AGRADECIMENTO

Agradecemos à família do autor, à esposa, e aos filhos, Daniel Monteiro Lima, Daniela Monteiro Lima e Carolina Rosa de Lima, por autorizar a divulgação destes livros.

Livros como estes, numa época que não existia tanta divulgação de informação, foram essenciais para o aprendizado de programação dos computadores da linha Sinclair ZX81, (TK85, CP200).

A obra de Vosso pai será para sempre lembrada pelos entusiastas de retrocomputação pelo mundo todo.

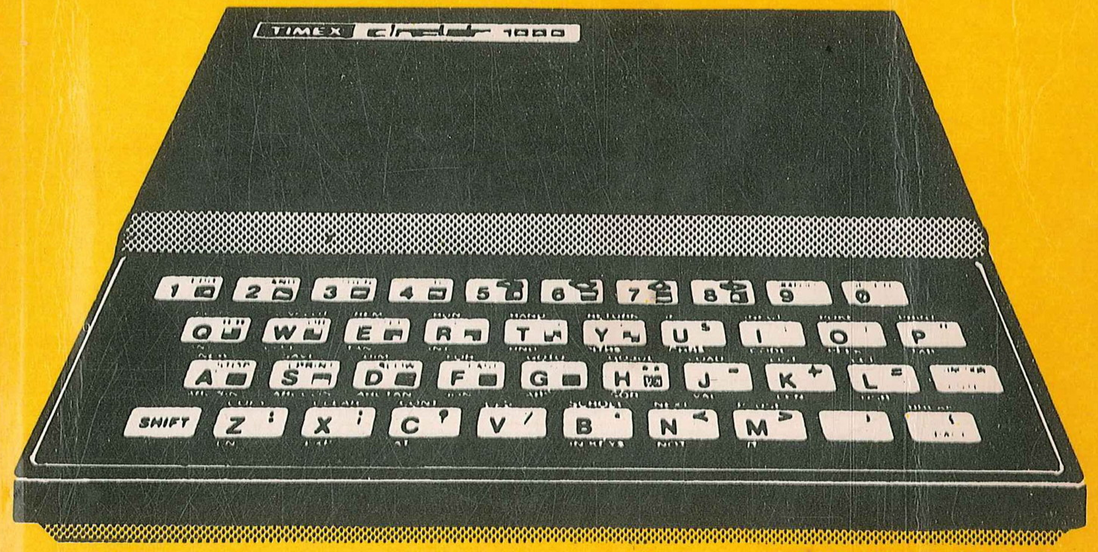
O Sr. Délio foi um homem à frente de seu tempo.

Muito obrigado Sr. Délio Santos Lima.

DELIO SANTOS LIMA

CÓDIGO DE MÁQUINA PARA TK E CP 200

CÓDIGO DE MÁQUINA PARA TK E CP 200



NÚMEROS BINÁRIOS E HEXADECIMAIS - ARQUITETURA DO Z80
EDITANDO EM CÓDIGO - PROGRAMAS PARA EDIÇÃO
AS INSTRUÇÕES DO Z80 EM EXEMPLOS - SUB-ROTINAS DA ROM
DICIONÁRIO DAS INSTRUÇÕES - MNEMÔNICOS X HEX
HEX X DECIMAL - DISASSEMBLER DA ROM - ETC. ...

DELIO SANTOS LIMA

Do mesmo autor:

45 PROGRAMAS PRONTOS PARA RODAR EM TK 82C E NE Z8000

APLICAÇÕES SÉRIAS PARA TK 82C E CP 200

30 JOGOS PARA TK 82C E CP 200

SINCLAIR 8K ROM - DISASSEMBLY COMPLETO COMENTADO

IMPRESSO POR J.A.C. EDITORA GRÁFICA

SÃO JOSÉ DOS CAMPOS - SP

CÓDIGO DE MÁQUINA PARA TK E CP 200
por Delio Santos Lima

1ª Edição - Novembro de 1983

Composto, editado e distribuído por
Micron Eletrônica Com. Ind. Ltda.
São José dos Campos - SP - Brasil

PREFÁCIO

Entramos talvez na mais importante era da eletrônica.

A era da MICRO ELETRÔNICA e do MICROPROCESSADOR.

Agora, com os microcomputadores de Lógica Sinclair, torna-se possível que, sem conhecimentos específicos de eletrônica e hardware, tenha-se acesso à linguagem do mais famoso e revolucionário microprocessador dos últimos anos, o Z80.

Neste volume, voce encontrará variados exemplos das instruções do Z80 em analogia com o Basic, um dicionário das instruções em ordem alfabética, um apêndice com as instruções e seus códigos, uma coleção de programas, além de sub-rotinas da ROM e etc...

Para concluir estas páginas, perdi alguns fios de cabelo e muitas horas de sono. Se voce pretende extrair tudo que aqui está, saiba desde já que não será fácil. Exigirá paciência, tempo e dedicação, apesar de ter sido escrito de forma a facilitar ao máximo o seu entendimento.

Delio santos Lima
Caixa Postal 100
12200 São José dos Campos
SP - Brasil

ÍNDICE

UMA INTRODUÇÃO

AS INSTRUÇÕES

SÓ PROGRAMAS

SUB-ROTINAS DA ROM

A ROM

APÊNDICES

BIBLIOGRAFIA

Zilog Inc. , Z80 Data Sheets e Instruction Set Summary

Leventhal, Lance A., Z80 Assembly Language Programming

Ciarcia, Steve, Build Your Own Z80 Computer

Logan, Ian, Understanding Your ZX 81 ROM.

Melbourne House Publishers, Machine Language Made Simple

Baker, Toni, Mastering Machine Code on Your ZX 81 and ZX 80

Time Data Ltd., The ZX 80 Magic Book with ZX 81 suplement

Hewson, Andrew D., Hints & Tips for the ZX 81

Maunder, Robert, The ZX 81 Companion

Sinclair Research Ltd., ZX 81 Operation Manual

Microdigital, TK 82C e TK 85, Manuais de Uso e Programação

Prológica, CP 200 Manual de Operação e Curso de Programação

Sucesu, Dicionário de Informática , Inglês - Português. 3ª Ed.

INTRODUÇÃO

MICRO, PROCESSADOR OU COMPUTADOR?

Os microcomputadores e computadores em geral são basicamente constituídos de módulos de:

A. Entrada e saída de dados, como teclado e vídeo.

B. Armazenamento de dados, memórias. RAM e ROM.

C. Controle do sistema incluindo uma unidade lógica e aritmética.

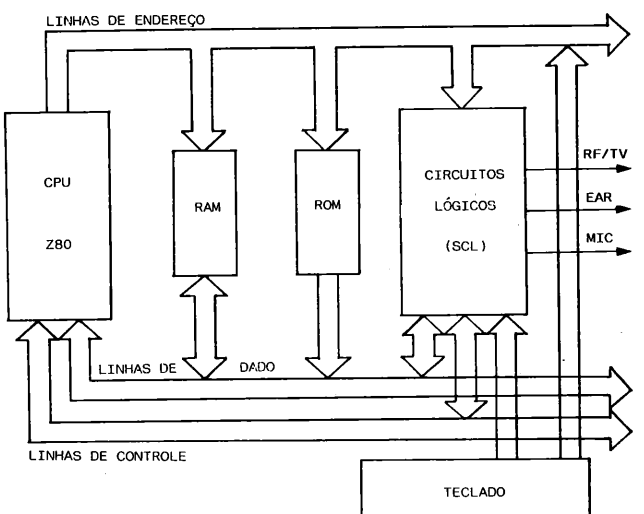
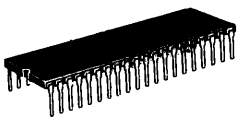


DIAGRAMA DE BLOCOS — MICROCOMPUTADORES ZX, TK, etc.

Todo o controle do sistema, incluindo as operações lógicas e aritméticas como soma e subtração, além de memórias que armazenam dados, resultados intermediários, instruções em uso, etc..., compõem a CPU, Unidade Central de Processamento.

A CPU do seu microcomputador está baseada em um único circuito integrado, o "chip" Z80, criado pela Zilog Inc. . Externamente é só uma famigerada pastilha plástica com aproximadamente 7 cm^2 e 40 "pernas". Contém milhares de componentes microscopicamente montados e é capaz de manipular centenas de milhares de dados em segundos, processando-os de acordo com instruções. É, portanto , uma máquina MICROprocessadora, programável em sua linguagem própria, utilizando um mínimo de circuitos adicionais externos.



Z80 — ASPECTO EXTERNO

O MICROPROCESSADOR É UM COMPONENTE.

O MICROCOMPUTADOR É UM SISTEMA.

Esta máquina microprocessadora, o Z80, é programável apenas em sua linguagem. A linguagem da máquina. Cada modelo de microprocessador possui a sua própria linguagem.

Um programa, nesta linguagem, está gravado na ROM do seu micro, implementando uma outra linguagem de programação, fácil e compreensiva no que se refere "a lógica humana", o Basic.

BINÁRIO E HEXADECIMAL

Os microprocessadores trabalham com valores no sistema binário. Alto ou baixo, zero ou um, verdadeiro ou falso, sim ou não são as formas de valores que estas máquinas são capazes de interpretar. São os chamados números binários, feitos de combinações de dígitos Ø e 1 . O sistema binário é de grande facilidade de interpretação das máquinas. Um dígito pode assumir valor Ø ou 1, assim como se fôsse um interruptor, ligado ou desligado.

No sistema decimal, com um dígito podemos representar 1Ø diferentes valores, porque existem 1Ø diferentes símbolos.

Em binário, com um dígito podemos representar apenas 2 diferentes valores, porque existem apenas 2 diferentes símbolos, Ø e 1 . Com dois dígitos podemos fazer as combinações:

BINÁRIO	EQUIVALENTE DECIMAL
0 0	0
0 1	1
1 0	2
1 1	3

e representar 4 diferentes valores, respectivamente de Ø a 3 . Com tres dígitos, as possibilidades ficam assim:

BINÁRIO	EQUIVALENTE DECIMAL
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

e representam 8 diferentes valores.

Abaixo, conversão do número binário 11011111, em decimal.

BINÁRIO 1 1 0 1 1 1 1 1

$1 \times 2^0 = 1$
$1 \times 2^1 = 2$
$1 \times 2^2 = 4$
$1 \times 2^3 = 8$
$1 \times 2^4 = 16$
$0 \times 2^5 = 0$
$1 \times 2^6 = 64$
$1 \times 2^7 = 128$

DECIMAL TOTAL = 223

O número binário acima possui 8 dígitos ou bits. Bit significa dígito binário. BInary digiT .

A um agrupamento de bits dá-se o nome de byte.

No seu computador, cada endereço ou posição da memória dispõe de 8 bits capazes de armazenar de 00000000 até 11111111, equivalendo de 0 até 255 decimal. Para trabalhar com valores acima de 255 é necessário utilizar mais de um endereço ou posição da memória.

Um programa, em código binário, seria algo do tipo:

```
00111010
01100000
00000000
00000001
00010100
11001011
```

com os seguintes problemas:

1. Difícil de entender ou traduzir, à primeira vista.
2. Os códigos não descrevem o que fazem, como em Basic.
3. Longa quantidade de elementos a serem digitados.
4. Fácil ocorrência de erros na digitação, com grande dificuldade para localizá-los.

Por estas e outras razões, em Assembly utiliza-se comumente o sistema numérico hexadecimal.

O sistema hexadecimal assemelha-se ao decimal, porém emprega 16 diferentes símbolos. São eles:

0 1 2 3 4 5 6 7 8 9 A B C D E F

As letras de A até F representam respectivamente de 10 a 15 decimal. O valor máximo de um número hexadecimal de 2 dígitos é FF. A seguir, conversão dos números hexadecimais 0E e FF para decimais:

HEXADECIMAL 0 E

$14 \times 16^0 = 14$
$0 \times 16^1 = 0$

DECIMAL TOTAL = 14

HEXADECIMAL F F

$15 \times 16^0 = 15$
$15 \times 16^1 = 240$

DECIMAL TOTAL = 255

Para converter um número hexadecimal de 2 dígitos em decimal, multiplique o dígito da esquerda por 16 e some o resultado ao dígito da direita.

A utilização do sistema hexadecimal oferece as seguintes vantagens em relação ao sistema binário:

1. Fácil conversão para o sistema decimal e vice-versa.
2. Fácil visualização quanto ao seu número de dígitos em binário e vice-versa. Oito bits "cabem" em dois dígitos em hexadecimal e quatro bits "cabem" em um dígito em hexadecimal.
3. Fácil conversão para o sistema binário e vice-versa, ou,pelo menos,melhor do que com o sistema decimal.

NÚMEROS BINÁRIOS MAIORES QUE 255

Não "cabem" em um único byte, de 8 bits, e ficam na forma:

1º byte 11111111 2º byte 11111111

para o máximo valor admissível em 2 bytes, 65535 em decimal.

A regra para transformar binários de 16 dígitos em decimais é a mesma que para 8, só que o bit mais significativo atinge 2^{15}, ficando:

$2^0 + 2^1 + 2^2 + 2^3 + 2^4 . . . . . + 2^{14} + 2^{15} = 65535$

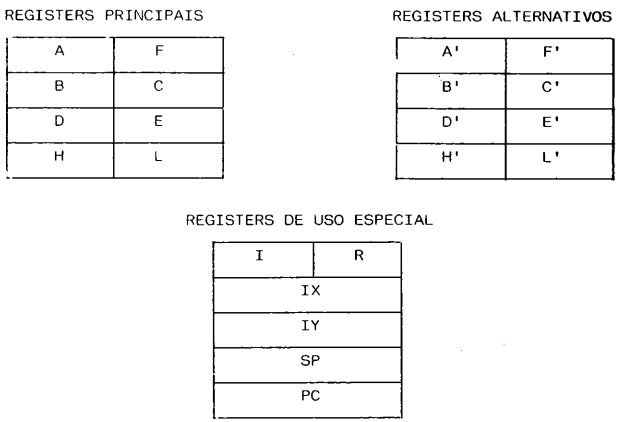
Caso se converta para decimal cada um dos bytes, como habitualmente é feito, isoladamente, é necessário que se multiplique o valor do 2º byte por 256, somando-o ao 1º para obter a resposta correta. A razão: é a diferença dos índices das potências do 2º byte.

O microprocessador Z80 possui 8 linhas de dados, manipulando simultaneamente 8 bits , com as memórias e outros dispositivos. É chamado de µprocessador de 8 bits.

Veja,no final deste volume, os apêndices de conversões numéricas.

Z80
REGISTERS, FLAGS E STACK

O Z80 é um microprocessador de oito bits orientado por registers. Os registers da CPU são células de memória ou posições de memória programável, como qualquer outra do sistema. O Z80 possui dezoito registers de 8 bits e quatro de 16, podendo armazenar e manipular vários números ao mesmo tempo.



Z80 — CONJUNTO DE REGISTERS

Os registers estão agrupados em pares. BC é um "register pair"de 16 bits, formado pelos registers B e C de 8 bits cada um. Podem ser usados isoladamente ou em par, assim como DE, HL e seus alternativos.

Os registers IX, IY, SP e PC só podem ser usados na forma de par.

O mais importante register de uso geral é o A, A de acumulador. A maioria das instruções aritméticas e lógicas oferecidas pelo Z80 referem-se a este register.

O register F, F de flag ou bandeira é um register totalmente diferente dos demais. Cada um dos seus bits é usado separadamente, como uma bandeira de indicação, fornecendo certos estados ou ocorrências no processador. As principais bandeiras são:

Sign flag é de valor um, quando o resultado da última operação for negativo.
Zero flag é um, quando o resultado da última operação for zero.
Carry flag é um, quando o resultado da última operação produzir um número maior que a capacidade de armazenamento.
Parity flag é um, quando o resultado for par e zero, se ímpar.
Overflow flag é um, quando o bit que fornece o sinal de um resultado tiver sido alterado por uma "sobrecarga" de valor.

A execução de certas instruções depende da situação de determinadas bandeiras.

Os registers BC, DE e HL são de uso geral. As letras do register HL sugestivamente referem-se a High (alto) e Low (baixo), servindo como lembrete de qual dos registers contém o valor mais significativo.

Os registers IX e IY são index, registers índices, como o nome diz, e servem para facilitar a construção de tabelas. Armazenam o endereço, quando operando no modo de endereçamento indexado. O register pair IR, interrupt memory refresh, não é habitualmente usado.

O register SP, Stack Pointer contém o endereço do stack, ou pilha de dados. O stack é uma área não fixa na RAM, junto ao final, onde o processador coloca dados temporariamente, pertinentes ao seu funcionamento, como endereços de sub-rotinas, o programa, etc. ... O stack funciona como uma pilha de fichas. Colocam-se as fichas umas sobre as outras, empilhando-as. Ao se retirar as fichas, começamos pelo topo da pilha, ou pela última que foi colocada lá. O endereço dado pelo register SP é o da última informação entrada. O stack pode ser usado pelo programador pelas instruções PUSH e POP. PUSH (algum register) no stack armazena-o e POP retira-o.

O register PC é o posicionador do programa. Fornece o endereço da próxima instrução a ser executada.

Observação : As bandeiras Parity e Overflown são dadas por um mesmo bit do register F. Se a operação executada for aritmética, é a overflown flag. Se for operação Booleana, de Input ou Rotate, é a parity flag .

OS CÓDIGOS DO CÓDIGO

O microprocessador não entende absolutamente nada de Basic, mas um conjunto de instruções e valores na forma binária. O código da máquina, "que a máquina entende", são números binários e é humanamente impossível memorizá-los ou sequer identificá-los, rapidamente. Mesmo utilizando o sistema decimal ou hexadecimal, isto não se resolve, razão pela qual é atribuído a cada código um mnemônico. Um conjunto de caracteres alfanuméricos relacionados ou apenas abreviados de uma palavra que exemplifique o sentido da instrução.

Aqui, neste volume, os mesmos permanecem em seu idioma de origem, o inglês, por se tratar de uma "linguagem" de programação, universal e intradutível, assim como o Basic, ou estaríamos criando uma outra "linguagem".

Abaixo, exemplo de códigos, mnemônicos e significados:

CÓDIGO DECIMAL	EM HEXADECIMAL	MNEMÔNICO	SIGNIFICA
33 00 00	21 00 00	LD HL,0	LOAD HL com 0
201	C9	RET	RETURN

Existem algumas regras para "se ler" um mnemônico e elas estão mencionadas no início do capítulo "As Instruções do Z80". Ainda no final deste volume voce encontrará as tabelas de conversão de códigos (hex) para mnemônicos e vice-versa.

CONCLUSÕES

Os microprocessadores só falam números binários, não entendem Basic e muito menos números de linhas. Só conhecem posições ou endereços de memória.

Humanos só entendem, facilmente, números decimais e hexadecimais.

Quando ligamos o microcomputador, ele está em Basic e não entende os hexadecimais ou binários. Como então usar os códigos e fazer o microprocessador interpretá-los em sua linguagem ?

É justamente no Basic do seu TK, CP, etc... que está a resposta, em comandos para este fim. PEEK, POKE e USR.

POKE coloca um valor decimal entre 0 e 255 em uma posição de memória dada por um endereço entre 0 e 65535.

PEEK lê a posição de memória dada por um endereço entre 0 e 65535.

USR roda uma rotina em código de máquina a partir do endereço dado.

Como PEEK e POKE só trabalham com números decimais, é preciso converter os hexadecimais. Colocá-los em binário na memória é problema dos circuitos e não vamos nos preocupar com isso. Este tipo de conversão e endereçamento para leitura e escrita por PEEK e POKE pode ser facilitado com o auxílio de "um outro" programa para edição de códigos.

Endereçar ? ler e escrever códigos ?, mas em que endereços ?

A ORGANIZAÇÃO DA MEMÓRIA

Assim como o Z80 possui 8 linhas para dados, identificando 8 bits, ele possui 16 linhas de endereços, identificando 16 bits que equivalem a 65535. Portanto, podemos identificar 65536 posições de memória de 8 bits cada. Dentro deste princípio, a memória do seu microcomputador foi ditribuída, como a seguir:

ENDEREÇO	UTILIZAÇÃO
00 000	sistema operacional
08 192	reserva para sistema operacional
16 384	variáveis do sistema operacional
16 509	programa do usuário
(D.File)	vídeo
(Vars.)	variáveis do programa usuário
(E.Line)	linha em edição
(StkBot)	stack do calculador
(StkEnd)	área livre
SP	stack da máquina
(ErrSP)	stack para endereços de sub-rotinas
(RamTop)	último endereço da memória

A ROM está nos endereços de 00 000 a 8 191.

A RAM está nos endereços de 16 384 em diante.

Para 2 K de RAM, de 16 384 a 18431.

Para 16 K de RAM, de 16 384 a 32767.

Para 48 K de RAM, de 16 384 a 65535.

Conforme o diagrama da página anterior, a área da memória destinada ao usuário, situa-se nos endereços 16509 em diante, existindo, após o mesmo, uma série de outras áreas reservadas para uso do sistema operacional. Estas outras áreas apresentam-se com comprimento ou consumo de memória variável. Suas posições, na memória, variam em função das variações de comprimento, do programa em Basic, da RAM instalada, etc...

O endereço de início de cada uma destas área é arquivado pela ROM na RAM do usuário nos endereços 16384 a 16508 e são as denominadas variáveis do sistema operacional. Não confundir com sistema de variáveis (do usuário). Seus usos e alterações estão amplamente discutidos em várias outras publicações, inclusive nos manuais das máquinas e omitimos aqui.

Enfim, temos um computador com um sistema operacional que utiliza a RAM do usuário para certos propósitos e, neste mesmo computador, precisamos escolher uma "parte desta RAM" para inserir os códigos.

Certos cuidados devem ser tomados, para que os códigos não sejam inseridos sobre outros que estejam na RAM, como o programa em Basic, a memória do vídeo, etc... . Isto deve ser feito dentro de certos critérios, descritos no próximo capítulo.

EDITANDO EM CÓDIGO

A seguir, apresentamos algumas áreas da RAM que se prestam ao armazenamento de programas em código, cada qual com suas vantagens e desvantagens.

1. Em uma instrução REM ou PRINT, na primeira linha do programa em Basic.
2. Nos endereços reservados para as variáveis do programa do usuário.
3. Acima do RamTop.
4. Acima do programa em Basic.

1. EM UMA INSTRUÇÃO REM OU PRINT

Coloque na primeira linha, de um programa em Basic, uma instrução REM com tantos caracteres quanto for o número de bytes do programa, em linguagem de máquina. Não importa o tipo do caractere empregado. Eles serão sobrepostos pelos códigos, uma vez que se prestam apenas para reservar os endereços conhecidos. O conteúdo da instrução REM não é interpretado em Basic.

VANTAGENS

O endereço de início da área reservada para os códigos é conhecido.

Adicionando ou retirando linhas do programa em Basic, não se alteram os endereços onde estiverem os códigos.

Os códigos são gravados para o cassette juntamente com o programa em Basic.

DESVANTAGENS

Impossível de serem lidos diretamente no vídeo. Os caracteres mostrados, equivalentes aos códigos, não fazem nenhum sentido. Os códigos 126 e 118 são reconhecidos pelo programa residente, como constante numérica e New Line, respectivamente, provocando o desaparecimento da listagem em Basic dos códigos subsequentes.

2. NOS ENDEREÇOS DAS VARIÁVEIS

Dimensione, como primeira instrução do programa em Basic, uma matriz. Por exemplo:

1 DIM A$(N) ou 1 DIM A(N/5), onde N é o número de bytes a ser reservado. O início dos endereços reservados pela matriz é dado pela variável do sistema operacional denominada Vars. e arquivada em 16400 e 16401. Para obtê-la, digite:

PRINT PEEK 16400 + 256 * PEEK 16401 + 6

O valor 6, somado ao endereço, é devido ao comprimento do nome da matriz.

VANTAGENS

O uso de matriz alfanumérica apresenta excelentes facilidades de edição de caracteres pelo Basic. Por exemplo:

LET A$(590 TO 606)="INSERINDO TEXTO".

Os códigos são gravados com o programa em Basic.

DESVANTAGENS

O endereço de início precisa ser calculado.

Sub-rotinas tornam-se quase impraticáveis, devido às constantes alterações nos endereços destinados às variáveis.

As instruções RUN e CLEAR apagam os códigos.

3. ACIMA DO RAMTOP

As posições de memória 16 388 e 16 389 armazenam a variável do sistema operacional que indica o último endereço da RAM.

Digite : PRINT PEEK 16388 + 256 * PEEK 16389 para obter o último endereço da RAM instalada. "Baixando-se" o RamTop, os endereços posteriores ficam totalmente imunes ao sistema operacional, incluindo as instruções New, Load, Save e outras.

VANTAGENS

O programa colocado nestes endereços, acima do RamTop , não será removido por Clear, New, Save, Load, etc. .

Com códigos acima do RamTop pode-se carregar outro programa,via teclado ou cassette.

DESVANTAGENS

O programa acima do RamTop não pode ser gravado no cassete, a menos que o mesmo seja transposto para "a memória baixa".

A alteração da variável RamTop deve ser feita, antes da entrada de qualquer programa.

4. ACIMA DO PROGRAMA EM BASIC

Coloque na primeira linha de um programa em Basic, uma instrução REM, seguida de alguns caracteres. O número de caracteres deve ser igual ao número de bytes do programa em código, menos cinco. Em seguida digite POKE 16 509,-1

O endereço de início da área reservada é dado por:

PRINT PEEK 16 396 + 256 * PEEK 16 397 - N, onde N é o número de bytes da rotina em código.

VANTAGENS

Os códigos podem ser gravados com o programa em Basic.

Os códigos não serão afetados por RUN e CLEAR.

DESVANTAGENS

O endereço de início precisa ser calculado.

Toda a listagem do programa em Basic, juntamente com os códigos da linha 1 REM, não serão mais listáveis. A razão disto é que o número da linha 1 foi transformado em valor superior ao admissível. Por POKE 16509,0 a listagem volta.

HEXLOAD

A seguir, apresentamos um programa em Basic para facilitar a edição de códigos, na forma hexadecimal. Permite entrar, alterar, listar, rodar e gravar os códigos. Possui também um conversor de números hexadecimais para decimais e vice versa.

O menu oferecido:

```
                HEXLOAD

P/                  DIGITE

EDITAR              E
RODAR               R
GRAVAR              G

DEC > HEX           DH
HEX > DEC           HD
```

E. Digite E para entrar, alterar e listar códigos. Esta opção inicialmente solicita: 1º ENDEREÇO ?. Voce deve informar o endereço a partir do qual voce quer editar. Usualmente,usaremos o endereço 16514 que corresponde ao primeiro caractere após a instrução da primeira linha do programa em Basic. Uma vez informado o endereço, aparecerão no vídeo o endereço solicitado e o seu conteúdo em hex. Digite apenas New Line (Enter) . O conteúdo deste endereço não muda, passando-se ao seguinte e assim por diante.

Digite M e volta-se ao menu.

Digite DH para conversão decimal-hex, ou HD para hex-decimal.

Durante as conversões voce pode digitar R, para retornar a edição, ou, M, para menu.

Caso voce queira apenas listar os códigos e não queira ficar digitando New Line sucessivamente, digite apenas uma vez e, rapidamente,pressione a tecla L e a segure presa . Existe no programa, um comando INKEY$, antes do INPUT CÓDIGO que, se a tecla L estiver presa, o INPUT será ignorado. Caso não esteja obtendo este efeito no modo FAST, tente no SLOW, até se adaptar.

R. Para rodar, digite R e será acionado PRINT USR. Após a execução da rotina em código, o programa irá parar em um STOP. Para voltar ao menu,digite CONT e New Line.

G. Para gravar digite G. O programa será gravado juntamente com os códigos. O programa continua no menu.

DH. Para converter decimais ,de 0 a 65535, em hex, digite DH.

HD. Para converter hexadecimais, de 0 a FF FF, em decimal, digite HD. Nestas rotinas de conversão, se chamadas durante a edição, deve-se digitar R para retornar. Se chamadas diretamente do menu, deve-se digitar M para continuar no menu. R fará o programa parar por indefinição de endereço de retorno.

Os programas em código encontram-se listados em quatro colunas. A primeira à esquerda contém os endereços em decimal. A segunda coluna, contém os códigos propriamente ditos, em hexadecimais, de dois dígitos. A terceira coluna fornece os mnemônicos (Z80), referentes aos códigos da linha. A quarta coluna contém um comentário sobre a aplicação da instrução ou dado.

Usando o HEXLOAD, voce só precisará entrar os códigos em hex, constantes na segunda coluna. Sempre dois dígitos seguidos de New Line

```
 900 GOTO 988
 902 REM MICRON ELETRONICA
 904 CLS
 906 PRINT "1. ENDERECO?"
 908 INPUT X
 910 CLS
 912 PRINT "ENDERECO"; TAB 10;"CO
DIGO"
 914 PRINT
 916 FOR N=0 TO 15
 918 LET Y=PEEK (X+N)
 920 GOSUB 976
 922 PRINT X+N;TAB 10;A$;
 924 IF INKEY$= "L" THEN GOTO 984

 926 INPUT X$
 928 IF X$="HD" THEN GOTO 1072
 930 IF X$="DH" THEN GOTO 1026
 932 IF X$="M" THEN GOTO 988
 934 IF X$="" THEN GOTO 940
 936 LET Y=16*CODE (X$)+CODE (X$
(2))-476
 938 POKE X+N,Y
 940 LET Y=PEEK (X+N)
 942 GOSUB 976
 944 PRINT TAB 14;A$
 946 NEXT N
 948 PRINT
 950 PRINT "ALTERA? S/N?"
 952 INPUT X$
 954 IF X$="N" THEN GOTO 960
 956 IF X$<>"S" THEN GOTO 952
 958 GOTO 910
 960 LET X=X+15
 962 GOTO 910
 964 CLS
 966 PRINT "ENDERECO P/ USR?"
 968 INPUT S
 970 PRINT USR (S)
 972 STOP
 974 GOTO 988
 976 LET H=INT (Y/16)
 978 LET L=Y-16*H
 980 LET A$=CHR$ (H+28)+CHR$ (L+
28)
 982 RETURN
 984 LET X$=""
 986 GOTO 940
 988 PRINT
 990 CLS
 992 PRINT ,,"            HEXLOAD
"
 994 PRINT ,,,,"P/","DIGITE"
 996 PRINT ,,"EDITAR","E"
 998 PRINT "RODAR","R"
1000 PRINT "GRAVAR","G"
1002 PRINT ,,"DEC > HEX","DH"
1004 PRINT "HEX > DEC","HD"
1006 INPUT X$
1008 IF X$="E" THEN GOTO 902
1010 IF X$="R" THEN GOTO 964
1012 IF X$="G" THEN GOTO 1020
1014 IF X$="DH" THEN GOTO 1024
1016 IF X$="HD" THEN GOTO 1070
1018 GOTO 1006
1020 SAVE "HEX"
1022 GOTO 988
1024 CLS
1026 PRINT AT 20,0;"DEC > ";
1028 INPUT W$
1030 IF W$="M" THEN GOTO 988
1032 IF W$="R" THEN GOTO 1066
1034 LET W=VAL W$
1036 IF W>65535 THEN GOTO 1028
```



```
1038 PRINT W,
1040 PRINT "HEX > ";
1042 LET K=INT (W/256)
1044 LET Y=W-(K*256)
1046 FOR T=1 TO 2
1048 LET H=INT (K/16)
1050 LET L=K-16*H
1052 LET A$=CHR$ (H+28)+CHR$ (L+
28)
1054 PRINT A$;" ";
1056 LET K=Y
1058 NEXT T
1060 PRINT
1062 SCROLL
1064 GOTO 1026
1066 LET X=X+N-1
1068 GOTO 908
1070 CLS
1072 PRINT AT 20,0;"HEX > ";
1074 INPUT X$
1076 IF X$="M" THEN GOTO 988
1078 IF X$="R" THEN GOTO 1066
1080 IF LEN X$<>4 THEN GOTO 1074
1082 LET AA=((CODE X$(1)-28)*16+
CODE X$(2)-28)*256
1084 LET BB=(CODE X$(3)-28)*16+C
ODE X$(4)-28
1086 PRINT X$( TO 2);" ";X$(3 TO
 );
1088 PRINT "DEC > ";
1090 PRINT AA+BB
1092 PRINT
1094 SCROLL
1096 GOTO 1072
```

1 REM XXXXX.....XXXXXX

Para colocar programas em código, da ordem de 1 Kbyte ou mais, na primeira linha do programa em Basic, a seguir de,por exemplo, uma instrução REM, não é nada "prático ou científico" digitarmos 1000 ou mais caracteres, a fim de reservar espaço. Isto é uma operação repetitiva que pode ser feita com o auxílio da máquina.

A seguir, um método para "criar" uma linha 1 REM com 4144 caracteres, ou mais, a partir da digitação de apenas 160.

