

BÀI BÁO CÁO TUẦN 4

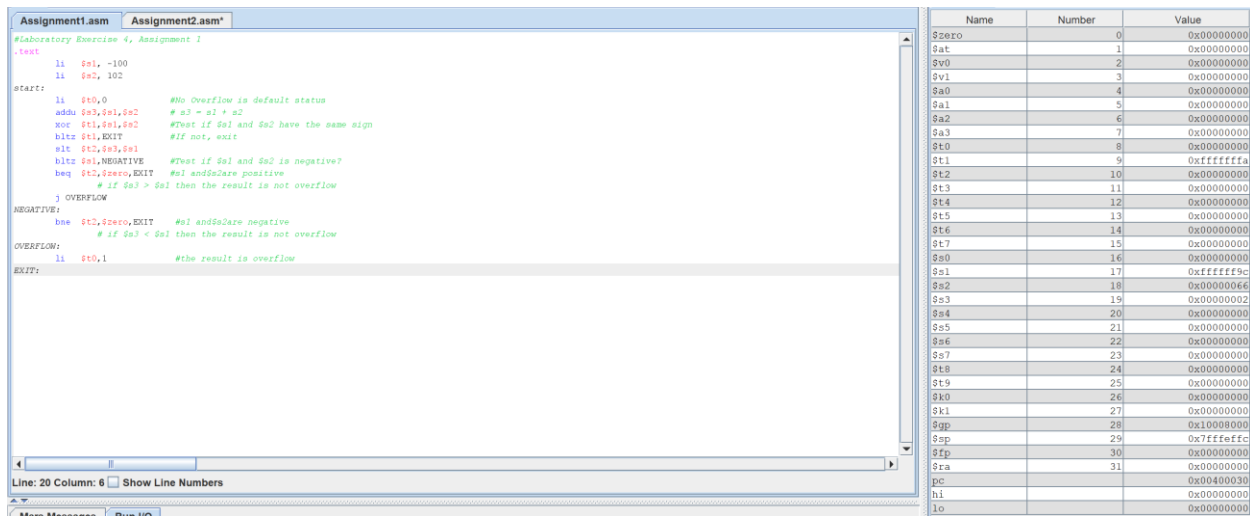
Họ và tên: Nguyễn Trọng Khánh Duy

MSSV: 20210284

Assignment 1

Trường hợp 1: Cộng 2 số khác dấu

Giả sử $\$s1 = -100$; $\$s2 = 102$



```
#Laboratory Exercise 4, Assignment 1
.text
li $s1, -100
li $s2, 102

start:
li $t0, 0           #No Overflow in default status
addu $s3,$s1,$s2    # s3 = s1 + s2
xor $t1,$s1,$s2     #Test if $s1 and $s2 have the same sign
hltr $t1,EXIT       #If not, exit
slt $t2,$s1,$s1     #Test if $s1 and $s2 is negative?
hltr $s1,NEGATIVE   #s1 and $s2 are positive
beq $t2,$zero,EXIT  # if $s2 > $s1 then the result is not overflow
} OVERFLOW
NEGATIVE:
bne $t2,$zero,EXIT  #s1 and $s2 are negative
# if $s2 < $s1 then the result is not overflow
OVERFLOW:
li $t0, 1           #the result is overflow
EXIT:

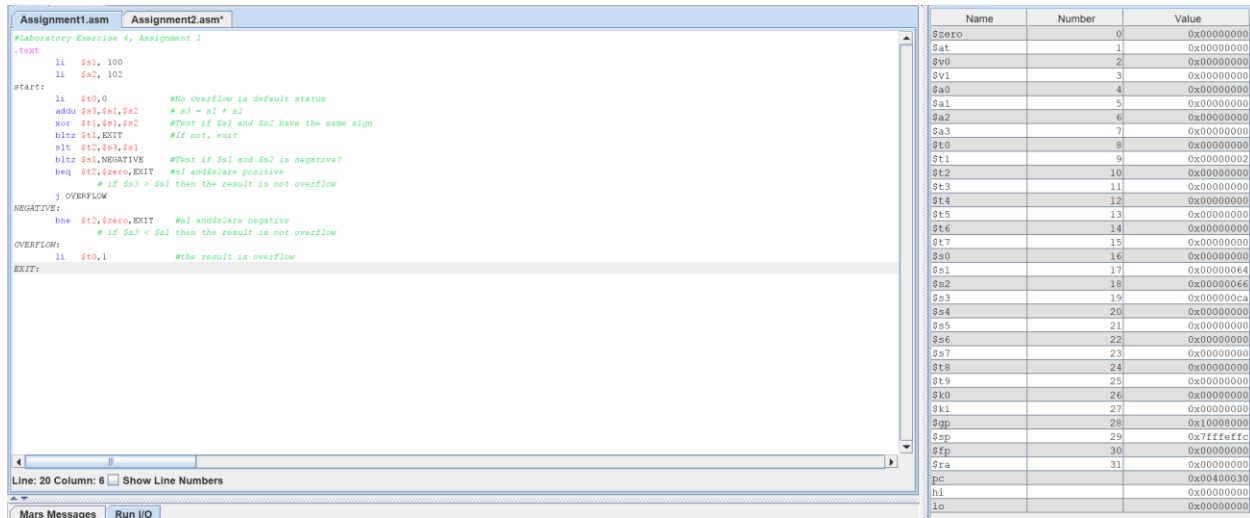
```

