GIÁO TRÌNH

KIẾN TRÚC MÁY TÍNH

CHƯƠNG I. GIỚI THIỆU CHUNG VỀ KIẾN TRÚC MÁY TÍNH.

I. Khái niệm về kiến trúc máy tính

Kiến trúc máy tính (Computer architecture) là một khái niệm trừu tượng của một hệ thống tính toán dưới quan điểm của người lập trình hoặc người viết chương trình dịch.

Nói cách khác, kiến trúc máy tính được xem xét theo khía cạnh mà người lập trình có thể can thiệp vào mọi mức đặc quyền, bao gồm các thanh ghi, ô nhớ các ngắt ... có thể được thâm nhập thông qua các lệnh.

II. Lịch sử phát triển của máy tính.

Chiếc máy tính điện tử đầu tiên là ENIAC được ra đời năm 1946, được chế tạo từ những đèn điện tử, rơle điện tử và các chuyển mạch cơ khí.

Lịch sử phát triển của máy tính điện tử có thể chia làm bốn thế hệ như sau:

- Thế hệ 1: (1945-1955). Máy tính được xây dựng trên cơ sở đèn điện tử mà mỗi đèn tượng trưng cho 1 bit nhị phân. Do đó máy có khối lượng rất lớn, tốc độ chậm và tiêu thụ điện năng lớn. Như máy ENIAC có khối lượng 30 tấn, tiêu thụ công suất 140KW.

- Thế hệ thứ 2: (1955-1965). Máy tính được xây dựng trên cơ sở là các đèn bán dẫn (transistor), máy tính đầu tiên thế hệ này có tênlà TX-0 (transistorized experimental computer 0).

- Thế hệ thứ ba: (1965-1980). Máy tính được xây dựng trên các vi mạch cỡ nhỏ (SSI) và cỡ vừa (MSI), điển hình là thế hệ máy System/360 của IBM. Thế hệ máy tính này có những bước đột phá mới như sau:

- Tính tương thích cao: Các máy tính trong cùng một họ có khả năng chạy các chương trình, phần mềm của nhau.
- Đặc tính đa chương trình: Tại một thời điểm có thể có vài chương trình nằm trong bộ nhớ và một trong số đó được cho chạy trong khi các chương trình khác chờ hoàn thành các thao tác vào/ra.
- Không gian địa chỉ rất lớn.

- Thế hệ thứ tư: (1980-). Máy tính được xây dựng trên các vi mạch cỡ lớn (LSI) và cực lớn (VLSI).

Đây là thế hệ máy tính số ngày nay, nhờ công nghệ bán dẫn phát triển vượt bậc, mà người ta có thể chế tạo các mạch tổ hợp ở mức độ cực lớn. Nhờ đó máy tính ngày càng nhỏ hơn, nhẹ hơn, mạnh hơn và giá thành rẻ hơn. Máy tính cá nhân bắt đầu xuất hiện và phát triển trong thời kỳ này.

Dựa vào kích thước vật lý, hiệu suất và lĩnh vực sử dụng, hiện nay người ta thường chia máy tính số thế hệ thứ tư thành 5 loại chính, các loại có thể trùm lên nhau một phần:

- Microcomputer: Còn gọi là PC (personal computer), là những máy tính nhỏ, có 1 chip vi xử lý và một số thiết bị ngoại vi. Thường dùng cho một người, có thể dùng độc lập hoặc dùng trong mạng máy tính.

- Minicomputer: Là những máy tính cỡ trung bình, kích thước thường lớn hơn PC. Nó có thể thực hiện được các ứng dụngmà máy tính cỡ lớn thực hiện. Nó có khả năng hỗ trợ hàng chục đến hàng trăm người làm việc. Minicomputer được sử dụng rộng rãi trong các ứng dụng thời gian thực, ví dụ trong điều khiển hàng không, trong tự động hoá sản xuất.

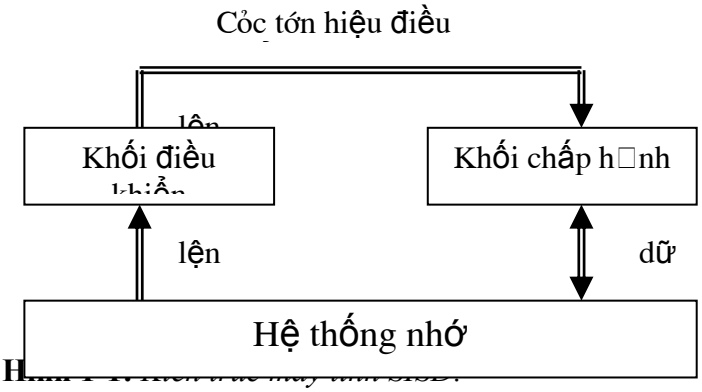
- Supermini: Là những máy Minicomputer có tốc độ xử lý nhanh nhất trong họ Mini ở những thời điểm nhất định. Supermini thường được dùng trong các hệ thống phân chia thời gian, ví dụ các máy quản gia của mạng.

- Mainframe: Là những máy tính cỡ lớn, có khả năng hỗ trợ cho hàng trăm đến hàng ngàn người sử dụng. Thường được sử dụng trong chế độ các công việc sắp xếp theo lô lớn (Large-Batch-Job) hoặc xử lý các giao dịch (Transaction Processing), ví dụ trong ngân hàng.

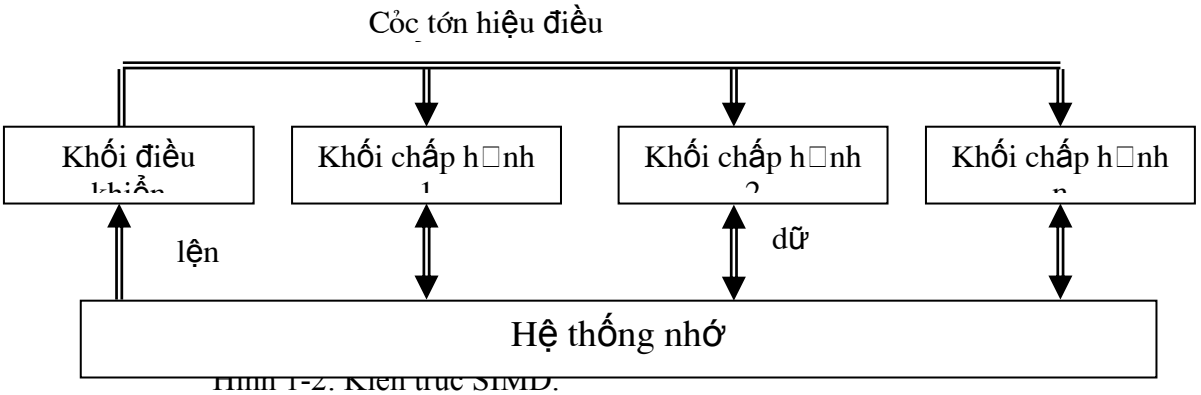
- Supercomputer: Đây là những siêu máy tính, được thiết kế đặc biệt để đạt tốc độ thực hiện các phép tính dấu phẩy động cao nhất có thể được. Chúng thường có kiến trúc song song, chỉ hoạt động hiệu quả cao trong một số lĩnh vực.

Dựa vào kiến trúc của máy tính người ta cũng phân máy tính ra các loại khác nhau như sau:

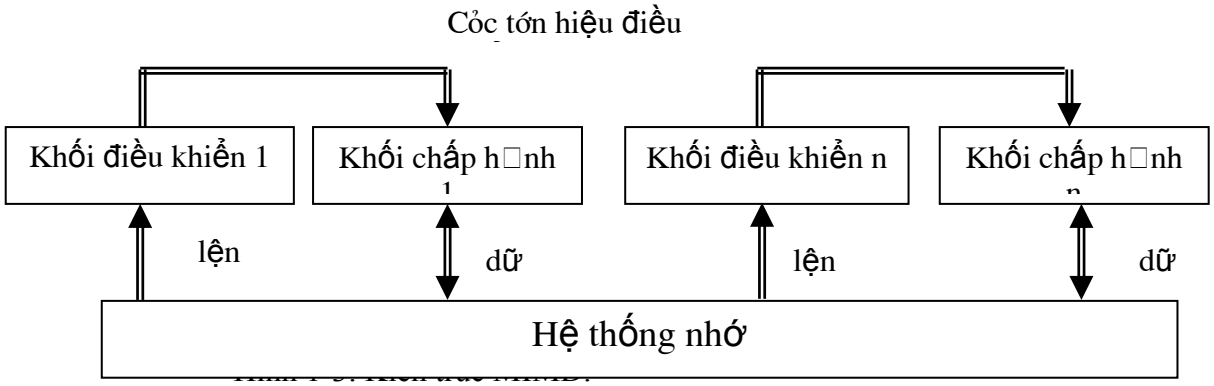
- Kiến trúc SISD (single instruction - single data, đơn dòng lệnh - đơn dòng dữ liệu), sơ đồ như hình 1-1.



- Kiến trúc CIMD (Single Instruction Multiple Data, đơn dòng lệnh- đa dữ liệu), sơ đồ như hình 1-2.



- Kiến trúc MIMD (Multiple Instruction Multiple Data, đa dòng lệnh- đa dữ liệu), sơ đồ như hình 1-3.



CHƯƠNG II. BIỂU DIỄN THÔNG TIN TRONG MÁY TÍNH

I. Hệ nhị phân (Binary)

I.1. Khái niệm:

Hệ nhị phân hay hệ đếm cơ số 2 chỉ có hai con số 0 và 1. Đó là hệ đếm dựa theo vị trí. Giá trị của một số bất kỳ nào đó tuỳ thuộc vào vị trí của nó. Các vị trí có trọng số bằng bậc luỹ thừa của cơ số 2. Chấm cơ số được gọi là chấm nhị phân trong hệ đếm cơ số 2. Mỗi một con số nhị phân được gọi là một bit (BInary digiT). Bit ngoài cùng bên trái là bit có trọng số lớn nhất (MSB, Most Significant Bit) và bit ngoài cùng bên phải là bit có trọng số nhỏ nhất (LSB, Least Significant Bit) như dưới đây:

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

Chấm nhị phân

Số nhị phân $(1010.11)_2$ có thể biểu diễn thành:

$(1010.11)_2 = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 1*2^{-2} = (10.75)_{10}$.

Chú ý: dùng dấu ngoặc đơn và chỉ số dưới để ký hiệu cơ số của hệ đếm.

I.2. Biến đổi từ nhị phân sang thập phân

Ví dụ : Biến đổi số nhị phân $(11001)_2$ thành số thập phân:

Trọng số vị trí: 2^4 2^3 2^2 2^1 2^0
Giá trị vị trí: 16 8 4 2 1
Số nhị phân: 1 1 0 0 1
Số thập phân: $1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = (25)_{10}$

I.3. Biến đổi thập phân thành nhị phân

Để thực hiện việc đổi từ thập phân sang nhị phân, ta áp dụng phương pháp chia lặp như sau: lấy số thập phân chia cho cơ số để thu được một thương số và số dư. Số dư được ghi lại để làm một thành tố của số nhị phân. Sau đó, số thương lại được chia cho cơ số một lần nữa để có thương số thứ 2 và số dư thứ 2. Số dư thứ hai là con số nhị phân thứ hai. Quá trình tiếp diễn cho đến khi số thương bằng 0.

Ví dụ 1: Biến đổi số thập phân $(29)_{10}$ thành nhị phân:

29/2 = 14 + 1(LSB)
14/2 = 7 + 0
7/2 = 3 + 1
3/2 = 1 + 1
1/2 = 0 + 1(MSB)

Vậy $(29)_{10} = (1101)_2$.

Đối với phần lẻ của các số thập phân, số lẻ được nhân với cơ số và số nhớ được ghi lại làm một số nhị phân. Trong quá trình biến đổi, số nhớ đầu chính là **bit** MSB và số nhớ cuối là **bit** LSB.

Ví dụ 2: Biến đổi số thập phân $(0.625)_{10}$ thành nhị phân:

0.625*2 = 1.250. Số nhớ là 1, là **bit** MSB.
0.250*2 = 0.500. Số nhớ là 0
0.500*2 = 1.000. Số nhớ là 1, là **bit** LSB.
Vậy : $(0.625)_{10} = (0.101)_2$.

II. Hệ thập lục phân (Hexadecima).

II.1. Khái niệm:

Các hệ máy tính hiện đại thường dùng một hệ đếm khác là hệ thập lục phân.

Hệ thập lục phân là hệ đếm dựa vào vị trí với cơ số là 16. Hệ này dùng các con số từ 0 đến 9 và các ký tự từ A đến F như trong bảng sau:

Bảng 2.1 *Hệ thập lục phân:*

Thập lục phân	Thập phân	Nhị phân
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000

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

II.2.Biến đổi thập lục phân thành thập phân.

Các số thập lục phân có thể được biến đổi thành thập phân bằng cách tính tổng của các con số nhân với giá trị vị trí của nó.

Ví dụ : Biến đổi các số a.$(5B)_{16}$. b. $(2AF)_{16}$ thành thập phân.

a. Số thập lục phân: 5 B
Trọng số vị trí: 16^1 16^0
Giá trị vị trí : 16 1
Số thập phân: $5*16 + B*1 = (91)_{10}$.

b. Số thập lục phân: 2 A F
Trọng số vị trí: 16^2 16^1 16^0
Giá trị vị trí : 256 16 1
Số thập phân: $2*256 + A*16 + F*1 = (687)_{10}$.

II.3.Biến đổi thập phân thành thập lục phân.

Để biến đổi các số thập phân thành thập lục phân, ta sử dụng phương pháp chia lặp, với cơ số 16.

Ví dụ : Biến đổi $(1776)_{10}$ thành thập lục phân.
1776/16 = 111 + 0 (LSB).
111/16 = 6 + 15 hoặc F.
6/16 = 0 + 6 (MSB).
Số thập lục phân: $(6F0)_{16}$.

II.4. Biến đổi thập lục phân thành nhị phân.

Các số thập lục phân rất dễ đổi thành nhị phân. Thực ra các số thập lục phân cũng chỉ là một cách biểu diễn các số nhị phân thuận lợi hơn mà thôi (bảng 2-1). Để đổi các số thập lục phân thành nhị phân, chỉ cần thay thế một cách đơn giản từng con số thập lục phân bằng bốn bit nhị phân tương đương của nó.

Ví dụ: Đổi số thập lục $(DF6)_{16}$ thành nhị phân:

D F 6
↓ ↓ ↓
1101 1111 0110

$(DF6)_{16} = (110111110110)_2$.

II.5. Biến đổi nhị phân thành thập lục phân.

Để biến đổi một số nhị phân thành số thập lục phân tương đương thì chỉ cần gộp lại thành từng nhóm gồm 4 bit nhị phân, bắt đầu từ dấu chấm nhị phân.

Ví dụ: Biến đổi số nhị phân $(1111101000010000)_2$ thành thập lục phân.

1111 1010 0001 0000
↓ ↓ ↓ ↓

F A 1 0 Số thập lục phân: $(FA10)_{16}$.

III. Hệ BCD (Binary Code decimal).

Giữa hệ thập phân và hệ nhị phân còn tồn tại một hệ lai: hệ BCD cho các số *hệ thập phân mã hoá bằng hệ nhị phân*, rất thích hợp cho các thiết bị đo có thêm phần hiển thị số ở đầu ra dùng các loại đèn hiện số khác nhau. Ở đây dùng bốn số hệ nhị phân (bốn bit) để mã hoá một số hệ thập phân có giá trị nằm trong khoảng từ 0..9. Như vậy ở đây ta không dùng hết các tổ hợp có thể có của 4 bit; vì tầm quan trọng của các số BCD nên các bộ vi xử lý thường có các lệnh thao tác với chúng.

Ví dụ: $(35)_{10} = (00110101)_2$.

IV. Bảng mã ASCII.(American **S**tandard **C**ode for **I**nformation **I**nterchange**).**

Người ta đã xây dựng bộ mã để biểu diễn cho các ký tự cũng như các con số Và các ký hiệu đặc biệt khác. Các mã đó gọi là **bộ mã ký tự và số.** Bảng mã ASCII là mã 7 bit được dùng phổ biến trong các hệ máy tính hiện nay. Với mã 7 bit nên có $2^7 = 128$ tổ hợp mã. Mỗi ký tự (chữ hoa và chữ thường) cũng như các con số thập phân từ 0..9 và các ký hiệu đặc biệt khác đều được biểu diễn bằng một mã số như bảng 2-2.

Việc biến đổi thành ASCII và các mã ký tự số khác, tốt nhất là sử dụng mã tương đương trong bảng.

Ví dụ: Đổi các ký tự BILL thành mã ASCII:

Ký tự	B	I	L	L
ASCII	1000010	1001001	1001100	1001100
HEXA	42	49	4C	4C

Bảng 2-2: *Mã ASCII.*

							Column bits($B_7B_6B_5$)							
	Bits(row)						000	001	010	011	100	101	110	111
ROW	B_4	B_3	B_2	B_1			0	1	2	3	4	5	6	7
0	0	0	0	0	→		NUL	DLE	SP	0	@	P	\	p
1	0	0	0	1	→		SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	→		STX	DC2	“	2	B	R	b	r
3	0	0	1	1	→		ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	→		EOT	DC4	$	4	D	T	d	t
5	0	1	0	1	→		ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	→		ACK	SYN	&	6	F	V	f	v
7	0	1	1	1	→		BEL	ETB	‘	7	G	W	g	w
8	1	0	0	0	→		BS	CAN	(	8	H	X	h	x
9	1	0	0	1	→		HT	EM	)	9	I	Y	i	y
A	1	0	1	0	→		LF	SUB	*	:	J	Z	j	z
B	1	0	1	1	→		VT	ESC	+	;	K	[	k	{
C	1	1	0	0	→		FF	FS	-	<	L	\	l	|
D	1	1	0	1	→		CR	GS	,	=	M	]	m	}
E	1	1	1	0	→		SO	RS	.	>	N	^	n	~
F	1	1	1	1	→		SI	US	/	?	O	_	o	DEL

Control characters:
NUL = Null; DLE = Data link escape; SOH = Start Of Heading;
DC1 = Device control 1; DC2 = Device control 2; DC3 = Device control 3.
DC4 = Device control 4; STX = Start of text; ETX = End of text;
EOT = End of transmission; ENQ = Enquiry; NAK = Negative acknowlege.
ACK = Acknowlege; SYN = Synidle; BEL = Bell.
ETB = End od transmission block; BS = Backspace; CAN = Cancel.
HT = Horizontal tab; EM = End of medium; LF = Line feed; SUB = Substitute.
VT = Vertical tab; ESC = Escape; FF = From feed; FS = File separator.
SO = Shift out; RS = Record separator; SI = Shift in; US = Unit separator.

V. Biểu diễn giá trị số trong máy tính.

V.I. Biểu diễn số nguyên.

a. Biểu diễn số nguyên không dấu:

Tất cả các số cũng như các mã ... trong máy vi tính đều được biểu diễn bằng các chữ số nhị phân. Để biểu diễn các số nguyên không dấu, người ta dùng n bit. Tương ứng với độ dài của số bit được sử dụng, ta có các khoảng giá trị xác định như sau:

Số bit	Khoảng giá trị
n bit:	0.. 2^n - 1
8 bit	0.. 255 Byte
16 bit	0.. 65535 Word

b. Biểu diễn số nguyên có dấu:

Người ta sử dụng bit cao nhất biểu diễn dấu; bit dấu có giá trị 0 tương ứng với số nguyên dương, bit dấu có giá trị 1 biểu diễn số âm. Như vậy khoảng giá trị số được biểu diễn sẽ được tính như sau:

Số bit	Khoảng giá trị:
n bit	2^{n-1}-1
8 bit	-128.. 127 Short integer
16 bit	-32768.. 32767 Integer
32 bit	-2^{31}.. 2^{31}-1 (-2147483648.. 2147483647) Long integer

V.2. Biểu diễn số thực(số có dấu chấm (phẩy) động).

Có hai cách biểu diễn số thực trong một hệ nhị phân: số có dấu chấm cố định (fïed point number) và số có dấu chấm động (floating point number). Cách thứ nhất được dùng trong những bộ VXL(micro processor) hay những bộ vi điều khiển (micro controller) cũ. Cách thứ 2 hay được dùng hiện nay có độ chính xác cao. Đối với cách biểu diễn số thực dấu chấm động có khả năng hiệu chỉnh theo giá trị của số thực. Cách biểu diễn chung cho mọi hệ đếm như sau:

$$R = m.B^e.$$

Trong đó m là phần định trị, trong hệ thập phân giá trị tuyệt đối của nó phải luôn nhỏ hơn 1. Số e là phần mũ và B là cơ số của hệ đếm.

Có hai chuẩn định dạng dấu chấm động quan trọng là: chuẩn MSBIN của Microsoft và chuẩn IEEE. Cả hai chuẩn này đều dùng hệ đếm nhị phân.

Thường dùng là theo tiêu chuẩn biểu diễn số thực của IEEE 754-1985(Institute of Electric & Electronic Engineers), là chuẩn được mọi hãng chấp nhận và được dùng trong bộ xử lý toán học của Intel. Bit dấu nằm tại vị trí cao nhất; kích thước phần mũ và khuôn dạng phần định trị thay đổi theo từng loại số thực.

Giá trị số thực IEEE được tính như sau:

$$R = (-1)^S*(1+M_1*2^{-1} + ... +M_n*2^{-n})*2^{E\ 7...E\ 0\ -127}.$$

Chú ý: giá trị đầu tiên M_0 luôn mặc định là 1.

- Dùng 32 bit để biểu diễn số thực, được số thực ngắn: $-3,4.10^{38} < R < 3,4.10^{38}$

31	30 23	22 0
S	E7 - E0	|Định trị (M1 - M23)

- Dùng 64 bit để biểu diễn số thực, được số thực dài: $-1,7.10^{308} < R < 1,7.10^{308}$

63	62 52	51 0
S	E10 - E0	Định trị (M1 - M52)

Ví dụ tính số thực:

0100 0010 1000 1100 1110 1001 1111 1100

Phần định trị: $2^{-4}+2^{-5}+2^{-8}+2^{-9}+2^{-10}+2^{-12}+2^{-15}+$
$+2^{-16}+2^{-17}+2^{-18}+2^{-19}+2^{-20}+2^{-21} = 0,1008906$.
Giá trị ngầm định là: 1,1008906.

Phần mũ: $2^8+2^2+2^0 = 133$
Giá trị thực (bit cao nhất là bit dấu): 133-128=6.

Dấu: 0 = số dương

Giá trị số thực là: $R = 1,1008906.2^6 = 70,457$.

Phương pháp đổi số thực sang số dấu phẩy động 32 bit:
- Đổi số thập phân thành số nhị phân.
- Biểu diễn số nhị phân dưới dạng ±1, xxxBy (B: cơ số 2).
- Bit cao nhất 31: lấy giá trị 0 với số dương, 1 với số âm.
- Phần mũ y đổi sang mã excess -127 của y, được xác định bằng cách: $y + (7F)_{16}$.
- Phần xxx là phần định trị, được đưa vào từ bit 22..0.

Ví dụ: Biểu diễn số thực $(9,75)_{10}$ dưới dạng dấu phẩy động.
Ta đổi sang dạng nhị phân: $(9,75)_{10} = (1001.11)_2 = 1,00111B3$.
Bit dấu: bit 31 = 0.
Mã excess - 127 của 3 là: $7F + 3 = (82)_{16} = 82H = (10000010)_2$. Được đưa vào các bit tiếp theo: từ bit 30 đến bit 23.
Bit 22 luôn mặc định là 0.
Cuối cùng số thực $(9,75)_{10}$ được biểu diễn dướiu dạng dấu phẩy động 32 bit như sau:

0100 0001 0001 1100 0000 0000 0000 0000

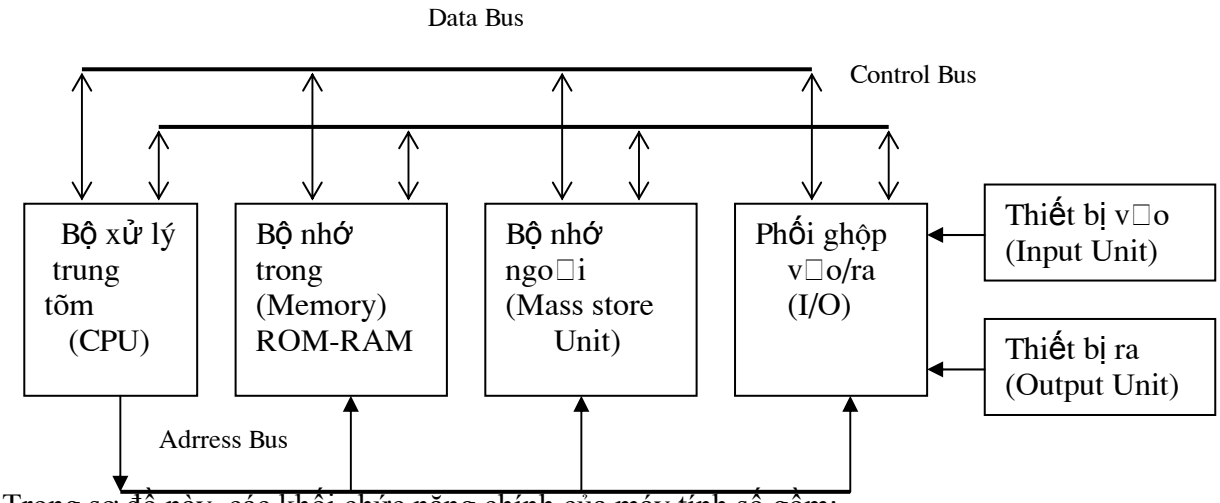
bit |31|30 23|22 0|

CHƯƠNG III. CÁC KHỐI CƠ BẢN CỦA MÁY TÍNH

I. Giới thiệu sơ lược cấu trúc của máy vi tính.

So với từ khi ra đời, cấu trúc cơ sở của các máy vi tính ngày nay không thay đổi mấy. Mọi máy tính số đều có thể coi như được hình thành từ sáu phần chính (như hình 3-1):

Hình 3-1: Giới thiệu sơ đồ khối tổng quát của máy tính số



Trong sơ đồ này, các khối chức năng chính của máy tính số gồm:

- Khối xử lý trung tâm (central processing unit, CPU),
- Bộ nhớ trong (memory), như RAM, ROM
- Bộ nhớ ngoài, như các loại ổ đĩa, băng từ
- Khối phối ghép với các thiết bị ngoại vi (vào/ra)
- Các bộ phận đầu vào, như bàn phím, chuột, máy quét ... .
- Các bộ phận đầu ra, như màn hình, máy in ... .

Bốn khối chức năng đầu liên hệ với nhau thông qua tập các đường dây để truyền tín hiệu, gọi chung là *bus hệ thống*. Bus hệ thống bao gồm 3 bus thành phần; ứng với các tín hiệu xác lập địa chỉ từ CPU đến các đơn vị thành phần ta có bus địa chỉ; với các dữ liệu được liên hệ giữa các khối qua bus dữ liệu (data bus); các tín hiệu điều khiển bao gồm các lệnh, các đáp ứng, các trạng thái của các khối được xác lập qua bus điều khiển.

Sự khác biệt quan trọng nhất của các hệ máy tính là kích thước và tốc độ, các máy tính nhỏ hơn và nhanh, mạnh hơn theo từng năm. Sự phát triển không ngừng của các thế hệ máy tính nhờ vào hai yếu tố quan trọng, đó là sự phát triển của công nghệ chế tạo IC và công nghệ chế tạo bộ nhớ.

II. Bộ nhớ trong.

II.1. Cơ sở về bộ nhớ.

Các bộ nhớ có thể chia làm hai loại tổng quát, ROM và RAM. ROM là Read-only Memory(bộ nhớ chỉ đọc) và RAM là Random-access Memory (bộ nhớ truy xuất ngẫu nhiên). Nói chung ROM chứa các dữ liệu một cách cố định và không thể thay đổi. Còn RAM có thể đọc ra và có thể ghi vào.

