

Báo cáo Thực hành KTMT buổi 4

Họ và tên: Nguyễn Đức Phú

MSSV: 20215116

Assignment 1:

- Trường hợp 1: Cộng hai số trái dấu

```
1 #Laboratory Exercise 4, Home Assignment 1
2 .text
3     li $s1, 123
4     li $s2, -321
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           #the result is overflow
20    EXIT:
21
```

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	0xffffffffc4
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x0000007b
\$s2	18	0xffffffffbf
\$s3	19	0xffffffff3a
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000

- Trường hợp 2: Cộng hai số dương không tràn

```
1 #Laboratory Exercise 4, Home Assignment 1
2 .text
3     li $s1, 123
4     li $s2, 321
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           #the result is overflow
20    EXIT:
21
```

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	0x0000013a
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x0000007b
\$s2	18	0x00000141
\$s3	19	0x000001bc
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000

• Trường hợp 3: Cộng hai số dương có tràn số

A3.asm	A5.asm	HA1.asm	Name	Number	Value
1	#Laboratory Exercise 4, Home Assignment 1		\$zero	0	0x00000000
2	.text		\$at	1	0x7fff0000
3	li \$s1, 0x7fffffff		\$v0	2	0x00000000
4	li \$s2, 1		\$v1	3	0x00000000
5	start:		\$a0	4	0x00000000
6	li \$t0, 0	#No Overflow is default status	\$a1	5	0x00000000
7	addu \$s3, \$s1, \$s2	# s3 = s1 + s2	\$a2	6	0x00000000
8	xor \$t1, \$s1, \$s2	#Test if \$s1 and \$s2 have the same sign	\$a3	7	0x00000000
9	bltz \$t1, EXIT	#If not, exit	\$t0	8	0x00000001
10	slt \$t2, \$s3, \$s1		\$t1	9	0x7fffffff
11	bltz \$s1, NEGATIVE	#Test if \$s1 and \$s2 is negative?	\$t2	10	0x00000001
12	beq \$t2, \$zero, EXIT	#s1 and \$s2 are positive	\$t3	11	0x00000000
13	# if \$s3 > \$s1 then the result is not overflow		\$t4	12	0x00000000
14	j OVERFLOW		\$t5	13	0x00000000
15	NEGATIVE:		\$t6	14	0x00000000
16	bne \$t2, \$zero, EXIT	#s1 and \$s2 are negative	\$t7	15	0x00000000
17	# if \$s3 < \$s1 then the result is not overflow		\$s0	16	0x00000000
18	OVERFLOW:		\$s1	17	0x7fffffff
19	li \$t0, 1	#the result is overflow	\$s2	18	0x00000001
20	EXIT:		\$s3	19	0x80000000
21			\$s4	20	0x00000000
			\$s5	21	0x00000000
			\$s6	22	0x00000000

• Trường hợp 4: Cộng hai số âm không tràn

A3.asm	A5.asm	HA1.asm	Name	Number	Value
1	#Laboratory Exercise 4, Home Assignment 1		\$zero	0	0x00000000
2	.text		\$at	1	0x00000000
3	li \$s1, -321		\$v0	2	0x00000000
4	li \$s2, -123		\$v1	3	0x00000000
5	start:		\$a0	4	0x00000000
6	li \$t0, 0	#No Overflow is default status	\$a1	5	0x00000000
7	addu \$s3, \$s1, \$s2	# s3 = s1 + s2	\$a2	6	0x00000000
8	xor \$t1, \$s1, \$s2	#Test if \$s1 and \$s2 have the same sign	\$a3	7	0x00000000
9	bltz \$t1, EXIT	#If not, exit	\$t0	8	0x00000000
10	slt \$t2, \$s3, \$s1		\$t1	9	0x0000013a
11	bltz \$s1, NEGATIVE	#Test if \$s1 and \$s2 is negative?	\$t2	10	0x00000001
12	beq \$t2, \$zero, EXIT	#s1 and \$s2 are positive	\$t3	11	0x00000000
13	# if \$s3 > \$s1 then the result is not overflow		\$t4	12	0x00000000
14	j OVERFLOW		\$t5	13	0x00000000
15	NEGATIVE:		\$t6	14	0x00000000
16	bne \$t2, \$zero, EXIT	#s1 and \$s2 are negative	\$t7	15	0x00000000
17	# if \$s3 < \$s1 then the result is not overflow		\$s0	16	0x00000000
18	OVERFLOW:		\$s1	17	0xfffffebf
19	li \$t0, 1	#the result is overflow	\$s2	18	0xfffffff85
20	EXIT:		\$s3	19	0xffffffe44
21			\$s4	20	0x00000000
			\$s5	21	0x00000000
			\$s6	22	0x00000000

