# KỸ THUẬT VI XỬ LÝ

# (MicroProcessor Technology)

## CHƯƠNG 2:
## BỘ VI XỬ LÝ INTEL 8086/8088

☺: Nguyễn Trung Kiên

: Kiennt@neu.edu.vn

# Chương 2: Bộ vi xử lý Intel 8088/8086

2.1. Cấu trúc của bộ vi xử lý Intel 8088

2.2. Các thanh ghi của bộ vi xử lý Intel 8088

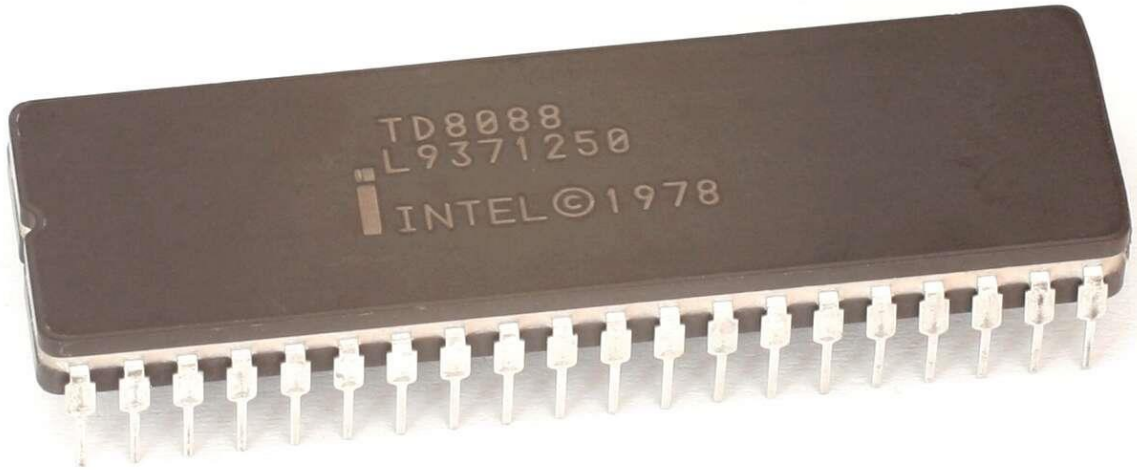
2.3. Nguyên tắc làm việc của bộ vi xử lý Intel 8088

2.4. Các chế độ địa chỉ của bộ vi xử lý 8088

2.5. Các bộ vi xử lý thế hệ sau 8088 của Intel

# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

- 8088:
  - 1/6/1979, Intel đã giới thiệu bộ vi xử lý 4,788 MHz 8088, có các thanh ghi 16 bit.
  - 8088 được thiết kế tại phòng thí nghiệm của Intel ở Haifa, Israel.



# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

- Bus nội (Internal bus) là đường dẫn để truyền dữ liệu giữa các thanh ghi và ALU trong VXL
- Bus ngoại (External bus) dùng cho bên ngoài nối đến RAM, ROM và I/O
- Độ rộng của bus nội và ngoại có thể khác nhau. Ví dụ
  - 8088: bus nội là 16 bit, bus ngoại là 8 bit
  - 8086: bus nội là 16 bit, bus ngoại là 16 bit

# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

- 8088 và 8086 có cấu tạo tương tự nhau, khác nhau cơ bản:
  - 8088: Bus dữ liệu ngoài là 8 bit
  - 8086: Bus dữ liệu ngoài là 16 bit
- Hệ thống máy tính dùng 8088 chậm hơn 8086 nhưng có giá thành rẻ hơn (do dùng bus dữ liệu ngoài 8 bit nên giảm được khá nhiều chip ghép nối và bổ trợ).
- Hãng IBM đã sử dụng 8088 để thiết kế máy IBM-PC (1981).

*Máy tính cá nhân IBM*

# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

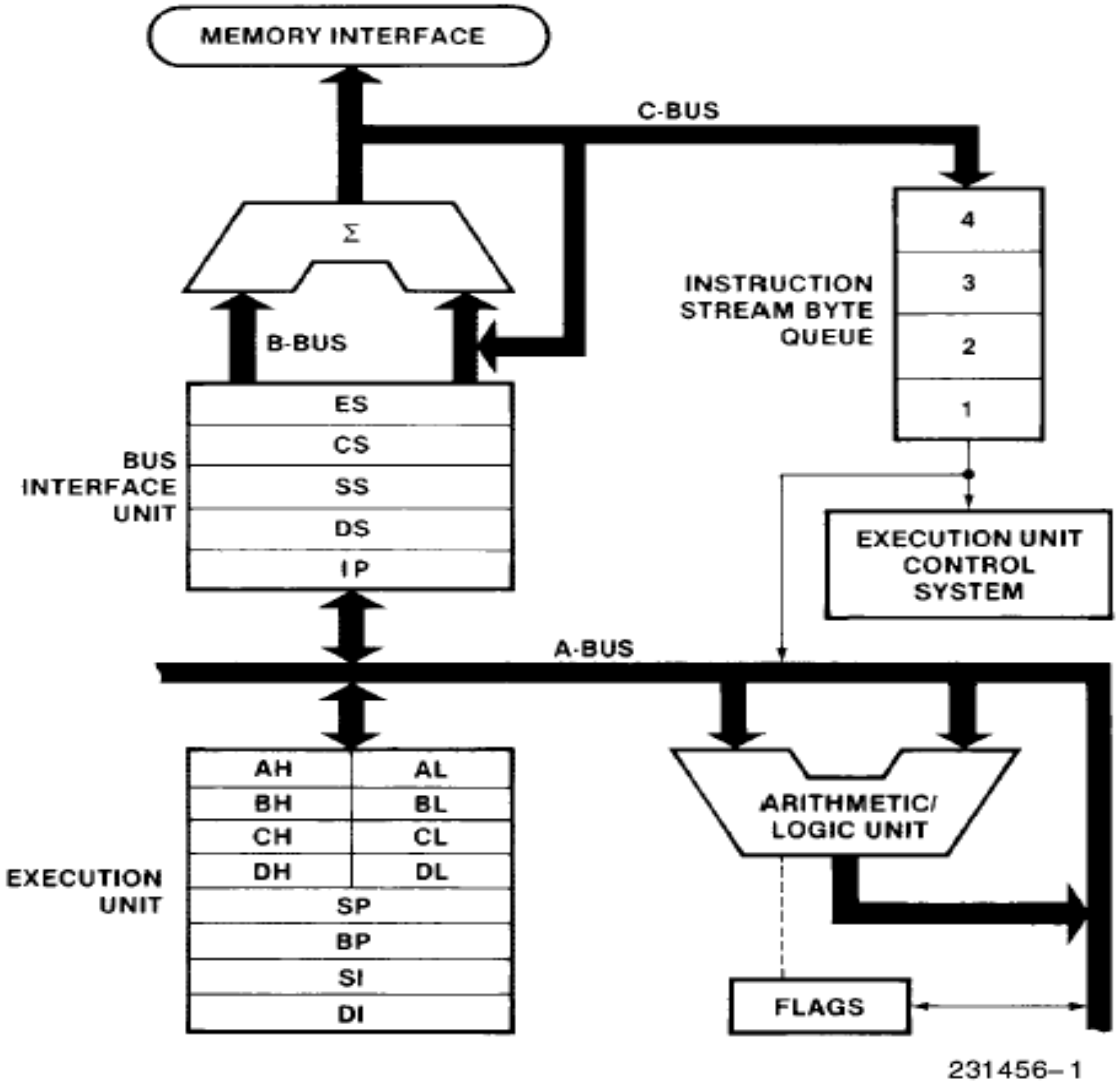



Figure 1. 8088 CPU Functional Block Diagram

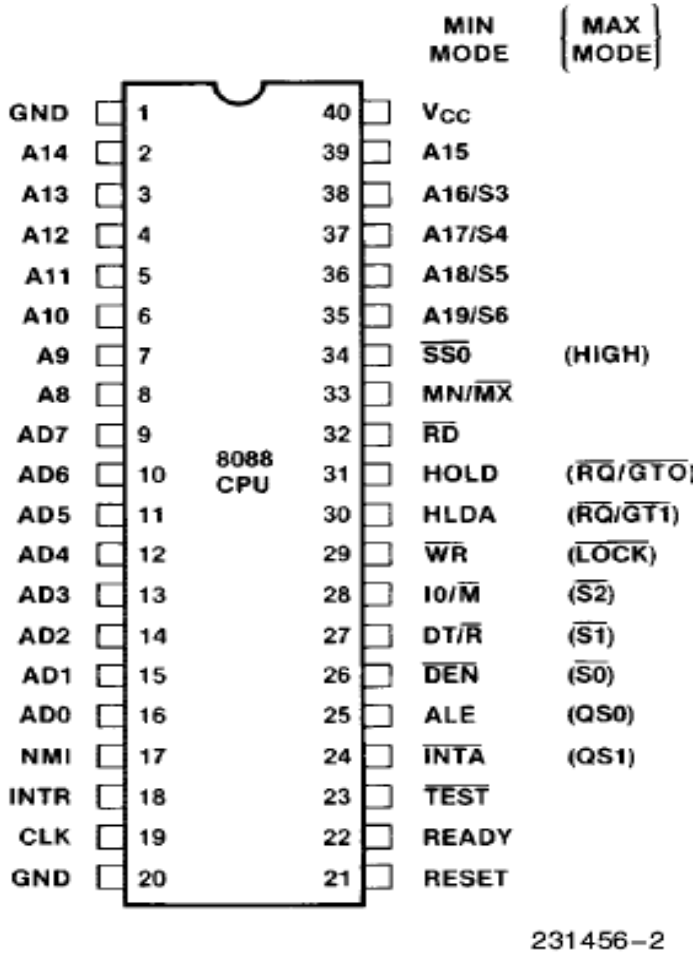
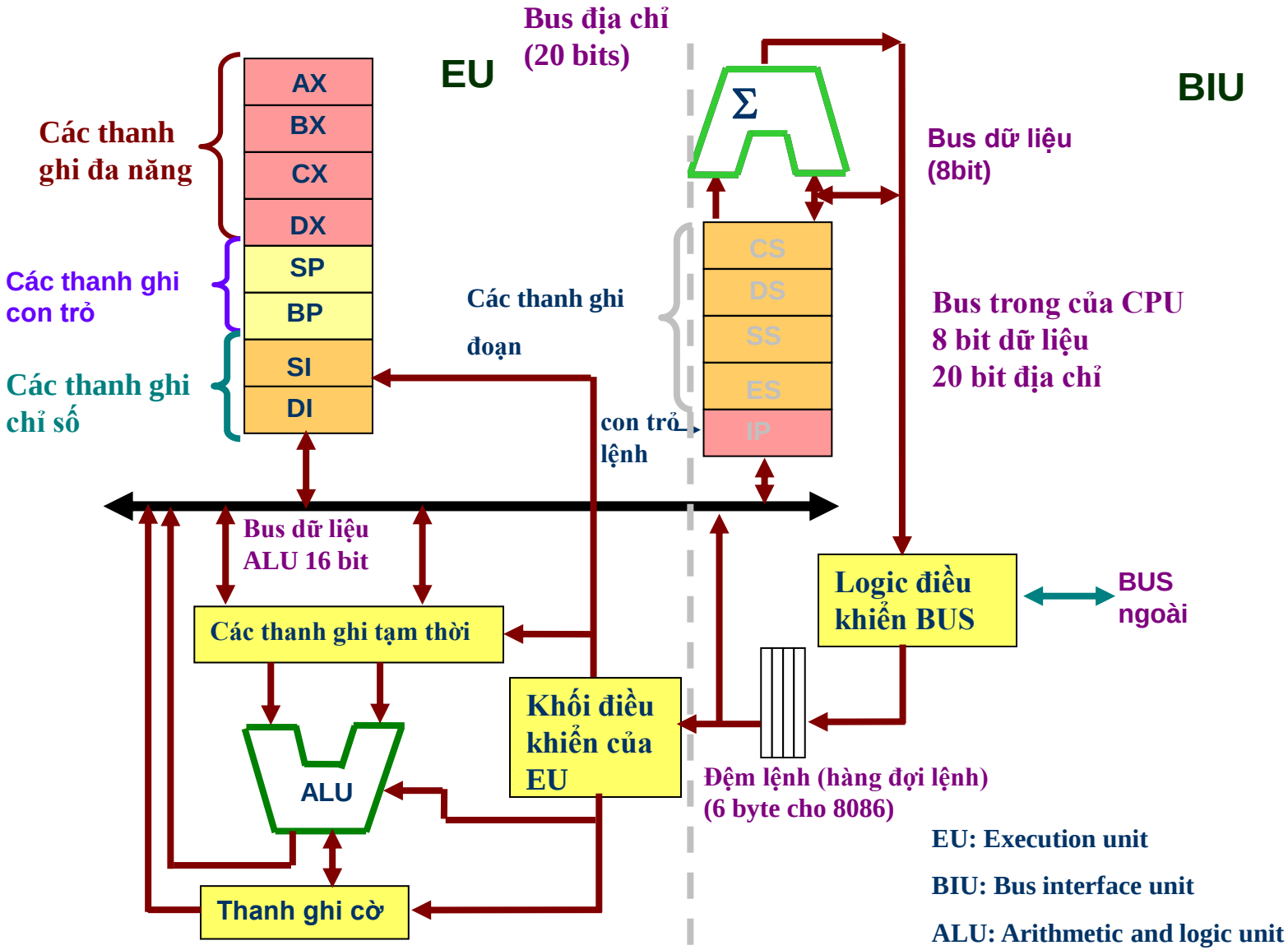



Figure 2. 8088 Pin Configuration

# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

*Sơ đồ khối của bộ vi xử lý 8088*



# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

- **Cấu trúc bên trong của bộ vi xử lý intel 8088 gồm:**
  - Đơn vị giao tiếp bus BIU (*Bus Interface Unit*).
  - Đơn vị định địa chỉ AU (*Addressing Unit*).
  - Đơn vị lệnh IU (*Instrustion Unit*).
  - Đơn vị thực hiện lệnh EU (*Execution Unit*)
    - Bộ tính số học logic ALU (*Arithmetical Logical Unit*)
    - Bộ điều khiển CU (*Control Unit*)
  - Các thanh ghi (*Registers*)

# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

- BIU đơn vị phối ghép bus (***Bus Interface Unit***):
  - Nhận các mã lệnh từ bộ nhớ và đặt chúng vào hàng chờ lệnh
  - Thực hiện các công việc điều khiển hệ thống BUS.
- EU đơn vị thực thi lệnh (Execution Unit):
  - Chịu trách nhiệm giải mã lệnh, thực thi lệnh và phát xung điều khiển các thành phần khác trong hệ thống.

# 2.1. Cấu trúc của bộ vi xử lý Intel 8086/8088

## BIU

- Bộ điều khiển logic bus (Control System)
- Hàng đợi lệnh (Instruction Stream Byte Queue – Prefetch Queue)
- Các thanh ghi đoạn và thanh ghi con trỏ: 16 bit

## EU

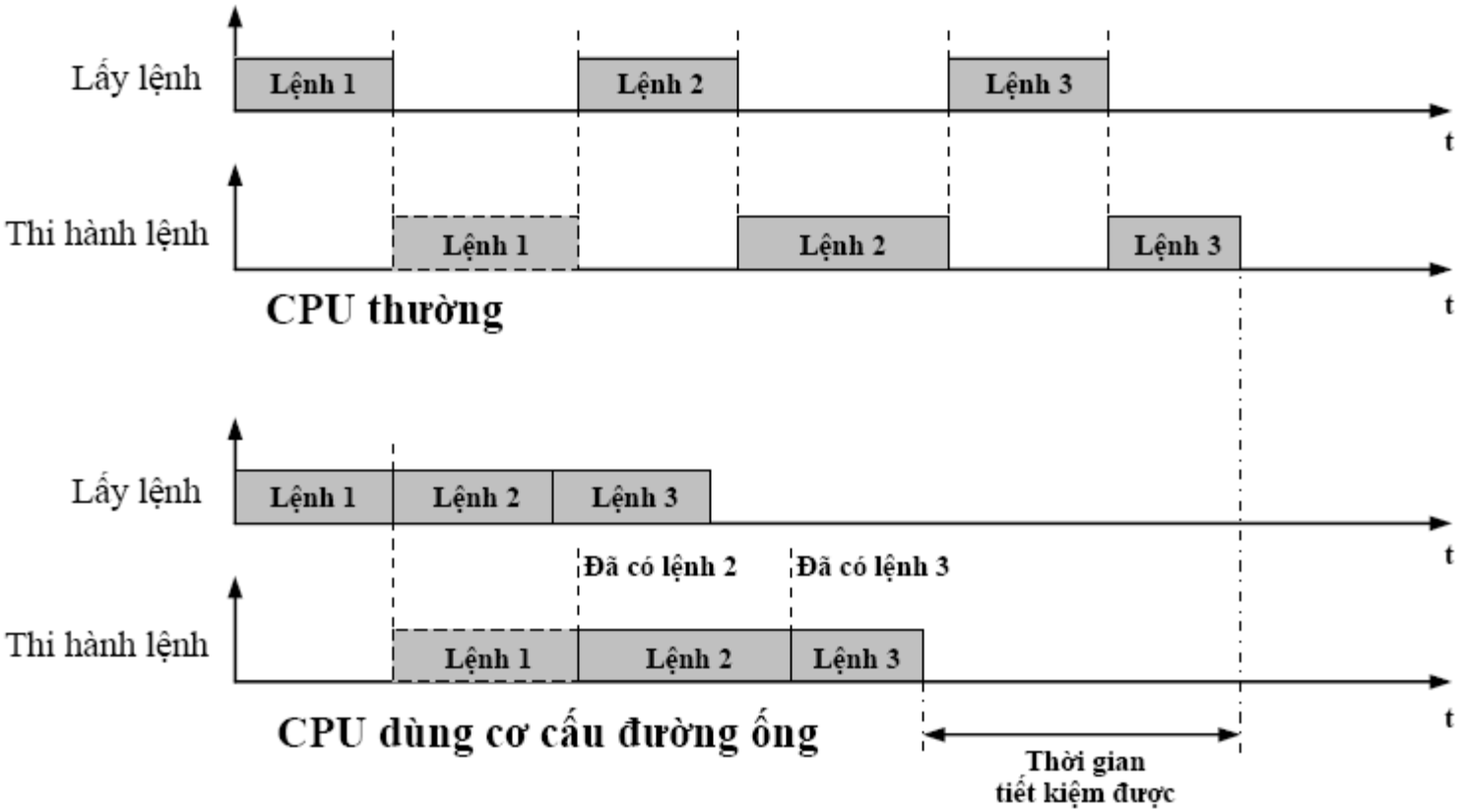
- CU (Control unit): Khối điều khiển
- ALU (Arithmetic Logic Unit): Bộ xử lí số học và logic
- Các thanh ghi đa năng và thanh ghi cờ

# BIU và EU

- Hoạt động của hai khối BIU và EU diễn ra độc lập với nhau nên quá trình lấy lệnh và thi hành lệnh được vi xử lý thực hiện đồng thời theo cơ cấu đường ống (pipeline).
- Điều này tuy không làm tăng tốc độ xử lý của CPU (giới hạn bởi tần số xung đồng bộ) nhưng làm giảm bớt thời gian thi hành của cả chương trình.