Digite uma linha REM com 160 caracteres X. A seguir por EDIT 1, obtido digitando-se simultaneamente as teclas SHIFT 1 , a linha 1 descerá para o modo de edição. Troque o seu número por 2 e volte a editá-la , digitando New Line. Repita este tipo de operação para obter linhas iguais de número 3, 4 e 5, ficando então, cinco linhas com 160 caracteres cada.

O programa residente do microcomputador (gravado na ROM) coloca o seu programa em Basic, na RAM, nos endereços de 16509 em diante.Mais especificamente o endereço 16509 é o da primeira linha de programa em Basic. Esta linha ocupa a memória da seguinte forma:
Os dois primeiros bytes são usados para armazenar o número da linha. Os dois seguintes armazenam o comprimento desta linha em número de bytes próximo byte é obrigatoriamente uma instrução. Após o conteúdo da linha, a mesma é encerrada pelo código 118 que ocupa outro byte. É o código de New Line

Digite as demais linhas do programa:

```
  1 REM XXXXXXX (160 caracteres "X") XXXXXXX
  2 REM ídem linha 1
  3 REM ídem linha 1
  4 REM ídem linha 1
  5 REM ídem linha 1
  6 POKE 16511,58
  7 POKE 16512,3
  8 LET X=16507
 10 FOR A=1 TO 4
 12 LET X=X+166
 14 GOSUB 100
 16 NEXT A
 50 STOP
100 FOR N=X TO (X+10)
110 IF N=17338 THEN STOP
120 POKE N,61
130 NEXT N
140 RETURN
```

Antes de rodar, experimente se PRINT PEEK 17338 produz 118.
Caso negativo, é porque não existem as linhas de 1 a 5, com 160 caracteres cada. Reconfira a linha 1. 160 caracteres são exatamente 5 linhas, de caracteres. As linhas 2, 3, 4 e 5 não foram e não devem ser digitadas diretamente, caractere por caractere. Elas podem ser obtidas pela reedição da linha 1.

Rode o programa, acima, e voce passará a ter apenas uma única linha REM, com 824 caracteres "X".

O princípio adotado foi o seguinte:

Nos microcomputadores com Lógica Sinclair, os endereços da RAM, destinados ao usuário, iniciam-se em 16509. Uma linha de programa do tipo:

```
1 REM XXX
```

É arquivada na memória do microcomputador da seguinte forma:

ENDEREÇO	CÓD.	CONTEÚDO
16509	0	Número da linha. Byte mais significativo. X 256.
16510	1	Número da linha. Byte menos significativo.
16511	5	Comprimento da linha em bytes. Dígito menos significativo.
16512	0	Comprimento da linha em bytes. Dígito mais significativo. X 256.
16513	234	Instrução REM. Seu código é 234.
16514	61	Caractere X.
16515	61	Caractere X.
16516	61	Caractere X.
16517	118	New Line. É o código de término de linha.

Com a linha de programa acima na máquina, digite:

```
PRINT PEEK 16511 + 256 * PEEK 16512
```

Fornecerá 5 de resposta. A contagem foi de 3 caracteres, 1 instrução e 1 byte com New Line. Use esta mesma fórmula para obter o comprimento da linha 1 do programa anterior. Se realmente houver os 160 caracteres, voce obterá 162 de resposta.

O looping do último programa, linhas 100 a 130, coloca o código do caractere X, sobre os números de linha, comprimentos, instruções e New line, das linhas 2, 3 e 4. Por esta razão, aplicou-se as instruções das linhas 6 e 7, colocando nos endereços 16511 e 12 o novo comprimento da linha, 826 bytes.

No último programa,digitamos uma linha com 160 caracteres e, usando as facilidades de edição do micro, reeditamos esta linha, mais

quatro vezes. Por fim, transformamos as cinco linhas em uma única linha, com 824 caracteres que reservaram 824 bytes para suas rotinas em código. Caso isto já lhe seja suficiente, apague todas as linhas do programa, exceto a 1,e a 6. Veja a advertência abaixo.

Para reservar uma quantidade maior de endereços, o mesmo processo pode ser repetido e reservaremos até além do endereço 20000 .

Após rodar o programa da penúltima página,este fica sem as linhas 2, 3, 4 e 5, passando o conteúdo destas para a linha 1. A linha 1 passa a ter 824 caracteres. Reedite a mesma, com números 2, 3, 4 e 5, como feito anteriormente com 160 caracteres. A seguir, altere:

```
  6 POKE 16511,50
  7 POKE 16512,16
 12 LET X=X+830
110 IF N=20658 THEN STOP
```

Antes de rodar o programa, teste PRINT PEEK 20658. Deve resultar em 118, caso contrário, as linhas de 1 a 5 não estão corretas.Após rodar o programa, a linha 1 passará a ter 4144 caracteres " X", reservando os endereços de 16514 a 20757, inclusive.

ATENÇÃO:

A primeira linha de programa,após a "nova linha com os caracteres " X",NÃO pode ser apagada em hipótese alguma, sob pena de destruição do programa. Caso voce queira apagar as linhas de nº 6 a 140, digite antes :

```
2 REM
```

e não mexa mais com esta linha.

AS INSTRUÇÕES

AS INSTRUÇÕES

O microprocessador Z80 possui 158 diferentes tipos de instruções com mais de 600 variações.

Podemos agrupá-las assim:

1. LOAD E EXCHANGE

   Cópia e troca

2. LÓGICAS E ARITMÉTICAS

3. JUMP, CALL E RETURN

   Instruções de desvio do programa

4. BLOCK TRANSFER E SEARCH

   Transferência e pesquisa por blocos

5. SHIFT E ROTATE

   Substituição e rotação

6. MANIPULAÇÃO DE BITS

7. CONTROLE DA CPU

8. INPUT E OUTPUT

   Entrada e saída de dados com dispositivos externos.

O capítulo a seguir descreve as principais instruções do Z80.

Os mnemônicos, usados nos textos a seguir, seguem certas regras e convenções, conforme abaixo:

1. Os registers ficam denominados pelas suas iniciais em maiúsculas; por exemplo: A, B, AF, etc...

2. Uma deslocação d é positiva, se de valor entre 0 e 127 e negativa, se entre 128 e 255. O valor negativo é dado por 256-d. Desvio de endereço, além destes valores, não é permitido.

3. Números na faixa de 0 a 255 (1 byte) são representados por n ou N ou por XX, se em hexadecimal.

4. Os números ou endereços na faixa de 0 a 65 535 são representados em dois bytes, nn ou NN ou por XX XX, se em hexadecimal. O valor total representado é dado, multiplicando-se o conteúdo do segundo byte por 256 e somando-o ao conteúdo do primeiro.

5. Os parênteses significam endereços. Por exemplo: (nn) ou (BC) significam respectivamente o conteúdo do endereço dado por nn ou BC.

6. O destino do resultado de uma instrução é o indicado antes da vírgula. Por exemplo: ADD A,B significa adicione B com A, colocando o resultado em A. Ou ainda: ADD A com B .

NOP

No OPeration é uma instrução de não execução. Não faz absolutamente nada, exceto que o register PC, Program Counter, será incrementado em 1, passando o programa à instrução seguinte.

A instrução NOP é comumente empregada para:

1. "Apagar códigos" da memória.
2. Reservar memória, no meio de programas (em código), para alterações futuras.
3. Retardo de tempo, da ordem de 1,2 µsegundo.

O código de NOP é 00, zero.

HALT

HALT para a execução do programa e só reinicia por um reset ou interrupt enviado à CPU. Por questões de hardware, nos microcomputadores com lógica Sinclair, não é possível continuar um programa após a instrução HALT. Não a use em hipótese alguma.

O código de HALT é 78h, ou 118 decimal.

RET

RET é RETURN. Quando em funcionamento, o micro está rodando um programa no código da máquina. O programa monitor que "interpreta o Basic". A função do Basic USR é uma sub-rotina desta rotina principal. Portanto, toda rotina em código deverá ser terminada pela instrução RETURN, ao Basic. O código é C9.

ATENÇÃO

Não só a falta da instrução de retorno ao término das rotinas, bem como qualquer erro no programa, poderão levar a uma perda de controle do teclado e evidentemente do programa.

Nestes casos, a única solução é desligar e ligar o micro, recomeçando.

OBSERVAÇÃO

O Basic Sinclair não aceita funções como único argumento de um comando. É necessário o uso de uma instrução para a chamada de USR que é apenas uma FUNÇÃO. É válido: PRINT USR X, LET L=USR X, RAND USR X, RUN USR X , etc....

LOAD

Uma das instruções mais importantes e utilizadas é LOAD. Ela copia ou carrega o conteúdo de um register, de um endereço, ou um valor, em outro register ou endereço. Existem mais de cem variações, oferecendo as seguintes possibilidades:

1. LOAD DIRETO REGISTER OU MEMÓRIA, 8 BITS

   Um número n pode ser colocado diretamente em uma posição de memória (HL), (IX+d), (IY+d) ou em um dos registers A, B, C, D,E, H, L. Repetindo, os parênteses significam endereços. Por exemplo: (HL) significa o conteúdo do endereço dado por HL.

2. LOAD REGISTER PARA REGISTER, 8 BITS

   O conteúdo de qualquer um dos registers A, B, C, D, E, H, L pode ser copiado em um outro destes.

3. LOAD REGISTER PARA MEMÓRIA, 8 BITS

   Qualquer um dos registers A, B, C, D, E, H, L pode ser copiado em (HL), (IX+d), (IY+d). O register A pode ser copiado em (BC), (DE), (nn).

4. LOAD MEMÓRIA PARA REGISTER, 8 BITS

   O conteúdo (HL),(IX+d), (IY+d) pode ser copiado em qualquer um dos registers A, B, C, D, E, H, L ou ainda (BC), (DE) ou (nn)podem ser copiados no register A.

5. LOAD DIRETO REGISTER, 16 BITS

Um número nn pode ser carregado em qualquer um dos registers BC, DE, HL, IX, IY ou SP.

6. LOAD REGISTER PARA REGISTER, 16 BITS

O conteúdo de qualquer um dos registers HL, IX ou IY pode ser copiado no SP.

7. LOAD REGISTER PARA MEMÓRIA, 16 BITS

O conteúdo de qualquer um dos registers BC, DE, HL, IX, IY ou SP pode ser copiado em (nn).

8. LOAD MEMÓRIA PARA REGISTER, 16 BITS

O conteúdo (nn) pode ser copiado em qualquer um dos registers BC, DE, HL, IX, IY ou SP.

De uso pouco habitual, os registers especiais R (memory refresh register) e I (interrupt page address register) podem ser copiados no register A e vice-versa.

O efeito das instruções LOAD é mais ou menos igual à instrução LET em Basic. Exemplo: LOAD A,B seria LET A=B. A variável ou o register A assume o valor de B, sem que este seja alterado.

Para cada instrução LOAD existe um código diferente. Por exemplo: LD A,B é obtido pelo código 78 (hexadecimal). LD A,n é obtido com o código 3E XX, onde XX é o número n.

Carregue o programa HEXLOAD II e experimente:

```
16514     06 00        LD B,0        carregue B com 0
16516     0E 00        LD C,0        carregue C com 0
16518     C9           RET           retorne ao Basic
```


PRINT USR 16514 retornará o valor zero, contido em BC.

O mesmo efeito pode ser obtido por:

```
16514     01 00 00     LD BC,0       carregue BC com 0
16517     C9           RET           retorne
```

Para que voce se esqueça de qual register, em um register pair, contém o valor mais significativo, a ser multiplicado por 256, teste:

```
16514     01 01 00     LD BC,1       carregue BC com 1
16517     C9           RET           retorne
```

PRINT USR 16514 retornará o valor 1, no entanto:

```
16514     01 00 01     LD BC, 256    carregue BC com 256
16517     C9           RET           retorne
```

retornará o valor 256.

Tratando-se de constantes númericas, o micro interpreta o segundo byte, como sendo o mais significativo. Atenção para isto.

ARITMÉTICAS

As operações aritméticas oferecidas pelo microprocessador Z80 são basicamente de adição e subtração de números de 8 e 16 bits.

A ROM do seu microcomputador possui sub-rotinas aritméticas com ponto flutuante além de funções trigonométricas e científicas, as quais podem ser utilizadas como sub-rotinas de seus programas em código.

ADD

A instrução ADD permite:

1. ADIÇÃO DIRETA REGISTER PARA REGISTER, 8 BITS
   Permite adicionar ao register A qualquer um dos registers A, B, C, D, E, H, L ou uma constante numérica.

2. ADIÇÃO DA MEMÓRIA PARA REGISTER, 8 BITS
   Adicionar o conteúdo de (HL), (IX+d) ou (IY+d) ao register A.
   Repetindo: parênteses significam endereço(s).

3. ADIÇÃO DIRETA REGISTER PARA REGISTER, 16 BITS
   Permite adicionar qualquer um dos register par BC, DE, HL, SP, IX, IY ao register HL ou aos registers IX ou IY qualquer um dos registers BC, DE ou SP ou eles próprios.

As instruções ADD de 8 bits usam o register A para colocar o resultado .

Utilizando o programa HEXLOAD II, carregue os códigos abaixo que exemplificam o uso de ADD somando os valores 16514 e 33.

16514	11 82 40	LD DE, 16514	carregue DE
16517	21 21 00	LD HL, 33	carregue HL
16520	19	ADD HL,DE	adicione DE com HL
16521	44	LD B,H	transfere H para B
16522	4D	LD C,L	transfere L para C
16523	C9	RET	retorne ao Basic

Teste por PRINT USR 16514

O resultado da soma DE com HL contido no register HL foi transferido para o register BC, porque o comando em Basic PRINT USR (endereço) retorna com o valor contido no register BC. Não existem instruções ADD BC,REGISTER PAIR, também não existe LD BC,HL. Esta última equivale, no exemplo acima, à LD B,H e LD C,L .

Outro exemplo com ADD:

Determinar o primeiro e o último endereço utilizado pela memória de vídeo , com uma tela completa.

No manual das máquinas, no capítulo "organização da memória", consta que nos endereços 16396 e 16397 está armazenada uma variável do programa interno, denominada DFILE. É o endereço de início da área da memória usada para armazenar (temporariamente) os códigos dos caracteres mostrados no vídeo.
As instruções LOAD register pair, com o conteúdo de um endereço, como, por exemplo, LD DE,(16396), faz com que o conteúdo do endereço indicado e seu subsequente sejam copiados no register pair. Um endereço para cada register. Se não ficou claro: LD DE,(16396) faz com que o conteúdo do endereço 16396 seja copiado no register D e o conteúdo do endereço 16397 seja copiado no register E . Portanto:

Teste com o HEXLOAD II

```
16514       ED 5B 0C 40      LD DE,(16396)
16518       42               LD B,D
16519       4B               LD C,E
16520       C9               RET
```

PRINT USR 16514 produzirá o primeiro endereço da memória de vídeo.

Para determinar o último endereço de uma tela completa, basta considerar:

1. Cada linha de caracteres, no vídeo, ocupa 33 bytes, sendo 32 códigos de caracteres e um Next Line ao final.
2. Uma tela completa possui 24 linhas.

Portanto, uma tela completa ocupa 792 bytes de memória. (24 X 33).

Sem apagar a rotina anterior, carregue a segunda parte:

```
16521       ED 5B 0C 40      LD DE,(16396)
16525       21 17 03         LD HL,791
16528       19               ADD HL,DE
16529       44               LD B,H
16530       4D               LD C,L
16531       C9               RET
```

Para obter a resposta, acrescente:

```
2 PRINT USR 16514
3 PRINT USR 16521
```

Use RUN .

Altere as linhas 2 e 3 para:

```
2 LET A=USR 16514
3 LET B=USR 16521
4 PRINT B-A+1
```

Isto é uma forma de "aproveitar", no Basic, o(s) resultado(s) da(s) rotina(s) em código.

ADC

ADC é ADD WITH CARRY. Adicione com o carry. Carry é um dos bits do register F. Veja o capítulo "arquitetura do Z80".

A instrução ADC permite a execução de adição, somando ao resultado o carry-bit. Oferece as mesmas variações de ADD, excluindo-se as operações em IX e IY.

SUB

SUB subtrai, assim como ADD adiciona. Oferece apenas:

1. SUBTRAÇÃO DIRETA REGISTER PARA REGISTER, 8 BITS

   Qualquer um dos registers A, B, C, D, E, H, L ou ainda uma constante numérica pode ser subtraída no register A .

SBC

SBC subtrai com o carry-bit. Como ADC, só que subtraindo, ou, como **SUB**, incluindo o carry-bit na subtração.

Se antes de se usar a instrução SBC, o carry-bit for "zerado", esta fica igual a SUB e oferece as operações com registers pair.

Oferece as possibilidades:

1. SUBTRAÇÃO DIRETA REGISTER PARA REGISTER, 8 BITS

   Qualquer um dos registers A, B, C, D, E, H, L ou ainda uma constante numérica pode ser subtraída do conteúdo do register A, juntamente com o carry-bit.

2. SUBTRAÇÃO MEMÓRIA PARA REGISTER, 8 BITS

   O conteúdo de um endereço dado por (HL), (IX+d) ou (IY+d) pode ser subtraído do conteúdo do register A, juntamente com o carry-bit.

3. SUBTRAÇÃO DIRETA REGISTER PARA REGISTER, 16 BITS

   Qualquer um dos registers pair BC, DE, HL ou SP pode ser subtraído do register HL, juntamente com o carry-bit.

Zera-se o carry-bit com, por exemplo, ADD A,0

Um exemplo com SBC:

Determinar o número de bytes ocupados pelo programa em Basic.

Em Basic, PRINT PEEK 16396+256*PEEK 16397-16509 fornece este dado.

Em código, fica:

```
16514     11 7D 40        LD DE,16509
16517     2A 0C 40        LD HL,(16396)
16520     C6 00           ADD A,0
16522     ED 52           SBC HL,DE
16524     44              LD B,H
16525     4D              LD C,L
16526     C9              RET
```

Teste por PRINT USR 16514

Um exemplo com SUB:

Determinar a memória instalada.

Os endereços 16388 e 16389 contém a variável do sistema operacional, denominada RAMTOP. É o endereço de término da RAM. Como esta inicia-se no endereço 16384, basta subtrair 16384 da variável RAM TOP, como a seguir:

Carregue com o auxílio do programa HEXLOAD II .

```
16514    21 00 40       LD HL,16384      carregue HL
16517    ED 5B 04 40    LD DE,(16388)      ,, ramtop
16521    94             SUB H,D          subtraia o fim
16522    95             SUB L,E          da ram,do início
16523    44             LD B,H           transfira HL  para
16524    4D             LD C,L           BC
16525    C9             RET              retorne
```

PRINT USR 16514

Para melhor compreensão de que endereços devem ser subtraídos ou somados a outros, para se obter o consumo de memória, memória disponível, etc... veja o apêndice "organização da memória" que fornece a organização e distribuição da memória dos microcomputadores com lógica Sinclair.

INC INCREMENT
E DEC DECREMENT

As instruções INC e DEC desempenham operaçoẽs de aritmética elementar, porém são poderosas no Assembly Z80.

O valor contido em qualquer um dos registers A, B, C, D, E, H, L , BC, DE, HL, IX, IY, SP ou em (HL), (IX+d) ou (IY+d) pode ser somado em um pela respectiva instrução INC ou subtraído em um pela respectiva instrução DEC. Abaixo, exemplos:

```
16514    01 00 00       LD BC,0
16517    0C             INC C
16518    C9             RET
```

PRINT USR 16514 resultará em 1 .

A seguir, apresentamos um exemplo de soma.

Experimente com o auxílio do programa HEXLOAD


```
16514    01 80 00       LD BC,128        LOAD C com 128
16517    3E 7F          LD A,127         LOAD A com 127
16519    81             ADD A,C          some C com A
16520    4F             LD C,A           transfira A p/ C
16521    C9             RET              retorne
```

PRINT USR 16514 produzirá 255. O a soma de 127 e 128. O máximo valor admissível em um byte(de 8 bits). Experimente somar 128 com 128 em um único byte (de 8 bits), alterando o programa acima por:

POKE 16518,128

PRINT USR 16514 produzirá zero. Porque ? É simples.

Por exemplo: dispomos de apenas dois dígitos e vamos somar os valores 99 e 01, no sistema decimal. O que ocorre:

```
  99
+ 01
= 00
```

Nos dois dígitos ficaram 00. "E vai um" caiu em um dígito inexistente. Houve um "overflow" ou sobrecarga. A mesma coisa ocorreu com a soma em binário de 128 e 128, produzindo um número maior que o possível de ser representado em 8 bits.

EM BINÁRIO: 127 + 128 128 + 128

```
  0 1 1 1 1 1 1 1      1 0 0 0 0 0 0 0
+ 1 0 0 0 0 0 0 0    + 1 0 0 0 0 0 0 0
= 1 1 1 1 1 1 1 1    =1 0 0 0 0 0 0 0 0
```

Caso não tenha ficado claro, releia o capítulo "Números Binários e Hexadecimais".

A seguir, exemplo de operação aritmética em 16 bits.

Experimente,com o auxílio do programa HEXLOAD, a soma de 128 e 128, em register pair.

```
16514    01 80 00       LD BC,128        LOAD BC com 128
16517    21 80 00       LD HL,128        LOAD HL com 128
16520    09             ADD HL,BC        some BC com HL
16521    E5             PUSH HL          troque HL com
16522    C1             POP BC           BC
16523    C9             RET              retorne
```

PRINT USR 16514 produzirá 256.

Como voce vê, usando register pair, não há problema nas operações aritméticas, que envolvam resultados na faixa de 0 a 65535.

A sobrecarga de um ou mais bytes, em operações aritméticas, poderia ser amplamente vista, bem como o necessário uso das bandeiras carry e overflow, mas exigiria um considerável número de páginas a mais, além de alguns fundamentos mais sólidos em números binários e algebra de Boole, o que foge aos objetivos deste volume:

"CÓDIGO DE MÁQUINA PARA TK E CP 200"

Para máquinas com um "sofisticado sistema operacional", do qual podemos aproveitar rotinas extremamente úteis, como: decodificação do teclado, impressão de caracteres, memória do vídeo, operações aritméticas com ponto flutuante, etc...

Por esta razão, além de ser desnecessário inventar o já inventado, usaremos as sub-rotinas aritméticas da ROM, descritas no capítulo "Sub-rotinas da ROM".

O STACK, PUSH E POP

O stack é uma área não fixa na RAM, próxima ao final, onde o conteúdo de um register pair pode ser armazenado. O STACK funciona como uma pilha de fichas. Colocam-se as fichas umas sobre as outras, empilhando-as. Ao se retirar as fichas, começamos pelo topo da pilha, ou pela última que foi colocada lá.

O register SP, STACK POINTER mantém o endereço do topo do stack.

PUSH

PUSH copia o conteúdo de um register de 16 bits no STACK . Existem apenas seis possibilidades e são:
PUSH AF, BC, DE, HL, IX, IY

A instrução PUSH faz com que o register SP seja subtraído em 2 , para que o processador saiba o novo endereço para a próxima instrução a ser PUSHed. O byte mais significativo é copiado primeiro.

POP

É o inverso de PUSH. Ela copia dois bytes do topo do STACK para um dos registers AF, BC, DE, HL, IX e IY .

Exemplo de PUSH e POP

As principais instruções aritméticas de 16 bits colocam o resultato no register HL. Em alguns exemplos anteriores usamos as instruções LD B,H e LD C,L para colocar o resultado obtido em HL no register BC e retorná-lo ao Basic. Podemos obter o mesmo efeito de "transferência" com PUSH e POP, que ficaria assim:

```
PUSH HL     coloca no STACK o valor de HL
POP BC      retira o valor do STACK para BC
```

Experimente:

```
16514       C5                  PUSH BC     preserve BC
16515       01 00 00            LD BC,0     "zere" BC
16518       C1                  POP BC      reassume BC
16519       C9                  RET         retorne
```

Por PRINT USR16514, o valor 16514 é enviado ao register BC. Pela instrução PUSH BC o valor de BC é enviado ao STACK e por POP será retirado. Para provar isto, aplicou-se "zere" BC com LD BC,0.

Algumas sub-rotinas da ROM, destroem os valores de todos os registers, sendo, então, de grande valia PUSH e POP. Aplica-se, por ex.:

PUSH AF, PUSH BC, PUSH DE e PUSH HL antes da chamada da sub-rotina. Após o retorno, aplica-se POP HL, POP DE, POP BC e POP AF para a recuperação dos valores. Observe que o último register PUSHed deve ser o primeiro POPed.

COMPARE

Compare é o mesmo que SUB, subtraia, exceto pelo que o resultado não será colocado em parte alguma, permanecendo o register A inalterado. Os bits do register F serão alterados apropriadamente e poderão ser usados como condicional em uma instrução seguinte. Por exemplo : COMPARE 01 ou CP 01 seguido por JP Z, significa: compare ou subtraia o valor 01 do register A, se zero JUMP.

Todas as instruções CP envolvem o register A e oferecem:

1. COMPARAÇÃO DIRETA DE DADO COM O ACUMULADOR
   Um número N pode ser comparado com o register A
2. COMPARAÇÃO DE REGISTER PARA ACUMULADOR
   O conteúdo de qualquer um dos resgisters A B, C, D, E, H, L
3. COMPARAÇÃO DE MEMÓRIA COM ACUMULADOR
   O conteúdo de uma posição de memória dada por (HL), (IX+d) , ou (IY+d) pode ser comparado com o register A.

Todas as instruções CP referem-se ao register A e por isto os seus mnemônicos omitem a representação de "A".
Exemplo: CP B significa CP A,B assim como CP 01 é CP A,01 .

O Z80 oferece variações de COMPARE, acrescidas de DEC, INC, DEC e REPEAT e INC e REPEAT. São CPD, CPI, CPDR e CPIR e estão descritas no apêndice "Z80 - Sumário das Instruções".

JUMP, CALL E RETURN

JP JUMP significa "pule". É uma instrução similar a GOTO em Basic. JP 30000 em Basic seria GOTO 30000. Na instrução JP o número indicado a seguir da mesma não é número de linha e sim endereço. O microprocessador Z80 possui 16 linhas de endereço, atuando de 0 até 65536, porque 16 bits podem representar até $2^{16} = 65536$.
Uma instrução deste tipo custa 3 bytes. 1 pela instrução em si, seguindo-se de 2 para o endereço.

Existem instruções JP condicional, como também existe GOTO condicional em Basic. Por exemplo: IF A=1 THEN GOTO 1000 . As possibilidades de condicionais são as seguintes:

1. JUMP IF ZERO OU NOT ZERO

   JP Z e JP NZ significam JUMP, se o bit denominado zero-bit ou zero flag do register F for zero ou se for um .

2. JUMP IF CARRY OU NOT CARRY

   JP C e JP NC significam, respectivamente, JUMP se o carry-bit for um ou se, for zero.

3. JUMP IF POSITIVE OU MINUS

   JP P e JP M são condicionais dadas pela zero flag ou zero-bit do register F.

4. JUMP IF PARITY ODD OU PARITY EVEN

   JP PO e JP PE significam, respectivamente, JUMP, se a bandeira (flag) denominada parity/overflow for zero ou um.

Para uma perfeita compreensão do funcionamento das condicionais, além de outras instruções, é imprescendível um estudo do register F, denominado flag register, cujos bits são usados isoladamente, como indicadores de certas ocorrências no processador, em decorrência do seu funcionamento. Reveja o capítulo "arquitetura do Z80 - flag-register".

Assim como em Basic Sinclair é possivel GOTO XY, existem instruções JUMP aos endereços dados por HL, IX ou IY. São :
JP (HL), JP (IX) e JP (IY) e não existem com condicionais.

JUMP RELATIVE

JR JUMP RELATIVE é JUMP a um endereço 127 posições à frente do endereço em execução ou 128 posições atrás. Este deslocamento citado é máximo, podendo na verdade variar de -128 a +127. Estas instruções custam um byte a menos que as JP, porque o deslocamento (d) pode ser indicado em um único byte, ao invés de um endereço em dois bytes.

As condicionais oferecidas por JR limitam-se a quatro. São: Z, NZ, C e NC.

Carregue usando o programa HEXLOAD II

```
16514     0B          DEC BC
16515     C9          RET
```

PRINT USR 16514 produzirá 16513, porque o register BC recebe o endereço chamado por USR. Como BC só foi alterado por DEC, o resultado será 16513.

Agora experimente:

```
16514     C3 86 40     JP 16518      "pule" a end 16518
16517     0B           DEC BC        subtraia BC em 1
16518     C9           RET           retorne
```

PRINT USR 16514 produzirá 16514, porque JUMP 16518 fez com que o programa "pulasse" ao endereço 16518, evitando a instrução DEC BC.

Agora altere para:

```
16514     18 03        JR+3          "pule" 3 bytes
16516     0B           DEC BC        subtraia BC em 1
16517     C9           RET           retorne
16518     03           INC BC        some 1 em BC
16519     C9           RET           retorne
```

PRINT USR 16514 produz 16514, porque ?

Troque o valor do endereço 16515 por 02 e por 00 e verifique PRINT USR 16514. Para trocar o valor deste endereço, voce não precisa "rodar" o HEXLOAD. Use POKE 16515,2 e New Line. Idem para zero.

CALL

CALL significa, em Basic, GOSUB e funciona de forma semelhante. CALL chama uma sub-rotina no endereço dado e retorna por RETURN. Oferece as mesmas condições que JUMP.

1. CALL IF ZERO OU NOT ZERO

   CALL Z e CALL NZ

2. CALL IF CARRY OU NOT CARRY

   CALL C e CALL NC

3. CALL IF POSITIVE OU MINUS

   CALL P e CALL M

4. CALL IF PARITY ODD OU PARITY EVEN

   CALL PO e CALL PV

As instruções CALL usam o Stack para manter o endereço de retorno. Por esta razão o Stack não deve ser alterado, durante as sub-rotinas ou, pelo menos, deve voltar a situação original antes de RET.

Carregue com o HEXLOAD II

```
16514     CD 86 40     CALL 16518
16517     0B           DEC BC
16518     C9           RET
```

PRINT USR 16514 produzirá 16514 ou 16513 ?

A instrução CALL 16518 chama a sub-rotina do endereço 16518 e lá existe apenas RETURN. O programa retorna à instrução seguinte a CALL

de origem, DEC BC,executa-a e,em seguida,encontra novamente RET, retornando ao Basic.

Como no Basic podem existir "sub-rotinas dentro de sub-rotinas" e todas as sub-rotinas voltam por uma única instrução RETURN, em código também.

Sub-rotina da ROM

Existe na ROM uma sub-rotina denominada PRINT, que imprime no vídeo o caracter cujo código esteja no register A, quando a mesma for chamada. Seu endereço é 2056. Experimente:

```
16514     3E 97          LD A,151
16516     CD 08 08       CALL 2056
16519     C9             RET
```

Teste por PRINT USR 16514 e altere

```
16519     C3 82 40       JP 16514
```

PRINT USR 16514 produzirá uma tela cheia (instântâneamente) com asteriscos inversos.

Exemplo de condicional

Subtrair o valor do register BC, quando com 16514, em 50, sem usar SUB ou SBC.

```
16514     3E 00          LD A,0          carregue A com 0
16516     16 32          LD D,50         carregue D com 50
16518     0B             DEC BC          subtraia BC em 1
16519     15             DEC D           subtraia D em 1
16520     BA             CP D            compare D com A
16521     C2 86 40       JP NZ,16518     if not 0 JUMP
16524     C9             RET             retorne
```

Exemplo de RETURN condicional

Assim como existe JP condicional, existe RET condicional. Releia a instrução COMPARE e carregue o exemplo:

```
16514     3E 32          LD A,50         carregue A com 50
16516     16 32          LD D,50         carregue D com 50
16518     BA             CP D            compare D com A
16519     C8             RET Z           retorne se 0
16520     0B             DEC BC          subtraia BC em 1
16521     C9             RET             retorne
```

PRINT USR 16514 produz 16514, porque CP D compara D com A e o resultado será zero, ficando válida a expressão RET Z (retorne se zero). Altere o conteúdo do endereço 16515 para 51 por POKE 16515,51 e experimente PRINT USR 16514..

Produzirá 16513 porque RET Z não é mais válido e será executada a instrução DEC BC antes do RETURN do endereço 16521.

RESTART

RESTART ou RST é também uma instrução de desvio do programa, assim como CALL. Existem oito variações de RST. São:

RST 00	RST 20
RST 08	RST 28
RST 10	RST 30
RST 18	RST 38

As instruções RST ocupam apenas um único byte, porque não especificam o endereço a seguir de seu código, o qual já inclui este dado. RST 08 é mesmo que CALL 0008. Existem oito variações de RST que, portanto, só permitem acessar oito diferentes rotinas, com endereços na faixa de 0 a 38h.

