

# Thực hành Kiến trúc máy tính

Giảng viên: Nguyễn Thị Thanh Nga  
Khoa Kỹ thuật máy tính  
Trường CNTT&TT

# Tuần 1

---

- Giới thiệu công cụ mô phỏng MARS
- Hướng dẫn cài đặt
- Giao diện lập trình IDE cơ bản
- Lập trình và tìm hiểu công cụ lập trình với chương trình “**HelloWorld**”
- Chạy giả lập chương trình mô phỏng
- Tra cứu HELP

# Mục tiêu

---

- Có khả năng cài đặt được công cụ MARS
- Viết một chương trình đơn giản để kiểm tra các tính năng của công cụ mô phỏng MARS như:
  - Lập trình bằng hợp ngữ
  - Chạy giả lập
  - Gỡ rối
  - ...

để hiểu rõ hơn về bản chất và các hoạt động thực sự xảy ra trong bộ xử lý MIPS

# Công cụ giả lập MIPS IT

---

- MIPS IT: là viết tắt của “Microprocessor without Interlocked Pipeline Stages”
- Là một kiến trúc tập lệnh RISC (Reduced Instruction Set Computer) được phát triển bởi MIPS Technologies.
- Năm 1981, John L.Hennessy bắt đầu nghiên cứu về bộ xử lý MIPS đầu tiên tại Stanford University
- Yêu cầu các câu lệnh phải hoàn thành trong 1 chu kỳ máy

# Công cụ giả lập MIPS IT

---

Một số ứng dụng:

- Pioneer DVR-57-H
- Kenwood HDV-810 Car Navigation System
- HP Color Laser Jet 2500 Printer
- 3COM IP phone, cordless phone
- EOS 10D digital camera
- Sony Play station PSX and High Definition Television
- Samsung Digital Photo Frame
- Sony Media Server Vaio VGX-X90P
- Pioneer Plasma Television

# Cài đặt

- Tải về Java Runtime Enviroment, JRE, để chạy công cụ MARS

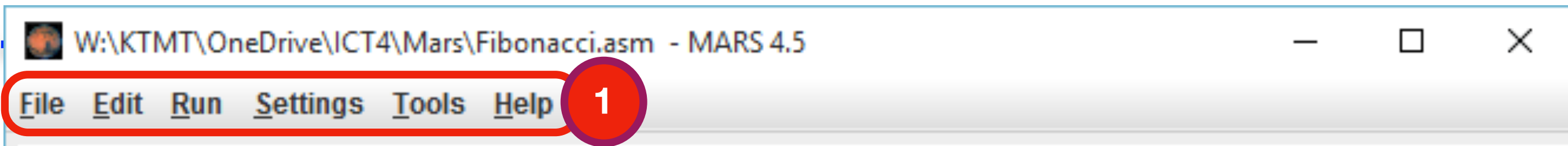
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

- Cài đặt JRE
- Tải về công cụ MARS, bao gồm:
  - Phiên bản mới nhất của MARS, và nên lấy thêm 2 tài liệu
  - MARS features
  - MARS tutorial

ở URL <http://courses.missouristate.edu/KenVollmar/MARS/download.htm>

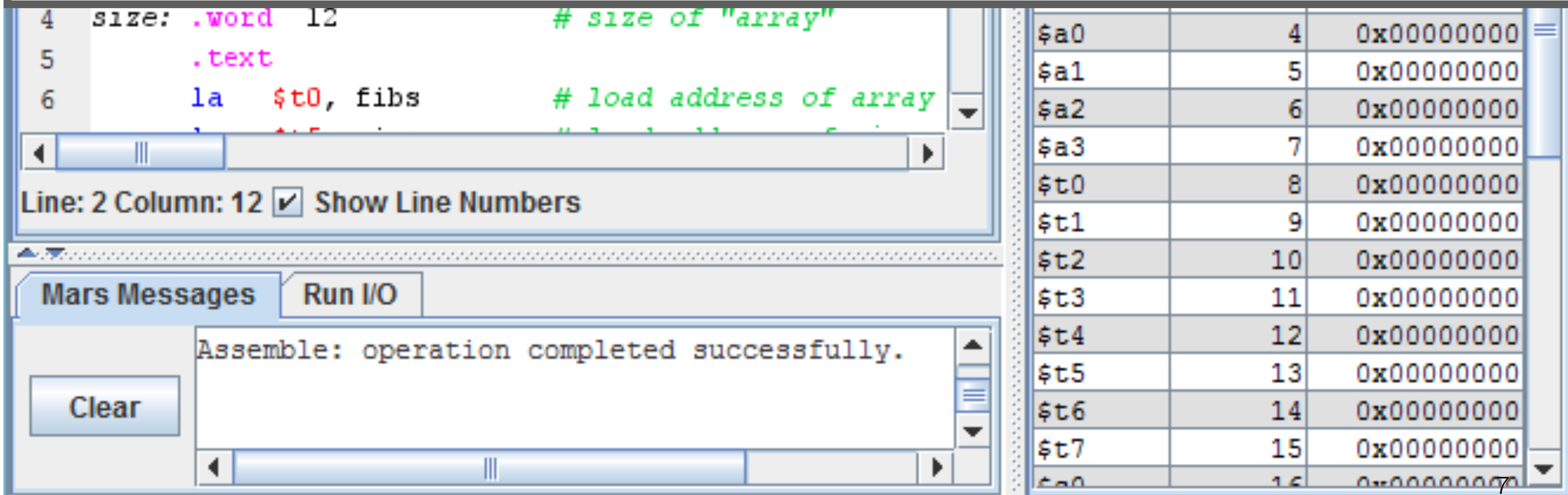
Công cụ MARS có thể thực hiện ngay mà không cần cài đặt. Click đúp vào file Mars.jar để chạy.

# Giao diện lập trình IDE cơ bản

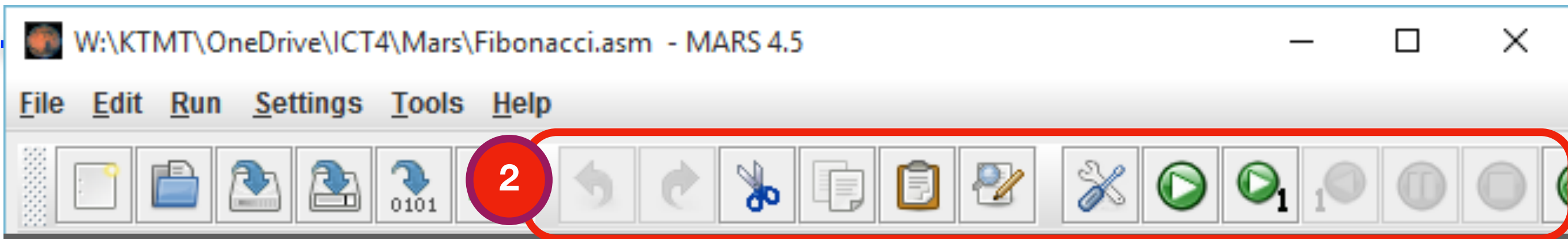


## 1 Menu

- Hầu hết các mục trong menu đều có các icon tương ứng
- Di chuyển chuột lên phía trên của icon, chức năng tương ứng sẽ hiển thị.
- Các mục trong menu cũng có phím tắt tương ứng.



