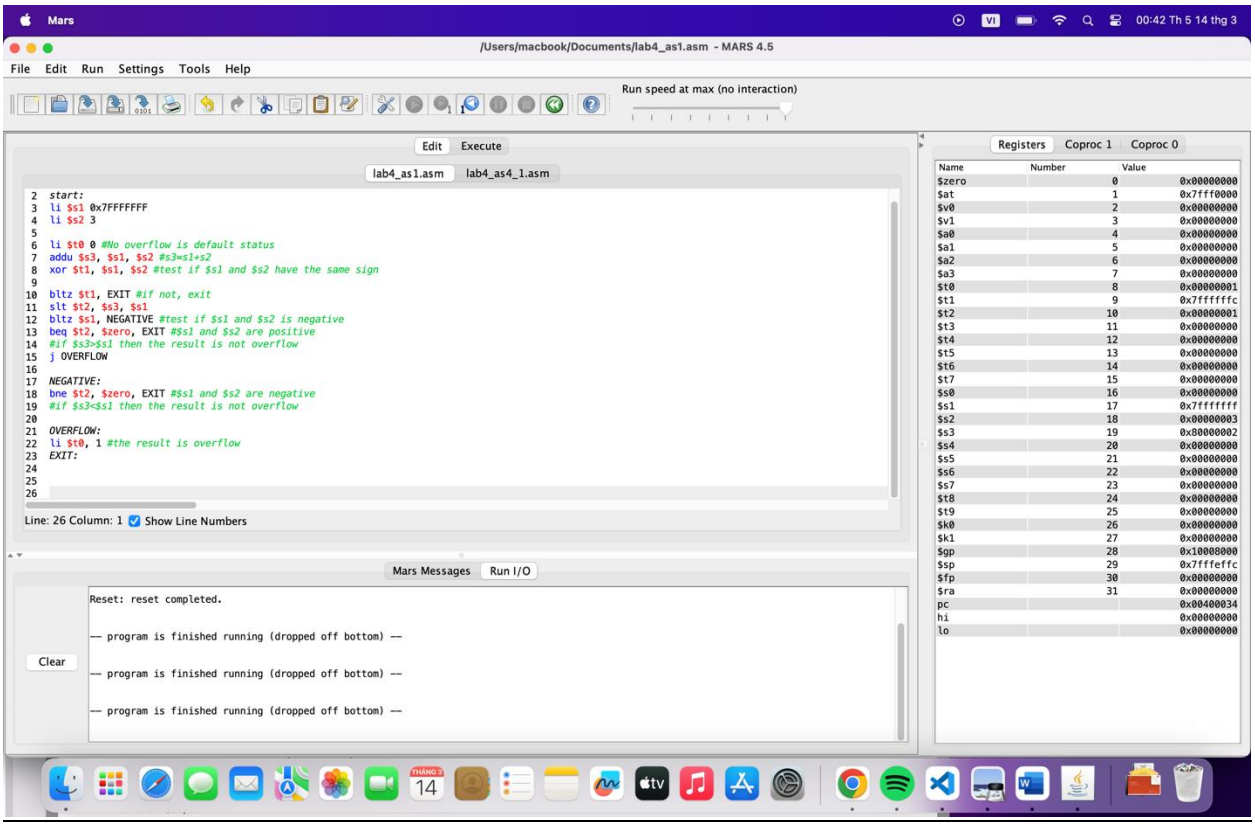
ASSIGNMENT WEEK 4

-Họ tên: Hồ Tuấn Huy
-MSSV: 20225856

Assignment 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.



Giải thích:

-Ban đầu ta set $t0=0 (tức không bị tràn).

-Tính tổng s1+s2 và lưu vào s3 bằng lệnh addu (phép cộng nhưng không báo lỗi khi tràn số, khác add).

-Sử dụng xor $t1, $s1, $s2 để kiểm tra $s1 và $s2 có cùng dấu không:

+Nếu cùng dấu (cùng bit đầu) thì trả về kết quả lưu vào $t1 có giá trị dương (do bit đầu là 0).

+Nếu không (khác bit đầu) thì trả về kết quả lưu vào $t1 có giá trị âm (do bit đầu là 1).