Em seus programas, voce somente poderá usar RST para acesso às rotinas da ROM. Voce não poderá escrever suas próprias rotinas para serem acessadas por RST, simplesmente, porque voce não poderá colocá-las nos endereços entre 0 e 38 que pertencem à ROM.

O programa monitor do micro usa todas as oito rotinas de RESTART, com os seguintes propósitos:

RST 00	INÍCIO
RST 08	REPORTAGEM DE ERRO
RST 10	PRINT CARACTERE
RST 18	RECUPERA CARACTERE DE UMA LINHA EM Basic
RST 20	ÍDEM, PRÓXIMO CARACTERE
RST 28	ENTRADA PARA ROTINA DE CÁLCULOS
RST 30	RESERVA BC BYTES NA ÁREA DAS VARIÁVEIS
RST 38	INTERRUPÇÃO PARA UMA LINHA DE TV

DJNZ

O Z80 possui várias instruções que englobam o funcionamento de duas outras ou mais. DJNZ é uma delas e significa DEC B e JR NZ. Exatamente isto. DECREMENT B e JR IF NOT ZERO em uma única instrução. Seu código é 10 XX , onde XX é o deslocamento. DJNZ, primeiro subtrai o valor do register B em 1, para em seguida testar a condicional NZ. Portanto o deslocamento ou o JR condicional só será executado quando B for 1. Se B for 0, subtraído em 1, reinicia em 255. DJNZ não altera as bandeiras até que B atinja o valor zero.

Exemplo: Rotina para inversão do vídeo.

Todos os caracteres mostrados em uma tela podem ser invertidos pelo seguinte princípio: a cada caracter corresponde um código numérico entre 0 e 64. O código do caracter inverso é o do original somado em 128. Por exemplo:

código 0	caractere " "	espaço
código 128	caractere "■"	espaço inverso
código 29	caractere "1"	1
código 157	caractere "1"	1 inverso

A área da memória utilizada nas instruções PRINT inicia-se em um endereço que pode ser obtido pela variável do sistema operacional denominada DFILE e armazenada nos endereços 16396 e 16397. Em Basic, é dado por PRINT PEEK 16396 + 256*PEEK 16397. É neste endereço e seus subsequentes que estão armazenados os códigos dos caracteres mostrados no vídeo, durante as instruções PRINT do Basic.

Somando-se ou subtraindo-se 128 nos códigos destes endereços, estaremos invertendo ou desinvertendo os caracteres mostrados na tela. Para inverter a tela completa é preciso lembrar que a mesma possui 22 linhas e que cada linha possui 32 caracteres e ao término de cada linha o código 118, de New Line, o qual não deve ser invertido.

Um "Top Down" para este programa seria:

. Determinar o endereço de início do vídeo.

. Fixar 22 linhas a inverter.

. Somar um ao endereço do vídeo.

. Verificar o código do caracter, se diferente de 118, somar 128 para inverter ou desinverter. Repetir até completar 22 linhas.

A seguir, um programa em Basic, que exemplifica cada linha da rotina em código, apresentada mais à frente. Se voce não puder compreender o funcionamento disto em Basic, não o tente em código.

```
9010 LET B=22
9020 LET HL=PEEK 16396+256*PEEK 16397
9030 LET HL=HL+1
9040 LET A=PEEK HL
9050 LET X=A-118
9060 IF X=0 THEN GOTO 9100
9070 LET A=A+128
9075 IF A>255 THEN LET A=A-256
9080 POKE HL,A
9090 GOTO 9030
9100 LET B=B-1
9110 IF B<>0 THEN GOTO 9030
9120 RETURN
```

Para testar, acrescente esta outra rotina, que apenas escreve alguma coisa no vídeo e chama a anterior como uma sub-rotina.

```
5000 SLOW
5010 FOR N=1 TO 37
5020 PRINT "TESTE ";
5030 NEXT N
5040 GOSUB 9000
5050 INPUT U$
5060 GOTO 5040
```

Use RUN.

Para cada New Line digitado (linha 5050), o vídeo será invertido, ou desinvertido, de uma forma uma tanto lenta.

A seguir, a rotina em código, para inversão do vídeo. Carregue com a ajuda do programa HEXLOAD.

16514	06 16	LD B,22	define 22 linhas
16516	2A OC 40	LD HL,(16396)	define end. vídeo
16519	23	INC HL	HL=HL+1
16520	7E	LD A,(HL)	define caractere
16521	FE 76	CP 118	verifica New Line
16523	28 05	JR Z,5	se New Line JR+5
16525	C6 80	ADD A,128	inverte caractere
16527	77	LD (HL),A	carrega no vídeo
16528	18 F5	JR,-11	pula end.16519
16530	05	DEC B	B=B-1
16531	20 F2	JR NZ,-14	pula end.16519
16533	C9	RET	retorne

Para testar, inclua a rotina em Basic, acima, substituindo a linha 5040 por 5040 LET L=USR 16514.

Agora, substitua as instruções DEC B e JR NZ, desta rotina acima, por DJNZ. Ficaria assim:

16530	10 F3	DJNZ,-13	B=B-1. Se 0 JR.
16532	C9	RET	retorne.

LDD LDDR LDI LDIR

Estas instruções, são as que anteriormente me referi como de transferência de dados por blocos. Veja porque.

LDD

LDD é LOAD WITH DECREMENT. Existe em uma única variação e executa o equivalente a LD (DE),(HL) , seguido de DEC HL, DEC DE e DEC BC. O conteúdo do endereço dado pelo register HL é copiado no endereço dado pelo register DE. Os valores contidos em HL, DE e BC são ainda subtraídos em um.

LDDR

LDDR é LOAD WITH DECREMENT AND REPEAT. Existe em única variação. É a mesma que LDD, repetindo-se até que o valor do register BC seja zero. Produz um efeito semelhante ao "looping", em Basic. Por exemplo: suponhamos que queiramos transferir uma rotina em código, com 100 bytes, da linha 1 REM, para os endereços acima do RamTop. Ficaria na forma:

```
LD BC,100       LOAD BC com 100
LD DE,32        LOAD DE com 32
LD HL,16614     LOAD HL com 16614
LDDR            LOAD (DE),(HL) DEC
                HL,DE e BC. Repita
                até que BC seja 0
RET             Retorne
```


Evidentemente, esta rotina NÃO é para ser carregada nos endereços de 16514 a 16614, cujos conteúdos serão transferidos para os endereços 32000 a 32100. Para utilizar os endereços acima do RamTop e outras possibilidades,reveja o capítulo "Editando em Código".

LDI

LOAD WITH INCREMENT é o mesmo que LD (DE),(HL), seguindo-se de INC DE, INC HL e DEC BC.

LDIR

Igual a LDI, repetindo-se até que o valor de BC seja zero. É ainda a mesma coisa que LDDR, só que os valores de DE e HL serão somados em um, ao invés de subtraídos em um. Podemos usar LDIR no exemplo anterior, se alterarmos os endereços, como abaixo:

```
LD BC,100
LD DE,32000
LD HL,16514
LDIR
RET
```

Os códigos das instruções acima, bem como todos os do Z80,podem ser encontrados no final deste volume.

EX E EXX

EX é a abreviatura de EXCHANGE, troca. Oferece as variações:

1. EXCHANGE MEMÓRIA COM REGISTER

   O conteúdo de qualquer um dos registers IX, IY e HL pode ser trocado com o conteúdo de uma posição de memória endereçada pelo register SP, STACK POINTER.

2. EXCHANGE REGISTER COM REGISTER

   Existe apenas uma posibilidade. É EX DE,HL . Troca o conteúdo do register DE com o conteúdo do register HL.

3. EXCHANGE REGISTER PRINCIPAL COM SECUNDÁRIO

   Apenas uma possibilidade. EX AF,AF'

Veja uma aplicação de EXCHANGE no programa "BOMBARDEIO".

EXX é uma variação de EX, EXCHANGE. Não apresenta variações e sua única possibilidade é EXX. Esta instrução, de um único byte, troca o conteúdo dos registers BC DE e HL com, BC', DE' e HL'. Respectivamente B com B', C com C', D com D', etc...

LÓGICAS

O Z80 possui as instruções lógicas AND, OR e XOR, com as possibilidades a seguir:

O conteúdo dos registers A, B, C, D, E, H, L ou (HL),(IX+d),(IY+d), ou ainda uma constante númerica pode ser combinado com o register A por qualquer uma das tres operações lógicas:

1 AND

Coloca nível "1" em cada um dos bits do resultado, quando ambos os mesmos bits dos operandos forem "1".

2 OR

Coloca "1" em cada bit do resultado,se o mesmo bit de um ou de ambos os operandos forem "1" .

3 XOR

Coloca "1" em cada bit do resultado-se o mesmo bit de um operando ou do outro for "1", mas não no caso de ambos serem "1".

As instruções lógicas AND, OR e XOR alteram todas as bandeiras,sendo sempre o carry-bit "zerado".

ROTATE E SHIFT

O Z80 possui um grupo de instruções que permite trocar sequencialmente de posições os bits de um determinado byte. Para estudá-las é preciso estar familiarizado com os números binários. Por exemplo: ROTATE LEFT, uma variação de ROTATE, dobra o valor de número, sob certas circunstâncias. ROTATE RIGHT divide.

Veja porque:

0 0 1 0 1 0 1 0 , em binário, equivale a:

$0 + 0 + 2^5 + 0 + 2^3 + 0 + 2^1 + 0 = 32 + 8 + 2 = 42$ (decimal)

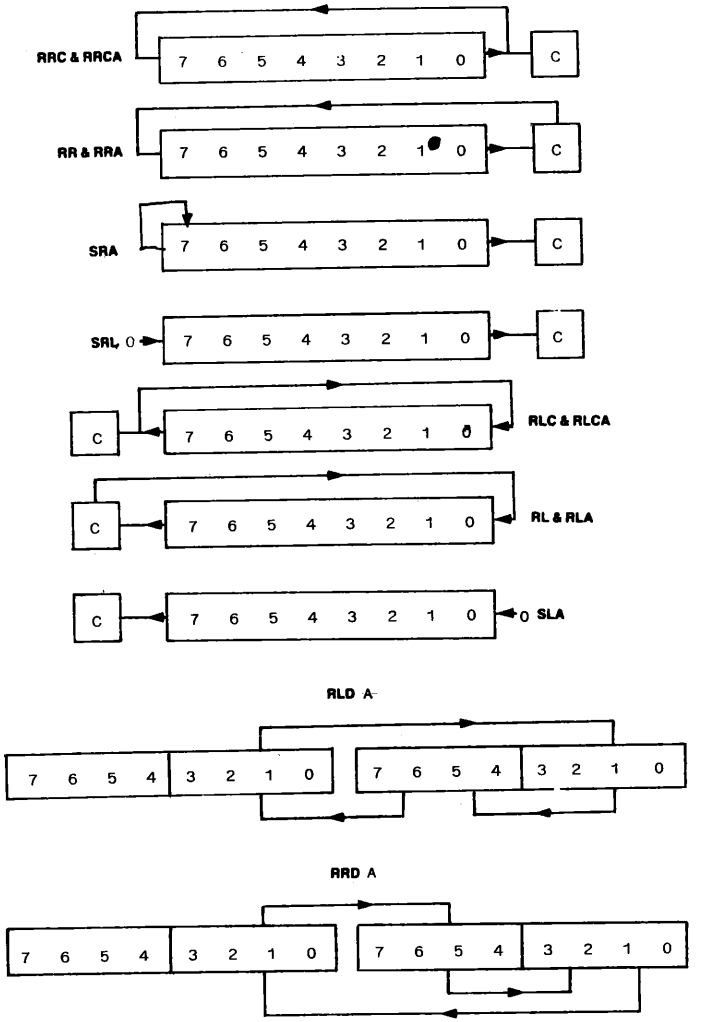
ROTATE LEFT move todos os bits uma posição à direita e o número acima ficaria:

0 1 0 1 0 1 0 0 , equivalendo a:

$0 + 2^6 + 0 + 2^4 + 0 + 2^2 + 0 + 0 = 64 + 16 + 4 = 84$ (decimal)

Todas as variações de ROTATE e SHIFT estão descritas no apêndice : "Z80 Sumário das Instruções".

Apresentamos na próxima página um diagrama com todas as possibilidades destas instruções. As setas indicam a direção de rotação e a letra "C" indica Carry Bit.



TIPOS DE ROTAÇÃO

IN E OUT

Até agora, as instruções do Z80, que vimos, manipulam dados apenas no próprio processador, ou dele para as memórias e vice-versa. Na verdade, o Z80 pode fazer muito mais que isto. É o caso das instruções de INPUT e OUTPUT que lhe permitem ler e enviar dados a dispositivos externos.

Um exemplo disto é o próprio programa monitor, onde a CPU lê dados do cassette e do teclado e envia para o monitor de vídeo , para o cassette ou para a impressora, usando IN e OUT.

Nas instruções IN e OUT,mediante um sinal, o processador copia o byte de dados, presente no barramento (de dados), bit por bit,para um determinado register. Nas instruções OUT o processo é inverso. Simultaneamente é colocado no barramento de endereço, um endereço especificado no programa . É o PORT ADDRESS. Permite ao dispositivo externo interpretar a qual dispositivo ou PORT deve enviar ou ler os dados.

Existem instruções IN e OUT acrescidas de INC, de INC e REPEAT, de DEC, de DEC e REPEAT. São INI, INIR, IND, INDR, OUTI, OUTIR, OUTD, e OUTDR.

O uso de IN e OUT está totalmente dependente do hardware da máquina em uso.

Todas as variações de IN e OUT estão descritas no apêndice "Z80 - Sumário das Instruções" e voce encontrará um exemplo no programa DATAFILE.

BIT TEST, SET E RESET

São instruções que permitem operações em um determindao BIT de um register ou posição de memória. As variações são:

1. BIT TEST

   Testa o valor de um BIT. O resultado é dado pelo zero-bit,do register F. A zero flag assume valor 1,se o bit testado é zero. E Ø,se o bit for 1.

2. BIT SET

   Permite tornar de valor 1, um determinado bit.

3. BIT RESET

   Permite tornar de valor Ø, um determinado bit.

E AS OUTRAS...?

Certamente,o Z80 possui muito mais instruções do que vimos. Não esqueça, voce encontrará um dicionario completo das instruções, nas últimas páginas deste volume.

OS PROGRAMAS

SCROLL HORIZONTAL

Aproveitando a experiência adquirida com os últimos exemplos, envolvendo impressão de caracteres, de cadeias de caracteres e outros com a memória de vídeo, apresentamos abaixo uma rotina para simular um SCROLL HORIZONTAL. A função SCROLL, existente no micro, apresenta apenas movimento da tela na vertical. Este é HORIZONTAL.

```
16514    3E 00        LD A,0              zere register A
16516    2A OC 40     LD HL,(16396)       LD HL com D.File
16519    06 1F        LD B,31             LD B c/ 31,colunas
16521    23           INC HL              HL=HL+1 p/ posição
16522    23           INC HL              HL=HL+1 p/ posição
16523    56           LD D,(HL)           LD D c/ caractere
16524    36 00        LD (HL),0           apaga caractere
16526    2B           DEC HL              HL-1 p/ pos. ant.
16527    72           LD (HL),D           PRINT caract.ant.
16528    10 F7        DJNZ,-9             BC=BC-1, é zero?
                                          não, JR próx.col.
16530    3C           INC A               A=A+1, nº da linha
16531    FE 16        CP 22               é a última linha?
16533    C8           RET Z               sim, retorne
16534    23           INC HL              HL=HL+1 p/ próx.
16535    23           INC HL              linha, pulando o 118
16536    18 ED        JR -19              JR próxima linha
```

Carregue com o HEXLOAD e imprima alguma coisa no vídeo, antes de chamar USR 16514. Para executar o SCROLL, em apenas uma parte do vídeo, altere o valor do endereço 16532 que é o limitador do número de linhas.

Podemos, ainda, executar o SCROLL HORIZONTAL, em uma faixa do vídeo, como, por exemplo, das linhas 5 a 10.

SOFTWARE PARA MICROCOMPUTADORES SINCLAIR

ZX SOFTWARE

OS QUATRO MELHORES PROGRAMAS EM CÓDIGO, JAMAIS PRODUZIDOS PARA UM SINCLAIR EM UM ÚNICO CASSETTE. POR APENAS Cr$ 15.000

Assembler

Escreva os seus programas em código, usando diretamente os mnemônicos do Z80 com este Editor e Monitor de Assembly. Interpreta todas as instruções do Z80. Oferece excelentes facilidades de edição, manipulação do cursor, códigos de erro, entrada de texto, números, labels, repetição de tecla, etc...

Disassembler

Lê códigos no Assembly do Z80. Fornece os endereços em decimal, os códigos em hexadecimal seguidos dos mnemônicos completos. Interpreta todas as instruções do Z80.

Compiler

Transforme instantaneamente os seus programas em Basic em programas em código, usando o Compiler. Aceita quase todos os comandos do Basic Sinclair e passa a rodar os programas até 50 vezes mais rápido.

Monitor

Programa para estudo de programas em código. Permite listar a Memória, Registers e Flags. Pode-se ainda introduzir Breakpoints, converter Hex para Decimal, decimal para Hex etc...

Distribuído por : ***MICRON ELETRÔNICA***

SAVE DISPLAY NO RAMTOP

A memória de vídeo pode ser preservada no RamTop com os seguintes propósitos e vantagens:

1. Armazenar várias imagens do vídeo nos endereços posteriores à variável RamTop e recuperá-las instantaneamente, seguidamente ou não. Por exemplo: no livro "30 JOGOS" existem quatro programas que geram quadros simétricos de arte moderna. O processo de geração é um tanto lento. Podemos gerar vários quadros, lentamente e arquivá-los acima do RamTop, instantaneamente, com uma pequena rotina em código. A recuperação da imagem será instantânea e poderá ser feita em um outro programa em Basic, mesmo se carregado do cassette.

2. O mesmo processo pode ser usado para passar dados de um programa a outro, simulando um "merging", se os mesmos tiverem sido impressos no vídeo.

Abaixo, a rotina SAVE DISPLAY FILE NO RAMTOP

16514	2A 0C 40	LD HL,(16396)	LD HL, end.D.File
16517	11 XX XX	LD BC,NN	LD BC, end.reservado no RamTop
16520	3E 16	LD A,22	LD A com o nº de linhas do vídeo.
16522	23	INC HL	HL=HL+1(próx.CHR$)
16523	01 2A 00	LD BC,32	LD BC c/o nº de caracteres por linha
16526	ED B0	LDIR	LD (DE),(HL). DEC DE HL e BC. Repita até BC=0

```
16528   3D          DEC A           A=A-1 p/próx.linha
16529   20 F7       JR NZ,-9        última linha ?
                                    não, JR próx.linha
16531   C9          RET             retorne
```

A seguir, a rotina LOAD RAMTOP NO DISPLAY FILE':

```
16540   ED 5B 0C 40   LD DE,(16396)   LD DE, end. D.File
16544   21 XX XX      LD HL,NN        LD HL, end reserva
                                      do acima do RamTop
16547   3E 16         LD A,22         LD A com o nº de li
                                      nhas do vídeo.
16549   13            INC DE          DE=DE+1(próx.CHR$)
16550   01 2A 00      LD BC,32        LD BC C/o nº de ca
                                      racteres por linha
16553   ED B0         LDIR            LD (DE),(HL). DEC
                                      DE HL e BC. Repita
                                      até BC=0
16555   3D            DEC A           DEC contador linha
16556   20 F7         JR NZ,-9        última linha ?
                                      não. JR próx.linha
16558   C9            RET             retorne
```

Carregue as rotinas com a ajuda do HEXLOAD.

Falta nas mesmas o endereço de início da área reservada para armazenar as imagens.A área reservada deve ser os últimos endereços da RAM. Basta alterar a variável do sistema operacional, denominada RamTop, para obter este efeito.

PRINT PEEK 16388 +256 * PEEK 16389 fornece o último endereço da memória disponível para o sistema Basic. Para alterar, use:

```
LET X=endereço desejado
POKE 16389, INT (X/256)
POKE 16388, X-(INT(X/256)
```

Veja mais sobre a variável RamTop no programa RENUMBER.

CONTADORES, DE PONTOS OU TEMPO

Apresentamos,a seguir, algumas rotinas para contar tempo ou pontos, em ordem crescente ou decrescente. Além do aspecto didático, servirão para implementar seus jogos, ainda que em Basic, dando um efeito especial devido à rápida mudança dos dígitos.

Começando do mais simples.

Uma instrução do tipo PRINT "9" faz com que o caractere 9 seja impresso no canto superior esquerdo da tela. Isto é o mesmo que colocar o código do caractere 9, no primeiro byte da memória de vídeo. Como já foi visto, a área da memória reservada para uso do vídeo, encontra-se em posição variável, na RAM. O endereço de início da mesma é dado pela variável do sistema operacional D.File e esta está armazenada nos endereços 16 396 e 16 397.

A rotina abaixo carrega no register HL o valor de D.File, contido nos endereços 16 396 e 7, soma 1 ao mesmo para obter a primeira posição. A seguir por LD A,(HL) , o conteúdo deste endereço, ou desta primeira posição da memória de vídeo,é copiado para o register A . Por LD C,A , o valor de A é transferido para C, para se poder obtê-lo, quandó do retorno ao Basic. Antes de usar o register C,foi preciso zerar BC, para evitar que ficasse algum valor remanescente em B. Carregue a rotina abaixo, com a ajuda do HEXLOAD.

```
16514   2A 0C 40      LH HL,(16396)     LOAD HL com D.File
16517   23            INC HL            some 1 a HL
16518   18 7E         LD A,(HL)         LOAD A com CHR$
```

```
16520   01 00 00        LD BC,0         zere BC
16523   4F              LD C,A          transfere C p/ A
16524   C9              RET             retorne
```

Para funcionar, acrescente:

```
2 PRINT "9"
3 PRINT USR 16514
```

RUN produzirá:

```
9
37
```

O 9 é da instrução PRINT "9" e o 37 é o código lido na memória de vídeo, pela rotina em código. 37 é o código do caractere 9. Os códigos de todos os caracteres de seu micro estão listados no respectivo manual. Altere a parte em Basic, deste último exemplo, para:

```
2 PRINT "9"
3 PRINT CHR$ (USR 16514)
```

Vamos fazer,a seguir, o número 9 impresso no vídeo ser subtraído em 1, consecutivamente, até atingir Ø .

```
16514   2A 0C 40        LD HL,(16396)   LOAD HL c/ D.File
16517   23              INC HL          1º end. de D.File
16518   7E              LD A,(HL)       LOAD A c/ CHR$
16519   FE 1C           CP 28           subtrai 28,CODE Ø
16521   C8              RET Z           se zero, retorne
16522   3D              DEC A           subtrai 1 do CHR$
16523   77              LD (HL),A       carrega no vídeo
16524   C3  86 40       JP 16518        repita do início
```

Acrescente:

```
2 PRINT "9"
3 LET L=USR 16514
```

Por RUN, será impresso o número 9, decrescendo até Ø, muito rapidamente.

Neste exemplo, o código do caractere da primeira posição do vídeo foi carregado no register A, em seguida subtraído em 1 e o novo valor recarregado na primeira posição do vídeo. A última instrução usada, JP 16518, faz com que o ciclo se repita, até que o valor do caractere ou o seu código seja 28. Isto é verificado pela instrução CP 28 que,se resultar em Ø, RET if ZERO, fará o retorno ao Basic. É bom lembrar que o código do caractere 9 é 37. O do 8 é 36,o do 7 é 35, e assim..,o 1 é 29, o Ø é 28, etc... Portanto, o nosso contador de 9 a Ø, na verdade conta de 37 a 28.

Para transformar esta rotina em um contador crescente, iniciando em Ø e acabando em 9, altere:

```
16519   FE 25           CP 37           subtrai 37,CODE 9
16521   3C              INC A           soma 1 no CODE CHR
```

```
2 PRINT "Ø"
3 LET L=USR 16514
```

Os contadores de 2 ou mais dígitos funcionam pelo mesmo princípio. Apenas, ao se esgotar um dígito, não é feito o retorno ao Basic e sim uma alteração no endereço, passando-se a um decréscimo no dígito seguinte. O endereço em uso volta ao original e volta-se a decrescer o dígito anterior, passando-se a um decréscimo no seguinte até que este atinja Ø. Volta-se ao endereço original e assim fica-se decrescendo, até uma condição de fim ou limitação da quantidade de algarismos em uso.

Ao invés de arrematar a rotina com uma instrução de repetição, vamos deixar que esta repetição se faça através de consetutivas chamadas do contador por USR 16514, o que permitirá pará-lo em Basic.

A seguir, um contador de 000000 até 999999,com opção de torná-lo decrescente,ou reduzí-lo a cinco algarismos,ou menos.

Carregue com o auxílio do HEXLOAD.

```
16514   2A 0C 40        LD HL,(16396)   LOAD HL c/ D.File
16517   11 07 00        LD DE,7         último nº na 7ªpos
16520   19              ADD HL,DE       posição últ. algar
16521   7E              LD A,(HL)       LOAD A c/ CODE CHR
16522   A7              AND A           é dígito excedente
16523   20 01           JR NZ,1         não, pule 1 byte
16525   C9              RET             retorne
16526   3C              INC A           some 1 ao CODE CHR
16527   FE 25           CP 37           é CODE CHR$ 9 ?
16529   20 05           JR NZ,5         não, pule 5 bytes
16531   36 1C           LD (HL),28      LOAD D.FILE c/ Ø
16533   2B              DEC HL          próximo dígito
16534   18 F1           JR -15          volte a DECREMENT
16536   77              LD (HL),A       LOAD D.File c/ CHR
16537   C9              RET             retorne
```

Acrescente:

```
2 PRINT "eØØØØØØ"
3 LET L=USR 16514
4 GOTO 3
```

A letra "e", na linha 2 acima, significa espaço e é obrigatório para a interpretação de "dígito excedente". A falta deste espaço destruirá o programa, ao final da contagem.

Para alterar a quantidade de algarismos do contador, altere a quantidade de algarismos da linha em Basic 2 PRINT "eØØ.. ØØ" para a quantidade pretendida. A seguir, digite POKE 16518,N , onde N é o número de algarismos pretendidos, mais um. O um é por conta do espaço.

DATAFILE

Dentro de seus preços e dimensões, os microcomputadores TK82, CP 200 e outros,produzidos à semelhança "de um Sinclair", são máquinas poderosas e com poucas limitações. Uma delas é o arquivo de dados. Nestes micros, as variáveis são gravadas para o cassette e recuperadas,juntamente com o programa. Como consequência,não podermos ter um arquivo de capacidade superior à RAM e nem utilizar dados de um programa em outro.

Várias soluções para este problema já foram pesquisadas e publicadas, cada uma com a suas vantagens e desvantagens.

Recentemente, fui procurado pelo Sr. Eng. Pedro Braga, o qual estava "às voltas" com este problema, procurando adaptar as soluções existentes às suas necessidades. Partindo de um farto material de pesquisa, desenvolvemos em conjunto, uma rotina para a transferência de dados, com as seguintes características:

1. Arquiva em cassette,uma matriz numérica ou alfanumérica de qualquer nome ou comprimento, mas apenas uma matriz.
2. Lê do cassette uma matriz anteriormente arquivada, desde que de mesmo comprimento.
3. Esta rotina pode ser usada em qualquer programa já existente, sem que tenha necessidade de se digitá-la em conjunto com este. Isto é obtido, colocando-se a mesma acima do RamTop, como nas fitas cassette FUNÇÕES da Micron Eletrônica.

A versão em cassete é comercializada diretamente pelo Eng. Braga.

DATAFILE

Para 16K de RAM. Carregue com o auxílio do HEXLOAD, com pelo menos 150 caracteres, na linha 1 REM .

16514	21 88 7F	LD HL,32650	define novo RamTop
16517	22 04 40	LD (16388),HL	altera var. RamTop
16520	01 64 00	LD BC,100	define nº de bytes
16523	11 94 7F	LD DE,32660	def.end.de destino
16526	21 A2 40	LD HL,16546	def.end.de origem
16529	ED B0	LDIR	transfere100 bytes
16531	C9	RET	retorna
16532	00 00 00	NOP	reserva espaço
16535	00 00 00	NOP	reserva espaço
16538	00 00 00	NOP	ídem
16541	00 00 00	NOP	ídem
16544	00 00	NOP	ídem
16546	CD E7 02	CALL FAST	CALL FAST
16549	01 FF FF	LD BC,65535	LOAD BC com 65535
16552	0B	DEC BC	BC=BC-1
16553	78	LD A,B	LOAD A com B
16554	FE 01	CP 1	compare 1 com A
16556	20 FA	JR NZ,-6	not zero,JR -6
16558	20 10 40	LD HL,(16400)	LOAD HL c/ end.var
16561	23	INC HL	HL=HL+1
16562	5E	LD E,(HL)	LOAD E com (HL)
16563	23	INC HL	HL=HL+1
16564	56	LD D,(HL)	LOAD D com (HL)
16565	23	INC HL	HL=HL+1
16566	D5	PUSH DE	preserve DE
16567	5E	LD E,(HL)	LOAD E com DATA
16568	CD 1F 03	CALL OUT REG	coloca E no K7
16571	D1	POP DE	reassume DE

16572	1B	DEC DE	próx. byte
16573	7A	LD A,D	LOAD A com D
16574	B3	OR E	compara D com E
16575	00	NOP	REM
16576	20 F3	JR NZ,-13	not zero, JR -13
16578	C9	RET	retorne
16579	00	NOP	REM
16580	CD E7 02	CALL FAST	CALL FAST
16583	2A 10 40	LD HL,(16400)	LOAD HL c/ end.var
16586	23	INC HL	HL=HL+1
16587	5E	LD E,(HL)	determina o compri
16588	23	INC HL	mento da variável
16589	56	LD D,(HL)	
16590	23	INC HL	
16591	D5	PUSH DE	preserve DE
16592	1E 08	LD E,8	LOAD E c/ 8 (bits)
16594	DB FE	IN A,254	LOAD A c/ porta 254
16596	D3 FF	OUT 255,A	LOAD porta 255 c/ A
16598	17	RLA	últ. bit?
16599	30 F9	JR NC,-7	not carry, JR -7
16601	OE 94	LD C,148	LOAD C com 148
16603	06 1A	LD B,26	LOAD B com 26
16605	0D	DEC C	decrement tempo
16606	DB FE	IN A,254	LOAD A c/ porta 254
16608	17	RLA	rotate left A
16609	CB 79	BIT 7,C	bit 7 do reg. C
16611	79	LD A,C	LOAD A com C
16612	38 F5	JR NC,-11	not carry,JR -11
16614	10 F5	DJNZ,-11	DEC B.Not zero,JR-
16616	20 04	JR NZ,+4	not zero, JR +4
16618	FE 56	CP 86	compare 86 c/ A
16620	30 E4	JR NC,-28	nor carry, JR -28
16622	3F	CCF	complement carry
16623	CB 16	RL,(HL)	rotate left (HL)
16625	1D	DEC E	decremente nº bit
16626	20 DE	JR NZ,-34	últ.? Não, JR -34
16628	D1	POP DE	preserve DE
16629	1B	DEC DE	decremente nº byte
16630	7A	LD A,D	LOAD A com D
16631	B3	OR E	compare E com D
16532	00	NOP	REM
16533	C8	RET Z	if zero, retorne
16534	18 D2	JR -46	não,JR próx.byte

Este programa está basicamente constituído de tres principais rotinas, as quais denominei como abaixo e que se aplicam a:

RAMTOP TRANSFER USR 16514

Transfere 100 bytes de códigos, dos endereços 16546 a 16645 para os endereços 32660 a 32759. Com 16K de RAM instalada, a variável do sistema operacional, RamTop, é convenientemnte reduzida ou "baixada" em 118 (bytes), passando de 32768 para 32650.

DATASAVE USR 16546/32660

Arquiva, em cassette, a primeira matriz dimensionada em um programa em Basic, independente de tamanho ou nome, quer ela seja numérica ou alfanumérica.

DATALOAD USR 16580/32694

Lê, do cassette, uma matriz anteriormente arquivada, desde que esta tenha o mesmo comprimento da matriz dimensionada para carga, mas não necessariamente o mesmo nome.

Após carregar e conferir os códigos, apague todas as linhas do HEX LOAD, excluindo a linha 1 e acrescente:

```
2 SAVE "DATAFILE"
3 FAST
4 LET L=USR 16514
5 PRINT,,"DATAFILE"
6 PRINT,,"DATASAVE USR 32660"
7 PRINT"DATALOAD USR 32694"
8 PRINT,, "DIGITE NEW"
9 LET L=USR 681
10 NEW
```

Cónfira e grave por GOTO 1 ou RUN. Quando terminar a gravação, ou a mesma for recuperada, o programa sairá rodando com as instruções de uso no vídeo. Digite New.

