

# Chương 3: Bộ xử lý CPU

- 3.1 Cấu trúc cơ bản của CPU
- 3.2 Tập lệnh
- 3.3 Hoạt động của CPU
- 3.4 Kiến trúc của các bộ xử lý tiên tiến
- 3.5 Kiến trúc tập lệnh Intel x86

# Chương 3: Bộ xử lý CPU

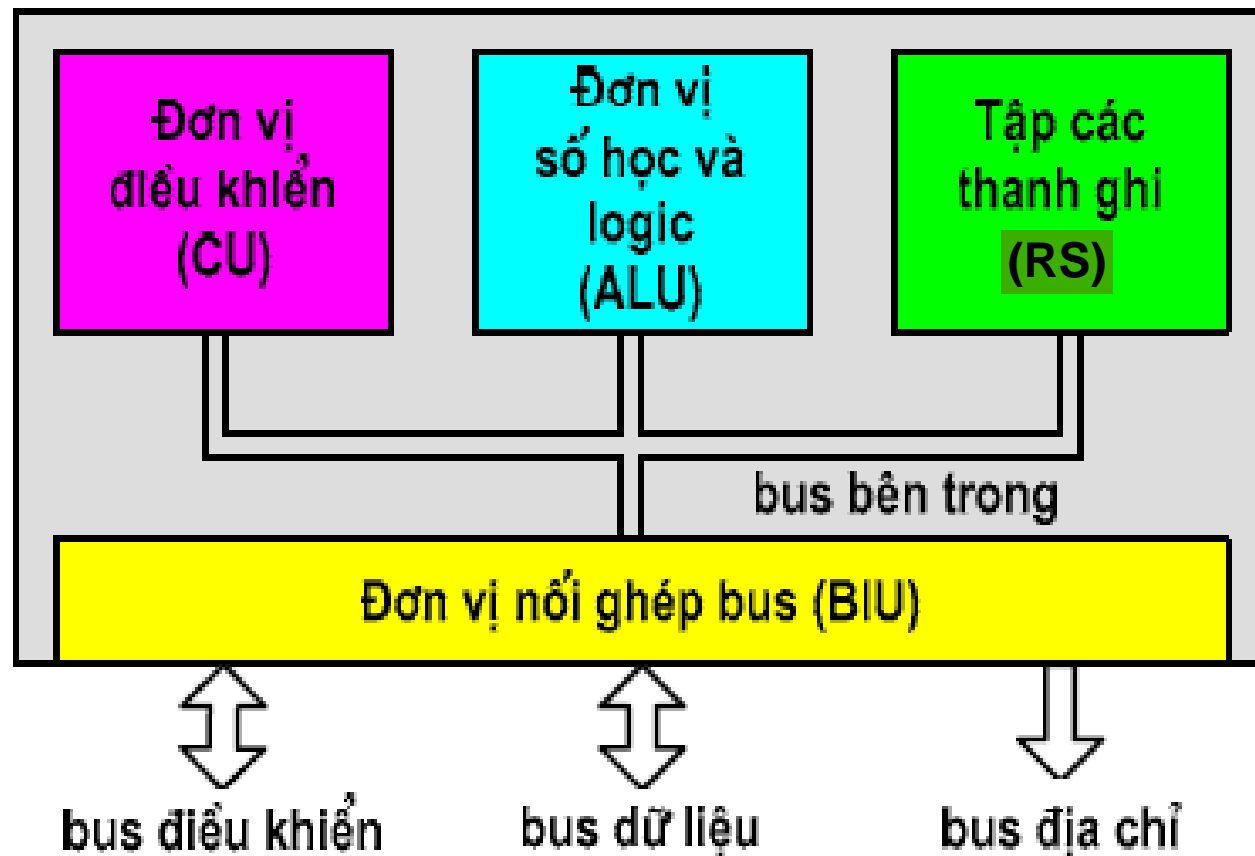
- 3.1 Cấu trúc cơ bản của CPU
- 3.2 Tập lệnh
- 3.3 Hoạt động của CPU
- 3.4 Kiến trúc của các bộ xử lý tiên tiến
- 3.5 Kiến trúc tập lệnh Intel x86

# 3.1. Cấu trúc cơ bản của CPU

## 1. Nhiệm vụ và cấu trúc của CPU

- Nhiệm vụ của CPU:
  - Nhận lệnh (Fetch Instruction): CPU đọc lệnh từ bộ nhớ.
  - Giải mã lệnh (Decode Instruction): xác định thao tác mà lệnh yêu cầu.
  - Nhận dữ liệu (Fetch Data): nhận dữ liệu từ bộ nhớ hoặc các cổng vào-ra.
  - Xử lý dữ liệu (Process Data): thực hiện phép toán số học hay phép toán logic với các dữ liệu.
  - Ghi dữ liệu (Write Data): ghi dữ liệu ra bộ nhớ hay cổng vào-ra

# Sơ đồ cấu trúc cơ bản của CPU



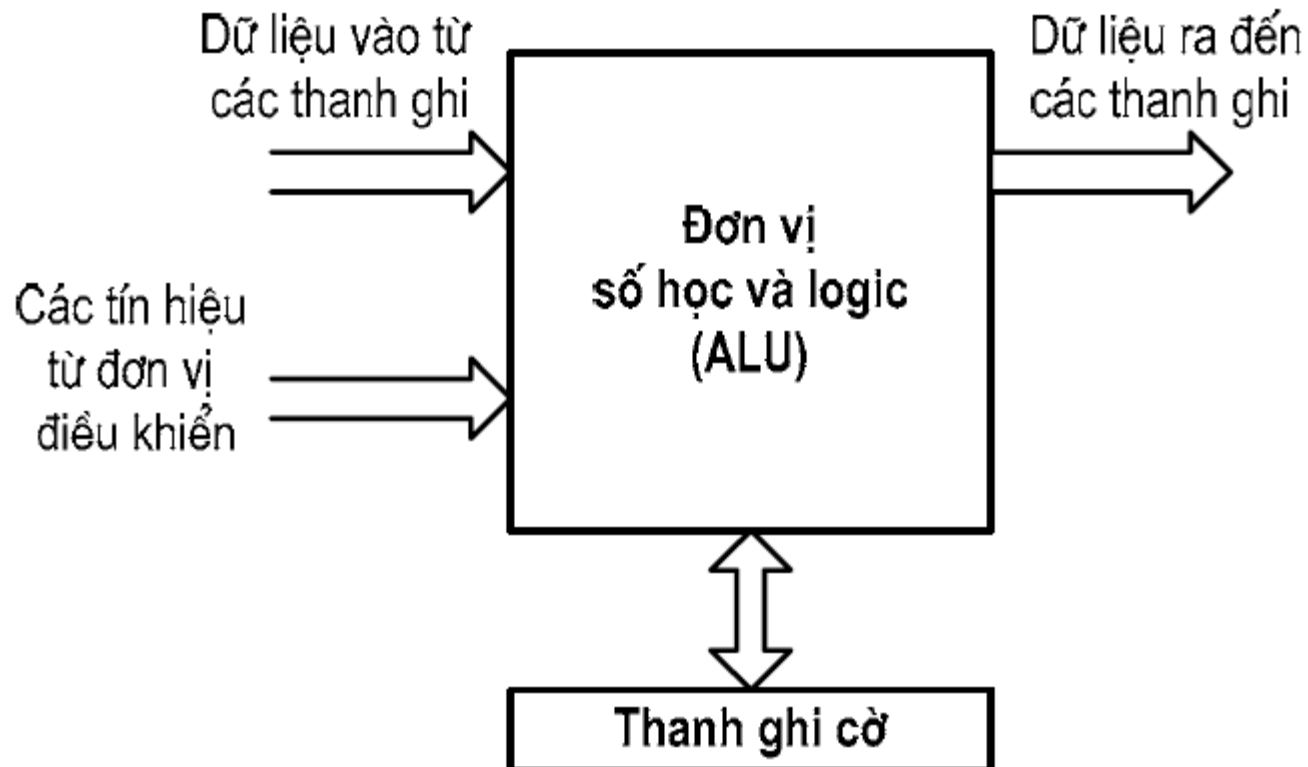
# Các thành phần cơ bản của CPU

- Đơn vị điều khiển (Control Unit - CU)
- Đơn vị số học và logic (Arithmetic and Logic Unit - ALU)
- Tập thanh ghi (Register Set - RS)
- Đơn vị nối ghép bus (Bus Interface Unit – BIU)
- Bus bên trong (Internal Bus)

## 2. Đơn vị số học và logic

- Chức năng: Thực hiện các phép toán số học và phép toán logic:
  - Số học: cộng, trừ, nhân, chia, tăng, giảm, đảo dấu
  - Logic: AND, OR, XOR, NOT, phép dịch bit.

# Mô hình kết nối ALU



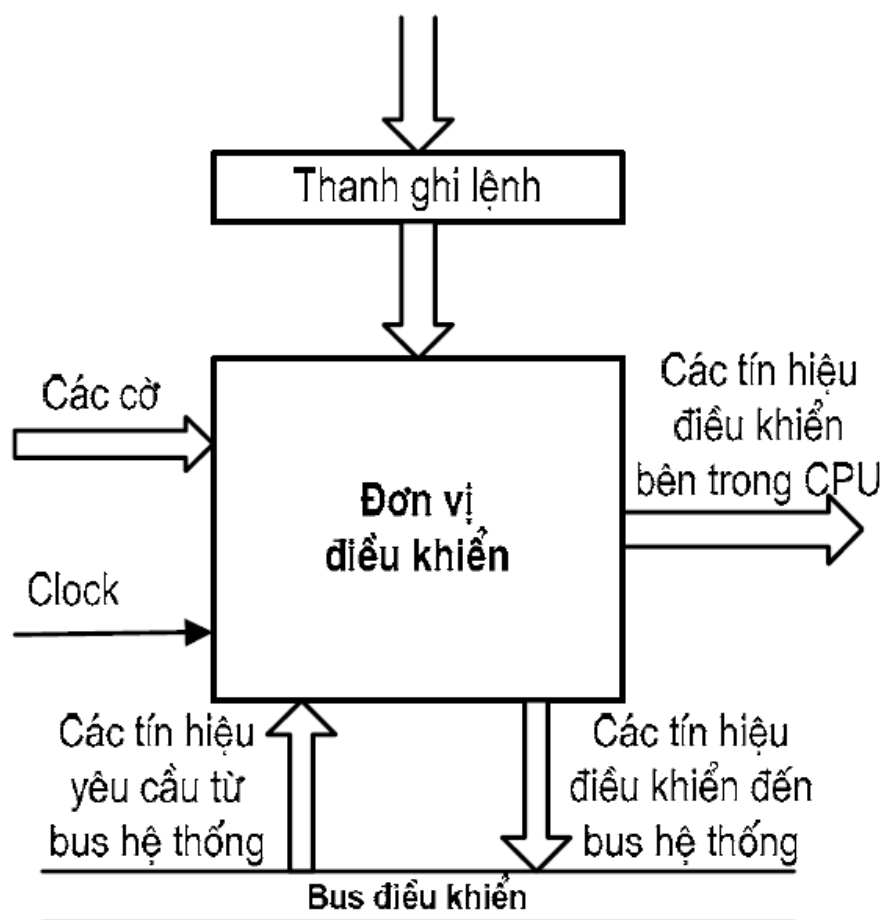
# 3. Đơn vị điều khiển

## ■ Chức năng

- Điều khiển nhận lệnh từ bộ nhớ đưa vào thanh ghi lệnh
- Tăng nội dung của PC để trở sang lệnh kế tiếp
- Giải mã lệnh đã được nhận để xác định thao tác mà lệnh yêu cầu
- Phát ra các tín hiệu điều khiển thực hiện lệnh
- Nhận các tín hiệu yêu cầu từ bus hệ thống và đáp ứng với các yêu cầu đó.



# Mô hình kết nối đơn vị điều khiển



# Các tín hiệu đưa đến đơn vị điều khiển

- Clock: tín hiệu nhịp từ mạch tạo dao động bên ngoài.
- Mã lệnh từ thanh ghi lệnh đưa đến để giải mã.
- Các cờ từ thanh ghi cờ cho biết trạng thái của CPU.
- Các tín hiệu yêu cầu từ bus điều khiển

# Các tín hiệu phát ra từ đơn vị điều khiển

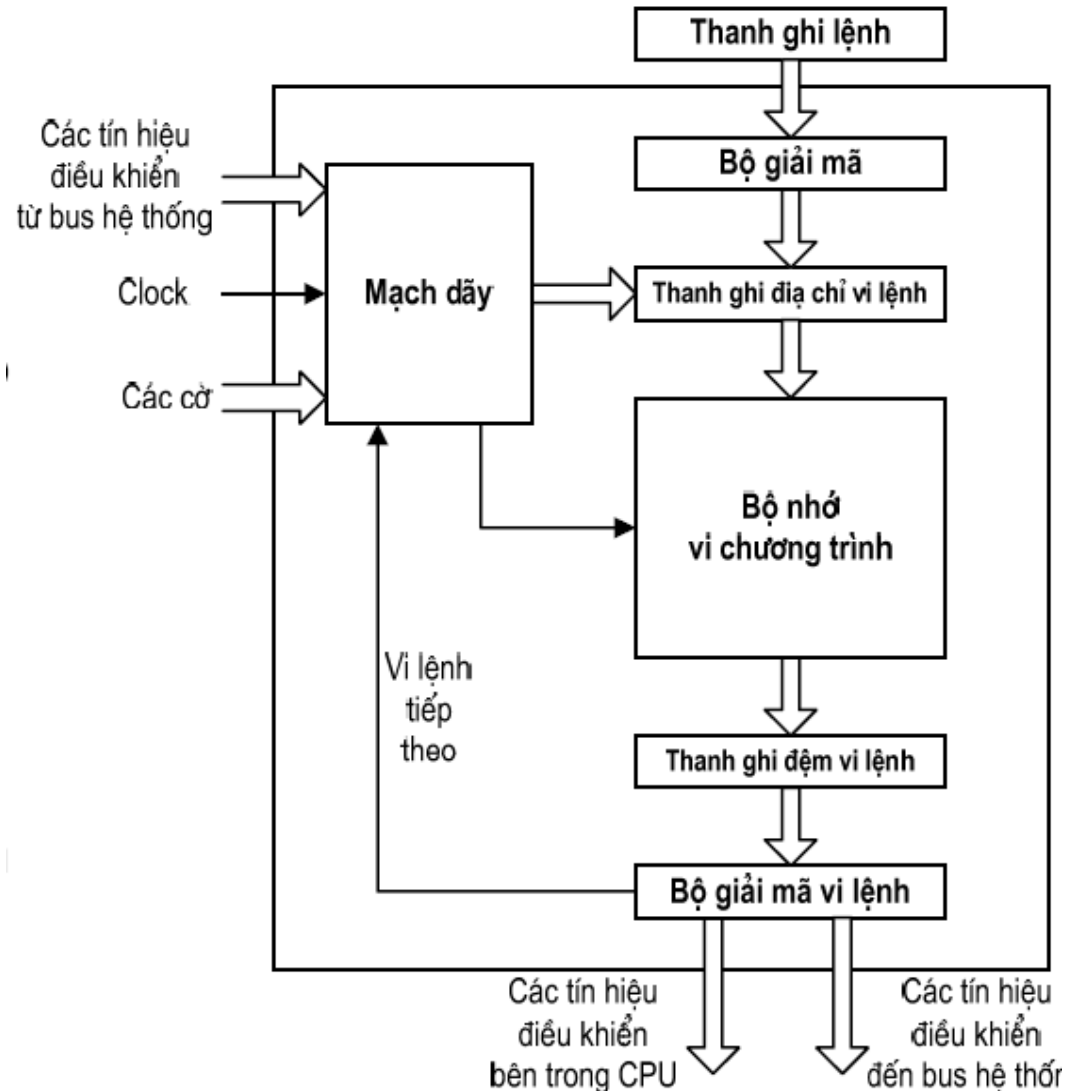
- Các tín hiệu điều khiển bên trong CPU:
  - Điều khiển các thanh ghi
  - Điều khiển ALU
- Các tín hiệu điều khiển bên ngoài CPU:
  - Điều khiển bộ nhớ
  - Điều khiển các mô-đun vào-ra

