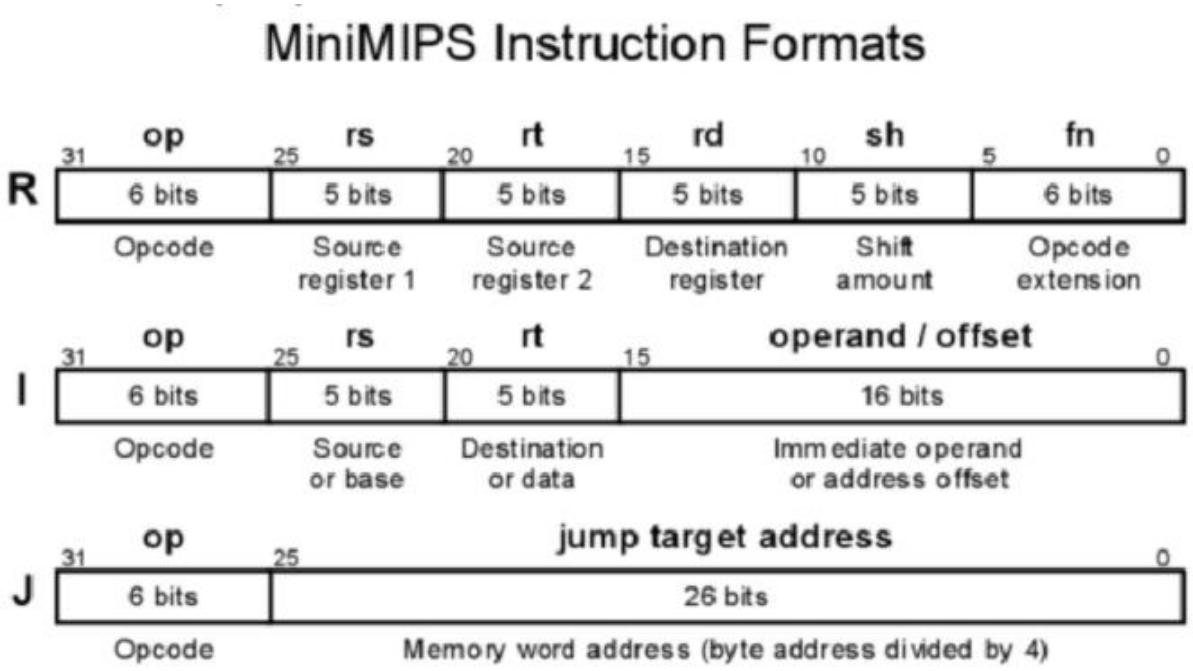
Báo cáo tuần 2

Home Assignment 1

- Tên và ý nghĩa của 32 thanh ghi:

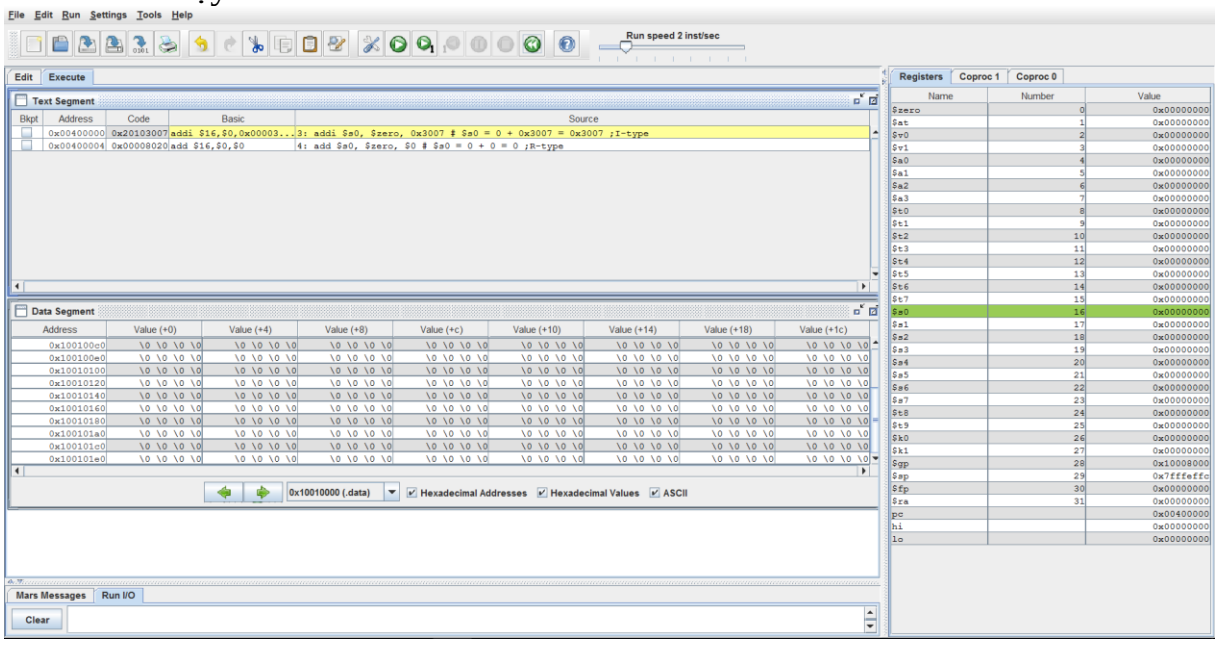
Tên thanh ghi	Số hiệu thanh ghi	Công dụng
$zero	0	Chứa hằng số bằng 0
$at	1	Giá trị tạm thời cho hợp ngữ
$v0-$v1	2-3	Các giá trị trả về của thủ tục
$a0-$a3	4-7	Các tham số vào của thủ tục
$t0-$t7	8-15	Chứa các giá trị tạm thời
$s0-$s7	16-23	Lưu các biến
$t8-$t9	24-25	Chứa các giá trị tạm thời
$k0-$k1	26-27	Các giá trị tạm thời của OS
$gp	28	Con trỏ toàn cục
$sp	29	Con trỏ ngăn xếp
$fp	30	Con trỏ khung
$ra	31	Địa chỉ trở về của thủ tục

- Các thanh ghi đặc biệt PC, HI, LO:
  - PC là thanh ghi của CPU giữ địa chỉ của lệnh cần nhận vào để thực hiện. CPU phát địa chỉ tfw PC đến bộ nhớ, lệnh được nhận vào. Sau khi lệnh được nhận vào, nội dung tự động tăng để trỏ sang lệnh kế tiếp. Sau mỗi lệnh, PC tăng 4.
  - Cặp thanh ghi HI/LO:
    - MIPS có hai thanh ghi 32 bit: HI (high), LO (low)
    - Tích 64bit nằm trong cặp thanh ghi HI/LO.
    - Trong phép chia thì HI: chứa phần dư, LO: chứa thương
- Khuôn dạng của 3 loại lệnh I, J, R:



Assignment 1: lệnh gán số 16-bit

- Màn hình chạy:



- Sự thay đổi của $s0 và pc:

$s0	0x00000000	0x00003007	0x00000000
pc	0x00400000	0x00400004	0x00400008

- Giải thích:
  - Chưa thực hiện lệnh nào:
    - $s0 = 0x00000000,
    - pc = 0x00400000

- Thực hiện lệnh 1, $s0 = 0x00000000 + 0x3007 = 0x00003007, mỗi lệnh pc tăng 4 byte
- Thực hiện lệnh 2, $s0 = 0 + 0 =0x00000000, pc tiếp tục tăng 4 byte.