• Trường hợp 5: Cộng hai số âm có tràn số

A3.asm	A5.asm	HA1.asm	Name	Number	Value
1	#Laboratory Exercise 4, Home Assignment 1		\$zero	0	0x00000000
2	.text		\$at	1	0x80000000
3	li \$s1, 0x80000000		\$v0	2	0x00000000
4	li \$s2, -1		\$v1	3	0x00000000
5	start:		\$a0	4	0x00000000
6	li \$t0, 0	#No Overflow is default status	\$a1	5	0x00000000
7	addu \$s3, \$s1, \$s2	# s3 = s1 + s2	\$a2	6	0x00000000
8	xor \$t1, \$s1, \$s2	#Test if \$s1 and \$s2 have the same sign	\$a3	7	0x00000000
9	bltz \$t1, EXIT	#If not, exit	\$t0	8	0x00000001
10	slt \$t2, \$s3, \$s1		\$t1	9	0x7fffffff
11	bltz \$s1, NEGATIVE	#Test if \$s1 and \$s2 is negative?	\$t2	10	0x00000000
12	beq \$t2, \$zero, EXIT	#s1 and \$s2 are positive	\$t3	11	0x00000000
13	# if \$s3 > \$s1 then the result is not overflow		\$t4	12	0x00000000
14	j OVERFLOW		\$t5	13	0x00000000
15	NEGATIVE:		\$t6	14	0x00000000
16	bne \$t2, \$zero, EXIT	#s1 and \$s2 are negative	\$t7	15	0x00000000
17	# if \$s3 < \$s1 then the result is not overflow		\$s0	16	0x00000000
18	OVERFLOW:		\$s1	17	0x80000000
19	li \$t0, 1	#the result is overflow	\$s2	18	0xffffffff
20	EXIT:		\$s3	19	0x7fffffff
21			\$s4	20	0x00000000
			\$s5	21	0x00000000
			\$s6	22	0x00000000

Assignment 2:

- **Code:**

```
.text
li $s0, 0x87654321          # example number
andi $t1, $s0, 0xff000000
srl $t1, $t1, 24             # extract MSB
andi $t2, $s0, 0xffffffff00 # clear LSB
or $t3, $s0, 0x000000ff      # set LSB
xor $s0, $s0, $s0            # clear $s0
```

- **Giải thích:**

- Logic AND từng bit có đặc điểm: and với bit 1 thì kết quả là chính nó, and bit 0 kết quả luôn là 0
 - ⇒ Lệnh andi với 0xff000000 sẽ giữ lại 8 bit cao (and với 0xff) và loại bỏ các bit còn lại (and với 0x000000) => Trích xuất MSB
 - ⇒ Lệnh andi với 0xffffffff00 hoạt động tương tự và clear LSB
- Logic OR từng bit có đặc điểm: or với bit 1 kết quả luôn là 1, or với bit 0 kết quả là chính nó
 - ⇒ Set LSB
- Logic XOR từng bit có đặc điểm: xor 2 bit trùng nhau luôn cho ra kết quả 0
 - ⇒ Lệnh xor \$s0 với chính nó sẽ làm giá trị tất cả các bit trở về 0
 - ⇒ Clear \$s0

