UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Estructura y Programación de Computadoras

Grupo: 02 - Semestre: 2021-1

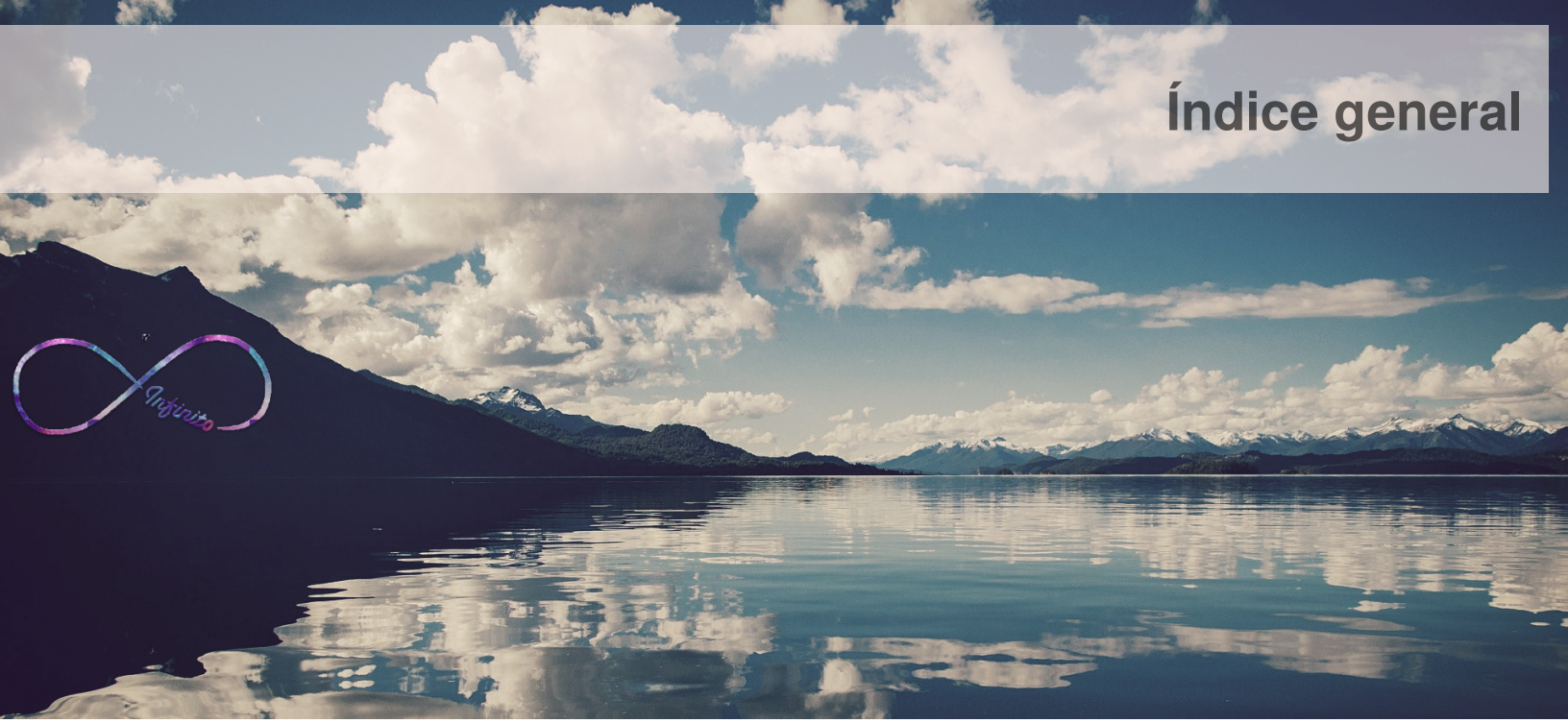
Proyecto 2:
"Reloj-cronómetro" con interfaz gráfica

FECHA DE ENTREGA: 03/02/2021

Alumnos:

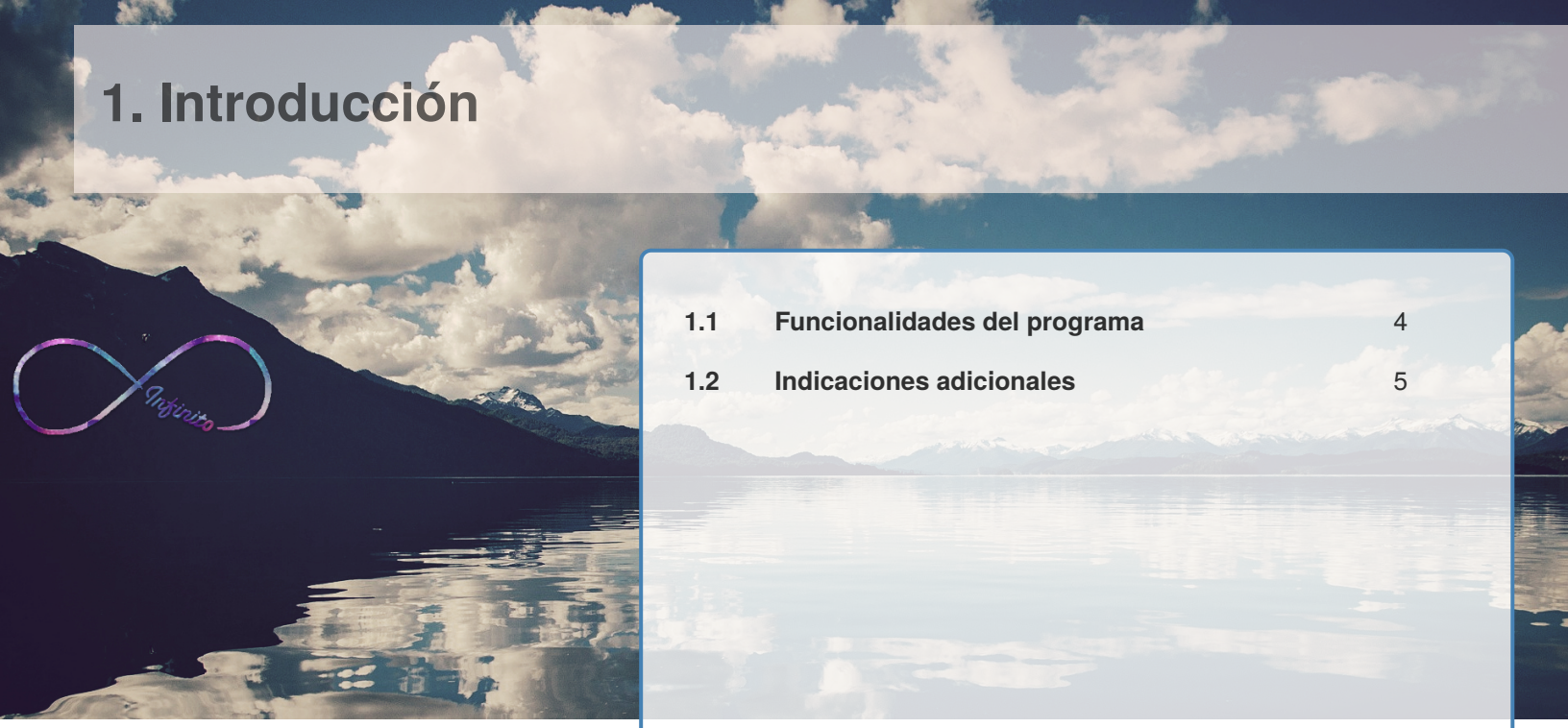
Esquivel Razo Israel

Téllez González Jorge Luis



Índice general

1. Introducción



En el capítulo inicial se introduce al problema planteado por la implementación gráfica del programa reloj-cronómetro desarrollado en x86.

