

# LẬP TRÌNH HỆ THỐNG

---

ThS. Đỗ Thị Hương Lan  
(landth@uit.edu.vn)



**TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN - ĐHQG-HCM**  
**KHOA MẠNG MÁY TÍNH & TRUYỀN THÔNG**  
FACULTY OF COMPUTER NETWORK AND COMMUNICATIONS

Tầng 8 - Tòa nhà E, trường ĐH Công nghệ Thông tin, ĐHQG-HCM  
Điện thoại: (08)3 725 1993 (122)

# Ôn thi cuối kỳ



# Nội dung

---

- **Các nội dung ôn tập:**
  - Các thành phần của hệ thống
  - Assembly Instruction (AT&T)
  - Lập trình mức máy tính
- **Giải đáp các bài tập**

# Nội dung

---

- **Các nội dung ôn tập:**
  - Các thành phần của hệ thống
  - Assembly Instruction (AT&T)
  - Lập trình mức máy tính
- Giải đáp các bài tập

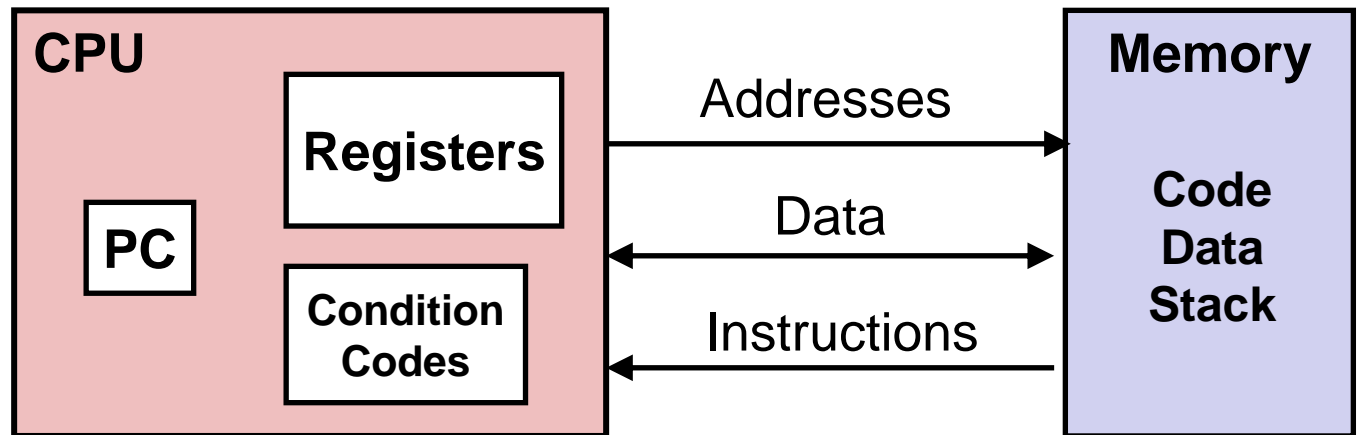
# Các thành phần của hệ thống

## ■ CPU

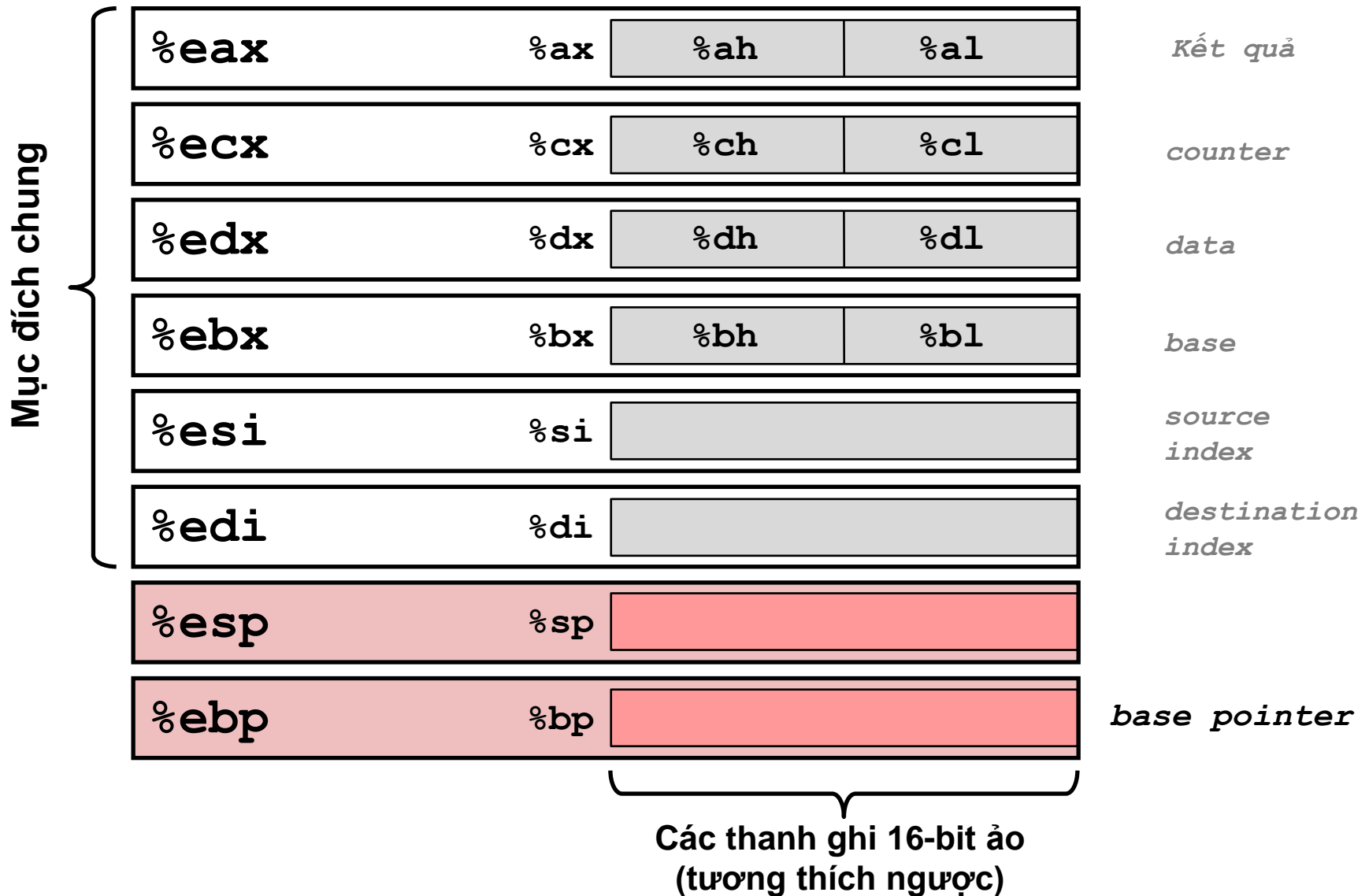
- Bộ tính toán ALU
- Thanh ghi (Register)
- Bộ nhớ cache (L1, L2, L3)

## ■ Bộ nhớ chính

- Ổ đĩa: SSD, HDD



# Các thanh ghi IA32 – 8 thanh ghi 32 bit



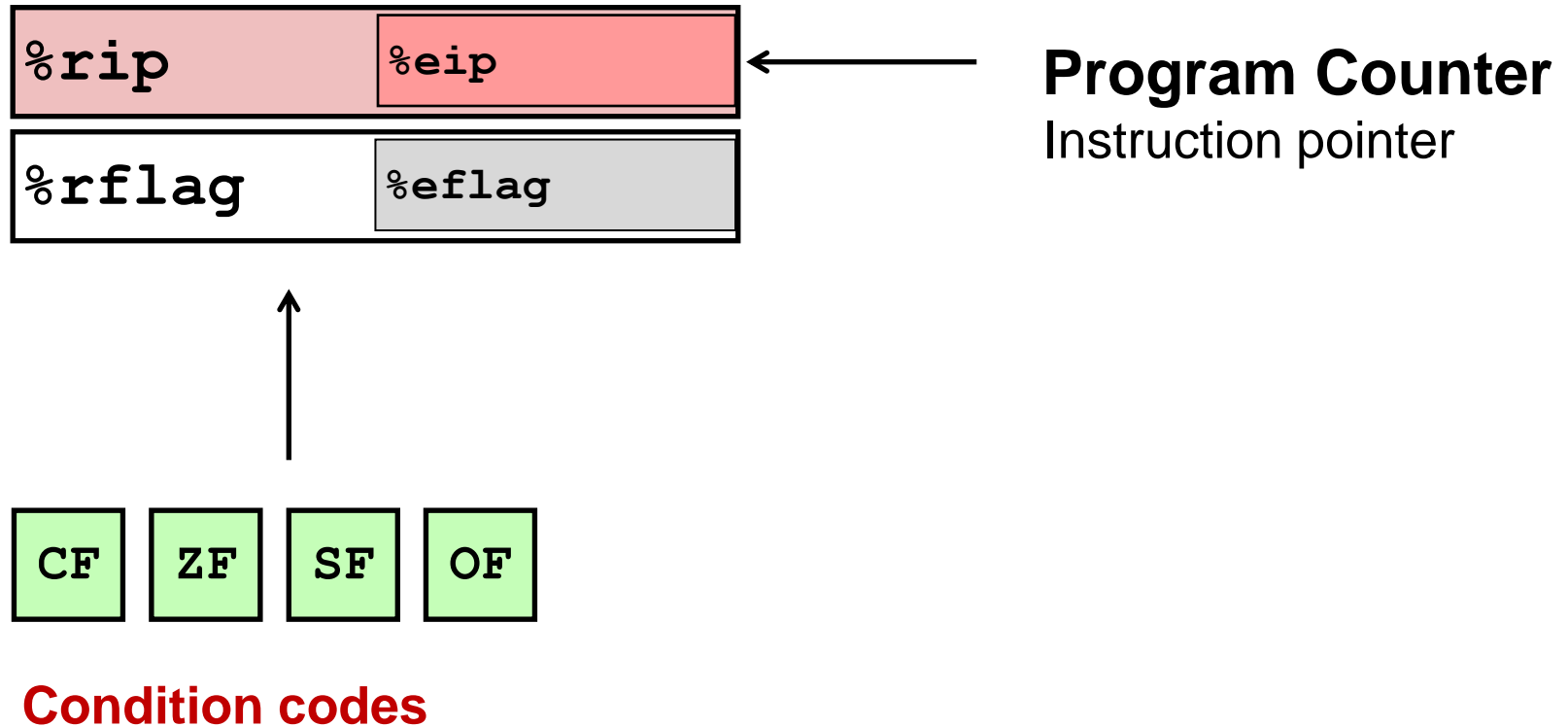
# Các thanh ghi trong x86-64

<b>%rax</b>	<b>%eax</b>
<b>%rbx</b>	<b>%ebx</b>
<b>%rcx</b>	<b>%ecx</b>
<b>%rdx</b>	<b>%edx</b>
<b>%rsi</b>	<b>%esi</b>
<b>%rdi</b>	<b>%edi</b>
<b>%rsp</b>	<b>%esp</b>
<b>%rbp</b>	<b>%ebp</b>

<b>%r8</b>	<b>%r8d</b>
<b>%r9</b>	<b>%r9d</b>
<b>%r10</b>	<b>%r10d</b>
<b>%r11</b>	<b>%r11d</b>
<b>%r12</b>	<b>%r12d</b>
<b>%r13</b>	<b>%r13d</b>
<b>%r14</b>	<b>%r14d</b>
<b>%r15</b>	<b>%r15d</b>

16 thanh ghi 64 bit

# Một vài thanh ghi đặc biệt





# Nội dung

---

- **Các nội dung ôn tập:**
  - Các thành phần của hệ thống
  - Assembly Instruction (AT&T)
  - Lập trình mức máy tính
- Giải đáp các bài tập

# Khác biệt giữa các định dạng AT&T vs Intel

## ■ Khác biệt giữa 2 định dạng assembly: AT&T vs Intel

	AT&T	Intel
Thứ tự toán hạng	<code>movl source, dest</code>	<code>mov dest, source</code>
Thanh ghi	Có % trước tên thanh ghi <code>%eax</code>	Không có prefix trước tên thanh ghi <code>eax</code>
Lệnh mov	Có suffix <code>movl, movlq, movb...</code>	Không có suffix <code>mov</code>
Địa chỉ ô nhớ	<code>8 (%ebp)</code>	<code>[ebp + 8]</code>
Có thể thấy ở đâu?	gcc: option <code>-masm=att</code> (mặc định) objdump: option <code>-M att</code> (mặc định)	<ul style="list-style-type: none"><li>• IDA Pro</li><li>• gcc: option <code>-masm=intel</code></li><li>• objdump: option <code>-M intel</code></li></ul>

# Lệnh mov với định dạng AT&T

	Source	Dest	Src, Dest	C Analog
movq	Imm	Reg	movl \$0x4, %rax	temp = 0x4;
		Mem	movl \$-147, (%rax)	*p = -147;
	Reg	Reg	movl %rax, %rdx	
		Mem	movl %rax, (%rdx)	*p = temp;
	Mem	Reg		
			movl (%rax), %rdx	temp = *p;

# Lệnh mov với định dạng AT&T

Instruction hợp lệ	Instruction không hợp lệ
<code>movl Imm, Reg</code> <code>movl Imm, Mem</code> <code>movl Reg, Reg</code> <code>movl Reg, Mem</code> <code>movl Mem, Reg</code>	<code>movl Reg, Imm</code> <code>movl Mem, Imm</code> <code>movl Imm, Imm</code> <code>movl Mem, Mem</code>

- **Imm:** hằng số, có ký hiệu \$ phía trước
- **Reg:** thanh ghi hỗ trợ trên hệ thống, có ký hiệu % phía trước
- **Mem:** địa chỉ ô nhớ, có thể là địa chỉ cụ thể như 0x100 hay (0x100), hoặc là biểu thức biểu diễn *Imm(Reg1, Reg2, Imm)*

# Lưu ý: Suffix cho lệnh trong AT&T

- Quyết định số byte dữ liệu sẽ được xử lý, ví dụ lệnh **mov**
  - **movb** 1 byte
  - **movw** 2 bytes
  - **movl** 4 bytes
  - **movq** 8 bytes (dùng với các thanh ghi x86\_64)
  - **mov** Số bytes tùy ý (phù hợp với tất cả số byte ở trên)
- Lưu ý: **Các thanh ghi** dùng trong lệnh **mov** cần đảm bảo phù hợp với **suffix**
  - Số byte dữ liệu sẽ được move

? Có bao nhiêu lệnh **mov** **hợp lệ** trong các lệnh bên?

**movl** %eax, %ebx

**movb** \$123, %b1

**movl** %eax, %b1 **×**

**movb** \$3, (%ecx)

**mov** (%eax), %b1

# Các chế độ đánh địa chỉ bộ nhớ đầy đủ

## ■ Dạng tổng quát nhất

$$D(Rb, Ri, S) \quad \text{Mem}[\text{Reg}[Rb] + S * \text{Reg}[Ri] + D]$$

- D: Hằng số “dịch chuyển” 1, 2, hoặc 4 bytes
- Rb: Base register: Bất kỳ thanh ghi nào được hỗ trợ
- Ri: Index register: Bất kỳ thanh ghi nào, ngoại trừ `%rsp` hoặc `%esp`
- S: Scale: 1, 2, 4, hoặc 8

## ■ Các trường hợp đặc biệt

$$(Rb, Ri) \quad \text{Mem}[\text{Reg}[Rb] + \text{Reg}[Ri]]$$

$$D(Rb, Ri) \quad \text{Mem}[\text{Reg}[Rb] + \text{Reg}[Ri] + D]$$

$$(Rb, Ri, S) \quad \text{Mem}[\text{Reg}[Rb] + S * \text{Reg}[Ri]]$$

# leal VS movl

- Với **src** là 1 biểu thức tính toán địa chỉ
- **movl Src, Dst**
  - Tính toán địa chỉ ô nhớ dựa trên biểu thức tính toán ở **src**
  - Truy xuất đến ô nhớ có địa chỉ tính toán được để lấy dữ liệu
  - Đưa dữ liệu lấy được vào **dst**
- **leal Src, Dst**
  - Tính toán địa chỉ ô nhớ dựa trên biểu thức tính toán ở **src**
  - **Không truy xuất ô nhớ**
  - Gán trực tiếp địa chỉ tính toán được cho **dst**
  - Ứng dụng: tính toán các biểu thức toán học

# Một số lệnh assembly toán học

Instruction		Effect	Description
leal	$S, D$	$D \leftarrow \&S$	Load effective address
INC	$D$	$D \leftarrow D + 1$	Increment
DEC	$D$	$D \leftarrow D - 1$	Decrement
NEG	$D$	$D \leftarrow -D$	Negate
NOT	$D$	$D \leftarrow \sim D$	Complement
ADD	$S, D$	$D \leftarrow D + S$	Add
SUB	$S, D$	$D \leftarrow D - S$	Subtract
IMUL	$S, D$	$D \leftarrow D * S$	Multiply
XOR	$S, D$	$D \leftarrow D \wedge S$	Exclusive-or
OR	$S, D$	$D \leftarrow D \mid S$	Or
AND	$S, D$	$D \leftarrow D \& S$	And
SAL	$k, D$	$D \leftarrow D \ll k$	Left shift
SHL	$k, D$	$D \leftarrow D \ll k$	Left shift (same as SAL)
SAR	$k, D$	$D \leftarrow D \gg_A k$	Arithmetic right shift
SHR	$k, D$	$D \leftarrow D \gg_L k$	Logical right shift

