Computer Architecture and Organisation - Assignment 3

Name : Nimitt

Roll No: 22110169

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.

Solution:

The following is an assembly program to subtract two 16 bit numbers for MIPS architecture. We define two numbers num1 and num2 and store num1 - num2 in result. The subtraction is performed by adding 2's complement of num2 to num1.

$$2\text{'s}\mathrm{Complement}(a) = \mathrm{not}(a) + 1$$

```
.data
    num1:   .half 35          # First 16-bit number
    num2:   .half 5           # Second 16-bit number
    result: .word 0           # To store the result of num1 - n

.text
.globl main

main:

    # Loading the numbers into temporary registers
    lh      $t0, num1         # Load num1 into $t0
    lh      $t1, num2         # Load num2 into $t1
```

```
    # Complementing num2
    not     $t1, $t1           # $t1 = ~num2
    addi    $t1, $t1, 1        # $t1 = ~num2 + 1

    # Addition giving num1-num2
    add     $t2, $t0, $t1      # $t2 = num1 + (two's complement

    # Storing the result back in memory
    sw      $t2, result        # Store the result

        # Exit
    li      $v0, 10            # Load syscall code for exit
    syscall                    # Make system call to exit
```

Question 2

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

Solution:

The following assembly code performs average of nums array containing 15 numbers. We keep a counter i intialised to 0 and bne construct to loop until i == 15 and find the sum of the nums. Then, divide by 15 to find the average.

$$\text{Average} = \frac{\sum arr[i]}{15}$$

```
.data
nums:       .word 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15  # Array c
count:      .word 15   # Number of elements in the array
sum:        .word 0    # Variable to store the sum
average:    .word 0    # Variable to store the average

.text
```

```
.globl main

main:
    # Initialising variables
    li      $t0, 0        # t0 will hold the index (i) in the lo
    la      $t1, nums     # t1 points to the base address of the
    li      $t2, 15       # t2 holds the count of numbers (15)
    li      $t3, 0        # t3 will accumulate the sum

loop:
    # Loading the element from numbers array
    lw      $t4, 0($t1)   # t4 = nums[i]

    # Adding the current number to the sum
    add     $t3, $t3, $t4          # t3 = t3 + t4 (sum = sum + c

    # Incrementing the index and array pointer
    addi    $t0, $t0, 1   # t0 = t0 + 1 (i = i+1)
    addi    $t1, $t1, 4   # t1 = t1 + 4 (t1 points to next elem

    # Loop until all 15 numbers are added
    bne     $t0, $t2, loop # jump to loop if (t0 != t2)

    # storing the sum tn t3
    sw      $t3, sum   # t3 = sum

    # Dividing the sum by 15 to get the average
    li      $t5, 15       # t5 = 15
    div     $t3, $t5      # t3 / 15
    mflo    $t6           # t6 = t3/5

    # Storing the result : average in memory
    sw      $t6, average

    # Exit
```

```
    li      $v0, 10       # System call for exit
    syscall
```

Question 3

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

The following computes LCM of two numbers num1 and num2 stored consecutively in numbers array. To compute LCM we use the following formula:

$$\mathrm{LCM}(a,b) = \frac{\mathrm{GCD}(a,b)}{a \times b}$$

We first compute the product of num1 and num2 and then compute gcd using Euclid's algorithm.

$$\mathrm{GCD}(a,b) = \mathrm{GCD}(b, a \mod b)$$

And use the first formula to compute LCM and store it in lcm.

Solution:

```
.data
    nums:    .word 7,3           # num1 = 7 num2 = 3
    lcm:     .word 0             # Store LCM  (num1,num2)

.text
.globl main

main:
    # Loading the numbers into temporary registers
    lw      $t0, numbers         # Load num1 into $t0
    lw      $t1, numbers+4       # Load num2 into $t1
```

```
        # Calculating num1 * num2
    mul     $t2, $t0, $t1        # $t2 = num1 * num2


    # jumping to gcd function
    jal     gcd


    # Dividing gcd by num1*num2 to get LCM
    div     $t2, $v0             # (num1 * num2) / gcd(num1,num2
    mflo    $t3                  # $t3 = lcm(num1,num2)

    # Store the LCM result in lcm
    sw      $t3, lcm

    # Exit
    li      $v0, 10              # Load system call code for exi
    syscall                      # Make the system call to exit


# GCD function
gcd:
        # loading num1 and num2 to function registers
    move    $a0, $t0             # Move num1 into $a0
    move    $a1, $t1             # Move num2 into $a1

gcd_loop:
    beq     $a1, $zero, gcd_done # If num2 == 0, gcd = num1 (in
    div     $a0, $a1             # num1/num2
    mfhi    $a2                  # a2 = num1/num2 (integer divis
    mul     $a2, $a2, $a1        # a2 = int(num1/num2)*num2
    sub     $a2, $a1, $a2        # a2 = num1 % num2
    move    $a0, $a1             # Update a0 = num2
    move    $a1, $a2             # Update a1 = num1 % num2
    j       gcd_loop             # Repeat until num2 == 0
```

```
gcd_done:
    move    $v0, $a0            # $v0 = gcd(num1,num2)
    jr      $ra                 # return
```

Question 4:

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

The following is an assembly language program to multiply two numbers num1 and num2 by adding num1 to itself num2 times using loop construct.

