Arquitectura de computacional

Implementar los siguientes códigos en c utilizando el ISA del procesador MIPS:

1)

```c
int main(void) {

    int selector = 0; //registro s0
    int a = 5;  // registro s1
    int b = 3;  // registro s2
    int c = 0; // registro s3


       switch (selector)
        {
            case 1:
                c = suma(a,b);
                break;
            case 2:
                c = resta(a,b);
                break;
            case 3:
                c = multiplica(a, b);
                break;
            default:
                c = andBitwise(a,b);
                break;
        }

}
```

```c
int suma(int a, int b)
{
    int c = 0;
    c = a + b;
    return c;
}
int resta(int a, int b)
{
    int c = 0;
    c = a - b;
    return c;
}
int multiplica(int a, int b)
{
    int c = 0;
    c = a * b;
    return c;
}
int andBitwise(int a, int b)
{
    int c = 0;
    c = a & b;
    return c;
}
```

2)

```
int Vector_1[9] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
int Vector_2[9] = { -1,2,-3,4,-5,6,-7,8,-9 };

int i = 0;
int resultado = 0;

int Producto(int, int);

int main(void) {

    for (i = 0; i < 9; i++)
    {
        result = result + ProductFunction(Vector_1[i], Vector_2[i]);
    }
}


int ProductFunction(int a, int b)
{
    return(a*b);

}
```

3)

```
int main(void) {
    int potencia;

    potencia = Potencia(6, 6);


    return 0 ;
}



int Potencia(int m,int n){

    if (n < 1){
        return(1);
    }
    else {
        return(m*Potencia(m,n - 1));
    }
```

4)

```c
#include <stdio.h>

int division (int a, int b);
int result, i=0;

int main(void) {

    result = division(10,4);

}


int division (int a, int b) {
    if(b > a)
    {
        return 0;
    }
    else
    {
        return division(a-b, b) + 1;
    }
}
```

5) Implementar un código que sume los elementos de un arreglo de enteros de manera recursiva. La función recibe como argumentos la dirección inicial del arreglo y el número de elemento que contiene el arreglo.

En esta tarea se debe entregar loa códigos en c (.c) y ensamblador (.asm) de cada uno de los ejercicios en un archivo .zip o .rar. Todos los códigos deben ser funcionales y estar comentados, en caso contrario se reducirá la calificación de la tarea. En esta tarea se no permite el uso de pseudoinstructions. Para el caso de cargar la dirección de memoria en un registro se tiene que usar lui y ori y no addi.