

# LAB 2

Nguyễn Khánh Nam - 20225749

## Assignment 1:

Code\_1:

#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
```

Code\_2:

#Laboratory Exercise 2, Assignment 1

.text

```
addi $s0, $zero, 0x2110003d
add $s0, $zero, $0         # $s0 = 0 + 0 = 0 ;    R-type
```

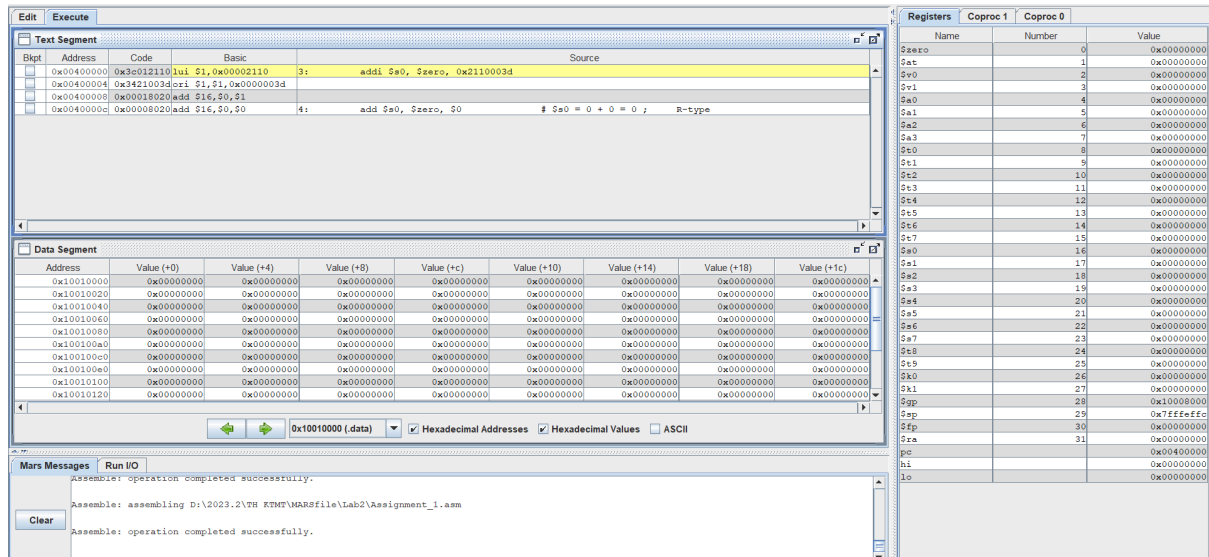
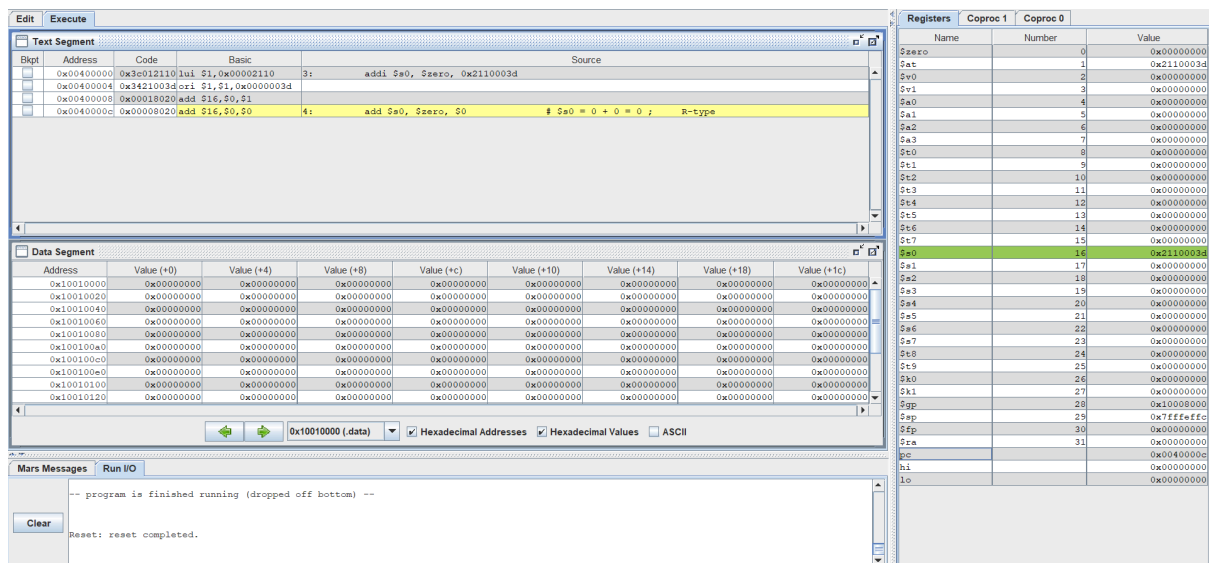
Result:

The screenshot shows the MARS 4.5 MIPS assembler simulator. The main window displays the assembly code for Code\_1 and Code\_2. The Text