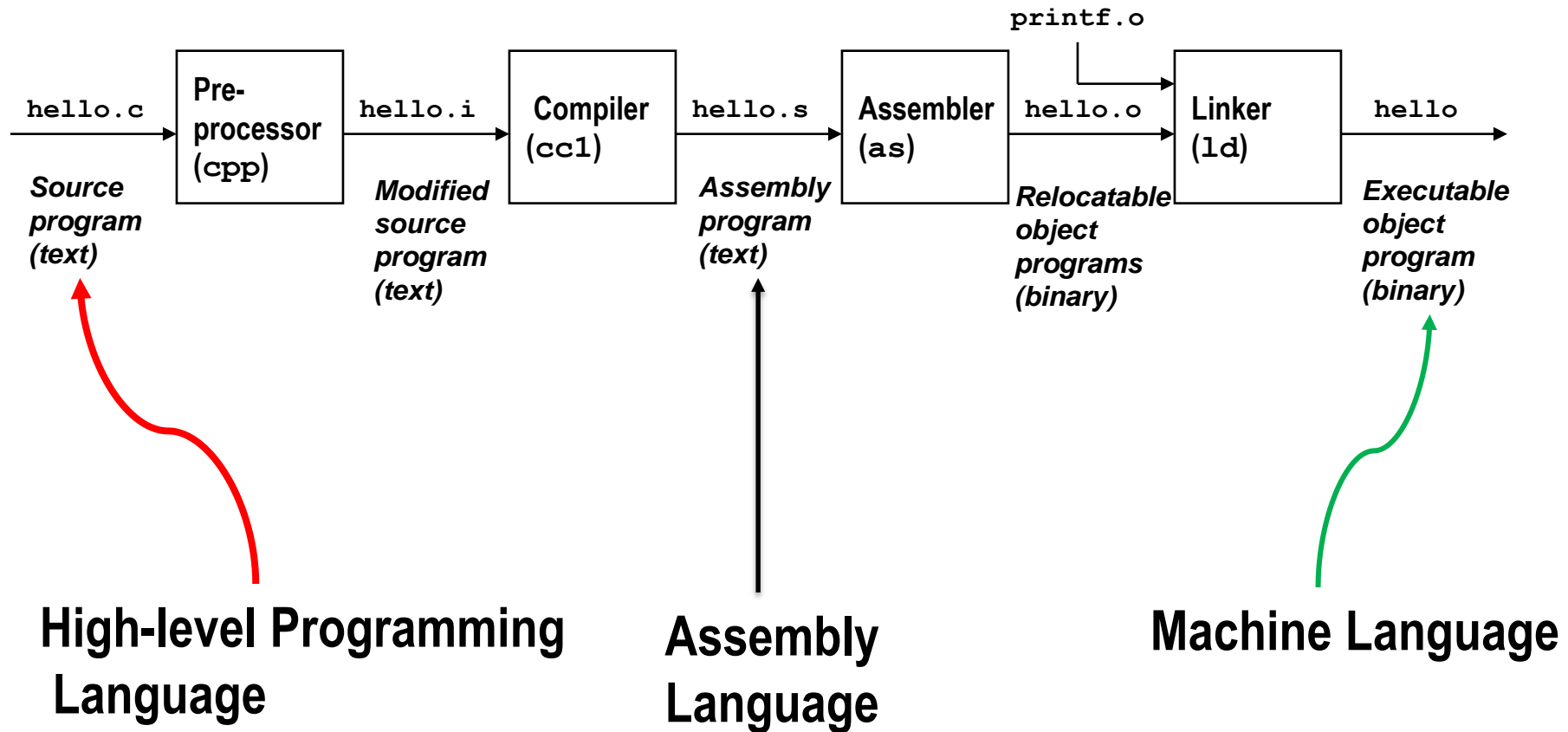


# Nội dung

---

- **Các nội dung ôn tập:**
  - Các thành phần của hệ thống
  - Assembly Instruction (AT&T)
  - Lập trình mức máy tính
- Giải đáp các bài tập