- **Kết quả khi chạy toàn bộ code:**

Edit Execute			Registers Coproc 1 Coproc 0		
A3.asm	A5.asm	HA1.asm	Name	Number	Value
1	.text		\$zero	0	0x00000000
2	li \$s0, 0x87654321 # example number		\$at	1	0xffffffff00
3	andi \$t1, \$s0, 0xff000000		\$v0	2	0x00000000
4	srl \$t1, \$t1, 24 # extract MSB		\$v1	3	0x00000000
5	andi \$t2, \$s0, 0xffffffff00 # clear LSB		\$a0	4	0x00000000
6	or \$t3, \$s0, 0x000000ff # set LSB		\$a1	5	0x00000000
7	xor \$s0, \$s0, \$s0 # clear \$s0		\$a2	6	0x00000000
8			\$a3	7	0x00000000
9			\$t0	8	0x00000000
			\$t1	9	0x00000087
			\$t2	10	0x87654300
			\$t3	11	0x876543ff
			\$t4	12	0x00000000
			\$t5	13	0x00000000
			\$t6	14	0x00000000
			\$t7	15	0x00000000
			\$s0	16	0x00000000
			\$s1	17	0x00000000

- ⇒ Các kết quả extract MSB, clear LSB, set LSB, lần lượt được lưu tại \$t1, \$t2, \$t3. Giá trị \$s0 đã clear và trở thành 0x00000000

Assignment 3:

a. `abs $s0, $s1:`

```
sra $t0, $s1, 31
xor $s0, $t0, $s1
subu $s0, $s0, $t0
```

- ⇒ Bit của `$t0` được điền toàn bộ bởi bit đầu của `$s1`
- ⇒ Nếu bit đầu là 1 lệnh XOR đảo bit toàn bộ `$s1`
 - Giá trị của `$s0` sẽ là số $(- \$s1 - 1)$
 - Lệnh `subu $s0` với `$t0` (`0xffffffff = -1`) sẽ là $(- \$s1 - 1) - 1$
 - Kết quả `$s0` thu được chính là giá trị tuyệt đối `$s1`
- ⇒ Nếu bit đầu là 0 lệnh XOR giữ nguyên toàn bộ bit
 - Giá trị `$s0` chính là `$s1` và `$t0` là (`0x00000000 = 0`)
 - Lệnh `subu $s0` và giá trị 0 sẽ giữ nguyên `$s0`
 - Kết quả `$s0` là `$s1` ban đầu (là số dương nên trị tuyệt đối là chính nó)

b. `move $s0, $s1:`

```
andi $s0, $s1, 0xffffffff
```

- ⇒ Lệnh And `$s1` với toàn bộ các bit 1 (`0xffffffff`) sẽ giữ nguyên các bit của `$s1` và lưu vào `$s0`

c. `not $s0, $s1:`

```
nor $s0, $s1, $zero
```

- ⇒ Do lệnh Or với giá trị có tất cả các bit là 0 giữ nguyên giá trị
- ⇒ Nor sẽ đảo ngược các bit

d. `ble $s1, $s2, label:`

```
slt $t0, $s2, $s1
beq $t0, $zero, label
```

`label:`

- ⇒ `slt` set giá trị `$t0` là 1 nếu `$s2 > $s1` và lưu vào `$t0` (bằng 0 trong trường hợp còn lại)
- ⇒ Thực hiện nhảy tới `label` khi `$s1 ≤ $s2` nên sử dụng lệnh `beq` giữa `$t0` và `$zero` (jump khi `$t0` có giá trị 0)

• Kết quả chạy thử:

a. `abs $s0, $s1:`

A3.asm		Registers		
		Name	Number	Value
1	<code>.text</code>	\$zero	0	0
2	<code>li \$s0, 0</code>	\$at	1	0
3	<code>li \$s1, -6</code>	\$v0	2	0
4	<code>li \$s2, 9</code>	\$v1	3	0
5	<code># abs \$s0, \$s1:</code>	\$a0	4	0
6	<code>sra \$t0, \$s1, 31</code>	\$a1	5	0
7	<code>xor \$s0, \$t0, \$s1</code>	\$a2	6	0
8	<code>subu \$s0, \$s0, \$t0</code>	\$a3	7	0
9		\$t0	8	-1
10	<code># move \$s0, \$s1:</code>	\$t1	9	0
11	<code>andi \$s0, \$s1, 0xffffffff</code>	\$t2	10	0
12		\$t3	11	0
13	<code># not \$s0, \$s1:</code>	\$t4	12	0
14	<code>nor \$s0, \$s1, \$zero</code>	\$t5	13	0
15		\$t6	14	0
16	<code># ble \$s1, \$s2, label:</code>	\$t7	15	0
17	<code>slt \$t0, \$s2, \$s1</code>	\$s0	16	6
18	<code>beq \$t0, \$zero, label</code>	\$s1	17	-6
19	<code>label:</code>	\$s2	18	9
20		\$s3	19	0
		\$s4	20	0

