Laboratory Exercise 1

Giới thiệu về công cụ MARS

Goals

Sau bài thực hành này, bạn sẽ cài đặt được công cụ MARS, viết thử chương trình đơn giản để thử nghiệm công cụ MARS như lập trình hợp ngữ, chạy giả lập, gỡ rối, và các phương tiện nhằm hiểu rõ hơn về bản chất và các hoạt động thực sự đã xảy $ratrong bộ xử lý MIPS.

Literature

- Tài liệu Kien truc MIPS.pptx
- Tài liệu MARS features, file .doc
- Tài liệu MARS Tutorial, file .doc

About MIPSIT

- MIPS (**Microprocessor without Interlocked Pipeline Stages**) hình thành trên cơ sở RISC.
- Năm 1981: John L. Hennessy đứng đầu một nhóm bắt đầu một công trình nghiên cứu về *bộ xử lý MIPS* đầu tiên tại **Stanford University**
- Một thiết kế chủ chốt trong MIPS là yêu cầu các câu lệnh phải hoàn thành trong 1 chu kì máy.
- Một số ứng dụng
    - Pioneer DVR-57-H
    - Kenwood HDV-810 Car Navigation System
    - HP Color Laser Jet 2500 Printer

Kick-off

Tải về và chạy

1. Tải về Java Runtime Environment, JRE, để chạy công cụ MARS
   http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. Cài đặt JRE
3. Tải về công cụ MARS, bao gồm
    a. Phiên bản mới nhất của MARS, và nên lấy thêm 2 tài liệu:
    b. MARS features, và
    c. MARS Tutorial

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

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

Chú ý: để tiện lợi, các sinh viên có thể tải về các tài nguyên liên quan tại URL duy nhất sau: ftp://dce.hust.edu.vn/tiennd/ict4/Mars/

Cơ bản về giao diện lập trình IDE

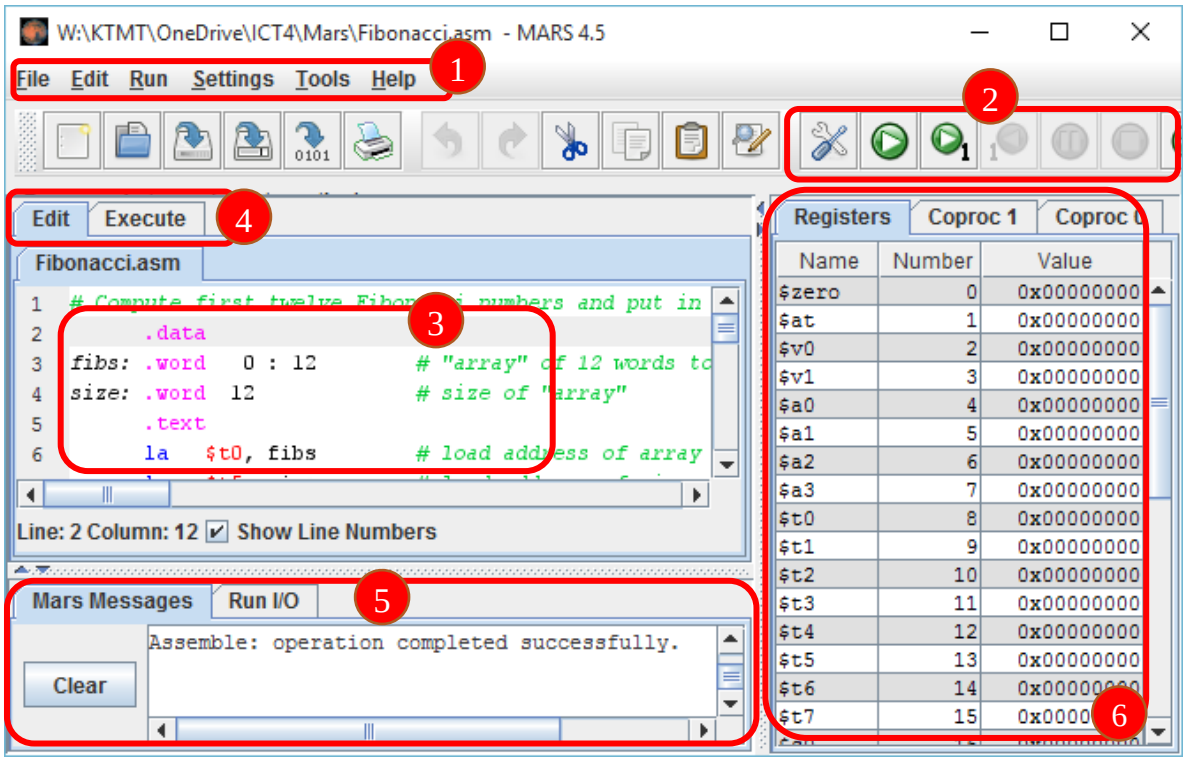


Figure 1. IDE of MARS Tool

1. **Menus**: Hầu hết các mục trong menu đều có các icon tương ứng
   - Di chuyển chuột lên trên của icon → tooltip giải thích về chức năng tương ứng sẽ hiển thị.

