MANUAL DE OPERAÇÃO - MEGA ASSEMBLER
Cibertron Software - 1987 - 1º Edição

Manual digitado por
Adriano C. R. da Cunha
A&L Software 1998/1999

Introdução

O MegaAssembler consiste de um cartucho de 16K de memória ROM, com um programa editor assembler/desassembler/ monitor e algumas ferramentas para programação em BASIC chamadas pela função CALL. Com este programa você tem acesso a todos os slots do micro, incluindo cartuchos e interfaces ligadas ao sistema.

Instalação e execução

O cartucho do MA (Mega Assembler) deve ser conectado com o micro desligado ao slot 1 de seu micro (slot A do Expert e slot superior do HotBit) para que entre em execução antes de qualquer cartucho ou interface ligada ao micro. Se esse slot não estiver disponível, o MA pode ser instalado em qualquer outro slot.

Ao ligar o micro com o MA no slot 1, este entrará em execução automaticamente. Se houver um cartucho no slot 2, este poderá ser examinado através do MA. Para que o controle do micro seja transferido para o BASIC ou para o DOS, basta entrar com o comando BA. Para que o MA não entre em execução automática basta manter a tecla CONTROL pressionada ao ligar o micro.

Para entrarmos no MA estando no BASIC contamos com dois comandos: _START (CALL START) e _ASM (CALL ASM).

_START inicializa todas as variáveis do MA e deve ser utilizado nas seguintes condições:

1. Quando entramos pela primeira vez no MA através do BASIC.
2. Quando retornamos ao BASIC após sairmos do DOS.
3. Quanto utilizarmos algum comando de disco via BASIC.

_ASM só deve ser utilizado quando sairmos do MA para o BASIC e não utilizarmos os comandos de disco, pois neste caso os registradores são modificados.

A versão 1.0a do MA, modificação feita pela A&L Software, não possui o inconveniente de perder o programa-fonte ao se utilizar algum comando de disco. Com isso, o ítem 3 nao se aplica a esta versão.

Estrutura do Mega Assembler

Para efeito de utilização podemos dividir o Mega Assembler em duas partes:

- a primeira é constituída pelo monitor e pelo assembler, e será chamada de EMA.
- a segunda é o próprio BASIC, que chamará os comandos do MA através da instrução CALL.

O Editor do Mega Assembler

O EMA trata os números de duas maneiras: Os comandos do monitor vão interpretar todos os dados numéricos como se estivessem na base hexadecimal; já os comandos do assembler vão interpretar os números como se estivessem na base decimal. Segue uma descrição de todos os comandos do MA, conforme a estrutura abaixo:

Sinal	Significado
end	- Toda menção a endereços de memória é feita em hexa
[...]	- Argumento opcional
<...>	- Explicação sobre argumento
xx/yy	- Ou xx ou yy, escolhido pelo usuário
Qualquer outro caracter	- Constante no argumento do comando

O Monitor

A maioria dos comandos é afetada pelo comando PAGE. Caso algum comando não seja afetado então será indicado por um astericso (*). A configuração dos slots para estes comandos é: Página 0 (0 a 3FFF) - slot 0 (ROM) Página 1 (4000 a 7FFF) - o slot no qual esteja o MA, Páginas 2 e 3 (8000 a FFFF) - slots que contenham a memória RAM do micro (slot 2 para Expert e slot 3 para HotBit).

Os comandos do Monitor

```
-PAGE
 PAGE
 PAGE?
 PAGE [[<slot>][,[<slot>][,[<slot>][,<slot>]]]]
```

Seleciona a disposição das páginas 0 a 2 nos slots 0 a 3.
Se nenhum argumento for colocasdo, o micro irá colocar todas as páginas no slot que contenha a RAM.
Se o argumento for um '?', o micro mostra a disposição atual das páginas.
Se o argumento contiver de um a quatro números entre 0 e 3 separados por vírgulas, será adotada a disposição determinada para as páginas. O primeiro argumento é o slot da página 0, o segundo o da página 1, etc. A página 3, apesar de aceitar um argumento, nunca será alterada, permanecendo sempre no slot que contém a RAM.

