

COE 301/ICS 233, Term 172

Computer Architecture & Assembly Language

HW# 2 Solution

- Q.1.** Carry out resulting from addition of unsigned numbers can be used to check if the result of addition is incorrect. Write the shortest sequence of MIPS instructions to determine if there is a carry out from the addition of two registers \$t1 and \$t2. Place the carry out (0 or 1) in register \$t0.

When adding two numbers and there is a carryout this implies that the result must be smaller than each of the added operands. Thus, we can use the following sequence of instructions to check that we have a carry out:

```
add $t0, $t1, $t2
sltu $t0, $t0, $t1
```

- Q.2.** Write a MIPS assembly program that asks the user to enter an integer, reads the integer and then displays the integer representation in both binary and hexadecimal, assuming 32-bit representation. A sample execution of the program is given below:

```
Enter an integer: -5
Number representation in binary is: 11111111111111111111111111111011
Number representation in hexadecimal is: FFFFFFFB
```

```
.DATA
prompt: .asciiz "\n Enter an integer:"
msg1: .asciiz "\n Number representation in binary is:"
msg2: .asciiz "\n Number representation in hexadecimal is:"
table: .asciiz "0123456789ABCDEF"
.TEXT
.GLOBL    main

main:

    li $t3, 32

# Printing prompt message to read an integer
    li $v0, 4          # system call code for print string
    la $a0, prompt    # loads address of prompt into $a0
    syscall           # print the prompt message

# Reading the integer. Read integer is stored $v0
```

```

    li $v0, 5          # system call code for read integer
    syscall

    move $t0, $v0

# Displaying the entered number in binary

# Printing msg1 to display binary number
    li $v0, 4          # system call code for print string
    la $a0, msg1       # loads address of prompt into $a0
    syscall            # print the prompt message

# Initializing loop counter $t3
    li $t3, 32

loop:
    rol $t0, $t0, 1
    andi $a0,$t0, 1
# Print the integer result in a0
    li $v0, 1          # Load the system call number
    syscall

    sub $t3, $t3, 1
    bne $t3, $zero, loop

# Printing msg2 to display hexadecimal number
    li $v0, 4          # system call code for print string
    la $a0, msg2       # loads address of prompt into $a0
    syscall            # print the prompt message

# Initializing loop counter $t3
    li $t3, 8

loop2:
    rol $t0, $t0, 4
    andi $a0,$t0, 15
# Converting number into in $a0 to hex character
    la $t1, table
    addu $t1, $t1, $a0
    lb $t1, 0($t1)
    move $a0, $t1

# Print the character result in a0
    li $v0, 11         # Load the system call number
    syscall

    sub $t3, $t3, 1
    bne $t3, $zero, loop2

# Return to operating system

```

```

li    $v0, 10    # Load the system call number.
syscall        # Return.

```

- Q.3.** Write a program to implement the procedure, **SelectionSort**, to sort an array of integers (i.e. 32-bit signed numbers) in an **ascending** order.

The pseudo code for the **SelectionSort** procedure is given below:

```

SelectionSort (Array, Size)
  for (position= 0 to Size-2)
    MinValue = Array[position]
    MinPosition = position
    for (j=position+1 to Size-1)
      if (Array[j] < MinValue) then
        MinValue = Array[j]
        MinPosition = j
      end if
    end for
    if (position ≠ MinPosition) then
      Array[MinPosition] = Array[Position]
      Array[Position] = MinValue
    end if
  end for
end SelectionSort

```

Store the array to be sorted in variable Array as defined below.

```

Array: .word 10, 2, 0, 15, 25, 30, 7, 22

```

Your program should display the following:

```

Array before sorting is: 10 2 0 15 25 30 7 22

```

```

Array after sorting is: 0 2 7 10 15 22 25 30

```

```

.DATA

```

```

Array: .word 10, 2, 0, 15, 25, 30, 7, 22

```

```

msg1: .asciiz "\n Array before sorting is:"

```

```

msg2: .asciiz "\n Array after sorting is:"

```

```

.TEXT

```

```

.GLOBL    main

```

```

main:

```

```

la $s0, Array

```

```

# Array address

```

```

li $s1, 8

```

```

# Size of the array

```

```

# Printing Array before sorting

# Printing msg1
    li $v0, 4                # system call code for print string
    la $a0, msg1            # loads address of prompt into $a0
    syscall                 # print the prompt message

# Printing the array
    move $t0, $s1           # loop counter
    li $t1, 0              # array index
Loop:
    addu $t2, $t1, $s0     # address of indexed element
    lw $a0, 0($t2)
# Print the integer in a0
    li $v0, 1              # Load the system call number
    syscall

# Printing a comma
    li $v0, 11
    li $a0, ','
    syscall

    addi $t1, $t1, 4       # increment array index
    addi $t0, $t0, -1
    bgtz $t0, Loop

# Sorting the Array

# for (position= 0 to Size-2)
    addi $s2, $s1, -2     # 1st for loop max=Size-2
    li $t0, 0             # position=0
ForLoop:
    bgt $t0,$s2, EndFor
    sll $t1, $t0, 2       # multiply position by 4
    addu $t1, $t1, $s0    # address of position
    lw $t2, 0($t1)       # Minvalue
    move $t3, $t0        # MinPosition
# for (j=position+1 to Size-1)
    addi $t4, $s1, -1     # 2nd for loop max=Size-1
    addi $t5, $t0, 1     # j
ForLoop2:
    bgt $t5,$t4, EndFor2
    sll $t6, $t5, 2       # multiply j by 4
    addu $t6, $t6, $s0    # address of j element
    lw $t7, 0($t6)
    slt $t8, $t7, $t2    # if (Array[j] < MinValue) then
    beqz $t8, Endif

```

```

        move $t2, $t7
        move $t3, $t5
    Endif:
        addi $t5, $t5, 1
        j ForLoop2
    EndFor2:
        beq $t0, $t3, EndIf2
        sll $t1, $t0, 2
        addu $t1, $t1, $s0
        lw $t9, 0($t1)
        sll $t6, $t3, 2
        addu $t6, $t6, $s0
        sw $t9, 0($t6)
        sw $t2, 0($t1)
    EndIf2:
        addi $t0, $t0, 1
        j ForLoop
    EndFor:

# Printing Array after sorting

# Printing msg2
    li $v0, 4
    la $a0, msg2
    syscall

# Printing the array
    move $t0, $s1
    li $t1, 0
    Loop2:
        addu $t2, $t1, $s0
        lw $a0, 0($t2)
        # Print the integer in a0
        li $v0, 1
        syscall
        # Printing a comma
        li $v0, 11
        li $a0, ''
        syscall

        addi $t1, $t1, 4
        addi $t0, $t0, -1
        bgtz $t0, Loop2

# Return to operating system
    li $v0, 10
    syscall

```

```

# MinValue = Array[j]
# MinPosition = j

# multiply position by 4

# Array[MinPosition] = Array[Position]
# multiply MinPosition by 4

# Array[Position] = MinValue

# system call code for print string
# loads address of prompt into $a0
# print the prompt message

# loop counter
# array index

# address of indexed element

# Load the system call number

# increment array index

# Load the system call number.
# Return.

```