Arellano Granados Angel Mariano

218123444

Seminario de Traductores de Lenguajes I

I7026 D02

Reporte de Actividades 5

Actividades 5 – Parte 1 – 3

Actividad 5 – Parte 1

Descripción

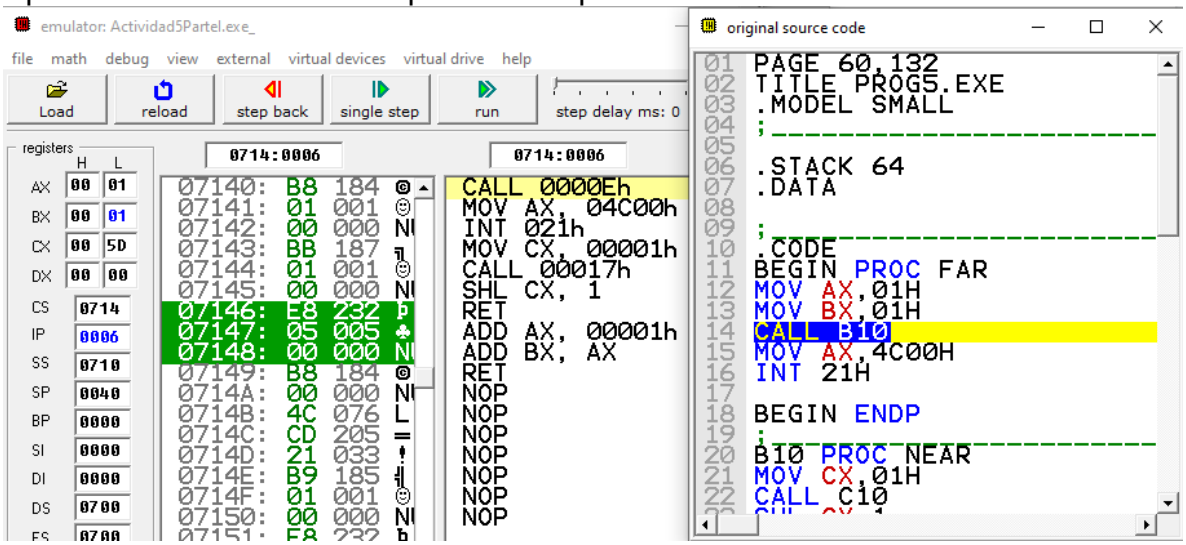
Ejecutar el siguiente programa y responda las siguientes preguntas

1. ¿Cómo se ejecutan las llamadas a procedimientos?
2. ¿Cuál es la diferencia entre FAR y NEAR?
3. ¿Qué produce la línea de código <SHL CX,1> en el registro CX?
4. ¿Qué registros participan en la ejecución de la instrucción RET?

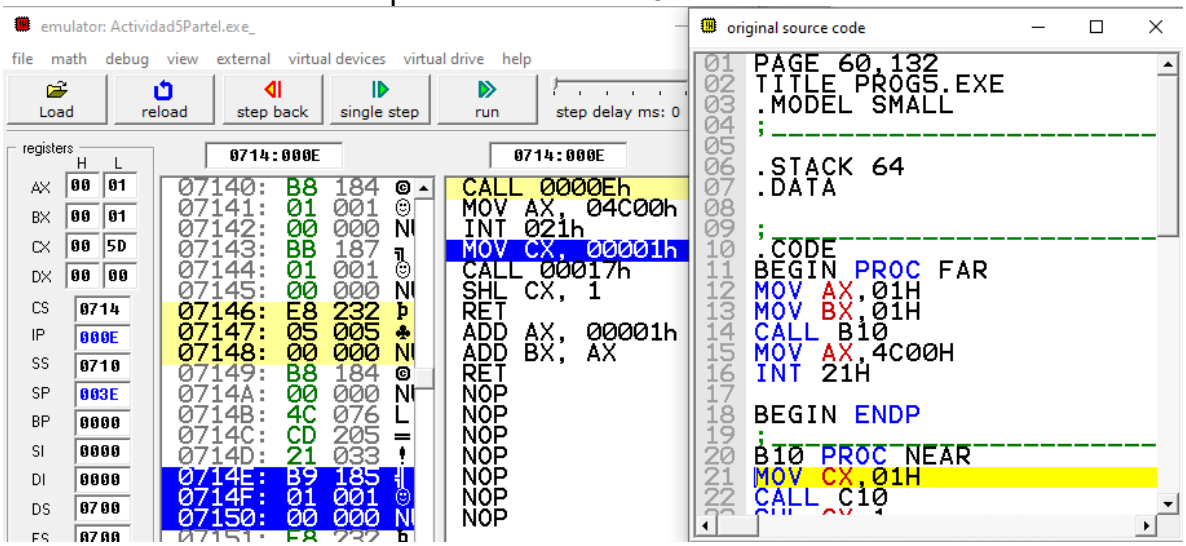
Desarrollo y Resultados

1. ¿Cómo se ejecutan las llamadas a procedimientos?

Con la instrucción CALL <nombre del procedimiento>.
Aparece la linea CALL B10 que llama al proceso llamado “B10”



Entonces salta al inicio del procedimiento “B10”

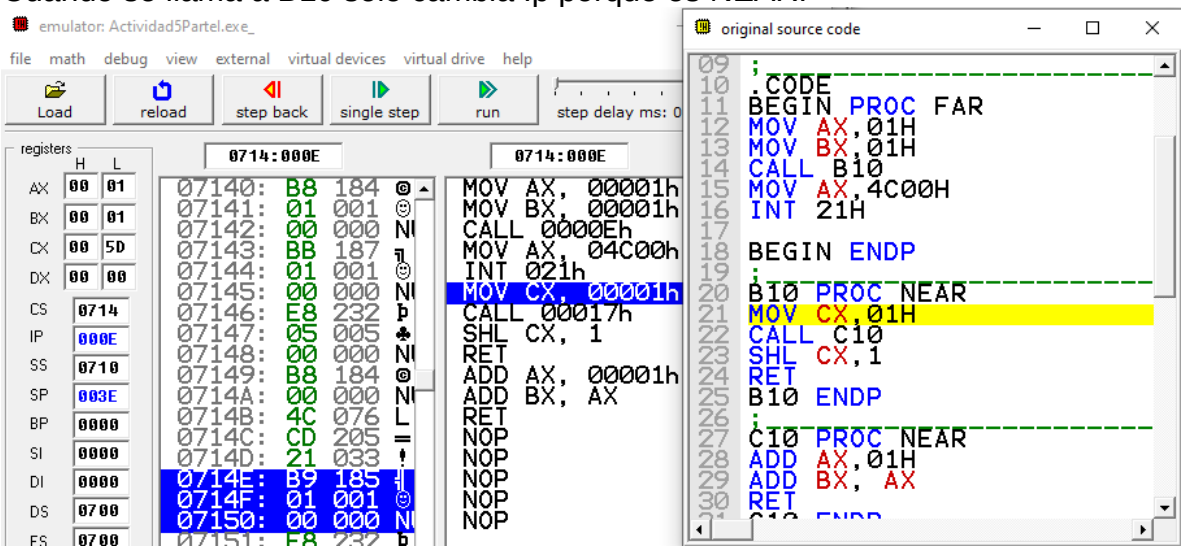


2. ¿Cuál es la diferencia entre FAR y NEAR?

