OUTROS LIVROS NA ÁREA

Guias do Usuário

Carvalho		Basic para o TK 90X
Hurley		Programas para jovens programadores TK-82, 83, 85 e CP 200
Hurley	—	TK 90X — Programas para jovens programadores

0-07-450227-1

CARVALHO ASSEMBLER Para o TK 90X

ASSEMBLER

Para o

TK 90X
TK 95X

JOSÉ EDUARDO M. DE CARVALHO

COMPATÍVEL COM A LINHA ZX SPECTRUM

McGraw-Hill

Assembler para o TK 90X

Assembler para o TK 90X

JOSÉ EDUARDO MALUF DE CARVALHO

McGraw-Hill
São Paulo
Rua Tabapuã, 1.105, Itaim-Bibi
CEP 04533
(011) 881-8604 e (011) 881-8528

Rio de Janeiro • Lisboa • Porto • Bogotá • Buenos Aires • Guatemala • Madrid • México • New York • Panamá • San Juan • Santiago

Auckland • Hamburg • Kuala Lumpur • London • Milan • Montreal • New Delhi • Paris • Singapore • Sydney • Tokyo • Toronto

Assembler para o TK 90X

Coordenadora de Revisão: Daisy Pereira Daniel
Supervisor de Produção: Edson Sant'Anna
Capa: Layout: Cyro Giordano

Dados para Catalogação-na-Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

C324a — Carvalho, José Eduardo Maluf de.
Assembler para o TK 90X / José Eduardo Maluf de Carvalho. – São Paulo: McGraw-Hill, 1986.

1. Assembler (Linguagem de programação para computadores) 2. TK 90X (Computador) – Programação I. Título.

86-0250 — CDD-001.6424
-001.642

Índices para catálogo sistemático:

1. Assembler: Linguagem de programação : Computadores : Processamento de dados 001.6424
2. TK 90X : Computadores : Programação : Processamento de dados 001.642

Agradeço a todos os que me incentivaram e encorajaram a escrever este livro, em particular a Jorge dos Santos, e mais particularmente ainda a minha adorável esposa Vivian e a meus queridos filhos Felipe e Marina, que se contentaram apenas em me olhar, enquanto escrevia este volume.

O Autor

José Eduardo Maluf de Carvalho, 32 anos, arquiteto, atuando em Planejamento Urbano, na Secretaria Municipal de Planejamento de São Paulo e em projetos de edificações e construção civil em seu escritório próprio, sentiu a necessidade de agilizar seu trabalho particular, nas tarefas mais repetitivas do arquiteto, desde uma consulta às leis de zoneamento, que regem o uso e aproveitamento do solo no Município de São Paulo, até as fases finais de detalhamento, quantificação e custo de um projeto de arquitetura.

Para isso, adquiriu, como um curioso, em março de 1982, um TK 82-C, e imediatamente conseguiu fazer deste pequeno micro uma poderosa ferramenta de trabalho.

Sentindo falta de uma melhor resolução gráfica, bem como de uma maior capacidade de armazenamento, partiu para um micro que foi um caso de "amor a primeira vista". Era o ZX Spectrum da Sinclair.

Desde setembro de 1983, o domínio sobre essa máquina foi num crescente cada vez maior, incluindo aí os periféricos ZX Microdrive, que nada mais é que um micro acionador de cartuchos (em vez de discos) e a Interface One (de compatibilização com o Microdrive, saída RC 232 C e rede local).

Trabalhando principalmente em linguagem Basic, com algumas rotinas em código de máquina, conseguiu elaborar um pacote de programas aplicativos para arquitetura, que vão desde a análise do zoneamento, otimizando a ocupação do terreno, definição do programa do projeto de arquitetura, elaboração da planta baixa, vegetação, detalhes construtivos, perspectivas, culminando com o orçamento e o cronograma da obra. Ou seja, não se sai do micro sem ter o projeto de arquitetura completo.

Hoje, após o lançamento do TK 90X e do TK 95, da Microdigital, compatíveis com o Spectrum, e até com algumas inovações e levando em consideração o ineditismo do seu trabalho, o autor fundou a empresa ARQUITRON INFORMÁTICA LTDA., de comércio e prestação de serviços em informática.

Mas, nem por isso abandona os deliciosos momentos de lazer que tem com este programa parceiro de *vídeo games.*

Atualmente gerencia o Departamento de Desenvolvimento de Software da Empresa Tropic Informática Ltda., onde desenvolve programas para as linhas Spectrum e MSX.

Sumário

Meus parabéns!

Se você realmente deseja aprender a linguagem de máquina deste microprocessador de 8 bits, muito poderoso, denominado Z80A, você tem muita coragem, e só lhe dando os parabéns!

Você provavelmente já aprendeu e já domina todo o potencial da linguagem Basic deste microcomputador (já deve inclusive ter lido meu livro de Basic e seus segredos), mas quer ir um pouco mais adiante e dominar agora diretamente a máquina, através de seus códigos de processamento.

Pois vá em frente! Não existe nenhuma linguagem de alto nível que explore todo o potencial de uma máquina. Somente através de sua própria linguagem é que exploramos todo o seu potencial.

Esqueça todos os princípios da estrutura de um programa em linguagem Basic. Lá, você trabalha com linhas de programa, comando a comando, que serão posteriormente executados, um a um, seqüencialmente.

Aqui não – a estrutura da linguagem de máquina é completamente diferente. Evidente que a seqüência lógica de execução permanece, mas aqui nós manipulamos bytes diretamente armazenados em endereços especificados da memória. Isto é uma relação biunívoca: a cada endereço corresponde um e somente um byte de 8 bits.

Você vai precisar de muita paciência, precisará ser "adivinho" em algumas ocasiões, e principalmente deverá tomar muito cuidado com os valores numéricos que vai manipular. E exercitar muito.

Eu, particularmente, não conheço ninguém que faça um programa inteiro em código de máquina.

Você já imaginou, por exemplo, o programa inglês VU 3D, da Psion Software, que é um mini CAD (*computer aided design* – projetos assistidos por computador) para elaboração de perspectivas, que possui uma parte de códigos, num total de 32183 endereços, ser feito todo manualmente, e de cabeça? É praticamente impossível; isso é coisa de loucos.

O mais normal é a elaboração de programas em partes, em linguagem de alto nível, residente, que são "compiladas" uma a uma, ou seja, através de um programa-ferramenta, chamado "compilador", são traduzidos automaticamente para a linguagem de máquina.

Você quer, por exemplo, fazer a tela rodar em diversas direções (scroll variado). Faça uma rotina de execução em linguagem de máquina para cada direção. Cada uma delas será posteriormente acessada através das senhas ou condições que você impuser, e poderá ser utilizada em qualquer outro programa.

Mas, vamos lá, então, à programação em código de máquina.

Você já sabe que a única coisa que seu computador entende são sinais, respectivamente, "com voltagem" e "sem voltagem", ou ainda "voltagem alta" e "voltagem baixa". No caso do Z80, a voltagem alta equivale a + 5 volts, e a voltagem baixa equivale a 0 volts, que convencionou-se padronizar através dos dígitos 0, para voltagem baixa, e 1, para voltagem alta, dando origem ao sistema binário, diretamente manipulado pelo microcomputador.

Esses algarismos são mais conhecidos por bits, e o agrupamento de 8 desses bits dá origem a um byte, que é a menor unidade de armazenamento de memória de um computador, cujos valores vão de 0 a 255, no sistema decimal, totalizando 256 valores, ou, no sistema hexadecimal, de valores que iniciam em 00 e terminam em FF.

Na linguagem Basic, chegamos a manipular bytes de memória, mas, veja a grande dificuldade da linguagem de máquina – aqui, manipularemos até apenas um bit de um determinado byte, para ordenar algo ao microcomputador. E, por esse caminho, você percebe que, se errar ou esquecer apenas 1 bit de 1 byte, põe a perder todo o seu trabalho de elaboração de uma rotina em linguagem de máquina, provocando um "crash" no sistema, ou seja, o não retorno do micro para a linguagem Basic. Nestes casos, não se preocupe – basta desligar e ligar novamente o computador.

Não tenha medo de digitar nada, que você nunca, através do teclado, vai provocar algum dano no computador. O máximo que pode acontecer é a situação descrita acima.

Mas, como então programar em linguagem de máquina?

Vamos lá – primeiramente, você deve ter em mente, muito bem definida, a concepção geral do seu programa, sobre como ele vai funcionar, como vai responder a determinadas condições etc.

Conheço casos de programadores fanáticos em código de máquina, que chegam a pintar, num papel apropriado, a idéia que têm sobre o desenho desejam ver aparecer na tela, incluindo pequenos caracteres gráficos que devam se movimentar, para em seguida, através de um papel transparente onde constam todas as locações da tela, passarem a escrever a rotina que irá executar aquele desenho ou figura. O resultado é, na verdade, um desenho magnífico, mas, na minha modesta opinião, o processo é muito sofisticado.

Bem, de posse da concepção do programa, evidentemente adequado à lógica do computador, você deve em seguida elaborar um fluxograma dele, ou seja, as suas etapas de execução.

Definidas estas etapas, com uma listagem das mais de 700 instruções em código de máquina, você passa então a escrever a sua rotina, através das "mnemônicas", ou símbolos (nomes) das instruções, sempre levando em consideração os endereços, ou locações iniciais ou intermediárias da memória, que também podem ser rotulados (por você é claro) para não se perder em números mais tarde.

Neste fase, você tem duas opções para continuar o trabalho, que são:

1 – usar um programa-ferramenta denominado ASSEMBLER, ou montador, que permite que você digite a sua listagem ASSEMBLY, ou montada, a partir das mnemônicas, endereço por endereço, byte a byte, para que ele converta automaticamente estes seus símbolos em códigos, números que serão entendidos pelo computador para serem processados posteriormente.

Este é o método mais simples, visto que a primeira etapa de elaboração de um programa, é a sua listagem por instruções, mas, na minha opinião, não é o método mais indicado, pois está sujeito a muitos erros. E, como já vimos, basta uma unidade de um byte estar com o valor errado...

2 – esta opção é praticamente continuação da anterior. De posse da listagem das mnemônicas do programa, você deve converter estes símbolos para os seus respectivos códigos no sistema hexadecimal. Apesar do trabalho, isto já é uma vantagem, pois é uma conferência do trabalho.

Convertida a listagem para códigos hexadecimais, a última etapa desta opção é a conversão destes códigos para números do sistema decimal, que serão diretamente armazenados nos endereços da memória RAM do seu mi-

cro que você determinar. Nesta etapa, nada lhe impede que use um "Editor Assembler", outro programa-ferramenta que permite que você digite os códigos em hexadecimal, para que ele os converta automaticamente no sistema decimal.

Como você deve ter notado, o trabalho de programação em código de máquina é muito exaustivo e muito sujeito a erros, mas também muito compensador.

A prática desta linguagem é um fator muito importante para seu perfeito domínio, além do pleno conhecimento de todo o conjunto das instruções do microprocessador Z80A.

Praticando, e muito, você vai guardar os códigos, tanto em hexadecimal quanto em decimal, das instruções mais utilizadas na programação desta linguagem.

Bem, você já elaborou então a sua rotina em linguagem de máquina. Para um leigo no assunto, todos aqueles números não quer dizer nada. Mas, para quem entende, significa o resultado de horas de pesquisa e trabalho árduo!

Cuidado – não rode esse programa ainda! Você pode pôr tudo a perder se ele estiver errado! Como prevenção, grave essa rotina na fita cassete ou, se você tiver, no microdrive (que aparelho fantástico!).

Agora sim, de posse de uma cópia dessa rotina gravada, você pode processá-la, e saber, caso não aconteça o que você esperava, onde está o erro conferindo naquele papel rascunho onde você escreveu a listagem.

Carregue aquela rotina que foi gravada e, através de "POKE endereço, byte", corrija-a para gravá-la novamente, agora na segunda versão. Teste-a, e continue com esse processo até atingir o seu objetivo. Assim que atingi-lo, dê-se parabéns! Agora você também programa em linguagem de máquina.

Este é o objetivo deste livro. Ensiná-lo a programar em linguagem de máquina. Alguns autores falam superficialmente e profundamente (para mim, quanto mais se pratica, mais se aprofunda no assunto), de uma maneira muito didática, considerando que você não entende nada desse assunto, e que obrigatoriamente deve partir dos princípios mais rudimentares do processo (foi assim que eu comecei – sem curso algum).

Por isso começo o livro com uma recordação da linguagem Basic, cujos conceitos também serão utilizados em linguagem de máquina.

Pena que a teoria desta linguagem seja muito extensa; não tenha pressa, assimile bem os conceitos antes de passar adiante.

Nos capítulos finais do livro coloquei diversas rotinas em código de máquina, que, além de úteis e divertidas, vão mostrar-lhe como o computador trabalha, além de fazê-lo entender essa linguagem. Também inseri nesses capítulos alguns programas híbridos, ou seja, que são escritos em Basic, mas que possuem algum processamento diretamente em código de máquina.

Observe a execução dessas rotinas e desses programas, onde a velocidade de processamento do micro aumentou, chegando a ser, em certos casos, cerca de 100 vezes mais rápida que a linguagem Basic.

Essa é uma das grandes vantagens da linguagem de máquina – a velocidade de processamento muito alta, já que não existe nenhum interpretador no meio do caminho.

A outra grande vantagem é a economia de memória, por motivos óbvios (números são armazenados mais facilmente que a linguagem Basic; veja seu manual, não existem indicadores de início e final de linha, indicadores de final de comando etc.).

Pois bem, chega de escrever, e mãos à obra!

Divirta-se!

O AUTOR

Nota: Todos os programas, bem como as rotinas em linguagem de máquina listados neste livro foram testadas num TK 90X de 48K de RAM, nº 7544/CQF/264/11015/9, e rodaram sem problemas. No entanto, mais uma vez ficou constatado que este micro não é 100% compatível com seu irmão inglês, o ZX Spectrum da Sinclair.

Portanto, muito cuidado com os programas que você adquirir, pois alguns deles não rodam no TK 90X.

Não se preocupe com o comando "BEEP" que você encontrar nas listagens Basic. É um comando do ZX Spectrum, equivalente ao "SOUND" do TK 90X. (É que eu prefiro o Spectrum!)

Na última Feira da Informática, vi e testei o novo TK 95, compatível com o TK 90.

Pelo contato que tive durante poucos minutos com esta máquina, devo parabenizar seu fabricante, pois parece (parece...) que é uma grande máquina, com um ótimo teclado.

Capítulo 1

O que é o TK 90X

Podemos descrever qualquer sistema de computação sob três aspectos diferentes:

O primeiro aspecto é relativo à descrição sucinta do sistema de um modo geral, a partir de sua CPU e seus periféricos.

O segundo aspecto refere-se ao hardware (partes físicas) do sistema propriamente dito.

E o terceiro aspecto é sobre o desempenho lógico do micro.

O SISTEMA

O microcomputador TK 90X é uma pequena caixa preta, de plástico, medindo 236 mm de largura, por 146 mm de profundidade e 44 mm de altura. Na superfície horizontal superior estão dispostas as quarenta teclas de um composto de silicone, com as legendas impressas pelo processo chamado *hot-stamping*, ou seja, *silk screen* à quente, para maior durabilidade, que compõe o seu teclado.

Na traseira da caixa estão dispostos, da esquerda para a direita, o plug que conecta com a entrada da TV (plug da antena externa), o soquete de entrada que conecta com a saída do gravador cassete, o soquete de saída que conecta com a entrada do gravador cassete, o plug de conexão para joysticks ti-

po Atari, o conector de expansão da máquina, onde se conectam diversos periféricos de entrada/saída, tais como uma impressora (ZX Printer ou Alphacom 32) uma interface RS 232, a interface de compatibilização com o microdrive etc. e finalmente o plug de voltagem.

A placa de circuitos, incluindo o microprocessador Z80 e outros componentes eletrônicos, encontra-se dentro dessa caixa preta, tomando toda a sua dimensão, e é separada do teclado por dois cabos de ligação.

O HARDWARE (OS COMPONENTES FÍSICOS)

A placa de circuitos do TK 90X pode facilmente ser acessada, retirando-se os quatro parafusos inferiores que prendem a tampa, onde está instalado o teclado, à parte inferior da caixa onde se encontra a placa de circuitos. Se o seu micro estiver na garantia não convém abri-lo, pois irá destruir o lacre da garantia, anulando-a. Se não, e se a curiosidade for muito grande, retire os parafusos, segurando as duas partes da caixa, para que não se separem, e com o micro em posição de funcionamento levante o teclado vagarosamente, procurando não desconectar os seus dois cabos de ligação com a placa, pois são muito frágeis e suas pontas se desgastam muito facilmente. Estes cabos podem ser desconectados puxando-os suavemente dos seus soquetes, mas procure não fazê-lo: use duas hastes (canetas) de igual comprimento para sustentar a tampa, tal qual o capô de um carro.

Os componentes principais da placa de circuito estão desenhados na Figura 1.

Cada um desses componentes principais será descrito a seguir:

– O MICROPROCESSADOR Z80

Este chip, muito conhecido entre os aficcionados de informática, visto que é o coração de muitos sistemas que existem por aí, é o mais importante do TK 90X, e o mais sofisticado (também o mais complicado, em termos de código de máquina) dos microprocessadores de 8 bits. É, também, o coração dos micros linha MSX (Expert e Hotbit).

Como microprocessador, tem capacidade de trabalhar como um "computador", de uma maneira genérica, capaz de executar instruções armazenadas (programa). Os programas para o TK 90X, quaisquer que sejam eles, serão sempre convertidos internamente em instruções em código de máquina do Z80.

No TK 90X, o *clock* (relógio – medidor de tempo interno de execução de instruções) é de 3,75 MHZ, e, a esta velocidade, ele é capaz de processar cerca de 900.000 das mais simples instruções em código de máquina. É interessante notar que a partir do momento que se liga a fonte de energia do mi-

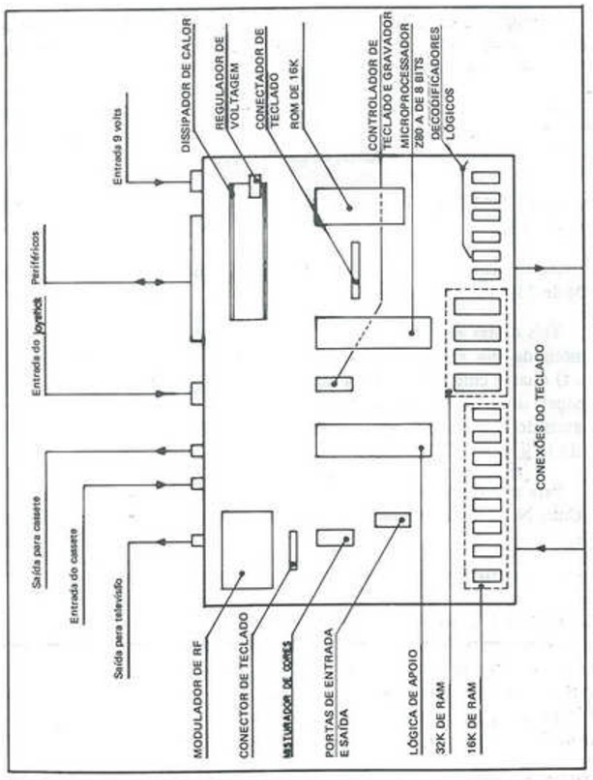


Figura 1 – *Placa de circuitos eletrônicos e componentes principais do TK 90X.*

cro, o seu microprocessador imediatamente começa a funcionar, apesar de não mostrar nenhum resultado. Ele simplesmente fica aguardando uma ordem. E são exatamente estas ordens que estudaremos neste livro.

– A ROM DE 16 KBYTES (ROM = READ ONLY MEMORY – OU MEMÓRIA APENAS DE LEITURA)

O programa em código de máquina que é normalmente executado pelo microprocessador Z80 está contido num único chip de ROM, que manipula 128 kbits ou 16 kbytes de informação.

Nestes 16 k de "programa monitor" do TK 90X, os aproximadamente primeiros 7 k estão dedicados ao sistema operacional, os 8 k seguintes, ao interpretador Basic e o 1 k restante, ao gerador de caracteres.

– A RAM (RANDOM ACCESS MEMORY – MEMÓRIA DE ACESSO RANDÔMICO)

Na versão standard de 16 k do TK 90X existem 8 chips de memória RAM de 2 k cada ou 16 kbits.

Três destes 8 chips formam o "arquivo da imagem" ou mapeamento da memória da tela, e podem normalmente ser usados somente com este propósito. O quarto chip é dedicado à manipulação dos bytes dos atributos (cores de papel, tinta, flash, bright etc.) das 768 posições de caracteres da tela, e as variáveis do sistema. Um pouco mais de 8 k está disponível ao usuário na versão de 16 k.

Para a RAM total de 48 k, são adicionados mais 4 chips de 8 k cada, ou 64 kbits. Nessa memória estão disponíveis para o usuário um pouco mais de 40 k.

– A LA (LÓGICA DE APOIO)

Este chip pode ser considerado como sendo um "grande chip feito de diversos pequenos chips". É um dos chips conhecidos como "customizado", ou seja, só serve para esse computador. Ele atua como um "centro de comunicação", verificando se tudo o que o microprocessador requer ou ordena está sendo efetuado; ele também "lê" a memória para ver em que consiste a imagem da TV, e envia os sinais apropriados para a interface da TV.

– MISTURADOR DE CORES

Este chip recebe as informações sobre cores da LA e converte-as para os sinais apropriados a serem enviados ao modulador VHF. O sinal produzido pelo modulador é ajustado para o canal 3.

– Em adição a estes componentes principais, existem ainda a interface que produz som no alto-falante da TV; a interface para o gravador cassete; o dissipador de calor; o regulador de voltagem, que converte a tensão de entrada em 5 volts absolutos, sem oscilação; alguns decodificadores de endereços e outros componentes menores.

A LÓGICA DO SISTEMA

Neste aspecto, todas as ligações entre os vários componentes internos do microcomputador são consideradas. Essas ligações possuem uma existência real – são "caminhos" impressos na placa de circuitos, ou eventualmente fios de ligação.

O microprocessador Z80 pode gerar um endereçamento individual de até 65536 endereços diferentes de memória (65536/1024 = 64 kb) de cada vez. No TK 90X, versão 16 k, apenas os endereços 0 a 32767 (32 kb, sendo 16 k de ROM e 16 k de RAM) estão disponíveis. Na versão de 48 K todos os endereços são utilizados.

No TK 90X, os endereços são produzidos na forma de 16 sinais binários. O endereço 0 é, portanto, 00000000 00000000, e o endereço 65535 é 11111111 11111111. Isto porque o computador trabalha com o sistema numérico binário, com os algarismos 0 e 1.

Vamos ver:

No nosso sistema decimal, temos que:

1985				
5×10^0	= 5 x	1 =	5	
8×10^1	= 8 x	10 =	80	+
9×10^2	= 9 x	100 =	900	
1×10^3	= 1 x	1000 =	1000	
			1985	

ou

4096				
6×10^0	= 6 x	1 =	6	
9×10^1	= 9 x	10 =	90	+
0×10^2	= 0 x	100 =	0	
4×10^3	= 4 x	1000 =	4000	
			4096	

Analogamente, no sistema binário, que utiliza somente os algarismos 0 e 1, ou bits, temos para 1 byte (conjunto de 8 bits)

```
1111 1111
|||||||↓ 1x2^0 = 1 x   1 =   1
||||||↓  1x2^1 = 1 x   2 =   2
|||||↓   1x2^2 = 1 x   4 =   4
||||↓    1x2^3 = 1 x   8 =   8
|||↓     1x2^4 = 1 x  16 =  16 +
||↓      1x2^5 = 1 x  32 =  32
|↓       1x2^6 = 1 x  64 =  64
↓        1x2^7 = 1 x 128 = 128
                           ---
                           255
```

Ou seja, 256 valores diferentes para um byte, que começam em 0 e terminam em 255. Portanto, $2^8 = 256$

$256 * 256 = 2^8 * 2^8 = 2^{16} = 65536$ ou 11111111 11111111

Da mesma forma, no sistema hexadecimal, que utiliza os algarismos 0 a 9 para valores decimais de 0 a 9, e letras de A até F para valores de 10 a 15, temos:

HEXADECIMAL	DECIMAL
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Portanto, para o valor decimal 255, temos em hexa FF e em binário 11111111

```
FF
|↓ 15 x 16^0 =  15 +
↓  15 x 16^1 = 240
             -----
             255
```

O programa a seguir faz as conversões automaticamente entre as bases numéricas decimal, binária e hexadecimal.

```
  1 REM ************************
  2 REM *Programa de conversao*
  3 REM *  de bases numericas *
  4 REM ************************
 10 CLS : PRINT "Este programa
faz a conversao dequalquer base
<=16 para qualqueroutra base <=1
6"
 15 INPUT "Digite a base anteri
or ";x$
 20 LET e=0
 25 IF VAL x$>16 OR x$="" THEN
GO TO 10
 30 GO SUB 185
 35 LET b=n
 40 IF n<2 OR n>16 THEN GO TO 1
0
 45 INPUT "Qual e o numero ";x$
 50 IF x$="" THEN GO TO 45
 55 GO SUB 205
 60 IF e=1 THEN PRINT "Erro ":
LET e=0: GO TO 45
 65 LET n1=n
 70 PRINT x$;" na base 10 e ";n
1
 75 IF n1<1000000 THEN GO TO 85
 80 PRINT "O numero na base 10
e >= 1000000podendo por isto oco
rrerem erros"
 85 INPUT "Qual e a nova base "
;x$
 90 IF x$="" THEN GO TO 85
 95 GO SUB 185
100 LET b1=n: IF n<2 OR n>16 TH
EN GO TO 85
105 LET b$=""
110 LET v=INT (n1/b1)
115 LET r=n1-v*b1
120 IF r>9 THEN GO TO 140
125 LET b$=b$+CHR$ (r+48)
130 LET n1=v: IF v=0 THEN GO TO
145
```

```
 135 GO TO 110
 140 LET r=r+55: LET b$=b$+CHR$
(r): LET n1=v: IF v<>0 THEN GO T
O 110
 145 PRINT "O numero na base ";b
1;" e ";
 150 FOR j=LEN b$ TO 1 STEP -1
 155 PRINT b$(j);: NEXT j
 160 PRINT
 165 INPUT "Mais numeros (s/n) ";
X$
 170 IF x$="s" THEN GO TO 10
 175 IF x$="n" THEN NEW
 180 GO TO 165
 185 LET n=0
 190 FOR j=1 TO LEN (x$): LET d=
CODE x$(j)
 195 LET n=n*10+d-48: NEXT j
 200 RETURN
 205 LET n=0
 210 FOR j=1 TO LEN x$: LET d=CO
DE x$(j)
 215 IF d>47 AND d<58 THEN LET d
=d-48: GO TO 230
 220 IF d>64 AND d<71 THEN LET d
=d-55: GO TO 230
 225 LET e=1: RETURN
 230 IF d>=b THEN LET e=1: RETUR
N
 235 LET n=n*b+d
 240 NEXT j
 245 RETURN
```

Os endereços são gerados pelo microprocessador Z80 e transportados pelo computador através das Vias de Endereçamento (Address Bus). São no total 16 vias, nas quais um endereço pode ser especificado considerando-se uma via possuindo uma "alta" voltagem (5 volts) ou uma "baixa" voltagem (0 volts), ou ainda, respectivamente, 1 para alta voltagem e 0 para baixa voltagem.

Enquanto as vias de endereço possuem 16 contatos, as vias de dados (Data Bus) do TK 90X possuem somente 8 contatos, pelo fato do seu microprocesador ser de 8 bits. Portanto, qualquer dado que estiver sendo manipulado pelo sistema, seja ele uma instrução em código de máquina, um valor qualquer, ou uma instrução em Basic, deve ser considerado como sendo um número decimal na faixa de 0 a 255 (valor máximo de 1 byte), ou na faixa binária de 00000000 até 11111111.

A figura a seguir mostra esquematicamente como as vias de endereçamento e as vias de dados são interligadas aos outros componentes do TK 90X.

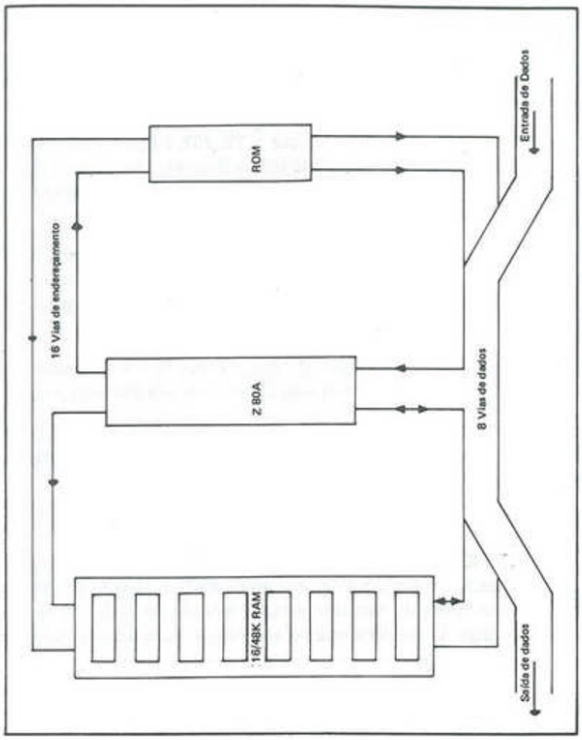


Figura 2 – *Vias de endereçamento e de dados do TK 90X*

O TK 90X é fornecido pela Microdigital com um programa monitor de 16 K, iniciando no endereço 0 e terminando no endereço 16383, que proporciona ao usuário o emprego de um sistema operacional bem como de um interpretador Basic. Também é possível "abandonar" este programa monitor e permitir que o microprocessador Z80 execute um programa em código de máquina escrito pelo usuário. Em modo normal de operação, o sistema operacional do TK 90X não requer nenhuma ação por parte do usuário, e todas as ações desse programa monitor são "transparentes" para ele.

Entretanto, isso acontece sempre que o TK 90X é ligado, mas não em operação; o seu interpretador Basic está constantemente "lendo" o teclado, através de uma rotina do programa monitor, aguardando uma declaração em Basic do usuário, seja em modo imediato ou em modo programado.

Note que de nenhuma maneira o microprocessador Z80 executará um programa em Basic – mas, sim o programa monitor, que é uma rotina em código de máquina do Z80. A única exceção a este procedimento é quando o próprio usuário insere no micro uma rotina em código de máquina.

O mapeamento da memória do TK 90X, versões 16 e 48 K é mostrado esquematicamente na Figura 3 e cada uma dessas áreas será discutida a seguir.

Nota: A partir daqui, todos os endereços que contiverem apenas algarismos serão da base decimal; se contiverem após os algarismos a letra "h" serão da base hexadecimal, e se contiverem a letra "b" serão da base binária.

– ÁREA DE ROM

Os 16 K de ROM contêm o sistema operacional, o interpretador Basic e o gerador de caracteres, ocupando os endereços de 0 a 16383, ou 0000 a 3FFFh. Como em qualquer microcomputador baseado no Z80, o início do programa em código de máquina está no endereço 0. Mais adiante, veremos esta área com mais detalhes.

– ÁREA DE MAPEAMENTO DA MEMÓRIA DA TELA – ARQUIVO DA IMAGEM

Os 6 K de memória, do endereço 16384 até 22527, ou 4000 a 57FFh, formam a área de alta resolução gráfica da tela. É importante notar que essa área está fixada nessa dimensão pelo hardware do TK 90X e não pode ser alterada por intermédio do software.

Existe uma relação de um para um entre todos os bits dessa área de memória e os pixéis (pixéis = *picture elements* – menor ponto de impressão) que aparecem na tela, e o cálculo seguinte mostra que o número de bits em 6 K de memória é igual ao número de pixéis que podem ser mostrados na tela.

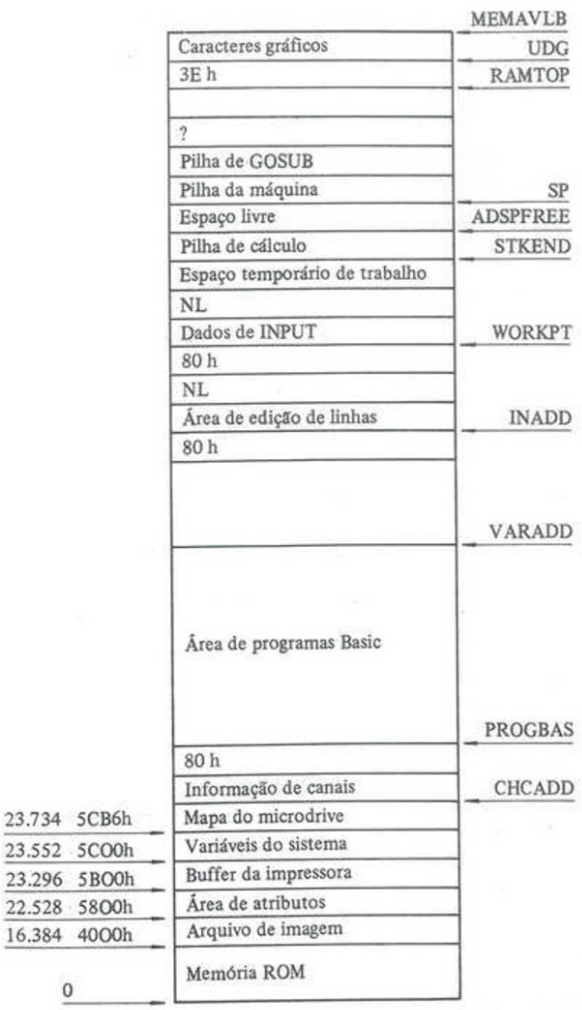


Figura 3 – *Mapeamento da memória do TK 90X*

Número de bytes em 6 K de memória = 1.024 x 6 = 6.144
Número de bits em 6 K de memória = 6.144 x 8 = 49.152
Número de pixéis em 32 colunas por 24 linhas da tela, com 64 pixéis por caractere (matriz de 8 x 8) = 32 x 24 x 64 = 49.152

Agora vamos ver isso em termos práticos, ou seja, como são dispostos todos esses elementos na tela.

Inicialmente, vamos considerar a tela da TV dividida em três blocos horizontais iguais. O bloco mais alto, começando na linha 0 e terminando na linha 7 é produzido através dos conteúdos dos endereços 16.384 a 18.431, ou 4000 a 47FFh. O bloco central, das linhas 8 a 15, utiliza os conteúdos dos endereços 18.432 a 20.479, ou 4800 a 4FFFh; e o bloco inferior, ou seja, o restante da tela do TK 90X, das linhas 16 a 23, utiliza os endereços 20.480 a 22.527, ou 5000 a 57FFh.

Preste atenção nos números em hexadecimal, que sugerem uma fácil manipulação do arquivo de imagem. Vamos analisar o bloco superior (e, por analogia, os outros dois blocos).

Cada posição PRINT necessita de 8 x 8 bits (64) e que haja 32 dessas posições em cada linha da tela (32h = 20d). Por outro lado, cada bloco da tela tem 32 x 8 x 8 = 2.048 bytes, que, em hexadecimal, equivalem a 0800h. Vimos que os bytes correspondem à ordem: todos os primeiros bytes do primeiro bloco, os segundos bytes, os terceiros etc., o que nos dá um salto de 8! Isso significa, por exemplo, que o primeiro byte da primeira linha PRINT da tela dista da segunda de 32 x 8 = 256, ou 0100h.

Conclusão:

– Um byte de uma posição PRINT dista 0100h do seu byte análogo na posição PRINT imediatamente abaixo ou acima, num mesmo bloco.

– Um byte de um bloco da tela dista 0800h ou 1600h do byte com a mesma posição relativa noutro bloco, dependendo do bloco de referência e do bloco a que se refere.

Se você já fez algumas experiências em Basic com seu TK 90X e gostou, não imagina o quanto isto pode ser útil em programas de manipulação de imagens, tanto em Basic como (e principalmente) em Assembler.

– ÁREA DE ATRIBUTOS

O arquivo de imagens possui ainda 768 áreas de caracteres, que vão do endereço 22.528 a 23.295. ou 5800 a 5AFFh, onde cada uma delas contém

os códigos para cor de papel, cor da tinta, flash e bright, e são usados para armazenar valores que determinam os atributos dos 768 caracteres que podem ser impressos na tela.

Os valores (menores que 255) dos atributos podem ser considerados a partir da expressão:

ATTR (lin., col.) = INK + PAPER * 8 + BRIGHT * 64 + FLASH * 128

Essa função ATTR (lin., col.) é equivalente a PEEK (22.528 + 32 * lin. + col.)

Ou seja, nos bytes dos atributos os bits 0, 1 e 2 determinam a cor da tinta; os bits 3, 4 e 5 determinam a cor do papel; o bit 6, se a posição está em modo bright ou não; e o bit 7, se está em modo flash ou não.

Por exemplo:

INK azul	= 1 ---------- 1	
PAPER branco	= 7 (7 x 8 = 56)	56
BRIGHT 0	= 0 ---------- 0	
FLASH 1	= 1 (1 x 128)	128
		185

A seguir, uma tabela de todos os valores possíveis para os atributos de qualquer posição. (Tabela 1).

Podemos imaginar cada pixel da tela da televisão (colorida) como sendo um pequeno triângulo eqüilátero, cujos vértices são iluminados. Para cada vértice há um pixel vermelho, um azul e um verde, e o arquivo de atributos é usado para controlar a iluminação (acesa ou não) dessas três cores diferentes. O arquivo da imagem vai indicar se um determinado pixel deve ser plotado com uma cor de tinta em particular. Os três bits mais baixos (0, 1 e 2) do valor do atributo daquela área são usados para decidir quais vértices do nosso triângulo imaginário, se o verde, o azul ou o vermelho, devem ser acesos. Nossos olhos contêm apenas três tipos de sensores de cores (verde, vermelho e azul). Nosso cérebro recebe sinal dos três vértices coloridos e combina-os em um simples pixel de cor composta. Por exemplo, se os três últimos bits de atributo são 111, equivalente à cor 7, nós temos verde+azul+vermelho, o que corresponde à luz branca. Os outros códigos de cores, quando escritos na forma binária, podem ser interpretados conforme tabela 2.

– BUFFER DA IMPRESSORA

Os endereços entre 23.296 e 23.551, ou 5BOO a 5BFFh são usados como o buffer da impressora (ZX Printer, ALphacom 32, Timex 2040 ou TK 500S quando e se for produzida). Esses 256 bytes são suficientes para

Tabela 1
TABELA DE ATRIBUTOS

PAPER	INK								MODO
	black	blue	red	mag.	green	cyan	yell.	white	
BLACK	0	1	2	3	4	5	6	7	normal
	64	65	66	67	68	69	70	71	bright
	128	129	130	131	132	133	134	135	flash
	192	193	194	195	196	197	198	199	bright+flash
BLUE	8	9	10	11	12	13	14	15	normal
	72	73	74	75	76	77	78	79	bright
	136	137	138	139	140	141	142	142	flash
	200	201	202	203	204	205	206	207	bright+flash
RED	16	17	18	19	20	21	22	23	normal
	80	81	82	83	84	85	86	87	bright
	144	145	146	147	148	149	150	151	flash
	208	209	210	211	212	213	214	215	bright+flash
MAGENTA	24	25	26	27	28	29	30	31	normal
	88	89	90	91	92	93	94	95	bright
	152	153	154	155	156	157	158	159	flash
	216	217	218	219	220	221	222	223	bright+flash
GREEN	32	33	34	35	36	37	38	39	normal
	96	97	98	99	100	101	102	103	bright
	160	161	162	163	164	165	166	167	flash
	224	225	226	227	228	229	230	231	bright+flash
CYAN	40	41	42	43	44	45	46	47	normal
	104	105	106	107	108	109	110	111	bright
	168	169	170	171	172	173	174	175	flash
	232	233	234	235	236	237	238	239	bright+flash
YELLOW	48	49	50	51	52	53	54	55	normal
	112	113.	114	115	116	117	118	119	bright
	176	177	178	179	180	181	182	183	flash
	240	241	242	243	244	245	246	247	bright+flash
WHITE	56	57	58	59	60	61	62	63	normal
	120	121	122	123	124	125	126	127	bright
	184	185	186	187	188	189	190	191	flash
	248	249	250	251	252	253	254	255	bright+flash

armazenar 32 caracteres em sua forma de alta resolução, com os primeiros 32 bytes armazenando os bits da primeira linha dos caracteres, os próximos 32 bytes armazenando os bits da segunda linha, e assim por diante.

Se não houver impressora ligada ao seu micro, esta área pode ser usada como área de armazenamento de rotinas em código de máquina, elaboradas pelo usuário. . . (boa dica essa !) Não há interferência com nenhum programa que esteja no micro, mesmo que ele ocupe toda a memória.

Tabela 2

COR	CÓDIGO	BINÁRIO	VÉRTICES ACESOS
Black	0	000	
Blue	1	001	Blue
Red	2	010	Red
Magenta	3	011	Red Blue
Green	4	100	Green
Cyan	5	101	Green Blue
Yellow	6	110	GreenRed
White	7	111	GreenRed Blue

Para os interessados, aí está uma boa dica para se obter mais que 8 cores do nosso TK 90X.

– VARIÁVEIS DE SISTEMA

As 182 posições de memória entre 23.552 e 23.733, ou 5COO a 5CB5h, armazenam as diferentes variáveis do sistema do TK 90X. Se estiver conectado ao seu micro uma Interface One da Sinclair, e um Microdrive, haverá uma ampliação dessa área, com o surgimento de novas variáveis de sistema que cuidarão exclusivamente desses periféricos. No capítulo seguinte estas variáveis serão melhor estudadas.

– MAPEAMENTO DO MICRODRIVE

Esta área da memória começa no endereço 23.734, ou 5CB6h, e tem apenas uma existência teórica na versão standard do TK 90X, ou seja, é uma área que não é usada com esse propósito, a menos que um Microdrive seja ligado a ela. (Como é, Microdigital, sai ou não sai esse periférico?)

Enquanto sem o periférico, esta área também pode ser usada para armazenar rotinas em código de máquina, elaboradas pelo usuário.

– ÁREA DE INFORMAÇÃO DE CANAIS

Esta área especial de memória tem início no endereço apontado pela variável de sistema CHCADD, armazenada nos endereços 23.631 e 23.632, ou 5C4F e 5C5Oh. Esta área tem tamanho variável, mas termina num endereço que armazena um marcador final de valor 128 ou 80h.

Na versão padrão do TK 90X, ou seja, o micro sem periféricos conectados, existem detalhes de entrada/saída para quatro canais. Esses canais são:

– Canal "K" *(keyboard)* – permite entradas via teclado e saídas para a parte inferior da tela.

– Canal "S" *(screen)* – não permite nenhuma entrada, mas somente saídas normais para a tela.

– Canal "R" *(RS 232?)* – também não permite entradas, mas permite saídas para a parte inferior da tela. O tamanho desta área de vídeo pode ser expandido, se necessário.

– Canal "P" *(printer)* – também não permite entrada, mas somente saída para a impressora, via conector traseiro.

As informações de canais consistem, por canal, em 5 bytes de dados. Esses bytes fornecem o endereço da rotina de saída, que toma dois bytes; o endereço da rotina de entrada, que toma outros dois bytes; e o nome do arquivo, que é um simples código de letra correspondente.

Como existem quatro canais-padrão e um indicador de final, esta área de informação de canais no TK 90X standard ocupa os 21 endereços entre 23.734 e 23.754, ou 5CB6 e 5CCAh.

Os percursos ou caminhos que os dados percorrem para chegar a ou vir desses canais é chamado de fluxo. Associados a esses quatro canais-padrão existem quatro fluxos:

Fluxo #0 – imprime dados na parte inferior da tela que
Fluxo #1 – são lidos do teclado.
Fluxo #2 – escreve dados na parte superior da tela, mas não pode ler dados.
Fluxo #3 – envia dados de saída para a impressora, mas também não lê dados.

Cada instrução de entrada/saída de dados usa automaticamente um desses fluxos. O comando PRINT usa o fluxo # 2 e o comando LPRINT usa o fluxo #3.

Portanto, PRINT "TK 90X" é uma abreviação de PRINT #2; "TK 90X"

Se você escrever LPRINT "TK 90X", o micro manda a mensagem para a impressora; mas, se você digitar LPRINT # 2; "TK 90X", essa mensagem é desviada para a tela (isso funciona com os diversos comandos de saída, tais como LIST, LLIST etc.).

Analogamente, se você escrever PRINT #1; AT O, O; "TK 90X" (ou PRINT #O), o micro imprimirá a mensagem na linha 0, coluna 0, da parte inferior da tela (ou seja, a partir da linha 23). Experimente, pois isso é muito interessante na impressão de mensagens, sem interferir no arquivo de imagens!

– ÁREA DE PROGRAMAS BASIC

Esta área da memória armazena as linhas de programas em Basic, se existirem. O seu tamanho depende justamente de quantas linhas existirem.

O início da área de programa é sempre dado pelo valor armazenado na variável de sistema PROGBAS, que ocupa os endereços 23635 e 23636, ou 5C53 e 5C54h.

Note que na versão standard do TK 90X, a variável de sistema PROGBAS vai indicar que esta área de programas Basic começa no endereço 23755 ou 5CCBh, e isto sempre vai acontecer, a menos que haja um microdrive conectado ao micro, ou seja, a área de mapeamento do microdrive está sendo usada ou endereços extras foram reservados para informações adicionais de canais.

Na área de programas Basic, as linhas são armazenadas no seguinte formato:

Os primeiros dois bytes de qualquer linha armazenam o seu número, sendo o primeiro byte o mais significativo, e o outro, o menos significativo.

O terceiro e o quarto bytes armazenam o comprimento da linha. Aqui, o byte menos significativo vem antes que o byte mais significativo. O comprimento da linha significa o número de bytes, a partir do quinto byte até o byte final, o ENTER (indicador de final de linha), inclusive.

