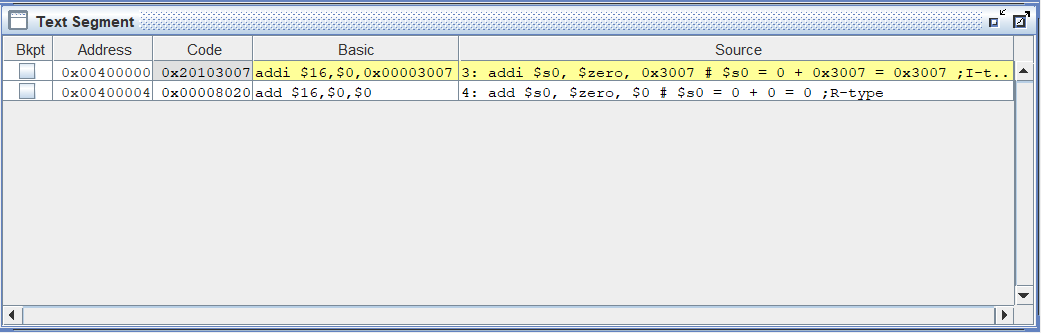
**Assignment 1: lệnh gán số 16-bit**

*Giá trị của thanh ghi $s0*

0x00000000 => 0x00003007 => 0x00000000

*Giá trị của thanh ghi $pc*

0x00400000 => 0x00400004 => 0x00400008

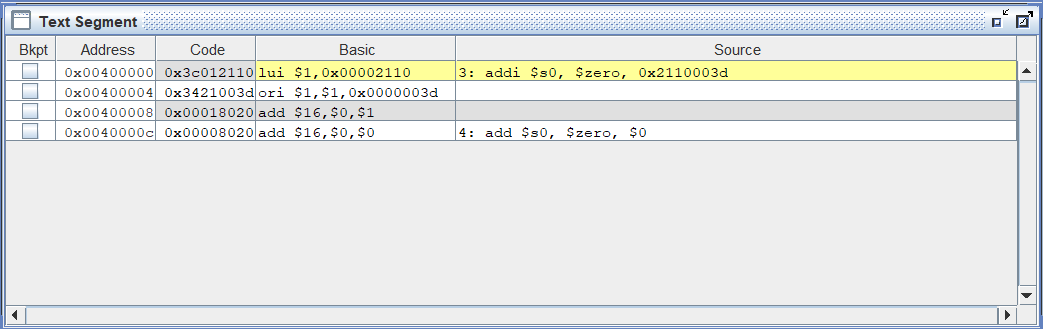


*Sửa lại:*

.text

addi $s0, $zero, 0x2110003d

add $s0, $zero, $0



Nhận xét:

- Lệnh addi chỉ dùng để cộng hằng số 16-bit

- 0x2110003d là số 32-bit

- MIPS sử dụng giả lệnh addi để cộng hằng số 32-bit: lệnh lui (load upper immediate) với chức năng ghi một hằng số 16-bit vào 2 byte cao của thanh ghi, 2 byte thấp sẽ được gán bằng 0