Exemplo: PAGE 0,0 seleciona as páginas 0 e 1 de memória de 0 a 7FFF para o slot 0.

-DM
DM <end>[,<dsloc>]

Display & Memory edit. Este comando dá um display de 128 bytes em ASCII e em hexa da memória.
O primeiro argumento é o endereço inicial de edição. O segundo, o deslocamento usado. Se não for determinado assumirá zero. O deslocamento serve para criptografar ou descriptografar um programa. Deve estar entre -7F e 80.
Ao ser dado o comando, o display será mostrado e estarão disponíveis as seguintes teclas:

Cursores	-Move o cursor pela memória.
<select>	-Seleciona edição em ASCII ou Hexa.
<RETURN>	-Sai do comando.
<ESC>	-Retrocede 128 bytes.
<TAB>	-Avança 128 bytes.
0-F	-Entram com um dado em hexa (no modo hexadecimal).
Qualquer caracter	-Entram com um dado em ASCII (no modo ASCII).

Exemplo: DM 4000,2 irá mostrar o conteúdo da memória a partir do endereço 4000 acrescentando dois ao valor de cada byte.

-ZAP
ZAP <setor inic>[,<dslc>]

Este comando é um editor de setores de disco. Sua estrutura é idêntica … do comando DM, exceto que aqui não se têm endereços, mas sim setores (o setor inicié indicado pelo primeiro argumento) e offset, que é a posição do cursor no setor(essa posição vai de 0 a 1FF.Além disso, quando o micro vai mudar de setor, e le pergunta se é para gravá-la, <RETURN> responderá sim. Se você deseja gravar o setor sem ter que mudar para outro, é só apertar <CTRL>+W. <RETURN> faz voltar … edição.

Exemplo: ZAP 5 mostrará o setor onde se encontra o diretório dos discos de 180k e 360k.ZAP 10 mostra o setor 10.

-SCR
SCR <endinic>,<dx>,<dy>[,<modo>]

Este comando mostra um display gráfico da memória. A memória é apresentadaa partir de <endinic> em blocos de 8 bytes, dispostos verticalmente, que por sua vez são apresentados em blocos com dimensão determinada por <dx> e <dy>. Esses blocos podem ser dispostos horizontalmente (quando <modo> = 0), ou verticalmente (quando <modo> = 1). Se <modo> não for especificado, assumirá 0. Ao entrar no comando, estarão … disposição as seguintes teclas:

Cursores:

esq. e dir.	-Avança ou retrocede 1 byte na memória.
cima e baixo	-Avança ou retrocede 1 bloco <dx>,<dy> na memória.
<RETURN>	-Entra no modo de edição.
<TAB>	-Mostra o endereço atual.
<ESC>	-Liga ou desliga a moldura 2x2.
<CTRL+STOP>	-Sai do comando.

Ao entrar no modo de edição, um quadrado ampliando o bloco em destaque será mostrado junto com um cursor. Nesse modo as seguintes teclas estão disponíveis:

Cursores	-Movem o cursor.
<ESPAÇO>	-Inverte o ponto sob o cursor.
<RETURN>	-Sai do modo de edição.
<CTRL+STOP>	-Cancela modificaç?es feitas.
<I>	-Inverte bloco 2x2.
<SHIFT+HOME>	-Apaga bloco 2x2.

Exemplo: SCR 1BBF,1,1 mostra um display da memória, de um por um caractere a partir do endereço 1BBF
SCR 4000,2,3,1 mostra um display gráfico da memória de um bloco de dois por três caracteres a partir do endereço 4000 no formato sprite.

-SH
SH [<end>],<bt>[,[<bt>] ... [,[<bt>]]]
SH [<end>],'<string>

Este comando serve para a busca de um texto ou conjunto de bytes pela memória.
O primeiro argumento, <end>, indica o endereço inicial da procura. Caso se queira continuar uma procura, retirar esse argumento. Nesse caso será usado o primeiro endereço após o último encontrado.
Caso o argumento seja uma seqüência de números, esta seqüência será procurada. Quando duas vírgulas são encontradas sem nenhum número entre elas, o computador entenderá que aquele número pode ser qualquer um. Caso a seqüência seja encontrada, seu endereço será mostrado.
Se o segundo argumento for um apóstrofe seguido de uma seqüência com, no mínimo, duas letras, esta seqüência será procurada na memória. Caso a seqüência seja encontrada, será mostrado seu endereço junto com o deslocamento que essa string possa ter.

