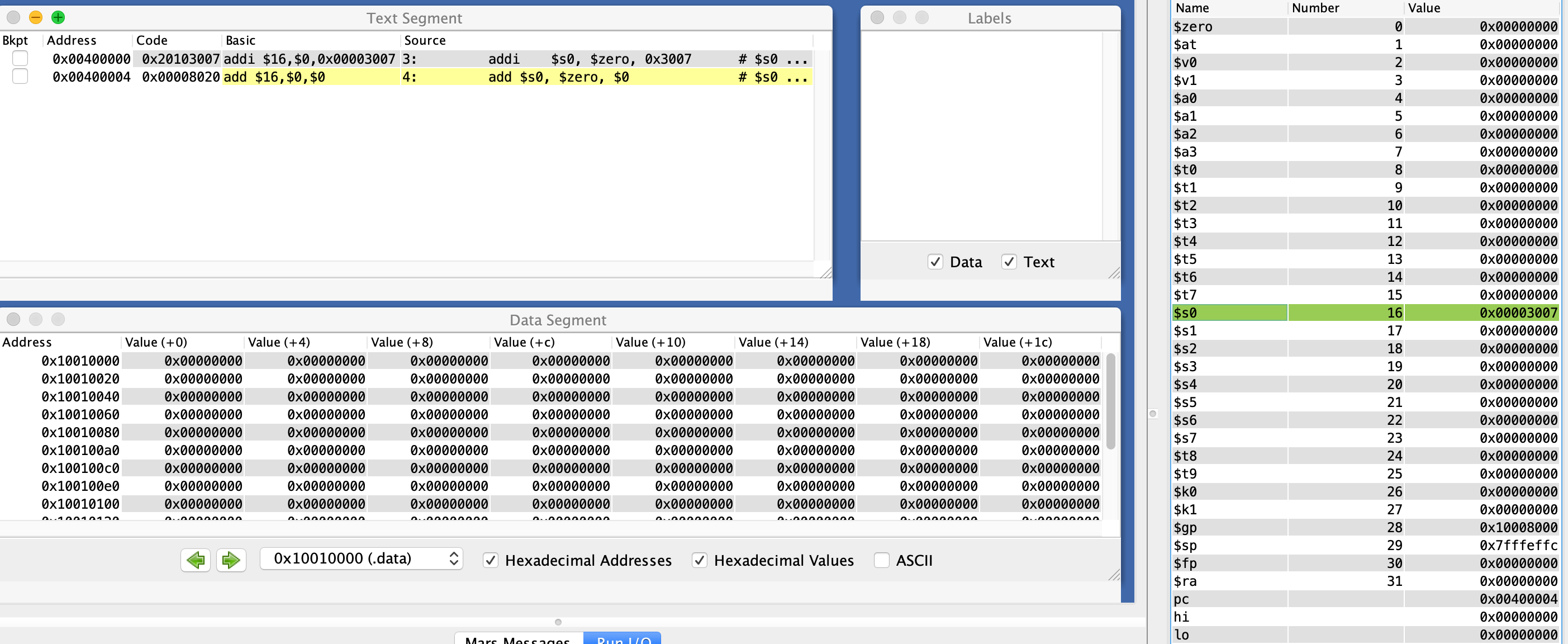


Báo Cáo Thực Hành

Môn: Thực Hành Kiến Trúc Máy Tính

**Assignment 1:**

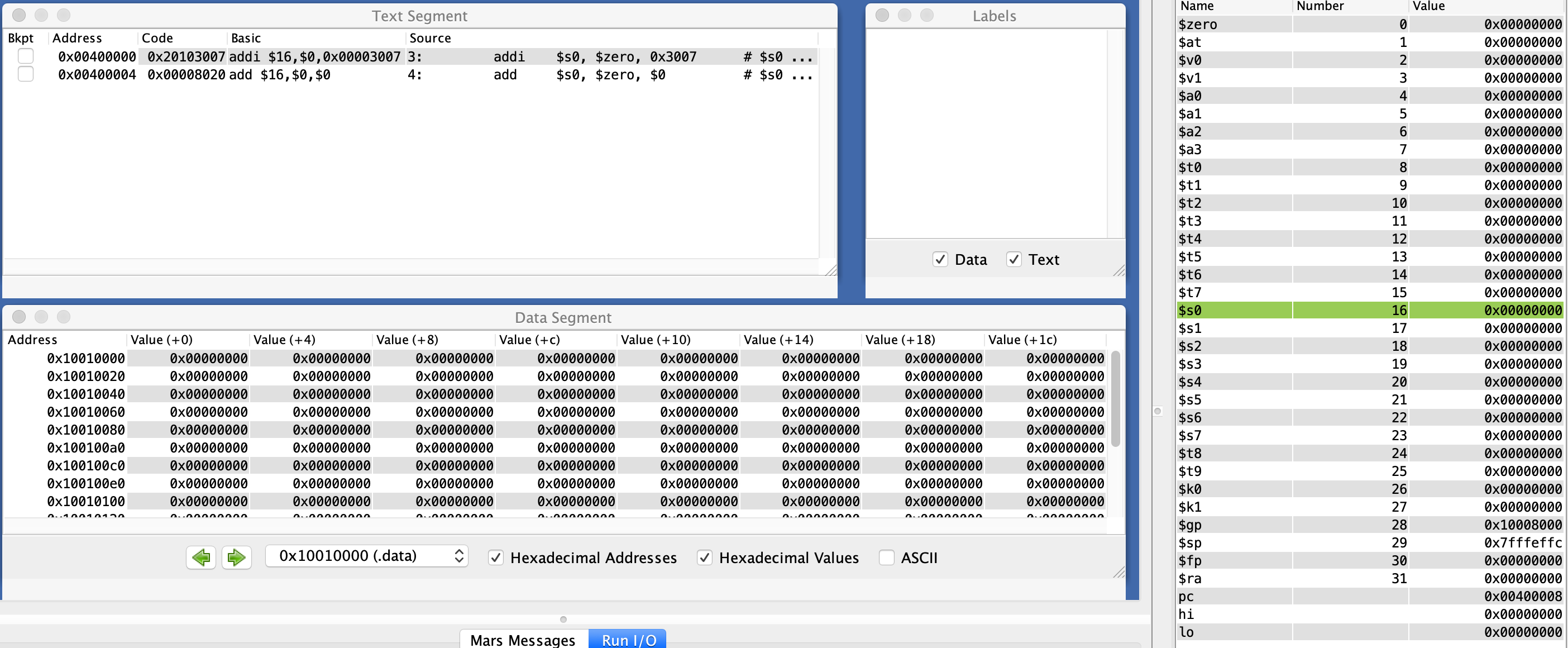
* Code ban đầu:
  + Dòng 1:



Thanh ghi $s0 thay đổi đúng bằng giá trị số đầu vào cộng với giá trị của thanh ghi $0 bằng 0x3007

Thanh ghi $pc chỉ đến địa chỉ của lệnh tiếp theo

* + Dòng 2:



Thanh ghi $s0 thay đổi đúng bằng giá trị thanh ghi $0

Thanh ghi $pc chỉ đến vùng nhớ tiếp theo ( cộng thêm 4 byte )

* Khuôn dạng tập lệnh
  + Lệnh addi $16, $0, 0x00003007:

20103007(16)

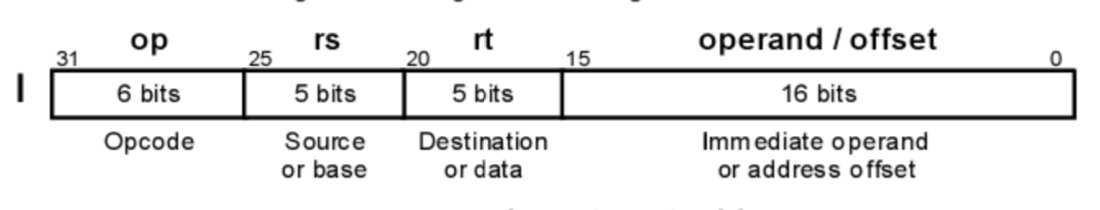
001000 00000 10000 0011000000000111 (2)

Opcode = 8

rs = 0

rt = 16

offset = 00003007(16)



* Chuẩn theo khuân dạng mẫu I
  + Lệnh add $16, $0, $0

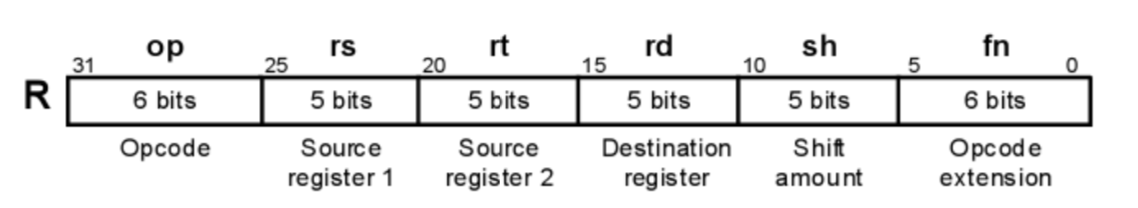
00008020(16)

000000 00000 00000 10000 00000 100000 (2)

