

ASSIGNMENT WEEK 2

-Họ tên: Hồ Tuấn Huy
-MSSV: 20225856

Assignment 1:

Assignment 1: lệnh gán số 16-bit

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 1
.text
    addi    $s0, $zero, 0x3007 # $s0 = 0 + 0x3007 = 0x3007 ;I-type
    add     $s0, $zero, $0      # $s0 = 0 + 0 = 0           ;R-type
```

Sau đó:

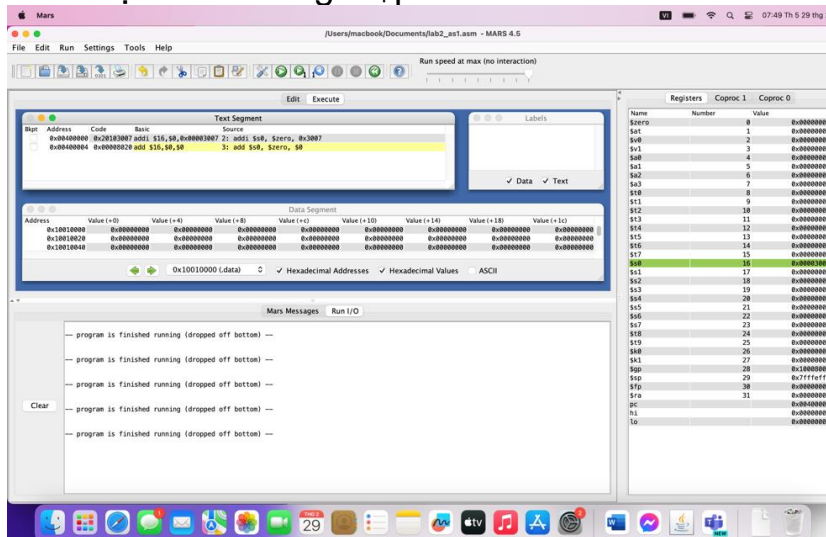
- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại,
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của thanh ghi \$s0
 - o Sự thay đổi giá trị của thanh ghi \$pc
- Ở cửa sổ Text Segment, hãy so sánh mã máy của các lệnh trên với khuôn dạng lệnh để chứng tỏ các lệnh đó đúng như tập lệnh đã qui định
- Sửa lại lệnh lui như bên dưới. Chuyện gì xảy ra sau đó. Hãy giải thích

