

Bài 4. Các lệnh số học và logic

Họ và tên: Nguyễn Thành Duy

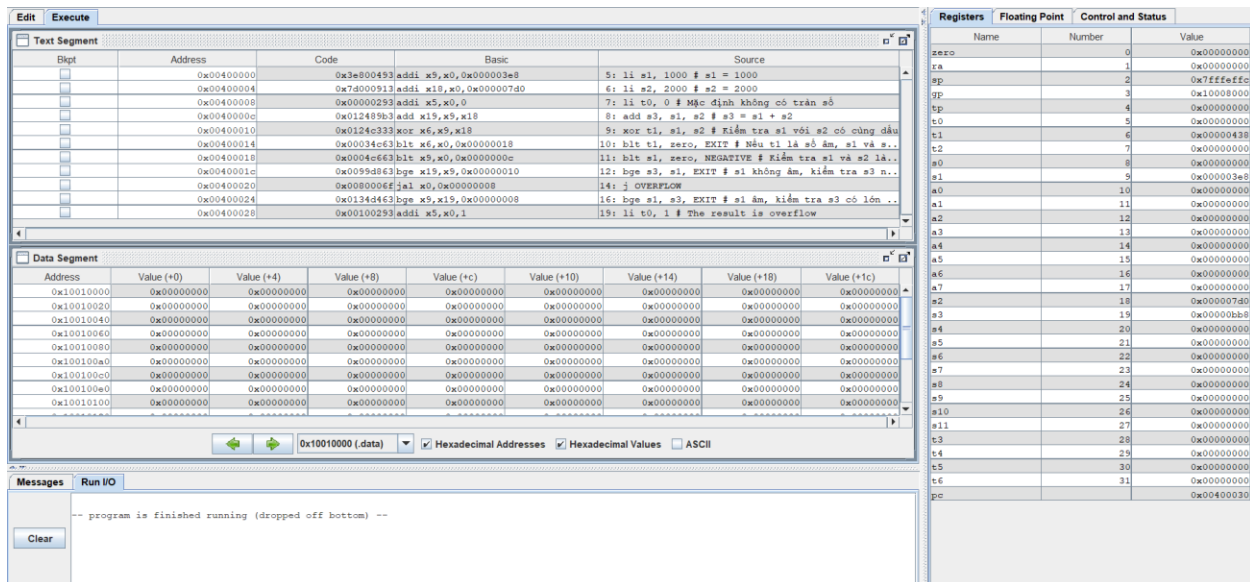
MSSV: 20235696

Assignment 1

1. Trường hợp 2 số dương và tổng không tràn dấu

Code:

```
# Laboratory Exercise 4, Home Assignment 1
.text
# TODO: Thiết lập giá trị cho s1 và s2 với trường hợp khác nhau
# Thuật toán xác định tràn số
li s1, 1000 # s1 = 1000
li s2, 2000 # s2 = 2000
li t0, 0 # Mặc định không có tràn số
add s3, s1, s2 # s3 = s1 + s2
xor t1, s1, s2 # Kiểm tra s1 với s2 có cùng dấu
blt t1, zero, EXIT # Nếu t1 là số âm, s1 và s2 khác dấu
blt s1, zero, NEGATIVE # Kiểm tra s1 và s2 là số âm hay không âm
bge s3, s1, EXIT # s1 không âm, kiểm tra s3 nhỏ hơn s1 không
# Nếu s3 >= s1, không tràn số
j OVERFLOW
NEGATIVE:
bge s1, s3, EXIT # s1 âm, kiểm tra s3 có lớn hơn s1 không
# Nếu s1 >= s3, không tràn số
OVERFLOW:
li t0, 1 # The result is overflow
EXIT:
```



Kết quả: Giá trị thanh ghi t0 = 0 -> không xảy ra hiện tượng tràn số thì kết quả tổng s3 = s1 + s2 = 1000 + 2000 = 3000 (Đúng như dự đoán)

