**Lab 4 – Assigments**

**Họ và tên: Phạm Long Khánh**

**MSSV: 20225640**

**Asignment 1:**

# condition : $s1 xor $s2 < 0

# if (condition is false) -> No Overflow

# else if ($s1 < 0) (if ($s3 < $s1) -> No Overflow else -> Overflow)

# else (if ($3 > $s1) -> No Overflow else -> Overflow) (L1)

.text

start:

addi $s1, $0, 0x7fffffff # Load 0x7fffffff to $s1

addi $s2, $s0, 0x1 # Load 0x1 to $s1

addi $t0, $0, 0 # Set default status is No Overflow

addu $s3, $s1, $s2 # s3 = s1 + s2

xor $t1, $s1, $s2 # Test if $s1 and $s2 have the same sign

bltz $t1, No\_Overflow # if not, No Overflow

slt $t2, $s3, $s1 # set t2 = 1 if (t3 < s1) else t2 = 0

bltz $s1, L1 # branch to L1 if ($s1 < 0)

beq $t2, 0, No\_Overflow # when $s1 and $s2 are not negative

# if (s3 >= s1) No Overflow

# else Overflow

j Overflow

L1: # if ($s3 < $s1) -> No Overflow ($s1, $s2 are negative)

# else -> Overflow

bne $t2, $0, No\_Overflow # if (s3 < s1) No Overflow

# else OverFlow

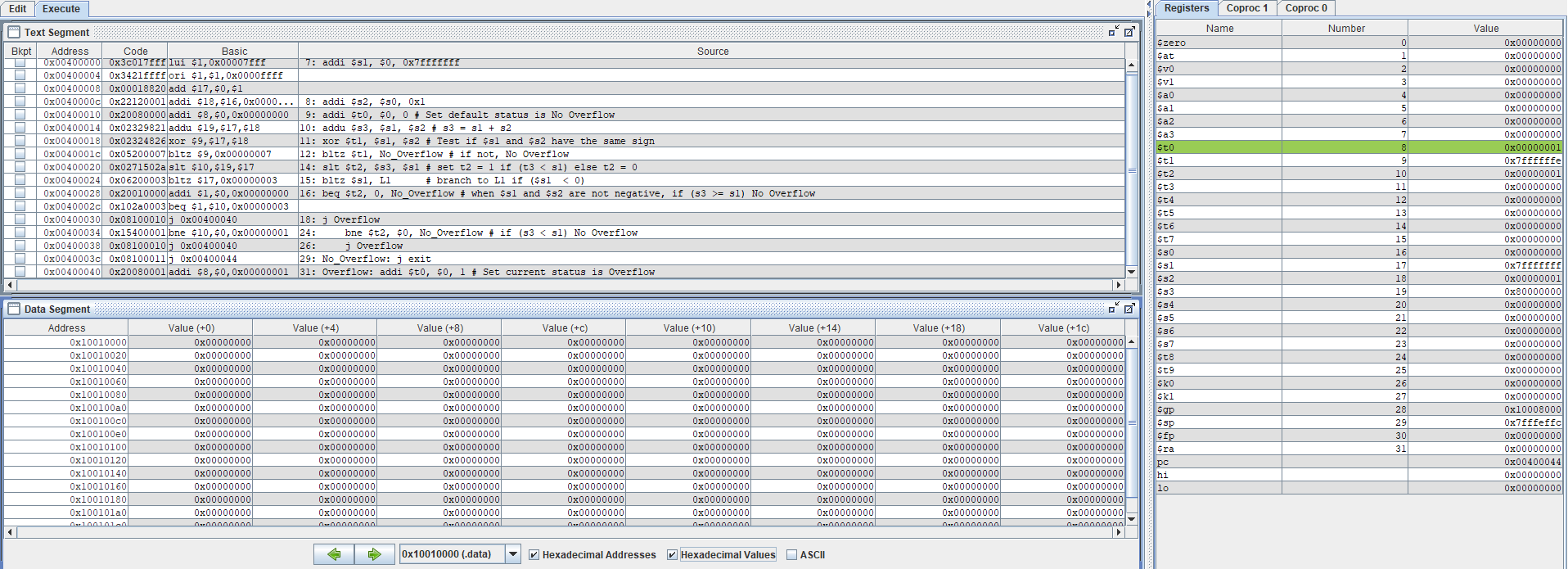
j Overflow

No\_Overflow: j exit

Overflow: addi $t0, $0, 1 # Set current status is Overflow

exit:

Trường hợp xảy ra tràn $s1 = 0x7fffffff > 0, $s2 = 1 > 0 => $s3 = 0x80000000 < 0



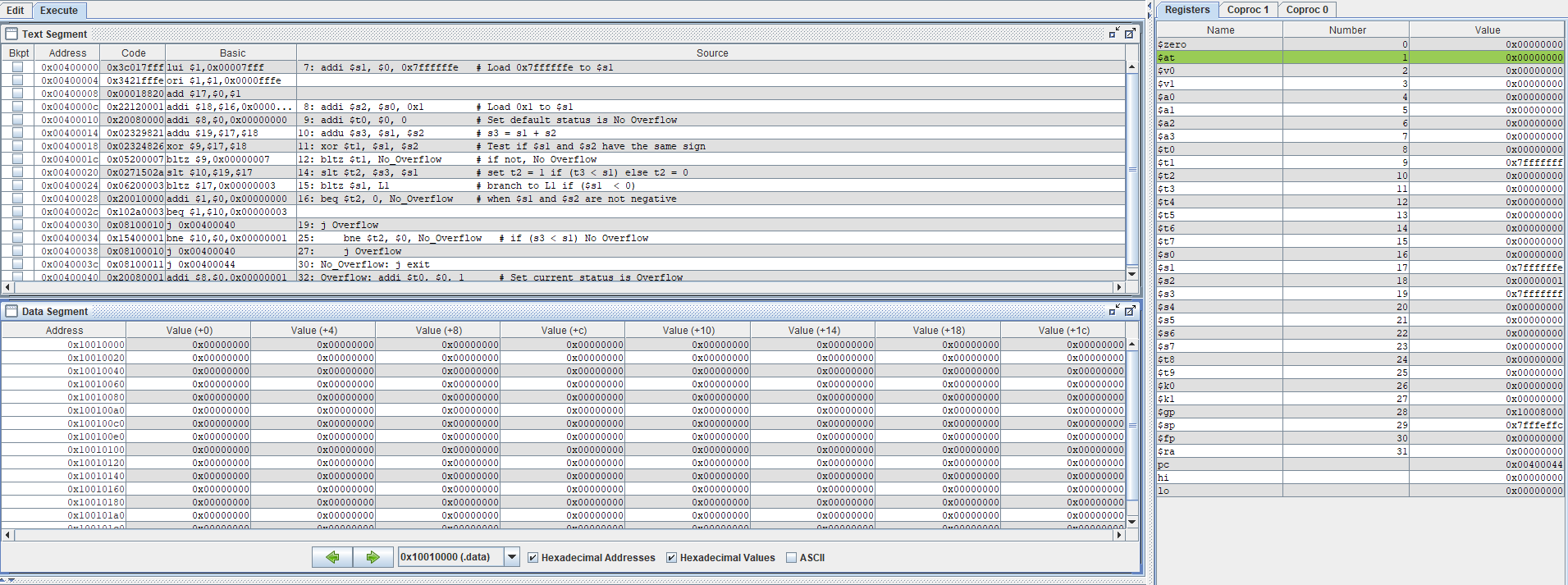
Chỉnh lại 2 dòng ở mã trên thành:

addi $s1, $0, 0x7ffffffe # Load 0x7ffffffe to $s1

addi $s2, $s0, 0x1 # Load 0x1 to $s1

Trường hợp không xảy ra tràn $s1 = 0x7ffffffe (2 ^ 31 – 1), $s2 = 0x1 (1)

