 **Northwestern Polytechnic University**

**EE488 - Computer Architecture**

**Homework Assignment #4**

**Due day: 11/11/2021**

**Instruction:**

1. **Push the answer sheet to GitHub in word file**
2. **Overdue homework submission could not be accepted.**
3. **Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**
4. Implement the following subroutine function in the *utils.asm* file, properly documenting them, and include programs to test them.

**utils.asm:**

.text

PrintNewLine:

li $v0, 4

la $a0, \_\_PNL\_newline

syscall

jr $ra

.data

\_\_PNL\_newline: .asciiz "\n"

# subprogram: PrintInt

# purpose: To print a string to the console

# input: $a0 - The address of the string to print.

# $a1 - The value of the int to print

# returns: None

# side effects: The String is printed followed by the integer value.

.text

PrintInt:

# Print string. The string address is already in $a0

li $v0, 4

syscall

# Print integer. The integer value is in $a1, and must

# be first moved to $a0.

move $a0, $a1

li $v0, 1

syscall

#return

jr $ra

# subprogram: PromptInt

# purpose: To print the user for an integer input, and

# to return that input value to the caller.

# input: $a0 - The address of the string to print.

# returns: $v0 - The value the user entered

# side effects: The String is printed followed by the integer value.

.text

PromptInt:

# Print the prompt, which is already in $a0

li $v0, 4

la $a0, promptint

syscall

# Read the integer value. Note that at the end of the

# syscall the value is already in $v0, so there is no

# need to move it anywhere.

move $a0, $a1

li $v0, 5

syscall

#return

jr $ra

.data

promptint: .asciiz "Enter in an integer: "

.text

PrintString:

addi $v0, $zero, 4

syscall

jr $ra

# subprogram: Exit

# purpose: to use syscall service 10 to exit a program

# input: None

# output: None

# side effects: The program is exited

.text

Exit:

li $v0, 10

syscall

#subprogam : MULT10

#purpose: take an input parameter and return that parameter multiplied by 10 using only shift and add operations.

.text

Mult10:

#promptInt is always called before this so the in will already be in $v0

move $t5, $v0 #moves integer to $t5

sll $t2, $t5, 3 #multiplies input by 2^3 = 8 stores answer in t2

sll $t3, $t5, 1 #multiplies input by 2^1 = 2 stores answer in t3

add $v0,$t2, $t3#adds the two sll ops($t2,$t3) together and stores in v0 return register

jr $ra#return

* 1. *Mult10* - take an input parameter, and return that parameter multiplied by *10* using ONLY shift and add operations.

.data

prompt1: .asciiz "Testing Mult10 subprogram...\n"

result: .asciiz "Result of subprogram call: "

result1: .asciiz "\nValue after multiplying 10: "

.text

main:

#testing mult10

li $v0, 4

la $a0, prompt1

syscall

jal PromptInt

jal Mult10 #returns the value in $v0

move $a1, $v0#moves the returned result of mult10 to a1 argument register for PrintInt

li $v0, 4

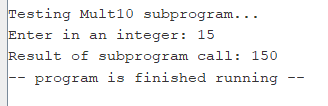
la $a0, result

jal PrintInt#display results with added string

#exit program

jal Exit#always call this to exit

.include "utils.asm"



* 1. *ToUpper* - take a *32* bit input which is *3* characters and a null, or a *3* character string. Convert the *3* characters to upper case if they are lower case, or do nothing if they are already upper case.

.data

inpt: .space 20

newline: .asciiz "\n"

.text

main:

li $v0, 8

li $a1, 20

la $a0, inpt

syscall

li $v0, 4

li $t0, 0

loop:

lb $t1, inpt($t0)

beq $t1, 0, exit

blt $t1, 'a', not\_lower

bgt $t1, 'z', not\_lower

sub $t1, $t1, 32

sb $t1, inpt($t0)

not\_lower:

addi $t0, $t0, 1

j loop

exit:

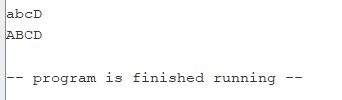
li $v0, 4

la $a0, inpt

syscall

li $v0, 10

syscall



* 1. *ToLower* - take a *32* bit input which is 3 characters and a null, or a *3* character string. Convert the *3* characters to lower case if they are upper case, or do nothing if they are already lower case.

.data

msg: .asciiz "Given string: "

input: .asciiz "'aBc'"

newline: .asciiz "\nLower case String: "

.text

main:

li $v0, 4 #print the msg

la $a0, msg

syscall

li $v0, 4 #print the input string

li $a1, 20

la $a0, input

syscall

li $v0, 4

li $t0, 0

li $v0, 4 #print the newline msg

li $a1, 20

la $a0, newline

syscall

li $v0, 4

li $t0, 0

loop:

lb $t1, input($t0) #check if character is upper case

beq $t1, 0, exit

blt $t1, 'A', case

bgt $t1, 'Z', case

add $t1, $t1, 0x20 #if character is upper case , add 0x20, to convert into lower case

sb $t1, input($t0) #store it

case:

addi $t0, $t0, 1

j loop

exit:

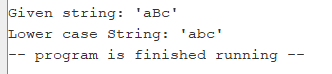
li $v0, 4 #print the lower case string

la $a0, input

syscall

li $v0, 10

syscall



1. Write a program to find prime numbers from *3* to *n* in a loop in MIPS assembly

.data

prompt: .asciiz "\nEnter a number to find prime numbers from 3 to N: "

newLine: .asciiz "\n"