# BIU và EU

- Hình minh họa về sự phân phối thời gian cho hai quá trình lấy lệnh và thi hành lệnh của CPU bình thường và của CPU dùng cơ cấu đường ống



# Chương 2: Bộ vi xử lý Intel 8088/8086

2.1. Cấu trúc của bộ vi xử lý Intel 8088

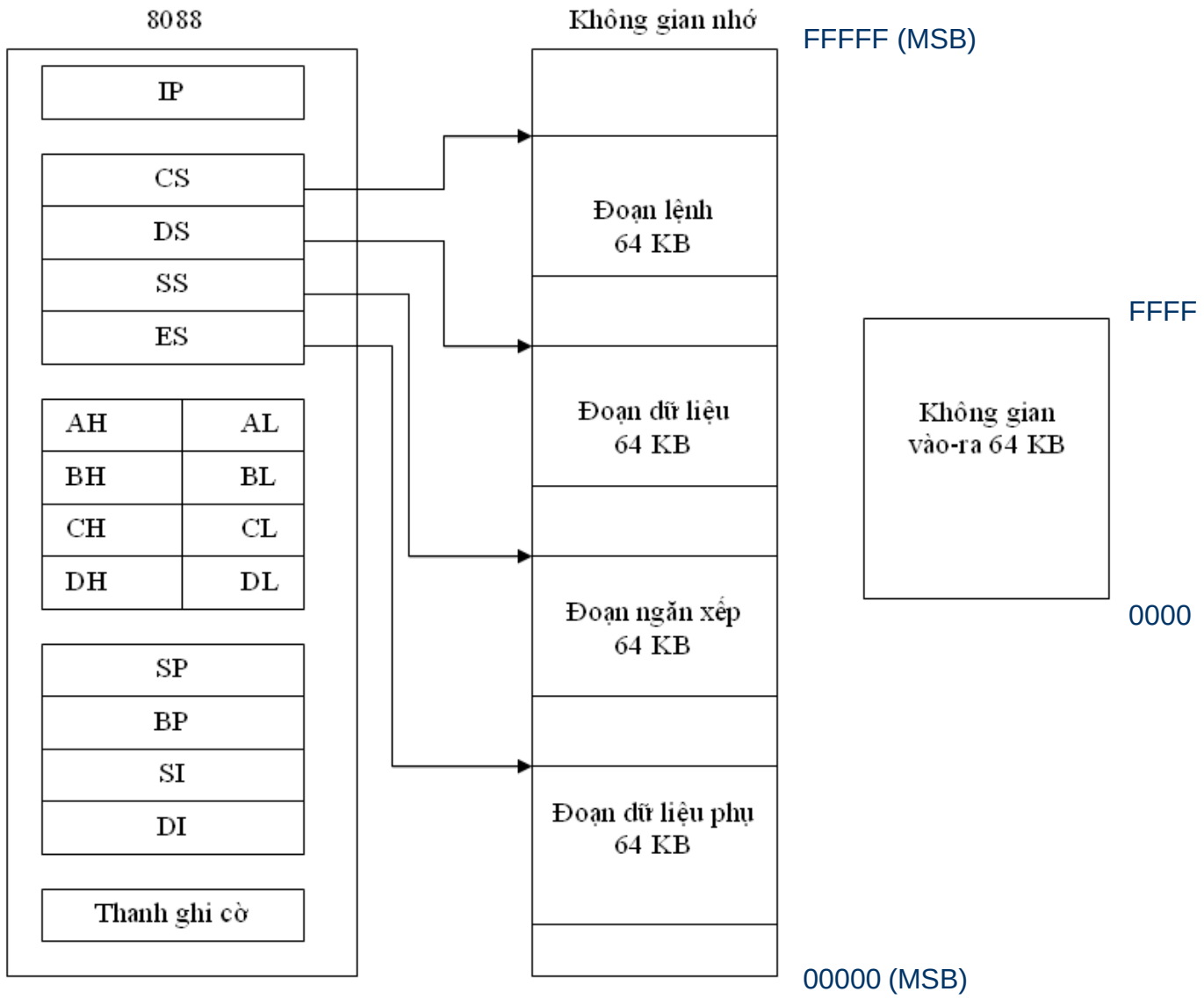
2.2. Các thanh ghi của bộ vi xử lý Intel 8088

2.3. Nguyên tắc làm việc của bộ vi xử lý Intel 8088

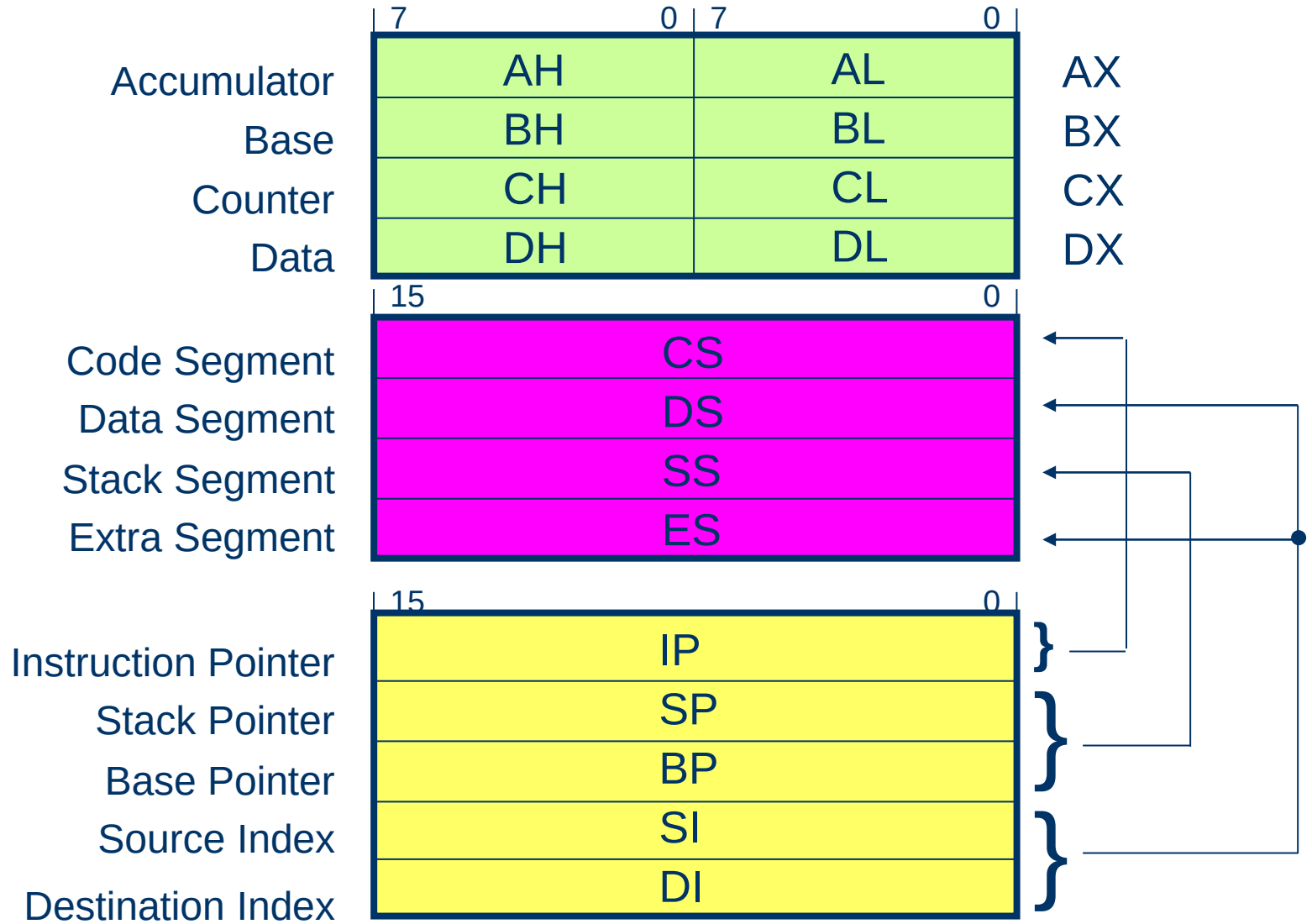
2.4. Các chế độ địa chỉ của bộ vi xử lý 8088

2.5. Các bộ vi xử lý thế hệ sau 8088 của Intel

# 2.2. Các thanh ghi của bộ VXL Intel 8086/8088



# Cấu trúc thanh ghi 8086/8088



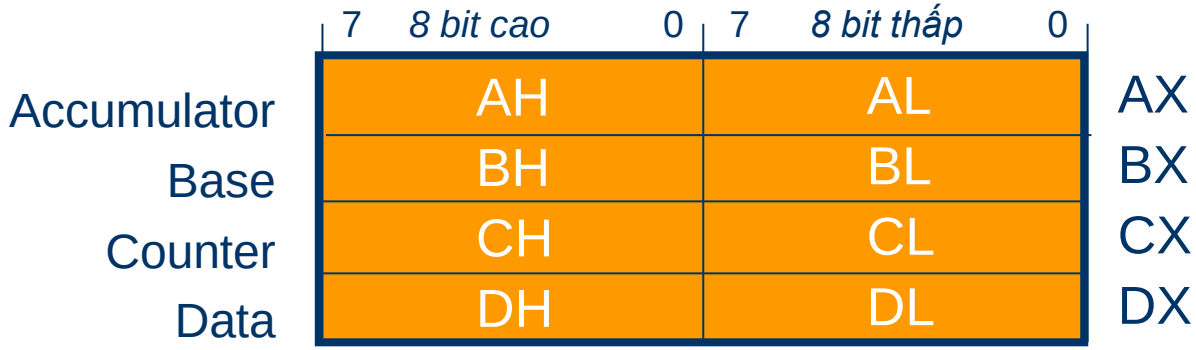
# Các thanh ghi của bộ VXL Intel 8086/8088

- 4 thanh ghi đoạn:
  - **CS** (Code Segment): thanh ghi đoạn lệnh
  - **DS** (Data Segment): thanh ghi đoạn dữ liệu
  - **SS** (Stack Segment): thanh ghi đoạn ngăn xếp
  - **ES** (Extra Segment): thanh ghi đoạn dữ liệu phụ
- 3 thanh ghi con trỏ:
  - **IP** (Instruction Pointer): thanh ghi con trỏ lệnh
  - **SP** (Stack Pointer): con trỏ ngăn xếp
  - **BP** (Base Pointer): thanh ghi con trỏ cơ sở
- 4 thanh ghi dữ liệu: Mỗi thanh ghi này đều có thể được chia ra thành 2 nửa có khả năng sử dụng độc lập.
  - **AX** (Accumulator): thanh chứa - thanh ghi tích lũy
  - **BX** (Base): thanh ghi cơ sở
  - **CX** (Count): thanh ghi đếm
  - **DX** (Data): thanh ghi dữ liệu
- Thanh ghi cờ
- 2 thanh ghi chỉ số SI, DI

# Các thanh ghi đoạn (16 bit)

- Thanh ghi đoạn mã CS (*Code Segment*)
  - Chứa mã lệnh chương trình
- Thanh ghi đoạn dữ liệu DS (*Data Segment*)
  - Lưu trữ dữ liệu cho chương trình
- Thanh ghi đoạn ngăn xếp SS (*Stack Segment*)
  - Lưu các địa chỉ sẽ trả về từ các chương trình con (subroutine) hay trình phục vụ ngắt (interrupt subroutine)
- Thanh ghi đoạn dữ liệu phụ ES (*Extra Segment*)
  - Dùng cho các dữ liệu dùng chung.
  - Dùng trong trường hợp các dữ liệu cần khai báo vượt quá kích thước cho phép của 1 đoạn (các khai báo mảng, file...)

# Các thanh ghi đa năng (16 bit)

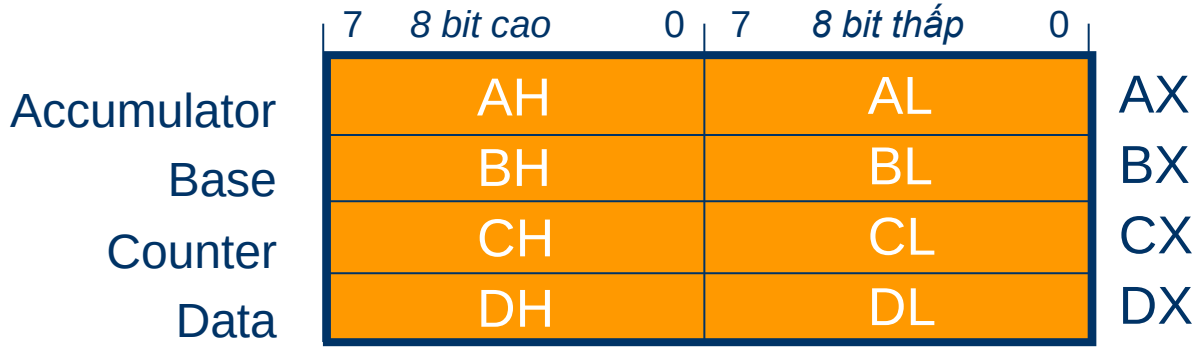



- **8088/8086 đến 80286 : 16 bits**
- **80386 trở lên: 32 bits EAX,EBX, ECX, EDX**

- Có thể truy nhập như các thanh ghi 8-bit
- Lưu trữ tạm thời dữ liệu để truy cập nhanh hơn và tránh khỏi phải truy cập bộ nhớ

# Các thanh ghi đa năng (16 bit)

- AX, BX, CX, DX là các thanh ghi 16 bit
- AH, AL, BH, BL, CH, CL, DH, DL là các thanh ghi 8 bit
- Chức năng chung: chứa dữ liệu tạm thời
- Chức năng riêng:
  - AX:
    - Dùng cho lệnh nhân chia theo word
    - Dùng cho vào ra theo word
  - AL:
    - Dùng cho lệnh nhân chia theo byte.
    - Dùng cho vào ra theo byte
    - Dùng cho các lệnh số học với số BCD
  - AH: Dùng cho các lệnh nhân chia theo byte
  - BX: Dùng để chứa địa chỉ cơ sở
  - CX: Dùng để chứa số lần lặp của lệnh LOOP và các lệnh xử lý xâu ký tự
  - CL: Dùng để chứa số lần dịch của lệnh dịch, lệnh quay
  - DX:
    - Dùng cho lệnh nhân chia theo word
    - Dùng chứa địa chỉ cổng vào ra




- **8088/8086 đến 80286 : 16 bits**
- **80386 trở lên: 32 bits**
  **EAX,EBX, ECX, EDX**

# Các thanh ghi con trỏ và chỉ số

- Chứa địa chỉ lệch (offset), gồm:
  - **IP** (Instruction Pointer): Con trỏ lệnh → trỏ vào lệnh tiếp theo sẽ được thực hiện trong đoạn mã lệnh CS.
    - Địa chỉ đầy đủ của lệnh tiếp theo là CS:IP
  - **BP** (Base Pointer): Con trỏ cơ sở → chứa địa chỉ của dữ liệu trong đoạn ngăn xếp SS hoặc các đoạn khác
    - Địa chỉ đầy đủ của một phần tử trong đoạn ngăn xếp là SS:BP
  - **SP** (Stack Pointer): Con trỏ ngăn xếp → chứa địa chỉ hiện thời của đỉnh ngăn xếp
    - SS:SP
  - **SI** (Source Index): Chỉ số nguồn → chứa địa chỉ dữ liệu nguồn trong đoạn dữ liệu DS trong các lệnh thao tác với chuỗi
    - DS:SI
  - **DI** (Destination Index): Chỉ số đích → chứa địa chỉ dữ liệu đích trong đoạn dữ liệu DS trong các lệnh thao tác với chuỗi
    - DS:DI
    - SI và DI có thể được sử dụng như thanh ghi đa năng

# Stack và các thanh ghi SP, BP

- Stack (ngăn xếp): vùng nhớ tổ chức theo cơ chế LIFO, dùng để cất giữ thông tin và có thể khôi phục lại.
- Đoạn Stack được quản lý nhờ thanh ghi SS.
- Thông tin được trao đổi với Stack theo word (16 bit).
- SP chứa địa chỉ offset của ngăn nhớ đỉnh Stack
  - Nếu cất thêm một thông tin vào Stack thì nội dung của SP giảm đi 2
  - Nếu lấy ra một thông tin của Stack thì nội dung của SP tăng lên 2
  - Nếu Stack rỗng thì SP trở vào đáy Stack
  - **SS:SP** chứa địa chỉ logic của ngăn nhớ đỉnh Stack
- BP là thanh ghi chứa địa chỉ offset của một ngăn nhớ nào đó trong Stack → địa chỉ logic của ngăn nhớ đó là SS:BP

# Các thanh ghi con trỏ và chỉ số

- Thanh ghi đoạn và thanh ghi lệch ngầm định:

| Segment | Offset | Chú thích |
|---|---|---|
| CS | IP | Địa chỉ lệnh |
| SS | SP hoặc BP | Địa chỉ ngăn xếp |
| DS | BX, DI, SI, số 8 bit hoặc số 16 bit | Địa chỉ dữ liệu |
| ES | DI | Địa chỉ chuỗi đích |

# Thanh ghi cờ FR (Flag Register)

- Không phải tất cả các bit đều được sử dụng
- Mỗi bit được sử dụng gọi là một cờ
- Các cờ đều có tên và có thể được Lập/Xoá riêng lẻ
- Bao gồm:
  - 6 cờ trạng thái: biểu thị trạng thái của kết quả phép toán.
  - 3 cờ điều khiển: đặt chế độ làm việc cho bộ vi xử lý.

