Ensamblador para ZX Spectrum

Hola Retro Parla

Anexo I

Contenido

Introducción

Una vez que hemos terminado con toda la materia de esta iniciación al ensamblador del Z80, más concretamente enfocado al ZX Spectrum, vamos a realizar tres pequeños programas, en los que podremos:

1. Ver la diferencia de velocidad entre ensamblador y Basic.
2. Leer el teclado.
3. Mover un byte pixel a pixel por la pantalla.

Diferencia de velocidad entre ensamblador y Basic

Al final de la página 44 del documento 02 Hola Retro Parla, se muestra el equivalente al programa que hemos realizado, pero esta vez en Basic.

Es muy posible que una vez analizado nos preguntemos, ¿Pero para que me meto a hacer las cosas en ensamblador, si con Basic es más sencillo y la velocidad es la misma?

Efectivamente, en este caso, en Basic es más sencillo y aparentemente igual de rápido.

A continuación, vamos a exponer un ejemplo con el que poder apreciar, de manera inequívoca, la gran diferencia de velocidad, en la ejecución, que hay entre el Basic y el ensamblador.

Este ejemplo lo hemos tomado prestado del Curso de ensamblador Z80 de Compiler Software, que podéis encontrar en speccy.org. En él se copian los primeros 6144 bytes de la ROM a la VideoRAM (pantalla).

El código en Basic es:

```
10 FOR I=0 TO 6143: POKE (16384+I), (PEEK I): NEXT I
```

Si ejecutamos el programa podemos comprobar que tarda, aproximadamente, 1 minuto y 4 segundos en ejecutarse.

Ahora vamos a escribir el mismo programa en ensamblador. Para copiar los primeros 6144 bytes de la ROM a la VideoRAM, vamos a usar

la instrucción LDIR. LDIR copia el contenido de la posición de memoria apuntada en HL a la posición de memoria apuntada por DE, la veces que diga BC. En cada ciclo se incrementa HL y DE.

```
org     $8000           ; Dirección dónde se carga el programa
ld      hl, $00         ; Apuntamos HL a 0, inicio de la ROM
ld      de, $4000       ; Apuntamos DE al inicio de la VideoRAM
ld      bc, $1800       ; Se van a copiar 6144 bytes
ldir                    ; Se copian 6144 bytes
ret                     ; Volvemos a Basic
end     $8000
```

Si compilamos y ejecutamos, veremos la diferencia en el tiempo de ejecución.

```
pasmo --tapbas fichero.asm fichero.tap
```

Lectura del teclado

El teclado lo vamos a leer a través del puerto $FE, diciéndole que semifila es la que vamos a comprobar.

El teclado del Spectrum está dividido en 8 semifilas, cada una de las cuales tiene un valor.

Caps Shift - V	1111 1110b ($FE)
A - G	1111 1101b ($FD)
Q - T	1111 1011b ($FB)
1 - 5	1111 0111b ($F7)
0 - 9	1110 1111b ($EF)
P - Y	1101 1111b ($DF)
Enter - H	1011 1111b ($BF)
Space - B	0111 1111 ($7F)

Como podemos observar, para pasar de una semifila a la siguiente, solo hay que hacer una rotación circular sobre el valor. Las rotaciones se verán en el siguiente ejemplo.

Para leer el teclado vamos a usar el registro A. En el registro A, una vez leído el estado de la semifila, los bits que representan a las teclas vendrán a 1 si no se han pulsado y a 0 si se han pulsado. Vamos a usar el registro D para marcar que teclas se han pulsado, QAOP/OPQA o como sería el caso AQPO.

El estado de la semifila viene en los 5 primeros bits, del 0 a 4, correspondiéndose el 0 con la tecla más alejada del centro del teclado,

