

# Laboratory Exercise 4

## Arithmetic and Logical operation

Trần Khánh Quỳnh – 20225762

### 1. Assignment 1

TH1: Cùng dương, không tràn số (2024 và 2023)

```

1  #Laboratory Assignment 1
2  .text
3      li $s1, 2024
4      li $s2, 2023
5  start:
6      li $t0, 0          #No Overflow is default status
7      addu $s3, $s1, $s2  # s3 = s1 + s2
8      xor $t1, $s1, $s2   #Test if $s1 and $s2 have the same sign
9      bltz $t1, EXIT      #If not, exit
10     slt $t2, $s3, $s1
11     bltz $s1, NEGATIVE  #Test if $s1 and $s2 is negative?
12     beq $t2, $zero, EXIT #s1 and $s2 are positive
13     # if $s3 > $s1 then the result is not overflow
14     j OVERFLOW
15  NEGATIVE:
16     bne $t2, $zero, EXIT #s1 and $s2 are negative
17     # if $s3 < $s1 then the result is not overflow
18     OVERFLOW:
19     li $t0, 1
20  EXIT:

```

-Kết quả chạy:

The screenshot shows the Mars MIPS simulator interface. The **Text Segment** window displays the assembly code with addresses and source code. The **Registers** window shows the values of registers \$s1 (2024), \$s2 (2023), and \$s3 (4047). The **Data Segment** window shows memory addresses and values.

Register	Value
\$zero	0
\$at	1
\$v0	2
\$v1	3
\$a0	4
\$a1	5
\$a2	6
\$a3	7
\$t0	8
\$t1	9
<b>\$t2</b>	<b>10</b>
\$t3	11
\$t4	12
\$t5	13
\$t6	14
\$t7	15
\$s0	16
\$s1	2024
\$s2	2023
\$s3	4047
\$s4	20
\$s5	21
\$s6	22
\$s7	23
\$t8	24
\$t9	25
\$k0	26
\$k1	27
\$gp	28
\$fp	29
\$ra	31
pc	4194352
hi	
lo	

Thanh ghi \$s1 có giá trị là 2024; thanh ghi \$s2 có giá trị là 2023. Hai câu lệnh li đầu tiên lưu giá trị vào 2 thanh ghi \$s1 và \$s2

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	2024
\$s2	18	2023
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194312
hi		0
lo		0

Câu lệnh li thứ 3 mặc định trạng thái hiện tại đang là không overflow. Câu lệnh addu \$s3, \$s2, \$s1 lưu kết quả phép tính cộng giá trị hai thanh ghi \$s2 và \$s1 vào thanh ghi \$s3

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	2024
\$s2	18	2023
\$s3	19	4047
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194320
hi		0
lo		0

Phép XOR cho ra kết quả là 15 → hai số này cùng dấu. Như vậy \$t1 có giá trị là 15, lớn hơn 0 nên ở câu lệnh bltz \$t1, EXIT, chương trình không nhảy xuống nhánh EXIT mà tiếp tục thực hiện các câu lệnh ở phía sau.

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	19
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	2024
\$s2	18	2023
\$s3	19	4047
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194324
hi		0
lo		0

Lệnh slt \$t2, \$s3, \$s1 so sánh xem \$s3 có nhỏ hơn \$s1 không. Trong trường hợp này thì \$s3 không nhỏ hơn \$s1 nên giá trị của \$t2 là 0

Lệnh bltz \$s1, NEGATIVE kiểm tra xem \$s1 có âm không. Vì trong trường hợp này không âm nên chương trình không nhảy xuống NEGATIVE.

Lệnh beq \$t2, \$zero, EXIT so sánh xem \$t2 có bằng 0 không. Ở trường hợp này = 0 → nhảy xuống nhánh EXIT, kết thúc chương trình.

