

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...
 - Công cụ: **as, ld.**

Mục tiêu

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

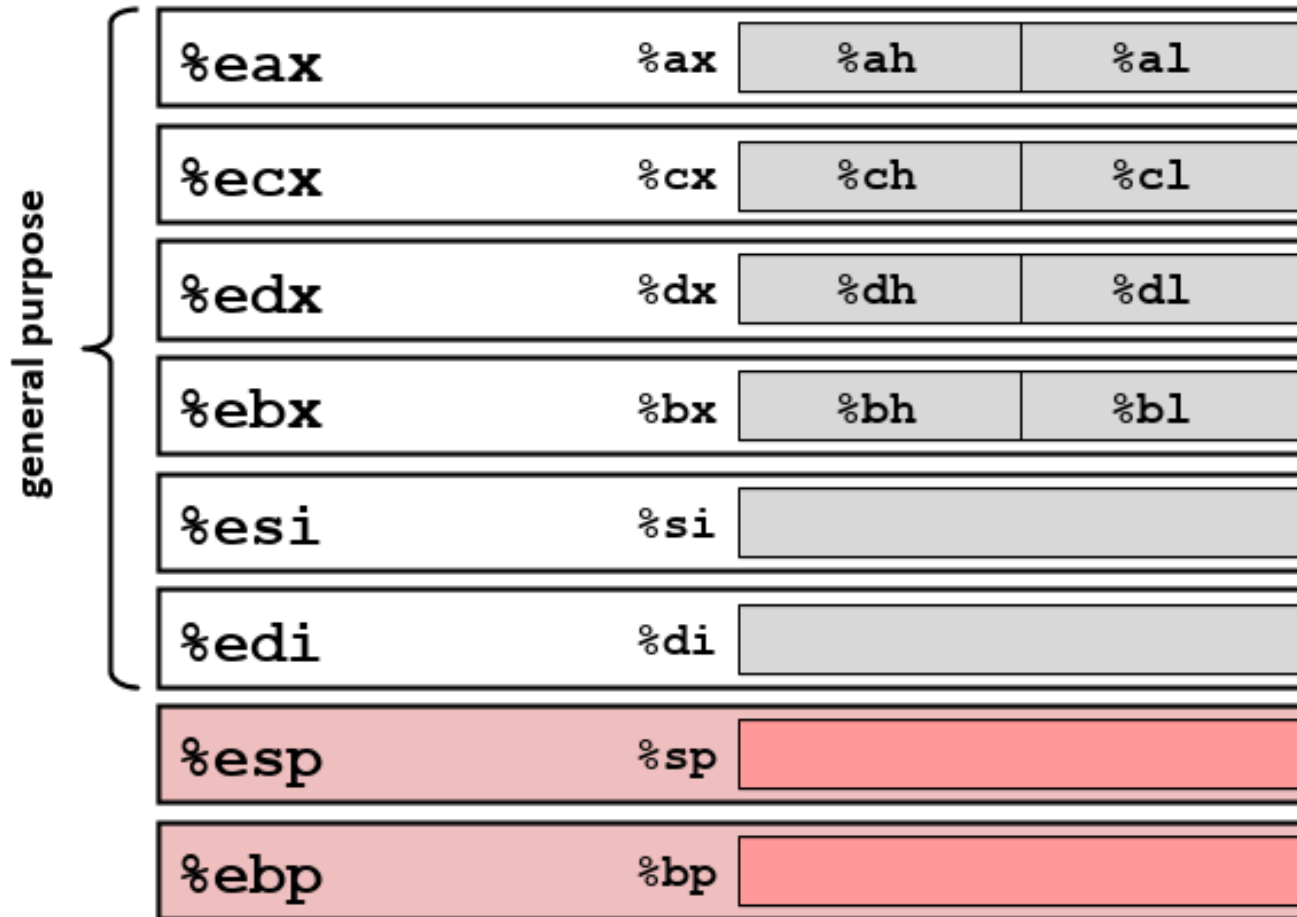
- Đọc giá trị của thanh ghi/lấy dữ liệu từ bộ nhớ
- Thực hiện xử lý các dữ liệu đã lấy
 - Xử lý thông thường
 - Xử lý dùng rẽ nhánh có điều kiện
- Nhập input và xuất output trên console

