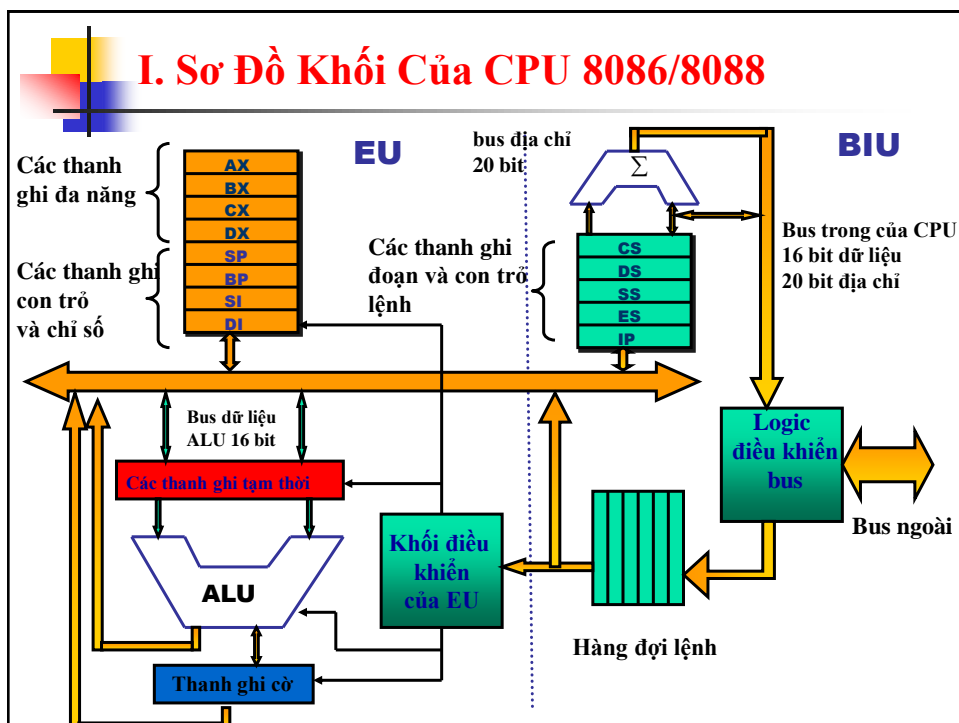


Chương 3

KIẾN TRÚC PHẦN MỀM





Các thành phần của CPU

- Là vi xử lý 16 bit
- Gồm 2 khối chính : BIU và EU
- BIU (Bus Interface Unit : Đơn vị giao tiếp bus)
 - Chức năng : lấy lệnh, đọc toán hạng, lưu kết quả, truy xuất bộ nhớ cache
 - Gồm :
 - Các thanh ghi phân đoạn : CS, DS, ES, SS
 - Bộ tính tổng → để xác định địa chỉ vật lý 20 bit
 - Hàng đợi lệnh (8088 : 4 byte, 8086: 6 byte) để tăng hiệu suất của CPU
 - Thanh ghi con trỏ lệnh IP : trỏ đến địa chỉ lệnh kế tiếp



Đơn vị thực thi

- EU (Execution unit=đơn vị thực thi)
 - Chức năng : thi hành lệnh, định thì, kiểm tra các tín hiệu trạng thái, tín hiệu yêu cầu ngắt, DMA, Ready
 - Gồm :
 - ALU (Arithmetic Logic Unit)
 - Thanh ghi cờ
 - Các thanh ghi đa dụng : AX, BX, CX, DX
 - Các thanh ghi chỉ số : SI, DI
 - Các thanh ghi nền : BP, SP
 - Các thanh ghi tạm thời nạp toán hạng tính toán cho ALU
 - Đơn vị điều khiển (CU) có chức năng : kiểm soát quá trình lấy lệnh, giải mã lệnh, thi hành lệnh

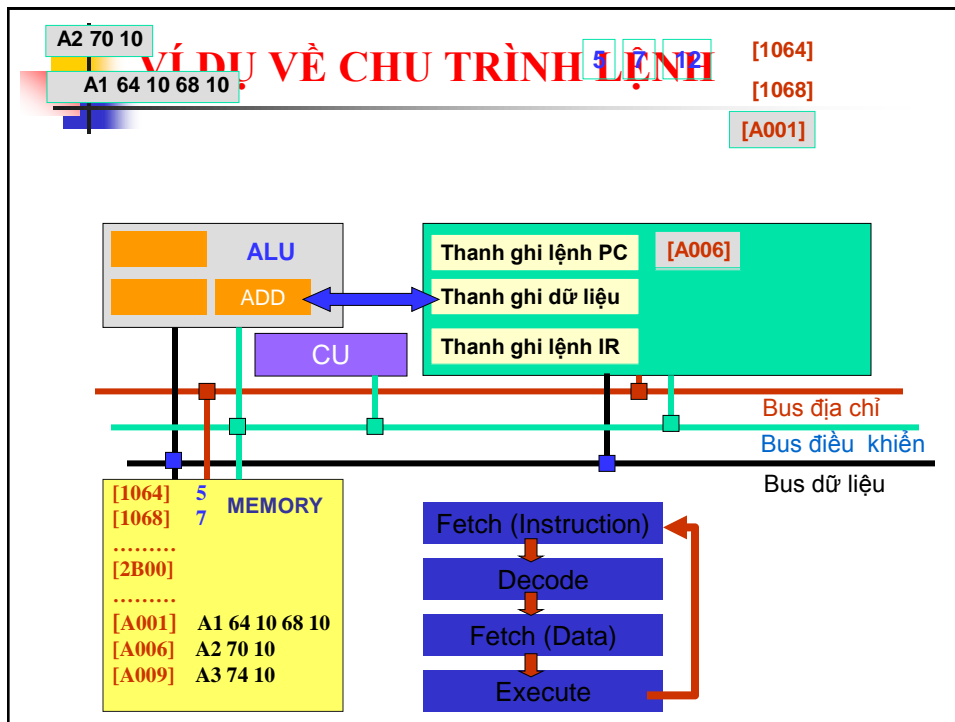


- Hoạt động của hai khối BIU và EU diễn ra độc lập với nhau nên quá trình lấy lệnh và thi hành lệnh được vi xử lý thực hiện đồng thời theo cơ cấu đường ống (pipeline).
- Điều này tuy không làm tăng tốc độ xử lý của CPU (giới hạn bởi tần số xung đồng bộ) nhưng làm giảm bớt thời gian thi hành của cả chương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



