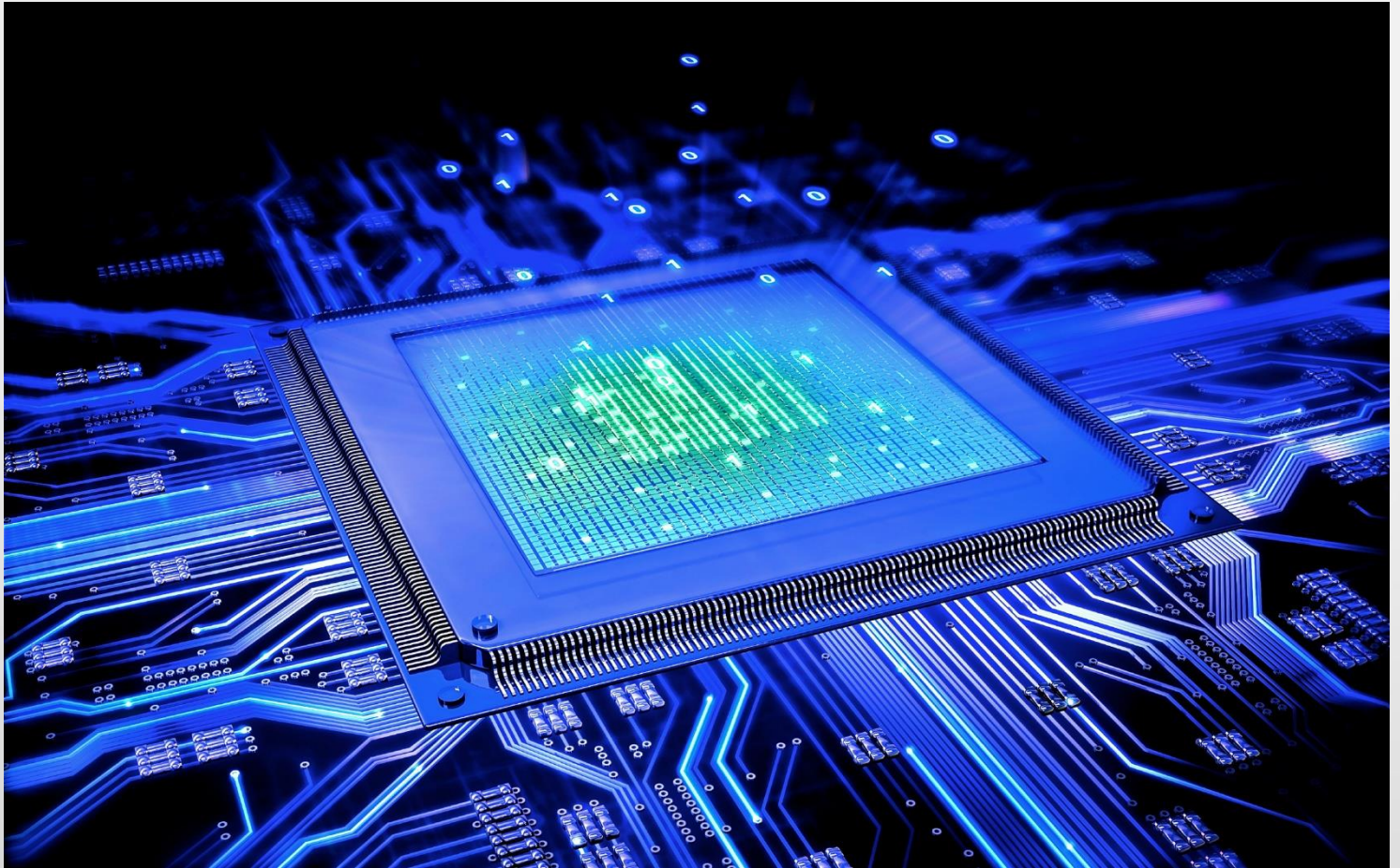


# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

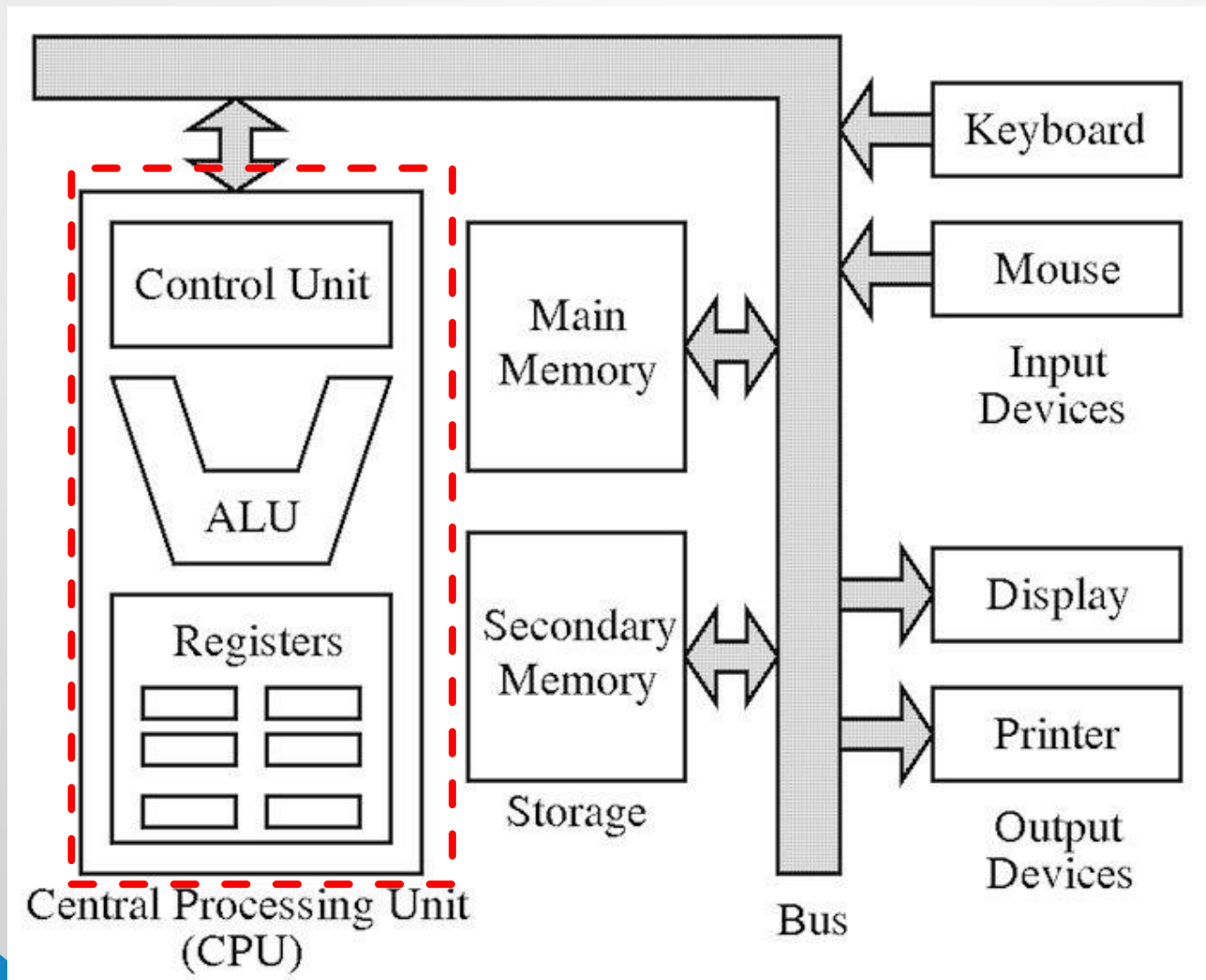
- Cấu trúc cơ bản của CPU
- Bộ xử lý đa lõi
- Kiến trúc tập lệnh của CPU



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

Sơ đồ cấu trúc cơ bản của CPU

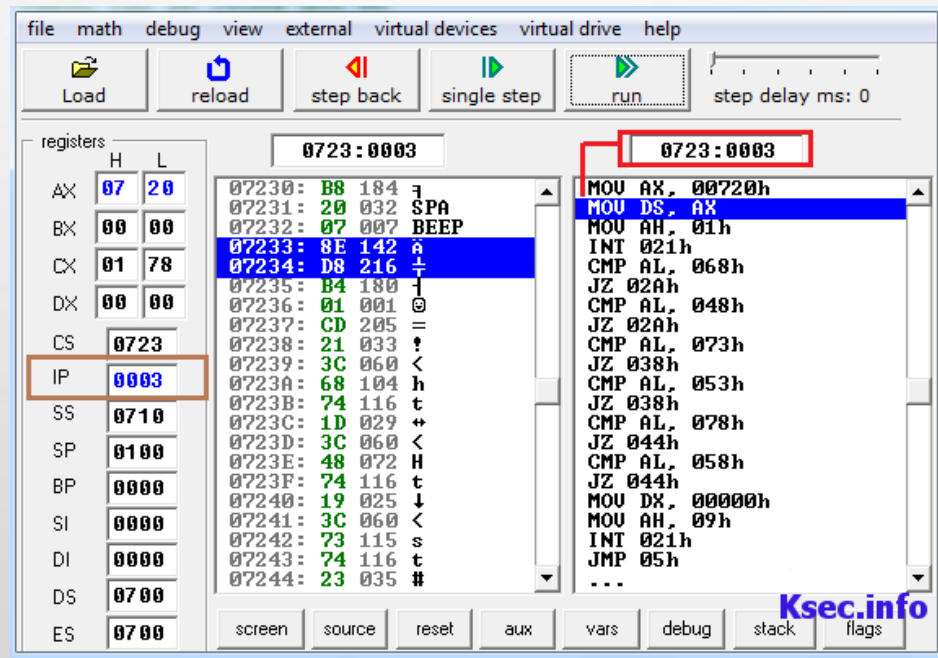


# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Là bộ nhớ có kích thước nhỏ nằm ngay trên CPU nên nội dung của nó được truy cập nhanh hơn so với các nơi lưu trữ khác có sẵn trong máy tính.





# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

*Phân loại theo khả năng lập trình:*

- ✓ Các thanh ghi lập trình được.
- ✓ Các thanh ghi không lập trình được.

*Phân loại theo chức năng:*

- ✓ Thanh ghi địa chỉ
- ✓ Thanh ghi dữ liệu
- ✓ Thanh ghi trạng thái

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

**Một số thanh ghi điển hình:**

- ✓ 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

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

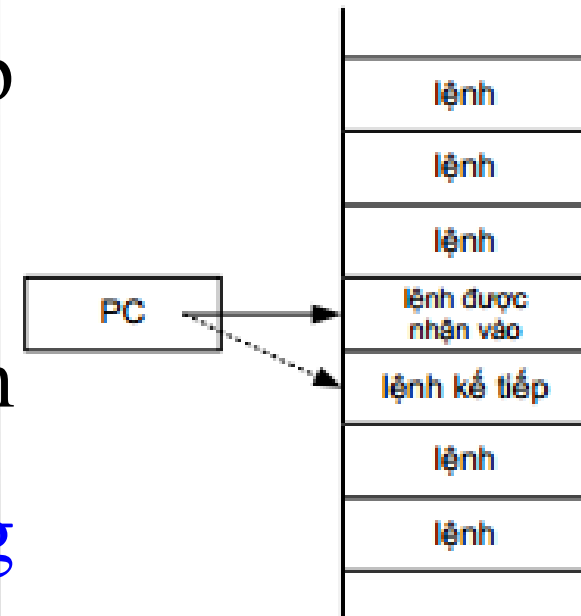
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

#### **Bộ đếm chương trình PC**

- ✓ 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 trỏ sang lệnh kế tiếp



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

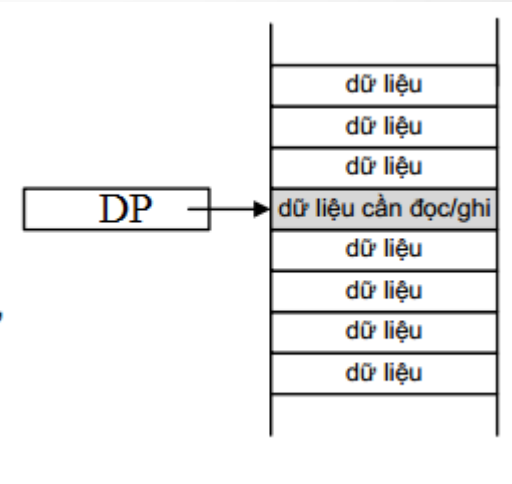
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

### Thanh ghi con trỏ dữ liệu

- ✓ Con trỏ dữ liệu DP (Data Pointer).
- ✓ Chứa địa chỉ của ngăn nhớ chứa dữ liệu mà CPU muốn truy nhập (đọc/ghi).





# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

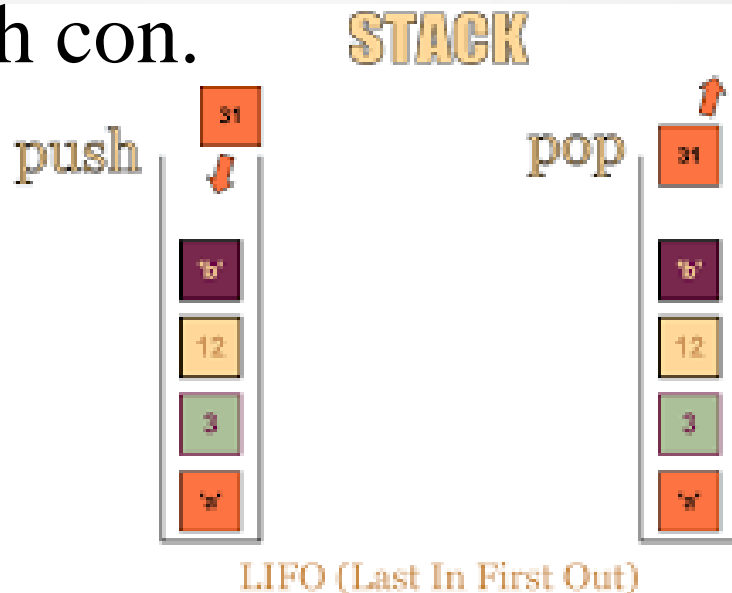
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

### Con trỏ ngăn xếp SP (Stack Pointer)

- ✓ Ngăn xếp là vùng nhớ có cấu trúc LIFO.
- ✓ Ngăn xếp thường được dùng để phục vụ cho chương trình con.



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

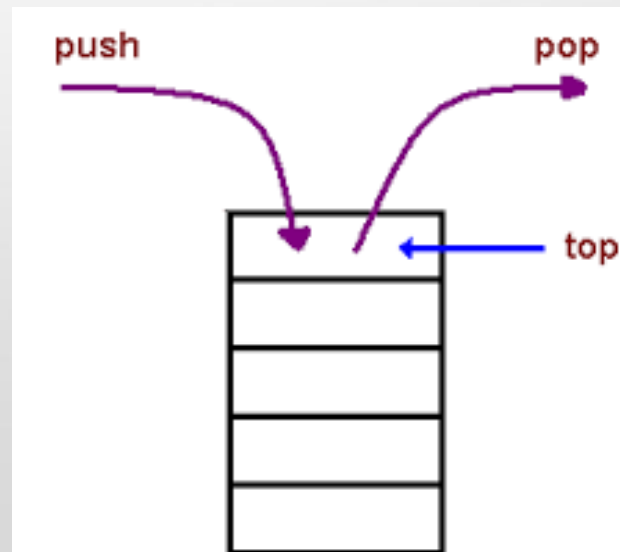
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

### Con trỏ ngăn xếp SP (Stack Pointer)

- ✓ Đáy ngăn xếp là một ngăn nhớ xác định, đỉnh ngăn xếp là thông tin nằm ở vị trí trên cùng của ngăn xếp, đỉnh của ngăn xếp có thể bị thay đổi.



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

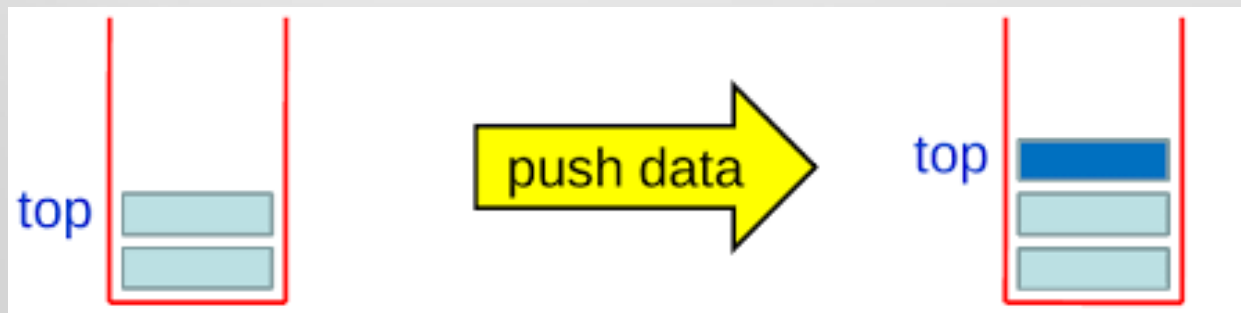
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

#### Con trỏ ngăn xếp SP (Stack Pointer)

- ✓ SP chứa địa chỉ của ngăn nhớ đỉnh ngăn xếp.
- ✓ Khi cất một thông tin vào ngăn xếp:
  - Nội dung của SP giảm
  - Thông tin được cất vào ngăn nhớ được trỏ bởi SP.



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

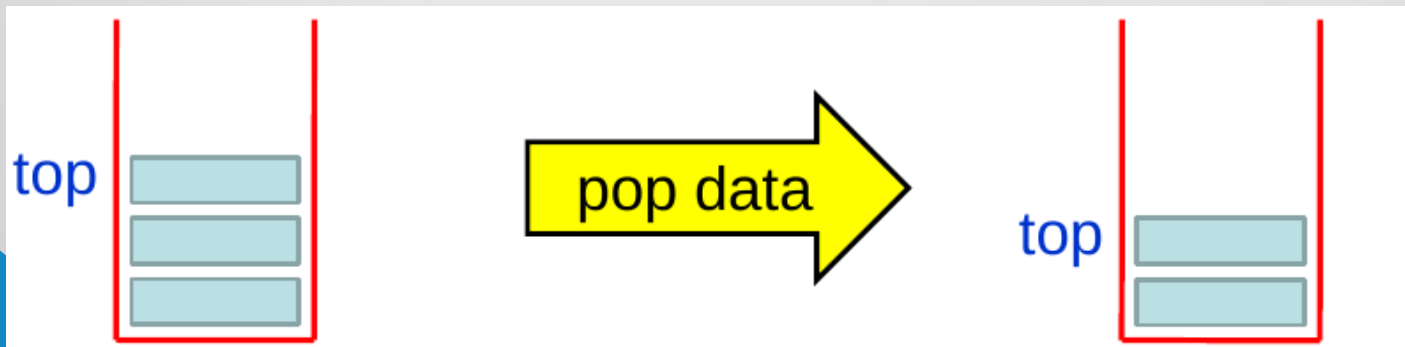
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

#### **Con trỏ ngăn xếp SP (Stack Pointer)**

- ✓ Khi lấy một thông tin ra khỏi ngăn xếp:
  - Thông tin được đọc từ ngăn nhớ được trỏ bởi SP.
  - Nội dung của SP tăng
- ✓ Khi ngăn xếp rỗng, SP trở vào đáy



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

### Thanh ghi cơ sở và Thanh ghi chỉ số

✓ Để truy nhập một ngăn nhớ có thể sử dụng hai tham số:

- Địa chỉ cơ sở (Base address)
- Phần dịch chuyển địa chỉ (offset)
- Địa chỉ của ngăn nhớ cần truy nhập = địa chỉ cơ sở + offset



✓ Có thể sử dụng các thanh ghi để quản lý các tham số này: thanh ghi cơ sở & thanh ghi chỉ số.

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

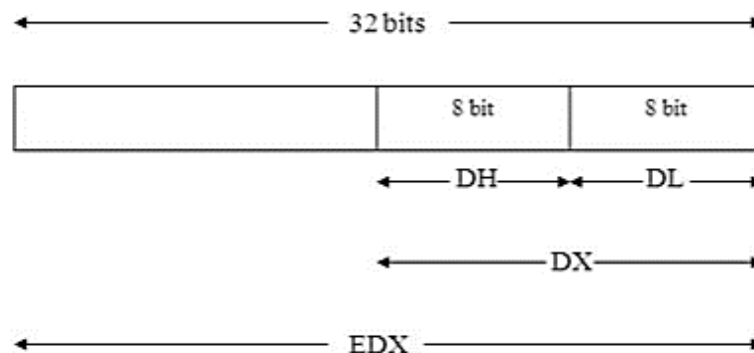
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

#### 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, 63 bit
- ✓ Các thanh ghi dấu phẩy động.





# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

**Một số thanh ghi điển hình:**

### **Thanh ghi trạng thái**

- ✓ 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.

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

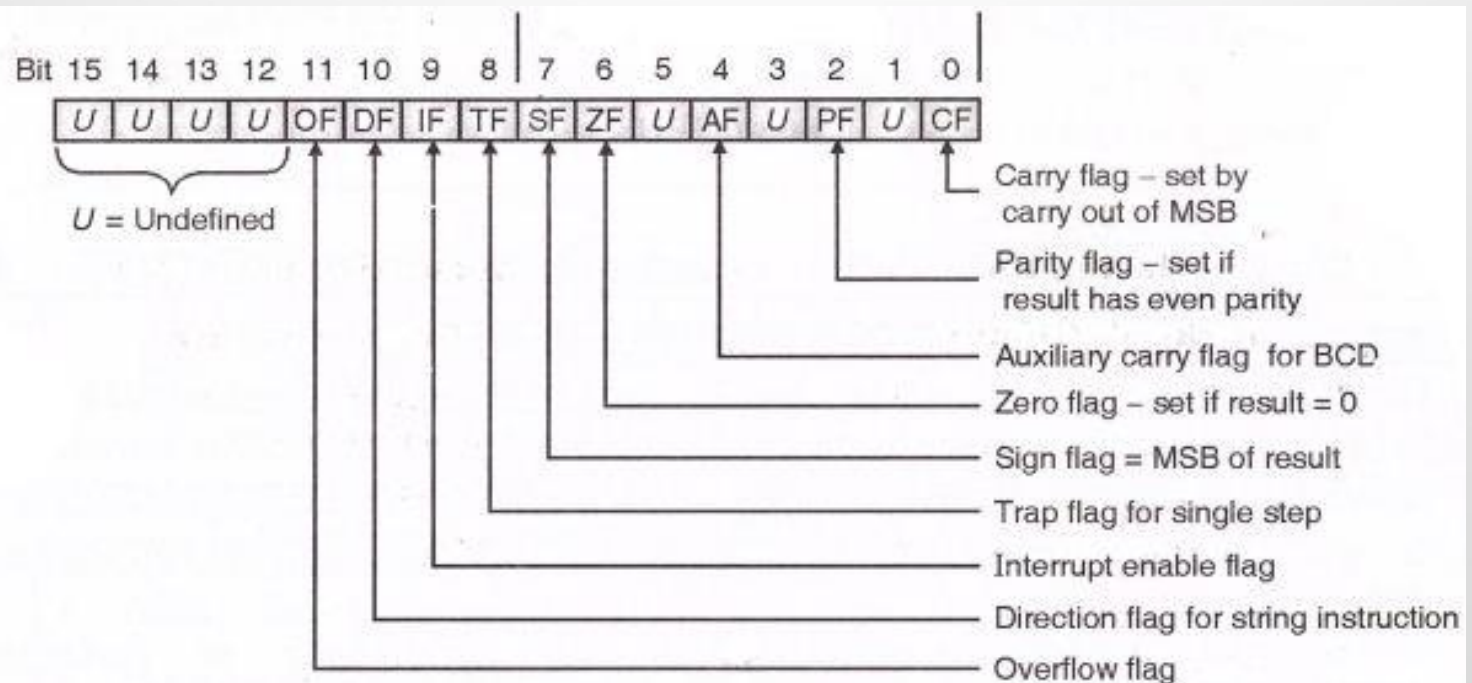
## 3.1. Cấu trúc cơ bản của CPU

### *a. Tập các thanh ghi*

Một số thanh ghi điển hình:

### Thanh ghi trạng thái

✓ Còn được gọi là thanh ghi cờ (Flag Register).



