MSX Article

A Memória do MSX (I)

Resumo

O objetivo deste artigo é mostrar como funciona o esquema de memória do MSX, que possui ROMs e RAMs compartilhadas em um espaço de 64 KB. Este é o primeiro de três artigos, no qual serão abordados os slots primários. No segundo artigo, serão analisados os slots secundários e também a Mapper e Megaram. O terceiro irá apresentar uma experiência prática na memória do MSX.

1. Introdução

Um processador Z-80 consegue endereçar no máximo 64 KB de memória. Mas, se a ROM do sistema e o interpretador Basic do MSX ocupam 32 KB e a RAM do usuário ocupa 64 KB, não totaliza mais de 64 KB? E quando um cartucho é inserido, o tamanho de memória não fica maior ainda? A resposta é sim. Então, o processador Z-80 pode acessar mais de 64 KB? Sim, mas não diretamente.

O Z-80 enxerga no máximo 64 KB, devido ao fato do endereçamento de memória ser de 16 bits, ou seja, só é possível representar números entre 0 e 65535. Entretanto, é possível ter outros bancos de memória no sistema. Na arquitetura original do MSX, podemos ter até quatro bancos. Assim, para acessar outras memórias acopladas ao sistema, será necessário trocar o acesso à memória em uso para a memória que se deseje acessar. Por exemplo, o Z-80 deixa de enxergar a ROM e passa a enxergar a RAM. Entretanto, essa troca não é feita em toda a área de memória de uma só vez.

Os projetistas do MSX dividiram a memória em 4 partes iguais, chamadas de páginas. Dessa forma, é possível trocar apenas partes da memória enxergada pelo Z-80, resultando em independência de acesso em cada página. Por exemplo podemos ter uma página acessando a ROM, enquanto outra acessa a RAM. A figura 1 apresenta o esquema de páginas de memória do MSX.

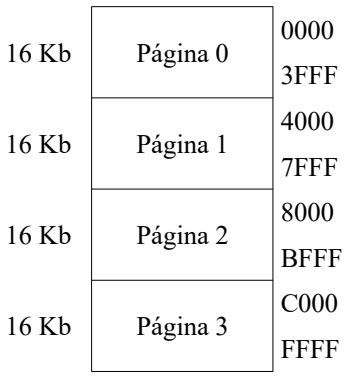

Figura 1. Divisão da memória do MSX em páginas.

As páginas são numeradas de 0 a 3, onde cada uma tem 16 KB. Os limites de endereço de cada página são mostrados na figura 1.

Quantas memórias podem ser compartilhadas no projeto original do MSX? É possível compartilhar até quatro memórias, com capacidade de até 64 KB. Cada espaço desse contendo uma memória é chamado de slot, onde o número de identificação varia de 0 a 3.

Para exemplificação, apresentamos na figura 2 a configuração de slots e páginas do Expert da Gradiente. As áreas sombreadas correspondem as memórias existentes neste modelo de MSX. Esta configuração varia nos MSXs, mas sempre com a BIOS no slot 0.

	Slot 0 ROM - BIOS	Slot 1 Cartucho A	Slot 2 RAM	Slot 3 Cartucho B	
16 Kb	Página 0	Página 0	Página 0	Página 0	0000 - 3FFF
16 Kb	Página 1	Página 1	Página 1	Página 1	4000 - 7FFF
16 Kb	Página 2	Página 2	Página 2	Página 2	8000 - BFFF
16 Kb	Página 3	Página 3	Página 3	Página 3	C000 - FFFF

Figura 2. Divisão da memória do MSX em slots e páginas.

Uma memória não necessita ocupar todos os 64 KB do slot, nem começar pela página 0. Um cartucho, por exemplo, pode ter a memória de 16 KB e localizada na página 1.

2. O chaveamento de memória

Foi dito na seção anterior, que é possível trocar a memória em uso das páginas de forma independente uma das outras. A PPI do MSX é a responsável por chavear a memória e configurar qual slot de memória o processador Z-80 irá acessar em cada página. É através da porta &HA8 que ajustamos ou obtemos a configuração de slots do MSX. A figura 3 apresenta configuração da porta &HA8.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Página	3		2		1		0	

Figura 3. Configuração dos dados enviados à porta &HA8.

O valor lido ou passado para a porta &HA8 é dividido em quatro partes, onde cada parte representa uma página da memória. Cada parte contém um valor de 2 bits, que corresponde ao número do slot no qual desejamos ativar naquela página. O exemplo a seguir ilustra como dados são configurados na porta &HA8.