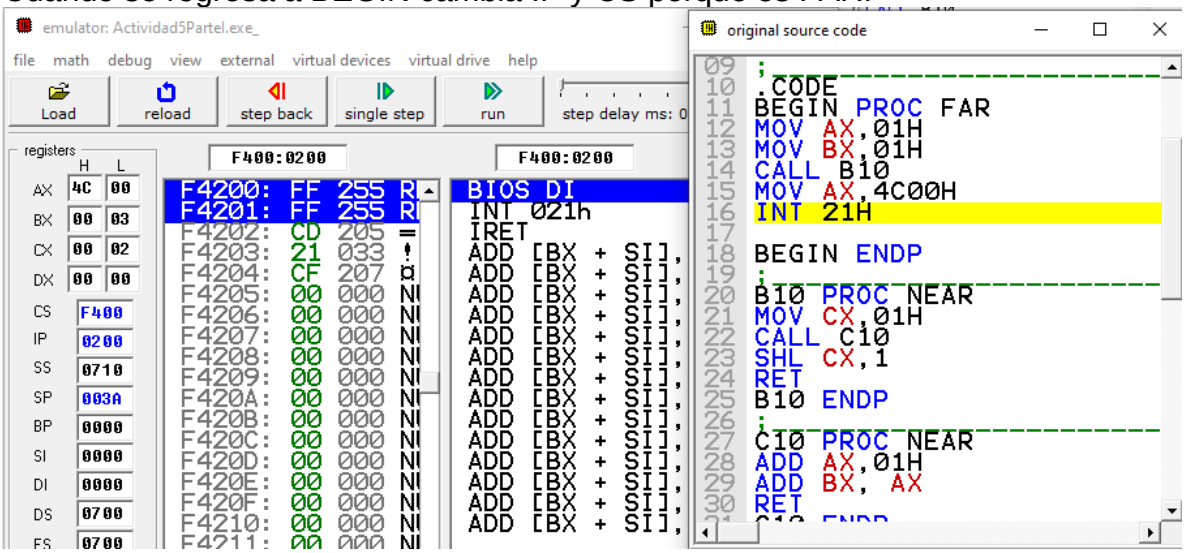
Near contains a 16-bit offset. For calls it will save the IP only.

Far contains a segment and a 16-bit offset. For calls it will save IP and CS.

Cuando se llama a B10 solo cambia Ip porque es NEAR:



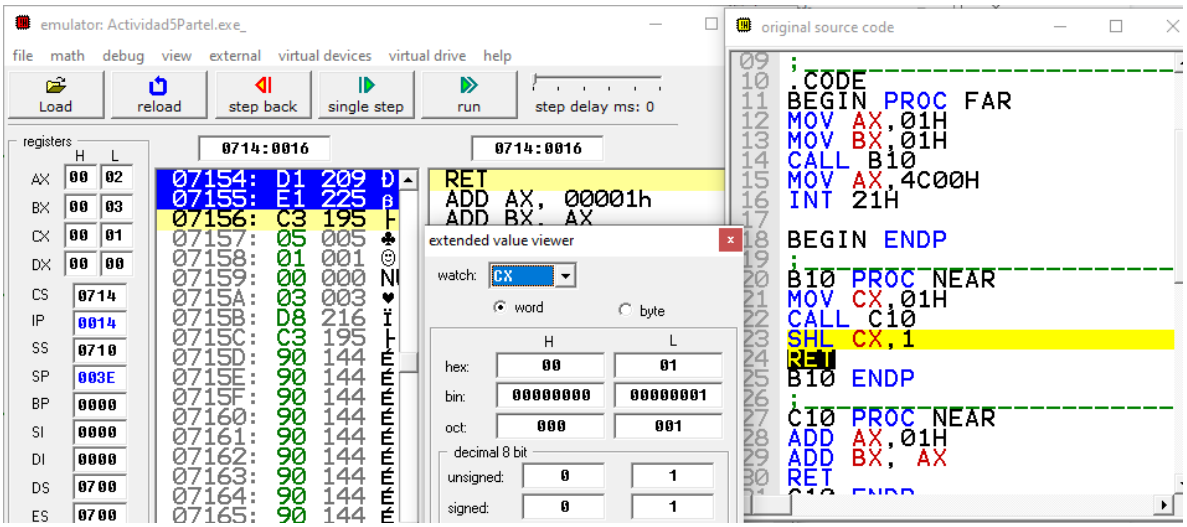
Cuando se regresa a BEGIN cambia IP y CS porque es FAR:



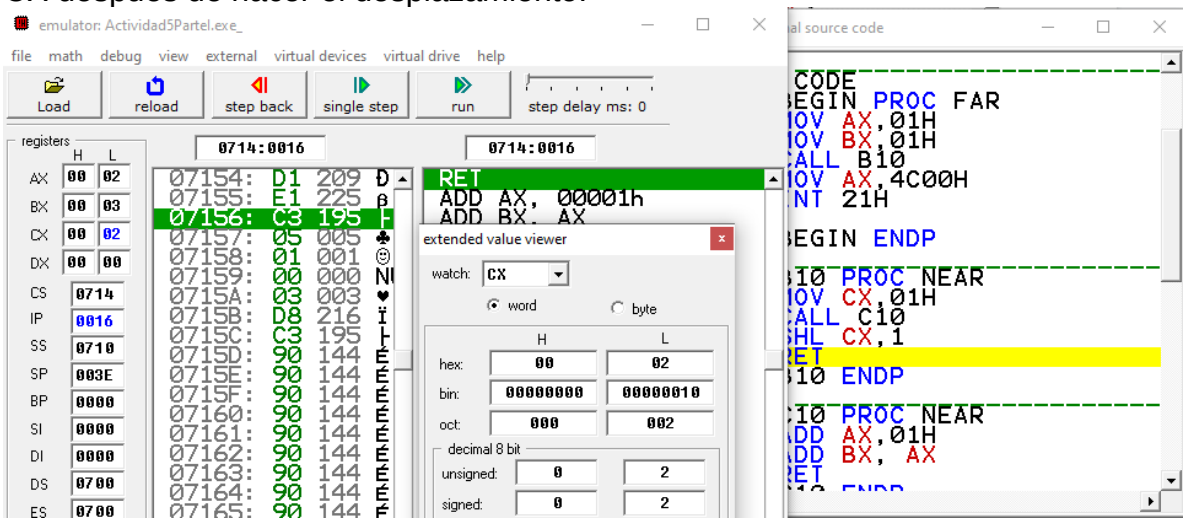
3. ¿Qué produce la línea de código <SHL CX,1> en el registro CX?

