

Thực hành
Lập trình hệ thống

GVTH: Đỗ Thị Thu Hiền
hiendtt@uit.edu.vn

Lab 2

Lập trình assembly cơ bản

Môi trường thực hành

- Operating System (OS): **Linux**
 - 32 hoặc 64 bit
 - Ubuntu, Kali, CentOS...
 - VMWare, Virtualbox...
 - Installed tools: **as, ld.**

Mục tiêu

Viết và biên dịch các chương trình **assembly đơn giản** với những lệnh đã học, để:

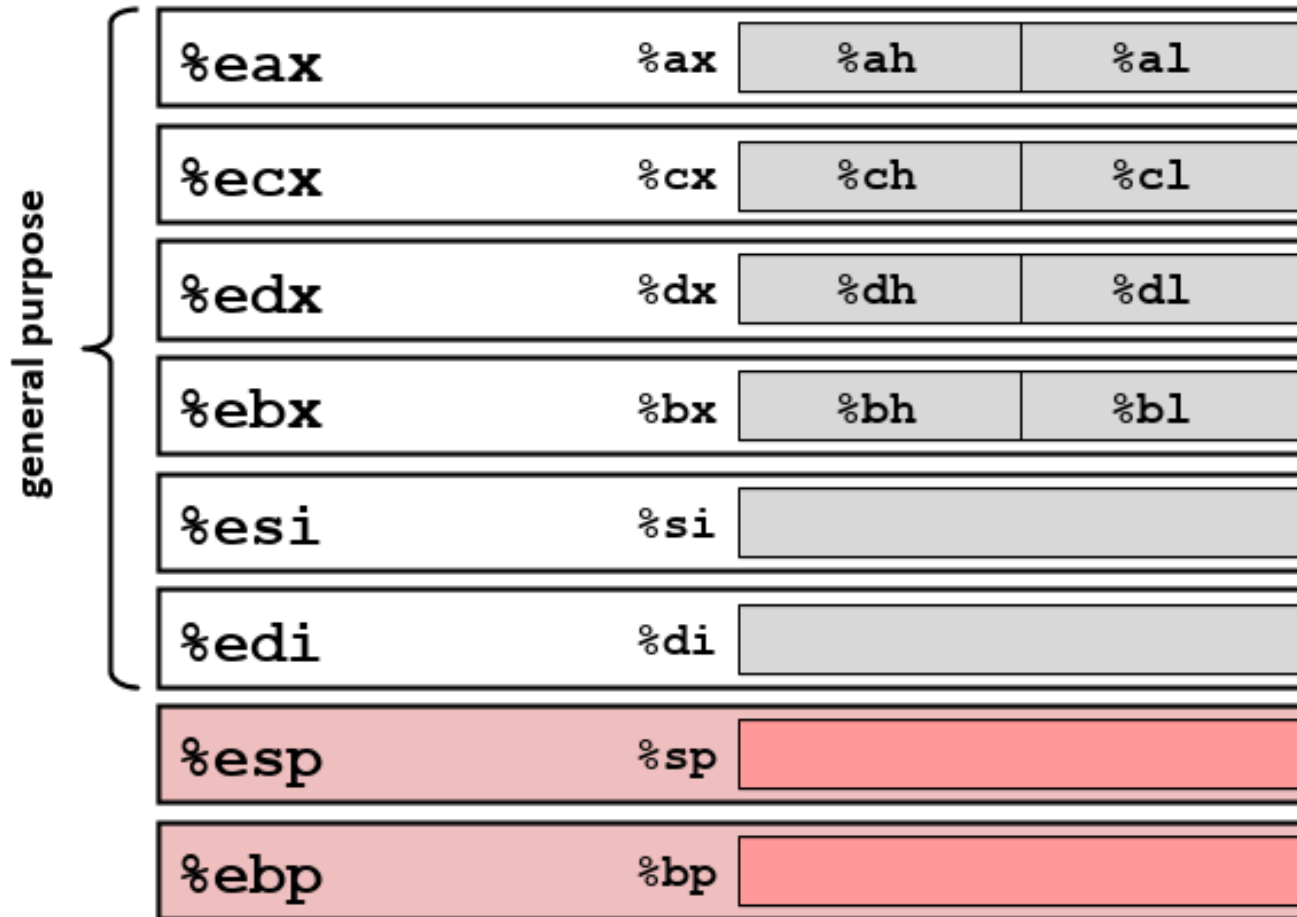
- Đọc giá trị của register và lấy dữ liệu từ memory
- Thực hiện xử lý các dữ liệu đã lấy
 - Xử lý thông thường
 - Xử lý dùng rẽ nhánh có điều kiện
- Nhập input và xuất output trên console

