

TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



**Hướng dẫn thực hành**  
**Môn: Kiến trúc máy tính - CO2008**  
**MARS**

# Nội dung

<b>1</b>	<b>Download và cài đặt MARS.</b>	<b>3</b>
<b>2</b>	<b>Các thao tác căn bản với MARS</b>	<b>3</b>
2.1	Khởi động MARS . . . . .	3
2.2	Thanh công cụ . . . . .	3
2.3	Tạo mới hoặc mở một file asm . . . . .	4
<b>3</b>	<b>Các kiểu dữ liệu</b>	<b>4</b>
<b>4</b>	<b>Tương tác với người dùng bằng syscall.</b>	<b>6</b>
4.1	Dùng thực thi chương trình . . . . .	6
4.2	Hiển thị số nguyên . . . . .	6
4.3	Hiển thị số thực . . . . .	6
4.4	Hiển thị chuỗi . . . . .	6
4.5	Nhập một số nguyên . . . . .	6
4.6	Nhập số thực . . . . .	6
4.7	Hiển thị ký tự . . . . .	6
4.8	Đọc một ký tự . . . . .	7
4.9	Nhập chuỗi . . . . .	7

# 1 Download và cài đặt MARS.

MARS simulator là công cụ dùng để soạn thảo và chạy mô phỏng chương trình hợp ngữ MIPS. Để cài đặt MARS chúng ta download file MARS\_[version 4.5].jar từ link: <http://courses.missouristate.edu/KenVollmar/mars/download.htm>

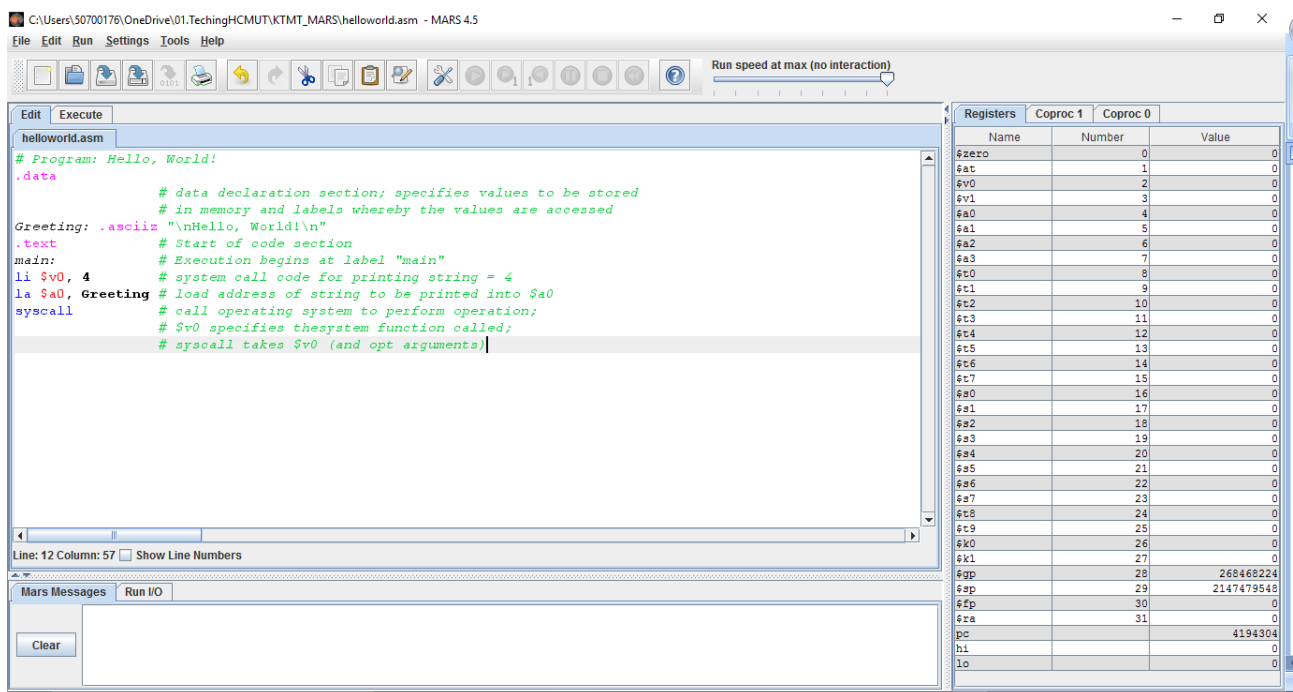
Chú ý: file jar là file java thực thi, do đó khi đã cài java trên máy thì chỉ cần click vào file MARS\_[version 4.5].jar là có thể chạy được. **Không bung nén file!!!**

MARS yêu cầu phải có java JRE 1.5 (hoặc các phiên bản sau). Link downloads JRE (Java Runtime Environment) <http://www.oracle.com/technetwork/java/javase/downloads/index.html> chọn JRE, sau đó chọn hệ điều hành tương ứng – chú ý phân biệt 64 bit (x64) và 32 bit (x86)