Khái niệm truy xuất ngẫu nhiên có nghĩa là bất kỳ một vị trí nhớ nào cũng có thể được mở ra hoặc được gọi ra ở bất kỳ lúc nào, các thông tin không cần phải đọc ra hay ghi vào một cách tuần tự. Về thực chất, cả RAM và ROM đều là truy xuất ngẫu nhiên. Chỉ có điều khác nhau cơ bản là ROM chỉ cho phép đọc mà không thể ghi vào nó, còn RAM là bộ nhớ có thể đọc và ghi, vì thế RAM được gọi là "bộ nhớ đọc/ghi".

Cấu trúc bộ nhớ

Hình 2-2 trình bày sơ đồ khối của một mạch nhớ. Mạch nhớ được nối với các bộ phận khác trong máy tính thông qua các đường đây địa chỉ và các đường dây dữ liệu của nó. Kiểm soát mạch nhớ bằng đường dây cho phép (enable), riêng đối với RAM còn có thêm đường dây kiểm soát đọc/ghi (Read/write). $\overline{\text{——}}$ $\overline{\text{——}}$

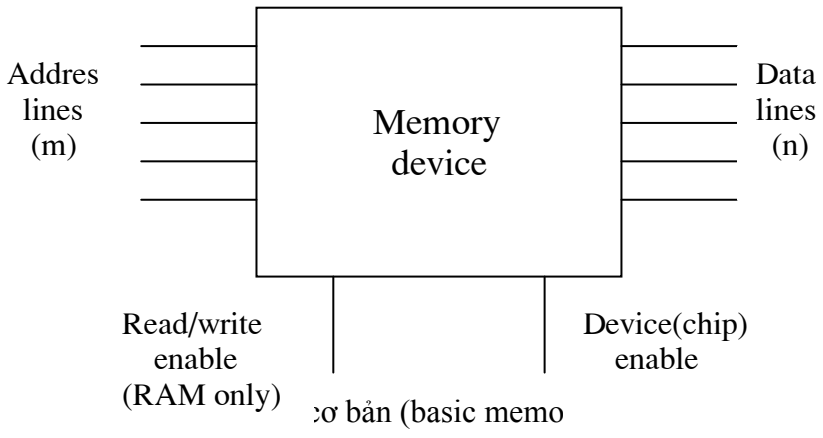
Các mạch nhớ nói chung được tổ chức dưới dạng ma trận, gồm những hàng và những cột để xác định vị trí hay địa chỉ nhớ. Mỗi ô trong ma trận gọi là một *phần tử* (cell) hay *vị trí nhớ* (memory location). Vị trí hay phần tử nhớ được dò tìm bằng cách chọn địa chỉ nhờ mạch giải mã địa chỉ. Mạch này gồm hai phần: **mạch chọn địa chỉ hàng RAS** (row-address selector) và **mạch chọn địa chỉ cột CAS** (Column-address selector). Các đường dây địa chỉ sẽ chọn địa chỉ hàng và cột. Đường dây enable dùng để mở các mạch điện lối ra bộ nhớ theo ba trạng thái. Còn đường dây Read/write quyết định dạng thao tác sẽ thực hiện.

Bộ nhớ hoặc là có tổ chức bit hoặc là loại có tổ chức lời (word organized). Bộ nhớ tổ chức bit có thể lưu giữ một bit đơn trong mỗi vị trí địa chỉ. Bộ nhớ tổ chức lời sẽ được lựa chọn cả một nhóm phần tử nhớ cùng một lúc với mỗi vị trí địa chỉ. Mỗi nhốm phần tử nhớ thường là một byte (8 bit), hoặc một lời (16 bit).

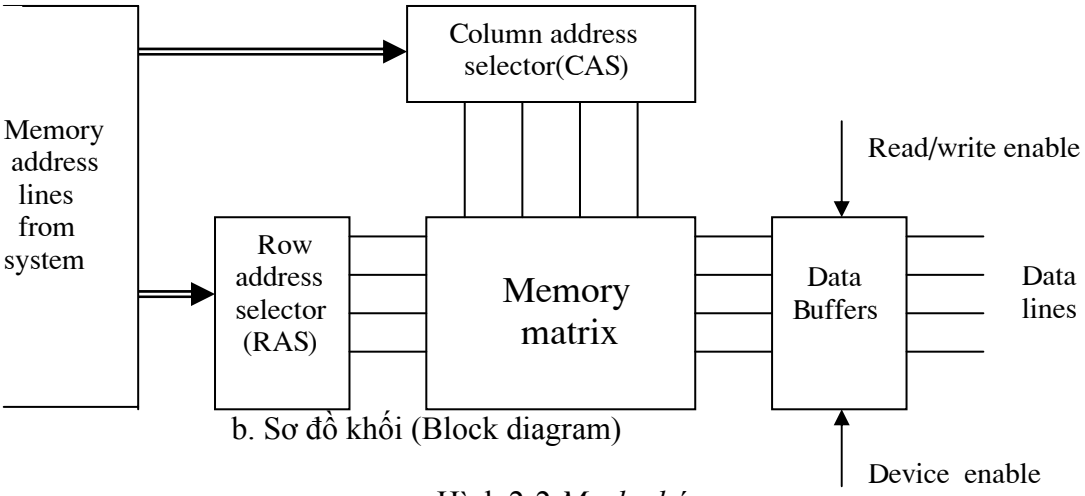
Số đường dây địa chỉ của mạch nhớ sẽ quyết định số vị trí nhớ cực đại tính theo công thức sau:

Số vị trí nhớ cực đại = 2^N.

trong đó, N là số lượng các đường địa chỉ.



:ơ bản (basic memo



b. Sơ đồ khối (Block diagram)

Hình 2-2 *Mạch nhớ.*

II.2. ROM-BIOS.

Bất cứ hệ máy tính nào cũng có một vi mạch ROM. vi mạch này chứa chương trình của hệ điều hành vào ra cơ sở BIOS (basic input/output system). Những chương trình này cần thiết để khởi động máy và cài đặt chế độ làm việc cơ sở cho các thiết bị ngoại vi.

Nói chung, có thể chia ROM thành bốn loại. **ROM mặt nạ** (maskable ROM) là loại ROM do nhà sản xuất đã nạp sẵn dữ liệu, khi đó dữ liệu không thể thay đổi được nữa. **ROM có thể nạp chương trình** (PROM - programable ROM) là loại mạch mà người dùng có thể nạp dữ liệu vào thông qua thiết bị **"đốt"** PROM. Khi đã nạp thì các dữ liệu trong PROM cũng không thể thay đổi. **PROM có thể xoá**, còn gọi là EPROM (erasable PROM) là loại ROM mà người dùng có thể nạp dữ liệu vào và các dữ liệu đó có thể xoá hoặc thay đổi bằng một thiết bị đặc biệt. EPROM có thể xoá bằng điện (electric EPROM) là loại ROM có thể nạp và xoá dữ liệu bằng điện được mà không phải sử dụng tia cực tms như với EPROM.

Trong các máy tính hiện đại, người ta thường sử dụng Flash BIOS dùng EEPROM. Như vậy nội dung BIOS của máy tính có thể được thay đổi để tương thích với những mở rộng và nâng cấp hệ thống, mà điều này là không thể thực hiện đối với những máy tính thế hệ cũ sử dụng BIOS dùng PROM hoặc EPROM.

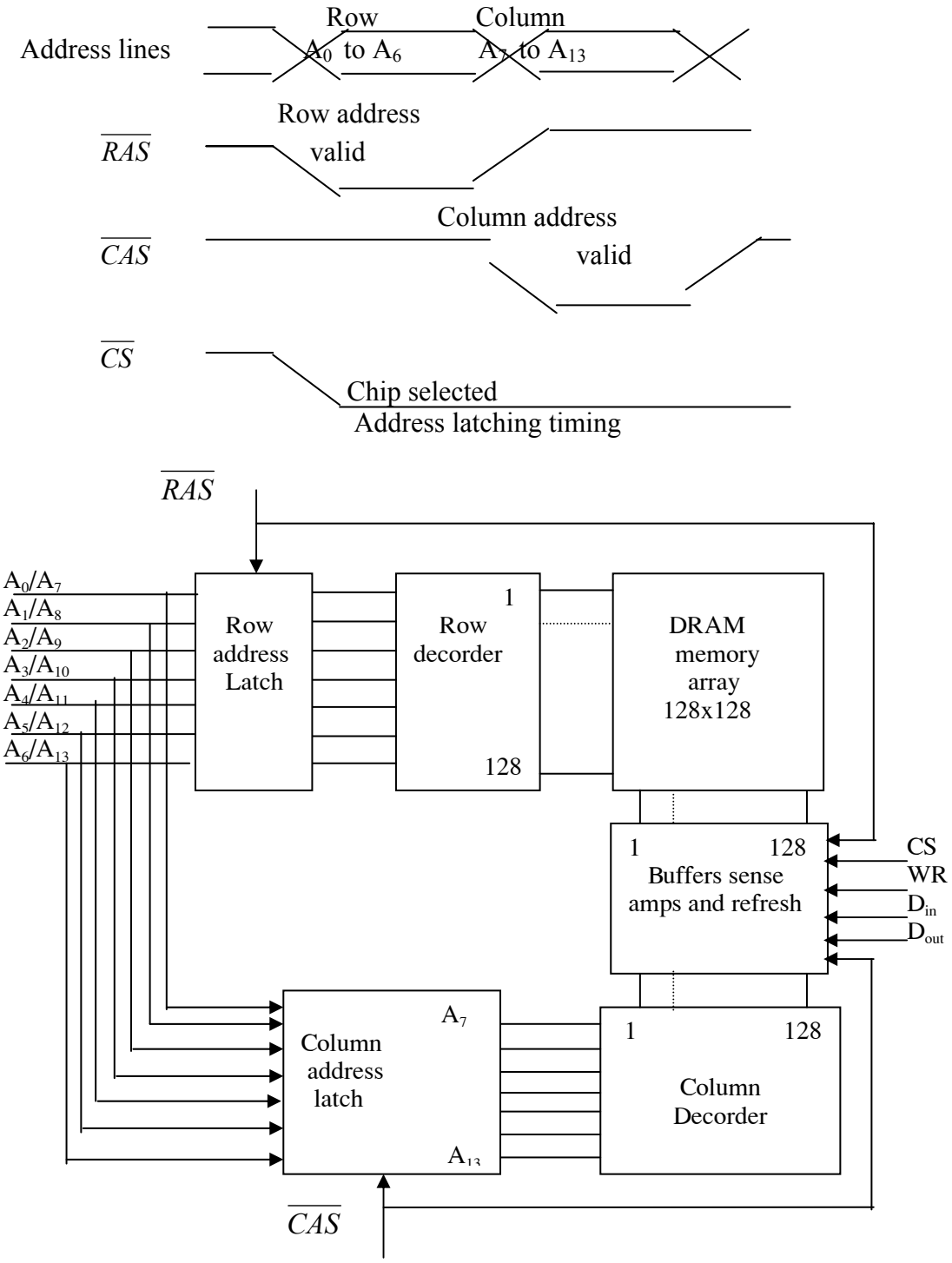
BIOS gồm nhiều chương trình và hàm. Phần đầu của chương trình BIOS kiểm tra hệ thống máy tính, quá trình này gọi là POST. Nếu hệ thống sử dụng các Card (thẻ cắm) Plug and Play thì giai đoạn này chính là lúc máy tính truy nhập tham số của thẻ. BIOS nào cũng có chương trình "Setup BIOS" để người dùng tự chỉnh tham số các thiết bị ngoại vi.

II.3. RAM.

Có thể chia RAM thành hai hoại, RAM tĩnh (SRAM), có khả năng lưu giữ số liệu mãi mãi nếu như không mất nguồn nuôi. Và RAM động (DRAM), là loại RAM phải được "làm tươi" (refresh) tức là phải nạp lại dữ liệu đang được lưu trữ theo từng chu kỳ. "Làm tươi" bằng cách thực hiện thao tác đọc hoặc ghi nhắc lại. Cũng có thể "làm tươi" bằng những thao tác đặc biệt khác. Loại DRAM có mật độ phần tử nhớ cao nên giá thành khá rẻ so với SRAM. Các mạch nhớ DRAM được dùng phổ biến trong các thế hệ máy tính hiện nay.

Để tiết kiệm số đường địa chỉ và giảm số chân trên IC, hầu hết các loại DRAM đều dùng phương pháp địa chỉ multiplex. Trong quá trình đọc hay ghi các đường địa chỉ đầu tiên chứa các thông tin về hàng rồi tiếp sau mang thông tin về cột. Để kiểm soát thao tác này, người ta dùng đường dây $\overline{RAS}$ và $\overline{CAS}$ như trên hình 2-3. Khi $\overline{RAS}$ thấp thì thông tin trên các đường địa chỉ sẽ được mở thông qua mạch chốt địa chỉ hàng (row-address latch). Khi $\overline{CAS}$ thấp thì thông tin trên các đường địa chỉ sẽ được mở thông qua mạch chốt địa chỉ cột (column-address latch).

Việc "làm tươi" bằng dữ liệu đọc, dữ liệu ghi hoặc bằng các thao tác riêng. Mạch điều khiển làm tươi phải chọn tuần tự từng hàng các phần tử nhớ, cứ mỗi hàng một lần, cho đến khi tất cả các hàng đều được "làm tươi". Đó là phương pháp làm tươi từng đợt. Trong quá trình đó không được đọc hay ghi dữ liệu vào bộ nhớ cho đến khi kết thúc quá trình. Một cách khác là "làm tươi" từng hàng trong các chu kỳ rời rạc và gọi là làm tươi theo chu kỳ đơn.



Hình 2-3. *Sơ đồ khối DRAM 16.384 bits(16Kb).*

III. Bộ xử lý trung tâm CPU.

Bộ xử lý trung tâm CPU là cốt lõi của một máy vi tính. CPU thực hiện mọi tính toán và xử lý của hệ thống -- ngoại trừ xử lý tăng cường tính toán đặc biệt trong những hệ thống có một chip đơn vị đồng xử lý toán, mà chip này cũng đã được tích hợp ngay trong các CPU hiện nay. Tất cả những máy tính IBM và tương thích IBM sử dụng những bộ xử lý họ Intel hoặc tương thích với bộ xử lý họ Intel, dù chính những bộ xử lý có thể đã được nhiều công ty khác nhau thiết kế và sản xuất, gồm AMD, IBM, Cyric... .

Một trong những bộ xử lý điển hình thuộc họ 80x86 của Intel là bộ xử lý 8088. Đây là bộ vi xử lý khá đơn giản và vì vậy việc tìm hiểu nó là tương đối dễ đối với những người bắt đầu thâm nhập vào lĩnh vực vi xử lý, mặt khác việc nắm vững các vấn đề kỹ thuật của bộ vi xử lý 8088 sẽ là cơ sở để nắm bắt được các kỹ thuật của các bộ xử lý khác trong họ 80x86 của Intel, của các họ khác và của các bộ xử lý hiện đại ngày nay.

III.1. Giới thiệu cấu trúc bên trong của bộ vi xử lý 8088.

Trên hình 3-1 là sơ đồ khối cấu trúc bên trong của bộ vi xử lý 8088.

III.3. Đơn vị giao diện bus (BIU).

Theo sơ đồ khối trên hình 3-1 ta thấy bên trong CPU 8088 có hai khối chính: *khối phối ghép bus* (bus interface unit, BIU) và *khối thực hiện lệnh* (execution unit, EU). Việc chia CPU thành hai phần đồng thời có liên hệ với nhau qua đệm lệnh làm tăng đáng kể tốc độ xử lý của CPU. Các bus bên trong CPU có nhiệm vụ chuyển tải tín hiệu của các khối khác. Trong số các bus có bus dữ liệu 16 bit của ALU, bus các tín hiệu điều khiển ở EU và bus trong của hệ thống ở BIU. Trước khi đi ra bus ngoài hoặc đi vào bus trong của bộ vi xử lý, các tín hiệu truyền trên bus thường được cho đi qua các bộ đệm để nâng cao tính tương thích cho nối ghép hoặc nâng cao khả năng phối ghép.

BIU bao gồm các thanh ghi đoạn (segment registers: CS, DS, SS, ES), con trỏ lệnh IP (instruction pointer) và bộ điều khiển logic bus (bus control logic, BCL). Đơn vị giao diện BIU còn có bộ nhớ đệm cho mã lệnh. Bộ nhớ này có chiều dài 4 byte (trong 8088) và 6 byte (trong 8086). Bộ nhớ đệm mã lệnh được nối với khối điều khển CB (control block) của đơn vị thực hiện lệnh EU. Bộ nhớ này lưu trữ tạm thời mã lệnh trong một dãy gọi là hàng đợi lệnh. Hàng đợi lệnh cho phép bộ vi xử lý có khả năng xử lý xen kẽ liên tục dòng mã lệnh (pipelining). Hoạt động của bộ CPU được chia làm ba giai đoạn: đọc mã lệnh (operation code fetching), giải mã lệnh (decording) và thực hiện lệnh (execution).

BIU đưa ra địa chỉ, đọc mã lệnh từ bộ nhớ, đọc/ghi dữ liệu từ các cổng vào hoặc bộ nhớ. Nói cách khác BIU chịu trách nhiệm đưa địa chỉ ra bus và trao đổi dữ liệu với bus.

III.3. Đơn vị thực hiện lệnh (EU).

Trong EU có khối điều khiển (control unit, CU). Chính tại bên trong khối điều khiển này có mạch giải mã lệnh. Mã lệnh đọc vào từ bộ nhớ được đưa đến đầu vào của bộ giải mã, các thông tin thu được từ đầu ra của nó sẽ được đưa đến mạch tạo xung điều khiển, kết quả thu được là các dãy xung khác nhau tuỳ theo mã lệnh, để điều khiển hoạt động của các bộ phận bên trong và bên ngoài CPU.

Trong EU có khối số học và lôgic (arithmatic and logic unit, ALU) chuyên thực hiện các phép tính số học và logic mã toán tử của nó nằm trong các thanh ghi đa năng. Kết quả thường được đặt về thanh ghi AX.

Ngoài ra trong EU còn có các thanh ghi đa năng (registers: AX, BX, CX, DX, SP, BP, SI, DI), thanh ghi cờ FR (flag register) mà công dụng của chúng sẽ đượoc đề cập đến trong phần sau.

Tóm lại, khi CPU hoạt động EU sẽ cung cấp thông tin về địa chỉ cho BIU để khối này đọc lệnh và dữ liệu, còn bản thân nó thì giải mã và thực hiện lệnh.

III.4. Các thanh ghi.

Các thanh ghi đa năng (general registers) Có nhiệm vụ ghi tham số cho mã lệnh, đây cũng là nơi lệnh trả kết quả về sau khi được thực hiện. Những thanh ghi đa năng của vi xử lý 16 bit là:

- AX (accumulator) rộng 16 bit, được chia làm hai phần: 1 byte cao AH và 1 byte thấp AL. Đây là thanh ghi quan trọng nhất và chuyên được dùng để chứa kết quả các thao tác lệnh. Cả ba cách viết AX, AH, AL đều có thể sử dụng như nững thanh ghi riêng biệt.

- BX (base) thanh ghi cơ sở, rộng 16 bit, cũng được chia ra làm BH và BL. Đây là thanh ghi thường dùng chứa địa chỉ cơ sở của một bảng dùng trong lệnh XLAT, Cả ba cách viết BX, BH, BL đều có thể sử dụng như những thanh ghi riêng biệt.

- CX (count) bộ đếm, rộng 16 bit. Được chia ra làm CH và CL. Thanh ghi CX được ùng để chứ số lần lặp trong trường hợp các lệnh LOOP. Thanh ghi thấp CL được dùng để chứa (nhớ) số lần quay hoặc dịch của các lệnh quay (rotate) và dịch (shift).

- DX (data) thanh ghi dữ liệu, rộng 16 bit. Thanh ghi này cùng thanh ghi AX tham gia vào các thao tác của phếp nhân hoặc chia các số 16 bit. DX còn dùng để chứa địa chỉ 16 bit của các cổng cứng (dài hơn 8 bit) trong các lệnh truy nhập các cổng ngoại vi (I/O port).

Các thanh ghi đoạn (segment registers) dùng để ghi địa chỉ một đoạn bộ nhớ. Vi mạch 8088/8086 có 20 đường dây trên bus địa chỉ. Do các thanh ghi con trỏ cà thanh ghi chỉ số chỉ rộng 16 bit nên không thể định địa chỉ cho toàn bộ nhớ vật lý của máy tính là (2^{20} = 1.048.576 = 1Mbyte). Vì vậy trong chế độ thực (real mode) bộ nhớ được chia làm nhiều đoạn để một thanh ghi con trỏ 16 bit có thể quản lý được. Các thanh ghi đoạn 16 bit sẽ chỉ ra địa chỉ đầu của 4 đoạn trong bộ nhớ, dung lượng lớn nhất của mỗi đoạn nhớ sẽ dài 2^{16} = 64 Kbyte và tại một thời điểm nhất định bộ vi xử lý chỉ làm việc được với 4 đoạn nhớ 64Kbyte này. Việc thay đổi giá trị của các thanh ghi đoạn làm cho các đoạn có thể dịch chuyển linh hoạt trong không gian 1 Mbyte, vì vậy các đoạn có thể nằm cách nhau khi thông tin cần lưu trong chúng đòi hỏi dung lượng đủ 64 Kbyte hoặc cũng có thể nắm trùm nhau do có những đoạn không dùng hết độ dài 64 Kbyte và vì thế các đoạn khác có thể bắt đầu nối tiếp ngay sau đó. Địa chỉ của ô nhớ nằm ở đầu đoạn được ghi trong một thanh ghi đoạn 16 bit, địa chỉ này gọi là *địa chỉ cơ sở*. Mười sáu bit này tương ứng với các đường dây địa chỉ từ A4 đến A20. Như vậy giá trị vật lý của địa chỉ đoạn là giá trị trong thanh ghi đoạn dịch sang trái 4 vị trí. Điều này tương đương với phép nhân với 2^4 = 16. Địa chỉ của các ô nhớ khác nằm trong đoạn tính được bằng cách cộng thêm vào địa chỉ cơ sở một giá trị gọi là địa chỉ lệch hay độ lệch (offset), gọi như thế vì nó ứng với khoảng lệch của toạ độ một ô nhớ cụ thể nào đó so với ô đầu đoạn. Độ lệch này được xác định bởi các thanh ghi 16 bit khác đóng vai trò thanh ghi lệch (offset register). Nguyên tắc này dẫn đến công thức tính địa chỉ vật lý (physical address) từ địa chỉ đoạn (segment) trong thanh ghi đoạn và địa chỉ lệch (offset) trong thanh ghi con trỏ như sau:

Địa chỉ vật lý = Thanh ghi đoạn x 16 + Thanh ghi lệch

Việc dùng hai thanh ghi để nhớ thông tin về địa chỉ thực chất tạo ra một loại địa chỉ gọi là địa chỉ logic và được ký hiệu như sau:

Thanh ghi đoạn : Thanh ghi lệch hay **segment:offset.**

Địa chỉ kiểu **segment : offset** là logic vì nó tồn tại dưới dạng giá trị của các thanh ghi cụ thể bên trong CPU và khi cần thiết truy nhập ô nhớ nào đó thì nó phải đổi ra địa chỉ vật lý để rồi đưa lên bus địa chỉ. Việc chuyển đổi này do một bộ tạo địa chỉ thực hiện (phần tử ∑ trên hình 3-1).

Vi xử lý 16 bit có 4 thanh ghi đoạn như sau:

- CS (code segment) là thanh ghi đoạn mã 16 bit. thanh ghi này phối hợp với con trỏ lệnh IP để ghi địa chỉ mã lệnh trong bộ nhớ. Địa chỉ đầy đủ là CS:IP.
- DS (data segment) là thanh ghi đoạn 16 bit cho một đoạn dữ liệu. Thanh ghi này phối hợp với hai thanh ghi chỉ số SI và DI để đánh địa chỉ cho dữ liệu. Địa chỉ đầy đủ cho dữ liệu cần đọc vào là DS:SI, cho dữ liệu cần ghi ra là DS:DI.
- SS (stack segment) là thanh ghi đoạn 16 bit cho một ngăn xếp. Địa chỉ đỉnh của ngăn xếp được biểu diễn cùng với con trỏ ngăn xếp SP là SS:SP.
- ES (extra segment) là thanh ghi dữ liệu phụ có chiều dài 16 bit. Thường được dùng để đánh địa chỉ một chuỗi. ES:DI là địa chỉ chuỗi cần viết đến (chuỗi đích) và DS:SI là địa chỉ chỗi đọc vào (chuỗi nguồn).

Các thanh ghi con trỏ và chỉ số có thể được dùng như một thanh ghi đa năng 16 bit. Vi mạch 8088 có tất cả ba thanh ghi con trỏ là (IP, BP, SP) và hai thanh ghi chỉ số (SI, DI). Nhiệm vụ của từng thanh ghi như sau:

- IP (instruction pointer) là con trỏ chỉ tới lệnh máy tiếp theo. Lệnh này nằm trong bô nhớ mà địa chỉ đoạn được ghi trong CS. Như vậy địa chỉ của mã k=lệnh này là CS:IP.
- BP (base pointer) là con trỏ cơ sở trỏ về dữ liệu bộ nhớ mà địa chỉ đoạn được ghi trong SS. Địa chỉ đầy đủ sẽ là SS:BP.
- SP (stack pointer) là con trỏ ngăn xếp luôn trỏ vào đỉnh ngăn xếp mà địa chỉ đoạn được ghi trong SS. Địa chỉ đầy đủ của dữ liệu là DS:SP.
- SI (source index) là chỉ số nguồn, trỏ vào dữ liệu mà địa chỉ đoạn được ghi trong DS. Địa chỉ đầy đủ của dữ liệu là DS:SI.
- DI (destination index) là chỉ số đích, cũng trỏ vào đoạn dữ liệu mà địa chỉ đoạn ghi trong DS. Địa chỉ đầy đủ của đoạn dữ liệu là DS:SI.

Thanh ghi cờ FR (flag register) đây là thanh ghi khá đặc biệt trong CPU, dùng để ghi trạng thái kết quả các phép xử lý trong đơn vị số học và logic ALU hoặc một trạng thái hoạt đọnh của EU. Dựa vào các cờ này người lập trình có thể có các lệnh thích hợp tiếp theo cho bộ vi xử lý (các leẹnh nhảy có điều kiện). Thanh ghi này là một thanh ghi 16 bit trong 8088/8086. Nhưng chỉ có 9 bit trong thanh ghi được định nghĩa và sử dụng, đó là:

x	x	x	x	O	D	I	T	S	Z	x	A	x	P	x	C

x: bit không được định nghĩa.

Hình 3-2. *Sơ đồ thanh ghi cờ của bộ vi xử lý 8086/8088.*

- Bit 0: CF (carry flag) cờ nhớ, CF=1 khi có nhớ hoặc mượn từ MSB.
- Bit 2: PF (parity flag) cờ parity, PF phản ánh tính chẵn (parity) của tổng số bit 1 có trong kết quả. Cở PF =1 khi tổng số bit 1 trong kết quả là chẵn (even parity, parity chẵn).
- Bit 4: AF (auxliary carry flag) cờ nhớ phụ dùng cho các phép tính với mã BCD. AF = 1 khi có nhớ hoặc mượn từ một số BCD thấp (4 bit thấp) sang một số BCD cao (4 bit cao).
- Bit 6: ZF (zero flag) cờ rỗng, ZF = 1 khi kết quả bằng 0.
- Bit 7: SF (sing flag) cờ dấu, SF = 1 khi kết quả âm.
- Bit 8: TF (trap flag) cờ bẫy, TF = 1 khi vi xử lý ở trong chế độ chạy từng lệnh (chế độ này dùng khi cần tìm lỗi trong một chương trình).
- Bit 9: IF (interrupt enable flag) cờ cho phép ngắt, IF = 1 cho phép các yêu cầu ngắt che được (maskable interrupt) được tác động.