Input A	Input B	XOR Output
0	0	0
0	1	1
1	0	1
1	1	0

-Lệnh tiếp theo là bltz (branch if less than zero)->so sánh nếu $t1<0 (tức $s1 và $s2 khác dấu) thì sẽ chuyển đến EXIT kết thúc. Nếu $t1>0 thì sẽ thực hiện lệnh tiếp là slt $t2, $s3, $s1->so sánh xem $s3<$s1:
+Nếu có thì trả về 1 lưu vào $t2.
+Nếu không thì trả về 0 lưu vào $t2.
-So sánh:
+Nếu $s1 và $s2 âm mà chỉ cần $s3 lớn hơn 1 trong 2 số hạng ($t2=0) thì bị tràn số ($t0=1). Còn ngược lại ($t2=1) thì không xảy ra tràn->EXIT.
+Nếu $s1 và $s2 dương mà tổng $s3 bé hơn 1 trong 2 số hạng ($t2=1) thì bị tràn số ($t0=1). Còn ngược lại ($t2=0) thì không xảy ra tràn->EXIT.
Kết quả: có overflow:

$t0	8	0x00000001
$t1	9	0x7ffffffc
$t2	10	0x00000001
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x7fffffff
$s2	18	0x00000003
$s3	19	0x80000002

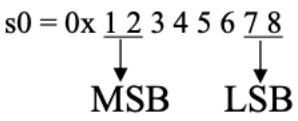
Assignment 2:

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

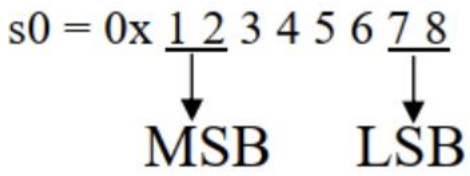


a. **Extract MSB of $s0:**

```
1  .text
2  li $s0, 0x0603ff22 #load test value for these function
3  andi $t0, $s0, 0xff000000 #extract MSB of $s0
```

-Trích ra phần MSB của thanh ghi $s0, ví dụ:

MSB: Most Significant Byte
LSB: Least Significant Byte



-Khi đó ta thực hiện lệnh andi của $s0 với 0xff000000 và lưu kết quả vào $t0.

AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

0xff000000= 1111 1111 0000…0000 do đó khi thực hiện andi $s0 với 0xff000000 thì nếu gặp bit 1 sẽ giữ nguyên còn gặp bit 0 thì clear về 0.

Kết quả:

$t0	8	0x06000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x0603ff22

=>Trong trường hợp này, byte cao nhất của giá trị 0x0603ff22 là 0x06 nên sau khi thực hiện lệnh andi thì $t0 chứa giá trị 0x06000000.

b. Clear LSB of $s0:

```
.text
li $s0, 0x0603ff22 #load test value for these function
andi $t0, $s0, 0xffffff00 #clear LSB of $s0
```

Tương tự, 0xffffff00=1111…1111 0000 0000. Trong trường hợp này, byte thấp nhất của 0x0603ff22 là 22 nên sẽ bị clear về 00:

Kết quả:

$t0	8	0x0603ff00
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x0603ff22

c. **Set LSB of $s0 (bits 7 to 0 are set to 1):**

```
.text
li $s0, 0x0603ff22 #load test value for these function
andi $t0, $s0, 0xffffff00 #clear LSB of $s0
ori $t1, $t0, 0xff #set LSB of $s0 (bits 7 to 0 are set to 1)
```

Thực hiện 1 lần clear LSB của $s0 rồi sử dụng ori $t0 và 0xff=0000…0000 1111 1111 để set LSB của $s0 thành 1.

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Kết quả được lưu vào $t1:

$t0	8	0x0603ff00
$t1	9	0x0603ffff
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x0603ff22

d. Clear $s0 (s0=0, must use logical instructions):

```
1  .text
2  li $s0, 0x0603ff22 #load test value for these function
3  andi $t0, $s0, 0x00000000 #clear $s0 (s0=0, must use  logical instructions)
```

Để clear $s0 thì chỉ cần andi với 0x00000000 vì khi gặp các bit 0 thì các bit của $s0 sẽ clear về 0:

$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	0x0603ff22

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`
   `s0 <= bit invert (s0)`

d. `ble $s1,s2,L`

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

a.

```
1  .text
2  #abs
3  li $s0, -5
4  start:
5  slt $t0, $s0, $zero
6  bne $t0, $zero, else
7  j endif
8  else:
9  sub $s1, $zero, $s0
10 endif:
```

=>Nếu s0>=0 thì giữ nguyên, nếu không thì thực hiện lấy đối số của $s0 bằng lệnh sub $s1, $zero, $s0 ($s1=0-$s0).

