TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

KIẾN TRÚC MÁY TÍNH HỢP NGỮ

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 6/2017

Chương 1. HỆ THỐNG SỐ VÀ MẠCH LOGIC

1. Hệ thống số đếm

1.1. Hệ nhị phân (binary) – hệ đếm cơ số 2

- Tập số cơ sở: 0, 1
- Qui tắc đếm: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111…
- Mỗi một chữ số được gọi là một bit (binary digit). Bit ngoài cùng bên trái là bit có trọng số lớn nhất (MSB – Most Significant Bit), bit ngoài cùng bên phải là bit có trọng số nhỏ nhất (LSB – Least Significant Bit)

Ví dụ:

	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	
MSB	1	0	1	0	.	1	1	LSB

1.2. Hệ bát phân (octal) – hệ đếm cơ số 8

- Tập số cơ sở: 0, 1, 2, 3, 4, 5, 6, 7
- Qui tắc đếm: 0_8, 1_8, 2_8, 3_8, 4_8, 5_8, 6_8, 7_8, 10_8, 11_8, …, 17_8, 20_8, 21_8…

1.3. Hệ thập phân (decimal) – hệ đếm cơ số 10

- Tập số cơ sở: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Qui tắc đếm: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11…

1.4. Hệ thập lục phân (hexadecimal) – hệ đếm cơ số 16

- Tập số cơ sở: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Qui tắc đếm: 0_{16}, 1_{16}, 2_{16}, …, F_{16}, 10_{16}, 11_{16}, …, 1F16, 20_{16}, 21_{16}, … $2F_{16}$, 30_{16}…

DEC	HEX	BIN
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111

DEC	HEX	BIN
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

2. Chuyển đổi số giữa các hệ

2.1. Chuyển từ hệ nhị phân (b) hệ thập phân (d) và ngược lại

Ví dụ 1: $\overset{3\,2\,1\,0}{1001}_2 = 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 = 9_{10}$

Ví dụ 2: $9_{10} = 1001_2$

$9_{10} \div 2 = 4$ dư 1
$4 \div 2 = 2$ dư 0
$2 \div 2 = 1$ dư 0
$1 \div 2 = 0$ dư 1 ↑

Ví dụ 3: $13.125_{10} = 1101.101_2$

- Phần nguyên: $13_{10} = 1101_2$

- Phần thập phân: $0.125_{10} = 0.101_2$

$0.625 \times 2 = 1.25$ lấy số 1, phần thập phân 0.25

$0.25 \times 2 = 0.5$ lấy số 0, phần thập phân 0.5

$0.5 \times 2 = 1$ lấy số 1, phần thập phân 0 $\Rightarrow$ Kết thúc

2.2. Chuyển từ hệ thập lục phân (h) sang hệ nhị phân (b) và ngược lại

Ví dụ: $1E4A_{16} = 0001111001001010$ (1 E 4 A)

2.3. Chuyển từ hệ thập lục phân (h) sang thập phân (d) và ngược lại

Ví dụ 1: $1E4A.6B_{16} = 1\times16^3 + 14\times16^2 + 4\times16 + 10 + 6\times16^{-1} + 11\times16^{-2} = 7754.41796875_{10}$ (mũ: 3 2 1 0 −1 −2)

Ví dụ 2: $9853_{10} = 267D_{16}$

$9853 \div 16 = 615$ dư 13 (D)

$615 \div 16 = 38$ dư 7

$38 \div 16 = 2$ dư 6

$2 \div 16 = 0$ dư 2

3. Biểu diễn số nguyên không dấu (unsigned number)

- Biểu diễn các số dương.
- Tất cả các bit đều đều được sử dụng để biểu diễn giá trị.
- Số nguyên không dấu 1 byte lớn nhất là $1111\ 1111_2 = 255_{10}$ $(2^8 - 1)$
- Số nguyên không dấu 1 word lớn nhất là $1111\ 1111\ 1111\ 1111_2 = 65535_{10}$ $(2^{16} - 1)$

3.1. Phép cộng

Qui tắc: **0 + 0 = 0**; **1 + 0 = 1**

1 + 1 = 10 (ghi 0 nhớ 1); **1 + 1 + 1 = 11 (ghi 1 nhớ 1)**

Ví dụ:

```
 1 1111 1          11 1
 01110100 (116)    11.011  (3.375)
 10111101 (189)    10.110  (2.75)
-----------       ---------
100110001 (305)   100.001  (6.125)
```

3.2. Phép trừ

Qui tắc: **0 – 0 = 0**; **1 – 0 = 1**

1 – 1 = 0; **0 – 1 = 1 (nợ 1)**

Ví dụ:

```
  11          1  1  1
 11010110 (214)   100.001  (6.125)
  1110100 (116)    10.110  (2.75)
----------       --------
 01100010 (98)    11.011  (6.125)
```

3.3. Phép nhân

Qui tắc: **0 × 0 = 0**; **1 × 0 = 1**; **1 × 1 = 1**

Ví dụ:

```
  10001    (17)
    110    (6)
  -----
  00000
 10001
10001
-------
1100110    (102)
```

4. Biểu diễn số có dấu (signed number)

- Biểu diễn các số dương hoặc số âm.

4.1. Lượng dấu

- Bit trái nhất (MSB): Biểu diễn dấu (1 – âm, 0 – dương)
- Các bit còn lại: Biểu diễn độ lớn của số (giá trị tuyệt đối của số)
- Số 0 có 2 biểu diễn: 000000 (+0) và 100000 (–0)

4.2. Số bù 1 (one's complement)

- Bit trái nhất (MSB): Biểu diễn dấu (0 – dương, 1 – âm)
- Các bit còn lại: Biểu diễn độ lớn của số. Nếu là số âm thì đảo tất cả các bit (0 thành 1, 1 thành 0)

Ví dụ: Số bù 1 của **0**0101011 (+43) là **1**1010100 (−43)

- Số 0 có 2 biểu diễn: **0**0000000 (+0) và **1**1111111 (−0)
- Khi thực hiện phép cộng, cũng thực hiện bình thường, tuy nhiên, nếu còn phát sinh bit nhớ thì phải cộng bit nhớ này vào kết quả vừa thu được.

Ví dụ:

```
 1111                    1
 001101   (13)            110010   (–13)
 001011   (11)            110100   (–11)
------                  -------
 011000   (24)           1100110
                              1
                        -------
                         100111   (–24)
```

4.3. Số bù 2 (two's complement)

- Khắc phục vấn đề số 0 có 2 biểu diễn khác nhau.
- Biểu diễn số bù 1, sau đó cộng thêm 1 vào kết quả.

Ví dụ: Số 12 (8 bit): 0000 1100 ($12 = 2^3 + 2^2$)

- Số bù 1 của 12: 1111 0011

```
                1
           ------
```

- Số bù 2 của 12: 1111 0100 ($-12 = -2^8 + 2^7 + 2^6 + 2^5 + 2^4 + 2^2$)

- Phép cộng số bù 2:

Ví dụ:

```
    1                  111  1                 1111 1
 00001100   (12)     11110100   (–12)      11110100   (–12)
 00001001   (9)      11110111   (–9)       00001100   (+12)
--------            --------              --------
 00010101   (21)    X11101011   (–21)     X00000000
```

Nhận xét: Số bù 2 của một số x cộng với x cho kết quả là dãy toàn bit 0 (bỏ bit 1 cao nhất do vượt quá phạm vi lưu trữ)

5. Phép dịch bit và phép xoay

- Dịch trái (shift left): Dời tất cả các bit sang trái 1 vị trí, bỏ bit trái nhất, thêm 0 ở bit phải nhất.

Ví dụ: 1100 1010 ⇒ 1001 0100

- Dịch phải (shift right): Dời tất cả các bit sang phải 1 vị trí, bỏ bít phải nhất, thêm 0 ở bit trái nhất

Ví dụ: 1001 0101 ⇒ 0100 1010

- Xoay trái (rotate left): Dời tất cả các bit sang trái, bit trái nhất thành bit phải nhất.

Ví dụ: 1100 1010 ⇒ 1001 0101

- Xoay phải (rotate right): Dời tất cả các bit sang phải, bit phải nhất thành bit trái nhất.

