Arellano Granados Angel Mariano

218123444

Seminario de Traductores de Lenguajes I

I7026 D02

Reporte de Actividades 7

Actividad 7 – Parte 1 – 3

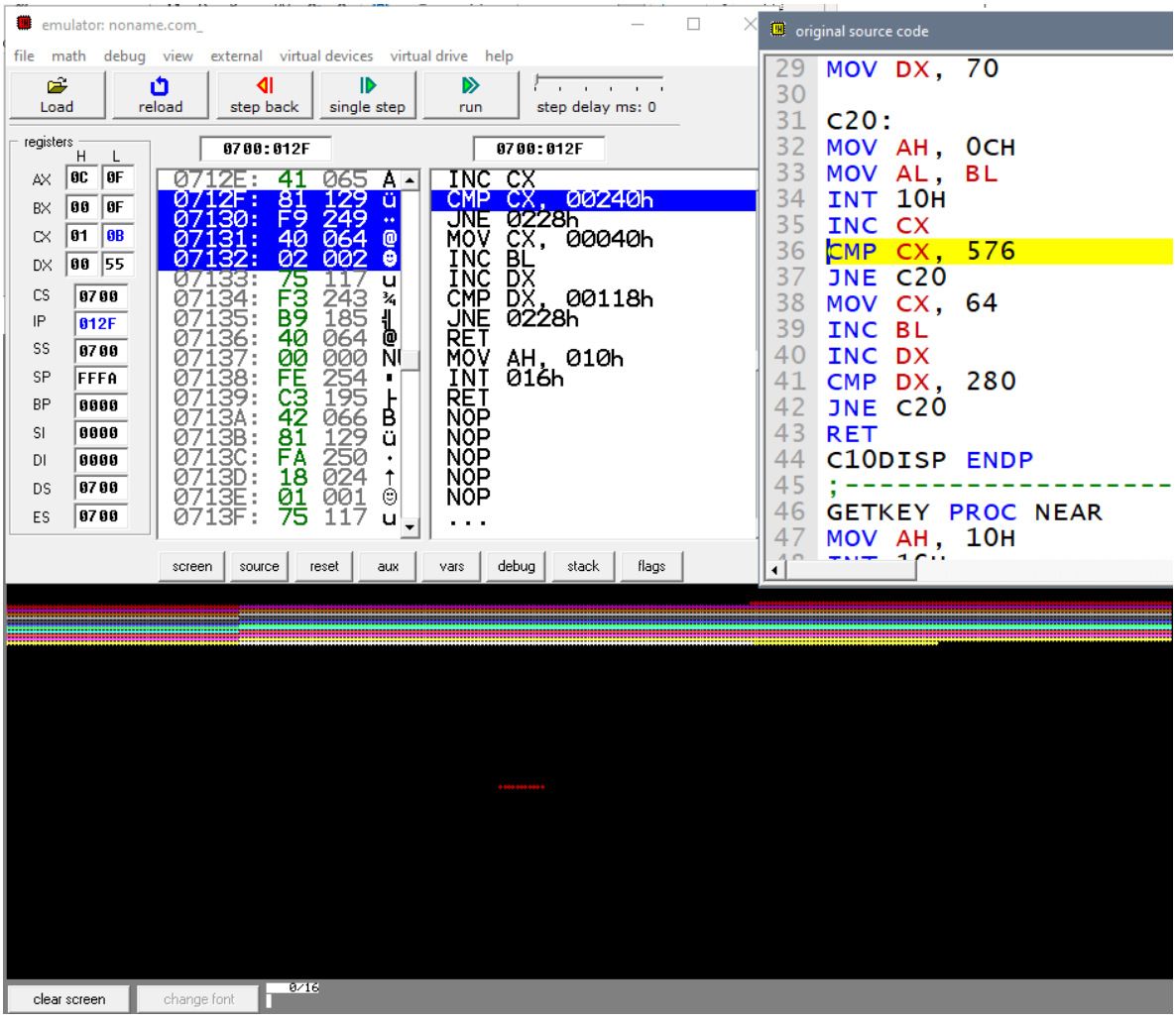
Actividad 7 – Parte 1

Descripción

Documente y ejecute el programa en ensamblador, haciendo énfasis en la instrucción 10H para el despliegue de gráficos.

Desarrollo y Resultados

El programa inicia obteniendo el actual modo de video y almacena el dato en la pila, después llama al procedimiento GRAPHMODE donde cambia el modo de video (Nota: se hizo un cambio al código eliminando la interrupción 10h con AH=0Bh que cambiaba el color del borde, porque emu8086 no la reconocía), ahora se entra al procedimiento principal C10DISP donde se inicializan los registros BX, CX y DX en 0, 64 y 70 respectivamente para con la interrupción 10h con AX=0Ch dibujar un pixel del color que indique BL en la coordenada dada por CX y DX incrementándolas para lograr tener líneas de colores diferentes apiladas unas sobre otras hasta llenar toda la consola.



