

BÀI BÁO CÁO TUẦN 2

Họ và tên: Nguyễn Trọng Khánh Duy

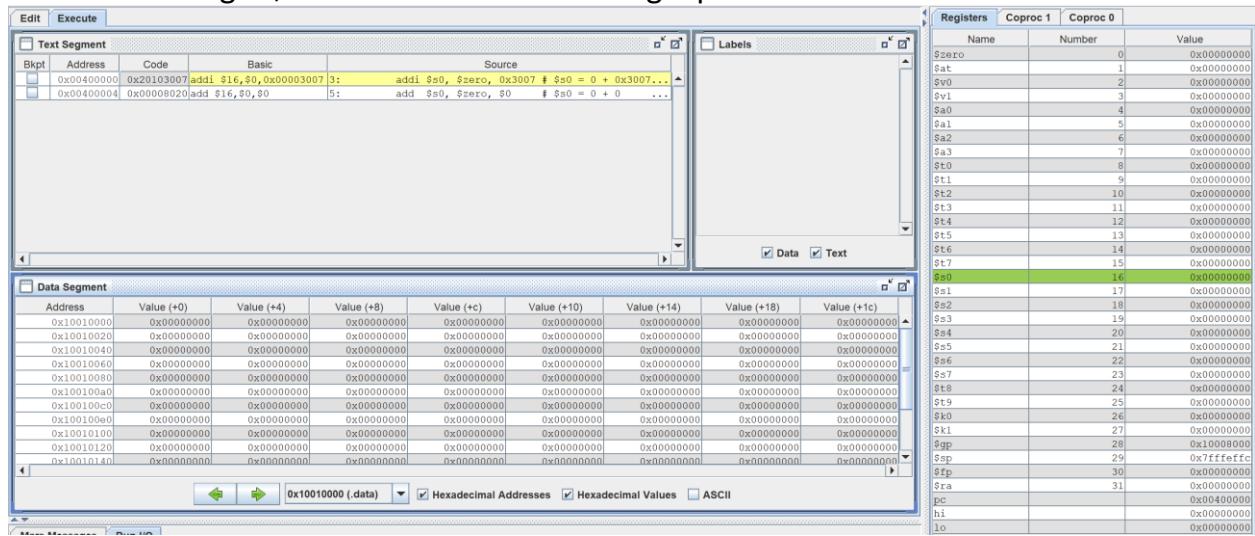
MSSV: 20210284

Assignment 1: Lệnh gán số 16-bit

1. Cửa sổ Register:

- Quan sát thanh ghi \$s0 và thanh ghi pc:

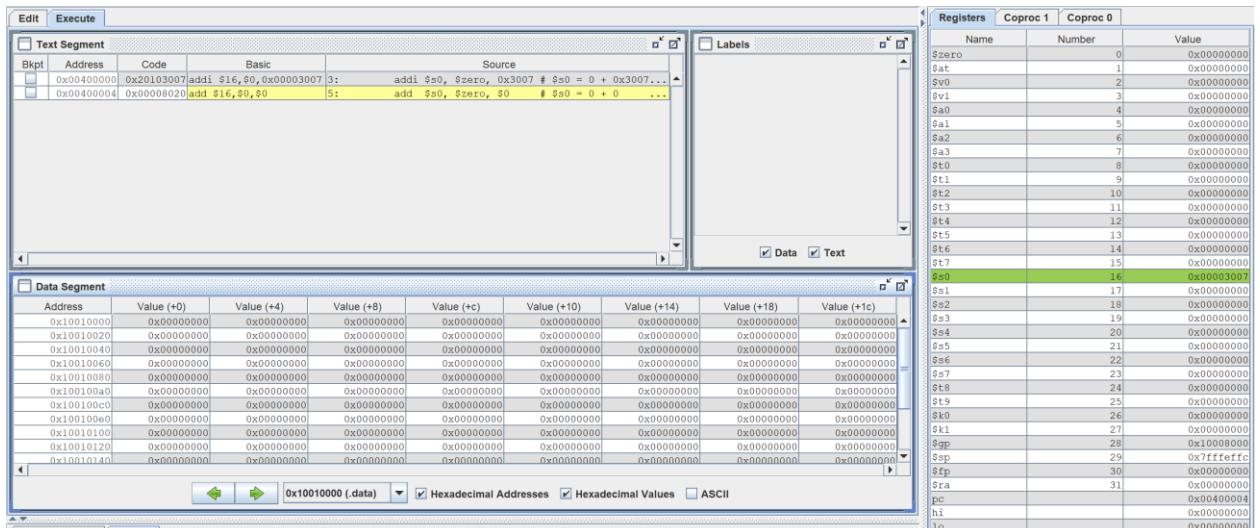
- Ban đầu thanh ghi \$s0 = 0x00000000 và thanh ghi pc = 0x00400000



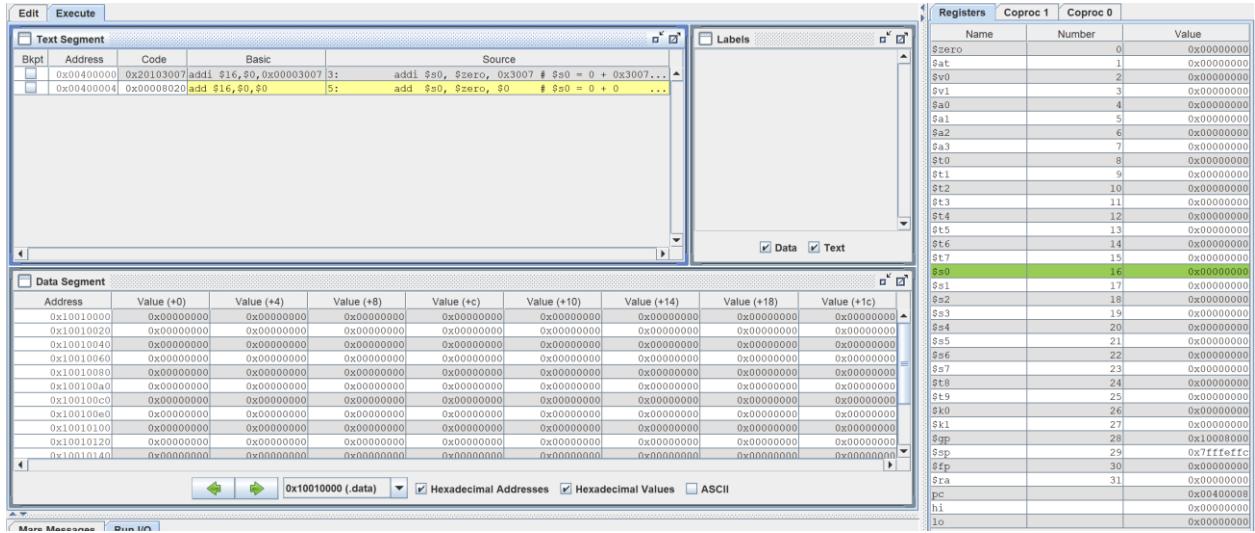
- Thực hiện lệnh addi \$s0, \$zero, 0x3007. Thanh ghi \$s0 = 0x00003007, thanh ghi pc = 0x00400004