1.1. Funcionalidades del programa

El problema planteado por el proyecto es el siguiente: *Desarrollar un programa en lenguaje ensamblador para arquitectura Intel x86 que tenga las funciones de reloj y de cronómetro, imprimiendo en pantalla una interfaz gráfica que permita la interacción computadora-usuario a través del mouse de la computadora. Se deberá imprimir en pantalla la hora del sistema si el usuario selecciona ver el reloj, o bien, deberá mostrar un cronómetro con varias funciones si selecciona ver el cronómetro.*

Se solicita que el programa contenga un menú **gráfico** que solicite al usuario realizar un click izquierdo con el ratón para seleccionar 3 opciones diferentes:

- **Inicio**
- **Cronómetro**
- **Salir**

Si el usuario selecciona la opción **Reloj**, el programa deberá mostrar em pantalla una interfaz gráfica con la hora del sistema, en formato HH:MM:SS (HH-horas, MM-minutos, SS-

segundos) y cada segundo se debe actualizar. Si selecciona la opción Cronómetro, el programa deberá funcionar como un cronómetro y mostrarlo en pantalla en formato MM:SS.mmm. El cronómetro debe tener 3 opciones que deben seleccionarse a través del mouse y se mostrarán como botones en la interfaz gráfica:

- **Iniciar/Continuar**
- **Detener (pausa)**
- **Reiniciar**

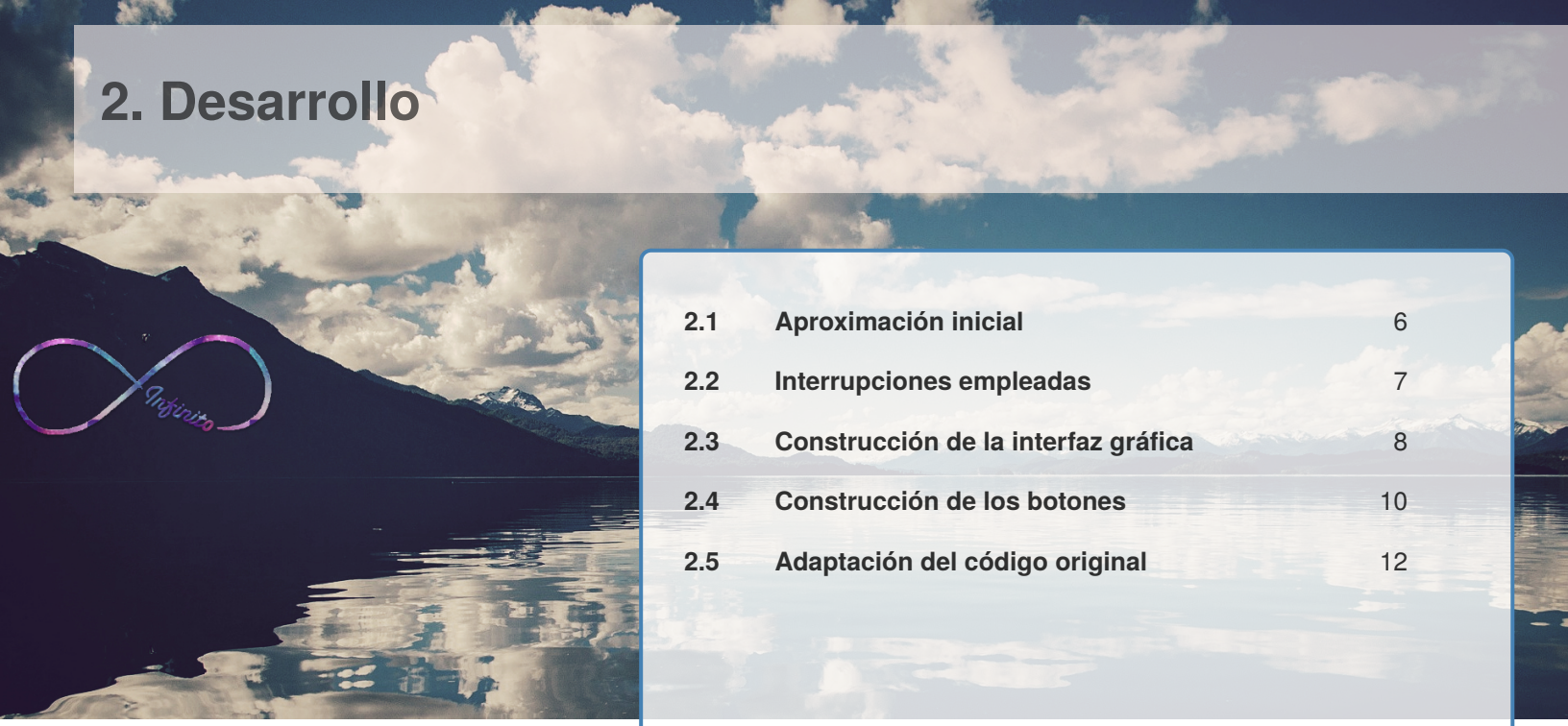
El cronómetro debe iniciar en ceros. Si el usuario selecciona la opción de **Iniciar/ Continuar**, el cronómetro arranca y empieza a contar, el cronómetro se debe actualizar todo el tiempo. Si mientras el cronómetro está corriendo, el usuario selecciona la opción **Detener (pausa)**, el cronómetro se detiene, y en cualquier momento si el usuario vuelve a seleccionar la opción **Iniciar/Continuar** el cronómetro continúa desde el punto en el que se quedó. Por último, si el usuario selecciona la opción **Reiniciar** en cualquier momento, el cronómetro vuelve a su estado inicial.

Finalmente, para regresar al menú, el usuario debe hacerlo a través de la interfaz gráfica, es decir, utilizando el mouse.

1.2. Indicaciones adicionales

- El reloj puede imprimirse en formato de 24 horas o de 12 horas, si es el segundo deberá mostrar si la hora es AM o PM.
- Cuando el cronómero pase de 59:59.999, se reinicia el conteo.
- La vista del reloj y la del cronómetro se deben hacer por separado, no será válido que en una sola vista se visualicen ambos. Cuando se seleccione una opción u otra, se deben visualizar cambios en la interfaz gráfica.

