**07/22/2024**

**Solution 1a:**

.text

.globl main

main:

# Prompt user for input

li $v0, 4

la $a0, prompt

syscall

# Read integer from user

li $v0, 5

syscall

move $a0, $v0

# Call Mult10 function

jal Mult10

# Print result

move $a0, $v0

li $v0, 1

syscall

# Exit program

li $v0, 10

syscall

# Mult10 - Multiply input by 10 using only shift and add operations

# Input: $a0 - number to multiply

# Output: $v0 - result of multiplication

Mult10:

sll $t0, $a0, 3 # Multiply by 8 (shift left by 3)

sll $t1, $a0, 1 # Multiply by 2 (shift left by 1)

add $v0, $t0, $t1 # Add results (8x + 2x = 10x)

jr $ra # Return

.data

prompt: .asciiz "Enter a number to multiply by 10: "

**Solution1b:**

.data

input\_str: .asciiz "abc"

output\_msg: .asciiz "Converted to uppercase: "

.text

.globl main

main:

la $a0, input\_str # Load address of input string

jal ToUpper # Call ToUpper subroutine

li $v0, 4 # Print string syscall

la $a0, output\_msg

syscall

li $v0, 4 # Print converted string

la $a0, input\_str

syscall

li $v0, 10 # Exit syscall

syscall

ToUpper:

li $t0, 0x20 # $t0 = 0x20 (mask to change case)

lb $t1, 0($a0) # load first character

blt $t1, 'a', check2 # if first character < 'a', skip

bgt $t1, 'z', check2 # if first character > 'z', skip

sub $t1, $t1, $t0 # convert to uppercase

check2:

sb $t1, 0($a0) # store first character

lb $t2, 1($a0) # load second character

blt $t2, 'a', check3 # if second character < 'a', skip

bgt $t2, 'z', check3 # if second character > 'z', skip

sub $t2, $t2, $t0 # convert to uppercase

check3:

sb $t2, 1($a0) # store second character

lb $t3, 2($a0) # load third character

blt $t3, 'a', done # if third character < 'a', skip

bgt $t3, 'z', done # if third character > 'z', skip

sub $t3, $t3, $t0 # convert to uppercase

done:

sb $t3, 2($a0) # store third character

jr $ra # return

**Solution 1c:**

.data

input\_str: .asciiz "ABC"

output\_msg: .asciiz "Converted to lowercase: "

.text

.globl main

main:

la $a0, input\_str # Load address of input string

jal ToLower # Call ToLower subroutine

li $v0, 4 # Print string syscall

la $a0, output\_msg

syscall

li $v0, 4 # Print converted string

la $a0, input\_str

syscall

li $v0, 10 # Exit syscall

syscall

ToLower:

li $t0, 0x20 # $t0 = 0x20 (mask to change case)

lb $t1, 0($a0) # load first character

blt $t1, 'A', check2 # if first character < 'A', skip

bgt $t1, 'Z', check2 # if first character > 'Z', skip

add $t1, $t1, $t0 # convert to lowercase

check2:

sb $t1, 0($a0) # store first character

lb $t2, 1($a0) # load second character

blt $t2, 'A', check3 # if second character < 'A', skip

bgt $t2, 'Z', check3 # if second character > 'Z', skip

add $t2, $t2, $t0 # convert to lowercase

check3:

sb $t2, 1($a0) # store second character

lb $t3, 2($a0) # load third character

blt $t3, 'A', done # if third character < 'A', skip

bgt $t3, 'Z', done # if third character > 'Z', skip

add $t3, $t3, $t0 # convert to lowercase

done:

sb $t3, 2($a0) # store third character

jr $ra # return

**Solution 2:**

# Find prime numbers from 3 to n

.data

prompt: .asciiz "Enter n: "

newline: .asciiz "\n"

.text

.globl main

main:

# Prompt for n

li $v0, 4

la $a0, prompt

syscall

# Read n

li $v0, 5

syscall

move $s0, $v0 # $s0 = n

# Loop from 3 to n

li $t0, 3 # Current number to check

loop:

bgt $t0, $s0, end

# Check if $t0 is prime

move $a0, $t0

jal is\_prime

beqz $v0, not\_prime

# Print prime number

move $a0, $t0

li $v0, 1

syscall

# Print newline

li $v0, 4

la $a0, newline

syscall

not\_prime:

addi $t0, $t0, 1

j loop

end:

li $v0, 10

syscall

# Function to check if a number is prime

is\_prime:

li $t1, 2 # Divisor

prime\_loop:

mult $t1, $t1

mflo $t2

bgt $t2, $a0, is\_prime\_true

div $a0, $t1

mfhi $t2

beqz $t2, is\_prime\_false

addi $t1, $t1, 1

j prime\_loop

is\_prime\_true:

li $v0, 1

jr $ra

is\_prime\_false:

li $v0, 0

jr $ra

**Solution 3:**

.data

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

newline: .asciiz "\n"

