# Lab Report 04

## Assignment 1

**Code:**

```
.text
init:
        j               case4
case1:
        addi            $s1, $zero, 1
        addi            $s2, $zero, 2
        j               start
case2:
        addi            $s1, $zero, 1
        addi            $s2, $zero, 0x7FFFFFFF   # 2^31 - 1
        j               start
case3:
        addi            $s1, $zero, -1
        addi            $s2, $zero, -2
        j               start
case4:
        addi            $s1, $zero, -1
        addi            $s2, $zero, 0x80000000 # -2^31
        j               start
case5:
        addi            $s1, $zero, 3
        addi            $s2, $zero, -2
        j               start
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, $s3, $s1
    bltz        $s1, NEGATIVE           # Test if $s1 and $s2 is negative?
    beq         $t2, $zero, EXIT        # s1 and $s2 are positive
          # If $s3 > $s1 then the result is not overflow
    j           OVERFLOW
NEGATIVE:
    bne         $t2, $zero, EXIT        # s1 and $s2 are negative
          # If $s3 < $s1 then the result is not overflow
OVERFLOW:
    li          $t0, 1                  # The result is overflow
EXIT:
```

**Comments:**

- Case1: 2 positive, no overflow
- Case2: 2 positive, overflow
- Case3: 2 negative, no overflow
- Case4: 2 negative, overflow
- Case5: 1 positive, 1 negative, no overflow

## Assignment 2

**Code:**

```
.text
    li          $s0, 0x12345678         # Load test value
    srl         $s1, $s0, 24            # Extract MSB
    andi        $s2, $s0, 0xFFFFFF00    # Clear LSB
    ori         $s3, $s0, 0x000000FF    # Set LSB
    xor         $s0, $s0, $s0           # Clear $s0
```

## Assignment 3

a. **abs            $s0, $s1**

```
.text
```

```
init:
      addi          $s1, $zero,-1
start:
      addu          $s0, $s1, $zero
      bgez          $s1, exit
      sub           $s0, $zero, $s1
exit:
```

**b. move $s0, $s1**

```
      add           $s0, $s1, $zero
```

**c. not $s0, $s1**

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

**d. ble $s1, $s2, label**

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

## Assignment 4

**Code:**

```
.text
init:
            j             case2
case1:
            addi          $s1, $zero, 1
            addi          $s2, $zero, 2
            j             start
case2:
            addi          $s1, $zero, 1
            addi          $s2, $zero, 0x7FFFFFFF  # 2^31 - 1
            j             start
start:
            li            $t0, 0                  # No Overflow is default status
            addu          $s3, $s1, $s2           # s3 = s1 + s2
```

```
        xor         $t1, $s3, $s1           # Test if $s3 and $s1 have the same sign
        beq         $t1, $zero, EXIT
        j           OVERFLOW
OVERFLOW:
        li          $t0, 1                  # The result is overflow
EXIT:
```

## Assignment 5

**Code:**

```
.text
init:
        li          $s0, 0                  # Used to store result
        li          $s1, 0x12
        li          $s2, 32
        li          $s3, 0                  # Position of first 1
        li          $t1, 1                  # Used to store number 1
count:
        beq         $s2, $t1, multiply      # Exit if $s2 == 1
        srl         $s2, $s2, 1             # Shift right by 1 bit
        addi        $s3, $s3, 1             # Count shifting operations
        j           count
multiply:
        sllv        $s0, $s1, $s3           # Multiply by shifting
```

**Comments:**

- Use count loop to divide $s2 until it equals to 1.
    - Each time a division was executed (right shifting), we update $s3 = $s3 + 1
    - By doing this, we will eventually get $s3 = log2($s2), which is the number of shifting bits.
- In the multiply label, simply shift $s1 by $3 to get the result.