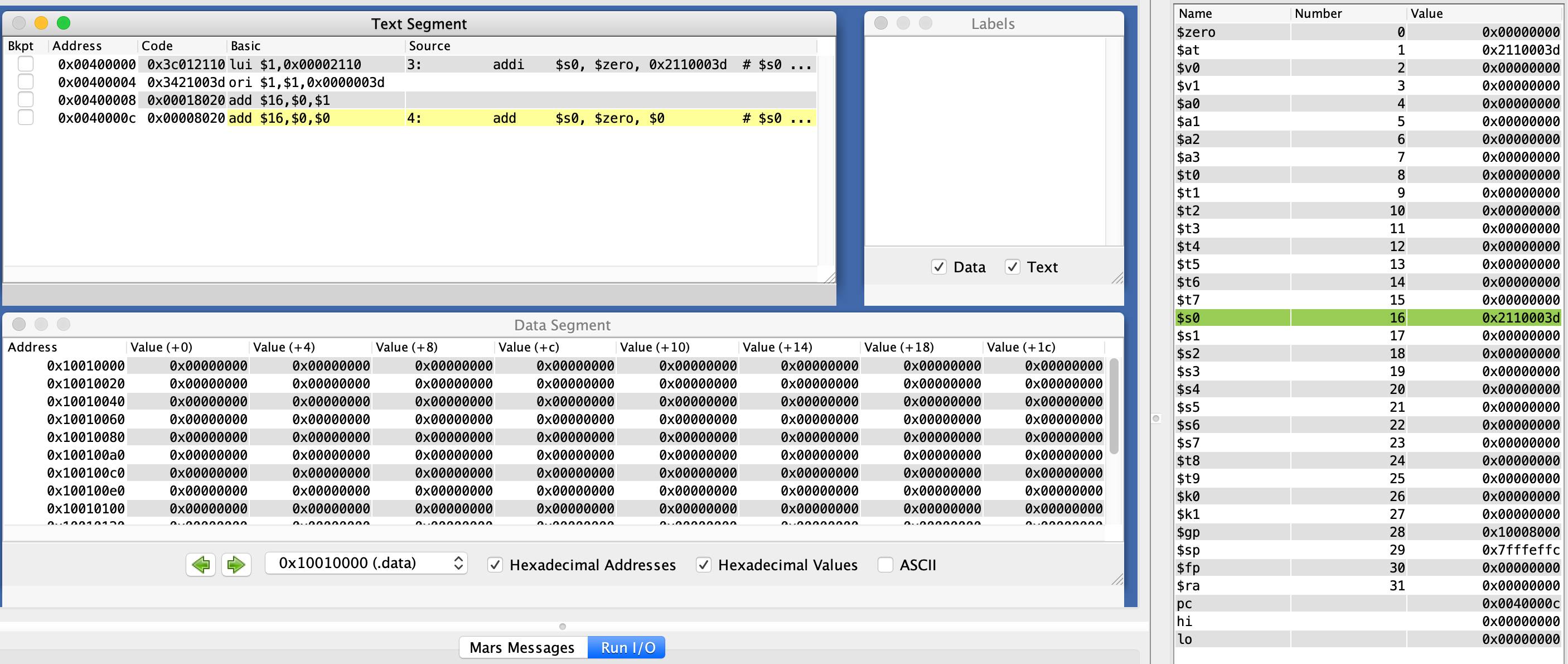
Opcode = 0, fn = 32

rs = 0, rt = 0

rd = 16



* Chuẩn khuôn dạng mẫu R
* Sửa thành addi $s0, $zero, 0x2110003d



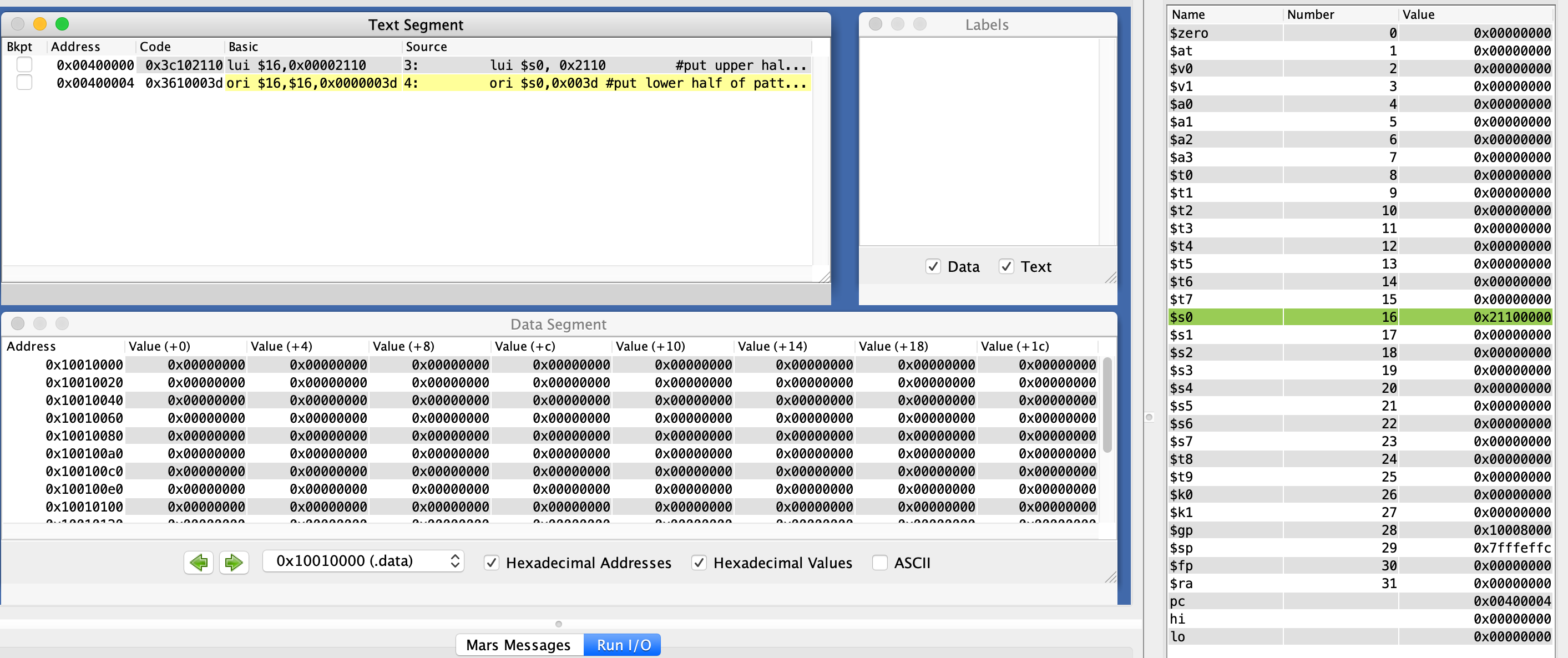
Câu lệnh addi sẽ được tách thành 3 câu lệnh nhỏ ở Basic

