CHƯƠNG 2: TỔ CHỨC HỆ THỐNG VI XỬ LÝ

1. Giới thiệu

Tất cả các máy vi tính IBM họ PC hoặc các máy vi tính tương thích IBM đều sử dụng µP Intel họ iAPX. Bảng 2.1 liệt kê các đặc tính cơ bản của một số µP của Intel trong đó 80486 chứa một bộ điều khiển cache tích hợp và 8 KB RAM tĩnh, Pentium chứa cache 16 KB RAM tĩnh.

Bảng 2.1: Kiến trúc các µP của Intel 8 bit, 16 bit và 32 bit

ĐẶC TÍNH	8080	8086	8088	80186	80188	80286	80386	386SX	486/Pentium
Bus địa chỉ (số bit)	8	16	8	16	8	16	32	16	32
Đường dữ liệu nội (số bit)	8	16	16	16	16	16	32	32	32/64
Tốc độ (MHz)	2,2.6, 6.3	5,8,10	5,8	8,10, 12.5	8,10, 12.5	6,8,10, 12.5,20	16,20,25, 33	16	25-66
Thanh ghi đến thanh ghi (µs/word)	1.3	0.3	0.38	0.2	0.3	0.125	0.125	0.125	0.04
Đáp ứng interrupt (µs)	7.3	6.1	8.6	3.36	6.2	2.52	3.5	2.52	3.5
Địa chỉ bộ nhớ	64K	1M	1M	1M	1M	16M	4G	4G	4G
Cách định địa chỉ	5	24	24	24	24	24	28	28	28
Coprocessor	0	8087	8087	8087	8087	80287	80287/ 80387	80287/ 80387	On chip
Số thanh ghi đa dụng	6	8	8	8	8	8	8	8	8
Số thanh ghi đoạn	0	4	4	4	4	4	6	6	6
Điều khiển interrupt	8259-A	8259 -A	8259-A	On chip	On chip	8259-A	8259-A	82335	µPLD
Timer – counter	8253	8253 /54	8253/ 54	On chip	On chip	8253/ 54	8253/5 4	8253/5 4	On chip

2. µP 8086/8088

2.1. Mô tả

2.1.1. Định thì chu kỳ bus

Mỗi chu kỳ bus bắt đầu bằng việc xuất địa chỉ bộ nhớ hoặc I/O port (chu kỳ xung nhịp T1). Với 8086 thì địa chỉ này có thể là địa chỉ bộ nhớ 20 bit, địa chỉ I/O gián tiếp 16 bit (thanh ghi DX) hay địa chỉ I/O trực tiếp 8 bit.

Bus điều khiển có 4 tín hiệu tác động mức thấp là $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ và $\overline{IOW}$.

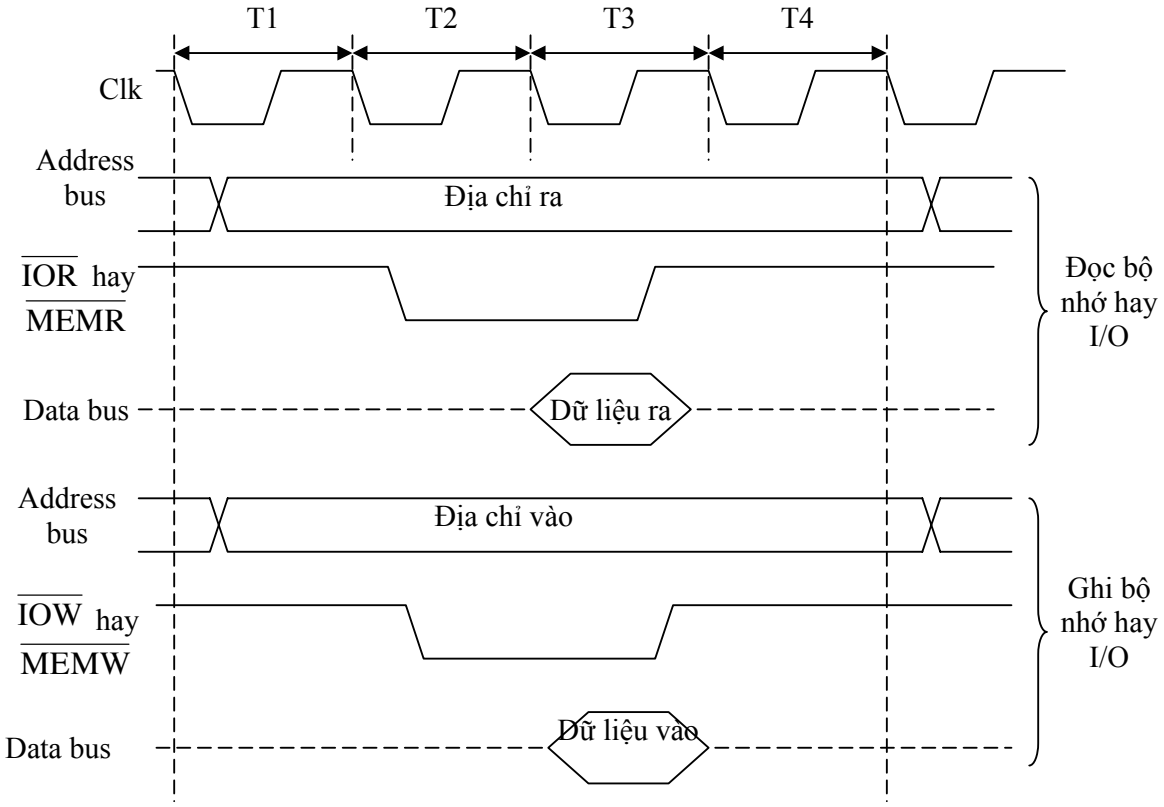
Các chuỗi sự kiện xảy ra trong một chu kỳ bus đọc bộ nhớ:

T1: µP xuất địa chỉ bộ nhớ 20 bit. Các đường dữ liệu không hoạt động và các đường điều khiển bị cấm

T2: Đường điều khiển $\overline{MEMR}$ xuống mức thấp. Đơn vị bộ nhớ ghi nhận chu kỳ bus này là quá trình đọc bộ nhớ và đặt byte hay word có địa chỉ đó lên data bus.

T3: µP đặt cấu hình để các đường data bus là nhập. Trạng thái này chủ yếu để bộ nhớ có thời gian tìm kiếm byte hay word dữ liệu

T4: µP đợi dữ liệu trên data bus. Do đó, nó thực hiện chốt data bus và giải phóng các đường điều khiển đọc bộ nhớ. Quá trình này sẽ kết thúc chu kỳ bus.

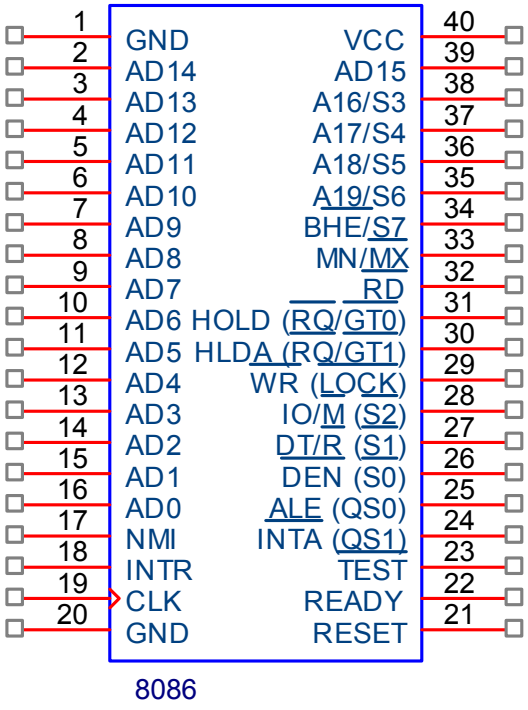


Hình 2.1 – Định thì chu kỳ bus

Trong một chu kỳ bus, µP có thể thực hiện đọc I/O, ghi I/O, đọc bộ nhớ hay ghi bộ nhớ. Các đường address bus và control bus dùng để xác định địa chỉ bộ nhớ hay I/O và hướng truyền dữ liệu trên data bus.

Chú ý rằng µP điều khiển tất cả các quá trình trên nên bộ nhớ bắt buộc phải cung cấp được dữ liệu vào lúc $\overline{MEMR}$ lên mức cao trong trạng thái T4. Nếu không, µP sẽ đọc dữ liệu ngẫu nhiên không mong muốn trên data bus. Để giải quyết vấn đề này, ta có thể dùng thêm các trạng thái chờ (wait state).

2.1.2. Mô tả chân



Hình 2.2 – Sơ đồ chân của 8086

8086 có bus địa chỉ 20 bit, bus dữ liệu 16 bit, 3 chân nguồn và 17 chân dùng cho các chức năng điều khiển. Tuy nhiên, ta có thể dùng kỹ thuật ghép kênh thời gian (time multiplexing) để cho phép một chân có nhiều chức năng nên các chân sẽ được phân ra:

- 16 chân dữ liệu và địa chỉ (AD0 ÷ AD15): các chân này sẽ là các đường địa chỉ trong trạng thái T1 và dữ liệu trong các trạng thái T2 – T4.
- 4 chân địa chỉ và trạng thái
- 3 chân nguồn
- 17 chân định thì và điều khiển

8086 có thể hoạt động ở chế độ tối thiểu (minimum mode) hay chế độ tối đa (maximum mode). Chế độ tối thiểu chỉ dùng cho các hệ thống µP đơn giản còn chế độ tối đa dùng cho các hệ thống phúc tạp hơn giao tiếp với các bộ nhớ và I/O riêng.

