

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

Kiến trúc máy tính

ThS. Đinh Xuân Trường

truongdx@ptit.edu.vn



Posts and Telecommunications
Institute of Technology
Faculty of Information Technology 1



CNTT1
Học viện Công nghệ Bưu chính Viễn thông

January 15, 2023

Mục tiêu Buổi 7

Kiến trúc bên trong 8086 / 8088

Sơ đồ khối

Các thành phần chức năng

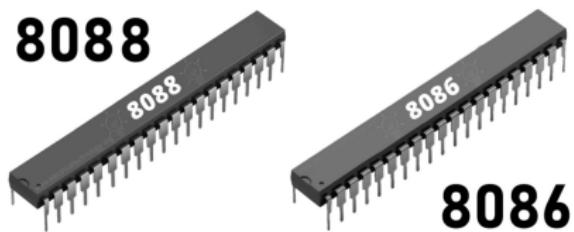
Phân đoạn bộ nhớ trong 8086/8088

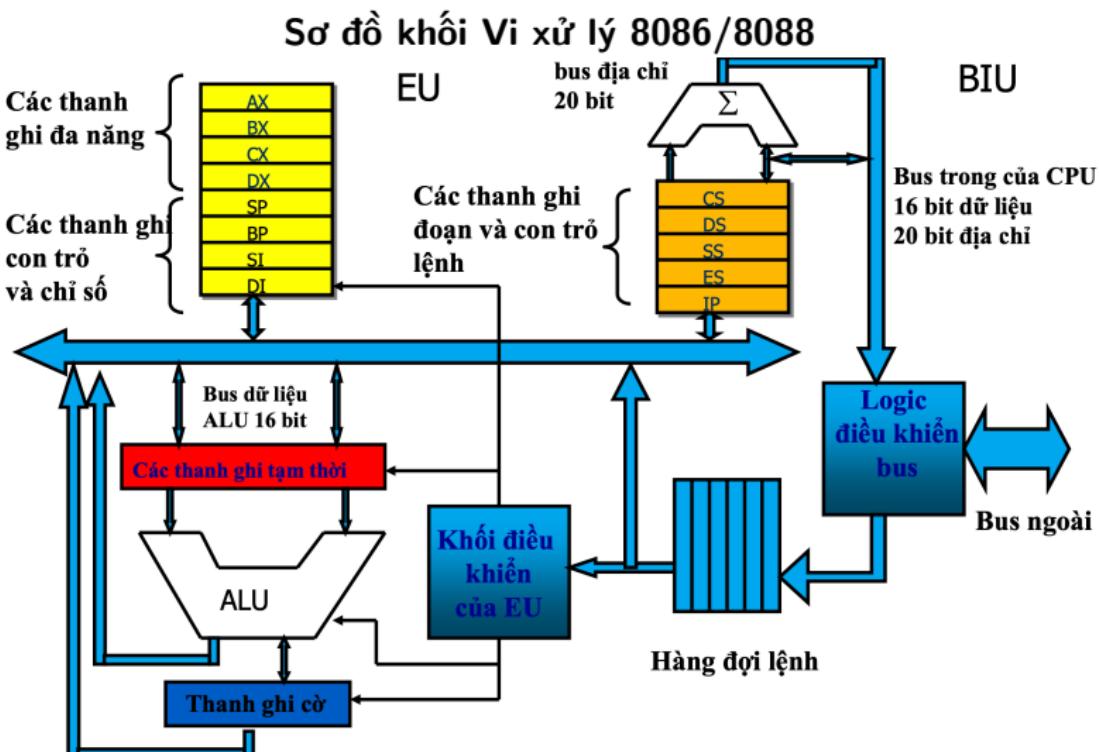
Tập lệnh của 8086 / 8088

Lệnh và cách mã hóa lệnh

Các chế độ địa chỉ của vi xử lý 8086/8088

Phân loại lệnh và mô tả tập lệnh của 8086/8088





► 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

► Đơn vị giao tiếp bus BIU (Bus Interface Unit)

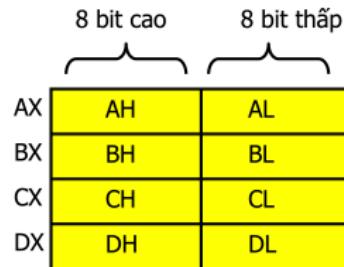
- Điều khiển bus hệ thống: đưa địa chỉ ra bus và trao đổi dữ liệu với bus:
 - ▶ Đưa ra địa chỉ
 - ▶ Đọc mã lệnh từ bộ nhớ
 - ▶ Đọc/ghi dữ liệu từ/vào bộ nhớ hoặc cổng vào/ra
- Các khôi:
 - ▶ Bộ cộng để tính địa chỉ
 - ▶ 4 thanh ghi đoạn 16-bit: CS, DS, SS, ES
 - ▶ Bộ đếm chương trình/con trỏ lệnh 16-bit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

Các thanh ghi của đa năng 8086/8088

► 4 thanh ghi 16 bits:

- **AX (accumulator)**: Thanh ghi tổng, thường dùng để lưu kết quả
- **BX (base)**: Thanh ghi cơ sở, thường dùng chứa địa chỉ ô nhớ
- **CX (count)**: Thanh ghi đếm, thường dùng làm con đếm cho các lệnh lặp (Loop), chứa số lần dịch hoặc quay trong các lệnh dịch và quay thanh ghi
- **DX (data)**: Thanh ghi dữ liệu, cùng AX chứa dữ liệu trong các phép tính nhân/chia số 16 bit hoặc chứa địa chỉ cổng trong các lệnh vào ra dữ liệu trực tiếp (IN/OUT)

► Hoặc 8 thanh ghi 8 bits: AH AL, BH, BL, CH, CL, DH, DL



Các thanh ghi con trỏ và chỉ số:

SP

BP

SI

DI

Stack Pointer

Base Pointer

Source Index

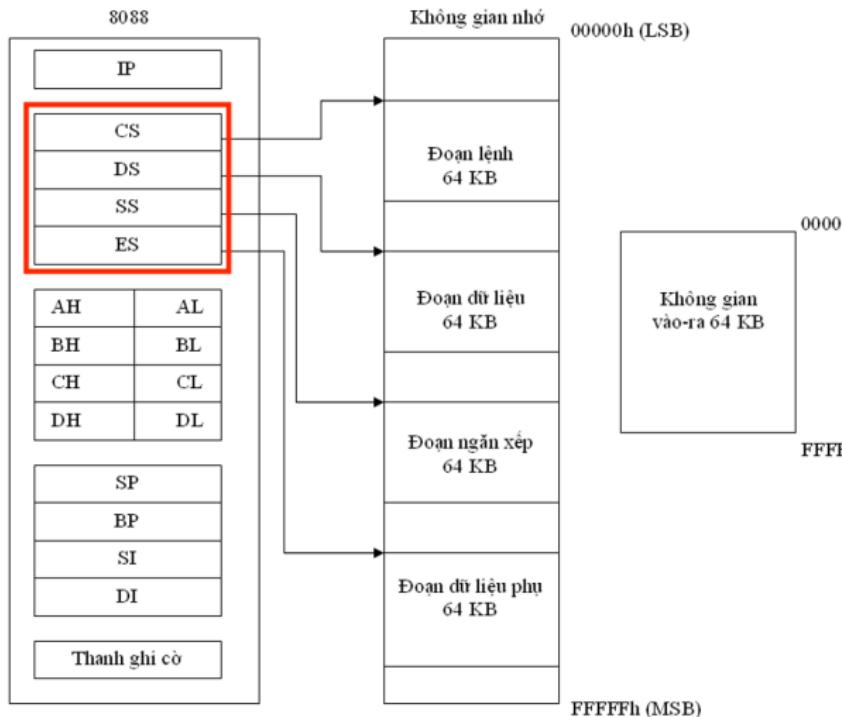
Destination Index

- ▶ **SP (Stack Pointer)**: con trỏ ngắn xếp. SP luôn chứa địa chỉ đỉnh ngăn xếp $SS:SP$
- ▶ **BP (Base Pointer)**: Con trỏ cơ sở, chứa địa chỉ của dữ liệu trong đoạn ngắn xếp SS hoặc các đoạn khác $SS:BP$
- ▶ **SI (Source Index)**: Thanh ghi chỉ số nguồn. SI thường dùng chứa địa chỉ ô nhớ nguồn ở đoạn dữ liệu DS như trong các lệnh chuỗi trong các thao tác chuyển dữ liệu $DS:SI$
- ▶ **DI (Destination Index)**: Thanh ghi chỉ số đích. DI thường dùng chứa địa chỉ ô nhớ đích ở đoạn dữ liệu DS trong các thao tác chuyển dữ liệu $DS:DI$
- ▶ SI và DI có thể được sử dụng như thanh ghi đa năng
- ▶ 80386 trở lên 32 bit: EIP, EBP, ESP, EDI, ESI

Cấu tạo và các thành phần của 8086 / 8088 (cont.)

Các thanh ghi đa năng

4 thanh ghi đoạn:



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.

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;
 - Thanh ghi IP (con trỏ lệnh) chứa địa chỉ offset của lệnh tiếp theo.
 - CS:IP chứa địa chỉ logic của lệnh tiếp theo đó.
- ▶ 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.

Các bit thanh ghi cờ trạng thái của VSL 8086/8088



► Các cờ trạng thái:

- CF (Carry): cờ nhớ. $CF = 1$ là phép tính có nhớ;
 - ▶ $CF = 0$ là phép tính không nhớ. Nếu phép cộng có nhớ ra khỏi bit cao nhất hay phép toán trừ có mượn ra khỏi bit cao nhất thì CF được thiết lập (báo tràn với số nguyên không dấu).
- AF (Auxiliary): cờ nhớ phụ. $AF = 1$ là có nhớ phụ;
 - ▶ $AF = 0$ là không nhớ phụ. Nếu phép cộng có nhớ từ bit 3 sang bit 4 hoặc phép trừ có mượn từ bit 3 sang bit 4 thì cờ AF được thiết lập.
- PF (Parity): cờ chẵn lẻ. $PF = 1$ khi tổng số bit 1 trong kết quả là lẻ và $PF = 0$ khi tổng số bit 1 trong kết quả là chẵn
- OF (Overflow): cờ tràn. $OF = 1$ khi kết quả bị tràn số. Nếu cộng 2 số cùng dấu mà kết quả có dấu ngược lại thì OF được thiết lập (báo tràn với số nguyên có dấu).

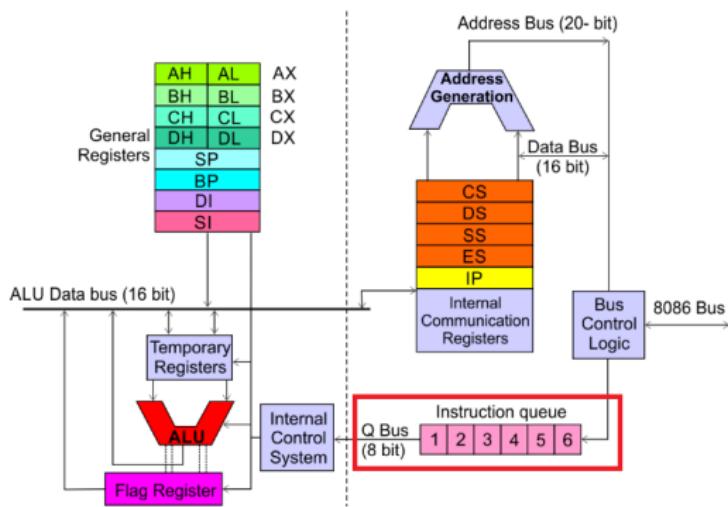
- ZF (Zero): cờ zero. ZF = 1 khi kết quả bằng 0; ngược lại ZF = 0
- SF (Sign): cờ dấu. SF = 1 khi kết quả âm.

► Các cờ điều khiển:

- DF (Direction): cờ hướng, chỉ hướng tăng giảm địa chỉ với các lệnh chuyển dữ liệu.
 - ▶ DF = 0 là địa chỉ tăng.
 - ▶ DF = 1 là địa chỉ giảm.
- TF (Trap/Trace): cờ bẫy/lần vết, được dùng khi gõ rồi chương trình. TF = 1 là CPU ở chế độ chạy từng lệnh (chế độ gõ rồi chương trình).
- IF (Interrupt): cờ ngắt.
 - ▶ Nếu IF = 1 thì bộ vi xử lý cho phép ngắt với yêu cầu ngắt đưa đến chân tín hiệu INTR (Interrupt Request) của bộ vi xử lý.
 - ▶ Nếu IF = 0 thì cầm ngắt.

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.

