52 DICAS E 7 PROGRAMAS INÉDITOS

# Dicas, Macetes & Programas em Assembly

## Para MSX DISK DRIVE

TESTADOS
NOS DRIVES
MICROSOL,
SHARP E TPX

Eduardo Alberto Barbosa

CIÊNCIA MODERNA
COMPUTAÇÃO LTDA.

52 DICAS E 7 PROGRAMAS INÉDITOS

# Dicas, Macetes & Programas em Assembly

TESTADOS
NOS DRIVES
MICROSOL,
SHARP E TPX



Eduardo Alberto Barbosa

DICAS, MACETES E PROGRAMAS EM ASSEMBLY PARA MSX DISK DRIVE - EDUARDO ALBERTO BARBOSA



CAPA : Renato Martins

EDITOR : Paulo André P. Marques

FICHA CATALOGRÁFICA

| Barbosa, Eduardo Alberto |
|---|
| DICAS, MACETES E PROGRAMAS EM ASSEMBLY PARA MSX DISK DRIVE / Eduardo Alberto Barbosa. - Rio de Janeiro : CIÊNCIA MODERNA COMPUTAÇÃO Ltda, 1988 |
| 1 - Programação (Computadores/Microcomputadores) |
| 2 - Linguagem Assembly (Programa de computador) |
| CDD - 001.642 |

MARCAS REGISTRADAS :

EXPERT e MSX - São marcas registradas da GRADIENTE ELETRÔNICA S.A.

HOTBIT - É marca registrada da SHARP EQUIPAMENTOS ELETRÔNICOS S.A.

MICROSOL - É marca registrada da MICROSOL TECNOLOGIA Ltda.

TPX - É marca registrada da TROPIC INFORMÁTICA Ltda.

AGRADECIMENTOS

Agradeço aos meus pais e à minha irmã pelo incentivo, e aos amigos Paulo André P. Marques e Bento Leite pela colaboração prestada.

## PREFÁCIO

O principal objetivo deste livro é o de apresentar ao leitor, todas as potencialidades que o sistema de disco do MSX oferece. Desta forma, o livro apresenta duas partes bem definidas. A primeira que se estende do capítulo 1 ao 3, apresenta todos os recursos e características do sistema de disco do MSX, e a segunda, apresenta as listagens e análises de 5 programas que englobam grande parte dos recursos oferecidos pelo sistema de disco.

Se você nunca leu nada a respeito de sistemas de disco, poderá encontrar algumas dificuldades nos primeiros capítulos deste livro. Mas seja persistente, tente analisar as listagens dos programas no final do livro, que têm como objetivo ajudar a esclarecer quaisquer dúvidas. Se mesmo assim as dúvidas persistirem, escreva para mim que terei o maior prazer em ajudá-lo. O meu endereço para correspondência é :

CAIXA POSTAL : 37669
CEP : 22642 - RIO DE JANEIRO - RJ

SUMÁRIO

# 1. CARACTERÍSTICAS DO MSX-DOS

Os fabricantes de periféricos para o MSX no Brasil, optaram pela utilização de controladores de discos flexíveis de 5 1/4 polegadas. Deste modo todos os conhecimentos contidos neste livro, deverão ser aplicados a este tipo de unidade controladora.

No Brasil temos basicamente dois tipos de drives, com capacidades de armazenamento distintas. O drive de face simples possui a capacidade de armazenar aproximadamente 180.000 caracteres de informação, já o de face dupla tem essa capacidade praticamente dobrada para 362.000 caracteres. Vejamos então, como se processa o mecanismo de gravação e leitura dos dados num disquete.

Quando o operador (usuário) ordena a gravação de um determinado arquivo, entra em operação o sistema operacional (DOS), que por sua vez interpreta a ordem. Se não houver nenhum erro de sintaxe, o sistema operacional irá ativar a rotina conveniente da BIOS de disco, que por sua vez, transformará o comando dado pelo operador numa série de comandos em linguagem de máquina, que ao serem interpretados pelo controlador, permitirão a gravação do arquivo.

Para ler ou gravar os dados presentes na memória do micro computador, o MSX-DOS reserva uma área da memória RAM para guardar os dados a serem gravados ou lidos, e uma outra área para guardar as informações pertinentes ao arquivo com que estamos lidando. A primeira área recebe o nome de DTA ou DMA (disk transfer address ou disk memory address), e a segunda o nome de FCB (file control block).

A área compreendida pelo DTA, é preparada pelo sistema operacional ou pelo usuário, para receber os dados enviados ou retirados do disquete. Já a área do FCB, é ocupada pelas informações pertinentes a um arquivo,como o seu nome e extensão (COM,BAS,BIN,etc.), a

hora e data de atualização e o seu comprimento total. As informações contidas no DTA são arquivadas nos setores públicos do disquete, já as informações contidas no FCB são armazenadas nos setores privados. Vejamos então, o que vêem a ser os setores públicos e os setores privados.

Os setores privados são aqueles que contêm todas as informações necessárias ao perfeito funcionamento do sistema, tais como: rotina de partida (BOOT), tabela de alocação dos setores públicos, e o diretório. Como se pode observar, estas informações são importantes demais para que o sistema operacional permita ao usuário a sua alteração. Os setores públicos são todos aqueles que armazenam somente os caracteres que compõem os diversos arquivos.

No drive de face simples os setores privados ocupam os 9 setores da trilha zero, conforme se pode observar no quadro abaixo:

Setor 0 : Rotina de partida
Setor 1 : FAT
Setor 2 : FAT
Setor 3 : FAT
Setor 4 : FAT
Setor 5 : Diretório
Setor 6 : Diretório
Setor 7 : Diretório
Setor 8 : Diretório

Como se pode observar a tabela de alocação de setores públicos e o diretório ocupam quatro setores.

No drive de face dupla, cada trilha está dividida em dois lados (0 e 1). Neste caso, os setores privados ocupam todo o lado 0 da trilha zero e parte do lado 1 da mesma trilha, conforme se pode observar no quadro a seguir:

LADO 0:
Setor 0 : Rotina de partida
Setor 1 : FAT
Setor 2 : FAT
Setor 3 : FAT
Setor 4 : FAT
Setor 5 : Diretório
Setor 6 : Diretório
Setor 7 : Diretório
Setor 8 : Diretório

LADO 1:
Setor 0 : Diretório
Setor 1 : Diretório
Setor 2 : Diretório

Como se pode observar, no caso do drive de face dupla, o número de setores ocupados pela tabela de alocação de setores públicos é igual ao de face simples, enquanto que o número de setores ocupados pelo diretório é praticamente o dobro. A explicação para esta aparente discrepância será vista mais tarde.

Vamos nos deter agora na forma como o sistema operacional encara os arquivos. O MSX-DOS trata os arquivos como sendo constituídos por blocos, records e bytes. Segundo esta divisão, um arquivo é constituído de blocos, os blocos são constituídos por até 127 records,e os records por bytes. Assim sendo, se tivéssemos 512 bytes num record, este ocuparia um setor do disquete por inteiro.

Ao criar um arquivo para gravação, o sistema operacional grava no diretório os dados pertinentes ao arquivo, como: nome, extensão, atributo, data, o número inicial do aglomerado de setores (Cluster em inglês), e o tamanho do arquivo. Desta forma, um arquivo gasta 32 bytes do diretório, conforme se pode observar na figura a seguir:

| BYTES | INFORMAÇÃO |
|---|---|
| 0 a 7 | Nome do arquivo |
| 8 a 10 | Extensão do nome |
| 11 | Atributo |
| 12 a 23 | Reservado |
| 24 a 25 | Data |
| 26 a 27 | Primeiro aglomerado |
| 28 a 31 | Tamanho do arquivo |

No parágrafo anterior,vimos um conceito novo: O termo aglomerado. O aglomerado, é a unidade mínima de espaço de um disquete alocado a um arquivo. No modelo de face simples um aglomerado corresponde a um setor de 512 bytes, já no drive de face dupla, um aglomerado corresponde a dois setores de 512 bytes cada. Deste modo podemos esclarecer a aparente discrepância sobre a àrea da FAT ser igual tanto nos disquetes de face simples, como nos de dupla face, pois um arquivo é alocado de acordo com o número de aglomerados que irá gastar, e não de acordo com o número de setores. Este modo de alocação, como quase tudo na vida, tem o seu defeito, pois suponhamos que desejássemos gravar um arquivo de 30 bytes num drive de face simples. A gravação deste arquivo irá custar um aglomerado, que por sua vez se compõem de um setor, logo gastaríamos 512 bytes do disquete para gravar 30 bytes. No modelo de face dupla o gasto é ainda maior, pois um aglomerado é constituído por dois setores, e assim, gastaríamos 1024 bytes para gravar 30.

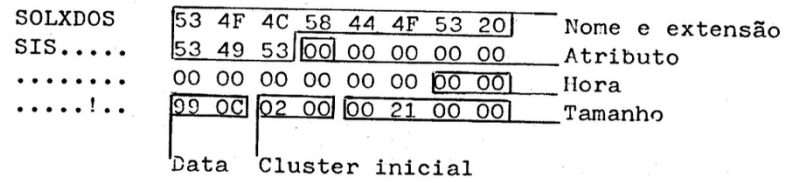
O número correspondente ao aglomerado, no diretório de um arquivo, é praticamente o início do arquivo a ser lido ou escrito no disquete. No processo de gravação de dados de um arquivo no disquete, o encadeamento dos diversos aglomerados que o formam, é feito com a ajuda da FAT, que desempenha por assim dizer, o papel de um mapa dos aglomerados que constituem

o arquivo. Para entender como este mecanismo funciona, suponha que no seu disquete existem 3 arquivos de 1024 bytes cada, chamados respectivamente de A, B e C. Se por um motivo qualquer você desejar apagar o arquivo B e no lugar dele inserir um arquivo "D" de 2048 bytes, o sistema irá substituir os 32 bytes do diretório anteriormente ocupados com as informações do arquivo "B" pelas novas informações do arquivo "D". Acontece porém, que temos dois setores livres para gravar metade das informações do arquivo "D" mas não mais que isso, uma vez que esta era a área ocupada pelo arquivo "B". Quando isto ocorre o sistema grava nessa área a primeira metade do arquivo "D", e começa a gravar a segunda metade no primeiro aglomerado disponível, após o último ocupado pelo arquivo "C". Deste modo ficamos com as duas metades do arquivo "D" separadas pelo arquivo "C", e é aí que entra a FAT, pois ela irá informar ao sistema quais os aglomerados que compõem um determinado arquivo. Como se pode observar, a FAT se comporta como um mapa dos aglomerados (setores) que compõem o arquivo. Na figura a seguir temos um quadro resumo da formatação de disquetes utilizada pelo MSX-DOS.

| | Face simples | Face dupla |
|---|---|---|
| Número de trilhas por disquete ..... | 40 | 40 |
| Número de setores por trilha ..... | 9 | 9 |
| Número de setores por disquete ..... | 360 | 720 |
| Número de setores públicos ..... | 351 | 708 |
| Número de setores privados ..... | 9 | 12 |
| Número de setores do BOOT ..... | 1 | 1 |
| Número de setores da FAT ..... | 4 | 4 |
| Número de setores do diretório ..... | 4 | 7 |
| Número de setores por aglomerado ..... | 1 | 2 |
| Número de aglomerados por disquete ..... | 351 | 354 |

## 1.2 O DIRETÓRIO

Na introdução deste livro, afirmei que o diretório ocupa de 4 a 7 setores, dependendo da formatação do disco. Mostrei também num quadro resumo, como são armazenadas as informações dos arquivos, no diretório. Vejamos agora, com mais detalhes, como realmente estão armazenadas, todas as informações necessárias de qualquer arquivo, nos setores do diretório. Como exemplo, vamos analisar o arquivo SOLX-DOS da empresa MICROSOL TECNOLOGIA. Assim sendo, examine a figura abaixo onde exibimos os 32 bytes do diretório pertencentes ao arquivo anteriormente citado.




Na esquerda temos a apresentação dos bytes, que compõem as informações do arquivo SOLX-DOS, na forma de caracteres ASCII. Na direita, temos a mesma representação em números hexadecimais. Devido à compatibilidade com a organização de disco do IBM-PC, algumas informações disponíveis não são aproveitadas no MSX, embora o sejam no IBM-PC, como o atributo e a hora da gravação do arquivo. Se você fôr bastante curioso, tente mudar o byte de atributo, para qualquer valor entre 04H e 07H, e veja se consegue listar o arquivo com o comando DIR ou FILES. Como você pode notar, as principais informações para o MSX, são na verdade a data, o cluster inicial e o tamanho do arquivo. Vamos então analisá-las detalhadamente.

DATA : A data é armazenada no diretório com 2 bytes, que no exemplo são 99H e 0CH. A data é então armazenada, por um número de 16 bits, que é decomposto da seguinte maneira:

bits F a 9 : representam o ano, com um valor que vai de 0 a 119, que deve ser somado a 1980 para dar a data exata.

bits 8 a 5 : indicam o mês (0 a 12).

bits 4 a 0 : indicam o dia (1 a 31).

O número 0C99H em binário é 0000110010011001, logo temos os seguintes valores :

ano : 0000110 = 6, logo o ano é 1986
mês : 0100 = 4, logo o mês é Abril
dia : 11001 = 25, logo o dia é 25

CLUSTER : O cluster (aglomerado) é representado por dois bytes também. No exemplo anterior, o cluster inicial, a partir do qual começou a ser gravado o arquivo, é o de número 0002H. Este valor ficará mais claro no tópico 1.3, que trata da FAT.

TAMANHO : O tamanho é representado por 4 bytes. No MSX o sistema só utiliza os 2 primeiros bytes, por uma limitação física do próprio MSX, que só pode endereçar 64 Kbytes. No exemplo anterior, o tamanho do arquivo é de 2100H = 8448 bytes.

Se por um acaso, o arquivo tivesse sido deletado, o primeiro byte não seria 53H (S), mas sim E5H, embora as demais informações (resto do nome, data, cluster inicial e tamanho) não sejam apagadas. De outra forma, se o primeiro byte fosse 00H, significaria que o diretório terminou.

