



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Estructura y Programación de Computadoras

Grupo: 02 - Semestre: 2021-1

Proyecto 1: Reloj-Cronómetro

FECHA DE ENTREGA: 18/12/2020

Alumnos:
Esquivel Razo Israel
Téllez González Jorge Luis



1	Introducción	5
1.1	Funcionalidades del programa	5
1.2	Indicaciones adicionales	6
2	Desarrollo	7
2.1	Aproximación inicial	7
2.2	Interrupciones empleadas	9
2.3	Construcción del reloj	9
2.4	Construcción del cronómetro	11
3	Pruebas de la implementación	17
3.1	Diagrama de flujo principal	17
3.2	Menú principal	17

3.3	Reloj	18
3.4	Cronómetro	18
4	Conclusiones	20
4.1	Conclusiones individuales	20
	Bibliografía	21

1. Introducción

1.1	Funcionalidades del programa	5
1.2	Indicaciones adicionales	6

En el capítulo inicial se introduce al problema planteado por el proyecto y los aspectos que deben de ser considerados al momento de plantear una solución.

1.1. Funcionalidades del programa

El problema planteado por el proyecto es el siguiente: *Desarrollar un programa en lenguaje ensamblador para la arquitectura Intel x86 que tenga las funciones de reloj y de cronómetro, imprimiendo en pantalla las diferentes opciones a través de menús que solicitan entrada por teclado. 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 que solicite al usuario presionar una tecla para seleccionar 3 opciones diferentes:

- **Reloj**
- **Cronómetro**
- **Salir**

Si el usuario selecciona la opción **Reloj**, el programa deberá mostrar la hora del sistema, en formato HH:MM:SS (HH-horas, MM-minutos, SS-segundos) y cada segundo se debe actualizar. Para regresar al menú, el usuario deberá presionar cualquier tecla.

En cambio, si el usuario 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 pueden seleccionarse por el usuario a través del teclado:

- **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 puede presionar cualquier otra tecla, y para finalizar la ejecución del programa, el usuario debe seleccionar la opción **Salir** en el menú principal.

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 cronómetro pase de 59:59.999, se reinicia el conteo.

2. Desarrollo

2.1	Aproximación inicial	7
2.2	Interrupciones empleadas	9
2.3	Construcción del reloj	9
2.4	Construcción del cronómetro	11

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 de un menú de opciones para el programa, como puede verse a continuación:

```
menu_opciones:
    ;Imprime menú de selección
    lea dx,[select]
    mov ah,09h
    int 21h
    ;ejecuta int 21h con opcion AH=09h
    ;Imprime caracteres ASCII a partir de la localidad apuntada
    ;por DX hasta encontrar el carácter '$' (fin de cadena)

    ;Imprime opciones
    ;Opción R - reloj
    lea dx,[opc_reloj]
    int 21h
    ;Opción C - cronómetro
    lea dx,[opc_crono]
    int 21h
    ;Opción S - salir
    lea dx,[opc_salir]
    int 21h
    ;nótese que no se ha modifica AH desde la interrupción anterior
    ;nótese que no se ha modifica AH desde la interrupción anterior
    ;nótese que no se ha modifica AH desde la interrupción anterior
```

Figura 2.1: Fragmento del código base con un menú implementado.

Utilizando este código como punto inicial de partida, se procedió a realizar modificaciones, como el uso de interrupciones para la limpieza de pantalla y operaciones **xor** para la limpieza inicial de registros así como el uso de nuevas etiquetas para que el menú dirigiese a ellas presionando las teclas especificadas por los requerimientos del programa.