As rotinas DATASAVE e DATALOAD são transferidas para o RamTop pelo comando LET L=USR 16514, de chamada da rotina RAMTOP TRANSFER. Desta forma, aparentemente, não existe qualquer programa na memória, e voce poderá carregar qualquer um, via teclado ou cassette, para efetuar a transferência de dados para o cassette, com as rotinas DATASAVE e DATALOAD.

Abaixo, exemplo para gravar dados:

```
2 DIM X(300)
3 FOR N=1 TO 300
4 LET X(N)=N
5 NEXT N
6 LET L=USR 32660
```

E para leitura de dados:

```
2 DIM B(300)
3 LET L=USR 32694
4 FOR N=1 TO 300
5 PRINT B(N)
6 NEXT N
```

O dimensionamento da matriz a ser transferida deve ser feito como primeira instrução do programa. Matrizes com menos de 100 elementos apresentam a transferência em menos de 1 segundo.

RENUMBER

Apresentamos,a seguir, um renumerador de linhas de programa para 2 Kbytes de RAM, em uma sub-rotina no código da máquina que ocupa apenas 40 bytes, protegidos contra NEW, SAVE e LIST. Este renumerador pode ser aplicado em programas carregados do cassette, sem que o mesmo seja apagado. Estando no RamTop, fica também protegido contra LOAD.

NÃO É o mesmo programa comercializado pela Micron Eletrônica Com.e Ind. Ltda. no cassette denominado FUNÇÕES I (16K), o qual ocupa cerca de 400 bytes de rotinas, também em código, renumerando inclusive as imstruções GOTO e GOSUB. Contudo a principal técnica de proteção a SAVE, NEW, LIST e LOAD é a mesma aqui descrita.

A listagem:

16625	11 7C 40	LD DE,16508	LOAD DE c/ início
16628	21 10 00	LD HL,10	LD HL c/nº1ªlinha
16531	1A	LD A,(DE)	LD A c/ posição
16532	3D	DEC A	A=A-1
16533	FE 75	CP 117	é New Line ?
16535	C0	RET NZ	não, continue
16536	13	INC DE	DE=DE+1
16537	1A	LD A,(DE)	LD A
16538	FE 27	CP 39	é o fim do prog.?
16540	D0	RET NC	sim, retorne
16541	01 0A 00	LD BC,10	LD BC c/line step
16544	09	ADD HL,BC	últ. nº linha + 10
16545	EB	EX DE,HL	troca HL c/ DE
16546	72	LD (HL),D	LOAD novo nº de
16547	23	INC HL	linha
16548	73	LD (HL),E	ídem

16549	23	INC HL	
16550	4E	LD C,(HL)	
16551	23	INC HL	
16552	46	LD B,(HL)	
16553	09	ADD HL,BC	
16554	EB	EX DE,HL	
16555	18 E6	JR -26	
16557	3E 76	LD A,118	
16559	32 7C 40	LD (16508),A	
16662	C3 00 71	JP 18176	

Para entrar os códigos listados, digite a parte em Basic abaixo e use RUN 1000. Antes, coloque na linha 2 REM pelo menos 40 caracteres quaisquer.

```
   1 SAVE "RE■"
   2 REM )?RND5■  X RETURN ?""
 RETURN 550R ■■ ■ FOR??????■ F
OR ; NEW Y
C924 ?
 120 FAST
 130 LET X=(PEEK 16388+256*PEEK
16389)-256
 140 LET K=PEEK (16389)-1
 150 POKE 16389, K
 160 FOR N=1 TO 40
 170 POKE (X+N),PEEK (16524+N)
 175 IF N=32 THEN LET X=X+92
 180 NEXT N
 185 PRINT "■ENUMERADOR","P/ 2K
RAM"
 190 PRINT "ESTE PROGRAMA,"
 200 PRINT "NO CODIGO DA MAQUINA
,"
 210 PRINT "EM UM RENUMERADOR DE
 LINHAS."
 220 PRINT
 230 PRINT "DIGITE NEW E O SEU P
ROGRAMA,"
 235 PRINT "OU CARREGUE-O DO CAS
SETTE."
 237 PRINT "SUA RAM SERAH 2K - 2
56 BYTES."
 240 PRINT
 250 PRINT "PARA RENUMERAR AS LI
NHAS."
 260 PRINT "DE 10 EM 10, DIGITE
DIRETO :"
 270 PRINT "LET L=USR 18300"
 272 PRINT "PARA RENUMERAR AS LI
NHAS,"
 274 PRINT "DE 1 EM 1, DIGITE AN
TES "
 276 PRINT "POKE 18194,1"
 280 PRINT
 290 PRINT "ESTE PROGRAMA ESTA P
ROTEGIDO"
 300 PRINT "CONTRA SAVE, LIST,NE
W,ETC..."
```

```
  305 PRINT
  310 PRINT "PROIBIDO REPRODUZIR"
  320 PRINT "TODOS OS DIREITOS RE
SERVADOS"
  340 PRINT "MICRON ELETRONICA"
  350 LET P=USR 681
  360 NEW
 1000 FOR X=1 TO 40
 1030 PRINT AT 20,0;"ENDERECO ";1
 6524+ X;"CODIGO: ";
 1040 INPUT C
 1050 PRINT C
 1060 POKE 16524+X,C
 1070 SCROLL
 1080 NEXT X
```

NÃO RODE o programa. Confira-o e grave-o pela instrução GOTO 1. Após a gravação, ou, quando carregado do cassette, o programa sairá rodando, com as instruções de uso no vídeo. Digite NEW e teste o renumerador, conforme as instruções. Só funcionará para 2 K de RAM.

Descrição do programa. Parte em Basic:

A linha 01 grava o programa e faz com que o mesmo "saia rodando", após a leitura do cassette. A linha 02 contém 40 bytes com códigos. As linhas 130 a 150 reduzem o RamTop em 256 bytes. As linhas 160 a 180 transferem os 40 bytes de códigos da linha 2 REM para os endereços acima do RamTop. As linhas 185 a 340 apresentam no vídeo uma descrição de uso do programa. A linha 350 faz com que o programa pare, sem a mensagem de erro ou cursor no vídeo, se no modo FAST. A linha 360 limpa a memória baixa.

MERGING PROGRAMAS

Sir Clive Sinclair não colocou nos micros ZX o comando MERGE, do Basic, mas nós podemos "dar um jeito" nisto. O programa RamTop, comercializado pela Micron Eletrônica, em cassette e publicado no livro APLICAÇÕES SÉRIAS é um exemplo. A seguir, descrevemos outras rotinas para este fim que facilmente serão adaptadas às suas necessidades. O procedimento a ser adotado é:

1. Reservar uma área de memória independente do sistema Basic, alterando a variável RamTop, end. 16388/9 .
2. Transferir um programa em Basic para a área protegida.
3. Carregar um segundo programa em Basic, via teclado ou cassette.
4. Transferir o primeiro programa, colocado no RamTop, para o sistema em Basic.

A rotina a seguir, copia um programa em Basic com as suas variáveis, nos endereços da área reservada, acima do RamTop. Voce deve reduzir a variável RamTop tantos bytes quanto tiver o seu programa em Basic, com as variáveis e mais 60 bytes, consumidos por estas rotinas que também ficarão no RamTop.

PRINT (PEEK 16396+256*PEEK 16397)-16509 fornece o número de bytes usados pelo seu programa, incluindo as variáveis.

16514	11 7D 40	LD DE,16509	LD DE end.programa
16517	2A 0C 40	LD HL,(16396)	LD HL c/ D.File
16520	B7	OR A	zere carry flag
16521	ED 52	SBC HL,DE	HL=bytes de progr.
16523	44 4D	LD B,H	transfira HL para
16525	4D	LD C,L	BC

16526	2A 04 40	LD HL,(16388)	LD HL,com o endereço da área reservada.variável RamTop
16529	71	LD (HL),C	transfira BC para
16530	23	INC HL	área reservada
16531	70	LD (HL),B	
16532	23	INC HL	
16533	36 76	LD (HL),118	ídem,código 118
16534	23	INC HL	próx. end.
16535	ED	EX DE,HL	troque HL com DE
16536	ED B0	LDIR	LOAD (DE) com (HL) INC DE e HL.DEC BC
16538	C9	RET	repita até BC=0

Esta outra rotina executà o inverso da primeira. Retira o programa armazenado na área reservada acima do RamTop, juntamente com as variáveis e coloca-os no sistema Basic. Neste processo é utilizada a sub-rotina da ROM denominada "MAKE ROOM" que cria BC espaços na memória, a partir do endereço dado por HL. Carregue com o HEXLOAD.

16544	ED 4B 04 40	LD BC,(16388)	LD BC c/ nº bytes do programa
16548	2A 0C 40.	LD HL,(16396)	LD HL c/ D.File
16551	2B	DEC HL	
16552	7F	PUSH BC	preserve BC
16553	E5	PUSH HL	preserve HL
16554	CD 93 09	CALL 2562	CALL MAKE ROOM
16557	D1	POP DE	recupera HL em DE
16558	C1	POP BC	recupera BC
16559	2A 04 40	LD HL,(16388)	LOAD HL c/ RamTop
16562	23 23 23	INC INC INC HL	HL=HL+3
16565	ED B0	LDIR	vide acima
16567	C9	RET	retorne

Estas rotinas não poderão ser usadas ou,mesmo testadas, nos endereços em que se encontram. Elas foram e devem ser colocadas inicialmente nestes endereços, posteriores a instrução REM de uma primeira linha de programa, para poderem ser gravadas. Coloque-as também nos endereços do RamTop, preferivelmente os últimos.

LABIRINTO

Programa para 2K de RAM com SLOW.

O objetivo é atravessar o labirinto no menor espaço de tempo possível. A contagem é regressiva iniciando-se em 99900. A direção de deslocamento é dada pelas teclas:

1 2 3 4 5 6 7 8 9 0, para cima

Q W E R T A S D F G, para esquerda

Y U I O P H J K L NL, para direita

Z X C V B N M . SPACE , SHIFT, para baixo.

Carregue os códigos listados com a ajuda do HEXLOAD, com pelo menos 480 caracteres na linha 1 REM . Não estão listados os códigos dos endereços 16668 a 16999 que se destinam a reservar memória para o caso de desejarmos expandir o tabuleiro do labirinto.

Para jogar acrescente:

```
2 POKE 16389,74
3 SLOW
4 LET L=USR 16524
```

LABIRINTO

Para 2K de RAM com SLOW. Carregue com o auxílio do HEXLOAD, com pe
lo menos 480 caracteres, na linha 1 REM.

```
16514   D1              POP DE
16515   1A              LD A,(DE)
16516   13              INC DE
16517   D5              PUSH DE
16518   FE FF           CP FF
16520   C8              RET Z
16521   D7              RST 10
16522   18 F6           JR -10
16524   CD 82 40        CALL 16514

16527   80 80 80 80 B2 AE A8 B7 B4 B3 80 80 80 76
16541   80 15 00 00 00 00 00 00 00 00 00 00 80 76
16555   80 80 80 80 80 80 80 80 80 80 80 00 80 76
16569   80 00 00 00 00 00 00 00 00 00 00 00 80 76
16583   80 00 80 80 80 80 80 80 80 80 80 80 80 76
16597   80 00 80 00 00 00 80 00 00 00 80 00 00 76
16611   80 00 80 00 80 00 80 00 80 00 80 00 80 76
16625   80 00 80 00 80 00 80 00 80 00 80 00 80 76
16639   80 00 00 00 80 00 00 00 80 00 00 00 80 76
16653   80 80 80 80 80 80 80 80 80 80 80 80 80 76
16667   76
16668   00 00 00 00 25 25 25 1C 1C FF

16800   2A OC 40        LD HL,(16396)
16803   01 10 00        LD BC,16
16806   09              ADD HL,BC
16807   22 40 41        LD (16704),HL
16810   21 00 00        LD HL,0
16813   22 42 41        LD (16706),HL
16816   2A OC 40        LD HL,(16396)
16819   11 94 00        LD DE,148
```



```
16822   19              ADD HL,DE
16823   7E              LD A,(HL)
16824   A7              AND A
16825   20 08           JR NZ,+8
16827   06 01           LD B,1
16829   78              LD A,B
16830   FE 01           CP 1
16832   C8              RET Z
16833   18 F9           JR -7
16835   3D              DEC A
16836   FE 1B           CP 27
16838   20 05           JR NZ,+5
16840   36 25           LD (HL),37
16842   2B              DEC HL
16843   18 EA           JR -22
16845   77              LD (HL),A
16846   11 00 03        LD DE,3
16849   1B              DEC DE
16850   7A              LD A,D
16851   B3              OR E
16852   20 FB           JR NZ,-5
16854   CD BB 02        CALL 699
16857   7D              LD A,L
16858   2F              CPL
16859   6F              LD L,A
16860   E6 81           AND 81
16862   28 05           JR Z,+5
16864   01 0E 00        LD BC,14
16867   18 1C           JR +28
16869   7D              LD A,L
16870   E6 60           AND 96
16872   28 05           JR Z,+5
16874   01 01 00        LD BC,1
16877   18 12           JR +18
16879   7D              LD A,L
16880   E6 18           AND 24
16882   28 05           JR Z,+5
16884   01 F2 FF        LD BC,-14
16887   18 08           JR +8
16889   7D              LD A,L
16890   E6 06           AND 6
16892   28 B2           JR Z,-78
16894   01 FF FF        LD BC,65535
16897   2A 42 41        LD HL,16706
16900   7D              LD A,L
```

```
16901   B4              OR H
16902   28 07           JR Z,+7
16904   09              ADD HL,BC
16905   7D              LD A,L
16906   B4              OR H
16907   28 02           JR Z,+2
16909   18 A1           JR -95
16911   2A 40 41        LD HL,(16704)
16914   7E              LD A,(HL)
16915   E6 80           AND 128
16917   77              LD (HL),A
16918   09              ADD HL,DE
16919   7E              LD A,(HL)
16920   F6 15           OR 21
16922   77              LD (HL),A
16923   22 40 41        LD (16704)
16926   21 00 00        LD HL,0
16929   17              RLA
16930   30 02           JR NC,+2
16932   60              LD H,B
16933   69              LD L,C
16934   22 42 41        LD (16706),HL
16937   2A 0C 40        LD HL,(16396)
16940   01 53 00        LD BC,83
16943   09              ADD HL,BC
16944   ED 4B 40 41     LD BC,(16704)
16948   ED 42           SBC HL,BC
16950   C2 B0 41        JP NZ,16816
16953   C9              RET
```


SOM POR SOFTWARE

Os microcomputadores baseados no Sinclair ZX80 e ZX81 não possuem a função SOM, como algumas outras máquinas. Contudo, com alguma engenhosidade, é possivel obtermos diretamente no teclado do micro, uma oitava completa, com bemois e sustenidos, armazenar as últimas cem notas tocadas e reproduzí-las em tres diferentes velocidades no próprio altofalante da TV.

O micro, na verdade, produz som,quando leva o programa e/ou os dados para o cassette, ainda que um tanto alienígena. Estes sinais elétricos audíveis podem ser ajeitados para simularem as notas musicais. Apenas, ao invés de mandarmos ao cassette um programa, mandamos repetitivamente um valor X, com uma frequência Y. Diferentes valores e diferentes frequências geram diferentes notas.

Com menos de 1 Kbyte de RAM e, ocupando apenas 180 bytes de códigos, podemos obter o efeito sonoro por um dos modos abaixo:

A. Com o micro ligado à TV pela antena, simplesmente aumente o volume da mesma e ajuste a sintonia fina. O melhor ponto de sintonia para o som não corresponde ao melhor ponto para o vídeo. É uma característica dos circuitos de TV.

B. Ligue a saída MIC do computador à entrada de um amplificador de aúdio, ou,à entrada MIC de um cassette, ligado no modo PLAY sem fita,ou,REC com uma fita para gravação.

C. Aproxime do computador um rádio FM sintonizado entre 102 e 106 Mhz.

Carregue o HEXLOAD com,pelo menos,180 caracteres na linha 2 REM e acrescente o programa abaixo, destinado a ler o teclado, armazenar as notas e apresentar o menu.

```
   1 REM XXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  30 LET K$=INKEY$
  35 REM MICRON ELETRONICA
  40 CLS
  46 LET K=0
  50 PRINT "SOM POR SOFTWARE"
  51 PRINT ,,"ESCOLHA A VELOCIDA
DE"
  52 PRINT ,,"PARA REPRODUZIR",,
,,"RAPIDO DIGITE","0","MODERADO"
;"1","LENTO","2"
  53 PRINT ,,,,,,,,,,
  54 PRINT "PARA TOCAR",,"DIGITE
","5"
  55 PRINT
  56 PRINT "USE TECLAS"
  57 PRINT "  2     4    5    6
8    9"
  58 PRINT "0    WE    R    T    YU
I"
  59 INPUT X
  60 IF X=5 THEN GOTO 106
  65 POKE 16524,PEEK (16585+K)
  70 LET K=K+1
  80 IF K=99 THEN LET K=0
  90 LET A=USR (16514)
  95 FOR I=0 TO (X*50)
  96 NEXT I
  97 IF INKEY$="M" THEN GOTO 46
 100 GOTO 65
 106 LET K=0
 110 IF INKEY$<>"" THEN GOTO 110
 120 IF INKEY$="" THEN GOTO 120
 121 IF INKEY$="M" THEN GOTO 46
 122 LET K$=INKEY$
 123 IF CODE K$=0 THEN GOTO 110
 125 POKE 16524,CODE K$
 126 POKE 16586+K,CODE K$
 127 LET K=K+1
 128 LET K=K-(K=100)
 130 LET A=USR (16514)
 140 GOTO 110
 196 REM
 199 REM
 200 SAVE "SOM"
 210 RUN
```

Apresentação do menu:


```
SOM POR SOFTWARE

ESCOLHA A VELOCIDADE

PARA REPRODUZIR

RAPIDO DIGITE    0
MODERADO         1
LENTO            2



PARA TOCAR
DIGITE           5

USE TECLAS
  2     4    5    6    8    9
0    WE    R    T    YU    I
```

A rotina em código, na verdade, ocupa apenas 35 bytes, sendo os demais 145 usados para armazenarem os códigos das teclas e as notas tocadas. Voce precisará entrar com o HEXLOAD apenas a rotina em si e os códigos das teclas, listados a seguir:

16514	00 00	NOP NOP	
16516	C5	PUSH BC	faça HL=BC
16517	E1	POP HL	
16518	16 00	LD D,0	zere D
16520	1E 24	LD E,36	E=36
16522	19	ADD HL,DE	HL=HL+DE = 16540
16523	3E 37	LD A,55	LD A com nota
16525	01 24 00	LD BC,36	BC=36(p/ looping)
16528	ED B1	CPIR	Compare (HL) com A INC HL e DEC BC . Repita até BC=0
16530	C0	RET NZ	tecla válida ? não, retorne
16531	0E FF	LD C,255	C=255
16533	56	LD D,(HL)	LD D com nota

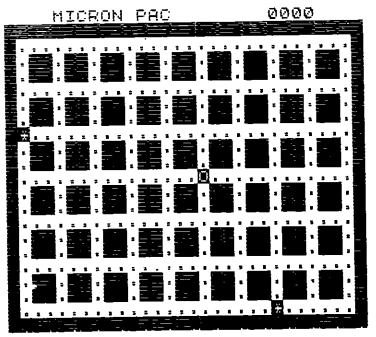
```
16534   42          LD B,D          LD B p/ looping 2
16535   D3 FF       OUT (255),A     envie A/ p/ K7
16537   10 FE       DJNZ -2         DEC B, NZ JR-2
16539   42          LD B,D          LD B p/ looping 3
16540   ED 40       IN B,(C)        envie sombra TV
16542   10 FE       DJNZ -2         DEC B, NZ JR-2
16544   0D          DEC C
16545   20 F3       JR NZ,-13       zero ? não JR loop2
16547   C9          RET             retorne

16548   00 00 00 36 E0              tabela com os códi
16553   3C 4B 2A 9D 37              gos das teclas vá
16558   7E 39 5D 3F 41              lidas como notas
16663   3A 32 2E 16 00              é interpretada du
16668   00 1E CA 20 8C              rante a instrução
16673   21 6C 22 4F 24              CPIR
16678   28 25 01 00 00
16683   00 00 76 36 3C

16688                               códigos das últi
                                    mas 100 notas toca
                                    das
                                    não precisam ser
                                    digitados, desde
                                    que os endereços
                                    tenham sidos reser
                                    dos
16693
```

Este programa produz uma escala musical de razoável qualidade.Caso voce não esteja obtendo som de boa qualidade, experimente:

1. Durante o menu, optar para tocar, 5
2. Volte ao menu, digitando M
3. Acione reprodução rápida ou normal, 0
4. Grave, no cassete

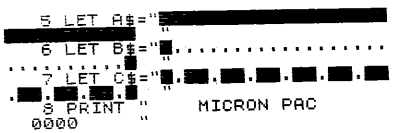
MICRON PAC

Com tanto PAC MAN, TK MAN, PUNCK MAN, MAG PAC, PAC MIC e outros do tipo por aí, não poderíamos deixar de apresentar uma versão do "famoso PAC MAN". O programa listado a seguir requer 16K de RAM e modo SLOW. Abaixo, o tabuleiro do jogo:



Voce é o ▣ . Para fugir, mova-se usando as teclas 5, 6, 7 e 8. Se us pontos serão constantemente marcados.

A seguir, a parte do programa em Basic:



```
 9 PRINT A$
10 FOR N=1 TO 6
11 PRINT B$
13 PRINT C$
15 PRINT C$
16 NEXT N
17 PRINT B$
18 PRINT A$
```

Para carregar os códigos listados a seguir, use o HEXLOAD, com pelo menos 870 caracteres na linha 1 REM. Os endereços de 17030 até 17130, não são usados e nem estão listados a seguir. Pule-os durante a edição.

```
16514    CD D0 41 2A 0C 40 11
16521    44 00 19 36 97 22 FB
16528    42 11 38 01 19 36 B4
16535    22 08 43 11 35 01 19
16542    36 97 22 02 43 CD 6F
16549    41 FE 7D C8 FE D7 28
16556    0E FE CF 28 0F FE EF
16563    28 10 FE E7 28 11 18
16570    53 11 FF FF 18 0D 11
16577    21 00 18 08 11 DF FF
16584    18 03 11 01 00 2A 08
16591    43 19 7E FE 80 28 38
16598    FE 97 CA 43 41 FE 1B
16605    20 23 E5 11 1A 00 2A
16612    0C 40 19 34 7E FE 26
16619    20 05 36 1C 2B 10 F5
16626    00 2A DC 43 2B 22 DC
16633    43 7C B5 E1 20 03 C3
16640    82 40 36 B4 E5 2A 08
16647    43 36 00 E1 22 08 43
16654    21 FA 42 11 0B 43 01
16661    07 00 ED B0 CD 75 41
16668    21 0B 43 11 FA 42 01
16675    07 00 ED B0 21 01 43
16682    11 0B 43 01 07 00 ED
16689    B0 CD 75 41 21 0B 43
16696    11 01 43 01 07 00 ED
16703    B0 C3 A3 40 21 FF FF
16710    CD C8 41 2A 08 43 36
16717    00 CD C2 41 36 B4 CD
16724    C2 41 E5 CD 6F 41 E1
16731    FE 7D C8 00 00 FE FF
16738    28 E8 C9 EB 11 03 00
16745    19 EB 01 1E 00 C9 2A
16752    25 40 7C A5 C9 06 03
16759    16 FF 2A 0E 43 7E FE
16766    00 20 06 2A 10 43 22
16773    0E 43 5E 7B FE 80 FC
16780    C0 41 23 22 0E 43 2A
```



```
16787    0C 43 19 7E FE B4 20
16794    04 E1 C3 43 41 F5 3A
16801    0B 43 2A 0C 43 77 F1
16808    FE 97 20 02 3E 1B 32
16815    0B 43 19 22 0C 43 36
16822    97 21 00 08 CD C8 41
16829    10 D3 C9 14 C9 3E FF
16836    3D 20 FD C9 F5 2B 7C
16843    B5 20 FB F1 C9 3E 1B
16850    32 FA 42 32 01 43 AF
16857    32 0A 43 21 36 01 22
16864    DC 43 21 12 43 22 FF
16871    42 22 FD 42 21 77 43
16878    22 06 43 22 04 43 C9
16885    22 FD 42 21 77 43 22
16892    06 43 22 04 43 C9 43
16899    C9 C9 14 C9 3E FF 3D
16906    20 FD C9 FD C9 11 44
16913    00 19 36 95 22 FB 40
16920    11 38 01 19 36 97 22
16927    08 41 11 35 01 19 36
16934    95 22 02 41 21 FF FF
16941    E5 CD 12 43 E1 CD 12
16948    43 CD 1A 43 FE D7 28
16955    13 FE CF 28 14 FE EF
16962    28 15 FE E7 28 16 FE
16969    7D C8 00 00 18 60 11
16976    FF FF 18 0D 11 21 00
16983    18 08 11 DF FF 18 03
16990    11 01 00 2A 08 41 19
16997    7E FE 80 28 45 FE 95
17004    CA E4 42 FE 1B 20 30
17011    E5 06 04 21 9D 40 34
17018    7E FE 26 20 05 36 1C
17025    2B 3D 3D 3D 3D 3D 3D
```

```
17130    3D 3D 3D 3D 3D 3D 3D
17137    3D 3D 3D 3D 1B 08 08
17144    1B 08 1B 8F 4F 14 43
17151    12 43 1B 30 51 79 43
17158    77 43 01 50 00 1B 30
17165    51 79 43 77 43 21 21
17172    21 21 21 21 01 DF DF
17179    DF DF DF DF 01 21 21
17186    21 21 21 21 01 DF DF
17193    DF DF DF DF 01 21 21
17200    21 21 21 21 01 DF DF
17207    DF DF DF DF 01 21 21
17214    21 21 21 21 01 DF DF
17221    DF DF DF DF 01 21 21
17228    21 21 21 21 01 DF DF
17235    DF DF DF DF FF FF FF
17242    FF 21 21 21 FF 21 FF
17249    DF FF 21 21 01 DF DF
17256    DF FF FF 21 21 01 DF
17263    DF DF FF 21 FF DF DF
17270    00 FF FF FF FF FF FF
17277    FF FF FF DF 01 01 01
17284    01 01 01 01 01 01 DF
17291    FF FF FF FF FF FF FF
17298    FF FF DF 01 01 01 01
```

```
17305    01 01 01 01 01 DF FF
17312    FF FF FF FF FF FF FF
17319    FF DF 01 01 01 01 01
17326    01 01 01 01 DF FF FF
17333    FF FF FF 21 21 FF FF
17340    21 01 01 01 21 FF FF
17347    FF 21 FF DF DF DF 01
17354    DF FF FF 21 21 21 01
17361    01 01 01 21 01 01 01
17368    01 01 01 00 38 01 1C
17375    1C 3D 3D 3D 3D 3D 3D
17382    3D 3D 3D 3D 3D 3D 3D
17389    3D 3D 3D 3D 3D 3D 3D
17396    3D 3D 3D 3D 3D 3D 3D
```

Para jogar acrescente:

```
19 SLOW
20 LET L=USR 16514
```


Neste programa, um avião cruza a tela, seguidas vezes, em diferentes alturas, para que voce tente alvejá-lo. Para facilitar o raciocínio da "montagem do programa", vamos fornecê-lo em partes:

Primeiro, o avião cruza a tela em diferentes alturas.

```
16530    3E 19          LD A,25            A=nº de posições
16532    2A 7B 40       LD HL,(16507)      HL=RND
16535    ED 5B 0C 40    LD DE,(16396)      LD DE com D.File
16539    19             ADD HL,DE          determina posição
16540    EB             EX DE,HL           troca DE com HL
16541    13             INC DE             1ª posição
16542    21 82 40       LD HL,16514        HL= end. avião
16545    01 06 00       LD BC,6            BC=6
16548    ED B0          LDIR               Copia 6 caracteres
16550    F5             PUSH AF            preserve AF
16551    CD BB 40       CALL looping       CALL retardo temp.
16554    F1             POP AF             recupere AF
16555    3D             DEC A              A=A-1 p/ próx.pos.
16556    FE 01          CP 1               última posição?
16558    28 15          JR Z looping       sim, JR apaga
16560    2A 7B 40       LD HL,(16507)      HL= posição vídeo
16563    23             INC HL             próxima posição
16564    22 7B 40       LD (16507),HL      preserve posição
16567    C3 94 40       JP 16532           repita próx.pos.
16570    C9             RET                retorne
16571    01 E8 03       LD BC,1000         rotina de retardo
16574    0B             DEC BC
16575    78             LD A,B
16576    FE 01          CP 1
16578    20 FA          JR NZ,-6
16580    C9             RET                retorne
16581    06 06          LD B,6             rotina para apagar
16583    EB             EX DE,HL           o avião ao final
16584    36 00          LD (HL),0          da linha
16586    2B             DEC HL
16587    10 FB          DJNZ,-5
16589    C9             RET                retorne
```

Para entrar os códigos,use o HEXLOAD. Porém, antes de tudo, coloque na linha 1 REM, como sendo os primeiros caracteres, os símbolos gráficos que formam a figura do avião.

O primeiro símbolo é obrigatoriamente um espaço. Os demais são os símbolos gráficos das teclas W, 6, 6, inverse space e 6.

Para testar, acrescente:

```
3 LET X=(INT (RND*6)+1)*33
4 POKE 16507,X
5 LET L=USR 16530
6 RUN
```

O programa em Basic coloca no endereço 16507 um valor aleatório, múltiplo de 33, até o máximo de 198. Isto faz com que o avião seja impresso em qualquer uma das seis primeiras linhas do vídeo. A rotina de impressão, usada neste exemplo, simplesmente copia os códigos dos seis primeiros caracteres após a primeira instrução do programa em Basic, para a posição desejada na memória de vídeo. Para cada nova posição em que o avião é impresso, não é necessário apagarmos "a imagem" anterior, porque a cauda deste é um caractere de espaço que elimina automaticamente o rastro deixado.

Esta outra parte interpreta se alguma tecla está pressionada e dispara o míssel. É uma continuação da anterior, a partir do endereço 16550.

```
16550   F5 PUSH AF      PUSH AF             preserva registers
16551   C5              PUSH BC
16552   D5              PUSH DE
16553   E5              PUSH HL
16554   CD C4 40        CALL looping        CALL retardo
16557   CD D7 40        CALL                CALL Keyboard
16560   E1              POP HL              recupera registers
16561   D1              POP DE
16562   C1              POP BC
16563   F1              POP AF
16564   3D              DEC A
16565   FE 01           CP 1
16567   28 15           JR Z,+21            JR  apaga avião
16569   2A 7B 40        LD HL,(16507)
16572   23              INC HL              próx. posição
16573   22 7B 40        LD (16507),HL       preserva posição
16576   C3 94 40        JP 16532            JP início
16579   C9              RET                 retorne
```



```
16580   01 E8 03        LD BC,1000          rotina de retardo
16583   0B              DEC BC              de tempo
16584   78              LD A,B              transfere B p/ A
16585   FE 01           CP 1                fim?
16587   20 FA           JR NZ,-6            não, repita
16589   C9              RET                 retorne
16590   06 06           LD BC,6             rotina apaga avião
16592   EB              EX DE,HL            troca DE com HL
16593   36 00           LD (HL),0
16595   2B              DEC HL
16596   10 FB           DJNZ,-5
16598   C9              RET                 retorne
16599   CD BB 02        CALL 699            CALL KEY SCAN
16602   2C              INC L               alguma tecla?
16603   C8              RET Z               não, retorne
16604   2A 88 40        LD HL,(16520)       rotina para impri-
16607   01 32 00        LD BC,50            mir o míssel
16610   A7              AND A               zera o carry-bit
16611   ED 42           SBC HL,BC           fim da tela?
16613   F2 EE 40        JP P,16622          não,JP míssel
16616   21 68 02        LD HL,616           HL=últ.linha vd.
16619   22 88 40        LD (16520),HL
16622   ED 5B 0C 40     LD DE,(16396)
16626   2A 88 40        LD HL,(16520)       recupera posição
16629   19              ADD HL,DE           do míssel.
16630   36 00           LD (HL),0           apaga pos. anterior
16632   11 21 00        LD DE,33
16635   2A 88 40        LD HL,(16520)
16638   A7              AND A               zera carry-bit
16639   ED 52           SBC HL,DE
16641   22 88 40        LD (16520),HL
16644   CD 12 41        CALL RET            CALL rotina PONTOS
16647   2A 88 40        LD HL,(16520)
16650   ED 5B 0C 40     LD DE,(16396)
16654   19              ADD HL,DE
16655   36 15           LD (HL),21          LD (HL) c/ míssel
16657   C9              RET                 retorna ao Basic
16658   C9              RET                 rotina pontos,  a
                                            ser escrita
```

Para que o programa fique completo, escreva à partir do endereço 16658, uma rotina que interprete se o avião foi atingido e, em caso positivo, incremente um placar de pontos.