## 1.3 A FAT

A FAT, como já vimos na introdução, é literalmente o mapa dos aglomerados (clusters) que constituem os arquivos. Para saber quais aglomerados compõem o arquivo, devemos obter do diretório do arquivo o cluster inicial. Após obtermos o cluster inicial, devemos inspecionar a FAT a partir da posição relativa dada pelo cluster inicial. Uma característica importante da FAT, é a de referenciar os aglomerados que constituem os arquivos, com 1 byte e meio. A lógica para esta característica é simples, pois com 1 byte só poderíamos referenciar 256 clusters, já com 2 bytes aumentaríamos este valor para 65536, que como se vê é um valor extremamente elevado, já que existirão no máximo 354 aglomerados. Diante deste impasse, os técnicos da MICROSOFT, partiram para uma solução intermediária que permitisse uma razoável economia de espaço na FAT, e ao mesmo tempo, fosse de fácil manipulação algébrica. A solução adotada foi então, a de referenciar os aglomerados com valores de 1 byte e meio, permitindo assim um número total de 4087 clusters. Assim sendo, acompanhe os exemplos a seguir, que exibem parte da FAT de um disco de face simples e de um disco de face dupla. Os dois exemplos referem-se ao diretório do arquivo SOLX-DOS, analisado no tópico anterior. Note que só são apresentados os bytes da FAT, relacinados com o arquivo SOLX-DOS.

FAT DE UM DISCO DE FACE SIMPLES (SETOR 1)

```
FC FF FF 03 40 00 05 60
00 07 80 00 09 A0 00 0B
C0 00 0D E0 00 0F 00 01
11 20 01 FF 4F .. .. ..
```

FAT DE UM DISCO DE FACE DUPLA (SETOR 1)

FD FF FF 03 40 00 05 60
00 07 80 00 09 A0 00 FF
CF .. .. .. .. .. .. ..

Analisando as duas FATs, torna-se claro que o número de bytes gasto na FAT do disco de face simples, é práticamente o dobro do número de bytes gasto na FAT do disco de face dupla. A causa desta diferença já foi vista na introdução, e se deve ao fato do aglomerado do disco de face simples ser constituído por apenas um setor, ao contrário do aglomerado do disco de dupla, que é constituído por dois setores. Vejamos agora, cada caso separadamente.

FACE SIMPLES :

Posição 0 = FFCH : Indica face simples
Posição 1 = FFFH : Indica que o próximo aglomerado pertence a um arquivo
Posição 2 = 003H : Corresponde ao setor 9
Posição 3 = 004H : Corresponde ao setor 10
Posição 4 = 005H : Corresponde ao setor 11
Posição 5 = 006H : Corresponde ao setor 12
Posição 6 = 007H : Corresponde ao setor 13
Posição 7 = 008H : Corresponde ao setor 14
Posição 8 = 009H : Corresponde ao setor 15
Posição 9 = 00AH : Corresponde ao setor 16
Posição 10 = 00BH : Corresponde ao setor 17
Posição 11 = 00CH : Corresponde ao setor 18
Posição 12 = 00DH : Corresponde ao setor 19
Posição 13 = 00EH : Corresponde ao setor 20
Posição 14 = 00FH : Corresponde ao setor 21
Posição 15 = 010H : Corresponde ao setor 22
Posição 16 = 011H : Corresponde ao setor 23

Posição 17 = 012H : Corresponde ao setor 24
Posição 18 = FFFH : Fim de arquivo. Corresponde ao setor 25

Observe atentamente no quadro da FAT do disco de face simples, como foram obtidos os valores de cada aglomerado.

Você deve estar estranhando o fato da posição 2, cujo aglomerado é 003H, estar se referindo ao setor 9. A mágica é simples, basta somar ao valor do aglomerado o número 6. Note que no disco analisado, os aglomerados pertencentes ao arquivo SOLX-DOS, são sequenciais, mas nada impede que tivéssemos na posição 2 o aglomerado 003H, e na posição 3 o aglomerado 00AH, isto aconteceria se você deletasse um arquivo pequeno, e no seu lugar gravasse um arquivo maior.

Note também, que no diretório do arquivo SOLX-DOS, o cluster inicial é o da posição 2, que como nós vimos, corresponde ao primeiro setor (9) do arquivo.

FACE DUPLA :

Posição 0 = FFDH : Indica disco de face dupla
Posição 1 = FFFH : Indica que o proximo aglomerado pertence a um arquivo
Posição 2 = 003H : Corresponde aos setores 12 e 13
Posição 3 = 004H : Corresponde aos setores 14 e 15
Posição 4 = 005H : Corresponde aos setores 16 e 17
Posição 5 = 006H : Corresponde aos setores 18 e 19
Posição 6 = 007H : Corresponde aos setores 20 e 21
Posição 7 = 008H : Corresponde aos setores 22 e 23
Posição 8 = 009H : Corresponde aos setores 24 e 25
Posição 9 = 00AH : Corresponde aos setores 26 e 27
Posição 10 = FFFH : Indica fim de arquivo. Corresponde aos setores 28 e 29

As conclusões do disco de face simples, se aplicam quase que totalmente ao disco de face dupla. O

único senão, está no fato de que a posição 2, cujo aglomerado é 003H, se refere aos setores 12 e 13. A mágica agora, é um pouco diferente, pois como cada aglomerado é formado por dois setores, devemos multiplicar o valor do aglomerado por 2 e somar 6, para obter o primeiro setor do aglomerado. Note também, que o número total de setores que constituem o arquivo SOLX-DOS, é práticamente o mesmo, pois no disco de face simples temos 17 setores, e no de face dupla 18.

Se por um acaso, o arquivo SOLX-DOS tivesse sido deletado, todos os aglomerados apresentariam o valor 00H, indicando que o espaço outrora ocupado está livre para a gravação de novos dados.

## 2. DICAS PARA ACESSAR AS ROTINAS DO BDOS

Neste capítulo iremos apresentar as diversas rotinas que compõem o BDOS (sistema operacional básico de disco) e maneira pela qual devemos acessá-las. No ambiente MSX-DOS devemos acessar as rotinas do BDOS, colocando no registro C o número da função desejada e nos demais registros as informações necessárias executando após, uma chamada para a subrotina do BDOS através da instrução CALL 0005H.

No BASIC DE DISCO a situação não é muito diferente, a única diferença está no endereço da subrotina do BDOS, que agora se localiza no endereço F37DH, e não mais no endereço 0005H.

Passemos então à apresentação das características das diversas funções que compõem o BDOS.

.00H - REINÍCIO DO SISTEMA : Esta função limpa todos os buffers e retorna para o MSX-DOS. No BASIC DE DISCO simplesmente resseta o sistema.

.01H - LEITURA DO TECLADO : Esta função lê uma tecla, imprime o caracter correspondente na tela e retorna o seu valor no registro A. A execução normal do programa é interrompida até que seja pressionada uma tecla.

.02H - IMPRESSÃO NA TELA : Esta função imprime na tela o caracter cujo código se encontra no registro E. O terminal utilizado pelo MSX é o VT52. As sequências da tecla ESC (escape) são :

| | |
|---|---|
| j | Limpa a tela |
| E | Limpa a tela |
| K | Limpa até o fim da linha |
| J | Limpa até o fim da tela |
| l | Limpa toda a linha |

| | |
|---|---|
| L | Insere uma linha |
| M | Deleta uma linha |
| Y | Posiciona o cursor |
| A | Cursor para cima |
| B | Cursor para baixo |
| C | Cursor para a direita |
| D | Cursor para a esquerda |
| H | Coloca o cursor na posição 0,0 |
| x4 | Cursor cheio (normal) |
| y4 | Cursor sublinhado |
| x5 | Cursor apagado |
| y5 | Cursor ativado |

.03H - ENTRADA AUXILIAR : Esta função espera pela chegada de um caracter pela porta serial. Este caracter é então colocado no registro A. Durante a execução desta subrotina, a subrotina para checagem do teclado é ativada para detectar um possível CONTROL-S, que suspenderá a leitura da porta serial.

.04H - SAÍDA AUXILIAR : Esta função envia para a porta serial o caracter cujo código se encontra no registro E. Da mesma forma que na função anterior realiza-se uma checagem do teclado à procura de um CONTROL-S.

.05H - SAÍDA PARA A IMPRESSORA : O caracter cujo código está contido no registo E, é enviado para a impressora. Uma checagem do teclado é feita para detectar uma possível ordem de interrupção.

.06H - I/O DIRETO DO TECLADO : No retorno desta função se o registro E tiver o valor FFH, então o registro A retorna com o valor do caracter, se encontrou algum, a flag zero é ressetada; de outro modo a flag zero é setada. Se no retorno desta subrotina o regitro E fôr diferente de FFH,

então o sistema assume que o registro E contêm um caracter válido que será impresso na tela. Nenhuma checagem de teclado para detectar os comandos CONTROL-S e CONTROL-C é feita.

.07H - LEITURA DIRETA DO TECLADO : Esta função espera pelo caracter a ser teclado, e retorna o seu código no registro A. Nenhuma checagem de status do teclado é feita.

.08H - LEITURA DE TECLADO SEM ECO : Esta função funciona de modo semelhante à função 01H, com a única diferença de não imprimir na tela o caracter lido.

.09H - IMPRIME UMA STRING : Quando chama-se esta função o par DE deverá apontar para o endereço de uma string, que por sua vez, deverá terminar com o caracter "$" (24H). Cada caracter será mostrado na tela da mesma forma que na função 02H.

.0AH - LEITURA DE UM BUFFER DO TECLADO : Quando chama-se esta função, o par DE deverá apontar para um endereço de memória que conterá o conjunto de caracteres lidos. o primeiro byte deste endereço de memória deverá conter o número de caracteres que irão compôr o buffer, devendo por este motivo, ser diferente de zero. Os caracteres lidos serão colocados a partir do terceiro byte do buffer. A leitura continua até que um Carriage Return (ENTER) seja pressionado. Se o número de caracteres teclados ultrapassar o máximo permitido, um beep irá soar até que um Carriage Return seja teclado. Todos os caracteres extras serão ignorados. O segundo byte do buffer irá conter o número de caracteres recebidos. Os caracteres de edição são os seguintes :

| | |
|---|---|
| ▶ | Copia um caracter |
| DEL | Pula um caracter |
| SEL | Copia até a primeira ocorrência do próximo caracter a ser teclado |
| CLS | Pula até a primeira ocorrência do próximo caracter a ser teclado |
| ▼ | Copia até o fim da linha |
| ▲,ESC | Apaga toda a linha |
| HOME | Edita a linha |
| ◀,BS | Apaga o caracter anterior |
| INS | Controla o modo de inserção |

.OBH - CHECAGEM DO STATUS DO TECLADO : Se algum caracter foi lido do teclado, então o registro A retorna com o valor FFH, caso contrário retorna com o valor OOH. Se o caracter lido for CONTROL-S, CONTROL-C ou CONTROL-P, a decisão apropriada será tomada.

.OCH - NÚMERO DA VERSÃO : Esta função simplesmente retorna a versão do sistema. No caso do MSX-DOS o par HL retorna com o valor 22H, indicando uma compatibilidade com o sistema CP/M-80 versão 2.2.

.ODH - RESET DO DISCO : Esta função limpa todos os arquivos, portanto tome o cuidado de fechar todos os arquivos antes de chamar esta função.

.OEH - SELECIONA O DISCO : Esta função seleciona o disco de acordo com o valor do registro E (O=A, 1=B, etc.).

.OFH - ABRE UM ARQUIVO : Esta função é muito usada para verificar se um determinado arquivo existe ou não e também para preparar um arquivo para leitura. Na chamada desta função o par DE deve apontar para o FCB. No retorno, se o registro A apresen-

tar o valor FFH, o arquivo não está presente no diretório, caso contrário, o registro A retorna com o valor OOH. O par DE deverá apontar para o FCB de um arquivo não aberto.

.10H - FECHA UM ARQUIVO : Esta função deve ser executada após o término da utilização de qualquer arquivo, para atualizar as informações do diretório pertinentes ao arquivo. na entrada o par DE deverá apontar para um arquivo aberto.

.11H - PROCURA O PRIMEIRO ARQUIVO : Quando chama-se esta função o par DE deve apontar para o FCB de um arquivo a ser pesquisado no diretório do disco. O nome do arquivo no FCB poderá conter o caracter "?". No retorno desta função o registro A conterá o valor FFH se o arquivo não foi encontrado, e o valor OOH em caso contrário. Se o arquivo fôr encontrado, o seu FCB será colocado a partir da posição 33 do DTA. Note que no CP/M padrão o FCB do arquivo encontrado é colocado na posição 128 do DTA. O par DE deverá apontar para o FCB de um arquivo não aberto.

.12H - PROCURA O PRÓXIMO ARQUIVO : Esta função deve ser ativada após a execução da instrução anterior, para procurar o próximo arquivo que também preencha os requisitos do FCB, uma vez que poderemos usar o caracter "?". Tanto as entradas como as saídas são iguais às da função anterior.

.13H - DELETA ARQUIVO : Esta função apaga o arquivo cujo nome se encontra no FCB. Quando chama-se esta função o par DE deve apontar para o FCB do arquivo que se deseja apagar. No retorno o registro A conterá o valor FFH se o arquivo não foi encontrado, e o valor OOH em caso contrário. Note que podemos usar o caracter "?", e que,

neste caso, todos os arquivos que preencherem os requesitos serão apagados. O par DE deverá apontar para o FCB de um arquivo não aberto.

.14H - LEITURA SEQUENCIAL : Esta função lê um bloco de 128 bytes de um arquivo especificado no FCB. Os 128 bytes lidos são colocados a partir do DTA. Na chamada o par DE aponta para o FCB do arquivo a ser lido. No retorno o registro A contém o valor 01H se o end-of-file (fim de arquivo) foi detectado, já se operação de leitura foi feita com êxito o registro A retorna com o valor 00H. Se houve algum problema de leitura o registro A retorna o valor 02H.

.15H - ESCRITA SEQUENCIAL : Esta função grava 128 bytes contidos no DTA, no arquivo cujo nome se encontra no FCB. Na chamada o par DE aponta para o FCB do arquivo. No retorno da função, se o registro A apresentar o valor 01H há falta de espaço no disco, já se apresentar o valor 00H, a escrita foi bem sucedida, e se apresentar outro valor, houve algum problema de escrita.

