CHƯƠNG 1: ĐẠI CƯƠNG

1. Các hệ thống số dùng trong máy tính và các loại mã

1.1. Hệ thập phân (Decimal Number System)

Trong thực tế, ta thường dùng hệ thập phân để biểu diễn các giá trị số. Ở hệ thống này, ta dùng các tổ hợp của các chữ số 0..9 để biểu diễn các giá trị. Một số trong hệ thập phân được biểu diễn theo các số mũ của 10.

VD: Số 5346,72 biểu diễn như sau:
$5346,72 = 5.10^3 + 3.10^2 + 4.10 + 6 + 7.10^{-1} + 2.10^{-2}$

Tuy nhiên, trong các mạch điện tử, việc lưu trữ và phân biệt 10 mức điện áp khác nhau rất khó khăn nhưng việc phân biệt hai mức điện áp thì lại dễ dàng. Do đó, người ta sử dụng hệ nhị phân để biểu diễn các giá trị trong hệ thống số.

1.2. Hệ nhị phân (Binary Number System)

Hệ nhị phân chỉ dùng các chữ số 0 và 1 để biểu diễn các giá trị số. Một số nhị phân (binary digit) thường được gọi là ***bit***. Một chuỗi gồm 4 bit nhị phân gọi là ***nibble***, chuỗi 8 bit gọi là ***byte***, chuỗi 16 bit gọi là ***word*** và chuỗi 32 bit gọi là ***double word***. Chữ số nhị phân bên phải nhất của chuỗi bit gọi là ***bit có ý nghĩa nhỏ nhất*** (***least significant bit – LSB***) và chữ số nhị phân bên trái nhất của chuỗi bit gọi là ***bit có ý nghĩa lớn nhất*** (***most significant bit – MSB***). Một số trong hệ nhị phân được biểu diễn theo số mũ của 2. Ta thường dùng chữ ***B*** cuối chuỗi bit để xác định đó là số nhị phân.

VD: Số 101110.01b biểu diễn giá trị số:
$101110.01b = 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 1x2^1 + 0 + 0x2^{-1} + 1x2^{-2}$

❖ **Chuyển số nhị phân thành số thập phân:**

Để chuyển một số nhị phân thành một số thập phân, ta chỉ cần nhân các chữ số của số nhị phân với giá trị thập phân của nó và cộng tất cả các giá trị lại.
VD: $1011.11B = 1x2^3 + 0x2^2 + 1x2^1 + 1 + 1x2^{-1} + 1x2^{-2} = 11.75$

❖ **Chuyển số thập phân thành số nhị phân:**

Để chuyển một số thập phân thành số nhị phân, ta dùng 2 phương pháp sau:

- **Phương pháp 1**: Ta lấy số thập phân cần chuyển trừ đi 2^i trong đó 2^i là số lớn nhất nhỏ hơn hay bằng số thập phân cần chuyển. Sau đó, ta lại lấy kết quả này và thực hiện tương tự cho đến 2^0 thì dừng. Trong quá trình thực hiện, ta sẽ ghi lại các giá trị 0 hay 1 cho các bit tuỳ theo trường hợp số thập phân nhỏ hơn 2^i (0) hay lớn hơn 2^i (1).

VD: Xét số 21 thì số 2^i lớn nhất là 2^4

	2^4	2^3	2^2	2^1	2^0
	16	8	4	2	1
21 =	1	0	1	0	1
	5	5	1	1	0

(21 = 10101B)

- **Phương pháp 2**: Lấy số cần chuyển chia cho 2, ta nhớ lại số dư và lấy tiếp thương của kết quả trên chia cho 2 và thực hiện tương tự cho đến khi thương cuối cùng bằng 0. Kết quả chuyển đổi sẽ là chuỗi các bit là các số dư lấy theo thứ tự ngược lại.

VD: Chuyển 227 ra số nhị phân

Số bị chia	Thương	Số dư
227	113	1 (LSB)
113	56	1
56	28	0
28	14	0
14	7	0
7	3	1
3	1	1
1	0	1 (MSB)

(227 = 11100011b)

Để thực hiện chuyển các số thập phân nhỏ hơn 1 sang các số nhị phân, ta làm như sau: lấy số cần chuyển nhân với 2, giữ lại phần nguyên và lại lấy phần lẻ nhân với 2. Quá trình tiếp tục cho đến khi phần lẻ bằng 0 thì dừng. Kết quả chuyển đổi là chuỗi các bit là giá trị các phần nguyên.

VD: Chuyển 0.625 thành số nhị phân

0.625 × 2 = 1.25
0.25 × 2 = 0.5
0.5 × 2 = 1.0
(0.625 = 0.101b)

1.3. Hệ thập lục phân (Hexadecimal Number System)

Như đã biết ở trên, nếu dùng hệ nhị phân thì sẽ cần một số lượng lớn các bit để biểu diễn. Giả sử như số 1024 = 2^{10} sẽ cần 10 bit để biểu diễn. Để rút ngắn kết quả biểu diễn, ta dùng hệ thập lục phân dựa cơ sở trên số mũ của 16. Khi đó, 4 bit trong hệ nhị phân (1 nibble) sẽ biểu diễn bằng 1 chữ số trong hệ thập lục phân (gọi là số hex).

Trong hệ thống này, ta dùng các số 0..9 và các kí tự A..F để biểu diễn cho một giá trị số. Thông thường, ta dùng chữ h ở cuối để xác định đó là số thập lục phân.

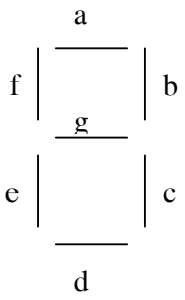
1.4. Mã BCD (Binary Coded Decimal)

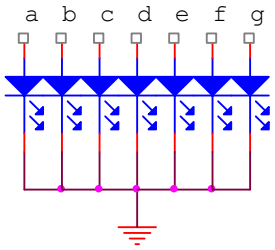
Trong thực tế, đối với một số ứng dụng như đếm tần, đo điện áp, … ngõ ra ở dạng số thập phân, ta dùng mã BCD. Mã BCD dùng 4 bit nhị phân để mã hoá cho một số thập phân 0..9. Như vậy, các số hex A..F không tồn tại trong mã BCD.

VD: Số thập phân 5 2 9
Số BCD 0101 0010 1001

1.5. Mã hiển thị Led 7 đoạn (7-segment display)

Đối với các ứng dụng dùng hiển thị số liệu ra Led 7 đoạn, ta dùng mã hiển thị Led 7 đoạn (bảng 1.1).





Bảng 1.1:

Số thập phân	Số thập lục phân	Số nhị phân	Mã Led 7 đoạn	
			a b c d e f g	Hiển thị
0	0	0000	1 1 1 1 1 1 0	0
1	1	0001	0 1 1 0 0 0 0	1
2	2	0010	1 1 0 1 1 0 1	2
3	3	0011	1 1 1 1 0 1 1	3
4	4	0100	0 1 1 0 0 1 1	4
5	5	0101	1 0 1 1 0 1 1	5
6	6	0110	1 0 1 1 1 1 1	6
7	7	0111	1 1 1 0 0 0 0	7
8	8	1000	1 1 1 1 1 1 1	8
9	9	1001	1 1 1 0 0 1 1	9
10	A	1010	1 1 1 1 1 0 1	A
11	B	1011	0 0 1 1 1 1 1	B
12	C	1100	0 0 0 1 1 0 1	C
13	D	1101	0 1 1 1 1 0 1	D
14	E	1110	1 1 0 1 1 1 1	E
15	F	1111	1 0 0 0 1 1 1	F

2. Các phép toán số học

2.1. Hệ nhị phân

2.1.1. Phép cộng

Phép cộng trong hệ nhị phân cũng thực hiện giống như trong hệ thập phân. Bảng sự thật của phép cộng 2 bit với 1 bit nhớ (carry) như sau:

Bảng 1.2:

Vào			Ra	
A	**B**	**C_{IN}**	**S**	**C_{OUT}**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = A \oplus B \oplus C_{IN}$
$C_{OUT} = AB + C_{IN}(A \oplus B)$

VD: 1001 1010
1 +1100 1100
Nhớ 0111 0110

2.1.2. Số bù 2 (2's component)

Trong hệ thống số thông thường, để biểu diễn số âm ta chỉ cần thêm dấu – vào các chữ số. Tuy nhiên, trong hệ thống máy tính, ta không thể biểu diễn được như trên. Phương pháp thông dụng là dùng bit có ý nghĩa lớn nhất (MSB) làm bit dấu (sign bit): nếu MSB = 1 sẽ là số âm còn MSB = 0 là số dương. Khi đó, các bit còn lại sẽ biểu diễn độ lớn (magnitude) của số. Như vậy, nếu ta dùng 8 bit để biểu diễn thì sẽ thu được 256 tổ hợp ứng với các giá trị 0..255 (số không dấu) hay –127.. –0 +0 … +127 (số có dấu).