2. Desarrollo



En este apartado se mostrarán y explicarán las propuestas diseñadas para la solución del problema planteado así como la justificación de su uso.

2.1. Aproximación inicial

Para iniciar el trabajo desarrollado se utilizó el código proporcionado como base para el proyecto, el cuál contiene una implementación básica de la interfaz gráfica del programa implementando macros para el uso del ratón y generar una interfaz gráfica básica que posibilita salir del programa haciendo click a un botón:

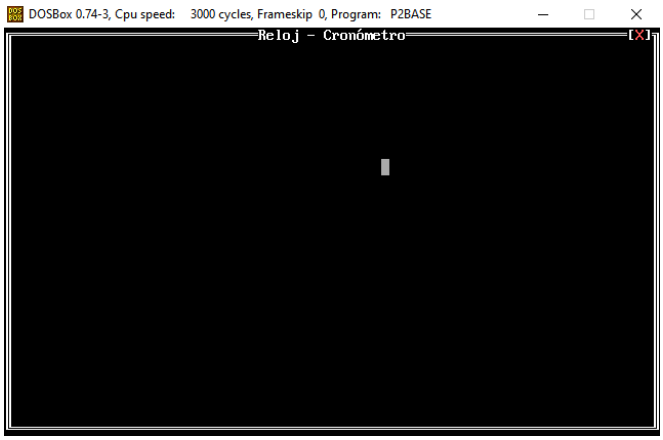


Figura 2.1: *Vista general del código base con una interfaz implementada.*

A partir del código proporcionado, se desarrollaron por medio de pruebas consecutivas los procedimientos y las macros empleadas para dibujar los elementos de la interfaz gráfica y realizar las adaptaciones necesarias para que el programa original admitiese el uso de una interfaz gráfica para modificar su flujo.

Para iniciar con la creación de la IU, se realizaron pruebas experimentales para comprender como realizar dibujos en pantalla y manipular su impresión de tal forma que se mostrasen en la ubicación que se quisiera previo a desarrollar la adaptación del código previo a una versión gráfica.

2.2. Interrupciones empleadas

A continuación se muestran las interrupciones empleadas durante el desarrollo del programa propuesto:

- **int 10h**: Esta interrupción permite acceder al modo video para realizar operaciones de impresion en pantalla entre otras utilidades.
  - Opción $AH = 00h$: Llama al modo video.
  - Opción $AH = 01h$: Modifica la visibilidad del cursor del teclado.
  - Opción $AH = 02h$: Permite posicionar el cursor utilizando los registros altos y bajos de DX.
  - Opción $AL = 03h$: Establece el modo texto a 16 colores y 80x25.
  - Opción $AH = 09h$: Permite escribir un caracter definiendo el número de página, su color y la cantidad de veces que se debe de escribir el caracter.
  - Opción $AH = 13h$: Permite escribir una cadena definiendo el número de página, su color y la cantidad de veces que se debe de escribir el caracter.
- **int 33h**: Provee servicios de uso para el ratón.
  - Opción $AX = 0000h$: Reinicia el servicio del mouse y verifica si hay soporte de drivers para su uso.
  - Opción $AL = 01h$: Habilita la visibilidad del cursor del mouse.
  - Opción $AL = 03h$: Pemite revisar el estado del raton y la posición actual del puntero.

2.3. Construcción de la interfaz gráfica

Como se comentó previamente, el enfoque inicial para la elaboración del proyecto consistió en realizar un análisis del funcionamiento del código proporcionado como base para comprender como realizar la impresión de caracteres y cadenas en la IU del programa. A partir de esto, se desarrollaron las siguientes *macros* para asistir en el proceso de dibujar cajas o rectángulos y poder manipular su tamaño y color:

1. La primer macro empleada para tal proposito se trata de **dibuja_rectan_color**, la cuál recibe como parámetros el renglón de inicio y final del rectángulo para definir su largo, el inicio y final de la columna para su ancho, el color del texto a introducir y el color de fondo del rectángulo generado.

   Para realizar el proceso de impresión, se utilizan dos variables auxiliares para almacenar el renglón y la columna de inicio con el fin de posicionar el cursor en ese punto usando la macro **posiciona_cursor** presente en el código base.

```
mov [ren_aux], renglon_inicio
renglon_2:
mov [col_aux], columna_inicio
columna_2:
posiciona_cursor [ren_aux], [col_aux]
imprime_caracter_color " ", color_Texto, color_Fondo
```

Figura 2.2: *Variables auxiliares y posición inicial de impresión.*

   Después, se utiliza la macro **imprime_caracter_color** para imprimir los colores definidos para ese recuadro, sin imprimir texto en el mismo. Posteriormente, se incrementa el valor de la variable auxiliar para las columnas y se compara si su valor es igual al definido como la columna final del rectángulo que se quiere generar; mientras lo anterior no se cumpla se generará un ciclo de impresión repetitivo sobre el renglón de inicio *dibujando* la primer línea del rectángulo.

```
posiciona_cursor [ren_aux], [col_aux]
imprime_caracter_color " ", color_Texto, color_Fondo
inc [col_aux]
cmp [col_aux], columna_fin
jb columna_2
je columna_2
jg siguiente_1
```

Figura 2.3: *Impresión de la primer línea del rectángulo.*

   Tras alcanzarse el valor de la columna final, en la etiqueta *siguiente_1* se realiza el incremento de la variable auxiliar de los renglones, y mientras su valor no sea igual al

del renglón final, se reiniciará el valor de la columna de inicio para dibujar nuevamente sobre ese renglón de tal forma que se va realizando una impresión progresiva línea por línea del rectángulo hasta que se llega al renglón final y no queda más por dibujar.

```
siguiente_1:
inc [ren_aux]
cmp [ren_aux], renglon_fin
jb renglon_2
je renglon_2
jg siguiente_2
siguiente_2:
```

Figura 2.4: *Impresión progresiva por renglón del rectángulo.*

2. La segunda y tercer macro llamadas **dibuja_humo_color** y **dibuja_humo_color2**, respectivamente, siguen una lógica muy similar al caso anterior, sin embargo, presentan modificaciones para que resulten útiles con el fin de imprimir un patrón alternado de colores que simule un efecto de *humo* que será empleado para dar forma a la interfaz que será generada con forma de un cohete espacial.