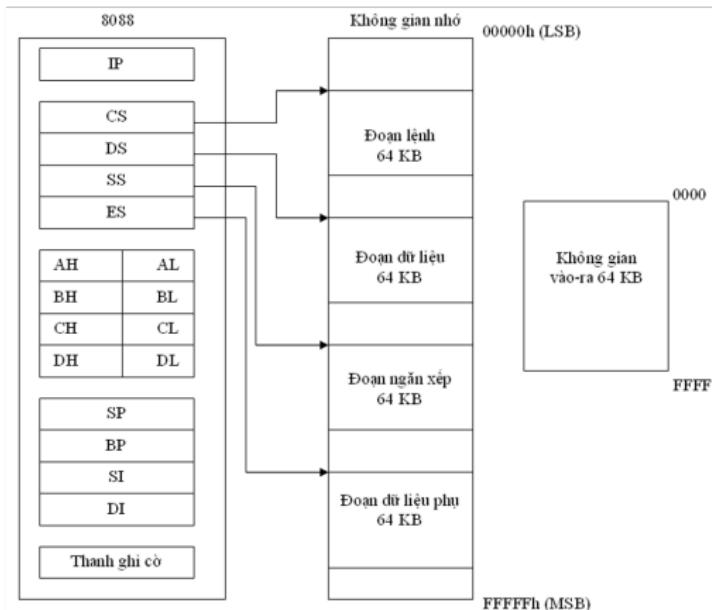


Tổ chức của bộ nhớ 1 Mbytes

VXL 8088/8086 sử dụng

20 bit để địa chỉ hoá bộ nhớ:

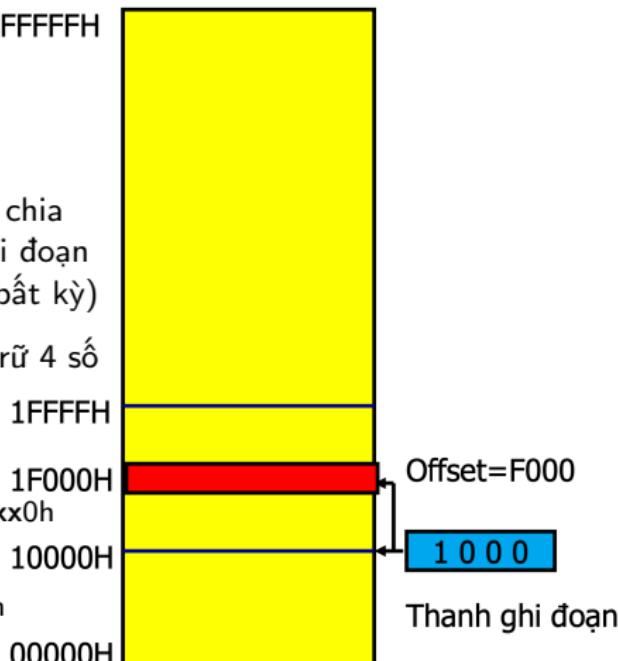
- ▶ Tổng dung lượng tối đa địa chỉ hoá của bộ nhớ là 2^{20} byte = 1MB;
- ▶ Địa chỉ vật lý được đánh từ 00000h đến FFFFFh.
- ▶ Không gian nhớ của 8088 được thành các đoạn nhớ (segment) có dung lượng 64 KB = 2^{16} Byte.



Tổ chức của bộ nhớ 1 Mbytes: FFFFFFFH

► Đoạn bộ nhớ (segment):

- Địa chỉ đầu của mỗi đoạn nhô chia hết cho 16 → Địa chỉ đầu mỗi đoạn có dạng xxxx0h (x: chữ Hexa bất kỳ)
- Địa chỉ đoạn nhô chỉ cần lưu trữ 4 số Hexa (16 bit cao)
 - ▶ Địa chỉ đoạn: xxxxh
 - ▶ Địa chỉ vật lý đầu đoạn: xxxx0h
 - ▶ Địa chỉ vật lý của ô nhớ cuối đoạn: xxxx0h + FFFFh



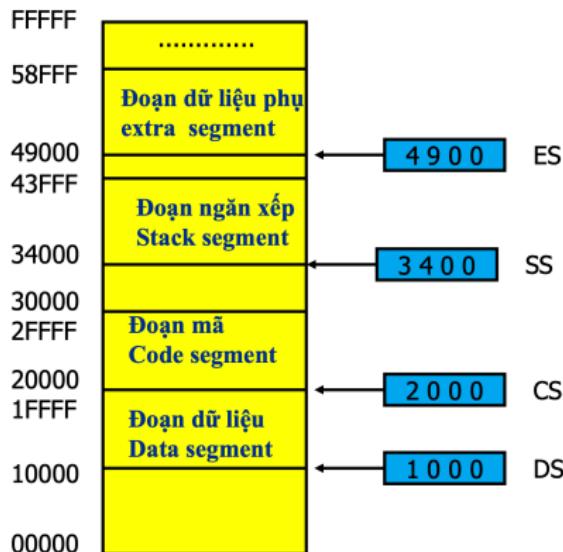
Ví dụ: Nếu địa chỉ đoạn là 1234h
thì địa chỉ vật lý của đầu đoạn nhô đó là 12340h

- ▶ Ô nhớ trong đoạn:
 - Địa chỉ đoạn do các thanh ghi đoạn quản lý.
 - Địa chỉ offset do các thanh ghi IP, BX, BP, SP, SI, DI quản lý.
 - ▶ Địa chỉ lệch: offset
 - ▶ Ô đầu tiên: offset: 0000
 - ▶ Ô cuối cùng: offset: FFFF
- ▶ Địa chỉ logic trong VXL 8086 có dạng: **Segment** (Địa chỉ đoạn 16 bit) : **offset** (16 bit)
- ▶ Địa chỉ vật lý (20 bit) = Địa chỉ đoạn X 10h + 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.

Phân đoạn bộ nhớ trong 8086/8088 (cont.)

VD: **CS:IP** chỉ ra địa chỉ lệnh sắp thực hiện trong đoạn mã.

Nếu CS=F000h và IP=FFF0h thì **CS:IP** tương ứng $F000h * 16 + FFF0h = F000h + FFF0h = FFFF0h$



- ▶ Người lập trình chỉ lập trình với địa chỉ logic, còn việc chuyển sang địa chỉ vật lý là do bộ vi xử lý thực hiện.

Chứa địa chỉ lệch (offset)

- ▶ Con trỏ lệnh IP (*instruction pointer*): chứa địa chỉ lệnh tiếp theo trong đoạn mã lệnh CS tương ứng **CS:IP**
- ▶ Con trỏ cơ sở BP (*Base Pointer*): chứa địa chỉ của dữ liệu trong đoạn ngắn xếp SS hoặc các đoạn khác tương ứng **SS:BP**
- ▶ Con trỏ ngắn xếp SP (*Stack Pointer*): chứa địa chỉ hiện thời của đỉnh ngắn xếp tương ứng **SS:SP**
- ▶ Chỉ số nguồn SI (*Source Index*): chứa địa chỉ dữ liệu nguồn trong đoạn dữ liệu DS trong các lệnh chuỗi tương ứng **DS:SI**
- ▶ Chỉ số đích (*Destination Index*): chứa địa chỉ dữ liệu đích trong đoạn dữ liệu DS trong các lệnh chuỗi tương ứng **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

► 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)

► 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 chu kỳ 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 ngắn
- Bus rỗi

Cách mã hoá lệnh:

- Dạng tổng quát của lệnh gồm 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

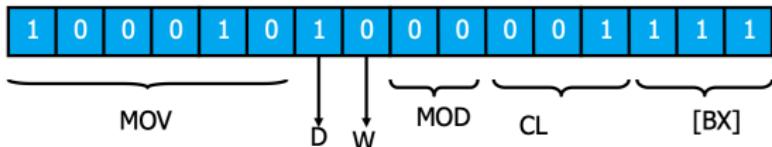
Mã lệnh	Đích, Gốc
---------	-----------

MOV

AX, 100

AX ← 100

Cách mã hóa các lệnh



- Một lệnh có độ dài từ 1 đến 6 byte:

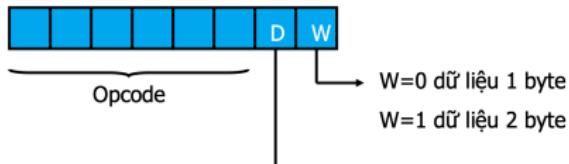


- **Opcode:** mã lệnh gồm 6 bít; Mã lệnh của MOV là 100010
- D: bít hướng, chỉ hướng vận chuyển dữ liệu;
 - D = 1: dữ liệu đi đến thanh ghi cho bởi 3 bit REG
 - D = 0: dữ liệu đi ra từ thanh ghi cho bởi 3 bit REG
- W: bít chỉ độ rộng toán hạng;
 - W=0: toán hạng 1 byte (8 bit);
 - W=1: toán hạng 2 bytes (16 bit)

Tập lệnh của 8086 / 8088 (cont.)

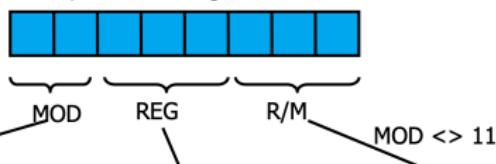
Lệnh và cách mã hóa lệnh

Cách mã hóa các lệnh



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

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



- 00 không có dịch chuyển
- 01 dịch chuyển 8 bit
- 10 dịch chuyển 16 bit
- 11 R/M là thanh ghi

Thanh ghi	Mã
W=1	W=0
AX	AL
BX	BL
CX	CL
DX	DL
SP	AH
DI	BH
BP	CH
SI	DH

Mã	Chế độ địa chỉ
000	DS:[BX+SI]
001	DS:[BX+DI]
010	SS:[BP+SI]
011	SS:[BP+DI]
100	DS:[SI]
101	DS:[DI]
110	SS:[BP]
111	DS:[BX]

Cách mã hóa các lệnh: Mã hóa các thanh ghi

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

Tập lệnh của 8086 / 8088 (cont.)

Lệnh và cách mã hóa lệnh

Cách mã hóa các lệnh: Mã hóa theo chế độ các lệnh

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

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 thanh ghi (Register Indirect Addressing Mode)
5. Chế độ địa chỉ tương đối cơ sở (Based Plus Displacement Addressing)
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)

Một số quy ước về lệnh của 8086/8088

- ▶ Assembly 8086/88 không phân biệt chữ hoa và chữ thường
- ▶ 100: Hằng số thập phân
- ▶ 80H, 0F9H: Hằng số hệ 16 bắt đầu bằng 1 chữ (A-F) thì thêm 0 vào đầu và thêm ký hiệu H (Hexa) ở cuối
- ▶ 0111B, 1000B: Hằng số nhị phân thêm ký hiệu B (Binary) ở cuối
- ▶ [AL]: Ô nhớ có địa chỉ được lưu trong thanh ghi AL
- ▶ [100]: Ô nhớ có địa chỉ 100
- ▶ [BX+100]: Ô nhớ có địa chỉ được lưu vào ô nhớ có địa chỉ BX + 100 ở đoạn dữ liệu

Chế độ địa chỉ thanh ghi:

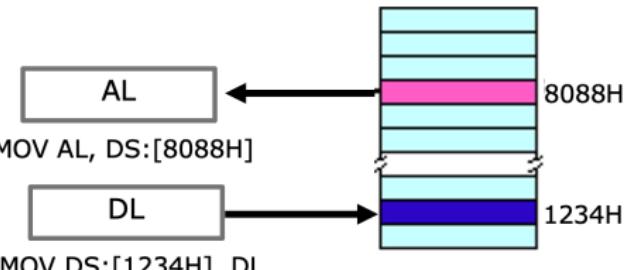
- ▶ Chế độ thanh ghi sử dụng các thanh ghi bên trong CPU như là các toán hạng để chứa dữ liệu cần thao tác.
- ▶ Cả toán hạng gốc và đích đều là các thanh ghi.
- ▶ Tốc độ thực hiện lệnh cao
- ▶ Ví dụ:
 - MOV BX, DX; BX <- DX
 - MOV DS, AX; DS <- AX
 - ADD AL, DL; AL <- AL + DL
 - MOV AL, BX ; *không hợp lệ* vì 2 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 dùng làm thanh ghi đích

Chế độ địa chỉ tức thì:

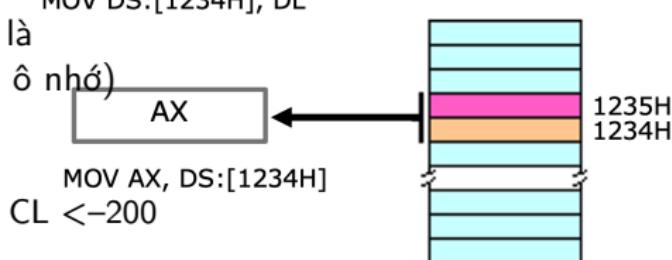
- ▶ Toán hạng đích là một thanh ghi hay một ô nhớ
- ▶ Toán hạng gốc là một hằng số
- ▶ Dùng để nạp hằng số vào thanh thi (trừ thanh ghi đoạn và thanh cờ) hoặc vào ô nhớ trong đoạn dữ liệu DS
- ▶ Ví dụ:
 - MOV CL, 200; CL <-200
 - MOV AX, 0ff0h; AX <- 0ff0h
 - MOV [BX], 200; Chuyển 200 vào ô nhớ có địa chỉ là DS:BX
 - MOV AL, 'A' ; Copy mã ASCII của A vào thanh ghi AL
 - MOV DS, 0FF0H ; *không hợp lệ*

Chế độ địa chỉ trực tiếp:

- Một toán hạng là một hằng số biểu diễn địa chỉ lêch (offset) của ô nhớ



- Toán hạng còn lại có thể là thanh ghi (không được là ô nhớ)

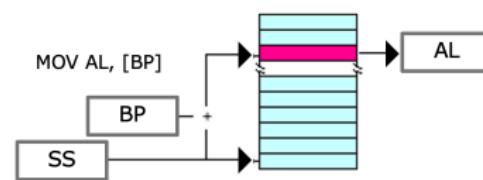
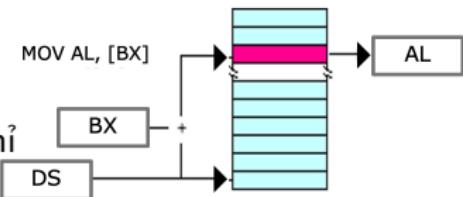


- Ví dụ:

- MOV CL, 200;
- MOV AX, 0ff0h;
- MOV [BX], 200; Chuyển 200 vào ô nhớ có địa chỉ là DS:BX
- MOV AL, 'A' ; Copy mã ASCII của A vào thanh ghi AL
- MOV DS, 0FF0H ; *không hợp lệ*

Chế độ địa chỉ gián tiếp qua thanh ghi:

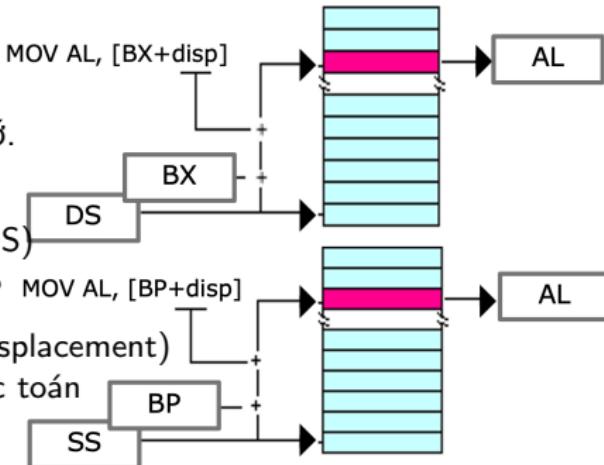
- ▶ Một toán hạng là một thanh ghi chứa địa chỉ lệch của ô nhớ
- ▶ Toán hạng còn lại có thể là thanh ghi
- ▶ Ví dụ:
 - $MOV AL, [BX]$; $AL \leftarrow [DS:BX]$
 - $MOV AL, [BP]$; $AL \leftarrow [SS:BP]$
 - $MOV [DI], AX$; Copy nội dung của AX vào 2 ô nhớ liên tiếp DS:DI và DS:(DI+1)



Chế độ địa chỉ tương đối cơ sở:

- Một toán hạng là địa chỉ của ô nhớ.

- Địa chỉ của ô nhớ được tạo bởi thanh ghi cơ sở như BX (đoạn DS) hoặc BP (đoạn SS) và 1 hằng số
- Hằng số là giá trị dịch chuyển(displacement) để tính địa chỉ hiệu dụng của các toán hạng trong vùng nhớ DS và SS.



- Toán hạng còn lại có thể là thanh ghi (Không được là ô nhớ)
- Ví dụ:

- $MOV AL, [BX+100]; \quad AL <- [DS: BX+100]$
- $MOV AL, [BP+200]; \quad AL <- [SS: BP+200]$
- $MOV AL, [BP]+200; \quad$ Cách viết khác của lệnh trên

Chế độ địa chỉ tương đối chỉ số:

- ▶ Một toán hạng là địa chỉ của ô nhớ.
 - Địa chỉ của ô nhớ tạo thanh ghi cơ sở SI hoặc DI và một hằng số.
 - Hằng số 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 (không được là ô nhớ)
- ▶ Ví dụ:
 - `MOV AL, [SI+100];` $AL \leftarrow [DS: SI+100]$
 - `MOV AL, [DI+200];` $AX \leftarrow [DS: DI+200]$
 - `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

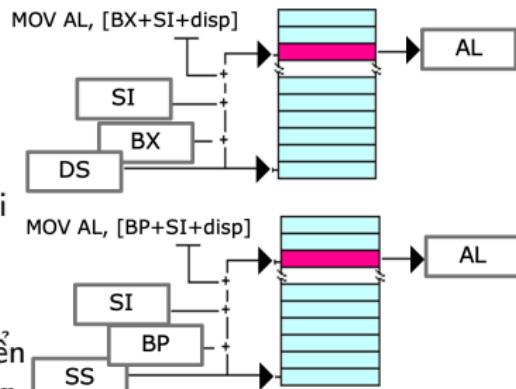
Chế độ địa chỉ tương đối chỉ số cơ sở:

- Một toán hạng là địa chỉ của ô nhớ

- Địa chỉ của ô nhớ tạo bởi các thanh ghi BX+SI/DI (đoạn DS) hoặc BP+SI/DI (đoạn SS) và một hằng số.
- Hằng số biểu diễn các giá trị dịch chuyển (displacement) 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 (không được là ô nhớ)

- $MOV AL, [BX+SI+100]; \quad AL \leftarrow [DS:BX+SI+100]$
- $MOV AX, [BX] [SI]+8; \quad$ Copy nội dung 2 ô nhớ liên tiếp có địa chỉ DS:BX+SI+8 và DS:BX+SI+9 vào AX
- $MOV AX, [BX+SI+8]; \quad$ Cách viết khác của lệnh trên



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

Quan hệ ngầm định giữa các thanh ghi đoạn và các thanh ghi lệnh:

Thanh ghi đoạn	CS	DS	ES	SS
Thanh ghi lệnh	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

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 hóa cho chương trình dịch
- Các lệnh có độ dài và thời gian thực hiện khác nhau -> giảm hiệu năng của cơ chế ông lệnh (pipeline)

► RISC (Reduced Instruction Set Computers)

- Tập lệnh tối thiểu: số lượng lệnh, các chế độ địa chỉ khuôn định lệnh và thời gian thực hiện
- Tăng được hiệu năng của cơ chế ông lệnh (pipeline)
- Dễ tối ưu hóa trong chương trình dịch
- Chương trình thường dài, cần nhiều bộ nhớ và tăng thời gian truy cập bộ nhớ

Phân loại tập lệnh của vi xử lý họ CISC

1. Vận chuyển Dữ liệu
2. Số học nguyên và logic
3. Dịch và quay
4. Chuyển điều khiển
5. Xử lý bit
6. Điều khiển hệ thống
7. Thao tác dấu phẩy động
8. Các lệnh của các đơn vị chức năng đặc biệt

Các lệnh vận chuyển dữ liệu

Vận chuyển dữ liệu giữa:

- ▶ Thanh ghi – thanh ghi;
- ▶ Thanh ghi – ô nhớ;
- ▶ hanh ghi – thiết bị vào ra.

Các lệnh:

- ▶ MOV, XCHG, POP, PUSH, POPF, PUSHF, IN, OUT
- ▶ LODSB, LODSW, STOSB, STOSW và các lệnh di chuyển chuỗi
MOVS, MOVS, MOVS

Các lệnh vận chuyển dữ liệu không ảnh hưởng đến các cờ trạng thái của thanh ghi cờ.