Để thuận tiện hơn trong việc tính toán số có dấu, ta dùng một dạng biểu diễn đặc biệt là số bù 2. Số bù 2 của một số nhị phân xác định bằng cách lấy đảo các bit rồi cộng thêm 1.

VD: Số 7 biểu diễn là : 0000 0111 có MSB = 0 (biểu diễn số dương)
Số bù 2 là : 1111 1000 + 1 = 1111 1001. Số này sẽ đại diện cho số – 7.

Ta thấy, để thực hiện việc xác định số bù 2 của một số A, cần phải:

- Biểu diễn số A theo mã bù 2 của nó.
- Đảo các bit (tìm số bù 1 của A).
- Cộng thêm 1 vào để nhận được số bù 2.

Khi biểu diễn theo số bù 2, nếu sử dụng 8 bit ta sẽ có các giá trị số thay đổi từ -128..127.

2.1.3. Phép trừ

Phép trừ các số nhị phân cũng được thực hiện tương tự như trong hệ thập phân. Bảng sự thật của phép trừ 2 bit với 1 bit mượn (borrow) như sau:

Bảng 1.3:

Vào			Ra	
A	**B**	**BIN**	**D**	$\mathbf{B_{OUT}}$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$S = A \oplus B \oplus BIN$

$BOUT = \overline{A}B + \left(\overline{A \oplus B}\right)B_{IN}$

VD:

```
  0110 1101      = 149
- 0011 0001      =  49
  ---------
  0011 1100      = 100
```

Ngoài cách trừ như trên, ta cũng có thể thực hiện phép trừ thông qua số bù 2 của số trừ.

VD:

```
  0110 1101               0110 1101
- 0011 0001     →       + 1100 1111   ←┐
     |               1    0011 1100    |
Số bù 1 |               Nhớ              |
     ↓                                 |
  1100 1110 + 1 = 1100 1111 (Số bù 2) ──┘
```

Trong phép cộng với số bù 2, ta bỏ qua bit nhớ cuối cùng → kết quả phép cộng số bù 2 là 0011 1100. Đây cũng chính là kết quả phép trừ, bit MSB = 0 cho biết kết quả là số dương.

VD:

```
   77        0100 1101          0100 1101
-  88     -  0101 1000   →    + 1010 1000
 ----                         ---------
 - 11                           1111 0101
```

Số 88 = 0101 1000 → số bù 1 là 1010 0111 → số bù 2: 1010 1000

Kết quả phép cộng số bù 2 là 1111 0101 có MSB = 1 nên là số âm. Số bù 1 là 0000 1010 → số bù 2: 0000 1011. Kết quả này chính là 11 nên phép trừ sẽ cho kết quả là –11.

Ta thấy, để thực hiện chuyển số bù 2 thành số có dấu thì cần thực hiện:

- Lấy bù các bit để tìm số bù 1.
- Cộng với 1.
- Thêm dấu trừ để xác định là số âm.

2.1.4. Phép nhân

Phép nhân các số nhị phân cũng tương tự như đối với các số thập phân. Chú ý rằng đối với phép nhân nếu nhân 2 số 4 bit sẽ có kết quả là số 8 bit, 2 số 8 bit sẽ có kết quả là số 16 bit, …

VD:

```
  11          1011b
 X 9         1001b
 --          -----
 99          1011
            0000
           0000
          1011
          -------
          1100011b
```

Đối với máy tính, phép nhân được thực hiện bằng phương pháp cộng và dịch phải (add-and-right-shift):

- Thành phần dầu tiên của tổng sẽ chính là số bị nhân nếu như LSB của số nhân là 1. Ngược lại, nếu LSB của số nhân bằng 0 thì thành phần này bằng 0.
- Mỗi thành phần thứ i kế tiếp sẽ được tính tương tự với điều kiện là phải dịch trái số bị nhân i bit.
- Kết quả cần tìm chính là tổng các thành phần nói trên.

2.1.5. Phép chia

Phép chia các số nhị phân cũng tương tự như đối với các số thập phân.

VD: 30/5 = 6

```
11110  | 110
110    | ---
 011   | 101
 000   |
  110  |
  110  |
  ---  |
    0  |
```

Tương tự như đối với phép nhân, ta có thể dùng phép trừ và phép dịch trái cho đến khi không thể thực hiện phép trừ được nữa. Tuy nhiên, để thuận tiện cho tính toán, thay vì dùng phép trừ đối với số chia, ta sẽ thực hiện phép cộng đối với số bù 2 của số chia.

- Đổi số chia ra số bù 2 của nó.
- Lấy số bị chia cộng với số bù 2 của số chia.
  + Nếu kết quả này có bit dấu = 0 thì bit tương ứng của thương = 1.
  + Nếu kết quả này có bit dấu = 1 thì bit tương ứng của thương = 0 và ta phải khôi phục lại giá trị của số bị chia bằng cách cộng kết quả này với số chia.
- Dịch trái kết quả thu được và thực hiện tiếp tục như trên cho đến khi kết quả là 0 hay nhỏ hơn số chia.

2.2. Hệ thập lục phân

2.2.1. Phép cộng

Thực hiện chuyển các số hex cần cộng thành các số nhị phân, tính kết quả trên số nhị phân và sau đó chuyển lại thành số hex.

VD: 7Ah → 0111 1010
3Fh → 0011 1111
B9h ← 1011 1001

Thực hiện cộng trực tiếp trên số hex, nếu kết quả cộng lớn hơn 15 thì sẽ nhớ và trừ cho 16.

VD: 7 A
3 F
10_{10} 25_{10} → B9h

Ah + Fh = $10_{10} + 15_{10} = 25_{10}$ → nhớ 1 và $25_{10} - 16_{10} = 9_{10} = 9h$

7h + 3h = $7_{10} + 3_{10} = 10_{10}$ → cộng số nhớ: $10_{10} + 1_{10} = 11_{10} = Bh$

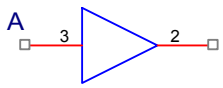
2.2.2. Phép trừ

Thực hiện tương tự như phép cộng.

3. Các thiết bị số cơ bản

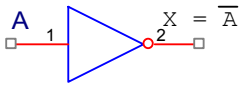
3.1. Cổng đệm (buffer) và các cổng logic (logic gate)

❖ **Cổng đệm:**



A	X
0	0
1	1

❖ **Cổng NOT:**

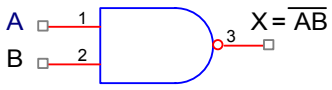


A	X
0	1
1	0

❖ **Cổng AND:**

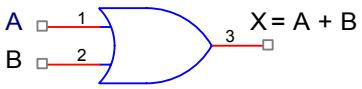
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

❖ **Cổng NAND:**



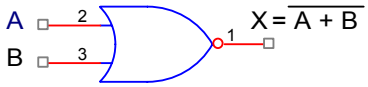
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

❖ **Cổng OR:**



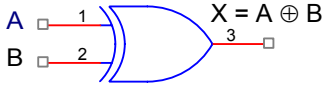
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

❖ **Cổng NOR:**



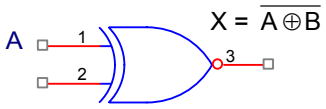
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

