**Logo

Description automatically generated San Francisco Bay University**

**EE488 - Computer Architecture**

**Homework Assignment #4**

**Due day: 3/18/2023**

**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.

.data

input: .asciiz "Enter value to be multiplied:" # Store the prompt and output messages

output: .asciiz "Result after multiplication:"

.text

Mult10:

la $a0,input # Load address of input to $a0

li $v0,4

syscall # Print input message

li $v0,5 # Read integer

syscall

move $t0,$v0 # Move read value to $t0

move $t1,$t0 # Copy $t0 to $t1

sll $t0,$t0,3 # Left shift $t0 by 3 bits, so $t0=$t0\*8

add $t0,$t0,$t1 # Add $t1 which was copied input, with $t0, so $t0 = $t0\*9

add $t0,$t0,$t1 # Add $t1 which was copied input, with $t0, so $t0 = $t0\*10

la $a0,output # Print output message

li $v0,4

syscall

move $a0,$t0 # Move result to $a0

li $v0,1

syscall # Print result

li $v0,10 # Terminate program

syscall

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

.data

input: .asciiz "Enter three character string:" # Store the prompt and output messages

output: .asciiz "String in lowercase:"

str: .space 4 # Reserve space for input string

.text

ToLower:

la $a0,input # Load address of input to $a0

li $v0,4

syscall # Print input message

la $a0,str # Read string to str

li $a1,4

li $v0,8

syscall

la $t0,str # Load address of str to $t0

li $t2,0 # Initialize $t2 to 0

loop:

lb $t1,0($t0) # Load first character to $t1

bge $t1,65, upper1 # If it is greater than or equal to 65 then branch to upper1

b next # If not then branch to next

upper1:

ble $t1,90,upper2 # If it is less than or equal to 90 also, then it is upper case so branch to upper2

b next # If not then branch to next

upper2:

addi $t1,$t1,32 # Convert to lower case

sb $t1,0($t0) # Replace upper case

next:

addi $t2,$t2,1 # Increment $t2 by 1

addi $t0,$t0,1 # Increment $t0 by 1

bne $t2,3,loop # If $t2 is 3 then all characters are checked if not then check again

la $a0,output # Print output message

li $v0,4

syscall

la $a0,str # Load base address of str

li $v0,4 # Print the updated string

syscall

li $v0,10 # Terminate program

syscall

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

.data

input: .asciiz "Enter three character string:" # Store the prompt and output messages

output: .asciiz "String in uppercase:"

str: .space 4 # Reserve space for input string

.text

ToUpper:

la $a0,input # Load address of input to $a0

li $v0,4

syscall # Print input message

la $a0,str # Read string to str

li $a1,4

li $v0,8

syscall

la $t0,str # Load address of str to $t0

li $t2,0 # Initialize $t2 to 0

loop:

lb $t1,0($t0) # Load first character to $t1

bge $t1,97, lower1 # If it is greater than or equal to 97 then branch to lower1

b next # If not then branch to next

lower1:

ble $t1,122,lower2 # If it is less than or equal to 122 also, then it is lower case so branch to lower2

b next # If not then branch to next

lower2:

addi $t1,$t1,-32 # Convert to upper case

sb $t1,0($t0) # Replace lower case

next:

addi $t2,$t2,1 # Increment $t2 by 1

addi $t0,$t0,1 # Increment $t0 by 1

bne $t2,3,loop # If $t2 is 3 then all characters are checked if not then check again

la $a0,output # Print output message

li $v0,4

syscall

la $a0,str # Load base address of str

li $v0,4 # Print the updated string

syscall

li $v0,10 # Terminate program

syscall

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

.data

n: .word 99 # 99 can be changed to any desired value of n

.text

.globl main

main:

li $t0, 3 # Initialize $t0 to 3, the first prime number

lw $t1, n # Load the value of n from memory

loop:

bgt $t0, $t1, done # Exit loop if $t0 > n

li $t2, 2 # Initialize $t2 to 2, the first divisor to check

test\_divisor:

ble $t2, $t0, next # Exit divisor loop if $t2 > $t0

div $t0, $t2 # Divide $t0 by $t2

mfhi $t3 # Get the remainder of the division

beq $t3, $0, next # If remainder is 0, $t0 is not prime, move to next number

addi $t2, $t2, 1 # Increment divisor by 1

j test\_divisor # Repeat divisor loop

next:

addi $t0, $t0, 2 # Increment $t0 by 2 to check next odd number

j loop # Repeat loop

done:

li $v0, 10 # Exit program

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 section

.data

prompt: .asciiz "Enter any number between 3 and 100 to print Prime factors: "

format: .asciiz "%d"

result: .asciiz "All Prime Factors of %d are: \n"

separator: .asciiz ", "