=> $s3 = 0x7fffffff



**Asignment 2:**

# Extract MSB of $s0 to $s1 (1)

# Clear LSB of $s0 (make a copy and save to $s2) (2)

# Set LSB of $s0 (make a copy and save to $s3) (3)

# Clear $s0 (must use logical instructions) (4)

.text

start:

lui $s0, 0x1234 # load 0x12345678 to $s0

ori $s0, $s0, 0x5678

# (1)

lui $t0, 0xff00 # load 0xff000000 to $t0

ori $t0, $t0, 0x0

and $s1, $s0, $t0 # Extract MSB of $s0 and load it to $s1

# (2)

srl $s2, $s0, 8 # Shift $s0 right by 8 bits and load result to $s2

sll $s2, $s2, 8 # Shift $s2 left by 8 bits

# Clear LSB of $s0 and result load to $s2

# (3)

ori $s3, $s0, 0x00ff # Set LSB of $s0 and load result to $s3 (all bits of LSB = 1)

# (4)

and $s0, $s0, $0 # Clear $s0 (all bits are 0)

end:

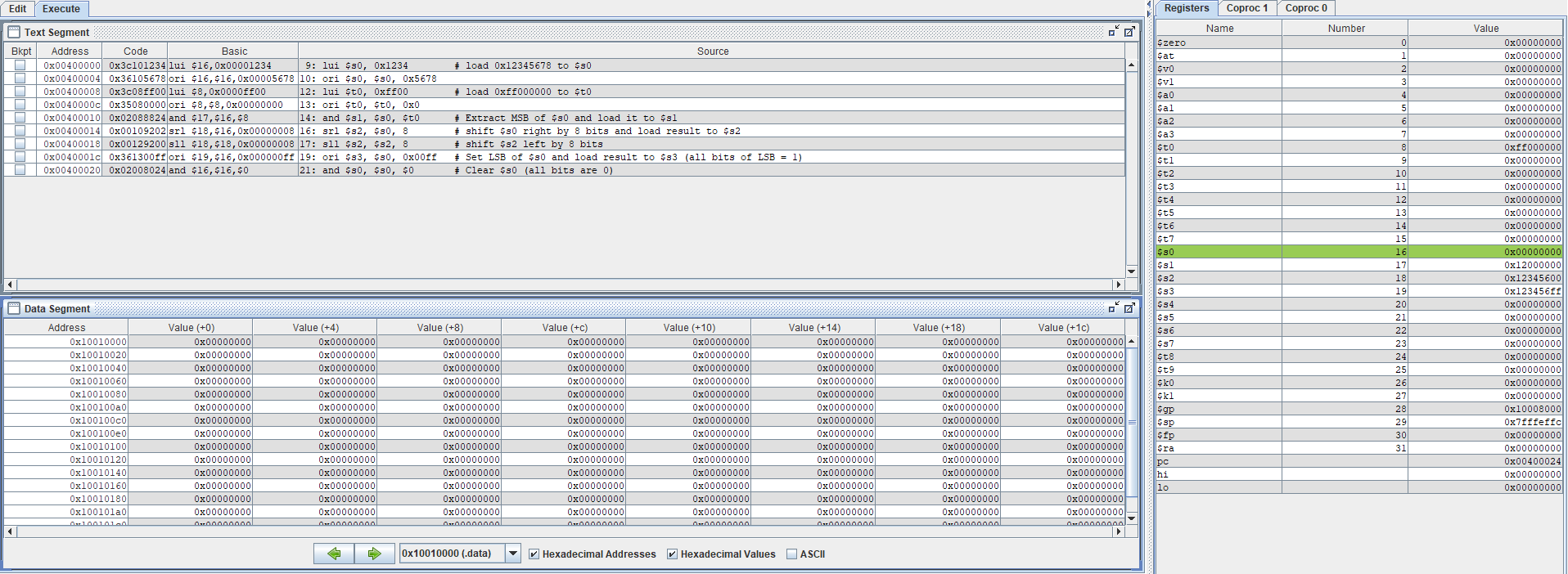
$s0 = 0x12345678

$s1 = 0x12000000 (Extract MSB of $s0)

$s2 = 0x12345600 (Clear LSB of $s0)

$s3 = 0x123456ff (Set LSB of $s0)

$s0 = 0x00000000 (Clear $s0)