```
addi    $s0, $zero, 0x2110003d
```

-Ở lệnh addi \$s0, \$zero, 0x3007:

+Giá trị của thanh ghi \$s0 là 0x00003007

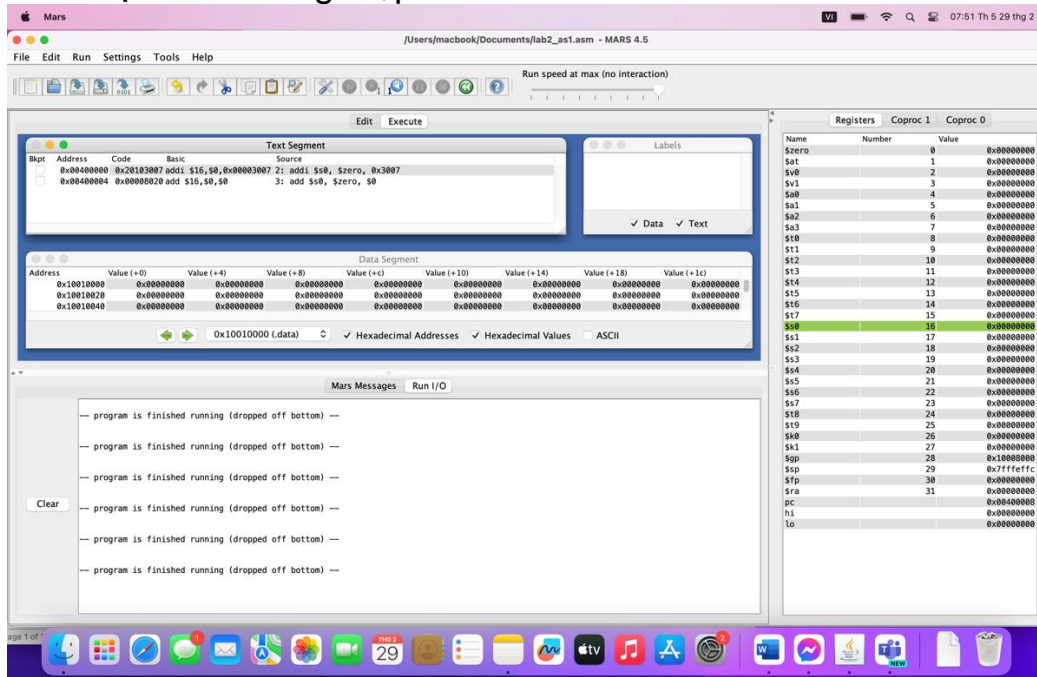
+Giá trị của thanh ghi \$pc là 0x00400004



-Ở lệnh **add \$s0, \$zero, \$0:**

+Giá trị của thanh ghi \$s0 là 0x00000000

+Giá trị của thanh ghi \$pc là 0x00400008



-Ở cửa sổ **Text Segment:**

addi \$s0, \$zero, 0x3007

addi \$16, \$0, 0x3007

opcode: 8 => 001000

rs: \$0 => 00000

rt: \$16 => 10000

imm: 0x00003007 => 0011 0000 0000 0111

Answer: 0010 0000 0001 0000 0011 0000 0000 0111

0x20103007

add \$s0, \$zero, \$0

add \$16, \$0, \$0

opcode: 0 => 000000

rs: \$0 => 00000

rt: \$0 => 00000

rd: \$16 => 10000

sh: 00000

fn: 100000

Answer: 0000 0000 0000 0000 1000 0000 0010 0000

0x00008020

-Sửa lại lệnh lui:

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c012110	lui \$1,0x00002110	2: addi \$s0, \$zero, 0x2110003d
<input type="checkbox"/>	0x00400004	0x3421003d	ori \$1,\$1,0x0000003d	
<input type="checkbox"/>	0x00400008	0x00018020	add \$16,\$0,\$1	
<input type="checkbox"/>	0x0040000c	0x00008020	add \$16,\$0,\$0	3: add \$s0, \$zero, \$0

Vì 0x2110003d là 32-bit nên khi chạy câu lệnh sẽ tách thành 2 phần, mỗi phần 16-bit là 0x00002110 (bằng lệnh lui) và 0x0000003d (bằng lệnh ori).

Assignment 2:

Assignment 2: lệnh gán số 32-bit


Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 2
```

*Ha Noi University of Science and Technology
School of Information and Communication Technology*

```
.text
lui    $s0,0x2110    #put upper half of pattern in $s0
ori    $s0,$s0,0x003d #put lower half of pattern in $s0
```

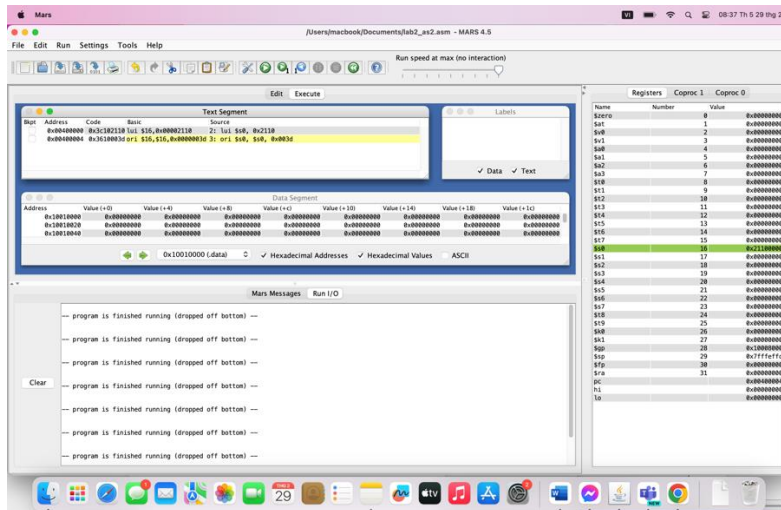
Sau đó:

- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của thanh ghi \$s0
 - o Sự thay đổi giá trị của thanh ghi \$pc
- Ở cửa sổ Data Segment, hãy click vào hộp combo để chuyển tới quan sát các byte trong vùng lệnh .text.
 - o Kiểm tra xem các byte đầu tiên ở vùng lệnh trùng với cột nào trong cửa sổ Text Segment.

