HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KIẾN TRÚC MÁY TÍNH
Chương 4

TS. Nguyễn Đức Minh
minhnd@ptit.edu.vn

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

4.2. Mã hóa lệnh và các chế độ địa chỉ

4.3 Tập lệnh và công cụ EMU8086.

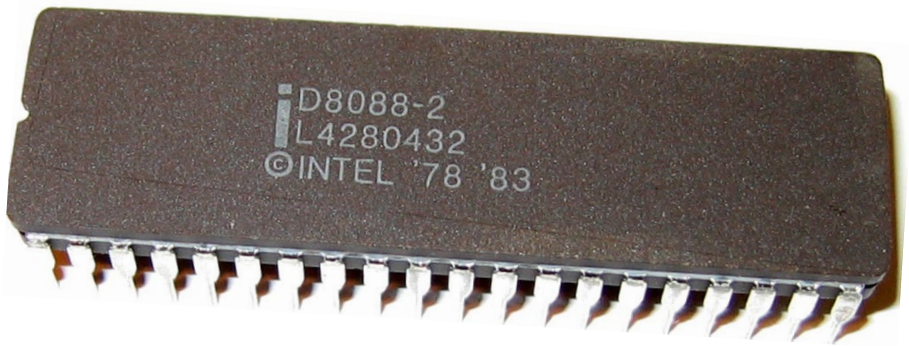
4.4 Lập trình hợp ngữ

4.5 Một số ví dụ

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

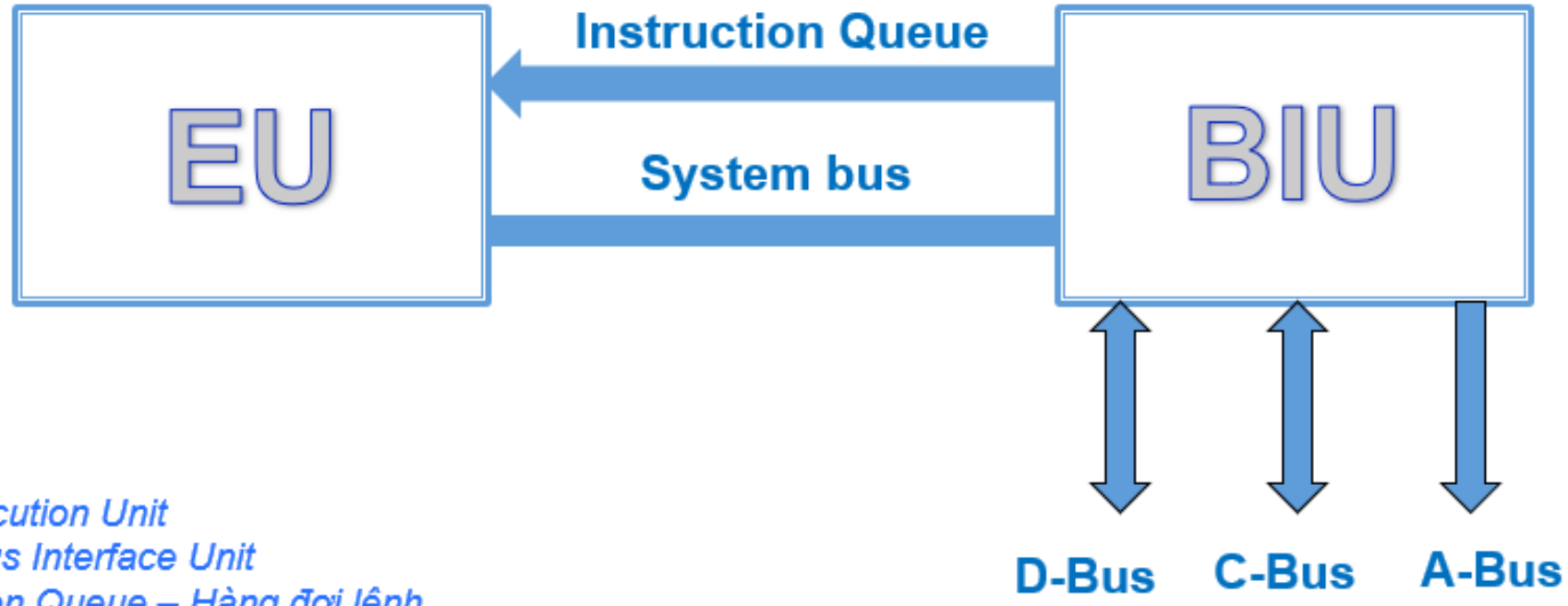
4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

Bus dữ liệu 16 bit
Bus địa chỉ 20 bit



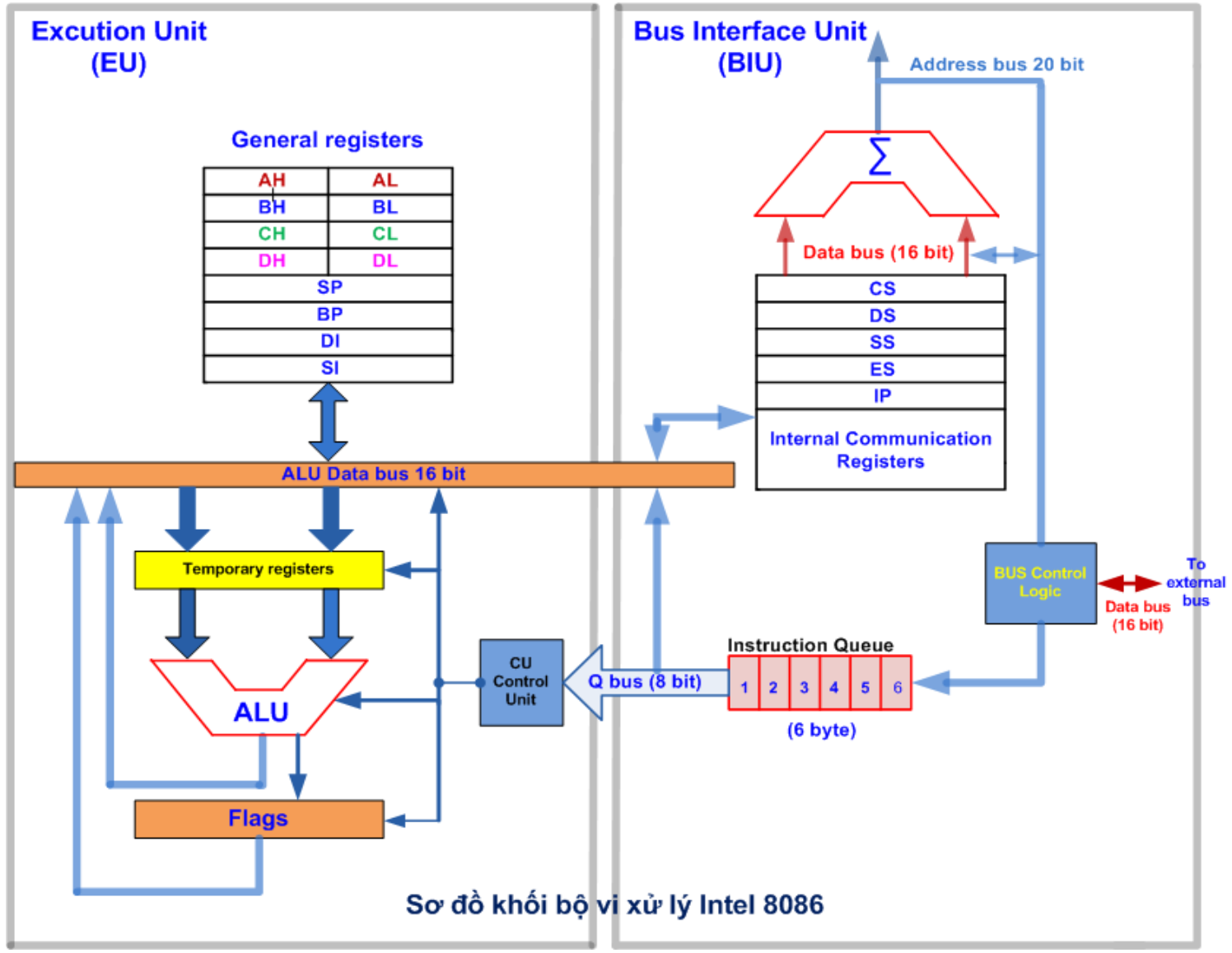
Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088



EU – Excution Unit
BIU – Bus Interface Unit
Instruction Queue – Hàng đợi lệnh
System bus – Bus hệ thống
D bus - Data bus
A bus - Address bus
C bus – Control bus

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088



Sơ đồ khối bộ vi xử lý Intel 8086

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

Chức năng và hoạt động của các khối EU và BIU

Bộ vi xử lý 8086 về mặt chức năng được chia ra làm hai khối chính:

- Khối giao tiếp bus BIU (Bus Interface Unit)
- Khối thực hiện lệnh EU (Excution Unit)

BIU

- ✓ Đưa ra địa chỉ, đọc mã lệnh từ bộ nhớ, đọc ghi dữ liệu từ các cổng vào ra hoặc bộ nhớ…..Chịu trách nhiệm đưa địa chỉ ra bus và trao đổi dữ liệu với bus.
- ✓ Trong BIU có hàng đợi lệnh (Instruction Queue) dung lượng 6 byte dùng để chứa các mã lệnh đọc được, nằm đợi EU xử lý.
- ✓ Liên quan đến cơ chế xử lý xen kẽ liên tục dòng mã lệnh (pipelining).

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

Chức năng và hoạt động của các khối EU và BIU

EU

- ✓ Bao gồm CU (Control Unit), bên trong CU có mạch giải mã lệnh để thi hành lệnh.
- ✓ Mã lệnh đọc vào từ bộ nhớ được đưa đến đầu vào của bộ giải mã, các thông tin thu được từ đầu ra của nó sẽ được đưa đến mạch tạo xung điều khiển….kết quả thu được các dãy xung khác nhau điều khiển các hoạt động của các bộ phận bên trong và bên ngoài CPU.
- ✓ Khối số học và logic (ALU) dùng để thực hiện các thao tác khác nhau với các toán hạng của lệnh.

EU cung cấp thông tin về địa chỉ cho BIU để BIU đọc lệnh và dữ liệu. EU giải mã và thi hành lệnh

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

Thành phần điều khiển bus (bus control logic)

- Điều khiển các loại tín hiệu trên các bus bao gồm: các tín hiệu trên bus địa chỉ, các tín hiệu trên bus dữ liệu và các tín hiệu trên bus điều khiển.
- Hỗ trợ giao tiếp giữa hệ thống bus trong và bus ngoài (hệ thống bus kết nối các thành phần của hệ vi xử lý với nhau: bộ nhớ, I/O...).

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

Hàng đợi lệnh (prefetch queue)

- Đây là một vùng không gian nhớ nhỏ nằm giữa BIU và EU, làm nhiệm vụ chứa mã lệnh chờ được xử lý và phối hợp hoạt động của BIU và EU theo cơ chế pipeline.
- Hàng đợi lệnh của 8086 là **6** byte
- Hàng đợi lệnh của 8088 là **4** byte
- Làm việc theo cơ chế FIFO (First In First Out – vào trước ra trước).

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

Khối điều khiển (Control Unit)

- Có hai chức năng chính: giải mã lệnh và tạo xung điều khiển.
- Đầu vào của khối điều khiển là mã lệnh đọc từ hàng đợi lệnh, đầu ra là các xung điều khiển gửi đến các bộ phận khác nhau bên trong vi xử lý.
- Thực hiện nhờ hai mạch: giải mã lệnh và mạch tạo xung.

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

Khối số học và logic ALU (Arithmatic and Logic Unit)

- Thực hiện các phép tính toán: cộng, trừ…các phép toán logic: AND, OR…
- Đầu vào là hai thanh ghi tạm, đầu ra đưa vào bus dữ liệu và phản ánh vào thanh ghi cờ.

Mô hình lập trình của bộ vi xử lý 8086

Mô hình lập trình của bộ vi xử lý 8086 là các thanh ghi nhìn thấy được của bộ vi xử lý mà người lập trình có thể sử dụng để lập trình ứng dụng điều khiển.

- Bao gồm các thanh ghi **AX, BX, CX, DX, CS, DS, ES, SS, IP, BP, SP, SI, DI, FR**

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.1 Kiến trúc và thành phần của bộ vi xử lý 8086/8088

8088 và 8086 có 2 điểm khác nhau cơ bản

- Thứ nhất: Kích thước bus dữ liệu (data bus): ở 8086 là 16 bit còn 8088 là 8 bit. Do vậy 8086 có thể đọc dữ liệu từ 2 ô nhớ thẳng hàng (địa chỉ chẵn ở byte thấp, lẻ ở byte cao) trong 1 chu kỳ ghi/đọc. 8088 mất hai chu kỳ ghi đọc để đọc dữ liệu ở hai ô nhớ thẳng hàng. → 8086 có hiệu năng cao hơn 8088
- Thứ hai: độ dài hàng đợi lệnh (instruction queue): ở 8088 là 4 byte còn 8086 là 6 byte → hiệu năng 8086 cao hơn 8088.

Cấu trúc và tập lệnh của 8086 và 8088 là giống nhau về cơ bản. Tất cả các chương trình chạy trên 8088 đều chạy trên 8086 và ngược lại.

Về quan điểm lập trình thì hai bộ vi xử lý này là tương đương !

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các thanh ghi (registers)

a. Các thanh ghi đoạn (segment registers)

8086 có 4 thanh ghi đoạn (segment regisster).

Là các thanh ghi 16 bit

CS (Code Segment)

DS (Data Segment)

SS (Stack Segment)

ES (Extra Segment)

Thanh ghi đoạn mã

Thanh ghi đoạn dữ liệu

Thanh ghi đoạn Ngăn xếp

Thanh ghi đoạn dữ liệu mở rộng

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

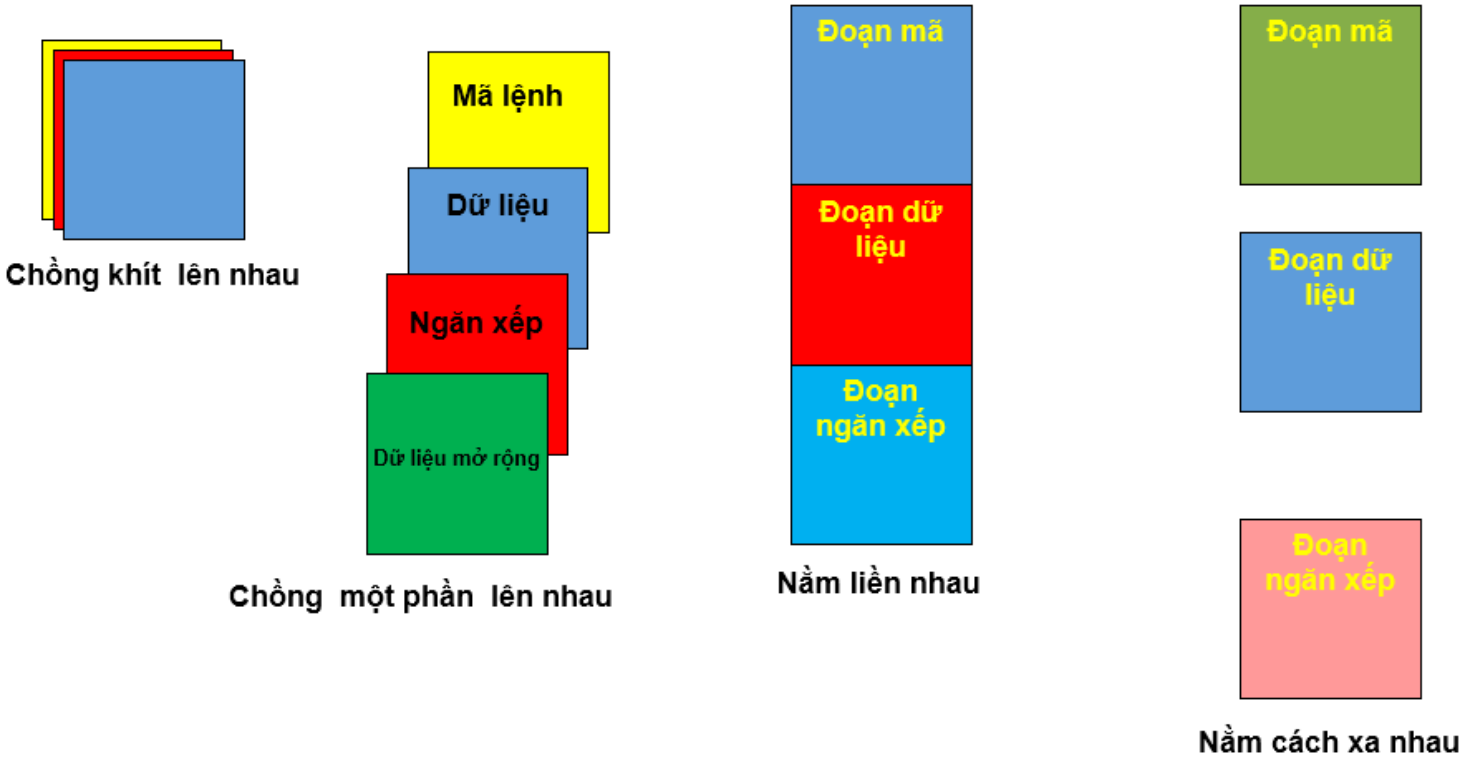
- Vi xử lý 8086 có bus địa chỉ 20 bit nên có khả năng phân biệt được 2^{20} vị trí nhớ (thực tế là các byte nhớ).

- Mỗi vị trí nhớ trong không gian nhớ của 8086 là 1 byte.

- 2^{20} byte = 1.048.576 byte = 1MByte → không gian nhớ mà 8086 quản lý được là 1 MB.
- Không gian nhớ của 8086 được chia một cách logic thành các vùng nhớ (đoạn bộ nhớ) khác nhau gọi là các segment. Mỗi đoạn có kích thước 64 Kbyte.
    - Các đoạn này được dùng để chứa:
        - Mã lệnh của chương trình (code segment)
        - Dữ liệu và kết quả tạm thời của chương trình…
        - Tạo ra một vùng nhớ đặc biệt được gọi là ngăn xếp (stack) làm việc theo cơ chế LIFO (first-in-last-out).

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- Có 16 đoạn 64 KB trong không gian nhớ 1 MB, các đoạn này có thể nằm sát liền nhau, nằm cách xa nhau, nằm chồng lên nhau 1 phần, nằm chồng khít lên nhau….và có thể di chuyển linh hoạt trong không gian nhớ 1 MB.
- Vi xử lý 8086 có thể truy cập vào bất kỳ một vùng nhớ nào nằm trong không gian nhớ 1MB.



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- Một thanh ghi của 8086 chỉ có kích thước 16bit (nếu dùng để định địa chỉ thì chỉ có thể xác định được 2^{16} vị trí nhớ = 64KB bộ nhớ) nên không thể dùng 1 thanh ghi để quản lý toàn bộ không gian nhớ 1MB.

- Người ta phải sử dụng 1 cặp hai thanh ghi 16 bit (1 thanh ghi đoạn và 1 thanh ghi đa năng) kết hợp với nhau để phục vụ việc định địa chỉ cho toàn bộ không gian nhớ 1MB → **Phương pháp định địa chỉ segment: offset** (phương pháp định địa chỉ đoạn: độ lệch)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Nội dung phương pháp

- Để xác định vị trí một byte nhớ trong không gian nhớ 1 MB, đầu tiên cần biết byte đó nằm thuộc đoạn 64KB nào (trong tổng số 16 đoạn).
- Địa chỉ đoạn của byte đó sẽ nằm trong 1 trong 4 thanh ghi đoạn (CS, DS, SS, ES).
- *Giá trị nội dung của thanh ghi đoạn sẽ chính là địa chỉ byte đầu tiên của đoạn* (được gọi là địa chỉ cơ sở đoạn). Đây chính là địa chỉ đoạn của byte nhớ cần tìm (địa chỉ segment).
- *Khoảng cách từ byte đầu tiên của đoạn tới vị trí byte nhớ cần tìm được gọi là độ lệch* (địa chỉ offset).

