**Logo

Description automatically generated San Francisco Bay University**

**EE488 - Computer Architecture**

**Homework Assignment #4**

**Due day: 7/22/2024**

**Instructions:**

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.

***Answer: Check the source file / go to “GitHub” Also given below:***

Utils.asm

.data

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

.text

.globl PrintNewLine

PrintNewLine:

li $v0, 4

la $a0, \_\_PNL\_newline

syscall

jr $ra

.globl PrintInt

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

.globl PromptInt

PromptInt:

# Print the prompt, which is already in $a0

li $v0, 4

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.

li $v0, 5

syscall

# Return

jr $ra

.globl PrintString

PrintString:

li $v0, 4

syscall

jr $ra

.globl Exit

Exit:

li $v0, 10

syscall

.globl Mult10

Mult10:

sll $t1, $a0, 3 # $t1 = $a0 \* 8 (shift left by 3)

sll $t2, $a0, 1 # $t2 = $a0 \* 2 (shift left by 1)

add $v0, $t1, $t2 # $v0 = $t1 + $t2 -> $v0 = $a0 \* 8 + $a0 \* 2 = $a0 \* 10

jr $ra # Return to the caller

main.asm

.data

prompt: .asciiz "Enter an integer value: "

result\_msg: .asciiz "The result of multiplying by 10 is: "

.text

.globl main

main:

# Prompt for integer input

la $a0, prompt # Load address of prompt message

jal PromptInt # Call PromptInt subroutine

move $a1, $v0 # Move input value to $a1

# Pass the input to $a0 for the subroutine

move $a0, $a1

# Call Mult10 subroutine

jal Mult10 # Jump and link to Mult10

# Print the result message

la $a0, result\_msg # Load address of result message

move $a1, $v0 # Move result value to $a1

jal PrintInt # Call PrintInt subroutine

# Exit the program

jal Exit # Call Exit subroutine

* 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.

***Answer: Check the source file / go to “GitHub” Also given below:***

Utils.asm

.data

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

.text

.globl PrintNewLine

PrintNewLine:

li $v0, 4

la $a0, \_\_PNL\_newline

syscall

jr $ra

.globl PrintInt

PrintInt:

li $v0, 4

syscall

move $a0, $a1

li $v0, 1

syscall

jr $ra

.globl PromptInt

PromptInt:

li $v0, 4

syscall

li $v0, 5

syscall

jr $ra

.globl PrintString

PrintString:

li $v0, 4

syscall

jr $ra

.globl Exit

Exit:

li $v0, 10

syscall

.globl Mult10

Mult10:

sll $t1, $a0, 3

sll $t2, $a0, 1

add $v0, $t1, $t2

jr $ra

.globl ToUpper

ToUpper:

# Load the string into registers $t0, $t1, $t2

lbu $t0, 0($a0)

lbu $t1, 1($a0)

lbu $t2, 2($a0)

# Convert $t0 to uppercase if it's lowercase

blt $t0, 'a', check\_t1

bgt $t0, 'z', check\_t1

addi $t0, $t0, -32

check\_t1:

# Convert $t1 to uppercase if it's lowercase

blt $t1, 'a', check\_t2

bgt $t1, 'z', check\_t2

addi $t1, $t1, -32

check\_t2:

# Convert $t2 to uppercase if it's lowercase

blt $t2, 'a', end\_to\_upper

bgt $t2, 'z', end\_to\_upper

addi $t2, $t2, -32

end\_to\_upper:

# Store the converted characters back into memory

sb $t0, 0($a0)

sb $t1, 1($a0)

sb $t2, 2($a0)

jr $ra

main.asm

.data

prompt: .asciiz "Enter a 3-character string: "

result\_msg: .asciiz "\nThe converted string is: "

input\_str: .space 4 # Reserve 4 bytes for the input string

.text

.globl main

main:

# Print prompt message

li $v0, 4

la $a0, prompt

syscall

# Read string input

li $v0, 8

la $a0, input\_str

li $a1, 4

syscall

# Call ToUpper subroutine

la $a0, input\_str

jal ToUpper

# Print the result message

li $v0, 4

la $a0, result\_msg

syscall

# Print the converted string

li $v0, 4

la $a0, input\_str

syscall

# Exit the program

jal Exit

* 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.

