

HỆ THỐNG MÁY TÍNH

THAM KHẢO

LẬP TRÌNH HỢP NGỮ CHO 8086

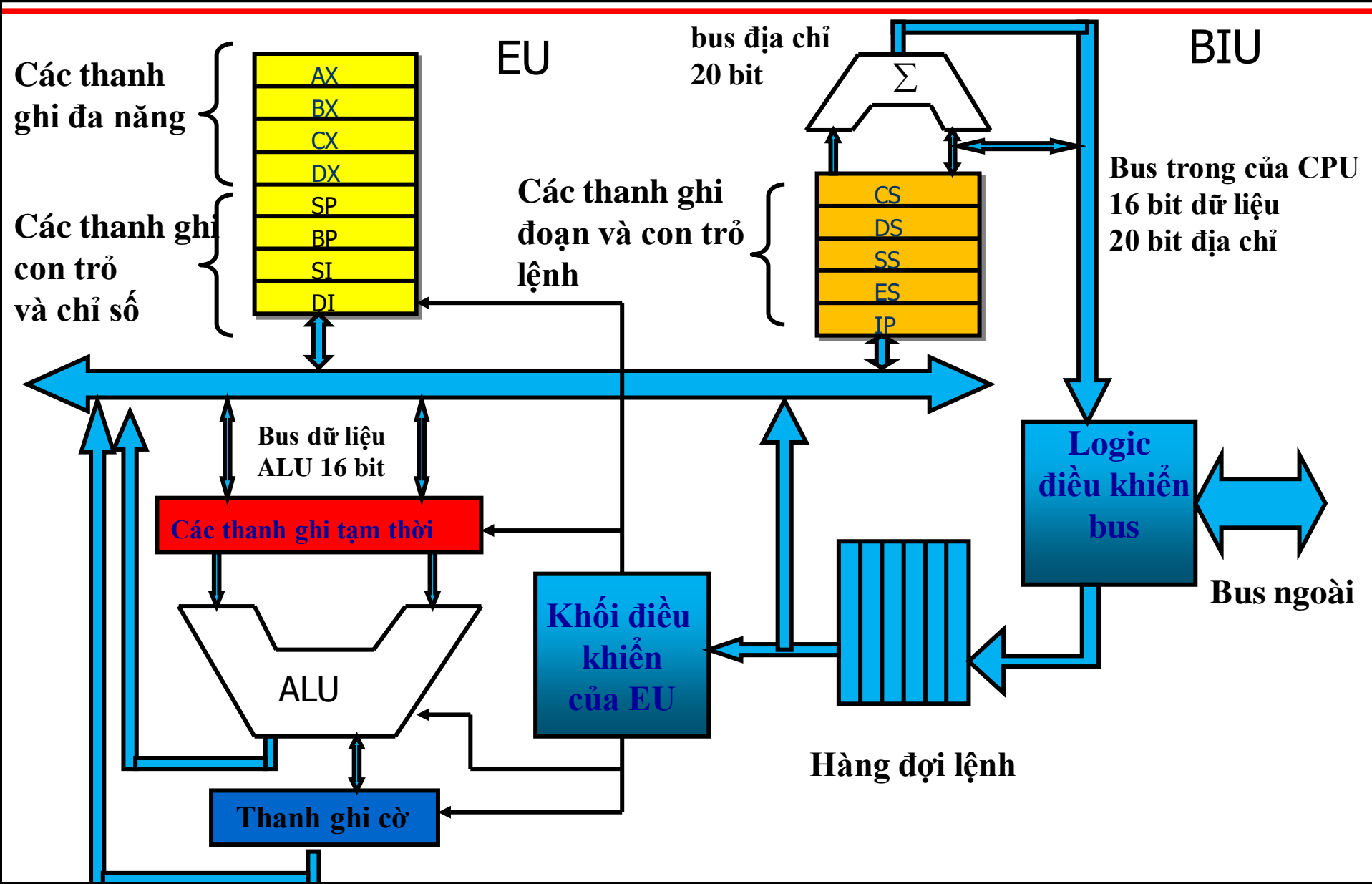
Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
- Mô tả tập lệnh của 8086
- Lập trình hợp ngữ 8086

Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
 - ☐ Sơ đồ khối
 - ☐ Các thanh ghi đa năng
 - ☐ Các thanh ghi đoạn
 - ☐ Các thanh ghi con trỏ và chỉ số
 - ☐ Thanh ghi cờ
 - ☐ Hàng đợi lệnh
- Mô tả tập lệnh của 8086
- Lập trình hợp ngữ 8086

Sơ đồ khối 8088/8086



Các thanh ghi đa năng của 8088/8086

	8 bit cao	8 bit thấp
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

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

- Thanh ghi chứa AX (accumulator): chứa kết quả của các phép tính. Kết quả 8 bit được chứa trong AL
- Thanh ghi cơ sở BX (base): chứa địa chỉ cơ sở, ví dụ của bảng dùng trong lệnh XLAT (Translate)
- Thanh ghi đếm CX (count): dùng để chứa số lần lặp trong các lệnh lặp (Loop). CL được dùng để chứa số lần dịch hoặc quay trong các lệnh dịch và quay thanh ghi
- Thanh ghi dữ liệu DX (data): cùng AX chứa dữ liệu trong các phép tính nhân chia số 16 bit. DX còn được dùng để chứa địa chỉ cổng trong các lệnh vào ra dữ liệu trực tiếp (IN/OUT)

Các thanh ghi đoạn

- Tổ chức của bộ nhớ 1 Mbytes

- Đoạn bộ nhớ (segment)

- ⇒ 2^{16} bytes = 64 KB

- ⇒ Đoạn 1: địa chỉ đầu 00000 H

- ⇒ Đoạn 2: địa chỉ đầu 00010 H

- ⇒ Đoạn cuối cùng: FFFF0 H

- Ô nhớ trong đoạn:

- ⇒ địa chỉ lệch: offset

- ⇒ Ô 1: offset: 0000

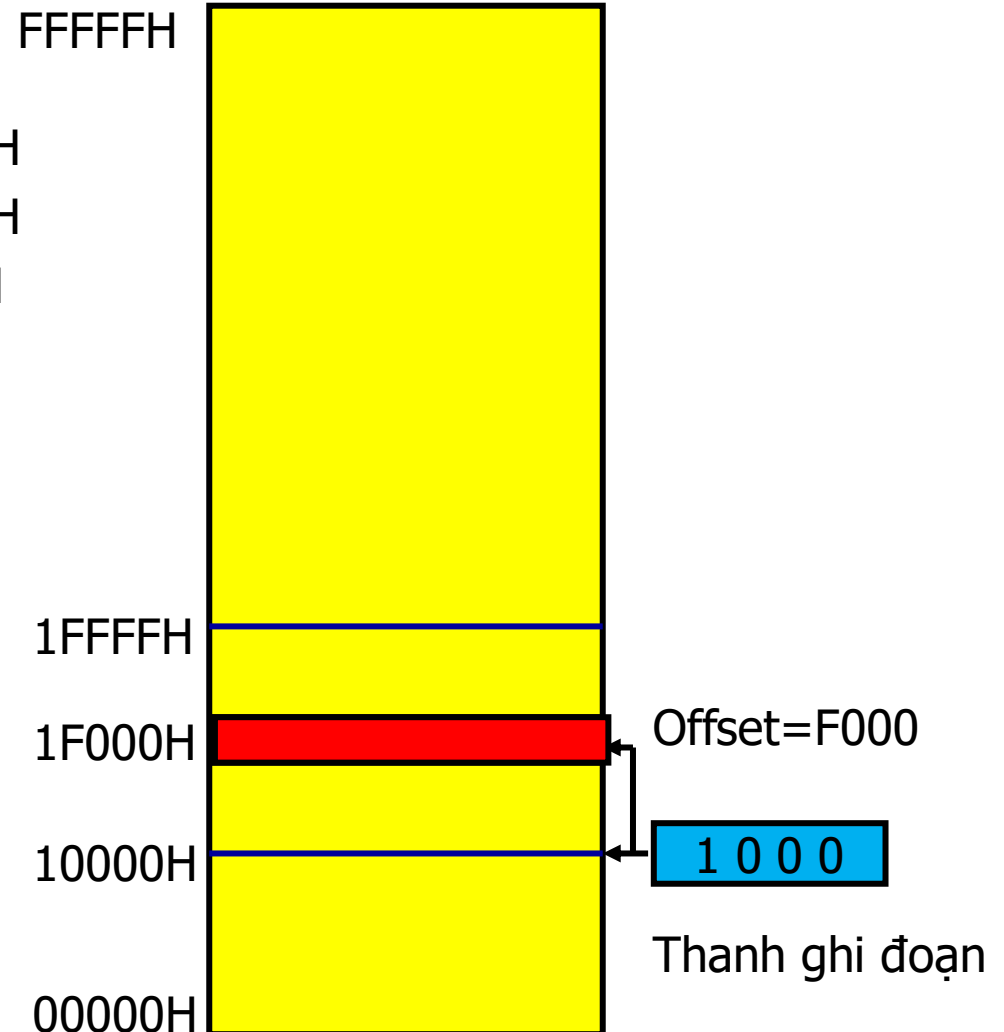
- ⇒ Ô cuối cùng: offset: FFFF

- Địa chỉ vật lý:

- ⇒ Segment : offset

Địa chỉ vật lý = Segment * 16 + offset

Chế độ thực (real mode)