El programa tarda tanto en completar el procedimiento anterior que no logre visualizar los siguientes procedimientos como GETKEY, que solicita una entrada por teclado y guarda el ASCII de la tecla en AL, y finaliza sacando de la pila el único dato guardado y cambiando una ves mas el modo de video.

Código:

```
.MODEL SMALL
.CODE
;-----------------------------------
ORG 100H
BEGIN PROC NEAR
   MOV AH, 0FH
   INT 10H
   PUSH AX
   CALL GRAPHMODE
   CALL C10DISP
   CALL GETKEY
   POP AX
   MOV AH, 00H
   INT 10H
   MOV AX, 4C00H
   INT 21H
BEGIN ENDP
;-----------------------------------
GRAPHMODE PROC NEAR
   MOV AH, 00H
   MOV AL, 13H
   INT 10H
   RET
GRAPHMODE ENDP
;-----------------------------------
C10DISP PROC NEAR
   MOV BX, 00
   MOV CX, 64
   MOV DX, 70

   C20:
   MOV AH, 0CH
   MOV AL, BL
   INT 10H
   INC CX
   CMP CX, 576
   JNE C20
   MOV CX, 64
   INC BL
```

```
    INC DX
    CMP DX, 280
    JNE C20
    RET
C10DISP ENDP
;---------------------------------
GETKEY PROC NEAR
    MOV AH, 10H
    INT 16H
    RET
GETKEY ENDP

END BEGIN
```

Reflexión

La actividad me pareció muy curiosa, pero nos muestra varias interrupciones útiles para el proyecto como dibujar pixeles en la consola para mostrar formas o rectas y el cómo aprovechar las entradas de teclado.

