COMP1047 Lab Week 05

1. Write a program in MIPS32 assembly language to read a number. The program should indicate if the number is odd or even.

Solution

```
   .data
prompt:     .asciiz "Please enter an integer: "
rs1_string: .asciiz "The input integer is odd. "
rs2_string: .asciiz "The input integer is even. "

   .text
   .globl main
main:
# Display the prompt
   la $a0, prompt
   li $v0, 4
   syscall

# Read input value and save it to s0
   li $v0, 5
   syscall
   move $s0, $v0

# Check if odd or even
   li $t0, 2
   div $t0, $s0, $t0
   mfhi $s1              # Save remainder in $s1
   beq $s1, $zero, else    # Check if the remainder is 0
   la $a0, rs1_string      # The result is odd, corresponding to a non-zero remai
   li $v0, 4
   syscall
   j exit

else:
   la $a0, rs2_string      # The result is even, corresponding to a zero remainder.
   li $v0, 4
   syscall

# Exit
exit:
   li $v0, 10
   syscall
```

2. Implement the following C functions using MIPS32 procedure.

```
int non_leaf (int g, int h, int i, int j){
     int f;
     f = leaf (g+h, i+j);
     return f;
}

int leaf (int m, int n){
     int f;
     f = m-n;
     return f;
}
```

Solution

```
   .data
i1: .asciiz "Please input g: "
i2: .asciiz "Please input h: "
i3: .asciiz "Please input i: "
i4: .asciiz "Please input j: "
o1: .asciiz "non-leaf of (g,h,i,j) is : "

   .text
   .globl main

leaf: # return f=m-n
   sub $v0, $a0, $a1
   jr $ra

nleaf: # return f=g+h-i-j
   addi $sp, $sp, -4      # allocate
   sw $ra, 0($sp)

   add $a0, $a0, $a1
   add $a1, $a2, $a3

   jal leaf

   # move $t0, $v0        # if further calc
   # ......
   # move $v0, $t0

   lw $ra, 0($sp)        # restore
   addi $sp, $sp, 4

   jr $ra

main:
   la $a0, i1
   li $v0, 4
   syscall
```

```
li $v0, 5             # read g to $s0
syscall
move $s0, $v0

la $a0, i2
li $v0, 4
syscall

li $v0, 5             # read h to $s1
syscall
move $s1, $v0

la $a0, i3
li $v0, 4
syscall

li $v0, 5             # read i to $s2
syscall
move $s2, $v0

la $a0, i4
li $v0, 4
syscall

li $v0, 5             # read j to $s3
syscall
move $s3, $v0

la $a0, o1             # print text o1
li $v0, 4
syscall

move $a0, $s0
move $a1, $s1
move $a2, $s2
move $a3, $s3

jal nleaf

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

li $v0, 10
syscall
```

3. Write an MIPS32 program that reads a string from console and then print out the string in its reverse alphabetical order. For example, if the string from user is "Hello", then you should print out "olleH".

   Hint: To read a string from user, you need to allocate a memory buffer (.data space) of appropriate sizes using `.space` directive. For example, the following statement requests 10-byte space of memory space with the starting address as `buffer`.

```
            .data
buffer:     .space 10
```

   The following segment reads a string from console. At the end of the syscall, the string is stored in buffer in data segment.

```
        la $a0, buffer     #buffer address to $a0
        li $a1, 10  #string length to $a1
        li $v0, 8   # read string
        syscall
```

Solution

```
  .data
prompt1:   .asciiz "Please type in a string no more than 99 characters: "
rs_string:  .asciiz "The reverse order is: "
buffer:     .space 100   # space to store the string, 1 extra byte to store null

   .text
   .globl main
main:
   la $a0, prompt1     # prompt for string
   li $v0, 4
   syscall

   la $a0, buffer      # string address to $a0
   li $a1, 100        # string length to $a1
   li $v0, 8          # read string
   syscall

   la $a0, rs_string   # The result =
   li $v0, 4
   syscall

   #reverse the order of the string
   la $s0, buffer

loop:
   lb $t0, ($s0)
   addi $s0, $s0, 1
   bne $t0, $zero, loop   #continue until end of string is reached
   addi $s0, $s0, -1      #set $s0 point to the end of the string (null)

   la $s1, buffer
   addi $s1, $s1, -1      #set $s1 1 byte lower than buffer
```

```
loop2:
   addi $s0, $s0, -1   # go to previous char
   lb $a0, ($s0)       # load a char
   beq $s0, $s1, stop
   li $v0, 11          # print char
   syscall
   j loop2             #continue until reach the first char

stop:
   li $v0, 10
   syscall
```

4. For the above question, instead of printing out the reverse order of the string, please change the third character of the string to upper case (assuming it was typed in as lower case) and then print out the string. For example, if the input is \Hello", then change it to \HeLlo" before printing out. Note, you can only use `lw` and `sw` instructions for data transfer to/from the main memory.

Solution

```
    .data
prompt1:   .asciiz "Please type in a string no more than 99 characters: "
rs_string:  .asciiz "The updated string is: "
masks:     .word 0x00FF0000 0xFF00FFFF   #masks
buffer:    .space 100   # space to store the string, 1 extra byte to store null

   .text
   .globl main
main:
   la $a0, prompt1     # prompt for string
   li $v0, 4
   syscall

   la $a0, buffer      # string address to $a0
   li $a1, 100         # string length to $a1
   li $v0, 8           # read string
   syscall

   la $a0, rs_string   # The updated string is
   li $v0, 4
   syscall

   #Update the string
   la $t0, buffer
   lw $s0, ($t0)       # 4 letters in s0
   la $t1, masks
   lw $s1, ($t1)       # mask1 in s1
   lw $s2, 4($t1)      # mask2 in s2
   and $s3, $s0, $s1   # mask out bytes 1,2 4.
   srl $s3, $s3, 16    # get the third character of the string
   addi $s3, $s3, -32  # to upper case letters, (lower case assumed)
   sll $s3, $s3, 16    # shift UPPER case letter in 3nd byte
   and $s4, $s0, $s2   # mask out third letter
   or $s4, $s4, $s3    # new string in $s4

   la $a0, buffer
   sw $s4, ($a0)       #store back to memory
   li $v0, 4           #print out updated string
   syscall

   li $v0, 10
   syscall
```

5. Given two integer arrays A and B, in which each integer is represented in 32-bit two's complement format. Assume that A and B are defined as follows.

```
      .data
A:    .word 4  6  12  -8  5
B:    .word 3  2   1   4  0
```

Update B[0]= 2*A[3]+B[4] and then print out all elements in B.

Solution

```
   .data
rs_string:  .asciiz "B[i]= \n"
nline:      .asciiz "\n"
A:         .word 4 6 12 -8 5
B:         .word 3 2 1 4 0

   .text
   .globl main
main:
   la $a0, rs_string   # The result =
   li $v0, 4
   syscall

   la $s0, A
   la $s1, B
   li $s2, 5           # array length of B
   lw $t0, 12($s0)     # $t0 = A[3]
   lw $t1, 16($s1)     # $t1 = B[4]
   sll $t0, $t0, 1     # 2*A[3]
   add $t0, $t0, $t1   # B[0] = $t0

   sw $t0, ($s1)

loop:
   lw $a0, ($s1)
   li $v0, 1
   syscall             # print integer

   la $a0, nline
```

```
li $v0, 4
syscall           #print new line

addi $s2, $s2, -1
addi $s1, $s1, 4    #next integer
bne $s2, $zero, loop  #continue until all elments are printed

li $v0, 10
syscall
```