**Assignment 2: lệnh gán số 32-bit**

*Code:*

#Laboratory Exercise 2, Assignment 2

.text

lui $s0, 0x2110

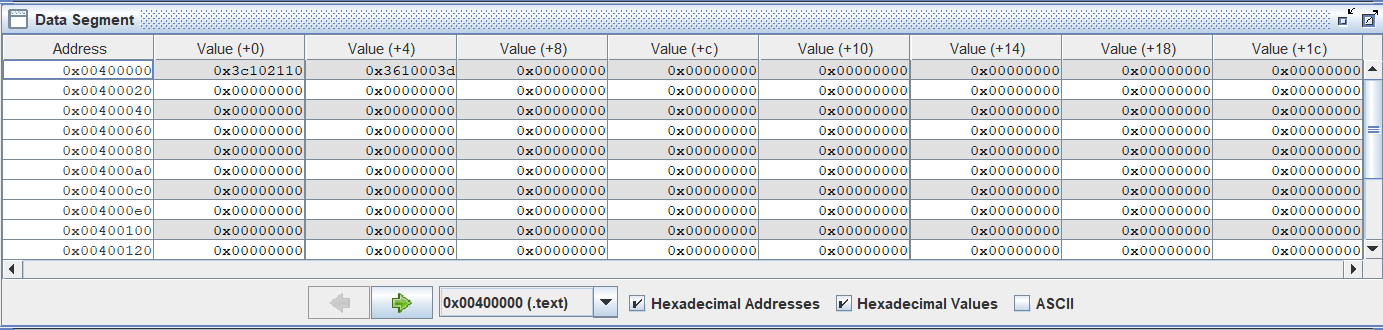
ori $s0, $s0, 0x003d

*Giá trị của thanh ghi $s0*

0x00000000 => 0x21100000 => 0x2110003d

*Giá trị của thanh ghi $pc*

0x00400000 => 0x00400004 => 0x00400008



- Địa chỉ: 0x00400000 và 0x00400004

**Assignment 3: lệnh gán (giả lệnh)**

*Code:*

#Laboratory Exercise 2, Assignment 3

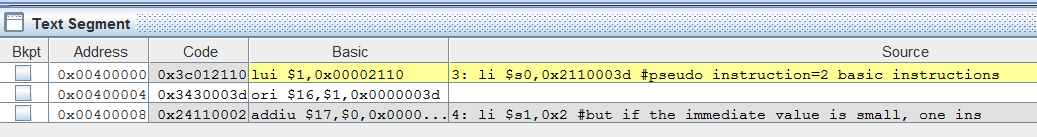
.text

li $s0, 0x2110003d

li $s1, 0x2

- Lệnh mã máy của 2 lệnh là khác nhau (do giá trị truyền vào khác nhau 32/16 bit)

- Lệnh thứ nhất là giả lệnh (dùng để cộng 32-bit): sử dụng lui và ori



+ Lệnh lui $1, 0x00002110 gán địa chỉ 0x00002110 vào $1

+ Lệnh ori $16, $1, 0x0000003d tiến hành so sánh và gán nửa dưới vào $16

- Lệnh thứ hai là lệnh máy (dùng để cộng 16-bit)

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

*Code*

#Laboratory Exercise 2, Assignment 4

.text

# Assign X, Y

addi $t1, $zero, 5 # X = $t1 = 5

addi $t2, $zero, -1 # Y = $t2 = -1

# Expression Z = 2X + Y

add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 10

add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y = 9

*Giá trị thanh ghi*

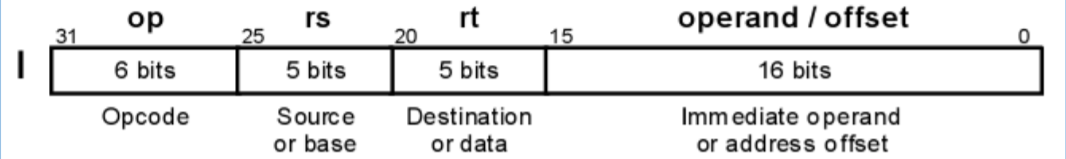
$pc: 0x00400000 => 0x00400004 => 0x00400008 => 0x0040000c => 0x00400010