SUB-ROTINAS DA ROM

PRINT E SPRINT

A rotina da ROM, denominada PRINT, coloca no vídeo o caractere cujo código esteja no register A, quando a mesma for chamada. Seu endereço de ínicio é 2056. Veja como usar esta rotina, em um exemplo do da instrução CALL.

CALL PRINT coloca no vídeo apenas um caractere. O que pretendemos é colocar no vídeo conjuntos de caracteres, textos, etc..., de uma só vez. Para isto precisaremos de uma pequena rotina, além da própria PRINT. Vamos chamar esta nova rotina de SPRINT. Ela tomará um caractere, armazenado em uma tabela à parte e enviará ao vídeo através de CALL PRINT.Repetirá a operação até que todos os códigos de caracteres se esgotem, o que será identificado pelo código 255, ao final dos mesmos.

A rotina proposta,a seguir, imprime no vídeo os caracteres cujos códigos estejam nos endereços de 16550 em diante, até encontrar o código 255. Carregue com o HEXLOAD.

16514	21 A6 40	LD HL,16550	LD HL,início CHR$
16517	7E	LD A,(HL)	LD A com o CHR$
16518	FE FF	CP 255	é o último CHR$?
16520	C8	RET Z	sim, retorne
16521	E5	PUSH HL	preserve HL
16522	CD 08 08	CALL 2056	CALL PRINT
16525	E1	POP HL	reassume HL
16526	23	INC HL	novo end.de CHR$
16527	C3 85 40	JP 16517	JP CHR$ seguinte

Para testar, voce precisará colocar alguns caracteres, a partir do endereço 16550. Para que voce não precise ficar procurando os códigos equivalentes aos caracteres desejados, acrescente estas linhas em Basic.

```
900 FOR N=16550 TO 20000
910 INPUT X$
920 IX X$="" THEN STOP
930 LET X=CODE(X$)
940 IF X$<>"Ø" THEN PRINT X$;
950 IF X$="Ø" THEN LET X=118
960 IF X$="Ø" THEN PRINT
970 POKE N,X
980 NEXT N
```

Para cada caractere digitado, acione New Line. Para mudar de linha não é preciso completar a mesma com espaços. Basta usar o código 118 de New Line. Para introduzí-lo digite Ø. Para parar a entrada de caracteres, digite apenas New Line. A seguir digite:

```
POKE N,255
```

É o código de término da tabela de caracteres. Experimente por:

```
RAND USR 16514
```


KEYBOARD SCAN

Existe na ROM uma rotina destinada a ler o valor do teclado. Sem circuitos codificadores ou decodificadores, o teclado do eu mi crocomputador gera um número binário único, para cada tecla pressionada, sem e com o SHIFT. O código de cada tecla é gerado exclusivamente por uma interessante ligação elétrica entre as mesmas. Veja abaixo:

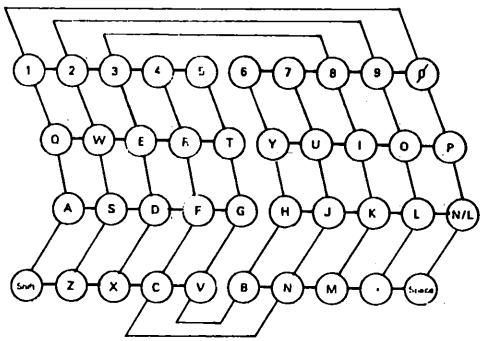
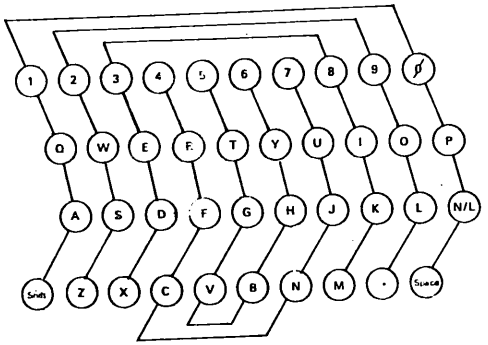


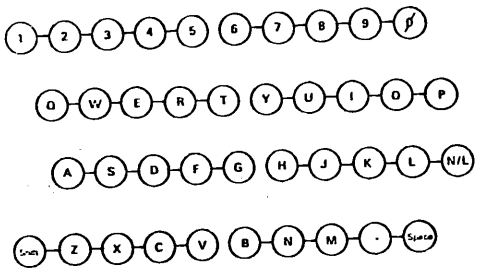
DIAGRAMA ELÉTRICO DO TECLADO

Podemos observar que as teclas estão estão interligadas em blocos, como a seguir

Em cinco blocos mais o SHIFT.



Em oito blocos.



Cada uma dessas seções equivale a um "bit do register HL". Se nenhuma tecla for pressionada, teremos o valor binário:

11111111 11111111 ou, em hex FFFF.

Para cada tecla pressionada teremos um valor diferente, com e sem o SHIFT acionado.

Para obter o valor do teclado no register HL, basta usar CALL KEYBOARD SCAN. O endereço é 02BBh .

A rotina abaixo fornece o valor do register HL, após o KEYBOARD SCAN. Experimente carregá-la com o auxílio do HEXLOAD.

```
16514   CD BB 02    CALL 02 BB      CALL KSCAN
16517   01 00 00    LD BC,0         BC=Ø
16520   4D          LD C,L          transfere C p/ L
16521   2C          INC L           alguma tecla ?
16522   CA 82 40    JP Z,16514      não ?, JP 16514
16525   C9          RET             retorne
```

Teste por PRINT USR 16514, ou inclua antes:

```
2 PRINT USR 16514
3 RUN      ou GOTO 2
```

Caso nenhuma tecla seja pressionada, o register C permanece com o valor 255. A instrução INC L produzirá resultado zero, se o valor de C for 255. Desta forma, a instrução JUMP IF ZERO 16514 será executada, sem que o programa retorne ao Basic, imprimindo o valor de C. Caso alguma tecla seja pressionada, JP if Z não será mais válido e o programa retorna ao Basic, imprimindo o valor de C. Como no programa em Basic a linha 3 é RUN ou GOTO 2, a rotina volta a ser executada, para permitir a interpretação de uma nova tecla.

Os valores produzidos no register L, pelo teclado, na rotina acima, são os seguintes:

TECLAS	VALOR
1 2 3 4 5	247
Q W E R T	251
A S D F G	253
Z X C V B	254
6 7 8 9 0	239
Y U I O P	223
H J K L N/L	191
B N M . space	127

Alterando a última rotina, para produzir o valor de HL, fica assim:

```
16514   CD BB 02    CAL 02 BB     CALL KSCAN
16517   01 00 00    LD BC,0       zere BC
16520   44          LD B,H        transfere H p/ B
16521   4D          LD C,L        transfere L p/ C
16522   2C          INC L         alguma tecla?
16523   CA 82 40    JP Z,16514    não,JP 16514
16526   C9          RET           retorne
```

Para testar, inclua:

```
2 PRINT USR 16514
3 GOTO 2
```

Verifique os valores produzidos pelas teclas 5, 6, 7, 8, J,K,M e N

TECLA	VALOR
5	57 335
6	57 327
7	61 423
8	63 471
J	61 375
K	63 423
N	61 311
M	63 359

Na rotina acima, a transferência dos valores de H e L para B e C foi feita por LD B,H e LD C,L . Experimente substituir estas instruções por PUSH HL e POP BC que produzirão o mesmo efeito. Os códigos de PUSH HL POP BC são respectivamente E5 e C1 .

FIND CHR$

FIND CHR$ é uma sub-rotina da ROM,para determinar o código de um caractere, após uma interpretação do teclado. O código do caractere é colocado no endereço por HL, após a chamada de FINDCHR$. Pode ser removido para o Basic, como no exemplo a seguir que utiliza a rotina KSCAN, KEYBOARD SCAN, mencionada no capítulo anterior. Carregue-a com a ajuda do HEXLOAD.

```
16514   CB BB 02    CALL 02 BB    CALL KSCAN
16517   01 00 00    LD BC,0       zere BC
16520   E5          PUSH HL       coloque HL
16521   C1          POP BC        em BC
16522   2C          INC L         alguma tecla ?
16523   CA 82 40    JP Z 16514    não, reinicie
16526   CD BD 07    CALL 07 BD    CALL FINDCHR$
16529   01 00 00    LD BC,0       zere BC
16532   42          LD C,(HL)     LD C com CHR$
16533   C9          RET           retorne
```

Para experimentar, inclua:

```
2 PRINT USR 16514
3 GOTO 2
```

Para cada tecla pressionada, será impresso o código do caractere equivalente. Mantendo-se a tecla presa, repete-se a impressão do código. Voce também pode imprimir os caracteres, velozmente, usando esta mesma rotina. Altere a parte em Basic, para:

```
2 PRINT CHR$ (USR 16514);
3 GOTO 2
```

SUB-ROTINAS ARITMÉTICAS

Não nos termos aprofundado no uso dos micros ZX80 com 4K de ROM, precursor dos ZX80, ZX81, TK82, NE Z8000, CP 200 e outros com 8K de ROM, NÃO foi uma vantagem, muito pelo contrário.

PERDEMOS A PRIMEIRA LIÇÃO.

As máquinas com 4K de ROM só trabalhavam com números inteiros, na faixa de -32768 a +32767, a mesma coisa que de 0 a 65535, o que pode ser manipulado em um único register pair. Uma sub-rotina aritmética seria:

```
LD DE,0050
LD HL,0111
CALL MULT
RET
```

O conteúdo dos registers DE e HL ficavam como operandos das sub-rotinas aritméticas, ficando o resultado em HL e este era dado, quando do retorno ao Basic, como no caso acima. Simples, não ?

O funcionamento destas rotinas será omitido por escassez de espaço, e porque as máquinas atuais são equipadas com a ROM de 8K.

Como já mencionado várias vezes, a ROM de 8K armazena as constantes numéricas ,i.e., os números, em cinco bytes com notação de ponto flutuante.

As sub-rotinas da ROM,das operações aritméticas, tomam como operandos, os dados armazenados em um STACK para este fim. O chamado STACK DO CALCULADOR.Para colocar dados no STACK do calculador, na forma Sinclair de 5 bytes, existem as rotinas:

1. STACK A — CALL 151D

   Coloca o conteúdo do register A no STACK do calculador.

2. STACK BC — CALL 1520

   Coloca o conteúdo do register pair BC no STACK do calculador.

3. UNSTACK BC — CALL A70E

   Retira o conteúdo do STACK do calculador para o register BC.

A rotina STACK A, quando chamada, zera o register B e coloca o conteúdo de A em C, prosseguindo então na rotina STACK BC.

O STACK do microprocessador funciona como uma pilha de dados, de cima para baixo. O STACK do calculador funciona ao contrário. De baixo para cima.

Resumindo. Um número, na forma Sinclair de 5 bytes,é armazenado nas posições de memória STKEND, STKEND +1, STKEND +2, STKEND +3 e STKEND +4 e pode ser multiplicado, dividido, somado ou calculado de um outro número colocado nas posições de memória STKEND +5, STKEND +6, STKEND +7, STKEND +8 e STKEND +9.

STKEND é a variável do sistema operacional que fornece o endereço do fim do STACK do calculador. Para colocar dados no STACK do calculador, usa-se CALL STACK A ou CALL STACK BC, conforme mencionado anteriormente. O procedimento para a execução de sub-rotinas aritméticas é o seguinte:

Colocar o(s) operando(s) no STACK do calculador por CALL STACK A ou CALL STACK BC. Aplicar a instrução RESTART endereço 28h seguida do código ou códigos das operações a executar, encerrando-se sempre por 34h e CALL UNSTACK BC .

Os códigos das operações são, em hex :

03	SUB	21	ATN
04	MUL	22	LN
05	DIV	23	EXP
0F	ADD	24	INT
19	CODE	25	SQR
1A	VAL	26	SGN
1B	LEN	27	ABS
1C	SIN	28	PEEK
1D	COS	29	USR
1E	TAN	2A	STR$
1F	ASN	2B	CHR$
20	ACS	2C	NOT

Estes códigos são obtidos a partir do código de caractere da função, menos 171 (decimal).

Para, por exemplo, multiplicarmos 15 por 10, a sequência de instruções seria:

16514	3E 0F	LD A,15	LOAD A com 15
16516	CD 1D 15	CALL 151D	CALL STACK A
16519	3E 0A	LD A,10	LOAD A com 10
16521	CD 1D 15	CALL 151D	CALL STACK A
16524	EF	RST 28	CALL CALCULADOR
16525	04		código MULT
16526	34		fim dos cálculos
16527	CD A7 0E	CALL 0EA7	CALL UNSTACK BC
16530	C9	RET	retorne

PRINT USR 16514 resultará em 150. Altere o final desta rotina para:

16525	0F		código de soma
16526	25		código de SQR
16527	34		fim dos cálculos
16528	CD A7 0E	CALL 0EA7	CALL UNSTACK BC
16531	C9	RET	retorne

Neste último exemplo, usamos dois códigos após o RST 28. OF e 25, respectivamente, soma e raiz quadrada. Podemos usar vários códigos de funções, em seguida de um único RST 28, desde que terminados pelo código 34h .

A rotina UNSTACK coloca no register BC, o valor do STACK, ou melhor, o valor dos primeiros cinco bytes do STACK, com as seguintes limitações:

1. O valor colocado em BC será sempre um número inteiro.
   As aproximações seguem o exemplo: 1,5 ≅ 2 e 1,4 ≅ 1
2. Números negativos ou maiores que 65535 interrompem a rotina, retornando ao Basic com o código de erro B. Fora de faixa.

Alguns dos códigos, fornecidos na página anterior, como CODE, STR$, VAL e outros não citados, mas que também são funções, sugerem o uso de strings no STACK do calculador. Por exemplo:

16514	01 C3 02	LD BC,707	LOAD BC com 707
16517	CD 20 15	CALL 1520	CALL STACK BC
16520	EF	RST 28	CALL CALCULADOR
16521	2A		código STR$
16522	19		código CODE
16523	34		código de término
16524	CD A7 0E	CALL 0EA7	CALL UNSTACK BC
16527	C9	RET	retorne

O valor 707 colocado no SATCK é convertido na STRING "707" pelos códigos RST 28 e 2A .

SUB-ROTINAS ARITMÉTICAS COM PONTO FLUTUANTE

A rotina UNSTACK BC, como vimos, só nos permite retornar números inteiros na faixa de 0 a 65535. Existe na ROM uma rotina destinada exclusivamente a remover números com ponto flutuante do STACK do calculador para os registers A, E, D, C e B. Esta rotina, denominada UNSTACK FPN (Floating Point Number), está no endereço 13F8h ou,05112 decimal.

Para retornar "A FLOATING POINT NUMBER", ao invés de encerrar a sub rotina aritmética por CALL UNSTACK BC, usa-se CALL UNSTACK FPN. Teremos então o nosso número,com ponto flutuante, nos registers A, E, D, C, e B. Para utilizá-lo, sobram duas opções:

1. Utilizá-lo tal como está, na forma de 5 bytes, seguindo uma codificação própria, lançada por Sir Clive Sinclair.
2. Colocá-lo no sistema de variáveis do usuário. Por exemplo:no lugar ou endereços da variável A. Para convertê-lo ao sistema decimal, usamos o próprio Basic Sinclair. O valor poderá ser dado pela variável A.

Abaixo, apresentamos uma sugestão de como transferir os valores dos registers A, E, D, C e B, para os endereços de uma variável do sistema do usuário:

```
20 10 40    LD HL,(16400)   LD HL c/ end.vars.
23          INC HL          HL=HL+1 p/ 1º end.
77          LD (HL),A       As instruções se
23          INC HL          guintes transferem
73          LD (HL),E       os valores de A,E,
23          INC HL          D, C e B, para os
72          LD (HL),D       endereços da 1ª va
```



```
23          INC HL          riável do sistema
71          LD (HL),C       em Basic.
23          INC HL
70          LD (HL),B
C9          RET             retorne
```

A seguir, propomos um programa para extrair a raiz quadrada de um e meio, utilizando as rotinas anteriormente citadas. Para entrar o valor 1,5 a ser extraída a raiz quadrada, usamos 15 dividido por 10. A seguir da instrução RST 28, CALL CALCULADOR, deixamos alguns bytes reservados com a instrução NOP, NO OPERATION, para que no futuro possam ser incluídas outras operações.

```
16514   3E 0F          LD A,15          LD A com 15
16516   CD 1D 15       CALL 151D        CALL STACK A
16519   3E 0A          LD A,10          LOAD A com 10
16521   CD 1D 15       CALL 151D        CALL STACK A
16524   EF             RST 28           CALL CALCULADOR
16525   05                              código DIV
16526   25                              código SQR
16527   34                              fim dos cálculos
16528   00 00 00 00    NOP              reserva memória
16532   CD F8 13       CALL 5112        CALL UNSTACK FPN
16535   2A 10 40       LD HL,(16400)    LD HL c/ end.vars.
16538   23             INC HL           HL=HL+1 p/ 1ª var.
16539   77             LD (HL),A        As instruções se
16540   23             INC HL           guintes transferem
16541   73             LD (HL),E        os valores de A,E,
16542   23             INC HL           D, C, e B, para os
16543   72             LD (HL),D        endereços da 1ª va
16544   23             INC HL           riável do usuário,
16545   71             LD (HL),C        no sistema em Basic.
16546   23             INC HL
16547   70             LD (HL),B
16548   C9             RET              retorne
```

Para testar o programa, acrescente em Basic:

```
2 CLEAR
3 LET A=0
4 RAND USR 16514
5 PRINT A
```

Digite RUN e New Line.

CALL SAVE, LOAD, NEW, LIST, ETC...

No capítulo A ROM, voce encontrará os endereços das principais rotinas, comandos e funções do monitor. Aqui, a título de exemplos, vamos ver como usar alguns comandos, diretamente por USR.

LOAD

No modo FAST, digite LET L=USR 839. Use isto com programas do tipo que "saem rodando" do cassette e se auto travam. O programa será "parado" após a carga.

SAVE

Digite LET L=USR 787 para SAVE, diretamente.

Digite LET L=USR 763 para SAVE, após cinco segundos de pausa.

CLS

Substitua uma linha de programa com CLS, por LET L=USR 2602

NEW

Use LET L=USR 963. Isto não altera a variável RamTop. LET L=USR 0 é o mesmo que ligar e desligar o micro, reinicializando todo o sistema.

FAST

Use LET L=USR 3875 . Para SLOW use USR 3883.

A ROM

A ROM

As memórias sómente de leitura destes microcomputadores, sejam elas ROMs, PROMs ou EPROMs, possuem gravada uma longa sequência de oito mil cento e noventa e dois códigos gravados, representando instruções do Z80, constituindo o software básico do equipamento. Nestas mesmas memórias estão também gravados os padrões dos caracteres, as palavras-chave e outras tabelas.

Este capítulo apresenta:

1. As duas versões da ROM de 8K
2. As diferenças das ROMs de 8K dos microcomputadores ZX 81, TK 82, NE Z8000, CP 200 e TK 85.
3. Os endereços de início das principais rotinas e seus usos.
4. A tabela com os padrões dos caracteres e o programa usado para gerá-la.
5. A tabela com os caracteres das palavras-chave e o programa usado para gerá-la.

AS DUAS VERSÕES DA ROM DE 8K

As primeiras ROMs de 8K produzidas pela Sinclair Research a partir de março de 81 e as produzidas aqui no Brasil, ainda até aproximadamente março de 82, continham vários erros, sendo os mais graves em certas operações aritméticas. Por esta razão, produziu-se uma nova versão da ROM. As máquinas, atualmente em produção, possuem a última versão da ROM e, as anteriormente produzidas, que recebe ram manutenção ou aperfeiçoamentos, como a inclusão de SLOW, em sua maioria também possuem a nova.

Para identificar uma ROM antiga, digite PRINT 0.25**2, que resultará em 3.1423844, ou PRINT PEEK 41, que resultará em 156.

Para alterar uns poucos bytes, causadores destes erros, foi preciso deslocar de seus endereços uma grande parte das rotinas da ROM, conforme abaixo:

ENDEREÇOS EM HEX	DESLOCAMENTO
0000 a 0EE9	alterado
0F20 a 1022	+ 3 bytes
1046 a 1716	+ 4 bytes
1737 a 1DE1	+ 1 byte
1E00 a 2000	inalterado

Devido à alteração dos endereços de início de várias rotinas da ROM, programas com sub-rotinas da ROM antiga não funcionam na nova. O mesmo ocorre com as sub-rotinas aritméticas citadas pela Senhora Baker, no livro Mastering Machine Code. Altere conforme acima.

ZX 81 VERSUS TK 85

Abaixo fornecemos um programa para a "comparação instantânea" das das ROMs dos microcomputadores ZX81 e TK85 .

```
   1 GOTO 1000
  10 REM MICRON ELETRONICA
 100 DIM A(2000)
 105 LET X=((PEEK 16400+256*PEEK
 16401))
 150 PRINT "ARQUIVA ROM EM A(200
0)"
 160 INPUT P$
 200 FOR N=0 TO 8192
 210 POKE (X+N+10),PEEK N
 220 NEXT N
 230 PRINT "OK - FIM"
 300 PRINT "DIGITE O NOME DO APA
RELHO"
 400 PRINT "LIGUE O CASSETTE P/
GRAVAR"
 450 INPUT P$
 460 SAVE P$
 470 GOTO 1000
 500 PRINT ,,"COMPARA"
 507 PRINT ,,"DIGITE NOME DO APA
RELHO"
 510 INPUT X$
 512 LET X=((PEEK 16400+256*PEEK
 16401))
 515 CLS
 518 PRINT "END.";TAB 10;P$,X$
 519 PRINT
 520 FOR N=0 TO 8191
 535 LET R=PEEK N
 540 IF PEEK (X+N+10)=R THEN NEX
T N
 550 PRINT N;TAB 10;PEEK (X+N+10
),R
 560 NEXT N
 570 PRINT ,,"OK - FIM"
1000 PRINT ,,,,"COMPARA ROM/S DE
 8K"
1005 PRINT ,,"P/","DIGITE"
1008 PRINT
1010 PRINT "GRAVAR","1"
1020 PRINT "COMPARAR","2"
1030 INPUT X$
1040 IF X$<>"1" AND X$<>"2" THEN
 GOTO 1030
1050 IF X$="1" THEN GOTO 100
1060 GOTO 500
```

Rodando o programa da página anterior, será mostrado no vídeo o menu abaixo:

```
COMPARA ROM/S DE 8K
P/                DIGITE
GRAVAR            1
COMPARAR          2
```

1 - Grava na memória não permanente (RAM) os valores contidos na ROM do microcomputador em uso e, a seguir, os grava no cassette.

2 - Compara os valores da memória não permanente (RAM), carregada com os dados do cassette, com a ROM do microcomputador em uso, fornecendo as diferenças.

Tendo comparado as ROMs de 8K dos microcomputadores TK 85 e ZX81, constatamos as diferenças (?) abaixo :

END.	ZX81	TK85
4	127	254
23	255	57
37	255	48
38	255	36
39	255	30
101	255	40
3901	253	205
3902	54	7
3903	53	2
3904	255	253
3905	205	54
3906	7	53
3907	2	255

Na comparação das ROMs dos microcomputadores CP 200 (16K de RAM)e ZX 81, constatamos as diferenças (?) abaixo :

END.	ZX81	CP200
3901	253	205
3902	54	7
3903	53	2
3904	255	253
3905	205	54
3906	7	53
3907	2	255

As diferenças acima são as mesmas obtidas na comparação de um TK 82C s/ SLOW, um NE Z8000 e o ZX 81.

Observem que no TK 85, no endereço 00004 da ROM, nós temos o valor 254,contra 127,do ZX 81 e primeiros TKs e CPs 200. A razão é que este valor significa o último endereço de RAM, viável. As primeiras máquinas tinham expansões de memória de até 16K e o último endereço seria FF7Fh. Já no TK 85, por exemplo, temos FFFEh.

Nas máquinas em que PRINT PEEK 4 produz 127, para se utilizar uma expansão de RAM acima de 16K, é preciso alterar a variável do sistema operacional denominada RamTop, ou o equipamento não identificará a existência de mais de 16K de RAM.

OS ENDEREÇOS

A seguir apresentamos os endereços de início das principais rotinas e tabelas da ROM. Os endereços fornecidos são os da 2ª versão da ROM e as diferenças com a 1ª versão estão resumidas na primeira parte deste capítulo.

0000 00000
Início do monitor. Todo sistema com Z80 inicia em 00000 .

0008 00008
Reportagem de erro . Determina o caractere do código de erro.

0010 00016
PRINT caractere. Imprime um caractere na posição seguinte do vídeo.

0018 00024
Obtém "current caracter" de uma linha em Basic.

0020 00032
Ídem 0018, para o próximo caractere.

0028 00040
Entrada para as rotinas de cálculos com ponto flutuante.

0030 00048
Reserva BC bytes na área das variáveis.

0038 00056
Rotina de interrupção para produzir uma linha de TV.

0066 00102
Produz uma linha de TV após um NMI (non maskable interrupt), no modo SLOW.

007E 00126
Até 00203 contém os códigos de cada tecla sem e com o SHIFT pressionado.

00CC 00204
Até 00242 contém os códigos de cada tecla no modo FUNCTION.

00F3 00243
Até 00273 contém os códigos de cada tecla no modo GRAPHICS.

0111 00273
Até 00507 contém os códigos dos caracteres de cada palavra-chave, estando o último caractere na forma inversa.

01FC 00508
Rotina "update" usada pelas de SAVE e LOAD.

027A 00634
Gerador da imagem de vídeo.

02BB 00699
Leitura do teclado. Cada tecla produz um valor único no register HL.

02F6 00758
Comando SAVE.

0340 00832
Comando LOAD.

03CB 00971
Verificação da RAM. É usada na inicialização e após o comando NEW.

03E5 00997
Rotina principal de inicialização.

0454 01108
Cursor Down. Desce o cursor.

0482 01154
Constroi uma linha no modo Edit.

052B 01323
Seleciona a tecla no modo Edit.

05C4 01476
Rotina de edição.

06E3 01763
Roda o programa em Basic.

0745 01861
Imprime uma linha completa, em Basic.

07BD 01981
É FIND CHR$. O valor produzido no register HL pela rotina do endereço 02BB, a KeyBoard Scan, deve ser colocado no register BC, para ser convertido em um valor entre 1 e 78h, que é o código da tecla digitada. Esta rotina,ainda, coloca em HL um endereço da tabela gravada nos endereços de 007Eh em diante, referente à tecla dada.

07F1 02033
Rotina PRINT CHR$,usada por RST 10 .

08F5 02293
Verifica e ajusta a validade da variável do sistema operacional, denominada DF-CC, Display File - Current Caracter.

0918 02328
Expande o Display File, DFile.

094B 02379
PRINT a Keyword. Imprime uma palavra-chave.

0A98 02712
É usada para imprimir os números das linhas de programa, em Basic, durante a listagem. Usando o register BC, fornece um número decimal na faixa de 1 a 9999.

0C29 03113
É a Pointer Table. Também fornece a sintaxe de cada comando e o endereço do mesmo.

0CBA 03258
Interpretador do Basic.

0EA7 03751
UNSTACK. Converte um número na forma Sinclair de cinco bytes com ponto flutuante, que esteja armazenado no STACK do calculador , em um inteiro na faixa de 0 a 65535. Números fora de faixa interrompem a rotina, com retorno ao Basic e código de erro B .

0F55 03925
É a rotina "Expression Evaluator".

14CE 05326
Trabalha os números com ponto flutuante.

151D 05405
Converte o valor do register A,para a forma Sinclair de cinco bytes e coloca nos próximos cinco endereços do STACK do calculador.

1520 05408
O mesmo que a rotina anterior, só que o valor convertido é o contido no register BC.

158E 05518
Rotina "Evaluate to Integer".

15DF 05599
PRINT "Last Value".

1915 06421
Tabela de funções para o calculador.

199D 06557
Calculador com ponto flutuante.

1AAA 06826
A partir deste endereço estão as rotinas das funções.

1E00 07680
Contém a tabela com os padrões dos caracteres, formados por sessenta e quatro bits, cada um. Oito bytes de oito bits,dão 64 bits.
O endereço de início,do caractere,é 7680 + (8 vezes o seu código).

A seguir, os endereços dos comandos,do Basic:

CLEAR	149A	05274	NEW	03C3	00963
CLS	0A2A	02602	NEXT	0E2E	03630
CONT	0E7C	03708	PAUSE	0F32	03890
COPY	0869	02153	PLOT	0BAF	02991
DIM	1409	05129	POKE	0E92	03730
FAST	0F23	03875	PRINT	0ACF	02767
FOR	0DB9	03513	RAND	0E6C	03692
GOSUB	0EB5	03765	REM	0D6A	03434
GOTO	0E81	03713	RETURN	0ED8	03800
IF	0DAB	03499	RUN	0EAF	03759
INPUT	0EE9	03817	SAVE	02F6	00758
LET	131D	04893	SCROLL	0C0E	03086
LIST	0730	01840	SLOW	0F2B	03883
LLIST	072C	01836	STOP	0CDC	03292
LOAD	0340	00832	UNPLOT	0BAF	02991
LPRINT	0ACB	02763			

OS CARACTERES DA ROM

Na ROM, nos endereços de 7680 a 8191, estão gravados os padrões de 61 caracteres. Cada caractere ocupa 8 bytes.

A localização de um caractere na ROM é dada por:

ENDEREÇO DE INÍCIO = 7680 + (código do caractere desejado X 8)

ENDEREÇO DE FIM = endereço de início + 8

Nas próximas páginas voce encontrará uma listagem da ROM, dos endereços 7680 a 8191, com quatro colunas distintas. A saber:

1ª coluna,à esquerda, fornece o endereço ou posição da memória.

2ª coluna fornece o valor gravado no endereço indicado pela 1ª coluna. O valor é dado no sistema decimal. Note que a máquina não arquiva valores no sistema decimal e sim binário.

3ª coluna contém 8 dígitos binários e é o valor equivalente à segunda coluna, em binário.

4ª coluna contém os 8 dígitos binários como a 3ª coluna, tendo os bits de valor "1" sidos substituídos por um ponto preto "■" que ilustra os caracteres gravados de forma ampliada.