Các thanh ghi đoạn

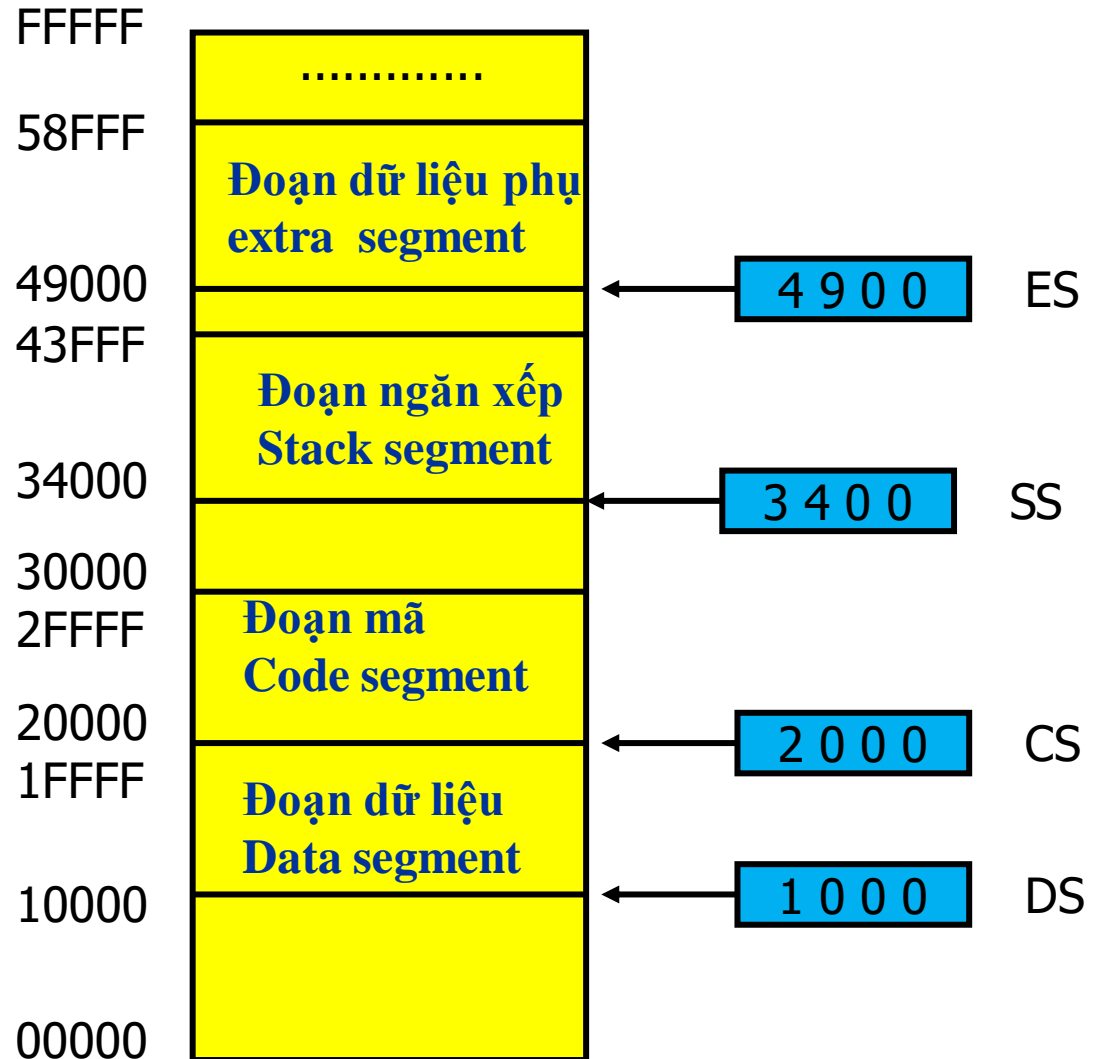
- Ví dụ: Địa chỉ vật lý 12345H

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

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

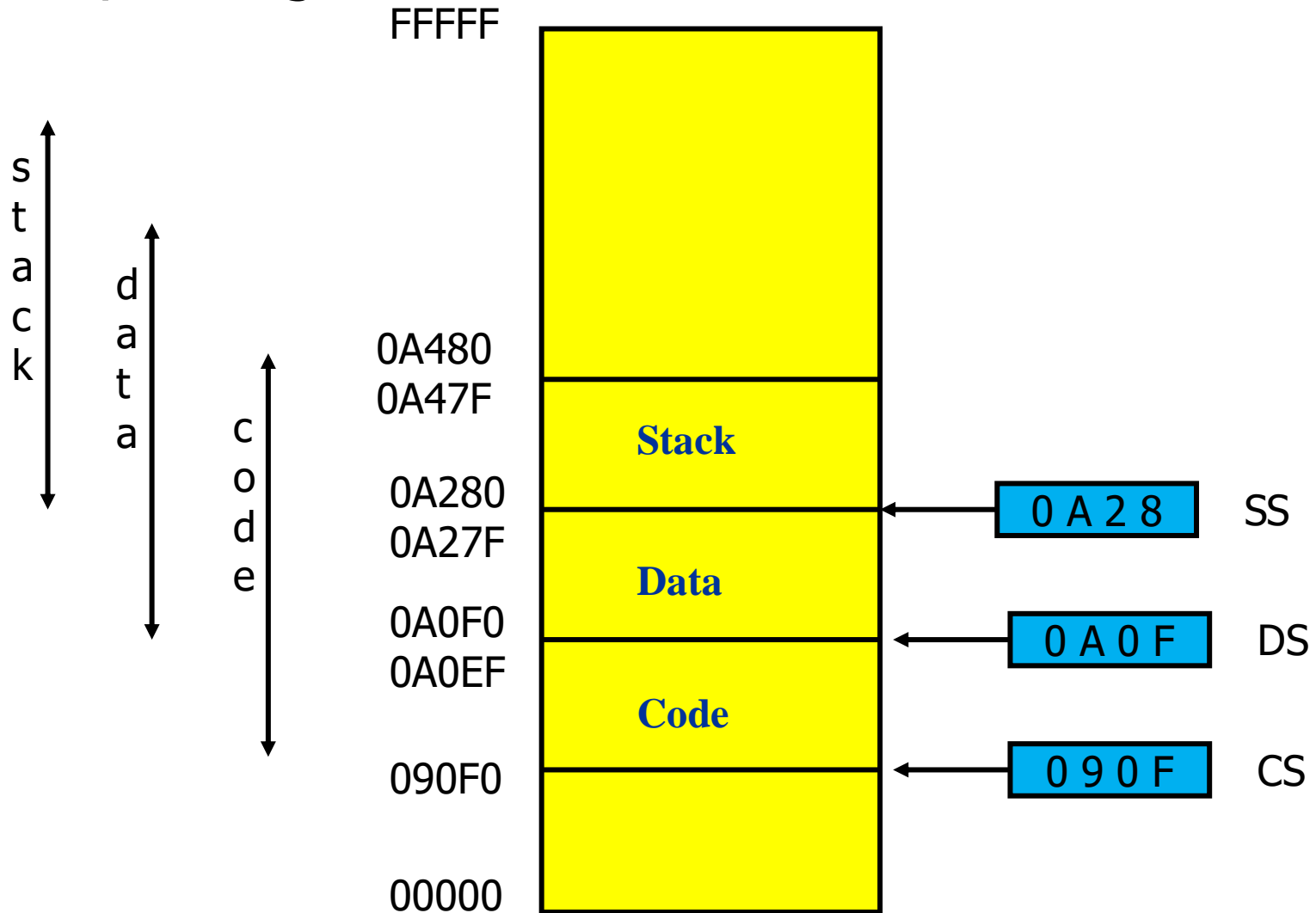
Các thanh ghi đoạn

- Các thanh ghi đoạn: chứa địa chỉ đoạn



Các thanh ghi đoạn

- Các đoạn chồng nhau



Các thanh ghi con trỏ và chỉ số

- Chứa địa chỉ lệch (offset)

- ☐ Con trỏ lệnh IP (instruction pointer): chứa địa chỉ lệnh tiếp theo trong đoạn mã lệnh CS.
⇒ CS:IP
- ☐ Con trỏ cơ sở BP (Base Pointer): chứa địa chỉ của dữ liệu trong đoạn ngăn xếp SS hoặc các đoạn khác
⇒ SS:BP
- ☐ Con trỏ ngăn xếp SP (Stack Pointer): chứa địa chỉ hiện thời của đỉnh ngăn xếp
⇒ SS:SP
- ☐ Chỉ số nguồn SI (Source Index): chứa địa chỉ dữ liệu nguồn trong đoạn dữ liệu DS trong các lệnh chuỗi
⇒ DS:SI
- ☐ Chỉ số đích (Destination Index): chứa địa chỉ dữ liệu đích trong đoạn dữ liệu DS trong các lệnh chuỗi
⇒ DS:DI
- ☐ SI và DI có thể được sử dụng như thanh ghi đa năng
- ☐ 80386 trở lên 32 bit: EIP, EBP, ESP, EDI, ESI

Các thanh ghi con trỏ và chỉ số

- Thanh ghi đoạn và thanh ghi lệch ngầm định

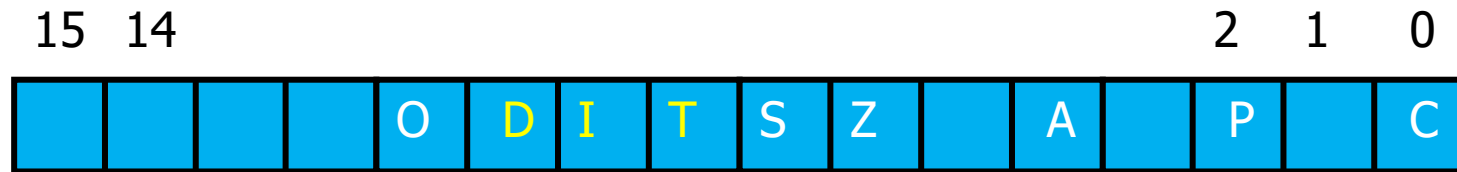
Segment	Offset	Chú thích
CS	IP	Địa chỉ lệnh
SS	SP hoặc BP	Địa chỉ ngăn xếp
DS	BX, DI, SI, số 8 bit hoặc số 16 bit	Địa chỉ dữ liệu
ES	DI	Địa chỉ chuỗi đích

Thanh ghi cờ (Flag Register)



- 9 bit được sử dụng, 6 cờ trạng thái:
 - ☐ C hoặc CF (carry flag): CF=1 khi có nhớ hoặc mượn từ MSB
 - ☐ P hoặc PF (parity flag): PF=1 (0) khi tổng số bit 1 trong kết quả là chẵn (lẻ)
 - ☐ A hoặc AF (auxiliary carry flag): cờ nhớ phụ, AF=1 khi có nhớ hoặc mượn từ một số BCD thấp sang BCD cao
 - ☐ Z hoặc ZF (zero flag): ZF=1 khi kết quả bằng 0
 - ☐ S hoặc SF (Sign flag): SF=1 khi kết quả âm
 - ☐ O hoặc OF (Overflow flag): cờ tràn OF=1 khi kết quả là một số vượt ra ngoài giới hạn biểu diễn của nó trong khi thực hiện phép toán cộng trừ số có dấu

Thanh ghi cờ (Flag Register)



- 3 cờ điều khiển
 - ☐ T hoặc TF (trap flag): cờ bẫy, TF=1 khi CPU làm việc ở chế độ chạy từng lệnh
 - ☐ I hoặc IF (Interrupt enable flag): cờ cho phép ngắt, IF=1 thì CPU sẽ cho phép các yêu cầu ngắt (ngắt che được) được tác động (Các lệnh: STI, CLI)
 - ☐ D hoặc DF (direction flag): cờ hướng, DF=1 khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải sang trái (lệnh STD, CLD)

Thanh ghi cờ (Flag Register)

- Ví dụ:

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

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

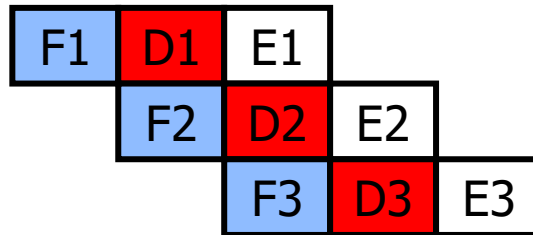
Hàng đợi lệnh

- 4 bytes đối với 8088 và 6 bytes đối với 8086
- Xử lý pipeline

Không có
pipelining



Có pipelining



Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
- Mô tả tập lệnh của 8086
 - ❑ Các lệnh di chuyển dữ liệu
 - ❑ Các lệnh số học và logic
 - ❑ Các lệnh điều khiển chương trình
- Lập trình hợp ngữ với 8086

Các lệnh di chuyển dữ liệu

- MOV, XCHG, POP, PUSH, POPF, PUSHF, IN, OUT
- Các lệnh di chuyển chuỗi MOVS, MOVSB, MOVSW
- 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, nguồn
 - ❑ Lệnh này không tác động đến cờ
 - ❑ Ví dụ:
 - ⇒ MOV AX, BX
 - ⇒ MOV AH, 'A'
 - ⇒ MOV AL, [1234H]

Các lệnh di chuyển dữ liệu

- Khả năng kết hợp toán hạng của lệnh MOV

<div>Đích Nguồn</div>	Thanh ghi đa năng	Thanh ghi đoạn	ô nhớ	Hằng số
Thanh ghi đa năng	YES	YES	YES	NO
Thanh ghi đoạn	YES	NO	YES	NO
Ô nhớ	YES	YES	NO	NO
Hằng số	YES	NO	YES	NO

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, nguồn
 - ❑ 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]