2. Trường hợp 2 số âm, tổng không tràn số

Code:

```
# Laboratory Exercise 4, Home Assignment 1
.text
# TODO: Thiết lập giá trị cho s1 và s2 với trường hợp khác nhau
# Thuật toán xác định tràn số
li s1, -1000 # s1 = 1000
li s2, -2000 # s2 = 2000
li t0, 0 # Mặc định không có tràn số
add s3, s1, s2 # s3 = s1 + s2
xor t1, s1, s2 # Kiểm tra s1 với s2 có cùng dấu
blt t1, zero, EXIT # Nếu t1 là số âm, s1 và s2 khác dấu
blt s1, zero, NEGATIVE # Kiểm tra s1 và s2 là số âm hay không âm
bge s3, s1, EXIT # s1 không âm, kiểm tra s3 nhỏ hơn s1 không
# Nếu s3 >= s1, không tràn số
j OVERFLOW
NEGATIVE:
bge s1, s3, EXIT # s1 âm, kiểm tra s3 có lớn hơn s1 không
# Nếu s1 >= s3, không tràn số
OVERFLOW:
li t0, 1 # The result is overflow
EXIT:
```

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0xc1800493	addl x9,x0,0xffffffffc18	5: li s1, -1000 \$ s1 = 1000
<input type="checkbox"/>	0x00400004	0xb3000913	addl r10,r0,0xffffffffb30	6: li s2, -2000 \$ s2 = 2000
<input type="checkbox"/>	0x00400008	0x00000293	addl r5,x0,0	7: li t0, 0 # Mặc định không có tràn số
<input type="checkbox"/>	0x0040000c	0x012499b3	add x15,x9,x10	8: add s3, s1, s2 \$ s3 = s1 + s2
<input type="checkbox"/>	0x00400010	0x0124e333	xor x9,x9,x18	9: xor t1, s1, s2 # Kiểm tra s1 với s2 có cùng dấu
<input type="checkbox"/>	0x00400014	0x00124ef3	btl x9,x0,0x00000018	10: btl t1, zero, EXIT # Nếu t1 là s0 âm, s1 và s2...
<input type="checkbox"/>	0x00400018	0x000a563b	btl x9,x0,0x0000000c	11: btl s1, zero, NEGATIVE # Kiểm tra s1 và s2 là...
<input type="checkbox"/>	0x0040001c	0x00994d63	bge r15,x9,0x00000010	12: bge s3, s1, EXIT # s1 không âm, kiểm tra s3 n...
<input type="checkbox"/>	0x00400020	0x0080000f	jnl x0,0x00000008	14: j OVERFLOW
<input type="checkbox"/>	0x00400024	0x0134d463	bge x15,x19,0x00000008	16: bge s1, s3, EXIT # s1 âm, kiểm tra s3 có lớn ...
<input type="checkbox"/>	0x00400028	0x00100293	addl x5,x0,1	19: li t0, 1 # The result is overflow

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

☒ Run I/O
☒ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

Messages

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

Clear

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

3. Trường hợp 2 số dương và bị tràn số

```
# Laboratory Exercise 4, Home Assignment 1
.text
# TODO: Thiết lập giá trị cho s1 và s2 với trường hợp khác nhau
# Thuật toán xác định tràn số
li s1, 0x7FFFFFFF # Load max signed 32-bit int
li s2, 1 # s2 = 1
li t0, 0 # Mặc định không có tràn số
add s3, s1, s2 # s3 = s1 + s2
xor t1, s1, s2 # Kiểm tra s1 với s2 có cùng dấu
blt t1, zero, EXIT # Nếu t1 là số âm, s1 và s2 khác dấu
blt s1, zero, NEGATIVE # Kiểm tra s1 và s2 là số âm hay không âm
bge s3, s1, EXIT # s1 không âm, kiểm tra s3 nhỏ hơn s1 không
# Nếu s3 >= s1, không tràn số
j OVERFLOW
NEGATIVE:
bge s1, s3, EXIT # s1 âm, kiểm tra s3 có lớn hơn s1 không
# Nếu s1 >= s3, không tràn số
OVERFLOW:
li t0, 1 # The result is overflow
EXIT:
```