Os valores lidos da ROM estarão sempre no sistema decimal, em consequência das funções PEEK e POKE, do Basic, oferecerem operandos no sistema decimal. Para uma melhor visualização da formação dos padrões dos caracteres, devemos observá-los em binário. Para este fim, propomos o programa a seguir:

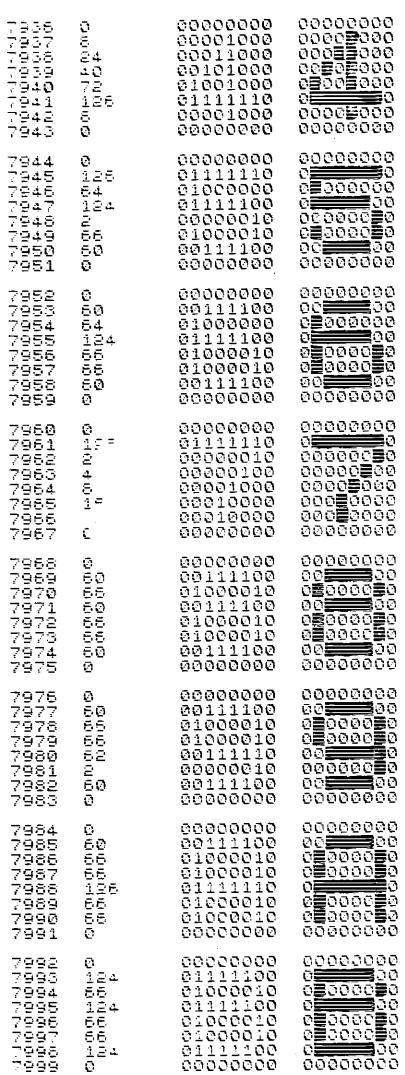

```
 10 REM MICRON ELETRONICA
 20 DIM A$(1,9)
 25 DIM B$(1,9)
 30 DIM A(9)
 60 FOR N=0 TO 7
 70 LET A(8-N)=2**N
 80 NEXT N
 82 GOTO 500
 85 LET T=0
 86 LET A$(1)="XXXXXXXX"
 87 LET B$(1)="        "
 90 FOR N=1 TO 8
100 IF X<T+A(N) THEN NEXT N
110 LET A$(1,N)="1"
120 LET T=T+A(N)
130 NEXT N
140 LET B$(1)=A$(1)
200 FOR N=1 TO 8
210 IF B$(1,N)="1" THEN LET B$(
1,N)="■"
212 IF B$(1,N)<>"■" THEN LET B$
(1,N)="0"
215 IF A$(1,N)<>"1" THEN LET A$
(1,N)="0"
220 NEXT N
230 RETURN
500 PRINT "LEITURA DOS CARACTER
ES"
510 PRINT
520 PRINT "DIGITE ENDERECO DE I
NICIO:";
530 INPUT E
531 LET CC=0
532 CLS
535 FOR Z=E TO 20000
536 LET CC=CC+1
545 LET X=PEEK Z
546 LPRINT Z;"  ";X;TAB 14;
547 GOSUB 85
550 LPRINT A$(1, TO 8);
554 LPRINT TAB 24;B$(1, TO 8)
556 IF CC=8 THEN LPRINT
557 IF CC=8 THEN LET CC=0
560 NEXT Z
900 STOP
901 SAVE "ROM-CARACTERE■"
902 RUN
```

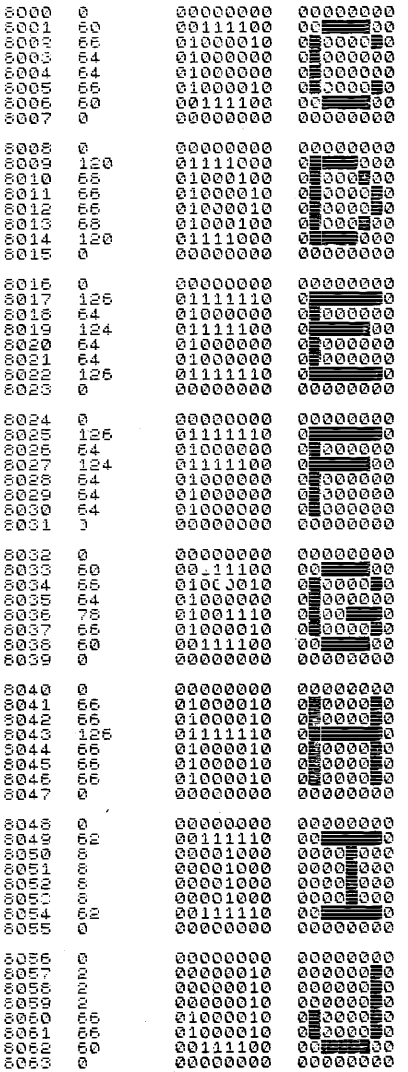
7680	0	00000000	00000000
7681	0	00000000	00000000
7682	0	00000000	00000000
7683	0	00000000	00000000
7684	0	00000000	00000000
7685	0	00000000	00000000
7686	0	00000000	00000000
7687	0	00000000	00000000
7688	240	11110000	0000
7689	240	11110000	0000
7690	240	11110000	0000
7691	240	11110000	0000
7692	0	00000000	00000000
7693	0	00000000	00000000
7694	0	00000000	00000000
7695	0	00000000	00000000
7696	15	00001111	0000
7697	15	00001111	0000
7698	15	00001111	0000
7699	15	00001111	0000
7700	0	00000000	00000000
7701	0	00000000	00000000
7702	0	00000000	00000000
7703	0	00000000	00000000
7704	255	11111111	
7705	255	11111111	
7706	255	11111111	
7707	255	11111111	
7708	0	00000000	00000000
7709	0	00000000	00000000
7710	0	00000000	00000000
7711	0	00000000	00000000
7712	0	00000000	00000000
7713	0	00000000	00000000
7714	0	00000000	00000000
7715	0	00000000	00000000
7716	240	11110000	0000
7717	240	11110000	0000
7718	240	11110000	0000
7719	240	11110000	0000
7720	240	11110000	0000
7721	240	11110000	0000
7722	240	11110000	0000
7723	240	11110000	0000
7724	240	11110000	0000
7725	240	11110000	0000
7726	240	11110000	0000
7727	240	11110000	0000
7728	15	00001111	0000
7729	15	00001111	0000
7730	15	00001111	0000
7731	15	00001111	0000
7732	240	11110000	0000
7733	240	11110000	0000
7734	240	11110000	0000
7735	240	11110000	0000
7736	255	11111111	
7737	255	11111111	
7738	255	11111111	
7739	255	11111111	
7740	240	11110000	0000
7741	240	11110000	0000
7742	240	11110000	0000
7743	240	11110000	0000

7744	170	10101010	
7745	85	01010101	
7746	170	10101010	
7747	85	01010101	
7748	170	10101010	
7749	85	01010101	
7750	170	10101010	
7751	85	01010101	
7752	0	00000000	00000000
7753	0	00000000	00000000
7754	0	00000000	00000000
7755	0	00000000	00000000
7756	170	10101010	
7757	85	01010101	
7758	170	10101010	
7759	85	01010101	
7760	170	10101010	
7761	85	01010101	
7762	170	10101010	
7763	85	01010101	
7764	0	00000000	00000000
7765	0	00000000	00000000
7766	0	00000000	00000000
7767	0	00000000	00000000
7768	0	00000000	00000000
7769	36	00100100	00 00 00
7770	36	00100100	00 00 00
7771	0	00000000	00000000
7772	0	00000000	00000000
7773	0	00000000	00000000
7774	0	00000000	00000000
7775	0	00000000	00000000
7776	0	00000000	00000000
7777	28	00011100	000 00
7778	34	00100010	00 000 0
7779	120	01111000	0 0000
7780	32	00100000	00 00000
7781	32	00100000	00 00000
7782	126	01111110	0 0000 0
7783	0	00000000	00000000
7784	0	00000000	00000000
7785	8	00001000	0000 000
7786	62	00111110	00 0000 0
7787	40	00101000	00 0 000
7788	62	00111110	00 0000 0
7789	10	00001010	0000 0 0
7790	62	00111110	00 0000 0
7791	8	00001000	0000 000
7792	0	00000000	00000000
7793	0	00000000	00000000
7794	0	00000000	00000000
7795	16	00010000	000 0000
7796	0	00000000	00000000
7797	0	00000000	00000000
7798	16	00010000	000 0000
7799	0	00000000	00000000
7800	0	00000000	00000000
7801	60	00111100	00 0000 00
7802	66	01000010	0 0000 0
7803	4	00000100	00000 00
7804	8	00001000	0000 000
7805	0	00000000	00000000
7806	8	00001000	0000 000
7807	0	00000000	00000000

7808	0	00000000	00000000
7809	4	00000100	00000000
7810	0	00001000	00000000
7811	0	00001000	00000000
7812	0	00001000	00000000
7813	0	00001000	00000000
7814	4	00000100	00000000
7815	0	00000000	00000000
7816	0	00000000	00000000
7817	16	00010000	00000000
7818	16	00010000	00000000
7819	16	00010000	00000000
7820	16	00010000	00000000
7821	16	00010000	00000000
7822	32	00100000	00000000
7823	0	00000000	00000000
7824	0	00000000	00000000
7825	0	00000000	00000000
7826	16	00010000	00000000
7827	8	00001000	00000000
7828	4	00000100	00000000
7829	8	00001000	00000000
7830	16	00010000	00000000
7831	0	00000000	00000000
7832	0	00000000	00000000
7833	0	00000000	00000000
7834	4	00000100	00000000
7835	8	00001000	00000000
7836	16	00010000	00000000
7837	8	00001000	00000000
7838	4	00000100	00000000
7839	0	00000000	00000000
7840	0	00000000	00000000
7841	0	00000000	00000000
7842	0	00000000	00000000
7843	62	00111110	00111100
7844	0	00000000	00000000
7845	62	00111110	00111100
7846	0	00000000	00000000
7847	0	00000000	00000000
7848	0	00000000	00000000
7849	0	00000000	00000000
7850	8	00001000	00000000
7851	8	00001000	00000000
7852	62	00111110	00111100
7853	8	00001000	00000000
7854	8	00001000	00000000
7855	0	00000000	00000000
7856	0	00000000	00000000
7857	0	00000000	00000000
7858	0	00000000	00000000
7859	62	00111110	00111100
7860	0	00000000	00000000
7861	0	00000000	00000000
7862	0	00000000	00000000
7863	0	00000000	00000000
7864	0	00000000	00000000
7865	0	00000000	00000000
7866	8	00001000	00000000
7867	8	00001000	00000000
7868	62	00111110	00111100
7869	8	00001000	00000000
7870	8	00001000	00000000
7871	0	00000000	00000000

7872	0	00000000	00000000
7873	0	00000000	00000000
7874	2	00000010	00000000
7875	4	00000100	00000000
7876	8	00001000	00000000
7877	16	00010000	00000000
7878	32	00100000	00000000
7879	0	00000000	00000000
7880	0	00000000	00000000
7881	0	00000000	00000000
7882	16	00010000	00000000
7883	0	00000000	00000000
7884	0	00000000	00000000
7885	16	00010000	00000000
7886	16	00010000	00000000
7887	32	00100000	00000000
7888	0	00000000	00000000
7889	0	00000000	00000000
7890	0	00000000	00000000
7891	0	00000000	00000000
7892	0	00000000	00000000
7893	8	00001000	00000000
7894	8	00001000	00000000
7895	16	00010000	00000000
7896	0	00000000	00000000
7897	0	00000000	00000000
7898	0	00000000	00000000
7899	0	00000000	00000000
7900	0	00000000	00000000
7901	24	00011000	00000000
7902	24	00011000	00000000
7903	0	00000000	00000000
7904	0	00000000	00000000
7905	60	00111100	00000000
7906	70	01000110	00000000
7907	74	01001010	00000000
7908	82	01010010	00000000
7909	98	01100010	00000000
7910	60	00111100	00000000
7911	0	00000000	00000000
7912	0	00000000	00000000
7913	24	00011000	00000000
7914	40	00101000	00000000
7915	8	00001000	00000000
7916	8	00001000	00000000
7917	8	00001000	00000000
7918	62	00111110	00111100
7919	0	00000000	00000000
7920	0	00000000	00000000
7921	60	00111100	00111100
7922	66	01000010	00000000
7923	2	00000010	00000000
7924	60	00111100	00111100
7925	64	01000000	00000000
7926	126	01111110	00111100
7927	0	00000000	00000000
7928	0	00000000	00000000
7929	60	00111100	00111100
7930	66	01000010	00000000
7931	12	00001100	00001100
7932	2	00000010	00000000
7933	66	01000010	00000000
7934	60	00111100	00111100
7935	0	00000000	00000000





8064	0	00000000	00000000
8065	68	01000100	0▒000▒00
8066	72	01001000	0▒00▒000
8067	112	01110000	0▒▒▒0000
8068	72	01001000	0▒00▒000
8069	68	01000100	0▒000▒00
8070	66	01000010	0▒0000▒0
8071	0	00000000	00000000
8072	0	00000000	00000000
8073	64	01000000	0▒000000
8074	64	01000000	0▒000000
8075	64	01000000	0▒000000
8076	64	01000000	0▒000000
8077	64	01000000	0▒000000
8078	126	01111110	0▒▒▒▒▒▒0
8079	0	00000000	00000000
8080	0	00000000	00000000
8081	66	01000010	0▒0000▒0
8082	102	01100110	0▒▒00▒▒0
8083	90	01011010	0▒0▒▒0▒0
8084	66	01000010	0▒0000▒0
8085	66	01000010	0▒0000▒0
8086	66	01000010	0▒0000▒0
8087	0	00000000	00000000
8088	0	00000000	00000000
8089	66	01000010	0▒0000▒0
8090	98	01100010	0▒▒000▒0
8091	82	01010010	0▒0▒00▒0
8092	74	01001010	0▒00▒0▒0
8093	70	01000110	0▒000▒▒0
8094	66	01000010	0▒0000▒0
8095	0	00000000	00000000
8096	0	00000000	00000000
8097	60	00111100	00▒▒▒▒00
8098	66	01000010	0▒0000▒0
8099	66	01000010	0▒0000▒0
8100	66	01000010	0▒0000▒0
8101	66	01000010	0▒0000▒0
8102	60	00111100	00▒▒▒▒00
8103	0	00000000	00000000
8104	0	00000000	00000000
8105	124	01111100	0▒▒▒▒▒00
8106	66	01000010	0▒0000▒0
8107	66	01000010	0▒0000▒0
8108	124	01111100	0▒▒▒▒▒00
8109	64	01000000	0▒000000
8110	64	01000000	0▒000000
8111	0	00000000	00000000
8112	0	00000000	00000000
8113	60	00111100	00▒▒▒▒00
8114	66	01000010	0▒0000▒0
8115	66	01000010	0▒0000▒0
8116	82	01010010	0▒0▒00▒0
8117	74	01001010	0▒00▒0▒0
8118	60	00111100	00▒▒▒▒00
8119	0	00000000	00000000
8120	0	00000000	00000000
8121	124	01111100	0▒▒▒▒▒00
8122	66	01000010	0▒0000▒0
8123	66	01000010	0▒0000▒0
8124	124	01111100	0▒▒▒▒▒00
8125	66	01000100	0▒000▒00
8126	66	01000010	0▒0000▒0
8127	0	00000000	00000000

8128	0	00000000	00000000
8129	60	00111100	00▒▒▒▒00
8130	64	01000000	0▒000000
8131	60	00111100	00▒▒▒▒00
8132	2	00000010	00000000
8133	66	01000010	0▒0000▒0
8134	60	00111100	00▒▒▒▒00
8135	0	00000000	00000000
8136	0	00000000	00000000
8137	254	11111110	▒▒▒▒▒▒▒0
8138	16	00010000	000▒0000
8139	16	00010000	000▒0000
8140	16	00010000	000▒0000
8141	16	00010000	000▒0000
8142	16	00010000	000▒0000
8143	0	00000000	00000000
8144	0	00000000	00000000
8145	66	01000010	0▒0000▒0
8146	66	01000010	0▒0000▒0
8147	66	01000010	0▒0000▒0
8148	66	01000010	0▒0000▒0
8149	66	01000010	0▒0000▒0
8150	60	00111100	00▒▒▒▒00
8151	0	00000000	00000000
8152	0	00000000	00000000
8153	66	01000010	0▒0000▒0
8154	66	01000010	0▒0000▒0
8155	66	01000010	0▒0000▒0
8156	66	01000010	0▒0000▒0
8157	36	00100100	00▒00▒00
8158	24	00011000	000▒▒000
8159	0	00000000	00000000
8160	0	00000000	00000000
8161	66	01000010	0▒0000▒0
8162	66	01000010	0▒0000▒0
8163	66	01000010	0▒0000▒0
8164	66	01000010	0▒0000▒0
8165	90	01011010	0▒0▒▒0▒0
8166	36	00100100	00▒00▒00
8167	0	00000000	00000000
8168	0	00000000	00000000
8169	66	01000010	0▒0000▒0
8170	36	00100100	00▒00▒00
8171	24	00011000	000▒▒000
8172	24	00011000	000▒▒000
8173	36	00100100	00▒00▒00
8174	66	01000010	0▒0000▒0
8175	0	00000000	00000000
8176	0	00000000	00000000
8177	130	10000010	▒00000▒0
8178	68	01000100	0▒000▒00
8179	40	00101000	00▒0▒000
8180	16	00010000	000▒0000
8181	16	00010000	000▒0000
8182	16	00010000	000▒0000
8183	0	00000000	00000000
8184	0	00000000	00000000
8185	126	01111110	0▒▒▒▒▒▒0
8186	4	00000100	00000▒00
8187	8	00001000	0000▒000
8188	16	00010000	000▒0000
8189	32	00100000	00▒00000
8190	126	01111110	0▒▒▒▒▒▒0
8191	0	00000000	00000000

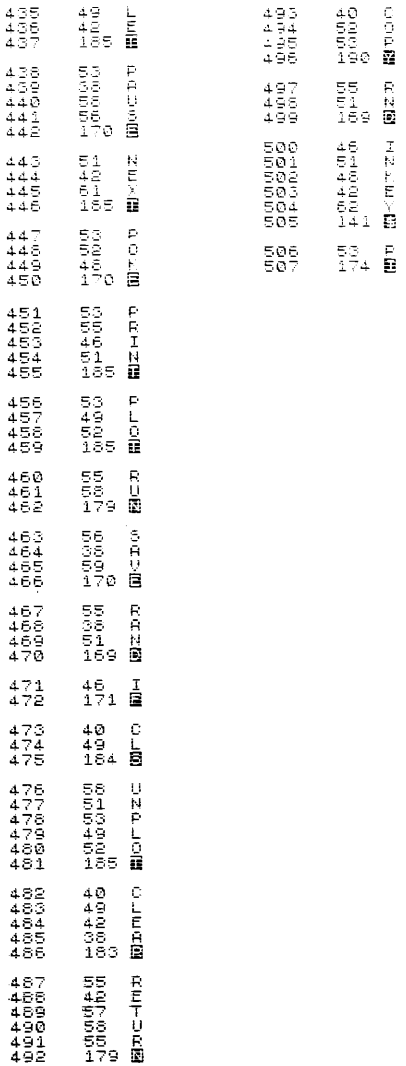
AS PALAVRAS-CHAVE

Estão gravadas nos endereços de 273 a 507 e listadas a seguir na em tres colunas representando: ENDEREÇO, VALOR e CARACTERE . O último caractere, de toda palavra-chave, encontra-se gravado invertido, indicando o fim da mesma. O programa usado:

```
  1 REM MICRON ELETRONICA
 20 FOR N=273 TO 507
 30 LET P=PEEK N
 40 LPRINT N;TAB 6;P;TAB 10;CHR
$ (P)
 50 IF P<99 THEN NEXT N
 80 LPRINT
100 NEXT N
```

273	143	
274	11	
275	139	
276	38	A
277	185	
278	57	T
279	38	A
280	167	
281	143	
282	40	C
283	52	O
284	41	D
285	170	
286	59	V
287	38	A
288	177	
289	49	L
290	42	E
291	179	
292	56	S
293	46	H
294	179	
295	40	C
296	52	O
297	184	
298	57	T
299	38	A
300	179	
301	38	A
302	56	S
303	179	
304	38	A
305	40	C
306	184	
307	38	A
308	57	T
309	179	
310	49	I
311	179	
312	42	E
313	61	=
314	181	
315	46	H
316	61	=
317	185	
318	56	S
319	54	V
320	165	
321	56	S
322	44	L
323	179	

324	38	A
325	56	S
326	184	
327	53	P
328	42	E
329	46	H
330	178	
331	58	U
332	56	S
333	183	
334	56	S
335	57	T
336	55	R
337	141	
338	40	C
339	45	H
340	55	R
341	141	
342	51	N
343	52	O
344	185	
345	23	*
346	151	
347	52	O
348	183	
349	38	A
350	51	N
351	169	
352	19	<
353	148	
354	18	>
355	148	
356	19	<
357	146	
358	57	T
359	45	H
360	42	E
361	179	
362	57	T
363	180	
364	56	S
365	57	T
366	42	E
367	181	
368	49	L
369	53	P
370	55	R
371	46	H
372	51	N
373	185	
374	49	L
375	49	L
376	46	H
377	56	S
378	185	
379	56	S
380	54	V
381	165	
382	56	S
383	44	L
384	179	
385	56	S
386	46	H
387	46	H
388	185	
389	51	N
390	42	E
391	188	
392	56	S
393	40	C
394	55	R
395	52	O
396	49	L
397	177	
398	40	C
399	52	O
400	51	N
401	185	
402	41	D
403	46	H
404	178	
405	55	R
406	42	E
407	178	
408	43	F
409	52	O
410	183	
411	44	G
412	52	O
413	57	T
414	180	
415	44	G
416	52	O
417	56	S
418	58	U
419	167	
420	46	I
421	51	N
422	53	P
423	55	R
424	58	U
425	56	S
426	185	
427	49	L
428	52	O
429	38	A
430	169	
431	49	L
432	46	H
433	56	S
434	185	



Z80

SUMÁRIO DAS INSTRUÇÕES

SUMÁRIO DAS INSTRUÇÕES Z80

Este apêndice apresenta um sumário das instruções do microprocessador Z80 em ordem alfabética, cujas abreviaturas e símbolos empregados são descritos abaixo:

qq	qualquer um dos registers BC, DE, HL ou AF
ss	qualquer um dos registers BC, DE, HL ou SP
pp	qualquer um dos registers BC, DE, IX ou SP
rr	qualquer um dos registers BC, DE, IY ou SP
dd	qualquer um dos registers BC, DE, SP
r ou r'	qualquer um dos registers A, B, C, D, E ,H ou L
n	valor numérico entre Ø e 255 (um byte)
nn	valor numérico entre Ø e 65535 (dois bytes)
(nn)	conteúdo de uma posição de memória dada por nn
(HL)	conteúdo de uma posição de memória dada por HL
m	r, (HL), (IX+d), (IY+d)
s	r, (HL), (IX+d), (IY+d) ou ainda n
d	valor numérico entre -128 e +127
e	valor numérico entre -126 e +129

cc — qualquer uma das condições abaixo, dadas pelos bits do register F

NZ	non zero	Z=Ø
Z	zero	Z=1
NC	non carry	C=Ø
C	carry	C=1
PO	parity odd	P=Ø
PE	parity even	P=1
P	sign positive	S=Ø
M	sign negative	S=1

OBSERVAÇÃO:

As colunas marcadas por S Z P C , referem-se às bandeiras.Respectivamente Signal, Zero, Parity/Overflow e Carry. Seus estados em decorrência da instrução estão representados por:

0 assume valor zero

1 assume valor um

- não é afetada

X depende do resultado da operação

? desconhecida

	S	Z	P	C
ADC A,s Adiciona o operando s e o carry-bit ao conteúdo do register A.	X	X	X	X
ADC HL,ss Adiciona ss e o carry-bit ao conteúdo do register HL.	X	X	X	X
ADD A,n Adiciona n ao register A.	X	X	X	X
ADD A,r Adiciona r ao register A.	X	X	X	X
ADD A,(HL) Adiciona o conteúdo do endereço dado por HL ao conteúdo do register A, o acumulador.	X	X	X	X
ADD A,(IX+d) Adiciona o conteúdo do endereço dado por IX+d ao register A.	X	X	X	X
ADD A,(IY+d) Adiciona o conteúdo do endereço dado por IY+d ao register A.	X	X	X	X
ADD HL,ss Adiciona ss ao conteúdo de HL.	-	-	-	X
ADD IX,pp Adiciona pp ao conteúdo de IX.	-	-	-	X
ADD IY,rr Adiciona rr ao conteúdo de IY.	-	-	-	X
AND s O mesmo que AND A,s. Combina por lógica AND o operando s com o register A.	X	X	X	0
BIT b,(HL) Fornece o valor do bit b do conteúdo do endereço dado por HL.	?	X	X	0
BIT b,(IX+d) Fornece o valor do bit b do conteúdo do endereço dado por IX+d.	?	X	?	-
BIT b,(IY+d) Fornece o valor do bit b do conteúdo do endereço dado por IY+d.	?	X	?	-
BIT b,r Fornece o valor do bit b de r.	?	X	?	-

S Z P C

CALL nn - - - -
Chama a sub-rotina do endereço nn.

CALL cc,nn - - - -
Se a condição cc for verdadeira,chama a sub-rotina do endereço nn.

CCF - - - X
Complementa o carry-bit,invertendo o seu valor.

CP s X X X X
O mesmo que CP A,s . Compara por subtração o operando s com o conteúdo do register A, sem alterar os seus valores. Altera as bandeiras.

CPD . X X X -
O mesmo que CP A,(HL) seguido de DEC BC e DEC HL, i.e., compara o conteúdo do endereço dado por HL com o register A, não alterando os seus valores e sim as bandeiras. Subtrai um dos valores de BC e HL.

CPDR X X X -
O mesmo que CPD, repetindo-se até de o valor de A=(HL) ou BC=Ø .

CPI X X X -
O mesmo que CP A,(HL) seguido de DEC BC e INC HL.

CPIR X X X -
O mesmo que CPDR, exceto que o valor de HL será somado em um, ao invés de subtraído.

CPL - - - -
Complementa o register A. Inverte o valor de todos os seus bits.Os de valor Ø tornam-se 1 e vice-versa.

DAA X X X X
Decimal adjust accumulator. Converte o conteúdo do register A para a forma BCD (binary code decimal).

DEC m X X X -
Subtrai 1 do valor do operando m.

DEC IX - - - -
Subtrai 1 do valor de IX .

DEC IY - - - -
Subtrai 1 do valor de IY.

S Z P C

DEC ss - - - -
Subtrai 1 do valor de ss.

DI - - - -
Disable interrupts. O processador passa a ignorar as instruções de interrupção até uma EI, enable interrupts.

DJNZ e - - - -
Decrement e Jump relative if not zero. Subtrai 1 do valor do register B. Se o resultado não for Ø,o processador pula "e" bytes do programa, onde "e" varia de -126 a +129.

EI - - - -
Enable interrupts. O processador passa a aceitar as instruções de interrupção.

EX AF,AF' - - - -
Troca o conteúdo dos registers AF com AF'.

EX DE,HL - - - -
Troca com conteúdo do register HL com o DE.

EX (SP),HL - - - -
Troca o conteúdo do endereço dado pelo STACK POINTER, (SP), com o conteúdo do register HL.(SP) é o "top of stack".

EX (SP),IX - - - -
Troca o conteúdo de IX com (SP). Vide EX (SP),HL .

EX (SP), IY - - - -
Troca o conteúdo de IY com (SP). Vide EX (SP),HL .

EXX - - - -
Permite trocar o conteúdo dos registers BC, DE e HL com o conteúdo dos seus secundários ou alternativos BC', DE' e HL', respectivamente.É uma única instrução para a troca dos tres pares de registers.

HALT - - - -
A instrução HALT para a execução do programa. Para reiniciar a execução é necessário um reset ou interrupt à CPU. HALT, cujo código é 78h (118 decimal),não deve ser usado nestes micros com lógica Sinclair, sob pena de perda de controle do programa.

IM 0 - - - -
Interrupt mode Ø . Coloca a CPU no modo de interrupção Ø .

S Z P C

IM 1 - - - -
Coloca a CPU no modo de interrupção 1.

IM 2 - - - -
Coloca a CPU no modo de interrupção 2.

IN A,(n) - - - -
Carrega o register A com os dados lidos do dispositivo externo I/O n. Seu uso está totalmente dependente do hardware.

IN r,(C) - - - -
Carrega r com o conteúdo do dispositivo externo determinado pelo register C.

INC (HL) X X X -
Soma um ao valor contido no endereço dado pelo register HL.

INC IX - - - -
Soma um ao valor do register IX .

INC (IX+d) X X X -
Soma um ao valor contido no endereço dado por IX mais o deslocamento d, displacement .

INC IY - - - -
Soma um ao valor do register IY.

INC (IY+d) X X X -
Soma um ao valor contido no endereço dado por IY+d .

INC r X X X -
Soma um ao valor de r.

INC ss - - - -
Soma um ao valor de ss.

IND ? X ? -
Input and decrement pointer. Coloca no endereço dado pelo register HL, o valor lido do dispositivo externo dado pelo register C. Subtrai em um o valor dos registers HL e B.

INDR ? X ? -
O mesmo que IND, repetindo-se até que o valor do register B seja zero. Veja IND. Nestas instruções habitualmente usa-se o register HL para endereço e B como um contador.

S Z P C

INI - - - -
Coloca o valor lido na porta I/O determinada pelo register C,no endereço dado pelo register HL. Subtrai o valor de B em 1 e soma HL em 1. O mesmo que dizer IN (HL),(C) mais DEC B e INC HL.

INIR - - - -
O mesmo que INI,repetindo-se até que o valor do register B seja zero. Instruções do tipo "INput" são comumente usadas nas rotinas de LOAD, em Basic.

JP (HL) - - - -
É uma instrução de deslocamento de programa. O mesmo prossegue a partir do endereço dado pelo register HL. JP é JUMP, pule.

JP (IX) - - - -
O mesmo que JP (HL) para o register IX.

JP (IY) - - - -
O mesmo que JP (HL) para o register IY.

JP cc,nn - - - -
Se a condição cc for verdadeira, prossegue o programa a partir do endereço dado por nn. É uma instrução de tres bytes, sendo um para o código da mesma propriamente dito e dois outros para armazenar o endereço de destino, nn . Vide cc,no início deste capítulo.

JP nn - - - -
Incondicionalmente prossegue o programa a partir do endereço nn.

JR C,e - - - -
O mesmo que JR e, exceto que o JUMP apenas será executado,se o carry-bit for de valor . O carry-bit é um dos bits do register F, utilizado como FLAG REGISTER.

JR e - - - -
Prossegue o programa pulando "e" bytes. JR é JUMP RELATIVE.

JR NC,e - - - -
Se o carry-bit for de valor zero, o programa prossegue,pulando os próximos "e" bytes.JR NC é JUMP RELATIVE IF NON CARRY.

JR NZ,e - - - -
Se o zero-bit for de valor zero,o programa prossegue,pulando os próximos "e" bytes.

S Z P C

JR Z,e - - - -
Se o zero-bit for de valor 1,o programa prossegue,pulando os próximos "e" bytes.Veja as outras instruções JR e JP.

LD A,(BC) - - - -
Carrega o register A com o conteúdo do endereço especificado pelo register BC.

LD A,(DE) - - - -
Carrega o register A com o conteúdo do endereço especificado pelo register DE.

LD A,(HL) - - - -
O mesmo que LD A,(BC) com endereço especificado por HL.

LD A,I - - - -
Carrega o register A com o conteúdo do register I ou ainda R.

LD A,(nn) - - - -
Copia no register A o conteúdo do endereço nn.

LD A,r - - - -
Copia no register A o conteúdo de r.

LD (BC),A - - - -
O conteúdo do register A é copiado para o endereço dado por BC. O endereço também pode ser especificado pelos registers DE ou HL.

LD (HL), r - - - -
O conteúdo de r ou ainda de uma constante n podem ser copiados para o endereço dado por HL.

LD dd,nn - - - -
Copia para dd o valor nn, especificado em dois bytes.

LD HL,(nn) - - - -
O conteúdo do endereço nn é copiado para HL.

LD I,A - - - -
O conteúdo do register A pode ser copiado para o register I ou ainda para o register R.

LD IX,nn - - - -
Carrega IX com nn. Existe também para IY.

LD IX,(nn) - - - -
Carrega IX com o conteúdo do endereço nn. Existe com IY.

LD (IX+d),n S Z P C
Carrega o endereço especificado pelo valor do register IX mais o deslocamento d indicado com o valor n ou ainda com r. - - - -

LD (IY+d),n
O mesmo que LD (IX+d),n,usando IY ao invés de IX. Válido ainda com r. - - - -

LD (nn),A
Copia no endereço nn o valor do register A. - - - -

LD (nn),dd
Copia dd no endereço nn. - - - -

LD (nn),HL
O valor de HL é copiado no endereço dado por nn.Válido com IX e IY. - - - -

LD r,r'
Instruções que copiam os conteúdos dos alternativos r,r', nos próprios r. - - - -

LD r,n
Carrega r com a constante n. - - - -

LD r,(HL)
Carrega r com o valor contido no endereço especificado por HL.Existem opções com endereços dados por IX+d ou IY+d - - - -

LD SP,HL
Carrega o register SP, STACK POINTER, com o valor do register HL. Oferece opções para copiar no SP,o register IX e IY.

LDD
Transfere o valor contido no endereço especificado por HL,para o endereço dado por DE. Em seguida, subtrai um nos valores dos registers HL, DE e BC. Se BC-1 for diferente de zero, o bit de paridade será 1,ou em caso contrário será Ø . LDD é uma abreviatura de LOAD e DECREMENT. - - X -

LDDR
O mesmo que LDD, repetindo-se até que o valor de BC seja zero.Altera o bit de paridade da mesma forma que LDD. - - X -

LDI
O mesmo que LDD, exceto que os valores dos registers HL e DE serão somados em 1,ao invés de subtraídos. Altera o bit de paridade da mesma forma que LDD. - - X -

S Z P C

LDIR - - X -

O mesmo que LDI, repetindo-se até que o valor contido no register BC seja zero. A bandeira de paridade é alterada como em LDD.

NEG X X X X

NEGATE negativa o valor contido no register A, como se o mesmo tivesse sido multiplicado por -1, ou o mesmmo que subtraído de zero.

NOP - - - -

NO OPERATION é uma instrução de não execução. Não faz absolutamente nada, exceto que o register PC, program counter, será incrementado em 1, passando o programa à instrução do endereço seguinte.

OR s X X X 0

Combina por lógica OR o operando s com o register A.

OUT (C),r - - - -

Coloca na porta de saída dada pelo register C, o conteúdo de r.

OUTD ? X ? -

Coloca o conteúdo do endereço dado por HL na porta de saída I/O, endereçada pelo register C. Subtrai o valor de HL e B em 1

OTDR ? X ? -

O mesmo que OUTD, repetindo-se até que B atinja o valor zero.

OUTI ? X ? -

Coloca o conteúdo da posição de memória especificada por HL na porta de saída I/O, endereçada pelo register C. Subtrai 1 do valor do register B e soma 1 ao valor de HL.

OTIR ? X ? -

O mesmo que OUTI, repetindo-se até que o valor do register B assuma o valor 0 .

OUT (n),A - - - -

Coloca na porta de saída I/O, dada por n, o valor contido no register A.