Por ejemplo, si se presiona la tecla **R/r**, el programa dirigirá a la etiqueta *et_reloj*, la cual contendrá la sección de código correspondiente al reloj a implementar. Y si la tecla seleccionada no coincide con ninguna de las opciones especificadas, el programa no realiza ninguna acción hasta que se introduzca una opción válida. Para lo anterior se hace uso de la instrucción **cmp** para realizar una comparación del valor leído del teclado con el *scancode* almacenado en *AL*.

```
lee_opcion:
;lectura de las opciones del teclado.
    mov ah,01h
    int 21h                ;int 21h opcion AH=01h. Lectura de teclado CON ECO.

    cmp al,'R'             ;compara si la tecla presionada fue 'R'
    je et_reloj            ;Si tecla es 'R', salta a etiqueta et_reloj
    cmp al,'r'             ;compara si la tecla presionada fue 'r'
    je et_reloj            ;Si tecla es 'r', salta a etiqueta et_reloj

    cmp al,'C'             ;compara si la tecla presionada fue 'C'
    je et_cronometro       ;Si tecla es 'C', salta a etiqueta et_cronometro
    cmp al,'c'             ;compara si la tecla presionada fue 'c'
    je et_cronometro       ;Si tecla es 'c', salta a etiqueta et_cronometro

    cmp al,'S'             ;compara si la tecla presionada fue 'S'
    je salir               ;Si tecla es 's', salta a etiqueta salir
    cmp al,'s'             ;compara si la tecla presionada fue 's'
    je salir               ;Si tecla es 's', salta a etiqueta salir

    jmp menu_opciones      ;Si ninguna de las teclas coincidió, regresa a menu_opciones.
```

Figura 2.2: Opciones del menú con las etiquetas de salto condicional.

De forma similar a este menú, al entrar a la opción del cronómetro se introduce un menú secundario que muestre las opciones disponibles para el cronómetro, de acuerdo a las especificaciones solicitadas en el proyecto.

```
lea dx,[selectcrono]      ;Imprime "Menú de opciones del cronómetro"
mov ah,09h
int 21h                  ;ejecuta int 21h con opcion AH=09h
;Imprime caracteres ASCII a partir de la localidad apuntada
;por DX hasta encontrar el carácter '$' (fin de cadena)

;Imprime opciones
;Opción I - Iniciar/Continuar
lea dx,[opc_ini]          ;Imprime "(I) Iniciar/Continuar"
int 21h                  ;nótese que no se ha modifica AH desde la interrupción anterior

;Opción D - Detener
lea dx,[opc_det]          ;Imprime "(D) Detener"
int 21h                  ;nótese que no se ha modifica AH desde la interrupción anterior

;Opción R - Reiniciar
```

Figura 2.3: Fragmento de las opciones implementadas para el submenú del cronómetro.

2.2. Interrupciones empleadas

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

- **int 21h:** Esta interrupción provee de servicios del DOS y la BIOS para distintas funcionalidades dentro del programa.
 - Opción *AH = 01h*: Permite la lectura del teclado con eco de salida.
 - Opción *AH = 02h*: Se envía el carácter depositado en el registro DL al dispositivo estándar de salida.
 - Opción *AH = 09h*: Permite imprimir caracteres ASCII a partir de una localidad apuntada por **BX**. Se acompaña uso con la instrucción **LEA**.
 - Opción *AH = 2Ch*: Permite obtener la hora del sistema almacenada en CX:DX en formato CH-Horas, CL-Minutos, DH-Segundos y DL-Milisegundos.
- **int 16h:** Permite controlar los servicios del teclado de la computadora.
 - Opción *AH = 00h*: Lee una pulsación de tecla.
 - Opción *AH = 01h*: Obtiene el estado del buffer del teclado.
- **int 1Ah:** Provee servicios enfocados en el controlador de tiempo real del CMOS de la computadora.
 - *AH = 00h*: Utilizada para obtener el contador de ticks del sistema. El contador se incrementa en su valor cada 55ms y permite obtener una medición relativa entre tiempos de ejecución de operaciones. Estos valores se almacenan en **DX** y **CX**.
 - *AH = 01h*: Permite configurar el contador de ticks del sistema manualmente, utilizando **DX** y **CX** para introducir los ticks deseados.