# Các phương pháp thiết kế đơn vị điều khiển

- Đơn vị điều khiển vi chương trình (Microprogrammed Control Unit)
- Đơn vị điều khiển nối kết cứng (Hardwired Control Unit)

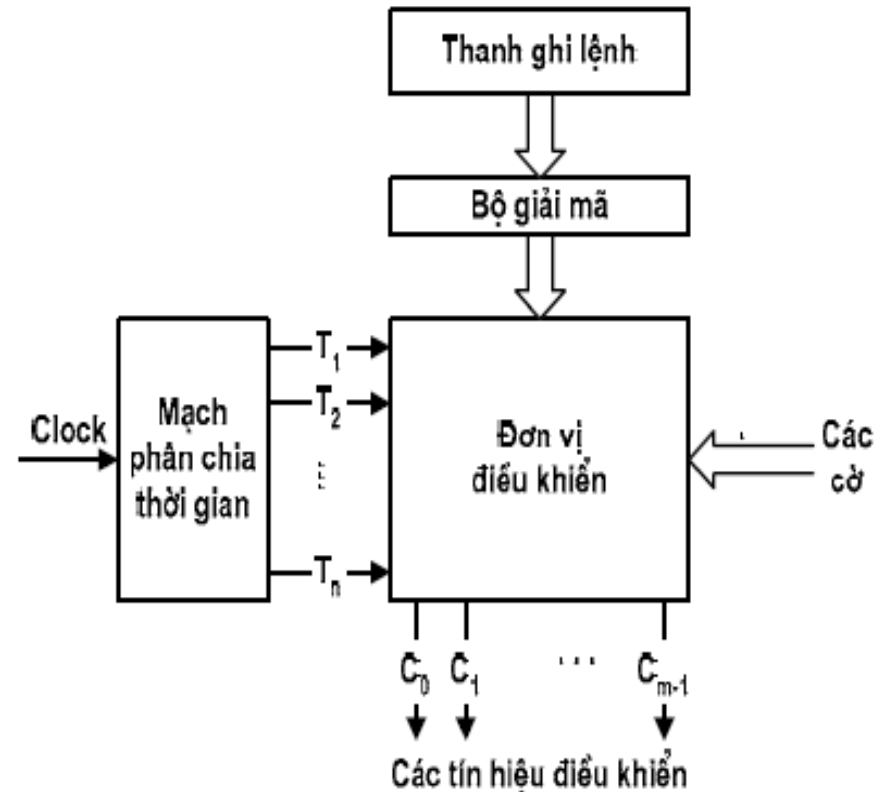
# Đơn vị điều khiển vi chương trình

- Bộ nhớ vi chương trình (ROM) lưu trữ các vi chương trình (microprogram)
- Một vi chương trình bao gồm các vi lệnh (microinstruction)
- Mỗi vi lệnh mã hoá cho một vi thao tác (microoperation)
- Để hoàn thành một lệnh cần thực hiện một hoặc một vài vi chương trình
- Tốc độ chậm



# Đơn vị điều khiển nối kết cứng

- Sử dụng mạch cứng để giải mã và tạo các tín hiệu điều khiển thực hiện lệnh
- Tốc độ nhanh
- Đơn vị điều khiển phức tạp



## 4. Tập thanh ghi

### ■ Chức năng và đặc điểm:

- Chứa các thông tin tạm thời phục vụ cho hoạt động ở thời điểm hiện tại của CPU
- Được coi là mức đầu tiên của hệ thống nhớ
- Số lượng thanh ghi nhiều → tăng hiệu năng của CPU
- Có hai loại thanh ghi:
  - Các thanh ghi lập trình được
  - Các thanh ghi không lập trình được

# Phân loại thanh ghi theo chức năng

- Thanh ghi địa chỉ: quản lý địa chỉ của ngăn nhớ hay cổng vào-ra.
- Thanh ghi dữ liệu: chứa tạm thời các dữ liệu.
- Thanh ghi đa năng: có thể chứa địa chỉ hoặc dữ liệu.
- Thanh ghi điều khiển/trạng thái: chứa các thông tin điều khiển và trạng thái của CPU.
- Thanh ghi lệnh: chứa lệnh đang được thực hiện



# Một số thanh ghi điển hình

## ■ Các thanh ghi địa chỉ

- Bộ đếm chương trình PC (Program Counter)
- Con trỏ dữ liệu DP (Data Pointer)
- Con trỏ ngăn xếp SP (Stack Pointer)
- Thanh ghi cơ sở và thanh ghi chỉ số (Base Register & Index Register)

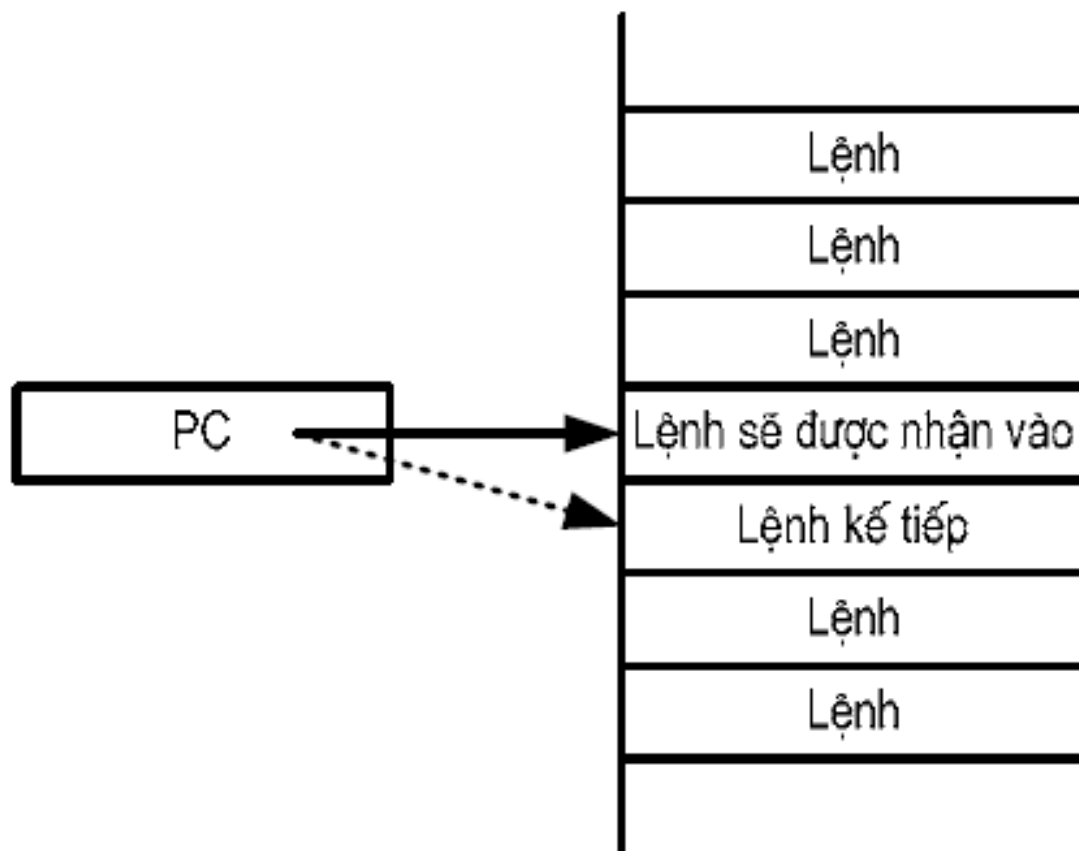
## ■ Các thanh ghi dữ liệu

## ■ Thanh ghi trạng thái

# Bộ đếm chương trình PC

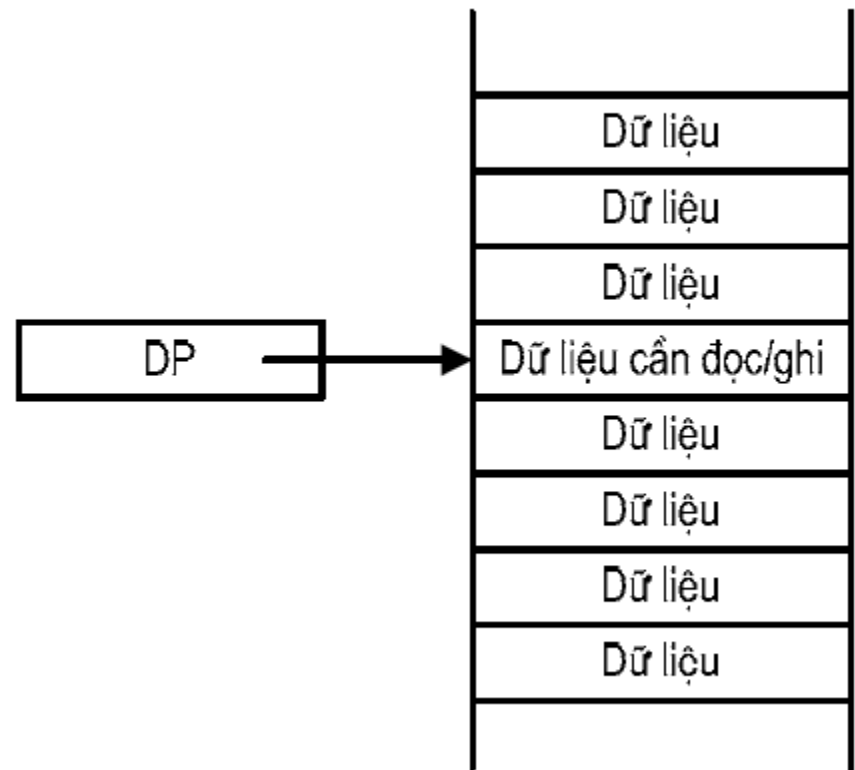
- Còn được gọi là con trỏ lệnh IP (Instruction Pointer)
- Giữ địa chỉ của lệnh tiếp theo sẽ được nhận vào.
- Sau khi một lệnh được nhận vào, nội dung PC tự động tăng để trở sang lệnh kế tiếp.

# Minh họa bộ đếm chương trình



# Thanh ghi con trỏ dữ liệu

- Chứa địa chỉ của ngăn nhớ dữ liệu mà CPU muốn truy nhập
- Thường có một số thanh ghi con trỏ dữ liệu



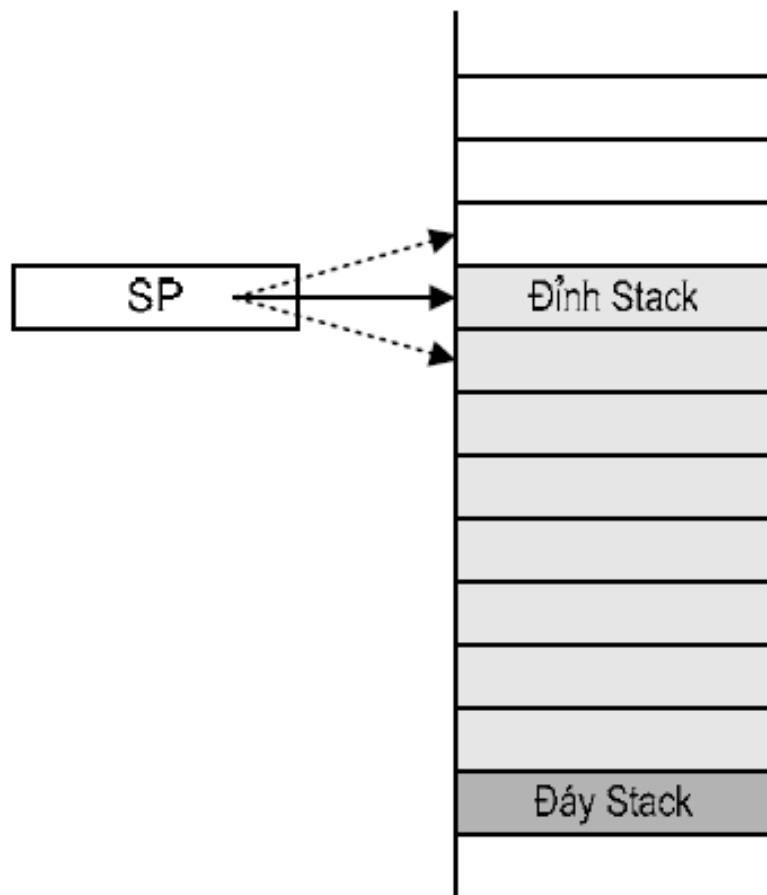
# Ngăn xếp (Stack)

- Ngăn xếp là vùng nhớ có cấu trúc LIFO (Last In - First Out)
- Ngăn xếp thường dùng để phục vụ cho chương trình con
- Đáy ngăn xếp là một ngăn nhớ xác định
- Đỉnh ngăn xếp là thông tin nằm ở vị trí trên cùng trong ngăn xếp
- Đỉnh ngăn xếp có thể bị thay đổi

# Con trỏ ngăn xếp SP (Stack Pointer)

- Chứa địa chỉ của ngăn nhớ đỉnh ngăn xếp
- Khi cất một thông tin vào ngăn xếp:
  - Nội dung của SP tự động giảm
  - Thông tin được cất vào ngăn nhớ được trỏ bởi SP
- Khi lấy một thông tin ra khỏi ngăn xếp:
  - Thông tin được đọc từ ngăn nhớ được trỏ bởi SP
  - Nội dung của SP tự động tăng
- Khi ngăn xếp rỗng, SP trỏ vào đáy

