Laboratory Exercise 4

Arithmetic and Logical operation

1. **Assignment 1:** *Create a new project to implement the Home Assigment 1. Compile and upload to simulator. Initialize two operands (register $s1 and $s2), run this program step by step, observe memory and registers value.*
   - Giả sử gán $s1 = 10 và $s2 = 20

```
2   .text
3           li $s1, 10
4           li $s2, 20
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
```

   - Dòng 8 và 9: Kiểm tra xem s1 s2 có cùng dấu không, nếu không sẽ thoát chương trình. Ở đây, 10 và 20 cùng dấu nên chương trình được tiếp diễn câu lệnh tiếp theo.
   - Dòng 11 và 12: Kiểm tra s1 và s2 có phải số âm không, nếu có thì thoát chương trình. Ở đây, 10 và 20 đều là số dương nên chương trình được tiếp diễn, s3 = s1 + s2 = 30 > s1 = 10 nên không có overflow

- Kết quả cuối: t0 = 0; s3 = 30

$t0	8	0x00000000
$t1	9	0x0000001e
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x0000000a
$s2	18	0x00000014
$s3	19	0x0000001e

2. **Assignment 2:** *Write a program to do the following tasks: Extract MSB of $s0; Clear LSB of $s0; Set LSB of $s0 (bits 7 to 0 are set to 1); Clear $s0 ($s0=0, must use logical instructions). MSB: Most Significant Byte LSB: Least Significant Byte*

```
1   .text
2
3           li $s0, 0x67bc3262
4
5   #Extract Msb:
6           andi $s1, $s0, 0xff000000
7
8   #Clear Lsb:
9           andi $s2, $s0, 0xffffff00
10
11  #Set Lsb:
12          ori $s3, $s0, 0x000000ff
13
14  #Clear $s0:
15          xor $s0, $s0, $s0
```

- Kết quả ta thu được:

$s0	16	0x00000000
$s1	17	0x67000000
$s2	18	0x67bc3200
$s3	19	0x67bc32ff

Trong đó: s0 = 0 sau khi Clear
s1 được có bởi Extract MSB
s2 được có bởi Clear LSB
s3 được có bởi Set LSB

- Giải thích: Khi and bit với 0 thì được kết quả là 0, với 1 thì được kết quả là 1. Vậy nên ta có thể dùng lệnh 'andi' để Extract MSB, Clear LSB và Clear $s0. Khi or bit với 1 thì luôn thu được kết quả là 1, or với 0 thì giữ nguyên nên ta dùng 'ori' để Set LSB.

3. Assignment 3:

Assignment 3

Pseudo instructions in MIPS are not-directly-run-on-MIPS-processor instructions which need to be converted to real-instructions of MIPS. Re-write the following pseudo instructions using real-instructions understood by MIPS processors:

a. `abs $s0,$s1`
```
$s0 <= | $s1 |
```
b. `move $s0,$s1`
```
$s0 <= $s1
```
c. `not $s0, $s1`
```
$s0 <= bit invert ($s1)
```
d. `ble $s1,$s2,label`


```
if ($s1 <= $s2)
    j label
```

a) abs

```
1   .text
2           li $s1, -66
3           li $s2, 2812
4           bltz $s1,Abs            # $s1 < 0
5           addi $s0,$s1, 0
6           j Continue
7   Abs:
8           sub $s0, $zero, $s1
9
10  Continue:
```

Để lưu giá trị tuyệt đối của $s1 vào $s0, ta có thể lưu $s1 vào $s0 nếu $s1>=0, trong trường hợp còn lại, ta lưu 0-$s1 vào $s0.