Giải thích:

- \$s0 = 0 + 0x3007 (theo kiểu I) => \$s0 = 0x00000000 + 0x00003007 = 0x0003007
- Thanh ghi pc tăng lên 4 bit để thực hiện tiếp lệnh trong chương trình => pc = 0x00400000 + 0x4 = 0x00400004



- Thực hiện lệnh **add \$s0, \$zero, \$0**. Thanh ghi \$s0 = 0x00000000, thanh ghi pc = 0x00400008
 - $\$s0 = 0 + 0$ (theo kiểu R) $\Rightarrow \$s0 = 0x00000000 + 0x00000000 = 0x00000000$
 - Thanh ghi pc tăng lên 4 bit để thực hiện tiếp lệnh trong chương trình $\Rightarrow pc = 0x00400004 + 0x4 = 0x00400008$



2. Cửa sổ Text Segment:

Text Segment								
Bkpt	Address	Code	Basic	Source				
	0x00400000	0x20103007	addi \$16,\$0,0x00003007	3:	addi \$s0, \$zero, 0x3007 # \$s0 = 0 + 0x3007...			
	0x00400004	0x00008020	add \$16,\$0,\$0	5:	add \$s0, \$zero, \$0 # \$s0 = 0 + 0 ...			

- addi \$s0, \$zero, 0x3007** là lệnh kiểu I
 $OP = 8 = 001000_2$
 $\$rs = \$zero = 0_{10} = 00000_2$
 $\$rt = \$s0 = 16_{10} = 10000_2$
 $imm = 0x3007 = 0011 0000 0000 1111_2$
 $\Rightarrow 001000 00000 10000 0011 0000 0000 1111_2 = 0010 0000 0001 0000 0011 0000 0000 1111_2 = 0x20103007$
 \Rightarrow Đúng với mã lệnh trong cửa sổ Text Segment

- add \$s0, \$zero, \$0** là lệnh kiểu R
 $OP = 0 = 00000_2$
 $\$rs = \$zero = 0_{10} = 000000_2$
 $\$rt = \$0 = 0_{10} = 00000_2$

$\$rd = \$s0 = 16_{10} = 10000_2$
 $shamt = 00000_2$
 $funct = 32_{10} = 100000$
 $\Rightarrow 000000\ 00000\ 00000\ 10000\ 00000\ 100000_2 = 0000\ 0000\ 0000\ 0000\ 0000\ 1000\ 0000$
 $0010\ 0000_2 = 0x00008020$
 \Rightarrow Đúng với mã lệnh trong cửa sổ Text Segment

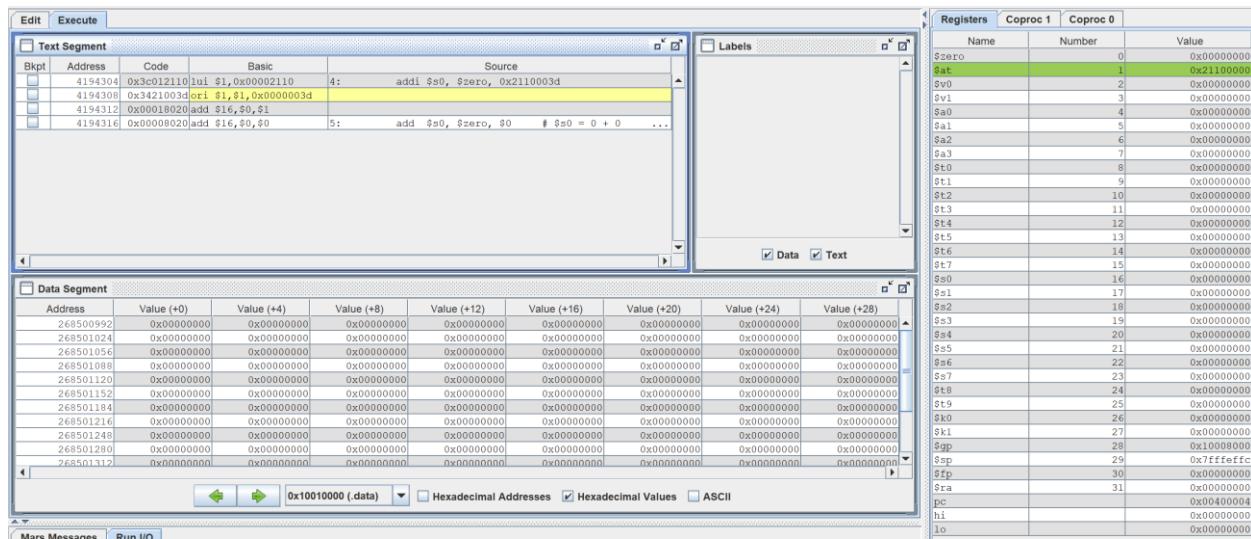
3. Khi sửa câu lệnh addi thành addi \$s0, \$zero, 0x2110003d thì có hiện tượng xảy ra:

- Câu lệnh **\$s0, \$zero, 0x2110003d** được thực hiện thành 3 câu lệnh :

```

lui $1, 0x00002110
ori $1, $1, 0x0000003d
add $16, $0, $1
  
```

- Giải thích cho hiện tượng trên:
- Do lệnh addi là lệnh kiểu I nên chỉ cộng với imm = 16 bit nhưng mà câu lệnh trên lại cộng với 0x2110003d là 32 bit nên chương trình thực hiện như sau:
Thực hiện lệnh **lui \$1, 0x00002110** => Lưu vào 16 bit đầu của thanh ghi \$1 lúc này giá trị thanh ghi \$1 = 0x21100000.



