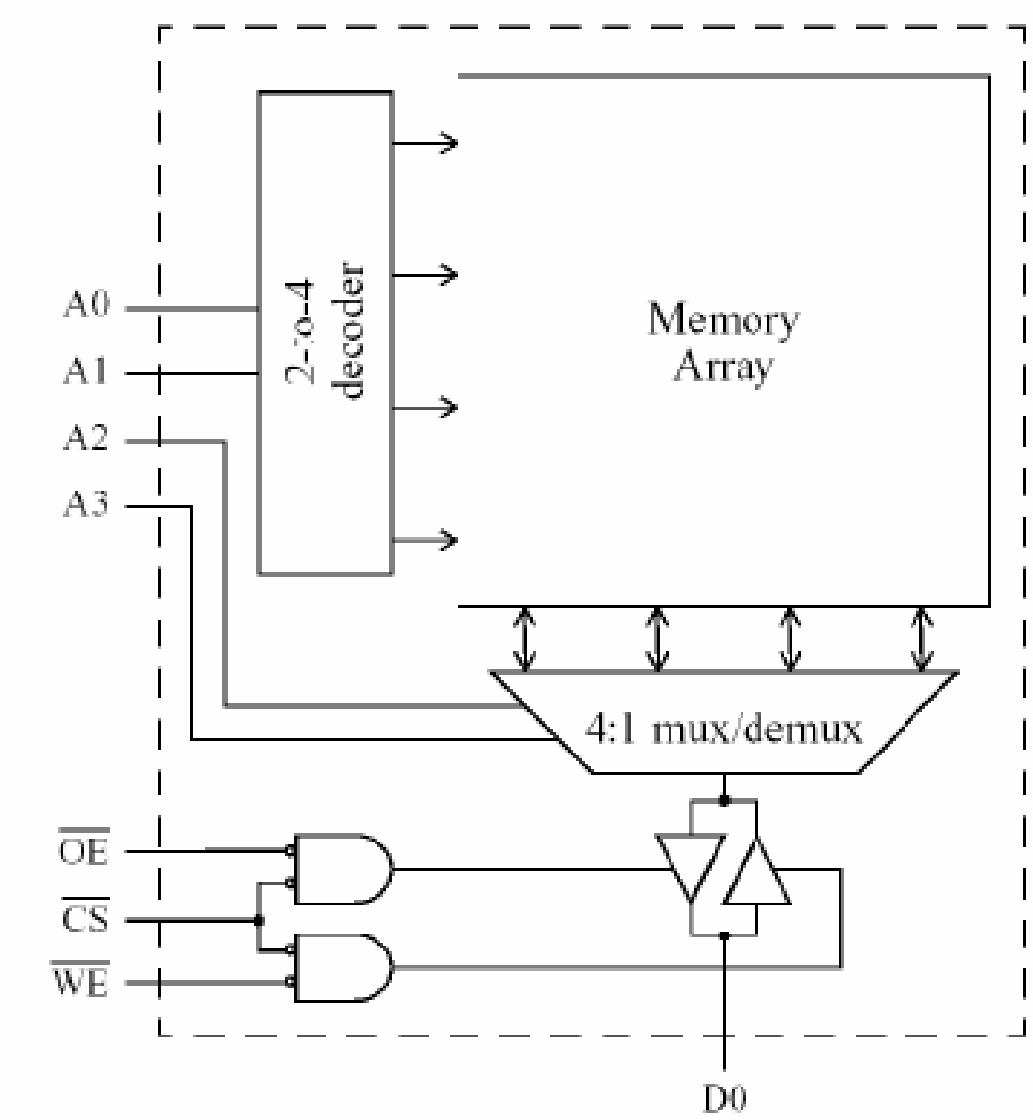


Mảng bộ nhớ (Memory array)



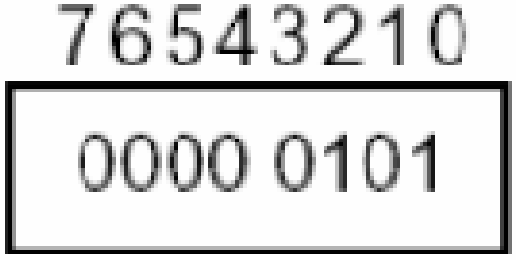
Tổ chức mảng 4 x 4 của bộ nhớ 16 bit

Mạch hỗ trợ cho bộ nhớ 16 x 1 với mảng bộ nhớ 4 x 4

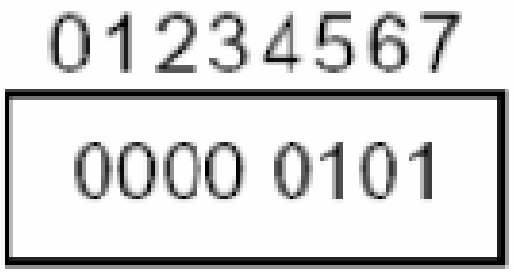


Little Endian và Big Endian

- Đánh số thứ tự từ phải sang trái (từ MSB đến LSB) gọi là ***little endian***
- Đánh số thứ tự từ trái sang phải (từ LSB đến MSB) gọi là ***big endian***



(a) Little Endian



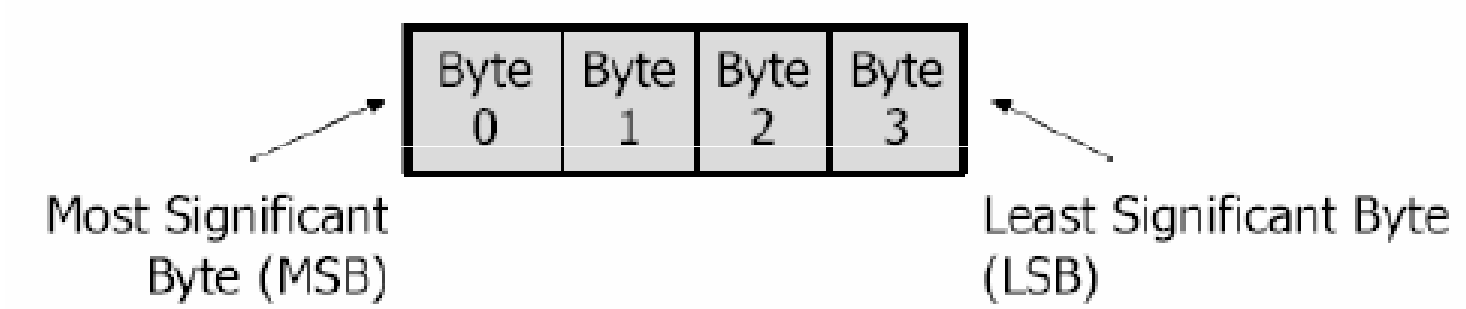
(b)Big Endian

Little Endian và Big Endian (tt)

Khi kể đánh số byte trong word thì ta có đến 4 khả năng:

- **nhất quán với little endian (TD: họ Intel 80x86)**
  đánh số byte từ phải sang trái, đánh số bit từ phải sang trái.
- nhất quán với big endian (TD: PDP11, TI 9900)
  đánh số byte từ trái sang phải, đánh số bit từ trái sang phải.
- không nhất quán với little endian
  đánh số byte từ phải sang trái, đánh số bit từ trái sang phải.
- **không nhất quán với big endian (TD: Motorola 68000)**
  đánh số byte từ trái sang phải, đánh số bit từ phải sang trái.

Thí dụ: thứ tự byte trong little endian và big endian

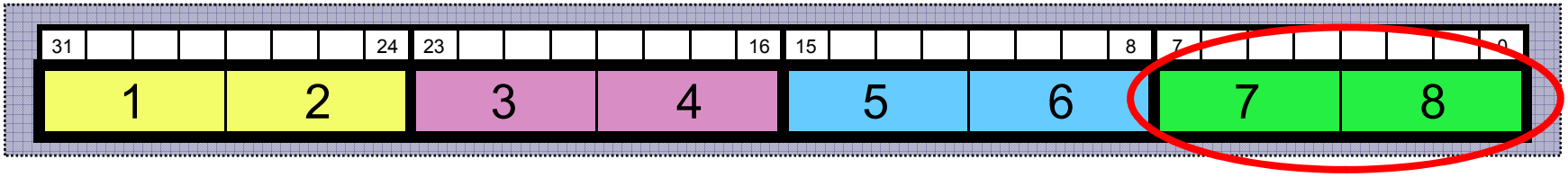


Địa chỉ bộ nhớ	000	001	002	003	
Big Endian	Byte 0	Byte 1	Byte 2	Byte 3	MSB ở địa chỉ bộ nhớ thấp nhất
Little Endian	Byte 3	Byte 2	Byte 1	Byte 0	MSB ở địa chỉ bộ nhớ cao nhất

One Annoying Thing: Byte Order

- Hosts differ in how they store data
  - E.g., four-byte number (byte3, byte2, byte1, byte0)
- **Little endian** ("little end comes first") ← Intel PCs!!!
  - Low-order byte stored at the lowest memory location
  - Byte0, byte1, byte2, byte3
- **Big endian** ("big end comes first")
  - High-order byte stored at lowest memory location
  - Byte3, byte2, byte1, byte 0
- Makes it more difficult to write portable code
  - Client may be big or little endian machine
  - Server may be big or little endian machine

Endian Example: Where is the Byte?



	8 bits memory		16 bits Memory			32 bits Memory			
				+1	+0		+3	+2	+1	+0
Little-Endian	1000	78	1000		78	1000				78
	1001	56	1002			1004				
	1002		1004			1008				
	1003		1006			100C				
				+0	+1		+0	+1	+2	+3
Big-Endian	1000	78	1000	78		1000	78			
	1001		1002			1004				
	1002		1004			1008				
	1003		1006			100C				

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

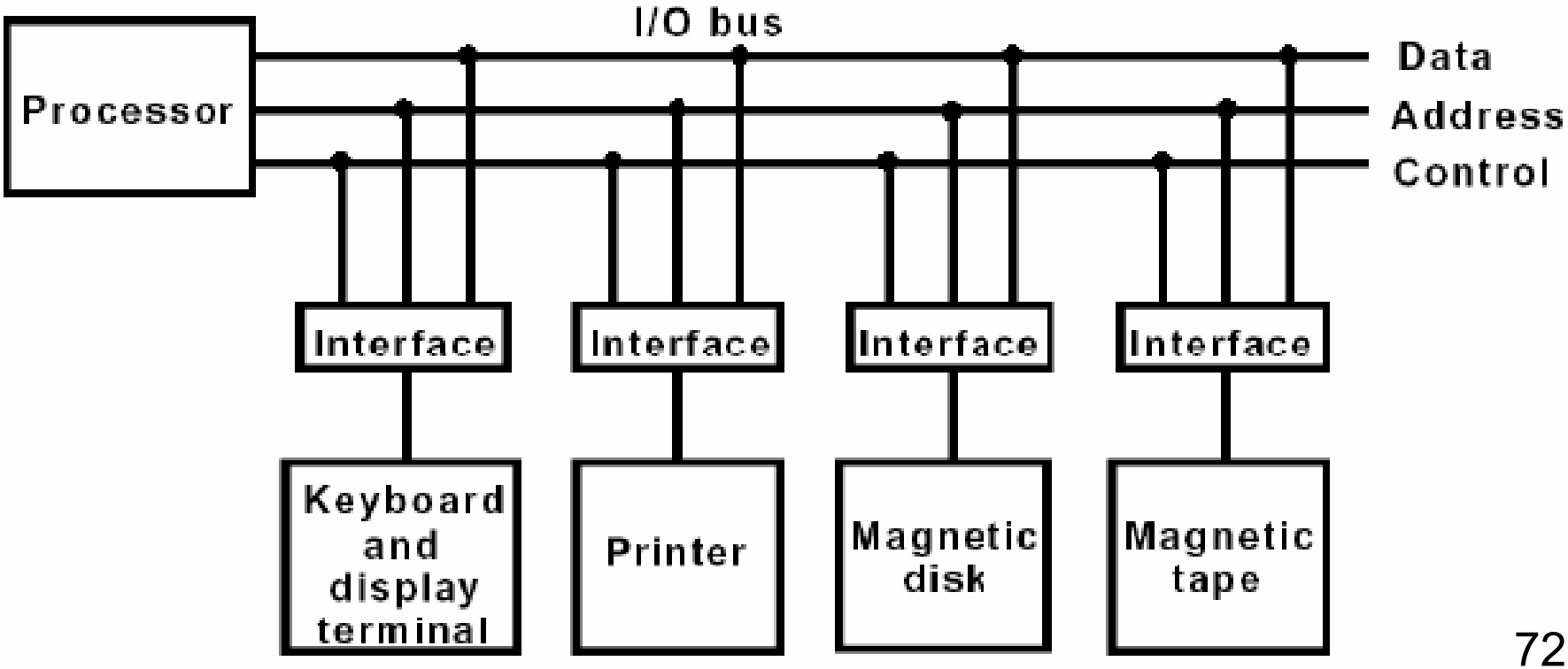
1.5 NGOẠI VI

Phân loại ngoại vi

Các thiết bị nhập	Các thiết bị xuất
Bàn phím	Máy đục phiếu, máy đục băng giấy
Các thiết bị nhập loại quang học -Máy đọc thẻ -Máy đọc băng giấy -Máy đọc mã vạch (bar code) -Máy số hóa dữ liệu (Digitizer)	Màn hình CRT Máy in Máy vẽ Thiết bị xuất analog Tiếng nói
Máy nhập loại từ - Máy đọc thẻ từ	
Thiết bị nhập theo màn hình -Màn hình nhạy với chạm -Bút quang (light pen) -Chuột (mouse)	
Thiết bị nhập analog	

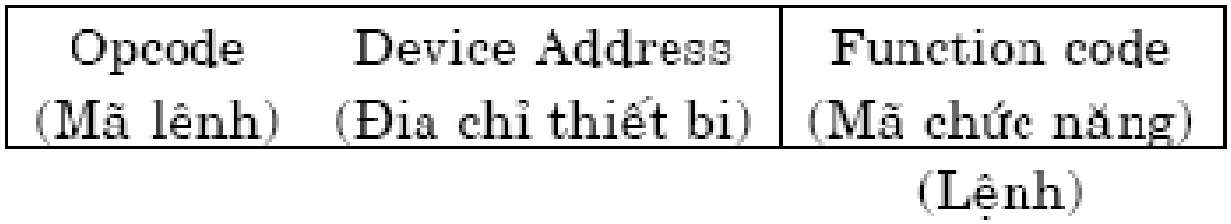
Bus I/O và các module giao tiếp

	CPU hoặc bộ nhớ	Thiết bị ngoại vi
Loại thiết bị	Điện tử	Điện cơ
Tốc độ chuyển dữ liệu	Thường nhanh hơn ngoại vi (do đó cần đồng bộ)	Thường chậm hơn
Đơn vị thông tin	Byte, word	bit, byte
Cách hoạt động	Đồng bộ	Không đồng bộ

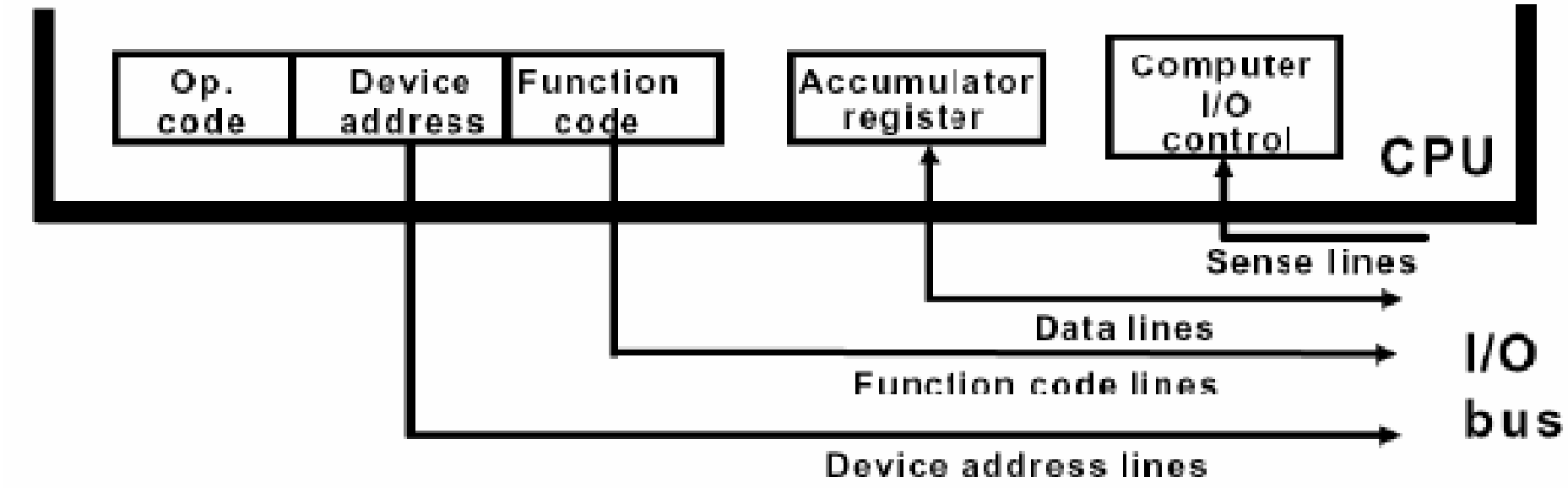


- Mỗi thiết bị ngoại vi có một module giao tiếp tương ứng với nó. Module giao tiếp có nhiệm vụ:
  - Giải mã địa chỉ thiết bị (mã thiết bị)
  - Giải mã các lệnh (tác vụ)
  - Cung cấp các tín hiệu cho bộ điều khiển ngoại vi.
  - Đồng bộ hóa luồng dữ liệu và giám sát tốc độ chuyển dữ liệu giữa ngoại vi với CPU hoặc bộ nhớ.

- Lệnh I/O tiêu biểu có dạng:

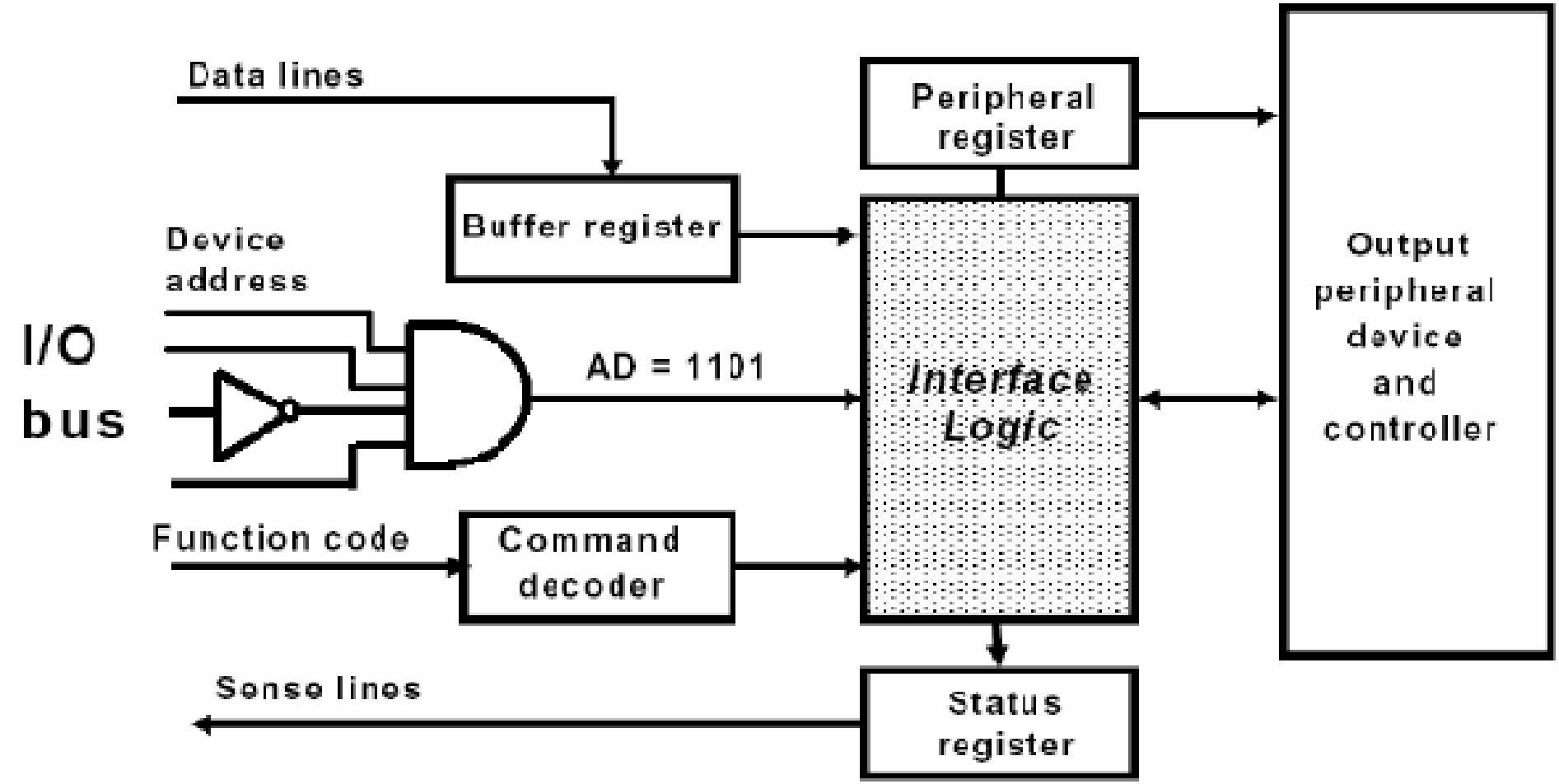


Kết nối bus I/O đến CPU được minh họa ở hình 1.20.