## TH2: Trái dấu (-2024 và 2023)

Thanh ghi \$s1 và \$s2 lưu lại giá trị, thanh ghi \$t0 mặc định ban đầu bằng 0 (Không overflow). Lệnh addu lưu lại giá trị \$s1 + \$s2 vào \$s3. Ở trường hợp này thì \$s3 bằng -1

Phép xor \$s1 và \$s2 có kết quả là -1, lưu lại vào \$t1. Vì \$t1 < 0 nên khi thực hiện xong câu lệnh bltz \$t1, EXIT thì nhảy thẳng xuống nhánh EXIT và kết thúc chương trình

## TH3: Cùng âm (-2024 và -2023), không tràn số

Lưu lại các giá trị vào thanh ghi \$s1 và \$s2, mặc định ban đầu không có overflow. Lưu kết quả giá trị \$s1 và \$s2 vào \$s3 → giá trị của \$s3 lúc này là -4047. Lệnh xor \$t1, \$s1, \$s2 lưu kết quả phép xor giữa giá trị của \$s1 và \$s2, vì vậy giá trị \$t1 là 1.  $1 > 0$  nên vẫn tiếp tục chạy các lệnh phía sau lệnh bltz \$t1, EXIT. Lệnh slt \$t2, \$s3, \$s1 so sánh giá trị của \$s3 có nhỏ hơn \$s1 không. Vì ở trường hợp này giá trị \$s1 < 0 nên nhảy xuống nhánh NEGATIVE.

Trong nhánh NEGATIVE, kiểm tra xem \$t2 và \$zero có bằng nhau không. Vì không bằng nhau (\$t2 = 1 do \$s3 < \$s1) nên nhảy xuống EXIT và kết thúc chương trình

#### TH4: Cùng dương, overflow

```

1  #Laboratory Assignment 1
2  .text
3      li $s1, 2147483646
4      li $s2, 2023
5
6  start:
7      li $t0,0                #No Overflow is default status
8      addu $s3,$s1,$s2        # s3 = s1 + s2
9      xor $t1,$s1,$s2        #Test if $s1 and $s2 have the same sign
10     bltz $t1,EXIT           #If not, exit
11     slt $t2,$s3,$s1
12     bltz $s1,NEGATIVE       #Test if $s1 and $s2 is negative?
13     beq $t2,$zero,EXIT      #s1 and $s2 are positive
14     # if $s3 > $s1 then the result is not overflow
15     j OVERFLOW
16  NEGATIVE:
17     bne $t2,$zero,EXIT
18     #s1 and $s2 are negative
19     # if $s3 < $s1 then the result is not overflow
20  OVERFLOW:
21     li $t0, 1
22  EXIT:

```

-Kết quả chạy:

Text Segment

View and control assembly language program execution

Enabled upon successful assemble

Bkpt	Address	Code	Basic	Source
	0x00400004	ori \$17,\$1,65534		
	0x00400008	addiu \$18,\$0,2023	4:	li \$s2, 2023
	0x0040000c	addiu \$8,\$0,0	7:	li \$t0,0
	0x00400010	addu \$19,\$17,\$18	8:	addu \$s3,\$s1,\$s2
	0x00400014	xor \$1,\$17,\$18	9:	xor \$t1,\$s1,\$s2
	0x00400018	bltz \$9,\$6	10:	bltz \$t1,EXIT
	0x0040001c	slt \$10,\$19,\$17	11:	slt \$t2,\$s3,\$s1
	0x00400020	bltz \$10,\$17	12:	bltz \$s1,NEGATIVE
	0x00400024	beq \$10,\$0,3	13:	beq \$t2,\$zero,EXIT
	0x00400028	jump \$0,\$0,15	14:	j OVERFLOW
	0x0040002c	bne \$10,\$0,1	17:	bne \$t2,\$zero,EXIT
	0x00400030	addiu \$8,\$0,1	21:	li \$t0, 1

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

0x10010000 (.data)

☒ Hexadecimal Addresses

☐ Hexadecimal Values

☐ ASCII

Mars Messages

Run IO

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

Name	Number	Value
\$zero	0	0
\$at	1	2147418112
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	2147401625
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	2147483646
\$s2	18	2023
\$s3	19	-2147481627
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$t6	26	0
\$t1	27	0
\$sp	28	268468224
\$sp	29	2147479540
\$fp	30	0
\$ra	31	0
pc		4194356
hi		0
lo		0

nhánh EXIT mà thực hiện nốt lệnh j OVERFLOW. Sau khi nhảy sang nhánh OVERFLOW thì lưu 1 vào \$t0, tức có tồn tại tràn số.