**Asignment 3:**

# a. abs $s0, $s1

# b. move $s0, $s1

# c. not $s0

# d. ble $s1, $s2, L

.text

start:

addi $s1, $0, -6 # set $s1 = -6

addi $s2, $0, 5 # set $s2 = 5

# a)

bltz $s1, negative # Test if $s1 is negative

add $s0, $0, $s1 # $s0 = $s1

j end\_a

negative: # $s1 < 0

sub $s0, $0, $s1 # $s0 = -$s1

end\_a:

# b)

add $s0, $s1, $0 # $s0 = $s1 + 0 = $s1

end\_b:

# c)

# Note: a nor a = not(a)

nor $s0, $s0, $s0 # Set $s0 to NOT($s0) (Invert bits of $s0)

end\_c:

# d)

sub $t3, $s1, $s2 # $t3 = $s1 - $s2

blez $t3, L # Branch to L if ($s1 - $s2 <= 0)

j end\_d

L:

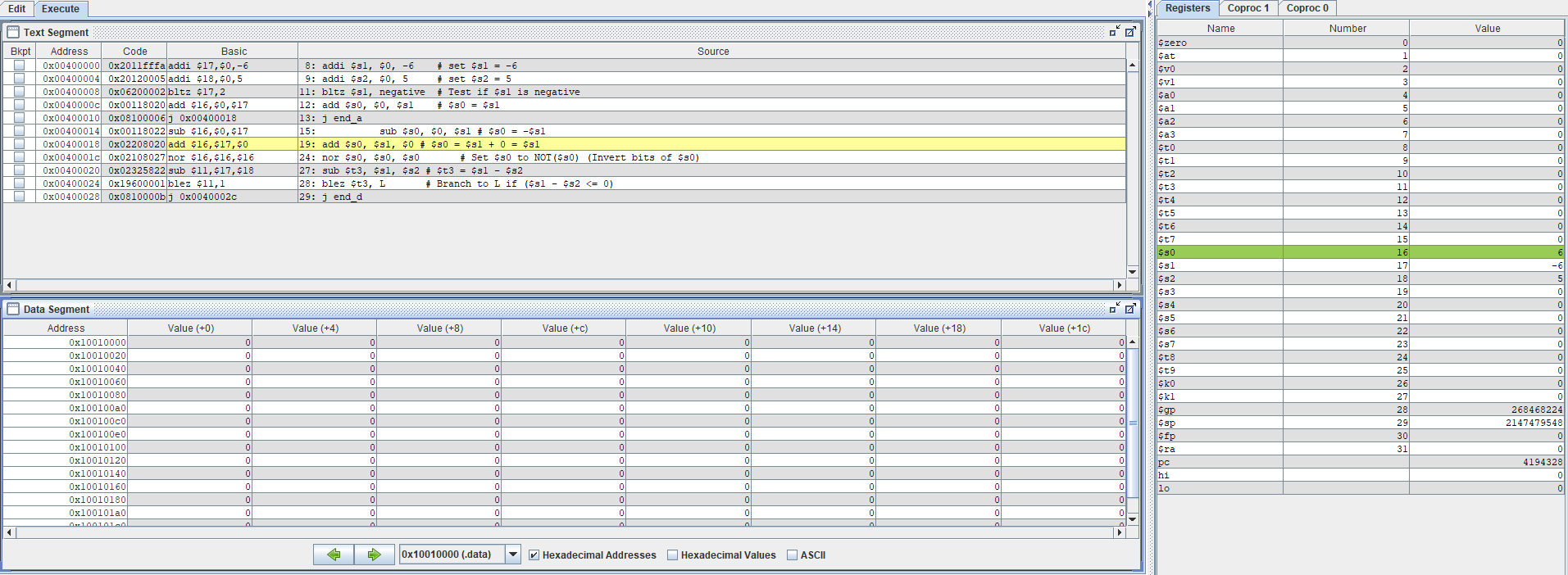
# some instructions ...

end\_d:

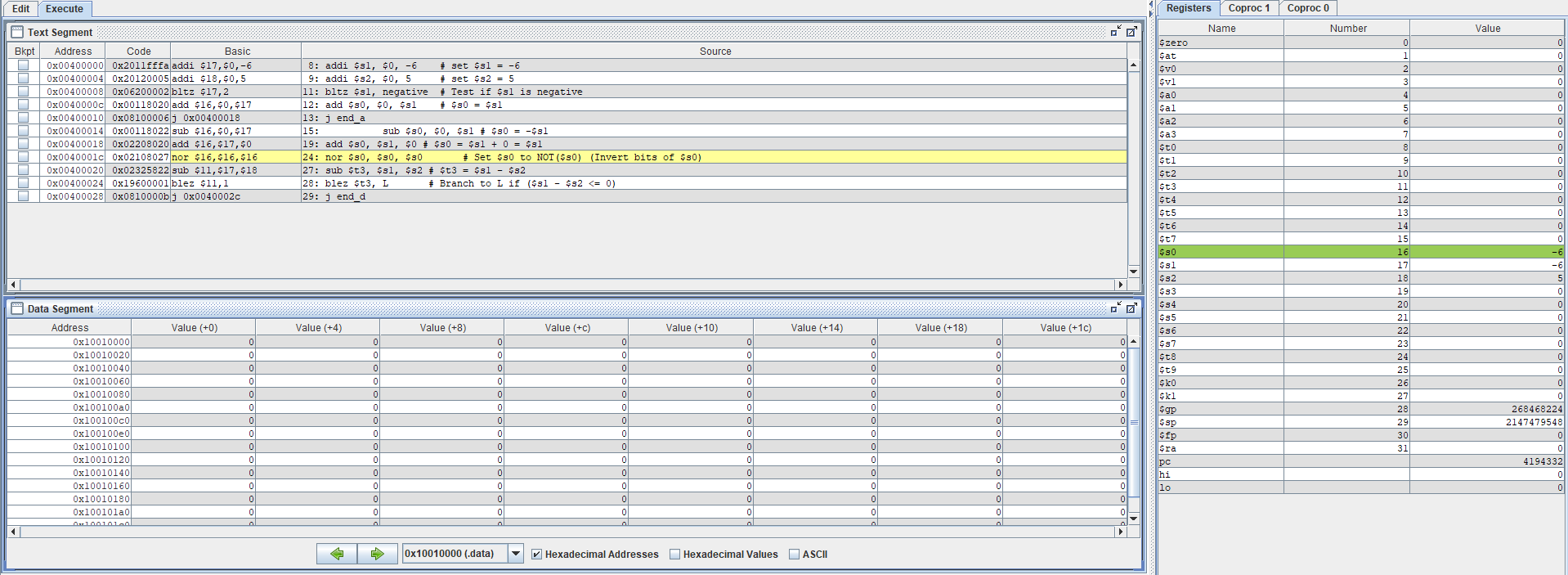
$s1 = -6

$s2 = 5

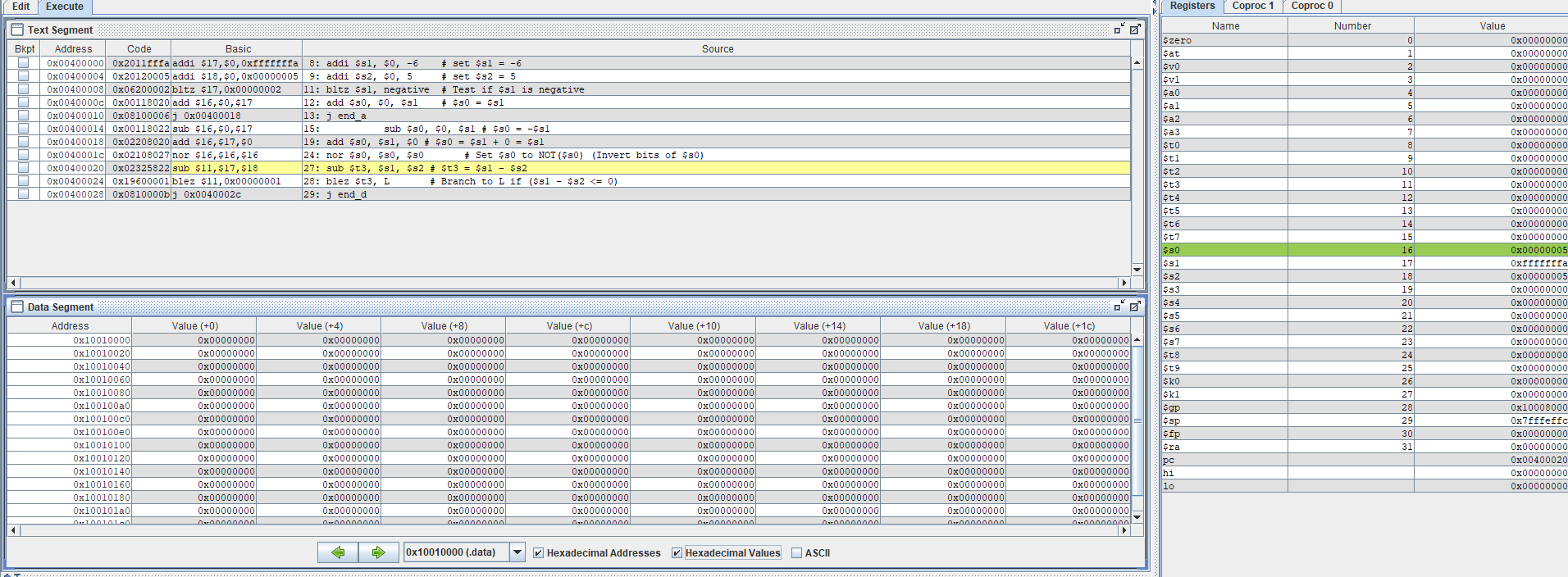
1. $s0 = |$s1| = 6