***Answer: Check the source file / go to “GitHub” Also given below:***

Utils.asm

.data

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

.text

.globl PrintNewLine

PrintNewLine:

li $v0, 4

la $a0, \_\_PNL\_newline

syscall

jr $ra

.globl PrintInt

PrintInt:

li $v0, 4

syscall

move $a0, $a1

li $v0, 1

syscall

jr $ra

.globl PromptInt

PromptInt:

li $v0, 4

syscall

li $v0, 5

syscall

jr $ra

.globl PrintString

PrintString:

li $v0, 4

syscall

jr $ra

.globl Exit

Exit:

li $v0, 10

syscall

.globl Mult10

Mult10:

sll $t1, $a0, 3

sll $t2, $a0, 1

add $v0, $t1, $t2

jr $ra

.globl ToUpper

ToUpper:

# Load the string into registers $t0, $t1, $t2

lbu $t0, 0($a0)

lbu $t1, 1($a0)

lbu $t2, 2($a0)

# Convert $t0 to uppercase if it's lowercase

blt $t0, 'a', check\_t1

bgt $t0, 'z', check\_t1

addi $t0, $t0, -32

check\_t1:

# Convert $t1 to uppercase if it's lowercase

blt $t1, 'a', check\_t2

bgt $t1, 'z', check\_t2

addi $t1, $t1, -32

check\_t2:

# Convert $t2 to uppercase if it's lowercase

blt $t2, 'a', end\_to\_upper

bgt $t2, 'z', end\_to\_upper

addi $t2, $t2, -32

end\_to\_upper:

# Store the converted characters back into memory

sb $t0, 0($a0)

sb $t1, 1($a0)

sb $t2, 2($a0)

jr $ra

.globl ToLower

ToLower:

# Load the string into registers $t0, $t1, $t2

lbu $t0, 0($a0)

lbu $t1, 1($a0)

lbu $t2, 2($a0)

# Convert $t0 to lowercase if it's uppercase

blt $t0, 'A', check\_t1\_lower

bgt $t0, 'Z', check\_t1\_lower

addi $t0, $t0, 32

check\_t1\_lower:

# Convert $t1 to lowercase if it's uppercase

blt $t1, 'A', check\_t2\_lower

bgt $t1, 'Z', check\_t2\_lower

addi $t1, $t1, 32

check\_t2\_lower:

# Convert $t2 to lowercase if it's uppercase

blt $t2, 'A', end\_to\_lower

bgt $t2, 'Z', end\_to\_lower

addi $t2, $t2, 32

end\_to\_lower:

# Store the converted characters back into memory

sb $t0, 0($a0)

sb $t1, 1($a0)

sb $t2, 2($a0)

jr $ra

main.asm

.data

prompt: .asciiz "Enter a 3-character string: "

result\_msg: .asciiz "\nThe converted string is: "

input\_str: .space 4 # Reserve 4 bytes for the input string

.text

.globl main

main:

# Print prompt message

li $v0, 4

la $a0, prompt

syscall

# Read string input

li $v0, 8

la $a0, input\_str

li $a1, 4

syscall

# Call ToLower subroutine

la $a0, input\_str

jal ToLower

# Print the result message

li $v0, 4

la $a0, result\_msg

syscall

# Print the converted string

li $v0, 4

la $a0, input\_str

syscall

# Exit the program

jal Exit

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

***Answer: Check the source file / go to “GitHub” Also given below:***

.data

prompt: .asciiz "Enter the upper limit n: "

prime\_msg: .asciiz ": Prime number\n"

not\_prime\_msg: .asciiz ": Not a prime number\n"

newline: .asciiz "\n"

.text

.globl main

main:

li $v0, 4

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0

li $t1, 3

loop\_start:

bgt $t1, $t0, end\_program

li $t2, 1

li $t3, 2

check\_divisibility:

bge $t3, $t1, check\_prime

div $t1, $t3

mfhi $t4

beq $t4, $zero, not\_prime

addi $t3, $t3, 1

j check\_divisibility

not\_prime:

li $t2, 0

check\_prime:

li $v0, 1

move $a0, $t1

syscall

beq $t2, 1, print\_prime

li $v0, 4

la $a0, not\_prime\_msg

syscall

j next\_iteration

print\_prime:

li $v0, 4

la $a0, prime\_msg

syscall

next\_iteration:

addi $t1, $t1, 1

j loop\_start

end\_program:

li $v0, 10

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.