❖ **Cổng EX-OR:**



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

❖ **Cổng EX-NOR:**



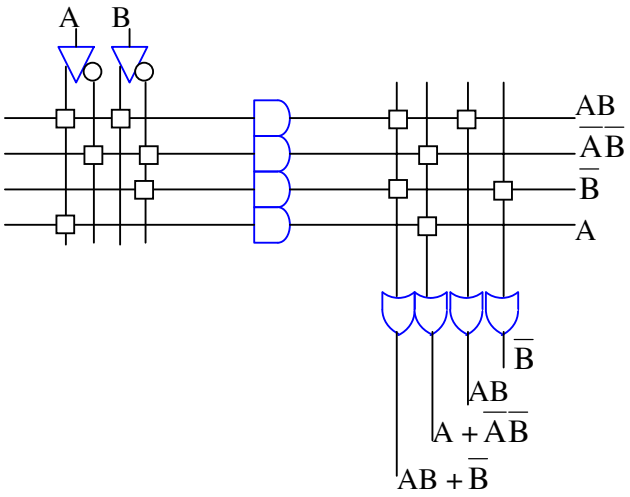
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

3.2. Thiết bị logic lập trình được

Thay vì sử dụng các cổng logic rời rạc, ta có thể dùng các thiết bị logic lập trình được (programmable logic device) như PLA (Programmable Logic Array), PAL (Programmable Array Logic) hay PROM (Programmable Read Only Memory) để liên kết các thiết bị LSI (Large Scale Intergration).

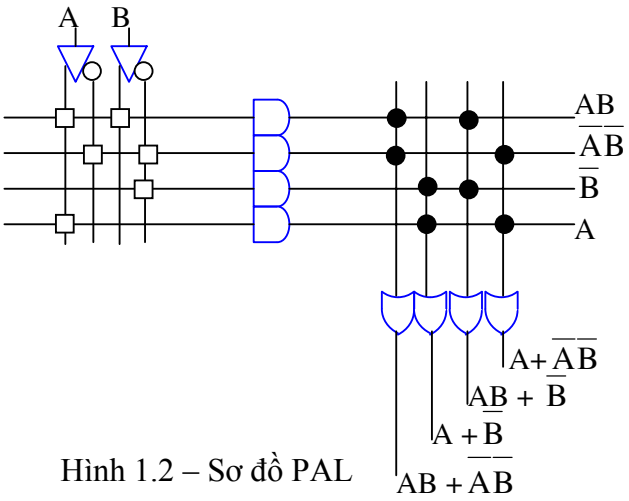
❖ **PLA (hay FPLA – Field PLA):**

Dùng ma trận cổng AND và OR để lập trình bằng cácc phá huỷ các cầu chì. FPLA rất linh động nhưng lại khó lập trình.



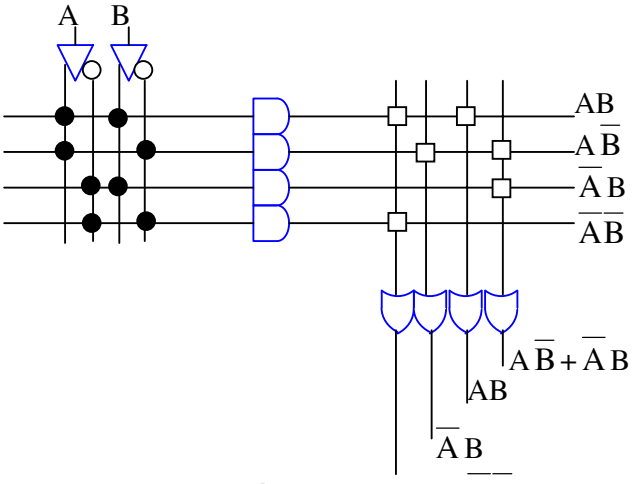
Hình 1.1 – Sơ đồ PLA

❖ **PAL**: ma trận OR đã cố định sẵn và ta chỉ lập trình trên ma trận AND.



Hình 1.2 – Sơ đồ PAL

❖ **PROM**: ma trận AND cố định sẵn và ta chỉ lập trình trên ma trận OR.

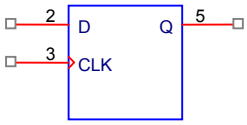


Hình 1.3 – Sơ đồ PROM

3.3. Chốt, flipflop và thanh ghi

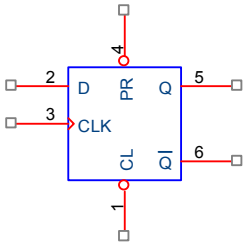
❖ **Chốt (latch):**

Chốt là thiết bị số lưu trữ lại giá trị số tại ngõ ra của nó.



D	CLK	Q
X	0	QN
0	1	0
1	1	1

❖ **Flipflop:**



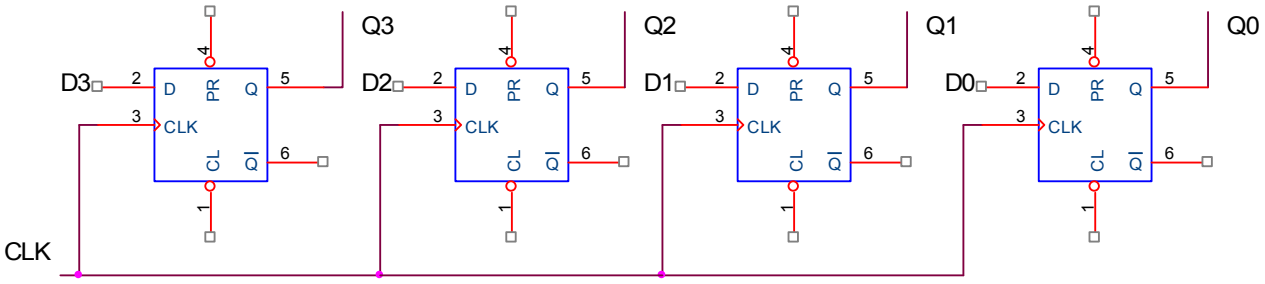
PR	CL	D	CLK	Q	$\overline{Q}$
1	1	1	↑	1	0
1	1	0	↑	0	1
1	1	X	0	Q_N	$\overline{Q_N}$
1	1	X	1	Q_N	$\overline{Q_N}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	.	.

CL: clear PR: Preset CLK: Clock

- Nếu xuất hiện cạnh lên của tín hiệu CLK thì ngõ ra Q sẽ có giá trị theo dữ liệu tại D.
- Nếu PR = 0 thì Q = 1. Nếu CL = 0 thì Q = 0.
- Trạng thái PR = CL = 0 là trạng thái cấm, ngõ ra sẽ không ổn định.

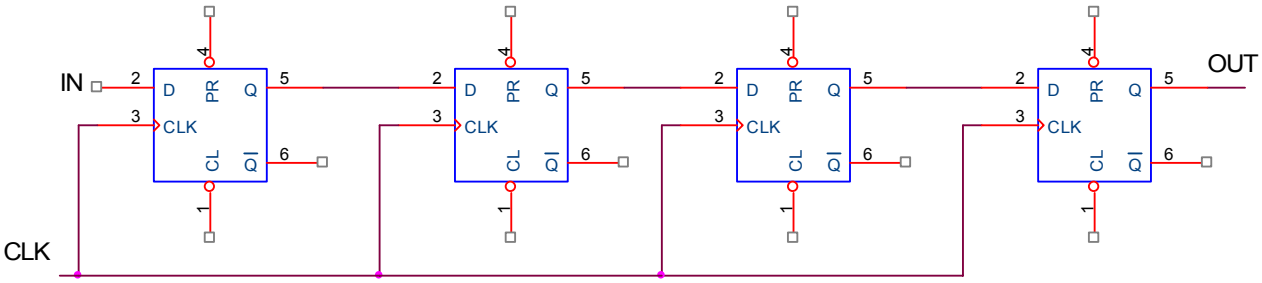
❖ **Thanh ghi (register):**

Thanh ghi là một nhóm các flipflop được kết nối song song để lưu trữ các số nhị phân. Giá trị nhị phân sẽ được đưa vào ngõ vào của các flipflop. Khi có tác động cạnh lên của tín hiệu CLK thì ngõ ra các flipflop sẽ lưu trữ giá trị nhị phân cho đến khi một số nhị phân mới được đưa vào và tác động một cạnh len cho tín hiệu CLK.