Ví dụ: 1001 0101 ⇒ 1100 1010

6. Mã BCD (Binary Coded Decimal)

- Dùng 4 bit để mã hóa các chữ số từ 0 đến 9: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001
- Các tổ hợp còn lại không dùng (tổ hợp cấm): 1010, 1011, 1100, 1101, 1110, 1111

Ví dụ: $61_{10} = 0110\ 0001_{(BCD)}$; $35.61_{10} = 0011\ 0101.0110\ 0001_{(BCD)}$

7. Biểu diễn số có dấu chấm động (floating point number)

Một số thực X được biểu diễn như sau: $X = (-1)^S \times M \times R^E$

- S là bit dấu (1 – âm, 0 – dương)
- M là phần định trị (mantissa)
- R là cơ số (radix)
- E là số mũ (exponent)

Ví dụ: $2017 = (-1)^0 \times 2.017 \times 10^3$

7.1. Số chấm động chính xác đơn (single precision) (32 bit)

31	30 … 23	22 … 0
S	e	m
1 bit	8 bit (e = E + 127)	23 bit (M = 1.m)

$X = (-1)^S \times 1.m \times 2^{e-127}$

Ví dụ 1: $X = -5.25_{10}$

⇨ Đổi sang hệ nhị phân: $X = -5.25_{10} = -101.01_2$

⇨ Đưa về dạng $1.m \times 2^E$: $X = -1.0101 \times 2^2$

⇨ Số âm ⇒ S = 1

⇨ $E = 2 \Rightarrow e = E + 127 = 129_{10} = 1000\ 0001_2$

⇨ m = 01010000000000000000000 (thêm 19 số 0 để đủ 23 bit)

Kết quả: X = $\underset{S}{1}\underbrace{10000001}_{e}\underbrace{01010000000000000000000}_{m}$

Ví dụ 2: X = $0\underbrace{10000010}_{e}\underbrace{10101100000000000000000}_{m}$ (S)

⇨ S = 0 ⇒ Số dương

⇨ e = $10000010_2 = 130_{10}$ ⇒ E = e – 127 = 3

⇨ m = 10101100000000000000000 = 101011

⇨ X = $(-1)^0 \times 1.101011 \times 2^3 = 1101.011 = 13.375_{10}$

7.2. Số chấm động chính xác kép (double precision) (64 bit)

63	62 … 52	51 … 0
S	e	m
1 bit	11 bit (e = E + 1023)	52 bit (M = 1.m)

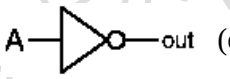
$X = (-1)^S \times 1.m \times 2^{e-1023}$

8. Cổng logic và phép tính

8.1. Các phép tính logic

X	Y	X + Y	X.Y	$\overline{X}$	$X \oplus Y$
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	0	1
1	1	1	1	0	0

8.2. Cổng NOT (cổng đảo)

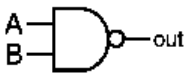


$(out = \overline{A})$

8.3. Cổng AND

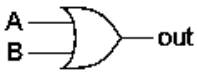
$(out = AB)$ (đầu ra chỉ bằng 1 khi tất cả đầu vào bằng 1)

8.4. Cổng NAND



$(out = \overline{AB})$ (đầu ra chỉ bằng 0 khi tất cả đầu vào bằng 1)

8.5. Cổng OR

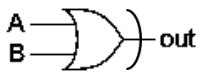


$(out = A + B)$ (đầu ra chỉ bằng 0 khi tất cả đầu vào bằng 0)

8.6. Cổng NOR

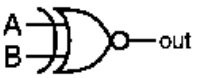
$(out = \overline{A + B})$ (đầu ra chỉ bằng 1 khi tất cả đầu vào bằng 0)

8.7. Cổng EXOR



$(out = A \oplus B = \overline{A}B + A\overline{B})$ (đầu ra chỉ bằng 0 khi tất cả đầu vào giống nhau)

8.8. Cổng EXNOR



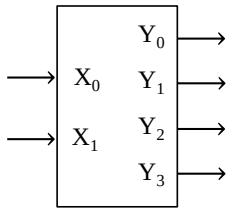
$(out = \overline{A \oplus B} = AB + \overline{A}\overline{B})$ (đầu ra chỉ bằng 1 khi tất cả đầu vào giống nhau)

9. Mạch mã hóa và mạch giải mã

9.1. Mạch mã hoá (Encoder)

9.2. Mạch giải mã (Decoder)

- Mạch giải mã 2 sang 4 (2 ngõ vào, 4 ngõ ra), ngõ ra tích cực cao.

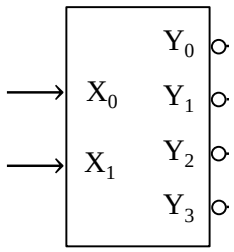


Bảng sự thật:

X_1	X_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$Y_0 = \overline{X_1}\,\overline{X_0}$; $Y_1 = \overline{X_1}X_0$; $Y_2 = X_1\overline{X_0}$; $Y_3 = X_1X_0$

- Mạch giải mã 2 sang 4, ngõ ra tích cực thấp.

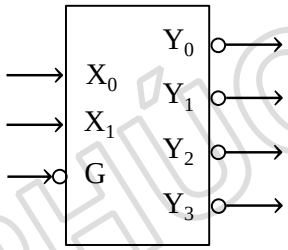


Bảng sự thật:

X_1	X_0	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	Y_0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

$\overline{Y_0} = X_1 + X_0 = \overline{\overline{X_1}\,\overline{X_0}}$; $\overline{Y_1} = \overline{\overline{X_1}X_0}$; $\overline{Y_2} = \overline{X_1\overline{X_0}}$; $\overline{Y_3} = \overline{X_1X_0}$

- Mạch giải mã 2 sang 4, ngõ ra tích cực thấp, 1 vào điều khiển tích cực thấp

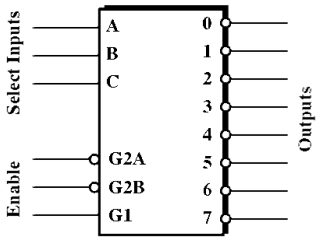


Bảng sự thật:

G	X_1	X_0	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	Y_0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	x	x	1	1	1	1

$\overline{Y_0} = \overline{G} + X_1 + X_0 = \overline{\overline{G}\,\overline{X_1}\,\overline{X_0}}$; $Y_1 = \overline{\overline{G}\,\overline{X_1}X_0}$; $Y_2 = \overline{\overline{G}X_1\overline{X_0}}$; $Y_2 = \overline{\overline{G}X_1X_0}$

- Mạch giải mã 3 sang 8 (74LS138)



G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	$\overline{Y7}$	$\overline{Y6}$	$\overline{Y5}$	$\overline{Y4}$	$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$
0	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	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

10. Mạch ghép kênh và phân kênh

10.1. Mạch ghép kênh (Multiplexer – MUX)

- Có 2^n ngõ vào, có 1 ngõ ra, có n ngõ điều khiển. Có thể có ngõ vào CS
- Tại mỗi thời điểm, chỉ 1 ngõ vào được nối tới ngõ ra, tổ hợp các ngõ điều khiển xác định ngõ vào nào được nối tới ngõ ra.

10.2. Mạch phân kệnh (DeMultiPlexer – DEMUX)

- Có 1 ngõ vào, có n ngõ vào điều khiển, có 2n ngõ ra.
- Tại mỗi thời điểm chỉ có một ngõ ra được nối với ngõ vào, tổ hợp các ngõ vào điều khiển quyết định ngõ ra nào được nối tới ngõ vào.

Chương 2. HỆ THỐNG MÁY TÍNH VÀ VI XỬ LÝ

1. Giới thiệu về vi xử lý

1.1. Vi xử lý (microprocessor)

Là một vi mạch kiểu VLSI (Very Large Scale Integrated Circuit), có thể lập trình được.

1.2. Chức năng

- Tính toán và vận chuyển dữ liệu.
- Vi xử lý thực hiện các chức năng thông qua việc thực hiện các lệnh
- Thực hiện các phép toán (số học, logic, dịch, quay…)
- Kết nối và trao đổi dữ liệu với các thiết bị bên ngoài thông qua các cổng vào ra.