8086 flag register format

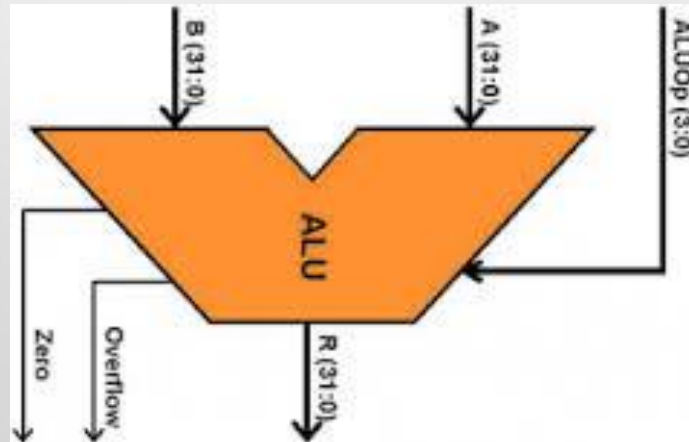
# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

### ***b. Đơn vị số học và logic***

Có chức năng là thực hiện các phép toán số học và phép toán logic:

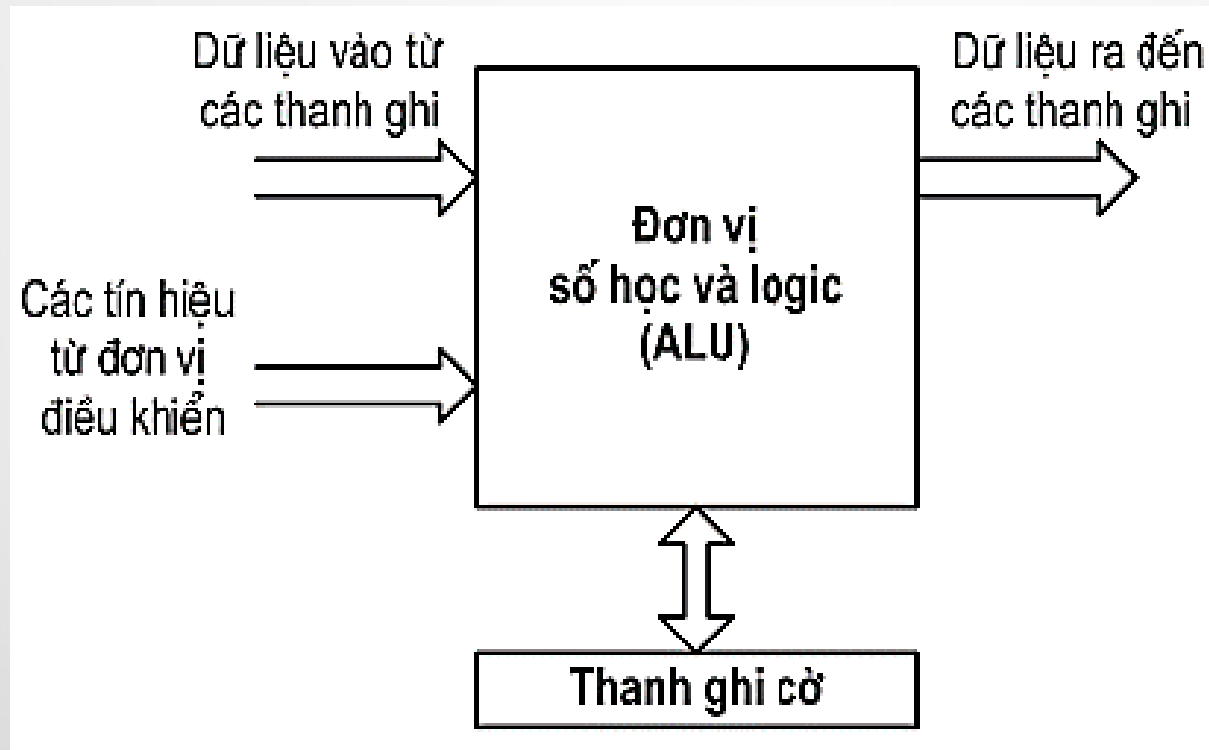
- ✓ Số học: cộng, trừ, nhân, chia, tăng, giảm, đảo dấu
- ✓ Logic: AND, OR, XOR, NOT, phép dịch bit



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

### *b. Đơn vị số học và logic*

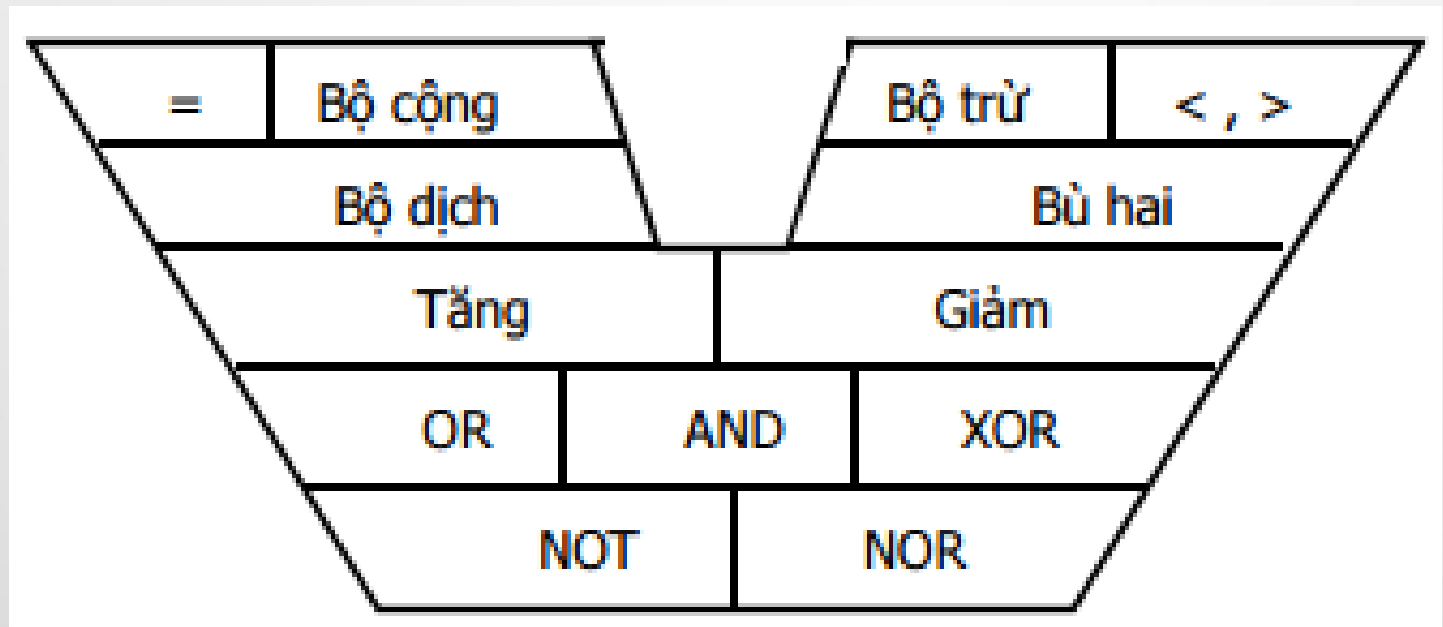


Mô hình kết nối ALU

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

### *b. Đơn vị số học và logic*



Các khối bên trong ALU

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

## 3.1. Cấu trúc cơ bản của CPU

### *c. Đơn vị điều khiển*

Có các chức năng:

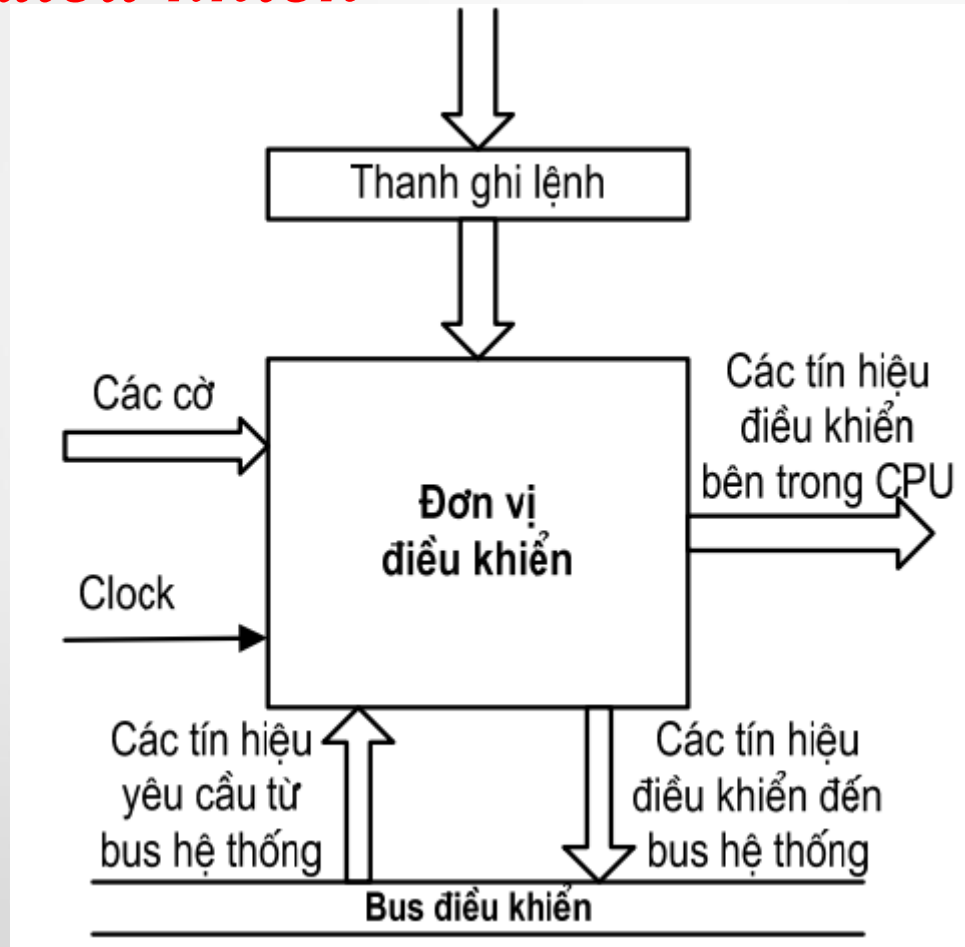
- Điều khiển nhận lệnh từ bộ nhớ đưa vào thanh ghi lệnh.
- Tăng nội dung của PC để trở sang lệnh kế tiếp.
- Giải mã lệnh đã được nhận để xác định thao tác mà lệnh yêu cầu.
- Phát ra các tín hiệu điều khiển thực hiện lệnh.
- Nhận các tín hiệu yêu cầu từ bus hệ thống và đáp ứng với các yêu cầu đó.



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

### *c. Đơn vị điều khiển*



Mô hình kết nối đơn vị điều khiển

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

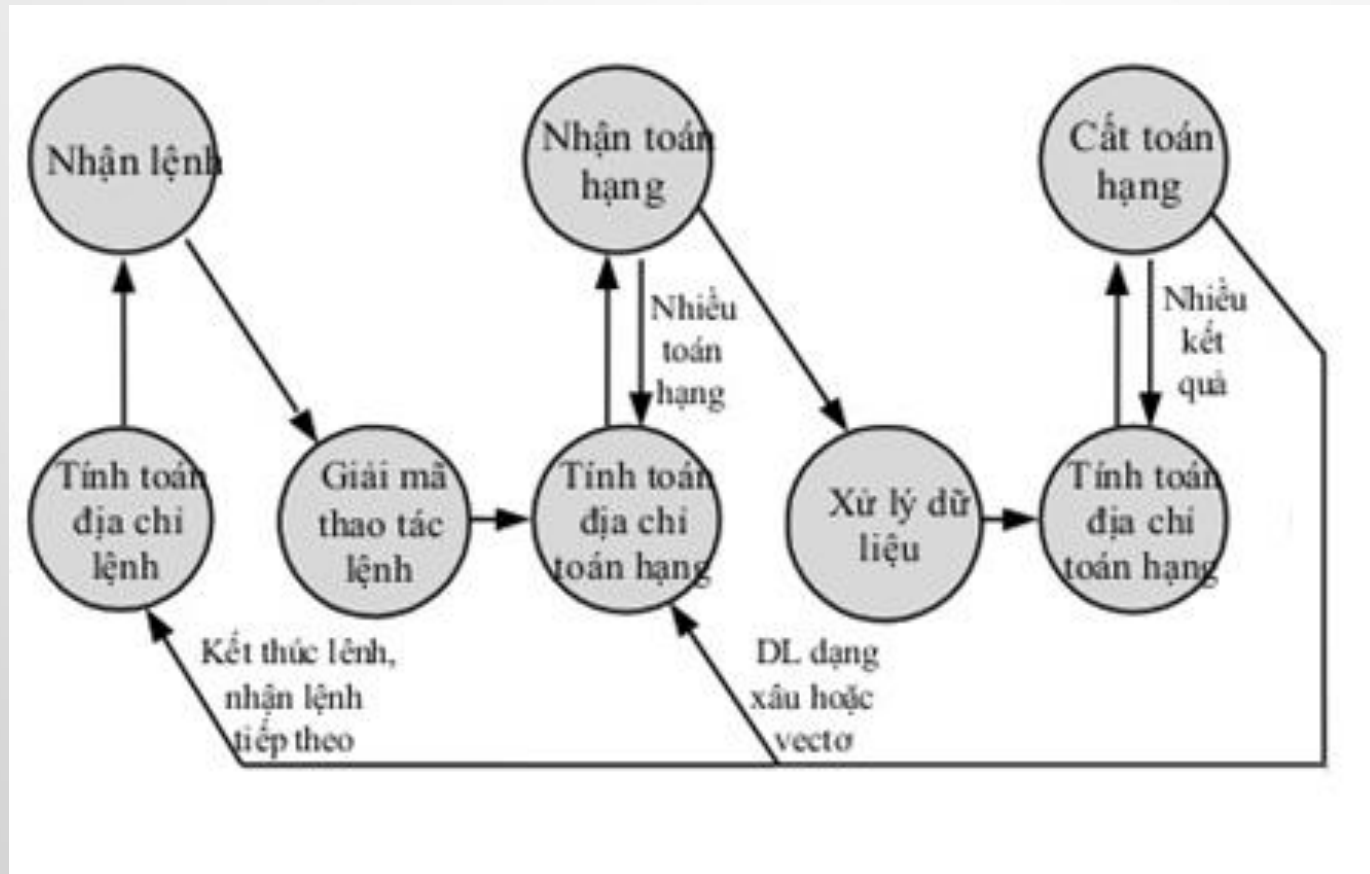
Nhiệm vụ của CPU:

- ✓ Nhận lệnh (Fetch Instruction): CPU đọc lệnh từ bộ nhớ chính.
- ✓ Nhận dữ liệu (Fetch Data): Nhận dữ liệu từ bộ nhớ hoặc các cổng vào- ra. (*Nhận toán hạng*)
- ✓ Giải mã lệnh (Decode Instruction): xác định thao tác mà lệnh yêu cầu.
- ✓ 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. (*Cất toán hạng*)

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

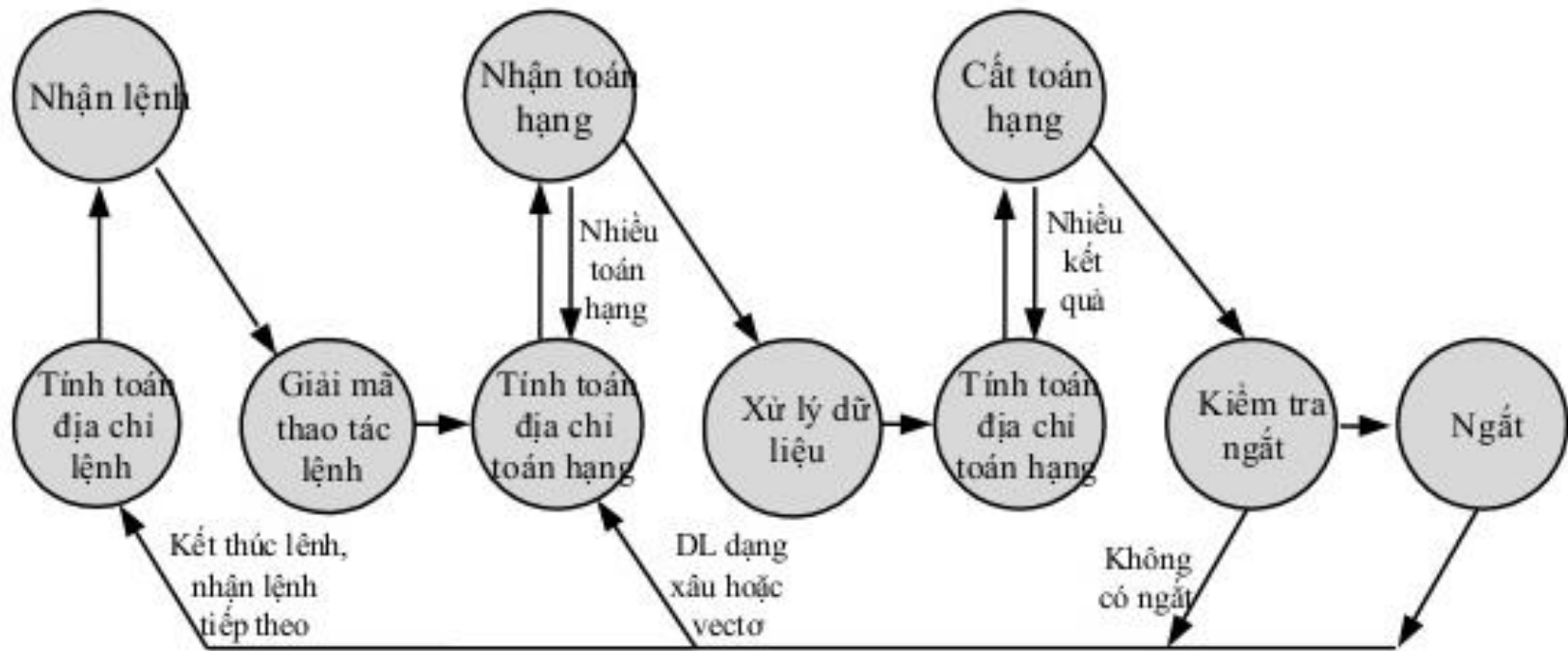
*Sơ đồ trạng thái chu kỳ lệnh:*



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.1. Cấu trúc cơ bản của CPU

*Sơ đồ trạng thái chu kỳ lệnh đầy đủ*



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

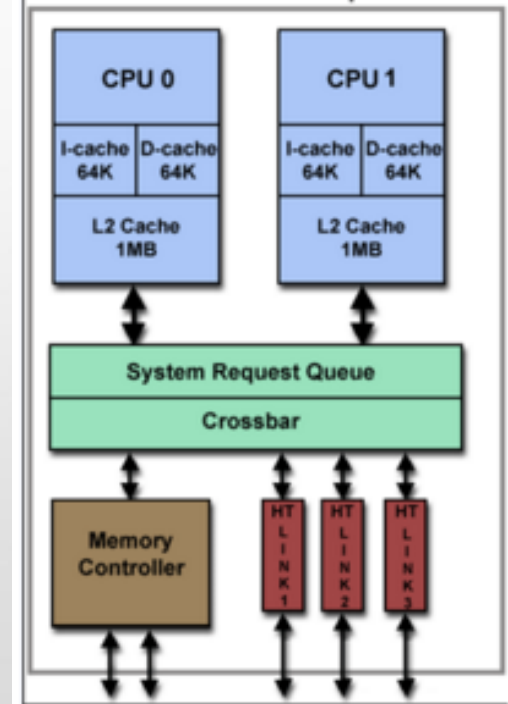
## 3.2. Bộ xử lý đa lõi

CPU đa nhân, CPU đa lõi (multi-core) là bộ vi xử lý trung tâm (Central Processing Unit) có nhiều đơn vị vi xử lý được tích hợp trên cùng một CPU vật lý duy nhất, như là sự ghép nối nhiều CPU thông thường thành một CPU duy nhất.

Hiệu năng của máy tính tăng lên do có tính chất đa nhân.

