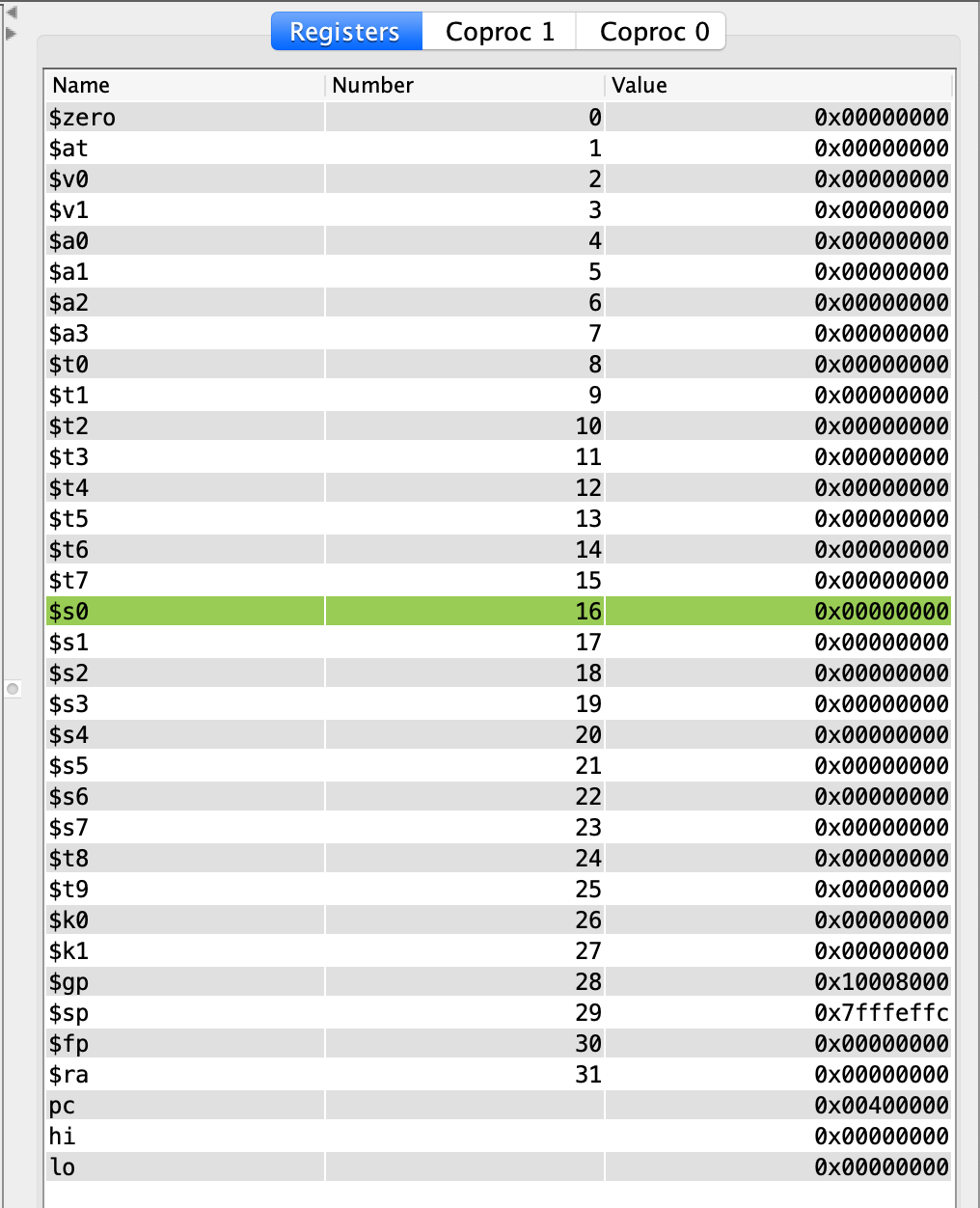
