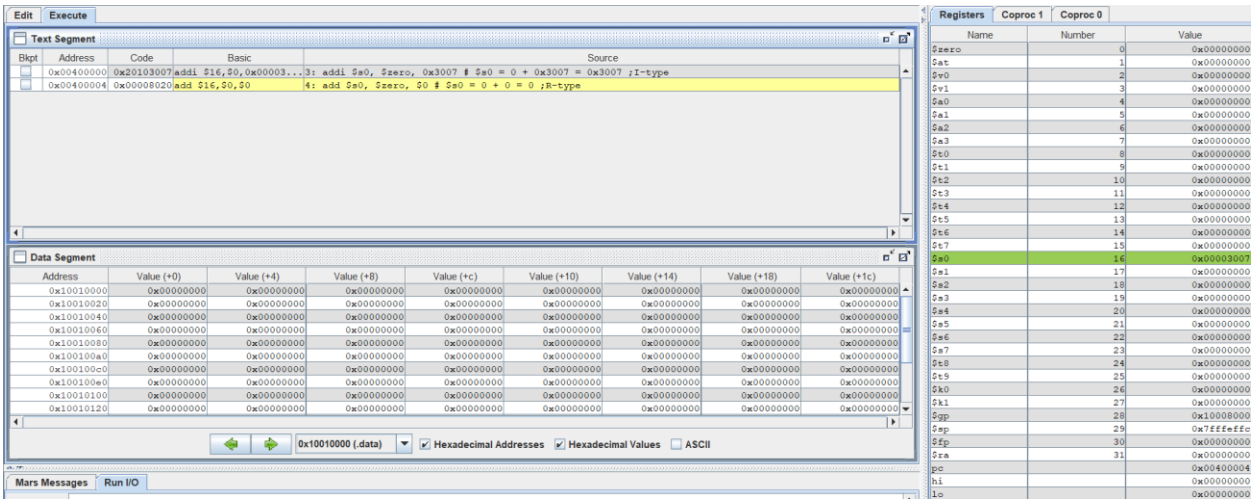
Họ và Tên: Hà Trung Chiến

MSSV: 20225794

Laboratory Exercise 2

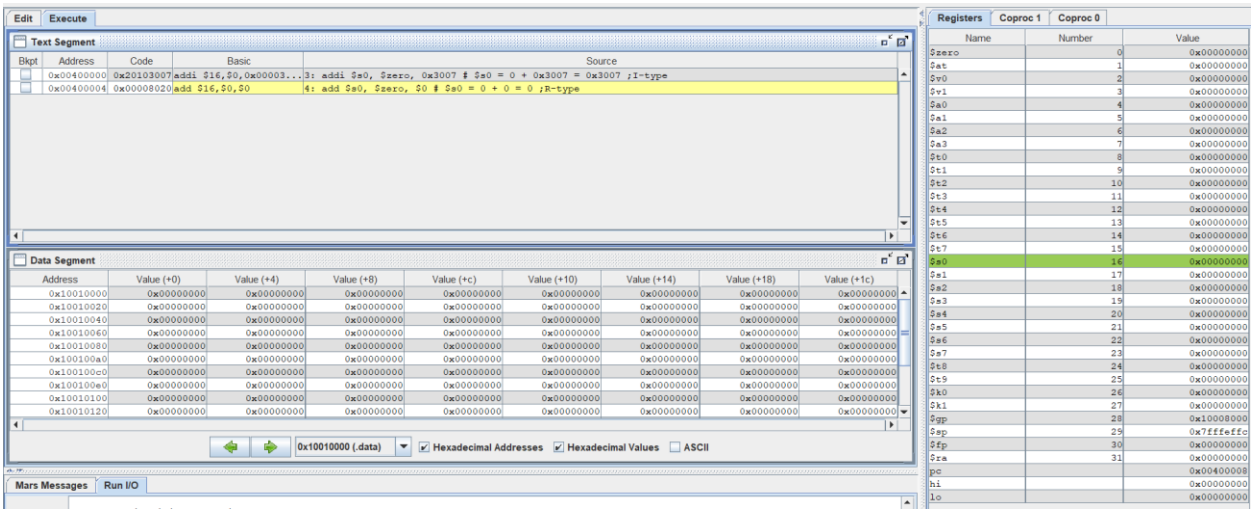
Assignment 1: lệnh gán số 16 – bit

- Sau câu lệnh đầu tiên:



- Giá trị thanh ghi $s0 thay đổi thành 0x00003007. Do câu lệnh “addi $s0, $zero, 0x3007” dùng để thay đổi địa chỉ của thanh ghi đích là $s0 thành 0x00003007.
- Giá trị thanh ghi pc thay đổi thanh 0x00400004 do pc lưu địa chỉ của dòng lệnh tiếp theo.

- Sau câu lệnh tiếp theo:



- Giá trị của thanh ghi $s0 thay đổi trở về 0x00000000. Do câu lệnh “add $s0, $zero, $0” dung để thay đổi địa chỉ của thanh ghi đích là $s0 thành 0x00000000.
- Giá trị thanh ghi pc thay đổi thành 0x00400008 do pc lưu địa chỉ của dòng lệnh tiếp theo.

- Kiểm tra mã máy của các lệnh so với khuôn dạng lệnh:

Lệnh addi $s0, $zero, 0x3007:

- I – format
- Op: 8 -> 001000
- rt: $s0 -> $16 -> 10000
- rs: $zero -> $0 -> 00000
- imm: 0x3007 -> 0011 0000 0000 0111

→ lệnh máy: 0010 0010 0000 0000 0011 0000 0000 0111 -> 0x20103007.

→ Vậy mã máy này đúng với khuôn dạng lệnh

Lệnh add $0, $zero, $0:

- R – format
- Op: 0 -> 000000
- rt: 0 -> 00000
- rd: 16 -> 10000
- sh: 0 -> 00000
- fn: 32 -> 100000

→ lệnh máy: 0000 0000 0001 0000 000 0010 0000 -> 0x00008020.

→ Vậy mã máy này đúng với khuôn dạng lệnh.

- Thay lệnh addi ban đầu thành "addi $s0, $zero, 0x2110003d"

  Sau khi chạy câu lệnh sẽ tự động chuyển thành:

  lui $1, 0x00002110

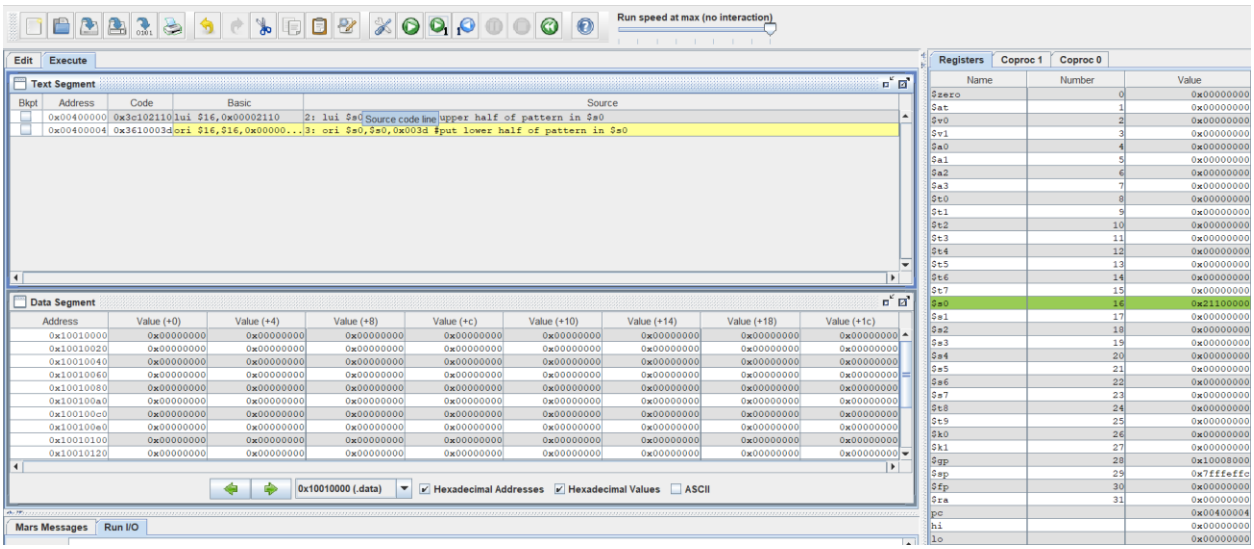
  ori $1, $1, 0x0000003d

  add $16, $0, $1

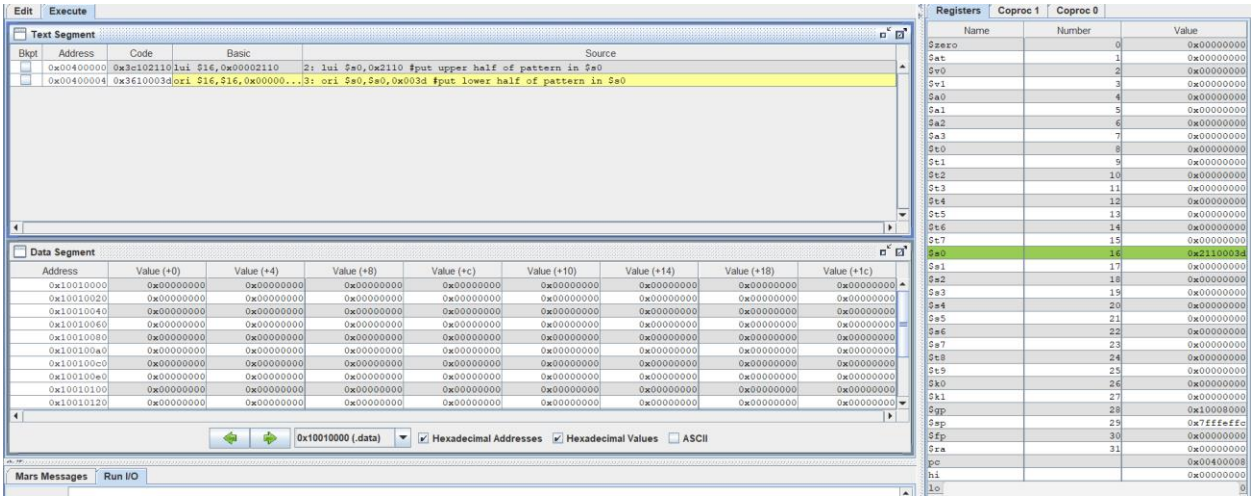
→ Dễ thấy các lệnh thao tác với hằng số ở trên đều có giới hạn 16 bit cho hằng số nhưng 0x2110003d là một số 32 – bit nên câu lệnh bị tách ra thành lui và ori.

Assignment 2: lệnh gán số 32 – bit

- Sau khi thực hiện câu lệnh đầu tiên:



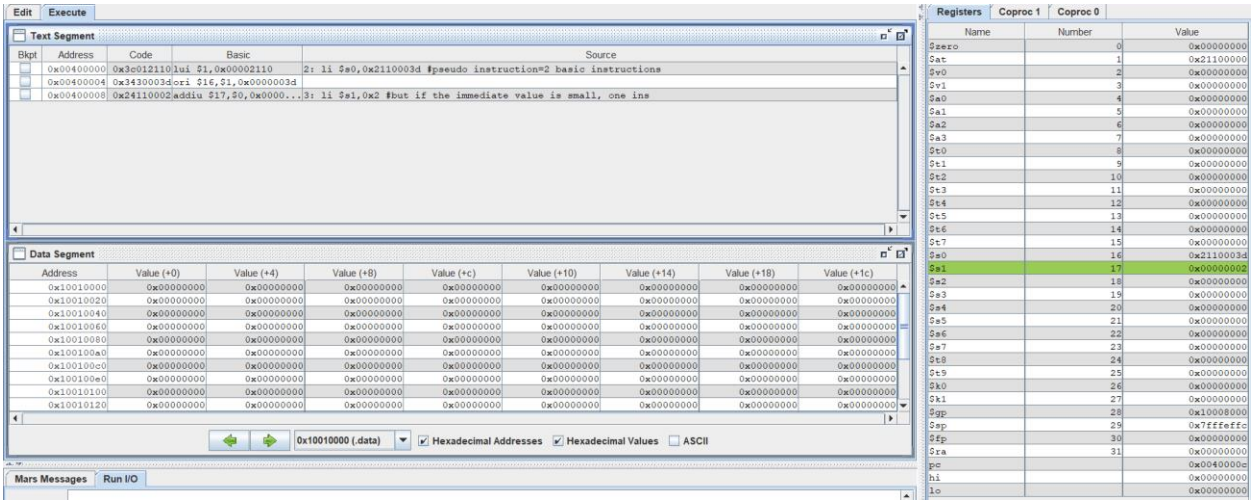
+ Giá trị của thanh ghi $s0 thay đổi thành 0x21100000. Do câu lệnh lui dịch chuyển giá trị ngay 16 bit sang trái và lưu vào thanh ghi đích.
+ Giá trị của pc thay đổi thành 0x00400004 do pc lưu địa chỉ dòng lệnh tiếp theo.
- Sau khi thực hiện câu lệnh thứ 2:



+ Giá trị của thanh ghi $s0 thay đổi thành 0x2110003d. Do lệnh ori được sử dụng để thực hiện phép OR giữa giá trị hiện tại của thanh ghi và giá trị 16 – bit của hằng số.
+ Giá trị của pc thay đổi thành 0x00400008 do pc lưu địa chỉ của dòng lệnh tiếp theo.

→ Ở cửa sổ Data Segment khi quan sát các byte đầu tiên ở vùng lệnh ta thấy nó trùng với cột Code trong Text Segment.

Assignment 3: lệnh gán (giả lệnh)



Khi dịch mã trên chương trình MARS sẽ dịch thành

lui $1, 0x00002110

ori $16, $1, 0x0000003d

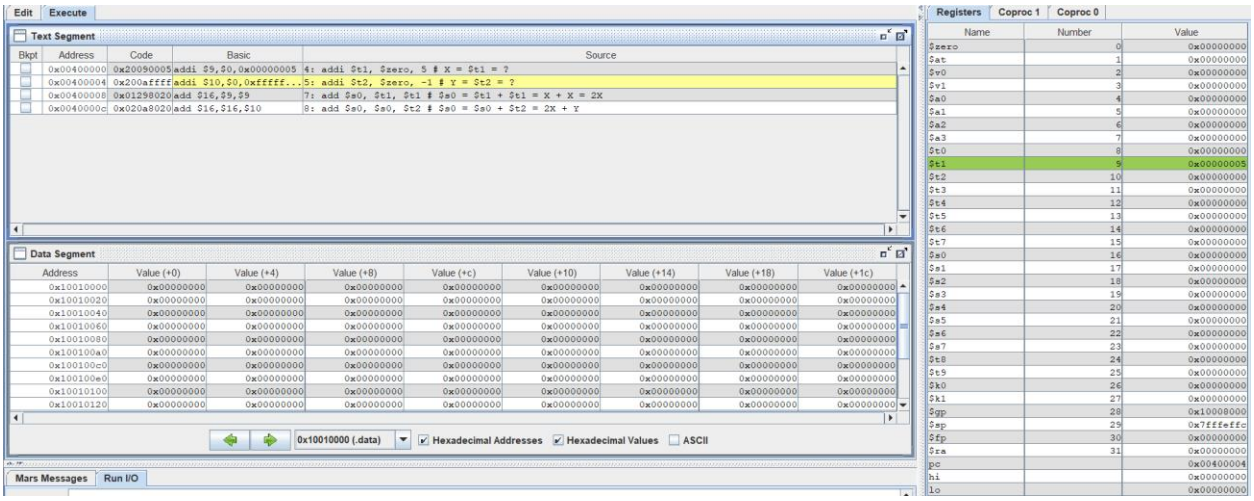
addiu $17, $0, 0x00000002

→ Sau khi dịch, ta thấy không có lệnh li nào được thực hiện vì thực tế li là một giả lệnh (pseudo instruction) nên bản chất khi sử dụng lệnh này nó sẽ tự động chuyển thành một tập các lệnh tương ứng để thực hiện chức năng gán:

→ Ta thấy dòng lệnh thứ nhất được tách thành 2 lệnh lui và ori do 0x2110003d là số 32-bit không thể lưu trực tiếp còn lệnh thứ 2 số được gán nhỏ nên có thể lưu trực tiếp.

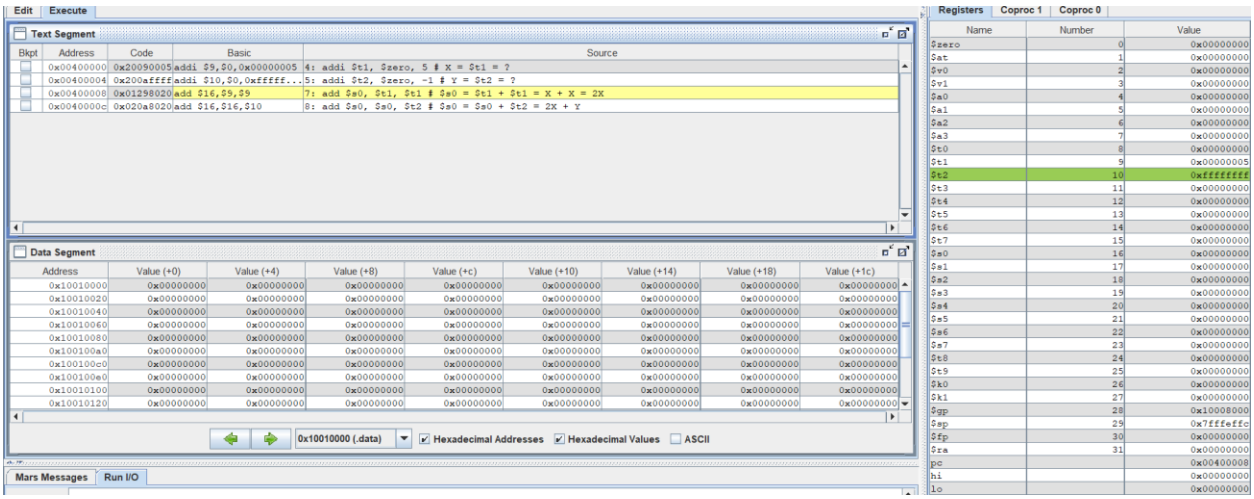
Assignment 4: tính biểu thức 2x + y = ?

- Sau khi thực hiện dòng lệnh đầu tiên:



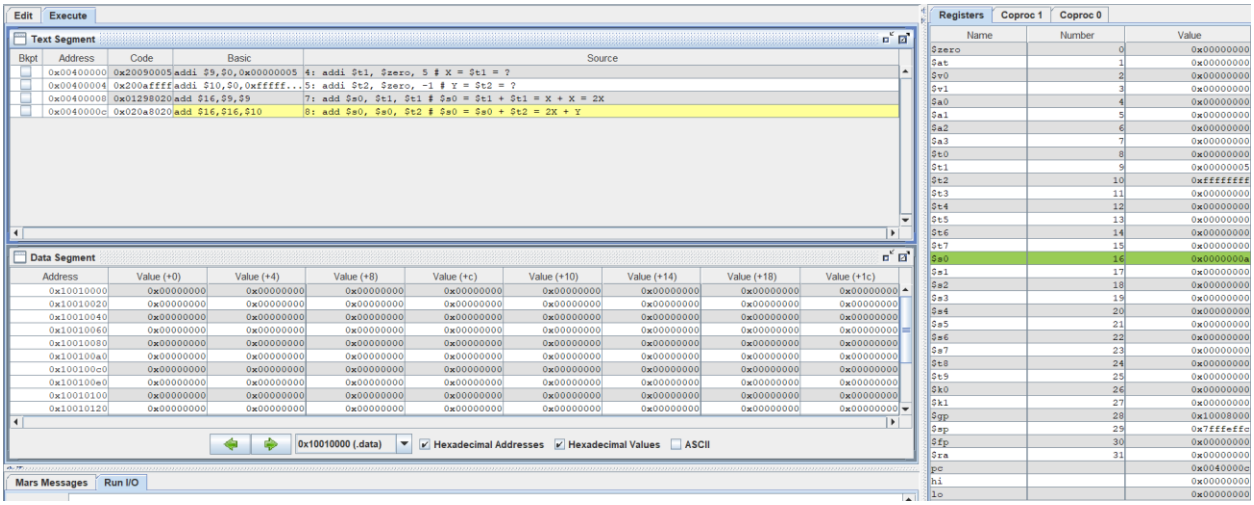
+ Giá trị của thanh ghi $t1 thay đổi thành 0x00000005. Do câu lệnh này sẽ lưu giá trị X = 5 vào thanh ghi $t1.

- Sau khi thực hiện dòng lệnh thứ 2:



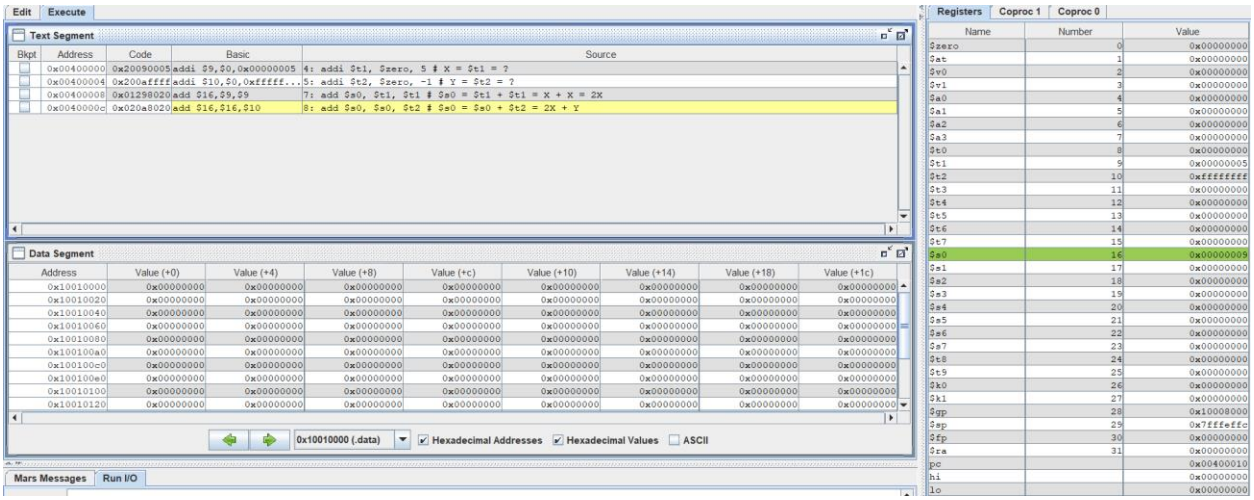
+ Giá trị của thanh ghi $t2 thay đổi thành 0xffffffff. Vì câu lệnh này sẽ lưu Y = -1 vào thanh ghi $t2, tuy nhiên do không thể biểu diễn số âm nên khi lưu trữ vào thanh ghi $t2 sẽ xảy ra hiện tượng tràn số khiến cho thanh $t2 có giá trị là 0xffffffff, tuy nhiên việc tràn số này không ảnh hưởng đến phép cộng.

- Sau khi thực hiện câu lệnh thứ 3:



+ Giá trị của thanh ghi $s0 thay đổi thành 0x0000000a. DO câu lệnh này sẽ thực hiện cộng $t1 với $t1 và lưu vào $s0 tương đương với phép tính 2X = X + X.

- Sau khi thực hiện câu lệnh thứ 4:



+ Giá trị của thanh ghi $s0 thay đổi thành 0x00000009. Vì câu lệnh này sẽ thực hiện cộng $s0 với $t2 và lưu vào $s0 tương đương với phép tính 2X + Y.

→ Sau khi kết thúc chương trình giá trị thanh ghi $s0 là 9 đúng với kết quả của phép tính 2*5-1 = 9.

- Quan sát câu lệnh addi:

- lệnh addi $t1, $zero, 5
  + opcode: 8 -> 001000
  + rt: $t1-> $9 -> 01001
  + rs: $zero -> $0 -> 00000
  + imm: 5 -> 0x00000005 -> 0000000000000101

-> Mã máy là: 0010 0000 0000 1001 0000 0000 0000 0101 -> 0x20090005.

->Vậy mã máy này đúng so với khuôn dạng lệnh.

- lệnh addi $t2, $zero, -1
  + opcode: 8 -> 001000
  + rt: $t2-> $10 -> 01010
  + rs: $zero -> $0 -> 00000
  + imm: -1 -> 0xffffffff -> 1111 1111 1111 1111

=> Mã máy là: 0010 0000 0000 1010 1111 1111 1111 1111 -> 0x200affff. Vậy mã máy này đúng so với khuôn dạng lệnh.

→ Vậy ta thấy điểm tương đồng giữa câu lệnh addi và hợp ngữ máy đều có cùng cấu trúc: op (addi) + rs ($t1) + rt ($zero) + Operand/Offset

- Xét lệnh add $s0, $t1, $t1, ta có

+ Op: 0 -> 000000
+ rs: $t1 -> 01001
+ rt: $t1 -> 01001

+ rd: $s0 -> 10000
+ sh: 00000
+ fn: 100000

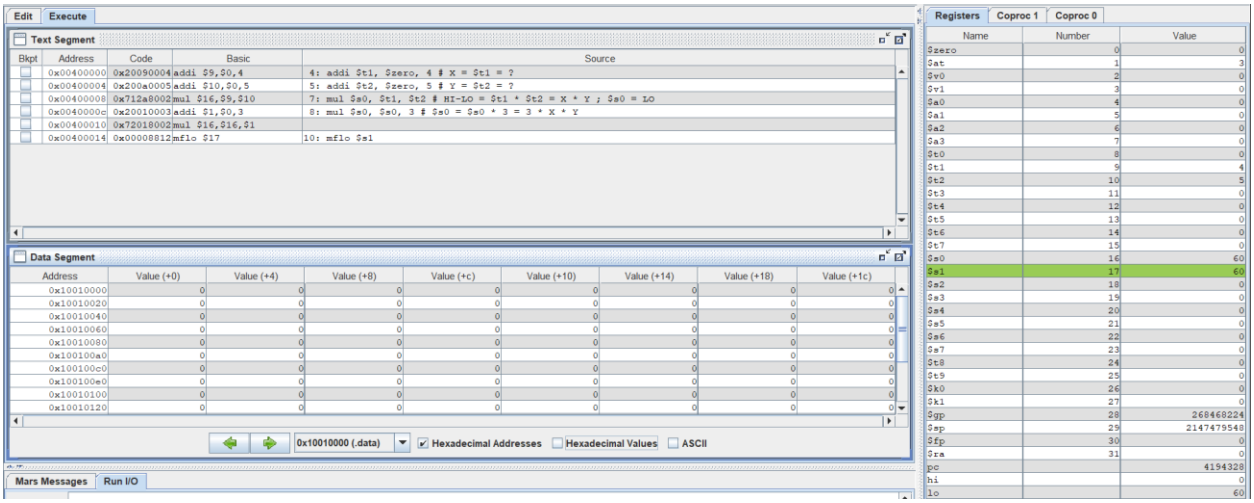
→ ở của số text segment mã code là 0x01298020 khi chuyển sang hệ 2 là 000000 01001 01001 10000 00000 100000 khớp với mã máy.

- Xét lệnh add $s0, $s0, $t2, ta có
+ Op: 0 -> 000000
+ rs: $s0 -> 10000
+ rt: $t2 -> 01010
+ rd: $s0 -> 10000
+ sh: 00000
+ fn: 100000

→ ở của số text segment mã code là 0x020a8020 khi chuyển sang hệ 2 là 000000 10000 01010 10000 00000 100000 khớp với mã máy.

➢ Vậy câu lệnh add khớp với khuôn mẫu của kiểu lệnh R.

Assignment 5: Phép nhân

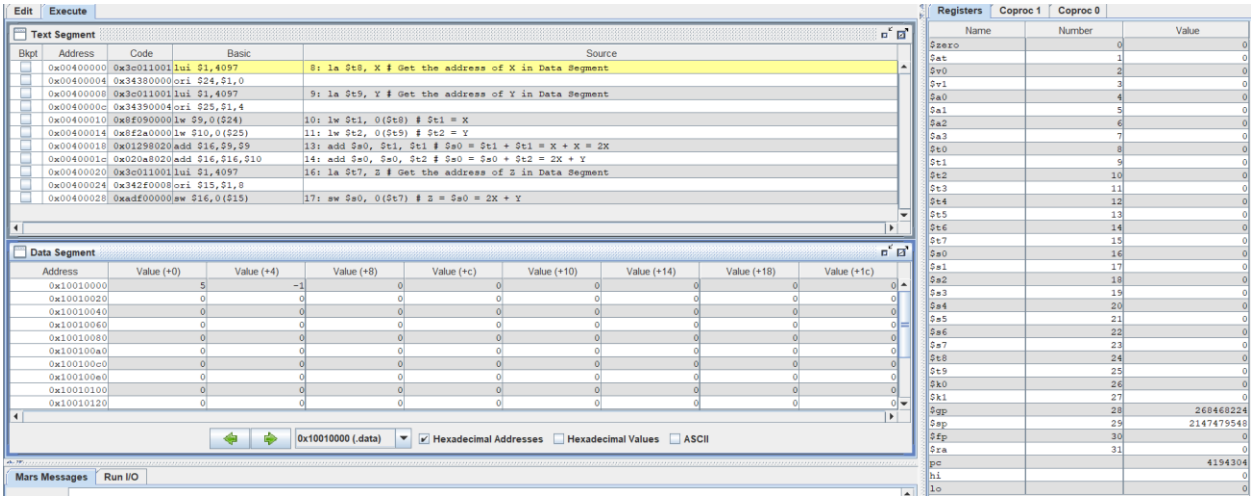


- Ta thấy điều bất thường là ở câu lệnh mul $s0, $s0, 3, khi nó được chuyển thành:
+ addi $1, $0, 0x00000003
+ mul $16, $16, $1

→ Ta thấy rằng lệnh mul chỉ có thể thực hiện được với giá trị thanh ghi nên khi truyền tham số trực tiếp nó sẽ lưu tham số đó vào thanh ghi $1, sử dụng lệnh add sau đó mới thực hiện lệnh mul với thanh ghi $1.

→ Kết quả sẽ được lưu lại vào thanh ghi $lo, câu lệnh cuối mflo $s1 giúp ta có thể lấy giá trị từ thanh ghi $lo và gán vào thanh ghi $1. Điều này giải thích tại sao cả thanh ghi $s0 và thanh ghi $s1 đều có giá trị là 60.

Assignment 6: tạo biến và truy cập biến



- Lệnh la được biên dịch thành lệnh lui và ori để có thể gán 1 số 32 bit vào thanh ghi.
- Địa chỉ của các biến:
+ X: 0x10010000
+ Y: 0x10010004
+ Z: 0x10010008
- Chương trình trên dùng để giải phương trình Z = 2X + Y.
- Lệnh lw có vai trò lưu địa chỉ của biến vào thanh ghi.
- Lệnh sw có vai trò gán giá trị của một thanh ghi vào một địa chỉ.