Các lệnh di chuyển dữ liệu

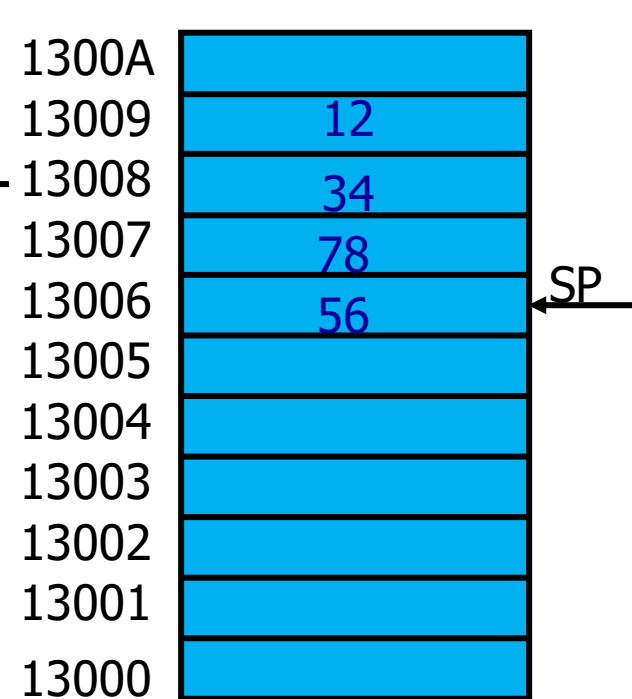
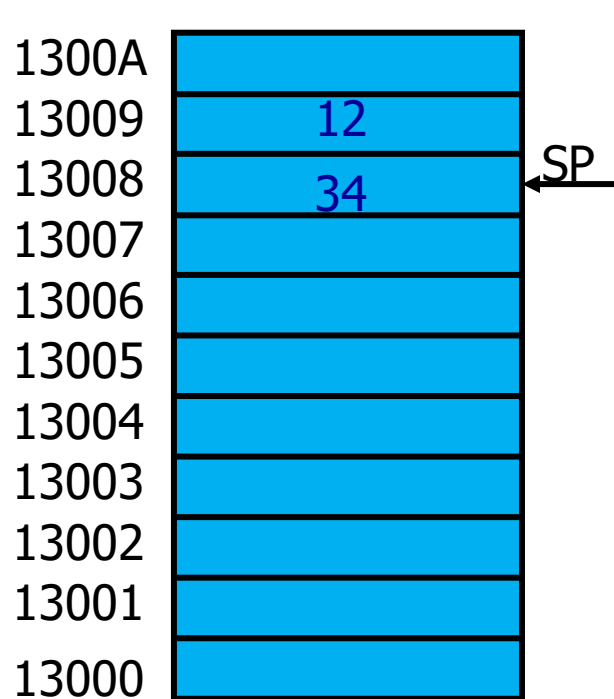
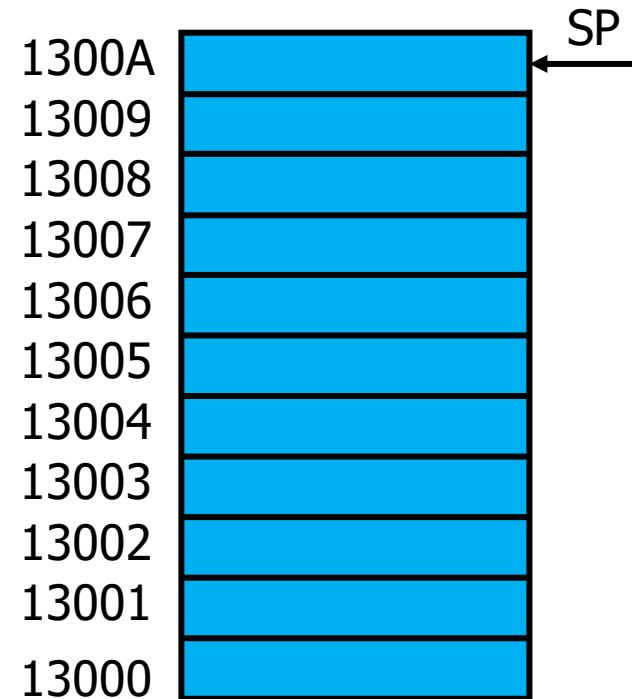
- Lệnh PUSH
 - ❑ Dùng để cất 1 từ từ thanh ghi hoặc ô nhớ vào đỉnh ngăn xếp
 - ❑ Cú pháp: PUSH Nguồn
 - ❑ Mô tả: $SP = SP - 2$, Nguồn $\Rightarrow \{SP\}$
 - ❑ Giới hạn: thanh ghi 16 bit hoặc là 1 từ nhớ
 - ❑ Lệnh này không tác động đến cờ
 - ❑ Ví dụ:
 - \Rightarrow PUSH BX
 - \Rightarrow PUSH PTR[BX]
- Lệnh PUSHF
 - ❑ Cất nội dung của thanh ghi cờ vào ngăn xếp

Các lệnh di chuyển dữ liệu

- Ví dụ về lệnh PUSH

PUSH AX

PUSH BX



SS 1 3 0 0

SS 1 3 0 0

SS 1 3 0 0

SP 0 0 0 A

SP 0 0 0 8

SP 0 0 0 6

AX 1 2 3 4

AX 1 2 3 4

BX 7 8 5 6

Các lệnh di chuyển dữ liệu

- Lệnh POP

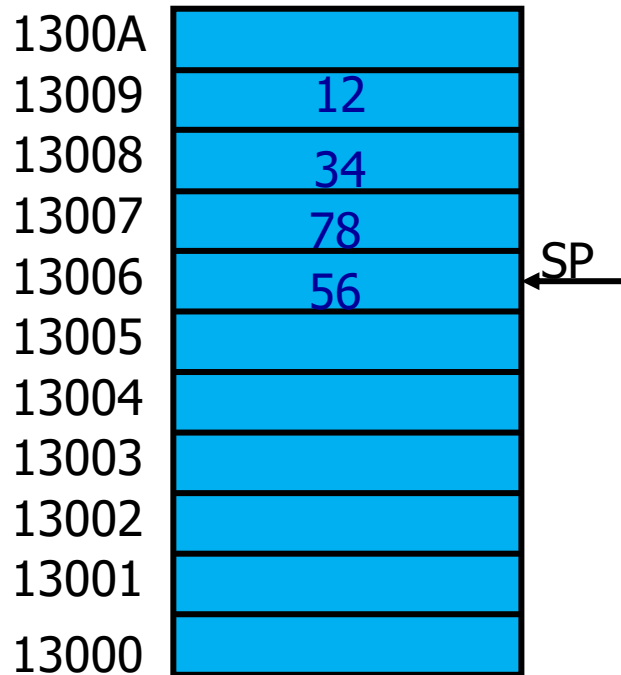
- ☐ Dùng để lấy lại 1 từ vào thanh ghi hoặc ô nhớ từ đỉnh ngăn xếp
- ☐ Cú pháp: POP Đích
- ☐ Mô tả: $\{SP\} \Rightarrow \text{Đích}, SP=SP+2$
- ☐ Giới hạn: thanh ghi 16 bit (trừ CS) hoặc là 1 từ nhớ
- ☐ Lệnh này không tác động đến cờ
- ☐ Ví dụ:
 - \Rightarrow POP BX
 - \Rightarrow POP PTR[BX]

- Lệnh POPF

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

Các lệnh di chuyển dữ liệu

- Ví dụ lệnh POP

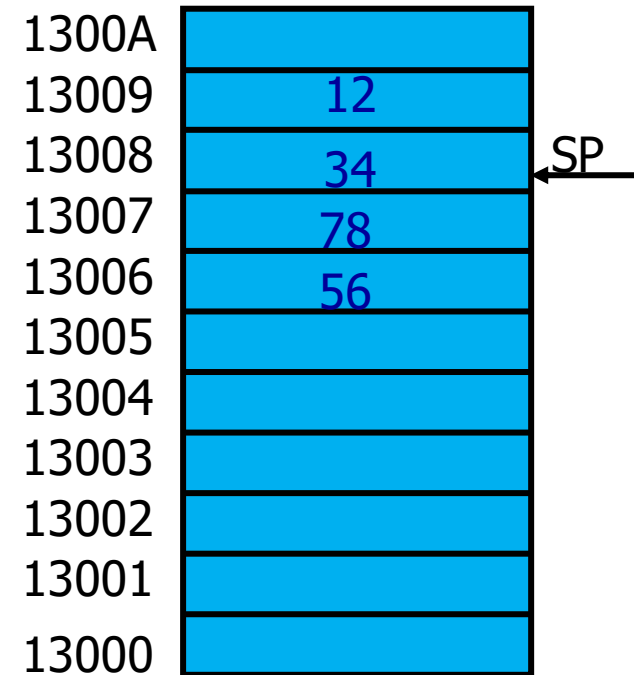


SS 1 3 0 0

SP 0 0 0 6

DX 3 2 5 4

POP DX



SS 1 3 0 0

SP 0 0 0 8

DX 7 8 5 6

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
 - ❑ Lệnh này không tác động đến cờ
 - ❑ Ví dụ:
 - ⇒ IN AX, 00H
 - ⇒ IN AL, F0H
 - ⇒ IN AX, DX
- 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
 - ❑ Lệnh này không tác động đến cờ
 - ❑ Ví dụ:
 - ⇒ OUT 00H, AX
 - ⇒ OUT F0H, AL
 - ⇒ OUT DX, AX

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 nguồn
MOVSB
MOVSW
 - ❑ 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 chuyển $SI = SI \pm 1$, $DI = DI \pm 1$ hoặc $SI = SI \pm 2$, $DI = DI \pm 2$ tùy thuộc vào cờ hướng DF là 0/1
 - ❑ Lệnh này không tác động đến cờ
 - ❑ Ví dụ:
 - ⇒ MOVS byte1, byte2

Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
- Mô tả tập lệnh của 8086
 - ☐ Các lệnh di chuyển dữ liệu
 - ☒ Các lệnh số học và logic
 - ☐ Các lệnh điều khiển chương trình
- Lập trình hợp ngữ với 8086

Các lệnh số học và logic

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

- Lệnh ADD
 - ❑ Lệnh cộng hai toán hạng
 - ❑ Cú pháp: ADD Đích, nguồn
 - ❑ Thực hiện: Đích=Đích + nguồn
 - ❑ 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:
 - ⇒ ADD AX, BX
 - ⇒ ADD AX, 40H

Các lệnh số học và logic

- Lệnh ADC

- ☐ Lệnh cộng có nhớ hai toán hạng
- ☐ Cú pháp: ADC Đích, nguồn
- ☐ Thực hiện: $\text{Đích} = \text{Đích} + \text{nguồn} + \text{CF}$
- ☐ 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

- Lệnh SUB

- ☐ Lệnh trừ
- ☐ Cú pháp: SUB Đích, nguồn
- ☐ Thực hiện: $\text{Đích} = \text{Đích} - \text{nguồn}$
- ☐ 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

Các lệnh số học và logic

- Lệnh MUL
 - ☐ Lệnh nhân số không dấu
 - ☐ Cú pháp: MUL nguồn
 - ☐ Thực hiện:
 - $\Rightarrow AX = AL * \text{nguồn}_{8\text{bit}}$
 - $\Rightarrow DXAX = AX * \text{nguồn}_{16\text{bit}}$
 - ☐ Lệnh này thay đổi cờ: CF, OF
 - ☐ Ví dụ:
 - $\Rightarrow \text{MUL BL}$
- Lệnh IMUL
 - ☐ nhân số có dấu

Các lệnh số học và logic

- Lệnh DIV

- ☐ Lệnh chia 2 số không dấu

- ☐ Cú pháp: DIV nguồn

- ☐ Thực hiện:

- ⇒ $AL = \text{thương} (AX / \text{nguồn8bit}) ; AH = \text{dư} (AX / \text{nguồn8bit})$

- ⇒ $AX = \text{thương} (DXAX / \text{nguồn16bit}) ; DX = \text{dư} (DXAX / \text{nguồn16bit})$

- ☐ Lệnh này không thay đổi cờ

- ☐ Ví dụ:

- ⇒ DIV BL

- Lệnh IDIV

- ☐ chia 2 số có dấu

Các lệnh số học và logic

- Lệnh INC

- ☐ Lệnh cộng 1 vào toán hạng là thanh ghi hoặc ô nhớ
- ☐ Cú pháp: INC Đích
- ☐ Thực hiện: $\text{Đích} = \text{Đích} + 1$
- ☐ Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
- ☐ Ví dụ:
 - ⇒ INC AX

- Lệnh DEC

- ☐ Lệnh trừ 1 từ nội dung một thanh ghi hoặc ô nhớ
- ☐ Cú pháp: DEC Đích
- ☐ Thực hiện: $\text{Đích} = \text{Đích} - 1$
- ☐ Lệnh này thay đổi cờ: AF, OF, PF, SF, ZF
- ☐ Ví dụ:
 - ⇒ DEC [BX]