Hình 1.4 – Thanh ghi dạng đơn giản

Trong trường hợp các flipflop được kết nối nối tiếp với nhau, ta sẽ có thanh ghi dịch (shift register).



Hình 1.5 – Thanh ghi dịch

3.4. Bộ nhớ

3.4.1. Các kiểu bộ nhớ

❖ **ROM (Read Only Memory):**

Đặc tính chung của ROM là dữ liệu lưu trữ sẽ không bị mất đi dù cho không còn nguồn cung cấp cho ROM (tính nonvolatile – ổn định). Ta chỉ có thể thực hiện tác vụ đọc đối với ROM. ROM có thể được chia thành: ROM che mặt nạ (Masked ROM), PROM (ROM lập trình được), EPROM (ROM có thể xoá bằng tia cực tím) và EEPROM (ROM có thể xoá bằng điện).

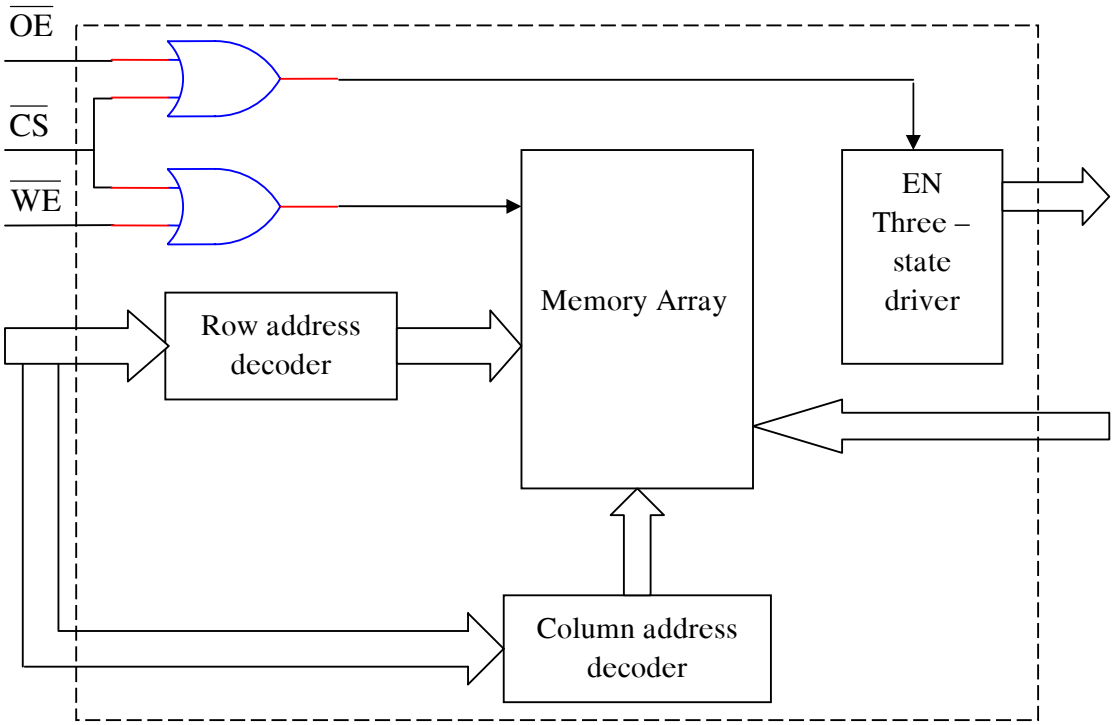
❖ **RAM (Random Access Memory):**

RAM có đặc tính là tất cả nội dung chứa trong RAM sẽ bị mất đi khi không còn nguồn cung cấp cho RAM (tính volatile – không ổn định). Có 2 loại RAM: tĩnh và động.

- **SRAM (Static RAM):** dùng các ma trận flipflop để lưu trữ dữ liệu nên ta có thể ghi các giá trị nhị phân vào RAM bằng cách đưa dữ liệu vào các ngõ vào các flipflop và cấp xung clock cho các flipflop này.

- **DRAM (Dynamic RAM):** tạo ra bằng các cổng transistor và lưu trữ bằng điện tích. Tuy nhiên, do hiện tượng rò rỉ điện tích theo thời gian, ta phải thực hiện nạp điện lại. Quá trình này gọi là làm tươi (refreshing) bộ nhớ. Thuận lợi của DRAM là một số lượng lớn transistor có thể được đặt trên một chip nhớ nên nó có dung lượng cao hơn và nhanh hơn SRAM.

3.4.2. Cấu trúc bên trong của bộ nhớ



Hình 1.6 – Cấu trúc nội một bộ nhớ tiêu biểu

$\overline{CS}$ (Chip Select):cho phép bộ nhớ hoạt động

$\overline{OE}$ (Output Enable): cho phép đọc dữ liệu từ bộ nhớ ra bên ngoài

$\overline{WE}$ (Write Enable): cho phép ghi dữ liệu vào trong bộ nhớ

Row address decoder, Column address decoder: các bộ giải mã hàng và cột để chọn vị trí của memory cell (flipflop hay tụ điện)

Three-state driver: bộ lái ngõ ra 3 trạng thái để đệm ngõ ra

4. Giới thiệu vi xử lý

4.1. Các thế hệ vi xử lý

- ***Thế hệ 1 (1971 – 1973):*** vi xử lý 4 bit, đại diện là 4004, 4040, 8080 (Intel) hay IPM-16 (National Semiconductor).
  + Độ dài word thường là 4 bit (có thể lớn hơn).
  + Chế tạo bằng công nghệ PMOS với mật độ phần tử nhỏ, tốc độ thấp, dòng tải thấp nhưng giá thành rẻ.
  + Tốc độ 10 ÷ 60 µs / lệnh với tần số xung nhịp 0.1 ÷ 0.8 MHz.
  + Tập lệnh đơn giản và phải cần nhiều vi mạch phụ trợ.
- ***Thế hệ 2 (1974 – 1977):*** vi xử lý 8 bit, đại diện là 8080, 8085 (Intel) hay Z80 (Zilog).
  + Tập lệnh phong phú hơn.
  + Địa chỉ có thể đến 64 KB. Một số bộ vi xử lý có thể phân biệt 256 địa chỉ cho thiết bị ngoại vi.
  + Sử dụng công nghệ NMOS hay CMOS.

+ Tốc độ 1 ÷ 8 µs / lệnh với tần số xung nhịp 1 ÷ 5 MHz

- ***Thế hệ 3 (1978 – 1982):*** vi xử lý 16 bit, đại diện là 68000/68010 (Motorola) hay 8086/80286/80386 (Intel)

+ Tập lệnh đa dạng với các lệnh nhân, chia và xử lý chuỗi.

+ Địa chỉ bộ nhớ có thể từ 1 ÷ 16 MB và có thể phân biệt tới 64KB địa chỉ cho ngoại vi

+ Sử dụng công nghệ HMOS.

+ Tốc độ 0.1 ÷ 1 µs / lệnh với tần số xung nhịp 5 ÷ 10 MHz.

- ***Thế hệ 4:*** vi xử lý 32 bit 68020/68030/68040/68060 (Motorola) hay 80386/80486 (Intel) và vi xử lý 32 bit Pentium (Intel)

+ Bus địa chỉ 32 bit, phân biệt 4 GB bộ nhớ.

+ Có thể dùng thêm các bộ đồng xử lý (coprocessor).

+ Có khả năng làm việc với bộ nhớ ảo.

+ Có các cơ chế pipeline, bộ nhớ cache.

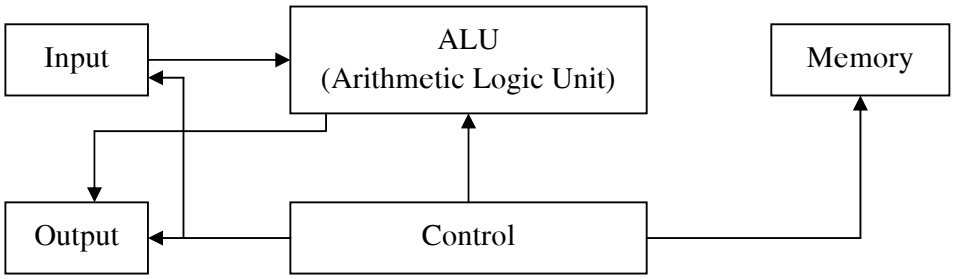
+ Sử dụng công nghệ HCMOS.