Thực hiện lệnh **ori \$1, \$1, 0x0000003d** => Thực hiện or 16 bit cuối của \$s1 và 0x0000003d và lưu vào thanh ghi \$1. Mà 16 bit cuối của \$1 = 0x0000 => Sau khi thực hiện lệnh trên \$1 = 0x2110003d

The screenshot shows the Mars 32-bit Simulator interface. The assembly code window displays the instruction `add $16, $0, $1` at address 0x10010020. The registers window shows the value of \$s0 being updated to 0x2110003d. The data segment window shows the memory starting at address 0x10010000.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x2110003d
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$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	0x10000000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400008
hi		0x00000000
lo		0x00000000

- Thực hiện lệnh `add $16, $0, $1` => Thực hiện cộng thanh ghi \$0 và thanh ghi \$1 và lưu vào thanh ghi \$16 (\$s0) => Giá trị của thanh ghi \$s0 = 0x2110003d

The screenshot shows the Mars 32-bit Simulator interface. The assembly code window displays the instruction `add $16, $0, $1` at address 0x10010020. The registers window shows the value of \$s0 being updated to 0x2110003d. The data segment window shows the memory starting at address 0x10010000.

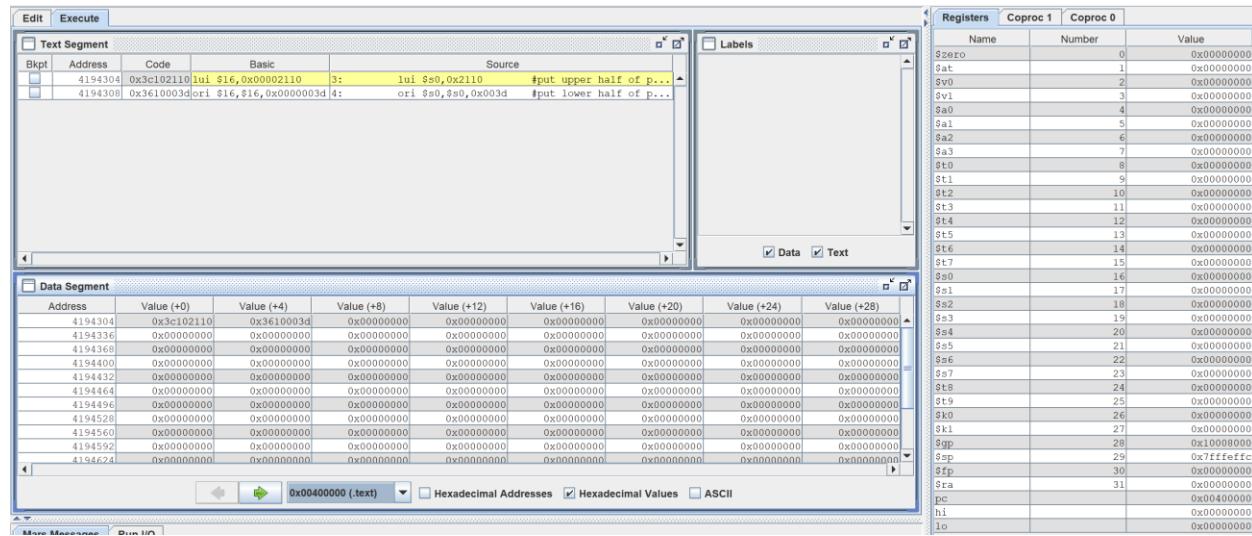
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x2110003d
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x2110003d
\$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	0x10000000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x0040000c
hi		0x00000000
lo		0x00000000

Assignment 2: lệnh gán số 32-bit

1. Cửa sổ Register:

Quan sát thanh ghi \$s0 và thanh ghi pc:

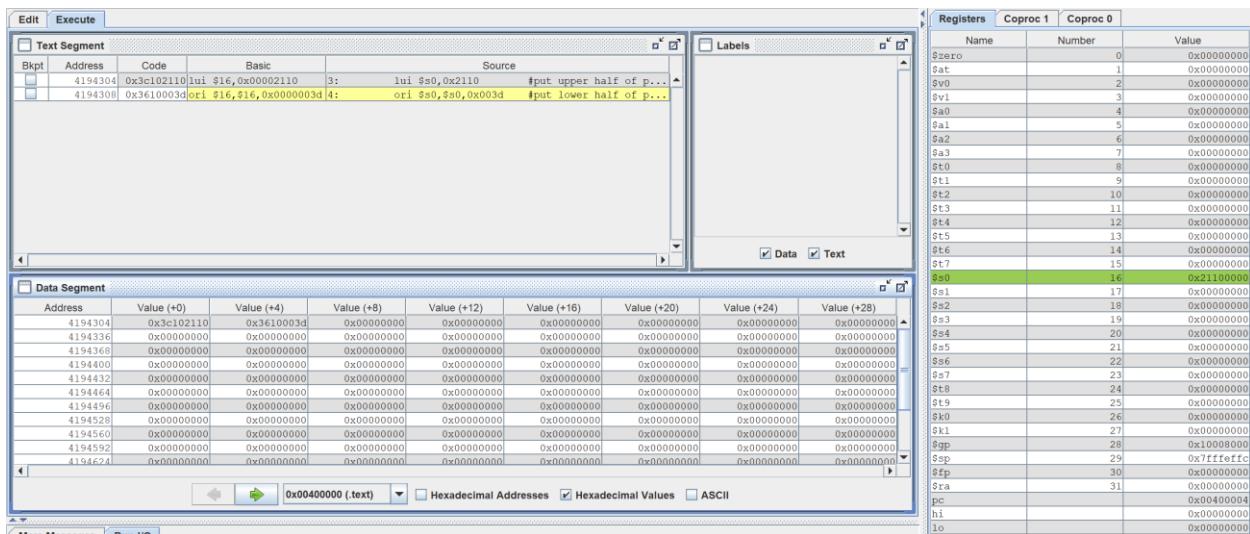
- Ban đầu thanh ghi \$s0 = 0x00000000 và thanh ghi pc = 0x00400000