2.3. Construcción del reloj

Para la construcción del reloj, se parte de la idea básica de obtener la hora del sistema e imprimirla en pantalla. Esto puede realizarse utilizando la interrupción 21h/Op. 2Ch para obtener la hora del sistema, convertir esos datos "crudos" en ASCII y finalmente, almacenarlos en una cadena *búfer* a la que apunta **BX** utilizando **lea** que se imprime en pantalla. Para facilitar esto, se utilizan operaciones en el stack y un procedimiento aparte para convertir los datos a ASCII.


```

PUSH AX                ;Inicio de operaciones con la pila.
PUSH CX

MOV AH, 2Ch            ; Se obtiene la hora del sistema
INT 21h

MOV AL, CH             ; AL=CH, CH=Horas
CALL CONVERT           ; Llamada al procedimiento CONVERT
MOV [BX], AX           ; [BX]=hr, [BX] apunta a hr
                        ; en la cadena TIME

MOV AL, CL             ; AL=CL, CL=Minutos
CALL CONVERT           ; Llamada al procedimiento CONVERT
MOV [BX+3], AX         ; [BX+3]=min, [BX] apunta a min
                        ; En la cadena TIME

MOV AL, DH             ; set AL=DH, DH=segundos
CALL CONVERT           ; Llamada al procedimiento CONVERT
MOV [BX+6], AX         ; [BX+6]=seg, [BX] apunta a seg
                        ; En la cadena TIME

POP CX                ;Fin de operaciones con la pila.
POP AX

```

Figura 2.4: Procedimiento para obtener la hora actual e imprimirla.

```

PUSH DX                ;Inicio de operaciones con la pila.

MOV AH, 0              ; AH=0
MOV DL, 10             ; DL=10
DIV DL                ; AX=AX/DL
OR AX, 3030H           ; Convierte los valores de la hora del sistema almacenados en AX en ASCII.

POP DX                ;Fin de operaciones con la pila.

```

Figura 2.5: Instrucciones de conversión a ASCII.

Regresando al código principal, el proceso que se realiza es el siguiente:

1. Se obtiene la primer impresión en pantalla de la hora usando el procedimiento *GET_TIME*, almacenando en **BX** la cadena *TIME* que será usada como búfer de impresión de la hora.
2. Luego, se accede a un bucle denominado *loopstart* en el cuál se realiza el procedimiento anterior continuamente hasta que se presiona cualquier tecla revisando el búfer del teclado. Lo anterior conduce a un salto de tipo **jnz** que envía a la etiqueta *limpiar*.
3. Finalmente, en la etiqueta *limpiar* se lee el teclado y se realiza un salto incondicional a la etiqueta *menu_opciones*, que representa al código del menú principal del programa.

```

lea dx,[presionar_sal]
mov ah,09h
int 21h

lea BX, TIME
call GET_TIME

loopstart:
;Loop de impresión del

mov DL, 13
mov AH, 02h
int 21h

call GET_TIME
lea DX, TIME
mov AH, 09H
int 21H

mov AH, 0Dh
int 21H

mov ah, 01h
mov cx,2607h
int 16h
jnz limpiar

loop loopstart

limpiar:
;Etiqueta de limpieza

mov ah, 00h
int 16h
jmp menu_opciones

```

Figura 2.6: Ejecución del reloj.

2.4. Construcción del cronómetro

La etiqueta *et_cronometro* contiene el código creado para implementar el cronómetro. Esta sección contiene un submenú que permite visualizar las opciones que el usuario del programa tiene a su disposición para manipular el funcionamiento del cronómetro, y funciona de forma idéntica al menú implementado en el código base.

```

lea dx,[selectcrono] ;Imprime "Menú de opciones del cronómetro"
mov ah,09h
int 21h              ;ejecuta int 21h con opcion AH=09h
                     ;Imprime caracteres ASCII a partir de la localidad apuntada
                     ;por DX hasta encontrar el carácter '$' (fin de cadena)

;Imprime opciones
;Opción I - Iniciar/Continuar
lea dx,[opc_ini]     ;Imprime "(I) Iniciar/Continuar"
int 21h              ;nótese que no se ha modifica AH desde la interrupción anterior

;Opción D - Detener
lea dx,[opc_det]     ;Imprime "(D) Detener"
int 21h              ;nótese que no se ha modifica AH desde la interrupción anterior

;Opción R - Reiniciar
lea dx,[opc_rei]     ;Imprime "(R) Reiniciar"
int 21h              ;nótese que no se ha modifica AH desde la interrupción anterior

lea dx,[presionar_sal] ;Imprime "Presione cualquier tecla para regresar al menu"
mov ah,09h
int 21h

```

Figura 2.7: Submenú de opciones del cronómetro.

