

Laboratory Exercise 4 – Report:

Arithmetic and Logical operation

Lê Văn Duẩn - 20194508

1. Assignment 1

- Mã nguồn:

#Laboratory Assignment 1

.text

li \$s1, 2019

li \$s2, 4508

start:

li \$t0, 0 # default status

addu \$s3, \$s1, \$s2 # s3 = s1 + s2

xor \$t1, \$s1, \$s2 # test if \$s1 and \$s2 have the same sign

bltz \$t1, EXIT # If not, exit

slt \$t2, \$s3, \$s1

bltz \$s1, NEGATIVE # test if \$s1 and \$s2 is negative?

beq \$t2, \$0, EXIT # s1 and s2 are positive

if \$s3 > \$s1 then the result is overflow

j OVERFLOW

NEGATIVE:

bne \$t2,\$0,EXIT

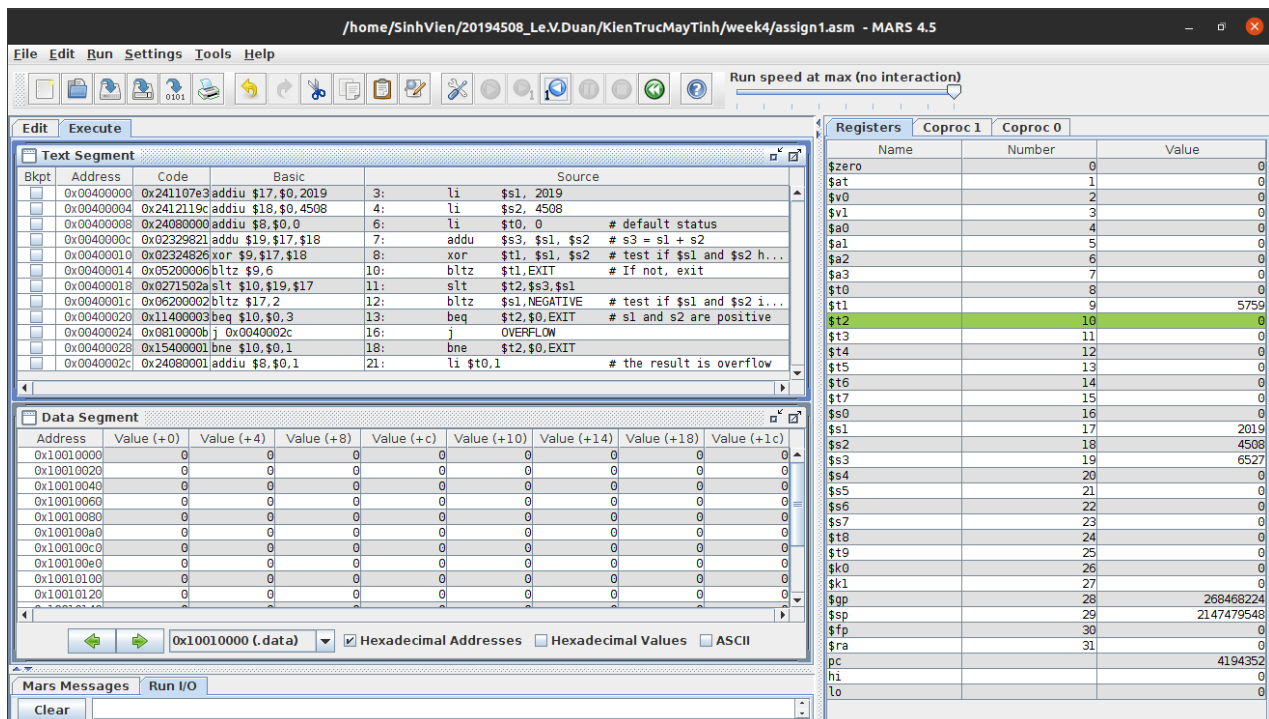
if \$s3 < \$s1 the the result is not overflow

OVERFLOW:

li \$t0,1 # the result is overflow

EXIT:

- Kết quả chạy mô phỏng:



- Giải thích:

Vì giá trị ban đầu khởi tạo cho s1 và s2 là cùng dấu và dương nên sau lệnh xor kết quả trong thanh ghi t1 là số dương vì vậy khi đến câu lệnh so sánh bltz nó sẽ không đi đến nhãn lệnh EXIT

