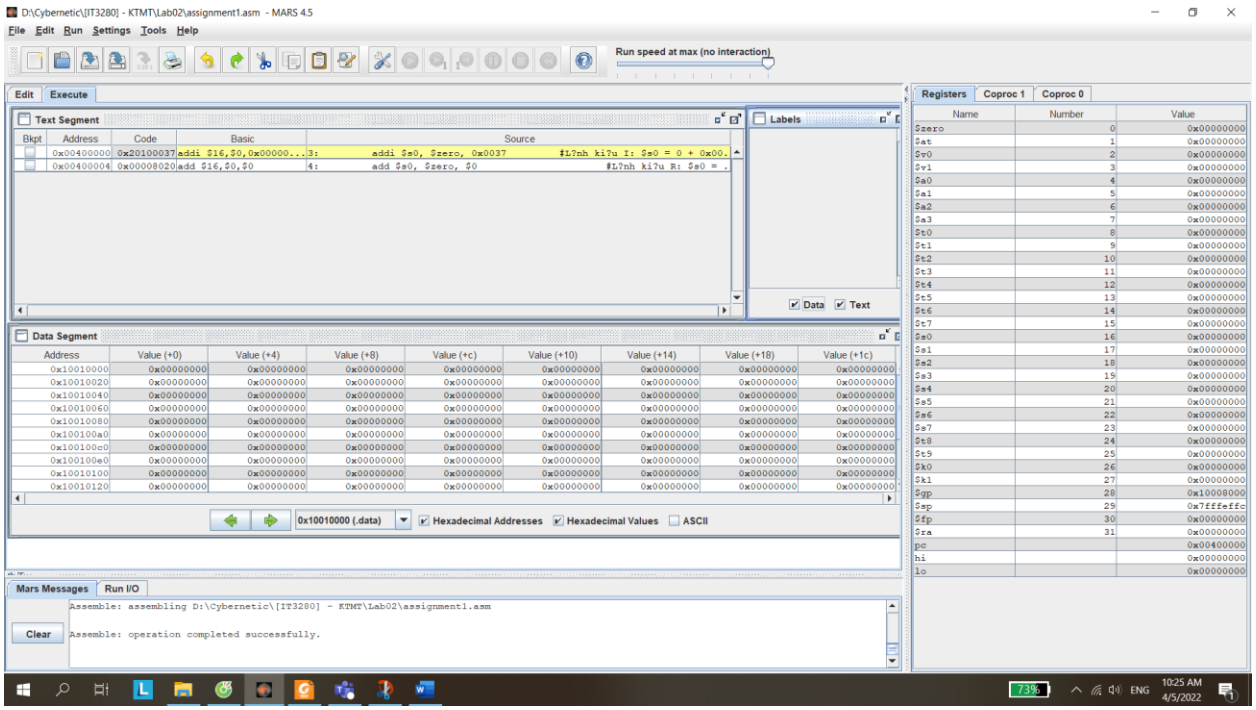
1. Assignment 1

```
#Laboratory Exercise 2, Assignment 1

.text

    addi $s0, $zero, 0x0037          #Lệnh kiểu I: $s0 = 0 + 0x0037

    add $s0, $zero, $0               #Lệnh kiểu R: $s0 = 0 + 0
```



- Giá trị của các thanh ghi sau khi thực hiện:
    - addi $s0, $zero, 0x0037 => $s0 = 0x00000037

Mã máy:

addi	$zero	$s0	0x0037
8	0	16	0x0037
001000	00000	10000	0000 0000 0011 0111

2	0	1	0	0	0	3	7

⇨ Mã máy: 0x20100037

- add $s0, $zero, $0 => $s0 = 0x00000000

Mã máy:

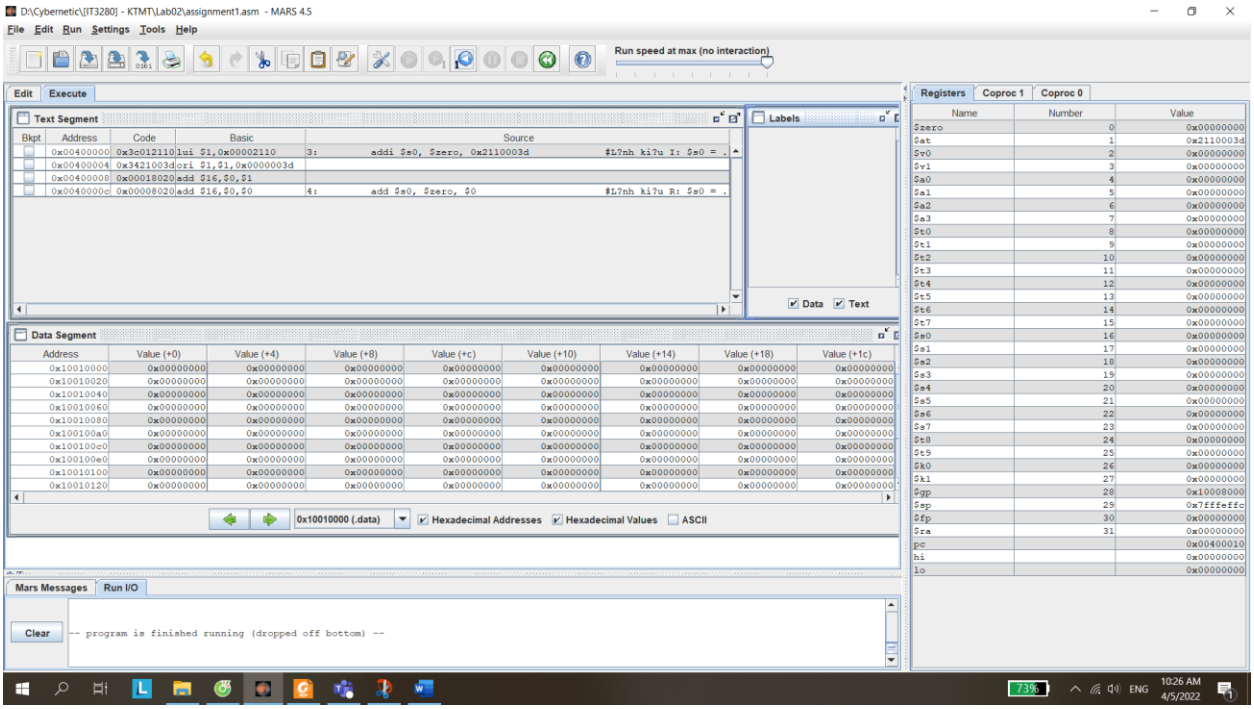
add	$zero	$0	$s0	0	0x20
0	0	0	16	0	32
000000	00000	00000	10000	00000	100000

0	0	0	0	8	0	2	0

⇨ Mã máy: 0x00008020

- Khi sửa thành addi $s0, $zero, 0x2110003d, khi biên dịch chương trình, do lệnh addi là lệnh kiểu I, phần hằng số immediate chỉ cho phép tối đa 16 bit => tự động được tách thành 2 lệnh thêm vào nửa cao và nửa thấp của thanh ghi 32 bit lui và ori

```
lui $1, 0x00002110        # $at = $1 = 0x21100000
ori $1, $1, 0x003d        # $at = $1 = 0x2110003d
```



- Sau đó, chương trình mới thực hiện tiếp câu lệnh addi $s0, $zero, 0x2110003d để đưa giá trị 0x2110003d vào thanh ghi $s0