b. `move $s0, $s1:`

A3.asm		Registers		
		Name	Number	Value
1	<code>.text</code>	\$zero	0	0
2	<code>li \$s0, 0</code>	\$at	1	-1
3	<code>li \$s1, -6</code>	\$v0	2	0
4	<code>li \$s2, 9</code>	\$v1	3	0
5	<code># abs \$s0, \$s1:</code>	\$a0	4	0
6	<code>sra \$t0, \$s1, 31</code>	\$a1	5	0
7	<code>xor \$s0, \$t0, \$s1</code>	\$a2	6	0
8	<code>subu \$s0, \$s0, \$t0</code>	\$a3	7	0
9		\$t0	8	-1
10	<code># move \$s0, \$s1:</code>	\$t1	9	0
11	<code>andi \$s0, \$s1, 0xffffffff</code>	\$t2	10	0
12		\$t3	11	0
13	<code># not \$s0, \$s1:</code>	\$t4	12	0
14	<code>nor \$s0, \$s1, \$zero</code>	\$t5	13	0
15		\$t6	14	0
16	<code># ble \$s1, \$s2, label:</code>	\$t7	15	0
17	<code>slt \$t0, \$s2, \$s1</code>	\$s0	16	-6
18	<code>beq \$t0, \$zero, label</code>	\$s1	17	-6
19	<code>label:</code>	\$s2	18	9
20		\$s3	19	0
		\$s4	20	0

c. not \$s0, \$s1:

A3.asm		Registers		
		Name	Number	Value
1	.text	\$zero	0	0x00000000
2	li \$s0, 0	\$at	1	0xffffffff
3	li \$s1, -6	\$v0	2	0x00000000
4	li \$s2, 9	\$v1	3	0x00000000
5	# abs \$s0, \$s1:	\$a0	4	0x00000000
6	sra \$t0, \$s1, 31	\$a1	5	0x00000000
7	xor \$s0, \$t0, \$s1	\$a2	6	0x00000000
8	subu \$s0, \$s0, \$t0	\$a3	7	0x00000000
9		\$t0	8	0xffffffff
10	# move \$s0, \$s1:	\$t1	9	0x00000000
11	andi \$s0, \$s1, 0xffffffff	\$t2	10	0x00000000
12		\$t3	11	0x00000000
13	# not \$s0, \$s1:	\$t4	12	0x00000000
14	nor \$s0, \$s1, \$zero	\$t5	13	0x00000000
15		\$t6	14	0x00000000
16	# ble \$s1, \$s2, label:	\$t7	15	0x00000000
17	slt \$t0, \$s2, \$s1	\$s0	16	0x00000005
18	beq \$t0, \$zero, label	\$s1	17	0xfffffffffa
19	label:	\$s2	18	0x00000009
20		\$s3	19	0x00000000
		\$s4	20	0x00000000

d. ble \$s1, \$s2, label:

A3.asm		Registers		
		Name	Number	Value
1	.text	\$zero	0	0x00000000
2	li \$s0, 0	\$at	1	0xffffffff
3	li \$s1, -6	\$v0	2	0x00000000
4	li \$s2, 9	\$v1	3	0x00000000
5	# abs \$s0, \$s1:	\$a0	4	0x00000000
6	sra \$t0, \$s1, 31	\$a1	5	0x00000000
7	xor \$s0, \$t0, \$s1	\$a2	6	0x00000000
8	subu \$s0, \$s0, \$t0	\$a3	7	0x00000000
9		\$t0	8	0x00000003
10	# move \$s0, \$s1:	\$t1	9	0x00000000
11	andi \$s0, \$s1, 0xffffffff	\$t2	10	0x00000000
12		\$t3	11	0x00000000
13	# not \$s0, \$s1:	\$t4	12	0x00000000
14	nor \$s0, \$s1, \$zero	\$t5	13	0x00000000
15		\$t6	14	0x00000000
16	# ble \$s1, \$s2, label:	\$t7	15	0x00000000
17	slt \$t0, \$s2, \$s1	\$s0	16	0x00000005
18	beq \$t0, \$zero, label	\$s1	17	0xfffffffffa
19	j EXIT	\$s2	18	0x00000009
20	label: li \$t0, 3	\$s3	19	0x00000000
21	EXIT:	\$s4	20	0x00000000
		\$s5	21	0x00000000