  - Địa chỉ offset thường được biểu diễn bởi các thanh ghi công dụng chung, các thanh ghi chỉ số hay con trỏ…

- Lúc này địa chỉ của byte nhớ cần tìm được biểu diễn bởi nội dung của hai thanh ghi : thanh ghi đoạn và thanh ghi độ lệch.

- Biểu diễn: **Segment: Offset**

Vi dụ: CS:IP, DS:BX… 5FFFh:6789h

- Địa chỉ này là địa chỉ logic, được dùng một cách rất linh hoạt cho người lập trình. Nó có thể được chuyển thành địa chỉ vật lý (physical address- là địa chỉ mà CPU đặt lên bus địa chỉ) theo công thức sau:

Địa chỉ vật lý = Segment * 16 + Offset

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- **Ví dụ**: Chuyển địa chỉ logic 500Ah:10B9h thành địa chỉ vật lý

500Ah * 16 + 10B9h = 500A0h + 10B9h = 51159h

Vậy địa chỉ vật lý tương ứng là 51159h.

- Cách chuyển đổi địa chỉ logic thành địa chỉ vật lý nhanh hơn (khi biểu diễn các giá trị dưới dạng hexa) là : thêm 1 số 0 vào bên phải giá trị segment và cộng với giá trị offset (do nhân với 16).

- Ví dụ: Chuyển đổi địa chỉ 1234h:5678h thành địa chỉ vật lý ?

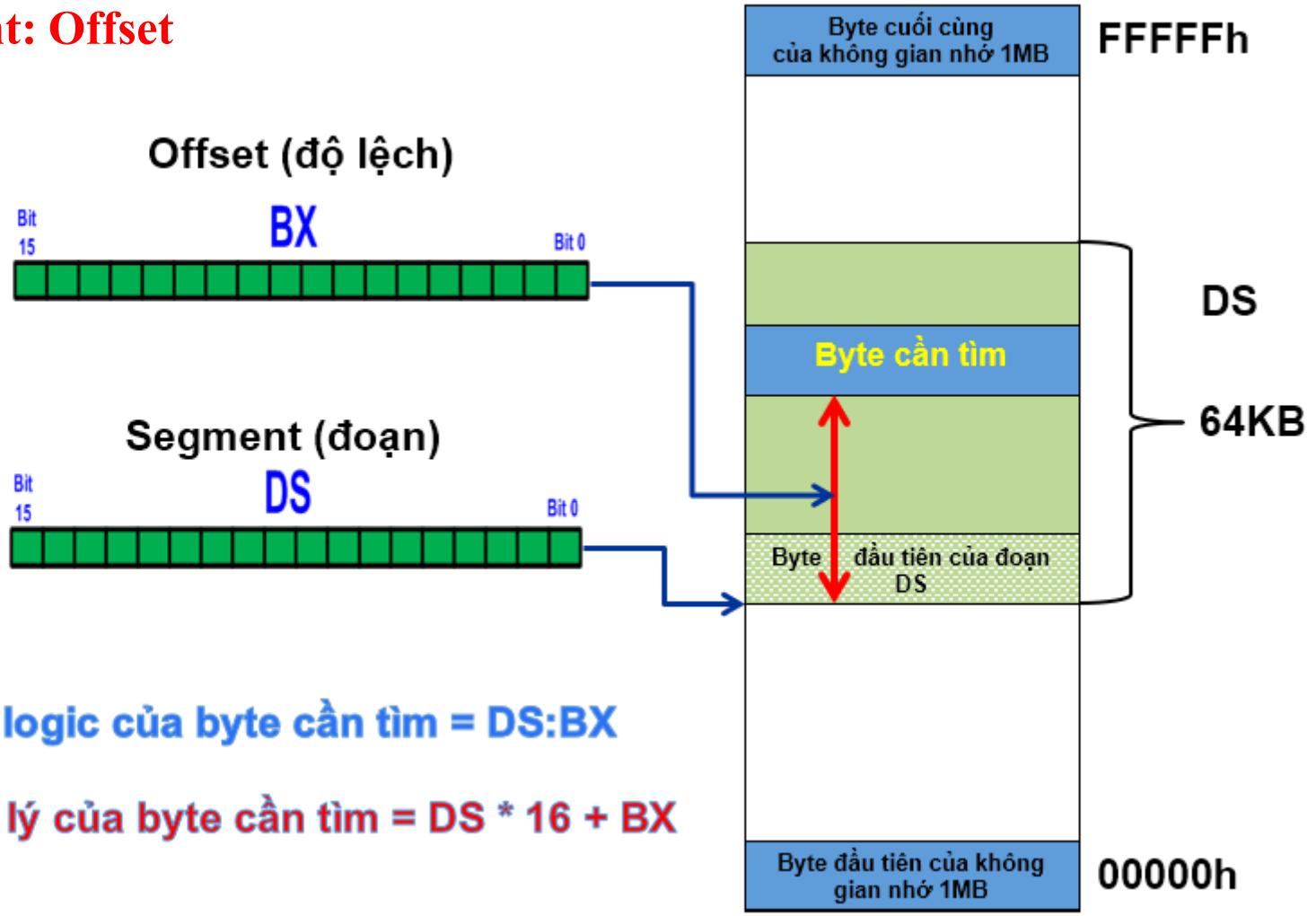
- Ví dụ: Chuyển đổi địa chỉ 2345h:6789h thành địa chỉ vật lý ?

Phương pháp định địa chỉ segment:offset *có tính chất đa trị* do 1 giá trị địa chỉ vật lý có thể được biểu diễn bởi nhiều địa chỉ logic.

(không có ánh xạ 1:1 giữa hai kiểu địa chỉ)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Phương pháp định địa chỉ Segment: Offset

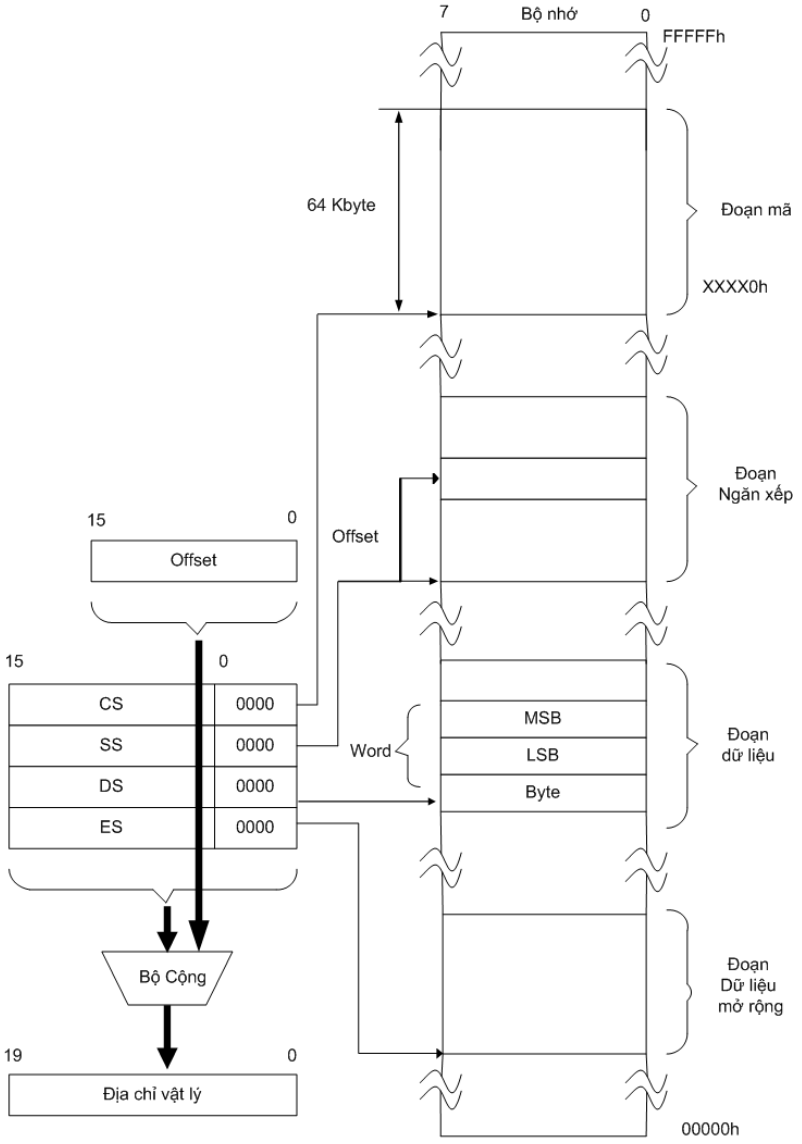


Địa chỉ logic của byte cần tìm = DS:BX

Địa chỉ vật lý của byte cần tìm = DS * 16 + BX

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Phương pháp định địa chỉ Segment: Offset



Phân đoạn bộ nhớ của 8086/8088

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

b. Các thanh ghi đa năng (general purpose registers)

8086 có **4** thanh ghi công dụng chung: đều là các thanh ghi 16 bit

AX (**A**ccumulator- thanh ghi tích lũy): chứa kết quả của các thao tác số học

BX (**B**ase – thanh ghi cơ sở) : chứa địa chỉ cơ sở của bảng dùng trong lệnh XLAT

CX (**C**ount – thanh ghi đếm): chứa số lần lặp trong lệnh LOOP

DX (**D**ata – thanh ghi dữ liệu): tham gia các thao tác nhân, chia số 16 bit, chứa địa chỉ cổng trong các lệnh vào ra dữ liệu trực tiếp (in, out)

Cả 4 thanh ghi AX, BX, CX, DX đều có thể tách (logic) thành hai thanh ghi 8 bit

AX = AH (byte cao của AX) và AL (byte thấp của AX)

BX = BH (byte cao của BX) và BL (byte thấp của BX)

CX = CH (byte cao của CX) và CL (byte thấp của CX)

DX = DH (byte cao của DX) và DL (byte thấp của DX)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

c. Các thanh ghi con trỏ (pointer) và chỉ số (index)

8086 có **3** thanh con trỏ và **2** thanh ghi chỉ số đều là các thanh ghi 16 bit

IP (Instruction pointer- thanh ghi con trỏ lệnh): luôn trỏ vào lệnh tiếp theo sẽ được thực thi nằm trong đoạn mã CS. Địa chỉ đầy đủ của lệnh này là CS:IP

BP (Base pointer – thanh ghi con trỏ cơ sở): luôn trỏ vào một dữ liệu nằm trong đoạn ngăn xếp SS. Địa chỉ đầy đủ của một phần tử trong đoạn ngăn xếp ứng với SS:BP

SP (Stack pointer – thanh ghi con trỏ ngăn xếp): luôn trỏ vào đỉnh hiện thời của ngăn xếp SS. Địa chỉ đầy đủ của đỉnh ngăn xếp là SS:SP

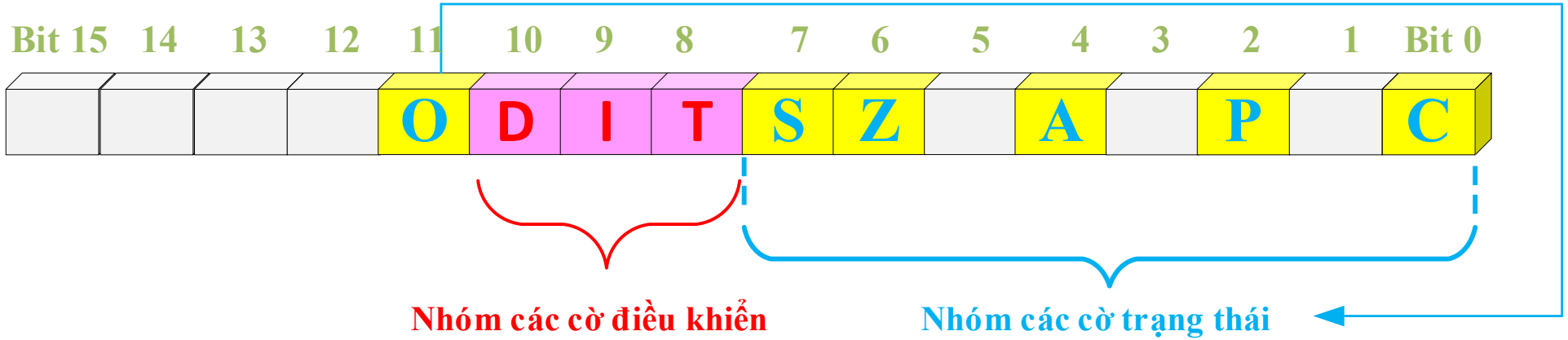
SI (Source Index – thanh ghi chỉ số gốc hay chỉ số nguồn): SI chỉ vào dữ liệu trong đoạn dữ liệu DS mà địa chỉ cụ thể ứng với DS:SI

DI (Destination index – thanh ghi chỉ số đích). DI chỉ vào dữ liệu nằm trong đoạn DS mà địa chỉ cụ thể ứng với DS:DI

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

d. Thanh ghi cờ (flag register)

- Là một thanh ghi **16 bit**, được gọi là thanh ghi cờ vì mỗi bít của nó ***báo hiệu*** trạng thái hoạt động của đơn vị số học & logic (**ALU**) và ***điều khiển*** một số thao tác của ***EU***.
- Dựa vào các cờ này người lập trình có thể có các lệnh thích hợp cho vi xử lý (ex: các lệnh nhảy có điều kiện, lệnh cộng..vv)
- Các bit của thanh ghi được gọi là các cờ (0 = xóa cờ, 1= lập cờ). Có **9** trong 16 bit của thanh ghi được sử dụng làm các cờ



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Nhóm các bit **cờ trạng thái** (status flag)

6 cờ trạng thái phản ánh các trạng thái khác nhau của kết quả sau mỗi một thao tác tính toán

Cờ C (CF-carry flag): Cờ nhớ

CF=1 khi có nhớ hoặc mượn từ MSB

CF=0 khi không có nhớ hoặc mượn từ MSB

Cờ P (PF-Parity Flag): Cờ chẵn lẻ. PF phản ánh tính chẵn lẻ của tổng số bit 1 có trong kết quả.

PF=1 khi tổng số bit 1 trong kết quả là chẵn (PE-parity even).

PF=0 khi tổng số bit 1 trong kết quả là lẻ (PO-Parity odd).

Cờ A (Auxiliary carry flag): cờ nhớ phụ.

AF= 1 khi có nhớ từ nibble thấp sang nibble cao (rất có tác dụng khi làm việc với các số BCD).

AF = 0 khi không có nhớ từ nible thấp sang nibble cao.

Cờ Z (Zero flag): cờ không hay cờ rỗng.

ZF = 1 khi kết quả phép toán bằng 0

ZF = 0 khi kết quả phép toán khác không

Cờ S (SF- sign flag): Cờ dấu

SF=1 khi kết quả phép toán là âm

SF=0 khi kết quả phép toán là dương

Cờ O (OF- Overflow Flag): Cờ tràn

OF= 1 khi kết quả phép toán là 1 số bù 2 vượt ra ngoài giới hạn biểu diễn dành cho nó.

OF= 0 khi kết quả phép toán không vượt ra ngoài giới hạn biểu diễn của nó.

Nhóm các bit cờ **điều khiển** (control flag)

3 cờ điều khiển dùng để điều khiển các hoạt động của bộ vi xử lý.

Có thể lập, xóa bằng các lệnh của người dùng

Cờ T (TF-Trap flag): Cờ bẫy

TF=1 CPU sẽ làm việc ở chế độ chạy từng lệnh

TF=0 CPU hoạt động bình thường

Cờ I (IF-Interrupt enable flag): Cờ cho phép ngắt

IF=1 CPU cho phép các yêu cầu ngắt (che được) được phép tác động.

IF=0 CPU cấm các ngắt (che được) không được phép tác động.

Cờ D (Direction flag): cờ hướng

DF= 1 CPU làm việc với chuỗi ký tự theo thứ tự từ phải sang trái (cờ lùi)

DF = 0 khi CPU làm việc với chuỗi ký tự theo thứ tự từ trái sang phải.

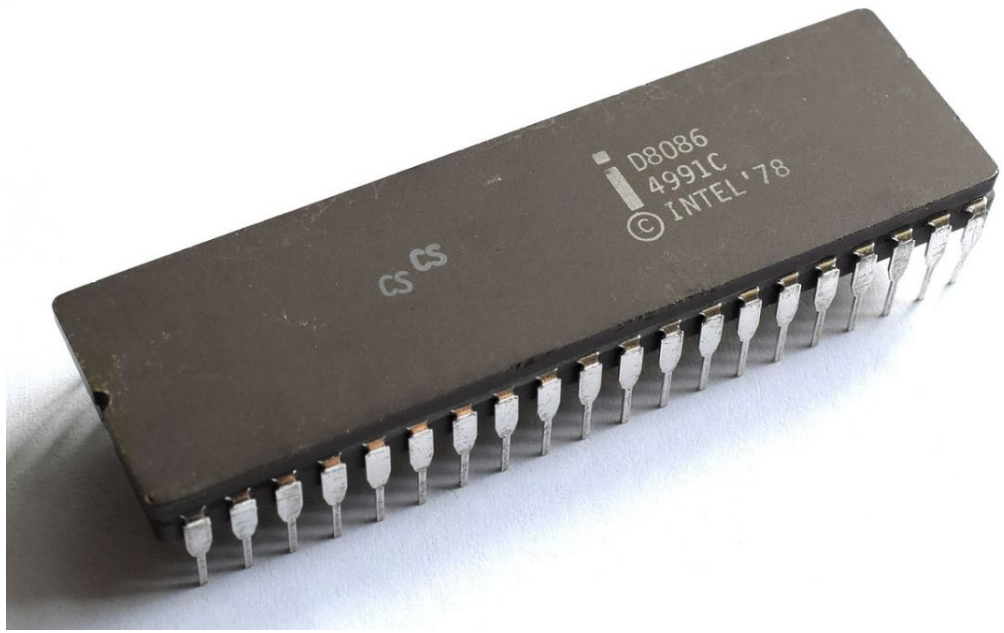
Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

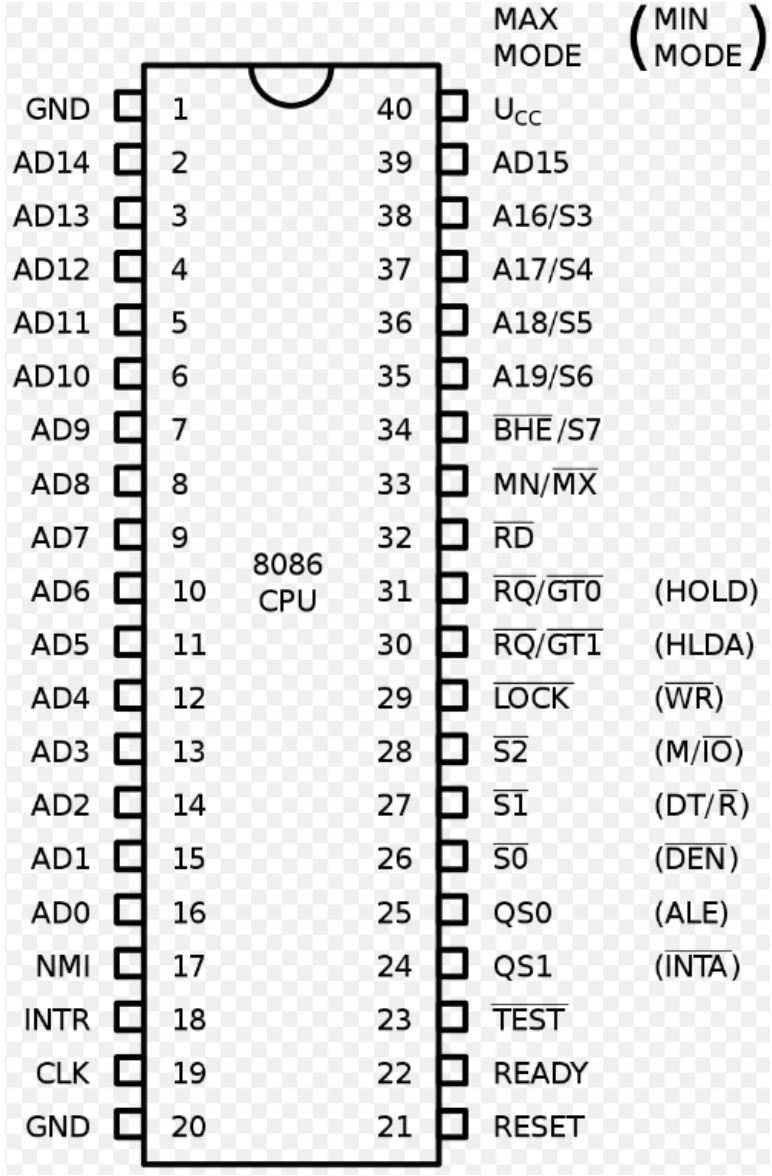
e. Một số thanh ghi khác

- Ngoài các thanh ghi trên, CPU còn có: các thanh ghi đệm đầu vào của bộ số học logic, các thanh ghi truyền thông nội bộ.