Hình 1.20 Kết nối bus I/O đến CPU

TD: Kết nối bus I/O đến mạch giao tiếp



Bus I/O và bus bộ nhớ (1/2)

- ***Chức năng của các bus***
  - Bus bộ nhớ: dành cho chuyển thông tin giữa CPU và bộ nhớ.
  - Bus I/O: dành cho chuyển thông tin giữa CPU và các thiết bị ngoại vi qua giao tiếp I/O

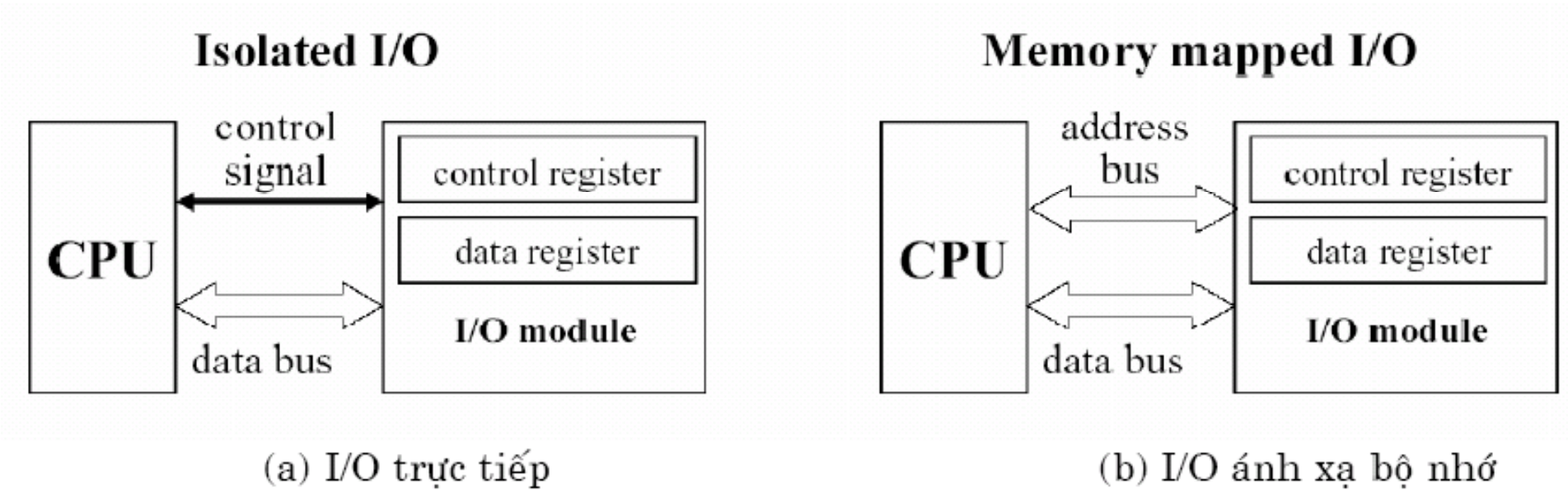
- ***Tổ chức vật lý***
  - Nhiều hệ thống sử dụng chung một bus hệ thống cho cả bộ nhớ và các đơn vị I/O. Sử dụng một bus chung với các đường điều khiển riêng cho mỗi chức năng hoặc dạng có những đường điều khiển chung cho cả 2 chức năng.
  - Một số hệ sử dụng 2 bus riêng, một để liên lạc với bộ nhớ, cái còn lại liên lạc với các giao tiếp I/O.

Bus I/O và bus bộ nhớ (2/2)

Bus I/O

- Liên lạc giữa CPU và tất cả các đơn vị giao tiếp qua bus I/O chung.
- Giao tiếp kết nối với thiết bị ngoại vi có thể có một số các thanh ghi dữ liệu, thanh ghi điều khiển và thanh ghi trạng thái.
- Lệnh được chuyển đến ngoại vi bằng cách gửi đến thanh ghi ngoại thích hợp.
- Có thể không cần mã chức năng và các đường dò (cảm nhận) (sense lines) (chuyển dữ liệu, điều khiển và thông tin trạng thái luôn luôn qua bus I/O chung).

I/O cách ly và I/O ánh xạ bộ nhớ (Isolated I/O and memory-mapped I/O)

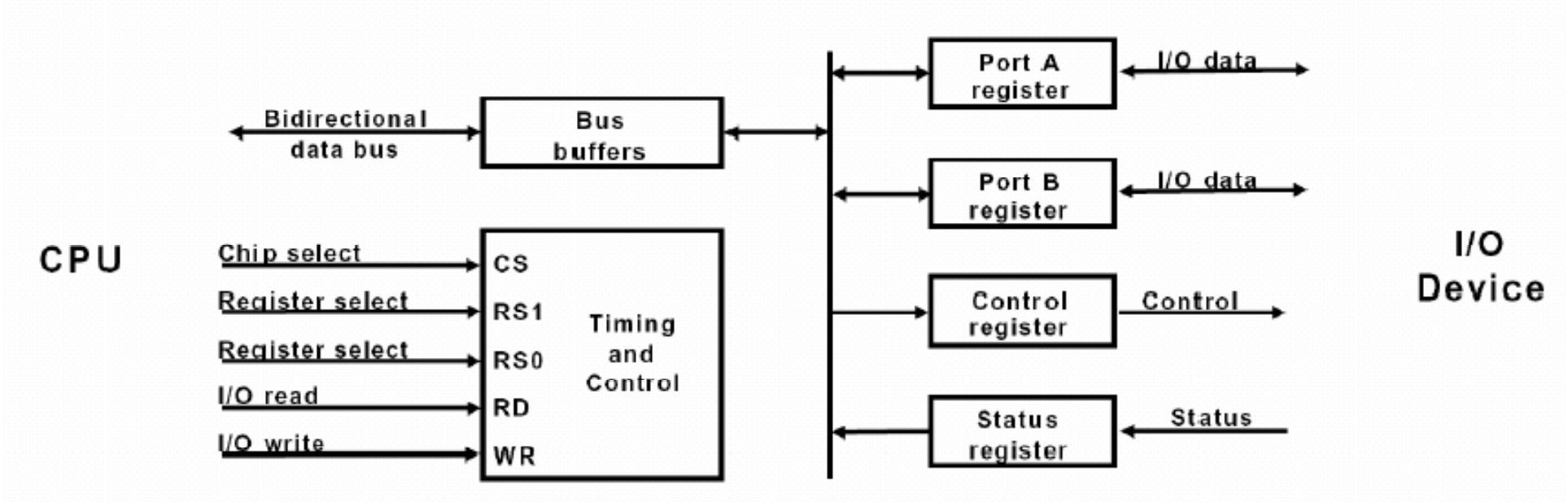


(a) I/O trực tiếp (b) I/O ánh xạ bộ nhớ

Hình 1.22 I/O trực tiếp và I/O ánh xạ bộ nhớ

Giao tiếp I/O lập trình được

CS	RS1	RS0	Thanh ghi được chọn
0	X	X	Không chọn thanh ghi nào cả Ngõ ra bus dữ liệu ở trạng thái hi-Z
1	0	0	Thanh ghi Port A
1	0	1	Thanh ghi Port B
1	1	0	Thanh ghi điều khiển
1	1	1	Thanh ghi trạng thái



Các vấn đề chuyển dữ liệu

Hoạt động đồng bộ và bất đồng bộ

– ***đồng bộ (synchronous)***:

Tất cả các thiết bị có được thông tin định thì từ đường xung nhịp chung.

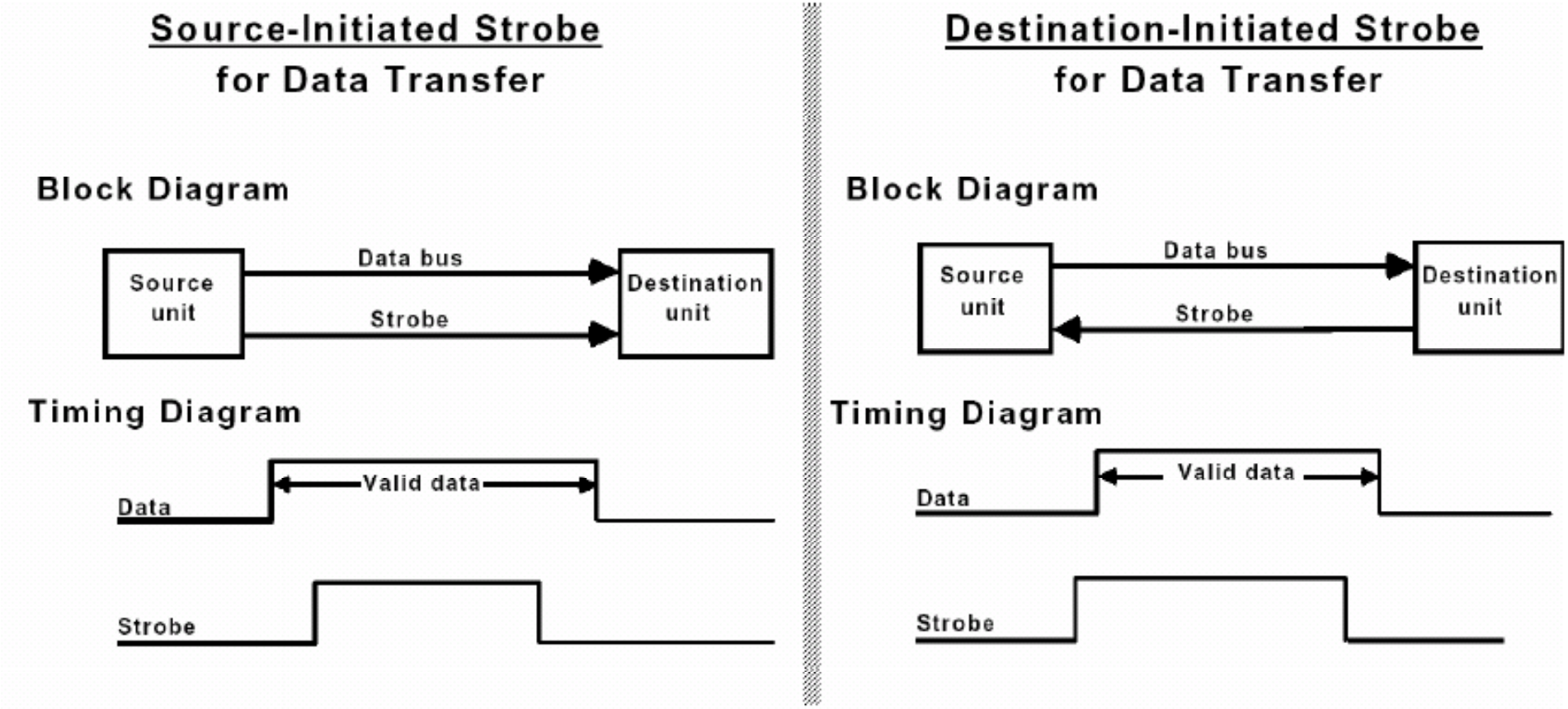
– ***bất đồng bộ (asynchronous)***:

Không có xung nhịp chung.

Chuyển dữ liệu bất đồng bộ

- Chuyển dữ liệu bất đồng bộ giữa 2 đơn vị độc lập cần có các tín hiệu điều khiển truyền giữa các đơn vị truyền thông để chỉ thời điểm mà dữ liệu sẽ được truyền.
- Có hai phương pháp chuyển dữ liệu bất đồng bộ:
  - ***Xung strobe***: Xung strobe được cung cấp bởi một đơn vị này để báo cho đơn vị kia khi việc chuyển dữ liệu xảy ra.
  - ***Thực hiện bắt tay (Handshaking)***: Một tín hiệu điều khiển được đi kèm với mỗi dữ liệu sẽ được truyền để chỉ sự hiện diện của dữ liệu. Đơn vị nhận trả lời với tín hiệu điều khiển khác để báo nhận được dữ liệu rồi.

