MSX Desarrollo en español

Ejercicios curso Assembler. Unidad 1.

1. Escribe un programa que genere una tabla de 256 bytes con valores aleatorios y a continuación los ordene mediante el algoritmo de la burbuja. El programa se podrá cargar desde MSX-BASIC con el comando BLOAD y se comprobarán los resultados desde el propio BASIC usando la función PEEK para recorrer la tabla e imprimir en pantalla el resultado.

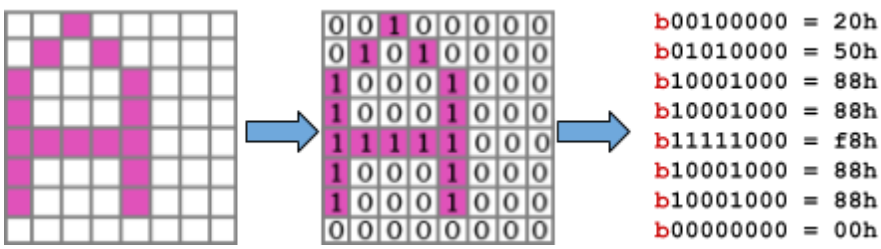
   Tips:

   a. Para obtener un número aleatorio, llamar a la función de BIOS existente en #2BDF y recoger el resultado en cualquiera de los bytes comprendidos entre #F800 y #F804.
   b. https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja

2. Escribe un programa que genere un tipo de fuente de letra idéntica a la del juego de Dinamic “AMC (Astro Marine Corps)”. Dicha fuente puede obtenerse algorítmicamente a partir de la fuente de letra habitual del MSX. El programa debe ejecutarse desde el BASIC en modo SCREEN 1 y los caracteres deben quedar redefinidos al volver al BASIC.

   Tips:

   a. Juego AMC en formato ROM para su análisis visual: https://drive.google.com/open?id=0B2CyZouCZjDhekI0clBjOW9Rb1k
   b. Usa las funciones BIOS RDVRM (004AH) y WRTVRM (004DH), que equivalen al VPEEK y VPOKE del MSX-BASIC. La dirección a leer/escribir se coloca en HL y el valor en el acumulador A.
   c. En SCREEN 1, la fuente de letra 8 x 8 se define a partir de la dirección VRAM 0, y cada carácter (siguiendo el orden de la tabla ASCII) está compuesto por 8 bytes, un byte por línea. El valor de cada byte, expresado en binario, coincide con los pixels de esa línea (8 pixels por línea), tal y como se ve en este ejemplo:



d. Las operaciones de operaciones con bits SRL y OR serán de mucha ayuda para la transformación de los pixeles.

3. Escribe un programa que mueva 32 sprites simultáneos en pantalla de tal forma que se cumplan las siguientes condiciones:

   a. Cada sprite se mueve siguiendo su propia ruta de forma individual al resto.
   b. El comportamiento de cada sprite es siempre el mismo: se moverá en incrementos de un pixel tanto en X como en Y (es decir, en cada iteración la coordenada X avanzará tal que X = X + 1 y la Y será Y = Y + 1).
   c. Cuando el sprite toque el borde inferior o superior, rebotará, es decir, cambiará el sentido (el incremento de Y se cambiará entre +1 y -1 para conseguir el cambio de dirección).
   d. Cuando el sprite toque el borde lateral (izquierdo o derecho), rebotará, es decir, cambiará el sentido (el incremento de X se cambiará entre +1 y -1 para conseguir el cambio de dirección).
   e. La posición inicial de cada uno de los 32 sprites será aleatoria.
   f. El color de cada sprite será aleatorio, de 1 a 15, evitando que ninguno de ellos adopte el color de fondo (lo que provocaría su invisibilidad).
   g. Todos los sprites serán en realidad el mismo sprite (mismo dibujo), de 8 x 8. El diseño es irrelevante, pudiendo ser de elección propia.

   Tips:

   a. Para definir un sprite de 8x8, hay que definirlo a partir de la dirección #3800 de VRAM de forma análoga a cómo se redefine un carácter (8 bytes, 1 byte por línea).
   b. Para ubicar la posición y colores de los sprites, hay que escribir en la VRAM a partir de la dirección #1B00. Cada uno de los 32 planos de sprites está definido por 4 bytes comenzando en dicha dirección:

Dirección VRAM	Número plano sprite (0-31)	Valor de byte
#1b00	0	Coordenada Y
#1b01	0	Coordenada X
#1b02	0	Número patrón sprite (usaremos 0)
#1b03	0	Color (0-15)
#1b04	1	Coordenada Y
…… (etc.)	….. (etc.)	……... (etc.)

Se recomienda consultar la siguiente documentación de referencia cuando sea necesario:

- http://www.konamiman.com/msx/msx-s.html#easymbler
- https://lookaside.fbsbx.com/file/TUTORIAL%20-%20Iniciaci%C3%B3n%20al%20Ensamblador%2C%20por%20PEPE%20VILA.pdf?token=AWwUy0UNY8gRDBVv1qDmjUHc-gBvuql5szsBvLbQGH7cXASWGSes7DDkqbpXFisepaZcbUg50J_rqRVUv2_SHzg7Et0ps0CuXCU9GS-7xTbubRoRfo-SQqfvSeW7l-DChKmHuKSF6GNmyRKOLWf03EHn
- Sección de archivos y documentación del grupo: https://www.facebook.com/groups/msxdesarrollo/files/