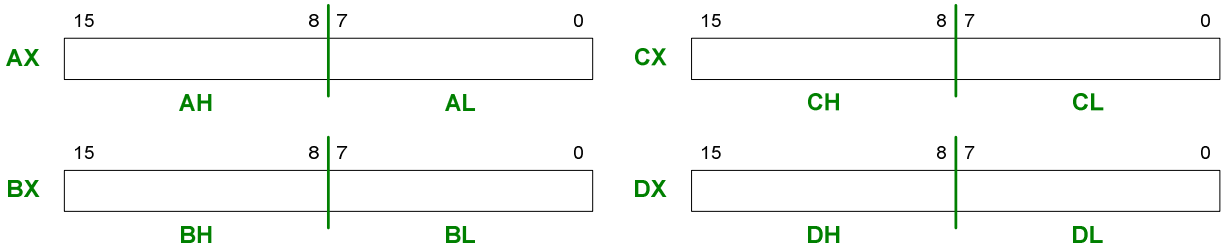
Bộ thanh ghi trong 8086

Các thanh ghi trong bộ vi xử lí 8086 đều là các thanh ghi 16 bit và được chia thành các nhóm như sau:

- Các thanh ghi công dụng chung

AX (accumulator), BX (base), CX (counter), DX (data): có thể được truy xuất độc lập như 2 thanh ghi 8 bit : AH và AL, BH và BL, CH và CL, DH và DL.



- Các thanh ghi con trỏ và chỉ mục (xem chi tiết ở các phần sau)

SP (Stack Pointer), BP (Base Pointer): con trỏ dùng khi làm việc với stack

SI (Source Index), DI (Destination Index): chỉ số mảng khi xử lí mảng (chuỗi)

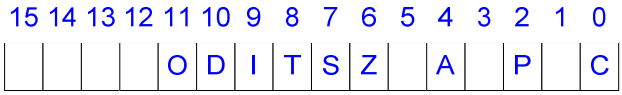
- Các thanh ghi phân đoạn

CS (Code Segment), DS (Data Segment), ES (Extra data Segment), SS (Stack Segment): tương ứng lưu địa chỉ phân đoạn mã lệnh, phân đoạn dữ liệu, phân đoạn dữ liệu bổ sung, phân đoạn ngăn xếp. Địa chỉ phân đoạn này sẽ được kết hợp với địa chỉ offset để truy xuất ô nhớ. (xem chi tiết ở các phần sau)

- Các thanh ghi con trỏ lệnh và trạng thái

IP (Intruction Pointer): thanh ghi chứa địa chỉ offset của lệnh kế tiếp cần thực hiện. Thanh ghi này không thể được truy xuất trực tiếp.

FLAGS: thanh ghi cờ trạng thái, dùng để chứa các bit mô tả trạng thái của lệnh vừa được thực hiện, hoặc chứa các bit điều khiển cần thiết lập trước khi gọi lệnh. Bao gồm các bit cờ sau đây: (xem chi tiết ở các phần sau)



CF (Carry Flag):	bật khi phép tính vừa thực hiện có sử dụng bit nhớ
PF (Parrity Flag):	bật khi kết quả của phép tính vừa thực hiện có chẵn bit 1
AF (Auxilary Flag):	bật khi phép tính vừa thực hiện có sử dụng bit nhớ phụ
ZF (Zero Flag):	bật khi kết quả của phép tính vừa thực hiện là 0
SF (Sign Flag):	bật khi kết quả của phép tính vừa thực hiện có bit dấu bật
TF (Trace Flag):	bật để chuyển sang chế độ chạy từng bước
IF (Interrupt Flag):	bật để cho phép các ngắt xảy ra
DF (Direction Flag):	bật để chọn chế độ giảm chỉ số tự động khi làm việc với mảng
OF (Overflow Flag):	bật khi phép tính vừa thực hiện gây ra tràn số

Tổ chức bộ nhớ trong 8086

Địa chỉ vật lí.

Bus địa chỉ có độ rộng 20 bit, nếu đánh địa chỉ tuần tự tăng dần cho các ô nhớ:

➔ số lượng tối đa các ô nhớ có thể được đánh địa chỉ là 2^20 ô nhớ. Mỗi ô nhớ có kích thước 1 byte

➔ kích thước bộ nhớ tối đa có thể truy cập là 2^20 byte = 1 MB.