# Giao diện lập trình IDE cơ bản

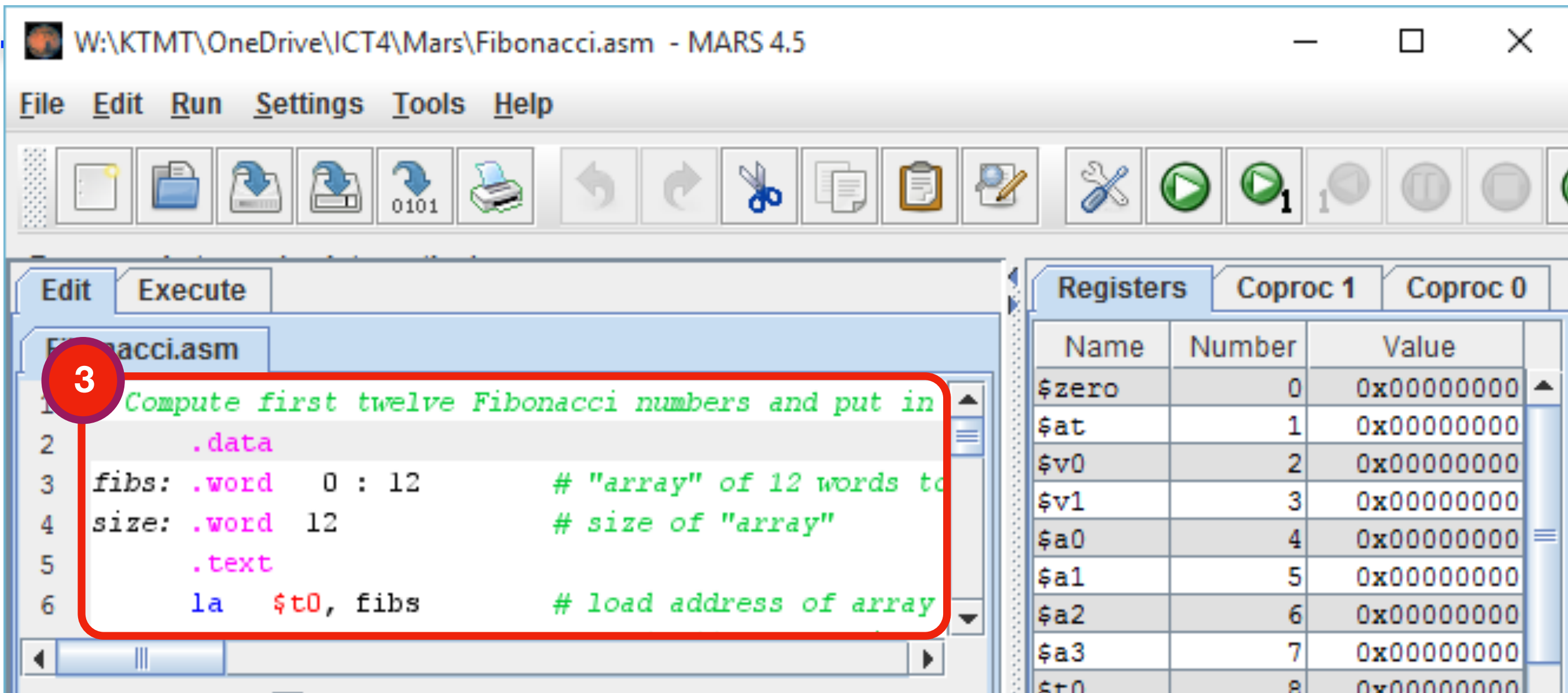


## 2 Toolbar

- Chứa một vài tính năng soạn thảo cơ bản như: copy, paste, open.
- Các tính năng gỡ rối (trong hình chữ nhật màu đỏ)
  - Run: chạy toàn bộ chương trình
  - Run one step at a time: chạy từng lệnh và dừng (rất hữu ích)
  - Undo the last step: khôi phục lại trạng thái ở lệnh trước đó (rất hữu ích)
  - Pause: tạm dừng quá trình chạy toàn bộ (Run)
  - Stop: kết thúc quá trình gỡ rối
  - Reset MIPS memory and register: Khởi động lại



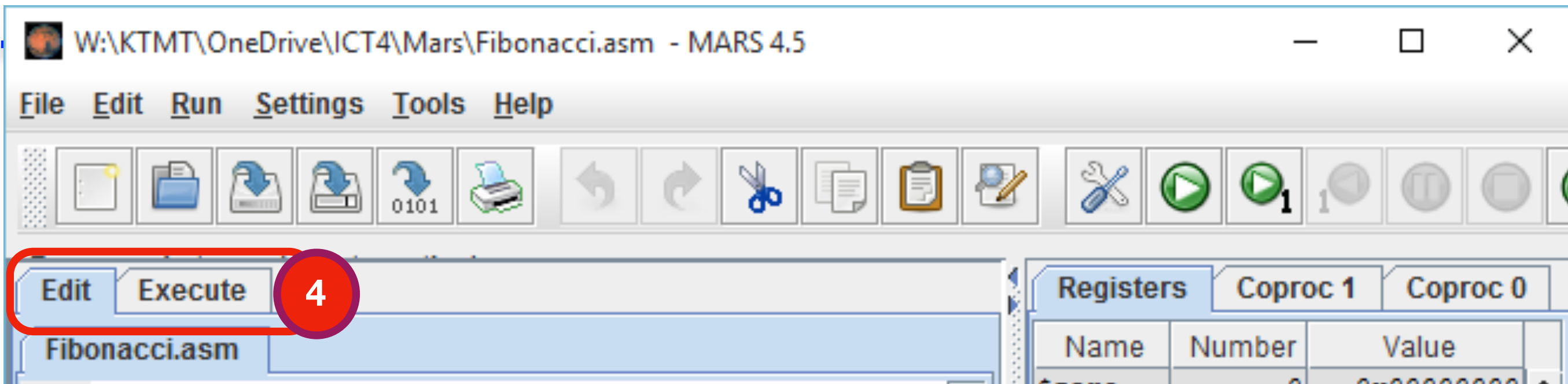
# Giao diện lập trình IDE cơ bản



## 3 Edit tab

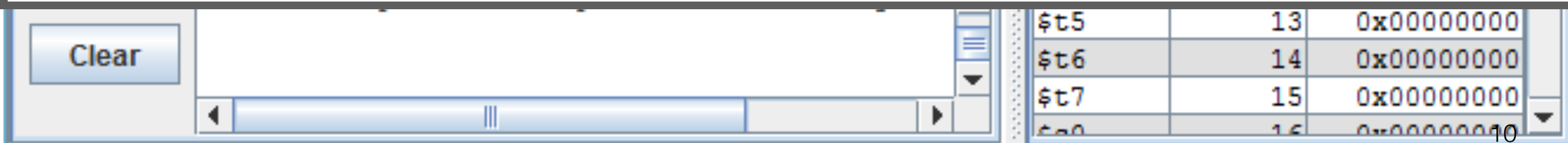
- Soạn thảo văn bản tích hợp sẵn với tính năng **tô màu theo cú pháp**, giúp người dùng dễ theo dõi mã nguồn.
- Khi gõ lệnh mà chưa hoàn tất, một cửa sổ bật lên để trợ giúp.
- Vào menu Settings / Editor... để thay đổi màu sắc, font...

# Giao diện lập trình IDE cơ bản



## 4 Edit/Execute

- Mỗi file mã nguồn ở giao diện soạn thảo có 2 cửa sổ - 2 tab: Edit và Execute
  - Edit tab:** viết chương trình hợp ngữ với tính năng **tô màu theo cú pháp**.
  - Execute tab:** biên dịch chương trình hợp ngữ đã viết ở Edit tab thành mã máy, **chạy và gỡ rối**.



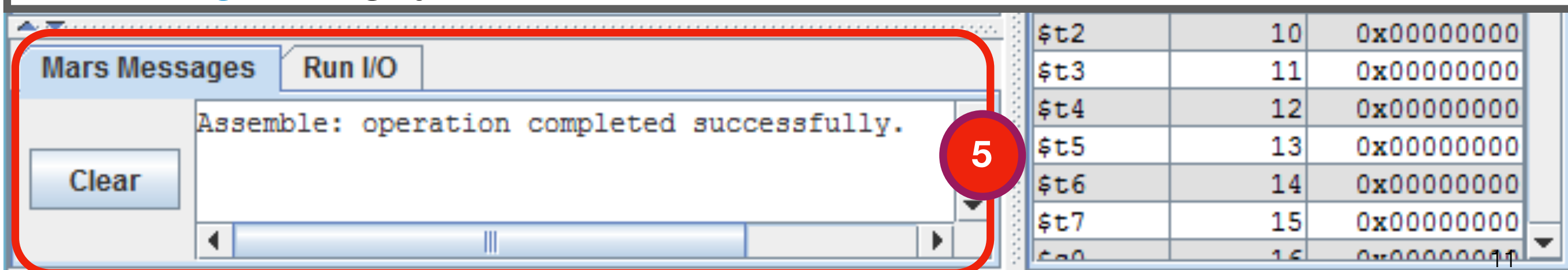
# Giao diện lập trình IDE cơ bản

**5 Message Areas:** Có 2 cửa sổ message ở cạnh dưới của giao diện IDE

- **The *Run I/O* tab** chỉ có tác dụng khi đang chạy run-time
  - Hiển thị các kết quả xuất ra console, và
  - Nhập dữ liệu vào cho chương trình qua console.

MARS có tùy chọn để mọi thông tin nhập liệu vào qua console sẽ được hiển thị lại ra message area.

- ***MARS Messages* tab** được dùng để hiển thị cho các thông báo còn lại như là các báo lỗi trong quá trình biên dịch hay trong quá trình thực hiện run-time.
  - Có thể **click vào thông báo lỗi để chương trình tự động nhảy tới dòng lệnh** gây ra lỗi.