# Text section

.text

.globl main

main:

# Function prologue

li $v0, 9 # allocate space for 9 words on the stack

li $a0, 36 # reserve 36 bytes for the stack

syscall

addi $sp, $sp, -8 # reserve space for $ra and $s0 on the stack

sw $ra, 0($sp) # save the return address

sw $s0, 4($sp) # save $s0 (base address of the stack frame)

# Display prompt

la $a0, prompt

li $v0, 4

syscall

# Read input

li $v0, 5

syscall

move $t0, $v0 # save the input value in $t0

# Check input

li $t1, 2

ble $t0, $t1, main # if input <= 2, go back to the prompt

li $t1, 100

bgt $t0, $t1, main # if input > 100, go back to the prompt

# Display result header

la $a0, result

li $v0, 4

syscall

move $a0, $t0

# Initialize variables

li $s0, 2 # initialize divisor to 2

loop:

# Divide input by divisor

div $t0, $s0

mfhi $t1 # remainder is in $t1

# Check for prime factor

bne $t1, $zero, else # if remainder is not zero, go to else

li $t2, 1 # set flag for prime factor

li $t3, 2 # initialize factor to 2

inner\_loop:

# Divide divisor by factor

div $s0, $t3

mfhi $t4 # remainder is in $t4

# Check for factor

bne $t4, $zero, increment # if remainder is not zero, go to increment

li $t2, 0 # clear flag for prime factor

j exit\_inner\_loop

increment:

addi $t3, $t3, 1 # increment factor

blt $t3, $s0, inner\_loop

exit\_inner\_loop:

# Print prime factor

beq $t2, $zero, else

li $v0, 1

move $a0, $s0

syscall

la $a0, separator

li $v0, 4

syscall

else:

addi $s0, $s0, 1 # increment divisor

blt $s0, $t0, loop # if divisor < input, go to loop

# Print newline

li $v0, 11

li $a0, '\n'

syscall

# Function epilogue

lw $ra, 0($sp)

lw $s0, 4($sp)

addi $sp, $sp, 8

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 an integer: "

even\_msg: .asciiz "The number is even\n"

odd\_msg: .asciiz "The number is odd\n"

.text

.globl main

main:

# Print prompt

li $v0, 4

la $a0, prompt

syscall

# Read integer input

li $v0, 5

syscall

move $t0, $v0 # Store input in $t0

# Check if number is even or odd

srl $t1, $t0, 1 # Shift right by 1 bit to get the least significant bit

sll $t1, $t1, 1 # Shift left by 1 bit to restore the original value

bne $t0, $t1, odd # If the original value is not equal to the shifted value, jump to odd label

# Print even message

li $v0, 4

la $a0, even\_msg

syscall

j end # Jump to end label

odd:

# Print odd message

li $v0, 4

la $a0, odd\_msg

syscall

end:

# Exit program

li $v0, 10

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

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

quartersMsg: .asciiz " quarters, "

dimesMsg: .asciiz " dimes, "

nickelsMsg: .asciiz " nickels, and "

penniesMsg: .asciiz " pennies\n"

quarters: .word 25

dimes: .word 10

nickels: .word 5

.text

.globl main

main:

# Prompt user for input

li $v0, 4

la $a0, prompt

syscall

# Read user input and store in $t0

li $v0, 5

syscall

move $t0, $v0

# Initialize count variables

li $t1, 0 # Quarters count

li $t2, 0 # Dimes count

li $t3, 0 # Nickels count

li $t4, 0 # Pennies count

# Calculate number of quarters

lw $t5, quarters

div $t0, $t5

mflo $t1

mfhi $t0

# Calculate number of dimes

lw $t5, dimes

div $t0, $t5

mflo $t2

mfhi $t0

# Calculate number of nickels

lw $t5, nickels

div $t0, $t5

mflo $t3

mfhi $t4

# The remaining value in $t0 is the number of pennies

move $t4, $t0

# Print results

li $v0, 1

move $a0, $t1

syscall

li $v0, 4

la $a0, quartersMsg

syscall

li $v0, 1

move $a0, $t2

syscall

li $v0, 4

la $a0, dimesMsg

syscall

li $v0, 1

move $a0, $t3

syscall

li $v0, 4

la $a0, nickelsMsg

syscall

li $v0, 1

move $a0, $t4

syscall

li $v0, 4

la $a0, penniesMsg

syscall

# Exit program

li $v0, 10

syscall