$s0	16	0x00000042
$s1	17	0xffffffbe
$s2	18	0x00000afc
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400018
hi		0x00000000
lo		0x00000000

b) move

```
1  .text
2          li $s1, -66
3  #move
4          addi $s0, $s1,0
```

$s0	16	0xffffffbe
$s1	17	0xffffffbe

Để lưu giá trị tuyệt đối của $s1 vào $s0, ta có thể lưu $s1 vào $s0 nếu $s1>=0, trong trường hợp còn lại, ta lưu 0-$s1 vào $s0.

c) not

```
1  .text
2          li $s1, -66
3  #not
4          nor $s0, $s1, $zero
```

$s0	16	0x00000041
$s1	17	0xffffffbe

Thay vì dùng not ta có thể dùng ‘nor $s0, $s1, $zero’ để lưu giá trị not $s1 vào $s0 vì khi 1 nor với 0 sẽ được 0 còn 0 nor với 0 sẽ được 1.

d) ble

```
1   .text
2           li $s1, -66
3           li $s2, 2812
4   #ble
5           slt $t1, $s2, $s1       #s2 < s1
6           bne $t1, $zero, L       # if s2 < S1 jump to L
7           j Exit
8   L:
9
10  Exit:
```

$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0xffffffbe
$s2	18	0x00000afc

Thay vì trực tiếp so sánh để nhảy bằng ble, ta có thể tính hiệu rồi so sánh nó với 0 rồi mới nhảy

4. **Assignment 4:** *To dectect overflow in additional operation, we also use other rule than the one in Assignment 1. This rule is: when add two operands that have the same sign, overflow will occur if the sum doesn't have the same sign with either operands. You need to use this rule to write another overflow detection program.*
   - Chương trình được viết như sau:

```
1   .text
2           li $s1, 0x9fffffff
3           li $s2, 0x9fffffff
4   start:
5           li $t0,0                #No Overflow is default status
6           addu $s3,$s1,$s2        # s3 = s1 + s2
7           xor $t1,$s1,$s2         #Test if $s1 and $s2 have the same sign
8           bltz $t1,EXIT           #If not, exit
9           xor $t2,$s1,$s3         #Test if $s1 and $s3 have the same sign
10          bltz $t2,OVERFLOW       #if not, jupm to overflow
11          j EXIT
12  OVERFLOW:
13          li $t0,1                #the result is overflow
14  EXIT:
```

- Test: 0x9fffffff + 0x9fffffff

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0x00000000
$at	1	0x9fff0000
$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	0xa0000001
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x9fffffff
$s2	18	0x9fffffff
$s3	19	0x3ffffffe
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400030
hi		0x00000000
lo		0x00000000

- Ở trường hợp này, t2<0 chứng tỏ $s1 và $s3 khác dấu. Điều này nói rõ đã có sự tràn số xảy ra
- Giải thích: Nếu $s1 và $s2 khác dấu thì không thể có sự tràn số. nếu $s1 và $s2 cùng dấu, ta xét xem $s1 và $s3 có cùng dấu không. Nếu có thì không tràn số, còn nếu không thì đã có sự tràn số xảy ra.

5. **Assignment 5:** *Write a program that implement multiply by a small power of 2. (2, 4, 8, 16, etc for example).*
   - Chương trình được viết như sau:

```
.text
        li $s0, 5       # n = 5
        li $t1 ,1       # i = 1
        addi $t1, $zero, 0
Loop:
        sra $t2, $s2, 1
        beq $t2, $zero, Endloop
        add $t1, $t1, 1
        add $s2, $t2, $zero
        j Loop
Endloop:
        sllv $s0, $s1, $t1
```

   - Giá trị thanh ghi:

Registers | Coproc 1 | Coproc 0

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	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

- Giải thích: Ở bài này, ta dùng vòng lặp loop để tính giá trị ‘n’ từ ‘2^n’, sau khi kết thúc vòng lặp, ta dịch trái ‘n’ bit giá trị trên thanh ghi $s1 rồi lưu vào $s0. Từ đó ta được giá trị trên thanh $s0 là kết quả phép nhân của giá trị trên thanh $s1 với 2^n.