**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.
   2. *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.
   3. *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.
5. Write a program to find prime numbers from *3* to *n* in a loop in MIPS assembly.
6. 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.
7. 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.
8. 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".

**QUESTION 1**



**.data**

**prompt: .asciiz "Enter an integer: "**

**.text**

**.globl main**

**# Main function to test MultBy10**

**main:**

**# Print the prompt**

**li $v0, 4 # syscall code for print string**

**la $a0, prompt**

**syscall**

**# Read an integer from the user**

**li $v0, 5 # syscall code for read integer**

**syscall**

**move $a0, $v0 # move the input integer to $a0**

**# Call the MultBy10 function**

**jal MultBy10**

**# The result will be in $v0**

**# We will now print $v0 to confirm the result**

**move $a0, $v0**

**li $v0, 1 # syscall code for print integer**

**syscall**

**# Exit the program**

**li $v0, 10 # syscall code for exit**

**syscall**

**# Function to multiply input by 10**

**MultBy10:**

**# Shift left by 3 (multiply by 8)**

**sll $t0, $a0, 3**

**# Shift left by 1 (multiply by 2)**

**sll $t1, $a0, 1**

**# Add the two results to get multiply by 10**

**add $v0, $t0, $t1**

**# Return from function**

**jr $ra**



**.data**

**prompt: .asciiz "Enter a string: "**

**buffer: .space 256 # Buffer to store the input string**

**.text**

**.globl main**

**main:**

**# Print the prompt**

**li $v0, 4 # syscall code for print string**

**la $a0, prompt**

**syscall**

**# Read the input string from the user**

**li $v0, 8 # syscall code for read string**

**la $a0, buffer**

**li $a1, 256 # Maximum number of characters to read**

**syscall**

**# Call the ToUpper function**

**la $a0, buffer # Load the address of the input buffer into $a0**

**jal ToUpper # Call the ToUpper function**

**# Print the converted string**

**li $v0, 4 # syscall code for print string**

**la $a0, buffer**

**syscall**

**# Exit the program**

**li $v0, 10 # syscall code for exit**

**syscall**

**.text**

**.globl ToUpper**

**ToUpper:**

**# Loop through each character in the string**

**loop:**

**lb $t0, 0($a0) # Load the current character**

**beqz $t0, end # If the character is null, end the loop**

**# Setup ASCII values and bitmask**

**li $t1, 0x61 # ASCII value for 'a'**

**li $t2, 0x7A # ASCII value for 'z'**

**li $t3, 0x20 # Bitmask to clear 5th bit (lowercase to uppercase)**

**# Check if the character is lowercase**

**blt $t0, $t1, next\_char**

**bgt $t0, $t2, next\_char**

**# Convert to uppercase**

**xor $t0, $t0, $t3**

**# Store the converted character back to memory**

**sb $t0, 0($a0)**

**next\_char:**

**addi $a0, $a0, 1 # Move to the next character**

**j loop**

**end:**

**jr $ra # Return from the function**



**.data**

**prompt: .asciiz "Enter a string: "**

**buffer: .space 256 # Buffer to store the input string**

**.text**

**.globl main**

**main:**

**# Print the prompt**

**li $v0, 4 # syscall code for print string**

**la $a0, prompt**

**syscall**

**# Read the input string from the user**

**li $v0, 8 # syscall code for read string**

**la $a0, buffer**

**li $a1, 256 # Maximum number of characters to read**

**syscall**

**# Call the ToLower function**

**la $a0, buffer # Load the address of the input buffer into $a0**

**jal ToLower # Call the ToLower function**

**# Print the converted string**

**li $v0, 4 # syscall code for print string**

**la $a0, buffer**

**syscall**

**# Exit the program**

**li $v0, 10 # syscall code for exit**

**syscall**

**.text**

**.globl ToLower**

**ToLower:**

**# Loop through each character in the string**

**loop:**

**lb $t0, 0($a0) # Load the current character**

**beqz $t0, end # If the character is null, end the loop**

**# Setup ASCII values and bitmask**

**li $t1, 0x41 # ASCII value for 'A'**

**li $t2, 0x5A # ASCII value for 'Z'**

**li $t3, 0x20 # Bitmask to set 5th bit (uppercase to lowercase)**

**# Check if the character is uppercase**

**blt $t0, $t1, next\_char**

**bgt $t0, $t2, next\_char**

**# Convert to lowercase**

**xor $t0, $t0, $t3**

**# Store the converted character back to memory**

**sb $t0, 0($a0)**

**next\_char:**

**addi $a0, $a0, 1 # Move to the next