Khi đó, địa chỉ của một ô nhớ là một con số 20 bit (hoặc 5 chữ số hex), gọi là địa chỉ vật lí.

Địa chỉ logic

Các thanh ghi trong 8086 đều là 16 bit. Nếu dùng các thanh ghi này để lưu trữ địa chỉ 20bit thì không tiện lợi. Người ta đã tìm cách giảm số bit dùng để đánh địa chỉ xuống còn 16bit.

Ý tưởng được sử dụng là: thông thường, các dữ liệu mà một chương trình cần truy cập nằm gần nhau và tạo thành một khối không lớn lắm; như vậy, nếu ta đánh địa chỉ tương đối (gọi là offset) trong một khối thì số lượng bit dùng để đánh địa chỉ sẽ giảm xuống.

Bộ nhớ được chia thành các khối 64KB, gọi là segment, các khối này không xếp tuần tự cạnh nhau mà xếp gối đầu, với khoảng cách 16byte. Nghĩa là, cứ 16 byte thì lại bắt đầu một segment mới. Như vậy, số lượng segment trong 1MB bộ nhớ là 1MB / 16byte = 65536 = 2^16. Do đó, để đánh địa chỉ segment ta cũng cần 16 bit. Trong phạm vi một segment 64KB (=65536 byte), chỉ cần dùng 16 bit làm địa chỉ offset để xác định một ô nhớ. (Xem Hình 1. Tổ chức bộ nhớ kiểu segment - offset)

Tóm lại, trong cách đánh địa chỉ logic, mỗi ô nhớ có địa chỉ là một cặp (segment:offset), tổng cộng 32bit. Tuy nhiên, như đã nói ở trên, các dữ liệu có liên quan trong bộ nhớ thường ở gần nhau trong một khối segment nên ta có thể *không cần xác định tường minh địa chỉ segment* mà chỉ cần ngầm hiểu. Khi bắt đầu làm việc với một segment nào thì ta sẽ *dùng một thanh ghi để ghi lại địa chỉ segment* đó. Thanh ghi này sẽ được *dùng chung* cho tất cả các phép truy xuất bộ nhớ tiếp theo. Và như thế, trong các phép truy xuất bộ nhớ tiếp theo, chỉ cần dùng thêm 16 bit địa chỉ offset là đủ để xác định vị trí một ô nhớ.

Nếu địa chỉ gồm cả hai phần segment:offset, người ta gọi đó là địa chỉ xa. Nếu địa chỉ chỉ có offset còn segment ngầm định thì người ta gọi là địa chỉ gần, hàm ý là nó xác định một ô nhớ chỉ ở trong phạm vi của segment ngầm định mà thôi.

Chuyển đổi giữa hai kiểu địa chỉ

Việc đổi từ địa chỉ logic sang địa chỉ vật lí rất đơn giản.

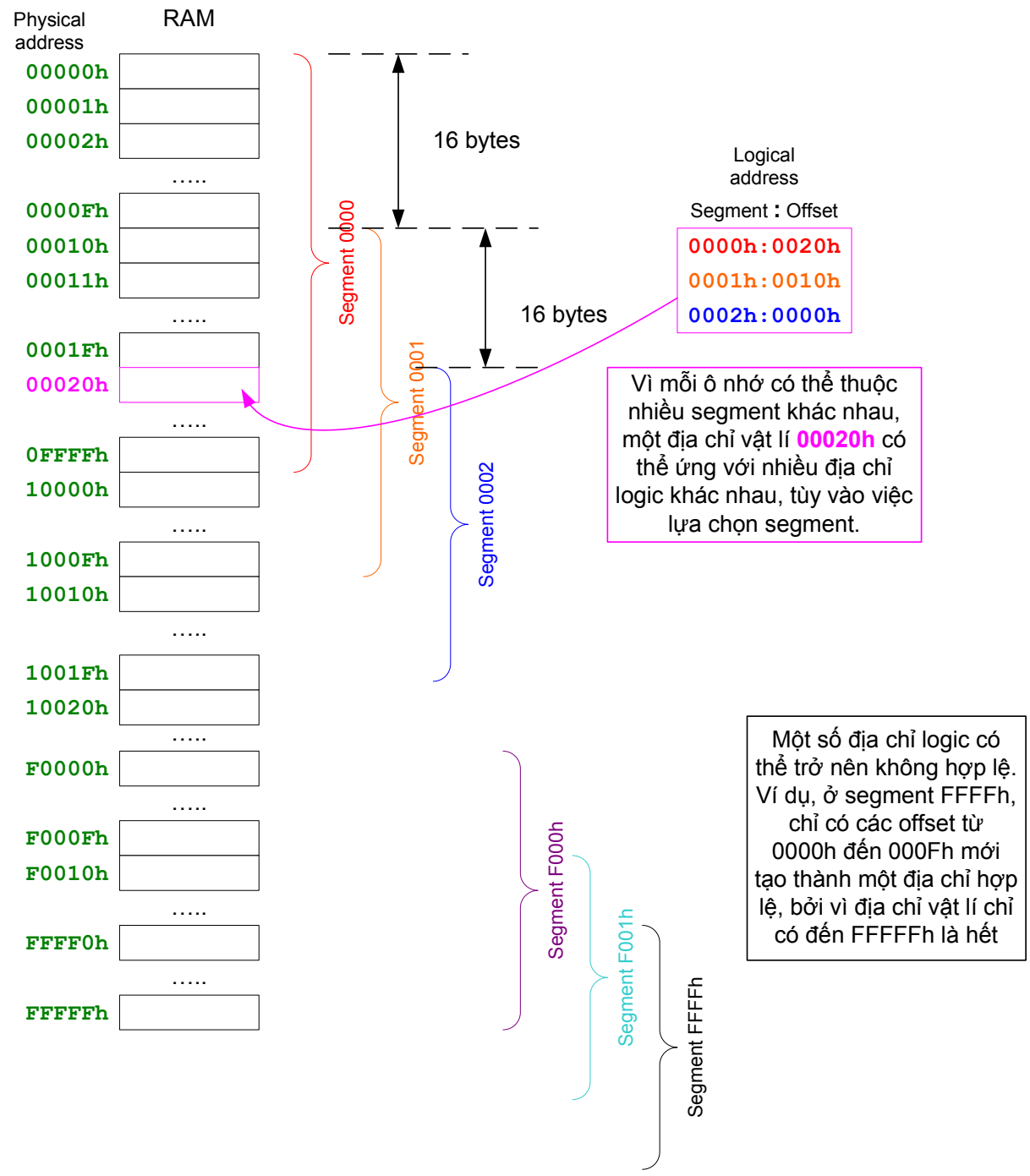
Phy_address = segment * 16 + offset

```
Vd: địa chỉ logic 1234h:0005h sẽ ứng với
địa chỉ vật lí 1234h * 16 + 0005h = 12340h + 0005h = 12345h
```

Việc đổi từ địa chỉ vật lí sang địa chỉ logic cũng không có gì phức tạp. Tuy nhiên, do các segment gối đầu nhau nên mỗi ô nhớ có thể thuộc một vài segment khác nhau. Vì vậy, một địa chỉ vật lí có thể ứng với nhiều địa chỉ logic khác nhau.

```
Vd: địa chỉ vật lí 12345h có thể ứng với
các địa chỉ logic sau:
   1234h:0005h
   1230h:0045h
   1200h:0345h
   1000h:2345h
   1232h:0025h
   …
```

Thông thường khi nói "địa chỉ", ngầm hiểu là địa chỉ logic, mà thường là địa chỉ gần.



Hình 1. Tổ chức bộ nhớ kiểu segment - offset

Một chương trình khi đã được nạp vào bộ nhớ để thực hiện *thông thường* chiếm 3 phân đoạn. Một phân đoạn dành cho mã lệnh (code segment), một phân đoạn dành cho dữ liệu (data segment), và một phân đoạn ngăn xếp (stack segment) dành để lưu các giá trị trung gian hoặc các địa chỉ trở về dùng khi gọi hàm. Địa chỉ của 3 phân đoạn tương ứng được nạp vào 3 thanh ghi CS, DS, SS. Thanh ghi IP chứa địa chỉ offset của lệnh sắp được thực hiện. Như vậy, cặp CS:IP sẽ cho biết địa chỉ logic của lệnh. Thanh ghi SP chứa địa chỉ offset của ô nhớ cuối cùng được lưu vào stack. Các phép truy xuất đến phân đoạn ngăn xếp sẽ dùng cặp SS:SP. Các lệnh truy xuất dữ liệu trong phân đoạn dữ liệu sẽ dùng đến DS để kết hợp với offset được tính toán từ mã lệnh để có được địa chỉ logic của ô nhớ cần truy xuất.