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	0xffffffffa
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0xffffffff9c
\$s0	18	0x00000066
\$s1	19	0x00000002
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x00000000
\$s7	25	0x00000000
\$s8	26	0x00000000
\$s9	27	0x00000000
\$sp	28	0x00000000
\$fp	29	0x7ffffeffc
\$ra	30	0x00000000
\$pc	31	0x00400030
hi		0x00000000
lo		0x00000000

Kết quả: $\$s3 = 2$; $\$t0 = 0$

Trường hợp 2: Cộng 2 số dương không tràn số

Giả sử $\$s1 = 100$; $\$s2 = 102$



```
#Laboratory Exercise 4, Assignment 1
.text
li $s1, 100
li $s2, 102

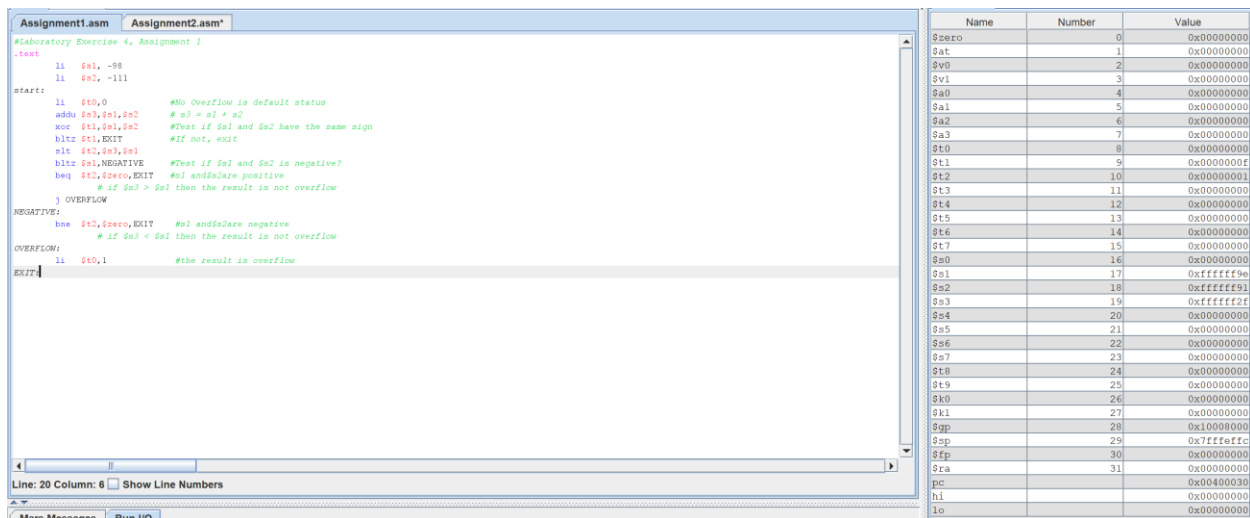
start:
li $t0, 0           #No Overflow in default status
addu $s3,$s1,$s2    # s3 = s1 + s2
xor $t1,$s1,$s2     #Test if $s1 and $s2 have the same sign
bltz $t1,EXIT       #If not, exit
slt $t2,$s1,$s1
bltz $s1,NEGATIVE   #Test if $s1 and $s2 is negative?
beq $t2,$zero,EXIT  #s1 and $s2 are positive
                    # if $s2 > $s1 then the result is not overflow
} OVERFLOW
NEGATIVE:
bne $t2,$zero,EXIT  #s1 and $s2 are negative
                    # if $s2 < $s1 then the result is not overflow
OVERFLOW:
li $t0, 1           #the result is overflow
EXIT:
```