# Minh họa con trỏ ngăn xếp SP

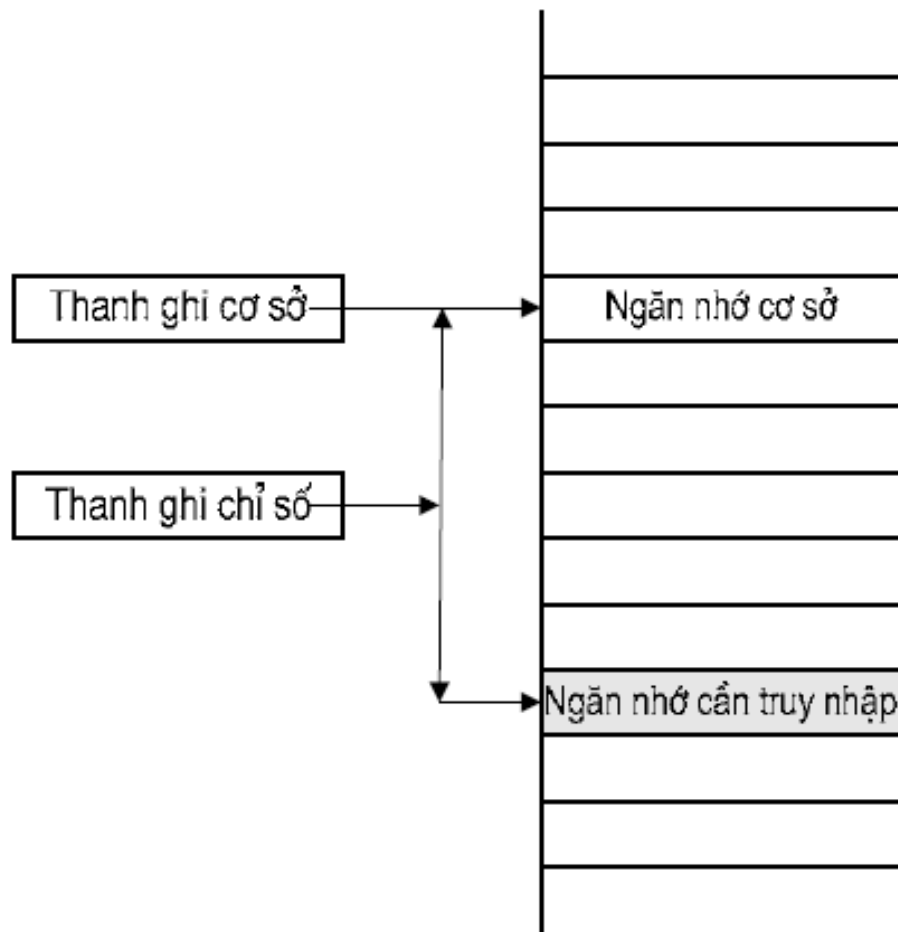


# Thanh ghi cơ sở và thanh ghi chỉ số

- Thanh ghi cơ sở: chứa địa chỉ của ngăn nhớ cơ sở (địa chỉ cơ sở)
- Thanh ghi chỉ số: chứa độ lệch địa chỉ giữa ngăn nhớ mà CPU cần truy nhập so với ngăn nhớ cơ sở (chỉ số)
- Địa chỉ của ngăn nhớ cần truy nhập = địa chỉ cơ sở + chỉ số



# Minh họa thanh ghi cơ sở và thanh ghi chỉ số



# Các thanh ghi dữ liệu

- Chứa các dữ liệu tạm thời hoặc các kết quả trung gian
- Cần có nhiều thanh ghi dữ liệu
- Các thanh ghi số nguyên: 8, 16, 32, 64 bit
- Các thanh ghi số dấu phẩy động

# Thanh ghi trạng thái (Status Register)

- Còn gọi là thanh ghi cờ (Flag Register)
- Chứa các thông tin trạng thái của CPU
  - Các cờ phép toán: báo hiệu trạng thái của kết quả phép toán
  - Các cờ điều khiển: biểu thị trạng thái điều khiển của CPU

# Ví dụ cờ phép toán

- Cờ Zero (cờ rỗng): được thiết lập lên 1 khi kết quả của phép toán bằng 0.
- Cờ Sign (cờ dấu): được thiết lập lên 1 khi kết quả phép toán nhỏ hơn 0
- Cờ Carry (cờ nhớ): được thiết lập lên 1 nếu phép toán có nhớ ra ngoài bit cao nhất → cờ báo tràn với số không dấu.
- Cờ Overflow (cờ tràn): được thiết lập lên 1 nếu cộng hai số nguyên cùng dấu mà kết quả có dấu ngược lại → cờ báo tràn với số có dấu.

# Ví dụ cờ điều khiển

- Cờ Interrupt (Cờ cho phép ngắt):
- Nếu  $IF = 1 \rightarrow$  CPU ở trạng thái cho phép ngắt với tín hiệu yêu cầu ngắt từ bên ngoài gửi tới
- Nếu  $IF = 0 \rightarrow$  CPU ở trạng thái cấm ngắt với tín hiệu yêu cầu ngắt từ bên ngoài gửi tới

# Tập thanh ghi của một số bộ xử lý

Data Registers	
D0	
D1	
D2	
D3	
D4	
D5	
D6	
D7	

Address Registers	
A0	
A1	
A2	
A3	
A4	
A5	
A6	
A7	
A7'	

Program Status	
Program Counter	
Status Register	

(a) MC68000

General Registers

AX	Accumulator
BX	Base
CX	Count
DX	Data

Pointer & Index

SP	Stack Pointer
BP	Base Pointer
SI	Source Index
DI	Dest Index

Segment

CS	Code
DS	Data
SS	Stack
ES	Extra

Program Status

Instr Ptr
Flags

(b) 8086

General Registers

EAX	AX
EBX	BX
ECX	CX
EDX	DX

ESP	SP
EBP	BP
ESI	SI
EDI	DI

Program Status

FLAGS Register
Instruction Pointer

(c) 80386 - Pentium II

# Chương 3: Bộ xử lý CPU

- 3.1 Cấu trúc cơ bản của CPU
- 3.2 Tập lệnh
- 3.3 Hoạt động của CPU
- 3.4 Kiến trúc của các bộ xử lý tiên tiến
- 3.5 Kiến trúc tập lệnh Intel x86

## 3.2. Tập lệnh

### 1. Giới thiệu chung về tập lệnh

- Mỗi bộ xử lý có một tập lệnh xác định
- Tập lệnh thường có hàng chục đến hàng trăm lệnh
- Mỗi lệnh là một chuỗi số nhị phân mà bộ xử lý hiểu được để thực hiện một thao tác xác định.
- Các lệnh được mô tả bằng các ký hiệu gọi nhớ  
→ chính là các lệnh của hợp ngữ



# Các thành phần của lệnh máy

Mã thao tác	Địa chỉ của các toán hạng
-------------	---------------------------

- Mã thao tác (operation code → opcode): mã hóa cho thao tác mà bộ xử lý phải thực hiện
- Địa chỉ toán hạng: chỉ ra nơi chứa các toán hạng mà thao tác sẽ tác động
  - Toán hạng nguồn: dữ liệu vào của thao tác
  - Toán hạng đích: dữ liệu ra của thao tác

# Số lượng địa chỉ toán hạng trong lệnh (1)

## ■ Ba địa chỉ toán hạng:

- 2 toán hạng nguồn, 1 toán hạng đích
- $c = a + b$
- Từ lệnh dài vì phải mã hoá địa chỉ cho cả ba toán hạng
- Được sử dụng trên các bộ xử lý tiên tiến

# Số lượng địa chỉ toán hạng trong lệnh (2)

## ■ Hai địa chỉ toán hạng:

- Một toán hạng vừa là toán hạng nguồn vừa là toán hạng đích; toán hạng còn lại là toán hạng nguồn
- $a = a + b$
- Giá trị cũ của 1 toán hạng nguồn bị mất vì phải chứa kết quả
- Rút gọn độ dài từ lệnh
- Phổ biến

# Số lượng địa chỉ toán hạng trong lệnh (3)

- Một địa chỉ toán hạng:
  - Một toán hạng được chỉ ra trong lệnh
  - Một toán hạng là ngầm định → thường là thanh ghi (thanh chứa –accumulator)
  - Được sử dụng trên các máy ở các thế hệ trước

# Số lượng địa chỉ toán hạng trong lệnh (4)

## ■ 0 địa chỉ toán hạng:

- Các toán hạng đều được ngầm định
- Sử dụng Stack
- Ví dụ:  
push a  
push b  
add  
pop c  
có nghĩa là :  $c = a + b$
- không thông dụng

# Đánh giá về số địa chỉ toán hạng

## ■ Nhiều địa chỉ toán hạng

- Các lệnh phức tạp hơn
- Cần nhiều thanh ghi
- Chương trình có ít lệnh hơn
- Nhận lệnh và thực hiện lệnh chậm hơn

## ■ Ít địa chỉ toán hạng

- Các lệnh đơn giản hơn
- Cần ít thanh ghi
- Chương trình có nhiều lệnh hơn
- Nhận lệnh và thực hiện lệnh nhanh hơn

# Các vấn đề của thiết kế tập lệnh (1)

## ■ Về thao tác

- Bao nhiêu thao tác ?
- Các thao tác nào ?
- Mức độ phức tạp của các thao tác ?

## ■ Các kiểu dữ liệu

## ■ Các khuôn dạng lệnh

- Độ dài của trường mã thao tác
- Số lượng địa chỉ toán hạng

# Các vấn đề của thiết kế tập lệnh (2)

## ■ Các thanh ghi

- Số thanh ghi của CPU được sử dụng
- Các thao tác nào được thực hiện trên các thanh ghi

## ■ Các phương pháp định địa chỉ (xét sau) (addressing modes)

## ■ RISC hay CISC (xét sau)

- Reduced Instruction Set Computing
- Complex Instruction Set Computing



## 2. Các kiểu thao tác cơ bản

- Chuyển dữ liệu
- Xử lý số học với số nguyên
- Xử lý logic
- Điều khiển vào-ra
- Chuyển điều khiển (rẽ nhánh)
- Điều khiển hệ thống

# Các lệnh chuyển dữ liệu

- MOVE Copy dữ liệu từ nguồn đến đích
- LOAD Nạp dữ liệu từ bộ nhớ đến bộ xử lý
- STORE cất dữ liệu từ bộ xử lý đến bộ nhớ
- EXCHANGE Trao đổi nội dung của nguồn và đích
- CLEAR Chuyển các bit 0 vào toán hạng đích
- SET Chuyển các bit 1 vào toán hạng đích
- PUSH Cất nội dung toán hạng nguồn vào ngăn xếp
- POP Lấy nội dung đỉnh ngăn xếp đưa đến toán hạng đích

# Các lệnh số học

- ADD Cộng hai toán hạng
- SUBTRACT Trừ hai toán hạng
- MULTIPLY Nhân hai toán hạng
- DIVIDE Chia hai toán hạng
- ABSOLUTE Lấy trị tuyệt đối toán hạng
- NEGATE Đổi dấu toán hạng (lấy bù 2)
- INCREMENT Tăng toán hạng thêm 1
- DECREMENT Giảm toán hạng đi 1
- COMPARE Trừ hai toán hạng để lập cờ

# Các lệnh logic

- AND Thực hiện phép AND hai toán hạng
- OR Thực hiện phép OR hai toán hạng
- XOR Thực hiện phép XOR hai toán hạng
- NOT Đảo bit của toán hạng (lấy bù 1)
- TEST Thực hiện phép AND hai toán hạng để lập cờ

# Minh hoạ các lệnh AND, OR, XOR

- Giả sử có hai thanh ghi chứa dữ liệu như sau:

$(R1) = 1010\ 1010$

$(R2) = 0000\ 1111$

- $R1 \rightarrow (R1) \text{ AND } (R2) = 0000\ 1010$

Phép toán AND dùng để xoá một số bit và giữ nguyên một số bit còn lại của toán hạng.

- $R1 \rightarrow (R1) \text{ OR } (R2) = 1010\ 1111$

Phép toán OR dùng để thiết lập một số bit và giữ nguyên một số bit còn lại của toán hạng.

- $R1 \rightarrow (R1) \text{ XOR } (R2) = 1010\ 0101$

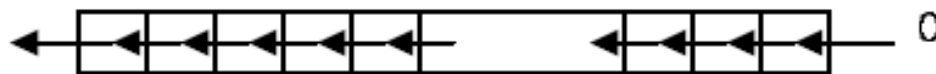
Phép toán XOR dùng để đảo một số bit và giữ nguyên một số bit còn lại của toán hạng.

# Các lệnh logic (tiếp)

- SHIFT Dịch trái (phải) toán hạng
- ROTATE Quay trái (phải) toán hạng

# Các thao tác SHIFT và ROTATE

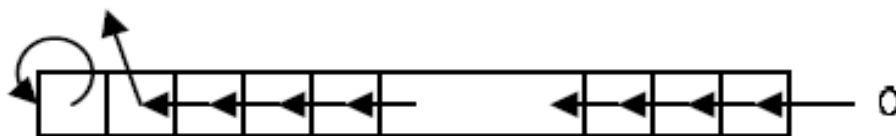
Dịch trái logic



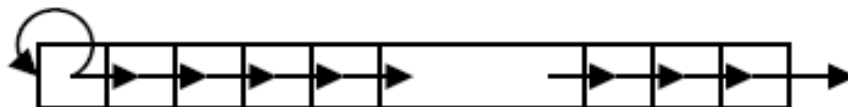
Dịch phải logic 0



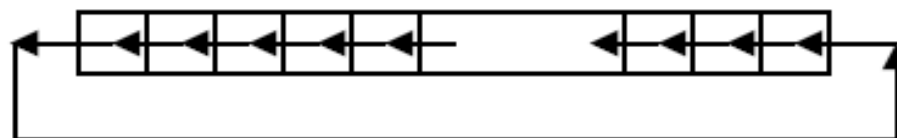
Dịch trái số học



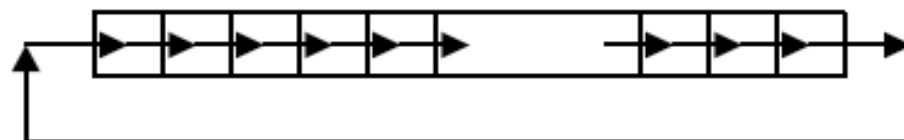
Dịch phải số học



Quay trái logic



Quay phải logic



# Các lệnh vào ra chuyên dụng

- INPUT            Copy dữ liệu từ một cổng xác định đưa đến đích
- OUTPUT        Copy dữ liệu từ nguồn đến một cổng xác định

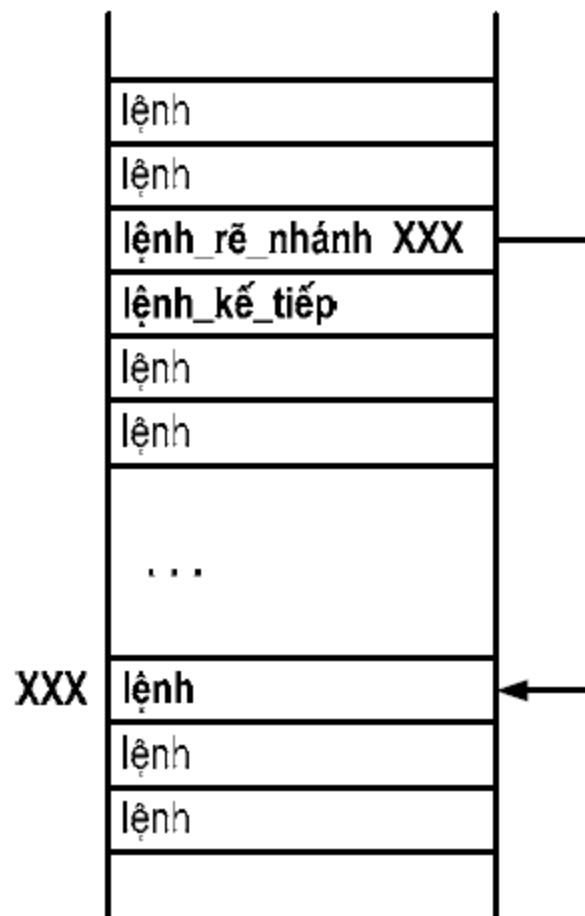


# Các lệnh chuyển điều khiển

- **JUMP (BRANCH)**      Lệnh nhảy không điều kiện:
  - nạp vào PC một địa chỉ xác định
- **JUMP CONDITIONAL**      Lệnh nhảy có điều kiện:
  - điều kiện đúng → nạp vào PC một địa chỉ xác định
  - điều kiện sai → không làm gì cả
- **CALL**      Lệnh gọi chương trình con:
  - cất nội dung của PC (địa chỉ trở về) ra một vị trí xác định (thường ở Stack)
  - Nạp vào PC địa chỉ của lệnh đầu tiên của chương trình con
- **RETURN**      Lệnh trở về từ chương trình con:
  - Khôi phục địa chỉ trở về trả lại cho PC để trở về chương trình chính