.16H - CRIA UM ARQUIVO : Esta função cria um arquivo no disco. A primeira atitude é verificar se há espaço suficiente no diretório para a inclusão de um novo arquivo. Se não houver espaço suficiente (diretório cheio) o registro A retorna com o valor FFH, em caso contrário retorna com o valor 00H. Na chamada o par DE deve apontar para o FCB de um arquivo não aberto.

.17H - RENOMEIA ARQUIVOS : Esta função é usada para renomear um ou mais arquivos. Na entrada o par DE aponta para o FCB do arquivo a ser renomeado. No FCB o primeiro byte conterá o número do drive (1=A, 2=B, etc.), seguido do nome do arquivo. A

partir da décima segunda posição do FCB (DE+12) devemos colocar o novo nome do arquivo. Se o arquivo original não fôr encontrado o registro A retorna com o valor FFH. Note que podemos usar o caracter "?" para renomear vários arquivos de uma só vez.

.18H - VETOR DE CONEXÃO : Esta função retorna com 1 bit setado no par HL, para todos os drives disponíveis no sitema. Assim se você só tiver um drive no sitema, o valor que retorna é o 1, já se tiver 2 drives o valor que retorna é o 3, e assim por diante até um máximo de 16 drives.

.19H - DRIVE ATUAL : No retorno desta função o registro A indica o drive atual (0=A, 1=B, etc.).

.1AH - DEFINE O ENDEREÇO DO DTA: Na chamada desta função o par DE deverá conter o endereço de memória a ser ocupado pelo DTA. Ao usarmos a função 0DH, o DTA é automaticamente deslocado para o endereço 0080H.

.1BH - ENDEREÇO DA FAT : Quando chama-se esta função o registro E deverá conter o número do drive (1=A, 2=B, etc.). No retorno o registro A contém o número de setores por aglomerado, ou o valor FFH se o drive especificado fôr inválido. O registro BC contém o tamanho do setor (512 bytes), o par DE contém o número de aglomerados do disco, o par HL contém o número de aglomerados disponíveis no disco, o registro IY aponta para o endereço do FAT e o registro IX aponta para o BPB.

.1CH - PROTEÇÃO CONTRA ESCRITA : Esta função não está implementada no MSX-DOS. Simplesmente retorna sem processamento algum.

.1DH - VETOR R/O : Esta função não está implementada no MSX-DOS. Simplesmente retorna com zeros indicando que o disco não está protegido.

.1EH - DETERMINA ATRIBUTOS DO ARQUIVO : Não está implementada.

.1FH - PEGA PARÂMETROS DO DISCO : Não está implementada.

.20H - PEGA CÓDIGO DO USUÁRIO : Esta função não está implementada.

.21H - LEITURA ALEATÓRIA : Na entrada o par DE aponta para o FCB. O record é lido e transferido para o DTA. Se o fim de arquivo (EOF) foi detectado, o registro A retorna com o valor 01H, já se a leitura foi bem sucedida retorna com o valor 00H. Se houve algum problema no carregamento do record, o registro A retorna com o valor 02H.

.22H - ESCRITA ALEATÓRIA : Na entrada o par DE aponta para o FCB do arquivo. Se não há espaço para escrita o registro A retorna com o valor 01H, indicando disco cheio, já se a operação foi completada com êxito o registro A retorna com o valor 00H.

.23H - TAMANHO DO ARQUIVO : Na entrada o par DE aponta para o FCB de um arquivo ainda não aberto. No retorno o registro A assume o valor FFH se não encontrou o arquivo. De outra forma o campo de arquivo randômico do FCB passa a contêr o número de records que compõem o arquivo, e o registro A assume o valor 00H.

.24H - DETERMINA REGISTRO ALEATÓRIO : Na entrada o par DE aponta para o FCB de um arquivo. Esta função seta o campo de arquivo randômico do FCB.

.25H - REINÍCIO DO DRIVE : Esta função não está implementada no MSX-DOS.

.26H - ESCRITA RANDÔMICA DE UM BLOCO : Esta função é essencialmente a mesma que a função 27H descrita abaixo, exceto pela operação de escrita e a consequente indicação de disco protegido para a escrita. Se não existir espaço suficiente no disco o registro A retorna com o valor 01H, e nenhum record será escrito, embora o arquivo seja setado com o tamanho especificado pelo campo arquivo randômico.

.27H - LEITURA RANDÔMICA DE UM BLOCO : Na entrada desta função o par DE aponta para o FCB de um arquivo, e o par HL deverá contêr um contador de records. O número específico de records (em termos do tamanho do campo de records) são lidos do endereço especificado pelo campo de record randômico (3 bytes se o tamanho do setor fôr menor ou igual a 40H, e 4 bytes no caso contrário) no DTA.Se o EOF (end-of-file=fim do arquivo) fôr achado o registro A retorna com o valor 01H, preenchendo com zeros os records parciais. O registro A retorna com o valor 00H numa leitura bem sucedida. Em qualquer caso o par HL retorna com o número do record lido.

.28H - ESCRITA RANDÔMICA COM ZEROS : Funciona de modo semelhante à função 22H, exceto sempre que uma escrita requesitada amplie um arquivo continuamente, o espaço intermediário é preenchido com zeros.

.29H - Esta função não está implementada.

.2AH - DATA : Esta função retorna a data. O par HL con-

tém o ano, o registro D contém o mês (1=Jan,etc.) e o registro E contém o dia. O registro A retorna o dia da semana (0=Domingo,etc.).

.2BH - MODIFICA A DATA : Na entrada o par DE e o par HL deverão conter os valores como descrito na função acima.

.2CH - HORA : Esta função retorna com a hora. O registro H contém as horas, o L os minutos, o D os segundos, e o registro E contém os centésimos de segundo.

.2DH - MODIFICA A HORA : Esta função modifica a hora de de acordo com os valores contidos nos registros como descrito na função anterior.

.2EH - VERIFICAÇÃO DE ESCRITA : Esta função difere de sistema para sistema, aconselho-o a usar a verificação através de um programa que empregue a rotina de leitura (sequencial, randômica, setorial, etc.) após a correspondente escrita. Desta forma, você estará evitando uma possível incompatibilidade.

.2FH - LEITURA DE SETORES : Esta função lê o setor especificado pelo par DE. Na entrada o registro H deverá contêr o número de setores a serem lidos e o registro L deverá contêr o valor 00H. O setor lido será colocado no DTA. Note que se o par DE tiver o valor 0005H, e o registro H o valor 02H, os setores 5 e 6 serão lidos e colocados no DTA.

.30H - ESCRITA DE SETORES : Esta função realiza a escrita de um ou mais setores, de acordo com o valor dos registros conforme visto na função anterior.

Após a apresentação de todas as rotinas disponíveis do BDOS, passaremos a descrição do formato do FCB no MSX-DOS, conforme se pode observar no quadro a seguir :

| BYTE | FUNÇÃO |
|---|---|
| 0 | Número do drive (1=A, 2=B) |
| 1-8 | Nome do arquivo |
| 9-11 | Extensão do nome do arquivo |
| 12 | Campo de extensão |
| 13 | Reservado |
| 14 | Extensão do campo de extensão |
| 15 | Contador de records |
| 16-19 | Tamanho do arquivo em bytes |
| 20-21 | Data<br>Bits F-9 (ano) = 0-119 (1980-2099)<br>Bits 8-5 (mês) = 0-12<br>Bits 4-0 (dia) = 1-31 |
| 22-23 | Hora<br>Bits F-B (hora) = 0-23<br>Bits A-5 (min.) = 0-59<br>Bits 4-0 (seg.) = 0-29<br>(2 segundos de incremento) |
| 24-31 | Reservado |
| 32 | Record relativo do bloco corrente |
| 33-36 | Número do record relativo ao início do arquivo,começando com zero. |

## 3. OUTRAS DICAS IMPORTANTES

Neste capítulo iremos apresentar algumas rotinas interessantes disponívies na interface controladora do drive, como também algumas dicas que acessam diretamente o disk drive.

### 3.1 - ROTINAS DA INTERFACE :

A) ROTINA PARA FORMATAÇÃO DE UM DISQUETE : Esta rotina encontra-se a partir do endereço 401CH da ROM, existente na controladora do disk drive. Ao chamá-la, o par HL deverá conter o valor 8088H, o par DE o valor 0051H, o par BC o valor 6015H, e o registro A o tipo de formatação com os seguintes possíveis valores :

01H - 40 trilhas face simples (180 Kb)
02H - 40 trilhas face dupla (360 Kb)
03H - 80 trilhas face simples (360 Kb)
04H - 80 trilhas face dupla (720 Kb)

Para ativarmos esta rotina, poderemos usar a rotina CALSLT (001CH),colocando no registro IX o valor 401CH e na parte alta do registro IY o número do slot, onde se encontra a interface do drive. Se você desejar tirar dúvidas sobre o procedimento acima, observe o programa de cópia de setores.

B) ROTINA PARA LEITURA DIRETA DE UM SETOR : Esta rotina está disponível na ROM da interface controladora, mas podemos acessá-la pela ROM da BIOS. O endereço na BIOS é 0144H, e os parâmetros de entrada são os seguintes :

HL - Endereço da RAM a partir do qual será colocado o setor lido
DE - Número do setor inicial a ser lido
B - Número de setores a serem lidos
C - Parâmetro de formatação. Os valores são :

F8H - 40 trilhas em face simples
F9H - 40 trilhas em face dupla

A - Número do drive (0=A)
Flag da Carry - Deverá estar ressetada.

C) ROTINA PARA ESCRITA DIRETA DE UM SETOR : Esta rotina é semelhante à anterior, exceto no que diz respeito à flag da carry, que agora deverá estar setada. O endereço de acesso e os demais parâmetros de entrada são os mesmos.

D) ROTINA PARA INICIALIZAÇÃO DO BASIC DE DISCO : Esta rotina é ativada pelo sistema operacional (MSX-DOS) quando desejamos voltar ao BASIC de DISCO. Como se encontra na ROM da interface, devemos acessá-la colocando no registro IX o valor 4022H, na parte alta do registro IY, o slot em que se encontra a interface do drive, executando então um salto (JP) para a rotina CALSLT (001CH).

E) ROTINA PARA PARAR O DRIVE APÓS O CARREGAMENTO DE UM ARQUIVO : Esta rotina pode ser ativada colocando no registro IX o valor 4029H, e na parte alta do registro IY, o slot em que se encontra a interface do drive, executando após uma chamada (CALL) para a rotina CALSLT (001CH).

## 3.2 - DICAS PARA O ACESSO DIRETO DO DISK DRIVE :

A) DETECÇÃO DA PROTEÇÃO PARA ESCRITA : Aqui surge a primeira incompatibilidade entre os modelos de interface controladora de disk drive, fabricados no Brasil. Se a sua interface foi fabricada pela MICROSOL, ou pela EXPAND, ou ainda pela TROPIC, você deverá fazer o teste lendo o conteúdo da porta DOH, já se a sua interface foi fabricada pela SHARP o processo é mais complicado, como veremos a seguir. A porta DOH é chamada de porta de comando status do disco. O bit 6 do valor lido desta porta, nos indica se o disquete está ou não protegido para escrita. No modelo da SHARP, devemos ler o conteúdo de uma posição de memória,ao invés de uma porta. A posição de memória equivalente à porta DOH, é o endereço 7FF8H,que se encontra na interface controladora da SHARP.

B) DETECÇÃO DO FURO DE ÍNDICE : Como você já deve ter notado, o disquete apresenta um pequeno furo ao lado do furo central, que é particularmente útil para determinarmos a velocidade do drive, como veremos no programa de medição de velocidade. O bit 1 da porta de comando ( entenda-se por porta de comando o endereço 7FF8H na controladora da SHARP, e a porta DOH nas demais controladoras nacionais ),ao assumir o valor 1 nos indica que o furo de índice foi detectado. Como a detecção do furo de índice é cíclica, podemos usá-la para determinar a velocidade do drive, que no nosso caso deverá ser igual a 300 rpm.

C) CONTROLE DO DRIVE : Nos drives da MICROSOL, EXPAND e TPX, para selecionarmos o drive A,devemos setar o bit zero do dado a ser enviado pela porta D4H. Este procedimento irá acender a luz do drive A. Se quisermos agora ligar o motor do referido drive,deveremos setar o bit 5 do dado a ser enviado pela porta D4H. O qua-

dro abaixo resume os dados a serem enviados pela porta D4H, para controlarmos o drive A :

21H - Seleciona o drive A e liga o motor
01H - Seleciona o drive A e desliga o motor
00H - Desliga todos os drives

Examinando o quadro anterior, fica claro que se quisermos parar o drive de qualquer um dos três fabricantes antes mencionados, deveremos enviar pela porta D4H, o valor 00H.

No caso do drive da SHARP,usaremos um endereço de memória e não uma porta, para controlar o drive. Desta forma, devemos colocar no endereço 7FFDH quaisquer um dos valores do quadro a seguir:

82H - Seleciona o drive A e liga o motor
01H - Desliga todos os drives

Um detalhe que não podemos deixar de mencionar, é o dos endereços de memória que controlam o drive da SHARP, se encontrarem na interface controladora. Assim sendo, para acessá-los, deveremos antes selecionar a página 1 do slot onde se encontra a interface. Para esclarecer qualquer dúvida observe o exemplo de medição da velocidade do drive, no capítulo 4. Note que é muito mais simples usar a rotina (4029H) da ROM da interface, para parar o drive.

## 3.3 - ENDEREÇOS ÚTEIS :

A) NÚMERO DE DRIVES DO SISTEMA : Em todos os sistemas comercializados no Brasil, o endereço F347H indica o número de drives disponíveis. Assim sendo, se você pressionou a tecla CONTROL ao ligar o seu micro, o valor contido neste endereço é 01H, indicando que o sistema só possui um drive. Se você possui um ou dois drives e não pressionou a tecla CONTROL, o valor do

endereço F347H passa a ser 02H. Este endereço pode ser usado para detectar se a tecla CONTROL foi,ou não pressionada.

B) SISTEMA : O endereço F346H indica se o sistema estava ou não presente no disquete, que se encontrava no drive quando ligamos o micro. Se este endereço contiver o valor 00H, o sistema não estava presente, de outra forma, o valor contido no referido endereço seria FFH.