II. Quy Trình Thi Hành Lệnh

- Các bước thi hành lệnh
 - Nạp lệnh từ bộ nhớ vào thanh ghi lệnh IR
 - Tăng thanh ghi IP (PC) để chỉ tới lệnh tiếp theo
 - Giải mã lệnh
 - Định vị toán hạng được dùng bởi lệnh
 - Nạp toán hạng từ bộ nhớ (nếu cần)
 - Thi hành lệnh
 - Lưu kết quả vào nơi thích hợp
 - Trở lại bước 1 để nạp lệnh kế.



Vấn Đề Thi Hành Lệnh Song Song

- Mong muốn : MT xử lý càng nhanh càng tốt
- Giải pháp : tăng tốc độ phần cứng → gặp 1 số khó khăn
 - Tốc độ truyền tín hiệu \leq vận tốc ánh sáng = const = c = 300000 km/s
 - Muốn cho máy tính có thời gian lệnh = 1 ns → khoảng cách giữa các bộ phận phải đủ nhỏ (≤ 20 cm)
 - Nghĩa là muốn tốc độ càng nhanh thì khoảng cách càng nhỏ → sinh ra nhiều ...
- Tìm giải pháp khác :

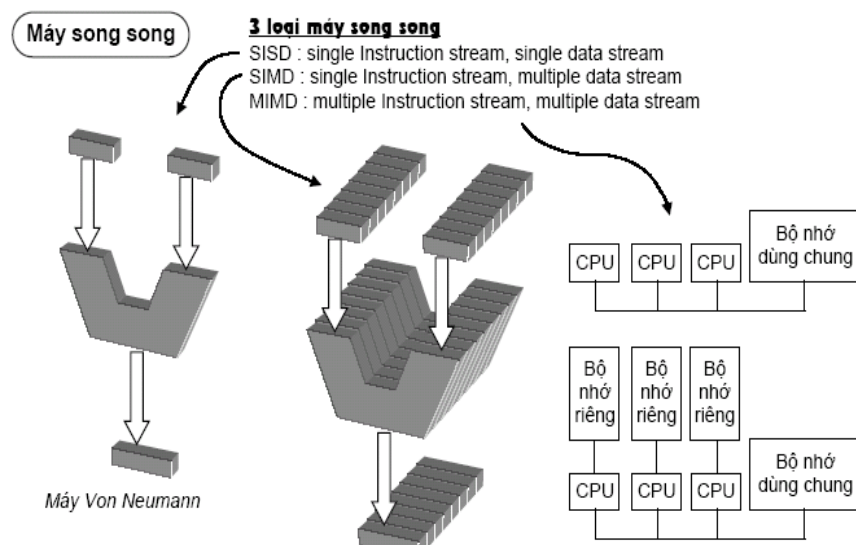


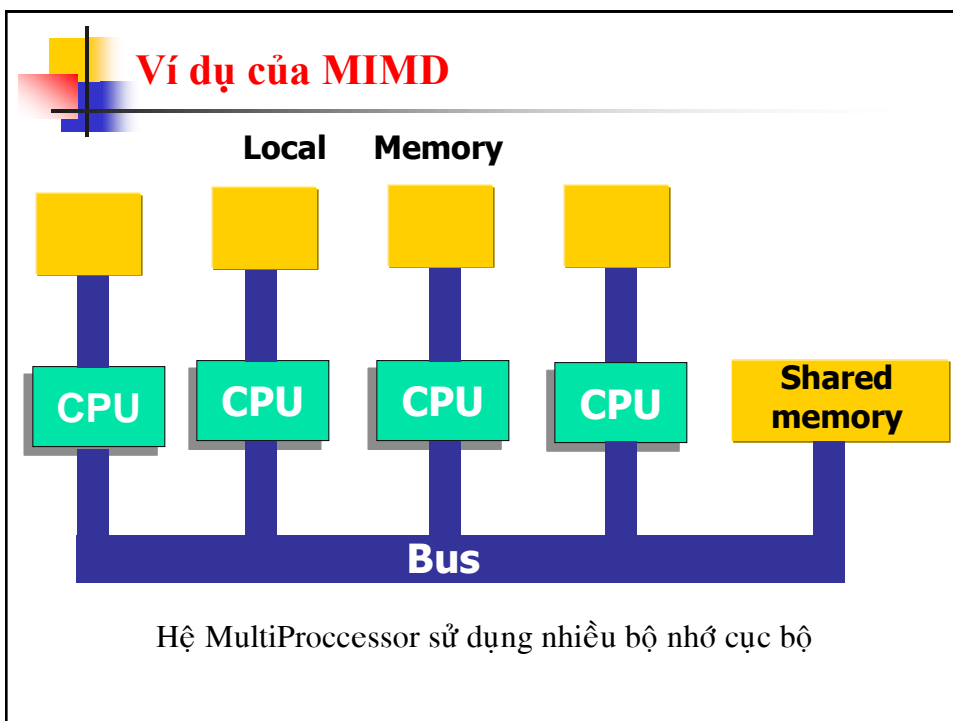
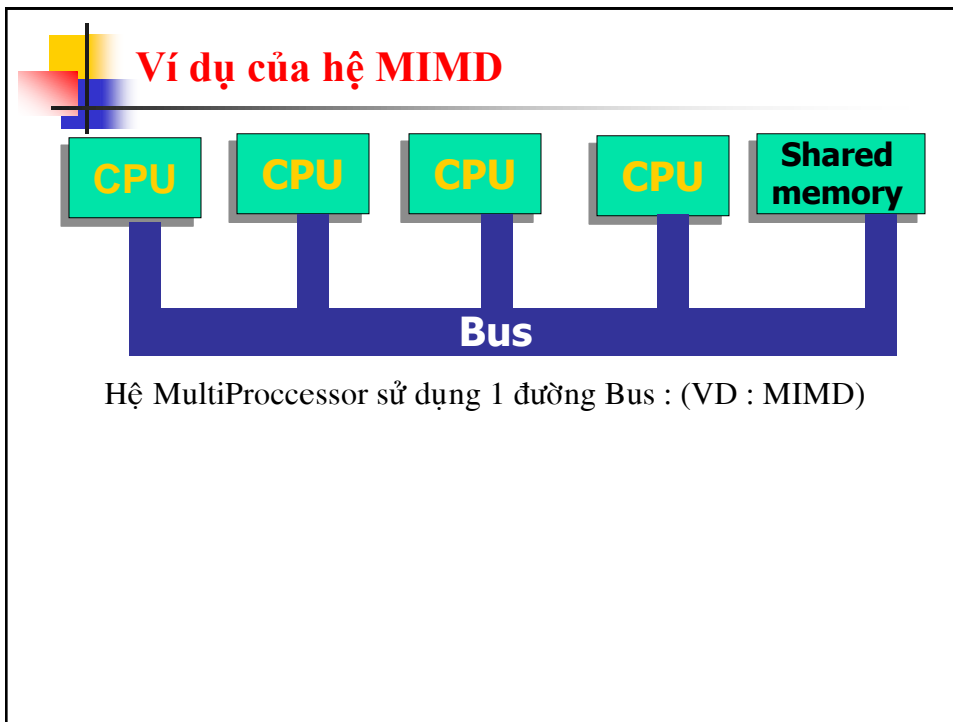