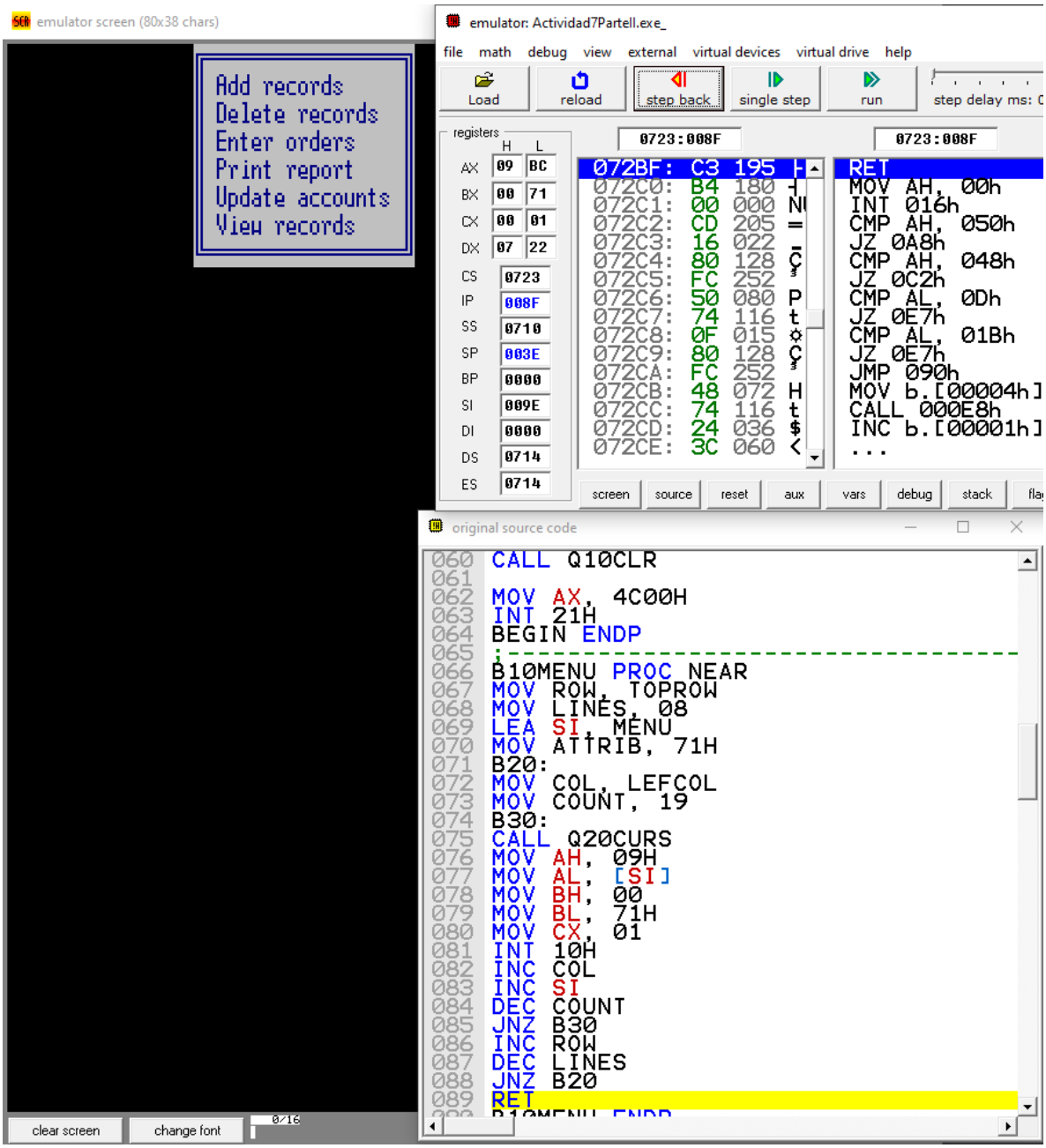
Actividad 7 – Parte 2

Descripción

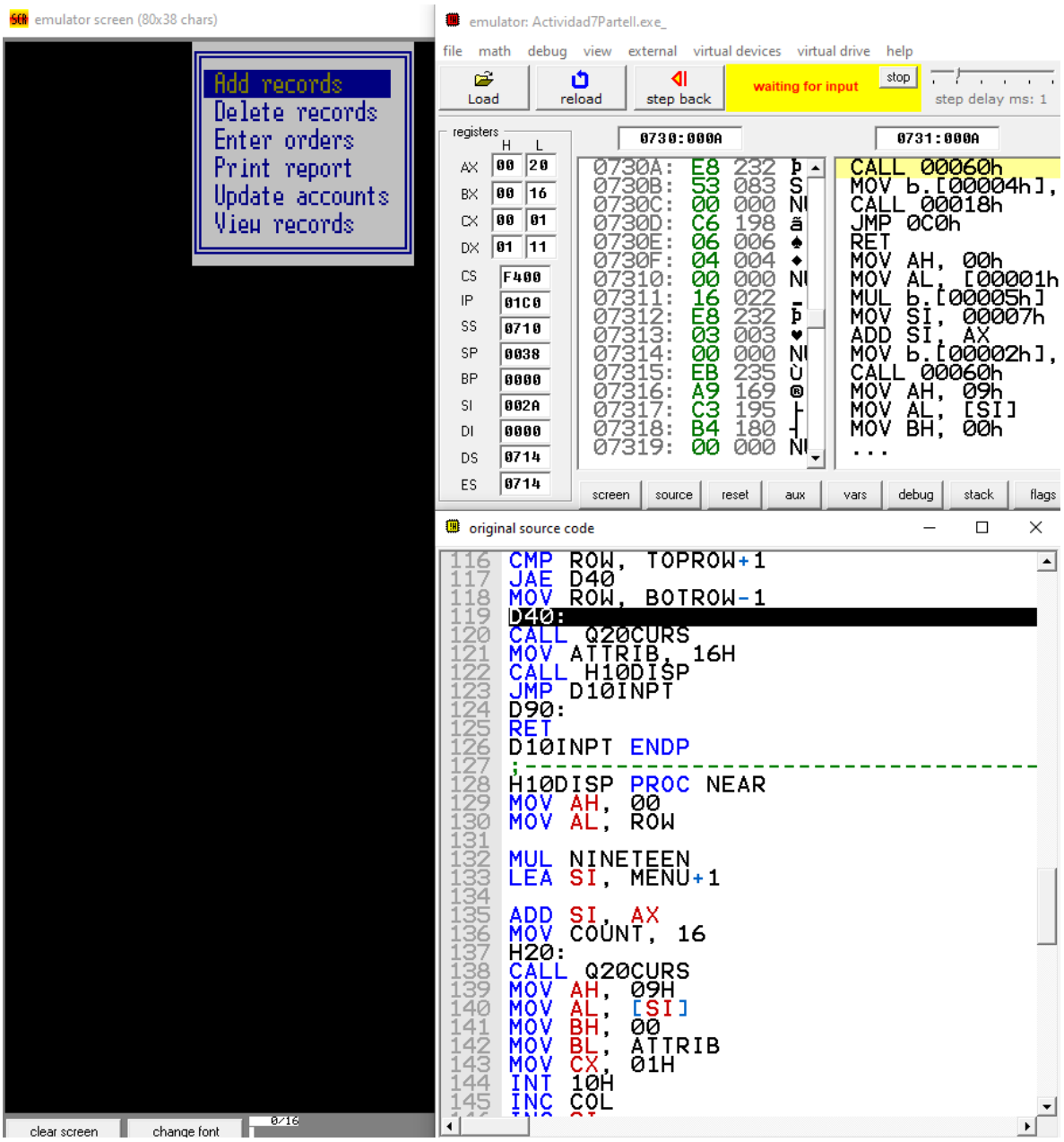
Ejecute y explique el programa en ensamblador para la creación de menús.

