COMP3350
Lab 1

Important Notes:

- There will be one submitted **assembly language file (.asm file)**. The **code** must be accompanied by adequate comments. Writing only the code could lead to **zero** credits.
- The submitted **lab report** should be **a single PDF file** unless otherwise mentioned. Texts of the reports are expected to be typed in for which you could use software programs like LaTeX, Microsoft Word etc. All screenshots (if any) should be included in this PDF file, clearly titled, and accompanied with adequate descriptions.
- The PDF file should be clearly formatted to help the TA/instructor to locate the answer to each corresponding question. If the student fails to label the answers with reasonable efforts or submit files with poor readability, the submission may only receive zero or partial score.
- Please use the following naming conventions for your submitted files.
    - Lab1.asm
    - Lab1report.pdf
- Students are encouraged to complete each task with best efforts. Even if the outcome isn't fully correct, partial credit may be awarded if the instructor and TA recognize that the student demonstrates a certain level of understanding.
- Solutions turned in must be your own. Please, mention references (if any) at the end of each question.

Please Complete the Following Task(s).

- **Task 1 (10 points) Write a MIPS program in MARS to swap the elements A[n] and A[m]. You may use Lab01template.asm as a reference template. Please take screenshots before and after assembling and executing your program. In your lab report, include a detailed description of your observations regarding the changes in memory and registers resulting from the execution of the program.**

Lab01template.asm:

```
.data
A: .word 7, 42, 0, 27, 16, 8, 4, 15, 31, 45
n: .word 3
m: .word 6

.text
.globl main

# Lab:
# Swap 2 elements A[n], A[m] in the array
# Please complete the code and record your observations with screenshots in a pdf
main:
    la $a0, A                       # Load address of array A to $a0
    lw $a1, n                        # Load word n
                                            # Load word m



    # shift left logical
    sll $t1, $a1, 2                 # Calculate offset for array index n
    add $t1, $a0, $t1                  # Calculate address of array index n
                                            # Calculate offset for array index m
                                            # Calculate address of array index m

    lw $t0, 0($t1)                   # Load value at array index n to $t0
                                    # Load value at array index m to a register

                                           # Store $t0 at array index m
                                            # Store $... at array index n
```