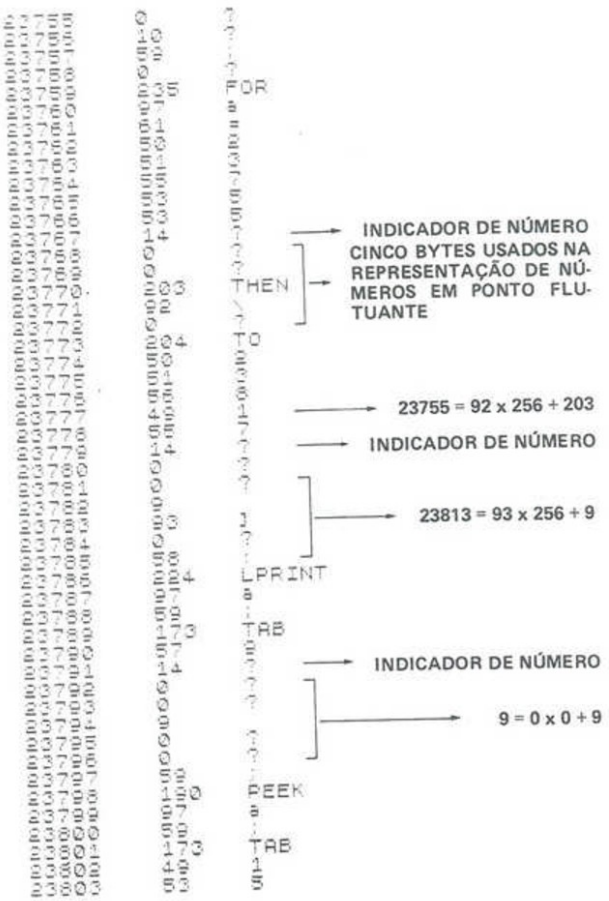
Agora começa a linha de Basic propriamente dita. Os códigos de TK 90X são usados para símbolos, palavras chaves, outros caracteres, e os códigos ASCII para caracteres alfanuméricos standard.

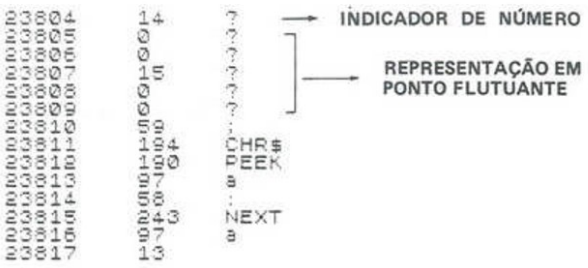
O último byte de qualquer linha é sempre o caractere ENTER.

Dentro de uma linha em Basic, múltiplas declarações são separadas entre si por dois pontos, caractere 58, ou 3AH. Note que se numa linha Basic existe um número, em decimal, ele é armazenado em caracteres ASCII e seguido pelo caractere NÚMERO – código 14 ou OEh, e em seguida a forma de ponto flutuante para números quaisquer ou a forma inteira para números inteiros da faixa – 65.535 a +65.535, onde, em ambos os casos, são consumidos cinco bytes. Isto significa que são reservados seis bytes de RAM para cada número decimal que estiver incluído num programa Basic !!!

O programa-demonstração a seguir, mostra o seu próprio conteúdo armazenado na área de programas.

```
 10 FOR a=23755 TO 23817: PRINT
a;TAB 9;PEEK a;TAB 15;CHR$ PEEK
a: NEXT a
```






```
23804    14      ?
23805    0       ?
23806    0       ?
23807    15      ?
23808    0       ?
23809    0       ?
23810    58      :
23811    194     CHR$
23812    190     PEEK
23813    97      a
23814    58      :
23815    243     NEXT
23816    97      a
23817    13
```

– ÁREA DE VARIÁVEIS

O endereço inicial desta área que armazena as variáveis correntes de um programa Basic é sempre dado pelo valor contido na variável de sistema VARADD, que ocupa os endereços 23.627 e 23.628, ou 5C4B e 5C4Ch.

No TK 90X, o tamanho dessa área não vai mudar durante a execução de um programa Basic, mas sim quando nela for introduzida uma nova variável.

O último endereço desta área sempre contém o marcador de final de área, caractere 128, ou 80h.

O programa a seguir mostra o conteúdo da área de variáveis, que contém apenas a variável de controle do loop.

Programa-demonstração da área de variáveis:

```
  0 REM Nao digite as linhas "R
EM" - somente a linha 10
  1 REM primeiro digite em modo
 direto:
  2 REM PRINT PEEK 23627+256*PE
EK 23628
  3 REM O resultado sera 23818
  4 REM Que por sua vez e o val
or inicial da variavel de contro
le (a) do loop
  5 REM
 10 FOR a=23804 TO 23823: PRINT
a;TAB 9;PEEK a;TAB 15;CHR$ PEEK
a: NEXT a
```

– ÁREA DE EDIÇÃO DE LINHAS

O início desta área que armazena a linha Basic que está sendo digitada, ou editada, é sempre dado pelo valor armazenado na variável de sistema INADD, que ocupa os endereços 23.641 e 23.642, ou 5C59 e 5C5ah.

Quando a parte inferior da tela mostra apenas o cursor intermitente, a área de edição de linhas tem três bytes reservados para ela. O primeiro, cujo endereço é dado pelo conteúdo da variável de sistema CURADD, armazena um caractere ENTER e o segundo armazena um marcador de final – caractere 128, ou 80h.

Então, os caracteres digitados a partir do teclado são transferidos para esta área que é expandida para armazená-los.

Um procedimento similar é obtido quando se digita a tecla EDIT (CAPS SHIFT e tecla 1), para corrigir uma linha de programa. Em primeiro lugar, a área de edição é expandida para o comprimento igual ao da linha a ser editada. Então esta linha é copiada sobre o conteúdo anterior da área. (Um modo de se "limpar" esta área.)

– INPUT DADOS E ÁREA DE TRABALHO TEMPORÁRIO

Esta área da memória é usada para muitos propósitos, como, por exemplo, dados de entrada, concatenação de cadeia de caracteres etc. O endereço inicial da área é dado pelo valor armazenado na variável de sistema WORKPT, de endereços 23.649 e 23.650, ou 5C61 e 5C62h. Quando se faz necessária uma maior área de trabalho, esta área é expandida. Após o uso, ela é então esvaziada.

– ÁREA DA PILHA DE CÁLCULO

Esta é uma área muito importante da memória, e começa no endereço apontado pela variável de sistema STKCEND, de endereços 23.651 e 23.652, ou 5C63 e 5C64h, e vai até o endereço apontado pela variável de sistema ADSPFREE, de endereços 23.653 e 23.654, ou 5C65 e 5C66h.

Esta área é usada para armazenar os números em ponto flutuante em cinco bytes e, quando manipulando séries de caracteres, utiliza grupos de cinco bytes como parâmetros delas.

A manipulação desta pilha de cálculo é feita sob a forma conhecida como "LIFO", do inglês "Last-in First-out", ou seja, o último valor depositado na pilha é o primeiro a ser retirado.

– ESPAÇO LIVRE

A área de memória entre a pilha de cálculo e a pilha da máquina representa a quantidade de memória que é disponível ao usuário. Na versão do TK 90X de 16 K existem nominalmente 8.839 endereços nesta área, quando se liga o computador. Para a versão de 48 K de RAM, existem esses 8.839 mais os adicionais 32 K. Entretanto, é interessante notar que o valor mais baixo aceitável para CLEAR é 23.821, que traz a RAMTOP 8.878 bytes para baixo, ou na RAM de 48 K, 8.878 mais 32 K (41646 K).

– ÁREA DA PILHA DA MÁQUINA

O microprocessador Z80 deve ter uma área de trabalho disponível apenas para ele, que é chamada a pilha da máquina, onde ele vai manipular valores dos programas armazenados na memória.

– ÁREA DA PILHA DE GOSUB

Sempre que existir um loop de GOSUB, o número da linha desse loop será armazenado nesta área.

Cada número de linha de GOSUB requer três endereços. O mais alto armazena o número da declaração dentro de uma linha de programa Basic, para onde o retorno deverá ser feito. O próximo endereço armazena a parte menos significativa do número da linha de programa, e o último armazena a parte mais significativa desse número.

A variável de sistema RAMTOP, que ocupa os endereços 23.730 e 23.731, ou 5CB2 e 5CB3h, armazena o endereço que contém o valor 62, considerado como sendo o último endereço da área de Basic.

– ÁREA DE DEFINIÇÃO DE GRÁFICOS DO USUÁRIO

Os últimos 168 endereços da memória RAM do TK 90X são reservados para armazenar as representações dos bits dos caracteres gráficos definidos pelo usuário, a menos que tenha sido invadida pelo sistema Basic; quer pela modificação de endereços, através do uso de CLEAR, ou por um programa Basic muito extenso.

Como parte do procedimento de inicialização do TK 90X, a representação bit a bit das letras de "A" até "U" são copiadas nesta área. O usuário, ao definir alguns ou todos os 21 caracteres gráficos, muda esta representação.

O último endereço da memória é sempre fornecido pela variável de sistema MEMAVLB, que ocupa os endereços 23.732 e 23.733, ou 5CB4 e 5CB5h.

Na versão de 16 K do TK 90X, este endereço deve ser 32.767; e na de 48K, deve ser 65535, valor esse obtido pela expressão Basic, em modo direto:

PRINT PEEK 23732+256 * PEEK 23733

Esses nomes complicados para estas variáveis não querem dizer absolutamente nada, bem como não têm nada a ver com a linguagem Basic nem com a linguagem de máquina. Não há, portanto, necessidade de se guardá-los.

2

Variáveis que controlam o sistema

Os bytes na memória, do endereço 23.552 até o endereço 23.733, são reservados para uso específico do sistema. Você pode ver o conteúdo de qualquer desses endereços; convém até anotar esses valores, pois podem ser úteis mais tarde. Para ver esses conteúdos, você simplesmente digita:

PRINT PEEK endereço

Podemos também mudar esses valores iniciais, sem perigo algum de alterar o hardware standard da máquina. É somente praticando, e "fuçando", que se descobre como ocorrem as coisas dentro dessa pequena maravilha. O máximo que pode acontecer, quando se altera uma variável de sistema, é um *crash*, e o passo seguinte nesse caso é desligar a máquina e tornar a ligá-la. Os nomes dessas variáveis não têm nada a ver com a linguagem Basic, e o computador não as reconhece pelo nome (felizmente...). Você pode usar em seus programas nomes semelhantes de variáveis, que o computador não vai confundir a sua variável com uma variável de sistema.

As abreviações utilizadas na coluna 1 da tabela a seguir tem o seguinte significado:

x – alterando o valor inicial da variável, provavelmente ocorrerá um *crash* no sistema.

n – não haverá efeito nenhum, alterando o valor inicial.

O número da coluna 1 é o número de bytes daquela variável.

Para dois bytes, o primeiro é o menos significativo – o contrário do que normalmente se espera. Portanto, para se dar um POKE de um valor v numa variável de sistema de dois bytes, de endereço n, use

POKE n, v-256 * INT (V/256)
POKE n + 1, INT (v/256)
E, para ver o valor, use a expressão:
PRINT PEEK n + 256 * PEEK (n + 1)

Se você quiser alterar algum endereço n, como no caso de aumentar a área de caracteres gráficos definidos pelo usuário, visto no capítulo anterior, deve efetuar alguns cálculos:

Chamemos de bms o byte menos significativo
BMS o byte mais significativo
VARSIT a variável de sistema que se queira alterar.

Os cálculos são:
XMS = INT (n/256)
bms = n – BMS *256
E, a seguir:
POKE VARSIST, bms
POKE POKE VARSIST + 1, BMS
Cuidado com as maiúsculas e minúsculas!

VARIÁVEIS DO SISTEMA – TABELA DE DADOS

Nota	Endereço	Nome	Conteúdo
n8	23552/5COO	KBDWORK	Esta variável consiste em 8 bytes, cada um deles contendo informação sobre a tecla pressionada, ou o período de repetição; o código em modo *extended* etc.
n1	23560/5CO8	KEYPRS	Esta variável contém sempre o código da última tecla pressionada, tendo em conta o modo. Só é alterada quando se pressiona outra tecla. A repetição automática afeta a variável. Passando-se a zero, pode-se verificar se foi pressionada alguma tecla.
1	23561/5CO9	RPTDLAY	Tempo, em 60 ciclos por segundo, durante o qual é necessário pressionar uma tecla para acionar o auto-repeat (repetição automática). Possui o valor inicial 35, podendo ser alterado para mais ou para menos. (Cuidado com valores muito baixos – a repetição é muito rápida...) Tente.

1	23562/5COA	RPTCCLE	Atraso, em 60 ciclos por segundo, entre as sucessivas repetições de uma tecla. Inicialmente contém o valor 5. Pode-se diminuir para o mínimo de 1, a fim de tornar mais rápida a repetição.
n2	23563/5COB	PT DEF	Endereço dos argumentos das funções definidas pelo usuário. Valor inicial 0.
n1	23565/5COD	K CLR	Quando é escrito diretamente no teclado um código de comando, o segundo byte, ou seja, a cor e o uso ou não de FLASH é guardado aqui, enquanto o código de INK, PAPER, FLASH ou INVERSE é impresso. Depois disso, a ROM consulta este byte para impressão a seguir do código de comando.
n2	23566/5COE	TVCLR	Esta variável é usada pela rotina de impressão para guardar AT, TAB e os comandos de cor dirigidos para a tela.
x38	23568/5C10	PSTRM	Esta variável é usada para guardar os deslocamentos de CHCADD. Para um total de 19 arquivos, sendo 16 do usuário e 3 da máquina, existe um deslocamento. Quando este é somado a CHCADD, aponta para um endereço que constitui o início da rotina de tratamento desse arquivo.

2	23606/5C36	PTBL CHR	Indica o endereço do conjunto de caracteres, menos 256 (incluindo, no conjunto, todos os caracteres de código entre 32 e 127, padrão ASCII). A variável contém normalmente o endereço 15360 ou 4C00 (conjunto de caracteres da ROM), mas este valor pode ser alterado, levando-a a apontar para um novo conjunto de caracteres definido pelo usuário.
1	23608/5C38	BUZCCLE	Som produzido pelo computador quando uma linha de programa em edição ultrapassa o comprimento de 23 linhas da tela (tela cheia). Para alterar este comprimento, modifique o conteúdo desta variável, inicialmente em 64.
1	23609/5C39	KCLICK	Duração do som produzido ao se pressionar uma tecla. Pode-se aumentar seu valor inicial, a fim de tornar mais "simpático" o som produzido.
1	23610/5C3A	ERRCD	Indica o código de mensagens menos 1. Inicialmente em 255. Se for alterada por uma listagem de programa Basic, quando este programa terminar, a máquina imprime a mensagem de erro desejada.
x1	23611/5C3B	SFLAG O	Esta variável contém um byte cujos valores contém bandeiras (indicadores) que controlam o sistema Basic.

x1	23612/5C3C	SFLAG 1	Bandeiras associadas à impressão na tela e na impressora.
x2	23613/5C3D	P ERR	Esta variável aponta para um elemento da pilha da máquina. Quando ocorre um erro, é para este endereço que a execução salta depois da pilha ser limpa por RST 08 (instrução em código de máquina). Alterando este elemento, é possível escrever novas rotinas de tratamento de erro, ou provocar um *crash* no sistema.
n2	23615/5C3F	PLIST	Esta variável aponta para o endereço de retorno da pilha da máquina, para o qual salta a execução após uma listagem automática.
n1	23617/5C41	CURSOR	Esta variável especifica o cursor em uso (K, L, C, E ou G).
2	23618/5C42	LNJMP	Linha para onde deve saltar a execução do programa. Usada nas instruções GOTO e GOSUB.
1	23620/5C44	INSTRNR	Número da declaração da linha para onde deve saltar a execução. Alterando primeiro a variável LNJMP e depois esta, força-se um salto para uma determinada declaração da linha.
2	23621/5C45	EXCLINE	Número da linha onde se encontra a instrução em execução.
1	23623/5C47	SUBLEXC	Número da instrução em execução dentro de uma linha.

1	23624/5C48	BORCLR	Esta variável contém os atributos da parte inferior da tela. É uma experiência muito interessante alterar seu valor e imprimir mensagens com diversos valores.
2	23625/5C49	CURLINE	Número da linha onde se encontra o cursor.
x2	23627/5C4B	VARADD	Indica o endereço inicial da área de variáveis Basic, quando da execução de um programa.
n2	23629/5C4D	XVARADD	Endereço da variável em atribuição.
x2	23631/5C4F	CHCADD	Indica tabela de endereços usada por PSTRM.
x2	23633/5C51	IOADD	Indica endereço da tabela anterior (de endereços de tratamentos de arquivos) que está sendo usado pela rotina de tratamento.
x2	23635/5C53	PROGBAS	Indica o endereço de início da área de programas Basic.
x2	23637/5C55	NEXEXC	Indica o endereço da linha de programa Basic seguinte a que está sendo executada.
x2	23639/5C57	ENDDATA	Aponta para o separador final do último elemento de uma listagem DATA. Se não existir DATA no programa, aponta para 80h, no fim das informações de canal.
x2	23641/5C59	INADD	Endereço da instrução a ser digitada no teclado.

2	23643/5C5B	CURADD	Endereço do cursor no interior da linha avaliada.
x2	23645/5C5D	CHNXADD	Endereço do próximo caractere a ser interpretado.
2	23647/5C5F	SYCHADD	Endereço do caractere que se encontra depois do sinal "?".
x2	23649/5C61	WORK PT	Endereço da área de trabalho temporário.
x2	23651/5C63	STKEND	Endereço inferior da pilha de cálculo.
x2	23653/5C65	ADSPFREE	Endereço do início da memória livre.
n1	23655/5C67	BREGCAL	Registro de cálculo usado para fins diversos (contagem).
n2	23656/5C68	MEMCADD	Endereço usado para as seis memórias de cálculo (normalmente MENSPCAL, mas nem sempre).
1	23658/5C6A	SFLAG 2	Mais bandeiras controladoras do sistema.
x1	23659/5C6B	SIZE	Número de linhas (incluindo uma em branco) da parte inferior da tela. Ela pode ser alterada fazendo com que o programa ocupe toda a tela, mas cuidado: antes de cada instrução que ocupe esta área (por exemplo, INPUT), a variável deverá conter o valor original(2), ou seja, reabrir espaço na parte inferior.
2	23660/5C6C	LIST NR	Número da linha superior em listagem automática.
2	23662/5C6E	CONTJMP	Número da linha para onde salta o comando CONT.

1	23664/5C70	CONTNR	Número da instrução da linha destino após o comando CONT.
n1	23665/5C71	SFLAG 3	Mais bandeiras controladoras do sistema.
1	23681/5C81		Não usada pelo sistema. Alguns programas em linguagem de máquina têm a instrução RET (201) armazenada nesta posição, para evitar que, ao se dar o comando LOAD "" CODE, se tenha acesso aos seus códigos. . .
2	23682/5C82	HVBFFIN	Número de colunas (32) e linhas (24) do final da área de entrada.
2	23684/5C84	DFPSPRT	Endereço da posição PRINT no arquivo de imagens. Pode ser dirigida para outra posição.
2	23686/5C86	DFPSPRTL	Idêntica à anterior, mas para a parte inferior da tela.
x1	23688/5C88	HVPOS	Número de colunas para a posição PRINT.
x1	23689/5C89		Número de linhas para a posição PRINT.
x2	23690/5C8A	HVPOSL	Idêntica à anterior, mas para a parte inferior da tela.
1	23692/5C8C	SCRINC	Contador de "scroll" da imagem: contém um a menos que o número de scrolls realizados antes de a imagem parar imprimindo a mensagem "scroll?". Se for colocado um valor maior aqui, a imagem rolará ininterruptamente.
1	23693/5C8D	ATCLRP	Atributos permanentes definidos por declarações INK, PAPER etc.

1	23694/5C8E	MASKCLRP	Usado para atributos transparentes. Qualquer bit igual a 1 indica que o atributo correspondente não é retirado de ATCLRP, mas do valor que já está na tela.
n1	23695/5C8F	ATCLRT	Atributos provisórios em uso, em instruções tipo PLOT, DRAW etc.
n1	23696/5C90	MASKCLRT	Como MASKCLRT, mas provisório.
1	23697/5C91	SFLAG 4	Outras bandeiras controladoras do sistema.
n30	23698/5C92	MEMSPCAL	É nesta área que a unidade de cálculo aritmético pode guardar até 6 números diferentes, em notação de ponto flutuante, com 5 bytes cada, utilizando memórias especiais.
2	23728/5CBO	NMIVCT	Ex-vetor de interrupção, não usado devido às características de programação da ROM. Pode ser usada sem restrições pelo usuário.
2	23730/5CB2	RAMTOP	Esta variável contém o endereço do último byte da área Basic.
2	23732/5CB4	MEMAVLB	Endereço do último byte físico da RAM.
n2	23666/5C72	STRVLEN	Comprimento da série de caracteres em atribuição.
n2	23668/5C74	SYTADD	Endereço do elemento seguinte na tabela de sintaxe da ROM. Esta tabela define o local onde se encontra a rotina correspondente a cada comando e o modo de obter a informação necessária.

2	23670/5C76	INITRND	O primeiro número para a instrução RND. Esta variável é inicializada por RAND.
3	23672/5C78	TVCOUNT	Contador de imagens de 3 bytes, incrementado a cada ciclo de alimentação.
2	23675/5C7B	UDGRAPH	Endereço do primeiro caractere gráfico definido pelo usuário. RAMTOP não afeta esta área.
1	23677/5C7D	LSTPLOT	Usada para armazenamento provisório da coordenada x quando são realizados cálculos para definição da posição de PLOT.
1	23678/5C7E	COORDS	Idêntica à anterior, mas para a coordenada y.
1	23679/5C7F	POSIMPR	Número da posição de impressão, em 32 colunas.
1	23680/5C80	PRTADD	Byte menos significativo do endereço da posição que se segue para LPRINT (no buffer da impressora).

Alguns comentários sobre as variáveis de sistema

1 – KBDWORK

Esta variável armazena:

a) 255 se nenhuma tecla foi pressionada ou
b) o código do maior caractere em branco à esquerda impresso em uma tecla.

No último caso, o código pode ser considerado como sendo aquele caractere que INKEY$ produziria se CAPS LOCK (tecla 2) estivesse acionado.

Esta propriedade pode ser usada com vantagens, quando se usa INKEY$ em um programa. Digite o programa a seguir e veja qual o efeito que CAPS SHIFT ou SYMBOL SHIFT produzem quando pressionados com outras teclas:

```
10 PRINT AT 0,0; INKEY$; "  ": REM 4 espaços dentro das aspas
20 GOTO 10
```

Como você pode notar, o caractere produzido não depende apenas de qual tecla foi pressionada, mas também em que modo se encontrava o computador. Isto conduz a linhas meio "complicadas" quando se usa INKEY$, por exemplo, ao final de um programa:

```
9000 PRINT "Quer rodar outra vez? (s/n)": PAUSE 0:
     IF INKEY$ = "s" OR INKEY$ = "S" OR INKEY$ = "NOT"
     THE RUN
9010 NEW
```

Podemos utilizar também o byte do endereço 23552:

```
10 PRINT AT 0,0; CHR$ PEEK 23552; "  ";
20 GOTO 10
```

Você vai perceber que qualquer que seja a tecla pressionada, o caractere produzido será aquele que aparece em branco no lado esquerdo da tecla que está sendo pressionada. A solução para o problema:

```
9000 PRINT "Quer rodar outra vez? (s/n)": PAUSE 0:
     IF CHR$ PEEK 23552 = "S" THEN RUN
9010 NEW
```

Se você pretende usar esta técnica em seus programas, faça da seguinte maneira:

```
10 LET kbd = 23552
```

E mais adiante,

```
... IF CHR$ PEEK kbd = ... THEN ...
```

Agora experimente o programa abaixo, para ver qual endereço desta variável utilizar em seus programas:

```
10 FOR f = 23552 TO 23559
20 PRINT INKEY$; "  "; CHR$ PEEK f;
30 GOTO 20
```

Quando quiser mudar para o próximo endereço, ou para o próximo f, basta interromper o programa e digitar NEXT f.

Notou que a penúltima tecla pressionada é impressa com a última, através de CHR$ PEEK f? Retire este comando da linha 20 e preste atenção nas diferenças.

2 – KEYPRS

O conteúdo deste endereço vai produzir o código da última tecla pressionada, esteja você pressionando outra ou não. Como no caso da variável anterior, se mais de uma tecla é pressionada, então o código da primeira a fazer contato tem prioridade sobre a outra. É importante notar, que embora CAPS SHIFT ou SYMBOL SHIFT sozinhas não alterem o conteúdo de KBDWORK, ou mesmo de INKEY$, juntas produzem código 14, que normalmente é usado para representação de números em listagens Basic do TK 90X.

Existem outras quatro combinações de teclas que produzem valores para bytes de 23560 e códigos de INKEY$, que não aparecem no manual do TK 90X. São elas:

Teclas normalmente usadas para produzir:	Valor do código INKEY$
GRÁFICOS (CAPS SHIFT & 9)	15
TRUE VIDEO (CAPS SHIFT & 3)	4
VÍDEO INVERSO (CAPS SHIFT & 4)	5
CAPS LOCK (CAPS SHIFT & 2)	6

3 – RPTDLAY E RPTCCLE

Experimente rodar o programa abaixo, que é auto-explicativo, para ver o que acontece com o dispositivo de repetição automática de teclado, e o tempo de leitura do teclado:

```
10 PRINT "RPTDLAY - 23561 = "; PEEK 23561 "RPTCCLE - 23562 = ";
   PEEK 23562
20 LET a = PEEK 23561 : LET b = PEEK 23562
30 INPUT "Digite valor para mudar RPTDLAY"; m
40 INPUT "Digite valor para mudar RPTCCLE"; n
50 POKE 23561, m: POKE 23562, n
60 INPUT " Experimente digitar qualquer coisa"; c$
70 PRINT c$
80 POKE 23561, a: POKE 23562, b
90 PRINT "Valor usado para RPTDLAY = "; m; "Valor usado para RPTCCLE
   = "; n
100 INPUT "Outra tentativa (s/n) ?"; a$
110 IF a$ = "s" THEN RUN
120 STOP
```


4 – MUDANDO MODOS DO CURSOR

A variável de sistema de endereço 23617 especifica o modo do cursor; através dela o programador pode forçar apenas o modo gráfico e o modo estendido, no próximo comando INPUT.

Para modo gráfico: POKE 23617,2: INPUT a$: PRINT a$

Para modo estendido: POKE 23617,1: INPUT a$: PRINT a$

Experimente o programa, alternando esses dois POKES:

```
10 FOR f = 0 TO 255
20 PRINT f; TAB 8;
30 INPUT a$: PRINT a$;: INPUT a$: PRINT a$
40 NEXT f
```

Portanto, através de CURSOR, podemos apenas entrar em modo gráfico e modo estendido.

Através de uma variável "misteriosa", localizada no endereço 23658, podemos também entrar em modo maiúsculo (CAPS LOCK). Digitando-se POKE 23658,8, o endereço de SFLAG 2, automaticamente estamos com letras maiúsculas.

Isso pode facilitar em certos programas que possuam uma linha do tipo

```
100 INPUT "Quer rodar outra vez (s/n)"; s$: IF s$ = "S" OR s$ = "s" THEN
    RUN
```

Podemos passar esta linha para:

```
100 POKE 23658,8: INPUT "Quer rodar outra vez (s/n); s$: IF s$ = "S" THEN
    RUN: POKE 23658,0: REM entra em modo maiúsculo e sai desse modo
    através de POKE 23658,0.
```

Você pode, também, dependendo do tipo de programa que esteja fazendo, perguntar logo de início se deseja trabalhar somente em letras maiúsculas ou não.

Esta técnica é muito interessante quando existem nos programas diversos comandos tipo INPUT e INKEY$.

5 – CORES DA TELA

BORCLR

Embora o TK 90X proteja o usuário contra acidentes que fazem INK e PAPER, em comandos INPUT, serem da mesma cor, você, se quiser, pode alterar somente as duas linhas da parte inferior do vídeo, alterando os valores da variável de sistema BORCLR, de endereço 23624. Isso pode ser particularmente interessante, quando você não quer de modo algum que alguém veja a listagem de seu programa, e, dessa forma, não é permitido imprimir qualquer coisa na parte inferior da tela.

É interessante notar que, através do uso dessa variável, podemos também fazer FLASH e BRIGHT nas linhas de INPUT, adicionando:

128 para FLASH 1 e
64 para BRIGHT 1

O novo valor de BORCLR permanece inalterado até ser executado um novo comando BORDER ou um NEW.

Por exemplo, para produzir uma borda magenta com tinta amarela e papel magenta, nas linhas inferiores, com FLASH 1 e BRIGHT 1, digite:

```
BORDER 3:POKE 23624, 128 + 64 + 3 * 8 + 6 : CLS
```

ATCLRP – MASKCLRP – ATCLRT – MASKCLRT

Estas variáveis do sistema simplesmente armazenam os valores que estão sendo usados para INK, PAPER, BRIGHT e FLASH. A diferença entre os nomes, ou seja, a letra P e T, significa Permanente ou Temporário (estas, quando incluídas em algum comando de saída de dados).

ATCLRP e ATCLRT são pouco utilizadas, mas para referência os valores a serem adicionados nelas, para tratamento de cores são

ATCLR = 8 * (cor de PAPER) + cor da INK + (128 para FLASH 1) + (64 para BRIGHT 1).

Já MASKCLRP e MASKCLRT são mais úteis. Ocupam respectivamente os endereços 23694 e 23696.

Qualquer bit de um byte de uma destas variáveis, que tiver o valor 1, mostra que o bit correspondente de atributos para impressão deve ser tomado da posição corrente de impressão da tela, como no caso de INK, PAPER, BRIGHT e FLASH 8.

O ponto mais interessante destas variáveis é que nós podemos limitar seu efeito a apenas cores primárias (azul, vermelho e verde) de códigos do TK, respectivamente 1, 2 e 4. Os valores para POKE nas variáveis são:

BIT	EFEITO	VALOR
0	azul INK 8	1
1	verm INK 8	2
2	verde INK 8	4
3	azul PAPER 8	8
4	verm PAPER 8	16
5	verde PAPER 8	32
6	BRIGHT 8	64
7	FLASH 8	128
		255

6 – O RELÓGIO DO TK 90X

Existe no conjunto das variáveis do sistema do TK 90X, uma variável, que constantemente altera seus valores, como que contando "tempo", chamada TVCOUNT, que possui seu valor inicial incrementado 60 vezes por segundo. Esta freqüência é igual à freqüência ou ciclagem da área, e também ao número de vezes por segundo que uma nova imagem é enviada à televisão para formar o que vemos.

TVCOUNT é inicializada em zero, quando o computador é ligado, e incrementada a cada 20 milissegundos aproximadamente, a menos que um comando SOUND, uma operação com fita cassete ou um outro periférico estejam ligados ao TK 90X.

Esta é a justificativa para o comando PAUSE n, que significa simplesmente "aguarde até que TVCOUNT seja incrementado de n".

TVCOUNT está armazenada em 3 bytes: 23672, 23673 e 23674. Cada byte tem 8 bits, e o valor máximo de TVCOUNT é 2 elevado a 24 menos 1, ou seja, 16777215, o que corresponde a aproximadamente 3 dias, 21 horas e alguns minutos, a partir do momento em que o computador for ligado (se você fizer esta experiência, faça-a antes que acabe a garantia). Atingindo esse valor, TVCOUNT é zerada e inicia nova contagem.

O valor de TVCOUNT pode ser encontrado através da linha:

```
PRINT PEEK 23672 + 256 * PEEK 23673 + 65536 * PEEK 23674
```

Experimente digitar o programa a seguir, que simula um relógio muito simples:

```
10 LET tvcount = 23672
20 LET min = 0
30 POKE tvcount,0: POKE tvcount + 1,0: POKE tvcount + 2, 0
40 PRINT AT 1,7; "Segundos = ";
50 LET t = PEEK tvcount + 256 * PEEK (tvcount + 1) + 65536 * PEEK
   (tvcount + 2)
60 LET seg = INT (t/50): REM desconte um tempo de processamento
70 PRINT AT 1,1; seg
80 LET min = INT (seg/60)
90 PRINT AT 3,7; "Minutos = "; AT 3, 18; min
100 GOTO 50
```

O potencial desta variável é muito grande: serve para contador de tempo em diversos tipos de programas, até mesmo para aqueles que pretendem dar assessoria por computador por hora!

7 – ROLANDO A IMAGEM

Um pequeno problema que surge para programadores em Basic do TK 90X é como não deixar o computador perguntar "Scroll?" a cada vez que a tela é preenchida com dados.

Existe uma variável de sistema, chamada SCRINC (de INCrementador de SCRoll), de endereço 23692 que possui o valor de 1 a menos que o número de linhas de tela que serão roladas para cima até a próxima pergunta "Scroll?" (normalmente, esse valor é menor que 23). Portanto, devemos colocar nessa variável um valor maior que o número de linhas que rolam para cima na tela, ou seja, de imediato, 255, que é o valor máximo.

Experimente:

```
10 PRINT AT 21,31'
20 PRINT PEEK 23692
30 GOTO 20
```

Rode o programa e introduza a seguir a linha 15 POKE 23692,255, e rode novamente.

8 – VARIÁVEIS DE CONTROLE DA MEMÓRIA

Existem algumas variáveis do sistema, que servem apenas para o computador saber o estado em que se encontra a sua memória, como, por exemplo, saber onde começa o programa em Basic, onde estão as variáveis desse programa etc. Muitas dessas variáveis do sistema têm pouco interesse para o programador. Veremos algumas delas.

Progbas

Os endereços 23635 e 23636 dizem ao computador onde começa o programa Basic.

PRINT PEEK 23635 + 256 * PEEK 23636 dá o valor de PROGBAS, ou seja, o início da área de programas Basic. Some cinco a este valor e você terá o endereço do primeiro caractere após a declaração REM da primeira linha de um programa qualquer (sem microdrive ou interface), cheia de códigos esquisitos, que são, na verdade, uma rotina em código de máquina.

Se você pretende criar uma rotina em linguagem de máquina e armazená-la dessa forma, não esqueça de deixar, após a declaração REM, tantos espaços quantos forem os bytes da sua rotina.

Para introduzir uma mensagem de autor de programa e deixá-la protegida na primeira linha, digite:

```
POKE (PROGBAS),0: POKE (PROGBAS + 1),0
```

Se o número anterior da primeira linha do seu programa for menor que 256, então, o primeiro dos dois comandos acima pode ser omitido. Este procedimento "renumera" a primeira linha do programa, atribuindo a ela o número 0, tornando impossível uma simples edição dessa linha.

Varadd

Esta variável armazena o endereço das variáveis Basic, através da expressão:

```
PRINT PEEK 23627 + 256 * PEEK 23628
```

As variáveis Basic estão situadas imediatamente após o programa Basic na memória do TK 90X; portanto, nós podemos encontrar o comprimento de um programa através da simples subtração de PROGBAS de VARADD:

PRINT "O programa tem"; 256 *((PEEK 23628 - PEEK 23636) + (PEEK 23627 - PEEK 23635));

Ramtop e Adspfree

Estas duas variáveis podem ser utilizadas para medir a quantidade de memória disponível ao usuário. No mapeamento da memória do TK 90X, você percebe que entre ADSPFREE e RAMTOP existem apenas o espaço livre, a pilha da máquina e a pilha de GOSUB.

RAMTOP é dada por:

```
PRINT PEEK 23730 + 256 * PEEK 23731
```

e ADSPFREE por:

```
PRINT PEEK 23653 + 256 * PEEK 23654
```

Como alternativa da segunda expressão, existe na ROM do TK 90X uma rotina que faz isso, e que começa no endereço 7962. Em vez de toda essa expressão você pode simplesmente digitar PRINT USR 7962, o que dá no mesmo.

Para estimar a quantidade de memória disponível, subtraia ADSPFREE de RAMTOP.

Ainda sobre ADSPFREE, você provavelmente deve saber como parar aquele programa maravilhoso que quando carrega da fita, automaticamente já entra rodando. Você sabe que ele foi gravado com SAVE. . . LINE. . . . Para interromper esse programa, você deve usar MERGE " ", em vez de LOAD " ", o que cria um grande problema para as software-houses. . . Pois bem, uma maneira de resolver esse problema é gravar o programa como sendo um bloco de bytes, através da inserção das seguintes linhas no final do programa:

```
9997 LET adspfree = PEEK 23653 + 256 * PEEK 23654
9998 SAVE "nome" CODE 23552, adspfree + 23500
9999 RUN
```

Dessa forma, toda a memória é enviada para a fita cassete, incluindo o programa, as variáveis Basic, as variáveis do sistema e a pilha do calculador, como um bloco de códigos, de tal forma que, quando for carregado de volta ao computador, este vai iniciar o seu processamento a partir da linha em que parou, ou seja, a partir da última linha do programa (não esqueça que, para carregar o programa, você deve usar LOAD " " CODE).

9 – DIVERSAS VARIÁVEIS

Para saber qual o número da próxima linha a ser interpretada, digite:

```
PRINT PEEK 23662 + 256 * PEEK 23663 (SERÁ O COMANDO TRACE?)
```

Para saber qual o número da próxima declaração naquela linha, digite:

```
PRINT PEEK 23664
```

Os endereços 23677 e 23678 armazenam respectivamente as coordenadas X e Y do último ponto plotado (variáveis de sistema LSTPLOT e COORDS). Você pode tratar isso criando duas variáveis em Basic, quando usando linhas com PLOT ou DRAW; se você começar o programa com

```
LET xo = 23677: LE yo = 23678
```

toda vez que precisar saber qual foi o último ponto plotado, PEEK xo e PEEK yo lhe darão as coordenadas desse ponto.

Para desenhar uma linha a partir do último ponto plotado DRAW (x - PEEK xo), (y - PEEK yo) ou, então, simplesmente, como você pode deduzir, dar POKEs nessas variáveis, para alterar a posição do último ponto plotado, ou então para desenhar pontos e retas sem utilizar os comandos PLOT e DRAW.

HVPOS – Armazena a posição corrente de impressão, mas não da maneira que você deve estar aguardando. Se você deu algum comando tipo PRINT AT lin, col, de linha e coluna, então,

endereço 23688 armazena 33 - col.
endereço 23689 armazena 24 - lin.

Portanto, para encontrar a posição corrente de impressão

LET lin. = 24 - PEEK 23689
LET col.= 33 - PEEK 23688

Se você pretende usar essa técnica em seus programas, até mesmo em conjunto com SCREEN$, então faça-o da seguinte forma:

DEF FN y( ) = 24 - PEEK 23689
DEF FN x( ) = 33 - PEEK 23688

INITRND – Esta é a variável utilizada para gerar o último número randômico (aleatório), e está armazenada nos endereços 23670 e 23671. Experimente digitar:

```
PRINT RND, (PEEK 23670 + 256 * PEEK 23671)/ 65536
```

e serão impressos dois valores iguais. Cada vez que RND é usada, INITRND é alterada pelo computador da seguinte maneira:

Novo INITRND = (75 (INITRND + 1)) mod. 65537 - 1, o que corresponde em Basic a:

```
LET INITRND = 75 * (INITRND + 1):LET INITRND = INITRND - 65537 * INT
(INITRND/65537) - 1
```

O novo valor de INITRND é então armazenado e dividido por 65536, para produzir um valor para RND entre 0 e 1.

DFPSPRT e DFPSPRTL – Estas duas variáveis de sistema, armazenadas nas locações 23684 e 23686, armazenam o endereço no arquivo de imagens das posições de impressão, tanto para a parte superior da tela, quanto para a parte inferior.

Normalmente não usamos PEEK e POKE no arquivo de imagens, pois temos três comandos que suprem qualquer necessidade, que são, POINT (x,y), PRINT e SCREEN$ (x,y).

Para qualquer posição de impressão na tela, nas posições X e Y, temos que:

```
DFPSPRT = 2048 + INT (Y/8 + 8) + (Y - 8 * INT (Y/8) ) * 32 + X
```

Não esqueça que cada caractere na tela é armazenado em 8 bytes na memória (um para cada linha de caractere). Os endereços desse grupo de 8 bytes de cada caractere da tela são separados de outro na mesma posição, em outra parte do arquivo de imagem, por 256. Portanto, se DFPSPRT é a primeira linha, na segunda é DFPSPRT + 256, na terceira é DFPSPRT + 512, e assim por diante.

Experimente digitar o programa a seguir, que ilustra bem essas experiências:

```
10 BORDER 0:PAPER 0:INK 6:CLS
20 PRINT AT INT (RND * 22), INT (RND * 32);
30 LET DF = PEEK 23684 + 256 * PEEK 23685
40 FOR c = 0 TO 1
50 FOR a = DF TO DF + 7 * 256 STEP 256
60 READ b:POKE a,b:NEXT a
70 NEXT c
80 RESTORE:GOTO 40
90 DATA 24,60,126,25,31,254,60,24,248,60,23,15,15,23,60,248
```

Se você realmente quiser proteger seus programas contra aqueles "piratas profissionais", dê um POKE na variável de sistema P ERR, de endereço 23613, com valor 0

```
POKE 23613,0
```

E coloque esse comando como sendo o primeiro da primeira linha do programa, para então experimentar dar um BREAK no programa e ver o que acontece...

Capítulo **3**

Algo sobre os comandos IN e OUT de Basic

Uma falha gritante do manual do TK 90X!

Estes dois comandos são importantíssimos, tanto a nível de Basic, quanto a nível de linguagem de máquina.

Os microcomputadores, ou melhor, qualquer computador deve poder se comunicar com o exterior. Este mundo exterior inclui normalmente o leitor, a sua televisão, o seu gravador, uma impressora, microdrives etc., enfim, tudo que possa ser conectado à CPU do micro.

Dividimos os modos de comunicação de um computador, em duas categorias principais: as entradas e as saídas. As entradas são as informações recebidas pelo computador, através de qualquer periférico de entrada. As saídas, ao contrário, são as informações enviadas pelo computador para um periférico de saída. Para poder se comunicar com o exterior, o computador, qualquer que seja ele, utiliza o que chamamos de portas (*ports*, em inglês). O nome em si tem lógica.

O microprocessador pode ler e escrever dados na memória através das instruções PEEK e POKE. Mas, para ele não interessa se está lendo ou escrevendo na ROM ou na RAM; ele sabe somente que existem no máximo 65536 endereços disponíveis de uma vez, para sua pesquisa.

Numa total analogia, podemos dizer que existem 65536 portas de entrada ou saída (em inglês, *I/0 ports* - de *Input* e *Output*). Estas, como já vimos, são usadas pelo sistema para comunicação com o meio exterior, e podem ser controladas pela linguagem Basic, através das instruções IN e OUT.

A instrução IN é similar a PEEK

IN endereço

Ela possui um argumento, o endereço da porta, e seu resultado é o byte lido daquela porta.

OUT é uma instrução similar a POKE

OUT endereço, valor

que escreve naquela porta especificada, o valor determinado.

A maneira como esse endereço é interpretado depende muito do estado do computador, ou seja, se existe um periférico conectado a ele. No TK 90X é preferível imaginarmos esses endereços escritos na sua forma binária, porque cada bit do endereço trabalha de maneira independente, ou seja, depende do estado da máquina, ou da finalidade do processo.

São no total, 16 bits, assim denominados:

A15, A14, ... A10, ... A2, A1, A0

onde A0 é o primeiro bit, A1 é o segundo, e assim por diante, até A15 que é o último. Normalmente esses bits têm o valor 1, mas se um deles tem valor zero, significa que o computador deve realizar uma tarefa específica. Os bits A0, A1, A2, A3 e A4 são os mais importantes. O computador não pode fazer mais que uma tarefa por vez, portanto apenas um destes cinco bits deve ser 0, de cada vez. Os bits A6 e A7 são ignorados. Os melhores endereços para serem utilizados são aqueles múltiplos de 32, menos 1.

Os bits de A8 em diante, algumas vezes, transmitem alguma informação extra ao computador.

O byte lido ou escrito tem 8 bits, que são chamados de D0 a D7.

Existe um grupo de endereços de entrada, que tanto lêem o teclado, como o soquete EAR do micro.

O teclado é dividido em oito meias linhas de cinco teclas cada:

- IN 65278 le de CAPS SHIFT até V
- IN 65022 le de A até G
- IN 64510 le de Q até T
- IN 63486 le de 1 até 5
- IN 61438 le de 0 até 6
- IN 57342 le de P até ↵
- IN 49150 le de ENTER até H
- IN 32766 le de SPACE até B

(Estes endereços são $254 + 256 * (255 - 2^n)$, com n variando de 0 a 7).

No byte lido, os bits D0 a D4 associam-se às cinco teclas daquela meia linha – sendo D0 associado à tecla mais externa e D4 associado à tecla mais próxima ao meio do teclado. O bit será 0 se a tecla foi pressionada, e 1 em caso contrário. D6 é o valor no soquete EAR traseiro.

A porta de endereço 254, com relação a saídas, dirige o som para a TV (D4), o soquete MIC traseiro (D3) e também muda as cores da borda da tela (D2, D1, D0).

A porta de endereço 251 dirige a impressora ligada no conector traseiro, tanto em leitura, quanto em escrita, ou seja, sabendo se a impressora terminou ou não a mensagem já enviada.

As portas de endereço 254, 247 e 239 são usadas para periféricos, como microdrives, interfaces seriais etc.

A porta de endereço 63 é usada para sintetizadores de voz (nem todos – são características de cada um).