Code:

```
.text
li s0, 0x12345678
li s1, 0x12345600
and s2, s0, s1
```

Text Segment

Offset	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x12345678 lui x8, 0x00012345		2: li s0, 0x12345678
<input type="checkbox"/>	0x00400004	0x67890123 addi x9, x8, 0x00000678		
<input type="checkbox"/>	0x00400008	0x12345678 lui x9, 0x00012345		3: li s1, 0x12345600
<input type="checkbox"/>	0x0040000c	0x60048493 addi x9, x9, 0x00000600		
<input type="checkbox"/>	0x00400010	0x00947933 and x18, x8, x9		4: and s2, s0, s1

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

0x10010000 (data)

☒ Hexadecimal Addresses

☒ Hexadecimal Values

☐ ASCII

Messages

Run I/O

Clear

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

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

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffeffc
fp	3	0x10000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x12345678
s1	9	0x12345600
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x12345600
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
s12	28	0x00000000
s13	29	0x00000000
s14	30	0x00000000
s15	31	0x00000000
pc		0x00400018

Kết quả: s2 = 0x12345600

3. Thiết lập LSB của thanh ghi s0 (bit 7 đến bit 0 được thiết lập bằng 1)

Code:

```
.text
li s0, 0x12345678 # Load immediate value
ori s0, s0, 0xFF # Set LSB (bit 7-0) to 1
```


The screenshot shows a debugger interface with two main panels. The left panel displays assembly code in a table with columns for Address, Code, Basic, and Source. The right panel shows the state of registers, with register s0 highlighted and its value set to 0x123456ff.

Text Segment	Address	Code	Basic	Source
	0x00400000	0x12345437	lui s0, 0x00012345	2: li s0, 0x12345678 # Load immediate value
	0x00400004	0x67840413	addi s0, s0, 0x00000678	
	0x00400008	0x00ff4641	ori s0, s0, 0x000000ff	3: ori s0, s0, 0xff # Set LSB (bit 7-0) to 1

Registers	Name	Number	Value
zero		0	0x00000000
ra		1	0x00000000
sp		2	0x7ffffefc
fp		3	0x10008000
tp		4	0x00000000
t0		5	0x00000000
t1		6	0x00000000
t2		7	0x00000000
s0		8	0x123456ff
s1		9	0x00000000
a0		10	0x00000000
a1		11	0x00000000
a2		12	0x00000000
a3		13	0x00000000
a4		14	0x00000000
a5		15	0x00000000
a6		16	0x00000000
a7		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
s10		26	0x00000000
s11		27	0x00000000
t3		28	0x00000000
t4		29	0x00000000
t5		30	0x00000000
t6		31	0x00000000
pc			0x00400010

Kết quả: Giá trị thanh ghi s0 = 0x123456ff (với bit 7 đến bit 0 được thiết lập bằng 1.)

4. Xóa thanh ghi s0 bằng cách dùng các lệnh logic (s0 = 0)

Code:

```
.text
li s0, 0x12345678 # Load immediate value
xor s0, s0, s0 # Set LSB (bit 7-0) to 1
```

The screenshot shows the same debugger interface after executing the new code. The assembly code now includes an XOR instruction that clears the value in register s0. The registers window shows that register s0 now contains the value 0x00000000.

Text Segment	Address	Code	Basic	Source
	0x00400000	0x12345437	lui s0, 0x00012345	2: li s0, 0x12345678 # Load immediate value
	0x00400004	0x67840413	addi s0, s0, 0x00000678	
	0x00400008	0x00ff4641	ori s0, s0, 0x000000ff	3: ori s0, s0, 0xff # Set LSB (bit 7-0) to 1
	0x0040000c	0x00000000	xor s0, s0, s0	4: xor s0, s0, s0 # Set s0 to 0

Registers	Name	Number	Value
zero		0	0x00000000
ra		1	0x00000000
sp		2	0x7ffffefc
fp		3	0x10008000
tp		4	0x00000000
t0		5	0x00000000
t1		6	0x00000000
t2		7	0x00000000
s0		8	0x00000000
s1		9	0x00000000
a0		10	0x00000000
a1		11	0x00000000
a2		12	0x00000000
a3		13	0x00000000
a4		14	0x00000000
a5		15	0x00000000
a6		16	0x00000000
a7		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
s10		26	0x00000000
s11		27	0x00000000
t3		28	0x00000000
t4		29	0x00000000
t5		30	0x00000000
t6		31	0x00000000
pc			0x00400010