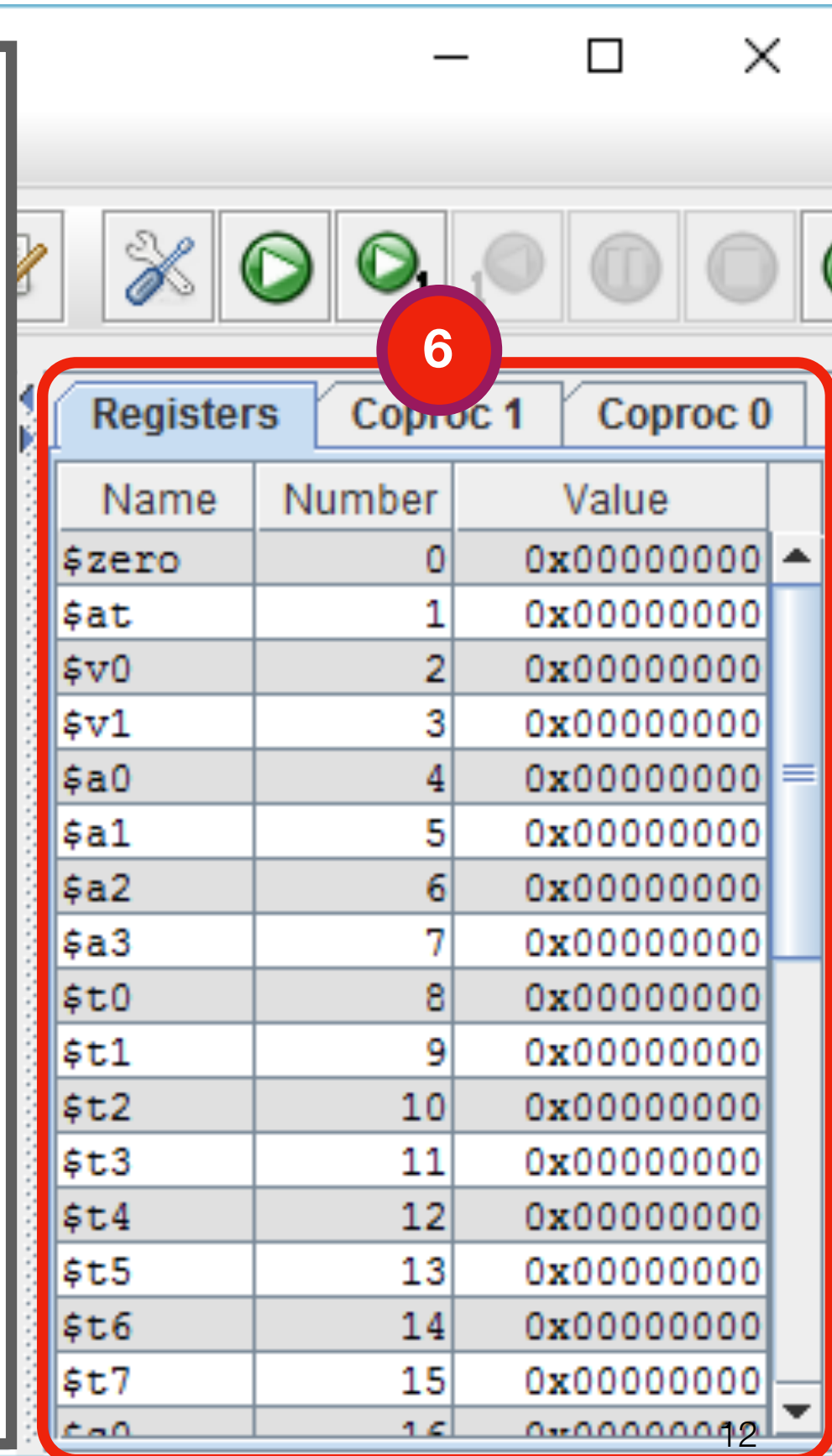


# Giao diện lập trình IDE cơ bản

**⑥ MIPS Registers:** Bảng hiển thị giá trị của các thanh ghi của bộ xử lý MIPS, luôn luôn được hiển thị, bất kể chương trình hợp ngữ có được chạy hay không. Khi viết chương trình, bảng này sẽ giúp người dùng nhớ tên của các thanh ghi và địa chỉ của chúng.

Có 3 tab trong bảng này:

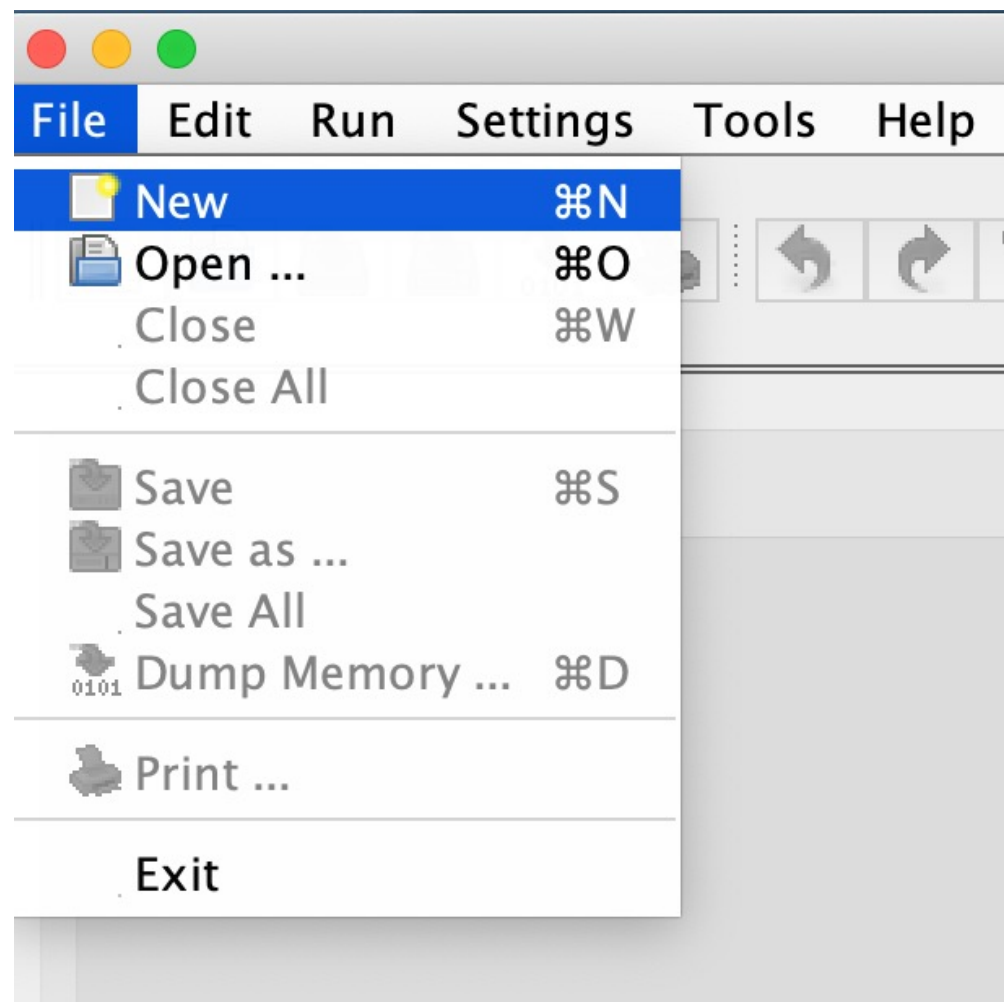
- **the Register File:** các thanh ghi số nguyên với địa chỉ từ \$0 tới \$31, và cả 3 thanh ghi đặc biệt LO, HI và thanh ghi Program Counter
- **the Coprocessor 0 registers:** các thanh ghi của bộ đồng xử lý C0, phục vụ cho xử lý ngắt
- **the Coprocessor 1 registers:** các thanh ghi số dấu phẩy động



Registers			Coproc 1	Coproc 0
Name	Number	Value		
\$zero	0	0x00000000		
\$at	1	0x00000000		
\$v0	2	0x00000000		
\$v1	3	0x00000000		
\$a0	4	0x00000000		
\$a1	5	0x00000000		
\$a2	6	0x00000000		
\$a3	7	0x00000000		
\$t0	8	0x00000000		
\$t1	9	0x00000000		
\$t2	10	0x00000000		
\$t3	11	0x00000000		
\$t4	12	0x00000000		
\$t5	13	0x00000000		
\$t6	14	0x00000000		
\$t7	15	0x00000000		
\$f0	16	0x00000000		

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

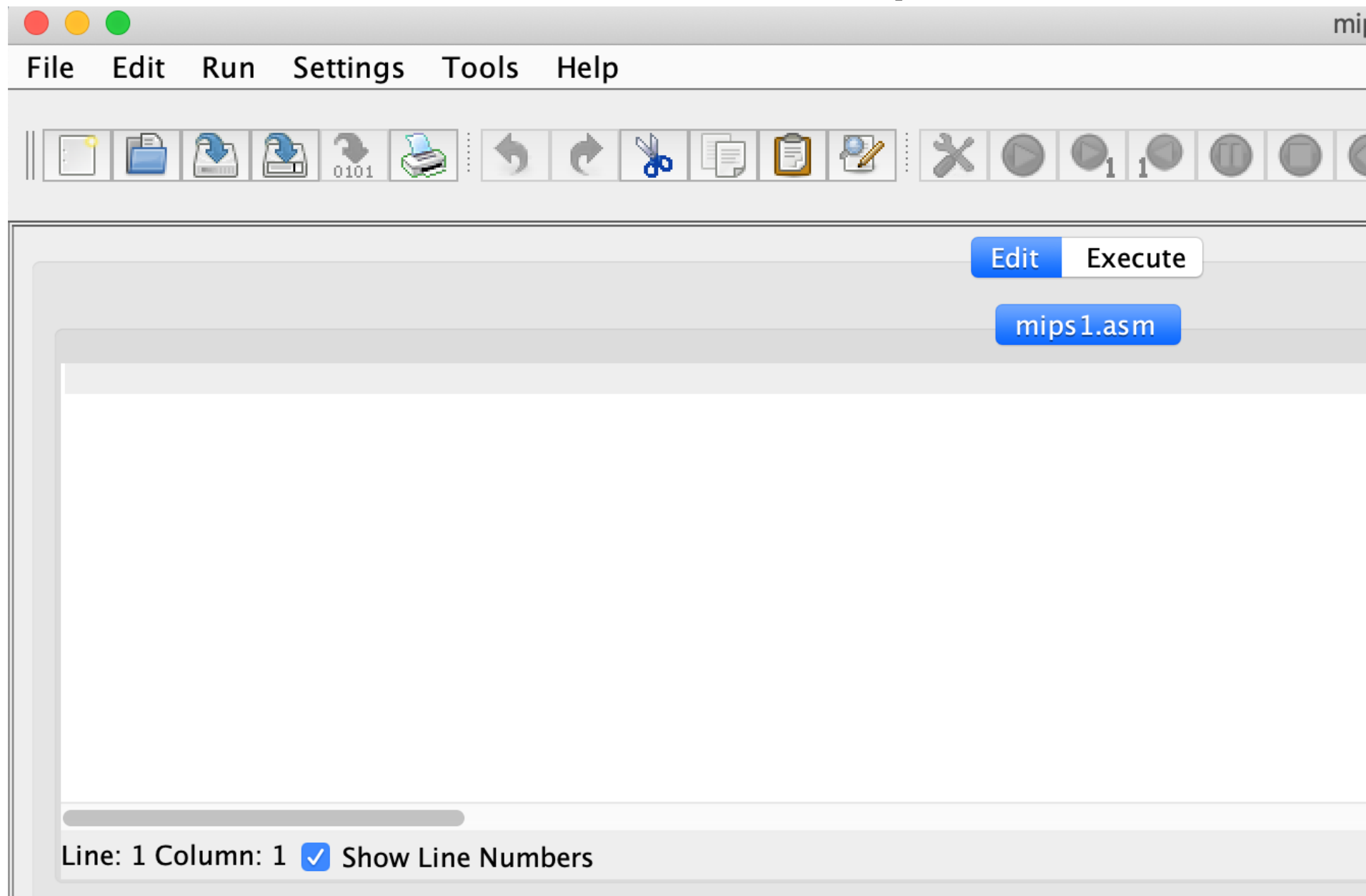
1. Click vào file mars.jar để bắt đầu chương trình
2. Ở thanh menu, chọn File/New để tạo một file hợp ngữ mới





# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

3. Cửa sổ soạn thảo file hợp ngữ sẽ hiện ra như hình bên. Có thể bắt đầu lập trình.



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

## 4. Hãy gõ đoạn lệnh sau vào cửa sổ soạn thảo

```
.data                # Vung du lieu, chua cac khai bao bien
x:      .word  0x01020304  # bien x, khoi tao gia tri
message: .asciiz "Bo mon Ky thuat May tinh"
.text               # Vung lenh, chua cac lenh hop ngu
    la $a0, message    #Dua dia chi bien mesage vao thanh ghi a0
    li $v0, 4           #Gan thanh ghi $v0 = 4
    syscall            #Goi ham so v0, ham so 4, la ham print

    addi $t1, $zero, 2   #Thanh ghi $t1 = 2
    addi $t2, $zero, 3   #Thanh ghi $t2 = 3
    add  $t0, $t1, $t2   #Thanh ghi t- = $t1 + $t2
```

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

Được kết quả như sau:

Edit Execute

mips1.asm\*


```
1  .data                # Vung du lieu, chua cac khai bao bien
2  x:      .word        0x01020304    # bien x, khoi tao gia tri
3  message: .asciiz     "Bo mon Ky thuat May tinh"
4
5  .text              # Vung lenh, chua cac lenh hop ngu
6      la    $a0, message    #Dua dia chi bien mesage vao thanh ghi a0
7      li    $v0, 4          #Gan thanh ghi $v0 = 4
8      syscall              #Goi ham so v0, ham so 4, la ham print
9
10     addi   $t1,$zero,2     #Thanh ghi $t1 = 2
11     addi   $t2,$zero,3     #Thanh ghi $t2 = 3
12     add    $t0, $t1, $t2   #Thanh ghi t- = $t1 + $t2
```



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

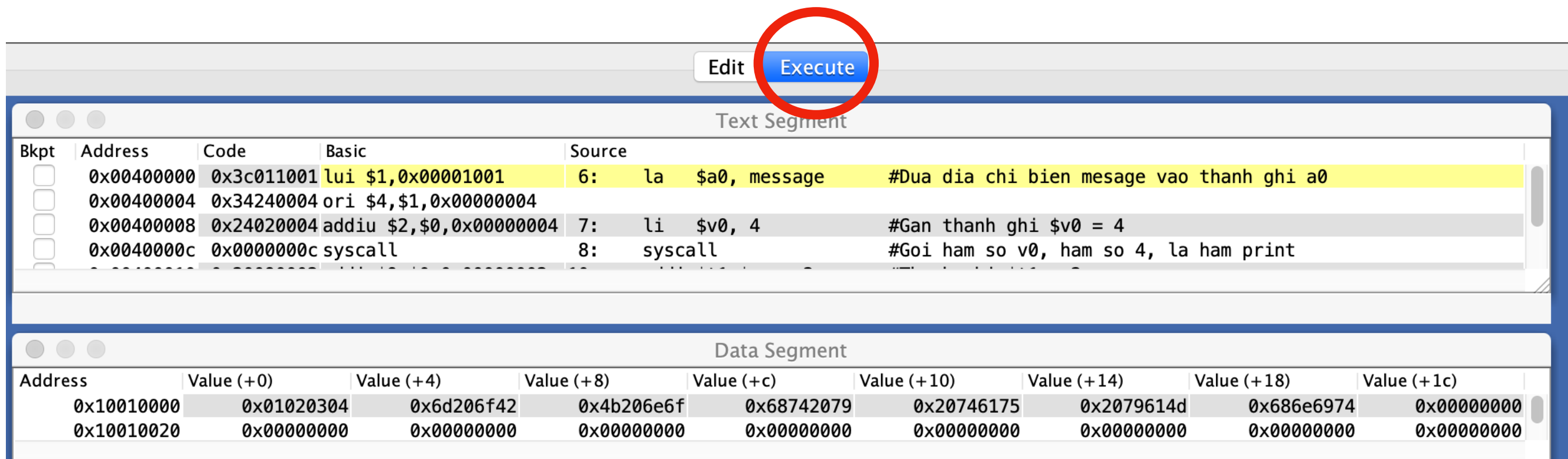
---

5. Để biên dịch chương trình hợp ngữ trên thành mã máy, thực hiện một trong các cách sau:

- Vào menu Run/Assemble, hoặc
- Trên thanh menu, bấm vào biểu tượng 
- Hoặc bấm phím tắt F3.

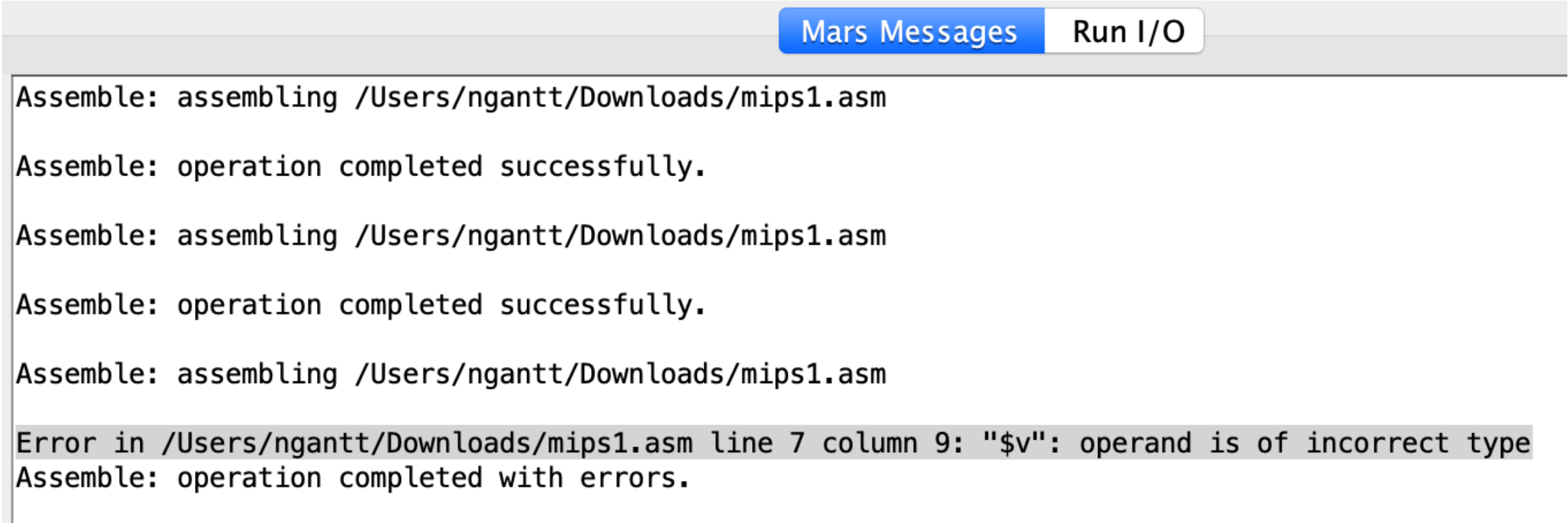
# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

6. Nếu đoạn hợp ngữ đúng, MARS sẽ chuyển từ Edit tab sang Execute tab



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

*Chú ý: nếu đoạn hợp ngữ có lỗi, cửa sổ Mars Messages sẽ hiển thị chi tiết lỗi. Bấm vào dòng thông báo lỗi để trình soạn thảo tự động nhảy tới dòng code bị lỗi, rồi tiến hành sửa lại cho đúng.*



The screenshot shows a window titled "Mars Messages" with a "Run I/O" button. The window contains the following text:

```
Assemble: assembling /Users/ngantt/Downloads/mips1.asm
Assemble: operation completed successfully.
Assemble: assembling /Users/ngantt/Downloads/mips1.asm
Assemble: operation completed successfully.
Assemble: assembling /Users/ngantt/Downloads/mips1.asm
Error in /Users/ngantt/Downloads/mips1.asm line 7 column 9: "$v": operand is of incorrect type
Assemble: operation completed with errors.
```

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

## 7. Ở Execute tab, có 2 cửa sổ chính là Text Segment, và Data Segment

The screenshot displays the Execute tab of a debugger, showing two main windows: Text Segment and Data Segment.

**Text Segment Window:**

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham so 4, la ham print

**Data Segment Window:**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

**Code Editor Window:**

```
1 .data # Vung du lieu, chua cac khai bao bien
2 x: .word 0x01020304 # bien x, khoi tao gia tri
3 message: .asciiz "Bo mon Ky thuat May tinh"
4
5 .text # Vung lenh, chua cac lenh hop ngu
6 la $a0, message #Dua dia chi bien mesage vao thanh ghi a0
7 li $v0, 4 #Gan thanh ghi $v0 = 4
8 syscall #Goi ham so v0, ham so 4, la ham print
9
10 addi $t1,$zero,2 #Thanh ghi $t1 = 2
11 addi $t2,$zero,3 #Thanh ghi $t2 = 3
12 add $t0, $t1, $t2 #Thanh ghi t- = $t1 + $t2
```

At the bottom, there are navigation arrows, a dropdown menu showing "0x10010000 (.data)", and checkboxes for "Hexadecimal Addresses" (checked), "Hexadecimal Values" (checked), and "ASCII" (unchecked).

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

## 7. Ở Execute tab, có 2 cửa sổ chính là Text Segment, và Data Segment

The screenshot displays the Execute tab of a debugger, showing two main windows: Text Segment and Data Segment.