4.2. Vi xử lý (µP – microproccessor)

4.2.1. Phân loại vi xử lý

- ***Multi chip***: dùng 2 hay nhiều chip LSI (Large Scale Intergration: tích hợp từ 1000 ÷ 10000 transistor) cho ALU và control.
- ***Microprocessor***: dùng 1 chip LSI/VLSI (Very Large Scale Intergration: tích hợp ÷ 10000 transistor) cho ALU và control.
- ***Single chip microprocessor*** (còn gọi là microcomputer / microcontroller): là 1 chip LSI/VLSI chứa toàn bộ các khối như hình 1.7.

4.2.2. Sơ đồ khối một máy tính cổ điển



Hình 1.7 – Sơ đồ khối một máy tính cổ điển

- *ALU (đơn vị logic số học)*: thực hiện các bài toán cho máy tính bao gồm: +, -, *, /, phép toán logic, …
- *Control (điều khiển)*: điều khiển, kiểm soát các đường dữ liệu giữa các thành phần của máy tính.
- *Memory (bộ nhớ)*: lưu trữ chương trình hay các kết quả trung gian.
- *Input (nhập), Output (Xuất)*: các thiết bị xuất nhập dữ liệu (còn gọi là thiết bị ngoại vi).

4.2.3. Sơ đồ khối của µP

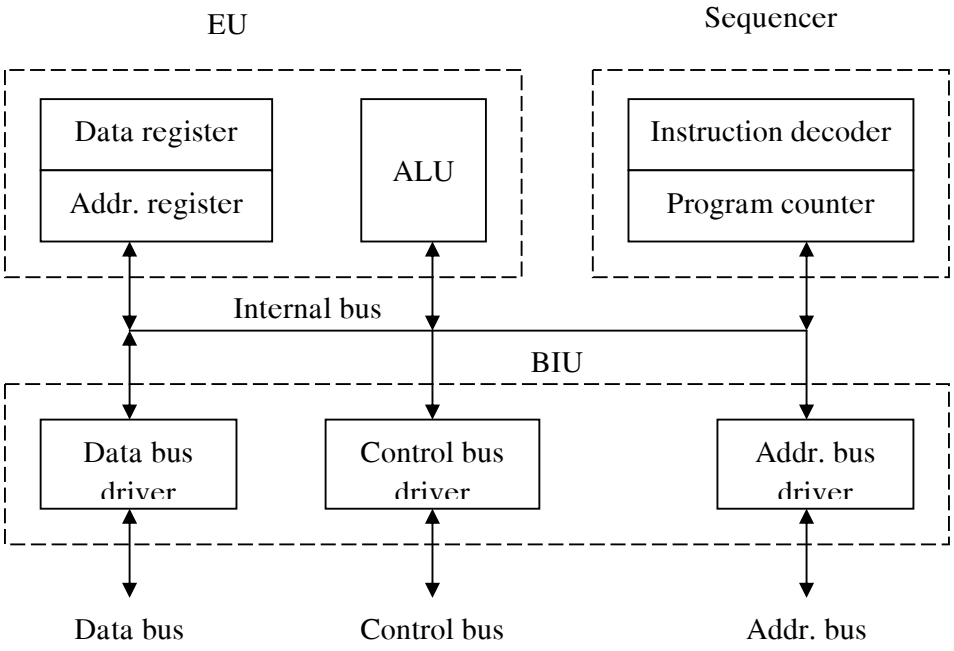
Có 3 khối chức năng: đơn vị thực thi (EU - Execution unit), bộ tuần tự (Sequencer) và đơn vị giao tiếp bus (BIU – Bus interface unit).

- EU: thực hiện các lệnh số học và logic. Các toán hạng được chứa trong các thanh ghi dữ liệu (data register) hay thanh ghi địa chỉ (address register), hay từ bus nội (internal bus).

- Bộ tuần tự: gồm bộ giải mã lệnh (instruction decoder) và bộ đếm chương trình (program counter)

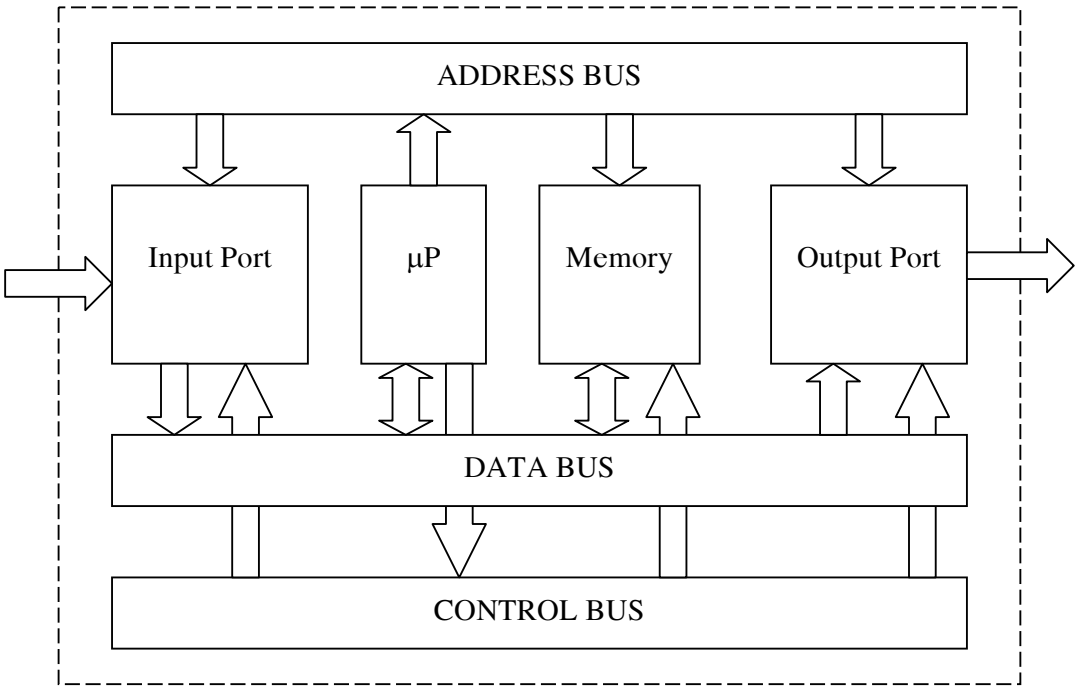
  + Bộ đếm chương trình chứa các lệnh kế tiếp sẽ thực hiện
  + Bộ giải mã sẽ thực hiện các bước cần thiết để thực thi lệnh.

Khi chương trình bắt đầu, bộ đếm chương trình (PC) sẽ ở địa chỉ bắt đầu. Địa chỉ này được chuyển qua bộ nhớ thông qua address bus. Khi tín hiệu Read đưa vào control bus, nội dung bộ nhớ liên quan sẽ đưa vào bộ giải mã lệnh. Bộ giải mã lệnh sẽ khởi động các phép toán cần thiết để thực thi lệnh. Quá trình này đòi hỏi một số chu kỳ máy (machine cycle) tuỳ theo lệnh. Sau khi lệnh đã thực thi, bộ giải mã lệnh sẽ đặt PC đến địa chỉ của lệnh kế.



Hình 1.8 – Sơ đồ khối của vi xử lý

4.2.4. Sơ đồ khối của hệ vi xử lý cơ bản



Hình 1.9 – Sơ đồ khối hệ vi xử lý

Mọi hoạt động cơ bản của một hệ vi xử lý đều giống nhau, không phụ thuộc loại vi xử lý hay quá trình thực hiện. µP sẽ đọc một lệnh từ bộ nhớ (memory), thực thi lệnh và sau đó đọc lệnh kế. Quá trình đọc lệnh gọi là instruction fetch còn quá trình thực hiện tuần tự như trên gọi là fetch – execute sequence. Tuy nhiên có một số µP sẽ nhận một số lệnh rồi mới bắt đầu thực thi.