	Slot 0 ROM - BIOS	Slot 1 Cartucho A	Slot 2 RAM	Slot 3 Cartucho B	
16 Kb	Página 0	Página 0	Página 0	Página 0	0000 - 3FFF
16 Kb	Página 1	Página 1	Página 1	Página 1	4000 - 7FFF
16 Kb	Página 2	Página 2	Página 2	Página 2	8000 - BFFF
16 Kb	Página 3	Página 3	Página 3	Página 3	C000 - FFFF

Figura 4. Configuração inicial de memória do Expert.

Quando o Expert da Gradiente é iniciado, as páginas de memória são configuradas de acordo com a figura 4. As páginas 0 e 1 acessam a ROM (slot 0) e as páginas 2 e 3 acessam a RAM (slot 2) – vide áreas sombreadas. Neste caso, a configuração da porta &HA8 fica:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Valor em Binário	1	0	1	0	0	0	0	0
Slot ativo	2		2		0		0	
Página	3		2		1		0	

Figura 5. Configuração da porta &HA8 para o Expert MSX da Gradiente (Brasil).

Desta forma, o valor da porta &HA8 para esta configuração é 160 (&B10100000). A porta &HA8 pode ser tanto lida, de forma a obter a configuração de slots, como escrita, de forma a modificá-los.

3. O cuidado na troca de slots

Quando uma página troca de slot, toda a região de memória relativa a essa página passa a acessar uma nova memória. Embora o conteúdo da memória anterior não seja perdido, ele se torna inacessível após a troca (a menos que este slot seja chaveado de volta). Assim, torna-se evidente que não devemos fazer o chaveamento de memória, caso o nosso programa estiver executando na mesma página que queremos trocar, sob pena do programa perder o controle de execução.

De forma a ilustrar esse fato (de forma grosseira), suponha dois programas em Basic na mesma página, mas localizados em slots diferentes. Suponha também que o Program Counter (PC) aponte para uma linha em Basic, em vez de um endereço de memória.

Programa no slot 2	Programa no slot 1

`8000 PRINT”Vou trocar o slot”`
`8010 OUT &HA8, &B10010000`
`8020 END`
	`8000 INPUT”Qual o seu nome”;N$`
	`8010 INPUT”Qual a sua idade”;I`
	`8020 PRINT”Nome”;N$`
	`8030 PRINT”Idade”;I`
	`8040 END`

Ambos os programas se localizam na mesma região de memória. Ao executar o programa do slot 2, seria obtido o seguinte fluxo de execução, assinalado em azul:

Programa no slot 2	Programa no slot 1

`8000 PRINT”Vou trocar o slot”`
`8010 OUT &HA8, &B10010000`
`8020 PRINT”Tenho que executar”`
`8030 END`
	`8000 INPUT”Qual o seu nome”;N$`
	`8010 INPUT”Qual a sua idade”;I`
	`8020 PRINT”Nome”;N$`
	`8030 PRINT”Idade”;I`
	`8040 END`

O fluxo de execução passou do programa do slot 2 para o programa do slot 1, fazendo com que o programa principal no slot 2 perdesse o controle de execução.

Caso seja necessária uma troca de slots como no exemplo acima, a solução é desviar a execução do programa para outra página e fazer a troca de slots. Uma vez que o programa passa a executar em outra página, o controle de execução não é perdido. Esta estratégia foi utilizada no programa Slot View [1].

Programa no slot 2	Programa no slot 1
8000 PRINT"Vou trocar o slot" 8010 GOTO C000 8020 END	8000 INPUT"Qual o seu nome";N$ 8010 INPUT"Qual a sua idade";I 8020 PRINT"Nome";N$ 8030 PRINT"Idade";I 8040 END A página é trocada
C000 OUT &HA8, &B10010000 C010 END A execução está em outra página - Ok	

4. As chamadas iter-slot

A BIOS é o sistema operacional do MSX, no qual contém várias instruções prontas, como escrever na tela, gerar sons e sprites, controlar joystick etc. Dessa forma, o usuário não precisa se preocupar em escrever esses códigos mais complexos, e somente focar em seu programa. A BIOS localiza-se na página 0 do slot 0.

O interpretador Basic se localiza nas páginas 0 e 1 do slot 0 e é utilizado quando o MSX opera em modo Basic, juntamente com a BIOS. Um programa em Assembly não necessita do interpretador Basic, exceto quando são feitas chamadas a funções específicas do Basic como as funções matemáticas. Assim, a página 1 fica livre para os programas.

A página 0 pode até ser utilizada por um programa em Assembly, entretanto, o acesso direto ao sistema operacional da BIOS fica prejudicado, caso seja necessário. A solução para este problema pode ser parecida com a adotada pelo MSX-DOS. Para as demais páginas, podemos utilizar as chamadas inter-slot do MSX.

A BIOS do MSX possui sub-rotinas para chamar programas em outros slots, de forma segura e eficiente. Elas são as chamadas inter-slot.

