PRELIMINARY REPORT

Lab 01

MUHAMMAD ARHAM KHAN

21701848 - CS

SECTION 06

Dated: 25 February, 2019

PART A: ARRAY TASK

```
.text
        la $a0,prompt  # output prompt message on terminal
        li $v0,4       # syscall 4 prints the string
        syscall

        #input the integer
        li $v0, 5      # syscall 5 reads an integer
        syscall

        #comparing the number to see if its less than 20
        blt $v0,21,func

        li $v0,10
        syscall

        func:
                #array first index
                lui $s0, 0x1000
                ori $s0, $s0, 0x7000

                add $t0, $0, $0
                add $t1, $0, $v0

                loopA:
                        beq     $t0, $t1, funcB

                        la $a0,promptB # output prompt message on terminal
                        li $v0,4       # syscall 4 prints the string
                        syscall
                        li $v0, 5      # syscall 5 reads an integer
                        syscall

                        sw $v0, 0($s0)
                        addi $s0, $s0, 4

                        addi    $t0, $t0, 1
                        j       loopA

        funcB:
                lui     $s0, 0x1000
                ori     $s0, $s0, 0x7000
                add     $t0, $0, $0

                loopB:
                        beq     $t0, $t1, funcC

                        #outputting the current value of array
                        lw      $t5, 0($s0)
                        li      $v0,1
                        move    $a0, $t5
                        syscall

                        la      $a0,newline
                        li      $v0,4
                        syscall

                        addi    $s0, $s0, 4
                        addi    $t0, $t0, 1
                        j       loopB

        funcC:
                lui     $s1, 0x1000
                ori     $s1, $s1, 0x7000 #final array pointer
```

```
        add     $t0, $0, $0
        addi    $t2, $t1, -1

        loopC:
                beq     $t0, $t2, funcD
                addi    $s1, $s1, 4
                addi    $t0, $t0, 1

                j       loopC

funcD:

        lui     $s0, 0x1000
        ori     $s0, $s0, 0x7000 #initial array pointer

        sra     $t4, $t1, 1
        add     $t0, $0, $0

        loopD:
                bge     $t0, $t4, funcE

                lw      $t6, 0($s0)
                lw      $t7, 0($s1)
                sw      $t7, 0($s0)
                sw      $t6, 0($s1)

                addi    $t0, $t0, 1
                addi    $s0, $s0, 4
                addi    $s1, $s1, -4

                j       loopD

funcE:
        la      $a0, reversed  # output prompt message on terminal
        li      $v0,4   # syscall 4 prints the string
        syscall
        add     $t0, $0, $0
        lui     $s0, 0x1000
        ori     $s0, $s0, 0x7000 #initial array pointer

        loopE:
                beq     $t0, $t1, endCall

                lw      $t5, 0($s0)
                li      $v0,1
                move    $a0, $t5
                syscall

                la      $a0,newline
                li      $v0,4
                syscall

                addi    $s0, $s0, 4
                addi    $t0, $t0, 1
                j       loopE

funcE:
        endCall:
        li      $v0,10          # system call to exit
        syscall         #    bye bye

.data
        prompt:         .asciiz "Enter number of array elements lower than or equal to 20: "
        promptB:        .asciiz "Enter number to be entered: "
        newline:        .asciiz "\n"
        reversed:       .asciiz "the reversed numbers are: \n"
```

PART B: PALINDROMES

```
.text
        la      $a0, prompt
        li      $v0,4
        syscall

        #inputting string
        la      $a0, stringBuffer
        li      $a1, 50
        li      $v0,8
        syscall

        add     $t0, $a0, $0#t0 is initial index
        add     $t9, $a0, $0
        add     $t3, $0, $0 #t3 is the length

        calculateLength:
                #finding the string length
                lb      $t1, 0($t0)
                beq     $t1, $0, findFinalIndex

                addi    $t0, $t0, 1
                addi    $t3, $t3, 1

                j       calculateLength

        findFinalIndex:
                add     $t0, $t9, $0
                add     $t1, $t0, $0 #t1 is the final index
                add     $t4, $0, $0
                addi    $t5, $t3, -2 #temp variable to iterate to lower length

                loopIndex:
                        beq     $t4, $t5, checkPalindrome

                        addi    $t1, $t1, 1
                        addi    $t4, $t4, 1

                        j       loopIndex

        checkPalindrome:
                #checking the palindromes through loop
                bgt     $t0, $t1, palindromeOutput
                lb      $t4, 0($t0)
                lb      $t5, 0($t1)
                bne     $t4, $t5, notPalindromeOutput

                addi    $t0, $t0, 1
                addi    $t1, $t1, -1

                j       checkPalindrome

        palindromeOutput:
                la      $a0, pal
                li      $v0,4
                syscall
                li      $v0,10
                syscall

        notPalindromeOutput:
                la      $a0, notPal
                li      $v0,4
                syscall
```

```
                li      $v0,10
                syscall

.data
        stringBuffer:  .space 50
        prompt:        .asciiz "Enter the string for palindrome check: "
        pal:           .asciiz "Yes, it is a palindrome."
        notPal:        .asciiz "No, it is not a palindrome."
```