# Thanh ghi cờ FR (Flag Register)

## 6 bit cờ trạng thái:

- Cờ ZF (Zero - cờ không/cờ rỗng):
  - ZF= 1 nếu kết quả phép toán bằng 0
  - ZF=0 nếu kết quả phép toán khác 0.
- Cờ SF (Sign - cờ dấu):
  - SF=1 nếu kết quả phép toán nhỏ hơn 0
  - SF=0 nếu kết quả phép toán lớn hơn hoặc bằng 0.
- Cờ CF (Carry - cờ nhớ):
  - Nếu phép cộng có nhớ ra khỏi bit cao nhất hay phép toán trừ có mượn ra khỏi bit cao nhất thì CF được thiết lập (báo tràn với số nguyên không dấu).
- Cờ OF (Overflow - cờ tràn):
  - Nếu cộng 2 số cùng dấu mà kết quả có dấu ngược lại thì OF được thiết lập (báo tràn với số nguyên có dấu).
- Cờ PF (Parity - cờ kiểm tra chẵn lẻ):
  - Nếu tổng số bit 1 của kết quả là chẵn thì cờ PF được thiết lập.
- Cờ AF (Auxiliary - cờ nhớ phụ):
  - Nếu phép cộng có nhớ từ bit 3 sang bit 4 hoặc phép trừ có mượn từ bit 3 sang bit 4 thì cờ AF được thiết lập.

# Thanh ghi cờ FR (Flag Register)

- **3 bit cờ điều khiển: (được lập hoặc xoá bằng các lệnh riêng)**
  - Cờ TF (Trap - cờ bẫy):
    - Nếu TF = 1 thì bộ vi xử lý hoạt động theo chế độ thực hiện từng lệnh (chế độ gỡ rối chương trình).
  - Cờ IF (Interrupt - cờ ngắt):
    - Nếu IF = 1 thì bộ vi xử lý cho phép ngắt với yêu cầu ngắt đưa đến chân tín hiệu INTR (Interrupt Request) của bộ vi xử lý.
    - Nếu IF = 0 thì cấm ngắt.
  - Cờ DF (Director - cờ hướng): chỉ hướng xử lý xâu ký tự.
    - Nếu DF = 0, xử lý từ trái sang phải.
    - Nếu DF = 1, xử lý từ phải sang trái.
- 8086 cho phép User lập trình bật tắt các cờ CF,DF,IF,TF

# Thanh ghi cờ FR (Flag Register)

- Ví dụ:

$$\begin{array}{r} 80h \\ +\ 80h \\ \hline 100h \end{array}$$

  - Hãy xác định các cờ sau khi thực hiện phép toán trên?

# Thanh ghi cờ FR (Flag Register)

- Ví dụ:

$$\begin{array}{r} 80h \\ +\ 80h \\ \hline 100h \end{array}$$

  - SF=0 vì MSB trong kết quả=0
  - PF=1 vì có tổng số bit 1 là không
  - ZF=1 vì kết quả thu được là 0
  - CF=1 vì có nhớ từ bit MSB trong phép cộng
  - OF=1 vì có tràn trong phép cộng 2 số âm

# Thanh ghi cờ FR (Flag Register)

- VD: Signed Overflow
  - **10010110+ 10100011=00111001**
    - *Carry in = 0, Carry out = 1*
    - *Neg+Neg=Pos*
    - *Signed overflow occurred*
    - *OF = 1 (set)*
  - **00110110+ 01100011=10011001**
    - *Carry in = 1, Carry out = 0*
    - *Pos+Pos=Neg*
    - *Signed overflow occurred*
    - *OF = 1 (set)*

# Thanh ghi cờ FR (Flag Register)

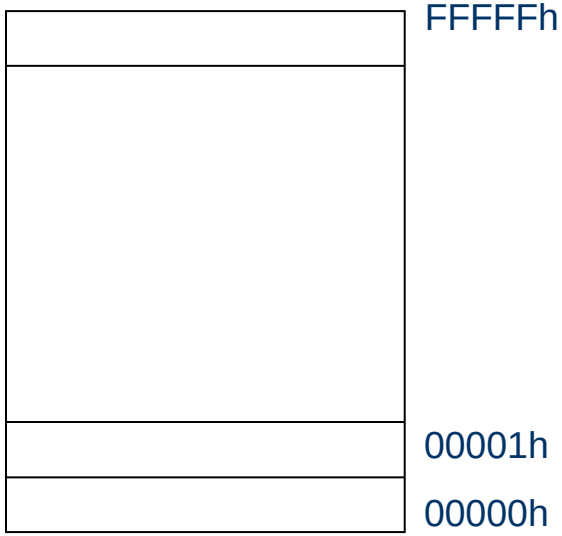
- VD: No Signed Overflow
  - **10010110+ 01100011 = 11111001**
    - *Carry in = 0, Carry out = 0*
    - *Neg+Pos=Neg*
    - *No Signed overflow occurred*
    - *OF = 0 (clear)*
  - **10010110+ 11110011=10001001**
    - *Carry in = 1, Carry out = 1*
    - *Neg+Neg=Neg*
    - *No Signed overflow occurred*
    - *OF = 0 (clear)*

# Thanh ghi cờ FR (Flag Register)

- VD: Unsigned Overflow
  - The carry flag is used to indicate if an unsigned operation overflowed
  - The processor only adds or subtracts – it does not care if the data is signed or unsigned!
  - **10010110 + 11110011 = 10001001**
    - *Carry out = 1*
    - *Unsigned overflow occurred*
    - *CF = 1 (set)*

# Không gian nhớ

- 8088 có bus địa chỉ 20 bit → Không gian địa chỉ bộ nhớ = $2^{20}$ byte = 1MB.
- 8088 có khả năng truy nhập bộ nhớ theo:
  - Từng byte
  - Từng word: truy nhập theo 2 byte có địa chỉ liên tiếp
- 8088 lưu trữ thông tin trong bộ nhớ chính theo kiểu đầu nhỏ (Little-endian)



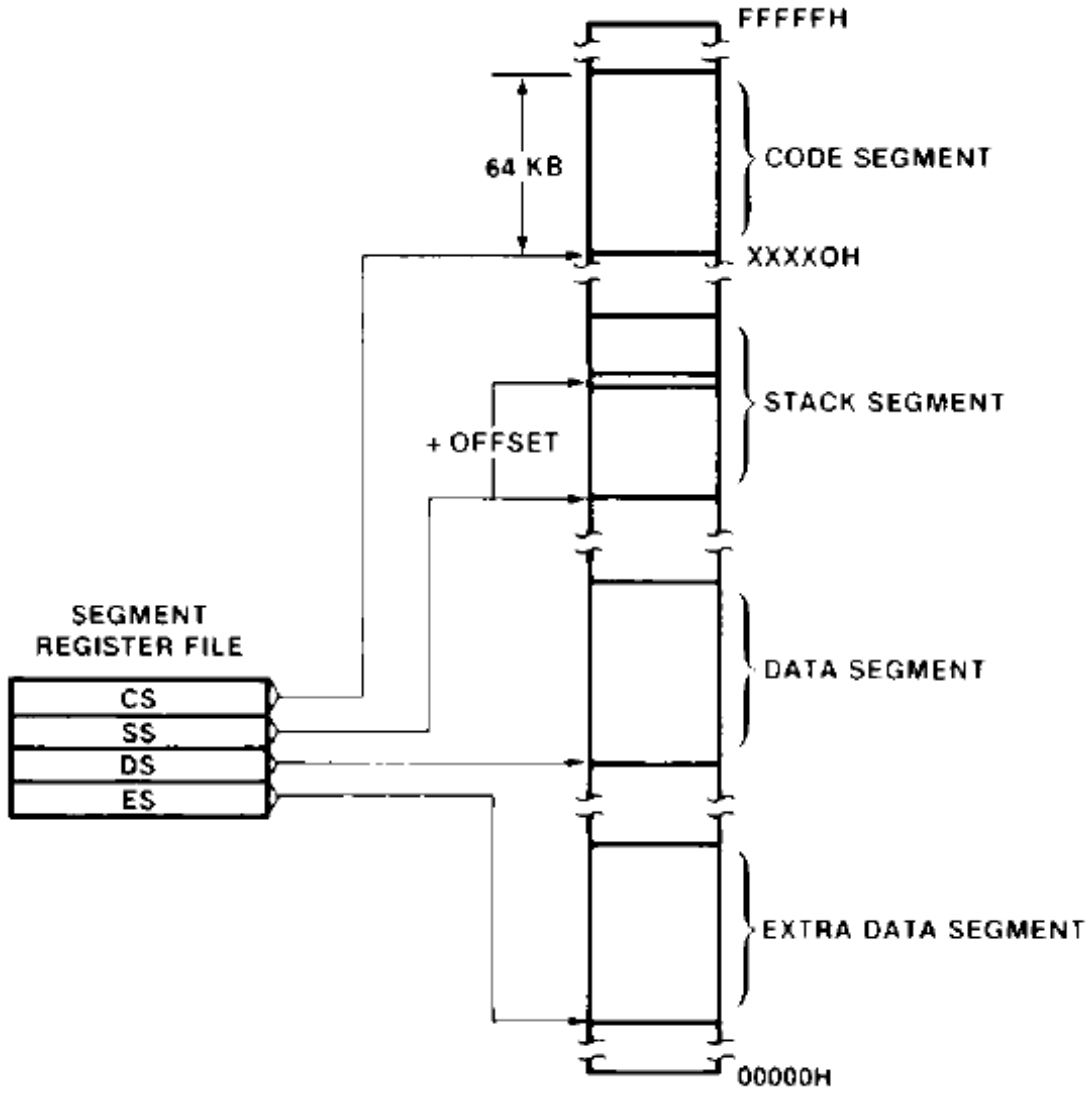
# Không gian vào-ra

- 8088 có khả năng quản lý không gian vào-ra 64 KB = $2^{16}$ Byte,
  - → sẽ phải phát ra 16 bit địa chỉ để tìm cổng vào-ra tương ứng trên các chân địa chỉ từ $A_0$ đến $A_{15}$.
- Trong trường hợp phát ra 8 bit địa chỉ từ $A_0$ đến $A_7$ để xác định một cổng vào-ra thì sẽ quản lý được $2^8$=256 cổng vào-ra.

# Các thanh ghi đoạn và phân đoạn BN

- 8088 có 4 thanh ghi đoạn 16 bit, do đó tại một thời điểm, 8088 chỉ làm việc được với 4 đoạn nhớ:
  - CS (Code Segment): quản lý đoạn lệnh
  - SS (Stack Segment): quản lý đoạn ngăn xếp
  - DS (Data Segment): quản lý đoạn dữ liệu
  - ES (Extra Data Segment): quản lý đoạn dữ liệu phụ
- 8088 phát ra một địa chỉ của ngăn nhớ = 20 bit, do đó không gian nhớ của nó là 1 MB (=$2^{20}$ Byte)
- Các thanh ghi bên trong 8088 đều có độ dài là 16 bit.

# Các thanh ghi đoạn



# Sự hình thành địa chỉ

- 8088 sử dụng 20 bit để đánh địa chỉ bộ nhớ ➔ quản lý $2^{20}$=1MB bộ nhớ.
- 8088 không có thanh ghi nào 20 bit, tất cả là 16 bit → 1 thanh ghi chỉ có thể đánh địa chỉ tối đa là $2^{16}$=64 KB bộ nhớ.
- → phải kết hợp 2 thanh ghi mới địa chỉ hoá toàn bộ bộ nhớ.
- Intel đề xuất 1 phương pháp để hình thành địa chỉ. 8088 sử dụng:
  - 1 thanh ghi segment: lưu địa chỉ cơ sở
  - 1 thanh ghi chỉ số hay thanh ghi con trỏ: lưu địa chỉ lệch (offset)
- Mỗi địa chỉ ô nhớ được hình thành từ 1 phép tính tổng 1 địa chỉ cơ sở và 1 địa chỉ offset.
- Phép cộng này sẽ tạo 1 địa chỉ 20 bit gọi là địa chỉ vật lý.

# Địa chỉ vật lý & địa chỉ logic

- Địa chỉ vật lý : dùng để thiết kế mạch
  - Số 20 bit $A_{19}A_{18}A_{17}\ldots A_1A_0$ (00000 ÷ FFFFF)
- Địa chỉ logic : dùng cho lập trình

  2 thành phần: **segment:offset**
  - Dùng thanh ghi đoạn chứa **segment**
  - Dùng thanh ghi đa năng chứa **offset**

# Phân đoạn bộ nhớ

- Intel chia không gian nhớ của 8088 thành các đoạn nhớ (segment) có dung lượng 64 KB $=2^{16}$ Byte
- Địa chỉ đầu của mỗi đoạn nhớ chia hết cho 16, do đó địa chỉ đầu của một đoạn nào đó sẽ có dạng: xxxx0h (x là chữ số Hexa bất kỳ).
- Để quản lý địa chỉ đầu của một đoạn nhớ chỉ cần lưu trữ 4 số Hexa (16 bit cao), đây gọi là địa chỉ đoạn.
  - VD: nếu địa chỉ đoạn là 1234h thì địa chỉ vật lý của đầu đoạn nhớ đó là 12340h

# Sự phân đoạn bộ nhớ

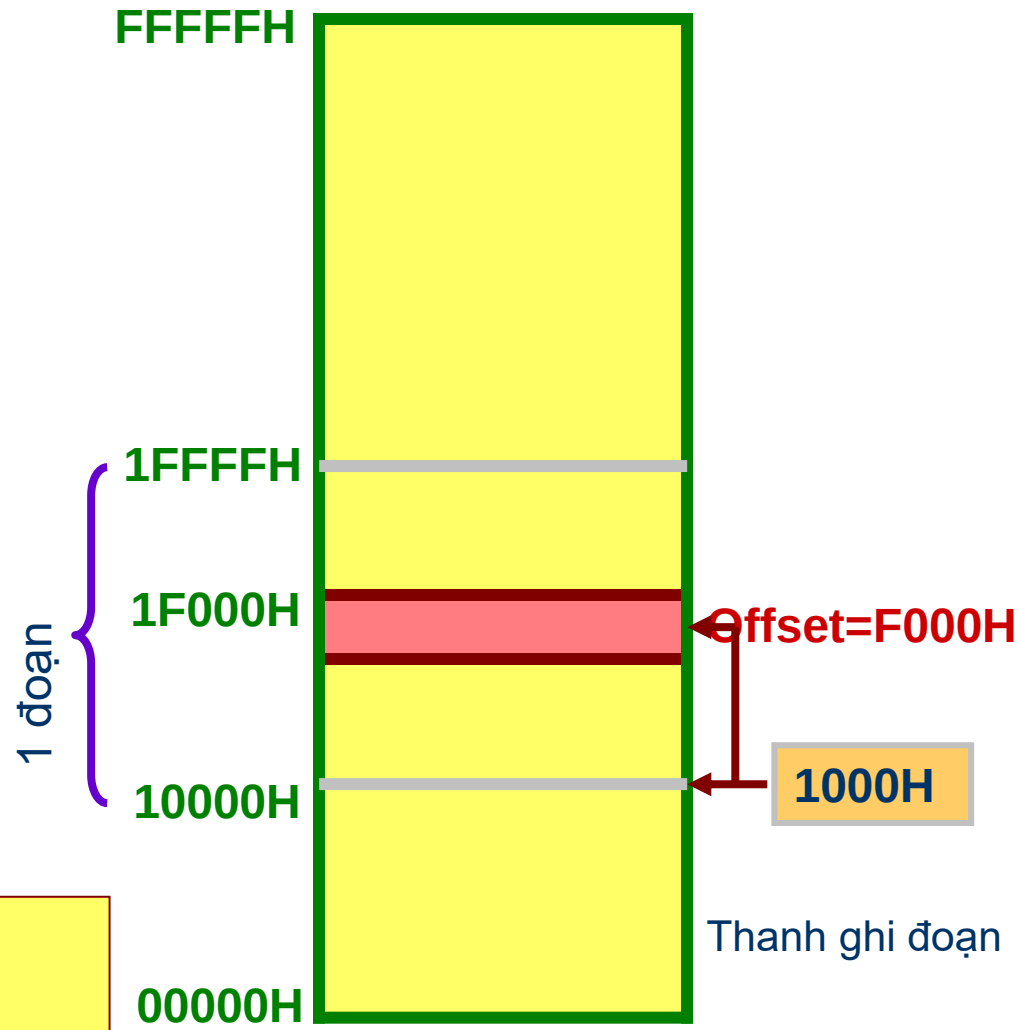
- CPU 8088/8086 dùng phương pháp phân đoạn bộ nhớ để quản lý bộ nhớ 1MB của nó.
- Địa chỉ 20 bit của bộ nhớ 1MB không thể chứa đủ trong các thanh ghi 16 bit của 8088/8086

  → bộ nhớ 1MB được chia ra thành các **đoạn (segment)** 64KB.
- Địa chỉ trong các đoạn 64KB chỉ có 16 bit nên 8088/8086 dễ dàng xử lý bằng các thanh ghi của nó.

