HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG MÔN

Kiến trúc máy tính

CHƯƠNG 4 – BỘ VI XỬ LÝ

INTEL 8086/8088

Giảng viên:

Điện thoại/E-mail:

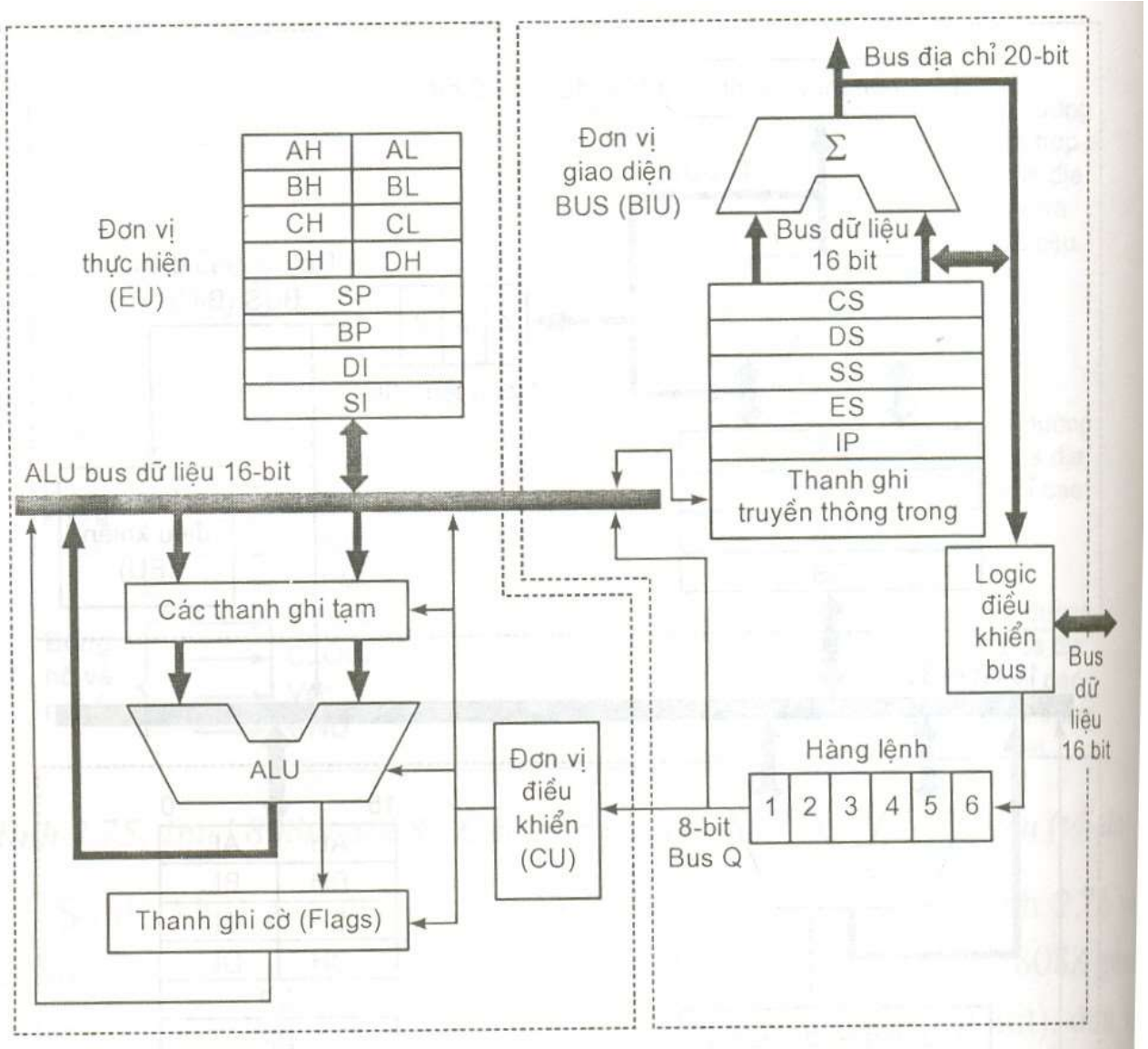
Bộ môn: Khoa học máy tính - Khoa CNTT1

NỘI DUNG

A. Kiến trúc bên trong của 8086/8088
   1. Sơ đồ khối
   2. Các đơn vị chức năng của 8088/8086
   3. Các thanh ghi của 8086/8088
   4. Phân đoạn bộ nhớ trong 8086/8088

B. Tập lệnh của 8088/8086
   5. Khái niệm về lệnh và cách mã hoá lệnh
   6. Các chế độ địa chỉ của vi xử lý 8086/8088
   7. Phân loại tập lệnh của vi xử lý
   8. Mô tả tập lệnh của 8086/8088

1. Sơ đồ khối vi xử lý 8086/8088



2. Các đơn vị chức năng của 8088/8086

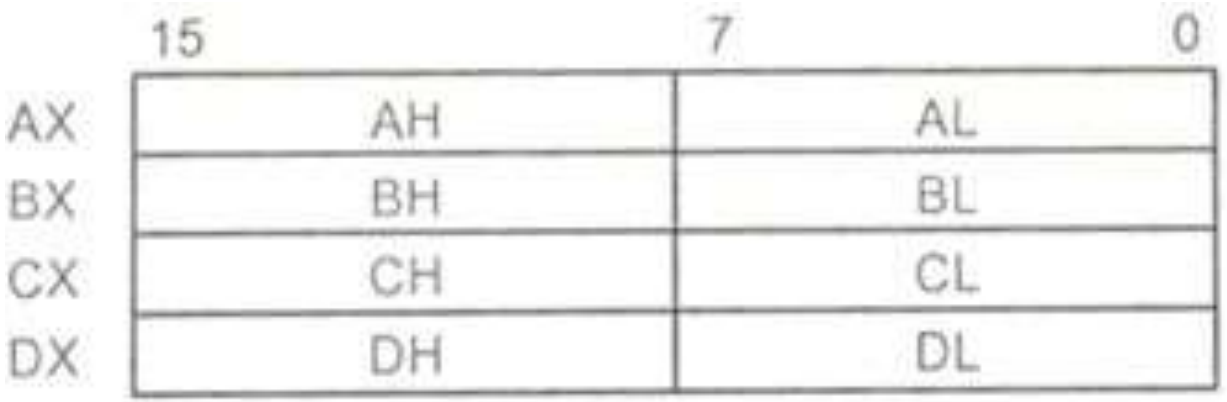
- Đơn vị giao tiếp bus BIU (Bus Interface Unit)
  - Điều khiển bus hệ thống: đưa địa chỉ ra bus và trao đổi dữ liệu với bus
    - Đưa ra địa chỉ
    - Đọc mã lệnh từ bộ nhớ
    - Đọc/ghi dữ liệu từ/vào bộ nhớ hoặc cổng vào/ra
  - Các khối:
    - Bộ cộng để tính địa chỉ
    - 4 thanh ghi đoạn 16-bit: CS, DS, SS, ES
    - Bộ đếm chương trình/con trỏ lệnh 16-bit (PC/IP)
    - Hàng đợi lệnh IQ (4 bytes trong 8088 và 6 bytes trong 8086)
    - Logic điều khiển bus

2. Các đơn vị chức năng của 8088/8086

- Đơn vị thực hiện EU (Execution Unit)
  - Chức năng: EU nhận lệnh & dữ liệu từ BIU để xử lý. Kết quả xử lý lệnh được chuyển ra bộ nhớ hoặc thiết bị I/O thông qua BIU.
  - Các khối:
    - ALU
    - CU
    - 8 thanh ghi 16-bit: AX, BX, CX, DX, SP, BP, SI, DI
    - Thanh ghi cờ FR
- Bus trong (Internal Bus): liên kết BIU và EU
  - 16-bit A-BUS trong 8088
  - 16-bit ALU-BUS trong 8086

3. Các thanh ghi của 8086/8088

- Các thanh ghi đa năng:
  - 4 thanh ghi 16 bits:
    - AX: Thanh ghi tổng, thường dùng để lưu kết quả
    - BX: Thanh ghi cơ sở, thường dùng chứa địa chỉ ô nhớ
    - CX: Thanh ghi đếm, thường dùng làm con đếm cho các lệnh lặp
    - DX: Thanh ghi dữ liệu
  - Hoặc 8 thanh ghi 8 bits: AH AL, BH, BL, CH, CL, DH, DL

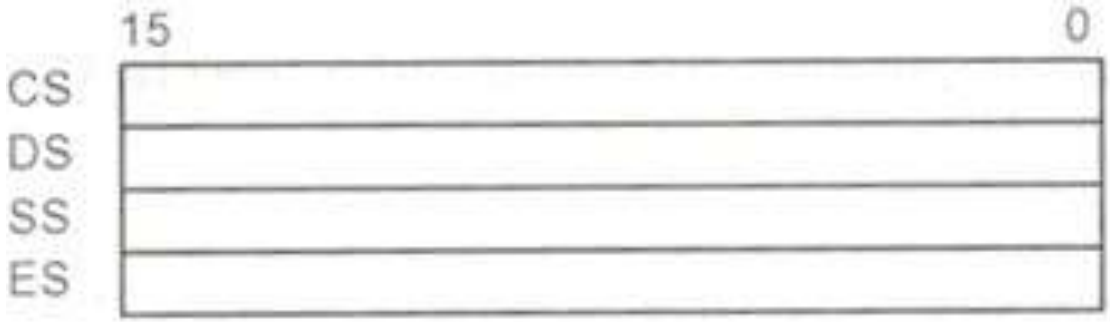


3. Các thanh ghi của 8086/8088

- Các thanh ghi con trỏ và chỉ số:
  - SP (Stack Pointer): con trỏ ngăn xếp. SP luôn chứa địa chỉ đỉnh ngăn xếp
  - BP (Base Pointer): Con trỏ cơ sở - sử dụng với đoạn ngăn xếp
  - SI (Source Index): Thanh ghi chỉ số nguồn. SI thường dùng chứa địa chỉ ô nhớ nguồn trong các thao tác chuyển dữ liệu
  - DI (Destination Index): Thanh ghi chỉ số đích. DI thường dùng chứa địa chỉ ô nhớ đích trong các thao tác chuyển dữ liệu

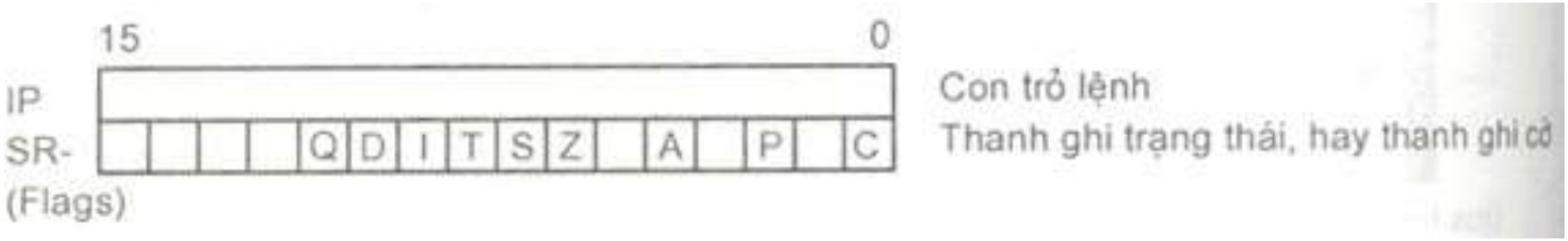
3. Các thanh ghi của 8086/8088

- Các thanh ghi đoạn:
  - CS (Code Segment): Thanh ghi đoạn mã. CS chứa địa chỉ bắt đầu đoạn mã
  - DS (Data Segment): Thanh ghi đoạn dữ liệu. DS chứa địa chỉ bắt đầu đoạn dữ liệu
  - SS (Stack Segment): Thanh ghi đoạn ngăn xếp. SS chứa địa chỉ bắt đầu đoạn ngăn xếp
  - ES (Extra Segment): Thanh ghi đoạn dữ liệu mở rộng. ES chứa địa chỉ bắt đầu đoạn dữ liệu mở rộng.



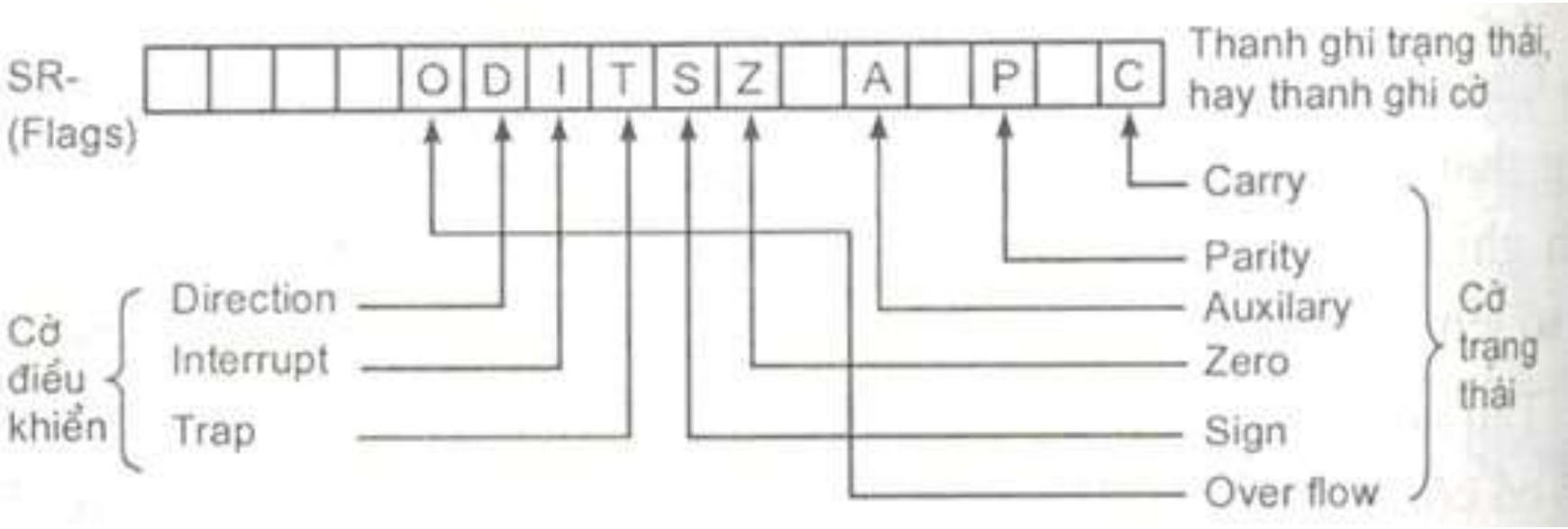
3. Các thanh ghi của 8086/8088

- Con trỏ lệnh và thanh ghi cờ:
  - IP (Instruction Pointer): Con trỏ lệnh (còn gọi là bộ đếm chương trình PC). IP luôn chứa địa chỉ của lệnh tiếp theo sẽ được thực hiện;
  - FR (Flag Register) hoặc SR (Status Register): Thanh ghi cờ hoặc thanh ghi trạng thái.
    - Cờ trạng thái: Các bit của FR lưu các trạng thái của kết quả phép toán ALU thực hiện
    - Cờ điều khiển: trạng thái của tín hiệu điều khiển.



