# Homework #3: Chapter 2 - Advanced MIPS Assembly Language

CS 219: Computer Organization

**Due beginning of the class Monday February 29, 2016**

NOTE: The link for downloading/installation of MARS assembler and some references to help improve the understanding of MIPS are provided on the class website.
`http://www.cse.unr.edu/~simingl/Teaching/S16CS219/CS219.html`

Please write MIPS assembly language programs using MIPS MARS assembler for the following questions. **Please turn in the snapshots of your assembly programs.**

**1.** [30 points] Convert the C function below to MIPS assembly language. Make sure that your assembly language code could be called from a standard C program (that is to say, make sure you follow the MIPS calling conventions).

```
unsigned int sum(unsigned int n)
{
    if (n == 0)
        return 0;
    else
        return n + sum(n-1);
}
```

**Answer:**
The MIPS code is as follows:

```
sum:
        addi    $sp,    $sp,     -8     # Set up the stack
        sw      $ra,    4($sp)          # Save return address
        sw      $a0,    0($sp)          # Save argument $a0
        bne     $a0,    $0,      L1     # Jump to L1 if $a0 not equal to 0
        li      $v0,    0               # Set return value for base case
        addi    $sp,    $sp      8      # Restore $sp
        jr      $ra                     # Base case return
L1:     addi    $a0,    $a0,     -1     # $a0 = n-1
        jal     $sum                    # Recursive call
        lw      $ra,    4($sp)          # Restore $ra
        lw      $a0,    0($sp)          # Restore $a0
        addi    $sp,    $sp,     8      # Add return value to $t0
        addu    $v0,    $a0,     $v0    # Add return value to $t0
        jr      $ra                     # Return
```

**2.** [30 points] Find the greatest of all integers stored in an integer array in memory location starting at "IntArray" (a label in data segment). The integer array values are 0x5, 0x9, 0x4, 0xFF, 0xFE, 0xFA, 0x7, 0x2, 0x8, 0xFD. After finding the smallest integer in the array, store the result in memory location "result" (a label in data segment).

Hint: The data segment of your program will look like this:

```
           .data
IntArray:  .word 0x5,0x9,0x4,0xFF,0xFE,0xFA,0x7,0x2,0x8,0xFD   # initial values of integer array
size:      .word 10                                            # number of integers in IntArray
result:    .space 4                                            # declare 4 bytes to hold integer result

           .text
# start of text/code segment, write your code below
```

**Answer:**

```
          la    $s0,  IntArray           #$s0 = base address of the array
          addi  $t0,  $zero,   0         # initialize loop variable
          lw    $t1,  size               #$t1 = 10
          addi  $t9,  $zero,   0         # $t9 = 0, used for contain the greatest number
loop:     beq   $t0,  $t1,     endloop   # LOOP to find greatest
          sll   $t2,  $t0,     2         # $t2 will contain the offset of IntArray from base address
          add   $t3,  $s0,     $t2       # $t3 = &IntArray[i]
          lw    $t4,  0($t3)             # $t4 = IntArray[i]
          addi  $t0,  $t0,     1         # increment loop variable
          blt   $t9,  $t4,     greater   # if $t4 > $t9, branch to greater
          j     loop
greater:  addi  $t9,  $t4,     0         # move $t4 to $t9
          j     loop
endloop:  la    $t8,  result             # load address of the result
          sw    $t9,  0($t8)             # store the greatest value to the memory
```

**3.** [30 points] Write a program that calls a leaf procedure to solve for the equation: f = (g + h) - (i + j). For this program, variables (integers) g, h, i and j are stored in memory (data segment) and have values 14, 15, 1, and 21, respectively. Equivalent C code for leaf procedure that you will implement in MIPS assembly language is:

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

For calling the procedure, you need to store the arguments g, h, i, j in $a0, $a1, $a2, $a3 from your main program (calling procedure). During calculation use $s0 to store f (hence, need to save $s0 on stack in the callee). Put your result in $v0 in the leaf procedure. The calling procedure will store the result in memory location f once the result is returned in $v0 by the callee.

Hint: Your programs data segment will look like this:

```
            .data
g:          .word 14        # initial value of variable g
h:          .word 15        # initial value of variable h
i:          .word 1         # initial value of variable i
j:          .word 21        # initial value of variable j
f:          .space 4        # declare 4 bytes of storage to hold integer result

            .text
```

**Answer:**

```
main:
        lw      $a0,    g
        lw      $a1,    h
        lw      $a2,    i
        lw      $a3,    j
        jal     leaf
        sw      $v0,    f
        j       exit
leaf:
        addi    $sp,    $sp,    -4
        sw      $s0,    0($sp)
        add     $t0,    $a0,    $a1
        add     $t1,    $a2,    $a3
        sub     $s0,    $t0,    $t1
        add     $v0,    $s0,    $0
        lw      $s0,    0($sp)
        addi    $sp,    $sp,    4
        jr      $ra
exit:
```