Segment window shows the assembly code and its machine code. The Data Segment window shows memory addresses and values. The Registers window shows the state of MIPS registers. The MARS Messages window shows the successful completion of the assembly process.

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	0x00000000
\$t2	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
\$a7	23	0x00000000
\$a8	24	0x00000000
\$a9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0xfffffff0
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000



- Theo trình tự chạy chương trình, thanh ghi \$s0 có sự thay đổi giá trị từ chứa 0 thành 12295 (0x3007 số 16 bits) rồi quay về 0.
  - + Thanh ghi pc chứa địa chỉ của lệnh tiếp theo trong quá trình chạy nên lần lượt cộng thêm 4 bytes.

- Sau khi thay đổi dòng lệnh addi thành addi \$s0, \$zero, 0x2110003d : chương trình thực hiện cộng với số 32 bits mà addi chỉ thực hiện được với số 16 bits nên đã có thêm 2 lệnh lui và ori để chia số 32 bits thành 2 phần mỗi phần 16 bits.

## Assignment 2:

Code:

#Lab Ex 2, Assignment 2

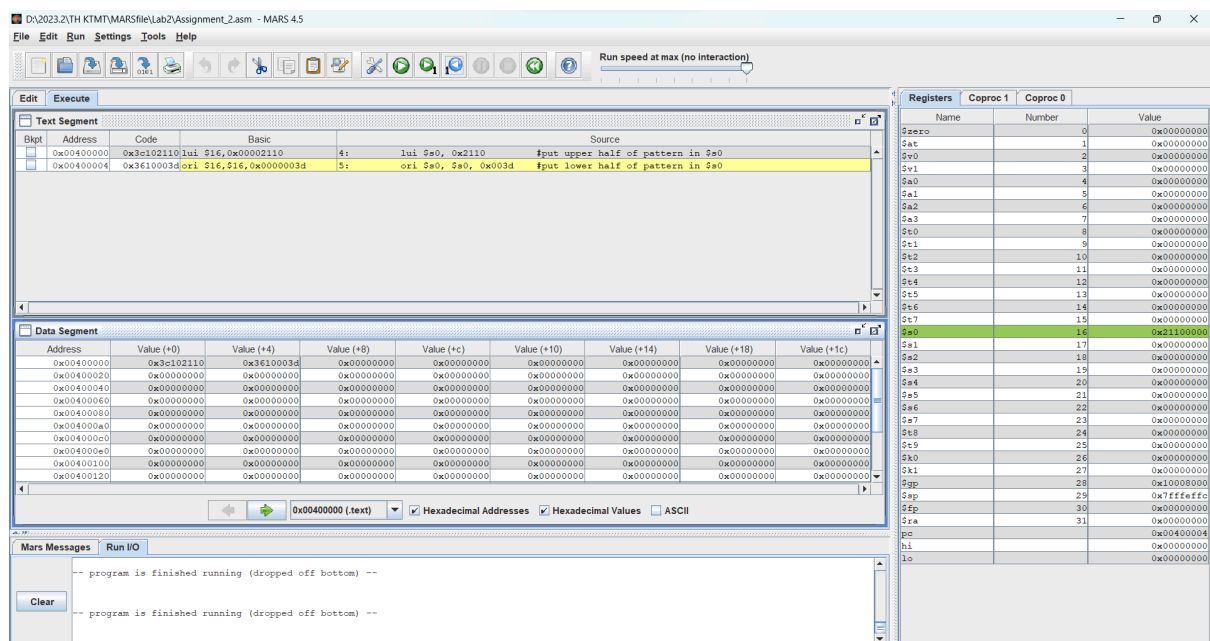
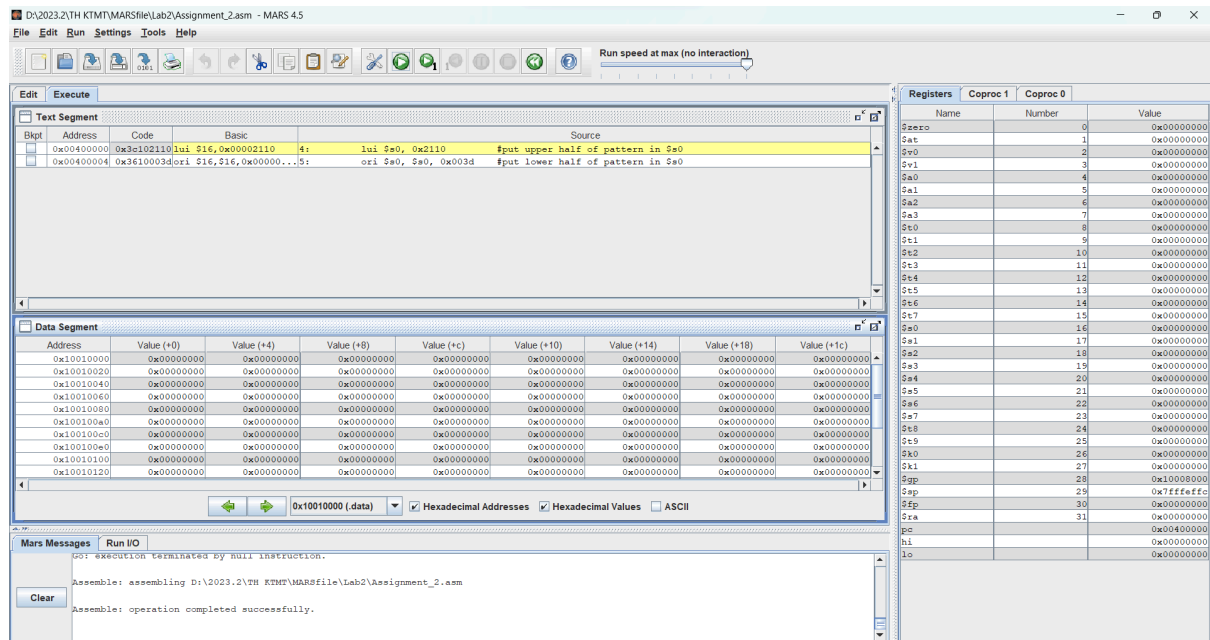
.text

lui \$s0, 0x2110

#put upper half of pattern in \$s0

ori \$s0, \$s0, 0x003d      #put lower half of pattern in \$s0

Result:



- Các byte đầu tiên trong vùng lệnh Data segment trùng với các byte trên cột CODE của vùng lệnh Text Segment.

## Assignment 3:

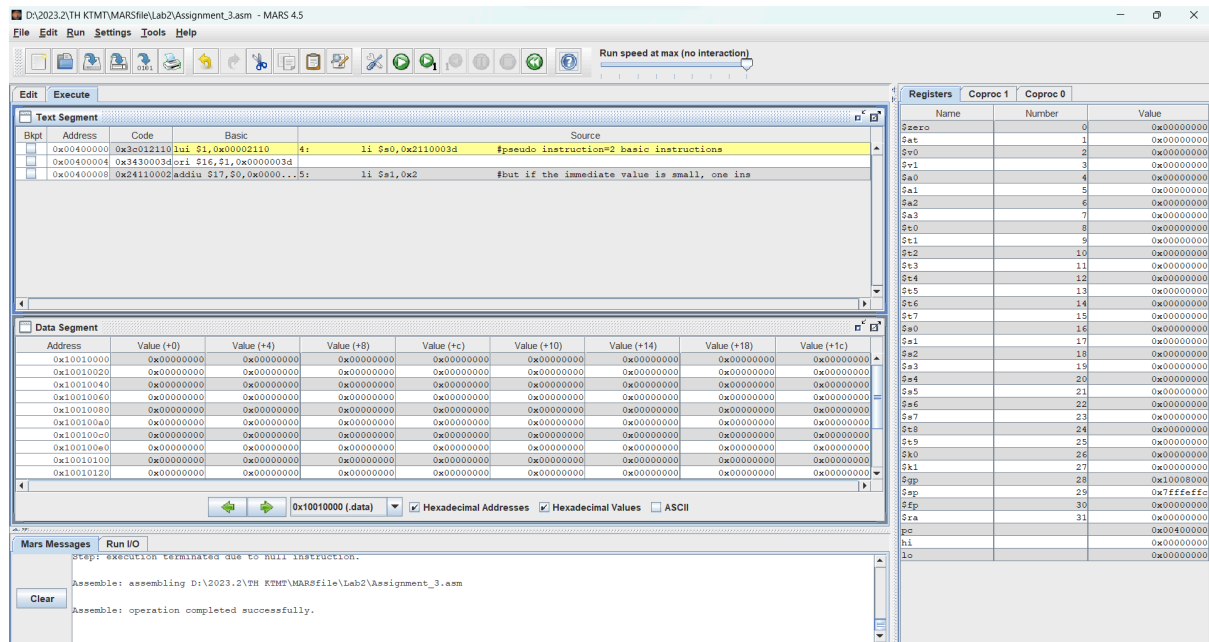
Code:

#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
```