Giải pháp tăng tốc CPU

- (tìm giải pháp khác) : chế tạo nhiều ALU hoặc nhiều CPU → tăng tốc → máy tính song song
- 3 loại máy tính song song
 - SISD (single instruction stream, single data stream) : MT Von Neuman bình thường
 - SIMD (single instruction stream, multiple data stream) : VD dự báo thời tiết, 1 chương trình xử lý nhiều nguồn dữ liệu khác nhau
 - MIMD (multiple instruction stream, multiple data stream) : VD nhiều chương trình khác nhau, nhiều tập dữ liệu khác nhau



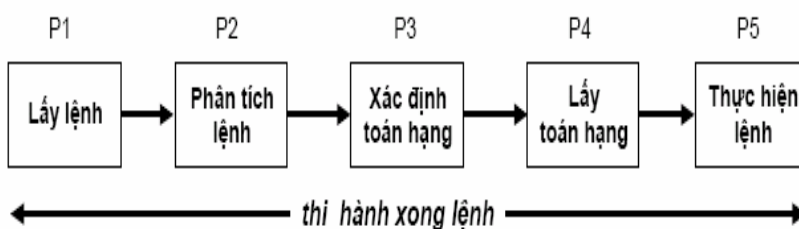
Máy tính song song



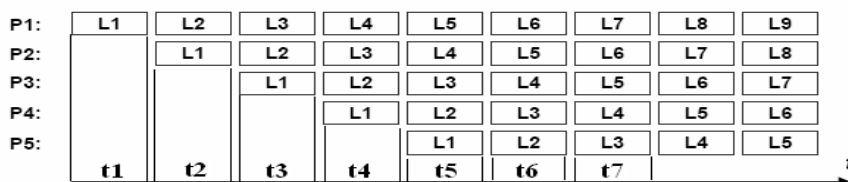


Giải pháp khác : Pipeling (cơ chế đường ống)

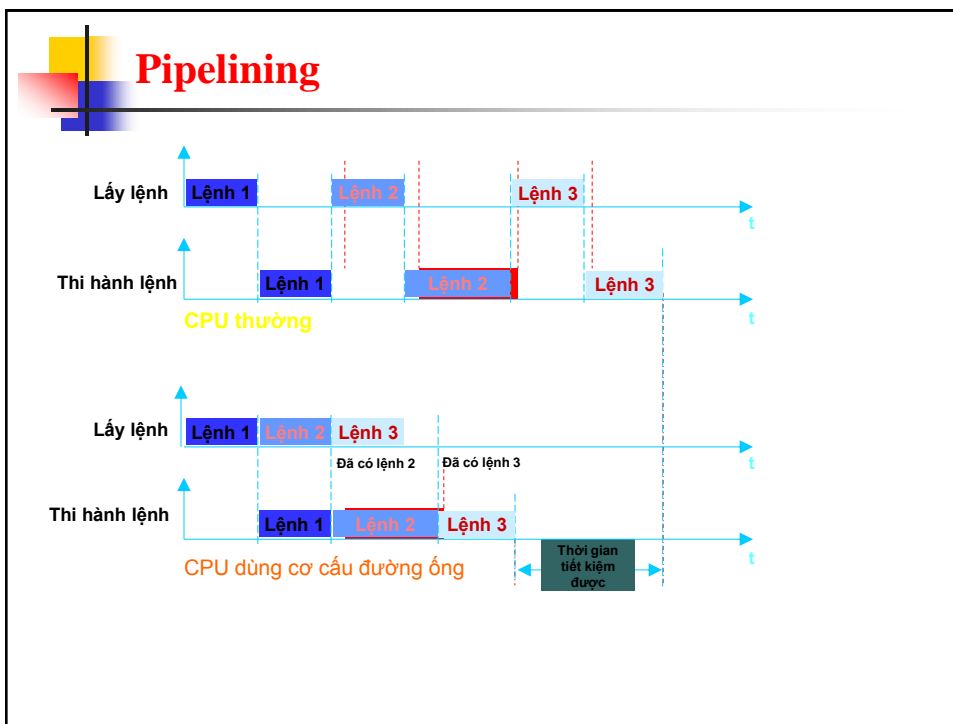
- Người ta chia CPU thành nhiều khối chức năng, mỗi khối thực hiện hiện các chức năng riêng, độc lập.
- Ví dụ mô hình CPU có 5 đơn vị xử lý song song P1 → P5