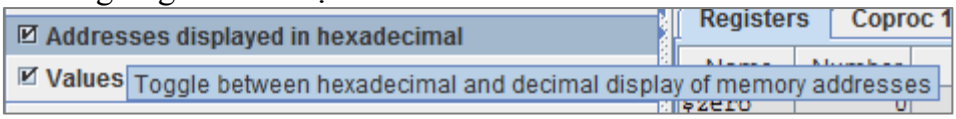


Figure 2. Tooltip giải thích chức năng trong các Menu

   - Các mục trong menu cũng có phím tắt tương ứng.
2. **Toolbar:**
   - Chứa một vài tính năng soạn thảo cơ bản như: copy, paste, open..
   - Các tính năng gỡ rỗi (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
3. **Edit tab:** MARS có bộ soạn thảo văn bản tích hợp sẵn với tính tô màu theo cú pháp, giúp sinh viên dễ theo dõi mã nguồn. Đồng thời, khi sinh

viên gõ lệnh mà chưa hoàn tất, một popup sẽ hiện $rađể trợ giúp. Vào menu Settings / Editor… để thay đổi màu sắc, font…

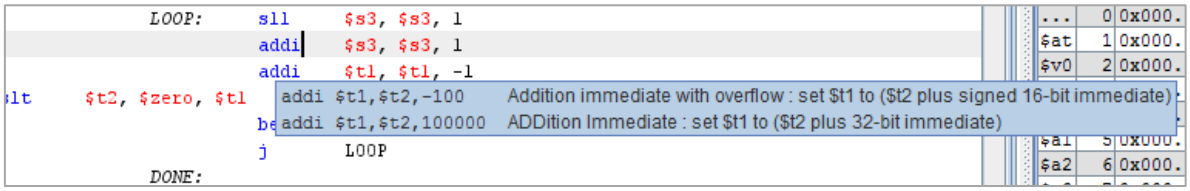


Figure 3. Popup trợ giúp hoàn tất lệnh và giải thích lệnh

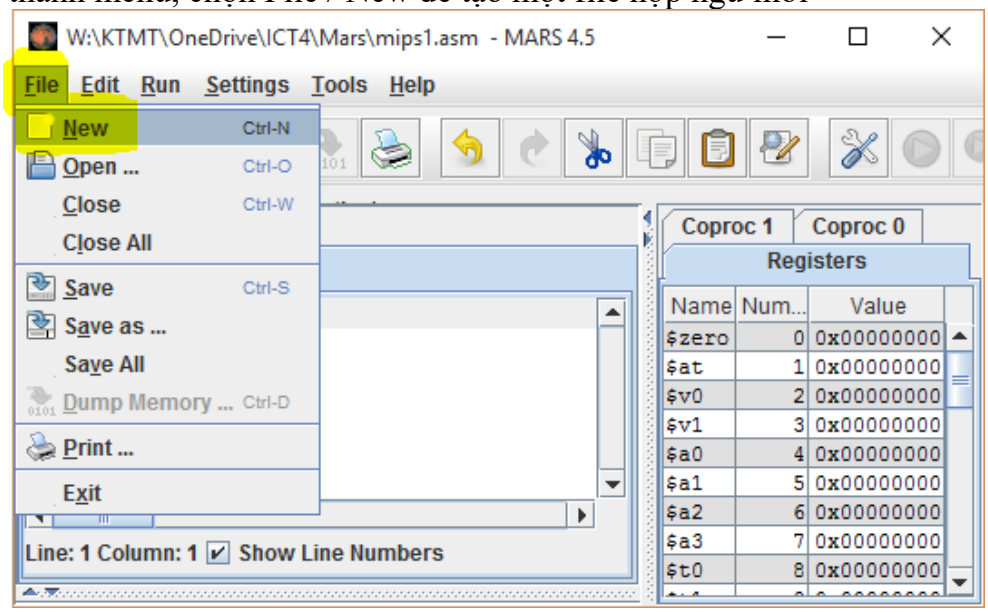
4. **Edit/Execute**:
   Mỗi file mã nguồn ở giao diện soạn thảo có 2 cửa sổ - 2 tab: Edit và Execute
   - **Edit tab:** viết chương trình hợp ngữ với tính năng tô màu theo cú pháp.
   - **Execute tab:** biên dịch chương trình hợp ngữ đã viết ở Edit tab thành mã máy, **chạy** và **gỡ rối**.
5. **Message 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 $raconsole, và
     - Nhập dữ liệu vào cho chương trình qua console.

     MARS có tùy chọn để bạn có thể mọi thông tin nhập liệu vào qua console sẽ được hiển thị lại $ramessage 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. Bạn có thể click vào thông báo lỗi để chương trình tự động nhảy tới dòng lệnh gây $ralỗi.
6. **MIPS Registers**: Bảng hiển thị giá trị của các thanh ghi của bộ xử lý MIPS, luôn luôn được hiện thị, bất kể chương trình hợp ngữ có được chạy hay không. Khi viết chương trình, bảng này sẽ giúp bạn nhớ được tên của các thành 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 LO, HI và thanh ghi Program Counter
   - **the Coprocesor 0 registers:** các thanh ghi của bộ đồng xử lý C0, phục vụ cho xử lý ngắt
   - **the Coprocesor 1 registers:** các thanh ghi số dấu phẩy động