## 2 Các thao tác căn bản với MARS

### 2.1 Khởi động MARS






Sau khi download và cài đặt xong JRE, chúng ta khởi động MARS bằng cách click vào file MARS\_[version 4.5].jar. Khi đó MARS sẽ có giao diện như hình dưới:





Hình. 1: Giao diện công cụ mô phỏng hợp ngữ MIPS

### 2.2 Thanh công cụ

- Tạo một file mới.
- Mở file đã tồn tại.
- Lưu file (chú ý là phải có .asm)
- Biên dịch file hợp ngữ.
- Điều chỉnh tốc độ thực thi (lệnh thực thi/thời gian) chức năng này thích hợp cho việc quan sát, debug chương trình đang chạy.
- Thực thi chương trình
- Thực thi từng bước chương trình



-  Reset chương trình.
-  Lùi lại lệnh trước.
-  Dừng chương trình đang chạy.
-  Kết thúc chương trình đang chạy.
-  Help, sv tham khảo các chức năng, lệnh hợp ngữ, và code mẫu.

## 2.3 Tạo mới hoặc mở một file asm

- Tạo file mới bằng tổ hợp phím “**Ctrl + N**” hoặc bấm vào biểu tượng “new file”  hoặc vào menu **File** -> **New** để tạo ra 1 file asm mới.
- Mở file bằng tổ hợp phím “**Ctrl + O**” hoặc bấm vào biểu tượng “open file”  hoặc vào menu **File** -> **Open** để mở 1 file .asm đã có.
- Bây giờ chúng ta có thể chỉnh sửa mã nguồn ở vùng soạn thảo (Hình 1). Nhập mã chương trình Hello-World vào vùng soạn thảo như đoạn chương trình bên dưới.

```
# Program: Hello, World!
.data
    # data declaration section; specifies values to be stored
    # in memory and labels whereby the values are accessed
Greeting: .asciiz "\nHello, World!\n"

.text
    # Start of code section
main:      # Execution begins at label "main"
    li $v0, 4      # system call code for printing string = 4
    la $a0, Greeting # load address of string to be printed into $a0
    syscall        # call operating system to perform operation;
                  # $v0 specifies the system function called;
                  # syscall takes $v0 (and opt arguments)
```

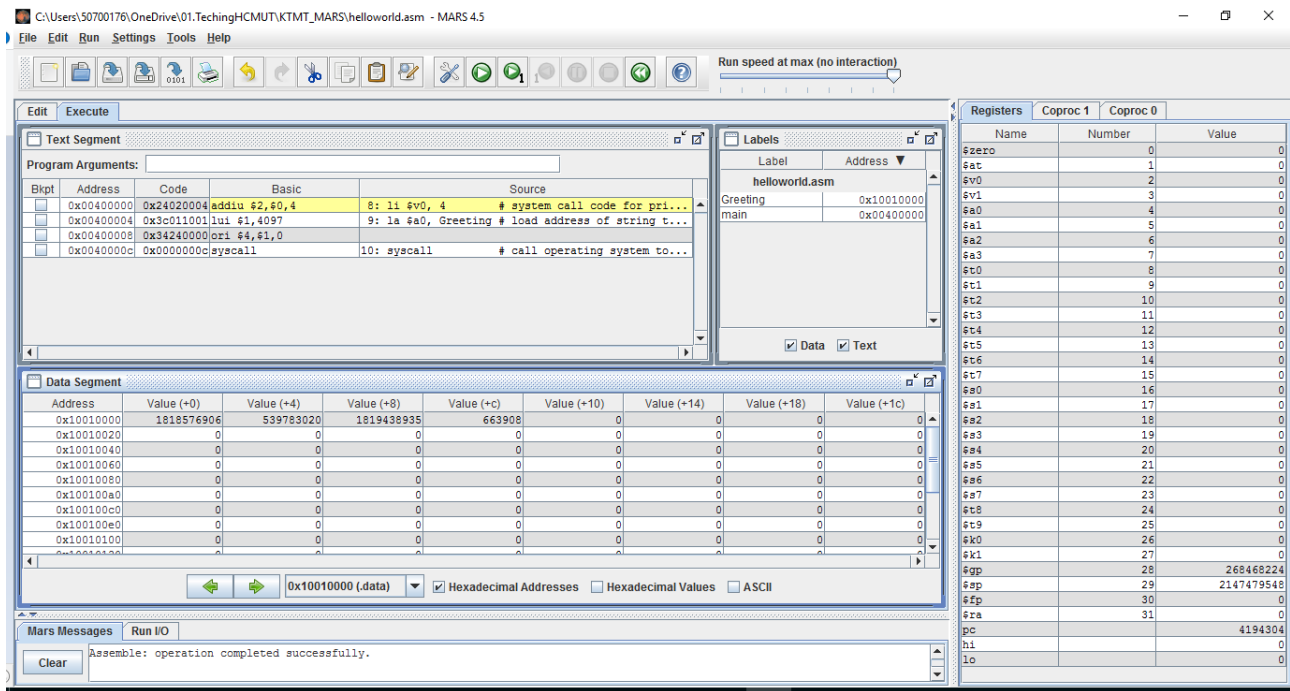
- Lưu lại với tên Hello-World.asm.
- Biên dịch chương trình bằng cách bấm vào biểu tượng  hoặc bấm **F3** hoặc vào menu **Run** -> **Assemble**. Sau khi biên dịch thành công, chương trình sẽ xuất hiện thông báo “**Assemble: operation completed successfully.**” ở vùng hiển thị thông báo; đồng thời chương trình sẽ tự động chuyển sang giao diện thực thi như Hình 2.
- Thực thi chương trình bằng cách bấm vào biểu tượng  hoặc bấm **F5**. Sau khi chạy thành công thì chương trình sẽ xuất hiện thông báo “**Hello, World!**” ở vùng hiển thị thông báo
- **Yêu cầu:** sinh viên thực hiện chạy lại từng bước (step by step ) chương trình Hello-World, quan sát sự thay đổi thanh ghi. Chuyển dạng hiển thị thì từ số hex sang số thập phân và ngược lại.