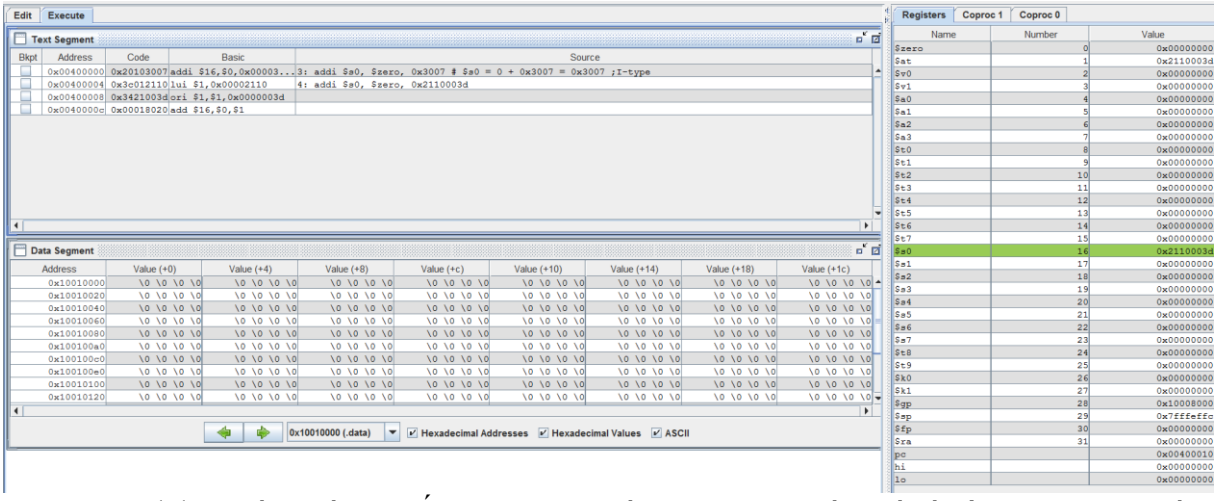
- So sánh ở Text Segment:
  - Mã máy: addi: 0x20103007
    => sang nhị phân (001000 00000 10000 00110 00000000111)
  - op = 001000 = 8

  => op = 8
  => đúng với khuôn dạng lệnh I

  - Mã máy: add :0x00108020
    - ⇨ Sang nhị phân (000000 00000 10000 10000 00000 100000)
  - op = 0; func = 32

  ⇨ op/func = 0/32
  ⇨ đúng với khuôn dạng lệnh R

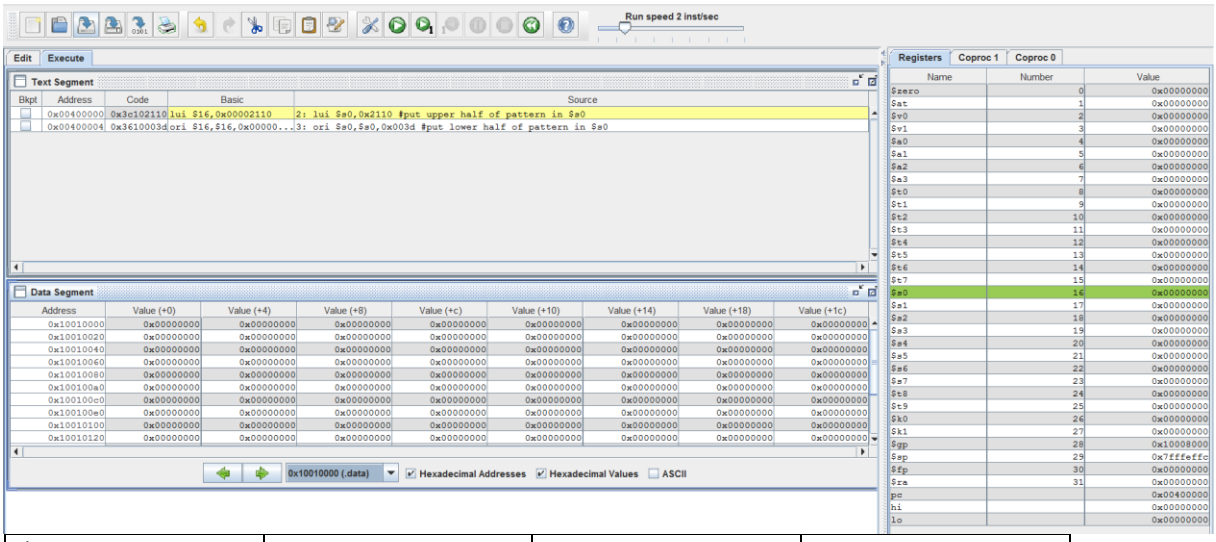
- Sau khi sửa:



0x2110003d được tách thành 2 số 0x00002110 và 0x0000003d lý do là vì công cụ mars ko làm việc với các số 32 bit mà nó tách thành 2 số 16 bit

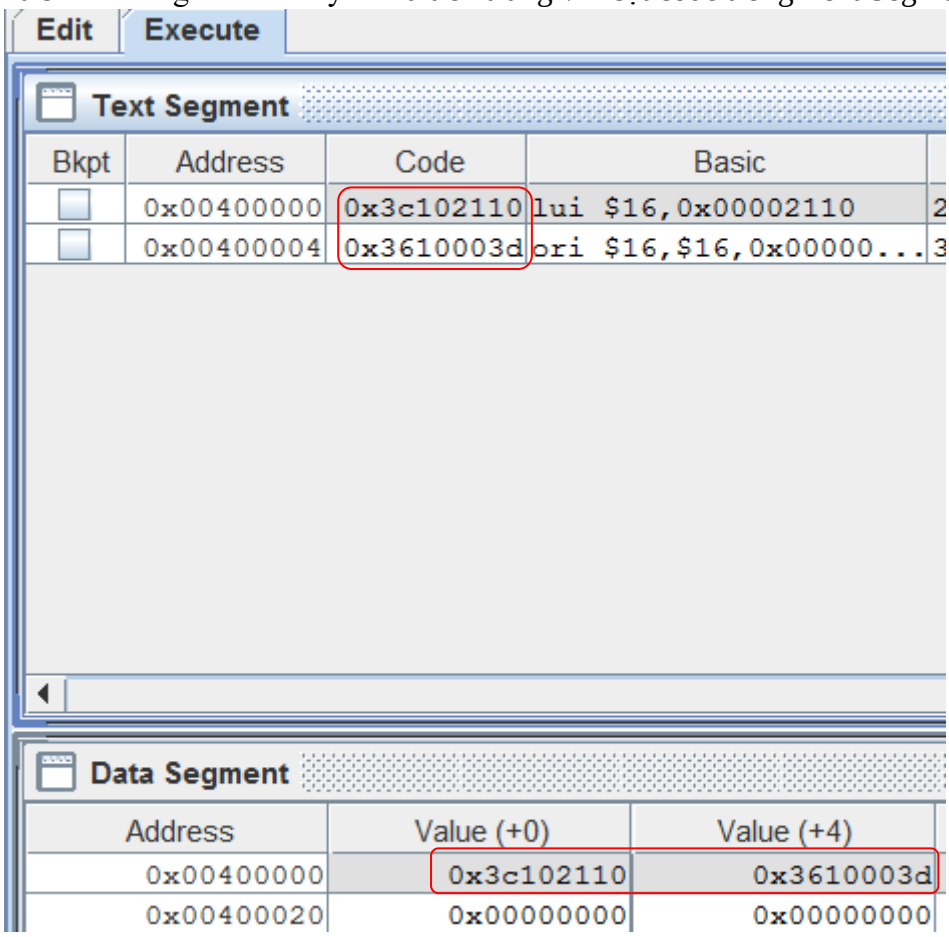
Assignment 2: lệnh gán số 32-bit

- Màn hình chạy:



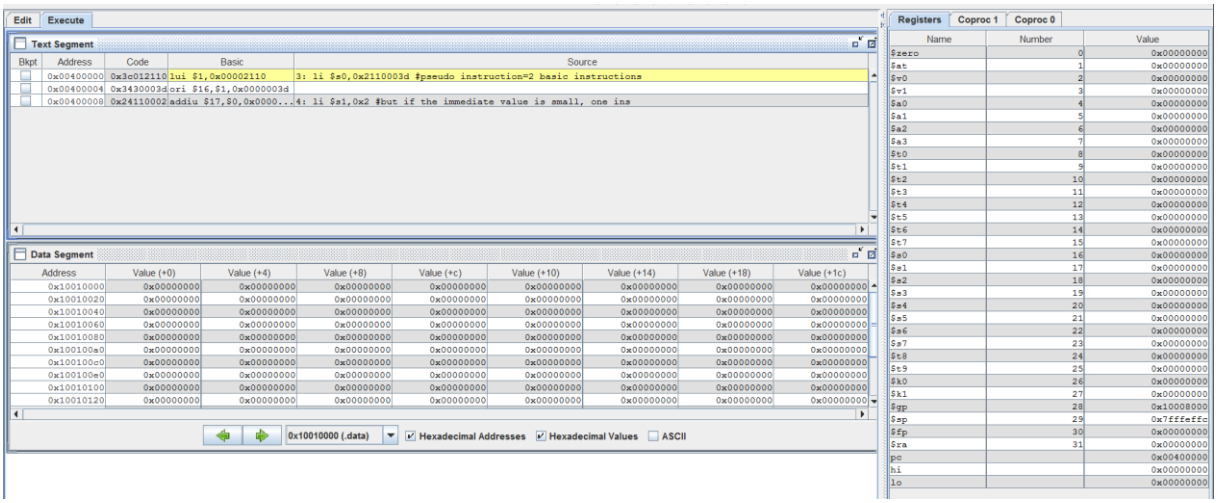
$s0	0x00000000	0x21100000	0x2110003d
pc	0x00400000	0x00400004	0x00400008

- Ở cửa sổ Data segment các byte đầu tiên trùng với cột code trong Text Segment

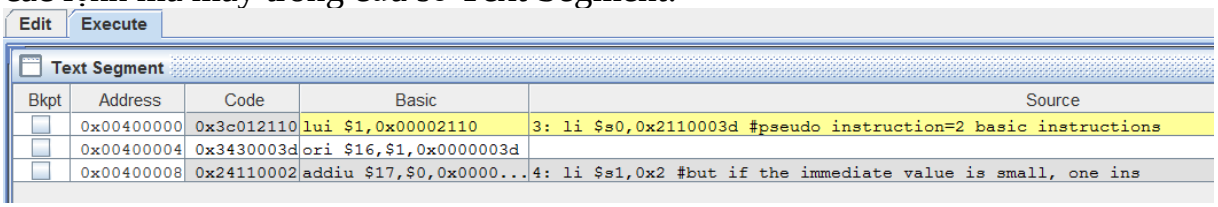


Assignment 3: lệnh gán (giả lệnh)

- Màn hình chạy:



- các lệnh mã máy trong cửa sổ Text Segment:

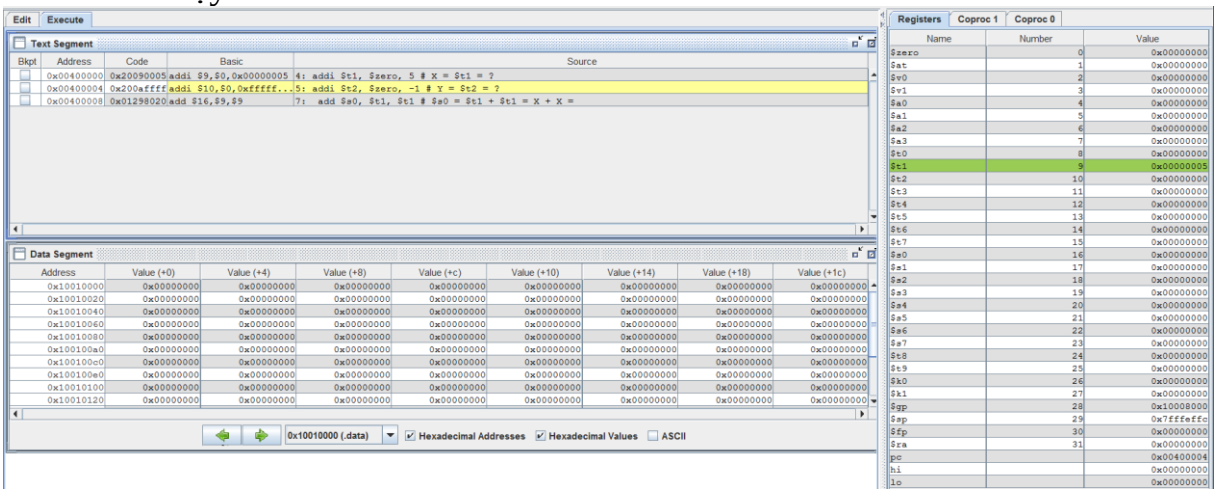


Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c012110	lui $1,0x00002110	3: li $s0,0x2110003d #pseudo instruction=2 basic instructions
	0x00400004	0x3430003d	ori $16,$1,0x0000003d	
	0x00400008	0x24110002	addiu $17,$0,0x0000...	4: li $s1,0x2 #but if the immediate value is small, one ins