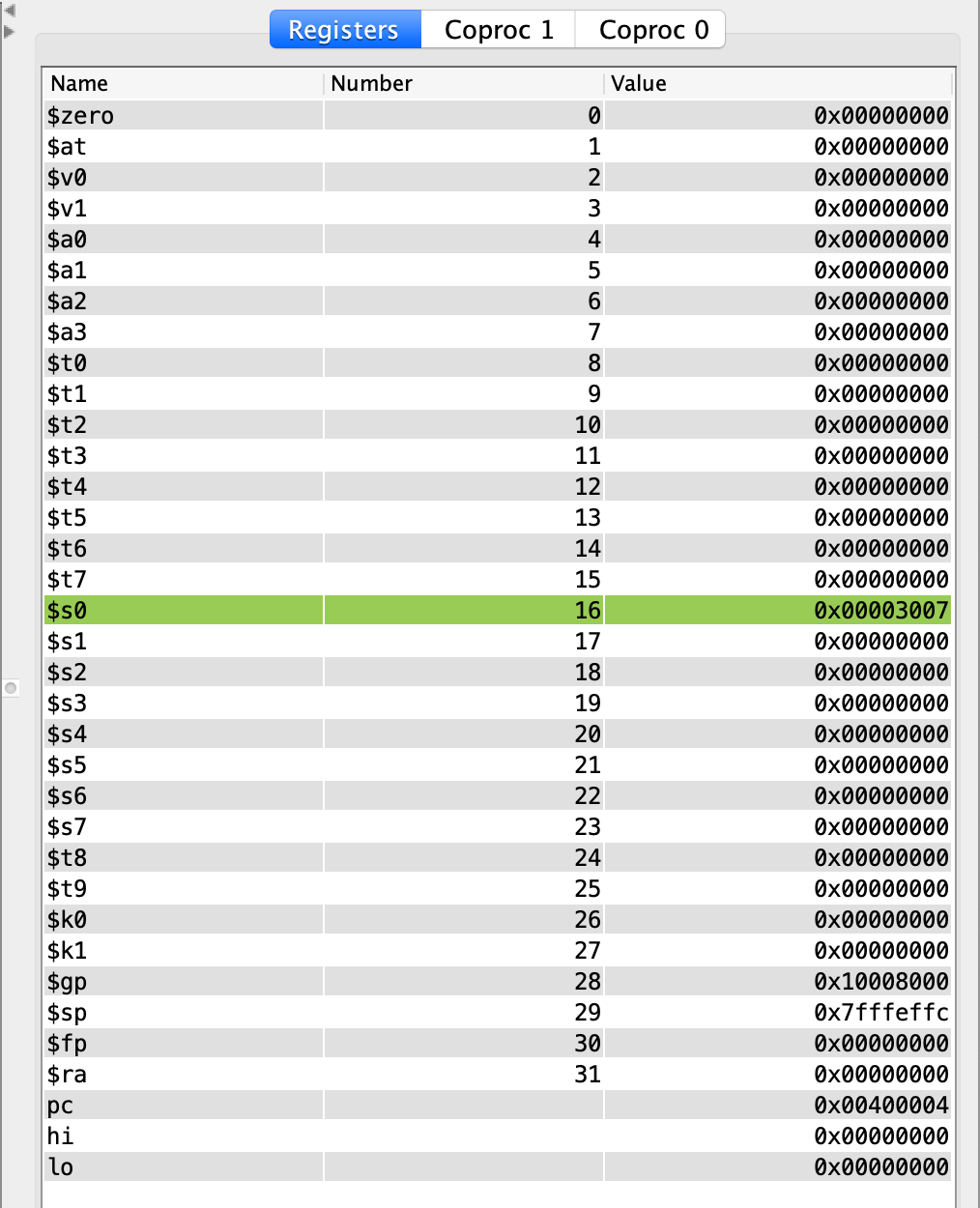
**Assignment 1: Lệnh gán số 16-bit**

