Lab 4: Integer, logic and shift instructions in MIPS

Objectives:

After completing this lab, you will get familiar with the basic MIPS integer arithmetic and logic instructions including:

- Integer addition and subtraction instructions
- Bitwise logic instructions
- Shift instructions

1) Integer Add/Subtract Instructions

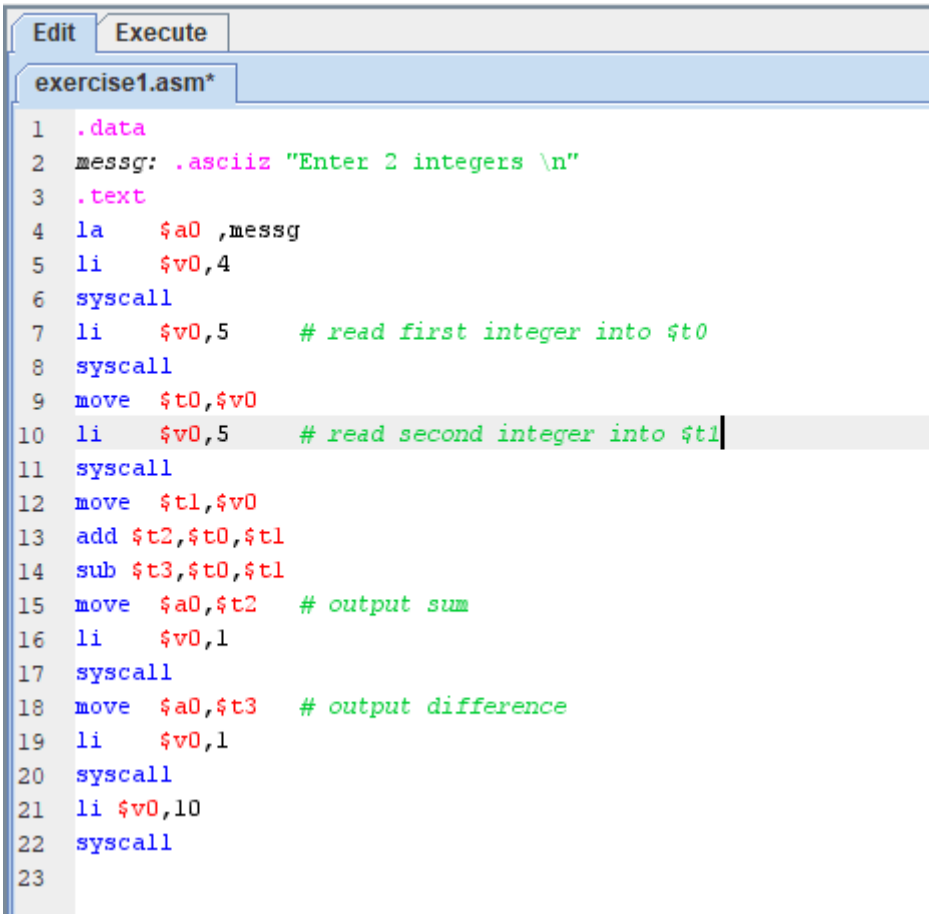
The MIPS add/subtract instructions are shown in Table 1 below. The R-type add/sub instructions have three registers where the first register is the destination register while the other two registers are the two registers to be added. Similarly the I-type add/sub instructions have the first register as the destination register, while the second register and the constant are the operands to be either added or subtracted.

Instruction	Meaning
`add $s1, $s2, $s3`	`$s1 = $s2 + $s3`
`addu $s1, $s2, $s3`	`$s1 = $s2 + $s3`
`sub $s1, $s2, $s3`	`$s1 = $s2 - $s3`
`subu $s1, $s2, $s3`	`$s1 = $s2 - $s3`
`addi $s1, $s2, 10`	`$s1 = $s2 + 10`
`addiu $s1, $s2, 10`	`$s1 = $s2 + 10`

Table 1 : Add/Subtract Instructions

The difference between **add/sub** and **addu/subu** instructions is that in case of overflow occurrence, the **add/sub** instructions will cause an arithmetic exception and the result will not be written to the destination register. However, for the instructions **addu/subu**, overflow occurrence is ignored.