- Lệnh li là giả lệnh
- Giá trị 0x2110003d là giá trị 32 bit
  - ⇨ phải tách thành 2 phần 16 bit dùng hàm lui và ori để lấy địa chỉ
- Lui $1, 0x00002110
  - ⇨ gán địa chỉ 0x00002110 vào $1
- Ori $16, $1, 0x0000003d
  - ⇨ so sánh và gán nửa dưới vào $16
- Lệnh addiu $17, $, 0x00000002
  - ⇨ gán giá trị vào $17

Assignment 4: tính biểu thức 2x + y = ?

- Màn hình chạy:



- Sự thay đổi của các thanh ghi:

- 1: $t1 0x00000005
- 2: $t2 0xffffffff
- 3: $s0 0x0000000a
- 4: $s0 0x00000009

⇨ Kết quả chính xác

- Điểm tương đồng giữa hợp ngữ và mã máy:
-

Text Segment

Bkpt	Address	Code	Basic
	0x00400000	0x20090005	addi $9,$0,0x00000005
	0x00400004	0x200affff	addi $10,$0,0xffff...
	0x00400008	0x01298020	add $16,$9,$9

⇨ 5 bit cuối của mã máy chính là đầu vào của hợp ngữ

Lệnh addi : 0x2009005 => 00100000000010010000000000101

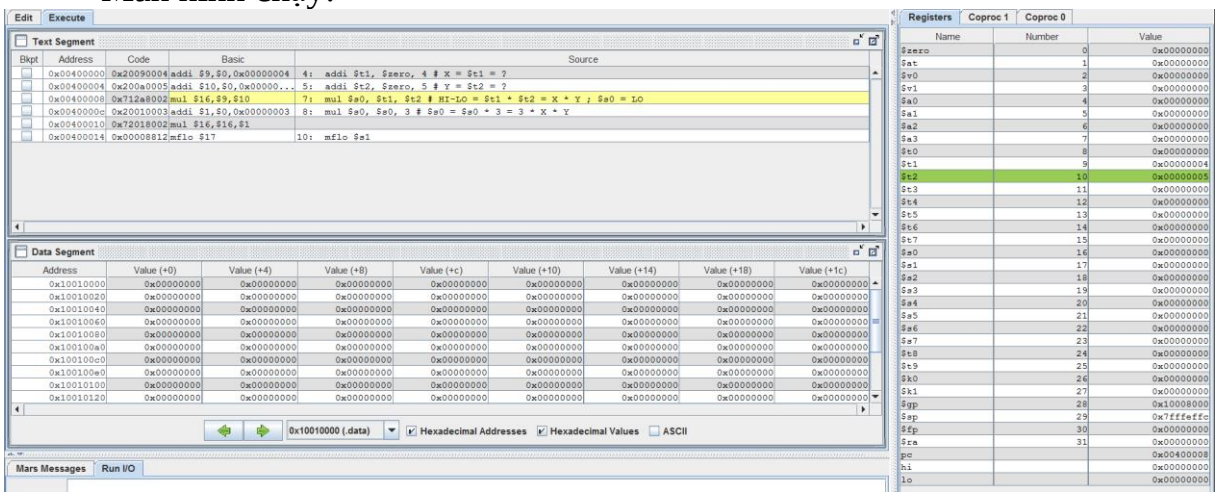
⇨ op = 8 => phù hợp với khuôn mẫu của kiểu lệnh I

Lệnh add: 0x01298020 => 00000001001010011000000000100000

⇨ op =0, func = 32 => phù hợp với khuôn mẫu của kiểu lệnh R

Assignment 5: phép nhân

- Màn hình chạy:



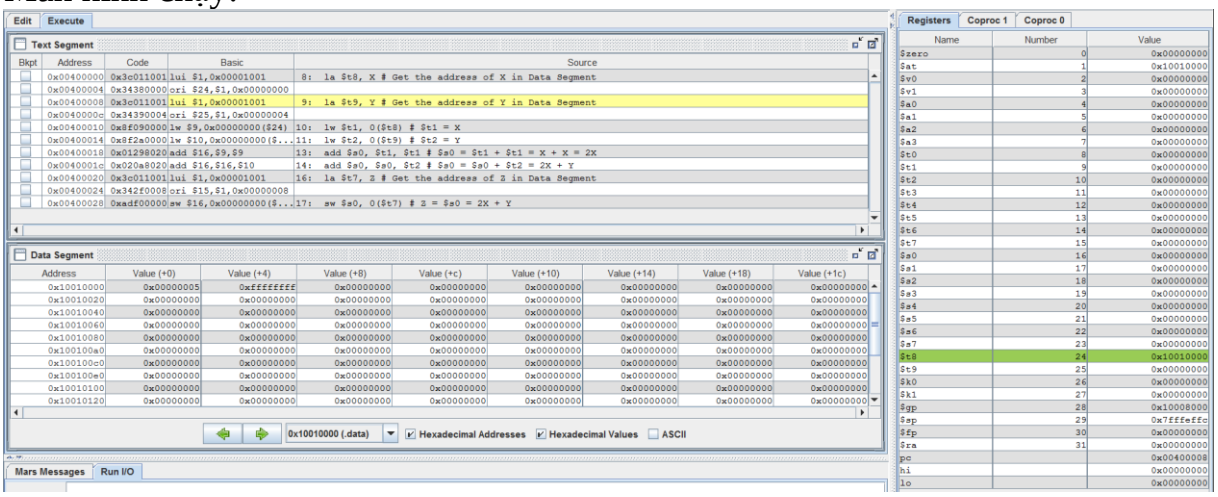
- Điều bất thường và giải thích:
  - lệnh mul đầu tiên là lệnh mul basic
  - ⇨ ko đổi
  - lệnh mul thứ 2 không phải là lệnh basic
  - ⇨ biến đổi thành 2 lệnh
  - lệnh mul tiếp theo thực hiện phép mul trên 3 thanh ghi
- Sự biến đổi khi debug từng lệnh khi debug
  - 1: $t1 = 0x00000004
  - 2: $t2 = 0x00000005
  - 3: lo = 0x00000014
  - 4: $at = 0x00000003
  - 5: lo = 0x0000003c
  - 6: $s1 = 0x0000003c

⇨ **kết quả chính xác**

- Thanh ghi HI không có sự biến đổi gì

Assignment 6: tạo biến và truy cập biến

- Màn hình chạy:



- Lệnh “la”:
  - lệnh gán địa chỉ.
  - được biên dịch bằng 2 lệnh “lui” và “ori”
- Ở cửa sổ Label và quan sát địa chỉ của X, Y, Z:
  - 16 bit cuối của địa chỉ X, Y, Z giống với 16 bit cuối sau khi biên dịch lệnh “La” thành mã máy
- Debug:
  - 1: $at = 0x10010000
  - 2: $t8 = 0x10010000
  - 3: $at = 0x10010000
  - 4: $t9 = 0x10010004
  - 5: $t1 = 0x00000005
  - 6: $t2 = 0xffffffff
  - 7: $s0 = 0x0000000a
  - 8: $s0 = 0x00000009
  - 9: $at = 0x10010000
  - 10 $t7 = 0x10010008
- Tìm hiểu các lệnh:
  - lw: Lấy giá trị từ bộ nhớ đưa vào thanh ghi.
  - sw: Lưu giá trị vào bộ nhớ từ thanh ghi.
  - lb: Nạp 1 byte từ bộ nhớ vào thanh ghi.
  - sb: Cất 1 byte bên phải của thanh ghi ra bộ nhớ.