❖ **Các tín hiệu chung cho cả hai chế độ tối đa và tối thiểu:**

Bảng 2.2:

Chân	Chức năng	Loại
AD15 ÷ AD0	Bus dữ liệu / địa chỉ	2 chiều, 3 trạng thái
A19/S6 ÷ A16/S3	Địa chỉ / trạng thái	Ngõ ra 3 trạng thái
$\overline{MX}$	Điều khiển chế độ	Ngõ vào
$\overline{RD}$	Điều khiển đọc	Ngõ ra 3 trạng thái
$\overline{TEST}$	Chờ kiểm tra điều khiển	Ngõ vào
READY	Chờ trạng thái điều khiển	Ngõ vào
RESET	Reset hệ thống	Ngõ vào
NMI	Yêu cầu ngắt không thể che	Ngõ vào
INTR	Yêu cầu ngắt	Ngõ vào
CLK	Xung nhịp hệ thống	Ngõ vào
VCC	+5V	Ngõ vào
GND	GND	Ngõ vào

❖ **Các tín hiệu chỉ dùng trong chế độ tối thiểu:**

Bảng 2.3:

Chân	Chức năng	Loại
HOLD	Yêu cầu giữ	Ngõ vào
HLDA	Ghi nhận giữ	Ngõ vào
$\overline{WR}$	Điều khiển ghi	Ngõ ra 3 trạng thái
IO/$\overline{M}$	Điều khiển I/O và bộ nhớ	Ngõ ra 3 trạng thái
DT/$\overline{R}$	Truyền / nhận dữ liệu	Ngõ ra 3 trạng thái
$\overline{DEN}$	Cho phép dữ liệu	Ngõ ra 3 trạng thái
$\overline{BHE}$/S7	Đường trạng thái	Ngõ ra 3 trạng thái
ALE	Cho phép chốt địa chỉ	Ngõ ra
$\overline{INTA}$	Ghi nhận ngắt	Ngõ ra

❖ **Các tín hiệu chỉ dùng trong chế độ tối đa:**

Bảng 2.4:

Chân	Chức năng	Loại
$\overline{RQ}$/$\overline{GT1,0}$	Yêu cầu / cấp bus	2 chiều
$\overline{LOCK}$	Điều khiển khóa ưu tiên bus	Ngõ ra 3 trạng thái
$\overline{S2} \div \overline{S0}$	Trạng thái chu kỳ bus	Ngõ ra 3 trạng thái
QS1, QS2	Trạng thái hàng lệnh	Ngõ ra

❖ **Trạng thái bus:**

Bảng 2.5:

Ngõ vào trạng thái			Chu kỳ CPU
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	0	0	Ghi nhận ngắt
0	0	1	Đọc I/O port
0	1	0	Ghi I/O port
0	1	1	Ngừng
1	0	0	Nhận lệnh
1	0	1	Đọc bộ nhớ
1	1	0	Ghi bộ nhớ
1	1	1	Thụ động

❖ **Trạng thái hàng lệnh:**

Bảng 2.6:

QS1	QS0	Trạng thái hàng lệnh
0	0	Không hoạt động
0	1	Lấy byte đầu tiên của lệnh
1	0	Hàng rỗng
1	1	Lấy byte kế tiếp

❖ **Nguồn cung cấp và xung nhịp (VCC, GND và CLK):**

- 8086 sử dụng nguồn cấp điện +5V và có 2 chân đất.
- Dòng điện cực đại là 340 mA (10 mA cho loại CMOS).
- Xung nhịp dùng dạng xung chữ nhật có chu kỳ với thời gian cạnh lên và xuống nhỏ hơn 10 ns.
- Tiêu hao công suất và tần số xung nhịp cực đại:

❖ **Các chân trạng thái trong chế độ tối đa (S0, S1 và S2 - status):**

Các chân này sử dụng bởi bộ điều khiển bus 8288 để tạo các tín hiệu điều khiển như bảng 2.5.

❖ **Các chân điều khiển bus (HOLD, HLDA, $\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$, $\overline{LOCK}$):**

Chế độ tối thiểu:

- HOLD (giữ): ngõ vào tác động mức cao làm cho µP hở mạch tất cả các bus của nó, tách µP khỏi bộ nhớ của nó và I/O để cho phép thiết bị khác xử lý

bus hệ thống. Quá trình này gọi là truy xuất bộ nhớ trực tiếp (DMA – Direct Memory Access).

- HLDA (Hold acknowledge): ghi nhận yêu cầu DMA đối với bộ điều khiển DMA.

Chế độ tối đa:

- $\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$ (Request / Grant): các chân này dùng cả hai chức năng vào (nhận yêu cầu) và ra (chấp nhận yêu cầu). Khi một thiết bị muốn lấy điều khiển của bus cục bộ, nó sẽ phát yêu cầu bằng cách đưa tín hiệu mức thấp vào chân yêu cầu. Sau khi nhận yêu cầu, 8086 sẽ ở trạng thái HOLD và gởi tín hiệu chấp nhận ra chân này. Ở đây, chân $\overline{RQ}/\overline{GT0}$ có độ ưu tiên cao hơn chân $\overline{RQ}/\overline{GT1}$.
- $\overline{LOCK}$: báo cho các thiết bị khác biết không thể lấy điều khiển của bus cục bộ.

❖ **Các chân ngắt (NMI, INTR và $\overline{INTA}$):**

INTR và NMI là các yêu cầu ngắt khởi động bằng phần cứng, làm việc chính xác như các ngắt mềm. NMI (Non-Maskable Interrupt) là ngõ vào tác động cạnh lên. NMI là ngắt không thể che được và luôn được phục vụ, thường dùng cho các sự kiện như hư nguồn hay các lỗi bộ nhớ. INTR tác động mức cao và có thể bị che bằng cách xoá cờ IF trong thanh ghi cờ (xem 2.3.4) bằng lệnh CLI.

Khi NMI tích cực, điều khiển sẽ được chuyển đến địa chỉ chứa trong các vị trí 00008h ÷ 0000Bh. Khi INTR tích cực, chu kỳ ghi nhận ngắt (interrupt acknowledge cycle) được thực hiện. Quá trình này giống như chu kỳ đọc bộ nhớ ngoại trừ $\overline{INTA}$ tích cực thay vì $\overline{RD}$. Thiết bị tạo ngắt sẽ đặt một giá trị 8 bit vào data bus và chuyển điều khiển đến vị trí giá trị × 4 đến giá trị × 4 + 3.

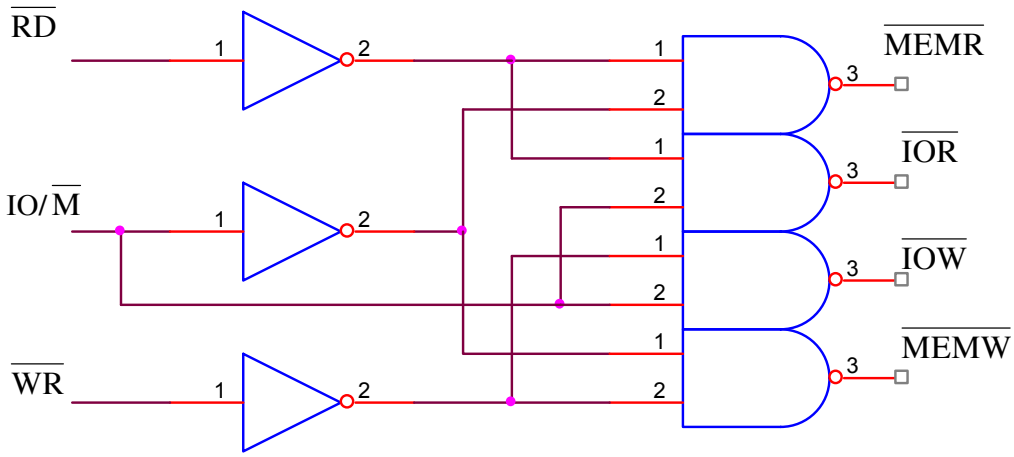
❖ **Chân RESET**: hoạt động khi có xung tác động mức cao, dùng để khởi động lại (P. Sau khi khởi động, (P sẽ đọc lệnh tại địa chỉ FFFF0h. RESET được sử dụng khi hệ thống có sự cố.

❖ **Các chân điều khiển bus (READY, $\overline{RD}$, ALE, $\overline{DEN}$, DT/$\overline{R}$, $\overline{WR}$ và IO/$\overline{M}$):**

Trong các chân điều khiển này, chỉ có hai chân READY và $\overline{RD}$ làm việc ở chế độ tối đa.

- Chân READY: ngõ vào READY được lấy mẫu ở cạnh lên của xung nhịp T2. Nếu chân này ở mức thấp (không sẵn sàng) thì sẽ thêm vào một chu kỳ T3 nữa. Chu trình này sẽ tiếp tục cho đến khi nào chân READY lên mức cao. Ngõ vào này thường được điều khiển bởi thiết bị bộ nhớ chậm, không thể cung cấp dữ liệu kịp thời cho μP.
- Chân IO/$\overline{M}$ (IO/Memory – Xuất nhập /Bộ nhớ): xác định chu kỳ bus hiện hành làm việc với bộ nhớ (mức thấp) hay I/O (mức cao).