Chuyển dữ liệu bất đồng bộ với xung strobe

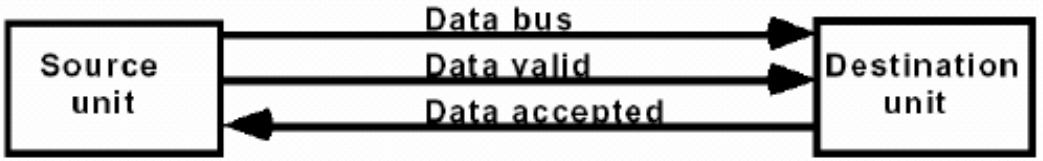


(a) Xung strobe từ đơn vị nguồn (b) Xung strobe từ đơn vị đích

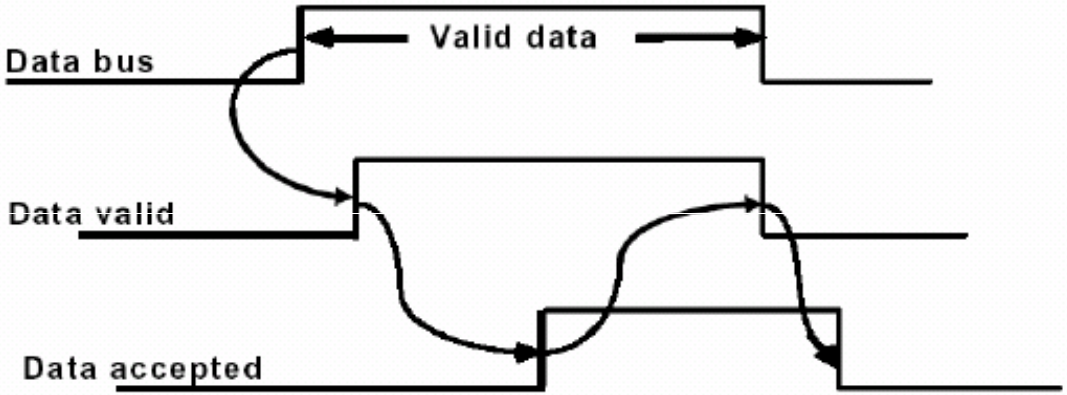
Hình 1.24 Điều khiển Strobe

Chuyển dữ liệu khởi động bởi đơn vị nguồn với phương pháp bắt tay

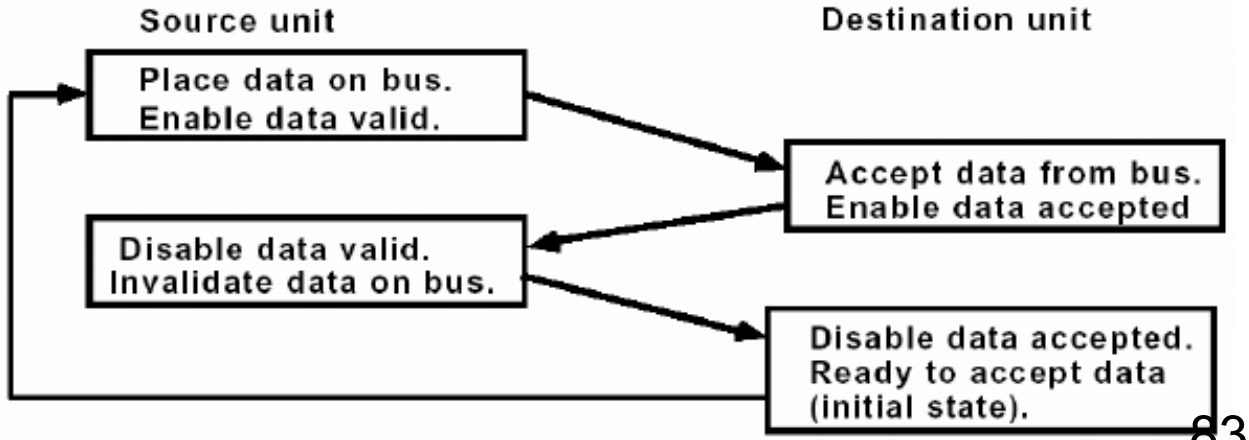
(a) Sơ đồ khối



(b) Giản đồ định thì

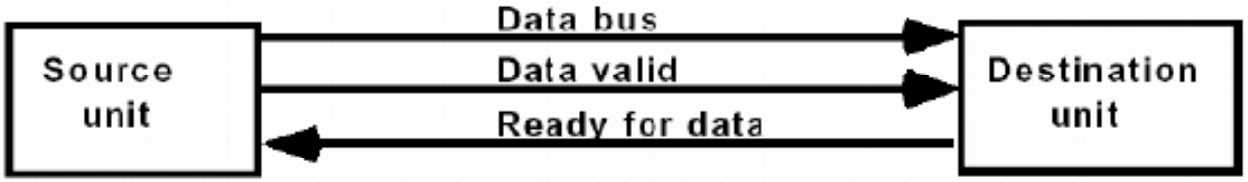


(c) Trình tự các sự kiện

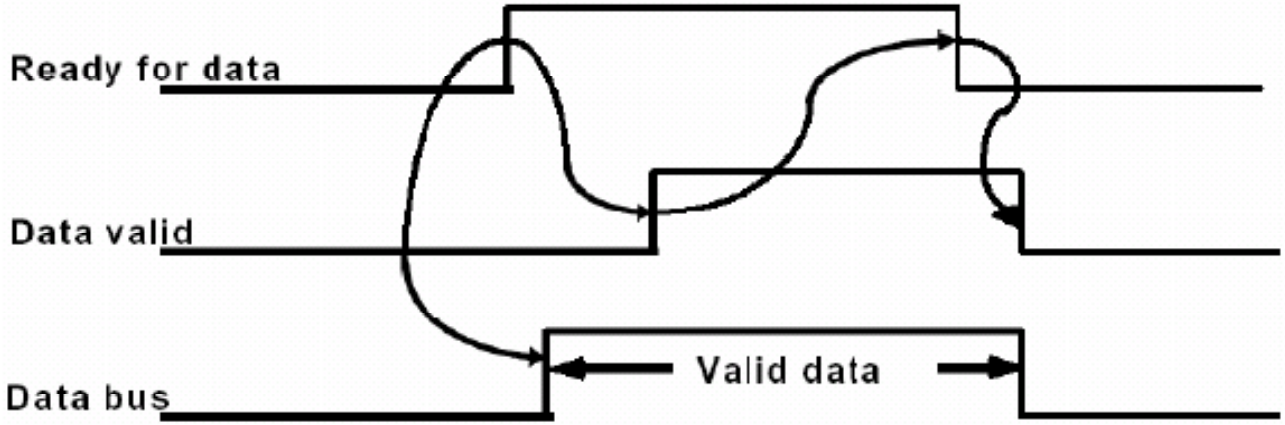


Chuyển dữ liệu khởi động bởi đơn vị đích với phương pháp bắt tay

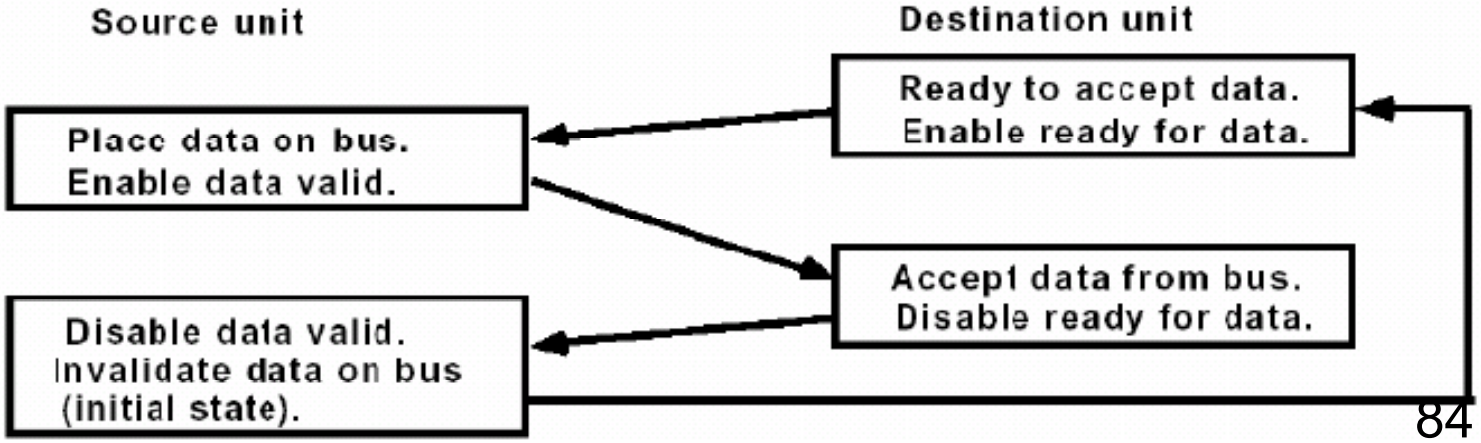
(a) Sơ đồ khối



(b) Giản đồ định thì



(c) Trình tự các sự kiện



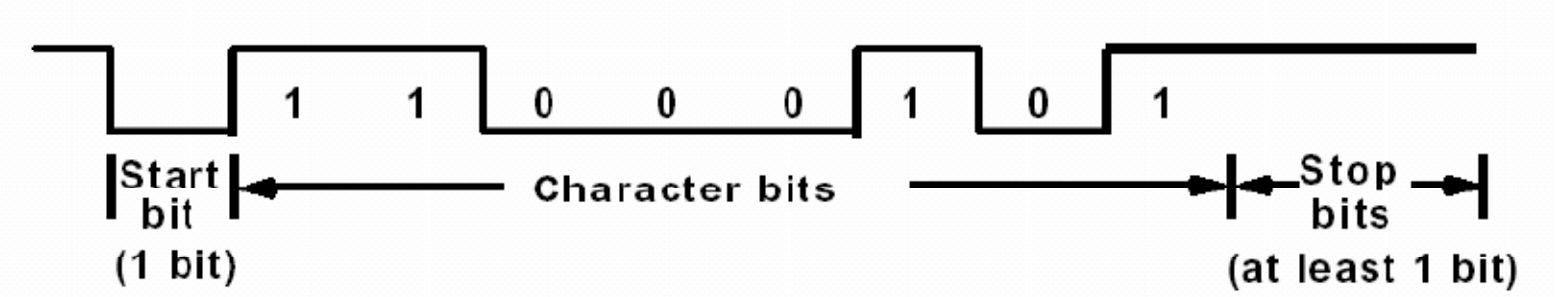
Truyền dữ liệu nối tiếp bất đồng bộ (Asynchronous serial transfer)

- Có 4 loại truyền dữ liệu:
  - Truyền nối tiếp bất đồng bộ.
  - Truyền nối tiếp đồng bộ.
  - Truyền song song bất đồng bộ.
  - Truyền song song đồng bộ.

- ***Truyền nối tiếp bất đồng bộ***

- Sử dụng các bit đặt biệt chèn vào cả 2 đầu của mã ký tự.
- Mỗi ký tự bao gồm 3 phần: bit bắt đầu (Start bit), các bit dữ liệu (Data bits) và các bit dừng (Stop bits) (có thể 1, 1½ hoặc 2 bit dừng) .

Truyền dữ liệu nối tiếp bất đồng bộ



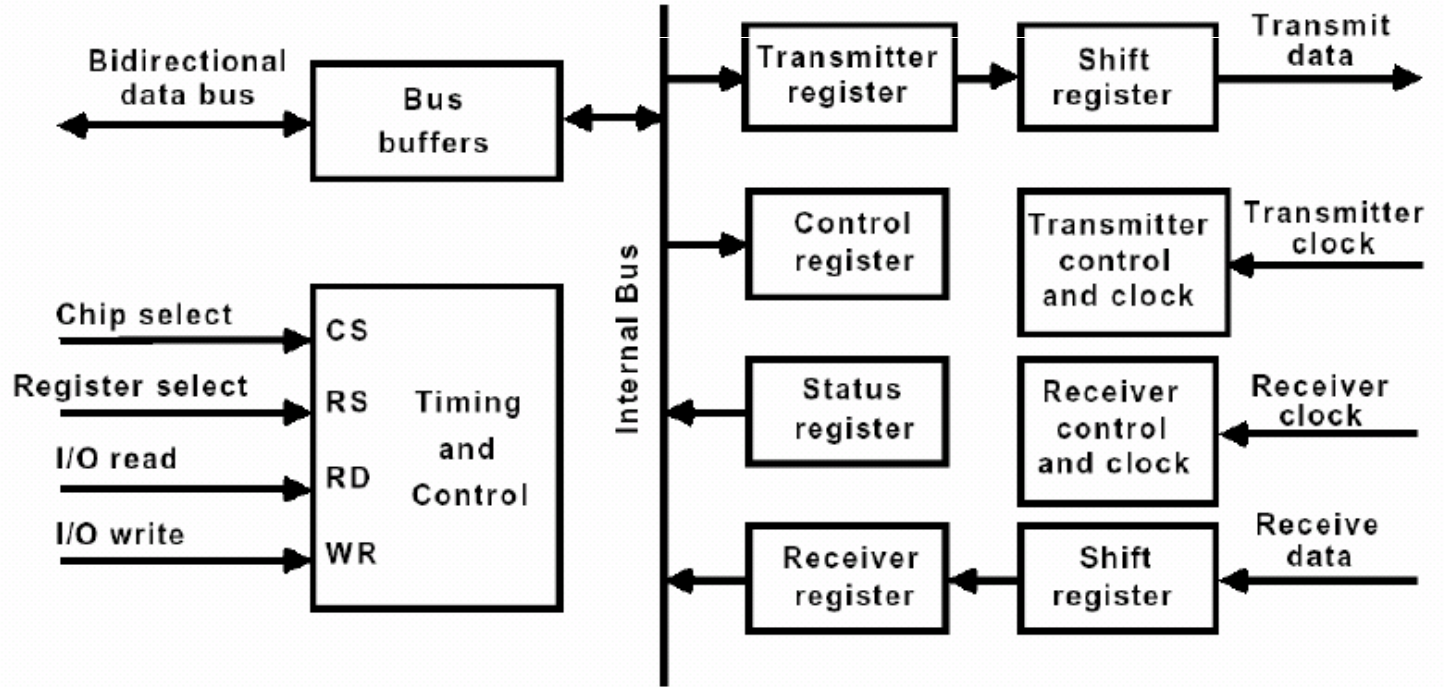
Bộ thu có thể phát hiện được ký tự dựa trên 4 quy tắc sau:

1. Khi dữ liệu không được gửi đi, đường dây được giữ ở trạng thái 1 (trạng thái nghỉ).
2. Khởi động truyền ký tự được phát hiện bởi "Start bit" (luôn luôn bằng 0).
3. Các bit ký tự luôn luôn theo sau "Start bit".
4. Sau khi hết ký tự, "Stop bit" được phát hiện khi đường dây quay về trạng thái 1 trong khoảng thời gian ít nhất 1 bit.

Bộ thu phải biết trước tốc truyền các bit và số bit thông tin mong muốn.

Bộ thu phát bất đồng bộ vạn năng UART (Universal Asysnchronous Receiver-Transmitter)

CS	RS	Hoạt động	Thanh ghi được chọn
0	X	X	Không chọn thanh ghi nào cả
1	0	WR	Thanh ghi bộ phát
1	1	WR	Thanh ghi điều khiển
1	0	RD	Thanh ghi bộ thu
1	1	RD	Thanh ghi trạng thái



Hình 1.27 Sơ đồ khối một UART tiêu biểu.

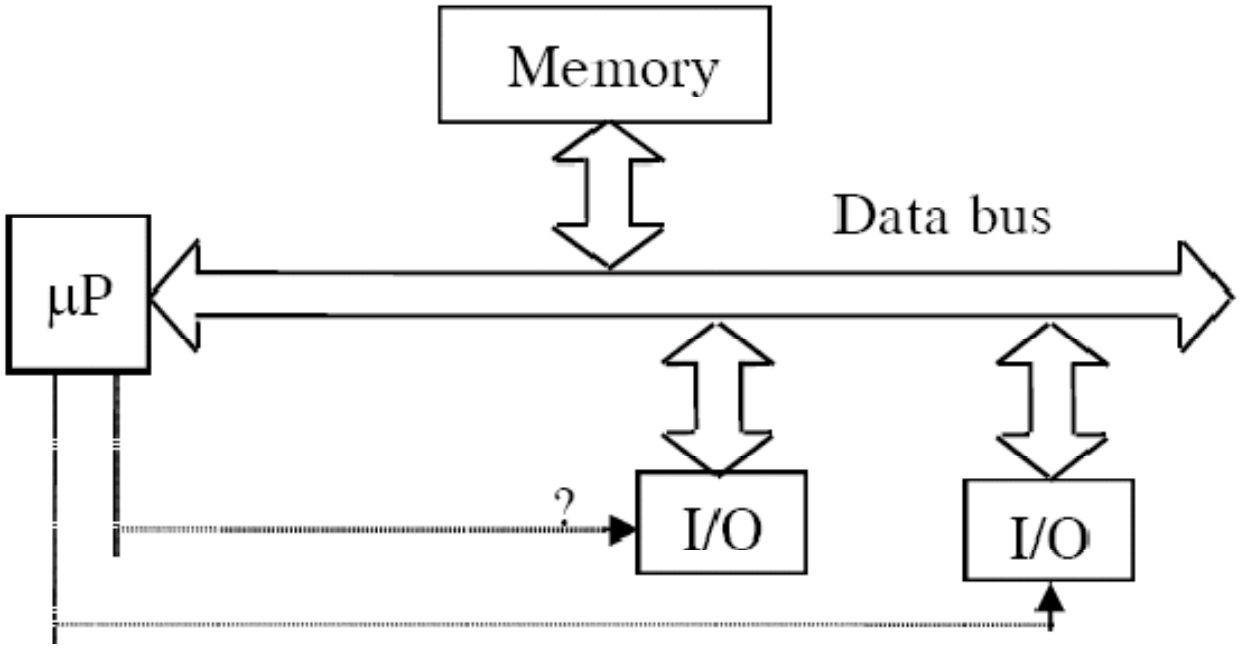
Các phương pháp điều khiển I/O

Có 3 phương pháp cơ bản để chuyển dữ liệu giữa máy tính trung tâm (CPU hay bộ nhớ) và các thiết bị ngoại vi:

1. Hỏi vòng (polling) hay còn gọi là I/O được điều khiển bằng chương trình (Program-controlled I/O)
2. I/O bằng ngắt (interrupt-initiated I/O), và
3. DMA (Direct Memory Access=Truy cập bộ nhớ trực tiếp).

Chú ý: Người ta cũng có thể kết hợp các phương pháp trên.

Polling hoặc I/O điều khiển bằng lập trình

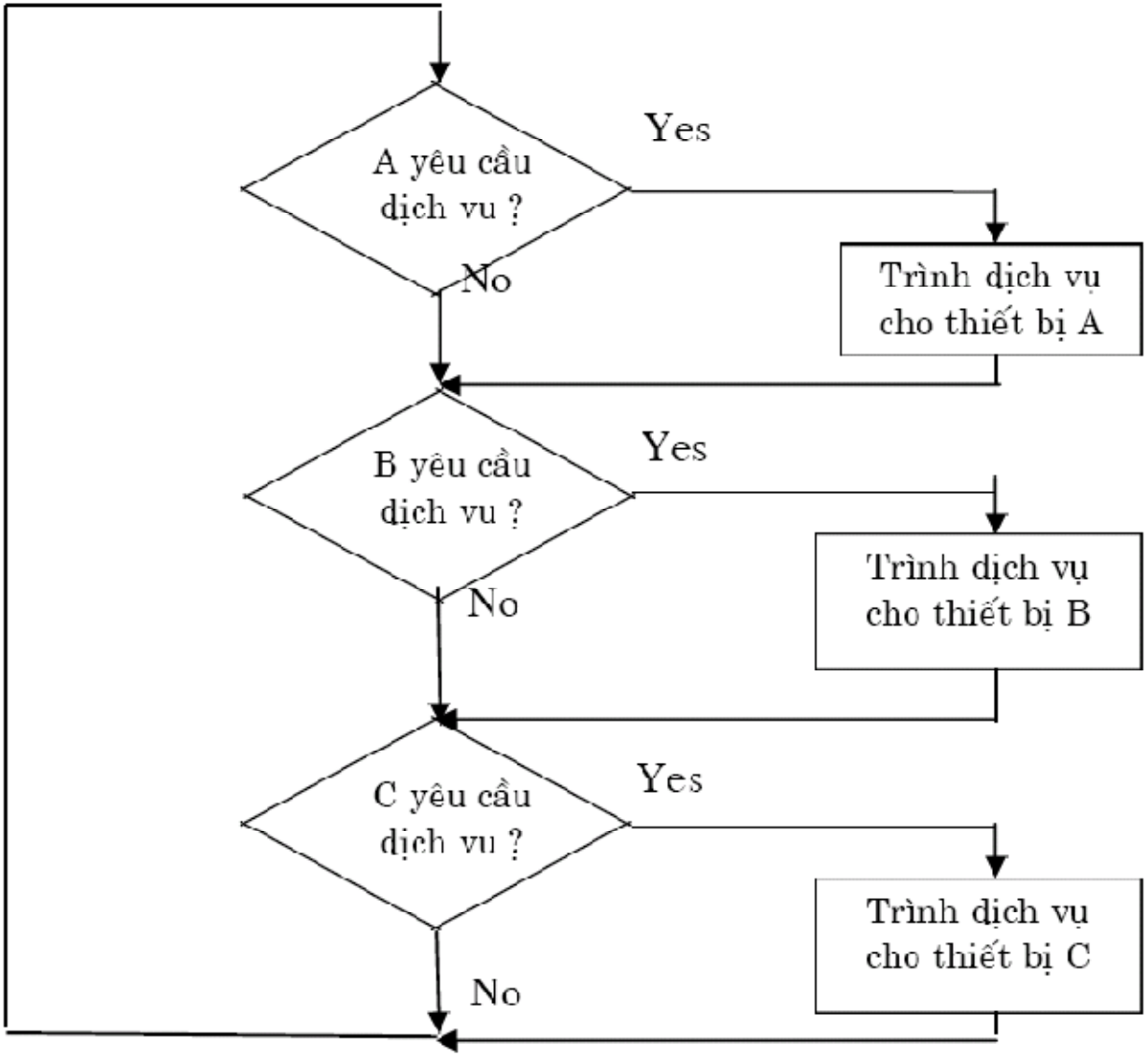


(a) Polling (Hỏi vòng)

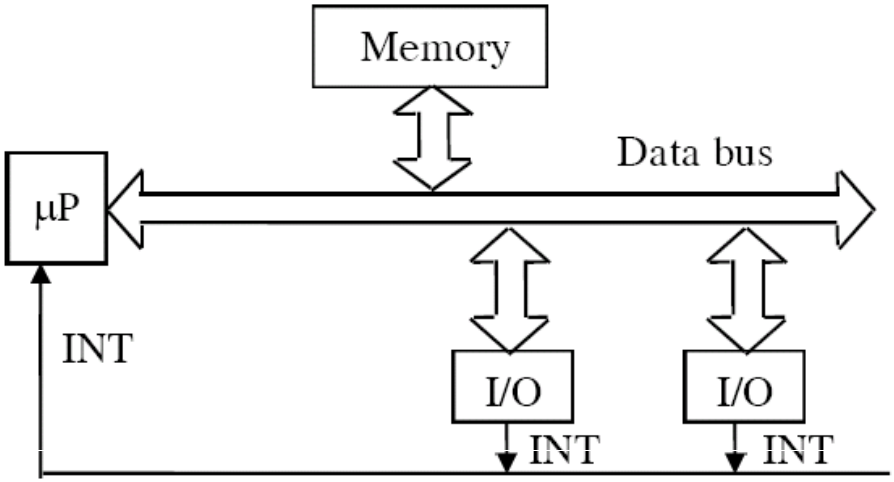
Kỹ thuật polling có 2 hạn chế:

1. Mất thời gian của MPU (do kiểm tra trạng thái của tất cả các ngoại vi thường xuyên).
2. Chậm, do đó làm trở ngại trong hệ thống thời gian thực, không thỏa mãn cho các thiết bị nhanh (thí dụ: disks hoặc CRT).