- Các thanh ghi này phục vụ hoạt động nội bộ của bộ vi xử lý, người dùng *không thể tương tác* với chúng.
- Được gọi là các thanh ghi không nhìn thấy được (invisible register)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088





Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.2 Mã hóa lệnh và các chế độ định địa chỉ

- Tập lệnh của vi xử lý là các lệnh được mã hóa bằng các **mã nhị phân** để phần mức logic số của vi xử lý có thể hiểu và thực hiện được các tác vụ.
- Tập lệnh hợp ngữ của vi xử lý là các lệnh biểu diễn bằng các ký tự dưới dạng gợi nhớ (**mnemonic**) để có thể dễ dàng sử dụng
- Có ánh xạ gần như 1-1 với tập lệnh hợp ngữ của bộ vi xử lý.
- Bao gồm **125 lệnh** và được chia thành các nhóm lệnh: lệnh di chuyển dữ liệu, lệnh lặp, lệnh chuyển điều khiển..vv

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.2 Mã hóa lệnh và các chế độ định địa chỉ

- Khuôn dạng lệnh

Lệnh của vi xử lý là một từ nhị phân có dạng chung gồm hai thành phần

- Mã lệnh (opcode)
- Địa chỉ toán hạng

- 8086 xử lý từ 16 bit. Có dạng lệnh dài từ 1 byte đến 6 byte

Opcode	Operands
Mã lệnh	Các toán hạng

Mã lệnh	Đích, Gốc
MOV	AX, 100

AX ← 100

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.2 Mã hóa lệnh và các chế độ định địa chỉ

Mã hóa lệnh

Lệnh của 8086 được mã hóa dưới dạng các từ nhị phân. Các trường và ý nghĩa được cho như hình

Byte 1								Byte 2								Byte 3	Byte 4	Byte 5	Byte 6
																Độ dịch/dữ liệu thấp	Độ dịch/dữ liệu cao	Dữ liệu thấp	Dữ liệu cao
Op-code						d	w	Mod		Reg*			R/M						

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Byte 1 gồm:

- **Opcode – Mã lệnh** (gồm 6 bit): mã lệnh xác định tác vụ cần thực hiện:

  Vì dụ: Lệnh MOV → mã lệnh 10010

- **Bit D (direction) - bit hướng**, chỉ hướng vận chuyển dữ liệu;

  D=1: dữ liệu đi đến thanh ghi cho bởi 3 bit REG

  D=0: dữ liệu đi ra từ thanh ghi cho bởi 3 bit REG

  D = 1→ đích ; D = 0 → nguồn;

- **Bit W (wide) xác định kích cỡ của toán hạng** (độ rộng toán hạng) là 1 byte hay 2 byte;

  W = 0 → 1 byte **W** = 1 → 2 byte

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Byte 2 gồm: trường chế độ MOD, trường thanh ghi REG và R/M

Byte 1			Byte 2			Byte 3	Byte 4	Byte 5	Byte 6
Op-Code	D	W	MOD	REG*	R/M	Độ dịch/ địa chỉ trực tiếp phần thấp	Độ dịch/ địa chỉ trực tiếp phần cao	Dữ liệu phần thấp	Dữ liệu phần cao

MOD (2 bit) và R/M (3 bít): MOD và R/M kết hợp với nhau để biểu diễn các chế độ địa chỉ của vi xử lý Intel 8086/8088.

- Trường MOD xác định chế độ định địa chỉ

00 → Độ dịch chuyển bằng 0 (no displacement)

01 → Độ dịch chuyển 8 bit

10 → Độ dịch chuyển 16 bit

11 → R/M là một thanh ghi (chế độ *định địa chỉ thanh ghi*)

Nếu MOD là 00, 01 hoặc 10, trường R/M chọn một trong các chế độ *định địa chỉ bộ nhớ*

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- **R/M** : toán hạng thanh ghi/các thanh ghi dùng để tính địa chỉ hiệu dụng (EA- Effective Address)
- **REG**: trường toán hạng thanh ghi/mở rộng cho mã lệnh.

Thanh ghi		Mã (tương ứng với 3bit REG)
W = 1	W = 0	
AX	AL	000
BX	BL	011
CX	CL	001
DX	DL	010
SP	AH	100
DI	BH	111
BP	CH	101
SI	DH	110

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

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

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Ví dụ: Lệnh 8A15h → 1000 1010 0001 0101b

Mã nhị phân: 100010 1 0 00 010 101

Mã lệnh 100010 → lệnh MOV

D = 1, các luồng dữ liệu từ R/M tới REG

W = 0 → 8 bit argument

MOD = 00 → không có độ dịch chuyển (no displacement)

REG = 010 (DL)

R/M = 101 ([DI] chế độ định địa chỉ)

Dạng lệnh cuối cùng là → **MOV DL, [DI]**

Ví dụ1: Mã hoá lệnh MOV CL, [SI] ?

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các chế độ định địa chỉ (Addressing modes) là phương thức CPU tổ chức các toán hạng của lệnh hay nói cách khác *Chế độ định địa chỉ là phương thức CPU tìm thấy các toán hạng của lệnh.*

- Chế độ định địa chỉ cũng cho phép CPU kiểm tra dạng của toán hạng

Vi xử lý Intel 8086/8088 có 7 chế độ định địa chỉ:

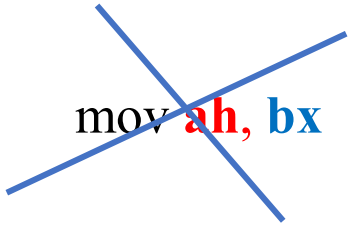
- Chế độ định địa chỉ **Thanh ghi** (Registers)
- Chế độ định địa chỉ **Tức thì** (Immediate)
- Chế độ định địa chỉ **Trực tiếp** (Direct)
- Chế độ định địa chỉ **Gián tiếp qua thanh ghi** (Registers Indirect)
- Chế độ định địa chỉ **Gián tiếp qua ô nhớ** (Memory Indirect)
- Chế độ định địa chỉ **Chỉ số** (Index)
- Chế độ định địa chỉ **Tương đối** (Relative)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Thanh ghi (Register Addressing Mode)

- Sử dụng các thanh ghi bên trong CPU như là các toán hạng để chứa dữ liệu cần thao tác.
- Cả toán hạng nguồn và đích đều là các thanh ghi
- Các thanh ghi được sử dụng trong 1 lệnh phải cùng kích thước độ lớn
- VD: lệnh hợp ngữ

mov bx, dx;	$bx \leftarrow dx$
mov ds, ax;	$ds \leftarrow ax$
add al, dl;	$al \leftarrow al + dl$



→ sai !

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Thanh ghi (Register Addressing Mode)

Assembly	Kích thước	Hoạt động
MOV AL,BL	8 bit	Sao chép BL vào AL
MOV CH,CL	8 bit	Sao chép CL vào CH
MOV AX,CX	16 bit	Sao chép CX vào AX
MOV DS,AX	16 bit	Sao chép AX vào DS
MOV BX,ES	16 bit	Sao chép ES vào BX
MOV ES,DS	-	Không được phép dùng hai thanh ghi đoạn
MOV BL,DX	-	Không được phép, hai thanh ghi khác kích cỡ

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tức thì (Immediate Addressing Mode)

- Toán hạng đích là một thanh ghi hay một ô nhớ
- Toán hạng nguồn là một hằng số
- Giá trị hằng của toán hạng nguồn (source operand) nằm ngay sau mã lệnh
- VD:

```
mov cl, 200;        cl ← 100
mov ax, 0ff0h;      ax ← 0ff0h
mov ds, ax
mox [bx], 200;      chuyển giá trị 200 vào ô nhớ có địa chỉ là DS:BX
```

Ngầm định thanh ghi đoạn ds

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tức thì (Immediate Addressing Mode)

Assembly	Kích thước	Hoạt động
MOV BL,44	8 bit	Đưa số 44 thập phân vào BL
MOV AX,44H	16 bit	Đưa 0044H vào AX
MOV SI,0	16 bit	Đưa 0000H vào SI
MOV AL,'A'	8 bit	Đưa mã ASCII của 'A' vào AL
MOV AX,'AB'	16 bit	Đưa mã ASCII của 'A' vào AH, 'B' vào AL

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Trực tiếp (Direct Addressing Mode)

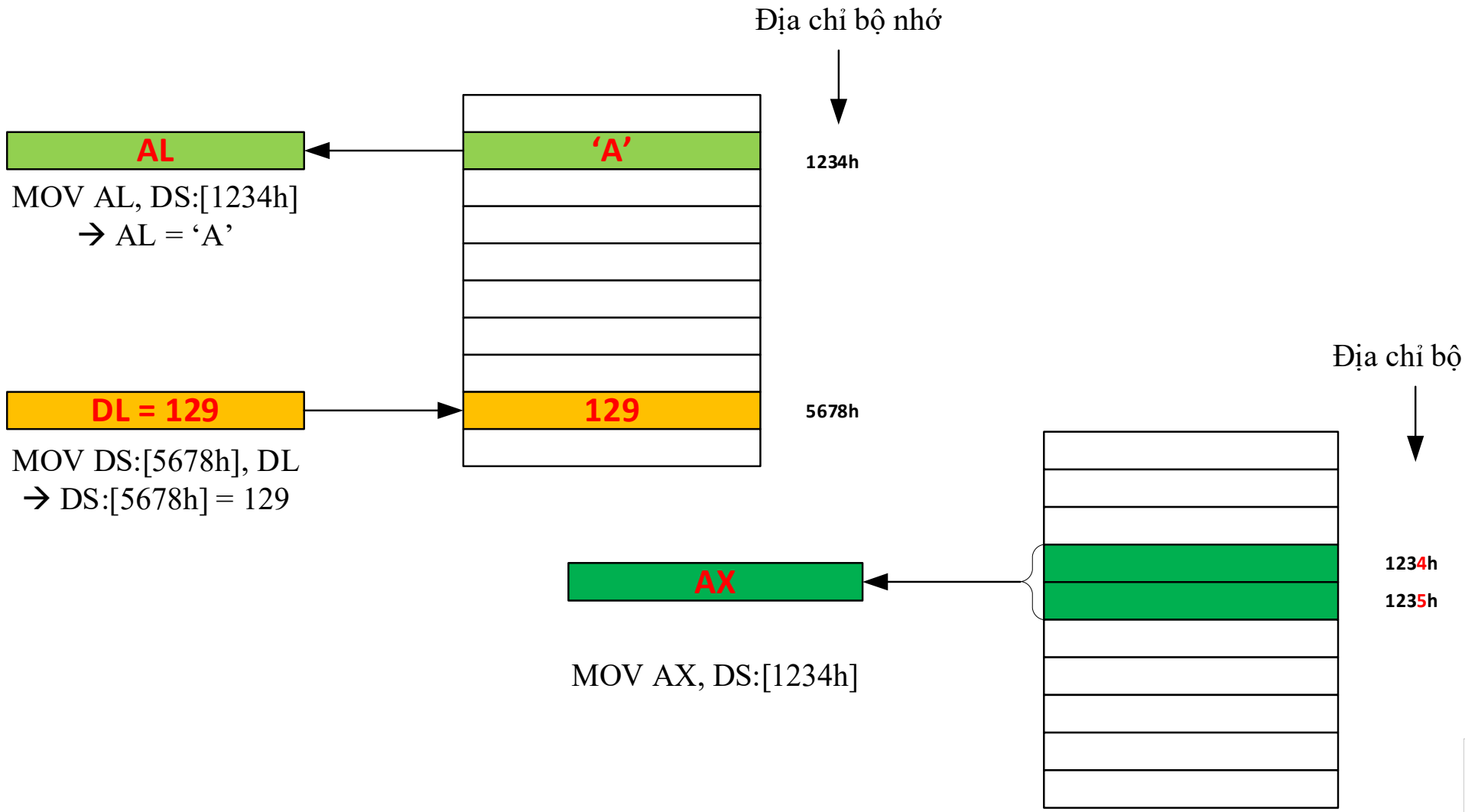
- Sử dụng một hằng để biểu diễn địa chỉ một ô nhớ làm một toán hạng
- Toán hạng còn lại có thể là 1 thanh ghi hoặc 1 địa chỉ ô nhớ
- Ví dụ:

```
MOV AL, [1234h]
MOV [5678h], DL
MOV AX, [1234h]
```

*Chú ý: **DS** là thanh ghi đoạn ngầm định trong chế độ địa chỉ trực tiếp !*

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Trực tiếp (Direct Addressing Mode)



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Gián tiếp qua thanh ghi (Register Indirect Addressing Mode)

- Một toán hạng là một thanh ghi chứa địa chỉ lệch của ô nhớ
- Toán hạng còn lại có thể là thanh ghi hoặc hằng (không được là ô nhớ)
- Ví Dụ:

MOV AL, [BX]; *AL ← [DS:BX]*

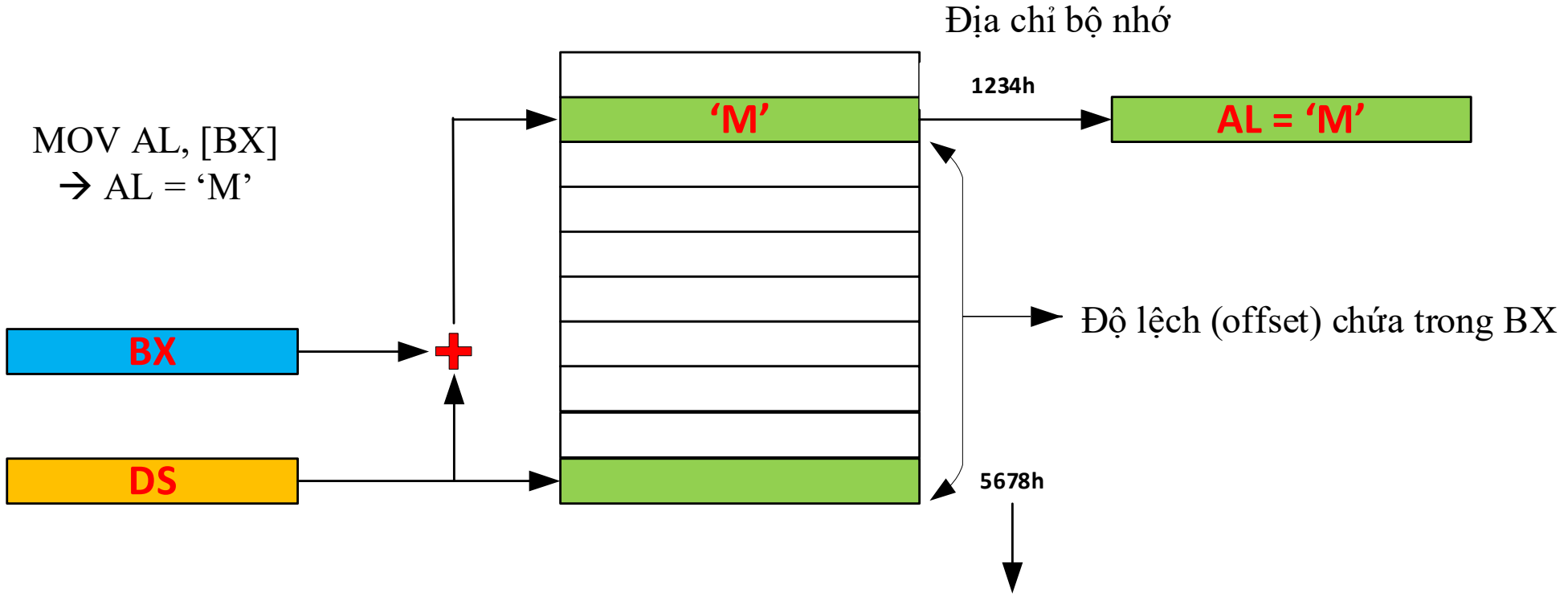
MOV AL, [BP]; *AL ← [SS:BP]*

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Gián tiếp qua thanh ghi (Register Indirect Addressing Mode)

- Ví Dụ: **MOV AL, [BX];** *AL ← [DS:BX]*

*Thanh ghi đoạn ngầm định **DS***

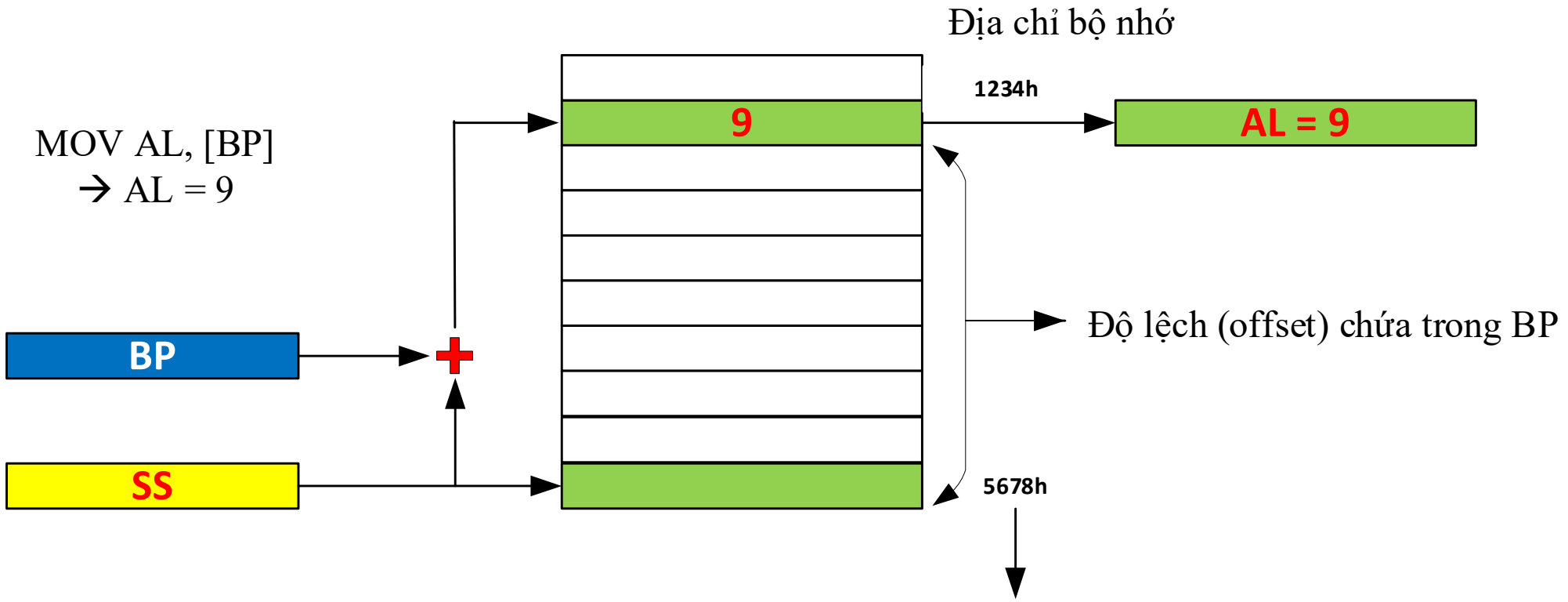


Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Gián tiếp qua thanh ghi (Register Indirect Addressing Mode)