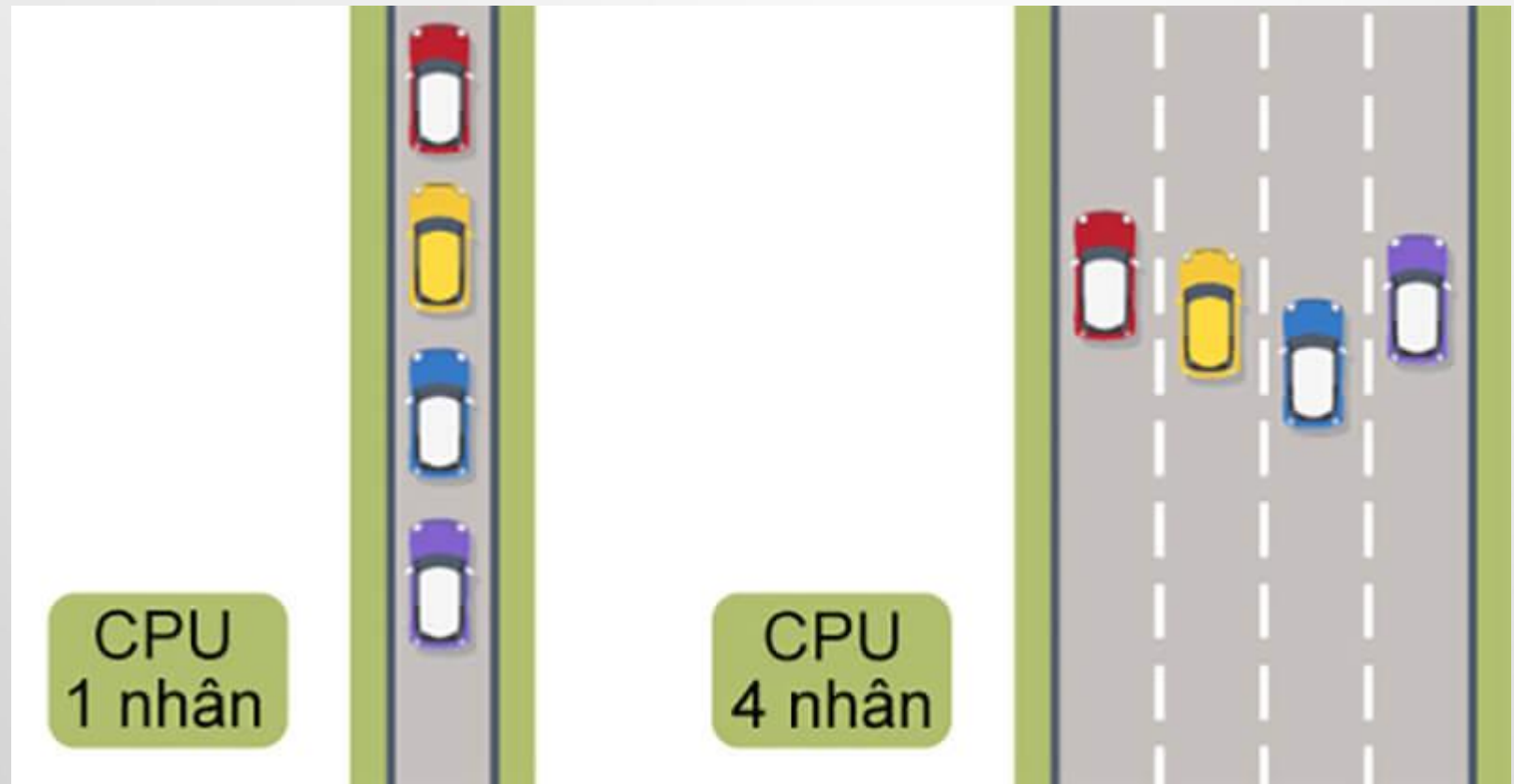


Dual-core AMD Opteron



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

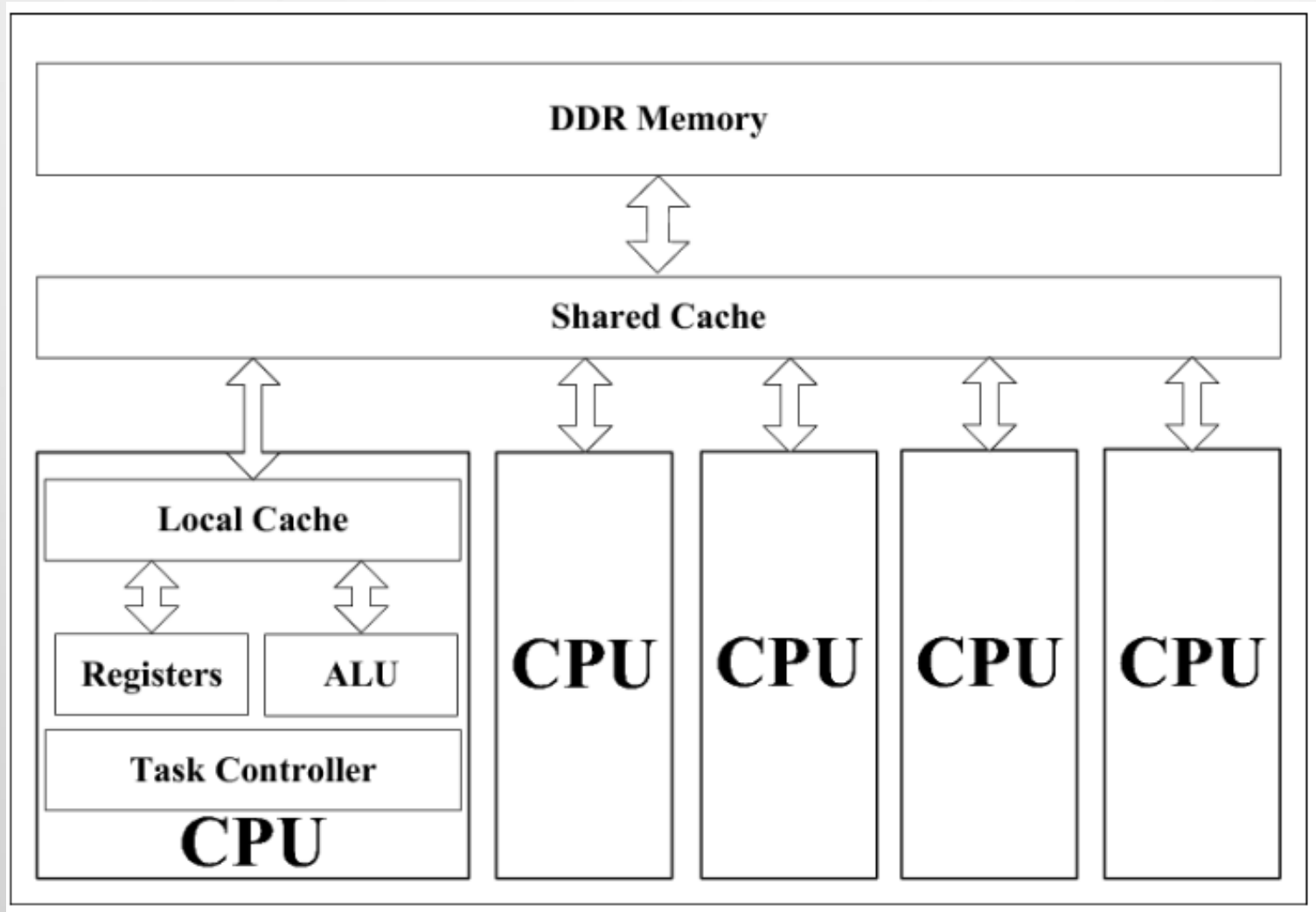
## 3.2. Bộ xử lý đa lõi





# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

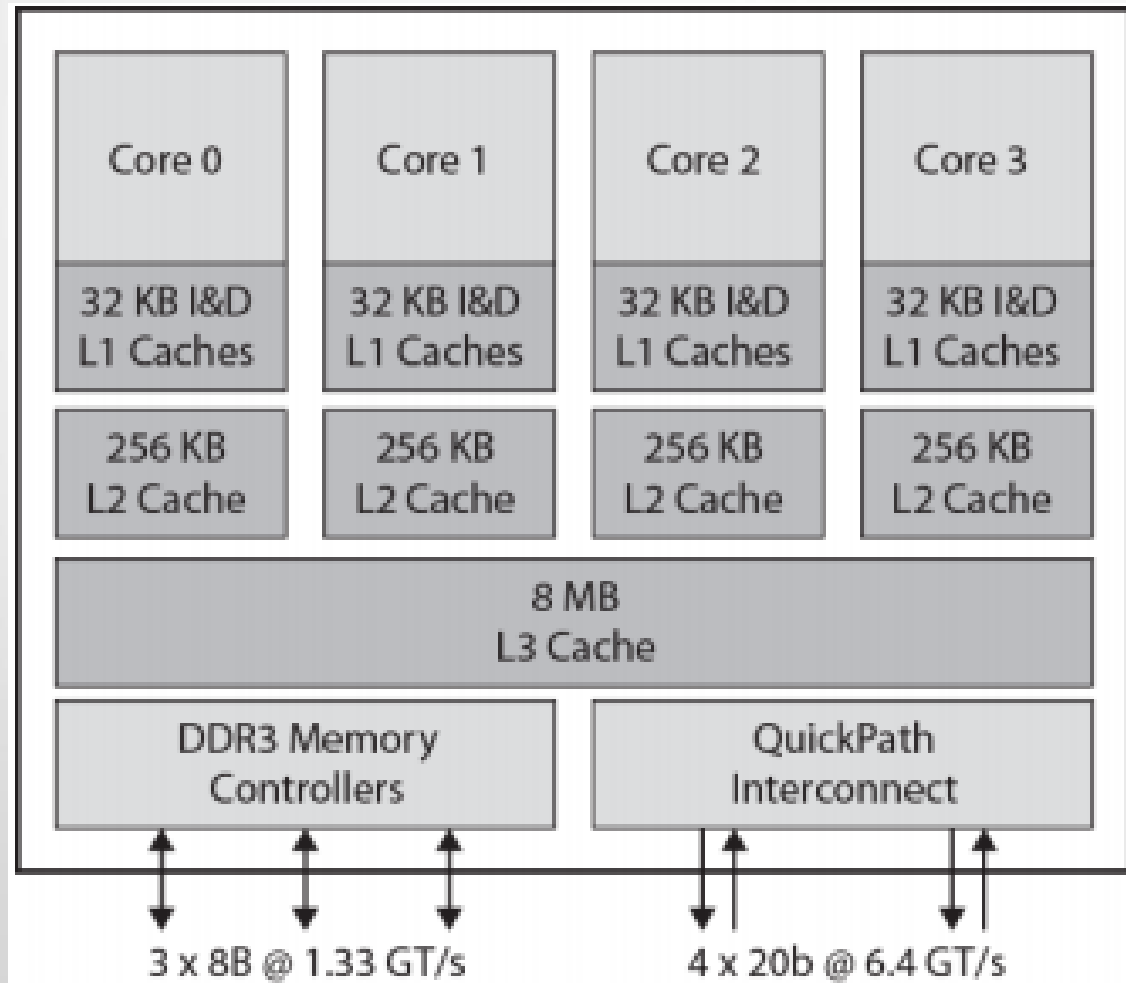
## 3.2. Bộ xử lý đa lõi



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

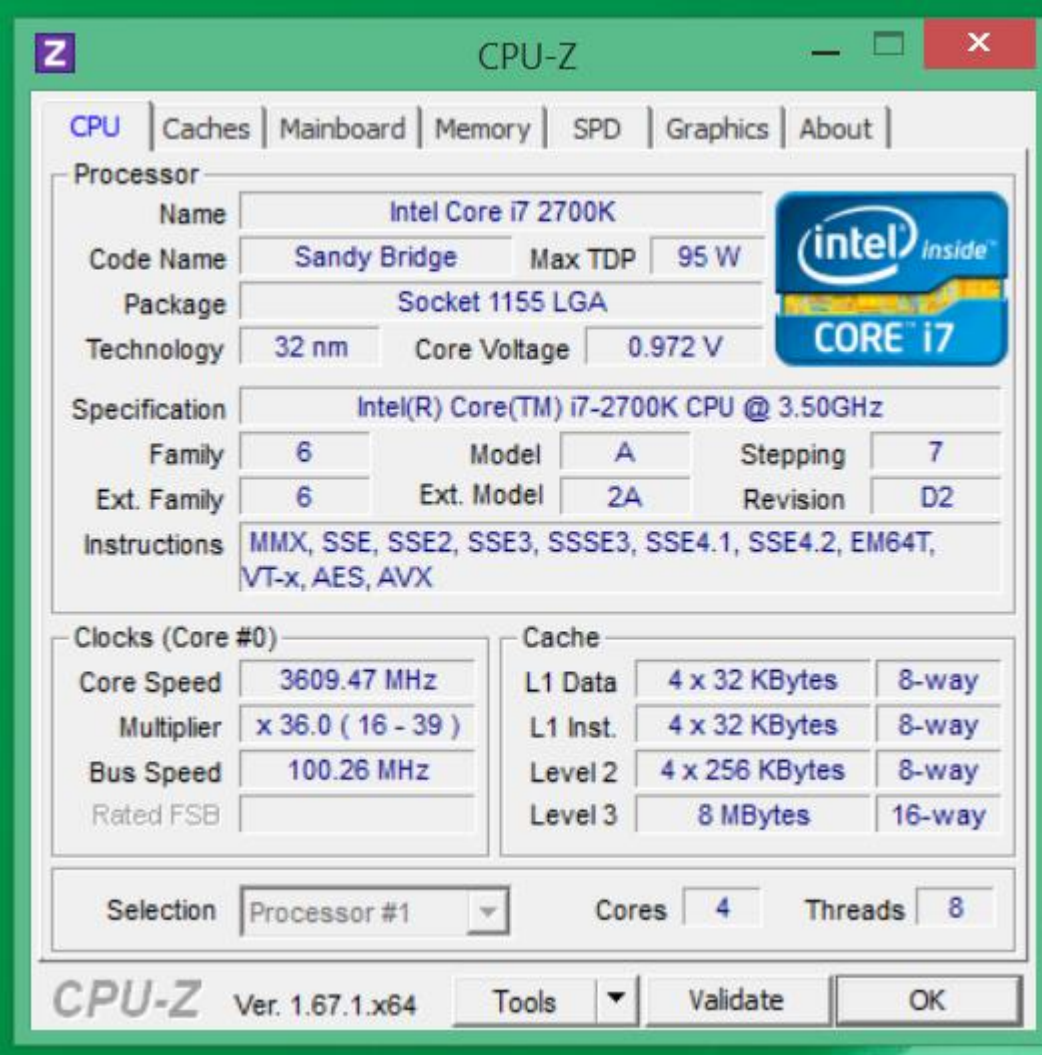
## 3.2. Bộ xử lý đa lõi

### Intel Core i7



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.2. Bộ xử lý đa lõi



The screenshot displays the CPU-Z application window, which provides detailed information about the system's central processing unit. The 'CPU' tab is selected, showing the following data:

Processor			
Name	Intel Core i7 2700K		
Code Name	Sandy Bridge	Max TDP	95 W
Package	Socket 1155 LGA		
Technology	32 nm	Core Voltage	0.972 V
Specification	Intel(R) Core(TM) i7-2700K CPU @ 3.50GHz		
Family	6	Model	A
Ext. Family	6	Ext. Model	2A
Instructions	MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x, AES, AVX		

Clocks (Core #0)		Cache	
Core Speed	3609.47 MHz	L1 Data	4 x 32 KBytes
Multiplier	x 36.0 ( 16 - 39 )	L1 Inst.	4 x 32 KBytes
Bus Speed	100.26 MHz	Level 2	4 x 256 KBytes
Rated FSB		Level 3	8 MBytes

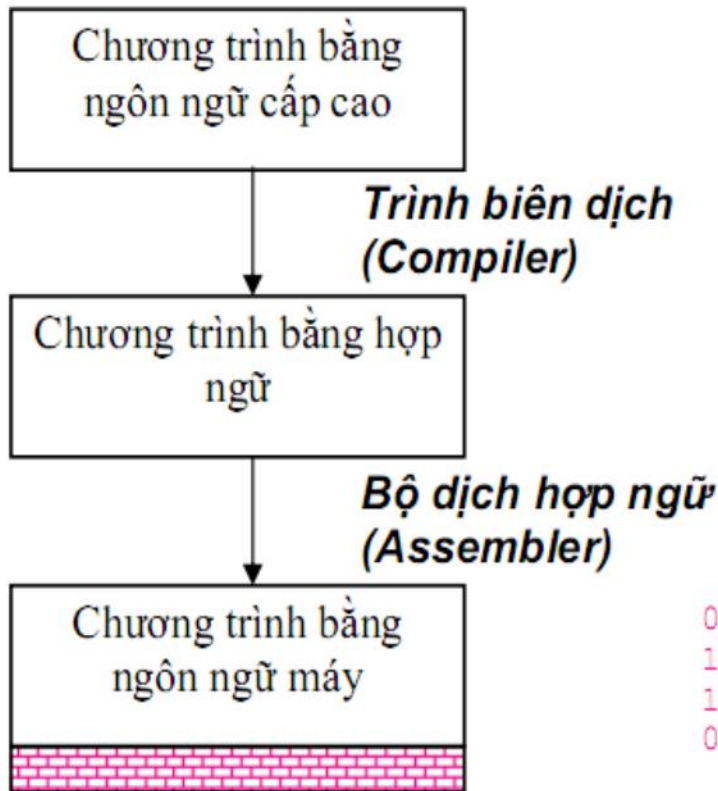
At the bottom, the 'Selection' dropdown is set to 'Processor #1', showing 'Cores: 4' and 'Threads: 8'. The CPU-Z logo and version 'Ver. 1.67.1.x64' are visible, along with 'Tools', 'Validate', and 'OK' buttons.