Experimente o programa:

```
10 FOR x = 0 TO 7
20 LET m = 254 + 256 * (255 - 2^X)
30 PRINT AT 0,0; IN m: GOTO 30
```

e vá pressionando as teclas para saber os valores respectivos de cada uma. Quando você completar a digitação de cada meia linha, pressione BREAK e dê, em modo direto, NEXT x, para passar para outra meia linha. Anote os valores encontrados.

NOTA: Infelizmente, nem todos os TK são iguais... Eu, particularmente, já estou no meu terceiro TK 90X, que foi o primeiro a aceitar o Microdrive, a Interface One e uma impressora Alphacon 32, todos juntos. Os outros TK que tive não reconheciam os comandos do Microdrive, assim que eu ligava a impressora neles. Particularidade muito interessante esta!

Capítulo **4**

As cores pelo teclado

Apesar de tudo o que já foi dito tanto no Manual como neste livro sobre como se obter cores para INK e PAPER, ou como entrar em modo FLASH ou BRIGHT, existe outra maneira de se colorir mensagens, gráficos criados pelo usuário, ou texto; através de códigos que são obtidos pressionando-se diretamente certas teclas.

Por exemplo, para se digitar uma linha de programa, que seja com INK 1 e PAPER 1, você deve fazer o seguinte:

1- Digite o número da linha;
2- Entre em modo EXTENDED (cursor E), pressionando CAPS SHIFT junto com SYMBOL SHIFT;
3- Pressione a tecla 1 – pronto! Você já obteve papel azul;
4- Pressione novamente CAPS SHIFT (ainda em modo EXTENDED) e outra vez a tecla 1 – e você passará a escrever com tinta azul, ou seja, com a tinta da mesma cor que o papel, para que, por exemplo, nenhum pirata tenha acesso ao seu *copyright* da primeira linha do programa.

Essa seqüência de comandos vale para qualquer tecla de 0 a 7, referente às cores.

A tecla 8 pode tanto desligar o FLASH quanto o BRIGHT, dependendo de qual tecla você pressionar, antes de pressionar a 8.

A tecla 9, ao contrário da 8, pode tanto ligar o FLASH quanto o BRIGHT.

Portanto, nessa primeira fileira de teclas, de 1 a 0, cada uma delas pode ter até 8 funções (não existe teclado mais inteligente do que este), dependendo somente do modo em que se encontra o computador naquele momento, e, principalmente, da sua prática com o teclado.

A seguir, um quadro ilustrativo dos diversos modos de se utilizar essas teclas, que por omissão, creio eu, não saiu no manual do TK 90X.

QUADRO RESUMO DAS TECLAS 1 ATÉ 0

Modo	Shift	Teclas										
E	Symbol	DEF FN	FN	LINE	OPEN	CLOSE	MOVE	ERASE	POINT	CAT	FORMAT
	Caps	INK BLUE	INK RED	INK MAGENTA	INK GREEN	INK CYAN	INK YELLOW	INK WHITE	FLASH 0	FLASH 1	INK BLACK
	Nenhum	PAPER BLUE	PAPER RED	PAPER MAGENTA	PAPER GREEN	PAPER CYAN	PAPER YELLOW	PAPER WHITE	BRIGHT 0	BRIGHT 1	PAPER BLACK
G	Caps ou Symbol									SAI DE GRAPHICS	DELETE
	Nenhum									SAI DE GRAPHICS	DELETE
K·L·C	Caps	EDIT	CAPS LOCK	TRUE VIDEO	INVERSE VIDEO					MODO GRAPHICS	DELETE
	Symbol	!	(A)	#	$	%	&	'	(	)	—
	Nenhum	1	2	3	4	5	6	7	8	9	0

Capítulo **5**

O modo gráfico

Você já deve ter notado que este micro não possui página gráfica de alta resolução.

Ótimo. Porque essa página gráfica de alta resolução é apenas uma área da memória RAM reservada estritamente para desenhos, não se misturando com textos.

No TK 90X, em Basic mesmo, você já deve ter visto que pode misturar texto e desenhos como quiser. Não tem que dar nenhuma instrução especial, para entrar e sair de modo gráfico algum. É uma grande vantagem desta máquina.

Mas, a operação de desenhar um ponto na tela possui algumas diferenças do modo de impressão de textos.

Um comando PRINT AT 0,0 imprime uma mensagem, ou um caractere na primeira linha do vídeo, ou seja, a linha de topo (a primeira superior). Você sabe que existem 22 linhas disponíveis. Para texto, começando em zero e terminando em 21. Mas se você quiser mais duas linhas (lembra-se das variáveis do sistema?), basta dar POKE 23659,0, que faz sumir a parte inferior da tela – tome cuidado, pois estas linhas da parte inferior obrigatoriamente devem existir antes de um comando INPUT, por exemplo, ou antes de ser impressa uma mensagem lá (até mesmo SCROLL?)

Já com o comando PLOT, ou outro comando de desenho, ocorre o inverso.

Um comando PLOT 0,0 desenha um ponto no canto inferior esquerdo do vídeo. Um PLOT 255,0 desenha um ponto no canto inferior direito, enquanto PLOT 175,0 desenha um ponto no canto superior esquerdo da tela e PLOT 255,175 desenha um ponto no canto superior direito da tela. Lembre-se de que a capacidade gráfica do TK 90X é de 256 pontos na horizontal (eixo X), e 176 pontos na vertical (eixo Y). Portanto, em computação gráfica neste micro, o eixo Y tem sentido inverso à maneira de formatar comandos de saída de textos.

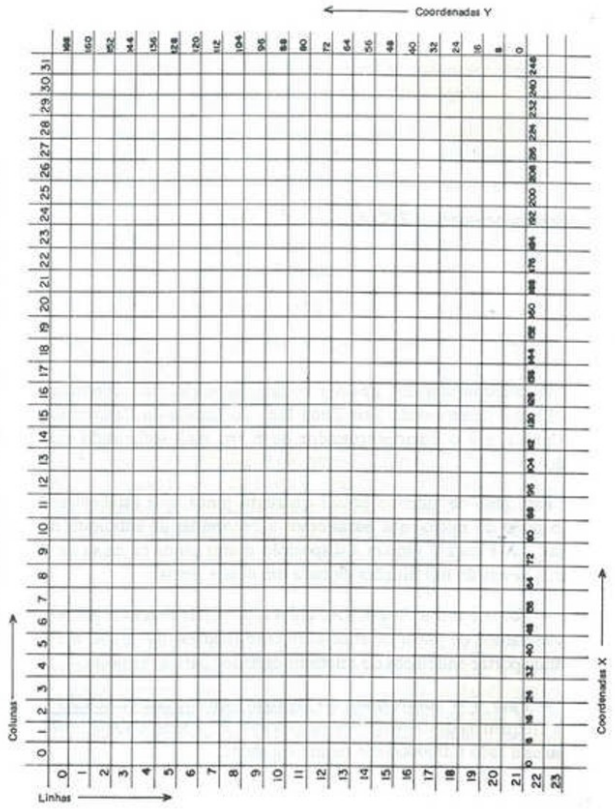
A seguir, uma tabela de orientação das posições gráficas para desenho na tela, bem como das posições de impressão de textos, que ajudará você na hora de converter gráficos em textos (coordenados), e vice-versa.

Por exemplo, você desenha, através de PLOT e DRAW, um retângulo na tela, e quer escrever alguma palavra bem no centro desse retângulo. Se os lados desse retângulo forem X e Y, teremos:

```
PRINT AT (175 - Y)/8; X/8; "..."
```

Note que as linhas (vertical) estão associadas ao eixo Y, e as colunas (horizontal) estão associadas ao eixo X. Veja também a inversão de valores na vertical.

DISPOSIÇÃO DOS ELEMENTOS DO VÍDEO DO TK 90X



Capítulo 6

O microprocessador Z80A

O microprocessador Z80A é o chip de silício mais importante do TK 90X. Ele foi desenvolvido pela Zilog Inc., do Estado da Califórnia, nos Estados Unidos, e é o microprocessador de 8 bits mais sofisticado existente no mundo.

Esse chip de silicone possui quarenta pinos, que estabelecem contato com o resto do micro, que passaremos a denominar de simplesmente "pinos ou vias". A Figura 1 mostra a disposição desses pinos na caixa do Z80, e, a seguir, a descrição das funções de cada um desses pinos:

– *Pinos 1 a 5 e 30 a 40 (A0 até A15)* Estes dezesseis pinos formam o que chamamos de pinos de endereçamento (*address bus*), que são utilizados para transportar endereços do microprocessador para a memória.

– *Pino 6 – controlador do "relógio" de entrada (CLOCK)* No TK 90X a freqüência deste relógico é de cerca de 3,75 Mhz, ou seja, um "relógio" que pulsa a cada 0,000000306 de um segundo.

– *Pinos 7 a 10 e 12 a 15 (D0 até D7)* São oito pinos que formam as vias de dados (*data bus*), que manipulam bytes de dados de e para o microprocessador.

– *Pino 11 – pino de voltagem (+ 5V)* Estabilizado em + 5 volts absolutos, requeridos pelo microprocessador.

– *Pino 16 – pino de "interrupções mascaradas" $\overline{INT}$ (INTERRUPT REQUEST)* No TK 90X a rotina de leitura do teclado, ou seja, a rotina da memória ROM que reconhece se alguma tecla foi pressionada, é considerada uma interrupção mascarada. Isto significa que o hardware da máquina contém um relógio que ativa o pino INT a cada 1/60 seg. (freqüência local), fazendo com que o microprocessador interrompa a sua atividade normal, para executar essa rotina de leitura de teclado. Essas interrupções podem ser controladas pelo programador com instruções especiais em código de máquina.

– *Pino 17 – pino de "interrupção não mascarada", NMI (NON MASKABLE INTERRUPT)* Quando este pino é ativado, ele faz com que o microprocessador pare a execução de um programa em *código* de máquina.

– *Pino 18 – pino HALT, $\overline{HALT}$* Quando este pino está no nível 0, indica que o microprocessador está executando a instrução HALT, ou seja, entra em "estado de espera", aguardando alguma instrução.

Esta instrução é usada basicamente em dois casos:

1- No final de um programa, após todas as instruções em código de máquina terem sido executadas.
2- Quando é necessário permanecer com o Z80 parado, aguardando uma instrução ou uma interrupção.

– *Pino 19 – pino de solicitação de memória, $\overline{MREQ}$* Este pino é uma saída do Z80, que, quando está no nível 0, indica que existe um endereço de memória a ser utilizado nos pinos de endereçamento, para leitura ou para escrita da/na memória. O $\overline{MREQ}$ faz parte da seleção do chip de memória, pois informa ao meio externo que o Z80 está realizando uma leitura ou escrita na memória.

– *Pino 20 – pino de entrada/saída – $\overline{IORQ}$ (INPUT/OUTPUT REQUEST)* Este sinal indica que existe, na metade inferior dos pinos de endereçamento (bits A0 até A7 – menos significativos), um endereço válido de entrada/saída (input/output), para uma operação de entrada/saída, de leitura ou escrita na memória. Um sinal $\overline{IORQ}$ também é gerado com um sinal $\overline{M1}$ quando uma interrupção é reconhecida, para indicar que uma resposta de vetor de interrupção pode ser colocada nas vias de dados.

– *Pino 21 - p de leitura, $\overline{RD}$. (memory read)* Quando este pino está no nível 0 indica que o microprocessador quer ler dados da memória ou de algum periférico de entrada/saída.

– *Pino 22 – pino de escrita, $\overline{WR}$ (memory write)* Quando está no nível 0, indica que existe nas vias de dados D0 a D7, um byte para ser armazenado no endereço da memória ou no periférico de entrada/saída.

– *Pino 23 – pino de reconhecimento – $\overline{BUSAK}$ (bus acknowledge)* O microprocessador reconhece uma "requisição externa", interrompendo a execução de qualquer instrução e ativando este pino.

– *Pino 24 – pino de espera – $\overline{WAIT}$* Este é um sinal de pedido de espera, com o objetivo de sincronizar memórias e periféricos de entrada/saída mais lentos que o Z80. Enquanto este sinal de entrada WAIT for mantido no nível 0, o microprocessador fica parado aguardando que o meio externo responda à sua solicitação de leitura ou escrita.

– *Pino 25 – pino de solicitação – $\overline{BUSRQ}$ (bus request)* O Z80 permite que periféricos externos usem os pinos de endereçamento ou os pinos de dados, através da ativação deste pino.

– *Pino 26 – pino de inicialização – $\overline{RESET}$* Este pino é uma entrada usada para inicializar o Z80. Ele é acionado imediatamente após ligar o TK 90X, ou quando se deseja inicializar o Z80. Quando o $\overline{RESET}$ vai para o nível 0, ocorre o seguinte:

1- os pinos de endereçamento e de dados são inicializados;
2- todos os sinais de controle ficam inativos;
3- os registros R e I ficam com valor 0;
4- o modo de interrupção é colocado em 0;
5- as interrupções provenientes da entrada INT são inibidas; e
6- o contador de programa é zerado.

– *Pino 27 – pino de "busca" da memória – $\overline{M1}$ (machine cycle one)* Quando vai para o nível 0, indica que está sendo executado um *fetch* (ciclo de busca de instrução), da instrução corrente. Toda instrução, ao ser executada, exige que o Z80 primeiramente realize a busca do seu código de operação (OP CODE) que está armazenado na memória. Em seguida, o Z80 deposita este código no registro de instrução, para, então, interpretá-lo.

– *Pino 28 – pino de "restauração" da memória – (refresh) $\overline{RFSH}$* Não está diretamente relacionado com leitura ou escrita na memória. É utilizado em memórias RAM dinâmicas, como um seu refrescamento (restauração).

Basicamente, o RFSH é uma operação de leitura em determinadas posições de memória, sem que haja efetivamente transferência de informações.

O Z80, através das vias de endereçamento (*address bus*), do registro R e do sinal $\overline{RFSH}$, implementa as funções de RFSH, sem necessidade de controladores externos.

Quando $\overline{RFSH}$ = 0 e $\overline{MREQ}$ = 0, o conteúdo do registro R é colocado nos 7 bits menos significativos (A0 até A6) das vias de endereçamento, e a cada busca de instrução (*fetch*), o conteúdo do registro R é incrementado de uma unidade.

– *Pino 29 – pino terra.*

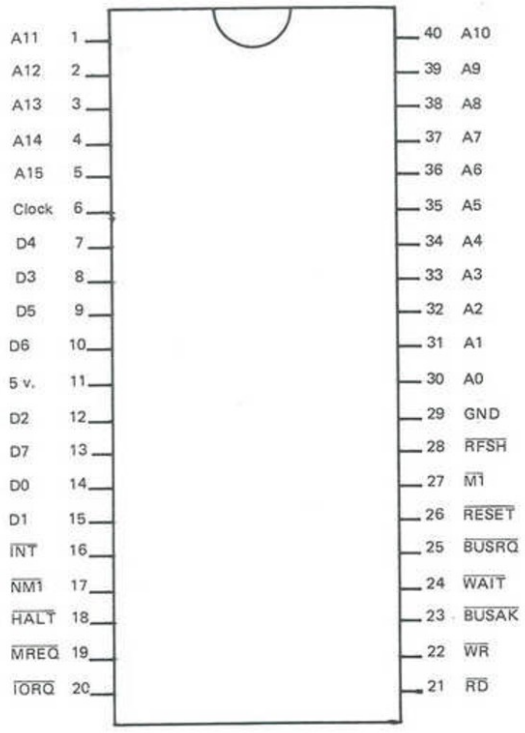


Figura 1 – *Pinagem do microprocessador Z80*

A ESTRUTURA INTERNA DO Z80

A estrutura interna desse chip é um tanto complicada, mas felizmente dividida em cinco partes que exercem funções diferentes. São elas:

1- Unidade de controle
2- Registro de instruções
3- Contador de programa
4- Registros disponíveis ao usuário
5- Unidade lógica e aritmética

A figura abaixo mostra esquematicamente a disposição dessas partes dentro do chip.

Partes funcionais do chip Z80

A seguir, a descrição dessas cinco partes:

UNIDADE DE CONTROLE

A unidade de controle do Z80 pode ser comparada a um "gerente de uma linha de produção de uma fábrica". É responsabilidade da unidade de controle a aquisição de matérias-primas (bytes de dados), que são transformadas pela fábrica (estrutura do Z80) em produtos finais acabados (também bytes de dados), que são enviados aos destinatários finais, garantindo assim sucesso na produção.

Essa unidade de controle produz um número muito grande de sinais de controle internos, que, através das vias de controle, vão para outras partes da estrutura interna do microprocessador, assim como esses sinais de controle vão para os pinos de controle $\overline{RD}$, $\overline{WR}$, $\overline{MREQ}$ etc.

Note, porém, que, da mesma maneira como um gerente de uma linha de produção, esta unidade de controle não é responsável sobre qual tarefa deva ser realizada, mas apenas sobre como fazer.

O Z80 tem condição de funcionar como computador porque tem a habilidade de seguir um programa armazenado. Este programa deve estar presente em algum lugar da memória, de modo que ele possa ler as instruções em código de máquina, uma a uma, para então executá-las.

REGISTRO DE INSTRUÇÕES

O termo "registro" é usado para descrever um dispositivo interno do Z80 que guarda temporariamente 8 bits de um byte qualquer, para poder manipulá-los.

Nos circuitos internos deste microprocessador existem vários registros, e o movimento de bytes entre esses registros é um dos recursos mais importantes em programação em linguagem de máquina.

O registro de instruções é um registro muito especial, que armazena o código de operação de uma instrução durante todo o tempo em que esta esteja sendo executada. Esse código de operação de instrução é quem determina o que deve ser feito pelo sistema durante uma instrução.

CONTADOR DE PROGRAMA

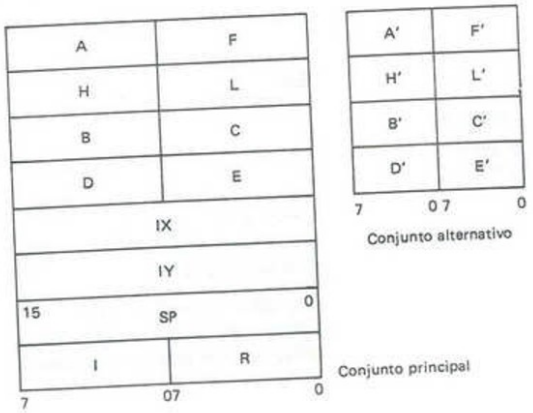
Este contador de programa não é um registro simples de 8 bits, mas sim a junção de dois registros de 8 bits, formando um par de registros, totalizando 16 bits, para serem usados juntos, no sentido de manter controle de endereçamentos de cada instrução guardada na memória. Sempre que uma instrução for lida na memória, a fim de ser executada, junto com ela deve ser fornecido um endereço.

O PC (Program Counter) terá então o endereçamento dessa instrução, sendo normalmente incrementado para, após tê-la executado, apontar para a instrução seguinte.

REGISTROS DISPONÍVEIS AO USUÁRIO (REGISTROS PRINCIPAIS)

Existem, no total, 24 registros disponíveis ao usuário no microprocessador Z80. Eles são assim denominados porque podem receber e armazenar bytes de dados especificados pelo usuário. Cada registro tem seu próprio nome, que não possui lógica na maioria dos casos, e alguns deles possuem funções específicas. São registros de 1 byte, ou seja, 8 bits (daí o microprocessador ser de 8 bits – manipular 1 byte por vez), que em determinadas situações são agrupados aos pares, formando um "par de registros", capaz de manipular, então, 16 bits.

O esquema abaixo mostra os registros do Z80.



A seguir a descrição dos registros disponíveis no Z80, com as suas respectivas funções:

REGISTRO A

Este é o mais importante registro do Z80. Ele é mais conhecido por "acumulador", porque na maioria das operações que dele se utilizam, usam-no para acumular seus resultados.

Ele é muito usado para desenvolver operações aritméticas e lógicas, e muitas delas se utilizam apenas deste registro para atingir o resultado.

Por conseguinte, existem diversos modos pelos quais um byte de dados pode ser armazenado pelo programador neste registro. Portanto existem muitas instruções em código de máquina que envolvem o registro A.

REGISTRO F

Também conhecido como *flag register*, ou seja, em tradução literal, "registros de bandeiras", cujo significado é "registro de indicadores de estado".

Normalmente ele é mais conhecido como uma coleção de 8 bits indicadores de algum estado específico de microprocessador, em vez de um registro propriamente dito.

O conceito destes bits indicadores de estado será visto com mais detalhes mais tarde, mas por enquanto basta saber que determinados indicadores querem dizer algo, quando estão com valor 1, e querem dizer outra coisa completamente diferente quando estão com valor 0.

REGISTROS H E L, FORMANDO O PAR HL

Normalmente, as instruções que endereçam bytes de dados na memória, o fazem através do par de registros HL, tornando então possível o endereçamento de até 65.536 posições de memória. Lembremos que um endereço de memória sempre se utiliza de 16 bits (2 bytes), e é subdividido em "parte alta" e "parte baixa", dando origem aos nomes desses registros H (de HIGH) e L (de LOW), significando que a parte alta do endereço será armazenada no registro H e a parte baixa no registro L. Por exemplo, o endereço 7682h, para ser armazenado no par HL, é subdividido em 76, que vai para o registro H, e 82, que vai para o outro registro.

Uma memória de 65.536 posições de endereçamento pode ser considerada como sendo dividida em 256 páginas de 256 posições, e, neste caso, o valor armazenado no registro H serve como indicador de qual página da memória está sendo utilizada.

No microprocessador Z80, o par de registros HL é um dos três pares empregados no endereçamento de registros, porém é o mais importante. Ele pode ser usado para armazenar um número de 16 bits em vez de endereços, pois existe um grande número de operações aritméticas que podem ser realizadas com esses números.

REGISTROS B, C, D, E, OU PARES BC E DE

Estes pares de registros são usados principalmente como registros de endereçamento, para auxiliar o par HL, ou utilizados individualmente, como registros comuns, embora o nome DE seja abreviação de "DESTINATION" (destino), e o registro B, quando utilizado individualmente serve como contador de loop, na maioria das vezes.

CONJUNTO DE REGISTROS ALTERNATIVOS

O conjunto de registros existentes, A, B, C, D, E, F, H e L podem não ser suficientes para o programador em código de máquina. Para isso existem os registros alternativos, A', B', C', D', E', F', H' e L', que, através de instruções em código de máquina especiais, permitem que o conteúdo dos registros principais seja momentaneamente trocado com o conteúdo do seu equivalente alternativo.

Assim, os dados anteriores ficam guardados nos registros alternativos, enquanto se trabalha com os registros principais. A cada troca, todos os registros são envolvidos, com exceção dos registros A e F.

PARES DE REGISTROS IX E IY

Estes dois pares de registros são usados em operações que envolvem indexação, que é uma facilidade que permite manipulação de itens de listagens ou tabelas, a fim de serem pesquisadas. Estes registros mantêm um endereço base, e as posições desejadas são sempre em função desse endereço base, que deve estar obrigatoriamente armazenado no par IX ou no par IY.

No programa monitor de 16 K do TK 90X, o par de registros IY é normalmente utilizado para apontar o endereço 23.610, ou 5C3Ah, endereço este que é considerado como base da tabela de variáveis do sistema.

O par IX é utilizado como apontador das rotinas dos comandos Basic, LOAD, SAVE, VERIFY e MERGE.

Por isso, não é conveniente mexer no par IY, e muito cuidado ao utilizar o par IX.

REGISTRO APONTADOR DA PILHA SP (STACK POINTER)

Este registro ainda é um registro de endereçamento. Ele é usado para apontar posições na área da pilha da máquina na memória RAM, e é sempre considerado como sendo um registro simples, porém de 2 bytes.

Essa pilha da máquina (*machine stack*) é utilizada para se "empilhar" endereços, da seguinte maneira: último a ser colocado, primeiro a ser retirado (LIFO = Last In, First Out), e o apontador da pilha (SP) é usado para armazenar o endereço da última posição a ser executada. Entretanto, quando uma nova entrada está para ser efetivada, a Unidade de Controle do microprocessador reduz o valor armazenado no ponteiro da pilha antes de aceitar essa entrada.

REGISTRO I

Este é o registro vetor de interrupção. Em outros sistemas baseados no Z80, que não o TK 90X, este registro normalmente é usado para armazenar o endereço base de uma tabela de endereços para manipulação de diferentes dispositivos de entrada/saída. Entretanto, esta facilidade não é utilizada no TK 90X, e este registro é usado para auxiliar na geração de sinais para a televisão.

REGISTRO R

Este é o registro de refrescamento de memórias. Na realidade, ele é um simples contador que é incrementado a cada ciclo de busca de instrução. O valor no registro se altera diversas vezes entre valores de 0 a 255.

O registro R é usado para gerar parte do endereço requerido para memórias dinâmicas, de forma que possa ser "refrescado" (recarregado).

A UNIDADE LÓGICA E ARITMÉTICA (ULA)

Este é o quinto bloco funcional do microprocessador Z80 tendo como função específica o procedimento de operações lógicas e aritméticas, de propósitos bem reduzidos; em aritmética apenas operações de soma e subtração binárias são possíveis, e sempre de um byte contra outro byte.

O usuário deverá programar rotinas em linguagem de máquinas repetitivas, caso deseje trabalhar com campos maiores que um byte. Por esse motivo, após cada soma ou subtração de um byte, um indicador do registro F, aquele chamado de CARRY FLAG, ou bandeira de transporte, poderá ser ativado (conter valor 1), simulando aquela nossa conhecida operação de "vai um", na operação aritmética dos próximos dois bytes.

Esta unidade também é capaz de desenvolver operações com bits, operações lógicas (AND, NOT, OR ou XOR – que veremos adiante) e operações de ativar ou desativar indicadores de estados (*flags*), a fim de mostrar determinados resultados.

Capítulo **7**

Estrutura de um programa em código de máquina

Conforme já vimos, o microprocessador Z80 trabalha como um computador, já que é uma pequena máquina capaz de seguir instruções de um programa armazenado. Este programa obrigatoriamente sempre é um conjunto de instruções em código de máquina, associado a alguns dados, ou bytes, ou posições da memória; tanto ROM como RAM, ou informações genéricas que o programa necessita, sempre armazenadas em posições consecutivas na memória.

Em microcomputadores baseados no microprocessador Z80, estas posições da memória armazenam no máximo 8 bits, ou 1 byte de dado. Portanto, um programa em código de máquina consiste em um conjunto de dados que aparecem como uma série de números de 8 bits.

A descrição mais elementar de um programa em código de máquina é a sua representação binária. Por exemplo:

Decimal	*Binário*
Posição 0	11110011
Posição 1	10101111
Posição 2	00010001
Posição 3	11111111
Posição 4	11111111
Posição 5	11000011
Posição 6	11001011
Posição 7	00010001

Essa representação é uma maneira perfeitamente válida para mostrar um programa em código de máquina. Mas você há de convir que é dificílimo para nós programarmos assim, e é também muito sujeito a erros. (Imagine, dada essa estrutura de programa, se você trocar apenas um bit de um byte qualquer...)

A seguir, o mesmo programa em código de máquina, só que com outra representação, expressa em decimal e em hexadecimal. Mas, que é igualmente difícil e não muito útil, dado que não sabemos o seu significado.

Decimal	*Hex*	*Decimal*	*Hex*
Posição 0	0000	243	F3
Posição 1	0001	175	AF
Posição 2	0002	17	11
Posição 3	0003	255	FF
Posição 4	0004	255	FF
Posição 5	0005	195	C3
Posição 6	0006	203	CB
Posição 7	0007	17	11

Experimente digitar o programa Basic a seguir, que mostra em valores hexadecimal, os conteúdos das sucessivas posições da ROM.

E o resultado, para as 21 primeiras posições:

Altere a linha 40 do programa, de forma que também seja mostrado o equivalente em decimal dos bytes da ROM.

Este programa foi incluído aqui para mostrar como usualmente se produz uma listagem com valores decimais ou hexadecimais de um programa em código de máquina.

Mas, o que querem dizer esses valores? Aparentemente não têm significado nenhum. Vamos ver isso:

Na tabela do final deste livro, que mostra todas as instruções do Z80, com seus respectivos códigos, ou mesmo no manual do TK 90X, você pode ver o que significam esses códigos:

Decimal	*Mnemônica*	*Comentário*
Posição 0	DI	Não permite interrupções
Posição 1	XOR A	Efetua a operação lógica XOR A com o acumulador (registro A).
Posição 2 - 4	LD DE, FFFF	Carrega o par de registros DE com o valor constante FFFFh.
Posição 5 - 7	JP 11CB	Efetua um salto relativo para o endereço CB11

Nessa descrição, "mnemônica" quer dizer o nome das instruções em código de máquina, que é uma maneira estilizada de representar a instrução, de forma que seja facilmente compreendida.

Todas as instruções em código de máquina do conjunto de instruções do Z80 possuem suas próprias mnemônicas, e normalmente um programa em código de máquina é descrito utilizando-se delas em vez de valores binários, decimais ou hexadecimais.

Note na listagem acima, que duas linhas de instrução usam posições simples enquanto outras duas linhas ocupam três posições. No último caso, a primeira posição armazena o código da instrução propriamente dito, enquanto as outras duas armazenam os dados associados a essas instruções.

A forma usual de se listar um programa em código de máquina é mostrada a seguir:

Endereço	*Código de Máquina*	*Mnemônica*	*Comentários*
0000	F3	DI	Não permite interrupções
0001	AF	XOR A	Operação lógica XOR ou OR exclusivo
0002	11 FF FF	LD DE, FFFF	Endereço de topo da memória, armazenado no par DE
0005	C3 CB 11	JP 11CB	Equivalente ao Basic "GOTO"

Esta forma de listagem normalmente é chamada de formato *assembly* (montado), e possui os endereços das posições, armazenando o primeiro byte da linha da instrução, em hexadecimal; os códigos das instruções e seus dados associados, também em notação hexadecimal; as mnemônicas de cada instrução, e, finalmente, um campo para comentários, onde o usuário pode escrever o que deseja que aconteça naquele momento.

O exemplo acima mostra como o formato *assembly* para um dado bloco de códigos de Z80 pode ser derivado – uma operação que normalmente é denominada *disassembly*, e o programa de computador que desenvolve essas operações é conhecido por *disassembler*. Portanto, programas *disassembler* são muito úteis, que mostram programas em código de máquina, com as posições da memória ou endereços, em hexadecimal, seus conteúdos, também nessa base numérica, e as mnemônicas das instruções correspondentes.

Em outras palavras, um programa *assen bler*, ou *montador*, permite-nos desenvolver listagens em código de máquina a partir de suas mnemônicas, para que ele faça a sua conversão para seus códigos equivalentes automaticamente, e os programas chamados de *disassembler*, ou *desmontadores*, permitem que *abramos* listagens em código de máquina, ou seja, mostram-nos listagens de instruções do Z80, seus endereços e respectivos códigos hexadecimais com as mnemônicas.

Às vezes, alguns programadores em código de máquina incluem rótulos (*labels*) em determinadas posições ou endereços da memória, que são constantemente utilizadas durante toda a execução de uma determinada rotina, para facilitar o trabalho de localizar aquela posição.

Escrevendo a rotina acima dessa forma:

INÍCIO = 0000	Endereço/posição 0
NOVO INÍCIO = 11CB	Continua aqui
TOPO MEM = FFFF	Decimal 65.535 – topo da memória

Endereço	*Rótulo*	*Mnemônica*	*Comentário*
0000	INÍCIO	DI	Não permite interrupções
0001		XOR A	Operação XOR A
0002		LD DE, TOPO MEM	Carrega par DE com o topo da memória
0005		JP NOVO INÍCIO	Salta para aquela posição

A seguir, alguns exemplos de listagens, que, apesar de serem completamente diferentes, querem dizer a mesma coisa.

```
>Disassemble 0
0000 F3          di
0001 AF          xor a
0002 11FFFF      ld de,FFFF
0005 C3CB11      jp 11CB
0008 2A5D5C      ld hl,(5C5D)
000B 225F5C      ld (5C5F),hl
000E 1843        jr 0053
0010 C3F215      jp 15F2
0013 FF          rst 38
0014 FF          rst 38
0015 FF          rst 38
0016 FF          rst 38
0017 FF          rst 38
0018 2A5D5C      ld hl,(5C5D)
```

Este é o método mais comum, descrito anteriormente. A primeira coluna da esquerda mostra os endereços, em valores hexadecimais, seguída dos códigos hexa, e as mnemônicas correspondentes.

Outra listagem, mais simples:

```
>Tabulate 0

0000  F3 AF 11 FF FF C3 CB 11
0008  2A 5D 5C 22 5F 5C 18 43
0010  C3 F2 15 FF FF FF FF FF
0018  2A 5D 5C 7E CD 7D 00 C0
0020  CD 74 00 18 F7 FF FF FF
0028  C3 5B 33 FF FF FF FF FF
0030  C5 2A 61 5C E5 C3 9E 16
0038  F5 E5 2A 78 5C 23 22 78
```

Aqui também, a primeira coluna da esquerda significa endereços, em hexadecimal. Compare os códigos da primeira linha com a listagem anterior.

Outro tipo de listagem, abaixo, que mostra, quando possível, os caracteres correspondentes àqueles valores dos códigos hexadecimais.

```
>Print 0

0000
0008  #  J  \  "  _  \     ©
0010
0018  #  J  \  "     >
0020     \
0028     [  ©
0030     #  £  \           ©
0038        #  ×  \  ¤  "  ×
```

E aqui, uma curiosidade: o conteúdo dos registros do Z80 e o estado das bandeiras, ou indicadores, que veremos em detalhe mais adiante.

```
READY
>Registers

  BC    DE    HL    IX    IY    A
0000  F3B2  5E1E  03D4  5C3A  85

  BC'   DE'   HL'   SP   (SP)  (HL)
0001  0000  595B  BF4D  62AC   30

BREAKPOINT :  -

FLAGS = SZ-H-PNC
        01100001


READY
>Exit
```

Um programa para mostrar bytes de uma forma "tabulada":

```
  10 LET a=65000  REM   coloque
aqui o endereço primeiro endereço
  15 PRINT a;
  20 FOR g=0 TO 31 STEP 3
  25 LET h=INT (PEEK a/16): LET
l=PEEK a-h*16
  30 LET a$=CHR$ (48+h+7*(h>9)):
LET b$=CHR$ (48+l+7*(l>9)): REM
estas duas variaveis alfanumeri
cas fazem a conversao do sistema
decimal (dos bytes) para o siste
ma hexadecimal
  30 PRINT TAB g;a$;b$;" ";
  35 LET a=a+1: NEXT g
  36 GO TO 15
```

Capítulo **8**

A matemática na programação em código de máquina

Já vimos que num microcomputador baseado no microprocessador Z80, como o TK 90X, toda e qualquer manipulação de dados em sua memória envolve um byte de 8 bits. A melhor forma de se representar esse dado é a que utiliza a notação binária para cada bit envolvido na operação. Mas, nesse formato, os números são muito difíceis de serem manipulados, e, por esta razão, entre outras de menor importância, é que se usa a notação hexadecimal.

Neste capítulo veremos, por seções, os códigos hexadecimais, aritmética binária absoluta, complemento de dois aritmético, representação integral de um número e representação de um número em ponto flutuante. Os três primeiros tópicos se aplicam em qualquer microcomputador de 8 bits, enquanto os dois últimos são um pouco diferentes no TK 90X.

VALORES OU CÓDIGOS HEXADECIMAIS

O sistema hexadecimal, da mesma forma que um sistema decimal, ou outro sistema qualquer, deve ter uma base para representar valores numéricos. Conforme o próprio nome diz, a base desse sistema é dezesseis, enquanto no sistema decimal, com o qual estamos acostumados a lidar, emprega-se base dez, e no sistema binário emprega-se base dois.

No nosso sistema decimal, possuímos 10 valores distintos para representar quantidades diferentes. Analogamente, o sistema binário utiliza-se de dois valores, quais sejam, o zero e o um, e o sistema hexadecimal possui nada me-

nos que dezesseis valores para representação numérica. Nesta, os primeiros dez caracteres são os dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 e os seis caracteres adicionais são as letras A, B, C, D, E e F.

Mas, o que representa essa base decimal? Significa que todo e qualquer valor numérico pode ser representado através dos caracteres que compõem aquela base.

Por exemplo: na base decimal, o que significa 1985? Vamos decompor essa quantidade numérica:

```
1985
   5 x 10^0 = 5 x 1 = 5
   8 x 10^1 = 8 x 10 = 80
   9 x 10^2 = 9 x 100 = 900
   1 x 10^3 = 1 x 1000 = 1000
                         1985
```

Ou, na base binária:

```
111011110011
   1 x 2^0 = 1 x     1 = 1
   1 x 2^1 = 1 x     2 = 2
   0 x 2^2 = 0 x     4 = 0
   0 x 2^3 = 0 x     8 = 0
   1 x 2^4 = 1 x    16 = 16
   1 x 2^5 = 1 x    32 = 32
   1 x 2^6 = 1 x    64 = 64
   1 x 2^7 = 1 x   128 = 128
   0 x 2^8 = 0 x   256 = 0
   1 x 2^9 = 1 x   512 = 512
   1 x 2^10 = 1 x 1024 = 1024
   1 x 2^11 = 1 x 2048 = 2048
                        3827
```

E na base hexadecimal:

```
FA8B
   B x 16^0 = 11 x    1 = 11
   8 x 16^1 =  8 x   16 = 128
   A x 16^2 = 10 x  256 = 2560
   F x 16^3 = 15 x 4096 = 61440
```


Aí estão, portanto, dois exemplos de conversão, respectivamente de base binária e hexadecimal para decimal.

A tabela abaixo mostra a representação decimal, binária e hexadecimal para números de 0 a 15.

Decimal	*Binário*	*Hexadecimal*
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Como você pôde notar na tabela anterior, cada caractere hexadecimal forma uma representação binária de 4 bits. Isso significa que um byte de 8 bits é representado por um par de caracteres hexadecimais, e um número de 16 bits necessita de quatro caracteres hexadecimais.

Por exemplo:

```
0000  0000b = 00h
0100  1111b = 4Fh
0000  0000  0000  0000b = 0000h
0100  1100  1010  1111b = 4CAFh
```

O programa abaixo faz a conversão automática de números na base decimal para números na base hexadecimal:

```
  1 REM Base decimal para hexad
ecimal
  5 BORDER 0: PAPER 0: INK 6: C
LS
  10 INPUT "Digite um numero men
or que 65535";d
  20 IF d>65535 THEN GO TO 10
```

```
  30 PRINT d;"   Dec.  ";
  40 DIM h(4): DIM h$(4)
  50 LET h(1)=INT (d/4096): LET
d=d-h(1)*4096
  60 LET h(2)=INT (d/256): LET d
=d-h(2)*256
  70 LET h(3)=INT (d/16): LET d=
d-h(3)*16
  80 LET h(4)=d
  90 FOR a=1 TO 4
 100 LET h$(a)=CHR$ (h(a)+48+7*(
h(a)>9))
 110 NEXT a
 120 PRINT INK 5; BRIGHT 1;h$; I
NK 6; BRIGHT 0;"  hexa  "
 130 GO TO 10
```

Repare nas linhas 50 até 80, as divisões sucessivas por, respectivamente, 16^3, 16^2 e 16^1, e o aproveitamento do resto dessas divisões.

Nas linhas 90 a 110 são feitas as conversões para caracteres ASCII.

Programa que converte números hexadecimais em decimais

```
   1 REM Base hexadecimal para d
ecimal
   2 POKE 23658,8
   5 BORDER 0: PAPER 0: INK 6: C
LS
  10 DIM h$(4)
  20 INPUT "Digite valor hexadec
imal ";h$
  30 IF CODE h$=32 THEN GO TO 20
  40 LET d=0
  50 FOR a=1 TO 4
  60 IF h$(a)=CHR$ 32 THEN GO TO
 100
  70 IF h$(a)<"0" OR h$(a)>"9" A
ND h$(a)<"A" OR h$(a)>"F" THEN G
O TO 120
  80 LET d=d+16^(4-a)*(CODE h$(a
)-48-7*(CODE h$(a)>57))
  90 NEXT a
 100 PRINT "Hexadecimal ";TAB 15
h$; ;
 110 PRINT "Decimal ";TAB 15;d
 120 GO TO 10
```

Portanto, para se converter um número na base decimal, para a base hexadecimal, divide-se esse número pelo valor correspondente à potência de 16,

cujo expoente corresponde àquela posição do bit, e converte-se o resultado para o equivalente hexadecimal, até o final. Por exemplo, para se converter 65535d em hexa:

1- 65535 ÷ 16 = 4095,9375
2- 65535 | 4096
24575 15
4095
Resultado = 15 decimal ou F hexa

3- 15 x 4096 = 61440
4- 65535 – 61440 = 4095
5- 4095 ÷ 16 = 255,9375
6- 4095 | 256
1535 15
255
Resultado = 15 decimal ou F hexa

7- 15 x 256 = 3840
8- 4095 – 3840 = 255
9- 255 ÷ 16 = 15,9375
10- 255 | 16
95 15
15
Resultado = 15 decimal ou F hexa

11- Resto final = 15 decimal ou F hexa
Portanto o número 65535 d corresponde à FFFF h.

De posse de uma calculadora, podemos simplificar o processo. Por exemplo, vamos converter 40000d para hexa:

```
40000 | 4096
36384   9,76 ---------- equivale a 9 hexa
 3136 | 256
 3072   12,25 ---------- equivale a C hexa
   64 | 16
    0   4   ---------- equivale a 4 hexa
          Resto final ---------- 0
```

Portanto, 40000d = 9C40h

Se você pretende se aprofundar nos estudos da linguagem de máquina, aconselho-o a se exercitar, e muito, nessas conversões, principalmente entre base decimal e base hexadecimal.

ARITMÉTICA BINÁRIA

Agora vamos ver as regras para operações aritméticas binárias.

I – ADIÇÃO

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 com transporte de 1
1 + 1 + 1 = com transporte de 1

Por exemplo:

```
  8 ..... 1000 +
+ 7 ..... 0111
 15 ..... 1111
```

Se a soma exceder o valor decimal 15, o resultado obrigatoriamente será expresso em cinco bits.

Por exemplo:

```
  9 ..... 1001 +
+ 8 ..... 1000
 17 ..... 10001
```

Se você pretende se aprofundar em código de máquina, meus parabéns! Mas, antes assimile perfeitamente essas regras iniciais de operações aritméticas básicas, porque, sem elas, não é possível se fazer nada, visto que teremos instruções em linguagem de máquina que manipulam apenas um bit de um byte. E se não soubermos o resultado...

II – SUBTRAÇÃO

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 com transporte de -1
1 - 1 - 1 = 1 com transporte de -1

Por exemplo:

```
 14 ..... 1110 -
 -3 ..... 0011
 11 ..... 1011
```


Numa subtração, o resultado tanto pode ser positivo como negativo. A fim de distinguir o sinal de um número, usa-se o bit mais à esquerda, qual seja o bit mais significativo, como bit de sinal. Normalmente usa-se a convenção que se atribui o valor 1 ao sinal negativo e 0 ao sinal positivo. Conseqüentemente, o microprocessador Z80 que possui 8 bits de dados, utiliza 7 deles para armazenar o dado propriamente dito e o último para indicar o seu sinal. Muito cuidado deve ser tomado na expressão desses valores.

São necessários complexos circuitos eletrônicos para se subtrair diretamente os valores binários, e nos microcomputadores é usual realizar estas subtrações somando-se o "complemento de 2" do diminuidor ao diminuendo. Isto significa que se torna possível dispensar o circuito que executa a subtração, realizando essa operação através de circuitos de soma, inclusive quando se determina a forma complementar do diminuidor.

O complemento de 2 de um número binário é determinado invertendo-se cada um dos seus bits e somando-se 1 ao resultado. Por exemplo, o complemento de 2 de 76 em representação binária é determinado da seguinte maneira:

76 = 01001100

Invertendo-se os bits:

```
 10110011 +
        1
 --------
 10110100 = -76 em forma de complemento de 2
```

Para se realizar uma subtração usando-se a complementação de 2 é necessário obter o complemento de 2 do diminuidor, que é em seguida somado ao diminuendo. Como exemplo, consideremos 44 subtraído de 15.

15 = 00001111 e 44 = 00101100
0 complemento de 2 de 44 é -44 = 11010011

Então:

```
 00001111
 11010011 +
 --------
 11100011 = -29, na forma de complemento de 2, isto é, trata-se do
```

complemento de 2 de +29. Nesse caso, o bit de sinal indica um número positivo que se encontra na forma complementar de 2.

III – MULTIPLICAÇÃO

Um método muito comum de realizar a multiplicação binária utiliza o princípio do deslocamento e soma dos algarismos. O multiplicando é multiplicado por cada bit do multiplicador, bit a bit, e o produto resultante obtém-se somando todos os produtos parciais convenientemente deslocados. Como os bits do multiplicador são 0 ou 1, os termos do produto parcial são iguais a 0 ou ao multiplicando, ou versões deslocadas deste.

Por exemplo, vamos multiplicar 193 por 21. Utilizemos para tanto um acumulador de 16 bits para guardar o resultado, e trabalhemos com representações binárias de 8 bits para o multiplicando e o multiplicador.

```
Multiplicando ..... 193 = 11000001
Multiplicador ..... 21 = 00010101
                     11000001 . . produtos parciais
                     11000001 . . convenientemente
                     11000001 . . deslocados
                   111111010101 . . soma dos termos
```

Quando se multiplicam dois números binários, o produto contém um número de bits igual à soma dos bits contidos nos dois valores envolvidos na operação. O número máximo de somar requerido para a multiplicação, usando o método de deslocamento e soma, é igual ao número de bits do multiplicador.

