COMP 300

FORMAN CHRISTIAN COLLEGE

(A CHARTERED UNIVERSITY)

COMPUTER ORGANIZATION WITH ASSEMBLY LANGUAGE

Programming Assignment 1

Total Marks: 100

Hard Deadline: Monday Jun 7, 11:59 pm

This assignment MUST be completed in isolation on individual basis.

It's an open books and open notes task. Use of Internet is allowed, but you must provide references to web sites and /or tutorials from where you got help for this assignment. You MUST NOT share your work with any of your class fellow. Any such attempt will result in a ZERO grade.

Important Note:

- **Start early.**
- **You need to go through the MIPS assembly video lectures uploaded in the last week as well as in this week to complete this assignment.**
- **A delayed submission will be capped with 20% reduction in marks per day. This means if your submission is five days late, you will get a ZERO grade in that assignment.**
- **You need to upload the file/s of working code on google classroom.**
- **If you have multiple parts in a home work, make sure to submit code file of each part separately.**
- **Each file should be saved following the naming format given below as an example:**

  `pa-1-part-1<your FCC roll number>.asm`

  Here pa stands for programming assignment 1, followed by part 1 (or 2, …) of the handout.
- **You should zip your files and name the file as <pa-1<your roll number>>, for example pa-1-25-10548**

Well formatted report should also accompany the code files. Report should be comprehensive and should carry step by step description of your logic.

You MUST add a data dictionary at the start of your program as follows:

```
#Program Name: a1_pb1.asm
#Programmer Name: Rauf Butt
#Programmer Roll Number: 99-99999
```

Problem-1 [50 Marks]

In this assignment you need to write a program in MIPS assembly language. Your program should perform following tasks:

- It should prompt the user to enter five numbers one by one.
- Once user has entered all five numbers, your program should add these numbers and display the result.

A skeleton of program is shown. You MUST follow the pattern

- Call a function get_input
- This function should use a function printPrompt to provide appropriate prompt to the user and obtain five numbers from the user one by one.
- Make sure to store all these five numbers on stack.
- Next your program calls add_Num function. This function accepts five arguments, adds these and returns the result back to main using stack.
- Finally in the main function your program should display the result using a function display_output. Note the output in the sample run. Your program should produce exactly the same output.
- A WORD OF CAUTION:
  - USE STACK TO STORE THE INPUT ARGUMENTS TO DIFFERENT FUNCTIONS.
  - YOUR RETURN VALUE/S AND RETURN ADDRESSES SHOULD ALSO BE STORED ON STACK, EVEN IF YOU THINK IT IS NOT NECESSARY.
  - **DO NOT USE $V0 OR $V1 FOR RETURNING VALUES FROM SUBPROCEDURES.**
  - YOU CAN ONLY USE $t0 AND $t1 AS TEMPORARY DATA REGISTERS IN YOUR CODE.

Given is a base line idea of how to formalize this task. Make sure not to deviate form the base line format. Use program stack intelligently.

```
.data
     #your data goes here
.text
main:
     jal  get_input
     jal  add_num
     jal  display_output
     jal  exit_program
```

```
get_input:

     #here you should repeatedly call printPrompt function  to
provide a #prompt followed by instructions to read an integer from
user.


printPrompt:

     #this function prints a string on the screen


add_num:

     #this function receives five arguments from the caller and adds
these     #numbers. It then returns the result to the caller.

     #Make sure that you CANNOT use any data register other than $t0
and $t1.


display_output:

     #this function displays the output as per format described in
the sample     #run shown below.


exit_program:

     #this function terminates the program gracefully.
```

A sample run of the program is shown:

```
Enter first number: 2

Enter second number: 3

Enter third number: 4

Enter fourth number: 5

Enter fifth number: 6

2 + 3 + 4 + 5 + 6 = 20
```

Problem-2 [50 Marks]

My friend Mr. Stupid has given you machine code of an R type MIPS instruction and wants you to write a program to dissect this instruction into its relevant portions.
Fortunately, we know that an R-type instruction in MIPS consists of the following sections:

Opcode	Rs	Rt	Rd	Shamt	funct
6bits <31…26>	5bits <25…21>	5bits <20…16>	5bits <15…11>	5bits <10…6>	6bits <5…0>

You need to write a program in MIPS assembly language that accepts a 32-bit hexa-decimal number (hard coded in the program). The program should output information for Mr. Stupid as shown below. Note that all values in the output are in decimal format except the one that starts with 0x which is a hexadecimal value:

```
Instruction in hexa-decimal format: 0x24ac15fd
Opcode field <6 bits>:               9
Rs field <5 bits>:                   1
Rt field <5 bits>:                   12
Rd field <5 bits>:                   2
Shamt field <5 bits>:                23
Funct field <6 bits>:                61
```