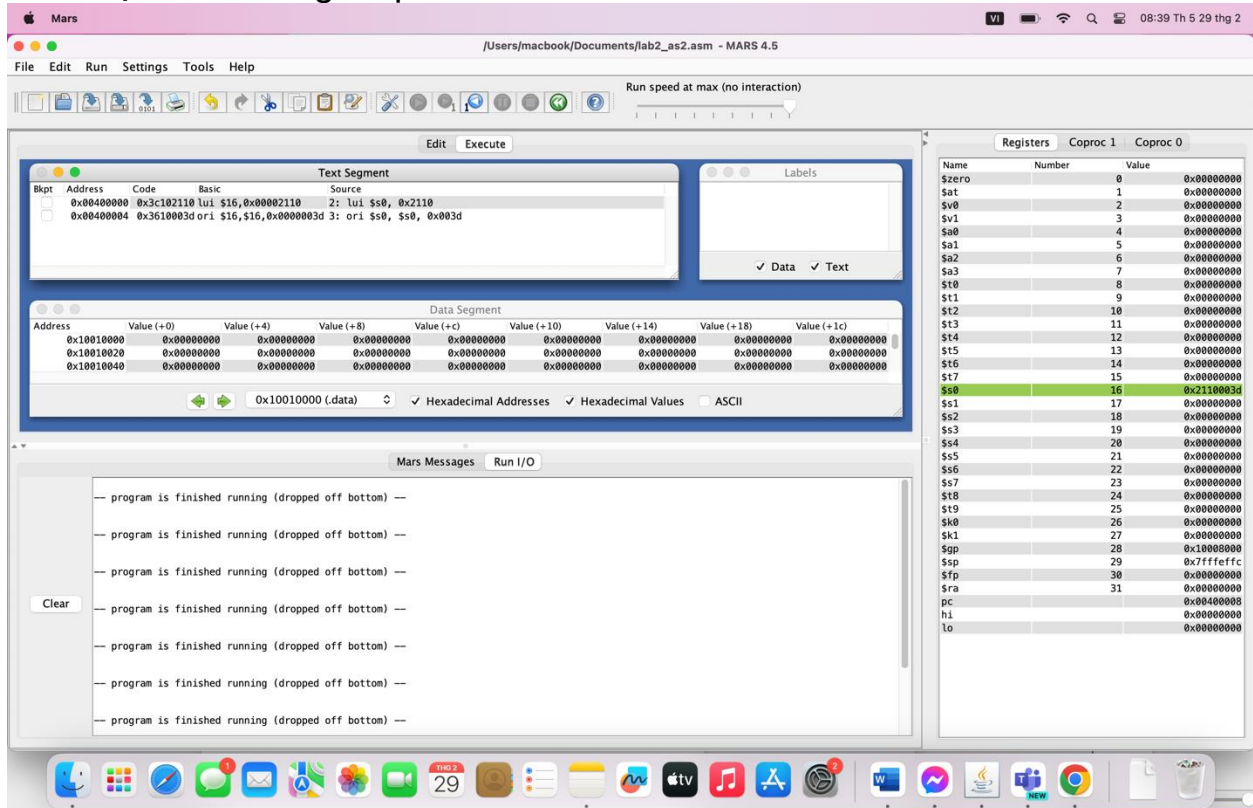
-Ở lệnh lui \$s0, 0x2110:

+Giá trị thanh ghi \$s0 là 0x21100000

+Giá trị thanh ghi \$pc là 0x00400004



-Ở lệnh `ori $s0, $s0, 0x003d`:
+Giá trị của thanh ghi \$s0 là 0x2110003d
+Giá trị của thanh ghi \$pc là 0x00400008



-Ở cửa sổ Data Segment:

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00400000	0x3c102110	0x3610003d	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x00400000 (.text) ☒ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

+Lệnh lui \$s0, 0x2110 có địa chỉ 0x00400000 nằm ở hàng 1, cột 1.