Các lệnh số học và logic

- Lệnh AND
 - ❑ Lệnh AND logic 2 toán hạng
 - ❑ Cú pháp: AND Đích, nguồn
 - ❑ Thực hiện: Đích=Đích And nguồn
 - ❑ 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: xác định số bù 2 của toán hạng

Các lệnh số học và logic

- Lệnh CMP

- ☐ Lệnh so sánh 2 byte hoặc 2 từ

- ☐ Cú pháp: CMP Đích, nguồn

- ☐ Thực hiện:

- ⇒ Đích = nguồn : CF=0 ZF=1

- ⇒ Đích > nguồn : CF=0 ZF=0

- ⇒ Đích < nguồn : CF=1 ZF=0

- ☐ Giới hạn: toán hạng phải cùng độ dài và không được là 2 ô nhớ

- Lệnh CMPS

- ☐ Dùng để so sánh từng phần tử của 2 chuỗi có các phần tử cùng loại

- ☐ Cú pháp: CMPS chuỗi đích, chuỗi nguồn

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

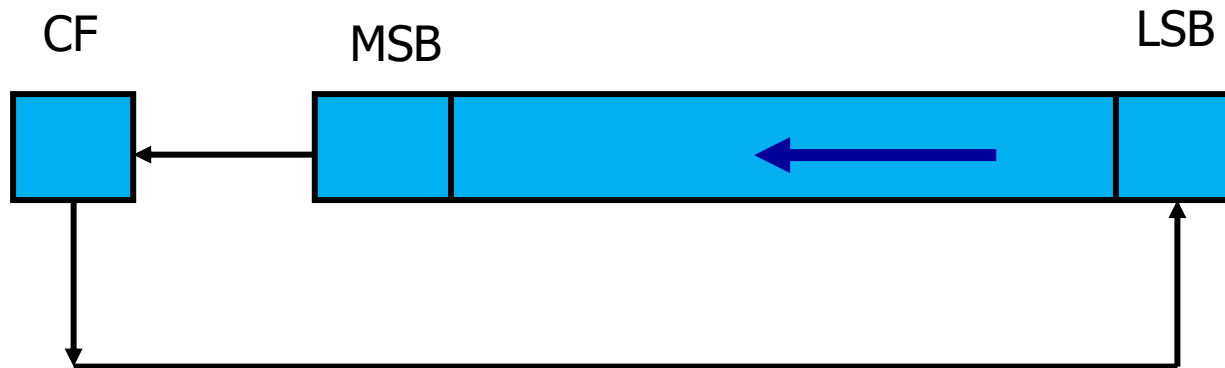
- ⇒ Sau mỗi lần so sánh SI=SI +/- 1, DI=DI +/- 1 hoặc SI=SI +/- 2, DI=DI +/- 2 tùy thuộc vào cờ hướng DF là 0/1

- ☐ Cập nhật cờ AF, CF, OF, PF, SF, ZF

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 CL lần
- ☐ Đích là thanh ghi (trừ thanh ghi đoạn) hoặc ô nhớ
- ☐ Lệnh này thay đổi cờ: CF, OF



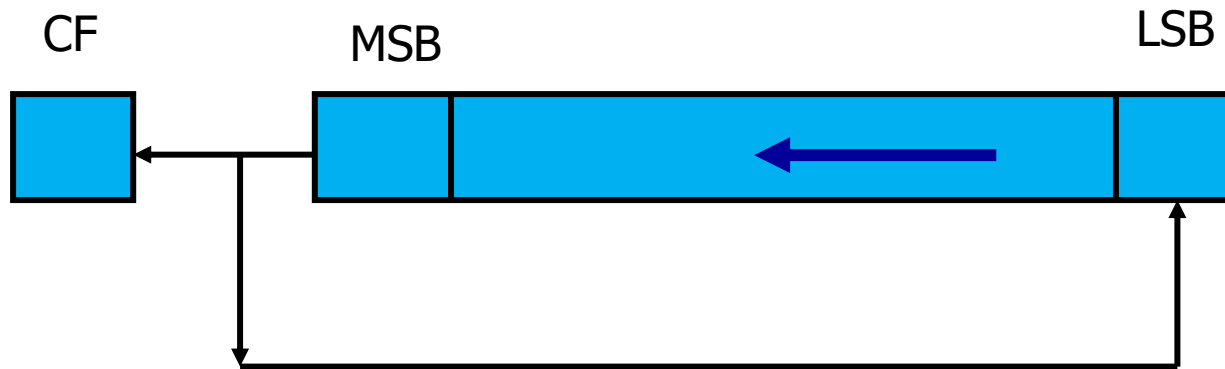
- Lệnh RCR

- ☐ Lệnh quay phải thông qua cờ nhớ

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

Các lệnh số học và logic

- Lệnh SAL

- ☐ 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 $\text{Đích} = \text{Đích} * 2^{\text{CL}}$
- ☐ Lệnh này thay đổi cờ SF, ZF, PF



- Lệnh SHL

- ☐ Lệnh dịch trái logic tương tự như SAL

Các lệnh số học và logic

- Lệnh SAR

- ☐ Lệnh dịch phải số học

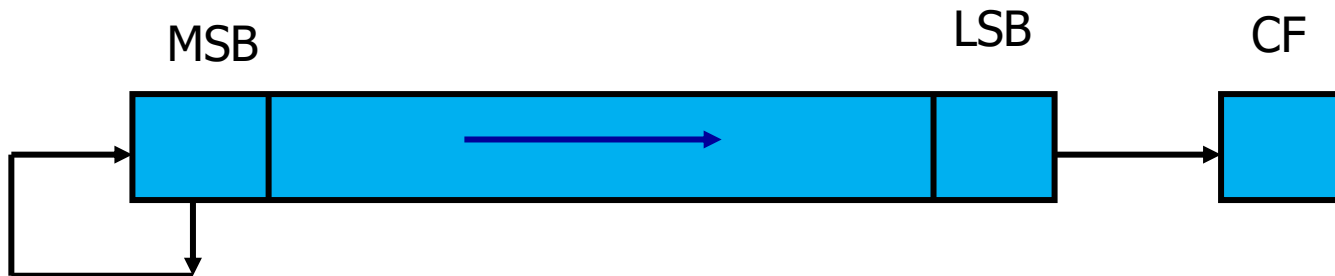
- ☐ Cú pháp: SAR Đích, CL (với số lần dịch lớn hơn 1)

- SAR Đích, 1

- hoặc SAR Đích, số lần dịch (80286 trở lên)

- ☐ Thực hiện: dịch phải đích CL bit

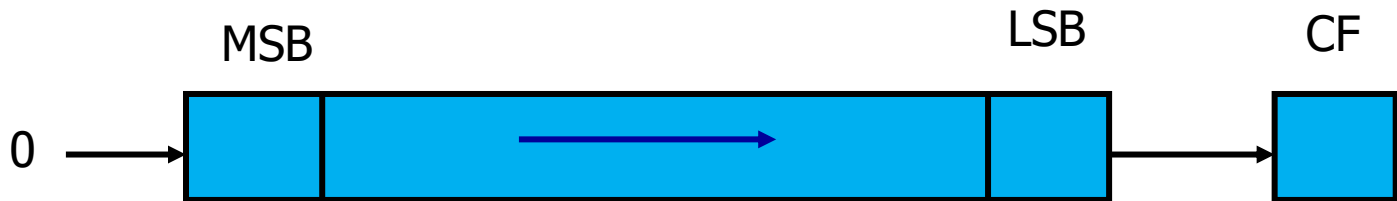
- ☐ Lệnh này thay đổi cờ SF, ZF, PF, CF mang giá trị của MSB



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
hoặc 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



Chú ý:

Trong các lệnh dịch và quay, toán hạng không được là thanh ghi đoạn

Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
- Mô tả tập lệnh của 8086
 - ☐ Các lệnh di chuyển dữ liệu
 - ☐ Các lệnh số học và logic
 - ☒ 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
 - ⇒ Lệnh gọi chương trình con phục vụ ngắt INT và IRET
- Lập trình hợp ngữ với 8086

Lệnh nhảy không điều kiện JMP

- Dùng để nhảy tới một địa chỉ trong bộ nhớ
- 3 loại: nhảy ngắn, gần và xa

□ Lệnh nhảy ngắn (short jump)

⇒ Độ dài lệnh 2 bytes:

E B	Độ lệch
-----	---------

⇒ Phạm vi nhảy: -128 đến 127 bytes so với lệnh tiếp theo lệnh JMP

⇒ Thực hiện: $IP = IP + \text{độ lệch}$

⇒ Ví dụ:

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

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:

XOR BX, BX

Nhan: MOV AX, 1

ADD AX, BX

JMP NEAR Nhan

XOR CX, CX

MOV AX, 1

ADD AX, BX

JMP NEAR PTR BX

XOR CX, CX

MOV AX, 1

ADD AX, BX

JMP WORD PTR [BX]

Thực hiện: $IP = IP + \text{độ lệch}$

$IP = BX$

$IP = [BX+1] [BX]$

E 9

Độ lệchLo

Độ lệchHi

Nhảy gián tiếp

Lệnh nhảy không điều kiện JMP

❑ Lệnh nhảy xa (far jump)

⇒ Độ dài lệnh 5 bytes đối với nhảy tới nhãn:



⇒ 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
```

```
XOR CX, CX
```

```
MOV AX, 1
```

```
ADD AX, BX
```

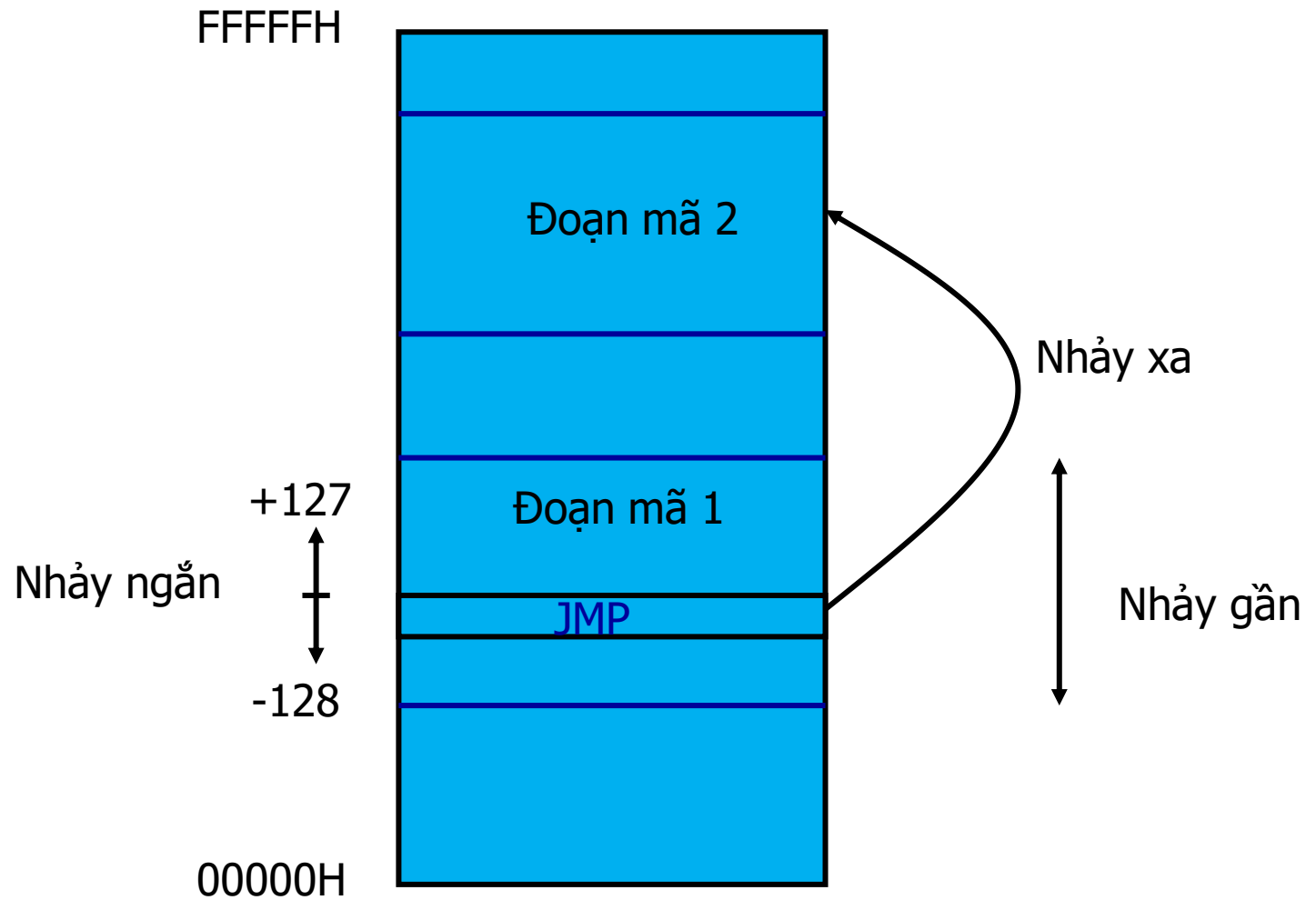
```
JMP DWORD PTR [BX]
```

IP = [BX+1][BX]

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

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

Tóm tắt lệnh JMP



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
- Ví dụ:

Nhan1: XOR BX, BX

Nhan2: MOV AX, 1

CMP AL, 10H

JNE Nhan1

JE Nhan2

Thực hiện: $IP = IP + \text{độ dịch}$

Lệnh lặp LOOP

- LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ
- Là lệnh phối hợp giữa DEC CX và JNZ

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

Lệnh 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

Tong PROC NEAR

ADD AX, BX

ADD AX, CX

RET

Tong ENDP

...

CALL Tong

Cất IP vào ngăn xếp
 $IP = IP + \text{dịch chuyển}$
RET: lấy IP từ ngăn xếp

Tong PROC NEAR

ADD AX, BX

ADD AX, CX

RET

Tong ENDP

...

MOV BX, OFFSET Tong

CALL BX

Cất IP vào ngăn xếp
 $IP = BX$
RET: lấy IP từ ngăn xếp

CALL WORD PTR [BX]

Cất IP vào ngăn xếp
 $IP = [BX+1] [BX]$
RET: lấy IP từ ngăn xếp

Lệnh CALL

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

Tong PROC FAR

ADD AX, BX

ADD AX, CX

RET

Tong ENDP

...

CALL Tong

CALL DWORD PTR [BX]

Cất CS vào ngăn xếp
Cất IP vào ngăn xếp
IP=IP của Tong
CS =CS của Tong
RET: lấy IP từ ngăn xếp
lấy CS từ ngăn xếp

Cất CS vào ngăn xếp
Cất IP vào ngăn xếp
IP = [BX+1][BX]
CS= [BX+3][BX+2]
RET: lấy IP từ ngăn xếp
lấy CS từ ngăn xếp

Lệnh 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
 - ❑ 32 vector đầu dành riêng cho Intel
 - ❑ 224 vector sau dành cho người dùng
- Cú pháp: INT Number
- Ví dụ: INT 21H gọi CTCPVN của DOS

Lệnh 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

Bộ vi xử lý Intel 8088/8086

- Cấu trúc bên trong
- Mô tả tập lệnh của 8086
- Lập trình hợp ngữ 8086

Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
 - ❑ Cú pháp của chương trình hợp ngữ
 - ❑ Dữ liệu cho chương trình
 - ❑ Biến và hằng
 - ❑ Khung của một chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

Cú pháp của chương trình hợp ngữ

1.	.Model Small	← khai báo kiểu kích thước bộ nhớ
2.	.Stack 100	← khai báo đoạn ngăn xếp
3.	.Data	← khai báo đoạn dữ liệu
4.	Tbao DB 'Chuoi da sap xep:', 10, 13	
5.	MGB DB 'a', 'Y', 'G', 'T', 'y', 'Z', 'U', 'B', 'D', 'E',	
6.	DB '\$'	
7.	.Code	← khai báo đoạn mã lệnh
8.	MAIN Proc	← bắt đầu chương trình chính
9.	MOV AX, @Data	;khởi đầu DS
10.	MOV DS, AX	
11.	MOV BX, 10	;BX: số phần tử của mảng
12.	LEA DX, MGB	;DX chỉ vào đầu mảng byte
13.	DEC BX	;số vòng so sánh phải làm
14.	LAP: MOV SI, DX	; SI chỉ vào đầu mảng
15.	MOV CX, BX	; CX số lần so của vòng so
16.	MOV DI, SI	; giá trị đầu là max
17.	MOV AL, [DI]	; AL chứa phần tử max
18.	TIMMAX:	
19.	INC SI	; chỉ vào phần tử bên cạnh
20.	CMP [SI], AL	; phần tử mới > max?
21.	JNG TIEP	; không, tìm max
22.	MOV DI, SI	; dùng, DI chỉ vào max
23.	MOV AL, [DI]	; AL chứa phần tử max
24.	TIEP: LOOP TIMMAX	; tìm max của một vòng so
25.	CALL DOICHO	; đổi cho max với số mới
26.	DEC BX	; số vòng so còn lại
27.	JNZ LAP	; làm tiếp vòng so mới
28.	MOV AH, 9	; hiển thị chuỗi đã sắp xếp
29.	MOV DX, Tbao	
30.	INT 21H	
31.	MOV AH, 4CH	; về DOS
32.	INT 21H	
33.	MAIN Endp	← kết thúc chương trình chính
34.	DOICHO Proc	← bắt đầu chương trình con
35.	PUSH AX	
36.	MOV AL, [SI]	
37.	XCHG AL, [DI]	
38.	MOV [SI], AL	
39.	POP AX	
40.	RET	
41.	DOICHO Endp	← kết thúc đoạn mã
42.	END MAIN	

Cú pháp của chương trình hợp ngữ

- Tên Mã lệnh Các toán hạng ; chú giải
- Chương trình dịch không phân biệt chữ hoa, chữ thường
- Trường tên:
 - ☐ chứa các nhãn, tên biến, tên thủ tục
 - ☐ độ dài: 1 đến 31 ký tự
 - ☐ tên không được có dấu cách, không bắt đầu bằng số
 - ☐ được dùng các ký tự đặc biệt: ? . @ _ \$ %
 - ☐ dấu . phải được đặt ở vị trí đầu tiên nếu sử dụng

Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
 - ☐ Cú pháp của chương trình hợp ngữ
 - ☒ **Dữ liệu cho chương trình**
 - ☐ Biến và hằng
 - ☐ Khung của một chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

Dữ liệu cho chương trình

- Dữ liệu:
 - ❑ các số hệ số 2: 0011B
 - ❑ hệ số 10: 1234
 - ❑ hệ số 16: 1EF1H, 0ABBAH
 - ❑ Ký tự, chuỗi ký tự: 'A', 'abcd'

Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
 - ☐ Cú pháp của chương trình hợp ngữ
 - ☐ Dữ liệu cho chương trình
 - ☒ **Biến và hằng**
 - ☐ Khung của một chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

Biến và hằng

- DB (define byte): định nghĩa biến kiểu byte
- DW (define word): định nghĩa biến kiểu từ
- DD (define double word): định nghĩa biến kiểu từ kép

- Biến byte:

- ❑ Tên DB gia_trị_khởi đầu

- ❑ Ví dụ:

- ⇒ B1 DB 4

MOV AL, B1

- ⇒ B1 DB ?

LEA BX, B1

- ⇒ C1 DB '\$'

MOV AL, [BX]

- ⇒ C1 DB 34

Biến và hằng

- Biến từ:

☐ Tên DW gia_trị_khởi đầu

☐ Ví dụ:

⇒ W1 DW 4

⇒ W2 DW ?

- Biến mảng:

☐ M1 DB 4, 5, 6, 7, 8, 9

☐ M2 DB 100 DUP(0)

☐ M3 DB 100 DUP(?)

☐ M4 DB 4, 3, 2, 2 DUP (1, 2 DUP(5), 6)

☐ M4 DB 4, 3, 2, 1, 5, 5, 6, 1, 5, 5, 6

1300A		
13009		
13008	9	
13007	8	
13006	7	
13005	6	
13004	5	
13003	4	M1
13002		
13001		
13000		

Biến và hằng

- Biến mảng 2 chiều:

$$\begin{bmatrix} 1 & 6 & 3 \\ 4 & 2 & 5 \end{bmatrix}$$

□ M1 DB 1, 6, 3
DB 4, 2, 5

□ M2 DB 1, 4
DB 6, 2
DB 3, 5

1300A		
13009		
13008	5	
13007	2	
13006	4	
13005	3	
13004	6	
13003	1	M1
13002		
13001		
13000		

```
MOV AL, M1 ; copy 1 vao AL
MOV AH, M1[2]
MOV BX, 1
MOV SI, 1
MOV CL, M1[BX+SI]
MOV AX, Word Ptr M1[BX+SI+2]
MOV DL, M1[BX][SI]
```

Biến và hằng

- Biến kiểu xâu ký tự

- ☐ STR1 DB 'string'
- ☐ STR2 DB 73h, 74h, 72h, 69h, 6Eh, 67h
- ☐ STR3 DB 73h, 74h, 'r', 'i', 6Eh, 67h

- Hằng có tên

- ☐ Có thể khai báo hằng ở trong chương trình
- ☐ Thường được khai báo ở đoạn dữ liệu
- ☐ Ví dụ:
 - ⇒ CR EQU 0Dh ;CR là carriage return
 - ⇒ LF EQU 0Ah ; LF là line feed
 - ⇒ CHAO EQU 'Hello'

 - ⇒ MSG DB CHAO, '\$'

Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
 - ☐ Cú pháp của chương trình hợp ngữ
 - ☐ Dữ liệu cho chương trình
 - ☐ Biến và hằng
 - ☒ Khung của một chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

Khung của chương trình hợp ngữ

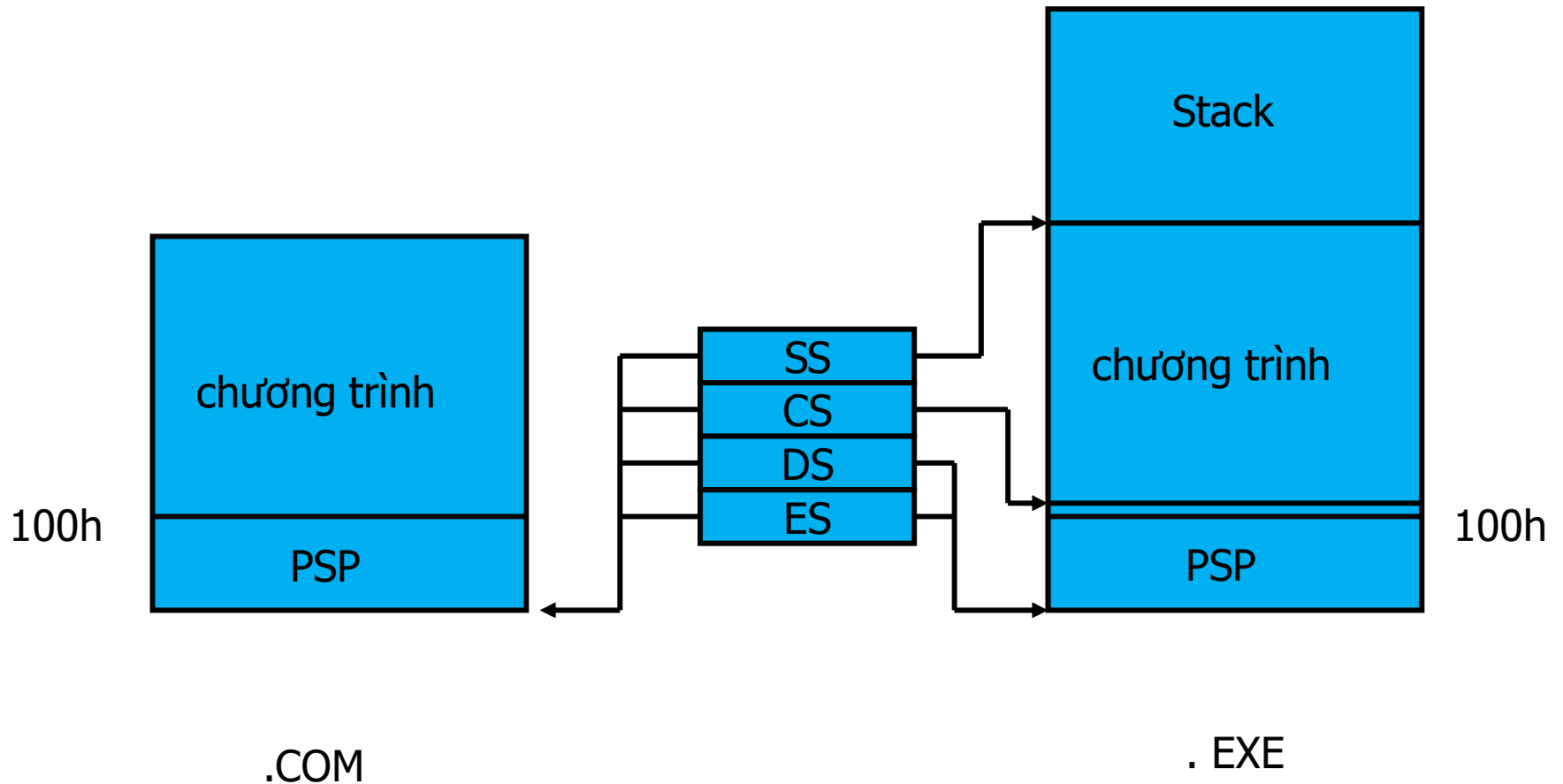
- Khai báo quy mô sử dụng bộ nhớ
 - ☐ .MODEL Kiểu kích thước bộ nhớ
 - ☐ Ví dụ: .Model Small

Kiểu	Mô tả
Tiny (hẹp)	mã lệnh và dữ liệu gói gọn trong một đoạn
Small (nhỏ)	mã lệnh nằm trong 1 đoạn, dữ liệu 1 đoạn
Medium (tB)	mã lệnh nằm trong nhiều đoạn, dữ liệu 1 đoạn
Compact (gọn)	mã lệnh nằm trong 1 đoạn, dữ liệu trong nhiều đoạn
Large (lớn)	mã lệnh nằm trong nhiều đoạn, dữ liệu trong nhiều đoạn, không có mảng nào lớn hơn 64 K
Huge (đồ sộ)	mã lệnh nằm trong nhiều đoạn, dữ liệu trong nhiều đoạn, các mảng có thể lớn hơn 64 K

Khung của chương trình hợp ngữ

- Khai báo đoạn ngăn xếp
 - ☐ .Stack kích thước (bytes)
 - ☐ Ví dụ:
 - ⇒ .Stack 100 ; khai báo stack có kích thước 100 bytes
 - ☐ Giá trị ngầm định 1KB
- Khai báo đoạn dữ liệu:
 - ☐ .Data
 - ☐ Khai báo các biến và hằng
- Khai báo đoạn mã
 - ☐ .Code

Khung của chương trình hợp ngữ



Khung của chương trình hợp ngữ

- Khung của chương trình hợp ngữ để dịch ra file .EXE