- Ví Dụ: **MOV AL, [BP];** AL← [SS:BP]

*Thanh ghi đoạn ngầm định **SS***



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối cơ sở (Based Plus Displacement Addressing Mode)

- Một toán hạng là địa chỉ của ô nhớ;
- Đ/c của ô nhớ được tạo bởi việc sử dụng thanh ghi cơ sở như BX (đoạn DS) hoặc BP (đoạn SS) và một hằng số biểu diễn độ dịch chuyển (displacement);
- Hằng số trong địa chỉ tương đối cơ sở biểu diễn các giá trị dịch chuyển (displacement) được dùng để tính địa chỉ hiệu dụng (EA- Effective Address) của các toán hạng trong các vùng nhớ DS và SS;
- Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ);

Ví dụ:

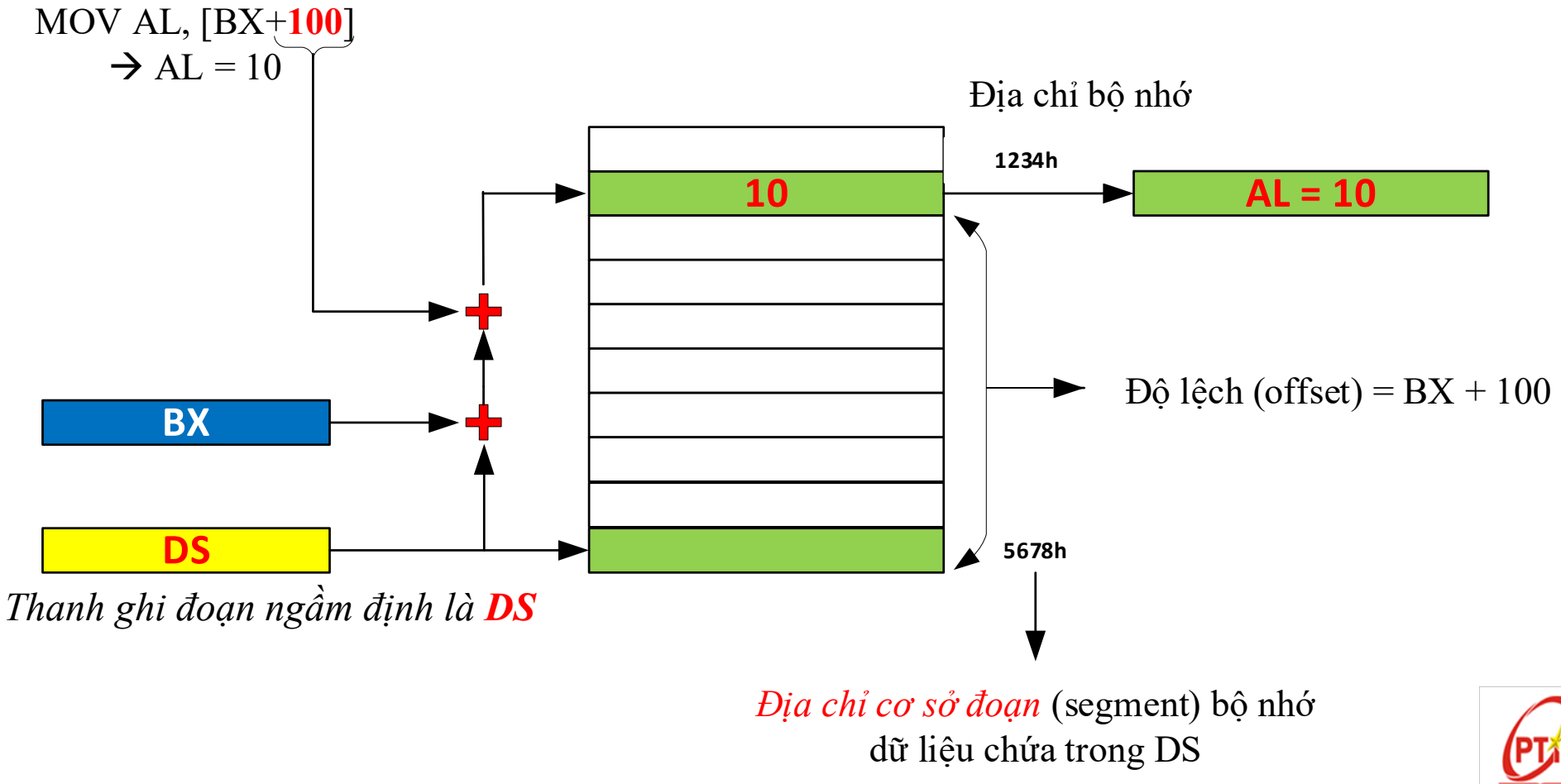
MOV AL, [BX+100]; *AL ← [DS: BX+100]*

MOV AL, [BP+200]; *AL ← [SS: BP+200]*

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối cơ sở (Based Plus Displacement Addressing Mode)

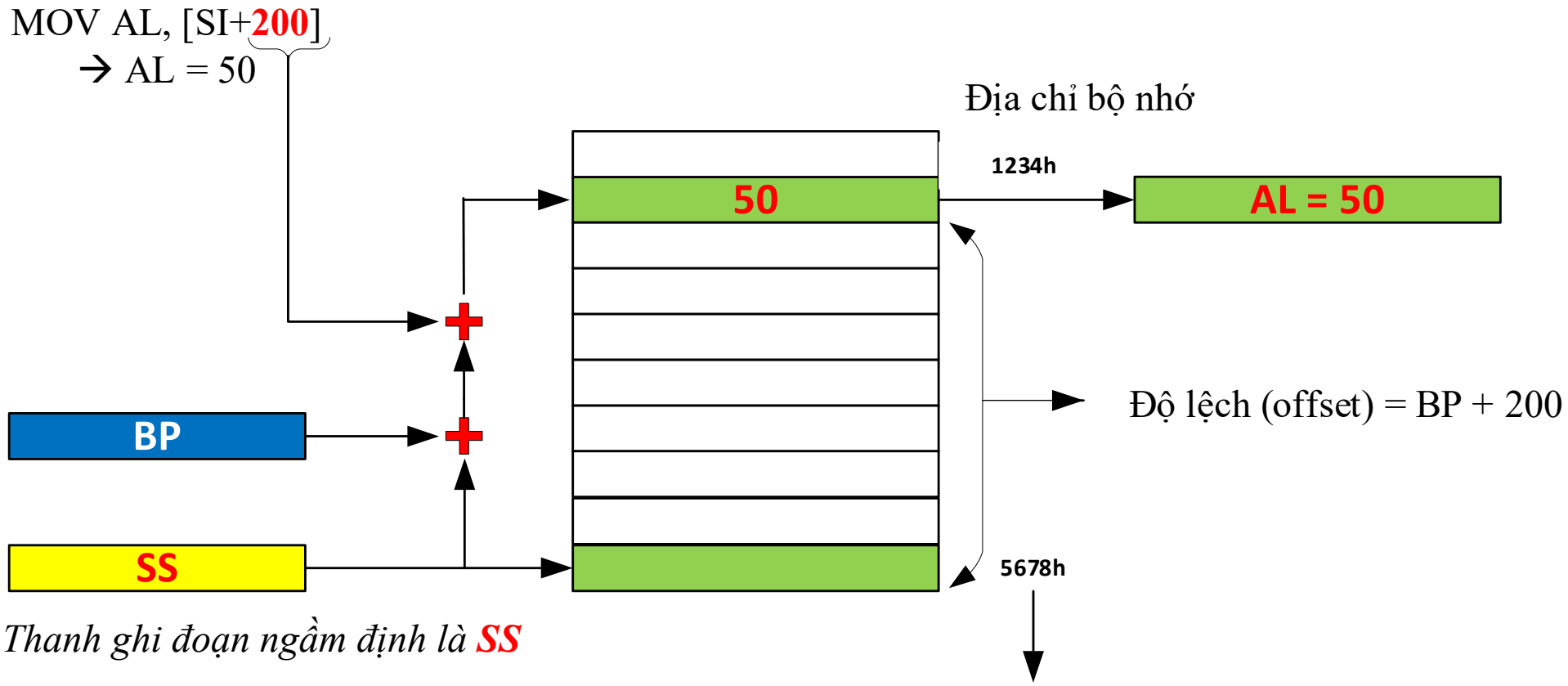
Ví dụ: **MOV AL, [BX+100];** *AL ← [DS: BX+100]*



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối cơ sở (Based Plus Displacement Addressing Mode)

Ví dụ: **MOV AL, [BP+200];** *AL ← [SS: BP+200]*



*Thanh ghi đoạn ngầm định là **SS***

Địa chỉ cơ sở đoạn (segment) bộ nhớ ngăn xếp chứa trong SS

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối chỉ số (Indexed Plus Displacement Addressing Mode)

- Một toán hạng là địa chỉ của ô nhớ.
- Địa chỉ của ô nhớ được tạo bởi việc sử dụng thanh ghi chỉ số SI hoặc DI và một hằng số độ dịch chuyển (displacement);
- Hằng số trong địa chỉ tương đối chỉ số biểu diễn các giá trị dịch chuyển (displacement) được dùng để tính địa chỉ hiệu dụng (EA) của các toán hạng trong các vùng nhớ DS.
- Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ);
- Ví dụ:

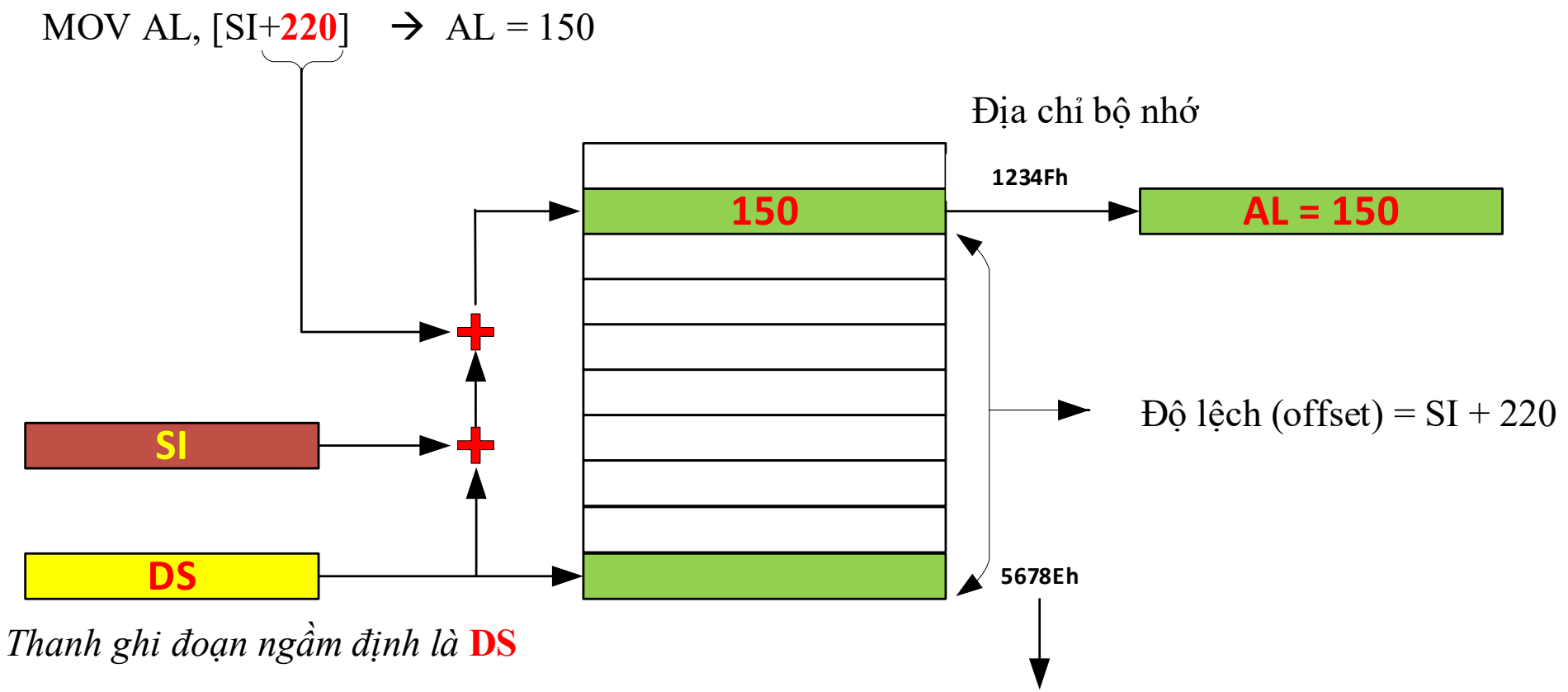
MOV AL, [SI+220]; *AL ← [DS: SI+220]*

MOV AL, [DI+300]; *AL ← [DS: DI+300]*

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối chỉ số (Indexed Plus Displacement Addressing Mode)

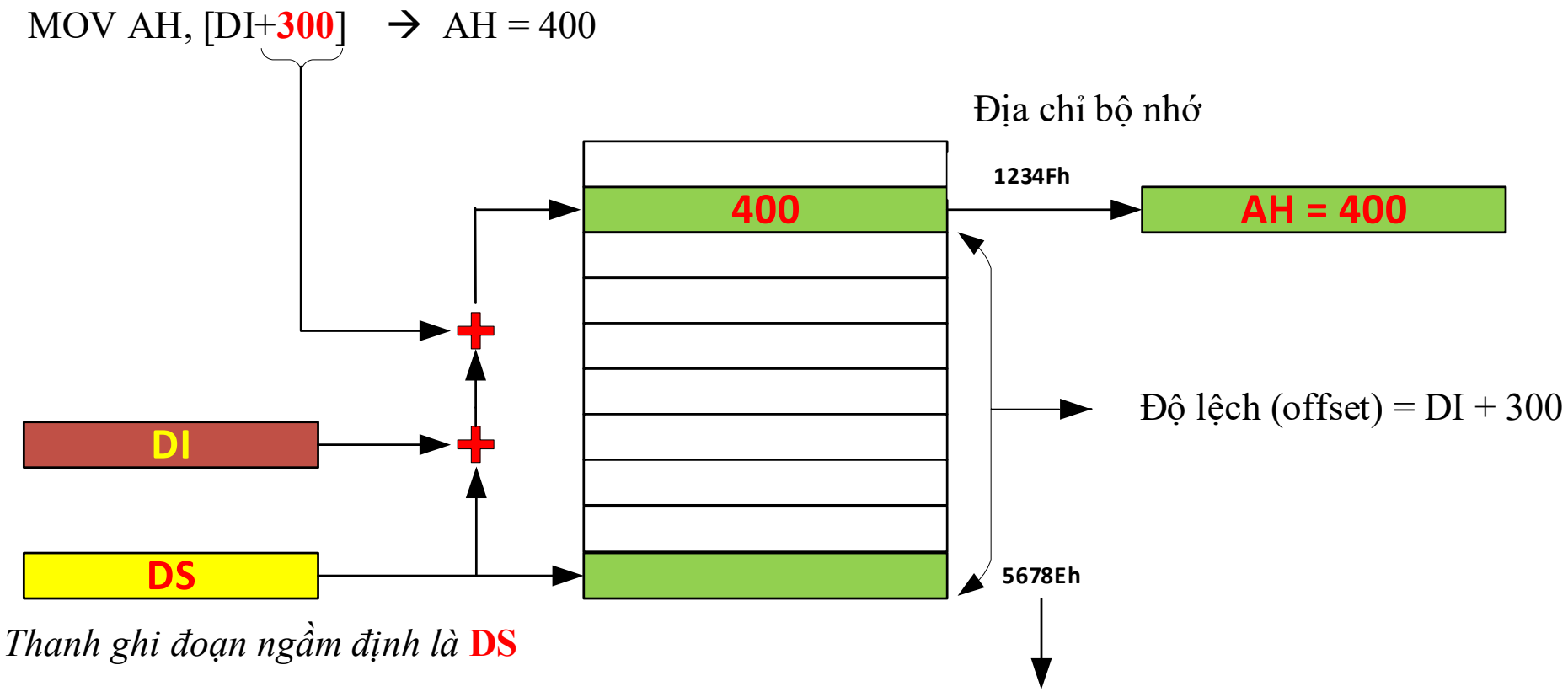
Ví dụ: **MOV AL, [SI+220];** *AL ← [DS: SI+220]*



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối chỉ số (Indexed Plus Displacement Addressing Mode)

Ví dụ: **MOV AH, [DI+300];** *AH ← [DS: DI+300]*



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối chỉ số cơ sở (Based Indexed Plus Displacement Addressing Mode)

- Một toán hạng là địa chỉ của ô nhớ.
- Địa chỉ của ô nhớ được tạo bởi việc sử dụng các thanh ghi cơ sở BX, BP cùng với các thanh ghi chỉ số SI, DI và một hằng số độ dịch chuyển (displacement);
- Việc kết hợp được tạo thành các cặp BX+SI/DI trong đoạn DS (dữ liệu) hoặc BP+SI/DI trong đoạn SS (ngăn xếp)
- Hằng số trong địa chỉ tương đối cơ sở biểu diễn các giá trị dịch chuyển (displacement) được dùng để tính địa chỉ hiệu dụng (EA) của các toán hạng trong các vùng nhớ DS và SS;
- Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ);
    - Ví dụ:

MOV AL, [BX+SI+100]; *AL ← [DS: BX+SI+100]*

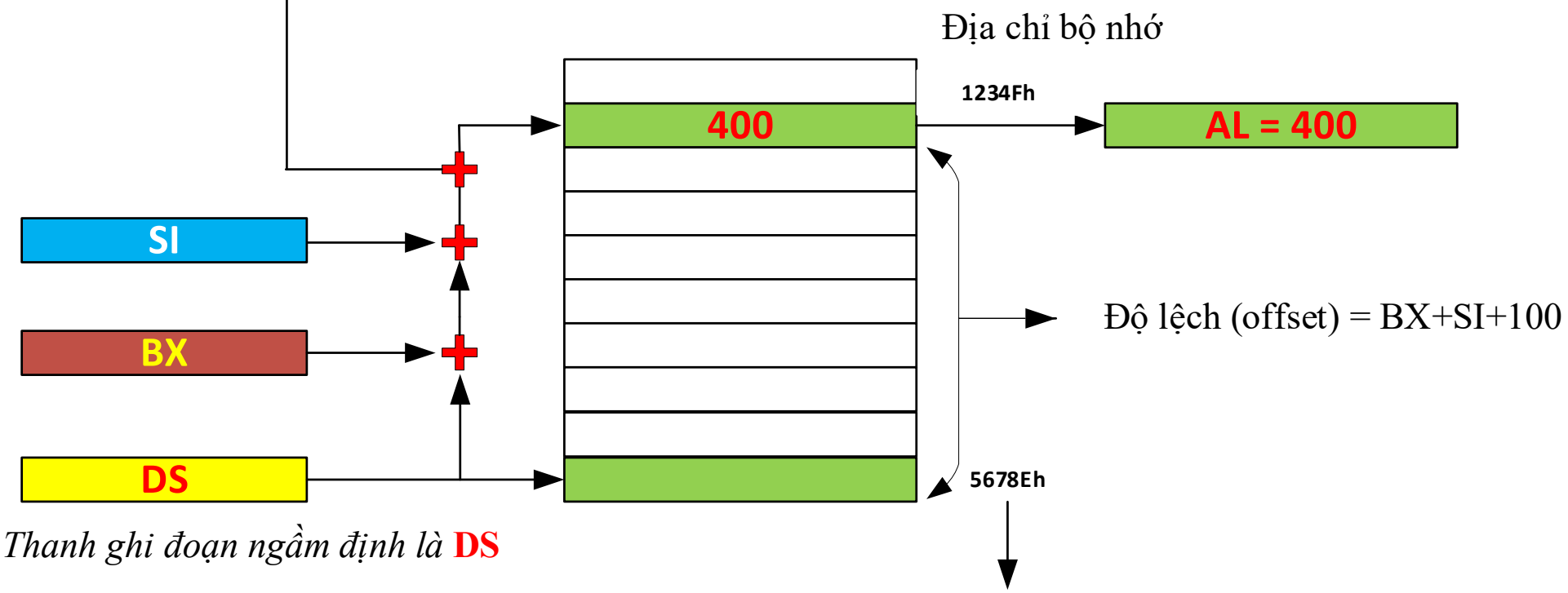
MOV AL, [BP+DI+200]; *AL ← [SS: BP+DI+200]*