# Lệnh rẽ nhánh không điều kiện

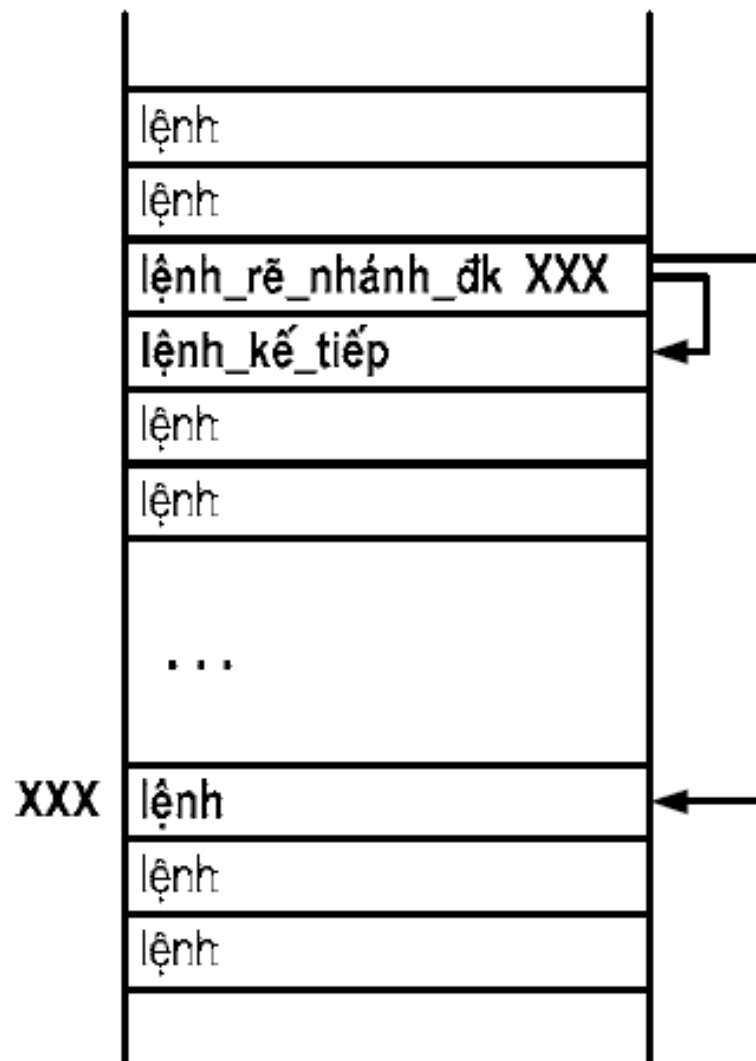
- Chuyển tới thực hiện lệnh ở vị trí có địa chỉ XXX:
- $PC \leftarrow XXX$



# Lệnh rẽ nhánh có điều kiện

- Trong lệnh có kèm theo điều kiện
- Kiểm tra điều kiện trong lệnh:
  - Nếu điều kiện đúng → chuyển tới thực hiện lệnh ở vị trí có địa chỉ XXX
$$PC \leftarrow XXX$$
  - Nếu điều kiện sai → chuyển sang thực hiện lệnh\_kế\_tiếp
- Điều kiện thường được kiểm tra thông qua các cờ
- Có nhiều lệnh rẽ nhánh có điều kiện

# Minh hoạ lệnh rẽ nhánh có điều kiện



# Lệnh CALL và RETURN

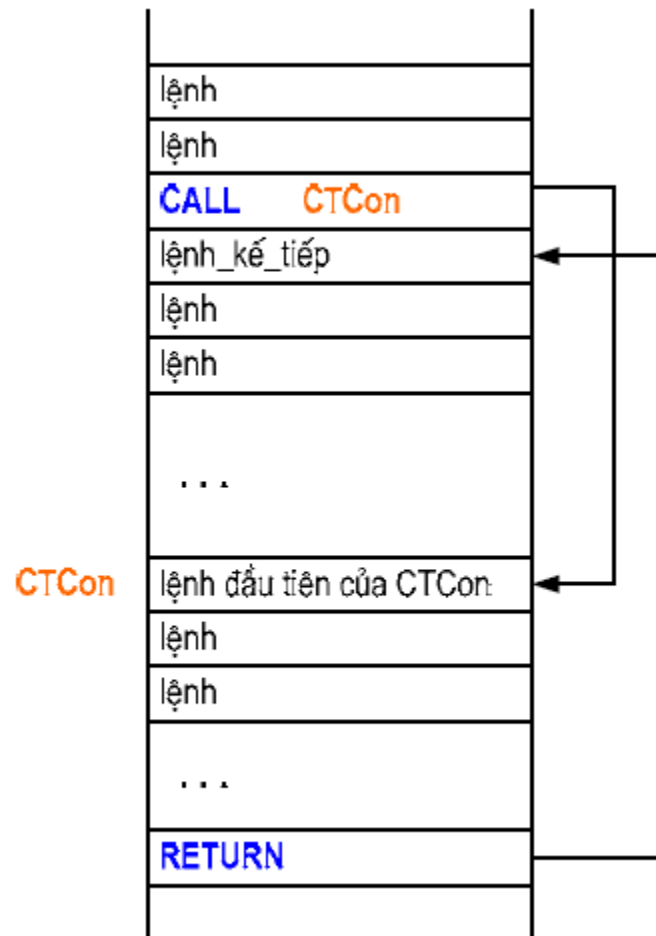
## ■ Lệnh gọi chương trình con: lệnh CALL

- Cất nội dung PC (chứa địa chỉ của lệnh\_kế\_tiếp) ra Stack
- Nạp vào PC địa chỉ của lệnh đầu tiên của chương trình con được gọi  
→ Bộ xử lý được chuyển sang thực hiện chương trình con tương ứng

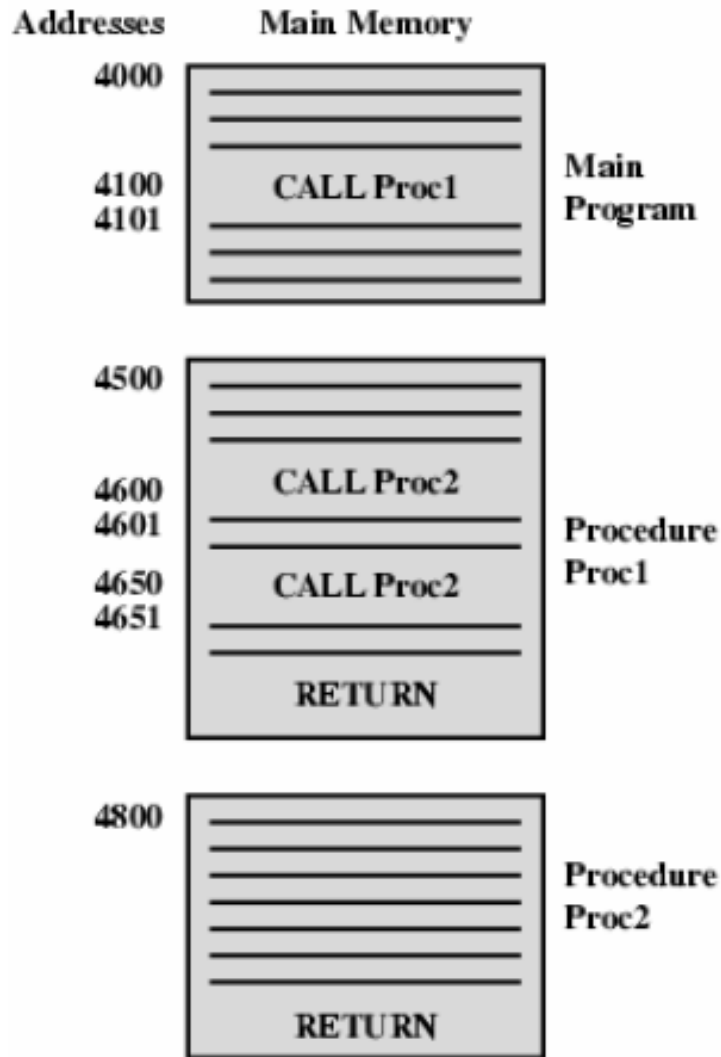
## ■ Lệnh trở về từ chương trình con: lệnh RETURN

- Lấy địa chỉ của lệnh\_kế\_tiếp được cất ở Stack nạp trả lại cho PC → Bộ xử lý được điều khiển quay trở về thực hiện tiếp lệnh nằm sau lệnh CALL

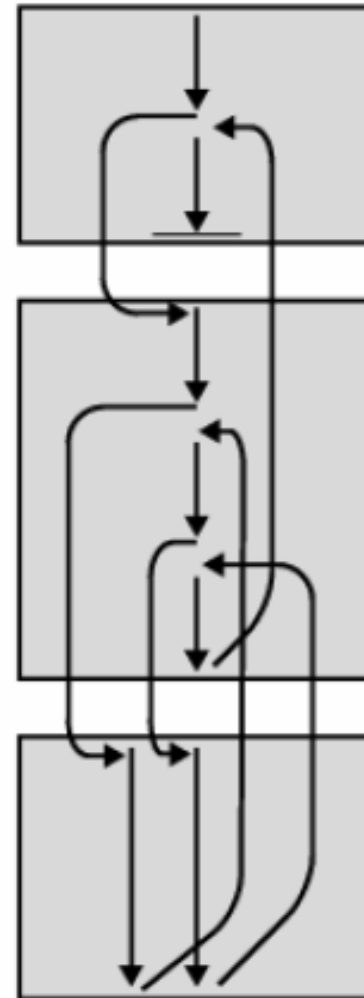
# Minh hoạ lệnh CALL và RETURN



# Gọi các thủ tục lồng nhau

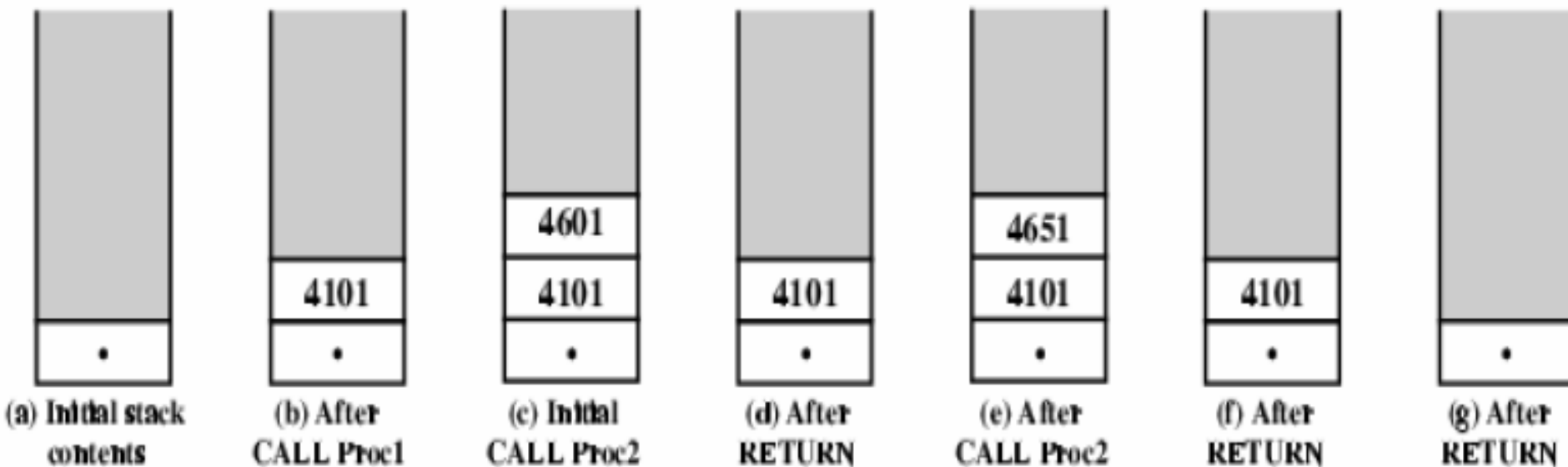


(a) Calls and returns



(b) Execution sequence

# Sử dụng Stack





# Các lệnh điều khiển hệ thống

- HALT      Dừng thực hiện chương trình
- WAIT      Tạm dừng thực hiện chương trình, lặp kiểm tra điều kiện cho đến khi thoả mãn thì tiếp tục thực hiện
- NO OPERATION      Không thực hiện gì cả
- LOCK      Cấm không cho xin chuyển nhượng bus
- UNLOCK      Cho phép xin chuyển nhượng bus

### 3. Các phương pháp định địa chỉ (addressing modes)

Khái niệm về định địa chỉ (addressing)

- Toán hạng của lệnh có thể là:
  - Một giá trị cụ thể nằm ngay trong lệnh
  - Nội dung của thanh ghi
  - Nội dung của ngăn nhớ hoặc cổng vào-ra
- Phương pháp định địa chỉ là cách thức địa chỉ hóa trong trường địa chỉ của lệnh để xác định nơi chứa toán hạng

# Các phương pháp định địa chỉ thông dụng

- Định địa chỉ tức thì
- Định địa chỉ thanh ghi
- Định địa chỉ trực tiếp
- Định địa chỉ gián tiếp qua thanh ghi
- Định địa chỉ gián tiếp
- Định địa chỉ dịch chuyển

# Định địa chỉ tức thì

- Toán hạng nằm ngay trong Trường địa chỉ của lệnh
- Chỉ có thể là toán hạng nguồn
- Ví dụ:

ADD R1, 5 ;  $R1 \leftarrow R1 + 5$

- Không tham chiếu bộ nhớ
- Truy nhập toán hạng rất nhanh
- Dải giá trị của toán hạng bị hạn chế