Assignment 4:

- **Code:**

```
.text
li $s0, -9999
li $s1, -2003
li $t0, 0
xor $t1, $s0, $s1
blez $t1, Exit
addu $t2, $s0, $s1
xor $t1, $s1, $t2
bgez $t1, Exit

Overflow:
li $t0, 1

Exit:
```

- **Giải thích:**

- Sử dụng phép XOR để kiểm tra xem \$s1 và \$s2 có cùng dấu không
 - Nếu \$t0 là số dương tức là hai số trên cùng dấu
 - Nếu trong trường hợp còn lại tức là hai số khác dấu
 - ⇒ Nhảy tới EXIT
- Trong trường hợp khác dấu, cộng hai số rồi lưu vào \$t2, sau đó tiếp tục sử dụng phép XOR để kiểm tra dấu của kết quả cộng đó với một trong hai số (ở đây là \$s1) => Lưu kết quả XOR vào \$t1
 - Nếu \$t1 lớn hơn hoặc bằng 0 tức là kết quả XOR là cùng dấu
 - ⇒ Không xảy ra tràn, nhảy tới EXIT
 - Trong trường hợp còn lại tức là đã xảy ra tràn số, chương trình tiếp tục chạy tới lệnh li và gán cho \$t0 giá trị 1

• Kết quả chạy thử

- Trường hợp 1: Hai số trái dấu

A5.asm		A4.asm	
1	.text		
2	li \$s0, -9999		
3	li \$s1, 3000		
4	li \$t0, 0		
5	xor \$t1, \$s0, \$s1		
6	blez \$t1, Exit		
7	addu \$t2, \$s0, \$s1		
8	xor \$t1, \$s1, \$t2		
9	bgez \$t1, Exit		
10	Overflow:		
11	li \$t0, 1		
12	Exit:		
13			

Registers		
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	-11447
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	-9999
\$s1	17	3000

- Trường hợp 2: Hai số cùng dương không tràn

Edit		Execute	
A5.asm		A4.asm	
1	.text		
2	li \$s0, 2003		
3	li \$s1, 3000		
4	li \$t0, 0		
5	xor \$t1, \$s0, \$s1		
6	blez \$t1, Exit		
7	addu \$t2, \$s0, \$s1		
8	xor \$t1, \$s1, \$t2		
9	bgez \$t1, Exit		
10	Overflow:		
11	li \$t0, 1		
12	Exit:		
13			

Coproc 1		Coproc 0	
Registers			
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	6195	
\$t2	10	5003	
\$t3	11	0	
\$t4	12	0	
\$t5	13	0	
\$t6	14	0	
\$t7	15	0	
\$s0	16	2003	
\$s1	17	3000	

- Trường hợp 3: Hai số cùng dương có gây tràn

A5.asm		A4.asm	
1	.text		
2	li \$s0, 0x7fffffff		
3	li \$s1, 3000		
4	li \$t0, 0		
5	xor \$t1, \$s0, \$s1		
6	blez \$t1, Exit		
7	addu \$t2, \$s0, \$s1		
8	xor \$t1, \$s1, \$t2		
9	bgez \$t1, Exit		
10	Overflow:		
11	li \$t0, 1		
12	Exit:		
13			

Registers		
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	1
\$t1	9	-2147483633
\$t2	10	-2147480649
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	2147483647
\$s1	17	3000

- Trường hợp 4: Hai số cùng âm không tràn

A5.asm		A4.asm	
1	.text		
2	li \$s0, -2003		
3	li \$s1, -3000		
4	li \$t0, 0		
5	xor \$t1, \$s0, \$s1		
6	blez \$t1, Exit		
7	addu \$t2, \$s0, \$s1		
8	xor \$t1, \$s1, \$t2		
9	bgez \$t1, Exit		
10	Overflow:		
11	li \$t0, 1		
12	Exit:		
13			