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.1. Đặc điểm tập lệnh

Thực thi chương trình



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw $15, 0($2)  
lw $16, 4($2)  
sw $16, 0($2)  
sw $15, 4($2)
```

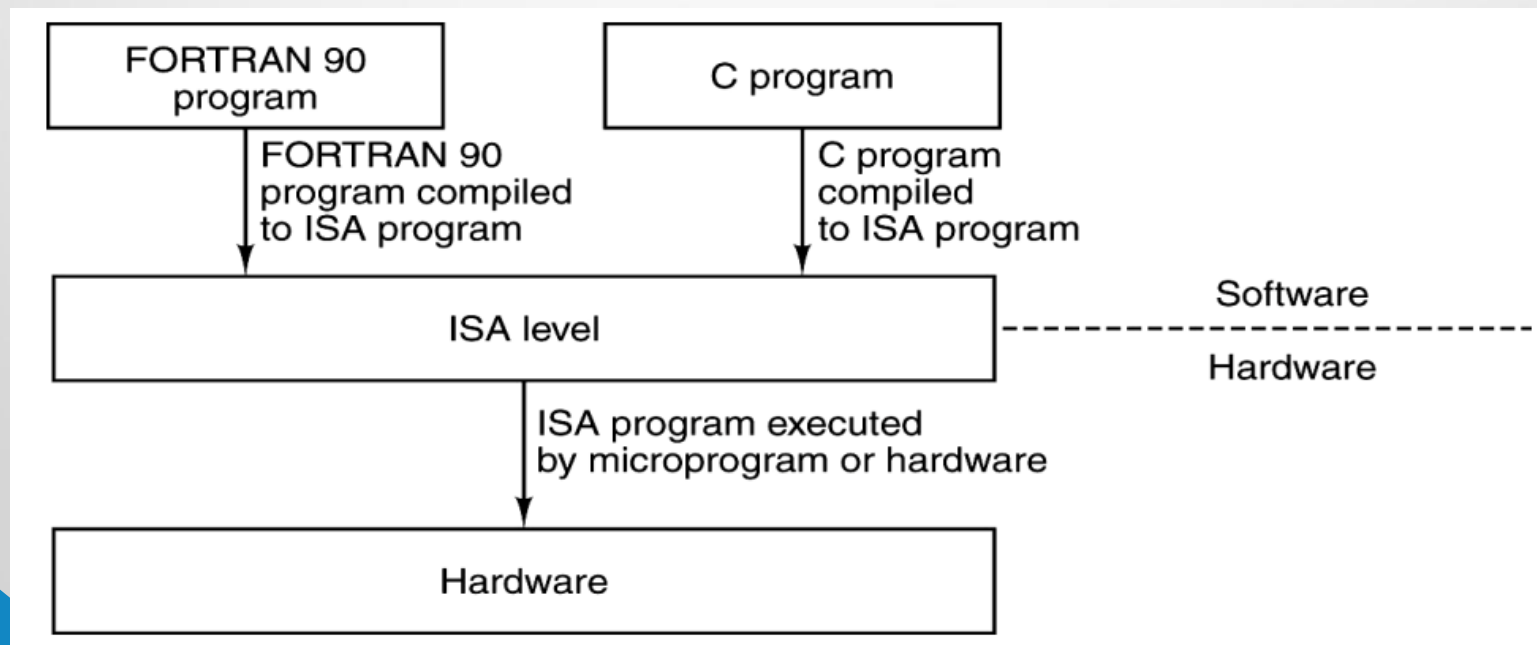
```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.1. Đặc điểm tập lệnh

- Vị trí kiến trúc tập lệnh trong máy tính
  - Nằm giữa phần cứng và ngôn ngữ lập trình bậc cao
  - Giúp cho phần mềm tương thích khi phần cứng thay đổi



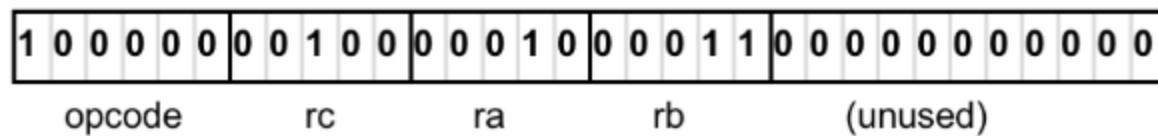
# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.1. Đặc điểm tập lệnh

- Lệnh máy (machine instructions): là các lệnh mà bộ xử lý thực hiện, quyết định hành động của bộ xử lý
- Mỗi bộ xử lý có một tập lệnh xác định. Tập lệnh (instruction set): tập hợp các lệnh khác nhau mà bộ xử lý có thể hiểu và thực hiện

32-bit (4-byte) ADD instruction:



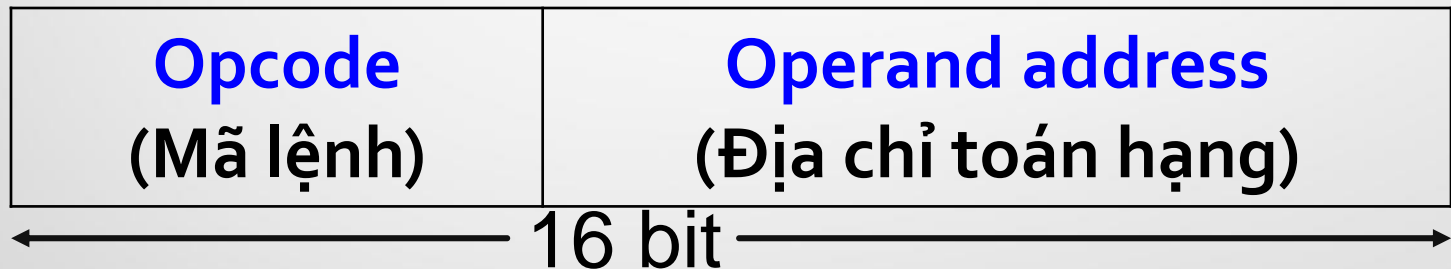


## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

#### *Biểu diễn lệnh:*

- Trong máy tính, mỗi lệnh được biểu diễn bằng một chuỗi bit.
- Lệnh được chia thành các trường tương ứng với các thành phần cấu thành của lệnh.



- Để thuận tiện: sử dụng cách biểu diễn bằng ký hiệu gọi nhớ. Ví dụ: ADD, SUB, LOAD, STORE,...

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

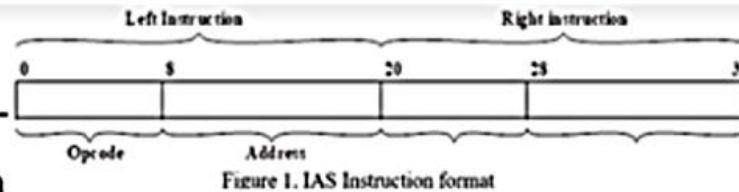
## 3.3. Kiến trúc tập lệnh của CPU

### *Các thành phần của lệnh máy:*

- Mã lệnh (opcode): mã hóa cho thao tác mà bộ xử lý phải thực hiện bằng số nhị phân (làm gì?)

Table 1 The IAS Instruction Set

Instruction Type	Opcode	Symbolic Representation	
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD  M(X)	Add  M(X)  to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB  M(X)	Subtract  M(X)  from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC. put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; that is, shift left one bit position
	00010101	RSH	Divide accumulator by 2; that is, shift right one position,
	00010011	MOV	Move the scalar value presented in the address field in AC



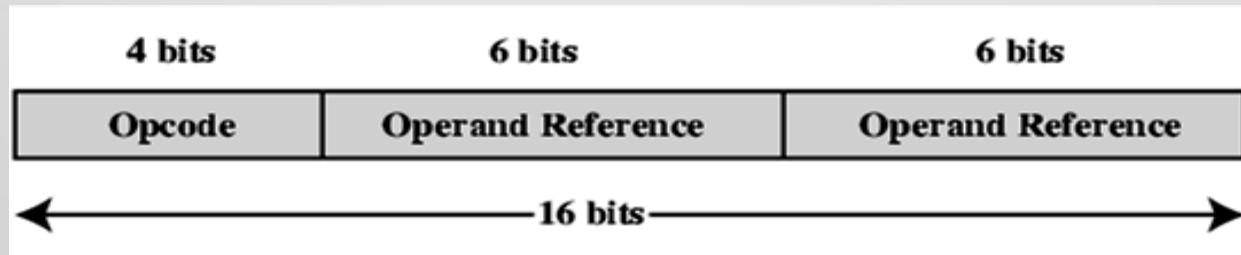
0F9 2  
0FA  
0FB  
0FC

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### *Các thành phần của lệnh máy:*

- Địa chỉ toán hạng (operand address): chỉ ra nơi chứa các toán hạng mà thao tác của VXL sẽ tác động (làm ở đâu?)
  - Toán hạng nguồn: dữ liệu vào của thao tác.
  - Toán hạng đích: dữ liệu ra của thao tác
  - Toán hạng: Thanh ghi, bộ nhớ, thiết bị ngoại vi,...
  - Ví dụ: 1 lệnh 16 bit có 2 toán hạng



## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

### 3.3. Kiến trúc tập lệnh của CPU

**Các kiểu thao tác thông dụng của tập lệnh:**

- ✓ Các lệnh chuyển dữ liệu
- ✓ Các lệnh xử lý số học
- ✓ Các lệnh xử lý logic
- ✓ Các lệnh chuyển điều khiển (rẽ nhánh, nhảy)

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

**Các kiểu thao tác thông dụng của tập lệnh:**

*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

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

**Các kiểu thao tác thông dụng của tập lệnh:**

*Các lệnh số học*

Lệnh			Giải thích
SUB	Y, A, B	$Y \leftarrow A - B$	
MPY	T, D, E	$T \leftarrow D \times E$	
ADD	T, T, C	$T \leftarrow T + C$	
DIV	Y, Y, T	$Y \leftarrow Y \div T$	

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

**Các kiểu thao tác thông dụng của tập lệnh:**

*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** Cát 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** Cát nội dung toán hạng nguồn vào ngăn xếp
- **POP** Lấy nội dung đỉnh ngăn xếp đưa đến toán hạng đích

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

**Các kiểu thao tác thông dụng của tập lệnh:**

*Các lệnh chuyển dữ liệu*

Lệnh		Giải thích
LOAD	D	$AC \leftarrow D$
MPY	E	$AC \leftarrow AC \times E$
ADD	C	$AC \leftarrow AC + C$
STOR	Y	$Y \leftarrow AC$
LOAD	A	$AC \leftarrow A$
SUB	B	$AC \leftarrow AC - B$
DIV	Y	$AC \leftarrow AC \div Y$
STOR	Y	$Y \leftarrow AC$



## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

### 3.3. Kiến trúc tập lệnh của CPU

**Các kiểu thao tác thông dụng của tập lệnh:**

*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

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

#### Các kiểu thao tác thông dụng của tập lệnh:

##### *Các lệnh logic*

- Giả sử có 2 thanh ghi chứa dữ liệu:

(R1) = 1010 1010

(R2) = 0000 1111

P	Q	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

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

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

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

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

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

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

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

**Các kiểu thao tác thông dụng của tập lệnh:**

*Các lệnh chuyển điều khiển*

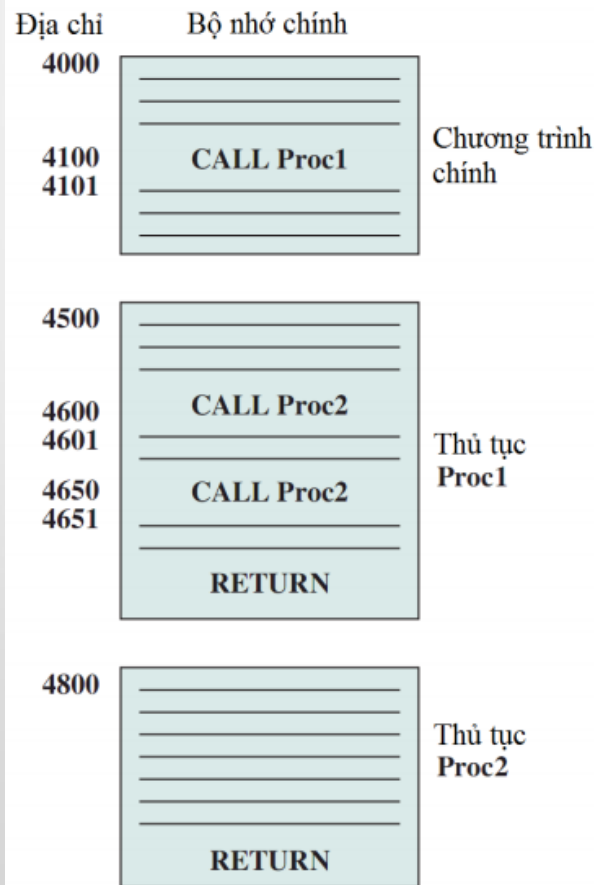
- **JUMP (BRANCH)** Lệnh nhảy không điều kiện:
  - nạp vào PC một địa chỉ xác định
- **JUMP CONDITIONAL** Lệnh nhảy có điều kiện:
  - điều kiện đúng → nạp vào PC một địa chỉ xác định
  - điều kiện sai → không làm gì cả
- **CALL** Lệnh gọi chương trình con:
  - cất nội dung của PC (địa chỉ trở về) ra 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

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

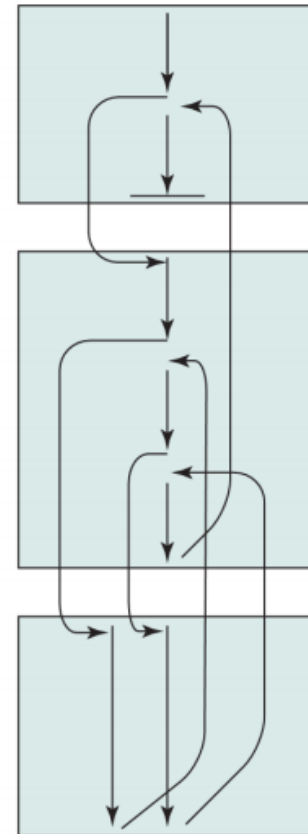
### 3.3. Kiến trúc tập lệnh của CPU

## Các kiểu thao tác thông dụng của tập lệnh:

## Các lệnh chuyển điều khiển



### (a) Lệnh Call và Return



(b) Chuỗi thực thi

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

### 3.3. Kiến trúc tập lệnh của CPU

#### ***3.3.2. Chế độ định địa chỉ toán hạng***

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 (addressing modes)***

là cách thức địa chỉ hóa trong trường địa chỉ của lệnh để xác định nơi chứa toán hạng.