Os casos mais comuns de chamadas a programas em outros slots são [2]:

- A BIOS é chamada a partir do MSX-DOS.
- A BIOS é chamada da SUB-ROM a partir do Basic (MSX 2)
- A BIOS é chamada a partir de um cartucho.

No caso do MSX-DOS, ele configura inicialmente as quatro páginas para a RAM. Com essa configuração, o acesso direto à BIOS não é possível. Então, quando uma chamada é feita à BIOS, é necessário trocar a página 0 para a ROM da BIOS e então realizar a chamada. Após as operações da BIOS, o estado original dos slots deverá ser restaurado.

A chama inter-slot é realizada facilmente, quando o programa reside em uma página diferente de 0. Entretanto, problemas podem surgir quando o programa que realiza a chamada reside na página 0, no qual é a mesma página da BIOS. Assim, cuidados são necessários para prevenir o programa de desaparecer e gerar resultados imprevisíveis. As chamadas inter-slot resolvem esse problema, através do desvio temporário do programa

para a página 3, para a troca dos slots. Algumas chamadas inter-slots estão incluídas no próprio MSX-DOS [2].

São exemplos de chamadas inter-slot [4]:

- RDSLT (&H000C)
- WRSLT (&H0014)
- CALSLT (&H001C)
- ENASLT (&H0024)

5. O funcionamento dos cartuchos do tipo ROM

Os computadores MSX possuem, pelo menos, um slot externo, no qual podemos inserir cartuchos de jogos ou outros dispositivos como o FM-PAC. Há cartuchos do tipo [2]:

- ROM – utilizados para jogos e aplicações.
- E/S – como disk-drives ou interface RS-232.
- RAM – expansão de memória RAM.
- Expansão – dispositivos para expandir o número de cartuchos.

Normalmente os jogos feitos em Assembly de cartuchos utilizam a memória a partir da posição &H4000, ou seja, a página 1. Quando o cartucho contém um programa em Basic, ele se localiza a partir do endereço &H8000 (página 2).

O arquivo da ROM é composto por um header mais o conteúdo de software:

Header	Conteúdo de software

Os cartuchos no formato ROM possuem um header de 16 bytes. Assim, quando o sistema é iniciado, o interpretador Basic analisa o header do cartucho e inicia o programa contido na ROM, baseado nessas informações. O formato do header é descrito na figura 6.

2 bytes	ID	+ 0000
2 bytes	INIT	+ 0002
2 bytes	STATEMENT	+ 0004
2 bytes	DEVICE	+ 0006
2 bytes	TEXT	+ 0008
6 bytes	Reserved	+ 000A

Figura 6. Header do cartucho ROM.

Na tabela 1, são descritos os campos do header [2] de um cartucho ROM de MSX.

Campo	Descrição
ID	No caso de cartuchos de ROM, a identificação no formato texto é “AB” (&H41, &H42). Já os cartuchos do tipo SUB-ROM, a identificação é “CD”.
INIT	Contém o endereço inicial da rotina a ser executada, quando o cartucho é auto-executável. Caso contrário, possui o valor igual a 0.
STATEMENT	Quando o cartucho dispõe da função CALL para chamar uma rotina sua a partir do Basic, esses dois bytes contêm o endereço inicial dessa rotina. Caso não possua, o valor é igual a 0.
DEVICE	Endereço inicial da rotina de expansão de dispositivo. Caso não possua, o valor é igual a 0.
TEXT	Ponteiro para programa em Basic, quando o cartucho é auto-executável. Caso contrário, o valor é igual a 0

Tabela 1. Campos do header de um cartucho ROM.

6. Carregamento de ROMs a partir do Basic

Foi visto na seção anterior que os programas em Assembly de cartuchos do tipo ROM residem na página 1. Suponha agora que tenhamos um aquivo ROM e que desejamos rodá-lo a partir do Basic. Essa ROM terá que executar obrigatoriamente na página 1, e a partir de uma memória do tipo RAM. Devemos lembrar que o modo Basic configura as páginas 0 e 1 para ROM da BIOS e as páginas 2 e 3 para a RAM.

Para ROMs de 16 KB, a solução é carregá-la na página 2 e copiar o bloco para a página 1 da RAM, fazendo-o funcionar a partir do endereço inicial do programa ali contido (INIT). A operação de cópia do bloco e execução deverá ser feita toda em Assembly. Normalmente o código que faz isso é colocado ao final do bloco do arquivo ROM. As figuras 7 e 8 ilustram esse processo.

Na figura 7, o bloco da ROM (em amarelo) mais o programa (em verde) são carregados na página 2, e o programa responsável por copiar na página 1 é posto em execução.