3. Các thanh ghi của 8086/8088

❖ Các bit của thanh ghi cờ:

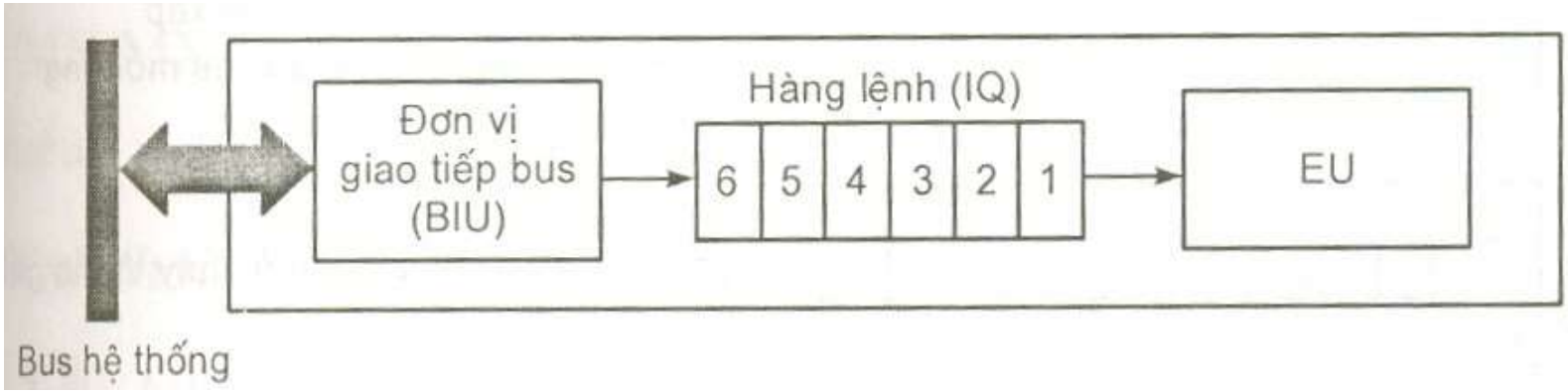


3. Các thanh ghi của 8086/8088

- Các cờ trạng thái:
  - C (Carry): cờ nhớ. C=1→ có nhớ; C=0→ không nhớ
  - A (Auxiliary): cờ nhớ phụ. A=1→ có nhớ phụ; A=0→ không nhớ phụ
  - P (Parity): cờ chẵn lẻ. P=1 khi tổng số bit 1 trong kết quả là lẻ, P=0 khi tổng số bit 1 trong kết quả là chẵn
  - O (Overflow): cờ tràn. O=1 khi kết quả bị tràn
  - Z (Zero): cờ zero. Z=1 khi kết quả bằng 0; ngược lại Z=0
  - S (Sign): cờ dấu. S=1 khi kết quả âm; S=0 khi kết quả không âm
- Các cờ điều khiển:
  - D (Direction): cờ hướng, chỉ hướng tăng giảm địa chỉ với các lệnh chuyển dữ liệu. D=0 → địa chỉ tăng. D=1 → địa chỉ giảm.
  - T (Trap/Trace): cờ bẫy/lần vết, được dùng khi gỡ rối chương trình. T=1 → CPU ở chế độ chạy từng lệnh
  - I (Interrupt): cờ ngắt. I=1 → cho phép ngắt; I=0 → cấm ngắt

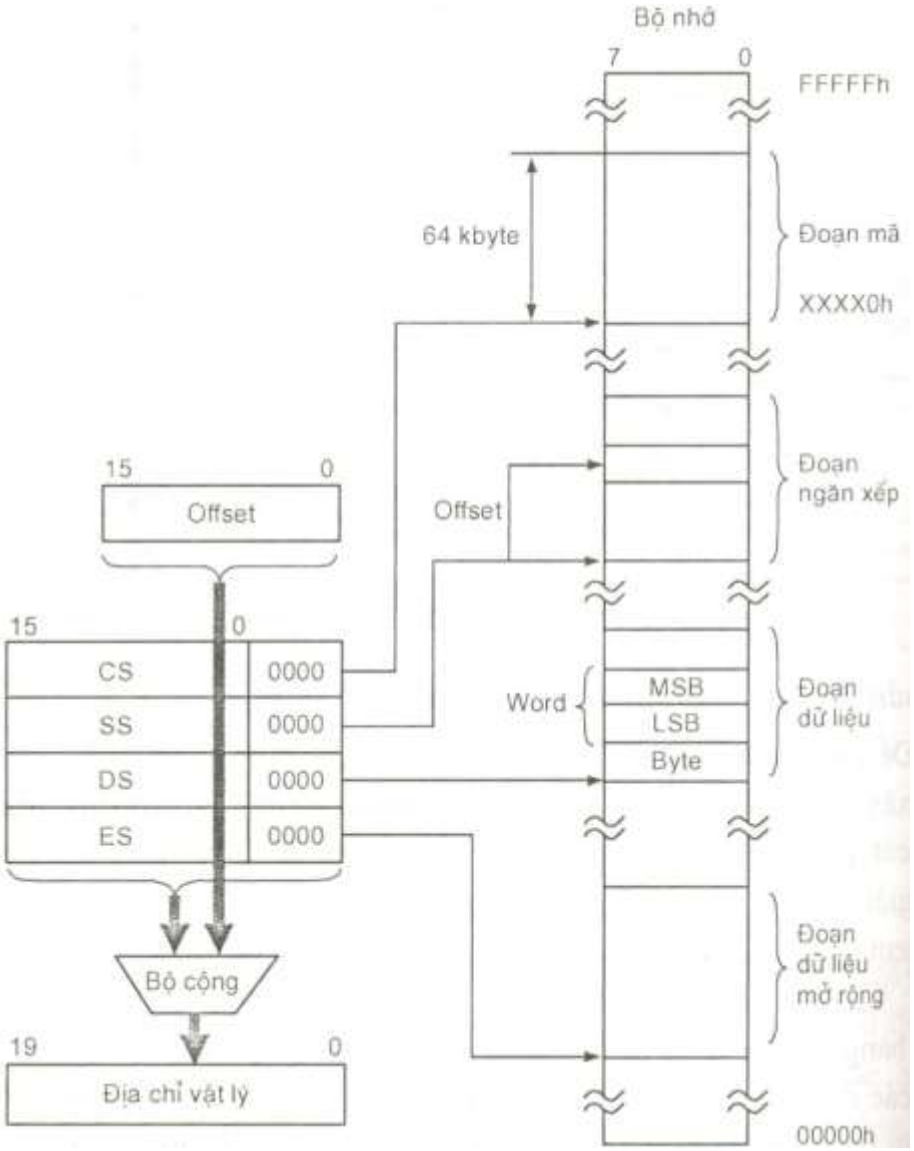
Hàng đợi lệnh IQ

- Hàng đợi lệnh IQ (Instruction Queue):
  - Chứa lệnh đọc từ bộ nhớ cho EU thực hiện.
  - Trong 8088, IQ có 4 bytes, còn trong 8086, IQ có 6 bytes.
  - IQ là một thành phần quan trọng của cơ chế ống lệnh giúp tăng tốc độ xử lý lệnh.



4. Phân đoạn bộ nhớ trong 8086/8088

- VXL 8088/8086 sử dụng 20 bit để địa chỉ hoá bộ nhớ:
  - Tổng dung lượng tối đa có thể địa chỉ hoá của bộ nhớ là $2^{20} = 1MB$;
  - Địa chỉ được đánh từ 00000h đến FFFFFh.



4. Phân đoạn bộ nhớ trong 8086/8088

- Bộ nhớ được chia thành các đoạn (segment):
  - Các thanh ghi đoạn (CS, DS, SS, ES) trỏ đến địa chỉ bắt đầu của các đoạn
  - Vị trí của ô nhớ trong đoạn được xác định bằng địa chỉ lệch Offset: 0000h-FFFFh
  - Địa chỉ logic đầy đủ của một ô nhớ là Segment:Offset
- Địa chỉ vật lý 20-bit của một ô nhớ được xác định bằng phép cộng giữa địa chỉ đoạn 16-bit được dịch trái 4 bít (nhân với 16) và địa chỉ lệch 16-bit.
  - VD: CS:IP chỉ ra địa chỉ lệnh sắp thực hiện trong đoạn mã. Nếu CS=F000h và IP=FFF0h thì:
  - CS:IP ~ F000h x 16 + FFF0h = F0000h + FFF0h = FFFF0h

5. Khái niệm về lệnh và cách mã hoá lệnh

- Lệnh (instruction) là gì?
  - Là một từ nhị phân
  - Lệnh được lưu trữ trong bộ nhớ
  - Lệnh được nạp vào CPU để thực hiện
  - Mỗi lệnh có một nhiệm vụ cụ thể
  - Các nhóm lệnh thông dụng: vận chuyển dữ liệu, điều khiển chương trình, tính toán, vv.
- Các pha (phase) chính thực hiện lệnh:
  - Đọc lệnh (IF: Instruction Fetch)
  - Giải mã lệnh (ID: Instruction Decode)
  - Thực hiện lệnh (EX: Instruction Execution)

5. Khái niệm về lệnh và cách mã hoá lệnh

- Chu kỳ lệnh (instruction cycle)
  - Là khoảng thời gian CPU thực hiện xong 1 lệnh
  - Mỗi pha của lệnh gồm một số chu kỳ máy
  - Mỗi chu kỳ máy gồm một số chu kỳ nhịp đồng hồ
  - Một CK lệnh có thể gồm:
    - Chu kỳ đọc lệnh
    - Chu kỳ đọc bộ nhớ (dữ liệu)
    - Chu kỳ ghi bộ nhớ (dữ liệu)
    - Chu kỳ đọc I/O (dữ liệu)
    - Chu kỳ ghi I/O (dữ liệu)
    - Chu kỳ chấp nhận ngắt
    - Bus rỗi

5. Khái niệm về lệnh và cách mã hoá lệnh

- Dạng lệnh
  - Dạng tổng quát của lệnh: 2 thành phần: mã lệnh và địa chỉ của các toán hạng
  - Độ dài của từ lệnh: 8, 16, 24, 32 và 64 bit.
  - Lệnh của 8086/8088 có thể có độ dài 1-6 byte

Opcode	Operands
Mã lệnh	Các toán hạng

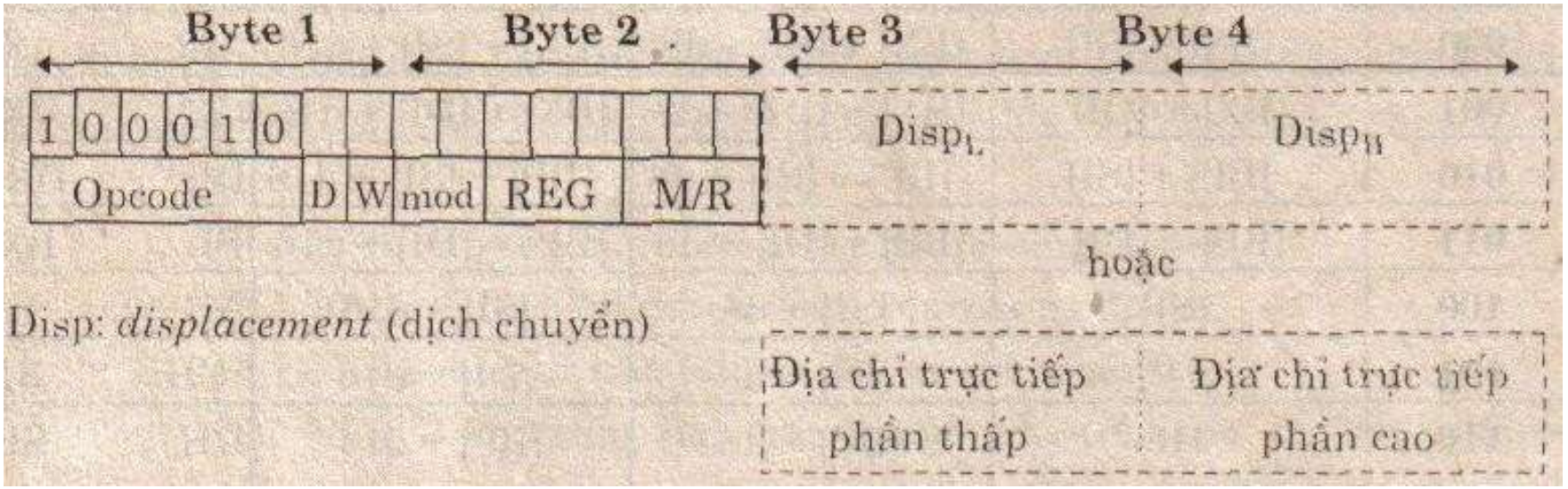
Mã lệnh	Đích, Gốc
MOV	AX, 100

AX ← 100

5. Khái niệm về lệnh và cách mã hoá lệnh

❖ Mã hoá lệnh

- Opcode: mã lệnh gồm 6 bít; Mã lệnh của MOV là 100010
- D: bít 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;
- W: bít chỉ độ rộng toán hạng; W=0: toán hạng 1 byte; W=1: toán hạng 2 bytes



5. Khái niệm về lệnh và cách mã hoá lệnh

❖ Mã hoá lệnh

- REG: 3 bít là mã của thanh ghi toán hạng theo hướng chuyển dữ liệu D:
  - Nếu D=1, REG biểu diễn toán hạng Đích
  - Neu D=0, REG biểu diễn toán hạng Gốc

Các thanh ghi đoạn	Mã thanh ghi
CS	01
DS	11
ES	00
SS	10

Các thanh ghi		Mã thanh ghi
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

5. Khái niệm về lệnh và cách mã hoá lệnh

- Mã hoá lệnh
  - 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 8086/8088
  - $Disp_L$: khoảng dịch chuyển phần thấp
  - $Disp_H$: khoảng dịch chuyển phần cao.