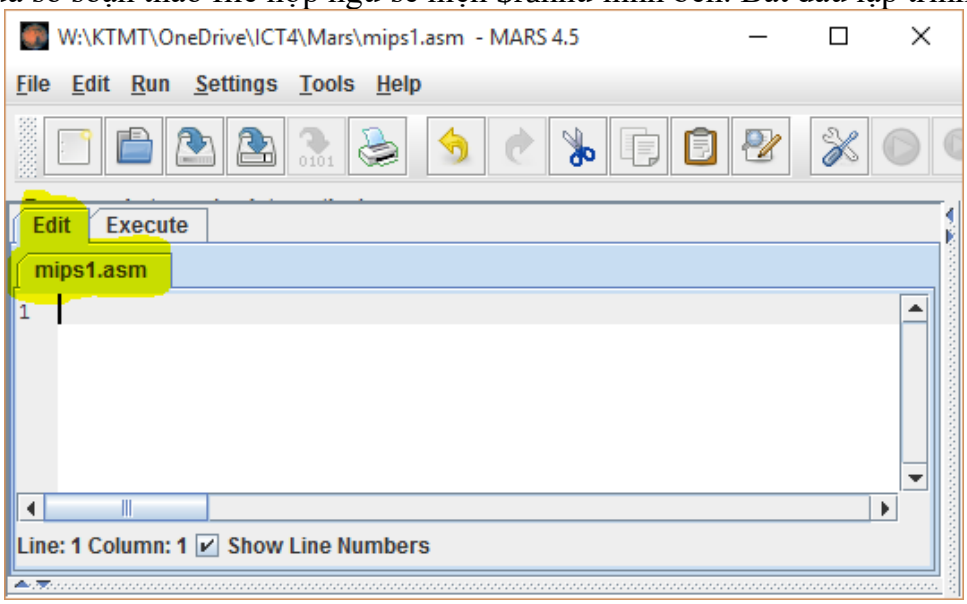
Bắt đầu lập trình và hiểu các công cụ với chương trình Helloworld