POP IX - - - -

Coloca no register IX, o valor contido nos dois primeiros bytes do topo do STACK. O endereço do STACK é dado pelo register SP, o STACK POINTER. Veja o "STACK",no capítulo "ARQUITETURA DO Z80".

POP IY - - - -

O mesmo que POP IX, exceto que com o register IY ao invés de IX.

S Z P C

POP qq - - - -

O mesmo que POP IX, exceto que oferece a variedade de qq.

PUSH IX - - - -

O oposto de POP IX. Coloca o valor do register IX no topo do STACK. O endereço do STACK é dado pelo register SP. Veja POP IX.

PUSH IY - - - -

O mesmo que PUSH IX, exceto que com o register IY.

PUSH qq - - - -

O mesmo que PUSH IX, exceto que oferece a variedade de qq.

RES b,r - - - -

Zera o bit b de r. Bit 0 é o menos significativo.

RES b,(HL) - - - -

Zera o bit b do conteúdo do endereço dado por HL.

RES b,(IX+d) - - - -

Zera o bit b do endereço dado pelo valor do register IX somado ao deslocamento indicado d.

RES b,(IY+d) - - - -

O mesmo que RES b,(IX+d), exceto que usa o valor de IY para determinar endereço.

RET - - - -

É RETURN. Instrução de retorno, de sub-rotina. Durante as sub-rotinas não se deve alterar o STACK, pois o processador coloca o endereço de retorno, contido no PC, Program Counter, no STACK.

RET cc - - - -

Se a condição cc for verdadeira,retorna da sub-rotina.

RETI - - - -

Retorno de rotina de interrupção. Instrução para uso com dispositivos de entrada e saída (I/O).

RETN - - - -

Retorno de rotina de interrupção non-maskable. Vide RETI.

RL m X X X X

Move todos os bits de m uma posição à esquerda. O bit da primeira posição à direita assume o valor do carry-bit e o bit excedente da primeira posição à esquerda é enviado ao carry-bit. RL são as iniciais de ROTATE LEFT.

S Z P C

RLA - - - X

O mesmo que RL, mas aplicando-se apenas ao register A. Não altera as bandeiras, exceto o carry-bit. Na instrução RL m, o operando m pode ser qualquer um dos registers, inclusive o A, existindo portando RL A, que difere de RLA, basicamente por ocupar dois bytes ao invés de um e porque altera todas as bandeiras. Vide RL m .

RLC (HL) X X X X

Move todos os bits do conteúdo do endereço dado por HL, uma posição à esquerda. O bit excedente, primeira posição à esquerda, é enviado ao carry-bit e ao bit da primeira posição à direita. RLC é a abreviatura de ROTATE LEFT CIRCULAR.

RLC (IX+d) X X X X

O mesmo que RLC (HL), exceto que move os bits do conteúdo do endereço dado pelo valor de IX somado ao deslocamento indicado d .

RLC (IY+d) X X X X

O mesmo que RLC (IX+d), com o uso de IY.

RLC r X X X X

O mesmo que ROTATE LEFT CIRCULAR r. Move todos os bits de r uma posição à esquerda. O bit da primeira posição à esquerda é enviado ao carry-bit e ao primeiro bit à direita.

RLCA - - - X

O mesmo que RLC r, exceto que se aplica apenas a register A e altera apenas o carry-bit, além do que ocupa apenas um byte ao invés de dois, com RLC A.

RLD X X X -

É uma instrução para manipulação de dígitos BCD (binary code decimal). Os quatro bits menos significativos de uma posição de memória especificada por HL são copiados na mesma ordem nos quatro bits mais significativos da mesma posição de memória. Os anteriores quatro bits mais significativos da posição de memória especificada por HL são copiados nos quatro bits menos significativos do register A. Os anteriores quatro bits,menos significativos de A são copiados nos quatro bits menos significativos da posição de memória especificada por HL.

RR m X X X X

O mesmo que RL m, exceto que os bits são movidos uma posição à direita. RR é ROTATE RIGHT. Move os bits de m uma posição à direita. O bit da primeira posição à direita, excedente, é enviado ao carry e o valor anterior do carry-bit é enviado ao primeiro bit à esquerda. Vide RL m .

S Z P C

RRA - - - X

O mesmo que RLA, exceto que os bits são movidos uma posição à direita. Difere de RR A que,pelas razões de RLA,difere de RL A. Vide RL m, RR m e RLA.

RRC m X X X X

O mesmo que RLC m, exceto que os bits são movidos uma posição à direita.

RRCA - - - X

O mesmo que RLCA, exceto que os bits são movidos uma posição à direita. RRCA é ROTATE ACCUMULATOR RIGHT CIRCULAR.

RRD X X X -

Semelhante a RLD, também para dígitos na forma BCD. Os quatro bits mais significativos de uma posição de memória especificada por HL são copiados na mesma ordem, nos quatro bits menos significativos desta mesma posição de memória. Os anteriores quatro bits menos significativos desta posição de memória são copiados nos quatro dígitos menos significativos do register A.Os anteriores quatro bits menos significativos de A são copiados nos quatro bits mais significativos da posição de memória especificada por HL.

RST n - - - -

É abreviatura de RESTART. Chama uma sub-rotina nos primeiros endereços da memória. Só é possivel RST 0, 08, 10, 18, 20, 28 30 e 38. Sua principal vantagem em relação a CALL é ocupar apenas um byte, enquanto CALL usa tres, incluindo os dois para o endereço.

SBC A,s X X X X

Subtrai o operando s e o carry-bit do valor contido no register A.

SBC HL,ss X X X X

Subtrai o operando ss e o carry-bit do valor contido no register HL. As operações de soma e subtração em 16 bits sempre encerram os resultàdos no register HL.SBC é SUBTRACT WITH CARRY.

SCF - - - 1

Faz com que o carry-bit assuma o valor um, independentemente de seu valor anterior. SCF é SET CARRY FLAG.

SET b,(HL) - - - -

Faz com que o bit b do endereço dado por HL assuma o valor um.

SET b,(IX+d) - - - -

Faz com que o bit b do endereço dado pela soma do valor do register IX com o deslocamento indicado d.

SET b,(IY+d) - - - -

Faz com que o bit b do valor contido no endereço IY+d assuma o valor um.

SET b,r - - - -

Faz com que o bit b do operando r assuma o valor um.

SLA m X X X X

Move todos os bits de m uma posição à esquerda. O bit excedente da primeira posição à esquerda é enviado ao carry-bit e o primeiro bit à direita assume o valor zero. SLA são as iniciais de SHIFT LEFT ARITHMETIC.

SRA m X X X X

Move os bits do operando m uma posição à direita. O bit excedente à direita é enviado ao carry-bit. O bit mais significativo, o primeiro à esquerda, permanece inalterado.

SRL m X X X X

Move os bits de m uma posição à direita. O bit excedente à direita é enviado ao carry-bit e o primeiro bit à esquerda. O mais significativo é zerado. SRL são as iniciais de SHIFT RIGHT LOGICAL.

SUB s X X X X

Subtrai o operando s do valor contido no register A. Algumas vezes é encontrada na forma SUB A,s , o que é a mesma coisa.

XOR s X X X 0

Executa a operação lógica Exclusive-OR do operando s com o register A.

TABELAS

HEX X MNEMÔNICOS

00	NOP
01 XX XX	LD BC,NN
02	LD (BC),A
03	INC BC
04	INC B
05	DEC B
06 XX	LD B,N
07	RLCA
08	EX AF,AF'
09	ADD HL,BC
0A	LD A,(BC)
0B	DEC BC
0C	INC C
0D	DEC C
0E XX	LD C,N
0F	RRCA
10 XX	DJNZ d
11 XX XX	LD DE,NN
12	LD (DE),A
13	INC DE
14	INC D
15	DEC D
16 XX	LD D,N
17	RLA
18 XX	JR d
19	ADD HL,DE
1A	LD A,(DE)
1B	DEC DE
1C	INC E
1D	DEC E
1E XX	LD E,N
1F	RRA
20 XX	JR NZ,d
21 XX XX	LD HL,NN
22 XX XX	LD (NN),HL
23	INC HL
24	INC H
25	DEC H
26 XX	LD H,N
27	DAA
28 XX	JR Z,d
29	ADD HL,HL
2A XX XX	LD HL,(NN)
2B	DEC HL
2C	INC L
2D	DEC L
2E XX	LD L,N
2F	CPL
30 XX	JR NC,d
31 XX XX	LD SP,NN
32 XX XX	LD (NN),A
33	INC SP
34	INC (HL)
35	DEC (HL)
36 20 XX	LD (HL),N
37	SCF
38 XX	JR C,d
39	ADD HL,SP
3A XX XX	LD A,(NN)
3B	DEC SP
3C	INC A
3D	DEC A
3E XX	LD A,N
3F	CCF
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
46	LD B,(HL)
47	LD B,A
48	LD C,B
49	LD C,C
4A	LD C,D
4B	LD C,E
4C	LD C,H
4D	LD C,L
4E	LD C,(HL)
4F	LD C,A
50	LD D,B
51	LD D,C

52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
56	LD D,(HL)
57	LD D,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
5E	LD E,(HL)
5F	LD E,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
66	LD H,(HL)
67	LD H,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
6E	LD L,(HL)
6F	LD L,A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
76	HALT
77	LD (HL),A
78	LD A,B
79	LB A,C
7A	LD A,D

7B	LD A,E
7C	LD A,H
7D	LD A,L
7E	LD A,(HL)
7F	LD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
86	ADD A,(HL)
87	ADD A,A
88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,E
8D	ADC A,L
8E	ADC A,(HL)
8F	ADC A,A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB (HL)
97	SUB A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
9E	SBC A,(HL)
9F	SBC A,A
A0	AND B
A1	AND C
A2	AND D
A3	AND E

A4	AND H
A5	AND L
A6	AND (HL)
A7	AND A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR (HL)
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C2 XX XX	JP NZ,NN
C3 XX XX	JP NN
C4 XX XX	CALL NZ,NN
C5	PUSH BC
C6 XX	ADD A,N
C7	RST 0
C8	RET Z
C9	RET
CA XX XX	JP Z,NN
CB 00	APÓS FF

CC XX XX	CALL Z,NN
CD XX XX	CALL NN
CE XX	ADC A,N
CF	RST 8
D0	RET NC
D1	POP DE
D2 XX XX	JP NC,NN
D3 XX	OUT (N),A
D4 XX XX	CAL NC,NN
D5	PUSH DE
D6 XX	SUB N
D7	RST 10H
D8	RET C
D9	EXX
DA XX XX	JP C,NN
DB XX	IN A,(N)
DC XX XX	CALL C,NN
DE XX	SBC A,N
DF	RST 18H
E0	RET PO
E1	POP HL
E2 XX XX	JP PO,NN
E3	EX (SP),HL
E4 XX XX	CALL PO,NN
E5	PUSH HL
E6 XX	AND N
E7	RST 20H
E8	RET PE
E9	JP (HL)
EA XX XX	JE PE NN
EB	EX DE,HL
EC XX XX	CALL PE,NN
EE XX	XOR N
EF	RST 28H
F0	RET P
F1	POP AF
F2 XX XX	JP P,NN
F3	DI
F4 XX XX	CALL P,NN
F5	PUSH AF
F6 XX	OR N

F7	RST 30
F8	RET M
F9	LD SP,HL
FA XX XX	JP M,NN
FB	EI
FC XX XX	CALL M,NN
FE XX	CP N
FF	RST 38H
CB 00	RLC B
CB 01	RLC C
CB 02	RLC D
CB 03	RLC E
CB 04	RLC H
CB 05	RLC L
CB 06	RLC (HL)
CB 07	RLC A
CB 08	RRC B
CB 09	RRC C
CB 0A	RRC D
CB 0B	RRC E
CB 0C	RRC H
CB 0D	RRC L
CB 0E	RRC (HL)
CB 0F	RRC A
CB 10	RL B
CB 11	RL C
CB 12	RL D
CB 13	RL E
CB 14	RL H
CB 15	RL L
CB 16	RL (HL)
CB 17	RL A
CB 18	RR B
CB 19	RR C
CB 1A	RR D
CB 1B	RR E
CB 1C	RR H
CB 1D	RR L
CB 1E	RR (HL)
CB 1F	RR A
CB 20	SLA B
CB 21	SLA C
CB 22	SLA D
CB 23	SLA E
CB 24	SLA H
CB 25	SLA L
CB 26	SLA (HL)
CB 27	SLA A
CB 28	SRA B
CB 29	SRA C
CB 2A	SRA D
CB 2B	SRA E
CB 2C	SRA H
CB 2D	SRA L
CB 2E	SRA (HL)
CB 2F	SRA A
CB 38	SRL B
CB 39	SRL C
CB 3A	SRL D
CB 3B	SRL E
CB 3C	SRL H
CB 3D	SRL L
CB 3E	SRL (HL)
CB 3F	SRL A
CB 40	BIT 0,B
CB 41	BIT 0,C
CB 42	BIT 0,D
CB 43	BIT 0,E
CB 44	BIT 0,H
CB 45	BIT 0,L
CB 46	BIT 0,(HL)
CB 47	BIT 0,A
CB 48	BIT 1,B
CB 49	BIT 1,C
CB 4A	BIT 1,D
CB 4B	BIT 1,E
CB 4C	BIT 1,H
CB 4D	BIT 1,L
CB 4E	BIT 1,(HL)
CB 4F	BIT 1,A
CB 50	BIT 2,B
CB 51	BIT 2,C
CB 52	BIT 2,D
CB 53	BIT 2,E

CB 54	BIT 2,H
CB 55	BIT 2,L
CB 56	BIT 2,(HL)
CB 57	BIT 2,A
CB 58	BIT 3,B
CB 59	BIT 3,C
CB 5A	BIT 3,D
CB 5B	BIT 3,E
CB 5C	BIT 3,H
CB 5D	BIT 3,L
CB 5E	BIT 3,(HL)
CB 5F	BIT 3,A
CB 60	BIT 4,B
CB 61	BIT 4,C
CB 62	BIT 4,D
CB 63	BIT 4,E
CB 64	BIT 4,H
CB 65	BIT 4,L
CB 66	BIT 4,(HL)
CB 67	BIT 4,A
CB 68	BIT 5,B
CB 69	BIT 5,C
CB 6A	BIT 5,D
CB 6B	BIT 5,E
CB 6C	BIT 5,H
CB 6D	BIT 5,L
CB 6E	BIT 5,(HL)
CB 6F	BIT 5,A
CB 70	BIT 6,B
CB 71	BIT 6,C
CB 72	BIT 6,D
CB 73	BIT 6,E
CB 74	BIT 6,H
CB 75	BIT 6,L
CB 76	BIT 6,(HL)
CB 77	BIT 6,A
CB 78	BIT 7,B
CB 79	BIT 7,C
CB 7A	BIT 7,D
CB 7B	BIT 7,E
CB 7C	BIT 7,H
CB 7D	BIT 7,L
CB 7E	BIT 7,(HL)
CB 7F	BIT 7,A
CB 80	RES 0,B
CB 81	RES 0,C
CB 82	RES 0,D
CB 83	RES 0,E
CB 84	RES 0,H
CB 85	RES 0,L
CB 86	RES 0,(HL)
CB 87	RES 0,A
CB 88	RES 1,B
CB 89	RES 1,C
CB 8A	RES 1,D
CB 8B	RES 1,E
CB 8C	RES 1,H
CB 8D	RES 1,L
CB 8E	RES 1,(HL)
CB 8F	RES 1,A
CB 90	RES 2,B
CB 91	RES 2,C
CB 92	RES 2,D
CB 93	RES 2,E
CB 94	RES 2,H
CB 95	RES 2,L
CB 96	RES 2,(HL)
CB 97	RES 2,A
CB 98	RES 3,B
CB 99	RES 3,C
CB 9A	RES 3,D
CB 9B	RES 3,E
CB 9C	RES 3,H
CB 9D	RES 3,L
CB 9E	RES 3,(HL)
CB 9F	RES 3,A
CB A0	RES 4,B
CB A1	RES 4,C
CB A2	RES 4,D
CB A3	RES 4,E
CB A4	RES 4,H
CB A5	RES 4,L

```
CB A6    RES 4,(HL)
CB A7    RES 4,A
CB A8    RES 5,B
CB A9    RES 5,C
CB AA    RES 5,D
CB AB    RES 5,E
CB AC    RES 5,H
CB AD    RES 5,L
CB AE    RES 5,(HL)
CB AF    RES 5,A
CB B0    RES 6,B
CB B1    RES 6,C
CB B2    RES 6,D
CB B3    RES 6,E
CB B4    RES 6,H
CB B5    RES 6,L
CB B6    RES 6,(HL)
CB B7    RES 7,A
CB B8    RES 7,B
CB B9    RES 7,C
CB BA    RES 7,D
CB BB    RES 7,E
CB BC    RES 7,H
CB BD    RES 7,L
CB BE    RES 7,(HL)
CB BF    RES 7,A
CB C0    SET 0,B
CB C1    SET 0,C
CB C2    SET 0,D
CB C3    SET 0,E
CB C4    SET 0,H
CB C5    SET 0,L
CB C6    SET 0,(HL)
CB C7    SET 0,A
CB C8    SET 1,B
CB C9    SET 1,C
CB CA    SET 1,D
CB CB    SET 1,E
CB CC    SET 1,H
CB CD    SET 1,L
CB CE    SET 1,(HL)
CB CF    SET 1,A
CB D0    SET 2,B
CB D1    SET 2,C
CB D2    SET 2,D
CB D3    SET 2,E
CB D4    SET 2,H
CB D5    SET 2,L
CB D6    SET 2,(HL)
CB D7    SET 2,A
CB D8    SET 3,B
CB D9    SET 3,C
CB DA    SET 3,D
CB DB    SET 3,E
CB DC    SET 3,H
CB DD    SET 3,L
CB DE    SET 3,(HL)
CB DF    SET 3,A
CB E0    SET 4,B
CB E1    SET 4,C
CB E2    SET 4,D
CB E3    SET 4,E
CB E4    SET 4,H
CB E5    SET 4,L
CB E6    SET 4,(HL)
CB E7    SET 4,A
CB E8    SET 5,B
CB E9    SET 5,C
CB EA    SET 5,D
CB EB    SET 5,E
CB EC    SET 5,H
CB ED    SET 5,L
CB EE    SET 5,(HL)
CB EF    SET 5,A
CB F0    SET 6,B
CB F1    SET 6,C
CB F2    SET 6,D
CB F3    SET 6,E
CB F4    SET 6,H
CB F5    SET 6,L
CB F6    SET 6,(HL)
CB F7    SET 6,A
```



```
CB F8       SET 7,B
CB F9       SET 7,C
CB FA       SET 7,D
CB FB       SET 7,E
CB FC       SET 7,H
CB FD       SET 7,L
CB FE       SET 7,(HL)
CB FF       SET 7,A
DD 09       ADD IX,BC
DD 19       ADD IX,DE
DD 21 XX XX LD IX,NN
DD 22 XX XX LD (NN),IX
DD 23       INC IX
DD 29       ADD IX,IX
DD 2A XX XX LD IX,(NN)
DD 2B       DEC IX
DD 34 XX    INC (IX+d)
DD 35 XX    DEC (IX+d)
DD 36 XX XX LD (IX+d),N
DD 39       ADD IX,SP
DD 46 XX    LD B,(IX+d)
DD 4E XX    LD C,(IX+d)
DD 56 XX    LD D,(IX+d)
DD 5E XX    LD E,(IX+d)
DD 66 XX    LD H,(IX+d)
DD 6E XX    LD L,(IX+d)
DD 70 XX    LD (IX+d),B
DD 71 XX    LD (IX+d),C
DD 72 XX    LD (IX+d),D
DD 73 XX    LD (IX+d),E
DD 74 XX    LD (IX+d),H
DD 75 XX    LD (IX+d),L
DD 77 XX    LD (IX+d),A
DD 7E XX    LD A,(IX+d)
DD 86 XX    ADD A,(IX+d)
DD 8E XX    ADC A,(IX+d)
DD 96 XX    SUB A,(IX+d)
DD 9E XX    SBC A,(IX+d)
DD A6 XX    AND (IX+d)
DD AE XX    XOR (IX+d)
DD B6 XX    OR (IX+d)
DD BE XX    CP (IX+d)
DD E1       POP IX
DD E3       EX (SP),IX
DD E5       PUSH IX
DD E9       JP (IX)
DD EB       EX DE,IX
DD F9       LD SP,IX
DD CB XX 06 RLC (IX+d)
DD CB XX 0E RRC (IX+d)
DD CB XX 16 RL (IX+d)
DD CB XX 1E RR (IX+d)
DD CB XX 26 SLA (IX+d)
DD CB XX 2E SRA (IX+d)
DD CB XX 3E SRL (IX+d)
DD CB XX 46 BIT 0,(IX+d)
DD CB XX 4E BIT 1,(IX+d)
DD CB XX 56 BIT 2,(IX+d)
DD CB XX 5E BIT 3,(IX+d)
DD CB XX 66 BIT 4,(IX+d)
DD CB XX 6E BIT 5,(IX+d)
DD CB XX 76 BIT 6,(IX+d)
DD CB XX 7E BIT 7,(IX+d)
DD CB XX 86 RES 0,(IX+d)
DD CB XX 8E RES 1,(IX+d)
DD CB XX 96 RES 2,(IX+d)
DD CB XX 9E RES 3,(IX+d)
DD CB XX A6 RES 4,(IX+d)
DD CB XX AE RES 5,(IX+d)
DD CB XX B6 RES 6,(IX+d)
DD CB XX BE RES 7,(IX+d)
DD CB XX C6 SET 0,(IX+d)
DD CB XX CE SET 1,(IX+d)
DD CB XX D6 SET 2,(IX+d)
DD CB XX DE SET 3,(IX+d)
DD CB XX E6 SET 4,(IX+d)
DD CB XX EE SET 5,(IX+d)
DD CB XX F6 SET 6,(IX+d)
DD CB XX FE SET 7,(IX+d)
ED 40       IN B,(C)
ED 41       OUT (C),B
ED 42       SBC HL,BC
```

```
ED 43 XX XX  LD (NN),BC
ED 44        NEG
ED 45        RETN
ED 46        IM 0
ED 47        LD I,A
ED 48        IN C,(C)
ED 49        OUT (C),C
ED 4A        ADC HL,BC
ED 4B XX XX  LD BC,NN
ED 4D        RETI
ED 50        IN D,(C)
ED 51        OUT (C),D
ED 52        SBC HL,DE
ED 53 XX XX  LD (NN),DE
ED 56        IM 1
ED 57        LD A,I
ED 58        IN E,(C)
ED 59        OUT (C),E
ED 5A        ADC HL,DE
ED 5B XX XX  LD DE,(NN)
ED 5E        IM 2
ED 60        IN H,(C)
ED 61        OUT (C),H
ED 62        SBC HL,HL
ED 67        RRD
ED 68        IN 1,(C)
ED 69        OUT (C),L
ED 6A        ADC HL,HL
ED 6F        RLD
ED 72        SBC HL,SP
ED 73 XX XX  LD (NN),SP
ED 78        IN A,(C)
ED 79        OUT (C),A
ED 7A        ADC HL,SP
ED 7B XX XX  LD SP,(NN)
ED A0        LDI
ED A1        CPI
ED A2        INI
ED A3        OUTI
ED A8        LDD
ED A9        CPD
ED AA        IND
ED B0        LDIR
ED B1        CPIR
ED B2        INIR
ED B3        OTIR
ED B8        LDDR
ED B9        CPDR
ED BA        INDR
ED BB        OTDR
FD 09        ADD IY,BC
FD 21 XX XX  LD IY,NN
FD 22 XX XX  LD (NN),IY
FD 23        INC IY
FD 29        ADD IY,IY
FD 2A XX XX  LD IY,(NN)
FD 2B        DEC IY
FD 34 XX     INC (IY+d)
FD 35 XX     DEC (IY+d)
FD 36 XX XX  LD (IY+d),N
FD 39 XX     ADD IY,SP
FD 46 XX     LD B,(IY+d)
FD 4E XX     LD C,(IY+d)
FD 56 XX     LD D,(IY+d)
FD 5E XX     LD E,(IY+d)
FD 66 XX     LD H,(IY+d)
FD 6E XX     LD L,(IY+d)
FD 70 XX     LD (IY+d),B
FD 71 XX     LD (IY+d),C
FD 72 XX     LD (IY+d),D
FD 73 XX     LD (IY+d),E
FD 74 XX     LD (IY+d),H
FD 75 XX     LD (IY+d),L
FD 77 XX     LD (IY+d),A
FD 7E XX     LD A,(IY+d)
FD 86 XX     ADD A,(IY+d)
FD 8E XX     ADC A,(IY+d)
FD 96 XX     SUB A,(IY+d)
FD 9E XX     SBC A,(IY+d)
FD A6 XX     AND (IY+d)
FD AE XX     XOR (IY+d)
FD B6 XX     OR (IY+d)
```



```
FD BE XX     CP (IY+d)
FD E1        POP IY
FD E3        EX (SP),IY
FD E5        PUSH IY
FD E9        JP (IY)
FD EB        EX DE,IY
FD F9        LD SP,IY
FD CB XX 06  RLC (IY+d)
FD CB XX 0E  RRC (IY+d)
FD CB XX 16  RL (IY+d)
FD CB XX 1E  RR (IY+d)
FD CB XX 26  SLA (IY+d)
FD CB XX 2E  SRA (IY+d)
FD CB XX 3E  SRL (IY+d)
FD CB XX 46  BIT 0,(IY+d)
FD CB XX 4E  BIT 1,(IY+d)
FD CB XX 56  BIT 2,(IY+d)
FD CB XX 5E  BIT 3,(IY+d)
FD CB XX 66  BIT 4,(IY+d)
FD CB XX 6E  BIT 5,(IY+d)
FD CB XX 76  BIT 6,(IY+d)
FD CB XX 7E  BIT 7,(IY+d)
FD CB XX 86  RES 0,(IY+d)
FD CB XX 8E  RES 1,(IY+d)
FD CB XX 96  RES 2,(IY+d)
FD CB XX 9E  RES 3,(IY+d)
FD CB XX A6  RES 4,(IY+d)
FD CB XX AE  RES 5,(IY+d)
FD CB XX B6  RES 6,(IY+d)
FD CB XX BE  RES 7,(IY+d)
FD CB XX C6  SET 0,(IY+d)
FD CB XX CE  SET 1,(IY+d)
FD CB XX D6  SET 2,(IY+d)
FD CB XX DE  SET 3,(IY+d)
FD CB XX E6  SET 4,(IY+d)
FD CB XX EE  SET 5,(IY+d)
FD CB XX F6  SET 6,(IY+d)
FD CB XX FE  SET 7,(IY+d)
```

MNEMÔNICOS X HEX

ADC A,(HL)	8E
ADC A,(IX+d)	DD 8E XX XX
ADC A,(IY+d)	FD 8E XX
ADC A,A	8F
ADC A,B	88
ADC A,C	89
ADC A,D	8A
ADC A,E	8B
ADC A,H	8C
ADC A,L	8D
ADC A,N	CE XX
ADC HL,BC	ED 4A
ADC HL,DE	ED 5A
ADC HL,HL	ED 6A
ADC HL,SP	ED 7A
ADD A,(HL)	86
ADD A,(IX+d)	DD 86 XX
ADD A,(IY+d)	FD 86 XX
ADD A,A	87
ADD A,B	80
ADD A,C	81
ADD A,D	82
ADD A,E	83
ADD A,H	84
ADD A,L	85
ADD A,N	C6 XX
ADD HL,BC	09
ADD HL,DE	19
ADD HL,HL	29
ADD HL,SP	39
ADD IX,BC	DD 09
ADD IX,DE	DD 19
ADD IX,IX	DD 29
ADD IX,SP	DD 39
ADD IY,BC	FD 09
ADD IY,DE,	FD 19
ADD IY,IY	FD 29
ADD IY,SP	FD 39
AND (HL)	A6
AND (IX+d)	DD A6 XX
AND (IY+d)	FD A6 XX
AND A	A7
AND B	A0
AND C	A1
AND D	A2
AND E	A3
AND H	A4
AND L	A5
AND N	E6 XX
BIT 0,(HL)	CB 46
BIT 0,(IX+d)	DD CB XX 46
BIT 0,(IY+d)	FD CB XX 46
BIT 0,A	CB 47
BIT 0,B	CB 40
BIT 0,C	CB 41
BIT 0,D	CB 42
BIT 0,E	CB 43
BIT 0,H	CB 44
BIT 0,L	CB 45
BIT 1,(HL)	CB 4E
BIT 1,(IX+d)	DD CB XX 4E
BIT 1,(IY+d)	FD CB XX 4E
BIT 1,A	CB 4F
BIT 1,B	CB 48
BIT 1,C	CB 49
BIT 1,D	CB 4A
BIT 1,E	CB 4B
BIT 1,H	CB 4C
BIT 1,L	CB 4D
BIT 2,(HL)	CB 56
BIT 2,(IX+d)	DD CB XX 56
BIT 2,(IY+d)	FD CB XX 56
BIT 2,A	CB 57
BIT 2,B	CB 50
BIT 2,C	CB 51
BIT 2,D	CB 52
BIT 2,E	CB 53
BIT 2,H	CB 54
BIT 2,L	CB 55
BIT 3,(HL)	CB 5E
BIT 3,(IX+d)	DD CB XX 5E
BIT 3,(IY+d)	FD CB XX 5E

BIT 3,A	CB 5F
BIT 3,B	CB 58
BIT 3,C	CB 59
BIT 3,D	CB 5A
BIT 3,E	CB 5B
BIT 3,H	CB 5C
BIT 3,L	CB 5D
BIT 4,(HL)	CB 66
BIT 4,(IX+d)	DD CB XX 66
BIT 4,(IY+d)	FD CB XX 66
BIT 4,A	CB 67
BIT 4,B	CB 60
BIT 4,C	CB 61
BIT 4,D	CB 62
BIT 4,E	CB 63
BIT 4,H	CB 64
BIT 4,L	CB 65
BIT 5,(HL)	CB 6E
BIT 5,(IX+d)	DD CB XX 6E
BIT 5,(IY+d)	FD CB XX 6E
BIT 5,A	CB 6F
BIT 5,B	CB 68
BIT 5,C	CB 69
BIT 5,D	CB 6A
BIT 5,E	CB 6B
BIT 5,H	CB 6C
BIT 5,L	CB 6D
BIT 6,(HL)	CB 76
BIT 6,(IX+d)	DD CB XX 76
BIT 6,(IY+d)	FD CB XX 76
BIT 6,A	CB 77
BIT 6,B	CB 70
BIT 6,C	CB 71
BIT 6,D	CB 72
BIT 6,E	CB 73
BIT 6,H	CB 74
BIT 6,L	CB 75
BIT 7,(HL)	CB 7E
BIT 7,(IX+d)	DD CB XX 7E
BIT 7,(IY+d)	FD CB XX 73
BIT 7,A	CB 7F
BIT 7,B	CB 78
BIT 7,C	CB 79
BIT 7,D	CB 7A
BIT 7,E	CB 7B
BIT 7,H	CB 7C
BIT 7,L	CB 7D
CALL ADDR	CD XX XX
CALL C,ADDR	DC XX XX
CALL M,ADDR	FC XX XX
CALL NC,ADDR	D4 XX XX
CALL NZ,ADDR	C4 XX XX
CALL P,ADDR	F4 XX XX
CALL PE,ADDR	EC XX XX
CALL PO,ADDR	E4 XX XX
CALL Z,ADDR	OC XX XX
CCF	3F
CP (HL)	BE
CP (IX+d)	DD BE XX
CP (IY+d)	FD BE XX
CP A	BF
CP B	B8
CP C	B9
CP D	BA
CP E	BB
CP H	BC
CP L	BD
CP N	FE XX
CPD	ED A9
CPDR	ED B9
CPI	ED A1
CPIR	ED B1
CPL	2F
DAA	27
DEC (HL)	35
DEC (IX+d)	DD 35 XX
DEC (IY+d)	FD 35 XX
DEC A	3D
DEC B	05
DEC BC	0B
DEC C	0D
DEC D	15