- Thực hiện lệnh lui \$s0,0x2110. Thanh ghi \$s0 = 0x21100000, thanh ghi pc = 0x00400004

Giải thích:

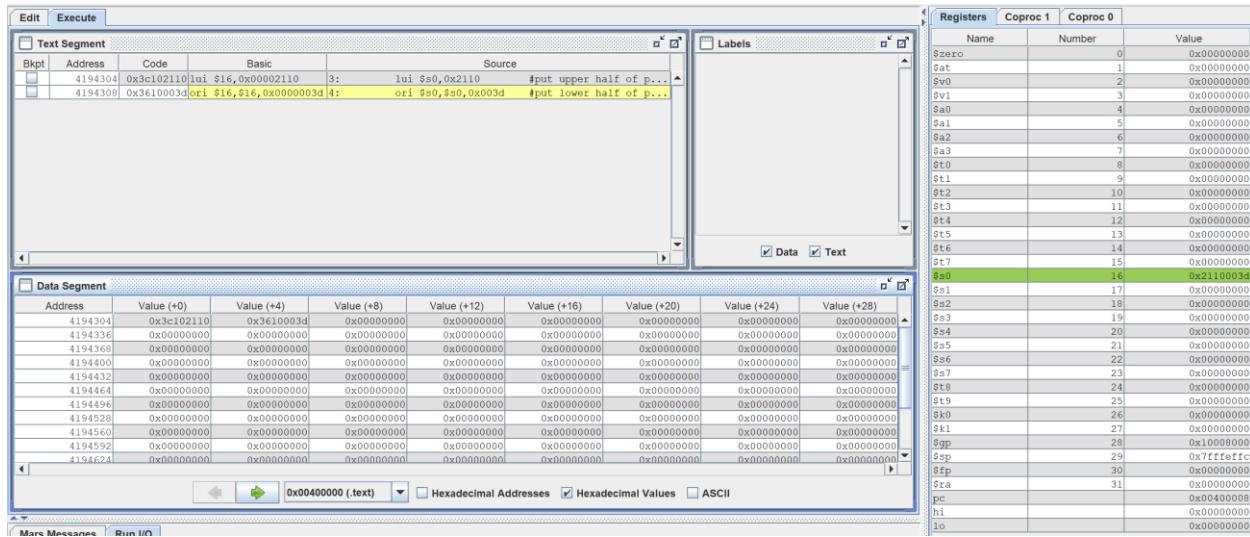
- Lưu vào 16 bit đầu của thanh ghi \$s0 lúc này giá trị thanh ghi \$s0 = 0x21100000
- Thanh ghi pc tăng lên 4 bit để thực hiện tiếp lệnh trong chương trình
=> pc = 0x00400000 + 0x4 = 0x00400004



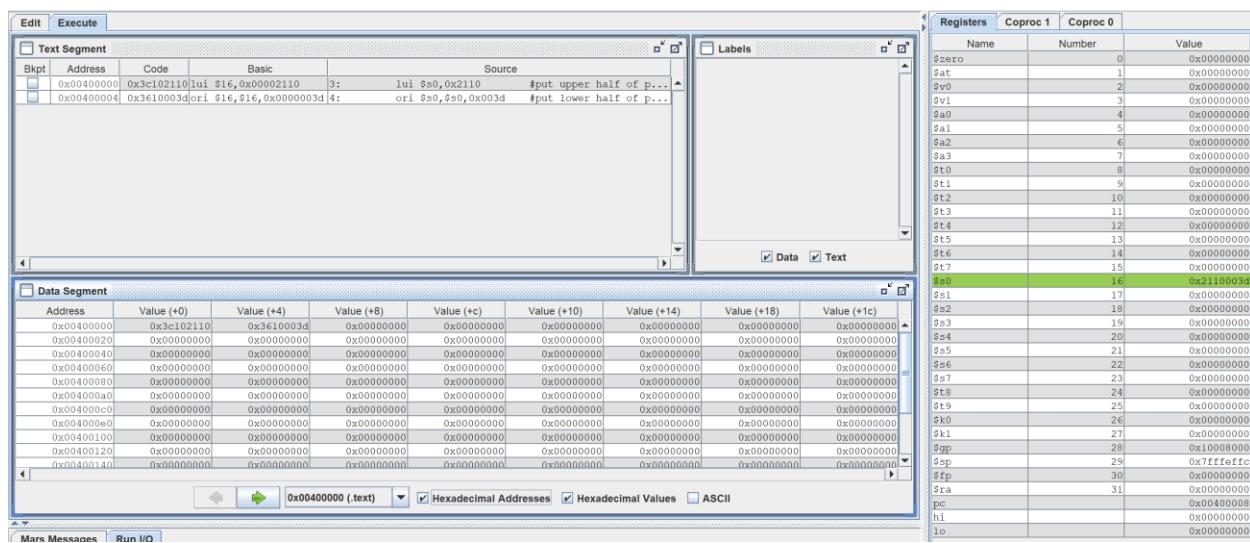
- Thực hiện lệnh **ori \$s0, \$s0, 0x003d**. Thanh ghi \$s0 = 0x2110003d, thanh ghi pc = 0x00400008

Giải thích:

- Lưu vào 16 bit đầu của thanh ghi \$s0 lúc này giá trị thanh ghi \$s0 = 0x2110003d
- Thanh ghi pc tăng lên 4 bit để thực hiện tiếp lệnh trong chương trình
=> pc = 0x00400004 + 0x4 = 0x00400008



