CS 224
COMPUTER ORGANIZATION

PRELIMINARY DESIGN REPORT

LAB 01

BERK YILDIZ
21502040
SECTION 4

1)

```
.data
array0:
     prompt: .asciiz "Enter the values (Max 20 values - Enter 0
to terminate): \n"
     print: .asciiz " "
     end: "\nEnd"
   .text
     main:
          #create array
          addi $t1, $zero, 0
          addi $v0, $zero, 4
          la $a0, prompt
          syscall
     topFirst:
          #reverse array
               addi $v0, $zero, 5
               syscall
               beq $v0, 0, done
               sw $v0, array0($t1)
               addi $t1, $t1, 4
      bottomFirst:   bne $t1, 80, topFirst
             done:  addi $t1, $t1, -4
       topSecond: lw $t2, array0($t1)
               addi $t1, $t1, -4
               beq $t2, 0, bottomSecond
               addi  $v0, $zero, 1
```

```
                add  $a0, $zero, $t2
                syscall
                addi $v0, $zero, 4
                la $a0, print
                syscall
       bottomSecond:      bne $t1, -4, topSecond
            addi $v0, $zero, 4
            la $a0, end
            syscall
```

2)

```
.data
     true: .asciiz "Palindrome!"
     false: .asciiz "Not palindrome!"
     prompt: .asciiz "Please enter a string: "
     input: .space 50
.text
     main:
          addi $v0, $zero, 4
          la $a0, prompt
          syscall
          addi $v0, $zero, 8
          la $a0, input
          addi $a1, $zero, 50
          syscall
          addi $v0, $zero, 4
          la $a0, input
          syscall
     size:
          lb $t0, input($t1)
```

```
        beq $t0, $zero, done
        addi $t1, $t1, 1
        j size
    done:
        addi $t1, $t1, -2
    palindromeCheck:
        lb  $t2, input($t3)
        lb $t4, input($t1)
        slt $t5, $t1, $t3
        bne $t2, $t4, falsePalindrome
        beq $t5, 1, truePalindrome
        addi $t3, $t3, 1
        addi $t1, $t1, -1
        j   palindromeCheck
    truePalindrome:
        addi $v0, $zero, 4
        la   $a0, true
        syscall
        j end
    falsePalindrome:
        addi $v0, $zero, 4
        la   $a0, false
        syscall
    end:
       addi $v0, $zero, 10
       syscall
```

3)

```
.data
.text
          li $t1, 85  # t1=9
          li $t2, 2  # t2=4
          li $s1, 8  # s1=2
          sub $t3, $t1, $t2  #t3 = t1 - t2
          #decrements the t3 until 8 to check remainder
     loop:
          sub $t3, $t3, $s1
          slt $t4, $t3, $s1
          beqz $t4, loop
         #print
          li $v0, 1
          move $a0, $t3
          syscall
```

4)

la $t1, a

lui $1, 0x00001001 0x3c011001

ori $9, $1, 0x00000014 0x34290014

la $t2, b

lui $1, 0x00001001 0x3c011001

ori $10, $1, 0x00000014 0x342a0014

5)

Symbolic Machine Instruction:

Already assembly language is a symbolic machine instruction. Machine instructions consists of 1s and 0s and symbolic machine instructions make these machine instructions human-readable because machine instructions may not be understandable for human.

Ex: (add $t1, $t2, $t3) , (bne $t0, $t1, $t2)

Machine Instruction

Machine instruction consists of code which is understandable by the machine. Machine instructions are executed by CPU. The processor looks at machine instructions one by one and performs one operation for each machine instruction.

Ex: (addi: 001000) , ($t4: reg 12 (011002)

Assembler Directive

Assembler directives are instructions that direct the assembler to do something. By assembler directives, assembler takes message for how to continue to process.

Ex: (.space) , (.asciiz)

Psuedo Instruction

Some operations can be performed with help of other instructions. Pseudo instructions are not real instructions however they can be coded in assembly language, and assembler will expand them to real instructions.

Ex: (move $t, $s) , (li $t, C)