- **➔ Phân đoạn bộ nhớ: là cách dùng các thanh ghi 16 bit để biểu diễn cho địa chỉ 20 bit.**

# Địa chỉ logic

- Tổ chức của bộ nhớ 1MB
  - Đoạn bộ nhớ (segment)
    - $2^{16}$ bytes =64 KB
    - Đoạn 1: địa chỉ đầu 00000 H
    - Đoạn 2: địa chỉ đầu 00010 H
    - Đoạn cuối cùng: FFFF0 H
  - Ô nhớ trong đoạn:
    - Địa chỉ lệch: offset
    - Ô 1: offset: 0000
    - Ô cuối cùng: offset: FFFF
  - Địa chỉ logic:
    Segment : offset




**Địa chỉ vật lý=Segment*16 + offset**

**Chế độ thực (real mode)**

# Sự hình thành địa chỉ tuyệt đối


CPU tự động lấy địa chỉ segment x 10 (hệ 16) thành 08F10

Sau đó nó cộng với địa chỉ Offset 0100

➔ địa chỉ tuyệt đối : 09010

# Cách tính địa chỉ vật lý từ địa chỉ logic

**Địa chỉ vật lý = (segment*16) + offset**

| | | |
|---|---|---|
| Segment | 0 | (segment dịch trái 4 bit hay nhân 16) |
| + offset | | (offset giữ nguyên) |
| Địa chỉ vật lý | | (địa chỉ vật lý 20 bit) |

Ví dụ : tính địa chỉ vật lý tương ứng địa chỉ logic B001:1234

Địa chỉ vật lý = B0010h + 1234h = B1244h

# Địa chỉ vật lý

- Ví dụ: Địa chỉ vật lý 12345H

| Địa chỉ đoạn | Địa chỉ lệch |
|---|---|
| 1000 H | 2345H |
| 1200 H | 0345H |
| 1004 H | ? |
| 0300 H | ? |

- Ví dụ: Cho địa chỉ đầu của đoạn: 49000 H, xác định địa chỉ cuối?

# Các thanh ghi đoạn

- Ví dụ: Địa chỉ vật lý 12345H

| Địa chỉ đoạn | Địa chỉ lệch |
|---|---|
| 1000 H | 2345H |
| 1200 H | 0345H |
| 1004 H | 2305H |
| 0300 H | F345H |

- Ví dụ: Cho địa chỉ đầu của đoạn: 49000 H, → địa chỉ cuối: 58FFF H

# Sự chồng xếp các đoạn

- Địa chỉ segment (gọi là địa chỉ nền) của đoạn → cho biết điểm bắt đầu của đoạn trong bộ nhớ.
- Địa chỉ offset thể hiện khoảng cách kể từ đầu đoạn của ô nhớ cần tham khảo.
- Do offset dài 16 bit nên chiều dài tối đa của mỗi đoạn là 64KB.
- Trong mỗi đoạn:
  - ô nhớ đầu tiên có offset là 0000h
  - ô nhớ cuối cùng có offset là FFFFh.

# Sự chồng xếp các đoạn

Trong mỗi đoạn: ô nhớ đầu tiên có offset là 0000h và ô nhớ cuối cùng là FFFFh.

# Sự chồng xếp các đoạn

- Mỗi ô nhớ chỉ có địa chỉ vật lý nhưng có thể có nhiều địa chỉ logic.

*Ví dụ : 1234:1234*

*1334:0234*

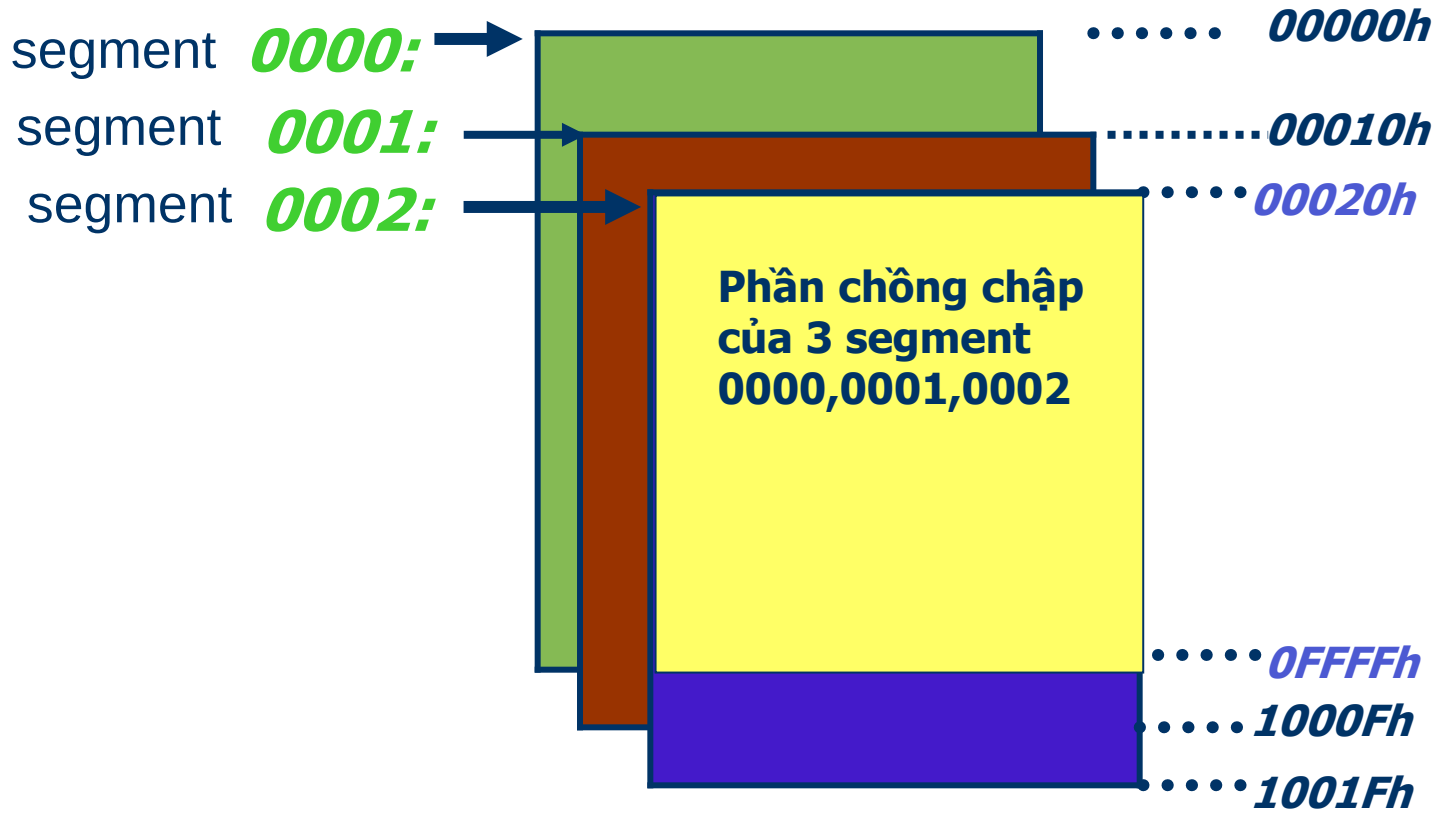
*1304:0534*

Đều có chung địa chỉ vật lý 13574h

***Tại sao ?***

# Sự chồng xếp các đoạn

Mối quan hệ giữa địa chỉ vật lý với segment và offset



# Giải thích

- 0000:0000 ➔Địa chỉ vật lý tương ứng là 00000h
- Giữ nguyên phần segment, tăng phần offset lên 1 thành ra địa chỉ logic là 0000:0001
  - → Địa chỉ vật lý là 00001h
- Tương tự với địa chỉ logic là 0000:0002
  - → Địa chỉ vật lý là 00002h
- Khi offset tăng 1 đơn vị thì địa chỉ vật lý  tăng 1 địa chỉ  hoặc là tăng 1 byte.
- Như vậy có thể xem đơn vị của offset là byte

# Giải thích

- Làm lại quá trình trên nhưng giữ nguyên phần offset chỉ tăng phần segment.

  0001:0000 ➔ 00010h

  0002:0000 ➔ 00020h

- Khi segment tăng 1 đơn vị thì địa chỉ vật lý tăng 10h địa chỉ hoặc là tăng 16 bytes
- Đơn vị của segment là paragraph

# Giải thích

- Ta thấy segment 0000 nằm ở đầu vùng nhớ, nhưng:
  - segment 0001 bắt đầu cách đầu vùng nhớ 16 bytes,
  - segment 0002 bắt đầu cách  đầu vùng nhớ 32 bytes…..
- Phần chồng chập 3 segment 0000, 0001, 0002 trên hình vẽ là vùng bộ nhớ mà bất kỳ ô nhớ nào nằm trong đó (địa chỉ vật lý từ 00020h đến 0FFFFh) đều có thể có địa chỉ logic tương ứng trong cả 3 segment.

# Giải thích

- Ví dụ: ô nhớ có địa chỉ 0002Dh sẽ có địa chỉ logic:
  - Trong segment 0000 là 0000:002D
  - Trong segment 0001 là 0001:001D
  - Trong segment 0002 là 0002:000D

➔ nếu vùng bộ nhớ nào càng có nhiều segment chồng chập lên nhau thì các ô nhớ trong đó càng có nhiều địa chỉ logic.

# Một ô nhớ có bao nhiêu địa chỉ logic

- **Một ô nhớ có:**
  - **Ít nhất 1 địa chỉ logic**
  - **Nhiều nhất là 65536/16 = 4096 địa chỉ logic**

# Địa chỉ logic và các thanh ghi

- Để tham khảo đến địa chỉ logic có segment trong thanh ghi DS, offset trong thanh ghi BX, ta viết DS:BX.
  - VD: Nếu lúc tham khảo, DS=2000h, BX=12A9h thì địa chỉ logic DS:BX chính là tham khảo đến ô nhớ 2000:12A9.
- Trong cách sử dụng địa chỉ logic, có một số cặp thanh ghi luôn luôn phải dùng chung với nhau một cách bắt buộc:
  - **CS:IP** *: lấy lệnh (địa chỉ lệnh sắp thi hành).*
  - **SS:SP** *: địa chỉ đỉnh chồng.*
  - **SS:BP** *: thông số trong chồng (dùng cho chương trình con).*
  - **DS:SI** *: địa chỉ chuỗi nguồn (chỉ có ý nghĩa trong các lệnh xử lý chuỗi).*
  - **ES:DI** *: địa chỉ chuỗi đích (chỉ có ý nghĩa trong các lệnh xử lý chuỗi).*

# Địa chỉ logic và các thanh ghi

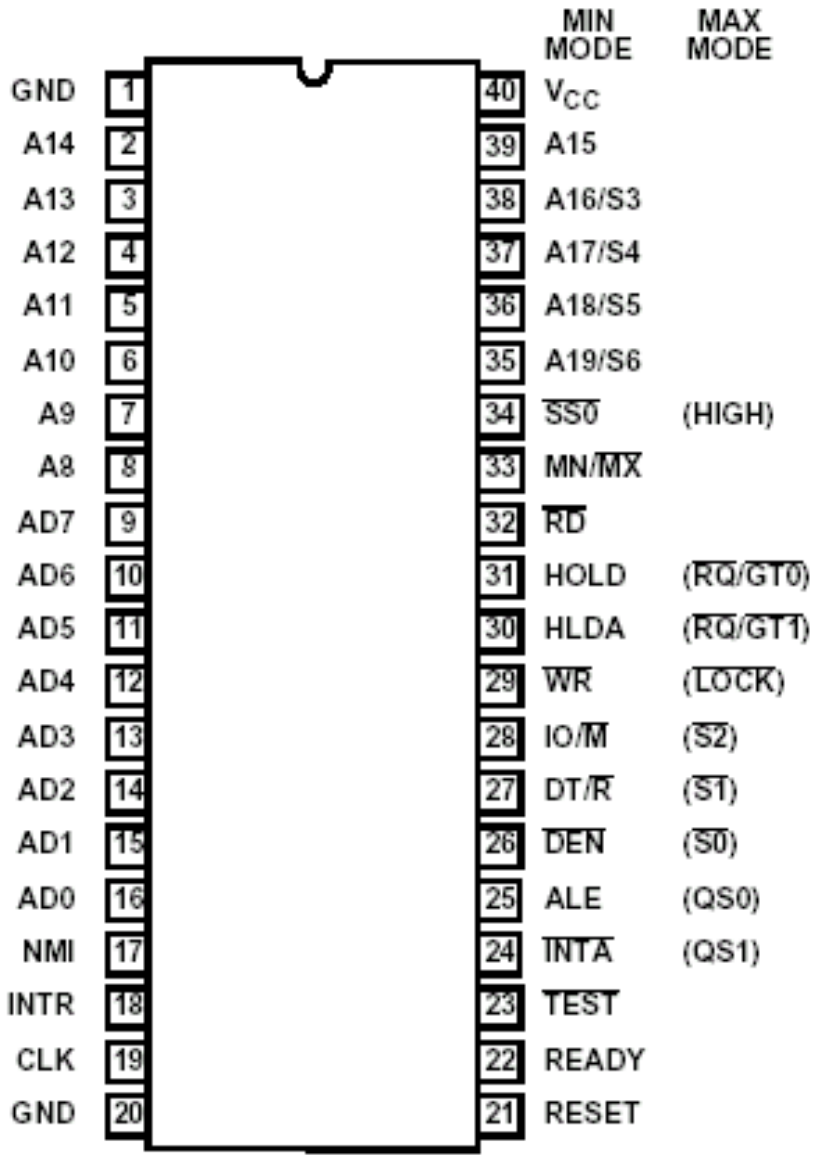
- Chương trình mà vi xử lý 8086 thi hành thường có 3 đoạn :
  - Đoạn chương trình có địa chỉ trong thanh ghi CS
  - Đoạn dữ liệu có địa chỉ trong thanh ghi DS
  - Đoạn ngăn xếp có địa chỉ trong thanh ghi SS

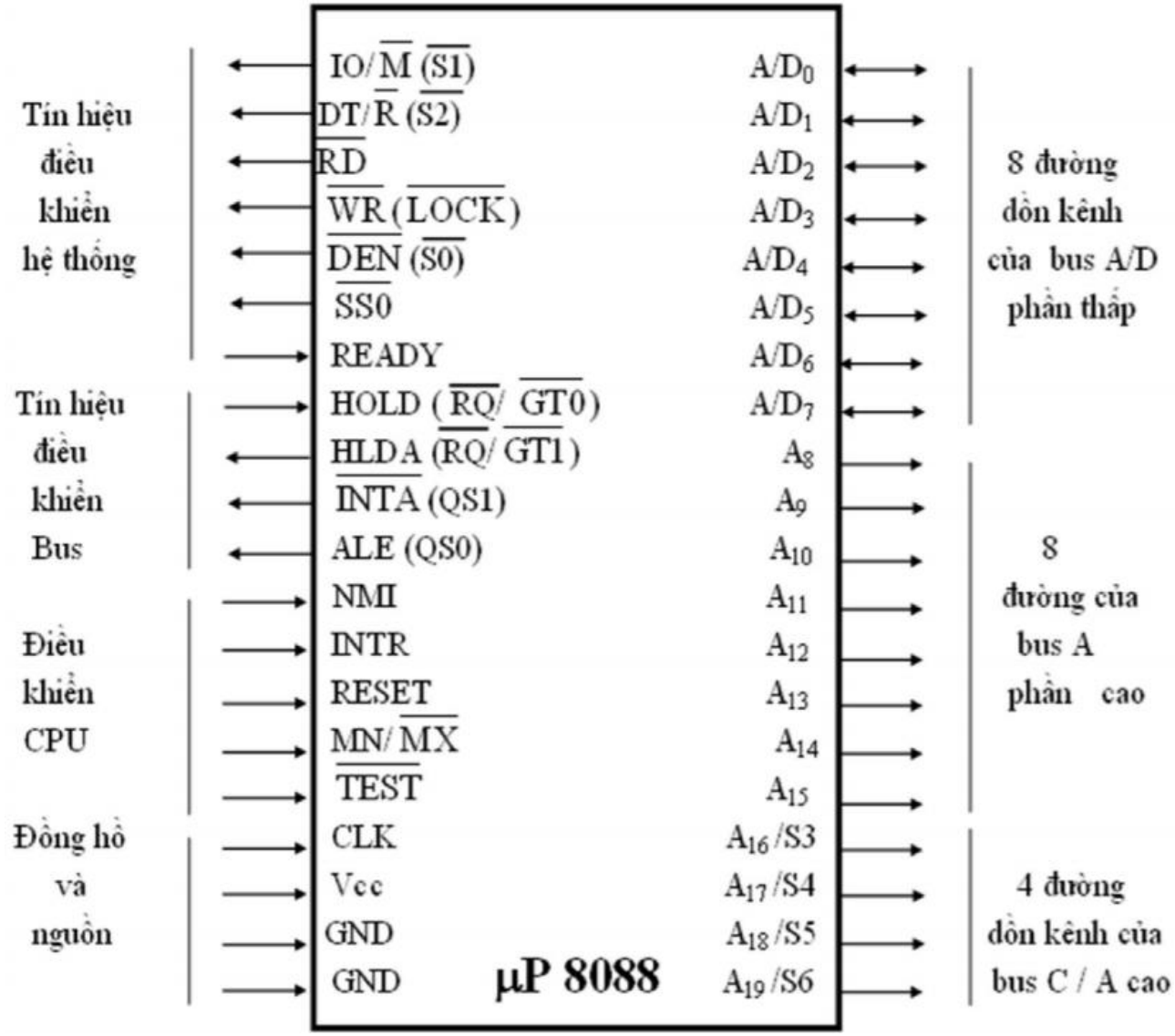
# Đoạn lệnh và thanh ghi con trỏ lệnh

- Thanh ghi CS sẽ xác định đoạn lệnh.
- Đoạn lệnh dùng để chứa lệnh của chương trình. Bộ vi xử lý sẽ nhận lần lượt từng lệnh ở đây để giải mã và thực hiện.
- Thanh ghi IP (con trỏ lệnh) chứa địa chỉ offset của lệnh tiếp theo sẽ được nhận vào.