MOD / R/M	00	01	10	11	
				W=0	W=1
000	[BX]+[SI]	[BX]+[SI]+d8	[BX]+[SI]+d16	AL	AX
001	[BX]+[DI]	[BX]+[DI]+d8	[BX]+[DI]+d16	CL	CX
010	[BP]+[SI]	[BP]+[SI]+d8	[BP]+[SI]+d16	DL	DX
011	[BP]+[DI]	[BP]+[DI]+d8	[BP]+[DI]+d16	BL	BX
100	[SI]	[SI]+d8	[SI]+d16	AH	SP
101	[DI]	[DI]+d8	[DI]+d16	CH	BP
110	d16	[BP]+d8	[BP]+d16	DH	SI
111	[BX]	[BX]+d8	[BX]+d16	BH	DI
	< Các chế độ bộ nhớ >			< Các chế độ thanh ghi >	

Ghi chú:
- d8: khoảng dịch chuyển, 8 bit
- d16**:** khoảng dịch chuyển, 16 bit

6. Các chế độ địa chỉ của 8086/8088

- Chế độ địa chỉ (Addressing Mode) là cách CPU tổ chức và lấy dữ liệu cho các toán hạng khi thực hiện lệnh;
- Một bộ vi xử lý có thể có nhiều chế độ địa chỉ. Vi xử lý 8086/8088 có 7 chế độ địa chỉ:

1. Chế độ địa chỉ thanh ghi (Register Addressing Mode)
2. Chế độ địa chỉ tức thì (Immediate Addressing Mode)
3. Chế độ địa chỉ trực tiếp (Direct Addressing Mode)
4. Chế độ địa chỉ gián tiếp qua thanh ghi (Register Indirect Addressing Mode)
5. Chế độ địa chỉ tương đối cơ sở (Based Plus Displacement Addressing Mode)
6. Chế độ địa chỉ tương đối chỉ số (Indexed Plus Displacement Addressing Mode)
7. Chế độ địa chỉ tương đối chỉ số cơ sở (Based Indexed Plus Displacement Addressing Mode)

6. Các chế độ địa chỉ của 8086/8088

- Chế độ địa chỉ thanh ghi:
  - Sử dụng các thanh ghi bên trong cpu như là các toán hạng để chứa dữ liệu cần thao tác.
  - Cả toán hạng gốc và đích đều là các thanh ghi
  - VD:

    mov bx, dx; bx ← dx

    mov ds, ax; ds ← ax

    add al, dl; al ← al + dl

6. Các chế độ địa chỉ của 8086/8088

- Chế độ địa chỉ tức thì:
  - Toán hạng đích là một thanh ghi hay một ô nhớ
  - Toán hạng gốc là một hằng số
  - VD:

    mov cl, 200;cl ← 200

    mov ax, 0ff0h; ax ← 0ff0h

    mov [bx], 200; chuyển 200 vào ô nhớ có địa chỉ là DS:BX

6. Các chế độ địa chỉ của 8086/8088

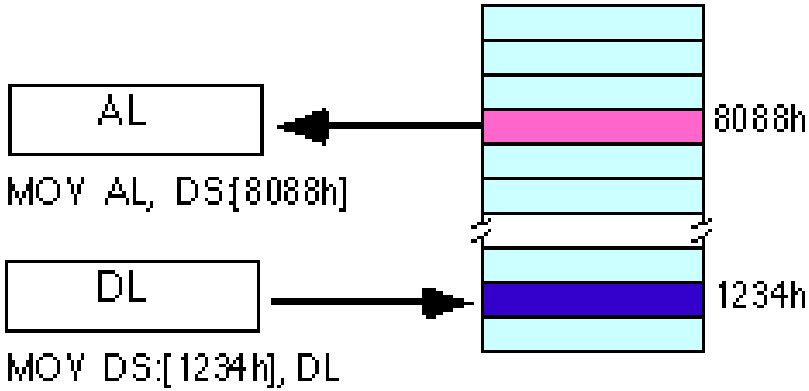
- Chế độ địa chỉ trực tiếp:
  - Một toán hạng là một hằng biểu diễn địa chỉ lệch (offset) của ô nhớ
  - Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ)
  - VD:

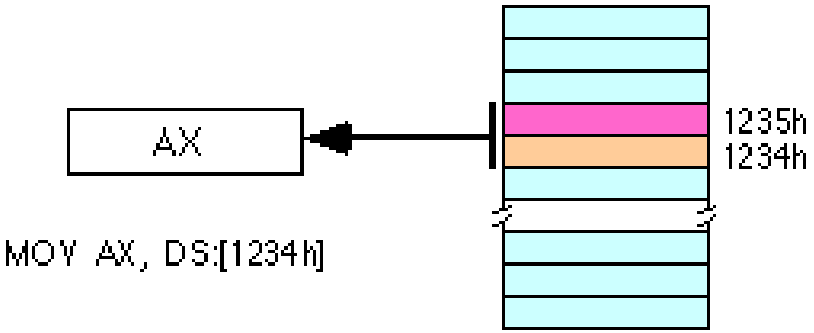
    MOV AL, [8088H]

    MOV [1234H], DL

    MOV AX, [1234H]

    DS là thanh ghi đoạn ngầm định trong chế độ địa chỉ trực tiếp.



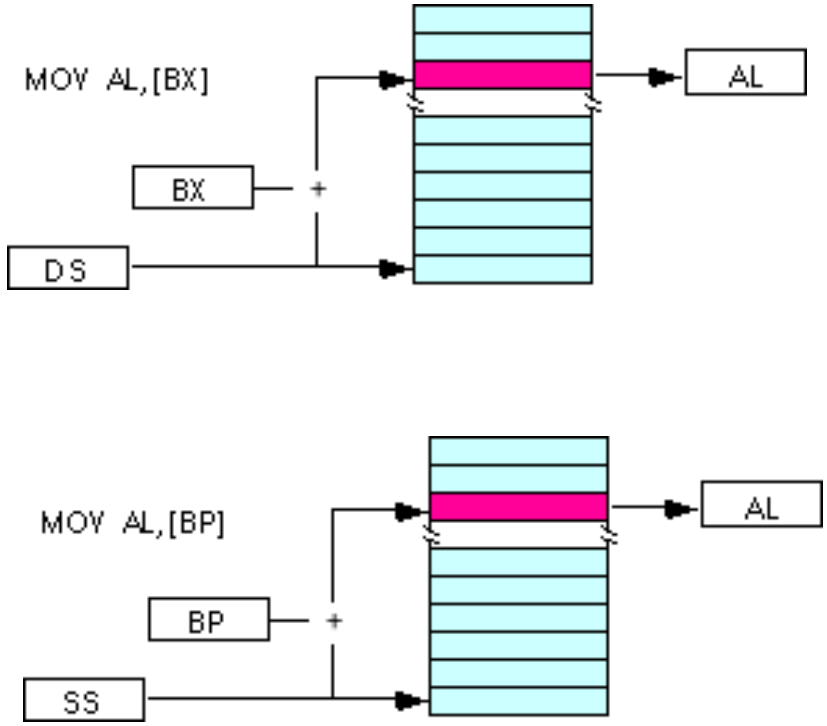


6. Các chế độ địa chỉ của 8086/8088

- Chế độ địa chỉ gián tiếp qua thanh ghi:
  - Một toán hạng là một thanh ghi chứa địa chỉ lệch của ô nhớ
  - Toán hạng còn lại có thể là thanh ghi
  - VD:

    MOV AL, [BX]; AL ← [DS:BX]

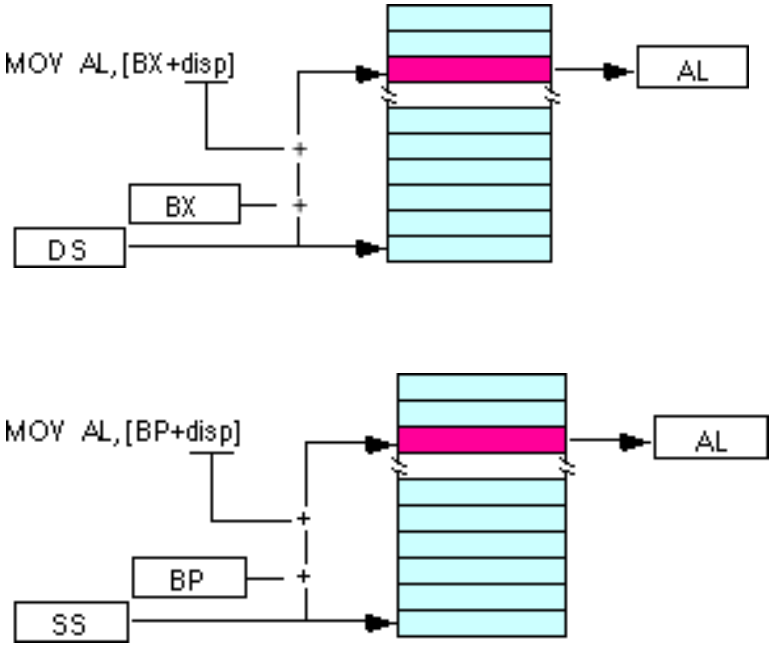
    MOV AL, [BP]; AL ← [SS:BP]



6. Các chế độ địa chỉ của 8086/8088

- Chế độ địa chỉ tương đối cơ sở:
  - Một toán hạng là đ/c 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.
    - 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 của các toán hạng trong các vùng nhớ DS và SS.
  - Toán hạng còn lại có thể là thanh ghi (ko được là ô nhớ)
  - VD: MOV AL, [BX+100]; AL ← [DS: BX+100]

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



6. Các chế độ địa chỉ của 8086/8088

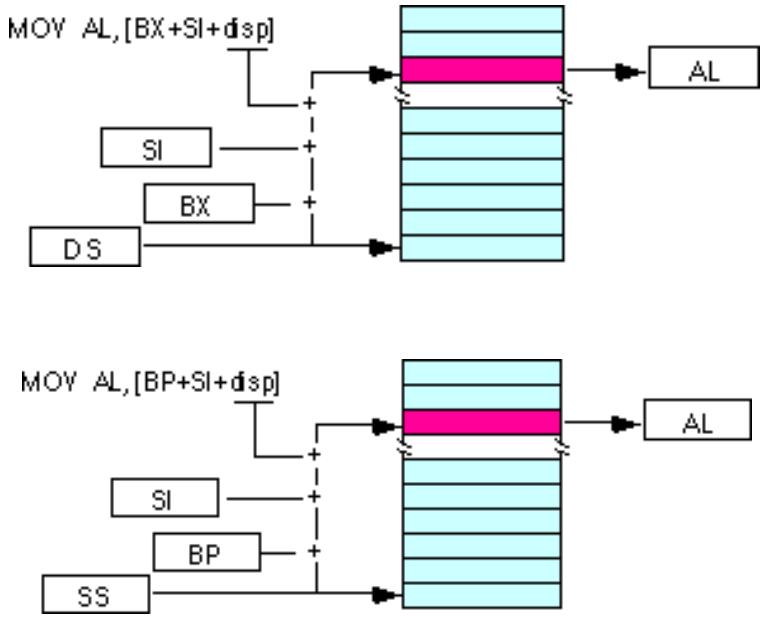
- Chế độ địa chỉ tương đối chỉ số:
  - Một toán hạng là đ/c của ô nhớ.
    - Đ/c của ô nhớ được tạo bởi việc sử dụng thanh ghi cơ sở SI hoặc DI và một hằng số.
    - Hằng số trong địa chỉ tương đối cơ sở biểu diễn các giá trị dịch chuyển (displacement) được dùng để tính địa chỉ hiệu dụng của các toán hạng trong các vùng nhớ DS.
  - Toán hạng còn lại có thể là thanh ghi (ko được là ô nhớ)
  - VD:

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

    MOV AL, [DI+200]; AL ← [DS: DI+200]

6. Các chế độ địa chỉ của 8086/8088

- Chế độ địa chỉ tương đối chỉ số cơ sở:
  - Một toán hạng là đ/c của ô nhớ.
    - Đ/c của ô nhớ được tạo bởi việc sử dụng các thanh ghi BX+SI/DI (đoạn DS) hoặc BP+SI/DI (đoạn SS) và một hằng số.
    - 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 của các toán hạng trong các vùng nhớ DS và SS.
  - Toán hạng còn lại có thể là thanh ghi (ko được là ô nhớ)
- VD:



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

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

Ánh xạ ngầm định trong các chế độ địa chỉ

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

Ánh xạ ngầm định giữa thanh ghi đoạn và lệch

- Quan hệ ngầm định giữa các thanh ghi đoạn và các 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 SS:[SI+300], AH; [SS:SI+300] ← AH

7. Phân loại tập lệnh của vi xử lý

- Tập lệnh phức hợp (CISC) và tập lệnh giảm thiểu (RISC)
  - CISC (Complex Instruction Set Computers)
    - Hỗ trợ tập lệnh phong phú -> giảm lượng mã chương trình
    - Tập lệnh lớn -> khó tối ưu hoá cho chương trình dịch
    - Các lệnh có độ dài và thời gian thực hiện khác nhau -> giảm hiệu năng của cơ chế ống lệnh (pipeline)
  - RISC (Reduced Instruction Set Computers)
    - Tập lệnh tối thiểu: số lượng lệnh, các chế độ đ/c khuôn dạng lệnh và thời gian thực hiện
    - Tăng được hiệu năng của cơ chế ống lệnh (pipeline)
    - Dễ tối ưu hoá trong chương trình dịch
    - Chương trình thường dài, cần nhiều bộ nhớ và tăng thời gian truy cập bộ nhớ

7. Phân loại tập lệnh của vi xử lý

- Phân loại tập lệnh của vi xử lý họ CISC
  - Vận chuyển DL
  - Số học nguyên và logic
  - Dịch và quay
  - Chuyển điều khiển
  - Xử lý bit
  - Điều khiển hệ thống
  - Thao tác dấu phảy động
  - Các lệnh của các đơn vị chức năng đặc biệt

8. Mô tả tập lệnh của 8086/8088

- Các lệnh vận chuyển dữ liệu: vận chuyển dữ liệu giữa:
  - thanh ghi – thanh ghi;
  - thanh ghi–ô nhớ;
  - thanh ghi – thiết bị vào ra.
- Các lệnh:
  - MOV
  - LODSB, LODSW, STOSB, STOSW
  - MOVSB, MOVSW
  - IN, OUT
- Các lệnh vận chuyển dữ liệu không ảnh hưởng đến các cờ trạng thái của thanh ghi cờ

8. Mô tả tập lệnh của 8086/8088

- Lệnh MOV:
  - Dạng lệnh: MOV Đích, Gốc; Đích ← Gốc
  - Ý nghĩa: chuyển (sao chép) dữ liệu từ Gốc sang Đích
  - Lưu ý: hai toán hạng Đích và Gốc phải tương thích về kích cỡ
  - Ví dụ: MOV AL, 100; AL ← 100

    MOV [BX], AH; [DS:BX] ← AH

    MOV DS, AX; DS ← AX

8. Mô tả tập lệnh của 8086/8088

❖ Lệnh LODSB, LODSW:

- Dạng lệnh:

	LODSB;	AL ← [DS: SI]
		SI ← SI ± 1
	LODSW;	AX ← [DS: SI]
		SI ← SI ± 2

- Ý nghĩa: Nạp nội dung ô nhớ có địa chỉ chứa trong SI thuộc đoạn DS vào thanh ghi AL/AX và tăng hoặc giảm nội dung của SI. Nếu DF = 0 → tăng, DF = 1 → giảm.
- Ví dụ:

MOV SI, 1000;	SI ← 1000
MOV [DS:SI], 200;	[DS:SI] ← 200
CLD;	DF ← 0
LODSB;	AL ← 200; SI ← SI + 1

8. Mô tả tập lệnh của 8086/8088

❖ Lệnh STOSB, STOSW:

- Dạng lệnh: STOSB; [ES: DI] ← AL
  DI ← DI ± 1
  STOSW; [ES: DI] ← AX
  DI ← DI ± 2
- Ý nghĩa: Lưu nội dung thanh ghi AL/AX vào ô nhớ có địa chỉ chứa trong DI thuộc đoạn ES và tăng hoặc giảm nội dung của DI. Nếu DF = 0 → tăng, DF = 1 → giảm.
- Ví dụ:

MOV DI, 1000;	DI ← 1000
MOV AL, 200;	AL ← 200
CLD;	DF ← 0
STOSB;	[ES:DI] ← AL ; DI ← DI + 1

8. Mô tả tập lệnh của 8086/8088

❖ Lệnh MOVSB, MOVSW:

- Dạng lệnh: MOVSB; [ES:DI] ← [DS: SI]
  SI ← SI ± 1; DI ← DI ± 1
  MOVSW; [ES:DI] ← [DS: SI]
  SI ← SI ± 2; DI ← DI ± 2
- Ý nghĩa: Chuyển nội dung ô nhớ tại địa chỉ DS:SI vào ô nhớ có địa chỉ ES:DI và tăng hoặc giảm nội dung của SI và DI. Nếu DF = 0 → tăng, DF = 1 → giảm.
- Ví dụ:

MOV SI, 1000;	SI ← 1000
MOV DI, 2000;	DI ← 2000
CLD;	DF ← 0
MOVSB;	[ES:DI] ← [DS: SI]

8. Tập lệnh - Các lệnh vận chuyển dữ liệu

- Lệnh IN:
  - Dạng lệnh: IN <thanh ghi>, <địa chỉ cổng vào>
  - Ý nghĩa: đọc dữ liệu từ <địa chỉ cổng vào> lưu vào <thanh ghi>. Có thể dùng giá trị số trực tiếp trong lệnh nếu <địa chỉ cổng vào> nằm trong khoảng 00-FFh; Nếu <địa chỉ cổng vào> lớn hơn FFh, địa chỉ cổng cần được lưu vào thanh ghi DX.
  - Ví dụ: IN AL, 0F8H; AL ← (0F8h)

    MOV DX, 02F8H

    IN AL, DX; AL ← (DX)

8. Tập lệnh - Các lệnh vận chuyển dữ liệu

❖ Lệnh OUT:

- Dạng lệnh: OUT <địa chỉ cổng ra>, <Gốc>
- Ý nghĩa: Lưu dữ liệu từ Gốc ra <địa chỉ cổng ra>. Có thể dùng giá trị số trực tiếp trong lệnh nếu <địa chỉ cổng ra> nằm trong khoảng 00-FFh; Nếu <địa chỉ cổng ra> lớn hơn FFh, địa chỉ cổng cần được lưu vào thanh ghi DX.
- Ví dụ:

```
OUT 0F8H, AL; (0F8h) ← AL
MOV DX, 02F8H
OUT DX, AL; (DX) ← AL
```

8. Tập lệnh - Các lệnh số học

- Là các lệnh thực hiện các phép toán số học: cộng (ADD), trừ (SUB), nhân (MUL) và chia (DIV);
- Lệnh ADD – cộng các số nguyên:
  - Dạng lệnh: ADD <Đích>, <Gốc>; Đích ← Đích + Gốc
  - Ý nghĩa: Lấy Gốc cộng với Đích, kết quả lưu vào Đích
  - Lệnh ADD ảnh hưởng đến các cờ: C, Z, S, P, O, A
  - Ví dụ: ADD AX, BX; AX ← AX + BX

    ADD AL, 10; AL ← AL + 10

    ADD [BX], AL; [DS:BX] ← [DS:BX] + AL

8. Tập lệnh - Các lệnh số học

❖ Lệnh SUB – trừ các số nguyên:

- Dạng lệnh: SUB <Đích>, <Gốc>; Đích ← Đích - Gốc
- Ý nghĩa: Lấy Đích trừ Gốc, kết quả lưu vào Đích
- Lệnh SUB ảnh hưởng đến các cờ: C, Z, S, P, O, A
- Ví dụ: SUB AX, BX; AX ← AX - BX

  SUB AL, 10; AL ← AL - 10

  SUB [BX], AL; [DS:BX] ← [DS:BX] - AL

8. Tập lệnh - Các lệnh số học

- Lệnh MUL – nhân các số nguyên:
  - Dạng lệnh: MUL <Gốc>;
  - Gốc phải là một thanh ghi hoặc địa chỉ ô nhớ
  - Ý nghĩa:
    - Nếu Gốc là 8 bit: AX ← AL * Gốc
    - Nếu Gốc là 16 bit: DXAX ← AX * Gốc
  - Lệnh MUL ảnh hưởng đến các cờ: Z, S, P
  - Ví dụ: tính 10 * 30

```
MOV AL, 10;     AL ← 10
MOV BL, 30;     BL ← 30
MUL BL;         AX ← AL * BL
```

8. Tập lệnh - Các lệnh số học

❖ Lệnh DIV – chia các số nguyên:

- Dạng lệnh: DIV <Gốc>;
- Gốc phải là một thanh ghi hoặc địa chỉ ô nhớ
- Ý nghĩa:
  - Nếu Gốc là 8 bit: AX : Gốc; AL chứa thương và AH chứa phần dư
  - Nếu Gốc là 16 bit: DXAX : Gốc; AX chứa thương và DX chứa phần dư
- Lệnh DIV ảnh hưởng đến các cờ: Z, S, P
- Ví dụ: tính 100 : 30

```
MOV AX, 100;    AL ← 100
MOV BL, 30;     BL ← 30
DIV BL;         AX : BL; AL = 3, AH = 10
```

8. Tập lệnh - Các lệnh logic

- Các lệnh logic: NOT (phủ định), AND (và), OR (hoặc) và XOR (hoặc loại trừ). Bảng giá trị của các phép toán logic:

X	Y	NOT	AND	OR	XOR
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

8. Tập lệnh - Các lệnh logic

❖ Lệnh NOT

- Dạng: NOT <Đích>
- Ý nghĩa: Đảo các bít của toán hạng Đích
- Lệnh NOT ảnh hưởng đến các cờ: Z, S, P
- VD:

```
MOV AL, 80H;        80H = 1000 0000B
NOT AL;             7FH = 0111 1111B
```

8. Tập lệnh - Các lệnh logic

- Lệnh AND
  - Dạng: AND <Đích>, <Gốc>
  - Ý nghĩa: Nhân các cặp bít của 2 toán hạng Đích, Gốc, kết quả chuyển vào Đích
  - Lệnh AND ảnh hưởng đến các cờ: Z, S, P
  - VD: AND có thể được dùng để xoá một hoặc một số bit

    Xoá bít thứ 3 của thanh ghi AL (0-7)

    AND AL, F7H; F7H = 1111 0111B

    Xoá 4 bit phần cao của thanh ghi AL (0-7)

    AND AL, 0FH; 0FH = 0000 1111B

8. Tập lệnh - Các lệnh logic

- Lệnh OR
  - Dạng: OR <Đích>, <Gốc>
  - Ý nghĩa: Cộng các cặp bít của 2 toán hạng Đích, Gốc, kết quả chuyển vào Đích
  - Lệnh OR ảnh hưởng đến các cờ: Z, S, P
  - VD: OR có thể được dùng để lập một hoặc một số bit

    lập bít thứ 3 của thanh ghi AL (0-7)

    OR AL, 08H; 08H = 0000 1000B

    lập bít thứ 7 của thanh ghi AL (0-7)

    OR AL, 80H; 80H = 1000 0000B

8. Tập lệnh - Các lệnh logic

❖ Lệnh XOR

- Dạng: XOR <Đích>, <Gốc>
- Ý nghĩa: Cộng đảo các cặp bít của 2 toán hạng Đích, Gốc, kết quả chuyển vào Đích
- Lệnh XOR ảnh hưởng đến các cờ: Z, S, P
- VD: Dùng XOR để xoá nội dung của thanh ghi/ô nhớ

  xoá thanh ghi AL

  XOR AL, AL; AL ← 0

  xoá thanh ghi BX

  XOR BX, BX; BX ← 0

8. Tập lệnh - Các lệnh dịch và quay

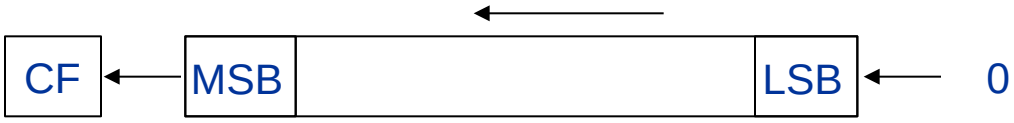
- Gồm các lệnh:
  - Dịch trái: SHL (Shift Left)
  - Dịch phải: SHR (Shift Right)
  - Quay trái: ROL (Rotate Left)
  - Quay phải: ROR (Rotate Right)
- Các lệnh dịch thường được dùng để thay cho phép nhân (dịch trái) và thay cho phép chia (dịch phải)
- Các lệnh dịch và quay còn có thể được sử dụng khi cần xử lý từng bit.