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



3. Cửa sổ soạn thảo file hợp ngữ sẽ hiện $ranhư hình bên. Bắt đầu lập trình



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
```

Kết quả như sau.

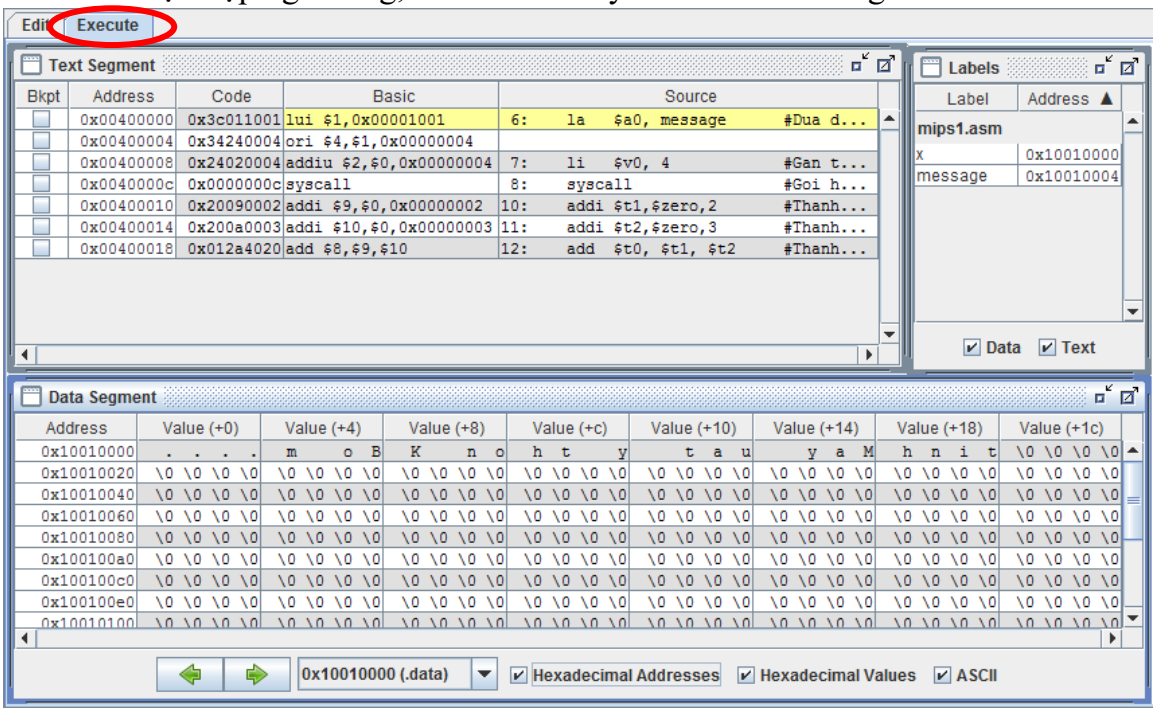
Edit | Execute

mips1.asm*

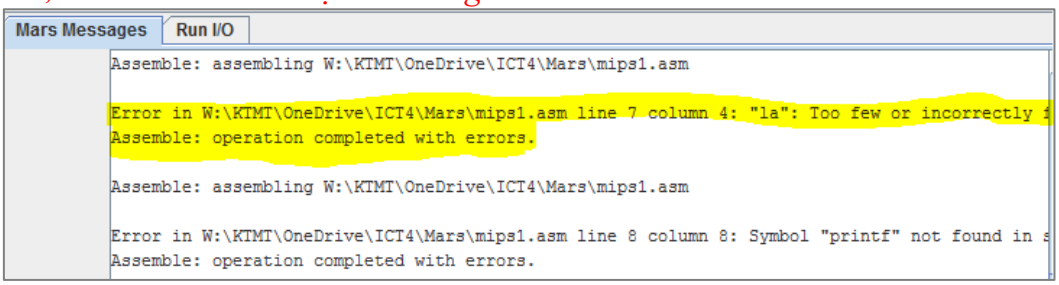
```
1   .data                    # Vung du lieu, chua cac khai bao bien
2   x: .word     0x01020304    # bien x, khoi tao gia tri
3   message:  .asciiz  "Bo mon Ky thuat May tinh"
4
5   .text                    # Vung lenh, chua cac lenh hop ngu
6      la    $a0, message      #Dua dia chi bien mesage 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
```

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.

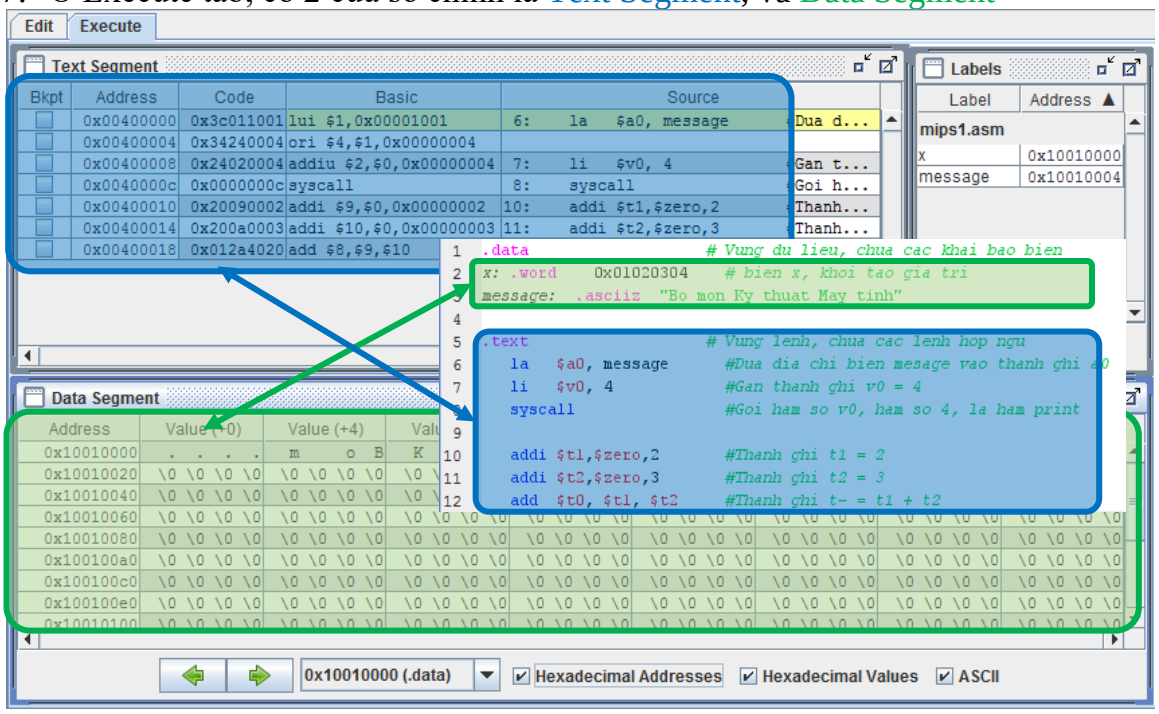
6. Nếu đoạn hợp ngữ đúng, MARS sẽ chuyển từ Edit tab sang Execute tab.



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.



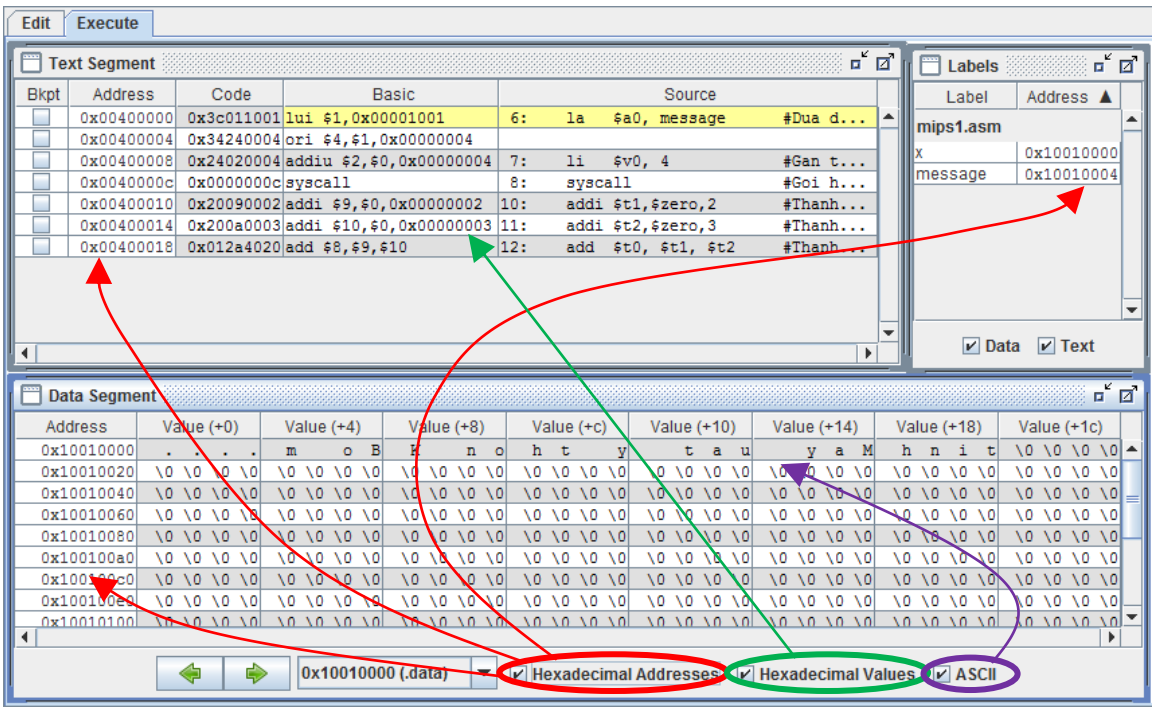
7. Ở Execute tab, có 2 cửa sổ chính là Text Segment, và Data Segment



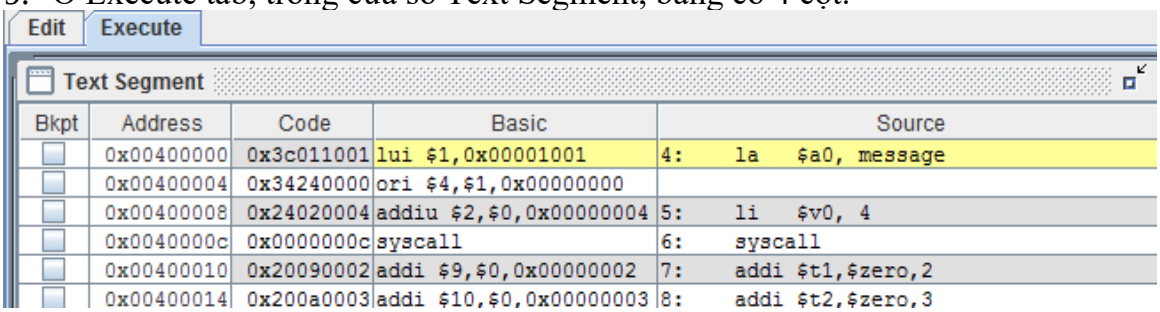