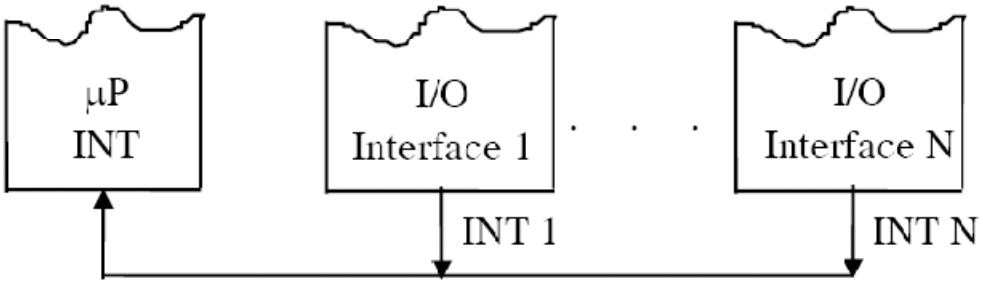
Lưu đồ vòng lặp polling



I/O bằng ngắt (Interrupt)

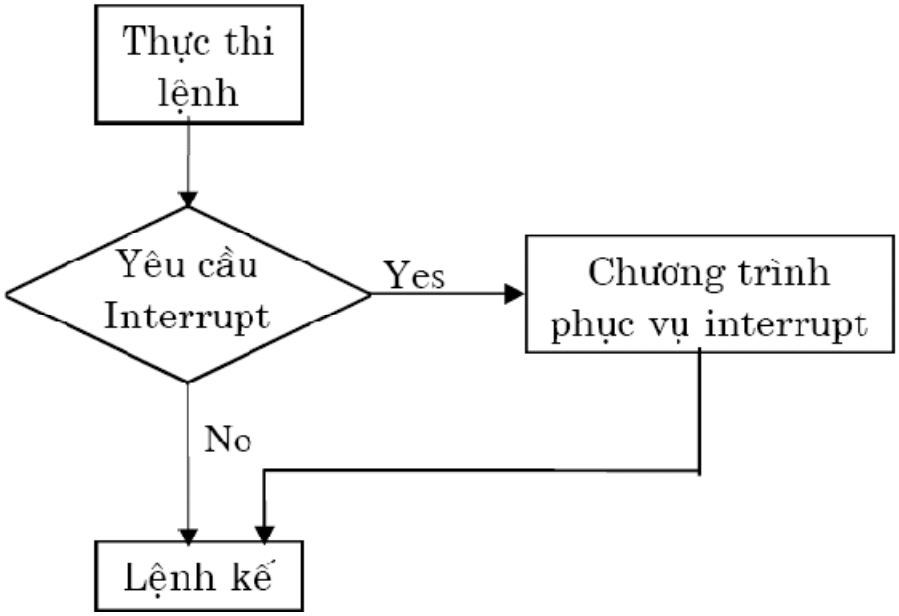


(b) I/O bằng ngắt



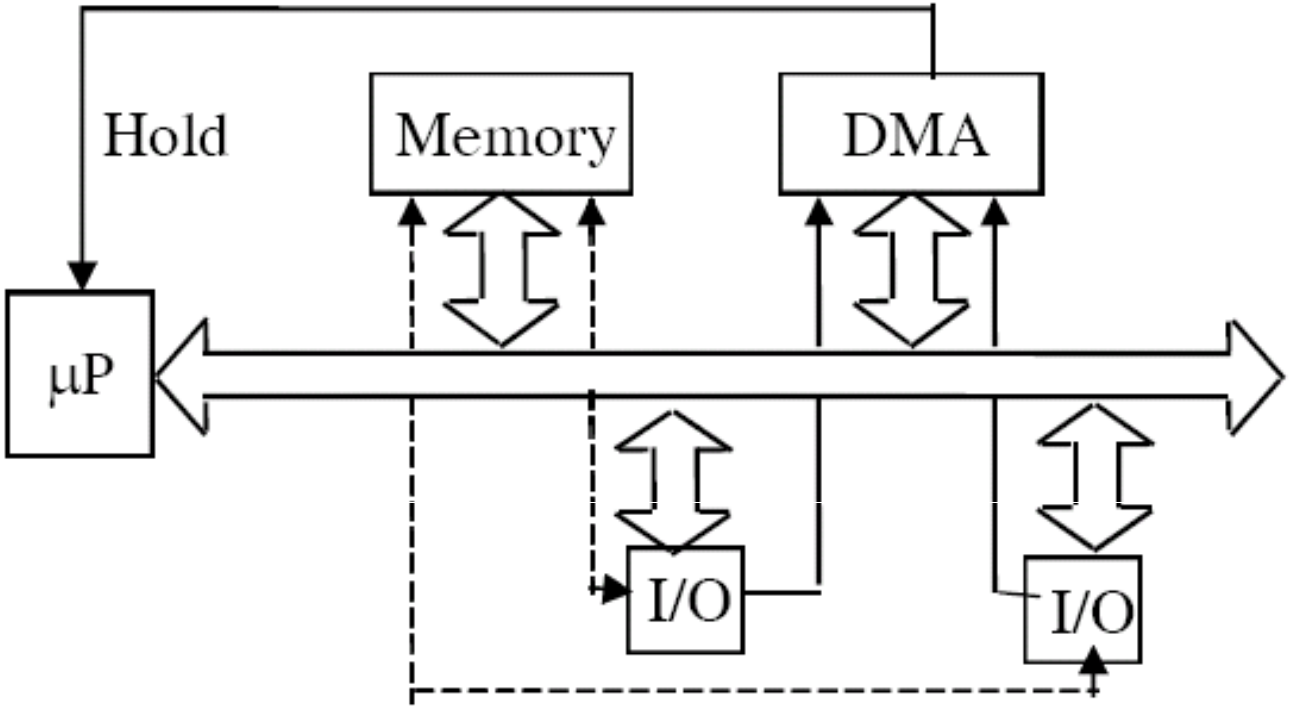
Nhiều thiết bị có thể yêu cầu dịch vụ đồng thời

Hình 1.30 Dãy Interrupt



Hình 1.31 Logic ngắt.

DMA (Direct Memory Access)



(c) I/O bằng DMA

Phần cứng DMAC (Direct Memory Access Controller= Bộ điều khiển truy cập bộ nhớ trực tiếp) được thiết kế để thực hiện chuyển dữ liệu tốc độ cao giữa bộ nhớ và thiết bị.
Do đó, DMAC sẽ cần sử dụng cả hai bus dữ liệu và bus địa chỉ.

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

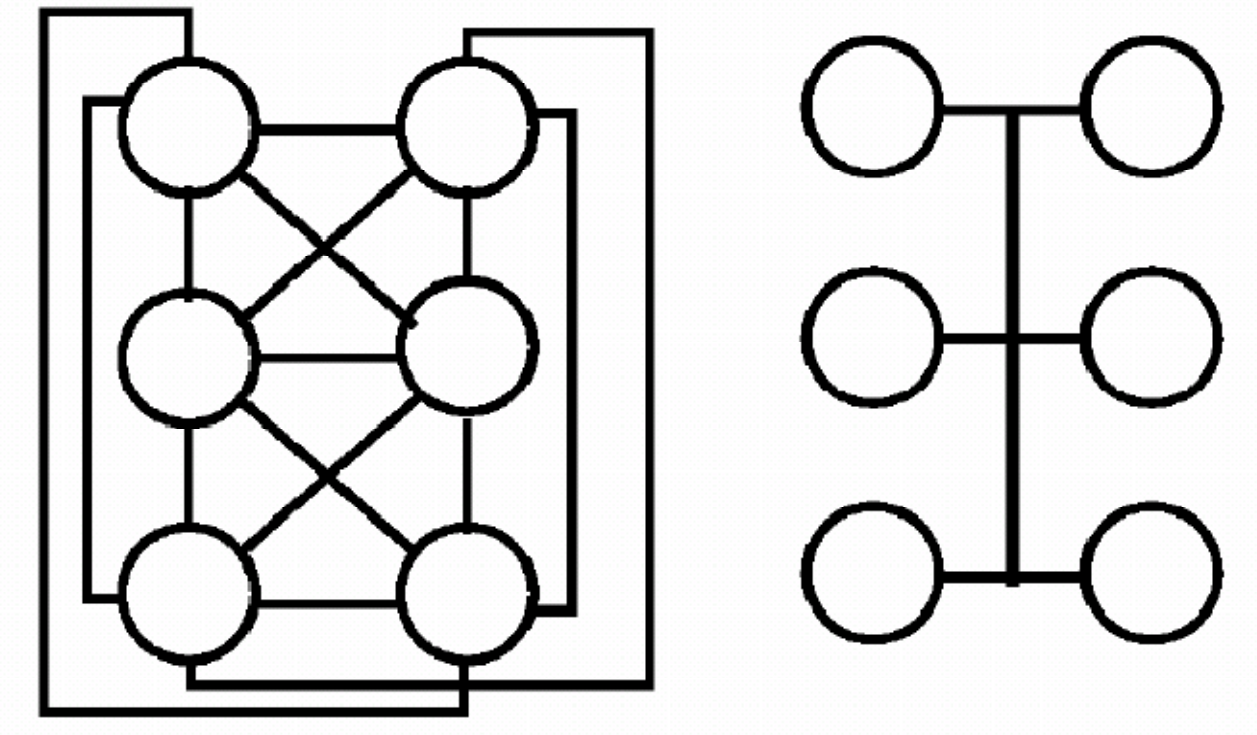
1.10 Vi điều khiển và vi xử lý

1.6 BUS HỆ THỐNG

Bus

- Các hệ thống con của MCU và CPU liên lạc với nhau qua “bus” (tuyến)
- Bus là tập hợp các đường tín hiệu mà qua đó có thể truyền đi thông tin về địa chỉ, dữ liệu và điều khiển

Bus

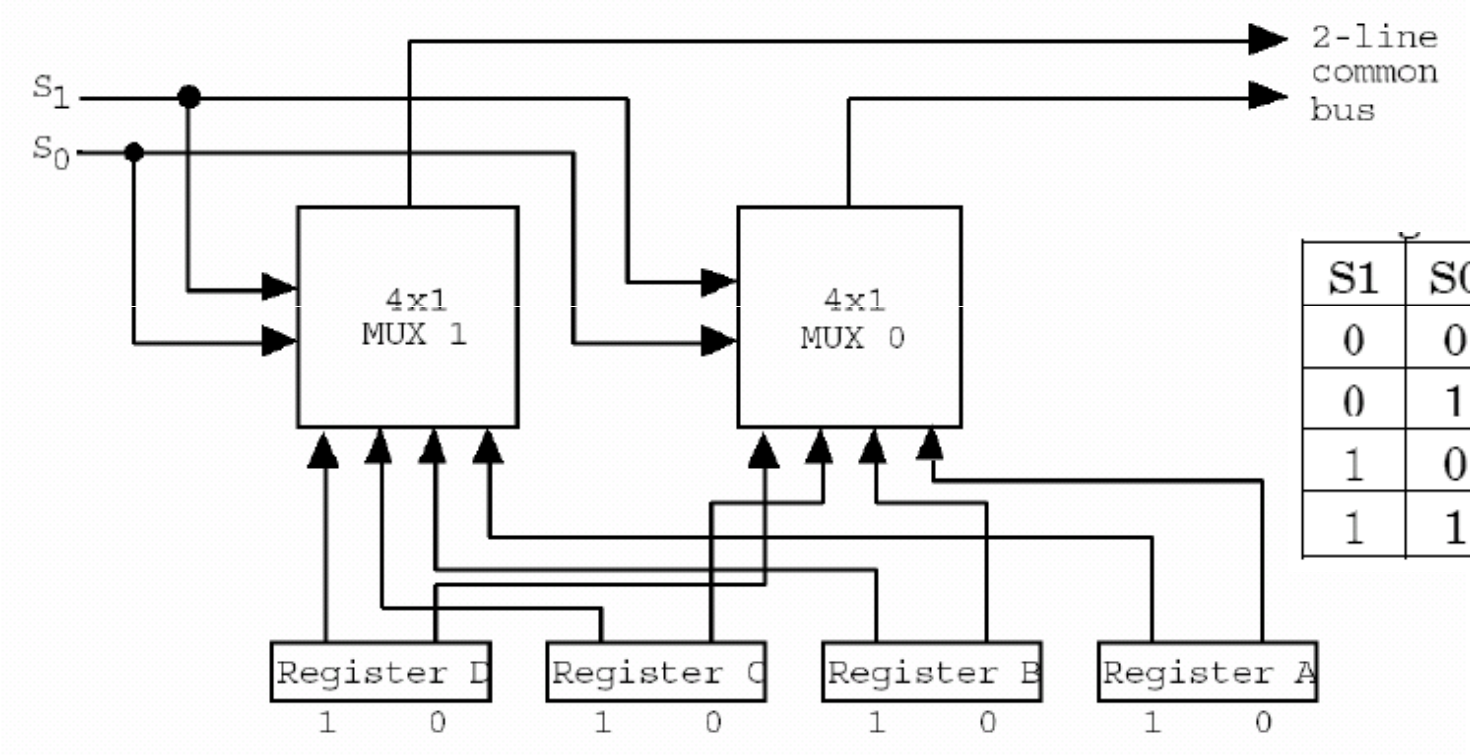


(a) Các kết nối dành riêng (b) Kiến trúc bus

Hình 1.32 Các kết nối nhiều điểm khi không có bus và có bus.

- Bus có thể **hai chiều** (bidirectional) hay **một chiều** (unidirectional)
- Nếu cả hai dữ liệu và địa chỉ được truyền trên cùng một bus thì người ta gọi bus có **dồn kênh** (multiplexed).
- Nếu hai thiết bị gửi thông tin đồng thời trên cùng bus thì xảy ra **tranh chấp bus** (bus contention) và có thiết bị hư.

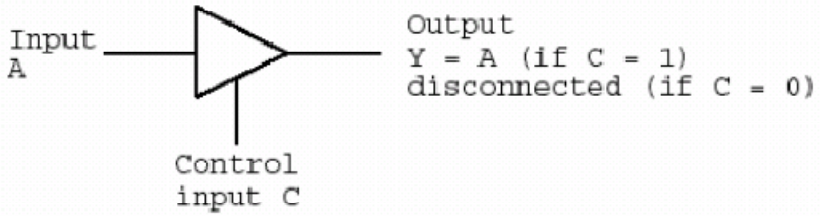
Cài đặt bus chung với *Bus dồn kênh (Multiplexer Bus)*



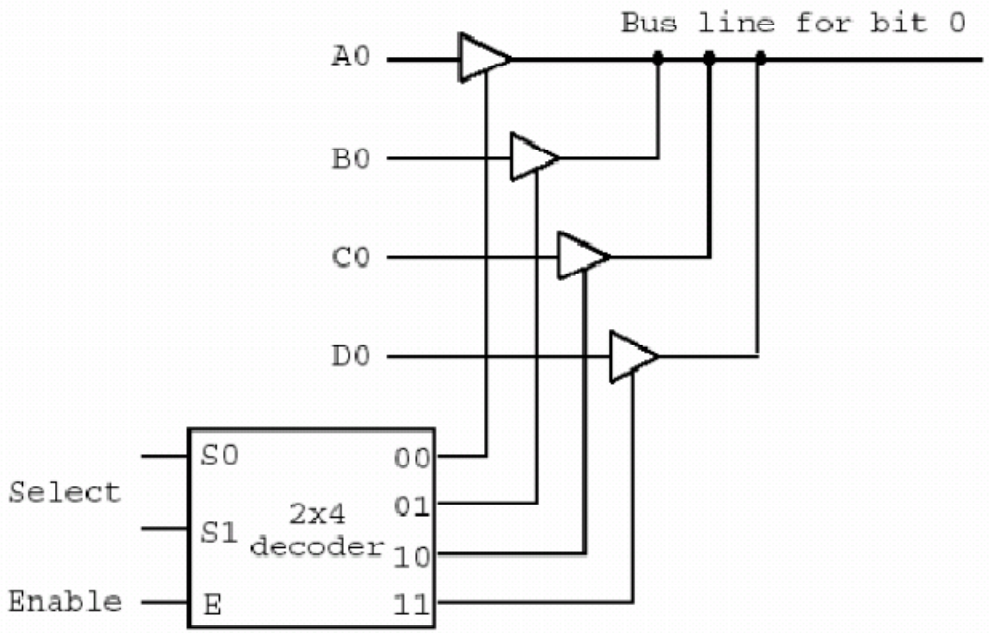
S1	S0	Chọn thanh ghi
0	0	A
0	1	B
1	0	C
1	1	D

Hình 1.33 Thí dụ Bus dồnkênh có 2 đường.

Cài đặt bus chung với
Các bộ đệm bus 3 trạng thái



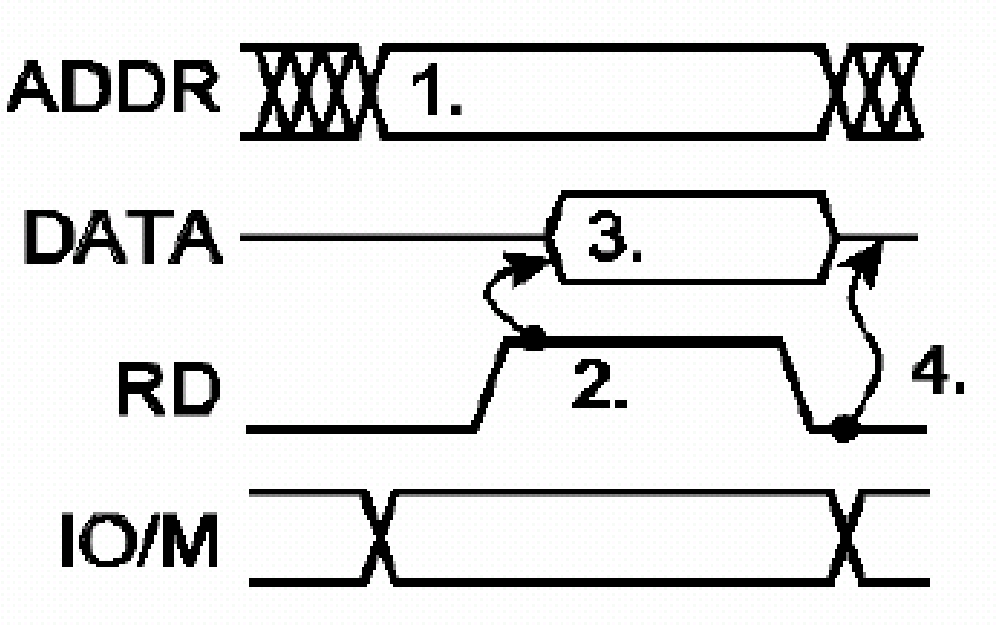
Three-state buffer.



Three-state buffer implementation of bus line.

Hình 1.34 Thí dụ bus chung với bộ đệm 3 trạng thái.

Chu kỳ đọc bus (Read Bus Cycle)



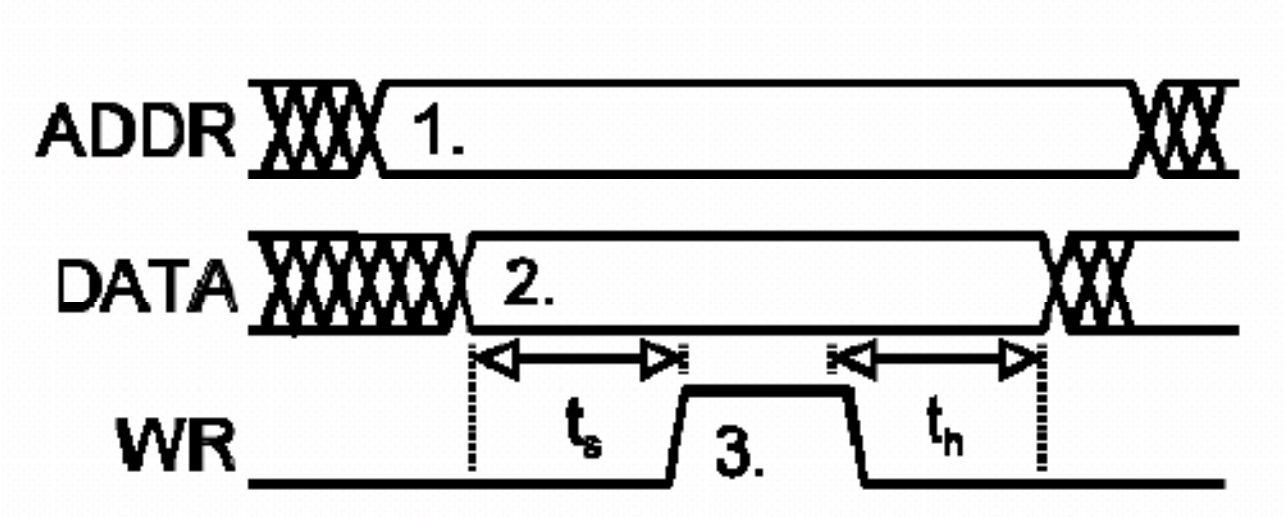
Hình 1.35 Thí dụ chu kỳ đọc bus.

Các chu kỳ đọc bus có thể chia làm

Memory Read (Đọc bộ nhớ)	MR	Đọc từ bộ nhớ
Opcode Fetch (Nhận mã lệnh)	OF	Đọc word thứ nhất của lệnh
IO Read (Đọc IO)	IOR	Đọc từ cổng nhập

Chu kỳ ghi bus (Write Bus Cycle)

Giản đồ định thì (bus hệ thống) cho chu kỳ đọc bus như sau



Hình 1.36 Thí dụ chu kỳ ghi bus.

1. Địa chỉ hợp lệ được cung cấp bởi CPU.
2. Dữ liệu hợp lệ trên bus dữ liệu được cung cấp bởi CPU.
3. CPU kích hoạt WR, rồi sau đó bỏ kích hoạt WR.