  → CS:IP chứa địa chỉ logic của lệnh tiếp theo sẽ được nhận vào.

# Sơ đồ chân tín hiệu của 8088



# Các chân tín hiệu của 8088

- $AD_0$ → $AD_7$ [I, O]: Các chân đồn kênh cho các tín hiệu phần thấp của bus dữ liệu và bus địa chỉ.
  - Các chân này ở trạng thái trở kháng cao khi µP chấp nhận treo.
  - Khi ALE = 0: Trên các chân đó là các tín hiệu dữ liệu
  - Khi ALE = 1: Trên các chân đó là các tín hiệu địa chỉ
- $A_8$ → $A_{15}$ [O]: Các bit phần cao của bus địa chỉ.
  - Các chân này ở trạng thái trở kháng cao khi µP chấp nhận treo.

# Các chân tín hiệu của 8088

- $A_{16}/S_3$, $A_{17}/S_4$, $A_{18}/S_5$, $A_{19}/S_6$ [O]: Các chân dồn kênh của địa chỉ phần cao và trạng thái.
  - Các chân này ở trạng thái trở kháng cao khi µP chấp nhận treo.
  - Khi ALE=0: Trên các chân đó là các tín hiệu trạng thái $S_3$–$S_6$
  - Khi ALE=1: Trên các chân đó là các tín hiệu địa chỉ $A_{16}$ – $A_{19}$
  - Bit trạng thái $S_6$ luôn =0
  - $S_5$ chỉ trạng thái cờ IF.
  - $S_4$, $S_3$ xác định đoạn nhớ đang được truy nhập:

| $S_4S_3$ | Truy nhập đến |
|---|---|
| 00 | Đoạn dữ liệu phụ |
| 01 | Đoạn ngăn xếp |
| 10 | Đoạn lệnh hoặc không đoạn nào |
| 11 | Đoạn dữ liệu |

# Các chân tín hiệu của 8088

- $\overline{RD}$ [O]: Tín hiệu điều khiển đọc dữ liệu từ bộ nhớ, thiết bị ngoại vi.
  - Chân này ở trạng thái trở kháng cao khi µP chấp nhận treo.
  - Khi $\overline{RD}$ = 0: Bus dữ liệu sẵn sàng nhận dữ liệu từ bộ nhớ hoặc thiết bị ngoại vi
  - Khi $\overline{RD}$ = 1: Ngược lại
- READY [I]: Tín hiệu vào thông báo cho CPU biết bộ nhớ hoặc thiết bị ngoại vi sẵn sàng làm việc.
  - Khi READY = 0: CPU thực hiện lệnh ghi/đọc mà không cần chèn thêm các chu kì đợi
  - Khi READY = 1: CPU tự kéo dài thời gian thực hiện lệnh ghi/đọc bằng cách chèn thêm các chu kì đợi
- INTR [I]: (*Interupt Request*) tín hiệu vào yêu cầu ngắt che được.
  - Khi có yêu cầu ngắt mà cờ cho phép ngắt IF=1 thì CPU kết thúc lệnh đang làm dở, sau đó nó đi vào chu kỳ chấp nhận ngắt và đưa ra bên ngoài tín hiệu INTA=0.

# Các chân tín hiệu của 8088

- $\overline{\text{TEST}}$ [I]: tín hiệu vào để kiểm tra bộ VXL bằng lệnh WAIT.
  - Khi CPU thực hiện lệnh WAIT mà lúc đó tín hiệu $\overline{\text{TEST}}$ = 1, nó sẽ chờ cho đến khi tín hiệu $\overline{\text{TEST}}$=0 thì mới thực hiện lệnh tiếp theo.
- NMI [I]: đầu vào tín hiệu yêu cầu ngắt không che được.
  - Tín hiệu này không bị khống chế bởi cờ IF và nó sẽ được CPU nhận biết bằng tác động của sườn lên của xung yêu cầu ngắt.
  - Nhận được yêu cầu này CPU kết thúc lệnh đang làm dở, sau đó nó chuyển sang thực hiện chương trình ngắt kiểu INT 2.
- CLK [I]: Tín hiệu đồng hồ (xung nhịp) đưa vào cho bộ VXL lấy từ bộ phát xung nhịp 8284 .

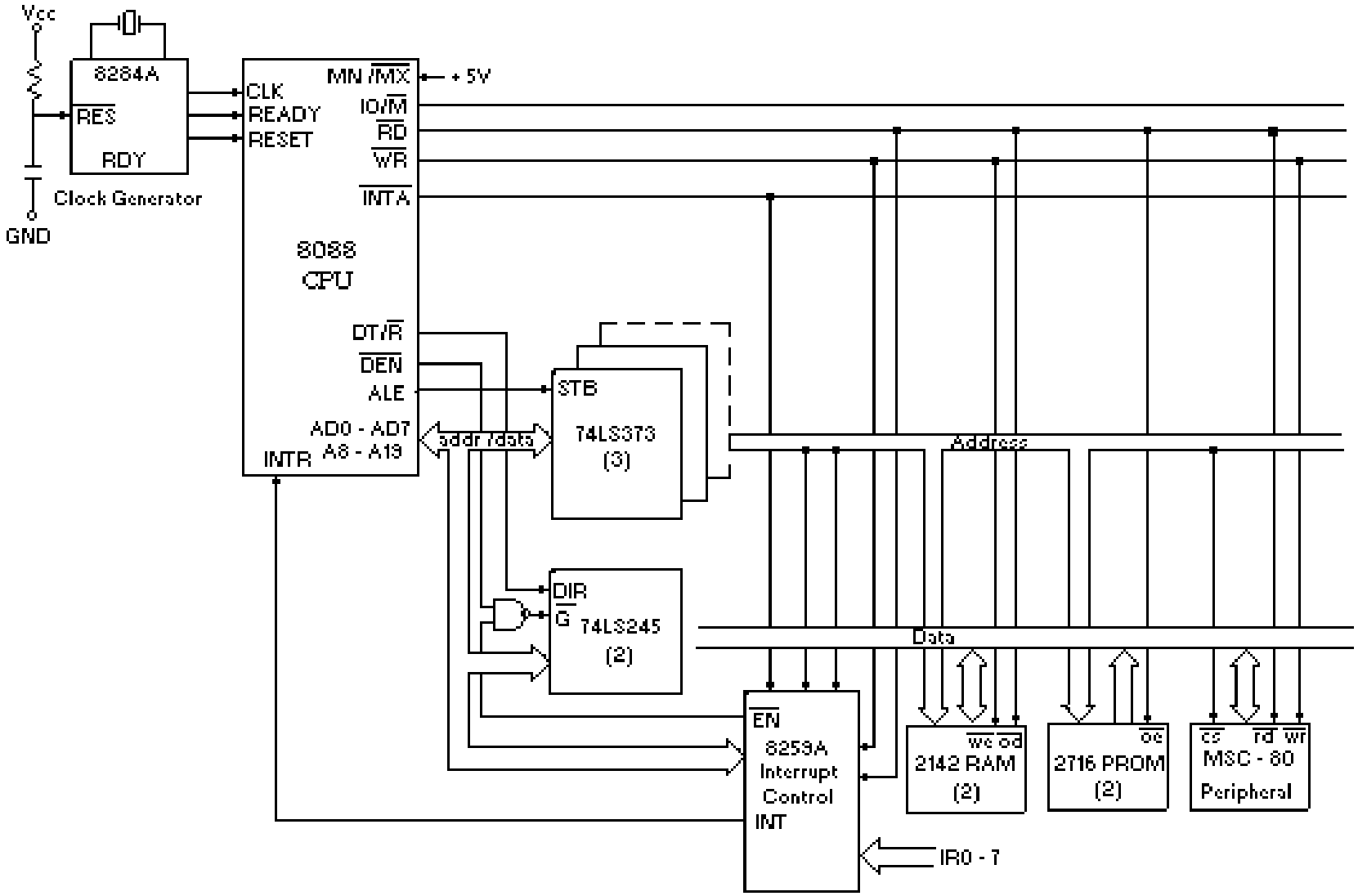
# Các chân tín hiệu của 8088

- RESET [I]: tín hiệu vào để khởi động lại bộ VXL, địa chỉ khởi động FFFF0h.
  - Khi RESET=1 kéo dài ít nhất trong thời gian 4 chu kỳ đồng hồ thì 8088 bị buộc phải khởi động lại:
    - Xoá các thanh ghi DS, ES, SS, IP và FR về 0
    - Bắt đầu thực hiện chương trình tại địa chỉ CS:IP=FFFFH:0000H
      - Cờ IF=0 để cấm các yêu cầu ngắt khác tác động vào CPU
      - Cờ TF=0 để bộ vi xử lý không bị đặt trong chế độ chạy từng lệnh.

# Các chân tín hiệu của 8088

- Vcc [I]:  Chân nguồn. Tại đây CPU được cung cấp +5V±10%, 340mA.
- GND (Ground -đất) [O]:  2 chân nguồn để nối với điểm 0V của nguồn nuôi.
- MN/MX [I]: Tín hiệu vào điều khiển hoạt động của CPU theo chế độ MIN/MAX, tín hiệu này được xác lập cố định
  - Chế độ min: MN/MX nối với +5V
  - Chế độ max: MN/MX nối với đất

# Các chân 8088 ở chế độ MIN



# Các chân 8088 ở chế độ MIN

- IO/$\overline{M}$ [O]: Chỉ ra rằng CPU đang thâm nhập bộ nhớ hay cổng I/O.
  - Tín hiệu phân biệt trong thời điểm nhất định phần tử nào trong các thiết bị vào/ra (IO) hoặc bộ nhớ được chọn làm việc với CPU.
  - Trên bus địa chỉ lúc đó sẽ có các địa chỉ tương ứng của thiết bị đó.
  - Chân này ở trạng thái trở kháng cao khi µP chấp nhận treo.
- $\overline{WR}$ [O]: Xung cho phép ghi.
  - Khi CPU đưa ra $\overline{WR}$=0 thì trên bus dữ liệu các dữ liệu đã ổn định vào bộ nhớ hoặc thiết bị ngoại vi tại thời điểm đột biến $\overline{WR}$=1.
  - Chân này ở trạng thái trở kháng cao khi µP chấp nhận treo.

# Các chân 8088 ở chế độ MIN

- $\overline{\text{INTA}}$ (*Interupt Acknowledge*) [O]:
  - Tín hiệu báo cho các mạch bên ngoài biết CPU chấp nhận yêu cầu ngắt INTR.
  - Lúc này CPU đưa ra INTA=0 để báo là nó đang chờ mạch ngoài đưa vào số hiệu ngắt (kiểu ngắt) trên bus dữ liệu.
- ALE (*Address Latch Enable*)[O]: Xung cho phép chốt địa chỉ.
  - ALE = 0: Trên bus dồn kênh AD có các địa chỉ của thiết bị vào ra hay ô nhớ
  - ALE = 0: Khi CPU bị treo.
  - Tín hiệu thông báo rằng bus địa chỉ dữ liệu có chứa địa chỉ.
  - ALE không bao giờ bị thả nổi (trong trạng thái trở kháng cao), khi CPU bị treo thì ALE=0.

# Các chân 8088 ở chế độ MIN

- DT/$\overline{R}$ (*Data Trasmit/Receiver* ) [O]: Tín hiệu điều khiển các đệm 2 chiều của bus dữ liệu để chọn chiều chuyển vận của dữ liệu trên bus D.
  - Khi DT/$\overline{R}$ = 0: dữ liệu đi ra từ CPU
  - Khi DT/$\overline{R}$ = 1: dữ liệu đi vào CPU
- $\overline{DEN}$ (*Data Enable*) [O]:
  - Tín hiệu ra báo cho bên ngoài biết là lúc này trên bus dồn kênh AD có dữ liệu ổn định.
  - ở trạng thái trở kháng cao khi µP chấp nhận treo.

# Các chân 8088 ở chế độ MIN

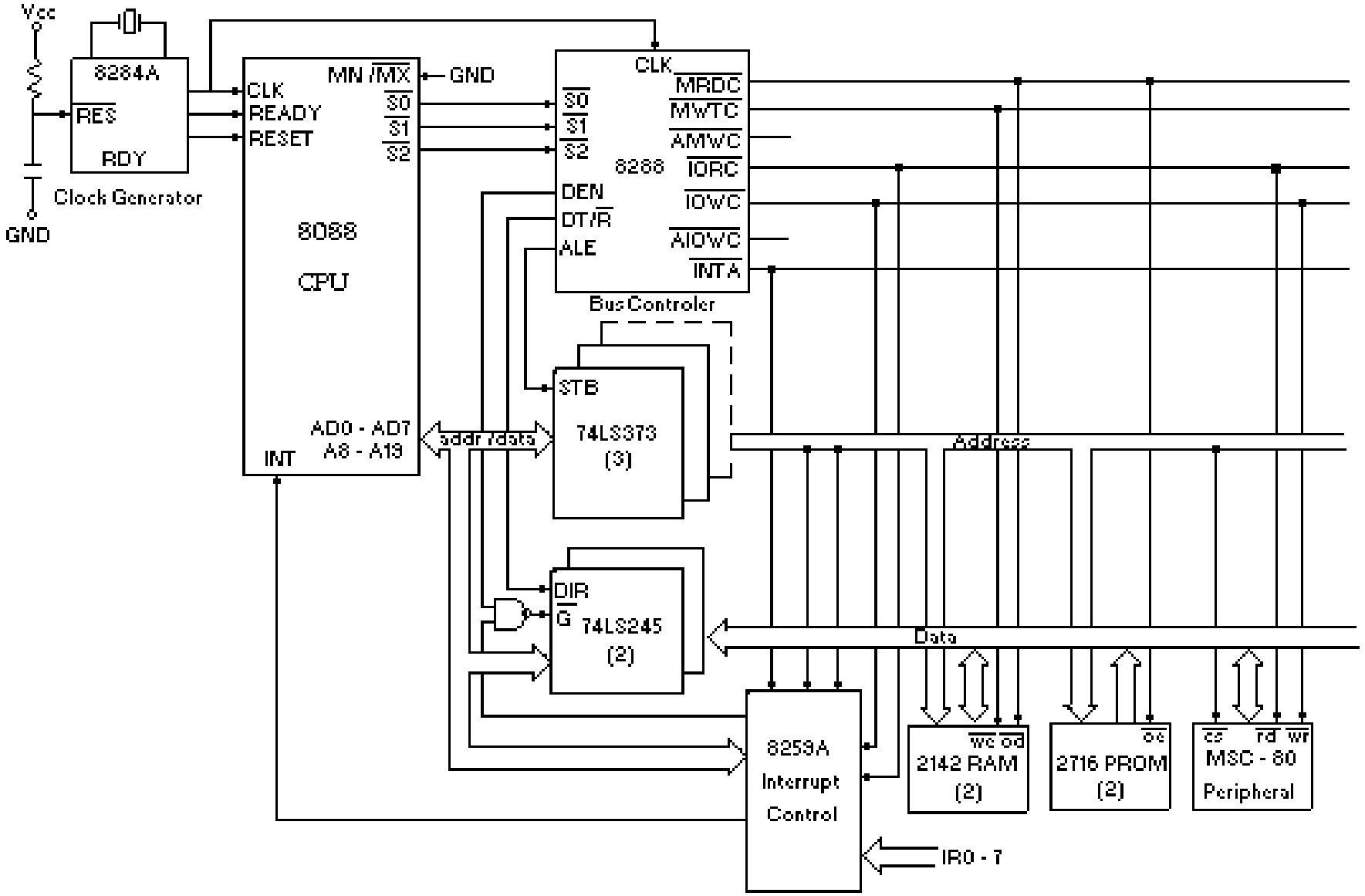
- HOLD [I]:
  - Tín hiệu vào yêu cầu treo (thả nổi) CPU để mạch ngoài thực hiện việc trao đổi dữ liệu với bộ nhớ bằng cách thâm nhập trực tiếp (DMA).
  - Khi HOLD=1, CPU 8088 sẽ tự tách ra khỏi hệ thống bằng cách treo tất cả các bus A, bus D, bus C của nó (các bus ở trạng thái trở kháng cao) để bộ điều khiển DMA (DMAC- DMA Controller) có thể lấy được quyền điều khiển hệ thống để làm các công việc trao đổi dữ liệu.