- **Text Segment:** là vùng không gian bộ nhớ chứa các mã lệnh hợp ngữ. Tương ứng với mã nguồn hợp ngữ, các dòng nào viết sau chỉ thị **.TEXT** tức là lệnh và sẽ thuộc Text Segment.
- **Data Segment:** là vùng không gian bộ nhớ chứa các biến. Tương ứng với mã nguồn hợp ngữ, các dòng nào viết sau chỉ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

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

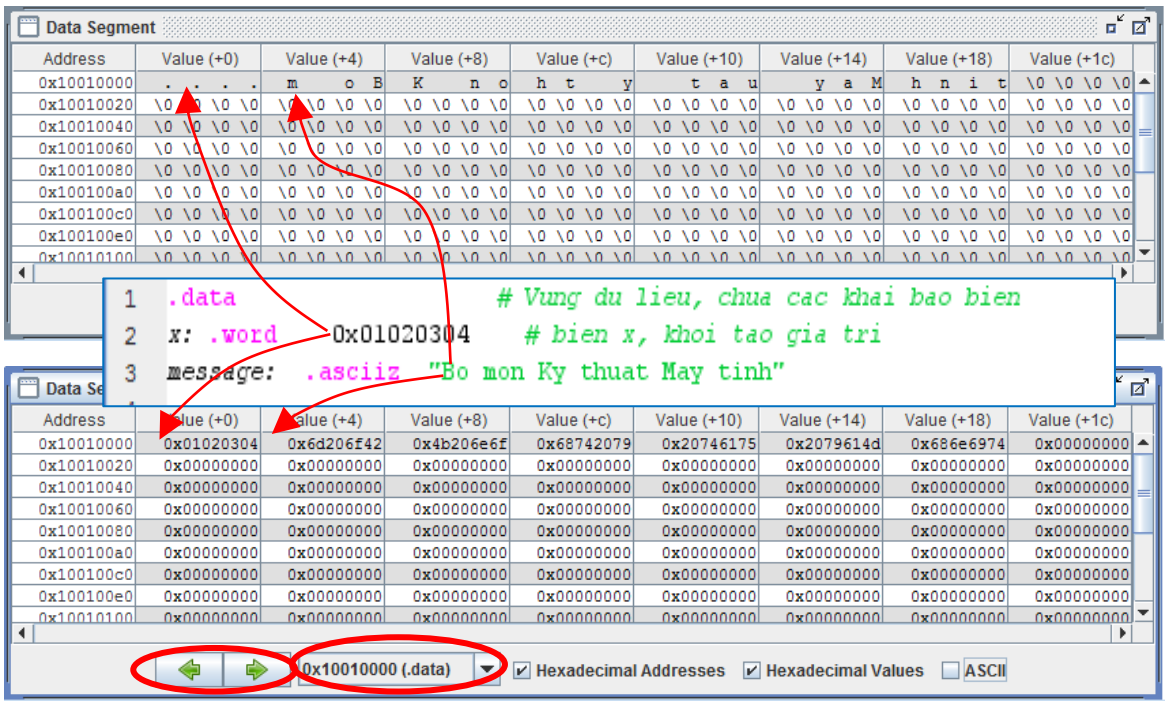


9. Ở Execute tab, trong cửa sổ Text Segment, bảng có 4 cột.



- **Bkpt**: Breakpoint, điểm dừng khi chạy toàn bộ chương trình chương trình bằng nút .
- **Address**: địa chỉ của lệnh ở dạng số nguyên *(xem thêm hướng dẫn về cửa sổ Label)*
- **Code**: lệnh ở dạng mã máy
- **Basic**: lệnh ở dạng hợp ngữ thuần, giống như qui định trong tập lệnh. Ở đây, tất cả các nhãn, tên gọi nhớ.. đều đã được chuyển đổi thành hằng số.
- **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. Trong ảnh minh họa bên dưới:
    i. 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.
    ii. Nhãn message trong lệnh la $a0, message trong cột Source → được dịch thành hằng số 0x00001001 *(xem thêm hướng dẫn về cửa sổ Label)*

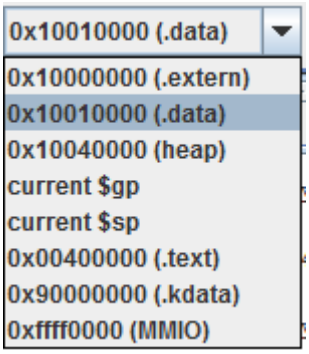
