ES 215: Assignment - 3

Name: Birudugadda Srivibhav
Roll no: 22110050

Question-1: Write a program in assembly language to subtract two 16 bit numbers without using the subtraction instruction. Note: the numbers have to be fetched from the memory.

Memory Locations:
The two 16-bit numbers (num1 and num2) are stored in memory and fetched using lw (load word) instructions. They are stored in 32-bit words.

Number Representation:
Both numbers are treated as 16-bit integers, and subtraction is performed using two's complement arithmetic by negating num2 and adding it to num1.

Result Storage:
The result of the subtraction is stored in a memory location labeled answer.

Assumptions: Taking num1 = 0x1234 (4660) and num2 = 0x00F0 (240) - we can take any random value.

```
.data
num1:   .word 0x1234   # First 16-bit number (4660)
num2:   .word 0x00F0   # Second 16-bit number (240)
result: .word 0
msg:    .asciiz "Result: "

.text
.globl main
```

Code:

```
main:
    lw $t0, num1          # $t0 = num1
    lw $t1, num2          # $t1 = num2

    not $t1, $t1          # $t1 = ~num2
    addi $t1, $t1, 1     # $t1 = ~num2 + 1 (2's complement)

    add $t2, $t0, $t1    # $t2 = num1 - num2

    sw $t2, result       # result = $t2
```

Printing the result:

```
# Printing the result:
    li $v0, 4
    la $a0, msg
    syscall

    # Print the result
    li $v0, 1
    lw $a0, result
    syscall

    # Exit the program (system call to exit)
    li $v0, 10
    syscall
```

Output:

```
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

Result: 4420
```

Question-2: Write an assembly language program to find an average of 15 numbers stored at consecutive locations in memory.

Memory Layout:
The 15 numbers are stored as 32-bit words in consecutive memory locations starting from the label numbers.

Loop Counter:
The loop iterates exactly 15 times, matching the number of elements in the numbers array.

Result Storage:
The computed average is stored in a memory location labeled average, and it is assumed to be a 32-bit word.

Assumptions:
The chosen 15 numbers are: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75
Their average should be 40

```
.data
numbers: .word 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75
average: .word 0
msg:    .asciiz "The average is: "

.text
.globl main
```

Code:

```
main:
    # Initialize variables
    li $t0, 0          # sum = 0
    li $t1, 15         # count = 15 (number of elements)
    li $t2, 0          # i = 0 (loop counter)
    la $t3, numbers    # array_address = address of numbers array

loop:
    beq $t2, $t1, done  # if i == count then break
    lw $t4, 0($t3)      # number = numbers[array_address]
    add $t0, $t0, $t4   # sum += number
    addi $t3, $t3, 4    # array_address += 4 (move to next element)
    addi $t2, $t2, 1    # i += 1
    j loop              # jump to start of loop

done:
    li $t5, 15          # divisor = 15
    div $t0, $t5        # quotient = sum / divisor
    mflo $t6            # average = quotient
    sw $t6, average     # store average in memory
```

Output:

```
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

The average is: 40
```

Question-3: Write an assembly language program to find an LCM of two numbers stored at consecutive locations in memory.

Memory Layout:
The two numbers whose LCM is to be calculated are stored at memory locations labeled number1 and number2, and each number is a 32-bit word.

Result Storage:
The computed Least Common Multiple (LCM) is stored in the memory location labeled lcm_result, and it is assumed to be a 32-bit word.

Formula used:
I used the following formula: LCM = (number1 * number2) / GCD
Where GCD is calculated using the euclidean algorithm.

Assumptions:
The chosen numbers are 12 and 15, so their LCM will be 60.