hace un desplazamiento lógico a la izquierda SHL destino, origen.

CX antes de ejecutar la instrucción:



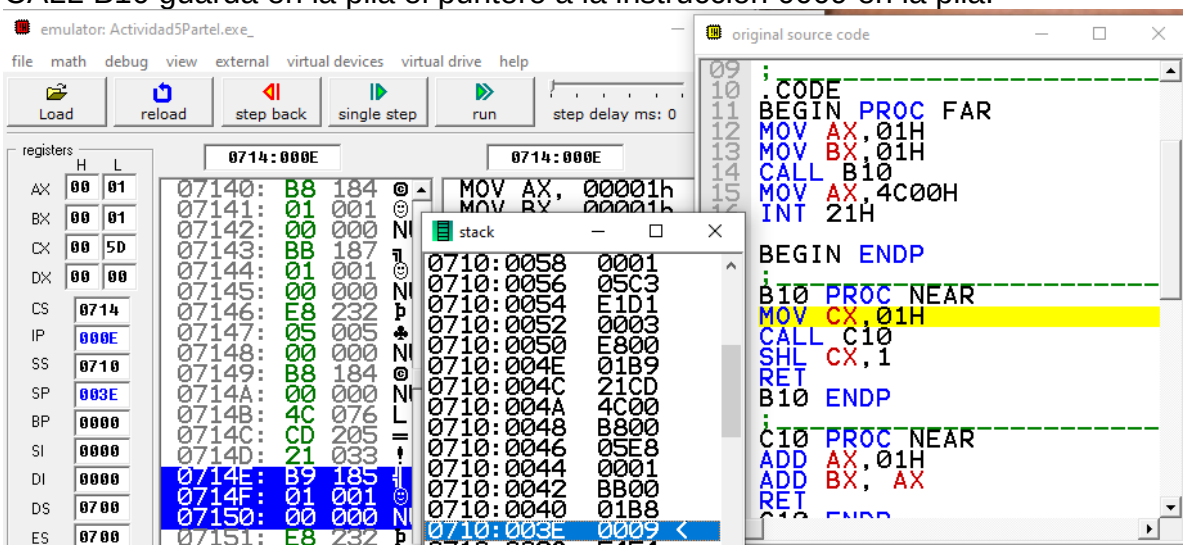
CX después de hacer el desplazamiento:



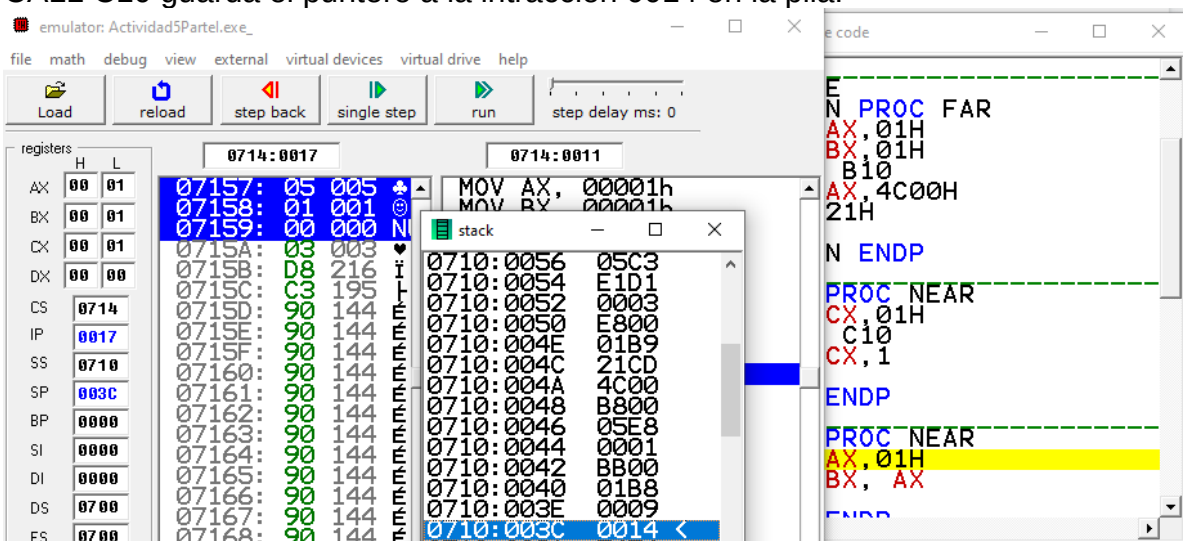
4. ¿Qué registros participan en la ejecución de la instrucción RET?

Se usan los registros IP y SP porque CALL guarda en la pila el puntero de la instrucción a la que tendrá que regresar tras terminar el proceso, RET toma el elemento más reciente de la pila y lo coloca en IP así se mantiene el flujo del programa.

