MSX Article

Memory Experiments

Summary

This article presents some experiments based on the articles “The MSX Memory”.

1- Introduction

Two experiments are proposed:

Experiment #1: how to copy a program block to RAM page 1.
Experiment #2: creating cartridge ROMs.

2- Experiment #1: how to copy a program block to RAM page 1

The goal of this experiment is to show how to copy a program from page 2 to page 1 and run it, once in Basic mode page 1 is set to ROM.

The next program will copy the yellow block from page 2 to page 1. The program starts running from the green part (&H9004), which is responsible for copying and running the yellow part.

Add	Assembly	Line	Label	Instruction	Commentaries
		10		ORG &H9000	; Initial address of the whole program
9000	CD C3 00	20		CALL &HC3	; BIOS CLS routine
9003	C9	30		RET	; Return
9004	DB A8	40		IN A,(&hA8)	; Read current slots configuretion
9006	47	50		LD B,A	; Copy to register B
9007	CB 3F	60		SRL A	; Apply right shift twice
9009	CB 3F	70		SRL A	
900B	B0	80		OR B	; Make slot pg 1 = slot pg 2
900C	D3 A8	90		OUT (&HA8),A	; Send data to port A8
900E	AF	100		XOR A	; Clear register F
900F	21 FF FF	110		LD HL,&HFFFF	;
9012	36 00	120		LD (HL),0	; Set sub-slot 0
9014	21 00 90	130		LD HL,&H9000	; Source block address
9017	11 00 40	140		LD DE,&H4000	; Destiny address
901A	06 04	150		LD B,4	; Counter
901C	7E	160	LOOP:	LD A,(HL)	; Read from memory to A
901D	12	170		LD (DE),A	; Copy to memory A register content
901E	23	180		INC HL	; Decrement source pointer
901F	13	190		INC DE	; Decrement destiny pointer
9020	10 FA	200		DJNZ LOOP	; B=B-1: If B>0, jump to LOOP
9022	CD 00 40	210		CALL &H4000	; Run program copied to page 1
9025	DB A8	220		IN A,(&HA8)	; After program end, read port A8
9027	E6 F0	230		AND &B1111000	; Mask to make pag 1 = slot 0
9029	D3 A8	240		OUT (&HA8),A	; Send data to port A8
902B	C9	250		RET	; Return

The program file starting address must be &H9000 (line 10), while the program executing address must be &H9004:

```
BSAVE"exp1.bin", &H9000, &H902B, &H9004
```

The yellow program's function is to clear the screen. This effect can easily noticed by anyone.

The next table explains the previous code grouped by functions.

Line	Label	Instruction	Detailed commentaries
40		LD A,&B10101000	Set page 1 to RAM. A little trick to find RAM slot and set page 1: * A = slsl0000, where sl is the RAM slot * B = A * A = 00slsl00 * B = slslsl00 OR * A = slslsl00
50		LD B,A	
60		SRL A	
70		SRL A	
80		OR B	
90		OUT (&HA8),A	
100		XOR A	Clear flag F. This is necessary.
110		LD HL,&FFFF	Forces page 1 sub-slot to be 0.
120		LD (HL),0	
130		LD HL,&H9000	Set source pointer.
140		LD DE,&H4000	Set destiny pointer.
150		LD B,4	Set counter, according to the block size.
160	LOOP:	LD A,(HL)	Copy data from page 2 to page 1.
170		LD (DE),A	
180		INC HL	
190		INC DE	
200		DJNZ LOOP	
210		CALL &H4000	Run the yellow code.
220		IN A,(&HA8)	After finishing, return page 1 to ROM.
230		AND &B1111000	
240		OUT (&HA8),A	
250		RET	Return to Basic.

This article is followed by the source and binaries of this program. The source code is compatible with RSCII assembler. They are:

- exp1.asm – source code for Macro Asemblador RSCII.
- exp1.txt – source code in text format.
- exp1.bin – binary. Run it on Basic environment adding the “,r” option.
- exp1.bas – program in Basic including the binaries.

In order to run this experiment in Basic, use the following program:

```
10 FOR E=&H9000 TO &H902B
20 READ A$
30 A = VAL("&h"+A$)
40 POKE E,A
50 NEXT E
60 DEFUSR=&H9004 : X=USR(0)
```

```
70 DATA CD,C3,00,C9,DB,A8,47,CB,3F,CB,3F,B0,D3,A8,AF,21
80 DATA FF,FF,36,00,21,00,90,11,00,40,06,04,7E,12,23,13
90 DATA 10,FA,CD,00,40,DB,A8,E6,F0,D3,A8,C9
```

2.1- Testing on blueMSX emulator debugger

The blueMSX emulator brings an excellent tool to inspect the MSX instructions set, memory and registers while the MSX runs. This is the debugger. The debugger is accessed through the Tools option, located on the blueMSX top menu. Figure 1 presents the blueMSX emulator and the debugger tool.