8. Tập lệnh - Các lệnh dịch và quay

❖ Lệnh dịch trái SHL

- Dạng: SHL <Đích>, 1
  SHL <Đích>, CL
- Ý nghĩa: Dịch trái một bít hoặc dịch trái số bit lưu trong thanh ghi CL nếu số bit cần dịch lớn hơn 1.
  - MSB (Most Significant Bit) chuyển sang cờ nhớ CF
  - 0 được điền vào LSB (Least Significant Bit)
  - Các bít giữa MSB và LSB được dịch sang trái 1 bit
- VD:

```
MOV AL, 08H;        0000 1000B (8)
SHL AL, 1;          0001 0000B (16)
MOV CL, 2
SHL AL, CL;         0100 0000B (64)
```

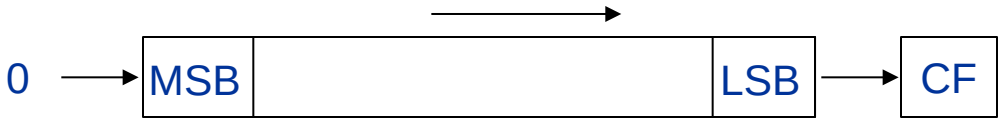


8. Tập lệnh - Các lệnh dịch và quay

❖ Lệnh dịch phải SHR

- Dạng: SHR <Đích>, 1
  SHR <Đích>, CL
- Ý nghĩa: Dịch phải một bít hoặc dịch phải số bit lưu trong thanh ghi CL nếu số bit cần dịch lớn hơn 1.
  - LSB (Least Significant Bit) chuyển sang cờ nhớ CF
  - 0 được điền vào MSB (Most Significant Bit)
  - Các bít giữa MSB và LSB được dịch sang phải 1 bit
- VD:

```
MOV AL, 80H;        1000 0000B (128)
SHR AL, 1;          0100 0000B (64)
MOV CL, 2
SHR AL, CL;         0001 0000B (16)
```

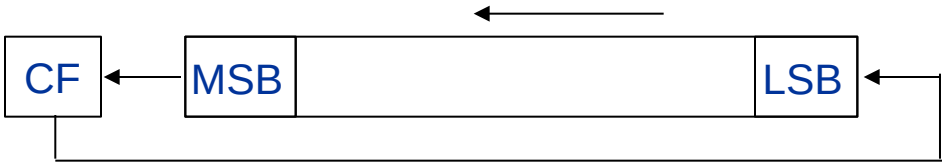


8. Tập lệnh - Các lệnh dịch và quay

❖ Lệnh quay trái ROL

- Dạng: ROL <Đích>, 1
  ROL <Đích>, CL
- Ý nghĩa: Quay trái một bít hoặc quay trái số bit lưu trong thanh ghi CL nếu số bit cần quay lớn hơn 1.
  - MSB (Most Significant Bit) chuyển sang cờ nhớ CF
  - MSB được chuyển đến LSB (Least Significant Bit)
  - Các bít giữa MSB và LSB được dịch sang trái 1 bit
- VD:

```
MOV AL, 88H;        1000 1000B
ROL AL, 1;          0001 0001B
MOV CL, 2
ROL AL, CL;         0100 0100B
```

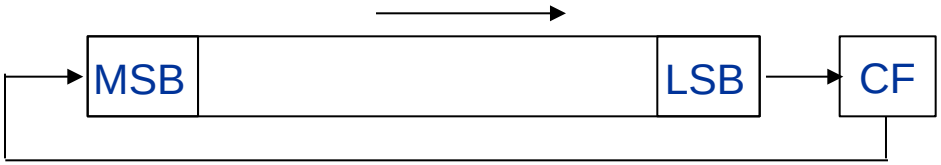


8. Tập lệnh - Các lệnh dịch và quay

❖ Lệnh quay phải ROR

- Dạng: ROR <Đích>, 1
  ROR <Đích>, CL
- Ý nghĩa: Quay phải một bít hoặc quay phải số bit lưu trong thanh ghi CL nếu số bit cần quay lớn hơn 1.
  - LSB (Least Significant Bit) chuyển sang cờ nhớ CF
  - LSB được chuyển đến MSB (Most Significant Bit)
  - Các bít giữa MSB và LSB được dịch sang phải 1 bit
- VD:

```
MOV AL, 88H;        1000 1000B
ROR AL, 1;          0100 0100B
MOV CL, 2
ROR AL, CL;         0001 0001B
```



8. Tập lệnh - Các lệnh chuyển điều khiển

- Các lệnh chuyển điều khiển (program flow control instructions) là các lệnh làm thay đổi trật tự thực hiện chương trình;
- Gồm các lệnh:
  - Lệnh nhảy không điều kiện JMP
  - Lệnh nhảy có điều kiện JE, JZ, JNE, JNZ, JL, JLE, JG, JGE, ...
  - Lệnh lặp LOOP, LOOPE, LOOPZ
  - Lệnh gọi thực hiện chương trình con CALL
  - Lệnh trở về từ chương trình con RET

8. Tập lệnh - Các lệnh chuyển điều khiển

❖ Lệnh nhảy không điều kiện JMP

- Dạng lệnh: JMP <nhãn>
- Ý nghĩa: chuyển đến thực hiện lệnh nằm ngay sau <nhãn>
- <nhãn> là một tên được đặt trước một lệnh, phân cách bằng dấu hai chấm (:). Khoảng nhảy của JMP có thể là ngắn (-128 ÷ +127), gần (-32768 ÷ +32767) và xa (sử dụng địa chỉ đầy đủ CS:IP).
- VD:

```
START:
    ADD AX, BX
    SUB BX, 1
    .....
    JMP START ; chuyển đến thực hiện lệnh nằm sau nhãn START
```

8. Tập lệnh - Các lệnh chuyển điều khiển

❖ Lệnh nhảy có điều kiện JE, JZ, JNE, JNZ, JL, JG

- Dạng lệnh:

  JE <nhãn> : nhảy nếu bằng nhau hoặc kết quả bằng 0

  JZ <nhãn> : nhảy nếu bằng nhau hoặc kết quả bằng 0

  JNE <nhãn> : nhảy nếu không bằng nhau hoặc kết quả khác 0

  JNZ <nhãn> : nhảy nếu không bằng nhau hoặc kết quả khác 0

  JL <nhãn> : nhảy nếu bé hơn

  JLE <nhãn> : nhảy nếu bé hơn hoặc bằng

  JG <nhãn> : nhảy nếu lớn hơn

  JGE <nhãn> : nhảy nếu lớn hơn hoặc bằng

- Khoảng nhảy của các lệnh nhảy có điều kiện là ngắn (-128 ÷ +127).

8. Tập lệnh - Các lệnh chuyển điều khiển

- Lệnh nhảy có điều kiện JE, JZ, JNE, JNZ, JL, JG
  - VD: viết đoạn chương trình tính tổng các số từ 1-20

```
        MOV AX, 0       ; AX chứa tổng
        MOV BX, 20      ; đặt giá trị cho biến đếm BX
START:
        ADD AX, BX      ; cộng dồn
        SUB BX, 1       ; giảm biến đếm
        JZ STOP         ; dừng nếu BX = 0
        JMP START       ; quay lại vòng lặp tiếp
STOP: ....
```

8. Tập lệnh - Các lệnh chuyển điều khiển

❖ Lệnh lặp LOOP

- Dạng lệnh: LOOP <nhãn>
- Ý nghĩa: chuyển đến thực hiện lệnh nằm ngay sau <nhãn> nếu giá trị trong thanh ghi CX khác 0. Tự động giảm giá trị của CX 1 đơn vị khi thực hiện.
- VD: viết đoạn chương trình tính tổng các số từ 1-20

```
        MOV AX, 0       ; AX chứa tổng
        MOV CX, 20      ; đặt giá trị cho biến đếm CX
START:
        ADD AX, CX      ; cộng dồn
        LOOP START      ; kiểm tra CX, nếu CX=0 → dừng
                        ; nếu CX khác 0: CX ← CX-1 và quay lại
                        ; bắt đầu vòng lặp mới từ vị trí của START
```

8. Tập lệnh - Các lệnh chuyển điều khiển

❖ Lệnh CALL và RET

- Dạng lệnh:
  - CALL <tên chương trình con>: gọi thực hiện chương trình con
  - RET : trở về từ chương trình con; thường đặt ở cuối chương trình con
- VD:

```
CALL GIAITHUA        ; gọi thực hiện chương trình con GIAITHUA
.....
; phần mã của chương trình con
GIAITHUA PROC        ; bắt đầu mã CT con
.....
   RET               ; trở về chương trình gọi
GIAITHUA ENDP        ; kết thúc mã CT con
```

8. Tập lệnh - Các lệnh xử lý bit

- Gồm nhóm các lệnh xử lý một số bít (D, C, I) của thanh ghi cờ FR;
- Các lệnh lập cờ (đặt bit cờ bằng 1)
  - STD: lập cờ hướng D
  - STC: lập cờ nhớ C
  - STI: lập cờ ngắt I
- Các lệnh xoá cờ (đặt bit cờ bằng 0)
  - CLD: xoá cờ hướng D
  - CLC: xoá cờ nhớ C
  - CLI: xoá cờ ngắt I

8. Tập lệnh - Các lệnh điều khiển hệ thống

- Gồm 2 lệnh:
  - Lệnh NOP (No Operation):
    - NOP không thực hiện nhiệm vụ cụ thể, chỉ tiêu tốn thời gian bằng 1 chu kỳ lệnh
  - Lệnh HLT (Halt)
    - HLT dừng việc thực hiện chương trình

8. Tập lệnh – Một số lệnh khác

- Lệnh tăng INC
  - Dạng: INC <Đích> ; Đích ← Đích + 1
- Lệnh giảm DEC
  - Dạng: DEC <Đích> ; Đích ← Đích – 1
- Lệnh so sánh CMP
  - Dạng: CMP <Đích>, <Gốc>
  - Ý nghĩa: Tính toán Đích - Gốc, kết quả chỉ dùng cập nhật các bít cờ trạng thái, không lưu vào Đích:

Trường hợp	C	Z	S
Đích > Gốc	0	0	0
Đích = Gốc	0	1	0
Đích < Gốc	1	0	1

8. Tập lệnh – Một số lệnh khác

- Lệnh PUSH – đẩy dữ liệu vào ngăn xếp
  - Dạng: PUSH <Gốc>
  - Ý nghĩa: Nạp Gốc vào đỉnh ngăn xếp; Gốc phải là toán hạng 2 bytes. Diễn giải:

    SP ← SP + 2 ; tăng con trỏ ngăn xếp SP

    {SP} ← Gốc ; nạp dữ liệu vào ngăn xếp
  - VD: PUSH AX
- Lệnh POP – lấy dữ liệu ra khỏi ngăn xếp
  - Dạng: POP <Đích>
  - Ý nghĩa: Lấy dữ liệu từ đỉnh ngăn xếp lưu vào Đích; Đích phải là toán hạng 2 bytes. Diễn giải:

    Đích ← {SP} ; lấy dữ liệu ra khỏi ngăn xếp

    SP ← SP - 2 ; giảm con trỏ ngăn xếp SP
  - VD: POP BX

8. Tập lệnh – Một số lệnh khác

- Lệnh NEG – đảo dấu giá trị của toán hạng
  - Dạng: NEG <Đích>
  - Ý nghĩa: Đảo dấu giá trị lưu trong Đích
  - VD: MOV AX, 1000; AX ← 1000
    NEG AX; AX ← - (AX) = -1000

- Lệnh XCHG – Tráo đổi giá trị hai toán hạng
  - Dạng: XCHG <Operand1>, < Operand2>
  - Ý nghĩa: Tráo đổi giá trị hai toán hạng <Operand1> và < Operand2>
  - VD: MOV BX, 100
    MOV AX, 200
    XCHG AX, BX; AX ← 100, BX ← 200

8. Tập lệnh – Một số lệnh khác

- Lệnh REP – lặp việc thực hiện các lệnh MOVSB, MOVSW, LODSB, LODSW, STOSB, STOSW một số lần – số lần lưu trong thanh ghi CX.
  - Dạng: REP <Lệnh cần lặp>
  - Ý nghĩa: Lặp CX lần việc thực hiện một lệnh khác
  - VD: MOV SI, 1000; Đặt địa chỉ nguồn

    MOV DI, 2000; Đặt địa chỉ đích

    MOV CX, 10; Đặt số lần lặp cho REP

    REP MOVSB; Thực hiện MOVSB 10 lần: chuyển nội dung 10 ô nhớ bắt đầu từ DS:SI sang 10 ô nhớ bắt đầu từ ES:DI

8. Tập lệnh – Một số lệnh khác

- Lệnh INT – Triệu gọi dịch vụ ngắt
  - Dạng: INT <Số hiệu ngắt>
  - Ý nghĩa: Gọi thực hiện chương trình con phục vụ ngắt tương ứng với <Số hiệu ngắt>
  - VD: MOV AH, 4Ch; Nạp hàm 4Ch
    INT 21h; Gọi ngắt DOS số 21h

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG MÔN

Kiến trúc máy tính

CHƯƠNG 4 (Tiếp) – LẬP TRÌNH HỢP NGỮ VỚI 8086/8088

Giảng viên:

Điện thoại/E-mail:

Bộ môn: Khoa học máy tính - Khoa CNTT1

NỘI DUNG

1. Giới thiệu về hợp ngữ
2. Cú pháp của chương trình hợp ngữ
3. Dữ liệu cho chương trình hợp ngữ
4. Biến và hằng
5. Khung chương trình hợp ngữ
6. Các cấu trúc điều khiển
7. Giới thiệu phần mềm mô phỏng emu8086
8. Một số ví dụ
9. Chương trình con
10. Marco
11. Giới thiệu thiết bị ảo – Đèn giao thông

3.1. Giới thiệu về hợp ngữ

- Hợp ngữ (Assembler) là ngôn ngữ lập trình bậc thấp, chỉ cao hơn ngôn ngữ máy;
- Hợp ngữ là ngôn ngữ gắn liền với các dòng vi xử lý (processor specific).
  - Các lệnh dùng trong hợp ngữ là lệnh của VXL
  - Chương trình hợp ngữ viết cho một VXL có thể không hoạt động trên VXL khác.
- Chương trình hợp ngữ khi dịch ra mã máy có kích thước nhỏ gọn, chiếm ít không gian nhớ.
- Hợp ngữ thường được sử dụng để viết:
  - Các trình điều khiển thiết bị
  - Các môđun chương trình cho vi điều khiển
  - Một số môđun trong nhân HĐH (đòi hỏi kích thước nhỏ gọn và tốc độ cao)

3.2. Cú pháp của chương trình hợp ngữ

- Trong chương trình hợp ngữ, mỗi lệnh được đặt trên một dòng – dòng lệnh;
- Lệnh có 2 dạng:
  - Lệnh thật: là các lệnh gợi nhớ của VXL
    - VD: MOV, SUB, ADD,...
    - Khi dịch, lệnh gợi nhớ được dịch ra mã máy
  - Lệnh giả: là các hướng dẫn chương trình dịch
    - VD: MAIN PROC, .DATA, END MAIN,...
    - Khi dịch, lệnh giả không được dịch ra mã máy mã chỉ có tác dụng định hướng cho chương trình dịch.
- Không phân biệt chữ hoa hay chữ thường trong các dòng lệnh hợp ngữ khi được dịch.