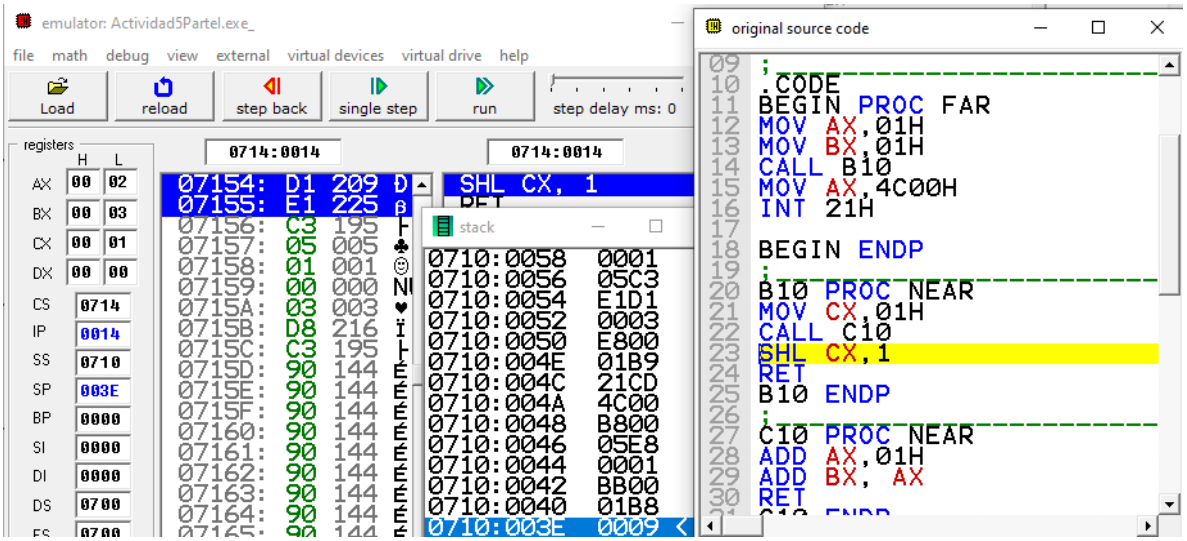
CALL B10 guarda en la pila el puntero a la instrucción 0009 en la pila:



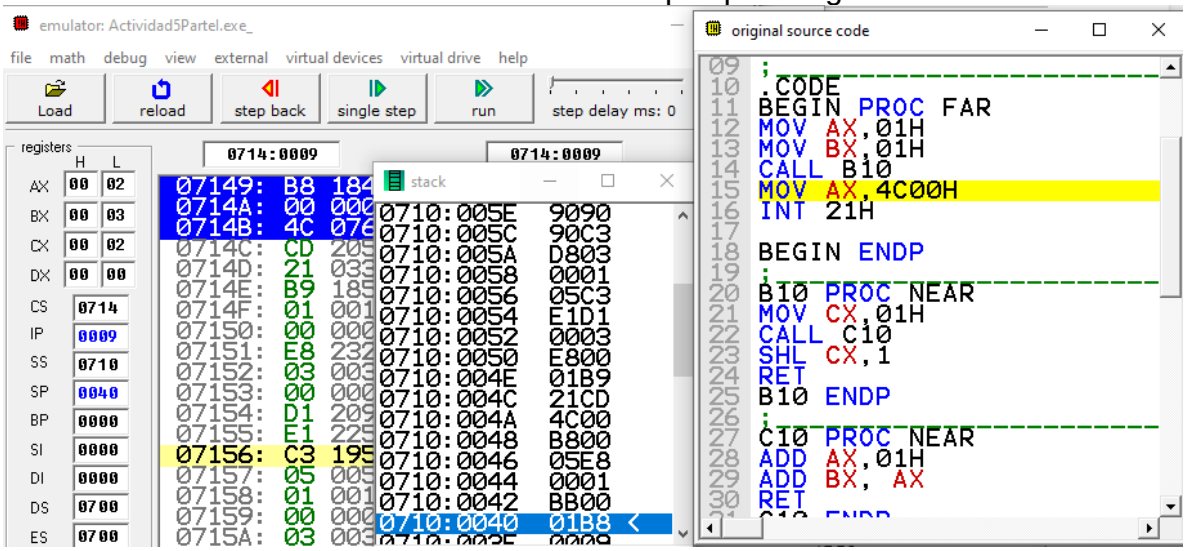
CALL C10 guarda el puntero a la intraccion 0014 en la pila:



La instrucción RET en C10 toma el 0014 de la pila para regresar a B10:



La instrucción RET en B10 toma el 0009 en la pila para regresar a BEGIN:



Reflexión

Los procedimientos son la instrucción equivalente a las funciones en C junto a las etiquetas y las macros que nos ayudan a modular nuestros programas y no repetir código innecesario, también vimos instrucciones mas complejas en sus aplicaciones como los desplazamientos de bits que en este punto del curso aun no les encuentro una utilidad.

Actividad 5 – Parte 2

Descripción

Investigue el funcionamiento de la instrucción LEA, ejecute el siguiente código fuente y explique paso a paso lo que sucede.

Desarrollo y Resultados

Instrucción LEA

Carga en un registro especificado la dirección efectiva especificada como en el operando origen:

LEA reg, mem

Ejemplos:

LEA AX, [BX] ; carga en AX la dirección apuntada por BX.

LEA AX, 3[BP+SI] ; carga en AX la dirección resultado de multiplicar por 3 la suma de los contenidos de BP y SI.

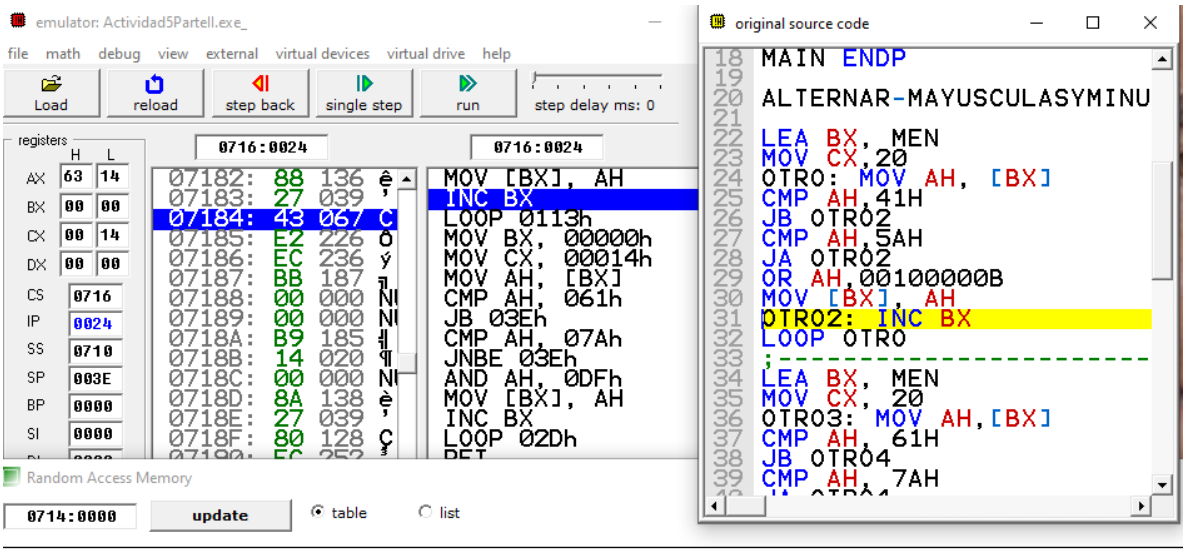
Para este tipo de acceso (el del segundo ejemplo) la instrucción LEA es más eficiente, que con la instrucción MOV e instrucciones de multiplicación.
https://www.cs.buap.mx/~mgonzalez/asm_mododir2.pdf

Reporte

Tras declarar todas las generalidades del programa y se declara una variable llamada “MEN” con la cadena 'CAMBIAR A MINUSCULAS'.