2. Assignment 2

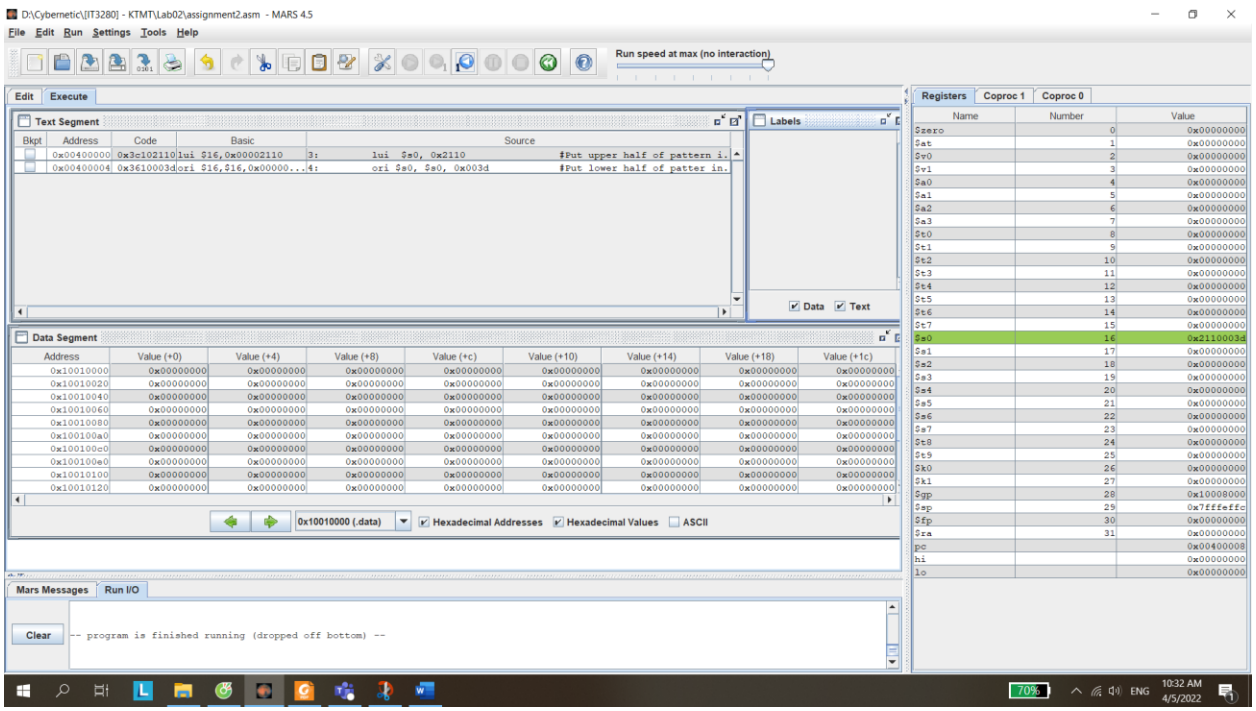
```
#Laboratory Exercise 2, Assignment 2

.text

        lui $s0, 0x2110             #Put upper half of pattern in $s0

        ori $s0, $s0, 0x003d        #Put lower half of patter in $s0
```

- Lệnh lui nạp 0x2110 vào nửa cao của thanh ghi $s0, lệnh ori thực hiện nạp nửa thấp còn lại 0x003d vào nửa thấp của thanh ghi $s0
- Kết quả, thanh ghi $s0 = 0x2110003d