Do số đầu vào là 32bit(2) nên quy trình thực hiện sẽ là:

Lệnh lui nạp 16 bit trên gán vào $1, sau đó lệnh ori nối 16 bit dưới với 16 bit trên vào $1 và cuối cùng cho $1 + $0 gán vào $16

**Assignment 2:**

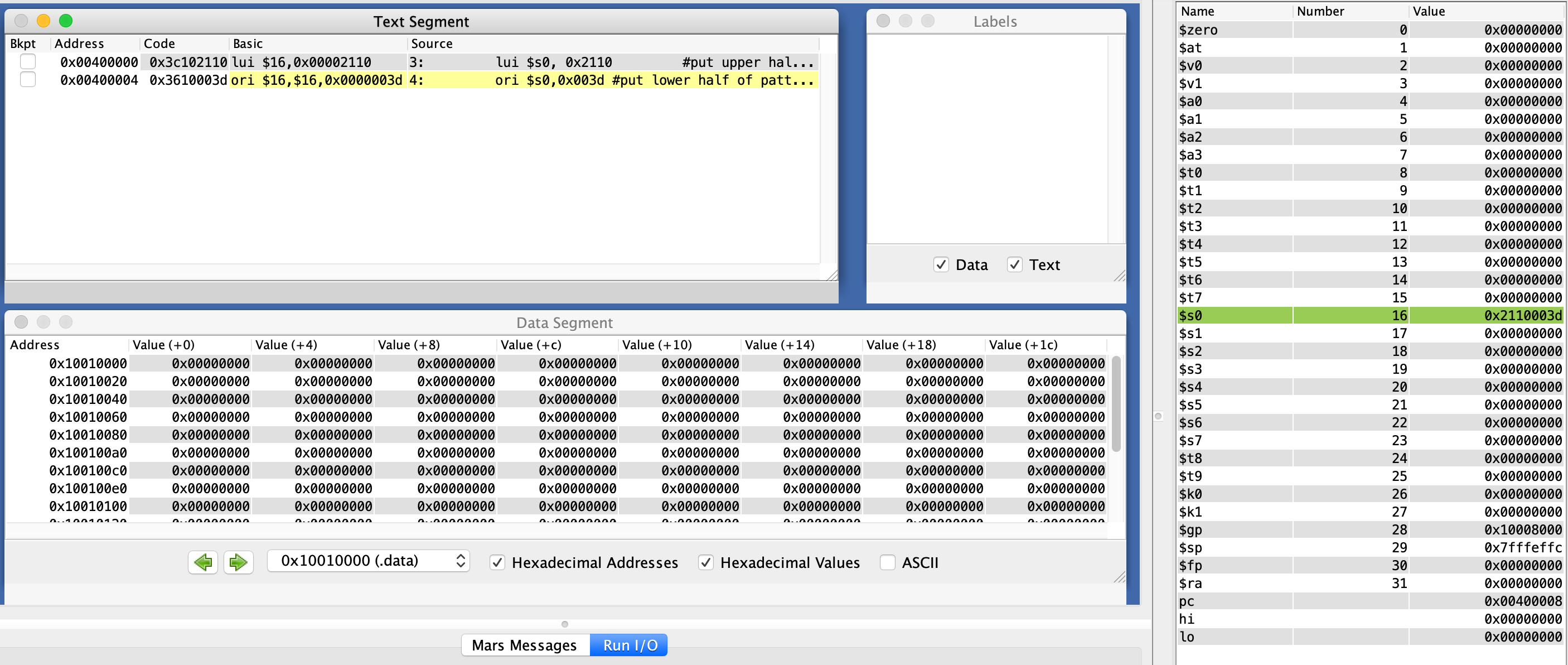
* Dòng 1:



Thanh ghi $s0 được gán là nửa trên từ giá trị ban đầu 0x00002110

Thanh ghi $pc chỉ đến địa chỉ của lệnh tiếp theo

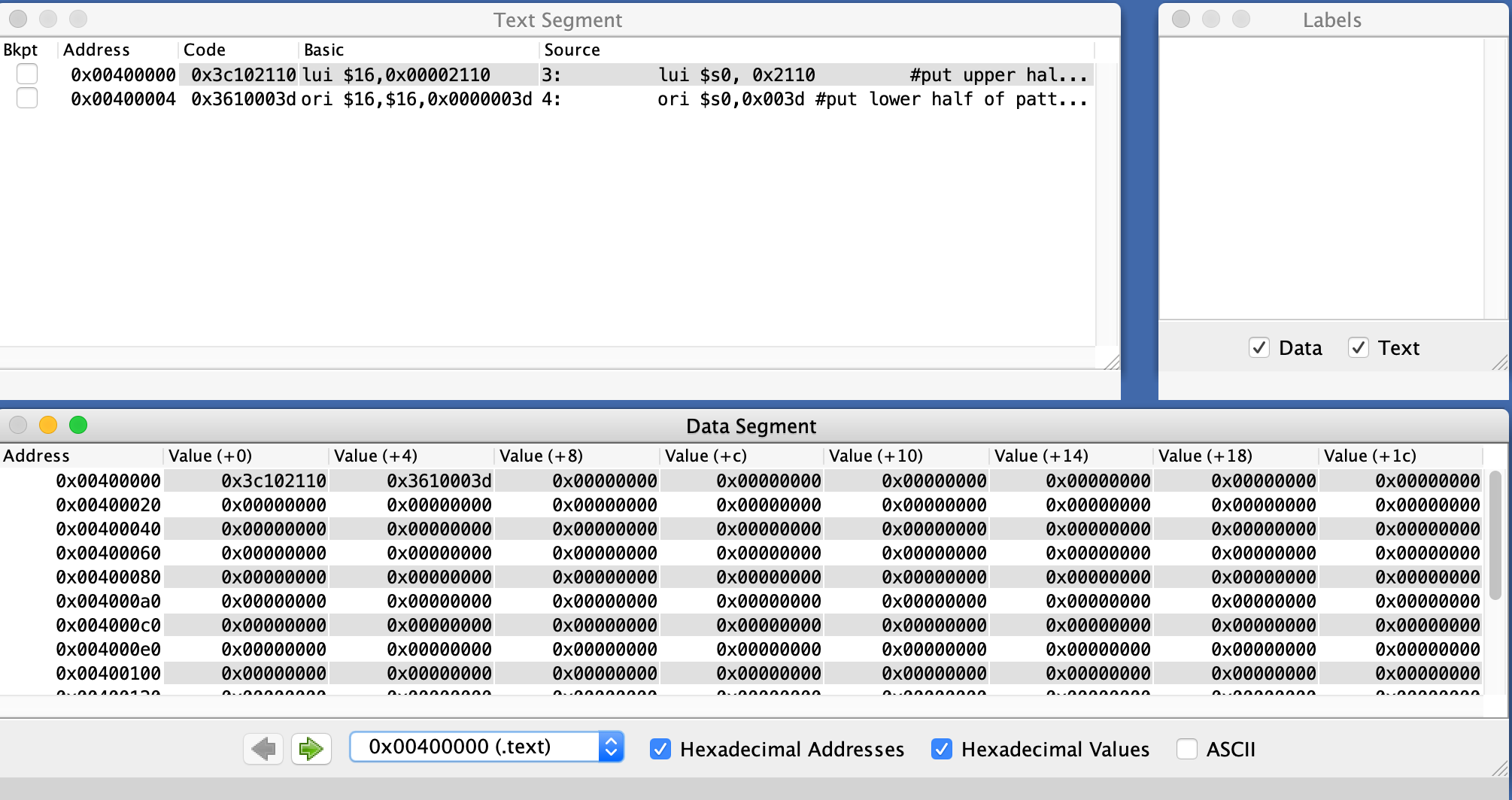
* Dòng 2:



Thanh ghi $s0 được gán nối tiếp nửa sau từ giá trị 0x0000003d

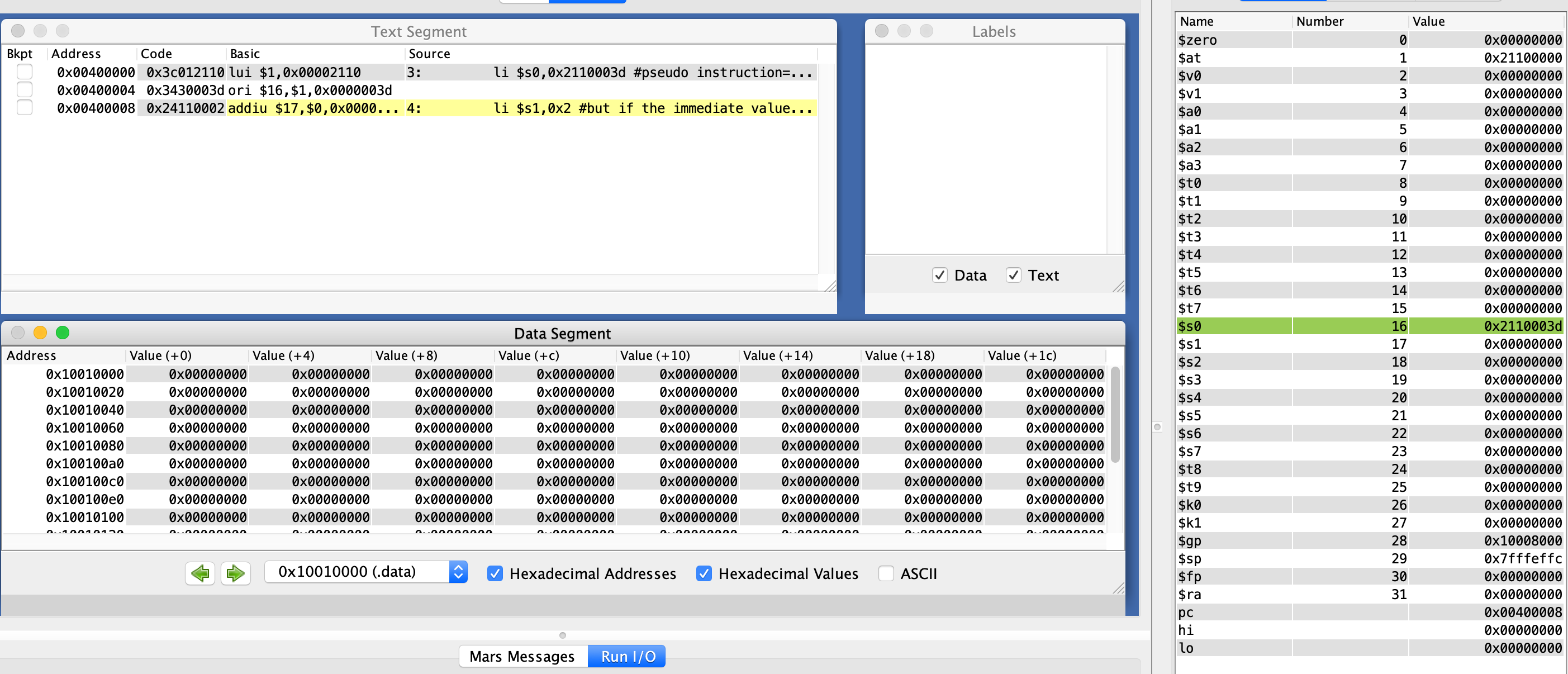
Thanh ghi $pc chỉ đến vùng nhớ tiếp theo ( cộng thêm 4 byte )

* Chuyển đết .text



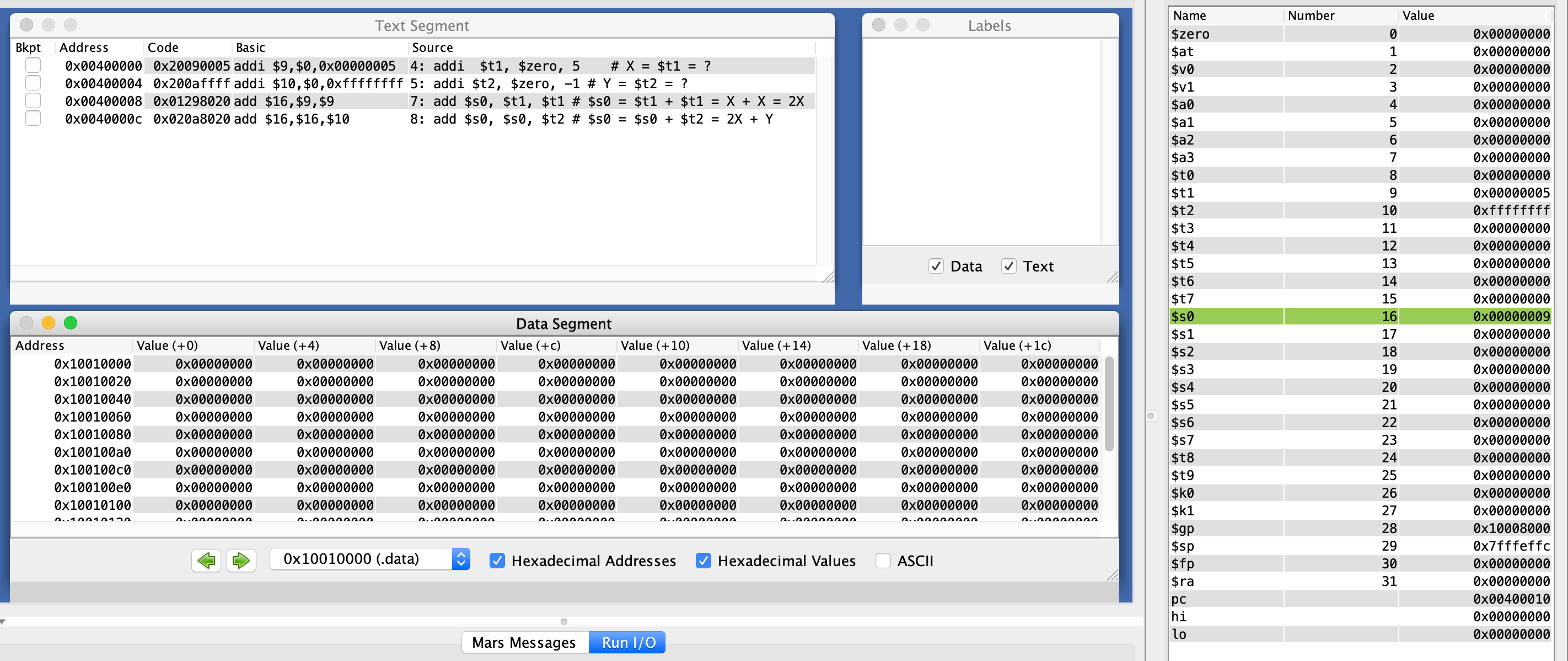
Byte đầu tiên cuả vùng lệnh trùng với hàng đầu tiên của cột Address trong Tt Segment

**Assignment 3:**



* Lệnh li $s0, 0x2110003d được tách thành 2 lệnh:
  + lui $1, 0x00002110
  + ori $16, $s1, 0x0000003d
* Vì lệnh I của MIPS chỉ có tham số 16 bits, nên được tách thành 2 lệnh.
* Lệnh li $s1, 0x2 có tham số nhỏ hơn 16 bits nên không cần tách.