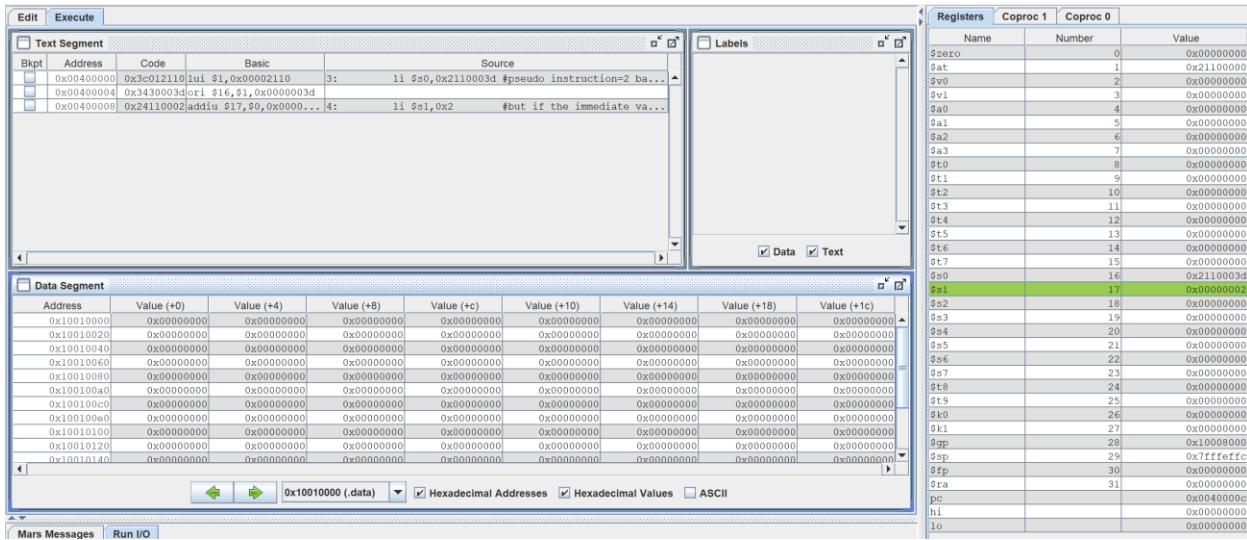
2. Cửa sổ Data Segment:



Nhận xét: Các byte đầu tiên của vùng lệnh trùng với cột Address trong cửa sổ Text Segment

Assignment 3: Lệnh gán (giả lệnh)

Sau khi biên dịch:



- Thực hiện lệnh **li \$s0, 0x2110003d**. Câu lệnh này được thực hiện bởi hai câu lệnh:

```
lui $1, 0x00002110
ori $16, $1, 0x0000003d
```

Giải thích: Phần immediate đã vượt quá mức 16 bits nên phải tách thành hai câu lệnh như trên để ghi lần lượt 16 bits cao và 16 bits thấp vào thanh ghi \$s0. Trước tiên sử dụng lệnh **lui \$1, 0x00002110** để ghi 2110_{16} vào 16 bits cao của thanh ghi \$1, khi đó thanh ghi $\$1 = 0x21100000$; sau đó sử dụng lệnh **ori \$16, \$1, 0x0000003d** để OR thanh ghi \$16 với thanh ghi \$1, khi đó thanh ghi $\$16 = 0x2110003d$. Vậy kết quả ta gán được giá trị $0x2110003d$ vào thanh ghi \$s0

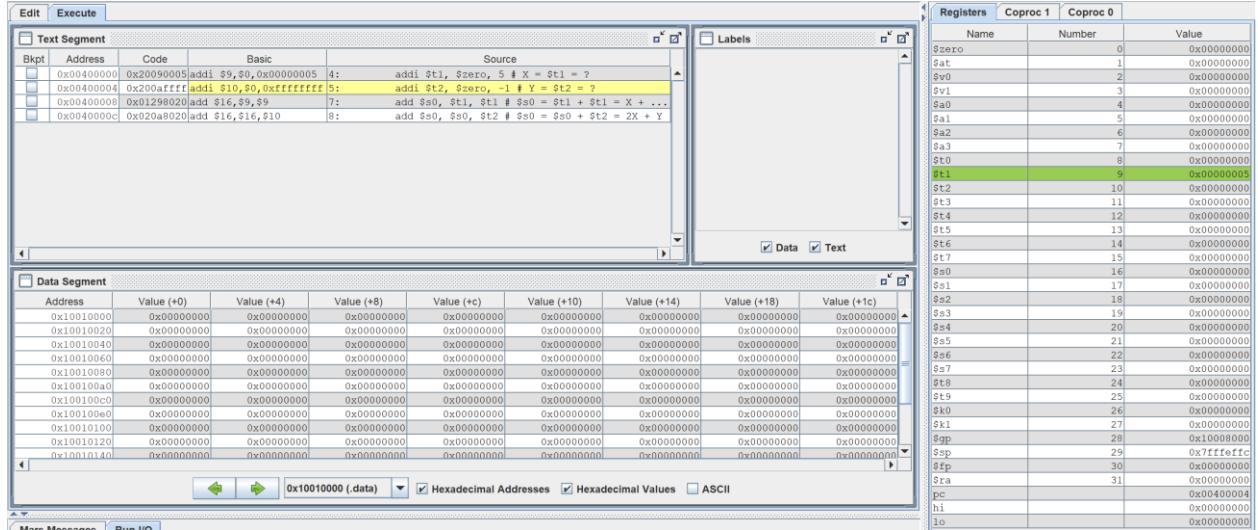
- Thực hiện lệnh **li \$s1, 0x2**. Câu lệnh này được thực hiện bởi một câu lệnh: **addi \$17, \$0, 0x00000002**

Giải thích: Phần immediate nằm trong mức cho phép nên câu lệnh trên chỉ cần thực hiện bằng một câu lệnh **addi \$17, \$0, 0x00000002**, khi đó ta được kết quả thanh ghi \$s1 được gán bằng 0x2