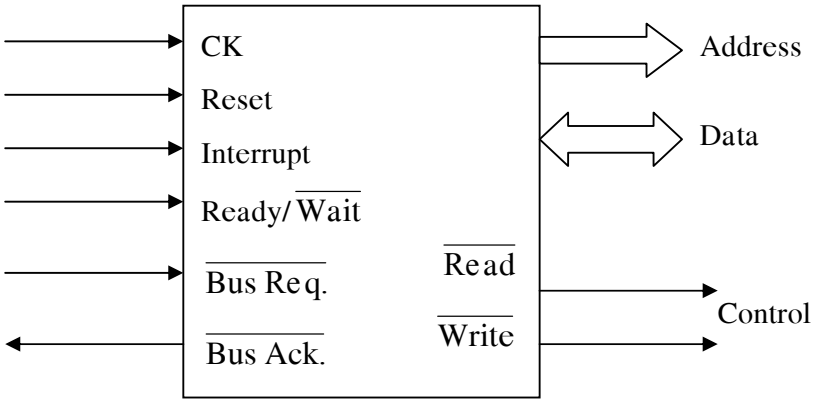
❖ **Các port I/O:**

Các port nhập (input) và xuất (output) dùng để giao tiếp giữa µP và thiết bị ngoại vi (không thể nối trực tiếp với các bus).

Port xuất là một thanh ghi. Khi µP ghi dữ liệu ra địa chỉ của Port thì Port sẽ chứa dữ liệu hiện tại trên data bus. Dữ liệu này sẽ được chốt tại Port cho đến khi µP ghi dữ liệu mới ra Port.

Port nhập là một driver 3 trạng thái. Khi µP đọc vào từ địa chỉ của Port, driver 3 trạng thái lái dữ liệu từ bên ngoài vào data bus. Sau đó, µP đọc dữ liệu từ bus.

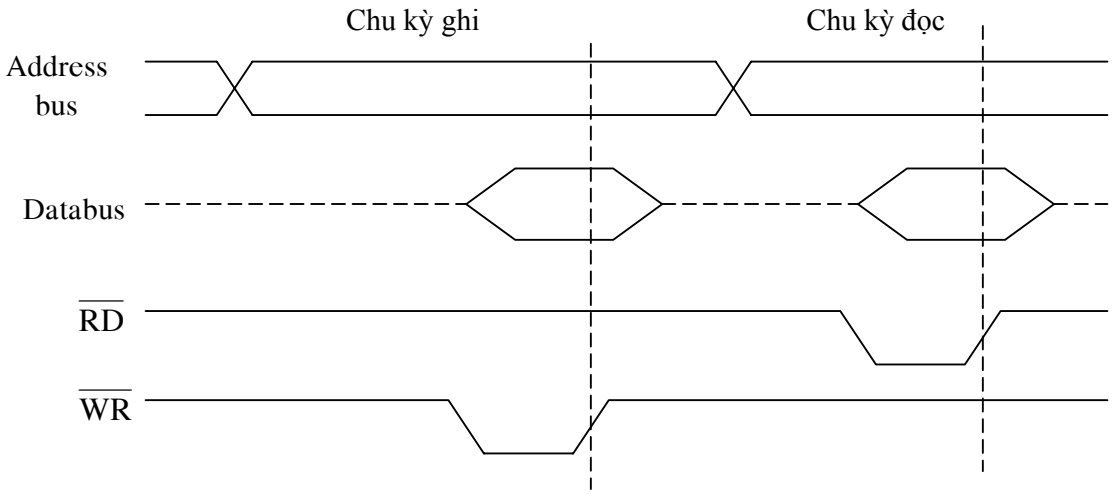
❖ **Các tín hiệu tiêu biểu của một µP:**



Hình 1.10 – Các tín hiệu cơ bản trong µP

Các bus dùng để liên kết các thành phần của hệ thống với µP. µP sẽ chọn một thiết bị cần sử dụng thông qua address bus và đọc hay ghi dữ liệu thông qua data bus. Data bus là bus 2 chiều, dùng chung cho tất cả các quá trình trao đổi dữ liệu. Mỗi chu kỳ bus (bus cycle) là việc thực hiện trao đổi một từ dữ liệu giữa µP và ô nhớ hay thiết bị I/O.

Mỗi chu kỳ bus bắt đầu khi µP xuất một địa chỉ nhằm chọn thiết bị I/O hay chọn một ô nhớ nào đó.



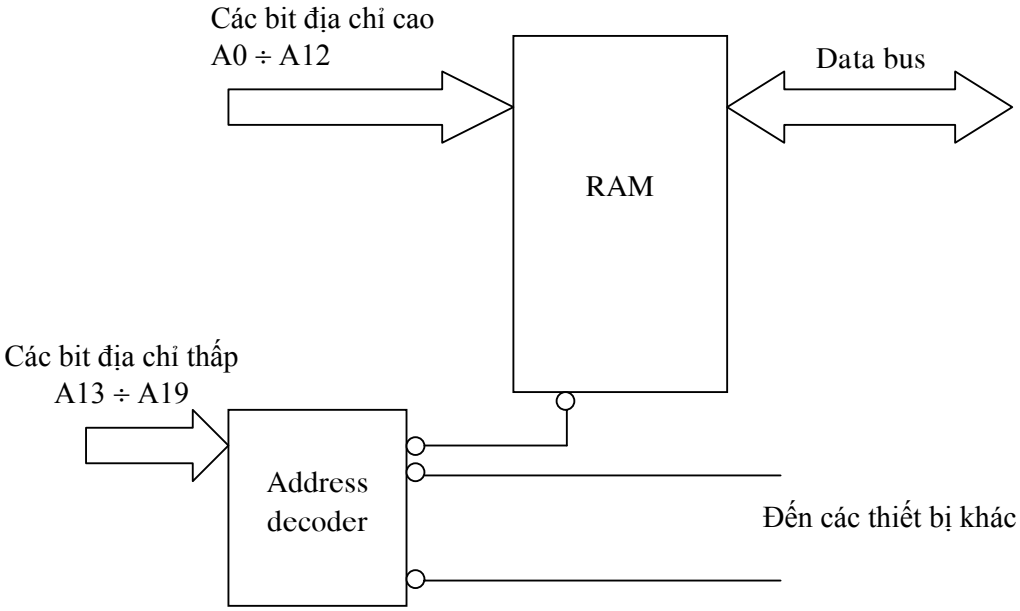
Hình 1.11 – Định thì bus cơ bản

4.3. Giao tiếp với bộ nhớ

4.3.1. Giao tiếp bus cơ bản

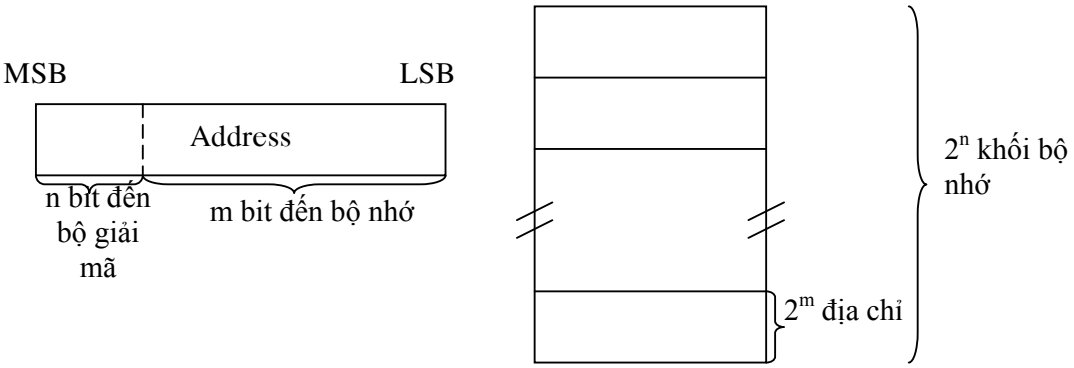
- Các bit địa chỉ thấp (giả sử 13 đường A0 ÷ A12) nối trực tiếp đến chip bộ nhớ (giả sử RAM có dung lượng 8K × 8)
- Các bit địa chỉ cao (giả sử A13 ÷ A19) nối với bộ giải mã địa chỉ (address decoder) tạo tín hiệu cho phép chip bộ nhớ. Do đó, khi thiết kế ta phải xác

định mỗi chip bộ nhớ thuộc vùng địa chỉ nào. Tập hợp các vùng này theo bảng gọi là bảng bộ nhớ (memory map).