=>Kết quả:

$s0	16	0xfffffffb
$s1	17	0x00000005

b.

```
1  .text #move
2  li $s0 0x3939
3  add $s1, $zero, $s0
```

=>Để chuyển giá trị một thanh ghi sang thanh ghi khác thì ta cộng thanh ghi đó với thanh ghi $zero:

$s0	16	0x00003939
$s1	17	0x00003939

c.

```
.text #not
li $s0, 0xf939
xori $s1, $s0, 0x7fffffff
```

Để thực hiện not (đảo bit) ta dùng lệnh xori:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Để đảo bit của 0xf939 ta đem xori với 0x7fffffff=1111 1111…1111. Do bit nào xor với 1 đều thu được bit đảo nên ta thu được kết quả:

$s0	16	0x0000f939
$s1	17	0x7fff06c6

d.

```
1   .text
2   #ble (branch if less or equal)
3   li $s1, 1
4   li $s2, 2
5   start:
6   slt $t0, $s2, $s1 #if s2<s1
7   beq $t0, $zero, else
8   sub $s3,$s1,$s2 #s3=s1-s2
9   j endif
10  else:
11  add $s3, $s1, $s2 #s3=s1+s2
12  endif:
```

Nếu $s2<$s1 thì $t0=1 và thực hiện $s3=$s1-$s2, nếu ngược lại thì thực hiện $s3=$s1+$s2. Trong trường hợp này, $s1<$s2 nên $t0=0=>thực hiện lệnh sau else là $s3=$s1+$s2 và thu được kết quả là 3:

$s1	17	0x00000001
$s2	18	0x00000002
$s3	19	0x00000003

Assignment 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.

-Đầu tiên vẫn set $t0=0 là trạng thái không bị tràn.
-Tính tổng s1+s2 và lưu vào s3 bằng lệnh addu (phép cộng nhưng không báo lỗi khi tràn số, khác add).
-Sử dụng xor $t1, $s1, $s2 để kiểm tra $s1 và $s2 có cùng dấu không:
+Nếu cùng dấu (cùng bit đầu) thì trả về kết quả lưu vào $t1 có giá trị dương (do bit đầu là 0).
+Nếu không (khác bit đầu) thì trả về kết quả lưu vào $t1 có giá trị âm (do bit đầu là 1).
-Lệnh tiếp theo là bltz (branch if less than zero)->so sánh nếu $t1<0 (tức $s1 và $s2 khác dấu) thì sẽ chuyển đến EXIT kết thúc. Nếu $t1>0 thì sẽ thực hiện lệnh tiếp.
-Khác với bài 1, ở đây không so sánh tổng $s3 với các số hạng mà đi kiểm tra dấu của $s3 với dấu các số hạng:

```
xor $t2, $s3, $s1
```

+Nếu $s3 cùng dấu 2 số hạng=>$t2=0 và không xảy ra tràn.
+Nếu ngược lại thì $t2=1 và xảy ra tràn.

TH1: Overflow:

```
.text
start:
li $s1 0x7fffffff
li $s2 3

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
xor $t2, $s3, $s1
bltz $t2, OVERFLOW #$s3, $s1 doesn't have same sign

j EXIT
OVERFLOW:
li $t0, 1 #the result is overflow
EXIT:
```

Kết quả:

$t0	8	0x00000001
$t1	9	0x7ffffffc
$t2	10	0xfffffffd
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x7fffffff
$s2	18	0x00000003
$s3	19	0x80000002

TH2: Not Overflow:

```
.text
start:
li $s1 0x234
li $s2 3

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
xor $t2, $s3, $s1
bltz $t2, OVERFLOW #$s3, $s1 doesn't have same sign

j EXIT
OVERFLOW:
li $t0, 1 #the result is overflow
EXIT:
```

Kết quả:

$t0	8	0x00000000
$t1	9	0x00000237
$t2	10	0x00000003
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000234
$s2	18	0x00000003
$s3	19	0x00000237

Assignment 5:

Assignment 5

Write a program that implement multiply by a small power of 2. (2, 4, 8, 16, etc for example).

```
1  .text
2  addi $s0, $zero, 9
3  addi $s1, $zero, 512
4  addi $t0, $zero, 1
5  loop:
6  beq $s1, $t0, exit #neu so nhan $s1=1 thi ket thuc
7  sll $s0, $s0, 1 #tang gia tri $s0 len 2 lan
8  srl $s1, $s1, 1 #giam gia tri $s1 xuong 2 lan
9  j loop
10 exit:
11  add $t3, $zero, $s0
```

-Lần lặp đầu: $s0=9, $s1=512.
-Lần lặp thứ 2: $s0=18, $s1=256.
…
-Lần lặp thứ 10: $s0=4608, $s1=1.
Sau lần lặp thứ 10 thì điều kiện beq $s1, $t0, exit được thoả mãn và vòng lặp kết thúc. Giá trị cuối của $s0 là 4608 được ghi vào thanh ghi $t3 ở hệ 10, chuyển sang hệ 16 là 0x00001200:

$t3	11	0x00001200
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00001200