CS224
Section No.: 2
Spring 2019
Lab No. 2
Munib Emre Sevilgen / 21602416

Part 1:

```
        .data
enterOctalNumber: .asciiz "\nEnter an octal number: "
enterNumber: .asciiz "\nEnter a decimal number: "
inputNumber: .space 20
menu: .asciiz "\n\n1.First\n2.Second\n3.Third\nEnter the number of your selection: "
result: .asciiz "\nThe result: "
errorMsg: .asciiz "\nError: Invalid number\n"
errorSelection: .asciiz "\nError: Invalid selection\n"

        .text
main:
        jal interactWithUser

end:
        li $v0, 10
        syscall

interactWithUser:
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        getSelection:
                li $v0, 4
                la $a0, menu
                syscall

                li $v0, 5
                syscall
                move $s0, $v0
                beq $s0, 1, jumpFirst
                beq $s0, 2, jumpSecond
                beq $s0, 3, interactWithUser_exit

        #Selection Error
        li $v0, 4
        la $a0, errorSelection
        syscall
```

```
        j getSelection

        jumpFirst:
                jal first
                j getSelection
        jumpSecond:
                jal second
                j getSelection

        interactWithUser_exit:
                lw $ra, 0($sp)
                addi $sp, $sp, 4
                jr $ra

first:
        addi $sp, $sp, -4
        sw $s0, 0($sp)
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        getNumber1:
                li $v0, 4
                la $a0, enterOctalNumber
                syscall

                li $v0, 8
                la $a0, inputNumber
                li $a1, 20
                syscall

        jal convertToDec
        move $s0, $v0
        beq $s0, -1, getNumber1 #Checks the error output

        li $v0, 4
        la $a0, result
        syscall

        li $v0, 1
        move $a0, $s0
        syscall

        lw $ra, 0($sp)
```

```
        addi $sp, $sp, 4
        lw $s0, 0($sp)
        addi $sp, $sp, 4
        jr $ra


second:
        addi $sp, $sp, -4
        sw $s0, 0($sp)
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        li $v0, 4
        la $a0, enterNumber
        syscall

        li $v0, 5
        syscall
        move $a0, $v0

        jal reverseNumber
        move $s0, $v0

        li $v0, 4
        la $a0, result
        syscall

        #Print in hexadecimal format
        li $v0, 34
        move $a0, $s0
        syscall

        lw $ra, 0($sp)
        addi $sp, $sp, 4
        lw $s0, 0($sp)
        addi $sp, $sp, 4
        jr $ra

convertToDec:
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        la $s0, ($a0) #Adress of first index in the string
        jal stringLength #Gets the length of the string
```

```
        move $s1, $v0 #Length of the string

        #Adress of the last index in the string
        add $s2, $s0, $s1
        subi $s2, $s2, 1

        li $s4, 1 #Multiplier for the digits
        li $s5, 0 #Sum
        sumLoop:
                ble $s1, $zero, sumLoop_exit
                addi $s1, $s1, -1
                lb $s3, 0($s2) #Last character in the string
                addi $s2, $s2, -1
                #Checks whether the digit is vaid or not
                blt $s3, '0', error
                ble $s3, '7', sum

        sum:
                addi $s3, $s3, -48
                mul $s3, $s3, $s4
                sll $s4, $s4, 3 #Multiply multiplier by 8
                add $s5, $s5, $s3
                j sumLoop

        error:
                li $v0, 4
                la $a0, errorMsg
                syscall

                #Return -1 if there is an error
                li $v0, -1
                lw $ra, 0($sp)
                addi $sp, $sp, 4
                jr $ra

        sumLoop_exit:
                move $v0, $s5
                lw $ra, 0($sp)
                addi $sp, $sp, 4
                jr $ra

reverseNumber:
        addi $sp, $sp, -4
        sw $ra, 0($sp)
```

```
        move $s0, $a0 #Decimal input
        li $s1, 0 #Decimal result

        reverseNumber_loop:
                beq $s0, $zero, reverseNumber_exit #Exits if the input becomes 0
                sll $s1, $s1, 4 #Shift left the result one hex digit by 4 bits
                andi $s2, $s0, 15 #And by 1111 to get the last 4 bits
                add $s1,$s1, $s2 #Add last four bits to the result
                srl $s0, $s0, 4 #Shift right the input one hex digit by 4 bits
                j reverseNumber_loop

        reverseNumber_exit:
        move $v0, $s1
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra

stringLength:
        addi $sp, $sp, -4
        sw $s0, 0($sp)
        addi $sp, $sp, -4
        sw $ra, 0($sp)
        li      $s0, 0
        li      $s2, 0

        stringLength_loop:
                add     $s2, $a0, $s0 #Adress of the current char
                lb      $s1, 0($s2) #Current char
                beq     $s1, $zero, stringLength_exit #End of the string
                addi    $s0, $s0, 1 #Counter
                j       stringLength_loop

        stringLength_exit:
                subi    $s0, $s0, 1
                add     $v0, $zero, $s0
                add     $s0, $zero, $zero

                lw $ra, 0($sp)
                addi $sp, $sp, 4
                lw $s0, 0($sp)
                addi $sp, $sp, 4
                jr $ra
```

Part 2:

beq $t0, $t1, next: 0x11090004
This is an I type instruction and it jumps to 4th instruction below, so the immediate is 4.

Opcode	Rs	Rt	Immediate
000100	01000	01001	0000 0000 0000 0100

bne $t2, $t3, again: 0x154BFFFA
This is an I type instruction and it jumps to 6th instruction above, so the immediate is -6.

Opcode	Rs	Rt	Immediate
000101	01010	01011	1111 1111 1111 1010

jr $ra: 0x03e00008
This is an R type instruction.

Opcode	Rs	Rt	Rd	Shamt	Funct
000000	11111	00000	00000	00000	001000

j again: 0x08100010
This is an R type instruction. By pass first four bits by pc and omitting the last bits, we get the address.
0x10010040 = 0001 0000 0000 0001 0000 0000 0100 0000

Opcode	Address
000010	0000 0000 0001 0000 0000 0100 00