1.3. Phân loại vi xử lý dựa trên chức năng

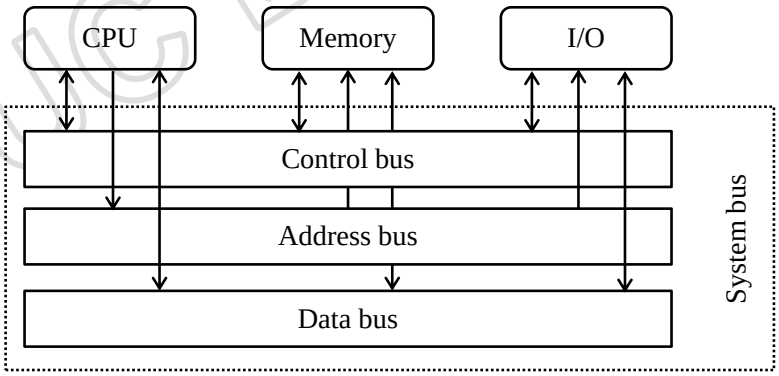
- Vi xử lý đa chức năng (General Purpose Microprocessor): Chứa tất cả các thành phần phục vụ tính toán và điều khiển, không bao gồm bộ nhớ và các cổng vào ra.
- Vi điều khiển (Microcontroller): Chứa tất cả các thành phần phục vụ tính toán và điều khiển, bao gồm bộ nhớ và các cổng vào ra. Tất cả các thành phần của vi điều khiển được tích hợp trên 1 chip đơn.

1.4. Hệ vi xử lý

Bao gồm:

- CPU (Central Processing Unit): Đơn vị xử lý trung tâm, có nhiệm vụ tính toán và điều khiển.
- Bộ nhớ (memory) lưu dữ liệu (data) và lệnh (instruction) cho CPU xử lý.
- Các thiết bị nhập, xuất (Inputs, Outputs)

1.5. Kiến trúc Von Neumann



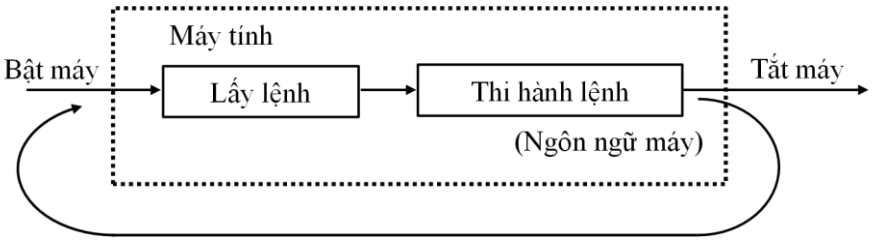
- Một bộ nhớ duy nhất được dùng để lưu dữ liệu và lệnh.
- Dữ liệu và lệnh được lưu trữ trong các phần riêng của bộ nhớ.
- Bộ nhớ được đánh địa chỉ theo vùng, không phụ thuộc vào loại dữ liệu mà nó lưu trữ.
- Quá trình thực hiện các lệnh diễn ra tuần tự.

1.6. Kiến trúc Harvard

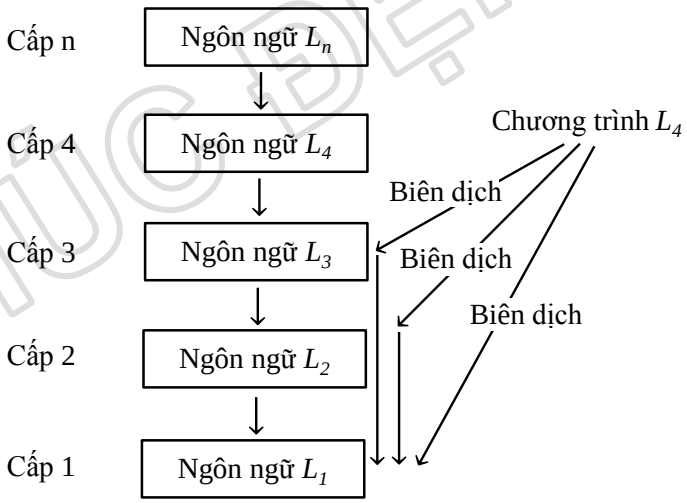
- Bộ nhớ được chia thành 2 phần riêng: Bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu.
- CPU sử dụng 2 hệ thống bus để giao tiếp với bộ nhớ: Hệ thống bus giao tiếp với bộ nhớ lưu chương trình và hệ thống bus giao tiếp với bộ nhớ lưu dữ liệu.
- Kiến trúc Harvard phức tạp hơn, nhanh hơn kiến trúc Von Neumann do CPU có thể giao tiếp đồng thời với cả bộ nhớ chương trình và dữ liệu, thích hợp với cơ chế đường ống và xử lý song song.

1.7. Mô hình máy tính nhiều cấp

- Máy tính đơn giản:



- Chương trình (program): Tập các lệnh theo một trình tự mô tả cách thực hiện công việc.
- Ngôn ngữ máy (machine language): Tập các lệnh (dạng nhị phân) mà các mạch điện tử trong máy tính có thể thực hiện trực tiếp (L_1). Tất cả các chương trình trước khi thực hiện phải chuyển qua ngôn ngữ máy.
- Ngôn ngữ L_2: Thân thiện và dễ dùng hơn.
- Chương trình $L_2 \rightarrow$ Chương trình L_1
- Cách 1: Biên dịch – Compile. Chương trình L_2 = Chương trình L_1
- Cách 2: Phiên dịch – Interpreter. Tạo ra Interpreter bằng L_1. Thực hiện trực tiếp từng lệnh L_2 (biến đổi thành các lệnh trong L_1).
- Ngôn ngữ L_n: L_2 tiện lợi hơn L_1, nhưng vẫn khó cho người dùng $\rightarrow$ Tạo ra L_n tiện lợi hơn L_{n-1}. Chương trình viết bằng L_n trước khi thực hiện phải chuyển sang L_{n-1}
- Mô hình tổng quát máy tính nhiều cấp



2. Hệ thống máy tính cá nhân dùng vi xử lý

Các máy tính từ lúc ra đời cho đến nay đều được chế tạo, cải tiến dựa trên mô hình Von Neumann.

2.1. Chức năng các khối

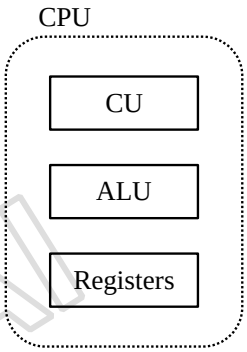
- CPU: Thi hành lệnh.
- I/O Device: Liên lạc với bên ngoài.
- Bộ nhớ: Lưu dữ liệu và chương trình.
- Bus: Truyền địa chỉ, dữ liệu, thông tin điều khiển giữa vi xử lý, bộ nhớ và I/O. CPU điều khiển bộ nhớ và I/O thông qua 3 bus: địa chỉ, dữ liệu và điều khiển/trạng thái.

2.2. Bộ nhớ

- Bộ nhớ gồm nhiều ô nhớ, mỗi ô nhớ có 1 địa chỉ.
- CPU truy cập 1 ô nhớ thông qua bus địa chỉ.
- Độ rộng bus địa chỉ quyết định 1 CPU có thể truy cập bao nhiêu ô nhớ.
- Không gian địa chỉ bộ nhớ: Số ô nhớ về mặt lý thuyết (tính theo byte) mà CPU có thể quản lý.

2.3. I/O

- Mỗi thiết bị I/O có 1 địa chỉ gọi là port.
- CPU truy cập 1 port thông qua bus địa chỉ.
- Không gian I/O của hệ thống: Số port về mặt lý thuyết mà CPU có thể quản lý.



2.4. CPU

- Lệnh CPU thi hành được nạp trước đó vào trong bộ nhớ.
- Các lệnh nằm liên tục trong bộ nhớ tạo thành chương trình.
- CPU là một hệ thống số tuần tự, đồng bộ nên việc cung cấp xung đồng hồ (clock) là cần thiết.
- CPU hoạt động được với xung clock có tần số càng cao thì chạy càng nhanh.
- Thành phần: Khối điều khiển – CU (Control Unit), khối tính toán – ALU (Arithmetic Logic Unit), các thanh ghi (Registers)

2.5. Bus

- Bus là các dây nối các thành phần trong một hệ thống máy tính.
- Bus được dùng để truyền địa chỉ, dữ liệu, điều khiển giữa vi xử lý, bộ nhớ và các thiết bị I/O

3. Các dạng dữ liệu

3.1. Dữ liệu BCD

Được lưu ở 1 trong 2 dạng: Nén hoặc không nén.