1. $s0 = $s1 = 6

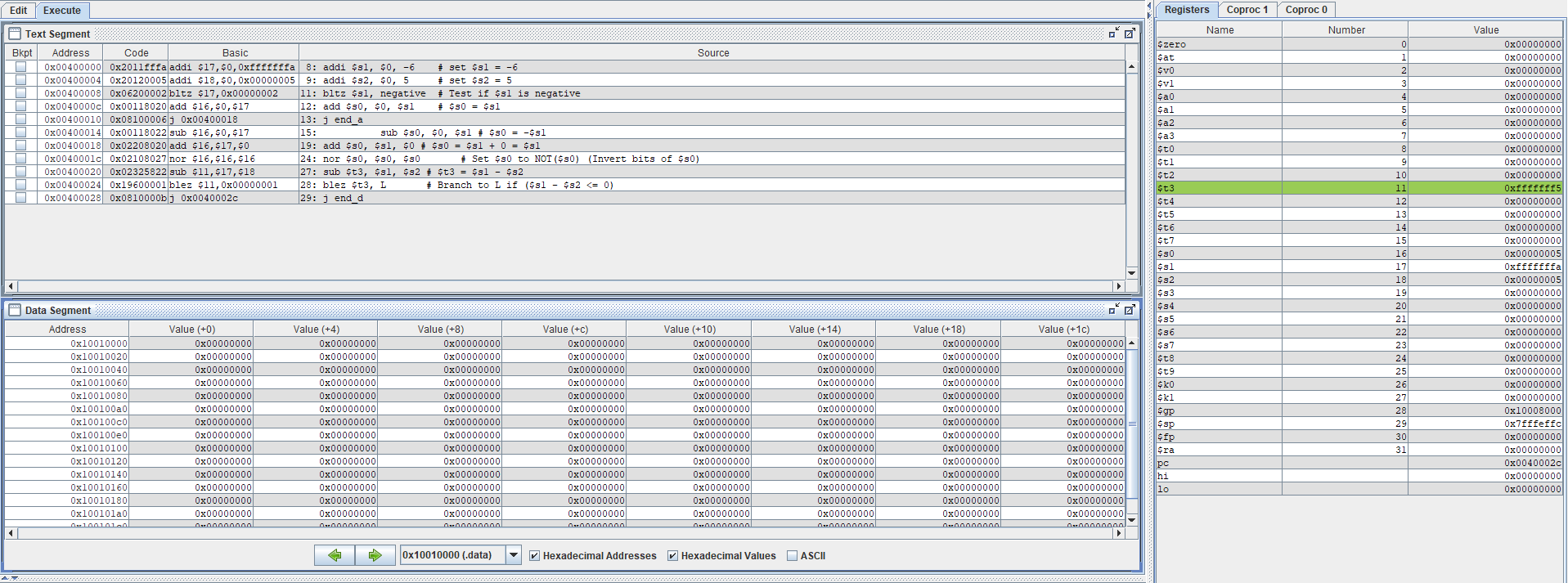


1. $s0 = NOT ($s1) = 0x00000005 ($s1 = 0xfffffffa)