Các chu kỳ ghi bus có thể chia làm

Memory Write (Ghi bộ nhớ)	MW	Ghi vào bộ nhớ
IO Write (Ghi IO)	IOW	Ghi ra cổng xuất

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

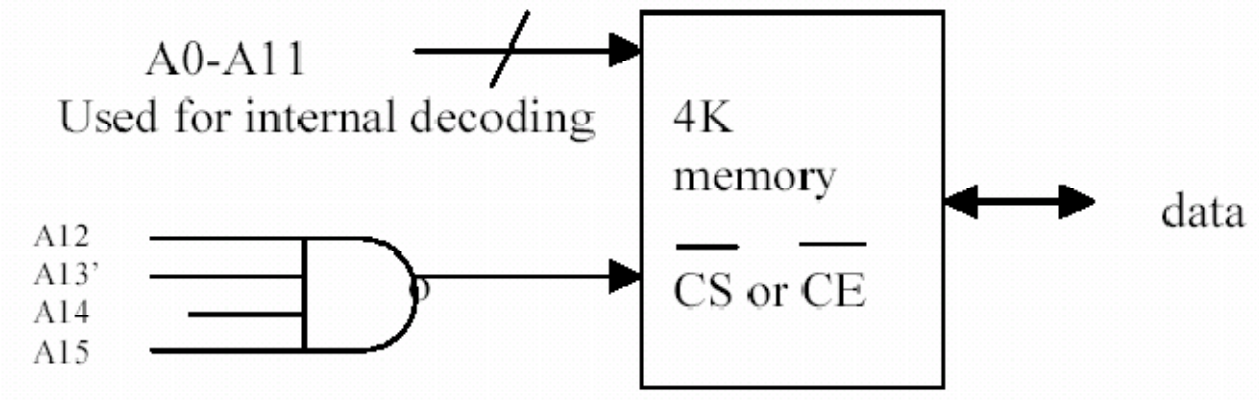
1.10 Vi điều khiển và vi xử lý

1.7 GIẢI MÃ ĐỊA CHỈ

Giải mã địa chỉ (Addressing Decoding)

- CPU cho phép một chip bộ nhớ (hoặc thiết bị I/O) chỉ khi nó muốn liên lạc với nó. Để thực hiện việc này CPU sử dụng mạch giải mã địa chỉ.
- Các phương pháp giải mã địa chỉ:
  1. ***Giải mã đầy đủ hay toàn phần (Full decoding):*** Mỗi ngoại vi được gán với một địa chỉ duy nhất. Tất cả các bit địa chỉ được dùng để định nghĩa vị trí tham chiếu.
  2. ***Giải mã một phần (Partial decoding)***: Không phải tất cả các bit được xử dụng trong quá trình giải mã. Các ngoại vi có thể đáp ứng với hơn 1 địa chỉ. Phương pháp này làm giảm độ phức tạp trong mạch giải mã địa chỉ.

Thí dụ: Ta có chip bộ nhớ 4K, CPU có 16 đường địa chỉ, muốn sử dụng 12 đường địa chỉ để giải mã nội và 4 đường để chọn chip. Hãy thiết kế mạch giải mã địa chỉ để khi CPU tham chiếu vùng nhớ D000H đến DFFFH thì sẽ truy cập bộ nhớ 4K này.



Hình 1.37 Dùng NAND để giải mã địa chỉ.

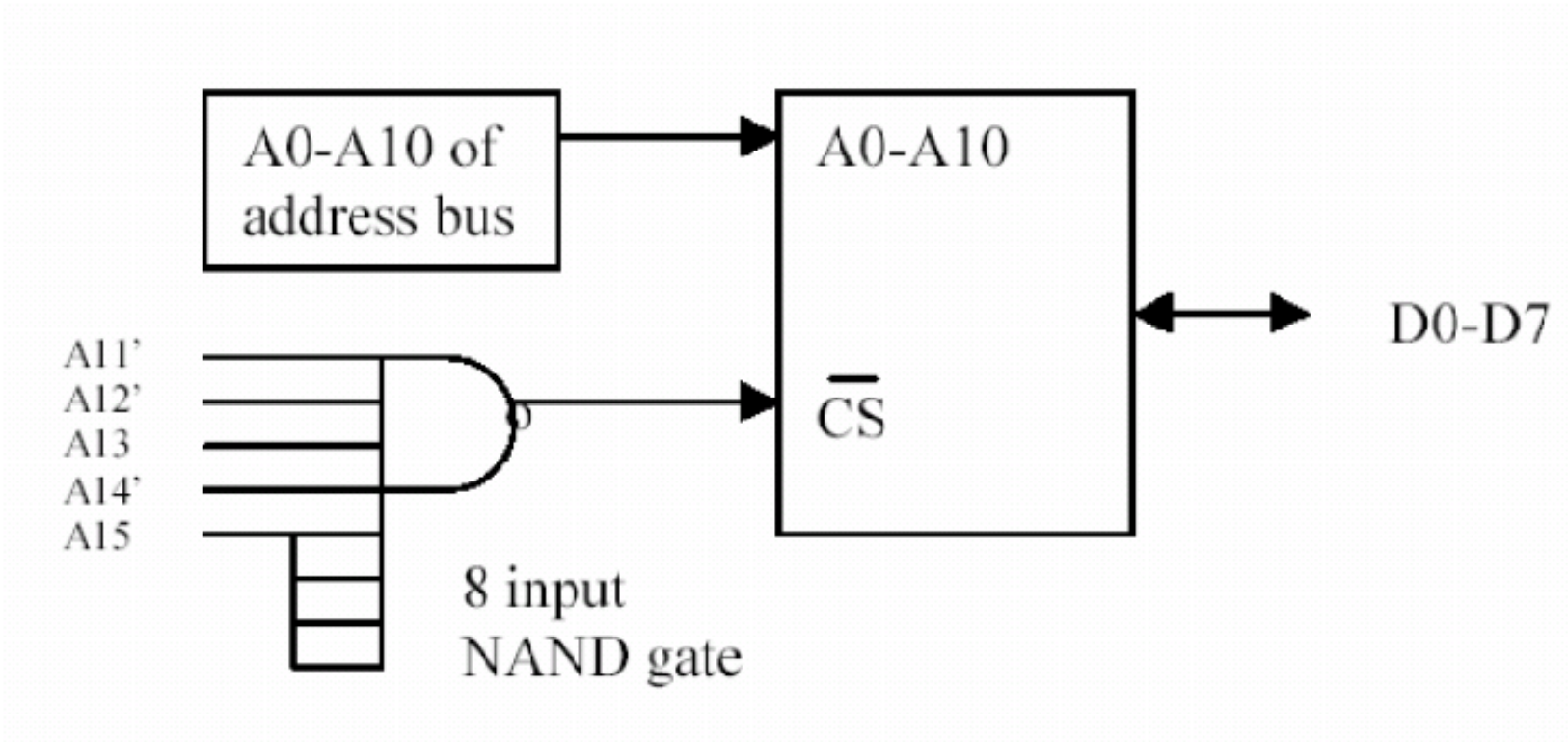
Vùng địa chỉ cần quản lý với chip nhớ 4K:

A15	A14	A13	A12	A11	A10	A9	. . .	A1	A0
1	1	0	1	0	0	0		0	0
...	...	...	...	...	...	...		...	...
1	1	0	1	1	1	1	. . .	1	1

Vì 4 bit đầu địa chỉ cần chọn là DH nên để có thể chọn chip được thì cần có mạch tạo tín hiệu chọn chip $\overline{CS}$ tích cực khi A15=A14=A12=1 và A13=0 hay $\overline{CS}$ =(A15.A14.A13'.A12).

Thí dụ: Giả sử là người ta dùng EPROM 2716 (2K x 8 bit) cho vùng nhớ từ A000H đến A7FFH Thiết kế mạch giải mã dùng các cổng NAND.

A15	A14	A13	A12	A11	A10	A9	. . .	A1	A0
1	0	1	0	0	0	0		0	0
...	...	...	...	...	...	...		...	...
1	0	1	0	1	1	1	. . .	1	1



Hình 1.38 Cài đặt giải mã địa chỉ bằng NAND.

Thí dụ:

Thiết kế mạch ánh xạ 8 chip bộ nhớ EPROM 2764 khác nhau (mỗi EPROM được tổ chức như 8K x 8 bits) thành khối bộ nhớ 64K byte với địa chỉ vật lý trong dãi F0000H đến FFFFFH.

Bài giải.

Mỗi EPROM 2764 có thể được ánh xạ trực tiếp vào khối 8 KB như sau

Khối thứ 1: F0000 - F1FFFh

Khối thứ 2: F2000 - F3FFFH

Khối thứ 3: F4000 - F5FFFH

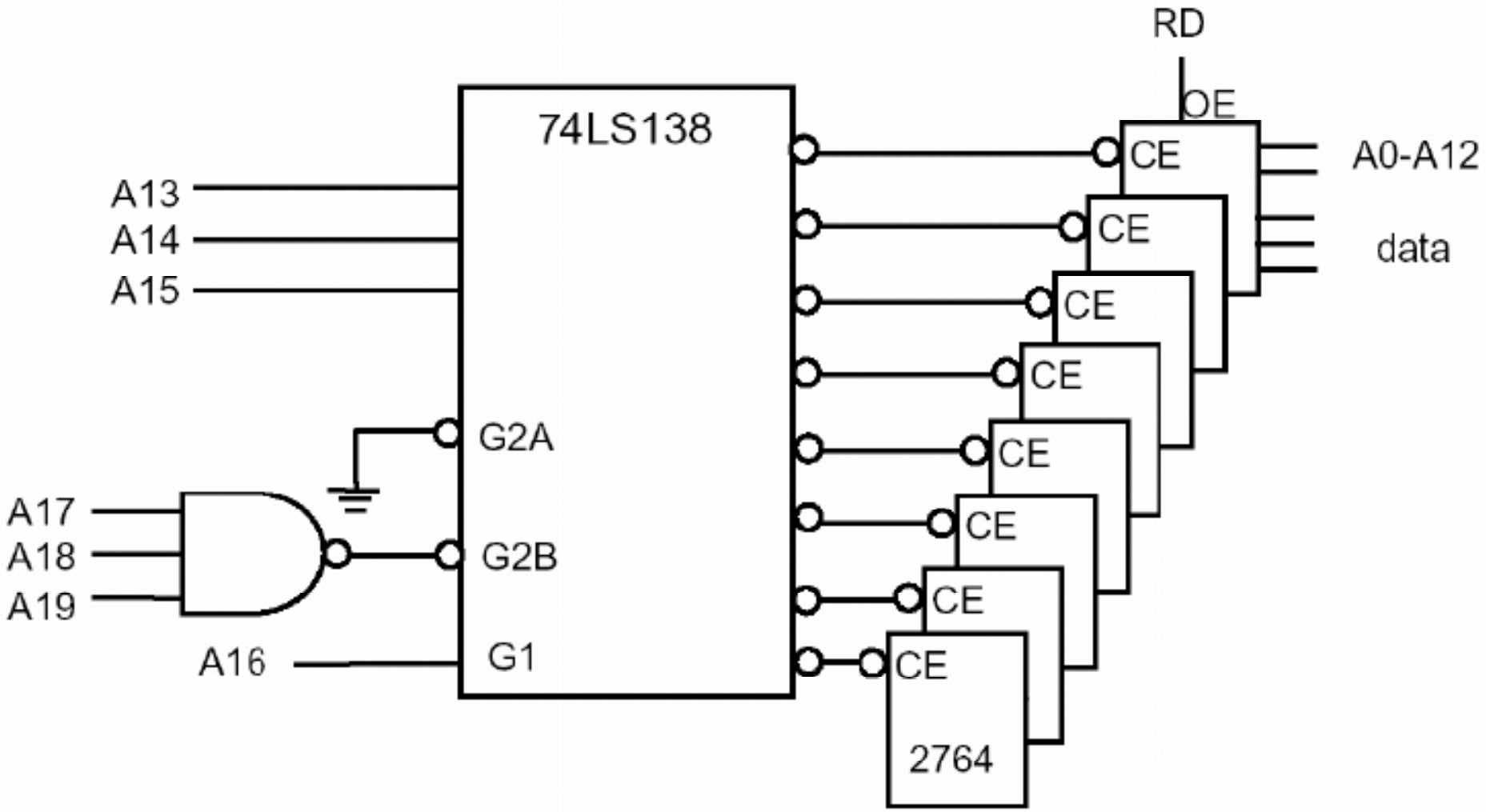
Khối thứ 4: F6000 - F7FFFH

Khối thứ 5: F8000 - F9FFFH

Khối thứ 6: FA000 - FBFFFH

Khối thứ 7: FC000 - FDFFFH

Khối thứ 8: FE000 - FFFFFH

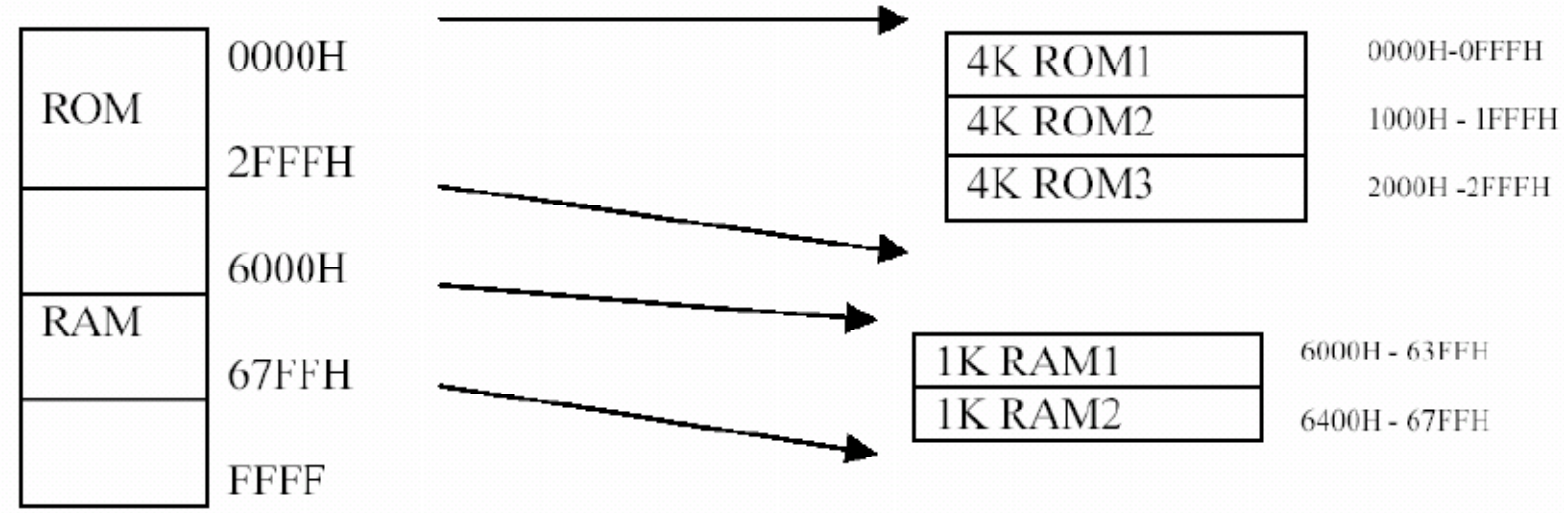


Hình 1.39 Giải mã địa chỉ F0000H –> FFFFF.

Bảng bộ nhớ (memory map) và bảng I/O

- **Bảng bộ nhớ**

  Bảng bộ nhớ minh họa những đoạn nào sẽ được sử dụng với RAM, ROM và một số trường hợp là các thiết bị I/O. Thí dụ:



- **Bảng I/O hay Vùng I/O**

  Được tổ chức và định địa chỉ như vùng nhớ. Mỗi địa chỉ tương ứng với một cổng I/O (thông thường rộng 8 bit).

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

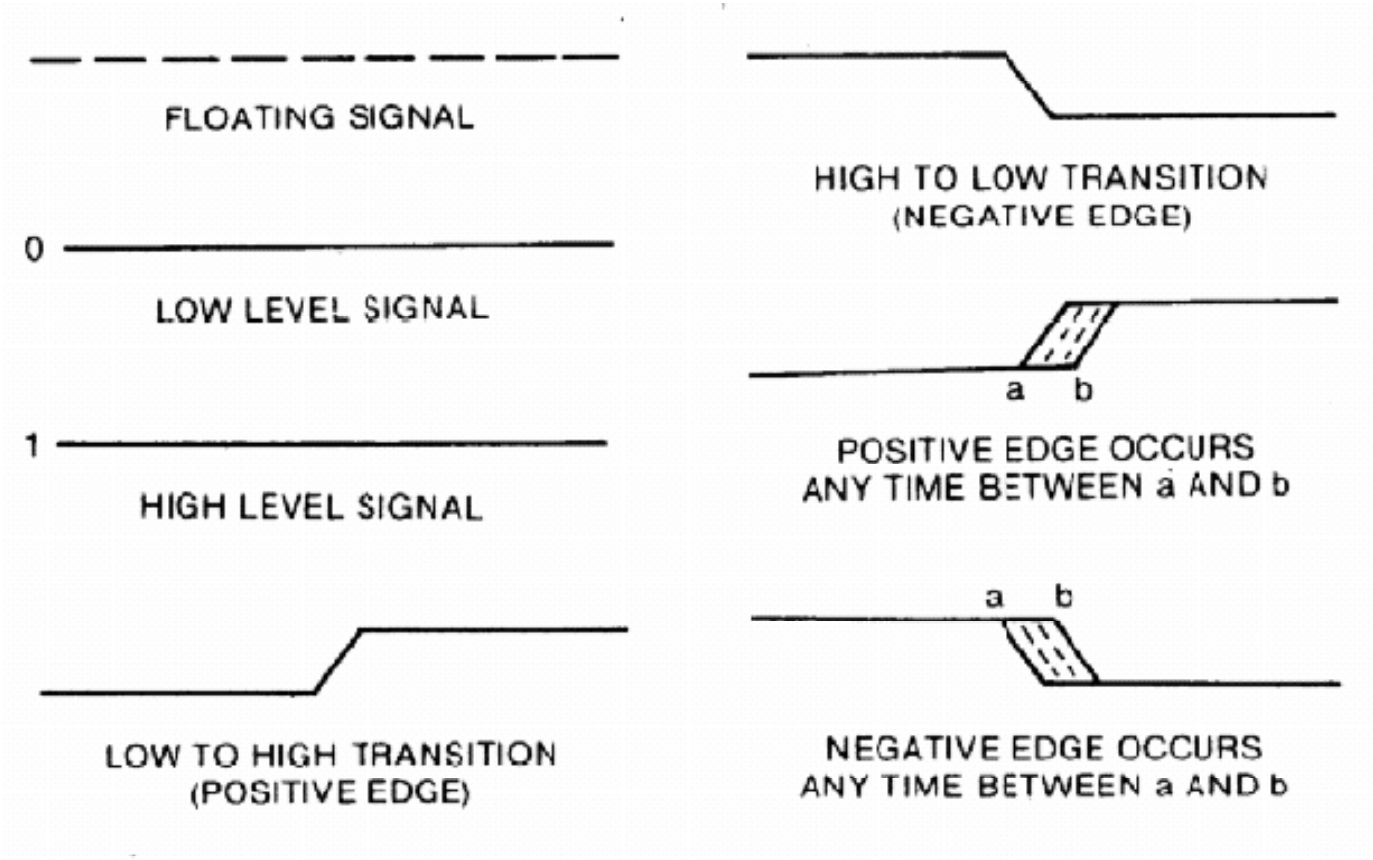
1.8 Định thì

1.9 Chương trình

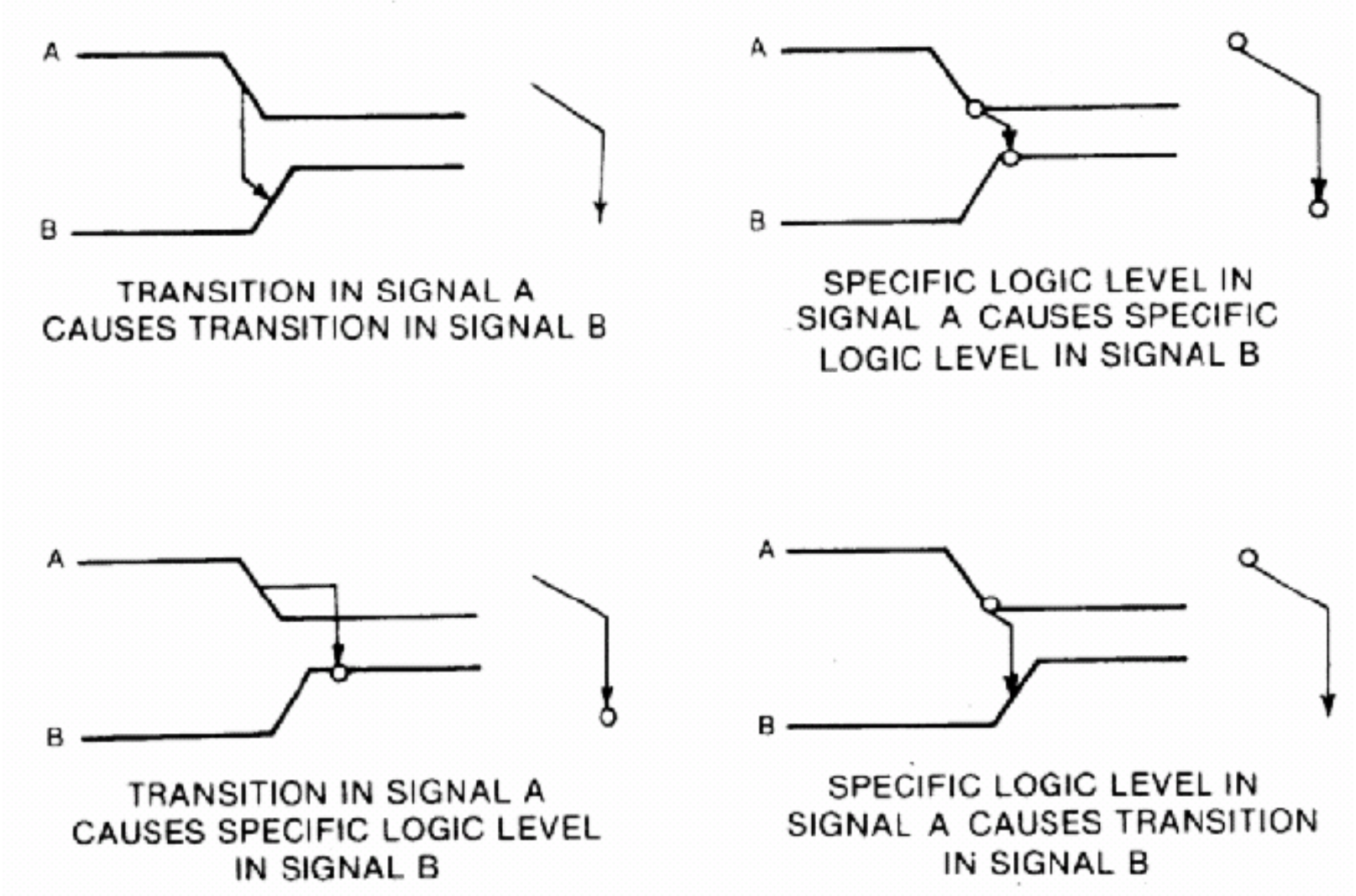
1.10 Vi điều khiển và vi xử lý

1.8 ĐỊNH THÌ (Timing)

