SAN FRANCISCO BAY UNIVERSITY

San Francisco Bay University

EE488 - Computer Architecture
Homework Assignment #4

Due day: 3/27/2025

Instructions:

1. Implement the following subroutine function in the *utils.asm* file, properly documenting them, and include programs to test them.

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

    b. *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.

    c. *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.

⇨ The answer:
⇨ Note: On my laptop, I was not able to link.
1a) .globl Mult10

```
# Subroutine: Mult10
# Description: Multiply input integer by 10 using only shifts and addition.
# Arguments:
#   $a0 - integer to multiply
# Returns:
#   $v0 - result of $a0 * 10
Mult10:
    sll $t0, $a0, 3      # $a0 * 8
    sll $t1, $a0, 1      # $a0 * 2
    add $v0, $t0, $t1    # $a0 * 10
    jr $ra
```

1a)

```
.data
prompt: .asciiz "Enter a number to multiply by 10: "
result: .asciiz "Result: "
newline: .asciiz "\n"

.text
.globl main

main:
    # Prompt user
    li $v0, 4
    la $a0, prompt
    syscall

    # Read number
    li $v0, 5
    syscall
    move $a0, $v0

    # Call Mult10 from 1a.asm
```

```
    jal Mult10

    # Print label
    li $v0, 4
    la $a0, result
    syscall

    # Print result
    li $v0, 1
    move $a0, $v0
    syscall

    # Newline
    li $v0, 4
    la $a0, newline
    syscall

    # Exit
    li $v0, 10
    syscall
```

1b) .globl ToUpper

```
# Subroutine: ToUpper
# Description: Converts 3-character string in $a0 to uppercase
# Input:  $a0 = address of the string (e.g., "abc")
# Output: In-place conversion to uppercase if needed

ToUpper:
    li $t0, 0             # index = 0

loop_upper:
```

```
    lb $t1, 0($a0)         # load byte from string
    beqz $t1, done_upper   # if null char, end

    li $t2, 97             # ASCII 'a'
    li $t3, 122            # ASCII 'z'
    blt $t1, $t2, skip     # if char < 'a', skip
    bgt $t1, $t3, skip     # if char > 'z', skip

    # Convert lowercase to uppercase
    li $t4, 32
    sub $t1, $t1, $t4
    sb $t1, 0($a0)         # store back uppercase

skip:
    addi $a0, $a0, 1       # move to next char
    addi $t0, $t0, 1
    bne $t0, 3, loop_upper # repeat for 3 chars

done_upper:
    jr $ra

1b) .data
inputStr:   .asciiz "abC"
newline:    .asciiz "\n"
result:     .asciiz "Result: "

.text
.globl main

main:
    la $a0, inputStr
    jal ToUpper
```

```
    # Print result
    li $v0, 4
    la $a0, result
    syscall

    li $v0, 4
    la $a0, inputStr
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 10
    syscall
```

1c) .globl ToLower

```
# Subroutine: ToLower
# Description: Converts 3-character string in $a0 to lowercase
# Input:  $a0 = address of the string (e.g., "ABC")
# Output: In-place conversion to lowercase if needed

ToLower:
    li $t0, 0             # index = 0

loop_lower:
    lb $t1, 0($a0)        # load byte from string
    beqz $t1, done_lower  # if null char, end

    li $t2, 65            # ASCII 'A'
    li $t3, 90            # ASCII 'Z'
```

```
    blt $t1, $t2, skip2    # if char < 'A', skip
    bgt $t1, $t3, skip2    # if char > 'Z', skip

    # Convert uppercase to lowercase
    li $t4, 32
    add $t1, $t1, $t4
    sb $t1, 0($a0)         # store back lowercase

skip2:
    addi $a0, $a0, 1       # move to next char
    addi $t0, $t0, 1
    bne $t0, 3, loop_lower # repeat for 3 chars

done_lower:
    jr $ra
```

1c)

```
.data
inputStr:   .asciiz "XYZ"
newline:    .asciiz "\n"
result:     .asciiz "Result: "

.text
.globl main

main:
    la $a0, inputStr
    jal ToLower

    # Print label
    li $v0, 4
    la $a0, result
    syscall
```

```
# Print modified string
li $v0, 4
la $a0, inputStr
syscall

li $v0, 4
la $a0, newline
syscall

li $v0, 10
syscall
```

2. Write a program to find prime numbers from *3* to *n* in a loop in MIPS assembly.
   ⇨ Qno2.asm
3. 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.
   ⇨ Qno3.asm

4. 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.
   ⇨ Qno4.asm
5. 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".
   ⇨ Qno5.asm