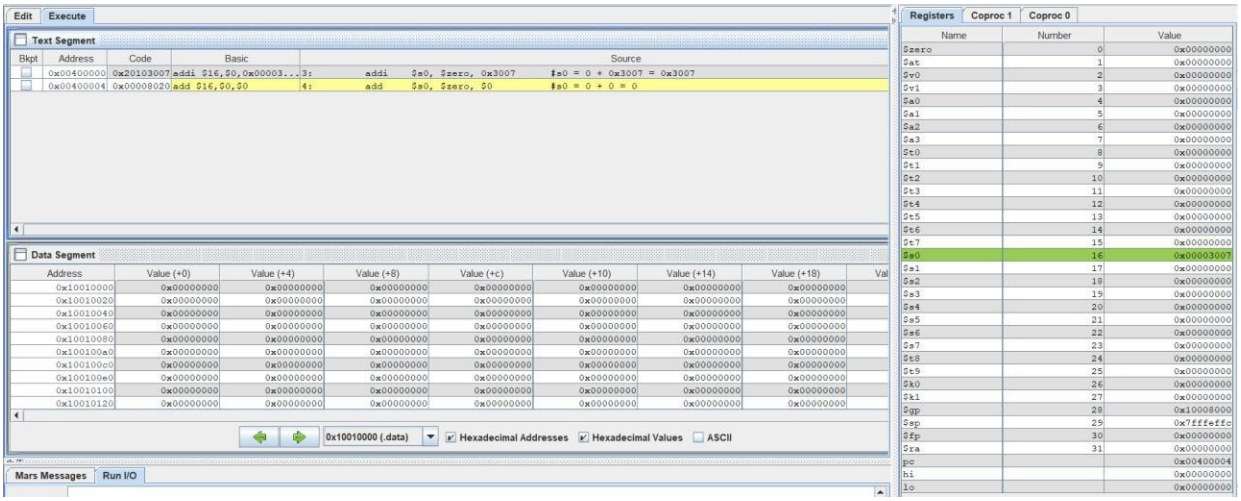
# Bài thực hành Kiến trúc máy tính tuần 2

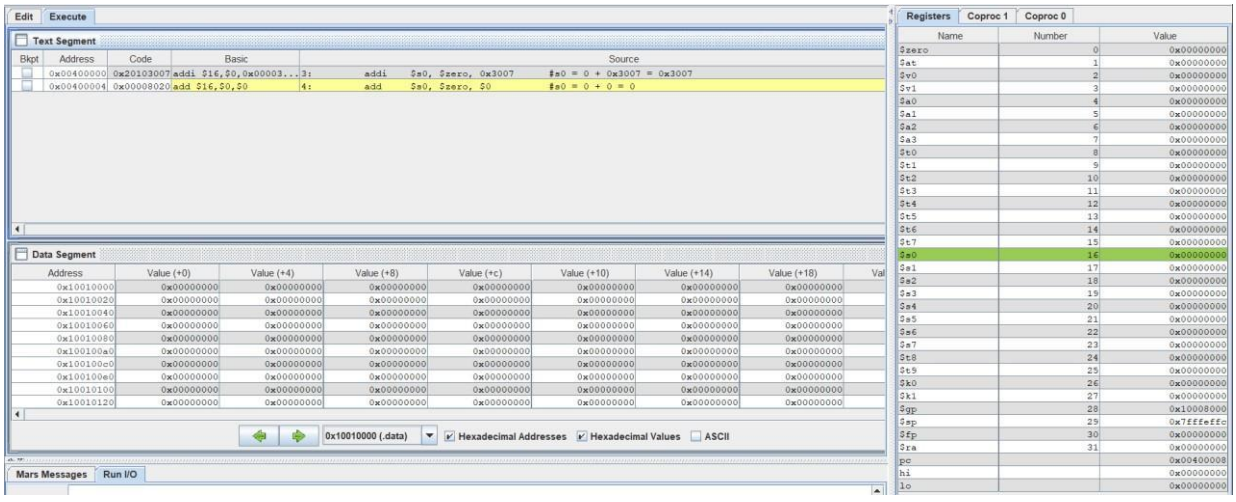
## Bài 1:

```
1  #Laboratory Exercise 2, Assignment 1
2  .text
3          addi    $s0, $zero, 0x3007      #s0 = 0 + 0x3007 = 0x3007
4          add     $s0, $zero, $0          #s0 = 0 + 0 = 0
```

- Thực hiện lệnh dòng 3: thanh ghi $s0 được nạp giá trị là tổng của giá trị ở thanh ghi $zero (giá trị là 0) và 0x3007, khi đó $s0 có nội dung là giá trị 0x00003007.