C) SLOT DA INTERFACE : O endereço F348H indica o slot onde se encontra a interface controladora do drive. Podemos utilizá-lo para acessar as rotinas da BIOS da interface, como no exemplo abaixo :

ROTINA PARA PARAR O DRIVE

```
LD    IY,(F348H-01H)   ;IY=slot da interface
LD    IX,4029H         ;IX=endereço da subrotina
CALL  001CH            ;Pára o drive
RET                    ;Retorna
```

## 4. PROGRAMAS

Neste capítulo iremos apresentar alguns programas, para fixar os conceitos vistos anteriormente. A apresentação dos programas, segue sempre o mesmo padrão, ou seja, primeiramente é apresentada a listagem, e logo após, uma explicação das principais rotinas que compõem o programa.

Os programas foram desenvolvidos, utilizando o DEVPAC 80 da empresa inglêsa HI-SOFT. As principais diferenças entre este pacote, e o SIMPLE ASM da CORAL, que é o mais utilizado no BRASIL, são as seguintes :

| | DEVPAC 80 | SIMPLE ASM |
|---|---|---|
| Números Hexadecimais | ǂF37F | 0F37FH |
| Números Binários | %11110000 | 11110000B |
| Números Decimais | 65000 | 65000 |
| Labels | LOOP | LOOP: |
| Comprimento dos labels | até 50 car. | 6 caracteres |
| Listagem fonte | até 43 Kbytes | até 16 Kbytes |

No quadro anterior, as quantidades numéricas são apenas exemplos, para que você possa adaptar as listagens deste livro, ao seu montador assembler. Uma outra diferença importante, reside no fato, de que no DEVPAC 80, não é necessário numerar as linhas, ao contrário do SIMPLE ASM da CORAL.

Alguns programas foram escritos, para rodarem no ambiente MSX-DOS. Deste modo, se você fôr utilizar o SIMPLE ASM, ou qualquer outro equivalente (ASMCOCAR, MEGA ASSEMBLER, HOTASM, etc.), deverá definir um ORG 0100H no início do programa fonte, e depois compilá-lo com um offset, com o seguinte comando :

```
AON/8F00
```

O comando anterior deslocou o programa objeto (compilado), para o endereço 9000H. Você deverá anotar então, o endereço final do programa objeto, que será certamente maior que 9000H, e utilizar o programa BASIC abaixo :

```
10 REM INI=&H9000 E FIM=VALOR FINAL
20 OPEN "A:NOME.COM" FOR OUTPUT AS #1
30 FOR A = INI TO FIM
40 PRINT #1,CHR$(PEEK(A));
50 NEXT A
60 CLOSE #1
70 END
```

Na listagem acima você deverá substituir a linha 10 por outra do tipo :

```
10 LET INI=&H9000:LET FIM=&HA000
```

Onde se supõe que o programa compilado tenha terminado no endereço A000H.

Segue agora, a apresentação dos programas fonte, seguida dos respectivos comentários.

O comando anterior deslocou o programa objeto (compilado), para o endereço 9000H. Você deverá anotar então, o endereço final do programa objeto, que será certamente maior que 9000H, e utilizar o programa BASIC abaixo :

```
10 REM INI=&H9000 E FIM=VALOR FINAL
20 OPEN "A:NOME.COM" FOR OUTPUT AS #1
30 FOR A = INI TO FIM
40 PRINT #1,CHR$(PEEK(A));
50 NEXT A
60 CLOSE #1
70 END
```

Na listagem acima você deverá substituir a linha 10 por outra do tipo :

```
10 LET INI=&H9000:LET FIM=&HA000
```

Onde se supõe que o programa compilado tenha terminado no endereço A000H.

Segue agora, a apresentação dos programas fonte, seguida dos respectivos comentários.

```
;====================================
;!        COPYSET VERSAO 1.0         !
;! PROIBIDA A REPRODUCAO PARA FINS   !
;!           COMERCIAIS              !
;!  AUTOR: EDUARDO BARBOSA - 1988    !
;====================================

            DEFB #FE
            DEFW START
            DEFW END
            DEFW START
PRISET      EQU  #F975
ULTSET      EQU  #F977
AUXSET      EQU  #F979
NUMSET      EQU  #F97B
DENSA       EQU  #F980
DENSB       EQU  #F981
HFORM       EQU  #FFAC
CALSLT      EQU  #001C
BUFFER      EQU  #F600
CSRY        EQU  #F3DC
BDOS        EQU  #F37D
PHYDIO      EQU  #0144
            ORG  #DA00
START       DI
            IN   A,(#A8)
            LD   E,A
            RRA
            RRA
            RRA
            RRA
            AND  #0F
            OR   E
            OUT  (#A8),A
INICIO      EI
            CALL CLS
            CALL QUADRO
```

```
            CALL VERFORM
            LD   (DENSA),A
            LD   A,E
            LD   (DENSB),A
            LD   A,(DENSA)
            CP   #01
            JR   NZ,DOBSET
            LD   HL,#0167
            LD   (NUMSET),HL
            JR   TESTFORM
DOBSET      LD   HL,#02CF
            LD   (NUMSET),HL
TESTFORM    LD   A,(DENSA)
            PUSH AF
            LD   A,(DENSB)
            POP  DE
            LD   E,D
            CP   E
            CALL NZ,FORMAT
            CALL COPYSET
            CALL NOVCOP
            JR   Z,INICIO
            LD   IX,#4022
            LD   HL,(HFORM)
            LD   L,#00
            PUSH HL
            POP  IY
FIMPRG      IN   A,(#A8)
            DI
            AND  #F0
            OUT  (#A8),A
            EI
            CALL CALSLT
            RET
CLS         PUSH AF
            PUSH BC
            DI
```

```
            XOR   A
            OUT   (#99),A
            OR    #40
            OUT   (#99),A
            LD    BC,#03C0
CLS1        LD    A,#20
            OUT   (#98),A
            DEC   BC
            LD    A,B
            OR    C
            JR    NZ,CLS1
            EI
            POP   BC
            POP   AF
            RET
QUADRO      LD    HL,#0005
            LD    A,#C0
            LD    B,#28
            CALL  POSIT
            CALL  IMPLCAR
            LD    HL,#0010
            LD    A,#C3
            LD    B,#28
            CALL  POSIT
            CALL  IMPLCAR
            LD    B,#0A
            LD    HL,#0006
            LD    A,#C6
COLUNA1     PUSH  AF
            PUSH  HL
            CALL  POSIT
            POP   HL
            POP   AF
            OUT   (#98),A
            INC   L
            DJNZ  COLUNA1
            LD    B,#0A
```

```
            LD    HL,#2706
            LD    A,#DE
COLUNA2     PUSH  AF
            PUSH  HL
            CALL  POSIT
            POP   HL
            POP   AF
            OUT   (#98),A
            INC   L
            DJNZ  COLUNA2
            RET
VERFORM     LD    HL,#0707
            LD    DE,MENSG4
            CALL  STRING
            LD    HL,#0A0A
            LD    DE,MENSG5
            CALL  STRING
            LD    HL,#130A
            LD    DE,MENSG2
            CALL  STRING
            CALL  DISCA
            CALL  LEFORM
            PUSH  AF
            CALL  ESPERA
            LD    HL,#010A
            LD    DE,MENSGC
            CALL  STRING
            LD    HL,#0A0A
            LD    DE,MENSG5
            CALL  STRING
            LD    HL,#130A
            LD    DE,MENSG3
            CALL  STRING
            CALL  DISCB
            CALL  LEFORM
            LD    E,A
            PUSH  DE
```

```
            CALL ESPERA
            CALL CLS
            CALL QUADRO
            POP  DE
            POP  AF
            RET
LEFORM      LD   E,#01
            LD   C,#1B
            CALL BDOS
            PUSH AF
            CP   02
SIMPLES1    JR   NC,DUPLA1
            LD   HL,#170A
            LD   DE,MENSG6
            CALL STRING
            JR   CONTVER
DUPLA1      LD   HL,#170A
            LD   DE,MENSG7
            CALL STRING
CONTVER     POP  AF
            RET
FORMAT      CALL CLS
            CALL QUADRO
            LD   HL,#080A
            LD   DE,MENSGF
            CALL STRING
            CALL DISCB
            LD   HL,(HFORM)
            LD   L,#00
            PUSH HL
            POP  IY
            LD   IX,#401C
            LD   A,(DENSA)
            LD   HL,#8088
            LD   DE,#0051
            LD   BC,#6015
            CALL CALSLT
```

```
            RET
COPYSET     CALL CLS
            CALL QUADRO
            LD   HL,#0C07
            LD   DE,MENSG8
            CALL STRING
            CALL DISCA
            LD   E,#01
            LD   C,#1B
            CALL BDOS
            LD   HL,#0000
            LD   (PRISET),HL
            LD   (ULTSET),HL
            LD   (AUXSET),HL
            CALL MENLETI
            LD   DE,#0000
LOOPLET     PUSH DE
            PUSH HL
            LD   HL,(#F351)
            LD   A,(DENSA)
            ADD  A,#F7
            LD   C,A
            XOR  A
            LD   B,#01
            LD   IX,PHYDIO
            LD   IY,#0000
            CALL CALSLT
            POP  HL
            PUSH HL
            LD   DE,(#F351)
            LD   BC,#0200
            EX   DE,HL
            LDIR
            POP  HL
            POP  DE
            PUSH DE
            PUSH HL
```

```
        EX    DE,HL
        CALL  CONVCEN
        LD    HL,#190A
        LD    DE,BUFFER
        CALL  STRING
        POP   HL
        POP   DE
        EX    DE,HL
        LD    BC,(NUMSET)
        XOR   A
        PUSH  HL
        SBC   HL,BC
        POP   HL
        EX    DE,HL
        JR    NC,GRVSET
        LD    BC,#0200
        ADD   HL,BC
        LD    A,H
        CP    #D9
        JR    Z,GRVSET
        INC   DE
        JR    LOOPLET
GRVSET  LD    (ULTSET),DE
        PUSH HL
        LD    HL,(PRISET)
        LD    (AUXSET),HL
        POP   HL
        PUSH  DE
        PUSH  HL
        LD    HL,#0A0D
        LD    DE,MENSG1
        CALL  STRING
        LD    HL,#1D0D
        LD    DE,MENSG3
        CALL  STRING
        LD    HL,#080A
        LD    DE,MENSGD
```

```
CALL STRING
LD   HL,(PRISET)
CALL CONVCEN
LD   HL,#180A
LD   DE,BUFFER
CALL STRING
LD   HL,(ULTSET)
CALL CONVCEN
LD   HL,#1E0A
LD   DE,BUFFER
CALL STRING
LD   HL,#070E
LD   DE,MENSGB
CALL STRING
LD   HL,#220F
LD   (CSRY),HL
LD   C,#08
CALL BDOS
LD   HL,#010A
LD   DE,MENSGC
CALL STRING
LD   HL,#0B0A
LD   DE,MENSGA
CALL STRING
LD   HL,(PRISET)
CALL CONVCEN
LD   HL,#1A0A
LD   DE,BUFFER
CALL STRING
LD   HL,#010D
LD   DE,MENSGC
CALL STRING
LD   HL,#010E
LD   DE,MENSGC
CALL STRING
POP  HL
POP  DE
```

```
        LD    HL,#0100
        LD    (ULTSET),DE
        LD    DE,(PRISET)
LOOPESC PUSH  DE
        PUSH  HL
        PUSH  DE
        LD    DE,(#F351)
        LD    BC,#0200
        LDIR
        POP   DE
        LD    HL,(#F351)
        LD    A,(DENSA)
        ADD   A,#F7
        LD    C,A
        LD    B,#01
        XOR   A
        SCF
        LD    IY,#0000
        LD    IX,PHYDIO
        CALL  CALSLT
        POP   HL
        POP   DE
        PUSH  DE
        PUSH  HL
        EX    DE,HL
        CALL  CONVCEN
        LD    HL,#1A0A
        LD    DE,BUFFER
        CALL  STRING
        POP   HL
        LD    BC,#0200
        ADD   HL,BC
        POP   DE
        PUSH  DE
        PUSH  HL
        LD    HL,(ULTSET)
        XOR   A
```

```
            SBC   HL,DE
            JR    Z,FIMGRV
            POP   HL
            POP   DE
            INC   DE
            JR    LOOPESC
FIMGRV      POP   HL
            POP   DE
            INC   DE
            LD    (PRISET),DE
            PUSH  DE
            PUSH  HL
            EX    DE,HL
            DEC   HL
            LD    BC,(NUMSET)
            XOR   A
            SBC   HL,BC
            POP   HL
            POP   DE
            RET   NC
            PUSH  DE
            CALL  MENLET
            POP   DE
            JP    LOOPLET
MENLET      LD    HL,#010A
            LD    DE,MENSGC
            CALL  STRING
            LD    HL,#0C07
            LD    DE,MENSG8
            CALL  STRING
            LD    HL,#060A
            LD    DE,MENSGE
            CALL  STRING
            LD    HL,(AUXSET)
            CALL  CONVCEN
            LD    HL,#190A
            LD    DE,BUFFER
```

```
            CALL STRING
            LD   HL,(ULTSET)
            CALL CONVCEN
            LD   HL,#1F0A
            LD   DE,BUFFER
            CALL STRING
            LD   HL,#0A0D
            LD   DE,MENSG1
            CALL STRING
            LD   HL,#1D0D
            LD   DE,MENSG2
            CALL STRING
            LD   HL,#070E
            LD   DE,MENSGB
            CALL STRING
            LD   HL,#220F
            LD   (CSRY),HL
            LD   C,#08
            CALL BDOS
            LD   HL,#010A
            LD   DE,MENSGC
            CALL STRING
MENLETI     LD   HL,#0C0A
            LD   DE,MENSG9
            CALL STRING
            LD   HL,(PRISET)
            CALL CONVCEN
            LD   HL,#190A
            LD   DE,BUFFER
            CALL STRING
            LD   HL,#010D
            LD   DE,MENSGC
            CALL STRING
            LD   HL,#010E
            LD   DE,MENSGC
            CALL STRING
            LD   HL,#0100
```

```
        RET
DISCA   LD    HL,#0A0D
        LD    DE,MENSG1
        CALL  STRING
        LD    HL,#1D0D
        LD    DE,MENSG2
        CALL  STRING
        LD    HL,#070E
        LD    DE,MENSGB
        CALL  STRING
        LD    HL,#220F
        LD    (CSRY),HL
        LD    C,#08
        CALL  BDOS
        RET
DISCB   LD    HL,#0A0D
        LD    DE,MENSG1
        CALL  STRING
        LD    HL,#1D0D
        LD    DE,MENSG3
        CALL  STRING
        LD    HL,#070E
        LD    DE,MENSGB
        CALL  STRING
        LD    HL,#220F
        LD    (CSRY),HL
        LD    C,#08
        CALL  BDOS
        RET
ESPERA  LD    HL,#010D
        LD    DE,MENSGC
        CALL  STRING
        LD    HL,#010E
        LD    DE,MENSGC
        CALL  STRING
        LD    HL,#060E
        LD    DE,MENSGB
```

```
            CALL STRING
            LD   HL,#060E
            CALL POSIT
            LD   A,#20
            OUT  (#98),A
            LD   HL,#210F
            LD   (CSRY),HL
            LD   C,#08
            CALL BDOS
            LD   HL,#010E
            LD   DE,MENSGC
            CALL STRING
            RET
NOVCOP      CALL CLS
            CALL QUADRO
            LD   HL,#080B
            LD   DE,MENSGG
            CALL STRING
            LD   HL,#200C
            LD   (CSRY),HL
            LD   C,#08
            CALL BDOS
            CP   #53
            RET  Z
            CP   #73
            RET  Z
            RET
STRING      CALL POSIT
            PUSH AF
            PUSH BC
            PUSH DE
IMPSTR      LD   A,(DE)
            CP   #24
            JR   Z,FIMIMP
            OUT  (#98),A
            INC  DE
            JR   IMPSTR
```

```
FIMIMP      EI
            POP   DE
            POP   BC
            POP   AF
            RET
POSIT       PUSH  AF
            PUSH  BC
            PUSH  HL
            PUSH  DE
            DI
            LD    A,H
            PUSH  AF
            LD    H,#00
            LD    A,L
            LD    E,A
            LD    D,#00
            LD    B,#27
POSIT1      ADD   HL,DE
            DJNZ  POSIT1
            POP   AF
            LD    D,#00
            LD    E,A
            ADD   HL,DE
            LD    A,L
            OUT   (#99),A
            LD    A,H
            OR    #40
            OUT   (#99),A
            POP   DE
            POP   HL
            POP   BC
            POP   AF
            RET
IMPLCAR     DI
IMPLCAR1    OUT   (#98),A
            DJNZ  IMPLCAR1
            EI
```

```
            RET
CONVCEN     PUSH DE
            PUSH BC
            PUSH AF
            LD   BC,#0064
            XOR  A
CONVCEN1    SBC  HL,BC
            INC  A
            JR   NC,CONVCEN1
            DEC  A
            ADD  A,#30
            ADD  HL,BC
            LD   (BUFFER),A
            LD   BC,#000A
            XOR  A
CONVDEZ     SBC  HL,BC
            INC  A
            JR   NC,CONVDEZ
            DEC  A
            ADD  A,#30
            ADD  HL,BC
            LD   (BUFFER+1),A
            LD   BC,#0001
            XOR  A
CONVUNI     SBC  HL,BC
            INC  A
            JR   NC,CONVUNI
            DEC  A
            ADD  A,#30
            LD   (BUFFER+2),A
            LD   A,#24
            LD   (BUFFER+3),A
            POP  AF
            POP  BC
            POP  DE
            RET
MENSG1      DEFM "COLOQUE O DISQUETE $"
```

```
MENSG2    DEFM "A$"
MENSG3    DEFM "B$"
MENSG4    DEFM "VERIFICANDO AS FORMATACOES$"
MENSG5    DEFM "DISQUETE    :$"
MENSG6    DEFM "SIMPLES$"
MENSG7    DEFM "DUPLA  $"
MENSG8    DEFM "COPIA DE SETORES$"
MENSG9    DEFM "LENDO SETOR  OOO$"
MENSGA    DEFM "GRAVANDO SETOR OOO$"
MENSGB    DEFM "E PRESSIONE QUALQUER TECLA$"
MENSGC    DEFM "                                        $"
MENSGD    DEFM "SETORES LIDOS :      A$"
MENSGE    DEFM "SETORES ESCRITOS :      A$"
MENSGF    DEFM "FORMATANDO O DISQUETE B$"
MENSGG    DEFM "DESEJA UMA NOVA COPIA ?$"
END       EQU  $
```

