UNIVERSIDAD DE SONORA

INTRODUCCIÓN AL SOFTWARE BASE

Proyecto Final (20 puntos)

Desarrollar un programa en Ensamblador para la familia x86 de microprocesadores de Intel que cumpla con las siguientes características:

0. Un encabezado con nombre del programa, un ejemplo de cómo puede utilizarse el programa en la línea de comando, nombre de los integrantes del equipo, fecha.

1. El programa, al mandar ejecutar, utilice los argumentos de la línea de comandos para extraer el nombre del archivo de texto a leer **(1 punto).**
    a. En caso de que exista el archivo a leer:
        i. Lea el archivo de texto especificado en modo de "Lectura únicamente" (Read only) y guarde en un buffer el texto leído.
        ii. Copie cada línea del buffer en un arreglo de enteros. (Cada línea está separada por el carácter 0xA "Line Feed")

            Ejemplo del contenido del archivo de texto:

            ```
            4
            7
            10
            ```

    b. Si no existe el archivo, el arreglo debe permanecer limpio.

2. Mostrar el siguiente menú en pantalla **(1 punto)**:

    ```
    *** MENU ***

    1. Agregar dato
    2. Generar línea
    3. Generar curva
    4. Mostrar datos (imprimir)
    5. Guardar archivo
    0. Salir
       Opción >_
    ```

    Las únicas opciones válidas son 0, 1, 2, 3, 4 y 5, cualquier otra opción hará que se imprima de nuevo el menú en pantalla. Después de ejecutar alguna opción, se deberá volver a mostrar el menú.

    a. En caso de seleccionar opción 1 (Agregar dato), se deberá mostrar la siguiente leyenda **(5 puntos)**:

```
Número entero>
```

y se deberá poder guardar el número en la primer celda disponible del arreglo de entrada (recordar convertir a entero con atoi).

b. En caso de seleccionar opción 2 (Generar línea), el arreglo de enteros de entrada será pasado por un ciclo y generará un nuevo arreglo de enteros, con el resultado del siguiente cómputo : 4x +3 **(5 puntos)**.

Arreglo de entrada	Arreglo de resultados
1	7
2	11
3	15

c. Opción 3 (Generar curva). Recorrer el arreglo números enteros y generar un nuevo arreglo de enteros, con el resultado del siguiente cómputo: x^3 - 4x^2 + 6x - 24 **(5 puntos)**. Ejemplo:

Arreglo de entrada	Arreglo de resultados
4	0
5	31
6	84

d. Opción 4 (Imprimir). Recorrer el arreglo de entrada y resultados e imprimir en pantalla cada celda. Después, volver a mostrar el menú principal **(1 puntos)**. Ejemplo:

```
Arreglo de entrada      Arreglo de resultados
==================      =====================
   1                          7
   2                          11
   3                          15
```

*(hasta **2 puntos extra**: Mostrar el cálculo de el número Máximo (1) y Mínimo (1) del arreglo de resultados)*

e. Opción 4 (Guardar archivo). Muestra en pantalla la pregunta:

```
¿Nombre de archivo a guardar?>
```

y después de aceptar el nombre del archivo por el usuario, crea el mismo, y debe guardar renglón por renglón el nombre del usuario y su calificación.
Como no podemos grabar enteros, es necesario convertir el número entero a cadena de

texto (en el Anexo 1 viene el código de la función itoa que permite convertir enteros a cadenas de texto) **(2 puntos)**. Ejemplo del archivo del contenido del archivo de texto:

```
1,7
2,11
3,15
```

f. Opción 0. Salir. Debe salir del programa sin mostrar mensaje de error **(1 punto)**.
g. **Puntos extra:** Que el programa genere 2 archivos de texto, uno para la línea y otro para la curva, al mismo tiempo (sin necesidad de que el usuario teclee dos veces el nombre del archivo) **(5 puntos)**.
h. **Puntos extra**: Programa limpio, en bloques, con comentarios de qué realiza cada bloque y cada línea **(3 puntos)**.

Anexo I. Código de itoa (Integer to ASCII)

```
;; -----------------------------------
;; itoa recibe un entero
;; y lo convierte en cadena de texto
;; recibe entero en EAX
;; recibe direccion de cadena en ESI
;; -----------------------------------
itoa:
    push ebx  ; preservamos ebx
    push ecx  ; preservamos ecx
    push edx  ; preservamos edx
    push esi  ; preservamos esi

    mov ebx, 10     ;vamos a dividir por 10
    mov ecx,  0     ;nuestro contador en  0
    push ecx        ;mandamos 0 al stack (fin de cadena)
    inc ecx
.dividir:
    inc ecx         ;incrementamos nuestro contador
    mov edx,  0     ; limpiamos EDX para dividir
    idiv ebx         ;dividimos EAX entre EBX
```

```
    add edx, 0x30   ;agregamos 48 (para obtener digitos de 0-9 en ASCII)
    push edx        ;enviamos el residuo al stack
    cmp eax, 0      ;checamos si el residuo es 0
    je .fuera       ;si es 0, salimos del ciclo
    jmp .dividir    ;seguimos obteniendo digitos

.fuera:
    mov ebx,0       ;limpiamos ebx
.guardar:
    pop eax         ;traemos un digito del stack
    mov byte[esi+ebx],al   ;movemos el digito a memoria
    inc ebx         ;incrementamos contador
    cmp ebx,ecx     ;comparamos con la cuenta de digitos
    jne .guardar    ;si no son iguales, obtenemos otro digito del stack

    pop esi         ; restauramos esi
    pop edx         ; restauramos edx
    pop ecx         ; restauramos ecx
    pop ebx         ; restauramos ebx
    ret             ; y regresamos
```