Tại slt \$t2,\$s3,\$s1 so sánh giá trị s1 và s3 vì trong ví dụ này $s3 > s1 \rightarrow t2 = 0$

Lệnh bltz \$s1,NEGATIVE -> kiểm tra xem s1 có âm hay không -> ví dụ này dương -> không đến nhãn NEGATIVE

lệnh beq \$t2,\$0,EXIT kiểm tra xem $t2 = 0$ hay ko nếu $= 0$ nghĩa là s3 không bị tràn -> EXIT vì vậy $t2 = 0$

- TH khác dấu:

The screenshot shows a MIPS simulator interface. The main window displays assembly code with columns for Address, Code, Basic, and Source. The code includes instructions like `addiu $t0, $s0, 4508`, `addiu $s2, $s1, $s2`, `bltz $t1, EXIT`, and `bltz $t1, EXIT`. The right-hand side shows a register window with columns for Name, Number, and Value. The register `$s1` is highlighted with a value of -2019, and `$t1` is highlighted with a value of -5755. Below the main window, there is a table showing memory values at various addresses.

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

Trong TH này $s1 < 0$ nên sau khi thực hiện lệnh xor kết quả trong thanh ghi $t1 < 0$ vì vậy 2 số trái dấu và sẽ không xảy ra tràn -> lệnh `bltz $t1,EXIT` sẽ nhảy đến nhãn lệnh EXIT và kết thúc.

2. Assignment 2

- Mã nguồn:

#Laboratory Assignment 2

.text

`li $s0,0x20194508 # test value MSSV`

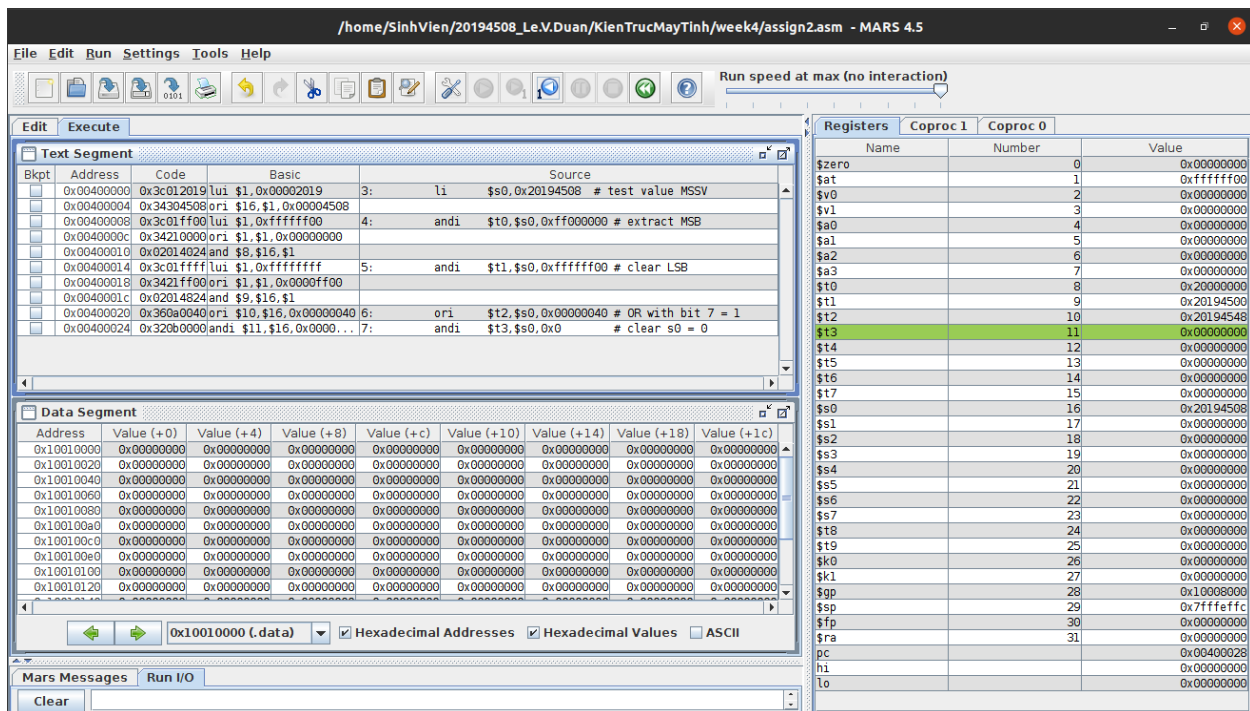
`andi $t0,$s0,0xff000000 # extract MSB`

`andi $t1,$s0,0xfffffff0 # clear LSB`

`ori $t2,$s0,0x00000040 # OR with bit 7 = 1`

`andi $t3,$s0,0x0 # clear s0 = 0`

- Kết quả chạy:



Thanh ghi t0 = 0x20000000 -> extract MSB

Thanh ghi t1 = 0x20194500 -> clear LSB

Thanh ghi t2 = 0x20194548 -> sau khi OR với dãy bit toàn 0 và bit thứ 7 = 1 (hex: 00000040)

Thanh ghi t3 = 0x00000000 -> reset s0 bằng cách and với 0.

3. Assignment 3

-Mã nguồn:

#Laboratory Assignment 3

.text

abs:

li \$s0, -38

bltz \$s0, NEGATIVE

nop

```
add $t0,$0,$s0
```

```
j EXIT
```

NEGATIVE:

```
nor $t0,$s0,$0 # t0 = not(s0)
```

```
add $t0,$t0,1
```

EXIT:

move:

```
li $s0,3
```

```
li $s1,8
```

```
add $s0,$0,$s1 # move s1 to s0 -> s0 = 8
```

not:

```
li $s0,4508
```

```
nor $t0,$s0,$0 # t0 = not(s0)
```

ble:

```
li $s1, -4508
```

```
li $s2, 308
```

```
sle $t0, $t1, $s2
```

```
beq $t0, 1, CASE
```

nop

```
addi $s2,$0,2
```

```
j EXIT
```

CASE:

```
addi $s1, $s1, 2
```

- Kết quả chạy mô phỏng:

+ abs:

Text Segment		Source		Registers				Coproc 1	Coproc 0
Bxpt	Address	Code	Basic	Name	Number		Value		
0x00400000	0x2410ffda	addiu \$t6,\$0,-38	4:	t6	\$a0, -38		0		
0x00400004	0x0040000d	bltz \$t6,3	5:	bltz	\$a0, NEGATIVE		1		
0x00400008	0x00000000	nop	6:	nop			2		
0x0040000c	0x0104020	add \$t0,\$0,\$t0	7:	add	\$t0,\$0,\$t0		3		
0x00400010	0x08100007	j EXIT	8:	j	EXIT		4		
0x00400014	0x02004027	nor \$t0,\$t6,\$0 # t0 = not(\$t0)	10:	nor	\$t0,\$a0,\$0 # t0 = not(\$a0)		5		
0x00400018	0x24108001	addi \$t6,\$t6,1	11:	add	\$t0,\$t0,1		6		
0x0040001c	0x24100001	addiu \$t6,\$t6,3	14:	t6	\$a0, 3		7		
0x00400020	0x24110003	addiu \$t7,\$t0,8	15:	t7	\$a1, 8		8		38
0x00400024	0x00118020	add \$t6,\$t6,\$t1 # move \$t1 to \$t0 -> \$t0 = \$t1	16:	add	\$a0,\$0,\$a1 # move \$a1 to \$a0 -> \$a0 = \$a1		9		
0x00400028	0x2410119c	addiu \$t6,\$t6,4508	18:	t6	\$a0, 4508		10		
0x0040002c	0x02004027	nor \$t0,\$t6,\$0 # t0 = not(\$t0)	19:	nor	\$t0,\$a0,\$0 # t0 = not(\$a0)		11		
0x00400030	0x2411ee4f	addiu \$t7,\$t6,-4508	21:	t7	\$a1, -4508		12		
0x00400034	0x24101134	addiu \$t8,\$t0,308	22:	t8	\$a0, 308		13		
0x00400038	0x0249402e	sllt \$t8,\$t8,\$t6	23:	sllt	\$t0, \$t1, \$a2		14		
0x0040003c	0x34010001	ori \$t1,\$t0,1					15		
0x00400040	0x00284023	muls \$t1,\$t0,\$t6					16		-38
0x00400044	0x02101001	addi \$t1,\$t0,1	24:	breq	\$t0, 1, CASE		17		
0x00400048	0x10280003	breq \$t1,\$t8,3					18		
0x0040004c	0x00000000	nop	25:	nop			19		
0x00400050	0x02102002	addi \$t0,\$t0,2	26:	addi	\$a2,\$0,2		20		
0x00400054	0x08100007	j EXIT	27:	j	EXIT		21		
0x00400058	0x22310002	addi \$t7,\$t1,2	29:	addi	\$a1, \$a1, 2		22		