Exercise 1: Write a program to ask the user to enter two integers and print their sum and difference.



Edit | Execute

exercise1.asm*

```
1  .data
2  messg: .asciiz "Enter 2 integers \n"
3  .text
4  la    $a0 ,messg
5  li    $v0,4
6  syscall
7  li    $v0,5     # read first integer into $t0
8  syscall
9  move  $t0,$v0
10 li    $v0,5     # read second integer into $t1
11 syscall
12 move  $t1,$v0
13 add $t2,$t0,$t1
14 sub $t3,$t0,$t1
15 move  $a0,$t2   # output sum
16 li    $v0,1
17 syscall
18 move  $a0,$t3   # output difference
19 li    $v0,1
20 syscall
21 li $v0,10
22 syscall
23
```

Exercise 2:

Convert the following C code into MIPS and determine the value of each variable after execution:

a=5;

b=20;

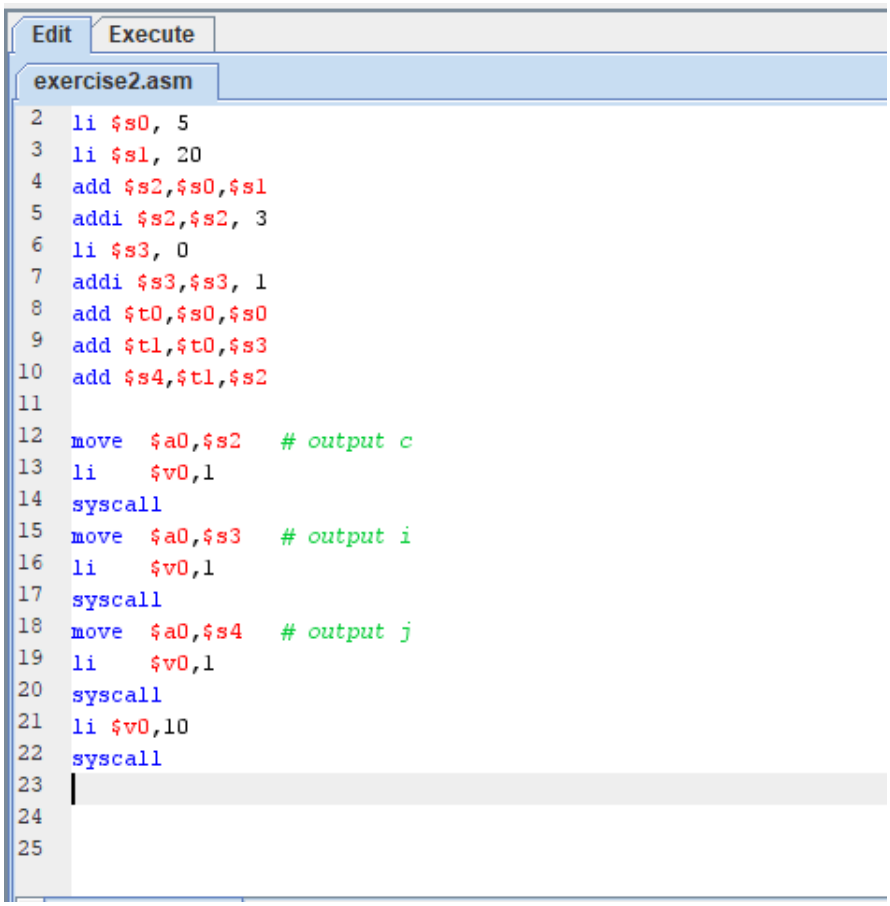
c= a+b+3;

i=0;

i++;

j=2*a+c+i;

Where a, b,c, i ,j in $s0, $s1, $s2, $s3, $s4



2) Logical Bitwise Instructions

The MIPS logical instructions are given in Table 2. These include the: and, or, nor and xor instructions. The operands for these instructions follow the same convention as the add/sub instructions. The immediate value for the andi, ori, and xori logical instructions is treated as unsigned constant, while it is treated as a signed constant for the addi and addiu instructions.