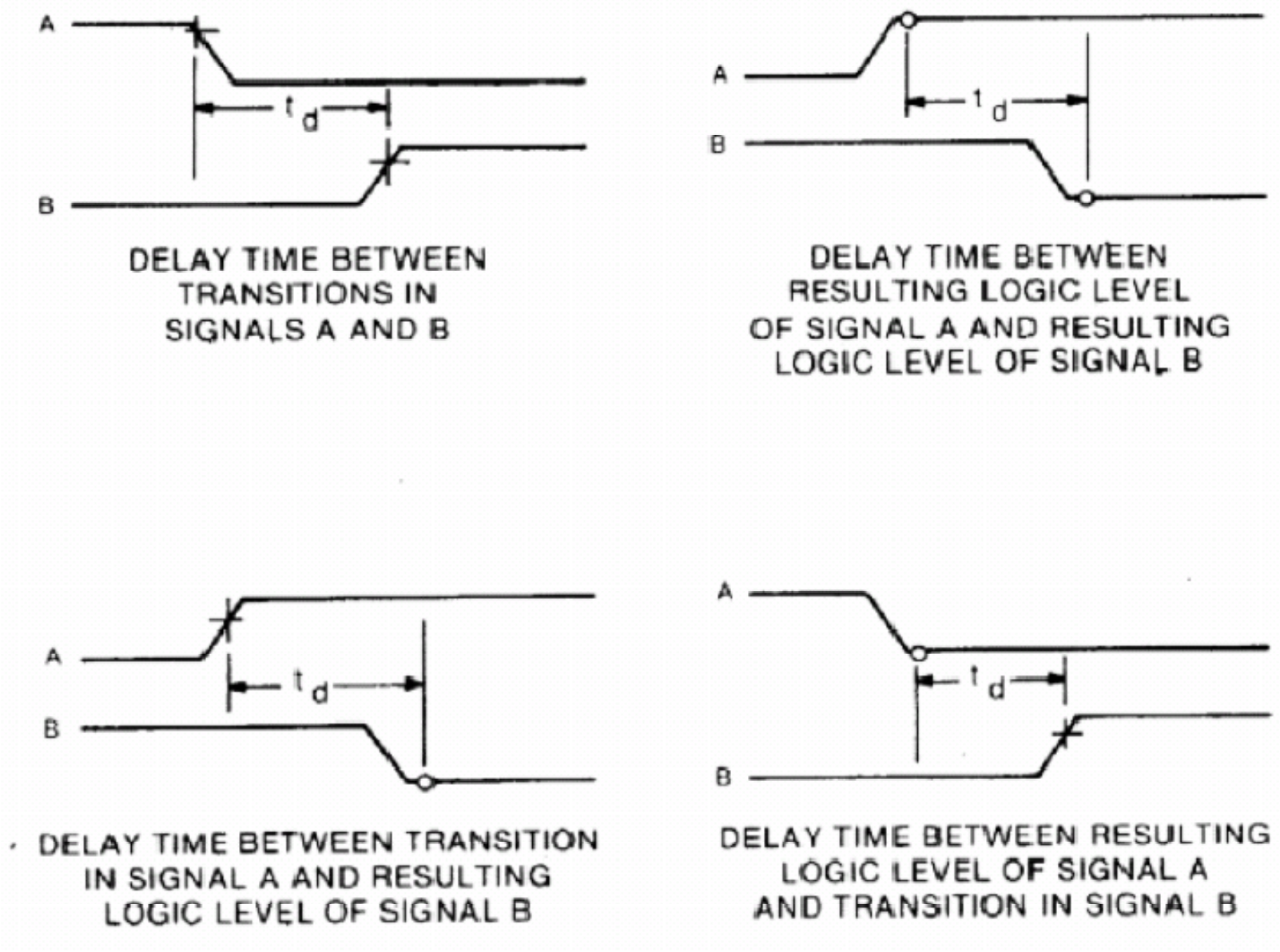
Các quy ước trong giản đồ định thì



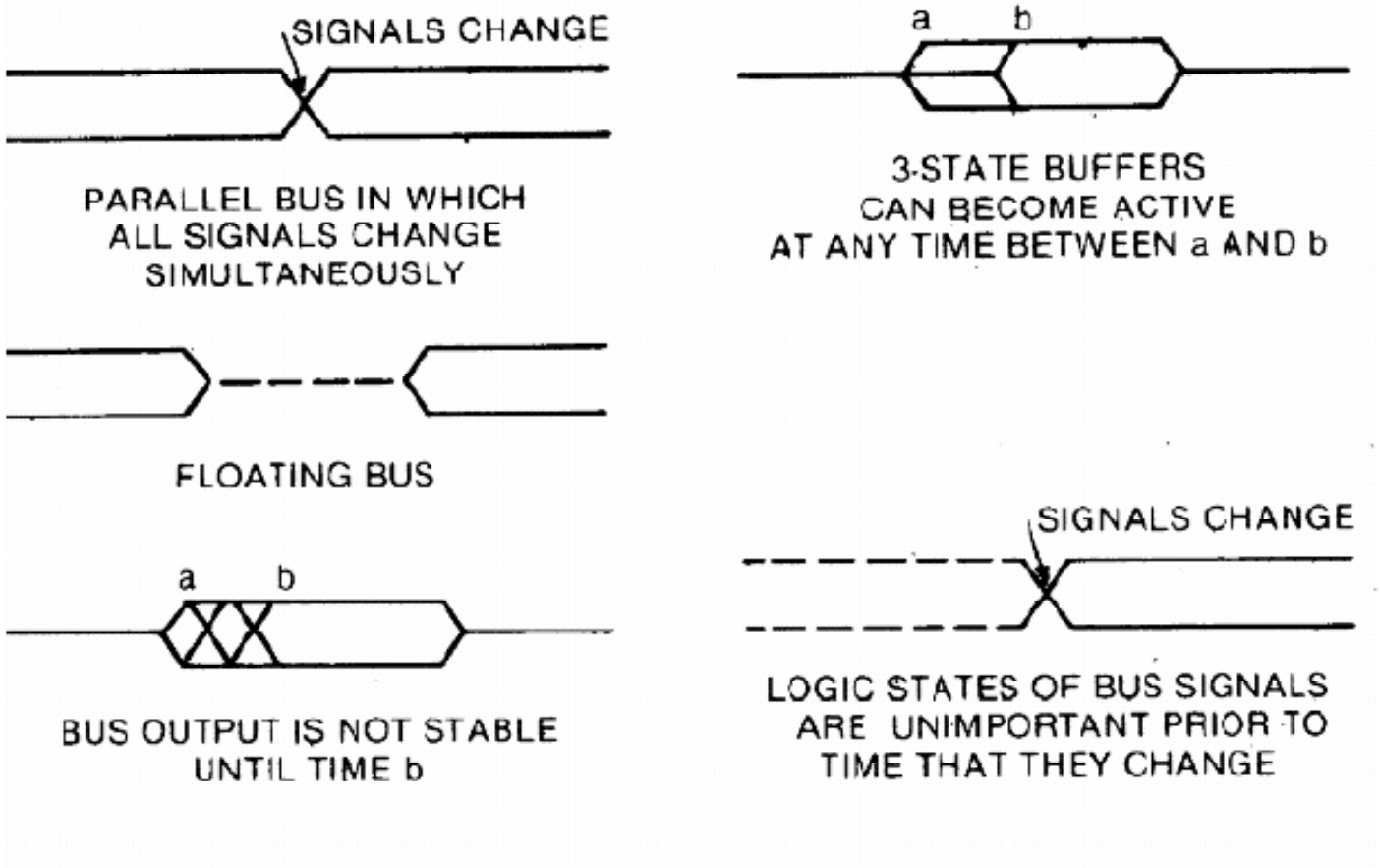
Hình 1.40 Các biểu diễn giản đồ định thì cho một tín hiệu Các đường ngang kết hợp với các đường chéo chỉ những chuyển tiếp trong các mức logic.



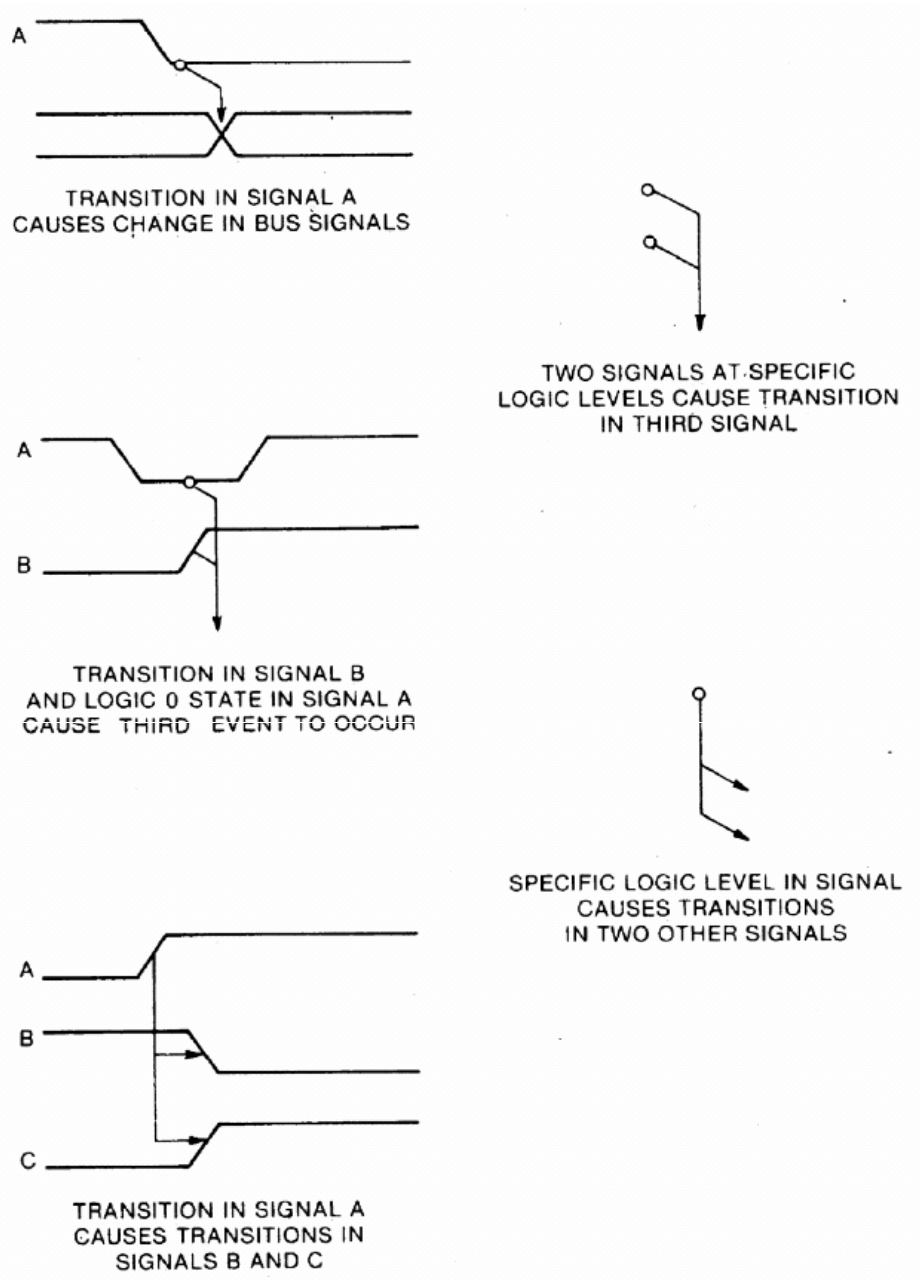
Hình 1.41 Các giản đồ định thì. Các quy ước cho cặp tín hiệu minh họa 4 cách mà một tín hiệu có thể cho phép hoặc kích một tín hiệu khác.



Hình 1.42 Thời gian trì hoãn. Các biểu diễn thời gian trì hoãn trong các giản đồ định thì cho cặp tín hiệu

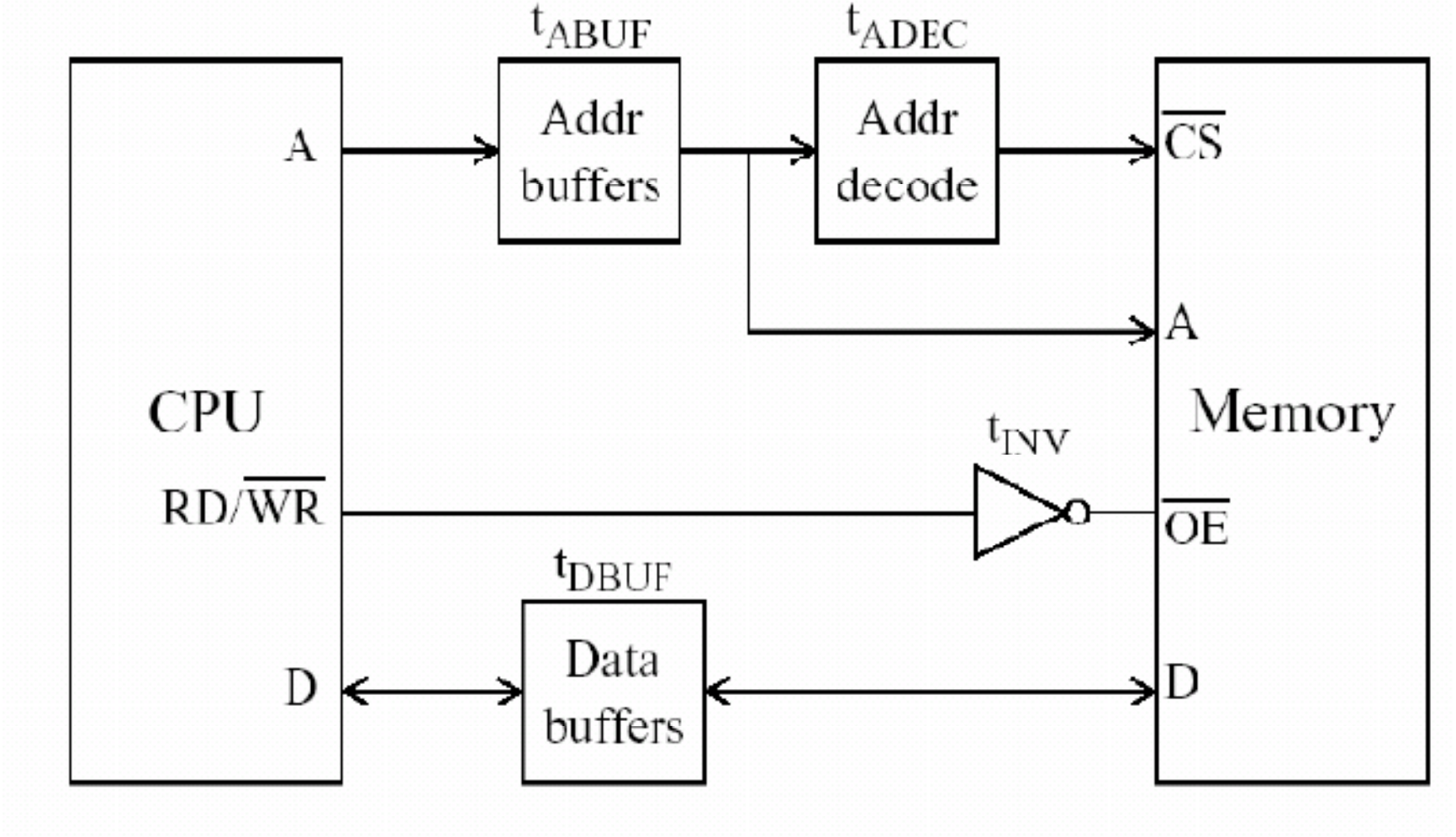


Hình 1.43 Các giản đồ định thì bus. Các biểu diễn minh họa 5 quy ước cho bus với 2 hay nhiều tín hiệu



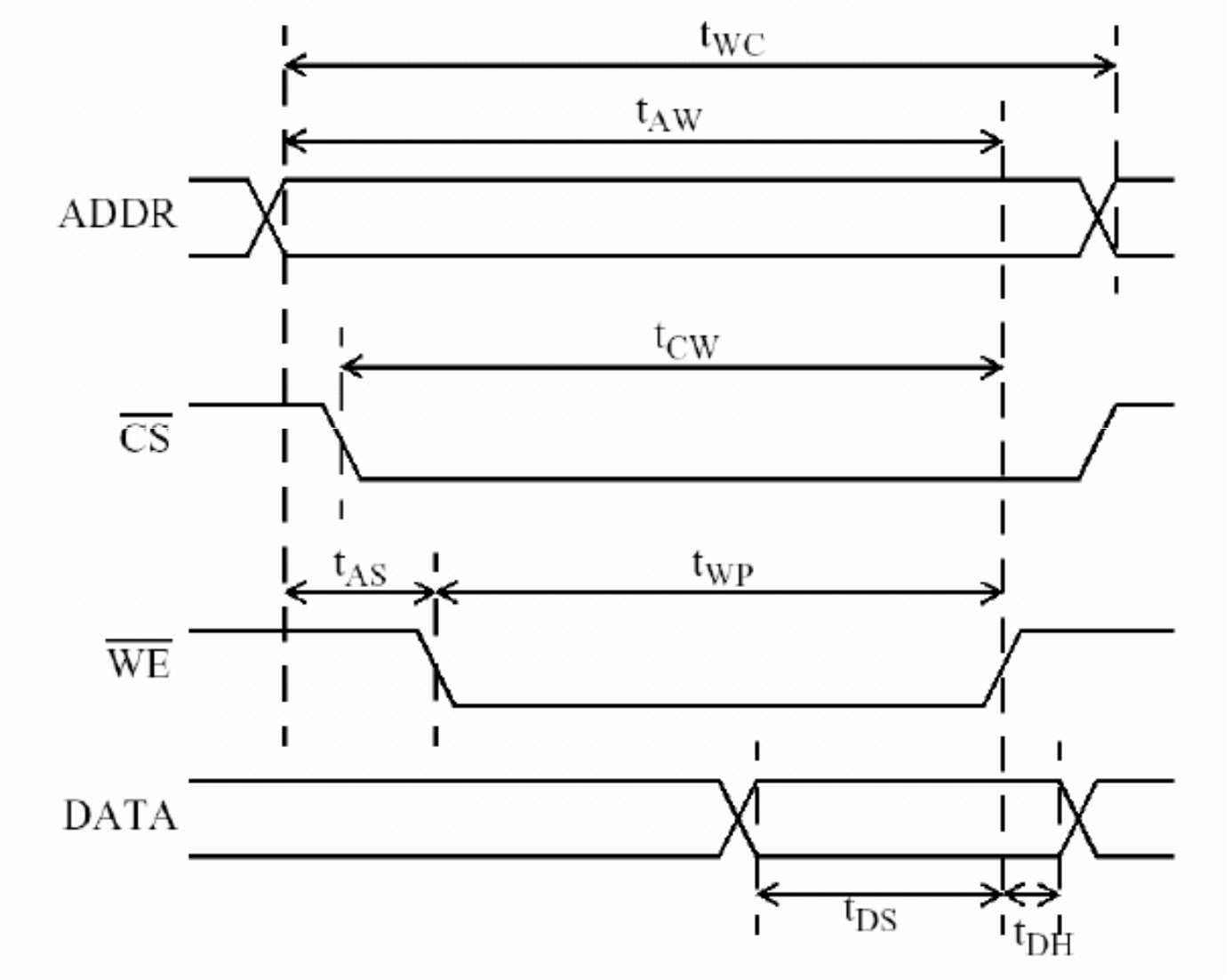
Hình 1.44 Các ứng dụng. Các kết hợp của các chuyển tiếp tín hiệu và các trạng thái logic làm thay đổi các tín hiệu khác.