Exemplo: SH 4000,2A,40,0C pesquisa a ocorrência da seqüência 2A,40,0C na memória a partir do endereço 4000.
SH 4000,2A,,0C pesquisa na memória a ocorrência de uma sequência de três bytes, sendo o primeiro 2A e o terceiro, 0C.
SH 3F41,'teste' pesquisa a ocorrência do string teste a partir do endereço 3F41.

-MS
MS <end>,[<dslc>],'<string>

Grava a partir do endereço de memória indicado por <end>, a string <string>, com o deslocamento <dslc>.
Se <dslc> não for indicado, assumirá zero.

Exemplo: MS 9A15,20,'nome coloca a string nome na memória a partir do endereço 9A15 com um deslocamento de 20.

-LOAD,B
LOAD <filename>,B

Carrega na memória um bloco em binário do periférico especificado. O <filename> pode ter os seguintes formatos:

NOME -Usa o periférico corrente.
CAS:NOME -Usa o cassete.
DRIVE:NOME -Usa o disco. Caso não haja disco o micro usará o cassete.
Caso o programa não exista ou não seja binário, uma mensagem de erro será emitida.

Exemplo: LOAD BLOCO.BIN,B carrega o arquivo BLOCO.BIN para a memória

-SAVE
SAVE <filename>,<endi>,<endf>,[<ende>]

Grava um bloco de memória em formato binário no periférico especificado. O parâmetro <filename> pode ter os mesmos formatos do comando LOAD. <endi> e <endf> especificam o início e o fim do bloco a ser gravados. <ende>, endereço de entrada, se não especificado, será igual a <endi>. O programa gravado por esse formato também poderá ser lido pelo BASIC.

Exemplo: SAVE ROM.BIN,0,7FFF,4000 grava um bloco de memória do endereço 0 até 7FFF com o endereço de execução 4000 no periférico corrente com o nome ROM.BIN.

-M (*)
M [<end>]

Edição de memória. Este comando edita memória a partir do endereço <end>. Se <end> não for especificado, será usado o último endereço editado. O comando mostrará o endereço atual, seu conteúdo e o cursor. Teclas de edição:

<ESPAÇO> -Avança um endereço.
<BS> -Retrocede.
<RETURN> -Sai do comando.
0-F -Entram com um valor em hexadecimal.

Exemplo: M A000 entra com valores em hexadecimal a partir do endereço

A000.

-S (*)
S [<end>]

Igual ao comando M, porém um teclado numérico reduzido foi simulado no teclado, de acordo com o esquema:

Teclado:	Equivale a:
7 8 9 0	7 8 9 A
U I O P	4 5 6 B
J K L Ç	1 2 3 C
M , . /	0 F E D

Exemplo: S A000 entra com valores em hexadecimal a partir do endereço A000.

-C
C <modo>

Especifica modo de display para os comandos D, P e V. O modo de display é selecionado por <modo>, que pode estar entre 0 e 3, segundo a seguinte tabela:

<modo>	Formato
0	-Apresentação em hexadecimal e ASCII, em linhas com 4 bytes cada.
1	-Idem, porém em linhas com 16 bytes (para 80 colunas ou impressora)
2	-Apresentação em hexadecimal, em linhas com 8 bytes, apresentando no final de cada linha a somatória dos bytes da linha, somada a parte menos significativa do endereço da linha.
3	-Idem, porém sem somar a parte menos significativa do primeiro endereço da linha.

Exemplo: C1

-D
-P
-V
D <endinic>[,<endfim>]
P <endinic>[,<endfim>]
V <endinic>[,<endfim>]

Estes três comandos dão um display de memória num periférico, segundo o formato especificado pelo comando C. As diferenças entre um comando e outro são dadas a seguir:

Comando:	Descrição:
D	-Dá um display da memória no vídeo.
P	-Dá um display da memória na impressora.
V	-Dá um display da memória VRAM na impressora.