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

☒ Hexadecimal Addresses
 ☐ Hexadecimal Values
 ☐ ASCII

- Nếu s0 là dương -> abs là chính nó. TH s0 âm -> abs là số đối của s0 -> tính not bằng lệnh nor và +1 để tính số bù 2 hay số đối

+ move:

Edit
Execute

Text Segment
Source

Bkpt	Address	Code	Basic
	0x00400000	0x2410ffff	addiu \$16,\$0,-38
	0x00400004	0x0000003b	lts \$16,3
	0x00400008	0x00000000	mop
	0x0040000c	0x00104020	add \$8,\$0,\$16
	0x00400010	0x00100007	j 0x0040001c
	0x00400014	0x02004027	nor \$8,\$16,\$0
	0x00400018	0x21080001	addi \$8,\$8,1
	0x0040001c	0x24100003	addiu \$16,\$0,3
	0x00400020	0x24100008	addiu \$17,\$0,8
	0x00400024	0x00118027	add \$16,\$0,\$17
	0x00400028	0x2410119c	addiu \$16,\$0,4508
	0x0040002c	0x02004027	nor \$8,\$16,\$0
	0x00400030	0x2411ee64	addiu \$17,\$0,-4508
	0x00400034	0x24101914	addiu \$18,\$0,308
	0x00400038	0x02484026	lts \$8,\$18,\$9
	0x0040003c	0x34010001	ori \$1,\$0,1
	0x00400040	0x00284023	subu \$8,\$1,\$8
	0x00400044	0x20010001	addi \$1,\$0,1
	0x00400048	0x0200003b	lts \$1,\$8,3
	0x0040004c	0x00000000	mop
	0x00400050	0x20120002	addi \$18,\$0,2
	0x00400054	0x00100007	0x0040001c
	0x00400058	0x23330002	addi \$17,\$17,2

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
\$a4	8	0
\$a5	9	0
\$a6	10	0
\$a7	11	0
\$a8	12	0
\$a9	13	0
\$a10	14	0
\$a11	15	0
\$a12	16	0
\$a13	17	8
\$a14	18	0
\$a15	19	0
\$a16	20	0
\$a17	21	0
\$a18	22	0
\$a19	23	0
\$a20	24	0
\$a21	25	0
\$a22	26	0
\$a23	27	0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)	Value (+10)	Value (+14)	Value (+18)	Value (+1C)
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