Nội dung Lab 2

1. Cấu trúc và cách viết một chương trình assembly
2. Biên dịch chương trình assembly đơn giản Hello World
3. Tự viết và biên dịch các chương trình assembly theo yêu cầu của bài thực hành

Ôn tập kiến thức

Register 32 bit



Ôn tập kiến thức

Định dạng assembly: AT&T

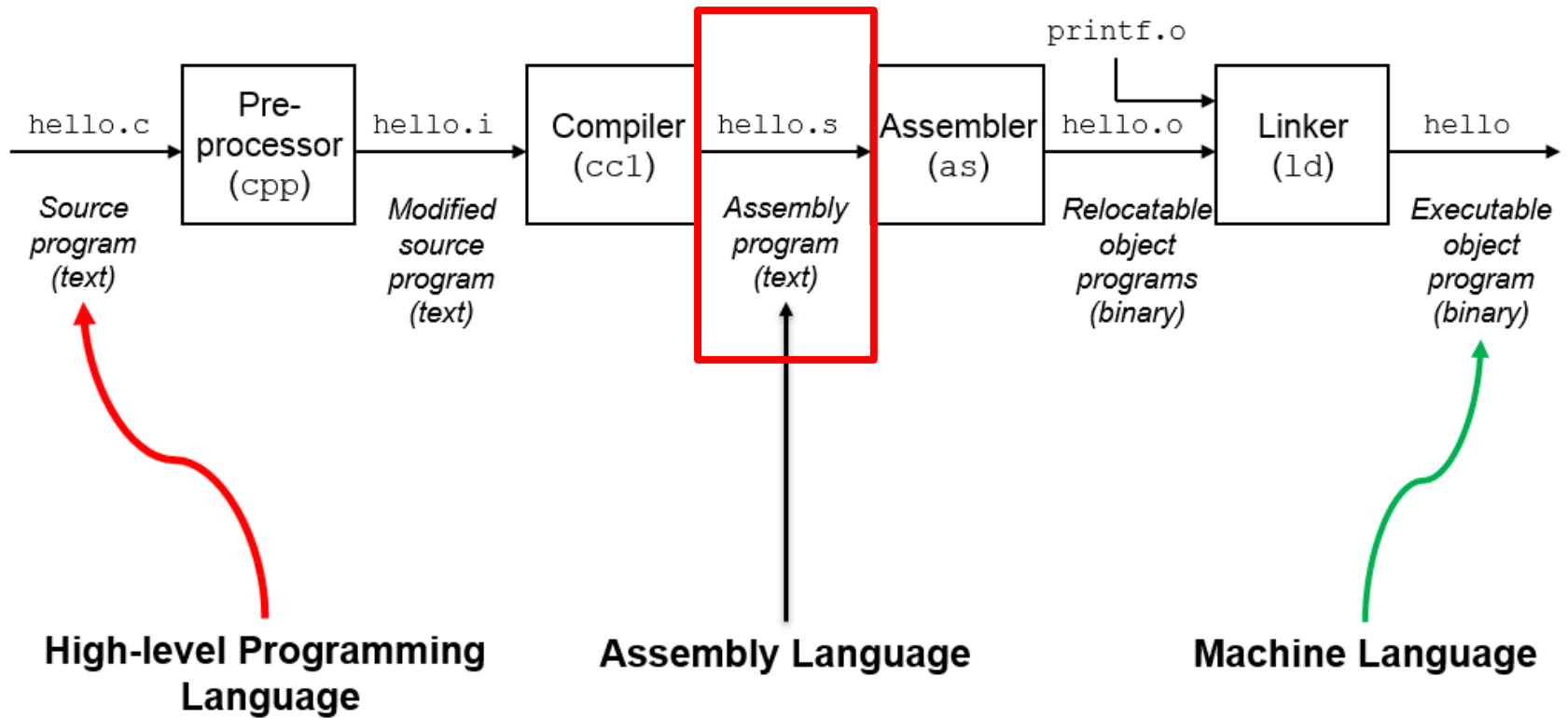
Ví dụ: Moving data: `mov [src], [dst]`



Immediates
Registers
Memories

Registers
Memories

Hợp ngữ (Assembly)



Chương trình hợp ngữ - File .s

```
1  .section .data
2  output:
3      .string "Hello, World "
4  .section .text
5      .globl _start
6  _start:
7      movl $13, %edx    ;message length
8      movl $output, %ecx ;message to write
9      movl $1, %ebx     ;file descriptor (stdout)
10     movl $4, %eax     ;system call number (sys_write)
11     int $0x80         ;call kernel
12     movl $1, %eax     ;system call number (sys_exit)
13     int $0x80         ;call kernel
14 |
```

```
.section .bss
.lcomm result, 1
```

Biến đã
khởi tạo

Code thực
thi chính
(bắt buộc)

Biến chưa
khởi tạo
(không bắt
buộc)

Chương trình hợp ngữ - File .s

- Cần khai báo “biến”?

```
.section .data
output:
.string "Hello, World "
```

```
.section .bss
.lcomm result, 1
```

- Trong **.data** hay **.bss** thực chất là các vùng nhớ được “gán” label gọi nhớ, các label này có thể dùng để truy cập đến vùng nhớ đó.