Mã thao tác		Toán hạng
-------------	--	-----------

# Định địa chỉ thanh ghi

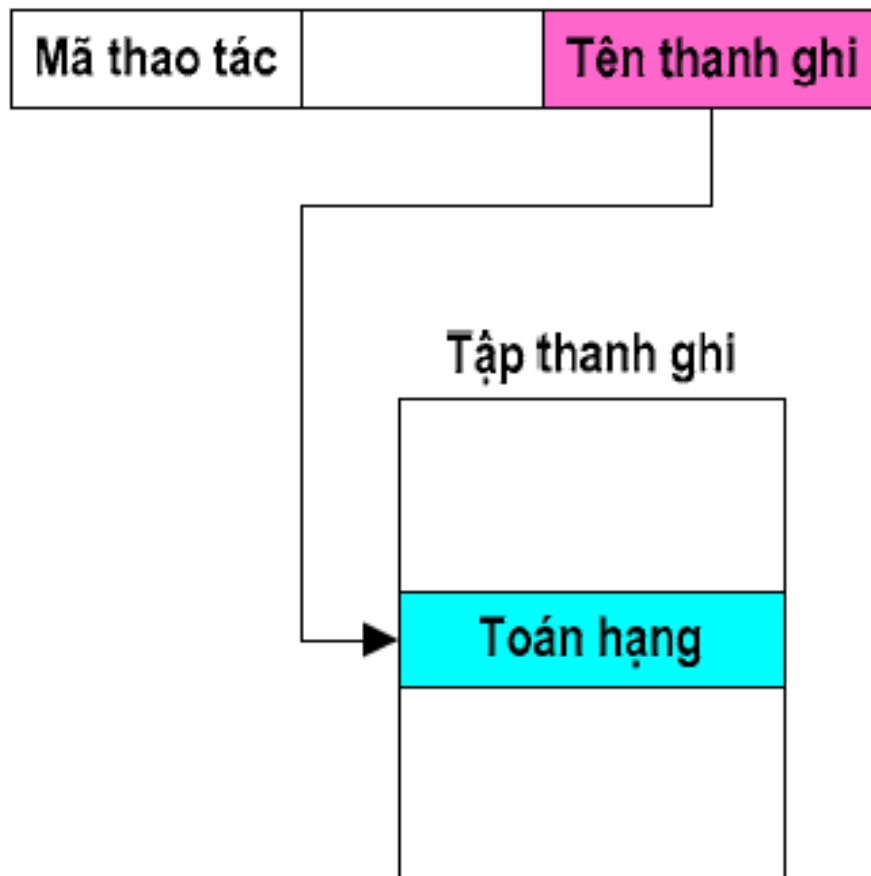
- Toán hạng được chứa trong thanh ghi có tên trong Trường địa chỉ

- Ví dụ:

ADD R1, R2 ;     $R1 \leftarrow R1 + R2$

- Số lượng thanh ghi ít  $\rightarrow$  Trường địa chỉ chỉ cần ít bit
- Không tham chiếu bộ nhớ
- Truy nhập toán hạng nhanh
- Tăng số lượng thanh ghi  $\rightarrow$  hiệu quả hơn

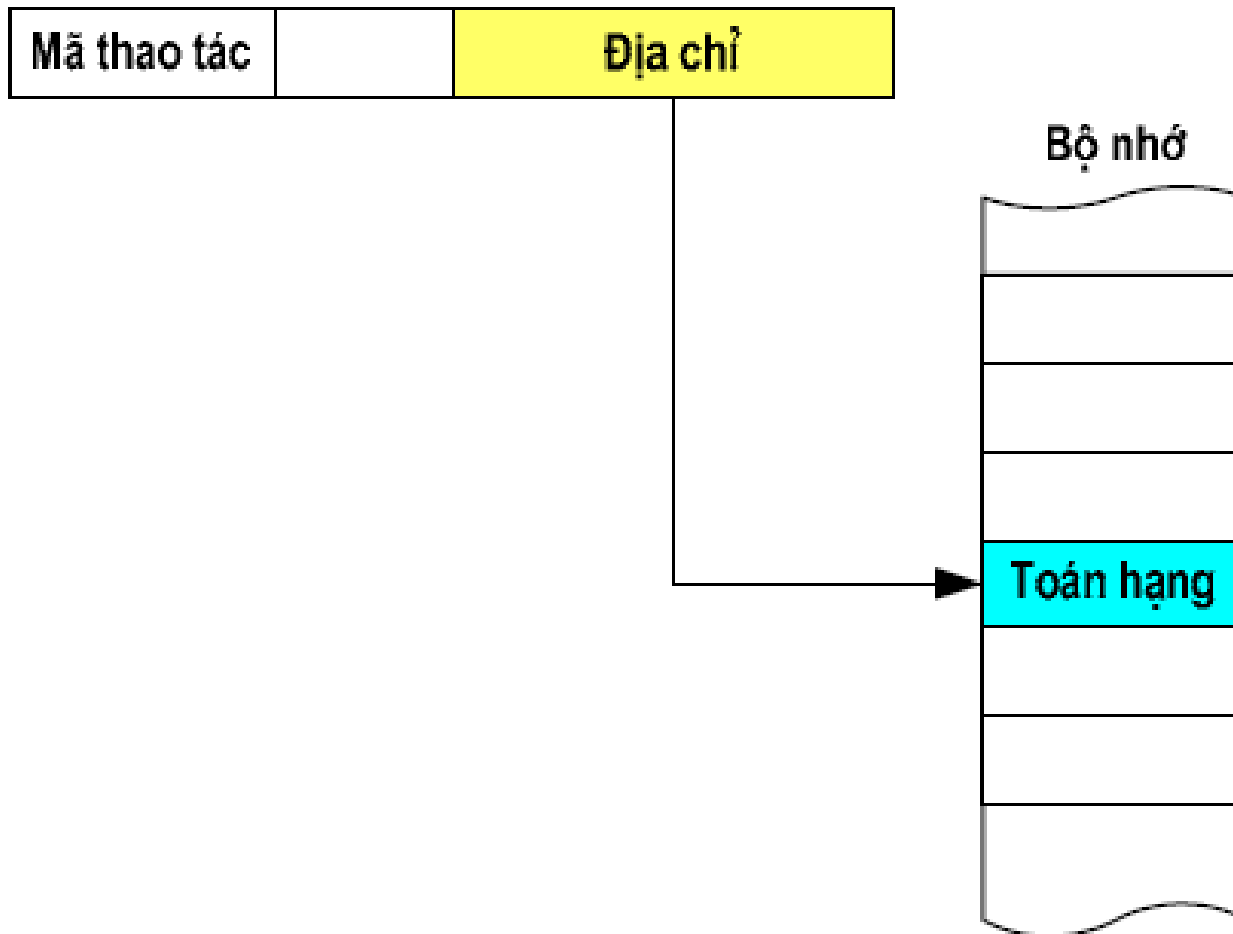
# Sơ đồ định địa chỉ thanh ghi



# Định địa chỉ trực tiếp

- Toán hạng là ngăn nhớ có địa chỉ được chỉ ra trực tiếp trong Trường địa chỉ của lệnh
- Ví dụ:  $\text{ADD R1, A ; R1} \leftarrow \text{R1} + (\text{A})$ 
  - Cộng nội dung thanh ghi R1 với nội dung của ngăn nhớ có địa chỉ là A
  - Tìm toán hạng trong bộ nhớ ở địa chỉ A
- CPU tham chiếu bộ nhớ một lần để truy nhập dữ liệu

# Sơ đồ định địa chỉ trực tiếp

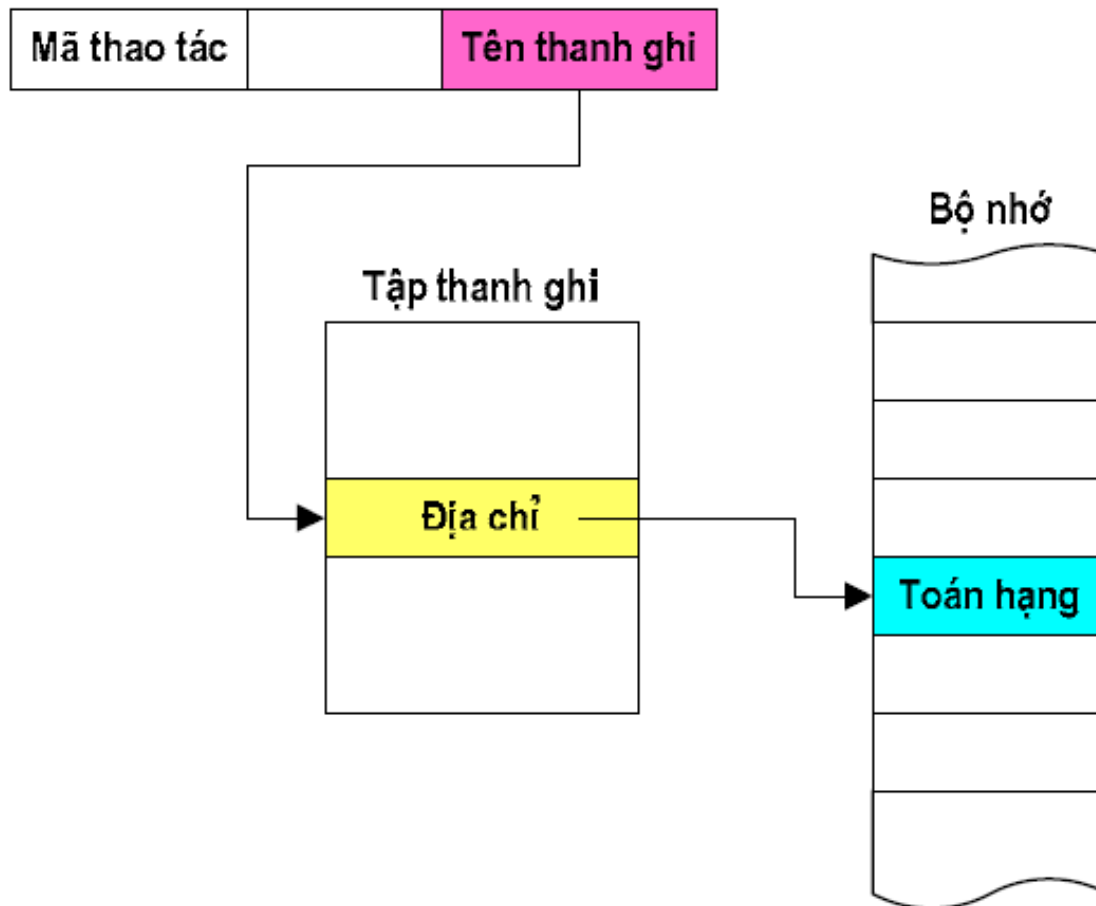




# Định địa chỉ gián tiếp qua thanh ghi

- Toán hạng là ngăn nhớ có địa chỉ nằm trong thanh ghi
- Trường địa chỉ cho biết tên thanh ghi đó
- Thanh ghi có thể là ngăn định
- Thanh ghi này được gọi là thanh ghi con trỏ
- Vùng nhớ có thể được tham chiếu là lớn ( $2^n$ ), (với  $n$  là độ dài của thanh ghi)

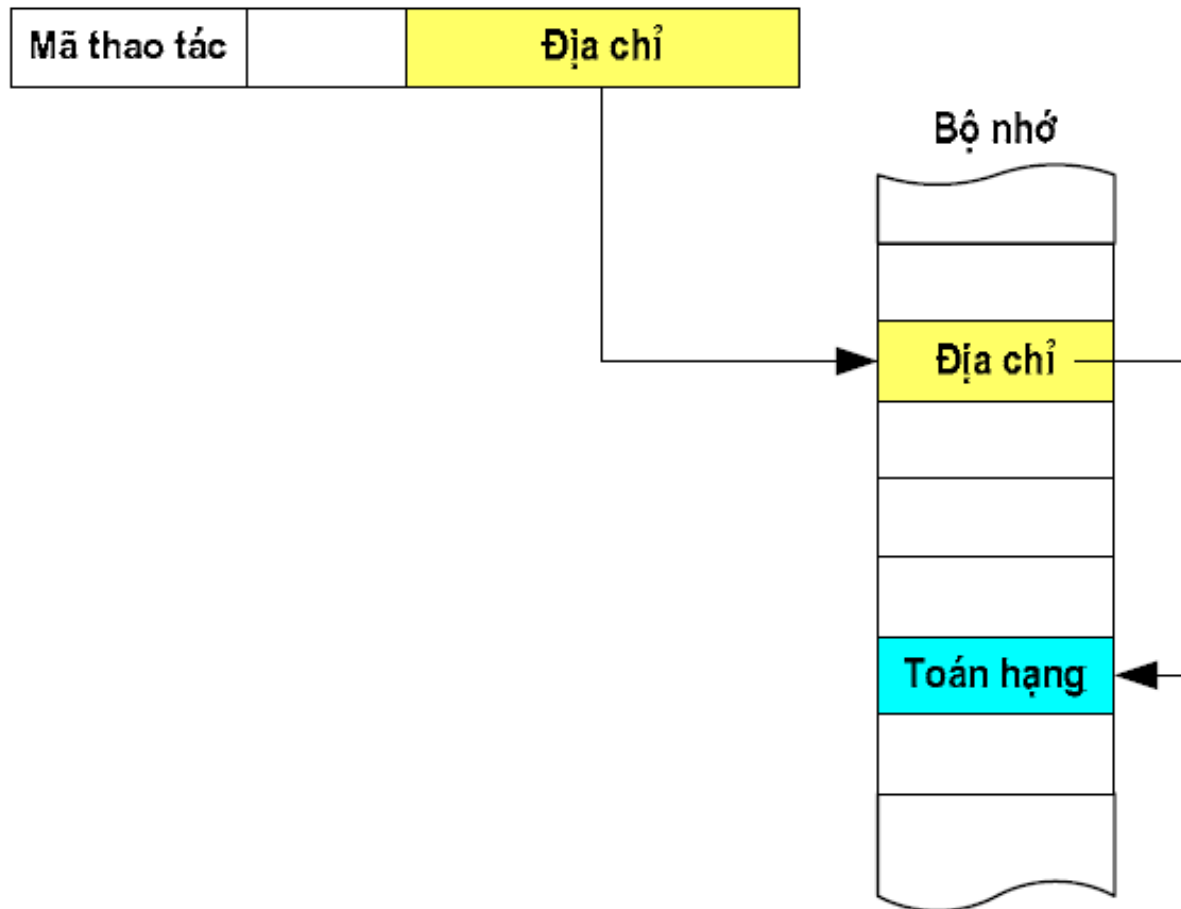
# Sơ đồ định địa chỉ gián tiếp qua thanh ghi



# Định địa chỉ gián tiếp qua ngăn nhớ

- Ngăn nhớ được trỏ bởi Trường địa chỉ của lệnh chứa địa chỉ của toán hạng
- Có thể gián tiếp nhiều lần
- Giống như khái niệm biến con trỏ và biến động trong lập trình
- CPU phải thực hiện tham chiếu bộ nhớ nhiều lần để tìm toán hạng → chậm
- Vùng nhớ có thể được tham chiếu là lớn