## 2. Assignment 2

```

1  #Laboratory Assignment 2
2  .text
3  li $s0, 0x12345678 #0001 0010 0011 0100 0101 0110 0111 1000
4  andi $t0, $s0, 0xff000000 #extract msb
5  andi $t1, $s0, 0xffffffff00 #clear LSB
6  ori $t2, $s0, 0x00000040 #or, bit 7 = 1
7  andi $t3, $s0, 0x0 #clear $s0
8

```

-Kết quả chạy:

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays the assembly code, and the 'Registers' window shows the values of the registers.

Register	Value
\$s0	0x12345678
\$t0	0x12000000
\$t1	0x12345600
\$t2	0x12345678
\$t3	0x00000000

Lưu giá trị vào thanh ghi \$s0. \$s0 = 0x012345678

Lưu MSB vào \$t0 → \$t0 = 0x12000000

Xóa LSB đi → \$t1 = 0x12345600

Lưu LSB, bit số 7 thành 1 → thực hiện phép OR giữa 0x12345678 và 0x00000040

0x12345678 = 0001 0010 0011 0100 0101 0110 0111 1000

0x00000040 = 0000 0000 0000 0000 0000 0000 0100 0000

→ Kết quả của phép OR sẽ là 0001 0010 0011 0100 0101 0110 0111 1000

= 0x12345678 → \$t2 = 0x12345678

Xóa \$s0 → \$t3 = 0x00000000

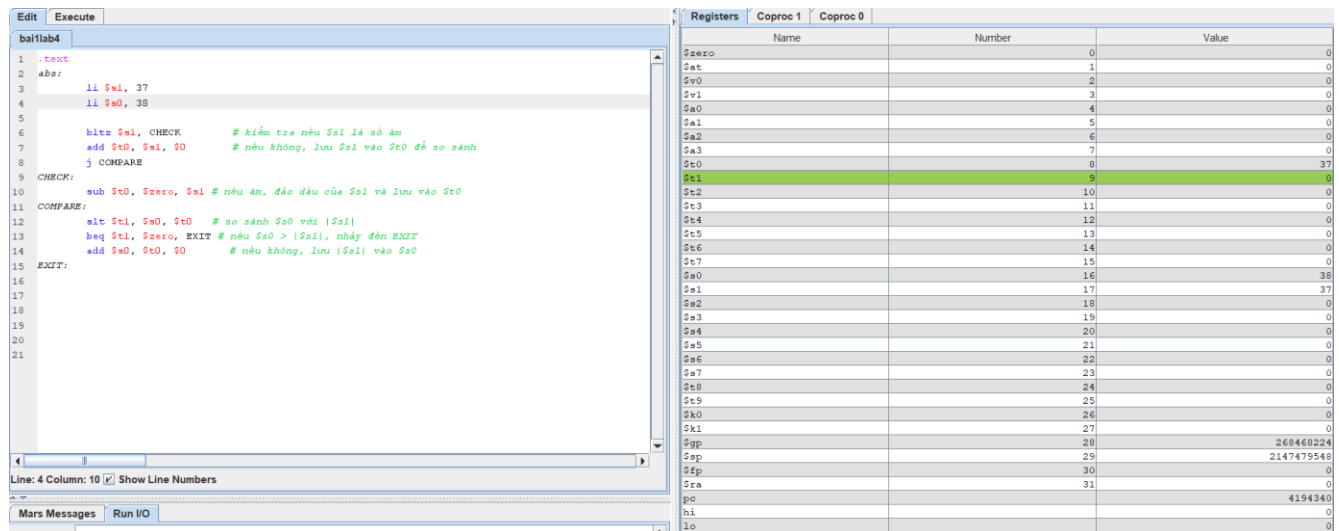
### 3. Assignment 3

a) `abs $s0, $s1` khi `$s0 <= |$s1|`

```
1  .text
2  abs:
3      li $s1, -37
4      li $s0, -38
5
6      bltz $s1, CHECK      # kiểm tra nếu $s1 là số âm
7      add $t0, $s1, $0     # nếu không, lưu $s1 vào $t0 để so sánh
8      j COMPARE
9  CHECK:
10     sub $t0, $zero, $s1  # nếu âm, đảo dấu của $s1 và lưu vào $t0
11  COMPARE:
12     slt $t1, $s0, $t0    # so sánh $s0 với |$s1|
13     beq $t1, $zero, EXIT # nếu $s0 > |$s1|, nhảy đến EXIT
14     add $s0, $t0, $0     # nếu không, lưu |$s1| vào $s0
15  EXIT:
16  |
```

-Kết quả chạy: Lưu kết quả `abs($s1)` vào thanh ghi `$s0`

TH1: `$s1 = 37; $s0 = 38`



Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$s1	9	37
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	38
\$a1	17	37
\$a2	18	0
\$a3	19	0
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$a8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268440224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
\$0		4194360
\$1		0
\$2		0