Tras esto, si el usuario presiona la tecla **I**, el programa realizará un salto **je** hacia la etiqueta *startcrono*, que representa el bloque inicial de instrucciones que se ejecuta al inicializar el cronómetro por primera vez. Si es presionada cualquier otra tecla, se hace un salto incondicional hacia el menú de opciones principal del programa.

```

cmp al,'I'           ;compara si la tecla presionada fue 'I'
je startcrono        ;Si tecla es 'I', salta a etiqueta startcrono

cmp al,'i'           ;compara si la tecla presionada fue 'i'
je startcrono        ;Si tecla es 'i', salta a etiqueta startcrono

jmp menu_opciones    ;Si ninguna de las teclas coincidió, regresa a menu_opciones.

```

Figura 2.8: Inicialización del cronómetro.

Para que el cronómetro funcione adecuadamente, en vez de utilizar la hora del sistema, se utilizan los *ticks* del sistema. Estos ticks se tratan de un contador que se incrementa internamente cada 55[ms] que transcurren en la computadora, por lo que pueden ser utilizados como una medida indirecta del tiempo transcurrido entre diferentes instantes de ejecución.

Para realizar este procedimiento, se deben de realizar 2 *muestreos* de los ticks: uno antes de iniciar un determinado bloque de instrucciones y otro después de ejecutarlo. Estos muestreos consistirán en utilizar la interrupción 1Ah/Op. 00h para recuperar el contador de ticks del sistema y almacenarlo en una variable denominada *t_inicial* desde los registros **CX** y **DX**. En sentido estricto se obtiene dos valores: una parte alta y otra baja del contador de ticks, sin embargo, para fines del proyecto se descarta la parte alta de **CX** y se utilizará la parte baja

almacenada en **DX** al invocar la interrupción.

Los valores del segundo muestreo realizado corresponderán a los ticks finales sobre los cuáles se realizará una resta con respecto a los ticks iniciales almacenados en la variable o registro *t_inicial*. Entonces, se realiza el siguiente algoritmo:

1. Se realiza una resta entre los valores finales e iniciales de ticks, enviando a los registros **AX** y **BX** los valores del primer muestreo almacenados en *t_inicial* y restándolos a los registros **DX** y **CX**, respectivamente.
2. Se descarta la parte alta de **CX** y se empieza a operar con la diferencia de ticks en la parte baja almacenada en **DX**. Para esto, se utiliza la instrucción **mov** para trasladar esta diferencia de ticks al registro **AX**.
3. Se multiplica la diferencia de ticks por 55[ms] para obtener la diferencia en milisegundos entre ambos muestreos.
4. El valor obtenido anteriormente se divide entre 1000 para obtener la cantidad de segundos y milisegundos transcurridos, guardándose en **AX** los segundos y el residuo de milisegundos en **DX**.
5. Se utiliza la instrucción **mov** para trasladar los milisegundos de **DX** a una variable llamada *milisegundos*.
6. Luego, el valor almacenados en **AX** se divide entre 60, para que al final de esta división el registro **AH** contenga el valor de los segundos transcurridos y **AL** el valor de los minutos.
7. Se utiliza nuevamente la instrucción **mov** para trasladar los segundos de **AH** a otra variable llamada *segundos*.
8. Se calcula el número de minutos transcurridos dividiendo nuevamente entre 60. Este resultado dará el número de horas, el cuál será descartado usando **xor** sobre **AH**. Hecha la operación anterior, se tomará la cantidad de minutos desde **AH** y se enviarán a una última variable denominada *minutos* usando **mov**.
9. Finalmente, se imprimen los minutos, segundos y milisegundos transcurridos entre ambos muestreos convirtiendo los valores hexadecimales obtenidos en ASCII imprimible en pantalla de forma similar al proceso realizado en el reloj.