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối chỉ số cơ sở (Based Indexed Plus Displacement Addressing Mode)

Ví dụ: **MOV AL, [BX+SI+100];** *AL ← [DS: BX+SI+100]*

MOV AL, [BX+SI+**100**] → AL = 400



Thanh ghi đoạn ngầm định là **DS**

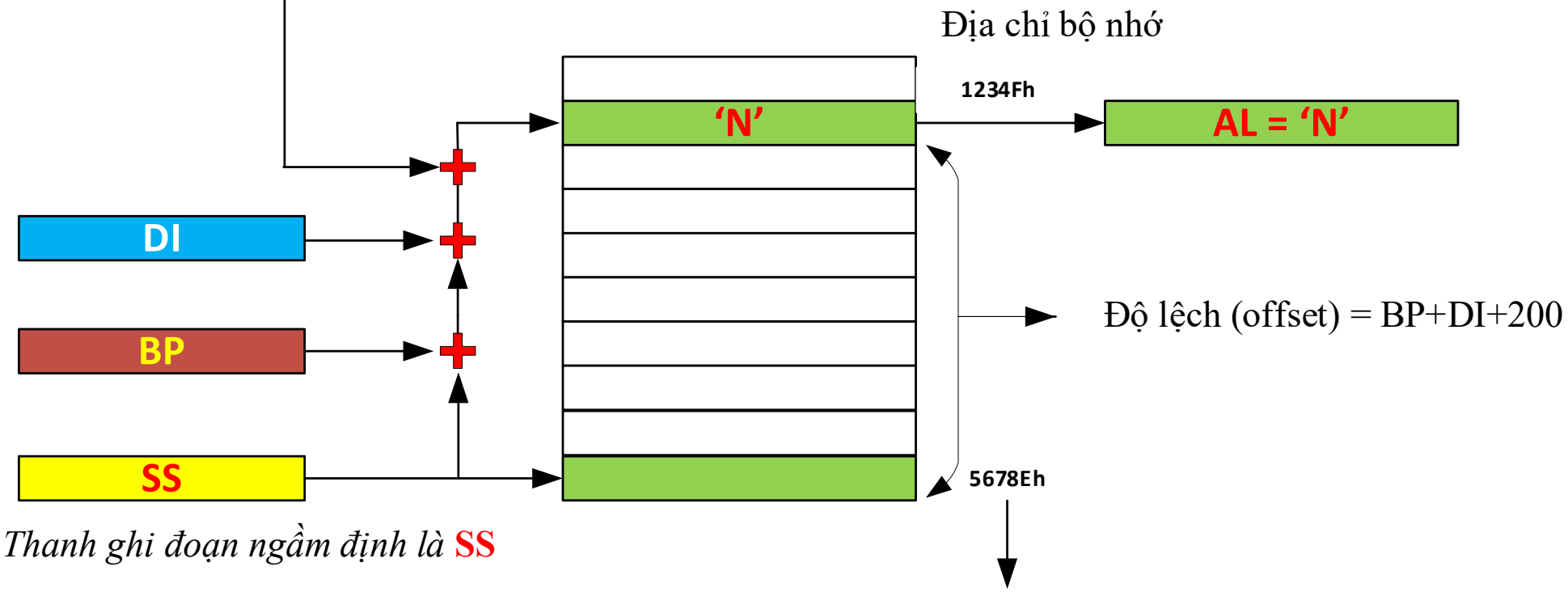
Địa chỉ cơ sở đoạn (segment) bộ nhớ dữ liệu chứa trong DS

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chế độ định địa chỉ Tương đối chỉ số cơ sở (Based Indexed Plus Displacement Addressing Mode)

Ví dụ: **MOV AL, [BP+DI+200];** *AL ← [DS: BP+DI+200]*

MOV AL, [BP+DI+**200**] → AL = 'N'



Thanh ghi đoạn ngầm định là **SS**

Địa chỉ cơ sở đoạn (segment) bộ nhớ ngăn xếp chứa trong SS

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Ánh xạ ngầm định trong các chế độ định địa chỉ

Chế độ định địa chỉ	Toán hạng	Thanh ghi đoạn ngầm định
Thanh ghi	Reg	
Tức thì	Data	
Trực tiếp	[Offset]	DS
Gián tiếp qua thanh ghi	[BX] [SI] [DI]	DS DS DS
Tương đối cơ sở	[BX]+Disp [BP]+Disp	DS SS
Tương đối chỉ số	[DI]+Disp [SI]+Disp	DS DS
Tương đối chỉ số cơ sở	[BX]+[DI]+Disp [BX]+[SI]+Disp [BP]+[SI]+Disp [BP]+[DI]+Disp	DS DS SS SS

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Ánh xạ ngầm định giữa các thanh ghi đoạn và thanh ghi lệch

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

- **Địa chỉ ngầm định**:

MOV AL, [BX]; *AL ← [DS:BX]*

MOV [SI+300], AH; *[DS:SI+300] ← AH*

- **Địa chỉ tường minh (đầy đủ):**

MOV AL, ES:[BX]; *AL ← [ES:BX]*

MOV AH, SS:[SI+300]; *AH ← [SS:SI+300]*

Khi không sử dụng các thanh ghi ngầm định mà sử dụng 1 thanh ghi đoạn khác thì cần phải viết rõ thanh ghi đó ra 1 cách tường minh trong mã lệnh. Được gọi là Bỏ Ngầm Định Thanh Ghi Đoạn (Segment Over Ride)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.3 Tập lệnh và công cụ EMU8086

4.3.1. Tập lệnh của 8086/8088

- Các lệnh di chuyển dữ liệu
- Các lệnh số học và logic
- Các lệnh điều khiển chương trình

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh di chuyển dữ liệu:

- **MOV, XCHG, POP, PUSH, POPF, PUSHF, IN, OUT…**
- Các lệnh di chuyển chuỗi **MOVS, MOVSB, MOVSW**.
  - ✓ Lệnh **MOV**
    - Dùng để chuyển giữa các thanh ghi, giữa 1 thanh ghi và 1 ô nhớ hoặc chuyển 1 số vào thanh ghi hoặc ô nhớ
    - Cú pháp: **MOV Đích, Gốc**
    - Lệnh này không tác động đến cờ
      - Ví dụ:

        MOV AX, BX

        MOV AH, ‘A’

        MOV AL, [1234H]

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh di chuyển dữ liệu:

✓ Lệnh **XCHG**

- Dùng để hoán chuyển nội dung giữa hai thanh ghi, giữa 1 thanh ghi và 1 ô nhớ.
- Cú pháp: **XCHG Đích, Gốc**
- Giới hạn: toán hạng không được là thanh ghi đoạn
- Lệnh này không tác động đến cờ
- Ví dụ:

```
XCHG AX, BX
XCHG AX, [BX]
XCHG AL, [1234H]
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh di chuyển dữ liệu:

✓ Lệnh **PUSH**

- Dùng để cất 1 từ từ thanh ghi hoặc ô nhớ vào đỉnh ngăn xếp.
- Cú pháp: **PUSH Gốc**
- Mô tả: SP=SP-2, Gốc => {SP}
- Giới hạn: thanh ghi 16 bit hoặc là 1 từ nhớ
- Lệnh này không tác động đến cờ
    - Ví dụ:
        - PUSH BX
        - PUSH [BX]

✓ Lệnh **PUSHF** → Cất nội dung của thanh ghi cờ vào ngăn xếp

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- **Các lệnh di chuyển dữ liệu:**

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh di chuyển dữ liệu:

✓ Lệnh **POP**

- Dùng để lấy lại 1 từ vào thanh ghi hoặc ô nhớ từ đỉnh ngăn xếp.
- Cú pháp: **POP Đích**
- Mô tả: {SP} => Đích, SP=SP+2
- Giới hạn: thanh ghi 16 bit (trừ CS) hoặc là 1 từ nhớ
    - Lệnh này không tác động đến cờ
        - Ví dụ:

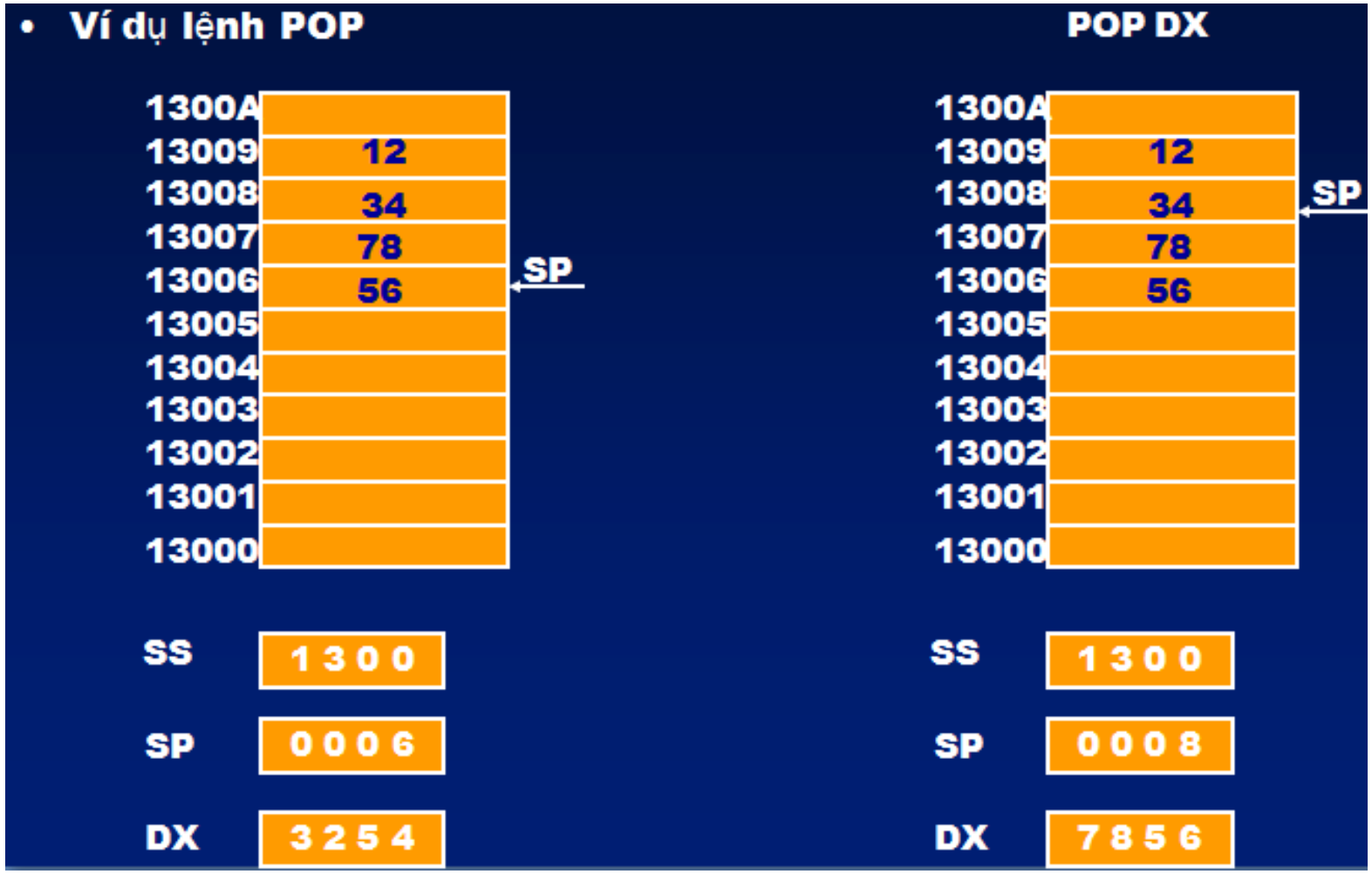
```
POP BX
POP PTR[BX]
```

✓ Lệnh **POPF**

- Lấy 1 từ từ đỉnh ngăn xếp rồi đưa vào thanh ghi cờ

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- **Các lệnh di chuyển dữ liệu:**



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh di chuyển dữ liệu:

- ✓ Lệnh **IN**
  - Dùng để đọc 1 byte hoặc 2 byte dữ liệu từ cổng vào thanh ghi AL hoặc AX
  - Cú pháp: **IN Acc, Port**
  - Port là địa chỉ cổng: 8bit hoặc 16bit. Nếu địa chỉ là 16 bit ta phải đặt địa chỉ vào DX trước khi truy xuất cổng.
    - Lệnh này không tác động đến cờ
      - Ví dụ:

```
IN AX, 00H
IN AL, F0H
IN AX, DX
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh di chuyển dữ liệu:

✓ Lệnh **OUT**

- Dùng để đưa 1 byte hoặc 2 byte dữ liệu từ thanh ghi AL hoặc AX ra cổng
- Cú pháp: **OUT Port, Acc**
- Port là địa chỉ cổng: 8bit hoặc 16bit. Nếu địa chỉ là 16 bit ta phải đặt địa chỉ vào DX trước khi truy xuất cổng.
- Lệnh này không tác động đến cờ
    - Ví dụ:

```
OUT 00H, AX
OUT F0H, AL
OUT DX, AX
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- **Các lệnh di chuyển dữ liệu:** Các lệnh di chuyển chuỗi **MOVS, MOVSB, MOVSW**
  - Dùng để chuyển một phần tử của chuỗi này sang một chuỗi khác
  - Cú pháp: **MOVS Chuỗi đích, Chuỗi gốc**

    MOVSB

    MOVSW
    - Thực hiện:
      - DS:SI là địa chỉ của phần tử trong chuỗi gốc
      - ES:DI là địa chỉ của phần tử trong chuỗi đích
    - Sau mỗi lần chuyển SI=SI +/- 1, DI=DI +/- 1 hoặc SI=SI +/- 2, DI=DI +/- 2 tuỳ thuộc vào cờ hướng DF là 0/1
    - Lệnh này không tác động đến cờ
      - Ví dụ: MOVS byte1, byte2

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- **Các lệnh di chuyển dữ liệu:** Các lệnh di chuyển chuỗi **MOVS, MOVSB, MOVSW**
  - Dùng để chuyển một phần tử của chuỗi này sang một chuỗi khác
  - Cú pháp: **MOVS Chuỗi đích, Chuỗi gốc**

    MOVSB

    MOVSW
    - Thực hiện:
      - DS:SI là địa chỉ của phần tử trong chuỗi gốc
      - ES:DI là địa chỉ của phần tử trong chuỗi đích
    - Sau mỗi lần chuyển SI=SI +/- 1, DI=DI +/- 1 hoặc SI=SI +/- 2, DI=DI +/- 2 tuỳ thuộc vào cờ hướng DF là 0/1
    - Lệnh này không tác động đến cờ
      - Ví dụ: MOVS byte1, byte2

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

- ADD, ADC, SUB, MUL, IMUL, DIV, IDIV, INC, DEC…
- AND, OR, NOT, NEG, XOR…
- Lệnh quay và dịch: ROL, RCL, ROR, RCR, SAL, SAR, SHL, SHR…
- Lệnh so sánh: CMP, CMPS

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **ADD**

- Lệnh cộng hai toán hạng
- Cú pháp: **ADD Đích, Gốc**
- Thực hiện: Đích = Đích + Gốc
- Giới hạn: toán hạng không được là 2 ô nhớ hoặc là thanh ghi đoạn
- Lệnh này thay đổi cờ: AF, CF, OF, PF, SF, ZF
    - Ví dụ:

```
ADD AX, BX
ADD AX, 40H
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **ADC**

- Lệnh cộng có nhớ hai toán hạng
- Cú pháp: **ADC Đích, Gốc**
- Thực hiện: **Đích=Đích + Gốc+CF**
- Giới hạn: toán hạng không được là 2 ô nhớ và thanh ghi đoạn
- Lệnh này thay đổi cờ: AF, CF, OF, PF, SF, ZF
- Ví dụ:

ADC AL, 30H

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **SUB**

- Lệnh trừ
- Cú pháp: **SUB Đích, Gốc**
- Thực hiện: **Đích = Đích – Gốc**
- Giới hạn: toán hạng không được là 2 ô nhớ và thanh ghi đoạn
- Lệnh này thay đổi cờ: AF, CF, OF, PF, SF, ZF
    - Ví dụ:
        - SUB AL, 30H

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

- ✓ Lệnh **MUL**
  - Lệnh nhân số không dấu
  - Cú pháp: MUL Gốc
  - Thực hiện:
    - AX=AL* Gốc(8bit)
    - DXAX=AX*Gốc(16bit)
      - Lệnh này thay đổi cờ: CF, OF
        - Ví dụ: MUL BL
- ✓ Lệnh **IMUL**
  - Lệnh nhân số có dấu

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

- ✓ Lệnh **DIV**
  - Lệnh chia 2 số không dấu
  - Cú pháp: **DIV Gốc**
  - Thực hiện:
    - AL = thương (AX / Gốc8bit); AH=dư (AX / Gốc8bit)
    - AX = thương (DXAX/Gốc16bit); DX=dư (DXAX/Gốc16bit)
  - Lệnh này không thay đổi cờ
    - Ví dụ: DIV BL
- ✓ Lệnh **IDIV**
  - Lệnh chia 2 số có dấu

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **INC**

- Lệnh tăng toán hạng là thanh ghi hoặc ô nhớ lên 1
- Cú pháp: **INC Đích** → Thực hiện: **Đích=Đích + 1**
- Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
- Ví dụ: INC AX

✓ Lệnh **DEC**

- Lệnh giảm từ nội dung một thanh ghi hoặc ô nhớ đi 1
- Cú pháp: **DEC Đích**
- Thực hiện: **Đích=Đích – 1**
- Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
- Ví dụ: DEC [BX]

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **AND**

- Lệnh AND logic 2 toán hạng
- Cú pháp: **AND Đích, Gốc**
- Thực hiện: **Đích= Đích And Gốc**
- Giới hạn: toán hạng không được là 2 ô nhớ hoặc thanh ghi đoạn
- Lệnh này thay đổi cờ: PF, SF, ZF và xoá cờ CF, OF
    - Ví dụ: AND BL, 0FH
        - ✓ Lệnh XOR, OR: tương tự như lệnh AND
        - ✓ Lệnh NOT: đảo từng bit của toán hạng
        - ✓ Lệnh NEG: Lấy bù 2 của toán hạng

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **CMP**

- Lệnh so sánh 2 byte hoặc 2 từ
- Cú pháp: **CMP Đích, Gốc**
- Thực hiện:
    - Đích = Gốc : CF=0 ZF=1
    - Đích> Gốc : CF=0 ZF=0
    - Đích < Gốc : CF=1 ZF=0
        - Giới hạn: toán hạng phải cùng độ dài và không được là 2 ô nhớ
        - Cập nhật: AF, CF, OF, PF, SF, ZF

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **CMPS**

- Dùng để so sánh từng phần tử của 2 chuỗi có các phần tử cùng loại
- Cú pháp: **CMPS Chuỗi đích, Chuỗi Gốc**