- Bit A: DF (direction flag) cờ hướng. DF = 1 khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải sang trái (lùi).
- Bit B: OF (overflow) cờ tràn, OF =1 khi kết quả vượt ra ngoài giới hạn, xảy ra đối với phép tính có dấu.

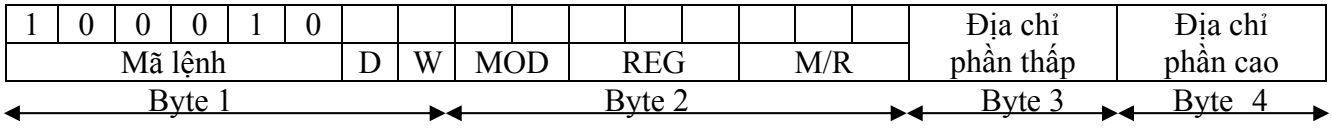
CHƯƠNG IV . LỆNH VÀ CHẾ ĐỘ ĐỊA CHỈ

I. Cấu trúc mã lệnh

Quy trình thực hiện một lệnh trong bộ vi xử lý được chia làm ba giai đoạn: Lấy lệnh (feeching), giải mã lệnh (decording) và xử lý lệnh (excution). Những bộ VXL cổ điển 8 bit tiến hành ba giai đoạn trên một cách tuần tự. Từ các bộ VXL 16 bit trở đi, bộ VXL dùng pipeline (xen kẽ dòng lệnh) để tiết kiệm thời gian xử lý. Mã lệnh dành cho VXL được viết dưới dạng mã nhị phân. Để con người có thể lập trình và hiểu được VXL, người ta dùng hợp ngữ (assembly language) để miêu tả các lệnh máy bằng tổ hợp các ký tự gợi nhớ (mnemonic).

Một lệnh mô tả bằng mã nhị phân có thể dài từ 1 đến 6 byte. Cấu trúc chung của một mã lệnh bao gồm:
- Prefix đi trước mã lệnh.
- Mã toán (operation code) phân biệt đó là lệnh gì, ví dụ với lệnh dịch chuyển MOV có mã toán là 100010.
- Toán hạng (operand) cho biết cái gì được xử lý (nội dung của thanh ghi hay bộ nhớ).
- Địa chỉ trực tiếp (2 byte).

Nội dung của mã lệnh được quy định khá chặt chẽ. hình 4-1 dưới đây cho thấy cấu trúc nhị phân của một lệnh dịch chuyển **MOV đích, nguồn** dùng để chuyển dữ liệu giữa 2 thanh ghi hoặc giữa ô nhớ và thanh ghi.



Hình 4-1: *Cấu trúc mã lệnh*

- Bit D (direction) chỉ hướng cho thanh ghi REG. D=1 chỉ dữ liệu đi đến REG; D=0 thì chỉ dữ liệu đi từ REG.

- Bit W (Word) chỉ xem thanh ghi được dùng là 8 bit hay 16 bit (1 word). W=1 có nghĩa là thanh ghi 16 bit được dùng. Bảng 4-1 cho thấy cách mã hoá các thanh ghi trong bộ VXL:

- Hai bit MOD (mode, chế độ) và **ba bit R/M** (register/memory, thanh ghi/bộ nhớ) tạo ra 5 bit, dùng để chỉ *chế độ địa chỉ* của lệnh. Những chế độ này được quy định trong bảng 4-1. Bảng 4-2 cho thấy cách mã hoá các chế độ địa chỉ (cách tìm ra các toán hạng) bằng các bit này.

Bảng 4-1: *Cách mã hoá các thanh ghi trong bộ VXL.*

Thanh ghi W=1	Thanh ghi W=0	Mã REG
AX	AL	000
BX	BL	011
CX	CL	001
DX	DL	010
SP	AH	100
DI	BH	111
BP	CH	101
SI	DH	110

Thanh ghi đoạn	Mã
ES	00
CS	01
SS	10
DS	11

Bảng 4-2: *Phối hợp MOD và R/M để tạo ra các chế độ địa chỉ.*

MOD / R/M	**00**	**01**	**10**	**11**	
				W=0	W=1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (Địa chỉ trực tiếp)	[BP] + d8	[BP] + d16	DH	SI

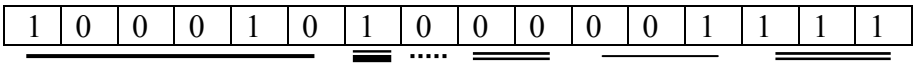
111	[BX]	[BX] + d8	[BX] + d16	BH DI
	← Chế độ bộ nhớ →			Chế độ thanh ghi

Ghi chú: - d8: disp. 8 bit, d16: disp. 16 bit.

- Các gía trị cho trong các cột 2, 3, 4 là các địa chỉ hiệu dụng (EA) sẽ được cộng với DS để tạo ra địa chỉ vật lý (riêng BP phải được cộng với SP).

Ví dụ 1: Mã hoá các lệnh: a. MOV CL,[BX]; b. MOV 0F3H[SI],CL.

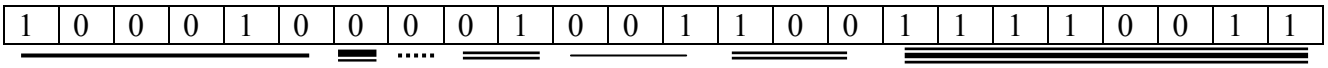
a. MOV CL, [BX]



____ Các bit mã hoá CL; ═══ ô nhớ có địa chỉ DS:BX;

····· Chuyển 1 bite; ═══ Opcode. ═══ Chuyển tới thanh ghi

b. MOV 0F3H[SI], CL



____ Các bit mã hoá CL; ═══ ô nhớ có địa chỉ DS:SI; ····· chuyển 1 bite;

____ Opcode. ═══ Chuyển từ thanh ghi; ═══ d8 = F3H.

II. Tập lệnh của bộ vi xử lý.

Mỗi bộ vi xử lý có một tập lệnh xác định, các bộ vi xử lý thế hệ sau thường có tập lệnh được bổ sung, mở rộng hơn so với các bộ vi xử lý thế hệ trước nó, điều đó có nghĩa các bộ vi xử lý thế hệ sau có thể chạy được các chương trình viết cho các bộ vi xử lý trước. Nhưng ngược lại thì không hoàn toàn đúng.

Như đã nói trên đây, chúng ta lấy bộ vi xử lý Intel 8088 làm cơ sở để nghiên cứu những vấn đề kỹ thuật của các bộ vi xử lý khác. Vì vậy ở đây chúng ta cũng sẽ nghiên cứu tập lệnh của chính bộ vi xử lý này.

Tập lệnh của 8086/8088 gồm hơn 100 ký hiệu gợi nhớ (mnemonic) của lệnh ngôn ngữ assembler cơ sở, để quy định cho bộ vi xử lý phải làm gì. Mỗi lệnh cơ sở có thể có nhiều biến cách. Ví dụ có tới 28 biến cách khác nhau cho lệnh dịch chuyển cơ sở (MOV) ... . Tuy nhiên trong chương trình môn học này, chúng ta chỉ xem xét một số lệnh cần thiết theo mục tiêu của môn học. Các lệnh mà chúng ta sẽ nghiên cứu được chia làm 6 nhóm:

1. Nhóm lệnh truyền dữ liệu.
2. Nhóm lệnh số học.
3. Nhóm lệnh logic.
4. Nhóm lệnh so sánh.
5. Nhóm lệnh điều khiển chương trình.
6. Các lệnh đặc biệt.

II.1 . Nhóm lệnh truyền dữ liệu *(không ảnh hưởng đến các cờ).*

MOV lệnh di chuyển dữ lệu cơ bản . Lệnh này cóthể sử dụng để di chuyển byte (8 bit) hoặc lời (16 bit) của dữ liệu. Cấu trúc lệnh :

MOV đích, nguồn.

Trong đó toán hạng đích và gốc có thể tìm theo các địa chỉ khác nhau, nhưng phải có cùng độ dài và không được phép đồng thời là 2 ô nhớ hoặc 2 thanh ghi đoạn.

Các ví dụ cho trong bảng 4-3:

Bảng 4-3 *các ví dụ về lệnh MOV.*

Đích	Nguồn	Ví dụ	Giải thích
1 Bộ nhớ	Thanh ghi	**MOV 100H, AX**	- chuyển nôi dung trong AX vào vị trí nhớ 100H.
2 Thanh ghi	Bộ nhớ	**MOV AX, MEM1**	- Chuyển nội dung trong vị trí nhớ do nhãn MEM1 chỉ ra vào thanh ghi AX.
3 Thanh ghi	Thanh ghi	**MOV AX, BX**	- Chuyển nội dung trong BX vào thanh ghi AX.
4 Thanh ghi	Tức thời	**MOV AX, 0FFFFH**	- Chuyển giá trị hằng số FFFFH vào thanh ghi AX; số 0 ở đầu được dùng để phân biệt và chỉ rõ FFFFH là một giá trị hằng chứ không phải là một nhãn.

XCHG -exchange two operands (hoán đổi nội dung 2 toán hạng).

Viết lệnh: XCHG Đích, Nguồn

Trong đó toán hạng đích và nguồn có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải có cùng độ dài và không được phép đồng thời là 2 ô nhớvà cũng không được là thanh ghi đoạn.

Ví dụ:

```
XCHG AH, AL      ; tráo nội dung AH và AL.
XCHG AL, [BX]    ; tráo nội dung AL với ô nhớ có địa chỉ DS:BX.
```

IN- Input data from a port (Đọc dữ liệu từ cổng vào thanh Acc)

Viết lệnh: IN Acc, Port

Port là địa chỉ 8 bit của cổng, nó có thể có giá trị trong khoảng 00H..FFH.

Nếu Acc là AL thì dữ liệu 8 bit được đưa vào từ cổng Port.

Nếu Acc là AX thì dữ liệu 16 bit được đưa vào từ cổng Port và Port+1.

Có thể biểu diễn địa chỉ cổng thông qua thanh ghi DX và như vậy địa chỉ cổng được địa chỉ hoá linh hoạt hơn. Lúc này địa chỉ cổng nằm trong dải 0000H..FFFFH và lệnh được viết như sau:

IN Acc, DX

Trong đó DX phải được gán từ trước giá trị ứng với cổng.

OUT- Output a byte or word to a port (Đưa dữ liệu ra cổng từ Acc).

Viết lệnh: OUT Port, Acc

Nếu Acc là AL thì dữ liệu 8 bit được đưa ra cổng Por

Nếu Acc là AH thì dữ liệu 16 bit được đưa ra cổng Port và cổng Port+1.

Tương tự với lệnh **IN**, ở đây cũng có thể dùng thanh ghi DX để chứa địa chỉ cổng. Khi đó lệnh được viết như sau:

OUT DX, Acc.

Thanh ghi DX phải được nạp địa chỉ cổng từ trước.

LEA (load effective address). Lệnh nạp địa chỉ hiệu dụng vào thanh ghi, nó không di chuyển nội dung chứa trong địa chỉ đó. Đây là lệnh để tính địa chỉ lệch hoặc địa chỉ của ô nhớ chọn làm gốc rồi nạp vào thanh ghi đã chọn.

Viết lệnh: **LEA Đích, nguồn.**

trong đó :

- Đích thường là một trong các thanh ghi BX, CX, DX, BP, SI, DI.
- Nguồn là tên biến trong đoạn DS được chỉ rõ trong lệnh hoặc ô nhớ cụ thể.

Ví dụ:

```
LEA DX, MSG          ; Nạp địa chỉ lệch của bản tin MSG vào DX.
LEA CX, [BX] [DI] ; Nạp vào CX địa chỉ hiệu dụng do
                     ; BX và DI chỉ ra: EA=BX+DI.
```

PUSH/POP Thanh ghi ngăn xếp là nơi rất thuận tiện để cất giữ tạm dữ liệu và các toán hạng cần nhớ của chương trình. Ví dụ, một chương trình có thể muốn cất lại các nội dung trong thanh ghi AX để dùng trong một số thao tác sau này. Để thực hiện nhiệm vụ đó có thể dùng các lệnh **PUSH** và **POP**.

- PUSH Cất dữ liệu vào ngăn xếp.

Viết lệnh: **PUSH nguồn**

Mô tả: SP ← SP - 2

Nguồn → {SP}.

trong đó toán hạng gốc có thể tìm được theo các chế độ địa chỉ khác nhau: có thể là các thanh ghi đa năng, thanh ghi đoạn hoặc ô nhớ. Lệnh này thường dùng với lệnh POP như một cặp đối ngẫu để xử lý các dữ liệu và trạng thái của chuương trình chính khi vào/ra chương trình con.

Ví dụ:

```
PUSH BX              ; cất BX vào ngăn xếp, tại vị trí do SP chỉ ra.
PUSH Table[BX]   ; cất 2 byte của vùng dữ liệu DS
                     ; có địa chỉ đầu tại (Table+BX).
```

- POP Lấy dữ liệu từ ngăn xếp.

Viết lệnh: **POP Đích**

Mô tả: Đích → {SP}.

SP ← SP + 2

trong đó toán hạng gốc có thể tìm được theo các chế độ địa chỉ khác nhau: có thể là các thanh ghi đa năng, thanh ghi đoạn (nhưng không được là thanh ghi đoạ mã CS) hoặc ô nhớ. Dữ liệu để tại ngăn xếp không thay đổi. Giá trị của SS không thay đổi. **Ví dụ:**

```
POP DX               ; lấy 2 byte từ đỉnh ngăn xếp, đưa vào DX.
PUSH Table[BX]   ; lấy 2 byte ở đỉnh ngăn xếp rồi để tại vùng DS
                     ; có địa chỉ đầu tại (Table+BX).
```

PUSHF/POPF Các nội dung của thanh ghi cờ có thể được gửi vào hay lấy ra khỏi ngăn xếp bằng các lệnh **PUSPF** và **POPF**.

- PUSHF Cất nội dung thanh ghi cờ vào ngăn xếp.

Viết lệnh: PUSHF

Mô tả: SP ← SP - 2
RF → {SP}.
Dữ liệu để tại thanh ghi cờ không thay đổi. SS không thay đổi.

- POPF Lấy 1 từ, từ đỉnh ngăn xếp đưa vào thanh ghi cờ.
Viết lệnh: POPF
Mô tả: RF → {SP}.
SP ← SP + 2
Sau lệnh này dữ liệu để tại ngăn xếp không thay đổi. SS không thay đổi.

II.2. Nhóm lệnh số học *(là nhóm lệnh có ảnh hưởng đến cờ).*

Các lệnh số học bao gồm bốn phép tính số học cơ bản là cộng, trừ , nhân, chia và đảo dấu toán hạng.

ADD/SUB Dạng tổng quát của các lệnh cộng (add) và trừ (subtract) là:
ADD đích, nguồn
SUB đích, nguồn
Mô tả: ADD: Đích ← Đích + Nguồn
SUB : Đích ← Đích -Nguồn
trong đó các toán hạng đích, nguồn có thể tìm được theo các địa chỉ khác nhau, nhưng phải chứa dữ liệu có cùng độ dài và không được phép đồng thời là hai ô nhớ và cũng không được là thanh ghi đoạn.

Bảng 4-4 tóm tắt các loại khác nhau của các toán hạng đích và nguồn dùng trong các lệnh cộng và trừ:

Bảng 4-4. *các dạng toán hạng trong lệnh ADD/SUB:*

Đích (nơi đến)	Nguồn (gốc)
Thanh ghi	Thanh ghi
Thanh ghi	Bộ nhớ
Bộ nhớ	Thanh ghi
Bộ nhớ	Tức thời (hằng số)
Thanh ghi	Tức thời(hằng số)

Ví dụ 1:
```
ADD AX, BX        ;  AX ← AX+BX
ADD AL, 74H       ;  AX ← AX+ 74H
SUB CL, AL              ; CL ← CL - AL
SUB AX, 0405H           ; AX ← AX - 0405H.
```
Ví dụ 2: Viết đoạn chương trình ngôn ngữ assembly để cộng 5H với 3H, dùng các thanh ghi AL, BL.
```
MOV AL, 05H       ; AL ← 05H
MOV BL, 03H       ; BL ← 03H
ADD AL, BL  ; AL ← 05H+03H =08H
MOV 100H, AL      ; Di chuyển kết quả từ AL vào vị trí nhớ DS:100H.
```

MUL/DIV Dạng tổng quát của lệnh nhân (multiply, MUL) và chia (divide, DIV) là:
MUL số nhân nguồn

DIV số chia nguồn

trong đó số nhân nguồn (toán hạng gốc) có thể tìm được theo các chế độ địa chỉ khác nhau. Khi dùng lệnh nhân, số được nhân phải được chuyển vào thanh ghi AX hoặc AL. Còn số nhân thì có thể chuyển vào thanh ghi khác bất kỳ hoặc một địa chỉ nhớ.

Ví dụ 2:

```
MUL BX          ; số nhân nằm trong thanh ghi BX
MUL MEM1   ; số nhân nằm trong địa chỉ nhớ mang nhãn MEM1
```

Khi hai byte nhân với nhau thì kết quả được gửi lưu vào thanh ghi AX.

Ví dụ 3. Viết đoạn chương trình nhân 5H với 3H, dùng thanh ghi CL.

```
MOV AL, 05H     ; AL ← 05H (số được nhân)
MOV CL, 03H     ; CL ← 03H (số nhân)
MUL CL          ; AL ← 0FH (kết quả)
MOV MEM1, AL    ; chuyển kết quả (0FH)
                ; từ AL vào vị trí nhớ có nhãn MEM1.
```

Khi nhân hai lời (16 bit) với nhau thì số được nhân phải chuyển vào thanh ghi AX, còn số nhân có thể ở trong một thanh ghi khác bất kỳ hoặc trong vị trí nhớ 16 bite. kết quả sẽ là con số 32 bit (hoặc hai lời) và được chứa trong các thanh ghi DX và AX. Lời có trọng số lớn sẽ ở trong thanh ghi DX và lời có trọng số nhỏ sẽ ở trong thanh ghi AX.

Ví dụ 4. Viết đoạn chương trình để nhân 3A62H với 2B14H.

```
MOV AX, 3A62H       ; AX ← 3A62H
MOV CX, 2B14H       ; CX ← 2B14H
MUL CX              ; DXAX ← tích = 289C63A8H
```

Các lệnh chia, về cơ bản, cũng giống như các lệnh nhân. Trong phép chia cỡ byte, số chia là một byte có thể ở trong một thanh ghi hoặc một vị trí nhớ. Số bị chia phải là một số không dấu 16 bit chứa trong thanh ghi AX. Kết quả thương số sẽ ở trong thanh ghi AL, còn số dư thì ở trong thanh ghi AH. Đối với phép chia cỡ lời thì số chia 16 bit có thể đặt trong thanh ghi hoặc một vị trí nhớ. Còn số bị chia phải là một số không dấu 32 bit được đặt trong các thanh ghi DX và AX. Thanh ghi DX sẽ giữ lời có trọng số cao, thanh ghi AX sẽ giữ lời có trọng số thấp. Kết quả thương đặt trong thanh ghi AX, còn số dư đặt trong thanh ghi DX.

Ví dụ 5: Viết đoạn chương trình để chia 6H cho 3H, dùng thanh ghi CL. MOV AX, 0006H ; AX ← 6H

```
MOV CL, 03H     ; CL ← 3H
DIV CL          ; AHAL ← 00H (số dư), 02H (thương số)
```

Chú ý: 6H được đưa vào thành 0006H để lấp đầy toàn bộ thanh ghi AX. Như vậy các byte trọng số cao của AX sẽ bị xoá để tránh bị lỗi.

Ví dụ 6: Viết đoạn chương trình để chia 1A034H cho 1002H, dùng thanh ghi BX

```
MOV AX, 0A034H  ; AX ← 0A034H
MOV DX, 0001H   ; DX ← 0001H
MOV BX, 1002H   ; BX ← 1002H
DIV BX          ; DXAX ← 00H (số dư)1AH (thương số)
```

INC/DEC Đây là lệnh tăng (increment) và giảm (decrement). Lệnh tăng sẽ cộng thêm một đơn vị vào toán hạng, còn lệnh giảm sẽ trừ một đơn vị vào toán hạng. Các lệnh này rất cần đối với thao tác đếm. Dạng tổng quát của các lệnh INC và DEC là:

INC đích Mô tả: Đích ← Đích +1

DEC đích Mô tả: Đích ← Đích -1

Toán hạng đích có thể là một thanh ghi hoặc một vị trí nhớ bất kỳ, có thể là 1 lời 16 bit hoặc 1 byte; có thể tìm được theo các chế độ địa chỉ khác nhau.

Chú ý:

- Trong lệnh tăng, nếu Đích = FFH (hoặc FFFFH) thì Đích + 1 = 00H (hoặc 0000H) mà không ảnh hưởng đến cờ nhớ. Lệnh này cho kết quả tương đương như lệnh **ADD Đích, 1** nhưng chạy nhanh hơn.
- Trong lệnh giảm, nếu đích là 00H (hoặc 0000H) thì Đích -1 = FFH (hoặc FFFFH) mà không ảnh hưởng đến cờ nhớ CF. Lệnh này cho kết quả tương đương với lệnh **SUB Đích, 1** nhưng chạy nhanh hơn.

NEG- Negative a Operand (lấy bù 2 của một toán hạng hay đảo dấu toán hạng).
Viết lệnh: NEG Đích

Ví dụ:

```
NEG AH                  ; AH ← 0 - (AH)
NEG BYTE PTR[BX]        ; lấy bù 2 của ô nhớ do BX chỉ ra trong DS.
```

II.3. Nhóm lệnh logic *(có ảnh hưởng đến cờ).*

Các lệnh logic nhằm thực hiện các phép tính Boolean NOT, AND và OR. Lệnh NOT thì đảo tất cả các bit trong toán hạng (byte boặc lời). Các lệnh AND/OR thực hiện các phép tính AND/OR đối với một đôi bit trong toán hạng nguồn và toán hạng đích. Các lệnh này có thể dùng với các toán hạng cỡ lời hoặc cỡ byte.

NOT Lấy bù của một toán hạng, đảo bit của một toán hạng.

Viết lệnh: **NOT Đích.** Mô tả: **Đích ← $\overline{\text{(Đích)}}$**

trong đó toán hạng đích có thể tìm được theo các chế độ địa chỉ khác nhau.

Lệnh này không tác động đến cờ.

Ví dụ 1: Xác định kết quả của đoạn chương trình sau:

```
MOV BL, 00110011B
NOT BL
MOV MEM1, BL
```

Nội dung của thanh ghi BL được nạp vào là 00110011B. Sau khi thực hiện phép NOT thì nội dung của thanh ghi BL là 11001100B và giá trị này được đưa vaò vị trí nhớ được chỉ ra bởi nhãn MEM1.

AND/OR: Và/Hoặc hai toán hạng. dạng tổng quát của lệnh AND/OR là:

```
AND Đích, Nguồn
OR Đích, Nguồn
```

trong đó toán hạng đích và nguồn có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa dữ liệu cùng độ dài và không được phép đồng thời là hai ô nhớ và cũng không được là thanh ghi đoạn.

AND/OR sẽ thực hiện phép tính Boolean đối với các toán hạng nguồn và đích. Phép AND thường dùng để che đi/giữ lại một vài bit nào đó của một toán hạng bằng cách nhân logic toán hạng đó với toán hạng tức thời có các bit 0/1 tại các vị trí cần che/giữ lại tương ứng. Phép

OR thường dùng để lập một vài bit nào đó của toán hạng bằng cách cộng lôgic toán hạng đó với toán hạng tức thời có các bit 1 tại các vị trí tương ứng cần thiết lập (toán hạng tức thời trong những trường hợp này còn được gọi là mặt nạ).

Ví dụ 2 :

```
AND AL, BL   ; nội dung thanh ghi BL được giao với nội dung trong
               ; thanh ghi AL và kết quả được lưu trong thanh ghi
               ; AL(AX). Nếu con số trong AL là 00001101B và
               ; trong BL là 00110011B thì kết quả trong thanh ghi
               ; AL sau phép AND là:  AL 0000001B.

OR AL, BL    ; nội dung thanh ghi BL được hợp với nội dung trong
               ; thanh ghi AL từng bit một và kết quả được lưu trong
               ;thanh ghi AL(AX). Nếu con số trong AL là
               ; 00001101B và trong BL là 00110011B thì kết quả
               ;trong thanh ghi AL sau phép AND là: AL 0011111B.
```

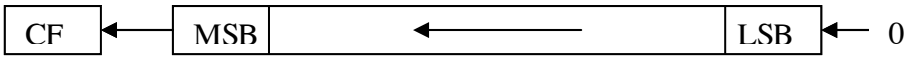
Ví dụ 3:

```
AND BL, 0FH  ; che 4 bit cao của BL.
OR BL, 30H   ; lập 4 bit b4 và b5 của BL lên 1.
```

SAL- Shift arithmetically Left (Dịch trái số học)/ **SHL- Shift (Logically) Left** (Dịch trái logic).

Viết lệnh: SAL Đích, CL
SHL Đích, CL

Mô tả:



Mỗi lần dịch MSB sẽ được đưa qua cờ CF và 0 được đưa vào LSB. Thao tác kiểu này được gọi là dịch logic. CL phải được chứa sẵn số lần dịch mong muốn. Thực chất mỗi lần dịch trái tương đương với một lần làm phép nhân với 2 của số không dấu. Vì vậy ta có thể làm phép nhân số bị nhân không dấu với 2^i bằng cách dịch trái số học số bị nhân i lần. Chính vì vậy thao tác này còn được gọi là dịch trái số học.

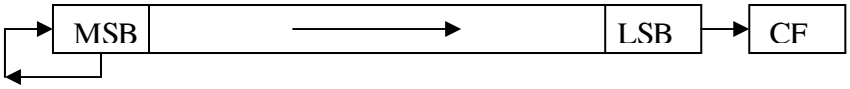
Sau lệnh SAL/SHL, cờ CF mang giá trị cũ của MSB, vì vậy lệnh này cò dùng để tạo cờ CF từ giá trị của MSB làm điều kiện cho các lệnh nhảy có điều kiện. Còn cờ OF ← 1 nếu sau khi dịch 1 lần mà bit MSB bị thay đổi so với trước khi dịch, cờ này không được xác định sau nhiều lần dịch.

Lệnh này cập nhật các cờ SF, ZF, PF. Trong đó PF chỉ có ý nghĩa khi toán hạng là 8 bit; cờ AF không xác định.

SAR - Shift Arithmetically Right (Dịch phải số học).

Viết lệnh: SAR Đích, CL

Mô tả:

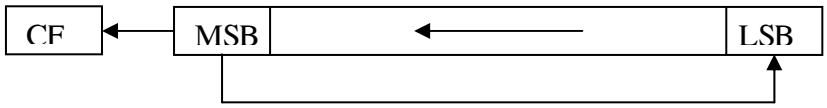


Sau mỗi lần dịch phải, MSB được giữ nguyên (nếu đây là bit dấu thì dấu luôn không đổi sau các lần dịch. Còn LSB được đưa vào cờ CF, CL phải được chứa sẵn số lần dịch mong muốn. Kiểu dịch này tương đương với một lần chia cho 2của số có dấu. Vì vậy có thể thay phép chia cho 2

ROL - Rotate All Bit to the Left (Quay vòng sang trái).

Viết lệnh: ROL Đích, CL

Mô tả:



Lệnh này dùng để quay toán hạng sang trái, MSB sẽ được đưa qua cờ CF và LSB. CL phải chứa số lần quay mong muốn.

