TAREA ACADÉMICA DEL LABORATORIO DE ORGANIZACIÓN Y ARQUITECTURA DE COMPUTADORAS

SESIÓN N°:	2

TEMA:	PROGRAMACIÓN EN ENSAMBLADOR DE 64 BITS Y RENDIMIENTO

INDICACIONES GENERALES

- Leer la guía teórica correspondiente a esta sesión para complementar su desarrollo.
- Cualquier consulta debe ser realizada a en los foros correspondientes
- Estructurar su archivo con el formato **T2_codigo.zip** (i.e. T2_20082169.zip). En el archivo comprimido debe contener 2 carpetas: Pregunta1 y Pregunta2 que contendrán los programas codificados.
- Presentar su solución antes del 06/05/2020 a las 19:50 en PAIDEIA sección Laboratorio 2/Tarea.

OBJETIVOS

- Brindar al alumno claridad de conceptos acerca de la programación en assembler de 64 bits y rendimiento.
- Codificar programas en lenguaje ensamblador de 64 bits utilizando sintaxis Intel.
- Aprender a utilizar el concepto de *Calling Convention*.
- Aprender a utilizar las formulas para medir el rendimiento de un programa C.

CUESTIONARIO

PREGUNTA N°1 **(2.5 ptos)**

La universidad ha solicitado a los alumnos la elección de representantes estudiantiles. Con este motivo, se realizó una votación virtual. Cada candidato será representado por un número menor a 5 [MAX] que se generará de manera aleatoria. Siendo así se le pide contar los votos por candidato. Para ello, deberá realizar lo siguiente:

a. Crear la función **MostrarVector** en lenguaje C que permita imprimir un vector. Esta función debe tener como parámetros la longitud del mismo y el vector a mostrar.

```
1    int len; //Longitud del arreglo
2    float* vector; //Puntero del arreglo de floats
```

b. Crear la función **GenerarVotos** en lenguaje C que pueda generar un vector el cual contenga los votos emitidos de forma aleatoria. Esta función debe tener como parámetros la cantidad de votos emitidos [LEN], el máximo número [MAX] que puede tomar un candidato y el vector [V] donde guardará los votos emitidos.

- **LEN = 100 (entero positivo)**: Cantidad de votos emitidos.
- **MAX = 5 (entero positivo)**: Máximo número que puede tomar un candidato.
- **V (coma flotante)**: Vector que contenga los votos emitidos.

```
1    int len; //Longitud del arreglo
2    float* vector; //Puntero del arreglo de floats
3    float max; //Mayor valor dentro del arreglo
```

c. Crear la función **ContarVotos** en lenguaje C que permita contar la cantidad de votos y almacenarlas en otro vector. En este vector, la cantidad de votos para cada número de candidato se guardará en la posición con dicho número como índice. Esta función debe tener como parámetros la cantidad de votos [LEN], el vector con los votos emitidos [V] y el vector donde guardará el conteo de votos [C]

```
1    int len; //Longitud del arreglo
2    float* vector; //Puntero del arreglo de floats
3    float* cuenta; //Puntero del arreglo de floats
4    float max; //Mayor valor dentro del arreglo
```

d. Codificar el programa principal **main** en lenguaje C . Este programa utilizará las funciones anteriores para generar y mostrar todos los vectores solicitados. El programar debe tener la siguiente estructura:

```
1   #include <time.h>
2   #include <math.h>
3   void GenerarVotos(int LEN, int MAX, float *v);
4   void MostrarVector(int LEN, float *vector);
5   void ContarVotos(int LEN, float *v, float *ch);
6   extern void ContarVotosAsm(int LEN, float *v, float *asmh);
7   int main(){
8
9       srand(time(NULL));
10
11      int i = 0;
12
13      int LEN=100, MAX=5;     //LEN longitud del histograma/
            vector y MAX numeros en el vector
14      float *v, *ch, *asmh;
15
16
17
18      GenerarVotos(LEN,MAX,v);
19
20      printf("V = [ ");
21      MostrarVector(LEN, v);
22
23      ContarVotos(LEN, v, ch);
24      printf("  cH = [ ");
25      MostrarVector(LEN, ch);
26
27      ContarVotosAsm(LEN, v, asmh);
28      printf("asmH = [ ");
29      MostrarVector(LEN, asmh);
30
31      return 0;
32  }
```

e. Codificar el programa **ContarVotosAsm** en lenguaje ensamblador que permita generar un vector con la cantidad de votos emitidos respetando la estructura utilizada en el main.

f. Con el procedimiento 'ContarVotos' obtenido en el indice "c)" en lenguaje C, realice una optimización mediante *(loop unrolling o loop fusion)* en un nuevo procedimiento llamado 'ContarVotosOptimizado'.