**Text Segment Window:**

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham so 4, la ham print

**Data Segment Window:**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

**Assembly View:**

Navigation: 0x10010000 (.data) ☒ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

```
1 .data # Vung du lieu, chua cac khai bao bien
2 x: .word 0x01020304 # bien x, khoi tao gia tri
3 message: .asciiz "Bo mon Ky thuat May tinh"
4
5 .text # Vung lenh, chua cac lenh hop ngu
6 la $a0, message #Dua dia chi bien mesage vao thanh ghi a0
7 li $v0, 4 #Gan thanh ghi $v0 = 4
8 syscall #Goi ham so v0, ham so 4, la ham print
9
10 addi $t1,$zero,2 #Thanh ghi $t1 = 2
11 addi $t2,$zero,3 #Thanh ghi $t2 = 3
12 add $t0, $t1, $t2 #Thanh ghi t- = $t1 + $t2
```

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

---

7. Ở Execute tab, có 2 cửa sổ chính là **Text Segment**, và **Data Segment**

- **Text Segment:** là vùng không gian bộ nhớ chứa các mã lệnh hợp ngữ. Tương ứng với mã nguồn hợp ngữ, các dòng nào viết sau chỉ thị **.TEXT** tức là lệnh và sẽ thuộc Text Segment.
- **Data Segment:** là vùng không gian bộ nhớ chứa các biến. Tương ứng với mã nguồn hợp ngữ, các dòng nào viết sau chỉ thị **.DATA** tức là lệnh và sẽ thuộc Text Segment.

*Chú ý: vì lý do nào đó, nếu ta khai báo biến sau chỉ thị **.TEXT** hoặc ngược lại thì trình biên dịch sẽ báo lỗi hoặc bỏ qua khai báo sai đó.*



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

---

8. Ở Execute tab, sử dụng checkbox bên dưới để thay đổi cách hiển thị dữ liệu cho dễ quan sát
- ☒ Hexadecimal Addresses : hiển thị địa chỉ ở dạng số nguyên hệ 16
  - ☒ Hexadecimal Values : hiển thị giá trị thanh ghi ở dạng số nguyên hệ 16.
  - ☒ ASCII : hiển thị giá trị trong bộ nhớ ở dạng kí tự ASCII

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

8. Ở Execute tab, sử dụng checkbox bên dưới để thay đổi cách hiển thị dữ liệu cho dễ quan sát.

The screenshot shows the Execute tab of a debugger. The left sidebar is highlighted with a red box. The main window displays the Text Segment and Data Segment. The Text Segment table shows assembly instructions. The Data Segment table shows memory values. At the bottom, the 'Hexadecimal Addresses' checkbox is checked and highlighted with a red box.

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,4097	6: la \$a0, message #Dua địa chỉ biến message vào thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,4	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,4	7: li \$v0, 4 #Gán thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Gọi hàm số v0, hàm số 4, là hàm print

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	16909060	1830842178	1260416623	1752440953	544498037	544825677	1752066420	0
0x10010020	0	0	0	0	0	0	0	0

0x10010000 (.data) ☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

8. Ở Execute tab, sử dụng checkbox bên dưới để thay đổi cách hiển thị dữ liệu cho dễ quan sát.
- ☒ **Hexadecimal Addresses** : hiển thị địa chỉ ở dạng cơ số 16

The screenshot displays the Execute tab of a debugger, showing the Text Segment and Data Segment. The Text Segment table lists assembly instructions with their addresses in hexadecimal. The Data Segment table shows memory values in decimal. The 'Hexadecimal Addresses' checkbox is checked, indicating that addresses are displayed in hexadecimal.

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,4097	6: la \$a0, message #Dua địa chỉ biến message vào thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,4	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,4	7: li \$v0, 4 #Gán thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Gọi hàm số v0, hàm số 4, là hàm print

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	16909060	1830842178	1260416623	1752440953	544498037	544825677	1752066420	0
0x10010020	0	0	0	0	0	0	0	0

Navigation: 0x10010000 (.data) ☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

8. Ở Execute tab, sử dụng checkbox bên dưới để thay đổi cách hiển thị dữ liệu cho dễ quan sát.
- ☒ **Hexadecimal Values** : hiển thị giá trị thanh ghi ở dạng cơ số 16.

**Text Segment**

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	4194304	0x3c011001	lui \$t0, 0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	4194308	0x34240004	ori \$t0, \$1, 0x00000004	
<input type="checkbox"/>	4194312	0x24020004	addiu \$t2, \$t0, 0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	4194316	0x0000000c	syscall	8: syscall #Goi ham so v0, ham so 4, la ham print

**Data Segment**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
268501024	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☐ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

8. Ở Execute tab, sử dụng checkbox bên dưới để thay đổi cách hiển thị dữ liệu cho dễ quan sát.
- ☒ **ASCII** : hiển thị giá trị trong bộ nhớ ở dạng ký tự ASCII

The screenshot displays a debugger interface with two main panels: 'Text Segment' and 'Data Segment'.

**Text Segment:**

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	4194304	0x3c011001	lui \$1,4097	6: la \$a0, message #Đưa địa chỉ biến message vào thanh ghi a0
<input type="checkbox"/>	4194308	0x34240004	ori \$4,\$1,4	
<input type="checkbox"/>	4194312	0x24020004	addiu \$2,\$0,4	7: li \$v0, 4 #Gán thanh ghi \$v0 = 4
<input type="checkbox"/>	4194316	0x0000000c	syscall	8: syscall #Gọi hàm số v0, hàm số 4, là hàm print

**Data Segment:**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	. . . .	m o B	K n o	h t y	t a u	y a M	h n i t	\0 \0 \0 \0
268501024	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0


At the bottom of the Data Segment panel, the following options are visible:

- Navigation arrows: left and right.
- Address field: 0x10010000 (.data)
- Hexadecimal Addresses: ☐
- Hexadecimal Values: ☐
- ASCII: ☒ (highlighted with a red box)

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

9. Ở Execute tab, trong cửa sổ Text Segment là bảng có 5 cột.

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham so 4, la ham print

- **Bkpt:** Breakpoint, điểm dừng khi chạy toàn bộ chương trình chương trình bằng nút 

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

9. Ở Execute tab, trong cửa sổ Text Segment, bảng có 5 cột.

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham so 4, la ham print

- **Address:** địa chỉ của lệnh ở dạng số nguyên.

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

9. Ở Execute tab, trong cửa sổ Text Segment, bảng có 5 cột.

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ri \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	ddiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	yscall	8: syscall #Goi ham so v0, ham so 4, la ham print

- **Code:** lệnh ở dạng mã máy

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

9. Ở Execute tab, trong cửa sổ Text Segment, bảng có 5 cột.

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham so 4, la ham print

- **Basic:** lệnh ở dạng hợp ngữ thuần, **giống như quy định trong tập lệnh.**

Ở đây, tất cả các nhãn, tên gởi nhớ.. đều đã được chuyển đổi thành hằng số.



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

## 9. Ở Execute tab, trong cửa sổ Text Segment, bảng có 5 cột.

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien mesage vao thanh ghi a0
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham so 4, la ham print

- **Source:** lệnh ở dạng hợp ngữ có bổ sung các macro, nhãn.. giúp lập trình nhanh hơn, dễ hiểu hơn, **không còn giống như tập lệnh** nữa.
  - Lệnh **la** trong cột Source là lệnh giả, không có trong tập lệnh được dịch tương ứng thành 2 lệnh **lui** và **ori** trong cột Basic.
  - Nhãn message trong lệnh **la \$a0, message** trong cột Source được dịch thành hằng số 0x00001001.



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

10. Ở Execute tab, trong cửa sổ Data Segment, bảng có 9 cột.

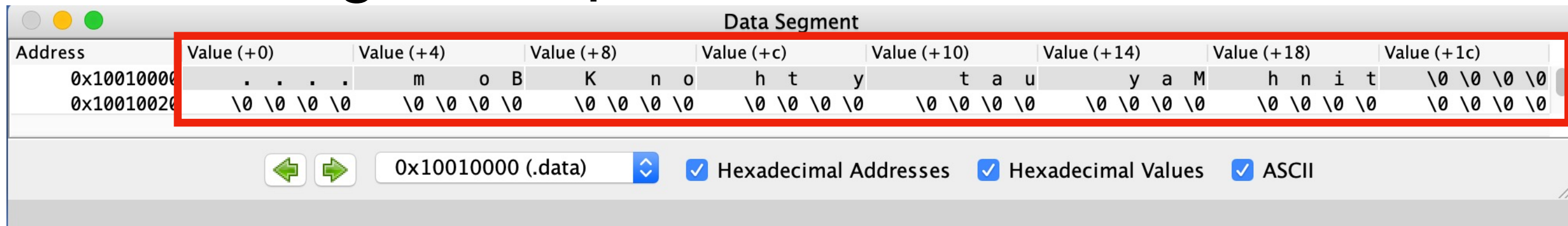
Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x10010000	.	m	K	h	t	y	a	M	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data)
 ☒ Hexadecimal Addresses
 ☒ Hexadecimal Values
 ☒ ASCII

- **Address:** địa chỉ của dữ liệu, biến ở dạng số nguyên. Giá trị mỗi dòng tăng 32 đơn vị (ở hệ 10, hoặc  $20_{(16)}$ ) bởi vì mỗi dòng sẽ trình bày 32 byte ở các địa chỉ liên tiếp nhau

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

10. Ở Execute tab, trong cửa sổ Data Segment, bảng có 9 cột.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	.	m	K	h	t	y	t	a
0x10010020	\0	\0	\0	\0	\0	\0	\0	\0

- **Các cột Value:** mỗi cột Value chứa 4 byte, và có 8 cột Value, tương ứng với 32 byte liên tiếp nhau.

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

10. Ở Execute tab, trong cửa sổ Data Segment, bảng có 9 cột.