# Các chân 8088 ở chế độ MIN

- HLDA (*Hold Acknowledge*) [O]:
  - Tín hiệu báo cho bên ngoài biết yêu cầu treo CPU để dùng các bus đã được chấp nhận, và CPU 8088 đã treo các bus A, bus D và một số tín hiệu của bus C.
- $\overline{SS_0}$ [O]: Tín hiệu trạng thái, tín hiệu này cũng giống như trong chế độ MAX và được dùng để kết hợp với IO/$\overline{M}$ và DT/$\overline{R}$ để giải mã các chu kỳ hoạt động của bus theo bảng sau

| DT/$\overline{R}$ | IO/$\overline{M}$ | $\overline{SS_0}$ | Chu kỳ điều khiển của bus I/O |
|---|---|---|---|
| **0** | **0** | **0** | Đọc mã lệnh |
| **0** | **0** | **1** | Đọc bộ nhớ |
| **0** | **1** | **0** | Ghi bộ nhớ |
| **0** | **1** | **1** | Bus rỗi (nghỉ) |
| **1** | **0** | **0** | Chấp nhận yêu cầu ngắt |
| **1** | **0** | **1** | Đọc cổng |
| **1** | **1** | **0** | Ghi cổng I/O |
| **1** | **1** | **1** | Dừng |

# Các chân 8088 ở chế độ MAX



# Các chân 8088 ở chế độ MAX

- $\overline{\text{AIOWC}}$, $\overline{\text{IOWC}}$, $\overline{\text{MWTC}}$ [O] :
  - Các chân trạng thái dùng trong chế độ MAX để ghép với mạch điều khiển bus 8288.
  - Các tín hiệu này được 8288 dùng để tạo ra các tín hiệu điều khiển trong các chu kỳ hoạt động của bus.
    - $\overline{\text{AMWC}}$: Advanced
    - $\overline{\text{AIOWC}}$: Advanced
    - IORC: Input Output Read Command
    - IOWC: Input Output Write Command
    - MRDC: Memory Read Command
    - MWTC: Memory Write Command

| Tín hiệu | Chu kỳ điều khiển của bus |
|---|---|
| 000 | Trả lời ngắt |
| 001 | Đọc cổng I/OIORC |
| 010 | Ghi cổng I/OIOWC |
| 011 | Dừng Không |
| 100 | Đọc mã lệnh MRDC |
| 101 | Đọc bộ nhớ MRDC |
| 110 | Ghi bộ nhớ MWTC |
| 111 | Bus rỗi (nghỉ)Không |

# Các chân 8088 ở chế độ MAX

- $\overline{RQ}/\overline{GT_1}$, $\overline{RQ}/\overline{GT_0}$ (Request /Grant)[I/O]:
  - Các tín hiệu yêu cầu dùng bus của các bộ xử lý khác hoặc thông báo chấp nhận treo của CPU để cho phép các bộ vi xử lý khác dùng bus.
  - $\overline{RQ}/\overline{GT_1}$ có mức ưu tiên cao hơn $\overline{RQ}/\overline{GT_0}$.
    - Tín hiệu hai chiều RQ -yêu cầu treo bus
    - GT - tín hiệu báo ra rằng CPU chấp nhận treo.

# Các chân 8088 ở chế độ MAX

- $\overline{\text{LOCK}}$ [O]: Tín hiệu do CPU đưa ra để cấm các bộ VXL khác (mạch điều khiển khác) sử dụng bus trong khi nó đang thi hành 1 lệnh nào đó đặt sau tiếp đầu LOCK.
- $QS_1$, $QS_0$ (*Queu Status*) [O]: Cho biết trạng thái của hàng đợi lệnh.

| $QS_1QS_0$ | Trạng thái đệm lệnh |
|---|---|
| 00 | Không hoạt động |
| 01 | Đọc Byte đầu tiên từ đệm lệnh |
| 10 | Hàng đệm lệnh rỗng |
| 11 | Đọc byte tiếp theo từ đệm lệnh |

# Chương 2: Bộ vi xử lý Intel 8088/8086

- 2.1. Cấu trúc của bộ vi xử lý Intel 8088
- 2.2. Các thanh ghi của bộ vi xử lý Intel 8088
- 2.3. Nguyên tắc làm việc của bộ vi xử lý Intel 8088
- 2.4. Các chế độ địa chỉ của bộ vi xử lý 8088
- 2.5. Các bộ vi xử lý thế hệ sau 8088 của Intel

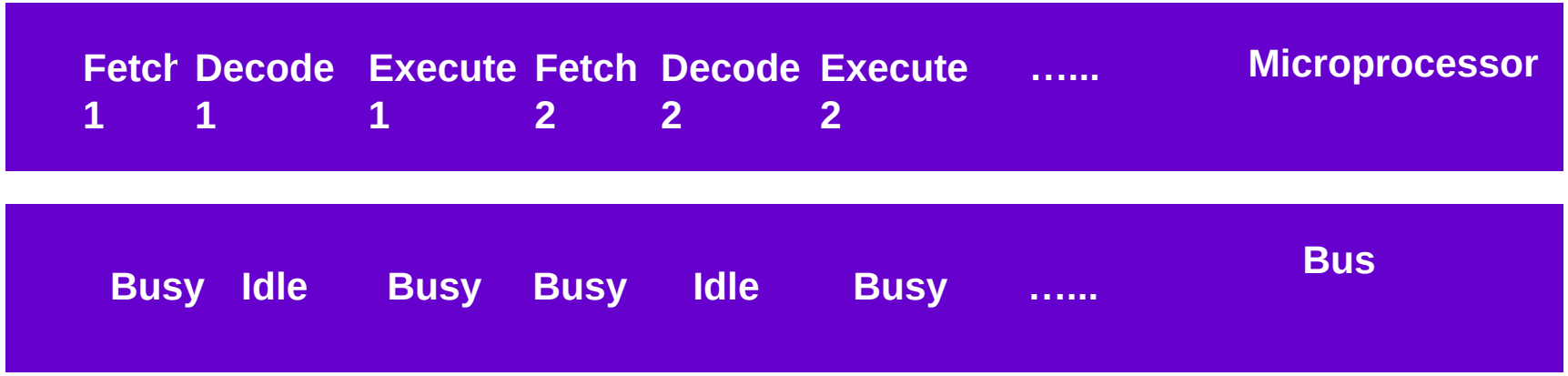
# 2.3. Nguyên tắc làm việc của VXL Intel 8086/8088

- Trong các bộ vi xử lý ở các thế hệ trước (vd 8085), hoạt động của CPU gồm 3 giai đoạn:
  - Đọc mã lệnh (opcode fetch)
  - Giải mã lệnh (decode)
  - Thực hiện lệnh (execution)

→8088/86 sử dụng cơ chế xen kẽ dòng mã lệnh

# Xử lý lệnh của các vi xử lý trước 8086/8088

- CPU thông thường dùng chu kỳ nhận và thực thi lệnh tuần tự.
- Một thủ tục đơn giản gồm 3 bước:
  - Lấy lệnh từ bộ nhớ
  - Giải mã lệnh
  - Thực hiện lệnh
    - Lấy các toán hạng từ bộ nhớ (nếu có)
    - Lưu trữ kết quả



# Nhận lệnh và thực thi lệnh của VXL 8088/86

- 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).
- Kiến trúc dạng pipeline của 8086/8088 cho phép thực thi các lệnh mà không bị trễ do quá trình nhận lệnh:
  - EU có thể thực thi các lệnh gần như liên tục thay vì phải đợi BIU nhận thêm lệnh mới.

# Quá trình nhận lệnh và thực thi lệnh

- BIU đưa nội dung của thanh ghi con trỏ lệnh IP ra bus địa chỉ để chọn byte hay word đọc vào BIU.
- Thanh ghi IP được tăng lên để chuẩn bị nhận lệnh kế tiếp
  - Số byte tăng lên của IP tùy thuộc vào kích thước lệnh trước đó.
- Lệnh ở trong BIU được đưa sang hàng lệnh (queue).
  - Đây là một thanh ghi lưu trữ dạng FIFO, dùng cơ chế xử lý xen kẽ liên tục các dòng mã lệnh (kỹ thuật đường ống – pipelining).

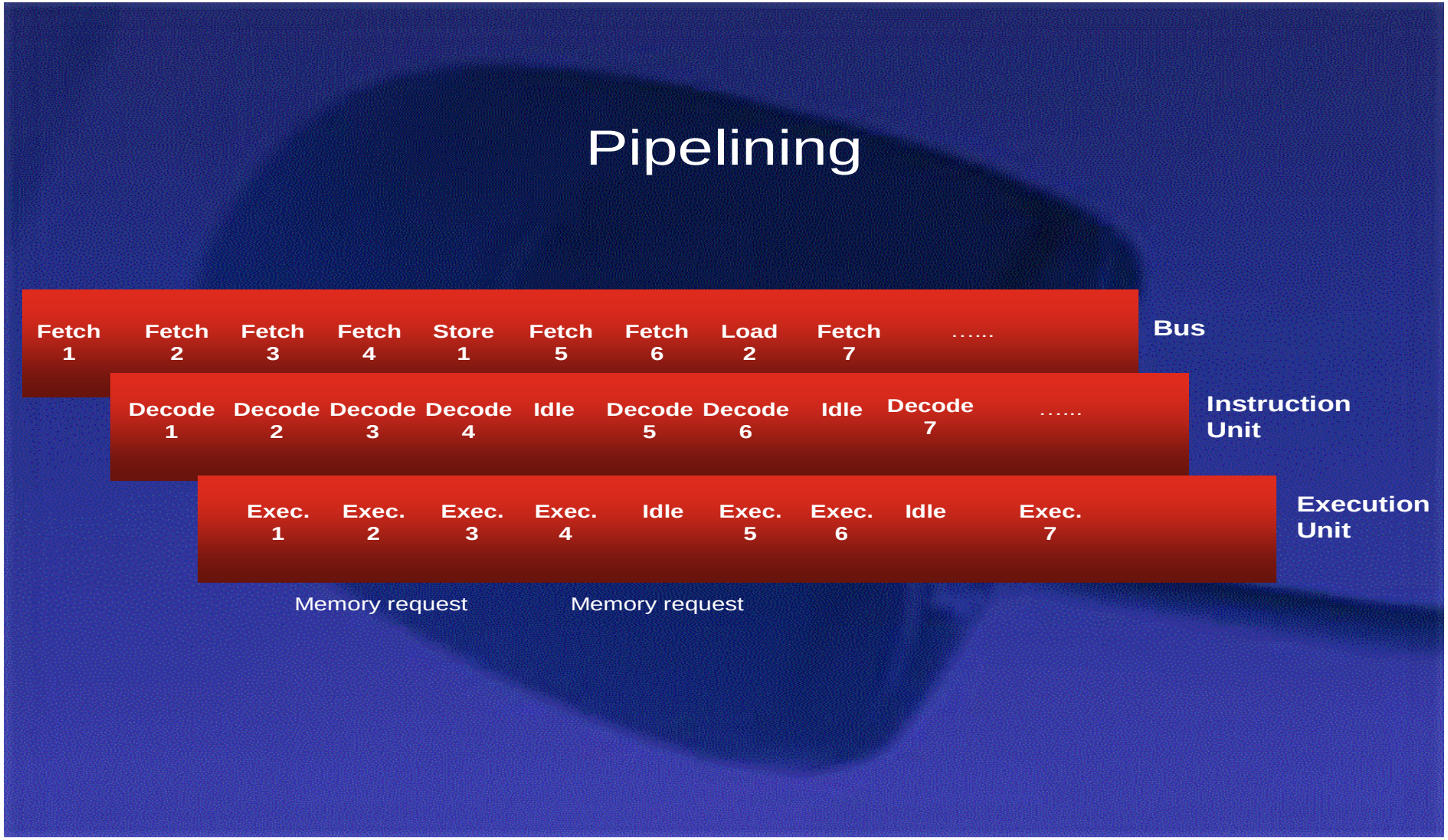
# Quá trình nhận lệnh và thực thi lệnh

- 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.
- 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

# Thực hiện lệnh

- CPU thực hiện lệnh tuần tự theo các bước :
  - B1: Nhận lệnh từ bộ nhớ → thanh ghi lệnh.
  - B2: Thay đổi IP để chỉ đến lệnh kế tiếp.
  - B3: Xác định kiểu lệnh vừa lấy ra.
  - B4: Xác định kiểu dữ liệu vừa yêu cầu và xác định vị trí dữ liệu trong bộ nhớ.
  - B5: Nếu lệnh cần dữ liệu trong bộ nhớ, nạp nó vào thanh ghi của CPU
  - B6: Thực hiện lệnh.
  - B7: Lưu kết quả ở nơi thích hợp.
  - B8: Trở về bước 1 để thực hiện lệnh kế tiếp.

# Cơ chế Pipelining



# Hoạt động đọc/ghi dữ liệu của 8088

- **Hoạt động đọc/ghi:**
  - Nếu dữ liệu được **ghi tới bộ nhớ**, bộ VXL sẽ:
    - địa chỉ ô nhớ trên bus địa chỉ
    - Dữ liệu cần ghi trên bus dữ liệu
    - Đưa tín hiệu điều khiển ghi $\overline{WR} = 0$ tới bộ nhớ
    - Đưa ra tín hiệu chọn bộ nhớ $IO/\overline{M} = 0$.
  - Nếu dữ liệu được **đọc từ bộ nhớ**, bộ VXL sẽ
    - Đưa ra địa chỉ bộ nhớ trên bus địa chỉ
    - Đưa ra tín hiệu đọc bộ nhớ $\overline{RD} = 0$
    - Đưa ra tín hiệu chọn bộ nhớ $IO/\overline{M} = 0$.

    →nhận dữ liệu qua bus dữ liệu.
  - Hoạt động đọc/ghi dữ liệu với cổng vào/ra cũng tương tự.

# Hoạt động đọc/ghi dữ liệu của 8088

- Mỗi chu kỳ đọc ghi (còn gọi là chu kỳ bus) của CPU kéo dài 4 chu kỳ đồng hồ.
- Các chu kỳ đồng hồ được ký hiệu là T1, T2, T3 và T4.
- Nếu CPU làm việc với tần số đồng hồ (xung nhịp) là 5 MHz thì một chu kỳ đồng hồ kéo dài T =200 ns và một chu kỳ bus kéo dài 4×T = 800 ns

# Đáp ứng ngắt

- Vi xử lý 8086 sử dụng 3 loại ngắt:
  - Ngắt hệ thống:
    - do CPU phát ra khi có một lỗi nghiêm trọng xảy ra trong quá trình hoạt động của nó.
    - VD: chia cho số 0, điện áp nguồn cung cấp giảm thấp, chia tràn . . .
  - Ngắt cứng:
    - Do thiết bị ngoại vi gây ra khi cần trao đổi thông tin với CPU.
    - Đặc trưng của ngắt cứng là tín hiệu yêu cầu ngắt INTR.
  - Ngắt mềm:
    - Do thi hành lệnh INT trong chương trình.
    - Thực chất của ngắt mềm chính là một dạng gọi đến chương trình con.

# Đáp ứng ngắt

- Mục đích của việc phục vụ ngắt là bằng cách nào đó chuyển điều khiển sang cho một chương trình con đặc biệt gọi là chương trình phục vụ ngắt của riêng ngắt được phục vụ.
- Đối với VXL 8086, việc phục vụ ngắt được thực hiện thông qua số ngắt của từng ngắt.
  - Mỗi ngắt có một số ngắt riêng.
  - Số ngắt là một số 1 byte

    →VXL 8086 chỉ có thể phục vụ cho tối đa 256 ngắt.
- VXL 8086 sử dụng phương pháp vector ngắt để chuyển điều khiển đến các chương trình phục vụ ngắt.