# Chuyển đổi từ mã nguồn sang file thực thi



# Hiểu các chương trình assembly

---

- Ngôn ngữ assembly (*chi tiết ở mục trước*)
- Lập trình mức máy tính (machine-level)
  - Kiểu dữ liệu
  - Điều khiển luồng (rẽ nhánh, vòng lặp,...)
  - Thủ tục/Hàm và Stack
  - Mảng, Structure, Union
- Linking
- Các topic liên quan đến ATTT
  - Reverse engineering
  - Buffer overflow
  - Truy xuất ngoài mảng

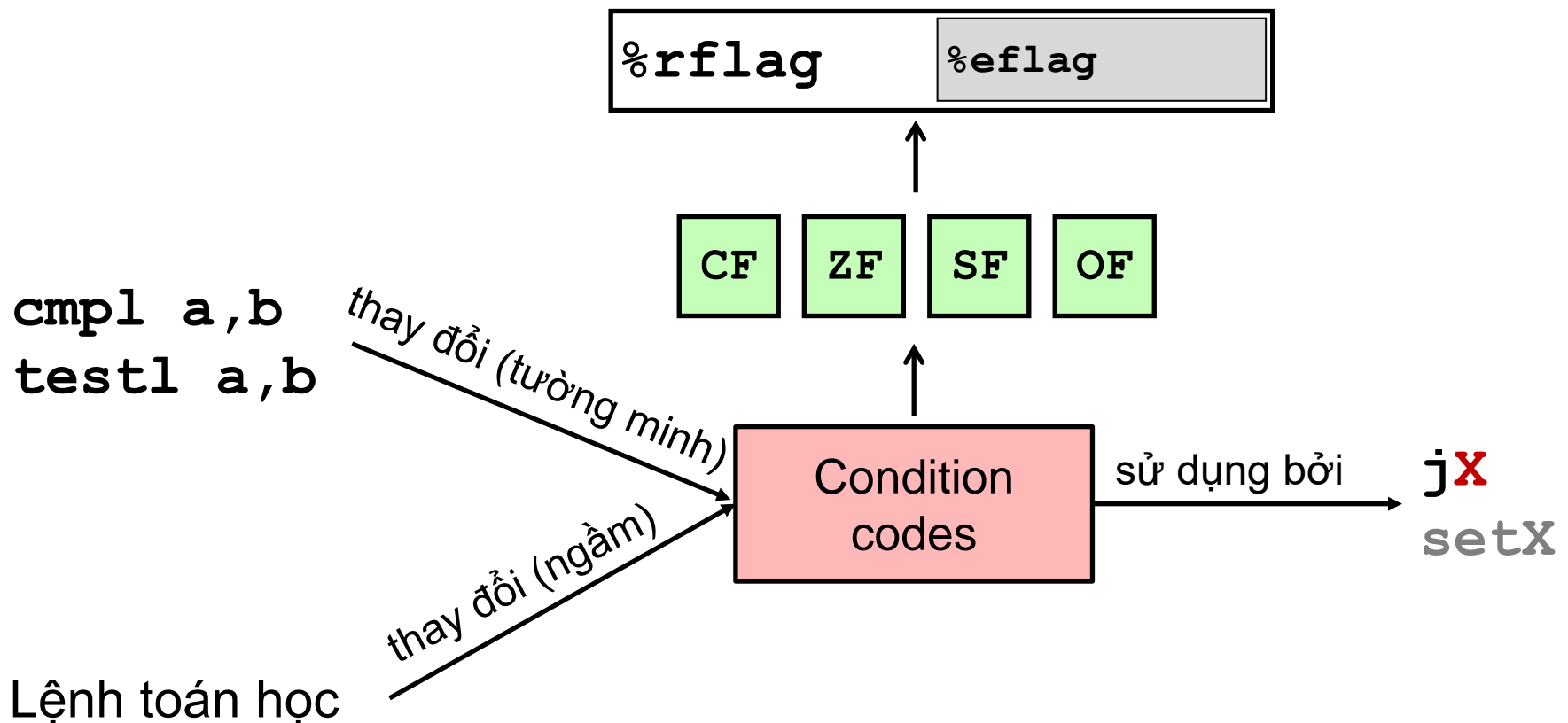
# Lập trình mức máy tính (1)