y el 4 con la más cercana. Este ejemplo también está tomado prestado del Curso de Ensamblador Z80 de Compiler Software.

```
org     $8000           ; Dirección en la que se carga el programa
CompruebaA:
ld      d, $00          ; Pone a cero el registro D
ld      a, $fd          ; Carga en A el valor de la semifila A-G
in      a, ($fe)        ; Lee el estado de la semifila
bit     $00, a          ; Comprueba el estado de la tecla A
jr      nz, CompruebaQ   ; Si está a 1, no se ha pulsado y salta
set     $00, d          ; Pone a 1 el bit con el que vamos a indicar
                        ; que se ha pulsado la tecla Abajo

CompruebaQ:
ld      a, $fb          ; Carga en A el valor de la semifila Q-T
```

```
in      a, ($fe)        ; Lee el estado de la semifila
bit     $00, a          ; Comprueba el estado de la tecla Q
jr      nz, CompruebaP   ; Si está a 1, no se ha pulsado y salta
set     $01, d          ; Pone a 1 el bit con el que vamos a indicar
                        ; que se ha pulsado la tecla Arriba

CompruebaP:
ld      a, $df          ; Carga en A el valor de la semifila P-Y
in      a, ($fe)        ; Lee el estado de la semifila
bit     $00, a          ; Comprueba el estado de la tecla P
jr      nz, CompruebaO   ; Si está a 1, no se ha pulsado y salta
set     $02, d          ; Pone a 1 el bit con el que vamos a indicar
                        ; que se ha pulsado la tecla Derecha
```

```
CompruebaO:
; No es necesario volver a leer el estado de la semifila
; La P y la O están en la misma semifila
bit     $01, a          ; Comprueba el estado de la tecla O
jr      nz, Fin         ; Si está a 1, no se ha pulsado y salta
set     $03, d          ; Pone a 1 el bit con el que vamos a indicar
                        ; que se ha pulsado la tecla Izquierda


Fin:
ld      hl, $4000       ; Apuntamos HL al inicio de la VideoRAM
ld      (hl), d         ; Pintamos el estado de las teclas
jr      CompruebaA      ; Vuelta a empezar


end     $8000
```

Compilamos y probamos.

```
pasmo --tapbas fichero.asm fichero.tap
```

Moviendo un byte pixel a pixel

En este último programa vamos a mover un byte, pixel a pixel, por la pantalla. Ojo, no confundir con mover un sprite, eso es más complejo.

Para mover el byte, vamos utilizar las rotaciones. En concreto vamos a rotar el byte hacia la derecha, y con eso vamos a conseguir nuestro desplazamiento horizontal pixel a pixel.

La rotación de un byte hacia derecha, pone el valor del bit 0 en el flag de acarreo, desplaza todos los bits a la derecha, y pone el valor que tenía el flag de acarreo en el bit 7:

0010 0000 C = 1

RR

1001 0000 C = 0

Si la rotación fuera a la izquierda, el valor del bit 7 pasaría al flag de acarreo, se desplazarían todos los bits a la izquierda, y el valor que tenía el flag de acarreo pasaría al bit 0:

0010 0000 C = 1

RL

0100 0001 C = 0

Las rotaciones circulares, a las que se han hecho referencia en el ejemplo anterior, si es a la derecha pone el valor del bit 0 en el flag de

acarreo, rota los bits a la derecha y pone el valor que tenía el bit 0 en el bit 7. Si la rotación es a la izquierda, hace lo mismo pero rotando a la izquierda.

0010 0000 C = 1

RRC

0001 0000 C = 0

0010 0000 C = 1

RLC

0100 0000 C = 0

En este ejemplo también usamos la instrucción DJNZ, que hace lo siguiente:

1. Decrementa B, DEC B
2. Si B no es 0, salta a la etiqueta especificada, JR NZ, Bucle_2

Y ahora vamos a mover nuestro byte pixel a pixel.

```
org     $8000           ; Dirección en la que se carga el programa

ld      hl, $4000       ; Apuntamos HL al inicio de la VideoRam
ld      (hl), $ff       ; Pintamos 8 píxeles en la pantalla

Bucle:
ld      b, $20          ; Vamos a rotar las 32 columnas
```

```
Bucle_2:
rr      (hl)            ; Rota a la derecha el valor de la dirección
                        ; de memoria apuntada por HL
inc     hl              ; HL = siguiente dirección (columna)
djnz    Bucle_2         ; Mientras B > 0
ld      hl, $4000       ; Apuntamos HL al inicio de la VideoRam
halt                    ; Espera a la siguiente interrupción
                        ; y así ralentiza el programa.
                        ; Probad a quitarlo
jr      Bucle           ; Vuelta a empezar
```