- Thực hiện lệnh dòng 4: thanh ghi $s0 được nạp giá trị là tổng của hai giá trị ở thanh ghi $zero và $0 (cùng có giá trị là 0), khi đó $s0 có nội dung là giá trị 0.



- Giá trị pc khi thực hiện mỗi lệnh đều tăng lên 4.
- Lệnh ở dòng 3: là lệnh kiểu I:

| op | rs ($zero) | rt ($s0) | imm |
|---|---|---|---|
| 001000 | 00000 | 10000 | 0011 0000 0000 0111 |

Mã máy:

0010 0000 0001 0000 0011 0000 0000 0111 = 0x20103007

- Lệnh ở dòng 4: là lệnh kiểu R:

| op | rs ($zero) | rt ($0) | rd ($s0) | shamt | funct |
|---|---|---|---|---|---|
| 000000 | 00000 | 00000 | 10000 | 00000 | 100000 |

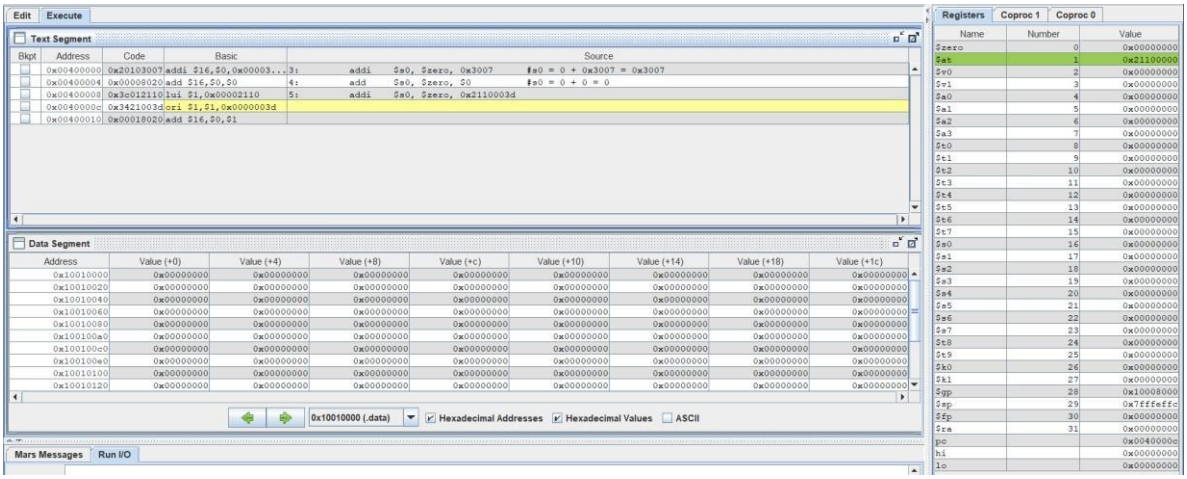
Mã máy:

0000 0000 0000 0000 1000 0000 0010 0000 = 0x00008020

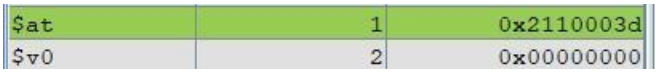
- Khi thực hiện lệnh `addi $s0, $zero, 0x2110003d`: chương trình sẽ tự động tách thành 3 lệnh (do hằng số imm ở trên có độ dài lớn hơn 16 bit):

```
lui $at, 0x00002110

ori $at, 0x0000003d

add $s0, $0, Sat
```

Lệnh lui: $at sẽ được nạp 4 chữ số cuối của số 0x00002110 vào 4 chữ số cao nhất của $at. Khi đó $at có giá trị là 0x21100000.



