Ensamblador para ZX Spectrum

Hola Retro Parla

Anexo II

Contenido

Introducción

Para dar por finalizado nuestro programa Hola Retro Parla, vamos a desplazar nuestro saludo, pixel a pixel, de izquierda a derecha. Cuando salga por la derecha volverá a aparecer por la izquierda.

Para poder realizar este desplazamiento necesitamos:

1. Haber leído los documentos 1, 2 y 3.
2. Calcular posiciones de pantalla en base a coordenadas de carácter.
3. Rotar bytes hacia la derecha.

Antes de continuar vamos a crear un archivo llamado Scroll.asm, en el que vamos a desarrollar las rutinas necesarias para desplazar nuestro mensaje.

Posiciones de memoria de la pantalla

Las posiciones de memoria de la pantalla de un ZX Spectrum van desde la dirección $4000 a la $57FF; 6144 posiciones de pantalla que hacen una resolución de 256x192 píxeles.

Para definir parpadeo, brillo, fondo y tinta se usan las direcciones de la $5800 a las $5AFF; 708 posiciones que hacen un total de 32*24 caracteres.

En este ejemplo nos vamos a centrar en las primeras 6144 posiciones, las que usamos para dibujar los píxeles.

La pantalla del ZX Spectrum se divide en tres tercios, con ocho líneas cada uno, las cuales tienen a su vez ocho scanlines.

Cada posición de memoria se codifica de la siguiente manera:

```
010T TSSS LLLC CCCC
```

Dónde 010 es un valor fijo, TT es el tercio (0 a 2), SSS es el scanline (0 a 7), LLL es la línea dentro del tercio (0 a 7) y CCCCC es la columna (de 0 a 31). Es muy importante recordar que el 0 es el primer tercio, scanline, línea o columna.

Si deseamos dibujar el patrón 01010101b en la columna 10, fila 8, debemos recordar que la columna seria la 9 y la fila 8 es la fila 0 del segundo tercio, por lo que la posición de memoria sería:

```
0100 1000 0000 1001

010T TSSS LLLC CCCC

$4809
```

Una vez visto esto, necesitamos una rutina a la que le pasemos las coordenadas X e Y de una posición de carácter, y nos devuelva la posición de memoria del primer scanline de dichas coordenadas.

Esta rutina recibe en B la coordenada Y, en C la coordenada X y retorna la dirección en HL. Para ello carga 0100 0000b en H y luego pone los bits 4 y 5 de Y en los bits 4 y 5 de H. Tras esto carga C en L y luego pone los bits 0, 1 y 2 de B en los bits de 5, 6, y 7 de L.

```
; ------------------------------------------------------------------------
; GetPointerCoord
; Obtiene la dirección de memoria del primer scanline correspondiente a las coordenadas especificadas.
; Las posiciones de memoria relativas a la pantalla se codifican de la siguiente manera:
;     010T TSSS LLLC CCCC
;     010 es un valor fijo.
;     TT es el tercio de la pantalla.
;     LLL es la línea dentro del tercio.
;     SSS es el scanline de la línea.
;     CCCCC es la columna.
;
; Entrada:      B -> Coordenada Y.
;               C -> Coordenada X.
```

```
; Salida:       HL -> Dirección de memoria resultante.
;
; Altera el valor de los registros AF y HL.
; ------------------------------------------------------------------------
GetPointerCoord:
; En B viene la coordenada Y.
;       Los bits 0 a 2 son la línea dentro del tercio.
;       Los bits 3 y 4 son el tercio.

ld      h, $40          ; Carga en H el valo fijo 0100
ld      a, b            ; Carga la coordenada Y en A
and     $18             ; Se queda con la parte del tercio
or      h               ; Agrega el valor fijo
ld      h, a            ; Parte alta de la dirección calculada

ld      a, b            ; Carga la coordenada Y en A
and     $07             ; Se queda con la parte de la línea
rrca
rrca
rrca                    ; Pasa el valor a los bits 5 a 7
or      c               ; Añade la columna
ld      l, a            ; Parte baja de la dirección calculada

ret
```