   Nótese que, a pesar de seguir la misma lógica inicial que la macro anterior, al momento de realizar la comparación se tiene un doble incremento en el auxiliar de los renglones y en la comparación se realiza con un *salto* de 4 unidades. De este modo, la impresión progresiva se realizará con *saltos* que serán utilizados para simular un efecto de humo que irá en unión a la generación de una IU con forma de un cohete espacial.

```
mov [ren_aux], renglon_inicio
renglon_2_humo:
mov [col_aux], columna_inicio
inc [ren_aux]
inc [ren_aux]
posiciona_cursor [ren_aux], [col_aux]
imprime_caracter_color " ", color_Texto, color_Fondo
cmp [ren_aux], renglon_inicio + 4
jb renglon_2_humo
je renglon_2_humo
jg siguiente_2_humo
siguiente_2_humo:
```

Figura 2.5: *Macro **dibuja_humo_color**.*

Así mismo, para facilitar la impresión de elementos de la IU durante la ejecución del programa se implementaron los procedimientos **palabra_hola**, **marco_2**, **marco_2_2**, **opciones_botones_dibujo** para realizar el dibujo de los bordes del recuadro principal del programa que imitará la forma del cohete espacial mencionado previamente y dibujar los botones a utilizar en el submenú del cronómetro. Estos métodos hacen uso de las macros **dibuja_rectan_color** y la macro proporcionada previamente para la impresión de cadenas.

```
marco_2_2 proc
;Dibuja el borde exterior del menu principal.

    dibuja_rectan_color 4, 14, 4, 66, cBlanco, bgCafe
    dibuja_rectan_color 18, 14, 18, 66, cBlanco, bgCafe
    dibuja_rectan_color 5, 14, 17, 14, cBlanco, bgCafe
    dibuja_rectan_color 5, 66, 17, 66, cBlanco, bgCafe
    ret
endp

opciones_botones_dibujo proc
;Dibuja los botones del submenu del cronometro y el texto dentro de ellos.

    dibuja_rectan_color 4, 14, 18, 66, cBlanco, bgAmarillo
    dibuja_rectan_color 11, 19, 13, 29, cBlanco, bgVerde
    dibuja_rectan_color 14, 19, 16, 29, cBlanco, bgVerdeClaro
    dibuja_rectan_color 14, 35, 16, 44, cBlanco, bgCyanClaro
    dibuja_rectan_color 14, 50, 16, 60, cBlanco, bgCyan
    posiciona_cursor 12, 21
    imprime_cadena_color crono_boton1,7,cNegro,bgVerde
    posiciona_cursor 15, 20
    imprime_cadena_color crono_boton2,9,cNegro,bgVerdeClaro
    posiciona_cursor 15, 37
    imprime_cadena_color crono_boton3,6,cNegro,bgCyanClaro
    posiciona_cursor 15, 51
    imprime_cadena_color crono_boton4,9,cNegro,bgCyan
    call marco_2_2
ret
endp
```

Figura 2.6: *Procedimientos para la impresión de elementos de la IU.*

2.4. Construcción de los botones

Si bien la impresión de los botones se ha hecho con las macros descritas anteriormente, estas impresiones carecen de cualquier tipo de funcionalidad. Para implementar el uso del ratón sobre ellos de forma que al hacer click izquierdo se modifique el flujo del programa, se ha adaptado el código proporcionado originalmente para definir la lógica del ratón y poder salir de la interfaz base con el botón **X** para extenderlo al uso de los botones dibujados para su uso en la interfaz de usuario.

El procedimiento que implementa lo anterior se denomina **espacios_que_no_hacen_nada**, y no es porque no hagan nada en sí, si no porque se define con este procedimiento de forma estricta qué es lo que ocurre con el flujo de ejecución al presionar en determinadas zonas de la pantalla del programa ejecutándose en **DOSBox**.

De esta forma, se busca que únicamente los botones dibujados para la interfaz modifiquen el flujo al hacer click izquierdo sobre ellos y que presionar sobre cualquier otra parte del programa no altere o interrumpa el flujo de ejecución del reloj o el cronómetro.

Dicho esto, se tomó como base la lógica descrita previamente y se establecieron los límites de cada botón de forma personalizada de modo que, si se realiza click izquierdo sobre un determinado rango de columnas y renglones (Es decir, los mismos sobre los que se generaron los botones previamente) se realice un salto incondicional a una sección específica del código. El registro DX es usado para verificar los renglones y CX para las columnas de la siguiente manera:

```
;Se establece el flujo al que dirigira el boton de INICIO en el menu principal.
boton_inicio0:
    cmp dx, 4
    jge boton_inicio1   ;Se verifica el boton despues de los renglones
    jmp boton_reloj0    ;Se verifica al siguiente boton

boton_inicio1:
    cmp dx, 6
    jbe boton_inicio2   ;Se verifica el boton antes de los renglones
    jmp boton_reloj0    ;Se verifica al siguiente boton

boton_inicio2:
    cmp cx,4
    jge boton_inicio3   ;Se verifica el boton despues de las columnas
    jmp boton_reloj0    ;Se verifica al siguiente boton

boton_inicio3:
    cmp cx,13
    jbe boton_inicio4   ;Se verifica el boton antes de las columnas
    jmp boton_reloj0    ;Se verifica al siguiente boton

boton_inicio4:
    jmp et_inicio       ;Va a la etiqueta et_inicio si todas las condiciones se cumplen.
```

Figura 2.7: *Lógica del ratón para modificar el flujo de ejecución.*

Esta misma lógica se utiliza para modelar el resto de botones en la interfaz. Sin embargo, existe otro método llamado **espacios_que_no_hacen_nada_2** que a simple vista parece una copia idéntica del procedimiento anterior, sin embargo, tiene una función vital y una diferencia sutíl. Cuando se desarrolló el programa con el procedimiento descrito anteriormente, un problema a enfrentar fue el traslape de la sección del cronómetro con la sección del reloj o el menú principal debido a que la funcionalidad de los botones del submenú del cronómetro se quedaban aún después de haber salido de esa opción.

Entonces, este procedimiento es utilizado cuando se entra a la opción del cronómetro para que realice de nuevo todo el proceso de mapeo descrito anteriormente, sin embargo, añade una lógica extra que corresponde a los botones del submenú con el fin de que pueda modificarse el curso del cronómetro en su ejecución. Cuando se sale de la opción del cronómetro, este procedimiento deja de utilizarse y se vuelve a utilizar el procedimiento de lógica inicial para que la funcionalidad de los botones del submenú del cronómetro se elimine y no se traslapen con otras secciones del programa.

```
;Se establece el flujo al que dirigira el boton de Iniciar al estar en el submenu del cronometro.
boton_continuar0_2:
    cmp dx, 11
    jge boton_continuar1        ;Se verifica el boton despues de los renglones
    jmp boton_continuar0_2_2    ;Se verifica al siguiente boton

boton_continuar1:
    cmp dx, 13
    jbe boton_continuar2        ;Se verifica el boton antes de los renglones
    jmp boton_continuar0_2_2    ;Se verifica al siguiente boton

boton_continuar2:
    cmp cx,19
    jge boton_continuar3        ;Se verifica el boton despues de las columnas
    jmp boton_continuar0_2_2    ;Se verifica al siguiente boton

boton_continuar3:
    cmp cx,29
    jbe boton_continuar4        ;Se verifica el boton antes de las columnas
    jmp boton_continuar0_2_2    ;Se verifica al siguiente boton

boton_continuar4:
    jmp startcrono              ;Va a la etiqueta startcrono si todas las condiciones se cumplen.
```

Figura 2.8: *Flujo modificado para los botones del cronómetro.*

2.5. Adaptación del código original

El código original del proyecto anterior requirió ciertas modificaciones para adaptarlo al uso de la interfaz gráfica, sin embargo, la lógica principal del algoritmo implementado tanto para el reloj como para el cronómetro se mantuvo sin cambios; salvo un cambio en la interrupción usada para obtener la hora del sistema la cuál se sustituyó exitosamente con la interrupción **1Ah** Op. **02h** para obtener la hora del sistema y no sufrir el desfase por tomar la hora del sistema de DOSBox con la interrupción empleada previamente.

El programa comienza inicializando el ratón y verificando el soporte de drivers del mismo. Si no se encuentra el driver, se enviará un mensaje de estado para que el usuario puede salir del programa presionando la tecla enter. Pasada exitosamente esa verificación, se procede a realizar un salto **jz** hacia la etiqueta *imprime_ui*, la cuál contiene el código empleado para dibujar el marco exterior y la IU principal del programa con forma de cohete.

Figura 2.9: *IU principal del programa.*

Tras lo anterior, se hace visible el cursor del mouse con **muestra_cursor_mouse** y se realiza un loop de forma que, mientras no se presione nada de lo mapeado con el click izquierdo con el método **espacios_que_no_hacen_nada**, el programa se quedará en esa pantalla a la espera de que se haga click sobre alguno de los botones para modificar el flujo del programa y pasar a alguna de las opciones disponibles.

La etiqueta *et_reloj* contiene el código empleado para mostrar el reloj en pantalla. Imprimiendo nuevamente el marco café de la IU, se utiliza el mismo procedimiento descrito en el proyecto anterior, con la diferencia de que se utilizan las macros **posicionar_cursor** e **imprime_cadena_color** para imprimir en un loop la cadena que contiene la hora actual del sistema. Y además, para detener el loop de ejecución se utiliza la macro **lee_mouse** y la instrucción **test bx, 0001h** para verificar si se ha presionado en alguno de los botones del programa para romper el flujo actual y modificarlo.

```
loopstart:
;Loop de impresión del reloj, el cual puede ser interrumpido presionando cualquier tecla.
    lea BX, TIME
    call GET_TIME

    mov ah, 2ch             ;Llama a la función int 21h, Op. 2Ch que captura la hora del sistema.
    int 21h

    call GET_TIME
    lea DX, TIME

 posiciona_cursor 8, 20                         ;Coloca el cursor en la posicion deseada
 imprime_cadena_color TIME,31,cAzul,bgAmarillo  ;Imprime la cadena de tiempo del reloj


 ;mov ah, 01h            ;Rotura del bucle presionando cualquier tecla.
 lee_mouse              ;lee el mouse
 test bx, 0001h         ;Si BX = 0001h, boton izquierdo presionado
 jz loopstart
 jmp metodo_espacio_blancos  ;Se llama a la etiqueta, conectada a los botones
metodo_espacio_blancos:
    call espacios_que_no_hacen_nada
 jmp loopstart
```

Figura 2.10: *Adaptación del reloj a su versión gráfica.*

En la etiqueta **et_cronometro** se procede en primer lugar a llamar al método **opciones_botones_dibujo** para realizar el proceso de dibujado de los botones del submenú:

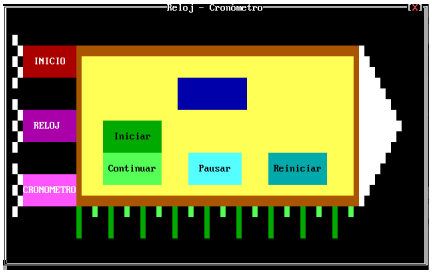


Figura 2.11: *Apariencia de los botones del cronómetro.*

Posteriormente, se limpia el búfer del teclado y el programa se queda a la espera de que se seleccione el botón **Iniciar** para que el cronómetro comience a correr. Cuando esto

ocurre, el programa salta a la etiqueta **startcrono** e inicia la ejecución del loop de impresión del cronómetro; considerando exactamente lo descrito en la sección del reloj para detener el loop según se seleccione otro recuadro del submenú, o bien, se desee cambiar al menú principal o a la sección del reloj.

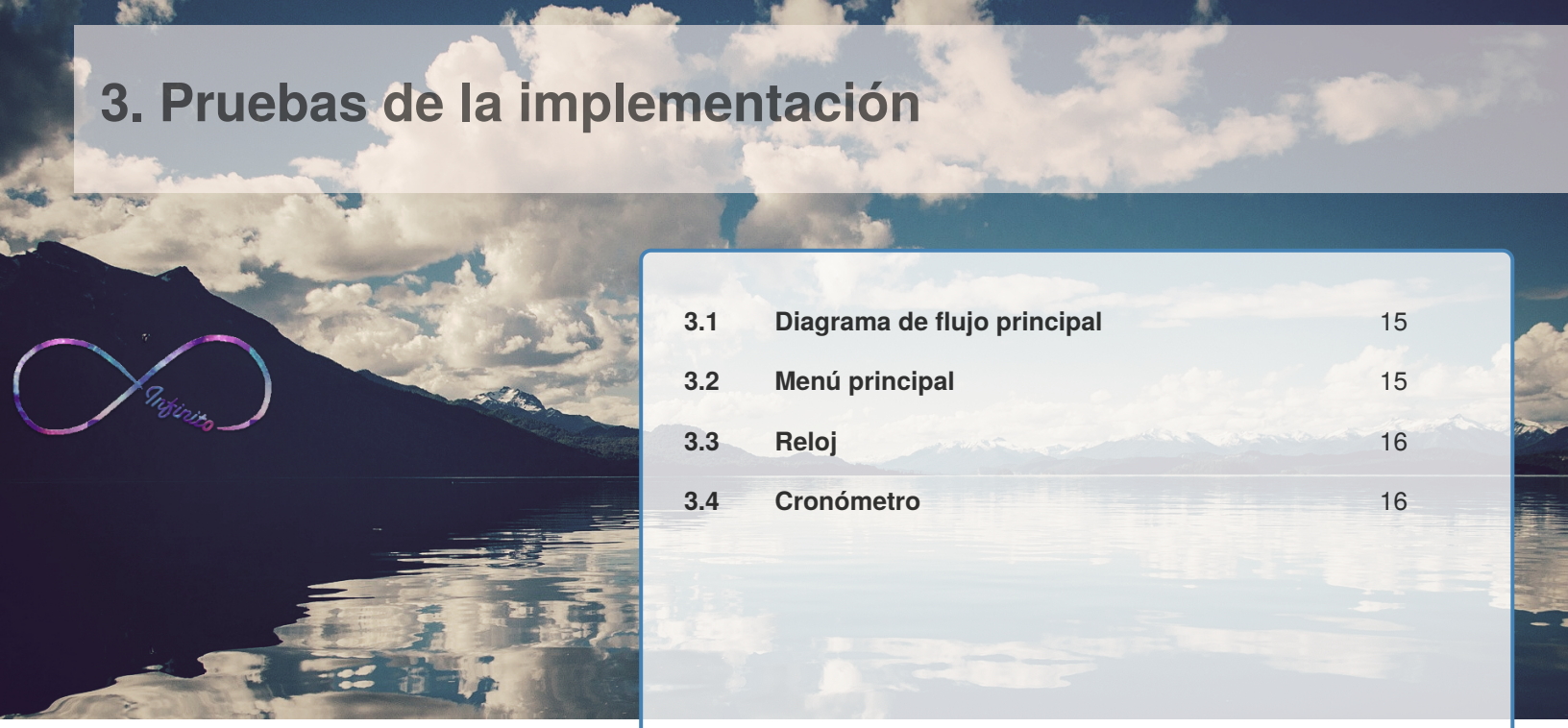
Debido a la naturaleza de saltos implementados en los botones, el menú de texto del cronómetro original se ha eliminado, y se incorporan dos botones separados para iniciar y continuar el cronómetro desde el punto en el que se dejó. Por otra parte, el algoritmo implementado en el proyecto anterior se mantuvo sin cambios notables respecto a la versión en texto, respetando los muestreos de tiempo y realizando los saltos entre etiquetas de forma idéntica para que la opción de detener y continuar desde un determinado punto no presente problema alguno.

```
cronometro:
        stopcrono:
        ;Muestreo de ticks del sistema tomado
        mov ah, 00h             ;Función 1Ah,
        int 1Ah

        mov [t_inter], dx
        mov [t_inter+2], cx
                                ;Una vez reali
        call espacios_que_no_hacen_nada_2


        restartcrono:
        xor al, al

        mov ah, 01h             ;Función 1Ah,
        mov dx,  [t_inter]
        mov cx, [t_inter +2]
        int 1Ah                 ;Ejecución del
        jmp loopcrono           ;Salto al loop
        startcrono:
        mov ah,00h              ;Función 1Ah,
        int 1Ah
        mov [t_inicial], dx
        mov [t_inicial+2], cx
        startcrono_2:
        loopcrono:
        ;Loop de cálculo de la diferencia de t
        posiciona_cursor 8, 35
         call crono_proc            ;Llamada a
         lee_mouse
         test bx, 0001h
         jz loopcrono           ;Mientras no s
         jmp metodo_espacio_blancos_3
         metodo_espacio_blancos_3:
            call espacios_que_no_hacen_nada_2

         jmp loopcrono
```

Figura 2.12: *Adaptación del cronómetro a su versión gráfica.*

Finalmente, se tiene la etiqueta **teclado** que maneja el caso cuando no se detecta el soporte para el ratón y finalmente la etiqueta **salir** para limpiar la pantalla y terminar la ejecución del programa.

3. Pruebas de la implementación



En este capítulo se aborda el diagrama de flujo de cada una de las secciones del programa, así como las pruebas de escritorio realizadas para verificar el correcto funcionamiento de la solución propuesta.

3.1. Diagrama de flujo principal

El diagrama de flujo que representa el curso de todo el programa se adjunta como documento en un PDF anexo a la entrega en *Classroom* del presente proyecto.

A continuación, se mostrarán las pruebas de escritorio del programa utilizando **DOSBox** por medio de capturas de pantalla. Así mismo, se anexará al documento de texto plano un enlace de *Google Drive* con una prueba de escritorio grabada.

3.2. Menú principal

Al correr el programa utilizando el ejecutable **.exe** en **DOSBox** se ingresa a la sección principal del programa con un menú principal de botones en la parte izquierda del mismo, como se muestra a continuación:

Figura 3.1: *Menú principal del programa.*

3.3. Reloj

Al presionar el botón **INICIO**, se accede a la opción del reloj, la cual muestra la hora actual en el sistema recuperada e impresa contínuamente. Para salir de esta opción, simplemente se presiona cualquier otro botón del menú de opciones.

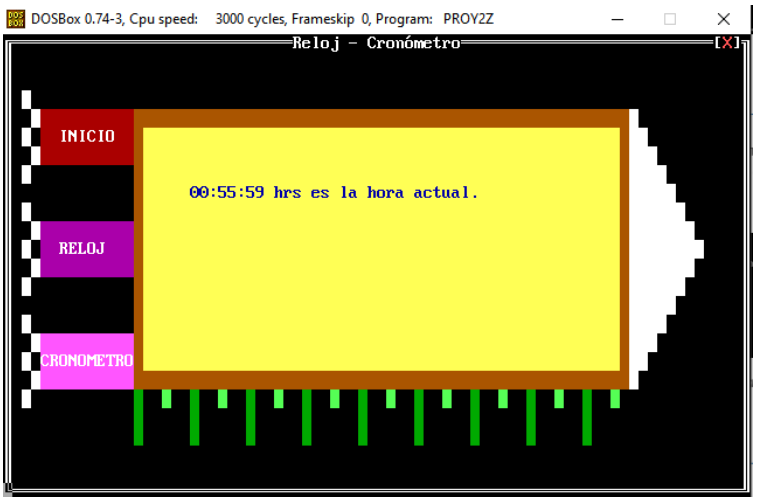


Figura 3.2: *Reloj del programa.*

3.4. Cronómetro

Al seleccionar la opción del cronómetro presionando **CRONOMETRO** se accede a una sección que contiene el submenú de uso para el cronómetro en formas de botones. Se puede

salir al menú principal si se selecciona cualquier otra opción del menú de opciones principal.

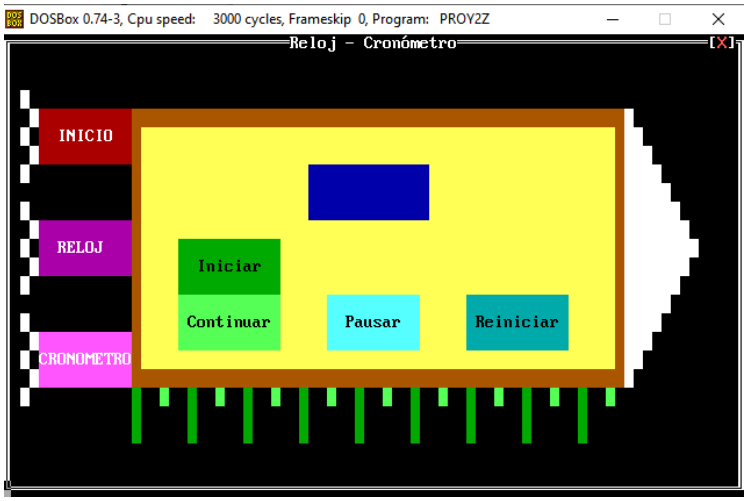


Figura 3.3: *Opción del cronómetro con el submenú de opciones.*

Si se presiona **Inicio** el cronómetro aparece en pantalla iniciando desde $00:00.000$, y comenzará a incrementarse hasta que se presione **Pausar** o se reinicie con **Reiniciar**. Presionar **Iniciar** reiniciará el cronómetro debido a la naturaleza del algoritmo implementado originalmente. Presionar cualquier otro botón del menú principal de opciones hará que el cronómetro termine su ejecución y el programa regrese al menú principal.

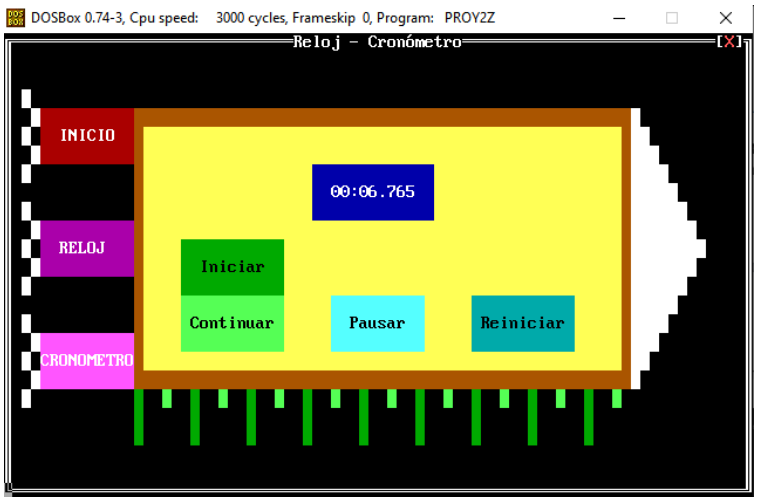


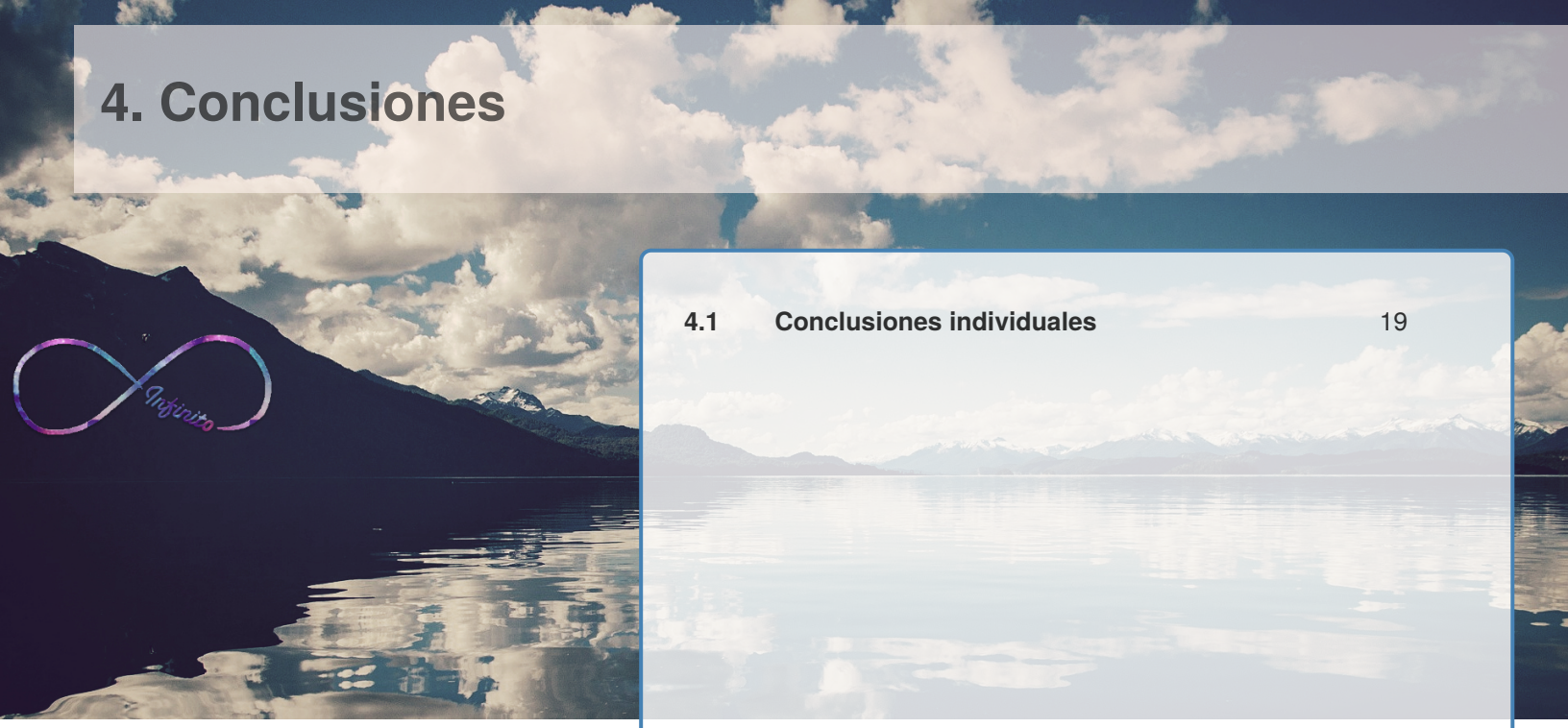
Figura 3.4: *Cronómetro ejecutándose.*

Si el cronómetro se detiene con **Pausar**, la impresión quedará congelada en el último estado. En este punto puede optarse por presionar **Reiniciar** para reiniciar el cronómetro o presionar **Continuar** para que el cronómetro continúe exactamente desde el punto en donde se quedó antes de presionar **Pausar**.

No se aconseja utilizar el botón **Continuar** más que cuando se utiliza previamente **Pau-**

sar, debido a que al presionarlo sin haber antes presionado **Iniciar** causará que el cronómetro no inicie desde 0. Como el caso anterior, esto se debe a la naturaleza del algoritmo implementado en el proyecto anterior.

4. Conclusiones



Este último capítulo aborda las conclusiones de cada uno de los integrantes del equipo con respecto al trabajo desarrollado en el proyecto.

4.1. Conclusiones individuales

- **Esquivel Razo Israel**: El correspondiente proyecto, me permitió entender la funcionalidad y la implementación del ambiente gráfico en el lenguaje ensamblador, además, de utilizar el código del proyecto pasado (“Reloj y Cronómetro”), para reforzar los conocimientos adquiridos al realizar estas funciones.