X10010000 (.data)
▼

☒ Hexadecimal Addresses
 ☐ Hexadecimal Values
 ☐ ASCII

X10010000 (.data)
▼

☒ Hexadecimal Addresses
 ☐ Hexadecimal Values
 ☐ ASCII

+ not: dùng lệnh nor để thay thế cho toán tử not

Edit Execute					Registers Coproc 1 Coproc 0			
Text Segment					Name	Number	Value	
Bkpt	Address	Code	Basic	Source				
	0x00400000	0x2410ffda	addiu \$16,\$0,-38	4: li \$a0, -38	\$zero	0	0	
	0x00400004	0xd4000003	bltz \$16,3	5: bltz \$a0, NEGATIVE	\$at	1	0	
	0x00400008	0x00000000	nop	6: nop	\$v0	2	0	
	0x0040000c	0x0104020	add \$8,\$0,\$16	7: add \$t0,\$0,\$a0	\$v1	3	0	
	0x00400010	0x00100007	j 0x0040001c	8: j EXIT	\$a0	4	0	
	0x00400014	0x02004027	nor \$8,\$16,\$0	10: nor \$t0,\$a0,\$0 # t0 = not(\$a0)	\$a1	5	0	
	0x00400018	0x21080001	addi \$8,\$8,1	11: add \$t0,\$t0,1	\$a2	6	0	
	0x0040001c	0x24100003	addiu \$16,\$0,3	14: li \$a0,3	\$a3	7	0	
	0x00400020	0x24110008	addiu \$17,\$0,8	15: li \$a1,8	\$t0	8	-4508	
	0x00400024	0x00110020	add \$16,\$0,\$17	16: add \$a0,\$0,\$a1 # move \$1 to \$0 -> \$0 = 8	\$t1	9	0	
	0x00400028	0x241019c	addiu \$16,\$0,4508	18: li \$a0,4508	\$t2	10	0	
	0x0040002c	0x02004027	nor \$8,\$16,\$0	19: nor \$t0,\$a0,\$0 # t0 = not(\$a0)	\$t3	11	0	
	0x00400030	0x2411ee64	addiu \$17,\$0,-4508	21: li \$a1,-4508	\$t4	12	0	
	0x00400034	0x24120134	addiu \$18,\$0,308	22: li \$a2,308	\$t5	13	0	
	0x00400038	0x0249402a	sllt \$8,\$18,\$9	23: sllt \$t0,\$t1,\$a2	\$t6	14	0	
	0x0040003c	0x34010001	ori \$1,\$0,1		\$t7	15	0	
	0x00400040	0x00284023	sltu \$8,\$8,\$8		\$a0	16	4508	
	0x00400044	0x20010001	addi \$1,\$0,1	24: beq \$t0,1,CASE	\$a1	17	8	
	0x00400048	0x10280003	beq \$1,\$8,3		\$a2	18	0	
	0x0040004c	0x00000000	nop	25: nop	\$a3	19	0	
	0x00400050	0x20120002	addi \$18,\$0,2	26: addi \$a2,\$0,2	\$a4	20	0	
	0x00400054	0x00100007	j 0x0040001c	27: j EXIT	\$a5	21	0	
	0x00400058	0x22310002	addi \$17,\$17,2	29: addi \$a1,\$a1,2	\$a6	22	0	
					\$a7	23	0	
					\$a8	24	0	
					\$t9	25	0	
					\$a0	26	0	
					\$k1	27	0	
					\$BP	28	268460224	
					\$SP	29	2147479548	
					\$FP	30	0	
					\$ra	31	0	
					PC		4194382	
					hi		0	
					lo		0	

+ ble:

Edit Execute					Registers Coproc 1 Coproc 0			
Text Segment					Name	Number	Value	
Bkpt	Address	Code	Basic	Source				
	0x00400000	0x2410ffda	addiu \$16,\$0,-38	4: li \$a0, -38	\$zero	0	0	
	0x00400004	0xd4000003	bltz \$16,3	5: bltz \$a0, NEGATIVE	\$at	1	1	
	0x00400008	0x00000000	nop	6: nop	\$v0	2	0	
	0x0040000c	0x0104020	add \$8,\$0,\$16	7: add \$t0,\$0,\$a0	\$v1	3	0	
	0x00400010	0x00100007	j 0x0040001c	8: j EXIT	\$a0	4	0	
	0x00400014	0x02004027	nor \$8,\$16,\$0	10: nor \$t0,\$a0,\$0 # t0 = not(\$a0)	\$a1	5	0	
	0x00400018	0x21080001	addi \$8,\$8,1	11: add \$t0,\$t0,1	\$a2	6	0	
	0x0040001c	0x24100003	addiu \$16,\$0,3	14: li \$a0,3	\$a3	7	0	
	0x00400020	0x24110008	addiu \$17,\$0,8	15: li \$a1,8	\$t0	8	1	
	0x00400024	0x00110020	add \$16,\$0,\$17	16: add \$a0,\$0,\$a1 # move \$1 to \$0 -> \$0 = 8	\$t1	9	0	
	0x00400028	0x241019c	addiu \$16,\$0,4508	18: li \$a0,4508	\$t2	10	0	
	0x0040002c	0x02004027	nor \$8,\$16,\$0	19: nor \$t0,\$a0,\$0 # t0 = not(\$a0)	\$t3	11	0	
	0x00400030	0x2411ee64	addiu \$17,\$0,-4508	21: li \$a1,-4508	\$t4	12	0	
	0x00400034	0x24120134	addiu \$18,\$0,308	22: li \$a2,308	\$t5	13	0	
	0x00400038	0x0249402a	sllt \$8,\$18,\$9	23: sllt \$t0,\$t1,\$a2	\$t6	14	0	
	0x0040003c	0x34010001	ori \$1,\$0,1		\$t7	15	0	
	0x00400040	0x00284023	sltu \$8,\$8,\$8		\$a0	16	4508	
	0x00400044	0x20010001	addi \$1,\$0,1	24: beq \$t0,1,CASE	\$a1	17	-4508	
	0x00400048	0x10280003	beq \$1,\$8,3		\$a2	18	308	
	0x0040004c	0x00000000	nop	25: nop	\$a3	19	0	
	0x00400050	0x20120002	addi \$18,\$0,2	26: addi \$a2,\$0,2	\$a4	20	0	
	0x00400054	0x00100007	j 0x0040001c	27: j EXIT	\$a5	21	0	
	0x00400058	0x22310002	addi \$17,\$17,2	29: addi \$a1,\$a1,2	\$a6	22	0	
					\$a7	23	0	
					\$t9	24	0	
					\$a0	25	0	
					\$k1	26	0	
					\$BP	27	0	
					\$SP	28	268460224	
					\$FP	29	2147479548	
					\$ra	30	0	
					PC	31	0	
					hi		4194396	
					lo		0	