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`

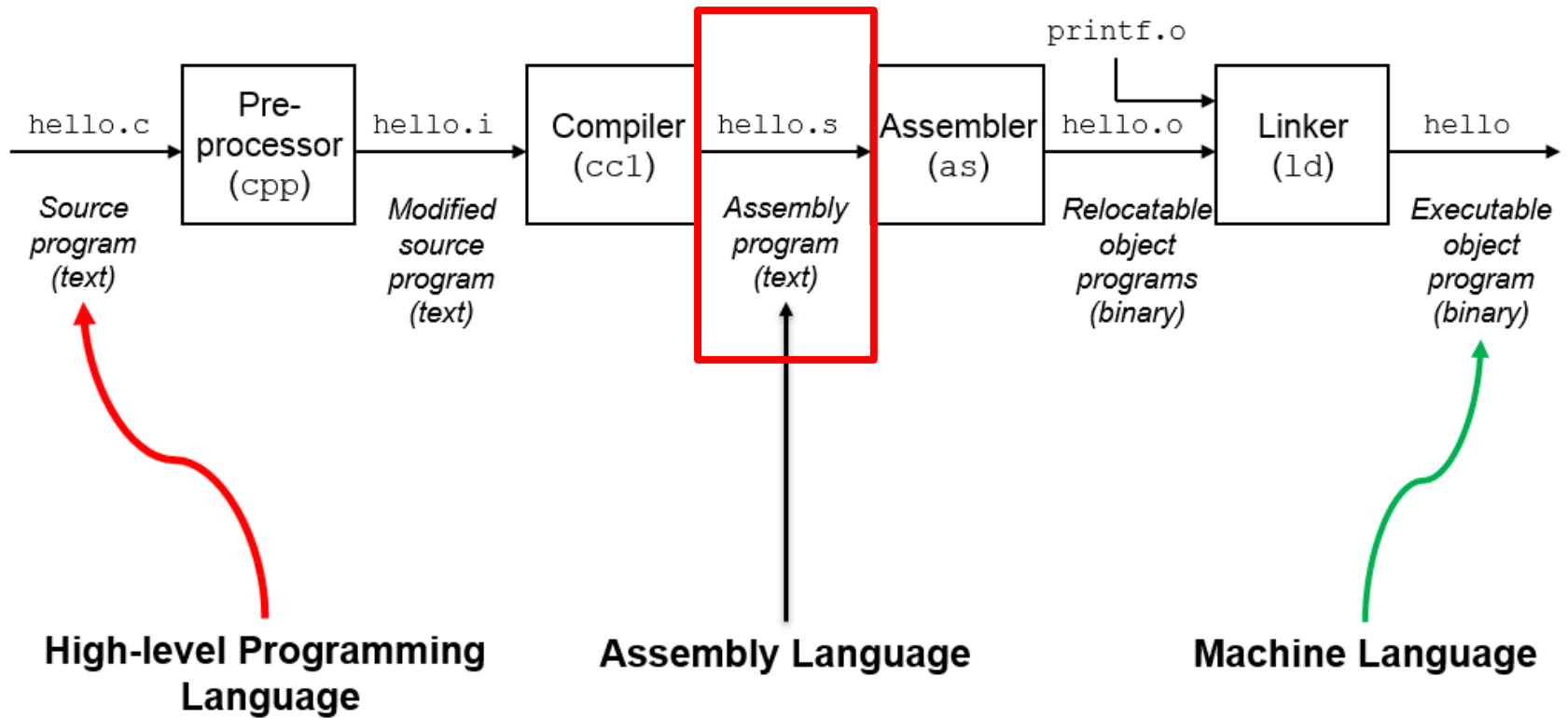


Immediates
Registers
Memories



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**”

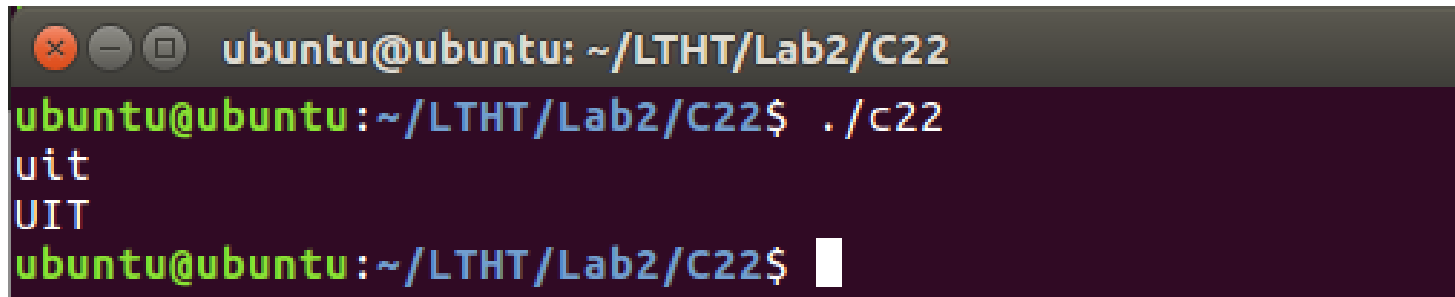
Yêu cầu 2: Viết các chương trình đơn giản