****

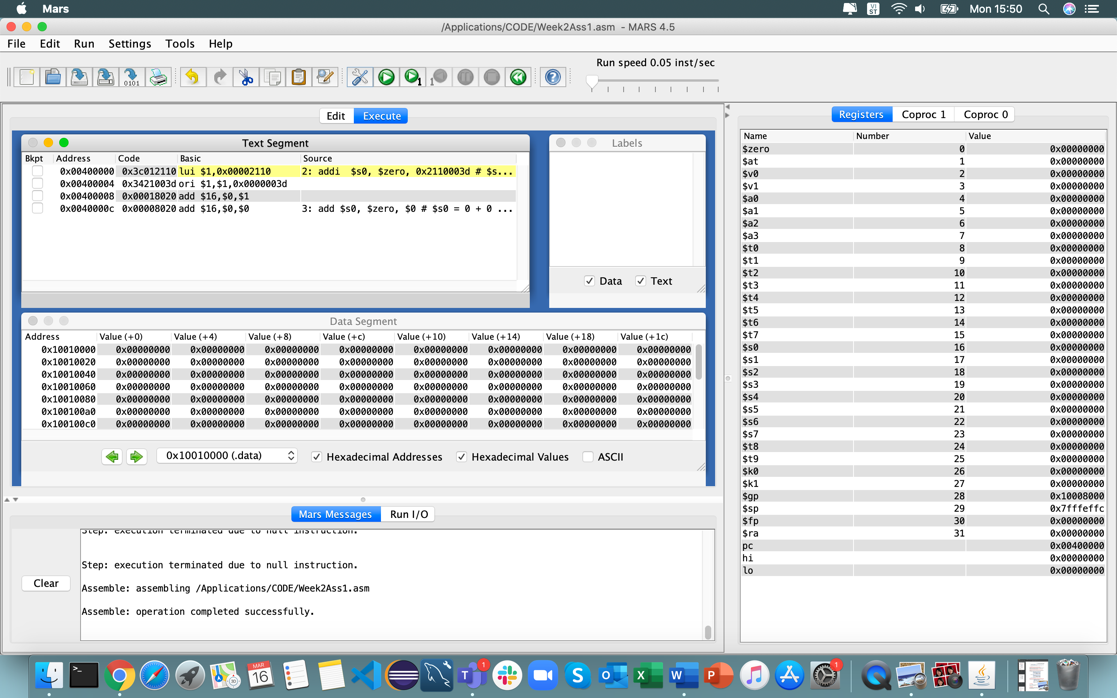
* Sự thay đổi giá trị của thanh ghi $s0 sau mỗi lệnh: 0x00000000 -> 0x00003007
* Sự thay đổi giá trị cuả thanh ghi $pc sau mỗi lệnh: 0x00400000 -> 0x00400004
* So sánh mã máy của các lệnh
  + Lệnh 1 Mã máy : 0x20103007 -> 00100000000100000011000000000111

|  |  |  |  |
| --- | --- | --- | --- |
| 001000 | 00000 | 01000 | 00011000000000111 |
| Opcode | Rs | Rt | Immediate |
| Op=8 |  |  |  |

* + Lệnh 2 – Mã máy = 0x00008020 -> 00000000000000001000000000100000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 000000 | 00000 | 00000 | 10000 | 00000 | 100000 |
| Opcode | rs | rt | rd | unused | add |
| Op=o | 0 -> $0 | 0 -> $0 | 16 -> $s0 |  | Add=32 |

* Sau khi sửa lại lệnh lui



Ta thấy lệnh gán giá trị 0x2110003d vào thanh ghi $s0 đã được tách ra thành 3 lệnh gán