Ví dụ:

Thập phân	BCD nén	BCD không nén
12	0001 0010	0000 0001 0000 0010
623	0000 0110 0010 0011	0000 0110 0000 0010 0000 0011

3.2. Dữ liệu kích thước byte và word: Lưu các số nguyên không dấu hoặc có dấu.

3.3. Dữ liệu kích thước từ: Một từ được tạo thành từ 2 byte dữ liệu.

3.4. Little endian và Big endian

- Little endian: Phần thấp được lưu ở đc thấp, phần cao được lưu ở địa chỉ cao
- Big endian: Ngược lại

4. Giải phẫu máy tính điện tử

Chương 3. KIẾN TRÚC PHẦN MỀM

1. Sơ đồ khối của CPU 8086/8088

- Đơn vị giao tiếp Bus – BIU (Bus Interface Unit): Lấy lệnh, đọc toán hạng, lưu kết quả, truy xuất cache.
- Đơn vị thực thi – EU (Execution unit): Thi hành lệnh, định thì, kiểm tra các tín hiệu trạng thái, tín hiệu yêu cầu ngắt, DMA, Ready
- Hoạt động của BIU và EU độc lập.
- Chu kì thi hành lệnh được thực hiện theo cơ chế đường ống (pipeline).

2. Qui trình thi hành lệnh

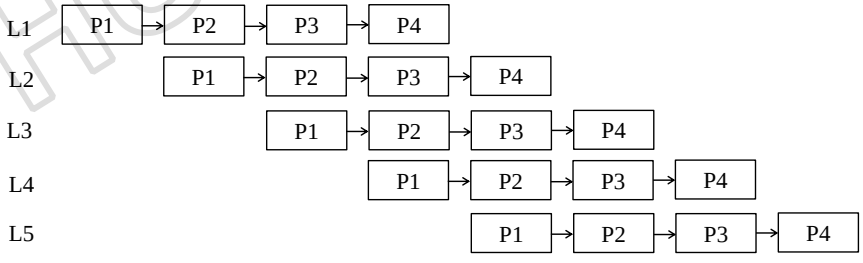
2.1. Các bước thi hành lệnh

- Nạp lệnh từ bộ nhớ vào thanh ghi lệnh IR
- Tăng thanh ghi IP (PC) để chỉ tới lệnh tiếp theo
- Giải mã lệnh
- Định vị toán hạng được dùng bởi lệnh
- Nạp toán hạng từ bộ nhớ (nếu cần)
- Thi hành lệnh
- Lưu kết quả và trạng thái (PSW)
- Trở lại bước 1 để nạp lệnh tiếp theo.

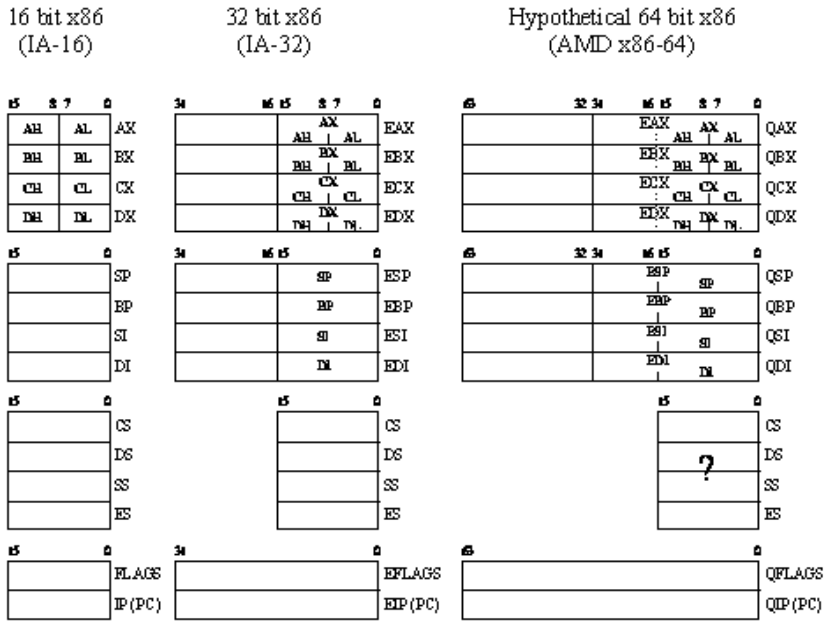
2.2. Thi hành lệnh song song

2.3. Giải pháp tăng tốc độ xử lý

- Giải pháp nhiều ALU hoặc nhiều CPU ⇒ Máy tính song song.
- Giải pháp pipeline (cơ chế đường ống): Chu kì thi hành lệnh của CPU được chia thành nhiều trạng thái độc lập.



3. Tổ chức các thanh ghi



3.1. Thanh ghi đa dụng (General Registers)

Gồm các thanh ghi 16 bit (8086):

- AX (Accumulator Registers): Lưu trữ dữ liệu, phép toán số học, logic, lệnh ngắt, lệnh xuất nhập.
- BX (Base Registers): Lưu trữ dữ liệu, phép toán số học, logic, con trỏ địa chỉ gián tiếp.
- CX (Count Registers): Lưu trữ dữ liệu, phép toán số học, logic, bộ (biến) đếm trong các vòng lặp (giảm về 0)
- DX (Data Registers): Lưu trữ dữ liệu, phép toán số học, logic, chứa dữ liệu tạm thời trong các phép nhân, chia, chứa địa chỉ port trong lệnh xuất nhập.
- Các thanh ghi AX, BX, CX, DX còn được sử dụng như các thanh ghi 8 bit độc lập: AH, AL, BH, BL, CH, CL, DH, DL (H – 8 bit cao, L – 8 bit thấp).

32 bit (386): EAX, EBX, ECX, EDX

3.2. Thanh ghi chỉ số (Index Registers)

Gồm các thanh ghi 16 bits (8086):

- SI (Source Index): chỉ số nguồn
- DI (Destination Index): chỉ số đích

32 bits (386): ESI, EDI

3.3. Thanh ghi con trỏ (Pointer Registers)

Gồm các thanh ghi 16 bits (8086):

- IP (Instruction Pointer): Thanh ghi con trỏ lệnh
- BP (Base Pointer): Thanh ghi con trỏ cơ sở
- SP (Stack Pointer): Con trỏ ngăn xếp.

32 bits (386): ESP, EBP

3.4. Thanh ghi phân đoạn (Segment Registers)

Gồm các thanh ghi 16 bit:

- CS (Code Segment): Thanh ghi đoạn lệnh
- DS (Data Segment): Thanh ghi đoạn dữ liệu
- ES (Extra Segment): Thanh ghi đoạn dữ liệu phụ
- SS (Stack Segment): Thanh ghi đoạn ngăn xếp

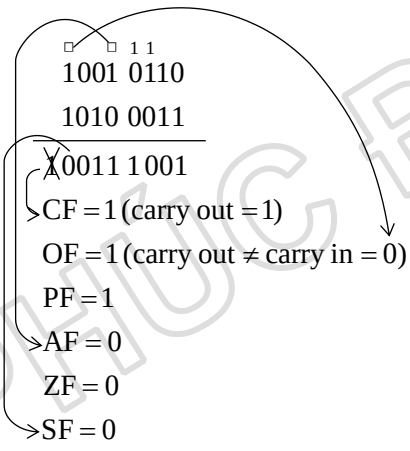
3.5. Thanh ghi cờ (flag register)

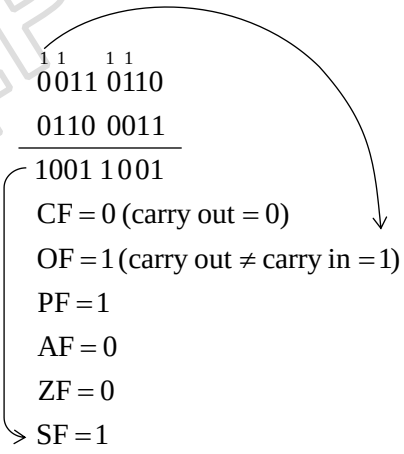
Chỉ có 9 trong 16 bit được sử dụng:

				O	D	I	T	S	Z		A		P		C

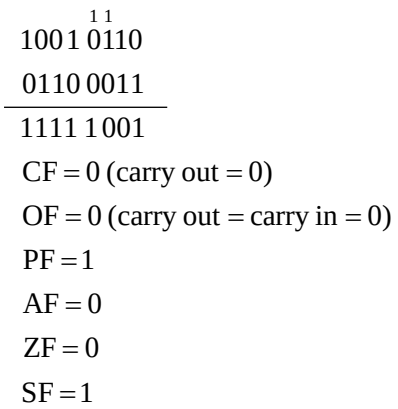
- CF (carry flag – cờ nhớ): CF = 1 khi kết quả có nhớ ra khỏi bit cao nhất (MSB)
- PF (parity flag – cờ chẵn lẻ): PF = 1 khi tổng số bit 1 trong kết quả là số chẵn
- AF (auxliary carry flag – for BCD – cờ nhớ phụ): AF = 1 khi có nhớ 4 bit thấp sang 4 bit cao
- ZF (zero flag – cờ rỗng): ZF = 1 khi kết quả bằng 0
- SF (sing flag – cờ dấu): SF = 1 khi kết quả âm
- TF (trap flag – cờ bẫy): TF = 1 khi CPU ở chế độ chạy từng lệnh (dùng tìm lỗi trong một chương trình)
- IF (interrupt enable flag – cờ ngắt)
- DF (direction flag – cờ hướng)
- OF (overflow flag – cờ tràn): OF =1 khi kết quả phép tính có dấu bị sai

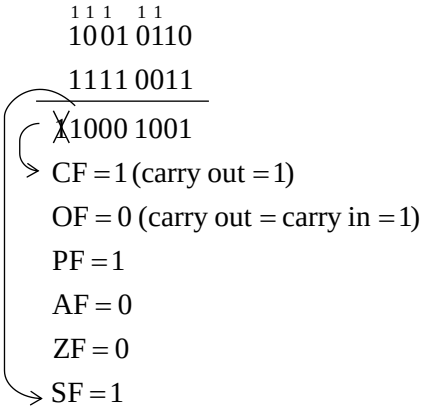
Ví dụ 1:





Ví dụ 2:





Ví dụ 3:

```
  1 1 1 1  1
  1001 1100
  0110 0100
 ------------
 (1)0000 0000
```

CF = 1
OF = 1
PF = 1
AF = 1
ZF = 1
SF = 0

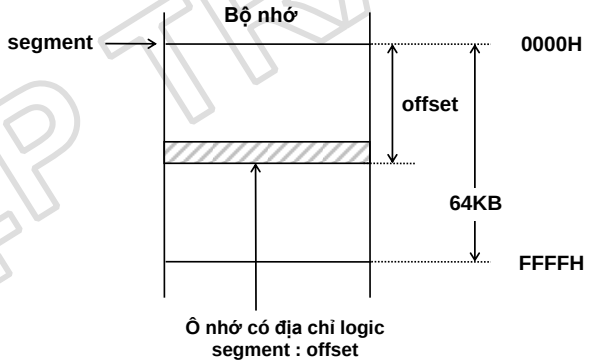
```
  1 1 1 1  1 1 1
  1011 0111
  1101 1001
 ------------
 (1)1001 0000
```

CF = 1
OF = 0
PF = 1
AF = 1
ZF = 0
SF = 1

4. Tổ chức bộ nhớ 8086/8088

4.1. Địa chỉ vật lý (physical address) (20 bit)

- Các ô nhớ trong không gian bộ nhớ vật lý được đánh số thứ tự 00000H – FFFFFH gọi là địa chỉ vật lý.
- Không gian địa chỉ vật lý chia thành nhiều khối 64KB gọi là segment. Địa chỉ segment chứa trong thanh ghi phân đoạn (CS, DS, SS, ES).
- Mỗi ô nhớ trong segment có 1 địa chỉ so với đầu segment, có giá trị từ 0000H – FFFFH gọi là địa chỉ offset. Địa chỉ offset chứa trong các thanh ghi đa dụng, chỉ số, con trỏ

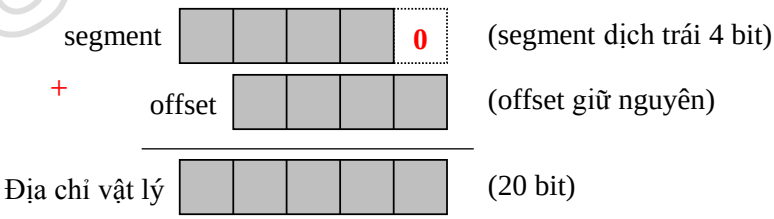


4.2. Địa chỉ logic (logical address)

Có dạng **`segment : offset`**

4.3. Quan hệ giữa địa chỉ vật lý và logic

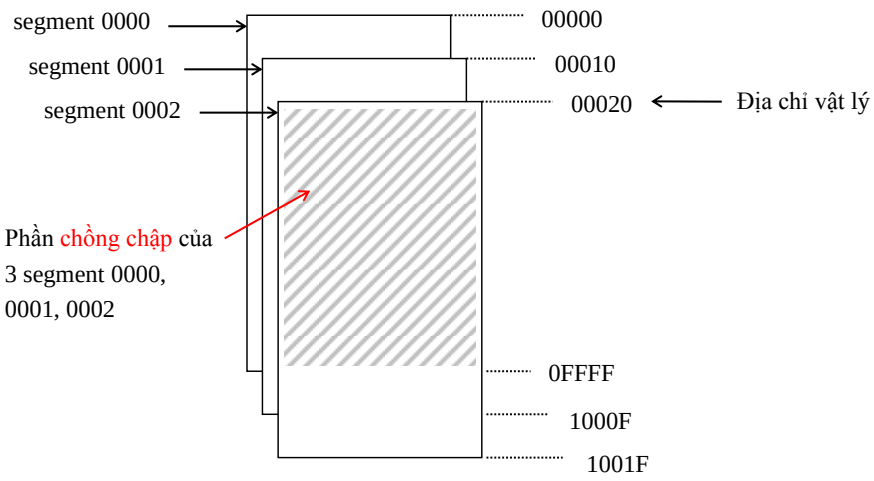
- Địa chỉ vật lý = segment × 16 + offset



Ví dụ 1: Cho địa chỉ logic 1ED2 : C200

```
  1ED20
+  C200
 -------
  2AF20 :  Địa chỉ vật lý
```

- Khi segment thay đổi 1 đơn vị (paragraph) thì địa chỉ vật lý thay đổi 16 byte.



- Một địa chỉ vật lý có thể được biểu diễn bằng nhiều địa chỉ logic khác nhau.

Ví dụ 2: Cho địa chỉ vật lý 2AF20. Tính địa chỉ logic trong trường hợp:

- Phân đoạn là 1ED2:

Offset = 2AF20 – 1ED2**0** = C200

⇒ Địa chỉ logic là 1ED2 : C200

- Phân đoạn là 2012:

Offset = 2AF20 – 2012**0** = AE00

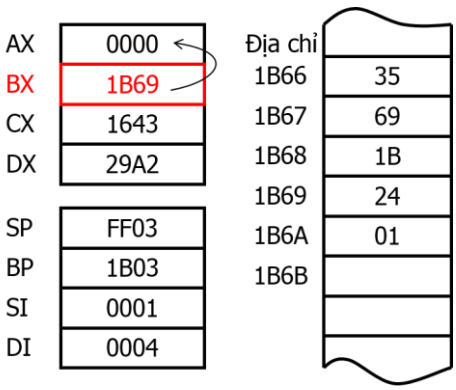
⇒ Địa chỉ logic là 2012 : AE00

5. Các chế độ định địa chỉ (addressing mode)

5.1. Định vị thanh ghi (register addressing): Lấy dữ liệu từ thanh ghi

Toán hạng là Tên thanh ghi. Địa chỉ chứa trong thanh ghi.

Ví dụ: MOV AX, **BX** ; Sao chép BX vào AX



5.2. Định vị tức thời

Địa chỉ nằm trực tiếp trong lệnh.