Result:



- Lệnh li (load immediate) chuyển thành 2 lệnh lui và ori với số 32 bits do li chỉ thực hiện với số 16 bits nên phải tách số 32 bits thành 2 phần, chuyển thành addiu (addition immediate unsigned) với số 16 bits.

## Assignment 4:

Code:

#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
```

Result:

D:\2023.2\TH KTM\T\MARSfile\Lab2\Assignment\_4.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090005	addi \$9,\$0,0x00000005	6: addi \$t1, \$zero, 5 # X = \$t1 = ?
	0x00400004	0x200affff	addi \$10,\$0,0xfffffff	7: addi \$t2, \$zero, -1 # Y = \$t2 = ?
	0x00400008	0x01298020	add \$16,\$9,\$9	10: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = X + X = 2X
	0x0040000c	0x020a8020	add \$16,\$16,\$10	11: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2X + Y

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	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

Mars Messages Run I/O

Assembly: assembling D:\2023.2\TH KTM\T\MARSfile\Lab2\Assignment\_4.asm

Assembly: operation completed successfully.

Registers Coproc 1 Coproc 0

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
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$fp	29	0x7fffffc0
\$ra	30	0x00000000
\$pc	31	0x00400000
\$hi		0x00000000
\$lo		0x00000000

D:\2023.2\TH KTM\T\MARSfile\Lab2\Assignment\_4.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090005	addi \$9,\$0,0x00000005	6: addi \$t1, \$zero, 5 # X = \$t1 = ?
	0x00400004	0x200affff	addi \$10,\$0,0xfffffff	7: addi \$t2, \$zero, -1 # Y = \$t2 = ?
	0x00400008	0x01298020	add \$16,\$9,\$9	10: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = X + X = 2X
	0x0040000c	0x020a8020	add \$16,\$16,\$10	11: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2X + Y

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	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

Mars Messages Run I/O

Reset: reset completed.

-- program is finished running (dropped off bottom) --

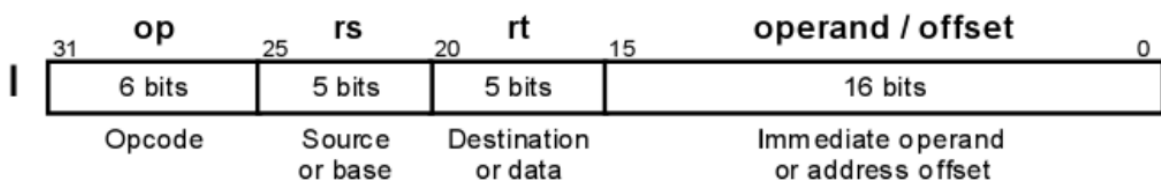
Registers Coproc 1 Coproc 0

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	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000005
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$fp	29	0x7fffffc0
\$ra	30	0x00000000
\$pc	31	0x00400010
\$hi		0x00000000
\$lo		0x00000000

- Có sự thay đổi giá trị ở thanh ghi \$t1, \$t2, \$s0 lần lượt chứa giá trị nhập vào từ lệnh và thực hiện đúng phép toán.
- Kết quả cuối cùng ra đúng.

Text Segment				
Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090005	addi \$9,\$0,0x00000005	6: addi \$t1, \$zero, 5 # X = \$t1 = ?
	0x00400004	0x200affff	addi \$10,\$0,0xfffffff	7: addi \$t2, \$zero, -1 # Y = \$t2 = ?
	0x00400008	0x01298020	add \$16,\$9,\$9	10: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = X + X = 2X
	0x0040000c	0x020a8020	add \$16,\$16,\$10	11: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2X + Y

- Lệnh addi:



addi \$t1, \$zero, 5

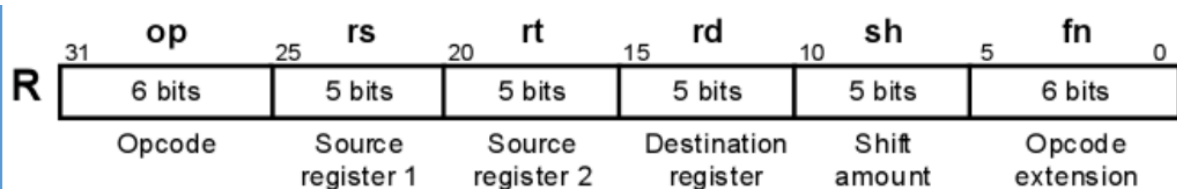
### Bit information

	28	24	20	16	12	8	4	0
Binary	0010	0000	0000	1001	0000	0000	0000	0101
Hex	2	0	0	9	0	0	0	5

opcode	rs	rt	immed
addi	\$zero	\$t1	0x5
001000	00000	01001	000000000000101

- + 6 bits đầu chứa toán tử addi
- + 5 bits tiếp theo chứa vị trí thanh ghi 0
- + 5 bits tiếp theo chứa vị trí thanh ghi t1
- + 16 bits tiếp theo chứa số 5

- Lệnh add



add \$s0, \$t1, \$t1

### Bit information

	28	24	20	16	12	8	4	0
Binary	0000	0001	0010	1001	1000	0000	0010	0000
Hex	0	1	2	9	8	0	2	0

### Info

opcode	rs	rt	rd	shamt	funct
R	\$t1	\$t1	\$s0	0	add
000000	01001	01001	10000	00000	100000

- + 6 bits đầu chứa toán tử
- + 5 bits tiếp theo chứa vị trí thanh ghi nguồn t1
- + 5 bits tiếp theo chứa vị trí thanh ghi nguồn t1
- + 5 bits tiếp theo chứa vị trí thanh ghi đích s0
- + 5 bits tiếp theo chứa giá trị dịch thanh ghi 0
- + 6 bits cuối chứa toán tử add

## Assignment 5:

Code:

#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