Code:

```
main:
    lw $t0, number1       # $t0 = number1
    lw $t1, number2       # $t1 = number2

    move $t2, $t0         # $t2 = number1
    move $t3, $t1         # $t3 = number2

gcd_loop:
    beq $t1, $zero, done_gcd  # if $t1 == 0, then GCD is in $t0
    div $t0, $t1              # Divide $t0 by $t1
    mfhi $t4                  # $t4 = remainder
    move $t0, $t1             # Move $t1 to $t0
    move $t1, $t4             # Move remainder to $t1
    j gcd_loop                # Repeat the loop

done_gcd:
    # Calculate LCM
    mul $t5, $t2, $t3        # $t5 = number1 * number2
    div $t5, $t0             # Divide $t5 by GCD
    mflo $t6                 # $t6 = LCM result
    sw $t6, lcm_result      # Store LCM result in memory
```

Output:

```
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

The LCM is: 60
```

Question-4: Write an assembly language program to calculate multiplication of two numbers without using MUL commands.

Memory Layout:
Two numbers are stored in consecutive memory locations labeled num1 and num2.

Result Storage:
The calculated result is stored in a memory location labeled product.

Formula used:
Multiplication is the same as repeated addition, I used the same principle.

Assumptions:
For testing purposes, the numbers chosen are 18 and 21, whose product is 378.

Code:

```
main:
    lw $t0, num1              # Load num1 into $t0
    lw $t1, num2              # Load num2 into $t1
    li $t2, 0                 # Initialize $t2 (result) to 0
    li $t3, 0                 # Initialize $t3 (counter) to 0

mul_loop:
    beq $t1, $t3, end         # If counter == num2, end loop
    add $t2, $t2, $t0         # Add num1 to result
    addi $t3, $t3, 1          # Increment counter
    j mul_loop                    # Jump back to loop

end:
    sw $t2, result            # Store the result in memory
```

Output:

```
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar

378
```

Question-5: Write an assembly language program to find a given number in the list of 10 numbers (assuming the numbers are sorted). If found store 1 in output, else store 2 in output.The given number has been loaded from X location in memory, the output has to be stored at the next location and if found store the number of iterations and the index of the element at the next at the next consecutive locations, if found.

Answer about to write.

Question-6:
Write an assembly language program to find a character in a string.

Method Used:
String and Character: The string to search is stored at the memory location labeled string, and the character to search for is stored at char.

Initialization: Initialize an index counter to 0.

Character Search: Iterate through each character in the string:

1) Load the current character from the string.
2) If the current character matches the target character, jump to the "found" section.
3) If the end of the string (null terminator) is reached, jump to the "not found" section.

Character Found: If the character is found, print the message indicating the index where the character was found.

Character Not Found: If the character is not found after scanning the entire string, print a message indicating the character was not found.

End Program: Exit the program after printing the appropriate message.

Assumptions:
The string is taken as “Computer Architecture” and the character to search for is “A”, which is at a position of 9 (following zero based indexing)

Code:

```
.data
    string:     .asciiz "Computer Architecture"     # String to search in
    char:       .byte 'A'                   # Character to search for
    notfound:   .asciiz "Character not found.\n"
    found:      .asciiz "Character found at index: "
    newline:    .asciiz "\n"

.text
    .globl main

main:
    la $t0, string            # Load address of the string
    lb $t1, char              # Load the character to find
    li $t2, 0                 # Initialize index

search_loop:
    lb $t3, 0($t0)            # Load current character
    beq $t3, $zero, not_found # End of string
    beq $t3, $t1, found_char  # Character found
    addi $t0, $t0, 1          # Next character
    addi $t2, $t2, 1          # Increment index
    j search_loop             # Continue loop

not_found:
    li $v0, 4                 # Print string
    la $a0, notfound          # Load address
    syscall                   # Print message
    j exit_program            # Exit

found_char:
    li $v0, 4                 # Print string
    la $a0, found             # Load address
    syscall                   # Print message

    li $v0, 1                 # Print integer
    move $a0, $t2             # Load index
    syscall                   # Print index

    li $v0, 4                 # Print newline
    la $a0, newline           # Load address
    syscall                   # Print newline

exit_program:
    li $v0, 10                # Exit syscall
    syscall                   # Exit program
```