CS224
Section No.: 2
Spring 2019
Lab No. 1
Munib Emre Sevilgen / 21602416

1. Question:

```
	.data
array: .space 80
sizeMessage: .asciiz "Enter the number of elements (at most 20): "
maxSizeError: .asciiz "Error: Invalid number of elements. Please enter the number of elements (at most 
20): "
enterElement: .asciiz "Enter an element: "
messageOfArray: .asciiz "Array: "
messageOfReversedArray: .asciiz "\nReversed array: "

	.text
main:
	#Shows the size message to the user
	li $v0, 4
	la $a0, sizeMessage
	syscall

	#Gets the size and puts it to $t0
	li $v0, 5
	syscall
	move $t0, $v0

	#Checks whether the size is less than or equal to 20
	li $t1, 20
	slt, $t2, $t1, $t0

getSize:
	beq $t2, $zero, getElements #If the size input is valid then it jumps to the getElements part

	#Shows the error message to the user and gets the size again
	li $v0, 4
	la $a0, maxSizeError
	syscall

	li $v0, 5
```

```
            syscall

            move $t0, $v0
            slt $t2, $t1, $t0

            j getSize #Jumps to the beginning of the getSize to check the size input again

getElements:
            li $t1, 0 #Counter i of the for loop
            la $s0, array #Makes $s0 point to the beginning of the array

forOfGetElements:#For loop to get the elements one by one
            beq $t1, $t0, displayArray #Exits the loop and continues with displaying the contents of the
array

            #Displays a message to make the user enter an element
            li $v0, 4
            la $a0, enterElement
            syscall

            #Gets the element and move it to the $t2
            li $v0, 5
            syscall
            move $t2, $v0

            #Adds the element to the array and passes to the next index of the array
            sw $t2, 0($s0)
            addi $s0, $s0, 4

            #Increments the counter
            addi $t1, $t1, 1
            j forOfGetElements

displayArray:
            #Gets the first and the last index location of the array
            la $t1, array
            la $t2, ($t0)
            sll $t2, $t2, 2 #Multiply $t2 to get the lenght of the array in the memory
            add $t3, $t1, $t2

            #Displays a message to make the user enter an element
            li $v0, 4
```

```
        la $a0, messageOfArray
        syscall

forOfDisplayArray: #For loop to display the array
        beq $t1, $t3, reverseArray #Exits the loop and continues with reversing the contents of the array

        #Load the integer to $a0 and displat it
        lw $a0, ($t1)
        li $v0, 1
        syscall

        #Increments the counter
        addi $t1, $t1, 4
        j forOfDisplayArray

reverseArray:
        #Gets the first and the last index location of the array
        la $t1, array
        la $t2, ($t0)
        sll $t2, $t2, 2 #Multiply $t2 to get the lenght of the array in the memory
        add $t3, $t1, $t2

forOfReverseArray: #For loop to reverse the array
        slt $t6, $t1, $t3
        beq $t6, $0, displayReversedArray
        addi $t3, $t3, -4
        lw $t5, ($t3)
        lw $t4, ($t1)
        sw $t5, ($t1)
        sw $t4, ($t3)
        addi $t1, $t1, 4
        j forOfReverseArray

displayReversedArray:
        la $t1, array

        la $t2, ($t0)
        sll $t2, $t2, 2 #Multiply $t2 to get the lenght of the array in the memory
        add $t3, $t1, $t2

        #Displays a message to make the user enter an element
        li $v0, 4
        la $a0, messageOfReversedArray
        syscall

forOfDisplayReversedArray: #For loop to display the array
        beq $t1, $t3, done #Exits the loop and continues with reversing the contents of the array

        #Load the integer to $a0 and displat it
        lw $a0, ($t1)
        li $v0, 1
        syscall

        #Increments the counter
        addi $t1, $t1, 4
        j forOfDisplayReversedArray

done:
        li $v0, 10
        syscall
```

2. Question:

```
        .data
getStringMessage: .asciiz "Enter a string: "
palindrome: .asciiz "The string is a palindrome."
notPalindrome: .asciiz "The string is not a palindrome."
stringInput: .space 30 #Space for input string

        .text
main:
        #Display the get string message and puts the input to $a0
        li $v0, 4
        la $a0, getStringMessage
        syscall

        li $v0, 8
        la $a0, stringInput # gets input
        li $a1, 30
        syscall

        la $s1, stringInput
```

```
stringLengthLoop: #Counts the length of the string
        lb $t0, 0($s1)
        beq $t0, $zero, stringLengthLoopExit
        addi $s1, $s1, 1
        j stringLengthLoop

stringLengthLoopExit:
        la $s0, stringInput #First index of the string
        addi $s1, $s1, -2 #Last index of the string


palindromeLoop: #Checks whether the string is palindrome or not
        bge $s0, $s1, displayPalindrome
        bne $t0, $t1, displayNotPalindrome
        lb $t0, 0($s0)
        lb $t1, 0($s1)
        addi $s1, $s1, -1
        addi $s0, $s0, +1
        j palindromeLoop

displayNotPalindrome: #Display the string is not palindrome
        li $v0 , 4
        la $a0, notPalindrome
        syscall
        j end

displayPalindrome: #Display the string is palindrome
        li $v0, 4
        la $a0, palindrome
        syscall
        j end

end:
        li $v0, 10
        syscall
```

3. Question:

```
        .data
enterC: .asciiz "Enter c: "
enterD: .asciiz "Enter d: "
```

```
        .text
main:
        #Gets c and puts it to $s1
        li $v0, 4
        la $a0, enterC
        syscall
        li $v0, 5
        syscall
        move $s1, $v0

        #Gets d and puts it to $s2
        li $v0, 4
        la $a0, enterD
        syscall
        li $v0, 5
        syscall
        move $s2, $v0

        sub $s0, $s1, $s2
        rem $s0, $s0, 16

        #Displays result
        li $v0, 1
        la $a0, ($s0)
        syscall

        li $v0, 10
        syscall
```

4. Question:

```
la      $t1, a:
        lui $1, 0x00001001              0x3c011001
        ori $9, $1, 0x00000014          0x34290014

la      $t2, b:
        lui $1, 0x00001001              0x3c011001
        ori $10, $1, 0x00000014         0x342a0020

lw      $t2, b:
```

	lui $1, 0x00001001	0x3c011001
	lw $10, 0x00000020($1)	0x8c2a0020
lw	$t2, b:	
	lui $1, 0x00001001	0x3c011001
	lw $10, 0x00000020($1)	0x8c2a0020

5. Question:

Symbolic machine instruction: These instructions are the translations of the machine instructions to make them human readable. By these instructions, humans can program the CPU's and this is called low level programming.
add $t0, $t1, $t2: This instruction sums the values in the register t1 and t2 and puts this sum to the register t0.
lw $t0, 0x0004($0): This instruction loads the content of the address 4 to the register $t0.

Machine instruction: These instructions are the binary instructions that machine can store, read and understand. These instructions executed by the machine directly.
add $t0, $t1, $t2: 0x012A4020
lw $t0, 0x0004($0): 0x8C080004
These hexadecimal instructions are the translations of the corresponding symbolic machine instructions.

Assembler directive: Instructions that directs the assembler to do some specific translations
.data: The data items below will be stored in the data segment of the memory.
.space n: Assembler allocates n bytes of space.

Pseudo instruction: These instructions help the users to implement more complicated tasks.

la $t1, a:	lui $1, 0x00001001
	ori $9, $1, 0x00000014
lw $t2, b:	lui $1, 0x00001001
	lw $10, 0x00000020($1)