Para todos os três comandos acima descritos, <endinic> indica o endereço onde deve-se iniciar a listagem e <endfim> indica o endereço final. Caso <endfim> não seja especificado, serão mostrados apenas 16 bytes.

Exemplo: D 5000,6000 - mostra o conteúdo da memória,do endereço 5000 até 6000, no vídeo.
P 5000,6000 - mostra o conteúdo da memória,do endereço 5000 até 6000, na impressora.
V 3800,3FFF - mostra o conteúdo da VRAM, do endereço 3800 até 3FFF, na impressora.

-T
T <endinic>,<endfim>,<enddest>

Transfere um bloco de memória contido entre os endereços <endinic> e <endfim>, inclusive, para o bloco de memória iniciado por <enddest>.

Exemplo: T 4000,7FFF,8FFF copia um bloco de memória do endereço 4000 ao endereço 7FFF para o endereço 8000.

-F
F <endinic>,<endfim>,<byte>

Preenche um bloco de memória desde o endereço <endinic> até <endfim> com o conteúdo <byte>.

Exemplo: F 8000,C000,FF preenche o bloco de memória compreendido entre os endereços 8000 e C000 com FF.

-G (*)
G <endinic>[,<brkpnt1>[,<brkpnt2>]]

Começa o processamento de um programa na memória iniciando no endereço <endinic> e carregando os registradores com o valor guardado pelo comando X. <brkpnt1> e <brkpnt2> são os endereços

de "breakpoint", ou seja, quando um desses endereços for executado o programa pára a execução e retorna ao EMA mostrando a disposição dos registradores no momento do "breakpoint". Para retornar ao EMA os slots devem estar na disposição ROM - EMA - RAM - RAM, e deve-se dar um salto para o endereço 4010 (JP 4010).

Exemplo: G 8000,80E0 executa uma rotina a partir do endereço 8000, retornando ao EMA ao atingir o endereço 80E0.

-X
X [<reg>]

Se não houver argumento, mostra todos os registradores, caso contrário entra no modo de edição de registradores a partir do registrador <reg>, que pode ser A, F, B, C, D, E, H e L para os registradores com esses mesmos nomes, e X, Y e S para os registradores IX, IY e SP, respectivamente. Ao ser mostrado o registrador, digite o novo conteúdo que ele deve assumir (dois dígitos hexadecimais ou quatro dígitos hexadecimais, respectivamente, para registradores de um ou dois bytes. O próximo registrador será mostrado. Se quiser parar, tecle <RETURN> no lugar de um número.

Exemplo: X mostra todos os registradores.
X A altera os registradores a partir do A.

-R
R [<offset>]

Lê de fita um programa gravado pela opção 'I' do comando A e o coloca no endereço de início do header mais o offset <offset>. Caso <offset> não seja especificado, será feita a gravação pelos endereços originais.

Exemplo: R

-L
L[P] [<endinic>[,<endfim>]]

Disassembla o conteúdo da memória do endereço <endinic> até <endfim>. Caso <endfim> não seja especificado, serão dassembladas dez linhas. Se <endinic> não for definido será desassemblada a memória a partir do último endereço listado pelo comando L. A opção P envia a listagem dos mnemŽnicos para a impressora. <CTRL+STOP> interrompe a disassemblagem.

Exemplo: L 7421 - disassembla a memória a partir do endereço 7421.
L 417F,5000 - disassembla a memória a partir do endereço 417F até 5000.

Programas em Assembly

Com o Mega Assembler é possível gerar programas em fonte Z80 como se fosse em BASIC, ou seja, digitando diretamente as instruções. Isto permite que possamos corrigir o programa antes de compilá-lo para código de máquina. As linhas podem ser editadas como se fossem em BASIC, ou seja, podemos mover o cursor por toda a tela e também utilizar as teclas INSERT, DELETE, BS, GRAPH (ou LGRA), CODE (ou RGRA) e CONTROL. Cada linha é dividida em 5 setores, que a comp”em da seguinte forma:

NN Label: instrução operando ;comentário

Onde
NN: 'E o número da linha (é obrigatória).
Label: Indica determinada instrução ou posição de memória.
Instrução: Qualquer instrução válida do Z80 ou pseudo-instrução.
Operando: Valor ou código que é necessário para a instrução.
Comentário: Qualquer observação sobre o programa. Pode estar em qualquer posição da linha.

