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

#save utils.asm

# File: utils.asm

# Purpose: To define utilities which will be used in MIPS program.

#

# Subprograms Index:

# Exit - Call syscall with a server 10 to exit the program

# PrintString - Print a string to the console

# PromptInt - Prompt the user to enter an integer, and return

.text

# 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.

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

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

# subprogram: PromptInt

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

# Print the prompt, which is already in $v0

.text

PromptInt:

li $v0,4

la $a0,prompt #it will print prompt

syscall

li $v0,5

syscall #ask user input

move $t1,$v0

#save it to t1

jr $ra

# subprogram: PrintString

# purpose: To print a string to the console

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

PrintString:

li $v0,4

la $a0,result #it will print prompt

syscall

move $a0,$s2

li $v0,1

syscall

jr $ra

#subprogam : MULT10

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

Mult10:

sll $s0,$t1,3

sll $s1,$t1,1

add $s2,$s0,$s1

jr $ra#return

# subprogram: Exit

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

# input: None

# output: None

# side effects: The program is exited

Exit:

li $v0, 10

syscall

**#testing mult10 subprogram**

.data

prompt: .asciiz "\nEnter an integer: "

result: .asciiz "\nResult after multiplying 10 is : "

.text

jal PromptInt

jal Mult10

jal PrintString

jal Exit

.include "utils.asm"

Output :

Enter an integer: 6

Result after multiplying 10 is : 60

* 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

testString: .asciiz "**testALL**"

uppercaseString: .space 32

.text

la $t4,uppercaseString

la $t0, testString

Case:

lb $t1, ($t0) #assigns char of string to $t1

beqz $t1, end #ends loop

li $t2, 'a'

bge $t1, $t2, toUpper

toUpper:

ori $t1,$t1,32

xori $t1, $t1, 0x20 #converts lower to uppercase

sb $t1, ($t4) #stores letter in $t4

j continue #iterates both $t0 and $t4

continue: #jumps to next letter in testString and revString

add $t0, $t0, 1

add $t4, $t4, 1

j Case

end: #prints reversed letter case string

la $a0, uppercaseString

li $v0, 4

syscall

Output : TESTALL

* 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

testString: .asciiz "TESTall"

lowercaseString:.space 33

.text

la $t4, lowercaseString

la $t0, testString

Case:

lb $t1, ($t0) #assigns char of string to $t1

beqz $t1, end

li $t3, 'A'

bge $t1, $t3, toLower

toLower:

ori $t1, $t1, 32 #converts upper to lowercase

sb $t1, ($t4) #stores letter in $t4

j continue

continue: #jumps to next letter in testString

add $t0, $t0, 1

add $t4, $t4, 1

j Case

end:

la $a0, lowercaseString

li $v0, 4

syscall

Output : testall

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

.data

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

newLine: .asciiz "\n"

noPrime : .asciiz "\nNo prime number bet'n num 3 and N !"

.text

.globl main

main:

addi $s6,$zero,3

li $v0,4

la $a0,prompt #it will print prompt1

syscall

li $v0,5

syscall #ask user input

move $s7,$v0

blt $s7,$s6,exit #if t2 less than or equal 0 exit

move $s5,$s6

loop:

move $a0, $s5

jal is\_prime # Send the number to the procedure!

beq $v0,1,prime

j skip

prime:

li $s4,1

move $a0, $s5 #move s1 to a0 for printing

li $v0,1

syscall

li $v0,11

li $a0,' ' #it will print new line

syscall

skip:

add $s5,$s5,1 #add m and check condition

ble $s5,$s7,loop

j exit

is\_prime:

addi $t0, $zero, 2 # int x = 2

is\_prime\_test:

slt $t1, $t0, $a0 # if (x > num)

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

addi $v0, $zero, 1 # It's prime!

jr $ra # return 1

is\_prime\_loop: # else

div $a0, $t0

mfhi $t3 # c = (num % x)

slti $t4, $t3, 1

beq $t4, $zero, continue # if (c == 0)

add $v0, $zero, $zero # not a prime

jr $ra

continue:

addi $t0, $t0, 1

j is\_prime\_test # continue the loop

primeF:

li $v0,10

syscall

exit:

bnez $s4,primeF

li $v0,4

la $a0,noPrime

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 its prime factors = "

NewLine: .asciiz "\n"

wrong: .asciiz " No. entered 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 #if t2 less than or equal 0 exit

bgt $t0,$t8,error

Loop:

bgt $t2, $t1, EndLoop

div $t0, $t2

mfhi $t3

beqz $t3, Helper

addi $t2, $t2, 1

j Loop

Helper:

mflo $t0

move $a0, $t2

li $v0, 1

syscall

la $a0, NewLine

li $v0, 4

syscall

div $t0, $t2

mfhi $t3

beqz $t3, Helper

j Loop

error: #When 100<n<3 , will throw error message

li $v0,4

la $a0,wrong #it will print prompt

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 "\nEnter a number to determine whether EVEN or ODD = "

result: .asciiz "\nResult"

resulte: .asciiz "\nThe number is even"

resulto: .asciiz "\nThe number is odd"

.text

main:

#prompt for input

li $v0,4

la $a0,prompt

syscall

li $v0,5

syscall

move $t0,$v0#move number from v0 to t0

srl $s0,$t0,1#Shift right by 1 bit

sll $t1,$s0,1#shift left by 1 bit

beq $t0,$t1, resultE#if true then jump to result Even

bne $t0,$t1, resultO#if true then jump to result Odd

#END

li $v0,10

la $a0,result

syscall

resultO:#jump here when number is odd

li $v0,4

la $a0,resulto

syscall

li $v0,10

la $a0,result

syscall

resultE:#Jump here when number is even

li $v0,4

la $a0,resulte

syscall

li $v0,10

la $a0,result

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

n: .word 5

q: .word 25

d: .word 10

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

nprompt: .asciiz " nickel, "

qprompt: .asciiz " quarters ,"

dprompt: .asciiz " dime , "

pprompt: .asciiz " penny \n"

wrong: .asciiz " No. entered is either less than 3 or greater than 100"

.text

addi $s6,$zero,3

addi $t8,$zero,100

li $v0, 4

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0

blt $t0,$s6,error #if t2 less than or equal 0 exit

bgt $t0,$t8,error

.globl main

main:

lw $t1, q

div $t0, $t1

mflo $t2 #quarters

mfhi $t0 #rest money

lw $t1, d

div $t0, $t1

mflo $t3 #dimes

mfhi $t0 #change

lw $t1, n

div $t0, $t1

mflo $t4 #nickels

mfhi $t0 #rest money

#display the results

li $v0, 1

move $a0, $t2

syscall

li $v0, 4

la $a0, qprompt

syscall

li $v0, 1

move $a0, $t3

syscall

li $v0, 4

la $a0, dprompt

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 4

la $a0, nprompt

syscall

li $v0, 1

move $a0, $t0

syscall

li $v0, 4

la $a0, pprompt

syscall

#kill

li $v0, 10

syscall

error: # when 100<n<3, print error

li $v0,4

la $a0,wrong

syscall

li $v0, 10

syscall