PART C: REMAINDERS

```
.text
        la      $a0, prompt
        li      $v0,4
        syscall

        la      $a0, prompt1
        li      $v0,4
        syscall

        li      $v0, 5  # input the value of c
        syscall
        add     $t0, $v0, $0 #t0 is c

        la      $a0, prompt2
        li      $v0,4
        syscall

        li      $v0, 5  # input the value of d
        syscall
        add     $t1, $v0, $0 #t1 is d
        sub     $t2, $t0, $t1 #t2 stores the c - d

        blt     $t2, $0, fixNegative

        #output if the difference is positive
        sra     $t3, $t2, 4
        mul     $t4, $t3, 16
        sub     $t4, $t2, $t4

        move    $a0,$t4 # print result
        li      $v0, 1
        syscall

        li      $v0,10
        syscall

        fixNegative:
                sub     $t3, $0, $t2

                #output if the difference is positive
                sra     $t4, $t3, 4
                mul     $t5, $t4, 16
                sub     $t5, $t3, $t5
                sub     $t5, $0, $t5

                move    $a0,$t5 # print result
                li      $v0, 1
                syscall

                li      $v0,10
                syscall

.data
        prompt: .asciiz "Evaluation of x = (c - d) % 16\n==============================\n"
        prompt1: .asciiz "Enter the value of c: "
        prompt2: .asciiz "Enter the value of d: "
```

PART D: INSTRUCTIONS TO HEX CODE

A. la $t1, a

Partial instructions:

1. lui $at, 4097

001111	00000	00001	0001000000000001

Machine instructions: 00111100000000010001000000000001
Hex instructions: 0x3C011001

2. ori $t1, $at, 20

001101	00001	01001	0000000000010100

Machine instructions: 00110100001010010000000000010100
Hex instructions: 0x34290014

B. la $t2, b

Partial instructions:

1. lui $at, 4097

001111	00000	00001	0001000000000001

Machine instructions: 00111100000000010001000000000001
Hex instructions: 0x3C011001

2. ori $t2, $at, 32

001101	00001	01010	0000000000100000

Machine instructions: 00110100001010100000000000100000
Hex instructions: 0x342A0020

C. lw $t2, b

Partial instructions:

1. lui $at, 4097

001111	00000	00001	0001000000000001

Machine instructions: 00111100000000010001000000000001
Hex instructions: 0x3C011001

2. lw $t2, 32($at)

100011	00001	01010	0000000000100000

Machine instructions: 10001100001010100000000000100000
Hex instructions: 0x8C2A0020

D. lw $t2, b

Partial instructions:

1. lui $at, 4097

001111	00000	00001	0001000000000001

Machine instructions: 00111100000000010001000000000001
Hex instructions: 0x3C011001

2. lw $t2, 32($at)

100011	00001	01010	0000000000100000

Machine instructions: 10001100001010100000000000100000
Hex instructions: 0x8C2A0020

PART E: DEFINITIONS

A. Symbolic Machine Instruction

Commands to a computer in a human-readable format.
Examples: “add $t0, $t1, $t2” or “srl $t0, $t1, 2”

B. Machine Instruction

Commands to a computer in a computer-readable format (1’s or 0’s).
Example: (00000001001010100100000000100000) add $t0, $t1, $t2
(00000000000010010100000010000010) srl $t0, $t1, 2

C. Assembler Directive

Assembler directive are instructions/ commands to the assembler as a detail to the programs execution to do something.

Example: “.data” or “.asciiz”

D. Pseudo Instruction

Macros for the assembly language that perform multiple instructions using a single assembly language command. The assembler expands these instructions and performs them.

Example: li $s0, 0x1234AA77 (“lui $s0, 0x1234” and “ori $s0, 0xAA77”)

move $s1, $s2 (“add $s2, $s1, $0”)