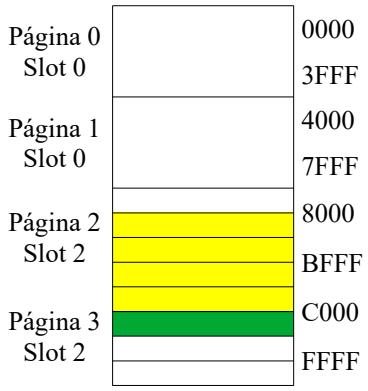

Figura 7. Carregamento de ROM na página 1.

A execução automática do programa deve ser feita através do comando BLOAD "bloco.bin", R. O header do arquivo que contém a ROM mais o programa deverá apontar o endereço de execução para o endereço inicial do bloco em verde na figura 7. Por exemplo, se o endereço inicial do arquivo for &H8200, o endereço inicial de execução será em &HC200.

Na figura 8, o bloco da ROM (em amarelo) é copiado para a página 1 da RAM e posto em execução.

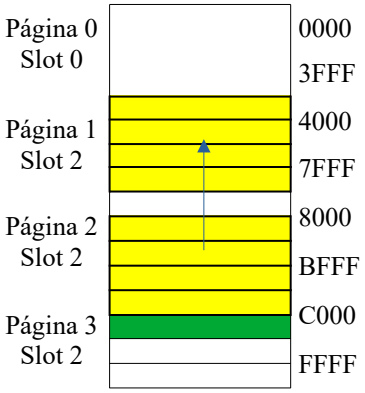

Figura 8. Processo finalizado.

Quando tivermos uma ROM de 32 KB, devemos dividi-la em blocos de 16 KB, onde para o primeiro bloco somente copiamos os dados na página 1, chaveando esta página de volta para ROM Basic. O offset em relação ao endereço &H8000 é obrigatório para este bloco, caso se deseje criar um programa em Basic para automatizar o processo. O segundo bloco pode ser carregado direto na posição &H8000, sem copiar nada, pois já está no lugar dele. Então, o programa anexado a este bloco tem que somente chavear a página 1 de volta para RAM e executar a partir da página 1. A figura 9 ilustra o processo completo de carga de ROM de 32 KB.

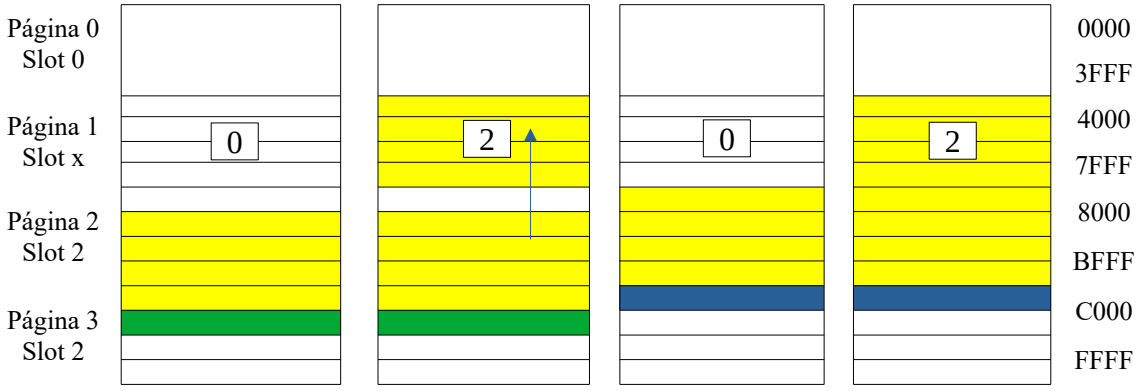

Figura 9. Carregamento de 32 KB em blocos.

O projeto MSX Digitized Sound Generator - DSG [5] apresenta um código fonte em Assembly para carga de blocos nas páginas 0 e 1, com chaveamentos de slots. Este projeto utiliza as páginas 0 e 1 da RAM como área de dados para os arquivos de sons digitalizados.

Créditos

Este artigo foi escrito por Marcelo Silveira, Engenheiro de Sistemas e Computação formado pela UERJ, Universidade do Estado do Rio de Janeiro, Brasil.

Escrito em: maio de 2004.
Revisões: Julho de 2017, Janeiro de 2019 e Junho de 2024.
E-mail: flamar98@hotmail.com
Homepage: http://marmsx.msxall.com

Referências:

[1] – Slot View, MarMSX Development, em http://marmsx.mxsall.com
[2] – MSX 2 Technical Handbook, ASCII Corporation, 1987.
[3] – O Livro Vermelho do MSX, editora Mc Graw Hill.
[4] – MSX Top Secret 2, Edison Pires Moraes, 2004.
[5] – MSX DSG, em https://marmsx.msxall.com/projetos/dsg/players.php