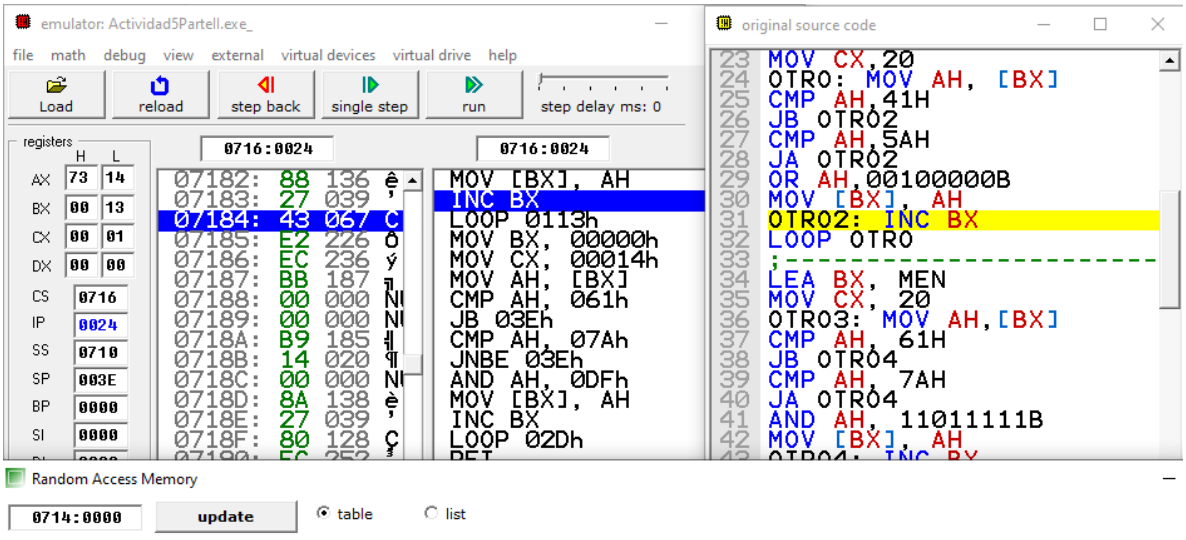
Ya iniciando el segmento de código se declara un procedimiento llamado “MAIN” que carga el segmento de datos y llama a otro procedimiento “ALTERNAR-MAYUSCULASYMINUSCULAS”

Donde este con la instrucción LEA coloca en la dirección de memoria de “MEN” en BX, coloca un 20d en CX (El número de caracteres en “MEN”) y comienza una etiqueta llamada “OTRO” que mueve a AH el valor de cada carácter en la cadena “MEN” y compara si AH es 41h si es menor salta directamente a la etiqueta llamada “OTRO2”, si no pasara de todas formas pero antes compara si AH es 5Ah si es mayor salta directamente a “OTRO2”, si no se hace una función OR en AX con 00100000b esto equivale a sumar 20h al número lo cual convierte al carácter a su misma letra en minúscula y terminar moviendo el número resultante de AH y moverlo a la dirección de memoria de “MEN”.



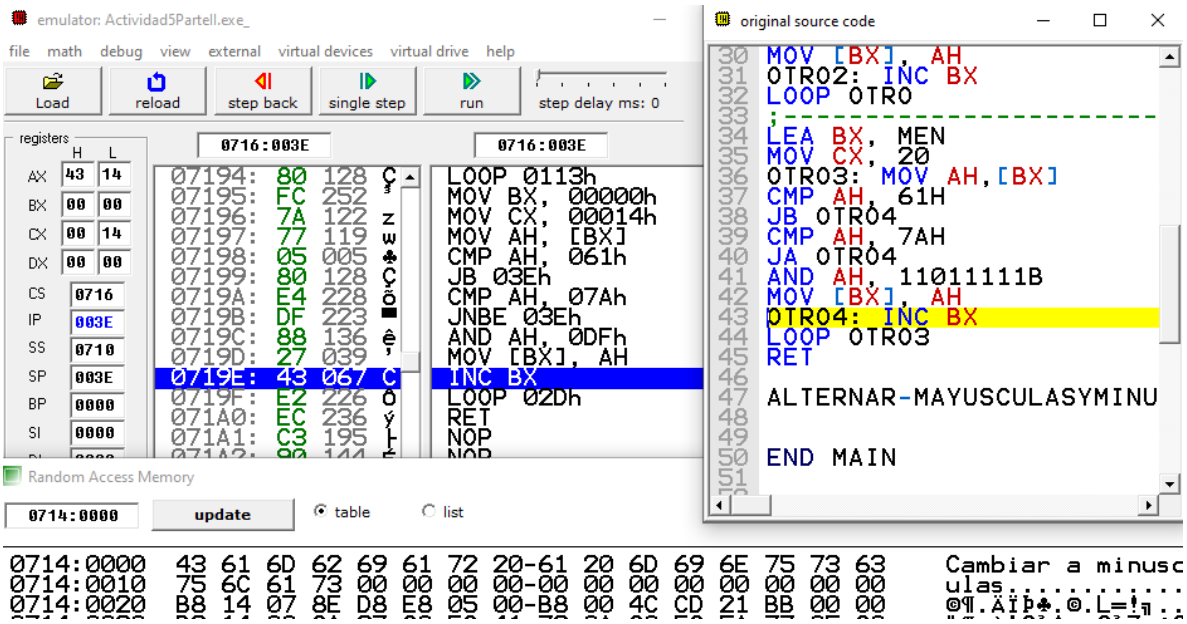

```
0714:0000  63 41 4D 42 49 41 52 20-41 20 4D 49 4E 55 53 43    cAMBIAR A MINUSC
0714:0010  55 4C 41 53 00 00 00 00-00 00 00 00 00 00 00 00    ULAS............
0714:0020  B8 14 07 8E D8 E8 05 00-B8 00 4C CD 21 BB 00 00    ®¶.ÄïÞ♣.◙.L=!╗..
0714:0030  B9 14 00 8A 27 80 FC 41-72 0A 80 FC 5A 77 05 80    ╣¶.è'Ç³Ar.Ç³Zw♣Ç
0714:0040  CC 20 88 27 43 E2 EC BB-00 00 B9 14 00 8A 27 80    ╠ ê'CôýÝ¶..╣¶.è'Ç
0714:0050  FC 61 72 0A 80 FC 7A 77-05 80 E4 DF 88 27 43 E2    ³ar.Ç³zw♣Çö■ê'Cô
0714:0060  EC C3 90 90 90 90 90 90-90 90 90 90 90 90 90 90    ýŀÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉ
0714:0070  90 90 90 90 90 90 F4 00-00 00 00 00 00 00 00 00    ÉÉÉÉÉÉ¶.........
```