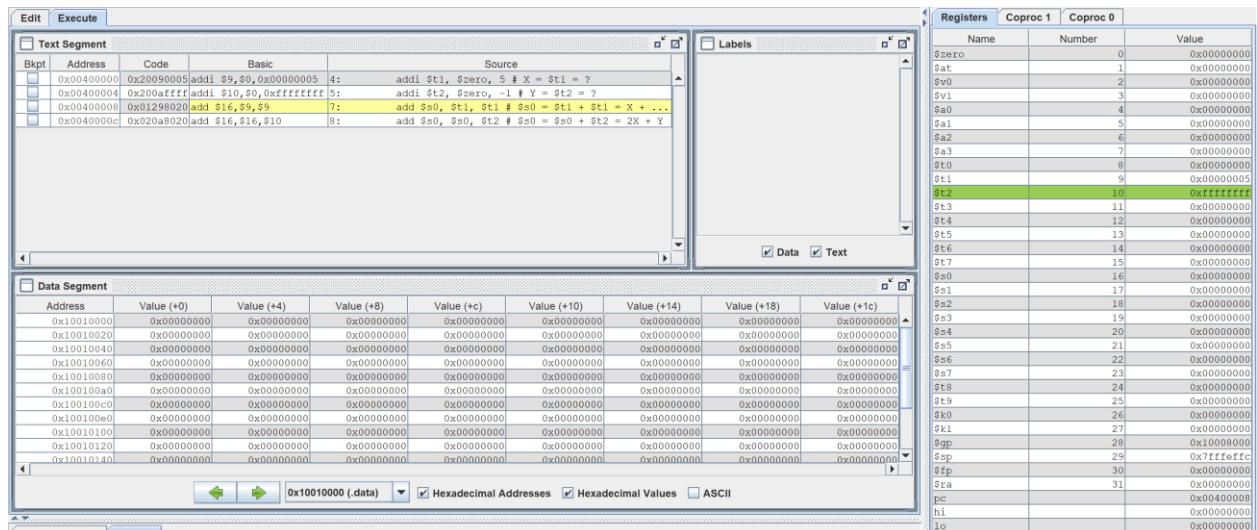
Assignment 4: Tính biểu thức $2x + y = ?$

1. Cửa sổ Register:

- Khi thực hiện lệnh **addi \$t1, \$zero, 5**. Thanh ghi $\$t1 = 0x00000005 = 5_{10}$



- Khi thực hiện lệnh **addi \$t2, \$zero, -1**. Thanh ghi $\$t2 = 0xffffffff = -1_{10}$



- Khi thực hiện lệnh **add \$s0, \$t1, \$t1**. Thanh ghi $\$s0 = 0x0000000a = 10_{10}$

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	0xffffffff
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$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
\$t0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7ffffefc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x0400000c
hi		0x00000000
lo		0x00000000

- Khi thực hiện lệnh **add \$s0, \$s0, \$t2**. Thanh ghi $\$s0 = 0x00000009 = 9_{10}$

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	0xffffffff
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$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
\$t0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7ffffefc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x04000010
hi		0x00000000
lo		0x00000000

=> Kết quả tính toán là đúng

2. Cửa sổ Text Segment:

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090005	addi \$9,\$0,0x00000005	4: addi \$t1, \$zero, 5 # X = \$t1 = ?
	0x00400004	0x200a0fff	addi \$10,\$0,0xffffffff	5: addi \$t2, \$zero, -1 # Y = \$t2 = ?
	0x00400008	0x01298020	add \$16,\$9,\$9	7: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = X + ...
	0x0040000c	0x020a08020	add \$16,\$16,\$10	8: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2X + Y

- **addi \$t1, \$zero, 5** là lệnh kiểu I

OP = 8 = 001000₂

\$rs = \$zero = 0₁₀ = 00000₂

\$rt = \$t1 = 9₁₀ = 01001₂

imm = 5 = 0000 0000 0000 0101₂

=> 001000 00000 01001 0000 0000 0000 0101₂ = 0010 0000 0000 1001 0000

0000 0000 0101₂ = 0x20090005

=> Đúng với mã lệnh trong cửa sổ Text Segment

- **addi \$t2, \$zero, -1** là lệnh kiểu I

OP = 8 = 001000₂

\$rs = \$zero = 0₁₀ = 00000₂

\$rt = \$t2 = 10₁₀ = 01010₂

imm = -1₁₀ = 1111 1111 1111 1111₂

=> 001000 00000 01010 1111 1111 1111 1111₂ = 0010 0000 0000 1010 1111

1111 1111 1111₂ = 0x200affff

=> Đúng với mã lệnh trong cửa sổ Text Segment

- **add \$s0, \$t1, \$t1** là lệnh kiểu R

OP = 0 = 00000₂

\$rs = \$9 = 9₁₀ = 01001₂

\$rt = \$9 = 9₁₀ = 01001₂

\$rd = \$16 = 16₁₀ = 10000₂

shamt = 00000₂

funct = 32₁₀ = 100000

=> 000000 01001 01001 10000 00000 100000₂ = 0000 0001 0010 1001 1000 0000

0010 0000₂ = 0x01298020

=> Đúng với mã lệnh trong cửa sổ Text Segment

- **add add \$s0, \$s0, \$t2** là lệnh kiểu R

OP = 0 = 00000₂

\$rs = \$16 = 16₁₀ = 10000₂

\$rt = \$10 = 10₁₀ = 01010₂

\$rd = \$16 = 16₁₀ = 10000₂

shamt = 00000₂

funct = 32₁₀ = 100000

=> 000000 10000 01010 10000 00000 100000₂ = 0000 0010 0000 1010 1000 0000

0010 0000₂ = 0x020a8020

=> Đúng với mã lệnh trong cửa sổ Text Segment

Assignment 5: Phép nhân

-Biên dịch và quan sát trong cửa sổ text:

Text Segment				
Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090004	addi \$9,\$0,4	4: addi \$t1, \$zero, 4 # X = \$t1 = ?
	0x00400004	0x200a0005	addi \$10,\$0,5	5: addi \$t2, \$zero, 5 # Y = \$t2 = ?
	0x00400008	0x712a8002	mul \$16,\$9,\$10	7: mul \$s0, \$t1, \$t2 # HI-LO = \$t1 * \$t2 = X * Y ; \$.
	0x0040000c	0x20010003	addi \$1,\$0,3	8: mul \$s0, \$s0, 3 # \$s0 = \$s0 * 3 = 3 * X * Y
	0x00400010	0x72018002	mul \$16,\$16,\$1	
	0x00400014	0x00008812	mflo \$17	10: mflo \$s1

Nhận xét: Điều bất thường nằm ở câu lệnh mul \$s0, \$s0, 3

Điều này là do không có lệnh nào hỗ trợ nhân trực tiếp hằng số vào giá trị thanh ghi

=> Câu lệnh này được chia làm 2 phần. Điều này là do giá trị 3 chưa được lưu vào thanh ghi nào cả nên máy sẽ thực hiện lệnh addi để lưu giá trị 3 vào thanh ghi \$1, rồi sau đó mới thực hiện lệnh mul để nhân.