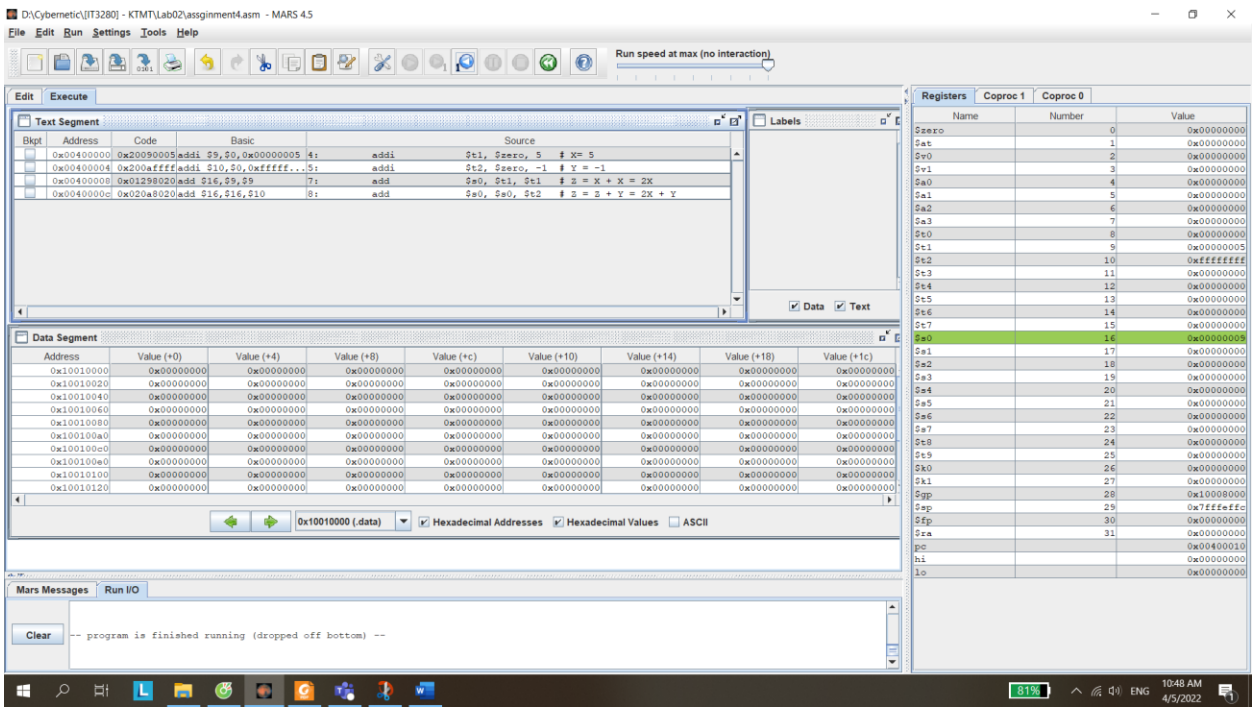
3. Assignment 3

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

- Điều bất thường xảy ra ở câu lệnh li $s1, 0x2, khi chương trình được biên dịch, câu lệnh chuyển thành addiu $s1, $0, 0x00000002.
- Sở dĩ có điều bất thường trên là do giá trị immediate ở đây rất nhỏ và không được viết theo dạng chuẩn của số hex 32bit nên câu lệnh được chuyển thành phép cộng số nguyên với toán hạng tức thì không dấu (opcode chuyển thành 9_{hex})

4. Assignment 4

```
#Laboratory Exercise 2, Assignment 4
.text
        #Assign X, Y
        addi   $t1, $zero, 5        # X = 5  ($t1 = 0 + 5 = 5)
        addi   $t2, $zero, -1       # Y = -1 ($t2 = 0 + (-1) = -1)
        #Expression Z = 2X + Y
        add    $s0, $t1, $t1        # Z = X + X = 2X = 10 ($s0 = $t1 + $t1)
        add    $s0, $s0, $t2        # Z = Z + Y = 2X + Y = 9 ($s0 = $s0 + $t2)
```



- addi $t1, $zero, 5 => $st1 = 5

Mã máy:

addi	$zero	$t1	5
8	0	9	5
001000	00000	01001	0000 0000 0000 0101

2	0	0	9	0	0	0	5

⇨ Mã máy: 0x20090005

- addi $t2, $zero, -1 => $st1 = -1

Mã máy:

addi	$zero	$t2	-1
8	0	10	ffff
001000	00000	01010	1111 1111 1111 1111

2	0	0	a	f	f	f	f

⇨ Mã máy: 0x200affff

- add $s0, $t1, $t1 => $s0 = $t1 + $t1 = 5 + 5 = 10

Mã máy:

add	$t1	$t1	$s0	0	0x20
0	9	9	16	0	32
000000	01001	01001	10000	00000	100000

0	1	2	9	8	0	2	0

⇨ Mã máy: 0x01298020

- add $s0, $s0, $t2 => $s0 = $s0 + $t2 = 10 + (-1) = 9

Mã máy:

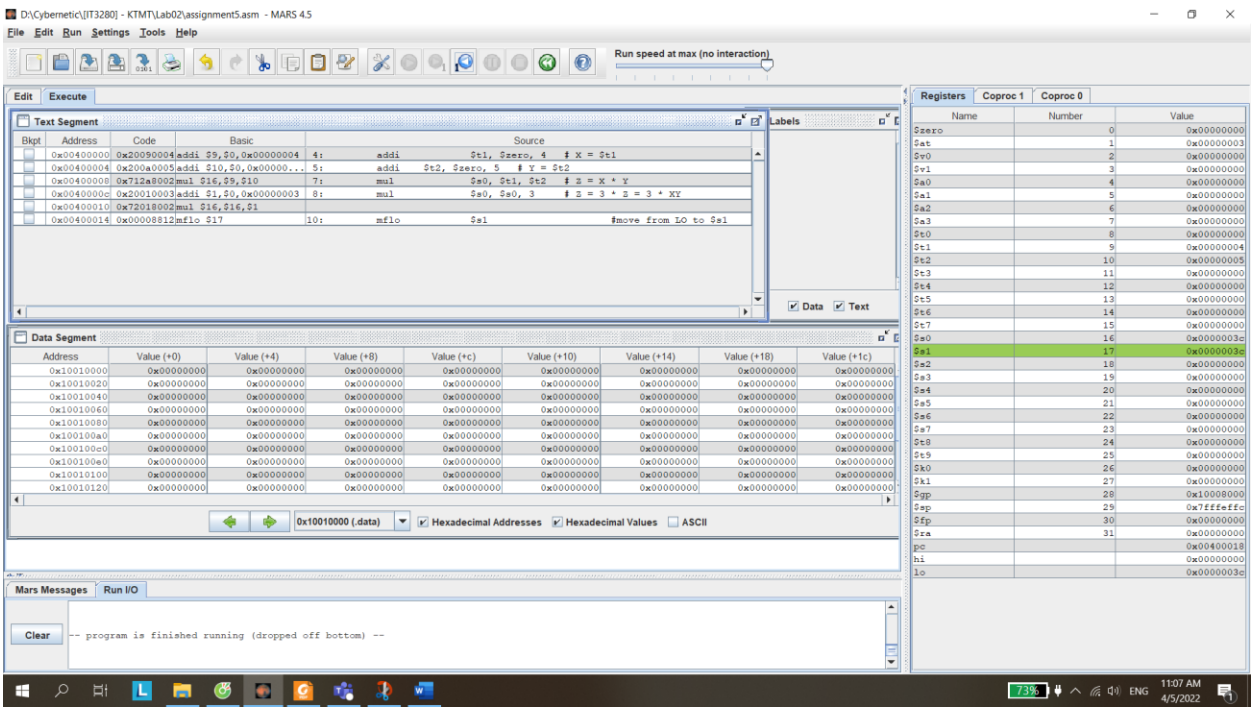
add	$s0	$t2	$s0	0	0x20
0	16	10	16	0	32
000000	10000	01010	10000	00000	100000

0	2	0	a	8	0	2	0

⇨ Mã máy: 0x020a8020

5. Assignment 5

```
#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		# Z = X * Y (HI-LO = $t1 * $t2, LO = 0x00000004 * 0x00000005 = 0x00000014)
	mul	$s0, $s0, 3		# Z = 3 * Z = 3 * XY ($at = 0x00000003, HI-LO = $s0 * $tat, LO = 0x00000014 * 0x00000003 = 0x0000003c)
	#Z' = Z
	mflo	$s1		# move from LO to $s1 ($s1 = Z’ = LO = 0x0000003c)
```