Lệnh ori: $at sẽ được nạp 4 chữ số cuối của số 0x0000003d vào 4 chữ số thấp nhất của $at. Khi đó $at có giá trị là 0x2110003d.



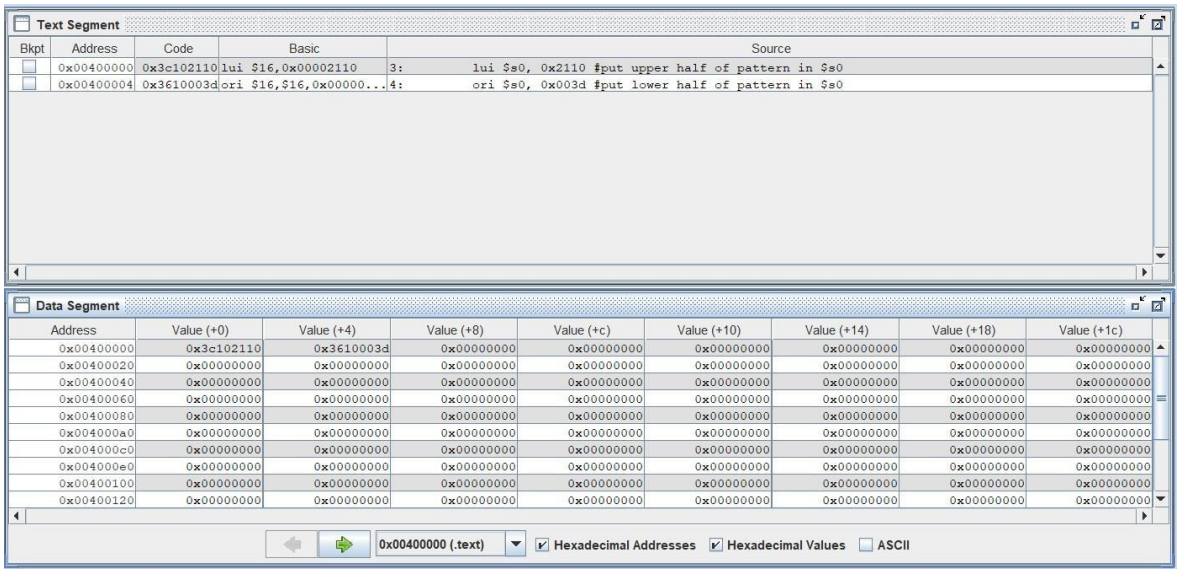

Lệnh add: $s0 được nạp giá trị là tổng của hai giá trị ở hai thanh ghi $s0, $0, $at. Khi đó $s0 có giá trị là 0x2110003d, đúng như ta mong muốn.

| | | |
|---|---|---|
| $t7 | 15 | 0x00000000 |
| $s0 | 16 | 0x2110003d |
| $s1 | 17 | 0x00000000 |

**Bài 2:**

```
1  #Laboratory Exercise 2, Assignment 2
2  .text
3          lui $s0, 0x2110 #put upper half of pattern in $s0
4          ori $s0, 0x003d #put lower half of pattern in $s0
```

- Tương tự như bài 1, lệnh lui làm cho $s0 được giá trị là 0x21100000, còn lệnh ori làm cho $s0 được nạp giá trị 0x2110003d.
- Ở Data Segment thì các byte đầu tiên ở vùng lệnh trùng với cột Code của Text Segment (đó là mã máy).