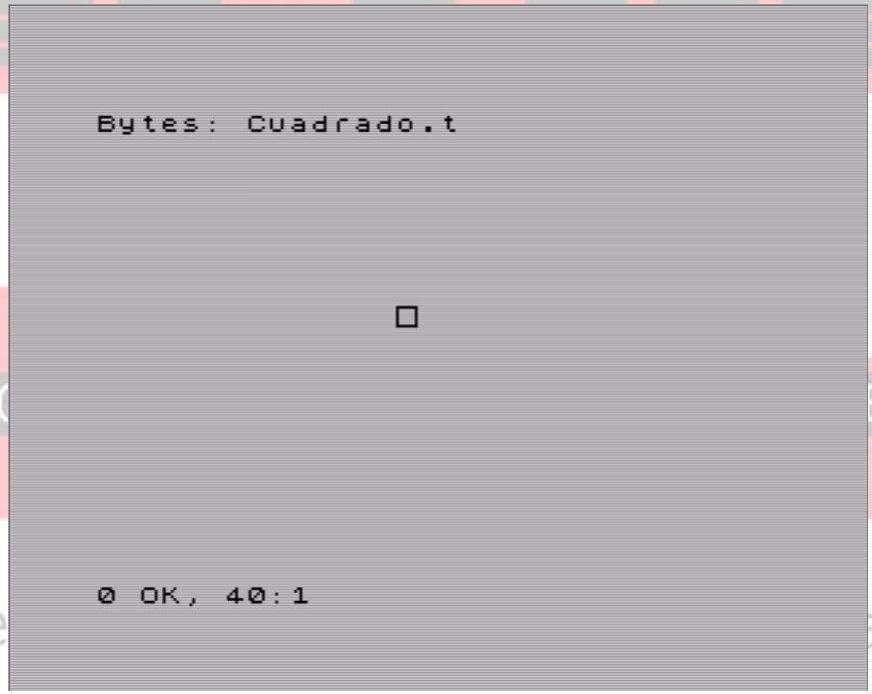
Para probar esta rutina, vamos a dibujar un cuadrado en la línea 11, columna 15. Creamos un nuevo fichero Cuadrado.asm y escribimos el siguiente código:

```
; Cuadrado.asm
; Dibuja un cuadrado en la línea 11, columna 15.
org     $8000           ; Dirección de memoria donde se carga el programa

Cuadrado:
ld      b, $0a          ; Carga en B la coordenada Y
ld      c, $0e          ; Carga en C la coordenada X
call    GetPointerCoord ; Obtiene la dirección de memoria correspondiente
ld      (hl), $ff       ; Dibuja el inicio del cuadrado
; En los siguientes 6 scanlines, se dibuja lo mismo
ld      b, $06
Loop:
inc     h               ; Avanza al siguiente scanline
ld      (hl), $81       ; Dibuja en pantalla . .
djnz    Loop            ; Se repite seis veces
inc     h               ; Avanza al último scanline
ld      (hl), $ff       ; Dibuja el final del cuadrado

include "Scroll.asm"    ; Incluye el fichero Scroll.asm para poder
                        ; llamar a la rutina GetPointerCoord
end     $8000
```

Compilamos:

```
pasmo --tapbas Cuadrado.asm Cuadrado.bas
```

Y vemos el resultado:



Desplazamiento de izquierda a derecha, pixel a pixel

Vamos a desplazar nuestro cuadrado de izquierda a derecha, aunque tiene truco, vamos a desplazar toda la línea.

Escribimos una rutina que desplace las 32 columnas de cada scanline donde hemos dibujado el cuadrado. Para ello vamos a usar la instrucción RR, que rota un byte a la derecha, poniendo en el bit 7 el valor del flag de acarreo y en el flag de acarreo el valor del bit 0.