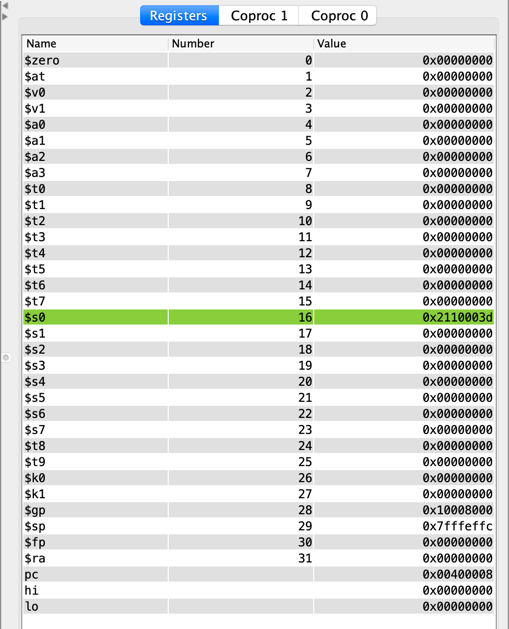
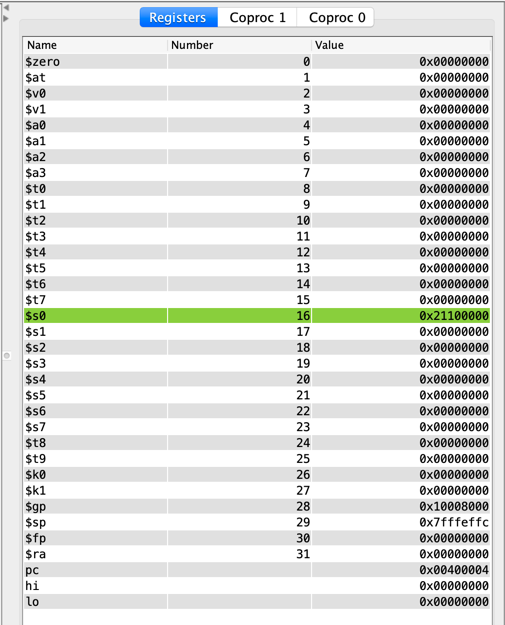
lui $1, 0x00002110

ori $1, $1, 0x0000003d

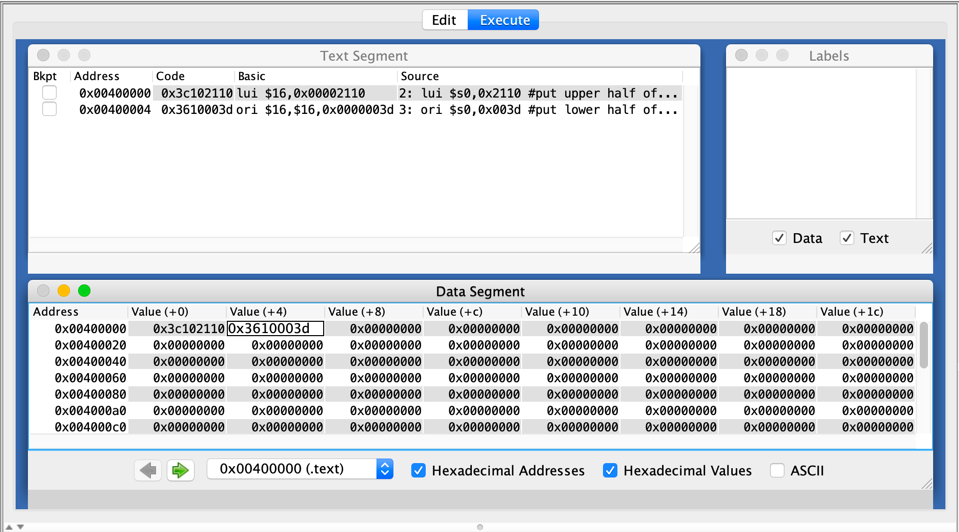
add $16, $0, $1

do các lệnh thao tác với hằng số đều có giới hạn 16 bit nên khi ta gán một gía trị 32-bit vào thanh ghi, MIPS cung cấp lệnh lui ghi một hằng số 16-bit vào 16-bit cao vào , ori dùng để gán 16-bit thấp của thanh ghi.