## ■ Kiểu dữ liệu

C Data Type	Typical 32-bit	Typical 64-bit	x86-64
<code>char</code>	1	1	1
<code>short</code>	2	2	2
<code>int</code>	4	4	4
<code>long</code>	4	8	8
<code>float</code>	4	4	4
<code>double</code>	8	8	8
<code>long double</code>	–	–	10/16
<code>pointer</code>	4	8	8

# Lập trình mức máy tính (2)

## ■ Điều khiển luồng ( rẽ nhánh, vòng lặp...)



# Lập trình mức máy tính (3)

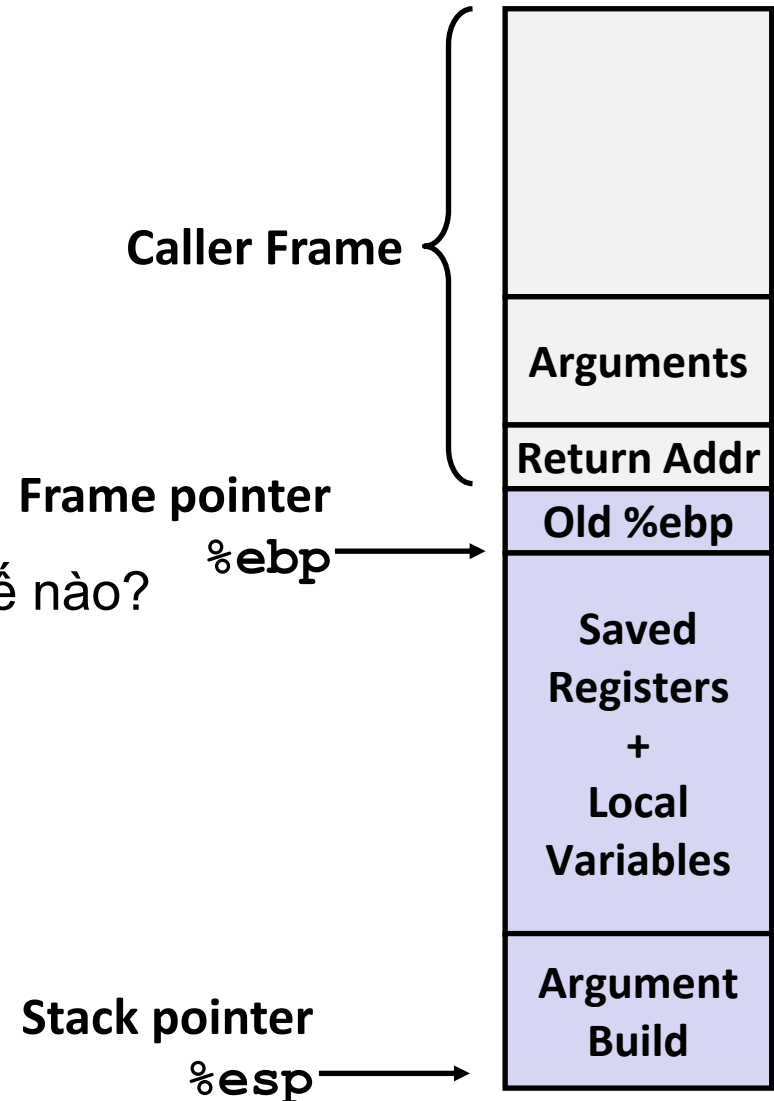
---

- Điều khiển luồng ( rẽ nhánh, vòng lặp...): Từ C sang assembly
  - Code C → Goto Version → Assembly
- **Điều khiển luồng ( rẽ nhánh, vòng lặp...): Từ assembly sang C**
  - Rẽ nhánh: các điều kiện so sánh, các đoạn code tương ứng với trường hợp đúng/sai
  - Vòng lặp: điều kiện dừng, cập nhật, than vòng lặp...

# Lập trình mức máy tính (4)

## ■ Thủ tục/Hàm và stack

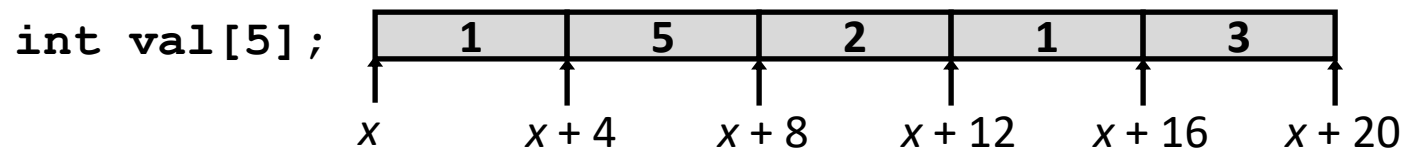
- Stack: %ebp và %esp
- Push/Pop
- Call
- Ret
- Return address?
- Vị trí các tham số/biến cục bộ
- IA32 và x86\_64 khác nhau như thế nào?