Desarrollo y Resultados

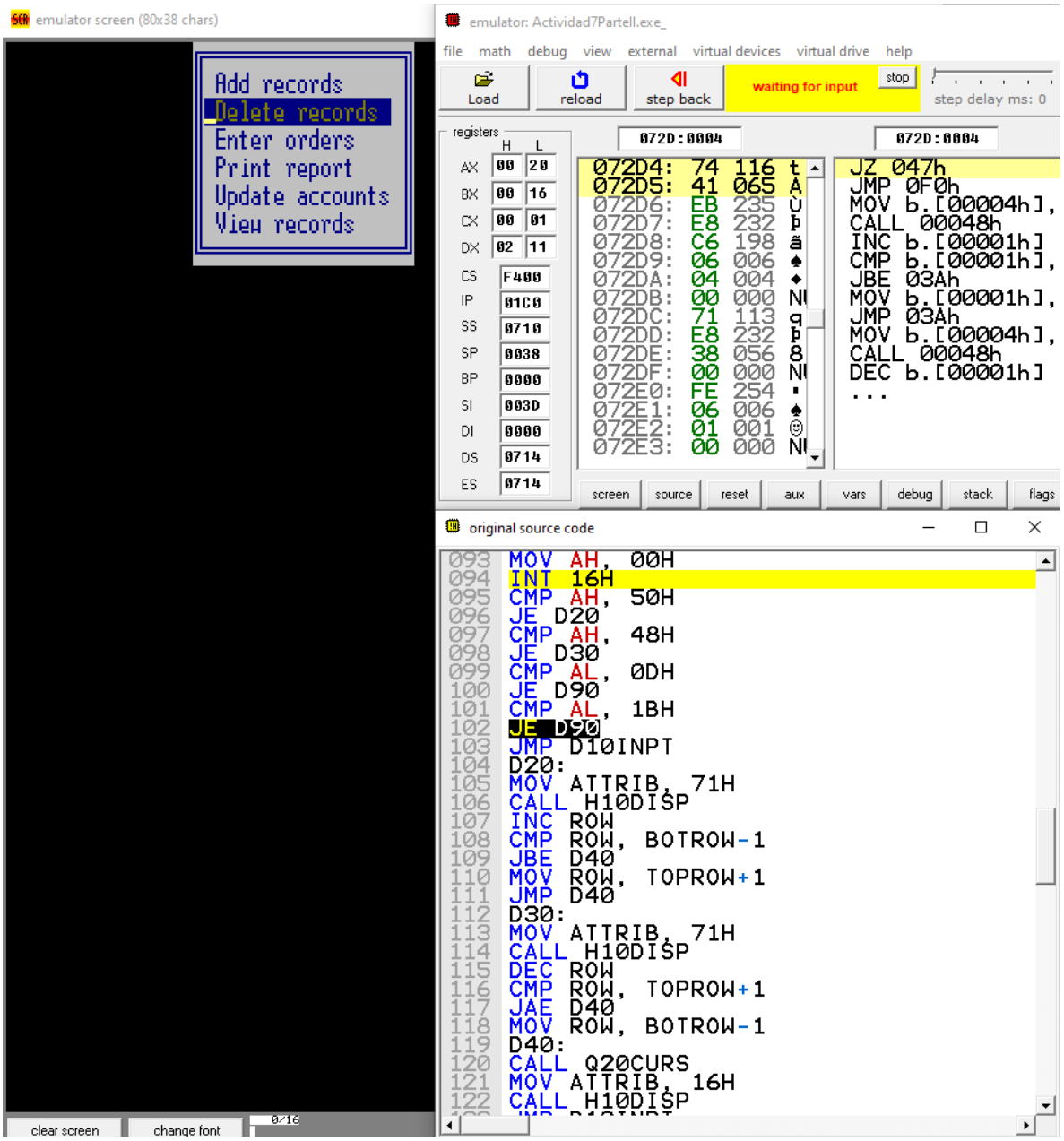
El programa inicia definiendo el segmento de datos, el scroll de la consola con el procedimiento Q10CLR y la posición inicial del cursor con el procedimiento Q20CURS como ya hemos visto varias veces en el curso, después usa la interrupción 21H con AH=40h para escribir las indicaciones del menú en la parte baja, pero no me aparece en mi consola.
Ya en la etiqueta principal A10LOOP primero se llama a B10MENU que imprime carácter a carácter de la cadena MENU considerando en qué columna y fila está para no imprimir caracteres de más o de menos terminando con un recuadro con 8 filas y columnas 19 donde vemos las 6 opciones del menú.