- Khi đó:

\$output: địa chỉ của ô nhớ

(output) hoặc **output**: giá trị nằm trong ô nhớ → dùng khi lưu hoặc lấy giá trị trong ô nhớ

Ví dụ:

movl \$output, %eax	#1 – lấy địa chỉ	movl \$output, %ebx	#3 – lấy địa chỉ
movl (output), %eax	#2 – lấy giá trị của ô nhớ	movl (%ebx), %eax	#4 – lấy giá trị

Chương trình hợp ngữ - File .s

- Code thực thi chính

```
4 ▾ .section .text
5   .globl _start
6 ▾ _start:
7   movl $13, %edx    ;message
8   movl $output, %ecx ;mess
9   movl $1, %ebx     ;file desc
10  movl $4, %eax      ;system ca
11  int $0x80          ;call ke
12  movl $1, %eax      ;system
13  int $0x80          ;call ke
```

Tên section chứa code

Vị trí bắt đầu code

Một số lưu ý khi viết assembly

- Phân biệt được:
 - Hằng số: **\$1**
 - Thanh ghi: **%eax**
 - Địa chỉ ô nhớ: **\$output** với output là nhãn trong .data hoặc .bss
 - Giá trị của ô nhớ: **(output)** hoặc **output**
- Nhớ vị trí của src và dst trong các câu lệnh!

Các instruction assembly

- **mov**
 - Lưu ý về suffix
- **Instruction toán học:**
 - addl
 - subl
 - idiv (?)
- **Instruction tính toán trên bit:**
 - andl, orl, sarl, sall,...
- **Rẽ nhánh có điều kiện:**
 - cmpl, jX...

Instruction để **nhập/xuất** dữ liệu?