# Lập trình mức máy tính (5)

## ■ Mảng, Structure, Union

- Mảng 1 chiều/2 chiều
- Căn chỉnh trong structure
- Cách xác định vị trí của các thành phần trong mảng/structure/union
- Kích thước tổng của mảng/structure/union





# Lập trình mức máy tính (6)

---

## ■ Linking

- Thành phần trong chương trình nào được xem xét là symbol?
- Các kiểu symbol: global, external, local
- Luật phân giải symbol trùng tên: strong hay weak symbol
- Tái định vị (relocation)
- Một số section trong cấu trúc ELF: `.text`, `.data`, `.bss`, `.symtab`

# Lập trình mức máy tính (7)

---

- **Các topic liên quan đến ATTT**
  - Reverse Engineering (Dịch ngược)
  - Buffer overflow (Tràn bộ đệm)
  - Truy xuất bên ngoài mảng

---

**HẾT NỘI DUNG ÔN THI :>**

# Nội dung

## ■ Các chủ đề chính:

- 1) Biểu diễn các kiểu dữ liệu và các phép tính toán bit
- 2) Ngôn ngữ assembly
- 3) Điều khiển luồng trong C với assembly
- 4) Các thủ tục/hàm (procedure) trong C ở mức assembly
- 5) Biểu diễn mảng, cấu trúc dữ liệu trong C
- 6) Một số topic ATTT: reverse engineering, bufferoverflow
- 7) Linking trong biên dịch file thực thi
- 8) Phân cấp bộ nhớ, cache (tự tìm hiểu)

## ■ Lab liên quan

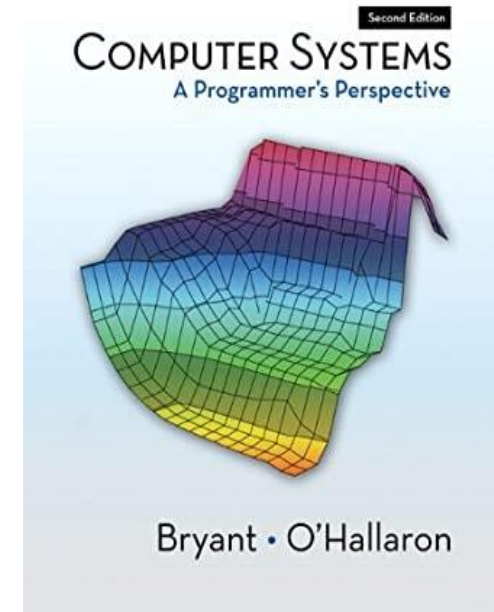
- |   |   |
|---|---|
| ▪ Lab 1: Nội dung <u>1</u>  | ▪ Lab 4: Nội dung 1, <u>2</u> , 3, <u>4</u> , 5, <u>6</u>                         |
| ▪ Lab 2: Nội dung 1, <u>2</u> , <u>3</u>                                  | ▪ Lab 5: Nội dung 1, <u>2</u> , 3, <u>4</u> , 5, <u>6</u>                         |
| ▪ Lab 3: Nội dung 1, <u>2</u> , <u>3</u> , <u>4</u> , <u>5</u> , <u>6</u> | ▪ Lab 6: Nội dung <u>1</u> , <u>2</u> , <u>3</u> , <u>4</u> , <u>5</u> , <u>6</u> |

# Giáo trình

## ■ Giáo trình chính

### ***Computer Systems: A Programmer's Perspective***

- Second Edition (CS:APP2e), Pearson, 2010
- Randal E. Bryant, David R. O'Hallaron
- <http://csapp.cs.cmu.edu>



## ■ Tài liệu khác

- *The C Programming Language*, Second Edition, Prentice Hall, 1988
  - Brian Kernighan and Dennis Ritchie
- *The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler*, 1st Edition, 2008
  - Chris Eagle
- *Reversing: Secrets of Reverse Engineering*, 1st Edition, 2011
  - Eldad Eilam



**KEEP  
CALM  
AND  
ENJOY YOUR  
SEMESTER :)**