-Chạy từng lệnh:

- Sau khi chạy dòng lệnh: addi \$t1, \$zero, 4

Text Segment				
Bkpt	Address	Code	Basic	Source
	0x04000000	0x20090004	addi \$9,\$0,0x00000004	4: addi \$t1, \$zero, 4 # X = \$t1 = 4
	0x04000004	0x200a0005	addi \$10,\$0,0x00000005	5: addi \$t2, \$zero, 5 # X = \$t2 = 5
	0x04000008	0x712a8002	mul \$16,\$9,\$10	7: mul \$s0, \$t1, \$t2 # HI-LO = \$t1 * \$t2 = X * Y ; \$.
	0x0400000c	0x20010003	addi \$1,\$0,0x00000003	8: mul \$s0, \$s0, 3 # \$s0 = \$s0 * 3 = 3 * X * Y
	0x04000010	0x72018002	mul \$16,\$16,\$1	
	0x04000014	0x00008812	mflo \$17	10: mflo \$s1

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$t0	7	0x00000000
\$t1	8	0x00000000
\$t2	9	0x00000000
\$s0	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$a4	20	0x00000000
\$a5	21	0x00000000
\$a6	22	0x00000000
\$t8	23	0x00000000
\$s8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffefc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400004
hi		0x00000000
lo		0x00000000

Ta thấy: Thanh ghi \$t1 = 0x00000004

Sau khi chạy dòng lệnh: addi \$t2, \$zero, 5

Registers	Coproc 1	Coproc 0
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$s0	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
\$a1	17	0x00000000
\$s2	18	0x00000000
\$a3	19	0x00000000
\$a4	20	0x00000000
\$a5	21	0x00000000
\$s6	22	0x00000000
\$a7	23	0x00000000
\$s8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7ffffefc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00000000
hi		0x00000000
lo		0x00000000

Ta thấy: Thanh ghi \$t2 = 0x00000005

- **Sau khi chạy dòng lệnh:** mul \$s0, \$t1, \$t2

Registers	Coproc 1	Coproc 0
\$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
\$a0	16	0x00000014
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$a4	20	0x00000000
\$a5	21	0x00000000
\$s6	22	0x00000000
\$a7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7ffffefc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00000000
hi		0x00000000
lo		0x00000014

Ta thấy: + Thanh ghi \$s0 = 0x00000014

+ Thanh ghi lo = 0x00000014

- Sau khi chạy dòng lệnh: $\text{mul } \$s0, \$s0, 3$

Registers	Coproc 1	Coproc 0
\$zero	0	0x00000000
\$at	1	0x00000003
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$s0	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	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	0x10000000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x04000014
hi		0x00000000
lo		0x0000003c

Ta thấy: + Đầu tiên thanh ghi \$1 được gán bằng 0x00000003

+ Sau lệnh nhân ta có $\$s0 = 0x0000003c$ và

$$\$lo = 0x0000003c$$

- Sau khi chạy dòng lệnh: $\text{mflo } \$s1$

Registers	Coproc 1	Coproc 0
\$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	0x10000000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x04000018
hi		0x00000000
lo		0x0000003c

Ta thấy: Giá trị của thanh ghi lo được ghi vào trong thanh ghi

$\$s1 = 0x0000003c$ hay giá trị bằng 60 theo hệ thập phân

=> Kết quả đúng

Assignment 6: Tạo biến và truy cập biến

- **Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment:**

Blkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,0x00001001	8: la \$t8, x # Get the address of X in Data Segment
	0x00400004	0x34380000	ori \$24,\$1,0x00000000	9: la \$t9, y # Get the address of Y in Data Segment
	0x00400008	0x3c011001	lui \$1,0x00001001	10: lw \$t1, 0(\$t8) # \$t1 = x
	0x0040000c	0x34390004	ori \$25,\$1,0x00000004	11: lw \$t2, 0(\$t9) # \$t2 = y
	0x00400010	0x8f090000	lw \$9,0x00000000(\$24)	12: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = x + x = 2x
	0x00400014	0x8f2a0000	lw \$10,0x00000000(\$25)	13: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2x + y
	0x00400018	0x01298020	add \$16,\$9,\$9	14: add \$s0, 0(\$s1) # z = \$s0 = 2x + y
	0x0040001c	0x020a8020	add \$16,\$16,\$10	15: la \$t7, z # Get the address of Z in Data Segment
	0x00400020	0x3c011001	lui \$1,0x00001001	16: sw \$s0, 0(\$t7) # z = \$s0 = 2x + y
	0x00400024	0x342f0008	ori \$15,\$1,0x00000008	
	0x00400028	0x腺df0000	sw \$16,0x00000000(\$15)	

Ta thấy: Lệnh la được biên dịch bằng hai lệnh lui và ori. Vì địa chỉ của X, Y, Z được khai báo với chỉ thị là .word nên địa chỉ của X, Y, Z là số 32 bits nên cần thực hiện bằng việc thực hiện hai câu lệnh lui và ori