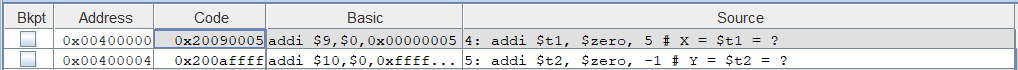
$t1: 0x00000000 => 0x00000005

$t2: 0x00000000 => 0xffffffff

$s0: 0x00000000 => 0x0000000a => 0x00000009

- Ứng với một lệnh hợp ngữ ta có 1 lệnh mã máy





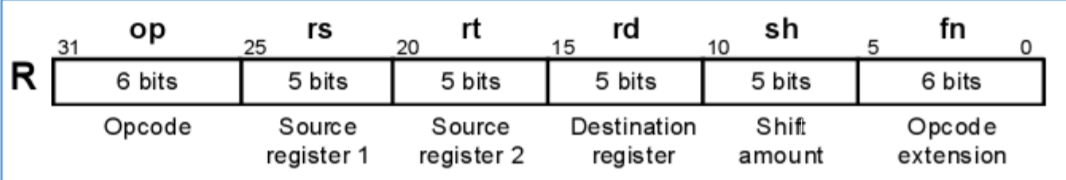
addi $rt, $rs, imm

Code: 100000000010010000000000000101 (0x20090005)

Immediate: 0000000000000101 (0x000000005)

$rt: 01001 (9)

$rs: 00000 (0)





add $rd, $rs, $rt

Code: 00000001001010011000000000100000 (0x01298020)

$rd: 10000 (16)

$rt: 01001 (9)

$rs: 01001 (9)

**Assignment 5: phép nhân**

*Code*

#Laboratory Exercise 2, Assignment 5

.text

# Assign X, Y

addi $t1, $zero, 4 # X = $t1 = 4

addi $t2, $zero, 5 # Y = $t2 = 5

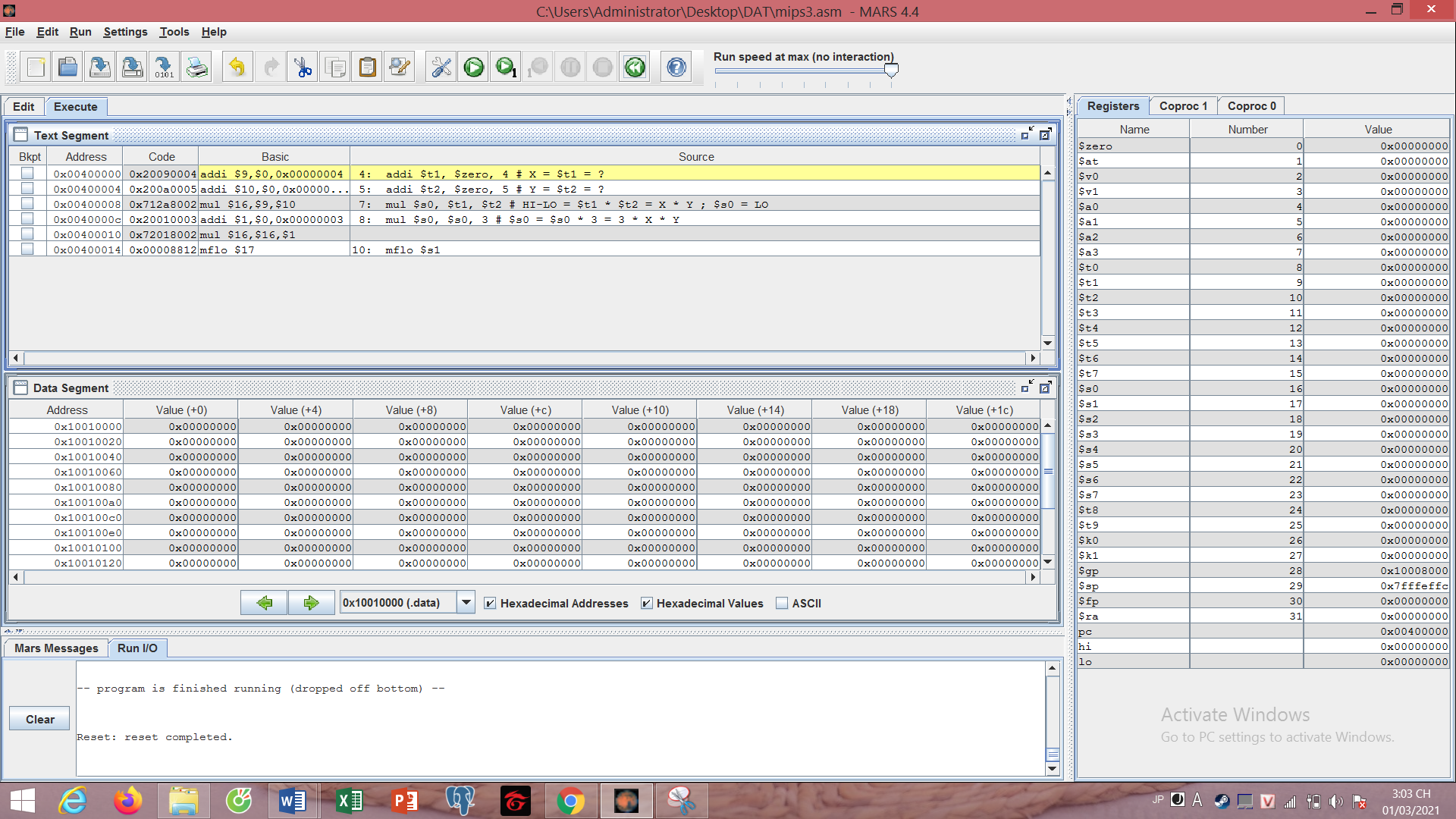
# Expression Z = 3\*XY

mul $s0, $t1, $t2 # HI-LO = $t1 \* $t2 = X \* Y ; $s0 = LO

mul $s0, $s0, 3 # $s0 = $s0 \* 3 = 3 \* X \* Y

# Z' = Z

mflo $s1



- Sau khi biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment, thấy:

+ Lệnh mul đầu tiên không đổi vì nó là lệnh mul basic

+ Lệnh mul thứ 2 không phải là lệnh basic nên nó biến đổi thành 2 lệnh

+ Lệnh mul tiếp theo thực hiện phép mul trên 3 thanh ghi

- Giá trị các thanh ghi:

+ $t1: 0x00000004 // gán giá trị 4 vào $t1

+ $t2: 0x00000005 // gán giá trị 5 vào $t2

+ lo: 0x00000000 => 0x00000014 => 0x0000003c

+ $at: 0x00000003

+ $s1: 0x0000003c

+ $pc: 0x00400000 => 0x00400004 => 0x00400008 => 0x0040000c => 0x00400010 => 0x00400014 => 0x00400018

+ **hi** không thay đổi giá trị.

* Kết quả đúng

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

*Code*

#Laboratory Exercise 2, Assignment 6

.data # DECLARE VARIABLES

X : .word 5 # Variable X, word type, init value = 0x00000005

Y : .word -1 # Variable Y, word type, init value = 0xffffffff

Z : .word # Variable Z, word type, no init value

.text # DECLARE INSTRUCTIONS

# Load X, Y to registers

la $t8, X # Get the address of X in Data Segment

la $t9, Y # Get the address of Y in Data Segment

lw $t1, 0($t8) # $t1 = X

lw $t2, 0($t9) # $t2 = Y

# Calcuate the expression Z = 2X + Y with registers only

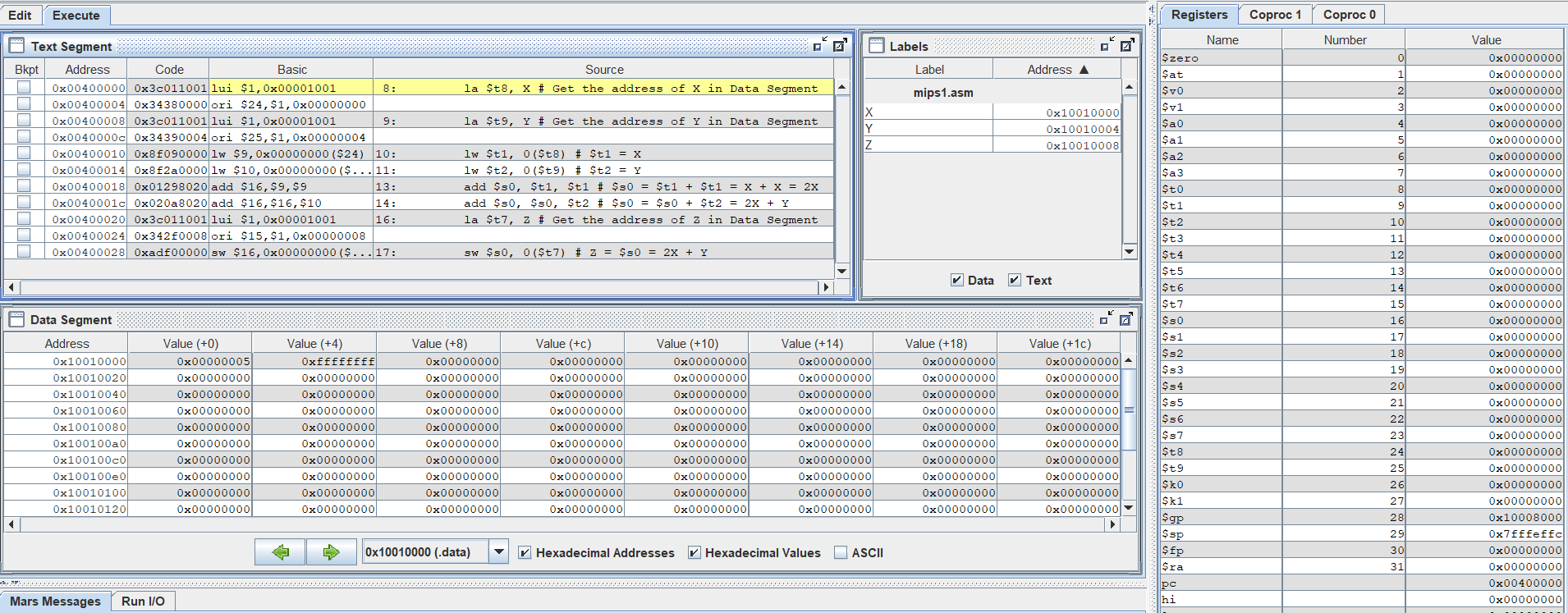
add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X

add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y

# Store result from register to variable Z

la $t7, Z # Get the address of Z in Data Segment

sw $s0, 0($t7) # Z = $s0 = 2X + Y



- Lệnh **la** (load address): Lấy địa chỉ biến trong Data Segment và lưu vào thanh ghi. Gồm 2 lệnh: lui và ori

- 16 bit cuối của địa chỉ X, Y, Z giống với 16 bit cuối sau khi biên dịch lệnh “La” thành mã máy

- Tên lệnh r1, offset(r2)

+ lw (load word): Đọc 4 byte.

+ lb (load byte): Đọc 1 byte. Vì thanh ghi có độ dài 4 byte, nhiều hơn lượng dữ liệu đọc được nên các bit trống sẽ được gán bằng bit dấu của số đọc được.

+ sw (store word): lưu 4 byte dữ liệu trong thanh ghi vào bộ nhớ.

+ sb (store byte): lưu 1 byte dữ liệu trong thanh ghi vào bộ nhớ.

- Sự biến đổi giá trị của các thanh ghi:

1. $at: 0x10010000

$pc: 0x00400004

2. $t8: 0x10010000

$pc: 0x00400008

3. $at: 0x10010000

$pc: 0x0040000c

4. $t9: 0x10010004

$pc: 0x00400010

5. $t1: 0x10000005

$pc: 0x00400014

6. $t2: 0xffffffff

$pc: 0x00400018

7. $s0: 0x0000000a

$pc: 0x0040001c

8. $s0: 0x10010009

$pc: 0x00400020

9. $at: 0x10010000

$pc: 0x00400024

10. $t7: 0x10010008

$pc: 0x00400028