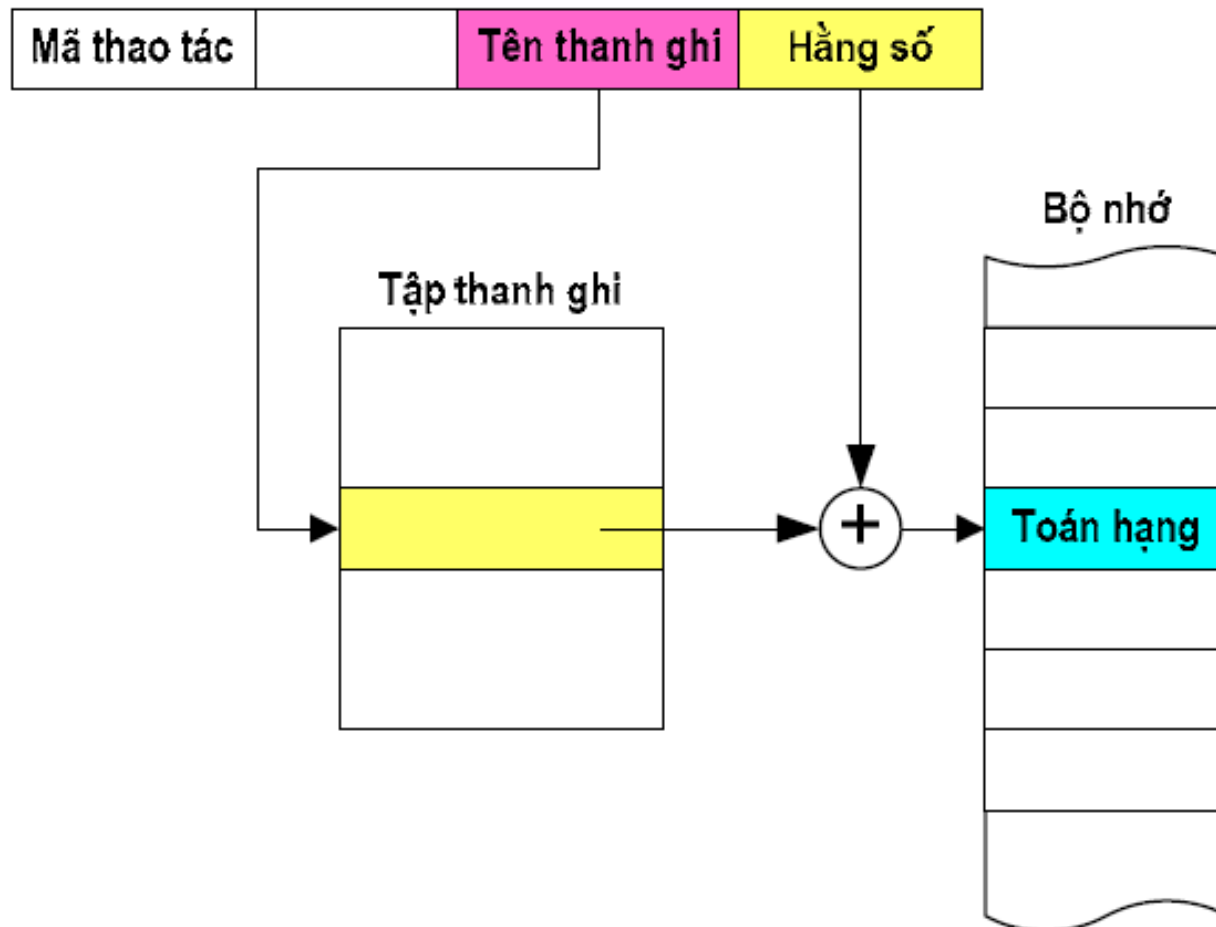
# Sơ đồ định địa chỉ gián tiếp qua ngăn nhớ



# Định địa chỉ dịch chuyển

- Để xác định toán hạng, Trường địa chỉ chứa hai thành phần:
  - Tên thanh ghi
  - Hằng số
- Địa chỉ của toán hạng = nội dung thanh ghi + hằng số
- Thanh ghi có thể được ngầm định

# Sơ đồ định địa chỉ dịch chuyển



# Các dạng của định địa chỉ dịch chuyển

- Địa chỉ hoá tương đối với PC
  - Thanh ghi là Bộ đếm chương trình PC
  - Toán hạng có địa chỉ cách ngăn nhớ được trở bởi PC một độ lệch xác định
- Định địa chỉ cơ sở
  - Thanh ghi chứa địa chỉ cơ sở
  - Hằng số là chỉ số
- Định địa chỉ chỉ số
  - Hằng số là địa chỉ cơ sở
  - Thanh ghi chứa chỉ số

# Chương 3: Bộ xử lý CPU

- 3.1 Cấu trúc cơ bản của CPU
- 3.2 Tập lệnh
- 3.3 Hoạt động của CPU
- 3.4 Kiến trúc của các bộ xử lý tiên tiến
- 3.5 Kiến trúc tập lệnh Intel x86



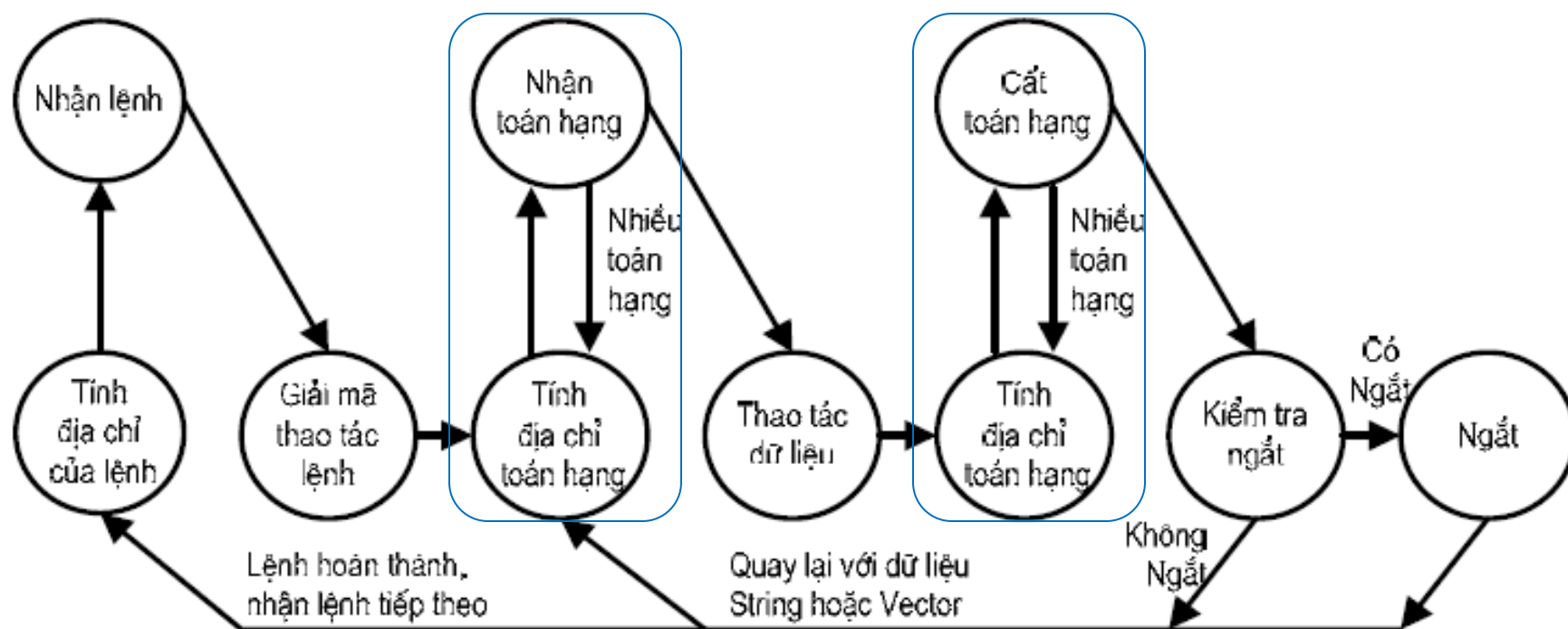


## 3.3. Hoạt động của CPU

### 1. Chu trình lệnh

- Nhận lệnh
- Giải mã lệnh
- Nhận toán hạng
- Thực hiện lệnh
- Cất toán hạng
- Ngắt

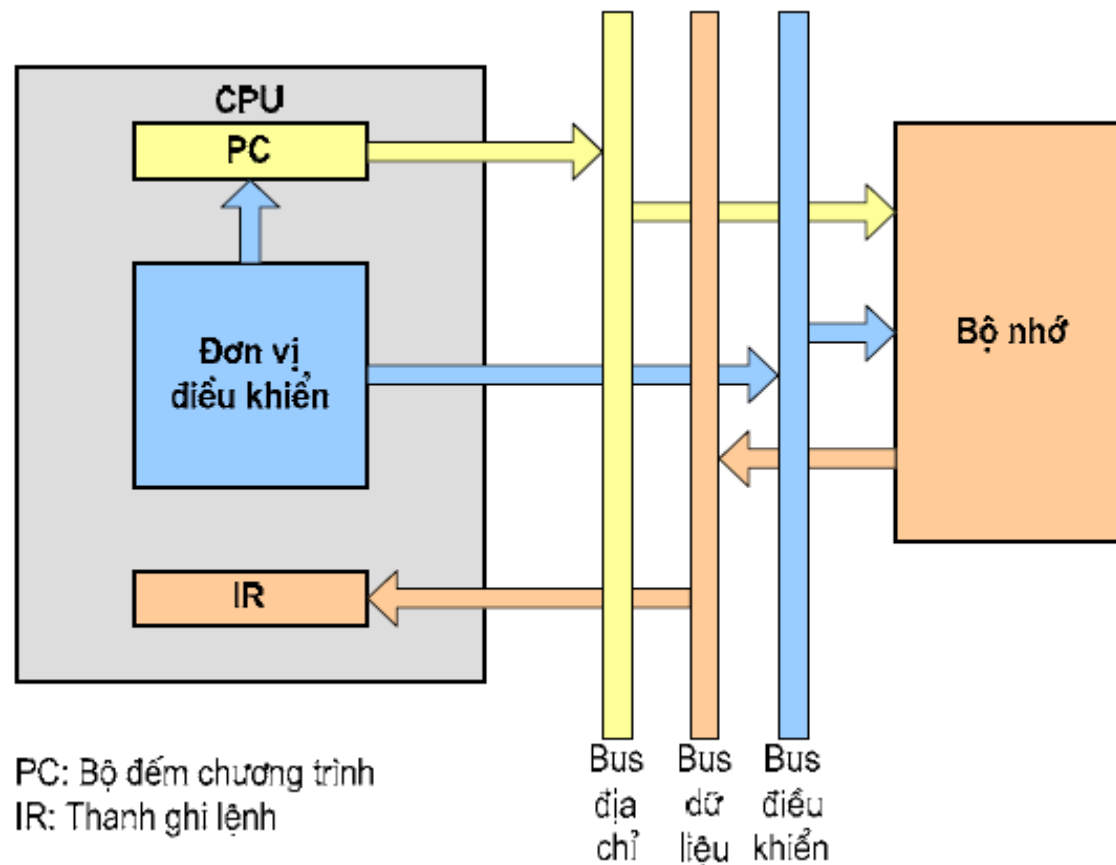
# Giải đồ trạng thái chu trình lệnh



# Nhận lệnh

- CPU đưa địa chỉ của lệnh cần nhận từ bộ đếm chương trình PC ra bus địa chỉ
- CPU phát tín hiệu điều khiển đọc bộ nhớ
- Lệnh từ bộ nhớ được đặt lên bus dữ liệu và được CPU copy vào thanh ghi lệnh IR
- CPU tăng nội dung PC để trở sang lệnh kế tiếp

# Sơ đồ mô tả quá trình nhận lệnh



# Giải mã lệnh

- Lệnh từ thanh ghi lệnh IR được đưa đến đơn vị điều khiển
- Đơn vị điều khiển tiến hành giải mã lệnh để xác định thao tác phải thực hiện
- Giải mã lệnh xảy ra bên trong CPU

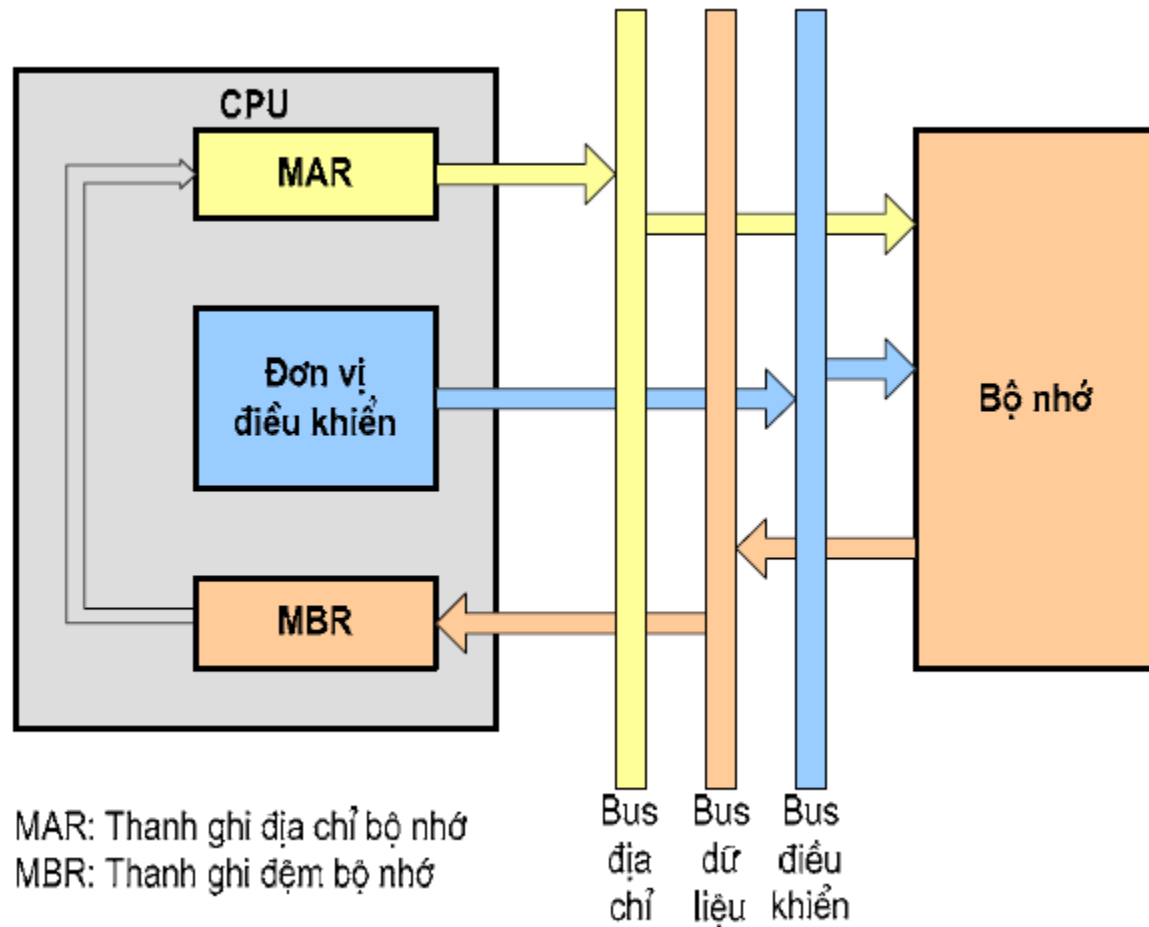
# Nhận dữ liệu

- CPU đưa địa chỉ của toán hạng ra bus địa chỉ
- CPU phát tín hiệu điều khiển đọc
- Toán hạng được đọc vào CPU
- Tương tự như nhận lệnh

# Nhận dữ liệu gián tiếp

- CPU đưa địa chỉ ra bus địa chỉ
- CPU phát tín hiệu điều khiển đọc
- Nội dung ngăn nhớ được đọc vào CPU, đó chính là địa chỉ của toán hạng
- Địa chỉ này được CPU phát ra bus địa chỉ để tìm ra toán hạng
- CPU phát tín hiệu điều khiển đọc
- Toán hạng được đọc vào CPU