As pseudo-instruções que estão disponíveis são:

ORG Indica em que posição de memória deve ser gerado o programa objeto.
Sintaxe: ORG endereço (*)
DEFB Coloca o byte(*) especificado na mória.
Sintaxe: DEFB byte(*)
DEFW Coloca a palavra(*) de dois bytes especificada na meória.
Sintaxe: DEFW palavra(*)
DEFM Coloca texto na memória.
Sintaxe: DEFM 'texto'
DEFS Reserva espaço na memória com o número de bytes indicado.
Sintaxe: DEFS número de bytes
EQU Indica que o label desta linha cor-

responderá ao valor indicado.
Sintaxe: Label: EQU endereço (*)

(*) Podemos definir os par*ƒ*metros para os comandos e pseudo instruções de várias maneiras:
byte: pode ser um número, um caractere entre apóstrofos (') ou um label onde esse byte se encontra.
palavra: pode ser um número de dois bytes ou um label.

Todos os números podem ser expressos em uma das tr^es bases:

1 Decimal:	Basta digitar o número.
2 Hexadecimal:	Um número em hexa seguido de "H". OBS: Se o primeiro dígito for uma letra, então deve ser precedido de 0 (zero).
3 Binário:	Número seguido da letra "B".

O Assembler

Nesse modo, os comandos servem para criar, editar e compilar um programa em assembly. Como já mencionado anteriormente, os números serão tratados como sendo na base decimal, salvo especificação em contrário. Todos os comandos que tenham alguma saída pelo vídeo ou impressora podem ser abortados por <CTRL+STOP> ou interrompidos temporariamente por <ESPAÇO>

Os comandos do Assembler

-NEW
NEW

O comando NEW simplesmente apaga o programa-fonte existente na memória do EMA.

-AUTO
AUTO [<lininic>[,<incr>]]

O comando AUTO gera numeração automática das lionhas. A primeira linha a ser gerada terá o número <lininic>. As linhas seguintes serão incrementadas de <incr>.Para interromper a numeração pressione <CTRL+STOP>.Se <lininic> ou <incr> não forem indicados assumirão valor dez (10).

-LIST
-LLIST
LIST [<lininic>[-<linfin>]]
LIST [<lininic>[-<linfin>]]

Os comandos LIST e LLIST listam o programa-fonte existente na memória no vídeo ou na impressora, respectivamente. A seguir descrevemos a sintaxe do comando LIST, que se aplica também para LLIST.

LIST	Lista todo o programa.
LIST	Lista a linha .
LIST -	Lista da linha até o fim do programa.
LIST -<lf>	Lista da linha até a linha <lf>.

-DELETE
DELETE <lininic>[-<linfin>]

Apaga linhas do programa-fonte. Caso <linfin> não seja especificada somente <lininic> será apagada, caso contrário serão apagadas as linhas entre <lininic> e <linfin>, inclusive.

-RENUM
RENUM [<novali>[,<antigali>[,<incr>]]]

Renumera as linhas do programa-fonte. <novali> indica qual deve ser o novo número da primeira linha do trecho a ser renumerado. <antigali> indica o número da linha do programa a partir da qual deve-se iniciar a renumeração. <incr> indica o incremente entre as linhas. Se nenhum par*ƒ*metro for fornecido todo o programa será renumerado com nova linha inicial 10 e incremento de 10.

-FILES
FILES

Mostra o diretório do disco corrente. Se não houver drive conectado simplesmente pula duas linhas.

-LOAD
LOAD <filename>

Carrega um programa-fonte do periférico especificado. Os formatos de <filename> são os mesmos descritos para LOAD,B.

Exemplo: LOAD CACO carrega de fita um programa-fonte em pseudo-ASCII previamente gravado pelo comando SAVE CACO.

-SAVE

SAVE <filename>

Grava um programa-fonte no periférico especificado.Os formatos para <filename> são os mesmos descritos no comando SAVE anteriormente. O formato do programa-fonte é próprio do Mega Assembler, não sendo compatível com outros assembladores nem com o BASIC.
O programa ConvASM,da A&L Software,distribuído gratuitamente, permite a conversão de programas-fonte do Mega Assembler para o formato ASCII e vice-versa.