.text

.globl main

main:

addi $s6,$zero,3

li $v0,4

la $a0,prompt

syscall

li $v0,5

syscall

move $s7,$v0

blt $s7,$s6,exit

move $s5,$s6

loop:

move $a0, $s5

jal is\_prime

beq $v0,1,prime

j skip

prime:

li $s4,1

move $a0, $s5

li $v0,1

syscall

li $v0,11

li $a0,' '

syscall

skip:

add $s5,$s5,1

ble $s5,$s7,loop

j exit

is\_prime:

addi $t0, $zero, 2

is\_prime\_test:

slt $t1, $t0, $a0

bne $t1, $zero, is\_prime\_loop

addi $v0, $zero, 1

jr $ra

is\_prime\_loop:

div $a0, $t0

mfhi $t3

slti $t4, $t3, 1

beq $t4, $zero, continue

add $v0, $zero, $zero

jr $ra

continue:

addi $t0, $t0, 1

j is\_prime\_test

primeF:

li $v0,10

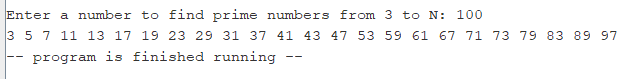
syscall

exit:

bnez $s4,primeF

li $v0,4

syscall



1. Prompt the user for a number from *3…100*, and determine the prime factors for that number. For example, *15* has prime factors *3* and *5*. *60* has prime factors *2, 3*, and *5*. You ONLY have to print out the prime factors.

.data

Integer: .asciiz "Enter integer between 3 to 100 to find prime factors:"

NewLine: .asciiz "\n"

Error: .asciiz " Entered number is either less than 3 or greater than 100"

.text

main:

addi $s6,$zero,3

addi $t8,$zero,100

la $a0, Integer

li $v0, 4

syscall

li $v0, 5

syscall

move $t0,$v0

move $t1, $v0

li $t2, 2

blt $t0,$s6,error

bgt $t0,$t8,error

Loop:

bgt $t2, $t1, EndLoop

div $t0, $t2

mfhi $t3

beqz $t3, function

addi $t2, $t2, 1

j Loop

function:

mflo $t0

move $a0, $t2

li $v0, 1

syscall

la $a0, NewLine

li $v0, 4

syscall

div $t0, $t2

mfhi $t3

beqz $t3, function

j Loop

error:

li $v0,4

la $a0,Error

syscall

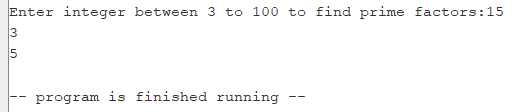
li $v0, 10

syscall

EndLoop:

li $v0, 10

syscall



1. Using only *sll* and *srl*, implement a program to check if a user input value is even or odd. The program should read a user input integer, and print out "The number is even" if the number is even, or "The number is odd", if the number is odd.

.data

prompt:.asciiz "Enter a number to be checked: "

message: .asciiz "\nResult"

msgEven: .asciiz "\n0\nThe Number is even"

msgOdd: .asciiz "\n1\nThe Number is odd"

.text

main:

li $v0,4

la $a0,prompt

syscall

li $v0,5

syscall

move $t0,$v0

srl $s0,$t0,1

sll $t1,$s0,1

#if number is even then original number is equal to the number after SLL and SRL

beq $t0,$t1, resultEven

bne $t0,$t1, resultOdd

#exit

li $v0,10

la $a0,message

syscall

resultEven:

li $v0,4

la $a0,msgEven

syscall

li $v0,10

la $a0,message

syscall

resultOdd:

li $v0,4

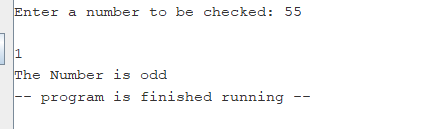
la $a0,msgOdd

syscall

li $v0,10

la $a0,message

syscall



1. Prompt the user for a number *n*, *0 < n < 100*. Print out the smallest number of coins

(quarters, dimes, nickels, and pennies) which will produce *n*. For example, if the user enters *"66",* your program should print out "*2* quarters, *1* dime, *1* nickel, and *1* penny".

.data

quarter: .word 25

dime: .word 10

nickel: .word 5

quarterMsg: .asciiz " quarter(s), "

dimesMsg: .asciiz " dime(s), "

nickelsMsg: .asciiz " nickel(s), "

penniesMsg: .asciiz " pennies \n"

prompt: .asciiz "Enter a number in range 0-100: "

.text

li $v0, 4

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0

lw $t1, quarter

div $t0, $t1

mflo $t2 #no. of quarters

mfhi $t0 #remaining money

lw $t1, dime

div $t0, $t1

mflo $t3 #no. of dimes

mfhi $t0 #remaining money

lw $t1, nickel

div $t0, $t1

mflo $t4 #no. of nickels

mfhi $t0 #remaining money, no. of pennies

li $v0, 1

move $a0, $t2

syscall

li $v0, 4

la $a0, quarterMsg

syscall

li $v0, 1

move $a0, $t3

syscall

li $v0, 4

la $a0, dimesMsg

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 4

la $a0, nickelsMsg

syscall

li $v0, 1

move $a0, $t0

syscall

li $v0, 4

la $a0, penniesMsg

syscall

#exit

li $v0, 10

syscall