   - La lógica del programa debe ser la misma que la usada para la parte "c)" de la experiencia (la optimización debe partir del mismo codigo).

g. Una vez completado los indices anteriores, calcule el tiempo total de invocar 1000

veces el procedimiento en C, y repita el proceso con el procedimiento en C optimizado y el procedimiento externo en Assembler.

- Use los siguientes los siguientes procedimientos creados anteriormente:

```
1  void ContarVotos(...);
2  void ContarVotosOptimizado(...);
3  extern void ContarVotosAsm(...);
```

- Debe mostrar en pantalla el tiempo total en microsegundos(us):

```
En C el tiempo total fue: %.2f us
En C optimizado el tiempo total fue: %.2f us
En ASM el tiempo total fue: %.2f us
```

h. Con los tiempos de ejecutar 1000 veces el programa en Assembler y en C, calcule el SpeedUp entre ellos.

- Debe mostrar en pantalla lo siguiente:

```
El speedup de ASM  con respecto a C es: %.2f
El speedup de C  con respecto a ASM es: %.2f
```

i. Interprete brevemente los valores de SpeedUp obtenidos en el indice anterior.
j. Si el valor del SpeedUp del programa en Assembler con respecto a C hubiera sido 0.1, ¿Qué podríamos concluir de la eficiencia de tiempo de ambos programas?.
k. Si el compilador de C recibe una actualización en el futuro, y antes el SpeedUp de nuestro programa en C, con respecto a nuestro programa en Assembler, era de 2; y ahora con el nuevo compilador es 5, esto cómo se puede interpretar. ¿Fue favorable la actualización de C?

PREGUNTA N°2 **(2.5 ptos)**

En matemáticas, una serie de Taylor es una aproximación de funciones mediante una serie de potencias o suma de potencias enteras de polinomios como $(x-a)^n$
llamados términos de la serie.

$$\frac{a}{1-x} = \sum_{n=0}^{N} ax^n$$

para

$$|x| > 1 \tag{2}$$

Se le pide desarrollar un programa en lenguaje C el cual permita aproximar el valor de un número mediante su expansión por serie de Taylor. Para ello, deberá realizar lo siguiente:

a) Crear la función **GenerarNumeros** en lenguaje C que permita crear un vector con números entre 0 y 1 (se recomienda usar los siguientes numeros: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9). Esta función recibirá los siguientes parámetros de entrada:

```
1    int len; //Longitud del arreglo
2    float* x; //Puntero del arreglo de floats que contendra
3              //numeros entre 0 y 1
4
5    float* y; //Vector donde se guardara
6              //el valor aproximado de cada elemento del
7              //vector X luego de aplicar Taylor(inicialmente
                   tendran el valor de 0)
```

b) Crear la función **Taylor** en lenguaje C que permita hallar la serie de Taylor de los números de un vector y almacenarlos en otro. Esta función recibirá los siguientes parámetros de entrada:

```
1    int len; //Longitud del arreglo
2    int aprox; //Aproximacion de la serie
3    float* x; //Puntero del arreglo de floats que contendra
4              //numeros entre 0 y 1
5
6    float* y; //Vector donde se guardara
7              //el valor aproximado de cada elemento del
8              //vector X luego de aplicar Taylor
9    float a; //
```

c) Crear la función **main** en lenguaje C que permita hallar la serie de Taylor utilizando las funciones anteriores. Debe seguir la siguiente estructura:

```
#include <time.h>
#include <math.h>

void GenerarNumeros(int LEN, float *x, float *y);
void Taylor(int LEN, int APROX, float *x, float *y, float a);
extern void asmTaylor(int LEN, int APROX, float *x, float *y,
    float a);

int main(){
    srand(time(NULL));

    int LEN = 10, APROX = 100, i;
    float *x, *y, a = 3.5;


    GenerarNumeros(LEN,x, y);
    cTaylor(LEN, APROX, x, y, a);
    printf("En C:\n");
    for(i = 0; i < LEN; i++){
            printf("Numero: %.2f    Aproximacion: %.3f \n", x[
                i], y[i]);
    }

    printf("\n");

    asmTaylor(LEN, APROX, x, y, a);
    printf("En ASM:\n");
    for(i = 0; i < LEN; i++){
            printf("Numero: %.2f    Aproximacion: %.3f \n", x[
                i], y[i]);
    }
}
```

d) Codificar el programa **TaylorAsm** en lenguaje ensamblador que permita hallar la serie de Taylor respetando la estructura utilizada en el main.c.