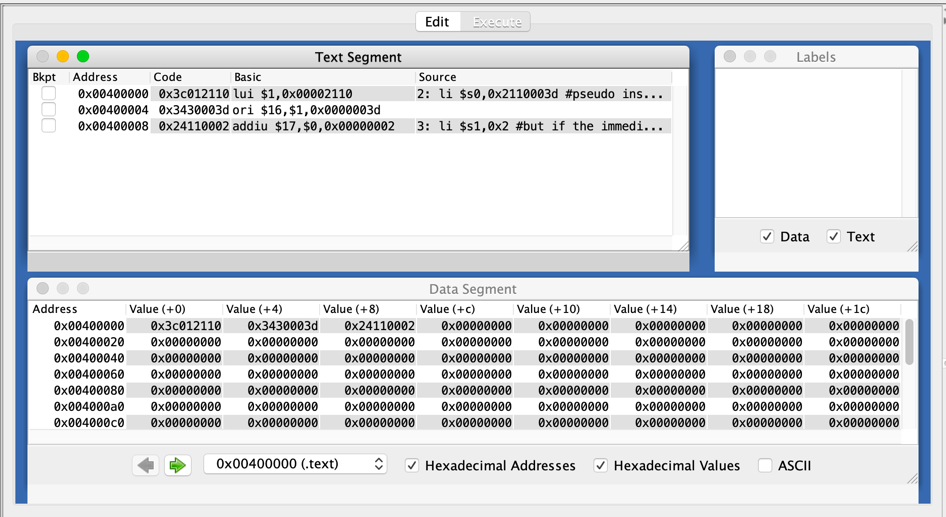
**Assignment 2: Lệnh gán số 32-bit**

****

* Sự thay đổi giá trị của thanh ghi $s0 sau mỗi lệnh: 0x21100000 -> 0x2110003d
* Sự thay đổi giá trị cuả thanh ghi $pc sau mỗi lệnh: 0x00400004 -> 0x00400008
* Các byte đầu tiên ở vùng lệnh trùng với cột mã máy của các lệnh trên cửa sổ Text Segment



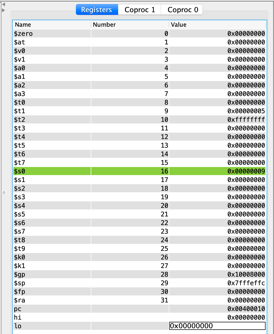
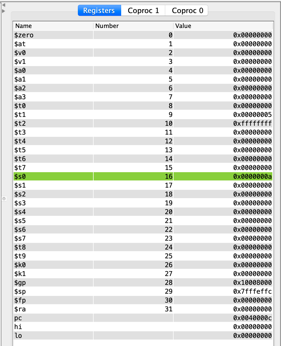
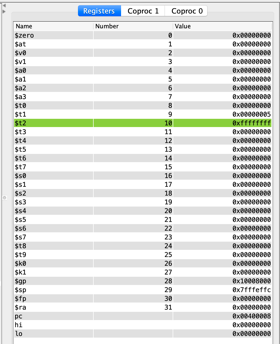
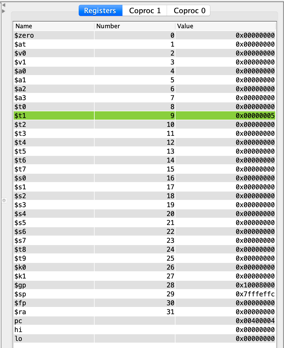
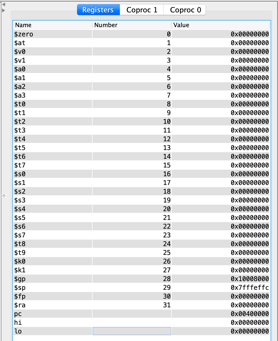
**Assignment 3: Lệnh gán ( Giả lệnh)**



Ta nhận thấy rằng lệnh li không được hiển thị trên cửa sổ Text Segment, do lệnh li là lệnh giả ( mã giả ).

Điều bất thường ở đây đó là 2 lệnh li , đều có mục đích khởi tạo thanh ghi với giá trị nhưng ở lệnh li đầu tiên , giá trị truyền vào có độ dài 32 bit nên MIPS đã cung cấp lệnh lui và ori với mục đích lưu lần lượt 16 bit cao và 16 bit thấp của giá trị cần truyền vào.