  Al desarrollar nuestro programa sin una interfaz de consola, logramos incluir, identificar y procesar el cursor del mouse con el click, aparentando el uso de botones en nuestro programa, esta funcionalidad, me parece una alternativa excelente al momento de programar, ya que existen diversos métodos para un mismo fin. Por otro lado, implementar y colocar el cursor para una correcta salida en el programa, fue de mucha utilidad al momento de dibujar o escribir en la interfaz.

  El proyecto me permitió reforzar e implementar los conocimientos adquiridos en el curso, por lo cual, se puede decir que se cumplieron los objetivos del proyecto

- **Téllez González Jorge Luis**: Por medio del trabajo desarrollado me ha sido posible elevar mi comprensión en el desarrollo de operaciones complejas en el lenguaje ensamblador y consolidar los conocimientos adquiridos durante el desarrollo del curso

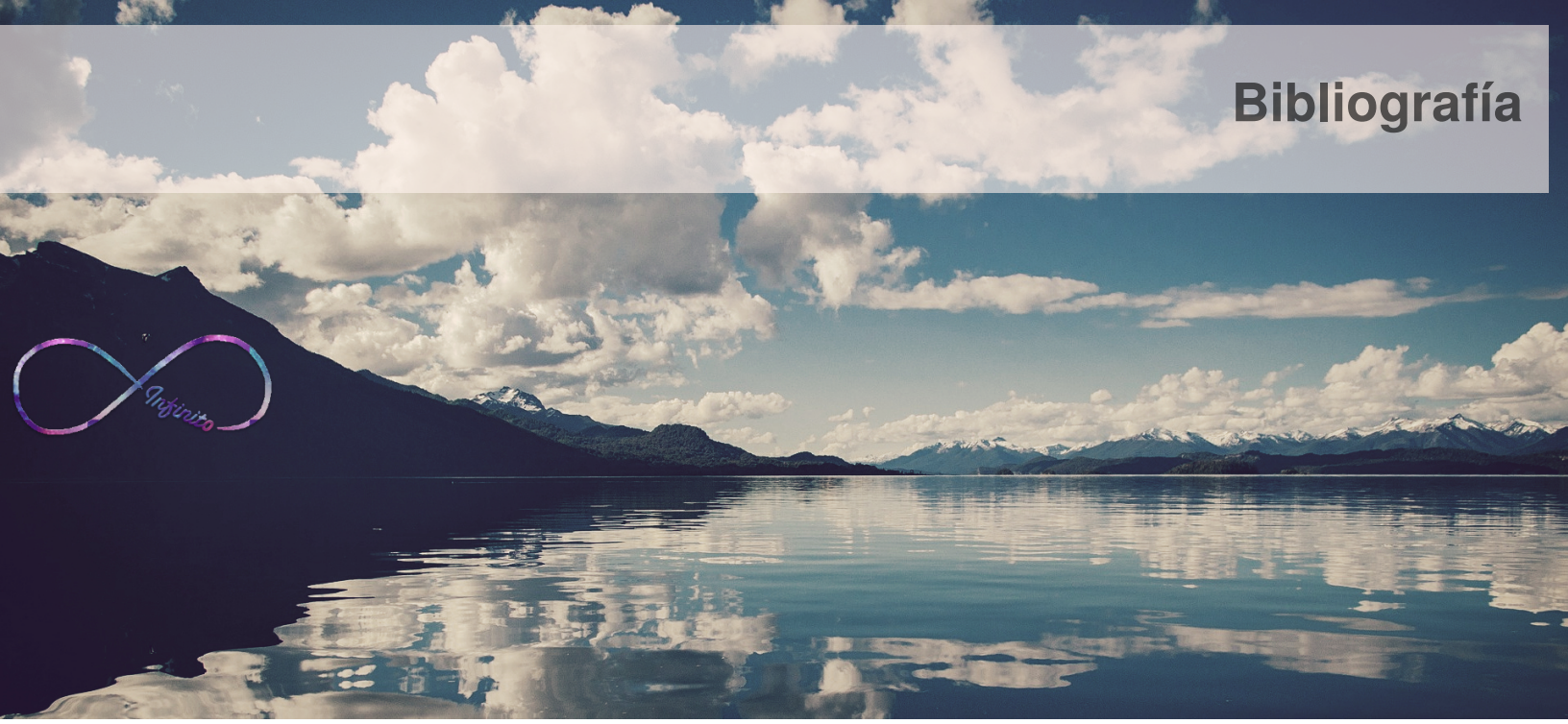
por medio de un proyecto con una propuesta de trabajo que requirió una importante tarea de prueba y error para la creación de elementos que nos pudiesen permitir posteriormente llevar a cabo la generación de la interfaz gráfica de forma sencilla.

De forma general, quiero mencionar los siguientes puntos más relevantes de todo el trabajo que estuve desarrollando:

- La mayor parte del código desarrollado en el proyecto anterior no presentó demasiadas complicaciones para adaptarlo a una versión gráfica, a excepción de la lógica de flujos implementada en el cronómetro que resultó más liosa de implementar y forzó a separar el botón de Inicio original en dos botones; uno para iniciar el cronómetro y otro para retomar su curso cuando este se detuviese. Más allá de ese aspecto, la conversión resultó exitosa a plenitud.
- Como mencioné previamente, la creación de herramientas para dibujar la IU fue una tarea árdua que requirió principalmente experimentación así como el análisis del código base proporcionado con el fin de aprovecharlo para escribir macros o procedimientos de utilidad.
- El enfoque elegido de bucles resultó facilitar enormemente la tarea de implementar la interfaz gráfica a diferencia de lo que pudiese haber ocurrido si se hubiese optado por un enfoque utilizando vectores de interrupción; como se había planteado en una fase temprana de desarrollo en el primer proyecto.

Gracias a toda esta experiencia, es posible concluir que el uso de interrupciones relacionadas al uso de los servicios de impresión y el ratón representan una base fundamental para comprender el funcionamiento de las aplicaciones que pueden encontrarse actualmente en cualquier computadora basada en x86-64, pues el fundamento base de todos esos programas reside en estos servicios manipulables por medio del lenguaje ensamblador. Finalmente, considero que este proyecto permitió elevar a un mayor nivel mi comprensión del lenguaje ensamblador y sin duda representará una experiencia notable para las asignaturas posteriores como *Sistemas Operativos*. Por todo esto, considero que el objetivo fundamental del proyecto se ha cumplido con éxito al haberse logrado la implementación de la interfaz gráfica funcional y que cumple con los requisitos solicitados.

[1, 2].



Bibliografía

[1] *INT 10H Función 00H*. Recuperado de: `http://ict.udlap.mx/people/oleg/docencia/Assembler/asm_interrup_10.html`. Fecha de consulta: 27/01/2021.

[2] *INT 33H: Mouse Support*. Recuperado de: `https://dos4gw.org/INT_33H_Mouse_Support`. Fecha de consulta: 27/01/2021.