# Đáp ứng ngắt (t)

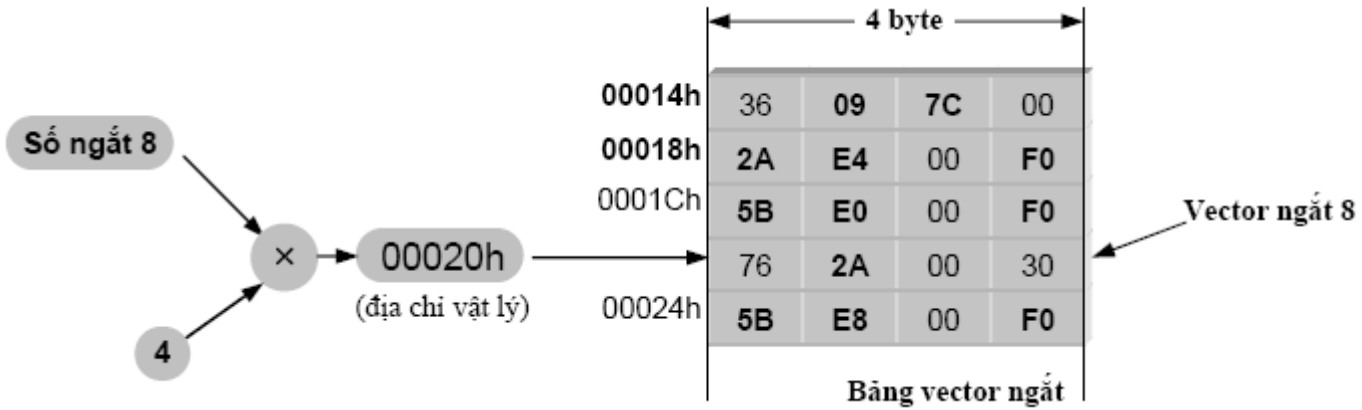
- Vector ngắt là các biến bộ nhớ dài 4 bytes, có khả năng chứa được một địa chỉ logic đầy đủ gồm 2 byte segment và 2 byte offset. Dùng vector ngắt để chứa địa chỉ bắt đầu của chương trình phục vụ ngắt.
- Các vector ngắt được xếp nối tiếp nhau kể từ đầu của vùng bộ nhớ tạo thành bảng vector ngắt.

# Đáp ứng ngắt (t)

- Chiều dài của bảng vector ngắt là 256x4=1024 hay 400h → bảng vector ngắt sẽ nằm trong vùng bộ nhớ có địa chỉ vật lý từ 00000h đến 003FFh.
- Số thứ tự của các vector ngắt được qui định chính là số ngắt tương ứng nên vị trí của vector ngắt được xác định theo cách:
  - **địa chỉ vật lý của vector ngắt = số ngắt × 4**

# Đáp ứng ngắt (t)

- Sau khi xác định được vị trí của vector ngắt rồi, CPU sẽ lấy địa chỉ chương trình phục vụ ngắt trong vector ngắt ra và chuyển điều khiển đến đó. Tức là nó sẽ thực hiện một lệnh gọi đến chương trình phục vụ ngắt.
- Ví dụ: để phục vụ cho ngắt số 8, CPU sẽ chạy chương trình con có địa chỉ trong vector ngắt 8 (ở địa chỉ vật lý 8x4=32=20h) mà cụ thể là địa chỉ logic 3000:2A76.



# Đáp ứng ngắt (t)

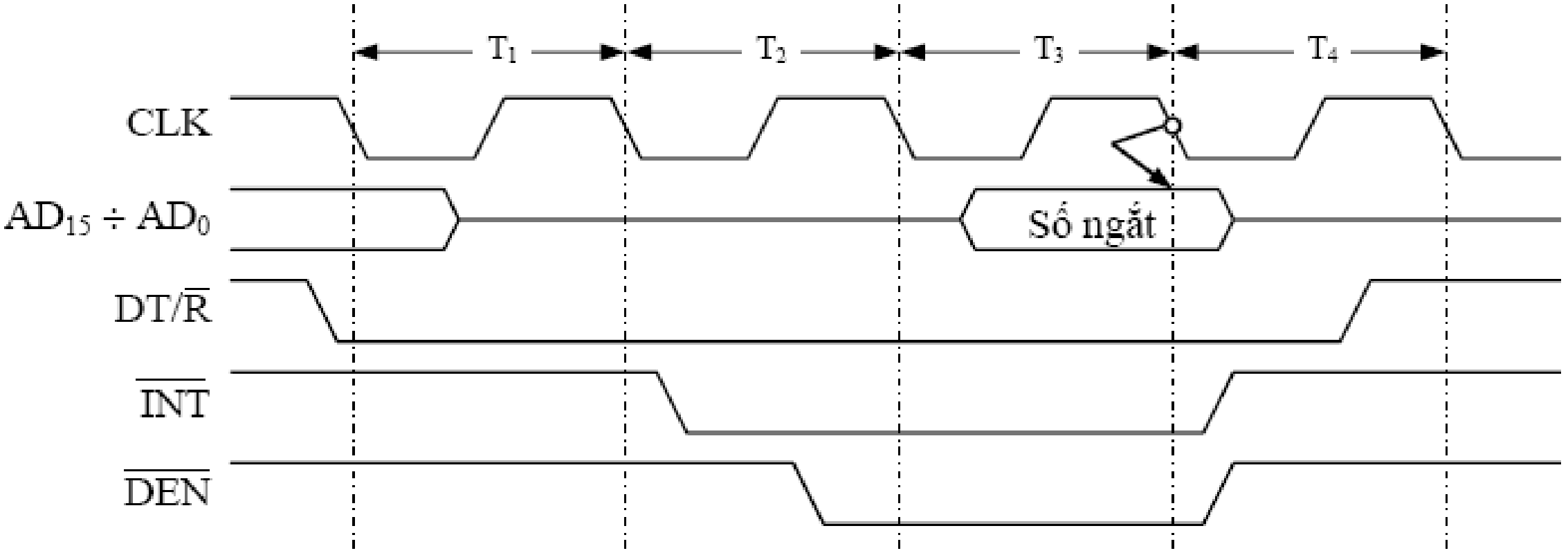
- Hoạt động đáp ứng ngắt của VXL 8086 chỉ dùng cho ngắt cứng.
- VXL 8086 dùng hoạt động này để đọc số ngắt tương ứng từ khối xuất nhập.
- Hoạt động đáp ứng ngắt được thực hiện bằng chu kỳ máy đáp ứng ngắt kéo dài trong 4T.
- Tuyến địa chỉ không được dùng trong chu kỳ đáp ứng ngắt.

# Đáp ứng ngắt (t)

- Các tín hiệu điều khiển gồm có :
  - $\overline{DEN}$=0
  - DT/$\overline{R}$=0
  - $\overline{INTA}$=0
- Tín hiệu INTA là tín hiệu đặc trưng cho chu kỳ máy đáp ứng ngắt.
- Tín hiệu yêu cầu ngắt INTR được kiểm tra ở cuối mỗi chu kỳ lệnh nghĩa là CPU phải thi hành xong lệnh hiện tại rồi mới chuyển sang hoạt động đáp ứng ngắt.
  - Khi đó nó phát ra 2 chu kỳ máy đáp ứng ngắt liên tiếp mà còn gọi là chu kỳ INTA.
- Sau chu kỳ INTA thứ 2, sự thi hành lệnh được chuyển sang chương trình con phục vụ ngắt.

# Đáp ứng ngắt (t)

- Giản đồ xung chu kỳ máy đáp ứng ngắt như sau:



# Trình tự khởi động

- Khi bật nguồn hoặc nhấn Reset:
  - CS=FFFFh và IP=0000 → địa chỉ FFFF0 chứa chỉ thị chuyển điều khiển đến điểm khởi đầu của các chương trình BIOS
  - Các chương trình BIOS kiểm tra hệ thống và bộ nhớ
  - Các chương trình BIOS khởi tạo bảng vector ngắt và vùng dữ liệu BIOS
  - BIOS nạp chương trình khởi động (boot program) từ đĩa vào bộnhớ
  - Chương trình khởi động nạp hệ điều hành từ đĩa vào bộ nhớ
  - Hệ điều hành nạp các chương trình ứng dụng

# Chương 2: Bộ vi xử lý Intel 8088/8086

- 2.1. Cấu trúc của bộ vi xử lý Intel 8088
- 2.2. Các thanh ghi của bộ vi xử lý Intel 8088
- 2.3. Nguyên tắc làm việc của bộ vi xử lý Intel 8088
- 2.4. Các chế độ địa chỉ của bộ vi xử lý 8088
- 2.5. Các bộ vi xử lý thế hệ sau 8088 của Intel

# 2.4. Các chế độ địa chỉ của bộ vi xử lý 8088

- Chế độ địa chỉ là cách xác định toán hạng của lệnh
- Toán hạng gồm: toán hạng nguồn và toán hạng đích
  - Họ Intel x86: nếu trong lệnh có 2 toán hạng thì toán hạng đích được viết ở bên trái.
- Toán hạng có thể là:
  - Hằng số (được cho ngay trong lệnh)
  - Nội dung của thanh ghi (trong lệnh cần cho biết tên thanh ghi)
  - Nội dung của ngăn nhớ
  - Nội dung của cổng vào-ra

# Các chế độ địa chỉ

- Chế độ địa chỉ thanh ghi
  - Register Addressing Mode
- Chế độ địa chỉ tức thì
  - Immediate Addressing Mode
- Chế độ địa chỉ trực tiếp
  - Register Direct Addressing Mode
- Chế độ địa chỉ gián tiếp qua thanh ghi
  - Register Indirect Addressing Mode
- Chế độ địa chỉ tương đối cơ sở
  - Based Relative Addressing Mode
- Chế độ địa chỉ tương đối chỉ số
  - Indexed Relative Addressing Mode
- Chế độ địa chỉ tương đối chỉ số cơ sở
  - Based Indexed Relative Addressing Mode

# Địa chỉ hiệu dụng (EA – Effective Address)

- Toán hạng bộ nhớ dùng trong tập lệnh vi xử lý 86 sử dụng phương pháp định địa chỉ tổng hợp: địa chỉ hiệu dụng
- Địa chỉ hiệu dụng: tổ hợp của 3 nhóm sau đặt trong dấu [ ]

**Nhóm thanh ghi chỉ số : SI , DI**
**Nhóm thanh ghi cơ sở: BX, BP**
**Địa chỉ trực tiếp: số 16 bit**

**Các thanh ghi trong cùng 1 nhóm không được xuất hiện trong cùng 1 địa chỉ hiệu dụng**

# Địa chỉ hiệu dụng

- **Địa chỉ hiệu dụng hợp lệ:**
  - [1000h], [SI], [DI] , [BX] , [BP]
  - [SI+BX], [SI+BP], [DI+BX], [DI+BP], [SI+1000h], [DI+100h]
  - [SI] [BX] [1000h], [SI+BP+1000h] , [DI+BX][1000h], [DI+1000h]+[BP]
- **Địa chỉ hiệu dụng không hợp lệ:**
  - [70000h], [AX] , [SI+DI+1000h], [BX] [BP]

# Địa chỉ hiệu dụng

- Địa chỉ hiệu dụng chính là phần offset của địa chỉ logic bộ nhớ
- Segment của địa chỉ hiệu dụng được mặc định là:
  - DS: nếu không sử dụng BP
  - SS: nếu có sử dụng BP

# Địa chỉ hiệu dụng

- Qui ước
  - Dữ liệu 8 bit bộ nhớ : [ địa chỉ ]
  - Dữ liệu 16 bit bộ nhớ : [ địa chỉ +1, địa chỉ ]
- Để xác định rõ hoạt động của bộ nhớ, ta phải dùng thêm toán tử PTR như sau :
  - 8 bit : BYTE PTR [1000H]
    - Tham khảo 1 byte bộ nhớ ở địa chỉ 1000h
  - 16 bit : WORD PTR [1000H]
    - Tham khảo 2 byte bộ nhớ liên tiếp ở địa chỉ 1000h và 1001h

# 2.4.1. Chế độ địa chỉ thanh ghi

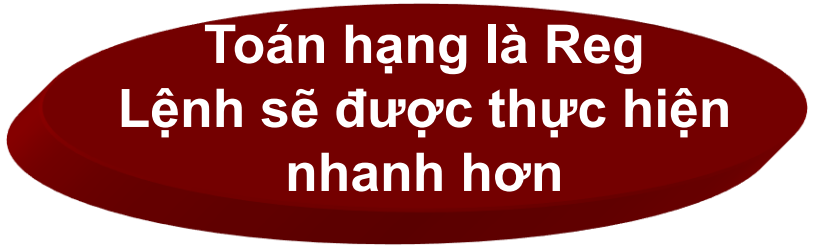
- Dùng các thanh ghi bên trong CPU như là các toán hạng để chứa dữ liệu cần thao tác.
- Tốc độ thực hiện lệnh cao
- Ví dụ:
  - MOV BX, DX
  - MOV AL, BL
  - MOV AL, BX

    ; không hợp lệ vì các thanh ghi có kích thước khác nhau
  - MOV ES, DS

    ; không hợp lệ (segment to segment)
  - MOV CS, AX

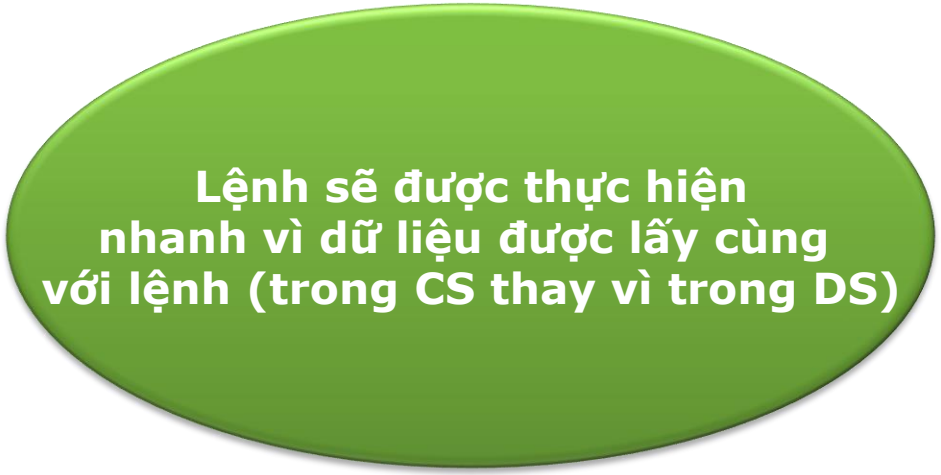
    ; không hợp lệ vì CS không được dùng làm thanh ghi đích
  - ADD AL, DL

    ; Cộng nội dung AL và DL rồi đưa vào AL



# 2.4.2. Chế độ địa chỉ tức thì

- Toán hạng đích là thanh ghi hoặc ô nhớ
  - Trừ thanh ghi đoạn và thanh cờ
- Toán hạng nguồn là hằng số (dữ liệu 8 bit hay 16 bit)
- Ví dụ:
  - MOV BL, 44
  - MOV AX, 44H
  - MOV AL, 'A'
    ; Copy mã ASCII của A vào thanh ghi AL
  - MOV DS, 0FF0H
    ; không hợp lệ
  - MOV AX, 0FF0H
  - MOV [BX], 10
    ; copy số thập phân 10 vào ô nhớ DS:BX



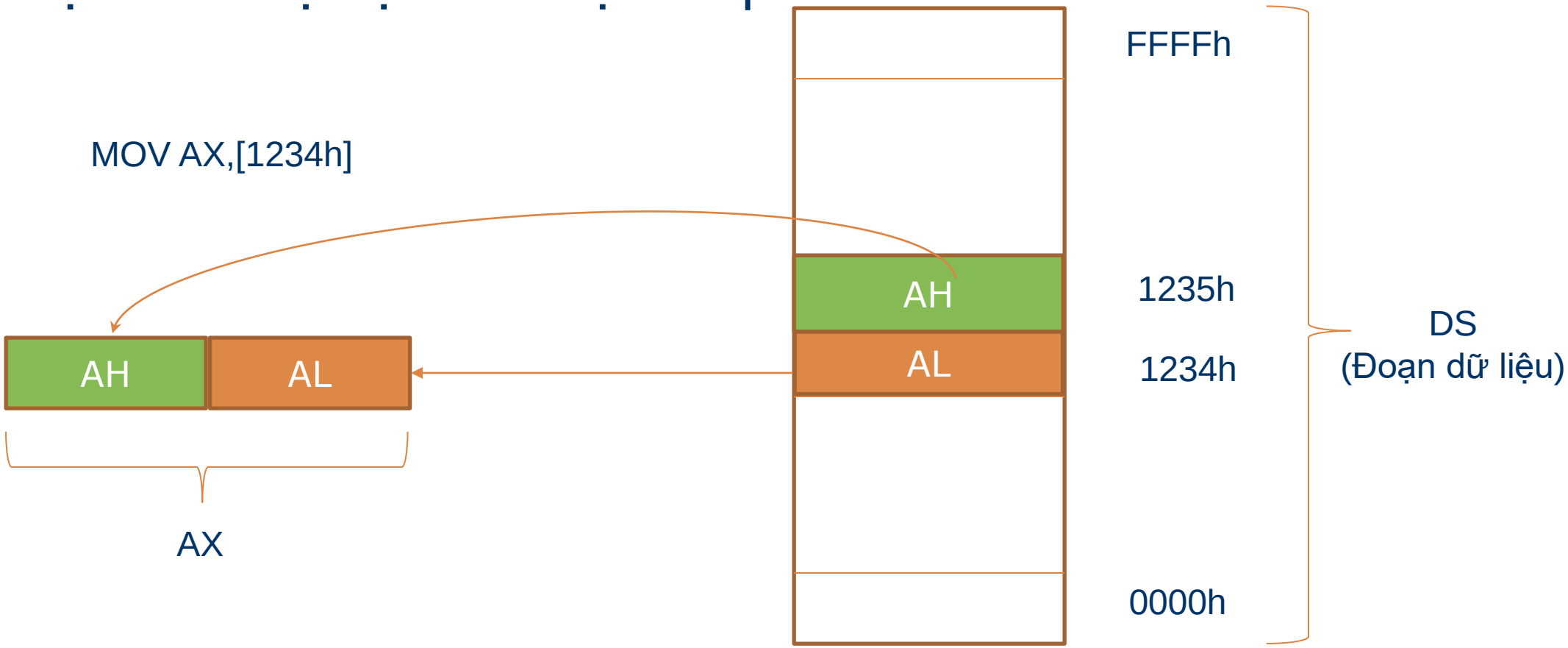
# 2.4.3. Chế độ địa chỉ trực tiếp qua thanh ghi