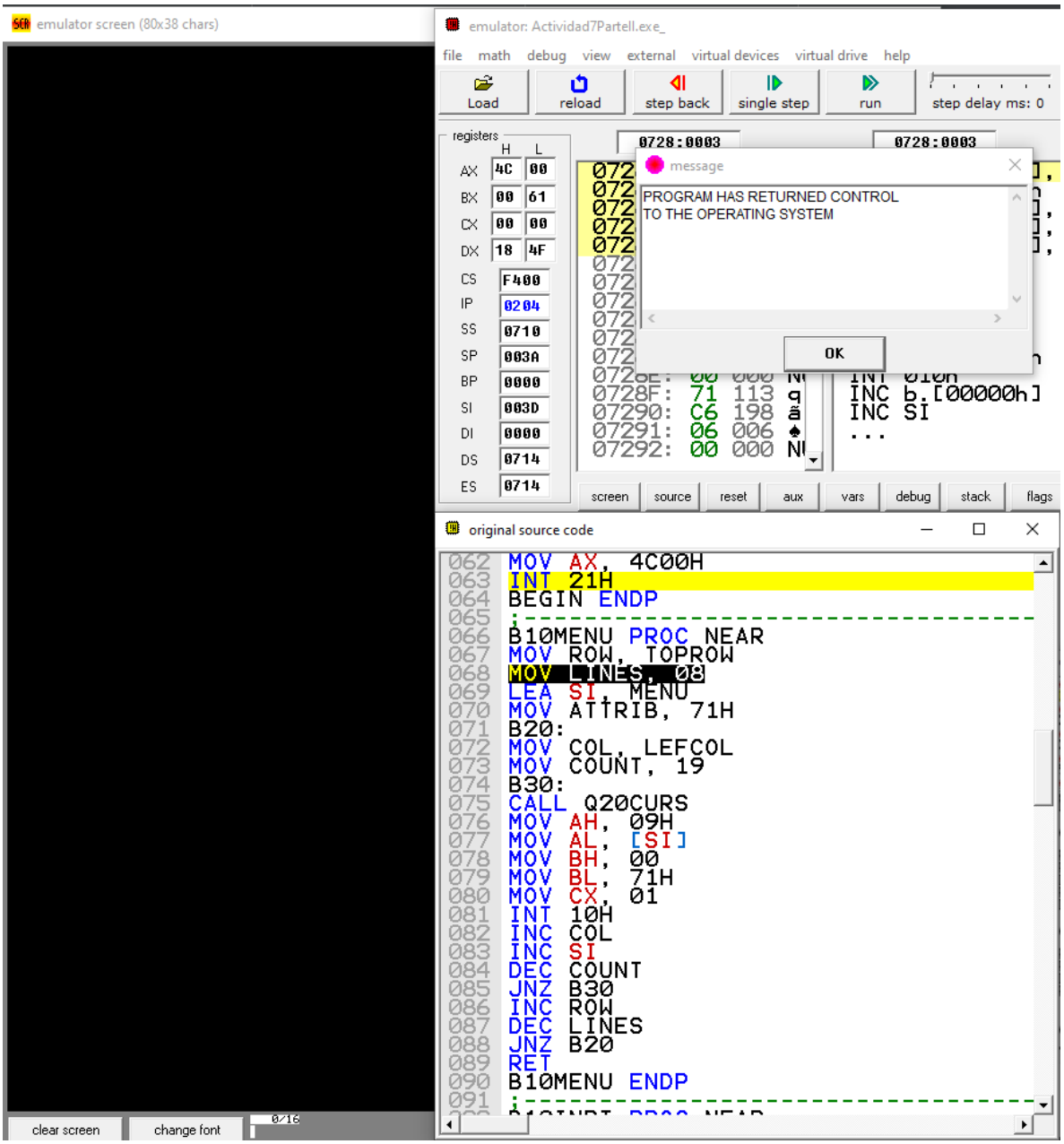


Tras terminar se llama al procedimiento H10DISP que cambiará el color de fondo de la primera opción indicando que estamos seleccionando esa opción del menú.



Por último, se llama al procedimiento D10INPT que espera un input del teclado como las flechas arriba y abajo, ENTER o ESC para poder interactuar con el menú, seleccionar opciones y terminar el ciclo tras pulsar ESC.


emulator screen (80x38 chars)
Add records
Delete records
Enter orders
Print report
Update accounts
View records
clear screen
change font
emulator: Actividad7Partell.exe_
file math debug view external virtual devices virtual drive help
Load
reload
step back
waiting for input
stop
step delay ms: 0
registers
H L
AX 00 20
BX 00 16
CX 00 01
DX 02 11
CS F400
IP 01C0
SS 0710
SP 0038
BP 0000
SI 003D
DI 0000
DS 0714
ES 0714
072D:0004
072D4: 74 116 t JZ 047h
072D5: 41 065 A JMP 0F0h
072D6: EB 235 ù MOV b.[00004h],
072D7: E8 232 Þ CALL 00048h
072D8: C6 198 ã INC b.[00001h]
072D9: 06 006 ♠ CMP b.[00001h],
072DA: 04 004 ♦ JBE 03Ah
072DB: 00 000 N MOV b.[00001h],
072DC: 71 113 q JMP 03Ah
072DD: E8 232 Þ MOV b.[00004h],
072DE: 38 056 8 CALL 00048h
072DF: 00 000 N DEC b.[00001h]
072E0: FE 254 ■ ...
072E1: 06 006 ♠
072E2: 01 001 ☺
072E3: 00 000 N
072D:0004
screen source reset aux vars debug stack flags
original source code
093 MOV AH, 00H
094 INT 16H
095 CMP AH, 50H
096 JE D20
097 CMP AH, 48H
098 JE D30
099 CMP AL, 0DH
100 JE D90
101 CMP AL, 1BH
102 JE D90
103 JMP D10INPT
104 D20:
105 MOV ATTRIB, 71H
106 CALL H10DISP
107 INC ROW
108 CMP ROW, BOTROW-1
109 JBE D40
110 MOV ROW, TOPROW+1
111 JMP D40
112 D30:
113 MOV ATTRIB, 71H
114 CALL H10DISP
115 DEC ROW
116 CMP ROW, TOPROW+1
117 JAE D40
118 MOV ROW, BOTROW-1
119 D40:
120 CALL Q20CURS
121 MOV ATTRIB, 16H
122 CALL H10DISP