## 4.1 ANÁLISE DO PROGRAMA COPYSET

Comecemos analisando os labels dos equates (EQU).

| LABEL | FUNÇÃO |
|---|---|
| PRISET | Indica o primeiro setor a ser lido ou escrito |
| ULTSET | Indica o último setor a ser lido ou escrito |
| AUXSET | Buffer de dois bytes usado na impressão |
| NUMSET | Número total de setores a serem lidos |
| DENSA | Guarda o tipo de formatação do disco A |
| DENSB | Guarda o tipo de formatação do disco B |
| HFORM | Gancho (HOOK) do format |
| BUFFER | Guarda os caracteres ASCII correspondentes ao número do setor |
| CSRY | Guarda a ordenada do último caracter impresso |
| CALSLT | Endereço da subrotina para chamada inter-slot |
| PHYDIO | Endereço da subrotina para ler ou gravar os setores |
| BDOS | Endereço da BDOS no DISK-BASIC |

Você pode estar estranhando as quatro primeiras linhas do programa, vamos então analisá-las. Como você deve saber, os arquivos gerados pelo MSX-DOS, são diferentes dos arquivos gerados pelo DISK-BASIC (SAVE e BSAVE), uma vez que estes últimos apresentam uma espécie de HEADER. Desta forma, o primeiro byte de um arquivo gerado pelo comando SAVE (sem o ",A" no final), é o FFH, indicando que este arquivo deve ser carregado com o comando LOAD. Já um arquivo gerado pelo comando BSAVE, apresenta no seu início 7 bytes, conforme pode ser observado no quadro a seguir :

| BYTE | FUNÇÃO |
|---|---|
| 1 | FEH, indica arquivo binário |
| 2 e 3 | Indicam o endereço inicial |
| 4 e 5 | Indicam o endereço final |
| 6 e 7 | Indicam o endereço de execução |

Fica claro então, que as quatro primeiras linhas, nada mais fazem do que simular, no MSX-DOS, o comando BSAVE do DISK-BASIC.

Passemos agora, à análise das subrotinas que compõem o programa.

CLS : Esta subrotina é a responsável pela limpeza da tela.

QUADRO : Esta subrotina desenha um quadro na tela. A sua função é meramente estética.

VERFORM : Esta subrotina verifica a formatação dos discos A e B. Na saída o registro A contém o código da formatação do disco A, e o registro E, o código da formatação do disco B.

TESTFORM : Esta subrotina testa a necessidade ou não de se formatar o disco B, com a mesma formatação do disco A. Se por exemplo o disco A fôr de face simples, e o B, de face dupla, não há necessidade de se formatar.

COPYSET : Esta é a principal subrotina do programa. É ela que lê os setores, testa o fim da leitura, e grava o número exato de setores carregados na memória. Note que para ler e gravar os setores, fizemos uso do buffer destinado para isso, pelo DISK-BASIC. O início deste buffer é dado pelos endereços

F351H e F352H. Isto foi necessário, porque fomos alocando os setores na memória a partir do endereço 0100H, logo, quando chegássemos a qualquer endereço, compreendido no intervalo de 4000H a 7FFFH, o programa tentaria gravar os setores não na memória RAM, mas sim na memória ROM, da interface do drive. Quando fôr realizar os seus próprios programas, tome muito cuidado com esta característica do MSX, pois todas as rotinas de I/O do disk drive, estão na ROM da interface do disk drive.

NOVCOP : Esta subrotina é a responsável pela pergunta de uma nova cópia, no final do programa.

STRING : Esta subrotina é a responsável pela impressão de uma string.

POSIT : A função desta subrotina é posicionar o cursor nas coordenadas dadas pelo par HL.

IMPLCAR : Esta subrotina apenas imprime uma linha de caracteres iguais, cujo código se encontra no registro A, e o comprimento no registro B.

CONCEN : Esta subrotina converte para caracteres ASCII, as centenas presentes no par HL. Note que o maior setor a ser chamado, é o de número 719, no drive de dupla, e o de número 359 no drive de simples.

Note que fizemos uso do gancho (HFORM) do comando FORMAT, para descobrir o slot onde se encontra a interface controladora. Como você já deve saber, poderíamos, ao invés, ter usado o endereço F348H.

Agora, analise a listagem, e faça as modifi-

cações que achar necessárias, só assim você irá adquirir confiança suficiente para começar a produzir os seus próprios programas.

```
;====================================
;!    AUTOEXEC PARA DISK-BASIC      !
;!  PROIBIDA A REPRODUCAO PARA FINS !
;!            COMERCIAIS            !
;!  AUTOR: EDUARDO BARBOSA - 1988   !
;====================================

          DEFB  #FE
          DEFW  START
          DEFW  END
          DEFW  START
FCB       EQU   #F975
DMA       EQU   #9000
DMA1      EQU   #B000
BDOS      EQU   #F37D
CHPUT     EQU   #00A2
POSIT     EQU   #00C6
ERAFNK    EQU   #00CC
CLS       EQU   #00C3
CHGET     EQU   #009F
KILBUF    EQU   #0156
RUN       EQU   #73AC
CSRSW     EQU   #FCA9
BLOAD     EQU   #6EC6
KEYBUF    EQU   #FBF0
PUTPNT    EQU   #F3F8
GETPNT    EQU   #F3FA
VARTAB    EQU   #F6C2
CSRX      EQU   #F3DD
CSRY      EQU   #F3DC
BUFFER1   EQU   #F600
BUFFER2   EQU   #F601
BUFFER3   EQU   #F603
BUFFER4   EQU   #F604
BUFFER5   EQU   #F620
          ORG   #DA00
START     LD    HL,#9000
```

```
        LD      (HL),#00
        LD      DE,#9001
        LD      BC,#3FFF
        LDIR
        DI
        LD      HL,#0800
        LD      DE,#0C00
        LD      BC,#0400
LOOP    CALL    LEBYTE
        CALL    GRBYTE
        INC     DE
        INC     HL
        DEC     BC
        LD      A,B
        OR      C
        JR      NZ,LOOP
        EI
        CALL    CLS
        CALL    ERAFNK
        CALL    MENU
        LD      DE,DMA
        LD      C,1AH
        CALL    BDOS
        CALL    LEDENS
        LD      DE,#0005
        LD      C,2FH
        XOR     A
        CALL    BDOS
        CALL    DIRET
        JP      IMPRIM
LEBYTE  LD      A,L
        OUT     (#99),A
        LD      A,H
        OUT     (#99),A
        IN      A,(#98)
        RET
GRBYTE  PUSH    AF
```

```
        LD      A,E
        OUT     (#99),A
        LD      A,D
        OR      #40
        OUT     (#99),A
        POP     AF
        CPL
        OUT     (#98),A
        RET
MENU    LD      HL,#0D02
        CALL    POSIT
        LD      HL,MENSG1
MENU1   LD      A,(HL)
        CP      #2F
        RET     Z
        SET     7,A
        CALL    CHPUT
        INC     HL
        JR      MENU1
LEDENS  LD      DE,#0000
        LD      HL,#0100
        LD      C,#2F
        CALL    BDOS
        LD      HL,#9000
        LD      BC,#0015
        ADD     HL,BC
        LD      A,(HL)
        AND     #01
        JR      NZ,DUPLA
SIMPLES LD      HL,#0400
        RET
DUPLA   LD      HL,#0700
        RET
DIRET   LD      HL,#9000
        LD      DE,#A000
        XOR     A
        LD      (BUFFER1),A
```

```
DIRET1    LD     A,(HL)
          CP     ǂE5
          JR     Z,PROX
          CP     ǂ00
          JR     Z,RETURN
          LD     B,ǂ08
DIRET2    LD     A,(HL)
          LD     (DE),A
          INC    HL
          INC    DE
          DJNZ   DIRET2
          LD     A,ǂ2E
          LD     (DE),A
          LD     B,ǂ03
          INC    DE
DIRET3    LD     A,(HL)
          LD     (DE),A
          INC    DE
          INC    HL
          DJNZ   DIRET3
          LD     BC,ǂ0015
          ADD    HL,BC
          LD     A,(BUFFER1)
          INC    A
          LD     (BUFFER1),A
          JR     DIRET1
PROX      LD     BC,ǂ0020
          ADD    HL,BC
          JR     DIRET1
RETURN    RET
IMPRIM    LD     HL,ǂ0106
          CALL   POSIT
          XOR    A
          LD     (BUFFER3),A
          LD     (BUFFER5),A
          LD     HL,ǂA000
IMP1      LD     A,(HL)
```

```
            CP      00H
            JR      Z,FIMARQ
            LD      B,#0C
            LD      A,#20
            CALL    CHPUT
IMP2        LD      A,(HL)
            CALL    CHPUT
            INC     HL
            DJNZ    IMP2
            LD      A,(CSRX)
            CP      #28
            JR      NZ,CONIMP
            LD      A,(CSRY)
            INC     A
            LD      (CSRY),A
            LD      A,#01
            LD      (CSRX),A
CONIMP      LD      A,(BUFFER3)
            INC     A
            LD      (BUFFER3),A
            CP      #1E
            JR      Z,FIMPAG
            JR      IMP1
FIMARQ      LD      A,#01
            LD      (BUFFER5),A
FIMPAG      CALL    INVCUR
FIMPAG0     PUSH    HL
FIMPAG1     CALL    KILBUF
            CALL    CHGET
            POP     HL
            CP      #20
            JP      Z,PROXPAG
            CP      #0D
            JP      Z,CARREGA
            CP      #1C
            JP      Z,INCX
            CP      #1D
```

```
            JP      Z,DECX
            CP      #1E
            JP      Z,DECY
            CP      #1F
            JP      Z,INCY
            JR      FIMPAGO
PROXPAG     PUSH    HL
            LD      HL,#0106
            CALL    POSIT
            LD      B,#FF
LIMPA1      LD      A,#20
            CALL    CHPUT
            DJNZ    LIMPA1
            LD      B,#FF
LIMPA2      LD      A,#20
            CALL    CHPUT
            DJNZ    LIMPA2
            LD      HL,#0106
            CALL    POSIT
            POP     HL
            XOR     A
            LD      (BUFFER3),A
            LD      A,(BUFFER5)
            CP      #01
            JP      NZ,IMP1
            JP      IMPRIM
CARREGA     PUSH    HL
            LD      HL,BUFFER4
            INC     HL
            PUSH    HL
            LD      HL,FCB
            LD      (HL),#00
            LD      DE,FCB+#01
            LD      BC,#0020
            LDIR
            POP     HL
            LD      DE,FCB
```

```
        LD.     A,#01
        LD      (DE),A
        INC     DE
        LD      B,#08
FAZFCB  LD      A,(HL)
        LD      (DE),A
        INC     HL
        INC     DE
        DJNZ    FAZFCB
        INC     HL
        LD      B,#03
FAZFCB1 LD      A,(HL)
        LD      (DE),A
        INC     DE
        INC     HL
        DJNZ    FAZFCB1
        LD      DE,DMA1
        LD      C,#1A
        CALL    BDOS
        LD      DE,FCB
        LD      C,#0F
        CALL    BDOS
        LD      DE,FCB
        LD      C,#14
        CALL    BDOS
        LD      DE,FCB
        LD      C,#10
        CALL    BDOS
        LD      HL,DMA1
        LD      A,(HL)
        CP      #FE
        JP      Z,BINARIO
        CP      #FF
        JP      Z,BASIC
        POP     HL
        JP      FIMPAGO
BASIC   POP     HL
```

```
                LD      HL,FCB+#0C
                LD      (HL),#00
                LD      DE,FCB+#0D
                LD      BC,#0020
                LDIR
                LD      DE,FCB
                LD      C,#0F
                CALL    BDOS
                LD      DE,#8000
BASIC1          PUSH    DE
                LD      C,#1A
                CALL    BDOS
                LD      DE,FCB
                LD      C,#14
                CALL    BDOS
                CP      #01
                POP     DE
                JR      Z,BASIC2
                EX      DE,HL
                LD      BC,#0080
                ADD     HL,BC
                EX      DE,HL
                JR      BASIC1
BASIC2          LD      HL,#8000
                LD      BC,#8000
BASIC3          LD      A,#00
                CPIR
                LD      A,(HL)
                CP      #00
                JR      NZ,BASIC3
                INC     HL
                LD      A,(HL)
                CP      #00
                JR      NZ,BASIC3
BASIC4          INC     HL
                LD      (VARTAB),HL
                LD      HL,#8000
```

```
        LD      (HL),#00
        JP      RUN
BINARIO POP     HL
        LD      HL,BUFFER4
        LD      (HL),#22
        LD      BC,#000D
        ADD     HL,BC
        LD      (HL),#22
        INC     HL
        LD      (HL),#2C
        INC     HL
        LD      (HL),#52
        INC     HL
        LD      (HL),#00
        LD      HL,BUFFER4
        JP      BLOAD
INCX    LD      A,(CSRX)
        LD      DE,(BUFFER2)
        ADD     A,#0D
        CP      D
        JR      NZ,INCX1
        LD      A,(CSRY)
        CP      E
        JP      Z,FIMPAGO
INCX1   LD      A,(CSRX)
        CP      #1B
        JP      Z,FIMPAGO
        ADD     A,#0D
        PUSH    AF
        CALL    INVNOME
        POP     AF
        LD      (CSRX),A
        CALL    INVNOME
        JP      FIMPAGO
DECX    LD      A,(CSRX)
        CP      #01
        JP      Z,FIMPAGO
```

```
        SUB   #0D
        PUSH  AF
        CALL  INVNOME
        POP   AF
        LD    (CSRX),A
        CALL  INVNOME
        JP    FIMPAGO
INCY    LD    DE,(BUFFER2)
        LD    A,(CSRY)
        INC   A
        CP    #10
        JP    Z,FIMPAGO
        DEC   A
        CP    E
        JP    Z,FIMPAGO
        INC   A
        CP    E
        JR    NZ,INCY1
        LD    A,D
        SUB   #0D
        LD    D,A
        LD    A,(CSRX)
        CP    D
        JR    Z,INCY1
        JP    NC,FIMPAGO
INCY1   LD    A,(CSRY)
        CP    #10
        JP    Z,FIMPAGO
        INC   A
        PUSH  AF
        CALL  INVNOME
        POP   AF
        LD    (CSRY),A
        CALL  INVNOME
        JP    FIMPAGO
DECY    LD    A,(CSRY)
        CP    #06
```

```
            JP      Z,FIMPAGO
            DEC     A
            PUSH    AF
            CALL    INVNOME
            POP     AF
            LD      (CSRY),A
            CALL    INVNOME
            JP      Z,FIMPAGO
INVCUR      LD      DE,(CSRY)
            LD      (BUFFER2),DE
            LD      DE,#0106
            LD      (CSRY),DE
            CALL    INVNOME
            RET
INVNOME     DI
            LD      DE,(CSRY)
            DEC     D
            DEC     E
            LD      B,E
            PUSH    HL
            LD      HL,#0000
            LD      A,B
            CP      #00
            JR      Z,FIMCAL
            PUSH    DE
            LD      DE,#0028
CALCULO     ADD     HL,DE
            DJNZ    CALCULO
            POP     DE
FIMCAL      LD      E,D
            LD      D,#00
            ADD     HL,DE
            EX      DE,HL
            POP     HL
            LD      A,E
            OUT     (#99),A
            LD      A,D
```

```
            OUT     (#99),A
            LD      B,#OD
            PUSH    HL
            LD      HL,BUFFER4
BUSCA       IN      A,(#98)
            LD      (HL),A
            INC     HL
            DJNZ    BUSCA
INVERTE     LD      HL,BUFFER4
            LD      A,E
            OUT     (#99),A
            LD      A,D
            OR      #40
            OUT     (#99),A
            LD      B,#OD
INVERT1     LD      A,(HL)
            BIT     7,A
            JR      Z,INVERT2
            RES     7,A
            JR      INVERT3
INVERT2     SET     7,A
INVERT3     OUT     (#98),A
            INC     HL
            DJNZ    INVERT1
            POP     HL
            EI
            RET
MENSG1      DEFM    'DIRETORIO  ATUAL/'
END         EQU     $
```

## 4.2 ANÁLISE DO PROGRAMA AUTOEXEC

Comecemos então, com a análise dos equates.

| LABEL | FUNÇÃO |
|---|---|
| FCB | Buffer destinado ao FCB |
| DMA | Buffer destinado aos setores que compõem o diretório |
| DMA1 | Buffer destinado aos primeiros 128 bytes de um arquivo |
| BDOS | Endereço da BDOS no DISK-BASIC |
| CHPUT | Subrotina da ROM que imprime na tela, o caracter cujo código se encontra no registro A |
| POSIT | Subrotina da ROM que posiciona o cursor, nas coordenadas dadas pelo par HL |
| ERAFNK | Subrotina da ROM que apaga as teclas de função do vídeo |
| CLS | Subrotina da ROM que limpa o vídeo |
| CHGET | Subrotina da ROM que lê o teclado. O código da tecla retorna no registro A |
| KILBUF | Subrotina da ROM que limpa o buffer do teclado, para que não seja efetuada uma leitura errada |
| RUN | Endereço da ROM equivalente ao comando RUN |
| BLOAD | Endereço na ROM do comando BLOAD |
| CSRSW | Endereço das variáveis do sistema, responsável pelo aparecimento ou não do cursor |
| KEYBUF | Label que contém o endereço do buffer do teclado |
| GETPNT | Endereço das variáveis do sistema, que contém o endereço (KEYBUF) do buffer do teclado |
| PUTPNT | Endereço das variáveis do sistema, que contém o endereço final do buffer do teclado mais um |
| VARTAB | Endereço das variáveis do sistema, que indica o endereço inicial das variáveis de um programa em BASIC, e portanto, o final deste mais um |

CSRY — Endereço das variáveis do sistema, que indica a ordenada do último ponto impresso na tela

CSRX — Endereço das variáveis do sistema, que indica a abscissa do último ponto impresso na tela mais um

BUFFER1 — Endereço que contém o número de arquivos do disco

BUFFER2 — Endereço reservado. Não é usado nesta versão

BUFFER3 — Endereço que indica o número máximo de arquivos a serem exibidos na tela (30)

BUFFER4 — Endereço que contém o nome completo do arquivo sob o cursor

BUFFER5 — Endereço que indica se estamos ou não na última página de arquivos.

Passemos então, à análise das subrotinas que compõem o programa.

LOOP : Esta subrotina, juntamente com as subrotinas LEBYTE e GRBYTE, é a responsável pela inversão dos primeiros 128 caracteres, e o consequente deslocamento destes, para a àrea dos últimos 128 caracteres. Deste modo, se quisermos inverter o caracter "A" bastará setarmos o bit 7 do código (41H) correspondente ao caracter "A".

MENU : Esta subrotina imprime em vídeo reverso, a mensagem "DIRETORIO ATUAL".

LEDENS : Esta subrotina verifica o tipo de formatação do disco, através do vigésimo primeiro byte do setor zero (BOOT). Note que o programa anterior (COPYSET), fez uso de outra rotina (preferível), para o mesmo fim.

DIRET : Esta subrotina é a responsável pela leitu-

ra do diretório. Desta forma, ela inicia a busca dos arquivos existentes a partir do endereço 9000H. A busca é feita, comparando se o byte inicial de um arquivo,é diferente de E5H (arquivo deletado), ou diferente de 00H (fim do diretório). Assim sendo, se o primeiro byte de um arquivo fôr diferente de E5H ou 00H, então o nome do arquivo é transferido para o buffer que inicia em A000H. Lembre-se que um arquivo, mesmo que esteja apagado, gasta 32 (20H) bytes no diretório, sendo que os 11 primeiros, constituem o nome do arquivo.

IMPRIM : Esta subrotina imprime no vídeo, 30 arquivos por vez.

FIMPAG : Esta subrotina é o cérebro do programa. É ela quem detecta as teclas do cursor, para mudarmos de arquivo, detecta se a barra de espaço foi pressionada para podermos visualizar os demais arquivos não exibidos, e também detecta se a tecla RETURN foi pressionada, o que por sua vez indicaria a intenção de carregar o arquivo selecionado.

CARREGA : Esta subrotina carrega os primeiros 128 bytes do arquivo selecionado, e verifica se o arquivo foi gerado pelo comando SAVE, ou pelo comando BSAVE. Na primeira hipótese, o programa é desviado para a subrotina BASIC, e na segunda hipótese é desviado para a subrotina BINARIO. Se o arquivo não se enquadrar nas exigências do teste, o programa é desviado para a subrotina FIMPAG.

BASIC : Esta subrotina promove o carregamento sequencial do programa em BASIC. No final

do carregamento, promove a organização das variáveis do sistema, e se desvia para o endereço do comando RUN na ROM do MSX.

BINÁRIO : Esta subrotina aproveita o comando BLOAD da ROM do MSX, para carregar o programa selecionado.

INVNOME : Esta subrotina inverte o nome do arquivo selecionado, e o transfere para o endereço indicado pelo BUFFER4. Observe que o nome é captado da própria VRAM, portanto, não espere que este programa funcione com a expansão de 80 colunas.

Como você já deve ter sentido, muita coisa pode ser melhorada neste programa. A versão aqui apresentada é a primeira de uma série de 5, sendo que na quinta já é possível o carregamento até de programas em MSX-DOS a partir do DISK-BASIC. Um detalhe que não pode deixar de ser mencionado, é que para carregar convenientemente este arquivo, devemos gravá-lo no disco com o nome de AUTOEXEC.BIN, juntamente com o programa BASIC a seguir, que deverá ser gravado com o nome de AUTOEXEC.BAS.

```
10 WIDTH 40:KEY OFF:COLOR ,1,1
20 BLOAD"AUTOEXEC.BIN",R
```

Note que o endereço inicial de compilação é DAOOH, logo você deverá pressionar a tecla CONTROL ao ligar o seu MSX.

```
;====================================
;!   STATUS DO DISCO VERSAO 1.0     !
;! PROIBIDA A REPRODUCAO PARA FINS  !
;!           COMERCIAIS             !
;!  AUTOR : EDUARDO BARBOSA - 1988  !
;====================================

          DEFB #FE
          DEFW START
          DEFW END
          DEFW START
CALSLT    EQU  #001C
POSIT     EQU  #00C6
CHPUT     EQU  #00A2
ERAFNK    EQU  #00CC
INITXT    EQU  #006C
CHGET     EQU  #009F
CHSNS     EQU  #009C
KILBUF    EQU  #0156
MULTINT   EQU  #3193
CONVINT   EQU  #3425
HFORM1    EQU  #FFAC
CSRY      EQU  #F3DC
BDOS      EQU  #F37D
          ORG  #DA00
START     CALL INITXT
          CALL ERAFNK
          CALL QUADRO
          CALL MENSG1
          CALL STATUS
          CALL NOVDISC
          JR   Z,START
          CALL INITXT
          RET
QUADRO    LD   HL,#0005
          CALL LOCATE
          LD   A,#C0
```

```
            LD    B,#28
            CALL  IMPLCAR
            LD    HL,#0010
            CALL  LOCATE
            LD    A,#C3
            LD    B,#28
            CALL  IMPLCAR
            LD    B,#0A
            LD    HL,#0006
            LD    A,#C6
COLUNA1     PUSH  AF
            PUSH  HL
            CALL  LOCATE
            POP   HL
            POP   AF
            CALL  CHPUT
            INC   L
            DJNZ  COLUNA1
            LD    B,#0A
            LD    HL,#2706
            LD    A,#DE
COLUNA2     PUSH  AF
            PUSH  HL
            CALL  LOCATE
            POP   HL
            POP   AF
            CALL  CHPUT
            INC   L
            DJNZ  COLUNA2
            RET
MENSG1      LD    HL,#0C07
            LD    DE,MENSAG1
            CALL  STRING
            LD    HL,#0A0D
            LD    DE,MENSAG2
            CALL  STRING
            LD    HL,#070E
```

```
            LD    DE,MENSAG3
            CALL  STRING
            CALL  KILBUF
            CALL  CHGET
            RET
STATUS      LD    HL,#010D
            LD    DE,MENSAG4
            CALL  STRING
            LD    HL,#010E
            LD    DE,MENSAG4
            CALL  STRING
            LD    HL,#0209
            LD    DE,MENSAG6
            CALL  STRING
            LD    HL,#020B
            LD    DE,MENSAG7
            CALL  STRING
            LD    HL,#020D
            LD    DE,MENSAG8
            CALL  STRING
            LD    E,#01
            LD    C,#1B
            CALL  BDOS
            PUSH  AF
            PUSH  HL
            CP    #02
            JR    Z,DUPLA
            LD    HL,#1709
            LD    DE,MENSAG9
            CALL  STRING
            JR    STATUS1
DUPLA       LD    HL,#1709
            LD    DE,MENSAGA
            CALL  STRING
STATUS1     POP   HL
            POP   AF
            CP    #02
```

```
            JR    NZ,IMPSETL
            ADD   HL,HL
IMPSETL     PUSH  HL
            LD    DE,#0001
            CALL  MULTINT
            CALL  CONVINT
            PUSH  HL
IMPSLT1     LD    A,(HL)
            CP    #00
            JR    Z,IMPSLT2
            INC   HL
            JR    IMPSLT1
IMPSLT2     LD    (HL),#24
            POP   DE
            LD    HL,#120B
            CALL  LOCATE
            LD    C,#09
            CALL  BDOS
            POP   HL
            LD    DE,#0200
            CALL  MULTINT
            CALL  CONVINT
            PUSH  HL
IMPBYTE     LD    A,(HL)
            CP    #00
            JR    Z,IMPBYTE1
            INC   HL
            JR    IMPBYTE
IMPBYTE1    LD    (HL),#24
            POP   DE
            LD    HL,#100D
            CALL  LOCATE
            LD    C,#09
            CALL  BDOS
            CALL  KILBUF
ESPERA      CALL  CHSNS
            JR    Z,ESPERA
```

```
            RET
NOVDISC     LD    HL,#0106
            LD    B,#09
NOVDISC1    PUSH  HL
            PUSH  BC
            LD    DE,MENSAG4
            CALL  STRING
            POP   BC
            POP   HL
            INC   L
            DJNZ  NOVDISC1
            LD    HL,#070B
            LD    DE,MENSAG5
            CALL  STRING
            CALL  KILBUF
            CALL  CHGET
            AND   #5F
            CP    #53
            RET
IMPLCAR     PUSH  AF
            CALL  CHPUT
            POP   AF
            DJNZ  IMPLCAR
            EI
            RET
LOCATE      INC   L
            INC   H
            CALL  POSIT
            RET
STRING      INC   L
            INC   H
            CALL  POSIT
            LD    C,#09
            CALL  BDOS
            RET
MENSAG1     DEFM  "STATUS DO DISCO$"
MENSAG2     DEFM  "COLOQUE O DISQUETE A$"
```

```
MENSAG3   DEFM "E PRESSIONE QUALQUER TECLA$"
MENSAG4   DEFM "                                    $"
MENSAG5   DEFM "DESEJA UMA NOVA CONSULTA ?$"
MENSAG6   DEFM "TIPO DE FORMATACAO :$"
MENSAG7   DEFM "SETORES LIVRES :$"
MENSAG8   DEFM "BYTES LIVRES :$"
MENSAG9   DEFM "SIMPLES$"
MENSAGA   DEFM "DUPLA$"
END       EQU  $
```

## 4.3 ANÁLISE DO PROGRAMA STATUS DO DISCO

Analisemos primeiramente, os equates.

| LABEL | FUNÇÃO |
|---|---|
| CALSLT | Subrotina do MSX empregada na chamada de outras subrotinas, em outros slots |
| POSIT | Subrotina do MSX que posiciona o cursor nas coordenadas dadas pelo par HL |
| CHPUT | Subrotina do MSX que imprime no vídeo o caracter cujo código se encontra no registro A |
| ERAFNK | Subrotina do MSX que apaga a exibição das teclas de função |
| INITXT | Subrotina do MSX que inicializa o VDP para a tela de 40 colunas |
| CHGET | Subrotina do MSX que lê o teclado |
| CHSNS | Subrotina do MSX que detecta se alguma tecla foi pressionada |
| KILBUF | Subrotina do MSX que limpa o buffer do teclado |
| MULTINT | Subrotina do interpretador BASIC que multiplica o conteúdo do par HL pelo conteúdo do par DE, e guarda o resultado no DAC |
| CONVINT | Subrotina do interpretador BASIC que converte o número contido no DAC, para caracteres ASCII. No retorno o par HL aponta para o primeiro caracter |
| HFORM | Gancho do comando FORMAT |
| CSRY | Endereço das variáveis do sistema, que contém a ordenada do último caracter impresso na tela |
| BDOS | Endereço da BDOS no DISK-BASIC |

Analisemos agora, a função de cada uma das subrotinas que compõem o programa.

QUADRO : Esta subrotina desenha um quadro na tela, entre os pontos (0,5) e (39,16).

MENSG1 : Esta subrotina imprime as mensagens de abertura do programa.

STATUS : Esta subrotina imprime o tipo de formatação, o número de setores livres e o número de bytes livres. Para tanto, esta subrotina faz uso da função 1BH da BDOS.

ESPERA : Esta subrotina espera que alguma tecla seja pressionada para que o programa se desvie para a subrotina NOVDISC.

NOVDISC : Esta subrotina é a responsável pela pergunta de uma nova consulta.

IMPLCAR : Esta subrotina imprime uma linha de caracteres iguais cujo código se encontra no registro A, e o tamanho no B.

LOCATE : Esta subrotina posiciona o cursor na posição definida pelo par HL.

STRING : Esta subrotina imprime uma string no vídeo usando a função 09H da BDOS.

```
;====================================
;!       CMDTYPE VERSAO 1.0          !
;! PROIBIDA A REPRODUCAO PARA FINS   !
;!           COMERCIAIS              !
;! AUTOR : EDUARDO BARBOSA - 1988    !
;====================================

          DEFB #FE
          DEFW START
          DEFW END
          DEFW ENTRY
DMA       EQU  #F351
FCB       EQU  #F975
BUFFER1   EQU  #F600
BUFFER2   EQU  #F602
BDOS      EQU  #F37D
HCMD      EQU  #FE0D
CHPUT     EQU  #00A2
          ORG  #DA00
START     INC  HL
          LD   A,(HL)
          AND  #5F
          CP   "T"
          RET  NZ
          INC  HL
          LD   A,(HL)
          CP   ":"
          RET  NZ
          INC  HL
          CALL INICIO
          INC  SP
          INC  SP
          RST  #10
          RET
INICIO    PUSH HL
          LD   HL,FCB
          LD   (HL),#00
```

```
            LD    DE,FCB+#01
            LD    BC,#0030
            LDIR
            LD    DE,(DMA)
            LD    C,#1A
            CALL  BDOS
            LD    DE,FCB
            LD    A,#01
            LD    (DE),A
            INC   DE
            PUSH  DE
            EX    DE,HL
            POP   DE
            PUSH  DE
            INC   DE
            LD    (HL),#20
            LD    BC,#000A
            LDIR
            POP   DE
            POP   HL
TRANSF      LD    A,(HL)
            CP    #22
            JR    Z,FIMTRAN
            CP    #2E
            JR    Z,FIMNOME
            CP    #41
            JR    C,CONTTRA
            AND   #5F
CONTTRA     LD    (DE),A
            INC   HL
            INC   DE
            JR    TRANSF
FIMNOME     LD    DE,FCB+#09
            INC   HL
FIMNOME1    LD    A,(HL)
            CP    #22
            JR    Z,FIMTRAN
```

```
            CP    ǂ41
            JR    C,CONTFIM
            AND   ǂ5F
CONTFIM     LD    (DE),A
            INC   HL
            INC   DE
            JR    FIMNOME1
FIMTRAN     PUSH  HL
            LD    DE,FCB
            LD    C,ǂOF
            CALL  BDOS
            INC   A
            JR    Z,ERRO1
IMPARQ      LD    DE,FCB
            LD    C,ǂ14
            CALL  BDOS
            CP    ǂ02
            JR    Z,ERRO2
            DEC   A
            JR    Z,FIMARQ
            CALL  IMPRIME
            JR    IMPARQ
FIMARQ      CALL  IMPRIME
            POP   HL
            RET
IMPRIME     LD    B,ǂ8O
            LD    HL,(DMA)
IMPRIME1    LD    A,(HL)
            CP    ǂ1A
            RET   Z
            CP    ǂOA
            JR    Z,CONTIMP
            CP    ǂOD
            JR    Z,CONTIMP
            CP    ǂ2O
            RET   C
            CP    ǂ7F
```

```
          RET   NC
CONTIMP   CALL  CHPUT
          PUSH  HL
          PUSH  BC
          LD    C,#0B
          CALL  BDOS
          POP   BC
          POP   HL
          INC   HL
          DJNZ  IMPRIME1
          RET
ERRO1     LD    DE,MENSG1
          LD    C,#09
          CALL  BDOS
          POP   HL
          RET
ERRO2     LD    DE,MENSG2
          LD    C,#09
          CALL  BDOS
          POP   HL
          RET
MENSG1    DEFM  "ARQUIVO INEXISTENTE"
          DEFB  #0A,#0D,#24
MENSG2    DEFM  "ERRO NO CARREGAMENTO"
          DEFB  #0A,#0D,#24
ENTRY     LD    HL,HCMD
          LD    (HL),#C3
          INC   HL
          LD    DE,START
          LD    (HL),E
          INC   HL
          LD    (HL),D
          RET
END       EQU   $
```

## 4.4 ANÁLISE DO PROGRAMA CMDTYPE

Comecemos por analisar os equates.

| LABEL | FUNÇÃO |
|---|---|
| DMA | O conteúdo deste endereço (F351H), indica-nos a posição inicial da memória RAM utilizada pelo DMA |
| FCB | Endereço do FCB na memória RAM |
| BUFFER1 | Buffer auxiliar não utilizado nesta versão |
| BUFFER2 | Buffer auxiliar não utilizado nesta versão |
| BDOS | Endereço da BDOS no DISK-BASIC |
| HCMD | Endereço do gancho do comando CMD |
| CHPUT | Endereço da subrotina da ROM do MSX, que imprime um caracter |

Analisemos agora, cada subrotina separadamente.

ENTRY : Esta subrotina é a primeira a ser executada após o carregamento. A sua função é desviar o HOOK do comando CMD para o endereço DA00H.

START : A função desta subrotina, é a de verificar a sintaxe do comando CMD, implementado por nós. Note que quando o HOOK desvia o programa para o endereço DA00H, o par HL já está apontando para o primeiro caracter após o comando CMD, que no nosso caso são as aspas (22H). Desta forma, a subrotina START incrementa o par HL, e verifica se o seu conteúdo é a letra "T". A instrução AND 5F tem como função transformar minúsculas em maiúsculas. Se o conteúdo do par HL fôr diferente da letra "T", o programa

retorna para o interpretador BASIC com uma mensagem de erro. Da mesma forma, o programa compara se o próximo caracter, é o dois pontos. Se a sintaxe estiver correta, o programa se desvia para a rotina INICIO.

INICIO : Esta subrotina tem a função de zerar o FCB, e indicar ao sistema a àrea que será utilizada pelo DMA.

TRANSF : Esta subrotina termina em FIMTRAN, e tem como função transferir o nome do arquivo para o FCB. Note que no final, esta subrotina verifica se o arquivo existe.

IMPARQ : Esta subrotina realiza a leitura sequencial do arquivo, e imprime na tela cada grupo de 128 bytes lidos. Note que a opção CONTROl-S está disponível para interrompermos o processamento. No final desta subrotina o arquivo é fechado, e retornamos ao interpretador BASIC. Note que no retorno ao BASIC, tivemos de destruir o conteúdo da pilha, para que o sistema não se desviasse para a mensagem de erro.

ERRO1 : Esta subrotina imprime a mensagem de arquivo inexistente.

ERRO2 : Esta subrotina imprime a mensagem de erro no carregamento.

```
;====================================
;!      SPEED DRIVE VERSAO 1.0       !
;! PROIBIDA A REPRODUCAO PARA FINS   !
;!            COMERCIAIS             !
;!  AUTOR : EDUARDO BARBOSA - 1988   !
;====================================

INITXT     EQU  ╪006C
ERAFNK     EQU  ╪00CC
POSIT      EQU  ╪00C6
CHPUT      EQU  ╪00A2
BREAKX     EQU  ╪00B7
TOKLIN     EQU  ╪42B2
PRINT      EQU  ╪4A24
BDOS       EQU  ╪F37D
BUF        EQU  ╪F55E
KBUF       EQU  ╪F41F
BUFFER1    EQU  ╪F600
BUFFER2    EQU  ╪E000
           DEFB ╪FE
           DEFW START
           DEFW END
           DEFW START
           ORG  ╪DA00
START      EXX
           CALL INITXT
           CALL ERAFNK
           CALL TOKEN
           CALL QUADRO
           CALL MENSG1
           CALL VELOC
           CALL INITXT
           XOR  A
           OUT  (╪D4),A
           EXX
           RET
TOKEN      LD   HL,LINHA
```

```
            LD    DE,BUF
            LD    BC,ENDLI-LINHA+#01
            LDIR
            LD    HL,BUF
            CALL  TOKLIN
            INC   HL
            LD    DE,BUFFER2
            LDIR
            LD    HL,KBUF
            LD    DE,KBUF+#01
            LD    BC,317
            LD    (HL),#00
            LDIR
            LD    HL,BUF
            LD    DE,BUF+#01
            LD    BC,258
            LD    (HL),#20
            LDIR
            RET
QUADRO      LD    HL,#0106
            CALL  POSIT
            LD    A,#C0
            LD    B,#28
            CALL  IMPLCAR
            LD    HL,#0111
            CALL  POSIT
            LD    A,#C3
            LD    B,#28
            CALL  IMPLCAR
            LD    HL,#0107
            LD    A,#C6
            LD    B,#0A
COLUNA1     PUSH  AF
            CALL  POSIT
            POP   AF
            CALL  CHPUT
            INC   L
```

```
            DJNZ COLUNA1
            LD   HL,#2807
            LD   A,#DE
            LD   B,#0A
COLUNA2     PUSH AF
            CALL POSIT
            POP  AF
            CALL CHPUT
            INC  L
            DJNZ COLUNA2
            RET
IMPLCAR     PUSH AF
            CALL CHPUT
            POP  AF
            DJNZ IMPLCAR
            RET
MENSG1      LD   HL,#0A08
            CALL POSIT
            LD   DE,MENSAG1
            LD   C,#09
            CALL BDOS
            LD   HL,#0B0E
            CALL POSIT
            LD   DE,MENSAG2
            LD   C,#09
            CALL BDOS
            LD   HL,#080F
            CALL POSIT
            LD   DE,MENSAG3
            LD   C,#09
            CALL BDOS
            LD   C,#08
            CALL BDOS
            LD   HL,#020E
            CALL POSIT
            LD   A,#20
            LD   B,#20
```

```
            CALL IMPLCAR
            LD   HL,#020F
            CALL POSIT
            LD   A,#20
            LD   B,#20
            CALL IMPLCAR
            LD   HL,#060F
            CALL POSIT
            LD   DE,MENSAG5
            LD   C,#09
            CALL BDOS
            LD   HL,#060B
            CALL POSIT
            LD   DE,MENSAG4
            LD   C,#09
            CALL BDOS
            RET
VELOC       LD   A,#D0
            OUT  (#D0),A
            LD   A,#21
            OUT  (#D4),A
ESPERA      IN   A,(#D0)
            RRA
            JR   C,ESPERA
INICIO      LD   HL,#0000
            DI
LOOP1       IN   A,(#D0)
            BIT  1,A
            JR   NZ,LOOP1
LOOP2       IN   A,(#D0)
            BIT  1,A
            JR   Z,LOOP2
LOOP3       INC  HL
            IN   A,(#D0)
            BIT  1,A
            JR   NZ,LOOP3
LOOP4       INC  HL
```

```
          IN    A,(#D0)
          BIT   1,A
          JR    Z,LOOP4
          LD    (BUFFER1),HL
          EI
          LD    HL,#1A0B
          CALL  POSIT
          LD    HL,BUFFER2
          CALL  PRINT
          CALL  LETECLA
          RET   Z
          JR    INICIO
LETECLA   IN    A,(#AA)
          AND   %11110000
          OR    %00000111
          OUT   (#AA),A
          IN    A,(#A9)
          BIT   2,A
          RET
LINHA     DEFM  "USING"
          DEFB  #22
          DEFM  "###.###"
          DEFB  #22
          DEFM  ";5124324.3/(PEEK(&HF600)+256*PEEK(&HF601))"
ENDLI     DEFB  #00
MENSAG1   DEFM  "TESTE DE VELOCIDADE$"
MENSAG2   DEFM  "COLOQUE O DISQUETE$"
MENSAG3   DEFM  "PRESSIONE QUALQUER TECLA$"
MENSAG4   DEFM  "VELOCIDADE (RPM) :$"
MENSAG5   DEFM  "PRESSIONE 'ESC' PARA TERMINAR$"
END       EQU   $
```

```
;====================================
;!  SPEED DRIVE SHARP  VERSAO 1.0   !
;! PROIBIDA A REPRODUCAO PARA FINS  !
;!           COMERCIAIS             !
;!  AUTOR : EDUARDO BARBOSA - 1988  !
;====================================

INITXT    EQU  #006C
ERAFNK    EQU  #00CC
POSIT     EQU  #00C6
CHPUT     EQU  #00A2
BREAKX    EQU  #00B7
TOKLIN    EQU  #42B2
PRINT     EQU  #4A24
BDOS      EQU  #F37D
BUF       EQU  #F55E
KBUF      EQU  #F41F
HFORM     EQU  #FFAC
BUFFER1   EQU  #F600
BUFFER2   EQU  #E000
          DEFB #FE
          DEFW START
          DEFW END
          DEFW START
          ORG  #DA00
START     EXX
          CALL INITXT
          CALL ERAFNK
          CALL TOKEN
          CALL QUADRO
          CALL MENSG1
          CALL VELOC
          CALL INITXT
          IN   A,(#A8)
          PUSH AF
          CALL CHAVEIA
          XOR  A
```

```
            INC  A
            LD   (#7FFD),A
            POP  AF
            DI
            OUT  (#A8),A
            EI
            EXX
            RET
TOKEN       LD   HL,LINHA
            LD   DE,BUF
            LD   BC,ENDLI-LINHA+#01
            LDIR
            LD   HL,BUF
            CALL TOKLIN
            INC  HL
            LD   DE,BUFFER2
            LDIR
            LD   HL,KBUF
            LD   DE,KBUF+#01
            LD   BC,317
            LD   (HL),#00
            LDIR
            LD   HL,BUF
            LD   DE,BUF+#01
            LD   BC,258
            LD   (HL),#20
            LDIR
            RET
QUADRO      LD   HL,#0106
            CALL POSIT
            LD   A,#C0
            LD   B,#28
            CALL IMPLCAR
            LD   HL,#0111
            CALL POSIT
            LD   A,#C3
            LD   B,#28
```

```
            CALL IMPLCAR
            LD   HL,#0107
            LD   A,#C6
            LD   B,#0A
COLUNA1     PUSH AF
            CALL POSIT
            POP  AF
            CALL CHPUT
            INC  L
            DJNZ COLUNA1
            LD   HL,#2807
            LD   A,#DE
            LD   B,#0A
COLUNA2     PUSH AF
            CALL POSIT
            POP  AF
            CALL CHPUT
            INC  L
            DJNZ COLUNA2
            RET
IMPLCAR     PUSH AF
            CALL CHPUT
            POP  AF
            DJNZ IMPLCAR
            RET
MENSG1      LD   HL,#0A08
            CALL POSIT
            LD   DE,MENSAG1
            LD   C,#09
            CALL BDOS
            LD   HL,#0B0E
            CALL POSIT
            LD   DE,MENSAG2
            LD   C,#09
            CALL BDOS
            LD   HL,#080F
            CALL POSIT
```

```
            LD    DE,MENSAG3
            LD    C,#09
            CALL  BDOS
            LD    C,#08
            CALL  BDOS
            LD    HL,#020E
            CALL  POSIT
            LD    A,#20
            LD    B,#20
            CALL  IMPLCAR
            LD    HL,#020F
            CALL  POSIT
            LD    A,#20
            LD    B,#20
            CALL  IMPLCAR
            LD    HL,#060F
            CALL  POSIT
            LD    DE,MENSAG5
            LD    C,#09
            CALL  BDOS
            LD    HL,#060B
            CALL  POSIT
            LD    DE,MENSAG4
            LD    C,#09
            CALL  BDOS
            RET
CHAVEIA     DI
            LD    HL,(HFORM)
            LD    A,H
            RLCA
            RLCA
            AND   %00001100
            LD    E,A
            IN    A,(#A8)
            AND   %11110011
            OR    E
            OUT   (#A8),A
```

```
            RET
VELOC       DI
            IN    A,(#A8)
            PUSH  AF
            CALL  CHAVEIA
            LD    A,#82
            LD    (#7FFD),A
            LD    BC,#0000
ESPERA      DEC   BC
            LD    A,B
            OR    C
            JR    NZ,ESPERA
INICIO      LD    HL,#0000
LOOP1       LD    A,(#7FF8)
            BIT   1,A
            JR    NZ,LOOP1
LOOP2       LD    A,(#7FF8)
            BIT   1,A
            JR    Z,LOOP2
LOOP3       INC   HL
            LD    A,(#7FF8)
            BIT   1,A
            JR    NZ,LOOP3
LOOP4       INC   HL
            LD    A,(#7FF8)
            BIT   1,A
            JR    Z,LOOP4
            LD    (BUFFER1),HL
            POP   AF
            OUT   (#A8),A
            EI
            LD    HL,#1BOB
            CALL  POSIT
            LD    HL,BUFFER2
            CALL  PRINT
            CALL  LETECLA
            RET   Z
```

```
            JR    VELOC
LETECLA     IN    A,(#AA)
            AND   %11110000
            OR    %00000111
            OUT   (#AA),A
            IN    A,(#A9)
            BIT   2,A
            RET
LINHA       DEFM "USING"
            DEFB #22
            DEFM "###.###"
            DEFB #22
            DEFM ";4861539/(PEEK(&HF600)+256*PEEK(&HF601))"
ENDLI       DEFB #00
MENSAG1     DEFM "TESTE DE VELOCIDADE$"
MENSAG2     DEFM "COLOQUE O DISQUETE$"
MENSAG3     DEFM "PRESSIONE QUALQUER TECLA$"
MENSAG4     DEFM "VELOCIDADE (RPM) :$"
MENSAG5     DEFM "PRESSIONE 'ESC' PARA TERMINAR$"
END         EQU  $
```

## 4.5 ANÁLISE DO PROGRAMA SPEED DRIVE

Comecemos por analisar os labels que compõem os programas SPEED DRIVE e SPEED DRIVE SHARP.

| LABEL | FUNÇÃO |
|---|---|
| INITXT | Subrotina da ROM do MSX equivalente ao comando SCREEN 0 do BASIC |
| ERAFNK | Subrotina da ROM do MSX que apaga as teclas de funções |
| POSIT | Subrotina da ROM do MSX que posiciona o cursor nas coordenadas dadas pelo par HL |
| CHPUT | Subrotina da ROM do MSX que imprime o caracter cujo código se encontra no registro A |
| BREAKX | Subrotina da ROM do MSX que detecta se as teclas CONTROL e STOP foram pressionadas. Não é utilizada nesta versão |
| TOKLIN | Subrotina do interpretador BASIC que "tokeniza" uma linha de programa. O texto da linha deverá estar contido a partir do endereço dado por BUF. No retorno a linha "tokenizada" se encontra a partir do endereço KBUF, sendo que o par HL aponta para o início da linha "tokenizada", e o par BC contém o seu comprimento |
| PRINT | Subrotina do interpretador BASIC que simula o comando PRINT. Ao chamá-la o par HL deverá apontar para o início da linha "tokenizada" |
| BUF | Endereço de memória a partir do qual fica armazenado o texto de uma linha em BASIC |
| KBUF | Endereço de memória a partir do qual fica armazenada a linha "tokenizada" |
| HFORMAT | Endereço do gancho do comando FORMAT, usado aqui para descobrir o slot da interface do drive |
| BDOS | Endereço de entrada da BDOS do DISK-BASIC |
| BUFFER1 | Buffer não utilizado nesta versão |

BUFFER2 Buffer que contém a linha "tokenizada"

Passemos agora, para a análise das subrotinas que compõem os programas.

TOKEN : Esta subrotina "tokeniza" a linha BASIC apontada pelo par HL. No final, a linha "tokenizada" é transferida para o BUFFER2.

QUADRO : Esta subrotina desenha um quadro entre as coordenadas (0,5) e (39,16).

MENSG1 : Esta subrotina imprime as mensagens de abertura e espera que alguma tecla seja pressionada para começar a medição.

VELOC : Esta subrotina é a responsável pelo cálculo da velocidade do drive. Vamos então explicar o seu funcionamento.

A primeira coisa a fazer é colocar o drive em movimento e esperar que este atinja a velocidade correta (ESPERA). O LOOP1 detecta se a luz de índice por um acaso está acesa. Se estiver, o programa permanece neste loop até que ela se apague. Este procedimento é necessário para que evitemos uma medição incorreta. Se a luz estiver apagada, o programa se desvia para o LOOP2. Este loop é executado até que a luz de índice seja detectada, quando então começa a medição da velocidade. O LOOP3 e o LOOP4 têm a mesma função dos loops 1 e 2 respectivamente. Assim sendo, no final do LOOP4 acabamos de detectar uma volta completa do disquete, e consequentemente do drive. Como vimos, a volta completa do drive é detectada pelos loops 3 e 4, logo

se conseguirmos determinar o tempo gasto nestes loops, obteremos uma quantidade que nos permitirá calcular a velocidade. A mágica é bastante simples como veremos.

Cada instrução do Z80 gasta um determinado número de ciclos de clock para se realizar, logo, se multiplicarmos este número pelo inverso do clock, obteremos o tempo em segundos gasto pela instrução. Observe no quadro abaixo, o número de ciclos gastos por cada uma das intruções que compõem os loops.

| INSTRUÇÃO | CICLOS |
|---|---|
| INC HL | 6 |
| IN A,(#DO) | 11 |
| BIT 1,A | 8 |
| JR Z,LOOP | 12 se Z=0 |
| JR NZ,LOOP | 12 se Z=1 |
| LD A,(#7FF8) | 13 |

Como se vê, os loops 3 e 4 gastam 37 ciclos (39 no caso da SHARP) a cada interação. Se multiplicarmos o número de ciclos pelo conteúdo do par HL, que atua como um contador de interações, teremos o tempo total (em ciclos) gasto numa volta. Logo o tempo gasto numa volta em segundos é :

39*(HL)/3.16E6 - No drive da Sharp
37*(HL)/3.16E6 - Nos demais drives

Basta-nos fazer então uma regra de três, para saber a velocidade do drive. Observe:

37*(HL)/3.16E6 seg -------- 1 volta
60 segundos -------- X voltas

logo X = 60*3.16E6/(37*(HL)) rpm

X = 5124324.3/(HL)

ou

X = 4861539/(HL) no drive da SHARP

Note que usamos a frequência do clock igual a 3.16MHz e não 3.58MHz. Esta diferença se estabeleceu, devido à necessidade de descontarmos o tempo perdido pelo Z80, em rotinas não mascaráveis.

**Eduardo Alberto R. S. Barbosa** é Engenheiro Eletricista formado pela Pontifícia Universidade Católica do Rio de Janeiro.

O interesse pela programação em assembly iniciou-se quando ao estudar a cadeira de Sistemas de Computação percebeu todas as potencialidades oferecidas pelo melhor microprocessador de 8 bits, o Z80. O primeiro passo foi comprar um microcomputador pessoal baseado no Z80. Na época, final de 83, o micro pessoal mais acessível era o TK-82C. Foi neste computador que o autor começou a desvendar, os segredos da linguagem de máquina.

Um ano após, Eduardo Alberto Barbosa, adquiriu um microcomputador DGT-100, com o qual desenvolveu ainda mais a programação em linguagem de máquina. O interesse pela linha MSX, veio no final de 85, com um convite da empresa CIBERNE para que o autor integrasse uma equipe de programadores destinada a desenvolver software nacional para o MSX.

Um ano após, o autor, juntamente com dois amigos, fundou a empresa TURBO EQUIPAMENTOS ELETRÔNICOS LTDA, ficando então responsável pela divisão de software, que atende exclusivamente a linha MSX.