Instruction		Meaning
and	$s1, $s2, $s3	$s1 = $s2 & $s3
or	$s1, $s2, $s3	$s1 = $s2 | $s3
xor	$s1, $s2, $s3	$s1 = $s2 ^ $s3
nor	$s1, $s2, $s3	$s1 = ~($s2|$s3)
andi	$s1, $s2, 10	$s1 = $s2 & 10
ori	$s1, $s2, 10	$s1 = $s2 | 10
xori	$s1, $s2, 10	$s1 = $s2 ^ 10

Table 2 : Logical Instructions

Exercise 3:

Assume that **$s1 = 0xabcd1234** and **$s2 = 0xffff0000**, write a program contains the following instructions and from the Mars screen determine the values of registers **$s3** to **$s6**:

```
and $s3, $s1, $s2
or $s4, $s1, $s2
xor $s5, $s1, $s2
nor $s6, $s1, $s2
```

Edit | Execute

exercise2.asm | exercise3.asm

```
1  .text
2  .globl main
3  main: # main program
4  li $s1, 0xabcd1234 # Pseudo instruction to initialize a register
5  li $s2, 0xffff0000
6  and $s3,$s1,$s2 # $s3 = 0xabcd0000
7  or $s4,$s1,$s2 # $s4 = 0xffff1234
8  xor $s5,$s1,$s2 # $s5 = 0x54321234
9  nor $s6,$s1,$s2 # $s6 = 0x0000edcb
10 li $v0, 10 # Exit program
11 syscall
12
```

$s0	16	0x00000000
$s1	17	0xabcd1234
$s2	18	0xffff0000
$s3	19	0xabcd0000
$s4	20	0xffff1234
$s5	21	0x54321234
$s6	22	0x0000edcb
$s7	23	0x00000000

Exercise 4:

Assume that **$s1 = 0x12345678** and **$s2 = 0xffff9a00**. Determine the content of registers **$s3** to **$s6** after executing the following instructions:

```
and $s3,$s1,$s2        # $s3 =
or $s4,$s1,$s2         # $s4 =
xor $s5,$s1,$s2        # $s5 =
nor $s6,$s1,$s2        # $s6 =
```

Write a program to execute these instructions and verify the content of registers **$s3** to **$s6**.

3) Shift Instructions

The MIPS shift instructions are given in Table 3. The first operand of the shift instructions is the destination register, the second operand is the register to be shifted while the third operand specifies the amount of shift. The amount of shift can be specified as a constant value or it can be stored in a register. For the instructions **sll, srl, sra**, the shift amount is a 5-bit constant while for the instructions **sllv, srlv, srav,** the shift amount is variable and is stored in a register.

Instruction	Meaning
sll $s1,$s2,10	$s1 = $s2 << 10
srl $s1,$s2,10	$s1 = $s2>>>10
sra $s1,$s2,10	$s1 = $s2 >> 10
sllv $s1,$s2,$s3	$s1 = $s2 << $s3
srlv $s1,$s2,$s3	$s1 = $s2>>>$s3
srav $s1,$s2,$s3	$s1 = $s2 >> $s3

Table 3 : Shift Instructions

Exercise 5:

Assume that **$s2 = 0xabcd1234** and **$s3 = 8**. Determine the content of registers **$s4** to **$s7** after executing the following instructions:

```
sll $s4, $s2, 12                # $s4 =
sllv  $s5, $s2, $s3             #$s5 =
srl $s6, $s2, 4                 #$s6 =
sra $s7, $s2, 8                #$s7 =
```

Write a program to execute these instructions and verify the content of registers **$s4** to **$s7**

Exercise 6:

Write a program that asks the user to enter number and read it. Then display the content of multiplying this number by **38**.