Phân tích cơ chế đường ống




- Trong t1 : lệnh L1 được nạp từ bộ nhớ bởi P1
- Trong t2 : L1 được chuyển sang P2 để giải mã
P1 nạp lệnh L2
- Trong t3 : P3 xác định toán hạng cho L1
P2 giải mã lệnh L2
P1 nạp lệnh L3
- Quá trình cứ tiếp tục



III. Tổ Chức Các Thanh Ghi

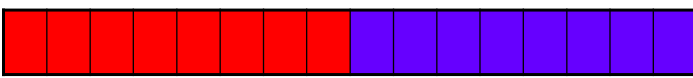
- Là các phần tử có khả năng lưu trữ thông tin với dung lượng 8, 16, 32, 64 bit.
- Được xây dựng từ các FlipFlop nên có tốc độ truy xuất rất nhanh.
- Các nhóm
 - Thanh ghi đa dụng
 - Thanh ghi chỉ số và con trỏ
 - Thanh ghi phân đoạn
 - Các thanh ghi đặc biệt khác




1. Thanh Ghi Đa Dụng (dữ liệu)

- Thanh ghi AX (Accumulator register)

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

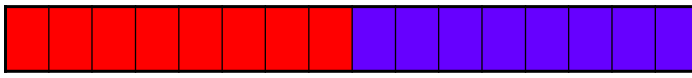


← AH
← AL
- Chức năng :
 - Lưu trữ dữ liệu, dùng trong phép toán số học, logic, dùng trong các lệnh ngắt, lệnh xuất nhập.
 - Có thể dùng như 3 thanh ghi độc lập AX, AH, AL
 - 386 AX → EAX, dài 32 bit, dùng như 4 thanh ghi



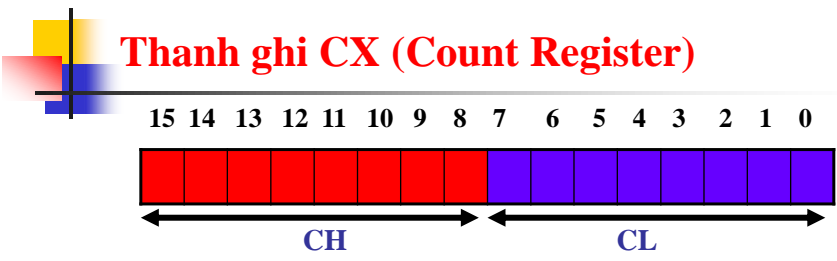
Thanh ghi BX (Base Register)

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0



← BH
← BL

- Thanh ghi nền
- Chức năng :
 - Dùng trong các phép toán số học, logic
 - Dùng như con trỏ để định địa chỉ gián tiếp
 - Dùng như 3 thanh ghi độc lập BX, BH, BL
 - 386 BX → EBX, 32 bit, dùng như 4 thanh ghi độc lập : EBX, BX, BH, BL

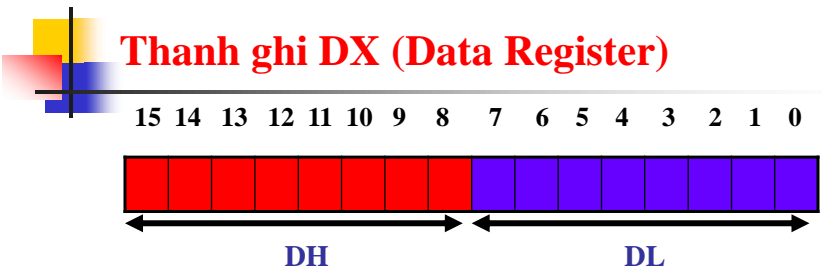


Thanh ghi CX (Count Register)

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

CH CL

- Thanh ghi đếm
- Chức năng :
 - Dùng trong các phép toán số học, logic
 - Dùng làm biến đếm trong các vòng lặp
 - CL chứa số lần quay, dịch trong các lệnh dịch, quay
 - Có thể dùng như 3 thanh ghi CX, CH, CL
 - 386 CX → ECX dài 32 bit, dùng như 4 thanh ghi



Thanh ghi DX (Data Register)

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

DH DL

- Thanh ghi dữ liệu
- Chức năng :
 - Dùng trong các phép toán số học, logic
 - Chứa dữ liệu tạm thời trong các phép nhân, chia
 - Dùng chứa địa chỉ port trong lệnh xuất nhập
 - 386 DX → EDX 32 bit, dùng như 4 thanh ghi

2. Thanh Ghi Chỉ Số và Con Trỏ

- 16 bit (8086) : SP, BP, SI, DI
- 32 bit (386) : ESP, EBP, ESI, EDI

Stack Pointer	SP
ESP	
Base Pointer	BP
EBP	
Dest Index	DI
EDI	
Source Index	SI
ESI	

