ARTÍCULOS **0**

Tutorial ensamblador (I) – Cómo crear una ROM de 48K

POR · PUBLICADA 05/11/2009 · ACTUALIZADO 06/11/2009

Primer tutorial de una serie de artículos orientados a la programación en ensamblador, elaborada por **Ramones**. Si queréis podéis descargar uno a uno los tutoriales desde esta entrada del blog.

1.- Introducción

Podemos empezar comentando en este tutorial que las ROMs de 48K en MSX no han sido algo habitual hasta nuestros días, exactamente hasta la aparición en la MSXDev de algunas entradas con este formato de ROM.

Realmente desconozco si existe alguna ROM de 48K de la época comercial en MSX. Si hay algunas ROMs de 64K (siempre hablando de ROMs que no hagan uso de ROM mapper), pero de 48K todavía no he sido capaz de encontrar ninguna ROM comercial.

El funcionamiento de una ROM de 48K, en general, no difiere mucho de cualquier otro tipo de ROM sin mapper. Y nos permite diferentes modos de trabajo y usos de esas 16K extra de memoria en ROM.

2.- Posicionamiento de las 16K Extra

Las 16K extra de ROM con las que contamos en un ROM de 48K «siempre» están ubicadas en la pagina 0 del Z80, es decir en el rango 0000h-03FFFh de la memoria que puede leer el micro.

Al estar ubicadas en el mismo rango de memoria que la MSXBIOS su tratamiento es algo especial, puesto que para usarla adecuadamente debemos de posicionar el slot/subslot de nuestro cartucho en esa zona, digamos, a mano.

Por desgracia el MSX, aunque cuenta con funciones en su BIOS preparadas para casi todo, no tiene soporte para manejar estas 16K. Las rutinas que manejas slots/subslots de la BIOS no son capaces de trabajar con esas 16K. Bueno, no es esto del todo cierto. Realmente la rutina que no permite «marcha atrás» es ENASLT.

3.- Posibilidades de uso de las 16K Extra

Los 16K Extra de memoria de una ROM de 48K pueden ser usados de varias maneras, sean directas o mixtas. Pongamos algunos ejemplos:

- Almacén de datos: Quizá el más común y sencillo de todos los usos es utilizar los 16K extra como Almacén de datos. Gráficos, Música, etc… pueden ser alojados en estos 16K extra. Cuando deseamos utilizarlos, lo más común es, o bien pasarlos a VRAM directamente (si son datos gráficos) o bien pasarlos a memoria y utilizarlos después (por ejemplo si fuese algo comprimido). Tambien es posible alojar música sin comprimir, pero para ello es necesario preparar el replayer para que posicione los 16K extra, reproduzca el frame de música que toque y vuelva a posicionar la BIOS.
- Código: Es perfectamente posible ejecutar código Z80 en esa página, pero aquí hay que ir con un poco más de cuidado. Sobre todo con las interrupciones. Si queremos ejecutar una

porción de código de esa página tenemos que: 1.- Hacerlo con las interrupciones deshabilitadas, 2.- Tener programado en esas 16K extra nuestro propio gancho en 038h y 3.- No usar la BIOS para nada. (Siempre hablando de trabajar en el modo natural IM1).

- Arranque de la ROM: Es perfectamente posible arrancar la ROM, posicionando la cabecera de ROM conocida en la pagina de 16K Extra. De esto existen muchos ejemplos de ROMs, que, aunque sean de 32k funcionan así. Hero, Crazy Train, son ROMs de 32k que son ubicadas en la zona 0000h-07FFFh, ejecutándose en la pagina 0 y con su propioa rutina de 038h parcheada. Ahora bien, no utilizan para nada la BIOS.

Como ejemplos palpables de todos estos tipos tenemos, como Almacén de datos ROMs como Majikazo o Operation Wolf (Extended version), y como código Universe Unknown y The Cure.

4. – Ejemplo de uso de ROM de 48K con 16K extra de Almacén de datos

En este tutorial se adjunta un ejemplo de uso de una ROM de 48K, que arranca en 04000h (como el 90% de las ROMs), y usa las 16K extra como Almacén de datos.

El programa empieza con un ORG 0, ya que, como habíamos comentado las 16K extra están ubicadas en ese rango (0000h-03FFFh). Aquí metemos una pantalla completa de MSX1 (16K de Vram) sin comprimir. Lógicamente es un ejemplo, pero en una ROM de este tipo hay que aprovechar ese espacio mejor, comprimiento, por ejemplo, estos datos.

Seguidamente ya tenemos la clásica cabecera de inicialización de una ROM. Ya en la conocida zona de 04000h-07FFFh.