+Lệnh ori \$s0, \$s0, 0x003d có địa chỉ 0x00400004 nằm ở hàng 1, cột 2.

Assignment 3:

Assignment 3: lệnh gán (giả lệnh)

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 3
.text
    li    $s0,0x2110003d #pseudo instruction=2 basic instructions
    li    $s1,0x2        #but if the immediate value is small, one ins
```

Sau đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment. Giải thích điều bất thường?

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c012110	lui \$1,0x00002110	2: li \$s0, 0x2110003d
<input type="checkbox"/>	0x00400004	0x3430003d	ori \$16,\$1,0x0000003d	
<input type="checkbox"/>	0x00400008	0x24110002	addiu \$17,\$0,0x0000...	3: li \$s1, 0x2

Có sự khác nhau ở 2 lệnh vì:

+Ở lệnh đầu tiên giá trị cần nạp vào là tương đối lớn nên lệnh li sẽ tách ra làm 2 lệnh là lui và ori.

+Ở lệnh thứ 2 giá trị 0x2 là nhỏ nên lệnh li sẽ tương đương lệnh addiu mà không cần tách ra thành 2 lệnh như ở lệnh trên.


Assignment 4:

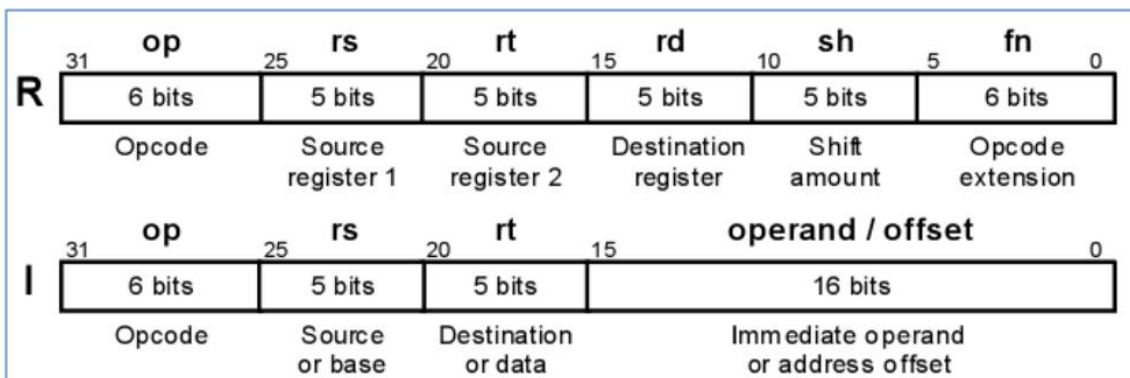
Assignment 4: tính biểu thức $2x + y = ?$

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 4
.text
# Assign X, Y
addi $t1, $zero, 5    # X = $t1 = ?
addi $t2, $zero, -1   # Y = $t2 = ?
# Expression Z = 2X + Y
add  $s0, $t1, $t1     # $s0 = $t1 + $t1 = X + X = 2X
add  $s0, $s0, $t2     # $s0 = $s0 + $t2 = 2X + Y
```

Sau đó:

- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của các thanh ghi
 - o Sau khi kết thúc chương trình, xem kết quả có đúng không?
- Ở cửa sổ Text Segment, xem các lệnh **addi** và cho biết điểm tương đồng với hợp ngữ và mã máy. Từ đó kiểm nghiệm với khuôn mẫu của kiểu lệnh



- Ở cửa sổ Text Segment, chuyển mã máy của lệnh **add** sang hệ 2. Từ đó kiểm nghiệm với khuôn mẫu của kiểu lệnh R.