Repitiendo esto un total de 20 veces con un LOOP cambia la cadena de “MEN” de 'CAMBIAR A MINUSCULAS' a ‘cambiar a minúsculas’



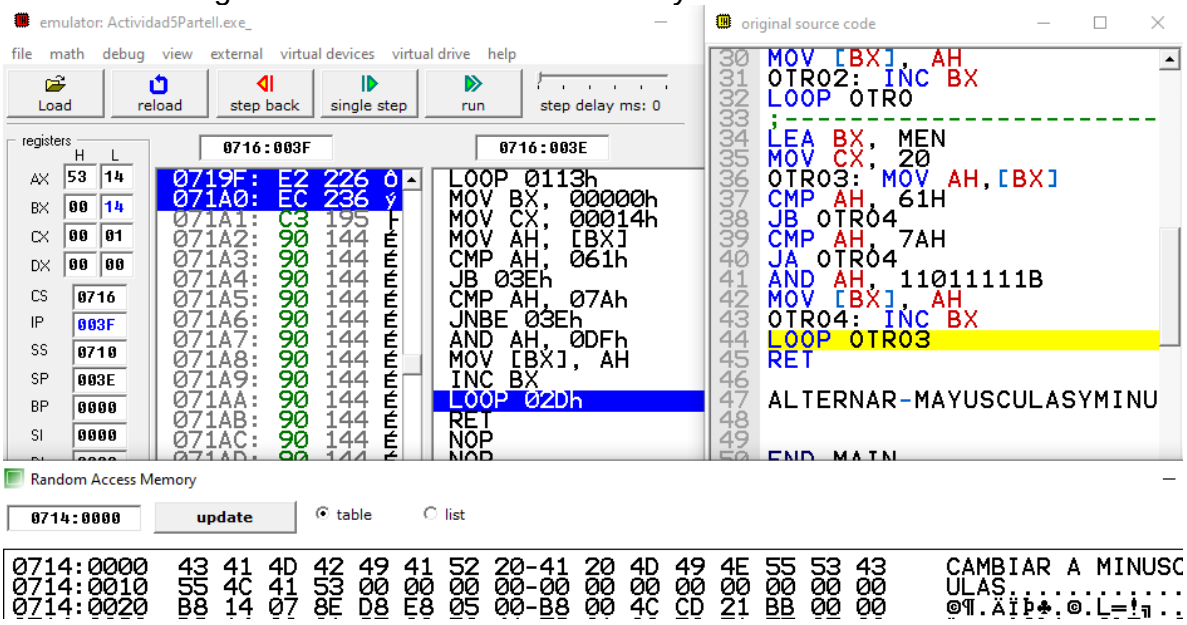

```
0714:0000  63 61 6D 62 69 61 72 20-61 20 6D 69 6E 75 73 63    cambiar a minusc
0714:0010  75 6C 61 73 00 00 00 00-00 00 00 00 00 00 00 00    ulas............
0714:0020  B8 14 07 8E D8 E8 05 00-B8 00 4C CD 21 BB 00 00    ®¶.ÄïÞ♣.◙.L=!╗..
```

Se reinicia el registro BX y se vuelve a poner 20d en CX para entrar a la etiqueta “OTRO3” que hace básicamente lo mismo que “OTRO” pero este cambia los números de las comparaciones y del OR para esta vez cambiar las letras de la cadena “MEN” de minúsculas a mayúsculas.




```
0714:0000  43 61 6D 62 69 61 72 20-61 20 6D 69 6E 75 73 63    Cambiar a minusc
0714:0010  75 6C 61 73 00 00 00 00-00 00 00 00 00 00 00 00    ulas............
0714:0020  B8 14 07 8E D8 E8 05 00-B8 00 4C CD 21 BB 00 00    ®¶.ÄïÞ♣.®.L═!╗..
```

También entrará en un LOOP que se repetirá 20 veces hasta que la cadena regrese a su estado original con todas sus letras en mayúsculas.




```
0714:0000  43 41 4D 42 49 41 52 20-41 20 4D 49 4E 55 53 43    CAMBIAR A MINUSC
0714:0010  55 4C 41 53 00 00 00 00-00 00 00 00 00 00 00 00    ULAS............
0714:0020  B8 14 07 8E D8 E8 05 00-B8 00 4C CD 21 BB 00 00    ®¶.ÄïÞ♣.®.L═!╗..
```

Para terminar retorna a “MAIN” para con la interrupción 21h termina el programa con un mensaje de aviso.

Código

```
PAGE 60,132
TITLE PROG6.EXE
.MODEL SMALL
;-------------------------------------
.STACK 64
.DATA
MEN DB 'CAMBIAR A MINUSCULAS'
;-------------------------------------
.CODE
MAIN PROC NEAR

    MOV AX, @DATA
    MOV DS, AX
    CALL ALTERNAR-MAYUSCULASYMINUSCULAS
    MOV AX,4C00H
    INT 21H

MAIN ENDP

ALTERNAR-MAYUSCULASYMINUSCULAS PROC NEAR

    LEA BX, MEN
    MOV CX,20
    OTRO: MOV AH, [BX]
    CMP AH,41H
    JB OTRO2
    CMP AH,5AH
    JA OTRO2
    OR AH,00100000B
    MOV [BX], AH
    OTRO2: INC BX
    LOOP OTRO
;---------------------------------------
    LEA BX, MEN
    MOV CX, 20
    OTRO3: MOV AH,[BX]
    CMP AH, 61H
    JB OTRO4
    CMP AH, 7AH
    JA OTRO4
    AND AH, 11011111B
    MOV [BX], AH
    OTRO4: INC BX
    LOOP OTRO3
    RET

ALTERNAR-MAYUSCULASYMINUSCULAS ENDP


END MAIN
```

Reflexión

En esta actividad vimos 2 instrucciones nuevas totalmente útiles e interesantes dado a que los ciclos repetitivos son una de las estructuras mas usadas en la programación para infinidad de implementaciones y LEA que nos ofrece una alternativa mas simple para encontrar las direcciones de memoria de nuestras vaciables sin usar MOV y offset.