character**

**j loop**

**end:**

**jr $ra # Return from the function**

**QUESTION 2**

**.data**

**prompt: .asciiz "Enter an integer: "**

**msg: .asciiz "Prime: "**

**.text**

**.globl main**

**main:**

**# Prompt for user input**

**li $v0, 4**

**la $a0, prompt**

**syscall**

**# Read integer input**

**li $v0, 5**

**syscall**

**move $a0, $v0**

**# Call FindPrimes**

**jal FindPrimes**

**# Exit**

**li $v0, 10**

**syscall**

**FindPrimes:**

**li $t0, 3 # Start from 3**

**move $t1, $a0 # Store n in $t1**

**Loop:**

**bgt $t0, $t1, Exit # If $t0 > $t1, exit loop**

**CheckPrime:**

**li $t2, 2 # Start dividing from 2**

**li $t4, 1 # Flag for prime check, assume prime**

**DivLoop:**

**mul $t5, $t2, $t2 # $t5 = $t2 \* $t2**

**bgt $t5, $t0, Prime # If $t2^2 > $t0, number is prime**

**rem $t6, $t0, $t2 # $t6 = $t0 % $t2**

**beq $t6, $zero, NotPrime # If remainder is 0, not prime**

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

**j DivLoop**

**Prime:**

**# Print the prime number**

**li $v0, 4**

**la $a0, msg**

**syscall**

**li $v0, 1**

**move $a0, $t0**

**syscall**

**j Continue**

**NotPrime:**

**# Continue to next number**

**addi $t0, $t0, 1**

**j Loop**

**Continue:**

**addi $t0, $t0, 1**

**j Loop**

**Exit:**

**jr $ra**

**QUESTION 3**

**.data**

**prompt: .asciiz "Enter an integer: "**

**msg: .asciiz " "**

**.text**

**.globl main**

**main:**

**# Prompt for user input**

**li $v0, 4**

**la $a0, prompt**

**syscall**

**# Read integer input**

**li $v0, 5**

**syscall**

**move $a0, $v0**

**# Call PrimeFactors**

**jal PrimeFactors**

**# Exit**

**li $v0, 10**

**syscall**

**PrimeFactors:**

**move $t0, $a0 # Store number in $t0**

**li $t1, 2 # Start with the smallest prime number**

**FactorLoop:**

**beq $t0, 1, Exit # If number is 1, exit**

**rem $t2, $t0, $t1**

**bne $t2, $zero, NextPrime**

**# Print the prime factor**

**li $v0, 4**

**la $a0, msg**

**syscall**

**li $v0, 1**

**move $a0, $t1**

**syscall**

**div $t0, $t0, $t1**

**j FactorLoop**

**NextPrime:**

**addi $t1, $t1, 1**

**j FactorLoop**

**Exit:**

**jr $ra**

**QUESTION 4**

**.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:**

**# Prompt for user input**

**li $v0, 4**

**la $a0, prompt**

**syscall**

**# Read integer input**

**li $v0, 5**

**syscall**

**move $a0, $v0**

**# Call CheckEvenOdd**

**jal CheckEvenOdd**

**# Exit**

**li $v0, 10**

**syscall**

**CheckEvenOdd:**

**# Using srl (shift right logical) to check the least significant bit**

**srl $t0, $a0, 1**

**sll $t0, $t0, 1**

**beq $a0, $t0, Even # If the number is equal to the result, it is even**

**# Print "odd" message**

**li $v0, 4**

**la $a0, odd\_msg**

**syscall**

**jr $ra**

**Even:**

**# Print "even" message**

**li $v0, 4**

**la $a0, even\_msg**

**syscall**

**jr $ra**

**QUESTION 5**

**.data**

**prompt: .asciiz "Enter an amount in cents: "**

**quarter\_msg: .asciiz " quarters "**

**dime\_msg: .asciiz " dimes "**

**nickel\_msg: .asciiz " nickels "**

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

**newline: .asciiz "\n"**

**.text**

**.globl main**

**main:**

**# Prompt for user input**

**li $v0, 4**

**la $a0, prompt**

**syscall**

**# Read integer input**

**li $v0, 5**

**syscall**

**move $a0, $v0**

**# Call SmallestCoins**

**jal SmallestCoins**

**# Exit**

**li $v0, 10**

**syscall**

**SmallestCoins:**

**li $t0, 25 # Value of a quarter**

**div $a0, $t0**

**mflo $t1 # Number of quarters**

**mfhi $a0 # Remaining amount**

**li $t0, 10 # Value of a dime**

**div $a0, $t0**

**mflo $t2 # Number of dimes**

**mfhi $a0 # Remaining amount**

**li $t0, 5 # Value of a nickel**

**div $a0, $t0**

**mflo $t3 # Number of nickels**

**mfhi $a0 # Remaining amount**

**move $t4, $a0 # Number of pennies**

**# Print number of quarters**

**li $v0, 1**

**move $a0, $t1**

**syscall**

**li $v0, 4**

**la $a0, quarter\_msg**

**syscall**

**# Print number of dimes**

**li $v0, 1**

**move $a0, $t2**

**syscall**

**li $v0, 4**

**la $a0, dime\_msg**

**syscall**

**# Print number of nickels**

**li $v0, 1**

**move $a0, $t3**

**syscall**

**li $v0, 4**

**la $a0, nickel\_msg**

**syscall**

**# Print number of pennies**

**li $v0, 1**

**move $a0, $t4**

**syscall**

**li $v0, 4**

**la $a0, penny\_msg**

**syscall**

**# Print newline**

**li $v0, 4**

**la $a0, newline**

**syscall**

**jr $ra**