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

### 3.3. Kiến trúc tập lệnh của CPU

#### *3.3.2. Chế độ định địa chỉ 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 dịch chuyển

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

#### 3.3.2. Chế độ định địa chỉ toán hạng

*Định địa chỉ tức thì*

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

- Toán hạng nằm ngay trong trường địa chỉ của lệnh
- Chỉ có thể là toán hạng nguồn
- Ví dụ: **ADD R1, 5** ;  $R1 = R1 + 5$
- Không tham chiếu bộ nhớ, truy nhập toán hạng nhanh
- Dải giá trị của toán hạng bị hạn chế

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

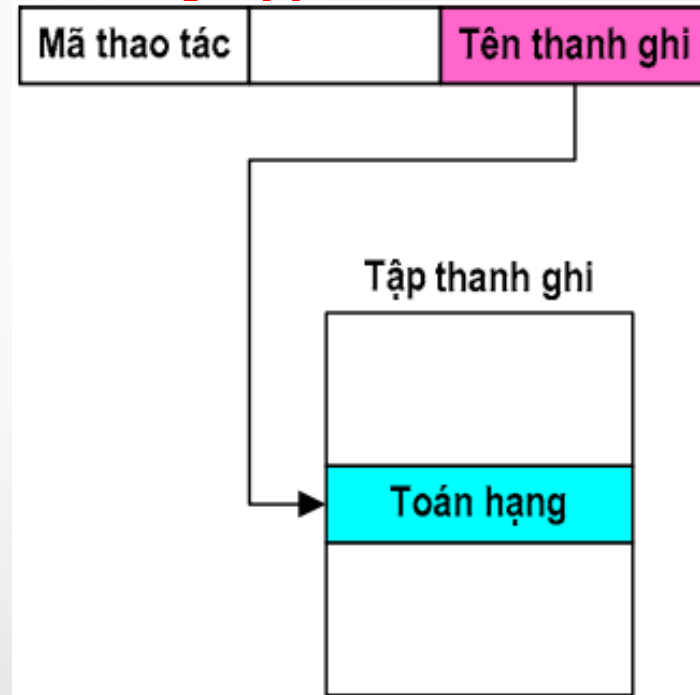
### 3.3.2. Chế độ định địa chỉ toán hạng

#### Định địa chỉ thanh ghi

- Trường địa chỉ chứa tên thanh ghi, trong thanh ghi có chứa toán hạng
- Ví dụ:

**ADD R1, R2** ;  $R1 = R1 + R2$

- Số lượng thanh ghi ít nên trường địa chỉ cần ít bit
- Không tham chiếu bộ nhớ nên truy nhập toán hạng nhanh





# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.2. Chế độ định địa chỉ toán hạng

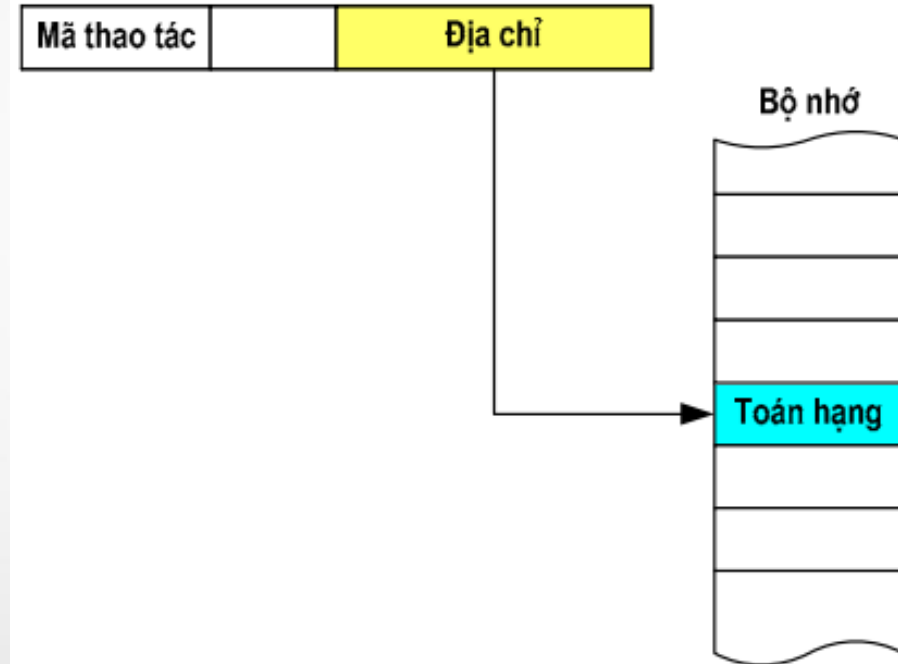
#### *Định địa chỉ trực tiếp*

- Trường địa chỉ chứa địa chỉ ngăn nhớ, trong ngăn nhớ có chứa toán hạng
- Ví dụ:

**ADD R1, A**

*(Cộng nội dung thanh ghi R1 với toán hạng nằm trong ngăn nhớ có địa chỉ là A)*

- CPU tham chiếu bộ nhớ một lần để truy nhập dữ liệu.



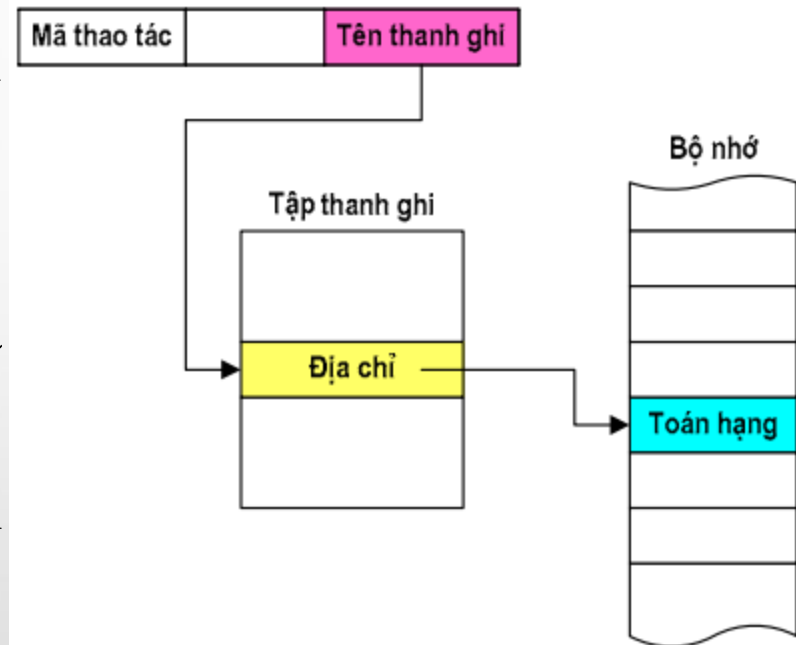
# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.2. Chế độ định địa chỉ toán hạng

#### *Định địa chỉ gián tiếp qua thanh ghi*

- Trường địa chỉ chứa tên thanh ghi, trong thanh ghi chứa địa chỉ của toán hạng
- Thanh ghi này được gọi là thanh ghi con trỏ
- Vùng nhớ có thể tham chiếu là lớn ( $2^n$ ) với  $n$  là độ dài thanh ghi



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

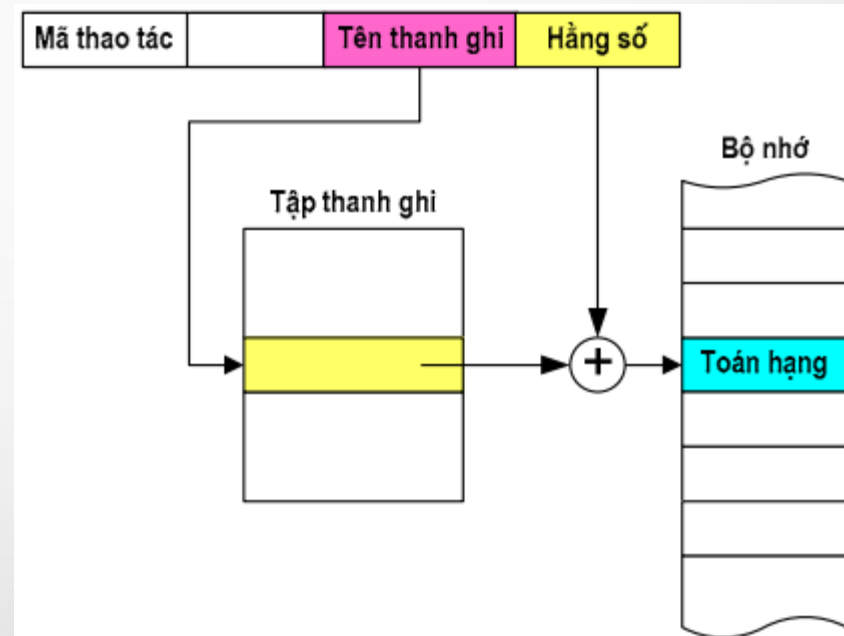
## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.2. Chế độ định địa chỉ toán hạng

#### Định địa chỉ dịch chuyển

- Để xác định vị trí toán hạng, trường địa chỉ chứa hai thành phần:

- Tên thanh ghi
- Hằng số



- Địa chỉ toán hạng = nội dung thanh ghi + hằng số

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.2. Chế độ định địa chỉ toán hạng

#### Số lượng địa chỉ toán hạng trong lệnh

- Ba địa chỉ toán hạng: phổ biến trên các kiến trúc hiện nay  
 $\text{ADD R1, R2, R3} \quad \#R1 = R2 + R3$
- Hai địa chỉ toán hạng: sử dụng trên Intel x86, Motorola 680x0  
 $\text{ADD R1, R2} \quad \#R1 = R1 + R2$
- Một địa chỉ toán hạng: sử dụng trên các kiến trúc thế hệ trước, một toán hạng là ngầm định, thường là thanh ghi tích lũy (accumulator)  $\text{ADD R1} \quad \#Acc = Acc + R1$
- Không địa chỉ toán hạng: Các toán hạng được ngầm định ở ngăn xếp, không thông dụng  
 $\text{PUSH A}$   
 $\text{PUSH B}$   
 $\text{ADD}$   
 $\text{POP C} \quad \#C = A + B$

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.2. Chế độ định địa chỉ toán hạng

Số lượng địa chỉ toán hạng trong lệnh

Lệnh	Giải thích
SUB Y, A, B	$Y \leftarrow A - B$
MPY T, D, E	$T \leftarrow D \times E$
ADD T, T, C	$T \leftarrow T + C$
DIV Y, Y, T	$Y \leftarrow Y \div T$

(a) Lệnh có ba địa chỉ

Lệnh	Giải thích
MOVE Y, A	$Y \leftarrow A$
SUB Y, B	$Y \leftarrow Y - B$
MOVE T, D	$T \leftarrow D$
MPY T, E	$T \leftarrow T \times E$
ADD T, C	$T \leftarrow T + C$
DIV Y, T	$Y \leftarrow Y \div T$

(b) Lệnh có hai địa chỉ

Lệnh	Giải thích
LOAD D	$AC \leftarrow D$
MPY E	$AC \leftarrow AC \times E$
ADD C	$AC \leftarrow AC + C$
STOR Y	$Y \leftarrow AC$
LOAD A	$AC \leftarrow A$
SUB B	$AC \leftarrow AC - B$
DIV Y	$AC \leftarrow AC \div Y$
STOR Y	$Y \leftarrow AC$

(c) Lệnh có một địa chỉ

Chương trình tính biểu thức  $Y = \frac{A - B}{C + (D \times E)}$

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

### 3.3. Kiến trúc tập lệnh của CPU

#### *3.3.3. Kiến trúc tập lệnh MIPS*

#### *Kiến trúc tập lệnh CISC và RISC:*

- **CISC:** Complex Instruction Set Computer:  
Máy tính với tập lệnh đầy đủ. Ví dụ: Intel x86, Motorola 680x0
- **RISC:** Reduced Instruction Set Computer:  
Máy tính với tập lệnh thu gọn. Ví dụ: SunSPARC, Power PC, MIPS, ARM ...