IV – DIVISÃO

A divisão binária é realizada usando-se um método de deslocamento e subtração, ao contrário da multiplicação que vimos acima. Diminui-se repetidamente o divisor do dividendo, depois deste ser convenientemente deslocado, e verifica-se o sinal do resto após cada subtração. Se o sinal do resto é positivo, o valor do quociente é 1, mas se o sinal é negativo, o valor é zero, e o dividendo é reconduzido ao seu valor anterior, somando-se de novo o divisor. Depois de a subtração dar um quociente positivo, ou, depois do tratamento anterior, no caso de ter dado um quociente negativo, o divisor é deslocado de uma posição, para a direita, sendo incluído o bit seguinte e repetindo-se a operação até todos os bits do divisor terem sido usados.

COMPLEMENTO DE DOIS ARITMÉTICO

Como vimos na subtração binária, o conceito de complemento de 2 aritmético é muito simples, mas, quando usado em programas em código de máquina, o resultado pode ser um pouco confuso.

O método permite que o programador considere valores na faixa binária entre 0000 0000 até 1111 1111, equivalentes a 0 até 127 do sistema decimal, e 000 0000 até 1111 1111 como equivalente à faixa decimal de -128 a -1.

O resultado dessa interpretação depende do bit 7 (o mais à esquerda na representação binária), atuando como bit de sinal. Ele valerá 0, em caso de números positivos, e 1, no caso de números negativos.

A tabela a seguir ilustra a aplicação do complemento de 2 aplicado a números de 8 bits.

	Binário	*Decimal*	*Hexadecimal*
Números positivos	0111 1111	+127	7F
	0111 1110	+126	7E
	0000 0010	+2	02
	0000 0001	+1	01
	0000 0000	0	00
Números negativos	1111 1111	-1	FF
	1111 1110	-2	FE
	1000 0001	-127	81
	1000 0000	-128	80

Note que é possível estender o conceito de complemento de dois aritmético a números que utilizam 16 bits na sua representação binária, atingindo a faixa decimal de -32.768 até +32.767.

Para se converter um número decimal negativo ao seu respectivo complemento de dois binário, realizam-se os seguintes passos:

1- Encontra-se a forma binária do valor absoluto decimal.
Por exemplo: -54 equivale a 00110110

2- Inverte-se o número binário, isto é, todos os zeros passam a valer 1, e todos os 1 passam a valer 0. Portanto, 00110110 passa a 11001001

3- Soma-se 1 a esse valor em aritmética binária

```
 11001001
        1 +
 --------
 11001010
```

4- Esta é a forma utilizada, tanto em binário, quanto em hexadecimal. Portanto, o complemento de dois de -54 é 11001010 ou CAh.

Essas operações podem ser realizadas em ordem inversa, quando convertendo-se números complemento de dois para números decimais.

REPRESENTAÇÃO INTEGRAL DE UM NÚMERO

O interpretador Basic do TK 90X utiliza cinco bytes para representar números inteiros entre -65535 e +65535, ao passo que números fracionários ou excedentes a essa faixa são tratados em cinco bytes, mas na forma de ponto flutuante.

Na forma integral, o primeiro byte é sempre zero. O segundo byte armazena zero se o inteiro é positivo e 255 se o inteiro é negativo. Os terceiro e quarto bytes armazenam o valor do número, na forma de complemento de dois de 16 bits, sem sinal, sendo o terceiro byte o menos significativo, e o quarto, o mais significativo. O quinto byte também não é usado e vale sempre zero.

O programa a seguir mostra como são manipulados os valores digitados pelo usuário, em decimal, na sua forma integral. Note as condições para somente números inteiros e contidos no limite dessa faixa, especificados na linha 20 do programa.

```
10 INPUT n
20 IF n <> INT n OR n < -65535 OR n > 65535 THEN GOTO 10
30 PRINT "Numero escolhido"; n
40 LET v = PEEK 23627 + 256 * PEEK 23628
50 FOR a = 1 TO 5
60 PRINT a; " . "; TAB 5; PEEK (a + v)
70 NEXT a
80 GOTO 10
```

O programa fornece resultados na forma:

```
- Número escolhido: 0
1. 0
2. 0
3. 0
4. 0
5. 0
```


Portanto, zero é positivo e tem o valor 0*1 + 0*256.

```
- Número escolhido: 1516
1. 0
2. 0
3. 236
4. 5
5. 0
```

Portanto, 1516 é positivo e tem o valor 236*1 + 5*256.

```
- Número escolhido: -1
1. 0
2. 255
3. 255
4. 255
5. 0
```

Ou seja, -1 é negativo, e tem a forma em complemento de dois, FFFF h.

REPRESENTAÇÃO EM PONTO FLUTUANTE

A representação de cinco bytes de um número em ponto flutuante permite a manipulação de valores entre .29 E-38 a 1.7 E38.

O valor zero é sempre armazenado com todos os cinco bytes valendo zero. Todos os outros valores são armazenados com o expoente no primeiro byte e a mantissa nos outros quatro bytes.

Encontrar o expoente e a mantissa de um número decimal é, em essência, a conversão de números decimais para seus equivalentes em formato E.

Primeiramente, consideremos o número 1234.5, que pode ser expresso no formato E, como .1234 E+4 (repare que o ponto andou 4 casas para a esquerda). Para obter a mantissa de um número, o ponto decimal é deslocado para a esquerda até o último algarismo significativo. O expoente é o número de deslocamentos requeridos para a mantissa.

Como o TK 90X trabalha com números binários em vez de decimais, as mesmas operações serão agora aplicadas para simples valores binários.

Primeiro, consideremos o valor binário 00001 1111, que equivale a 31 decimal.

O ponto binário é colocado à direita dos 8 bits, e isso toma 5 deslocações para que ele fique à frente do bit mais significativo.

Nesta fase, o expoente e a mantissa são:

Exp. = + 5 e a mantissa = .1111 10000. . . .

No TK 90X essas partes são manipuladas de uma maneira um pouco diferente.

Para o expoente O verdadeiro expoente, + 5 acima, é sempre incrementado pelo decimal 128, ou 80h, para fornecer o expoente aumentado. Neste exemplo, +5 + 128 = 133.

Para a mantissa O bit mais significativo da mantissa é sempre 1, e, como é redundante, é substituído pelo bit de sinal, ou seja, quando tratando com números positivos será 0, e quando tratando com números negativos, será sempre 1.

No nosso exemplo, a mantissa é .01111 1000. . . , e agora precisamos expressar o número em cinco bytes. No caso da mantissa, os bytes que não são usados sempre valerão 0.

Portanto, a forma de representação em ponto flutuante do número 31 será 133, 120, 0,0,0; em hexadecimal será 85,78,00, 00,00 e em binário, 1000 0101 0111 1000 0000 0000 0000 0000 0000 0000

As mesmas operações se aplicam a números negativos e fracionários, embora as etapas de conversão não sejam tão fáceis quanto para números inteiros e positivos.

O programa a seguir mostra a forma decimal de números em ponto flutuante:

```
 10 INPUT n
 20 IF n = 0 THEN GOTO 40
 30 LET n = n + .2E -38
 40 PRINT "Numero escolhido"; n
 50 PRINT
 60 PRINT "Exp."; TAB 9; "Mantissa"
 70 LET v = PEEK 23627 + 256 * PEEK 23628
 80 PRINT PEEK (v + 1); TAB 9;
 90 FOR a = 2 TO 5
100 PRINT PEEK (v + a); CHR$ 32;
110 NEXT a
120 PRINT ' ' '
130 GOTO 10
```


Esse programa mostra que para os números seguintes, a forma de ponto flutuante é:

		Exp.	*Mantissa*
1.	=	129	0 0 0 0
2	=	130	0 0 0 0
35456	=	144	10 128 0 0
-1	=	129	128 0 0 0
-35456	=	144	138 128 0 0
6.333	=	131	74 167 239 158

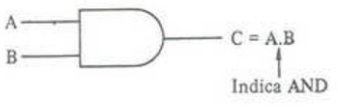
Capítulo **9**

Operações lógicas

O TK 90X utiliza circuitos eletrônicos lógicos que produzem uma saída que depende do valor de uma ou mais variáveis de entrada. Os valores de entrada e saída encontram-se no nível lógico 0 (0 volts) ou no nível lógico 1 (+ 5 volts).

OPERAÇÃO LÓGICA AND

A porta AND produz uma saída lógica 1 quando todas as entradas se encontram nesse mesmo nível lógico, ou seja, quando todas as entradas também valem 1; em qualquer outro caso, o resultado ou a saída é zero. O símbolo e a tabela verdade da porta AND de duas entradas são apresentadas a seguir:



Entrada		*Saída*
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Por exemplo:

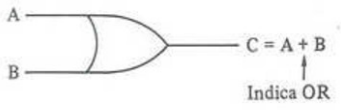
A = 0110 0101
B = 1101 0010

Portanto, A.B resultará:

0110 0101
1101 0010
0100 0000

OPERAÇÃO LÓGICA OR

A porta OR produz uma saída lógica 1 quando pelo menos uma das entradas se encontra no nível lógico 1. Isto significa que a saída só é 0 quando todas as entradas também valem 0.



Entrada		*Saída*
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Por exemplo:

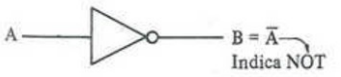
A = 0110 0101
B = 1101 0010

Portanto, A + B resultará:

0110 0101
1101 0010
1111 0111

OPERAÇÃO LÓGICA NOT

A porta NOT (porta inversora ou porta complementar) produz uma saída inversa (complementar) da entrada.



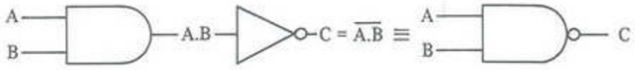
Entrada	*Saída*
A	B
0	1
1	0

Por exemplo:

A = 0110 0101
$\overline{A}$ = 1001 1010

OPERAÇÕES LÓGICAS NAND E NOR

Podem-se executar estas operações lógicas usando-se combinações de portas lógicas. *A porta NAND* é obtida combinando-se uma porta AND e uma porta NOT. Isto é mostrado, a seguir, para o caso de um elemento NAND de duas entradas. Esta porta só fornece uma saída lógica 0 quando todas as entradas se encontram no nível lógico 1; em qualquer outro caso o resultado será sempre 1.



Entrada		*Saída*
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

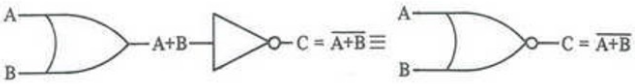
Por exemplo:

A = 0110 0101
B = 1101 0010
$\overline{A.B}$ = 1011 1111

A porta NOR é obtida combinando-se uma porta lógica OR e uma porta lógica NOT. Isto é mostrado a seguir, para o caso de um elemento NOR de duas entradas. Esta porta só produz uma saída lógica 1 quando todas as entradas possuem nível lógico 0; em todos os outros casos produz saída 0.

Utilizando-se os valores anteriores para A e B, teremos:

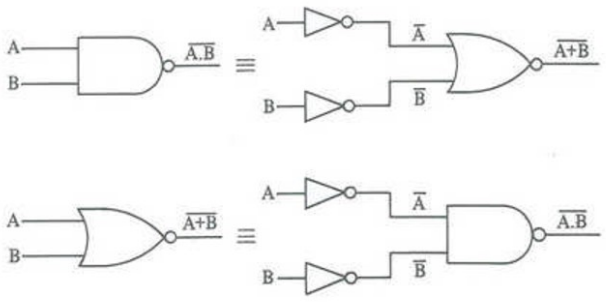
$\overline{A+B}$ = 0000 1000



É igualmente interessante notar que as operações lógicas NAND e NOR sobre as variáveis numéricas A e B podem ser substituídas por uma forma equivalente:

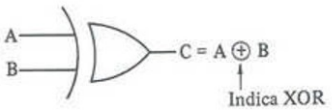
NAND ⟶ $\overline{A.B}$ = $\overline{A}+\overline{B}$
NOR ⟶ $\overline{A+B}$ = $\overline{A}.\overline{B}$

Ou através dos diagramas lógicos:



OPERAÇÃO LÓGICA OR EXCLUSIVO (XOR)

Pode-se executar esta operação utilizando-se um circuito de porta lógico. *A porta OR* exclusiva produz uma saída lógica 1 quando qualquer das entradas, mas não todas, estiver no nível lógico 1. Isto implica que a saída só estará no nível 0 quando todas as entradas tiverem o mesmo valor lógico, seja 0 ou 1.



Entrada		*Saída*
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Capítulo 10

As instruções em código de máquina do Z80

Essas instruções, que a partir deste capítulo passaremos a discutir e posteriormente aplicar em exemplos práticos, são divididas em 18 grupos, onde cada instrução tem alguma semelhança com as outras... Porém, antes de discuti-las, devemos mencionar seis classes de dados que podem completar uma instrução do Z80.

Essas classes são:

1- Um número, na faixa de um byte simples, ou seja, na faixa de 0 até 255, ou 0h até 255h. As instruções que requeiram um byte terão sua indicação seguida de "dd".

Por exemplo, a mnemônica "LD D, dd".

2- Um número de dois bytes, na faixa de 0 a 65535, ou 0h a FFFFh, representado por "dddd", como na mnemônica LD BC, dddd.

3- Um endereço (2 bytes), na mesma faixa numérica que a anterior, porém com a representação junto com as mnemônicas de "end", como, por exemplo, a instrução "JP end".

4- Um deslocamento de um byte, ou seja, um número na faixa idêntica a de um byte, considerado obrigatoriamente em sua forma de complemento de dois aritmético, com representação após a mnemônica de "e", como, por exemplo, na instrução "JR e".

5- Um byte para deslocamento indexado, também na faixa numérica de um byte, e aqui nesta classe, também considerado em sua forma de complemento de dois aritmético, com representação nas instruções que requeiram este tipo de deslocamento de "d", como, por exemplo, "LD A, (IX + d)".

6- Um byte para deslocamento indexado e um simples byte na faixa numérica de -128d a +127d, para o primeiro byte, e na faixa numérica de 0d a 255d, para o segundo valor, com as respectivas representações de "d" e "dd", como na instrução "LD (IX + d), dd".

Passemos então aos grupos de instruções do Z80.

GRUPO 1 – AS INSTRUÇÕES DE NÃO-OPERAÇÃO"

Mnemônica	Código hexa
NOP	00

Esta instrução NOP, quando executada pelo microprocessador, faz com que este interrompa suas atividades por cerca de 1,14 microssegundos. Nenhum dos registros ou indicadores é afetado por esta instrução.

Esta instrução é muito útil em casos de se querer uma pausa determinada em certas rotinas de linguagem de máquina, ou para cancelar, ou apagar instruções em código de máquina em rotinas já prontas (como aquela de seu programa predileto, que exibe a mensagem de copyright...).

GRUPO 2 – INSTRUÇÕES DE CARREGAR REGISTROS COM VALORES NUMÉRICOS

As instruções a seguir são desenvolvidas no sentido de carregar registros com simples bytes, constantes.

Mnemônica	*Código hexa*
LD A, dd	3Edd
LD H, dd	26dd
LD L, dd	2Edd
LD B, dd	06dd
LD C, dd	0Edd
LD D, dd	16dd
LD E, dd	1Edd

Como se pode notar pelos códigos hexa, estas instruções requerem duas locações da memória, uma para o código da instrução e outra para o valor envolvido. As instruções acima podem ser consideradas análogas às instruções de atribuição em Basic, onde se atribui um valor qualquer a uma variável de nome conhecido. No nosso caso, atribuímos a um registro determinado o valor de um byte especificado, ou, em outros termos, carregamos aquele registro com o conteúdo especificado. Esta valor será então armazenado no registro, anulando o anterior.

As instruções a seguir são análogas às anteriores, porém envolvem pares de registros e valores de dois bytes.

Mnemônica	*Código hexa*
LD HL, dddd	21dddd
LD BC, dddd	01dddd
LD DE, dddd	11dddd
LD IX, dddd	DD21dddd
LD IY, dddd	FD21dddd
LD SP, dddd	31dddd

Estas instruções vão requerer três ou quatro locações na memória.

O primeiro byte do valor numérico da instrução vai sempre para o registro menos significativo do par de registros envolvido na instrução, enquanto o segundo byte, logicamente, vai para o outro registro envolvido, o registro mais significativo do par. Entende-se por registro menos significativo os registros L, C, E, X, Y ou P, e por registro mais significativo, os registros H, B, D, I, ou S.

Também estas instruções são análogas às instruções de atribuição de valores da linguagem em Basic. Aqui, elas carregam aquele par de registros com um valor especificado.

As instruções deste grupo não afetam as bandeiras.

GRUPO 3 – INSTRUÇÕES DE COPIAR E TROCAR CONTEÚDO DE REGISTROS

São ao todo 59 instruções do universo de instruções do microprocessador Z80, que tratam de copiar conteúdos de registros ou par de registros. Estas instruções podem ser subdivididas em quatro subgrupos.

SUBGRUPO 1 – INSTRUÇÕES DE COPIAR CONTEÚDOS DE REGISTROS SIMPLES

A tabela abaixo fornece o código das instruções que tratam da cópia de conteúdos de simples registros, genericamente denominados registros r, para outros registros especificados.

Registro r	LD A,r	LD H,r	LD L,r	LD B,r	LD C,r	LD D,r	LD E,r
A	7F	67	6F	47	4F	57	5F
H	7C	64	6C	44	4C	54	5C
L	7D	65	6D	45	4D	55	5D
B	78	60	68	40	48	50	58
C	79	61	69	41	49	51	59
D	7A	62	6A	42	4A	52	5A
E	7B	63	6B	43	4B	53	5B

Nenhuma das instruções contidas nessa tabela afetam as bandeiras indicadoras de estado. Existem ainda quatro instruções envolvendo os registros I e R:

Mnemônica	*Código hexa*
LD A, I	ED57
LD A, R	ED5F
LD I, A	ED47
LD R, A	ED4F

Estas instruções afetam a bandeira indicadora de paridade ou excesso (mais adiante você verá o que faz esta bandeira).

SUBGRUPO 2 – INSTRUÇÕES DE COPIAR CONTEÚDOS DE PAR DE REGISTROS

São apenas três instruções neste subgrupo, e todas envolvem o par de registro de função especial, chamado "ponteiro da pilha" (stack pointer).

Mnemônica	*Código hexa*
LD SP, HL	F9
LD SP, IX	DDF9
LD SP, IY	FDF9

Estas instruções não afetam as bandeiras indicadoras de estado.

Note que não existem operações de cópia de conteúdo de pares de registros genéricos e, portanto, as instruções acima não são apropriadas para o caso. Esta operação é desenvolvida com duas instruções de carregamento de registros simples e cópia de registros simples.

Por exemplo, para se desenvolver a operação: LD HL, DE, utilizamos primeiramente LD H, D e em seguida LD L, E.

Como alternativa, que consome mais memória e mais tempo de execução, o conteúdo do primeiro par de registros pode ser armazenado na pilha da máquina, para em seguida ser copiado no segundo registro (veja instruções da pilha).

SUBGRUPO 3 – INSTRUÇÃO EX DE,HL

Mnemônica	*Código hexa*
EX DE,HL	EB

Esta instrução, muito útil por sinal, permite que o programador troque o conteúdo do par de registros DE com aquele armazenado pelo par de registros HL, sem afetar qualquer bandeira indicadora de estado, com grande velocidade e economia de memória.

Esta instrução é normalmente utilizada quando um endereço ou um valor numérico que ocupa dois bytes deve ser movido do par de registros DE para o par HL, mas sem cancelar ou perder o conteúdo original do par HL.

SUBGRUPO 4 – INSTRUÇÕES COM O GRUPO ALTERNATIVO DE REGISTROS

Mnemônica	*Código hexa*
EXX	D9
EX AF, A'F'	08

A instrução EXX faz com que o conteúdo dos registros H, L, B, C, D, e E sejam trocados respectivamente com os conteúdos dos registros H', L', B', C', D' e E'.

A instrução EX AF, A'F', conforme o seu nome sugere, faz a troca entre os registros AF e A'F'.

Estes registros alternativos são sempre utilizados para armazenar endereços ou valores numéricos, protegendo estes valores contra qualquer erro ou acidente no decorrer do programa, podendo ser utilizados a qualquer momento, de um modo muito rápido e fácil.

GRUPO 4 – INSTRUÇÕES PARA CARREGAMENTO DE REGISTROS COM VALORES NUMÉRICOS COPIADOS DE LOCAÇÕES DA MEMÓRIA

O conjunto das instruções do Z80 possui muitas instruções que procuram dados em endereços da memória, para então carregá-los em determinados registros. Todas estas instruções requerem do programador a especificação do endereço, ou endereços, de onde os dados devam ser copiados, para, então, o registro, ou o par de registros receber estes dados.

As instruções deste grupo podem ser subdivididas em três subgrupos, dependendo da técnica de endereçamento selecionada pelo programador.

Estas técnicas de endereçamento são:

1- Endereçamento absoluto – o atual endereço de dois bytes é especificado segundo sua própria instrução.

2- Endereçamento indireto – o endereço de dois bytes está sempre disponível em algum par de registros de endereçamento.

3- Endereçamento indexado – o endereço da locação *a* deve ser computado pela adição de um valor de deslocamento, *d*, ao endereço base armazenado no par de registros IX ou IY.

SUBGRUPO 1 – INSTRUÇÕES UTILIZANDO ENDEREÇAMENTO ABSOLUTO

Mnemônica	*Código hexa*	
LD A, (end)	3A endereço	
LD HL, (end)	2A end	forma usual
	ED6B end	forma não usual
LD BC, (end)	ED4B end	
LD DE, (end)	ED5B end	
LD IX, (end)	DD2A end	
LD IY, (end)	FD2A end	
LD SP, (end)	ED7B end	

A instrução LD A, (end) é a única instrução do conjunto de instruções do Z80 que permite carregar, em endereçamento direto ou absoluto, o conteúdo especificado daquela locação da memória em um registro simples. Note que as instruções remanescentes deste grupo podem ser consideradas como sendo instruções duplas, ou seja, por exemplo, a instrução LD BC, (end) pode ser considerada como primeiramente LD C, (end) seguida de LD B, (end + 1).

Em qualquer caso, o conteúdo do endereço especificado é copiado no registro menos significativo, e o conteúdo do endereço seguinte é copiado no registro mais significativo.

SUBGRUPO 2 – INSTRUÇÕES QUE UTILIZAM ENDEREÇAMENTO INDIRETO

Mnemônica	*Código hexa*
LD A, (HL)	7E
LD A, (BC)	0A
LD A, (DE)	1A
LD A, (HL)	66
LD L, (HL)	6E
LD B, (HL)	46
LD C, (HL)	4E
LD D, (HL)	56
LD E, (HL)	5E

Em todos os casos, o endereço da locação de onde o byte deve ser copiado deve estar presente obrigatoriamente em algum par de registro entre HL, DE ou BC.

Note que, por exemplo, a instrução LD D, (BC) não existe, e, portanto, devem ser executadas outras instruções que efetuem o mesmo processamento:

1- LD A, (BC) seguida de LD D, A, que alterará o conteúdo do registro A, ou:

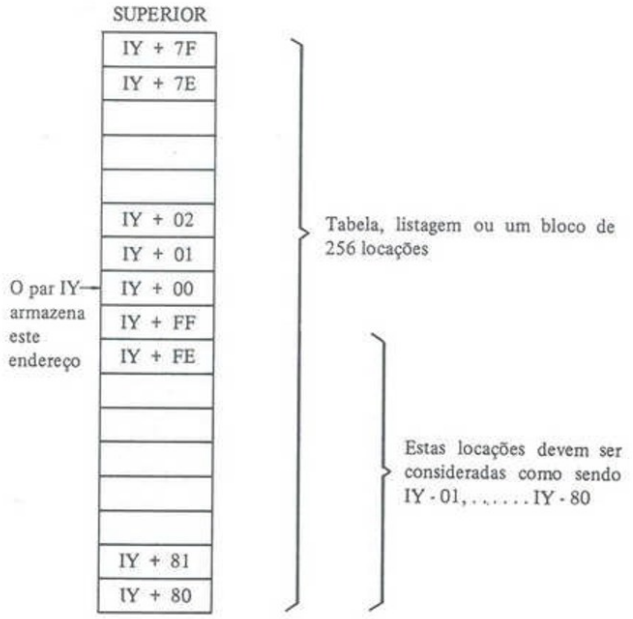
2- LD H, B, seguida de LD L, C e finalmente LD D, (HL), que alterará o conteúdo de HL.

SUBGRUPO 3 – INSTRUÇÕES USANDO ENDEREÇAMENTO INDEXADO

As instruções deste subgrupo permitem ao programador carregar simples bytes de dados, em registros simples, armazenados sob a forma de listagens, tabelas, ou apenas blocos de dados. O endereço base é armazenado no par de registros de indexação apropriado.

NOTA: Você já deve ter notado que as instruções envolvem os pares de registros IX e IY diferem apenas no código inicial, ou seja, para o par IX utiliza-se o código DD, e para o par IY utiliza-se o código FD. Portanto, no grupo de instruções anterior, para as instruções IY, substitua os códigos DD por FD.

O diagrama abaixo ilustra esse efeito:



Mnemônica	*Código hexa*
LD A, (IX + d)	DD73d
LD H, (IX + d)	DD66d
LD L, (IX + d)	DD6Ed
LD B, (IX + d)	DD46d
LD C, (IX + d)	DD4Ed
LD D, (IX + d)	DD56d
LD E, (IX + d)	DD5Ed

É interessante notar o tempo que o microprocessador Z80 leva para executar essas instruções. As instruções mais rápidas são as que compõem o subgrupo 2, que requerem do Z80 a busca de um simples byte de código e, então, o byte seguinte de dado.

As instruções do subgrupo 1 são mais complicadas e conseqüentemente levam mais tempo para ser executadas. Em termos técnicos, elas necessitam de 16 a 20 ciclos de tempo, e, finalmente, as instruções do subgrupo 3 são as que levam mais tempo ainda, apesar da sua grande praticidade. Elas necessitam de 19 ciclos de tempo para ser executadas.

Nenhuma das instruções deste grupo afeta as bandeiras indicadoras.

GRUPO 5 – INSTRUÇÕES PARA ARMAZENAR DADOS COPIADOS DE REGISTROS, OU VALORES NUMÉRICOS, EM ENDEREÇOS OU LOCAÇÕES DA MEMÓRIA

De uma maneira geral, as instruções deste grupo efetuam operações opostas às do grupo anterior.

Estas instruções permitem que conteúdos de registros especificados pelo programador sejam copiados em endereços da memória específicos, ou que valores numéricos sejam armazenados nesses endereços.

Outra vez, estas instruções são melhor analisadas, se divididas em três subgrupos:

SUBGRUPO 1 – INSTRUÇÕES UTILIZANDO ENDEREÇAMENTO ABSOLUTO

Mnemônica	*Código hexa*	
LD (end), A	32 end	
LD (end), HL	22 end	ou ED63 end (forma não usual)
LD (end), BC	ED43 end	
LD (end), DE	ED53 end	
LD (end), IX	DD22 end	
LD (end), IY	FD22 end	
LD (end), SP	ED73 end	

As instruções acima são as únicas a utilizar endereçamento absoluto, e é importante notar que não existe uma instrução para carregar um endereço específico com um valor numérico. Se o programador necessita efetuar essa operação, deve fazê-lo, carregando primeiramente o registro A, com o valor especificado, para então armazenar o valor desejado no endereço especificado.

Novamente uma instrução do tipo "LD (end), HL" é na realidade uma instrução dupla, pois requer LD (end), L e então LD (end + 1), H.

As instruções deste subgrupo são muito utilizadas para armazenar endereços e números em locações da memória, quando estes valores são considerados variáveis.

Por exemplo, é muito comum programadores em linguagem de máquina escreverem:

LD (RAMTOP), HL

SUBGRUPO 2 – INSTRUÇÕES QUE USAM ENDEREÇAMENTO INDIRETO

As instruções deste subgrupo permitem ao programador a cópia de conteúdos de registros simples em endereços da memória, que estejam armazenados nos pares de registros HL, BC, DE. Também existe uma instrução para carregar um byte em uma locação endereçada pelo par de registros HL.

Mnemônica	*Código hexa*
LD (HL), A	77
LD (BC), A	02
LD (DE), A	12
LD (HL), H	74
LD (HL), L	75
LD (HL), B	70
LD (HL), C	71
LD (HL), D	72
LD (HL), E	73
LD (HL), dd	36dd

SUBGRUPO 3 – INSTRUÇÕES UTILIZANDO ENDEREÇAMENTO INDEXADO

Mnemônica	*Código hexa*
LD (IX + d), A	DD77d
LD (IX + d), H	DD74d
LD (IX + d), L	DD75d
LD (IX + d), B	DD70d
LD (IX + d), C	DD71d
LD (IX + d), D	DD72d
LD (IX + d), E	DD73d
LD (IX + d), dd	DD36d dd

Novamente, para instruções que envolvem o par de registros IY, mude DD para FD e IX para IY.

GRUPO 6 – INSTRUÇÕES DE ADIÇÃO

Este grupo é o primeiro dos quatro grupos do conjunto de instruções do microprocessador Z80, que envolve operações aritméticas ou lógicas.

As instruções de adição permitem ao programador adicionar, em aritmética binária absoluta, um número especificado ao conteúdo de um registro simples, ou ao conteúdo de um par de registros ou, ainda, a um endereço indexado da memória.

As instruções deste grupo podem ser subdivididas em três subgrupos, de acordo com as suas mnemônicas.

Estes três subgrupos são:

1- Instruções de adição (ADD)

2- Instruções de incremento (INC), ou seja, um caso especial de adição, quando apenas uma unidade (1) é adicionada a um número existente.

3- Instruções de adição, porém considerando-se o estado da bandeira indicadora de transporte (carry flag) (ADC). Conforme já vimos, esta bandeira indicadora de transporte é um bit do registro F, utilizado para nos avisar se houve aquele nosso conhecido "vai um" nas operações aritméticas de soma.

As instruções de adição, bem como as instruções de adição com transporte, afetam a bandeira indicadora de transporte, mas as instruções de incremento não, fato esse que em algumas situações oferece vantagens.

SUBGRUPO 1 – INSTRUÇÕES DE ADIÇÃO – ADD

Mnemônica	*Código hexa*
ADD A,dd	C6dd
ADD A,A	87
ADD A,H	84
ADD A,L	85
ADD A,B	80
ADD A,C	81
ADD A,D	82
ADD A,E	83
ADD A,(HL)	86
ADD A,(IX + d)	DD86d
ADD A,(IY + d)	FD86d
ADD HL,SP	39

ADD IX,IX	DD29
ADD IX,BC	DD09
ADD IX,DE	DD19
ADD IX,SP	DD39

Para instruções que envolvem o par de registros IY, substitua IX por IY e os códigos DD por FD.

Veja os exemplos:

1- Registro A armazenando 60h
Registro B armazenando 90h
ADD A,B resultará:
Registro A armazenando F0h
Registro B armazenando 90h (conteúdo inalterado)
Bandeira indicadora de transporte resetada

2- Registro A armazenando A8h
Registro B armazenando 7Eh
ADD A,B resultará:
Registro A armazenando 26h
Registro B armazenando 7Eh (inalterado)
Bandeira indicadora de transporte setada (houve transporte).

Confira com lápis e papel.

SUBGRUPO 2 – INSTRUÇÕES DE INCREMENTO

As instruções deste grupo permitem que o conteúdo de um simples registro, ou de um par de registros, ou mesmo de uma locação da memória seja incrementado em uma unidade. Em todos os casos, a bandeira indicadora de transporte é ignorada, ou seja, inalterada.

Mnemônica	*Código hexa*
INC A	3C
INC H	24
INC L	2C
INC B	04
INC C	0C
INC D	14
INC E	1C
INC (HL)	34
INC (IX + d)	DD34d
INC (IY + d)	FD34d

INC HL	23
INC BC	03
INC DE	13
INC SP	33
INC IX	DD23
INC IY	FD23

SUBGRUPO 3 – INSTRUÇÕES DE ADIÇÃO COM TRANSPORTE (ADC)

Mnemônica	*Código hexa*
ADC A,dd	CEdd
ADC A,A	8F
ADC A,H	8C
ADC A,L	8D
ADC A,B	88
ADC A,C	89
ADC A,D	8A
ADC A,E	8B
ADC A,(HL)	8E
ADC A,(IX + d)	DD8Ed
ADC A,(IY + d)	FD8Ed
ADC HL,HL	ED6A
ADC HL,BC	ED4A
ADC HL,DE	ED5A
ADC HL,SP	ED7A

As instruções deste subgrupo permitem ao programador adicionar dois números, juntamente com o estado da bandeira indicadora de transporte, pois todas as instruções deste grupo, conforme as próprias mnemônicas dizem, afetam essa bandeira. Esta será resetada (0) se a instrução ADC em execução não der excesso, e será setada (1), se houver excesso na operação corrente.

Por exemplo:

1- Registro A armazenando 60h
Registro B armazenando 90h
Bandeira indicadora de transporte setada
A operação ADC A,B resultará:
Registro A armazenando F1h
Registro B armazenando 90
Bandeira indicadora de transporte resetada

2- Registro A armazenando A8h
Registro B armazenando 7Eh
Bandeira indicadora de transporte setada
A operação ADC A,B resultará:
Registro A armazenando 27h
Registro B armazenando 7Eh
E a bandeira de transporte setada

Agora, pegue um lápis e um papel, pegue esses valores hexadecimais acima, converta-os para o sistema decimal e confira o resultado, podendo considerar a bandeira indicadora de transporte como sendo um "nono bit" do acumulador (registro A).

GRUPO 7 – INSTRUÇÕES DE SUBTRAÇÃO

As instruções de subtração permitem que o programador subtraia, em aritmética binária absoluta, um número especificado do conteúdo de um registro simples, de um par de registros ou de uma locação endereçada da memória.

Novamente este grupo pode ser subdividido em três subgrupos, conforme suas mnemônicas.

Esses três subgrupos são:

1- Instruções SUB – de subtração simples

2- Instruções DEC – casos especiais de subtração, onde um número específico é decrementado de uma unidade.

3- Instruções SBC – o valor da bandeira indicadora de transporte também é subtraído do resultado.

Deste grupo, apenas as instruções DEC não afetam a bandeira indicadora de transporte.

SUBGRUPO 1 – INSTRUÇÕES SUB

Mnemônica	*Código hexa*
SUB dd	D6dd
SUB A	97
SUB H	94
SUB L	95
SUB B	90

SUB C	91
SUB D	92
SUB E	93
SUB (HL)	96
SUB (IX + d)	DD96d
SUB (IY + d)	FD96d

Nota: As mnemônicas para as instruções de subtração SUB são normalmente escritas na forma acima, ou seja, "SUB L" é o mesmo que "SUB A,L" etc, pois todas as instruções de subtração deste subgrupo envolvem o acumulador, ou o registro A.

No Z80, as instruções de subtração fornecem uma subtração binária absoluta "verdadeira", conforme exemplo a seguir. A bandeira indicadora de transporte é resetada se o valor original do registro A é "maior que" ou "igual" ao subtraendo (o segundo número na subtração), e é setada se o valor de registro A é "menor que".

Exemplos:

1- Com o registro A armazenando DCh
e o registro B armazenando AAh

A instrução SUB B resultará em A armazenando 32h, e a bandeira indicadora de transporte estará resetada, porque não houve excesso.

2- Com o registro A armazenando AAh
e o registro B armazenando DCh

A instrução SUB B resultará em A armazenando CEh, o registro B armazenando DCh e a bandeira indicadora de transporte estará setada, pois houve excesso.

Confira.

SUBGRUPO 2 – INSTRUÇÕES DEC (DE DECREMENTO)

As instruções deste subgrupo permitem que seja subtraída uma unidade do conteúdo de um registro simples de 8 bits, ou de um par de registros, ou de um endereço da memória. Em qualquer caso, a bandeira indicadora de transporte não é afetada.

Mnemônica	*Código hexa*
DEC A	3D
DEC H	25
DEC L	2D
DEC B	05
DEC C	0d
DEC D	15
DEC E	LD
DEC (HL)	35
DEC (IX + d)	DD35d
DEC (IY + d)	FD35d
DEC HL	2B
DEC BC	0B
DEC DE	1B
DEC SP	3B
DEC IX	DD2B
DEC IY	FD2B

SUBGRUPO 3 – INSTRUÇÕES DE SUBTRAÇÃO LEVANDO-SE EM CONSIDERAÇÃO A BANDEIRA INDICADORA DE TRANSPORTE

Mnemônica	*Código hexa*
SBC A,dd	DEdd
SBC A,A	9F
SBC A,H	9C
SBC A,L	9D
SBC A,B	98
SBC A,C	99
SBC A,D	9A
SBC A,E	9B
SBC A,(HL)	9E
SBC A,(IX + d)	DD9Ed
SBC A,(IY + d)	FD9Ed
SBC HL,HL	ED62
SBC HL,BC	ED42
SBC HL,DE	ED52
SBC HL,SP	ED72

Uma operação SBC efetuará uma subtração verdadeira se a bandeira indicadora de transporte estiver resetada, mas executará uma subtração, considerando o excesso, se a bandeira de transporte estiver setada.

GRUPO 8 – INSTRUÇÕES DE COMPARAÇÃO

As instruções deste grupo são usadas muito freqüentemente em qualquer rotina de código de máquina. Elas permitem que o programador compare o valor armazenado no registro A, com uma constante, um valor armazenado em um registro qualquer ou um valor armazenado em um endereço da memória.

Uma instrução de comparação efetua uma operação de subtração, sem transporte, mas descarta a resposta após usá-la para setar as devidas bandeiras indicadoras do registro F. O valor original do registro A permanece inalterado.

A bandeira indicadora de transporte é afetada da mesma maneira que numa operação de subtração. Uma comparação que seja "maior que" ou "igual a" *reseta* a bandeira de transporte, e uma comparação que seja "menor que", *seta* a bandeira de transporte.

As instruções deste grupo são instruções de comparação simples, e as instruções de comparações de blocos serão consideradas mais tarde.

Mnemônica	*Código hexa*
CP dd	FEdd
CP A	BF
CP H	BC
CP L	BD
CP B	B8
CP C	B9
CP D	BA
CP E	BB
CP (HL)	BE
CP (IX + d)	DDBEd
CP (IY + d)	FDBEd

Essas instruções desenvolvem processamento análogo às instruções em Basic, de operações e decisões lógicas, tipo IF...THEN...

GRUPO 9 – INSTRUÇÕES LÓGICAS

No conjunto de instruções do Z80 existem instruções para processarem operações AND, OR e XOR que comparam o conteúdo do acumulador com o conteúdo de outra locação especificada. A operação é desenvolvida bit a bit, e o resultado de 8 bits é armazenado no acumulador.

Dividimos este grupo em três subgrupos, de acordo com suas mnemônicas.

SUBGRUPO 1 – INSTRUÇÕES AND

A operação lógica AND efetuada entre dois bits dará como resultado um bit com valor 1, apenas se os dois bits envolvidos na operação valerem 1 também. Em outro caso, o resultado será 0.

O exemplo a seguir motra como uma instrução AND desenvolve oito operações separadas bit a bit.

10101010	hexa AA
AND	AND
11000000	CO
=10000000	= 80

Outra vez, pegue um lápis e papel, e confira.

Mnemônica	*Código hexa*
AND dd	E6dd
AND A	A7
AND H	A4
AND L	A5
AND B	A0
AND C	A1
AND D	A2
AND E	A3
AND (HL)	A6
AND (IX + d)	DDA6d
AND (IY + d)	FDA6d

Ao usar uma instrução AND, todos os bits do acumulador serão resetados. Esses processo permite ao programador controlar certos bits de um byte de dados.

SUBGRUPO 2 – INSTRUÇÕES OR

A operação lógica OR, executada entre dois bits, dará como resultado um terceiro bit, valendo 1, se apenas um ou até os dois bits envolvidos valerem 1.

O exemplo a seguir ilustra esse processo.

10101010	AA
OR	OR
11000000	CO
=11101010	=EA
Mnemônica	*Código hexa*
OR dd	F6dd
OR A	B7
OR H	b4
OR L	B5
OR B	B0
OR C	B1
OR D	B2
OR E	B3
OR (HL)	B6
OR (IX + d)	DDB6d
OR (IY + d)	FDB6d

SUBGRUPO 3 – INSTRUÇÕES XOR

O operação lógica XOR, desenvolvida entre dois bits, dará como resultado um terceiro bit, que valerá 1 se apenas um bit dos dois envolvidos também valer 1, mas, veja bem, somente se um dos bits envolvidos valer 1, e nunca os dois!

O exemplo a seguir demonstra essa operação.

10101010	AA
XOR	XOR
11000000	CO
=01101010	=6A
Mnemônica	*Código hexa*
XOR dd	EEdd
XOR A	AF
XOR H	AC
XOR L	AD
XOR B	A8
XOR C	A9
XOR D	AA
XOR E	AB
XOR (IX + d)	DDAEd
XOR (IY + d)	FDAEd

Ao se usar a instrução XOR, os bits do registro A serão alterados se necessário for.

O uso das instruções XOR em rotinas em código de máquina são um pouco complicadas, mas a instrução XOR A é freqüentemente usada como uma alternativa a LD A,0. Ambas as instruções limpam o acumulador, mas a primeira utiliza apenas um endereço, enquanto a segunda consome dois endereços.

GRUPO 10 – INSTRUÇÕES DE SALTO (JUMP)

No conjunto de instruções do Z80 existem 17 instruções de salto, que permitem que o programador efetue saltos dentro de um programa. Um salto em código de máquina pode ser comparado à instrução BASIC "GOTO".

As instruções deste grupo são melhor analisadas, se divididas em oito subgrupos. Quatro destes subgrupos contêm instruções condicionais que dependem do estado das bandeiras indicadoras do registro F.

SUBGRUPO 1 – INSTRUÇÃO DE SALTO ABSOLUTO (JP de jump)

Esta é a instrução clássica de salto. Quando a instrução JP end é executada, permite que o endereço especificado seja armazenado no PC, ou contador de programas, fazendo com que a execução do programa continue a partir daquele endereço.

Mnemônica	*Código hexa*
JP end	C3 end

SUBGRUPO 2 – INSTRUÇÕES DE SALTO QUE UTILIZAM ENDEREÇAMENTO INDIRETO

Mnemônica	*Código hexa*
JP (HL)	E9
JP (IX)	DDE9
JP (IY)	FDE9

Estas três instruções permitem que um byte devidamente armazenado no par de registros especificado seja carregado no contador de programas. As instruções deste subgrupo são freqüentemente utilizadas quando deve ser feito um salto para uma locação especificada em uma tabela de endereços.

SUBGRUPO 3 – INSTRUÇÕES DE SALTO RELATIVO
(JR de jump relative)

Mnemônica	*Código hexa*
JR e	18 e

Esta instrução permite que o programador salte 127 endereços para a frente (positivos) e 128 endereços para trás (negativos), a partir do endereço corrente. Note que o endereço corrente é de fato o seguinte ao deslocamento e, especificado na instrução.

O deslocamento e é sempre considerado, em aritmética, complemento de dois, e um deslocamento e positivo dá o número de locações que devem ser saltados acima, enquanto um deslocamento e negativo mostra em quanto o contador de programas deve ser reduzido.

SUBGRUPO 4 – INSTRUÇÕES DE SALTO CONDICIONAIS RELATIVAS AO ESTADO DA BANDEIRA INDICADORA DE TRANSPORTE

São quatro instruções neste subgrupo, que permitem que seja feito um salto somente se a bandeira indicadora de transporte estiver no estado especificado pela instrução.

Agora vamos ver realmente o que é esta bandeira indicadora de transporte.

Esta bandeira é o bit 0 do registro F e é essencialmente um indicador que mostra se houve ou não um excesso numa operação binária, ou seja, se ela está setada em certas situações e resetada em outras ocasiões. Em muitas situações, a bandeira indicadora de transporte não é afetada pela execução de algumas instruções.

Em resumo:

1. Todas as instruções ADD e ADC afetam esta bandeira. Se não houver excesso a bandeira será resetada, mas se houver excesso, a bandeira será setada.

2. Todas as instruções SUB, SBC e CP afetam também esta bandeira.

3. Instruções como AND, OR e XOR resetam a bandeira indicadora de transporte.

4. As instruções de rotação que veremos adiante também afetam esta bandeira.

As instruções deste subgrupo são:

Mnemônica	*Código hexa*
JP NC, end	D2 end
JR NC, e	30 e
JP C, end	DA end
JR C, e	38 e

Nas duas primeiras instruções, o salto somente será executado se a bandeira indicadora de transporte estiver resetada.

Nas duas últimas, o salto será executado se a bandeira de transporte estiver setada (valer 1).

SUBGRUPO 5 – INSTRUÇÕES DE SALTO CONDICIONAIS RELATIVAS AO ESTADO DA BANDEIRA ZERO

Esta bandeira zero é o bit 6 do registro F e em muitas vezes indica se o resultado de uma determinada operação foi zero (quando ela estará setada, ou seja, valerá 1) ou se o resultado de uma operação foi diferente de zero (quando então ela estará resetada, quer dizer, valerá 0).

Por exemplo:

6C ADD 5A resultará C6 e a bandeira zero resultada, mas
6C ADD 94 resultará 00 e a bandeira zero setada.

Pegue o lápis e papel confira estes resultados, pois na minha opinião esta é a melhor maneira de você fixar estes conceitos fundamentais.

Resumindo:

1. Instruções ADD, INC, ADC, SUB, DEC, SBC, CP, AND, OR e XOR usando registros simples de um byte, e as instruções ADC e SBC usando pares de registros irão setar a bandeira zero, se o resultado da operação em questão for zero.

2. Instruções de rotação, ou instruções de testar determinados bits (que também veremos mais à frente) ou instruções de procura de blocos afetam a bandeira zero.

3. Instruções LD, com exceção de LD A, I e LD A, R, não afetam esta bandeira.

Neste subgrupo são quatro instruções, que permitem que seja efetuado um salto apenas se o estado da bandeira zero coincidir com o especificado pela instrução.