$$A \times B = \sum_{i=1}^{B} A$$

Solution:

```
.data
    num1:   .word 10        # num1 = 10
    num2:   .word 5         # num2 = 5
    result: .word 0         # result

.text
.globl main

main:
    # Loading the numbs and initialising loop variables
    lw      $t0, num1       # $t0 = num1
    lw      $t1, num2       # $t1 = num2
    li      $t2, 0          # $t2 = 0 (storing current sum)
    li      $t3, 0          # $t3 = 0 (counter)

    # adding num1 to itself num2 times
multiply_loop:
```

```
    beq     $t3, $t1, end_multiply # If ($t3 == $t1), end loop
    add     $t2, $t2, $t0          # $t2 (current sum) = $t2 + r
    addi    $t3, $t3, 1            # $t3 = $t3 + 1
    j       multiply_loop           # Repeat until done

end_multiply:
    # Storing in result
    sw      $t2, result             # Store the result in memory

    # Exit
    li      $v0, 10                 # Load syscall code for exit
    syscall                         # Make system call to exit
```

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.

The following assembly program finds the number (x) and stores the required results. Because the array is sorted we use binary search to find the number.

```
BinarySearch(array, target):

    low ← 0
    high ← length(array) - 1

    while low ≤ high:
        mid ← (low + high) / 2
```

```
        if array[mid] = target:
            Target found

        else if array[mid] < target:
                Search in right half
            low ← mid + 1

        else:
                Search in left half
            high ← mid - 1

    Target not found
```

Solution:

```
.data
    nums:   .word  1,2,3,4,5,6,7,8,9,10  # array of 10 nums
    X:          .word 10   # num to be seached (x)
    output:     .word 0    # 1 if found, 2 if not found
    iters:      .word 0    # number of iterations if found
    index:      .word -1   # index of the found number if found

.text
.globl main

main:
    # Load the num into $t0
    lw      $t0, X              # $t0 = x

    # Initialize search variables
    li      $t1, 0              # $t1 = low = 0
    li      $t2, 9              # $t2 = high = 9 (last index of
    li      $t3, 0              # $t3 = iteration counter
```

```
binary_search:
    # if (low <= high)
    bgt     $t1, $t2, not_found # If low > high, number is not i

    # Calculate mid = (low+high)/2
    add     $t4, $t1, $t2       # $t4 = low + high
    sra     $t4, $t4, 1         # $t4 = mid   (low + high) / 2

    # Load the value at the mid-point index
    sll     $t5, $t4, 2         # $t5 = mid * 4 (byte offset for
    lw      $t6, nums($t5)      # $t6 = nums[mid]

    # Compare nums[mid] with x
    beq     $t6, $t0, found     # If numbers[mid] == x, jmp to f

    # If numbers[mid] < x, search in the upper half
    blt     $t6, $t0, search_upper

    # Else, search in the lower half
    search_lower:
    addi    $t2, $t4, -1        # high = mid - 1
    j       continue_search

    search_upper:
    addi    $t1, $t4, 1         # low = mid + 1

    continue_search:
    addi    $t3, $t3, 1         # increment iteration counter
    j       binary_search       # continue the search

found:
    li      $t7, 1              # Store 1 in $t7 (indicating fou
    sw      $t7, output         # Store 1 in output

    sw      $t3, iterations     # Store the iteration count
    sw      $t4, index          # Store the index where the numb
```

```
    # Exit
    li      $v0, 10             # Load syscall code for exit
    syscall                     # Make system call to exit

not_found:
    li      $t7, 2              # Store 2 in $t7 (indicating not
    sw      $t7, output         # Store 2 in output

    # Exit
    li      $v0, 10             # Load syscall code for exit
    syscall                     # Make system call to exit
```

Question 6:

Write an assembly language program to find a character in a string.

The following code finds the character char in string. We linearly search whole string to check if any character is the char.

Solution:

```
.data
    string:     .asciiz "Random String"      # string to search
    char:       .byte 'm'                    # character to sear
    output:     .word 0                      # store 1 if char i
    index:      .word -1                     # store index if th

.text
.globl main

main:
    # loading the base address of the string into $t0
    la      $t0, string         # $t0 = string address
```

```
    # load the character to search into $t1
    lb      $t1, char          # $t1 = character to search for (

    # initialising loop variables
    li      $t2, 0             # $t2 = index (starting from 0)

find_char:
    lb      $t3, 0($t0)        #  $t3 = next char in string
    beq     $t3, $zero, not_found  # If we hit the null terminat

    beq     $t1, $t3, found    # If the character matches, go to

    # Increment the address and index
    addi    $t0, $t0, 1        # Move to the next character in t
    addi    $t2, $t2, 1        # Increment the index
    j       find_char          # Repeat the search in next part

found:
    li      $t4, 1             # $t4 = 1 if character is found
    sw      $t4, output        # output = 1
    sw      $t2, index         # Store the index of the found ch

    # Exit
    li      $v0, 10            # Load syscall code for exit
    syscall                    # Make system call to exit

not_found:
    li      $t4, 0             # $t4 = 0 as character is not fou
    sw      $t4, output        # output = 0

    # Exit
    li      $v0, 10            # Load syscall code for exit
    syscall                    # Make system call to exit
```