Entonces, cuando se inicializa el cronómetro por primera vez, en la etiqueta *startcrono* se realiza el muestreo inicial del contador de ticks del sistema, el cual se guarda en *t_inicial*.

Después de esto, se prosigue a una siguiente sección de código enmarcada por la etiqueta *loopcrono*, la cuál contiene un bucle que realiza el algoritmo descrito anteriormente de forma continua hasta que se detecte una pulsación de una tecla durante la ejecución del bucle.

```
startcrono:
;Realiza el muestreo inicial

mov ah,00h
int 1Ah

mov [t_inicial], dx
mov [t_inicial+2], cx

mov DL, 13;
mov AH, 02h
int 21h

loopcrono:
;Loop de cálculo de la diferencia

mov DL, 13
mov AH, 02h
int 21h
call crono

MOV AH, 0Dh
INT 21H

mov ah, 01h
int 16h

jz loopcrono
jmp salto

loop loopcrono
```

Figura 2.9: Ejecución del cronómetro.

Cuando una pulsación de tecla es detectada por el programa, se realiza un salto hacia la etiqueta *salto*, la cuál se trata de una etiqueta de interrupción del bucle de impresión del cronómetro que recupera la pulsación insertada y, con base en eso, realizará diferentes acciones:

- Si la tecla presionada es **I/i**, se envía la etiqueta *restartcrono*. Esta tecla si se presiona durante la ejecución del bucle de impresión no realizará ninguna interrupción sobre el mismo.
- Si la tecla presionada es **R/r**, se envía la etiqueta *startcrono*, reiniciando efectivamente el cronómetro desde 0 como si se hubiese entrado al mismo por primera vez.
- Si la tecla presionada es **D/d**, se envía la etiqueta *stopcrono* que manejará el proceso

interno para detener el cronómetro y posteriormente retomarlo desde el punto en que se quedó pausado.

- Si la tecla presionada es **S/s**, se envía la etiqueta *menu_opciones*, saliendo del cronómetro y regresando al menú principal del programa. Así mismo, si la tecla pulsada tampoco coincide con ninguna de las opciones anteriores, se envía al menú de opciones principal.

```
salto:
;Etiqueta de interrupción del bucle de impresión del cronómetro al presionar cualquier

xor ax,ax           ;limpieza del búfer del teclado.
int 16h             ;lectura del teclado.

cmp al, 'I'          ;compara si la tecla presionada fue 'I'
je restartcrono      ;Si tecla es 'I', salta a etiqueta restartcrono

cmp al, 'i'          ;compara si la tecla presionada fue 'i'
je restartcrono      ;Si tecla es 'i', salta a etiqueta restartcrono

cmp al, 'R'          ;compara si la tecla presionada fue 'R'
je startcrono        ;Si tecla es 'R', salta a etiqueta startcrono

cmp al, 'r'          ;compara si la tecla presionada fue 'r'
je startcrono        ;Si tecla es 'r', salta a etiqueta startcrono

cmp al, 'D'          ;compara si la tecla presionada fue 'D'
je stopcrono         ;Si tecla es 'D', salta a etiqueta stopcrono

cmp al, 'd'          ;compara si la tecla presionada fue 'd'
je stopcrono         ;Si tecla es 'd', salta a etiqueta stopcrono

cmp al, 'S'          ;compara si la tecla presionada fue 'S'
je menu_opciones     ;Si tecla es 'S', salta a etiqueta menu_opciones

cmp al, 's'          ;compara si la tecla presionada fue 's'
je menu_opciones     ;Si tecla es 's', salta a etiqueta menu_opciones

jmp menu_opciones    ;Si ninguna de las teclas coincidió, regresa a menu_opciones.
```

Figura 2.10: Opciones de manipulación del cronómetro.