Exemplo: SAVE CACO grava o programa-fonte em fita no formato pseudo-ASCII.

-MERGE
MERGE <filename>

Intercala dois (2) programas, um na memória e outro na fita, gravado em formato pseudo-ASCII. No caso de existir coincidência do número das linhas, a existente na memória será apagada, prevalecendo a linha lida da fita. equivalente ao comando MERGE do BASIC.

-SEARCH
SEARCH <string>

Procura por todo o programa a string <string> e lista as linhas do programa-fonte que a contém. Note que os espaços entre o comando e o texto da string serão contados.

Exemplo: SEARCH LO procura a string " LO".

-LSEARCH
LSEARCH <string>

Idêntico ao comando SEARCH, porém as linhas onde for encontrada a string serão listadas na impressora.

Exemplo: LSEARCH@LO procura a string "@LO".

-FIND
FIND <string>

Similar ao SEARCH, porém procura a string no início de cada linha.Tem a vantagem de ser mais rápido.

Exemplo: FIND MENS procura a string " MENS".

-CHANGE
CHANGE '<string1>'[<string2>]

Troca as ocorrêcias de <string1> no programa-fonte por <string2> e lista as linhas após a troca.Se <string2> for nula, apaga as ocorrência de <string1>.

Exemplo: CHANGE 'LO'@@CA

-MAP
MAP

Mostra os endereços inicial e final do programa-fonte contido na memória.

-A
A [NUPOIRSDH/<offset>]

Assembla (monta) o programa-fonte contido na memória usando as opções do usuário, que podem ser:

N	-Não lista o número das linhas.
U	-Não lista o programa.
P	-A listagem sairá na impressora.
O	-Gera o código-objeto.
I	-O código-objeto será armazenado em fita para ser lido pelo comando R.
R	-Mostra uma listagem em referência cruzada dos labels após assemblar o programa.
S	-Gera uma listagem em ordem alfabética dos labels após assemblar o programa.
D	-Gera uma listagem dos labels após assemblar o programa.
H	-Lista na impressora os labels.
/<offset>	-Assembla o programa para o endereço indicado pela pseudo-instrução ORG gerando o código-objeto no endereço dado pelo ORG mais <offset>.

Durante o processo de montagem podem ocorrer erros. Caso haja erro, a(s) letra(s) correspondente(s) ao(s) erro(s) na linha será(ão) listada(s) no início desta. Essa(s) letra(s), e seu(s) significado(s), está(ão) na tabela a seguir:

D	Número relativo menor que -128 ou maior que 128. Ocorre em instruções 'JR' e em instruções que usam IX e IY de modo indexado.
F	Erro de sintaxe. Linha muito grande,um

erro na escrita ou estrutura errada.
M Label não única.Quando você já usou um label com o mesmo nome.
U Label desconhecido.Você tentou usar um label não definido.
Q Uma instrução não existente foi utilizada.
O Operando inadequado … instrução.

Caso não seja especificado o ORG e você tente montar o programa usando a opção 'O', a mensagem 'Falta memória' será gerada. Esse mesmo erro ocorre caso o endereço do programa seja menor que 4000 ou ultrapasse EBFF.

Exemplo: se em um programa a pseudo-instrução ORG indicar 0C000h e o programa for assemblado pelo comando AO/1000, o código-objeto será alocado no endereço D000.

-BASIC
-BA
BASIC
BA

Sai do EMA (retorna ao BASIC desde que você não altere variáveis do sistema utilizadas pelo interpretador).

O BASIC

O Mega Assembler,além do assembler e do monitor, fornece ao usuário algumas ferramentas para o BASIC. Essas ferramentas foram concebidas para proporcionar ao usuário uma facilidade maior para operar seu computador MSX. Todos os comandos desse módulo devem ser precedidos da instrução CALL do BASIC. Todos os argumentos desse módulo serão tratados como qualquer expressão do BASIC, podendo ter qualquer um de seus formatos. Note que nenhum comando deste módulo será afetado pela disposição do comando PAGE.