-Ở lệnh **addi \$t1, \$zero, 5:**

+Giá trị thanh ghi \$t1 là 0x00000005

+Giá trị thanh ghi \$pc là 0x00400004

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000005
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffefffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400004
hi		0x00000000
lo		0x00000000

-Ở lệnh addi \$t2, \$zero, -1:

+Giá trị thanh ghi \$t2 là 0xffffffff

+Giá trị thanh ghi \$pc là 0x00400008

\$t2	10	0xffffffff
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffefffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400008

-Ở lệnh add \$s0, \$t1, \$t1:

+Giá trị thanh ghi \$s0 là 0x0000000a

+Giá trị thanh ghi \$pc là 0x0040000c

\$t7	15	0x00000000
\$s0	16	0x0000000a
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400010
hi		0x00000000
lo		0x00000000

- Ở lệnh add \$s0, \$s0, \$t2:

+Giá trị thanh ghi \$s0 là 0x00000009

+Giá trị thanh ghi \$pc là 0x00400010

=>Sau khi kết thúc chương trình thì thu được kết quả đúng, với giá trị thanh ghi \$s0 là 0x00000009:

\$t7	15	0x00000000
\$s0	16	0x00000009
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400010
hi		0x00000000
lo		0x00000000

- Ở cửa sổ Text Segment, xem các lệnh addi và cho biết điểm tương đồng với hợp ngữ và mã máy. Từ đó kiểm nghiệm với khuôn mẫu của kiểu lệnh I:

0x20090005

001000 00000 01001 0000 0000 0000 0101:

opcode: 001000

rs: \$0

rt: \$9

imm: 5

Answer: addi \$9, \$0, 5

0x200affff

001000 00000 01010 1111 1111 1111 1111:

opcode: 001000

rs: \$0

rt: \$10

imm: -1

Answer: addi \$10, \$0, -1

- Ở cửa sổ Text Segment, chuyển mã máy của lệnh add sang hệ 2. Từ đó kiểm nghiệm với khuôn mẫu của kiểu lệnh R:

0x01298020

000000 01001 01001 1000 0000 0010 0000:

opcode: 000000

rs: \$9

rt: \$9

rd:\$16

shamt: 00000

funct: 100000

Answer: add \$16, \$9, \$9

0x020a8020

000000 10000 01010 1000 0000 0010 0000:

opcode: 000000

rs: \$16

rt: \$10

rd:\$16

shamt: 00000

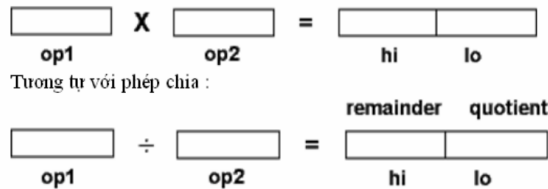
funct: 100000

Answer: add \$16, \$16, \$10

Assignment 5:

b) Thanh ghi HI và LO


Thao tác nhân của MIPS có kết quả chứa 2 thanh ghi HI và LO, đây không phải là thanh ghi đa năng. Bit 32 đến 63 thuộc HI và 0 đến 31 thuộc LO



Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 5
.text
# Assign X, Y
addi $t1, $zero, 4    # X = $t1 = ?
addi $t2, $zero, 5    # Y = $t2 = ?
# 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 đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment. Giải thích điều bất thường?
- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của các thanh ghi, đặc biệt là Hi, Lo
 - o Sau khi kết thúc chương trình, xem kết quả có đúng không?

-Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment. Giải thích điều bất thường:

Gán giá trị cho X và Y:

addi \$t1, \$zero, 4: Lệnh này đặt giá trị của thanh ghi \$t1 (X) thành 4.

addi \$t2, \$zero, 5: Tương tự, điều này đặt giá trị của thanh ghi \$t2 (Y) thành 5.

Biểu thức $Z = 3 * XY$:

mul \$s0, \$t1, \$t2: Nhân giá trị của X và Y và lưu kết quả vào \$s0. Sản phẩm được tính toán chính xác.

mul \$s0, \$s0, 3: Tiếp theo, chúng ta nhân kết quả của \$s0 với 3 (để được $3 * XY$).

$Z' = Z$:

mflo \$s1: Lệnh này di chuyển giá trị từ phần thấp nhất của kết quả nhân (được lưu trong \$s0) sang \$s1.