- *Register Direct Addressing Mode*
  - Một toán hạng là địa chỉ ô nhớ chứa dữ liệu, có:
    - Địa chỉ Offset: nằm trực tiếp trong câu lệnh
    - Địa chỉ segment: ngầm định chứa trong DS
  - Một toán hạng là thanh ghi
- Ví dụ:
  - MOV AL, [1234H]

    ; Copy nội dung ô nhớ có địa chỉ DS:1234H vào AL
  - MOV [ 4320H ], CX

    ; Copy nội dung của CX vào 2 ô nhớ liên tiếp DS:4320h và DS: 4321H

# Minh họa chế độ địa chỉ trực tiếp

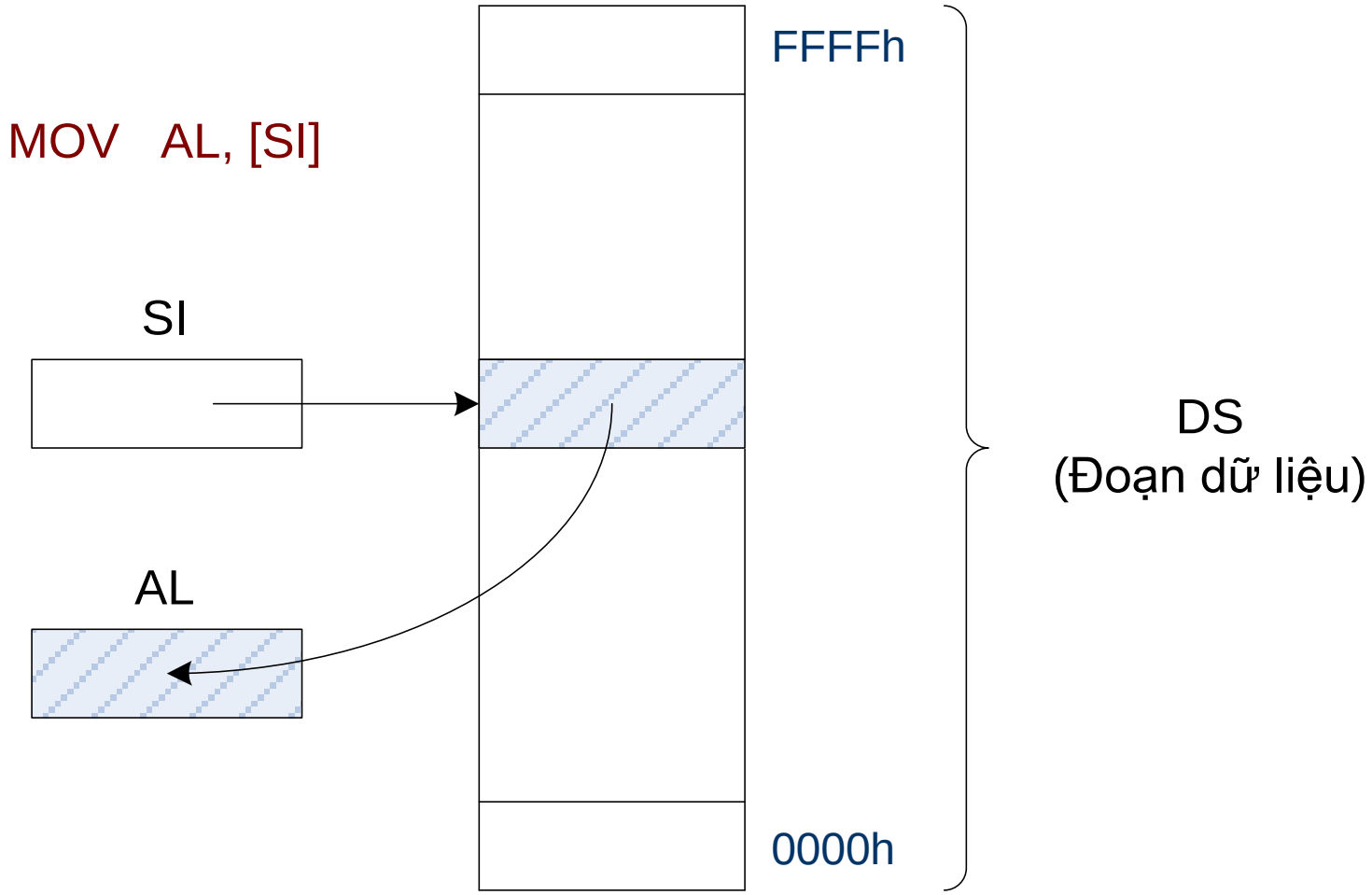


# 2.4.4. Chế độ địa chỉ gián tiếp qua thanh ghi

- Địa chỉ toán hạng không chứa trực tiếp trong lệnh mà gián tiếp thông qua một thanh ghi.
- Ô nhớ chứa nội dung của toán hạng này có:
  - Địa chỉ Offset: nằm trong các thanh ghi BX, BP, SI, DI.
  - Địa chỉ Segment ngầm định chứa trong:
    - DS nếu dùng BX,SI,DI
    - SS nếu dùng BP
- Ví dụ:
  - MOV AL, [BX] ; DS:BX → AL
  - MOV [ SI ], CL ; CL → DS:SI
  - MOV [ DI ], AX ; AX → DS: DI và DS:(DI +1)

# 2.4.4. Chế độ địa chỉ gián tiếp qua thanh ghi

## Minh họa chế độ địa chỉ gián tiếp



# 2.4.5. Chế độ địa chỉ tương đối cơ sở

- Một toán hạng là thanh ghi cơ sở BX, BP và các hằng số biểu diễn giá trị dịch chuyển (*displacement*), có:
  - Địa chỉ Offset của toán hạng được tính là tổng của nội dung thanh ghi BX hoặc BP và 1 độ dịch.
    - Độ dịch là 1 số nguyên âm hoặc dương.
  - Địa chỉ Segment là: DS hoặc SS.
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
  - MOV CX, [BX]+10

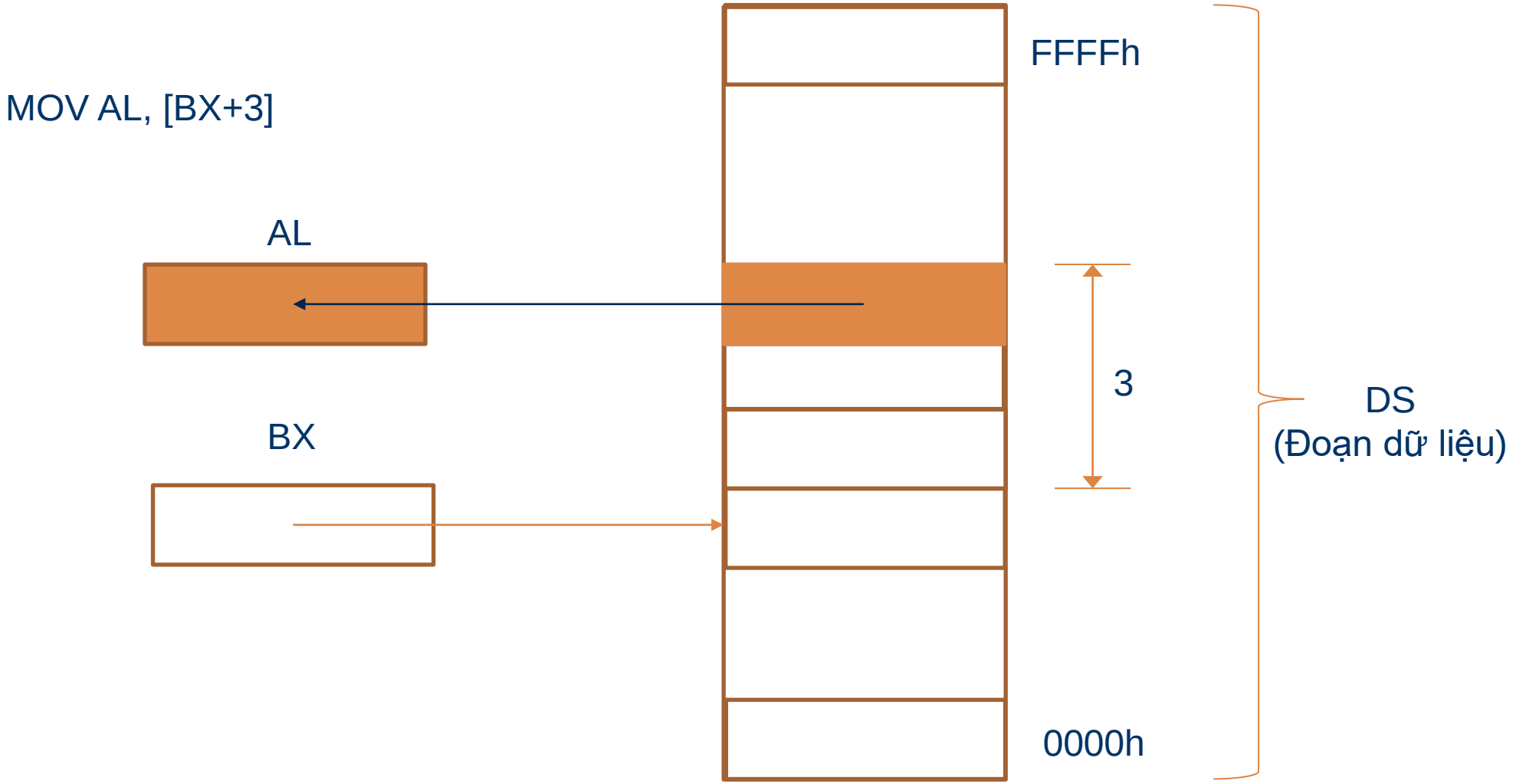
    *; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+10 và DS:BX+11 vào CX*
  - MOV CX, [BX+10]

    ; Cách viết khác của lệnh trên
  - MOV AL, [BP]+5

    *; copy nội dung của ô nhớ SS:BP+5 vào thanh ghi AL*

# 2.4.5. Chế độ địa chỉ tương đối cơ sở

- Minh họa chế độ địa chỉ tương đối cơ sở



# 2.4.6. Chế độ địa chỉ tương đối chỉ số

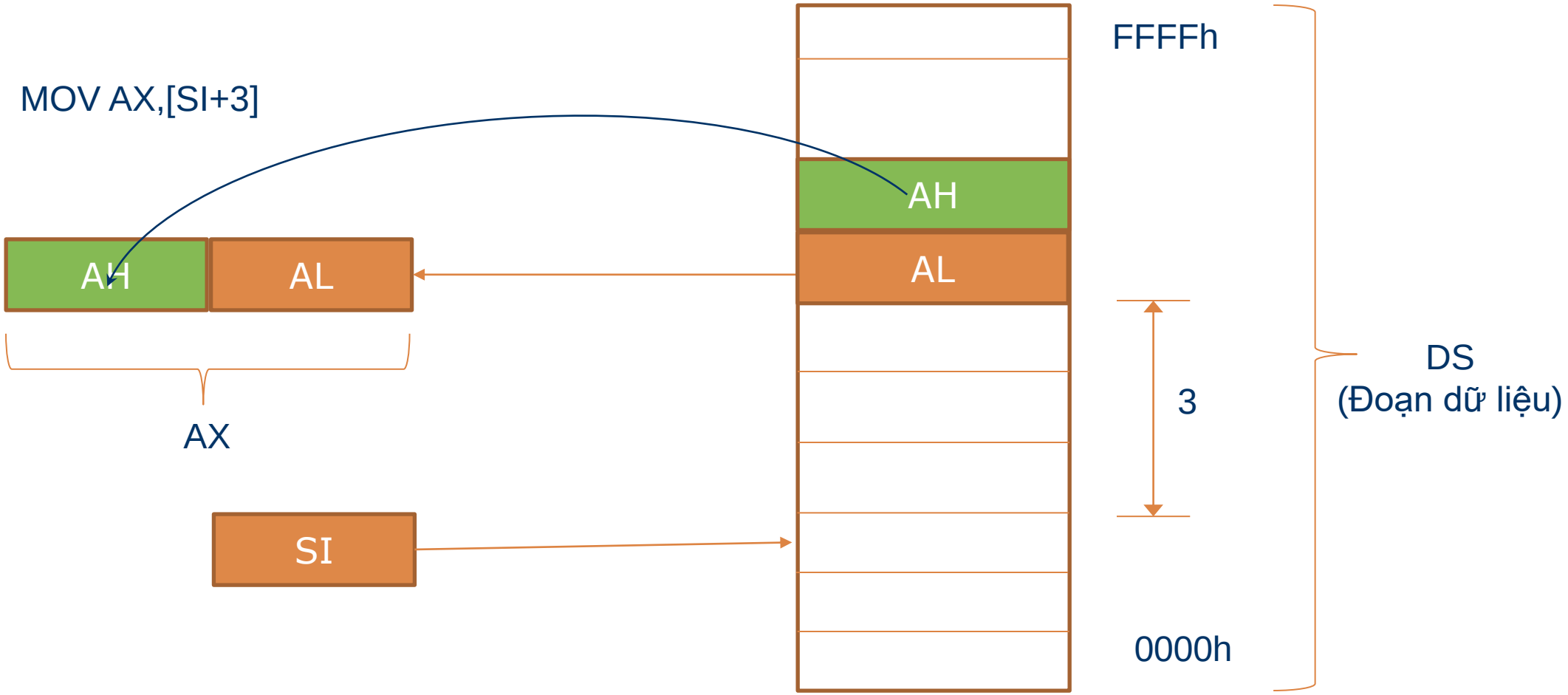
- Một toán hạng là thanh ghi chỉ số SI, DI và các hằng số biểu diễn giá trị dịch chuyển, có:
  - Địa chỉ Offset: tổng của nội dung thanh ghi SI hoặc DI và 1 độ dịch
    - Độ dịch ((*displacment*): 1 số nguyên âm hoặc dương
  - Địa chỉ Segment: DS
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
  - MOV AX, [SI]+10

    ; Copy nội dung 1 ô nhớ có địa chỉ DS:SI+10  AX
  - MOV AX, [SI+10]

    ; Cách viết khác của lệnh trên
  - MOV AL, [DI]+5

    ; copy nội dung của ô nhớ DS:DI+5 vào thanh ghi AL

# Minh họa chế độ địa chỉ tương đối chỉ số



# 2.4.7. Chế độ địa chỉ tương đối chỉ số cơ sở

- Dùng cả các thanh ghi chỉ số lẫn các thanh ghi cơ sở để tính địa chỉ của toán hạng:
  - Địa chỉ Offset: tổng của nội dung thanh ghi SI (hoặc DI) và nội dung thanh ghi BX (hoặc BP) và 1 độ dịch.
  - Địa chỉ Segment:
    - DS nếu sử dụng BX
    - SS nếu sử dụng BP

Thích hợp cho việc địa chỉ hoá các mảng 2 chiều

Thứ tự các phần tử trong mảng là tuỳ ý

- Ví dụ:
  - MOV AX, [BX] [SI]+8

    ;Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+SI+8 và ;DS:BX+SI+9 vào AX
  - MOV AX, [BX+SI+8]

    ; Cách viết khác của lệnh trên
  - MOV CL, [BP+DI+5]

    ; copy nội dung của ô nhớ SS:BP+DI+5 vào thanh ghi CL

# Tóm tắt các chế độ địa chỉ

| Chế độ địa chỉ | Toán hạng | Thanh ghi đoạn ngầm định |
|---|---|---|
| Thanh ghi | Thanh ghi | |
| Tức thì | Dữ liệu | |
| Trực tiếp qua thanh ghi | [offset] | DS |
| Gián tiếp qua thanh ghi | [BX]<br>[SI]<br>[DI] | DS<br>DS<br>DS |
| Tương đối cơ sở | [BX] + dịch chuyển<br>[BP] + dịch chuyển | DS<br>SS |
| Tương đối chỉ số | [DI] + dịch chuyển<br>[SI] + dịch chuyển | DS<br>DS |
| Tương đối chỉ số cơ sở | [BX] + [DI]+ dịch chuyển<br>[BX] + [SI]+ dịch chuyển<br>[BP] + [DI]+ dịch chuyển<br>[BP] + [SI]+ dịch chuyển | DS<br>DS<br>SS<br>SS |

# Các cặp thanh ghi đoạn:lệch ngầm định

| Thanh ghi đoạn | CS | DS | ES | SS |
|---|---|---|---|---|
| Thanh ghi lệch | IP | BX, SI, DI | DI | SP, BP |

*Ghi chú: các cặp DS:SI và ES:DI dùng với các lệnh thao tác chuỗi*

# Chương 2: Bộ vi xử lý Intel 8088/8086

- 2.1. Cấu trúc của bộ vi xử lý Intel 8088
- 2.2. Các thanh ghi của bộ vi xử lý Intel 80286
- 2.3. Nguyên tắc làm việc của bộ vi xử lý Intel 8088
- 2.4. Các chế độ địa chỉ của bộ vi xử lý 8088
- 2.5. Các bộ vi xử lý thế hệ sau 8088 của Intel

# 2.5. Các bộ vi xử lý thế hệ sau 8088 của Intel

- SV tự tìm hiểu

# Thank You!