Chương trình dừng lại, không thực hiện lệnh `abs` do `$s0` không nhỏ hơn hoặc bằng `|$s1|`

TH2: `$s1 = 37, $s0 = -38`

```

1 .text
2 abs:
3     li $s1, 37
4     li $s0, -38
5
6     hltz $s1, CHECK    # kiểm tra nếu $s1 là số âm
7     add $t0, $s1, $0    # nếu không, lưu $s1 vào $t0 để so sánh
8     j COMPARE
9
10 CHECK:
11     sub $t0, $zero, $s1 # nếu âm, đảo dấu của $s1 và lưu vào $t0
12 COMPARE:
13     slt $t1, $s0, $t0    # so sánh $s0 với |$s1|
14     beq $t1, $zero, EXIT # nếu $s0 > |$s1|, nhảy đến EXIT
15     add $s0, $t0, $0    # nếu không, lưu |$s1| vào $s0
16 EXIT:
17
18
19
20
21

```

Register	Value	Comment
\$zero	0	
\$at	1	
\$v0	2	
\$v1	3	
\$a0	4	
\$a1	5	
\$a2	6	
\$a3	7	
\$t0	8	37
\$t1	9	1
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
<b>\$s0</b>	<b>16</b>	<b>38</b>
\$s1	17	37
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268460224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194340
hi		0
lo		0

Line: 4 Column: 13 ☒ Show Line Numbers

Mars Messages Run I/O

Đã thỏa mãn điều kiện  $s0 \leq |s1|$ , vì vậy chương trình  $\text{abs } s0, s1$  chạy  $\rightarrow s0 =$

TH3:  $s_1 = -37$ ;  $s_0 = -38$

```

1 .text
2 abs:
3     li $a1, -37
4     li $a0, -38
5
6     bltz $a1, CHECK      # kiểm tra nếu $a1 là số âm
7     add $t0, $a1, $0     # nếu không, lưu $a1 vào $t0 để so sánh
8
9     j COMPARE
10
11 CHECK:
12     sub $t0, $zero, $a1 # nếu âm, đảo dấu của $a1 và lưu vào $t0
13
14 COMPARE:
15     slt $t1, $a0, $t0    # so sánh $a0 với |$a1|
16     beq $t1, $zero, EXIT # nếu $a0 > |$a1|, nhảy đến EXIT
17     add $a0, $t0, $0     # nếu không, lưu |$a1| vào $a0
18
19 EXIT:
20
21

```

\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	37
\$t1	9	1
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$a0	16	37
\$a1	17	-37
\$a2	18	0
\$a3	19	0
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$f0	28	268469224
\$f1	29	2147479548
\$f2	30	0
\$f3	31	0
pc		4194340
hi		0
lo		0

Line: 2 Column: 5 ☒ Show Line Numbers

Mars Messages

Run I/O

Thỏa mãn điều kiện  $s_0 \leq |s_1| \rightarrow$  chương trình abs chạy  $\rightarrow s_0 = 37$