3.2. Cú pháp của chương trình hợp ngữ

- Cấu trúc dòng lệnh hợp ngữ:

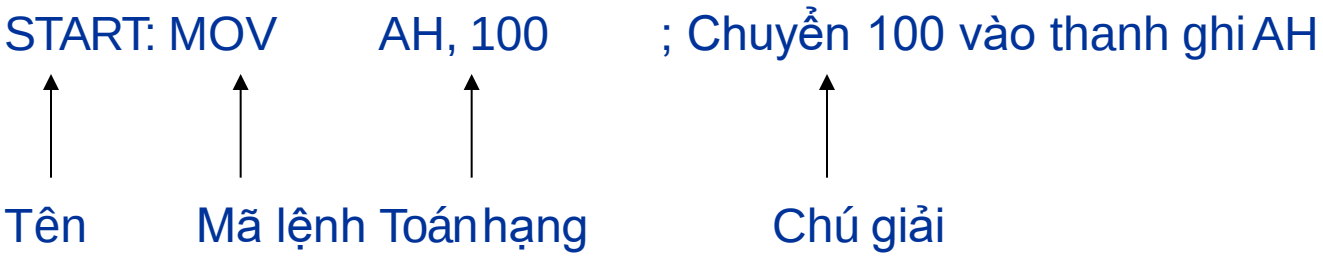
  [Tên] [Mã lệnh] [Các toán hạng] [Chú giải]

  START: MOV AH, 100 ; Chuyển 100 vào thanh ghi AH

- Các trường của dòng lệnh:
  - 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
  - Mã lệnh: có thể gồm lệnh thật và giả

3.2. Cú pháp của chương trình hợp ngữ

- Các trường của dòng lệnh:
  - Toán hạng:
    - Số lượng toán hạng phụ thuộc vào lệnh cụ thể
    - Có thể có 0, 1 và 2 toán hạng.
  - Chú giải:
    - Là chú thích cho dòng lệnh
    - Bắt đầu bằng dấu chấm phảy (;)



3.3. 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) thì thêm 0 vào đầu
    - Thêm ký hiệu H (Hexa) ở cuối
    - VD: 80H, 0F9H
  - Nhị phân: 0-1
    - Thêm ký hiệu B (Binary) ở cuối
    - VD: 0111B, 1000B
- Dữ liệu ký tự:
  - Bao trong cặp nháy đơn hoặc kép
  - Có thể dùng ở dạng ký tự hoặc mã ASCII
    - ‘A’ = 65, ‘a’ = 97

3.4. Hằng và biến

- 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>

    VD:

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

3.4. Hằng và biến

- Biến (variable):
  - Là các đại lượng có thể thay đổi giá trị
  - Các loại biến:
    - Biến đơn
    - Biến mảng
    - Biến xâu ký tự
  - Khi dịch biến được chuyển thành địa chỉ ô nhớ

3.4. Hằng và biến

❖ Định nghĩa biến đơn:

Tên biến DB Giá trị khởi đầu: Định nghĩa biến byte

Tên biến DW Giá trị khởi đầu: Định nghĩa biến word

Tên biến DD Giá trị khởi đầu: Định nghĩa biến 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 X và khởi trị 1000
```

3.4. Hằng và biến

❖ Định nghĩa biến mảng:

Tên mảng DB D/s 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
```

3.4. Hằng và biến

- 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

  str3 DB 73H, 74H, ‘r’, ‘i’, 69H, 6EH, 67H

3.5. Khung chương trình hợp ngữ

- Khai báo qui mô sử dụng bộ nhớ:

  .Model <Kiểu kích thước bộ nhớ>

- Các kiểu kích thước bộ nhớ:
  - 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.

3.5. Khung chương trình hợp ngữ

- Khai báo đoạn ngăn xếp:

  .Stack <Kích thước ngăn xếp>

  VD:

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

- Khai báo đoạn dữ liệu:

  .Data

  ;Định nghĩa các biến và hằng

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

  VD:

```
.Data
    MSG     DB      'Hello!$'
    ENTER   DB      13
    MAX     DW      1000
```

3.5. Khung chương trình hợp ngữ

❖ Khai báo đoạn mã:

```
.Code
; Các lệnh của chương trình
        .Code
        Jmp Start
        ; khai bao du lieu
        Start:
         mov AX,@Data
         mov DS, AX
           ; các lệnh của chương trình chính
        Mov AH, 4CH
        Int 21h
        End Start          ; kết thúc chương trình chính
           ; các chương trình con – nếu có
```

3.5. Khung chương trình hợp ngữ - tổng hợp

```
.Model Small
.Stack 100H
.Data
    ; khai báo các biến và hằng
.Code
    jmp Start
Start:
    ; 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

    ; kết thúc, trở về chương trình gọi dùng hàm 4CH của ngắt 21H
    MOV AH, 4CH
    INT 21H
End Start
    ; các chương trình con (nếu có)
```

3.5. Khung chương trình hợp ngữ - ví dụ

```
; Chương trình in ra thông điệp: Hello World!
.Model Small
.Stack 100H
.Data
    ; khai báo các biến và hằng
    CRLF  DB 13, 10, *         ; xuống dòng
    MSG   DB 'Hello World!$'
.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
```

3.5. Khung chương trình hợp ngữ - ví dụ

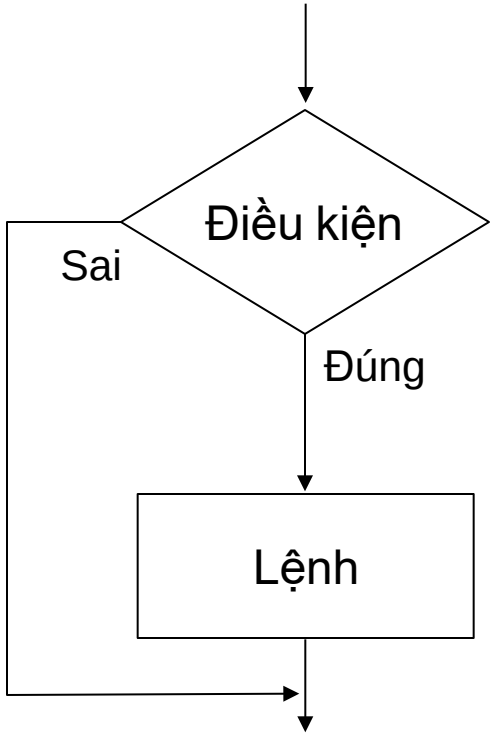
```
    ; xuống dòng
    MOV AH, 9
    LEA DX, CRLF    ; nạp địa chỉ CRLF vào DX
    INT 21H
    ; hiện lời chào dùng hàm 9 của ngắt 21H
    MOV AH, 9
    LEA DX, MSG     ; nạp địa chỉ thông điệp vào DX
    INT 21H         ; hiện thông điệp
    ; kết thúc, trở về chương trình gọi dùng hàm 4CH của ngắt 21H
    MOV AH, 4CH
    INT 21H
MAIN Endp
END MAIN
```

3.6. Các cấu trúc điều khiển

- Cấu trúc lựa chọn
  - Rẽ nhánh kiểu IF ... THEN
  - Rẽ nhánh kiểu IF ... THEN ... ELSE
  - Rẽ nhiều nhánh
- Cấu trúc lặp
  - Lặp kiểu for
  - Lặp kiểu repeat ... until

3.6. Các cấu trúc điều khiển - IF ... THEN

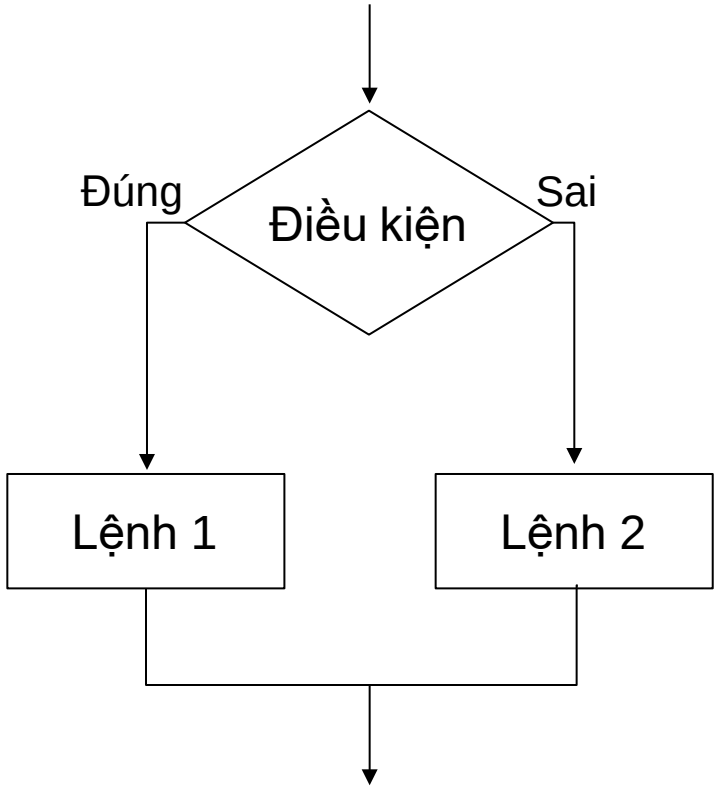
- IF điều kiện THEN thao tác
- Gán BX giá trị tuyệt đối AX
  1. CMP AX,0
  2. JNL GAN
  3. NEG AX
  4. GAN: MOV BX, AX



3.6. Các cấu trúc điều khiển - IF ... THEN ... ELSE

Gán bít dấu của AX cho CL:

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

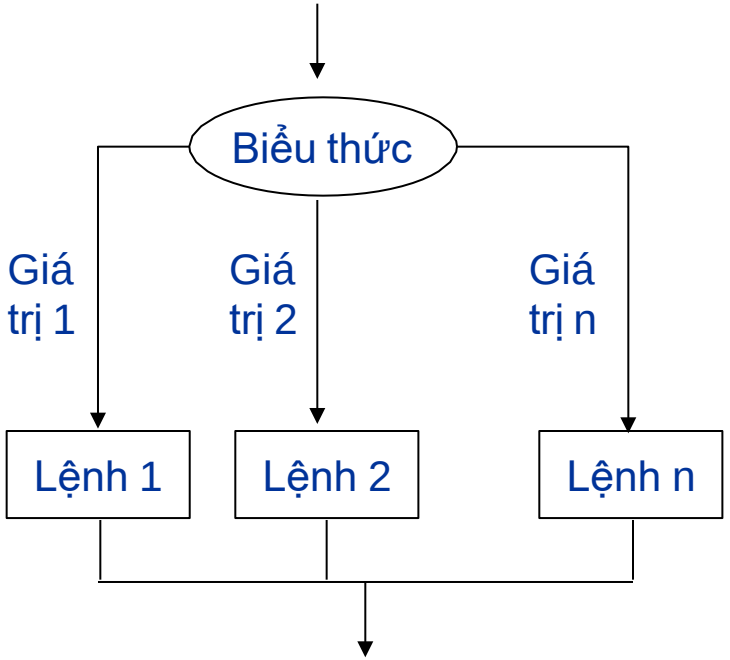


3.6. Các cấu trúc điều khiển - Rẽ nhiều nhánh

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

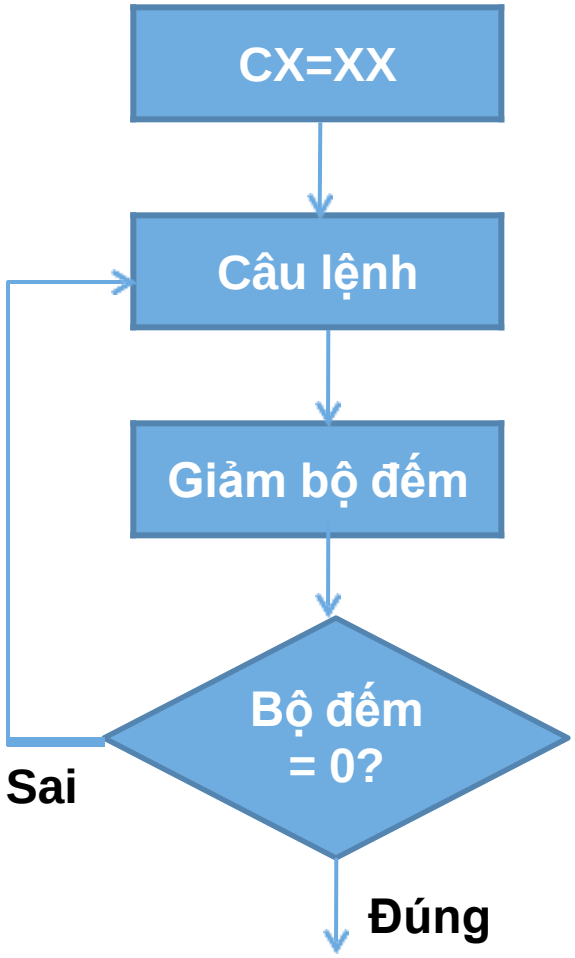
```
        CMP AX, 0
        JL AM
        JE KHONG
        JG DUONG
AM:     MOV CX, -1
        JMP RA
DUONG:  MOV CX, 1
        JMP RA
KHONG:  MOV CX, 0
RA:
```