The top screenshot shows the Data Segment window with the following table:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	.	m	o	B	K	n	o	h
0x10010020	\0	\0	\0	\0	\0	\0	\0	\0

The assembly code below shows:

```
1 .data # Vung du lieu, chua cac khai bao bien
2 x: .word 0x01020304 # bien x, khoi tao gia tri
3 message: .ascii "Bo mon Ky thuat May tinh"
```

The bottom screenshot shows the Data Segment window after execution, with the following table:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

The assembly code below shows:

```
1 .data # Vung du lieu, chua cac khai bao bien
2 x: .word 0x01020304 # bien x, khoi tao gia tri
3 message: .ascii "Bo mon Ky thuat May tinh"
```

- Có thể thấy rõ giá trị của biến **x = 0x01020304** được hiển thị chính xác trong Data Segment khi hiển thị dữ liệu ở dạng số ☒ **Hexadecimal Values**, và giá trị của chuỗi “Bo mon Ky thuat May tinh” khi hiển thị ở dạng kí tự ☒ **ASCII**

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

10. Ở Execute tab, trong cửa sổ Data Segment, bảng có 9 cột.

The top screenshot shows the Data Segment window with the following table:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	.	m	o	B	K	n	o	h
0x10010020	\0	\0	\0	\0	\0	\0	\0	\0

The assembly code below shows:

```
1 .data # Vung du lieu, chua cac khai bao bien
2 x: .word 0x01020304 # bien x, khoi tao gia tri
3 message: .asciz "Bo mon Ky thuat May tinh"
```

The bottom screenshot shows the Data Segment window after execution, with the following table:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

- Lưu ý rằng việc lưu trữ chuỗi trong bộ nhớ ở dạng little-endian là do cách lập trình của hàm **syscall**, chứ không phải do bộ xử lý MIPS quy định. Có thể thấy, ở công cụ giả lập MIPS IT, hàm **print** lại quy định chuỗi ở dạng big-endian.

# Little endian và Big endian

---

- Là hai phương thức khác nhau để lưu trữ dữ liệu dạng nhị phân.
- Little endian (little-end) là cơ chế lưu trữ mà byte cuối cùng trong biểu diễn nhị phân được ghi đầu tiên.
- Big endian (big-end) là cơ chế lưu trữ mà byte cuối cùng trong biểu diễn nhị phân được ghi sau cùng.
- Lưu ý: hai cơ chế trên chỉ đảo thứ tự của byte dữ liệu chứ không đảo thứ tự của từng bit trong byte.



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

Bấm vào cặp nút



để dịch chuyển tới vùng địa chỉ lân cận.

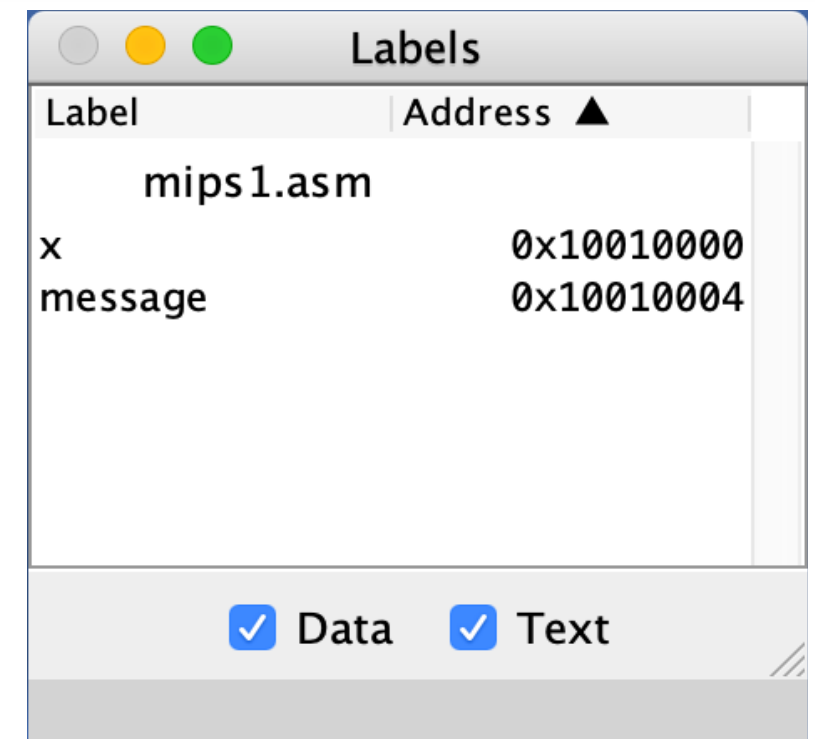
- Bấm vào combobox để dịch tới vùng bộ nhớ chứa loại dữ liệu được chỉ định. Trong đó:

- **.data**: vùng dữ liệu
- **.text**: vùng lệnh
- **\$sp**: vùng ngăn xếp

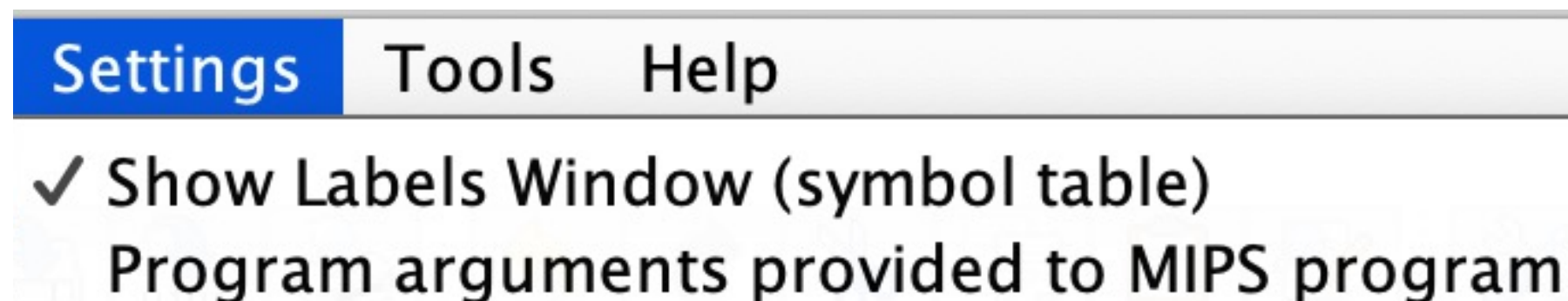
```
0x10000000 (.extern)
✓ 0x10010000 (.data)
0x10040000 (heap)
current $gp
current $sp
0x00400000 (.text)
0x90000000 (.kdata)
0xffff0000 (MMIO)
```

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

11. Cửa sổ **Label**: hiển thị tên nhãn và hằng số địa chỉ tương ứng với nhãn khi được biên dịch ra mã máy.



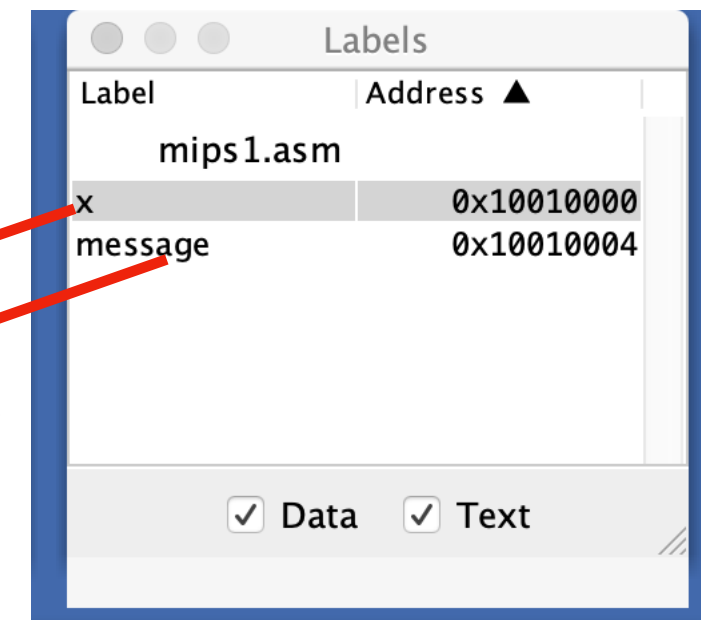
- Cửa sổ **Label** không tự động hiển thị. Cần thiết lập trong menu **Settings/Show Labels Windows**.



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

Trong cửa sổ **Labels**:

- **x** là tên gọi nhớ, được quy đổi thành hằng số 0x10010000.
- **message** là tên gọi nhớ, được quy đổi thành hằng số 0x10010004



Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

← →

0x10010000 (.data)

Hexadecimal Addresses

Hexadecimal Values

ASCII

- Nhấp đúp vào tên biến sẽ tự động chuyển sang vị trí tương ứng trong cửa sổ Data Segment.



# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

The screenshot displays two windows from a debugger. The 'Text Segment' window shows assembly code with the instruction `lui $1, 0x00001001` at address `0x00400000` and `ori $4, $1, 0x00000004` at address `0x00400004`. The 'Labels' window shows the `message` label at address `0x10010004`. The 'Data Segment' window shows a memory dump starting at address `0x10010000`.

**Text Segment**

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1, 0x00001001	6: la \$a0, message #Dua dia chi bien m...
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4, \$1, 0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2, \$0, 0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham...
<input type="checkbox"/>	0x00400010	0x20090002	addi \$9, \$0, 0x00000002	10: addi \$t1, \$zero, 2 #Thanh ghi \$t1 = 2
<input type="checkbox"/>	0x00400014	0x200a0003	addi \$10, \$0, 0x00000003	11: addi \$t2, \$zero, 3 #Thanh ghi \$t2 = 3
<input type="checkbox"/>	0x00400018	0x012a4020	add \$8, \$9, \$10	12: add \$t0, \$t1, \$t2 #Thanh ghi t- = \$t1...