Os novos comandos do BASIC

-EDITOR
CALL EDITOR [(<end>)]

Chama a rotina do editor de caracteres. Caso <end> seja especificado, os caracteres editados ficarão a partir do endereço <end>. Se <end> não for fornecido, será assumido o endereço C000. Ao entrar no comando você verá umatela com os seguintes elementos: o desenho dos caracteres contidos no endereço (se for a primeira vez que o comando é usado desde que o micro foi ligado, provavelmente este desenho será "lixo"); um cursor evidenciado pelo caracter que estiver inverso e a ampliação do caractere sob o qual se encontra o cursor. Nesse momento você estará no modo de seleção, e estarão disponíveis as seguintes teclas:

<TAB> -Volta ao BASIC.
<CTRL+STOP> -Volta ao BASIC, com a mensagem no HotBit 'PAREI' e 'Break' no Expert e em outros MSX.
<SHIFT+HOME>-Transfere a tabela de caracteres do micro para o endereço de edição.
Cursores -Movem o cursor sobre os caracteres.
<RETURN> -Vai para o modo de edição.

No modo de edição o cursor usado será o que está na ampliação do caractere selecionado.Estarão disponíveis as seguintes teclas:

<ESPAÇO> -Inverte o estado do ponto sob o cursor.
Cursores -Movem o cursor sobre o caractere.
<ESPAÇO+I> -Inverte o caractere em edição (deve ser o I maiúsculo, ou seja, em BASIC deve-se estar em modo "maiúsculas").
<ESPAÇO+S> -Espera uma tecla. Se for uma tecla do cursor, move o caractere em edição para a direção indicada pela tecla (note que deve ser um S maiúsculo).
<SHIFT+HOME>-Apaga o caractere em edição e faz o cursor voltar ao canto superior esquerdo, voltando-se ao modo de seleção.
<CTRL+STOP> -Volta diretamente ao BASIC.
<RETURN> -Volta ao modo de seleção.

A tabela de caracteres contém 2048 bytes, contados a partir do endereço de edição.

Exemplo de implementação de nova tabela de caracteres:

Após haver gerado uma nova tabela de caracteres (japoneses, por exemplo), o seguinte programa a coloca em "ação":

```
10 CALL EDITOR
20 SCREEN 0
30 CALL COPYRN(&HC000,(&HC000+256*8,BAS
   E(2))
40 END
```

-RENEW

```
CALL RENEW
```

Recupera o último programa contido na memória, caso este tenha se perdido devido a um reset ou um comando NEW. Caso alguma linha em BASIC tenha sido inserida após a perda o programa será destruído.

-BVERIFY

```
CALL BVERIFY
```

Compara um programa gravado em fita no formato binário com o contido na memória logo após ter sido gravado por BSAVE. Se forem iguais aparecerá uma mensagem "Ok", caso contrário será dada a mensagem de erro de verificação.

-HEADER

```
CALL HEADER
```

Mostra o cabeçalho de um programa gravado em fita.Mostrará seu nome, tipo e, se for binário, seus endereços de início, fim e execução.

-COPY

```
CALL COPYRV (<eRAMi>,<eRAMf>,<eVRAM>)
CALL COPYVR (<eVRAMi>,<eVRAMf>,<eRAM>)
```

O comando COPY copia um bloco da memória para a VRAM e vice-versa.
COPYRV copia da RAM para a VRAM.<eRAMi> e <eRAMf> delimitam o bloco fonte contido na RAM e <eVRAM> o endereço inicial do bloco destino contido na VRAM.
COPYVR copia da VRAM para a RAM. <eVRAMi> e <eVRAMf> delimitam o bloco fonte, contido na VRAM, e <eRAM> o endereço do bloco destino, contido na RAM.

Todos os argumentos devem estar entre -32768 e 65536.

Exemplo: vide exemplo do CALL EDITOR dado anteriormente.

-SETGREY

```
CALL SETGREY ( 0 / 1 )
```

Seleciona se a cópia da tela deve ser ou não impressa com escala de cinza. "1" indica impressão com escala de cinza."0" indica impressão som escala de cinza.