```
.Model    Small  
.Stack    100  
.Data  
          ;các định nghĩa cho biến và hằng  
.Code  
MAIN     Proc  
          ;khởi đầu cho DS  
          MOV     AX, @data  
          MOV     DS, AX  
          ;các lệnh của chương trình  
          ;trở về DOS dùng hàm 4CH của INT 21H  
          MOV     AH, 4CH  
          INT     21H  
MAIN     Endp  
          ;các chương trình con nếu có  
END MAIN
```

Khung của chương trình hợp ngữ

- Chương trình Hello.EXE

```
.Model      Small
.Stack      100
.Data

        CRLF      DB      13,10,'$'
        MSG        DB      'Hello! $'

.Code
MAIN      Proc
;khởi đầu cho DS
MOV       AX, @data
MOV       DS, AX
;về đầu dòng mới dùng hàm 9 của INT 21H
MOV       AH,9
LEA       DX, CRLF
INT       21H
;Hiển thị lời chào dùng hàm 9 của INT 21H
MOV       AH,9
LEA       DX, MSG
INT       21H
;về đầu dòng mới dùng hàm 9 của INT 21H
MOV       AH,9
LEA       DX, CRLF
INT       21H
;trở về DOS dùng hàm 4CH của INT 21H
MOV       AH, 4CH
INT       21H

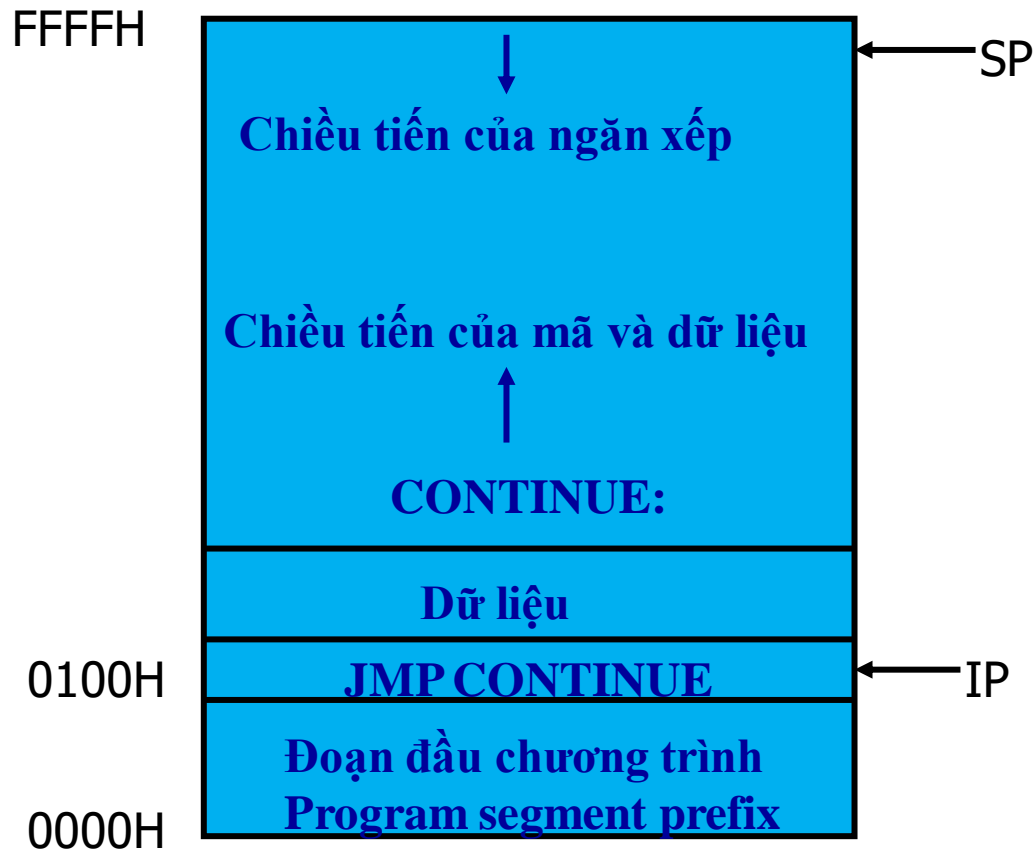
MAIN      Endp
END MAIN
```

Khung của chương trình hợp ngữ

- Khung của chương trình hợp ngữ để dịch ra file .COM

```
.Model Tiny  
.Code  
ORG 100h  
START: JMP CONTINUE  
;các định nghĩa cho biến và hằng  
CONTINUE:  
MAIN Proc  
;các lệnh của chương trình  
INT 20H ;trở về DOS  
MAIN Endp  
;các chương trình con nếu có  
END START
```

Khung của chương trình hợp ngữ



Khung của chương trình hợp ngữ

- Chương trình Hello.COM

```
.Model      Tiny
.Code

          ORG      100H
START: JMP CONTINUE

          CRLF      DB      13,10,'$'
          MSG       DB      'Hello! $'

CONTINUE:

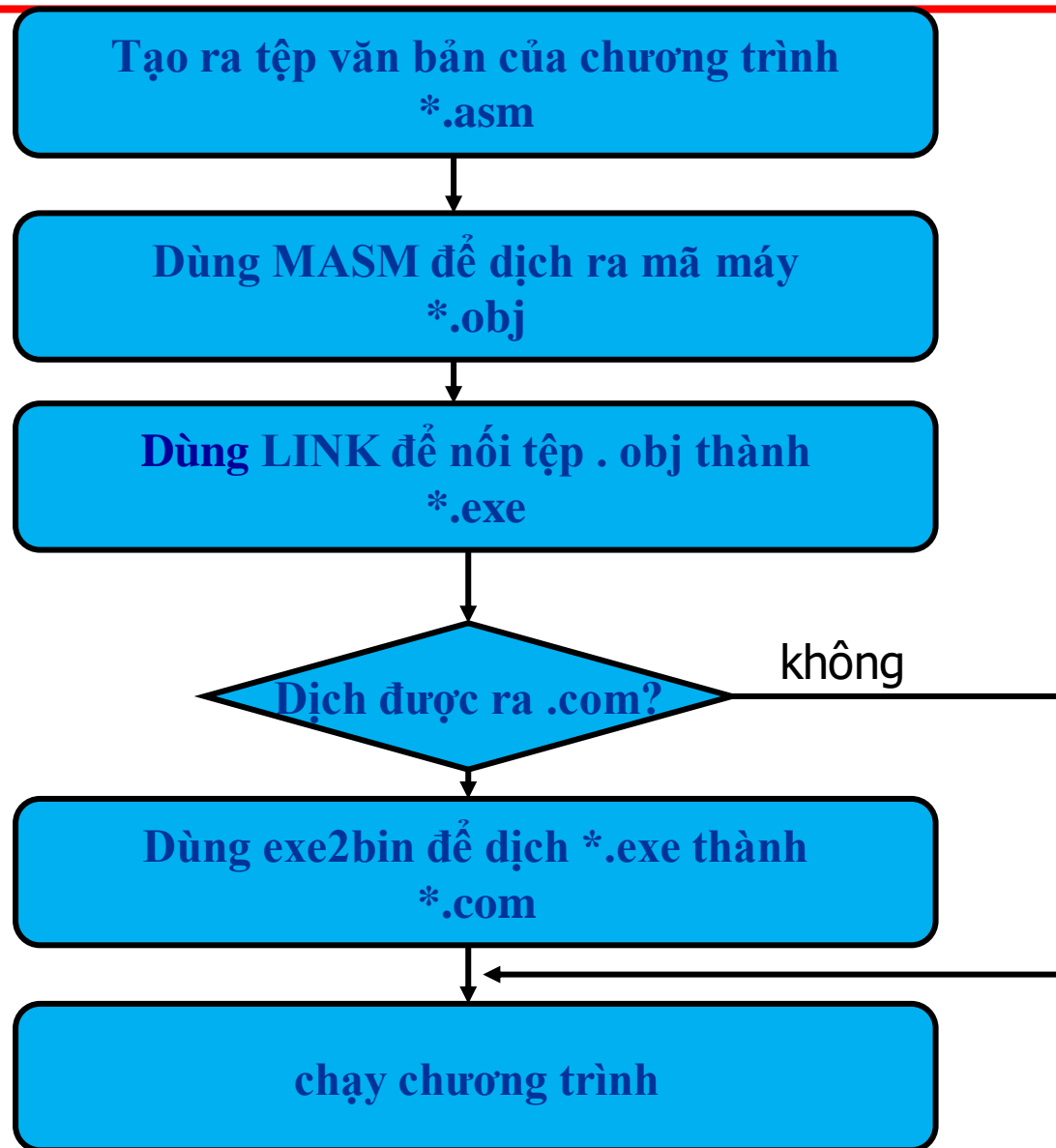
MAIN      Proc
          ;về đầu dòng mới dùng hàm 9 của INT 21H
          MOV       AH,9
          LEA       DX, CRLF
          INT       21H
          ;Hiển thị lời chào dùng hàm 9 của INT 21H
          MOV       AH,9
          LEA       DX, MSG
          INT       21H
          ;về đầu dòng mới dùng hàm 9 của INT 21H
          MOV       AH,9
          LEA       DX, CRLF
          INT       21H
          ;trở về DOS
          INT       20H