1. $s1 < $s2 (-6 < 5)

Do L instructions



**Asignment 4:**

.text

start:

addi $s1, $0, 0x7fffffff # Load 0x7fffffff to $s1

addi $s2, $s0, 0x1 # Load 0x1 to $s1

addi $t0, $0, 0 # Set default status is No Overflow

addu $s3, $s1, $s2 # $s3 = $s1 + $s2

xor $t1, $s1, $s2 # Test if $s1 and $s2 have the same sign

bltz $t1, No\_Overflow # if not, no Overflow

# if have

xor $t2, $s3, $s1 # Test if $s1 and $s3 have the same sign

bgez $t2, No\_Overflow # if have, no Overflow

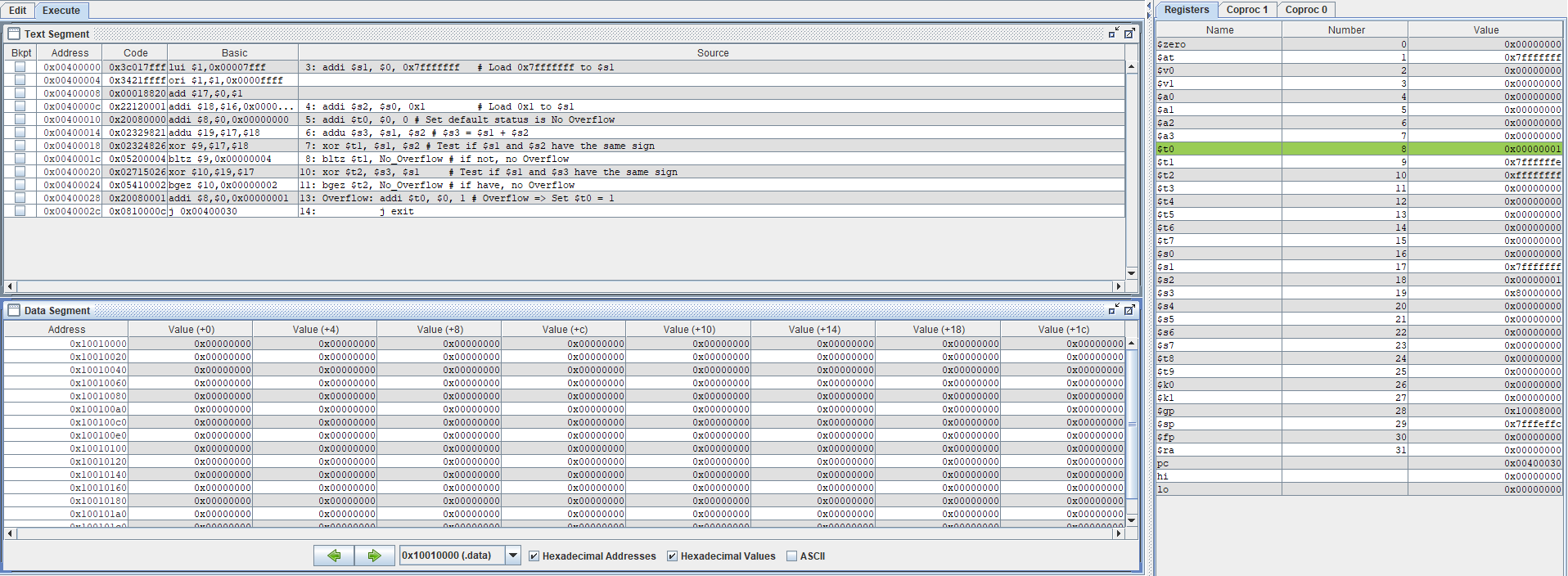
Overflow: addi $t0, $0, 1 # Overflow => Set $t0 = 1

j exit

No\_Overflow:

exit:

Trường hợp xảy ra tràn $s1 = 0x7fffffff > 0, $s2 = 1 > 0 => $s3 = 0x80000000 < 0



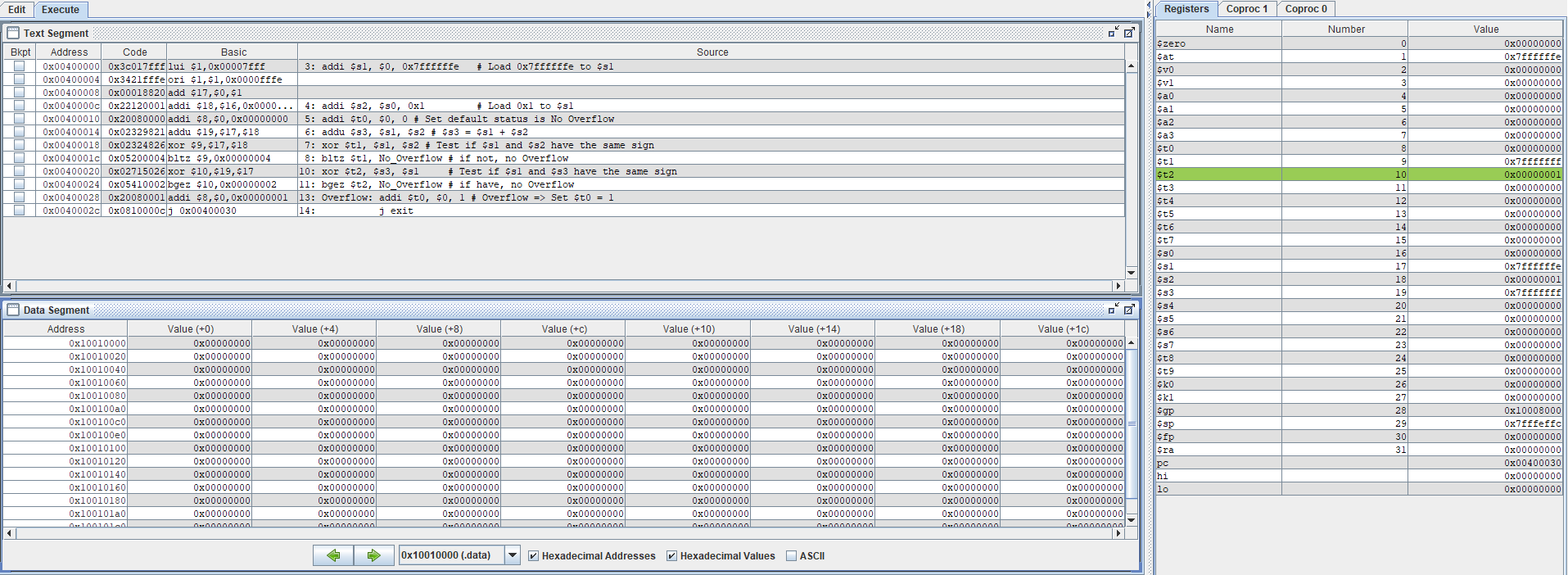
Chỉnh lại 2 dòng ở mã trên thành:

addi $s1, $0, 0x7ffffffe # Load 0x7ffffffe to $s1

addi $s2, $s0, 0x1 # Load 0x1 to $s1

Trường hợp không xảy ra tràn $s1 = 0x7ffffffe (2 ^ 31 – 1), $s2 = 0x1 (1)

=> $s3 = 0x7fffffff



**Asignment 5:**

# The power load in $s2

# The number to be multiplied load in $s1

# The result load in $s3

.text

start:

addi $s1, $0, 5 # number to be multiplied = 5

addi $s2, $0, 10 # power = 10

sllv $s3, $s1, $s2 # $s3 = 5 \* (2 ^ 4)

exit:

$s1 = 5 (Number to be multiplied)

$s2 = 10 (Power)

$s3 = $s1 \* (2 ^ $s2) = 5 \* ( 2 ^ 10) = 5120