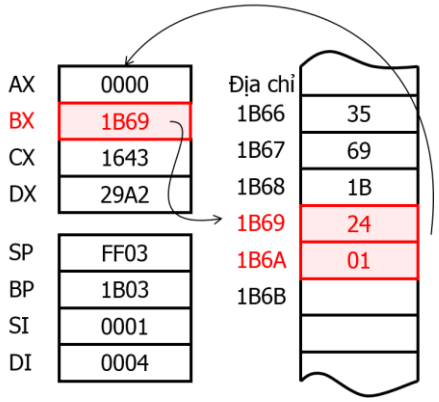
Ví dụ: MOV AX, 1B67h ; Sao chép 1B67h vào AX

5.3. Định vị gián tiếp thanh ghi (register indirect addressing): Lấy dữ liệu từ ô nhớ

Toán hạng được viết dưới dạng [reg] để xác định **ô nhớ** cần truy xuất.

Các thanh ghi có thể sử dụng là: **BX, SI, DI** (địa chỉ đoạn chứa trong DS), **BP** (địa chỉ đoạn chứa trong SS).

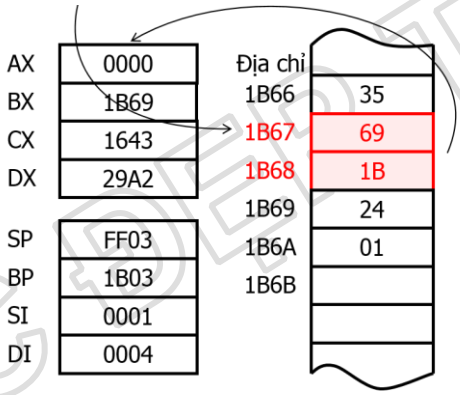
Ví dụ: MOV AX, **[BX]** ; Chép bộ nhớ tại địa chỉ chứa trong BX vào AX



5.4. Định vị trực tiếp ô nhớ (register memory addressing)

Toán hạng được viết dưới dạng [mem] để xác định **địa chỉ ô nhớ** cần truy xuất.

Ví dụ: MOV AX, **[1B67h]** ; Chép bộ nhớ tại địa chỉ 1B67h vào AX



5.5. Định vị chỉ số (indexed addressing)

Địa chỉ là tổng giá trị thanh ghi chỉ số SI và DI và một số bù 2 có dấu 8 hoặc 16 bits gọi là độ dời (offset).

Ví dụ: MOV AX, [SI – 2]

MOV [SI]4, AX

5.6. Định vị cơ sở (based addressing)

Tương tự định vị chỉ số, nhưng dùng thanh ghi cơ sở BX, BP thay vì SI, DI.

Ví dụ: MOV BX, [BX + 2]

5.7. Định vị cơ sở chỉ số (based indexed addressing)

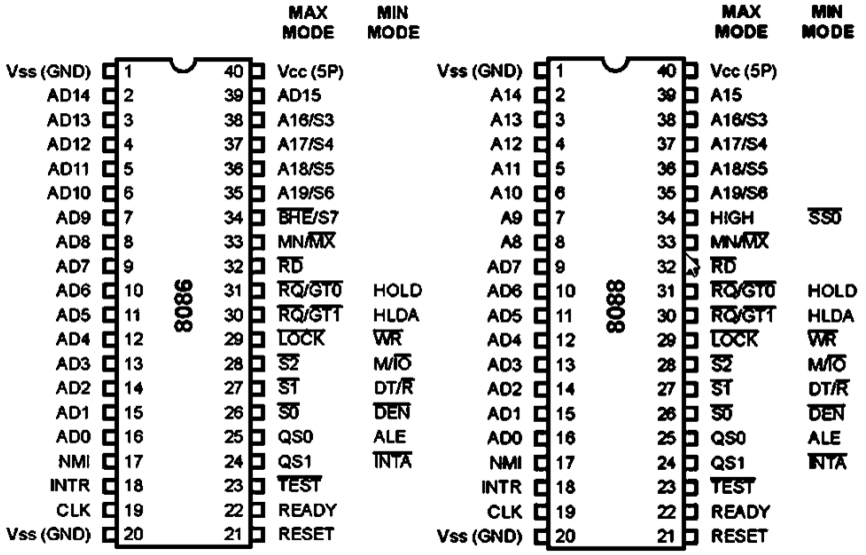
Tương tự định vị chỉ số, nhưng dùng thanh ghi cơ sở BX, BP thay vì SI, DI.

Ví dụ: MOV AX, [BX+DI]

MOV [BX+SI+2], AX

MOV AX, 2[BX][SI]

1. Sơ đồ chân 8086/8088



1.1. Mô tả chung

- Độ rộng bus dữ liệu bên trong 16 bit. Độ rộng bus dữ liệu bên ngoài 16 bit (8086); 8 bit (8088)
- Sơ đồ chân 8086, 8088 gần tương tự nhau (40 chân): 16 chân địa chỉ/data, 4 chân địa chỉ cao/trạng thái, 17 chân điều khiển/định thì, 3 chân nguồn, 2 chế độ hoạt động (max và min mode)
- Cùng 1 tập lệnh

1.2. Sơ đồ chân 8086 theo nhóm chức năng

- Các tín hiệu địa chỉ, dữ liệu: AD0 – AD15
- Các tín hiệu trạng thái: A16/S3 – A19/S6
- Các tín hiệu điều khiển: $\overline{BHE}$ / S7, $\overline{RD}$, READY, INTR, NMI, CLK, $\overline{TEST}$, MN / $\overline{MX}$
- Các tín hiệu Clock, nguồn: CLK, Vcc, GND
- Các tín hiệu ở chế độ MIN: ALE, $\overline{DEN}$, DR / $\overline{R}$, $\overline{HOLD}$, HLDA, $\overline{WR}$, $\overline{INTA}$, M / $\overline{IO}$
- Các tín hiệu ở chế độ MAX: $\overline{LOCK}$, $\overline{RQ}$ / $\overline{GT0}$, $\overline{RQ}$ / $\overline{GT1}$, QS0, QS1, S0, S1, S2

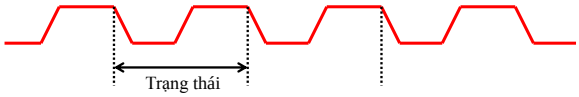
2. Tổng quan về cấu hình của hệ thống máy tính

2.1. Cấu hình ở chế độ MIN

2.2. Cấu hình ở chế độ MAX

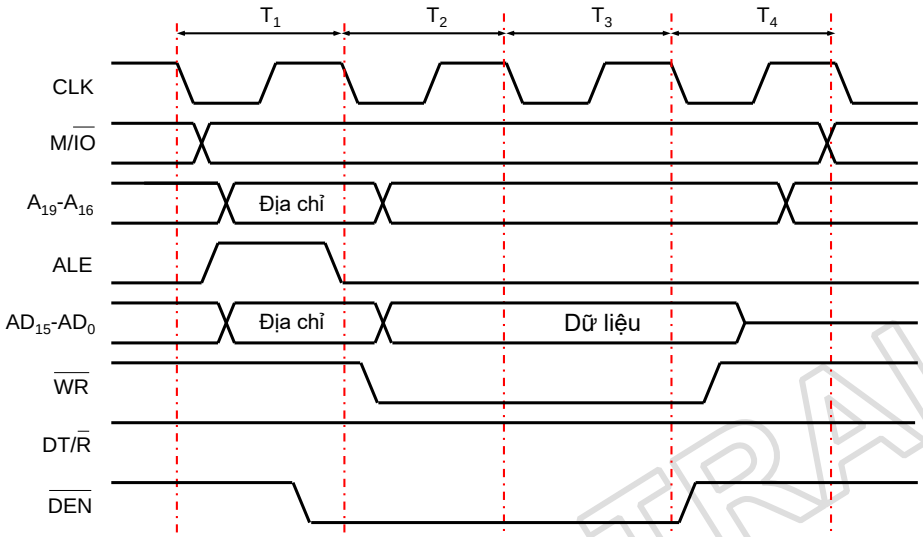
2.3. Mạch phát xung clock

- Tạo xung clock cho CPU, Ready, và Reset cho hệ thống
- Dạng xung clock: Tuần hoàn
- Mỗi chu kì của xung clock gọi là 1 trạng thái. Bắt đầu từ sườn âm của 1 xung và kết thúc ở sườn âm của xung tiếp theo.