**Labels**

Label	Address ▲
mips1.asm	
x	0x10010000
message	0x10010004

☒ Data ☒ Text

**Data Segment**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☒ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

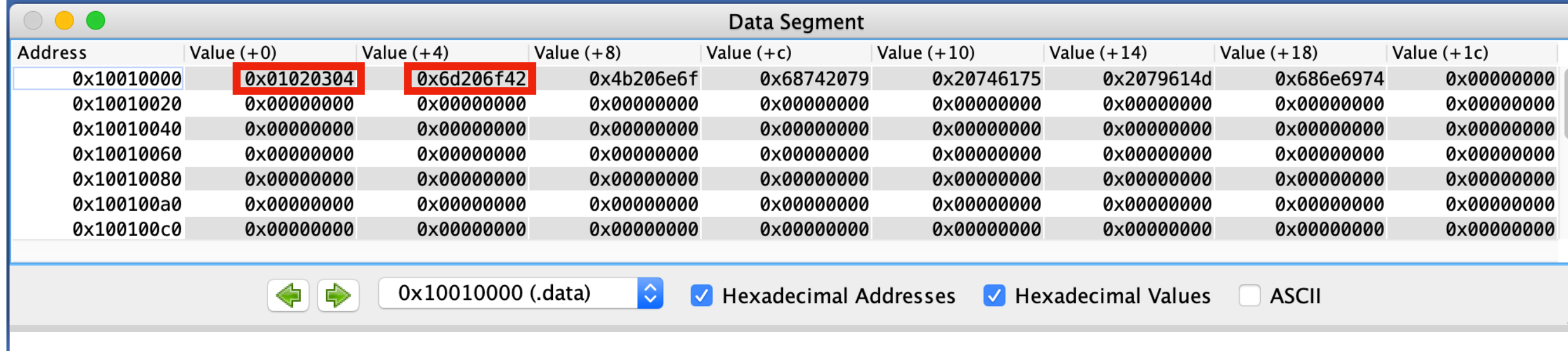
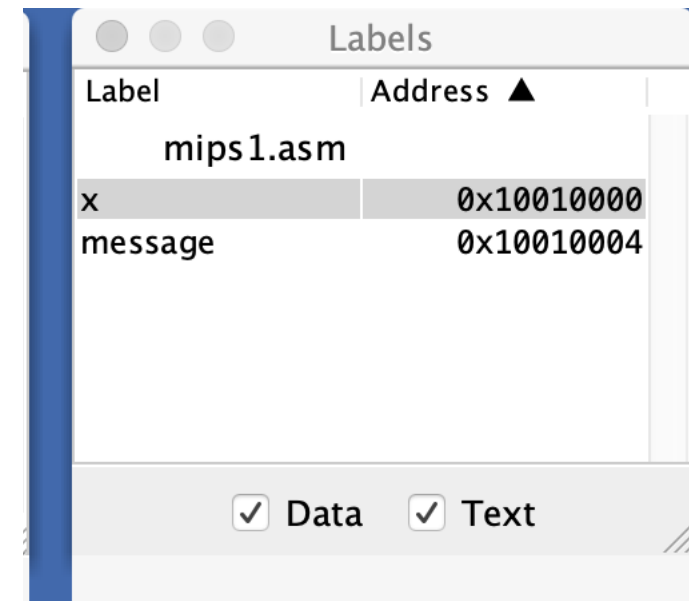
Trong cửa sổ **Text Segment**:

- Ở lệnh gán **la \$a0, message** tên gọi nhớ **message** đã được chuyển thành hằng số `0x10010004` thông qua cặp lệnh **lui, ori**

# Bắt đầu lập trình và tìm hiểu các công cụ với chương trình Helloworld

Trong cửa sổ **Data Segment**:

- Để giám sát giá trị của biến **x**, mở Data Segment ở hằng số 0x10010000 sẽ nhìn thấy giá trị của **x**.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Trong cửa sổ **Data Segment**:

- Để giám sát giá trị của biến **message**, mở Data Segment ở hằng số 0x10010004 sẽ nhìn thấy giá trị của **message**.

# Chạy giả lập

## 1. Tiếp tục chạy chương trình HelloWorld ở trên.

EditExecute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	6: la \$a0, message #Dua dia chi bien m...
<input type="checkbox"/>	0x00400004	0x34240004	ori \$4,\$1,0x00000004	
<input type="checkbox"/>	0x00400008	0x24020004	addiu \$2,\$0,0x00000004	7: li \$v0, 4 #Gan thanh ghi \$v0 = 4
<input type="checkbox"/>	0x0040000c	0x0000000c	syscall	8: syscall #Goi ham so v0, ham...
<input type="checkbox"/>	0x00400010	0x20090002	addi \$9,\$0,0x00000002	10: addi \$t1,\$zero,2 #Thanh ghi \$t1 = 2
<input type="checkbox"/>	0x00400014	0x200a0003	addi \$10,\$0,0x00000003	11: addi \$t2,\$zero,3 #Thanh ghi \$t2 = 3
<input type="checkbox"/>	0x00400018	0x012a4020	add \$8,\$9,\$10	12: add \$t0, \$t1, \$t2 #Thanh ghi t- = \$t1...

Labels

Label	Address ▲
mips1.asm	
x	0x10010000
message	0x10010004

☒ Data☒ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x01020304	0x6d206f42	0x4b206e6f	0x68742079	0x20746175	0x2079614d	0x686e6974	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

←→

0x10010000 (.data)

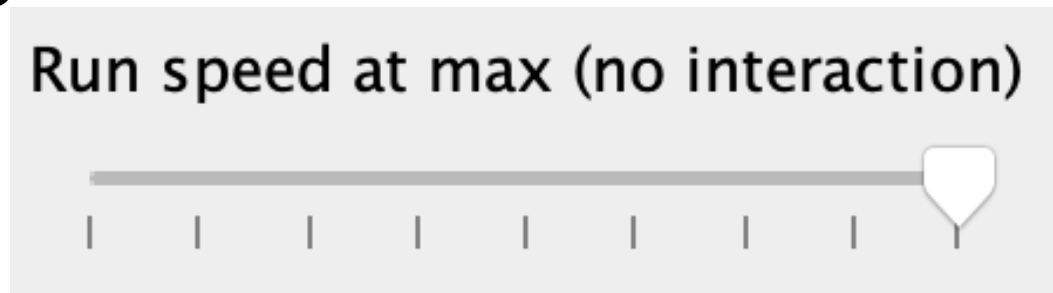
Hexadecimal Addresses

Hexadecimal Values

ASCII

# Chạy giả lập





2. Sử dụng slider bar để thay đổi tốc độ thực thi lệnh hợp ngữ.



- Mặc định, tốc độ thực thi là tối đa, và ở mức này, người dùng không thể can thiệp được nhiều vào quá trình hoạt động của các lệnh và kiểm soát chúng.
- Có thể dịch chuyển slider bar về khoảng 2 lệnh/giây để dễ quan sát.

# Chạy giả lập

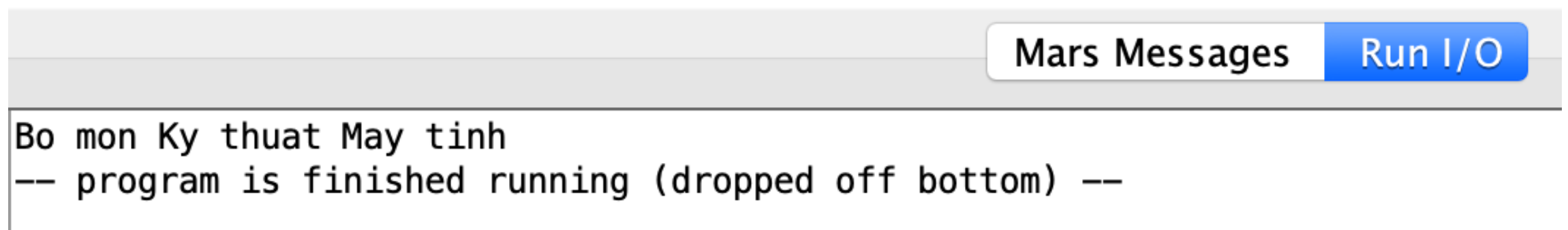
## 2. Ở **Execute tab**, chọn cách thực thi chương trình

- Bấm vào icon  **Run**, để thực hiện toàn bộ chương trình. Khi sử dụng **Run**, quan sát dòng lệnh được tô màu vàng cho biết chương trình hợp ngữ đang được xử lý tới lệnh nào. Đồng thời, quan sát sự biến đổi dữ liệu trong cửa sổ Data Segment và cửa sổ Register.
- Bấm vào icon  **Reset**, để khởi động lại trình giả lập về trạng thái ban đầu. Tất cả các ngăn nhớ và các thanh ghi đều được gán lại về 0.
- Bấm vào icon  **Run one step**, để thực thi chỉ duy nhất 1 lệnh rồi chờ bấm tiếp vào icon đó, để thực hiện lệnh kế tiếp.
- Bấm vào icon  **Run one step backwards**, để khôi phục lại trạng thái và quay trở lại lệnh đã thực thi trước đó.



# Chạy giả lập

2. Ở **Execute tab**, chọn cách để thực thi chương trình
- Sau khi chạy xong tất cả các lệnh của chương trình HelloWorld, sẽ thấy cửa sổ **Run I/O** hiển thị chuỗi.



The screenshot shows a window titled "Mars Messages" with a blue "Run I/O" button. The output text is as follows:

```
Bo mon Ky thuat May tinh
-- program is finished running (dropped off bottom) --
```

# Giả lập & gỡ rối

## Quan sát sự thay đổi của các biến

---

- Trong quá trình chạy giả lập, hãy chạy từng lệnh với chức năng **Run one step**.
- Ở mỗi lệnh, quan sát sự thay đổi giá trị trong cửa sổ **Data Segment** và cửa sổ **Register**.
- Tìm hiểu ý nghĩa của sự thay đổi đó.