MAIN      Endp
END START
```

Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

Cách tạo một chương trình hợp ngữ



Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
 - ☐ Cấu trúc lựa chọn
 - ☐ Cấu trúc lặp
- Một số chương trình cụ thể

Cấu trúc lựa chọn If-then

- If (điều_kiện) then (công_việc)
- Ví dụ: Gán cho BX giá trị tuyệt đối của AX

```
; If AX<0
CMP      AX, 0           ; AX<0 ?
JNL End_if              ; không, thoát ra

; then
NEGAX                      ; đúng, đảo dấu
End_if: MOV      BX, AX   ;gán
```

Cấu trúc lựa chọn If-then-else

- If (điều_kiện) then (công_việc1)
else (công_việc2)
- Ví dụ: if $AX < BX$ then $CX = 0$ else $CX = 1$

```
; if AX < BX
CMP      AX, BX      ; AX < BX ?
JL       Then_       ; đúng, CX=0

;else
MOV      CX, 1        ; sai, CX=1
JMP      End_if

Then_:   MOV      CX, 0;
End_if:
```

Cấu trúc lựa chọn case

- case Biểu thức

Giá trị 1: công việc 1

Giá trị 2: công việc 2

...

Giá trị N: công việc N

End Case

- Ví dụ:

Nếu $AX < 0$ thì $CX = -1$

Nếu $AX = 0$ thì $CX = 0$

Nếu $AX > 0$ thì $CX = 1$

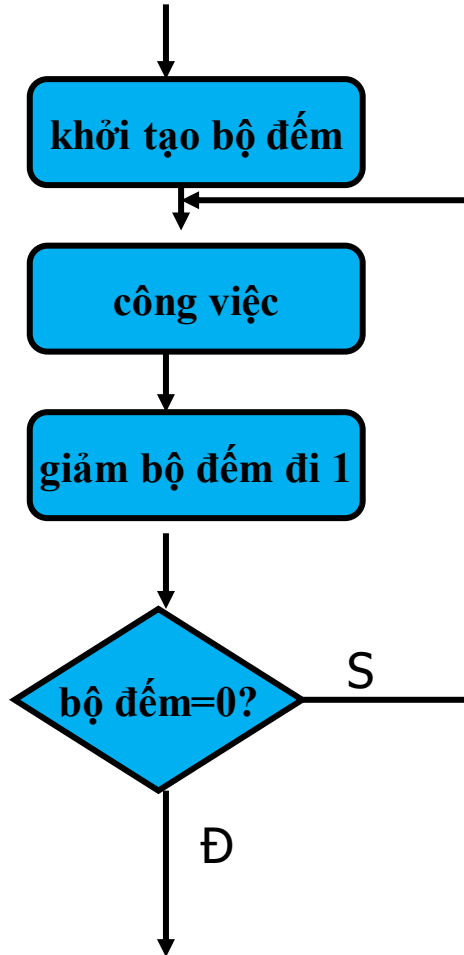
```
CMP      AX, 0      ;
JL       AM         ; AX < 0
JE       Khong      ; AX = 0
JG       DUONG      ; AX > 0
AM:      MOV        CX, -1
          JMP        End_case

Khong:   MOV        CX, 0
          JMP        End_case

DUONG:   MOV        CX, 1
End_case:
```

Cấu trúc lặp FOR-DO

- for (số lần lặp) do (công việc)

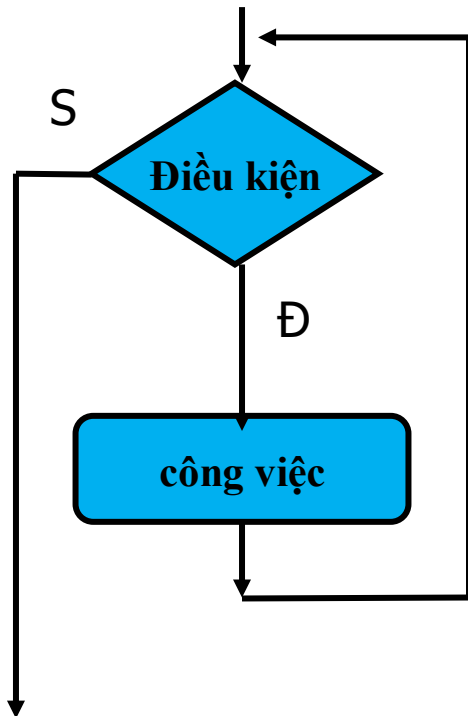


ví dụ: Hiển thị một dòng ký tự \$ trên màn hình

```
MOV CX, 80      ;số lần lặp
MOV AH,2        ;hàm hiển thị
MOV DL,'$'      ;DL chứa ký tự cần hiển thị
HIEN: INT 21H    ; Hiển thị
      LOOP HIEN
End_for
```

Cấu trúc lặp While-DO

- while (điều kiện) do (công việc)

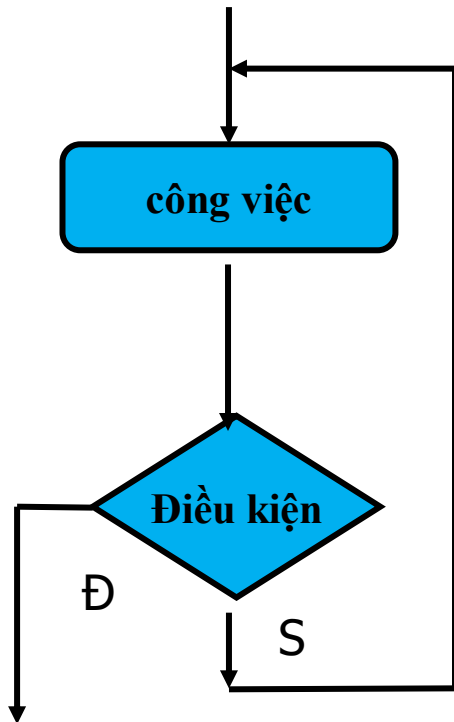


ví dụ: đếm số ký tự đọc được từ bàn phím,
khi gặp ký tự CR thì thôi

XOR CX, CX	;CX=0
MOV AH,1	;hàm đọc ký tự từ bàn phím
TIEP:	
INT 21H	; đọc một ký tự vào AL
CMP AL, 13	; đọc CR?
JE End_while	; đúng, thoát
INC CX	; sai, thêm 1 ký tự vào tổng
JMP TIEP	; đọc tiếp
End_while:	

Cấu trúc lặp Repeat-until

- Repeat (công việc) until (điều kiện)



ví dụ: đọc từ bàn phím cho tới khi gặp ký tự CR thì thôi