  CMPSB

  CMPSW

- Thực hiện:
  - DS:SI là địa chỉ của phần tử trong chuỗi nguồn
  - ES:DI là địa chỉ của phần tử trong chuỗi đích
  - Sau mỗi lần so sánh SI=SI +/- 1, DI=DI +/- 1 hoặc SI=SI +/- 2, DI=DI +/- 2 tuỳ thuộc vào cờ hướng DF là 0/1
    - Cập nhật cờ: AF, CF, OF, PF, SF, ZF

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **ROL**

- Lệnh quay trái
- Cú pháp: **ROL Đích, CL (với số lần quay lớn hơn 1)**

ROL Đích, 1

ROL Đích, Số lần quay (80286 trở lên)

- Thực hiện: quay trái **Đích** CL lần
- Đích là thanh ghi (trừ thanh ghi đoạn) hoặc ô nhớ
- Lệnh này thay đổi cờ: CF, OF

✓ Lệnh **ROR**

- Lệnh quay phải

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **RCL**

- Lệnh quay trái thông qua cờ nhớ
- Cú pháp: **RCL Đích, CL (với số lần quay lớn hơn 1)**

RCL Đích, 1

RCL Đích, Số lần quay (80286 trở lên)

- Thực hiện: quay trái **Đích** qua cờ nhớ CL lần
- Đích là thanh ghi (trừ thanh ghi đoạn) hoặc ô nhớ
- Lệnh này thay đổi cờ: CF, OF

✓ Lệnh **RCR**

- Lệnh quay phải thông qua cờ nhớ

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

- Lệnh dịch trái số học
- Cú pháp: **SAL Đích, CL (với số lần dịch lớn hơn 1)**

  SAL Đích, 1

  SAL Đích, số lần dịch (80286 trở lên)
- Thực hiện: dịch trái **Đích** CL bit tương đương với Đích=Đích*2^{CL}
- Lệnh này thay đổi cờ SF, ZF, PF

✓ Lệnh **SHL**

- Lệnh dịch trái logic, tương tự như SAL

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **SAR**

- Lệnh dịch phải số học
- Cú pháp: **SAR Đích, CL (với số lần dịch lớn hơn 1)**

  SAR Đích, 1

  SAR Đích, số lần dịch (80286 trở lên)

- Thực hiện: dịch phải **Đích** CL bit
  - Lệnh này thay đổi cờ SF, ZF, PF

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh số học và logic

✓ Lệnh **SHR**

- Lệnh dịch phải logic
- Cú pháp: **SHR Đích, CL (với số lần dịch lớn hơn 1)**

SHR Đích, 1

SHR Đích, số lần dịch (80286 trở lên)

- Thực hiện: dịch phải **Đích** CL bit
- Lệnh này thay đổi cờ SF, ZF, PF, CF mang giá trị của LSB

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các lệnh điều khiển chương trình:

- Lệnh nhảy không điều kiện: JMP
- Lệnh nhảy có điều kiện JE, JG, JGE, JL, JLE…
- Lệnh lặp LOOP
- Lệnh gọi chương trình con CALL và RET
- Lệnh gọi chương trình con phục vụ ngắt INT và IRET

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh nhảy không điều kiện **JMP**

- Dùng để nhảy tới một địa chỉ trong bộ nhớ
- Có 3 loại: nhảy ngắn, gần và xa
- Lệnh nhảy ngắn (short jump):
  - Độ dài lệnh 2 bytes:
  - Phạm vi nhảy: -128 đến 127 bytes so với lệnh tiếp theo lệnh JMP
    - Thực hiện: IP=IP + độ lệch
    - Ví dụ:

```
        XOR BX, BX
Nhan:   MOV AX, 1
        ADD AX, BX
        JMP SHORT Nhan
        MOV AH, 10
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

- ✓ Lệnh nhảy không điều kiện **JMP**
  - ▪ Lệnh nhảy gần (near jump):
    - ○ Phạm vi nhảy: ± 32 Kbytes so với lệnh tiếp theo lệnh JMP
    - ○ Ví dụ:

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

- ✓ Lệnh nhảy không điều kiện **JMP**
  - ▪ Lệnh nhảy xa (far jump) :
    - ○ Phạm vi nhảy: nhảy trong 1 đoạn mã hoặc nhảy sang đoạn mã khác
    - ○ Ví dụ:

```
EXTRN   Nhan: FAR
Next: MOV AX, 1
      ADD AX, BX
      JMP FAR PTR Next
      ........
      JMP  FAR Nhan
```

Thực hiện: IP=IP của nhãn
CS=CS của nhãn

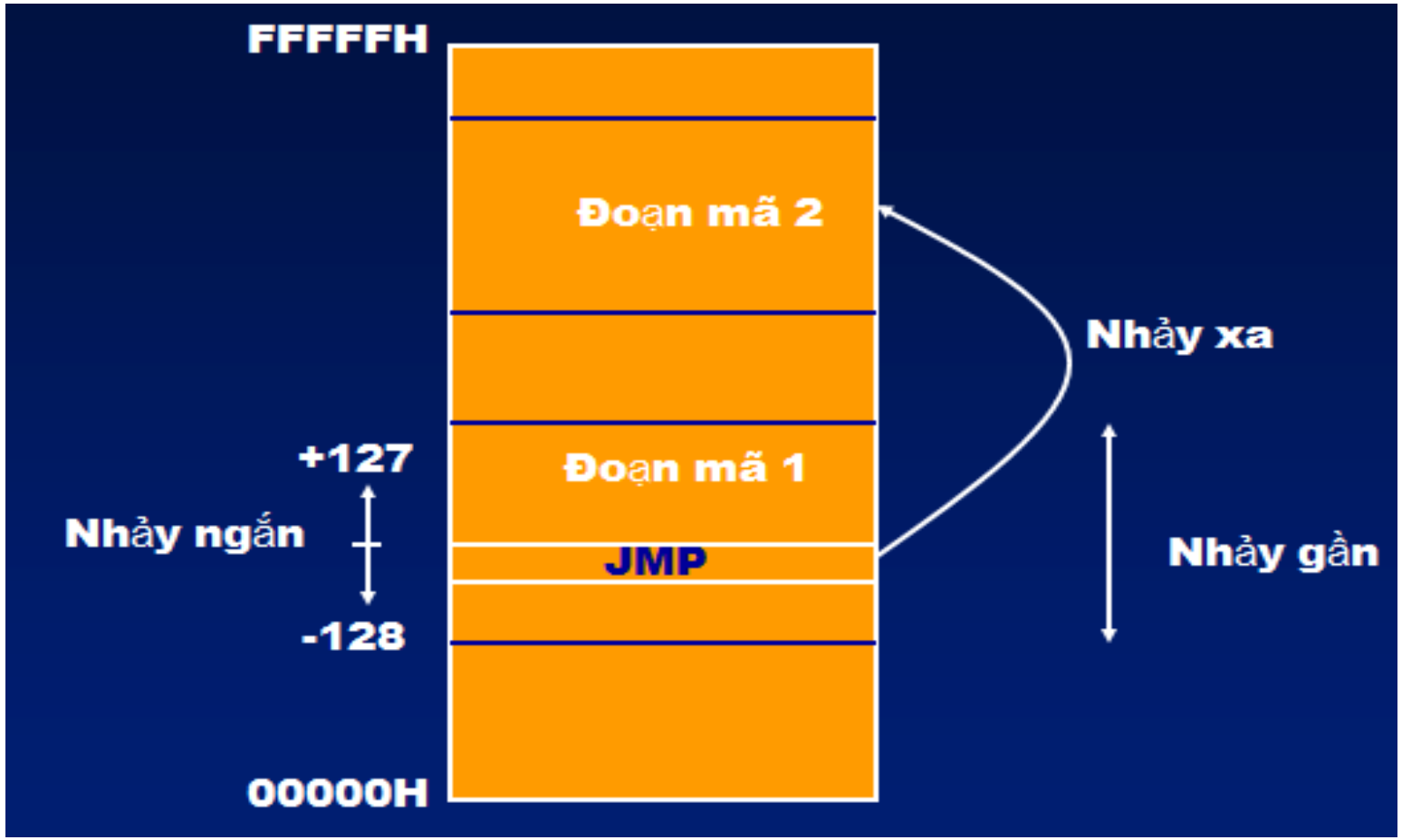
```
XOR CX, CX
MOV AX, 1
ADD AX, BX
JMP DWORD PTR [BX]
```

IP = [BX+1][BX]
CS= [BX+3][BX+2]

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh nhảy không điều kiện **JMP**



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh nhảy có điều kiện: **JE or JZ, JNE or JNZ, JG, JGE, JL, JLE (dùng cho số có dấu) và JA, JB, JAE, JBE (dùng cho số không dấu) ...**

- Nhảy được thực hiện phụ thuộc vào các cờ
- Là các lệnh nhảy ngắn
- Thực hiện: IP=IP + độ lệch
- Ví dụ:

```
Nhan1: XOR BX, BX
Nhan2: MOV AX, 1
       CMP AL, 10H
       JNE Nhan1
       JE  Nhan2
       MOV AL,10
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh lặp: **LOOP, LOOPE/LOOPZ, LOOPNE/ LOOPNZ...**

- Thực hiện các vòng lặp
- Ví dụ:

```
        XOR AL, AL
        MOV CX, 16
Lap:    INC AL
        LOOP Lap
```

Lặp đến khi CX=0

```
        XOR AL, AL
        MOV CX, 16
Lap:    INC AL
        CMP AL, 10
        LOOPE Lap
```

Lặp đến khi CX=0 hoặc AL<>10

```
        XOR AL, AL
        MOV CX, 16
Lap:    INC AL
        CMP AL, 10
        LOOPNE Lap
```

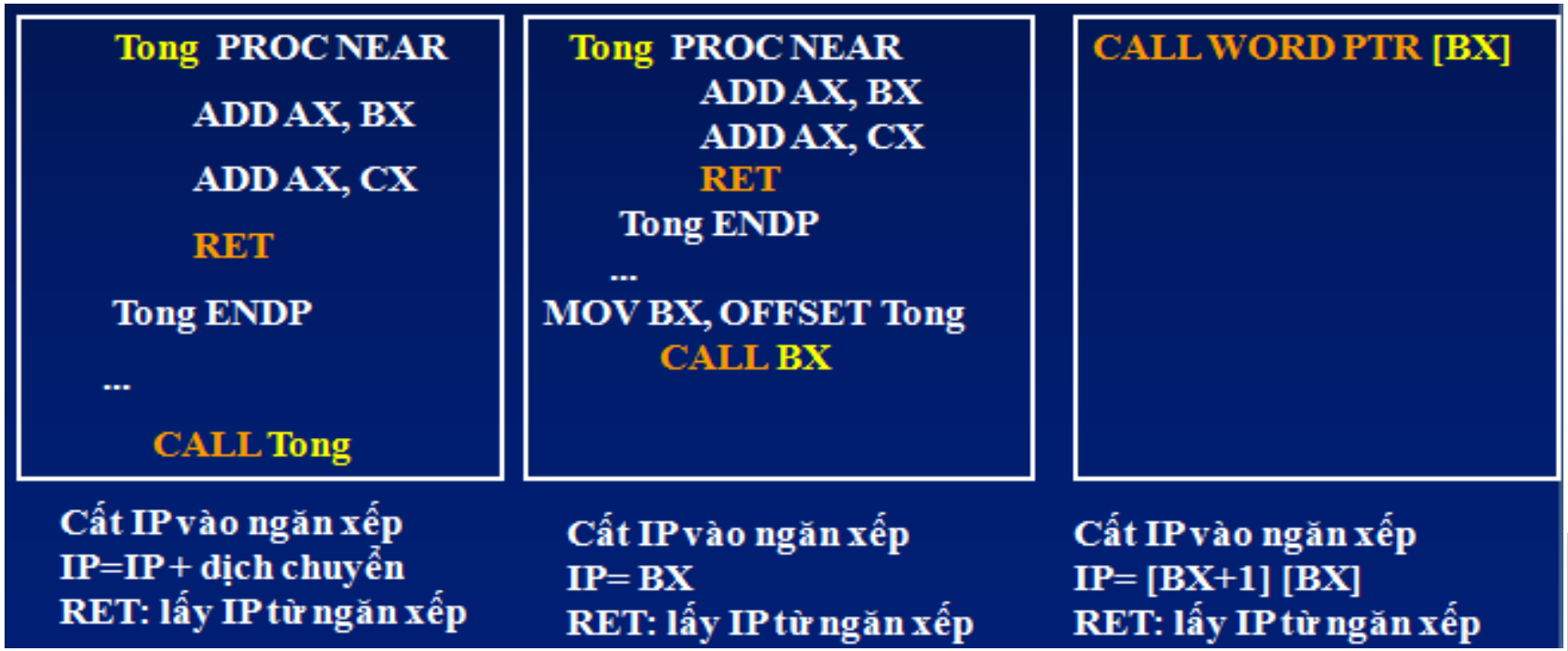
Lặp đến khi CX=0 hoặc AL=10

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh gọi chương trình con CALL và RET

- CALL dùng để gọi chương trình con
- Có 2 loại: CALL gần và CALL xa
- CALL gần (near call): tương tự như nhảy gần. Gọi chương trình con ở trong cùng một đoạn mã

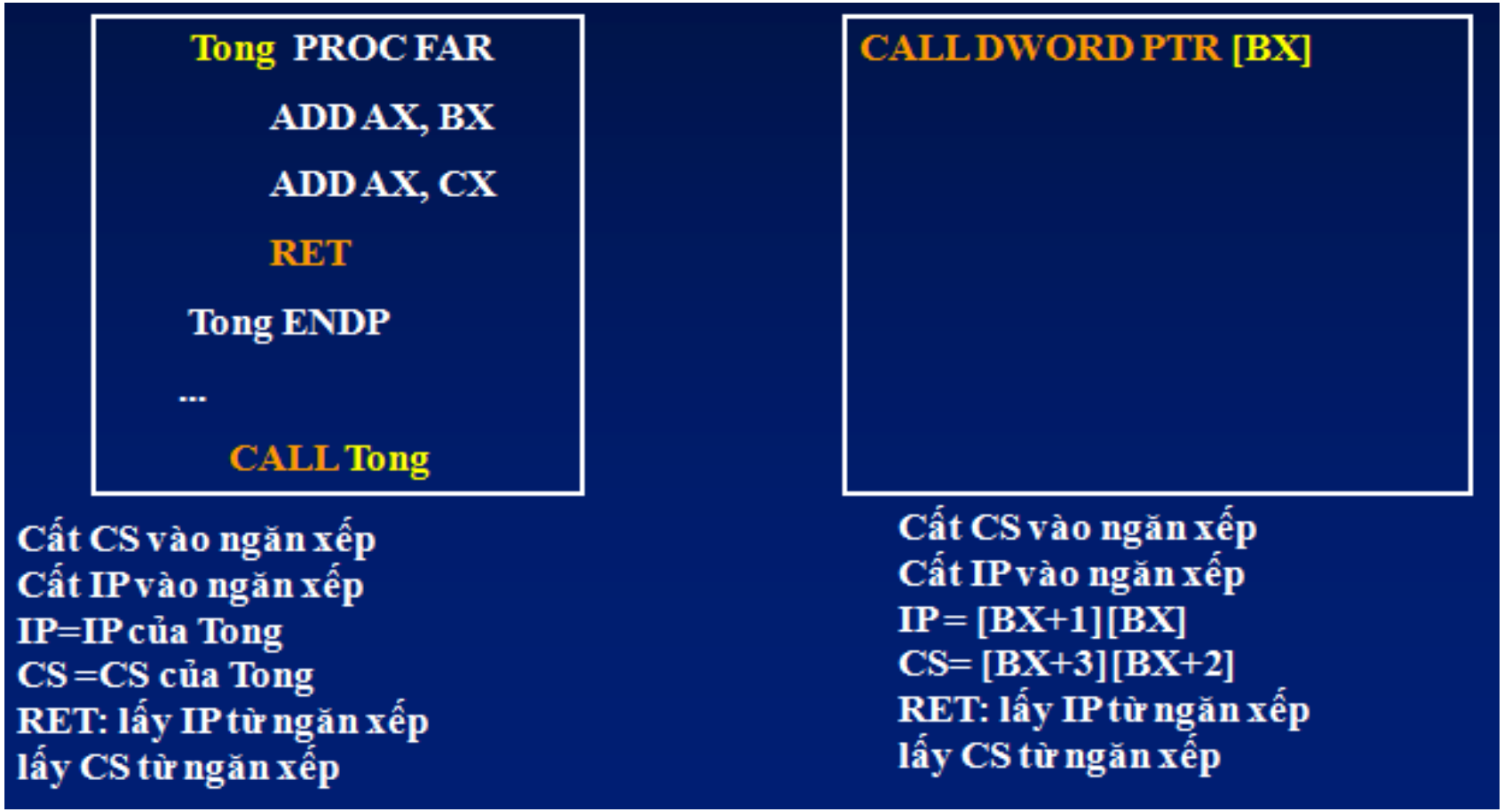


Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh gọi chương trình con CALL và RET

- CALL xa (far call): tương tự như nhảy xa. Gọi chương trình con ở ngoài đoạn mã.