Text Segment

| Bkpt | Address | Code | Basic | Source |
|---|---|---|---|---|
| | 0x00400000 | 0x3c102110 | lui $16,0x00002110 | 3: lui $s0, 0x2110 #put upper half of pattern in $s0 |
| | 0x00400004 | 0x3610003d | ori $16,$16,0x00000... | 4: ori $s0, 0x003d #put lower half of pattern in $s0 |

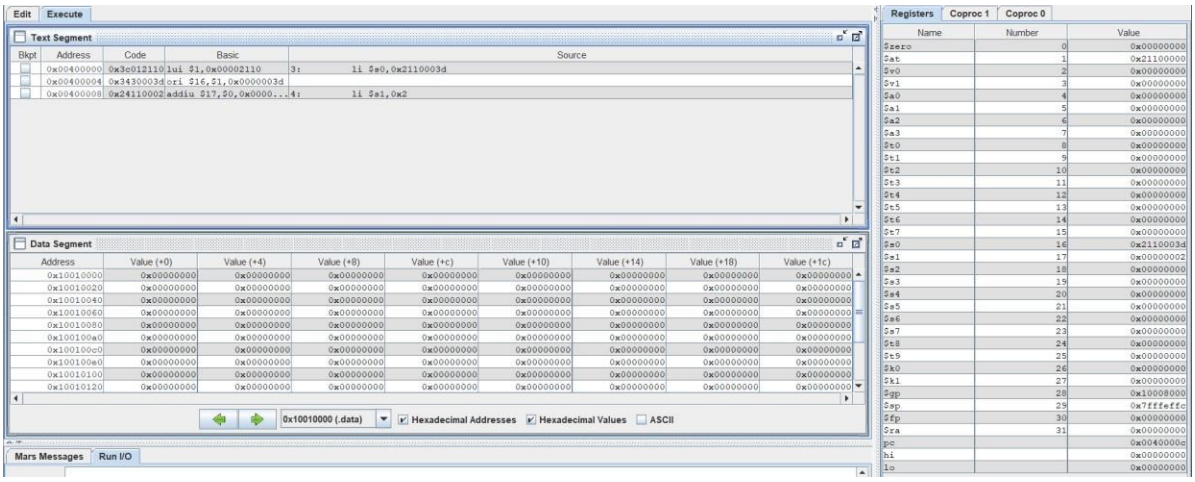
Data Segment

| 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 |
| 0x00400060 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x00400080 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x004000a0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x004000c0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x004000e0 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x00400100 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 0x00400120 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |

0x00400000 (.text) Hexadecimal Addresses Hexadecimal Values ASCII

**Bài 3:**

```
1  #Laboratory Exercise 2, Assignment 3
2  .text
3          li $s0,0x2110003d
4          li $s1,0x2
```




Điều bất thường ở đây là hai lệnh `li` khi được quy đổi ra lệnh cơ bản thì chúng có quy đổi lệnh không giống nhau. Lệnh `li` có chức năng nạp giá trị vào thanh ghi.

Lệnh `li` thứ 1: vì hằng số trong lệnh có độ dài lớn hơn 16 bit nên chương trình sẽ tách lệnh thành 2 lệnh lui và ori.

Lệnh `li` thứ 2: vì hằng số trong lệnh có độ dài nhỏ hơn 16 bit nên chương trình quy đổi thành lệnh addiu để cộng đơn giản mà thôi.

**Bài 4:**

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

- Thực hiện lệnh dòng 4: $t1 được nạp giá trị là tổng của giá trị ở thanh ghi $zero và 5. Khi đó $t1 = 5.

| | | |
|---|---|---|
| $t0 | 8 | 0x00000000 |
| $t1 | 9 | 0x00000005 |
| $t2 | 10 | 0x00000000 |

- Thực hiện lệnh dòng 5: $t2 được nạp giá trị là tổng của giá trị ở thanh ghi $zero và −1. Khi đó $t2 = −1, giá trị được nạp vào là số nguyên có dấu 32 bit, nên nó là 0xffffffff.

| | | |
|---|---|---|
| $t1 | 9 | 0x00000005 |
| $t2 | 10 | 0xffffffff |
| $t3 | 11 | 0x00000000 |

•

Thực hiện lệnh dòng 7: $s0 được nạp giá trị là tổng của giá trị ở thanh ghi $s1 và $s1. Khi đó $s0 = 5 + 5 = 10, dưới dạng hexa là 0x0000000a.

| | | |
|---|---|---|
| $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 |

- Thực hiện lệnh dòng 8: $s0 được nạp giá trị là tổng của giá trị ở thanh ghi $s0 và $t2. Khi đó $s0 = 10 + (−1) = 9.

| | | |
|---|---|---|
| $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 |

- Lệnh addi:

Dòng 4:

| op | rs ($zero) | rt ($t1) | imm |
|---|---|---|---|
| 001000 | 00000 | 01001 | 0000 0000 0000 0101 |

Mã máy:

0010 0000 0000 1001 0000 0000 0000 0101 = 0x20090005

Dòng 5:

| op | rs ($zero) | rt ($t2) | imm |
|---|---|---|---|
| 001000 | 00000 | 01010 | 1111 1111 1111 1111 |

Mã máy:

0010 0000 0000 1010 1111 1111 1111 1111 = 0x200affff

•

| Bkpt | Address | Code | Basic | | | Source |
|---|---|---|---|---|---|---|
| | 0x00400000 | 0x20090005 | addi $9,$0,0x00000005 | 4: | addi | $t1, $zero, 5 # X = $t1 = ? |
| | 0x00400004 | 0x200affff | addi $10,$0,0xffffff... | 5: | addi | $t2, $zero, -1 # Y = $t2 = ? |
| | 0x00400008 | 0x01298020 | add $16,$9,$9 | 7: | add | $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X |
| | 0x0040000c | 0x020a8020 | add $16,$16,$10 | 8: | add | $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y |

Lệnh add:

Dòng 7:

| op | rs ($t1) | rt ($t1) | rd ($s0) | shamt | funct |
|---|---|---|---|---|---|
| 000000 | 01001 | 01001 | 10000 | 00000 | 100000 |

Mã máy:

0000 0001 0010 1001 1000 0000 0010 0000 = 0x01298020

Dòng 8:

| op | rs ($s0) | rt ($t2) | rd ($s0) | shamt | funct |
|---|---|---|---|---|---|
| 000000 | 10000 | 01010 | 10000 | 00000 | 100000 |

Mã máy:

0000 0010 0000 1010 1000 0000 0010 0000 = 0x020a8020


| Bkpt | Address | Code | Basic | | | Source |
|---|---|---|---|---|---|---|
| | 0x00400000 | 0x20090005 | addi $9,$0,0x00000005 | 4: | addi | $t1, $zero, 5 # X = $t1 = ? |
| | 0x00400004 | 0x200affff | addi $10,$0,0xffffff... | 5: | addi | $t2, $zero, -1 # Y = $t2 = ? |
| | 0x00400008 | 0x01298020 | add $16,$9,$9 | 7: | add | $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X |
| | 0x0040000c | 0x020a8020 | add $16,$16,$10 | 8: | add | $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y |


**Bài 5:**

```
1  #Laboratory Exercise 2, Assignment 5
2  .text
3   # Assign X, Y
4         addi    $t1, $zero, 4 # X = $t1 = ?
5         addi    $t2, $zero, 5 # Y = $t2 = ?
6   # Expression Z = 3*XY
7         mul     $s0, $t1, $t2 # HI-LO = $t1 * $t2 = X * Y ; $s0 = LO
8         mul     $s0, $s0, 3   # $s0 = $s0 * 3 = 3 * X * Y
9   # Z' = Z
10        mflo    $s1
```

| Bkpt | Address | Code | Basic | | | Source |
|---|---|---|---|---|---|---|
| | 0x00400000 | 0x20090004 | addi $9,$0,0x00000004 | 4: | addi | $t1, $zero, 4 # X = $t1 = ? |
| | 0x00400004 | 0x200a0005 | addi $10,$0,0x00000... | 5: | addi | $t2, $zero, 5 # Y = $t2 = ? |
| | 0x00400008 | 0x712a8002 | mul $16,$9,$10 | 7: | mul | $s0, $t1, $t2 # HI-LO = $t1 * $t2 = X * Y ; $s0 = LO |
| | 0x0040000c | 0x20010003 | addi $1,$0,0x00000003 | 8: | mul | $s0, $s0, 3 # $s0 = $s0 * 3 = 3 * X * Y |
| | 0x00400010 | 0x72018002 | mul $16,$16,$1 | | | |
| | 0x00400014 | 0x00008812 | mflo $17 | 10: | mflo | $s1 |

- Điều bất thường xuất hiện ở đây là lệnh mul có 2 thanh ghi sẽ được chương trình tách thành hai lệnh cơ bản (trong đó có lệnh mul 3 thanh ghi), tức là lệnh mul 3 thanh ghi là lệnh cơ bản còn mul 2 thanh ghi thì không.