Các lệnh vận chuyển dữ liệu

MOV = Move:

Register/Memory to/from Register

7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
1 0 0 0 1 0 d w	mod reg r/m		
1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1
1 0 1 1 w reg		data	data if w = 1
1 0 1 0 0 0 0 w		addr-low	addr-high
1 0 1 0 0 0 1 w		addr-low	addr-high
1 0 0 0 1 1 1 0	mod 0 reg r/m		
1 0 0 0 1 1 0 0	mod 0 reg r/m		

Immediate to Register

Memory to Accumulator

Accumulator to Memory

Register/Memory to Segment Register

Segment Register to Register/Memory

PUSH = Push:

Register/Memory

1 1 1 1 1 1 1 1	mod 1 1 0 r/m
0 1 0 1 0 reg	
0 0 0 reg 1 1 0	

Register

Segment Register

POP = Pop:

Register/Memory

1 0 0 0 1 1 1 1	mod 0 0 0 r/m
0 1 0 1 1 reg	
0 0 0 reg 1 1 1	

Register

Segment Register

Các lệnh vận chuyển dữ liệu

XCHG = Exchange:

Register/Memory with Register

1000011w	mod reg r/m
----------	-------------

Register with Accumulator

10010reg

IN = Input from:

Fixed Port

1110010w	port
----------	------

Variable Port

1110110w

OUT = Output to:

Fixed Port

1110011w	port
----------	------

Variable Port

1110111w

XLAT = Translate Byte to AL

11010111

LEA = Load EA to Register

10001101	mod reg r/m
----------	-------------

LDS = Load Pointer to DS

11000101	mod reg r/m
----------	-------------

LES = Load Pointer to ES

11000100	mod reg r/m
----------	-------------

LAHF = Load AH with Flags

10011111

SAHF = Store AH into Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101

Các lệnh vận chuyển dữ liệu

Lệnh MOV:

- ▶ Dạng lệnh: MOV Đích, Gốc; Đích \leftarrow 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 \leftarrow 100
 - MOV [BX], AH; [DS:BX] \leftarrow AH
 - MOV DS, AX; DS \leftarrow AX

Các lệnh vận chuyển dữ liệu

Lệnh LODSB, LODSW:

- ▶ Dạng lệnh: LODSB; $AL \leftarrow [DS: SI]$
 $SI \leftarrow SI \pm 1$
- ▶ LODSW; $AL \leftarrow [DS: SI]$
 $SI \leftarrow SI \pm 2$
- ▶ 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 \leftarrow tăng, DF = 1 \leftarrow giảm.
- ▶ Ví dụ:
 - MOV SI, 1000; $SI \leftarrow 1000$
 - MOV [DS:SI], 200; $[DS:SI] \leftarrow 200$
 - CLD; DF $\leftarrow 0$
 - LODSB; $AL \leftarrow 200; SI \leftarrow SI + 1$

Các lệnh vận chuyển dữ liệu

Lệnh STOSB, STOSW:

- ▶ Dạng lệnh: STOSB; $[ES: DI] \leftarrow AL$
 $DI \leftarrow DI \pm 1$
- ▶ STOSW; $[ES: DI] \leftarrow AX$
 $DI \leftarrow DI \pm 2$
- ▶ 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 \leftarrow tăng, DF = 1 \leftarrow giảm.
- ▶ Ví dụ:
 - MOV DI, 1000; DI $\leftarrow 1000$
 - MOV AL, 200; AL $\leftarrow 200$
 - CLD; DF $\leftarrow 0$
 - STOSB; $[ES:DI] \leftarrow AL; DI \leftarrow DI + 1$

Các lệnh vận chuyển dữ liệu

Lệnh MOVSB, MOVSW:

- ▶ Dạng lệnh: $\text{MOVSB}; \quad [\text{ES: DI}] \leftarrow [\text{DS: SI}]$
 $\text{DI} \leftarrow \text{DI} \pm 1; \text{SI} \leftarrow \text{SI} \pm 1$
- ▶ $\text{MOVSW}; \quad [\text{ES: DI}] \leftarrow [\text{DS: SI}]$
 $\text{DI} \leftarrow \text{DI} \pm 2; \text{SI} \leftarrow \text{SI} \pm 2$
- ▶ 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:
 - $\text{MOV SI}, 1000; \quad \text{SI} \leftarrow 1000$
 - $\text{MOV DI}, 200; \quad \text{DI} \leftarrow 200$
 - $\text{CLD}; \quad \text{DF} \leftarrow 0$
 - $\text{STOSB}; \quad [\text{ES:DI}] \leftarrow [\text{DS: SI}]$

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)

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

Phân loại và mô tả tập lệnh 8086/8088 (cont.)

Phân loại tập lệnh của vi xử lý



Các lệnh số học

ARITHMETIC

ADD = Add:

Reg./Memory with Register to Either

Immediate to Register/Memory

Immediate to Accumulator

ADC = Add with Carry:

Reg./Memory with Register to Either

Immediate to Register/Memory

Immediate to Accumulator

INC = Increment:

Register/Memory

Register

AAA = ASCII Adjust for Add

BAA = Decimal Adjust for Add

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

0 0 0 0 0 0 d w	mod reg r/m		
1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s: w = 0 1
0 0 0 0 0 1 0 w	data	data if w = 1	

0 0 0 1 0 0 d w	mod reg r/m		
1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s: w = 0 1
0 0 0 1 0 1 0 w	data	data if w = 1	

1 1 1 1 1 1 1 w	mod 0 0 0 r/m
0 1 0 0 0 reg	
0 0 1 1 0 1 1 1	
0 0 1 0 0 1 1 1	

Các lệnh số học

SUB = Subtract:

Reg./Memory and Register to Either

0 0 1 0 1 0 d w	mod reg r/m		
1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s w = 01
0 0 1 0 1 1 0 w	data	data if w = 1	

Immediate from Register/Memory

Immediate from Accumulator

SSB = Subtract with Borrow:

Reg./Memory and Register to Either

0 0 0 1 1 0 d w	mod reg r/m		
1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s w = 01
0 0 0 1 1 1 w	data	data if w = 1	

Immediate from Register/Memory

Immediate from Accumulator

DEC = Decrement:

Register/memory

1 1 1 1 1 1 1 w	mod 0 0 1 r/m	
0 1 0 0 1 reg		
1 1 1 1 0 1 1 w	mod 0 1 1 r/m	

Register

NEG = Change sign

Các lệnh số học

CMP = Compare:

Register/Memory and Register

Immediate with Register/Memory

Immediate with Accumulator

AAS = ASCII Adjust for Subtract

DAS = Decimal Adjust for Subtract

MUL = Multiply (Unsigned)

IMUL = Integer Multiply (Signed)

AAM = ASCII Adjust for Multiply

DIV = Divide (Unsigned)

IDIV = Integer Divide (Signed)

AAD = ASCII Adjust for Divide

CBW = Convert Byte to Word

CWD = Convert Word to Double Word

001110 d w	mod reg r/m				
100000 s w	mod 111 r/m	data	data if s w = 01		
0011110 w	data		data if w = 1		
00111111					
00101111					
11110111 w	mod 100 r/m				
11110111 w	mod 101 r/m				
11010100	00001010				
11110111 w	mod 110 r/m				
11110111 w	mod 111 r/m				
11010101	00001010				
10011000					
10011001					