Hình 1.12 – Giao tiếp bus cơ bản

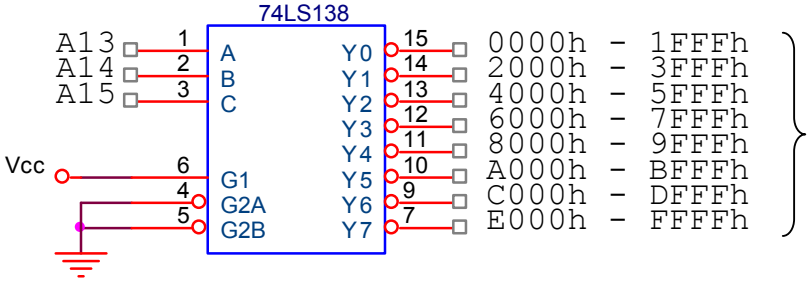
Quan hệ giữa giải mã địa chỉ và bảng bộ nhớ:



Hình 1.13 – Bảng bộ nhớ

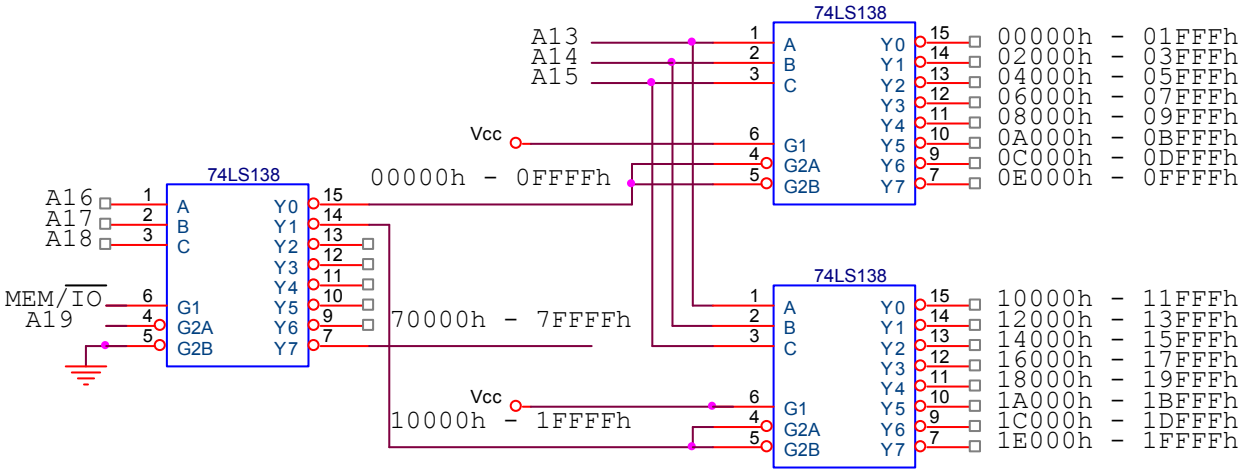
4.3.2. Giải mã địa chỉ

4.3.2.1. Dùng 74LS138



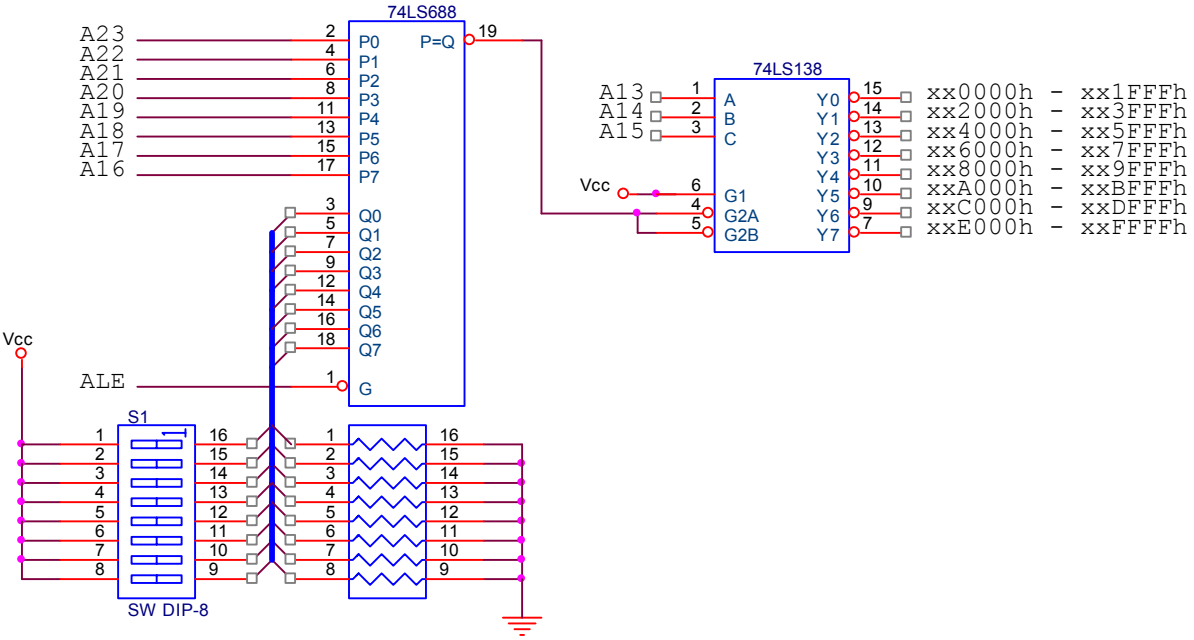
Hình 1.14 – Giải mã địa chỉ dùng 74LS138

4.3.2.2. Dùng nhiều 74LS138



Hình 1.15 – 74LS138 mắc cascaded (xâu chuỗi)

4.3.2.3. Dùng bộ so sánh



Hình 1.16 – Giải mã dùng bộ so sánh

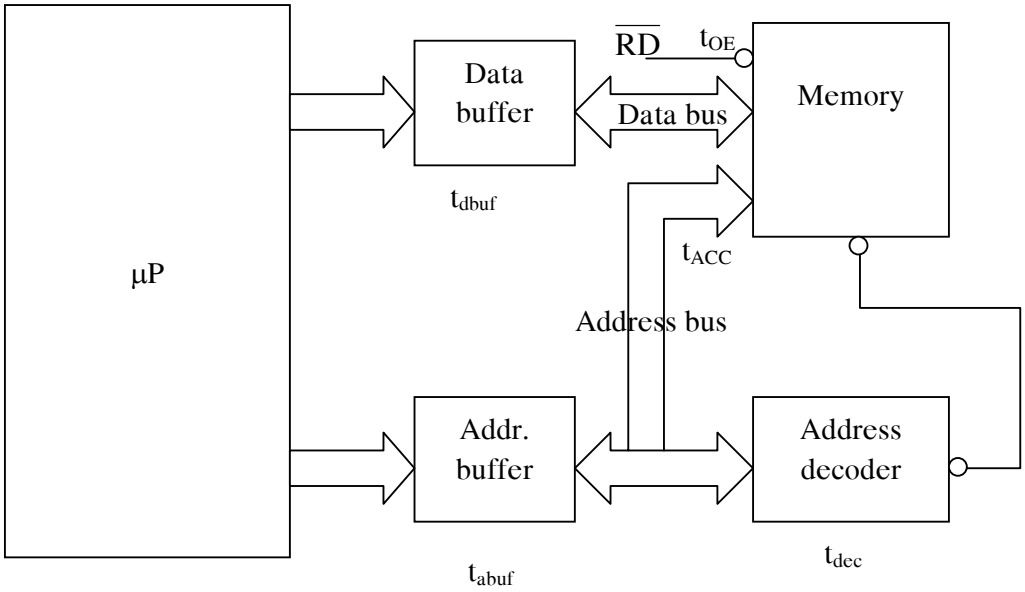
4.3.3. Định thì bộ nhớ

❖ **Thời gian truy xuất (access time):**

- Với chu kỳ đọc: thời gian truy xuất là thời gian tính từ lúc địa chỉ mới xuất hiện ở bộ nhớ cho đến khi có dữ liệu đúng ở ngõ ra của bộ nhớ.
- Với chu kỳ ghi: thời gian truy xuất là thời gian tính từ lúc địa chỉ mới xuất hiện ở bộ nhớ cho đến khi dữ liệu đã đưa vào bộ nhớ.