**Assignment 4:**



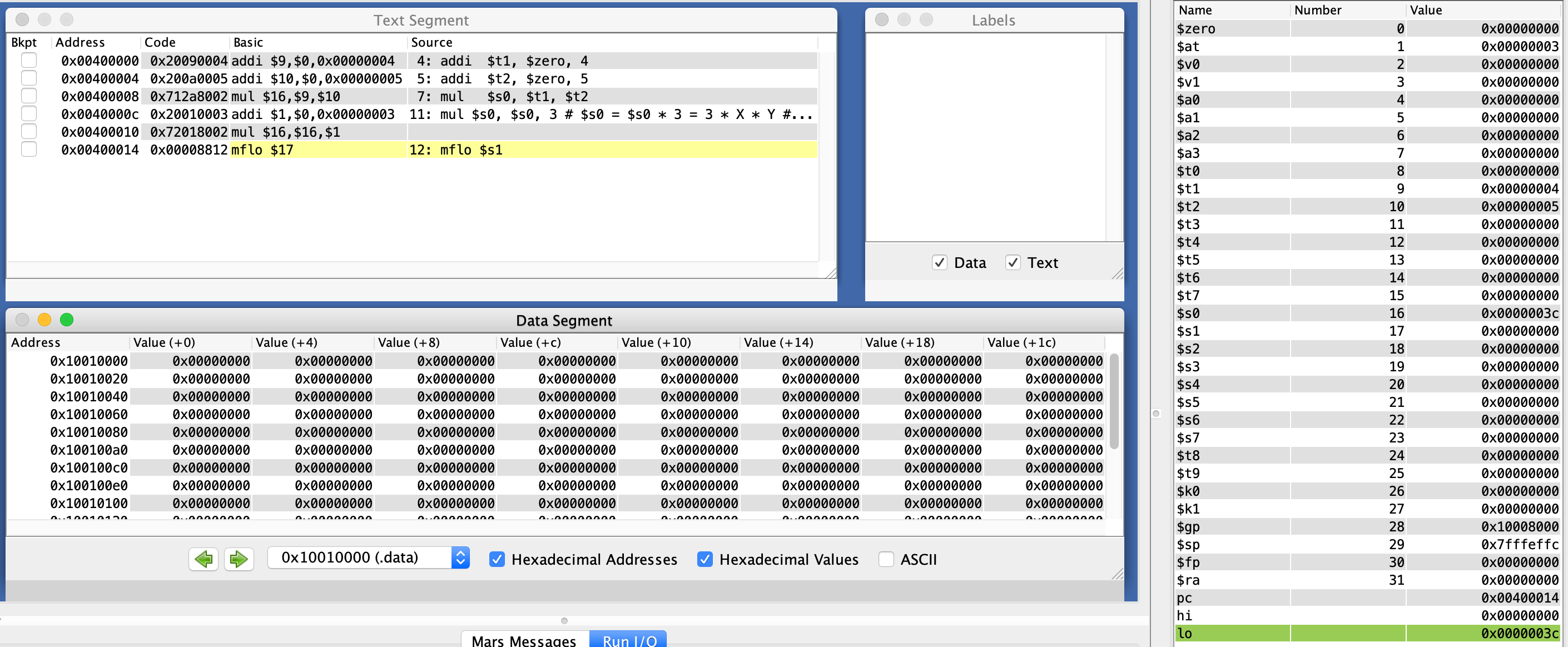
* Sau  lệnh 1 ,thanh $t1 có giá trị 0x00000005
* Sau lệnh 2, thanh $t2 có giá trị 0xfffffffff
* Sau lệnh 3, thanh $s0 có giá trị 0x0000000a

- Sau lệnh 4 , thanh $s0 có giá trị 0x00000009

=> Kết quả đúng

 Sau kiểm nghiệm , lệnh addi phù hợp với khuôn mẫu lệnh I, lệnh add phù hợp với khuôn mẫu lệnh R