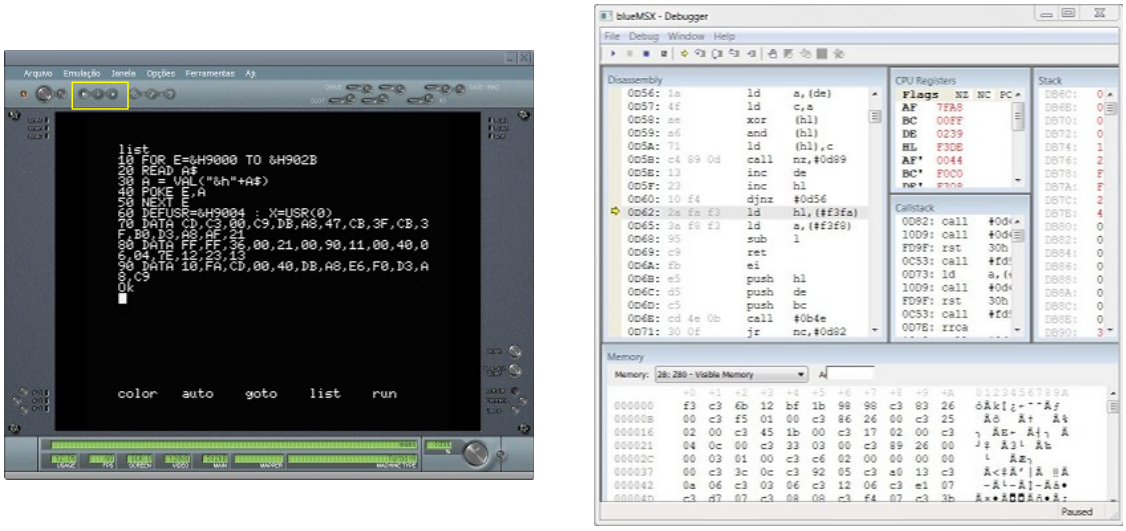

Figure 1. blueMSX emulator and debugger tool.

After opening the debugger, the reader will notice that no data is modified. In order to inspect the MSX content, the emulator must be paused (middle button in the yellow rectangle signed on figure 1).

Before using the debugger, we must load the program “exp1.bin” or “exp1.bas” on the MSX memory, without running it. Use BLOAD without “,r” (binary) or LOAD (Basic).

The debugger allows the step by step instruction execution, as well as creating breakpoints, which stop program's execution when such point is reached. We will create a breakpoints at &H9004, &H9014 and &H9022, where the first one is the green program starting address. But before that, let's see how to set the debugger.

Preparing the debugger: pause the emulator, then click on the debugger window to activate it an then press “control + G”. A dialog window is opened, asking for the address to be shown at “Disassembly” window. Type 9004. To add a breakpoint, click on the left side if the desired address. A red dot confirms the breakpoint. Click once again to remove the break point.

Starting the experience: after preparing the debugger, run the program. Click on play button (right button on yellow rectangle), and then proceed according to the chosen file:

- exp1.bin → `DEFUSR=&H9004 : X=USR(0)`
- exp1.bas → `RUN`

As soon as the program starts, the MSX emulator will stop executing at address &H9004 (imediatelly before), as seen o figure 2.

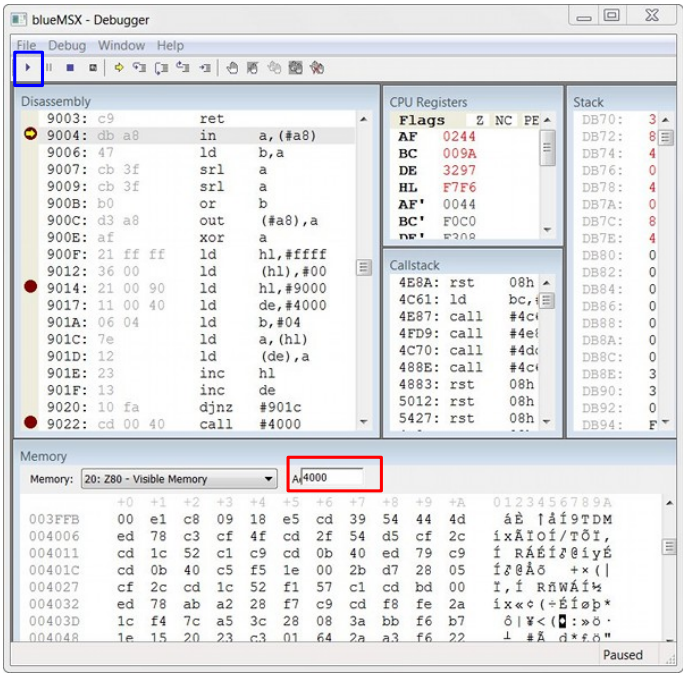

Figure 2. Program starting to run on debugger.

Pay attention to the red dot on the left side of address &H9004. There is a yellow arrow inside it, indicating the next instruction to be run. The “Disassembly” windows shows the memory address, machine code and mnemonics. Can you see our program there?