Định thì đọc cấp hệ thống



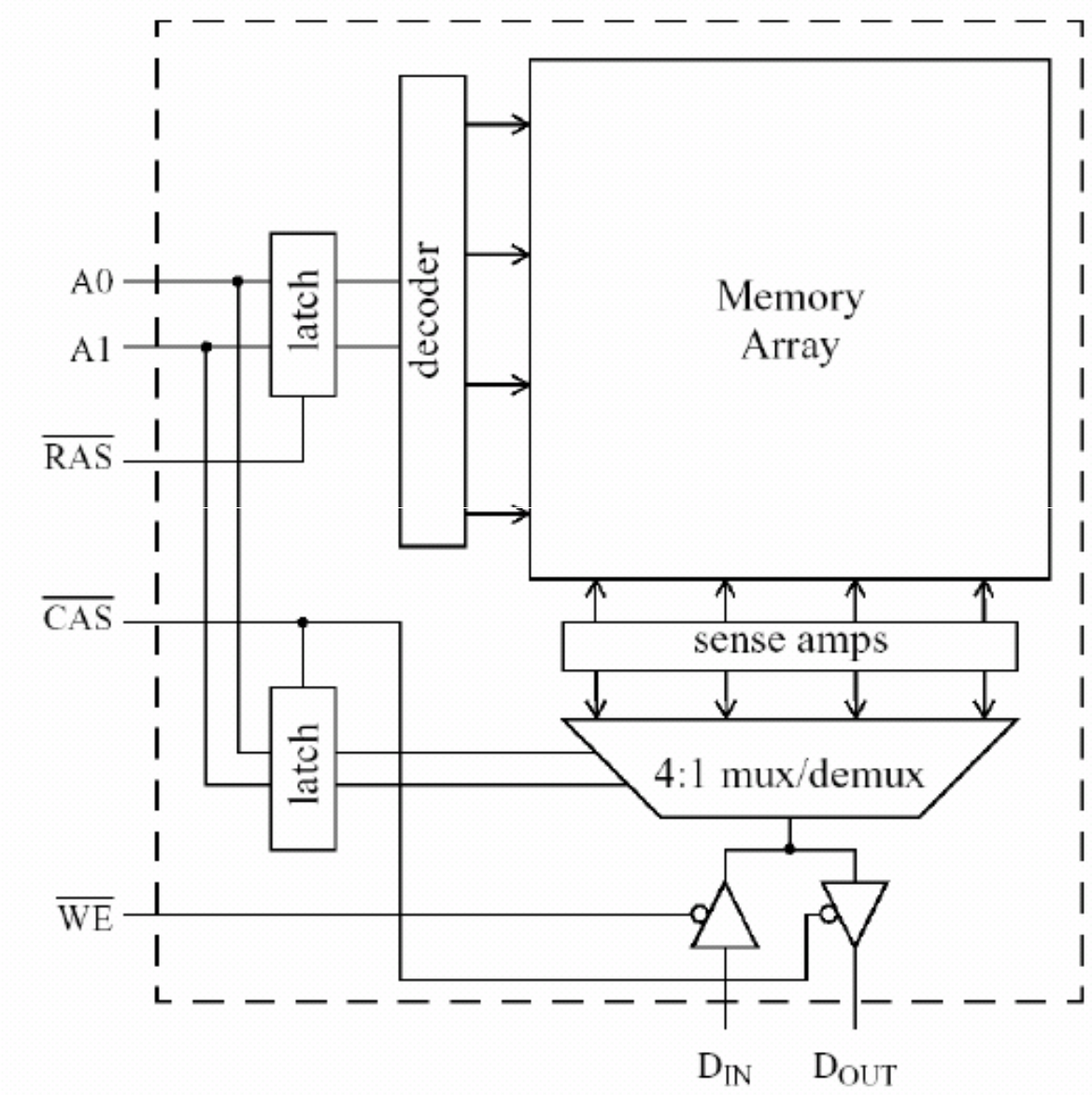
Hình 1.45 Các trì hoãn (trễ) trong giao tiếp giữa CPU và bộ nhớ.

Định thì ghi bộ nhớ



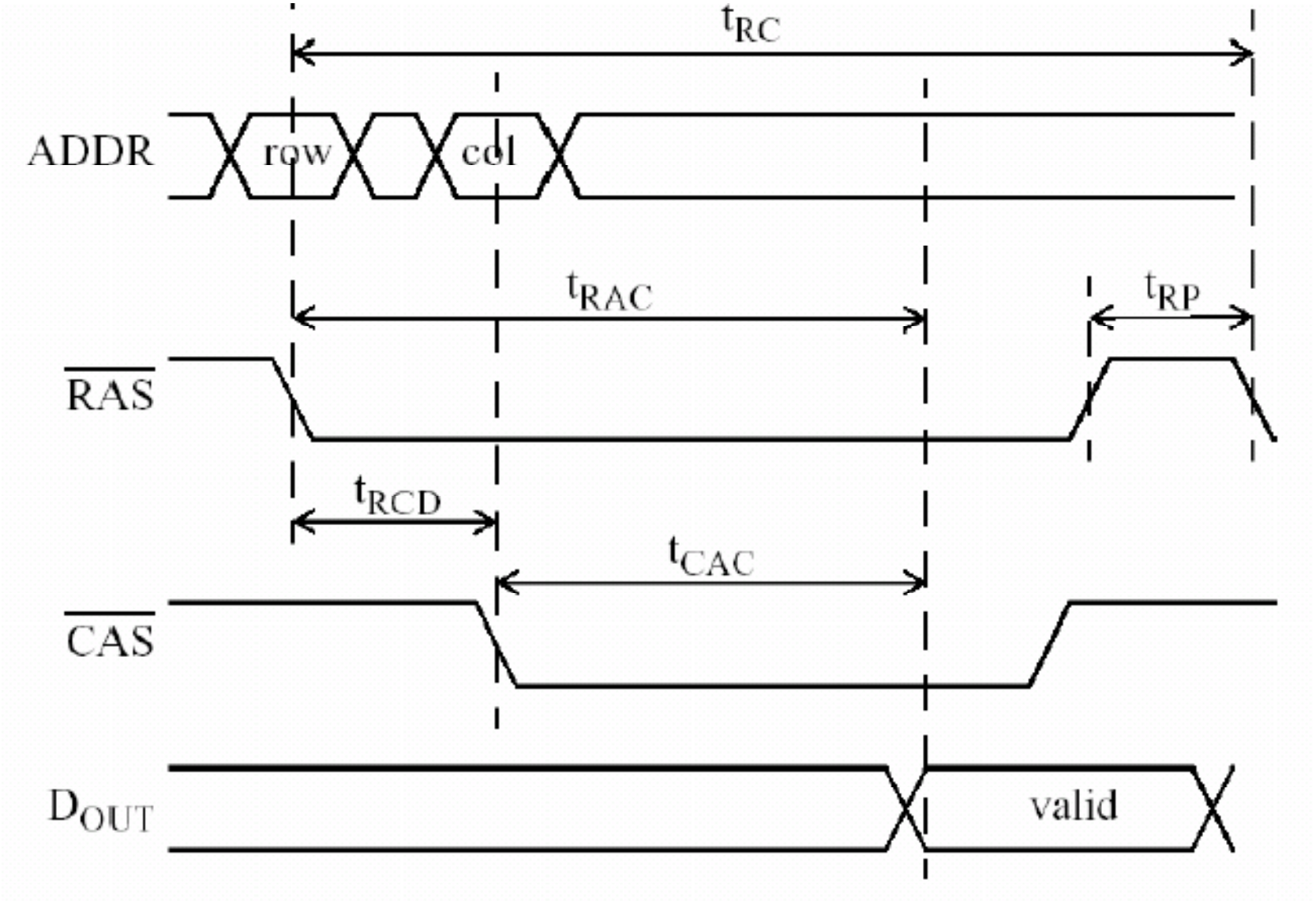
Hình 1.46 Định thì ghi bộ nhớ tiêu biểu.

Giao tiếp DRAM



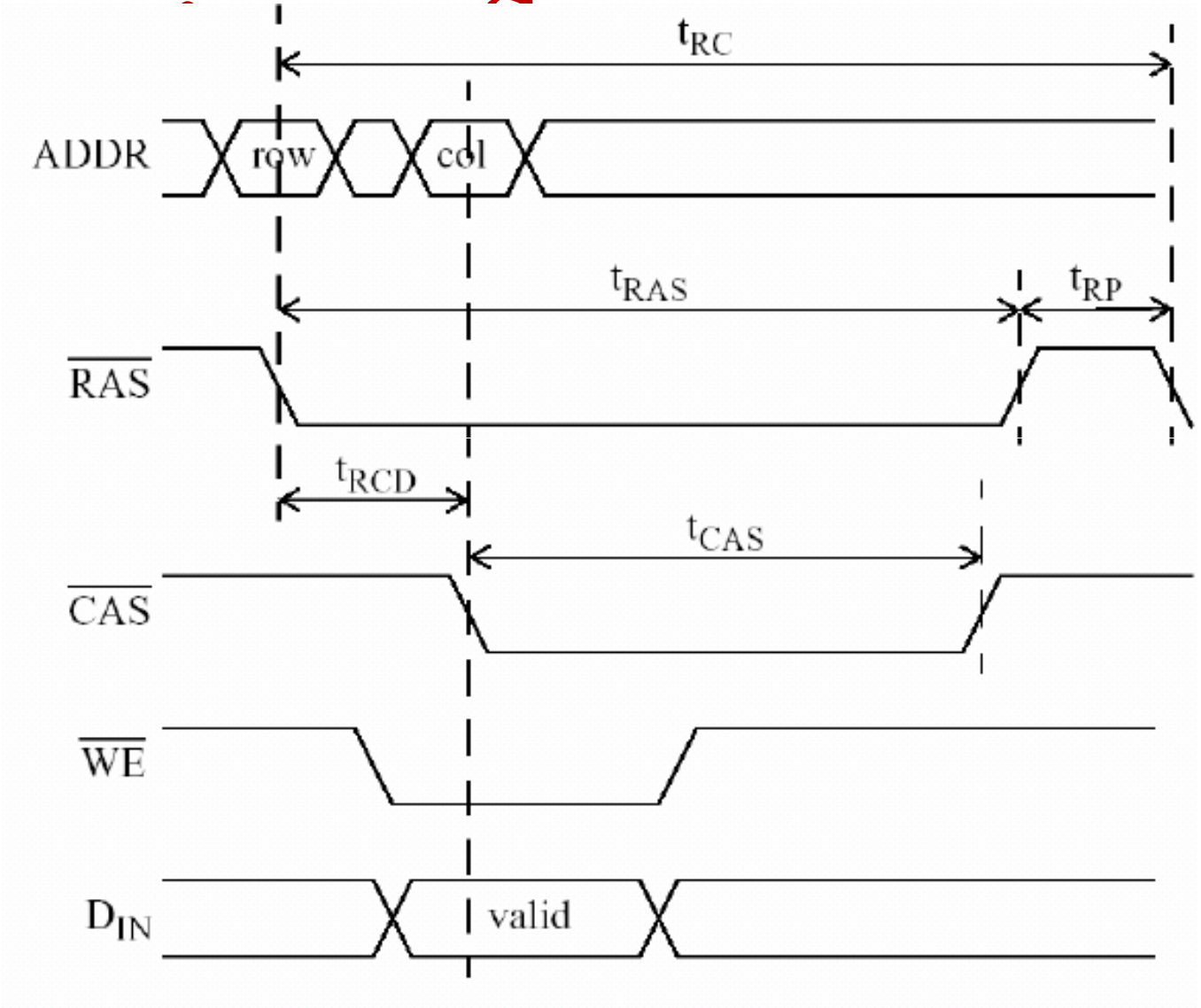
Hình 1.47 Sơ đồ khối của DRAM 4 x 1.

Định thì đọc với DRAM



Hình 1.48 Định thì đọc DRAM

Định thì ghi với DRAM



Hình 1.49 Định thì ghi DRAM

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

1.9 CHƯƠNG TRÌNH

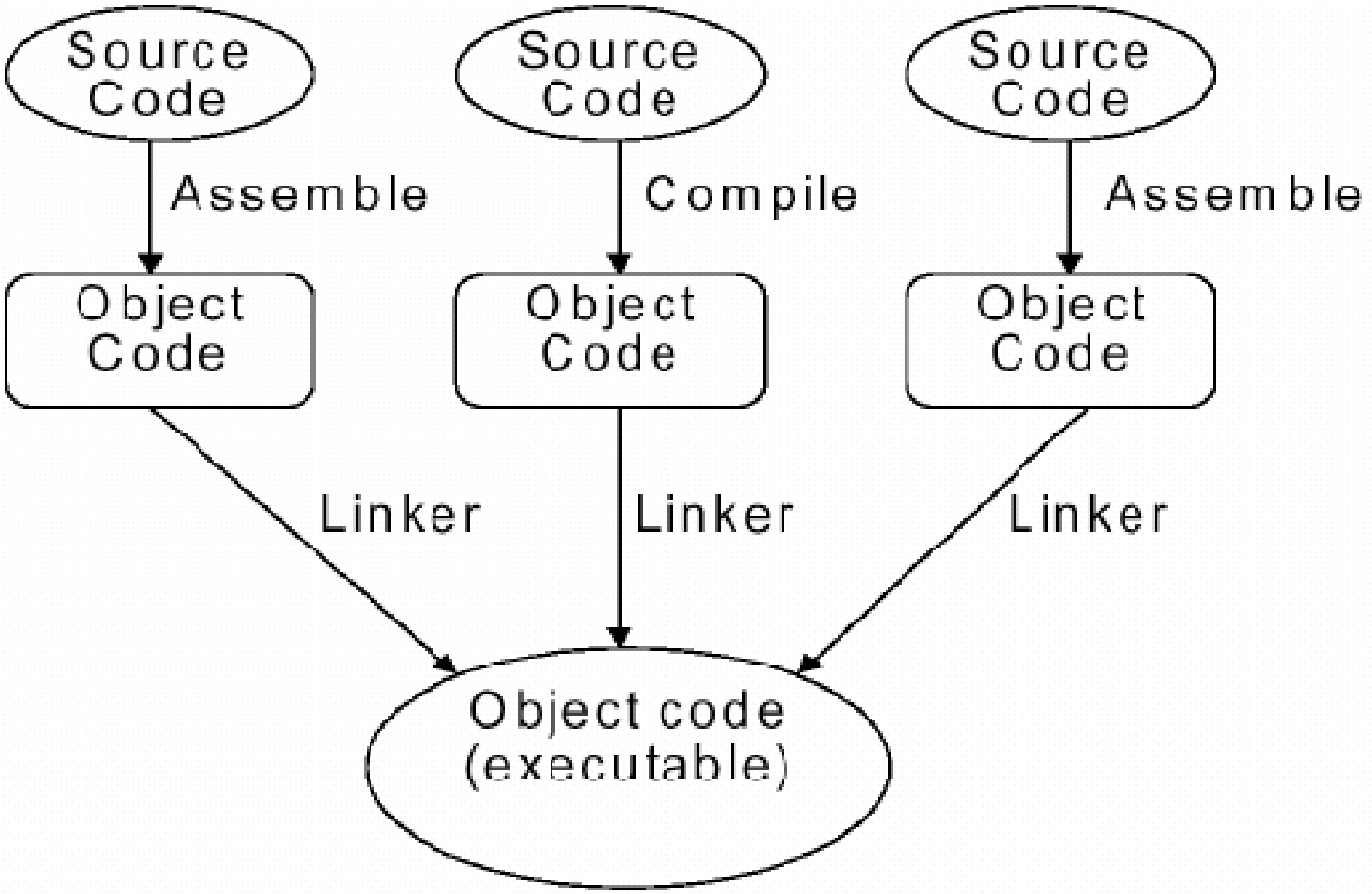
Chương trình

- Chương trình (program) là danh sách các lệnh (instruction=lệnh, chỉ thị) hay các phát biểu (statement) để điều khiển máy tính hay CPU thực hiện công việc xử lý dữ liệu mong muốn.
- Có nhiều loại ngôn ngữ lập trình:
  - Ngôn ngữ máy (machine language)
    - + Mã nhị phân
    - + Mã bát phân hay thập lục phân
  - Hợp ngữ (Assembly Language) (cần có Assembler [Trìnhdịch hợp ngữ]) → Mã ký hiệu
  - Ngôn ngữ cấp cao (cần có Compiler [Trình biên dịch])

Ngôn ngữ máy

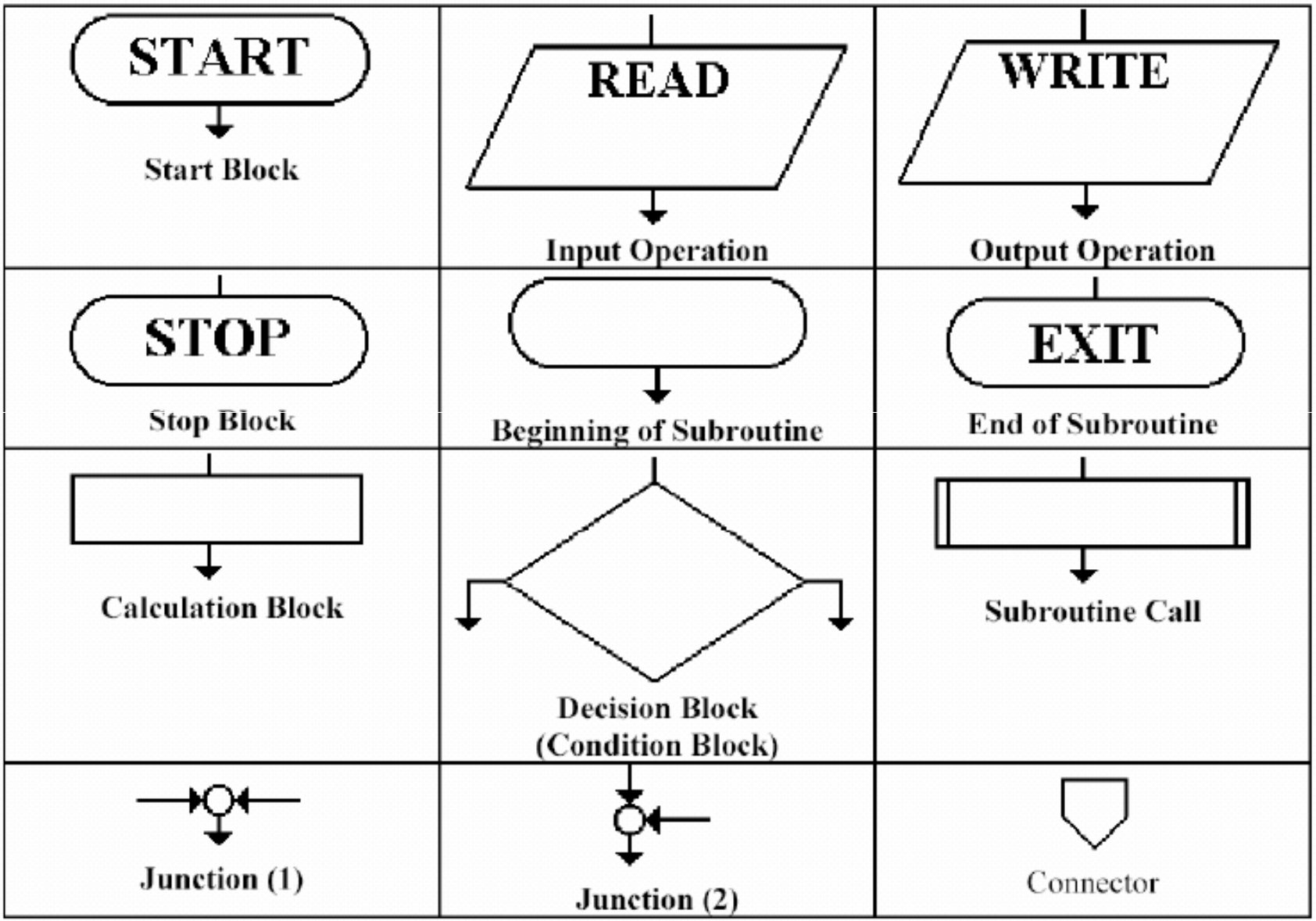
- Một chuỗi các mã nhị phân biểu diễn các công việc mà vi xử lý sẽ thực thi. Dạng dài các bit có thể được đơn giản hóa bằng dạng số Hex hay Octal. Ngôn ngữ này khó lập trình. Các vi xử lý khác nhau sẽ có những ngôn ngữ máy khác nhau.
- Thí dụ:

Địa chỉ bộ nhớ	Nội dung (nhị phân)	Nội dung (hex)	Tác vụ
00100H	11100100	E4	INPUT FROM
00101H	00000101	05	Port 05H
00102H	00000100	04	ADD
00103H	00000111	07	07H
00104H	11100110	E6	OUTPUT O
00105H	00000010	02	PORT 02

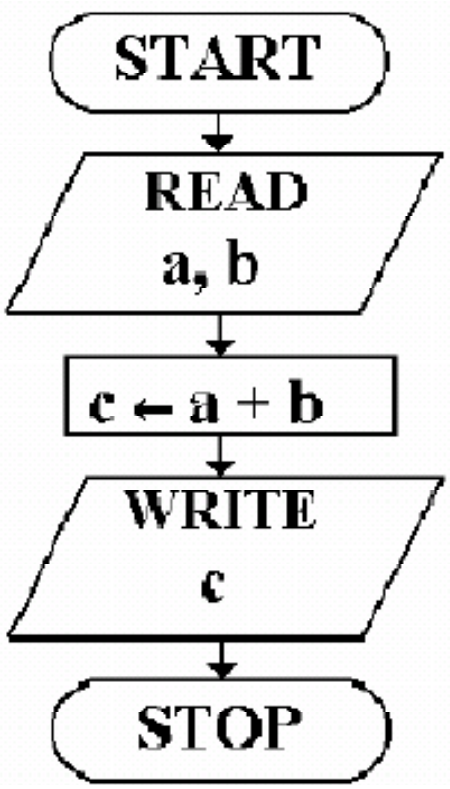


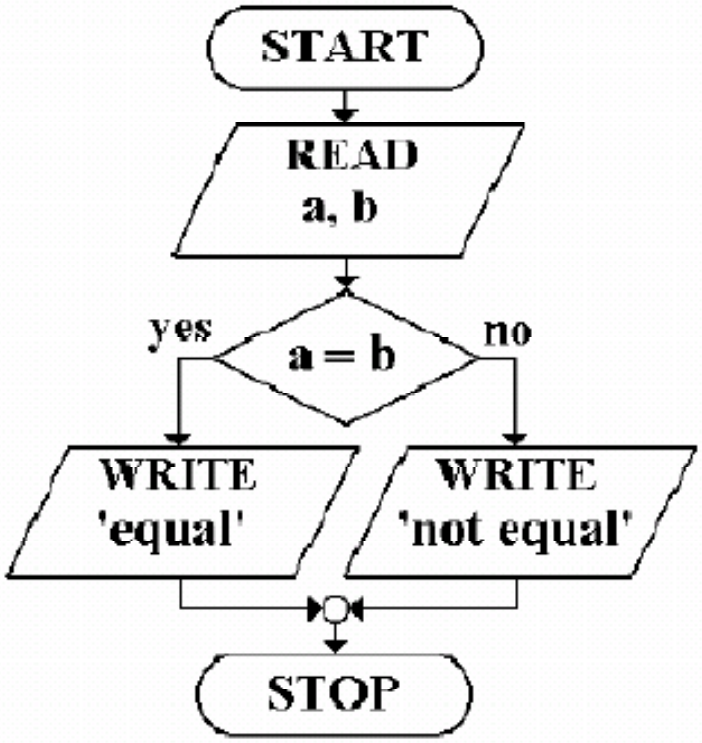
Hình 1.50 Quá trình tạo mã đối tượng khả thi.

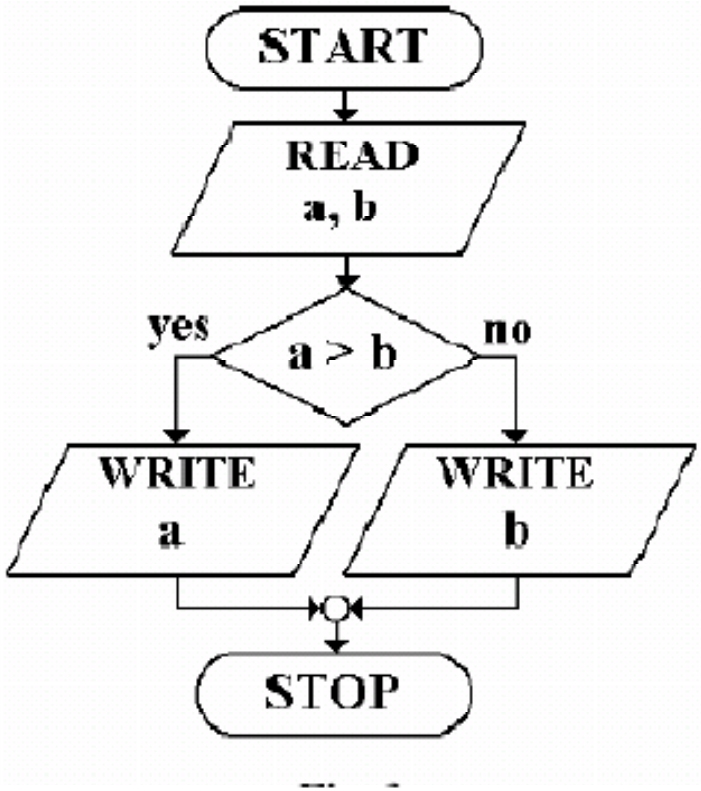
Lưu đồ chương trình (Program flowchart)



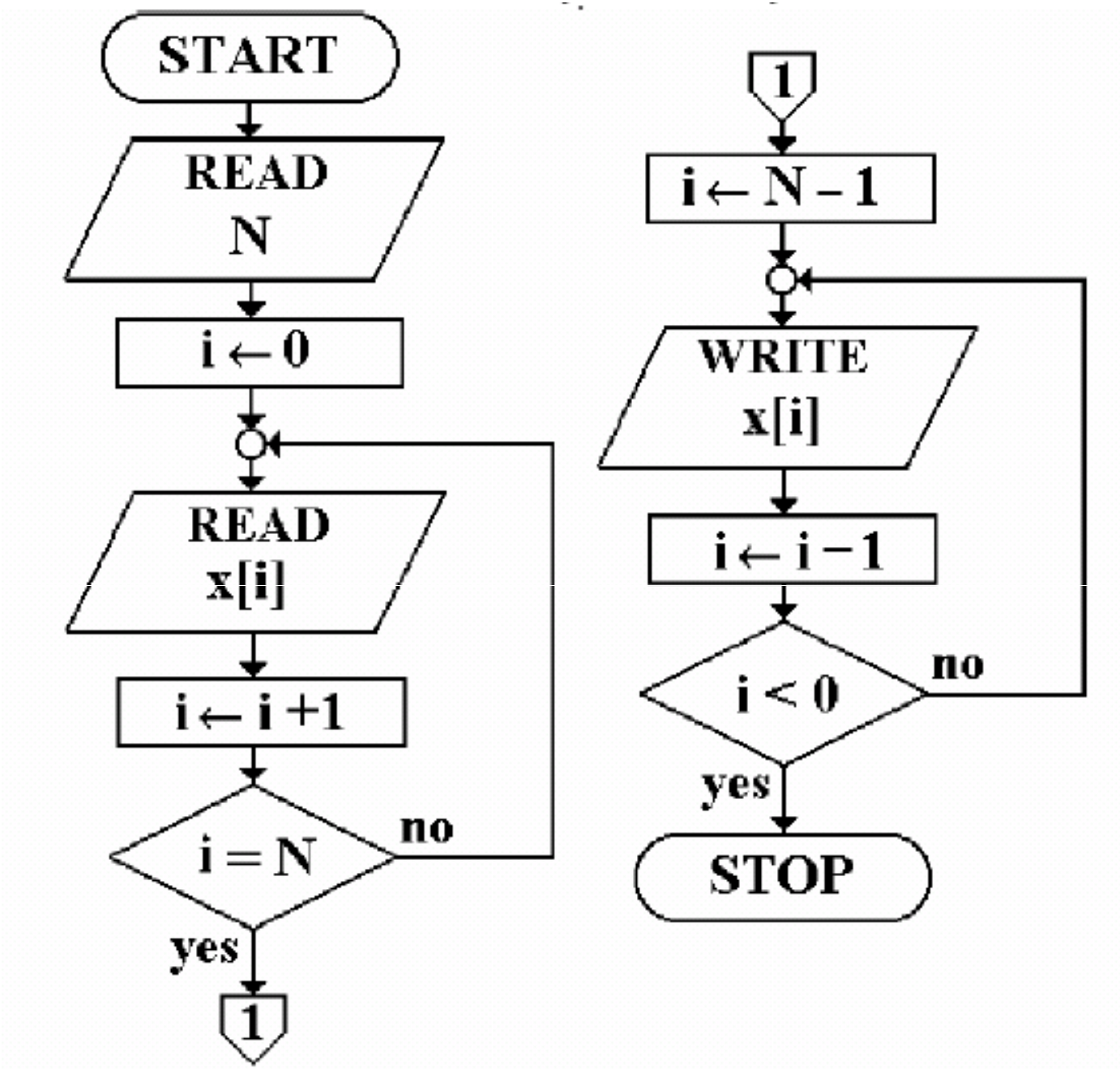
Hình 1.51 Các phần tử lưu đồ chính







Hình 1.52 Thí dụ các lưu đồ.



Hình 1.53 Thí dụ lưu đồ tính tổng vector X.

Nội dung

1.1 Sự phát triển của các hệ vi xử lý

1.2 Sơ đồ khối một hệ vi xử lý cơ bản

1.3 CPU

1.4 Bộ nhớ

1.5 Ngoại vi

1.6 Bus hệ thống

1.7 Giãi mã địa chỉ

1.8 Định thì

1.9 Chương trình

1.10 Vi điều khiển và vi xử lý

1.10 VI ĐIỀU KHIỂN

Các giới hạn của vi xử lý

– Cần bộ nhớ ngoài để thực thi chương trình.
– Không thể giao tiếp trực tiếp với các thiết bị I/O.

So sánh vi xử lý (MPU) và vi điều khiển (MCU)

- MPU:
  - Được thiết kế để thực hiện chức năng CPU trong hệ máy vi tính.
  - Tập lệnh được sắp xếp để cho phép mã và một lượng lớn dữ liệu được chuyển đi giữa vi xử lý với bộ nhớ và thanh ghi ngoài.
  - Các tác vụ tác động với nhóm bit không nhỏ hơn 4 bit.
- MCU:
  - Được thiết kế để làm việc với mạch ngoài tối thiểu.
  - Tập lệnh đơn giản (khoảng 255 lệnh).
  - Các tác vụ có thể tác động lên từng bit.
- MCU là máy tính với tất cả trong một chip:

MCU = CPU + Bộ nhớ + Giao tiếp I/O

1.10 Vi điều khiển

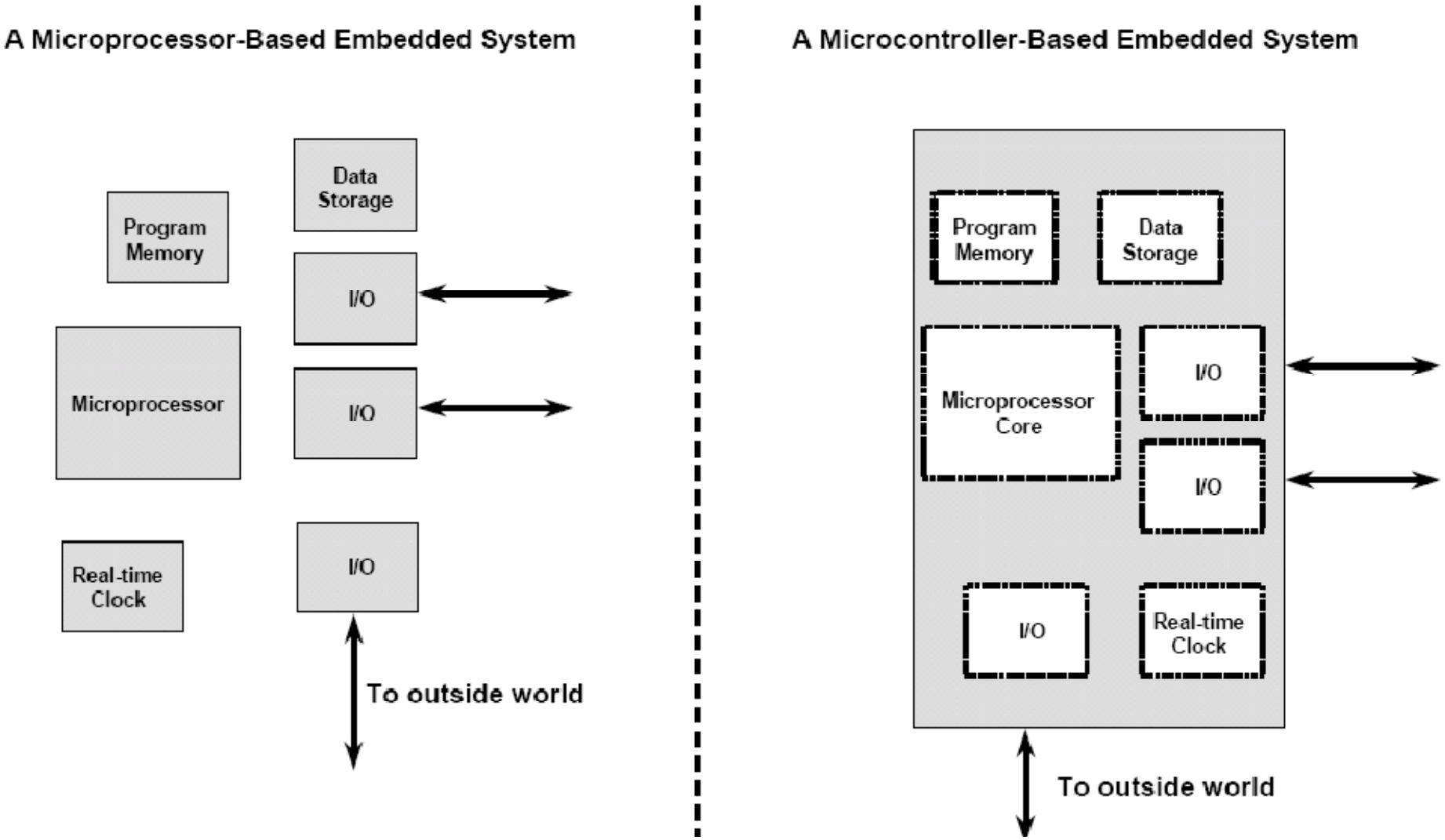
Vi xử lý (MPU)

- CPU là một chip riêng, RAM, ROM, I/O, timer là các phần riêng biệt
- Người thiết kế tùy ý chọn kích cỡ bộ nhớ, các cổng I/O …
- Có thể mở rộng được
- Đa chức năng
- Đa mục đích

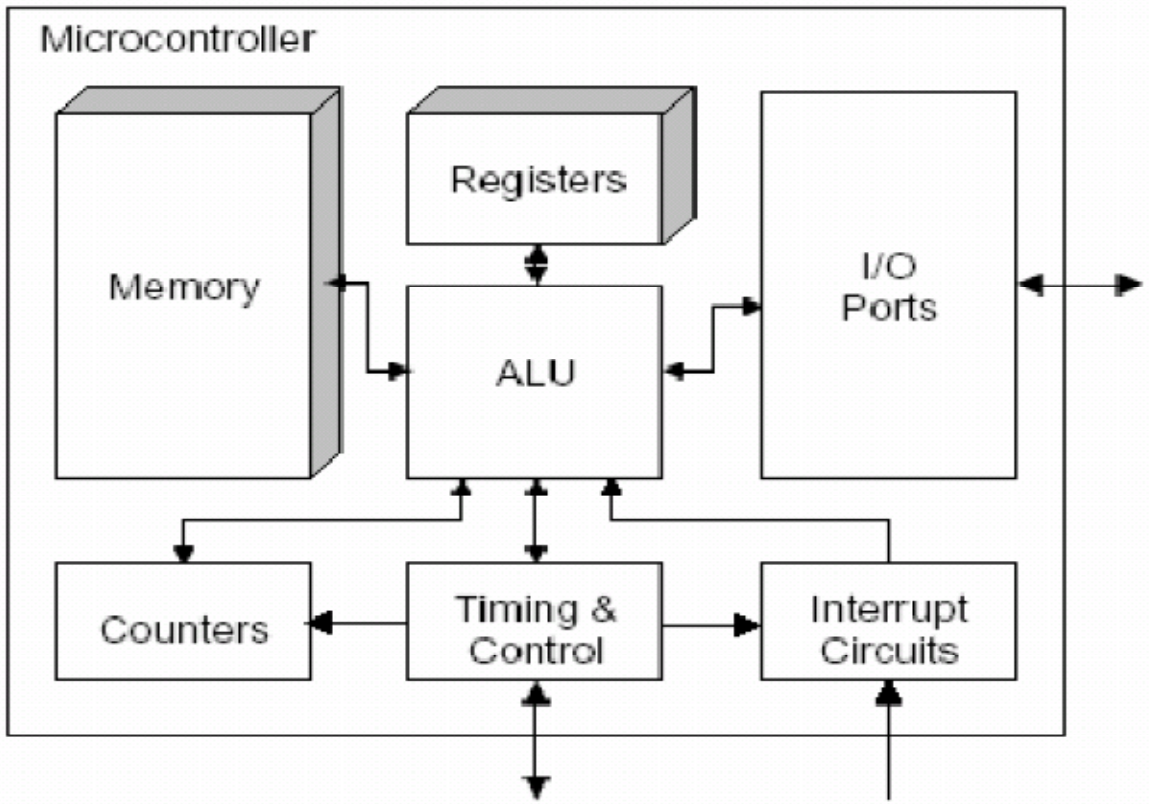
Vi điều khiển (MCU)

- CPU, RAM, ROM, I/O và timer đều trong 1 chip
- On-chip ROM, RAM và I/O port là cố định
- cho ứng dụng mà giới han về giá cả, năng lượng và không gian
- Chỉ có 1 mục đích

Microprocessor and Microcontroller



Sơ đồ khối của một MCU



Hình 1.54 Sơ đồ khối một MCU

Các MCU tiêu biểu

- 8051 (Intel và các hãng khác): là MCU thế hệ thứ hai của Intel.
- 68HC11 (Motorola và Toshiba): có kiến trúc bộ nhớ chung trong đó các lệnh, dữ liệu, I/O và các mạch định thì tất cả chia xẻ cùng vùng nhớ.
- PIC (Microchip): Họ MCU RISC đầu tiên (33 lệnh).