3.6. Các cấu trúc điều khiển – Lặp kiểu for

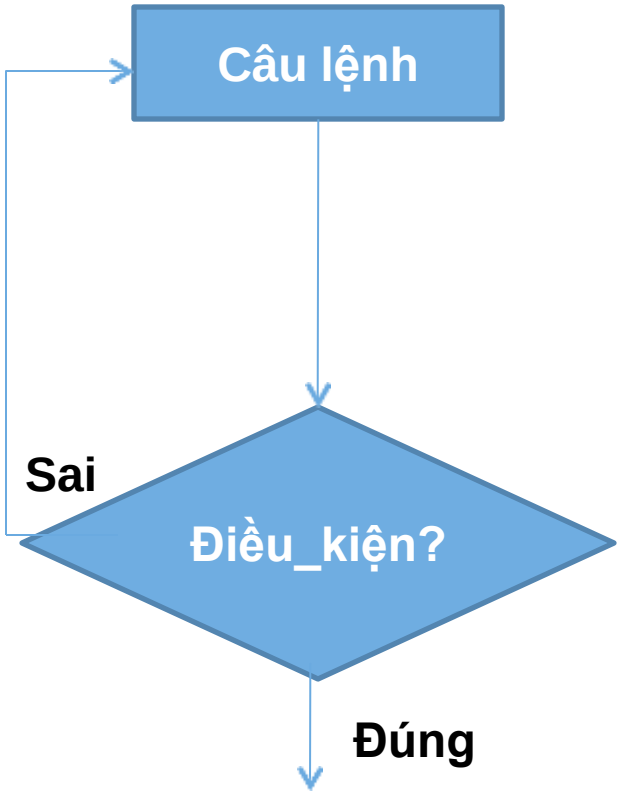
- Sử dụng lệnh LOOP
- Số lần lặp CX

```
1.      MOV CX,10
2.      MOV AH,2
3.      MOV DL,'9'
4.  Hien: INT 21H
5.      LOOP Hien
```

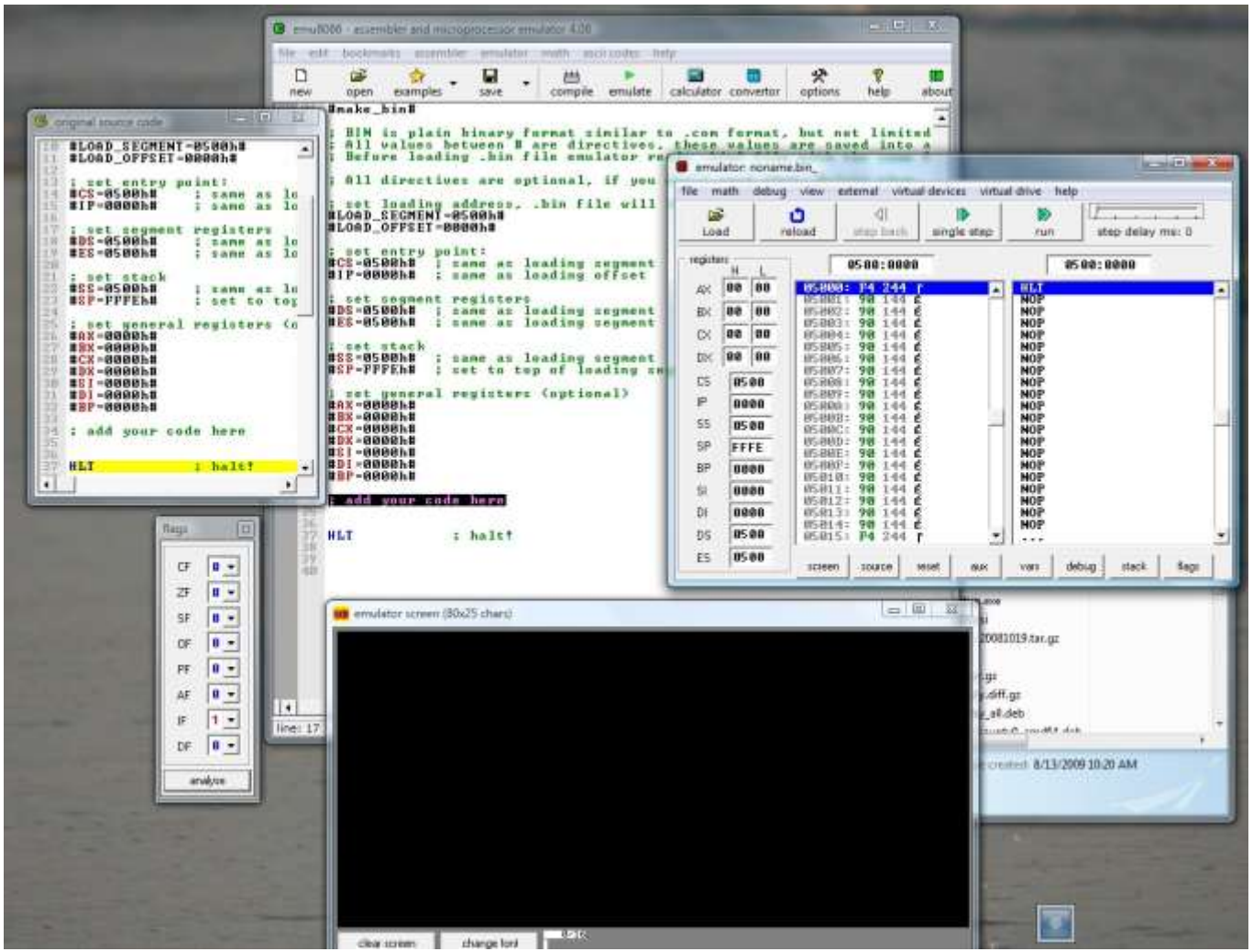


3.6. Các cấu trúc điều khiển – Lặp kiểu repeat ... until

1. …
2. Tiep:…
3. ….
4. CMP X,Y; điều kiện
5. Quay lại Tiếp nếu điều_kiện=sai;



3.7. Giới thiệu phần mềm mô phỏng emu8086



3.8. Một số ví dụ - Một số dịch vụ của ngắt 21H

- **Hàm 1 của ngắt INT 21H**: đọc 1 ký tự từ bàn phím

  Vào:AH = 1

  Ra:AL = mã ASCII của ký tự cần hiện thị

  AL = 0 khi ký tự gõ vào là phím chức năng

- **Hàm 2 của ngắt INT 21H**: hiện 1 ký tự lên màn hình

  Vào: AH = 2

  DL = mã ASCIIcủa ký tự cần hiện thị.

  Ra: Không

3.8. Một số ví dụ - Một số dịch vụ của ngắt 21H

- **Hàm 9 của ngắt INT 21H**: hiện chuỗi ký tự với $ ở cuối lên màn hình

  Vào: AH = 9

  DX = địa chỉ lệch của chuỗi ký tự cần hiện thị.

  Ra: Không

- **Hàm 4CH của ngắt INT 21H**: kết thúc chương trình kiểu EXE

  Vào: AH = 4CH

  Ra: Không

VD1- Hiện các lời chào ta và tây

```
. Model Small
. Stack 100
. Data
        CRLF        DB      13, 10, '$'
        Chao tay    DB      'hello!$'
        ChaoTa  DB          'Chao ban!$'
. Code
MAIN Proc
        MOV AX, @ Data ; khởi đầu thanh ghi DS
        MOV DS, AX
        ; hiện thị lời chào dùng hàm 9 của INT 21H
        MOV AH, 9
        LEA DX, ChaoTay
        INT 21H
```

VD1- Hiện các lời chào ta và tây

```
        ; cách 5 dòng dùng hàm 9 của INT 21H
        LEA DX, CRLF
        MOV CX, 6          ;CX chứa số dòng cách +1
LAP:    INT 21H
        LOOP LAP
        ; hiện thị lời chào dùng hàm 9 của INT 21H
        LEA DX, ChaoTa
        INT 21H
        ; trở về DOS dùng hàm 4 CH của INT 21H
        MOV AH, 4CH
        INT 21H
MAIN Endp
        END MAIN
```

VD2- Đổi các ký tự thường trong 1 chuỗi thành chữ hoa

```
.Model small
.Stack 100H
.Data
; source string
str1 DB 'a','5', 'B', '?', 'd', 'g', 'P','N','k','*'
     DB 10,13,'$'
; destination string
str2 DB 10 DUP(' ')
       DB '$'
.code
main proc
    ; initilize the ds and es registers
    mov ax, @Data
    mov ds,ax
    mov es,ax
```

VD2- Đổi các ký tự thường trong 1 chuỗi thành chữ hoa

```
 ; make SI points to str1 and DI to str2
    lea si, str1
    lea di, str2
    cld
    mov cx, 10
Start:
    lodsb
    ; check if it is lower case
    cmp al, 'a'
    jl NotLowerCase
    cmp al, 'z'
    jg NotLowerCase
    ; is lower case, convert to upper case
    sub al, 20H
    ; store to new string
 NotLowerCase: stosb
   loop Start
```

VD2- Đổi các ký tự thường trong 1 chuỗi thành chữ hoa

```
 ; print the original string
    lea dx, str1
    mov ah, 9
    int 21H
    ; print the output
    lea dx, str2
    mov ah, 9
    int 21H

    ; end program
    mov ah, 4CH
    int 21H
main endp
end main
```

VD3- Tìm số lớn nhất trong 1 dãy

```
.Model small
.Stack 100H
.Data
; source string
list DB 1,4,0,9,7,2,4,6,2,5
.code
main proc
    ; initilize the ds and es registers
    mov ax, @Data
    mov ds,ax
    cld
    mov cx, 9
    lea si, list      ; si points to list
    mov bl, [si]      ; max <-- 1st element
    inc si
```

VD3- Tìm số lớn nhất trong 1 dãy

```
Start:
    lodsb
    cmp al, bl
    jle BYPASS
    mov bl, al; al>bl --> bl to store new max
    BYPASS:
    loop Start
    ; print the max
    add bl, '0' ; digit to char
    mov dl,bl
    mov ah, 2
    int 21H
        ; end program
    mov ah, 4CH
    int 21H
main endp
End Main
```

3.9. Tạo và sử dụng chương trình con

- Chương trình con (còn gọi là thủ tục (procedure) hoặc hàm (function)):
  - Thường gồm một nhóm các lệnh gộp lại;
  - Được sử dụng thông qua tên và các tham số.
- Ý nghĩa của việc sử dụng chương trình con:
  - Chia chức năng giúp chương trình trong sáng, dễ hiểu, dễ bảo trì;
  - Chương trình con được viết một lần và có thể sử dụng nhiều lần.

3.9.1 Chương trình con – Khai báo và sử dụng

❖ Khai báo

```
<name> PROC

    ; here goes the code
    ; of the procedure ...

RET
<name> ENDP
```

❖ Sử dụng: gọi chương trình con

```
Call  <proc_name>
```

3.9.1 Chương trình con – Khai báo và sử dụng

```
MOV    AL, 1
MOV    BL, 2

CALL   m2
; other instructions
MOV CX, 30
;_____________________________
; define a proc
; input: AL, BL
; Output: AX
m2     PROC
     MUL    BL          ; AX = AL * BL.
     RET                ; return to caller.
m2     ENDP
```

3.9.2 Chương trình con – Truyền tham số

- Phục vụ trao đổi dữ liệu giữa chương trình gọi và chương trình con;
- Các phương pháp truyền tham số:
  - Truyền tham số thông qua các thanh ghi
    - Đưa giá trị vào các thanh ghi lưu tham số cần truyền trước khi gọi hoặc trở về từ chương trình con
  - Truyền tham số thông qua các biến toàn cục
    - Biến toàn cục (định nghĩa trong đoạn dữ liệu ở chương trình chính) có thể được truy nhập ở cả chương trình chính và chương trình con.
  - Truyền tham số thông qua ngăn xếp
    - Sử dụng kết hợp các lệnh PUSH / POP để truyền tham số.

3.9.2 Chương trình con – Truyền tham số

- Bảo vệ các thanh ghi:
  - Cần thiết phải bảo vệ giá trị các thanh ghi sử dụng trong chương trình gọi khi chúng cũng được sử dụng trong chương trình con.
  - Giá trị của các thanh ghi có thể bị thay đổi trong chương trình con → sai kết quả ở chương trình gọi.
- Các phương pháp bảo vệ các thanh ghi:
  - Sử dụng PUSH và POP cho các thanh ghi tổng quát, chỉ số và con trỏ;
  - Sử dụng PUSHF và POPF cho thanh ghi cờ;
  - Sử dụng qui ước thống nhất về sử dụng các thanh ghi.

