Midterm Report

## Assignment 1

**Code:**

.data

Message1: .asciiz "Enter N (> 0): "

Message2: .asciiz "Invalid input!\n"

Message3: .asciiz "N is a perfect square!\n"

Message4: .asciiz "N is not a perfect square!\n"

.text

init: li $s0, 0 # N

main: li $v0, 4 # print string

la $a0, Message1

syscall

li $v0, 5 # input integer

syscall

add $s0, $zero, $v0 # update N

ble $s0, $zero, printInvalid # invalid if N <= 0

add $a0, $zero, $s0 # pass N to isSquare()

jal isSquare

beq $v0, 1, printTrue

beq $v0, 0, printFalse

j endMain

printInvalid: li $v0, 4 # print string

la $a0, Message2

syscall

j endMain

printTrue: li $v0, 4 # print string

la $a0, Message3

syscall

j endMain

printFalse: li $v0, 4 # print string

la $a0, Message4

syscall

j endMain

################################################################################

# function isSquare:

# param[in] $a0 the interger needed to be checked

# return $v0 boolean

################################################################################

isSquare: add $sp, $sp, -12 # expand stack

sw $ra, 08($sp) # save

sw $s0, 04($sp) # save

sw $s1, 00($sp) # save

isSquareInit: li $s0, 1 # i

li $s1, 1 # i^2

li $v0, 0 # 0 = false, 1 = true

isSquareLoop: bgt $s0, $a0, endIsSquareLoop # exit i > n

mul $s1, $s0, $s0 # i \* i, 32-bit result

beq $s1, $a0, foundSquare # exit if i^2 == n

add $s0, $s0, 1 # i++

j isSquareLoop

foundSquare: li $v0, 1 # true

j endIsSquareLoop

endIsSquareLoop: lw $s1, 00($sp) # load

lw $s0, 04($sp) # load

lw $ra, 08($sp) # load

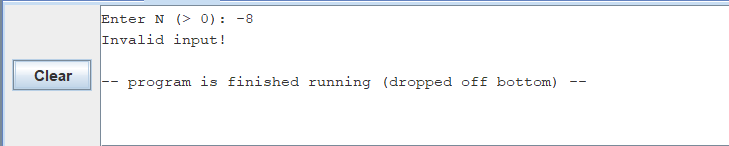
add $sp, $sp, +8 # shrink stack

jr $ra # return

################################################################################

endMain:

**Results:**

* ****
* **Graphical user interface, text, application, Word

  Description automatically generated**
* **Graphical user interface, application

  Description automatically generated**

**Comments:**

* The program uses ` isSquare` function with parameter $a0 being the interger needed to be checked. The return value $v0 will be 0 or 1, corresponding to false or true.
* The programs can handle invalid integers <= 0.

## Assignment 2

**Code:**

.data

Message1: .asciiz "Enter size: "

Message2: .asciiz "Enter element: "

Message3: .asciiz "Found satisfied odd: "

Message4: .asciiz "Found no odds!\n"

.text

init: li $s0, 0 # N = size

li $s1, 0 # 4 \* N = size in bytes

# address(A[0]) = address($sp) = 0x7fffeffc

li $s2, 0 # i

li $s3, 0 # 4 \* i

li $s4, 0 # address(A[i])

main: li $v0, 4 # print string

la $a0, Message1

syscall

li $v0, 5 # input integer

syscall

add $s0, $zero, $v0 # update N

sll $s1, $s0, 2 # 4 \* N

sub $sp, $sp, $s1 # expand stack

input: beq $s2, $s0, endInput # exit if i == n

sll $s3, $s2, 2 # 4 \* i

add $s4, $s3, 0x7fffeffc # address(A[i])

li $v0, 4 # print string

la $a0, Message2

syscall

li $v0, 5 # input integer

syscall

sw $v0, ($s4) # push A[i] to address(A[i])

add $s2, $s2, 1 # i++

j input

endInput: li $a0, 0x7fffeffc # address of array

add $a1, $zero, $s0 # length

jal minEven

li $a0, 0x7fffeffc

add $a1, $zero, $s0 # length

add $a2, $zero, $v0 # minEven

jal maxOdd

add $t0, $zero, $v0 # store result

beq $t0, -100000, noOdds # no odds found

li $v0, 4 # print string

la $a0, Message3

syscall

li $v0, 1 # print integer

add $a0, $zero, $t0

syscall

li $v0, 11 # print character

li $a0, '\n'

syscall

j endMain

noOdds: li $v0, 4 # print string

la $a0, Message4

syscall

j endMain

################################################################################

minEven: add $sp, $sp, -24 # expand stack

sw $ra, 20($sp) # save

sw $s0, 16($sp) # save

sw $s1, 12($sp) # save

sw $s2, 08($sp) # save

sw $s3, 04($sp) # save

sw $s4, 00($sp) # save

minEvenInit: li $s0, 0 # i

li $s1, 0 # address(A[i])

li $s2, 0 # A[i]

li $s3, 0 # quotient

li $s4, 0 # remainder

li $v0, +100000 # result

minEvenLoop: beq $s0, $a1, endMinEven # exit if i == n

add $s1, $s0, $s0 # 2 \* i

add $s1, $s1, $s1 # 4 \* i

add $s1, $s1, $a0 # address(A[i])