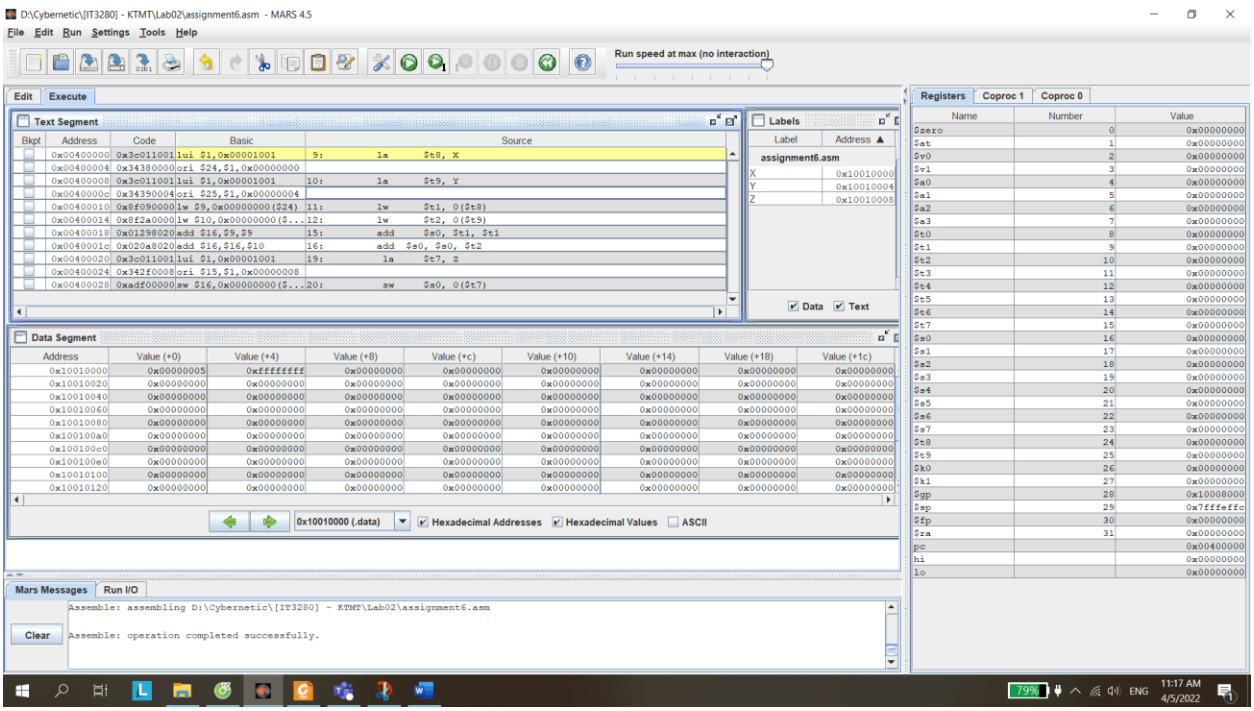
6. Assignment 6

```
#Laboratory Exercise 2, Assignment 6
.data                                   #DECLARE VARIABLES
X:      .word  5
Y:      .word  -1
Z:      .word

.text                                   #DECLARE INSTRUCTION
        #Load X, Y to registers
        la     $t8, X
        la     $t9, Y
        lw     $t1, 0($t8)
        lw     $t2, 0($t9)

        #Calculate the expression Z = 2X + Y with registers only
        add    $s0, $t1, $t1
        add  $s0, $s0, $t2

         #Store result from register to variable Z
         la    $t7, Z
         sw    $s0, 0($t7)
```



- Lệnh la (load address) được biên dịch
  - Chuyển thành 2 lệnh lui, ori
  - Lưu địa chỉ biến vào thanh ghi:
    - $t8 = địa chỉ X
    - $t9 = địa chỉ Y

- Vai trò của lệnh lw, sw
  - lw rd, imm(rs) # với rd = địa chỉ bộ nhớ[rs + imm], imm = offset *
    - Vai trò: đọc dữ liệu word từ địa chỉ imm(rs) lưu vào thanh ghi rd
  - sw rd, imm(rs) # với rd = địa chỉ bộ nhớ[rs + imm], imm = offset *
    - Vai trò: ghi dữ liệu word từ thanh ghi rs vào ngăn nhớ địa chỉ imm(rs)
  -