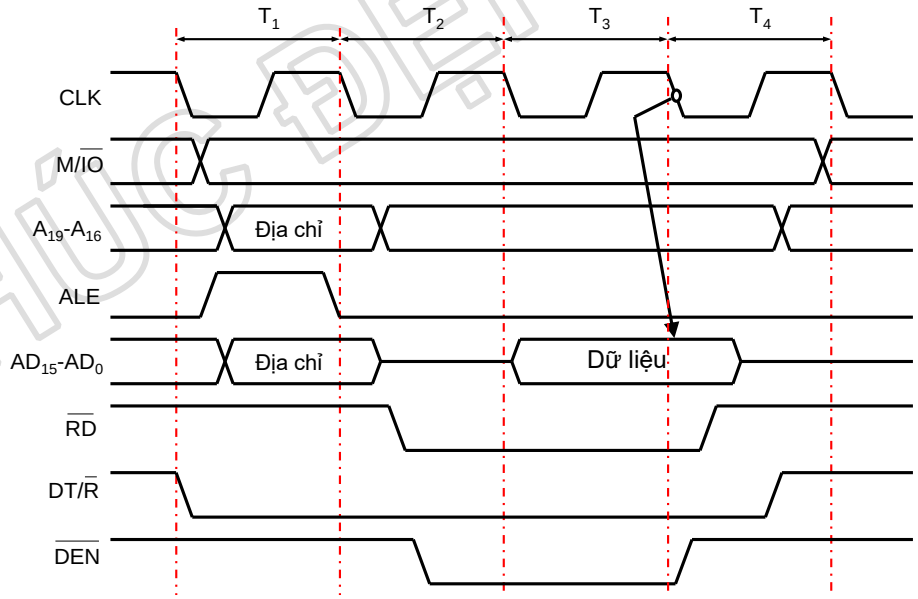


- Mỗi hoạt động cơ bản như đọc/ghi bộ nhớ hoặc I/O mất 1 khoảng thời gian được gọi là chu kì bus (chu kì máy). Mỗi chu kì bus kéo dài ít nhất 4 chu kì xung clock (4 trạng thái)
- Thực hiện một lệnh mất khoảng thời gian gọi là chu kì lệnh. Một chu kì lệnh có thể mất nhiều chu kì bus.

2.4. Chu kì đọc bộ nhớ



2.5. Chu kì ghi bộ nhớ

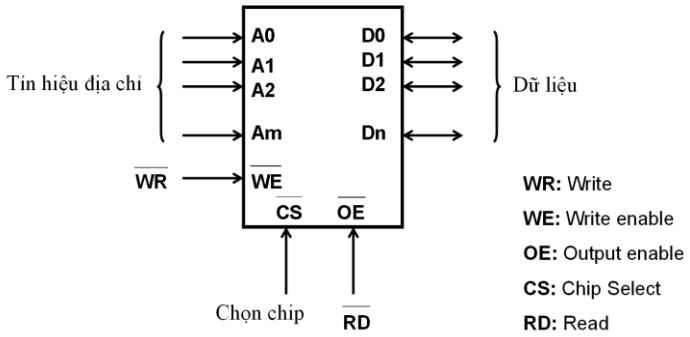


3. Bộ nhớ - I/O và giải mã địa chỉ

3.1. Tổ chức bộ nhớ

- Không gian địa chỉ 1MB của 8086 được chia thành 2 bank, mỗi bank 512KB. Bank chẵn (thấp) chứa các byte dữ liệu ở địa chỉ chẵn, bank lẻ (cao) chứa các byte dữ liệu ở địa chỉ lẻ.
- A19 – A1: Xác định vị trí ô nhớ cần truy cập, đưa vào cả 2 bank.
- A0 và $\overline{BHE}$: Dùng làm tín hiệu chọn bank.

3.2. Cấu tạo chung của một chip nhớ



3.3. Phân loại bộ nhớ

- Bộ nhớ không bị mất dữ liệu (Non-Volatile):
- Bộ nhớ bị mất dữ liệu (Volatile)

3.4. Các tín hiệu điều khiển bộ nhớ

4. Giải mã địa chỉ

4.1. Giải mã toàn phần (Full Decoding):

- Tất cả các bit địa chỉ được dùng để xác định vị trí truy cập
- Mỗi IC nhớ hoặc thiết bị I/O được gán 1 địa chỉ duy nhất

4.2. Giải mã một phần (Partial Decoding)

- Không dùng tất cả các bit để xác định vị trí truy cập
- Một số ngoại vi có thể có nhiều hơn 1 địa chỉ (1 ô nhớ vật lý có thể có nhiều hơn 1 địa chỉ, nhiều địa chỉ cùng chỉ tới 1 ô nhớ vật lý)

5. Các loại nhập xuất (I/O)

- Hệ thống I/O của vi xử lý cho phép ngoại vi cung cấp dữ liệu, nhận kết quả xử lý dữ liệu.
- Thực hiện thông qua các cổng I/O (I/O port)
- Có 2 loại I/O port: I/O cách ly (Isolated I/O) và I/O ánh xạ bộ nhớ (Memory – mapped I/O)

Chương 5. NGẮT VÀ I/O SYSTEM

1. Các phương pháp vào ra

1.1. Vai trò của vào ra dữ liệu

- Là phương tiện giúp CPU giao tiếp với thế giới bên ngoài.
- Cung cấp dữ liệu đầu vào cho CPU xử lí.
- Cung cấp phương tiện để CPU kết xuất dữ liệu đầu ra.

1.2. Các phương pháp vào ra chính

- Thăm dò (polling)
- Ngắt (Interrupt)
- Truy nhập trực tiếp bộ nhớ (DMA – Direct Memory Access)

1.3. Các cổng vào ra của máy tính

- PS/2: Cổng ghép nối với bàn phím và chuột
- COM: Các cổng ghép nối nối tiếp
- LPT: Các cổng ghép nối song song
- IDE, SATA, SCSI: Các cổng ghép nối ổ đĩa
- LAN: Cổng ghép nối mạng cục bộ
- Audio: Cổng ghép nối âm thanh (speaker, mic và line-in)
- Video: Cổng ghép nối với màn hình
- DVI: Cổng ghép nối với màn hình (số)
- USB: Cổng ghép nối theo chuẩn USB (USB 1.0: 12Mb/s, USB 2.0: 480Mb/s, USB 3.0: 1.5Gb/s…)

2. Ngắt và xử lý ngắt

2.1. Ngắt (interrupt)

- Ngắt là một sự kiện mà CPU tạm dừng thực hiện một chương trình để thực hiện một đoạn chương trình khác theo yêu cầu từ bên ngoài
- Thông thường các yêu cầu từ bên ngoài thường xuất phát từ các thiết bị vào ra. Các yêu cầu này gọi là các yêu cầu ngắt
- Đoạn chương trình CPU thực hiện trong thời gian ngắt được gọi là chương trình con phục vụ ngắt.
- Các chương trình con phục vụ ngắt là các đoạn chương trình được viết sẵn và lưu trong ROM; mỗi chương trình con phục vụ ngắt có nhiệm vụ riêng và thường là đảm nhiệm việc trao đổi dữ liệu với thiết bị vào ra.
- CPU kiểm tra yêu cầu ngắt tại chu kỳ đồng hồ cuối cùng của chu kỳ lệnh.

2.2. Phân loại ngắt

- Ngắt cứng: Các ngắt được kích hoạt bởi các bộ phận phần cứng gửi đến chân NMI và INTR của CPU, gồm:
- Ngắt không che được NMI (Non-Maskable Interrupt): Ngắt gửi đến chân NMI của CPU, không chịu sự ảnh hưởng của cờ ngắt. Ví dụ: Ngắt Reset…
- Ngắt che được INTR (Maskable Interrupt): Ngắt gửi đến chân INTR của CPU, chịu sự chi phối của cờ ngắt; Cờ IF = 1 → cho phép ngắt, IF = 0 → cấm ngắt.