A esta rutina le podríamos pasar las coordenadas X e Y, o la dirección de pantalla, pero optamos por simplificar ya que sabemos en que línea está el cuadrado.

Abrimos el fichero Scroll.asm y añadimos la siguiente rutina:


```
; ------------------------------------------------------------------------
; Scroll
; Realiza el scroll de la línea 11 (10 en base 0), en un bucle infinito.
; Altera el valor de los registros AF, BC, HL e IX.
; ------------------------------------------------------------------------
Scroll:
ld      b, $0a          ; Carga en B la coordenada Y
ld      c, $00          ; Carga en C la coordenada X. El inicio de la línea.
call    GetPointerCoord ; Calcula la posición de memoria

scroll_scan:
ld      c, $08          ; Número de scanlines que se van a mover. Un carácter completo.
scroll_col:
ld      b, $20          ; Número de columnas que se van a mover. Todas.
scroll_col_loop:
rr      (hl)            ; Rota la posición de memoria a la derecha
inc     l               ; Incrementa l, pasa a la siguiente columna
djnz    scroll_col_loop ; Hasta que llega a la última columna

jr      nc, scroll_col_nc       ; Si no hay acarreo, el bit 0 de la última columna era 0

; Hay acarreo, el bit 0 de la última columna era 1
push    hl              ; Pasa el valor del registro HL
pop     ix              ; a IX
set     $07, (ix-$20)   ; Activa el bit 7 del primer byte de la línea (primer píxel)
```

```
; Ha movido el scanline completo
scroll_col_nc:
dec     l               ; Vuelve a poner L apuntando a la última columna
ld      a, l            ; Carga la línea y columna actuales
and     $e0             ; Se queda con la línea
ld      l, a            ; Lo carga en L
inc     h               ; Pasa al siguiente scanline
dec     c
jr      nz, scroll_col  ; Hasta que llega al último scanline

dec     h               ; Vuelve a poner H apuntando al scanline 7, y el tercio original
ld      a, h            ; Carga el valor en A
and     $f8             ; Deja la parte del scanline a 0
ld      h, a            ; Carga el valor en H
halt                    ; Espera a que se refresque la pantalla. Comenta esta línea si lo quieres más rápido

; Llegados aquí, se ha desplazado todo el mensaje un pixel
jr      scroll_scan     ; Sigue con el scroll
```

En el fichero Cuadrado.asm, antes del include añadimos la llamada al scroll.


```
call    Scroll          ; Hace el scroll

include "Scroll.asm"    ; Incluye el fichero Scroll.asm para poder
                        ; llamar a las rutinas GetPointerCoord y Scroll
```

Compilamos:

```
pasmo --tapbas Cuadrado.asm Cuadrado.bas
```

Y vemos el resultado. Si quieres verlo más rápido, comenta el halt.

Por último, abrimos el fichero HolaRetroParla.asm, añadimos, antes de end $8000, include “Scroll.asm” y sustituimos la línea jr Fin por call Scroll.

```
Fin:
        call    Scroll          ; Scroll del mensaje
        ;jr     Fin             ; Bucle infinito, para que se vea bien el resultado
        ;ret
; ------------------------------------------------------------------------
; Mensaje que se va a imprimir, terminado en 0 = NULL
; ------------------------------------------------------------------------
Msg:    defm "Hola Retro Parla!", $00
; ------------------------------------------------------------------------
; Fichero donde se encuentra la rutina de scroll
; ------------------------------------------------------------------------
include "Scroll.asm"
; Directiva de PASMO para que además de incluir un cargador BASIC para el programa
; también lo ejecute al cargarlo
end $8000
```

Compilamos:

```
pasmo --tapbas HolaRetroParla.asm HolaRetroParla.bas
```

Y vemos el resultado.

Hemos finalizado Hola Retro Parla.