4. Assignment 4

- Mã nguồn:

#Laboratory Assignment 4

.text

li \$s0, 0x8ffffff

li \$s1, 0x80194508

start:

```
li    $t0, 0        # default status
```

```
add   $s3, $s1, $s2    # s3 = s1 + s2
```

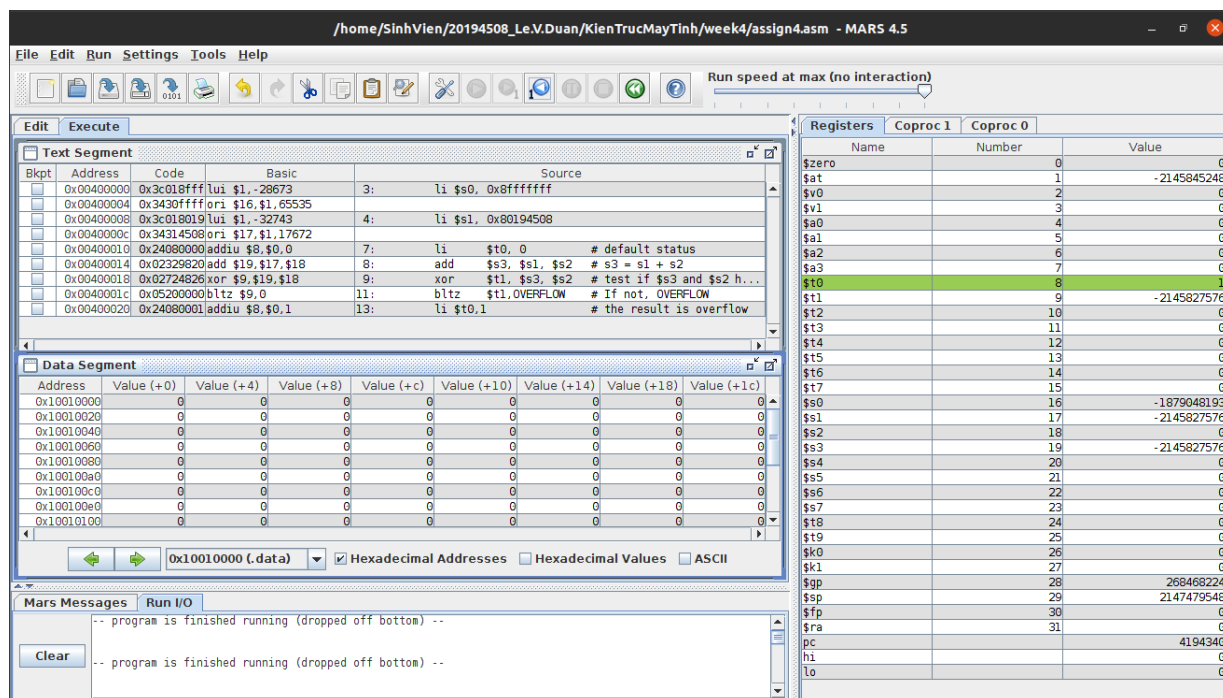
```
xor   $t1, $s3, $s2    # test if $s3 and $s2 have the same sign
```

```
bltz  $t1, OVERFLOW   # If not, OVERFLOW
```