Linux system call – Lời gọi hệ thống

- Sử dụng 4 thanh ghi để định nghĩa 1 system call
- Thực thi bằng **int \$0x80**

%eax	Name	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	int	-	-	-	-
2	sys_fork	struct pt_regs	-	-	-	-
3	sys_read	unsigned int	char *	size_t	-	-
4	sys_write	unsigned int	const char *	size_t	-	-
5	sys_open	const char *	int	int	-	-
6	sys_close	unsigned int	-	-	-	-

// Thoát chương trình

// Đọc dữ liệu

// Xuất dữ liệu

Linux system call cho nhập input/ xuất output

Thanh ghi	Ý Nghĩa	Giá trị
%eax	Mã lệnh tương ứng với hành động muốn thực hiện	3 (system_read) hoặc 4 (system_write)
%ebx	Đọc dữ liệu từ đâu hoặc xuất dữ liệu ra đâu?	0 (STDIN) cho input hoặc 1 (STDOUT) cho output
%ecx	Lưu dữ liệu vào đâu hoặc xuất dữ liệu từ đâu?	<u>Địa chỉ ô nhớ</u> lưu trữ
%edx	Độ dài của dữ liệu muốn đọc hoặc xuất	Tính bằng byte

Linux system call – Ví dụ

```
.section .data
output:
    .string "Hello, World "
.section .text
.globl _start
_start:
{   movl $13, %edx        ;message length
    movl $output, %ecx    ;message to write
    movl $1, %ebx        ;file descriptor (stdout)
    movl $4, %eax        ;system call number (sys_write)
    int $0x80            ;call kernel
    movl $1, %eax        ;system call number (sys_exit)
    int $0x80            ;call kernel
```


Biên dịch và chạy chương trình hợp ngữ

- Viết mã assembly trong file **.s**
- Sử dụng các công cụ **as**, **ld** để tạo file thực thi

```
lando@ubuntu:~/Test$ as -o example.o example.s
lando@ubuntu:~/Test$ ld -o example example.o
lando@ubuntu:~/Test$ ./example
Hello, World
lando@ubuntu:~/Test$
```

Yêu cầu 1: Thiết lập môi trường - Linux

1.1 Kiểm tra các công cụ đã cài đặt

\$ ld --version

\$ as --version

1.2 Chạy thử ví dụ Hello World

```
lando@ubuntu:~/Test$ as -o example.o example.s
lando@ubuntu:~/Test$ ld -o example example.o
lando@ubuntu:~/Test$ ./example
Hello, World
lando@ubuntu:~/Test$
```

Demo nhập/ xuất dữ liệu

- Nhập/xuất 1 chuỗi
- Nhập/xuất 1 số

Yêu cầu 2: Viết các chương trình đơn giản

- **4 chương trình** giải các bài toán đơn giản
- **1 chương trình = 1 file .s**
 - Gồm ít nhất 2 section .data và .text
 - Cần sử dụng system call để in dữ liệu ra console
 - Có thể yêu cầu nhập input → sử dụng system call
 - **Luôn luôn** dùng system call **exit** để thoát chương trình khi kết thúc.

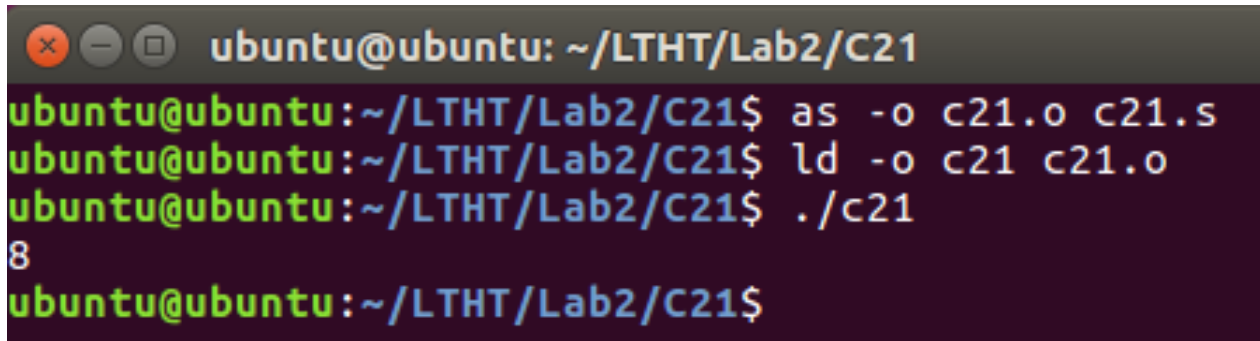
Yêu cầu 2: Viết các chương trình đơn giản