Các lệnh số học

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 \leftarrow 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 \leftarrow AX + BX
 - ADD AL, 10; AL \leftarrow AL + 10
 - ADD [BX], AL; [DS:BX] \leftarrow [DS:BX] + AL

Các lệnh số học

Lệnh SUB - trừ các số nguyên:

- ▶ Dạng lệnh: SUB <Đích>, <Gốc>; Đích <- Đích - Gốc
- ▶ Ý nghĩa: Lấy Gốc trừ với Đích, 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

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; AL <- 30
 - MUL BL; AX <- AL * BL

Các lệnh số học: Các lệnh logic

LOGIC

NOT = Invert

SHL/SAL = Shift Logical/Arithmetic Left

SHR = Shift Logical Right

SAR = Shift Arithmetic Right

ROL = Rotate Left

ROR = Rotate Right

RCL = Rotate Through Carry Flag Left

RCR = Rotate Through Carry Right

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

1 1 1 1 0 1 1 w	mod 0 1 0 r/m
1 1 0 1 0 0 v w	mod 1 0 0 r/m
1 1 0 1 0 0 v w	mod 1 0 1 r/m
1 1 0 1 0 0 v w	mod 1 1 1 r/m
1 1 0 1 0 0 v w	mod 0 0 0 r/m
1 1 0 1 0 0 v w	mod 0 0 1 r/m
1 1 0 1 0 0 v w	mod 0 1 0 r/m
1 1 0 1 0 0 v w	mod 0 1 1 r/m

Các lệnh số học: Các lệnh logic

AND = And:

Reg./Memory and Register to Either

001000 d w	mod reg r/m		
1000000 w	mod 100 r/m	data	data if w = 1
0010010 w	data	data if w = 1	

Immediate to Register/Memory

Immediate to Accumulator

TEST = And Function to Flags, No Result:

Register/Memory and Register

1000010 w	mod reg r/m		
1111011 w	mod 000 r/m	data	data if w = 1
1010100 w	data	data if w = 1	

Immediate Data and Register/Memory

Immediate Data and Accumulator

OR = Or:

Reg./Memory and Register to Either

000010 d w	mod reg r/m		
1000000 w	mod 001 r/m	data	data if w = 1
0000110 w	data	data if w = 1	

Immediate to Register/Memory

Immediate to Accumulator

XOR = Exclusive or:

Reg./Memory and Register to Either

001100 d w	mod reg r/m		
1000000 w	mod 110 r/m	data	data if w = 1
0011010 w	data	data if w = 1	

Immediate to Register/Memory

Immediate to Accumulator

Các lệnh số học: Các lệnh logic

Lệnh NOT:

- ▶ Dạng lệnh: NOT <Đích>
- ▶ Ý nghĩa: Đảo các bít của toán hạng Đích
- ▶ Lệnh NOT ảnh hưởng đến các cờ: Z, S, P
- ▶ Ví dụ:
 - MOV AL, 80H; $80H = 1000\ 0000B$
 - NOT AL; $7FH = 0111\ 1111B$

Các lệnh số học: Các lệnh logic

Lệnh AND:

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

Các lệnh số học: Các lệnh logic

Lệnh OR:

- ▶ Dạng lệnh: 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
- ▶ Ví dụ: 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

Các lệnh số học: Các lệnh logic

Lệnh XOR:

- ▶ Dạng lệnh: 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
- ▶ Ví dụ: Dùng XOR để xoá nội dung của thanh ghi/ô nhớ
 - Xoá thanh ghi AL (0-7):
XOR AL, AL; AL <- 0
 - Xoá thanh ghi BX:
XOR BX, BX ; BX <- 0

Các lệnh số học: Các lệnh dịch và lệnh 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.

Các lệnh dịch và lệnh quay

Lệnh dịch trái: SHL



- ▶ Dạng lệnh SHL <Đích>, 1
SHL <Đích>, CL
- ▶ Ý nghĩa: Dịch trái một bit hoặc số bit trong CL nếu dịch hơn 1 bit
 - MSB (Most Significant Bit) chuyển sang cờ nhớ CF
 - 0 được diền vào LSB (Least Significant Bit)
 - Các bít giữa MSB và LSB được dịch sang trái 1 bit
- ▶ Ví dụ:
 - MOV AL, 08H; 0000 1000B
 - SHL AL, 1; 0001 0000B
 - MOV CL, 2;
 - SHL AL, CL; 0100 0000B

Các lệnh dịch và lệnh quay

Lệnh dịch phải: SHR



- ▶ Dạng lệnh SHR <Đích>, 1
SHR <Đích>, CL
- ▶ Ý nghĩa: Dịch phải một bit hoặc số bit trong CL nếu dịch hơn 1 bit
 - 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
- ▶ Ví dụ:
 - MOV AL, 08H; 1000 0000B (128)
 - SHR AL, 1; 0100 0000B (64)
 - MOV CL, 2;
 - SHR AL, CL; 0001 0000B (16)

Các lệnh dịch và lệnh quay

Lệnh quay trái: ROL



- ▶ Dạng lệnh ROL <Đích>, 1
ROL <Đích>, CL
- ▶ Ý nghĩa: Quay trái một bit hoặc số bit trong CL nếu quay hơn 1 bit
 - 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
- ▶ Ví dụ:
 - MOV AL, 88H; 1000 1000B
 - ROL AL, 1; 0001 0001B
 - MOV CL, 2;
 - ROL AL, CL; 0100 0100B

Các lệnh dịch và lệnh quay

Lệnh quay phải: ROR



- ▶ Dạng lệnh ROR <Đích>, 1
ROR <Đích>, CL
- ▶ Ý nghĩa: Quay phải một bit hoặc số bit trong CL nếu quay hơn 1 bit
 - 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
- ▶ Ví dụ:
 - MOV AL, 88H; 1000 1000B
 - ROR AL, 1; 0100 0100B
 - MOV CL, 2;
 - ROR AL, CL; 0001 0001B

Các lệnh chuỗi

STRING MANIPULATION

REP = Repeat

1111001z

MOVS = Move Byte/Word

1010010w

CMPS = Compare Byte/Word

1010011w

SCAS = Scan Byte/Word

1010111w

LODS = Load Byte/Wd to AL/AX

1010110w

STOS = Stor Byte/Wd from AL/A

1010101w

Các lệnh nhảy và điều khiển

JMP = Unconditional Jump:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Direct within Segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct within Segment-Short	1 1 1 0 1 0 1 1	disp	
Indirect within Segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct Intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	
RET = Return from CALL:			
Within Segment	1 1 0 0 0 0 1 1		
Within Seg Adding Immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment Adding Immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high