OVERFLOW:

```
li    $t0, 1        # the result is overflow
```

- Kết quả chạy:



5. Assignment 5

- Mã nguồn:

#Laboratory Assignment 5

.text


```

li    $s0, 38

sll   $t1,$s0,1    # s0 * (2^1)

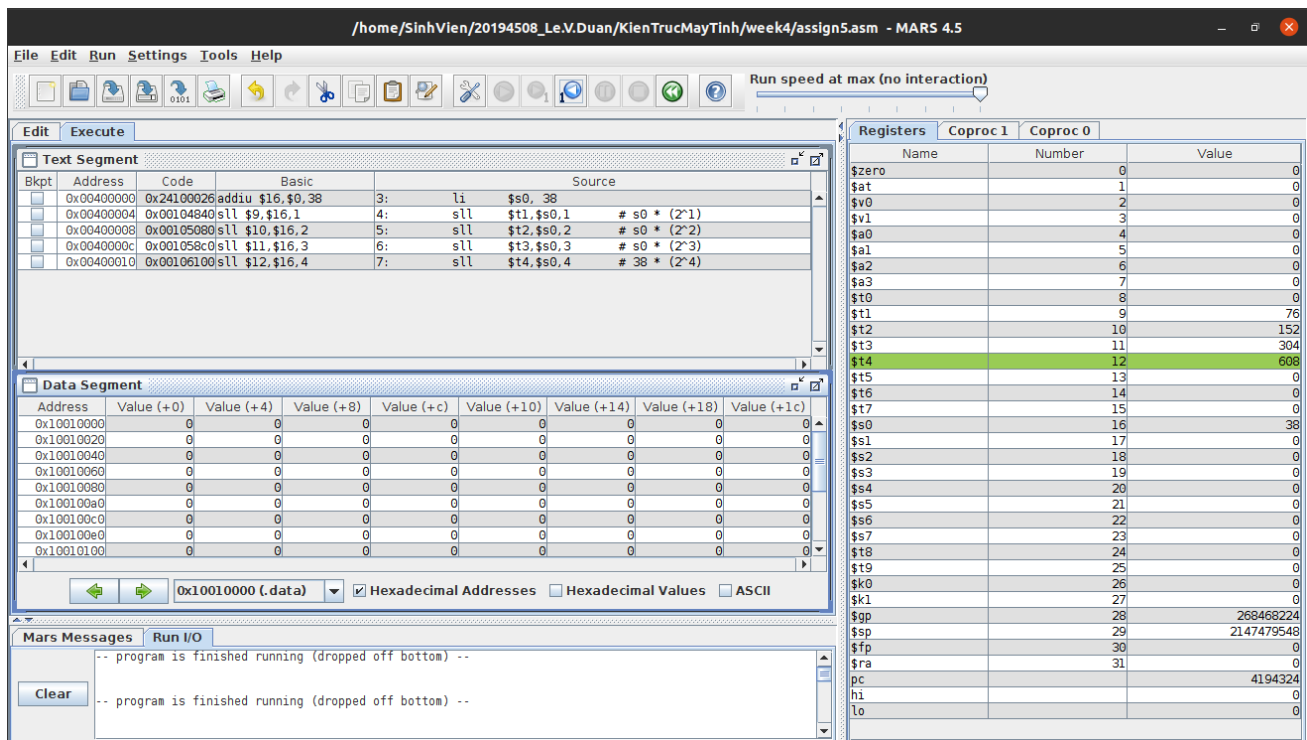
sll   $t2,$s0,2    # s0 * (2^2)

sll   $t3,$s0,3    # s0 * (2^3)

sll   $t4,$s0,4    # 38 * (2^4)

```

- Kết quả chạy:



- Nhân một số với lũy thừa với 2 bằng cách dịch bit

$$t1 = 38 * 2 = 76$$

$$t2 = 38 * 4 = 152$$

$$t3 = 38 * 8 = 304$$

$$t4 = 38 * 16 = 608$$