Vì kết quả phép nhân sẽ là **64bit** nên kết quả của phép nhân sẽ gồm **2 phần hi và lo**.

-Ở mỗi lệnh, quan sát cửa sổ Register và chú ý sự thay đổi các thanh ghi:

+Ở lệnh addi \$t1, \$zero, 4:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000004
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400004
hi		0x00000000
lo		0x00000000

+Ở lệnh addi \$t2, \$zero, 5:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000004
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffefffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400008
hi		0x00000000
lo		0x00000000

+Ở lệnh mul \$s0, \$t1, \$t2:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000004
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000014
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffefffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x0040000c
hi		0x00000000
lo		0x00000014

+Ở lệnh mul \$s0, \$s0, 3:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000003
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000004
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000014
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400010
hi		0x00000000
lo		0x00000014

+Ở lệnh mflo \$s1:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000003
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000004
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x0000003c
\$s1	17	0x0000003c
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400018
hi		0x00000000
lo		0x0000003c

=> Sau khi kết thúc chương trình, kết quả của \$s0 là 0x0000003c => Chuyển sang số thập phân là $3 \cdot 16 + 12 = 60$ => Đúng với kết quả phép nhân.

Assignment 6:

Assignment 6: tạo biến và truy cập biến

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 6
.data                                # DECLARE VARIABLES
X : .word    5                      # Variable X, word type, init value =
```

*Ha Noi University of Science and Technology
School of Information and Communication Technology*


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

# Calculate 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
```

Sau đó:

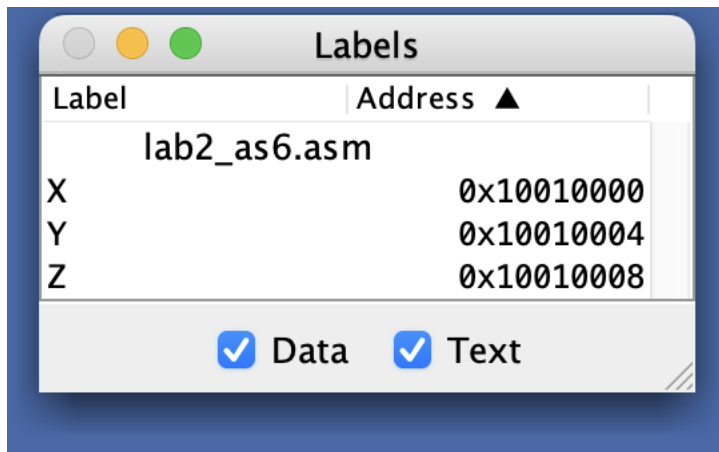
- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment.
 - o Lệnh **la** được biên dịch như thế nào?
- Ở cửa sổ Label và quan sát địa chỉ của X, Y, Z.
 - o So sánh chúng với hằng số khi biên dịch lệnh **la** thành mã máy
 - o Click đúp vào các biến X, Y, Z để công cụ tự động nhảy tới vị trí của biến X, Y, Z trong bộ nhớ ở cửa sổ Data Segment. Hãy bảo đảm các giá trị đó đúng như các giá trị khởi tạo.
- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của các thanh ghi
 - o Xác định vai trò của lệnh **lw** và **sw**
- Ghi nhớ qui tắc xử lý
 - o Đưa tất cả các biến vào thanh ghi bằng cặp lệnh **la, lw**
 - o Xử lý dữ liệu trên thanh ghi
 - o Lưu kết quả từ thanh ghi trở lại biến bằng cặp lệnh **la, sw**
- Tìm hiểu thêm các lệnh **lb, sb**

- **Lệnh la trong tập hợp MIPS được sử dụng để tải địa chỉ của nhãn hoặc biến vào thanh ghi.**

+la \$t8, X: Lệnh này lấy địa chỉ của nhãn X trong đoạn dữ liệu và lưu nó vào thanh ghi \$t8.

+la \$t9, Y: Tương tự, nó lấy địa chỉ của nhãn Y và lưu vào thanh ghi \$t9.