- Ở dòng 4, $t1 được nạp giá trị 4. Còn ở dòng 5, $t2 được nạp giá trị 5.

  Ở dòng 7, $s0 được nạp giá trị là tích của hai giá trị ở hai thanh ghi $t1 và $t2, khi đó $s0 = 4 * 5 = 20, ở hexa là 0x14. Đồng thời giá trị ở thanh ghi `lo` cũng là 14.

| | | |
|---|---|---|
| $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 | 0x7fffeffc |
| $fp | 30 | 0x00000000 |
| $ra | 31 | 0x00000000 |
| pc | | 0x0040000c |
| hi | | 0x00000000 |
| lo | | 0x00000014 |

- Ở dòng 8, lệnh mul được tách thành hai lệnh:

```
addi $at, $0, 0x00000003

mul $s0, $s0, $at
```

$at sẽ được nạp giá trị là 3, rồi sau đó $s0 sẽ được nạp giá trị là tích của giá trị ở hai thanh ghi $s0 và $at, tức là 20 * 3 = 60, ở hexa là 0x3c.

- Nhận thấy giá trị ở thanh ghi `hi` không đổi, vì các tích trong chương trình có kết quả chưa lớn để có thể phải dùng `hi` để nạp giá trị.

| 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 | 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 | 0x7fffeffc |
| $fp | 30 | 0x00000000 |
| $ra | 31 | 0x00000000 |
| pc | | 0x00400014 |
| hi | | 0x00000000 |
| lo | | 0x0000003c |

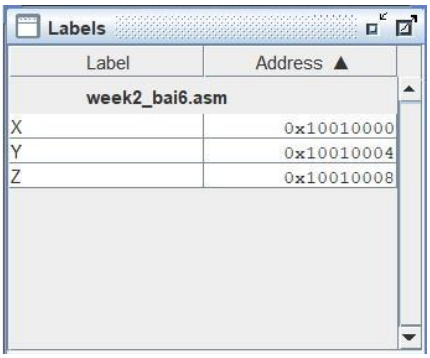
## Bài 6:

```
1  #Laboratory Exercise 2, Assignment 5
2  .data # DECLARE VARIABLES
3          X : .word 5 # Variable X, word type, init value =
4          Y : .word -1 # Variable Y, word type, init value =
5          Z : .word # Variable Z, word type, no init value
6  .text # DECLARE INSTRUCTIONS
7   # Load X, Y to registers
8          la      $t8, X # Get the address of X in Data Segment
9          la      $t9, Y # Get the address of Y in Data Segment
10         lw      $t1, 0($t8) # $t1 = X
11         lw      $t2, 0($t9) # $t2 = Y
12  # Calcuate the expression Z = 2X + Y with registers only
13         add     $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
14         add     $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
15  # Store result from register to variable Z
16         la      $t7, Z # Get the address of Z in Data Segment
17         sw      $s0, 0($t7) # Z = $s0 = 2X + Y
18
```

- Các lệnh **la** đều đã được chuyển thành hai lệnh là lui và ori. Còn việc thực hiện lệnh lui và ori thì tương tự như bài 1.

| Bkpt | 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 | | |
| | 0x00400008 | 0x3c011001 | lui $1,0x00001001 | 9: | la $t9, Y # Get the address of Y in Data Segment |
| | 0x0040000c | 0x34390004 | ori $25,$1,0x00000004 | | |
| | 0x00400010 | 0x8f090000 | lw $9,0x00000000($24) | 10: | lw $t1, 0($t8) # $t1 = X |
| | 0x00400014 | 0x8f2a0000 | lw $10,0x00000000($... | 11: | lw $t2, 0($t9) # $t2 = Y |
| | 0x00400018 | 0x01298020 | add $16,$9,$9 | 13: | add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X |
| | 0x0040001c | 0x020a8020 | add $16,$16,$10 | 14: | add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y |
| | 0x00400020 | 0x3c011001 | lui $1,0x00001001 | 16: | la $t7, Z # Get the address of Z in Data Segment |
| | 0x00400024 | 0x342f0008 | ori $15,$1,0x00000008 | | |
| | 0x00400028 | 0xadf00000 | sw $16,0x00000000($... | 17: | sw $s0, 0($t7) # Z = $s0 = 2X + Y |

- Trong mỗi địa chỉ của X, Y, Z, 4 chữ số đầu giống với hằng số ở lệnh lui, 4 chữ số cuối giống với hằng số của lệnh ori.




- Ở dòng lệnh 8 và 9, $t8 và $t9 sẽ được nạp giá trị là địa chỉ của X và Y. Tức là $t8 = 0x10010000 và $t9 = 0x10010004.
- Thanh ghi $at được nạp giá trị 0x10010000 nhờ lệnh lui.
- Ở dòng 10 và 11, lệnh lw làm cho $t1 và $t2 được nạp giá trị là giá trị của X và Y (độ dài là 1 word), vì thanh ghi cơ sở lần lượt là $t8 và $t9 (chứa giá trị là địa chỉ của X và Y), hằng số dịch chuyển đều bằng 0. Do đó $t1 = 0x00000005 (5) và $t2 = 0xffffffff (–1).

Registers | Coproc 1 | Coproc 0

| Name | Number | Value |
|---|---|---|
| $zero | 0 | 0x00000000 |
| $at | 1 | 0x10010000 |
| $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 | 0x00000000 |
| $s1 | 17 | 0x00000000 |
| $s2 | 18 | 0x00000000 |
| $s3 | 19 | 0x00000000 |
| $s4 | 20 | 0x00000000 |
| $s5 | 21 | 0x00000000 |
| $s6 | 22 | 0x00000000 |
| $s7 | 23 | 0x00000000 |
| $t8 | 24 | 0x10010000 |
| $t9 | 25 | 0x10010004 |
| $k0 | 26 | 0x00000000 |
| $k1 | 27 | 0x00000000 |
| $gp | 28 | 0x10008000 |
| $sp | 29 | 0x7fffeffc |
| $fp | 30 | 0x00000000 |
| $ra | 31 | 0x00000000 |
| pc | | 0x00400018 |
| hi | | 0x00000000 |
| lo | | 0x00000000 |

- Ở dòng 13, $s0 = $t1 + $1 = X + X = 5 + 5 = 10. Do đó $s0 = 0x0000000a.

| | | |
|---|---|---|
| $s0 | 16 | 0x0000000a |

- Ở dòng 14, $s0 = $s0 + $t2 = 10 + (–1) = 9. Do đó $s0 = 0x00000009.

| | | |
|---|---|---|
| $s0 | 16 | 0x00000009 |

- Ở dòng 16, lệnh **la** làm cho $t7 được nạp giá trị là địa chỉ của Z. Do đó $t7 = 0x10010008.

| | | |
|---|---|---|
| $t7 | 15 | 0x10010008 |

- Ở dòng 17, lệnh sw làm cho giá trị của $s0 (có độ dài 1 word) được lưu ra bộ nhớ tại ngăn nhớ có địa chỉ là giá trị ở thanh ghi $t7 (vì địa chỉ cơ sở là $t7 và hằng số dịch chuyển là 0). Do đó tại ngăn nhớ có địa chỉ 0x10000008 thì giá trị ở đó là 9.

Data Segment

| 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 |

- Như vậy, lệnh lw dùng để nạp giá trị có độ dài 1 word vào 1 thanh ghi từ thanh ghi chứa giá trị là địa chỉ cơ sở và hằng số dịch chuyển. Còn lệnh sw dùng để lưu giá trị có độ dài 1 word từ 1 thanh ghi ra ngăn nhớ có địa chỉ mà địa chỉ đó được xác định bởi một thanh ghi chứa địa chỉ cơ sở và hằng số dịch chuyển.

- Vai trò của lệnh lb và sb tương tự như lw và sw, chỉ khác ở chỗ thay vì nạp/ lưu 1 word thì là nạp/ lưu 1 byte.