- **C2.2:** Chuyển đổi chuỗi chữ thường sang chữ in hoa

Input: Chuỗi in thường có 3 chữ cái nhập từ bàn phím.

Output: Xuất ra màn hình chuỗi chứa các ký tự đã được chuyển sang chữ in hoa.

Yêu cầu: Sử dụng tối đa 02 lần in ra màn hình.

A terminal window with a dark background and light-colored text. The window title bar shows standard Ubuntu window controls (close, maximize, minimize) and the text 'ubuntu@ubuntu: ~/LTHT/Lab2/C22'. The terminal content shows a prompt 'ubuntu@ubuntu:~/LTHT/Lab2/C22\$' followed by the command './c22'. The program outputs 'uit' on the next line and 'UIT' on the following line. The prompt returns to 'ubuntu@ubuntu:~/LTHT/Lab2/C22\$' with a cursor.

```
ubuntu@ubuntu: ~/LTHT/Lab2/C22
ubuntu@ubuntu:~/LTHT/Lab2/C22$ ./c22
uit
UIT
ubuntu@ubuntu:~/LTHT/Lab2/C22$
```

Yêu cầu 2: Viết các chương trình đơn giản

- **C2.3:** Kiểm tra tính giảm 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 “Giảm dần” hoặc “Không giảm dần”.

```
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 number (3-digit): 321
Giảm dần
ubuntu@ubuntu:~/LTHT/Lab2/C23$ ./c23
Enter a number (3-digit): 124
Không giảm dần
ubuntu@ubuntu:~/LTHT/Lab2/C23$ ./c23
Enter a number (3-digit): 433
Giảm dần
ubuntu@ubuntu:~/LTHT/Lab2/C23$ ./c23
Enter a number (3-digit): 326
Không giảm dần
ubuntu@ubuntu:~/LTHT/Lab2/C23$
```

Yêu cầu 2: Viết các chương trình đơn giản

- C2.4: Kiểm tra học lực dựa trên điểm số (thang điểm 100)**

Input: Điểm số a ($10 \leq a \leq 99$).

Output: $a > 80 \rightarrow$ “Gioi”

$55 \leq a < 80 \rightarrow$ “Dat”

$a < 55 \rightarrow$ “Khong dat”

```
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 grade (2-digit): 95
Gioi
ubuntu@ubuntu:~/LTHT/Lab2/C24$ ./c24
Enter a grade (2-digit): 55
Dat
ubuntu@ubuntu:~/LTHT/Lab2/C24$ ./c24
Enter a grade (2-digit): 16
Khong dat
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 Chuyển chữ thường sang chữ hoa

- Sử dụng system call **sys_read** để đọc ký tự
 - Lưu vào vùng nhớ **input**
- Gợi ý:
 - Lấy từng ký tự (từng byte) từ vùng nhớ **input**

```
movl $input, %eax  
mov 1(%eax), %bl
```
 - Xử lý từng ký tự để chuyển sang chữ hoa: xử lý giá trị trong **%bl**
- Chuyển sang chữ hoa?? → xem mã ASCII

Gợi ý cho từng chương trình

C2.3 Kiểm tra tính giảm 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 → Nên chuyển về số nguyên

Gợi ý cho từng chương trình

C2.4 Kiểm tra học lực trên điểm số

- Sử dụng system call **sys_read** để đọc ký tự
 - Lưu vào vùng nhớ **input**
- Số vừa nhập thực chất là một **chuỗi các ký tự số**
 - Làm sao so sánh với các giá trị 80, 55?
 - '15' ?? 55

→ Cần có bước chuyển chuỗi số thành giá trị số nguyên → Xem **D.2**

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.
- Nộp file nén các file **.s** trên moodle.