A continuación se detallará el procedimiento a seguir en caso de que se presione la tecla **D/d** durante la ejecución del cronómetro.

1. Se realizará un salto **je** hacia la etiqueta *stopcrono*, en donde se utilizará la interrupción 1Ah/Op. 00h para realiza un muestreo adicional (El cual para fines prácticos puede considerarse como un tercer muestreo) que permitirá muestrear el contador de ticks del sistema durante el momento en que se interrumpió el cronómetro y almacenarlo temporalmente en una variable llamada *t_inter*.

```
stopcrono:
;Muestreo de ticks del

mov ah, 00h
int 1Ah

mov [t_inter], dx
mov [t_inter+2], cx

jmp salto
```

Figura 2.11: Etiqueta stopcrono.

2. Posteriormente, se realiza un salto **jmp** hacia *salto*, en espera de que se presione la tecla **I/i**. Cuando esta tecla sea presionada, se realizará otro salto hacia la etiqueta **restartcrono**, la cual se encargará de tomar los valores muestreados en *stopcrono* para reestablecer el contador de ticks del sistema con esos valores haciendo uso de `int 1AH/01h` y enviando a **DX** y **CX** los valores muestreados y almacenados previamente en *t_inter*. Finalmente, se hace un salto **jmp** hacia *loopcrono*. De este forma, el muestreo inicial corresponderá al muestreo final realizado antes de que se interrumpiera el cronómetro y la impresión del cronómetro continuará desde el punto en donde se quedó hasta que sea interrumpida o reiniciada de nuevo.

```
restartcrono:
;Etiqueta utilizada pa

mov ah, 01h
mov dx, [t_inter]
mov cx, [t_inter +2]
int 1Ah

jmp loopcrono
```

Figura 2.12: Etiqueta stopcrono.

3. Pruebas de la implementación

3.1	Diagrama de flujo principal	17
3.2	Menú principal	17
3.3	Reloj	18
3.4	Cronómetro	18

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.

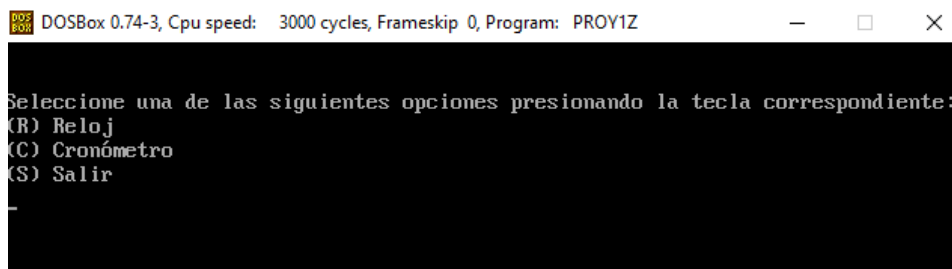
3.1. Diagrama de flujo principal

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.

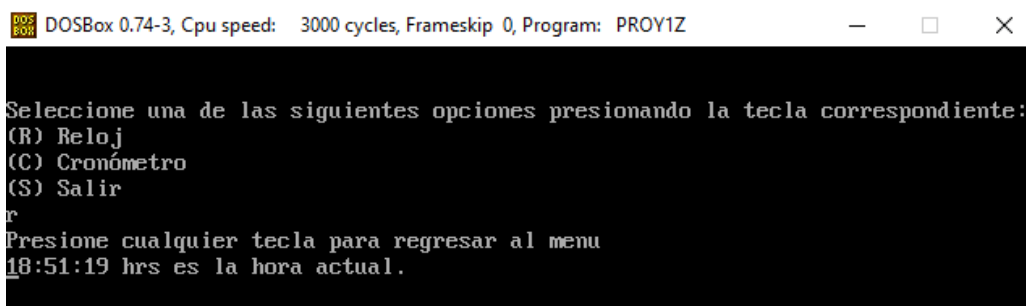
3.2. Menú principal

Al correr el programa utilizando el ejecutable **.exe** en **DOSBox** se ingresa al menú principal del programa, como se muestra a continuación.