Các lệnh nhảy và điều khiển

JE/JZ = Jump on Equal/Zero	0 1 1 1 0 1 0 0	disp
JL/JNGE = Jump on Less/Not Greater or Equal	0 1 1 1 1 1 0 0	disp
JLE/JNG = Jump on Less or Equal/Not Greater	0 1 1 1 1 1 1 0	disp
JB/JNAE = Jump on Below/Not Above or Equal	0 1 1 1 0 0 1 0	disp
JBE/JNA = Jump on Below or Equal/Not Above	0 1 1 1 0 1 1 0	disp
JP/JPE = Jump on Parity/Parity Even	0 1 1 1 1 0 1 0	disp
JO = Jump on Overflow	0 1 1 1 0 0 0 0	disp
JS = Jump on Sign	0 1 1 1 1 0 0 0	disp
JNE/JNZ = Jump on Not Equal/Not Zero	0 1 1 1 0 1 0 1	disp
JNL/JGE = Jump on Not Less/Greater or Equal	0 1 1 1 1 1 0 1	disp
JNLE/JG = Jump on Not Less or Equal/Greater	0 1 1 1 1 1 1 1	disp
JNB/JAE = Jump on Not Below/Above or Equal	0 1 1 1 0 0 1 1	disp
JNBE/JA = Jump on Not Below or Equal/Above	0 1 1 1 0 1 1 1	disp
JNP/JPO = Jump on Not Par/Par Odd	0 1 1 1 1 0 1 1	disp

Các lệnh nhảy và điều khiển

JNO = Jump on Not Overflow

0 1 1 1 0 0 0 1	disp
-----------------	------

JNS = Jump on Not Sign

0 1 1 1 1 0 0 1	disp
-----------------	------

LOOP = Loop CX Times

1 1 1 0 0 0 1 0	disp
-----------------	------

LOOPZ/LOOPE = Loop While Zero/Equal

1 1 1 0 0 0 0 1	disp
-----------------	------

LOOPNZ/LOOPNE = Loop While Not Zero/Equal

1 1 1 0 0 0 0 0	disp
-----------------	------

JCXZ = Jump on CX Zero

1 1 1 0 0 0 1 1	disp
-----------------	------

INT = Interrupt

Type Specified

1 1 0 0 1 1 0 1	type
-----------------	------

Type 3

1 1 0 0 1 1 0 0

INTO = Interrupt on Overflow

1 1 0 0 1 1 1 0

IRET = Interrupt Return

1 1 0 0 1 1 1 1

Các lệnh nhảy và điều khiển

7 6 5 4 3 2 1 0

7 6 5 4 3 2 1 0

PROCESSOR CONTROL

CLC = Clear Carry

1 1 1 1 1 0 0 0

CMC = Complement Carry

1 1 1 1 0 1 0 1

STC = Set Carry

1 1 1 1 1 0 0 1

CLD = Clear Direction

1 1 1 1 1 1 0 0

STD = Set Direction

1 1 1 1 1 1 0 1

CLI = Clear Interrupt

1 1 1 1 1 0 1 0

STI = Set Interrupt

1 1 1 1 1 0 1 1

HLT = Halt

1 1 1 1 0 1 0 0

WAIT = Wait

1 0 0 1 1 0 1 1

ESC = Escape (to External Device)

1 1 0 1 1 x x x

mod x x x r/m

LOCK = Bus Lock Prefix

1 1 1 1 0 0 0 0

Các lệnh nhảy và điều khiển

CONTROL TRANSFER

CALL = Call:

Direct within Segment

11101000	disp-low	disp-high
----------	----------	-----------

Indirect within Segment

11111111	mod 0 10 r/m
----------	--------------

Direct Intersegment

10011010	offset-low	offset-high
----------	------------	-------------

seg-low seg-high

Indirect Intersegment

11111111	mod 0 11 r/m
----------	--------------

Các lệnh nhảy và 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 trả về từ chương trình con RET

Các lệnh nhảy và 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).
- ▶ Ví dụ:

START:

ADD AX, BX
SUB BX, 1

.....

JMP START; chuyển đến thực hiện lệnh sau nhãn START

Các lệnh nhảy và 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).

Các lệnh nhảy và 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:

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

Các lệnh nhảy và 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 à dừng

; nếu CX khác 0: CX <- CX-1 và quay lại

; bắt đầu vòng lặp mới từ vị trí của START

Các lệnh nhảy và điều khiển

Lệnh CALL và RET

► Dạng lệnh:

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

► VD:

CALL GIAITHUA; gọi chương trình con GIAITHUA

.....
; phần mã của chương trình con
GIAITHUA PROC; bắt đầu mã CT con

.....
RET; trả về chương trình gọi
GIAITHUA ENDP; ; kết thúc mã CT con

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

Một số các lệnh khác : NOP và Halt

► 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

► 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

Một số các lệnh khác

► Lệnh tăng INC:

- Dạng: INC <Đích>; Đích \leftarrow Đích + 1

► Lệnh giảm DEC:

- Dạng: DEC <Đích>; Đích \leftarrow Đích - 1

► Lệnh so sánh CMP

- Dạng: CMP <Đích>, <Đối số>

- 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

Một số các lệnh khác

► Lệnh PUSH – đẩy dữ liệu vào ngăn xếp

- Dạng: PUSH <Đốc>
- Ý nghĩa: Nạp Gốc vào đỉnh ngăn xếp; Gốc phải là toán hạng 2 bytes.
- Diễn giải
 - ▶ $SP \leftarrow SP + 2$; tăng con trỏ ngăn xếp SP
 - ▶ $SP \leftarrow Gốc$; nạp dữ liệu vào ngăn xếp
- Ví dụ: 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:

Phân loại và mô tả tập lệnh 8086/8088 (cont.)

Phân loại tập lệnh của vi xử lý



- ▶ Dịch $<- SP$; lấy dữ liệu ra khỏi ngăn xếp
- ▶ $SP <- SP - 2$; giảm con trỏ ngăn xếp SP

- Ví dụ: POP BX

► Lệnh NEG – đảo dấu giá trị của toán hạng

- Dạng: NEG <Đích>
- Ý nghĩa: Đảo dấu giá trị lưu trong Đích
- Ví dụ:

- ▶ 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>

Phân loại và mô tả tập lệnh 8086/8088 (cont.)

Phân loại tập lệnh của vi xử lý



- Ví dụ:

- ▶ MOV BX, 100

- ▶ MOV AX, 200

- ▶ XCHG AX, BX; AX <- 100, BX <- 200

- ▶ 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

- Ví dụ:

- 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

Chương 4 - Intel 8086/8088

- ▶ Kiến trúc bên trong 8086/8088
- ▶ Tập lệnh 8086/8088

Tiếp theo Chương 4 - Lập trình hợp ngữ với bộ vi xử lý 8086/8088

- ▶ Giới thiệu về hợp ngữ
- ▶ Cú pháp của chương trình hợp ngữ
- ▶ Dữ liệu cho chương trình hợp ngữ
- ▶ Biến và hằng
- ▶ Khung chương trình hợp ngữ
- ▶ Các cấu trúc điều khiển
- ▶ Giới thiệu phần mềm mô phỏng emu8086