- Chân $\overline{RD}$ (Read): tín hiệu tác động mức thấp chỉ chiều truyền dữ liệu từ bộ nhớ hay I/O đến µP. Ta có thể kết hợp với tín hiệu này với IO/$\overline{M}$ để tạo các tín hiệu $\overline{MEMR}$ và $\overline{IOR}$. Nó được xuất ra trong trạng thái T2 và lấy đi trong trạng thái T4. Thiết bị bộ nhớ hay I/O giả sử là đã đặt byte hay word vào các đường dữ liệu khi $\overline{RD}$ trở về mức cao.
- Chân $\overline{WR}$ (Write): tín hiệu này ngược với $\overline{RD}$, nó xác định chiều truyền dữ liệu từ µP đến I/O hay bộ nhớ.

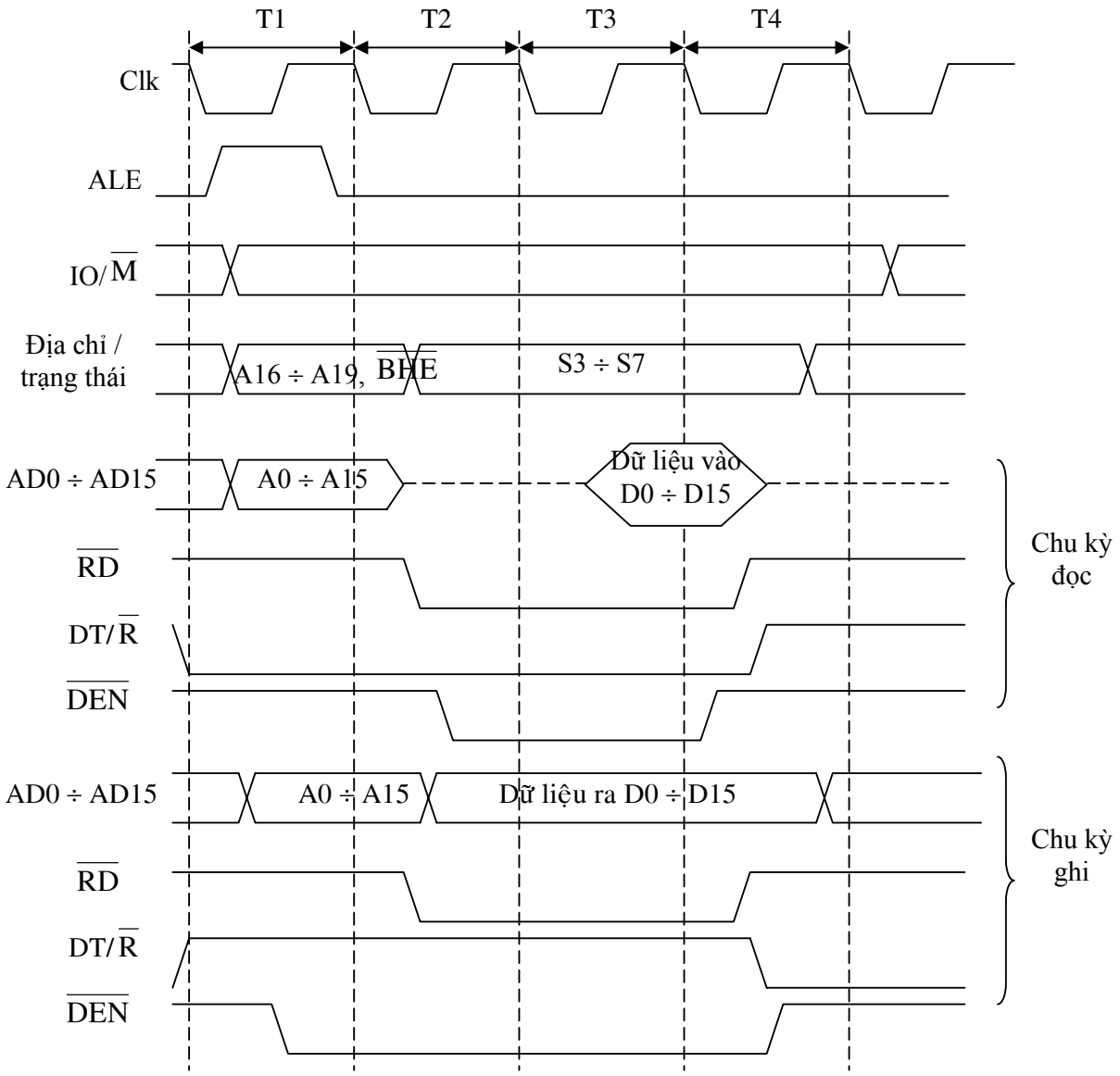


Hình 2.3 – Tạo tín hiệu điều khiển bộ nhớ và I/O

- Chân ALE (Address Latch Enable - cho phép chốt địa chỉ): tín hiệu ra trên chân này có thể dùng để phân kênh các đường địa chỉ, dữ liệu và trạng thái trên AD0 ÷ AD15, A16/S3 ÷ A19/S6 và $\overline{BHE}$/S7. Mọi chu kỳ bắt đầu với xung ALE trong trạng thái T1. Địa chỉ 20 bit được bảo đảm sẽ hợp lệ khi ALE chuyển từ mức cao xuống mức thấp.

- Chân $\overline{DEN}$ (Data Enable – cho phép dữ liệu): tín hiệu này được dùng với DT/$\overline{R}$ để cho phép nối các bộ đệm hai chiều vào data bus. Nó ngăn ngừa sự tranh chấp bus bằng cách cấm các bộ đệm dữ liệu cho đến trạng thái T2 khi các đường dữ liệu / địa chỉ không còn lưu trữ địa chỉ của bộ nhớ hay I/O.

- Chân DT/$\overline{R}$ (Data transmit/receive – truyền/nhận dữ liệu): dùng để điều khiển chiều của luồng dữ liệu qua các bộ đệm (nếu có) vào bus dữ liệu của hệ thống. Khi ở mức thấp, nó chỉ thực hiện tác vụ đọc và khi ở mức cao nó chỉ thực hiện tác vụ ghi.



Hình 2.4 – Các chu kỳ đọc và ghi của 8086

❖ **Các chân trạng thái (AD16/S3 ÷ AD19/S6 và $\overline{BHE}$/S7):**

5 tín hiệu trạng thái này được xuất ra trong các trạng thái T2 ÷ T4, dùng cho các mục đích kiểm tra. Bit S7 là bit trạng thái dư (không dùng), bit S6 luôn bằng 0, S5 mô tả trạng thái của cờ ngắt IF còn S3, S4 dùng để xác định đoạn đang sử dụng:

Bảng 2.7:

S4	S3	Đoạn
0	0	Thêm
0	1	Stack
1	0	Mã (hay không)
1	1	Dữ liệu

Tín hiệu $\overline{BHE}$/S7 (Bus High Enable) chỉ được xuất trong trạng thái T1. Khi chân này ở mức thấp, nó sẽ chỉ AD8 ÷ AD15 liên quan đến việc truyền dữ liệu. Quá trình này có thể xảy ra đối với các truy xuất bộ nhớ, I/O hay truy xuất 1 byte dữ liệu từ địa chỉ lẻ.

❖ **Bus dữ liệu (AD0 ÷ AD15):**

16 chân này tạo thành bus dữ liệu hai chiều. Các đường này chỉ hợp lệ trong các trạng thái T2 ÷ T4. Trong trạng thái T1, chúng giữ 16 bit thấp của địa chỉ bộ nhớ hoặc I/O.

❖ **Bus địa chỉ (AD0 ÷ AD15 và AD16/S3 ÷ AD19/S6):**

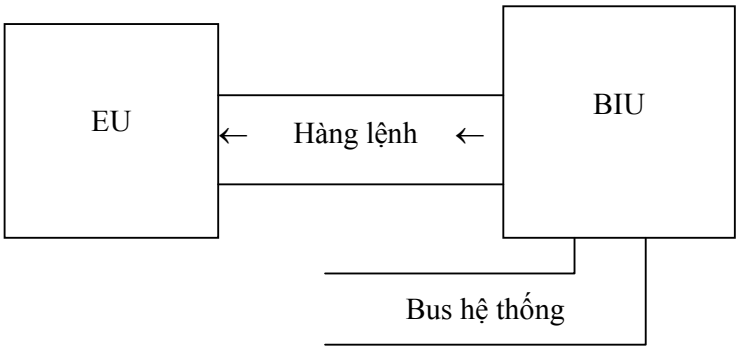
20 chân này tương ứng với bus địa chỉ 20 bit và cho phép µP truy xuất 1 MB vị trí bộ nhớ. Các đường ra này chỉ hợp lệ trong trạng thái T1, chuyển thành các đường dữ liệu và trạng thái trong trạng thái T2 ÷ T4.

❖ **Chọn chế độ $\overline{MX}$:**

Chân này dùng để chọn chế độ hoạt động cho 8086, nếu ở mức cao thì sẽ hoạt động ở chế độ tối thiểu còn ở mức thấp thì sẽ hoạt động ở chế độ tối đa.