**b) move \$s0, \$s1 với  $\$s0 \leq \$s1$**

Lưu nội dung trong thanh ghi \$s0 sang thanh ghi \$s1 khi \$s0 <= \$s1

```

.text
move:  li $s1, -1
        li $s0, 2
        #dk: $s0 <= $s1
        slt $t0, $s1, $s0 # $s0 > $s1 --> $t0 = 1
        addi $at, $0, 1
        beq $t0, $at, EXIT
        add $s1, $s0, 0

EXIT:

```

TH1: \$s1 = -1; \$s0 = 2 (\$s0 > \$s1)

The screenshot shows the Mars MIPS simulator interface. On the left, the code window displays the assembly code for the first test case. On the right, the registers window shows the state of the registers. The program has finished running, and the registers show the following values:

Register	Value
\$zero	0
\$at	1
\$v0	0
\$v1	0
\$a0	0
\$a1	0
\$a2	0
\$a3	0
\$t0	1
\$t1	0
\$t2	0
\$t3	0
\$t4	0
\$t5	0
\$t6	0
\$t7	0
\$s0	2
\$s1	-1
\$s2	0
\$s3	0
\$s4	0
\$s5	0
\$s6	0
\$s7	0
\$t8	0
\$t9	0
\$k0	0
\$k1	0
\$gp	268469224
\$sp	2147479540
\$fp	0
\$ra	0
pc	4194328
hi	0
lo	0

Kết quả: Không lưu giá trị từ \$s0 sang \$s1 do không thỏa mãn điều kiện \$s0 <= \$s1

TH2: \$s1 = -1; \$s0 = -2 (\$s0 <= \$s1)

The screenshot shows the Mars MIPS simulator interface. On the left, the code window displays the assembly code for the second test case. On the right, the registers window shows the state of the registers. The program has finished running, and the registers show the following values:

Register	Value
\$zero	0
\$at	1
\$v0	0
\$v1	0
\$a0	0
\$a1	0
\$a2	0
\$a3	0
\$t0	1
\$t1	0
\$t2	0
\$t3	0
\$t4	0
\$t5	0
\$t6	0
\$t7	0
\$s0	-2
\$s1	-2
\$s2	0
\$s3	0
\$s4	0
\$s5	0
\$s6	0
\$s7	0
\$t8	0
\$t9	0
\$k0	0
\$k1	0
\$gp	268469224
\$sp	2147479540
\$fp	0
\$ra	0
pc	4194328
hi	0
lo	0

Thỏa mãn điều kiện → Lưu vào thanh ghi \$s1 giá trị của \$s0 → \$s1 = -2



c) not \$s0

s0 <= bit invert(s0)

```
.text
not:
    li $s0, 0x1
    nor $t0, $s0, $0
```

-Kết quả chạy:

Text Segment				
Bkpt	Address	Code	Basic	Source
	0x00400000	0x24100001	addiu \$16,\$0,0x0000...3:	li \$s0, 0x1
	0x00400004	0x02004027	nor \$8,\$16,\$0	4: nor \$t0, \$s0, \$0

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

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	0xffffffff
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000001
\$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	0x10000000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pe		0x00400008
hi		0x00000000
lo		0x00000000