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.3. Kiến trúc tập lệnh MIPS

*Kiến trúc tập lệnh CISC và RISC:*

Loại Hãng SX Hệ thống MT Năm SX	CISC			RISC	
	IBM	DEC VAX	Intel	Motorola	MIPS
	370/168	11/780	486	88000	R4000
	1973	1978	1989	1988	1991
Số lượng lệnh	208	303	235	51	94
Kích thước lệnh (B)	2-6	2-57	1-11	4	32
Addressing modes	4	22	11	3	1
Số lượng thanh ghi	16	16	8	32	32
Vi ChươngTrình (KB)	420	480	246	0	0

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### *3.3.3. Kiến trúc tập lệnh MIPS*

#### *Các đặc trưng của RISC*

- ✓ Số lượng lệnh ít
- ✓ Hầu hết các lệnh truy nhập toán hạng ở các thanh ghi
- ✓ Truy cập bộ nhớ bằng lệnh LOAD/STORE
- ✓ Thời gian thực hiện lệnh là như nhau
- ✓ Các lệnh có độ dài cố định (32bit)
- ✓ Số lượng dạng lệnh ít
- ✓ Có ít phương pháp định địa chỉ toán hạng
- ✓ CPU có tập thanh ghi lớn
- ✓ Hỗ trợ các thao tác của ngôn ngữ bậc cao



## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

### 3.3. Kiến trúc tập lệnh của CPU

#### *3.3.3. Kiến trúc tập lệnh MIPS*

**MIPS** (Microprocessor without Interlocked Pipeline Stages), là kiến trúc bộ tập lệnh RISC phát triển bởi MIPS Technologies. Ban đầu kiến trúc MIPS là 32bit, và sau đó là phiên bản 64 bit. Nhiều sửa đổi của MIPS, bao gồm MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS32 và MIPS64. Phiên bản hiện tại là MIPS32 và MIPS64.

MIPS là kiến trúc RISC điển hình và được sử dụng trong nhiều sản phẩm thực tế.

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### *3.3.3. Kiến trúc tập lệnh MIPS*

#### **Tập thanh ghi của MIPS**

- MIPS có tập 32 thanh ghi 32 – bit, được đánh số từ 0 đến 31 (mã hóa bằng 5- bit)
- Chương trình hợp dịch Assembler đặt tên:
  - Bắt đầu bằng dấu \$
  - \$t0, \$t1,...\$t9 chứa các giá trị tạm thời
  - \$s0, \$s1,...\$s7 cất các biến
- Qui ước gọi dữ liệu trong MIPS
  - Dữ liệu 32 bit được gọi là “word”
  - Dữ liệu 16 bit gọi là “halfword”

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.3. Kiến trúc tập lệnh MIPS

#### Tập thanh ghi của MIPS

Tên thanh ghi	Số hiệu thanh ghi	Công dụng
\$zero	0	the constant value 0, chứa hằng số = 0
\$at	1	assembler temporary, giá trị tạm thời cho hợp ngữ
\$v0-\$v1	2-3	procedure return values, các giá trị trả về của thủ tục
\$a0-\$a3	4-7	procedure arguments, các tham số vào của thủ tục
\$t0-\$t7	8-15	temporaries, chứa các giá trị tạm thời
\$s0-\$s7	16-23	saved variables, lưu các biến
\$t8-\$t9	24-25	more temporarie, chứa các giá trị tạm thời
\$k0-\$k1	26-27	OS temporaries, các giá trị tạm thời của OS
\$gp	28	global pointer, con trỏ toàn cục
\$sp	29	stack pointer, con trỏ ngăn xếp
\$fp	30	frame pointer, con trỏ khung
\$ra	31	procedure return address, địa chỉ trở về của thủ tục

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### *3.3.3. Kiến trúc tập lệnh MIPS*

#### Mã máy trong MIPS

- Các lệnh của MIPS:
  - Được mã hóa bằng các từ lệnh 32 bit
  - Mỗi lệnh chiếm 4 byte trong bộ nhớ
  - Có ít dạng lệnh

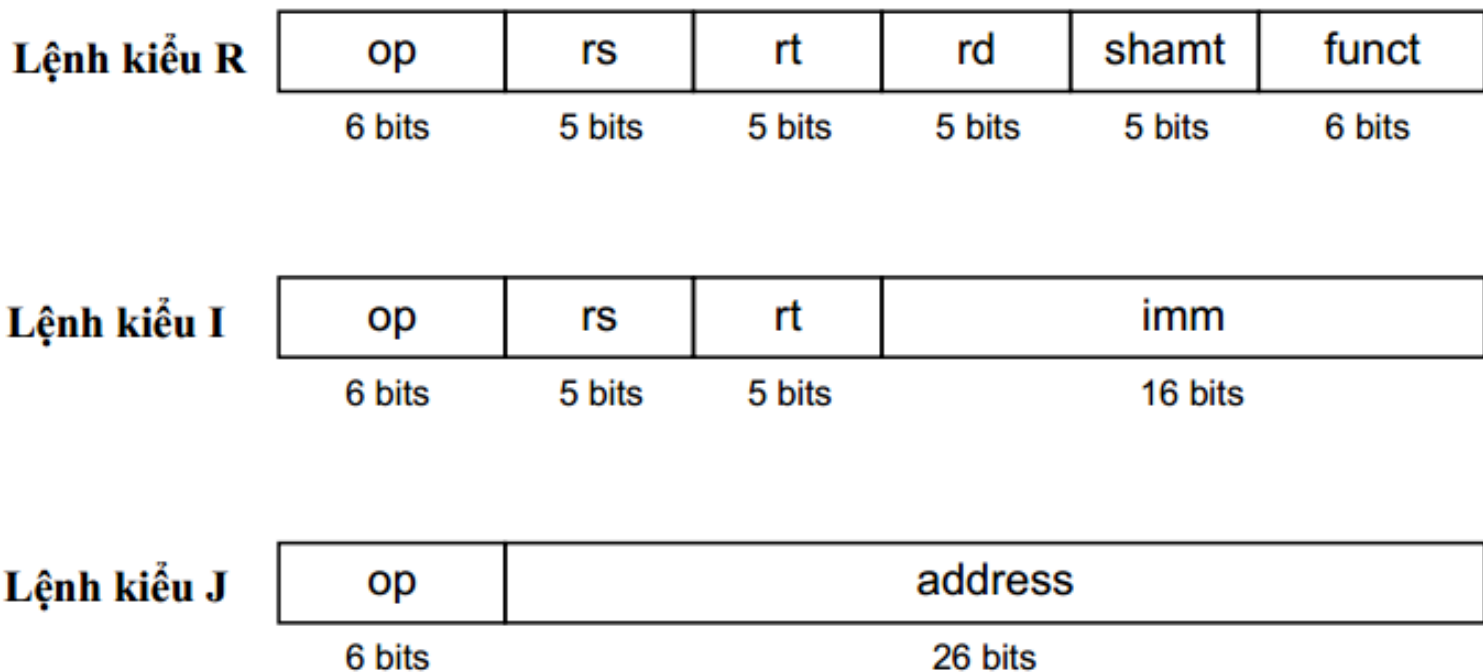
```
add $t0, $s1, $s2
add $t1, $s3, $s4
sub $s0, $t0, $t1
```

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.3. Kiến trúc tập lệnh MIPS

#### Các kiểu lệnh máy của MIPS



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.3. Kiến trúc tập lệnh MIPS

#### Các kiểu lệnh máy của MIPS

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

#### ■ Các trường của lệnh

- op (operation code - opcode): mã thao tác
  - với các lệnh kiểu R, op = 000000
- rs: số hiệu thanh ghi nguồn thứ nhất
- rt: số hiệu thanh ghi nguồn thứ hai
- rd: số hiệu thanh ghi đích
- shamt (shift amount): số bit được dịch, chỉ dùng cho lệnh dịch bit, với các lệnh khác shamt = 00000
- funct (function code): mã hàm

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.3. Kiến trúc tập lệnh MIPS

#### Các kiểu lệnh máy của MIPS

Ví dụ: Mã máy của lệnh add, sub

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

*add \$t0, \$s1, \$s2*

0	\$s1	\$s2	\$t0	0	add
0	17	18	8	0	32
000000	10001	10010	01000	00000	100000

(0x02324020)

*sub \$s0, \$t3, \$t5*

0	\$t3	\$t5	\$s0	0	sub
0	11	13	16	0	34
000000	01011	01101	10000	00000	100010

(0x016D8022)

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

---

### 3.3. Kiến trúc tập lệnh của CPU

#### *3.3.3. Kiến trúc tập lệnh Intel x86*

- ✓ Thuật ngữ x86 dùng để chỉ tới kiến trúc tập lệnh của dòng vi xử lý 8086 của Intel. 8086 được Intel đưa ra năm 1978.
- ✓ Kiến trúc x86 có độ dài chỉ lệnh không cố định.
- ✓ Các phép toán có thể thực hiện với 8, 16, 32, 64 bit tùy theo thể hệ kiến trúc. Opcode chính của x86 có thể lên đến 3 byte nên rất linh hoạt.



# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.3. Kiến trúc tập lệnh Intel x86

#### Moves

MOV DST, SRC	Move SRC to DST
PUSH SRC	Push SRC onto the stack
POP DST	Pop a word from the stack to DST
XCHG DS1, DS2	Exchange DS1 and DS2
LEA DST, SRC	Load effective addr of SRC into DST
CMOVCc DST, SRC	Conditional move

#### Arithmetic

ADD DST, SRC	Add SRC to DST
SUB DST, SRC	Subtract SRC from DST
MUL SRC	Multiply EAX by SRC (unsigned)
IMUL SRC	Multiply EAX by SRC (signed)
DIV SRC	Divide EDX:EAX by SRC (unsigned)
IDIV SRC	Divide EDX:EAX by SRC (signed)
ADC DST, SRC	Add SRC to DST, then add carry bit
SBB DST, SRC	Subtract SRC & carry from DST
INC DST	Add 1 to DST
DEC DST	Subtract 1 from DST
NEG DST	Negate DST (subtract it from 0)

#### Binary coded decimal

DAA	Decimal adjust
DAS	Decimal adjust for subtraction
AAA	ASCII adjust for addition
AAS	ASCII adjust for subtraction
AAM	ASCII adjust for multiplication
AAD	ASCII adjust for division

#### Boolean

AND DST, SRC	Boolean AND SRC into DST
OR DST, SRC	Boolean OR SRC into DST
XOR DST, SRC	Boolean Exclusive OR SRC to DST
NOT DST	Replace DST with 1's complement

#### Shift/rotate

SAL/SAR DST, #	Shift DST left/right # bits
SHL/SHR DST, #	Logical shift DST left/right # bits
ROL/ROR DST, #	Rotate DST left/right # bits
RCL/RCR DST, #	Rotate DST through carry # bits

# CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

## 3.3. Kiến trúc tập lệnh của CPU

### 3.3.3. Kiến trúc tập lệnh Intel x86

#### Transfer of control

JMP ADDR	Jump to ADDR
Jxx ADDR	Conditional jumps based on flags
CALL ADDR	Call procedure at ADDR
RET	Return from procedure
IRET	Return from interrupt
LOOPxx	Loop until condition met
INT n	Initiate a software interrupt
INTO	Interrupt if overflow bit is set

#### Strings

LODS	Load string
STOS	Store string
MOVS	Move string
CMPS	Compare two strings
SCAS	Scan Strings

#### Test/compare

TEST SRC1, SRC2	Boolean AND operands, set flags
CMP SRC1, SRC2	Set flags based on SRC1 - SRC2

#### Miscellaneous

SWAP DST	Change endianness of DST
CWQ	Extend EAX to EDX:EAX for division
CWDE	Extend 16-bit number in AX to EAX
ENTER SIZE, LV	Create stack frame with SIZE bytes
LEAVE	Undo stack frame built by ENTER
NOP	No operation
HLT	Halt
IN AL, PORT	Input a byte from PORT to AL
OUT PORT, AL	Output a byte from AL to PORT
WAIT	Wait for an interrupt

SRC = source  
DST = destination

# = shift/rotate count  
LV = # locals

## CHƯƠNG 3. BỘ XỬ LÝ TRUNG TÂM

### 3.3. Kiến trúc tập lệnh của CPU

#### *3.3.3. Kiến trúc tập lệnh Intel x86*

*Thống kê 10 lệnh Intel x86 sử dụng nhiều nhất*

TT Lệnh		Tỷ lệ (%)
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
Total		96%



**Thank you!**