## 3 Các kiểu dữ liệu

File hợp ngữ chia làm 2 phần chính:

```
.data
    \\ phần data để chứa dữ liệu
.text
    \\ phần text để chứa code
```

Yêu cầu tất cả các file code phải có đủ 2 phần **.data** và **.text**.



Hình. 2: Giao diện thực thi MARS.

## .data - Vùng dữ liệu

Các khai báo sau đây nằm trong vùng dữ liệu **.data**.

- Khai báo 1 integer hoặc 1 word (4 byte trong kiến trúc 32 bit):

```
integerA: .word 12345678
```

integerA là nhãn, dùng để đánh dấu vị trí, nhãn có thể có hoặc không.

- Khai báo 1 byte:

```
byteA: .byte 123
```

- Khai báo half word hoặc (2 byte):

```
halfwordA: .half 12345
```

- Khai báo 1 chuỗi ascii không có ký tự kết thúc chuỗi ở cuối chuỗi:

```
strA: .ascii "Welcome to CSE HCMUT"
```

- Khai báo 1 chuỗi ascii với ký tự kết thúc ở cuối chuỗi:

```
strB: .asciiz "Welcome to CSE HCMUT"
```

- Khai báo 1 số thực:

```
floatA: .float 12.345
```

- Khai báo không gian trống 100 byte:

```
space100: .space 100
```

- Khai báo 1 dãy 5 integer hoặc 5 word (4 byte):

```
intergerArray: .word 123582, 12328, 3233, 21233, 487675
```

- Khởi tạo 1 dãy 5 integer cùng giá trị.

```
integerArray: .word 123:5
```

- Cách khai báo mảng của các kiểu dữ liệu khác tương tự như cách khai báo đã nêu ở trên

## 4 Tương tác với người dùng bằng syscall.

Dưới đây là các chức năng cơ bản mà lệnh syscall cung cấp. Các lệnh ở ví dụ dưới viết trong phần **.text**. Để tìm hiểu rõ hơn về lệnh syscall và phần **help/syscall** của phần mềm MARS hoặc tìm trên Internet.

### 4.1 Dừng thực thi chương trình

```
li $v0, 10    #terminate execution
syscall
```

### 4.2 Hiển thị số nguyên

Muốn hiển thị 1 số nguyên thì gán số nguyên đó vào thanh ghi \$a0 và khởi tạo code 1 cho thanh ghi \$v0.

```
addi $a0, $0, 100    # Assign an integer to a0
li $v0, 1             # Print integer a0
syscall
```

### 4.3 Hiển thị số thực

Muốn hiển thị 1 số thực thì gán số thực đó vào thanh ghi \$f12 và khởi tạo \$v0 = 2

```
mov.s $f12, $f3      # Move contents of register $f3 to register $f12
li $v0, 2             # Print float number
syscall
```

### 4.4 Hiển thị chuỗi

Xem ví dụ hello-world ở phần trên.

### 4.5 Nhập một số nguyên

```
li $v0, 5             # Get integer mode,
                      # $v0 contains integer read
syscall
add $a0, $0, $v0      # Move integer from $v0 to register $a0
```

### 4.6 Nhập số thực

```
li $v0, 6             # Get float mode
syscall               # $f0 contains float read
```

### 4.7 Hiển thị ký tự

```
addi $a0, $0, 'A'     # Display character 'A'
li $v0, 11            # print char
syscall
```

## 4.8 Đọc một ký tự

```
li    $v0, 12           # Get character mode
                        # $v0 contains character read
syscall
add   $a0, $0, $v0      # Move character to register
```

## 4.9 Nhập chuỗi

```
.data
strIn: .space 100      #create 100 bytes space

.text
main:
    la    $a0, strIn    # Get address to store string
    addi   $a1, $0, 10   # Input 10 characters into string (includes end of string)
    li     $v0, 8        # Get string mode
    syscall
```