```
MOV  AH,1           ;hàm đọc ký tự từ bàn phím
TIEP:
INT   21H           ; đọc một ký tự vào AL
CMP   AL, 13        ; đọc CR?
JNE   TIEP          ; chưa, đọc tiếp
End_:
```


Lập trình hợp ngữ với 8086

- Giới thiệu khung của chương trình hợp ngữ
- Cách tạo và chạy một chương trình hợp ngữ trên máy IBM PC
- Các cấu trúc lập trình cơ bản thực hiện bằng hợp ngữ
- Một số chương trình cụ thể

Xuất nhập dữ liệu

- 2 cách:

- ❑ Dùng lệnh IN, OUT để trao đổi với các thiết bị ngoại vi
 - ⇒ phức tạp vì phải biết địa chỉ cổng ghép nối thiết bị
 - ⇒ Các hệ thống khác nhau có địa chỉ khác nhau
- ❑ Dùng các chương trình con phục vụ ngắt của DOS và BIOS
 - ⇒ đơn giản, dễ sử dụng
 - ⇒ không phụ thuộc vào hệ thống

- Ngắt 21h của DOS:

- ❑ Hàm 1: đọc 1 ký tự từ bàn phím
 - ⇒ Vào: AH=1
 - ⇒ Ra: AL=mã ASCII của ký tự, AL=0 khi ký tự là phím chức năng
- ❑ Hàm 2: hiện 1 ký tự lên màn hình
 - ⇒ Vào: AH=2
 - DL=mã ASCII của ký tự cần hiển thị
- ❑ Hàm 9: hiện chuỗi ký tự với \$ ở cuối lên màn hình
 - ⇒ Vào: AH=9
 - DX=địa chỉ lệch của chuỗi ký tự cần hiển thị
- ❑ Hàm 4CH: kết thúc chương trình loại .exe
 - ⇒ Vào: AH=4CH

Một số chương trình cụ thể

- Ví dụ 1: Lập chương trình yêu cầu người sử dụng gõ vào một chữ cái thường và hiển thị dạng chữ hoa và mã ASCII dưới dạng nhị phân của chữ cái đó lên màn hình
 - Ví dụ:
 - ⇒ Hay nhập vào một chữ cái thường: a
 - ⇒ Mã ASCII dưới dạng nhị phân của a là: 11000001
 - ⇒ Dạng chữ hoa của a là: A
- Ví dụ 2: Đọc từ bàn phím một số hệ hai (dài nhất là 16 bit), kết quả đọc được để tại thanh ghi BX. Sau đó hiển thị nội dung thanh ghi BX ra màn hình.
- Ví dụ 3: Nhập một dãy số 8 bit ở dạng thập phân, các số cách nhau bằng 1 dấu cách và kết thúc bằng phím Enter. Sắp xếp dãy số theo thứ tự tăng dần và in dãy số đã sắp xếp ra màn hình.

Một số chương trình cụ thể

- Ví dụ 4: Viết chương trình cho phép nhập vào kích thước $M \times N$ và các phần tử của một mảng 2 chiều gồm các số thập phân 8 bit.
 - ⇒ Tìm số lớn nhất và nhỏ nhất của mảng, in ra màn hình
 - ⇒ Tính tổng các phần tử của mảng và in ra màn hình
 - ⇒ Chuyển thành mảng $N \times M$ và in mảng mới ra màn hình

Hãy nhập giá trị M=

Hãy nhập giá trị N=

Nhập phần tử [1,1]=

Nhập phần tử [1,2]

.....

Số lớn nhất là phần tử [3,4]=15

Số nhỏ nhất là phần tử [1,2]=2

Tổng=256

...

PHỤ LỤC

Bộ vi xử lý Intel 8088/8086

- Các chế độ địa chỉ của 8086
 - ☐ Chế độ địa chỉ thanh ghi
 - ☐ Chế độ địa chỉ tức thì
 - ☐ Chế độ địa chỉ trực tiếp
 - ☐ Chế độ địa chỉ gián tiếp qua thanh ghi
 - ☐ Chế độ địa chỉ tương đối cơ sở
 - ☐ Chế độ địa chỉ tương đối chỉ số
 - ☐ Chế độ địa chỉ tương đối chỉ số cơ sở
- Cách mã hoá lệnh của 8086

Chế độ địa chỉ thanh ghi (Register Addressing Mode)

- Dùng các thanh ghi như là các toán hạng
- Tốc độ thực hiện lệnh cao
- Ví dụ:
 - ☐ MOV BX, DX ; Copy nội dung DX vào BX
 - ☐ MOV AL, BL ; Copy nội dung BL vào AL
 - ☐ MOV AL, BX ; không hợp lệ vì các thanh ghi có kích thước khác nhau
 - ☐ MOV ES, DS ; không hợp lệ (segment to segment)
 - ☐ MOV CS, AX ; không hợp lệ vì CS không được dùng làm thanh ghi đích
 - ☐ ADD AL, DL ; Cộng nội dung AL và DL rồi đưa vào AL

Chế độ địa chỉ tức thì (Immediate Addressing Mode)

- Toán hạng đích là thanh ghi hoặc ô nhớ
- Toán hạng nguồn là hằng số
- Dùng để nạp hằng số vào thanh ghi (trừ thanh ghi đoạn và thanh cờ) hoặc vào ô nhớ trong đoạn dữ liệu DS
- Ví dụ:
 - ☐ MOV BL, 44 ; Copy số thập phân 44 vào thanh ghi BL
 - ☐ MOV AX, 44H ; Copy 0044H vào thanh ghi AX
 - ☐ MOV AL, 'A' ; Copy mã ASCII của A vào thanh ghi AL
 - ☐ MOV DS, 0FF0H ; không hợp lệ
 - ☐ MOV AX, 0FF0H ;
 - ☐ MOV DS, AX ;
- ☐ MOV [BX], 10 ; copy số thập phân 10 vào ô nhớ DS:BX

Chế độ địa chỉ trực tiếp (Direct Addressing Mode)

- Một toán hạng là địa chỉ ô nhớ chứa dữ liệu
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
 - ❑ `MOV AL, [1234H]` ; Copy nội dung ô nhớ có địa chỉ DS:1234 vào AL
 - ❑ `MOV [4320H], CX` ; Copy nội dung của CX vào 2 ô nhớ liên tiếp DS: 4320 và DS: 4321

Chế độ địa chỉ gián tiếp qua thanh ghi (Register indirect Addressing Mode)

- Một toán hạng là thanh ghi chứa địa chỉ của 1 ô nhớ dữ liệu
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
 - ❑ `MOV AL, [BX]` ; Copy nội dung ô nhớ có địa chỉ DS:BX vào AL
 - ❑ `MOV [SI], CL` ; Copy nội dung của CL vào ô nhớ có địa chỉ DS:SI
 - ❑ `MOV [DI], AX` ; copy nội dung của AX vào 2 ô nhớ liên tiếp DS: DI và DS: (DI +1)

Chế độ địa chỉ tương đối cơ sở (Based relative Addressing Mode)

- Một toán hạng là thanh ghi cơ sở BX, BP và các hằng số biểu diễn giá trị dịch chuyển
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
 - ❑ `MOV CX, [BX]+10` ; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+10 và DS:BX+11 vào CX
 - ❑ `MOV CX, [BX+10]` ; Cách viết khác của lệnh trên
 - ❑ `MOV AL, [BP]+5` ; copy nội dung của ô nhớ SS:BP+5 vào thanh ghi AL

Chế độ địa chỉ tương đối chỉ số (Indexed relative Addressing Mode)

- Một toán hạng là thanh ghi chỉ số SI, DI và các hằng số biểu diễn giá trị dịch chuyển
- Toán hạng kia chỉ có thể là thanh ghi
- Ví dụ:
 - ❑ `MOV AX, [SI]+10` ; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:SI+10 và DS:SI+11 vào AX
 - ❑ `MOV AX, [SI+10]` ; Cách viết khác của lệnh trên
 - ❑ `MOV AL, [DI]+5` ; copy nội dung của ô nhớ DS:DI+5 vào thanh ghi AL

Chế độ địa chỉ tương đối chỉ số cơ sở (Based Indexed relative Addressing Mode)

- Ví dụ:

- ☐ `MOV AX, [BX] [SI]+8` ; Copy nội dung 2 ô nhớ liên tiếp có địa chỉ `DS:BX+SI+8` và `DS:BX+SI+9` vào `AX`
- ☐ `MOV AX, [BX+SI+8]` ; Cách viết khác của lệnh trên
- ☐ `MOV CL, [BP+DI+5]` ; copy nội dung của ô nhớ `SS:BP+DI+5` vào thanh ghi `CL`

Tóm tắt các chế độ địa chỉ

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

Bỏ chế độ ngầm định thanh ghi đoạn (Segment override)

- Ví dụ:

- ☐ `MOV AL, [BX]`; Copy nội dung ô nhớ có địa chỉ DS:BX vào AL
- ☐ `MOV AL, ES:[BX]` ; Copy nội dung ô nhớ có địa chỉ ES:BX vào AL

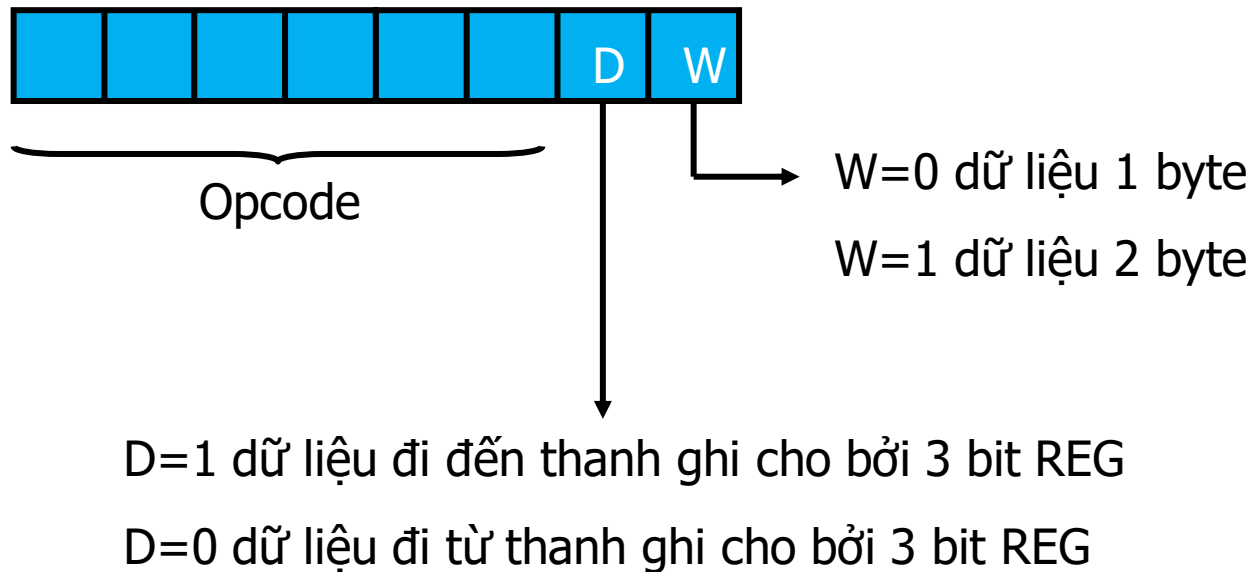
Bộ vi xử lý Intel 8088/8086

- Các chế độ địa chỉ của 8086
- Cách mã hoá lệnh của 8086

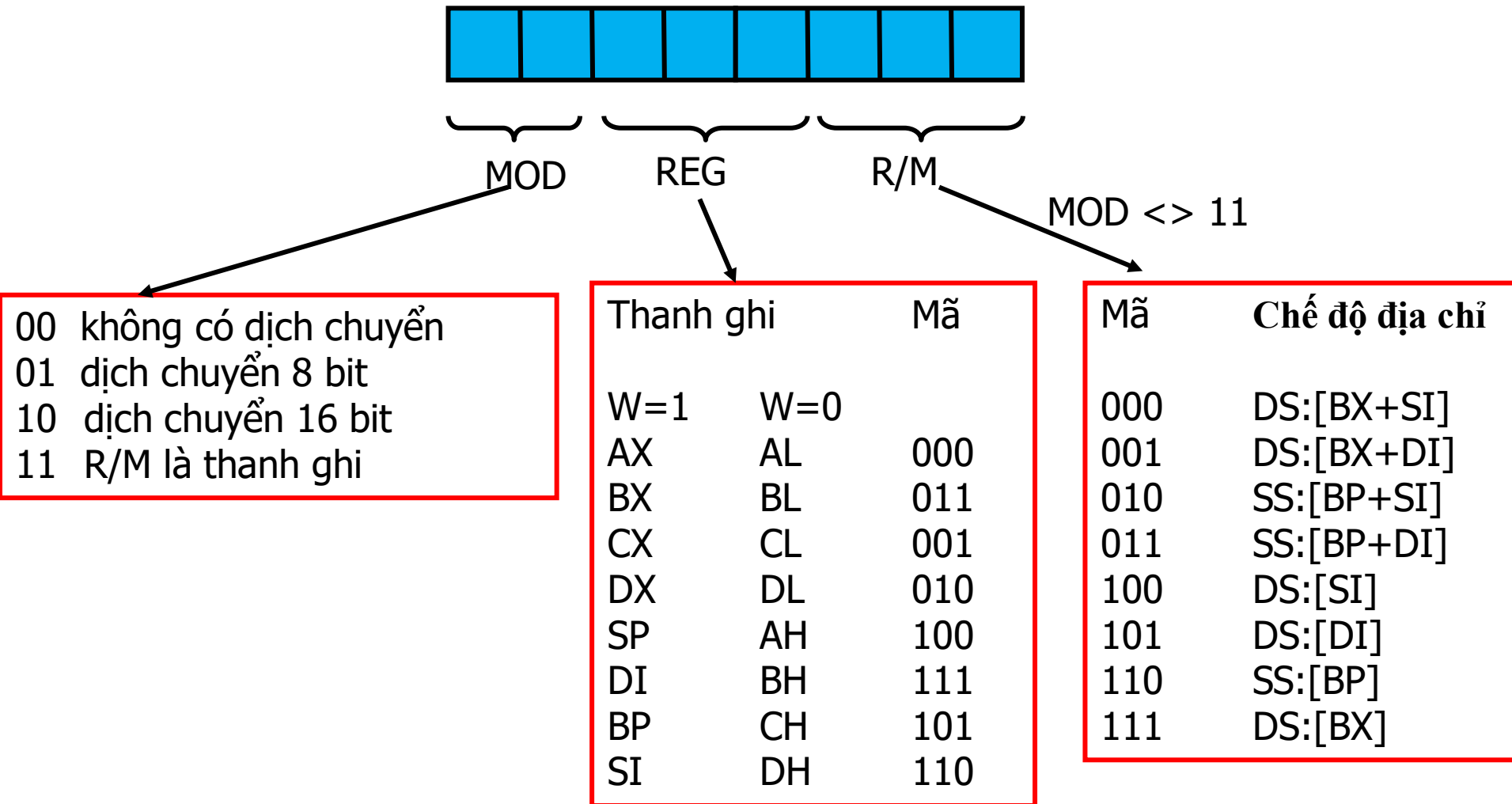
Cách mã hoá lệnh của 8086

Opcode 1-2 byte	MOD-REG-R/M 0-1 byte	Dịch chuyển 0-2 byte	Tức thì 0-2 byte
--------------------	-------------------------	-------------------------	---------------------

- Một lệnh có độ dài từ 1 đến 6 byte



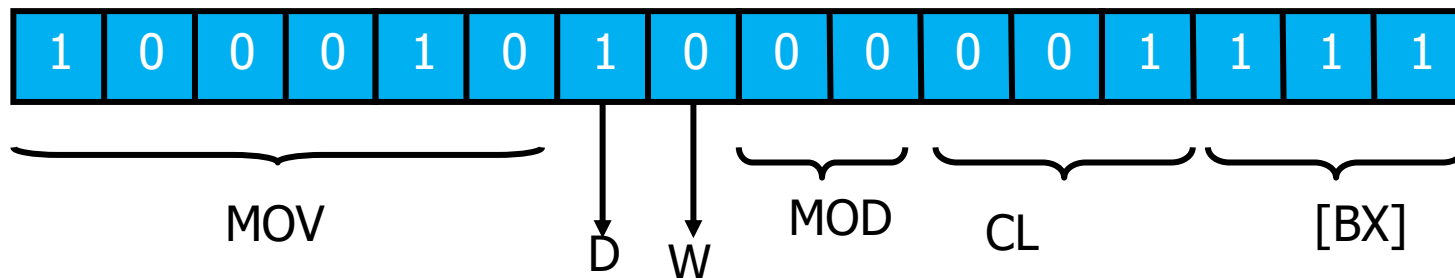
Cách mã hoá lệnh của 8086



Cách mã hoá lệnh của 8086

- Ví dụ: chuyển lệnh MOV CL, [BX] sang mã máy

- ☐ opcode MOV: 100010
- ☐ Dữ liệu là 1 byte: W=0
- ☐ Chuyển tới thanh ghi: D=1
- ☐ Không có dịch chuyển: MOD=00
- ☐ [BX] nên R/M=111
- ☐ CL nên REG=001



Ví dụ 2: chuyển lệnh MOV [SI+F3H], CL sang mã máy