0x1 = 0000 0000 0000 0000 0000 0000 0000 0001

→ Đảo bit: 1111 1111 1111 1111 1111 1111 1111 1110 = 0xffffffff

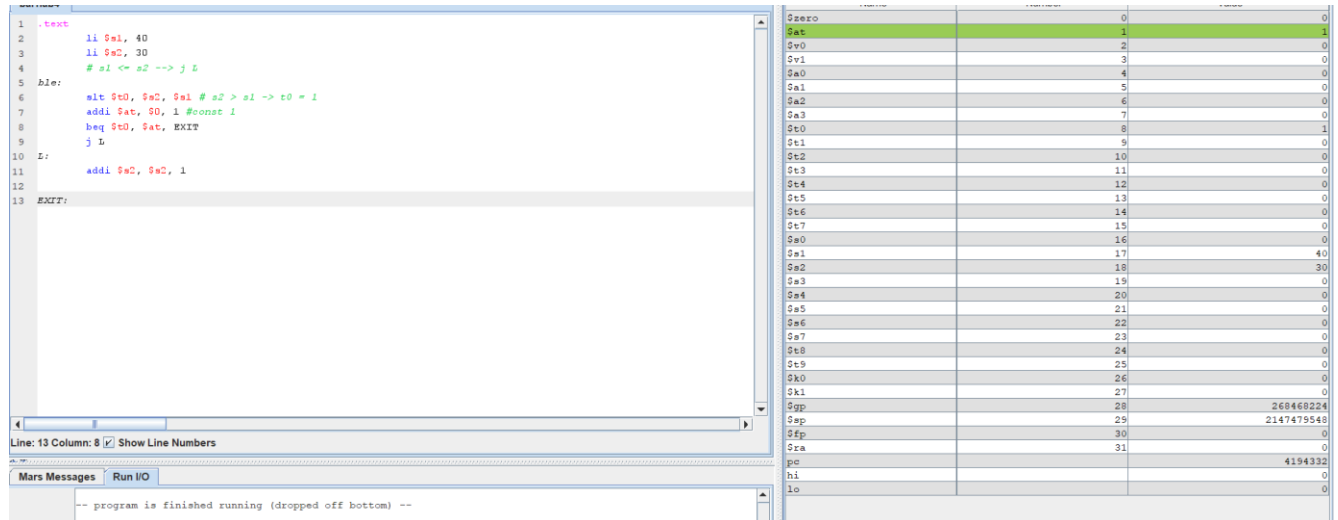
d) ble \$s1, \$s2, L

if(\$s1 <= \$s2)

j L

1	.text
2	li \$s1, 40
3	li \$s2, 30
4	
5	# s1 <= s2 --> j L
6	ble:
7	slt \$t0, \$s2, \$s1 # s1 > s2 --> \$t0 = 1
8	addi \$at, \$0, 1 #const 1
9	beq \$t0, \$at, EXIT
10	j L
11	L:
12	addi \$s2, \$s2, 1
13	EXIT:

TH1: \$s1 = 40; \$s2 = 30 (\$s1 > \$s2)

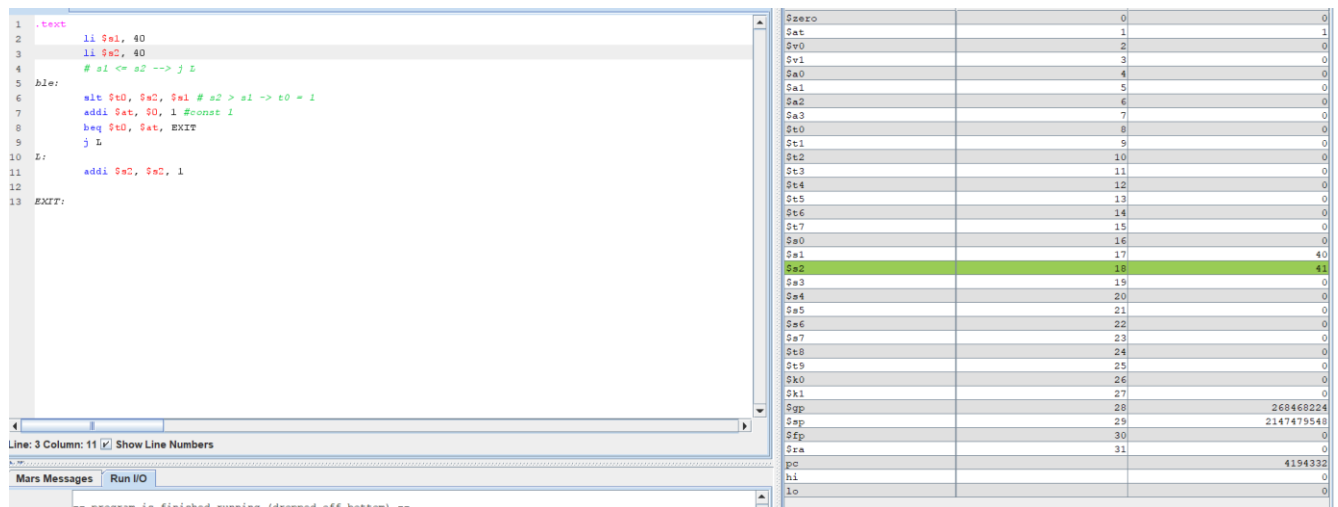


```
1 .text
2   li $s1, 40
3   li $s2, 30
4   # s1 <= s2 --> j L
5 ble:
6   sllt $t0, $s2, $s1 # s2 > s1 -> t0 = 1
7   addi $at, $0, 1 #const 1
8   beq $t0, $at, EXIT
9   j L
10  L:
11   addi $s2, $s2, 1
12
13  EXIT:
-- program is finished running (dropped off bottom) --
```