Actividad 5 – Parte 3

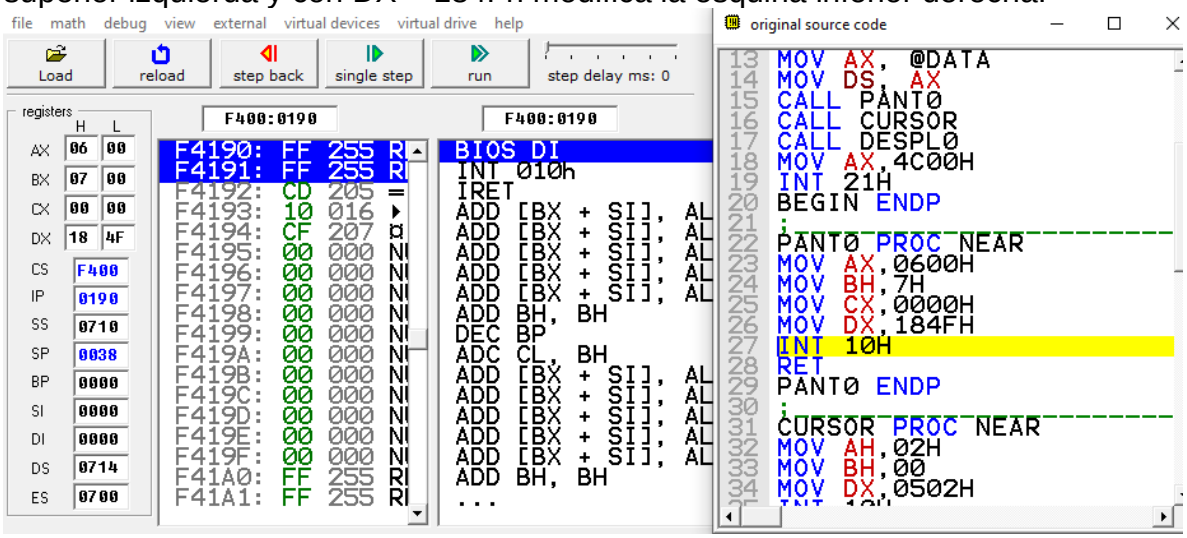
Descripción

Explique el funcionamiento de la interrupción 10H en el siguiente programa.

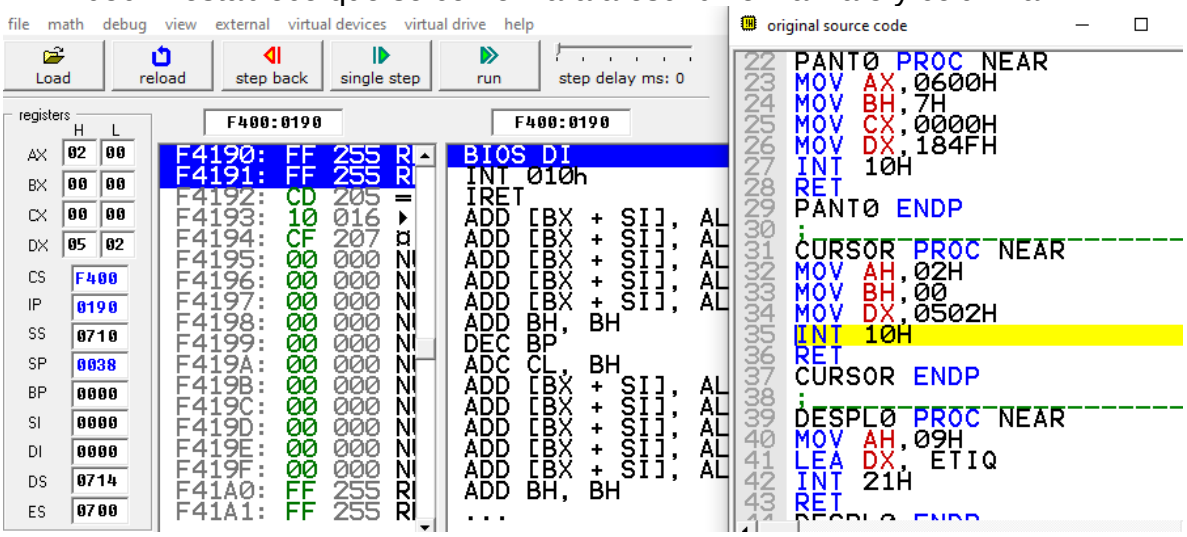
Desarrollo y Resultados

Reporte:

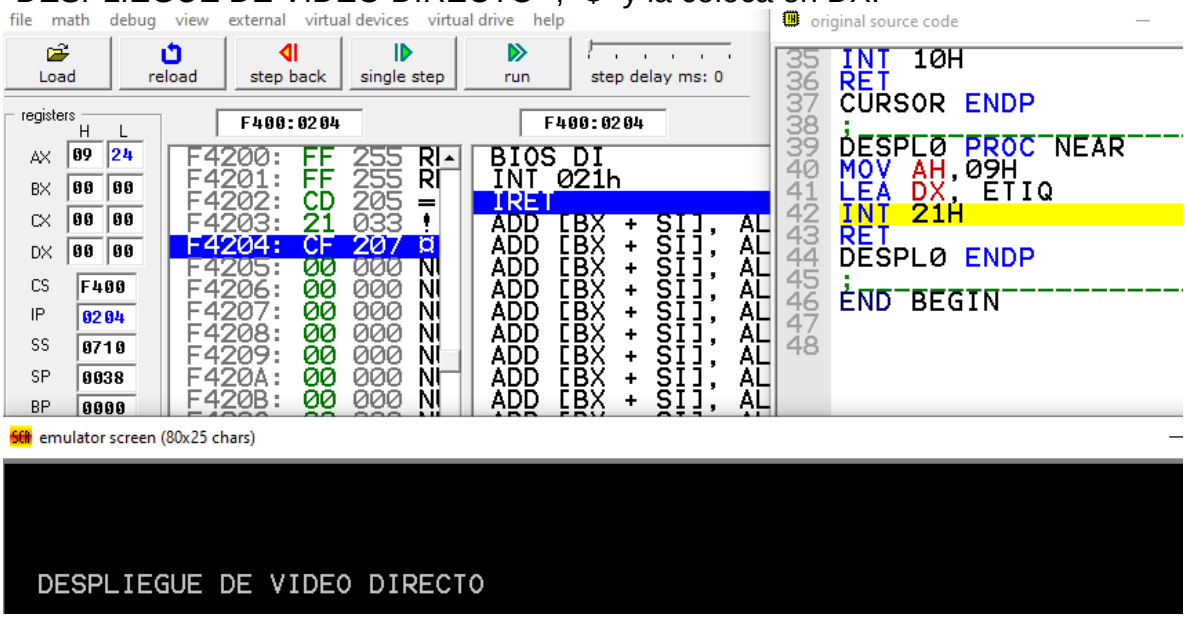
El programa inicia en un procedimiento llamado BEGIN que inicia el segmento de datos y llama otros 3 procedimientos para despues terminar el programa, primero llama a PANT0 donde se usa la interrupción 10H con 06h en AH para recorrer hacia arriba la pantalla donde con AL= 0 no recorre ninguna línea hacia arriba, con BH = 7 escribe 7 líneas vacías en el fondo de la ventana, con CX = 0 no cambia la esquina superior izquierda y con DX = 184Fh modifica la esquina inferior derecha.