```

Result:

The screenshot shows the MARS 4.5 IDE with the following components:

- Text Segment:** Displays the assembly code with comments. The code calculates  $Z = 3 \times XY$  and stores the result in  $s1$ .
- Data Segment:** Shows memory addresses and values. The values are all 0x00000000.
- Registers:** Shows the state of registers. The registers  $s0$  and  $s1$  are highlighted, showing their values.
- MARS Messages:** Shows the execution progress, including the message "Assemble: operation completed successfully."

The screenshot shows the MARS 4.5 IDE with the following components:

- Text Segment:** Displays the assembly code with comments. The code calculates  $Z = 3 \times XY$  and stores the result in  $s1$ .
- Data Segment:** Shows memory addresses and values. The values are all 0x00000000.
- Registers:** Shows the state of registers. The registers  $s0$  and  $s1$  are highlighted, showing their values.
- MARS Messages:** Shows the execution progress, including the message "program is finished running (dropped off bottom)".

- Thực hiện phép nhân trong chương trình, kết quả không trực tiếp ghi vào thanh ghi đích mà được ghi tạm ở thanh ghi LO và HI với LO ghi kết quả, HI ghi số dư.



- + mul \$s0, \$t1, \$t2 thực hiện phép nhân sau đó ghi kết quả vào thanh ghi LO là 20 (0x00000014 ở hệ 16)
- + Thanh ghi HI vì dư 0 nên ghi là 0 (0x00000000)
- Sau đó, lệnh mflo \$s1 dịch chuyển kết quả giá trị từ thanh ghi LO sang thanh ghi s1 bên ngoài lưu giá trị.

## Assignment 6:

Code:

#Laboratory Exercise 2, Assignment 6

```
.data          # DECLARE VARIABLES
    X : .word 5          # Variable X, word type, init value =
    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
```

Result:

Assembly code snippet from the first image:

```

0x00400000 0x3c011001 lui $1,0x00001001      8:      la $t0, X # Get the address of X in Data Segment
0x00400004 0x34380000 ori $24,$1,0x00000000
0x00400008 0x3c011001 lui $1,0x00001001      9:      la $t5, Y # Get the address of Y in Data Segment
0x0040000c 0x34390004 ori $25,$1,0x00000004
0x00400010 0x8f090000 lw $9,0x00000000($24)    10:     lw $t1, 0($t0) # $t1 = X
0x00400014 0x8f2a0000 lw $10,0x00000000($... 11:     lw $t2, 0($t9) # $t2 = Y
0x00400018 0x01298020 add $16,$9,$9          14:     add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
0x0040001c 0x020a8020 add $16,$16,$10         15:     add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
0x00400020 0x3c011001 lui $1,0x00001001      18:     la $t7, Z # Get the address of Z in Data Segment
0x00400024 0x342f0008 ori $15,$1,0x00000008
0x00400028 0xadf00000 sw $16,0x00000000($... 19:     sw $s0, 0($t7) # Z = $s0 = 2X + Y

```

Registers table from the first image:

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
\$a4	8	0x00000000
\$a5	9	0x00000000
\$t0	10	0x00000000
\$t1	11	0x00000000
\$t2	12	0x00000000
\$t3	13	0x00000000
\$t4	14	0x00000000
\$t5	15	0x00000000
\$t6	16	0x00000000
\$t7	17	0x00000000
\$s0	18	0x00000000
\$s1	19	0x00000000
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x00000000
\$s7	25	0x00000000
\$s8	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x0040002c
\$hi		0x00000000
\$lo		0x00000000