2.2. Kiến trúc nội

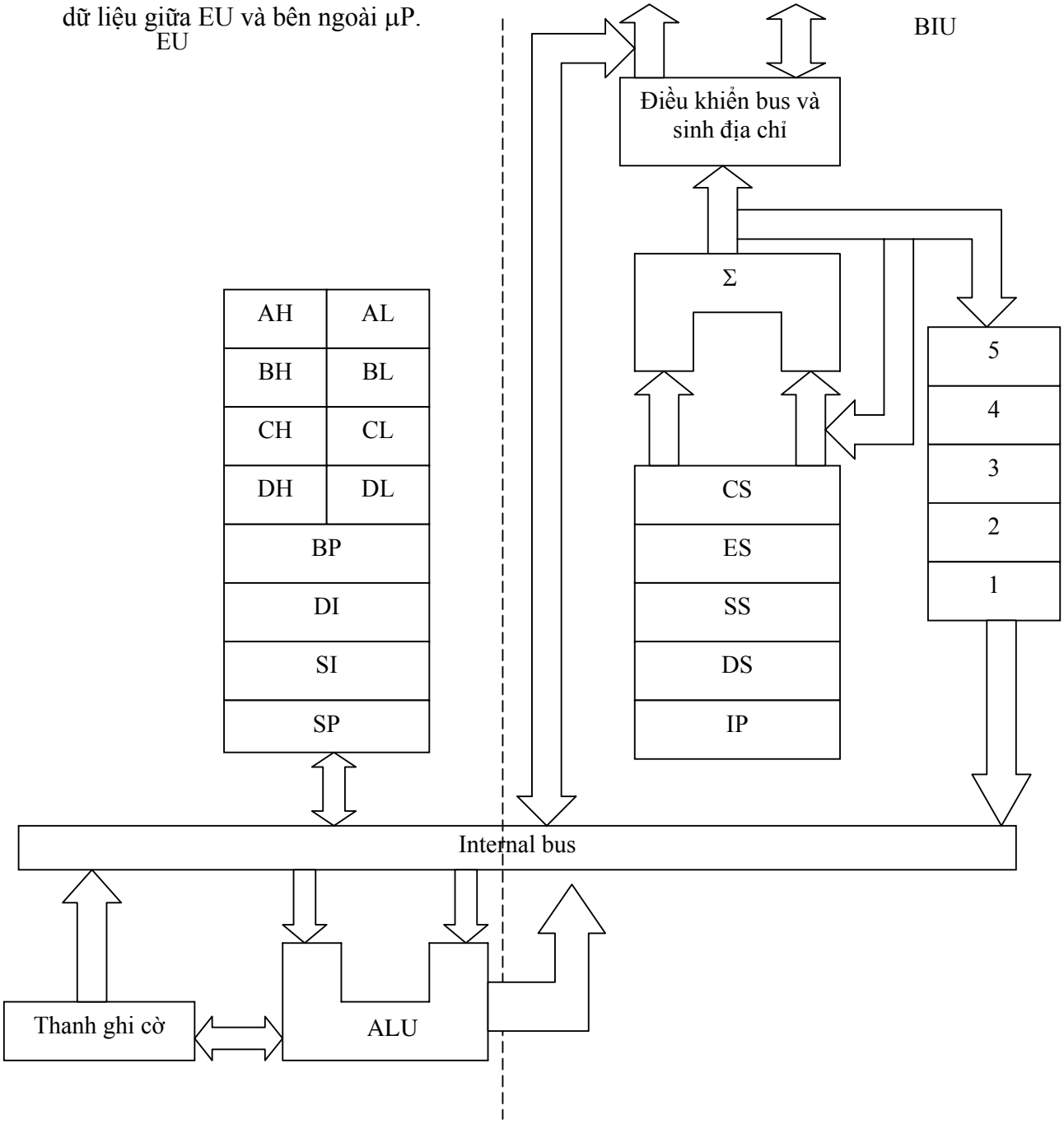
µP có khả năng thực hiện các tác vụ dữ liệu theo tập lệnh bên trong. Một lệnh được ghi nhận bằng mã đã được định nghĩa trước, gọi là mã lệnh (opcode). Trước khi thực thi một lệnh, µP phải nhận được mã lệnh từ bộ nhớ chương trình của nó. Quá trình xử lý này gọi là chu kỳ nhận lệnh (fetch cycle). Một khi các mã được nhận và được giải mã thì mạch bên trong µP có thể tiến hành thực thi (execute) mã lệnh.



Hình 2.5 – Kiến trúc nội của µP 8086

BIU (Bus Interface Unit – đơn vị giao tiếp bus) nhận các mã lệnh từ bộ nhớ và đặt chúng vào hàng chờ lệnh. EU (Execute Unit – đơn vị thực thi) sẽ giải mã và thực hiện các lệnh trong hàng. Chú ý rằng các đơn vị EU và BIU làm việc độc lập với nhau nên BIU có khả năng đang nhận một lệnh mới trong khi EU đang thực thi lệnh trước đó. Khi EU đã thực hiện xong lệnh, nó sẽ lấy mã lệnh kế tiếp trong hàng đợi lệnh (instruction queue).

Kiến trúc nội của µP 8086 ở hình 2.2. Nó có 2 bộ xử lý riêng: BIU và EU. BIU cung cấp các chức năng phần cứng, bao gồm tạo các địa chỉ bộ nhớ và I/O để chuyển dữ liệu giữa EU và bên ngoài µP.



Hình 2.6 – Kiến trúc nội của 8086

EU nhận các mã lệnh chương trình và dữ liệu từ BIU, thực thi các lệnh này và chứa các kết quả trong các thanh ghi. Ngoài ra, dữ liệu cũng có thể chứa trong một vị trí bộ nhớ hay được ghi vào thiết bị xuất. Chú ý rằng EU không có bus hệ thống nên phải thực hiện nhận và xuất tất cả các dữ liệu của nó thông qua BIU.

Sự khác biệt giữa µP 8086 và 8088 là BIU. Trong 8088, đường bus dữ liệu là 8 bit trong khi của 8086 là 16 bit. Ngoài ra hàng lệnh của 8088 dài 4 byte trong khi của 8086 là 6 byte.

Tuy nhiên do EU giữa hai loại µP này giống nhau nên ***các chương trình viết cho 8086 có thể chạy được trên 8088 mà không cần thay đổi gì cả***.

Quá trình nhận lệnh và thực thi lệnh:

1/ BIU xuất nội dung của thanh ghi con trỏ lệnh IP (Instruction Pointer) ra bus địa chỉ để chọn byte hay word đọc vào BIU.

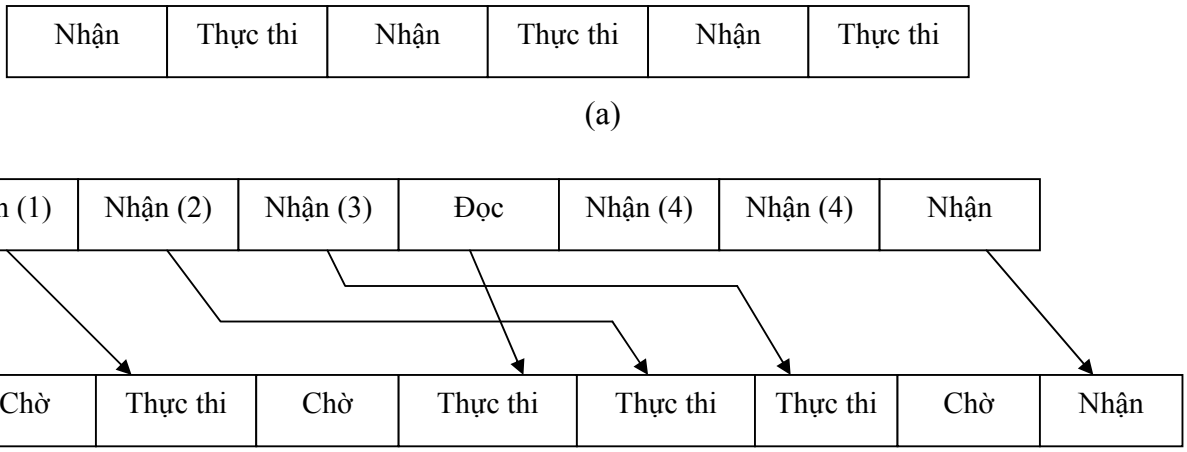
2/ Thanh ghi IP được tăng thêm 1 để chuẩn bị nhận lệnh kế.

3/ Khi lệnh ở trong BIU, nó được đưa sang hàng lệnh (queue). Đây là một thanh ghi lưu trữ dạng FIFO (First In First Out – Vào trước ra trước), dùng cơ chế xử lý xen kẽ liên tục các dòng mã lệnh (kỹ thuật đường ống – pipelining).

4/ Giả sử ban đầu hàng lệnh trống, EU sẽ không làm gì cả cho đến khi bắt đầu xuất hiện một lệnh trong hàng, EU sẽ lấy lệnh ra khỏi hàng và bắt đầu thực thi lệnh đó.

5/ Trong khi EU đang thực thi lệnh, BIU tiến hành nhận lệnh mới. Tuỳ theo thời gian thực thi lệnh mà BIU có thể đưa vào hàng lệnh nhiều lệnh mới trước khi EU thực hiện lệnh xong và tiếp tục lấy lệnh mới.

BIU được lập trình để có thể nhận một lệnh mới bất kỳ lúc nào hàng lệnh có chỗ cho 1 byte (8088) hay 2 byte (8086). Lợi ích của phương pháp xử lý theo cơ chế pipeline là EU có thể thực thi các lệnh gần như liên tục thay vì phải đợi BIU nhận thêm lệnh mới.



(1): lệnh thực thi không cần dữ liệu trong hàng
(2): lệnh thực thi cần dữ liệu trong hàng
(3): lệnh nhảy
(4): các lệnh bị bỏ qua do lệnh nhảy

Hình 2.7

(a) µP thông thường dùng chu kỳ nhận và thực thi lệnh tuần tự
(b) kiến trúc dạng pipeline của 8086/8088 cho phép thực thi các lệnh mà không bị trễ do quá trình nhận lệnh

Có 3 điều kiện làm cho EU ở chế độ chờ:

- Điều kiện thứ nhất xảy ra khi lệnh cần truy xuất đến một vị trí bộ nhớ không ở trong hàng. BIU phải treo quá trình nhận lệnh và xuất ra địa chỉ của ô nhớ này. Sau khi truy xuất bộ nhớ, EU có thể tiếp tục quá trình thực thi lệnh từ hàng lệnh và BIU có thể tiếp tục đưa các lệnh vào hàng.