Sau lệnh ROL cờ CF mang giá trị cũ của MSB, vì vậy lệnh này cò dùng để tạo cờ CF từ giá trị của MSB làm điều kiện cho các lệnh nhảy có điều kiện. Còn cờ OF ← 1 nếu sau khi dịch 1 lần mà bit MSB bị thay đổi so với trước khi dịch, cờ này không được xác định sau nhiều lần dịch. Lệnh này tác động vào các cờ CF, OF.

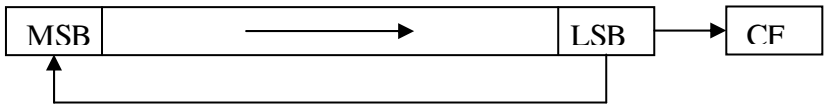
Ví dụ:

```
ROL BX, 1          ; quay vòng sang trái thanh ghi BX.
MOV CL, 4          ; đặt số lần quay vào thanh ghi CL.
ROL AL, CL  ; quay vòng sang trái thanh ghi AL 4 lần.
```

ROR - Rotate All Bit to the Right (Quay vòng sang phải).

Viết lệnh: ROR Đích, CL

Mô tả:



Lệnh này dùng để quay toán hạng sang phải, LSB sẽ được đưa qua cờ CF và MSB. CL phải chứa số lần quay mong muốn.

II.4. Nhóm lệnh so sánh.

CMP - Compare Byte or Word (so sánh 2 byte hay 2 từ).

Viết lệnh: CMP Đích, Gốc.

Trong đó toán hạng đích và gốc có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chữa dữ liệu có cùng độ dài và không được phép đồng thời là 2 ô nhớ.

Lệnh này chỉ tạo các cờ, không lưu kết quả so sánh; sau lệnh so sánh, các toán hạng không bị thay đổi. lệnh này thường được dùng để tạo cờ cho các lệnh nhảy có điều kiện.

Các cờ chính theo quan hệ đích và nguồn khi so sánh 2 số không dấu:

CF ZF

Đích = Nguồn 0 1
Đích > Nguồn 0 0
Đích < Nguồn 1 0.

TEST - And Operands to Update Flag (và 2 toán hạng để tạo cờ).

Viết lệnh: TEST Đích, Nguồn

Trong đó toán hạng đích và nguồn có thể tìm được theo các chế độ địa chỉ khác nhau, nhưng phải chứa dữ liệu cùng độ dài và không được phép đồng thời là 2 ô nhớ và cũng không được là thanh ghi đoạn. Sau lệnh này các toán hạng không bị thay đổi và kết quả không được lưu giữ. Các cờ được tạo ra sẽ được dùng làm điều kiện cho các lệnh nhảy có điều kiện. Lệnh này cũng có tác dụng che như một mặt nạ.

Tác động: Xoá: CF, OF
Cập nhật: PF, SF, ZF (PF chỉ liên quan đến 8 bit thấp)
Không xác định: AF.

Ví dụ:

```
TEST AH, AL          ; Và AH với AL để tạo cờ.
TEST AH, 01H                 ; Bit 0 của AH = 0?
TEST BP, [BX][DI]    ; Và BP với ô nhớ DS:BX+DI.
```

II.5 Các lệnh điều khiển chương trình.

- Lệnh nhảy không điều kiện: Lệnh này khiến bộ vi xử lý bắt đầu thực hiện một lệnh mới tại địa chỉ được mô tả trong lệnh.

Viết lệnh: JMP Nhãn

Lệnh mới bắt đầu tại địa chỉ ứng với nhãn. Chương trình dịch sẽ căn cứ vào vị trí nhãn để xác định giá trị dịch chuyển.

- Lệnh nhảy có điều kiện: Lệnh này biểu diễn thao tác: nhảy (có điều kiện) tới nhãn, tức là chỉ thực hiện nhảy tới nhãn nếu điều kiện chỉ ra đúng. Nhãn phải nằm cách xa (dịch đi một khoảng) -128.. +127 byte so với lệnh tiếp theo sau lệnh nhảy có điều kiện. Chương trình dịch sẽ căn cứ vào vị trí của nhãn để xác định giá trị dịch chuyển.

Các lệnh này không tác động đến cờ.

Người ta phân biệt các kiểu nhảy có điều kiện:

+ Nhảy theo kiểu không dấu:

JA/JNBE - Jump if Above/ Jump if Not Below or Equal.
Viết lệnh: JA Nhãn
JNBE Nhãn

JAE/JNB- Jump if Above or Equal/ Jump if Not Below.
Viết lệnh: JAE Nhãn
JNB Nhãn

JB/JNAE- Jump if Below/ Jump if Not Above or Equal.
Viết lệnh: JB Nhãn
JNAE Nhãn.

Ví dụ 1:

```
CMP AL, 10H; so sánh AL với 10H.
JA MEM1          ; nhảy đến nhãn MEM1 nếu AL cao hơn 10H.
JB MEM2          ;nhảy đến nhãn MEM2 nếu AL thấp hơn 10H.
```

+ Nhảy theo kiểu có dấu:

JG/JNLE- Jump if Greater than/ Jump if Not Less than or Equal.

Viết lệnh: JG Nhãn
JNLE Nhãn.

JGE/JNL- Jump if Greater than or Equal/ Jump if Not Less than.

Viết lệnh: JGE Nhãn
JNL Nhãn.

JL/JNGE- Jump if Less than/ Jump if Not Greater than or Equal.

JLE/JNG- Jump if Less than or Equal/ Jump if Not Greater than.

+ Nhảy theo kiểu đơn.

JE/JZ- Jump if Equal/ Jump if Zero.

JNE/JNZ- Jump if Not Equal/ Jump if Not Zero.

JC- Jump if Carry

JNC- Jump if Not Carry

JO- Jump if Overflow

JNO- Jump if Not Overflow

JS- Jump if Sign

JNS- Jump if Not Sign

JP/JPE- Jump if Parity/ Jump if Parity Even

JNP/JPO- Jump if Not Parity/ Jump if Parity Odd

- Lệnh lặp: Lệnh này dùng để lặp lại đoạn chương trình (bao gồm các lệnh nằm trong khoảng tử nhãn đến hết lệnh **LOOP Nhãn** cho đến khi số lần lặp CX=0. Điều này có nghĩa là trước khi vào vòng lặp, ta phải đưa số lần lặp mong muốn vào thanh ghi CX và sau mỗi lần thực hiện lệnh **LOOP Nhãn** thì CX tự động giảm đi 1.

Nhãn phải nằm cách xa (dịch một khoảng) -128 byte so với lệnh tiếp theo sau lệnh LOOP.

Lệnh này không tác động đến cờ.

Viết lệnh: LOOP Nhãn

Ví dụ:

```
        XOR AL, Al   ; xoá AL
        MOV CX, 16   ; số lần lặp đưa vào CX
Lap:    INC AL       ; tăng AL lên 1
        LOOP Lap          ; lặp lại 16 lần, AL =16.
```

- **Lệnh JCXZ**- Jump if CX is Zero (nhảy nếu CX = 0).

Viết lệnh: JCXZ Nhãn

Đây là lệnh nhảy có điều kiện tới nhãn nếu nội dung thanh đếm bằng 0 và không có liên hệ gì với cò ZF. Nhãn phải nằm cách xa (dịch đi một khoảng) -128.. +127 byte so với lệnh tiếp theo sau lệnh JCXZ. Chương trình dịch sẽ căn cứ vào vị trí nhãn để xác định giá trị dịch chuyển.

- Lệnh gọi chương trình con CALL: Lệnh này dùng để chuyển hoạt động của bộ vi xử lý từ chương trình chính (CTC) sang chương trình con (ctc). Nếu ctc ở cùng một đoạn mã với CTC thì ta có gọi gần. Nếu CTC và ctc nằm trong hai đoạn mã khác nhau thì ta có gọi xa. Gọi gần và gọi xa khác nhau về cách tạo địa chỉ trở về. Địa chỉ trở về là địa chỉ tiếp theo ngay sau lệnh CALL. Khi gọi gần thì chỉ cần cất IP của địa chỉ trở về, khi gọi xa thì phải cất cả CS và IP của địa chỉ trở về. Địa chỉ trở về được tự động cất vào ngăn xếp khi bắt đầu thựuc hiện lệnh gọi và được tự động lấy ra khi gặp lệnh trở về RET.

-RET - Return from Procedure to Calling Program (Trở về CTC từ ctc).

Viết lệnh: RET

Khi gặp lệnh trở về RET, vi xử lý kết thúc ctc lấy lại địa chỉ trở về, bao gồm địa chỉ IP (trường hợp gọi gần) hoặc IP và CS (trong trường hợp gọi xa) của lệnh tiếp theo sau lệnh CALL, được đặt trong ngăn xếp.

- INT - Interrupt Program Excution (Ngắt, gián đoạn chương trình đang chạy).

Viết lệnh: INT N, N = 0.. FFH

Mô tả: Các thao tác của bộ vi xử lý khi chạy lệnh INT :

1. SP ← SP - 2. {SP} ← FR
2. IF ← 0 (cấm các ngắt khác tác động), TF ← 0 (chạy suốt).
3. SP ← SP - 2, {SP} ← CS.
4. SP ← SP - 2, {SP} ← IP.
5. {N x 4} → IP, {5N x 4 + 2} → CS.

Mỗi lệnh ngắt ứng với một chương trình phục vụ ngắt khác nhau có địa chỉ lấy từ bảng véc tơ ngắt. Bảng này gômg 256 vec tơ, chứa địa chỉ của các chương trình phục vụ ngắt tương ứng và chiếm 1 Kb RAM có địa chỉ thấp nhất.

Ví dụ như các chương trình phục vụ ngắt của BIOS, của DOS như IO.SYS, MSDOS.SYS.

Ví dụ:

```
INT 21H
```

III. Các chế độ địa chỉ

Những phương pháp định địa chỉ hay còn gọi là chế độ địa chỉ (addressing mod) được dùng để vi xử lý tìm ra (định vị, addressing) các toán hạng cần thiết cho một lệnh nào đó. Một bộ vi xử lý có thể có nhiều chế độ địa chỉ, các chế độ địa chỉ này được xác định ngay từ khi chế tạo bộ vi xử lý và sau này không thể thay đổi được. Họ vi xử lý Intel có bảy chế độ địa chỉ như sau:

1. Chế độ địa chỉ thanh ghi
2. Chế độ địa chỉ tức thì
3. Chế độ địa chỉ trực tiếp
4. Chế độ địa chỉ gián tiếp thanh ghi
5. Chế độ địa chỉ tương đối cơ sở
6. Chế độ địa chỉ tương đối chỉ số
7. Chế độ địa chỉ tương đối chỉ số cơ sở

III.1. Chế độ địa chỉ thanh ghi (register addressing).

Trong chế độ địa chỉ này người ta dùng các thanh ghi bên trong CPU như là các toán hạng để chứa dữ liệu cần thao tác. Vì vậy khi thực hiện lệnh có thể đạt tốc độ truy nhập cao hơn so với các lệnh có truy nhập đến bộ nhớ.

Ví dụ:

```
MOV AX, BX          ; chuyển nội dung BX vào AX.
ADD  DS, DL ; cộng nội dung AL và DL , kết quả giữ trong AL.
```

III.2. Chế độ địa chỉ tức thì (immediate addressing)

Trong chế độ địa chỉ này toán hạng đích là một thanh ghi hay một ô nhớ, còn toán hạng nguồn là một *hằng số* và ta có thể tìm thấy toán hạng này ở ngay sau mã lệnh (chính vì vậy chế độ địa chỉ này gọi là chế độ địa chỉ tức thì). Ta có thể dùng chế độ này để nạp dữ liệu cần thao tác vào bất kỳ thanh ghi nào (trừ các thanh ghi đoạn và các thanh ghi cờ) hoặc vào bất kỳ ô nhớ nào trong đoạn dữ liệu DS.

Ví dụ:

```
MOV AX, 4EH             ; chuyển giá trị 4EH vào thanh ghi AX.
MOV AX, 0FFOH   ; chuyển 0FF0H vào thanh ghi AX
MOV DS, AX      ; để đưa vào DS.
MOV [BX], 4EH           ; chuyển 4EH vào địa chỉ ô nhớ DS:BX
```

III.3. Chế độ địa chỉ trực tiếp (direct addresssing mode)

Trong chế độ địa chỉ này một toán hạng chứa địa chỉ lệch của ô nhớ dùng để chứa dữ liệu, còn toán hạng kia chỉ có thể là một thanh ghi mà không thể là một vị trí nhớ.

Nếu so sánh với chế độ địa chỉ tức thì ta thấy ở đây ngay sau mã lệnh không phải là một toán hạng mà là một địa chỉ lệch của toán hạng. Xét về phương diện địa chỉ thì đó là địa chỉ trực tiếp.

Ví dụ:

```
MOV AL, [1234H]   ; chuyển nội dung ô nhớ DS:1234H vào AL.
MOV [4321H], CX   ; chuyển nội dung CX vào 2 vị trí nhớ
                          ; liên tiếp là DS:4321 và DS:4322.
```

III.4. Chế độ địa chỉ gián tiếp qua thanh ghi (register indirect addressing).

Trong chế độ địa chỉ này một toán hạng là một thanh ghi được sử dụng để chứa địa chỉ lệch của ô nhớ chứa dữ liệu, còn toán hạng kia chỉ có thể là một thanh ghi mà không được là ô nhớ.

Ví dụ:

```
MOV AL, [BX]            ; chuyển nội dung tại ô nhớ DS:BX vào AL.
MOV [SI], CL    ; chuyển nội dung CL vào ô nhớ DS:SI.
```

III.5. Chế độ địa chỉ tương đối cơ sở (based relative addresing).

Trong chế độ này các thanh ghi cơ sở như BX và BP và các hằng số biểu diễn các giá trị dịch chuyển (displacement values) được dùng để tính địa chỉ hiệu dụng của toán hạng trong các vùng nhớ DS và SS. Sự có mặt của các giá trị diạch chuyển xác định tính tương đối (so với cơ sở) của địa chỉ.

Ví dụ:

```
MOV CL, [BX] + 10  ; chuyển nội dung 2 ô nhớ liên tiếp có địa
                          ; chỉ DS:(BX+10) và DS:(BX+11) vào CX.
MOV CX, [BX +10]   ; tương tự như lệnh trên.
MOV AL, [BP] + 10  ; chuyển nội dung ô nhớ SS:(BP+10) vào AL
```

III. 6. Chế độ địa chỉ tương đối chỉ số (indexed relative addressing).

Trong chế độ địa chỉ này các thanh ghi chỉ số như SI và DI và các hằng số biểu diễn các giá trị dịch chuyển (displacement values) được dùng để tính địa chỉ của toán hạng trong vùng nhớ DS.

Ví dụ:

```
MOV AL, [SI]+10          ; chuyển nội dung ô nhớ DS:(SI+10) vào AL.
MOV AL, [SI+10]          ; tương tự như trên.
```

III.7. Chế độ địa chỉ tương đối chỉ số cơ sở (based indexed relative addressing).

Kết hợp hai chế độ địa chỉ chỉ số và cơ sở ta có chế độ địa chỉ chỉ số cơ sở. Trong chế độ địa chỉ này ta dùng cả thanh ghi cơ sở và thanh ghi chỉ số để tính địa chỉ của toán hạng. Nếu ta dùng thêm cả thành phần biểu diễn sự dịch chuyển của địa chỉ thì ta có chế độ địa chỉ phức hợp cho chế độ địa chỉ hoá các mảng hai chiều.

Ví dụ:

```
MOV AL, [BP][SI]+10      ; chuyển nội dung DS:(BX+SI+10) vào AL.
MOV AL, [BP+SI+10]       ; tương tự như trên.
```

Khi dùng thanh ghi chỉ số , thanh ghi cơ sở và thanh ghi con trỏ thì những cặp địa chỉ đoạn và địa chỉ lệch sau sẽ được định nghĩa trước:

CS:IP, DS:SI, DS:DI, DS:BX, ES:DI, SS:SP, SS:BP.

Muốn loại bỏ giá trị ngầm định cho BX trong thanh ghi đoạn DS và dùng giá trị trong thanh ghi đoạn ES ta cần viết:

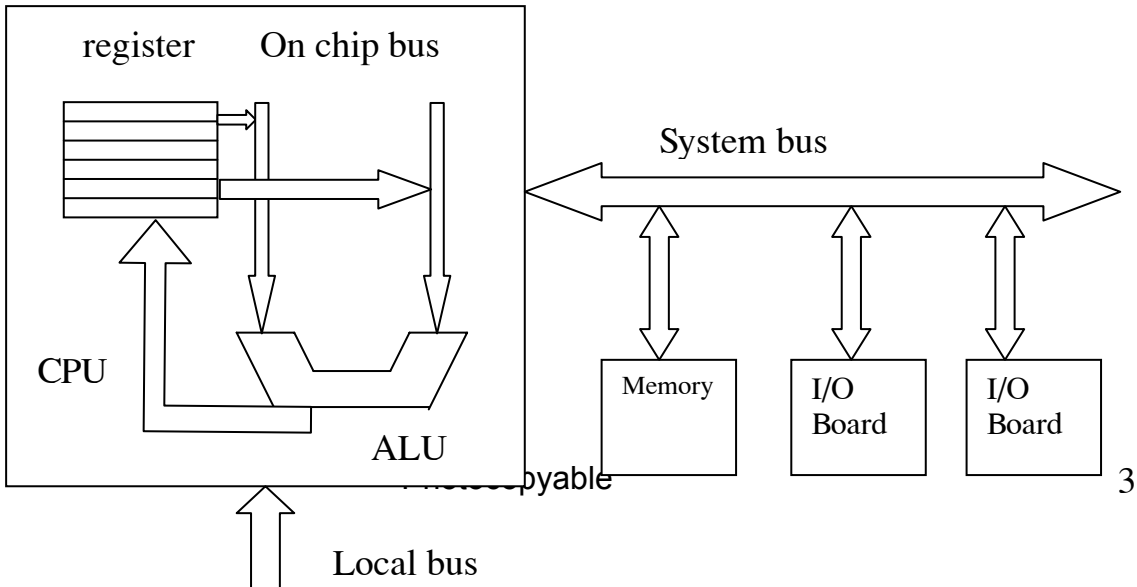
```
MOV AL, ES:[BX]    ; chuyển nội dung ES:BX vào AL.
```

CHƯƠNG V. CÁC BUS TRONG VI XỬ LÝ VÀ MÁY VI TÍNH

I. Chức năng và thông số của BUS

Một trong những hoạt động và chức năng cơ bản của máy tính là truyền số liệu (data transfer). Sự hoạt động của máy tính do các bộ vi xử lý điều khiển. Bộ vi xử lý và các chip hỗ trợ khác đến lượt mình cũng thường xuyên phải truyền số liệu giữa các khối, bộ phận trong và ngoài chúng với nhau.

Vì có rất nhiều các bộ phận , khối riêng rẽ trong bản thân các Chip và các đường truyền số liệu rất đa dạng, nên một cách hợp lý ta không thể thực hiện các đường nối giữa các bộ phận , khối từng đôi một với nhau mà ta nối chung tất cả các lối vào/ lối ra của các khối riêng rẽ với nhau lên một hệ thống các đường dẫn chung; hệ thống này được gọi là bus.



Hình 5.1. *Các bus trong một hệ thống máy tính.*

Các bộ phận, khối được nối lên bus phải thoả mãn một yêu cầu là có khả năng được cắt ra hoặc nối trở lại theo lệnh của điều khiển. Lúc một output được cắt ra khỏi bus, nó ở trạng thái trở kháng cao (High impedance, Hi-Z).

Quy tắc nghiêm ngặt của truyền số liệu là trong mỗi thời điểm, tối đa chỉ có một output được cấp số liệu lên bus.

Do trong mỗi thời điểm một output thường cần phải đồng thời cấp số liệu cho nhiều input, cho nên nó cần phải có khả năng phát ra (source) ở mức logic cao hoặc nuốt vào (sink) ở mức logic thấp, một dòng điện lớn tới vài chục mA cấp cho các input đó, đóng vai trò tải của output.

Thông số đặc trưng cho đường bus là trở kháng vào của nó (gồm có điện trở thuần và dung kháng). Thường điện trở thuần khoảng vài KΩ là thoả mãn yêu cầu của output, chỉ có dung kháng của bus gây khó khăn cho các thiết bị output, (vì nó cản trở tăng tốc độ biến thiên của các mức điện áp trên bus), do đó dung kháng được xem là thông số đặc trưng của bus.

Ví dụ xét trường hợp một bus có điện dung vào 100 pF. Nếu muốn tốc độ biến thiên điện áp trên bus là du/dt =2V/10ns thì thiết bị output phải nuốt được dòng điện điện dung là

$$i = dq/dt = C(du/dt) = 20 \text{ mA}.$$

Căn cứ theo cấu hình của các thiết bị nối vào bus, người ta phân chúng thành 3 nhóm như sau:

- Output cấp số liệu cho bus.
- Input nhận số liệu từ bus.
- In/ Out khi là input, khi là output.

II. BUS trong máy vi tính.

II.1. Bus trong vi xử lý và bus bộ xử lý

Trong các bộ vi xử lý có một hệ thống các bus dùng để truyền số liệu, lệnh, các tín hiệu điều khiển ,... , giữa các khối bên trong của nó. Ngoài ra có một hệ thống các bus đưa ra ngoài qua các chân của nó. Các đường bus trong được điều khiển bởi khối điều khiển tuỳ thuộc hoặc vào nội dung lệnh được giải mã hoặc theo các điều khiển ngắt của bên ngoài đưa vào vi xử lý. Các đường bus này hoạt động theo nhịp của một clock bên trong vi xử lý.

Xét với ví dụ các đường bus trong kiến trúc của vi xử lý 8088 như đã giới thiệu trong chương III.

Các bus trong vi xử lý truyền số liệu giữa các khối với nhau, có hai loại đường truyền, một chiều và hai chiều. Hệ các đường bus nối với các bộ phận, khối bên ngoài vi xử lý gồm 20 đường địa chỉ (AD0 - AD 19), 8 đường số liệu ( ), và các đường thuộc bus điều khiển.

Chính khối điều khiển phát các tín hiệu điều khiển các bus.

Bus bộ vi xử lý là đường truyền dẫn giữa CPU và các chip hỗ trợ trung gian. Những chip hỗ trợ này được gọi là bộ chip (chip set). Bus này dùng để truyền dữ liệu giữa CPU và bus hệ thống chính hoặc giữa CPU và cache ngoài.

Vì mục đích của bus bộ xử lý để gửi hoặc nhận thông tin từ CPU với tốc độ nhanh nhất có thể, nên bus này hoạt động nhanh hơn nhiều so với bất kỳ bus nào khác trong hệ thống và đảm bảo tránh hiện tượng tắc nghẽn ở đây. Bus bộ xử lý bao gồm bus dữ liệu, bus địa chỉ và bus điều khiển. Trong một hệ thống thiết kế cho VXL Pentium, bus bộ xử lý có 64 đường dữ liệu, 32 đường địa chỉ. Pentium Pro và Pentium II có 36 đường địa chỉ.

Bus bộ xử lý hoạt động ở tốc độ đồng hồ cơ sở giống như CPU chạy ngoại trú. Ví dụ Pentium II 333MHz chạy ở tốc độ đồng hồ 333MHz nội trú nhưng chỉ ở 66,6 MHz ngoại trú.

Tốc độ truyền của bus bộ xử lý được xác định bằng cách nhân độ rộng dữ liệu với tốc độ đồng hồ cơ sở rồi chia cho 8.

Khi thiết kế các bộ vi xử lý, có thể tuỳ ý lựa chọn loại bus bên trong vi xử lý, còn với các bus liên hệ với bên ngoài cần phải xác định rõ các quy tắc làm việc cũng như các đặc điểm kỹ thuật về điện và cơ khí để người thiết kế Main Board có thể ghép nối vi xử lý với các thiết bị khác. nói cách khác, các bus này phải tuân theo một chuẩn nhất định. Tập các quy tắc của chuẩn còn được gọi là nghi thức bus (bus protocol).

Trong thế giới máy tính có rất nhiều loại bus khác nhauđược sử dụng, các bus này nói chung là không tương thích với nhau. Sau đây là một số loại bus được dùng phổ biến:

Tên bus	**Lĩnh vực áp dụng**
- Camac	Vật lý hạt nhân
- EISA	Một sô hệ thống dùng bộ VXL 8036
- IBM PC, PC/AT	Máy tính IBM PC, IBM/PC/AT
- Massbus	Máy PDP - 1 và VAX
- Microchannel	Máy PS/2
- Multibus I	Một số hệ thống có VXL 8088, 8086
- Multibus II	Một số hệ thống có VXL 80386
- Versabus	Một số hệ thống dùng VXL Motorola
- VME	Một số hệ thống dùng VXL 68x0 của Motorola.

Người ta thường phân loại bus theo ba cách sau:

1. Theo tổ chức phần cứng (như các loại bus nêu trên)
2. Theo nghi thức truyền thông (bus đồng bộ và không đồng bộ).
3. Theo loại tín hiệu truyền trên bus (bus địa chỉ, bus dữ liệu ...)

Sự làm việc của các bus

Thường có nhiều thiết bị nối với bus, một số là thiết bị tích cực và có thể đòi hỏi truyền thông tin trên bus, trong khi đó lại có các thiết bị thụ động chờ các yêu cầu từ các thiết bị khác. Các thiết bị tích cực được gọi là chủ bus (master), còn các thiết bị thụ động là tớ (slave).

Khi CPU ra lệnh cho bộ điều khiển đĩa đọc/ ghi một khối dữ liệu thì CPU là master cò bộ điều khiển đĩa là slave. Tuy nhiên khi bộ điều khiển đĩa ra lệnh cho bộ nhớ nhận dữ liệu mà nó đọc từ đĩa thì nó lại giữ vai trò của master.

Bus Driver và Bus Receiver.

Tín hiệu điện mà các thiết bị trong máy tính phát ra thường không đủ mạnh để điều khiển được bus, nhất là khi bus khá dài và có nhiều thiết bị nối với nó. Chính vì vậy mà hầu hết các bus master được nối với bus thông qua một chip được gọi là bus driver, về căn bản đó là bộ khuyếch đại tín hiệu số. Tương tự như vậy, hầu hết các slave bus được nối với bus thông qua bus receiver. Đối với các thiết bị có thể khi thì đóng vai trò master, khi thì đóng vai trò slave, người ta sử dụng

một chip kết hợp, gọi là transceiver. Các chip này đóng vai trò ghép nối và thường là các thiết bị 3 trạng thái, cho phép có thể ở trạng thái thứ ba: hở mạch (còn gọi là thả nổi).

Giống như MPU, bus có các đường địa chỉ, đường số liệu và đường điều khiển. Tuy nhiên không nhất thiết phải có ánh xạ một - một giữa các tín hiệu ở các chân ra của MPU và các đường dây của bus.

Những vấn đề quan trọng nhất liên quan đến thiết kế bus là: Nhịp đồng hồ bus (sự phân chia thời gian, hay còn gọi là bus cloking), cơ chế trọng tài bus (bus arbitration), xử lý ngắt và xử lý lỗi.

Các bus có thể được chia theo nghi thức truyền thông tin thành hai loại riêng biệt là bus đồng bộ và bus không đồng bộ phụ thuộc vào việc sử dụng nhịp đồng hồ bus.

II.2. Bus đồng bộ (Synchronous bus)

Bus đồng bộ có một đường dây điều khiển bởi một bô dao động thạch anh, tín hiệu trên đường dây này có dạng sóng vuông, với tần số thường nằm trong khoảng 5MHz - 50 MHz. Mọi hoạt động bus xảy ra trong một số nguyên lần chu kỳ này và được gọi là chu kỳ bus.

Giản đồ thời gian của một bus đồng bộ với tần số đồng hồ là 4MHz, như vậy chu kỳ bus là 250nS.