Figura 3.1: *Menú principal del programa.*

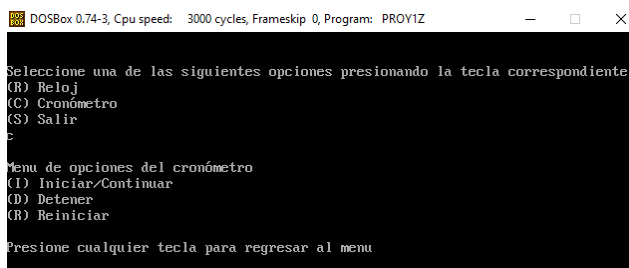
3.3. Reloj

Al presionar las teclas **R/r**, se accede a la opción del reloj, la cual muestra la hora actual en el sistema recuperada e impresa continuamente. Para salir de esta opción, simplemente se presiona cualquier tecla para regresar al menú.

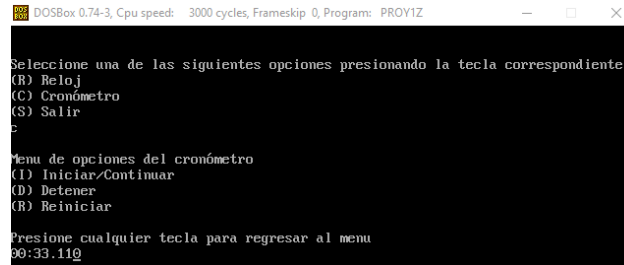
Figura 3.2: *Opción del reloj en el programa escrito.*

3.4. Cronómetro

Al seleccionar la opción del cronómetro presionando **C/c** en el menú principal se accede a un submenú que contiene las opciones de uso para el cronómetro. Se puede salir al menú principal si se selecciona cualquier otra tecla.

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

Si se presiona **I/i** el cronómetro aparece en pantalla iniciando desde 00 : 00.000, y comenzará a incrementarse hasta que se presione **D/d** o se reinicie con **R/r**. Presionar **I/i** no realizará ninguna acción sobre el cronómetro y presionar cualquier otra tecla hará que el cronómetro termine su ejecución y el programa regrese al menú principal.

A screenshot of a DOSBox window titled "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip: 0, Program: PROY1Z". The window displays a text-based menu on a black background with white text. The menu options are: (R) Reloj, (C) Cronómetro, (S) Salir, and a blank line. Below these is a section titled "Menu de opciones del cronómetro" with options: (I) Iniciar/Continuar, (D) Detener, and (R) Reiniciar. At the bottom, it says "Presione cualquier tecla para regresar al menu" and shows a timer "00:33.110".