Mnemônica	*Código hexa*
JP NZ, end	C2 end
JR NZ, e	20 e
JP Z, end	CA end
JR Z, e	28 e

Como o caso anterior, as duas primeiras instruções permitem que seja considerado o salto apenas se a bandeira zero estiver resetada, enquanto nas duas últimas, o salto somente será executado se a bandeira zero estiver setada.

SUBGRUPO 6 – INSTRUÇÕES DE SALTO CONDICIONAIS RELATIVAS AO ESTADO DA BANDEIRA DE SINAL

Esta bandeira de sinal é o bit 7 do registro F, e em muitos casos é uma cópia do bit mais à esquerda do resultado.

Sempre que um número de 8 bits, ou um número de 16 bits é considerado na sua forma de complemento de 2 aritmético, então o bit mais à esquerda, ou seja, o bit 7 ou o bit 15 é considerado como bit de sinal. Atenção – *este bit de sinal será resetado para números positivos e setado para números negativos.*

Resumindo:

1. Instruções ADD, INC, ADC, SUB, DEC, SBC, CP, AND, OR e XOR usando registros simples de 8 bits, e as instruções ADC e SBC usando pares de registros afetam a bandeira de sinal.

2. Instruções de busca de blocos e muitas instruções de rotação também afetam esta bandeira.

3. Instruções LD, com exceção de LD A,I e LD A,R, não afetam a bandeira de sinal.

Neste subgrupo, são duas instruções que permitem que seja realizado um salto somente se o estado da bandeira de sinal coincidir com a especificação da instrução.

Mnemônica	*Código hexa*
JP P, end	F2 end
JP M, end	FA end

Na primeira instrução o salto somente será executado se o resultado for positivo, e na segunda, se for negativo.

As instruções deste subgrupo não são comumente usadas, porque requerem endereçamento absoluto e também porque um bit de sinal pode ser lido de diversas maneiras.

SUBGRUPO 7 – INSTRUÇÕES DE SALTO CONDICIONAIS RELATIVAS AO ESTADO DA BANDEIRA DE PARIDADE/EXCESSO

São duas instruções que permitem que seja efetuado um salto somente se a bandeira de paridade/excesso estiver nas condições especificadas pela instrução.

Esta bandeira é o bit 2 do registro F e é uma bandeira de propósito duplo. Certas instruções usam-na para indicar "excesso", enquanto outras instruções utilizam-na para armazenar o resultado de um teste de paridade.

O conceito de excesso, aqui, não se aplica ao excesso binário, mas ao excesso de um complemento de dois aritmético ilustrado a seguir:

Consideremos:
Em hexa OA ADD 5C = 66
Em decimal 10 ADD 92 = 102
Correto – não houve excesso

Em hexa 6A ADD 32 = 9C
Em decimal 106 ADD 50 = -100 (lembre-se do bit de sinal)
Errado – houve excesso

Excessos também acontecem com subtração. Veja:
Em hexa 83 SUB 14 = 6F
Em decimal -125 SUB 20 = 111
Errado – houve excesso

A bandeira de excesso/paridade é setada quando ocorrem excessos.

Lápis e papel na mão, para descobrir as operações exemplificadas acima.

O conceito de paridade refere-se ao número de bits setados em um determinado byte. A paridade existirá quando o número de bits setados for par.

Por exemplo:

– o byte 01010101 tem paridade par e a bandeira setada;
– o byte 00000001 tem paridade ímpar e a bandeira resetada.

Resumindo

1. Instruções ADD, ADC, SBC, CP usando registros simples e instruções ADC e SBC usando pares de registros têm seu resultado testado em função do excesso.

2. Instruções AND, OR e XOR e rotações têm seu resultado testado em função da paridade.

3. Uma instrução INC vai setar esta bandeira se o resultado for 80h, e uma instrução DEC vai setar a bandeira se o resultado for 7Fh.

4. Várias outras instruções também afetam a bandeira de excesso/paridade.

As instruções deste subgrupo são:

Mnemônica	*Código hexa*
JP PO, end	E2 end
JP PE, end	EA end

Na primeira instrução, o salto será executado se a paridade for ímpar ou não houver excesso.

Na segunda, o salto será executado se a paridade for par ou houver excesso.

As instruções deste subgrupo não são muito utilizadas, devido à confusão que podem causar, e porque também podem ser substituídas por outras instruções.

GRUPO 11 – INSTRUÇÃO DJNZ, e

Esta única instrução deste grupo é uma das mais úteis e das mais usadas de todo o conjunto de instruções do microprocessador Z80.

Mnemônica	*Código hexa*
DJNZ, e	10 e

A mnemônica significa "decremente o registro B e efetue um salto relativo se a bandeira zero estiver resetada".

Esta instrução pode ser comparada a um loop Basic, com passo negativo, do tipo:

```
FOR f = 10 to 1 STEP -1: ... :NEXT f
```

Nesse loop, a variável de controle **f** é inicializada em 10, e a cada passagem pela variável, ela é decrementada em uma unidade, até atingir o valor limite.

A instrução "DJNZ, e" é usada de uma maneira similar. Primeiramente o programador deve especificar o tamanho da variável do loop e armazená-la no registro B; a seguir, as instruções que serão repetidas, e, finalmente, muito cuidado deve ser tomado, a fim de que o valor "e" seja apropriado.

O exemplo abaixo mostra como se imprime o alfabeto na tela, com o emprego da instrução "DJNZ, e".

```
      LD B, 1A      São 26 letras no alfabeto
loop  LD A, 5B      A letra A é a primeira
      SUB B         Em hexa 5B - 1A = 41
      RST 0010      Instrução para imprimir mensagens na
                        tela (veremos adiante)
      DJNZ, loop    Pula para B, C etc., da mesma maneira
                        que NEXT f
```

Os códigos hexa para o exemplo acima são:

06, 1A, 3E, 5B, 90, D7, 10, FA

Lápis e papel, através do Anexo A deste livro, converta esses valores para decimal e tente fazer um programa que execute essa rotina.

GRUPO 12 – INSTRUÇÕES DA PILHA (STACK)

Em muitos programas em código de máquina, o uso extensivo da pilha da máquina é feito pelo programador como um lugar para guardar dados, e pelo microprocessador, para guardar endereços que serão utilizados posteriormente. As instruções que formam este grupo podem ser subdivididas em dois grupos do usuário e três subgrupos do microprocessador.

SUBGRUPO 1 – INSTRUÇÕES PUSH E POP

Estas instruções permitem que o programador "PUSH", isto é, guarde, ou salve, dois bytes de dados na pilha da máquina, para mais tarde "POP", ou seja, copiar da pilha aqueles dois bytes.

Este par de bytes pode ser copiado de e para um par de registros especificado.

Mnemônica	*Código hexa*
PUSH AF	F5
PUSH HL	E5
PUSH BC	C5
PUSH DE	D5
PUSH IX	DDE5
PUSH IY	FDE5
POP AF	F1
POP HL	E1
POP BC	C1
POP DE	D1
POP IX	DDE1
POP IY	FDE1

Quando uma instrução PUSH é executada, o ponteiro da pilha (*stack pointer*) é primeiramente decrementado para apontar para uma locação livre. Uma cópia do conteúdo do registro mais significativo do par de registros envolvido é então feita nesta locação. Então o ponteiro da pilha é decrementado novamente para armazenar na nova locação o valor contido no registro menos significativo do par envolvido.

As ações inversas e essas descritas são executadas no caso de instruções POP. Note que sempre que uma dessas instruções é executada, ao final delas, o ponteiro da pilha vai apontar para aquela locação especificada pelo programador, ou aquela onde estava o processamento normal do programa.

Nota importante – estas instruções são sempre usadas aos pares, ou seja, para cada instrução PUSH utilizada deve obrigatoriamente haver uma instrução POP equivalente.

SUBGRUPO 2 – INSTRUÇÕES DE MUDANÇA DO VALOR DA PILHA

As instruções deste subgrupo não são muito usadas, mas em algumas ocasiões são muito úteis.

Mnemônicas	*Código hexa*
EX (SP), HL	E3
EX (SP), IX	DDE3
EX (SP), IY	FDE3

Estas instruções permitem ao programador mudar, ou trocar, o valor corrente armazenado num par de registros especificado pela última entrada na pilha da máquina. E o ponteiro da pilha, qual seja, o par de registros SP não é alterado.

O uso destas instruções é um pouco confuso, principalmente em se tratando de rotinas em código de máquina mais extensas, e elas são melhores consideradas como alternativas a PUSH e POP em casos especiais.

Considere, por exemplo, a situação a seguir:

O valor AA está na pilha da máquina, e o valor BB está armazenado no par de registros HL.

É desejo do programador trocar estes valores por outros.

Existem duas maneiras de fazê-lo:

1. Usando a instrução EX (SP), HL;

2. Usando outro par de registros para armazenamento temporário para AA.

POP BC	Salva AA em BC
PUSH HL	B é colocado na pilha
PUSH BC	Move AA para HL
POP HL	de um modo ou de outro

As instruções deste subgrupo também podem ser utilizadas para manipular endereços de retorno.

SUBGRUPO 3 – INSTRUÇÕES CALL

As instruções em código de máquina CALL são diretamente equivalentes ao comando Basic GOSUB. As instruções estão incluídas neste grupo porque o microprocessador usa a pilha da máquina como uma área onde os endereços de retorno são armazenados.

Existem nove instruções neste subgrupo que permitem que uma sub-rotina seja chamada condicional ou incondicionalmente em relação ao estado das principais bandeiras indicadoras de estado.

Mnemônica	*Código hexa*	*Comentários*
CALL end	CD end	Incondicional
CALL C, end	DC end	Bandeira C setada
CALL NC, end	D4 end	Bandeira C resetada
CALL Z, end	CC end	Bandeira zero setada
CALL NZ, end	C4 end	Bandeira zero resetada
CALL M, end	FC end	Bandeira sinal setada
CALL P, end	F4 end	Bandeira sinal resetada
CALL PE, end	EC end	Bandeira paridade/excesso setada
CALL PO, end	E4 end	Bandeira paridade/excesso resetada

As ações de uma instrução CALL são:

1. O valor corrente do PC, ou contador de programas, isto é, o endereço da primeira locação após "end" da instrução CALL, é salvo na pilha da máquina. O ponteiro da pilha é manipulado como numa instrução PUSH. O byte mais significativo do contador de programas vai para a locação seguinte à do byte menos significativo.

2. O endereço é então copiado no contador de programas e a execução do programa propriamente dito continua.

SUBGRUPO 4 – INSTRUÇÕES RET

As instruções em código de máquina RET são diretamente equivalentes ao comando Basic RETURN. Também são nove instruções neste subgrupo, que permitem o retorno condicional ou incondicional, dependendo do estado das bandeiras principais.

Mnemônica	*Código hexa*	*Comentários*
RET	C9	Incondicional
RET C	D8	Bandeira C setada
RET NC	D0	Bandeira C resetada
RET Z	C8	Bandeira zero setada
RET NZ	C0	Bandeira zero resetada
RET M	F8	Bandeira sinal setada
RET P	F0	Bandeira sinal resetada
RET PE	E8	Bandeira paridade/excesso setada
RET PO	E0	Bandeira paridade/excesso resetada

A ação de uma instrução RET é a de copiar a última entrada na pilha da máquina para o contador de programas. No entanto, o ponteiro da pilha é incrementado duas vezes.

Não é nada comum, em Basic, manipular a pilha de GOSUB, mas em linguagem de máquina freqüentemente somos obrigados a fazê-lo, o que requer muitos cuidados com os valores a serem processados, bem como com os endereços a serem manipulados.

SUBGRUPO 5 – INSTRUÇÕES RST (RESTART)

O último subgrupo de instruções deste grupo contém as instruções especiais RST, ou RESTART (reinício). Estas instruções, muito úteis por sinal, são na verdade instruções CALL, que diferem das CALL verdadeiras apenas por não requererem a especificação de endereço.

Mnemônica	*Código hexa*	*Comentário*
RST 0000	C7	CALL 0000
RST 0008	CF	CALL 0008
RST 0010	D7	CALL 0010
RST 0018	DF	CALL 0018
RST 0020	E7	CALL 0020
RST 0028	EF	CALL 0028
RST 0030	F7	CALL 0030
RST 0038	FF	CALL 0038

No TK 90X, as oito instruções RST referem-se a endereços de entrada em sub-rotinas em linguagem de máquina armazenados na memória ROM do micro, e serão discutidas mais tarde.

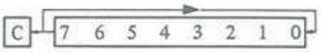
GRUPO 13 – INSTRUÇÕES DE ROTAÇÃO

No conjunto de instruções do Z80, existe um grande número de instruções para rotação de bits de determinados bytes. São instruções quase sempre muito úteis.

Rodar um byte para a esquerda tem o efeito de duplicar o seu valor, sem perder o conteúdo do bit mais significativo, enquanto, rodar o byte à direita significa reduzir à metade o seu valor.

O diagrama a seguir mostra a variedade de rotações que são possíveis.

RLC & RCLA
rotate left with carry
rotação à esquerda com transporte



Bit 7 vai para . Bandeira C e para Bit 0.

RL & RLA
rotate left
rotação à esquerda

Bit 7 vai para Bandeira C, que vai para Bit 0.

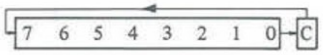
SLA
Shift left
Deslocamento à esquerda

Bit 0 é resetado. Bit 7 vai para Bandeira C.

RRC & RRCA
rotate right with carry
rotação à direita com transporte

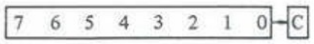
Bit 0 vai para Bandeira C e para Bit 7.

RR & RRA
rotate right
rotação à direita



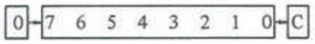
Bit 0 vai para Bandeira C. Bandeira C vai para Bit 7.

SRA
Shift right
Deslocamento à direita



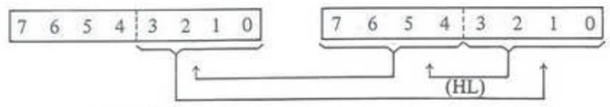
Bit 0 vai para Bandeira C.

SRL
Logical shift right
Deslocamento lógico à direita



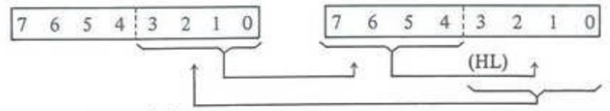
Bit 0 vai para Bandeira C. Bit 7 é resetado.

RLD
rotate digit left and right between Accumulator and location (hl)



Acumulador

RRD
rotate digit right and left between Accumulator and location (HL)



Acumulador

A tabela abaixo mostra as instruções deste grupo:

	RLC	RL	SLA	RRC	RR	SRA	SRL
A	CB07	CB17	CB27	CB0F	CB1F	CB2F	CB3F
H	CB04	CB14	CB24	CB0C	CB1C	CB2C	CB3C
L	CB05	CB15	CB25	CB0D	CB1D	CB2D	CB3D
B	CB00	CB10	CB20	CB08	CB18	CB28	CB38
C	CB01	CB11	CB21	CB09	CB19	CB29	CB39
D	CB02	CB12	CB	CB	CB1A	CB2A	CB3A
E	CB03	CB13	CB23	CB0B	CB1B	CB2B	CB3B
(HL)	CB06	CB16	CB26	CB0E	CB1E	CB2E	CB3E
(IX + d)	DDCB d06	DDCB d16	DDCB d26	DDCB d0E	DDCB d1E	DDCB d2E	DDCB d3E
(IY + d)	FDCB d06	FDCB d16	FDCB d26	FDCB d0E	FDCB d1E	FDCB d2E	FDCB d3E

Existem ainda quatro instruções de um byte, para rodar o acumulador, e duas instruções de manipulação de "nibbles", ou seja, quatro bits de um byte dividido ao meio, do bit 0 ao bit 3 e do bit 4 ao bit 7, respectivamente, nibble menos significativo e nibble mais significativo.

Mnemônica	*Código hexa*
RLCA	07
RLA	17
RRCA	0F
RRA	1F
RRD	ED67
RLD	ED6F

Quanto às bandeiras indicadoras:

1. Todas as instruções, exceto RLD e RRD afetam a bandeira indicadora de transporte.

2. Todas as instruções, exceto as quatro instruções de bytes simples, afetam respectivamente as bandeiras zero, de sinal e de paridade/excesso.

GRUPO 14 – INSTRUÇÕES DE MANIPULAÇÃO DE BITS

Existem, no conjunto das instruções em código de máquina do microprocessador Z80, instruções que permitem ao programador testar, setar ou resetar um bit especificado de um byte qualquer armazenado em um registro ou em um endereço da memória. Estes três tipos de instrução serão vistos a seguir.

SUBGRUPO 1 – INSTRUÇÕES BIT

Estas instruções permitem que o programador determine o estado de um bit específico.

Uma instrução BIT seta a bandeira zero, se o bit testado está resetado (vale 0), e vice-versa.

SUBGRUPO 2 – INSTRUÇÕES SET

Estas instruções permitem ao programador setar determinado bit de um byte. Nenhuma bandeira é afetada.

SUBGRUPO 3 – INSTRUÇÕES RES (RESET)

Inversamente ao grupo anterior, estas instruções permitem que o programador resete um determinado bit de um byte. Também não afetam nenhuma das bandeiras indicadoras de estado.

As instruções destes três subgrupos estão na tabela a seguir.

		BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
Reg.	BIT	47	4F	57	5F	67	6F	77	7F
A	RES	87	8F	97	9F	A7	AF	B7	BF
CB ...	SET	C7	CF	D7	DF	E7	EF	F7	FF
Reg.	BIT	44	4C	54	5C	64	6C	74	7C
H	RES	84	8C	94	9C	A4	AC	B4	BC
CB ...	SET	C4	CC	D4	DC	E4	EC	F4	FC
Reg.	BIT	45	4D	55	5D	65	6D	75	7D
L	RES	85	8D	95	9D	A5	AD	B5	BD
CB ...	SET	C5	CD	D5	DD	E5	ED	F5	FD
Reg.	BIT	40	48	50	58	60	68	70	78
B	RES	80	88	90	98	A0	A8	B0	B8
CB ...	SET	C0	C8	D0	D8	E0	E8	F0	F8
Reg.	BIT	41	49	51	59	61	69	71	79
C	RES	81	89	91	99	A1	A9	B1	B9
CB ...	SET	C1	C9	D1	D9	E1	E9	F1	F9
Reg.	BIT	42	4A	52	5A	62	6A	72	7A
D	RES	82	8A	92	9A	A2	AA	B2	BA
CB ...	SET	C2	CA	D2	DA	E2	EA	F2	FA
Reg.	BIT	43	4B	53	5B	63	6B	73	7B
E	RES	83	8B	93	9B	A3	AB	B3	BB
CB ...	SET	C3	CB	D3	DB	E3	EB	F3	FB
(HL)	BIT	46	4E	56	5E	66	6E	76	7E
CB ...	RES	86	8E	96	9E	A6	AE	B6	BE
	SET	C6	CE	D6	DE	E6	EE	F6	FE

GRUPO 15 – INSTRUÇÕES DE MANIPULAÇÃO DE BLOCOS

São, no total, oito instruções de manipulação de blocos, do conjunto de instruções do microprocessador Z80. Estas instruções são muito úteis e muito interessantes, pois permitem ao programador mover um bloco de dados de uma área da memória para outra, ou procurar uma área da memória.

Para mover um bloco de dados, o endereço base deve estar armazenado no par de registros HL, o endereço de destino deve estar armazenado no par de registros DE, e o número de bytes do bloco, ou seja, o seu comprimento deve estar armazenado no par de registros BC.

Para procurar na memória a primeira ocorrência de um determinado valor, o endereço base também deve estar armazenado no par de registros HL, o número de bytes da área de pesquisa deve estar no par de registros BC, e o registro A deve armazenar uma cópia do valor a ser encontrado.

As instruções deste grupo são:

	Mnemônica	*Código hexa*	*Comentários*
AUTO-MÁTICAS	LDIR	EDB0	Mover bloco – incrementa
	LDDR	EDB8	Mover bloco – decrementa
	CPIR	EDB1	Procura bloco – incrementa
	CPDR	EDB9	Procura bloco – decrementa
NÃO AUTO-MÁTICAS	LDI	EDA0	Mover byte – incrementa
	LDD	EDA8	Mover byte – decrementa
	CPI	EDA1	Compara byte – incrementa
	CPD	EDA9	Compara byte – decrementa

Como você deve ter notado, existem instruções automáticas e instruções não-automáticas. As automáticas são geralmente mais úteis, mais rápidas e, portanto, mais utilizadas.

Vamos analisar cada instrução detalhadamente.

INSTRUÇÕES AUTOMÁTICAS

1. *LDIR*: load location (DE) with location (HL), increment DE, HL, decrement BC and repeat until BC = 0. Esse é o nome em inglês. Traduzindo, essa instrução, LDIR, vai mover um bloco de dados cujo endereço fonte está armazenado no par de registros HL, para uma área da memória, cujo endereço inicial está armazenado em DE (DEstination = destino...), e o comprimento desse bloco está armazenado no par de registros BC.

Quando em operação, um simples byte é transferido de HL, ou seja, para onde HL está apontando, para DE, o destino. O valor armazenado no par BC é então decrementado, e os valores armazenados nos pares HL e DE são incrementados. Enquanto o contador de bytes BC não chegar a zero, o processo será repetido.

Esta instrução reseta a bandeira de paridade/excesso.

2. *LDDR*: load location (DE) with location (HL), decrement DE, HL and BC and repeat until BC = 0.

Repare nos títulos desta e da instrução anterior e note qual a única diferença entre elas. Enquanto a primeira incrementa DE e HL, esta decrementa estes dois pares de registros, juntamente com o contador de bytes, BC.

Portanto, esta instrução requer como endereço base do bloco a sua última locação.

CPIR: Compare location (HL) and Accumulator, increment HL, decrement BC and repeat until BC = 0.

Esta instrução procura, em uma área específica da memória, pela ocorrência de um determinado valor, pela primeira vez. O par de registros HL deve armazenar o endereço base; o par BC deve armazenar o número de bytes a serem pesquisados; o acumulador (registro A) deve armazenar o valor, em particular, a ser pesquisado.

Em operação, o byte armazenado no par HL é comparado com aquele armazenado no registro A. Se a comparação não for verdadeira, então a instrução decrementa o contador de bytes e incrementa o endereço base mantido no par HL para proceder à próxima comparação.

A operação continua até uma delas ter resultado verdadeiro, ou seja, o conteúdo do endereço apontado por HL ser igual ao conteúdo do acumulador, ou então, se não acontecer a verdade, a operação termina quando BC atingir o valor 0. Nesse caso, as bandeiras zero e de paridade/excesso serão resetadas.

CPDR: Compare location (HL) and accumulator, decrement HL and BC repeat until BC = 0.

A operação desenvolvida por esta instrução é similar à anterior, com a única diferença que o bloco de dados é pesquisado a partir do seu último endereço.

Instruções não-automáticas

1. *LDI*: Load location (DE) with location (HL), increment DE, HL, decrement BC.

A execução desta instrução faz com que um byte simples armazenado no endereço apontado por (HL) seja movido, ou transferido para o endereço apontado por (DE). O valor armazenado no par BC é decrementado. A bandeira de paridade/excesso será setada, a menos que o par BC assuma o valor 0. Os valores nos pares HL e DE são incrementados.

2. *LDD*: Load location (DE) with location (HL), decrement DE, HL and BC.

A única diferença desta instrução para a anterior é que esta decrementa os pares de registros DE e HL, em vez de incrementá-los.

3. *CPI*: Compare location (HL) and accumulator, increment HL and decrement BC.

A execução desta instrução vai fazer com que o byte endereçado pelo par HL seja copiado no microprocessador e armazenado, enquanto o valor em HL é incrementado e o valor em BC é decrementado. O valor armazenado no microprocessador é então comparado com o valor do acumulador. Se o resultado da comparação for verdadeiro, então a bandeira zero é setada, e, de outra forma, ela é resetada. A bandeira de sinal é resetada e a bandeira de paridade/excesso também é resetada, até que o valor em BC atinja zero, quando ela passa a setada.

4. *CPD*: Compare location (HL) and accumulator, decrement HL and BC.

Esta instrução é similar à instrução CPI, exceto que o valor armazenado no par de registros HL é decrementado.

GRUPO 16 – INSTRUÇÕES DE ENTRADA E SAÍDA (INPUT/OUTPUT)

Estas instruções de entrada ou de saída de dados (bytes) permitem que o programador receba dados de uma fonte externa (IN) ou envie dados para um dispositivo externo (OUT).

São instruções simples, não-automáticas e automáticas.

Em todos os casos de instruções de entrada/saída, os dados manipulados são bytes de 8 bits enviados em paralelo.

Quando o microprocessador está executando uma instrução IN, ele pega o byte determinado nas vias de dados e copia-o em um determinado registro. A via de controle $\overline{IOQR}$ é ativada, bem como a linha $\overline{RD}$ durante a sua execução.

Quando está executando uma instrução OUT, o microprocessador coloca uma cópia do valor armazenado em um registro especificado nas vias de dados, de onde ele será coletado por um dispositivo externo. As vias $\overline{IORQ}$ e $\overline{WR}$ se tornarão ativas durante sua execução.

Em adição ao estado das vias $\overline{RD}$, $\overline{WR}$ e $\overline{IORQ}$, um dispositivo externo também será ativado pelo uso de um endereço apropriado colocado nas vias de endereçamento, durante a execução de instruções tipo IN ou OUT. Este endereço é denominado "porta de endereço" que no caso do Z80 é um endereço de 16 bits.

As instruções deste grupo são:

Mnemônica	*Código hexa*	*I/O Reg.*	*Alto E.P.*	*Baixo E.P.*
IN A, (dd)	DBdd	A	A	dd
IN A, (C)	ED78	A	B	C
IN H, (C)	ED60	H	B	C
IN L, (C)	ED68	L	B	C
IN B, (C)	ED40	B	B	C
IN C, (C)	ED48	C	B	C
IN D, (C)	ED50	D	B	C
IN E, (C)	ED58	E	B	C
OUT (dd), A	D3dd	A	A	dd
OUT (C), A	ED79	A	B	C
OUT (C), H	ED61	H	B	C
OUT (C), L	ED69	L	B	C
OUT (C), B	ED41	B	B	C
OUT (C), C	ED49	C	B	C
OUT (C), D	ED51	D	B	C
OUT (C), E	ED59	E	B	C

Nessa listagem, Alto E.P. quer dizer byte mais significativo do endereço da porta, e Baixo E.P. quer dizer byte menos significativo do endereço da porta.

As instruções automáticas e não-automáticas são:

Mnemônica	*Código hexa*	*Comentários*
INI	EDA2	Não-automática – Incrementa
INIR	EDB2	Automática – Incrementa
IND	EDAA	Não-automática – Decrementa
INDR	EDBA	Automática – Decrementa
OUTI	EDA3	Não-automática – Incrementa
OUTIR	EDB3	Automática – Incrementa
OUTD	EDAB	Não-automática – Decrementa
OUTDR	EDBB	Aumotática – Decrementa

As mnemônicas querem dizer:

INI carrega locação apontada pelo par HL, com entrada da porta (C), incrementa HL e decrementa B.

INIR carrega locação apontada pelo par HL, com entrada da porta (C), incrementa HL e decrementa B, repete até que B = 0.

IND carrega locação apontada pelo par HL, com entrada da porta (C), decrementa HL e B.

INDR carrega locação apontada pelo par HL, com entrada da porta (C), decrementa HL e decrementa B, até que B = 0.

OUTI carrega porta de saída (C) com locação apontada pelo par (HL), incrementa HL e decrementa B.

OUTIR carrega porta de saída (C) com locação apontada pelo par (HL), incrementa HL, decrementa B e repete até que B = 0.

OUTD carrega porta de saída (C) com locação apontada pelo par (HL), decrementa HL e B.

OUTDR carrega porta de saída (C) com locação apontada pelo par (HL), decrementa HL e B, e repete até que B = 0.

GRUPO 17 – INSTRUÇÕES DE INTERRUPÇÃO

Existem, ao todo, sete instruções que permitem que o programador manipule o sistema de interrupções do microprocessador Z80. São elas:

Mnemônica	*Código hexa*
E1	FB
D1	F3
IM 0	ED46
IM 1	ED56
IM 2	ED5E
RETI	ED4D
RETN	ED45

Vamos analisar cada uma destas instruções.

EI (*enable interrupt*): Quando ligamos o microcomputador, um sistema de interrupção "mascarada" é habilitado para interromper o funcionamento do microprocessador Z80. Em outros termos, quando ligamos o microcomputador, o seu microprocessador imediatamente começa a trabalhar, executando as rotinas do seu sistema monitor. Nestas rotinas, necessariamente deve haver um sistema de interrupção do microprocessador, para que ele possa reconhecer alguma tecla que pressionemos, ou para que possamos dar-lhe algum comando em sua linguagem residente, qual seja, a Basic.

No TK 90X, o sistema de interrupção mascarada é usado para um relógio de tempo real, e para a rotina de reconhecimento de teclado, para saber se e qual tecla foi pressionada, e a interrupção é gerada a cada 1/60 segundos.

DI (*disable interrupt*): Em qualquer ponto de qualquer rotina em linguagem de máquina, o programador pode decidir "desligar" o sistema de interrupção mascarada, através desta instrução DI, o que torna o microprocessador insensível a qualquer sinal da linha INT. Através da utilização desta instrução, em alguns casos, chegamos a ganhar mais de 50% de tempo de processamento, já que não existe mais o reconhecimento do teclado.

IM0: São três modos de interrupção. Este modo é selecionado automaticamente pelo microprocessador quando ligamos o microcomputador pela primeira vez, ou também pela execução desta própria instrução. Mas, este modo de interrupção não é utilizado pelo TK 90X.

IM1: Este modo de interrupção é selecionado somente pela execução desta instrução, e é o modo utilizado pelo TK 90X. O programa monitor contido nos 16K da ROM do microcomputador possui esta instrução como parte da rotina de inicialização.

Neste modo, a instrução RESTART 0038h será sempre selecionada após receber um sinal da linha INT, o que significa que o sistema de interrupção mascarado foi habilitado. No TK 90X, a rotina em código de máquina, com início em 0038, atualiza o relógio de tempo real e faz o esquadrinhamento do teclado (reconhecimento).

IM2: Este modo não é utilizado pelo TK 90X, mas, dos três modos de interrupção possíveis é o mais poderoso. Neste modo, um dispositivo periférico pode indicar ao microprocessador qual das 128 diferentes sub-rotinas deve ser executada após receber a interrupção mascarada. O conteúdo do registro I e o byte fornecido pelo dispositivo periférico são usados juntos para formar um endereço de 16 bits, que é utilizado para endereçar uma tabela de vetores, previamente preparada na memória.

RETI: Esta instrução é um "retorno" especial, para ser empregado em rotinas de interrupção mascaradas. O efeito desta instrução é o de retornar com a mesma interrupção mascarada depois de ter sido interrompido o processamento normal.

RETN: Esta instrução é similar à anterior, mas é aplicada no fim de uma rotina de interrupção não mascarada.

GRUPO 18 – INSTRUÇÕES DIVERSAS

Existem ainda seis instruções que não foram mencionadas. São elas:

Mnemônica	*Código hexa*
CPL	2F
NEG	ED44
SCF	37
CCF	3F
HALT	76
DAA	27

As mnemônicas significam:

CPL	–	Complementar acumulador
NEG	–	Negativo do acumulador (complemento de 2)
SCF	–	Seta bandeira de transporte
CCF	–	Complementa bandeira de transporte
HALT	–	Aguarde por uma interrupção ou por um resetar
DAA	–	Ajuste decimal do acumulador

Vamos analisá-las:

CPL: Esta é uma instrução simples que complementa o acumulador, ou seja, seta o bit que está resetado, e vice-versa. Esta operação é chamada complemento de 1. Não afeta bandeiras.

NEG: Esta instrução calcula o complemento de 2 do acumulador. As bandeiras de sinal e a zero dependem do resultado para serem alteradas. A bandeira de transporte será resetada se o valor original for zero, de outra forma será setada, e a bandeira de paridade/excesso será setada se o valor original for 80h.

SCF: Seta a bandeira indicadora de transporte.

CCF: Complementa a bandeira indicadora de transporte.

HALT: Esta é uma instrução especial que faz com que o microprocessador pare o seu trabalho até que ocorra uma interrupção. No TK 90X, as únicas interrupções que podem ocorrer são as mascaradas. O comando PAUSE usa deste artifício para contar 1/60 segundo.

DAA: Esta é a instrução que faz o ajuste decimal do acumulador. Em aritmética binária BCD (*binary coded arithmetic*), os algarismos de 0 a 9 são representados pelos "nibbles binários" 0000-1001, e os nibbles 1010-1111 não são utilizados. Portanto:

– o byte 0000 0000 representa o número 0;
– o byte 0011 1001 representa o número 39.

= 3 = 9

Esta instrução converte, portanto, bytes em sua forma binária absoluta para a forma BCD.

A bandeira indicadora de sinal e a bandeira zero são afetadas pelo resultado, e a bandeira de paridade/excesso será setada se houver paridade par. O

efeito na bandeira indicadora de transporte vai depender se houve excesso nas adições ou subtrações na forma BCD.

E assim nós encerramos este capítulo chato que trata de todas as instruções do microprocessador Z80. Não é à toa que ele é considerado o mais complicado dos microprocessadores de 8 bits.

Capítulo **11**

As rotinas da ROM de 16K

O TK 90X tem uma ROM de 16K que está dividida em:

1. O sistema operacional
2. O interpretador Basic
3. O conjunto de 96 caracteres

Esta ROM ocupa os endereços de 0 a 16383, ou 0000 a 3FFFh, e não pode ser movida desta área, na versão padrão da máquina. A instrução em código de máquina no endereço 0 é a primeira a ser executada quando ligamos o micro pela primeira vez.

As sub-rotinas do programa monitor podem ser utilizadas pelo programador em seus próprios programas em linguagem de máquina, fazendo com que estes sejam relativamente mais curtos, por não repetirem certos processamentos, e conseqüentemente mais rápidos.

É muito instrutivo para os aficionados estudarem o programa monitor, para perceberem como um programa tão grande pode ser estruturado. Isso você faz através de um programa ferramenta denominado "Monitor & Disassembler", que mostra as mnemônicas dos endereços e seus respectivos códigos.

Vamos agora ver alguns detalhes da ROM.

SISTEMA OPERACIONAL – ROTINAS

1. **Rotina de inicialização** É a primeira rotina a ser executada, quando ligamos o micro. Ela ocupa os endereços 0000 a 0007h, e 11CBh a 12A1h.

Um dos propósitos principais desta rotina é o de checar o quanto de memória RAM está disponível para o usuário e acertar os devidos valores das variáveis de sistema (valores padrão).

2. **Rotina de execução principal** Esta rotina ocupa os endereços de 12A2h a 15AEh e é a dominante de todo o programa monitor, pois é ela que chama, se necessário, a rotina do comando LIST, a rotina do Editor, a rotina de checar sintaxe de linhas em Basic. No caso de se usar o micro em modo imediato, é chamada então a rotina de rodar linha (LINE RUN) para interpretá-la e executá-la. Se houver erro, então é feita uma referência a uma tabela de mensagens de erros, que ocupa os endereços 1391h a 1536h.

3. **Rotina do Editor** Ocupa os endereços 0F2Ch a 10A7h.

Esta rotina permite que o usuário construa uma linha em Basic, aparentemente na parte inferior da tela. Digo aparentemente porque na realidade a linha é formada na área de edição e então copiada, com símbolos expandidos, para a área da tela.

Uma única entrada para esta rotina é feita pela chamada da rotina Entrada do Teclado (KEYBOARD INPUT).

4. **Entrada do teclado (KEYBOARD INPUT)** Esta rotina ocupa os endereços 10A8h a 111CH.

Ela pega o código da última tecla pressionada pela leitura da variável de sistema KEYPRS, de endereço 23560d.

5. **Teclado** O esquadrinhamento do teclado é uma interrupção que ocorre a cada 1/60 segundos. São cinco sub-rotinas separadas envolvidas pela execução da rotina principal de teclado, que ocupa os endereços 02BFh a 030Fh.

O esquadrinhamento do teclado é feito pela sub-rotina KEY SCAN, nos endereços 028Eh a 02BEh. Esta sub-rotina retorna o valor do código da tecla no par de registros DE, para que outras rotinas dele se utilizem.

6. **Saída de dados (PRINT OUTPUT)** A rotina de impressão de saída de dados na tela que ocupa os endereços 09F4h a 0D4Ch é uma das mais importantes do programa monitor.

Esta rotina é em efeito a rotina chamada pela instrução em código de máquina RST 0010h. O endereço 09F4h é obtido através da área de informação de canais.

A instrução RST 0010h, ou a rotina PRINT OUTPUT vai permitir que o caractere, cujo código está armazenado no registro A, seja impresso ou na tela da TV, ou na impressora standard. A rotina testa uma determinada bandeira indicadora de estado de uma variável de sistema, para saber qual caminho de saída deve ser adotado. Esta é uma rotina muito poderosa, visto que manipula tanto códigos de caracteres, quanto códigos de controle.

No caso da saída de dados ser pela TV, a posição de impressão é coletada nas variáveis de sistema apropriadas, utilizadas e armazenadas novamente no lugar de onde foram retiradas. A posição de impressão indica o número da linha e o número da coluna da área a ser utilizada, bem como o endereço correspondente no arquivo de imagem.

Esta rotina possui sub-rotinas que são chamadas, se necessário, para transferir os 64 bits de um caractere da área de conjunto de caracteres para a locação requisitada no arquivo de imagens.

Outra sub-rotina altera o byte dos atributos daquela área de caractere, de acordo com o valor armazenado nas variáveis de sistema apropriadas.

INTERPRETADOR BASIC

A rotina de interpretação de comandos do programa monitor é chamada tanto para checar sintaxe quanto para execução de linhas.

As diferentes partes do interpretador Basic são:

1. **Tabela de comandos** Esta tabela é encontrada nos endereços 1A48h a 1B16h.

No TK 90X são 51 comandos Basic e esta tabela contém detalhes de cada comando, os caracteres separadores e os endereços das rotinas de execução desses comandos.

2. **Rotina controladora** Contida nos endereços 1B17h a 1C00h, esta rotina contém as instruções de controle que permitem que o interpretador Basic passe de uma declaração Basic para outra, conforme requerido pelo programa.

3. **Rotinas das classes de comandos** Estas rotinas, nos endereços de 1C01h a 1CDDh são principalmente concebidas para a análise dos parâmetros que seguem os comandos em Basic.

4. **Rotinas dos comandos** Muitas das rotinas de execução dos comandos em Basic estão contidas nos endereços 1CDEh a 24FAh. Existe uma rotina de execução para cada um dos comandos, e a sua execução é a essência da interpretação Basic.

5. **Avaliador de expressões** Esta rotina ocupa os endereços 24FBh a 28B1h.

O resultado obtido pela avaliação de uma expressão qualquer tanto pode ser numérico, quanto alfanumérico. Um resultado numérico será retornado pelo avaliador de expressões como um número de 5 bytes em representação de ponto flutuante, colocado no topo da pilha do calculador. No caso de um resultado alfanumérico os 5 bytes vão representar um conjunto de parâmetros que descrevem a cadeia de caracteres.

6. **Rotinas de manipulação de variáveis** Este conjunto de rotinas ocupa a área da ROM desde o endereço 2813h até o endereço 2ACBh.

Estas rotinas retornam o valor corrente dos parâmetros de uma dada variável armazenada na área de variáveis da RAM.

A CALCULADORA

Esta grande e variada, bem como, complicada rotina ocupa os endereços 2F9Bh a 386Dh. É normalmente chamada pela instrução em linguagem de máquina RST 0028, que atua como um salto indireto para o endereço 335Bh. No total são 66 sub-rotinas, onde cada uma desempenha a sua função. A chamada destas sub-rotinas normalmente não se faz através da instrução em linguagem de máquina CALL, mas sim pelo uso de valores hexadecimais literais entre 00h e 41h.

Por exemplo, o valor literal 04 equivale à multiplicação; o valor literal 17 equivale à concatenação de cadeias de caracteres.

Estes valores literais são incluídos em uma rotina em linguagem de máquina, como bytes de dados (DEFB's = defined bytes é como você vai encontrar em rotinas de linguagem de máquina) que seguem a instrução RST 0028. O byte final é sempre 38, que desenvolve uma operação de fim de cálculo, e atua como uma saída, um retorno da calculadora.

AS DIFERENTES PARTES DO PROGRAMA MONITOR

A seguir, mostrarei alguns endereços e áreas das principais rotinas do programa monitor, na medida em que eles ocorrem na ROM.

0000-0007 – RST 0000h – início – equivale a RAND USR 0, que desabilita interrupções mascaradas, limpa o registro A, carrega o par de registros DE com FFFFh e salta para o endereço 11CBh.

0008-000F – RST 0008h – rotina de erros, que traz a mensagem apropriada quando ocorreu um erro.

0010-0012 – RST 0010h – rotina de saída de dados.

0018-0024 – RST 0018h e RST 0020 – procura pelo caractere em uso.

0028-0029 – RST 0028h – chamada calculadora.

0030-0037 – RST 0030 – rotina para liberar área de trabalho da memória.

0038-0052 – rotina de interrupção mascarada.

0066-0072 – rotina de interrupção não mascarada.

0095-0204 – tabela de símbolos.

0205-028D – tabela de teclas; no total são 6 tabelas, uma para cada modo permitido.

028E-02BE – sub-rotina de esquadrinhamento do teclado.

02BF-03B4 – sub-rotinas do teclado.

03B5-03F7 – sub-rotina de som.

03F8-046D – rotina do comando SOUND.

04C2-09F3 – rotinas dos comandos SAVE, LOAD, VERIFY e MERGE.

09F4-0D4C – rotina de saída de dados.

0D4D-0D6A – sub-rotina de conjuntos de cores temporárias.

06DB-0EAB – rotina do comando CLS.

0EAC-0F2B – rotinas da impressora (comandos COPY, LLIST e LPRINT).

0F2C-10A7 – editor.

10A8-111C – sub-rotina de entrada do teclado.

11B7-11CA – rotina do comando NEW.

11CB-12A1 – rotina de inicialização.

12A2-15AE – rotina de execução principal.

15AF-15C5 – tabela de dados de canais.

15C6-15D2 – tabela de dados de fluxo.

15D4-1651 – rotina de acesso aos canais.

1652-16E4 – sub-rotinas diversas.

16DB-16E5 – sub-rotinas de indexação de tabelas.

16E6-1792 – rotinas dos comandos OPEN e CLOSE.

1795-1A47 – rotinas de listagem.

1A48-1B16 – duas tabelas de comandos. A primeira é uma tabela de comandos que indexa, a segunda, uma tabela de parâmetros.

1B17-1C00 – rotinas de controle do interpretador Basic.

1C01-1C0C – tabela de classes de comandos.

1C0D-1CDD – rotinas de classes de comandos.

1CDE-24FA – rotinas dos comandos.

24FB-28B1 – avaliador de expressões.

2AFF-2BF0 – rotina do comando LET.

2C02-2C87 – rotina do comando DIM.

2C88-2F9A – diversas rotinas aritméticas.

2F9B-386D – sub-rotina calculadora.

386E-3CFF – conjunto de caracteres.

Capítulo **12**

Como utilizar rotinas do programa monitor

O objetivo deste capítulo é o de mostrar como se pode escrever rotinas em linguagem de máquina de uma maneira relativamente fácil, através da utilização das diversas sub-rotinas que estão sempre disponíveis para o usuário, no programa monitor.

Você pode fazer uso de um programa Basic carregador de códigos de máquina, que lhe permite digitar os códigos hexa da sua listagem mnemônicas, para que ele automaticamente converta esses valores em números decimais (bytes) e armazene-os nos endereços que você determinar.

Eis a listagem de um carregador hexa muito simples, mas que nem por isso deixa de ser útil.

```
   1 REM Carregador de codigos h
exa
  10 POKE 23658,8
  20 INPUT "Endereco inicial ";e
nd: LET inicio=end
  30 INPUT "Entre com o valor em
 hexa ";h$
  32 IF h$(1)="#" THEN LET end=e
nd-1: GO TO 30
  34 IF h$(1)="$" THEN STOP
  36 IF h$(1)="l" THEN INPUT "No
me do arquivo ";y$: SAVE y$CODE
inicio,(end-inicio)
  40 PRINT end;TAB 10;h$;TAB 20;
  50 LET byte=16*(CODE h$(1)-48-
```

```
(7 AND h$(1)>"9"))+(CODE h$(2)-4
8-(7 AND h$(2)>"9"))
  60 PRINT byte
  70 POKE end,byte
  80 LET end=end+1
  90 GO TO 30
```

Vamos ver então algumas rotinas de execução de comandos:

1. **Rotina do comando SOUND** Em Basic, o comando SOUND tem o formato:

SOUND duração, tom

Onde a duração deve ser um número positivo menor que 10 e o tom deve ser um número positivo ou negativo, porém dentro da faixa de abrangência do comando, em função do DO central.

Em linguagem de máquina existem 2 maneiras de se produzir um SOUND. A primeira é chamando a sub-rotina SOUNDER, com os valores apropriados armazenados nos pares de registros DE e HL; a segunda é chamando a rotina de comando SOUND com os valores para duração e tom armazenados na pilha do calculador.

Qualquer rotina que você elabore em linguagem de máquina pode ser armazenada, a princípio, em qualquer lugar da memória.

A princípio porque um programa Basic que execute aquela rotina ocupa uma certa área, e seus códigos de máquina devem começar após o término do programa Basic.

Através das variáveis de sistema apropriadas, você pode determinar onde começa a área de memória desocupada.