- T1 bắt đầu bằng sườn lên của tín hiệu đồng hồ Φ, trong một phần thời gian của T1, MPU đặt địa chỉ của byte cần đọc lên bus địa chỉ. Sau khi tín hiệu địa chỉ được thiết lập giá trị mới, MPU đặt các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ tích cực. Tín hiệu $\overline{MREQ}$ (memory request, truy cập bộ nhớ) chứ không phải thiết bị I/O; còn tín hiệu $\overline{RD}$ (Read) chọn Read.

- T2 là thời gian cần thiết để bộ nhớ giải mã địa chỉ và đưa dữ liệu lên bus dữ liệu.

- T3 tại sườn xung xuống của T3, MPU nhận dữ liệu trên bus dữ liệu, chứa vào thanh ghi bên trong MPU và chốt dữ liệu. Sau đó MPU đảo các tín hiệu $\overline{MREQ}$ và $\overline{RD}$.

Như vậy đã kết thúc một thao tác đọc, tại chu kỳ máy tiếp theo MPU có thể thực hiện một thao tác khác.

- T_{AD} : theo giản đồ thời gian, $T_{AD} \leq 110ns$, đây là thông số do nhà sản xuất đảm bảo, MPU sẽ đưa ra tín hiệu địa chỉ không chậm hơn 110ns tính từ thời điểm giữa sườn lên của T1.

- T_{DS} : Giá trị nhỏ nhất là 50ns, thông số này cho phép dữ liệu được đưa ra ổn định trên bus dữ liệu ít nhất là 50ns trước thời điểm giữa sườn xuống của T3. Yêu cầu về thời gian này đảm bảo cho MPU đọc dữ liệu liệu tin cậy.

Khoảng thời gian bắt buộc đối với T_{AD} và T_{DS} cũng nói lên rằng, trong trường hợp xấu nhất, bộ nhớ chỉ có 250 + 250+ 125 - 110 - 50 = 465ns tính từ thời điểm có tín hiệu địa chỉ cho tới khi nó đưa dữ liệu ra bus địa chỉ. Nếu bộ nhớ không đáp ứng đủ nhanh, nó cần phải phát tín hiệu xin chờ $\overline{WAIT}$ trước sườn xuống của T2. Thao tác này đưa thêm vào một trạng thái chờ (wait state), khi bộ nhớ đã đưa ra dữ liệu ổn định, nó sẽ đảo tín hiệu $\overline{WAIT}$ thành $WAIT$.

- T_{ML}: Đảm bảo rằng tín hiệu địa chỉ sẽ được thiết lập trước tín hiệu $\overline{MREQ}$ ít nhất là 60ns. Khoảng thời gian này là quan trọng nếu tín hiệu $\overline{MREQ}$ điều khiển sự tạo ra tín hiệu chọn chip CS, bởi vì một số chip nhớ đòi hỏi phải nhận được tín hiệu địa chỉ trước tín hiệu chọn chip. ***Như vậy không thể chọn chip nhớ với thời gian thiết lập là 75ns.***

- T_M, T_{RL}: Các giá trị bắt buộc đối với 2 đại lượng này có ý nghĩa là cả hai tín hiệu $\overline{MREQ}$ và $\overline{RD}$ sẽ là tích cực trong khoảng thời gian 85ns tính từ thừi điểm xuống của xung đồng hồ T1. Trong trường hợp xấu nhất, chip nhớ chỉ có 250 + 250 - 85- 50 = 365ns sau khi hai tín hiệu trên là tích cực để đưa dữ liệu ra bus. Sự bắt buộc về thời gian này bổ sung thêm sự bắt buộc thời gian với tín hiệu đồng hồ.

- T_{MH}, T_{RH}: Hai đại lượng này cho biết cần có bao nhiêu thời gian để các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ sẽ được đảo sau khi dữ liệu đã được MPU đọc vào.

- T_{DH}: Cho biết bộ nhớ cần phải lưu dữ liệu bao lâu trên bus sau khi tín hiệu $\overline{RD}$ đã đảo.

Block Transfer, truyền tải khối dữ liệu.

Ngoài các chu kỳ đọc/ ghi, một số bus đồng bộ còn hỗ trợ truyền dữ liệu theo khối. Khi một thao tác đọc/ ghi bắt đầu, bus master báo cho slave biết có bao nhiêu byte cần truyền đi, sau đó slave sẽ liên tục đưa ra mỗi chu kỳ một byte, cho đến khi đủ số byte được thông báo. Như vậy, khi đọc dữ liệu theo khối, n byte dữ liệu cần n+2 chu kỳ, thay cho 3n chu kỳ như trước.

Cách khác làm cho bus truyền dữ liệu nhanh hơn là làm cho các chu kỳ ngắn lại. Trong ví dụ trên, mỗi byte được truyền đi trong 750ns, vậy bus có dải thông là 1.33MBs. Nếu xung đồng hồ là 8MHz, thời gian một chu kỳ chỉ còn một nửa, giải thông sẽ là 2.67MBs.

Tuy vậy việc giảm chu kỳ bus dẫn đến các khó khăn về mặt kỹ thuật, các bit tín hiệu truyền trên các đường dây khác nhau trong bus không phải luôn có cùng vận tốc, dẫn đến một hiệu ứng, gọi là **bus skew**.

Khi nghiên cứu về bus cần phải quan tâm đến vấn đề tín hiệu tích cực nên là mức thấp hay mức cao. Điều này tuỳ thuộc vào người thiết kế bus xác định mức nào là thuận lợi hơn.

Bảng 5.1. *Giá trị của một số thông số thời gian*

Ký hiệu	Tham số	Min	Max
T_{AD}	Thời gian trễ của tín hiệu địa chỉ		110
T_{ML}	Thời gian địa chỉ ổn định trước tín hiệu $\overline{MREQ}$	60	
T_{M}	Thời gian trễ của $\overline{MREQ}$ so với sườn xuống của tín hiệu đồng hồ T1		85
T_{RL}	Thời gian trễ của $\overline{RD}$ so với sườn xuống của tín hiệu đồng hồ T1		85
T_{DS}	Thời gian thiết lập dữ liệu trước sườn xuống của tín hiệu đồng hồ T3	50	
T_{MH}	Thời gian trễ của $\overline{MREQ}$ so với sườn xuống của tín hiệu đồng hồ T3		85
T_{RH}	Thời gian trễ của $\overline{RD}$ so với sườn xuống của tín hiệu đồng hồ T3		85
T_{DH}	Thời gian lưu trữ dữ liệu từ lúc đảo tín hiệu $\overline{RD}$	0	

II.3. Bus không đồng bộ (asynchronous bus).

Bus không đồng bộ không sử dụng một xung đồng hồ định nhịp. Chu kỳ của nó có thể kéo dài tuỳ ý và có thể khác nhau đối với các cặp thiết bị trao đổi tin khác nhau.

Làm việc với bus đồng bộ dễ dàng hơn do nó được định thời một cách gián đoạn, tuy vậy chính đặc điểm này cũng dẫn đến nhược điểm. Thứ nhất là: mọi công việc được tiến hành trong những khoảng thời gian là bội số nhịp đồng hồ bus, nếu một thao tác nào đó của CPU hay bộ nhớ có thể hoàn thành trong 3,2 chu kỳ thì nó sẽ phải kéo dài thành 4 chu kỳ. Điều hạn chế lớn nữa là đã chọn chu kỳ bus và đã xây dựng bộ nhớ, I/O Card cho bus này thì khó có thể tận dụng được được những tiến bộ của công nghệ. Chẳng hạn sau khi đã xây dựng bus với sự định thời như trên, công nghệ mới đưa ra các chip CPU và chip nhớ có thời gian chu kỳ là 100ns (thay cho 250ns như cũ), chúng vẫn cứ phải chạy với tốc độ thấp như các CPU và chip nhớ loại cũ, bởi vì nghi thức bus đòi hỏi chip nhớ phải đưa ra dữ liệu và ổn định dữ liệu ngay trước thời điểm ứng với sườn xuống của T3. Nếu có nhiều thiết bị khác nhau nối với một bus, trong đó có một số thiết bị có thể hoạt động nhanh hơn các thiết bị khác thì cần phải đặt bus hoạt động phù hợp với thiết bị chậm nhất.

Bus không đồng bộ ra đời nhằm khắc phục các nhược điểm của bus đồng bộ. Hình 5.3 minh hoạ sự hoạt động của bus không đồng bộ, trong đó master yêu cầu đọc bộ nhớ.

Trước hết master cần phát ra địa chỉ nhớ mà nó muốn truy cập, sau đó phát tín hiệu $\overline{MREQ}$ tích cực để báo rằng nó muốn truy cập bộ nhớ chứ không phải cổng I/O. Tín hiệu này là cần thiết vì bộ nhớ và các cổng I/O đều có thể dùng chung một miền địa chỉ. Tiếp theo master phải phát tín hiệu $\overline{RD}$ tích cực để bên slave biết rằng master sẽ thực hiện thao tác đọc chứ không phải là thao tác ghi.

Các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ được đưa ra sau tín hiệu định địa chỉ bao lâu tuỳ thuộc vào tốc độ của master. Sau khi hai tín hiệu này đã ổn định, master sẽ phát tín hiệu đặc biệt, là $\overline{\text{MSYN}}$ (Master SYNchronization) ở mức tích cực để báo cho slave biết rằng các tín hiệu cần thiết đã sẵn sàng trên bus, slave có thể nhận lấy. Khi slave nhận các tín hiệu này, nó sẽ thực hiện công việc với tốc độ nhanh nhất có thể được (nhanh chóng đưa dữ liệu của ô nhớ yêu cầu lên bus dữ liệu). Khi hoàn thành, slave sẽ phát tín hiệu $\overline{\text{SSYN}}$ (Slave SYNchronization) tích cực.

Khi master nhận được tín hiệu $\overline{\text{SSYN}}$ tích cực, nó biết rằng dữ liệu của slave đã sẵn sàng và thực hiện việc chốt dữ liệu, sau đó đảo các đường địa chỉ cũng như các tín hiệu $\overline{MREQ}$ và $\overline{RD}$ và $\overline{\text{MSYN}}$.

Khi slave nhận được sự đảo tín hiệu $\overline{\text{MSYN}}$ thành không tích cực, nó biết rằng một chu kỳ đã kết thúc và đảo tín hiệu $\overline{\text{SSYN}}$. Bây giờ bus lại trở lại trạng thái ban đầu, mọi tín hiệu đều là không tích cực, tất cả sẵn sàng chờ bus master mới.

Trên giản đồ thời gian của bus không đồng bộ, ta sử dụng mũi tên để thể hiện nguyên nhân và kết quả. Việc đưa $\overline{\text{MSYN}}$ lên mức tích cực dẫn đến việc truyền dữ liệu ra bus dữ liệu và đồng thời cũng dẫn đến việc slave phát ra tín hiệu $\overline{\text{SSYN}}$ tích cực. Đến lượt mình, tín hiệu $\overline{\text{SSYN}}$ lại gây ra sự đảo mức của các đường địa chỉ, $\overline{MREQ}$ và $\overline{RD}$ và $\overline{\text{MSYN}}$. Cuối cùng sự đảo mức của $\overline{\text{MSYN}}$ lại gây ra sự đảo mức tín hiệu $\overline{\text{SSYN}}$ và kết thúc một chu kỳ đọc.

Full handshake.

Tập các tín hiệu phối hợp với nhau như vậy được gọi là Full handshake, nó chủ yếu gồm có 4 sự kiện sau:

1. $\overline{\text{MSYN}}$ được đặt lên mức tích cực.
2. $\overline{\text{SSYN}}$ được đặt tích cực để đáp lại tín hiệu $\overline{\text{MSYN}}$
3. $\overline{\text{MSYN}}$ được đảo để đáp lại tín hiệu $\overline{\text{SSYN}}$
4. $\overline{\text{SSYN}}$ được đảo để đáp lại tín hiệu $\overline{\text{MSYN}}$ thành không tích cực.

Ta có thể nhận thấy Full handshake là quan hệ nhân quả, độc lập với thời gian. Nếu một cặp master-slave nào đó hoạt động chậm hoặc thời gian bị kéo dài thì cặp master-slave kế tiếp không hề bị ảnh hưởng.

Tuy ưu điểm của bus không đồng bộ rất rõ ràng, nhưng trong thực tế phần lớn các bus đang được sử dụng là loại bus đồng bộ. Lý do căn bản là các hệ thống sử dụng bus đồng bộ là dễ thiết kế hơn. CPU chỉ cần chuyển các mức tín hiệu cần thiết sang trạng thái tích cực là các chip nhớ đáp ứng ngay, không cần tín hiệu phản hồi. Chỉ cần các chip được chọn phù hợp thì mọi hoạt động đều trôi chảy.

III. Trọng tài bus (bus arbitration).

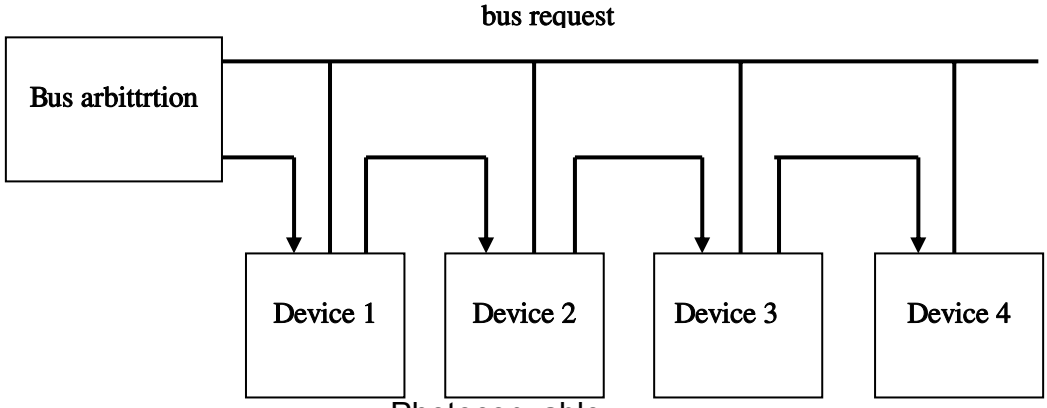
Trong hệ thống máy tính không phải chỉ có CPU làm bus master, thực tế các chip I/O cũng có lúc phải làm chủ bus để có thể đọc hoặc ghi vào bộ nhớ và để gọi ngắt; các bộ đồng xử lý cũng có thể làm chủ bus. Như vậy cần phải giải quyết vấn đề tranh chấp khi có từ hai thiết bị trở lên đồng thời muốn làm chủ bus. Để giải quyết vấn đề này cần có một cơ chế trọng tài để tránh sự xung đột. Cơ chế trọng tài có thể là tập trung hoặc không tập trung.

III.1 Trọng tài bus tập trung

Hình 5.4 là một ví dụ đơn giản về trọng tài bus tập trung. ở đây, một trọng tài bus duy nhất sẽ quyết định thiết bị nào được là chủ bus tiếp theo. Nhiều bộ VXL có đơn vị trọng tài bus được thiết kế ngay trong chip VXL, trong một số máy tính mini, đơn vị trọng tài bus nằm ngoài CPU.

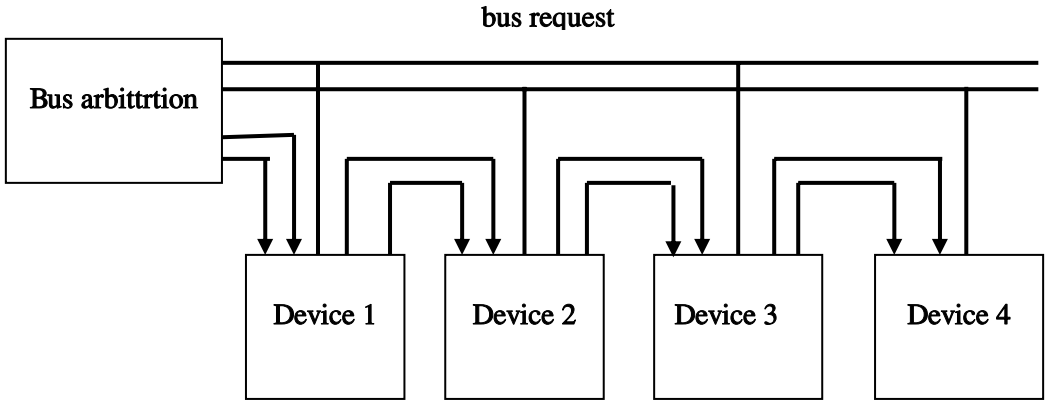
Theo cơ chế này, trọng tài chỉ có thể biết là có yêu cầu chiếm dụng bus hay không, chứ không biết có bao nhiêu đơn vị muốn chiếm bus. Khi trọng tài bus nhận được một yêu cầu, nó sẽ phát ra một tín hiệu cho phép trên đường dây bus grant (cho dùng bus). Đường dây này nối qua tất cả các thiết bị vào/ ra theo kiểu nối tiếp.

Khi thiết bị nằm gần trọng tài nhất nhận được tín hiệu cho phép, nó sẽ kiểm tra xem có phải chính nó đã phát yêu cầu chiếm bus không? Nếu đúng thì nó sẽ chiếm lấy bus và không truyền tiếp tín hiêuh cho phép trên đường dây. Nếu nó kiểm tra thấy không phải là yêu cầu của mình thì tiếp tục truyền tín hiệu cho phép tới thiết bị kế tiếp trên đường dây.



Hình 5.4. *Trọng tài bus tập trung có một mức, mắc nối tiếp.*

Một số loại bus có nhiều mức độ ưu tiên, với mỗi mức ưu tiên có một đường dây yêu cầu bus và một đường dây cho chiếm bus. Hình 5.5 là một ví dụ về bus có hai mức (các bus trong thực tế thường có 4, 8 hay 16 mức). Mỗi thiết bị trong hệ thống máy tính nối với một trong các mức yêu cầu bus, các thiết bị thường được sử dụng hơn được gắn với đường dây có mức ưu tiên cao hơn.



Hình 5.5. *Trọng tài bus tập trung có hai mức, mắc nối tiếp.*

Nếu có một số thiết bị ở các mức ưu tiên khác nhau cùng yêu cầu, trọng tài bus sẽ chỉ phát tín hiệu cho phép đối với yêu cầu có mức ưu tiên cao nhất. Trong số các thiết bị có cùng mức ưu tiên, thiết bị gần trọng tài bus hơn sẽ có quyền ưu tiên cao hơn.

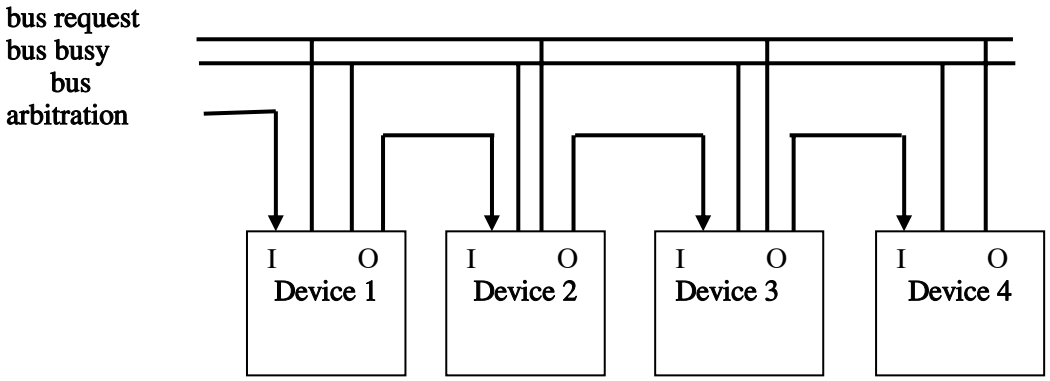
Một số trọng tài bus có đường dây thứ ba nối tới các thiết bị để các thiết bị xác nhận việc nhận được tín hiệu cho phép và chiếm dụng bus, gọi là đường dây biên nhận acknowledgement (ACK). Ngay sau khi một thiết bị phát tín hiệu tích cực trên đường dây ACK, trọng bus có thể đảo tín hiệu trên các đường dây trên các đường dây yêu cầu bus và cho phép dùng bus thành mức không tích cực. Kết quả là các thiết bị khác có thể đòi hỏi chiếm dụng bus trong khi thiết bị đầu tiên đang dùng bus. Khi kết thúc phiên làm việc hiện thời, bus master kế tiếp sẽ được lựa chọn. Cách làm việc như vậy làm tăng hiệu quả sử dụng bus, nhưng cần thêm một đường truyền tín hiệu ACK và cấu trúc của các thiết bị cũng phức tạp hơn. Các chip của Motorola sử dụng các bus loại này.

III.2 Trọng tài bus không tập trung

Trong cơ chế trọng tài bus không tập trung, không cần sử dụng một đơn vị riêng làm trọng tài bus, nhờ vậy giảm được giá thành phần cứng. Trong một số loại máy tính khác nhau, người ta đã sử dụng một vài kiểu trọng tài bus theo cơ chế này.

Trọng tài bus không tập trong trong multibus

Trong Multibus, người ta cho phép có thể lựa chọn cơ chế trọng tài bus không tập trung hoặc không tập trung, cơ chế trọng tài bus không tập trung được thực hiện theo sơ đồ trên hình 5.6



Hình 5.6. *Trọng tài bus không tập trung trong Multibus.*

Người ta chỉ sử dụng 3 đường dây, không phụ thuộc vào số lượng thiết bị nối với bus. Bao gồm:

+ Yêu cầu chiếm dụng bus (bus request)
+ Trạng thái bus (bus busy), được bus master đặt ở mức tích cực
+ Trọng tài bus, được mắc nối tiếp qua các thiết bị

Khi không có thiết bị nào yêu cầu chiếm bus, đường dây trọng tài bus truyền mức tích cực tới tất cả các thiết bị. Khi một đơn vị nào đó muốn chiếm dụng bus, đầu tiên nó kiểm tra bus có rỗi không và kiểm tra đầu vào của đường trọng tài bus, nếu thấy có điện áp IN = 5V thì nó có thể xin bus bằng cách đưa tín hiệu yêu cầu bus (Request) và xoá tín hiệu OUT, tức là đặt OUT = 0V. Do đó các thiết bị ưu tiên thấp hơn sẽ không xin được bus. Lúc này nó trở thành bus master.

IV. Xử lý ngắt

Một chức năng quan trọng của bus là xử lý ngắt. Khi CPU ra lệnh cho một thiết bị trong máy tính thực hiện việc đọc, ghi hay xử lý tin, nó thường chờ đợi tín hiệu ngắt do thiết bị I/O phát ra khi hoàn thành công việc được CPU yêu cầu. Khi nhận được tín hiệu ngắt, CPU đáp ứng ngay, đó có thể là việc nhận dữ liệu do thiết bị I/O chuyển về, cũng có thể là việc tiếp tục gửi dữ liệu tới thiết bị I/O hoặc CPU sử dụng bus cho một thao tác khác ... . Như vậy chính ngắt phát ra tín hiệu yêu cầu bus.

Vì có thể có nhiều thiết bị ngoại vi cùng phát tín hiệu ngắt, cho nên cũng cần có một cơ chế trọng tài giống như đối với các bus thông thường. Giải pháp thường dùng là gán các mức độ ưu tiên cho các thiết bị và sử dụng một trọng tài tập trung để trao quyền ưu tiên cho các thiết bị và sử dụng một trongh tài tập trung để trao quyền ưu tiên cho các thiết bị quan trọng thường xuyên được sử dụng.

V. Một số bus thông dụng

V.1 Bus IBM PC

Đây là ví dụ điển hình về một loại bus được sử dụng trong các hệ thống thương mại, nó được sử dụng rộng rãi trong các hệ thống vi xử lý dựa trên chip 8088. Hầu hết họ PC, kể cả các máy tương thích đều sử dụng bus này. Chính bus IBM PC tạo nên cơ sở cho bus IBM PC/AT và nhiều loại bus khác. Bus này có 62 đường dây, trong đó có 20 đường địa chỉ, 8 đường số liệu và các đường tín hiệu khác. .... Được liệt kê trong bảng 5.1

Về mặt vật lý, bus IBM PC được thiết kế ngay trên bo mạch chính và có khoảng gần chục đầu nối dạng khe cắm (slot) để cắm các card mở rộng, trên mỗi khe cắm có 62 chân được chia thành hai hàng.

Tín hiệu	Số dây	In	Out	giải thích
OSC	1		x	Chân dao động (14,31818 MHz)
CLK	1		x	Xung đồng hồ (4,77 MHHz)
RESET	1		x	Tín hiệu reset CPU và các thiết bị I/O
A0 - A9	20		x	Các đường dây địa chỉ
D0 - D7	8		x	Các đường truyền dữ liệu
ALE	1		x	Chốt địa chỉ
(MEMR)	1		x	Đọc bộ nhớ
(MEMW)	1		x	Ghi vào bộ nhớ
(IOR)	1		x	Đọc cổng I/O
(IOW)	1		x	Ghi ra cổng I/O
AEN	1	x		Adress ENable, yêu cầu bus địa chỉ
(IOCHCHK)	1	x		I/O Chanel Check
IOCHRDY	1	x		I/O Chanel Ready
IRQ2 - IRQ7	6	x		Các đường yêu cầu ngắt
DRQ1 - DRQ3	3	x		DMA Request
DACK0 - DACK3	4		x	DMA Acknowleage
T/C	1		x	Terminal/Count
Power	5			
GND	3			
Reserved	1			

Tín hiệu đồng hồ OSC và CLK

Các máy IBM PC đầu tiên sử dụng các phần tử dao động thạch anh, ở tần số 14,31818 MHz, tần số này được chọn phải thoả mãn việc tạo ra tín hiệu đồng bộ màu hệ NTSC. Tần số này cao hơn so với chuẩn 8088 (tần số cực đại là 5 MHz), do đó nó được chia ba thành 4,77MHz. Tần số 4,77 MHz được dùng làm đồng hồ chính để xác định chu kỳ bus. Tín hiệu tần số 4,77 MHz cũng có trên bus IBM PC và ký hiệu là CLK. Tín hiệu này không cân xứng như tín hiệu đông hồ 14,31818 MHz mà trong một chu kỳ bao gồm 2/3 thời gian ở mức thấp và 1/3 thời gian ở mức cao.

Tín hiệu RESET

Tín hiệu RESET trên bus do chip 8284A tạo ra. Để RESET CPU, các mạch điện bên ngoài gửi tín hiệu tới 8284A, nó sẽ đặt tín hiệu reset lên mức tchs cực, buộc CPU và các thiết bị I/O khở tạo lại.

Các đường địa chỉ và dữ liệu

CPU không nối trực tiếp với các đường địa chỉ và đương số liệu của bus, mà thông qua các chip khác. Các đường địa chỉ được chốt bằng cách dùng 3 chip 74LS373, mỗi chip là một bộ 8 thanh ghi chốt, tuy nhiên chỉ sử dụng 20 trong số 24 đường có thể.

Các đường dữ liệu sẽ được lấy mẫu (đọc nhanh giá trị) hoặc cung cấp giá trị trong những thời gian xác định, như trong sườn dương của một tín hiệu đồng hồ nào đó, vì vậy không cần chốt. Các đường dữ liệu của bus được điều khiển bởi bus transceiver (chip 74LS245). Chân DIR xác định hướng của tín hiệu là đi đến hay đi ra từ CPU.

Lý do chính của việc nối các chân của MPU với bên ngoài thông qua các bộ đệm chính là vì nó được chế tạo theo công nghệ MOS, CPU không có khả năng cung cấp đủ dòng để điều

khiển tất cả các phần tử nối với bus. Các chip làm bộ đệm dùng công nghệ TTL có khả năng cung cấp đủ dòng cho cả các thiết bị nối với bus.

Ngoài ra còn lý do khác là, khi có một thiết bị nào đó khác CPU muốn trở thành bus master (như DMAC), CPU cần phải thả nổi các bus. Phương pháp đơn giản nhất được áp dụng là thiết bị đó phải phát tín hiệu AEN (Address ENable) để đảo tín hiệu cho phép đưa ra trên các thanh ghi chốt và transceiver, làm cho các bus được thả nổi.

Tín hiệu ALE (Address Latch Enable)

Tín hiệu ALE được đặt mức tích cực khi CPU đang điều khiển các đường tín hiệu địa chỉ, cho phép các chip 74LS373 biết khi nào cần chốt địa chỉ lại. Tín hiệu này củ bus cũng cho bộ nhớ và các chip I/O biết khi nào các tín hiệu trên bus địa chỉ là hợp lệ.

Các đường tín hiệu $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$

Để điều khiển việc đọc/ ghi vào bộ nhớ hoặc các thiết bị vào/ra. Nhờ các tín hiệu này, bus cung cấp hai thông gian địa chỉ riêng biệt, một cho MEM và một cho I/O. Bộ nhớ sẽ không phản ứng khi thấy tín hiệu $\overline{\text{IOR}}$, $\overline{\text{IOW}}$ ở mức tích cực.

CPU sử dụng các tín hiệu $\overline{\text{S0}}$- $\overline{\text{S2}}$ đưa vào chip điều khiển bus 8288 để tạo ra các tín hiệu $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$ cùng với tín hiệu ALE. Chip điều khiển bus cũng nhận tín hiệu điều khiển ANE từ bus, tín hiệu này do một thiết bị muốn trở thành bus master đưa ra, khi nhận được tín hiệu ANE, chip điều khiển bus sẽ phát tín hiệu điều khiển các chip chốt địa chỉ nà chip bus transceiver thả nổi bus.

Tín hiệu $\overline{\text{IOCHCHK}}$ (I/O CHanel CHeck)

Tín hiệu này sẽ tích cực khi có lỗi chẵn /lẻ bị phát hiện trên bus. Tín hiệu này sẽ tác động mmọt ngắt NMI.

Tín hiệu IOCHRDY (I/O CHanel ReaDY)

Tín hiệu này do bộ nhớ đưa ra khi tốc độ hoạt động của nó thấp, yêu cầu CPU cho thêm một số chu kỳ để đợi, bằng cách chèn wait states vào các chu kỳ đọc/ghi bộ nhớ.

Các tín hiệu IRQ2-IRQ7.

Là các tín hiệu do các thiết bị ngoại vi đưa ra, đưa đến chip điều khiển ngắt 8259A. Khi có tín hiệu gửi đến chip điều khiển ngắt, nó sẽ kiểm soát các tín hiệu này và đưa ra một tín hiệu yêu cầu ngắt tới CPU và đặt số hiệu vectơ ngắt lên đường dữ liệu khi CPU cần đến. IRQ0 thường được mạch đồng hồ và IRQ1 được bàn phím sử dụng.

Các tín hiệu liên quan đến DMA

Các tín hiệu còn lại nói chung liên quan đến hoạt động DMA, chẳng hạn khi CPU yêu cầu ổ đĩa đọc một khối dữ liệu, mạch điều khiển ổ đĩa sẽ chờ nhận được byte đầu tiên từ ổ đĩa đưa ra, sau đó phát ra một yêu cầu trở thành bus master để ghi byte đó vào bộ nhớ.

Chip 8237A được INTEL thiết kế nhằm quản lý các nghi thức bus và thực hiện DMA trong đó có việc taưng địa chỉ bộ nhớ và giảm con đếm sau khi truyền mỗi byte. Việc này nó thực hiện thay cho các thiết bị I/O, giúp giảm giá thành của chúng.

Về căn bản, chip 8237A là một CPU nhỏ, có các chương trình được ghi sẵn bên trong. Khi 8088 muốn bắt đầu hoạt động DMA đối với một thiết bị ngoại vi nào đó, nó nạp số hiệu vào thiết bị, địa chỉ ô nhớ, số byte, hướng truyền và các thông tin khác vào các thanh ghi bên trong 8237A. Khi chip điều khiển đã sẵn sàng đọc hoặc ghi byte đầu tiên, nó đặt mức tích cực lên một trong các đường DRQ của bus để đưa vào chip 8237A. Khi nhận được tín hiệu, 8237A đòi chiếm bus và sẵn sàng truyền một byte. Chip 8237A phát tín hiệu $\overline{\text{DACK}}$ tới chip điều khiển báo cho

nó biết hãy ghi hoặc đọc byte của mình (trong thao tác đọc hoặc ghi tương ứng). Trong khoảng một chu kỳ này, chip 8237A điều khiển hoạt động của bus như một bus master.

Chip 8237A có 4 kênh độc lập và có thể quản lý đồng thời 4 đường truyền.

Tín hiệu T/C (Terminal/Count)

Đường T/C được chip 8237A đặt mức tích cực khi con đếm byte (byte count) bàng 0, báo cho bộ điều khiển I/O biết rằng công việc yêu cầu đã hoàn tất, đã đến lúc báo hiệu cho 8258A gọi ngắt tới CPU.

V.2. Bus IBM PC/AT

Bus IBM PC/AT là bước phát triển tiếp theo của thế hệ bus IBM PC nhằm phát huy được những khả năng hơn hẳn của bộ VXL 80286 so với 8088 trước nó. Với bus địa chỉ 24 dây, có khả năng đánh địa chỉ cho 2^{24} = 16MB bộ nhớ và có bus dữ liệu 16 bit.

Với giải pháp mở rộng PC bus, bổ sung thêm vào các khe cắm cũ một đoạn khe cắm ngắn, trên đó có 36 dây tín hiệu, tăng thêm cho bus địa chỉ 4 dây, bus dữ liệu 8 dây, các đường yêu cầu ngắt, kênh DMA, ... Nhờ vậy các card mở rộng trước đây vẫn dùng cho IBM PC có thể dùng cho IBM PC/AT.

Ngoài việc mở rộng bus, tần số tín hiệu đồng hồ bus cũng được tăng từ 4,77 MHz ở PC bus thành 8MHz, nhờ đó tốc độ truyền thông trên bus cũng tăng lên nhiều.

Năm 1991 tổ chức IEEE (Institute of Electrical and Electronic Engineers) đã đưa ra tiêu chuẩn quốc tế cho bus của máy AT, gọi là bus ISA (Industrial Standard Architecture)

Các nhà sản xuất PC khác đã đưa ra một chuẩn khác, đó là bus EISA (Extended ISA), về căn bản bus này là sự mở rộng bus PC/AT thành 32 bit, giữ nguyên tính tương thích với các máy tính và các card mở rộng đã có.

Ở thế hệ PS/2, thế hệ sau của IBM PC/AT một bus hoàn toàn mới được áp dụng, bus Micro chanel.

V.3. Bus PCI

Vào đầu năm 1992, Intel đã thành lập nhóm công nghệ mới. Nhằm nghiên cứu cải thiện các đặc tính kỹ thuật và những hạn chế của các bus hiện có như: bus ISA, bus EISA.

PCI (Peripheral Component Interconnect, liên kết các thành phần ngoại vi). Định chuẩn bus PCI đã được đưa ra vào tháng 6 năm 1992 và được cập nhật vào tháng 4 năm 1993, đã thiết kế lại bus PC truyền thống bằng cách bổ sung thêm một bus khác vào giữa CPU và bus I/O.

Bus PCI thường được gọi là bus mezzanine vì nó bổ sung thêm một tầng khác vào cấu hình bus truyền thống. PCI bỏ qua bus I/O tiêu chuẩn, nó sử dụng bus hệ thống để tăng tốc độ đồng hồ bus lên và khai thác hết lợi thế của đường dẫn dữ liệu của CPU.

Thông tin được truyền qua bus PCI ở 33MHz và độ rộng dữ liệu đầy đủ của CPU. Khi bus ấy được sử dụng để nối với CPU 32 bit, dải thông là 132 MBit/s, được tính theo công thức: 33MHz*32bit/8 = 132MBit/s. Khi bus ấy được sử dụng với những hệ thống bổ sung 64 bit, dải thông tăng gấp đôi, nghĩa là tốc đọ truyền dữ liệu đạt tới 264MBs. Lý do chíng mà bus PCI đã tăng tốc độ nhanh hơn các bus khác là nó có thể hoạt động đồng thời với bus vi xử lý. CPU có thể được xử lý dữ liệu trong các cache ngoại trú, trong khi bus PCI phải truyền thông tin liên tục giữa các thành phần khác của hệ thống, đây là ưu điểm thiết kế chính của bus PCI.

Định chuẩn PCI có ba cấu hình, mỗi cấu hình được thiết kế cho một kiểu hệ thống riêng biệt với những quy định nguồn riêng. Định chuẩn 5V cho những hệ thống máy tính văn phòng, định chuẩn 3,3V cho các hệ thống máy tính xách tay và những định chuẩn chung cho những bo mẹ và các cạc hoạt động trong hai kiểu ấy.

V.4. Bus nối tiếp chung USB

Bus USB (Universal Serial Bus) là một công nghệ bus mới đầy triển vọng, nhanh chóng phổ biến trong những thế máy tính ngày nay. Chủ yếu USB là cáp cho phép nối lên tới 127 thiết bị bằng cách sử dụng chuỗi xích. Tuy nhiên nó truyền dữ liệu không nhanh bằng FireWire, ở tốc độ 12MBs nó có khả năng đáp ứng cho hầu hết các thiết bị ngoại vi. Định chuẩn USB được đưa ra vào năm 1996 do một hội đồng gồm những đại diện của các nhà sản xuất máy tính lớn như Compaq, Digital, IBM, NEC và Northen Telecom.

Một ưu điểm nổi bật của USB là những thiết bị ngoại vi tự nhận dạng, một đặc trưng hết sức thuận lợi cho việc cài đặt, xác lập các thiết bị ngoại vi. Đặc trưng này hoàn toàn tương thích với những công nghệ PnP và cung cấp tiêu chuẩn công nghệ cho kết nối tương lai. Hơn nữa, những thiết bị USB có khả năng cắm nóng.

CHƯƠNG VI. KIẾN TRÚC BỘ NHỚ MÁY VI TÍNH

I. Các khái niệm chung

Một trong các hoạt động cơ bản của máy tính là lưu trữ dữ liệu dạng nhị phân. Các dữ liệu này là các chương trình hoặc số liệu mà Vi xử lý đưa ra hoặc đọc vào tuỳ theo yêu cầu. Bộ nhớ là các thiết bị để thực hiện nhiệm vụ lưu trữ dữ liệu của máy vi tính.

Mỗi ô nhớ được xác định bởi một địa chỉ. Thông thường mỗi ô nhớ có dung lượng là 1 byte. Các byte được ghép thành từ. Những máy 16 bit số liệu thì tổ chức 2 byte/từ, còn các máy 32 bit số liệu thì độ dài từ gấp đôi (4 byte/từ).

I.1. Trật tự các byte trong từ.

Có thể là từ phải sang trái (vi xử lý họ Intel) hoặc ngược lại từ trái sang phải (vi xử lý họ Motorola). Trường hợp dữ liệu lưu giữ là số nguyên thì hai cách sắp xếp trên không có trở ngại gì. Nhưng khi dữ liệu bao gồm cả số nguyên và cả xâu ký tự ... thì có vấn đề.

Ví dụ, xét một bản ghi (h 7.1) gồm có xâu là tên nhân viên BILL GATE và trường là số nguyên: tuổi 42. Xâu kết thúc bằng các byte 0 ở cuối để điền kín chỗ trống của từ, còn số nguyên thì được thêm vào các byte ở phần có trọng số cao hơn. Do vậy nếu dịch cách sắp xếp nọ sang cách kia của xâu giống như của số nguyên thì sẽ bị nhầm.

I.2. Mã phát hiện lỗi và sửa sai.

Số các vị trí bit khác nhau trong hai từ gọi là khoảng cách Hamming. Ví dụ, trong hai từ:
10001001 và 10110001 có khoảng cách Hammming bằng 3.

Để sửa sai, bên cạnh m số bit số liệu của từ, người ta thêm vào r bit dư (redundant bits) và chiều dài tổng của từ là n : $n = m + r$

Để phát hiện d bit lỗi đơn, cần dùng mã có khoảng cách d+1. Tương tự, để sửa lỗi d bit đơn, cần dùng mã có khoảng cách 2d+1. Ví dụ, dùng mã bit parity thêm vào byte số liệu, mã này có khoảng cách bằng 2, dùng để phát hiện 1 bit sai, nhưng không sửa được lỗi.

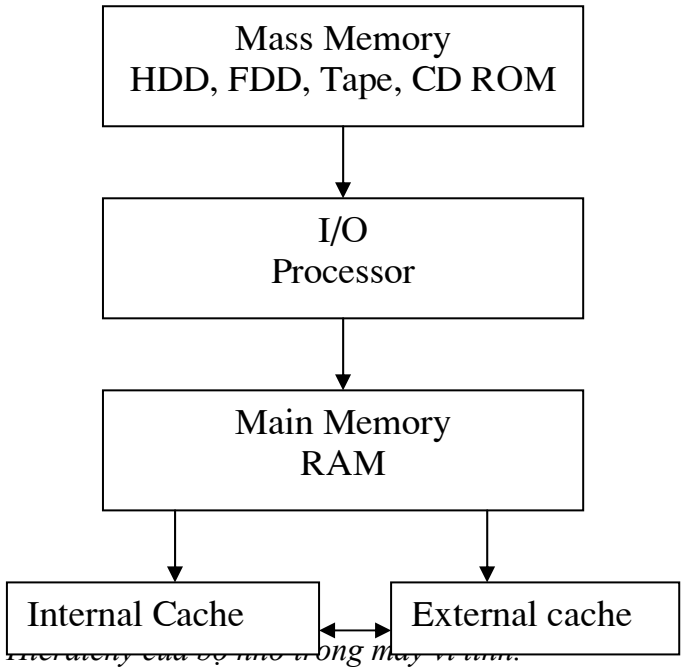
Trong truyền 1 khối ký tự, mỗi ký tự có một bit parity để kiểm tra. ở cuối mỗi khối, ta truyền thêm một ký tự là parity của toàn thể bản tin, gọi là longitudinal check (LRC). Phía thu sẽ tính LRC và so với LRC nhận được để kiểm tra lỗi. Một phương pháp nữa để kiểm tra lỗi khi truyền số liệu là dùng CRC (Cyclic redundance check), đó là một đa thức nhị phân dư thu được khi chia đa thức các bit của bản tin cho một đa thức quy định.

Ví dụ mã sửa sai là mã có 4 từ dài 10 bit như sau:
0000000000, 0000011111, 1111100000, 1111111111. Mã này có khoảng cách là 5, tức là nó có thể sửa được các lỗi kép. Ví dụ nếu ta nhận được từ 0000000111, máy thu sẽ biết rằng từ đó phải là 0000011111 (nếu coi như không có nhiều hơn một lỗi kép). Nhưng nếu một lỗi ba xảy ra, biến 0000000000 thành 0000000111 thì ta không sửa lỗi được.

Để sửa lỗi, người ta dùng thuật toán của Hamming.

I.3. Kiến trúc tổng thể của bộ nhớ. (h 7.2)

Xét một cách tổng thể, bộ nhớ của máy tính có kiến trúc theo cung bậc (hierarchy) trải dài từ bộ nhớ ngoài đến bộ nhớ trong và cuối cùng là đến bộ nhớ đệm (cache) trong và ngoài CPU.



Hình 7.2. *Hierarchy của bộ nhớ trong máy vi tính.*

I.4. Quản lý bộ nhớ (MMU, Memory Management Unit)

Công việc quản lý bộ nhớ của máy vi tính chủ yếu là do bộ vi xử lý đảm nhiệm. Dên cạnh đó còn có DMAC (Direct Memory Acess Controller) cũng tham gia quản lý bộ nhớ trong việc truyền số liệu giữa controller ổ đĩa với bộ nhớ và làm tươi bộ nhớ. Ở những máy có Cache Memory thì Cache Memory Controller thực hiện các công việc truyền số liệu giữa Cache Memory và RAM.

Ở khu vực trung tâm của máy vi tính (bộ vi xử lý, ROM, RAM, các bus...), thực chất của việc quản lý bộ nhớ là các thanh ghi của vi xử lý đưa ra các địa chỉ của ô nhớ hoặc của cổng I/O qua bus địa chỉ, cùng các lệnh điều khiển/ trạng thái khác và đọc vào/ viết ra các số liệu của các ô nhớ ấy. Các bộ phận bên ngoài VXL sẽ giải mã các địa chỉ và các tín hiệu điều khiển/ trạng thái đó để trỏ vào các byte/ từ/ từ kép... của bộ nhớ để thực hiện các thao tác tương ứng.

Còn từ các ổ đĩa trở đi, việc quản lý bộ nhớ là thực hiện các lệnh coả hên điều hành lên các file (có địa chỉ 3 chiều là C-H-S), cụ thể là truyền số liệu nhờ DMAC giữa vùng đệm (buffer) của bộ điều khiển ổ đĩa với bộ nhớ RAM.

Các bộ vi xử lý Intel từ thế hệ 286 trở đi phân biệt hai mode địa chỉ: mode địa chỉ thực (chỉ quản lý 20 bit địa chỉ vật lý của bộ nhớ) và mode địa chỉ bảo vệ (quản lý tới 32 bit địa chỉ ảo nhờ các thanh ghi ẩn trong bộ vi xử lý).

Ở cấp dưới, tức cấp ngoại vi, như bộ điều khiển ổ đĩa, bộ điều khiển màn hình, máy in... cũng có tổ chức bộ nhớ riêng của chúng để tiện cho việc cất giữ và xử lý với các đặc thù riêng.

Các bộ nhớ RAM-ROM và các vùng nhớ của bộ nhớ ngoài (trên các ổ đĩa), khác nhau về cách mã hoá các bit, cách tổ chức, do đó cả cách truy nhập cũng khác nhau.

II. Tổ chức bộ nhớ của vi xử lý.

Bộ nhớ của vi xử lý có thể xem như bao gồm có bộ nhớ ROM và bộ nhớ RAM. Bộ nhớ RAM của vi xử lý chính là các thanh ghi (thanh ghi chung, thanh ghi chỉ số, thanh ghi đoạn, thanh ghi ngăn xếp, thanh ghi trạng thái, thanh ghi cờ, các bộ đệm số liệu/ địa chỉ/ điều khiển...). Còn bộ nhớ RAM là bộ phận giải mã lệnh để phát ra các vi lệnh.

Nhằm mục đích quản lý được số lượng địa chỉ nhớ (ảo) nhiều hơn số đường địa chỉ của bộ vi xử lý và bảo vệ các vùng nhớ của các nhiệm vụ khác nhau (task) và của hạt nhân (kernal)

chống truy nhập không hợp pháp, các vi xử lý có các cách tổ chức đặc biệt các thanh ghi địa chỉ (bộ phận phân trang, điều khiển đoạn của các nhiệm vụ).

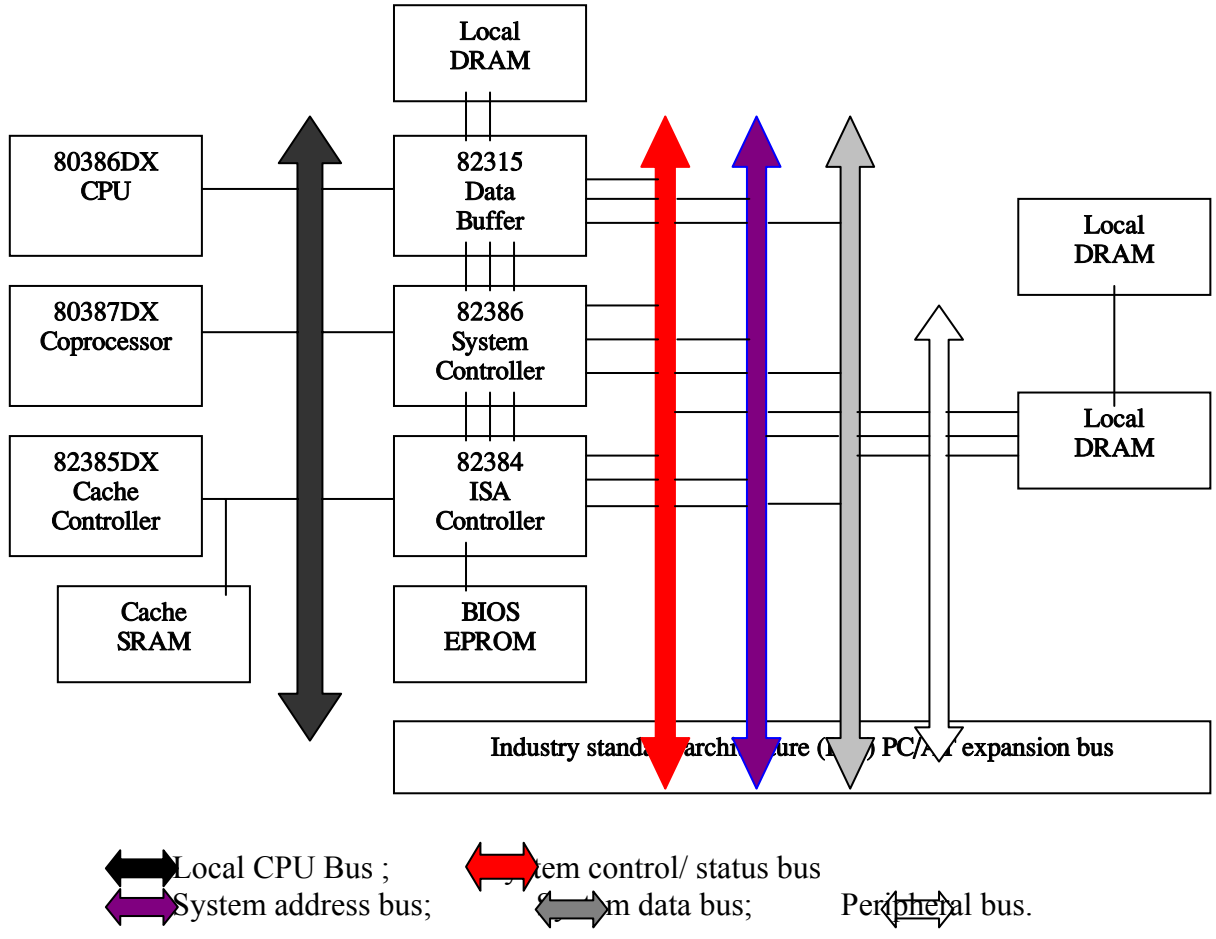
Các bộ vi xử lý từ thế hệ 486 trở đi còn có một bộ nhớ Cache Memory với kích thước nhiều Kbyte để chứa mảng các lệnh và số liệu đang thường dùng lấy từ bộ nhớ RAM, nhằm tăng tốc độ truy nhập.

Để tăng tốc độ tính toán các phép toán dấu chấm động, trong các bộ vi xử lý từ 486 trở đi còn có bộ phận dấu chấm động (FPU, Floating Point Unit), bộ phận này cũng có các thanh ghi FPU phục vụ riêng cho nó.

III. Tổ chức bộ nhớ trong của máy vi tính

Bộ nhớ trong của máy tính dùng để chứa chương trình và số liệu của phần chương trình hạt nhân và các nhiệm vụ. Mỗi byte được gán cho một địa chỉ để vi lý và DMAC có thể truy nhập tới.

Bộ nhớ RAM ở những máy từ 386 trở đi có thể được tách riêng ra bộ nhớ đệm (cache memory), là RAM tĩnh với thời gian truy nhập nhanh, có kích thước dưới 1Mb được nối ngay vào bus nội bộ của máy tính sát ngay vi xử lý và được điều khiển bởi Cache controller. Phần còn lại là DRAM, chậm hơn nhưng rẻ hơn và có dung lượng lớn hơn. Hình 7.3 thể hiện sơ đồ khối bên trong một máy 386.



Hình 7.3. *Phần trung tâm máy tính AT 386*

Trong sơ đồ: Vi xử lý là 80386, đồng xử lý toán là 80387, cache controller 82385 được nối trực tiếp với nhau thành một bus local. Các đường địa chỉ A2-A31 của 386 nối trực tiếp tới các đường cùng tên của 82385DX, các đường số liệu D0-D31 của 386 được nối trực tiếp tới các đường số liệu cùng tên của 387DX. Hơn nữa, các chân quy định chu kỳ bus D/C#, W/R# và M/IO# được nối trực tiếp tới các chân tương ứng của 82385DX.

Từ bus local của VXL, các đường địa chỉ được đệm ra bằng các chốt địa chỉ 8 bit 74373 (không vẽ trong hình). Các đường số liệu của bus local được đệm hai chiều bằng Data Buffer 82345.

System Controller 82346 là trái tim của các chipset 340. Nó nối tới bus local của 386, bus mở rộng ISA, Data buffer 345, ISA Controller 344. Nó thực hiện một số chức năng sau:

- Nhận xung đồng hồ từ bên ngoài để phát nhịp clock TURBO và clock chậm hơn.
- Làm trọng tài bus (các việc về DMA và làm tươi bộ nhớ)
- Phát các tín hiệu địa chỉ hàng RAS và địa chỉ cột CAS đến các dãy nhớ của toàn bộ bộ nhớ DRAM trên MainBoard, phát tín hiệu ghi vào RAM
- Phát tín hiệu ready, tín hiệu Reset CPU
- Giao tiếp giữa đồng xử lý với CPU.

Controller ISA 82344 nối giữa bus local của CPU với bus hệ thống để làm các chức năng giao tiếp với CPU, system controller 346, data buffer 345, ROM, bus, các thiết bị ngoại vi như sau:

- Nhận các tín hiệu BE0# - BE3# của CPU, ROM8# và IOCHRDY từ bus ISA để sinh ra các tín hiệu chọn byte chẵn và byte lẻ SA0# và SBHE#
- Tạo các tín hiệu giao tiếp giữa 344, 345 và 346.
- Chứa khối điều khiển ngoại vi Peripheral Control gồm các vi mạch có độ tích hợp cực cao (VLSI) quen thuộc: hai 82C59 (ngắt), hai chip 82C37A (DMAC), vi mạch định thời 82C54, thanh ghi địa chỉ trang 74LS612, bộ driver cho loa, port B parallel I/O, đồng hồ thời gian thực và bộ đếm làm tươi bộ nhớ.
- Giải mã địa chỉ để tạo ra các tín hiệu chọn chip CS8042# cho controller bàn phím 8042 và ROMCS# để cho phép chọn ROM BIOS.

Vi mạch Peripheral Combo 82341 được ghép vào bus mở rộng của bus ISA, nó chứa các VLSI để thực hiện một số chức năng của các thiết bị ngoại vi sau đây:

- Hai cổng nối tiếp không đồng bộ 16C450
- Một cổng song song cho máy in
- Đồng hồ thời gian thực
- RAM sổ tay, các controller cho bàn phím và chuột.
- Interface cho đĩa cứng (tiêu chuẩn IDE).

Controller đĩa mềm 82077 có thể điều khiển tới 4 ổ đĩa mềm các loại 5”1/2 và 3”1/2.

III.2. Tổ chức bộ nhớ RAM của máy tính.

Xét trường hợp máy 386, nó có 32 bit địa chỉ, từ 00000000H đến FFFFFFFFH, ứng với 4 GByte không gian nhớ vật lý. Về quan điểm phần cứng, ta chia không gian đó thành 4 dãy nhớ độc lập nhau, là bank0 - bank3, mỗi bank kích thước 1 GByte. Chúng cần các tín hiệu Bank Enable BE0# tới BE3#. Trong hình 7.4 sau, ta thấy các địa chỉ A2 - A31 được đặt song song vào tất cả 4 bank nhớ. Còn mỗi bank nhớ chỉ cung cấp 1 byte số liệu cho 32 đường số liệu.

Ở chế độ thực, 386 chỉ dùng các đường địa chỉ A2 - A19 và 4 tín hiệu BE# dùng để chọn bank nhớ. Mỗi bank chỉ có 256 KByte.

Từ hình 7.4 ta thấy không gian nhớ vật lý được tổ chức thành dãy các từ kép (32bit). Do đó mmỗi từ kép xếp đúng hàng (aligned) bắt đầu ở địa chỉ bội số của 4.

Dùng tổ hợp các tín hiệu BE# có thể truy nhập được vào các fỏmat khác nhau (byte, từ, từ kép) như hình 7.5. Việc truy nhập vào địa chỉ đầu của từ kép có thể cần 1 chu kỳ bus (khi từ kép xếp đúng hàng) hoặc 2 chu kỳ bus (khi từ kép xếp lệch hàng, misaligned).

II.3. Interface giữa VXL và bộ nhớ (h 7.7).

Sơ đồ giao tiếp giữa vi xử lý 386 với bộ nhớ ở chế độ bảo vệ được vẽ trên hình 7.6. Ta thấy rằng giao tiếp bao gồm các việc:

- Giải mã các trạng thái của vi xử lý (ADS#, M/IO#, D/C#, W/R#) để cấp ra các tín hiệu điều khiển bus (ALE#, MWTC#, MRDC#, OE# cho bộ nhớ, DT/R# và DEN#).
- Giải mã 3 địa chỉ cao nhất (A29-A31) để có được 8 tín hiệu chọn chip CE0# - CE7#, cho trường hợp mỗi chip 1 bit, rồi chốt các địa chỉ A2-A28 và CE0# - CE7# để đưa sang bộ nhớ.
- Đệm truyền số liệu hai chiều giữa VXL và bộ nhớ được điều khiển bởi các tín hiệu cho phép đưa ra số liệu EN# và định hướng truyền DIR.
- Từ các tín hiệu BE0# - BE3# và MWTC# cấp điều khiển viết lên các bank nhớ WEB0# - WEB3#.
- Bộ nhớ cấp các tín hiệu NA#, BS# và READY# cho VXL.

III.4. Giải mã địa chỉ và Latch địa chỉ, đệm hai chiều số liệu.

Bộ giải mã địa chỉ có thể đặt trước hoặc sau bộ chốt (h 7.7a,b). Sau bộ chốt địa chỉ có khi cần đệm riêng cho địa chỉ I/O. Ví dụ dùng 4F244 có thể sink được 64 mA (h 7.7c).

Để giải mã địa chỉ người ta dùng mạch 74F138 với 8 đường ra (hoặc 74F139 hai mạch giải mã, mỗi mạch có 4 đường ra). Trên hình 7.8 ta thấy 2 địa chỉ cao nhất dùng để giải mã ra 4 tín hiệu chọn chip CE0# - CE3#. Để Latch ta dùng các vi mạch 74F373 (có thể sink được 24 mA max). Chân ra 3 trạng thái OC# nối đất, còn chân CLK của 373 được cấp ALE# lúc cần Latch địa chỉ ra. Chân ra 3 trạng thái OC# nối đất, còn chân CLK của 373 được cấp ALE# lúc cần latch địa chỉ ra.

Hình 7.8 Giải mã và latch địa chỉ của máy 386.

Để đệm và truyền số liệu hai chiều (hình 7.9) cho bus số liệu của VXL (dòng max 4mA) ta dùng các đệm 8 bit hai chiều 74F245 với dòng sink max là 64mA. Ta cũng dùng vi mạch 74F646 là các đệm 2 chiều với thanh ghi, nó có thể dùng như một bộ đệm đơn giản hoặc dùng với chức năng đệm - thanh ghi trong đó số liệu truyền từ bus này vào một thanh ghi bên trong với một dãy tín hiệu điều khiển, và từ thanh ghi trong ra bus kia với tín hiệu điều khiển khác.

II.5. Giải mã trạng thái bus VXL

VXL 386 cấp trực tiếp ra ba tín hiệu quy định kiêuy của chu kỳ nhớ hiện hành của bus là: Mem/IO#, Data/Control# và Write/Read#. Bảng 7.1 chỉ ra 8 kiểu của chu kỳ bus của 386. Ngoài ra, VXL còn cấp AM, và tín hiệu ADS# (Address Status) hạ xuống mức 0 để báo rằng 3 tín hiệu trên AM là bình ổn hữu hiệu. Ở hình 7.6 ta thấy một mạch logic điều khiển bus, được dung để giải mã kiểu của chu kỳ bus nhằm cấp ra các điều khiển tương ứng tới Mem/IO, Latch Address.

Controller bus có thể được chế tạo bởi các PLA (Programable Logic Arrays), nó là các mạch có nhiều lối ra, mỗi lối ra thứ i là nghịch đảo của tổng các tích các lối vào thứ j.

$$\text{Output i} = \sum_{k=1}^{7} \prod_{j=1}^{16} \text{Input j}$$

Các PLA thường có cửa ra ba trạng thái (với chân điều khiển CE#). Có loại còn có thanh ghi D - Latch ở lối ra.

Việc lập trình PLA thực hiện ở nhà máy, bằng cách đốt cháy những mối nối không muốn có tại các nút.

II.6. Bộ phận Cache Memory và Controller Cache Memory.

Mặc dù có dùng các thiết bị nhớ DRAM tốc độ truy nhập tới 60nS, EPROM 120nS,... , nhưng nó vẫn chậm ngay cả với các hệ máy 386 zero-wait-state. Ví dụ 386 loại 25 MHz đã đòi hỏi nhớ có thời gian truy nhập nhỏ hơn 40nS. Vì vậy ta vẫn phải đưa thêm các wait-state vào các chu kỳ bus truy nhập có nhớ.

Vì vậy ta đưa vào giữa VXL và bộ nhớ trong chậm, rẻ tiền một vùng nhớ SRAM có dung lượng nhỏ, thời gian truy nhập rất nhanh để cải thiện vấn đề truy nhập bộ nhớ của máy vi tính. Bộ phận đó gọi là Cache Memory. Bộ phận nhớ này nhanh và có thể được truy nhập không có chu kỳ đợi.

Như vậy Cache Mem giữ các lệnh và số liệu mà CPU lấy từ bộ nhớ chính để đưa và xử lý. Và mỗi khi tìm lệnh hay số liệu, CPU phải xác định xem chúng đã được cất trong Cache chưa; nếu nó tìm thấy trong Cache, ta gọi là trúng Cache, nếu không, gọi là trượt.

Hình 7. là sơ đồ bố trí và tương tác giữa VXL, Cache, bộ nhớ chính trong trường hợp thực hiện một roitine lặp (loop).

Thường dùng hai cách tổ chức cache. Cách thứ nhất là dùng cache trực tiếp (direct-mapped cache) vùng nhớ có địa chỉ offset ở trang nhớ cache 64KB (h 7.). Cách thứ hai là dùng cache hai đường(two way set associative cache) theo đó ta chia trang nhớ cache thành hai bank, mỗi bank 32 KByte. Và vùng nhớ ở các trang của bộ nhớ chính có thể được nạp sang bank A hoặc bank B cảu cache. Do đó tỷ lệ cache trúng sẽ tăng lên. (h 7.).

Thuật toán đổi mới bộ phận nhớ cache thực chất là bỏ phần nội dung nhớ đã lâu không được dùng (least recent used, LRU) và thay vào đó bằng nội dung mới cần dùng. Thuật toán này cùng với dùng cache 2 đường cho phép tăng tỷ lệ cache trúng lên nhiều.

Cache Controler 82385 được thiết kế để nối trực tiếp với CPU 80386. Nó có thể được dùng để cài đặt nhiều cấu hình khác cache nhau. Hình 7. là kiến trúc của một hệ cache với CPU 386, Cache Controller 82385, nhớ cache cùng các đệm số liệu và địa chỉ.

Ta thấy các đường địa chỉ A2 - A31 và BE0# - BE3#, các đường số liệu D0 - D31, các trạng thái bus (M/IO#, D/C#, W/R#) do CPU cấp cho Cache Controller và các Buffer địa chỉ, số liệu, điều khiển, còn Controller cấp một số tín hiệu điều khiển tới bộ nhớ Cache và ra bus local của nó.

Xét ví dụ điều khiển 32 KByte nhớ Cache theo hai phương pháp Cache trực tiếp và Cache 2 đường ở hình 7. . Các tín hiệu điều khiển của Cache Mem gồm:

- CALEN (Cache Address Latch Enable) cấp cho pin E của Latch 373 cho nhớ cache.
- CT/R# (Cache Transmit/ Receive) để điều khiển truyền số liệu DIR ở bộ nhận 245 trên bus số liệu của bộ nhớ cache.
- CS0# - CS3# (Cache chip select) dùng để chọn chip cho bốn vi mạch SRAM.
- COEA#, COEB# (Cache Output Enable) và CWEA#, CWEB# (Cache Write Enable) dùng cho chân OE# của bộ nhận số liệu 245và chân WE# của SRAM.

Ngoài ra còn có các tín hiệu do Controller cấp là

- BACP (Bus Address Clock Pulse) tạo xung nhịp cho các mạch Latch.
- BAOE (Bus Address Output Enable) điều khiển pin OE# của Latch.
- BT/R# (Bus Transmit/ Receive), DOE# (Data Ouput Enable) và LDSTB (Local Data Strobe) điều khiển transceive số liệu 646.

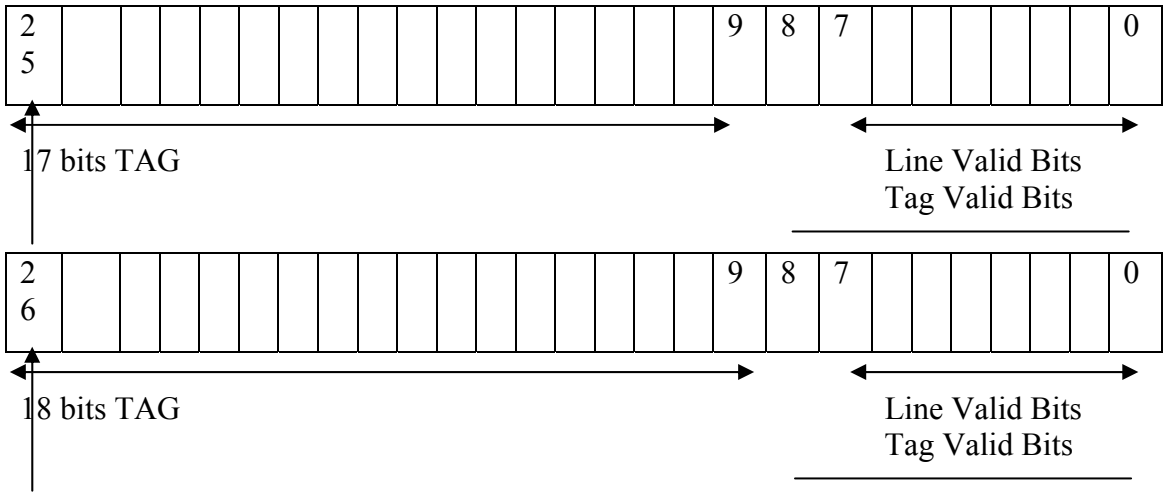
Những tín hiệu giao tiếp giữa Controller với bus local của nó gồm:

- BBE0# - BBE3# (Bus Byte Enable).
- BADS# (Bus Next Address Request)
- BLOCK# (Bus Lock), BHOLD, BHLDA(Bus Hold Acknowledge)
- FLUSH để khởi đầu xoá nhớ Cache bởi thiết bị ngoài.
- MISS (Cache Miss) chỉ ra rằng địa chỉ hiện hành trên bus không tương ứng với số liệu đang có trong Cache và phải đọc lại thông tin từ bộ nhớ chính.
-WBS (Write Buffer Status) chỉ ra rằng các thanh ghi trong 646 chứa những số liệu (để viết vào bộ nhớ chính) đã không được viết vào bộ nhớ chính.

II.7. Hoạt động của Cache trực tiếp và Cache hai đường.

Các hoạt động của Cache trực tiếp và Cache 2 đường được mô tả ở hình 7. . Trong máy tính 386 toàn bộ không gian nhớ vật lý 4 GByte được chia thành 2^{17}-1 trang nhớ 32 KByte. Vì máy 386 có tổ chức số liệu 32 bit, nên mỗi trang có 8Kb từ bép.

Controller chứa 1024 lối vào 26 bit, có tên là SET 0 - SET 1023 để chứa trạng thái của các ô nhớ của Cache Directory. trong trường hợp Cache trực tiếp, mỗi lối vào tương ứng với 8 dòng liên tiếp (từ kép) trong dãy nhớ Cache. Trong trường hợp Cache 2 đường, có hai Cache Directory là A và B ứng với các Bank A và Bank B của nhớ Cache, mỗi Bank chứa 4 KByte từ kép, do đó trong Controller chứa hai tập lối vào (Set Entry) dài 27 bit. Mỗi Set chỉ có 512 lối vào. Định dạng của thông tin đưa tới các lối vào gồm có 8 bit Line Valid Bits, Tag Valid Bit và Tag 17 bit (với Cache trực tiếp), 18 bit (với Cache 2 đường). Hình 7. .



Hình 7. . Format của Entry SET của Cache Directory trực tiếp và hai đường.

Phần TAG dài 17/18 bit chỉ ra số hiệu của 1 trong 131972 trang 32 KB (hoặc 262144 trang 16 KB) trong bộ nhớ chính. Còn TAG_BIT chỉ ra TAG có hữu hiệu hay không. Nếu TAG_BIT = 0 thì tất cả các dòng trong SET là không hữu hiệu. Nếu TAG_BIT = 1 thì mỗi bit trong 8 bit của LINE_VALID_BITS bằng 1 có nghĩa rằng dòng tương ứng trong Cache chứa thông tin hữu hiệu, tức là thông tin trong đó sẽ được cập nhật tự động.

Ví dụ: Nếu SET 1 = 00005FFh, ta chuyển sang dạng nhị phân:
SET 1 = 0000 0000 0000 0000 0101 1111 1111. Từ đó ta có: TAG = 0000 0000 0000 0000 010 = $2_{(10)}$

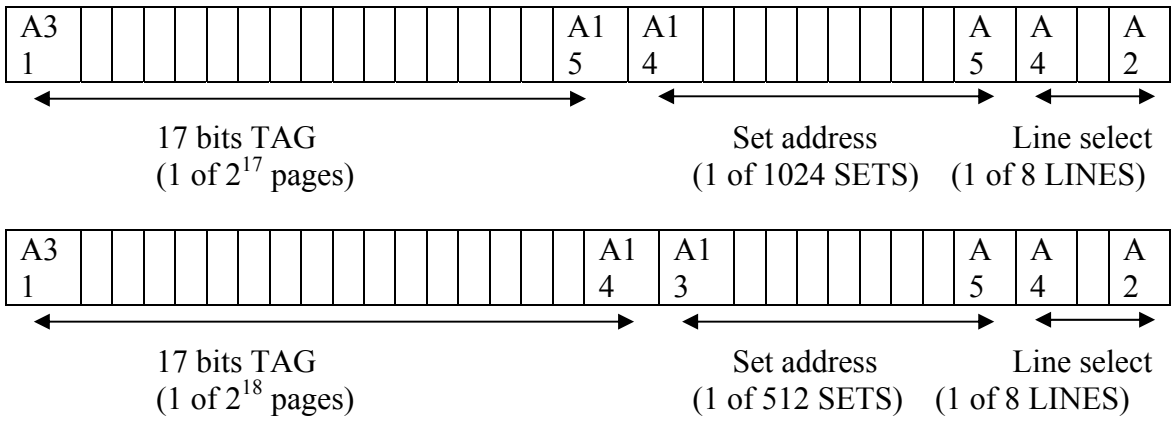
TAG_VALID = 1, do đó những dòng trong LINE_VALID_BIT = 1111 1111 sẽ hữu hiệu. Tức là tất cả 8 dòng trong Cache đều hữu hiệu.

***Cache trực tiếp.**

Khi VXL 386 bắt đầu chu kỳ đọc nhớ, nó cấp địa chỉ song song ra cho 3 nơi là Latch địa chỉ của local bus của controller, lối vào địa chỉ của controller và interface nhớ Cache. khi đó, Cache Controller quyết định là VXL cần đọc từ bộ nhớ chính hay từ Cache. Nó thực hiện điều đó bằng cách thông dịch địa chỉ và so sánh với ENTRY của Cache Directory.

Hình 7. là các trường (field) của bit địa chỉ cho Cache trực tiếp cà Cache hai đường. Trong đó 17/ 18 bit lớn nhất A15 - A31 (hoặc A14 - A31) là TAG để chỉ ra trang của bộ nhớ

chính cần đọc thông tin từ đó vào VXL. Các bit tiếp theo, A5 - A14 (hoặc A5 - A13) gọi là địa chỉ của SET của nhớ Cache, chỗ cần truy nhập vào. Còn 3 bit bé nhất A2 - A4 để chọn dòng trong SET.



Hình 7. . Các trường bit địa chỉ dùng cho Cache trực tiếp và hai đường.

Khi một địa chỉ do VXL đặt vào lối vào địa chỉ của Controller, phần SET của địa chỉ đó được dùng để chọn 1 trong 1024 ENTRY của SET trong Cache Directory. Sau đó Controller tiến hành 3 kiểm tra như sau:

- So sánh trường TAG trong địa chỉ với TAG trong ENTRY của SET đã được chọn, chúng phải trùng nhau.
- Bit TAG_VALID_BIT của ENTRY SET được chọn phải bằng 1.
- LINE_VALID_BIT của ENTRY tương ứng với giá trị trong phần LINE_SELECT của địa chỉ phải = 1.

Nếu cả ba điều kiện trên thoả mãn thì thông tin cần phải đọc từ bộ nhớ đã được lưu trong bộ nhớ Cache và hữu hiệu. Và Controller khởi đầu chu kỳ đọc dữ liệu từ Cache thay vì từ bộ nhớ chính. Đây là trường hợp trúng Cache.

Nếu hai điều kiện đầu thoả mãn, còn LINE_VALID_BIT = 0 thì trượt Cache, tức là ENTRY của SET trong Directory tương ứng với trang đúng của nhớ chính, nhưng dòng từ kép cần phải đọc vào VXL lại chưa được chuyển sang Cache, gọi là trượt dòng. Khi đó VXL phải đọc từ bộ nhớ chính một từ kép, đồng thời được đưa vào nhớ Cache và LINE_VALID_BIT trong ENTRY của Cache Directory được xác định bằng 1. Do đó thông tin được đọc vào Cache và đánh dấu là hữu hiệu.

Nếu trong khi kiểm tra hoặc các TAG không khớp hoặc TAG_VALID_BIT = 0 thì xảy ra trượt TAG (tag miss). Đó là trường hợp đọc một trang đã không được Cache, hoặc đã Cache nhưng không hữu hiệu. Trong trường hợp này Controller phải khởi đầu một chu kỳ đọc từ bộ nhớ chính viết vào bộ nhớ Cache. Lúc đó TAG trong SET ENTRY của Directory được cập nhật bằng phần TAG của địa chỉ, TAG_VALID_BIT được lập bằng 1, một LINE_VALID_BIT do địa chỉ trỏ ra được lập bằng 1, một LINE_VALID_BITS bị xoá đi. Bằng cách này một trang hữu hiệu và ENTRY dòng được lập nên và tất cả các ENTRY khác trong SET baay giờ tương ứng với thông tin trong một trang khác của nhớ chính trở nên không hữu hiệu.

*** Cache hai đường.**

Ở các hình đã nêu ra cách tổ chức nhớ Cache, cùng các format của ENTRY SET, các trường địa chỉ của cả hai trường hợp Cache trực tiếp và Cache hai đường.

Trong trường hợp (hình 7.) Cache hai đường ngoài hai Directory A và B ứng với hai bộ ENTRY, còn có thêm 512 cờ Least Recently Used dài 1 bit (LRU bit). Những cờ này theo dõi xem BANK A hoặc BANK B đang giữ thông tin lâu không sử dụng. Những cờ này được Controller kiểm tra bằng thuật toán thay thế những thông tin lâu không dùng.

Thao tác đọc thông tin từ nhớ Cache hai đường cũng giống như ở Cache trực tiếp. Biết rằng (ở sơ đồ h) SET_ADDRESS chỉ có 9 bit. Đầu tiên địa chỉ 9 bit này được dùng để chọn 1 trong 512 lối vào SET của cả hai Directory A và B. Tiếp theo TAG_ADDRESS 18 bit được so sánh với TAG trong mỗi lối vào SET, TAG_VALID_BITS được kiểm tra, và LINE_VALID_BIT tương ứng với mã của LINE_SELECT (A2 đến A4) được kiểm tra trong mỗi lối vào SET. Nếu ba điều kiện kiểm tra được thoả mãn đối với một trong hai lối vào SET thì ta nói là trúng Cache và thông tin của dòng được đọc vào VXL từ BANK tương ứng của nhớ Cache.

Mạt khác, sẽ xảy ra trượt Cache nếu không khớp các TAG hoặc nếu cả hai VALID_BIT bị xoá, hoặc nếu LINE_VALID_BIT không được lập trong bất cứ lối vào nào, khi đó algorithm sẽ kiểm tra bit cờ LRU đối với SET được chọn bởi địa chỉ SET để xác định xem lối vào của BANK A hay BANK B là lâu không được dùng hơn, sau đó thông tin được đọc vào từ bộ nhớ chính và viết vào BANK nhớ nào lâu không được dùng.

II.9. Làm tươi bộ nhớ DRAM

Bộ nhớ DRAM có các hàng cần phải được làm tươi trong mỗi chu kỳ 2mS. Mạch làm tươi trong chip nhớ phải kiểm tra điện áp các ô nhớ, nếu nó lớn hơn $V_{cc}/2$ thì nạp nó tới V_{cc} , nếu bé hơn $V_{cc}/2$ thì xả hết về 0V.

Để đọc một từ từ BANK nhớ DRAM, trước hết DRAM Controller hoặc một mạch khác cấp tín hiệu WE# = 1. Sau đó gửi nửa thấp của địa chỉ, ứng với địa chỉ hàng, rồi tín hiệu RAS# = 0. Sau 1 thời gian, controller cấp nửa địa chỉ cao, ứng với địa chỉ cột, rồi tín hiệu CAS# = 0. Sau thời gian nhất định, từ cần có sẽ xuất hiện trên Output Data của nhớ.

Để viết vào DRAM, các tín hiệu cũng tương tự, ngoại trừ sau tín hiệu CAS# = 0, controller cấp WE# = 0 để quy định viết vào RAM.

Controller làm tươi DRAM bằng cách gửi ra mỗi địa chỉ trong 512 địa chỉ hàng và cấp RAS# = 0 theo chu kỳ, khoảng 4mS. Việc làm tươi được tiến hành hoặc theo *burst mode* hoặc theo *distributed mode*. Trong burst mode toàn bộ 512 hàng được định địa chỉ và đánh nhịp lần lượt cách nhau 4mS. Còn ở distributed mode hàng được định địa chỉ và đánh nhịp sau 4/512 mS. Hình 7. là mạch làm tươi DRAM với controller làm tươi 8208.

Hình 7. . *Mạch làm tươi bộ nhớ dùng 8028.*

Những nhiệm vụ chính của việc điều khiển nhớ DRAM của máy tính là:

- Làm tươi mỗi ô nhớ sau một khoảng thời gian vài mS.
- Cấp hai nửa địa chỉ cùng các tín hiệu RAS#, CAS# thích hợp.
- Bảo đảm thao tác đọc/viết và làm tươi không xảy ra đồng thời.
- Cấp tín hiệu đọc/viết để điều khiển chiều số liệu.

Hình 7. mô tả sơ đồ Controller 8208 làm tươi 1 MByte cho hệ VXL 8086. Bộ nhớ chia thành 2 BANK (mỗi BANK 8 bit). Controller bảo đảm cấp các địa chỉ hàng và địa chỉ cột, tín hiệu RAS#, CAS#, và các tín hiệu READ/WRITE. Các chân trạng thái ra S0 - S3 của VXL đấu thẳng tới các chân vào của 8208. Controller giải mã các tín hiệu này để cho ra các tín hiệu đọc và viết mà VXL yêu cầu. Do đó, đa số thòi gian của VXLđược dùng để đọc byte/từ của RAM mà không cần có các chu kỳ chờ. Nếu trong khi 8208 đang ở giữa chu kỳ làm tươi nhớ mà VXL muốn đọc RAM thì 8208 lưu giữ AACK cao và buộc VXL cấp thêm một chu kỳ đợi để 8208 kịp hoàn thành chu kỳ làm tươi. Để tiết kiệm chân, không có các chân số liệu (để nạp từ điều khiển), chân PDI nối mass sẽ cho phép 8208 tự khởi đầu hoạt động trong đa số các ứng dụng. Còn các trường hợp khác thì chân PDI sẽ được điều khiển bởi một thanh ghi dịch vào song song - ra nối tiếp, nhờ đó từ điều khiển được nạp vào 8208. Sau khi Reset chân WE/PCLK sẽ cấp ra một dãy xung đánh nhịp cho từ điều khiển từ thanh ghi dịch nạp vào 8208. Từ điều khiển được thực hiện bằng nối ở lối vào của thanh ghi dịch.

Ta cũng có thể dùng DMAC để làm tươi bộ nhớ. Hình 7. là ví dụ mạch 4 BANK với dung lượng 256KB nhớ. Ở đây máy tính dùng chế độ đọc DMA ảo. Bộ định thời 8253 lập trình để phát xung nhịp 15μS. Xung này được nối vào một trong các lối vào xin DMA (DMA Request) là DREQ0 của 8237 DMAC được lập trình để đọc từ nhớ và viết vào một cổng không tồn tại. Khi DMAC nhận xung này, nó gửi một tín hiệu HOLD_REQUEST tới VXL rồi VXL trả lời bằng tín hiệu HLDA và đặt các chân của nó ở trạng thái trở kháng cao. Khi đó: 8237 chiếm lấy bus, gửi ra các địa chỉ nhớ, tín hiệu đọc nhớ và tín hiệu chấp nhận DMA kênh 0 (DACK0).

Tám bit địa chỉ thấp gửi tới nhớ, còn DACK0 để cung cấp xung RAS# cho các bank DRAM để làm tươi nhớ động. Sau mỗi thao tác DMA thanh ghi địa chỉ hiện hành trong DMAC được tự động tăng/giảm (tuỳ thuộc cách lập trình lúc đầu) để làm tươi hàng (row) nhớ sau. Nếu 8237 lập trình để truyền 64 kByte, khởi đầu ở địa chỉ 0, tăng đếm sau mỗi lần DMA, và tự khởi động (autoinitialize), thì dãy các địa chỉ gửi ra sẽ làm tươi tất cả 256 trong hàng DRAM. Mỗi hàng làm tươi 15ns.

Ví dụ với tần số clock 4.77MHz dùng trong IBM PC, một chu kỳ DMA để làm tươi mất 820 ns mỗi 15 ns, tức 5% thời gian của VXL.

Để kiểm tra Parity mỗi bank nhớ có 9 bit, 8 bit để giữ số liệu, bit thứ 9 là bit Parity. Mỗi mạch 74 LS280 dùng để phát/ kiểm parity cho mỗi byte và cất vào parity bit mỗi khi byte được viết vào nhớ. Khi 9 bit đươc đọc ra, parity được kiểm tra. Nếu parity sai thì tín hiệu báo lỗi sẽ được gửi tới cổng 8255 để cho VXL đọc. Khi bắt đầu bật máy, thì quá trình POST xảy ra, nó viết mẫu byte vào tất cả ô nhớ, rồi kiểm tra bằng cách đọc lại chúng cùng với parity bit.

II.10 . Chuyển một mảng số liệu bằng DMA

Thường xuyên có các nhu cầu chuyển mảng số liệu nhớ và ngoại vi. Lúc đó ta dùng DMAC. Hình mô tả cơ chế hoạt động của DMAC với VXL để truyền số liệu giữa nhớ và ngoại vi (ổ đĩa thông minh).

Khi ta bật máy lúc đầu các khoá ở vị trí đóng từ VXL tới ngoại vi, và nhớ. Chúng ta lập trình để chạy DMAC, ví dụ để đọc file từ ổ đĩa để viết vào nhớ. Muốn thế phải gửi một loạt lệnh tới controller ổ đĩa yêu cầu nó đọc những block dữ liệu từ đĩa. Khi controller đã có byte đầu tiên, nó gửi DMA Request(DREQ) cho DMAC, nếu channel đó của DMAC không bị che chắn , DMAC gửi HOLD REQUEST tới chân HOLD của VXL, VXL treo các bus cao và gửi ra HLDA cho DMAC, khi DMAC nhận HLDA của VXL, nó cho tín hiệu điều khiển để đặt ba khoá về vị trí DMA, cắt VXL ra, sau đó DMAC cho ra địa chỉ cấp cho nhớ, DMAC gửi DMA-Acknowlege (DACK0) cho ổ đĩa để nó đưa ra số liệu, cuối cùng nó cấp MEMW#=0 và IOR#=0 ra bus điều khiển, nhờ vậy liệu được đọc vào từ ngoại vi và viết ra ô nhớ, khi truyền số liệu hoàn thành DMAC thu lại tín hiệu HRQ, do đó VXL lấy lại các bus của nó cho đến lần DMA sau.

Hình là mạch chi tiết của sơ đồ hình . Trong đó 8237 là DMAC còn 8272 là controller ổ đĩa mềm, 8282 dùng để latch 8 bit địa chỉ gửi ra từ VXL (do ALE của 8086 điều khiển) hoặc 8237 (do AEN và AD dress STrobe điều khiển).

Khi đóng điện DMAC cấp AEN = 0, các vi mạch U1, U2, U4 được hữu hiệu. Và ALE từ VXL được dùng để đánh nhịp (STroBe) cho 3 vi mạch này. Do đó chúng chốt các địa chỉ A0-A19 của VXL ra bus địa chỉ như trường hợp thông thường (không DMA).

Khi DMAC muốn chiếm lấy các bus, nó cấp AEN= 1, dẫn đến:

- Khoá không cho U1 làm việc, cắt các địa chỉ A0 -A7 từ VXL, DMAC trực tiếp cấp ra 8 địa chỉ thấp cho nhớ trong truyến số liệu,
- AEN =1 làm đổi vị trí Multiplex khiến cho việc đánh nhịp cho U2 thực hiện bởi ADSTB của DMAC. Để tiết kiệm chân, DMAC 8 bit địa chỉ cao qua các chân số liệu D0-D7, cùng với ADSTB=1 báo rằng đó là các địa chỉ cao A15- A8 do DMAC cấp cho qua nhớ latch U2.
-Cũng do AEN =1, các bit A16- A19 do U3 cấp từ các bit D10 -D13 do ta lập trình cứng .
-Cuối cùng, các tín hiệu điều khiển được đổi nối từ các output cảu VXL sang các output của DMAC (gồn IOR#, IƠ#, MEMƯ#, MEMR#).

Các buffer số liệu hai chiều 8286 cho phép có thể truyền 8 bit số liệu tới/từ controller đĩa từ/tới hoặc byte cao hoặc byte thấp của bộ nhớ. Bit địa chỉ A0 dùng để chọn đường cho hai byte nhớ chẵn/lẻ đó.

DMAC có 4 kênh (channel), nhiều thanh ghi trong để:

-Ghi địa chỉ nhớ cơ sở(16 bit).
-Địa chỉ nhớ hiện hành .
-Ghi số đếm tạm thời.
-Ghi địa chỉ lệch
-Các thanh ghi mode
-Ghi yêu cầu xin DMA
-Ghi số đếm từ (word) nhớ cơ sở
-Ghi địa chỉ tạm thời
-Ghi trạng thái
-Ghi tạm thời
-Ghi chắn DMA

DMAC có 4 chân địa chỉ và 2 bit vào IOR#, IOW# để điều khiển hoạt động đọc/viết các thanh ghi của nó. Nó còn có một flip flop để trỏ địa chỉ byte cao/byte thấp đang có ở 8 chân số liệu của nó. Các flip flop này được lần lượt tự động lật trạng thái để cho phép cấp ra 16 bit địa chỉ nhờ chỉ một cổng 8 bit. Tất nhiên để điều khiển hoạt động của DMAC cần phải lập trình khởi đầu nó, và lập trình các hoạt động sau đó của nó. DMAC có thể lập trình để truyền 1byte cho mỗi request, 1 khối các byte cho mỗi request, hay truyền cho đến khi nhận được 1 tín hiệu dừng từ chân vào/ra EOP#.

Đại thể phải làm các việc sau:

- Viết từ điều khiển vào địa chỉ trong 1101 để xoá flip flop trong
- Viết từ điều khiển vào địa chỉ trong 1000

- Viết từ mode cho mỗi channel (dùng địa chỉ trong 1011)
- Viết ra địa chỉ nhớ đầu tiên tới địa chỉ trong của thanh ghi cơ sở cho mỗi channel ta cần
- Viết ra số byte ta muốn truyền tới địa chỉ trong của thanh ghi đếm số lượng từ cơ sở cho mỗi kênh
- Viết từ/ các từ điều khiển để xoá mặt nạ cho channel/ các channel cần dùng.

Table a

SIGNALS						OPERATION
A3	**A2**	**A1**	**A0**	$\overline{\text{IOR}}$	$\overline{\text{IOW}}$	
1	0	0	0	0	1	READ STATUS REGISTER
1	0	0	0	1	0	WRITE COMMAND REGISTER
1	0	0	1	0	1	ILLEGAL
1	0	0	1	1	0	WRITE REQUEST REGISTER
1	0	1	0	0	1	ILLEGAL
1	0	1	0	1	0	WRITE SINGLE MASK REGISTER SET
1	0	1	1	0	1	ILLEGAL
1	0	1	1	1	0	WRITE MODE REGISTER
1	1	0	0	0	1	ILLEGAL
1	1	0	0	1	0	CLEAR BYTE POINTER FLIP/ FLOP
1	1	0	1	0	1	READ TEMPORARY REGISTER
1	1	0	1	1	0	MASTER CLEAR
1	1	1	0	0	1	ILLEGAL
1	1	1	0	1	0	CLEAR MASK REGISTER
1	1	1	1	0	1	ILLEGAL
1	1	1	1	1	0	WRITE ALL MASK REGISTER BITS

Table b

NAME	**SIZE**	**NUMBER**
BASE ADDRESS REGISTER	16 BITS	4
BASE WORD COUNT REGISTER	16 BITS	4
CURRENT ADDRESS REGISTER	16 BITS	4
CURRENT WORD COUNT REGISTER	16 BITS	4
TEMPORARY ADDRESS REGISTER	16 BITS	1
TEMPORARY WORD COUNT REGISTER	16 BITS	1
STATUS REGISTER	8 BITS	1
COMMAND REGISTER	8 BITS	1
TEMPORARY REGISTER	8 BITS	1
MODE REGISTER	6 BITS	4
MASK REGISTER	4 BITS	1
REQUEST REGISTER	4 BITS	1

CHANEL	REGISTER	OPERATION	SIGNAL							INTERNAL FLIP - FLOP	DATA BUS DB0 - DB7
			$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A3	A2	A1	A0		
0	BASE AND CURRENT ADDRESS	WRITE	0	1	0	0	0	0	0	0	A0 - A7
			0							1	A8 - A15
		READ	0	1	0	0	0	0	0	0	A0 - A7
	CURRENT ADDRESS		0							1	A8 - A15
		WRITE	0	0	1	0	0	0	0	0	W0 - W7
	BASE AND CURRENT WORD COUNT		0							1	W8 - W15
		READ	0	0	1	0	0	0	0	0	W0 - W7
			0							1	W8 - W15
	CURRENT WORD COUNT		0	1	0	0	0	0			
			1								
			0	1	0	0	0	0			
			1								
			0	0	1	0	0	0			
			1								
			0	0	1	0	0	0			
			1								
1	BASE AND CURRENT ADDRESS	WRITE	0	1	0	0	0	1	0	0	A0 - A7
			0							1	A8 - A15
		READ	0	1	0	0	0	1	0	0	A0 - A7
	CURRENT ADDRESS		0							1	A8 - A15
		WRITE	0	0	1	0	0	1	0	0	W0 - W7
	BASE AND CURRENT WORD COUNT		0							1	W8 - W15
		READ	0	0	1	0	0	1	0	0	W0 - W7
			0							1	W8 - W15
	CURRENT WORD COUNT		0	1	0	0	0	1			
			1								
			0	1	0	0	0	1			
			1								
			0	0	1	0	0	1			
			1								
			0	0	1	0	0	1			
			1								

2	BASE AND CURRENT ADDRESS CURRENT ADDRESS BASE AND CURRENT WORD COUNT CURRENT WORD COUNT	WRITE READ WRITE READ	0 0 0 0 0 0 0 0 0 1 0 1 0 1	1 1 0 0 1 1 0 0	0 0 1 1 0 0 1 1	0 0 0 0 1 1 0 0	1 1 1 1 0 0 1 1	0 0 0 0 0 0 1 1	0 1 0 1 0 1 0 1	A0 - A7 A8 - A15 A0 - A7 A8 - A15 W0 - W7 W8 - W15 W0 - W7 W8 - W15
3	BASE AND CURRENT ADDRESS CURRENT ADDRESS BASE AND CURRENT WORD COUNT CURRENT WORD COUNT	WRITE READ WRITE READ	0 0 0 0 0 0 0 0 0 1 0 1 0 1	1 1 0 0 1 1 0 0	0 0 1 1 0 0 1 1	0 0 0 0 1 1 0 0	1 1 1 1 0 0 1 1	0 0 0 0 0 0 1 1	0 1 0 1 0 1 0 1	A0 - A7 A8 - A15 A0 - A7 A8 - A15 W0 - W7 W8 - W15 W0 - W7 W8 - W15

Bảng : *Các thanh ghi và địa chỉ trong của DMAC8237.*

Trong máy tính AT ta dùng hai DMAC, địa chỉ của chúng trong mapping I/O là như sau:

000 -01F : DMAC 1(8237A)

0C0 -0DF : DMAC 2 (8237)

087, 083, 081, 082, 08B, 089, 08A, 08F: DMA Page Register (cấp các địa chỉ A16 -A23 cho các kênh 0, 1, 2, 3, 5, 6, 7, và làm tươi).

Bốn kênh của DMAC 1 (đánh số từ 0 tới 3) dùng để truyền số liệu 8 bit giữa các adapter I/O 8 bit với nhớ 16 bit. Mỗi kênh có thể giúp truyền 16 MByte số liệu tổ chức thành các khối 64 kByte. (Các chân BHE là đảo của A0).

DMAC2 có các kênh từ 4 -7. Kênh 4 dùng để nối tầng bốn kênh 0 đến 3 vào VXL. Ba kênh 5, 6, 7 dùng truyền số liệu 16 bit giữa các adapter I/O16 bit với nhớ 16 bit. Các kênh DMA có thể truyền 16 MByte của các khối 128 kByte. Các kênh 5, 6, 7 không thể truyền số liệu của các byte bắt đầu bằng địa chỉ lẻ (các chân A0, và BHE đều = 0).

Trong slot ISA của máy vi tính AT có các chân sau dùng cho hai DMAC:

DRQ0, DRQ1, , DRQ2, DRQ3, DRQ4, DRQ5, DRQ6, DRQ7 và

DACK0 ACK1, DACK2, DACK3, DACK4, DACK5, DACK6, DACK7.

CHƯƠNG VII. GIAO DIỆN TRONG MÁY VI TÍNH

Một hệ thống máy tính điển hình từ cỡ nhỏ đến cỡ trung bình, bao gồm một bộ vi xử lý trung tâm, bộ nhớ trong và hệ thống phối ghép vào/ ra. Các thành phần này liên hệ với nhau thông qua hệ thống các bus. Chương này sẽ nghiên cứu phần cuối cùng của hệ thống máy tính, là bộ phối ghép vào/ ra. Cụ thể là các chip phối ghép vào/ ra, máy tính được liên hệ với thế giới bên ngoài thông qua các chip này.

I. Các chip vào/ ra (I/O chip)

Trong thế giới máy tính, đã có rất nhiều loại chip vào/ra và các chủng loại chip mới cũng liên tục xuất hiện. Trong số các chip thông dụng có thể nói đến các chip điều khiển truyền thông UART, USART, chip điều khiển hiển thị màn hình CRTC, chip điều khiển các đơn vị ổ đĩa HDC/FDC và các chip điều khiển vào/ ra qua các cổng song song PIO.

I.1. Chip nhận - phát không đồng bộ UART

Chip UART (Universal Asynchronous Receiver Transmitter), có thể đọc một byte dữ liệu từ bus dữ liệu và chuyển từng bit dữ liệu của nó lên đường dây nối tiếp tới các thiết bị đầu cuối (terminal) hoặc nhận dữ liệu từ terminal. Các chip UART thường hoạt động ở tốc độ từ 50bps tới 19,2 Kbps.

I.2. Chip nhận - phát đồng bộ/không đồng bộ USART

Chip USART(Universal Synchronouns Asynchronous Receiver Transmitter) có thể quản lý việc truyền dữ liệu đồng bộ bằng việc sử dụng nhiều giao thức khác nhau, cũng như có thể sử dụng tất cả các chức năng của UART.

I.3. Các chip vào/ra song song PIO(Parallel I/O)

Một trong những chip PIO điển hình là chip 8255A, như hình 7.1. Nó có 24 cổng vào/ra, có thể ghép nối với mọi thiết bị tương thích TTL, như bàn phím, các chuyển mạch, máy in. Cho phép CPU đọc hoặc ghi các bit dữ liệu trên mọi cổng vào/ra, làm cho chip này hoạt động rất linh hoạt.

CPU có thể định cấu hình cho 8255A bằng cách nạp giá trị cho các thanh ghi trạng thái bên trong vi mạch này.

Vi mạch gồm:

a. Phần ghép nối với vi xử lý có:

- Bộ đệm số liệu để trao đổi dữ liệu hai chiều (vào, ra) giữa vi xử lý và vi mạch.
- Bộ logic điều khiển đọc/ghi, tức là bộ giải mã địa chỉ lệnh cho các thanh ghi đệm và thanh ghi điều khiển.

b. Phần ghép nối với các thiết bị ngoài có:

- Trạm A và trạm B, mỗi trạm này được gắn với một thanh ghi chốt 8 bit, có chức năng vào hoặc ra tuỳ theo chương trình khởi phát.
- Trạm C 8 bit, chia thành hai phần, nửa thấp 4 bit và nửa cao 4 bit.

Tuỳ theo chế độ sử dụng được xác lập bởi lời điều khiển, trạm C có thể được dùng để trao đổi dữ liệu vào hoặc ra (chế độ 0); điều khiển hoặc đối thoại với thiết bị ngoài và vi xử lý khi trạm A và trạm B ở chế độ 0 bằng cách xác lập và xoá từng bit PC_j ; điều khiển hoặc đối thoại với thiết bị ngoài và vi xử lý khi các trạm A và B ở chế đọ 1 và 2.

Ở các chế độ 1 và 2, đọc các bit của trạm C, ta biết được trangh thái của trạm A và B.

c. Phần các mạch điều khiển nội bộ:

Có các khối điều khiển (nhóm A hay nhóm B) các trạm A, B và C.

I.1.1. Các lệnh ghi và đọc các cửa (trạm) và các thanh ghi điều khiển

Với tổ hợp của các tín hiệu địa chỉ (A_0, A_1), chọn chip ($\overline{CS}$), các lệnh đọc ($\overline{RD}$), và ghi ($\overline{WR}$) của vi xử lý

CHƯƠNG VII. VÀO RA DỮ LIỆU VỚI THIẾT BỊ NGOẠI VI

I. Vai trò và nhiệm vụ của bộ phối ghép

I.1. Vai trò của bộ phối ghép

Bộ phối ghép nằm trung gian giữa máy vi tính và các thiết bị ngoài, đóng vai trò trung chuyển dữ liệu (nhận và truyền) giữa chúng.

Khi truyền dữ liệu từ máy vi tính ra thiết bị ngoài, bộ phối ghép đóng vai trò nhận dữ liệu từ máy tính và là nguồn cấp dữ liệu cho thiết bị ngoài.

Khi truyền dữ liệu từ thiết bị ngoài vào máy vi tính, bộ phối ghép đóng vai trò nhận dữ liệu từ thiết bị ngoài và là nguồn cấp dữ liệu vào cho máy tính.

I.2. Nhiệm vụ của bộ phối ghép.

Bộ phối ghép làm nhiệm vụ phối hợp trao đổi dữ liệu giữa máy tính và thiết bị ngoài về mức và công suất của tín hiệu, về dạng tín hiệu, về tốc độ và phương thức trao đổi.

I.2.1. Phối hợp về mức và công suất tín hiệu

Mức tín hiệu của máy vi tính thường là mức (0V, 5V) trong khi của các thiết bị ngoài, hoặc ở mức cao (± 15V, ± 48V) hoặc rất thấp (<<1V). Do đó, bộ phối ghép phải biến đổi các mức trên cho phù hợp.

Công suất của các tín hiệu trên bus dữ liệu của máy vi tính rất nhỏ (cỡ vài chục mA), trong khi cần công suất lớn hơn nhiều cho thiết bị ngoài. Do đó bộ phối ghép phải biến đổi công suất cho phù hợp.

Ở các ngõ vào và ngõ ra của bộ phối ghép thường dùng các mạch đệm ba trạng thái.

I.2.2. Phối hợp về dạng dữ liệu (tín hiệu).

Bộ phối ghép phải đảm bảo tính tương thích về cơ chế trao đổi dữ liệu giữa máy tính và thiết bị ngoài.

I.2.3. Phối hợp về tốc độ trao đổi dữ liệu.

Máy tính thường hoạt động với tốc độ cao, trong khi các thiết bị ngoài thường hoạt động chậm hơn. Do đó bộ phối ghép phải có khả năng cấp, nhận dữ liệu nhanh với máy tính, nhưng vói thiết bị ngoại thì ngược lại .

I.2.4. Phối hợp về phương thức trao đổi dữ liệu.

Để đảm bảo sự trao đổi dữ liệu một cách tin cậy, cần có bộ phối ghép và phương thức trao đổi dữ liệu diễn ra theo một trình tự nhất định và hợp lý.

- *Nếu việc trao đổi dữ liệu do máy tính yêu cầu thì quá trình diễn ra như sau:*

Máy tính đưa lệnh điều khiển để khởi động bộ phối ghép hay thiết bị ngoài.

Máy tính đọc tín hiệu trả lời. Nếu có tín hiệu sẵn sàng mới trao đổi tin, nếu không, thêm một chu kỳ chờ và đọc lại trạng thái.

Máy tính trao đổi tin khi đọc thấy trạng thái sẵn sàng.

- *Nếu việc trao đổi tin do TBN yêu cầu:* để giảm thời gian cồ đợi trạng thái sẵn sàng của TBN, máy tính có thể khởi động TBN rồi thực hiện các nhiệm vụ khác. Việc trao đổi tin diễn ra khi:

TBN gửi yêu cầu trao đổi tin tới bộ xử lý ngắt của khối ghép nối, để đưa yêu cầu ngắt chương trình đến máy tính.

Nếu có nhiều thiết bị ngoài cùng gửi yêu cầu, KGN xử lý theo mức ưu tiên ngắt định trước, rồi đưa yêu cầu trao đổi tin cho máy tính.

Máy tính nhận yêu cầu, chuẩn bị trao đổi và gửi tín hiệu xác nhận sẵn sàng trao đổi.

KGN nhận và truyền tín hiệu xác nhận cho TBN.

TBN trao đổi tin với KGN và KGN trao đổi tin với máy tính (nếu là đưa tin vào) hoặc máy tính trao đổi tin với KGN và KGN trao đổi tin với TBN (nếu là đưa tin ra).

II. Cấu trúc chung của khối ghép nối

II.1. Nhiệm vụ của các khối trong KGN.

KGN có nhiệm vụ chung là nhận và chuyển tin giữa máy tính và TBN. Nhưng cụ thể, có những nhiệm vụ nhỏ khác nhau trong sơ đồ khối. Những nhiệm vụ và các khối tương ứng là:

1. Ghép nối và biến đổi tin giữa MT - KGN và KGN - TBN về:

- Mức và công suất tín hiệu.
- Dạng tin (song song, nối tiếp, tín hiệu số, tín hiệu analog).

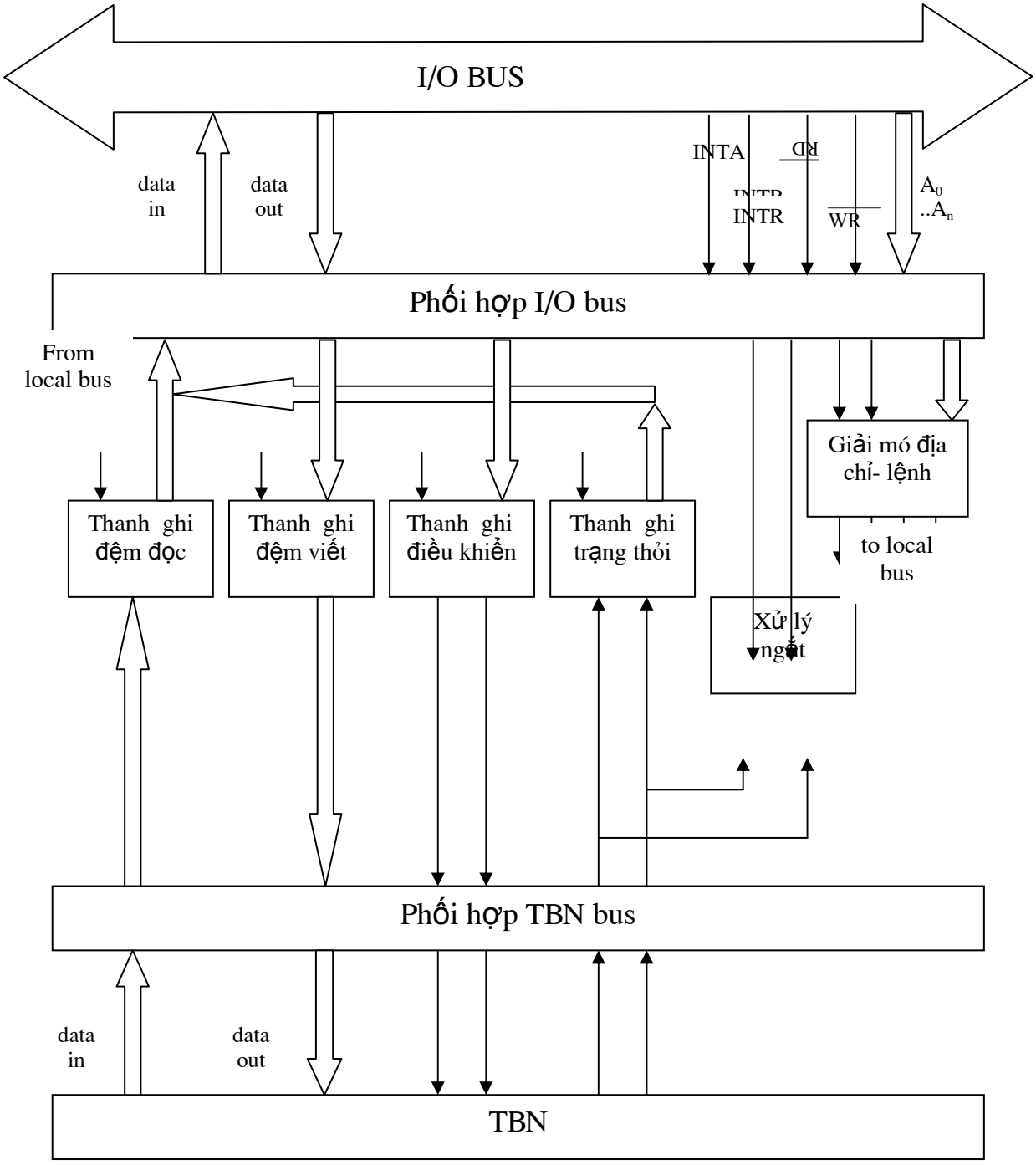
2. Giải mã địa chỉ, giải mã lệnh cho các thanh ghi đệm của KGN.

3. Ghi nhận trạng thái TBN hay yêu cầu trao đổi tin của TBN, xử lý yêu cầu ưu tiên, gửi yêu cầu vào MT và xác nhận trao đổi tin từ MT.

4. Ghi nhận, biến đổi dạng tin, phát tin cho thiết bị nhận tin.

5. Nhận và phát tín hiệu nhịp thời gian trao đổi tin cho các khối trong KGN và TBN.

II.2. Sơ đồ khối.



1. Khối phối hợp đường dây MT.

Khối có nhiệm vụ:

- Phối hợp mức và công suất tín hiệu với bus I/O của MT.
- Cô lập bus I/O với các TBN khi không trao đổi tin.
- Điều khiển đưa tin ra, đưa tin vào bus I/O.

Các nhiệm vụ trên dược thực hiện nhờ các vi mạch đệm ba trạng thái.

2. Khối giải mã địa chỉ - lệnh.

Mỗi thanh ghi đệm (điều khiển, trạng thái, số liệu đọc vào, số liệu đưa ra) của KGN được chọn để ghi và đọc tin nhờ các lệnh đọc, ghi từ khối giả mã địa chỉ - lệnh. Khối giải mã này là những vi mạch giải mã hay tổ hợp các cổng logic. Lối vào được nối với bus I/O của MT, để nhận các tín hiệu địa chỉ (A_0 .. A_n), tín hiệu điều khiển đọc, ghi, các tín hiệu chốt địa chỉ, chốt dữ liệu. Lối ra của khối này là các tín hiệu đọc, ghi cho từng thanh ghi đệm của KGN.

3. Các thanh ghi đệm gồm:

- Thanh ghi điều khiển chế độ hoạt động, thanh ghi điều khiển TBN.

- Thanh ghi trạng thái hay yêu cầu trao đổi tin của TBN.

- Thanh ghi đệm số liệu ghi

- Thanh ghi đệm số liệu đọc.

4. Khối xử lý ngắt.

Khi nhận, che chắn yêu cầu trao đổi tin của TBN, xử lý ưu tiên và đưa yêu cầu trao đổi tin vào MT.

5. Khối phát nhịp thời gian.

Phát nhịp thời gian cho các hoạt động truyền và xử lý tin trong KGN hay TBN. Đôi khi, để đồng bộ, khối còn nhận tín hiệu nhịp đồng hồ từ MT.

6. Khối đệm TBN.

Khối có thể biến đổi mức (TTL), biến đổi công suất (cho các TBN là các mạch điều khiển công suất) và biến đổi về dạng tin.

7. Khối điều khiển:

Điều khiển hoạt động của các khối, như khối phát nhịp thời gian, chế độ hoạt động, vv... .

III. Giải mã địa chỉ cho bộ ghép nối.

Việc giải mã địa chỉ cho bộ ghép nối cũng gần giống như giải mã địa chỉ cho mạch nhớ. Chủ yếu ta nghiên cứu việc giải mã địa chỉ cho các cổng. Thông thường các cổng có địa chỉ 8 bit tại A0-A7 hoặc có địa chỉ 16 bit tại A0-A15. Tuỳ theo độ dài của toán hạng trong lệnh là 8 hay 16 bit ta sẽ có 1 cổng 8 bit hay 2 cổng 16 bit có địa chỉ liền nhau để tạo nên từ với độ dài tương ứng. Trong thực tế ít có hệ sử dụng hết 256 cổng I/O khác nhau, nên ta chỉ xét ở đây các bộ giải mã địa chỉ 8 bit A0-A7 và mạch giải mã thông dụng như 74LS138 để tạo ra các xung chọn thiết bị.