- Điều kiện thứ hai xảy ra khi lệnh được thực thi là lệnh nhảy (jump). Trong trường hợp này, thay vì dùng địa chỉ lệnh kế tiếp, ta phải chuyển đến địa chỉ mới (không tuần tự). Tuy nhiên, BIU vẫn luôn đặt các lệnh theo tuần tự và do đó sẽ lưu các lệnh không sử dụng. Trong khi nhận lệnh kế tiếp tại địa chỉ do lệnh jump chỉ đến, EU phải đợi và tất cả các byte trong hàng phải bỏ.

- Điều kiện thứ ba có thể làm BIU treo quá trình nhận lệnh đó là khi thực thi các lệnh có thời gian thực thi lớn. Giả sử như lệnh AAM (ASCII Adjust for Multiplication) cần 83 chu kỳ xung nhịp để hoàn tất trong khi đó với 4 chu kỳ xung nhịp cho quá trình nhận lệnh thì hàng sẽ bị đầy. Như vậy BIU phải đợi cho đến khi lệnh được thực hiện xong và EU nhận mã lệnh từ hàng thì mới có thể tiếp tục quá trình nhận lệnh.

2.3. Các thanh ghi

µP 8086/8088 có tất cả 14 thanh ghi nội. Các thanh ghi này có thể phân loại như sau:

- Thanh ghi dữ liệu (data register)
- Thanh ghi chỉ số và con trỏ (index & pointer register)
- Thanh ghi đoạn (segment register)
- Thanh ghi trạng thái và điều khiển (status & control register)

2.3.1. Các thanh ghi dữ liệu

Các thanh ghi dữ liệu gồm có các thanh ghi 16 bit AX, BX, CX và DX trong đó nửa cao và nửa thấp của mỗi thanh ghi có thể định địa chỉ một cách độc lập. Các nửa thanh ghi này (8 bit) có tên là AH và AL, BH và BL, CH và CL, DH và DL.

Các thanh ghi này được sử dụng trong các phép toán số học và logic hay trong quá trình chuyển dữ liệu.

Bảng 2.8:

Thanh ghi	Sử dụng trong
AX	MUL, IMUL (toán hạng nguồn kích thước word) DIV, IDIV (toán hạng nguồn kích thước word) IN (nhập word) OUT (xuất word) CWD Các phép toán xử lý chuỗi (string)
AL	MUL, IMUL (toán hạng nguồn kích thước byte) DIV, IDIV (toán hạng nguồn kích thước byte) IN (nhập byte) OUT (xuất byte) XLAT AAA, AAD, AAM, AAS (các phép toán ASCII) CBW (đổi sang word) DAA, DAS (số thập phân) Các phép toán xử lý chuỗi (string)

AH	MUL, IMUL (toán hạng nguồn kích thước byte) DIV, IDIV (toán hạng nguồn kích thước byte) CBW (đổi sang word)
BX	XLAT
CX	LOOP, LOOPE, LOOPNE Các phép toán string với tiếp dầu ngữ REP
CL	RCR, RCL, ROR, ROL (quay với số đếm byte) SHR, SAR, SAL (dịch với số đếm byte)
DX	MUL, IMUL (toán hạng nguồn kích thước word) DIV, IDIV (toán hạng nguồn kích thước word)

AX (ACC – Accumulator): thanh ghi tích luỹ
BX (Base): thanh ghi cơ sở
CX (Count): đếm
DX (Data): thanh ghi dữ liệu

2.3.2. Các thanh ghi chỉ số và con trỏ

Bao gồm các thanh ghi 16 bit SP, BP, SI và DI, thường chứa các giá trị offset (độ lệch) cho các phần tử định địa chỉ trong một phân đoạn (segment). Chúng có thể được sử dụng trong các phép toán số học và logic. Hai thanh ghi con trỏ (SP – Stack Pointer và BP – Base Pointer) cho phép truy xuất dễ dàng đến các phần tử đang ở trong ngăn xếp (stack) hiện hành. Các thanh ghi chỉ số (SI – Source Index và DI – Destination Index) được dùng để truy xuất các phần tử trong các đoạn dữ liệu và doạn thêm (extra segment). Thông thường, các thanh ghi con trỏ liên hệ đến đoạn stack hiện hành và các thanh ghi chỉ số liên hệ đến doạn dữ liệu hiện hành. SI và DI dùng trong các phép toán chuỗi.

2.3.3. Các thanh ghi đoạn

Bao gồm các thanh ghi 16 bit CS (Code segment), DS (Data segment), SS (stack segment) và ES (extra segment), dùng để định địa chỉ vùng nhớ 1 MB bằng cách chia thành 16 đoạn 64 KB.

Tất cả các lệnh phải ở trong đoạn mã hiện hành, được định địa chỉ thông qua thanh ghi CS. Offset (độ lệch) của mã được xác định bằng thanh ghi IP. Dữ liệu chương trình thường được đặt ở đoạn dữ liệu, định vị thông qua thanh ghi DS. Stack định vị thông qua thanh ghi SS. Thanh ghi đoạn thêm có thể sử dụng để định địa chỉ các toán hạng, dữ liệu, bộ nhớ và các phần tử khác ngoài đoạn dữ liệu và stack hiện hành.

2.3.4. Các thanh ghi điều khiển và trạng thái

Thanh ghi con trỏ lệnh IP (Instruction Pointer) giống như bộ đếm chương trình (Program Counter). Thanh ghi điều khiển này do BIU quản lý nhằm lưu trữ offset từ bắt đầu đoạn mã đến lệnh thực thi kế tiếp. Ta không thể xử lý trực tiếp trên thanh ghi IP.

Thanh ghi cờ (Flag register) hay từ trạng thái 16 bit chứa 3 bit điều khiển (TF, IF và DF) và 6 bit trạng thái (OF, SF, ZF, AF, PF và CF) còn các bit còn lại mà 8086/8088 không sử dụng thì không thể truy xuất được.

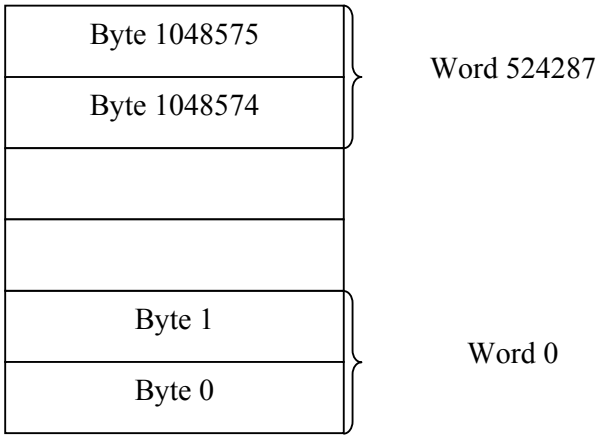
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

- OF (Overflow - tràn): OF = 1 xác định tràn số học, xảy ra khi kết quả vượt ra ngoài phạm vi biểu diễn
- DF (Direction- hướng): xác định hướng chuyển string, DF = 1 khi µP làm việc với string theo thứ tự từ phải sang trái.
- IF (Interrupt - ngắt): cho phép hay cấm các interrupt có mặt nạ
- TF (Trap - bẫy): đặt µP vào chế độ từng bước, dùng cho các chương trình gỡ rối (debugger).
- SF (Sign - dấu): dùng để chỉ các kết quả số học là số dương (SF = 0) hay âm (SF = 1).
- ZF (Zero): = 1 nếu kết quả của phép toán trước là 0.
- AF (Auxiliary – nhớ phụ): dùng trong các số thập phân để chỉ nhớ từ nửa byte thấp hay mượn từ nửa byte cao.
- PF (Parity): PF = 1 nếu kết quả của phép toán là có tổng số bit 1 là chẵn (dùng để kiểm tra lỗi truyền dữ liệu)
- CF (Carry): CF = 1 nếu có nhớ hay mượn từ bit cao nhất của kết quả. Cờ này cũng dùng cho các lệnh quay.

2.4. Phân đoạn bộ nhớ

Ta biết rằng dù 8086 là µP 16 bit (có bus dữ liệu 16 bit) nhưng vẫn dùng bộ nhớ theo các byte. Điều này cho phép µP làm việc với byte cũng như word, nó rất quan trọng trong giao tiếp với các thiết bị I/O như máy in, thiết bị đầu cuối và modem (chúng được thiết kế để chuyển dữ liệu mã hoá ASCII 7 hay 8 bit). Ngoài ra, nhiều mã lệnh của 8086/8088 có chiều dài 1 byte nên cần phải truy xuất được các byte riêng biệt để có thể xử lý các lệnh này.

8086/8088 có bus địa chỉ 20 bit nên có thể cho phép truy xuất 2^{20} = 1048576 địa chỉ bộ nhớ khác nhau.

