UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1

SECCIÓN “N”

VACACIONES DEL PRIMER SEMESTRE 2023

MANUAL TÉCNICO

PRÁCTICA 2

Diego Andrés Huite Alvarez

202003585

22/06/2023

Índice

INTRODUCCIÓN

Bienvenido al manual técnico de la práctica 2 de arqui1, un sistema nuevo de un puesto de ventas en el cual usted tendrá el control de productos, ventas y podrá ver reportes de las acciones que ha hecho.

REQUERIMIENTOS

Software:

- Sistema operativo windows, linux o macOS
- DOSbox instalado
- MASM611

Hardware:

- Mouse
- Teclado
- Monitor
- 2gb de ram

PROGRAMA

Variables del programa:

```
p_html              db      "<p>"
pc_html             db      "</p>"
letraMin            db      0000
letraMay            db      0000

;CLONES
   CloncodigoProducto db 04 dup(0)
   ClondescProducto db 20 dup(0)
   ClonproductPrice db 05 dup(0)
   ClonproductUnits db 05 dup(0)
   cerosProd db 2E dup(0)
   numero     db    05 dup (30)
   numeroInvert     db    05 dup (0000)

   ;mensajes ----------------------------------------------------------------------------------------------
   mensajeBienvenida db "-----------------------------------------------",0A,"Universidad de San Carlos de Guatemala" ,0A , "Facultad de ingenieria", 0A,"Escuela de Vacaciones",0A,"Arquitectura de Computadores
   userMenu db "Ingrese el caracter entre parentesis",0A, "--------MENU PRINCIPAL--------" ,0A,"(p)roductos",0A, "(v)entas",0A, "(h)erramientas", 0A,  "$"
   productMenu db "Ingrese el indice",0A, "--------MENU PRODUCTOS--------" ,0A,"1.Ingreso",0A, "2.Eliminacion",0A, "3.Visualizacion", 0A, "$"
   toolsMenu db "Ingrese el indice",0A, "--------MENU HERRAMIENTAS--------" ,0A,"1.Catalogo",0A, "2.Reporte alfabetico",0A, "3.Reporte ventas", 0A, "4.Productos sin existencias", 0A, "$"
   salesTitle db  "-------Ventas--------" , "$"
   bienvenido db "BIENVENIDO: ","$"
   credentialFileConfirmation db "EXISTE EL ARCHIVO",0A,"$"
   credentialFileNotExistent  db "CREDENCIALES NO ENCONTRADAS","$"
   lexicalError  db "ERROR EN ARCHIVO","$"
   continueFiveProducts db "Ingrese enter si quiere ver otros 5 productos o 'q' si desea salir", "$"
```

Todo lo que esté adentro de la directiva .data serán mensajes, plantillas html, y las estructuras que se usan para el correcto funcionamiento del programa, tales como la estructura de un producto, de una venta, etc.

Archivo de credenciales:

Para el correcto funcionamiento del programa, se necesita un archivo de credenciales que se llama “PRAII.CON” este archivo contendrá credenciales con la siguiente estructura:

```
[credenciales]
usuario    =    "dalvarez"
clave      =    "202003585"
```

Para poder leer este archivo en masm se abre con la interrupción 21, subcódigo ah 3d

```
; ABRE EL ARCHIVO PRAII.CON
mov AL, 02
mov AH, 3d
mov DX, offset pathcredentialFile
int 21
;------------------------------
; si no lo logra abrir, entonces imprimimos mensaje de fallo
jc  credentialFileFailed ; jc - jump if condition met, es decir, salta si se abrio el archivo
; si lo logra abrir mostramos menu inicial
mov [handlecredentialFile] , AX

jmp credentialFileSuccess

credentialFileFailed:
    printString newLine
    printString credentialFileNotExistent
    jmp exit

credentialFileSuccess:
    printString newLine
    printString credentialFileConfirmation


readCredentialFile:
    mov BX, [handlecredentialFile]
```

Cabe recalcar que verificamos la existencia de este para luego poder leerlo, con la interrupción 21-3F, guardamos la información en un buffer y se comparan carácter a carácter la estructura de este archivo, el programa es capaz de ignorar espacios en blanco entre los “=”, y guarda los índices de las comillas donde está la información para luego comparar si esta es correcta.

El archivo utils.asm:

Este archivo trabaja en conjunto con el main.asm, contiene macros que ahorran el escribir código para pedir un input por char, o para ver si un archivo existe.