Registers		
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	6205
\$t2	10	-5003
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	-2003
\$s1	17	-3000

- Trường hợp 5: Hai số cùng âm có gây tràn

A5.asm		A4.asm	
1	.text		
2	li \$s0, 0x80000000		
3	li \$s1, -3000		
4	li \$t0, 0		
5	xor \$t1, \$s0, \$s1		
6	blez \$t1, Exit		
7	addu \$t2, \$s0, \$s1		
8	xor \$t1, \$s1, \$t2		
9	bgez \$t1, Exit		
10	Overflow:		
11	li \$t0, 1		
12	Exit:		
13			

Registers		
Name	Number	Value
\$zero	0	0
\$at	1	-2147483648
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	1
\$t1	9	-2147483648
\$t2	10	2147480648
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	-2147483648
\$s1	17	-3000

Assignment 5:

- Code:

```
.text
li $s0, 9
li $s1, 32
li $s2, 0
move $t1, $s1
```

```

loop:
    beq $t1, 1, multiple
    srl $t1, $t1, 1
    addi $s2, $s2, 1
    j loop
multiple:
    sllv $t0, $s0, $s2

```

• Giải thích:

- Một số là bội của 2 khi đổi sang nhị phân sẽ có duy nhất một bit bằng 1 còn lại là 0. Khi liên tục dịch phải từng bit của số đó, dịch tới khi 1 nằm ở vị trí bit số 0 thì số lần dịch sẽ chính là số mũ của 2
- Nguyên tắc trên được áp dụng trong phần **loop**, thoát khỏi vòng lặp khi giá trị \$t1 chỉ còn là 1 và nhảy tới phần multiple
- Ở **multiple** ta dịch trái \$s0 số bit tương ứng với số được lưu tại \$s2 (chính là số mũ của 2 đã được tính tại vòng lặp)
 - Dịch trái 1 bit là $\times 2^1$, 2 bit là $\times 2^2$, ..., n bit là $\times 2^n$
- Kết quả được lưu tại \$t0 chính là kết quả của phép nhân

• Kết quả chạy thử:

```

1  #Laboratory Exercise 4, Assignment 5
2  .text
3      li $s0, 9
4      li $s1, 32
5      li $s2, 0          # i = 0
6      move $t1, $s1
7  loop:
8      beq $t1, 1, multiple # Kiểm tra t1 = 1 thì nhảy đến multiple
9      srl $t1, $t1, 1      # Dịch phải t1 sang 1 bit (t1 = t1 / 2)
10     addi $s2, $s2, 1      # i = i + 1
11     j loop
12 multiple:
13     sllv $t0, $s0, $s2    # Dịch trái s0 sang s2 bit (t0 = s0 * 2^s2)
14
15

```

Registers		
Name	Number	Value
\$zero	0	0
\$at	1	1
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	288
\$t1	9	1
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	9
\$s1	17	32
\$s2	18	5
\$s3	19	0
\$s4	20	0
\$s5	21	0

- Số mũ của 32 là 5 được lưu tại \$s2
- Giá trị của phép nhân $9 \times 32 = 288$ được lưu tại \$t0

Consolutions

1. What is the difference between SLLV and SLL instructions?

- Lệnh `sll $s1, $s2, imm:`
⇒ Dịch trái `$s2` số bit được quy định ở phần immediate, sau đó lưu kết quả vào `$s1`.
- Lệnh `sllv $s1, $s2, $s3:`
⇒ Dịch trái `$s2` số bit được quy định bởi 5 bit trật tự thấp (low-order) của `$s3`, mang giá trị từ 0-31 và lưu kết quả vào `$s1`.

2. What is the difference between SRLV and SRL instructions?

- Lệnh `srl $s1, $s2, imm:`
⇒ Dịch phải `$s2` số bit được quy định ở phần immediate, sau đó lưu kết quả vào `$s1`.
- Lệnh `srlv $s1, $s2, $s3:`
⇒ Dịch phải `$s2` số bit được quy định bởi 5 bit trật tự thấp (low-order) của `$s3`, mang giá trị từ 0-31 và lưu kết quả vào `$s1`.