Assembly code snippet from the second image:

```

0x00400000 0x3c011001 lui $1,0x00001001      8:      la $t0, X # Get the address of X in Data Segment
0x00400004 0x34380000 ori $24,$1,0x00000000
0x00400008 0x3c011001 lui $1,0x00001001      9:      la $t5, Y # Get the address of Y in Data Segment
0x0040000c 0x34390004 ori $25,$1,0x00000004
0x00400010 0x8f090000 lw $9,0x00000000($24)    10:     lw $t1, 0($t0) # $t1 = X
0x00400014 0x8f2a0000 lw $10,0x00000000($... 11:     lw $t2, 0($t9) # $t2 = Y
0x00400018 0x01298020 add $16,$9,$9          14:     add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
0x0040001c 0x020a8020 add $16,$16,$10         15:     add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
0x00400020 0x3c011001 lui $1,0x00001001      18:     la $t7, Z # Get the address of Z in Data Segment
0x00400024 0x342f0008 ori $15,$1,0x00000008
0x00400028 0xadf00000 sw $16,0x00000000($... 19:     sw $s0, 0($t7) # Z = $s0 = 2X + Y

```

Registers table from the second image:

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
\$a4	8	0x00000000
\$a5	9	0x00000000
\$t0	10	0x00000000
\$t1	11	0x00000000
\$t2	12	0x00000000
\$t3	13	0x00000000
\$t4	14	0x00000000
\$t5	15	0x00000000
\$t6	16	0x00000000
\$t7	17	0x10010008
\$s0	18	0x00000000
\$s1	19	0x00000000
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x10010000
\$s7	25	0x10010004
\$s8	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x0040002c
\$hi		0x00000000
\$lo		0x00000000

Assembly code snippet from the third image:

```

0x00400000 0x3c011001 lui $1,0x00001001      8:      la $t0, X # Get the address of X in Data Segment
0x00400004 0x34380000 ori $24,$1,0x00000000
0x00400008 0x3c011001 lui $1,0x00001001      9:      la $t5, Y # Get the address of Y in Data Segment
0x0040000c 0x34390004 ori $25,$1,0x00000004
0x00400010 0x8f090000 lw $9,0x00000000($24)    10:     lw $t1, 0($t0) # $t1 = X
0x00400014 0x8f2a0000 lw $10,0x00000000($... 11:     lw $t2, 0($t9) # $t2 = Y
0x00400018 0x01298020 add $16,$9,$9          14:     add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
0x0040001c 0x020a8020 add $16,$16,$10         15:     add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
0x00400020 0x3c011001 lui $1,0x00001001      18:     la $t7, Z # Get the address of Z in Data Segment
0x00400024 0x342f0008 ori $15,$1,0x00000008
0x00400028 0xadf00000 sw $16,0x00000000($... 19:     sw $s0, 0($t7) # Z = $s0 = 2X + Y

```

Registers table from the third image:

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
\$a4	8	0x00000000
\$a5	9	0x00000000
\$t0	10	0x00000000
\$t1	11	0x00000000
\$t2	12	0x00000000
\$t3	13	0x00000000
\$t4	14	0x00000000
\$t5	15	0x00000000
\$t6	16	0x00000000
\$t7	17	0x10010008
\$s0	18	0x00000000
\$s1	19	0x00000000
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x10010000
\$s7	25	0x10010004
\$s8	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x0040002c
\$hi		0x00000000
\$lo		0x00000000

- Lệnh la (load address) được chuyển thành 2 lệnh lui và ori dùng để lưu địa chỉ của biến được khởi tạo trên vùng .data
  - + Các biến X Y Z được lệnh la lưu lại giá trị địa chỉ lần lượt theo bảng Labels.
- Các lệnh lw (load word) và sw (store word) có nhiệm vụ ghi vào thanh ghi và lưu lại giá trị word (4 byte) của thanh ghi vào 1 thanh ghi khác
  - + Lệnh sw lưu lại nội dung của thanh ghi này vào thanh ghi khác.