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	0x00000002
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000064
\$s2	18	0x00000066
\$s3	19	0x000000ca
\$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	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400030
hi		0x00000000
lo		0x00000000

Kết quả : $\$s3 = 0x000000ca = 202_{10}$; $\$t0 = 0$

Trường hợp 3: Cộng 2 số âm không tràn số

Giả sử $\$s1 = -98$; $\$s2 = -111$



```
#Laboratory Exercise 4, Assignment 1
.text
li $s1, -98
li $s2, -111

start:
li $t0, 0           #No Overflow in 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 $t2,$s1,$s1
bltz $s1,NEGATIVE   #Test if $s1 and $s2 is negative?
beq $t2,$zero,EXIT  #s1 and $s2 are positive
                    # if $s2 > $s1 then the result is not overflow
} OVERFLOW
NEGATIVE:
bne $t2,$zero,EXIT  #s1 and $s2 are negative
                    # if $s2 < $s1 then the result is not overflow
OVERFLOW:
li $t0, 1           #the result is overflow
EXIT:
```

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	0x0000000f
\$t2	10	0x00000001
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0xffffffff9e
\$s2	18	0xffffffff91
\$s3	19	0xffffffff2f
\$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	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400030
hi		0x00000000
lo		0x00000000

Kết quả : $\$s3 = 0xffffffff2f = -209_{10}$; $\$t0 = 0$

Trường hợp 4: Cộng 2 số dương tràn số

Giả sử $\$s1 = 0x7ffffff$; $\$s2 = 102$

```
#Laboratory Exercise 4, Assignment 1
.text
li $s1, 0x7ffffff
li $s2, 102

start:
li $t0, 0          #No Overflow is 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, $s1, $s1  #Test if $s1 and $s2 is negative?
bltz $t2, NEGATIVE #if $s1 > $s2 then the result is not overflow
beq $t2, $zero, EXIT #if $s1 < $s2 then the result is not overflow
# if $s3 > $s1 then the result is not overflow
} OVERFLOW
NEGATIVE:
bne $t2, $zero, EXIT #if $s1 < $s2 then the result is not overflow
OVERFLOW:
li $t0, 1          #the result is overflow
EXIT:
```

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x7ffff000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000001
\$t1	9	0x7ffffff9
\$t2	10	0x00000001
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x7ffffff
\$s2	18	0x00000066
\$s3	19	0x80000065
\$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	0x10008000
\$sp	29	0x7ffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400034
hi		0x00000000
lo		0x00000000

Kết quả : $\$s3 = 0x80000065$; $\$t0 = 1$

Trường hợp 5: Cộng 2 số âm tràn số

Giả sử $\$s1 = 0x80000000$; $\$s2 = -143$

```
#Laboratory Exercise 4, Assignment 1
.text
li $s1, 0x80000000
li $s2, -143

start:
li $t0, 0          #No Overflow is 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, $s1, $s1  #Test if $s1 and $s2 is negative?
bltz $t2, NEGATIVE #if $s1 > $s2 then the result is not overflow
beq $t2, $zero, EXIT #if $s1 < $s2 then the result is not overflow
# if $s3 > $s1 then the result is not overflow
} OVERFLOW
NEGATIVE:
bne $t2, $zero, EXIT #if $s1 < $s2 then the result is not overflow
OVERFLOW:
li $t0, 1          #the result is overflow
EXIT:
```

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x80000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000001
\$t1	9	0x7ffffff7
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x80000000
\$s2	18	0xfffffff7
\$s3	19	0x7ffffff7
\$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	0x10008000
\$sp	29	0x7ffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400034
hi		0x00000000
lo		0x00000000

Kết quả : $\$s3 = 0x7ffffff7$; $\$t0 = 1$

Assignment 2

CODE:

#Laboratory Exercise 4, Assignment 2

.text

```
li $s0, 0x12345678      # Load test value for these function

andi $t0, $s0, 0xff000000 # Extract the MSB of $s0

sra $t0, $t0, 24         # Shift right 24 bits

andi $t1, $s0, 0xffffffff #Clear LSB of $s0

ori $t2, $s0, 0x000000ff  #Set LSB of $s0(bits 7 to 0 are set to 1)

andi $t3, $s0, 0          #Clear $s0($s0=0, must use logical instructions)
```

Kết quả:

\$t0 = 0x00000012

\$t1 = 0x12345600

\$t2 = 0x123456ff

\$t3 = 0x00000000

The screenshot shows a MIPS assembly simulator interface. The main window displays the assembly code for 'Assignment2.asm' with line numbers 1 through 6. The code is as follows:

```
1: .text
2: li $s0, 0x12345678      # Load test value for these function
3: andi $t0, $s0, 0xff000000 # Extract the MSB of $s0
4: sra $t0, $t0, 24         # Shift right 24 bits
5: andi $t1, $s0, 0xffffffff #Clear LSB of $s0
6: ori $t2, $s0, 0x000000ff  #Set LSB of $s0(bits 7 to 0 are set to 1)
7: andi $t3, $s0, 0          #Clear $s0($s0=0, must use logical instructions)
```

On the right side, there is a register window showing the values of registers \$s0 through \$t3. The values are:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0xffffffff
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000012
\$t1	9	0x12345600
\$t2	10	0x123456ff
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x12345678
\$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	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400020
\$hi		0x00000000
\$lo		0x00000000

Giải thích:

Logic AND từng bit có đặc điểm: and với 1 ra kết quả chính nó, and với 0 thì ra 0

- Lệnh `andi $t0, $s0, 0xff000000` sẽ giữ lại 8 bits cao (vì nhân với 0xff000000) và cho tất cả các bit còn lại bằng 0. Sau đó dịch phải 24 bit kết quả vừa nhận được => Trích xuất MSB
- Lệnh `andi $t1, $s0, 0xffff00` sẽ xóa đi 8 bit cuối và giữ lại 24 bit cao => xóa LBS

Logic OR từng bit có đặc điểm : khi or với 1 sẽ trả về 1, or với 0 trả về chính nó. Lệnh `ori $t2, $s0, 0x000000ff` sẽ giữ nguyên 24 bit cao (do or với toàn bộ là bit 0) và đưa tất cả còn lại lên 1 => Lấy được giá trị của LSB

Assignment 3

a. abs \$s0, \$s1

```
sra $t0, $s1, 31
```

```
xor $s0, $t0, $s1
```

```
subu $s0, $s0, $t0
```

Giải thích:

Các bit của \$t0 được điền bởi giá trị bit dấu của \$s1

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

b) move \$s0, \$s1

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

Giải thích: Lệnh AND \$s1 với toàn bộ bit 1 sẽ giữ nguyên bit của \$s1 và lưu vào \$s0

c) not \$s0, \$s1

nor \$s0, \$s1, \$zero

Lệnh OR với tất cả các giá trị 0 sẽ trả về giá trị ban đầu => Lệnh nor sẽ đảo ngược các bit

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

slt \$t0, \$s2, \$s1

beq \$t0, \$zero, label

label:

Lệnh slt \$t0, \$s2, \$s1 sẽ cho ta kết quả: \$t0 = 1 nếu \$s2 > \$s1 hoặc \$t0 = 0 nếu \$s2 < \$s1

=> Thực hiện nhảy tới label khi \$s1 < \$s2 nên lệnh beq \$t0, \$zero, label sẽ nhảy đến label khi \$t0 = 0

Assignment 4

CODE:

.text

li \$s0, -2003

li \$s1, -3020

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 XOR để phân biệt \$s0 và \$s1 có dùng dấu hay không.