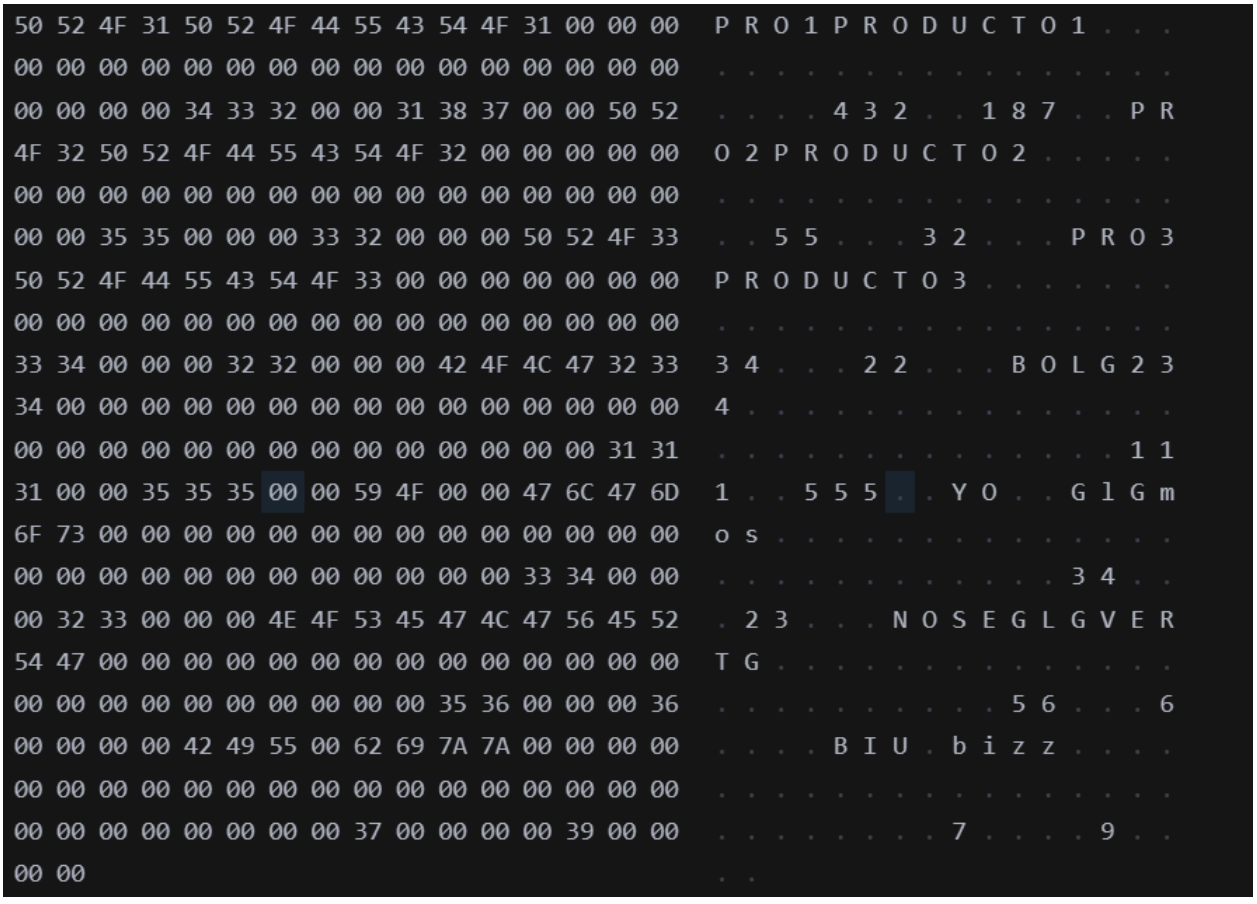
```
fileCreator macro path2
    mov CX, 0000
   mov DX, offset path2
   mov AH, 3c
   int 21h
endm

saveBufferedInput macro buffer
    mov ah, 0Ah
    mov dx, offset buffer
    int 21h
endm

saveCurrentProduct macro handle
    mov bx, [handleprodFile]
    mov cx, 28
    mov dx, offset codigoProducto
    mov ah, 40
    int 21
endm
```

Nuestras “bases de datos”

Para que el programa guarde y tenga acceso a productos, ventas y demás, existen ciertos archivos para llevar el control

PROD.BIN:

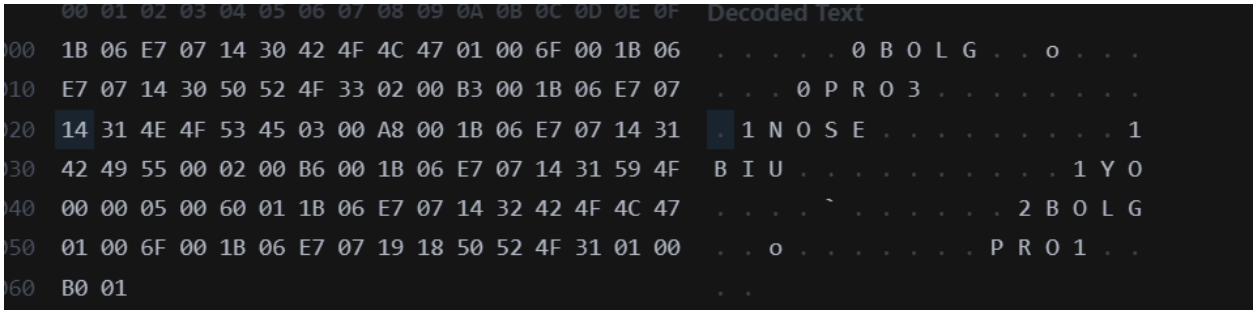
Este es un archivo binario que guarda las estructuras de productos definidas en la directiva .data



Precisamente de este archivo se extrae información sobre el costo de productos, sus códigos, descripciones, etc.

VENT.BIN:

Este archivo de igual manera guarda información, pero en este caso se trata de información sobre las ventas, como su fecha de generación, monto total, etc.



INTERRUPCIONES USADAS:

- **INT 21 - AH = 3Dh DOS 2+ - OPEN DISK FILE WITH HANDLE:**

  Esta interrupción es de suma importancia para poder abrir archivos, es utilizada al revisar la existencia de archivos como prod.bin y vent.bin. Por ejemplo cuando generamos el reporte de ventas, hacemos uso de esta interrupción para leer del archivo

```
generateSalesReport:

; abrimos el vent.bin
        mov al, 2
        mov ah, 3d
        mov dx, offset pathVentas
        int 21
        mov [handleVENTASFile], ax


        mov [puntero_temp], 0000
```

- **INT 21 - AH = 3Ch DOS 2+ - CREATE A FILE WITH HANDLE (CREAT)**

  Esta interrupción es bastante útil para cuando necesitamos crear archivos, es de uso cuando se generan reportes para el usuario, y cuando se crea por primera vez el prod.bin y vent.bin.

```
generateCatalog:
    ; abrimos el prod.bin
        mov al, 2
        mov ah, 3d
        mov dx, offset pathProductFile
        int 21
        mov [handleprodFile], ax



   ; creamos el archivo
    mov ah, 3c
    mov cx, 0
    mov dx, offset pathCatalogFile
    int 21
    mov [handleCatalogFile], ax
```

- **INT 21 - AH = 3Eh DOS 2+ - CLOSE A FILE WITH HANDLE**

  Esta interrupción es útil para cuando terminamos de escribir en un archivo. Sirve para cerrar el stream entre masm y el archivo. Es de utilidad cuando terminamos de escribir un reporte para el usuario o un archivo de nuestra pequeña base de datos

```
ContinueForLetras:
    ;cerramos el archivo.bin para luego reabrirlo luego
    mov bx, [handleprodFile]
    mov ah, 3E
    int 21
    ;continue
```

- **INT 21 - AH = 3Fh DOS 2+ - READ FROM FILE WITH HANDLE:**

  Esta interrupción es usada para la lectura de archivos, es de suma importancia para obtener información de nuestros archivos de bases de datos, o incluso el de credenciales.

```
loopReadProductData:
    cmp si, 5
    je exitloopReadProductData
    mov ah, 3F
    mov bx, [handleprodFile]
    mov cx, 2E
    mov dx, offset codigoProducto
    int 21
    cmp AX, 00
    je exitloopReadProductData
```

- **INT 21 - AH = 40h DOS 2+ - WRITE TO FILE WITH HANDLE**

Es usada para poder escribir en archivos, tales como los reportes, o en nuestras bases de datos.

```
; escribimos el producto en si
mov cx, 2E
mov dx, offset codigoProducto
mov ah, 40
int 21
```

- **INT 21 - AH = 42h DOS 2+ - MOVE FILE READ/WRITE POINTER (LSEEK)**

  Es usada para mover el puntero del archivo, es útil cuando queremos escribir en una parte en específico del archivo, es usada para cuando queremos eliminar un producto, por ejemplo

```
delProduct:

    ; nos posicionamos en el producto a eliminar
    mov dx, [puntero_temp]
    sub dx, 2E
    mov cx, 0000
    mov bx, [handleprodFile]
    mov al, 0
    mov ah, 42
    int 21
    ; escribimos 0s en el archivo
    mov cx, 2E
    mov dx, offset cerosProd
    mov ah, 40
    int 21
    jmp closeProdFile
```