DEC DE	1B
DEC E	1D
DEC H	25
DEC HL	2B
DEC IX	DD 2B
DEC IY	FD 2B
DEC L	2D
DEC SP	3B
DI	F3
DJNZ,d	10 XX
EI	FB
EX (SP),HL	E3
EX (SP),IX	DD E3
EX (SP),IY	FD E3
EX AF,AF'	08
EX DE,HL	EB
EXX	D9
HALT	76
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E
IN A,(C)	ED 78
IN A,port	DB XX
IN B,(C)	ED 40
IN C,(C)	ED 48
IN D,(C)	ED 50
IN E,(C)	ED 58
IN H,(C)	ED 60
IN L,(C)	ED 68
INC (HL)	34
INC (IX+d)	DD 34
INC (IY+d)	FD 34
INC A	3C
INC B	04
INC BC	03
INC C	0C
INC D	14
INC DE	13
INC E	1C
INC H	24
INC HL	23
INC IX	DD 23
INC IY	FD 23
INC L	2C
INC SP	33
IND	ED AA
INDR	ED BA
INI	ED A2
INIR	ED B2
JP (HL)	E9
JP (IX)	DD E9
JP (IY)	FD E9
JP NN	C3 XX XX
JP C,NN	DA XX XX
JP M,NN	FA XX XX
JP NC,NN	D2 XX XX
JP NZ,NN	C2 XX XX
JP P,NN	F2 XX XX
JP PE,NN	EA XX XX
JP PO,NN	E2 XX XX
JP Z,NN	CA XX XX
JR C,d	38 XX
JR d	18 XX
JR NC,d	30 XX
JR NZ,d	20 XX
JR Z,d	28 XX
LD (NN),A	32 XX XX
LD (NN),BC	ED 43 XX XX
LD (NN),DE	ED 53 XX XX
LD (NN),HL	22 XX XX
LD (NN),IX	DD 22 XX XX
LD (NN),IY	FD 22 XX XX
LD (NN),SP	ED 73 XX XX
LD (BC),A	02
LD (DE),A	12
LD (HL),A	77
LD (HL),B	70
LD (HL),C	71
LD (HL),D	72
LD (HL),N	36 XX
LD (HL),E	73
LD (HL),H	74
LD (HL),L	75
LD (IX+d),A	DD 77 XX

LD (IX+d),B	DD 70 XX
LD (IX+d),C	DD 71 XX
LD (IX+d),D	DD 72 XX
LD (IX+d),N	DD 36 XX XX
LD (IX+d),E	DD 73 XX
LD (IX+d),H	DD 74 XX
LD (IX+d),L	DD 75 XX
LD (IY+d),A	FD 77 XX
LD (IY+d),B	FD 70 XX
LD (IY+d),C	FD 71 XX
LD (IY+d),D	FD 72 XX
LD (IY+d),N	FD 36 XX XX
LD (IY+d),E	FD 73 XX
LD (IY+d),H	FD 74 XX
LD (IY+d),L	FD 75 XX
LD A,(NN)	3A XX XX
LD A,(BC)	0A
LD A,(DE)	1A
LD A,(HL)	7E
LD A,(IX+d)	DD 7E XX
LD A,(IY+d)	FD 7E XX
LD A,A	7F
LD A,B	78
LD A,C	79
LD A,D	7A
LD A,N	3E XX
LD A,E	7B
LD A,H	7C
LD A,I	ED 57
LD A,L	7D
LD A,R	ED 5F
LD B,(HL)	46
LD B,(IX+d)	DD 46 XX
LD B,(IY+d)	FD 46 XX
LD B,A	47
LD B,B	40
LD B,C	41
LD B,D	42
LD B,N	06 XX
LD B,E	43
LD B,H	44
LD B,L	45
LD BC,(NN)	ED 4B XX XX
LD BC,NN	01 XX XX
LD C,(HL)	4E
LD C,(IX+d)	DD 4E XX
LD C,(IY+d)	FD 4E XX
LD C,A	4F
LD C,B	48
LD C,C	49
LD C,D	4A
LD C,N	0E XX
LD C,E	4B
LD C,H	4C
LD C,L	4D
LD D,(HL)	56
LD D,(IX+d)	DD 56 XX
LD D,(IY+d)	FD 56 XX
LD D,A	57
LD D,B	50
LD D,C	51
LD D,D	52
LD D,N	16 XX
LD D,E	53
LD D,H	54
LD D,L	55
LD DE,(NN)	ED 5B XX XX
LD DE,NN	11 XX XX
LD E,(HL)	5E
LD E,(IX+d)	DD 5E XX
LD E,(IY+d)	FD 5E XX
LD E,A	5F
LD E,B	58
LD E,C	59
LD E,D	5A
LD E,N	1E XX
LD E,E	5B
LD E,H	5C
LD E,L	5D
LD H,(HL)	66
LD H,(IX+d)	DD 66 XX
LD H,(IY+d)	FD 66 XX

LD H,A	67
LD H,B	60
LD H,C	61
LD H,D	62
LD H,N	26 XX
LD H,E	63
LD H,H	64
LD H,L	65
LD HL,(NN)	ED 6B XX XX
LD HL,(NN)	2A XX XX
LD HL,NN	21 XX XX
LD I,A	ED 47
LD IX,(NN)	DD 2A XX XX
LD IX,NN	DD 21 XX XX
LD IY,(NN)	FD 2A XX XX
LD IY,NN	FD 21 XX XX
LD L,A	6F
LD L,B	68
LD L,C	69
LD L,D	6A
LD L,N	2E XX
LD L,E	6B
LD L,H	6C
LD L,L	6D
LD R,A	ED 4F
LD SP,(NN)	ED 7B XX XX
LD SP,NN	31 XX XX
LD SP,HL	F9
LD SP,IX	DD F9
LD SP,IY	FD F9
LDD	ED A8
LDDR	ED B8
LDI	ED A0
LDIR	ED B0
NEG	ED 44
NOP	00
OR (HL)	B6
OR (IX+d)	DD B6 XX
OR (IY+d)	FD B6 XX
OR A	B7
OR B	B0
OR C	B1
OR D	B2
OR N	F6 XX
OR E	B3
OR H	B4
OR L	B5
OTDR	ED BB
OTIR	ED B3
OUT (C),A	ED 79
OUT (C),B	ED 41
OUT (C),C	ED 49
OUT (C),D	ED 51
OUT (C),E	ED 59
OUT (C),H	ED 61
OUT (C),L	ED 69
OUT port,A	D3
OUTD	ED AB
OUTI	ED A3
POP AF	F1
POP BC	C1
POP DE	D1
POP HL	E1
POP IX	DD E1
POP IY	FD E1
PUSH AF	F5
PUSH BC	C5
PUSH DE	D5
PUSH HL	E5
PUSH IX	DD E5
PUSH IY	FD E5
RES 0,(HL)	CB 86
RES 0,(IX+d)	DD CB XX 86
RES 0,(IY+d)	FD CB XX 86
RES 0,A	CB 87
RES 0,B	CB 80
RES 0,C	CB 81
RES 0,D	CB 82
RES 0,E	CB 83
RES 0,H	CB 84
RES 0,L	CB 85
RES 1,(HL)	CB 8E

RES 1,(IX+d)	DD CB XX 8E
RES 1,(IY+d)	FD CB XX 8E
RES 1,A	CB 8F
RES 1,B	CB 88
RES 1,C	CB 89
RES 1,D	CB 8A
RES 1,E	CB 8B
RES 1,H	CB 8C
RES 1,L	CB 8D
RES 2,(HL)	CB 96
RES 2,(IX+d)	DD CB XX 96
RES 2,(IY+d)	FD CB XX 96
RES 2,A	CB 97
RES 2,B	CB 90
RES 2,C	CB 91
RES 2,D	CB 92
RES 2,E	CB 93
RES 2,H	CB 94
RES 2,L	CB 95
RES 3,(HL)	CB 9E
RES 3,(IX+d)	DD CB XX 9E
RES 3,(IY+d)	FD CB XX 9E
RES 3,A	CB 9F
RES 3,B	CB 98
RES 3,C	CB 99
RES 3,D	CB 9A
RES 3,E	CB 9B
RES 3,H	CB 9C
RES 3,L	CB 9D
RES 4,(HL)	CB A6
RES 4,(IX+d)	DD CB XX A6
RES 4,(IY+d)	FD CB XX A6
RES 4,A	CB A7
RES 4,B	CB A0
RES 4,C	CB A1
RES 4,D	CB A2
RES 4,E	CB A3
RES 4,H	CB A4
RES 4,L	CB A5
RES 5,(HL)	CB AE
RES 5,(IX+d)	DD CB XX AE
RES 5,(IY+d)	FD CB XX AE
RES 5,A	CB AF
RES 5,B	CB A8
RES 5,C	CB A9
RES 5,D	CB AA
RES 5,E	CB AB
RES 5,H	CB AC
RES 5,L	CB AD
RES 6,(HL)	CB B6
RES 6,(IX+d)	DD CB XX B6
RES 6,(IY+d)	FD CB XX B6
RES 6,A	CB B7
RES 6,B	CB B0
RES 6,C	CB B1
RES 6,D	CB B2
RES 6,E	CB B3
RES 6,H	CB B4
RES 6,L	CB B5
RES 7,(HL)	CB BE
RES 7,(IX+d)	DD CB XX BE
RES 7,(IY+d)	FD CB XX BE
RES 7,A	CB BF
RES 7,B	CB B8
RES 7,C	CB B9
RES 7,D	CB BA
RES 7,E	CB BB
RES 7,H	CB BC
RES 7,L	CB BD
RET	C9
RET C	D8
RET M	F8
RET NC	D0
RET NZ	C0
RET P	F0
RET PE	E8
RET PO	E0
RET Z	C8
RETI	ED 4D
RETN	ED 45
RL (HL)	CB 16
RL (IX+d)	DD CB XX 16

RL (IY+d)	FD CB XX 16
RL A	CB 17
RL B	CB 10
RL C	CB 11
RL D	CB 12
RL E	CB 13
RL H	CB 14
RL L	CB 15
RLA	17
RLC (HL)	CB 06
RLC (IX+d)	DD CB XX 06
RLC (IY+d)	FD CB XX 06
RLC A	CB 07
RLC B	CB 00
RLC C	CB 01
RLC D	CB 02
RLC E	CB 03
RLC H	CB 04
RLC L	CB 05
RLCA	07
RLD	ED 6F
RR (HL)	CB 1E
RR (IX+d)	DD CB XX 1E
RR (IY+d)	FD CB XX 1E
RR A	CB 1F
RR B	CB 18
RR C	CB 19
RR D	CB 1A
RR E	CB 1B
RR H	CB 1C
RR L	CB 1D
RRA	1F
RRC (HL)	CB 0E
RRC (IX+d)	DD CB XX 0E
RRC (IY+d)	FD CB XX 0E
RRC A	CB 0F
RRC B	CB 08
RRC C	CB 09
RRC D	CB 0A
RRC E	CB 0B
RRC H	CB 0C
RRC L	CB 0D
RRCA	0F
RRD	ED 67
RST 00	C7
RST 08	CF
RST 10	D7
RST 18	DF
RST 20	E7
RST 28	EF
RST 30	F7
RST 38	FF
SBC A,(HL)	9E
SBC A,(IX+d)	DD 9E XX
SBC A,(IY+d)	FD 9E XX
SBC A,A	9F
SBC A,B	98
SBC A,C	99
SBC A,D	9A
SBC A,N	DE XX
SBC A,E	9B
SBC A,H	9C
SBC A,L	9D
SBC HL,BC	ED 42
SBC HL,DE	ED 52
SBC HL,HL	ED 62
SBC HL,SP	ED 72
SCF	37
SET 0,(HL)	CB C6
SET 0,(IX+d)	DD CB XX C6
SET 0,(IY+d)	FD CB XX C6
SET 0,A	CB C7
SET 0,B	CB C0
SET 0,C	CB C1
SET 0,D	CB C2
SET 0,E	CB C3
SET 0,H	CB C4
SET 0,L	CB C5
SET 1,(HL)	CB CE
SET 1,(IX+d)	DD CB XX CE
SET 1,(IY+d)	FD CB XX CE
SET 1,A	CB CF

SET 1,B	CB C8
SET 1,C	CB C9
SET 1,D	CB CA
SET 1,E	CB CB
SET 1,H	CB CC
SET 1,L	CB CD
SET 2,(HL)	CB D6
SET 2,(IX+d)	DD CB XX D6
SET 2,(IY+d)	FD CB XX D6
SET 2,A	CB D7
SET 2,B	CB D0
SET 2,C	CB D1
SET 2,D	CB D2
SET 2,E	CB D3
SET 2,H	CB D4
SET 2,L	CB D5
SET 3,(HL)	CB DE
SET 3,(IX+d)	DD CB XX DE
SET 3,(IY+d)	FD CB XX DE
SET 3,A	CB DF
SET 3,B	CB D8
SET 3,C	CB D9
SET 3,D	CB DA
SET 3,E	CB DB
SET 3,H	CB DC
SET 3,L	CB DD
SET 4,(HL)	CB E6
SET 4,(IX+d)	DD CB XX E6
SET 4,(IY+d)	FD CB XX E6
SET 4,A	CB E7
SET 4,B	CB E0
SET 4,C	CB E1
SET 4,D	CB E2
SET 4,E	CB E3
SET 4,H	CB E4
SET 4,L	CB E5
SET 5,(HL)	CB EE
SET 5,(IX+d)	DD CB XX EE
SET 5,(IY+d)	FD CB XX EE
SET 5,A	CB EF
SET 5,B	CB E8

SET 5,C	CB E9
SET 5,D	CB EA
SET 5,E	CB EB
SET 5,H	CB EC
SET 5,L	CB ED
SET 6,(HL)	CB F6
SET 6,(IX+d)	DD CB XX F6
SET 6,(IY+d)	FD CB XX F6
SET 6,A	CB F7
SET 6,B	CB F0
SET 6,C	CB F1
SET 6,D	CB F2
SET 6,E	CB F3
SET 6,H	CB F4
SET 6,L	CB F5
SET 7,(HL)	CB FE
SET 7,(IX+d)	DD CB XX FE
SET 7,(IY+d)	FD CB XX FE
SET 7,A	CB FF
SET 7,B	CB F8
SET 7,C	CB FA
SET 7,D	CB FB
SET 7,E	CB FC
SET 7,H	CB FD
SET 7,L	CB FE
SLA (HL)	CB 26
SLA (IX+d)	DD CB XX 26
SLA (IY+d)	FD CB XX 26
SLA A	CB 27
SLA B	CB 20
SLA C	CB 21
SLA D	CB 22
SLA E	CB 23
SLA H	CB 24
SLA L	CB 25
SRA (HL)	CB 2E
SRA (IX+d)	DD CB XX 2E
SRA (IY+d)	FD CB XX 2E
SRA A	CB 2F
SRA B	CB 28
SRA C	CB 29

SRA D	CB 2A
SRA E	CB 2B
SRA H	CB 2C
SRA L	CB 2D
SRL (HL)	CB 3E
SRL (IX+d)	DD CB XX 3E
SRL (IY+d)	FD CB XX 3E
SRL A	CB 3F
SRL B	CB 38
SRL C	CB 39
SRL D	CB 3A
SRL E	CB 3B
SRL H	CB 3C
SRL L	CB 3D
SUB (HL)	96
SUB (IX+d)	DD 96 XX
SUB (IY+d)	FD 96 XX
SUB A	97
SUB B	90
SUB C	91
SUB D	92
SUB N	D6 XX
SUB E	93
SUB H	94
SUB L	95
XOR (HL)	AE
XOR (IX+d)	DD AE XX
XOR (IY+d)	FD AE XX
XOR A	AF
XOR B	A8
XOR C	A9
XOR D	AA
XOR N	EE XX
XOR E	AB
XOR H	AC
XOR L	AD

1 CONVERSÃO DEC — HEX

DEC	HEX
0	00
1	01
2	02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
10	0A
11	0B
12	0C
13	0D
14	0E
15	0F
16	10
17	11
18	12
19	13
20	14
21	15
22	16
23	17
24	18
25	19
26	1A
27	1B
28	1C
29	1D
30	1E
31	1F
32	20
33	21
34	22
35	23
36	24
37	25
38	26
39	27

DEC	HEX
40	28
41	29
42	2A
43	2B
44	2C
45	2D
46	2E
47	2F
48	30
49	31
50	32
51	33
52	34
53	35
54	36
55	37
56	38
57	39
58	3A
59	3B
60	3C
61	3D
62	3E
63	3F
64	40
65	41
66	42
67	43
68	44
69	45
70	46
71	47
72	48
73	49
74	4A
75	4B
76	4C
77	4D
78	4E
79	4F

DEC	HEX
80	50
81	51
82	52
83	53
84	54
85	55
86	56
87	57
88	58
89	59
90	5A
91	5B
92	5C
93	5D
94	5E
95	5F
96	60
97	61
98	62
99	63
100	64
101	65
102	66
103	67
104	68
105	69
106	6A
107	6B
108	6C
109	6D
110	6E
111	6F
112	70
113	71
114	72
115	73
116	74
117	75
118	76
119	77

DEC	HEX	DEC	HEX	DEC	HEX
120	78	165	A5	210	D2
121	79	166	A6	211	D3
122	7A	167	A7	212	D4
123	7B	168	A8	213	D5
124	7C	169	A9	214	D6
125	7D	170	AA	215	D7
126	7E	171	AB	216	D8
127	7F	172	AC	217	D9
128	80	173	AD	218	DA
129	81	174	AE	219	DB
130	82	175	AF	220	DC
131	83	176	B0	221	DD
132	84	177	B1	222	DE
133	85	178	B2	223	DF
134	86	179	B3	224	E0
135	87	180	B4	225	E1
136	88	181	B5	226	E2
137	89	182	B6	227	E3
138	8A	183	B7	228	E4
139	8B	184	B8	229	E5
140	8C	185	B9	230	E6
141	8D	186	BA	231	E7
142	8E	187	BB	232	E8
143	8F	188	BC	233	E9
144	90	189	BD	234	EA
145	91	190	BE	235	EB
146	92	191	BF	236	EC
147	93	192	C0	237	ED
148	94	193	C1	238	EE
149	95	194	C2	239	EF
150	96	195	C3	240	F0
151	97	196	C4	241	F1
152	98	197	C5	242	F2
153	99	198	C6	243	F3
154	9A	199	C7	244	F4
155	9B	200	C8	245	F5
156	9C	201	C9	246	F6
157	9D	202	CA	247	F7
158	9E	203	CB	248	F8
159	9F	204	CC	249	F9
160	A0	205	CD	250	FA
161	A1	206	CE	251	FB
162	A2	207	CF	252	FC
163	A3	208	D0	253	FD
164	A4	209	D1	254	FE
				255	FF

2 CONVERSÃO DEC — HEX

DEC	HEX	DEC	HEX	DEC	HEX
0	00	10240	28	20480	50
256	01	10496	29	20736	51
512	02	10752	2A	20992	52
768	03	11008	2B	21248	53
1024	04	11264	2C	21504	54
1280	05	11520	2D	21760	55
1536	06	11776	2E	22016	56
1792	07	12032	2F	22272	57
2048	08	12288	30	22528	58
2304	09	12544	31	22784	59
2560	0A	12800	32	23040	5A
2816	0B	13056	33	23296	5B
3072	0C	13312	34	23552	5C
3328	0D	13568	35	23808	5D
3584	0E	13824	36	24064	5E
3840	0F	14080	37	24320	5F
4096	10	14336	38	24576	60
4352	11	14592	39	24832	61
4608	12	14848	3A	25088	62
4864	13	15104	3B	25344	63
5120	14	15360	3C	25600	64
5376	15	15616	3D	25856	65
5632	16	15872	3E	26112	66
5888	17	16128	3F	26368	67
6144	18	16384	40	26624	68
6400	19	16640	41	26880	69
6656	1A	16896	42	27136	6A
6912	1B	17152	43	27392	6B
7168	1C	17408	44	27648	6C
7424	1D	17664	45	27904	6D
7680	1E	17920	46	28160	6E
7936	1F	18176	47	28416	6F
8192	20	18432	48	28672	70
8448	21	18688	49	28928	71
8704	22	18944	4A	29184	72
8960	23	19200	4B	29440	73
9216	24	19456	4C	29696	74
9472	25	19712	4D	29952	75
9728	26	19968	4E	30208	76
9984	27	20224	4F	30464	77

DEC	HEX	DEC	HEX	DEC	HEX
30720	78	42240	A5	53760	D2
30976	79	42496	A6	54016	D3
31232	7A	42752	A7	54272	D4
31488	7B	43008	A8	54528	D5
31744	7C	43264	A9	54784	D6
32000	7D	43520	AA	55040	D7
32256	7E	43776	AB	55296	D8
32512	7F	44032	AC	55552	D9
32768	80	44288	AD	55808	DA
33024	81	44544	AE	56064	DB
33280	82	44800	AF	56320	DC
33536	83	45056	B0	56576	DD
33792	84	45312	B1	56832	DE
34048	85	45568	B2	57088	DF
34304	86	45824	B3	57344	E0
34560	87	46080	B4	57600	E1
34816	88	46336	B5	57856	E2
35072	89	46592	B6	58112	E3
35328	8A	46848	B7	58368	E4
35584	8B	47104	B8	58624	E5
35840	8C	47360	B9	58880	E6
36096	8D	47616	BA	59136	E7
36352	8E	47872	BB	59392	E8
36608	8F	48128	BC	59648	E9
36864	90	48384	BD	59904	EA
37120	91	48640	BE	60160	EB
37376	92	48896	BF	60416	EC
37632	93	49152	C0	60672	ED
37888	94	49408	C1	60928	EE
38144	95	49664	C2	61184	EF
38400	96	49920	C3	61440	F0
38656	97	50176	C4	61696	F1
38912	98	50432	C5	61952	F2
39168	99	50688	C6	62208	F3
39424	9A	50944	C7	62464	F4
39680	9B	51200	C8	62720	F5
36936	9C	51456	C9	62976	F6
40192	9D	51712	CA	63232	F7
40448	9E	51968	CB	63488	F8
40704	9F	52224	CC	63744	F9
40960	A0	52480	CD	64000	FA
41216	A1	52736	CE	64256	FB
41472	A2	52992	CF	64512	FC
41728	A3	53248	D0	64768	FD
41984	A4	53504	D1	65024	FE
				65280	FF

CLUBE NACIONAL DOS TK/NE/SINCLAIR

O QUE É O CLUBE ?

O CLUBE NACIONAL DOS TK/NE/SINCLAIR é o primeiro (e o maior) clube brasileiro para os usuários dos microcomputadores compátiveis com o SINCLAIR ZX 81. Voce tem um CP 200, RINGO, NE Z8000, TK 82C, TK 83, TK 85, ZX 81 ou TIMEX 1000 ? Então este clube é para voce.

QUAIS OS BENEFÍCIOS DO CLUBE ?

Como associado do Clube voce receberá regularmente o jornal MICRO BITS, a única publicação periódica para estes micros que é completamente independente dos fabricantes. Assim as análises de micros, periféricos, acessórios e software são imparciais. Também voce pode comprar livros e programas em cassette com desconto - o primeiro Clube de micros que oferece esta vantagem. Voce pode recuperar o valor da assinatura com a compra de algumas fitas de nossas ofertas!

E O JORNAL MICRO BITS ?

É a maneira mais barata de conseguir programas para o seu micro. No MICRO BITS há artigos sobre programação em Basic e linguagem de máquina, análise de equipamentos e programas em fita cassette,, dicas, novidades cartas dos leitores, clubes locais,etc.,etc...

COMO TORNAR-SE SÓCIO DO CLUBE:

Envie o cheque nominal e cruzado no valor de Cr$ 8.400,00, junto com a ficha de assinatura no verso para:

MICRO BITS INFORMÁTICA LTDA.

CAIXA POSTAL 12.464

04798 - SÃO PAULO - SP

À

MICRON ELETRONICA COM IND LTDA

CAIXA POSTAL 100

12200 SAO JOSE DOS CAMPOS - SP

PEDIDO DE MATERIAL

Quant	Unid.	DISCRIMINAÇÃO	Unit.	TOTAL
		+ 10%		
		T O T A L		

INCLUSO CHEQUE Nº_____ VALOR Cr$_____

C/ BANCO_____

REMETENTE:_____

END._____ CID._____ EST._____

203

FICHA DE ASSINATURA PARA MICRO BITS E CLUBE NACIONAL DOS TK/NE/SINCLAIR

Valor remetido : $ 8.400,00 - Válido até 28/02/83 - Cheque nº_____ Banco_____

NOME_____ TEL._____ DDD_____

ENDEREÇO_____

CEP_____ CIDADE_____ ESTADO_____

PROFISSÃO_____ GRAU DE INSTRUÇÃO_____

SE POSSUI UM MICRO (OU TERÁ UM), QUAL ?_____ QUANTOS K DE MEMÓRIA ?_____

TEM IMPRESSORA ? (S/N)_____ QUAL ?_____ TEM JOYSTICK ? (S/N)_____

Voce se considera:_____

1 - INICIANTE 2 - PROGRAMA RAZOAVELMENTE EM BASIC 3 - PROGRAMA BEM EM BASIC.

4 - PROGRAMA BEM EM BASIC E RAZOAVELMENTE EM LINGUAGEM DE MÁQUINA 5 - UM "EXPERT"!

QUAIS AS PRINCIPAIS ÁREAS DE INTERESSE ?(em ordem de preferência)_____

1 - RECREATIVA (JOGOS) 2 - EDUCACIONAL 3 - CIENTIFICA 4 - COMERCIAL

TEM OUTROS INTERESSES (OU HOBBIES) QUE POSSAM SE BENEFICIAR COM O USO DO MICRO ?_____

_____

ASSINATURA EM / /

À

MICRON ELETRONICA COM IND LTDA

CAIXA POSTAL 100

12200 SAO JOSE DOS CAMPOS - SP

PEDIDO DE MATERIAL

Quant	Unid.	DISCRIMINAÇÃO	Unit.	TOTAL
		+ 10%		
		TOTAL		

INCLUSO CHEQUE Nº_____

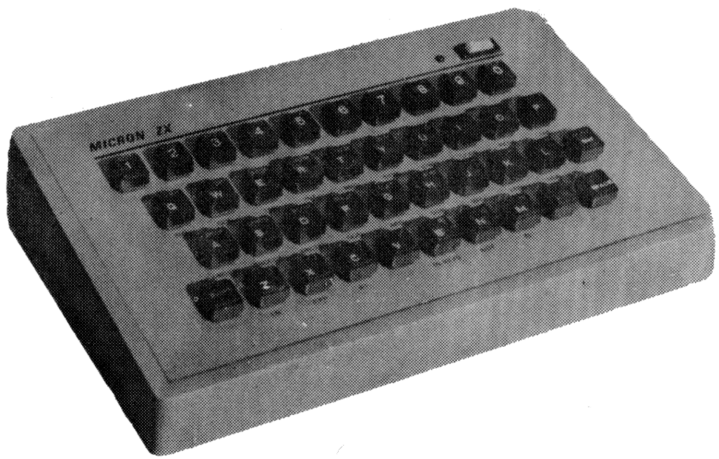
C/ BANCO_____ VALOR Cr$_____

REMETENTE:_____

END._____ CID._____ EST._____

TECLADO

PARA MICROCOMPUTADORES ZX 80/81 TK 82C NE Z8000



TECLADO MECÂNICO

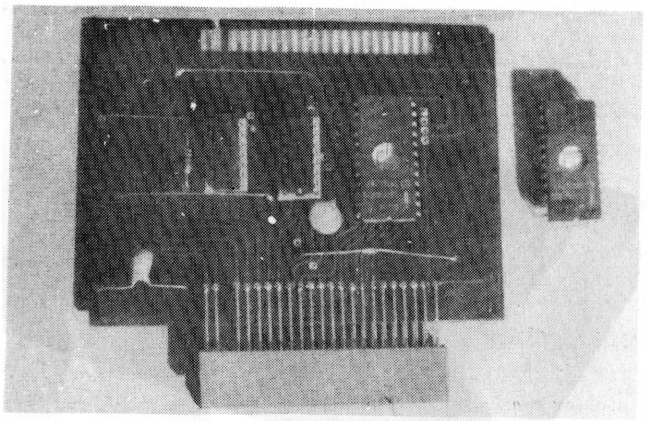
Kit para montagem de um teclado de acionamento mecânico. Contém 40 teclas gravadas em baixo relevo a duas cores, excluindo os símbolos gráficos, tres painéis pintados em esmalte poliuretano cinza metálico, sendo dois deles em fiberglass com letreiros em resina epoxi preta. Contém adesivos, interruptor e o gabinete em madeira, sem acabamento.

TECLADO MECÂNICO EM KIT

SOFTWARE EM EPROM

PARA MICROCOMPUTADORES ZX 80/81 TK 82C NE Z8000

ACRESCENTE AO SEU MICRO UMA EPROM DE 2 KBYTES
COM PROGRAMAS PRÉ-GRAVADOS



Circuito leitor de EPROM, ligado entre o micro e a expansão de memoria, lê programas de uma EPROM pré-gravada com o usc da função USR.
Montado em placa de fiberglass com 9 x 10 cm possui termina is em banho de ouro. Não requer alimentação externa ou alte rações no micro. Pode ser usado com Expansões e Printer.

CIRCUITO LEITOR DE EPROM

NÃO INCLUI E P R O M*

VANTAGENS:
PROGRAMA(S) PRONTO(S) PARA RODAR, SEM OCUPAR ESPAÇO NA RAM.
NÃO PRECISAM SER CARREGADOS DO CASSETTE. É SÓ LIGAR.
FUNCIONAM COM OUTROS PROGRAMAS CARREGADOS VIA TECLADO OU K7.

EPROM PRÉ-GRAVADA - FUNÇÕES I
o mesmo programa do K7
EPROM PRÉ-GRAVADA - RAM TOPER
o mesmo programa do K7

SOFTWARE PARA MICROCOMPUTADORES SINCLAIR

ZX SOFTWARE

OS QUATRO MELHORES PROGRAMAS EM CÓDIGO, JAMAIS PRODUZIDOS PARA UM SINCLAIR EM UM ÚNICO CASSETTE. POR APENAS Cr$ 15.000

Assembler

Escreva os seus programas em código, usando diretamente os mnemô nicos do Z80 com este Editor e Monitor de Assembly. Interpreta to das as instruções do Z80. Oferece excelentes facilidades de edi ção, manipulação do cursor, códigos de erro, entrada de texto, nú meros, labels, repetição de tecla, etc...

Disassembler

Lê códigos no Assembly do Z80. Fornece os endereços em decimal, os códigos em hexadecimal seguidos dos mnemônicos completos. In terpreta todas as instruções do Z80.

Compiler

Transforme instantaneamente os seus programas em Basic em progra mas em código, usando o Compiler. Aceita quase todos os comandos do Basic Sinclair e passa a rodar os programas até 50 vezes mais rápido.

Monitor

Programa para estudo de programas em código. Permite listar a Me mória, Registers e Flags. Pode-se ainda introduzir Breakpoints, converter Hex para Decimal, decimal para Hex etc...

Distribuído por : ***MICRON ELETRÔNICA***

LITERATURA PARA MICROCOMPUTADORES SINCLAIR

MICRON ELETRÔNICA

● 45 PROGRAMAS PRONTOS PARA RODAR EM TK 82C E NE Z8000

8ª Edição, reimpressa — Cr$ 5.500

Arquivos - Estoque - Plano Contábil - Folha de Pagamento - Agenda Telefônica - Caça ao Pato - Trilha - Jogo da Velha - Forca - Dado Tabelas - Tabuadas - Conversão de Coordenadas - Média - Progressão - Tabela Price - Fibonacci - Depreciação - Renumerador de linhas em Código - etc...

● APLICAÇÕES SÉRIAS PARA TK 82C E CP 200

3ª Edição, atualizada e com nova composição gráfica — Cr$ 7.500

Quem é Sinclair - Convertendo outros Basics - Contando os Bytes Economizando Memória - Fluxogramas - Top Down - Erros da ROM - Conhecendo a Impressora - Chaining Programas - Sub-rotinas em Cassette - Folha de Pagamento - Balancete - Correção Monetária do Imobilizado - Das Contribuições do IAPAS - Contas a Receber - Cadastro de Clientes - Conta Bancária - Correção de Provas - Processador de Textos - Estatística - Custos - Orçamento Doméstico - Ram Toper em Código - etc...

● 30 JOGOS PARA TK82C E CP 200

2ª Edição — Cr$ 6.000

Damas - Labirinto - Enterprise - Golfe - Velha - Visita ao Castelo Cassino - Roleta Russa - Corrida de Cavalos - Vinte e Um - Cubo Mágico - Senha - Banco Imobiliário - Forca - Dados - Invasores - etc.

PROGRAMAS NO CÓDIGO DA MÁQUINA

Inversão de Vídeo - Som por Software - Labirinto - Destrava Soft

● CÓDIGO DE MÁQUINA PARA TK E CP 200

1ª Edição — Cr$ 9.000

Números Binários e Hexadecimais - Arquitetura do Z80 - Editando em Código - Programa para Edição - As Instruções do Z80 em Exemplos Sub-rotinas da ROM - A ROM de 8K - Dicionário das Instruções - Hex X Mnemônicos - Hex X Decimal - Incluindo os Programas Scroll - Save Display no Ram Top - Contadores de Pontos ou Tempo - DataFile Renumber - Labirinto - som por Software - Micron Pac - Bombardeio etc...

● SINCLAIR 8K ROM — DISASSEMBLY COMPLETO COMENTADO

Em Lançamento — MARÇO/84