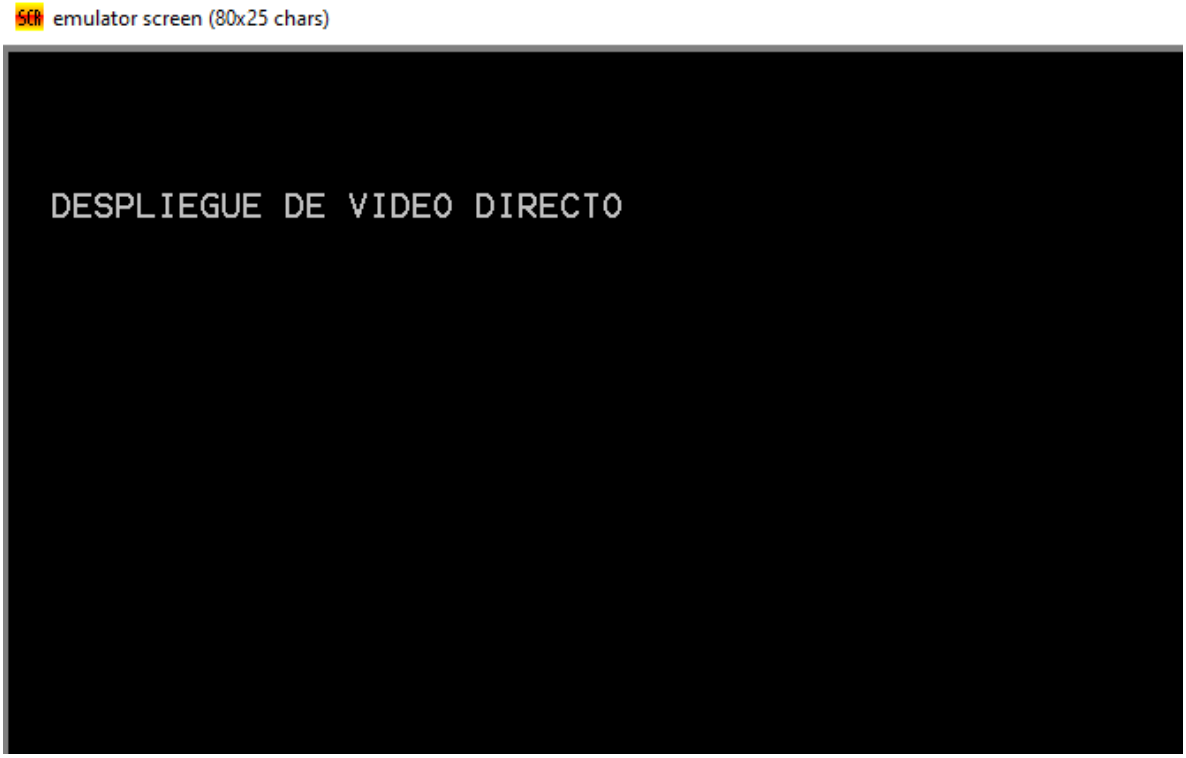
Después llama al proceso CUSOR que con interrupción 10H con 02h en AH establece la posición del cursor con BH = 0 establece el número de la página y con DX = 0502h establece que se comenzara a escribir en la fila 5 y columna 2.



Por último, llama al procedimiento llamado DESPL0 que con la interrupción 21H y 09h en AH escribe una cadena de caracteres en la consola con LEA obtiene la dirección de memoria de la cadena previamente declarada en el segmento de datos "DESPLIEGUE DE VIDEO DIRECTO ", "$" y la coloca en DX.



Así terminamos con la consola donde se puede leer la cadena en un punto centrado gracias a las modificaciones que se hicieron con la interrupción 10H.



Código:

```
PAGE 60,132
TITLE PROG7.EXE
.MODEL SMALL
.STACK 64
;---------------------------------------

.DATA

ETIQ DB "DESPLIEGUE DE VIDEO DIRECTO ", "$"
;---------------------------------------
.CODE
BEGIN PROC FAR
MOV AX, @DATA
MOV DS, AX
CALL PANT0
CALL CURSOR
CALL DESPL0
MOV AX,4C00H
INT 21H
BEGIN ENDP
;---------------------------------------
PANT0 PROC NEAR
MOV AX,0600H
MOV BH,7H
MOV CX,0000H
MOV DX,184FH
INT 10H
RET
PANT0 ENDP
;---------------------------------------
CURSOR PROC NEAR
MOV AH,02H
MOV BH,00
MOV DX,0502H
INT 10H
RET
CURSOR ENDP
;---------------------------------------
DESPL0 PROC NEAR
MOV AH,09H
LEA DX, ETIQ
INT 21H
RET
DESPL0 ENDP
;---------------------------------------
END BEGIN
```

Reflexión

En esta actividad pudimos implementar la interrupción 10h usando solo 2 de sus funciones pero por actividades anteriores sabemos que esta se basa en su mayor parte en modificar la salida visual del programa aquí solo imprimimos una simple cadena, pero marca un inicio para hacer programas mas complejos en el futuro.