- Ngắt mềm: Các ngắt được kích hoạt bởi các chương trình thông qua lệnh gọi ngắt INT <N> (N là số hiệu ngắt, N = 0 – 255).
- Các ngắt ngoại lệ: Các ngắt do các lỗi nảy sinh trong quá trình hoạt động của CPU: Ngắt chia cho 0 (divide by zero), Ngắt do tràn (overflow)

2.3. Nguyên tắc hoạt động các ngắt

- Các ngắt được phục vụ trên cơ sở độ ưu tiên.
- Mỗi ngắt được gán một con số (loại ngắt): số nhỏ ưu tiên cao, số lớn ưu tiên thấp
- Khi một chương trình phục vụ ngắt (loại n) đang được phục vụ thì chỉ có những ngắt có độ ưu tiên cao hơn (loại < n) mới được phép ngắt chương trình phục vụ ngắt đang hoạt động.
- Các thiết bị có độ ưu tiên thấp hơn phải chờ cho đến khi trình phục vụ ngắt hiện thời hoàn tất trước khi yêu cầu của nó được xem xét.
- Mức ưu tiên các yêu cầu ngắt (từ cao nhất đến thấp nhất):
    1. Ngắt nội bộ: INT 0 (chia cho 0), INT N (N<>0)
    2. Ngắt không che được NMI
    3. Ngắt che được INTR
    4. Ngắt chạy từng lệnh: INT 1
- Một vector ngắt gồm có số hiệu ngắt N (N=0 – 255 hoặc 00-FFH và địa chỉ đầy đủ chương trình con phục vụ ngắt lưu trong bộ nhớ ROM.

2.4. Bảng Vector ngắt

- Là bảng con trỏ địa chỉ (address pointer table) được dùng để liên kết (tìm tới) loại ngắt với vị trí của chương trình phục vụ ngắt trong bộ nhớ chương trình.
- Nội dung của bảng này có thể được lưu trong ROM, hoặc nạp vào RAM như 1 phần của chương trình khởi động hệ thống.

2.5. Trình tự xử lý ngắt

- Khi nhận được yêu cầu ngắt, CPU thực hiện:
- Hoàn tất lệnh đang thực hiện của chương trình chính
- Lưu giá trị của thanh ghi cờ FR vào ngăn xếp
- Xoá cờ ngắt IF và cờ bẫy TF
- Lưu giá trị của các t.ghi CS và IP vào ngăn xếp
- Từ số hiệu ngắt N, lấy địa chỉ của chương trình con phục vụ ngắt từ bảng vector ngắt
- Nạp địa chỉ của chương trình con phục vụ ngắt vào CS và IP, CPU thực hiện chương trình con phục vụ ngắt:
- Lưu giá trị các thanh ghi dùng chung vào ngăn xếp
- Thực hiện mã chính của chương trình con phục vụ ngắt
- Khôi phục giá trị các thanh ghi dùng chung
- Gặp lệnh IRET kết thúc CTCPVN, CPU thực hiện:
- Khôi phục giá trị của CS và IP
- Khôi phục giá trị của thanh ghi cờ FR
- Đặt cờ ngắt IF và cờ bẫy TF
- CPU tiếp tục thực hiện lệnh tiếp theo của CTC (nằm sau lệnh xảy ra ngắt).

2.6. Các tín hiệu giao tiếp ngắt cứng

Chương 7. HỢP NGỮ

1. Giới thiệu

2. Cấu trúc

```
.MODEL small
.STACK 100h
.DATA
      ; Khai báo các biến ở đây
.CODE
      ; Các lệnh chương trình ghi ở đây
END
```

3. Các thành phần cơ bản

3.1. Tên (name)

- Từ khóa (reserved name): Gồm các chỉ thị (PROC, END…) và các tên lệnh (MOV, ADD, SUB…)
- Tên tự đặt (symbol): Chỉ có 31 ký tự đầu có nghĩa, tên không trùng với từ khóa, không bắt đầu bằng số, không có khoảng trắng. Tên không phân biệt chữ hoa, chữ thường.

3.2. Nhãn (label)

Dùng để đánh dấu lệnh, vị trí trong chương trình, thay thế cho địa chỉ. Nhãn có hai thuộc tính NEAR (được dịch thành địa chỉ 2 byte) và FAR (được dịch thành địa chỉ 4 byte).

Khai báo nhãn:

- Thường đặt trước một lệnh hoặc chỉ thị: **<tên nhãn>:**

Ví dụ: `Main:`

- Khai báo tường minh: **<tên nhãn> LABEL <thuộc tính>**

Ví dụ: `Screen LABEL FAR`

- Khai báo một chương trình con:

```
<tên nhãn> PROC [NEAR/FAR]
      ...
<tên nhãn> ENDP
```

Ví dụ:

```
Dump PROC FAR
...
RET
Dump ENDP
```

3.3. Ghi chú (comment)

3.4. Dòng lệnh

[label :] Operation [operand 1] [, operand 2] [;comment]

4. Khai báo dữ liệu

4.1. Hằng

- Hằng nguyên: Có thể viết dạng nhị phân, thập phân hoặc thập lục phân.

Ví dụ: 11110000b, 200, 200d, 200h…

- Hằng ký tự, hằng chuỗi: Có thể viết trong cặp dấu nháy đơn hoặc nháy kép.

Ví dụ: 'a', 'B', "stack overflow"…

Hằng ký tự có kich thước 1byte, hằng chuỗi có kích thước 1 byte cho mỗi ký tự.

Khai báo: **<tên hằng> EQU <hằng số>**

4.2. Biến (variable)

Dùng thay cho địa chỉ các ô nhớ chứa dữ liệu. Biến có 5 thuộc tính:

Kiểu	Thuộc tính	Kích thước biến (byte)	
DB	BYTE	1	Data
DW	WORD	2	Data
DD	DWORD	4	Con trỏ
DQ	QWORD	8	Số dấu chấm động
DT	TBYTE	10	Số dấu chấm động

Khai báo: **<tên biến> <kiểu> <giá trị> [,...]**

Ví dụ:

```
Value1 DB 20        ; Khai báo biến Value1 = 20 kiểu Byte
Value2 DB ?         ; Khai báo biến Value2 với giá trị ban đầu không xác định
Value3 DW 2AB6h     ; Giá trị được lưu trữ theo thứ tự byte thấp trước, byte cao sau
```

Offset	00	01
Giá trị	B6	2A

```
List DB 10h, 20h, 41, 'A'
```

Offset	00	01	02	03
Giá trị	10	20	41	02

Để truy xuất các ô nhớ liên tiếp của vùng nhớ, có thể viết:

```
MOV DL, List        ; Chép giá trị 10h vào DL
MOV DL, List+1      ; Chép giá trị 20h vào DL
```

4.3. Toán tử, chỉ thị

- Toán tử số học: +, –, *, /, MOD (chia dư), SHR (dịch phải), SHL (dịch trái)
- Toán tử quan hệ: EQ (bằng), NE (khác), LT (nhỏ hơn), LE (nhỏ hơn hoặc bằng), GT (lớn hơn), GE (lớn hơn hoặc bằng).
- Toán tử logic: AND, OR, XOR, NOT
- Toán tử SEG: **SEG <expression>**

Cho địa chỉ đoạn của biểu thức expression (có thể là biến, nhãn, tên segment hoặc toán hạng bộ nhớ khác)

- Toán tử OFFSET: **OFFSET <expression>**

Cho địa chỉ OFFSET của biểu thức expression (có thể là biến, nhãn, tên segment hoặc toán hạng trực tiếp bộ nhớ khác).

Ví dụ: Chép địa chỉ segment vaà offset của biến table vào DS : AX

```
Table DB ?
MOV AX, SEG Table
MOV DS, AX
MOV DX, OFFSET Table
```

- Toán tử PTR: **<type> PTR < expression>**

Cho phép thay đổi dạng của expression. Nếu expr là biến hoặc toán hạng bộ nhớ thì type có thể là BYTE, WORD…. Nếu expr là nhãn thì type có thể là NEAR hoặc FAR.

Ví dụ:

```
CALL FAR PTR table[BX]
MOV BYTE PTR array, 1
MOV [BX], 1 ; Chương trình dịch sẽ báo lỗi vì không biết BX trỏ đến BYTE hay WORD
MOV BYTE PTR [BX], 1
```

- Độ ưu tiên của các toán tử:

- Chỉ thị LABEL

Gán một thuộc tính (BYTE, WORD…) cho một nhãn.