***Answer: Check the source file / go to “GitHub” Also given below:***

.data

prompt: .asciiz "Enter a number between 3 and 100: "

newline: .asciiz "\n"

msg\_factors: .asciiz "Prime factors are: \n"

comma\_space: .asciiz "\n"

already\_printed: .word 0

.text

.globl main

main:

li $v0, 4

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0

li $t1, 3

bge $t0, $t1, check\_upper\_bound

li $v0, 10

syscall

check\_upper\_bound:

li $t1, 100

ble $t0, $t1, find\_factors

li $v0, 10

syscall

find\_factors:

li $v0, 4

la $a0, msg\_factors

syscall

li $t1, 2

move $t2, $t0

print\_factors:

bgt $t1, $t2, end\_program

div $t2, $t1

mfhi $t3

beq $t3, $zero, check\_printed

addi $t1, $t1, 1

j print\_factors

check\_printed:

lw $t3, already\_printed

beq $t1, $t3, continue\_check

li $v0, 1

move $a0, $t1

syscall

li $v0, 4

la $a0, comma\_space

syscall

sw $t1, already\_printed

continue\_check:

div $t2, $t1

mflo $t2

li $t1, 2

j print\_factors

end\_program:

li $v0, 4

la $a0, newline

syscall

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.

***Answer: Check the source file / go to “GitHub” Also given below:***

.data

prompt: .asciiz "Enter an integer value: "

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

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

.text

.globl main

main:

li $v0, 4

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0

# Check number is even or odd using sll and srl

srl $t1, $t0, 1

sll $t2, $t1, 1

sub $t3, $t0, $t2

# Check result of the subtraction

beq $t3, $zero, print\_even

print\_odd:

li $v0, 4

la $a0, odd\_msg

syscall

j end\_program

print\_even:

li $v0, 4

la $a0, even\_msg

syscall

end\_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".

***Answer: Check the source file / go to “GitHub” Also given below:***

.data

prompt: .asciiz "Enter a number (0 < n < 100): "

result\_msg: .asciiz "The smallest number of coins for the amount is: "

quarters\_msg: .asciiz " quarters, "

dimes\_msg: .asciiz " dime, "

nickels\_msg: .asciiz " nickel, and "

pennies\_msg: .asciiz " penny.\n"

.text

.globl main

main:

# Prompt the user for input

li $v0, 4

la $a0, prompt

syscall

# Read the integer input

li $v0, 5

syscall

move $t0, $v0

# Check if input is within the valid range

li $t1, 0

li $t2, 100

blt $t0, $t1, error

bge $t0, $t2, error

# Calculate the number of quarters

li $t3, 25

divu $t4, $t0, $t3

mflo $t5

mfhi $t6

# Calculate the number of dimes

li $t3, 10

divu $t7, $t6, $t3

mflo $t8

mfhi $t9

# Calculate the number of nickels

li $t3, 5

divu $t0, $t9, $t3

mflo $t1

mfhi $t2

# Remaining amount is the number of pennies

move $t3, $t2

# Print the results

li $v0, 4

la $a0, result\_msg

syscall

li $v0, 1

move $a0, $t5

syscall

li $v0, 4

la $a0, quarters\_msg

syscall

li $v0, 1

move $a0, $t8 # print number of dimes

syscall

li $v0, 4

la $a0, dimes\_msg # load address of dimes message

syscall

li $v0, 1

move $a0, $t1 # print number of nickels

syscall

li $v0, 4

la $a0, nickels\_msg # load address of nickels message

syscall

li $v0, 1

move $a0, $t3 # print number of pennies

syscall

li $v0, 4

la $a0, pennies\_msg

syscall

li $v0, 10

syscall

error:

li $v0, 10

syscall