❖ **Thời gian chu kỳ (cycle time):** là thời gian từ lúc bắt đầu chu kỳ bộ nhớ đến khi bắt đầu chu kỳ kế tiếp.

Ngoài ra, µP có thể sử dụng thêm một số trạng thái chờ khi đọc bộ nhớ.



Hình 1.17 – Các đường trì hoãn trong giao tiếp µP với bộ nhớ

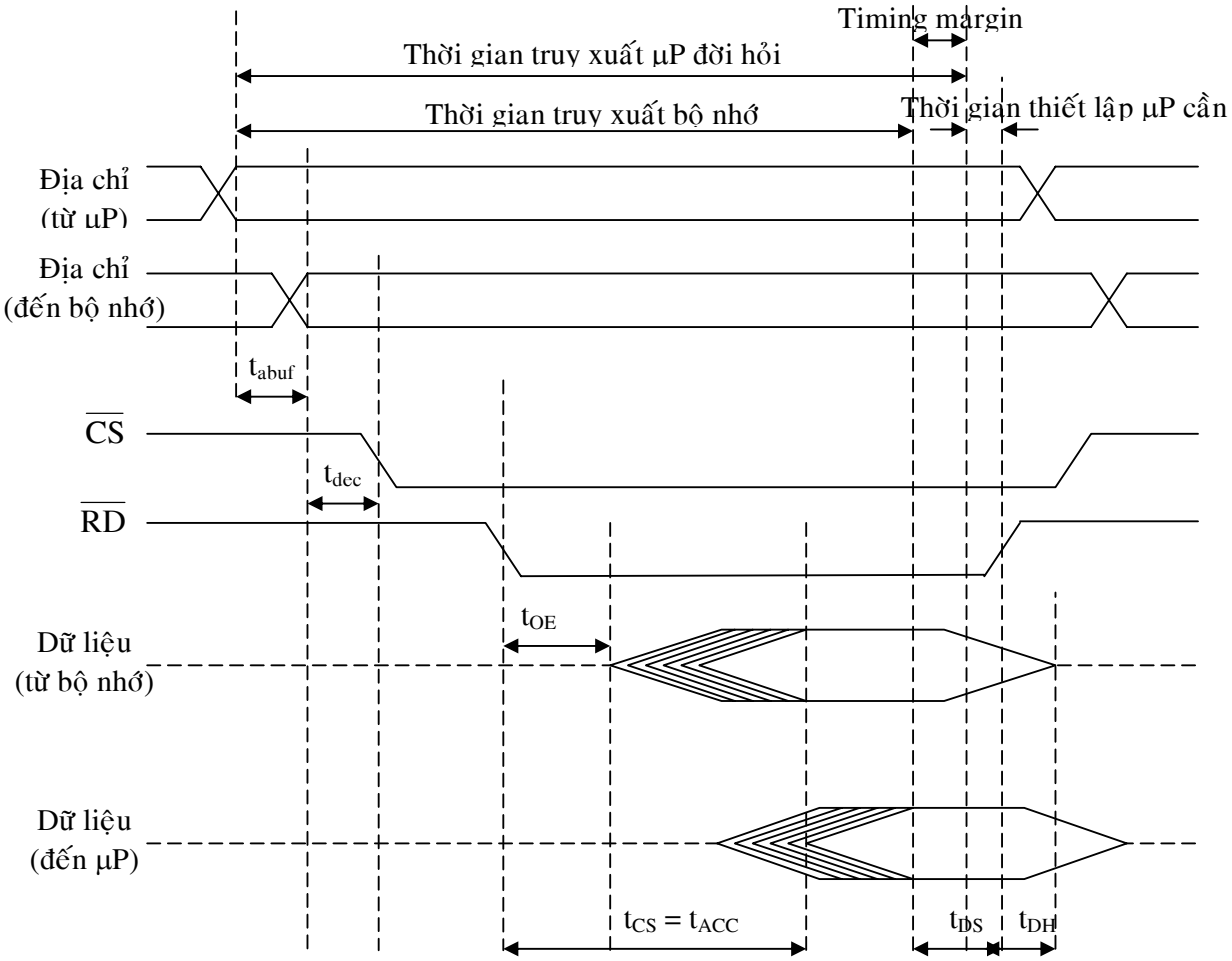
t_{dbuf}: thời gian trì hoãn ở bộ đệm dữ liệu (data buffer)
t_{abuf}: thời gian trì hoãn ở bộ đệm địa chỉ (address buffer)
t_{OE}: thời gian đáp ứng của bộ nhớ với tín hiệu cho phép ngõ ra (ouput enable)
t_{CS}: thời gian bộ nhớ truy xuất từ Chip Select
t_{ACC}: thời gian bộ nhớ truy xuất từ địa chỉ, thông thường tACC = tcs
t_{dec}: thời gian trì hoãn ở bộ giải mã (decoder)

❖ **Định thì đọc bộ nhớ:**

Thời gian truy xuất tổng cộng của hệ thống bộ nhớ chính là tổng thời gian trì hoãn trong các bộ đệm và thời gian truy xuất (access time) bộ nhớ.

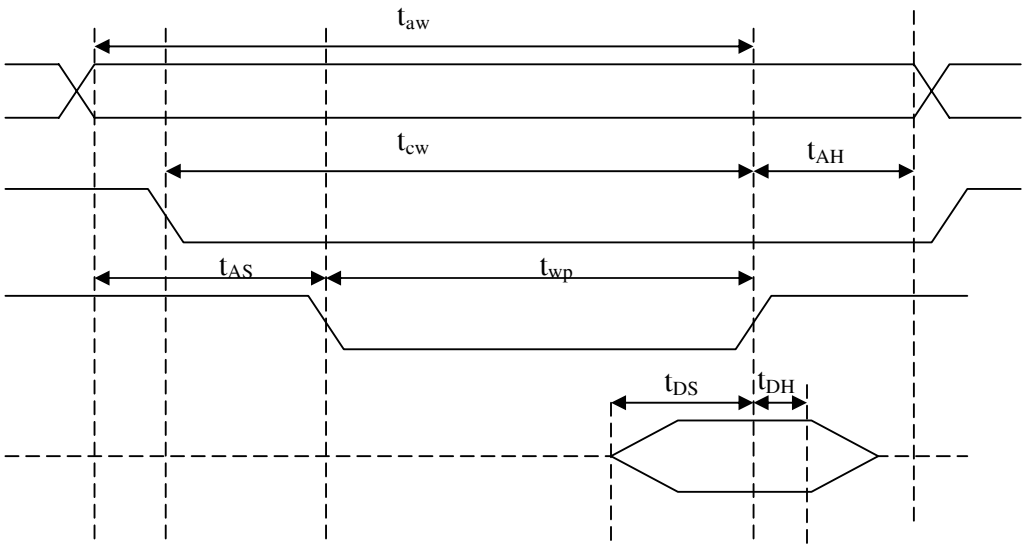
Hiệu giữa thời gian truy xuất cần thiết bởi µP với thời gian truy xuất thật sự của bộ nhớ gọi là biên định thì (timing margin).

t_{DS} (Data Setup): thời gian thiết lập dữ liệu cung cấp bởi hệ thống bộ nhớ
t_{DH} (Data Hold): thời gian giữ dữ liệu cung cấp bởi hệ thống bộ nhớ



Hình 1.18 – Định thì đọc bộ nhớ

❖ **Định thì ghi bộ nhớ:**



Hình 1.19 – Định thì ghi bộ nhớ

t_{aw}: thời gian truy xuất ghi (access write)

t_{wp}: độ rộng xung ghi tối thiểu (write pulse)

t_{AS}: thời gian địa chỉ hợp lệ trước khi $\overline{WR} = 0$

Thông thường, ta không quan tâm đến địa chỉ cho đến khi xác nhận $\overline{CS}$ nên thường $t_{cw} = t_{aw}$.