Register	Value
\$zero	0
\$at	1
\$v0	2
\$v1	3
\$a0	4
\$a1	5
\$a2	6
\$a3	7
\$t0	8
\$t1	9
\$t2	10
\$t3	11
\$t4	12
\$t5	13
\$t6	14
\$t7	15
\$s0	16
\$s1	17
\$s2	18
\$s3	19
\$s4	20
\$s5	21
\$s6	22
\$s7	23
\$s8	24
\$s9	25
\$k0	26
\$k1	27
\$gp	28
\$sp	29
\$fp	30
\$ra	31
pc	4194332
hi	0
lo	0

Chương trình kết thúc do không thỏa mãn điều kiện \$s1 <= \$s2

TH2: \$s1 = 40; \$s2 = 40



```
1 .text
2   li $s1, 40
3   li $s2, 40
4   # s1 <= s2 --> j L
5 ble:
6   sllt $t0, $s2, $s1 # s2 > s1 -> t0 = 1
7   addi $at, $0, 1 #const 1
8   beq $t0, $at, EXIT
9   j L
10  L:
11   addi $s2, $s2, 1
12
13  EXIT:
-- program is finished running (dropped off bottom) --
```

Register	Value
\$zero	0
\$at	1
\$v0	2
\$v1	3
\$a0	4
\$a1	5
\$a2	6
\$a3	7
\$t0	8
\$t1	9
\$t2	10
\$t3	11
\$t4	12
\$t5	13
\$t6	14
\$t7	15
\$s0	16
\$s1	17
\$s2	18
\$s3	19
\$s4	20
\$s5	21
\$s6	22
\$s7	23
\$s8	24
\$s9	25
\$k0	26
\$k1	27
\$gp	28
\$sp	29
\$fp	30
\$ra	31
pc	4194332
hi	0
lo	0

Thỏa mãn điều kiện \$s1 <= \$s2 nên nhảy đến nhánh L. Nhánh L ở đây tăng \$s2 thêm 1 nên \$s2 lúc này = 41

## 4. Assignment 4

```
1  .text
2      li $s1, 0x7fffffff
3      li $s2, 0x00000001
4
5      bltz $s1, NEGATIVE
6      li $t3, 0 #not overflow
7      addu $s3, $s1, $s2
8      slt $t0, $s3, $0 #s3 < 0 -> t0 = 1
9      addi $at, $0, 1
10     beq $t0, $at, OVERFLOW
11     j EXIT
12 NEGATIVE:
13     slt $t1, $s3, $0 # s3 < 0 --> t1 =1
14     beq $s1, $0, OVERFLOW
15     j EXIT
16 OVERFLOW:
17     li $t3, 1
18 EXIT:
19
20
```

-Kết quả chạy:

ADDRESS	INSTR	PC	VALUE
1	.text	0	0
2	li \$s1, 0x7fffffff	1	1
3	li \$s2, 0x00000001	2	0
4		3	0
5	bltz \$s1, NEGATIVE	4	0
6	li \$t3, 0 #not overflow	5	0
7	addu \$s3, \$s1, \$s2	6	0
8	slt \$t0, \$s3, \$0 #s3 < 0 -> t0 = 1	7	0
9	addi \$at, \$0, 1	8	1
10	beq \$t0, \$at, OVERFLOW	9	0
11	j EXIT	10	0
12	NEGATIVE:	11	0
13	slt \$t1, \$s3, \$0 # s3 < 0 --> t1 =1	12	0
14	beq \$s1, \$0, OVERFLOW	13	0
15	j EXIT	14	0
16	OVERFLOW:	15	0
17	li \$t3, 1	16	0
18	EXIT:	17	2147483647
19		18	1
20		19	-2147483648
		20	0
		21	0
		22	0
		23	0
		24	0
		25	0
		26	0
		27	0
		28	268468224
		29	2147479548
		30	0
		31	0
		pc	4194360
		hi	0
		lo	0