Después de posicionar los segundos 16K de la ROM (terceros en el caso de 48K) en la zona 08000h-0BFFFh, previamente posicionando el puntero de pila y poniendo el MSX en modo im1 (esto es recomendable 100% puesto que otra ROM ejecutada antes que la nuestra ha podido cambiar el modo y la pila), pone el MSX en Screen 2 con sprites de 16×16 sin ampliar, y carga los datos de VRAM de la pagina 0, quedándose ya el MSX en un bucle infinito, típico de cualquier juego.

Anteriormente se buscan y guardan en memoria los puertos del VDP del MSX en cuestión, ya que la pantalla cargada NO utiliza la Bios.

5. – Slots

Las rutinas de slots del ejemplo son lo más importante aquí. Pasamos a comentarlas.

- **search_slotset**

Hace uso de la subrutina search_slot. Almacéna el slot donde se esté ejecutando nuestro cartucho, en la variable slot_var. Esto es muy importante para poder tratar luego los 16K extra. Seguidamente, haciendo uso de la rutina BIOS ENASLT, posiciona los 16K altos de nuestra ROM, aunque en este ejemplo, no se usen para nada.

- **search_slot**

Rutina recomendada en todos los TH del MSX. Busca el slot/subslot de nuestra ROM, y devuelve en el acumulador (A) el slot/subslot donde se está ejecutando nuestro ROM en formato FxxxSSPP, que es el utilizado en TODAS las rutinas de la BIOS de tratamiento de slots. Aunque no tenga nada que ver con el ejemplo, esta rutina es necesaria y 100% compatible con todos los ROMS, y nos da la seguridad de poder ejecutar nuestro ROM en cualquier slot/subslot. Esto es algo que se ha descuidado últimamente en las modernas ROMs. Y es un error «garrafal» no tener en cuenta los subslots en nuestra ROM, ya que, si se ejecuta la ROM en un expansor de slots, no funcionará jamás.

- **search_slotram**

No tiene uso en nuestro ejemplo. Pero la dejamos por si alguien se atreve a crear un ROM de 64K. Le hará falta está rutina.

- **setROMpage0**

Posiciona en la pagina 0 del Z80 (0000h-03FFFh), el slot/subslot de nuestro ROM, es decir, nos posiciona las 16K extra para poder trabajar con ellas. Esta rutina hace uso de la variable previamente Almacénada «slot_var» y d ela subrutina «setslotpage0», que posiciona cualquier slot/subslot pasado en A en formato FxxxSSPP en la pagina 0.

- **recbios**

Básicamente es idéntica a la anterior, solo que le pasamos el slot de la BIOS ROM para que la reposicione.

- **setslotpage0**

La rutina más compleja e «importante» de todo el ejemplo. Posiciona un slot/subslot pasado en el acumulador en formato FxxxSSPP en la página 0 del Z80. Esta rutina, como se ve, hace uso de todo el bajo nivel de slot/subslots, puesto que, como hemos dicho anteriormente NO ES POSIBLE usar ENASLT para posicionar algo en la pagina 0. No vamos a explicarla en profundidad en este tutorial. Esto debería de dejarse para un tutorial de manejo de slots/subslots del MSX.

6.- LOADSCREEN

Una vez vistas todas las subrutinas importantes, esta rutina no tiene mucho misterio. Posiciona la pagina 0 de nuestro ROM (los 16K Extra), y manda todo su contenido a la Vram. No es posible, como sabemos ya, usar la BIOS para ello, por lo tanto se hace uso de rutinas de manejo de Vram propias.

Importante, como se ve, que las interrupciones estén desactivadas. Y por último, reposiciona la Bios de nuevo.

7. – ¿Por qué esa manía con las interrupciones desactivadas?

Lo he repetido muchas veces en el tutorial, pero no ha sido explicado. En modo IM1 de interrupción el MSX salta a la dirección 038h cada 50 o 60 veces por segundo (según el tipo de refresco). Al posicionar las 16K extra, quitamos la BIOS que es donde se encuentra habitualmente el tratamiento de esa interrupción.

Por eso mismo, hay que desactivarlas, para evitar que el MSX salte a esa dirección, lo cual, si no tenemos nada preparado en nuestras 16K extra lo único que producirá será en un 90% de los casos un cuelgue. 🙂

8.- Otros ejemplos

Quedando este ejemplo claro, el realizar otros tipos de ROM de 48K no debería de ser complejo para el programador. En cualquier caso, dudas, consultas y sugerencias… pues al de siempre.