10. Ở Execute tab, trong cửa sổ Data Segment, bảng có 9 cột



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

Trong hình ảnh trên, 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 ☐ ASCII, và giá trị của chuỗi "Bo mon Ky thuat May tinh" khi hiển thị ở dạng kí tự ☑ ASCII. *Lưu ý rằng việc lưu trữ chuỗi trong bộ nhớ ở dạng little-endian là do cách lập trình hàm phần mềm syscall, chứ không phải do bộ xử lý MIPS qui định. Có thể thấy, ở công cụ giả lập MIPS IT, hàm print lại qui định chuỗi theo kiểu big-endian.*

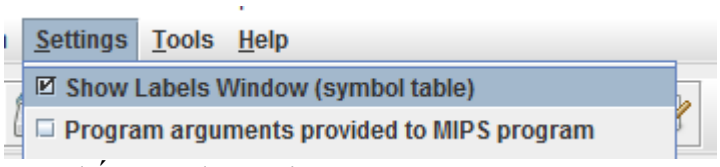
Bấm vào cặp nút để dịch chuyển tới vùng địa chỉ lân cận



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

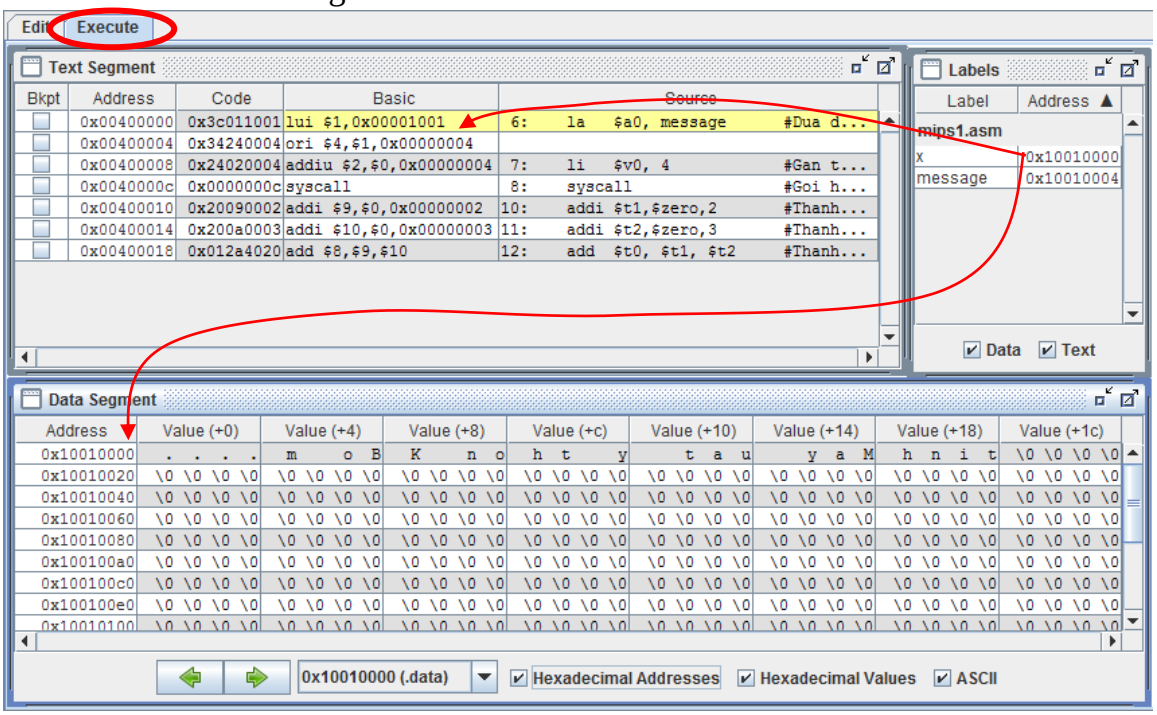
- .data: vùng dữ liệu
- .text: vùng lệnh
- $sp: vùng ngăn xếp

11. Cửa sổ Label: hiển thị tên nhãn và hằng số địa chỉ tương ứng với nhãn khi được biên dịch $ramã máy.
    Cửa sổ Label không tự động hiển thị. Phải vào menu Settings / chọn Show Labels Windows



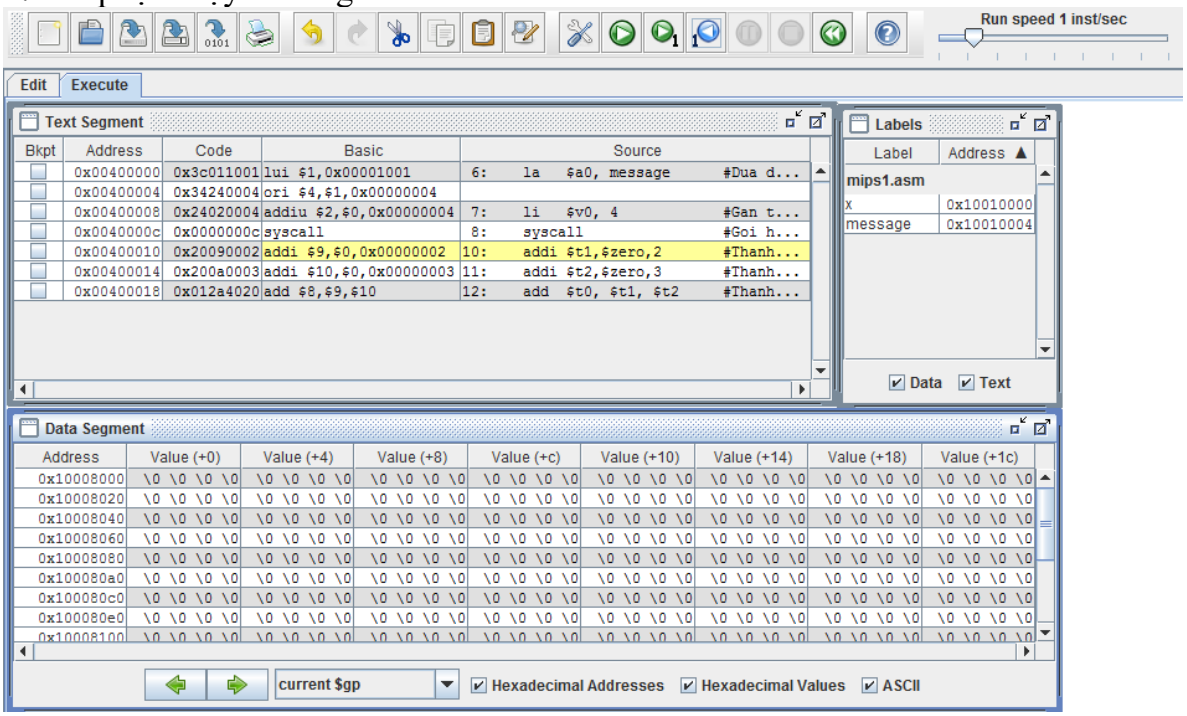
Trong ảnh sau, ta thấy các thông tin quan trọng :

- Trong cửa sổ Labels cho biết :