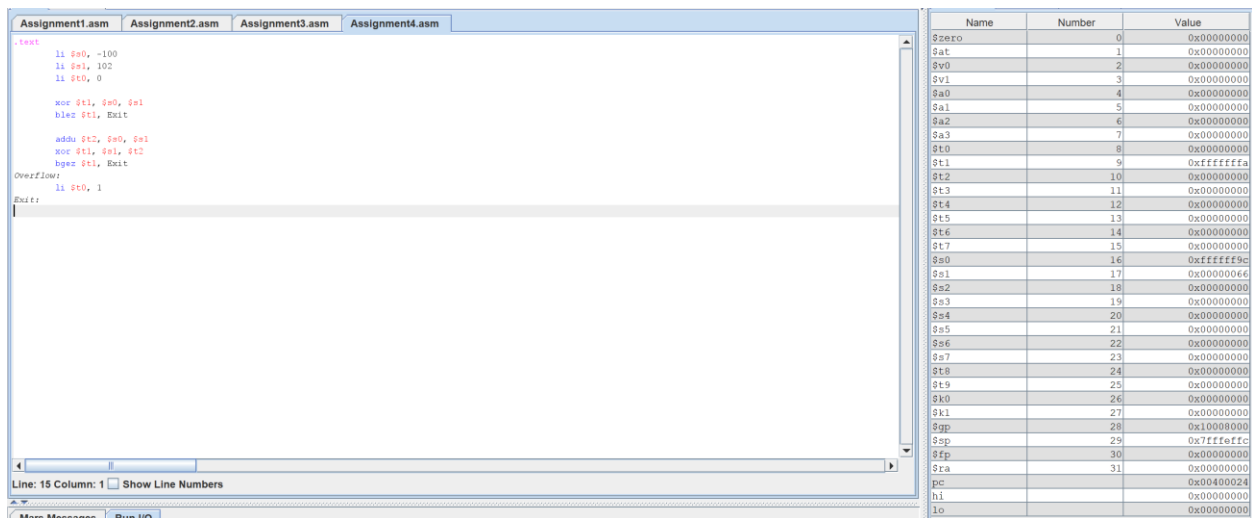
- Nếu khác dấu thì lệnh **blez \$t1, Exit** sẽ nhảy đến Exit và kết thúc chương trình trả về **\$t0 = 0** => Không tràn số
- Nếu cùng dấu thì sẽ thực hiện lệnh **addu \$t2, \$s0, \$s1** và lưu tổng giá trị của 2 thanh ghi **\$s0** và **\$s1** vào thanh ghi **\$t2** và sau đó thực hiện

lệnh `xor $t1, $s1, $t2` để kiểm tra dấu của 2 số đó với tổng của chính nó.

- Nếu cùng dấu thì `bgez $t1`, Exit sẽ nhảy đến exit và kết thúc chương trình và trả về `$t0 = 0` => Không tràn số
- Nếu khác dấu thì sẽ thực hiện lệnh `li $t0, 1` => trả về `$t0 = 1` => tràn số

Trường hợp 1: Cộng 2 số khác dấu

Giả sử `$s0 = -100`; `$s1 = 102`



```
.text
li $s0, -100
li $s1, 102
li $t0, 0

xor $t1, $s0, $s1
bgez $t1, Exit

addu $t2, $s0, $s1
xor $t1, $s1, $t2
bgez $t1, Exit

Overflow:
li $t0, 1

Exit:
```

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	0xfffffffffa
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0xffffffff9c
\$a1	17	0x00000066
\$a2	18	0x00000000
\$a3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00000000
\$sp	29	0xfffffffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

Kết quả: `$t0 = 0`

Trường hợp 2: Cộng 2 số cùng dấu không tràn số

Giả sử $\$s0 = 100$; $\$s1 = 102$

```
.text
li $s0, 100
li $s1, 102
li $t0, 0

xor $t1, $s0, $s1
bnez $t1, Exit

addu $t2, $s0, $s1
xor $t1, $s1, $t2
bgez $t1, Exit

Overflow:
li $t0, 1

Exit:
```

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	0x000000ac
\$t2	10	0x000000ca
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000064
\$s1	17	0x00000066
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

Kết quả: $\$t0 = 0$

Trường hợp 3: Cộng 2 số cùng dấu tràn số

Giả sử $\$s1 = 0x80000000$; $\$s2 = -143$

```
.text
li $s0, 0x80000000
li $s1, -143
li $t0, 0

xor $t1, $s0, $s1
bnez $t1, Exit

addu $t2, $s0, $s1
xor $t1, $s1, $t2
bgez $t1, Exit

Overflow:
li $t0, 1

Exit:
```

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	0x00000001
\$t1	9	0x00000000
\$t2	10	0x7fffff71
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x80000000
\$s1	17	0xfffff71
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400028
hi		0x00000000
lo		0x00000000

Kết quả: $\$t0 = 1$

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 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
- **Assignment 5**

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`.