-**Ở cửa sổ Label:**

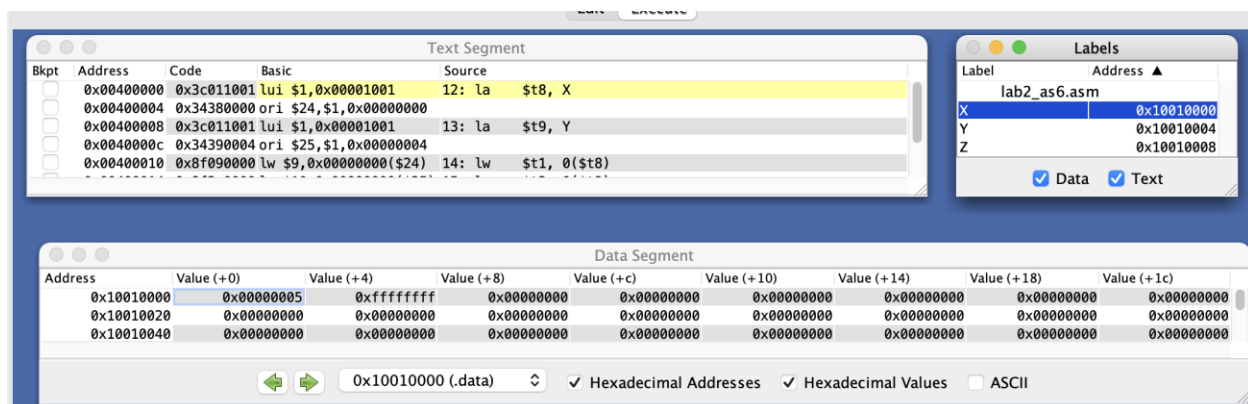


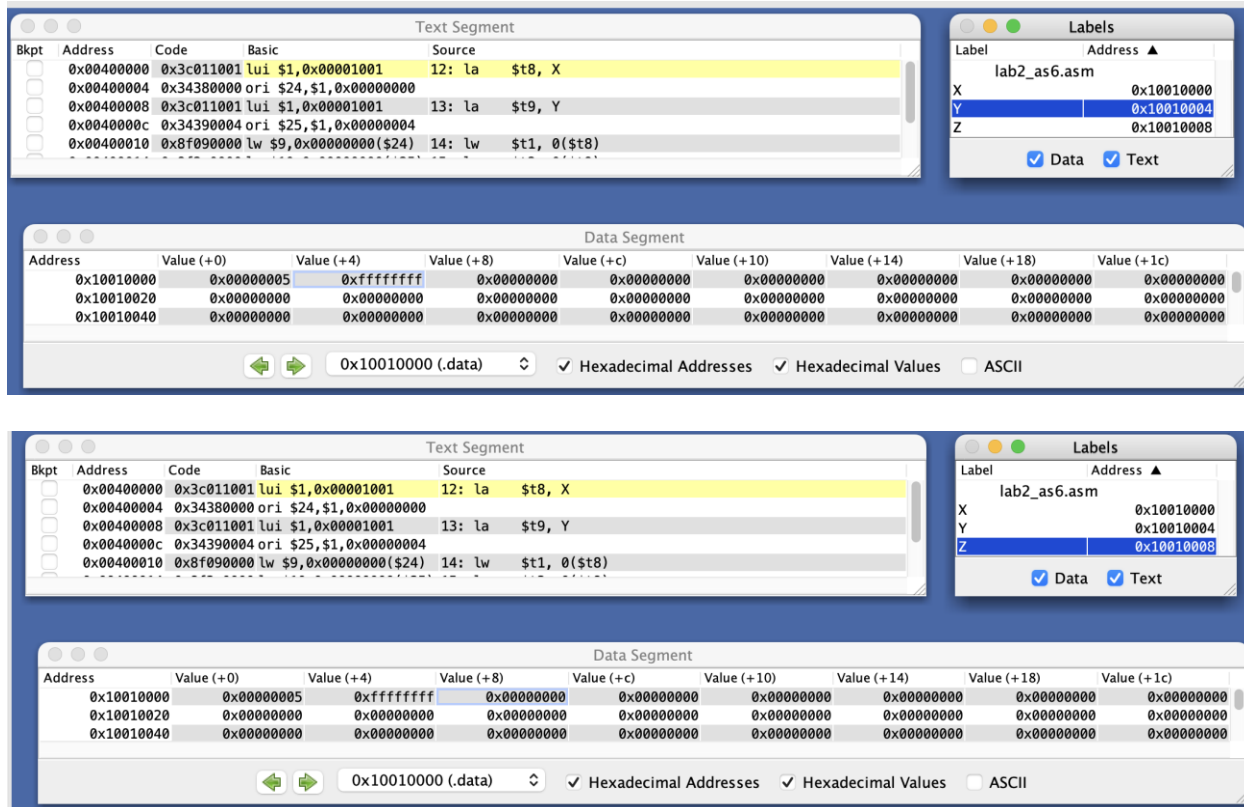
+Địa chỉ của X là 0x10010000, giá trị khởi tạo của X là 5.