  - X chỉ là tên gợi nhớ, x sẽ được qui đổi thành hằng số 0x10010000.
  - Message cũng chỉ là tên gợi nhớ, sẽ được qui đổi thành hằng số 0x10010004
  - Click đúp vào tên biến, sẽ tự động chuyển sang vị trí tương ứng trong cửa sổ Data Segment.
- Trong cửa sổ Text Segment cho biết
  - Quả thực, ở lệnh gán *la $a0, message* tên gợi nhớ message đã được chuyển thành hằng số 0x1001004 thông qua cặp lệnh lui, ori
- Trong cửa sổ Data Segment cho biết
  - Quả thực, để giám sát giá trị của biến X, ta mở Data Segment ở hằng số 0x10010000 sẽ nhìn thấy giá trị của X.
  - Quả thực, để giám sát giá trị của biến message, ta mở Data Segment ở hằng số 0x10010004 sẽ nhìn thấy giá trị của message.

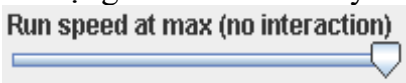


Chạy giả lập

1. Tiếp tục chạy chương trinh Hello World ở trên.



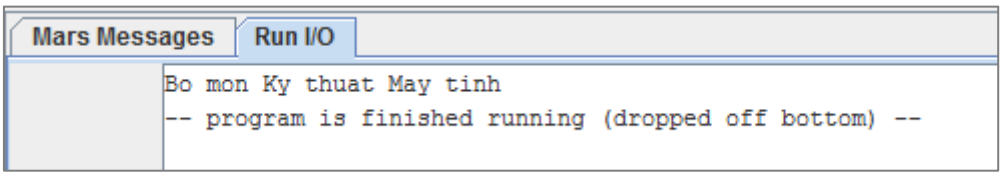
2. Sử dụng slider bar để thay đổi tốc độ thực thi lệnh hợp ngữ.



Mặc định, tốc độ thực thi là tối đa, và ở mức này, ta 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.

3. Ở Execute tab, chọn cách để thực thi chương trình