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh gọi chương trình con phục vụ ngắt INT và IRET

- INT gọi chương trình con phục vụ ngắt (CTCPVN)
- Bảng vector ngắt: 1 Kbytes 00000H đến 003FF H
  - 256 vector ngắt
  - 1 vector 4 bytes, chứa IP và CS của CTCPVN
- Cú pháp: INT Number (0..FFH)
    - Ví dụ: INT 21H gọi CTCPVN có số hiệu 21H

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

❑ Các lệnh điều khiển chương trình:

✓ Lệnh gọi chương trình con phục vụ ngắt INT và IRET

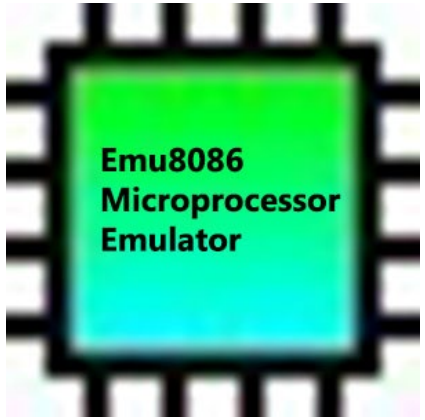
- **Thực hiện INT:**
  - Cất thanh ghi cờ vào ngăn xếp
  - IF=0 (cấm các ngắt khác tác động), TF=0 (chạy suốt)
  - Cất CS vào ngăn xếp
    - Cất IP vào ngăn xếp
    - IP=[N*4], CS=[N*4+2]
      - **Gặp IRET**
        - Lấy IP từ ngăn xếp
        - Lấy CS từ ngăn xếp
        - Lấy thanh ghi cờ từ ngăn xếp

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.3 Tập lệnh và công cụ EMU8086

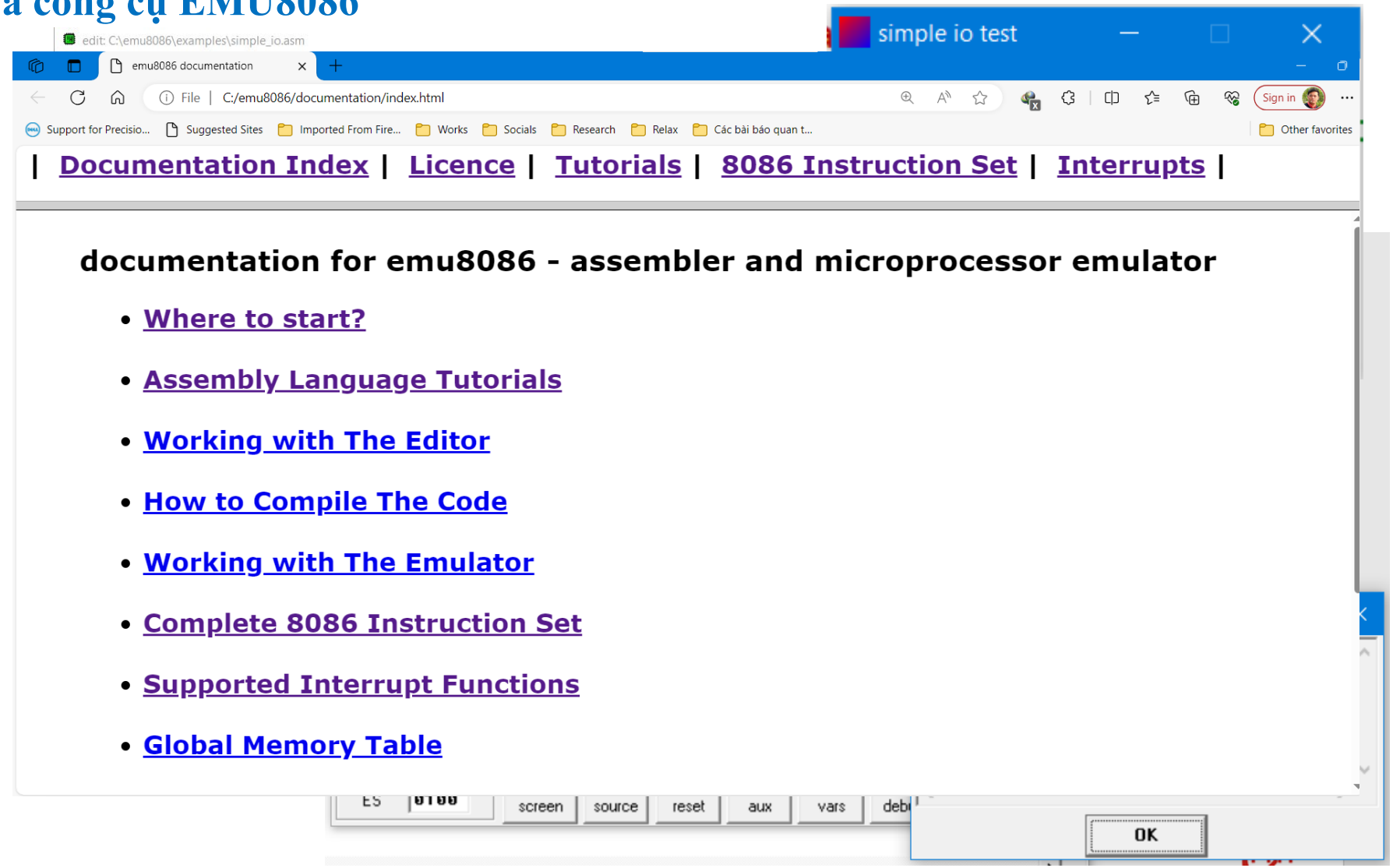
4.3.2 Công cụ EMU8086

- EMU8086 - MICROPROCESSOR EMULATOR là trình giả lập miễn phí cho nhiều nền tảng.
- Cung cấp cho người dùng khả năng mô phỏng bộ xử lý 8086 cũ, được sử dụng trong máy tính Macintosh và Windows từ những năm 1980 và đầu những năm 1990.
- Có thể mô phỏng việc viết mã và thực thi một lượng lớn các chương trình hợp ngữ.
- Mô phỏng phần cứng, màn hình, bộ nhớ và các thiết bị I/O thực.
- Cung cấp cho người dùng đặc quyền kiểm tra thiết bị ảo của họ được lập trình bằng hợp ngữ hoặc bất kỳ ngôn ngữ nào khác.



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.3 Tập lệnh và công cụ EMU8086



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

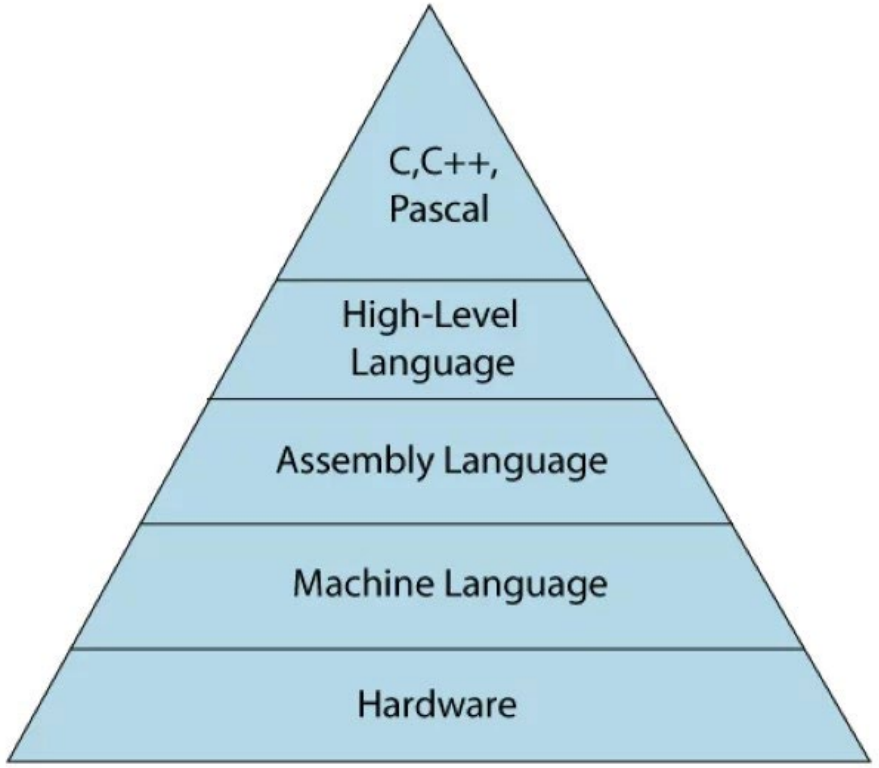
4.4 Lập trình hợp ngữ

- *Assembly language* = *Hợp ngữ* → là ngôn ngữ lập trình cấp thấp có sự tương ứng rất mạnh mẽ giữa tập lệnh của ngôn ngữ và tập lệnh mã máy của kiến trúc máy tính cụ thể.
- Hợp ngữ còn được gọi là mã máy tượng trưng *(symbolic machine code)* → ngôn ngữ hướng phần cứng !
- Mã hợp ngữ được chuyển đổi thành mã máy thực thi bằng một chương trình được gọi là *assembler (hợp dịch)*. Quá trình chuyển đổi được gọi là *assembling.* Hợp ngữ thường có một câu lệnh trên một lệnh máy (1:1), nhưng cũng hỗ trợ các comment (chú thích) là chỉ thị trình biên dịch, các macros và các nhãn chương trình và địa chỉ bộ nhớ .

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.4 Lập trình hợp ngữ

- Hợp ngữ đã từng được dùng rộng rãi trong tất cả các khía cạnh lập trình, nhưng ngày nay nó có xu hướng chỉ được dùng trong một số lãnh vực hẹp, chủ yếu để giao tiếp trực tiếp với phần cứng hoặc xử lý các vấn đề liên quan đến tốc độ cao điển hình như các trình điều khiển thiết bị, các hệ thống nhúng cấp thấp và các ứng dụng thời gian thực.



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.4 Lập trình hợp ngữ

Hợp ngữ cấp thấp LLA (low level assembly) và cấp cao HLA (hight level assembly)

- Hợp ngữ cấp thấp LLA mỗi một lệnh của nó được hợp dịch trực tiếp thành một lệnh máy của vi xử lý.
- Hợp ngữ cấp cao HLA trước hết phải được dịch nhờ trình dịch (HLA compiler) để trở thành chương trình hợp ngữ cấp thấp, sau đó hợp dịch (assembling) để trở thành mã máy.
- HLA là một gói phần mềm gồm : phần mềm biên dịch (compiler) chương trình ở ngôn ngữ bậc cao thành chương trình ở ngôn ngữ bậc thấp, thư viện chuẩn của HLA, tập các tệp INCLUDE cho thư viện chuẩn HLA.
- HLA là một ứng dụng cho các hệ điều hành Linux và Windows do nó có khả năng đáp ứng được những tiêu chuẩn của ngôn ngữ bậc cao (dễ viết..gần gũi) nhưng lại cho phép biên dịch thành chương trình hợp ngữ bậc thấp..

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Cách tạo và chạy một chương trình hợp ngữ

Để tạo và chạy một chương trình hợp ngữ trên máy tính cần phải thực hiện 4 bước.

1. Soạn thảo tệp chương trình nguồn hợp ngữ (file *.asm)
2. Hợp dịch chương trình nguồn thành file đối tượng ở ngôn ngữ máy (*.obj)
3. Liên kết một hay nhiều đối tượng tạo ra một tệp chương trình chạy được (program file).
4. Chạy chương trình

➢ **Soạn thảo tệp chương trình nguồn (coding)**

- ✓ Dùng một chương trình soạn thảo văn bản bất kỳ (NotePad, WordPad…MS Word trên Windows hoặc EDIT trong DOS) để soạn tệp chương trình nguồn theo quy định cấu trúc của một chương trình hợp ngữ. Cũng có thể dùng chương trình EMU8086 để soạn thảo chương trình nguồn
- ✓ Đặt tên file với phần mở rộng mặc định là *.ASM

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Cách tạo và chạy một chương trình hợp ngữ

➢ Hợp dịch tệp chương trình nguồn thành tệp đối tượng (asembling)

- ✓ Để hợp dịch file chương trình nguồn *.ASM thành file đối tượng (*.OBJ) ngôn ngữ máy cần sử dụng chương trình hợp dịch Assembler.
- ✓ Thông dụng là TASM của Borlan hay MASM (Macro Assembler) của Microsoft.
- ✓ File đối tượng có phần mở rộng là *.OBJ và trùng với tên file nguồn.
- ✓ File đối tượng là một file ngôn ngữ máy (bao gồm các chỉ dẫn) tuy nhiên chưa thể thực hiện được ngay vì nó chưa biết được sẽ được chứa ở vùng nhớ nào để thực hiện.

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Cách tạo và chạy một chương trình hợp ngữ

➢ **Liên kết một hay nhiều file đối tượng để tạo ra một file chương trình chạy được (linking)**

✓ Dùng chương trình Tlink của Borlan hoặc link của MASM để thực hiện việc kết hợp một hay nhiều file đối tượng thành một file đối tượng chung. Các bước này được thực hiện tự động với EMU8086

✓ Xác định các tên và địa chỉ chưa được định nghĩa và tạo thành một file chương trình chạy được với một không gian nhớ được cấp phát, file này có phần mở rộng là *.EXE hoặc *.COM.

➢ **Chạy chương trình**

✓ Để chạy chương trình chỉ cần gõ tên chương trình ở dòng lệnh mà không cần gõ phần mở rộng .EXE hay .COM vì đây là mặc định.

✓ Trên màn hình xuất hiện kết quả của chương trình được chạy

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Cú pháp của chương trình hợp ngữ

- ✓ Một chương trình mã nguồn hợp ngữ cấp thấp (assembly) phải được viết theo quy định cho phù hợp với trình hợp dịch (asembler).
- ✓ Mã lệnh được viết theo từng dòng và không phân biệt chữ hoa, chữ thường.
- ✓ Lệnh có 2 dạng:

    * Lệnh thật: là các lệnh gợi nhớ của vi xử lý ví dụ: MOV, SUB, ADD,...

    Khi dịch, lệnh gợi nhớ được dịch ra mã máy

    * Giả lệnh: là các hướng dẫn chương trình hợp dịch:

        Ví Dụ: MAIN PROC, .DATA, END MAIN,... Khi dịch, lệnh giả không được dịch ra mã máy mà chỉ có tác dụng định hướng cho chương trình dịch.

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các trường của một dòng lệnh thông thường phải tuân thủ theo thứ tự sau:

TÊN: MÃ_LỆNH TOÁN_HẠNG ; *CHÚ GIẢI*

Ví dụ : Nhan_1: MOV AX, 01Fh ; *nạp giá trị 01Fh vào thanh ghi AX*

Trường tên

- Là nhãn, tên biến, hằng hoặc thủ tục. Sau nhãn là dấu hai chấm (:)
- Các tên sẽ được chương trình dịch gán địa chỉ ô nhớ.
- Tên chỉ có thể gồm các chữ cái, chữ số, dấu gạch dưới và phải bắt đầu bằng 1 chữ cái

Trường Mã lệnh

- Có thể gồm lệnh thật và lệnh giả: Ví dụ: **mov**…**add**….**sub**…**int**

Trường Toán hạng

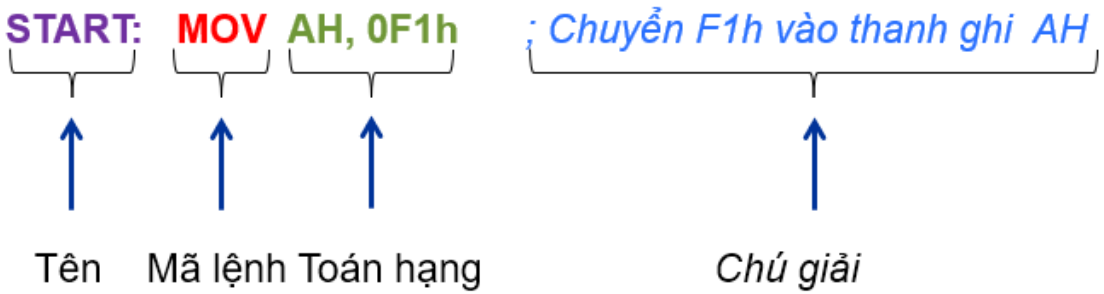
- Số lượng toán hạng phụ thuộc vào lệnh cụ thể
- Có thể có 0, 1 và 2 toán hạng

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Trường Chú giải (comment)

Là chú thích cho dòng lệnh. Bắt đầu bằng dấu chấm phảy (;)

Ví dụ:



Dữ liệu cho chương trình hợp ngữ

- Dữ liệu số:

  * Thập phân: 0-9, Thập lục phân: 0-9, A-F , Bắt đầu bằng 1 chữ (A-F) thêm 0 vào đầu. Thêm ký hiệu H (Hexa) ở cuối. Ví dụ: 80H, 0F9H

  * Nhị phân: 0-1 Thêm ký hiệu B (Binary) ở cuối. Ví dụ: 0111B, 1000B

- Dữ liệu ký tự:

  * Bao trong cặp nháy đơn hoặc kép. Có thể dùng ở dạng ký tự hoặc mã ASCII. Ví dụ: 'A' = 65, 'a' = 97…….

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Biến và hằng

- **Hằng** (constant):là các đại lượng không thay đổi giá trị. Hai loại hằng:

  Hằng giá trị: ví dụ 100, 'A'

  Hằng có tên: ví dụ MAX_VALUE

Định nghĩa hằng có tên:

<Tên hằng> EQU <Giá trị>

Ví dụ:

```
MAX     EQU  100
ENTER   EQU  13
ESC     EQU  27
```

- **Biến (variable):** Là các đại lượng có thể thay đổi giá trị

  Các loại biến: *Biến đơn*, *Biến mảng*, *Biến xâu ký tự*

  Khi hợp dịch biến được chuyển thành địa chỉ ô nhớ

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Định nghĩa biến đơn:

Tên_biến **DB** Giá trị khởi đầu → Định nghĩa biến kiểu byte

Tên_biến **DW** Giá trị khởi đầu → Định nghĩa biến kiểu word

Tên_biến **DD** Giá trị khởi đầu → Định nghĩa biến kiểu double word

Ví dụ:
```
X  DB  10    ; Khai báo biến X và khởi trị 10
Y  DW  ?     ; Khai báo biến Y và không khởi trị
Z  DD  1000  ; Khai báo biến Z và khởi trị 1000
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Định nghĩa biến mảng:

Tên mảng **DB** Danh sách giá trị khởi đầu

Tên mảng **DB** Số phần tử Dup (Giá trị khởi đầu)

Tên mảng **DB** Số phần tử Dup (?)

Định nghĩa tương tự cho các kiểu DW và DD

Ví dụ:

```
X  DB  10, 2, 5, 6, 1   ; Khai báo mảng X gồm 5 phần tử có khởi trị
Y  DB  5  DUP(0)        ; Khai báo mảng Y gồm 5 phần tử khởi trị 0
Z  DB  5  DUP(?)        ; Khai báo mảng Z gồm 5 phần tử không khởi trị
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Định nghĩa biến xâu ký tự: có thể được định nghĩa như một xâu ký tự hoặc một mảng các ký tự.

Ví dụ:

str1 **DB** ‘string’

str2 **DB** 73H, 74H, 72H, 69H, 6EH, 67H

str2 **DB** 73H, 74H, ‘r’, ‘i’, 69H, 6EH, 67H

Định nghĩa biến mảng:

Tên_mảng DB Danh sách giá trị khởi đầu

Tên_mảng DB Số phần tử Dup(Giá trị khởi đầu)

Tên_mảng DB Số phần tử Dup(?)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Định nghĩa tương tự cho các kiểu DW và DD

Ví dụ:

```
X   DW   10, 2, 5, 6, 1 ; Khai báo mảng X gồm 5 phần tử có khởi trị
Y   DW   5   DUP(0)    ; Khai báo mảng Y gồm 5 phần tử khởi trị 0
Z   DD   5   DUP(?)    ; Khai báo mảng Z gồm 5 phần tử không khởi trị
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Khung của một chương trình hợp ngữ

- ✓ Một chương trình hợp ngữ khi hợp dịch và liên kết thành công sẽ tạo ra một file chạy .
- ✓ Trên môi trường DOS có 2 loại file được thực thi trực tiếp từ dòng lệnh: *.COM và *.EXE

Các mô hình bộ nhớ : Các mô hình bộ nhớ quy định kích thước của chương trình sau khi hợp dịch.

- ➢ Tiny (hẹp): mã lệnh và dữ liệu gói gọn trong một đoạn
- ➢ Small (nhỏ): mã lệnh gói gọn trong một đoạn, dữ liệu gói gọn trong một đoạn
- ➢ Medium (vừa):mã lệnh không gói gọn trong một đoạn, dữ liệu gói gọn trong một đoạn
- ➢ Compact (gọn):mã lệnh gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn.
- ➢ Large (lớn): mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, không có mảng lớn hơn 64K.
- ➢ Huge (rất lớn): mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, có mảng lớn hơn 64K.

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- **Khai báo mô hình bộ nhớ**

.Model <kiểu kích thước bộ nhớ>

Ví dụ:

.Model Tiny ; *hợp dịch trực tiếp ra file *.COM*

- **Khai báo kích thước ngăn xếp**

.Stack <Kích thước ngăn xếp>

Ví dụ:

.Stack 100H ; *khai báo kích thước ngăn xếp 100H=256 byte*

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

➢ Khai báo đoạn dữ liệu

.Data

;Tất cả các biến và hằng phải được khai báo ở đoạn dữ liệu

Ví dụ:

```
.Data
loichao   DB   'Hello!$'   ; một chuỗi luôn kết thúc bằng dấu $
Enter     DB   13          ; mã ASCII của phím enter
Newline   DB   10          ; line feed-cấp dòng mới
Bien_X    DW   2011        ; biến kiểu word
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- Khai báo đoạn mã

.Code

; bắt đầu các lệnh của chương trình

Ví dụ:

```
.Code
MAIN    Proc        ; bắt đầu chương trình chính
                    ; các lệnh của chương trình chính
……
……
MAIN    Endp        ; kết thúc chương trình chính
……                  ; các chương trình con – nếu có
……
End     MAIN
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Khung chương trình dịch ra file *.EXE

```
.Model Small
.Stack 100H
.Data
; khai báo các biến và hằng
.Code
MAIN        Proc
; khởi đầu cho thanh ghi DS
        MOV AX, @Data ; nạp địa chỉ đoạn dữ liệu vào AX ;
        MOV DS, AX    ; nạp địa chỉ đoạn dữ liệu vào DS;
    ; các lệnh của chương trình chính
            MOV AH, 4CH   ; kết thúc chương trình trả quyền điều khiển về cho hệ điều
            INT 21H       ; hành dùng hàm 4CH của ngắt 21H;
            MAIN Endp
        ; các chương trình con để ở đây (nếu có)
            END MAIN      ; kết thúc toàn bộ chương trình
```

TẠI SAO ?

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Khung chương trình dịch ra file *.COM