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip: 0, Program: PROY1Z
Seleccione una de las siguientes opciones presionando la tecla correspondiente:
(R) Reloj
(C) Cronómetro
(S) Salir
C
Menu de opciones del cronómetro
(I) Iniciar/Continuar
(D) Detener
(R) Reiniciar
Presione cualquier tecla para regresar al menu
00:33.110
```

Figura 3.4: *Cronómetro ejecutándose en el programa.*

Si el cronómetro se detiene con **D/d**, la impresión quedará congelada en el último estado. En este punto puede optarse por presionar **R/r** para reiniciar el cronómetro o presionar **I/i** para que el cronómetro continúe exactamente desde el punto en donde se quedó antes de presionar **D/d**.

4. Conclusiones

4.1 Conclusiones individuales

20

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:** En el correspondiente proyecto, implemente lo adquirido en el curso del lenguaje ensamblador, es decir, utilicé diferentes tipos de datos, conjuntos de instrucciones y nuevas funciones para la obtención del reloj interno y los ticks en el sistema, con el fin de desarrollar e implementar diferentes opciones en el modelado del reloj y del cronómetro. A medida que iba desarrollando el proyecto, observé que existen diferentes rutas para un mismo objetivo, lo cual, me permitió visualizar de diferente forma las rutas y mapeo en el lenguaje ensamblador.

El cronómetro fue el objetivo más difícil de implementar, ya que el manejo de los ticks no fue tan simple, sin embargo, una vez que se entendió el funcionamiento del mismo, podríamos realizar diversas operaciones, cabe resaltar, el uso y limpieza del buffer, el cual, guardaba de manera temporal un dato para un proceso siguiente.

El proyecto realizado, me permitió entender y observar el funcionamiento de diversas instrucciones dentro del programa, con lo cual, practiqué y reforcé lo conocimientos adquiridos en la parte teórica, además de investigar nuevas instrucciones para la realización del proyecto.

- **Téllez González Jorge Luis:** A raíz del trabajo desarrollado en el proyecto ha sido posible combinar los conocimientos adquiridos durante el curso de **EyPC** para llevarlos a

la práctica en un programa que ha resultado ser una experiencia distinta y muy retadora frente a lo que se estuvo trabajando previamente. Para lograr llevar a cabo el programa planteado, se requirió combinar el uso de múltiples interrupciones e instrucciones en el desarrollo del programa, así como establecer algoritmos definidos para cada uno de los procedimientos utilizados.

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

- El programa correspondiente al reloj logró ser desarrollado en relativamente poco tiempo, debido a que existían ejemplos similares [2], que abordaban el problema planteado y que fueron valiosas referencias para el camino a seguir. En este caso, se utilizaron bucles a diferencia de la fuente consultada que hacía uso de vectores de interrupción personalizados para el mismo fin.
- Por otra parte, el desarrollo del cronómetro resultó mucho más complicado y requirió asistencia adicional por medio del contacto por **WhatsApp** que finalmente logró llevar a que el programa funcionase correctamente al momento de detener el cronómetro y tratar de reiniciar la cuenta desde el punto en el que se quedó.
- Durante el desarrollo se probaron 2 enfoques distintos: el uso repetido de bucles y vectores de interrupción, sin embargo, debido a las dificultades asociadas al uso de interrupciones personalizadas y su naturaleza, se optó por desechar tal enfoque y concentrarse en realizar un programa basado enteramente en bucles y diversos tipos de saltos.

Gracias a toda esta experiencia, es posible concluir que el uso de interrupciones como **16h o 1Ah** resultan ser indispensables para actividades que pudieran parecer no estar relacionadas a los servicios de ejecución de una computadora (como el caso de los ticks y el cronómetro). Así mismo, pudo verificarse la importancia de establecer adecuadamente los flujos de salto con el fin de que el programa realice las operaciones correctamente, ya que una de las mayores dificultades a vencer estuvo precisamente en utilizar adecuadamente los diversos tipos de saltos para modificar el flujo del programa de acuerdo a las necesidades requeridas. La experiencia obtenida con este proyecto sin lugar a dudas resultará muy importante para las próximas actividades a desarrollar durante lo que resta del curso.

[1, 2, 5, 3, 4, 6].



- [1] *DOS INT 21h - DOS Function Codes*. Recuperado de: <http://spike.scu.edu.au/~barry/interrupts.html>. Fecha de consulta: 13/12/2020.
- [2] *DOS Interrupts*. Recuperado de: <https://faculty.kfupm.edu.sa/COE/aimane/assembly/pagegen-142.aspx.htm>. Fecha de consulta: 13/12/2020.
- [3] *Int 1Ah*. Recuperado de: https://vitaly_filatov.tripod.com/ng/asm/asm_029.html. Fecha de consulta: 13/12/2020.
- [4] *Int 1Ah*. Recuperado de: https://dos4gw.org/INT_1aH_00H_Read_System_Clock_Ticks. Fecha de consulta: 13/12/2020.
- [5] Medina, Ramón. *Programación Avanzada en Lenguaje Ensamblador*. Recuperado de: <http://index-of.co.uk/Winasm-studio-tutorial/ac0001.PDF>. Fecha de consulta: 13/12/2020.
- [6] *Recursos del Sistema: Servicios del DOS y de la BIOS*. Recuperado de: http://arantxa.ii.uam.es/~gdrivera/labetcii/int_dos.htm. Fecha de consulta: 13/12/2020.