- Bấm vào icon , Run, để thực hiện toàn bộ chương trình. Khi sử dụng Run, ta quan sát dòng lệnh được tô màu vàng cho biết chương trình hợp ngữ đang được xử lý tới chỗ 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.
- Sau khi chạy xong tất cả các lệnh của phần mềm Hello Word, sẽ thấy cửa sổ Run I/O hiển thị chuỗi.



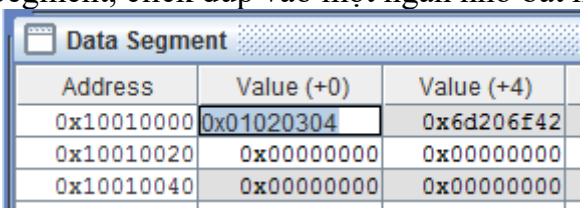
Giả lập & gỡ rối: quan sát sự thay đổi của các biến

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, và hiểu rõ ý nghĩa của sự thay đổi đó.

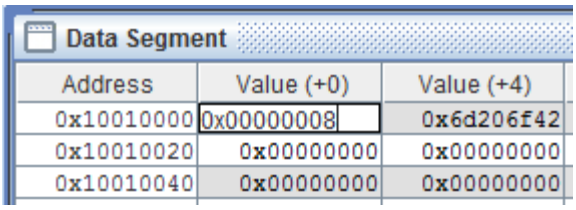
Giả lập & gỡ rối: thay đổi giá trị biến khi đang chạy run-time

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

1. Trong Data Segment, click đúp vào một ngăn nhớ bất kì



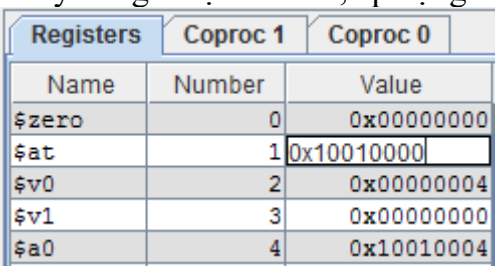
2. Nhập giá trị mới



3. Tiếp tục chạy chương trình

Giả lập & gỡ rối: thay đổi giá trị thanh ghi khi đang chạy run-time

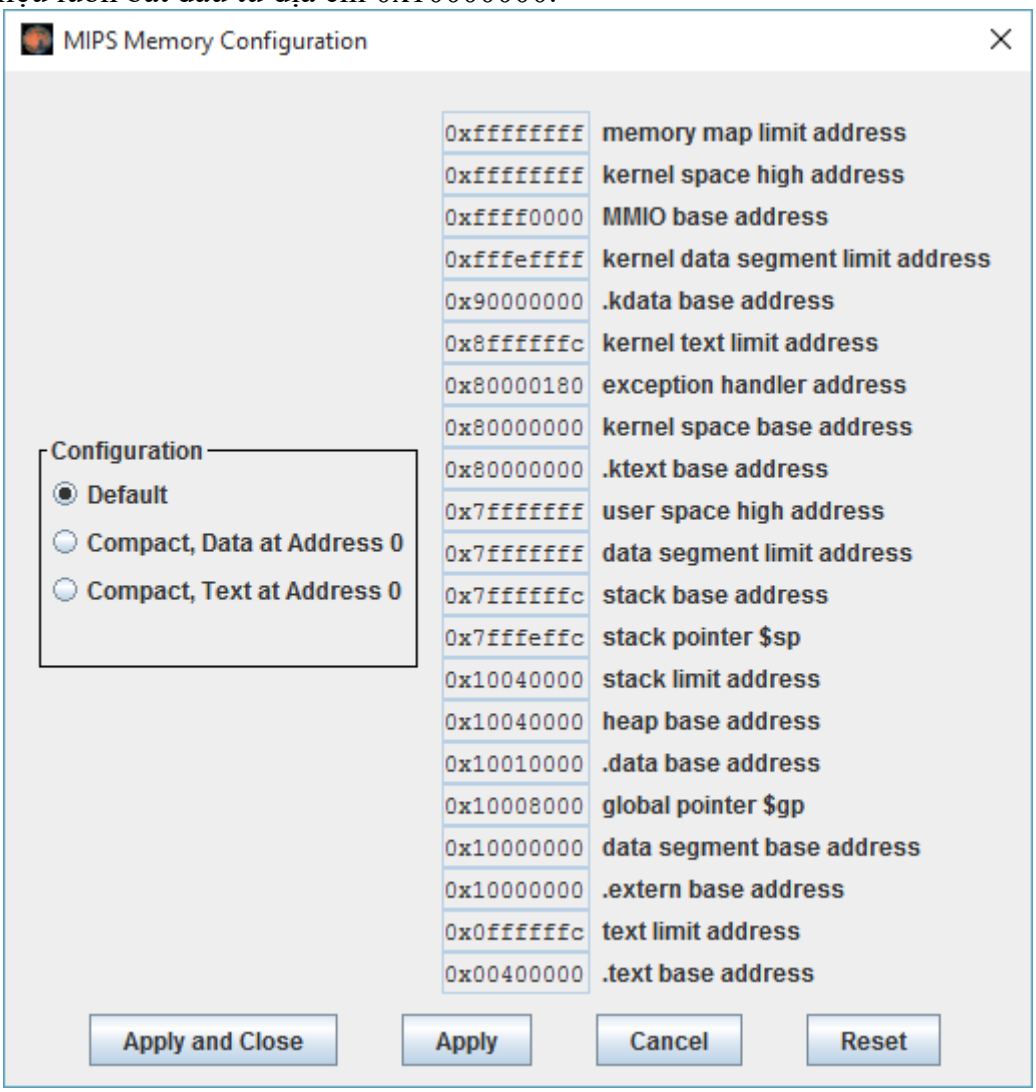
Cách làm tương tự như thay đổi giá trị của biến, áp dụng cho cửa sổ Registers



Tra cứu Help

Bâm 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…
Cửa sổ MIPS Memory Configuration chứa bảng qui định các hằng địa chỉ mà công cụ 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.