lw $s2, 0($s1) # A[i]

div $s3, $s2, 2 # A[i] % 2

mfhi $s4

beq $s4, 1, minEvenLoopCon # next loop if odd

blt $v0, $s2, minEvenLoopCon # next loop if result < A[i]

add $v0, $zero, $s2 # else update result

minEvenLoopCon: add $s0, $s0, 1 # i++

j minEvenLoop

endMinEven: lw $s4, 00($sp) # load

lw $s3, 04($sp) # load

lw $s2, 08($sp) # load

lw $s1, 12($sp) # load

lw $s0, 16($sp) # load

lw $ra, 20($sp) # load

add $sp, $sp, +24 # shrink stack

jr $ra # return

################################################################################

maxOdd: add $sp, $sp, -24 # expand stack

sw $ra, 20($sp) # save

sw $s0, 16($sp) # save

sw $s1, 12($sp) # save

sw $s2, 08($sp) # save

sw $s3, 04($sp) # save

sw $s4, 00($sp) # save

maxOddInit: li $s0, 0 # i

li $s1, 0 # address(A[i])

li $s2, 0 # A[i]

li $s3, 0 # quotient

li $s4, 0 # remainder

li $v0, -100000 # result

maxOddLoop: beq $s0, $a1, endMaxOdd # exit if i == n

add $s1, $s0, $s0 # 2 \* i

add $s1, $s1, $s1 # 4 \* i

add $s1, $s1, $a0 # address(A[i])

lw $s2, 0($s1) # A[i]

div $s3, $s2, 2 # A[i] % 2

mfhi $s4

beq $s4, 0, maxOddLoopCon # next loop if even

bgt $v0, $s2, maxOddLoopCon # next loop if result > A[i]

bgt $s2, $a2, maxOddLoopCon # next loop if A[i] > minEven

add $v0, $zero, $s2 # else update result

maxOddLoopCon: add $s0, $s0, 1 # i++

j maxOddLoop

endMaxOdd: lw $s4, 00($sp) # load

lw $s3, 04($sp) # load

lw $s2, 08($sp) # load

lw $s1, 12($sp) # load

lw $s0, 16($sp) # load

lw $ra, 20($sp) # load

add $sp, $sp, +24 # shrink stack

jr $ra # return

################################################################################

endMain:

**Results:**

* **Graphical user interface, text, table

  Description automatically generated with medium confidence**

**Comments:**

* The program uses `minEven` function to find the minimum even of the array. It will return +100000 if not found. $a0 is the address of the array. $a1 is the length of the array. $v0 is the returned value.
* The program uses `maxOdd` function to find the maximum odd of the array < minEven. It will return -100000 if not found. $a0 is the address of the array. $a1 is the length of the array. $a2 is the minEven of the array. $v0 is the returned value.

## Assignment 3

**Code:**

.data

Str1: .space 51

Str2: .space 51

Msg1: .asciiz "Enter s1 (Max 50): "

Msg2: .asciiz "Enter s2 (Max 50): "

Msg3: .asciiz "Same!\n"

Msg4: .asciiz "Different!\n"

.text

init: li $s0, 0 # i = 0

li $s1, 50 # n = 50

la $s2, Str1 # address(Str1[0])

la $s3, Str2 # address(Str2[0])

la $s4, Str1 # address(Str1[i])

la $s5, Str2 # address(Str2[i])

li $s6, -1 # Str1[i]

li $s7, -1 # Str2[i]

main: li $v0, 4 # print string

la $a0, Msg1

syscall

li $v0, 8

la $a0, Str1 # input string

li $a1, 51

syscall

li $v0, 4 # print string

la $a0, Msg2

syscall

li $v0, 8

la $a0, Str2 # input string

li $a1, 51

syscall

loop: beq $s0, $s1, same # exit if i == n

add $s4, $s2, $s0 # address(Str1[i])

add $s5, $s3, $s0 # address(Str2[i])

lb $s6, ($s4) # Str1[i]

lb $s7, ($s5) # Str2[i]

add $t0, $s6, $s7 # reached both ends if $t0 = 0

beq $t0, 00, same # "\0", "\0"

add $t0, $s6, $s7 # reached both ends if $t0 = 10

beq $t0, 10, same # "\0", "\n" OR "\n", "\0"

add $t0, $s6, $s7 # reached both ends if $t0 = 20

beq $t0, 20, same # "\n", "\n"

bge $s6, 97, lower1 # lower(Str1[i])

afterLower1: bge $s7, 97, lower2 # lower(Str2[i])

afterLower2: bne $s6, $s7, diff # diff found if Str1[i] != Str2[i]

loopCon: add $s0, $s0, 1 # i++

j loop

same: li $v0, 4 # print string

la $a0, Msg3

syscall

j endMain

diff: li $v0, 4 # print string

la $a0, Msg4

syscall

j endMain

lower1: sub $s6, $s6, 32

j afterLower1

lower2: sub $s7, $s7, 32

j afterLower2

endMain:

**Results:**

* **A picture containing text

  Description automatically generated**
* **A picture containing graphical user interface

  Description automatically generated**

**Comments:**

* The program converts uppercase character to lowercase one to do the comparision.
* The program can handle different endings of strings: “\0”, “\n”.