**Assignment 5:**

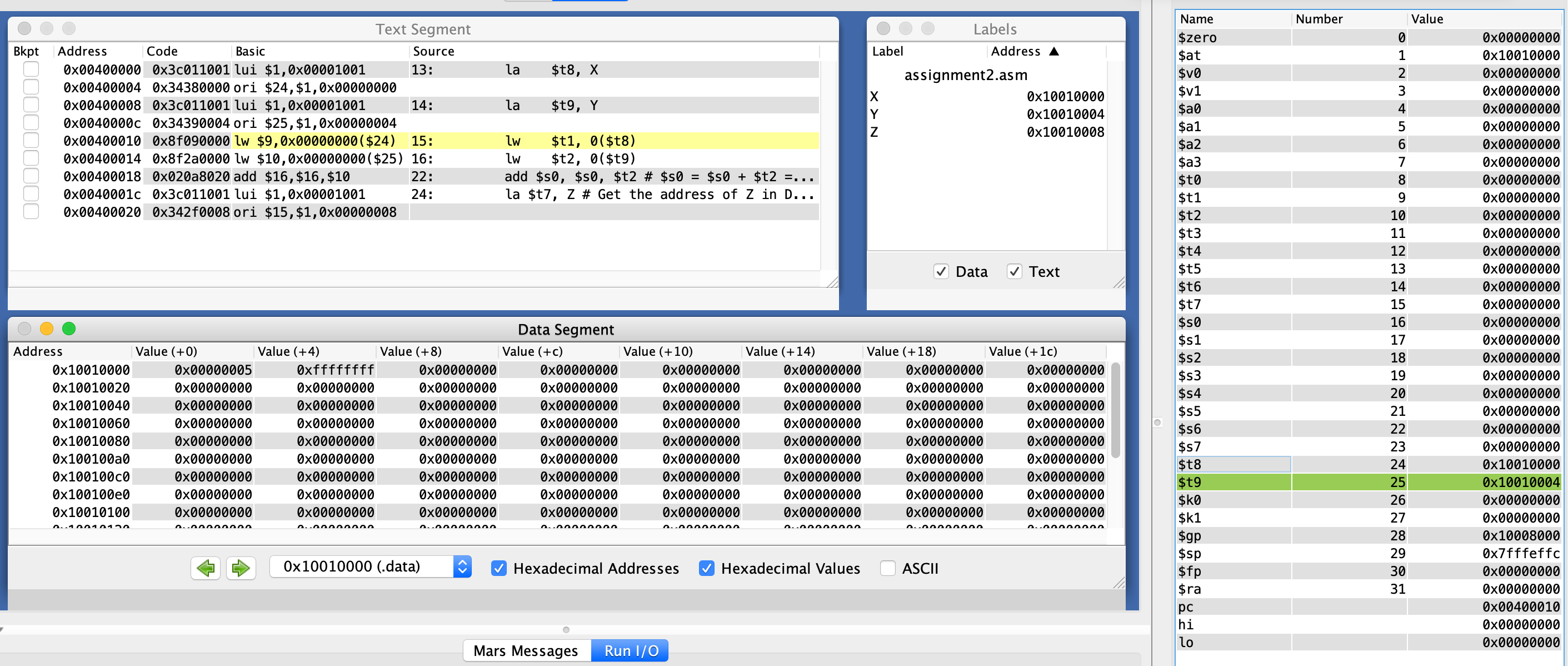


* Sau lệnh 1,2 $t1 và $t2 có giá trị lần lượt là 0x00000004 và 0x00000005
* Sau lệnh 3, $s0 và lo có giá trị là 0x00000014
* Lệnh 4 được tách làm 2 lệnh mul và addi.  Lệnh addi lưu 3 vào thanh ghi at. Sau đó lệnh mul mới tính tích của at và $s0. Sau lệnh  thanh lo, $s0 có giá trị 0x0000003c
* Lệnh cuối cùng thay đổi giá trị $s1 =$s0

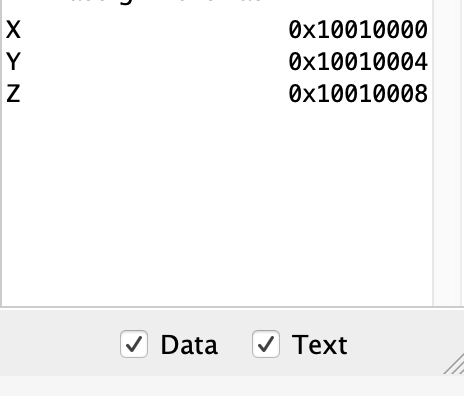
**Assignment 6:**

* Lệnh la được tách thành 2 lệnh lui và ori để load địa chỉ của biến x vì địa chỉ biến x là 32 bits.
* Địa chỉ của x, y, z được tách thành 2 nửa

|  |  |  |  |
| --- | --- | --- | --- |
| Biến | Địa chỉ | Nửa trên | Nửa dưới |
| x | 0x10010000 | 0x00001001 | 0x00000000 |
| y | 0x10010004 | 0x00001001 | 0x00000004 |
| z | 0x10010008 | 0x00001001 | 0x00000008 |



* Bảng label:



* Giá trị của x, y, z:

|  |  |
| --- | --- |
| Biến | Giá trị |
| x | 0x00000005 |
| y | 0xffffffff |
| z | 0x00000000 (không khởi tạo giá trị) |

* Lệnh lw: load giá trị kiểu word từ thanh ghi $t8, $t9 vào thanh ghi $t1, $t2.
* Lệnh sw: store giá trị kiểu word từ thanh ghi $s0 vào thanh ghi $t7.
* Lệnh lb: load giá trị kiểu byte vào một thanh ghi.
* Lệnh sb: store giá trị low-order 8 bits vào một thanh ghi.