- **C2.1: In độ dài của một chuỗi cho trước**

Input: Chuỗi `msg` được khai báo sẵn trong section `.data`.

Output: Xuất ra màn hình độ dài của chuỗi (số ký tự - không tính ký tự null).

Giới hạn: Chuỗi có độ dài tối đa 9 ký tự



```
ubuntu@ubuntu: ~/LTHT/Lab2/C21
ubuntu@ubuntu:~/LTHT/Lab2/C21$ as -o c21.o c21.s
ubuntu@ubuntu:~/LTHT/Lab2/C21$ ld -o c21 c21.o
ubuntu@ubuntu:~/LTHT/Lab2/C21$ ./c21
8
ubuntu@ubuntu:~/LTHT/Lab2/C21$
```

Với chuỗi “**Love UIT**”

Yêu cầu 2: Viết các chương trình đơn giản

- **C2.2:** tính giá trị trung bình cộng của 4 số (1 chữ số)

Input: 4 số nguyên (1 chữ số) nhập vào từ bàn phím

Output: Giá trị trung bình cộng (lấy phần nguyên) của 4 số đã nhập.

Yêu cầu: Các số a, b, c, d nguyên và < 10 .

```
ubuntu@ubuntu: ~/LTHT/Lab2/C22
ubuntu@ubuntu:~/LTHT/Lab2/C22$ as -o c22.o c22.s
ubuntu@ubuntu:~/LTHT/Lab2/C22$ ld -o c22 c22.o
ubuntu@ubuntu:~/LTHT/Lab2/C22$ ./c22
Enter a number (1-digit): 2
Enter a number (1-digit): 3
Enter a number (1-digit): 5
Enter a number (1-digit): 7
4
ubuntu@ubuntu:~/LTHT/Lab2/C22$
```

Yêu cầu 2: Viết các chương trình đơn giản

- C2.3: Kiểm tra 1 ký tự nhập vào là chữ hoa hay chữ thường**

Input: Nhập vào 1 ký tự từ bàn phím

Output: Nhận định “Chu hoa” cho chữ in hoa hoặc “Chu thuong” cho chữ thường

```
ubuntu@ubuntu: ~/LTHT/Lab2/C23
ubuntu@ubuntu:~/LTHT/Lab2/C23$ as -o c23.o c23.s
ubuntu@ubuntu:~/LTHT/Lab2/C23$ ld -o c23 c23.o
ubuntu@ubuntu:~/LTHT/Lab2/C23$ ./c23
Enter a character: h
Chu thuong
ubuntu@ubuntu:~/LTHT/Lab2/C23$ ./c23
Enter a character: K
Chu hoa
ubuntu@ubuntu:~/LTHT/Lab2/C23$
```

Yêu cầu 2: Viết các chương trình đơn giản

- **C2.4:** Kiểm tra tính tăng dần của các chữ số trong 1 số 3 chữ số

Input: 1 số có 3 chữ số nhập từ bàn phím.