A fim de padronizarmos nossas rotinas deste livro, eu sugiro como endereço inicial da memória, para armazenarmos as rotinas em linguagem de máquina, a locação decimal 65000, que vamos converter para hexadecimal:

```
 65000 | 256
-512   | 253
 1380
-1280
 1000
-768
 232
```

Portanto, 65000 = 253 × 256 + 232.

Consultando o Apêndice A deste livro, temos para

253 = FD e
232 = E8

Portanto, 65000d = FDE8h

Conferindo, através do processo inverso:

```
F  D  E  8
|  |  |  ↓
|  |  |  8 × 16^0 =  8 × 1   =     8
|  |  ↓  14 × 16^1 = 14 × 16  =   224
|  ↓  13 × 16^2    = 13 × 256 =  3328 +
↓  15 × 16^3       = 15 × 4096 = 61440
                                 65000 e confere!
```

Podemos converter de outra forma:

```
 65000 | 256
-512   | 253
 1380
-1280
 1000
-768
 232

 232 | 1
-16  | 14
 72
-64
 8
```

Portanto 1ª casa = 8d = 8h
2ª casa = 14d = Eh

```
 253 | 16
-16  |
 93
-80
 13
```

3ª casa = 13d = Dh
4ª casa = 15d = Fh

Ou seja, FDE8h!

Guarde muito bem esses métodos de conversão, pois às vezes você está no meio da digitação de uma rotina qualquer e esqueceu de converter um só valor, e não tem condições de carregar um programa que faça a conversão para você.

Vamos, então, chamar este endereço FDE8 de END, e voltemos ao SOUND.

1º método:

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	110501	LD DE, 0105	Duração 1 segundo
END + 3	216606	LD HL, 0666	Tom DO central
END + 6	CDB503	CALL SOUNDER	Aciona SOUND
END + 9	C9	RET	Retorna à Basic

Os valores para DE e HL são assim calculados:

– considere uma nota de freqüência f; por exemplo, DO central tem a freqüência de 261.63 hz;

– a duração requerida para um período p é calculada por "f x p". Este valor é colocado no par DE;

– o valor a ser armazenado no par HL é dado por:

437.500/f – 30125.

Note que com este método não existe a limitação de 10 segundos de duração.

2º método:

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	3E01	LD A, 01	Duração de 1 segundo.
END + 2	CD282D	CALL STACK A	Coloca na pilha da calculadora.
END + 5	3E00	LD A, 00	Tom DO central.
END + 7	CD282D	CALL STACK A	Coloca na pilha da calculadora.
END + 10	CDF803	CALL SOUND	Aciona SOUND.
END + 11	C9	RET	Retorna à Basic.

2. **Rotinas de SAVE e LOAD** As sub-rotinas SAVE bytes e LOAD bytes podem ser chamadas em linguagem de máquina de uma maneira muito particular.

Em ambos os casos, o par de registros IX deve armazenar o endereço destino, e o par de registros DE, o contador de bytes.

O registro A deve armazenar FF para mostrar que um bloco de dados está sendo movimentado.

Finalmente, para LOAD, a bandeira indicadora de transporte deve ser setada.

A rotina a seguir executa o comando SAVE bytes.

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	3EFF	LD A, FF	Significa bloco de dados.
END + 2	DD21dd	LD IX, dd (início)	Carrega par IX c/endereço destino.
END + 6	11xx	LD DE, xx (comprim)	Carrega par DE c/quantidade de bytes.
END + 9	CDC204	CALL SAVE byte	Executa SAVE.
END + 12	C9	RET	Retorna à Basic.

Note que os bytes serão gravados sem o cabeçalho (*header*), que dá as informações sobre o programa, e conseqüentemente este bloco só será carregado por LOAD se o contador de bytes for conhecido.

A rotina para LOAD:

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	37	SCF	Seta bandeira de transporte (será resetado para VERIFY).
END + 1	3EFF	LD A, FF	Significa bloco de dados.
END + 3	DD21dd	LD IX, dd	Carrega par IX.
END + 7	11dd	LD DE, dd	Carrega par DE.
END + 10	CD5605	CALL LOAD bytes	Executa LOAD.
END + 13	C9	RET	Retorna à Basic.

Uma pequena nota sobre o cabeçalho (header)

Ele tem 17 bytes de comprimento, que respectivamente informam:

– Tipo – 1 byte	Contém o tipo de arquivo: 0 = programa Basic 1 = matriz numérica 2 = matriz alfanumérica 3 = códigos de máquina
– Nome – 10 bytes	Contém o nome do arquivo.
– Comprimento – 2 bytes	Contém o comprimento do arquivo.
– Endereço Inicial – 2 bytes	Contém o número da linha para AUTO. RUN ou o endereço inicial dos códigos.
– ? – 2 bytes	?

Aqui já dá para você mesmo escrever um programa copiador ...

3. **Rotinas dos comandos de cores** – no TK 90X todos os bytes de atributos têm o seguinte formato:

BIT 7 – FLASH
BIT 6 – BRIGHT
BIT 5 a 3 – PAPER
BIT 2 a 0 – INK

O comando BORDER e os seis comandos de cores serão discutidos agora.

3.1. **BORDER** – a qualquer hora, a cor da borda pode ser alterada pelo uso da instrução

OUT (FE), A

A rotina mais simples para alterar BORDER:

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	3E01	LD A, "blue"	Cor azul = 1
END + 2	CD9B22	CALL BORD-1	Altera borda (cor)
END + 5	C9	RET	Retorna à Basic

Note que com esta rotina, a cor da tinta passa a ser contrastante em relação à cor da borda.

3.2. **PAPER** – a cor permanente do papel é dada pelos bits 5 a 3 da variável de sistema ATCLR P. A rotina abaixo mostra a alteração de PAPER 0 – PAPER 7 apenas uma vez na qual os bits podem ser alterados sem se alterar os outros valores de atributos.

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	3A8D5C	LD A, (5C8D)	Carrega acumulador com valor da variável de sistema.
END + 3	OF	RRCA	Move os bits 5 a 3
END + 4	OF	RRCA	para bits 2 a 0.
END + 5	OF	RRCA	
END + 6	E6F8	AND F8	Descarta cor antiga.
END + 8	C604	ADD A, "green"	Carrega cor verde.
END + 10	07	RLCA	
END + 11	07	RLCA	Roda o byte três vezes
END + 12	07	RLCA	para a esquerda.
END + 13	328D5C	LD (5C8D), A	Restaura variável de sistema.
END + 16	C9	RET	Retorna à Basic.

3.3. **INK** – a cor da tinta permanente é dada pelos bits 2 a 0 da variável de sistema armazenada no endereço 5C8D. A rotina a seguir mostra como esses bits podem ser alterados para INK 0 a INK 7.

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	3EF8	LD A, F8	Prepara máscara.
END + 2	FDA653	AND (53A6)	Busca bits 7 a 3 da variável de sistema em 53A6.
END + 5	C602	ADD A, red	Tinta vermelha.
END + 7	328D5C	LD (5C8D), A	Restaura variável de sistema.
END + 10	C9	RET	Retorna à Basic.

3.4. **FLASH** – os três estados de FLASH podem ser assim programados:

FLASH 0 – reseta bit 7 de ATCLR P "RES 7, (ATCLR P)"
FLASH 1 – seta bit 7 de ATCLR P "SET 7, (ATCLR P)"
FLASH 8 – seta bit 7 de MASKCLRP "SET 7, (MASKCLRP)"

Similarmente, os três estados de BRIGHT:

BRIGHT 0 – reseta bit 6 de ATCLR P "RES 6, (ATCLR P)"
BRIGHT 1 – seta bit 6 de ATCLR P "SET 6, (ATCLR P)"
BRIGHT 8 – seta bit 6 de MASKCLRP "SET 6, (MASKCLRP)"

Os dois estados de OVER e INVERSE:

OVER 1 – seta bit 1 de SFLAG4 "SET 1, (SFLAG4)"
OVER 0 – reseta bit 1 de SFLAG4 "RES 1, (SFLAG4)"

INVERSE 1 – seta bit 3 de SFLAG4 "SET 3, (SFLAG4)"
INVERSE 0 – reseta bit 3 de SFLAG4 "RES 3, (SLFAG4)"

4. **Rotinas dos comandos CLS e SCROLL** – uma das grandes vantagens na utilização da linguagem de máquina ao invés de Basic é que o usuário não está limitado à sintaxe dos comandos Basic apenas.

No programa monitor do TK 90X existem sub-rotinas para limpar apenas parte da tela, bem como para fazer rolar apenas algumas linhas dela (scroll). Mas, note que estas sub-rotinas somente podem ser utilizadas por programas em linguagem de máquina.

4.1. **CLS** – a operação completa de limpar a tela e atribuir a todos os bytes de atributos os valores permanentes é obtida por:

CALL CLS ou mnemônica CALL 0D6B, mas é importante destacar que o canal "S" (de screen) deve ser aberto antes da chamada dessa sub-rotina. Veja como:

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	3E02	LD A, 02	Abre canal "S".
END + 2	CD0116	CALL CHAN OPEN	
END + 5	CD6B0D	CALL CLS	Limpa a tela.
END + 8	C9	RET	Retorna à Basic.

Outra forma, onde se pode limpar apenas parte da tela é utilizando-se a sub-rotina CL LINE.

Antes de se chamar esta sub-rotina, o registro B deve armazenar um valor na faixa 01 a 18h, onde 18h significa limpar a tela toda (24 linhas), assim como o canal S também deve ser aberto.

A rotina:

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	0617	LD B, 17	17h = 23d limpa 23 linhas, deixando apenas a superior.
END + 2	CD440E	CALL CL LINE	Limpa a tela.
END + 5	C9	RET	Retorna à Basic.

4.2. **SCROLL** – esta é uma sub-rotina muito interessante que permite ao usuário "rolar" somente algumas linhas ou a tela toda.

O registro B é novamente utilizado para armazenar o número de linhas a ser rolado, mas aqui, é o número atual de linhas menos uma, isto é, na faixa de 01 a 17h (um mínimo de duas linhas é requerido por esta sub-rotina).

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	0616	LD B, 16	Deixa de rolar somente a primeira linha.
END + 2	CD000E	CALL CL SCROLL	
END + 5	C9	RET	Retorna à Basic.

5. **Sub-rotinas de saída de dados** – para se utilizar qualquer sub-rotina de impressão de caracteres ou mensagens na tela, é preciso, antes de mais nada, abrir-se o canal "S" (de screen). Como vimos anteriormente, para isso basta:

```
LD A, 02
CALL CHAN OPEN
```

5.1. **Rotina RST 0010** – no TK 90X toda e qualquer impressão de caracteres, ou mensagens na tela é feita usando-se esta instrução. Com o canal S aberto, ela tem o efeito de usar a rotina PRINT OUTPUT, que começa no endereço 09F4, como rotina de saída.

A instrução RST 0010 é muito poderosa e pode ser utilizada na impressão de qualquer caractere, para a alteração da posição corrente de impressão pelo uso de AT e TAB, para impressão de palavras expandidas, bem como nos itens de cores provisórias.

A rotina abaixo mostra esses usos:

Endereço	*Código de máquina*	*Mnemônica*	*Comentários*
END	3E02	LD A, 02	
END + 2	CD0116	CALL CHAN OPEN	Abre canal "S".
END + 5	0618	LD B, 18	
END + 7	CD440E	CALL CL LINE	Limpa toda a tela.
END + 10	3E16	LD A, "AT"	
END + 12	D7	RST 0010	
END + 13	3E05	LD A, 05	
END + 15	D7	RST 0010	Similar a PRINT AT
END + 16	3E00	LD A, 00	5, 0
END + 18	D7	RST 0010	
END + 19	3E41	LD A, "A"	PRINT "A"
END + 21	D7	RST 0010	
END + 22	3E0D	LD A, "ENTER"	PRINT
END + 24	D7	RST 0010	
END + 25	3EF9	LD A, CHR$ 249	PRINT CHR$ 249
END + 27	D7	RST 0010	
END + 28	3E0D	LD A, "ENTER"	PRINT
END + 30	D7	RST 0010	
END + 31	3E11	LD A, "PAPER"	
END + 33	D7	RST 0010	
END + 34	3E02	LDA, "RED"	Similar a PAPER 2
END + 36	D7	RST 0010	
END + 37	3E06	LD A, ","	PRINT,
END + 39	D7	RST 0010	
END + 40	3E42	LD A, "B"	PRINT "B"
END + 42	D7	RST 0010	
END + 43	C9	RET	Retorna à Basic

Note nessa sub-rotina, que é feita uma chamada para a sub-rotina CL LINE, que limpa a tela, mas que isto não leva a posição de impressão para a posição padrão, ou seja, linha 0 e coluna 0 (mas CALL CLS leva).

Quando usando a instrução RST 0010 para imprimir um item de cor, duas chamadas separadas são necessárias, e quando usando "AT" ou "TAB" como delimitadores, três chamadas são necessárias.

Um método alternativo para alterar a posição de impressão é o seguinte:

– carregar o par de registros BC com os devidos valores da nova posição de impressão;

– CALL CL SET & CALL 0DD9, que coloca os valores de posicionamento nas variáveis de sistema HV POS e DFPSRT.

Os valores para o par de registros BC para uma determinada posição de impressão "AT x, y" são, respectivamente, para o registro B, (18 - x)h e para o registro C, (21 - y)h.

Por exemplo:

PRINT AT 5,0 requer:

```
LD BC, 1321
CALL 0DD9
```

5.2. **Impressão de série de caracteres** – uma cadeia de caracteres, para ser impressa, deve ter sempre no par de registros DE o endereço do primeiro caractere dela, e o par de registros BC deve armazenar um valor igual ao seu número total de caracteres.

Na rotina do comando PRINT, a sub-rotina PR STRING é utilizada para imprimir qualquer série de caracteres.

Os detalhes da sub-rotina são:

```
PR STRING equ. 203C
```

NOTA: Esta primeira linha é aquela notação citada anteriormente, que associa o nome determinado ao início de uma sub-rotina específica em linguagem de máquina.

```
PR STRING   LD A, B        Dependendo do comprimento da série
            OR C           de caracteres em questão, o seu valor
                           está em B ou em C.
            DEC            Decrementa o contador.
            RET Z          Se o contador for zero, então fim.
            LD A, (DE)     Carrega acumulador com caractere.
            INC DE         Passa para o próximo caractere.
            RST 0010       Imprime o caractere.
            JR PR STRING   Retorna ao início para outra impressão.
```

Portanto, qualquer série de caracteres pode ser impressa desta outra maneira:

- carregar seu endereço inicial no par de registros DE;
- carregar seu comprimento no par de registros BC;
- CALL PR STRING – CALL 203C.

Como exemplo, digite a rotina abaixo:

Endereço	*Código de máquina*	*Mnemônica*	*Comentários*
END	11 01 10 09 16 0A 05 41 52 51 55 49 54 52 4F 4E 20 49 4E 46 4F 52 4D 41 54 49 43 41 20 4C 54 44 41 20 20 20 20 20 20 41 56 20 42 52 49 47 20 46 41 52 49 41 20 4C 49 4D 41 2C 20 31 35 38 30 20 43 4A 20 32 32		Código da série de caracteres.
END + 62	3E02	LD A, 02	
END + 64	CD0116	CALL CHAN OPEN	Abre canal "S".
END + 67	0618	LD B, 18	
END + 69	CD440E	CALL CL LINE	Limpa a tela.
END + 72	11 END	LD DE, END	Endereço inicial da série de caracteres.
END + 75	014500	LD BC, 0045	Comprimento da série.
END + 78	CD3C20	CALL PR STRING	Imprime a mensagem.
END + 81	C9	RET	Retorna à Basic.

5.3. **Imprimindo números** – a sub-rotina PRINT FP (FP de floating point – número em ponto flutuante) é usada para imprimir em formato decimal qualquer número de formato em ponto flutuante de 5 bytes. Esta sub-rotina considera como operando a última entrada na pilha da calculadora. Após isso, o valor é removido da pilha e abandonado.

A rotina a seguir usa esta sub-rotina para imprimir PI/2 a partir da tabela de constantes da ROM.

Endereço	*Código de máquina*	*Mnemônica*	*Comentários*
END	3E02	LD A, 02	
END + 2	CD0116	CALL CHAN OPEN	Abre canal S.
END + 5	CD6B0D	CALL CLS	Limpa a tela.
END + 8	3E02	LD A, 02	Reabre canal S.
END + 10	CD0116	CALL CHAN OPEN	
END + 13	EF	RST 0028	Chama a calculadora.
END + 14	A3	DEFB A3	Pega a quarta constante.
END + 15	38	DEFB 38	Final de cálculo.
END + 16	CDE32D	CALL PRINT FP	Imprime o resultado.
END + 19	C9	RET	Retorna à Basic.

O uso da linha, em modo direto, RAND USR END tem o mesmo efeito que PRINT PI/2.

Existe outra sub-rotina para imprimir números, só que na faixa decimal, de 0 a 9999, pois esta sub-rotina é empregada para produzir números de linhas de programas em listagens, ou em mensagens da parte inferior da tela. Quando em uso, esta sub-rotina requer que o número em questão esteja armazenado no par de registros BC, na ordem normal, ou seja, "byte alto/byte baixo" (mais significativo/menos significativo).

Por exemplo:

Endereço	*Código de máquina*	*Mnemônica*	*Comentários*
END	3E02	LD A, 02	
END + 2	CD0116	CALL CHAN OPEN	Abre canal S.
END + 5	CD6B0D	CALL CLS	Limpa toda a tela.
END + 8	3E02	LD A, 02	
END + 10	CD0116	CALL CHAN OPEN	Reabre canal S.
END + 13	010F27	LD BC, 270F	Carrega BC com o valor 9999d.
END + 16	CD1B1A	CALL OUT NUM	Imprime o número.
END + 19	C9	RET	Retorna à Basic.

6. **PLOT, DRAW e CIRCLE** – estes três comandos tratam com as pixéis na tela. Para as 22 linhas superiores, são no total 256 x 176 pontos.

As variáveis de sistema LSTPLOT e COORDS armazenam as coordenadas do último ponto plotado.

6.1. **PLOT** – destes três comandos, este é o mais fácil de ser manipulado, pois envolve somente a identificação de um simples bit no arquivo de imagens, e seu subseqüente desenho ou não.

Existem três pontos de entrada na rotina do comando PLOT:

1. **CALL PLOT-CALL 22DC** – esta entrada requer os valores de x e y no topo da pilha da calculadora, sendo o valor de x abaixo do valor de y.

2. **CALL PLOT 1-CALL 22DF** – aqui, o registro B deve armazenar o valor y, e o registro C deve armazenar o valor de x.

3. **CALL PLOT BC-CALL 22E5** – novamente o registro B deve conter o valor de y, e o registro C deve conter o valor de x. É esta sub-rotina que realmente desenha o ponto na tela.

Por exemplo:

Endereço	*Código de máquina*	*Mnemônica*	*Comentário*
END	3E02	LD A, 02	
END + 2	CD0116	CALL CHAN OPEN	Abre canal S.
END + 5	CD6B0D	CALL CLS	Limpa toda a tela.
END + 8	3E02	LD A, 02	
END + 10	CD0116	CALL CHAN OPEN	Reabre canal S.
END + 13	016432	LD BC, 3264	
END + 16	CDE522	CALL PLOT BC	PLOT 100, 50.
END + 19	C9	RET	Retorna à Basic.

6.2. **DRAW** – para este comando, existem dois pontos de entrada:

1. **DRAW 1-CALL 2477** – requer os valores das coordenadas x e y no topo da pilha da calculadora.

2. **DRAW 2-CALL 24BA** – requer para o registro B, o valor absoluto de y, e, para o registro C, o valor absoluto de x. O registro D deve conter o sinal de x e o registro E deve conter o sinal de y.

A rotina a seguir ilustra esse processo. Note que o par de registros H'L' tem seu valor guardado (salvo), para ser recuperado mais tarde, caso necessário.

Endereço	*Código de máquina*	*Mnemônica*	*Comentários*
END	3E02	LD A, 02	
END + 2	CD0116	CALL CHAN OPEN	
END + 5	CD6B0D	CALL CLS	
END + 8	3E02	LD A, 02	
END + 10	CD0116	CALL CHAN OPEN	
END + 13	D9	EXX	Troca HL por H'L' e guarda
END + 14	E5	PUSH HL	na pilha da máquina.
END + 15	D9	EXX	
END + 16	FD364364	LD (LSTPLOT), 64	Últimos valores de
END + 20	FD364464	LD (COORDS), 64	coordenadas.
END + 24	013232	LD BC, 3232	B = 50 e C = 50.
END + 27	110101	LD DE, 0101	Sinais positivos.
END + 30	CDBA24	CALL DRAW 2	DRAW 50, 50.
END + 33	D9	EXX	Recupera valor de HL.
END + 34	E1	POP HL	
END + 35	D9	EXX	
END + 36	C9	RET	Retorna à Basic.

Essa rotina coloca, como últimas coordenadas, os valores 100 e 100 e então traça a linha 50, 50.

Uma variante do comando DRAW, é quando se quer desenhar um arco de circunferência. Para tanto, os três valores x, y e raio devem ser colocados na pilha da calculadora, para que então se faça a chamada para a sub-rotina DRAW ARC-CALL 2394.

A rotina seguinte mostra a execução de um comando DRAW 50, 50, 1 a partir de um ponto de coordenadas 100, 100.

Endereço	*Código de máquina*	*Mnemônica*	*Comentários*
END			
:		IDÊNTICO À ANTERIOR	
END + 20			
END + 24	34	DEFB 34	Dado da pilha.
END + 25	40B00032	DEFBs	= 50 decimal.
END + 29	31	DEFB 31	"Duplica".
END + 30	A1	DEFB A1	Coloca na pilha.
END + 31	38	DEFB 38	Final de cálculo.
END + 32	CD9423	CALL DRAW ARC	Traça o arco.
END + 35	D9	EXX	
END + 36	E1	POP HL	
END + 37	D9	EXX	
END + 38	C9	RET	

6.3. **Circle** – mais uma vez, os três operandos do comando CIRCLE devem ser colocados na pilha da calculadora, antes de ser feita a chamada para CIRCLE 1 - CALL 232D.

Repare como o comando CIRCLE é lento.

Endereço	*Código de máquina*	*Mnemônica*	*Comentários*
END			
:		IDÊNTICO AO DRAW	
END + 15			
END + 16	EF	RST 0028	Usa calculadora.
END + 17	34	DEFB 34	Dado na pilha.
END + 18	40B00064	DEFBs	= 100 decimal.
END + 22	31	DEFB 31	Duplica.
END + 23	34	DEFB 34	Dado na pilha.
END + 24	40B00030	DEFBs	= 48 decimal.
END + 28	38	DEFB 38	Final de cálculo.
END + 29	CD2D23	CALL CIRCLE 1	Traça círculo.
END + 32	D9		
END + 33	E1		
END + 34	D9		
END + 35	C9		

Esta rotina traça o círculo 100, 100, 48.

NOTA: Os nomes das rotinas aqui utilizadas não querem dizer absolutamente nada. Somente foram baseadas na ROM do Sindair 2X Spectrum, como referência.

NOTAS INICIAIS

1. O byte mais significativo e o byte menos significativo.

Tomemos como exemplo o nosso endereço padrão, 65000.

Sabemos que

$$(253 \times 256) + 232 = 65000$$

Onde,

253 equivale ao byte mais significativo e
232 equivale ao byte menos significativo.

Esse processo é análogo ao sistema decimal, onde trabalhamos com, por exemplo, dezenas e unidades. Por exemplo:

$$27 = (2 \times 10) + 7$$

Onde,

2 é o valor mais significativo e
7 é o valor menos significativo.

Para fazer POKE de um valor x, em uma variável de sistema que consuma dois endereços, você deve se utilizar da fórmula:

POKE (variável sistema), x - 256 x INT (x/256)
POKE (variável sistema + 1), INT (x/256)

E, para saber o valor contido naquela variável, utilize:

PRINT PEEK (variável sistema) -256 x PEEK (variável sistema +1).

2. Você deve ter observado, no conjunto de instruções do microprocessador Z80, a diferença entre instruções que empregam parênteses em suas mnemônicas e instruções que não possuem parênteses.

Nas primeiras, ou seja, nas instruções que usam parênteses, o uso destes faz referência ao conteúdo daquele endereço apontado pelo registro ou par de registros envolvido na instrução.

Por exemplo:

LD A, B — carrega acumulador com conteúdo do registro B.
LD A, (HL) — carrega acumulador com conteúdo da locação apontada pelo par de registros HL.

Passemos então às rotinas e aos programas.

JOGO DE DAMAS

Este programa é uma combinação de Basic e linguagem de máquina, cujo objetivo é jogar gostosas partidas de dama com você.

O micro mostra o tabuleiro, com as suas peças (vermelhas) e as dele, pretas, em cima. As posições são identificadas por letras e números, respectivamente de A até H e de 1 a 8.

O computador checa a validade da sua jogada, e não permite saltos múltiplos.

O programa trabalha da seguinte maneira:

A parte em Basic serve para mostrar o tabuleiro, definir caracteres gráficos, aceitar ou não as jogadas do adversário e executá-las.

A parte em linguagem de máquina é para as jogadas do computador, a fim de que você não fique esperando muito tempo para ele jogar.

Os códigos decimais da linguagem de máquina estão nas declarações DATA, das linhas 9000 até 9450. Procure calmamente traduzir estes códigos para as respectivas mnemônicas, utilizando para tanto os Apêndices deste livro, juntamente com os Apêndices do manual do micro. É um grande exercício para quem está disposto a aprender realmente linguagem de máquina. Ou, então, use um programa Disassembler.

Como parte da linguagem de máquina, o computador tem uma representação do tabuleiro na memória, como sendo uma série de bytes, motivo dos diversos POKE e PEEK do programa, e, porque também são uma opção melhor que uma matriz, para efeito de jogadas.

Note na linha 3000, a chamada para a execução da rotina em linguagem de máquina.

A seguir, a listagem do programa.

```
   1 REM Jogo de damas
   2 REM A subrotina da linha 90
00 mao os codigos de maquina
   3 REM abcdefgh
   4 REM ABCDEFGH
  10 CLEAR 32019
 300 GO SUB 9000
 400 RESTORE 70
  40 BORDER 7: PAPER 7: INK 0: C
LS
  50 FOR x=32420 TO 32420+39
  60 READ a: POKE x,a: NEXT x
  70 DATA 255,255,255,255,255,25
5,255,255,255
  71 DATA 0,1,0,1,0,1,0,1,255
  72 DATA 1,0,1,0,1,0,1,0,6666
  73 DATA 0,1,0,1,0,1,0,1,6666
  74 DATA 0,0,0,0,0,0,0,0,6666
  75 DATA 0,0,0,0,0,0,0,0,6666
  76 DATA 2,0,2,0,2,0,2,0,2666
  77 DATA 0,2,0,2,0,2,0,2,2666
  78 DATA 0,0,0,0,0,0,0,0,2666
  79 DATA 255,255,255,255,255,25
5,255,255,255
  80 FOR x=USR "a" TO 63+USR "a"
  90 READ a: POKE x,a: NEXT x
 100 DATA 0,7,31,63,63,127,127,1
27
 101 DATA 0,224,248,252,252,254,
254,254
 102 DATA 127,127,127,63,63,31,7
,0
 103 DATA 254,254,254,252,252,24
```

```
8,224,0
  104 DATA 0,7,31,60,56,115,98,97
  105 DATA 0,224,248,60,28,46,70,
134
  106 DATA 97,98,115,56,60,31,7,0
  107 DATA 134,70,46,28,60,248,22
4,0
  110 PRINT "  A B C D E F G H":
PRINT
  115 LET d=3: LET l=5
  120 FOR y=1 TO 8: PRINT AT y+y,
0;y;" ";
  130 FOR z=1 TO 2: FOR x=1 TO 4
  140 PRINT PAPER d;"  "; PAPER l
;" ";: NEXT x: PRINT : PRINT "
";: NEXT z
  150 LET t=d: LET d=l: LET l=t
  160 NEXT y
  170 LET p=1
  180 FOR x=2 TO 8 STEP 2
  190 LET c=0: LET y=1: GO SUB 10
00: LET y=3: GO SUB 1000
  200 LET c=2: LET y=7: GO SUB 10
00: NEXT x
  210 FOR x=1 TO 7 STEP 2
  220 LET c=0: LET y=2: GO SUB 10
00
  230 LET c=2: LET y=6: GO SUB 10
00: LET y=8: GO SUB 1000: NEXT x
  300 GO SUB 2000
  310 GO SUB 3000
  320 GO TO 300
1000 PAPER l: INK c: PRINT AT y+
y,x+x;
1005 IF p=0 THEN PRINT "  ";
1010 IF p=1 THEN PRINT "AB";
1020 IF p=2 THEN PRINT "EF";
1030 PRINT AT y+y+1,x+x;
1035 IF p=0 THEN PRINT "  ";
1040 IF p=1 THEN PRINT "CD";
1050 IF p=2 THEN PRINT "GH";
1060 RETURN
2000 PAPER 7: INK 0: PRINT AT 6,
22;"Sua vez"
2010 BEEP .5,10: PRINT AT 7,22;"
De :    ";
2020 PRINT AT 8,22;"Para :    ";
2030 LET l=7: GO SUB 4000: LET f
x=x: LET fy=y
2040 LET fp=32419+9*fy+fx: LET f
=PEEK fp
2050 IF f<>2 AND f<>130 THEN GO
TO 2010
2060 LET l=8: GO SUB 4000: LET t
x=x: LET ty=y
2070 LET tp=32419+9*ty+tx: LET t
=PEEK tp
```



```
2080 LET dx=tx-fx: LET dy=ty-fy
2090 IF (ABS dx<>1 AND ABS dx<>2
) OR ABS dx<>ABS dy THEN GO TO 2
010
2100 IF (dy=1 OR dy=2) AND f=2 T
HEN GO TO 2010
2110 IF ABS dx=1 THEN GO TO 2500
2120 IF t<>0 THEN GO TO 2010
2130 LET jx=fx+dx/2: LET jy=fy+d
y/2
2140 LET jp=32419+9*jy+jx: LET j
=PEEK jp
2150 IF j<>1 AND j<>129 THEN GO
TO 2010
2160 LET c=2: LET p=0: LET x=fx:
 LET y=fy: GO SUB 1000: POKE fp,
0
2170 LET p=1: IF f=130 OR ty=1 T
HEN LET p=2
2180 LET x=tx: LET y=ty: GO SUB
1000: POKE tp,f: IF ty=1 THEN PO
KE tp,130
2190 LET p=0: LET x=jx: LET y=jy
: GO SUB 1000: POKE jp,0
2200 RETURN
2500 IF t<>0 THEN GO TO 2010
2510 LET p=0: LET c=2: LET x=fx:
 LET y=fy: GO SUB 1000: POKE fp,
0
2520 LET p=1: IF f=130 OR ty=1 T
HEN LET p=2
2530 LET x=tx: LET y=ty: GO SUB
1000: POKE tp,f: IF y=1 THEN POK
E tp,130
2540 RETURN
3000 LET mp=USR 32020
3002 PAPER 7: INK 0: PRINT AT 10
,22;"Minha vez"
3003 PRINT AT 11,22;"De :    ";
3004 PRINT AT 12,22;"Para:    ";
3010 IF mp=0 THEN GO TO 3900
3020 LET fy=INT (mp/256): LET ty
=mp-256*fy
3030 LET fx=INT (fy/16): LET fy=
fy-16*fx
3040 LET tx=INT (ty/16): LET ty=
ty-16*tx
3045 LET x=fx: LET y=fy: LET z=1
1: GO SUB 3500: LET x=tx: LET y=
ty: LET z=12: GO SUB 3500
3050 LET p=0: LET x=fx: LET y=fy
: GO SUB 1000
3060 LET tp=32419+9*ty+tx: LET t
=PEEK tp
3070 LET p=1: IF t=129 THEN LET
p=2
3080 LET c=0: LET x=tx: LET y=ty
```

```
: GO SUB 1000
3090 IF ABS (tx-fx)<>2 THEN RETU
RN
3100 LET x=fx+(tx-fx)/2: LET y=f
y+(ty-fy)/2
3110 LET p=0: GO SUB 1000: RETUR
N
3500 PRINT AT z,28;CHR$ (x+64);C
HR$ (y+48): RETURN
3900 PRINT INK 0;AT 20,0;"Voce v
enceu - Parabens"
3901 PRINT AT 21,0;"Outra partid
a?"
3902 IF INKEY$="" THEN GO TO 390
2
3903 IF INKEY$="s" THEN GO TO 10
3904 STOP
4000 PRINT AT l,28; FLASH 1;" ";
CHR$ 8;
4010 LET a$=INKEY$
4015 IF a$="0" THEN GO TO 5000
4020 IF a$<"a" OR a$>"h" THEN GO
 TO 4010
4030 PRINT a$: LET x=CODE a$-96
4040 PRINT AT l,29; FLASH 1;" ";
CHR$ 8;
4050 LET a$=INKEY$
4060 IF a$<"1" OR a$>"8" THEN GO
 TO 4050
4070 PRINT a$: LET y=CODE a$-48
4080 RETURN
5000 PRINT INK 0;AT 20,0;"Azar s
eu - Eu venci"
5010 GO TO 3901
9000 RESTORE 9100: FOR x=32020 T
O 32301
9010 READ a: POKE x,a: NEXT x
9020 RETURN
9100 DATA 175,33,177,125,6,10,11
6,35
9110 DATA 16,252,6,35,33,174,126
,126
9120 DATA 254,1,40,16,254,129,32
,24
9130 DATA 17,246,255,205,187,125
,17,248
9140 DATA 255,205,187,125,17,8,0
,205
9150 DATA 187,125,17,10,0,205,18
7,125
9160 DATA 35,35,16,219,58,182,12
5,167
9170 DATA 40,18,42,183,125,237,9
1,185
9180 DATA 125,229,126,54,0,25,54
,0
9190 DATA 25,119,24,19,58,177,12
```



```
5,167
9200 DATA 40,29,42,178,125,237,9
1,180
9210 DATA 125,126,54,0,229,25,11
9,235
9220 DATA 225,205,135,125,121,23
0,15,254
9230 DATA 8,192,26,246,128,18,20
1,1
9240 DATA 0,0,201,235,205,146,12
5,79
9250 DATA 235,205,146,125,71,201
,229,213
9260 DATA 197,17,164,126,175,237
,82,61
9270 DATA 17,9,0,60,237,82,48,25
1
9280 DATA 71,25,125,60,7,7,7,7
9290 DATA 128,193,209,225,201,0,
0,0
9300 DATA 0,0,0,0,0,0,0,205
9310 DATA 19,126,254,255,200,230
,127,254
9320 DATA 1,200,254,2,40,78,229,
25
9330 DATA 213,17,246,255,205,13,
126,40
9340 DATA 39,17,248,255,205,13,1
26,40
9350 DATA 31,17,8,0,205,5,126,40
9360 DATA 23,17,10,0,205,5,126,4
0
9370 DATA 15,209,225,34,178,125,
237,83
9380 DATA 180,125,62,1,50,177,12
5,201
9390 DATA 58,177,125,167,40,235,
209,225
9400 DATA 201,205,19,126,230,127
,254,2
9410 DATA 201,205,19,126,254,130
,201,229
9420 DATA 25,126,225,201,229,25,
205,19
9430 DATA 126,225,254,0,192,62,1
,50
9440 DATA 182,125,34,183,125,237
,83,185
9450 DATA 125,201
```

PROGRAMA REPRODUTOR DA FALA

Este programa, muito interessante por sinal, permite que o computador TK 90X armazene e reproduza posteriormente a sua fala, transformando-a numa sucessão de bits armazenados em sua memória, agrupados por bytes. Para reproduzir esses "bytes", ele envia os bits para a saída de som.

Para manter o som que ele recebeu o mais semelhante possível com a voz, muita memória é consumida, na razão de 1 a 2K de RAM por segundo de fala.

Digite o programa cuidadosamente, e grave-o.

A seguir, digite a rotina "ouvir", a partir de um endereço determinado, digamos, 65000. Grave-a também, logo a seguir, do programa Basic. Finalmente, digite a rotina "falar", num endereço posterior aos utilizados pela rotina "ouvir", para não haver sobreposição. Grave-a igualmente, imediatamente após a última.

Siga estas instruções para utilizar o programa:

1. Coloque uma fita cassete virgem no seu gravador, ou simplesmente acione-o em modo de gravação, anulando a trava antigravação, para que você possa utilizar o seu microfone (supondo-se que ele tenha um — em caso negativo, providencie a ligação de um qualquer no gravador);

2. Desconecte o cabo de microfome entre o gravador e o computador, e ligue o cabo "ear" para "ear" (fone de ouvido-monitor de som);

3. Carregue o programa no computador. Se você gravou-o sem a opção de "AUTO RUN", então dê RUN GØ. Para gravar com essa opção, digite "SAVE falador" LINE 610;

4. Responda as perguntas do computador, relativas ao número de palavras e seu respectivo comprimento. A linha 150 permite que o gravador simplesmente armazene aquela palavra, não tendo nada a ver com a reprodução do som. Na linha 420, o tempo de atraso faz referência ao tom da sua voz. Valores baixos, tipo 10, são melhores para a posterior reprodução. Quando aparecer a mensagem da linha 450, e você pressionar ENTER, o computador estará pronto para receber a sua palavra, para armazenamento. Acabando o tempo, automaticamente ele passa para a sua reprodução. Pressione SPACE para interromper o processo;

5. Não se esqueça de aumentar o volume da televisão.

Estude as rotinas em linguagem de máquina, e veja como o computador utiliza as portas de entrada/saída para receber e enviar sinais.

A seguir as listagens do programa.


```
  0>REM  ARQUITRON INFORMATICA
LTDA
  10 BORDER 0: PAPER 0: INK 6: B
RIGHT 1: CLS
  15 GO TO 1000
  30 CLEAR 32767
  50 LET lis=65000: LET sp=65100
  70 LET ze=SIN PI: LET on=SGN P
I: LET tw=on+on: LET th=INT PI:
LET po=256
  80 LET st=64999: LET lim=32768
 110 INPUT "Numero maximo de pal
avras ";m: IF m<on THEN GO TO 11
0
 120 INPUT "Comprimento maximo d
e palavras ";n: IF n<ze THEN GO
TO 120
 130 DIM a(m): DIM n$(m,n)
 140 FOR c=on TO m
 150 INPUT "Digite a palavra ";(
c);": ";a$
 160 LET n$(c)=a$( TO (LEN a$ AN
D LEN a$<=n)+(n AND LEN a$>n))
 170 GO SUB 370
 180 IF a(c,th)=lim THEN PRINT "
Não cabe na memoria ": GO TO 100
0
 190 NEXT c
 200 PRINT "Vocabulario completo
": GO TO 1000
 220 CLS : PRINT "Numero da pala
vra ";TAB 15;"Atraso";TAB 21;"In
icio";TAB 28;"Fim"
 230 FOR c=on TO m
 240 PRINT c;TAB th;n$(c);TAB 18
;a(c,on);TAB 21;a(c,tw);TAB 27;a
(c,th)
 250 GO SUB 520
 260 NEXT c
 270 GO TO 1000
 290 INPUT "Qual das ";(m);" pal
avras voce quer ouvir ";c
 300 IF c>m OR c<on THEN GO TO 2
90
 310 GO SUB 520: GO TO 1000
 330 INPUT "Qual das ";(m);" pal
avras voce quer mudar ";c
 340 IF c>m OR c<on THEN GO TO 3
30
 350 LET st=a(c,tw): LET lim=a(c
,th)
 360 GO SUB 370: GO TO 1000
 380 POKE lis+on,st-po*INT (st/p
o)
 390 POKE lis+tw,INT (st/po)
 400 POKE lis+4,lim-po*INT (lim/
po)
```

```
 410 POKE lis+5,INT (lim/po)
 420 INPUT "Tempo de atraso (1-2
55) ";d
 430 POKE lis+48,d
 440 LET a(c,on)=d: LET a(c,tw)=
st
 450 INPUT "Pressione ENTER para
 falar e em seguida ouvir "; LIN
E a$
 460 CLS : PRINT "Pressione BREA
K ou aguarde o     final"
 470 LET a(c,th)=(USR lis)+on
 480 IF a(c,th)>st THEN PRINT "N
ao ha som ": LET c=c-on: RETURN
 490 GO SUB 520
 500 LET st=a(c,th)-on
 510 RETURN
 530 POKE sp+on,a(c,tw)-po*INT (
a(c,tw)/po)
 540 POKE sp+tw,INT (a(c,tw)/po)
 550 POKE sp+4,a(c,th)-po*INT (a
(c,th)/po)
 560 POKE sp+5,INT (a(c,th)/po)
 570 POKE sp+50,a(c,on)
 580 LET a=USR sp
 590 RETURN
 610 CLEAR 64999: LOAD "ouvir"CO
DE : LOAD "falar"CODE : RUN
 630 INPUT "Gravar da palavra ";
c1;" ate palavra ";c2
 640 IF c1>m OR c2>m OR c1<ze OR
 c2<ze OR c1>c2 THEN GO TO 630
 650 LET stt=a(c2,th): LET le=a(
c1,tw)-stt+on
 660 IF le<ze THEN PRINT "Compri
mento negativo ": GO TO 630
 670 CLS : PRINT " Escrever abai
xo "'"Endereco inicial = ";stt
 680 INPUT "Nome do arquivo ";a$
 690 IF a$="" OR LEN a$>10 THEN
GO TO 680
 700 SAVE a$CODE stt,le
 710 GO TO 1000
1000 CLS : PRINT "          "
      COMANDOS"
1010 PRINT " 1 - CONSTRUIR VOCAB
ULARIO"
1020 PRINT " 2 - VER VOCABULARIO
"
1030 PRINT " 3 - OUVIR ALGUMA PA
LAVRA "'" 4 - MUDAR ALGUMA PALAV
RA"
1040 PRINT " 5 - GRAVAR PALAVRAS
"
1050 PAUSE 0: LET e$=INKEY$
1070 IF e$="1" THEN GO TO 30
1080 IF e$="2" THEN GO TO 210
```



```
1090 IF e$="3" THEN GO TO 280
1100 IF e$="4" THEN GO TO 320
1110 IF e$="5" THEN GO TO 620
1120 GO TO 1000
```

ROTINA OUVIR

```
>Disassemble £5E000
FDE0 21FF5A     ld hl,5AFF
FDE3 110040     ld de,4000
FDE6 E5         push hl
FDE7 A7         and a
FDE8 ED52       sbc hl,de
FDEA 23         inc hl
FDEB 4C         ld c,h
FDEC 45         ld b,l
FDED E1         pop hl
FDEE 383A       jr c,FE32
FDF0 0C         inc c
FDF1 F3         di
FDF2 167F       ld d,7F
FDF4 D9         exx
FDF5 1680       ld d,80
FDF7 D9         exx
FDF8 3E7F       ld a,7F
FDFA DBFE       in a,(FE)
FDFC 1F         rra
FDFD 302B       jr nc,FE32
FDFF CB6F       bit 5,a
FE01 20F5       jr nz,FDF8
FE03 D9         exx
FE04 5A         ld e,d
FE05 DBFE       in a,(FE)
FE07 CB17       rl a
FE09 CB17       rl a
FE0B CB1B       rr e
FE0D 3806       jr c,FE15
FE0F 060C       ld b,0C
FE11 10FE       djnz FE11
FE13 18F0       jr FE05
FE15 7B         ld a,e
FE16 D9         exx
FE17 77         ld (hl),a
FE18 2B         dec hl
FE19 7A         ld a,d
FE1A DBFE       in a,(FE)
FE1C 1F         rra
FE1D 3005       jr nc,FE24
FE1F 10E2       djnz FE03
FE21 0D         dec c
FE22 20DF       jr nz,FE03
FE24 23         inc hl
FE25 7E         ld a,(hl)
FE26 3C         inc a
FE27 28FB       jr z,FE24
FE29 2B         dec hl
FE2A 44         ld b,h
FE2B 4D         ld c,l
```

```
FE34 FB          ei
FE35 C9          ret
```

ROTINA FALAR

```
>Disassemble fe4c
FE4C 21FF5A      ld hl,5AFF
FE4F 110040      ld de,4000
FE52 E5          push hl
FE53 A7          and a
FE54 ED52        sbc hl,de
FE56 23          inc hl
FE57 4C          ld c,h
FE58 45          ld b,l
FE59 E1          pop hl
FE5A 0C          inc c
FE5B 3A485C      ld a,(5C48)
FE5E 0F          rrca
FE5F 0F          rrca
FE60 0F          rrca
FE61 E607        and 07
FE63 D9          exx
FE64 4F          ld c,a
FE65 E5          push hl
FE66 2610        ld h,10
FE68 D9          exx
FE69 F3          di
FE6A 7E          ld a,(hl)
FE6B 0F          rrca
FE6C 0F          rrca
FE6D 0F          rrca
FE6E 0F          rrca
FE6F D9          exx
FE70 5F          ld e,a
FE71 1608        ld d,08
FE73 78          ld a,e
FE74 A4          and h
FE75 B1          or c
FE76 D3FE        out (FE),a
FE78 CB08        rrc e
FE7A 15          dec d
FE7B 2805        jr z,FE83
FE7D 0600        ld b,00
FE7F 10FE        djnz FE7F
FE81 18F0        jr FE73
FE83 D9          exx
FE84 2B          dec hl
FE85 3E00        ld a,00
FE87 10E1        djnz FE6A
FE89 0D          dec c
FE8A 20DE        jr nz,FE6A
FE8C D9          exx
FE8D E1          pop hl
FE8E D9          exx
FE8F FB          ei
FE90 C9          ret
```


ROTINAS DE SONS DIVERSOS

A única maneira de se extrair sons do seu micro, na linguagem Basic, é através do comando SOUND, o que, convenhamos, é um tanto limitado, ainda mais considerando-se a sua capacidade, que vai muito além desse SOUND.

A seguir são listadas três rotinas que produzem sons diversos e jamais conseguidos através da linguagem Basic.

