Computer Architecture and Organization
CE 201
Second 2023/2024

Task 1:

Using MARS simulator, write the **equivalent assembly code** (MIPS instructions) of the below C program. **Note:** consider the data type and function while writing your assembly code.

Task 2:

Translate the below assembly code (MIPS instruction) into the **equivalent machine code**. **Note:** Use the attached datasheets for the translation.

The submission

The solutions should be in a word document (**HARD COPY**), including the code, your comments, and print screens of the MARS outputs. For task 2, create a table to show the machine code of each instruction. **The due for submission is 02/06/2024.**

Task 1:

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
//Function Definitions
int modulus(int x, int y)
{
    int mod = 0;
    if (y == 0){
        printf("\nError: Please enter another value of Y.");
    }
    mod = x%y;
    return mod;
}
int power(int x, int y)
{
    int p;
    p=pow(x,y);
    return p;
}
// Global variabels
int Loop = 1;
// Main Program
int main()
{
    int X;
    int Y;
    char operation;
    int Results;
    printf("\nPlease enter first number   : ");
    scanf("%d",&X);
    printf("\nPlease enter second number   : ");
    scanf("\n%d",&Y);
    while(Loop)
    {
        printf("\nPlease enter the operation   : ");
        scanf("\n%c",&operation);
        switch(operation)
        {
            case '%': Results = modulus(X,Y);
                      printf("%u mod %u = %u \n",X,Y,Results);
                      break;
            case '^': Results = power(X,Y);
                      printf("\n%u to the power %u = %u \n",X,Y,Results);
                      break;
            case 'q': exit(0);
                      break;
            default : system("cls");
        }
    }
}
```

Task 2:

```
         move $s0, $zero
Loop:    slti $t0, $s1, 0
         bne  $t0, $zero, exit
         sll  $t1, $s1, 2
         add  $t2, $s2, $t1
         lw   $t3, 0($t2)
         lw   $t4, 4($t2)
         slt  $t0, $t4, $t3
         beq  $t0, $zero, exit
         move $a0, $s2
         move $a1, $s1
         jal  Function
         addi $s1, $s1, -1
         j    Loop
exit:    addi $s0, $s0, 1
```