Código:

```
;-----------------------------------------------------------------------
.MODEL SMALL
.STACK 64
;-----------------------------------------------------------------------
.DATA
   TOPROW  EQU 00
   BOTROW  EQU 07
   LEFCOL  EQU 16
   COL     DB 00
```

```
   ROW     DB 00
   COUNT   DB ?
   LINES   DB ?
   ATTRIB  DB ?
   NINETEEN DB 19
   MENU    DB 0C9H, 17 DUP(0CDH), 0BBH
           DB 0BAH, ' Add records     ', 0BAH
           DB 0BAH, ' Delete records  ', 0BAH
           DB 0BAH, ' Enter orders    ', 0BAH
           DB 0BAH, ' Print report    ', 0BAH
           DB 0BAH, ' Update accounts ', 0BAH
           DB 0BAH, ' View records    ', 0BAH
           DB 0C8H, 17 DUP(0CDH), 0BCH

   PROMPT  DB  09, 'To select an item, use Up/Down arrow'
           DB  ' and press ENTER.'
           DB  13, 10, 09, 'Press Esc to exit'
;-------------------------------------------------------------------------
.CODE
BEGIN PROC NEAR
   MOV AX, @DATA
   MOV DS, AX
   MOV ES, AX

   CALL Q10CLR
   MOV ROW, BOTROW+2
   MOV COL, 00

   CALL Q20CURS
   MOV AH, 40H
   MOV BX, 01
   MOV CX, 75
   LEA DX, PROMPT
   INT 21H

   A10LOOP:
      CALL B10MENU
      MOV COL, LEFCOL+1

      CALL Q20CURS
      MOV ROW, TOPROW+1
      MOV ATTRIB, 16H

      CALL H10DISP
      CALL D10INPT

      CMP AL, 0DH
```

```
        JE A10LOOP

        MOV AX, 0600H
        CALL Q10CLR

        MOV AX, 4C00H
        INT 21H
BEGIN ENDP
;---------------------------------------------------------------------------
B10MENU PROC NEAR
    MOV ROW, TOPROW
    MOV LINES, 08
    LEA SI, MENU
    MOV ATTRIB, 71H
B20:
    MOV COL, LEFCOL
    MOV COUNT, 19
B30:
    CALL Q20CURS
    MOV AH, 09H
    MOV AL, [SI]
    MOV BH, 00
    MOV BL, 71H
    MOV CX, 01
    INT 10H
    INC COL
    INC SI
    DEC COUNT
    JNZ B30
    INC ROW
    DEC LINES
    JNZ B20
    RET
B10MENU ENDP
;---------------------------------------------------------------------------
D10INPT PROC NEAR
    MOV AH, 00H
    INT 16H
    CMP AH, 50H
    JE D20
    CMP AH, 48H
    JE D30
    CMP AL, 0DH
    JE D90
    CMP AL, 1BH
    JE D90
    JMP D10INPT
```

```
    D20:
       MOV ATTRIB, 71H
       CALL H10DISP
       INC ROW
       CMP ROW, BOTROW-1
       JBE D40
       MOV ROW, TOPROW+1
       JMP D40
    D30:
       MOV ATTRIB, 71H
       CALL H10DISP
       DEC ROW
       CMP ROW, TOPROW+1
       JAE D40
       MOV ROW, BOTROW-1
    D40:
       CALL Q20CURS
       MOV ATTRIB, 16H
       CALL H10DISP
       JMP D10INPT
     D90:
       RET
D10INPT ENDP
;---------------------------------------------------------------------------
H10DISP PROC NEAR
    MOV AH, 00
    MOV AL, ROW

    MUL NINETEEN
    LEA SI, MENU+1

    ADD SI, AX
    MOV COUNT, 16
    H20:
       CALL Q20CURS
       MOV AH, 09H
       MOV AL, [SI]
       MOV BH, 00
       MOV BL, ATTRIB
       MOV CX, 01H
       INT 10H
       INC COL
       INC SI
       DEC COUNT
       JNZ H20
       MOV COL, LEFCOL+1
       CALL Q20CURS
```

```
        RET
H10DISP ENDP
;--------------------------------------------------------------------------
Q10CLR PROC NEAR
   MOV AX, 0600H
   MOV BX, 61H
   MOV CX, 0000
   MOV DX, 184FH
   INT 10H
   RET
Q10CLR ENDP
;--------------------------------------------------------------------------
Q20CURS PROC NEAR
   MOV AH, 02
   MOV BH, 00
   MOV DH, ROW
   MOV DL, COL
   INT 10H
   RET
Q20CURS ENDP
;--------------------------------------------------------------------------
END BEGIN
```

Reflexión

Me pareció interesante el diseño del menú, pero el tiempo que tardaba en mostrarse y seleccionar una opción dentro del emu era demasiado alto que considero que es más útil usar un menú típico de consola.