Repare com muita atenção nas duas primeiras listagens, e note a pequena diferença entre elas. Enquanto a primeira utiliza a instrução INC E, a segunda usa a instrução DEC E, para justamente alterar a nota musical. Você também pode alterar os tons, através do comando POKE END + 7, para carregar o registro E com outros valores.

A terceira rotina produz outro tipo de som, que, dependendo de um determinado valor, pode parecer o ronco de um motor de um carro, o barulho de um avião, ou até um tiro de revólver.

Após digitar esta última, execute-a através de RAND USR END, para ver seu efeito.

Agora, digite o pequeno programa a seguir:

```
10 INPUT "Duração "; d
20 POKE END + 11, d - 256 x INT (d/256)
30 POKE END + 12, INT (d/256)
40 PRINT d
50 INF INKEY$ <> " " THEN RAND USR END
60 GOTO 50
```

1ª ROTINA

```
>Disassemble fde8
FDE8 3A485C      ld a,(5C48)
FDEB 0F          rrca
FDEC 0F          rrca
FDED 0F          rrca
FDEE 1E01        ld e,01
FDF0 F3          di
FDF1 D3FE        out (FE),a
FDF3 EE10        xor 10
FDF5 43          ld b,e
FDF6 10FE        djnz FDF6
FDF8 1C          inc e
FDF9 20F6        jr nz,FDF1
FDFB FB          ei
FDFC C9          ret
```

2ª ROTINA

```
>Disassemble fde8
FDE8 3A485C    ld a,(5C48)
FDEB 0F        rrca
FDEC 0F        rrca
FDED 0F        rrca
FDEE 1E00      ld e,00
FDF0 F3        di
FDF1 D3FE      out (FE),a
FDF3 EE10      xor 10
FDF5 43        ld b,e
FDF6 10FE      djnz FDF6
FDF8 1D        dec e
FDF9 20F6      jr nz,FDF1
FDFB FB        ei
FDFC C9        ret
```

3ª ROTINA

```
>Disassemble fde8
FDE8 F3        di
FDE9 3A485C    ld a,(5C48)
FDEC 0F        rrca
FDED 0F        rrca
FDEE 0F        rrca
FDEF 08        ex af,af'
FDF0 2600      ld h,00
FDF2 018000    ld bc,0080
FDF5 08        ex af,af'
FDF6 D3FE      out (FE),a
FDF8 EE10      xor 10
FDFA 08        ex af,af'
FDFB 2E00      ld l,00
FDFD 55        ld d,l
FDFE 5C        ld e,h
FDFF A7        and a
FE00 ED52      sbc hl,de
FE02 ED52      sbc hl,de
FE04 11FE00    ld de,00FE
FE07 19        add hl,de
FE08 7D        ld a,l
FE09 94        sub h
FE0A 3801      jr c,FE0D
FE0C 3D        dec a
FE0D 67        ld h,a
FE0E 3D        dec a
FE0F 20FD      jr nz,FE0E
FE11 0B        dec bc
FE12 78        ld a,b
FE13 B1        or c
FE14 20DF      jr nz,FDF5
FE16 FB        ei
```


ROTINA PARA ROLAR A TELA (SCROLL) LENTAMENTE

Esta rotina é mais conhecida como "scroll lento", e serve como apresentação de textos de muitos jogos ingleses. Você deve conhecê-la. Digite-a, preencha a sua tela com palavras, ou qualquer coisa, e dê, por exemplo, os comandos em modo direto:

FOR f = 1 TO 176 : RAND USR 65000 : NEXT f

```
>Disassemble fde8
FDE8 210040    ld hl,4000
FDEB E5        push hl
FDEC 11005B    ld de,5B00
FDEF 012000    ld bc,0020
FDF2 EDB0      ldir
FDF4 E1        pop hl
FDF5 0603      ld b,03
FDF7 C5        push bc
FDF8 E5        push hl
FDF9 0608      ld b,08
FDFB C5        push bc
FDFC E5        push hl
FDFD 0607      ld b,07
FDFF C5        push bc
FE00 E5        push hl
FE01 E5        push hl
FE02 D1        pop de
FE03 24        inc h
FE04 012000    ld bc,0020
FE07 EDB0      ldir
FE09 E1        pop hl
FE0A 24        inc h
FE0B C1        pop bc
FE0C 10F1      djnz FDFF
FE0E E5        push hl
FE0F 11E005    ld de,05E0
FE12 ED52      sbc hl,de
FE14 D1        pop de
FE15 012000    ld bc,0020
FE18 EDB0      ldir
FE1A E1        pop hl
FE1B 112000    ld de,0020
FE1E 19        add hl,de
FE1F C1        pop bc
FE20 10D9      djnz FDFB
FE22 11E005    ld de,05E0
FE25 19        add hl,de
FE26 E5        push hl
FE27 112000    ld de,0020
FE2A 19        add hl,de
FE2B D1        pop de
FE2C 012000    ld bc,0020
FE2F EDB0      ldir
FE31 E1        pop hl
```

```
FE32 110005     ld de,0800
FE35 19         add hl,de
FE36 C1         pop bc
FE37 10BE       djnz FDF7
FE39 112000     ld de,0020
FE3C ED52       sbc hl,de
FE3E EB         ex de,hl
FE3F 21005B     ld hl,5B00
FE42 0620       ld b,20
FE44 7E         ld a,(hl)
FE45 00         nop
FE46 12         ld (de),a
FE47 13         inc de
FE48 23         inc hl
FE49 10F9       djnz FE44
FE4B C9         ret
```

ROTINA PARA REPRODUÇÃO DE FITAS CASSETES

Esta rotina trabalha de um modo similar ao programa falador. Ela lê a porta de entrada EAR aproximadamente a cada 17 microssegundos (um microssegundo é um milionésimo de segundo), o que produz uma faixa de leitura de 57000 vezes por segundo (57 khz).

Para usar esta rotina, faça o seguinte:

1. desconecte o cabo MIC para MIC entre o gravador e o micro;

2. conecte o cabo EAR para EAR entre o gravador e o micro;

3. coloque uma fita com músicas gravadas e, com o volume no máximo, coloque o gravador em modo PLAY, após acionar esta rotina através de LET x = USR END. Para interromper a rotina, utilize a tecla BREAK.

```
>Disassemble fde8
FDE8 F3         di
FDE9 010000     ld bc,0000
FDEC DBFE       in a,(FE)
FDEE 0F         rrca
FDEF 0F         rrca
FDF0 D3FE       out (FE),a
FDF2 AF         xor a
FDF3 D3FE       out (FE),a
FDF5 10F5       djnz FDEC
FDF7 0D         dec c
FDF8 20F2       jr nz,FDEC
FDFA 3E7F       ld a,7F
FDFC DBFE       in a,(FE)
FDFE 1F         rra
FDFF 38EB       jr c,FDEC
FE01 FB         ei
FE02 C9         ret
```


ROTINA PARA LER TECLADO (RECONHECER CÓDIGOS DE TECLAS PRESSIONADAS)

Infelizmente, no Manual do TK 90X, o capítulo sobre as instruções Basic, IN e OUT, saiu fraco. Mas, se você leu o meu livro de Basic, sabe que é possível ler o teclado com a instrução IN, cuja grande vantagem sobre INKEYS ou qualquer outro método em Basic é o de poder reconhecer diversas teclas ao mesmo tempo, fazendo com que a combinação de mais de uma tecla pressionada forneça uma outra ordem a ser executada.

A rotina abaixo faz exatamente isso, da seguinte maneira:

Tecla Q – movimento para cima.
Tecla A – movimento para baixo.
Tecla O – movimento para a esquerda.
Tecla P – movimento para a direita.
Teclas de CAPS SHIFT a SPACE – atirar.

Você pode introduzir esta rotina em seus jogos ou programas, fazendo com que novas direções sejam aceitas, como as quatro diagonais, através das combinações das teclas padrão da rotina.

```
>Disassemble fde8
FDE8 010000     ld bc,0000
FDEB 3EDF       ld a,DF
FDED DBFE       in a,(FE)
FDEF 1F         rra
FDF0 3803       jr c,FDF5
FDF2 0C         inc c
FDF3 1805       jr FDFA
FDF5 1F         rra
FDF6 3802       jr c,FDFA
FDF8 CBC9       set 1,c
FDFA 3EFD       ld a,FD
FDFC DBFE       in a,(FE)
FDFE 1F         rra
FDFF 3804       jr c,FE05
FE01 CBD1       set 2,c
FE03 1809       jr FE0E
FE05 3EF8       ld a,F8
FE07 DBFE       in a,(FE)
FE09 1F         rra
FE0A 3802       jr c,FE0E
FE0C CBD9       set 3,c
FE0E 3E7E       ld a,7E
FE10 DBFE       in a,(FE)
FE12 2F         cpl
FE13 E61F       and 1F
FE15 C8         ret z
FE16 CBE1       set 4,c
FE18 C9         ret
```

ROTINA PARA ATRIBUTOS NA TELA

Como você sabe, pode escolher uma das oito cores do TK 90X tanto para INK quanto para PAPER. Você também pode especificar onde quer BRIGHT ou FLASH 1 ou 0. O único problema entretanto é que em Basic você não pode mudar qualquer desses atributos facilmente, em relação a qualquer caractere na tela, sem especificá-lo com os novos atributos.

A rotina a seguir permite que você faça essas alterações instantaneamente, operando num retângulo especificado na tela, sem alterar o que já está impresso ali.

A rotina opera num retângulo especificado da tela, usando as coordenadas do comando PRINT AT.

Consideremos, então, (X1, Y1) as coordenadas do canto superior esquerdo, e as coordenadas (X2, Y2) do canto inferior direito.

Estas coordenadas devem estar armazenadas nos endereços 23332, 23333, 23334 e 23335, através dos comandos:

POKE 23332, X1
POKE 23333, Y1
POKE 23334, X2
POKE 23335, Y2

Todas as cores do TK 90X derivam das três cores primárias: verde, vermelho e azul.

As operações possíveis com esta rotina são:

Pôr atributos.
Tirar atributos.
Mudar atributos.
Deixar atributos como estão.

Através das combinações das três cores primárias, surgem as outras cores do micro.

Para os atributos, devemos associar certos valores:

Tinta azul	1
Tinta vermelha	2
Tinta verde	4
Papel azul	8
Papel vermelho	16
Papel verde	32
BRIGHT	64
FLASH	128

Para as operações funcionarem com os nossos atributos, precisamos:

POKE 23328, soma dos valores dos atributos sem alterá-los.
POKE 23329, soma dos valores dos atributos a serem impressos.
POKE 23331, soma dos valores dos atributos a serem alterados.

Para executar a rotina, você deve dar o comando RAND USR END ou LET t = USR END, após, evidentemente ter dado os devidos POKE, para as suas modificações das cores.

```
>Disassemble fde8
FDE8 21205B     ld hl,5B20
FDEB 7E         ld a,(hl)
FDEC 2F         cpl
FDED 23         inc hl
FDEE A6         and (hl)
FDEF 23         inc hl
FDF0 77         ld (hl),a
FDF1 23         inc hl
FDF2 23         inc hl
FDF3 46         ld b,(hl)
FDF4 23         inc hl
FDF5 23         inc hl
FDF6 7E         ld a,(hl)
FDF7 FE18       cp 18
FDF9 D0         ret nc
FDFA 90         sub b
FDFB D8         ret c
FDFC 3C         inc a
FDFD F5         push af
FDFE 04         inc b
FDFF 05         dec b
FE00 E5         push hl
FE01 21005B     ld hl,5B00
FE04 2806       jr z,FE0C
FE06 112000     ld de,0020
FE09 19         add hl,de
FE0A 10FA       djnz FE06
FE0C EB         ex de,hl
FE0D E1         pop hl
FE0E D5         push de
FE0F 23         inc hl
FE10 7E         ld a,(hl)
FE11 FE20       cp 20
FE13 3803       jr c,FE18
FE15 C1         pop bc
FE16 C1         pop bc
FE17 C9         ret
```

```
FE18 2B         dec hl
FE19 2B         dec hl
FE1A 4E         ld c,(hl)
FE1B E1         pop hl
FE1C 09         add hl,bc
FE1D 91         sub c
FE1E 38F6       jr c,FE16
FE20 3C         inc a
FE21 47         ld b,a
FE22 3E20       ld a,20
FE24 90         sub b
FE25 5F         ld e,a
FE26 1600       ld d,00
FE28 C5         push bc
FE29 D5         push de
FE2A 3A235B     ld a,(5B23)
FE2D 57         ld d,a
FE2E 3A205B     ld a,(5B20)
FE31 5F         ld e,a
FE32 7E         ld a,(hl)
FE33 AA         xor d
FE34 A3         and e
FE35 5F         ld e,a
FE36 3A225B     ld a,(5B22)
FE39 83         add a,e
FE3A 77         ld (hl),a
FE3B 23         inc hl
FE3C 10F0       djnz FE2E
FE3E D1         pop de
FE3F 19         add hl,de
FE40 C1         pop bc
FE41 F1         pop af
FE42 3D         dec a
FE43 C8         ret z
FE44 F5         push af
FE45 18E1       jr FE28
```

ROTINA DE INVERSÃO DOS ATRIBUTOS

Esta rotina pega todas as "TINTAS" e "PAPÉIS" de cada caractere dentro de um retângulo especificado da tela, e os inverte.

Este retângulo é especificado exatamente da mesma maneira que a rotina anterior. Portanto, você deve, antes de mais nada, dar aqueles quatro POKEs para criar o retângulo. A seguir, para executar a rotina, digite LET t = USR END.

```
)Disassemble fde8
FDE8 ED4B245B ld bc,(5B24)
FDEC 2A265B   ld hl,(5B26)
FDEF 7D       ld a,l
FDF0 FE18     cp 18
FDF2 D0       ret nc
FDF3 91       sub c
```



```
FDF4 D8       ret c
FDF5 5F       ld e,a
FDF6 7C       ld a,h
FDF7 FE20     cp 20
FDF9 D0       ret nc
FDFA 90       sub b
FDFB D8       ret c
FDFC 57       ld d,a
FDFD 14       inc d
FDFE 1C       inc e
FDFF D5       push de
FE00 210058   ld hl,5800
FE03 AF       xor a
FE04 B9       cp c
FE05 112000   ld de,0020
FE08 2804     jr z,FE0E
FE0A 19       add hl,de
FE0B 0D       dec c
FE0C 20FC     jr nz,FE0A
FE0E B8       cp b
FE0F 2803     jr z,FE14
FE11 23       inc hl
FE12 10FD     djnz FE11
FE14 C1       pop bc
FE15 D5       push de
FE16 E5       push hl
FE17 C5       push bc
FE18 3E07     ld a,07
FE1A A6       and (hl)
FE1B 0F       rrca
FE1C 0F       rrca
FE1D 4F       ld c,a
FE1E 3E38     ld a,38
FE20 A6       and (hl)
FE21 81       add a,c
FE22 0F       rrca
FE23 0F       rrca
FE24 0F       rrca
FE25 4F       ld c,a
FE26 3EC0     ld a,C0
FE28 A6       and (hl)
FE29 8A       adc a,d
FE2A 77       ld (hl),a
FE2B 23       inc hl
FE2C 10EA     djnz FE18
FE2E C1       pop bc
FE2F E1       pop hl
FE30 D1       pop de
FE31 0D       dec c
FE32 19       add hl,de
FE33 20E0     jr nz,FE15
FE35 C9       ret
```

Aqui está uma versão desta rotina de inversão de atributos, mas, só para a tela inteira. Não é necessário dar-se nenhum POKE, e esta ocupa somente 29 bytes.

```
:Disassemble fde8
FDE8 210058     ld hl,5800
FDEB 3E07       ld a,07
FDED A6         and (hl)
FDEE 0F         rrca
FDEF 0F         rrca
FDF0 57         ld d,a
FDF1 3E38       ld a,38
FDF3 A6         and (hl)
FDF4 82         add a,d
FDF5 0F         rrca
FDF6 0F         rrca
FDF7 57         ld d,a
FDF8 3EC0       ld a,C0
FDFA A6         and (hl)
FDFB 82         add a,d
FDFC 77         ld (hl),a
FDFD 23         inc hl
FDFE 7C         ld a,h
FDFF FE58       cp 58
FE01 38E8       jr c,FDEB
FE03 C9         ret
```

ROTINAS PARA "SCROLL" TEXTOS E GRÁFICOS

Aqui está um conjunto de rotinas para que você faça diversos SCROLL na tela. Através do estudo destas, você pode criar o movimento que quiser, bem como na área que quiser.

Da mesma forma que a rotina anterior, aqui, para cada movimento existem duas opções:

– a primeira é para rolar um determinado retângulo da tela; e

– a segunda, para rolar a tela inteira.

No caso de se utilizar a primeira opção, primeiramente você deve então criar o seu retângulo, através daqueles mesmos POKEs anteriores. Você tem três opções ao utilizar estar rotinas:

1. manter a linha ou coluna que não foi movimentada exatamente como estava antes;

2. rolar a linha ou coluna que seria deletada pelo movimento escolhido; e

3. preencher a linha ou coluna que não foi movimentada com um dos 256 padrões definidos por um POKE 23347, um número entre 0 e 255.

A rotina a seguir faz o movimento de SCROLL para a direita, em um retângulo especificado por você. Para fazê-la trabalhar em sentido contrário, você deve fazer estas três pequenas alterações:

a.	NOP	00
b.	INC HL	23
c.	LDIR	EDB0

Se você não quiser armazenar as duas rotinas separadamente, poderá converter de uma para outra, durante um programa Basic, ou através de comandos diretos:

LET T = END	(endereço inicial)
ESQUERDA	DIREITA
POKE T + 23, 0	POKE T + 23, 68
POKE T + 44, 35	POKE T + 44, 43
POKE T + 52, 176	POKE T + 52, 184

Pois então, não perca mais tempo e digite a rotina.

```
:Disassemble fde8
FDE8 ED4B2458   ld bc,(5824)
FDEC 2A2658     ld hl,(5826)
FDEF 7D         ld a,l
FDF0 FE18       cp 18
FDF2 D0         ret nc
FDF3 91         sub c
FDF4 D8         ret c
FDF5 57         ld d,a
FDF6 7C         ld a,h
FDF7 FE20       cp 20
FDF9 D0         ret nc
FDFA 90         sub b
FDFB D8         ret c
FDFC 5F         ld e,a
FDFD 14         inc d
FDFE D5         push de
FDFF 44         ld b,h
FE00 79         ld a,c
FE01 E618       and 18
FE03 C640       add a,40
FE05 57         ld h,a
FE06 79         ld a,c
FE07 87         add a,a
FE08 87         add a,a
FE09 87         add a,a
FE0A 87         add a,a
FE0B 87         add a,a
FE0C 80         add a,b
```

```
FE0D 6F         ld l,a
FE0E C1         pop bc
FE0F C5         push bc
FE10 E5         push hl
FE11 54         ld d,h
FE12 5D         ld e,l
FE13 7E         ld a,(hl)
FE14 2B         dec hl
FE15 08         ex af,af'
FE16 AF         xor a
FE17 47         ld b,a
FE18 B9         cp c
FE19 2802       jr z,FE1D
FE1B EDB8       lddr
FE1D 2A3258     ld hl,(5B32)
FE20 7D         ld a,l
FE21 FE01       cp 01
FE23 3805       jr c,FE2A
FE25 7C         ld a,h
FE26 2001       jr nz,FE29
FE28 08         ex af,af'
FE29 12         ld (de),a
FE2A E1         pop hl
FE2B 24         inc h
FE2C 7C         ld a,h
FE2D E607       and 07
FE2F 20DD       jr nz,FE0E
FE31 1120F8     ld de,F820
FE34 19         add hl,de
FE35 7D         ld a,l
FE36 FE20       cp 20
FE38 3003       jr nc,FE3D
FE3A 0607       ld b,07
FE3C 09         add hl,bc
FE3D C1         pop bc
FE3E 10CF       djnz FE0F
FE40 C9         ret
```

E aqui estão as versões simplificadas da rotina anterior, trabalhando apenas com a tela toda.

Elas requerem, para funcionar, apenas dois POKEs nos endereços 23346 e 23347.

Esquerda

```
>Disassemble fde8
FDE8 210040     ld hl,4000
FDEB 011F00     ld bc,001F
FDEE 7E         ld a,(hl)
FDEF 54         ld d,h
FDF0 5D         ld e,l
FDF1 23         inc hl
```



```
FDF2 EDB0       ldir
FDF4 08         ex af,af'
FDF5 ED4B325B   ld bc,(5B32)
FDF9 79         ld a,c
FDFA FE01       cp 01
FDFC 3805       jr c,FE03
FDFE 78         ld a,b
FDFF 2001       jr nz,FE02
FE01 08         ex af,af'
FE02 12         ld (de),a
FE03 7C         ld a,h
FE04 FE58       cp 58
FE06 38E3       jr c,FDEB
FE08 C9         ret
```

Direita

```
>Disassemble fde8
FDE8 21FF57     ld hl,57FF
FDEB 011F00     ld bc,001F
FDEE 7E         ld a,(hl)
FDEF 54         ld d,h
FDF0 5D         ld e,l
FDF1 2B         dec hl
FDF2 EDB8       lddr
FDF4 08         ex af,af'
FDF5 ED4B325B   ld bc,(5B32)
FDF9 79         ld a,c
FDFA FE01       cp 01
FDFC 3805       jr c,FE03
FDFE 78         ld a,b
FDFF 2001       jr nz,FE02
FE01 08         ex af,af'
FE02 12         ld (de),a
FE03 7C         ld a,h
FE04 FE58       cp 58
FE06 38E3       jr c,FDEB
FE08 C9         ret
```

A seguir, as versões, para a tela toda, de scroll para cima e para baixo.

São rotinas simples, que ocupam respectivamente 67 e 69 bytes.

As rotinas de scroll para cima e para baixo em áreas de retângulos definidos na tela são um pouco mais complicadas que as rotinas de movimento lateral, já que ocupam respectivamente 109 bytes cada uma. Só que infelizmente não as incluí no livro, para que fiquem como exercício.

A seguir, estão listadas as rotinas que movimentam a tela inteira, respectivamente, para cima e para baixo, com 67 e 69 bytes de comprimento cada. Compare esta rotina de scroll para cima com aquela anterior de scroll para cima lentamente, e veja a diferença entre as duas.

```
>Disassemble fde8
FDE8 210040    ld hl,4000
FDEB 11E05B    ld de,5BE0
FDEE D5        push de
FDEF E5        push hl
FDF0 3E03      ld a,03
FDF2 012000    ld bc,0020
FDF5 C5        push bc
FDF6 E5        push hl
FDF7 EDB0      ldir
FDF9 D1        pop de
FDFA 0EE0      ld c,E0
FDFC EDB0      ldir
FDFE 0607      ld b,07
FE00 09        add hl,bc
FE01 3D        dec a
FE02 C1        pop bc
FE03 20F0      jr nz,FDF5
FE05 3A325B    ld a,(5B32)
FE08 FE01      cp 01
FE0A 3811      jr c,FE1D
FE0C 2007      jr nz,FE15
FE0E 21E05B    ld hl,5BE0
FE11 EDB0      ldir
FE13 1808      jr FE1D
FE15 41        ld b,c
FE16 3A335B    ld a,(5B33)
FE19 12        ld (de),a
FE1A 13        inc de
FE1B 10FC      djnz FE19
FE1D E1        pop hl
FE1E D1        pop de
FE1F 24        inc h
FE20 7C        ld a,h
FE21 FE48      cp 48
FE23 38C9      jr c,FDEE
FE25 EB        ex de,hl
FE26 73        ld (hl),e
FE27 2C        inc l
FE28 20FC      jr nz,FE26
FE2A C9        ret
```

Rotina para scroll para cima na tela inteira.

E aqui, a rotina de scroll para a tela inteira para baixo.

```
>Disassemble fde8
FDE8 21FF57    ld hl,57FF
FDEB 11FF5B    ld de,5BFF
FDEE D5        push de
FDEF E5        push hl
FDF0 3E03      ld a,03
FDF2 012000    ld bc,0020
FDF5 C5        push bc
FDF6 E5        push hl
```



```
FDF7 EDB8      lddr
FDF9 D1        pop de
FDFA 0EE0      ld c,E0
FDFC EDB8      lddr
FDFE 06F9      ld b,F9
FE00 09        add hl,bc
FE01 C1        pop bc
FE02 3D        dec a
FE03 20F0      jr nz,FDF5
FE05 3A325B    ld a,(5B32)
FE08 FE01      cp 01
FE0A 3811      jr c,FE1D
FE0C 2007      jr nz,FE15
FE0E 21FF5B    ld hl,5BFF
FE11 EDB8      lddr
FE13 1808      jr FE20
FE15 41        ld b,c
FE16 3A335B    ld a,(5B33)
FE19 12        ld (de),a
FE1A 1B        dec de
FE1B 10FC      djnz FE19
FE1D E1        pop hl
FE1E D1        pop de
FE1F 25        dec h
FE20 7C        ld a,h
FE21 FE50      cp 50
FE23 30C9      jr nc,FDEE
FE25 AF        xor a
FE26 0620      ld b,20
FE28 12        ld (de),a
FE29 1B        dec de
FE2A 10FC      djnz FE28
FE2C C9        ret
```

ROTINA DE LIMPAR A TELA (CLS) DE UMA MANEIRA DIFERENTE

Aqui estão duas rotinas para limpar a tela, só que de uma maneira diferente, e muito mais bonita, além de interessante.

Imagine que a sua tela, os desenhos, textos, caracteres e gráficos estejam sobre uma toalha que você enrola de um lado ou de outro, para limpar.

Pois é exatamente isso que estas rotinas fazem. O papel dessa toalha imaginária que limpa a tela.

A primeira rotina enrola a toalha para a direita, e a seguinte enrola em sentido inverso.

Você pode variar a velocidade da toalha, apenas com um simples POKE, que controla o comprimento de uma pausa após a limpeza de cada coluna da tela.

Para tanto, digite POKE END + 5, comprimento da pausa. Note que esse valor de comprimento da pausa, se for 0, corresponde a uma PAUSE 256. Para remover essa pausa, digite POKE END + 6, 0. Para voltar ao normal, dê POKE + 6, 118.

Lembre-se que, para utilizar estas rotinas, evidentemente você deve ter alguma coisa impressa na tela, para poder limpar.

Chame-as de maneira usual, através de RAND USR END ou LET B = USR END.

A rotina abaixo faz o movimento da esquerda para a direita, e a seguinte faz o movimento inverso, ou seja, da direita para a esquerda.

```
>Disassemble fde8
FDE8 2E00      ld l,00
FDEA 2640      ld h,40
FDEC 0603      ld b,03
FDEE 76        halt
FDEF 10FD      djnz FDEE
FDF1 4E        ld c,(hl)
FDF2 3600      ld (hl),00
FDF4 0608      ld b,08
FDF6 CB19      rr c
FDF8 17        rla
FDF9 10FB      djnz FDF5
FDFB 23        inc hl
FDFC B6        or (hl)
FDFD 77        ld (hl),a
FDFE 111F00    ld de,001F
FE01 19        add hl,de
FE02 7C        ld a,h
FE03 FE58      cp 58
FE05 20EA      jr nz,FDF1
FE07 13        inc de
FE08 0616      ld b,16
FE0A 3A8D5C    ld a,(5C8D)
FE0D 77        ld (hl),a
FE0E 19        add hl,de
FE0F 10FC      djnz FE0D
FE11 3A845C    ld a,(5C84)
FE14 77        ld (hl),a
FE15 19        add hl,de
FE16 77        ld (hl),a
FE17 19        add hl,de
FE18 2C        inc l
FE19 7D        ld a,l
FE1A FE1F      cp 1F
```



```
FE1C 20CC      jr nz,FDEA
FE1E 2640      ld h,40
FE20 06C0      ld b,C0
FE22 72        ld (hl),d
FE23 19        add hl,de
FE24 10FC      djnz FE22
FE26 3A8D5C    ld a,(5C8D)
FE29 0616      ld b,16
FE2B 77        ld (hl),a
FE2C 19        add hl,de
FE2D 10FC      djnz FE2B
FE2F 3A845C    ld a,(5C84)
FE32 77        ld (hl),a
FE33 19        add hl,de
FE34 77        ld (hl),a
FE35 C9        ret

>Disassemble fde8
FDE8 2E1F      ld l,1F
FDEA 2640      ld h,40
FDEC 0603      ld b,03
FDEE 76        halt
FDEF 10FD      djnz FDEE
FDF1 4E        ld c,(hl)
FDF2 3600      ld (hl),00
FDF4 0608      ld b,08
FDF6 CB19      rr c
FDF8 17        rla
FDF9 10FB      djnz FDF6
FDFB 2B        dec hl
FDFC B6        or (hl)
FDFD 77        ld (hl),a
FDFE 112100    ld de,0021
FE01 19        add hl,de
FE02 7C        ld a,h
FE03 FE58      cp 58
FE05 20EA      jr nz,FDF1
FE07 1B        dec de
FE08 0616      ld b,16
FE0A 3A8D5C    ld a,(5C8D)
FE0D 77        ld (hl),a
FE0E 19        add hl,de
FE0F 10FC      djnz FE0D
FE11 3A485C    ld a,(5C48)
FE14 77        ld (hl),a
FE15 19        add hl,de
FE16 77        ld (hl),a
FE17 19        add hl,de
FE18 2D        dec l
FE19 7D        ld a,l
FE1A FE1F      cp 1F
FE1C 20CF      jr nz,FDED
FE1E 2640      ld h,40
FE20 06C0      ld b,C0
FE22 72        ld (hl),d
FE23 19        add hl,de
FE24 10FC      djnz FE22
```

```
FE26 3A8D5C    ld a,(5C8D)
FE29 0616      ld b,16
FE2B 77        ld (hl),a
FE2C 19        add hl,de
FE2D 10FC      djnz FE2B
FE2F 3A485C    ld a,(5C48)
FE32 77        ld (hl),a
FE33 19        add hl,de
FE34 77        ld (hl),a
FE35 C9        ret
```

ROTINAS DE ESPELHO

Aqui estão duas pequenas rotinas cujo efeito é o de refletir cada caractere da tela, a partir de um eixo imaginário "y" (vertical), criando o efeito de um "espelho", e como o resultado é uma reflexão do caractere, pode-se retornar ao estado normal de impressão, reacionando-se a rotina específica.

A primeira rotina:

```
>Disassemble fde8
FDE8 210040    ld hl,4000
FDEB 4E        ld c,(hl)
FDEC 0608      ld b,08
FDEE CB11      rl c
FDF0 1F        rra
FDF1 10FB      djnz FDEE
FDF3 77        ld (hl),a
FDF4 23        inc hl
FDF5 7C        ld a,h
FDF6 FE58      cp 58
FDF8 20F1      jr nz,FDEB
FDFA C9        ret
```

Para utilizá-la, digite RAND USR END.

Você pode alterar a rotina para que ela trabalhe em apenas uma terça parte da tela. As alterações são as seguintes:

Área afetada	POKE END + 2,	POKE END + 15,
Tela toda	64	88
1/3 superior	64	72
1/3 central	72	80
1/3 inferior	80	88
2/3 superior	64	80
2/3 inferior	72	88

A rotina a seguir vai refletir qualquer número de UDG consecutivo.

Para usá-la, precisamos numerar os caracteres gráficos. Para facilitar, utilizei UDG A = 1, UDG B = 2, ..., e UDG U = 21. A tabela:

A–1	F–6	K–11	P–16	U–21
B–2	G–7	L–12	Q–17	
C–3	H–8	M–13	R–18	
D–4	I–9	N–14	S–19	
E–5	J–10	O–15	T–20	

Para utilizar esta rotina em caracteres gráficos, você deve dar primeiramente POKE (END + 4), número do UDG (o primeiro a ser refletido), e POKE (END + 14), número de caracteres a serem refletidos.

A segunda rotina:

```
>Disassemble fde8
FDE8 2A7B5C    ld hl,(5C7B)
FDEB 3E01      ld a,01
FDED 3D        dec a
FDEE 87        add a,a
FDEF 87        add a,a
FDF0 85        add a,l
FDF1 5F        ld e,a
FDF2 1600      ld d,00
FDF4 19        add hl,de
FDF5 3E15      ld a,15
FDF7 87        add a,a
FDF8 87        add a,a
FDF9 87        add a,a
FDFA 4F        ld c,a
FDFB 0608      ld b,08
FDFD 5E        ld e,(hl)
FDFE CB13      rl e
FE00 1F        rra
FE01 10FB      djnz FDFE
FE03 77        ld (hl),a
FE04 23        inc hl
FE05 0D        dec c
FE06 20F3      jr nz,FDFB
FE08 C9        ret
```

ROTINA PARA LIMPAR A TELA, COM EFEITO "FADE-OUT"

Esta é uma nova técnica de limpar a tela toda. Ela tem o efeito do "fade-out" do cinema ou da televisão, ou seja, apaga vagarosamente cada elemento impresso na tela.

Não é necessário nenhum POKE para fazê-la funcionar.

Basta digitá-la, e, através do comando RAND USR END você verá a sua execução.

A rotina:

```
>Disassemble fde8
FDE8 11FE08     ld de,08FE
FDEB 7B         ld a,e
FDEC 07         rlca
FDED 07         rlca
FDEE 07         rlca
FDEF 5F         ld e,a
FDF0 210040     ld hl,4000
FDF3 011800     ld bc,0018
FDF6 7E         ld a,(hl)
FDF7 A3         and e
FDF8 77         ld (hl),a
FDF9 23         inc hl
FDFA 10FA       djnz FDF6
FDFC 0D         dec c
FDFD 20F7       jr nz,FDF6
FDFF 15         dec d
FE00 20E9       jr nz,FDEB
FE02 3A8D5C     ld a,(5C8D)
FE05 77         ld (hl),a
FE06 54         ld d,h
FE07 5D         ld e,l
FE08 13         inc de
FE09 01C002     ld bc,02C0
FE0C EDB0       ldir
FE0E 3A485C     ld a,(5C48)
FE11 77         ld (hl),a
FE12 0E3F       ld c,3F
FE14 EDB0       ldir
FE16 C9         ret
```

ROTINA SCREENs 2

Se você já tentou usar a função SCREENS no TK 90X para reconhecer um caractere UDG, deve ter percebido, que por via direta em Basic, não há condição, e o resultado é um caractere vazio. Isso pode ser ilustrado pelo simples programa:

```
10 PRINT AT 10, 15 ; " "
20 PRINT AT 1, 1; "O caractere da posição 10, 15 é " ;
   SCREENS (0, 0)
```

Neste caso o caractere que a função não reconhece é o CHRS (137), mas o mesmo se aplica a todos os caracteres de código entre 129 a 164. Esta

deficiência de SCREENS faz com que ela não seja utilizada com toda a sua potencialidade, a não ser que o usuário faça as devidas modificações nas variáveis de sistema, para permitir que a função reconheça novos caracteres...

E é por esta razão, que introduzi esta rotina no livro.

Para utilizá-la, você primeiro deve ter as coordenadas da posição em questão (de 0, 0 até 23, 31), e através de POKE, armazená-las nos endereços 23354 e 23355 respectivamente. Por exemplo, para a posição (10, 21), você precisa:

```
POKE 23354, 10
POKE 23355, 21
```

Agora sim, chame a rotina da maneira usual.

Mas, se você deseja usar esta rotina mais de uma vez, algumas providências devem ser tomadas.

Primeiro, defina uma variável, como LET screen = END (endereço inicial da rotina).

A seguir, você usa a função USR para retornar o caractere na posição (L, C), de linha e coluna.

Por exemplo:

```
POKE 23354, 5 :  POKE 23355, 10
LET screen = END
LET aS = CHRS USR screen
IF USR SCREEN = 144 THEN PRINT "O caractere da posição
5, 10 é um A gráfico"
PRINT AT 0, 0; CHRS USR screen
```

A seguir a rotina:

```
.Disassemble fde8
FDE8 ED4B3A5B   ld bc,(5B3A)
FDEC 79         ld a,c
FDED E618       and 18
FDEF C640       add a,40
FDF1 67         ld h,a
FDF2 79         ld a,c
FDF3 87         add a,a
FDF4 87         add a,a
FDF5 87         add a,a
FDF6 87         add a,a
FDF7 87         add a,a
FDF8 80         add a,b
```

```
FDF9 6F         ld l,a
FDFA 0E20       ld c,20
FDFC ED5B365C   ld de,(5C36)
FE00 14         inc d
FE01 E5         push hl
FE02 0608       ld b,08
FE04 1A         ld a,(de)
FE05 BE         cp (hl)
FE06 2006       jr nz,FE0E
FE08 24         inc h
FE09 13         inc de
FE0A 10F8       djnz FE04
FE0C E1         pop hl
FE0D C9         ret
FE0E E1         pop hl
FE0F 13         inc de
FE10 10FD       djnz FE0F
FE12 0C         inc c
FE13 79         ld a,c
FE14 FEA5       cp A5
FE16 280C       jr z,FE24
FE18 FE80       cp 80
FE1A 20E5       jr nz,FE01
FE1C 0E90       ld c,90
FE1E ED5B7B5C   ld de,(5C7B)
FE22 18DD       jr FE01
FE24 E5         push hl
FE25 0608       ld b,08
FE27 7E         ld a,(hl)
FE28 3C         inc a
FE29 2810       jr z,FE3B
FE2B 3D         dec a
FE2C 280D       jr z,FE3B
FE2E FE0F       cp 0F
FE30 2809       jr z,FE3B
FE32 FEF0       cp F0
FE34 2805       jr z,FE3B
FE36 010000     ld bc,0000
FE39 E1         pop hl
FE3A C9         ret
FE3B 24         inc h
FE3C 10E9       djnz FE27
FE3E E1         pop hl
FE3F 0E80       ld c,80
FE41 1601       ld d,01
FE43 1E0F       ld e,0F
FE45 E5         push hl
FE46 0604       ld b,04
FE48 7E         ld a,(hl)
FE49 A3         and e
FE4A BB         cp e
FE4B 2006       jr nz,FE53
FE4D 24         inc h
FE4E 10F8       djnz FE48
FE50 79         ld a,c
FE51 82         add a,d
FE52 4F         ld c,a
```



```
FE53 CB22       sla d
FE55 7B         ld a,e
FE56 87         add a,a
FE57 87         add a,a
FE58 87         add a,a
FE59 87         add a,a
FE5A 5F         ld e,a
FE5B E1         pop hl
FE5C 20E7       jr nz,FE45
FE5E CB62       bit 4,d
FE60 2803       jr z,FE65
FE62 0600       ld b,00
FE64 C9         ret
FE65 CBD4       set 2,h
FE67 18DA       jr FE43
```

ROTINA PARA PROCURAR E ALTERAR CARACTERES EM LISTAGENS DE PROGRAMAS BASIC

Com esta rotina, você pode procurar em um programa em Basic, instantaneamente, por um caractere qualquer, ou mesmo uma palavra chave, e, através de POKEs, alterar aquele conteúdo. São necessários dois POKEs:

POKE 23352, CODE "caractere a encontrar"
POKE 23353, CODE "caractere novo"

Por exemplo, para se alterar todos os símbolos "$" para "&", você deve digitar:

POKE 23352, código de $
POKE 23353, código de & e
LET t = USR END

Nota: a rotina ignora qualquer declaração em uma linha de programa após a palavra chave Basic, REM (de remarks). Portanto, se você tiver uma declaração REM com códigos de máquina, e quiser alterá-los, esta rotina não fará a tarefa.

```
;Disassemble fde8
FDE8 2A535C     ld hl,(5C53)
FDEB ED5B4B5C   ld de,(5C4B)
FDEF EB         ex de,hl
FDF0 A7         and a
FDF1 ED52       sbc hl,de
FDF3 C8         ret z
FDF4 EB         ex de,hl
FDF5 23         inc hl
```

```
FDF6 23          inc hl
FDF7 5E          ld e,(hl)
FDF8 23          inc hl
FDF9 56          ld d,(hl)
FDFA 23          inc hl
FDFB 7E          ld a,(hl)
FDFC FEEA        cp EA
FDFE 282D        jr z,FE2D
FE00 0E00        ld c,00
FE02 7E          ld a,(hl)
FE03 FE22        cp 22
FE05 2001        jr nz,FE08
FE07 0C          inc c
FE08 3A385B      ld a,(5B38)
FE0B BE          cp (hl)
FE0C 2004        jr nz,FE12
FE0E 3A395B      ld a,(5B39)
FE11 77          ld (hl),a
FE12 7E          ld a,(hl)
FE13 23          inc hl
FE14 1B          dec de
FE15 CB41        bit 0,c
FE17 20E9        jr nz,FE02
FE19 FE0E        cp 0E
FE1B 2006        jr nz,FE23
FE1D 0605        ld b,05
FE1F 23          inc hl
FE20 1B          dec de
FE21 10FC        djnz FE1F
FE23 FE3A        cp 3A
FE25 2804        jr z,FE2B
FE27 FE0D        cp 0D
FE29 20D7        jr nz,FE02
FE2B 18BE        jr FDEB
FE2D ED4B385B    ld bc,(5B38)
FE31 79          ld a,c
FE32 BE          cp (hl)
FE33 2001        jr nz,FE36
FE35 70          ld (hl),b
FE36 19          add hl,de
FE37 18B2        jr FDEB
```

ROTINA PARA DELETAR LINHAS BASIC

Esta é outra rotina utilitária para programadores em linguagem Basic. Ela permite que você delete qualquer parte de um programa, desde a sua primeira linha, até o final, instantaneamente. Obviamente, para utilizá-la, você deve especificar dois valores: os números da primeira e da última linha a serem deletadas.

Vamos chamar a primeira linha a ser deletada de P, e a última de U. Os POKEs necessários são:

POKE 23357, P - 256 x INT (P/256)
POKE 23358, INT (P/256)
POKE 23359, U - 256 x INT (U/256)
POKE 23360, INT (U/256)

Ambos os valores são incluídos no bloco a ser deletado, e o segundo e/ou o quarto POKE pode ser omitido se o número da linha correspondente for menor que 256.

Chame a rotina através de RAND USR END ou LET E = USR END.

Se você quiser deletar a partir de uma determinada linha até o final do programa, então qualquer número de linha maior que o número da última linha de uma listagem (9999) serve como segundo valor. Por exemplo:

POKE 23360, 40

Porque 40 x 256 = 10240, seguramente maior que o maior número de linha admissível, que é 9999.

A seguir, a rotina:

```
:Disassemble fde8
FDE8 2A535C      ld hl,(5C53)
FDEB ED4B3D5B    ld bc,(5B3D)
FDEF EB          ex de,hl
FDF0 2A4B5C      ld hl,(5C4B)
FDF3 A7          and a
FDF4 ED52        sbc hl,de
FDF6 C8          ret z
FDF7 EB          ex de,hl
FDF8 56          ld d,(hl)
FDF9 23          inc hl
FDFA 5E          ld e,(hl)
FDFB EB          ex de,hl
FDFC ED42        sbc hl,bc
FDFE EB          ex de,hl
FDFF 3008        jr nc,FE09
FE01 23          inc hl
FE02 5E          ld e,(hl)
FE03 23          inc hl
FE04 56          ld d,(hl)
FE05 23          inc hl
FE06 19          add hl,de
FE07 18E5        jr FDEF
FE09 E5          push hl
FE0A ED4B3F5B    ld bc,(5B3F)
FE0E 23          inc hl
FE0F 5E          ld e,(hl)
FE10 23          inc hl
FE11 56          ld d,(hl)
```

```
FE12 23          inc hl
FE13 19          add hl,de
FE14 EB          ex de,hl
FE15 2A4B5C      ld hl,(5C4B)
FE18 A7          and a
FE19 ED52        sbc hl,de
FE1B EB          ex de,hl
FE1C 280B        jr z,FE29
FE1E 56          ld d,(hl)
FE1F 23          inc hl
FE20 5E          ld e,(hl)
FE21 EB          ex de,hl
FE22 37          scf
FE23 ED42        sbc hl,bc
FE25 EB          ex de,hl
FE26 38E5        jr c,FE0E
FE28 2B          dec hl
FE29 D1          pop de
FE2A 1B          dec de
FE2B C3E519      jp 19E5
```

ROTINA DE RENUMERAR LINHAS DE PROGRAMAS BASIC

Esta é uma das mais úteis rotinas deste livro, visto que renumera todo e qualquer programa Basic, inclusive comandos como GOTO, GOSUB, RUN, LLIST, LIST, RESTORE e SAVE ... LINE, desde que não haja expressões numéricas do tipo "GOTO a x b/c", porque estas variáveis não são linhas de programa!

Antes de chamar a rotina, obviamente você deve fornecer dois parâmetros: o número da primeira linha, para a qual você quer alterar, e o intervalo entre números de linhas. Isso você faz através de:

POKE 23348, P - 256 x INT (P/256)
POKE 23349, INT (P/256)
POKE 23350, I - 256 x INT (I/256)
POKE 23351, INT (I/256)

onde P é o número da primeira linha, e I é o intervalo entre os números de linhas.

Você também pode omitir o segundo e/ou o quarto POKE, se P ou I forem menores que 256.

Por exemplo, para P = 10 e I = 10:

POKE 23348, 10
POKE 23350, 10
RAND USR END

Por segurança, a rotina incorpora um dispositivo que retorna a Basic com erro 4 – Memória lotada se houver menos que 256 bytes disponíveis na memória.

Ela ignora qualquer declaração que vier após uma declaração REM, dentro de uma linha de programa.

A velocidade de trabalho dela é de cerca de 2K de programa por segundo, e utiliza os três bytes disponíveis dentro da área de variáveis de sistema, portanto, cuidado, não utilize aqueles endereços.

