**Preliminary Work / Preliminary Design Report**

CS224

Section No.: 05

Spring 2018

Lab No.: 01

Asuman Aydın/21502604

**1**. Write a MIPS program that

Creates an array of maximum size of 20 elements that asks the user first the number of elements and then enters the elements one by one.

Displays array contents

Reverses the array contents and displays the array (for example 1, 2, 3 becomes 3, 2, 1).

**Solution:**

.data

array: .space 80

x: .word 20

space : .asciiz " "

newLine: .asciiz "\n"

askSize: .asciiz "Enter array size which is less than or equal to 20:"

ask: .asciiz "Enter the elements: "

warning: .asciiz "the size should be less than or equal to 20 "

.text

.globl main

main:

li $t1, 1

la $s0, array

li $v0, 4 # print askSize

la $a0, askSize

syscall

li $v0, 5

syscall

move $s2,$v0

lw $s1, x

slt $t1,$s1,$s2 # checks if $s1 > $s2

li $v0, 4 # print askSize

la $a0, warning #printing the warning

syscall

li $v0, 5

syscall

beq $t1,1,main

while:

beq $t1, $s2 , endwhile

li $v0, 4 # we print space between each output

la $a0, ask

syscall

li $v0, 5

syscall

sw $v0, 0($s0)

addi $t1, $t1, 1

addi $s0, $s0, 4

j while

endwhile:

print:

beq $t0, $s0, endPrint

lw $t6, array( $t0)

addi $t0, $t0, 4

li $v0, 1

move $a0, $t6

syscall

li $v0, 4 # we print space between each output

la $a0, space

syscall

bgt $t0,1, print

j print

endPrint:

li $v0, 4 # we print space between each output

la $a0, newLine

syscall

printReverse:

addi $t1, $t1, -1 # counting with -1

addi $s0, $s0, -4 # also counting down in address

li $v0, 1

lw $a0, 0($s0)

syscall

li $v0, 4 # we print space between each output

la $a0, space

syscall

bgt $t1, 0, printReverse # if there is more element, continue print

#endOfProgram

li $v0, 10

syscall

**2**. Write a MIPS program that

Implements the following expression without using div. If necessary use instructions other that we have seen in the class.

x= (c - d) % 2

**Solution:**

.data

c: .word

d: .word

a: .word # difference of c and d

number: .asciiz "Please enter first number: "

number2: .asciiz "Please enter second number: "

Result: .asciiz "The difference of c and d is: "

.text

# I subtract c and d first

#then I said if I subtract the difference with 2 in loop everytime

#and it stops if the differnce is less than 2

#so the remainder will be either 1 or 0

main:

li $v0, 4

la $a0, number

syscall

#read the integer

li $v0, 5

la $t0, c

syscall

move $t0, $v0

sw $t0, c

li $v0, 4

la $a0, number2

syscall

#Read second integer

li $v0, 5

la $t1, d

syscall

move $t1, $v0

sw $t1, d

la $t2, a

sub $t2, $t0, $t1

sw $t2, a

la $a0, Result

li $v0, 4

syscall

lw $a0, a

li $v0, 1

syscall

while :

sub $a0, $a0, 2 #subtract 2 from the difference everytime till it is less than 2

li $v0, 1

syscall

bgt $a0, 2, while

exit:

sw $a0, a

#EndOfProgram

li $v0, 10

syscall

**3**. Generate the object code (in hex) for the following la instructions. Show your work for the intermediate steps (please remember that la is a pseudo instruction and implemented by two instructions).

la $t1, a

la $t2, b

lui $1, 0x00001001

ori $9, $1, 0x00000008

lui $1, 0x00001001

ori $10, $1 , 0x00000018

.data

str: .asciiz "\nHello\n"

a: .word 1, 2, 3, 4

b: .word 1

**4**. Define the following terms by providing two examples for each of them.

a. Symbolic machine instruction:

These are the core instruction that we usually use for arithmetic instructions and also storage.

-add

-sub

b. Machine instruction:

it is the instruction that is encoded as binary numbers in a format.

-R type

-I type

c. Assembler directive:

These are the instruction that is been used for storing the bookkeeping variables and perform them. They are starting with a period.

-.word

-.asciiz

d. Pseudo instruction:

These instructions do not have pointed functionality in the assembly and the program does not know them but it is produced as the program understand what you are pointing to do.

-la

-bgt