Line: 19 Column: 2 Show Line Numbers

Mars Messages Run I/O

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

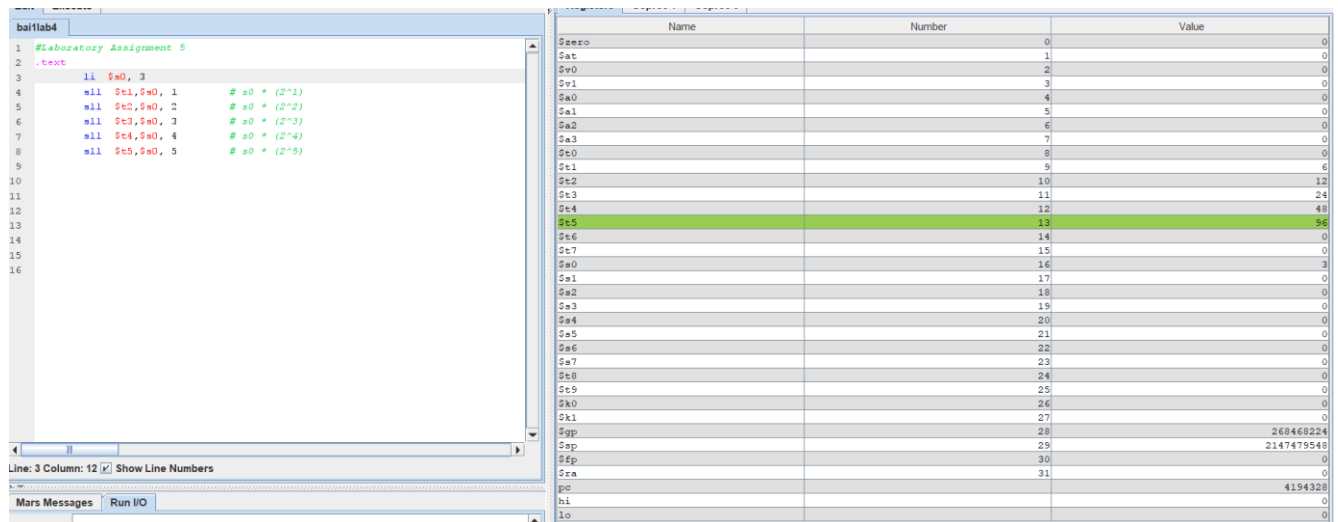
Ban đầu mặc định \$t3 = 0, tức là không có tràn số

Hai số dương ở thanh ghi \$s1 và \$s2 cộng lại ra kết quả là số âm lưu ở \$s3 → \$t3 = 1 do overflow

## 5. Assignment 5

```
1  #Laboratory Assignment 5
2  .text
3      li  $s0, 3
4      sll  $t1,$s0, 1      # s0 * (2^1)
5      sll  $t2,$s0, 2      # s0 * (2^2)
6      sll  $t3,$s0, 3      # s0 * (2^3)
7      sll  $t4,$s0, 4      # s0 * (2^4)
8      sll  $t5,$s0, 5      # s0 * (2^5)
9
10
11
```

-Kết quả chạy:



The screenshot shows the MARS MIPS simulator interface. On the left, the assembly code for 'Laboratory Assignment 5' is displayed. On the right, the register window shows the values of the registers after execution. The registers \$t1 through \$t5 contain the values 3, 6, 12, 24, and 48 respectively, which are the results of the shift operations performed on \$s0 (value 3).

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	3
\$t2	10	6
\$t3	11	12
\$t4	12	24
\$t5	13	48
\$t6	14	0
\$t7	15	0
\$s0	16	3
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268469224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194328
hi		0
lo		0