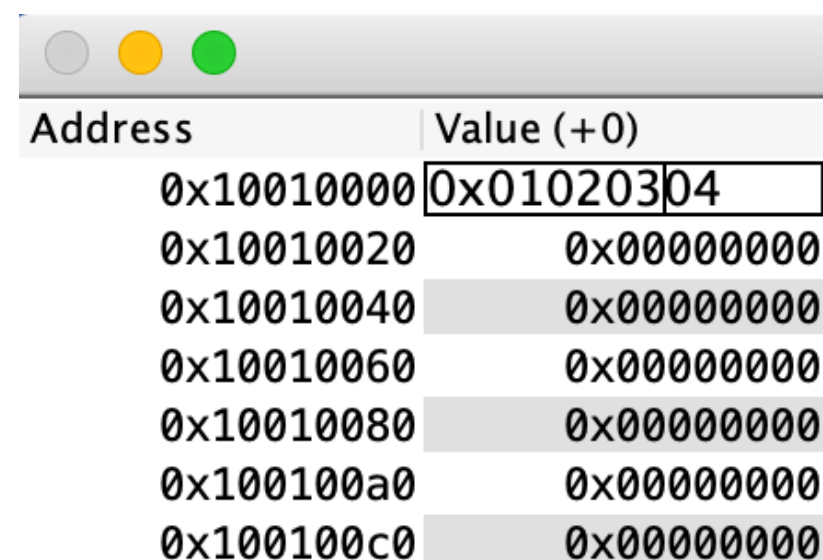


# Giả lập & gỡ rối

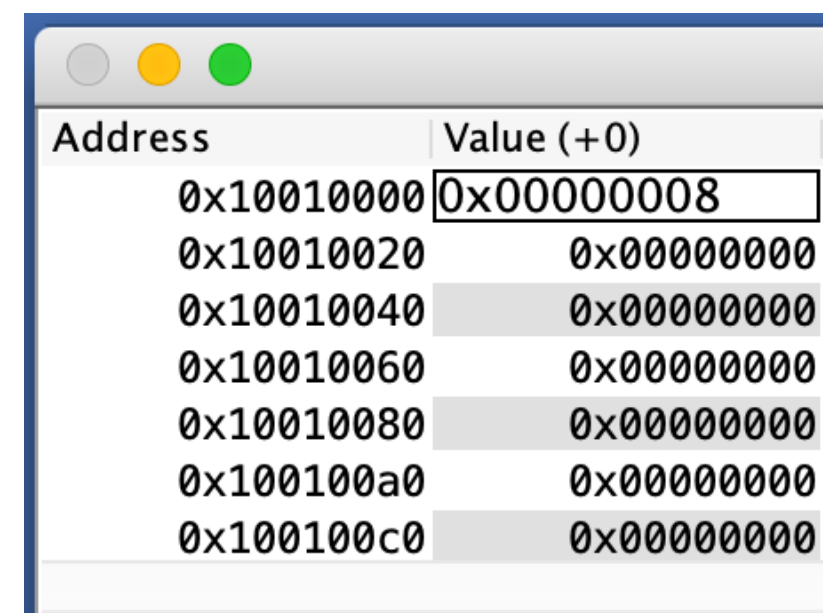
## Thay đổi giá trị biến khi đang chạy run-time

Trong khi đang chạy giả lập, có thể thay đổi giá trị của một ngăn nhớ bất kì bằng cách:

- Trong **Data Segment**, nhấp đúp vào một ngăn nhớ bất kì
- Nhập giá trị mới
- Tiếp tục chạy chương trình



Address	Value (+0)
0x10010000	0x01020304
0x10010020	0x00000000
0x10010040	0x00000000
0x10010060	0x00000000
0x10010080	0x00000000
0x100100a0	0x00000000
0x100100c0	0x00000000



Address	Value (+0)
0x10010000	0x00000008
0x10010020	0x00000000
0x10010040	0x00000000
0x10010060	0x00000000
0x10010080	0x00000000
0x100100a0	0x00000000
0x100100c0	0x00000000

**Chú ý:** nếu chương trình biên dịch từ đầu thì giá trị mới không được áp dụng.

# Giả lập & gỡ rối


Thay đổi giá trị thanh ghi khi đang chạy run-time

Tương tự, có thể thay đổi giá trị của thanh ghi trong cửa sổ **Registers**.

Registers			Coproc 1	Coproc 0
Name	Number	Value		
\$zero	0	0x00000000		
\$at	1	0x00000000		
\$v0	2	0x00000000		
\$v1	3	0x00000000		
\$a0	4	0x00000000		
\$a1	5	0x00000000		

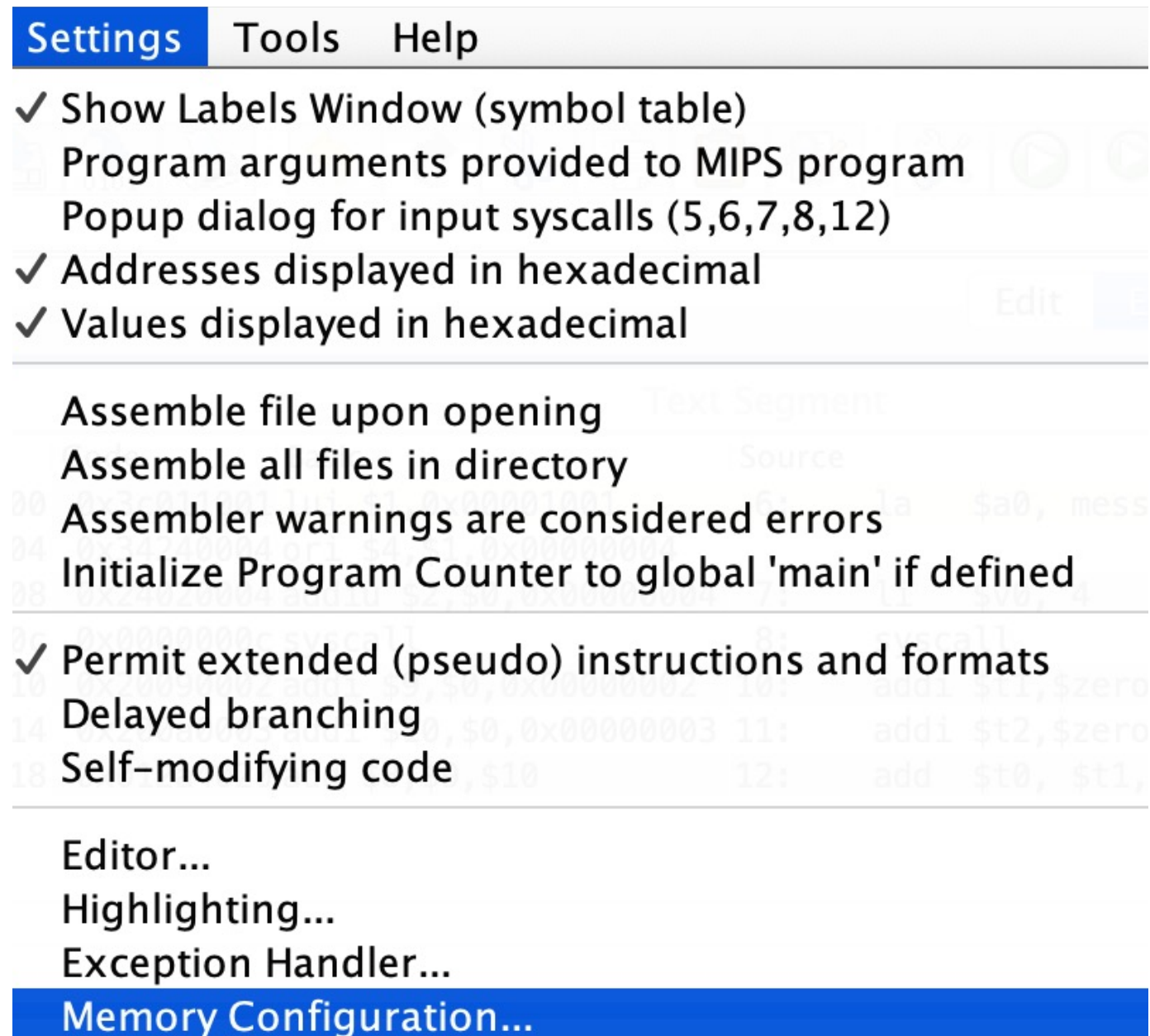
# Tra cứu Help

---

- Bấm nút **Help**  để xem giải thích các lệnh của MIPS, các giả lệnh, chỉ dẫn biên dịch và các hàm của syscalls.

# Các hằng địa chỉ

- Chọn menu Settings/Memory Configuration



# Các hằng địa chỉ

- Cửa sổ MIPS Memory Configuration chứa bảng quy định các hằng địa chỉ mà MARS sử dụng.
- Theo bảng này, có thể thấy các mã lệnh luôn bắt đầu từ địa chỉ **0x00400000**, còn dữ liệu luôn bắt đầu từ địa chỉ **0x10000000**.

MIPS Memory Configuration

0xffffffff	memory map limit address
0xffffffff	kernel space high address
0xffff0000	MMIO base address
0xffffefff	kernel data segment limit address
0x90000000	.kdata base address
0x8fffffff	kernel text limit address
0x80000180	exception handler address
0x80000000	kernel space base address
0x80000000	.ktext base address
0x7fffffff	user space high address
0x7fffffff	data segment limit address
0x7ffffffc	stack base address
0x7fffeffc	stack pointer \$sp
0x10040000	stack limit address
0x10040000	heap base address
0x10010000	.data base address
0x10008000	global pointer \$gp
0x10000000	data segment base address
0x10000000	.extern base address
0x0ffffffc	text limit address
0x00400000	.text base address

Configuration

☒ Default

☐ Compact, Data at Address 0

☐ Compact, Text at Address 0

Apply and Close   Apply   Cancel   Reset

# Tuần 1

---

- Giới thiệu công cụ mô phỏng MARS
- Hướng dẫn cài đặt
- Giao diện lập trình IDE cơ bản
- Lập trình và tìm hiểu công cụ lập trình với chương trình “**HelloWorld**”
- Chạy giả lập chương trình mô phỏng
- Tra cứu HELP

---

End of week 1