😉

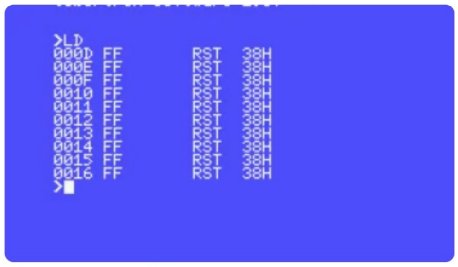
Enlace relacionado: Archivo RAR con el tutorial y archivos de ejemplo.

Si te gusta este contenido no dudes en compartirlo

Me gusta esto:

Relacionado

Tutoriales de programación en ensamblador (Ramones)
Armando Pérez (Ramones) se ha currado unos estupendos tutoriales de programación en ensamblador que serán de mucha utilidad para todos los que hemos tenido dudas de cómo
05/11/2009
En «Artículos»



Mega Assembler
27/11/2011
En «Software MSX»

Tutorial ensamblador (V) – La memoria de los MSX
Quinta parte de esta serie de interesantes artículos sobre ensamblador, creada por Armando Pérez (Ramones). La memoria de nuestros MSX es, fundamentalmente,
21/11/2009
En «Artículos»

Etiquetas: Majikazo Operation Wolf Ramones ROM de 48K The Cure Tutorial Universe Unknown

DEJA TU COMENTARIO SOBRE ESTO

Introduce aquí tu comentario...

MSXBlog
1170 Me gusta

Me gusta esta página

Sé el primero de tus amigos en indicar qu

MSXBlog
el domingo

Alberto De Hoyo estará con nosotr hablar del MSXVR. Tened en cuent antes: a las 21 horas https://www.msxblog.es/alberto-de-nosotros…/

MSXBLOG.ES
Alberto De Hoyo estará con

Charlamos con J...

junio 2020

L	M	X	J	V	S	D
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

« May

SÍGUENOS

SUSCRÍBETE

Únete a otros 5.779 suscriptores

Dirección de correo electrónico

Enviar

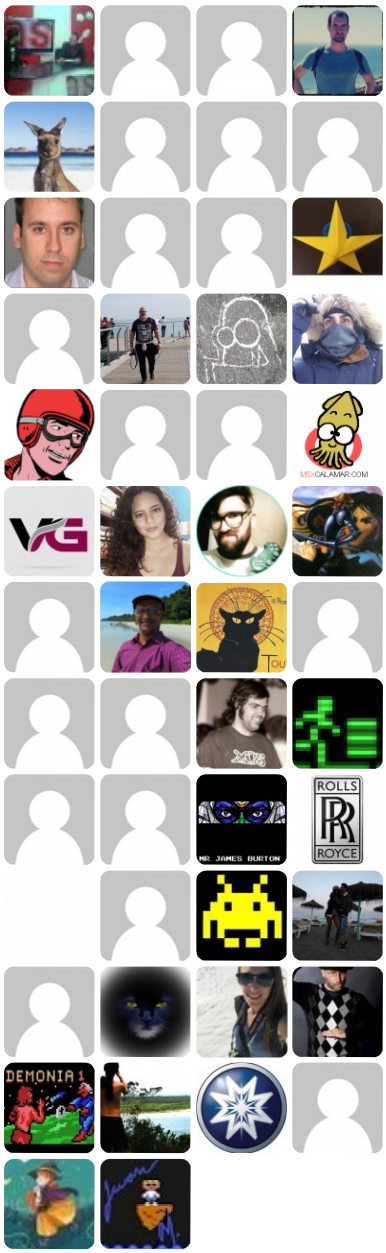
ARCHIVOS

Elegir el mes

ETIQUETAS

AAMSX Concursos Cómo terminar... Demo Dinamic Django El PixeBlog de Pedja Emuladores Encuestas Entrevista Eventos Gráficos Hardware Hideo Kojima Historia Juegos en desarrollo Juegos MSX Konami La Abadía del Crimen Libro Libros Los "Patitos Feos" del Software Español Metal Gear MSX-BASIC MSX-Club MSX Solutions Música Noticias de Konamito.com Passion MSX Penguin Adventure Preservación Programa en directo Recuerdos de 8 Bits RELEVO Videogames Remake Remakes Reuniones y ferias The Maze of Galious The Punisher Topo Soft Traducciones Tutorial Vídeo Vídeos YouTube

COMUNIDAD



Cookies y privacidad
¿Qué es esto?
Créditos
Publicaciones
Fanzines
Revistas
Libros
Manuales
Catálogos
Enciclopedias
Galerías de imágenes
Concursos