Kết quả: Thanh ghi s0 bị xóa về 0

Assignment 3

a. neq s0, s1

```
.text
addi s1, s1, -100
sub s0, zero, s1
```

The screenshot shows the QtConsole IDE with the following components:

- Text Segment:** Displays assembly instructions:

Bkpt	Address	Code	Basic	Source
	0x00400000	0xffff00433	addi x5,x0,0xffffffff	2: li s1, -3
	0x00400004	0x40900433	sub x8,x0,x9	3: neg s0, s1
- Data Segment:** A table of memory addresses and their values (all are 0x00000000).
- Registers:** A table of registers and their values:

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0xffffffff
fp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0x00000003
s1	9	0xffffffff
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
a8	18	0x00000000
a9	19	0x00000000
s2	20	0x00000000
s3	21	0x00000000
s4	22	0x00000000
s5	23	0x00000000
s6	24	0x00000000
s7	25	0x00000000
s8	26	0x00000000
s9	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040000c
- Messages:** Shows the message "program is finished running (dropped off bottom)".

b. mv s0, s1

```
.text
addi s1, s1, -100
add s0, zero, s1
```

The screenshot shows the QtConsole IDE with the following components:

- Text Segment:** Displays assembly instructions:

Bkpt	Address	Code	Basic	Source
	0x00400000	0xf9c48493	addi x5,x5,0xffffffff5c	2: addi s1, s1, -100
	0x00400004	0x00900433	add x8,x0,x9	3: add s0, zero, s1
- Data Segment:** A table of memory addresses and their values (all are 0x00000000).
- Registers:** A table of registers and their values:

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0xffffffff
fp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0xffffffff
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
a8	18	0x00000000
a9	19	0x00000000
s2	20	0x00000000
s3	21	0x00000000
s4	22	0x00000000
s5	23	0x00000000
s6	24	0x00000000
s7	25	0x00000000
s8	26	0x00000000
s9	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040000c
- Messages:** Shows the message "program is finished running (dropped off bottom)".

c. Not s0

Code:

```
.text
    addi s0, s0, -1
    xori s1, s0, 0xffffffff
```

```
addi s0, s0, -1
xori s1, s0, 0xffffffff
```

Edit Execute
Registers Floating Point Control and Status

Text Segment

Blkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0xffff40413	addi x8,x8,0xffffffff	2: addi x0, x0, -1
<input type="checkbox"/>	0x00400004	0xffff44493	xori x9,x8,0xffffffff	3: xori x1, x0, 0xffffffff

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

0x10010000 (data) ☒ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

Registers

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7ffffefc
gp	3	0x10000000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000000
s0	8	0xffffffff
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
a8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
s3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400008

Messages **Run IO**

```

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

Clear

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

```

Kết quả: Các bit đã bị đảo 1- \rightarrow 0

d. `ble s1, s2, label`

Code:

```
.text
li s1, 3
li s2, 4
bge s2, s1, label
add s2, s1, s1
label:
add s2, s1, zero
```

The screenshot shows a debugger window with two main panes. The left pane displays assembly code with columns for Address, Code, Basic, and Source. The right pane shows the state of registers, with the 'Registers' tab selected. The register 's2' is highlighted in green, showing a value of 0x00000003. The 'Messages' pane at the bottom shows a message: '-- program is finished running (dropped off bottom) --'.