3.9.3 Chương trình con – Ví dụ 1

```
; Find max of a list and print out the max
.Model small
.Stack 100H
.Data
; source string
list DB 1,4,0,9,7,2,4,6,2,5
.code
main proc
    ; initilize the ds and es registers
    mov ax, @Data
    mov ds,ax
    cld
    mov cx, 9
    lea si, list     ; si points to list
    mov bl, [si]     ; max <-- 1st element
    inc si
```

3.9.3 Chương trình con – Ví dụ 1

```
Start:
    lodsb
    cmp al, bl
    jle BYPASS
    mov bl, al; al>bl --> bl to store new max
    BYPASS:
    loop Start

    ; print the max
    call printSingleDigit

    ; end program
    mov ah, 4CH
    int 21H
main endp
```

3.9.3 Chương trình con – Ví dụ 1

```
;_____________________________
; proc to print out a single digit number
; input: bl to contain the digit to print
printSingleDigit proc
    push dx
    push ax
    add bl, '0' ; digit to char
    mov dl,bl
    mov ah, 2
    int 21H
    pop ax
    pop dx
    ret
printSingleDigit endp
end main
```

3.9.3 Chương trình con – Ví dụ 2

```
; convert lower case chars to upper cases
.Model small
.Stack 100H
.Data
; source string
str1 DB 'a','5', 'B', '?', 'd', 'g', 'P','N','k','*'
     DB 10,13,'$'
; destination string
str2 DB 10 DUP(' ')
     DB '$'
.code
main proc
    ; initilize the ds and es registers
    mov ax, @Data
    mov ds,ax
    mov es,ax
    ; make SI points to str1 and DI to str2
    lea si, str1
    lea di, str2
    cld
    mov cx, 10
```

3.9.3 Chương trình con – Ví dụ 2

```
Start:
    lodsb
    ; check if it is lower case
    cmp al, 'a'
    jl NotLowerCase
    cmp al, 'z'
    jg NotLowerCase
    ; is lower case, convert to upper case
    sub al, 20H

    ; store to new string
    NotLowerCase:
    stosb
    loop Start

    ; print the original string
     lea dx, str1
    call printString
```

3.9.3 Chương trình con – Ví dụ 2

```
; print the output
    lea dx, str2
    call printString

    ; end program
    mov ah, 4CH
    int 21H
main endp
; ---------------------------
; proc to print a string
; input: DX to contain the relative address of the string
printString proc
    push ax    ; store AX into stack
    mov ah, 9
    int 21H
    pop ax     ; restore AX from stack
    ret
printString endp
end main
```

3.9.3 Chương trình con – Ví dụ 3

```
; Sort a list to accending order
; print out the original and sorted lists
.Model small
.Stack 100H
.Data
    LIST_COUNT EQU 10
    list DB 1,4,0,3,7,2,8,6,2,5
    CRLF DB 13,10,'$'
.code
main proc
    ; initilize the ds and es registers
    mov ax, @Data
    mov ds,ax

    ; print the original list
    mov cx, LIST_COUNT
    lea si, list
    call printList
```

3.9.3 Chương trình con – Ví dụ 3

```
lea si, list     ; si points to list
mov bl, 1        ; main counter
MainLoop:
   mov al, [si]    ; al <-- [si]
    mov di, si
   mov bh, bl      ; sub-counter
   mov dx, di      ; dx to store min position
    SubLoop:
       inc di
       inc bh
       cmp al, [di]
     jle NotMin
     mov al, [di]
     mov dx, di
     NotMin:
       cmp bh, LIST_COUNT
       je ExitSub
       jmp SubLoop
   ExitSub:
```

3.9.3 Chương trình con – Ví dụ 3

```
; swap the position if min is different from first place
   mov di, dx
   cmp si, di
   je NoSwap
   call swapMemLocation
 NoSwap:
   inc bl
   cmp bl, LIST_COUNT
   je ExitMain
   inc si
   jmp MainLoop
 ExitMain:
```

3.9.3 Chương trình con – Ví dụ 3

```
; print the new line chars
    lea dx, CRLF
    call printString

    ; print the sorted list
    mov cx, LIST_COUNT
    lea si, list     ; si points to list
    call printList

    ; end program
    mov ah, 4CH
    int 21H
main endp
```

3.9.3 Chương trình con – Ví dụ 3

```
;_____________________________
; swap the value of 2 memory locations
; input: si points to the 1st memory location
;        di points to the 2nd memory location
swapMemLocation proc
    push ax
    mov al, [si]
    mov ah, [di]
    mov [si], ah
    mov [di], al
    pop ax
    ret
swapMemLocation endp
```

3.9.3 Chương trình con – Ví dụ 3

```
;_____________________
; print the list
; input: SI to store the start address of the list
;        CX to store the number of elements
printList proc
    push dx
  StartPrint:
    mov dl, [si]
    call printSingleDigit
    inc si
    loop StartPrint
    pop dx
    ret
printList endp
```

3.9.3 Chương trình con – Ví dụ 3

```
; print a string ending with $
;input: DX to point to string
printString proc
    push ax
    mov ah, 9
    int 21H
    pop ax
    ret
printString endp
;_____________________________
; proc to print out a single digit number
; input: dl to contain the digit to print
printSingleDigit proc
    push ax
    add dl, '0' ; digit to char
    mov ah, 2
    int 21H
    pop ax
    ret
printSingleDigit endp
end main
```

3.10 Tạo và sử dụng macro

- Macro là một đoạn mã được đặt tên và có thể được chèn vào bất cứ vị trí nào trong đoạn mã của chương trình
- Đặc điểm của macro:
  - Macro hỗ trợ danh sách các tham số
  - Macro chỉ tồn tại khi soạn thảo mã. Khi dịch, các macro sẽ được thay thế bằng đoạn mã thực của macro.
  - Nếu một macro không được sử dụng, mã của nó sẽ bị loại khỏi chương trình sau khi dịch.
  - Macro nhanh hơn thủ tục/hàm do mã của macro được chèn trực tiếp vào chương trình và nó không đòi hỏi cơ chế gọi thực hiện (lưu địa chỉ) và trở về (khôi phục địa chỉ trở về) như chương trình con.

3.10 Tạo và sử dụng macro

❖ Định nghĩa macro:

name MACRO [parameters,...]

<instructions>

ENDM

❖ Sử dụng macro:

<macro_name> [real parameters]

3.10 Tạo và sử dụng macro

❖ Ví dụ

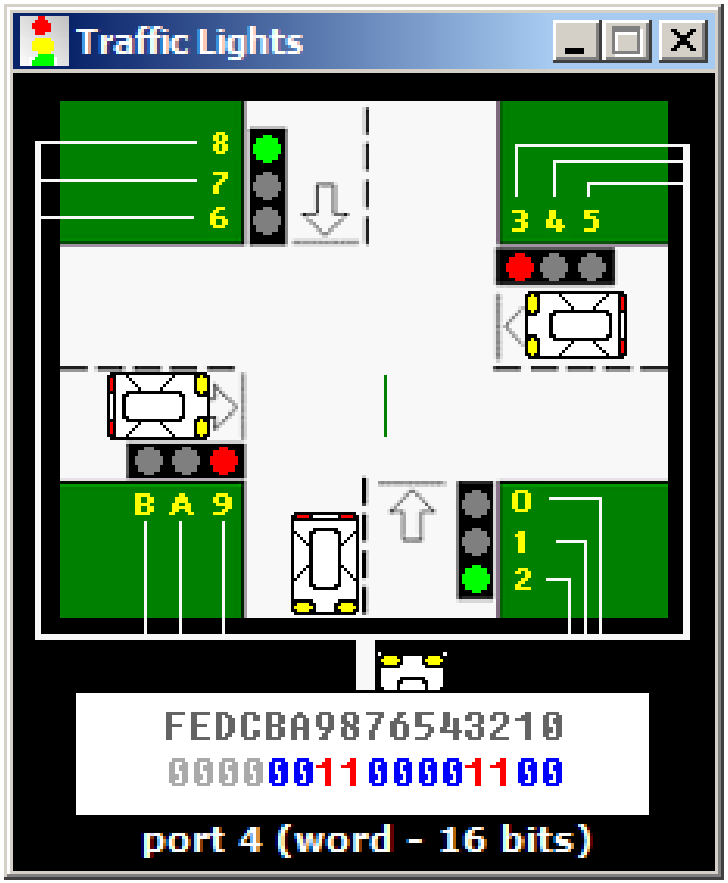
```
MyMacro    MACRO p1, p2, p3
    MOV AX, p1
    MOV BX, p2
    MOV CX, p3
ENDM
;...
MyMacro 1, 2, 3
MyMacro 4, 5, DX
```

Được chuyển thành sau dịch:

```
MOV AX, 00001h
MOV BX, 00002h
MOV CX, 00003h
MOV AX, 00004h
MOV BX, 00005h
MOV CX, DX
```

3.11 Giới thiệu thiết bị ảo – Đèn giao thông

- Thiết bị ảo hệ thống đèn giao thông sử dụng cổng số 4 – cổng 16 bít để nhận thông tin điều khiển;
- Sử dụng 12 bít (0-11) cho 4 cụm đèn:
  - Mỗi cụm gồm 3 đèn Green, Yellow và Red;
  - Bít 0 – tắt đèn, bít 1 – bật đèn
- 4 bít (12-15) không sử dụng – nên đặt là 0.



3.11 Giới thiệu thiết bị ảo – Đèn giao thông

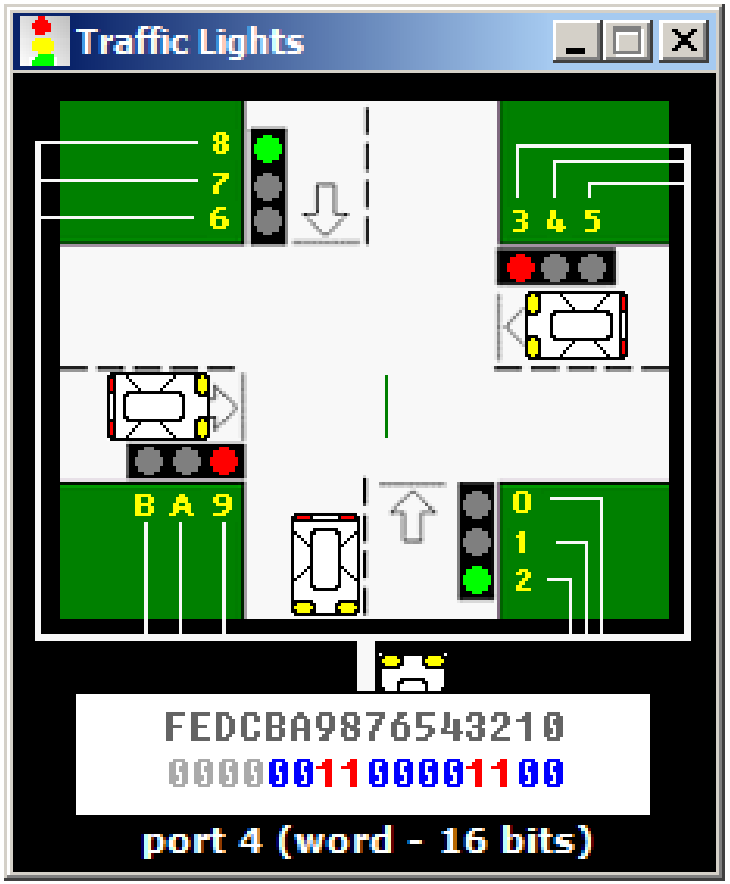
- Điều khiển đèn giao thông:
  - Gửi từ điều khiển (2 bytes) ra cổng số 4;
  - Các bít của từ điều khiển được đặt sao cho phù hợp với ý đồ điều khiển đèn (Bít 0 – tắt đèn, bít 1 – bật đèn)

VD: từ điều khiển:

0000 001 100 001 100

GYR GYR GYR GYR

  - Dùng hàm 86h của ngắt BIOS 15h để tạo thời gian đợi – thời gian giữ trạng thái vừa thiết lập của cụm đèn. Số micro giây được đặt vào CX:DX trước khi gọi ngắt.



3.11 Giới thiệu thiết bị ảo – Đèn giao thông

```
mov ax, all_red  out 4, ax
mov si, offset situation  next:
mov ax, [si]  out 4, ax

; wait 5 seconds (5 million microseconds)
    mov cx, 4Ch ; 004C4B40h = 5,000,000
    mov     dx, 4B40h
    mov     ah, 86h
    int     15h
    add si, 2 ; next situation
    cmp si, sit_end
    jb  next
    mov si, offset situation
    jmp next
```

3.11 Giới thiệu thiết bị ảo – Đèn giao thông

```
;                   FEDC_BA98_7654_3210
situation       dw      0000_0011_0000_1100b
s1              dw      0000_0110_1001_1010b
s2              dw      0000_1000_0110_0001b
s3              dw      0000_1000_0110_0001b
s4              dw  0000_0100_1101_0011b
sit_end = $


all_red         equ     0000_0010_0100_1001b
```

3.11 Giới thiệu thiết bị ảo – Đèn giao thông

```
.Model small
.Stack 100H
.Data
  ;                          GYR GYR GYR GYR
  R1 DW                    0000_0011_0000_1100b
  R2 DW                    0000_0010_1000_1010b
  R3 DW                    0000_1000_0110_0001b
  R4 DW                    0000_0100_0101_0001b
  ;                        FEDC_BA9 876 543 210
all_red         equ     0000_0010_0100_1001b
PORT EQU  4     ; output port

  ; time constants (in secs)
  WAIT_3_SEC_CX   EQU   2Dh
  WAIT_3_SEC_DX   EQU   0C6C0h

  WAIT_10_SEC_CX   EQU   98h
  WAIT_10_SEC_DX   EQU   9680h
```

3.11 Giới thiệu thiết bị ảo – Đèn giao thông

```
.code
; define a macro
waitMacro macro t1, t2
  mov cx, t1
  mov dx, t2
  mov     ah, 86h
  int     15h
waitMacro endm

main proc
  ; initilize the ds and es registers
  mov ax, @Data
  mov ds,ax

  ; set lights to Red for all direction
  mov ax, all_red
  out PORT, ax
  waitMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX
```

3.11 Giới thiệu thiết bị ảo – Đèn giao thông

```
Start:
  lea si, R1
  mov ax, [si]
  out PORT, ax

  waitMacro WAIT_10_SEC_CX, WAIT_10_SEC_DX

  lea si, R2
  mov ax, [si]
  out PORT, ax

  waitMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX

  lea si, R3
  mov ax, [si]
  out PORT, ax
```

3.11 Giới thiệu thiết bị ảo – Đèn giao thông

```
waitMacro WAIT_10_SEC_CX, WAIT_10_SEC_DX

  lea si, R4
  mov ax, [si]
  out PORT, ax

  waitMacro WAIT_3_SEC_CX, WAIT_3_SEC_DX

  jmp Start


  ; end program
  mov ah, 4CH
  int 21H
main endp

end main
```