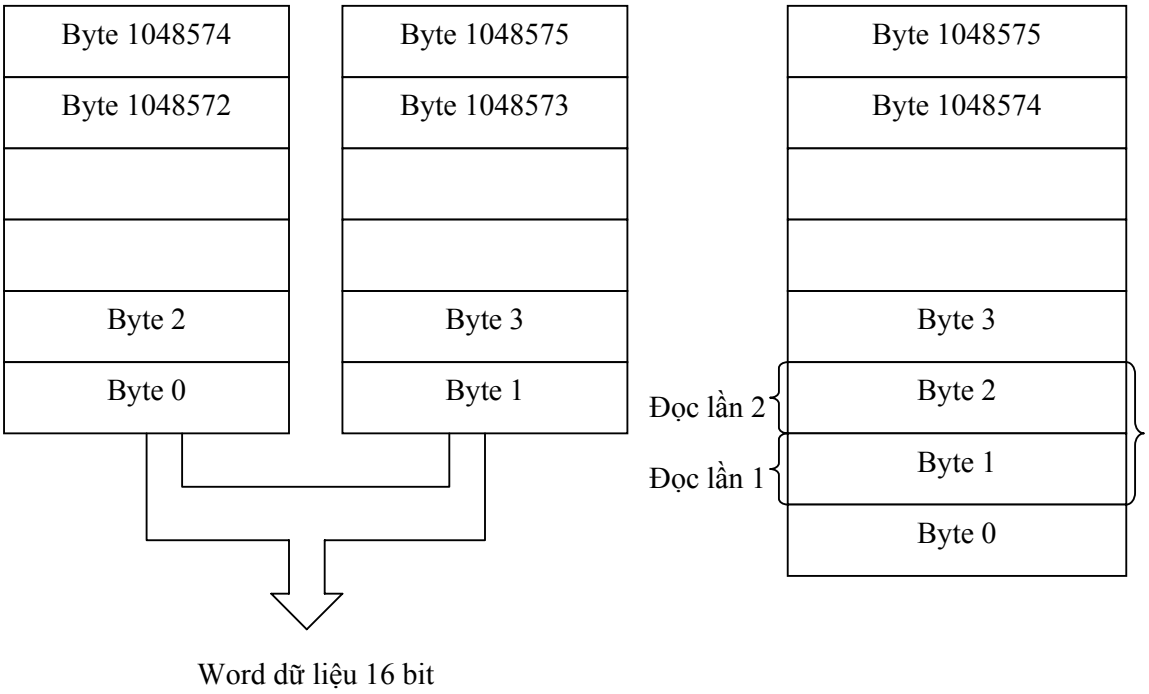


Hình 2.8 – Vùng nhớ của 8086/8088 có 1048576 byte hay 524288 word

Để thực hiện đọc 16 bit từ bộ nhớ, 8086 sẽ thực hiện đọc đồng thời byte có địa chỉ lẻ và byte có địa chỉ chẵn. Do đó, 8086 tổ chức bộ nhớ thành các bank chẵn và lẻ. Theo hình 2.8, ta có thể thấy rằng các word luôn bắt đầu tại địa chỉ chẵn nhưng ta vẫn

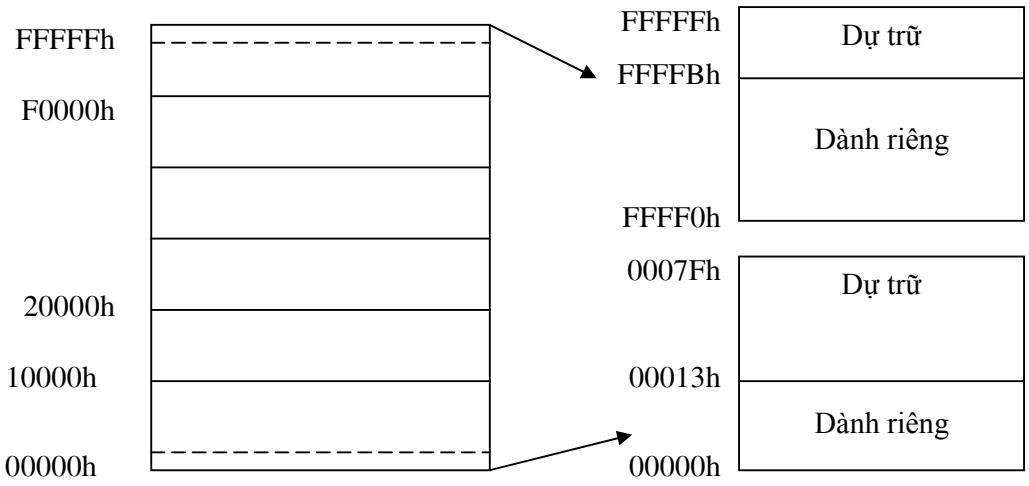
có thể đọc word có địa chỉ lẻ bằng cách thực hiện 2 chu kỳ đọc bộ nhớ: một chu kỳ đọc byte thấp và một chu kỳ đọc byte cao. Điều này sẽ làm chậm tốc độ xử lý.

Đối với 8088 thì do bus dữ liệu 8 bit nên dù word có địa chỉ chẵn hay lẻ, nó cũng cần phải thực hiện 2 chu kỳ đọc hay ghi bộ nhớ và giao tiếp với bộ nhớ như một bank.



Hình 2.9 – Đọc word địa chỉ chẵn và địa chỉ lẻ

Ngoài ra bộ nhớ cũng chia thành 16 khối, mỗi khối có kích thước 64 KB, bắt đầu ở địa chỉ 00000h và kết thúc ở FFFFFh. Địa chỉ bắt đầu mỗi khối sẽ tăng lên 1 ở số hex có ý nghĩa nhiều nhất khi thay đổi từ khối này sang khối kia. Ví dụ như khối 00000h → 10000h → 20000h …



Hình 2.10 – Bảng bộ nhớ cho 8086/8088

❖ Các thanh ghi phân đoạn:

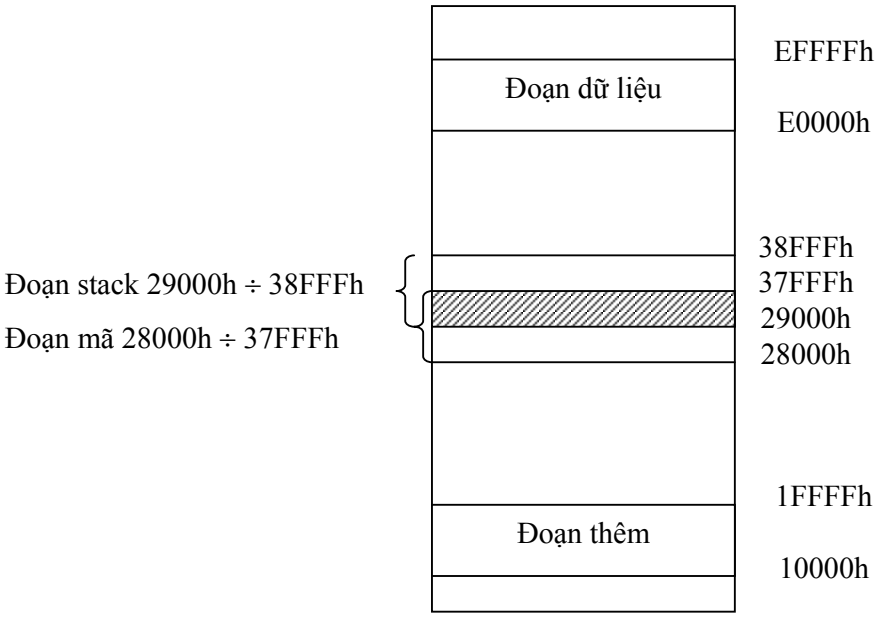
8086/8088 định nghĩa 4 khối bộ nhớ 64KB: đoạn mã (code segment) giữ các mã lệnh chương trình, đoạn ngăn xếp (stack segment) lưu các địa chỉ sẽ trả về từ các chương trình con (subroutine) hay trình phục vụ ngắt (interrupt subroutine), đoạn dữ liệu (data segment) lưu trữ dữ liệu cho chương trình và đoạn thêm (extra segment) thường dùng cho các dữ liệu dùng chung.

Các thanh ghi đoạn (CS, DS, SS và ES) dùng để chỉ vị trí nền của mỗi đoạn. Các thanh ghi này có 16 bit trong khi địa chỉ bộ nhớ là 20 bit nên để xác dịnh vị trí bộ nhớ, ta sẽ thêm 4 bit 0 vào các bit thấp của thanh ghi đoạn. Giả sử như thanh ghi CS chứa giá trị 1111h thì nó sẽ chỉ tới địa chỉ nền là 11110h. Chú ý rằng địa chỉ bắt đầu một đoạn không thể tuỳ ý mà phải bắt đầu tại một địa chỉ chia hết cho 16. Nghĩa là 4 bit thấp phải là 0. Ta cũng chú ý rằng 4 đoạn có thể không tách rời nhau mà chồng lấp lên nhau và ta cũng có thể cho 4 giá trị của các thanh ghi đoạn bằng nhau nghĩa là 4 đoạn này trùng nhau.

VD: Thanh ghi DS có giá trị là 1000h thì địa chỉ nền là 10000h. Địa chỉ kết thúc tìm được bằng cách cộng địa chỉ nền với giá trị FFFFh (64K) → địa chỉ kết thúc là 10000h + FFFFh = 1FFFFh. Như vậy đoạn dữ liệu có địa chỉ từ 10000h = 1FFFFh.

Các vị trí bộ nhớ không được định nghĩa trong các đoạn hiện hành không thể truy xuất được. Muốn truy xuất đến các vị trí đó, ta phải định nghĩa lại một trong các thanh ghi đoạn sau cho đoạn phải chứa vị trí đó. Như vậy, tại một thời điểm bất kỳ ta chỉ có thể truy xuất tối đa 4 × 64 KB = 256 KB bộ nhớ. Nội dung của các thanh ghi đoạn chỉ có thể xác định thông qua phần mềm.