In order to change the memory location on “Memory” window, type 4000 in the box detached in the red rectangle on figure 2. Notice that this area remains as ROM.

It is possible to run each instruction step by step using F11 key and follow any changes on Z-80 registers. Thus, let's run automatically to the next breakpoint.

The first part of the program will change page 1 from ROM to RAM. To run directly from &H9004 to &H9014, click on debugger's “play” button (blue rectangle). So, the page 1 is changed as seen on figure 3.

The next part, from &H9014 to &H9022, the program will copy the green code to the memory region between &H4000 and &H4003. Figure 4 shows it.

At last, run the last part of the program.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A
003FFB	00	e1	c8	09	18	e5	cd	39	54	44	4d
004006	ed	78	c3	cf	4f	cd	2f	54	d5	cf	2c
004011	cd	1c	52	c1	c9	cd	0b	40	ed	79	c9
00401C	cd	0b	40	cd	f5	1e	00	2b	d7	28	05

a) page 1 in ROM mode

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A
003FFB	00	e1	c8	09	18	ff	ff	ff	ff	ff	ff
004006	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff
004011	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff
00401C	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff

b) page 1 in RAM mode

Figure 3. Page 1 slot configuration.

```
         +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A
003FFB   00 e1 c8 09 18 cd c3 00 c9 ff ff
004006   ff ff ff ff ff ff ff ff ff ff ff
004011   ff ff ff ff ff ff ff ff ff ff ff
00401C   ff ff ff ff ff ff ff ff ff ff ff
```

Figure 4. Program copied to page 1.

3- Experiment #2: creating cartridge ROMs

As seen on the “The MSX Memory” first article, a cartridge ROM has a 16-byte header with some important data used by the system to manipulate it. In addition, the ROMs can be of two types: Assembly, starting at &H4000, and Basic, starting at &H8000.

3.1 - ROM in Basic Adapted from [1]

The first step to create a Basic ROM type is to change the initial address from a program in Basic, in order to introduce the header just before this program. Also, the first byte must be 0.

```
POKE &HF676,&H11 : POKE &HF677,&H80 : POKE &H8010,0 : NEW
```

Notice: run the previous instruction in one line.

The TEXT parameter will point to &H8010 and the program will start at &H8011.
The next step is to create the ROM header.

```
10 AD = &H8000
20 FOR I = 0 TO 15
30 POKE AD + I, 0
40 NEXT I
50 POKE &H8000,ASC("A")
60 POKE &H8001,ASC("B")
70 POKE &H8008,&H10
80 POKE &H8009,&H80
```

Modify the TEXT

After filling the header, just create or load the Basic program which will be added to the ROM. Even after the instruction NEW, the starting address remains at &H8011.
Create a file, by saving the address from &H8000 to &HBFFF. This is the ROM.

Obs: the saved file in Basic mode will contain a 7-byte header. In order to remove the MSX file header, use the noheader [2] program or any hexadecimal editor.

3.2 - ROM in Assembly

The following program was taken from the MarMSX Development Assembly course [3], and will be used on our Assembly ROM cartridge.

Add	Assembly	Line	Label	Instruction	Commentaries
		10		ORG &H4010	; Program staring address
4010	CD C6 00	20		CALL &H6C	; Set screen 0 (INITXT)
4013	11 00 00	30		LD DE,0	; VRAM address
4016	21 21 40	40		LD HL,NOME	; Phrase initial address
4019	01 0A 00	50		LD BC,10	; String size
401C	CD 5C 00	60		CALL &H5C	; Call print on screen rountine
401F	18 FE	70	AQUI:	JR AQUI	; Halt
4021 - 402A	4F 20 4D 53 58 20 76 69 76 65	80	NOME:	DEFM “O MSX vive”	; Phrase

The INITXT (line 20) is necessary to change the screen mode after the MSX boot.

Any tool can be used to generate the binary code from the Assembly code. Thus, it is necessary to reserve 16 bytes to the header. This header must contain the “AB” (values &H41 and &H42) plus the program stating address at &H4010 on the two following bytes. The program must be placed after the header. Create a file size of 16 KB.

The file “omsxvive.rom” initial data is then:

```
0000 41 42 10 40 00 00 00 00 00 00 00 00 00 00 00 00  | AB.@............
0010 CD 6C 00 11 00 00 21 21 40 01 0A 00 CD 5C 00 18  | .l....!!@....\..
0020 FE 4F 20 4D 53 58 20 76 69 76 65 FF FF FF FF FF  | .O MSX vive.....
```

4- Credits and references

This article was originally written in portuguese and translated into English by Marcelo Silveira.

Written in: May, 2004.
Revised in: July, 2017.
E-mail: flamar98@hotmail.com
Homepage: http://marmsx.msxall.com

References:

[1] - MSX 2 Technical Handbook, ASCII Corporation, 1987.
[2] - Noheader, Tools, MarMSX Development em http://marmsx.msxall.com
[3] - Curso de Assembly, Tools, MarMSX Development em http://marmsx.msxall.com