+Địa chỉ của Y là 0x10010004 (vì X đã chiếm 4 byte trước đó), giá trị khởi tạo của Y là -1.

+Địa chỉ của Z là 0x10010008 (vì Y đã chiếm 4 byte trước đó), giá trị khởi tạo ban đầu của Z không có.

-**Click đúp vào các biến X, Y, Z để công cụ tự động nhảy tới vị trí của biến X, Y, Z trong bộ nhớ ở cửa sổ Data Segment. Hãy bảo đảm các giá trị đó đúng như các giá trị khởi tạo:**





- Ở mỗi lệnh, quan sát cửa sổ Registers, sự thay đổi giá trị của các thanh ghi và vai trò của lệnh lw và sw

1. lw \$t1, 0(\$t8):
 - Lệnh này tải giá trị của biến X từ vị trí bộ nhớ có địa chỉ được lưu trong thanh ghi \$t8.
 - Sau khi thực hiện lệnh này, giá trị của X (5) được nạp vào thanh ghi \$t1.
2. lw \$t2, 0(\$t9):
 - Tương tự, lệnh này tải giá trị của biến Y từ vị trí bộ nhớ có địa chỉ được lưu trong thanh ghi \$t9.
 - Sau khi thực hiện lệnh này, giá trị của Y (-1) được nạp vào thanh ghi \$t2.
3. add \$s0, \$t1, \$t1 và add \$s0, \$s0, \$t2:
 - Hai lệnh này thực hiện phép tính $Z = 2X + Y$.
 - Giá trị của X (5) được nhân với 2 và cộng với giá trị của Y (-1) để tính toán giá trị của Z.
 - Kết quả được lưu trong thanh ghi \$s0.
4. sw \$s0, 0(\$t7):

- Lệnh này lưu giá trị của Z từ thanh ghi \$s0 vào vị trí bộ nhớ có địa chỉ được lưu trong thanh ghi \$t7.
- Điều này đảm bảo rằng giá trị của Z được cập nhật trong bộ nhớ.

- Tìm hiểu thêm các lệnh lb, sb:

1. lb (**load byte**):

- Lệnh này được sử dụng để tải một byte từ một vị trí trong bộ nhớ vào thanh ghi.
- Cú pháp: lb R, Address, trong đó R là một thanh ghi tổng quát và Address là một địa chỉ nhãn hoặc địa chỉ dịch chuyển cơ sở. Địa chỉ ở đây là địa chỉ byte.
- Ví dụ: lb \$t0, 1(\$s3) tải byte thứ nhất từ vị trí bộ nhớ có địa chỉ là \$s3+1.

2. sb (**store byte**):

- Lệnh này được sử dụng để lưu một byte từ thanh ghi vào một địa chỉ trong bộ nhớ.
- Cú pháp: sb R, Address, trong đó R là một thanh ghi tổng quát và Address là một địa chỉ nhãn hoặc địa chỉ dịch chuyển cơ sở. Địa chỉ ở đây cũng là địa chỉ byte.
- Ví dụ: sb \$t0, 6(\$s3) lưu byte từ thanh ghi \$t0 vào địa chỉ \$s3+6 trong bộ nhớ.