Actividad 7 – Parte 3

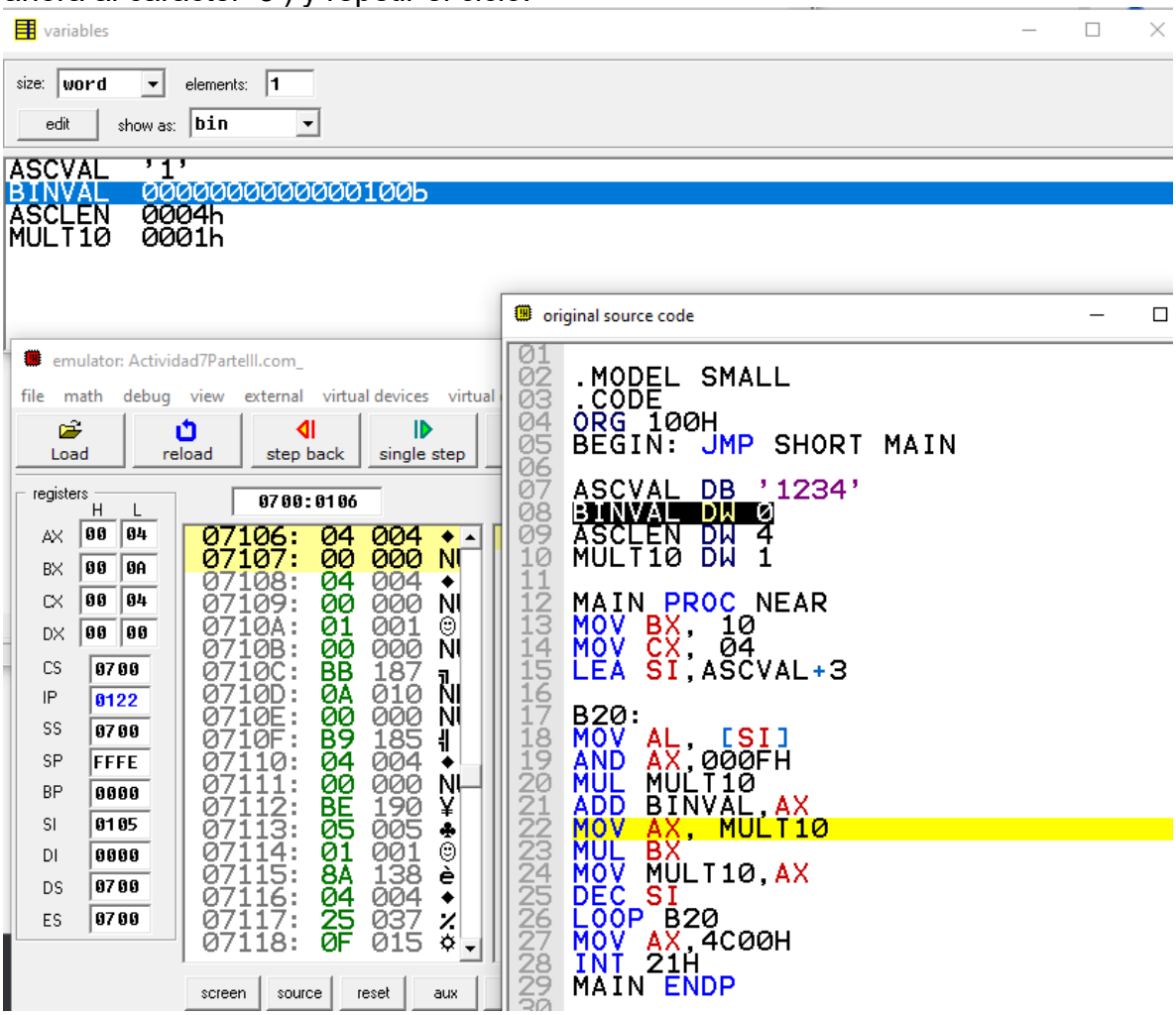
Descripción

Ejecute y explique el siguiente programa para la conversión de ASCII a binario.