# Sơ đồ tả nhận toán hạng gián tiếp





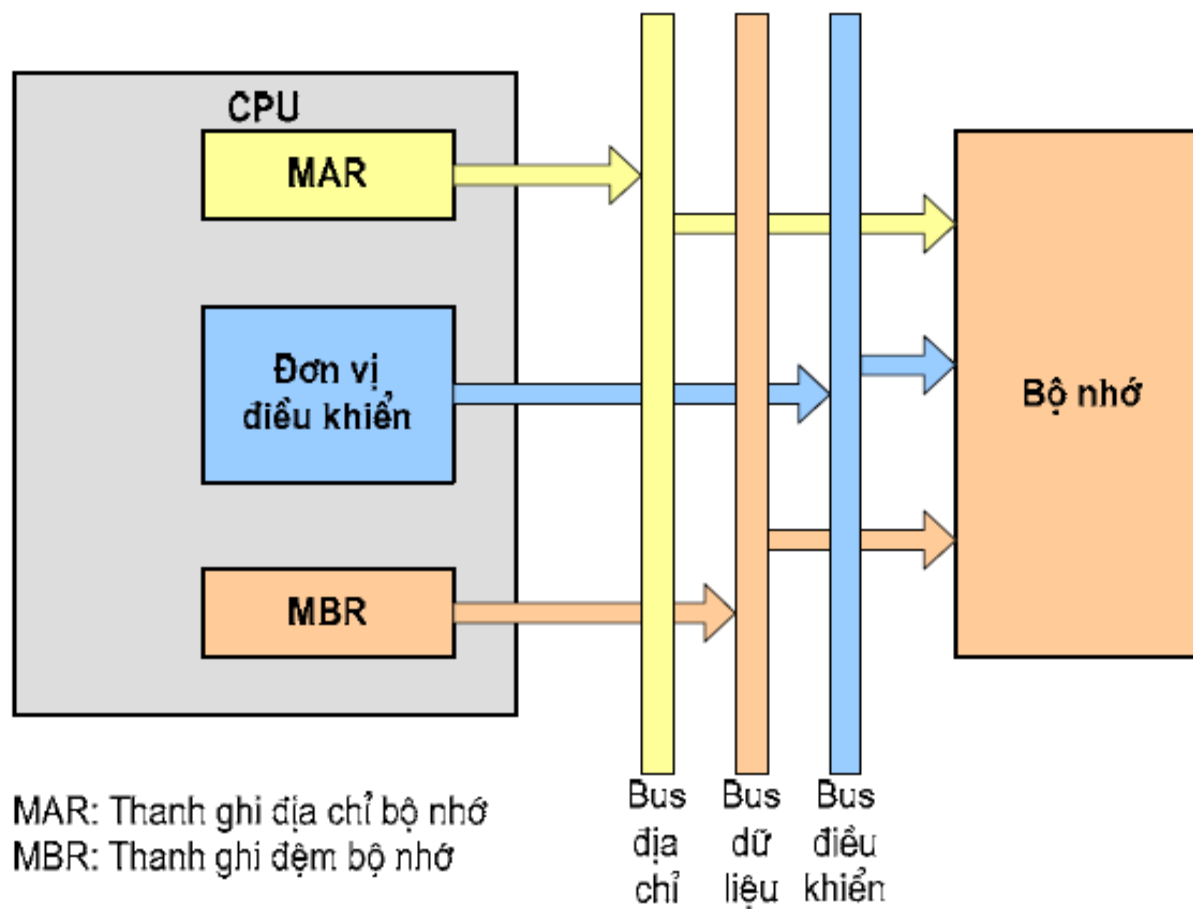
# Thực hiện lệnh

- Có nhiều dạng tùy thuộc vào lệnh
- Có thể là:
  - Đọc/Ghi bộ nhớ
  - Vào/Ra
  - Chuyển giữa các thanh ghi
  - Thao tác số học/logic
  - Chuyển điều khiển (rẽ nhánh)
  - ...

# Ghi toán hạng

- CPU đưa địa chỉ ra bus địa chỉ
- CPU đưa dữ liệu cần ghi ra bus dữ liệu
- CPU phát tín hiệu điều khiển ghi
- Dữ liệu trên bus dữ liệu được copy đến vị trí xác định

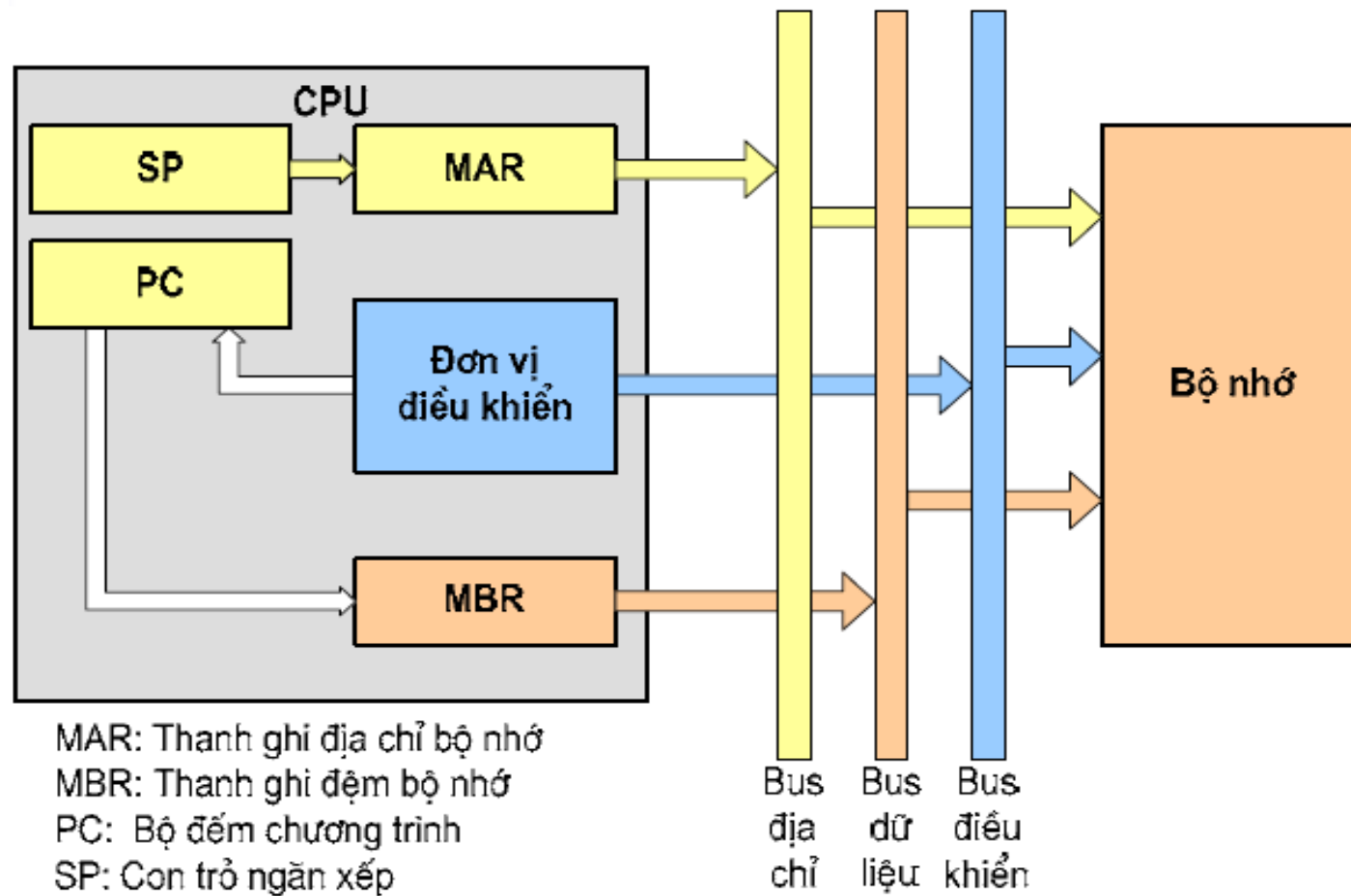
# Sơ đồ mô tả quá trình ghi toán hạng



# Ngắt

- Nội dung của bộ đếm chương trình PC (địa chỉ trở về sau khi ngắt) được đưa ra bus dữ liệu
- CPU đưa địa chỉ (thường được lấy từ con trỏ ngăn xếp SP) ra bus địa chỉ
- CPU phát tín hiệu điều khiển ghi bộ nhớ
- Địa chỉ trở về trên bus dữ liệu được ghi ra vị trí xác định (ở ngăn xếp)
- Địa chỉ lệnh đầu tiên của chương trình con điều khiển ngắt được nạp vào PC

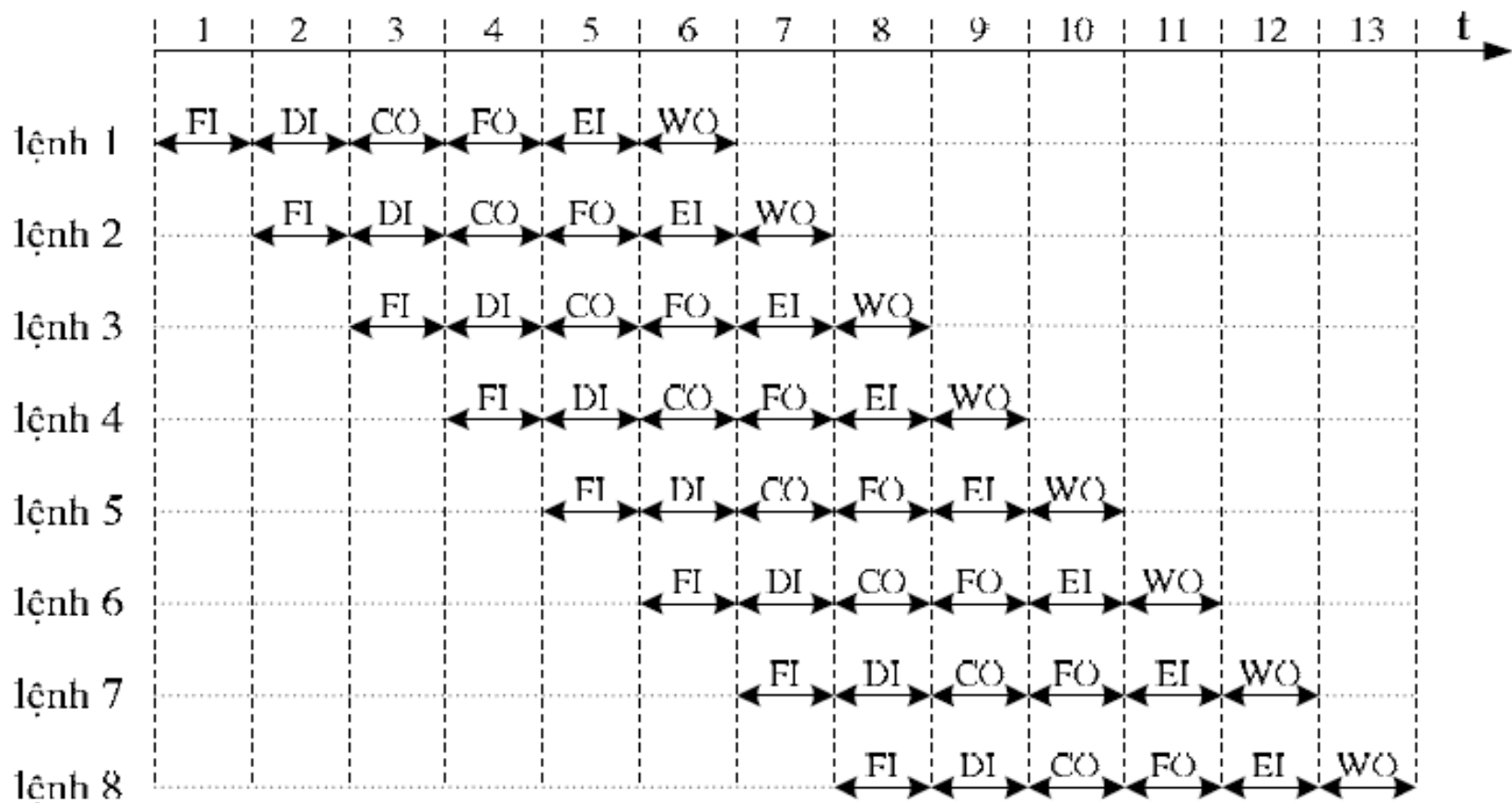
# Sơ đồ mô tả chu trình ngắt



## 2. Đường ống lệnh (Instruction Pipelining)

- Chia chu trình lệnh thành các công đoạn và cho phép thực hiện gối lên nhau (như dây chuyền lắp ráp)
- Chẳng hạn có 6 công đoạn:
  - Nhận lệnh (Fetch Instruction - FI)
  - Giải mã lệnh (Decode Instruction - DI)
  - Tính địa chỉ toán hạng (Calculate Operand Address-CO)
  - Nhận toán hạng (Fetch Operands - FO)
  - Thực hiện lệnh (Execute Instruction - EI)
  - Ghi toán hạng (Write Operands – WO)

# Biểu đồ thời gian của đường ống lệnh



# Các Hazard của đường ống lệnh

- Hazard cấu trúc: do nhiều công đoạn dùng chung một tài nguyên
- Hazard dữ liệu: lệnh sau sử dụng dữ liệu kết quả của lệnh trước
- Hazard điều khiển: do rẽ nhánh gây ra



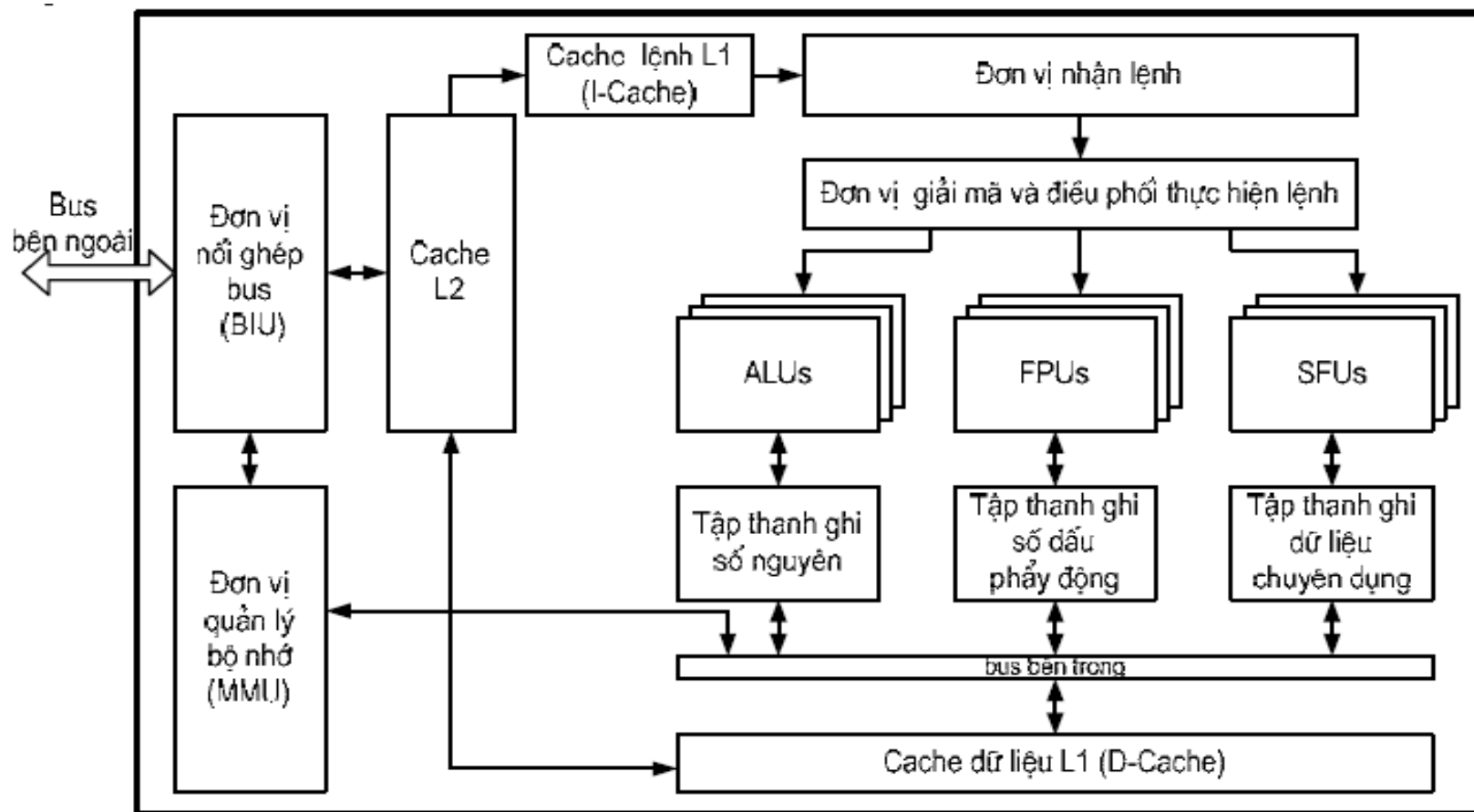
# Chương 3: Bộ xử lý CPU

- 3.1 Cấu trúc cơ bản của CPU
- 3.2 Tập lệnh
- 3.3 Hoạt động của CPU
- 3.4 Kiến trúc của các bộ xử lý tiên tiến
- 3.5 Kiến trúc tập lệnh Intel x86