```
        .Model  Tiny
        .Code
        Org 100h
Start:  jmp Continue
        ; định nghĩa các biến và hằng để tại đây
Continue:
        Main    Proc
                ; Các lệnh của chương trình chính để tại đây

                INT 20H         ;  dùng ngắt 20H kết thúc chương trình và trả quyền về
                                ; cho hệ điều hành DOS
                Main    Endp
        ; các chương trình con để ở đây (nếu có)
        END     Start           ; kết thúc toàn bộ chương trình
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

*Ví dụ 1. Chương trình dạng *.EXE*

```
; Chương trình in ra màn hình lời chào: Hello World!.
        .Model Small
        .Stack 100H
        .Data
; khai báo các biến và hằng ở đây
        CRLF  DB  13, 10, '$'          ; xuống dòng và cấp dòng mới
        MSG   DB  'Hello World!$'
              .Code
              MAIN    Proc
                                       ; khởi tạo đoạn dữ liệu
              MOV AX, @Data            ; nạp địa chỉ đoạn dữ liệu vào AX
              MOV DS, AX               ; nạp địa chỉ đoạn dữ liệu vào DS
```

```
MOV AH, 9
LEA DX, CRLF        ; nạp địa chỉ CRLF vào DX
INT 21H             ;xuống dòng

MOV AH, 09
LEA DX, MSG          ; nạp địa chỉ thông điệp vào DX
INT 21H              ; In chuỗi ký tự ra màn hình dùng hàm 09 của ngắt 21H

MOV AH, 4CH         ; kết thúc chương trình dạng *.EXE  và trả quyền điều
INT 21H              ; khiển về cho hệ điều hành bằng hàm 4Ch ngắt 21h

MAIN    Endp
END     MAIN
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

➢ *Ví dụ 2. Chương trình dạng *.COM*

```
; Chương trình in ra màn hình lời chào: Welcome to PTIT !
                .Model   Tiny
                .Code
                .Org 100h
START:     jmp Continue
                ; khai báo các biến và hằng ở đây
          CRLF  DB  13, 10, '$'          ; xuống dòng và cấp dòng mới
          MSG   DB  'Welcome to PTIT !$'

                Continue:
                                Main      Proc

                                MOV AH, 9
                                LEA DX, CRLF        ; nạp địa chỉ CRLF vào DX
                                INT 21H             ;xuống dòng
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

➢ *Ví dụ 2. Chương trình dạng *.COM*

```
; Chương trình in ra màn hình lời chào: Welcome to PTIT !
                .Model   Tiny
                .Code
                .Org 100h
START:     jmp Continue
                ; khai báo các biến và hằng ở đây
          CRLF  DB  13, 10, '$'          ; xuống dòng và cấp dòng mới
          MSG   DB  'Welcome to PTIT !$'

                Continue:
                                Main      Proc

                                MOV AH, 9
                                LEA DX, CRLF        ; nạp địa chỉ CRLF vào DX
                                INT 21H             ;xuống dòng
```

```
MOV AH, 09
LEA DX, MSG      ; nạp địa chỉ  lời chào vào DX
INT 21H          ; In chuỗi ký tự ra màn hình dùng hàm 09 của ngắt 21H

 INT 20H         ; kết thúc chương trình dạng *.COM  và trả quyền điều
                 ; khiển về cho hệ điều hành bằng ngắt 20h

Main    Endp
END     START    ; kết thúc toàn bộ chương trình
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các cấu trúc điều khiển

➢ Cấu trúc tuần tự:

- ✓ Là cấu trúc thông dụng và đơn giản nhất.
- ✓ Trong cấu trúc tuần tự các lệnh được sắp xếp tuần tự, lệnh nọ kế tiếp sau lệnh kia. Sau khi thực hiện xong lệnh cuối cùng của cấu trúc thì chương trình cũng phải hoàn tất.
- ✓ Ví dụ: Kiểm tra AX, nếu AX > 0 thì gán C

Cú pháp:

Lệnh 1

Lệnh 2

……

……

Lệnh n

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các cấu trúc điều khiển

➢ Cấu trúc tuần tự:

Ví dụ: Các thanh ghi CX và BX chứa giá trị của biến c và b. hãy tính giá trị của biểu thức a=2*(c+b) và chứa kết quả trong thanh ghi AX.

Giải:

```
        XOR AX, AX      ; AX=0
        ADD AX, BX      ; cộng thêm b
        ADD AX, CX      ; cộng thêm c
        SHL  AX, 1      ; nhân đôi kết quả trong AX
RA:                     ;lối ra của cấu trúc
```

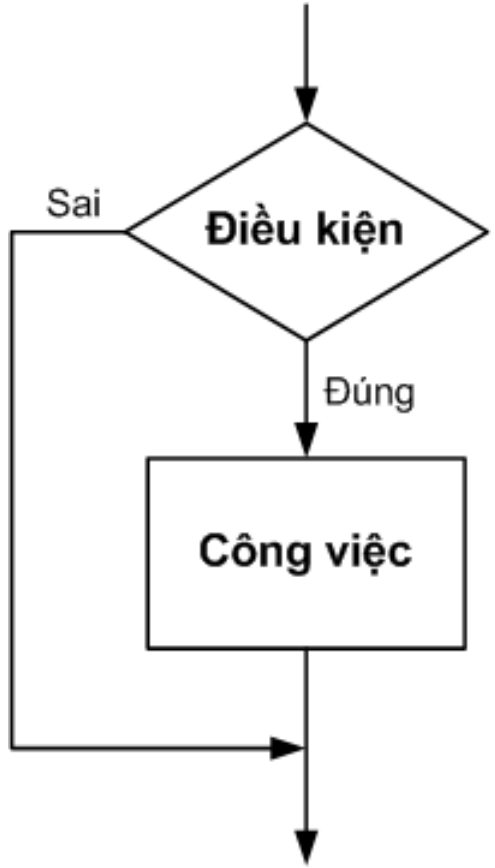
Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các cấu trúc điều khiển

➢ Cấu trúc điều khiển kiểu rẽ nhánh IF…THEN

Ngữ pháp: **IF <điều kiện> THEN <công việc>**

Nếu điều kiện thoả mãn thì công việc được thực hiện, nếu không thoả mãn công việc sẽ bị bỏ qua.



Ví dụ: Hãy gán cho BX giá trị tuyệt đối của AX

```
        CMP  AX, 0      ; AX < 0 ?
        JNL  GAN        ; không, gán luôn
        NEG  AX         ; Đảo dấu rồi gán
GAN:    MOV BX, AX      ;
RA:                     ; lối ra của cấu trúc
```

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

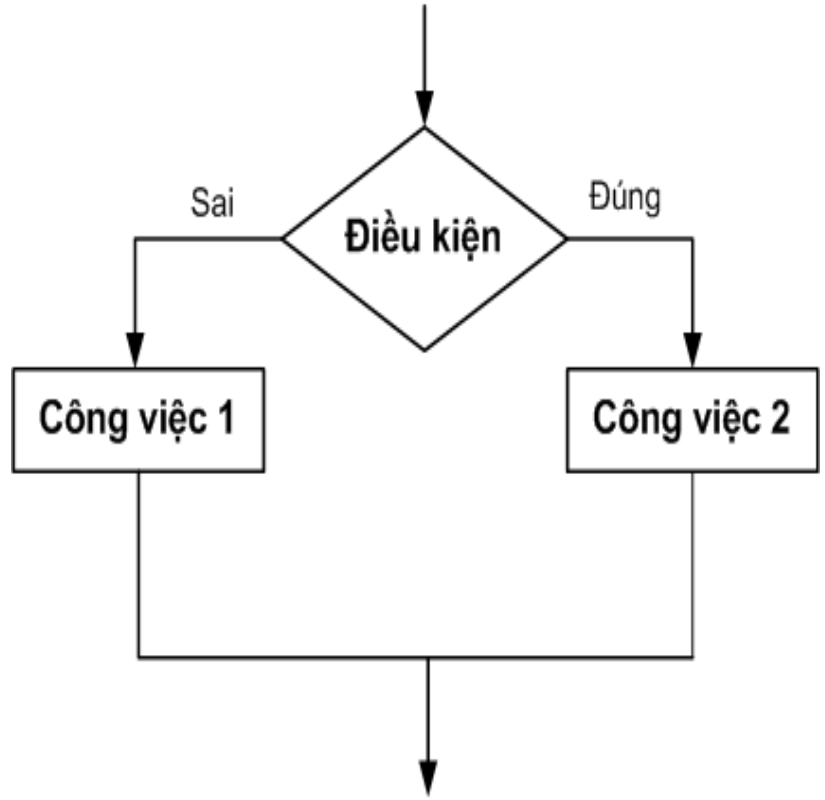
Các cấu trúc điều khiển

➢ Cấu trúc điều khiển kiểu rẽ nhánh IF…THEN….ELSE

Ví dụ: Kiểm tra AX, nếu AX > 0 thì gán CX = 0

nếu AX < 0 thì gán CX = 1

```
        OR AX, AX       ; AX > 0 ?
        JNS Duong       ; đúng
        MOV CX, 1       ; không đúng  CL ← 1
        JMP RA          ; nhảy qua nhánh kia
Duong:  MOV CX, 0       ; CL ← 0
RA:
```



JNS: short jump if not sign (nhảy nếu dương)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Các cấu trúc điều khiển

➢ Cấu trúc điều khiển kiểu rẽ nhiều nhánh : CASE

Ngữ pháp:

CASE <biểu thức>

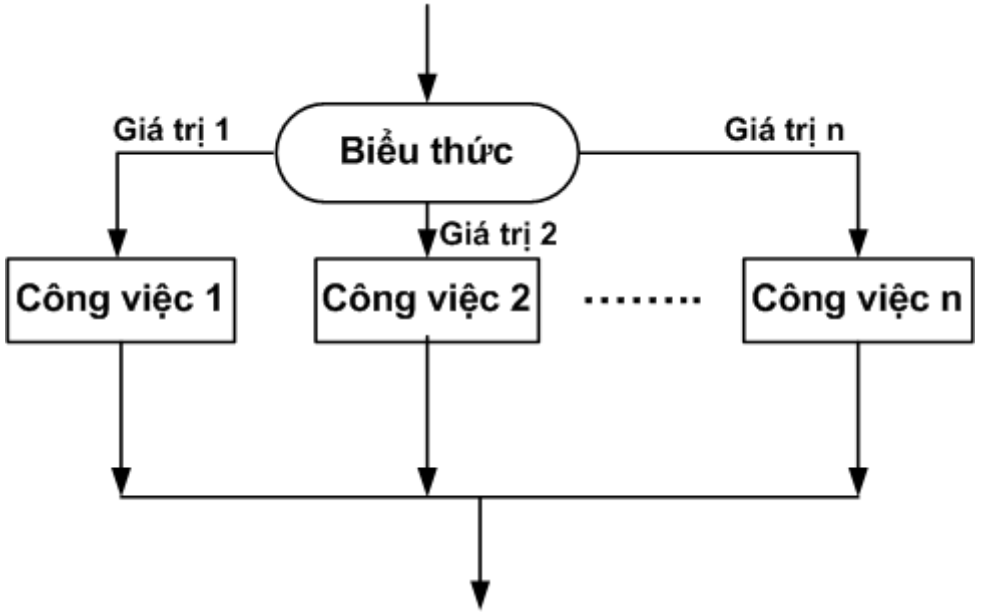
Giá trị 1: công việc 1

Giá trị 2: công việc 2

…….: ……..

Giá trị n: công việc n

END CASE



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Ví dụ: Gán giá trị cho CX theo qui tắc:

- Nếu AX<0 thì CX= -1
- Nếu AX=0 thì CX= 0
- Nếu AX>0 thì CX= 1

Giải:

```
                CMP AX, 0       ; kiểm tra dấu của AX
                JL  AM          ; AX < 0
                JE KHONG        ; AX = 0
                JG DUONG        ; AX > 0
AM:             MOV CX, -1
                JMP RA
DUONG:          MOV CX, 1
                JMP RA
KHONG:          XOR CX, CX
RA:                             ; lối ra của cấu trúc
```

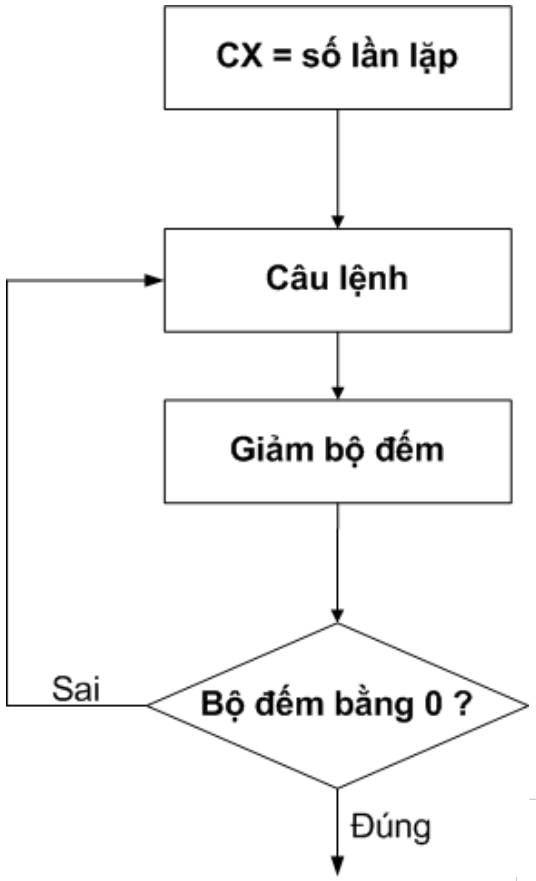
Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

➢ Cấu trúc điều khiển kiểu lặp : For…do

Ngữ pháp: **FOR <số lần lặp> DO <công việc>**

Ví dụ: hiển thị chữ "M" lên màn hình 10 lần. Sử dụng lệnh LOOP. Số lần lặp gán cho CX

```
        MOV CX,10
        MOV AH,2
        MOV DL,'M'
Hien:   INT 21H
        LOOP Hien
```



Chú ý: ngắt 21H hàm 02: viết 1 ký tự ra màn hình

Vào: AH=2

DL = ký tự được ghi ra màn hình

Ra: Sau khi thực thi AL = DL.

Ký tự được in ra màn hình máy tính

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

➢ Cấu trúc điều khiển kiểu lặp : Repeat…until

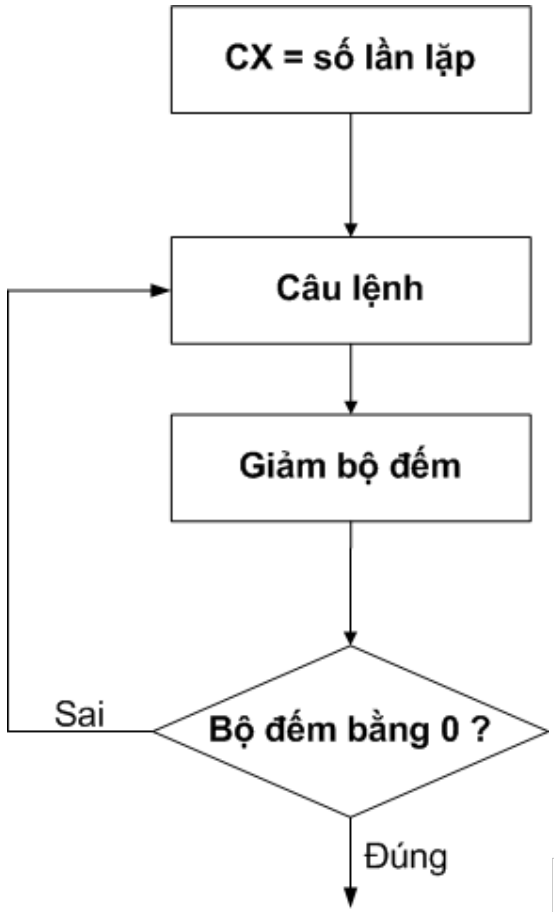
Công việc được thực hiện trước (ít nhất 1 lần). Điều kiện được kiểm tra sau đó.

Ngữ pháp: **REPEAT <công việc> UNTIL <điều kiện>**

Ví dụ: đọc ký tự từ bàn phím cho tới khi gặp phím "$" thì thôi .

Giải:

```
        MOV AH, 01      ; hàm đọc ký tự từ bàn phím
TIEP:   INT 21H         ; đọc 1 ký tự
        CMP AL, '$'     ; đọc được ký tự '$' ?
        JNE  TIẾP       ; chưa, đọc tiếp.
RA:                     ; lối ra của cấu trúc
```



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

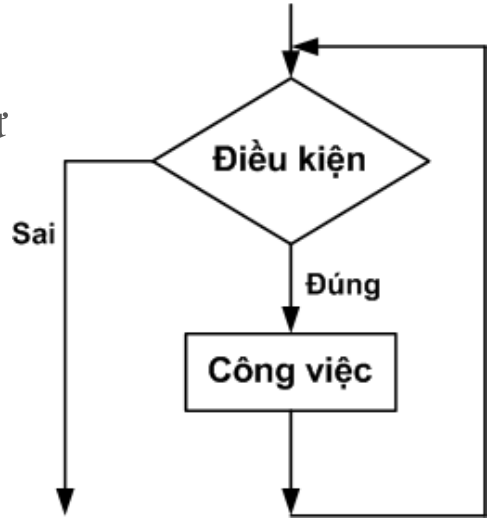
➢ Cấu trúc điều khiển kiểu lặp : WHILE…DO

Ngữ pháp: **WHILE <điều kiện> DO <công việc>**

Điều kiện được kiểm tra đầu tiên, công việc được thực hiện sau đó và lặp đi lặp lại chừng nào điều kiện còn đúng.

Ví dụ: Đếm số ký tự đọc được từ bàn phím cho tới khi gặp phím "CR" thì thôi .

```
        XOR CX, CX      ; Tổng số ký tự đọc được lúc đầu bằng 0
        MOV AH, 01      ; hàm đọc ký tự từ bàn phím
TIEP:   INT 21H         ; đọc 1 ký tự, AL chứa mã ký tự
        CMP AL, 13      ; đọc được ký tự 'CR' ?
        JE RA           ; đúng, thoát ra
        INC CX          ; sai, thêm 1 ký tự vào tổng
        JMP  TIẾP       ; đọc tiếp.
RA:                     ; lối ra của cấu trúc
```



Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

Chương trình con

(tham khảo tài liệu)

Truyền tham số cho chương trình con

* Truyền tham số qua thanh ghi
* Truyền tham số qua ô nhớ-biến
* truyền tham số qua ô nhớ có địa chỉ do thanh ghi chỉ ra
* Truyền tham số qua ngăn xếp

(tham khảo tài liệu)

Chương 4: Lập trình hợp ngữ với bộ vi xử lý 8086/8088

4.5 Một số ví dụ và bài tập lập trình hợp ngữ

Bài 1: Lập trình đọc và hiển thị thời gian của hệ thống lên màn hình

Bài 2: Viết chương trình xác định loại máy tính đang sử dụng là loại máy tính PC, PCJR, PC/XT, PC/AT hay PS/2 bằng cách kiểm tra nội dung của ô nhớ trong ROM BIOS tại địa chỉ F000:FFFE. Nếu byte nhớ này có giá trị hex là

F8: máy PS/2 model 70 and 80

FA: máy PS/2 model 30

FB: máy PC/XT (later model, 101 keyboard)

FC: máy PC/AT

FD: máy PCJR

FE: máy PC/XT (early model)

FF: máy PC classic

Bài 3: Viết chương trình nhập vào từ bàn phím một ký tự thường và in ra màn hình ký tự hoa.

(Tham khảo tài liệu)