-DUMP

```
CALL DUMP
```

Tira uma cópia da tela (nas screens 0 a 3) na impressora. Caso a screen seja 1, 2 ou 3,os sprites também serão copiados. A cópia pode ser em modo simples ou em graduação de cinza. No modo simples um ponto só será impresso para cores com código maior ou igual a 8. Em modo de graduação de cinza será impresso um padrão para cada cor da tela.
Note que neste último comando espera-se que a impressora seja uma Epson FX80 ou compatível, não funcionando na GRAFIX MTA quando SETGREY(1) for selecionado.

-SETKEY

```
CALL SETKEY ( 0 / 1 )
```

Liga ou desliga a cópia automática, respectivamente se o argumento for 1 ou 0. A cópia automática é feita toda vez em que as interrupções estiverem ligadas e a tecla <ESC> for pressionada. Ao se entrar no EMA essa função será desligada até que se retorne ao BASIC.

Variáveis úteis do Mega Assembler

```
EC01 - End. inicial do programa fonte.
EC03 - End. final do programa fonte.t
FFF0   Ligado ou não.
```

Após um SAVE ou LOAD em binário:

```
FA30/31 - End. inicial do bloco.
FA32/33 - End. final do bloco.
FA34/35 - End. de execução.
```

Como recuperar um programa fonte perdido

Digite os comandos PAGE e DM 0.
Procure a última instrução do programa (que deve ser conhecida por você). A instrução deve ser seguida de um byte 0. Ponha o cursor sobre esse byte e anote o endereço.
Saia do comando DM e tecle o comando 'M EC03' e digite o byte menos significativo do endereço anotado. Em seguida digite o byte mais significativo do mesmo

endereço.

Como gravar um cartucho

Com o micro desligado, coloque o MA no slot de menor número do seu micro (o superior do HotBit e o esquerdo do Expert) e o cartucho a ser gravado no slot restante.
Ligue o micro. O cabeçalho inicial do MA deve aparecer no vídeo. Tecle 'PAGE 2,2', no HotBit,ou 'PAGE 3,3' no Expert. Procure o cartucho nos endereços 4000 e/ou 8000. Prepare o gravador (o disco não pode ser usado pois ele não foi inicializado ainda) e tecle 'SAVE <prog>, <endi>,<endf>', sendo <prog>, <endi> e <endf> o nome e os endereços inicial e final, respectivamente, que devem ser determinados com o estudo do cartucho. Para ler pelo BASIC, tecle 'BLOAD"CAS:", &h4000'. Um programa copiado por este procedimento não poderá ser executado, pois você apenas tem uma cópia do cartucho em RAM. Para executarmos o programa a ser copiado devemos descobrir seu endereço de execução e colocá-lo no seu endereço original de memória. Este procedimento varia de cartucho para cartucho e não é possível uma padronização.
Para maiores informação recomendamos a consulta de literatura especializada como "THE MSX RED BOOK" da Avalon Software ou "THE MSX2 TECHNICAL HANDBOOK".

Mnemônicos para Z80 - MSX

Exemplo de uso do assembler na geração de um programa:
Digite AUTO <RETURN> e na seqüência digite o seguinte programa:

```
10              ORG 0C100H
20 CHPUT:       EQU 00A2H
30              LD HL,PRINT
40 SALT:        LD A,(HL)
50              AND A
60              RET Z
70              CALL CHPUT
80              INC HL
90              JR SALT
100 PRINT:      DB 'MEGA ASSEMBLER'
110             DB 0
120             END
```

Após terminar a digitação tecle <CTRL+STOP> para sair do AUTO. Para uma compilação de verificação de erros tecle

>A <RETURN>

A compilação aparecerá na tela.
Se nenhuma mensagem de erro surgir, compile novamente, só que agora tecle:

>AOU <RETURN>

Com L C100 você poderá verificar se a compilação foi feita corretamente.
Para executar o programa acima, vá para o BASIC:

>BA <RETURN>

Tecle em seguida:

DEFUSR = &HC100 : PR = USR(0) <RETURN>

Deverá aparecer na tela a mensagem seguinte:

MEGA ASSEMBLER

Rotinas de José Alvaro Toledo Jr
Colaboração César Augusto Othero Tiossi
Correções e atualizações por A&L Soft