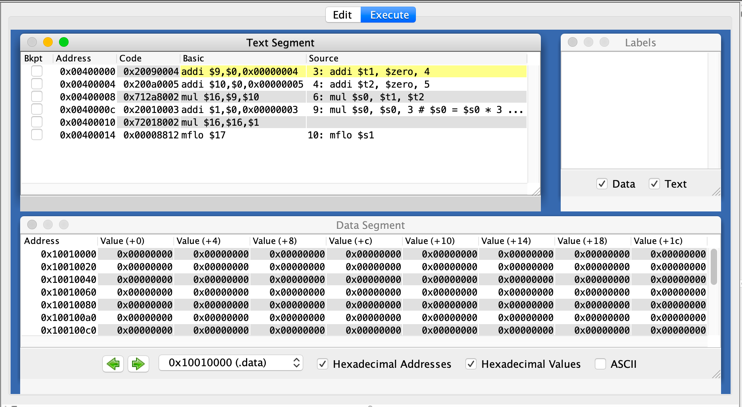
**Assignment 4: Tính biểu thức 2x + y ?**

****

Sự thay đổi của các thanh ghi được biểu diễn như hình trên. Sau khi kết thúc chương trình , kết quả thu được của phép tính 2X + Y với tham số truyền vào X =5 và Y = -1 là 9 => Đúng!

Ở 2 lệnh thuộc kiểu lệnh addi trong cửa sổ Text Segment ta thấy khi chuyển 2 mã máy của 2 lệnh đó giống nhau ( 001000 – opcode = 8) . Đúng với khuôn mẫu của kiểu lệnh I.

**Assignment 5: Phép nhân**

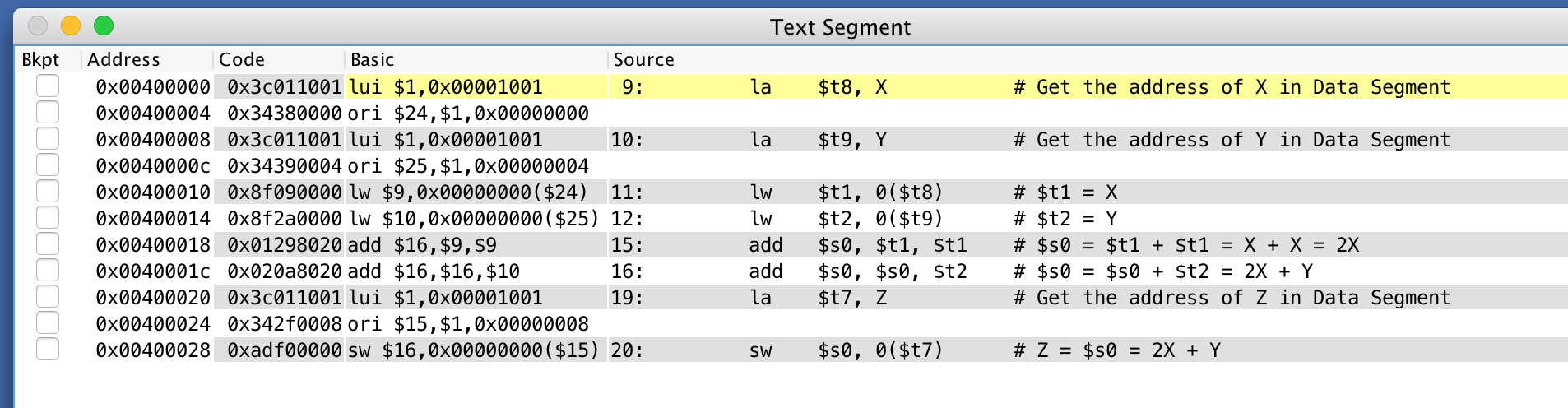
****

Điều bất thường đó là 2 lệnh mul được thực hiện, 1 lệnh có thêm lệnh addi còn một lệnh thì không. Cú pháp của lệnh mul là nhân giá trị của 2 thanh ghi và truyền vào 1 thanh ghi, nên với lệnh mul ở phía dưới, chúng ta phải truyền giá trị 3 vào một thanh ghi $1 trước khi thực hiện lệnh mul !

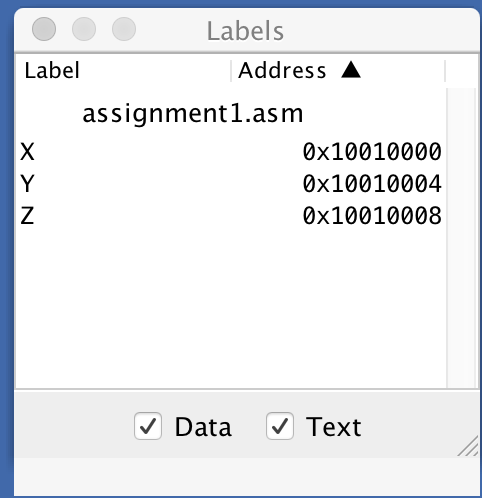
Lệnh thực hiện ra kết quả đúng !

**Assignment 6: Tạo biến và truy cập biến**

Kết quả chạy:



Của sổ label:



Nhận xét:

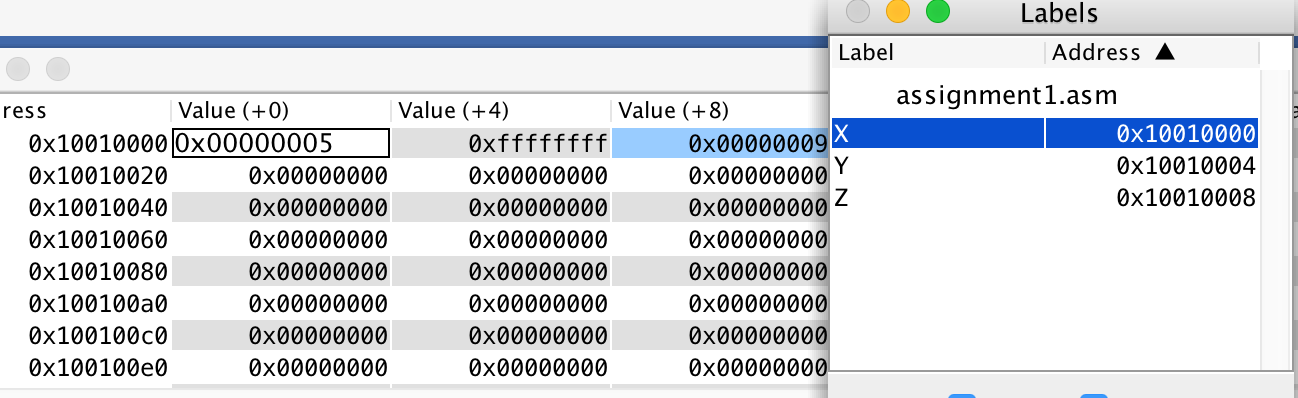
Lệnh la được biên dịch thành 2 lệnh: lui và ori.

Click đúp vào các biến X, Y, Z => Nó cho đúng giá trị khởi tạo: X = 5; Y = -1. Và sau khi chạy xong chương trình Z = 9

Z ban đầu = 0;



Sau khi chạy xong Z = 9;



**Lệnh lw và sw:**

+ lw: load word => đọc giá trị từ bộ nhớ đưa vào thanh ghi.

+ sw: store word => ghi giá trị thanh ghi vào bộ nhớ.

Thay đổi của thanh ghi:

Sau lệnh 1: 

Sau lệnh 1: 

Sau lệnh 2: 

Sau lệnh 3: 

Sau lệnh 4: 

Sau lệnh 5: 

Sau lệnh 6: 

Sau lệnh 7: 

Sau lệnh 8: 

Sau lệnh 9: 

Sau lệnh 10: 