- **Ở cửa sổ Label và quan sát địa chỉ của X, Y, Z:**

Blkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,0x00001001	8: la \$t8, x # Get the address of X in Data Segment
	0x00400004	0x34380000	ori \$24,\$1,0x00000000	9: la \$t9, y # Get the address of Y in Data Segment
	0x00400008	0x3c011001	lui \$1,0x00001001	10: lw \$t1, 0(\$t8) # \$t1 = x
	0x0040000c	0x34390004	ori \$25,\$1,0x00000004	11: lw \$t2, 0(\$t9) # \$t2 = y
	0x00400010	0x8f090000	lw \$9,0x00000000(\$24)	12: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = 2x
	0x00400014	0x8f2a0000	lw \$10,0x00000000(\$25)	13: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2x + y
	0x00400018	0x01298020	add \$16,\$9,\$9	14: add \$s0, 0(\$s1) # z = \$s0 = 2x + y
	0x0040001c	0x020a8020	add \$16,\$16,\$10	15: la \$t7, z # Get the address of Z in Data Segment
	0x00400020	0x3c011001	lui \$1,0x00001001	16: sw \$s0, 0(\$t7) # z = \$s0 = 2x + y
	0x00400024	0x342f0008	ori \$15,\$1,0x00000008	
	0x00400028	0x腺df0000	sw \$16,0x00000000(\$15)	

Label	Address
X	0x10010000
Y	0x10010004
Z	0x10010008

Ta thấy: Địa chỉ của X, Y, Z được chia làm hai phần: 2 bytes cao và 2 bytes thấp. Tương ứng với mỗi lệnh la thì lệnh lui sẽ ghi 2 bytes cao của địa chỉ vào thanh ghi \$1 và lệnh ori sẽ OR thanh ghi \$1 với 2 bytes thấp của địa chỉ.

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,0x00001001	8: la \$t8, x # Get the add..
	0x00400004	0x34380000	ori \$24,\$1,0x00000000	9: la \$t9, y # Get the add..
	0x00400008	0x3c011001	lui \$1,0x00001001	10: lw \$t1, 0(\$t8) # \$t1 = x
	0x0040000c	0x34390004	ori \$25,\$1,0x00000004	11: lw \$t2, 0(\$t9) # \$t2 = y
	0x00400010	0x8f090000	lw \$9,0x00000000(\$24)	12: add \$s0, \$t1, \$t1 # \$s0 = \$t1 +..
	0x00400014	0x8f2a0000	lw \$10,0x00000000(\$25)	13: add \$s0, \$s0, \$t2 # \$s0 = \$s0 +..
	0x00400018	0x01298020	add \$16,\$9,\$9	14: add \$s0, \$s0, \$t2 # \$s0 = \$s0 +..
	0x0040001c	0x020a8020	add \$16,\$16,\$10	15: add \$s0, \$s0, \$t2 # \$s0 = \$s0 +..
	0x00400020	0x3c011001	lui \$1,0x00001001	16: la \$t7, z # Get the add..
	0x00400024	0x342f0008	ori \$15,\$1,0x00000008	17: sw \$s0, 0(\$t7) # z = \$s0 = 2..
	0x00400028	0xadf00000	sw \$16,0x00000000(\$15)	18: sw \$s0, 0(\$t7) # z = \$s0 = 2..
				19: sw \$s0, 0(\$t7) # z = \$s0 = 2..

Labels

Label	Address
X	0x10010000
	0x10010004
Z	0x10010008

Data Text

Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000005 0xffffffff 0x00000009 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010160 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010180 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui \$1,0x00001001	8: la \$t8, x # Get the add..
	0x00400004	0x34380000	ori \$24,\$1,0x00000000	9: la \$t9, y # Get the add..
	0x00400008	0x3c011001	lui \$1,0x00001001	10: lw \$t1, 0(\$t8) # \$t1 = x
	0x0040000c	0x34390004	ori \$25,\$1,0x00000004	11: lw \$t2, 0(\$t9) # \$t2 = y
	0x00400010	0x8f090000	lw \$9,0x00000000(\$24)	12: add \$s0, \$t1, \$t1 # \$s0 = \$t1 +..
	0x00400014	0x8f2a0000	lw \$10,0x00000000(\$25)	13: add \$s0, \$s0, \$t2 # \$s0 = \$s0 +..
	0x00400018	0x01298020	add \$16,\$9,\$9	14: add \$s0, \$s0, \$t2 # \$s0 = \$s0 +..
	0x0040001c	0x020a8020	add \$16,\$16,\$10	15: add \$s0, \$s0, \$t2 # \$s0 = \$s0 +..
	0x00400020	0x3c011001	lui \$1,0x00001001	16: la \$t7, z # Get the add..
	0x00400024	0x342f0008	ori \$15,\$1,0x00000008	17: sw \$s0, 0(\$t7) # z = \$s0 = 2..
	0x00400028	0xadf00000	sw \$16,0x00000000(\$15)	18: sw \$s0, 0(\$t7) # z = \$s0 = 2..
				19: sw \$s0, 0(\$t7) # z = \$s0 = 2..

Labels

Label	Address
X	0x10010000
	0x10010004
Y	0x10010008

Data Text

Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 0x00000005 0xffffffff 0x00000009 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010020 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010060 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010120 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010140 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010160 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x10010180 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101a0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101c0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000
0x100101e0 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000 0x00000000

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

- **Sau khi chạy lần lượt các dòng lệnh:**

Ta thấy: + Sau khi lấy địa chỉ của các biến X, Y, Z và ghi địa chỉ đó vào các thanh ghi \$t8, \$t9, \$t7 bằng lệnh la

la \$t8, X

la \$t9, Y

+ Tiếp tục lấy ra giá trị 32 bit (word) nằm trong các ô nhớ có địa chỉ vừa ghi được vào thanh ghi \$t8, \$t9 bằng lệnh lw vào các thanh ghi \$t1, \$t2

lw \$t1, 0(\$t8)

lw \$t2, 0(\$t9)

+ Sau khi thực hiện các câu lệnh add, kết quả đang được ghi trong thanh ghi \$s0. Để ghi giá trị 32 bit (word) trong thanh ghi \$s0 vào ô nhớ có địa chỉ được ghi trong thanh ghi \$t7 thì ta sử dụng lệnh sw

sw \$s0, 0(\$t7)

- Lệnh lb, sb

+ lb - Để lấy ra dữ liệu kiểu byte (8 bit) tại ô nhớ thông qua địa chỉ trỏ đến ô nhớ đó ra thanh ghi

+ sb - Để ghi dữ liệu kiểu byte (8 bit, 8 bit này được lưu vào 8 bit thấp của ô nhớ) vào ô nhớ thông qua địa chỉ trỏ đến ô nhớ đó