## 3.4. Các kỹ thuật tiên tiến của bộ xử lý

- Cấu trúc chung của các bộ xử lý tiên tiến
- Các kiến trúc song song mức lệnh
- Kiến trúc RISC

# 1. Cấu trúc chung của các bộ xử lý tiên tiến



# Các đơn vị xử lý dữ liệu

- Các đơn vị số nguyên
- Các đơn vị số dấu phẩy động
- Các đơn vị chức năng đặc biệt
- Đơn vị xử lý dữ liệu âm thanh
- Đơn vị xử lý dữ liệu hình ảnh
- Đơn vị xử lý dữ liệu vector

# Bộ nhớ cache

- Được tích hợp trên chip vi xử lý
- Bao gồm hai mức cache:
  - Cache L1 gồm hai phần tách rời:
    - Cache lệnh
    - Cache dữ liệu

→ giải quyết xung đột khi nhận lệnh và dữ liệu
  - Cache L2: chung cho lệnh và dữ liệu

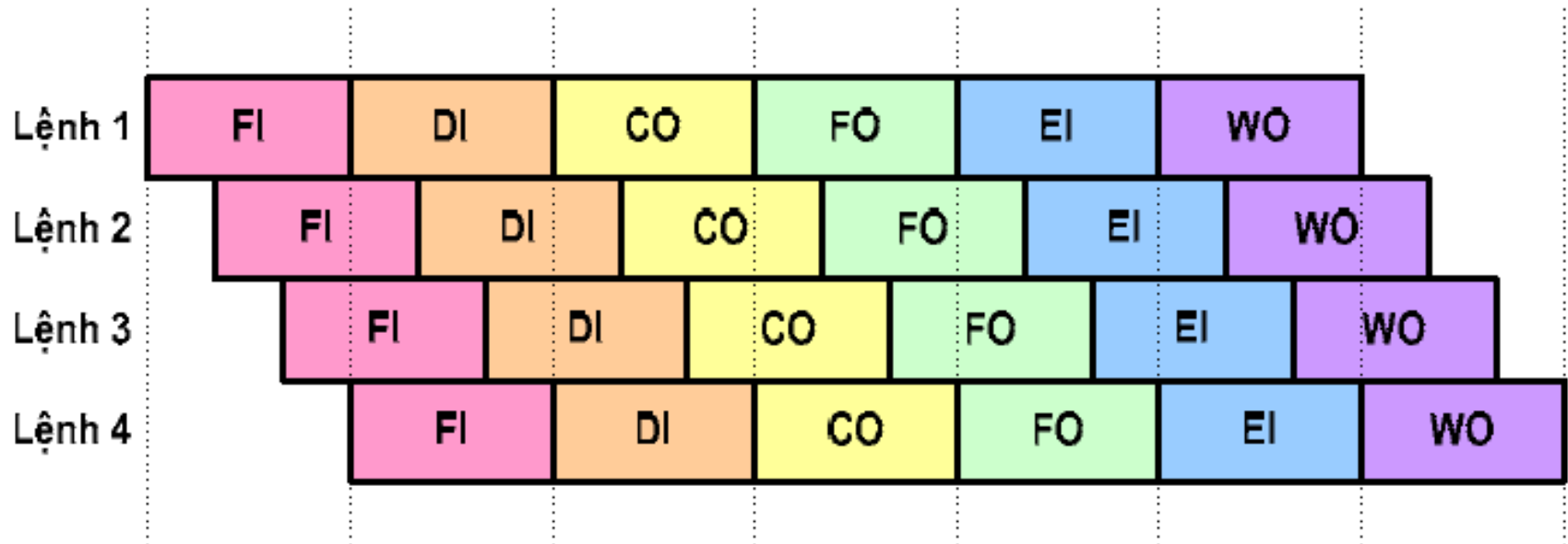
# Đơn vị quản lý bộ nhớ

- Chuyển đổi địa chỉ logic thành địa chỉ vật lý
- Cung cấp cơ chế phân trang/phân đoạn
- Cung cấp chế độ bảo vệ bộ nhớ

## 2. Các kiến trúc song song mức lệnh

- Siêu đường ống (Superpipeline & Hyperpipeline)
- Siêu vô hướng (Superscalar)
- VLIW (Very Long Instruction Word)

# Superpipeline





# Superscalar

Lệnh 1	FI	DI	CO	FO	EI	WO		
Lệnh 2	FI	DI	CO	FO	EI	WO		
Lệnh 3		FI	DI	CO	FO	EI	WO	
Lệnh 4		FI	DI	CO	FO	EI	WO	
Lệnh 5			FI	DI	CO	FO	EI	WO
Lệnh 6			FI	DI	CO	FO	EI	WO

# VLIW (Very Long Instruction Word)

Từ lệnh thông thường



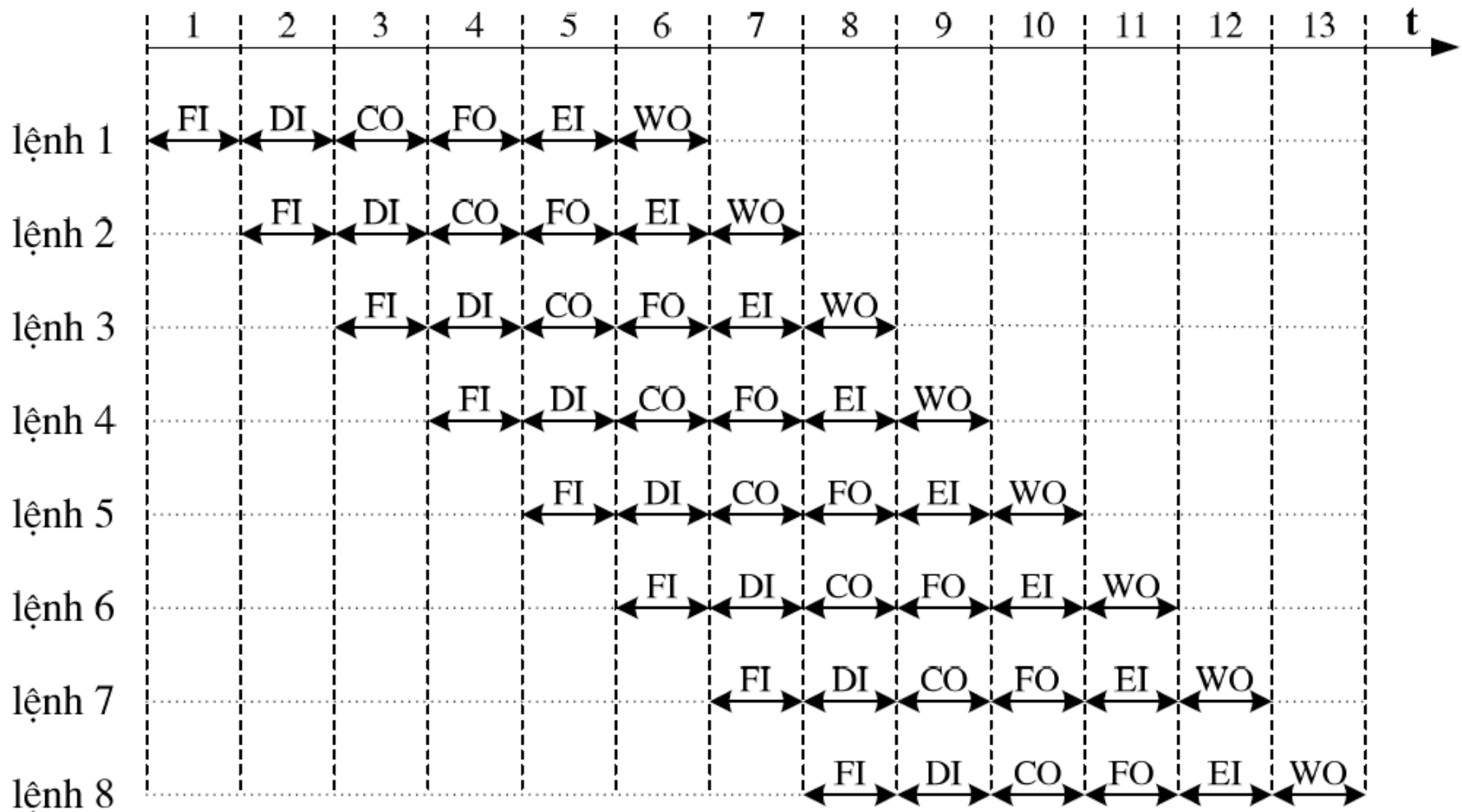
Từ lệnh dài




# Kỹ thuật đường ống lệnh và song song mức lệnh

- Kỹ thuật đường ống lệnh (Instruction Pipelining): Chia chu trình lệnh thành các công đoạn và cho phép thực hiện gộp lên nhau (như dây chuyền lắp ráp)
- Chẳng hạn có 6 công đoạn:
  - Nhận lệnh (Fetch Instruction - FI)
  - Giải mã lệnh (Decode Instruction - DI)
  - Tính địa chỉ toán hạng (Calculate Operand Address - CO)
  - Nhận toán hạng (Fetch Operands - FO)
  - Thực hiện lệnh (Execute Instruction - EI)
  - Ghi toán hạng (Write Operands – WO)

# Biểu đồ thời gian của đường ống lệnh





Các Hazard (trở ngại) của đường ống lệnh

- Hazard: Tình huống ngăn cản bắt đầu của lệnh tiếp theo ở chu kỳ tiếp theo.
  - Hazard cấu trúc: do tài nguyên được yêu cầu đang bận
  - Hazard dữ liệu: cần phải đợi để lệnh trước hoàn thành việc đọc/ghi dữ liệu
  - Hazard điều khiển: do rẽ nhánh gây ra

# Hazard về cấu trúc

## ■ Khắc phục:

- nhân tài nguyên để tránh xung đột
- Làm trễ

## ■ Ví dụ:

- Bus dữ liệu: truyền lệnh và dữ liệu
- Bus lệnh riêng, bus dữ liệu riêng (cache lệnh và cache dữ liệu)

# Chương 3: Bộ xử lý CPU

- 3.1 Cấu trúc cơ bản của CPU
- 3.2 Tập lệnh
- 3.3 Hoạt động của CPU
- 3.4 Kiến trúc của các bộ xử lý tiên tiến
- 3.5 Kiến trúc tập lệnh Intel x86

## 3.5. Kiến trúc Intel

- Sự tiến hóa của các bộ xử lý Intel
  - 8080 (1974): 8-bit microprocessor
    - Accumulator, plus 3 index-register pairs
  - 8086 (1978): 16-bit extension to 8080
    - Complex instruction set (CISC)
  - 8087 (1980): floating-point coprocessor
    - Adds FP instructions and register stack
  - 80286 (1982): 24-bit addresses, MMU
    - Segmented memory mapping and protection
  - 80386 (1985): 32-bit extension (now IA-32)
    - Additional addressing modes and operations
    - Paged memory mapping as well as segments



## 3.5. Kiến trúc Intel

- i486 (1989): pipelined, on-chip caches and FPU
  - Compatible competitors: AMD, Cyrix, ...
- Pentium (1993): superscalar, 64-bit datapath
  - Later versions added MMX (Multi-Media eXtension) instructions
  - The infamous FDIV bug
- Pentium Pro (1995), Pentium II (1997)
  - New microarchitecture
- Pentium III (1999)
  - Added SSE (Streaming SIMD Extensions) and associated registers
- Pentium 4 (2001)
  - New microarchitecture
  - Added SSE2 instructions
- Intel Core (2006)
  - Added SSE4 instructions, virtual machine support


# Các phương pháp định địa chỉ cơ bản

## ■ Hai toán hạng của lệnh

Source/dest operand	Second source operand
Register	Register
Register	Immediate
Register	Memory
Memory	Register
Memory	Immediate

## ■ Các phương pháp định địa chỉ

- Address in register
- $\text{Address} = R_{\text{base}} + \text{displacement}$
- $\text{Address} = R_{\text{base}} + 2^{\text{scale}} \times R_{\text{index}}$  (scale = 0, 1, 2, or 3)
- $\text{Address} = R_{\text{base}} + 2^{\text{scale}} \times R_{\text{index}} + \text{displacement}$



# 1. Kiến trúc 16-bit (IA-16)

- Các thanh ghi bên trong: 16-bit
- Xử lý các phép toán số nguyên với 16-bit
- Quản lý bộ nhớ theo đoạn 64KBytes
- Mở đầu cho dòng máy tính IBM-PC

## 2. Kiến trúc 32-bit (IA-32)

- Các thanh ghi bên trong: 32-bit
- Xử lý các phép toán số nguyên với 32-bit
- Có ba chế độ làm việc:
  - Chế độ 8086 thực (Real 8086 mode): làm việc như một bộ xử lý 8086
  - Chế độ 8086 ảo (Virtual 8086 mode): làm việc như nhiều bộ xử lý 8086 (đa nhiệm 16-bit)
  - Chế độ bảo vệ (Protected mode)
    - đa nhiệm 32-bit
    - quản lý bộ nhớ ảo
- Xử lý các phép toán số dấu phẩy động (từ 80486)

### 3. Kiến trúc 64-bit (IA-64)

- Các thanh ghi bên trong: 64-bit
- Xử lý các phép toán số nguyên với 64-bit
- Xử lý các phép toán số dấu phẩy động
- Không tương thích phần cứng với các bộ xử lý trước đó
- Tương thích phần mềm bằng cách giả lập môi trường

# CPU qua các thời kỳ





HẾT Chương 3