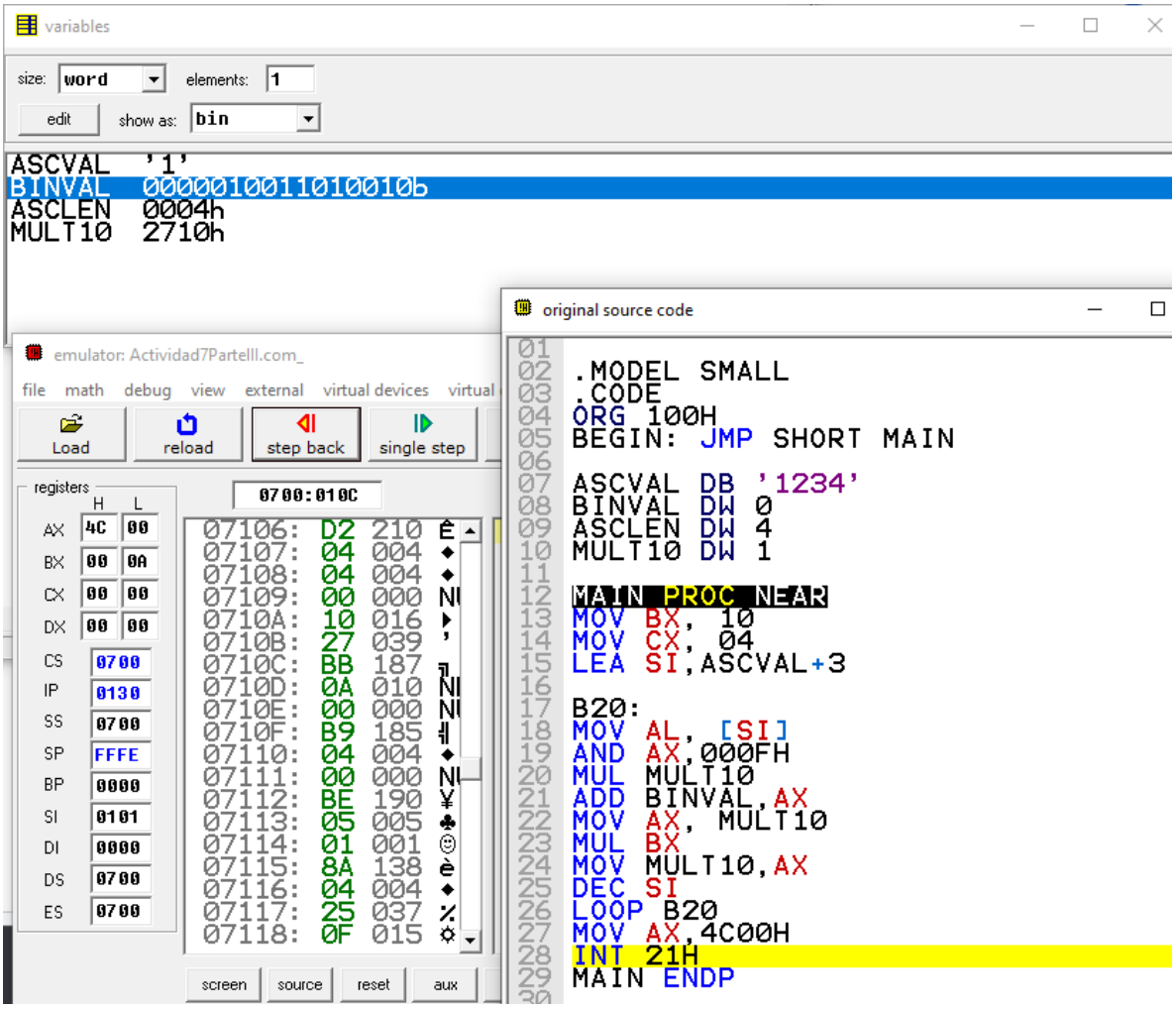
Desarrollo y Resultados

Se declaran cuatro variables:
ASCVAL: que contiene el número que queremos convertir de ASCII a binario en este caso ‘1234’.
BINVAl: donde se almacenará el resultado.
ASCLEN: que contiene la longitud de la cadena, pero no se usa.
MULT10: que es una variable de control que se usa para multiplicar los números y guardarlos como unidades, decenas, etc. Iniciando en 1 y terminando en 1000.

Inicia el procedimiento MAIN donde se mueve 10 a BX para aumentar MULT10 tras cada ciclo, se mueve 4 a CX para controlar la cantidad de ciclos y se guarda la dirección de memoria de ASCVAL en SI sumándole 3 para que apunte al carácter ‘4’.
Dentro del ciclo B20 se mueve el carácter al que apunta SI a Al (en el primer ciclo ‘4’ igual a 34h), se aplica una operación lógica AND para restarle 30 a AL y terminar con el número que necesitamos (en el primer ciclo 4), este se multiplica por MULT10 y se suma a BINVAl, se multiplica por 10 MULT10 y se almacena de nuevo, por último, de decrementa SI para que este apunte al siguiente carácter (apuntando ahora al carácter ‘3’) y repetir el ciclo.



Este ciclo se repite cuatro veces donde en cada uno se suma a BINVAl 4, 30, 200 y 1000 respectivamente en cada ciclo, terminando con el valor 0000010011010010b que en decimal es 1234.



Código:

```
.MODEL SMALL
.CODE
ORG 100H
BEGIN: JMP SHORT MAIN

ASCVAL DB '1234'
BINVAL DW 0
ASCLEN DW 4
MULT10 DW 1

MAIN PROC NEAR
   MOV BX, 10
   MOV CX, 04
   LEA SI,ASCVAL+3

B20:
```

```
    MOV AL, [SI]
    AND AX,000FH
    MUL MULT10
    ADD BINVAL,AX
    MOV AX, MULT10
    MUL BX
    MOV MULT10,AX
    DEC SI
    LOOP B20
    MOV AX,4C00H
    INT 21H
MAIN ENDP

END BEGIN
```

Reflexión

En esta actividad pudimos ver un método para convertir cadenas de caracteres numéricos a binario e incluso a otros sistemas de conteo, demostrando así las posibilidades que tenemos de convertir datos capturados en otros formatos para usarlos en diferentes procesos o operaciones.