- SI (source index = chỉ số nguồn) kết hợp DS (DS:SI) chứa địa chỉ dữ liệu nguồn
- DI (destination index = chỉ số đích) kết hợp với ES (ES:DI) chứa địa chỉ dữ liệu đích
- SI và DI có thể dùng như các thanh ghi đa dụng

Thanh ghi con trỏ

- BP (Base pointer = con trỏ nền) :
 - vai trò như BX
 - Truy xuất các thành phần khác nhau trong stack mà không cần tuân thủ nguyên tắc LIFO (SS:BP)
- SP (stack pointer = con trỏ stack) :
 - Chứa địa chỉ hiện thời của đỉnh stack (SS:SP)

3. Thanh Ghi Phân Đoạn (segment register)

- Kích thước 16 bit
- CS (Code segment) : chứa địa chỉ của phân đoạn mã lệnh của chương trình trong bộ nhớ.
- DS (Data segment) : trỏ đến đoạn dữ liệu
- SS (Stack segment) : trỏ đến đoạn ngăn xếp
- ES (Extra segment) : trỏ đến đoạn thêm (phụ), thường dùng để bổ xung cho đoạn dữ liệu → có kích thước > 64KB

Bộ thanh ghi đoạn

CS	Code segment
DS	Data segment
SS	Stack segment
ES	Extra segment
IP	Instruction pointer

4. Các Thanh Ghi Đặc Biệt

- Thanh ghi IP (instruction pointer=con trỏ lệnh) lưu địa chỉ lệnh kế
- Thanh ghi cờ IF (instruction flag) :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	x	x	OF	DF	IF	TF	SF	ZF	x	AF	x	PF	x	CF

CF (Carry flag) CF=1 khi có nhớ hoặc mượn từ MSB

Cờ ZF (Zero flag) ZF=1 khi kết quả thao tác =0, hoặc bằng nhau

Cờ dấu SF (Sign flag) SF=1 nếu kết quả thao tác âm

Cờ tràn OF (Overflow flag) OF=1 khi kết quả là 1 số vượt ra ngoài giới hạn biểu diễn của nó trong khi thực hiện phép toán cộng – trừ số có dấu



Thanh ghi cờ

Cờ kiểm tra PF (Parity flag) PF=1 báo tổng số bit 1 trong kết quả chẵn

Cờ trung gian AF (Auxiliary flag) AF=1 cho biết có nhớ hoặc mượn từ số BCD thấp sang số BCD cao

Cờ IF (Interrupt flag) IF=1 cho phép ngắt
IF=0 che ngắt


Cờ hướng DF (Direction flag) cho biết hướng xử lý chuỗi
DF=1 : hướng xử lý từ phải sang trái (theo chiều giảm địa chỉ)
DF=0 : hướng xử lý từ trái sang phải (theo chiều tăng địa chỉ)

Cờ bẫy TF (trap flag) : TF=1 CPU vào chế độ chạy từng lệnh




Tràn số (overflow)

- Chỉ có thể xảy ra khi cộng các số cùng dấu (hoặc trừ số khác dấu)
- Được phát hiện khi nhớ vào MSB không bằng nhớ ra khỏi MSB
- Khi tràn xảy ra OF=1



Ví dụ về tràn số có dấu

$\begin{array}{r} 10010110 \\ + 10100011 \\ \hline 00111001 \end{array}$	$\begin{array}{r} 00110110 \\ + 01100011 \\ \hline 10011001 \end{array}$
<ul style="list-style-type: none"> ■ <i>Carry in = 0, Carry out = 1</i> ■ <i>Neg+Neg=Pos</i> ■ <i>Signed overflow occurred</i> ■ <i>OF = 1 (set)</i> 	<ul style="list-style-type: none"> ■ <i>Carry in = 1, Carry out = 0</i> ■ <i>Pos+Pos=Neg</i> ■ <i>Signed overflow occurred</i> ■ <i>OF = 1 (set)</i>



Ví dụ về không tràn số có dấu

$\begin{array}{r} 10010110 \\ + 01100011 \\ \hline 11111001 \end{array}$	$\begin{array}{r} 10010110 \\ + 11110011 \\ \hline 10001001 \end{array}$
<ul style="list-style-type: none"> ■ <i>Carry in = 0, Carry out = 0</i> ■ <i>Neg+Pos=Neg</i> ■ <i>No Signed overflow occurred</i> ■ <i>OF = 0 (clear)</i> 	<ul style="list-style-type: none"> ■ <i>Carry in = 1, Carry out = 1</i> ■ <i>Neg+Neg=Neg</i> ■ <i>No Signed overflow occurred</i> ■ <i>OF = 0 (clear)</i>



Tràn số không dấu

- Cờ nhớ được dùng để chỉ ra nếu một thao tác trên số không dấu tràn

10010110	
+ 11110011	
10001001	
- Các bộ xử lý chỉ cộng hoặc trừ - nó không quan tâm dữ liệu có dấu hay không dấu.
 - *Carry out = 1*
 - *Unsigned overflow occurred*
 - *CF = 1 (set)*

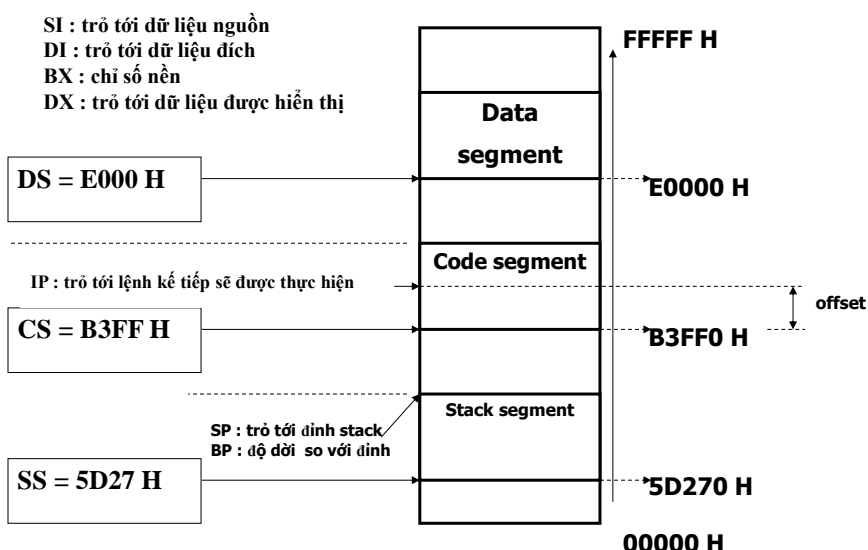


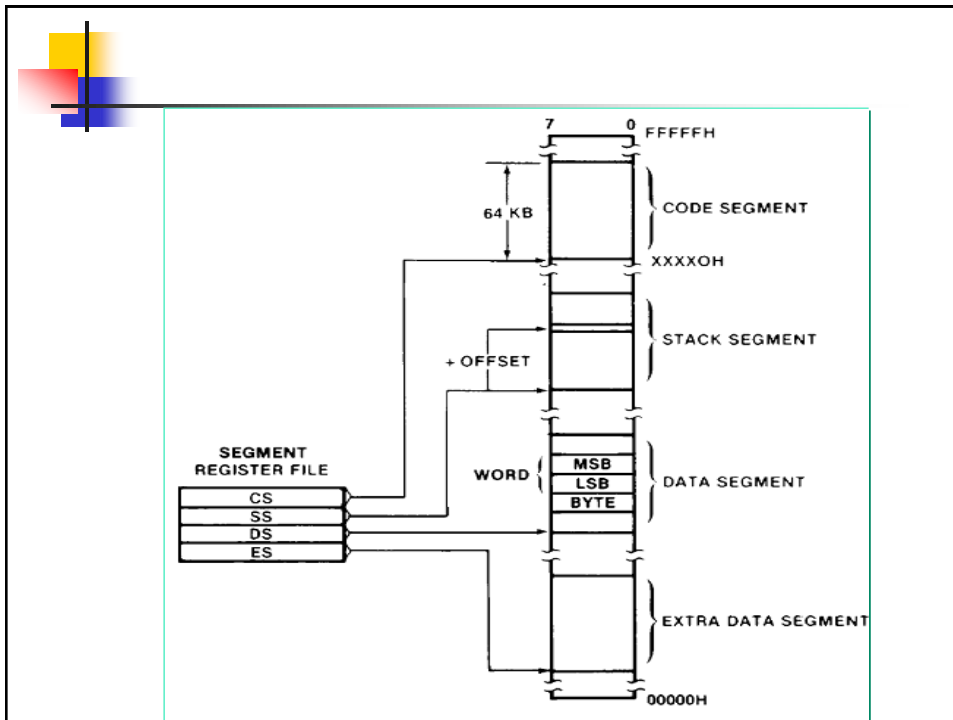
IV. Tổ Chức Bộ Nhớ

- Cấu trúc tổng quát của 1 chương trình : gồm 3 phần
 - Code (phần mã, chứa các lệnh) : chứa mã chương trình, phần này chứa trong 1 phần bộ nhớ gọi là code segment
 - Data : chứa dữ liệu có sẵn trong chương trình và dữ liệu có thể phát sinh, phần này được chứa trong một phần bộ nhớ gọi là data segment
 - Stack : chứa các địa chỉ quay về, cất và khôi phục các thanh ghi, phần này được chứa trong 1 phần bộ nhớ gọi là stack segment.

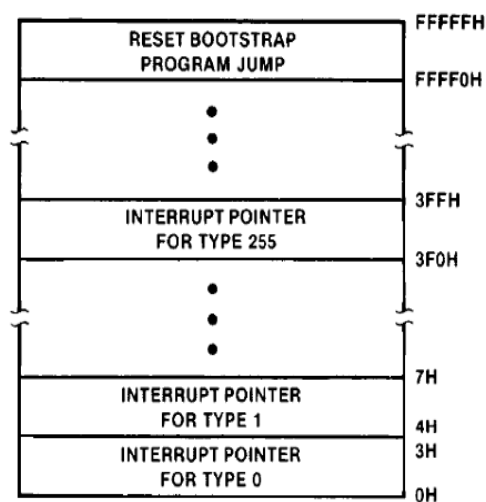
- Mỗi segment chiếm chỗ trong 1 phần bộ nhớ và độc lập nhau.
- Vậy người ta cần :
 - Có địa chỉ để xác định đầu segment (segment được bắt đầu ở chỗ nào trong bộ nhớ), gọi là đ/c segment.
 - Trong mỗi segment, mỗi phần tử (dữ liệu hoặc mã lệnh) có 1 địa chỉ cho biết nó nằm ở chỗ nào trong segment đó, gọi là địa chỉ offset.
- Ví dụ :

Ví dụ về tổ chức bộ nhớ





- ▶ 00000-003FFF:
Bảng véctor ngắt
- ▶ FFFF0-FFFFF:
Đoạn mã khởi động máy





2. Tổ Chức Bộ Nhớ Của 8088/8086

- Chế độ real – mode
- Có 20 đường địa chỉ (bus địa chỉ 20 đường) → không gian bộ nhớ là $2^{20} = 1 \text{ MB}$, đánh địa chỉ từ 00000H – FFFFFH (mỗi ô nhớ lưu 1 Byte).
- Địa chỉ vật lý : là con số 20 bit, xác định ô nhớ trong không gian bộ nhớ vật lý 1MB.
- Chia không gian địa chỉ vật lý thành nhiều khối nhỏ hơn gọi là segment vật lý, kích thước 64KB
- Mỗi segment có 1 địa chỉ (cho biết đầu segment, segment bắt đầu ở vị trí nào trong không gian 1MB). Địa chỉ này phải :
 - Chia hết cho 16
 - Được chứa trong 1 thanh ghi đoạn (DS,SS,CS,ES)



Tổ chức bộ nhớ

- Mỗi phần tử trong 1 segment cũng có 1 địa chỉ so với đầu segment, có giá trị từ 0000H – FFFFH gọi là địa chỉ offset.
- Mỗi segment có thể tách rời, có thể chồng lấp nhau 1 phần, nhưng phải cách nhau 1 khoảng tối thiểu 16 byte (gọi là 1 paragraph)
- Địa chỉ luận lý (logic) dùng cách viết
segment : offset
- Địa chỉ vật lý = segment * 16 + offset

Tóm tắt : Quản lý bộ nhớ

Địa chỉ vật lý : dùng để thiết kế mạch
số 20 bit $A_{19}A_{18}A_{17} \dots A_1A_0$ (00000 ÷ FFFFF)

Địa chỉ luận lý : dùng cho lập trình
2 thành phần

segment : offset

16 bit 16 bit

Bộ nhớ

Ô nhớ có địa chỉ luận lý
segment : offset

Thanh ghi đoạn chứa SEGMENT

Thanh ghi đa dụng chứa OFFSET

Địa chỉ luận lý

$\text{Địa chỉ vật lý} = (\text{SEGMENT} * 16) + \text{OFFSET}$

+

SEGMENT 0 (dịch trái 4 bit)

OFFSET (offset giữ nguyên)

Địa chỉ vật lý (Địa chỉ vật lý 20 bit)

PARAGRAPH

BYTE

segment 0000 →

segment 0001 →

segment 0002 →

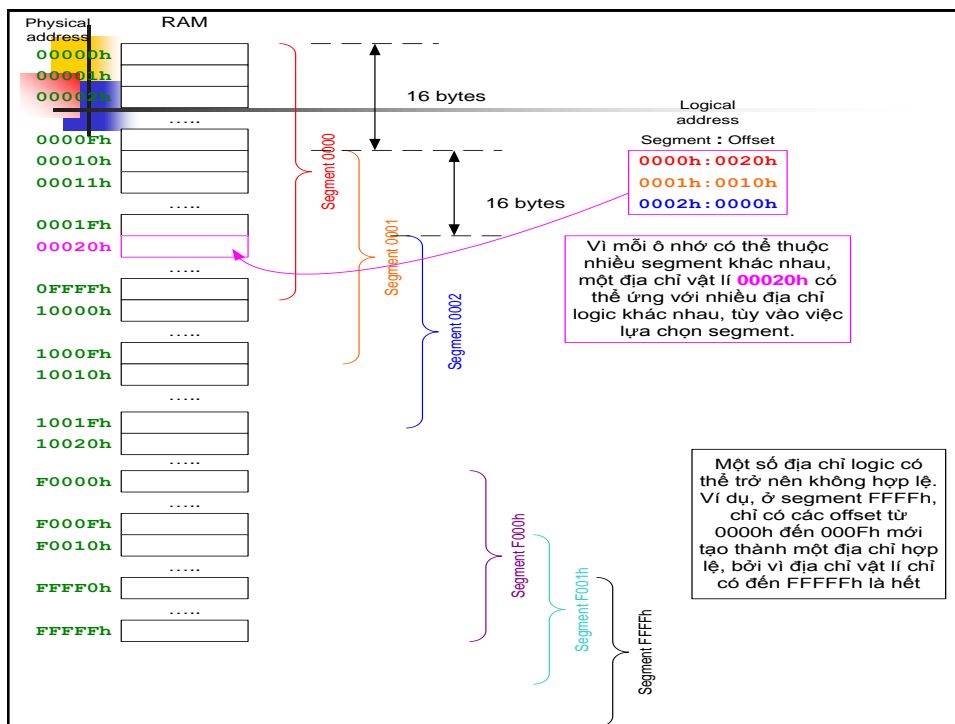
Phần chồng chập của 3 segment 0000, 0001, 0002.

Địa chỉ vật lý

⇒ 1 paragraph = 16 byte


⇒ Khi segment thay đổi 1 đơn vị (paragraph) thì địa chỉ vật lý thay đổi 16 byte.

⇒ Một địa chỉ vật lý có thể được biểu diễn bằng nhiều địa chỉ luận lý khác nhau.




Ví dụ :

- **Ví dụ 1 :** Thanh ghi CS chứa trị B3FF H, địa chỉ vật lý của segment mã là ?
- **Ví dụ 2 :** Tính địa chỉ bắt đầu và kết thúc cho đoạn dữ liệu, giả sử thanh ghi DS=E000 H
- **Ví dụ 3 :** Tính địa chỉ vật lý tương ứng với địa chỉ offset D470H trong đoạn phụ ES=52B9H
- **Ví dụ 4 :** Tính địa chỉ vật lý cho ô nhớ có địa chỉ offset = 2D90H trong đoạn stack SS = 5D27H.



Nhận xét

- 1 cặp segment : offset → cho 1 địa chỉ vật lý duy nhất
- 1 địa chỉ vật lý có thể được suy ra từ nhiều cặp segment : offset khác nhau
- Địa chỉ offset thường được chứa trong 1 thanh ghi



Kết hợp mặc định thanh ghi segment + offset

Segment	Offset	Chú thích
CS	IP, EIP	Địa chỉ lệnh kế
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



V. Các Chế Độ Định Địa Chỉ

- Cơ bản về cấu trúc lệnh : 1 lệnh thường chứa
 - Opcode : xác định hoạt động (thao tác) được thực hiện
 - Operands : có thể có hoặc không có.
- Chế độ định địa chỉ
 - Operand được yêu cầu trong 1 số phép toán ở đâu ?
 - Chế độ định địa chỉ : cho biết làm thế nào để tính (xác định) vị trí chính xác của data (operand) muốn xử lý.
 - CPU càng mạnh, càng hỗ trợ nhiều chế độ.



- Có các chế độ định địa chỉ
 - Chế độ hiệu ngàm
 - Chế độ định vị tức thời
 - Chế độ định vị thanh ghi
 - Chế độ định vị bộ nhớ
 - Định vị gián tiếp thanh ghi
 - Định vị chỉ số nền
 - ...

a. Chế độ thanh ghi

- Toán hạng nằm trong thanh ghi
- Ví dụ : Mov AX, **BX**

b. Định vị tức thời

- Toán hạng nằm trực tiếp trong lệnh
- Ví dụ :
Mov AX, 1B67H

AX	0000	Địa chỉ	
BX	1B69	1B66	35
CX	1643	1B67	69
DX	29A2	1B68	1B
		1B69	24
SP	FF03	1B6A	01
BP	1B03	1B6B	
SI	0001		
DI	0004		


c. Định Vị Bộ Nhớ

- Toán hạng nằm trong 1 ô nhớ
- Cần tính được địa chỉ vật lý của ô nhớ (PA=Physical address)
- Nhìn chung

$$PA = \text{segment base address} + \text{effective address}$$

(SBA)
(EA)

$$PA = SBA : EA$$
 - SBA = xác định vị trí đầu của segment trong bộ nhớ
= địa chỉ segment của ô nhớ
 - EA = địa chỉ offset của ô nhớ trong segment




Chế độ định địa chỉ

- Trong đó EA có thể có 3 thành phần

$$EA = \underset{\substack{\downarrow \\ \text{BX} \\ \text{BP}}}{\text{Base}} + \underset{\substack{\downarrow \\ \text{DI} \\ \text{SI}}}{\text{Index}} + \underset{\substack{\downarrow \\ \text{8 bit} \\ \text{16 bit}}}{\text{Displacement}}$$
- Vậy PA = SBA : EA

$$= \text{Segment} : \text{Base} + \text{Index} + \text{Disp}$$

$$= \left\{ \begin{array}{c} \text{CS} \\ \text{DS} \\ \text{SS} \\ \text{ES} \end{array} \right\} : \left\{ \begin{array}{c} \text{BX} \\ \text{BP} \end{array} \right\} + \left\{ \begin{array}{c} \text{SI} \\ \text{DI} \end{array} \right\} + \left\{ \begin{array}{c} \text{8 bit - disp} \\ \text{16 bit - disp} \end{array} \right\}$$



Chế độ định địa chỉ

- Tùy vào sự có mặt của Base, Index, Disp → có phân loại khác nhau :
 - Định vị trực tiếp (địa chỉ ô nhớ nằm trong lệnh)
 - Định vị gián tiếp thanh ghi (địa chỉ ô nhớ nằm trong thanh ghi, EA chỉ có disp)
 - Định vị chỉ số nền (EA có base và index)
 - Định vị tương đối thanh ghi (EA có base và disp)
 - ...

Định vị trực tiếp

- EA=disp
- Ví dụ :
Mov AX, [1B67h]
- Kết quả AX=?

AX	0000
BX	1B69
CX	1643
DX	29A2
SP	FF03
BP	1B03
SI	0001
DI	0004

Địa chỉ	
1B66	35
1B67	69
1B68	1B
1B69	24
1B6A	01
1B6B	

Định vị gián tiếp thanh ghi

- Register indirect addressing mode : EA =Base hoặc Index
- Ví dụ
Mov AX, [BX]
- Kết quả AX = ?

AX	0000
BX	1B69
CX	1643
DX	29A2
SP	FF03
BP	1B03
SI	0001
DI	0004

Địa chỉ	
1B66	35
1B67	69
1B68	1B
1B69	24
1B6A	01
1B6B	

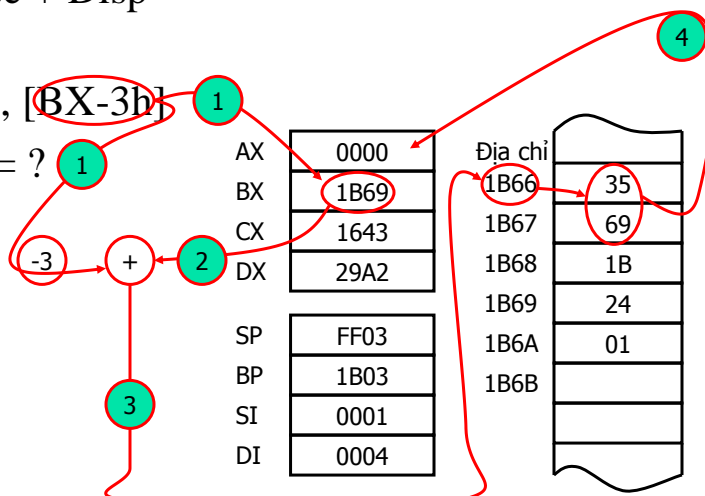
Định địa chỉ nền (base addressing mode)

- EA = Base + Disp

- Ví dụ

Mov AX, [BX-3h]

- KQ AX = ?



Định vị chỉ số

- EA = Index + Disp

- Ví dụ :

Mov AX, [SI]+1234h

- KQ AX = ?

- Tương tự trên

AX	0000	Địa chỉ	
BX	1B69	1B66	35
CX	1643	1B67	69
DX	29A2	1B68	1B
		1B69	24
SP	FF03	1B6A	01
BP	1B03	1B6B	
SI	0001		
DI	0004		