Register	Name	Number	Value
zero		0	0x00000000
ra		1	0x00000000
sp		2	0x7ffffefc
fp		3	0x10008000
tp		4	0x00000000
t0		5	0x00000000
t1		6	0x00000000
t2		7	0x00000000
a0		8	0x00000000
a1		9	0x00000000
a2		10	0x00000000
a3		11	0x00000000
a4		12	0x00000000
a5		13	0x00000000
a6		14	0x00000000
a7		15	0x00000000
s2		16	0x00000003
s3		17	0x00000000
s4		18	0x00000000
s5		19	0x00000000
s6		20	0x00000000
s7		21	0x00000000
s8		22	0x00000000
s9		23	0x00000000
s10		24	0x00000000
s11		25	0x00000000
s12		26	0x00000000
s13		27	0x00000000
s14		28	0x00000000
s15		29	0x00000000
s16		30	0x00000000
pc		31	0x00400014

Kết quả: $s2 = s1 = 3$ do $s1 \leq s2$ nên nhảy tới label để thực hiện. Nếu ngược lại $s1 > s2$ thì sẽ thực hiện $s2 = 2s1 = 4$.

Assignment 4

1. Trường hợp 2 số trái dấu

Code:

```
.text
li s1, 2
li s2, -3
li t0, 0

add s3, s1, s2
xor t1, s1, s2
blt t1, zero, EXIT
xor t2, s1, s3
bge t2, zero, EXIT
li t0, 1
```

EXIT:

The screenshot shows a debugger window with three main panes. The top pane displays assembly code with columns for Disasm, Address, Code, Basic, and Source. The middle pane shows the Data Segment with columns for Address and various Value (+offset) fields. The bottom pane shows the Messages window with a 'Run I/O' button and a message: '-- program is finished running(dropped off bottom) --'. On the right side, there is a 'Registers' pane showing the state of various registers, including zero, ra, sp, gp, tp, t0, t1, t2, a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, t3, t4, t5, t6, and pc. The 't0' register is highlighted in green, and its value is 0.

Kết quả: t0 = 0 -> không xảy ra hiện tượng tràn số.

2. Trường hợp 2 số cùng dấu tràn số

Code:

```
.text
    li s1, 0x7FFFFFFF
    li s2, 1
    li t0, 0

    add s3, s1, s2
    xor t1, s1, s2
    blt t1, zero, EXIT
    xor t2, s1, s3
    bge t2, zero, EXIT
    li t0, 1
```

EXIT:

The screenshot shows a debugger window with three main panes. The top pane displays assembly code with columns for Bkpt, Address, Code, Basic, and Source. The middle pane shows the Data Segment with columns for Address and various Value (+) offsets. The bottom pane shows the Registers window with columns for Name, Number, and Value. The assembly code includes instructions like `li s1, 0x7fffffff`, `add s3, s1, s2`, `blt t1, zero, EXIT`, and `bge t2, zero, EXIT`. The registers window shows the state of various registers, including `s1` and `s2`.

Kết quả: `t0 = 1` -> xảy ra hiện tượng tràn số

3. Trường hợp 2 số âm tràn số

Code:

.text

```
li s1, 0x80000000
li s2, -1
li t0, 0
```

```
add s3, s1, s2
xor t1, s1, s2
blt t1, zero, EXIT
xor t2, s1, s3
bge t2, zero, EXIT
li t0, 1
```

EXIT:

The screenshot displays a MIPS simulator interface with three main panels:

- Text Segment:** Shows assembly code with columns for Bkpt, Address, Code, Basic, and Source. The code includes instructions like `lui $9, 0xffff0000`, `addi $9, $9, 0`, `addi $18, $0, 0xffffffff`, `addi $5, $0, 0`, `add $3, $1, $2`, `xor $1, $1, $2`, `bit $1, zero, EXIT`, `xor $2, $1, $3`, `bge $2, zero, EXIT`, and `addi $5, $0, 1`.
- Data Segment:** A table with columns for Address and various Value (+0) to Value (+1c) slots. All values are currently 0x00000000.
- Registers:** A table listing registers from \$zero to \$pc, their numbers (0-31), and their values. Most registers contain 0x00000000, while \$pc contains 0x0040002e.

At the bottom, there is a Messages panel showing the message: "program is finished running(dropped off bottom)".

Kết quả: $t_0 = 1 \rightarrow$ Xảy ra hiện tượng tràn số

Assignment 5

.text

li \$1, 6

li \$2, 8

li \$t0, 1

loop:

beq \$2, \$t0, endloop

sll \$1, \$1, \$t0

srl \$2, \$2, \$t0

j loop

endloop:

