RICHARD SPIEGEL

MICROINFORMÁTICA

ROTINAS PRONTAS MSX

LIVROS TÉCNICOS E CIENTÍFICOS EDITORA LTDA.

ROTINAS PRONTAS MSX 4

SÉRIE: MICROINFORMÁTICA

Coordenador Técnico:
ANTÔNIO ROBERTO RAMOS NOGUEIRA

AVERY	– Logo e o Apple
BARBUTO	– 35 Programas BASIC Para Microcomputadores
BARBUTO	– Programas BASIC Para o IBM PC
BHARUCHA	– dBASE II: Manual do Usuário
BOLOCAN	– Conquistado o Symphony
BOVE/RHODES/THOMAS	– DESKTOP PUBLISHING – A Arte da Editoração Eletrônica
BUI	– Planejamento Executivo com BASIC
CAMPBELL/SIMINOFF/YATES	– Micros de Lógica Sinclair
CRUZ	– Guia Prático de Sistemas no Apple II/IIe
DEWITT	– Arte e Gráficos no Apple II/IIe
DOBLER	– Linguagem C
EAGER	– Introdução ao PC DOS
ERSKINE	– Linguagem Assembly: 8086 e 8088
EVANS	– Norton Utilities Versão 4.0 – Uma Orientação Ilustrada
FINKEL/BROWN	– TRS-80 Programação Usando Arquivos de Dados
GARCIA/NOGUEIRA	– dBASE Total
GRILLO	– Turbo Pascal
GRILLO	– Programação e Técnicas TURBO PASCAL Versão 4.0
GRUENBERGER	– Programando no Apple
HYMAN	– Executando o OS/2
JANTZ	– Ventura Publisher Para o IBM PC
MILLER	– Dominando o TK 90X e TK 95
OLIVEIRA	– WordStar Para Micros de 8 e 16 Bits
PASSOS	– Framework II
PIMENTEL	– Apple: Assembly 6502
PIMENTEL	– Introdução ao TURBO BASIC
PRESS	– IBM PC AT e Suas Aplicações
ROBERTS	– Turbo Prolog
RUBEL	– Programando em dBASE III Plus
SCHNEIDER	– Multiplan – Manual do Usuário
SEABRA	– Meu Primeiro Encontro com o Microcomputador MSX
SEABRA	– Introdução ao Processador de Textos CARTA CERTA
SEABRA	– Domine o Drive do Seu MSX
SEABRA	– WordPerfect
SILVA/HEIBEL	– SuperCalc Total
SMITH	– IBM PC AT – Guia de Programação
SOUSA	– WordStar
SWENSSON	– Programando em Lotus 1-2-3 MACROS
SWENSSON/POMBEIRO	– Lotus 1-2-3
TABEIRA	– Criando Gráficos Profissionais com o MS CHART
TREVISAN	– Curso de Programação BASIC
VIDAL	– Clipper
WILLIAMS	– Lotus 1-2-3
ZAKIR	– Redes Locais

CONHEÇA TAMBÉM

ALBRECHT	– Análise Numérica
AMARAL	– Projeto Estruturado: Fundamentos e Técnicas
ARARIBÓIA	– Inteligência Artificial
BACK	– CP/M Sistema Operacional Para Microcomputadores
BASTOS	– Programação COBOL
BIANCHI	– Introdução aos Microcomputadores
BORATO	– BASIC Para Engenheiros e Cientistas
BORGES	– BASIC Aplicações Comerciais
BOSCH	– COBOL Fundamentos e Aplicações
CAMARÃO	– Glossário de Informática
CARDOSO	– Programação Estruturada em COBOL
CARVALHO	– Automação de Escritórios
CHINELATO	– Organização e Métodos Integrados à Informática
COBURN	– Informática na Educação
CORDEIRO PAULO	– COBOL – Técnicas e Dispositivos Especiais
COUCEIRO/BARRENECHA	– Sistemas de Gerência de Banco de Dados Distribuídos
DACHS	– Estatística Computacional
DAVIS	– Análise e Projeto de Sistemas
DIAS	– O Sistema de Informação e a Empresa
DIAS/GAZZANEO	– Projeto de Sistema e Processamento de Dados
EADIE	– Microcomputadores – Teoria e Prática
FERNANDES/KUGLER	– Gerência de Projetos de Sistemas
FILGUEIRAS/TORI/ ARAKAKI/MASSOLA	– Fundamentos de Computação Gráfica
FURTADO	– Teoria de Grafos
GANE	– Desenvolvimento Rápido de Sistemas
GANE/SARSON	– Análise Estruturada de Sistemas
GAINES/SHAN	– A Interação Computador-Usuário
GENARO	– Sistema Especialistas
GILLERSON/GOLDBERG	– Planejamento Estratégico, Análise de Sistema, Projeto de Bancos de Dados
GRILLO	– Programação Estruturada com FORTRAN e WATFIV
GUIMARÃES/LAGES	– Introdução à Ciência da Computação
GUIMARÃES/LAGES	– Algoritmos e Estruturas de Dados
HELT/BONFIM/MARTINS	– PECS – Estatísticas em Microcomputadores
KATZAN	– Segurança de Dados em Computação
LARSEN	– Computação Para Crianças
LOMUTO/LOMUTO	– Introdução ao Unix
MACHADO	– Mumps
MACIEL/MOREIRA/PINHEIRO	– Transcrição de Dados
MAYNARD	– Programação Modular
McCALEB	– A Microinformática na Empresa
McNITT	– Simulação em BASIC
MODULO	– Linguagem C
NETO	– Introdução à Compilação
NETO/PORTO/LIMA	– Introdução à Teleinformática
NOGUEIRA/GARCIA	– Avaliação e Seleção de Sistemas

PACITTI	– Programação, Princípios
PACITTI/ATIKINSON	– Programação e Métodos Computacionais, Vol 1 e 2
PEREIRA	– Computes, Grilo!
PERSIANO/OLIVEIRA	– Introdução à Computação Gráfica
PRAÇA	– COBOL Para Micros
PRATES	– BASIC Aplicado – Um Enfoque Profissional
PUCCINI/PIZZOLATO	– Programação Linear
RIBEIRO	– Introdução aos Sistemas Especialistas
SÁ CARVALHO	– Análise de Sistemas – O Outro Lado da Informática
SCHMITZ/TELES	– PASCAL e Técnicas de Programação
SOUZA	– Introdução à Linguagem BASIC
STRACK	– GPSS – Modelagem e Simulação de Sistemas
STRACK	– Sistemas de Processamento Distribuído
SUCESU	– Dicionário de Informática
TAROUCO	– Redes de Comunicação de Dados
VASCONCELLOS	– Computadores Eletrônicos e Processamento
VASCONCELLOS/SZERMAN	– O Centro de Processamento de Dados
VON STAA	– Engenharia de Programas
WARD	– Desenvolvendo Sistemas Sem Complicação
WEISS/KULIKOWSKI	– Guia Prático Para Projetar Sistemas Especialistas
WINSTON	– Inteligência Artificial
ZUCCHI	– Transmissão de Dados em Rede de Computadores
ZWICKER	– IBM BASIC

SÉRIE: CARTÃO DE REFERÊNCIA

Coordenador Técnico:
RUBENS PRATES

ALVES	– dBASE II
ALVES	– APPLE II
CINTRA LEITE	– Interrupções do MS-DOS/BIOS
CINTRA LEITE	– 8086/8088
CINTRA LEITE/PRATES	– Turbo Pascal Toolbox Versão 3.0
CINTRA LEITE/PRATES	Turbo Pascal Toolbox Versão 4.0
DICHAUNE	– WordStar Versão 3.45
HADA	– VisiCalc
INOUE/PRATES	– Lotus 1-2-3 Versão 1A
INOUE/PRATES	– Lotus 1-2-3 Versão 2
KLEIBER	– COBOL 80
PRATES	– CP/M
PRATES	– MS-DOS Versão 2.1
PRATES	– MS-DOS Versão 3.2
PRATES	– MS-DOS Versão 3.3
PRATES	– dBASE III
PRATES	– dBASE III Plus
PRATES	– MBASIC
PRATES	– PC BASIC
PRATES	– Turbo BASIC

PRATES	– GW BASIC
PRATES	– CP-500
PRATES	– MSX
PRATES	– Clipper Versão Autumn 86
PRATES	– Clipper Versão Summer 87
PRATES/COHEN	– DATAFLEX Versão 2.3
PRATES	– SISNE
PRATES	– Dialog Plus/C
PRATES	– WordStar Versão 4.0
PRATES	– NORTON UTILITIES Versão 4.0
PRATES/INOUE	– SuperCalc^4
PRATES/INOUE	– QUATTRO Versão 1.0
RAMALHO/PRATES	– Turbo Pascal Versão 3.0
SANTOS	– Z-80
SILVIA/HEIBEL	– SUPERCALC/SUPERCALC^2
VALENÇA	– OPEN ACCESS
ZIEGLER	– PageMaker Versão 3.0 Para IBM/PC e Compatíveis

ROTINAS PRONTAS MSX

RICHARD SPIEGEL

LIVROS
TÉCNICOS E
CIENTÍFICOS EDITORA LTDA.

Rio de Janeiro-RJ • São Paulo-SP

MSX é marca registrada da MICROSOFT CORPORATION.

Comissão de Computação:
Antônio Roberto Ramos Nogueira
Donaldo de Souza Dias
João José Neto
Luiz de Castro Martins
Ysmar Vianna e Silva Filho

Revisora de Texto:
Rosa Maria Oliveira de Queiroz

Capa:
AG Comunicação Visual Assessoria e Projetos Ltda.

Revisor de Provas:
Walter Veríssimo Croccia

CIP-Brasil. Catalogação-na-fonte
Sindicato Nacional dos Editores de Livros, RJ.

Spiegel, Richard
S734r Rotinas prontas MSX / Richard Spiegel. – Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora Ltda., 1989.
(Microinformática)

Bibliografia.
Apêndice.

1. MSX (Microcomputador). 2. Programação (Computadores). I. Título. II. Série.

CDD - 001.64
001.6424
CDU – 681.3
681.3.06

88-0867

ISBN: 85-216-0638-9

LIVROS TECNICOS E CIENTIFICOS EDITORA LTDA.

MATRIZ	FILIAL
Rua Vieira Bueno, 21 20.920 – Rio de Janeiro – RJ Brasil – End. Telegráfico: LITECE Tels.: 580-6055 Vendas: 580-9374	Rua Vitória, 486 – 2º andar 01.210 – São Paulo – SP Tel.: (011) 223-9866 Caixa Postal 4.817

SUMÁRIO

Introdução

Durante toda a minha vida profissional sempre procurei aprender a melhor maneira de programar, aquela que fosse mais rápida, ocupasse menos memória, etc. Mas nunca encontrei ninguém disposto a ensinar ou um livro que mostrasse alguns truques. Foi com o objetivo de eliminar essa lacuna que iniciei este livro.

Obviamente, não será possível esgotar o assunto num único livro, portanto, escolhi um micro que tivesse grande aceitação no mercado, fosse barato e poderoso. Assim, a escolha foi os micros padrão MSX, e por extensão todos os micros com padrão CP/M. (O Expert possui um sistema operacional que obedece praticamente a todas as funções do CP/M 2.2; e o Hot-bit possui o sistema operacional CP/M como opção). Na verdade, este trabalho está voltado principalmente para o Expert, pois foi o micro que comprei para poder testar todas as rotinas. Acredito que a maioria das rotinas funcionarão no Hot-bit, sendo que com toda certeza as que servirem para qualquer micro CP/M.

Como consultar este livro.

As rotinas estão agrupadas por função, assim basta procurar no resumo a página onde se iniciam as rotinas com a função desejada. Procura-se então a rotina desejada, consultando-se o resumo da rotina que está no início de cada rotina. Caso a rotina chame outras existentes no livro, as mesmas não serão escritas. Mas para serem usadas, é necessário que seu programa também tenha estas rotinas. Portanto, procure no Sumário onde encontrá-las e copie-as para o seu programa.

Capítulo 1

Conhecimentos Básicos

Definições:

Diretório: É uma área do disco onde estão armazenadas as informações sobre todos os arquivos existentes no disco. Contém informações sobre o tamanho de cada arquivo, onde está localizado no disco, etc...

FAT: Exclusivo para discos padrão MSX ou PC, é uma área do disco que complementa o diretório, com informações sobre a seqüência de aglomerados que formam cada arquivo.

FCB: É uma área de memória que contém informações sobre o arquivo que se está operando. Seu tamanho é de 36 bytes. O primeiro byte informa qual o disco que contém o arquivo (0 = default, 1 = A, 2 = B, ...); os 8 seguintes contêm o nome do arquivo em letras maiúsculas (bytes não usados deverão conter o byte 20H); os 3 seguintes à extensão do nome, obedecendo os mesmos critérios. Os outros 24 bytes possuem funções diversas, conforme a operação que está sendo executada. Devem ser zerados para as operações de abertura/fechamento/criação/apagamento de arquivo. Para renomear o arquivo devem ser zerados exceto aqueles que informam o novo nome.

Arquivo: É uma seqüência de bytes armazenados em algum meio de armazenamento (fita, disco, etc...) com significado lógico. Pode ser um programa, um texto, dados de algum programa, etc...

Registro: É uma seqüência de bytes de tamanho fixo e pré-definido. É um subgrupo do arquivo, com utilidade principalmente em arquivos de dados. Para o caso de escrita ou leitura no disco (padrão MSX ou CP/M) possui o tamanho de 128 bytes.

Setor: É uma subdivisão do disco. É onde está armazenada a menor seqüência de bytes consecutivos no disco. Esta seqüência não pode ser dividida em outras menores para fins de armazenamento no disco, e sempre contém um número inteiro de registros. O sistema operacional divide sempre um setor nos seus registros. Conforme o padrão adotado no computador, o setor pode variar de 1 registro (128 bytes) até 4 registros (512 bytes), como é o caso do MSX. Note, porém, que para discos Winchester o setor pode ser ainda maior, num compromisso que envolve a capacidade de armazenamento do disco, o espaço mínimo que se deseja que um arquivo sempre ocupe, a fragmentação aceitável de um arquivo, tempo de acesso, etc...

Aglomerado: Exclusivo para discos padrão MSX ou PC, é um grupo de setores. Usado para minimizar o tamanho da FAT, no sentido de que a FAT aponta para um aglomerado, e o mesmo sempre representa os mesmos setores. No disquete de face simples representa 1 setor dado pela fórmula: aglomerado+6. No disquete dupla face representa 2 setores consecutivos cujo primeiro setor é obtido pela fórmula: (2*aglomerado)+6.

Assembladores: Receitas de bolo para usar os principais assembladores existentes no mercado.

Antes de iniciar, vamos discutir as diretivas que os assembladores aceitam, os endereços de início de programa, label e números.

Diretiva **ORG:** Usada para indicar o endereço de montagem do programa. Não significa que o programa será realmente montado neste endereço, mas sim que será montado como se estivesse neste endereço.

Diretiva **EQU:** Usada para atribuir um valor a um label. É a única situação em que o label não possui o caracter ":" (2 pontos) no final.

Diretiva **DB:** Usada para definir um byte ou uma string dentro de um programa. Para se definir uma string deve-se separar cada byte por uma vírgula ou escrever-se como um texto limitado pelo símbolo <"> (aspas). No caso do SIMPLE ASM (da CORAL) esta diretiva é substituída por **DEFB** para byte e **DEFM** para string. Ex: Label: DB "texto".

Diretiva **DS** (**DEFS** para SIMPLE ASM): Usada para definir uma área livre na memória onde não será escrito nenhum trecho de programa.

Diretiva **DW** (**DEFW** para SIMPLE ASM): Usada para se escrever uma word no programa, na maneira que o 8080, o 8085 e o Z80 acessa

(isto é, primeiro o byte menos significativo e depois o mais significativo).

Diretiva **END**: Deve ser usada sempre no fim do programa para indicar ao assemblador o fim do programa. Exceção ao SIMPLE ASM.

Endereços de início de programa: Nos sistemas operacionais CP/M e MSX-DOS o endereço de início de programa sempre é 100H. Porém, o MSX pode também ter programas em linguagem de máquina carregados através do BASIC com o comando **LOAD**. Todas as rotinas deste livro que acessam o sistema operacional não funcionam neste modo de operação, e, portanto, este modo de operação não é objeto deste livro. A título de informação, um programa neste modo deve começar com a seqüência:

```
        DB      0FEH
        DW      INICIO      ;Inicio da área de memória para carga do
                            ;programa
        DW      FIM         ;endereço do final do programa
        DW      START       ;endereço do inicio de operação do pro-
                            ;grama
INICIO: ...
        ...

        ...
FIM:    END
```

Label: Seqüência de 1 a 6 caracteres alfanuméricos seguidos pelo caracter ":". O primeiro caracter deve ser alfabético. Letra minúscula é tratada como se fosse maiúscula. Não pode ter acento.

Números: Números em decimal são os próprios números sem nenhuma complementação; números em hexadecimal sempre devem iniciar por um caracter numérico (use o 0 para hexadecimais que deveriam começar por letra) e terminar com o caracter "H".

Assemblador **ASM/LOAD:** Primeiro assemblador no mercado para CP/M, só aceita mnemônicos 8080 ou 8085. ASM trabalha sobre um arquivo fonte com a extensão **.ASM**, gera um arquivo intermediário com a extensão **.HEX**, e pode gerar um arquivo de listagem com a extensão **.PRN**. A maneira de usá-lo é:

A<ASM NOME.xyz<Cr>

Onde x é o drive que contém o arquivo **.ASM**, y é o que contém o arquivo intermediário, e z o drive que conterá o arquivo listável. O

caracter "Z" pode ser usado nas posições "y" e "z" para indicar que os arquivos correspondentes não devem ser gerados. O caracter "X" pode ser usado na posição "z" para indicar que deve ser feita uma saída no vídeo, e não a geração de um arquivo. O nome é o nome do arquivo fonte sem a extensão. Após gerar o arquivo intermediário, gera-se o arquivo executável com o comando:

A<LOAD NOME<Cr>

Assemblador **M80/L80:** Assemblador mais utilizado em sistemas CP/M, foi criado pela MICROSOFT para gerar programas em linguagem de máquina que pudessem ser ligados a programas gerados por um compilador BASIC. Todavia, a sua aplicação mais usual é a de gerar programas em linguagem de máquina pura e simplesmente. Permite que se subdivida o programa fonte em vários subprogramas que são unidos na fase final, bastando para tal que as variáveis que são compartilhadas pelos diversos subprogramas sejam declaradas **PUBLIC** no subprograma onde é definida e **EXTERNAL** nos subprogramas onde é usada sem ser definida. O(s) programa(s) fonte devem ter a extensão **.MAC**, o arquivo intermediário tem a extensão **.REL**, e o arquivo listável a extensão **.PRN.** Dentro do corpo do programa o default é reconhecer mnemônicos 8080 ou 8085. Para reconhecer mnemônicos Z80 deve-se usar a diretiva ".Z80". Para voltar ao default usa-se a diretiva ".8080". O comando inicial é:

A > M80 = NOME < Cr > ou A > M80 = NOME/L < Cr > (para se gerar o arquivo listável)

Para se gerar o arquivo executável deve-se usar o comando:

A > L80 NOME1, NOME2, NOME3,..., NOMEN, NOMEF/N/E < Cr > onde NOME1 a NOMEN são os nomes dos diversos subprogramas (já arquivos intermediários) e NOMEF é o nome do arquivo executável. Todos os nomes são sem extensão.

Assemblador **SIMPLE ASM:** Criado para rodar especificamente no MSX, é carregado debaixo da ROM de BASIC, e é iniciado por um CALL dentro do BASIC. Tem vários inconvenientes: O tamanho máximo do fonte é 16K, as linhas de programa devem ser numeradas, e é carregado diretamente na memória sendo necessário gerar-se um deslocamento na hora de carregar e depois escrever um pequeno programa em BASIC para salvá-lo no disco.
Para carregar o assemblador, entre no modo BASIC e digite:
BLOAD "SIMPLE",R

Após a carga, o MSX se reseta, mas não destrói o assemblador. Para ativá-lo, entre no modo BASIC e digite: **CALL START**.

Para começar a escrever um programa, chame a numeração automática de linha com o comando: **AUTO.**

O comando de geração com o deslocamento é: **AON/8F00**.

Como o deslocamento montou o programa no endereço 9000H (ORG 100H + deslocamento), então o programa em BASIC para salvá-lo em disco é:

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

Para obter o endereço de **FIM** observe o último endereço na listagem que aparece no vídeo durante a "assemblagem" do programa. Some o número de bytes desta última instrução e o deslocamento 8F00, tudo em hexadecimal.

Para os programas escritos neste livro, elimine os comentários para diminuir o tamanho, e elimine as linhas:

```
ASEG
.Z80

END
```

Reconhece os mnemônicos Z80.

Nota: Todas as rotinas neste livro estão escritas em mnemônicos Z80.

Capítulo 2

Rotinas Agrupadas de Acesso ao Sistema Operacional

Toda vez que se deseja chamar o sistema operacional, é necessário colocar-se o número da função desejada no registrador "C", carregar-se algum outro registrador com informações auxiliares, e então chamar-se o sistema operacional com um "CALL 5". Se ao invés de carregarmos os registradores toda a vez que desejarmos uma função, fizermos isto só uma vez e depois chamarmos sempre este programa que carrega os registradores, no conjunto acabaremos usando menos memória no programa. Assim criei um grupo de rotinas para as principais funções, buscando minimizar ao máximo a ocupação de memória. Note que uso muito o pulo relativo, portanto, mantenha todas as rotinas agrupadas. Pode suprimir as não usadas, mas não separe as rotinas. Descrevo a seguir as rotinas:

INKEY: Lê o teclado sem ecoar, retorna no acumulador a tecla pressionada, ou zero se nenhuma tecla foi pressionada. Funciona em qualquer micro CP/M.

PUT: Ecoa na tela o caracter presente no acumulador. Funciona em qualquer micro CP/M.

BIOS6: Ecoa na tela o caracter presente no registrador "E" se diferente de 255. Se for 255 então lê o teclado. Funciona em qualquer micro CP/M.

BIOS5: Envia para a impressora o caracter presente no registrador "E". Funciona em qualquer micro CP/M.

SETDMA: Envia ao sistema operacional o próximo endereço de registro a ser lido/gravado no disco. Funciona em qualquer micro CP/M.

ABRE: Abre o arquivo indicado pelo FCB para leitura. Funciona em qualquer micro CP/M.

CRIA: Cria ou abre o arquivo indicado pelo FCB para gravação. Se o arquivo já existe, apaga o mesmo primeiro. Funciona em qualquer micro CP/M.

FECHA: Fecha um arquivo aberto. Só é necessário usar após uma gravação. Funciona em qualquer micro CP/M.

APAGA: Apaga o arquivo indicado pelo FCB. Funciona em qualquer micro CP/M.

RENOME: Renomeia o arquivo indicado pelo FCB. Funciona em qualquer micro CP/M.

LE: Lê o próximo registro do arquivo indicado pelo FCB para o endereço indicado pela rotina SETDMA. Funciona em qualquer micro CP/M.

LEAL: Lê o registro apontado pelo par "HL". Lê do arquivo indicado pelo FCB para o endereço indicado pela rotina SETDMA. Funciona em qualquer micro CP/M.

GRAVA: Grava o próximo registro do arquivo indicado pelo FCB para o endereço indicado pela rotina SETDMA. Funciona em qualquer micro CP/M.

```
INKEY:    LD    A,255
PUT:      LD    E,A
BIOS6:    LD    C,6       ;BIOS6 envia o byte em "E" para o vídeo
                          ;se for diferente de 255, pois se for 255
                          ;então lê o teclado, retornando a tecla
                          ;lida no acumulador, ou 0 se nenhuma
                          ;for pressionada
BIOS:     JP    5
BIOS5:    LD    C,5
          JR    BIOS
SETDMA:   LD    C,1AH
          JR    BIOS
ABRE:     LD    C,15
DISCO:    LD    DE,FCB    ;FCB deve ser definido em outra parte do
                          ;programa, normalmente pode-se usar o
                          ;endereço 5CH.
          JR    BIOS
CRIA:     LD    C,22
          JR    DISCO
FECHA:    LD    C,16
          JR    DISCO
APAGA:    LD    C,19
          JR    DISCO
RENOME:   LD    C,23
          JR    DISCO
```

```
LE:     LD      C,20
        JR      DISCO
LEAL:   LD      C,33        ;o número do registro desejado deve ser
                            ;colocado nos bytes 33, 34 e 35 do FCB.
                            ;O byte menos significativo no 33, o
                            ;seguinte no 34, e se necessário o res-
                            ;tante no 35 (normalmente o 35 é zera-
                            ;do)
        JR      DISCO
GRAVA:  LD      C,21
        JR      DISCO
```

Capítulo 3

Rotinas de Teclado

GET: Aguarda pressionar qualquer tecla, retorna no acumulador. Não ecoa no vídeo. Funciona em qualquer micro CP/M.

```
GET:    CALL    INKEY       ;lê o teclado
        OR      A           ;testa se veio tecla
        JR      Z,GET       ;se não veio, repete a operação
        RET
```

SIMNAO: Aguarda somente as teclas "S" ou "N", Maiúscula ou minúscula. Ecoa a tecla pressionada. Retorna com o flag "C" setado se a tecla foi "S". Funciona em qualquer micro CP/M.

```
SIMNAO: CALL    INKEY       ;lê o teclado
        LD      E,A         ;salva tecla para ecoar
        RES     5,A         ;converte para maiúscula
        CP      "S"         ;testa se sim
        SCF                 ;seta o CARRY caso seja sim
        JR      Z,SIMNA     ;se foi sim
        CP      "N"         ;testa se foi não, além disso se der a
                            ;igualdade reseta o CARRY
        JR      NZ,SIMNAO   ;se foi o não então vá ler o teclado nova
                            ;mente. Isto fecha o laço até vir a
                            ;resposta desejada
SIMNA:  PUSH    AF          ;salva o CARRY (que contém a informa-
                            ;ção se foi sim ou não)
        CALL    BIOS6       ;ecoa a tecla pressionada sem conversão
        POP     AF          ;retorna o CARRY
        RET
```

LELIN: Lê uma linha de teclado de tamanho máximo indicado pelo acumulador. Se nada foi digitado, retorna com o flag "Z" setado. Se algo foi digitado, retorna o número de bytes no registrador "B", e o par "HL" aponta para o primeiro byte do texto digitado. Funciona em qualquer micro CP/M.

```
LELIN:  LD    HL,LINHA    ;buffer de entrada do texto a ser digitado
        LD    (IIL).A     ;primeiro byte = número máximo de
                          ;caracteres
        EX    DE,HL       ;o endereço do buffer deve estar no par
                          ;"DE"
        LD    C.10
        CALL  5           ;chama função de leitura de linha
        LD    A,(LINHA+1)
        OR    A           ;se zero, então nada foi digitado
        RET   Z
        LD    B,A         ;"B" = número de caracteres digitados
        LD    HL,LINHA+2  ;início dos caracteres digitados
        RET
LINHA:  DS    257         ;número máximo de caracteres a serem
                          ;lidos é 255, pois é o maior valor possí-
                          ;vel de 1 byte
```

LEHEX: Lê um número hexadecimal, retornando no par "IIL" o hexadecimal teclado. O número rola na tela mantendo na tela sempre os últimos 4 números teclados. Se nada foi entrado, seta o flag "Z". Se algo foi teclado, mesmo que o número 0000, o flag "Z" é resetado. Não altera os pares "BC", "DE", e só altera "HL" se algo for digitado. Funciona em qualquer micro CP/M.

```
LEHEX:  CALL  VIDEO       ;VIDEO não destrói "BC" ou "DE" ou
                          ;"IIL"
        DB    "0000",0    ;envia quatro zeros para a tela
LEHEX0: CALL  LEHEX2      ;lê teclado sem destruir "BC", "DE" e
                          ;"HL" testa também se é o retorno de
                          ;carro < Cr >
        RET   Z           ;se foi retorno de carro
        CALL  LEHEX3      ;verifica se é hexadecimal, se for, retorna
                          ;em "A" o hexadecimal correspondente
        JR    NC,LEHEX0   ;se não for, aguarde próxima tecla
        LD    HL,0        ;zera "HL"
        JR    LEHEX1+3    ;pula a chamada de LEHEX3
LEIIEX1: CALL LEHEX3      ;testa se é hexadecimal
        CALL  C,LEHEX4    ;se for, apresente o valor, rolando o
                          ;número na tela
```

```
            CALL    LEHEX2      ;aguarde a próxima tecla
            JR      NZ,LEHEX1   ;se não for < Cr >
            LD      A,1
            OR      A           ;resete o flag "Z"
            RET
;subrotinas de LEHEX

LEHEX2:     PUSH    BC          ;salva os pares "BC", "DE" e "HL". Note
            PUSH    DE          ;que poderia usar a instrução EXX, mas
            PUSH    HL          ;o MSX muda os registradores alternati-
                                ;vos nas interrupções
            CALL    GET         ;aguarda uma tecla
            POP     HL          ;recupera os pares "HL", "DE", e "BC"
            POP     DE
            POP     BC
            CP      0DH         ;testa se é < Cr >
            RET

LEHEX3:     CALL    MAIUS       ;converte para maiúscula se for letra
                                ;minúscula
            SUB     "0"         ;para ASCII 0 a 9, o acumulador ficará
                                ;com o valor correspondente
            CP      10          ;testa se é número
            RET     C           ;se for número
            SUB     7           ;diferença entre o ASCII de "9" e o de
                                ;"A"
            CP      10          ;se menor que 10 então era um dos
                                ;ASCII intermediários
            CCF                 ;complementa o CARRY para que fique
                                ;resetado se não for número
            RET     NC          ;e, portanto, retorna pois não é hexade-
                                ;cimal
            CP      16          ;teste final para ver se é hexadecimal
            RET

LEHEX4:     PUSH    AF          ;salva o hexadecimal teclado
            CALL    VIDEO       ;retorna para o início do número na tela
            DB      8,8,8,8,0
            POP     AF          ;recupera o hexadecimal
            ADD     HL,HL       ;multiplica por 2
            ADD     HL,HL       ;multiplica por 4
            ADD     HL,HL       ;multiplica por 8
            ADD     HL,HL       ;multiplica por 16
            ADD     A,L         ;e soma ao último hexadecimal teclado
            LD      L,A
            PUSH    BC          ;salva os pares "BC" e "DE"
            PUSH    DE
            PUSH    HL          ;salva os últimos 4 hexadecimais tecla-
                                ;dos
```

```
        CALL    SAIHL       ;envia para tela (já rolou quando multi-
                            ;plicou o par "HL" por 16)
        POP     HL          ;recupera os hexadecimais
        POP     DE          ;e os pares "DE" e "BC"
        POP     BC
        RET
```

LEBCD: Idêntico a LEHEX, mas só aceita numerais (usa LEBCD2)

```
LEBCD:  CALL    VIDEO       ;VIDEO não destrói "BC" ou "DE" ou
                            ;"HL"
        DB      "0000",0    ;envia quatro zeros para a tela
LEBCD0: CALL    LEHEX2      ;lê teclado sem destruir "BC", "DE" e
                            ;"HL".
                            ;testa também se é o retorno de carro
                            ;<Cr>
        RET     Z           ;se foi retorno de carro
        CALL    LEBCD2      ;verifica se é numérico, se for retorna
                            ;em "A" o número correspondente
        JR      NC,LEBCD0   ;se não for, aguarde próxima tecla
        LD      HL,0        ;zera "HL"
        JR      LEBCD1+3    ;pula a chamada de LEBCD2
LEBCD1: CALL    LEBCD2      ;testa se é numérico
        CALL    C,LEHEX1    ;se for, apresente o valor, rolando o
                            ;número na tela
        CALL    LEHEX2      ;aguarde a próxima tecla
        JR      NZ,LEBCD1   ;se não for < Cr >
        LD      A,1
        OR      A           ;resete o flag "Z"
        RET
LEBCD2: SUB     "0"         ;ASCII de 0 a 9 tornar-se-ão seus valores
                            ;no acumulador
        CP      10          ;testa se é número
        RET
```

STOP: Testa a tecla <STOP>, resetando o flag "Z" se estiver pressionada. Exclusivo para MSX.

```
STOP:   DI                  ;paralisa as interrupções, pois a leitura
                            ;de teclado altera a porta 0AAH
        LD      A,0F7H
        OUT     (0AAH),A    ;programa leitura da tecla <STOP>
        IN      A,(0A9H)    ;lê o teclado
        BIT     4,A         ;e testa se a tecla <STOP> foi pressionada
        EI                  ;permite que leitura de teclado ocorra
                            ;normalmente
        RET
```

Capítulo 4

Rotinas de Vídeo e Impressora

VIDEO: Envia para o vídeo o texto iniciado imediatamente após a chamada da rotina. O texto deve terminar com o byte 00. Tem como vantagem não ser necessário apontar o texto, ocupando, portanto, menos espaço de memória. Funciona em qualquer micro CP/M. Não destrói os pares: "BC", "DE" e "HL"

```
VIDEO:  EX    (SP),HL     ;salva "HL" na pilha. "HL" aponta para o
                          ;começo do texto
        CALL  PRINT       ;envia texto para o vídeo
        EX    (SP),HL     ;recupera "HL". Manda para pilha posi-
                          ;ção depois do byte 00
        RET               ;que será o endereço de retorno
```

Exemplo:

```
INICIO: LD    BC,6        ;B = 0, C = 6 (BIOS6)
VOLTA:  PUSH  BC          ;salva o par BC
        LD    E,B         ;envia o byte em "B" para a tela
        CALL  5
        POP   BC          ;recupera o par BC
        DJNZ  VOLTA       ;decrementa "B" e vai para VOLTA se
                          ;não chegou a zero
        CALL  VIDEO       ;envia para o vídeo o texto abaixo
        DB    0AH,"repete (S/N)?",0
;note que o 0AH inicial faz passar para o início da próxima linha
        CALL  SIMNAO      ;aguarda a resposta
        PUSH  AF          ;salva a resposta
        LD    E,0AH
```

```
        CALL    BIOS6       ;muda de linha
        POP     AF          ;retorna a resposta
        JR      C,INICIO    ;se resposta afirmativa, pule para INICIO
```

PRINT/LPRINT: Envia para o vídeo/impresssora o texto apontado pelo par "HL". O texto deve terminar com o byte 00. Funciona em qualquer micro CP/M.

```
LPRINT: PUSH    BC          ;salva o par "BC"
        LD      C,5         ;indica saida na impressora
        JR      PRINT0      ;pula para a parte comum do programa
PRINT:  PUSH    BC          ;salva o par "BC"
        LD      C,6         ;indica saida no video
PRINT0: PUSH    DE          ;salva o par "DE"
PRINT1: LD      A,(HL)      ;busca um caracter
        INC     HL          ;incrementa o ponteiro
        OR      A           ;testa se é o byte 00
        JR      Z,PRINT2    ;se for, então é fim de texto
        LD      E,A         ;não é, portanto vamos enviar o caracter
        PUSH    BC          ;salva indicador de video/impressora
        PUSH    HL          ;salva ponteiro
        CALL    5           ;envia o caracter
        POP     HL          ;retorna o ponteiro
        POP     BC          ;e o indicador de video/impressora
        JR      PRINT1      ;vá buscar o próximo caracter
PRINT2: POP     DE          ;recupera os pares "DE" e "BC"
        POP     BC
        RET
```

SAIHL: Envia o conteúdo do par "HL" para o vídeo. Funciona em qualquer micro CP/M. SAIHL e SAIHEX devem ser escritos em seqüência.

SAIHEX: Envia o conteúdo do acumulador para o vídeo. Funciona em qualquer micro CP/M.

```
SAIHL:  PUSH    HL          ;salva o registrador "L"
        LD      A,H         ;e envia o registrador "H"
        CALL    SAIHEX      ;uma rotina chamando o seu término é
                            ;um macete utilizável em qualquer
                            ;rotina cujo trecho final ocorra várias
                            ;vezes dentro dela mesma, com intuito
                            ;de minimizar a memória ocupada
        POP     HL          ;recupera o registrador "L"
        LD      A,L         ;e envia
SAIHEX: PUSH    AF          ;salva o byte
        RLCA                ;troca de posição os 4 bits superiores
                            ;com os
        RLCA                ;4 inferiores
```

```
        RLCA
        RLCA
        CALL    SAIHE0          ;chamada de trecho final para enviar os
                                ;bits superiores do byte
        POP     AF              ;recupera o byte para enviar os bits infe-
                                ;riores
SAIHE0: AND     0FH             ;separa os 4 bits inferiores
        OR      30H             ;ASCII de "0"
        CP      3AH             ;primeiro ASCII maior que o ASCII de
                                ;"9"
        JP      C,PUT           ;se é número, então pode enviar
        ADD     A,7             ;soma para obter os ASCII das letras
                                ;correspondentes
        JP      PUT
```

;note que se as rotinas agrupadas de acesso ao sistema operacio-
;nal estiverem próximas de SAIHE0, podemos usar "JR" ao invés
;de "JP"

LOCATE: Posiciona o cursor na linha indicada pelo registrador "L" e coluna indicada pelo registrador "H". Funciona no MSX e em qualquer micro CP/M cujo vídeo obedeça ao padrão VT52.

```
LOCATE: LD      DE,2020H        ;offset exigido pelo padrão VT52
        ADD     HL,DE
        LD      (LOCAL),HL      ;armazena para enviar
        CALL    VIDEO           ;envia comando de posicionamento
                                ;padrão VT52
        DB      1BH,"Y"         ;bytes iniciais do comando
LOCAL:  DB      0,0,0           ;posição e byte final 00 para a rotina
                                ;VIDEO
        RET
```

COR: Seleciona as cores de fundo e de caracter no vídeo. As cores a serem utilizadas são indicadas pelo conteúdo do acumulador. Nos 4 bits mais significativos está a cor dos caracteres, e nos 4 menos significativos a cor de fundo. Exclusivo do MSX-DOS ou tela 0 (carregado pelo BASIC).

```
COR:    DI                      ;inibe interrupção
        PUSH    AF              ;salva padrão de cor
        IN      A,(99H)         ;sincroniza o chip controlador de vídeo
        POP     AF              ;recupera o padrão de cor
        OUT     (99H),A         ;e envia
        LD      A,87H           ;ao registro interno número 7
        OUT     (99H),A         ;do controlador de vídeo
```

```
        EI                      ;libera a interrupção
        RET
```

NEGRO: Semelhante a LPRINT, mas coloca em negrito o(s) trecho(s) indicados pelo byte 02. Funciona em qualquer micro CP/M.

```
NEGRO:  PUSH    BC      ;salva os pares "BC" e "DE"
        PUSH    DE
        LD      BC,305H ;"B" = número de passagens, "C" = saída
                        para a impressora
        XOR     A       ;inicia sem negrito
NEGRO0: LD      D,A     ;situação de negrito no início da linha
                        ;está no bit 0 do acumulador
        PUSH    HL      ;salva o início da linha atual
        RES     7,D     ;reseta indicação de negrito
        BIT     0,D     ;testa se primeiro byte tem negrito
        JR      Z,NEGRO1 ;se não
        SET     7,D     ;indica negrito
NEGRO1: LD      A,(HL)  ;busca um caracter
        OR      A       ;testa se é fim do arquivo
        JR      Z,NEGRO2 ;se for
        CP      10      ;testa se é mudança de linha
        JR      Z,NEGRO2 ;se for
        CP      13      ;testa se é retorno de carro sem mudar
                        ;de linha
        JR      Z,NEGRO2 ;se for
        INC     HL      ;incrementa ponteiro de caracteres
        CALL    NEGRO4  ;chama a rotina de enviar o caracter em
                        ;função do número da passagem e da
                        ;indicação de negrito
        JR      NEGRO1  ;continua com os próximos caracteres
NEGRO2: LD      E,13    ;qualquer que seja a situação é necessá-
                        ;rio
        CALL    NEGRO5  ;que seja enviado um retorno de carro
                        ;para imprimir os caracteres enviados
        LD      A,D     ;situação de negrito atual e no início da
                        ;linha vai para o acumulador pois o
                        ;registrador "D" será alterado
        EX      DE,HL   ;"DE" aponta para o byte que forçou a
                        ;mudança para esta fase do programa
        POP     HL      ;e "HL" para o início da linha recém-
                        ;transmitida
        DJNZ    NEGRO0  ;repete o laço até terminar as 3 passa-
                        ;gens
        EX      DE,HL   ;abandona o início da linha, "HL" aponta
                        ;para a continuação do texto
        RRCA            ;o acumulador contém a situação do
                        ;negrito, e
```

```
        AND     1           ;esta é passada para o bit 0(situação do
                            ;início da linha)
        LD      D,A         ;quem deve conter a situação de negrito
                            ; é "D"
        LD      B,3         ;número de passagens
        LD      A,(HL)      ;busca o byte que ocasionou o desvio
        INC     HL          ;incrementa o ponteiro para o primeiro
                            ;byte da próxima linha
        CP      13          ;testa se era retorno de carro
        JR      Z,NEGRO0+1  ;se era, já foi transmitido e pode iniciar
                            ;a próxima linha
        OR      A           ;testa se era fim de texto
        JR      Z,NEGRO3    ;se era
        LD      E,A         ;se não era, então só resta a mudança
                            ;de linha, que deve ser enviada
        CALL    NEGRO5
        JR      NEGRO0+1    ;para depois continuar o processo nor-
                            ;mal da próxima linha
NEGRO3: POP     DE          ;restaura os pares "BC" e "DE"
        POP     BC
        RET
NEGRO4: CP      2           ;testa se é limitador de negrito
        JR      Z,NEGRO6    ;se for
        LD      E,A         ;se não for, prepara para enviar
        LD      A,B         ;busca o número da passagem
        CP      3           ;testa se é a primeira passagem
        JR      Z,NEGRO5    ;se for, então deve imprimir
        BIT     7,D         ;se não for, testa se é negrito
        RET     Z           ;se não for, então não deve imprimir
NEGRO5: PUSH    BC          ;salva os pares "BC", "DE" e "HL"
        PUSH    DE
        PUSH    HL
        CALL    5           ;imprime
        POP     HL          ;e restaura os pares
        POP     DE
        POP     BC
        RET
NEGRO6: LD      A,D         ;busca indicação de negrito
        XOR     80H         ;troca flag de negrito
        LD      D,A         ;e retorna flag para o registrador "D"
        RET
```

Capítulo 5

Rotinas de Acesso a Disco

MKFCB: Prepara o FCB em função do texto apontado pelo par "HL", tamanho do texto vem no registrador "B". Exatamente as condições de saída da rotina LELIN. Portanto, para se preparar um FCB de um arquivo cujo nome será digitado, use LELIN e em seguida MKFCB. Funciona em qualquer micro CP/M.

```
MKFCB:  DI              ;bloqueia interrupção
        EXX             ;salva os pares
        LD      HL,FCB+1
        LD      DE,FCB+2
        LD      BC,10
        LD      (HL),20H
        LDIR            ;bytes correspondendo ao nome no FCB
                        ;recebem espaços brancos (ASCII 20H)
                        ;como default
        EXX             ;restaura os pares
        EI              ;libera interrupção
        INC     HL
        LD      A,(HL)  ;busca o segundo caracter da string
        DEC     HL
        CP      ":"     ;e testa se é ":" (indicativo de que o pri-
                        ;meiro caracter é indicativo de disco, e
                        ;não parte do nome)
        LD      A,0     ;indica disco default
        JR      NZ,MKFCB0 ;para o caso de não ser indicativo de
                        ;disco
```

```
        LD      A,(HL)      ;é indicativo, vejamos de que disco
        INC     HL
        INC     HL
        DEC     B
        DEC     B
        SCF                 ;indicativo de falha
        RET     Z           ;se só tem indicativo de disco
        AND     3
        SCF                 ;indicativo de falha
        RET     Z           ;se indica disco "D"
        CP      3
        SCF                 ;indicativo de falha
        RET     Z           ;se indica disco "C"
MKFCB0: LD      DE,FCB
        LD      (DE),A      ;armazena o disco no FCB
        INC     DE          ;e aponta para o início do nome
        LD      C,B         ;número de caracteres que restam na
                            ;string
        LD      B,8         ;número máximo de caracteres no nome
        CALL    MKFCB1      ;transfere o nome
        CALL    NC,MKFCB2   ;se não achou "." chama a busca do
                            ;mesmo
        JR      NC,CLRFCB   ;se ainda assim não achou,
        LD      DE,FCB+9    ;aponta para o início da extensão
        LD      B,3         ;número máximo de caracteres na exten-
                            ;são
        CALL    MKFCB1      ;transfere a extensão
CLRFCB: LD      HL,FCB+12   ;limpa o restante do FCB
        LD      DE,FCB+13
        LD      BC,23
        LD      (HL),B
        LDIR
        OR      A           ;reseta CARRY para indicar operação
                            ;bem-sucedida
        RET
MKFCB1: LD      A,C
        OR      A           ;testa se ainda tem caracteres
        RET     Z           ;se não
        LD      A,(HL)      ;busca um caracter
        INC     HL          ;incrementa ponteiro do nome
        DEC     C           ;decrementa contador de caracteres do
                            ;nome
        CP      "."         ;testa se é fim do nome
        SCF                 ;flag indicativo
        RET     Z           ;se for fim do nome
        CALL    MAIUS       ;converte para maiúscula
        LD      (DE),A      ;salva no FCB
        INC     DE          ;incrementa ponteiro do FCB
        DJNZ    MKFCB1      ;continua transferência enquanto for
```

```
                                    ;válido
        OR      A                   ;reseta o CARRY para indicar que não
                                    ;achou o caracter "."
        RET
MKFCB2: LD      A,(HL)              ;busca 1 byte
        INC     HL                  ;incrementa ponteiro
        OR      A                   ;reseta o CARRY para indicar que não
                                    ;achou
        DEC     C                   ;decrementa contador de bytes
        RET     Z                   ;se terminou a string
        CP      "."                 ;testa se é o caracter procurado
        SCF                         ;para indicar que achou
        RET     Z                   ;se for
        JR      MKFCB2              ;continua a busca
```

LOAD: Lê o arquivo apontado pelo FCB para a memória apontada pelo par "DE". Se não encontrar o arquivo retorna com o flag "Z" setado. Não testa fim de área de memória, portanto tenha certeza de que o arquivo cabe na memória. No caso do MSX, o último endereço de memória disponivel é 0D5FFH. Funciona em qualquer micro CP/M.

```
LOAD:   PUSH    DE                  ;salva início da posição de memória para
                                    ;a leitura do arquivo
        CALL    ABRE                ;tenta abrir o arquivo
        POP     DE                  ;recupera o início da memória
        INC     A                   ;testa se conseguiu abrir ("A"=0FFH se
                                    ;não)
        RET     Z                   ;se não
LOAD0:  LD      HL,128              ;tamanho de um registro a ser lido
        ADD     HL,DE               ;calcula a próxima posição de memória
        PUSH    HL                  ;e salva
        CALL    SETDMA              ;envia posição de memória atual para o
                                    ;sistema operacional
        CALL    LE                  ;lê um registro
        POP     DE                  ;retorna endereço do próximo registro
        OR      A                   ;testa se leu sem erro o último registro
        JR      Z,LOAD0             ;se leu sem erro, vá ler o próximo
        RET                         ;se errou, então é fim de arquivo
```

UPDATE: Semelhante a rotina SAVE, difere no sentido de transformar arquivo já existente em ".BAK". Apaga o arquivo ".BAK" anterior. Funciona em qualquer micro CP/M.

```
UPDATE: PUSH    DE                  ;salva início da memória
        PUSH    BC                  ;salva o tamanho a ser gravado
```

```
        LD      HL,FCB+9        ;salva a extensão
        LD      DE,EXSV
        LD      BC,3
        LDIR
        EX      DE,HL           ;e muda a mesma para "$$$" (caso haja
                                ;alguma falha para não comprometer
                                ;arquivos OK)
        LD      DE,FCB+9
        LD      C,3
        LDIR
        POP     BC              ;recupera o tamanho da gravação
        POP     DE              ;restaura início da memória
        CALL    SAVE            ;grava o arquivo com a extensão "$$$"
        LD      HL,EXBAK        ;muda a extensão no FCB para "BAK"
        LD      DE,FCB+9
        LD      BC,3
        LDIR
        CALL    CLRFCB          ;esta rotina está dentro de MKFCB e
                                ;limpa o restante do FCB
        CALL    APAGA           ;apaga o arquivo com extensão "BAK"
                                ;anterior
        CALL    CLRFCB          ;limpa o resto do FCB
        LD      HL,FCB          ;prepara para mudar o nome de um
                                ;arquivo para "BAK"
        LD      DE,FCB+16
        LD      BC,16
        LDIR
        LD      HL,EXSV         ;e este arquivo é o que tem a extensão
                                ;que
        LD      DE,FCB+9        ;foi salva
        LD      C,3
        LDIR
        CALL    RENOME          ;muda o nome
        CALL    CLRFCB          ;limpa restante do FCB
        LD      HL,FCB          ;prepara para mudar o nome de um
                                ;arquivo para
        LD      DE,FCB+16       ;a extensão que foi salva
        LD      BC,16
        LDIR
        LD      HL,EXDOL        ;e este arquivo é o recém-gravado com a
                                ;extensão "$$$"
        LD      DE,FCB+9
        LD      C,3
        LDIR
        JP      RENOME          ;muda o nome e termina a rotina
EXSV:   DS      3               ;espaço para salvar a extensão
EXDOL:  DB      "$$$"           ;extensão de rascunho
EXBAK:  DB      "BAK"           ;extensão de back-up
```

SAVE: Salva a memória apontada pelo par "DE", no arquivo indicado pelo FCB. O tamanho é dado pelo registrador "B" em múltiplos de 128 (número de registros a serem gravados). Assim o tamanho máximo de memória a ser gravado é 32K (B=0). Se já existia algum arquivo com o mesmo nome, o mesmo é apagado antes de gravar o novo arquivo. Funciona em qualquer micro CP/M.

```
SAVE:   PUSH   DE          ;salva inicio da memória e tamanho
        PUSH   BC
        CALL   CRIA        ;cria o arquivo, apaga anterior de mesmo
                           ;nome
        POP    BC          ;restaura tamanho
SAVE0:  POP    DE          ;restaura posição de memória do
                           ;próximo registro a ser gravado
        LD     HL,128      ;tamanho de um registro
        ADD    HL,DE       ;calcula endereço do próximo registro
        PUSH   HL          ;e salva
        PUSH   BC          ;salva número de registros que faltam
                           ;gravar
        CALL   SETDMA      ;envia endereço para o sistema operacio-
                           ;nal
        CALL   GRAVA       ;e grava o registro
        POP    BC          ;recupera o número de registros que fal-
                           ;tam gravar
        DJNZ   SAVE0       ;decrementa, e se não terminou vai gra-
                           ;var o próximo registro
        POP    DE          ;acerta a pilha
        JP     FECHA       ;fecha o arquivo (obrigatório numa grava-
                           ;ção)
```

PARA: Para o disk-drive. Deve ser usado após as rotinas de disco anteriores, exceto se ao invés desta chamar-se alguma rotina de teclado (salvo STOP). Se não chamar esta rotina ou alguma das rotinas de teclado, o drive ficará ativo permanentemente. Isto poderá danificar o drive, principalmente a fonte do mesmo que esquentará muito. Exclusivo para MSX.

```
PARA:   LD     IY,(0F347H) ;busca para IY o endereço do slot da
                           ;interface de disco
        LD     IX,4029H    ;endereço da rotina que pára o disco
        JP     1CH         ;pula para a rotina no endereço 4029 da
                           ;interface
```

Capítulo 6

Rotinas de Conversão

MAIUS: Testa o byte presente no acumulador. Se for letra minúscula, converte para maiúscula. Se não, não faz nada. Funciona em qualquer micro CP/M.

```
MAIUS:  CP    61H     ;testa se é letra minúscula
        RET   C       ;se não é
        CP    7BH
        RET   NC      ;se não é letra minúscula
        RES   5,A     ;converte para maiúscula
        RET
```

ATOI: Converte um string numérico com sinal em um número inteiro binário no par "HL". O tamanho do string deve vir no registrador "B", e a rotina é abordada se achar algum caracter não numérico. O string deve ser apontado pelo registrador "IX". Retorna com o registrador "IX", apontando para o primeiro byte após o string convertido. Funciona em qualquer micro CP/M.

```
ATOI:   PUSH  DE      ;salva o par "DE"
        LD    HL,0    ;valor inicial
        LD    A,H
        LD    (SINAL),A ;inicialmente positivo, sinal está definido
                        ;na rotinas matemáticas

ATOI0:  CALL  ENUM    ;testa se o byte apontado por IX é nume-
                      ;ral
        JR    C,ATOI4 ;se for
        INC   IX      ;não é, aponta para o próximo
```

```
        CP      "-"         ;testa se é sinal <->
        JR      Z,ATOI2     ;se for
        SUB     "+"         ;testa se é sinal <+>, zerando acumulador
                            ;se for
        JR      Z,ATOI2     ;se for
        DJNZ    ATOI0       ;decrementa contador de bytes e volta
                            ;para buscar o próximo byte se ainda
                            ;tiver
ATOI1:  POP     DE          ;recupera o par "DE", toda saida de ATOI
                            ;é feita por aqui
        LD      A,(SINAL)   ;busca o sinal
        OR      A           ;testa se positivo
        RET     Z           ;se for, termina a rotina
        PUSH    DE          ;salva o par "DE"
        EX      DE,HL       ;"DE"="HL" é a operação importante aqui
        LD      HL,0
        SBC     HL,DE       ;0-número = número negativo
        POP     DE          ;recupera o par "DE"
        RET

ATOI2:  DJNZ    ATOI3       ;decrementa contador de bytes e pula se
                            ;ainda houver bytes
        JR      ATOI1       ;se não, então finaliza

ATOI3:  LD      (SINAL),A   ;salva o sinal
        CALL    ENUM        ;busca o próximo byte, convertendo para
                            ;o número correspondente
        JR      NC,ATOI1    ;se não é número

ATOI4:  CALL    NOVOV       ;multiplica "HL" por 10 e soma o acu-
                            ;mulador
        JR      NZ,ATOI1    ;se estourou o limite dos inteiros, "HL"
                            ;então contém o número anterior ao
                            ;estouro e termina a rotina
        INC     IX          ;incrementa apontador de bytes
        CALL    ENUM        ;converte o próximo byte para número
        JR      NC,ATOI1    ;se não era número
        DJNZ    ATOI4       ;decrementa o contador de byte, e se o
                            ;byte for válido vai somá-lo ao número
        JR      ATOI1       ;se não, termina a rotina

ENUM:   LD      A,(IX+0)    ;busca o byte
        SUB     30H         ;tenta converter para número, sub-
                            ;traindo o valor ASCII de <0>
        CP      10          ;testa se é realmente número
        RET     C           ;retorna se for
        LD      A,(IX+0)    ;busca o byte que não era número para
                            ;outros testes pela rotina ATOI
        RET
```

```
NOVOV:  PUSH    HL          ;salva o número para o caso de estourar
        ADD     HL,HL       ;multiplica o número por 2
        BIT     7,H         ;testa se estourou
        JR      NZ,NOVOV0   ;se estourou
        LD      D,H         ;salva o valor (número * 2) no par "DE"
        LD      E,L
        ADD     HL,HL       ;multiplica novamente por 2 (total = *4)
        BIT     7,H
        JR      NZ,NOVOV0   ;se estourou
        ADD     HL,HL       ;multiplica novamente por 2 (total = *8)
        BIT     7,H
        JR      NZ,NOVOV0   ;se estourou
        ADD     HL,DE       ;soma ao valor salvo, resultando em
                            ;multiplicação total por 10 (8+2=10)
        BIT     7,H
        JR      NZ,NOVOV0   ;se estourou
        LD      D,0
        LD      E,A
        ADD     HL,DE       ;soma com o número que estava no acu-
                            :mulador
        POP     DE          ;retorna número antigo em "DE"
        BIT     7,H
        RET     Z           ;se não estourou, então termina a rotina
        EX      DE,HL       ;muda número antigo para "HL"
        RET

NOVOV0: POP     HL          ;retorna número antigo para "HL"
        RET
```

ITOA: Converte o número inteiro binário em complemento de 2 para uma string com sinal. O número deve vir no par "HL", e a área de memória aonde será montado o string deve ser apontado pelo registrador "IX". Na volta, o registrador "IX" estará apontando para o primeiro byte após o string, servindo assim de ponteiro para uma possível continuação da string. Funciona em qualquer micro CP/M.

```
ITOA:   PUSH    DE          ;salva "DE"
        LD      DE,10000    :não será alterado por QSINAL ou POSIT
                            ;pois já é positivo
        CALL    QSINAL      ;verifica o sinal de "HL", pois "DE" é
                            ;positivo
        CALL    POSIT       ;converte para número positivo
        LD      A,(SINAL)   ;busca o sinal
        OR      A           ;testa se é positivo
```

```
        JR      Z,ITOA0         ;se for
        LD      A,"-"
        LD      (IX+0),A        ;escreve o sinal <-> no inicio da string
        INC     IX              ;incrementa ponteiro de string
        XOR     A

ITOA0:  LD      (STRING),A      ;zera contador de números já converti-
                                ;dos para ASCII
        CALL    ITOA1           ;converte e escreve os múltiplos de
                                ;10000
        LD      DE,1000
        CALL    ITOA1           ;converte e escreve os múltiplos de 1000
        LD      DE,100
        CALL    ITOA1           ;converte e escreve os múltiplos de 100
        LD      DE,10
        CALL    ITOA1           ;converte e escreve os múltiplos de 10
        LD      A,30H
        OR      L               ;converte as unidades
        LD      (IX+0),A        ;e escreve
        INC     IX              ;aponta para próximo byte após a string
        POP     DE              ;retorna o par "DE"
        RET

;cálculo dos múltiplos de "DE" em "HL", "A" vem zerado e CARRY
resetado

ITOA1:  SBC     HL,DE           ;subtrai
        JR      C,ITOA2         ;se ultrapassou
        INC     A               ;incrementa contador de múltiplo
        JR      ITOA1           ;e volta para continuar o teste

ITOA2:  ADD     HL,DE           ;soma para recuperar o valor de resto
        OR      A               ;testa se havia múltiplo
        JR      Z,ITOA3         ;se não
        OR      30H             ;converte para ASCII
        LD      (IX+0),A        ;e armazena
        INC     IX              ;incrementa ponteiro da string
        LD      A,(STRING)      ;e contador de números já convertidos
        INC     A
        LD      (STRING),A
        XOR     A               ;zera o acumulador e reseta o
                                ;CARRY para a próxima chamada
        RET

ITOA3:  LD      A,(STRING)      ;busca contador de números convertido
        OR      A               ;testa se já houve alguma conversão
        RET     Z               ;retorna se não, pois não interessa
                                ;escrever zeros no início da string
```

```
        INC     A           ;vai converter, portanto, incremente con-
                            ;tador
        LD      (STRING),A
        LD      A,30H       ; ASCII de < 0 >
        LD      (IX+0),A    ;armazena
        INC     IX          ;incrementa ponteiro da string
        XOR     A           ;zera o acumulador e reseta o CARRY
        RET

STRING: DS      1
```

Capítulo 7

Rotinas Matemáticas de Número Inteiro

Nota: As rotinas SOMA e SUBTR são as próprias instruções do Z80, e, portanto, considerei que só usarão estas rotinas aqueles que desejam as operações de multiplicação e divisão. Havia duas otimizações possíveis: Tempo ou Memória. A escolhida foi minimizar a memória ocupada, mesmo porque esta é mais simples de ser entendida e a perda de tempo não chega a ser perceptível. As rotinas estão apresentadas como um grupo, em que algumas interagem com outras com o objetivo de minimizar a memória ocupada.

SOMA: Soma os números inteiros presentes nos pares "HL" e "DE" retornando no par "HL". Obedece os sinais e seta o sinal correto. Funciona em qualquer micro CP/M.

SUBTR: Subtrai o número inteiro presente no par "DE" do número presente no par "HL", retornando no par "HL". Obedece aos sinais e seta o sinal correto. Funciona em qualquer micro CP/M.

MULTI: Multiplica os números inteiros presentes nos pares "HL" e "DE", retornando no par "HL". Obedece aos sinais e seta o sinal correto. Seta o flag "C" se estourar o limite dos inteiros (limite idêntico ao do BASIC). Funciona em qualquer micro CP/M.

DIVI: Divide o número inteiro presente no par "HL" pelo presente no par "DE", retornando no par "HL".

Obedece aos sinais e seta o sinal correto. Funciona em qualquer micro CP/M.

```
SOMA:   ADD   HL,DE     ;a soma é a própria instrução
        RET
DIVI:   CALL  QSINAL    ;pesquisa o sinal resultante da divisão
        CALL  POSIT     ;e converte ambos os números para seus
                        ;valores positivos
        LD    BC,0      ;"BC" é o contador da divisão, e, portan-
                        ;to, é zerado no início
        OR    A         ;reseta o CARRY
DIVI0:  SBC   HL,DE     ;subtrai "DE" de "HL"
        JR    C,DIVI1   ;se terminou
        INC   BC        ;incrementa o contador
        JR    DIVI0     ;e continua
DIVI1:  LD    H,B       ;transfere o valor da divisão para "HL"
        LD    L,C
PSINAL: LD    A,(SINAL) ;testa se o sinal do resultado é positivo
        OR    A
        RET   Z         ;se for
        EX    DE,HL     ;se não é, então converte "HL" para
                        ;negativo
        LD    HL,0      ;fazendo 0-número
SUBTR:  OR    A         ;reseta o CARRY
        SBC   HL,DE     ;e subtrai usando a própria instrução do
                        ;Z80
        RET
MULTI:  CALL  QSINAL    ;pesquisa o sinal resultante da multipli-
                        ;cação
        CALL  POSIT     ;e converte ambos os números para seus
                        ;valores positivos
        LD    B,H       ;transfere um dos fatores para o par
                        ;"BC"
        LD    C,L
        LD    HL,0      ;zera o resultado como valor inicial
MULTI0: LD    A,D       ;testa se o outro fator chegou a zero
        OR    E
        JR    Z,MULTI1  ;se chegou a zero, terminou a multipli-
                        ;cação
        ADD   HL,BC     ;soma o fator em "BC"
        DEC   DE        ;e decrementa o fator em "DE"
        BIT   7,H       ;testa se estourou o limite dos inteiros
        JR    Z,MULTI0  ;se não, então continua a multiplicação
        RES   7,H
        SCF
MULTI1: PUSH  AF        ;salva a condição de estouro
        CALL  PSINAL    ;e acerta o sinal do resultado
        POP   AF        ;retorna condição de estouro
        RET
QSINAL: LD    A,H       ;executa um OU-EXCLUSIVO dos bits
```

```
        XOR     D           ;mais significativos dos 2 fatores
        AND     80H
        LD      (SINAL),A   ;se forem diferentes, então o resultado é
                            ;negativo, e sinal é diferente de zero
        RET
POSIT:  BIT     7,D         ;testa se número em "DE" é negativo
        JR      Z,POSITO    ;se não é
        PUSH    HL          ;salva número em "HL"
        LD      HL,0
        OR      A
        SBC     HL,DE
        EX      DE,HL       ;e faz "DE"=0-"DE"
        POP     HL          ;retorna o número
POSITO: BIT     7,H         ;testa se número em "HL" é negativo
        RET     Z           ;se não é
        PUSH    DE          ;salva número em "DE"
        EX      DE,HL
        LD      HL,0
        OR      A
        SBC     HL,DE       ;e faz "HL"=0-"HL"
        POP     DE          ;retorna o número
        RET
SINAL:  DB      0
```

Capítulo 8

Rotinas Genéricas

COMPAR: Compara uma string apontada pelo par "DE" e terminada pelo byte 00, com uma seqüência de strings apontada pelo par "HL". O fim de cada string individual é o byte 00, e a marca de fim de seqüência de strings é o primeiro byte de uma string ser 00. Retorna com o flag "Z" setado se nenhuma string apontada por "HL" é igual a string apontada por "DE", ou flag "Z" resetado e número relativo da seqüência que igualou a apontada por "DE" no par "BC", em caso contrário. Funciona em qualquer micro CP/M.

```
COMPAR: LD      BC,0        ;posição relativa inicial
COMPA0: LD      A,(HL)      ;busca o primeiro byte de uma string da
                            ;seqüência de strings
        OR      A           ;e testa se é zero
        RET     Z           ;se for então nenhuma string é igual a
                            ;apontada pelo par "DE", e portanto
                            ;retorna com flag Z setado
        PUSH    DE          ;salva início da string de comparação
COMPA1: LD      A,(DE)      ;busca um byte da string de comparação
        CP      (HL)        ;compara com o correspondente da
                            ;seqüência de strings
        JR      NZ,COMPA2   ;se for diferente
        INC     HL          ;incrementa os ponteiros
        INC     DE
        OR      A           ;testa se terminou a string de compara-
                            ;ção. Se terminou, então também termi-
                            ;nou a string na seqüência, pois já se
```

```
                                    ;sabe que é igual ao que está no acumu-
                                    ;lador
        JR      NZ,COMPA1           ;se não terminou
        POP     DE                  ;libera valor da pilha
        INC     A                   ;reseta o flag Z, pois o acumulador
                                    ;estava com o byte 0
        RET
COMPA2: LD      A,(HL)              ;busca um byte da seqüência
        INC     HL                  ;incrementa ponteiro
        OR      A                   ;testa se é fim de string na seqüência
        JR      NZ,COMPA2           ;se não
        POP     DE                  ;retorna início da string de comparação
        INC     BC                  ;incrementa posição relativa
        JR      COMPA0              ;e vai comparar a nova string da seqüên-
                                    ;cia
```

FIND: Procura a ocorrência de uma string apontada por "DE" e terminada pelo byte 00, dentro de uma string maior apontada pelo par "HL" e terminada também pelo byte 00. Em caso de sucesso, retorna com o flag "Z" resetado e o par "HL", apontando para o primeiro byte da string maior após a ocorrência da string procurada. É fácil alterar para o par "HL" apontar para o primeiro byte da string. Funciona em qualquer micro CP/M.

```
FIND:   PUSH    DE                  ;salva início da string de comparação
        PUSH    HL                  ;salva endereço que se supõe ser o iní-
                                    ;cio da igualdade na comparação
FIND0   LD      A,(DE)              ;busca um byte da string de comparação
        OR      A                   ;verifica se a string terminou
        JR      Z,FIND2             ;se terminou
        CP      (HL)                ;testa a igualdade
        JR      NZ,FIND1            ;se diferente
        INC     HL                  ;incrementa os ponteiros
        INC     DE
        JR      FIND0               ;e continua a comparação
FIND1:  POP     HL                  ;retorna o endereço que se supunha ser
                                    ;o início da igualdade
        INC     HL                  ;e aponta para o próximo byte como
                                    ;novo possível início da igualdade
        POP     DE                  ;retorna o início da string de compara-
                                    ;ção
        LD      A,(HL)              ;busca o byte do novo início
        OR      A                   ;e testa se é fim da string maior
        JR      NZ,FIND             ;se não for vai reexecutar a comparação
        RET
FIND2:  POP     DE                  ;mude para POP HL para obter o início
                                    ;da string
        POP     DE                  ;retorna início da string de comparação
```

```
                                    ;e libera a pilha
        INC     A                   ;reseta o flag "Z" indicando que achou a
                                    ;string de comparação na string maior
        RET
```

TRANSF: Copia a string apontada pelo par "HL" e terminada pelo byte 00, para a posição de memória apontada pelo par "DE". Funciona em qualquer micro CP/M.

```
TRANSF: LD      A,(HL)              ;busca um byte
        OR      A                   ;verifica se é fim da string
        RET     Z                   ;se for
        LDI                         ;transfere, incrementa os pares "DE" e
                                    ;"HL" decrementa o par "BC", o que
                                    equivale a uma destruição do conteúdo
                                    ;deste par, pois não se sabe quantos
                                    ;bytes foram transferidos
        JR      TRANSF              ;continua a transferência
```

TEMPO: Espera o número de milissegundos indicados pelo par "DE". Funciona em qualquer micro CP/M.

```
TEMPO:  CALL    MILIS               ;espera 1 milissegundo
        DEC     DE                  ;decrementa contador de milissegundos
        LD      A,D
        OR      E                   ;e testa se terminou
        JR      NZ,TEMPO            ;se não
        RET
```

MILIS: Espera 1 milissegundo. O valor de carga do registrador "B" deve ser variado conforme o clock do computador, utilizando as fórmulas: NC=Freqüência em Hz/1000, LP=NC-29, e B=LP/13. Para o exemplo abaixo iremos considerar o clock da CPU de 3.57MHz, o que dará NC=3570, LP=3541, e B por arredondamento será 272. Como o maior número de loops possível é 256 para B-0, faremos 2 loops: o primeiro de valor 272-256=16, e o segundo será automaticamente de 256, pois um loop termina com B=0. Funciona em qualquer micro CP/M.

```
MILIS:  LD      B,16
        DJNZ    $
        DJNZ    $
        RET
```

Nota: É conveniente desabilitar a interrupção durante essas rotinas

pois o tratamento da interrupção também consome tempo, e altera o tempo total. Não esqueça de reliberar (no MSX a leitura do teclado depende da interrupção).

INICIA: Envia várias seqüências de bytes para várias portas, para inicialização do hardware do computador. Não deve ser utilizada como subrotina, pois destrói o registro de pilha. Deve ser usada no início de um programa antes de se inicializar o registrador de pilha "SP". Funciona em qualquer micro baseado no Z80, independente de ser ou não CP/M.

```
;não tem label pois não deve ser utilizada como subrotina
        LD      D,N             ;N=número de portas a serem programa-
                                ;das
        LD      SP,TABELA       ;tabela de portas e número de bytes
        LD      HL,BYTES        ;tabela de bytes a serem enviados
INICIO: POP     BC              ;busca endereço e número de bytes
        OTIR                    ;envia
        DEC     D
        JR      NZ,INICIO       ;se ainda tem portas
        LD      SP,PILHA        ;o endereço da pilha real do programa
        ...                     ;corpo do programa
        ...


        ...
        ...
TABELA: DB      P1,N1           ;endereço da porta1 e seu número de
                                ; bytes
        DB      P2,N2
        ...


        ...
        DB      PN,NN
BYTES:  DB      X1,X2,X3,...    ;N1 bytes
        DB      Y1,Y2,...
        ...


        ...
        DB      Z1,Z2,...       ;NN bytes
```

Capítulo 9

Informações Úteis sobre o MSX

Byte retornado pela rotina INKEY para as teclas de seta, <HOME>,<CLS>,<SELECT>,<INSERT>,<DELETE>, <BS>,<ESC>,<TAB>,<CONTROL-LETRA>.

Seta para direita	:	1CH
Seta para esquerda	:	1DH
Seta para cima	:	1EH
Seta para baixo	:	1FH
<HOME>	:	0BH
<CLS>	:	0CH
<SELECT>	:	18H
<INSERT>	:	12H
<DELETE>	:	7FH
<BS>	:	08H
<ESC>	:	1BH
<TAB>	:	09H
<CONTROL-LETRA>	:	Pegue o código ASCII da letra maiúscula e subtraia o valor 40H.

Endereço de memória da variável time (do BASIC).

Esta é uma variável de 2 bytes. O menos significativo está no endereço 0FC9EH e o mais significativo está em 0FC9FH. O valor desta variável é incrementado a cada 1/60 do segundo, desde que não se desabilite a interrupção ou se chame as rotinas de disco ou fita (que também bloqueiam as interrupções).
Nota: As rotinas de fita só são acessáveis pelo BASIC, ou em rotina assembler que tenha sido chamada pelo BASIC.

Endereço das strings resultantes do pressionamento das teclas F1 a F10. Como alterar estas strings.

O endereço para F1 é 0F87FH, F2 é 0F88FH, e assim por diante até F10, cujo endereço é 0F90FH. A string deve começar nestes endereços e seu tamanho máximo é de 16 bytes. Devem terminar com o byte 00 caso possuam menos de 16 bytes. Assim, para mudar o conteúdo das teclas, basta escrever por cima da string já existente. Suponha que se deseje que a tecla F1 seja a chamada do diretório do disco. Assim se mudaria o conteúdo das memórias 0F87FH até 0F883H para os seguintes valores

0F87FH - 44H	letra D
0F880H - 49H	letra I
0F881H - 52H	letra R
0F882H - 0DH	retorno de carro
0F883H - 00H	fim da string

Endereço de entrada do restante da linha digitada na chamada de um programa.

O restante de uma linha digitada para a chamada de um programa (bytes após o nome do programa e o primeiro espaço) é 80H, sendo que em 80H vem o número de bytes (0 se nada foi digitado após a chamada) e os bytes subseqüentes são o restante da linha digitada. Isto é válido também para qualquer micro CP/M.

Endereços de rotinas quando em modo chamado pelo BASIC

00A8H	Se a impressora está pronta para receber um caracter esta rotina retorna 255 no acumulador e reseta o flag Z.
00D5H	Testa teclas de direção ou joystick: se o acumulador contém 0, então testa as teclas, se 1 o joystick A, se 2 o joystick 2. Retorna 0 se nenhuma direção é acionada, 1 para subir, 2 para subir à direita, 3 para à direita, 4 para descer à direita, 5 para descer, 6 para descer à esquerda, 7 para a esquerda, e 8 para subir à esquerda.
00D8H	Testa o triger indicado pelo acumulador. Se acionado, retorna com 255 no acumulador. A indicação é 0 para a tecla de espaço, 1 para o botão 1 do joystick A, 2 para o botão 2 do joystick A, 3 para o botão 1 do joystick B, e 4 para o botão 2 do joystick B.
00E1H	Liga a fita para entrada de dados.
00E4H	Lê um byte da fita para o acumulador.
00E7H	Desliga a fita para leitura.

00EAH	Liga a fita para gravação. Se o acumulador estiver zerado, usa header curto, se não usa header longo.
00EDH	Grava o byte do acumulador na fita.
00F0H	Pára a fita para gravação.
005FH	Muda o modo do vídeo para o indicado no acumulador (0, 1, 2, ou 3).
0062H	Muda a cor do vídeo de acordo com as memórias: 0F3E9H - primeiro plano, 0F3EAH - fundo, 0F3EBH - borda.

Capítulo 10

Gráficos no MSX-DOS

Introdução

Apesar do MSX possuir um processador de vídeo extremamente poderoso, no sistema operacional MSX-DOS se trabalha com a tela 0, que não é gráfica. Todavia, pode-se definir os caracteres que o vídeo usará. Isto é válido também para o modo carregado pelo BASIC, trabalhando com a tela 0. Neste capítulo é mostrado como isto pode ser feito, e como utilizar estes caracteres.

Escrevendo na RAM de vídeo

Para se escrever na RAM de vídeo é suficiente indicar a posição do primeiro byte na porta 99H, e depois enviar o trem de bytes a serem gravados na RAM para a porta 98H. Para se enviar o endereço para a porta 99H envia-se primeiro o byte baixo, e depois o byte alto com o bit 6 setado para indicar que é um endereço. Isto pode ser bem entendido no terceiro exemplo do próximo capítulo.

Lendo a RAM de vídeo

Para se ler a RAM de vídeo é necessário que se envie cada endereço que se deseja ler para a porta 99H, e depois se leia a porta 98H. Note porém que a leitura da porta 98H lê na verdade o byte presente no buffer de entrada/saída, e faz com que o byte realmente desejado vá para este buffer. Assim só a segunda leitura desta porta realmente trará o byte desejado. Se se deseja ler uma seqüência de endereços, deve-se ler somente uma vez cada endereço, e mais uma leitura para o último endereço. A seqüência assim lida estará defasada de uma posição.

A formação dos caracteres gráficos

Os caracteres ocupam 8 bytes consecutivos para armazenar sua forma. A área de memória de vídeo onde estão armazenados estes bytes começa no endereço 800H. Assim o caracter 41H (letra "A") tem sua forma armazenada no endereço 0A08H (8*41H=208H). Nesta posição estão armazenados 8 bytes: 20H, 50H, 88H, 88H, 0F8H, 88H, 88H e 00H

O caracter "A"
endereço 0A08H - conteúdo : 00100000
endereço 0A09H - conteúdo : 01010000
endereço 0A0AH - conteúdo : 10001000
endereço 0A0BH - conteúdo : 10001000
endereço 0A0CH - conteúdo : 11111000
endereço 0A0DH - conteúdo : 10001000
endereço 0A0EH - conteúdo : 10001000
endereço 0A0FH - conteúdo : 00000000

Utilizando os caracteres gráficos

Podemos usar as funções do DOS para enviar para a tela os caracteres que desejamos usar, mas por este processo os caracteres de 00H a 1FH ficam proibidos. A maneira de usarmos estes caracteres é enviá-los diretamente para a memória de vídeo. O endereço neste caso é 0000H para o canto superior esquerdo da tela, 0001H para o logo a direita, e assim por diante. O canto esquerdo da segunda linha fica no endereço 0028H, o do canto esquerdo da terceira linha em 0050H, e assim por diante. No manual do MSX existe uma tabela que indica quais os desenhos de cada caracter, e podemos ver que há caracteres gráficos no meio. Podemos usá-los, ou definir novos caracteres conforme explicado anteriormente.

Capítulo 11

Exemplos

CALCULO Este programa executa as funções de uma calculadora de números inteiros. Executa somente as quatro operações básicas. Digite a operação desejada e termine com retorno de carro. Não é necessário colocar o sinal < = > no fim da operação. Funciona em qualquer micro CP/M.

```
.Z80
ASEG                    ;necessário somente se usar M80 e L80
                        ;como assemblador
ORG     100H
LD      SP,100H         ;reserva espaço entre 80H e 100H para
                        ;pilha note que nem o MSX nem o CP/M
                        ;usam normalmente esta área, a não ser
                        ;como buffer do restante da linha de
                        ;chamada do programa, o que não é o
                        ;caso
CALL    VIDEO           ;mensagem inicial
DB      13,10,"Digite a conta desejada",13,10,0
LD      A,80            ;80 bytes máximos
CALLL   LELIN           ;lê uma linha do teclado
PUSH    HL              ;transfere o ponteiro da linha para o
                        ;registrador
POP     IX
CALL    ATOI            ;busca o primeiro número
LD      (NUMA),HL       ;salva o número
CALL    FINDOP          ;busca a operação
JR      Z,EXOP          ;se achou uma operação
JR      NC,FINAL        ;se a string terminou
LD      A,"+"           ;operação default
```

```
EXOP:   PUSH    AF          ;salva a operação
        CALL    ATOI        ;busca o segundo número
        EX      DE,HL       ;que vai para o par "DE"
        LD      HL,(NUMA)   ;retorna o primeiro número para o par
                            ;"HL"
        POP     AF          ;retorna a operação
        CP      "+"         ;testa qual é a operação desejada
        JR      Z,EXOP0     ;e vai executá-la
        CP      "-"
        JR      Z,EXOP1
        CP      "*"
        JR      Z,EXOP2
        CALL    DIVI        ;só resta a divisão como operação possi
                            ;vel
        JR      EXOP3       ;vai apresentar o resultado
EXOP0:  CALL    SOMA        ;soma
        JR      EXOP3       ;vai apresentar o resultado
EXOP1:  CALL    SUBTR       ;subtração
        JR      EXOP3       ;vai apresentar o resultado
EXOP2:  CALL    MULTI       ;multiplicação
EXOP3:  CALL    VIDEO       ;apresenta o início da mensagem sobre o
                            ;resultado. Note que os pares "BC",
                            "DE", "HL" não são destruídos
        DB      13,10,"O resultado é ",0
        LD      IX,LINHA    ;esta área de memória não é mais neces-
                            ;sária
        CALL    ITOA        ;e, portanto, convertemos o resultado na
                            ;mesma
        LD      (IX+0),0    ;marca de fim de string
        LD      HL,LINHA    ;início da string com o resultado
        CALL    PRINT       ;envia a string para a tela
FINAL:  CALL    VIDEO       ;pergunta se repete o programa
        DB      13,10,"Outra conta (S/N)?",0
        CALL    SIMNAO
        JP      C,100H      ;se repete o programa
        JP      0           ;fim de programa, volta ao sistema ope-
                            ;racional
NUMA:   DS      2

FINDOP: CALL    ENUM        ;busca um byte
        RET     C           ;se for numeral
        INC     IX          ;incrementa ponteiro
        CP      "+"         ;testa se é uma das operações matemáti-
                            ;cas
        RET     Z           ;retornando se for
        CP      "-"
        RET     Z
        CP      "*"
        RET     Z
```

```
        CP      "/"
        RET     Z
        DJNZ    FINDOP      ;decrementa contador de bytes ("B") e
                            ;continua a pesquisa se ainda tem bytes
                            ;válidos

        LD      A,1
        OR      A           ;reseta o flag "Z" para indicar que termi-
                            ;nou a string sem achar número ou a
                            ;operação

        RET

ATOI:   PUSH    DE          ;salva o par "DE"
        LD      HL,0        ;valor inicial
        LD      A,H
        LD      (SINAL),A   ;inicialmente positivo, sinal está definido
                            ;nas rotinas matemáticas
ATOI0:  CALL    ENUM        ;testa se o byte apontado por IX é nume-
                            ;ral
        JR      C,ATOI4     ;se for
        INC     IX          ;não é, aponta para o próximo
        CP      "-"         ;testa se é sinal < - >
        JR      Z,ATOI2     ;se for
        SUB     "+"         ;testa se é sinal < + >, zerando acumulador
                            ;se for
        JR      Z,ATOI2     ;se for
        DJNZ    ATOI0       ;decrementa contador de bytes e volta
                            ;para buscar o próximo byte se ainda
                            ;tiver
ATOI1:  POP     DE          ;recupera o par "DE", toda saída de ATOI
                            ;é feita por aqui
        LD      A,(SINAL)   ;busca o sinal
        OR      A           ;testa se positivo
        RET     Z           ;se for, termina a rotina
        PUSH    DE          ;salva o par "DE"
        EX      DE,HL       ;"DE"="HL" é a operação importante aqui
        LD      HL,0
        SBC     HL,DE       ;0-número = número negativo
        POP     DE          ;recupera o par "DE"
        RET
ATOI2:  DJNZ    ATOI3       ;decrementa contador de bytes e pula se
                            ;ainda houver bytes
        JR      ATOI1       ;se não, então finaliza
ATOI3:  LD      (SINAL),A   ;salva o sinal
        CALL    ENUM        ;busca o próximo byte convertendo
                            ;para o número correspondente
        JR      NC,ATOI1    ;se não é número
ATOI4:  CALL    NOVOV       ;multiplica "HL" por 10 e soma o acu-
                            ;mulador
        JR      NZ,ATOI1    ;se estourou o limite dos inteiros, "HL"
```

```
                                    ;então contém o número anterior ao
                                    ;estouro e termina a rotina
        INC     IX                  ;incrementa apontador de bytes
        CALL    ENUM                ;converte o próximo byte para número
        JR      NC,ATOI1            ;se não era número
        DJNZ    ATOI4               ;decrementa o contador de byte, e se o
                                    ;byte for válido vai somá-lo ao número
        JR      ATOI1               ;se não, termina a rotina
ENUM    LD      A,(IX+0)            ;busca o byte
        SUB     30H                 ;tenta converter para número subtraindo
                                    ;o valor ASCII de < 0 >
        CP      10                  ;testa se é realmente número
        RET     C                   ;retorna se for
        LD      A,(IX+0)            ;busca o byte que não era número para
                                    ;outros testes pela rotina ATOI
        RET
NOVOV:  PUSH    HL                  ;salva o número para o caso de estourar
        ADD     HL,HL               ;multiplica o número por 2
        BIT     7,H                 ;testa se estourou
        JR      NZ,NOVOV0           ;se estourou
        LD      D,H                 ;salva o valor (número * 2) no par "DE"
        LD      E,L
        ADD     HL,HL               ;multiplica novamente por 2 (total = *4)
        BIT     7,H
        JR      NZ,NOVOV0           ;se estourou
        ADD     HL,HL               ;multiplica novamente por 2 (total = *8)
        BIT     7,H
        JR      NZ,NOVOV0           ;se estourou
        ADD     HL,DE               ;soma ao valor salvo, resultando em
                                    ;multiplicação total por 10 (8+2=10)
        BIT     7,H
        JR      NZ,NOVOV0           ;se estourou
        LD      D,0
        LD      E,A
        ADD     HL,DE               ;soma com o número que estava no acu-
                                    ;mulador
        POP     DE                  ;retorna número antigo em "DE"
        BIT     7,H
        RET     Z                   ;se não estourou então termina a rotina
        EX      DE,HL               ;muda número antigo para "HL"
        RET
NOVOV0: POP     HL                  ;retorna número antigo para "HL"
        RET
;note que retirei a salva do par "DE", pois é desnecessário no pro-
;grama
ITOA:   LD      DE,10000            ;QSINAL e POSIT não alteram o par
                                    ;"DE" pois já é positivo
        CALL    QSINAL              ;verifica o sinal de "HL", pois "DE" é
```

```
                                        ;positivo
        CALL    POSIT       ;converte para número positivo
        LD      A,(SINAL)   ;busca o sinal
        OR      A           ;testa se é positivo
        JR      Z,,ITOA0    ;se for
        LD      A,"-"
        LD      (IX+0),A    ;escreve o sinal < -  > no início da string
        INC     IX          ;incrementa ponteiro de string
        XOR     A
ITOA0:  LD      (STRING),A  ;zera contador de números já converti-
                            ;dos para ASCII
        CALL    ITOA1       ;converte e escreve os múltiplos de
                            ;10000
        LD      DE,1000
        CALL    ITOA1       ;converte e escreve os múltiplos de 1000
        LD      DE,100
        CALL    ITOA1       ;converte e escreve os múltiplos de 100
        LD      DE,10
        CALL    ITOA1       ;converte e escreve os múltiplos de 10
        LD      A,30H
        OR      L           ;converte as unidades
        LD      (IX+0),A    ;e escreve
        INC     IX          ;aponta para o próximo byte após a
                            ;string
        RET

;cálculo dos múltiplos de "DE" em "HL", "A" vem zerado e CARRY
resetado

ITOA1:  SBC     HL,DE       ;subtrai
        JR      C,ITOA2     ;se ultrapassou
        INC     A           ;incrementa contador de múltiplo
        JR      ITOA1       ;e volta para continuar o teste

ITOA2:  ADD     HL,DE       ;soma para recuperar o valor de resto
        OR      A           ;testa se havia múltiplo
        JR      Z,ITOA3     ;se não
        OR      30H         ;converte para ASCII
        LD      (IX+0),A    ;e armazena
        INC     IX          ;incrementa ponteiro da string
        LD      A,(STRING)  ;e contador de números já convertidos
        INC     A
        LD      (STRING),A
        XOR     A           ;zera o acumulador e reseta o CARRY
                            ;para a próxima chamada
        RET

ITOA3:  LD      A,(STRING)  ;busca contador de números convertido
```

```
        OR      A           ;testa se já houve alguma conversão
        RET     Z           ;retorna se não, pois não interessa
                            ;escrever zeros no início da string
        INC     A           ;vai converter, portanto, incremente con-
                            ;tador
        LD      (STRING),A
        LD      A,30H       ;ASCII de < 0 >
        LD      (IX+0),A    ;armazena
        INC     IX          ;incrementa ponteiro da string
        XOR     A           ;zera o acumulador e reseta o CARRY
        RET

STRING: DS      1           ;usado para indicar quantos caracteres
                            ;ASCII já possui a string onde está
                            ;sendo escrito o número

SOMA:   ADD     HL,DE       ;a soma é a própria instrução
        RET

DIVI:   CALL    QSINAL      ;pesquisa o sinal resultante da multipli-
                            ;cação
        CALL    POSIT       ;e converte ambos os números para seus
                            ;valores positivos
        LD      BC,0        ;"BC" é o contador da divisão, e, portan-
                            ;to, é zerado no início
        OR      A           ;reseta o CARRY

DIVI0:  SBC     HL,DE       ;subtrai "DE" de "HL"
        JR      C,DIVI1     ;se terminou
        INC     BC          ;incrementa o contador
        JR      DIVI0       ;e continua

DIVI1:  LD      H,B         ;transfere o valor da divisão para "HL"
        LD      L,C

PSINAL: LD      A,(SINAL)   ;testa se o sinal do resultado é positivo
        OR      A
        RET     Z           ;se for
        EX      DE,HL       ;se não é, então converte "HL" para
                            ;negativo
        LD      HL,0        ;fazendo 0-número

SUBTR:  OR      A           ;reseta o CARRY
        SBC     HL,DE       ;e subtrai usando a própria instrução do
                            ;Z80
        RET

MULTI:  CALL    QSINAL      ;pesquisa o sinal resultante da divisão
        CALL    POSIT       ;e converte ambos os números para seus
```

```
                                    ;valores positivos
        LD      B,H                 ;transfere um dos fatores para o par
                                    ;"BC"
        LD      C,L
        LD      HL,0                ;zera o resultado como valor inicial

MULTI0: LD      A,D                 ;testa se o outro fator chegou a zero
        OR      E
        JR      Z,MULTI             ;se chegou a zero, terminou a multipli
                                    ;caçao
        ADD     HL,BC               ;soma o fator em "BC"
        DEC     DE                  ;e decrementa o fator em "DE"
        BIT     7,H                 ;testa se estourou o limite dos inteiros
        JR      Z,MULTI0            ;se não, então continua a multiplicação
        RES     7,H
        SCF

MULTI1: PUSH    AF                  ;salva a condição de estouro
        CALL    PSINAL              ;e acerta o sinal do resultado
        POP     AF                  ;retorna condição de estouro
        RET

QSINAL: LD      A,H                 ;executa um OU-EXCLUSIVO dos bits
        XOR     D                   ;mais significativos dos dois fatores
        AND     80H
        LD      (SINAL),A           ;se forem diferentes então o resultado é
                                    ;negativo, e sinal é diferente de zero
        RET

POSIT:  BIT     7,D                 ;testa se número em "DE" é negativo
        JR      Z,POSIT0            ;se não é
        PUSH    HL                  ;salva número em "HL"
        LD      HL,0
        OR      A
        SBC     HL,DE
        EX      DE,HL               ;e faz "DE"=0-"DE"
        POP     HL                  ;retorna o número

POSIT0: BIT     7,H                 ;testa se número em "HL" é negativo
        RET     Z                   ;se não é
        PUSH    DE                  ;salva número em "DE"
        EX      DE,HL
        LD      HL,0
        OR      A
        SBC     HL,DE               ;e faz "HL"=0-"HL"
        POP     DE                  ;retorna o número
        RET
```

```
SINAL:   DB     0          ;se 0 então o número é positivo

INKEY:   LD     E,255

BIOS6:   LD     C,6

BIOS:    JP     5

GET:     CALL   INKEY      ;lê o teclado
         OR     A          ;testa se veio tecla
         JR     Z,GET      ;se não veio repete a operação
         RET

SIMNAO:  CALL   INKEY      ;lê o teclado
         LD     E,A        ;salva tecla para ecoar
         RES    5,A        ;converte para maiúscula
         CP     "S"        ;testa se sim
         SCF               ;seta o CARRY para caso seja sim
         JR     Z,SIMNA    ;se foi sim
         CP     "N"        ;testa se foi não, além disso se der a
                           ;igualdade reseta o CARRY
         JR     NZ,SIMNAO  ;se não foi o não, então vá ler o teclado
                           ;novamente. Isto fecha o laço até vir a
                           ;resposta desejada

SIMNA:   PUSH   AF         ;salva o CARRY (que contém a informa-
                           ;ção se foi sim ou não)
         CALL   BIOS6      ;ecoa a tecla pressionada sem conversão
         POP    AF         ;retorna o CARRY
         RET

LELIN:   LD     HL,LINHA   ;buffer de entrada do texto a ser digitado
         LD     (HL),A     ;primeiro byte = número máximo de
                           ;caracteres
         EX     DE,HL      ;o endereço do buffer deve estar no par
                           ;"DE"
         LD     C,10
         CALL   5          ;chama função de leitura de linha
         LDD    A,(LINHA+1)
         OR     A          ;se zero, então nada foi digitado
         RET    Z
         LD     B,A        ;"B"=número de caracteres digitados
         LD     HL,LINHA+2 ;início dos caracteres digitados
         RET

LINHA:   DS     257        ;número máximo de caracteres a serem
                           ;lidos é 255, pois é o maior valor possí-
                           ;vel de 1 byte
```

```
VIDEO:  EX    (SP),HL    ;salva "HL" na pilha, "HL" aponta para o
                         ;começo do texto
        CALL  PRINT      ;envia texto para o vídeo
        EX    (SP),HL    ;recupera "HL", manda para pilha posi-
                         ;ção depois do byte 00
        RET              ;que será o endereço de retorno

PRINT:  PUSH  BC         ; salva o par "BC"
        LD    C,6        ;indica saída no vídeo

PRINT0: PUSH  DE         ;salva o par "DE"

PRINT1: LD    A,(HL)     ;busca um caracter
        INC   HL         ;incrementa o ponteiro
        OR    A          ;testa se é o byte 00
        JR    Z,PRINT2   ;se for, então é fim de texto
        LD    E,A        ;não é, portanto vamos enviar o caracter
        PUSH  BC         ;salva indicador de vídeo/impressora
        PUSH  HL         ;salva ponteiro
        CALL  5          ;envia o caracter
        POP   HL         ;retorna o ponteiro
        POP   BC         ;e o indicador de vídeo/impressora
        JR    PRINT1     ;vá buscar o próximo caracter

PRINT2: POP   DE         ;recupera os pares "DE" e "BC"
        POP   BC
        RET
        END
```

IMPRIME

Este programa envia um arquivo para a impressora. Este arquivo deve ser menor que a área máxima disponível de memória (52K). Este programa também contorna o problema provocado pelo fato da impressora GRAFIX MTA não obedecer ao caracter de tabulação. Se o nome do arquivo vier junto com a chamada do programa, este arquivo é impresso e o programa termina. Em caso contrário, o programa pergunta qual o arquivo desejado, imprime, e pergunta se deseja imprimir outro arquivo.

```
FCB     EQU   5CH        ;definição inicial de FCB
        .Z80
        ASEG             ;necessário somente se usar M80 e L80
                         ;como assemblador
        ORG   100H
        LD    SP,100H    ;reserva espaço entre 80H e 100H para
```

```
                                ;pilha note que nem o MSX nem o CP/M
                                ;usam normalmente esta área, a não ser
                                ;como buffer do restante da linha de
                                ;chamada do programa, o que é o caso,
                                ;mas logo é transferido.
        LD      A,(80H)         ;inicio da continuaçao da chamada de
                                programa
        OR      A
        JR      Z,INICIO        ;se não veio mais nada
        LD      (FONTE),A       ;indica que veio o nome na chamada do
                                ;programa
        LD      HL,81H          ;inicio do texto
        LD      B,A             ;tamanho do texto
        LD      DE,LINHA+2      ;área de armazenamento do texto

PESQ:   LD      A,(HL)          ;pesquisa para achar o inicio do texto
        CP      20H             ;testa se é espaço
        JR      Z,PESQ0         ;se for
        CP      9               ;testa se é tabulaçao
        JR      NZ,FPESQ        ;se não for

PESQ0:  INC     HL              ;incrementa ponteiro do texto e
        DJNZ    PESQ            ;decrementa contador, continuando se = 0
        XOR     A
        LD      (FONTE),A       ;se terminou o texto, indica que não veio
                                ;o nome e vai buscar um nome
        JR      INICIO

FPESQ:  LD      C,B             ;"BC"=tamanho ainda válido do texto
        LD      B,0
        LD      A,C             ;salva o tamanho
        LDIR                    ;transfere o arquivo
        LD      HL,LINHA+2      ;inicio do texto válido
        LD      B,A             ;e seu tamanho
        JR      LENDO           ;vai ler o arquivo

INICIO: CALL    VIDEO           ;pergunta qual o arquivo
        DB      13,10,"Qual o nome do arquivo?", 13,10,0
        LD      A,80            ;reserva área de 80 bytes para ler
        CALL    LELIN           ;lê o teclado
        JR      Z,FINAL         ;se nada foi digitado
LENDO:  CALL    MKFCB           ;monta o FCB
        CALL    VIDEO           ;desce uma linha no vídeo
        DB      13,10,0
        LD      DE,BUFFER       ;área para leitura do arquivo
        CALL    LOAD            ;lê o arquivo
        JR      Z,FINAL         ;se arquivo não existe
        LD      HL,BUFFER       ;inicio do arquivo
        CALL    IMPRI           ;vai imprimir
```

```
FINAL:  LD      A,(FONTE)   ;testa se o nome veio na chamada do
                            ;programa
        OR      A
        JP      NZ,0        ;se veio termina o programa
        CALL    VIDEO       ;pergunta se repete o programa
        DB      13,10,"Outro arquivo(S/N)? ",0
        CALL    SIMNAO      ;aguarda a resposta
        JR      C, INICIO   ;se repete o programa
        JP      0           ;fim de programa, volta ao sistema ope-
                            ;racional

MAIUS:  CP      61H         ;testa se é letra minúscula
        RET     C           ;se não é
        CP      7BH
        RET     NC          ;se não é letra minúscula
        RES     5,A         ;converte para maiúscula
        RET

GET:    CALL    INKEY       ;lê o teclado
        OR      A           ;testa se veio tecla
        JR      Z,GET       ;se não veio repete a operação
        RET

INKEY:  LD      A,255

PUT:    LD      E,A

BIOS6:  LD      C,6         ;BIOS6 envia o byte em "E" para o video
                            ;se for diferente de 255, pois se for 255
                            ;então lê o teclado, retornando a tecla
                            ;lida no acumulador ou 0 se nenhuma
                            ;tecla for pressionada

BIOS:   JP      5

SETDMA: LD      C,1AH
        JR      BIOS

ABRE:   LD      C,15

DISCO:  LD      DE,FCB      ;FCB deve ser definido em outra parte do
                            ;programa, normalmente pode-se usar o
                            ;endereço 5CH
        JR      BIOS

LE:     LD      C,20
        JR      DISCO
```

```
SIMNAO:  CALL   INKEY       ;lê o teclado
         LD     E,A         ;salva tecla para ecoar
         RES    5,A         ;converte para maiúscula
         CP     "S"         ;testa se sim
         SCF                ;seta o CARRY para caso seja sim
         JR     Z,SIMNA     ;se foi sim
         CP     "N"         ;testa se foi não, além disso se der a
                            ;igualdade reseta o CARRY
         JR     NZ,SIMNAO   ;se não foi o não, então vá ler o teclado
                            ;novamente. Isto fecha o laço até vir a
                            ;resposta desejada

SIMNA:   PUSH   AF          ;salva o CARRY (que contém a informa-
                            ;ção se foi sim ou não)
         CALL   BIOS6       ;ecoa a tecla pressionada sem conversão
         POP    AF          ;retorna o CARRY
         RET

LELIN:   LD     HL,LINHA    ;buffer de entrada do texto a ser digitado
         LD     (HL),A      ;primeiro byte = número máximo de
                            ;caracteres
         EX     DE,HL       ;o endereço do buffer deve estar no par
                            ;"DE"
         LD     C,10
         CALL   5           ;chama função de leitura de linha
         LD     A,(LINHA+1)
         OR     A           ;se zero, então nada foi digitado
         RET    Z
         LD     B,A         ;"B" = número de caracteres digitados
         LD     HL,LINHA+2  ;início dos caracteres digitados
         RET

LINHA:   DS     257         ;número máximo de caracteres a serem
                            ;lidos é 255, pois é o maior valor possí-
                            ;vel de 1 byte

FONTE:   DB     0

IMPRI:   LD     DE,SAIDA    ;área de montagem da linha de saída

IMPRIO:  LD     A,(HL)      ;busca um byte
         CP     1AH         ;testa se é fim de arquivo
         RET    Z           ;se for
         CP     9           ;testa se é tabulação
         CALL   Z,IMPRI1    ;se for
         LD     (DE),A      ;armazena
         INC    HL          ;incrementa ponteiros
         INC    DE
```

```
        SUB     0AH         testa se é mudança de linha
        JR      NZ,IMPRI0   ;se não é
        LD      (DE),A      ;0 indica fim de linha para LPRINT
        EX      DE,HL       ;salva "HL" pois LPRINT não destrói
                            ;"DE"
        LD      HL,SAIDA
        CALL    LPRINT      ;envia a linha
        EX      DE,HL       ;retorna ponteiro ("HL") de arquivo
        JR      IMPRI       ;e continua imprimindo

IMPRI1: PUSH    HL          ;salva ponteiro de arquivo
        PUSH    DE
        POP     HL          ;copia "DE" em "HL"
        LD      BC,SAIDA    ;início da linha
        OR      A           ;reseta o CARRY
        SBC     HL,BC       ;calcula a posição relativa ao início da
        LD      A,L
        AND     7           ;linha em resto da divisão por 8
        LD      C,A
        LD      HL,7        ;deveria ser 8-resto, mas ao retornar ao
                            ;a rotina principal, haverá mais um
        LD      B,H         ;armazenamento de espaço
        SBC     HL,BC       ;calcula quantos espaços são necessá-
                            ;rios
        LD      A,L         ;número de espaços
        POP     HL          ;recupera ponteiro de arquivo
        OR      A           ;testa se deve imprimir espaços
        LD      B,A         ;salva o número de espaços
        LD      A,20H       ;ASCII de espaço
        RET     Z           ;se não tem espaços

IMPRI2: LD      (DE),A      ;armazena um espaço na linha
        INC     DE          ;incrementa ponteiro da linha
        DJNZ    IMPRI2      ;decrementa contador de espaços e con-
                            ;tinua se # 0
        RET

VIDEO:  EX      (SP),HL     ;salva "HL" na pilha, "HL" aponta para o
                            ;começo do texto
        CALL    PRINT       ;envia texto para o vídeo
        EX      (SP),HL     ;recupera "HL", manda para pilha posi-
                            ;ção depois do byte 00
        RET                 ;que será o endereço de retorno

LPRINT: PUSH    BC          ;salva o par "BC"
        LD      C,5         ;indica saída na impressora
        JR      PRINT0

PRINT:  PUSH    BC          ;salva o par "BC"
```

```
        LD      C,6             ;indica saída no vídeo

PRINT0: PUSH    DE              ;salva o par "DE"

PRINT1: LD      A,(HL)          ;busca um caracter
        INC     HL              ;incrementa o ponteiro
        OR      A               ;testa se é o byte 00
        JR      Z,PRINT2        ;se for, então é fim de texto
        LD      E,A             ;não é, portanto vamos enviar o caracter
        PUSH    BC              ;salva indicador de vídeo/impressora
        PUSH    HL              ;salva ponteiro
        CALL    5               ;envia o caracter
        POP     HL              ;retorna o ponteiro
        POP     BC              ;e o indicador de vídeo/impressora
        JR      PRINT1          ;vá buscar o próximo caracter

PRINT2: POP     DE              ;recupera os pares "DE" e "BC"
        POP     BC
        RET

MKFCB:  DI                      ;bloqueia interrupção
        EXX                     ;salva os pares
        LD      HL,FCB+1
        LD      DE,FCB+2
        LD      BC,10
        LD      (HL),20H
        LDIR                    ;bytes correspondendo ao nome no FCB
                                ;recebem espaços brancos (ASCII 20H)
                                ;como default
        EXX                     ;restaura os pares
        EI                      ;libera interrupção
        INC     HL
        LD      A,(HL)          ;busca o segundo caracter da string
        DEC     HL
        CP      ":"             ;e testa se é ":" (indicativo de que o pri-
                                ;meiro caracter é indicativo de disco, e
                                ;não parte do nome)
        LD      A,0             ;indica disco default
        JR      NZ,MKFCB0       ;para o caso de não ser indicativo de
                                ;disco
        LD      A,(HL)          ;é indicativo, vejamos de que disco
        INC     HL
        INC     HL
        DEC     B
        DEC     B
        SCF                     ;indicativo de falha
        RET     Z               ;se só tem indicativo de disco
        AND     3
```

```
        SCF             ;indicativo de falha
        RET     Z       ;se indica disco "D"
        CP      3
        SCF             ;indicativo de falha
        RET     Z       ;se indica disco "C"

MKFCB0: LD      DE,FCB
        LD      (DE),A  ;armazena o disco no FCB
        INC     DE      ;e aponta para o inicio do nome
        LD      C,B     ;número de caracteres que restam na
                        ;string
        LD      B,8     ;número máximo de caracteres no nome
        CALL    MKFCB1  ;transfere o nome
        CALL    NC,MKFCB2 ;se não achou "." chama a busca do
                        ;mesmo
        JR      NC,CLRFCB ;se ainda assim não achou
        LD      DE,FCB+9 ;aponta para o inicio da extensão
        LD      B,3     ;número máximo de caracteres na exten-
                        ;são
        CALL    MKFCB1  ;transfere a extensão

CLRFCB: LD      HL,FCB+12 ;limpa o restante do FCB
        LD      DE,FCB+13
        LD      BC,23
        LD      (HL),B
        LDIR
        OR      A       ;reseta CARRY p/indicar operação bem-
                        ;sucedida
        RET

MKFCB1: LD      A,C
        OR      A       ;testa se ainda tem caracteres
        RET     Z       ;se não
        LD      A,(HL)  ;busca um caracter
        INC     HL      ;incrementa ponteiro do nome
        DEC     C       ;decrementa contador de caracteres do
                        ;nome
        CP      "."     ;testa se é fim do nome
        SCF             ;flag indicativo
        RET     Z       ;se for fim do nome
        CALL    MAIUS   ;converte para maiúscula
        LD      (DE),A  ;salva no FCB
        INC     DE      ;incrementa ponteiro do FCB
        DJNZ    MKFCB1  ;continua transferência enquanto for
                        ;válido
        OR      A       ;reseta o CARRY para indicar que não
                        ;achou o caracter "."
        RET
```

```
MKFCB2:  LD      A,(HL)      ;Busca 1 byte
         INC     HL          ;incrementa ponteiro
         OR      A           ;reseta o CARRY para indicar que não
                             ;achou
         DEC     C           ;decrementa contador de bytes
         RET     Z           ;se terminou a string
         CP      "."         ;testa se é o caracter procurado
         SCF                 ;para indicar que achou
         RET     Z           ;se for
         JR      MKFCB2      ;continua a busca

LOAD     PUSH    DE          ;salva início da posição de memória para
                             ;a leitura do arquivo
         CALL    ABRE        ;tenta abrir o arquivo
         POP     DE          ;recupera o início da memória
         INC     A           ;testa se conseguiu abrir ("A" = 0FFH se
                             ;não)
         RET     Z           ;se não

LOAD0:   LD      HL,128      ;tamanho de um registro a ser lido
         ADD     HL,DE       ;calcula a próxima posição de memória
         PUSH    HL          ;e salva
         CALL    SETDMA      ;envia posição de memória atual para o
                             ;sistema operacional
         CALL    LE          ;lê um registro
         POP     DE          ;retorna endereço do próximo registro
         OR      A           ;testa se leu sem erro o último registro
         JR      Z,LOAD0     ;se leu sem erro, vá ler o próximo
         RET                 ;se errou então é fim de arquivo
```

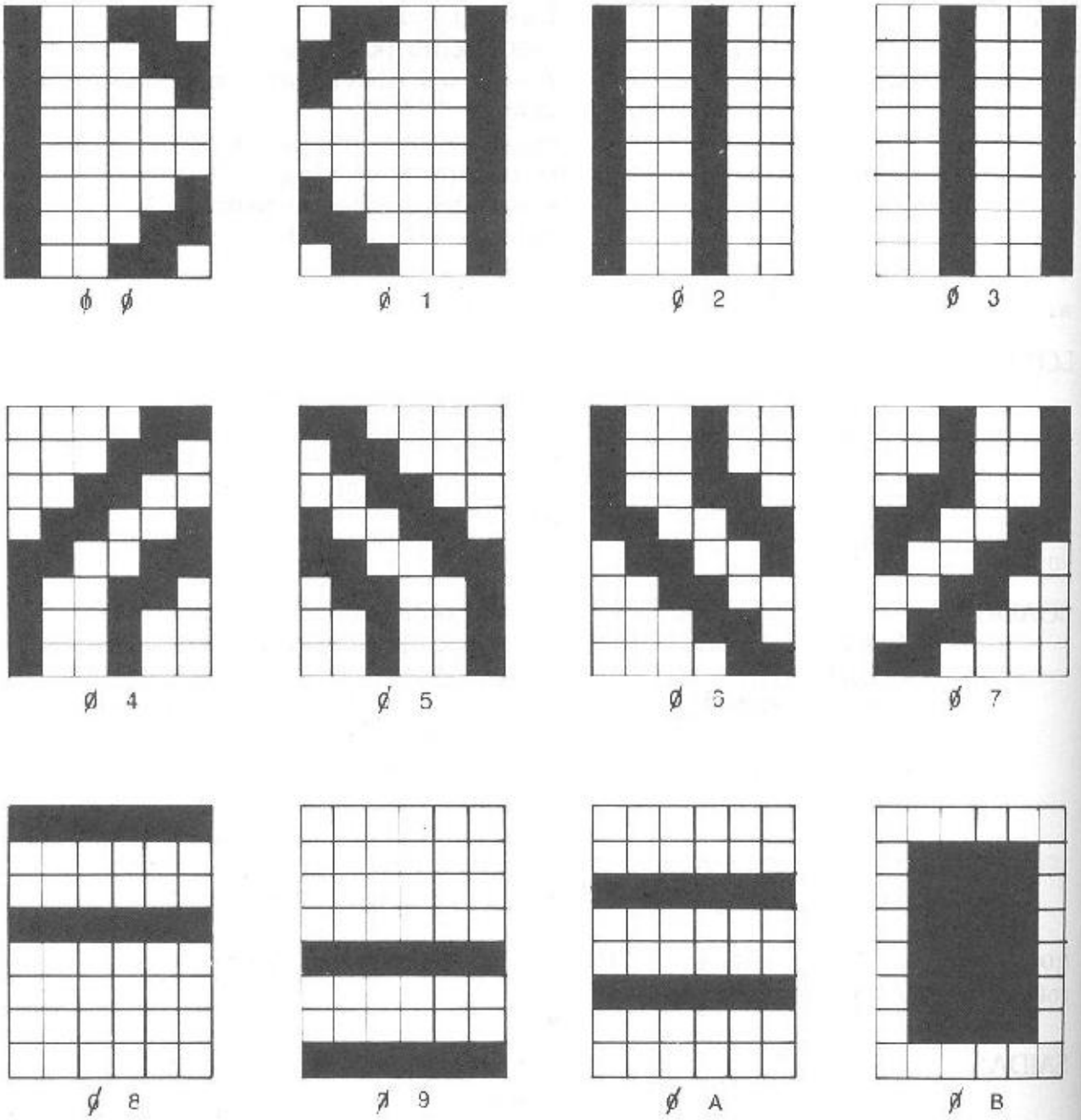
nota: se for MSX, antes de "RET" use um "CALL PARA", e copie a rotina "PARA" para o programa.

```
SAIDA:   DS      512         ;área de montagem de uma linha de
                             ;saída

BUFFER:  DB      0           ;só para indicar início da área de buffer
         END
```

TELA

Este programa desenha três telas no vídeo, e fornece meios para usá-las. Está dividido em três partes: área de dados, rotinas utilitárias e programa principal. O programa principal pode ser alterado para se tornar o seu programa, mas as outras duas partes devem permanecer inalteradas, à exceção dos bytes nos endereços 0102H a 0105H. Estes bytes possuem as seguintes funções: 0102H - número de telas (de 1 a 3), 0103H -



ØØ – CANTO ESQUERDO INTERMEDIÁRIO
Ø1 – CANTO DIREITO INTERMEDIÁRIO
Ø2 – MARGEM ESQUERDA
Ø3 – MARGEM DIREITA
Ø4 – CANTO ESQUERDO SUPERIOR
Ø5 – CANTO DIREITO SUPERIOR
Ø6 – CANTO ESQUERDO INFERIOR
Ø7 – CANTO DIREITO INFERIOR
Ø8 – MARGEM SUPERIOR
Ø9 – MARGEM INFERIOR
ØA – LINHA INTERMEDIÁRIA
ØB – CURSOR

Fig. 1 Os caracteres programados

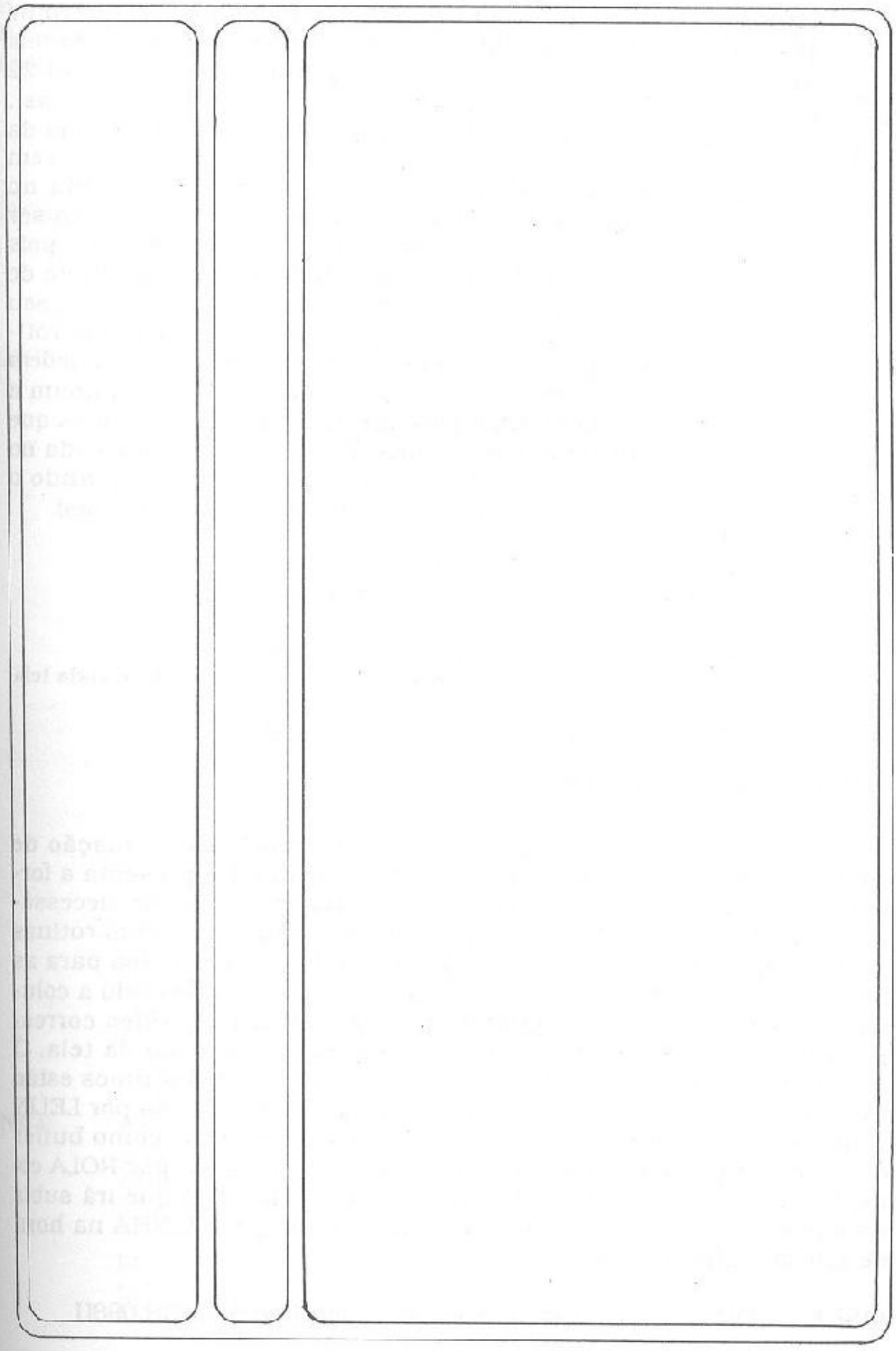


Fig. 2 As 3 telas desenhadas no vídeo

número de linhas da tela 1, 0104H - número de linhas da tela 2, 0105H - número de linhas da tela 3. A soma do número de linhas das telas será 22 (se só tiver uma tela), ou 21 (se tiver duas telas), ou 20 (se tiver todas as três telas). As rotinas da segunda parte (rotinas utilitárias) que tiverem mesmo nome que alguma outra já definida no livro executam a mesma função, mas devem ser usadas em lugar das definidas anteriormente pois estão adaptadas para trabalhar com o conjunto de telas. Tornar uma tela ativa significa colocar o seu endereço base no registrador "IX". Algumas rotinas pedem que a tela já esteja ativa, outras pedem o número da tela no registrador "C" e chamam a rotina SETAIX para ativar a tela. Estas rotinas que ativam a tela não fazem nada se a tela indicada no registrador "C" for inválida (por ex: 3 quando o valor em 0102H for 2, indicando só duas telas).

```
        .Z80            ;mnemônicos Z80
        ASEG            ;necessário para M80/L80
        ORG     100H
        JR      INICO   ;pula área de dados
        DB      3,4,1,15 ;número de telas e tamanho de cada tela

CURSOR: DB      1BH,"x5$"  ;para eliminar o cursor

BYTES:  DB      0F4H,087H,000H,048H

;muda as cores da tela e aponta para a memória de formação de
;caracteres de vídeo a partir deste ponto cada linha representa a for-
;mação de 1 byte. Após a programação, estes bytes são desnecessá-
;rios, e a área de memória é liberada para a utilização pelas rotinas
;utilitárias. BASE1, BASE2 e BASE3 são vetores de 8 bytes para as
;telas 1, 2 e 3. O primeiro byte é a linha do cursor, o segundo a colu-
;na, o terceiro e o quarto apontam para a memória de vídeo corres-
;pondente à linha 0 coluna 0 (canto superior esquerdo) da tela. O
;quinto byte possui o número de linhas da tela, e os 3 últimos estão
;livres para utilização pelo seu programa. LINHA é utilizada por LELIN
;como buffer de entrada da linha digitada e por ROLA como buffer
;de saída para o vídeo de uma linha. Já LINHAT é usada por ROLA co-
;mo buffer de entrada de vídeo (lê uma linha de vídeo que irá subir
;uma posição). O seu conteúdo será transferido para LINHA na hora
;de sair na linha superior.

BASE1:  DB      098H,08CH,084H,080H,080H,084H,08CH,098H
```

```
BASE2:  DB      064H,0C4H,084H,004H,004H,084H,0C4H,064H
BASE3:  DB      090H,090H,090H,090H,090H,090H,090H,090H
LINHA:  DB      024H,024H,024H,024H,024H,024H,024H,024H
        DB      00CH,018H,030H,064H,0CCH,098H,090H,090H
        DB      0C0H,060H,030H,098H,0CCH,064H,024H,024H
        DB      090H,090H,098H,0CCH,064H,030H,018H,00CH
        DB      024H,024H,064H,0CCH,098H,030H,060H,0C0H
LINHAT: DB      0FCH,000H,000H,0FCH,000H,000H,000H,000H
        DB      000H,000H,000H,000H,0FCH,000H,000,H0FCH
        DB      000H,000H,0FCH,000H,000H,0FCH,000H,000H
        DB      000H,078H,078H,078H,078H,078H,078H,000H

INIC0:  DI                      ;bloqueia interrupção para poder progra-
                                ;mar o video
        LD      HL,BYTES        ;inicio dos bytes de programação
        LD      BC,0499H        ;bytes para a porta 99H
        IN      A,(99H)         ;sincroniza o controlador de video
        OTIR                    ;envia os 4 bytes
        LD      BC,6098H        ;96 bytes para 98H (formação dos carac-
                                ;teres)
        OTIR                    ;envia os bytes para porta
        EI                      ;libera interrupção
        LD      SP,INIC0        ;pilha temporária, pois por enquanto-
                                ;LINHAT e LINHA não serão utilizadas.
        LD      DE,CURSOR
        LD      C,9
        CALL    5               ;desliga cursor
        LD      DE,40           ;tamanho de uma linha real do video
        LD      HL,41           ;primeira posição de memória da pri-
                                ;meira tela
        LD      (BASE1+2),HL    ;independente do seu tamanho
        LD      BC,(102H)       ;"C" = número de telas, "B" = tamanho da
                                ;primeira tela
        LD      A,B
        DEC     C
        JR      NZ,INIC1        ;se tem mais de uma tela
        LD      A,22            ;tamanho de tela quando é única
        LD      (BASE1+4),A
        JR      INIC4           ;vai desenhar a tela

INIC1:  ADD     HL,DE           ;cálculo da primeira posição de memória
        DJNZ    INIC1           ;da segunda tela
        ADD     HL,DE
        LD      (BASE2+2),HL    ;armazena a posição de memória calcu-
                                ;lada
        LD      B,A             ;recupera tamanho da primeira tela
        LD      (BASE1+4),A     ;e armazena
        LD      A,(104H)        ;busca o tamanho da segunda tela
```

```
        DEC     C               ;testa se tem uma terceira tela
        JR      Z,INIC3         ;se não tem

;chegando aqui então são três telas
        LD      (BASE2+4),A     ;armazena o tamanho da segunda tela
        LD      C,A             ;salva temporária
        ADD     A,B             ;soma com o tamanho da primeira tela
        LD      B,C             ;tamanho da segunda tela fica em "B"
        LD      C,A             ;soma dos dois tamanhos
        LD      A,20            ;a soma dos três tamanhos deve ser 20
        SUB     C               ;portanto, calcula-se o tamanho da ter-
                                ;ceira tela
        LD      (BASE3+4),A     ;e armazena

INIC2:  ADD     HL,DE           ;cálculo da primeira posição de memória
                                ;da terceira tela
        DJNZ    INIC2
        ADD     HL,DE
        LD      (BASE3+2),HL    ;armazena a posição de memória calcu-
                                ;lada
        JR      INIC4           ;vai desenhar, a tela

INIC3:  LD      A,21            ;a soma dos dois tamanhos deve ser 21
        SUB     B               ;calcula o tamanho da segunda tela
        LD      (BASE2+4),A     ;e armazena

INIC4:  DI                      ;bloqueia a interrupção, não há necessi-
                                ;dade de sincronismo pois não hou-
        XOR     A               ;ve nenhuma possibilidade de comando
                                ;para o controle de video
        OUT     (99H),A         ;envia endereço 0000H (caracter 00H)
        LD      A,40H
        OUT     (99H),A
        LD      A,4             ;caracter que desenha o canto esquerdo
                                ;superior da tela para a primeira posição
                                ;de tela
        OUT     (98H),A
        LD      A,8             ;caracter que desenha a linha superior
        CALL    INIC7           ;envia 38 vezes este caracter
        LD      A,5             ;caracter do canto direito superior da
                                ;tela
        OUT     (98H),A
        LD      C,22            ;22 linhas

INIC5:  LD      A,2             ;caracter da margem esquerda
        OUT     (98H),A
        LD      A,20H           ;caracter ASCII de espaço em branco
        CALL    INIC7           ;é enviado 38 vezes
        LD      A,3             ;caracter da margem direita
```

```
        OUT     (98H),A
        DEC     C
        JR      NZ,INIC5        ;se ainda tem mais linhas
        LD      A,6             ;caracter do canto esquerdo inferior
        OUT     (98H),A
        LD      A,9             ;caracter da linha inferior
        CALL    INIC7           ;é enviado 38 vezes
        LD      A,7             ;caracter do canto direito inferior
        OUT     (98H),A
        LD      A,(102H)        ;busca o número de telas
        DEC     A
        JR      Z,INIC6         ;se só tem uma tela
        LD      C,A             ;salva número de telas que restam
        LD      HL,(BASE2+)     ;busca o início da segunda tela
        CALL    INIC9           ;e desenha uma linha divisória na linha
                                ;superior
        DEC     C
        JR      Z,INIC6         ;se eram só duas telas
        LD      HL,(BASE3+2)    ;busca o início da terceira tela
        CALL    INIC9           ;e desenha uma linha divisória na linha
                                ;superior
INC6:   LD      SP,INIC9        ;endereço de pilha definitivo
        LD      HL,0            ;posiciona os cursores de todas as telas
                                ;no canto esquerdo superior
        LD      (BASE1),HL
        LD      (BASE2),HL
        LD      (BASE3),HL
        EI                      ;libera a interrupção
        JP      PROG            ;e vai executar o programa principal

INIC7:  LD      B,38            ;número de vezes a enviar o caracter

INIC8:  OUT     (98H),A         ;envia
        DJNZ    INIC8           ;decrementa contador e volta a enviar se
                                ;não terminou
        RET

INIC9:  LD      DE,41           ;cálculo da posição de memória onde
        OR      A               ;inicia a linha divisória entre duas telas
        SBC     HL,DE
        LD      A,L             ;envia o endereço
        OUT     (99H),A
        LD      A,H
        SET     6,A
        OUT     (99H),A
        XOR     A               ;caracter do canto esquerdo da linha
                                ;divisória
        OUT     (98H),A
```

```
        LD      A,0AH       ;caracter de linha intermediária
        CALL    INIC7       ;é enviado 38 vezes
        LD      A,1         ;caracter do canto direito da linha divi-
                            ;sória
        OUT     (98H),A
        RET

;ROTINAS UTILITÁRIAS
;seta o ponteiro do vetor de tela em função da tela pedida em "C"

SETAIX: LD      A,C         ;busca a tela desejada e testa se é válida
        OR      A
        SCF                 ;para indicar não válida
        RET     Z           ;se não válida
        LD      A,(102H)    ;busca o número de telas existentes
        CP      C           ;testa se a tela desejada existe
        RET     C           ;se não existe
        LD      IX,BASE1    ;ponteiro da primeira tela
        DEC     C
        RET     Z           ;se era a primeira tela a desejada
        LD      IX,BASE2    ;ponteiro da segunda tela
        DEC     C
        RET     Z           ;se era a segunda tela a desejada
        LD      IX,BASE3    ;ponteiro da terceira tela
        RET

;envia a posição de memória do cursor. O ponteiro da tela já deve
;estar em "IX"

SETMEM: DI                  ;bloqueia interrupção
        EXX                 ;salva os registradores
        LD      H,(IX+3)    ;busca a posição de memória do canto
                            esquerdo
        LD      L,(IX+2)    ;superior da tela apontada por "IX"
        LD      DE,40       ;tamanho de linha real
        LD      A,(IX+0)    ;busca o número da linha do cursor
        OR      A
        JR      Z,SETME1    ;se é a primeira linha
        LD      B,A

SETME0: ADD     HL,DE       ;cálculo da posição de memória do canto
        DJNZ                ;esquerdo da linha desejada

SETME1: LD      E,(IX+1)    ;busca a coluna do cursor
        ADD     HL,DE       ;cálculo final da posição do cursor
        SET     6,H         ;para indicar que é um endereço de
                            ;memória
        LD      C,99H       ;endereço da porta
```

```
        IN      A,(C)       ;sincronismo
        OUT     (C),L       ;envia o endereço de memória calculado
        OUT     (C),H
        EXX                 ;retorna os registradores
        RET
```

;rotina GET já definida anteriormente no livro

```
GET:    CALL    INKEY
        OR      A
        JR      Z,GET
        RET
```

;rotina para ler o teclado sem destruir os registradores importantes

```
INKEY:  PUSH    BC
        PUSH    DE
        PUSH    HL
        PUSH    IX
        LD      E,255
        LD      C,6
        CALL    5
        POP     IX
        POP     HL
        POP     DE
        POP     BC
        RET
```

;nova rotina BIOS6, obedecendo a tela ativa (indicada por "IX"): Obedece as margens, não altera as outras telas, rola somente a tela ativa, limpa somente a tela ativa, etc.

```
BIOS6:  LD      A,E
        CP      20H         ;testa se é caracter de controle
        JR      C,BIOS6D    ;se for
        INC     A           ;testa se é para ler o teclado
        JR      Z,INKEY     ;se for
        CALL    SETMEM      ;envia o endereço da posição do cursor
        LD      A,E
        OUT     (98H),A     ;envia o caracter
        EI                  ;libera a interrupção (bloqueada por
                            ;SETMEM)
        LD      A,(IX+1)    ;busca a coluna do cursor
        INC     A           ;incrementa
        CP      38          ;testa se terminou a linha
        JR      C,BIOS6C    ;se não terminou
```

```
BIOS6A:  LD      A,(IX+0)      ;busca a linha do cursor
         INC     A             ;incrementa
         CP      (IX+4)        ;testa se terminou a tela
         CALL    NC,ROLA       ;se terminou rola a tela para criar nova
                               ;linha
         LD      (IX+0),A      ;armazena a nova linha do cursor

BIOS6B:  XOR     A             ;coluna 0

BIOS6C:  D,(IX+1),A            ;armazena a nova coluna do cursor
         RET

BIOS6D:  CP      0AH           ;testa se é para mudar de linha
         JR      Z,BIOS6A      ;se for
         CP      0DH           ;testa se é para ir para o inicio da linha
         JR      Z,BIOS6B      ;se for
         CP      0CH           ;testa se é para limpar a tela
         JR      Z,LIMPA0      ;se for
         CP      8             ;testa se é para retornar uma posição
         RET     NZ            ;se não for
         LD      A,(IX+)       ;busca a coluna do cursor
         SUB     1             ;decrementa, setando o CARRY se era 0
         JR      C,BIOS6B      ;se era, então continua 0
         JR      BIOS6C        ;vai armazenar a nova coluna

;rotina de limpar a tela apontada por "C"

LIMPAT:  CALL    SETAIX        ;seta o ponteiro da tela
         RET     C             ;se tela não válida

;aqui inicia a rotina se a tela já está ativa

LIMPA0:  LD      C,(IX+4)      ;busca o número de linhas da tela

LIMPA1:  DEC     C
         LD      (IX+0),C      ;envia um número de linha para o cur-
                               ;sor
         CALL    LIMPAL        ;e limpa toda a linha apontada pelo
                               ;mesmo
         LD      A,C
         OR      A             ;testa se ainda tem linhas para limpar
         JR      NZ,LIMPA1     ;se tem
         RET

;rotina para limpar toda a linha apontada pelo cursor, posicionando o
mesmo no início da referida linha

LIMPAL:  LD      (IX+1),0      ;indica coluna 0
```

```
;início da rotina para limpar a linha da posição do cursor até o fim da
mesma, sem alterar a posição do cursor

LIMPAR:  CALL   SETMEM      ;envia o endereço do cursor
         LD     A,38
         SUB    (IX+1)      ;calcula quantas colunas devem ser lim-
                            ;pas
         LD     B,A
         LD     A,20H       ;ASCII de espaço em branco

LIMPA2:  OUT    (98H),A     ;limpa uma coluna
         DJNZ   LIMPA2      ;decrementa contador, pula se não ter-
                            ;minou

         EI                 ;libera interrupção
         RET

;rotina para rolar a tela ativa

ROLA:    PUSH   HL          ;salva o par "HL"
         LD     HL,LINHA    ;gera uma linha limpa
         LD     DE,LINHA+1
         LD     BC,37
         LD     (HL),20H
         LDIR
         LD     C,(IX+4)    ;busca o número de linhas a rolar

ROLA0:   CALL   ROLA1       ;rola a linha apontada por "C"-1
         DEC    C           ;decrementa o contador
         JR     NZ,ROLA0    ;se não terminou
         POP    HL          ;recupera o par "HL"
         LD     A,(IX+4)    ;busca o número de linhas da tela
         DEC    A           ;número da última linha válida
         RET

ROLA1:   LD     A,C
         DEC    A           ;calcula a linha desejada
         LD     (IX+0),A    ;posiciona o cursor no início desta linha
         LD     (IX+1),0
         LD     B,39        ;vamos ler 39 bytes, pois o primeiro será
         LD     HL,LINHAT   ;o último enviado para o vídeo. Lê pa-
                            ;ra LINHAT

ROLA2:   CALL   SETMEM      ;envia endereço do cursor
         IN     A,(98H)     ;lê o vídeo
         LD     (HL),A      ;armazena
         INC    HL          ;incrementa ponteiro
         INC    (IX+1)      ;incrementa coluna do cursor
```

```
        DJNZ    ROLA2       ;decrementa contador, pula se não ter-
                            ;minou
        LD      (IX+1),0    ;seta coluna 0
        PUSH    BC          ;salva o par "BC"
        LD      BC,2698H    ;38 bytes para a porta 98H
        LD      HL,LINHA    ;enviados do buffer LINHA
        CALL    SETMEM      ;envia o endereço do cursor
        OTIR                ;e envia os bytes da linha
        LD      HL,LINHAT+1
        LD      DE,LINHA
        LD      BC,38
        LDIR                ;transfere os bytes lidos da linha para
                            ;LINHA
        POP     BC          ;recupera o par "BC"
        EI                  ;libera a interrupção
        RET

;rotina para posicionar o cursor da tela indicada por "C" na linha
;indicada por "L" e coluna indicada por "H"

LOCATE: CALL    SETAIX      ;ativa a tela desejada
        RET     C           ;se não for válida
        LD      A,(IX+4)    ;busca o número de linhas da tela
        DEC     A           ;"A" = maior linha válida
        CP      L           ;testa se a linha desejada é válida
        JR      C,$+3       ;se não for "A" permanece com o maior
                            ;valor possivel
        LD      A,L         ;busca o número da linha desejada
        LD      (IX+0),A    ;armazena o número da linha
        LD      A,37        ;maior valor possivel para coluna
        CP      H           ;testa se a coluna desejada é válida
        JR      C,$+3       ;se não for, "A" permanece com o maior
                            ;valor possivel
        LD      A,H         ;busca o número da coluna desejada
        LD      (IX+1),A    ;armazena o número da coluna
        RET

;rotina para ler uma linha do teclado. O tamanho máximo é dado pela
;posição do cursor, pois no máximo irá até a coluna número 36. A
;tela já deve estar ativa (ponteiro "IX" setado).

LELIN:  LD      HL,LINHA+1  ;inicio do buffer de teclado
        LD      B,0         ;número inicial de caracteres lidos
        LD      A,(IX+1)    ;busca a coluna do cursor
        CP      37          ;testa se é válido ler o teclado a partir
                            ;da mesma
        RET     NC          ;se não for
```

```
LELIN0:  CALL   SETMEM      ;envia o endereço do cursor
         LD     A,0BH       ;envia o caracter de cursor para orienta-
                            ;ção
         OUT    (98H),A     ;de quem estiver digitando
         EI                 ;libera a interrupção
         CALL   GET         ;lê uma tecla
         CP     20H         ;testa se é caracter de controle
         JR     C,LELIN1    ;se for
         LD     E,A         ;salva o caracter lido
         LD     A,(IX+1)    ;busca a coluna do cursor
         CP     37          ;testa se ainda é válida
         JR     Z,LELIN2    ;se não
         LD     (HL),E      ;armazena o caracter lido
         INC    HL          ;incrementa o ponteiro
         INC    B           ;incrementa o contador
         PUSH   BC          ;salva o contador
         CALL   BIOS6       ;envia o caracter para a tela (apaga cur-
                            ;sor)
         POP    BC          ;retorna o contador
         JR     LELIN0      ;e vai continuar lendo o teclado

LELIN1:  CP     0DH         ;testa se é retorno de carro
         JR     Z,LELIN2    ;se for
         CP     8           ;testa se é retrocesso
         JR     NZ,LELIN0   ;se não for
         LD     A,B         ;busca o número de caracteres lidos
         OR     A           ;testa se é 0
         JR     Z.LELIN0    ;se for
         DEC    B           ;decrementa o contador
         CALL   SETMEM      ;envia endereço do cursor
         LD     A,20H       ;ASCII de espaço em branco
         OUT    (98H),A     ;apaga o cursor
         LD     (HL),A      ;apaga no buffer
         DEC    HL          ;decrementa ponteiro do buffer
         DEC    (IX+1)      ;decrementa coluna do cursor
         JR     LELIN0      ;vai ler a próxima tecla

LELIN2:  CALL   SETMEM      ;envia endereço do cursor
         LD     A,20H       ;ASCII de espaço em branco
         OUT    (98H),A     ;apaga o cursor para indicar fim de lei-
                            ;tura
         EI                 ;libera a interrupção
         LD     HL,LINHA
         LD     (HL),B      ;torna a saida compatível com a rotina
                            ;LELIN
         INC    HL          ;original até mesmo a nível de buffer
         RET
```

```
;rotina para enviar um texto apontado pelo par "HL" e terminado pelo
;byte 00H para a tela indicada por "C"

PRINIT:  CALL   SETAIX      ;ativa a tela
         RET    C           ;se não for válida

;inicio da rotina para enviar um texto para uma tela já ativa

PRINT:   LD     A,(HL)      ;busca um byte
         INC    HL          ;incrementa o ponteiro
         OR     A           ;testa se é fim de texto
         RET    Z           ;se for
         PUSH   BC          ;salva os pares "BC" e "DE"
         PUSH   DE
         LD     E,A
         CALL   BIOS6       ;envia o byte
         POP    DE          ;recupera os pares salvos
         POP    BC
         JR     PRINT       ;e vai enviar o próximo byte

;rotina para enviar o texto que segue a chamada para a tela indicada
por "C".

TELA:    CALL   SETAIX      ;ativa a tela
         JR     NC,VIDEO    ;se tela válida
         EX     (SP),HL     ;busca o inicio do texto

TELA0:   LD     A,(HL)      ;busca um byte
         INC    HL          ;incrementa o ponteiro
         OR     A           ;testa se é fim do texto
         JR     NZ,TELA0    ;se não
         EX     (SP),HL     ;aponta como endereço de retorno o
                            ;byte seguinte ao texto
         RET

;rotina para enviar o texto que segue a chamada para a tela ativa

VIDEO:   EX     (SP),HL     ;busca o inicio do texto
         CALL   PRINT       ;envia
         EX     (SP),HL     ;envia para a pilha o endereço de
                            ;retorno
         RET

;inicio do programa principal. Pode ser o seu programa particular.
Aqui será um programa, exemplificando as rotinas aqui apresentadas

PROG:    LD     HL,3
         LD     C,1         ;posiciona o cursor da tela 1
```

```
        CALL    LOCATE      ;no inicio da última linha de tela
        LD      HL,MENS     ;ponteiro da mensagem inicial
        LD      B,10        ;que é composta de 10 linhas

PROG0:  CALL    PRINT       ;envia texto até encontrar o byte 00H
        LD      C,120       ;2 segundos
        CALL    TEMPO       ;é o tempo de espera para
        DJNZ    PROG0       ;decrementar o contador e enviar a
                            ;próxima linha, se houver
        CALL    GET         ;aguarda teclar algo para continuar o
                            ;programa
        CALL    VIDEO       ;já está na tela 1, envia o texto abaixo
        DB      0CH," Cronômetro    LIMPAT" ;0CH limpa a tela 1
        DB      0AH," LIMPAL        LIMPAR"
        DB      0AH," LOCATE        Caracteres"
        DB      0AH," LELIN         Fim",0
        LD      HL,0
        LD      (BASE1),HL  ;coloca o cursor da tela 1 na posição 0,0
        JR      LACO4       ;e vai colocar a mensagem da tela 2

LACO:   CALL    GET         ;busca um comando
        LD      IX,BASE1    ;ativa a tella 1 (não usou SETAIX)
        CP      1FH         ;testa se é seta para baixo
        JR      Z,LACO0     ;se for
        JR      NC,LACO     ;se for letra, número ou simbolo ( AS-
                            CII > 1FH )
        CP      1DH         ;testa se é seta para esquerda
        JR      Z,LACO3     ;se for
        JR      NC,LACO1    ;se for seta para cima (1EH)
        CP      1CH         ;testa se é seta para direita
        JR      Z,LACO5     ;se for
        CP      18H         ;testa se é a tecla <  SELECT  >
        JR      NZ,LACO     ;se não for
        CALL    PULAHL      ;chama a rotina apontada por "HL"
        LD      C,3
        CALL    LIMPAT      ;limpa a tela 3
        LD      IX,BASE1    ;ativa a tela 1
        JR      LACO4       ;e vai verificar posição do cursor para
                            ;gerar a mensagem correspondente na
                            ;tela 2
LACO0:  LD      A,(IX+0)    ;busca a linha
        INC     A           ;incrementa
        JR      LACO2

LACO1:  LD      A,(IX+0)    ;busca a linha
        DEC     A           ;decrementa

LACO2:  AND     3           ;obtém o resto da divisão por 4
        PUSH    AF          ;salva
```

```
        CALL    SETMEM      ;envia o endereço do cursor atual
        LD      A,20H
        OUT     (98H),A     ;limpa esta posição
        POP     AF          ;recupera a nova linha
        EI                  ;habilita a interrupção
        LD      (IX+0),A    ;armazena a nova linha
        JR      LACO4       ;vai gerar a mensagem da tela 2 corres-
                            ;pondente à nova posição do cursor

LACO3   LD      A,(IX+1)    ;busca a coluna. Como só existem duas
                            ;posições, esquerda ou direita, é sim-
                            ;plesmente trocá-las
        XOR     19
        PUSH    AF          ;salva a nova coluna
        CALL    SETMEM      ;envia o endereço do cursor atual
        LD      A,20H
        OUT     (98H),A     ;limpa esta posição
        POP     AF          ;recupera a nova coluna
        EI                  ;habilita a interrupção
        LD      (IX+1),A    ;e armazena a nova coluna

LACO4:  CALL    SETMEM      ;envia o endereço da nova posição de
                            ;cursor
        LD      A," > "     ;símbolo do cursor na tela 1
        OUT     (98H),A     ;coloca o símbolo do cursor na nova
                            ;posição
        EI                  ;habilita a interrupção
        LD      A,(IX+1)    ;busca a coluna (00H ou 13H)
        INC     A
        AND     4           ;o resultado será 00H ou 04H
        ADD     A,(IX+0)    ;soma com o número da linha
                            ;(de 00H a 03h)
        ADD     A,A         ;multiplica por 2 para obter a posição
                            ;relativa da rotina de tratamento desta
                            ;posição
        LD      L,A
        LD      H,0         ;de cursor na string "VETOR"
        LD      DE,VETOR    ;início da string
        ADD     HL,DE       ;obtém a posição relativa
        LD      E,(HL)      ;busca o endereço do início da rotina de
                            ;tratamento no par "DE"
        INC     HL
        LD      D,(HL)
        EX      DE,HL       ;coloca o endereço no par "HL"
        CALL    PULAHL      ;e chama a rotina que irá colocar uma
                            ;mensagem na tela 2 e o endereço do
                            ;programa de demonstração no par "HL"
                            ;para ser usado quando se pressionar a
                            ;tecla < SELECT .
```

```
        JR      LACO        ;vai aguardar a próxima tecla

PULAHL: JP      (HL)        ;vá para o endereço apontado pelo par
                            ;"HL"

PROG1:  LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      0CH,"Cronômetro com 1/10 do segundo",0
        LD      HL,PROG10   ;endereço de rotina correspondente
        RET

PROG2:  LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      0CH,"Limpa uma linha",0
        LD      HL,PROG20   ;endereço de rotina correspondente
        RET

PROG3:  LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      CH,"Demonstra a rotina LOCATE",0
        LD      HL,PROG30:  ;endereço de rotina correspondente
        RET

PROG4:  LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      0CH,"Demonstra a rotina LELIN",0
        LD      HL,PROG40   ;endereço de rotina correspondente
        RET

PROG5:  LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      0CH,"Limpa a tela toda",0
        LD      HL,PROG50   ;endereço de rotina correspondente
        RET

PROG6:  LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      0CH,"Limpa uma linha a partir do cursor",0
        LD      HL,PROG60   ;endereço de rotina correspondente
        RET

PROG7:  LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      0CH,"Apresenta todos os caracteres",0
        LD      HL,PROG70   ;endereço de rotina correspondente
        RET
```

```
PROG8:  LD      C,2
        CALL    TELA            ;mensagem na tela 2
        DB      0CH,"Fim de programa",0
        LD      HL,PROG80       ;endereço de rotina correspondente
        RET

;BUFFER do cronômetro
HORA:   DB      0
MINUTO: DB      0
SEGUND: DB      0
DECIMO: DB      0
ATIVO:  DB      0               ;se diferente de 0 então está ativo
PROG10: LD      C,2
        CALL    TELA            ;mensagem na tela 2
        DB      0CH,"Comandos: Início Para Limpa Fim",0
        XOR     A
        LD      (0FC9EH),A      ;zera parte baixa de "TIME"

PROG11: CALL    PROG15          ;parte ativa do cronômetro, só retorna se
                                ;algo for teclado
        CP      "I"             ;testa se é comando de ativar o cronô-
                                ;metro
        JR      Z,PROG13        ;se for
        CP      "P"             ;testa se é comando de parar o cronôme-
                                ;tro
        JR      Z,PROG14        ;se for
        CP      "L"             ;testa se é comando de limpar o cronô-
                                ;metro
        CALL    Z,PROG12        ;se for
        CP      "F"             ;testa se é fim da rotina
        JR      NZ,PROG11       ;se não for
        XOR     A
        LD      (ATIVO),A       ;pára o cronômetro

PROG12: LD      A,(ATIVO)
        OR      A               ;testa se o cronômetro está ativo
        RET     NZ              ;se estiver
        LD      HL,0
        LD      (HORA),HL       ;zera hora e minuto
        LD      (SEGUND),HL     ;zera segundo e décimo de segundo
        RET

PROG13: LD      A,1
        LD      (ATIVO),A       ;ativa o cronômetro
        JR      PROG11          ;continua processo principal do cronô-
                                ;metro

PROG14: XOR     A
```

```
        LD      (ATIVO),A   ;pára o cronômetro
        JR      PROG11      ;continua processo principal do cronô-
                            ;metro

PROG15: LD      C,3
        LD      HL,0E07H
        CALL    LOCATE      ;posiciona o cursor da tela 3
        LD      HL,HORA     ;ponteiro de saída dos bytes
        LD      B,3         ;3 bytes

PROG16: PUSH    BC          ;salva contador de bytes
        LD      A,(HL)      ;busca um byte
        INC     HL          ;incrementa ponteiro
        CALL    PROG73      ;envia o byte; note que esta rotina a
                            ;menos de alguns detalhes, é igual a
                            ;rotina SAIHEX
        LD      E,":"
        CALL    BIOS6       ;envia ":" para a tela
        POP     BC          ;retorna contador de bytes
        DJNZ    PROG16      ;decrementa e pula se não terminou
        LD      A,(HL)      ;busca os décimos de segundo
        CALL    PROG74      ;e envia somente os 4 bits menos signi-
                            ;ficativos para a tela
        LD      HL,0FC9EH   ;endereço da parte menos significativa
                            ;de "TIME"

PROG17: LD      A,(HL)      ;busca "TIME"
        SUB     6           ;testa se já passou um décimo de
                            ;segundo
        JR      NZ,PROG17   ;se não
        LD      (HL),A      ;zera "TIME"
        LD      A,(ATIVO)
        OR      A           ;testa se cronômetro está ativo
        JR      Z,PROG18    ;se não
        LD      HL,DECIMO   ;ponteiro
        LD      C,10H       ;valor de virada
        CALL    PROG19      ;incrementa décimo de segundo
        LD      C,60H       ;valor de virada para segundo e minuto
        CALL    Z,PROG19    ;se décimo virou, incrementa segundo
        CALL    Z,PROG19    ;se segundo virou, incrementa minuto
        LD      C,24        ;valor de virada para hora
        CALL    Z,PROG19    ;se minuto virou, incrementa hora

PROG18: LD      A,(0FC9EH)
        OR      A           ;testa se "TIME" ainda é 0
        JR      Z,PROG18    ;se ainda for
        CALL    INKEY       ;lê o teclado
        OR      A           ;testa se veio alguma tecla
```

```
        JR      Z,PROG15    ;se não veio continua o processo princi-
                            ;pal
        RES     5,A         ;converte para letra maiúscula qualquer
                            ;letra
        RET
PROG19: LD      A,(HL)      ;busca o byte apontado
        INC     A           ;incrementa
        DAA                 ;em decimal
        CP      C           ;testa se é valor de virada
        JR      NZ,$+3      ;se não for
        XOR     A           ;virou = zera o valor e seta o flag "Z"
        LD      (HL),A      ;armazena novo valor
        DEC     HL          ;aponta para o próximo byte a ser traba-
                            ;lhado
        RET

PROG20: CALL    ENCHE       ;enche a tela 3 com "*"
        LD      C,2
        CALL    TELA        ;mensagem na tela 2
        DB      0CH,"Qual a linha desejada? ",0
        CALL    LENUM       ;busca um número de dois dígitos
        LD      L,A
        LD      C,3
        CALL    LOCATE      ;que será a nova linha do cursor de
                            ;tela 3
        CALL    LIMPAL      ;limpa esta linha
        CALL    REPETE      ;pergunta se repete a rotina, aguarda
                            ;resposta
        JR      C,PROG20    ;se afirmativo
        RET

PROG30: CALL    PROG61      ;pede linha e coluna, coloca em "HL"
        LD      C,3
        CALL    SETAIX      ;ativa a tela 3
        CALL    SETMEM      ;envia endereço do cursor
        LD      A,20H
        OUT     (98H),A     ;limpa a posição
        LD      C,3         ;indica tela 3
        CALL    LOCATE      ;novo cursor = "HL"
        CALL    SETMEM      ;envia o endereço do novo cursor
        LD      A,0BH
        OUT     (98H),A     ;envia o caracter de cursor para a tela
        EI                  ;habilita a interrupção
        CALL    REPETE      ;pergunta se repete a rotina, aguarda
                            ;resposta
        JR      C,PROG30    ;se afirmativo
        RET
```

```
PROG40:  LD      C,2
         CALL    TELA          ;mensagem na tela 2
         DB      CH,"Digite um texto",0
         LD      C,3
         LD      HL,2
         CALL    LOCATE        ;posiciona o cursor no início da linha 2
         CALL    LELIN         ;lê uma linha de teclado
         PUSH    HL            ;salva o ponteiro da linha lida
         PUSH    BC            ;e o número de bytes desta linha
         LD      HL,206H
         LD      C,3
         CALL    LOCATE        ;nova posição de cursor
         CALL    VIDEO         ;mensagem na tela 3 (que está ativa)
         DB      "O texto digitado foi:",0AH,0AH,0
         POP     BC            ;recupera número de bytes
         POP     HL            ;e o ponteiro da linha lida

PROG41:  LD      E,(HL)        ;busca um byte
         INC     HL            ;incrementa o ponteiro da linha
         PUSH    BC            ;salva o contador
         CALL    BIOS6         ;envia o byte para a tela 3
         POP     BC            ;retorna o contador
         DJNZ    PROG41        ;decrementa e pula se não terminou
         CALL    REPETE        ;pergunta se repete a rotina, aguarda
                               ;resposta
         RET     NC            ;se não repete
         LD      C,3
         CALL    LIMPAT        ;limpa a tela 3
         JR      PROG40        ;e repete a rotina

PROG50:  CALL    ENCHE         ;enche a tela 3 com "*"
         CALL    TECLE         ;pede para teclar algo e aguarda uma
                               ;tecla
         LD      C,3
         CALL    LIMPAT        ;e limpa a tela 3
         CALL    REPETE        ;pergunta se repete a rotina, aguarda
                               ;resposta
         JR      C,PROG50      ;se afirmativo
         RET

PROG60:  CALL    ENCHE         ;enche a tela 3 com "*"
         CALL    PROG61        ;pede linha e coluna, coloca em "HL"
         LD      C,3
         CALL    LOCATE        ;posiciona o cursor
         CALL    LIMPAR        ;e limpa uma linha a partir do mesmo
         CALL    REPETE        ;pergunta se repete a rotina, aguarda
                               resposta
         JR      C,PROG60      ;se afirmativo
         RET
```

```
PROG61:  LD      C,2
         CALL    TELA          ;mensagem na tela 2
         DB      0CH,"Qual a linha desejada? ",0
         CALL    LENUM         ;lê um número
         PUSH    AF            ;salva
         CALL    VIDEO         ;mensagem na tela 2
         DB      0CH,"Qual a coluna desejada? ",0
         CALL    LENUM         ;lê outro número
         LD      H,A           ;envia para "H" (coluna)
         POP     AF            ;recupera o número da linha
         ID      L,A           ;envia para "L"
         RET
PROG70:  LD      BC,0DF03H     ;contador de caracteres e número da tela
         CALL    SETAIX        ;ativa a tela 3
PROG71:  CALL    PROG72        ;envia um caracter
         DJNZ    PROG71        ;decrementa contador e vai enviar outro
                               ;caracter se não terminou
         CALL    REPETE        ;pergunta se repete a rotina, aguarda
                               ;resposta
         RET     NC            ;se não
         LD      C,3
         CALL    LIMPAT        ;limpa a tela 3
         CALL    TECLE         ;pede para teclar algo e aguarda uma
                               ;tecla
         JR      PROG70        ;vai repetir a rotina
PROG72:  PUSH    BC            ;salva contador
         LD      A,0FFH
         SUB     B             ;gera um byte em função do contador
                               ;(de 20H a FEH)
         PUSH    AF            ;salva
         CALL    PROG73        ;envia em hexadecimal
         LD      E,"-"
         CALL    BIOS6         ;envia o caracter "-"
         POP     AF            ;retorna o byte
         CALL    PROG75        ;e envia normalmente
         LD      E,20H
         CALL    BIOS6         ;envia um espaço em branco
         POP     BC            ;recupera o contador
         LD      A,(IX+1)      ;busca o número da coluna atual
         CP      33            ;verifica se é maior que 32
         RET     C             ;se não
         PUSH    BC            ;salva o contador
         LD      0,0AH
         CALL    BIOS6         ;e muda de linha
         LD      C,60
         CALL    TEMPO         ;espera 1 segundo
         POP     BC            ;recupera o contador
         RET
```

```
PROG73:  PUSH   AF          ;salva o byte
         RLCA               ;roda 4 vezes para passar os 4 bits mais
         RLCA               ;significativos para a posição de bits
                            ;menos
         RLCA               ;significativos
         RLCA
         CALL   PROG74      ;envia os 4 bits mais significativos
         POP    AF          ;recupera o byte para enviar os 4 bits
                            ;menos significativos
PROG74:  AND                :0FH$4separa os 4 bits menos significa-
                            tivos
         OR     30H         ;converte para ASCII
         CP     3AH         ;testa se era número
         JR     C,PROG75    ;se era
         ADD    A,7         ;converte para a letra correspondente
PROG75:  LD     E,A
         JP     BIOS6       ;envia para o vídeo
PROG80:  LD     C,2
         CALL   TELA        ;mensagem na tela 2
         DB     0CH,"Confirma (S/N)? ",0
         CALL   SIMNAO      ;aguarda a resposta
         RET    NC          ;se negativa
         JP     0           ;volta ao sistema operacional
TECLE:   LD     C,2
         CALL   TELA        ;mensagem na tela 2
         DB     0CH,"Tecle algo para continuar",0
         JP     GET         ;vai aguardar uma tecla
ENCHE:   LD     C,3
         LD     HL,0
         CALL   LOCATE      ;posiciona o cursor da tela 3 na posição
                            ;0,0
         LD     HL,569      ;tela toda menos uma posição (para evi-
                            ;tar que role a tela)
ENCHE0:  LD     E,"*"
         CALL   BIOS6       ;envia o caracter "*" para a tela
         DEC    HL          ;decrementa o contador
         LD     A,H
         OR     L           ;verifica se terminou
         JR     NZ,ENCHE0   ;se não
         CALL   SETMEM      ;envia o último endereço
         LD     A,"*"
         OUT    (98H),A     ;e envia o caracter "*" sem mudar o cur
                            ;sor
         EI                 ;habilita a interrupção
         RET
REPETE:  LD     C,2
         CALL   TELA        ;mensagem na tela 2
         DB     0CH,"Repete (S/N)? ",0
```

```
SIMNAO:  CALL   GET         ;aguarda uma tecla
         LD     E,A         ;salva
         RES    5,A         ;converte para maiúscula
         CP     "S"         ;testa se era "S"
         SCF                ;seta o flag "C"
         JR     Z,SIMNA     ;se era "S"
         CP     "N"         ;testa se era "N"
         JR     NZ,SIMNAO   ;se não era
SIMNA:   PUSH   AF          ;salva o flag "C"
         CALL   BIOS6       ;ecoa a tecla no vídeo
         LD     C,30
         CALL   TEMPO       ;espera 0,5 segundo
         POP    AF          ;retorna a condição do flag "C"
         RET
TEMPO:   XOR    A           ;zera a parte baixa de "TIME"
         LD     (0FC9EH),A
         EI                 ;habilita a interrupção a cada 1/60 do
                            ;segundo
TEMPO0:  LD     A,(0FC9EH)
         CP     C           ;testa se "TIME" igualou o valor dese-
                            ;jado
         JR     NZ,TEMPO0   ;se não
         RET
LENUM:   LD     D,3         ;máximo de 2 dígitos
         LD     HL,0        ;valor inicial
LENUM0:  CALL   SETMEM      ;envia endereço do cursor
         LD     A,0BH       ;caracter de cursor
         OUT    (98H),      ;envia para a tela
         EI                 ;habilita a interrupção
LENUM1:  CALL   GET         ;aguarda uma tecla
         LD     E,A         ;salva
         CP     8           ;testa se é volta um espaço
         JR     Z,LENUM2    ;se for
         CP     0DH         ;testa se é retorno de carro
         JR     Z,LENUM3    ;se for
         SUB    30H         ;converte para número
         CP     10          ;verifica se realmente era número
         JR     NC,LENUM1   ;se não era
         DEC    D           ;decrementa contador de dígitos
         JR     Z,LENUM3    ;se já tem 2 dígitos
         LD     H,L         ;transfere dígito já armazenado
         LD     L,A         ;armazena novo dígito
         CALL   BIOS6       ;envia o caracter para a tela, avança cur-
                            ;sor
         JR     LENUM0      ;vai ler a próxima tecla
LENUM2:  LD     A,D
         CP     3           ;testa se já foi armazenado algum dígito
         JR     Z,LENUM     ;se não
         INC    D           ;decrementa contador (contador reverso)
```

```
            CALL    SETMEM      ;envia endereço do cursor
            LD      A,20H
            OUT     (98H),A     ;limpa a posição
            EI                  ;habilita a interrupção
            CALL    BIOS6       ;volta uma posição do cursor
            LD      L,H         ;retorna a ser primeiro digito salvo
            LD      H,0         ;zera este digito
            JR      LENUM0      ;vai ler a próxima tecla
LENUM3:     LD      E,20H
            CALL    BIOS6       ;apaga o cursor
            LD      A,H         ;busca o primeiro digito
            ADD     A,A         ;multiplica por 2
            ADD     A,A         ;por 4
            ADD     A,H         ;por 5
            ADD     A,A         ;por 10
            ADD     A,L         ;soma o segundo digito
            RET
;string com endereços de rotinas que escrevem mensagens na tela
;2 em função da posição do cursor da tela1
VETOR:      DW      PROG1,PROG2,PROG3,PROG4
            DW      PROG5,PROG6,PROG7,PROG8
;mensagem inicial da tela 1
MENS:       DB      "Esta fase do programa demonstra como a",0
            DB      0AH, "tela rola, usando a tela 1 para isto.",0
            DB      0AH,0
            DB      0AH, "Na segunda fase do programa, use as",0
            DB      0AH, "teclas de seta para selecionar a ro-",0
            DB      0AH, "tina e a tecla <SELECT> para ativar a",0
            DB      0AH, "mesma. Dentro de uma rotina, responda",0
            DB      0AH, "às perguntas que aparecem na tela 2 e",0
            DB      0AH, "observe o resultado na tela 3.",0
            DB      0AH, "Tecle qualquer coisa para continuar.",0
            END
```

TELEFONE Este é um programa que irá permitir arquivar até 1000 nomes e telefones em um arquivo de nome "TELEFONE.DAT". Este arquivo cabe integralmente na memória, permitindo que a pesquisa seja toda em memória, ao contrário do dBASEII. Isto permite velocidade de acesso muito maior, mas não confunda: isto é um programa específico, o dBASEII permite que você crie programas conforme a sua necessidade. Neste programa, caso você entre com um novo telefone ou altere algum já existente, a alteração é só em memória. Só será gravado novo arquivo quando se chamar o fim de programa, quando então o arquivo antigo será transformado em ".BAK", apagando o ".BAK" ante-

rior. Assim, saia do programa sempre pela função "5", para salvar as alterações feitas. Caso deseje retornar o arquivo anterior, apague o arquivo atual e renomeie "TELEFONE.BAK" para "TELEFONE.DAT". No início do programa não haverá um arquivo "TELEFONE.DAT", mas não se preocupe, a memória será inicializada como se houvesse tal arquivo, e quando for sair do programa será criado um arquivo com os telefones digitados.

```
        ASEG                    ;necessário para o M80/L80
        ORG     100
        .Z80                    ;necessário para o M80/L80
FCB     EQU     5CH
PILHA   EQU     100H            ;de 80H a 100H
BASE    EQU     800H            ;área de carga da base de dados
TOTAL   EQU     BASE+20         ;dentro da base de dados, local onde
                                ;está armazenado o número total de tele-
                                ;fones do arquivo "TELEFONE.DAT"
        JR      INÍCIO          ;pula parâmetros internos
;parâmetros internos
MUDOU:  DB      0FFH            ;se 0 houve mudança no arquivo
TERM:   DS      2               ;espaço para armazenar a última posição
                                ;de memória do arquivo

FONTE:  DB      0,"TELEFONEDAT"
;usado para formação do FCB do arquivo de dados

INICIO: LD      SP,PILHA        ;carrega o endereço da pilha
        LD      HL,BASE
        LD      DE,BASE+1
        LD      BC,80FFH
        LD      (HL),C          ;carrega a área de memória do arquivo
                                ;de dados
        LDIR                    ;+256 bytes com 0FFH
        LD      HL,FONTE
        LD      DE,FCB
        LD      C,12
        LDIR                    ;carrega o nome do arquivo no FCB
        LD      HL,0            ;zera o número de telefones armazena-
                                ;dos
        LD      (TOTAL),HL
        CALL    CLRFCB          ;limpa restante do FCB para poder usar
                                ;"LOAD"
        LD      DE,BASE
        CALL    LOAD            ;tenta ler arquivo. Se não existe, perma-
                                ;nece o arquivo preparado na memória,
```

```
                                    ;se existe, grava por cima do arquivo
                                    ;preparado na memória.
        CALL    VIDEO       ;limpa a tela e apresenta os comandos
        DB      0CH,0AH,9,"1- Pesquisa telefone"
        DB      0DH,0AH,9,"2- Imprime arquivo"
        DB      0DH,0AH,9,"3- Entra novo telefone"
        DB      0DH,0AH,9,"4- Altera dados"
        DB      0DH,0AH,9,"5- Fim de programa",0DH,0AH,0
        LD      HL,10
        CALL    LOCATE      ;posiciona o cursor
LACO:   CALL    INKEY       ;lê o teclado
        SUB     31H         ;subtrai ASCII de "1"
        CP      4           ;testa se era "5"
        JR      Z,LACO3     ;se era
        JR      NC,LACO     ;se era maior que "5" ou não era nume-
                            ;ral
        OR      A
        JR      Z,LACO0     ;se era "1"
        CP      2
        JR      Z,LACO2     ;se era "3"
        JR      C,LACO1     ;se era "2"
        CALL    MUDA        ;era "4", chama rotina de alterar arquivo
        JR      LACO        ;volta a aguardar comando
LACO0:  CALL    PESQ        ;chama rotina de pesquisar o telefone
        JR      LACO        ;volta a aguardar comando
LACO1:  CALL    IMPRI       ;chama rotina de imprimir todo o
                            ;arquivo
        JR      LACO        ;volta a aguardar comando
LACO2:  CALL    ENTRA       ;chama rotina de entrada de novos tele-
                            ;fones
        JR      LACO        ;volta a aguardar comando
LACO3:  LD      A,(MUDOU)   ;fim de programa, testa se houve
                            ;mudança no arquivo
        OR      A
        JP      NZ,0        ;se não houve, volta ao sistema opera-
                            ;cional
        CALL    CLRFCB      ;houve mudança, limpa restante do FCB
                            ;para salvar o novo arquivo
        LD      HL,(TERM)   ;busca endereço da última posição de
                            ;arquivo
        LD      DE,BASE     ;início da área a ser gravada
        OR      A
        SBC     HL,DE       ;calcula tamanho do arquivo
        ADD     HL,HL       ;multiplica por 2, de maneira que "H"
                            ;contenha o número de blocos comple-
                            ;tos de 128 bytes do arquivo
        INC     H           ;incrementa número de blocos para
                            ;pegar bloco parcialmente cheio
        LD      B,H         ;"B" = número de registros a serem gra-
```

```
                                    ;vados
        CALL    UPDATE              ;grava o arquivo, transformando arquivo
                                    ;anterior em ".BAK"
        JP      0                   ;volta ao sistema operacional
PESQ:   CALL    QPES                ;pergunta o nome desejado, retorna com
                                    ;o par "DE" apontando para o início do
                                    ;registro da pessoa desejada
        RET     NC
        LD      HL,MENS3
        CALL    PRINT               ;apresenta na tela: "Nome:"
        CALL    PRNOME              ;prepara o nome para sair
        LD      HL,LINHAS
        CALL    PRINT               ;envia o nome para a tela
        LD      HL,MENS0
        CALL    PRINT               ;apresenta na tela "DDI/DDD:"
        PUSH    HL                  ;salva início da próxima mensagem
        LD      HL,20
        ADD     HL,DE
        EX      DE,HL               ;"DE" = início do número de DDI/DDD
        LD      B,3                 ;são 3 bytes que podem ser expandidos
                                    ;a até 6
        CALL    DCONV               ;expande e prepara para saída
        CALL    PRINT               ;sai no vídeo o DDI/DDD
        POP     HL
        CALL    PRINT               ;sai no vídeo: "Telefone:"
        PUSH    HL                  ;salva início da próxima mensagem
        LD      B,4                 ;são 4 bytes que podem ser expandidos
                                    ;a até 8
        CALL    DCONV               ;expande e prepara para saída
        CALL    PRINT               ;sai no vídeo o telefone
        POP     HL
        CALL    PRINT               ;sai no vídeo: "Ramal:"
        LD      B,2                 ;são 2 bytes que podem ser expandidos
                                    ;a até 4
        CALL    DCONV               ;expande e prepara para saída
        CALL    PRINT               ;sai no vídeo o ramal
        LD      HL,18
        CALL    LOCATE              ;posiciona o cursor
        CALL    VIDEO               ;apresenta a mensagem abaixo
        DB      "Pesquisa novo nome (S/N)? ",0
        CALL    SIMNAO              ;aguarda resposta, limpa a tela a partir
                                    ;da linha 10 após a resposta, e posicio-
                                    ;na o cursor no início da linha 10
        RET     NC                  ;se não
        JP      PESQ                ;se sim
IMPRI:  LD      DE,BASE+32          ;início da área de registros no arquivo
IMPRI0: LD      A,(DE)              ;busca o início de um nome
        INC     A                   ;testa se existe um nome armazenado
        RET     Z                   ;se não tem mais nomes
```

```
        LD      HL,32
        ADD     HL,DE           ;calcula início do próximo registro de
                                ;nome
        PUSH    HL              ;e salva
        LD      HL,MENS
        CALL    LPRINT          ;imprime: "Nome:"
        CALL    PRNOME          ;prepara o nome para sair
        LD      HL,LINHAS
        CALL    LPRINT          ;imprime o nome
        LD      HL,MENS0
        CALL    LPRINT          ;muda de linha e imprime: "DDI/DDD:"
        PUSH    HL              ;salva início da próxima mensagem
        LD      HL,20
        ADD     HL,DE
        EX      DE,HL           ;"DE" = início do DDI/DDD
        LD      B,3
        CALL    DCONV           ;expande o DDI/DDD
        CALL    LPRINT          ;imprime o DDI/DDD
        POP     HL              ;retorna início de mensagem
        CALL    LPRINT          ;muda de linha e imprime: "Telefone:"
        PUSH    HL              ;salva início da próxima mensagem
        LD      B,4
        CALL    DCONV           ;expande o telefone
        CALL    LPRINT          ;imprime o telefone
        POP     HL              ;retorna início da mensagem
        CALL    LPRINT          ;muda de linha e imprime: "Ramal:"
        LD      B,2
        CALL    DCONV           ;expande o ramal
        CALL    LPRINT          ;imprime o ramal
        LD      B,2
IMPRI1: LD      HL,MENSF
        CALL    LPRINT          ;muda de linha 2 vezes
        DJNZ    IMPRI1
        POP     DE              ;retorna início do próximo registro
        JR      IMPRI0          ;e vai imprimi-lo
CHEIO:  CALL    VIDEO           ;envia para o vídeo a mensagem abaixo
        DB      "Arquivo cheio ",0
        RET                     ;retorna mantendo a mensagem no vídeo
ENTRA:  LD      HL,10
        CALL    LOCATE          ;posiciona o cursor
        LD      HL,(TOTAL)      ;busca o número de telefones armazena-
                                ;dos
        LD      DE,1000         ;número máximo de telefones no arquivo
        OR      A
        SBC     HL,DE           ;testa se arquivo está cheio
        JR      Z,CHEIO         ;se estiver cheio
        CALL    VIDEO           ;apresenta a mensagem abaixo
        DB      "Digite o nome: ",0
```

```
        LD      A,20            ;20 caracteres no máximo
        CALL    LELIN           ;lê um nome
        JR      Z,ENTRA0        ;se não foi digitado nenhum nome
        CALL    FIND            ;tenta achar o nome digitado no arquivo
ENTRA2: JR      C,ENTRA1        ;se não achou, "DE" apontará para área
                                ;livre
        INC     A               ;testa se a concordância é parcial
        JR      Z,ENTRA3        ;se não for
        CALL    FINDP           ;procura o próximo nome
        JR      ENTRA2          ;e volta a testar
ENTRA3: LD      HL,20
        CALL    LOCATE          ;posiciona o cursor
        CALL    VIDEO           ;apresenta a mensagem abaixo
        DB      "Nome já existe",0
ENTRA0: LD      HL,21
        CALL    LOCATE          ;posiciona o cursor
        CALL    VIDEO           ;apresenta a mensagem abaixo
        DB      "Outra entrada (S/N)? ",0
        CALL    SIMNAO          ;aguarda resposta, limpa a tela a partir
                                ;da linha 10 após a resposta, e posiciona
                                ;o cursor no início da linha 10
        JR      C,ENTRA         ;se sim
        RET                     ;se não
ENTRA1: LD      B,0             ;"C" retorna da rotina "FIND" = "B"
        PUSH    HL              ;salva início do nome digitado
        LD      HL,20
        ADD     HL,DE           ;calcula início da área de armazenar
                                ;números
        EX      (SP),HL         ;troca na pilha, retornando o início do
                                ;nome
        LDIR                    ;armazena o nome
        XOR     A
        LD      (MUDOU),A       ;indica que houve mudança no arquivo
        LD      HL,MENDS0
        CALL    PRINT           ;muda de linha e apresenta: "DDI/DDD: "
        LD      B,7             ;máximo de 6 numerais
        CALL    LENUM           ;aguarda digitar o número
        POP     DE              ;retorna início da área de números
        LD      BC,3            ;forma 3 bytes comprimidos
        CALL    CONV            ;comprime e armazena
        PUSH    DE              ;salva área do próximo número
        LD      HL,MENS1
        CALL    PRINT           ;muda de linha e apresenta: "Telefone:"
        LD      B,9             ;máximo de 8 numerais
        CALL    LENUM           ;aguarda digitar o número
        POP     DE              ;retorna área de armazenamento do
                                ;número
        LD      BC,4            ;forma 4 bytes comprimidos
        CALL    CONV            ;comprime e armazena
```

```
        PUSH    DE          ;salva a área do próximo número
        LD      HL,MENS2
        CALL    PRINT       ;muda de linha e apresenta: "Ramal:"
        LD      B,5         ;máximo de 4 numerais
        CALL    LENUM       ;aguarda digitar o número
        POP     DE          ;retorna área de armazenamento do
                            número
        LD      BC,2        ;forma 2 bytes comprimidos
        CALL    CONV        ;comprime e armazena
        LD      (TERM),DE   ;armazena último endereço utilizado
        LD      HL,(TOTAL)
        INC     HL
        LD      (TOTAL),HL
        JR      ENTRA0      ;vai verificar se tem nova entrada
MUDA:   CALL    QPES        ;pergunta o nome desejado, retorna com
                            ;o par "DE" apontando para o início do
                            ;registro da pessoa desejada
        RET     NC          ;se não veio nenhum nome
        CALL    PRNOME      ;prepara o nome para sair
        LD      HL,MENS3
        CALL    PRINT       ;apresenta na tela: "Nome:"
        LD      HL,LINHAS
        CALL    PRINT       ;apresenta o nome
        LD      HL,MENS4
        CALL    PRINT       ;apresenta a mensagem: "muda para:"
        LD      HL,MENSF
        CALL    PRINT       ;muda de linha
        PUSH    DE          ;salva o início do registro
MUDA0:  LD      A,20        ;máximo de 20 caracteres
        CALL    LELIN       ;aguarda o novo nome
        JR      Z,MUDA0     ;se não foi digitado nada
        POP     DE          ;retorna o início do registro
        PUSH    HL          ;salva o início do nome
        LD      HL,20
        LD      A,19
        ADD     HL,DE       ;calcula o início da área dos números
        EX      (SP),HL     ;troca com a pilha, retornando início do
                            ;nome
        SUB     B           ;calcula 19-número de caracteres
        LD      C,B         ;"C" = número de caracteres
        LD      B,0
        LDIR                ;armazena o novo nome
        LD      C,A
        CP      0FFH        ;testa se foram 20 caracteres
        JR      Z,MUDA1     ;se foram
        LD      H,D         ;copia "DE" em "HL"
        LD      L,E
        LD      (HL),0FFH   ;pelo menos 1 byte deve ser levado a
                            ;0FFH
```

```
        OR      A               ;testa se foram 19 caracteres
        JR      Z,MUDA1         ;se foram
        INC     DE
        LDIR                    ;leva a 0FFH os caracteres que faltam
MUDA1:  LD      HL,MENSF
        CALL    PRINT           ;muda de linha
        LD      HL,MENS0        ;mensagem: "DDI/DDD:"
        LD      B,3             ;formar 3 bytes
        POP     DE              ;retorna endereço da área dos números
        CALL    MUDA2           ;apresenta a mensagem, apresenta o
                                ;número, pergunta para que número
                                ;deve mudar, lê o novo número e arma
                                ;zena
        LD      HL,MENS1        ;mensagem: "Telefone:"
        LD      B,4             ;formar 4 bytes
        CALL    MUDA2           ;apresenta a mensagem, apresenta o
                                número, pergunta para que número deve
                                mudar, lê o novo número e armazena
        LD      HL,MENS2        ;mensagem: "Ramal:"
        LD      B,2             ;formar 2 bytes
        CALL    MUDA2           ;apresenta a mensagem, apresenta o
                                ;número, pergunta para que número
                                ;deve mudar, lê o novo número e arma-
                                ;zena
        LD      HL,20
        CALL    LOCATE          ;posiciona o cursor
        CALL    VIDEO           ;apresenta a mensagem abaixo, note que
                                ;87H é o ASCII de ç, micros CP/M podem
                                ;não ter este caracter na sua formação
                                ;de video
        DB      "Outra mudança ",87H,"a(S/N)? ",0
        CALL    SIMNAO          ;aguarda resposta, limpa a tela a partir
                                ;da linha 10 após a resposta, e posiciona
                                ;o cursor no início da linha 10
        JP      C,MUDA
        LD      HL,MENSF
        LD      B,2
        CALL    FIND            ;busca a primeira área livre, pois nunca
                                ;um registro terá o caracter 0DH
        DEC     DE              ;decrementa para apontar dentro do
                                ;registro anterior
        LD      (TERM),DE       ;e salva como último endereço válido
        XOR     A
        LD      (MUDOU),A       ;indica que houve mudança no arquivo
        RET
MUDA2:  CALL    PRINT           ;imprime a mensagem
        PUSH    DE              ;salva área do número
        PUSH    BC              ;e o número de bytes que o compõe
        CALL    DCONV           ;expande o número
```

```
        CALL    PRINT       ;apresenta o número na tela
        LD      HL,MENS4
        CALL    PRINT       ;apresenta a mensagem: "muda para:"
        POP     BC          ;retorna o número de bytes
        PUSH    BC          ;e salva novamente
        LD      A,B
        ADD     A,A         ;multiplica por 2 para obter o número
                            ; máximo de numerais a serem digitados
        INC     A           ;incrementa (exigência de "LENUM")
        LD      B,A         ;número máximo de numerais
        CALL    LENUM       ;aguarda digitar o número
        POP     BC          ;retorna o número de bytes
        POP     DE          ;e a área de armazenamento
        LD      C,B         ;"C" = número de bytes
        LD      B,0
;programa de compactação e armazenamento de números. "HL"
;aponta para os números digitados, "DE" para a área de armazena-
;mento, e "BC" contém o número de bytes a serem formados.

CONV:   LD      A,(HL)      ;busca um número
        INC     HL          ;incrementa ponteiro
        RLD                 ;no byte apontado por "HL" os 4 bits
                            ;mais significativos são perdidos, os 4
                            ;menos significativos passam para a
                            ;posição de 4 mais significativos, e os 4
                            ;bits menos significativos do acumu-
                            ;lador passam a ser os 4 bits menos
                            ;significativos do byte apontado p/"HL"
        LDI                 ;copia o byte apontado por "HL" na posi-
                            ;ção apontada por "DE", decremen-
                            ;ta "BC", incrementa "HL" e "DE"
        LD      A,C
        OR      A           ;testa se terminou os bytes
        JR      NZ,CONV     ;se não
        RET

DCONV:  EX      DE,HL       ;"HL" aponta para o número a ser
                            ;expandido
        LD      DE,LINHAS   ;área de montagem do número
DCONV0: LD      C,(HL)      ;salva o byte compacto
        CALL    DCONV1      ;busca e armazena em ASCII um número
        CALL    DCONV1      ;busca e armazena em ASCII um número
        LD      (HL),C      ;recupera o byte compactado
        INC     HL          ;incrementa ponteiro
        DJNZ    DCONV0      ;decrementa contador e pula se não ter-
                            ;minou
        EX      DE,HL       ;"DE" = posição do primeiro byte após o
                            ;número convertido, Ó(HL" = fim da string
        LD      (HL),B      ;coloca um 0 para indicar fim de string
```

```
        LD      HL,LINHAS   ;início da string com o número expan-
                            ;dido
        RET
DCONV1: LD      A,30H
        RRD                 ;os 4 bits menos significativos do
                            ;byte apontado por "HL" passam a ser
                            ;os 4 bits menos significativos do acu-
                            ;mulador, os 4 bits mais significativos
                            ;do byte passam para a posição de 4
                            ;bits menos significativos
        CP      3FH         ;testa se os 4 bits eram "F", significando
                            ;ausência de número
        RET     Z           ;se era
        LD      (DE),A      ;armazena
        INC     DE          ;incrementa ponteiro
        RET

LINHAS: DS      21          ;buffer de nome/número para saída
PRNOME: DI                  ;bloqueia a interrupção
        PUSH    DE          ;salva início do nome
        EXX                 ;salva os pares "BC", "DE", e "HL"
        POP     HL          ;retorna início do nome
        LD      DE,LINHAS   ;início do buffer de saída
        LD      A,0FFH      ;byte indicativo de fim de arquivo
        LD      BC,1440H    ;"B" = 20, "C" > 20
PRNOM0: CP      (HL)        ;testa se terminou
        JR      Z,PRNOM1    ;se terminou
        LDI                 ;transfere, incrementa "DE" e "HL",
                            ;decrementa "BC" (na verdade só "C"
                            ;pois é > 14)
        DJNZ    PRNOM0      ;decrementa "B", pulando se não zerou
PRNOM1: XOR     A
        LD      (DE),A      ;indica fim de string
        EXX                 ;recupera os pares "BC", "DE" e "HL"
        EI                  ;libera a interrupção
        RET

FINDP:  PUSH    HL          ;salva início do nome para comparação
        LD      HL,32
        ADD     HL,DE       ;calcula início do próximo registro
        EX      DE,HL
        POP     HL          ;retorna início do nome para compara-
                            ;ção
        JR      FIND0       ;e vai comparar
FIND:   LD      DE,BASE     ;primeiro ponto de comparação, início de
                            ;arquivo
FIND0:  PUSH    HL          ;salva início do nome para comparação
        PUSH    HL          ;salva novamente
        LD      HL,32
```

```
        ADD     HL,DE       ;calcula início do próximo registro
        EX      (SP),HL     ;salva início do próximo registro, retorna
                            ;o início do nome para comparação
        LD      C,B         ;salva número de bytes do nome para
                            ;comparação
FIND1:  LD      A,(DE)      ;busca um caracter
        CP      (HL)        ;compara com o nome desejado
        JR      NZ,FIND2    ;se for diferente
        INC     DE          ;incrementa ponteiro do registro
        INC     HL          ;incrementa ponteiro do nome
        DJNZ    FIND1       ;decrementa contador de bytes, pula se
                            ;não terminou o nome pesquisado
        LD      A,(DE)      ;busca o byte seguinte ao trecho que
                            ;igualou o nome desejado
        LD      B,C         ;retorna o número de bytes
        POP     HL          ;retorna início do próximo registro
        LD      DE,32
        OR      A
        SBC     HL,DE       ;calcula início do registro que igualou
        EX      DE,HL       ;coloca em "DE"
        POP     HL          ;retorna início do nome
        CP      0FFH        ;testa se o byte seguinte ao trecho que
                            ;igualou é 0FFH, indicando concordância
                            total
        RET     Z           ;retorna se for
        LD      A,20
        CP      B           ;testa se eram 20 bytes
        RET     NZ          ;retorna se não eram, concordância par-
                            ;cial
        LD      A,0FFH      ;aqui se eram 20 bytes, indique concor
                            ;dância total (o byte poderia ser = 0FFH
                            ;se houvesse DDI/DDD).
        RET
FIND2:  POP     DE          ;retorna início do próximo registro
        POP     HL          ;e o início do nome pesquisado
        LD      B,C         ;retorna número de bytes
        LD      A,(DE)      ;busca o primeiro byte do novo registro
        INC     A           ;testa se é 0FFH, indicando fim de
                            ;arquivo
        JR      NZ,FIND0    ;se não era
        SCF                 ;seta CARRY indicando fim de arquivo
        RET
LENUM:  LD      HL,LINHA    ;buffer de entrada
        LD      C,0         ;número de caracteres já digitados
LENUM0  PUSH    BC          ;salva contadores
        PUSH    HL          ;salva ponteiro do buffer de entrada
        CALL    INKEY       ;lê o teclado
        POP     HL          ;retorna ponteiro
        POP     BC          ;e contadores
```

```
            LD      E,A         ;salva a tecla lida
            CP      8           ;testa se era para voltar um espaço
            JR      Z,LENUM2    ;se era
            CP      0DH         ;testa se era retorno de carro
            JR      Z,LENUM1    ;se era
            SUB     30H         ;tenta converter para número
            CP      10          ;testa se era numeral
            JR      NC,LENUM0   ;se não era
            LD      (HL),A      ;salva número
            LD      A,B         ;busca contador de bytes que faltam ser
                                ;lidos
            DEC     A
            JR      Z,LENUM1    ;se este era o último byte, então era
                                ;inválido
            CALL    LENUM4      ;envia a tecla para a tela
            INC     C           ;incrementa contador de numerais
            INC     HL          ;incrementa ponteiro do buffer
            DJNZ    LENUM0      ;decrementa contador de bytes que fal-
                                ;tam e pula
LENUM1:     LD      BC,7
            LD      D,H
            LD      E,L
            INC     DE
            LD      (HL),0FH
            LDIR                ;coloca mais 8 bytes 0FH para fins de
                                ;compactação, indicando ausência de
                                ;numeral
            LD      HL,LINHA    ;retorna, apontando para o início do buf-
                                ;fer
            RET
LENUM2:     LC      A,C         ;busca o contador de numerais já digita-
                                ;dos
            OR      A           ;testa se já foi digitado algum numeral
            JR      Z,LENUM0    ;se não
            DEC     C           ;decrementa contador de numerais já
                                ;digitados
            INC     B           ;incrementa contador dos que faltam
                                ;digitar
            DEC     HL          ;decrementa ponteiro do buffer
            CALL    LENUM3      ;apaga o último número da tela e volta
                                ;um espaço
            JR      LENUM0      ;continua leitura
LENUM3:     CALL    LENUM4      ;envia retorno de caracter que estava em
                                ;"E"
            LD      E,20H
            CALL    LENUM4      ;envia espaço, limpa o número da tela
            LD      E,8         ;e retorna um caracter novamente
LENUM4:     PUSH    HL          ;salva o ponteiro
            PUSH    BC          ;e os contadores
```

```
        CALL    BIOS6       ;envia o caracter em "E" para a tela
        POP     BC          ;retorna contadores
        POP     HL          ;e ponteiro
        RET
LELIN:  LD      DE,LINHA    ;buffer de entrada
        LD      (DE),A      ;número máximo de caracteres a serem
                            ;lidos
        LD      C,0AH
        CALL    5           ;lê uma linha de teclado
        LD      HL,LINHA+1  ;endereço onde está o número de carac-
                            ;teres
        LD      A,(HL)      ;busca o número de caracteres digitados
        INC     HL          ;aponta para o primeiro caracter
        OR      A           ;seta flag "Z" se nada foi digitado
        LD      B,A         ;número de caracteres digitados
        RET
LINHA:  DS      22
VIDEO:  EX      (SP),HL     ;salva "HL" na pilha, "HL" aponta para o
                            ;começo do texto
        CALL    PRINT       ;envia texto para o video
        EX      (SP),HL     ;recupera "HL", manda para pilha posi-
                            ;ção depois do byte 00
        RET                 ;que será o endereço de retorno
LPRINT: PUSH    BC          ;salva o par "BC"
        LD      C,5         ;indica saida na impressora
        JR      PRINT0      ;pula para a parte comum do programa
PRINT:  PUSH    BC          ;salva o par "BC"
        LD      C,6         ;indica saida no video
PRINT0: PUSH    DE          ;salva o par "DE"
PRINT1: LD      A,(HL)      ;busca um caracter
        INC     HL          ;incrementa o ponteiro
        OR      A           ;testa se é o byte 00
        JR      Z,PRINT2    ;se for, então é fim de texto
        LD      E,A         ;não é, portanto, vamos enviar o caracter
        PUSH    BC          ;salva indicador de video/impressora
        PUSH    HL          ;salva ponteiro
        CALL    5           ;envia o caracter
        POP     HL          ;retorna o ponteiro
        POP     BC          ;e o indicador de video/impressora
        JR      PRINT1      ;vá buscar o próximo caracter
PRINT2: POP     DE          ;recupera os pares "DE" e "BC"
        POP     BC
        RET
QPES:   LD      HL,10
        CALL    LOCATE      ;posiciona o cursor
        CALL    VIDEO       ;envia a mensagem abaixo
        DB      "Digite o nome da pessoa desejada",0DH,0AH,0
        LD      A,20
        CALL    LELIN       ;lê até 20 caracteres
```

```
            CALL    FIND        ;pesquisa o nome no arquivo
            JR      NC,QPES1    ;se houve concordância de nome
QPES0:      LD      HL,15
            CALL    LOCATE      ;posiciona cursor
            CALL    VIDEO       ;envia a mensagem abaixo
            DB      "Inexistente, tenta novo nome (S/N)? ",0
            CALL    SIMNAO      ;e aguarda resposta
            JR      C,QPES      ;se a resposta é afirmativa
            RET
QPES1:      CALL    CONFIR      ;mostra nome achado e pede confirma-
                                ;ção
            RET     C           ;se houve confirmação
            CALL    FINDP       ;procura a próxima concordância
            JR      NC,QPES1    ;se achou nova concordância
            JR      QPES0       ;vai informar não haver mais concordân-
                                ;cias
CONFIR:     PUSH    HL          ;salva início do nome pesquisado
            PUSH    DE          ;salva início do registro
            PUSH    BC          ;salva tamanho do nome pesquisado
            CALL    PRNOME      ;prepara o nome para saída
            LD      HL,15
            CALL    LOCATE      ;posiciona o cursor
            LD      HL,MENS3
            CALL    PRINT       ;envia para a tela: "Nome:"
            LD      HL,LINHAS
            CALL    PRINT       ;envia o nome para a tela
            LD      HL,18
            CALL    LOCATE      ;posiciona o cursor
            CALL    VIDEO       ;envia a mensagem abaixo
            DB      "É esta a pessoa desejada (S/N)? ",0
            CALL    SIMNAO      ;aguarda resposta, limpa a tela a partir
                                ;da linha 10, após a resposta, e posi-
                                ;ciona o cursor no início da linha 10
            POP     BC          ;retorna tamanho do nome
            POP     DE          ;retorna início do registro
            POP     HL          ;retorna início do nome pesquisado
            RET
SIMNAO:     CALL    INKEY       ;lê o teclado
            LD      E,A         ;salva a tecla lida
            RES     5,A         ;converte para maiúscula
            CP      "S"         ;testa se era sim
            SCF                 ;seta CARRY para caso seja "sim"
            JR      Z,SIMNAO    ;se era
            CP      "N"         ;testa se era não
            JR      NZ,SIMNAO   ;se não era
SIMNAO:     PUSH    AF          ;salva o CARRY, com indicação de S/N
            CALL    BIOS6       ;envia a tecla para a tela
            LD      HL,10
            CALL    LOCATE      ;posiciona o cursor
```

```
        LD      B,12        ;12 linhas para limpar
SIMNA1: CALL    VIDEO       ;envia linha em branco para limpar a
                            ;linha são 39 espaços em branco
        DB      "                                       ",0DH,0AH,0
        DJNZ    SIMNA1      ;decrementa contador e pula se não ter-
                            ;minou
        LD      HL,10
        CALL    LOCATE      ;reposiciona o cursor
        POP     AF          ;retorna o CARRY
        RET
LOCATE: LD      DE,2020H    ;offset exigido pelo padrão VT52
        ADD     HL,DE
        LD      (LOCAL),HL  ;armazena para enviar
        CALL    VIDEO       ;envia comando de posicionamento
                            ;padrão VT52
        DB      1BH,"Y"     ;bytes iniciais do comando
LOCAL:  DB      0,0,0       ;posição e byte final 00 para a rotina
                            ;VIDEO
        RET
INKEY:  LD      A,255
PUT:    LD      E,A
BIOS6:  LD      C,6         ;BIOS6 envia o byte em "E" para o vídeo
                            ;se for diferente de 255, pois se for 255
                            ;então lê o teclado, retornando a tecla
                            ;lida no acumulador ou 0, se nenhuma
                            ;tecla for pressionada
BIOS:   JP      5
SETDMA: LD      C,1AH
        JR      BIOS
ABRE:   LD      C,15
DISCO:  LD      DE,FCB      ;FCB deve ser definido em outra parte do
                            ;programa, normalmente pode-se usar o
                            ;endereço 5CH
        JR      BIOS
CRIA:   LD      C,22
        JR      DISCO
FECHA:  LD      C,16
        JR      DISCO
APAGA:  LD      C,19
        JR      DISCO
RENOME: LD      C,23
        JR      DISCO
LE:     LD      C,20
        JR      DISCO
GRAVA:  LD      C,21
        JR      DISCO
CLRFCB: LD      HL,FCB+12   ;início da área a ser limpa
```

```
        LD      DE,FCB+13       ;byte seguinte
        LD      BC,23           ;número de bytes a limpar
        LD      (HL),B          ;envia "0"
        LDIR                    ;limpa restante do FCB
        RET
LOAD:   PUSH    DE              ;salva início da posição de memória para
                                ;a leitura do arquivo
        CALL    ABRE            ;tenta abrir o arquivo
        POP     DE              ;recupera o início da memória
        INC     A               ;testa se conseguiu abrir ("A" = 0FFH se
                                ;não)
        RET     Z               ;se não
LOAD0:  LD      HL,128          ;tamanho de um registro a ser lido
        ADD     HL,DE           ;calcula a próxima posição de memória
        PUSH    HL              ;e salva
        CALL    SETDMA          ;envia posição de memória atual para o
                                ;sistema operacional
        CALL    LE              ;lê um registro
        POP     DE              ;retorna endereço do próximo registro
        OR      A               ;testa se leu sem erro o último registro
        JR      Z,LOAD0         ;se leu sem erro, vá ler o próximo
        RET                     ;se errou então é fim de arquivo
UPDATE: PUSH    DE              ;salva início da memória
        PUSH    BC              ;salva o tamanho a ser gravado
        LD      HL,FCB+9        ;salva a extensão
        LD      DE,EXSV
        LD      BC,3
        LDIR
        EX      DE,HL           ;e muda a mesma para "$$$" (para caso
                                ;haja alguma falha não comprometer
                                ;arquivos OK)
        LD      DE,FCB+9
        LD      C,3
        LDIR
        POP     BC              ;recupera o tamanho da gravação
        POP     DE              ;restaura início da memória
        CALL    SAVE            ;grava o arquivo com a extensão "$$$"
        LD      HL,EXBAK        ;muda a extensão no FCB para "BAK"
        LD      DE,FCB+9
        LD      BC,3
        LDIR
        CALL    CLRFCB          ;esta rotina está dentro do MKFCB e
                                ;limpa o restante do FCB
        CALL    APAGA           ;apaga o arquivo com extensão "BAK"
                                ;anterior
        CALL    CLRFCB          ;limpa o resto do FCB
        LD      HL,FCB          ;prepara para mudar o nome de um
                                ;arquivo para
        LD      DE,FCB+16       ;"BAK"
```

```
        LD      BC,16
        LDIR
        LD      HL,EXSV         ;e este arquivo é o que tem a extensão
                                ;que
        LD      DE,FCB+9        ;foi salva
        LD      C,3
        LDIR
        CALL    RENOME          ;muda o nome
        CALL    CLRFCB          ;limpa restante do FCB
        LD      HL,FCB          ;prepara para mudar o nome de um
                                ;arquivo para a extensão que foi salva
        LD      DE,FCB+16
        LD      BC,16
        LDIR
        LD      HL,EXDOL        ;e este arquivo é o recém-gravado com a
                                ;extensão "$$$"
        LD      DE,FCB+9
        LD      C,3
        LDIR
        JP      RENOME          ;muda o nome e termina a rotina
EXSV:   DS      3               ;espaço para salvar a extensão
EXDOL:  DB      "$$$"           ;extensão de rascunho
EXBAK:  DB      "BAK"           ;extensão de back-up
SAVE:   PUSH    DE              ;salva início da memória e tamanho
        PUSH    BC
        CALL    CRIA            ;cria o arquivo, apaga anterior de mesmo
                                ;nome
        POP     BC              ;restaura tamanho
SAVE0:  POP     DE              ;restaura posição de memória do
                                ;próximo registro a ser gravado
        LD      HL,128          ;tamanho de um registro
        ADD     HL,DE           ;calcula endereço do próximo registro
        PUSH    HL              ;e salva
        PUSH    BC              ;salva número de registros que faltam
                                ;gravar
        CALL    SETDMA          ;envia endereço para o sistema operacio-
                                ;nal
        CALL    GRAVA           ;e grava o registro
        POP     BC              ;recupera o número de registros que fal
                                ;tam gravar
        DJNZ    SAVE0           ;decrementa, e se não terminou vai gra-
                                ;var o próximo registro
        POP     DE              ;acerta a pilha
        JP      FECHA           ;fecha o arquivo (obrigatório numa grava-
                                ;ção)

;MENS0: DB      0DH,0AH,"DDI/DDD: ",0
;MENS1: DB      0DH,0AH,"Telefone: ",0
;MENS2: DB      0DH,0AH,"Ramal: ",0
```

```
;MENS3:   DB      "Nome" ",0
;MENS4:   DB      "Muda para: ",0
;MENSF:   DB      0DH,0AH,0
         END
```

Apêndices

Comparação entre as Chamadas de Bios do CP/M 2.2 e do MSX.

A chamada ao bios é feita por uma instrução "CALL BIOS", em que o endereço de BIOS é 0005H. No registrador "C" coloca-se o número da função desejada, os registradores "D" e "E" recebem parâmetros de entrada, o registrador "A" serve como saída, e o par "HL" pode funcionar como entrada ou saída. No caso do modo carregado pelo BASIC, a única coisa que muda é o endereço do BIOS, que passa a ser 0F37DH.

Passamos agora à comparação dos dois sistemas operacionais:

Função 00 - Reinicio de sistema. Saída de programa. Também se pode usar "JP 0000H". No caso do modo chamado pelo BASIC, esta rotina é ainda mais drástica: resceta o computador. É idêntica no CP/M e no MSX-DOS.

Função 01 - Leitura de teclado. Aguarda uma tecla ser digitada e ecoa a mesma no vídeo. A tecla lida retorna no acumulador. As teclas especiais como CONTROL-P, CONTROL-I, etc são reconhecidas e executadas, o que, às vezes, pode ser um problema. É idêntica no CP/M e no MSX-DOS.

Função 02 - Saída no vídeo. Ecoa no vídeo o caracter presente no registrador "E". Teclas especiais são reconhecidas e executadas. É idêntica no CP/M e no MSX-DOS.

Função 03 - Entrada auxiliar. Aguarda um caracter pela linha auxiliar que no caso do CP/M é a leitora de fita de papel perfurado, e

no caso do MSX é a porta serial. No caso do MSX a leitura do teclado continua e o ato de teclar CONTROL-S aborta esta função. O caracter lido é retornado no registrador "A"

Função 04 - Saída auxiliar. Envia para a saída auxiliar o caracter presente no registrador "E". A saída auxiliar no caso do CP/M é a perfuradora de fita de papel, e no caso do MSX é a porta serial. No MSX teclar CONTROL-S aborta a função

Função 05 - Saída para impressora. O caracter presente no registrador "E" é enviado para a impressora. No caso do MSX teclar CONTROL-S, aborta a função.

Função 06 - I/O direto do teclado. Se o registrador "E" contiver algo diferente de OFFH, este caracter é enviado para o vídeo. Se for OFFH, o teclado é lido, e a tecla pressionada é retornada no acumulador. Se nada foi teclado, retorna 00H. Não ecoa nem executa (quando usada para leitura de teclado) as teclas especiais. É idêntica no CP/M e no MSX-DOS.

Função 07 - No CP/M coloca no acumulador o byte presente no endereço 0003H. No MSX esta função aguarda uma tecla ser pressionada, e retorna seu código no acumulador. Não ecoa nem reconhece as teclas especiais.

Função 08 - No CP/M coloca o byte presente no registrador "E" na posição de memória 0003H. No MSX é idêntica à função 01, exceto que não ecoa a tecla no vídeo.

Função 09 - Saída de texto para o vídeo. O texto tem o seu primeiro byte apontado pelo par "DE", e é terminado pelo byte 24H (ASCII de <$>). É idêntica no CP/M e no MSX-DOS.

Função 0A - Leitura de um buffer de teclado. O par "DE" deve apontar para a primeira posição de memória a funcionar como buffer. Este primeiro byte deve conter o número máximo de caracteres a serem lidos, o segundo byte recebe o número de bytes efetivamente lidos, e os restantes recebem o texto digitado. É idêntica no CP/M e no MSX-DOS.

Função 0B - Checa estatus do teclado. No CP/M retorna no acumulador 00H se não há caracteres, e 0FFH se há. No MSX executa o mesmo que no CP/M, só que reconhece as teclas especiais CONTROL-S, CONTROL-P, e CONTROL-C tomando a decisão apropriada.

Função 0C - número da versão. Retorna o número da versão de CP/M. No caso do MSX retorna 22 para indicar que é compatível com CP/M 2.2.

Função 0D - Reseta o sistema de discos, limpa todos os arquivos. Portanto, feche todo e qualquer arquivo aberto antes de chamar esta função. Deve ser usada toda vez que um disco é trocado, especialmente em micros CP/M que bloqueiam gravação após a abertura do drive. É idêntica no CP/M e no MSX-DOS.

Função 0E - Seleciona o disco a ser usado como default nas operações de disco subseqüentes. 01 representa o disco "A", 02 o disco "B", e assim por diante. É idêntica no CP/M e no MSX-DOS.

Função 0F - Abre um arquivo. O par "DE" deve apontar para o primeiro byte do FCB. Se o arquivo não existe, o acumulador retorna o byte 0FFH. Se existe, o retorno pode ser 00H, 01H, 02H, 03H ou 04H. É idêntica no CP/M e no MSX-DOS.

Função 10 - Fecha um arquivo. O par "DE" deve apontar o FCB. Deve ser utilizada sempre após o término da gravação de um arquivo. É idêntica no CP/M e no MSX-DOS.

Função 11 - Procura o primeiro arquivo do diretório cujo nome combina com o presente no FCB apontado pelo par "DE". Neste caso, o nome no FCB pode ter o caracter coringa "?". Assim, vários arquivos podem ser apontados pelo mesmo nome. O acumulador retorna 0FFH se não existe nenhum arquivo cujo nome coincida com o presente no FCB. O acumulador retorna 00H se houve a coincidência e o FCB do arquivo que foi achado é colocado da posição 33 (128 para o caso de CP/M) do endereço de DMA.

Função 12 - Procura o próximo arquivo coincidente. Entradas/saídas idênticas à função 11.

Função 13 - Deleta os arquivos cujos nomes coincidam com o do FCB que pode ter caracter coringa. O FCB deve ser apontado pelo par "DE". É idêntica no CP/M e no MSX-DOS.

Função 14 - Leitura seqüencial. Lê o próximo registro para o endereço de DMA. O arquivo é aquele apontado pelo FCB, que por sua vez é apontado pelo par "DE". O arquivo já deve estar aberto. O acumulador retorna 00H se a leitura foi bem sucedida, 01H se achou o fim de arquivo, e 02H se houve erro de leitura. É idêntica no CP/M e no MSX-DOS

Função 15 - Escrita seqüencial. Grava um registro com os 128 bytes presentes no endereço do DMA. O par "DE" deve apontar para o FCB do arquivo onde está sendo feita a gravação, e este arquivo deve ter sido criado pela função 16. O acumulador retorna 00H se a gravação foi bem sucedida, 01H se não há mais espaço, ou outro valor se houve algum problema de gravação. É idêntica no CP/M e no MSX-DOS.

Função 16 - Cria um arquivo. O par "DE" deve apontar para o FCB do arquivo a ser criado. Se não houver espaço no diretório, o acumulador retorna 0FFH, se houver espaço retorna 00H. É idêntica no CP/M e no MSX-DOS.

Função 17 - Renomeia arquivo. O par "DE" deve apontar o FCB. Troca o nome do arquivo apontado pelos primeiros 16 bytes do FCB para o nome presente nos 16 bytes seguintes. É idêntica no CP/M e no MSX-DOS.

Função 18 - Vetor de conexão. Retorna no par "HL" os drives presentes. O bit 0 de "L" representa o drive "A", o bit 1 o drive "B", e assim por diante até o bit 7 de "H". Drive presente faz o bit correspondente ir a 1, drive ausente leva o bit a 0. É idêntica no CP/M e no MSX-DOS.

Função 19 - Drive atual. Retorna no acumulador o drive ativo por default. (0 = "A", 1 = "B", 2 = "C" etc.) É idêntica no CP/M e no MSX-DOS.

Função 1A - Endereço de DMA. Informa ao sistema operacional que o endereço apontado pelo par "DE" será o próximo endereço de DMA. É idêntica no CP/M e no MSX-DOS.

Função 1B - Tabela de alocação. No MSX-DOS quando se chama esta função, o registrador "E" deverá conter o número do drive (1 = "A", 2 = "B", etc.). No retorno, o acumulador contém o número de setores por aglomerado, ou o valor 0FFH se o drive for inválido. O par "BC" retorna o tamanho de um setor (512); o par "DE" retorna o número de aglomerados do disco; o par "HL" retorna o número de aglomerados disponíveis no disco; e o registrador "IY" aponta para o endereço de memória aonde está armazenada a FAT do disco. No CP/M retorna no par "HL" o endereço inicial do vetor de alocação (tabela mantida na memória que indica a porção do disco ativo que está sendo usada.

Função 1C - Proteção contra gravação. No CP/M torna o disco protegido contra gravação. No MSX-DOS não faz nada, pois esta função não está implementada.

Função 1D - Vetor R/O. No CP/M esta função retorna no par "HL" os drives que estão protegidos contra gravação. No MSX-DOS não faz nada pois a proteção contra escrita não está implementada.

Função 1E - Atributos do arquivo. No CP/M busca os atributos do arquivo cujo FCB está apontado pelo par "DE" (sistema só leitura, etc.). No MSX-DOS não faz nada, não está implementado.

Função 1F - Busca parâmetros do disco. No CP/M retorna no par "HL" o endereço de memória onde estão os parâmetros do disco, tais como o tamanho do setor, números de setores do disco, etc. Estas informações são para o drive ativo por default, e podem ser usadas entre outros para determinar o espaço ainda disponível no disco. No MSX-DOS não faz nada, não está implementado.

Função 20 - Código de usuário. No MSX-DOS não faz nada, não está implementado. No CP/M, se o registrador "E" contiver o byte 0FFH então o acumulador retornará o usuário corrente. Caso contrário, se o valor presente no registrador "E" for um usuário válido (0 a 31), este será o novo usuário.

Função 21 - Leitura aleatória. Lê um registro indicado pelos bytes 33, 34 e 35 do FCB indicado pelo par "DE" para o endereço de DMA. Se a leitura foi bem sucedida o acumulador retorna o valor 0. Se achou o fim do arquivo, retorna 1. Se houve algum erro, retorna 2. É idêntica no CP/M e no MSX-DOS.

Função 22 - Escrita aleatória. Escreve no registro indicado da mesma maneira que na função acima a área de DMA. Se a escrita foi bem sucedida, o acumulador retorna o valor 0. Se não houver espaço retorna o valor 1. É idêntica no CP/M e no MSX-DOS.

Função 23 - Tamanho de arquivo. O par "DE" aponta para o FCB de um arquivo ainda não aberto. Se o arquivo não existe, o acumulador retorna o byte 0FFH. Se existe, retorna o byte 00H e coloca nos bytes 33, 34 e 35 do FCB o número de registros do arquivo. É idêntica no CP/M e no MSX-DOS.

Função 24 - Lê o próximo registro aleatório. O par "DE" aponta o FCB. Altera o campo do número do registro incrementando-o, e lê o registro assim apontado. É idêntica no CP/M e no MSX-DOS.

Função 25 - Reset do drive. No CP/M deve ser usada após a troca de discos para reabilitar a gravação, ou após permanecer com o drive desligado por algum tempo no meio de uma leitura ou gravação (quando a operação não é feita de uma só vez, por se tratar de arquivo grande demais). No MSX-DOS não é necessária e não está implementada.

Função 26 - Escreve zeros no registro aleatório. Funciona no CP/M da mesma maneira que a função 22, só que o registro é preenchido com zeros. No MSX-DOS, todavia, é a função 28 que faz esta tarefa. A função 26 escreve um bloco de registros aleatórios. O par "DE" aponta para o FCB onde o número do registro inicial já está setado, e o par "HL" contém o número de registros a serem escritos. Se não houver espaço, o acumulador retorna 1, mas o tamanho do arquivo é setado como se a escrita tivesse sido bem sucedida. Se a escrita foi bem sucedida, o acumulador retorna 0.

As próximas funções só existem no MSX-DOS.

Função 27 - Leitura de bloco aleatório. Lê um bloco de registros cujo registro inicial está no campo de registro aleatório do FCB. O número de registros do bloco vem no par "HL", e o FCB está apontado pelo par "DE". Se a leitura for bem sucedida o acumulador retorna 0, se o fim de arquivo foi achado retorna 1, e se houve algum erro retorna o valor 2.

Função 29 - Não está implementada.

Função 2A - Data. Esta função retorna a data armazenada na memória. O par "HL" retorna o ano, o registrador "D" o mês, o registrador "E" o dia, e o acumulador o dia da semana (0 = domingo, 1 = segunda, etc.). Os valores retornados são binários, e devem ser convertidos se se quiser sair no vídeo.

Função 2B - Modifica data. Os pares "DE" e "HL" deverão conter as informações de data conforme descrito na função 2A.

Função 2C - Hora. Somente em computadores com relógio implementado por HARDWARE. O registrador "H" retorna a hora, "L" retorna os minutos, "D" retorna os segundos, e "E" retorna os centésimos de segundo.

Função 2D - Modifica a hora. Modifica a hora em computadores

com relógio implementado. Os registradores "H", "L", "D" e "E" devem conter os valores conforme descrito na função 2C.

Função 2E - Verificação de escrita. Confere se a gravação foi bem sucedida. Esta função varia conforme o sistema operacional (SOLX-DOS, DDXDOS, etc.). Portanto, é aconselhável experimentar para verificar como é que funciona no seu sistema.

Função 2F - Leitura de setores. Lê os setores especificados pelo par "DE" para o endereço de DMA. O registrador "H" deverá conter o número de setores a serem lidos. O registrador "L" conterá o drive de onde ler (0 = "A", 1 = "B", etc.). Use somente drives válidos, ou o seu programa irá parar, pois o sistema tentará, eternamente, ler o drive que não existe.

Função 30 - Escrita de setores. O par "DE" e os registradores "H" e "L" devem ser setados como na função 2F. O endereço inicial é o do DMA.

O Diretório no MSX

O diretório de um disquete está localizado nos setores 5, 6, 7 e 8 em discos de face simples, e nos setores 5, 6, 7, 8, 9, 10 e 11 em disco de dupla face. O conjunto dos bytes armazenados nestes setores estão divididos em grupos de 32 bytes, cada um dos quais aponta para um único arquivo. Estes 32 bytes estão organizados da seguinte maneira:

- Os 8 primeiros bytes contêm o nome do arquivo.
- Os 3 bytes seguintes, a extensão do nome.
- O byte seguinte é o byte de atributo.
- Os bytes 23 e 24 possuem a hora de criação dos arquivos (só quando o seu computador tiver relógio interno, nem todas as versões de MSX o possui).
- Os bytes 25 e 26 possuem a data de criação do arquivo.
- Os bytes 27 e 28 apontam para o primeiro aglomerado do arquivo. Na FAT, este aglomerado aponta o próximo, e assim sucessivamente até que o último aponta para o aglomerado FFF, que não existe e serve para marcar o fim da seqüência de aglomerados
- Finalmente, os últimos 4 bytes possuem o tamanho do arquivo.

O Byte de Atributo no Diretório

Significado de cada bit dentro do byte. O que acontece se forem ativados. Exclusivo para MSX e PC.

Os diversos bits dentro do byte de atributo têm significados diferentes, ou sejam:

Bit 0 - Só leitura. se estiver setado, o arquivo não pode ser apagado ou modificado. Infelizmente, só funciona no PC. No MSX este bit não é verificado.

Bit 1 - Escondido: se este bit estiver setado, o arquivo não aparece no diretório. Infelizmente, no MSX o arquivo fica completamente bloqueado, não sendo possível acessá-lo mesmo de dentro de um programa.

Bit 2 - Sistema: o arquivo contém o sistema operacional ou parte do mesmo, só sendo acessível pelo sistema. No MSX não existe arquivos auxiliares para sistema, nem o sistema tem este bit setado. Assim, o efeito é o mesmo do Bit 1.

Bit 3 - Label: indica que este, na verdade, não é um arquivo, mas sim o nome do volume de disco. No MSX não existe função de chamada do mesmo, e assim este nome nunca aparece.

Bit 4 - Diretório: se este bit estiver setado, o arquivo na verdade é um subdiretório. O MSX reconhece como subdiretório, marca na saída da função DIR este arquivo como sendo diretório, mas não possui função para buscar arquivos dentro deste subdiretório.

Bit 5 - Arquivo: se estiver setado, indica que é um arquivo, e não um programa.

Bits 6 e 7 - Até a edição deste livro, não havia nenhuma definição para estes bits.

Compatibilidade de Arquivos em Disquete entre o MSX e o PC.

O padrão MSX foi criado pela MICROSOFT, que também criou o sistema operacional do PC. A nível de disquete, o padrão PC era o que permitia o maior armazenamento, e, portanto, foi adotado para o padrão MSX. A nível de arquivo ASCII os disquetes são completamente compatíveis, e um arquivo texto ou dBASE escrito por um MSX pode ser lido por um PC e vice-versa. Somente a nível de programa não há compatibilidade, pois as CPUs são diferentes.

Como funciona um Subdiretório no PC.

O subdiretório é um arquivo comum, obedecendo às mesmas regras para sua localização dentro do arquivo principal. Só que o bit 4 do seu byte de atributo está setado, e, internamente, ele está organizado da mesma maneira que o diretório, apontando para a mesma FAT que o diretório principal. Isto permite ao sistema saber quais as áreas do disco estão ocupadas.

As Instruções do Z80

Conjunto de Instruções - Z80 possui mais de 300 instruções. Podemos agrupar estas instruções em nove grupos conforme suas funções:

• Instruções de movimentação de dados:
LD r1,r2 - "r" pode ser acumulador, "B", "C", "D", "E", "H", "L", a posição de memória indicada pelo par "HL" ou a posição de memória indicada pelo conteúdo do registrador índice somado ao deslocamento que veio com a instrução. Exceção neste grupo é o fato de não existir **LD (HL), (HL)**. Nesta instrução inclui-se para o caso específico de o outro registro ser o acumulador, como registros possíveis para "r" os registradores de refresh e o de interrupção.
Ex: **LD B**, (IX+Deslocamento).

LD r,n - carrega o registrador "r" com a constante "n". Nesta instrução inclui-se como registrador possível "r" a memória apontada pelo registrador index somado ao deslocamento dado na instrução.

LD A, (ss) - carrega o acumulador com a memória apontada pelo par "ss".

LD (ss), **A** - função oposta a **LD A**, (ss)

LD ss,nn - carrega o par ss com a constante "nn". Nesta instrução inclui-se como registradores possíveis para "ss" os registradores de pilha e os de index.

LD ss,(nn) - carrega o registrador "ss" com o conteúdo da memória indicada por "nn". Nesta instrução inclui-se como registradores possíveis para "ss" os registradores de pilha e os de index.

LD (nn),ss - função oposta a **LD** ss,(nn).

LD SP,HL - carrega o apontador de pilha com o conteúdo de "HL".

LDI - carrega a memória apontada pelo par **"DE"** com o conteúdo da memória apontada por **"HL"**. Incrementa **"HL"** e **"DE"**. Decrementa **"BC"**.

LDIR - idêntico a anterior, só que se repete automaticamente até o par **"BC"** igualar 0000.

LDD - idêntico a **LDI**, só que decrementa **"HL"** e **"DE"**

LDDR - idêntico a anterior, só que se repete, automaticamente, até o par **"BC"** igualar 0000.

PUSH rp - salva na pilha o par ou registro "rp". O "rp" pode significar "AF" (acumulador e flags), "BC", "DE", "HL", "IX" e "IY".

POP rp - função oposta a **PUSH** rp.

• Instruções matemáticas: Nas instruções deste grupo entenda por "r" o acumulador, os registradores "B", "C", "D", "E", "H", "L", a memória apontada pelo par "HL" ou a memória apontada pelo registrador de index com o deslocamento dado na instrução. Por "ss" entenda os registradores "BC", "DE", "HL", "SP", e os registradores de index.
ADD A, r/SUB r - soma/subtrai "r" ao/do acumulador.

ADC/SBC A,r - soma/subtrai "r" ao/do acumulador com carry.

ADD A, n/**SUB** n - soma/subtrai a constante "n" ao/do acumulador.

ADD/SBC A,n - soma/subtrai a constante "n" ao/do acumulador com carry.

ADD HL,ss - soma o registrador "ss" com o par "HL".

ADC/SBC HL,ss - soma/subtrai o registrador "ss" ao/do par "HL" com carry.

CP r - equivalente a "SUB r", mas não destrói o acumulador.

CP n - equivalente a "SUB n", mas não destrói o acumulador.

CPI - equivalente a SUB A,(HL), só que não destrói o acumulador, incrementa "HL", decrementa "BC".

CPIR - equivalente ao anterior, só que se auto-repete até "BC" se tor-

nar zero ou a subtração se tornar zero. Se o flag "Z" vier setado isto significa que a subtração deu zero, se não, que "BC" se tornou zero.

CPD - equivalente a "CPI", só decrementa "HL".

CPDR - equivalente ao anterior, só que se auto-repete até "BC" se tornar zero ou a subtração se tornar zero. Se o flag "Z" vier setado isto significa que a subtração deu zero, se não, que "BC" se tornou zero.

CPL - complementa o acumulador, tornando os bits "1" em "0" e vice-versa.

NEG - gera o complemento a 2 do valor do acumulador.

DAA - executa o ajuste decimal para as operações em "BCD".

INC r - incrementa o registrador "r".

INC ss - incrementa o par "ss".

DEC r - decrementa o registrador "r".

DEC ss - decrementa o par "ss".

• Instruções lógicas:

AND r - executa um "e" lógico do acumulador com o registrador "r".
OR r - executa um "ou" lógico do acumulador com o registrador "r".
XOR r - executa um "ou exclusivo" (bits iguais = 0, bits diferentes = 1) do acumulador com o registrador "r".
AND n - executa um "e" lógico do acumulador com a constante "n"
OR n - executa um "ou" lógico do acumulador com a constante "n"
XOR n - executa um "ou exclusivo" do acumulador com a constante "n".

• Instruções de controle de programa:

CALL nn - salva "PC" na pilha e vai executar a subrotina que se inicia no endereço "nn".
CALL - condição,nn - se a condição for válida, executa o mesmo que a instrução anterior. Se não, não faz nada.
JP nn - pula para a posição "nn" (posição de memória de programa).
JP condição, nn - se a condição for válida, executa o mesmo que a instrução anterior. Se não, não faz nada.

JR d - pula para a posição calculada pela soma do conteúdo de "PC" com o deslocamento "d". O deslocamento é dado em complemento de 2, podendo variar de -128 a +127.
JR condição,n - se a condição for válida, executa o mesmo que a instrução anterior. Se não, não faz nada. Só testa as condições relacionadas com os flags "Z" e CARRY.
JP (HL) - JP (IX) - JP (IY) - pula para a posição indicada pelo conteúdo do registrador indicado pela instrução.
RET - retorno de subrotina. Restaura "PC" com o primeiro elemento da pilha.
RETI - equivalente a anterior, só que os periféricos da família Z80 que podem trabalhar em "daisy chain" reconhecem este código e permitem que a próxima interrupção, mais prioritária que por acaso estiver ativa, também interrompa o Z80.
RETN - equivalente à seqüência "EI"; "RET". Deve ser usada após uma interrupção "NMI".
RST n - equivalente a "CALL n", mas "n" só pode assumir os valores "0", "8", "10H", "18H", "20H", "28H", "30H", "38H". Tem como vantagem, nestes casos, o fato de só ocupar uma posição de memória.

• Instruções de troca de registros:

EX AF,AF' - troca o acumulador e os flags pelas suas duplicatas.
EXX - troca os pares "BC", "DE", "HL", pelas suas duplicatas.
EX DE,HL - troca o par "DE" com o par "HL"
EX (SP),HL - troca a primeira palavra da pilha com o par "HL".
EX (SP), registrador de index - troca a primeira palavra da pilha com o registrador de index.

• Instruções de entrada/saída:

IN A, (n) - carrega o acumulador com o conteúdo da porta "n".
IN r,(C) - carrega o registrador "r" com o conteúdo da porta cujo endereço é o conteúdo do registrador "C".
INI - carrega a memória apontada por "HL" com o conteúdo da porta cujo endereço é o conteúdo do registrador "C". Incrementa "HL", decrementa "B".
INIR - idêntico a anterior, só que se repete, automaticamente, até que "B" iguale 00.

Os opostos das instruções de entrada são respectivamente: **OUT** (n),A; **OUT** (C),r; **OUTI; OTIR.**

• Instruções de manipulação de bits:

Nas instruções deste grupo, "b" representa um número de 0 a 7, indicando um bit do registro "r". Por sua vez, "r" pode ser o acumulador, os registradores "B", "C", "D" "E", "H", "L", e a memória apontada por "HL".
BIT b,r - testa o bit "b" do registrador "r", colocando sua condição de zerado ou não no flag "Z".
CLR b,r - zera o bit "b" do registrador "r".
SET b,r - leva a 1 o bit "b" do registrador "r".

• Instruções de controle:

NOP - não faz nada, só consome 4 ciclos de máquina (tempo).
HALT - para todas as funções do Z80, com exceção do refresh. Só sai desta condição por interrupção ou resetando o Z80.
EI - libera a interrupção "INT".
DI - bloqueia a interrupção "INT", "NMI" não é afetada.

IM n - define modo de interrupção "n":
n = 0: idêntico ao 8080. Aguarda uma instrução externa, geralmente a instrução "RST n".
n = 1: endereço fixo de interrupção = 38H.
n = 2: daisy-chain. Um circuito externo envia um byte que deve ser par. Este byte forma a parte baixa de um endereço, a parte alta está no registrador de interrupção "I". Neste endereço, então, o Z80 busca o endereço de tratamento da interrupção. Assim, até 128 periféricos podem interromper o Z80, e cada um ter a sua rotina particular de tratamento. A prioridade de interrupção é feita por circuito externo que deve reconhecer a instrução "RETI" como liberação de nova interrupção.
SCF - seta o flag CARRY.
CCF - complementa o flag CARRY.

• Instruções de rotação:

RLC r - roda o registrador "r" para a esquerda, ou seja, o conteúdo do bit 0 vai para o bit 1, o do bit 1 para o bit 2, ..., e o do bit 7 vai para o bit 0 e o flag CARRY. Ocupa 2 bytes.
RLCA - idêntico a anterior, para o caso específico do acumulador. Só ocupa 1 byte.
RL r - idêntico "RLC r", só que o bit 7 vai somente para o flag CARRY, e o flag CARRY é que vai para o bit 0.
RLA - mesmas considerações que "RLCA".

RRC r - roda o registrador "r" para a direita, ou seja, bit 7 para bit 6, bit6 para bit 5, ..., bit 0 para bit 7 e o flag CARRY.
RRCA/RR r/RRA - mesmas considerações que nos casos de rotação-para esquerda.
SLA r - shift para esquerda matemático: o bit 0 vai para o bit 1, obit 1 para o bit 2, ..., bit 7 vai para o flag CARRY. A posição do bit 0 é zerada. Equivale a multiplicar por 2.
SRA r - shift para direita matemático: o bit 7 vai para o bit 6 e é copiado em si mesmo, o bit 6 vai para o bit 5, o bit 5 vai para o bit 4, ..., o bit 0 vai para o flag CARRY . Equivale a dividir por 2.
SRL r - idêntico ao anterior, só que o bit 7 é zerado.
Nota: Nas duas próximas instruções, **NBAA** significará os 4 bits mais do acumulador, **NBAM** os 4 bits mais significativos da memória apontada pelo par "HL", e **NBBM** os 4 bits menos significativos desta mesma memória.
RLD - Rotação de nibbles para esquerda: **NBBM** passa a posição **NBAM**, o **NBAM** passa para a posição **NBBA**, e **NBBA** passa para a posição **NBBM**. O **NBAA** não muda.
RRD - rotação de nibble para a direita: **NBAM** passa para **NBBM**, **NBBM** passa para **NBBA**, **NBBA** passa para **NBAM**. O **NBAA** não muda.

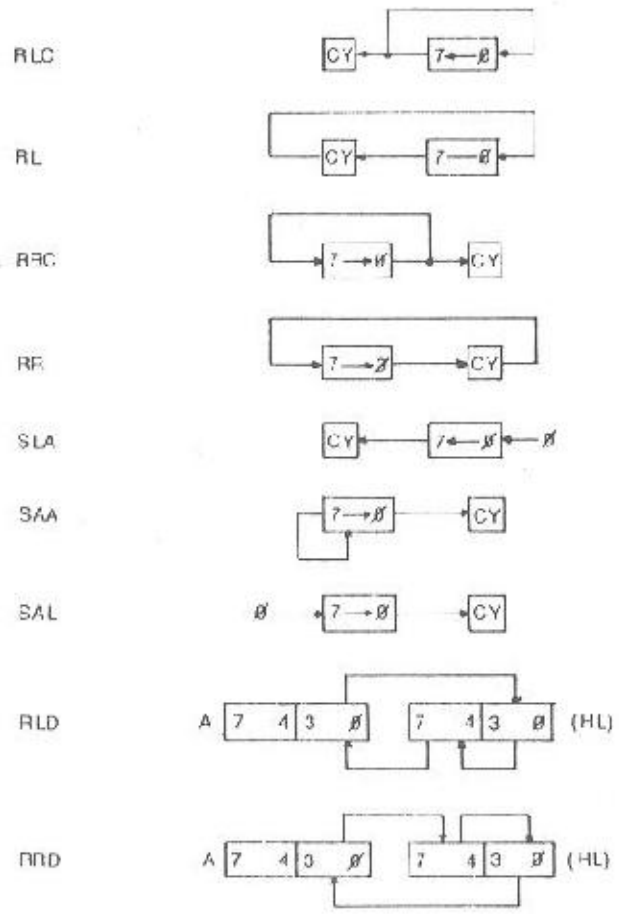


Fig. 3 As instruções de rotação

Bibliografia

Livros

Pritchard, Joe. Lenguaje Máquina para MSX.ANAYA

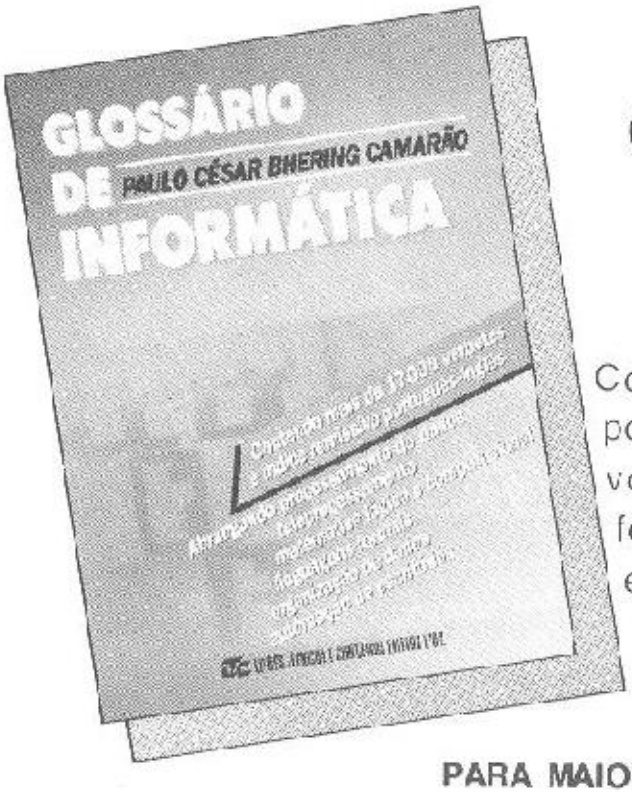
Barbosa, Eduardo Alberto. Dicas, Macetes e Programas em Assembly para MSX DISK DRIVE.

Ciência Moderna

Hogan, Thom. CP/M – Guia do Usuário. McGraw-Hill

DOBRE

CONHEÇA O MAIS ATUAL E COMPLETO **GLOSSÁRIO DE INFORMÁTICA**



Com mais de 17.000 verbetes e enriquecido por um índice remissivo português-inglês, você terá nas mãos uma indispensável ferramenta de consulta, tanto para estudantes, quanto profissionais, que precisam conhecer e utilizar corretamente esta linguagem técnica.

PARA MAIORES INFORMAÇÕES:

LIVROS TÉCNICOS E CIENTÍFICOS EDITORA LTDA.

Rua Vieira Bueno, 21
20920 — Rio de Janeiro — RJ
Tels.: (021) 580-6055 (Geral)
580-9374 (Vendas)

Rua Vitória, 486 — 2º andar
01210 — São Paulo — SP
Tel.: (011) 223-9866

DESTAQUE E ENVIE

FAÇA PARTE DE NOSSO MAILLING-LIST

SIM! DESEJO RECEBER GRATUITAMENTE TODO O MATERIAL INFORMATIVO SOBRE OS LANÇAMENTOS NA ÁREA DE INFORMÁTICA.

NOME: _____

ENDEREÇO: _____

CEP: _____ CIDADE: _____ ESTADO: _____

TEL.: (_____) _____ PROFISSÃO: _____

DATA: _____ ASSINATURA: _____

DOBRE

COLE
O SELO
AQUI

LTC LIVROS TÉCNICOS E CIENTÍFICOS EDITORA LTDA.

Caixa Postal 21162 Rio de Janeiro / RJ

20081

DOBRE

RICHARD SPIEGEL

ROTINAS PRONTAS MSX

Este livro fornece dicas importantes aos usuários de micros da linha MSX e CP/M que desejam escrever programas em linguagem de máquina, usando o sistema operacional MSX-DOS. Porém, serve também como livro-texto para os cursos de Assembler Z-80.

Todas as rotinas apresentadas foram convenientemente testadas, visando acelerar o trabalho do usuário na geração de qualquer programa. Elas são comentadas linha a linha com sugestões de como alterá-las para obter outros resultados.

As dicas aqui apresentadas não são encontradas em nenhum outro livro nacional e também não são ensinadas por nenhum professor, portanto, com este livro em mãos você saberá qual a melhor maneira de programar.

MAIS UM LANÇAMENTO
DA

LIVROS TECNICOS E CIENTIFICOS EDITORA LTDA.

ISBN 85-216-0638-9