.text

.globl main

main:

# Prompt user for input

li $v0, 4 # syscall: print string

la $a0, prompt # address of prompt string

syscall

# Read integer from user

li $v0, 5 # syscall: read integer

syscall

move $t0, $v0 # store user input in $t0

# Ensure the input is within the specified range (3 to 100)

li $t1, 3

li $t2, 100

blt $t0, $t1, exit # if input < 3, exit

bgt $t0, $t2, exit # if input > 100, exit

# Find and print prime factors

li $t1, 2 # start checking from 2

factor\_loop:

beq $t0, 1, exit # if $t0 == 1, all factors are found, exit

div $t0, $t1 # divide $t0 by $t1

mfhi $t2 # get remainder

bne $t2, 0, next\_factor # if remainder != 0, try next factor

# Print factor

move $a0, $t1

li $v0, 1 # syscall: print integer

syscall

li $v0, 4 # syscall: print newline

la $a0, newline

syscall

div $t0, $t0, $t1 # divide $t0 by $t1 to continue factorizing

j factor\_loop # repeat factorization loop

next\_factor:

addi $t1, $t1, 1 # try next factor

j factor\_loop # repeat factorization loop

exit:

li $v0, 10 # syscall: exit

syscall

**Solution 4:**

.data

prompt: .asciiz "Enter an integer: "

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

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

.text

.globl main

main:

# Prompt user for input

li $v0, 4 # syscall: print string

la $a0, prompt # address of prompt string

syscall

# Read integer from user

li $v0, 5 # syscall: read integer

syscall

move $t0, $v0 # store user input in $t0

# Check if the number is even or odd

andi $t1, $t0, 1 # $t1 = $t0 & 1 (check last bit)

beq $t1, 0, even # if last bit is 0, it's even

odd:

li $v0, 4 # syscall: print string

la $a0, odd\_msg # address of odd message

syscall

j exit # jump to exit

even:

li $v0, 4 # syscall: print string

la $a0, even\_msg # address of even message

syscall

exit:

li $v0, 10 # syscall: exit

syscall

**Solution 5:**

.data

prompt: .asciiz "Enter an amount in cents (0 < n < 100): "

quarter\_msg: .asciiz " quarters\n"

dime\_msg: .asciiz " dimes\n"

nickel\_msg: .asciiz " nickels\n"

penny\_msg: .asciiz " pennies\n"

newline: .asciiz "\n"

.text

.globl main

main:

# Prompt user for input

li $v0, 4 # syscall: print string

la $a0, prompt # address of prompt string

syscall

# Read integer from user

li $v0, 5 # syscall: read integer

syscall

move $t0, $v0 # store user input in $t0

# Calculate quarters

li $t1, 25 # quarter value

div $t0, $t1 # divide input by quarter value

mflo $t2 # $t2 = number of quarters

mfhi $t0 # $t0 = remaining cents

move $a0, $t2

li $v0, 1 # syscall: print integer

syscall

li $v0, 4 # syscall: print string

la $a0, quarter\_msg

syscall

# Calculate dimes

li $t1, 10 # dime value

div $t0, $t1 # divide remaining cents by dime value

mflo $t2 # $t2 = number of dimes

mfhi $t0 # $t0 = remaining cents

move $a0, $t2

li $v0, 1 # syscall: print integer

syscall

li $v0, 4 # syscall: print string

la $a0, dime\_msg

syscall

# Calculate nickels

li $t1, 5 # nickel value

div $t0, $t1 # divide remaining cents by nickel value

mflo $t2 # $t2 = number of nickels

mfhi $t0 # $t0 = remaining cents

move $a0, $t2

li $v0, 1 # syscall: print integer

syscall

li $v0, 4 # syscall: print string

la $a0, nickel\_msg

syscall

# Calculate pennies

move $a0, $t0 # remaining cents are pennies

li $v0, 1 # syscall: print integer

syscall

li $v0, 4 # syscall: print string

la $a0, penny\_msg

syscall

# Exit cleanly

exit:

li $v0, 10 # syscall: exit

syscall

# Ensure no additional instructions are executed

j exit