VD: Giả sử các thanh ghi đoạn có các giá trị CS = 2800h, DS = E000h, SS = 2900h và ES = 1000h. Ta có vị trí các đoạn trong bảng bộ nhớ như sau:



Hình 2.11 – Vị trí các phân đoạn theo giá trị các thanh ghi đoạn

❖ Địa chỉ logic và địa chỉ vật lý:

Các địa chỉ trong một đoạn thay đổi từ 0000h ÷ FFFFh, tương ứng với chiều dài đoạn là 64 KB. Một địa chỉ trong một đoạn được gọi là ***địa chỉ logic hay offset***. Ví dụ như địa chỉ logic 0010h của đoạn mã trong hình 2.11 sẽ có địa chỉ thật sự là 28000h + 0010h = 28010h. Địa chỉ này gọi là ***địa chỉ vật lý***.

Như vậy, địa chỉ vật lý chính là địa chỉ thật sự xuất hiện ở bus địa chỉ, nó có chiều dài 20 bit còn địa chỉ logic là độ lệch (offset) từ vị trí 0 của một đoạn cho trước.

VD: Giả sử xét các đoạn như hình 2.11. Địa chỉ vật lý tương ứng với địa chỉ logic 1000h trong đoạn stack là:

29000h + 1000h = 2A000h

Địa chỉ vật lý tương ứng với địa chỉ logic 2000h trong đoạn mã là:

28000h + 2000h = 2A000h

Ta thấy rằng có thể địa chỉ vật lý trùng nhau khi địa chỉ logic khác nhau nghĩa là một địa chỉ vật lý có thể có nhiều địa chỉ logic khác nhau.

Để chỉ địa chỉ logic 1000h trong đoạn mã, ta dùng ký hiệu CS:1000h. Tương tự như vậy cho các đoạn khác, nghĩa là địa chỉ logic 1111h trong đoạn dữ liệu sẽ là DS:1111h.

Mọi lệnh tham chiếu bộ nhớ sẽ có một thanh ghi đoạn mặc nhiên. Thanh ghi IP cung cấp địa chỉ offset khi truy xuất đến đoạn mã và BP cho đoạn stack. Ví dụ như IP = 1000h và CS = 2000h thì BIU sẽ truy xuất đến địa chỉ 20000h + 1000h = 21000h và nhận byte tại vị trí này.

Bảng 2.9:

Tham chiếu bộ nhớ	Đoạn mặc nhiên	Đoạn khác	Offset
Nhận lệnh	CS	Không	IP
Tác vụ stack	SS	Không	SP
Dữ liệu tổng quát	DS	CS,ES,SS	Địa chỉ hiệu dụng
Nguồn của string	DS	CS,ES,SS	SI
Đích của string	ES	Không	DI
BX dùng làm con trỏ	DS	CS,ES,SS	Địa chỉ hiệu dụng
BP dùng làm con trỏ	SS	CS,ES,SS	Địa chỉ hiệu dụng

VD: Ta sử dụng lệnh MOV [BP],AL với BP = 2C00h. Ở đây BP dùng làm con trỏ nên dùng đoạn stack. Giả sử các phân đoạn như hình 2.11 thì địa chỉ vật lý sẽ là 29000h + 2C00h = 2BC00h

❖ Định nghĩa các vị trí bộ nhớ:

Thông thường ít khi nào ta cần biết đến địa chỉ vật lý của một vị trí bộ nhớ mà ta chỉ quan tâm đến địa chỉ logic của nó mà thôi. Lý do là vì địa chỉ vật lý còn phải phụ thuộc vào nội dung của các thanh ghi đoạn ngay cả khi địa chỉ logic giữ không đổi như đã xét ở trên.

Khi viết các chương trình hợp ngữ, thường gán cho các địa chỉ logic bằng các nhãn (label) hay các tên (name). Ví dụ:

```
DATA      SEGMENT
SAMPLEB   DB    ?
DATA      ENDS
```

sẽ gán nhãn SAMPLE cho byte ở địa chỉ logic 0 trong đoạn dữ liệu. Các phát biểu này không phải là các lệnh µP mà chỉ là các lệnh giả (pseudo instruction) dùng cho các chương trình dịch.

Toán tử DATA SEGMENT báo cho chương trình dịch biết các lệnh theo sau sẽ nằm trong đoạn dữ liệu. Toán tử DB (Define Byte) gán cho nhãn SAMPLEB 1 byte trong đoạn dữ liệu. Ký hiệu ? xác định rằng không cần định nghĩa nội dung của byte đó. Do SAMPLEB là dòng đầu tiên nên nó sẽ có địa chỉ logic là 0. Phát biểu DATA ENDS kết thúc đoạn dữ liệu (ở đây chỉ dịnh nghĩa 1 byte). Trong trường hợp muốn định nghĩa 1 word, ta dùng toán tử DW (Define Word).

VD:

```
SAMPLEW   DW    1000h
```

Phát biểu này định nghĩa nhãn SAMPLEW ứng với vị trí word và nội dung của vị trí này là 1000h.

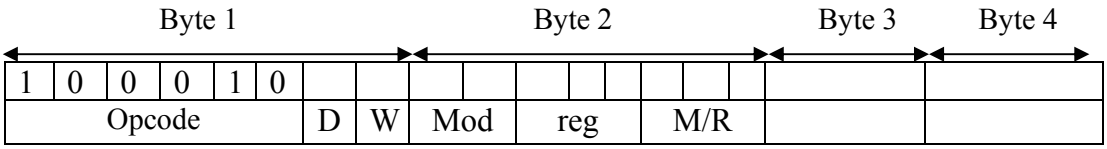
Ngoài ra, ta có thể dùng các toán tử DD định nghĩa từ kép (double word), DQ định nghĩa từ bộ bốn (8 byte) và DT định nghĩa 10 byte.

3. Cách mã hoá lệnh

Lệnh của bộ vi xử lý sẽ biểu diễn bằng các ký tự dưới dạng gợi nhớ (mnemonic) để có thể dễ dàng sử dụng. Đối với vi xử lý thì các lệnh được biểu diễn bằng các mã lệnh (opcode) nên sau khi nhận lệnh vi xử lý phải thực hiện giải mã lệnh rồi mới thực thi nó. Một lệnh vi xử lý có thể dài 1 byte hay nhiều byte. Nếu ta dùng 1 byte để mã hoá thì sẽ mã hoá được 256 lệnh khác nhau. Tuy nhiên do một lệnh không phải chỉ có một cách thực hiện nên ta không thể thực hiện đơn giản như trên.

Để tìm hiểu cách mã hoá lệnh, ta xét lệnh **MOV des,src** dùng để chuyển dữ liệu giữa hai thanh ghi hay một ô nhớ và một thanh ghi.

Lệnh MOV mã hoá như sau:



Để mã hóa lệnh MOV, ta cần dùng ít nhất là 2 byte trong đó 6 bit dùng cho mã lệnh.

Bit D xác định hướng truyền của dữ liệu, D = 0 xác định dữ liệu sẽ đi từ thanh ghi cho bởi 3 bit Reg, D = 1 xác định dữ liệu sẽ đi đến thanh ghi cho bởi 3 bit Reg.

Bit W xác định sẽ truyền 1 byte (W = 0) hay 1 word (W = 1).

3 bit Reg dùng để chọn thanh ghi sử dụng:

Bảng 2.10:

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

2 bit mod và 3 bit R/M (Register / Memory) dùng để xác định chế độ địa chỉ cho các toán hạng của lệnh.

Bảng 2.11:

MOD / R/M	00	01	10	11	
				W = 1	W = 0
000	[BX]+[SI]	[BX]+[SI]+addr8	[BX]+[SI]+addr16	AX	AL
001	[BX]+[DI]	[BX]+[DI]+addr8	[BX]+[DI]+addr16	CX	CL
010	[BP]+[SI]	[BP]+[SI] +addr8	[BP]+[SI] +addr16	DX	DL
011	[BP]+[DI]	[BP]+[DI] +addr8	[BP]+[DI] +addr16	BX	BL
100	[SI]	[SI] +addr8	[SI] +addr16	SP	AH
101	[DI]	[DI] +addr8	[DI] +addr16	BP	CH
110	addr16	[BP] +addr8	[BP] +addr16	SI	DH
111	[BX]	[BX] +addr8	[BX] +addr16	DI	BH

Tổng quát, 8086/8088 có khoảng 300 tác vụ có thể có trong tập lệnh của nó. Mỗi lệnh kéo dài từ 1 đến 6 byte. Từ ví dụ trên, ta thấy mã lệnh có các vùng:

- Vùng mã lệnh (opcode): chứa mã lệnh của lệnh sẽ thực thi
- Vùng thanh ghi (reg): chứa các thanh ghi sẽ thực hiện (bảng 2.10)
- Vùng chế độ (mod): (bảng 2.11)

00: toán hạng bộ nhớ trực tiếp nếu R/M = 110, ngược lại là toán hạng gián tiếp
01: toán hạng gián tiếp, độ dời 8 bit
10: toán hạng gián tiếp, độ dời 16 bit
11: sử dụng 2 thanh ghi, vùng R/M sẽ là vùng Reg

- Vùng thanh ghi / bộ nhớ R/M (Reg/Mem): (bảng 2.11)

4. Các cách định địa chỉ

Bảng 2.12:

Cách định địa chỉ	Mã đối tượng	Ví dụ			
		Từ gợi nhớ	Đoạn truy xuất	Hoạt động	Mô tả
Tức thời	B80010	MOV AX,1000h	Mã	AH ← 10h AL ← 00h	(1)
Thanh ghi	8BD1	MOV DX,CX	Trong µP	DX ← CX	(2)
Trực tiếp	8A260010	MOV AH,[1000h]	Döõ lieäu	AH ← [1000h]	(3)
Gián tiếp thanh ghi	8B04 FF25 FE4600 FF0F	MOV AX,[SI] JMP [DI] INC BYTE PTR [BP] DEC WORD PTR [BX]	Dữ liệu Dữ liệu Stack Dữ liệu	AL ← [SI]; AH ←[SI+1] IP←[DI+1:DI] [BP]←[BP]+1 [BX+1:BX]← [BX+1:BX]-1	(4)
Có chỉ số	8B4406 FF6506	MOV AX,[SI+6] JMP [DI+6]	Dữ liệu Dữ liệu	AL ← [SI+6]; AH ←[SI+7] IP←[DI+7:DI+6]	(5)
Có nền	8B4602 FF6702	MOV AX,[BP+2] JMP [BP+2]	Stack Dữ liệu	AL←[BP+2]; AH ←[BP+3] IP←[BX+3:BX+6]	(6)
Có nền và có chỉ số	8B00 FF21 FE02 FF0B	MOV AX,[BX+SI] JMP [BX+DI] INC BYTE PTR [BP+SI] DEC WORD PTR [BP+DI]	Dữ liệu Dữ liệu Stack Stack	AL←[BX+SI];AH←[BX+SI+1] IP←[BX+DI+1:BX+DI] [BP+SI]←[BP+SI]+1 [BP+DI+1:BP+DI]← [BP+DI+1:BP+DI]-1	(7)
Có nền và có chỉ số với độ dời	8B4005 FF6105 FE4205 FF4B05	MOV AX,[BX+SI+5] JMP [BX+DI+5] INC BYTE PTR [BP+SI+5] DEC WORD PTR [BP+DI+5]	Dữ liệu Dữ liệu Stack Stack	AL←[BX+SI+5] AH←[BX+SI+1] IP←[BX+DI+6:BX+DI+5] [BP+SI+5]←[BP+SI+5]+1 [BP+DI+6:BP+DI+5]← [BP+DI+6:BP+DI+5]-1	(8)
String	A4	MOVSB	Thêm, dữ liệu	[ES:DI] ← [DS:DI] Nếu DF = 0 thì SI ← SI + 1; DI ← DI + 1 Nếu DF = 1 thì SI ← SI - 1; DI ← DI - 1	(9)

- BYTE PTR và WORD PTR tránh lầm giữa truy xuất byte và word.
- Độ dời được cộng vào thanh ghi con trỏ hay nền là số nhị phân dạng bù 2.
- (1): nguồn dữ liệu trong lệnh
- (2): đích và nguồn là các thanh ghi của µP
- (3): địa chỉ bộ nhớ cung cấp trong lệnh
- (4): địa chỉ bộ nhớ cung cấp trong thanh ghi con trỏ hay chỉ số
- (5): địa chỉ bộ nhớ là tổng của thanh ghi chỉ số cộng với độ dời trong lệnh
- (6): địa chỉ bộ nhớ là tổng của thanh ghi BX hay BP cộng với độ dời trong lệnh
- (7): địa chỉ bộ nhớ là tổng của thanh ghi chỉ số và thanh ghi nền

- (8): địa chỉ bộ nhớ là tổng của thanh ghi chỉ số, thanh ghi nền và độ dời trong lệnh
- (9): địa chỉ nguồn bộ nhớ là thanh ghi SI trong đoạn dữ liệu và địa chỉ đích bộ nhớ là thanh ghi DI trong đoạn thêm

4.1. Định địa chỉ tức thời

Các lệnh dùng cách định địa chỉ tức thời lấy dữ liệu trong lệnh làm một phần của lệnh. Trong cách này, dữ liệu sẽ được chứa trong đoạn mã thay vì trong đoạn dữ liệu. Dữ liệu cho lệnh MOV AX,1000h được cung cấp tức thời sau mã lệnh B8. Chú ý rằng trong mã đối tượng byte dữ liệu cao đi sau byte dữ liệu thấp.

Cách định địa chỉ tức thời thường dùng để nạp một thanh ghi hay vị trí bộ nhớ với các dữ liệu ban đầu. Sau đó, các lệnh kế tiếp sẽ làm việc với các dữ liệu này. Tuy nhiên, cách định địa chỉ này không sử dụng được cho các thanh ghi đoạn.

4.2. Định địa chỉ thanh ghi

Một số lệnh chỉ làm công việc chuyển dữ liệu giữa các thanh ghi của µP. Ví dụ như MOV DX,CX sẽ chuyển dữ liệu từ thanh ghi CX vào thanh ghi DX. Ở đây ta không cần thực hiện tham chiếu bộ nhớ.

Ta có thể kết hợp cách định địa chỉ tức thời và định địa chỉ thanh ghi để nạp dữ liệu cho các thanh ghi đoạn.

VD:

```
MOV  AX, 1000h
MOV CS,AX
```

Sau khi thực hiện 2 lệnh này, giá trị của thanh ghi CS sẽ là 1000h.

4.3. Định địa chỉ trực tiếp

Ngoài 2 cách định địa chỉ trên, tất cả các cách định địa chỉ còn lại cho trong bảng 2.6 đều cần phải truy xuất đến bộ nhớ với ít nhất một toán hạng. Trong cách định địa chỉ trực tiếp, địa chỉ bộ nhớ được cung cấp trực tiếp như là một phần của lệnh. Ví dụ như lệnh MOV AH,[1000h] sẽ đưa nội dung chứa trong ô nhớ DS:1000h vào thanh ghi AH hay lệnh MOV [2000h],AX sẽ đưa nội dung chứa trong AX vào 2 ô nhớ liên tiếp DS:2000h và DS:2001h

4.4. Định địa chỉ truy xuất bộ nhớ gián tiếp

Các cách định địa chỉ trực tiếp sẽ thuận lợi cho các truy xuất bộ nhớ không thường xuyên. Tuy nhiên, nếu một ô nhớ cần phải truy xuất nhiều lần trong một chương trình thì quá trình nhận địa chỉ (2 byte) sẽ phải thực hiện nhiều lần. Điều này sẽ không hiệu quả.

Để giải quyết vấn đề này, ta thực hiện lưu trữ địa chỉ của ô nhớ cần truy xuất trong một thanh ghi con trỏ, chỉ số hay thanh ghi cơ sở (BX, BP, SI hay DI). Ngoài ra, ta có thể sử dụng độ dời bù 2 bằng cách cộng vào các thanh ghi để dời đi so với vị trí được các thanh ghi chỉ đến.

Bảng 2.13:

Cách định địa chỉ	Địa chỉ hiệu dụng (EA – Effective Address)		
	Độ dời	**Thanh ghi nền**	**Thanh ghi chỉ số**
Gián tiếp thanh ghi	Không	BX hay BP	Không
	Không	Không	SI hay DI
Có chỉ số	-128 ÷ 127	Không	SI hay DI
Có nền	-128 ÷ 127	BX hay BP	Không
Có nền và chỉ số	Không	BX hay BP	SI hay DI
Có nền và chỉ số với độ dời	-128 ÷ 127	BX hay BP	SI hay DI

Như vậy, một độ dời có thể được cộng vào thanh ghi nền và kết quả này được cộng tiếp vào thanh ghi chỉ số. Địa chỉ thu được gọi là địa chỉ hiệu dụng EA.

Ngoài ra ta cũng có thể viết cách định địa chỉ gián tiếp như sau:

MOV AX,table[SI]

Trong đó ***table*** là nhãn gán cho một vị trí ô nhớ nào đó. Lệnh này sẽ truy xuất phần tử thứ SI trong dãy ***table*** (giả sử SI = 2 thì sẽ truy xuất phần tử thứ 2). Ta cũng có thể viết lệnh trên như sau:

MOV AX,[table + SI]

Chú ý rằng các đoạn mặc định cho các cách định địa chỉ gián tiếp là đoạn stack khi dùng BP, là đoạn dữ liệu khi dùng BX, SI hay DI.

VD: Lệnh:

```
MOV  AH,10h          thực hiện định địa chỉ tức thời
MOV  AX,[BP + 10]    thực hiện định địa chỉ có nền
MOV  AH,[BP + SI]    thực hiện định địa chỉ có nền và có chỉ số
```

4.5. Định địa chỉ chuỗi

Chuỗi là một dãy liên tục các byte hay word lưu trữ trong bộ nhớ dưới dạng các ký tự ASCII. 8086/8088 có các lệnh dùng để xử lý chuỗi, các lệnh này sử dụng cặp thanh ghi DS:SI để chỉ nguồn chuỗi ký tự và ES:DI để chỉ đích chuỗi. Lệnh MOVSB sẽ chuyển byte dữ liệu nguồn đến vị trí đích trong đó SI và DI sẽ tăng hay giảm tuỳ theo giá trị của DF (xem 2.3.4 và bảng 2.13)

4.6. Thay đổi thanh ghi đoạn mặc định

Như đã nói ở phần trên, khi sử dụng các lệnh định địa chỉ thanh ghi, ta chỉ cần dùng các thanh ghi để xác định độ lệch còn các thanh ghi đoạn thì được hiểu mặc định. Ví dụ như ta dùng lệnh MOV AH,[BP] thì sẽ đưa dữ liệu tại ô nhớ SS:BP vào thanh ghi AH. Trong trường hợp không muốn dùng thanh ghi đoạn mặc định, ta có thể thay đổi bằng cách thêm tên thanh ghi đoạn vào để loại bỏ thanh ghi đoạn mặc định. Ví dụ lệnh MOV AH,CS:[BP] sẽ đưa dữ liệu tại CS:[BP] vào AH.