Output: Nhận định “Tang dan” hoặc “Khong tang dan”.

```
ubuntu@ubuntu: ~/LTHT/Lab2/C24
ubuntu@ubuntu:~/LTHT/Lab2/C24$ as -o c24.o c24.s
ubuntu@ubuntu:~/LTHT/Lab2/C24$ ld -o c24 c24.o
ubuntu@ubuntu:~/LTHT/Lab2/C24$ ./c24
Enter a number (3-digit): 123
Tang dan
ubuntu@ubuntu:~/LTHT/Lab2/C24$ ./c24
Enter a number (3-digit): 115
Tang dan
ubuntu@ubuntu:~/LTHT/Lab2/C24$ ./c24
Enter a number (3-digit): 362
Khong tang dan
ubuntu@ubuntu:~/LTHT/Lab2/C24$
```


Một số lưu ý khi viết assembly

- Khi sử dụng system call **sys_write** và **sys_read** để ghi output hoặc đọc input:
 - **%ecx** phải là **địa chỉ ô nhớ**
 - **Được khai báo trong .data hoặc .bss**
- Khi in ra console: đầu ra luôn là ký tự/chuỗi
 - Hệ thống tự động xem giá trị cần in là mã ascii của 1 ký tự: giá trị 0 → in ra ký tự có mã ascii là 0.
 - Vậy muốn in số 1 → cần truyền vào mã ascii của ký tự số '1'
 - Xem phần D.2
- Khi nhận input từ console:
 - Nhận vào luôn là ký tự/chuỗi
 - Nếu nhập ký tự số cần có bước chuyển giá trị số nguyên tương ứng để dùng khi tính toán
 - Xem phần D.2

Gợi ý cho từng chương trình

C2.1 In độ dài chuỗi

- Xem gợi ý **D.3** để tìm độ dài của một chuỗi.

rs:

```
.string "hello world"
```

len = . -rs

len là **1 hằng số!!** (dù nằm trong .data)

- Để xuất một dữ liệu nào đó ra màn hình, cần đảm bảo:
 - Dữ liệu đang **nằm trong 1 ô nhớ** (trong section .bss hoặc .data)
 - Ta cần thiết lập giá trị các thanh ghi cho **1 system call xuất dữ liệu**
 - Hệ thống xem giá trị trong ô nhớ là mã ASCII của 1 ký tự để in.

Gợi ý cho từng chương trình

C2.2 tính giá trị trung bình cộng của 4 số (1 chữ số)

- Nhập 4 số (1 chữ số)
 - Khai báo các vùng nhớ **input** trong **.bss** để lưu 4 số
 - Sử dụng system call `sys_read` để đọc dữ liệu
- Xử lý:
 - Cần chuyển các ký tự số sang giá trị số nguyên để tính toán
 - Lấy dữ liệu đã nhập và tính tổng 4 số
 - **Chia trung bình?**
 - Phép tính toán bit?
 - Phép chia với **div**

Gợi ý cho từng chương trình

C2.3 Kiểm tra 1 ký tự nhập vào là chữ hoa hay chữ thường

- Xem bảng mã ASCII để biết 1 ký tự thường hoặc hoa có giá trị nằm trong khoảng nào.
- Ví dụ ký tự thường x : $'a' \leq x \leq 'z'$

Gợi ý cho từng chương trình

C2.4 Kiểm tra tính tăng dần của các chữ số trong 1 số 3 chữ số

- Lấy từng chữ số trong số đã nhập
 - Số vừa nhập thực chất là một **chuỗi các ký tự số**
 - Truy xuất từng ký tự trong chuỗi: lấy từng ký tự (từng byte) từ vùng nhớ **input**

```
movl $input, %eax
```

```
mov 1(%eax), %bl
```

- Lấy giá trị và kiểm tra ký tự trong %bl
- So sánh từng cặp chữ số liền kề

Yêu cầu

- **Mỗi chương trình** được code riêng trong **1 file .s**.
Mỗi chương trình cần có comment chức năng của các câu lệnh quan trọng. Nếu không sẽ bị xem là **sao chép!**

Nộp bài

- Thực hiện theo **nhóm tối đa 2 sinh viên**.
- Yêu cầu: Hoàn thành ít nhất **C2.1 – C2.2** trên lớp.
- Báo cáo (demo) trực tiếp hoặc nộp file nén các file **.s** trên moodle.