A seguir a rotina de 411 bytes.

```
>Disassemble fde8
FDE8 AF          xor a
FDE9 67          ld h,a
FDEA 6F          ld l,a
FDEB 39          add hl,sp
FDEC ED4B6550    ld bc,(5065)
FDF0 ED42        sbc hl,bc
FDF2 BC          cp h
FDF3 2002        jr nz,FDF7
FDF5 CF          rst 08
FDF6 03          inc bc
FDF7 2A4B5C      ld hl,(5C4B)
FDFA 7E          ld a,(hl)
FDFB 328150      ld (5081),a
FDFE 36FF        ld (hl),FF
FE00 2A535C      ld hl,(5C53)
FE03 7E          ld a,(hl)
FE04 3C          inc a
FE05 285D        jr z,FE64
FE07 010000      ld bc,0000
FE0A 23          inc hl
FE0B 23          inc hl
FE0C 23          inc hl
FE0D E5          push hl
FE0E C5          push bc
FE0F 23          inc hl
FE10 7E          ld a,(hl)
FE11 0E00        ld c,00
FE13 FEEA        cp EA
FE15 283E        jr z,FE55
FE17 08          ex af,af'
FE18 7E          ld a,(hl)
FE19 23          inc hl
FE1A FE22        cp 22
FE1C 2001        jr nz,FE1F
FE1E 0C          inc c
FE1F CB41        bit 0,c
FE21 20F4        jr nz,FE17
FE23 FECA        cp CA
FE25 2006        jr nz,FE2D
FE27 08          ex af,af'
```

```
FE28 FEEE       cp EE
FE2A 205D       jr nz,FE88
FE2C 08         ex af,af'
FE2D FE3A       cp 3A
FE2F 28DF       jr z,FE10
FE31 FEF0       cp F0
FE33 2853       jr z,FE88
FE35 FE0E       cp 0E
FE37 2004       jr nz,FE3D
FE39 110500     ld de,0005
FE3C 19         add hl,de
FE3D FEEC       cp EC
FE3F 2847       jr z,FE88
FE41 FEED       cp ED
FE43 2843       jr z,FE88
FE45 FEE5       cp E5
FE47 283F       jr z,FE88
FE49 FEF7       cp F7
FE4B 283B       jr z,FE88
FE4D FEE1       cp E1
FE4F 2837       jr z,FE88
FE51 FE0D       cp 0D
FE53 20C2       jr nz,FE17
FE55 C1         pop bc
FE56 E1         pop hl
FE57 56         ld d,(hl)
FE58 2B         dec hl
FE59 5E         ld e;(hl)
FE5A EB         ex de,hl
FE5B 09         add hl,bc
FE5C EB         ex de,hl
FE5D 73         ld (hl),e
FE5E 23         inc hl
FE5F 72         ld (hl),d
FE60 23         inc hl
FE61 19         add hl,de
FE62 189F       jr FE03
FE64 ED5B345B   ld de,(5B34)
FE68 ED4B365B   ld bc,(5B36)
FE6C 2A535C     ld hl,(5C53)
FE6F 7E         ld a,(hl)
FE70 3C         inc a
FE71 2005       jr nz,FE78
FE73 3A815C     ld a,(5C81)
FE76 77         ld (hl),a
FE77 C9         ret
FE78 72         ld (hl),d
FE79 23         inc hl
FE7A 73         ld (hl),e
FE7B 23         inc hl
FE7C EB         ex de,hl
FE7D 09         add hl,bc
FE7E EB         ex de,hl
FE7F D5         push de
FE80 5E         ld e,(hl)
FE81 23         inc hl
```



```
FE82 56         ld d,(hl)
FE83 23         inc hl
FE84 19         add hl,de
FE85 D1         pop de
FE86 18E7       jr FE6F
FE88 7E         ld a,(hl)
FE89 FE3A       cp 3A
FE8B 2882       jr z,FE0F
FE8D FE0D       cp 0D
FE8F 28C4       jr z,FE55
FE91 23         inc hl
FE92 FE20       cp 20
FE94 38F2       jr c,FE88
FE96 2B         dec hl
FE97 22B05C     ld (5CB0),hl
FE9A 0600       ld b,00
FE9C 7E         ld a,(hl)
FE9D 23         inc hl
FE9E FE0E       cp 0E
FEA0 282E       jr z,FED0
FEA2 FE20       cp 20
FEA4 38F6       jr c,FE9C
FEA6 FE3A       cp 3A
FEA8 3007       jr nc,FEB1
FEAA FE30       cp 30
FEAC 3803       jr c,FEB1
FEAE 04         inc b
FEAF 18EB       jr FE9C
FEB1 7E         ld a,(hl)
FEB2 FE22       cp 22
FEB4 2001       jr nz,FEB7
FEB6 0C         inc c
FEB7 CB41       bit 0,c
FEB9 2010       jr nz,FECB
FEBB FE3A       cp 3A
FEBD 28CC       jr z,FE8B
FEBF FE0D       cp 0D
FEC1 28C8       jr z,FE8B
FEC3 FE0E       cp 0E
FEC5 2004       jr nz,FECB
FEC7 110500     ld de,0005
FECA 19         add hl,de
FECB 23         inc hl
FECC 18E3       jr FEB1
FECE 1883       jr FE53
FED0 110500     ld de,0005
FED3 19         add hl,de
FED4 7E         ld a,(hl)
FED5 FE0D       cp 0D
FED7 2804       jr z,FEDD
FED9 FE3A       cp 3A
FEDB 20D4       jr nz,FEB1
FEDD 2B         dec hl
FEDE 2B         dec hl
FEDF C5         push bc
FEE0 E5         push hl
FEE1 56         ld d,(hl)
```

```
FEE2 2B         dec hl
FEE3 5E         ld e,(hl)
FEE4 ED4B345B   ld bc,(5B34)
FEE8 C5         push bc
FEE9 2A535C     ld hl,(5C53)
FEEC 7E         ld a,(hl)
FEED 3C         inc a
FEEE 281D       jr z,FF0D
FEF0 46         ld b,(hl)
FEF1 23         inc hl
FEF2 4E         ld c,(hl)
FEF3 23         inc hl
FEF4 EB         ex de,hl
FEF5 E5         push hl
FEF6 37         scf
FEF7 ED42       sbc hl,bc
FEF9 E1         pop hl
FEFA 3811       jr c,FF0D
FEFC EB         ex de,hl
FEFD E3         ex (sp),hl
FEFE ED4B365B   ld bc,(5B36)
FF02 09         add hl,bc
FF03 E3         ex (sp),hl
FF04 4E         ld c,(hl)
FF05 23         inc hl
FF06 46         ld b,(hl)
FF07 23         inc hl
FF08 09         add hl,bc
FF09 18E1       jr FEEC
FF0B 18C1       jr FECE
FF0D D1         pop de
FF0E E1         pop hl
FF0F 72         ld (hl),d
FF10 2B         dec hl
FF11 73         ld (hl),e
FF12 3E01       ld a,01
FF14 21F8FF     ld hl,FFF8
FF17 19         add hl,de
FF18 300F       jr nc,FF29
FF1A 3C         inc a
FF1B 219CFF     ld hl,FF9C
FF1E 19         add hl,de
FF1F 3008       jr nc,FF29
FF21 3C         inc a
FF22 2118FC     ld hl,FC18
FF25 19         add hl,de
FF26 3001       jr nc,FF29
FF28 3C         inc a
FF29 C1         pop bc
FF2A E1         pop hl
FF2B F5         push af
FF2C D5         push de
FF2D ED5B805C   ld de,(5C80)
FF31 D5         push de
FF32 90         sub b
FF33 281C       jr z,FF51
FF35 4F         ld c,a
```



```
FF36 D5         push de
FF37 3809       jr c,FF42
FF39 0600       ld b,00
FF3B 09         add hl,bc
FF3C E3         ex (sp),hl
FF3D CD5516     call 1655
FF40 180E       jr FF50
FF42 2F         cpl
FF43 3C         inc a
FF44 5F         ld e,a
FF45 1600       ld d,00
FF47 06FF       ld b,FF
FF49 09         add hl,bc
FF4A E3         ex (sp),hl
FF4B EB         ex de,hl
FF4C 19         add hl,de
FF4D CDE519     call 19E5
FF50 E1         pop hl
FF51 C1         pop bc
FF52 D1         pop de
FF53 F1         pop af
FF54 E5         push hl
FF55 210100     ld hl,0001
FF58 3D         dec a
FF59 2810       jr z,FF6B
FF5B E5         push hl
FF5C 2E0A       ld l,0A
FF5E 3D         dec a
FF5F 280A       jr z,FF6B
FF61 E5         push hl
FF62 2E64       ld l,64
FF64 3D         dec a
FF65 2804       jr z,FF6B
FF67 E5         push hl
FF68 21E803     ld hl,03E8
FF6B EB         ex de,hl
FF6C 3E2F       ld a,2F
FF6E A7         and a
FF6F ED52       sbc hl,de
FF71 3C         inc a
FF72 30FB       jr nc,FF6F
FF74 19         add hl,de
FF75 02         ld (bc),a
FF76 03         inc bc
FF77 1D         dec e
FF78 2006       jr nz,FF80
FF7A 60         ld h,b
FF7B 69         ld l,c
FF7C 4B         ld c,e
FF7D 1C         inc e
FF7E 188B       jr FF0B
FF80 D1         pop de
FF81 18E9       jr FF6C
```

Programa "DESENHISTA"

Eis aqui um programa de desenho muito complexo, com facilidades de utilização muito avançadas, encontradas somente em programas comerciais importados.

Estas facilidades incluem, por exemplo, uma rotina de preenchimento de um polígono qualquer, com uma das 20 texturas pré-selecionadas existentes no programa, desenho de círculos cheios e vazios, linhas inteiras e pontilhadas, velocidade variável do cursor, ampliação do desenho em 4 ou 16 vezes, e armazenamento de duas telas.

Para utilizar o programa, primeiro digite o programa de controle da listagem 1, e grave-o com "auto" run a partir da linha 500. (SAVE "DESENHISTA" LINE 500.)

Não faça alterações nele, exceto no comando LOAD da linha 500, se você deseja armazenar tudo no Microdrive, porque a área de programas Basic está muito reduzida, devido ao endereço inicial da área de códigos de máquina.

A seguir digite o programa da LISTAGEM 2, que é um CARREGADOR DE CÓDIGOS HEXADECIMAIS, com um controle de soma de bytes por linha, facilitando a conferência da digitação, pois ele automaticamente acusa se houve erro em algum byte digitado.

Nesse caso, você deve digitar:

```
POKE 30456, 128
POKE 30457, 96
```

Para em seguida gravar os códigos com

```
SAVE "códigos" CODE 25000, 5512
```

Repare que o endereço inicial dos códigos encontra-se na locação 25000, e por isso a área Basic não pode ser alterada.

Assim que você terminar de digitar os códigos, se tiver digitado o comando CLEAR citado acima, pode digitar o comando NEW para limpar a área Basic, antes de gravar os códigos. Grave-os duas vezes seguidas e não se esqueça do comando VERIFY.

Para testar o programa, carregue-o com LOAD " ", que carregará automaticamente os códigos. Antes disso, você deve resetar o sistema, ou desligando o micro e tornando-o a ligá-lo, ou através de PRINT USR O, que executa a sub-rotina do endereço O da ROM: limpa toda a memória.

Se você tiver paciência para digitar todos estes códigos, verá que, ao utilizar o programa, o trabalho realmente compensou, pois ele é incrível.

Se você possuir um programa Monitor Disassembler, estude as rotinas em linguagem de máquina, através das mnemônicas, para saber exatamente como se plota um ponto na tela, variando a sua velocidade, ou como se utilizam os atributos de uma posição. Para não se perder nestes valores malucos, preste atenção nas instruções em linguagem de máquina, do tipo "CP ... ", ou "OR ... ", ou similares que são as equivalentes ao comando "IF ... THEN ... " da Basic, que executam a função de aguardar a pressão sobre determinadas teclas.

Com o programa rodando, surge um menu de 6 opções, numeradas de 0 a 5. A opção 0 permite que você utilize o programa, no sentido de poder executar seus desenhos; a opção 5 faz um retorno da linguagem de máquina para a Basic, e as outras são opções de armazenamento ou carregamento de códigos, que representam desenhos.

Se você pressionar 0, a tela ficará toda vazia, com a exceção de uma pequena janela de informações no canto superior esquerdo. Assumindo que não existe nada na memória do micro, antes de você carregar o programa, os atributos da tela terão valor 0, e você não poderá utilizar o cursor. Pressione CAPS SHIFT e V e selecione a opção 3 para limpar a tela.

Como usar o programa

Para desenhar, obviamente a primeira coisa que você deve saber é movimentar o cursor. As teclas ao redor da tecla S escarregam-se disso, da seguinte maneira:

```
Q  W  E
 ↖ ↑ ↗
A ← S → D
 ↙ ↓ ↘
Z  X  C
```

Ou seja, pressionando a tecla A, você leva o cursor para a esquerda, na horizontal.

Estas teclas, quando pressionadas em conjunto com SYMBOL SHIFT, rolam a tela nas respectivas direções, e com CAPS SHIFT, movem a janela de ampliação da tela.

O cursor pode ser utilizado em quatro modos de desenho:

1. SET – desenha o ponto ou o que você quiser;
2. RES – equivale a INVERSE 1, ou seja, se tiver algo desenhado sob o cursor, este o apagará;
3. XOR – inverte pixéis; equivale a OVER 1; e
4. SALT – permite que você movimente o cursor sem alterar o que está embaixo – você "salta" por cima.

Estes modos de desenho são obtidos respectivamente com as teclas P, O, I e U.

Para mudar as cores da tela, você deve mudar a partir do cursor, indicado por TELA, na janela de informação, para atributos do cursor – ATRI. A tecla L executa esta alteração. O cursor de atributos comporta-se exatamente como o cursor de pixéis, com exceção do seu tamanho. Para retornar, pressione a tecla L novamente.

Em modo SET, você vai desenhar com valores para tinta, papel, brilho e intermitência (flash), indicados na janela de informação, na sua última linha. Estes valores podem ser alterados através das teclas numéricas de 0 a 7 que diretamente pressionadas, alteram a tinta; pressionadas com CAPS SHIFT, alteram a cor do papel; as teclas 8 e 9 sozinhas, ou com CAPS SHIFT, acionam ou desligam o brilho e a intermitência.

Note que quando utilizando o cursor de atributos, pressionando-se SYMBOL SHIFT juntamente com as teclas que movimentam o cursor, os atributos da tela irão rolar. (Scroll)

Finalmente, se você deseja aumentar ou diminuir a velocidade do cursor, pressione SYMBOL SHIFT com o valor da velocidade desejada, começando em 0 (sem repetição) e aumentando até chegar à tecla 1.

Para as rotinas de desenho de segmento de retas e circunferências, você deve especificar dois pontos que, no caso do segmento, são seus extremos e, no caso da circunferência, especificam o comprimento do raio, a partir do centro.

Para fixar um desses pontos, pressione CAPS SHIFT junto com ENTER, que define a posição do cursor como último ponto desenhado. Em seguida, mova o cursor para o outro extremo da linha, ou o ponto da circunferência, e repita o procedimento.

Se você pressionar SYMBOL SHIFT e ENTER juntas vai ver o último ponto desenhado piscar, procedimento este muito útil quando se esquece qual foi a última posição plotada.

Para traçar uma linha, evidente que em modo SET, pressione CAPS SHIFT e a tecla J. Para traçar uma circunferência, pressione CAPS SHIFT e H. Se você pressionar SYMBOL SHIFT e H, o círculo será preenchido com a cor da tinta especificada. Se pressionar SYMBOL SHIFT e J, a linha especificada será desenhada em modo pontilhado. Para alterar os valores para este tipo de desenho, inicialmente fixados em 10101010, pressione CAPS SHIFT e K, e digite o novo valor, bit a bit.

Para facilitar a utilização dos atributos no seu desenho, já que é muito mais fácil fazer o desenho em preto e branco, para depois colori-lo, existem duas malhas que se sobrepõem ao desenho, para mostrar-lhe com exatidão o arquivo de atributos. Pressione G para uma malha em preto e branco, ou CAPS SHIFT e G para uma malha com as cores da tela.

Agora vamos ver a rotina de preenchimento de figuras. Mova o cursor (em modo SALT) para dentro da sua figura, e pressione CAPS SHIFT e F, para preenchê-la. Existem 20 texturas predefinidas no programa, que você pode utilizar através de números. Para as dez primeiras, você seleciona diretamente pressionando uma tecla entre 1 e 0; para as outras 10, você seleciona através de CAPS SHIFT e o número da textura desejada. Para utilizá-las no preenchimento de uma figura, você deve pressionar SYMBOL SHIFT junto com a tecla F, seguida do número da textura.

Mas você também pode definir o seu próprio padrão de desenho, que consiste de uma área de tamanho idêntico ao de um atributo definido de uma maneira muito similar a UDG. Para tanto, crie a sua textura, no tamanho apropriado, e com o cursor dentro dessa área, pressione CAPS SHIFT e P. Selecione, então, o padrão como na rotina de preenchimento de figuras, para em seguida pressionar SYMBOL SHIFT com o número da textura a ser substituído, a fim de que o procedimento tenha sucesso.

Similar a esse método de definição de texturas, existe um modo de definição de UDG – para selecioná-lo, pressione CAPS SHIFT e O. Responda à pergunta "UDG ?", pressionando a tecla correspondente ao caractere gráfico que você irá definir na faixa de A até U.

Existem duas áreas de armazenamento de desenhos na memória para que você possa utilizar uma como rascunho, e a outra para o desenho definitivo. Para "gravar" o desenho ou a tela na memória, pressione CAPS SHIFT com a tecla I. Para retornar este desenho, pressione CAPS SHIFT junto com U.

Para escrever na tela, existe o modo texto, obtido através da tecla T. Use as teclas numéricas do cursor, para movimentar o cursor de texto, mas note que a maneira de escrever está invertida: diretamente pressionadas, as teclas resultam em letras maiúsculas, e com CAPS SHIFT resultam em minúsculas. Para sair deste modo, pressione BREAK.

O modo gráfico para UDG, obtém-se pressionando CAPS SHIFT com 9.

A janela de informação

Os dois números que aparecem na sua linha superior, são as coordenadas X e Y do cursor. Estes valores são baseados nos valores normais do comando PLOT x, y da Basic, exceto que nas duas linhas inferiores (22 e 23), as abscissas Y passam a ser negativas.

A segunda linha da janela mostra o modo com o qual você está trabalhando – SET, RES, XOR, SALT ou TEXT e "SE", que significa modo "seguir cursor", em operação, e que será explicado adiante.

A terceira linha da janela mostra o modo do cursor – TELA ou ATRI para cursor normal, GRAF ou nada para cursor de texto, e a ampliação do desenho, cujo padrão é 1. Na última linha desta janela estão as informações sobre tinta, papel, brilho e intermitência – estes últimos, se acionados, aparecem como "B" e "F". Você pode também fazer esta janela sumir, pressionando CAPS SHIFT e N uma vez, e duas vezes para que ela retorne à tela. Normalmente não é preciso fazê-la desaparecer, já que se move automaticamente, procurando não interferir com o seu desenho.

Um dos procedimentos mais úteis do programa é a facilidade de ampliação do seu desenho, em 4 ou 16 vezes.

A área a ser ampliada é chamada janela de ampliação e pode ser obtida pressionando-se CAPS SHIFT e M uma vez, e outra vez para fazê-la sumir. Esta janela pode ser movimentada na tela, através de CAPS SHIFT e as teclas que movem o cursor. SYMBOL SHIFT e M selecionam o tamanho da ampliação, para finalmente através da tecla M sozinha, esta se efetivar.

Se você está desenhando em tamanho ampliado, automaticamente a janela de ampliação segue o cursor, em qualquer direção da tela. Este é o chamado "modo seguir cursor", citado anteriormente. Para desativá-lo, pressione SYMBOL SHIFT junto com a tecla S. Para centralizar sua janela, pressione simplesmente a tecla S, e para colocar o cursor no centro desta, pressione CAPS SHIFT e S.

Para limpar a tela, pressione CAPS SHIFT e V, fazendo surgir o menu de limpeza, e selecione o que limpar.

SYMBOL SHIFT e V selecionam o comando de inversão de atributos, por exemplo, para escrever frases em modo inverso.

Você pode também limpar partes da tela, invertendo-a, contornando-a com uma linha e preenchendo-a ainda em modo inverso, para em seguida retornar ao modo geral.

Para alterar a cor da borda do desenho, pressione B e o número equivalente à cor desejada. A última tecla funcional do programa é R, que traz o menu de armazenamento ou carregamento de desenhos.

RESUMO DAS TECLAS

TECLA	CAPS SHIFT	SYMBOL SHIFT	FUNÇÃO
0 a 5			Menu principal.
Q W E A D Z X C			Movem cursor.
Q a C		x	Rolam a tela nas diversas direções.
Q a C	x		Movem a janela de ampliação.
P			Modo SET.
O			Modo RESET.
I			Modo XOR.
U			Modo SALT.
L			Muda cursor TELA para ATRI e vice-versa.
0 a 7			Alteram cor da tinta.
0 a 7	x		Alteram cor do papel.
8 e 9			Acionam brilho e intermitência.
8 e 9	x		Desligam brilho e intermitência.
0 a 9		x	Alteram velocidade do cursor.
ENTER	x		Fixa extremos da linha e raio e circunferência.
ENTER		x	Mostra piscando o último ponto plotado.
J	x		Traça linha cheia.
J		x	Traça linha pontilhada.
H	x		Traça circunferência vazia.
H		x	Traça circunferência cheia.
K	x		Altera padrão do pontilhado.
G			Insere malha branca e preta.
G	x		Insere malha com as cores da tela.

TECLA	CAPS SHIFT	SYMBOL SHIFT	FUNÇÃO
F	x		Preenche um polígono.
F		x	Permite selecionar textura de preenchimento.
0 a 9			Primeiras dez texturas.
0 a 9	x		Últimas dez texturas.
P	x		Permite criar nova textura.
0 a 9			Substitui textura (entre dez primeiras).
0 a 9	x		Substitui textura (entre dez últimas).
0	x		Permite criar caractere UDG na letra pré-selecionada.
I	x		Grava desenho na memória.
U	x		Traz desenho da memória para a tela.
T			Aciona modo texto.
BREAK			Sai do modo texto.
5 a 8	x		Movem cursor de texto.
9	x		Aciona modo UDG.
N	x		Tira/põe janela de informação.
M	x		Tira/põe janela de ampliação.
M		x	Seleciona tamanho da ampliação.
M			Amplia desenho.
S			Centraliza janela de ampliação.
S		x	Desliga modo de seguir cursor.
S	x		Coloca cursor no centro da janela de ampliação.
V	x		Mostra menu de limpeza.
V		x	Mostra menu de inversão.
R			Mostra menu principal.
B			Altera cor da borda.

Paciência, que o programa é longo, mas compensa.

Muito cuidado, que basta um valor errado e o programa não funcionará: não confunda 8 com B, nem D com O.

De resto, divirta-se!

LISTAGEM 1 – PROGRAMA DE CONTROLE

```
  10 LET a=USR VAL "25383"
  20 LET o=INT (a/VAL "256"): LE
T b=a-VAL "256"*o
  30 IF NOT o THEN CLS : PRINT "
Run para iniciar": STOP
  40 INPUT n$: IF LEN n$>VAL "10
" OR NOT LEN n$ THEN GO TO VAL "
30"
  50 RESTORE VAL "100"+b: READ c
,l: GO SUB VAL "199"+o
  60 RUN
 100 DATA USR "a",VAL "168"
 101 DATA VAL "30285",VAL "160"
 102 DATA VAL "57344",VAL "6912"
 200 SAVE n$CODE c,l: VERIFY n$C
ODE : RETURN
 201 LOAD n$CODE c,l: RETURN
 500 CLEAR 24999: LOAD "c"CODE :
 RANDOMIZE USR VAL "25360": RUN
```

LISTAGEM 2 – PROGRAMA CARREGADOR DOS CÓDIGOS HEXADECIMAIS

```
   1 DEF FN b(a$)=CODE a$-48-7*(
a$>"9")
   2 DEF FN h(a$)=FN b(a$(1))*16
+FN b(a$(2))
   3 DEF FN b$(a)=CHR$ (a+48+7*(
a>9))
   4 DEF FN h$(a)=FN b$(INT (a/1
6))+FN b$(a-16*INT (a/16))
   5 POKE 23658,8
  10 FOR a=25000 TO 30448 STEP 8
  20 PRINT a;" "; LET c=0: LET
a$=" "
  30 FOR g=0 TO 7: IF a$(1)=" "
THEN INPUT LINE a$: LET a$=a$+"
"
  60 LET z=FN h(a$): LET a$=a$(3
 TO ): POKE a+g,z: PRINT " ";FN
h$(z); : LET c=c+z
  70 NEXT g: INPUT "soma:";k: IF
 c<>k THEN PRINT "Erro": GO TO 2
0
  80 PRINT : NEXT a: POKE 30455,
128: POKE 30457,96
  90 SAVE "Desenho"CODE 25000,55
12
```


LISTAGEM DOS CÓDIGOS EM HEXADECIMAL

```
25000 20 00 4D 4E 42 0D 4C 48
                        SOMA = 417
25008 4A 48 50 4F 49 55 59 30
                        SOMA = 600
25016 39 38 37 36 31 32 33 34
                        SOMA = 424
25024 35 51 57 45 52 54 41 53
                        SOMA = 604
25032 44 46 47 01 5A 58 43 56
                        SOMA = 541
25040 FF 00 6D 6E 62 0D 5C 66
                        SOMA = 500
25048 6A 68 70 6F 69 75 79 00
                        SOMA = 776
25056 09 08 07 06 00 00 00 00
                        SOMA = 30
25064 05 71 77 65 72 74 61 73
                        SOMA = 780
25072 64 66 67 00 7A 78 63 76
                        SOMA = 764
25080 20 00 2E 2C 2A 0D 3D 28
                        SOMA = 281
25088 2D 5E 22 3B 7F 5D 5B 5F
                        SOMA = 838
25096 29 28 27 26 21 40 23 24
                        SOMA = 326
25104 25 20 20 20 3C 3E 7E 7C
                        SOMA = 505
25112 5C 75 7D 00 3A 60 3F 2F
                        SOMA = 804
25120 00 00 00 00 DE 66 99 75
                        SOMA = 594
25128 51 67 66 65 D3 66 00 00
                        SOMA = 734
25136 00 00 00 00 BA 66 C2 66
                        SOMA = 5864
25144 BE 66 C6 66 00 00 7E 66
                        SOMA = 6020
25152 7E 66 7E 66 7E 66 7E 66
                        SOMA = 9112
25160 7E 66 7E 66 7E 66 7E 66
                        SOMA = 9112
25168 7E 66 8D 65 92 65 97 65
                        SOMA = 5669
25176 00 00 80 71 9C 65 63 6D
                        SOMA = 7105
25184 A1 65 00 00 BC 67 00 00
                        SOMA = 6653
25192 A6 65 AB 65 B0 65 00 00
                        SOMA = 6615
25200 00 00 00 00 E2 66 B2 75
                        SOMA = 6623
25208 51 67 69 65 D3 65 E6 75
                        SOMA = 1050
```

```
25216 8C 73 E8 72 57 70 75 70
                        SOMA = 1029
25224 F8 6F FE 6F 00 00 7A 66
                        SOMA = 948
25232 7A 66 7A 66 7A 66 7A 66
                        SOMA = 896
25240 7A 66 7A 66 7A 66 7A 66
                        SOMA = 896
25248 7A 66 5C 64 61 64 66 64
                        SOMA = 815
25256 00 00 00 00 6B 64 39 6D
                        SOMA = 373
25264 70 64 FE 6E C0 67 00 00
                        SOMA = 871
25272 75 64 7A 64 7F 64 69 6A
                        SOMA = 877
25280 00 00 00 00 E6 65 B2 75
                        SOMA = 627
25288 51 67 70 65 D3 66 00 00
                        SOMA = 710
25296 87 73 E4 72 BA 65 C2 66
                        SOMA = 1176
25304 BE 66 C6 66 00 00 AB 66
                        SOMA = 855
25312 AB 66 AB 66 AB 66 AB 66
                        SOMA = 1092
25320 AB 66 AB 66 AB 66 AB 66
                        SOMA = 1092
25328 AB 66 9D 6D A2 6D A7 6D
                        SOMA = 1086
25336 00 00 00 00 AC 6D DA 66
                        SOMA = 601
25344 B1 6D 15 6F C0 67 00 00
                        SOMA = 713
25352 B6 6D BB 6D C0 6D A5 6A
                        SOMA = 1159
25360 11 00 E0 ED 53 2E 77 CD
                        SOMA = 931
25368 0B 70 AF 32 3B 76 32 F6
                        SOMA = 821
25376 76 3E 18 32 2D 77 C9 ED
                        SOMA = 856
25384 73 FE 76 31 00 E0 CD 6A
                        SOMA = 1055
25392 63 ED 7B FE 76 C9 3A F7
                        SOMA = 1337
25400 76 D3 FE CD 0D 6B CD 2E
                        SOMA = 1159
25408 67 CD 3D 67 CD 26 64 CD
                        SOMA = 1020
25416 04 74 3A 25 77 E6 08 20
                        SOMA = 604
25424 09 3A 2D 77 A7 CC 2E 67
                        SOMA = 751
25432 18 E7 CD A6 67 DD 21 BE
                        SOMA = 1173
25440 63 CD 56 68 A7 28 0F FE
                        SOMA = 1162
```



```
25448 05 01 FF 00 C8 F5 CD 0D
                        SOMA = 924
25456 6B DD 21 94 63 CD 56 68
                        SOMA = 1003
25464 C1 4F C5 CD 0D 6B DD 21
                        SOMA = 1048
25472 8C 63 CD B6 68 CD 62 68
                        SOMA = 1137
25480 C1 C9 4F C9 07 03 4E 4F
                        SOMA = 841
25488 4D 45 3F 27 11 07 53 41
                        SOMA = 420
25496 56 45 2F 4C 4F 41 44 27
                        SOMA = 529
25504 41 52 45 41 20 55 44 47
                        SOMA = 537
25512 27 50 49 4E 54 41 52 20
                        SOMA = 533
25520 50 41 44 52 41 4F 27 54
                        SOMA = 562
25528 45 4C 41 20 20 27 18 0A
                        SOMA = 347
25536 4F 50 43 4F 45 53 20 27
                        SOMA = 528
25544 45 44 49 54 41 52 20 54
                        SOMA = 557
25552 45 4C 41 20 27 53 41 56
                        SOMA = 515
25560 45 20 50 2F 20 46 49 54
                        SOMA = 487
25568 41 27 4C 4F 41 44 20 20
                        SOMA = 456
25576 44 41 20 20 46 49 54 41
                        SOMA = 489
25584 27 53 41 56 45 20 50 2F
                        SOMA = 501
25592 20 4D 49 43 52 4F 44 52
                        SOMA = 560
25600 49 56 45 27 4C 4F 41 44
                        SOMA = 555
25608 20 20 44 45 20 20 4D 49
                        SOMA = 415
25616 43 52 4F 44 52 49 56 45
                        SOMA = 606
25624 27 53 41 49 20 44 4F 20
                        SOMA = 471
25632 50 52 4F 47 2E 27 21 20
                        SOMA = 462
25640 62 3A 28 77 A7 28 09 21
                        SOMA = 564
25648 70 62 3D 28 03 21 C0 62
                        SOMA = 837
25656 DD 21 20 77 0E 08 DD 7E
                        SOMA = 774
25664 00 06 05 1F 38 0A 23 23
                        SOMA = 178
25672 10 F9 DD 23 0D 20 EF C9
                        SOMA = 1006
```

```
25680 5E 23 56 7A B3 C8 21 2E
                    SOMA = 795
25688 67 E5 EB E9 11 FF FF 18
                    SOMA = 1351
25696 21 11 00 FF 18 1C 11 01
                    SOMA = 375
25704 FF 18 17 11 FF 00 18 12
                    SOMA = 616
25712 11 01 00 18 0D 11 FF 01
                    SOMA = 328
25720 18 08 11 00 01 18 03 11
                    SOMA = 94
25728 01 01 21 94 6B E3 ED 48
                    SOMA = 829
25736 2E 77 79 E6 1F 83 FE 20
                    SOMA = 954
25744 D0 7B 81 4F 7A A7 28 11
                    SOMA = 885
25752 FE 01 28 12 79 D6 20 4F
                    SOMA = 759
25760 30 07 78 D6 08 47 FE E0
                    SOMA = 946
25768 D8 ED 43 2E 77 C9 08 79
                    SOMA = 1015
25776 C6 20 4F 30 F4 78 C6 08
                    SOMA = 927
25784 47 E6 18 FE 18 20 EA C9
                    SOMA = 1070
25792 7B 07 07 07 AA E6 07 AA
                    SOMA = 721
25800 07 07 07 AA E6 F8 AA C9
                    SOMA = 1040
25808 57 07 07 AB E6 E0 AB 5F
                    SOMA = 992
25816 7A 37 1F 37 1F 37 1F AA
                    SOMA = 550
25824 E6 F8 AA E6 F8 57 C9 7C
                    SOMA = 1538
25832 0F 0F 0F E6 03 C6 58 67
                    SOMA = 667
25840 C9 22 2A 77 24 7C E6 07
                    SOMA = 793
25848 C0 7D C6 20 6F 38 05 7C
                    SOMA = 843
25856 D6 08 67 C9 7C 2F E6 18
                    SOMA = 951
25864 C0 2A 2A 77 C9 22 2A 77
                    SOMA = 791
25872 25 7C 2F E6 07 C0 7D D6
                    SOMA = 976
25880 20 6F 38 E8 7C C6 08 67
                    SOMA = 864
25888 C9 7D 07 07 07 AC E6 07
                    SOMA = 756
25896 AC 07 07 07 AC E6 F8 AC
                    SOMA = 1015
25904 57 C9 0C CB 08 D0 7D 3C
                    SOMA = 904
```



```
25912 E6 1F 28 02 2C C9 0D CB
                    SOMA = 764
25920 00 37 C9 0D CB 00 D0 7D
                    SOMA = 805
25928 3D 2F E6 1F 28 02 2D C9
                    SOMA = 657
25936 0C CB 08 37 C9 7D E6 E0
                    SOMA = 1056
25944 6F 7D C6 20 6F D0 7C C6
                    SOMA = 1107
25952 08 67 2F E6 18 C0 26 40
                    SOMA = 706
25960 C9 2A F8 76 22 FA 76 C9
                    SOMA = 1212
25968 CD F4 86 ED 4B FA 76 CD
                    SOMA = 1436
25976 97 69 CD 93 69 CD 16 67
                    SOMA = 1043
25984 20 F1 CD 0A 67 C3 0D 6B
                    SOMA = 906
25992 11 00 00 18 26 11 FF FF
                    SOMA = 606
26000 18 21 11 00 FF 18 1C 11
                    SOMA = 398
26008 01 FF 18 17 11 FF 00 18
                    SOMA = 599
26016 12 11 01 00 18 0D 11 FF
                    SOMA = 345
26024 01 18 08 11 00 01 18 03
                    SOMA = 78
26032 11 01 01 C1 ED 4B F8 76
                    SOMA = 890
26040 3A 3B 76 E6 40 28 0E 7B
                    SOMA = 706
26048 E6 1F 07 07 07 5F 7A E6
                    SOMA = 729
26056 1F 07 07 07 57 78 82 FE
                    SOMA = 843
26064 C0 D0 47 7B 07 CB 12 0F
                    SOMA = 837
26072 81 CB 12 08 CB 1A 17 AA
                    SOMA = 780
26080 E6 01 C0 08 4F 2A F8 76
                    SOMA = 918
26088 ED 43 F8 76 4D 44 CD E6
                    SOMA = 1250
26096 69 C3 00 69 79 07 07 07
                    SOMA = 647
26104 A8 E6 C7 A8 07 07 6F 3A
                    SOMA = 948
26112 3B 76 E6 40 20 16 78 A7
                    SOMA = 812
26120 1F 37 1F A7 1F A8 E6 F8
                    SOMA = 961
26128 A8 67 79 E6 07 3C ED 4B
                    SOMA = 1001
26136 3B 76 47 C9 78 07 07 EE
                    SOMA = 821
```

```
26144 58 E6 03 EE 58 67 ED 4B
                        SOMA = 1062
26152 3B 76 06 00 C9 3A 3B 76
                        SOMA = 619
26160 F5 E6 BF 32 3B 76 CD 3E
                        SOMA = 1160
26168 66 F1 32 3B 76 C9 79 07
                        SOMA = 899
26176 07 07 A8 E6 C7 A8 07 07
                        SOMA = 793
26184 6F 3A 3B 76 E6 40 20 1B
                        SOMA = 699
26192 78 F6 07 0F 0F 0F A8 E6
                        SOMA = 816
26200 F8 A8 67 79 E6 07 3C 47
                        SOMA = 1008
26208 3E FE 0F 10 FD ED 4B 3B
                        SOMA = 971
26216 76 47 C9 78 07 07 EE F8
                        SOMA = 1010
26224 E6 03 EE F8 67 ED 4B 3B
                        SOMA = 1193
26232 76 C9 0E 01 18 02 0E 00
                        SOMA = 374
26240 C5 CD 7F 67 C1 47 FE 08
                        SOMA = 1158
26248 38 0C D6 09 2F 0D 06 7F
                        SOMA = 484
26256 28 0E CB 08 18 0A 06 F8
                        SOMA = 553
26264 0D 20 05 07 07 07 06 C7
                        SOMA = 276
26272 4F 3A 3C 76 A9 A0 A9 32
                        SOMA = 863
26280 3C 76 C9 CD 7F 67 6F 26
                        SOMA = 963
26288 00 11 ED 76 19 7E 32 2D
                        SOMA = 618
26296 77 C9 05 00 18 0A 06 01
                        SOMA = 367
26304 18 05 05 02 18 02 06 03
                        SOMA = 73
26312 3A 3B 76 A8 E6 FC A8 32
                        SOMA = 1103
26320 3B 76 C9 CD A6 67 06 40
                        SOMA = 922
26328 18 0E 06 80 18 0A 06 04
                        SOMA = 216
26336 18 05 06 10 18 02 06 08
                        SOMA = 92
26344 3A 3B 76 A8 32 3B 76 C3
                        SOMA = 825
26352 94 6B 00 00 2A 3B 76 3A
                        SOMA = 532
26360 F6 76 67 22 F2 66 F6 08
                        SOMA = 1099
26368 32 F5 76 7D F6 80 32 3B
                        SOMA = 1022
```



```
26376 76 C9 2A F2 66 7D 32 3B
                        SOMA = 939
26384 76 7C 32 F6 76 C9 3E FE
                        SOMA = 1173
26392 DB FE F6 E1 47 3E 7F DB
                        SOMA = 1423
26400 FE F6 E2 A0 47 3E 81 DB
                        SOMA = 1367
26408 FE F6 E0 A0 3C C9 CD 3A
                        SOMA = 1408
26416 6B CD 04 74 CD 93 69 CD
                        SOMA = 1094
26424 16 67 C8 18 F1 CD 3A 6B
                        SOMA = 960
26432 CD 04 74 CD 93 69 CD 16
                        SOMA = 1009
26440 67 28 F2 CD F6 6B C3 CF
                        SOMA = 1345
26448 6B DD 21 8A 67 CD B6 68
                        SOMA = 1061
26456 CD 62 58 CD 74 67 FE 08
                        SOMA = 1093
26464 30 F9 32 F7 76 D3 FE C3
                        SOMA = 1372
26472 0D 6B 09 03 42 4F 52 44
                        SOMA = 427
26480 41 20 3F 27 CD CF 6B CD
                        SOMA = 923
26488 7F 67 FE FF 28 F6 C9 ED
                        SOMA = 1463
26496 4B 23 77 78 F6 20 06 00
                        SOMA = 633
26504 0F 04 30 FC 78 FE 06 D8
                        SOMA = 915
26512 06 0B 79 F6 20 0F 05 30
                        SOMA = 484
26520 FC 78 FE 06 30 03 3E FF
                        SOMA = 1000
26528 C9 FE 0A D8 AF C9 3A F6
                        SOMA = 1361
26536 76 0F D0 3F 17 32 F6 76
                        SOMA = 841
26544 21 00 FB 11 00 F8 01 00
                        SOMA = 550
26552 03 ED B0 C9 06 01 18 02
                        SOMA = 650
26560 06 02 3A 3B 76 E6 40 C0
                        SOMA = 729
26568 3A F6 76 21 0D 6B E5 E6
                        SOMA = 1034
26576 01 20 D3 C5 21 00 F8 11
                        SOMA = 739
26584 00 FB 01 00 03 ED B0 3A
                        SOMA = 726
26592 F6 76 F6 01 32 F6 76 C1
                        SOMA = 1218
26600 05 20 26 21 01 F8 11 02
                        SOMA = 376
```

```
26608 F8 36 38 2B 36 76 01 1E
                      SOMA = 606
26616 00 ED B0 23 ED A0 2E 00
                      SOMA = 891
26624 01 1F 00 ED B0 21 00 F8
                      SOMA = 726
26632 11 40 F8 01 C0 02 ED B0
                      SOMA = 937
26640 C9 21 00 F8 11 00 40 01
                      SOMA = 564
26648 18 10 7E E6 BF B2 77 23
                      SOMA = 919
26656 7E E6 BF B3 77 23 10 F2
                      SOMA = 1138
26664 06 10 7A 53 5F 0D 20 EA
                      SOMA = 601
26672 C9 CD B6 68 DD 7E 00 D6
                      SOMA = 1253
26680 04 47 48 E5 CD 62 68 E1
                      SOMA = 1008
26688 CD 59 65 CD 59 65 E5 79
                      SOMA = 1140
26696 90 C6 30 CD 72 68 2C CD
                      SOMA = 1062
26704 62 68 E1 10 EE C9 CD 31
                      SOMA = 1136
26712 68 C5 CD 74 67 C1 B9 D8
                      SOMA = 1319
26720 18 F7 3E 47 08 DD 23 DD
                      SOMA = 889
26728 7E 00 FE 27 C8 CD 72 68
                      SOMA = 1042
26736 18 F3 C5 E5 6F 26 00 29
                      SOMA = 683
26744 29 29 11 00 3C 19 EB E1
                      SOMA = 644
26752 E5 CD 8F 68 25 CD E7 64
                      SOMA = 1254
26760 08 77 08 E1 2C C1 C9 06
                      SOMA = 804
26768 08 1A 77 24 13 10 FA C9
                      SOMA = 675
26776 0A 08 4C 49 4D 50 41 27
                      SOMA = 428
26784 4E 41 44 41 27 54 45 4C
                      SOMA = 544
26792 41 20 20 27 41 54 52 49
                      SOMA = 472
26800 27 54 55 44 4F 27 DD 46
                      SOMA = 685
26808 00 DD 4E 01 04 04 21 00
                      SOMA = 341
26816 40 3E 70 08 3E 23 CD 72
                      SOMA = 662
26824 68 10 F9 CD 55 65 3E 23
                      SOMA = 857
26832 CD 72 68 3E 47 08 DD 46
                      SOMA = 855
```



```
26840 00 3E 20 CD 72 68 10 F9
                      SOMA = 782
26848 3E 70 08 3E 23 CD 72 68
                      SOMA = 702
26856 CD 55 65 0D 20 E0 DD 46
                      SOMA = 951
26864 00 04 04 3E 23 CD 72 68
                      SOMA = 526
26872 10 F9 21 42 40 DD 23 C9
                      SOMA = 885
26880 3A 3B 76 E6 20 C4 97 69
                      SOMA = 949
26888 3A 3B 76 E6 04 CA C6 69
                      SOMA = 974
26896 CD 23 69 D8 CD 57 69 D6
                      SOMA = 1174
26904 3A 3B 76 E6 08 CA 05 6A
                      SOMA = 786
26912 C3 2D 6A 2A 2E 77 CD 21
                      SOMA = 791
26920 65 7D 07 07 07 E6 F8 5F
                      SOMA = 820
26928 79 93 38 50 07 4F 0F 0F
                      SOMA = 520
26936 0F 0F E6 1F 2A 38 76 BD
                      SOMA = 596
26944 30 42 78 92 38 3E BC 30
                      SOMA = 734
26952 3B 07 47 3A 3B 76 E6 08
                      SOMA = 610
26960 C8 CB 00 CB 01 A7 C9 3A
                      SOMA = 1033
26968 F8 76 E6 08 C0 C5 2A 02
                      SOMA = 1035
26976 74 CD 21 65 78 92 38 19
                      SOMA = 802
26984 FE 20 30 15 7D E6 1F 07
                      SOMA = 748
26992 07 07 6F 7E 95 38 E0 FE
                      SOMA = 929
27000 38 30 06 CD 99 75 C1 37
                      SOMA = 833
27008 C9 C1 A7 C9 3A 3B 76 E6
                      SOMA = 1227
27016 80 37 C0 CD 63 6D CD 15
                      SOMA = 1015
27024 75 37 C9 ED 4B F8 76 ED
                      SOMA = 1288
27032 5B 3B 76 3A F6 76 E6 04
                      SOMA = 924
27040 C2 DF 70 7B EE 20 E6 FC
                      SOMA = 1404
27048 F6 01 32 3B 76 D5 3E 3F
                      SOMA = 812
27056 32 3C 76 C5 CD 08 69 C1
                      SOMA = 936
27064 D1 3A 3B 76 AB E6 FC AB
                      SOMA = 1268
```

```
27072 5F ED 53 3B 76 C9 CD 57
                      SOMA = 1085
27080 69 D8 CD F4 65 79 E6 40
                      SOMA = 1286
27088 20 22 3E FE 0F 10 FD 47
                      SOMA = 737
27096 7E CB 41 20 01 A0 CB 49
                      SOMA = 863
27104 20 02 A8 2F 77 C9 C5 CD
                      SOMA = 971
27112 3E 66 CD EF 69 C1 C9 79
                      SOMA = 1228
27120 E6 40 28 E4 3A 3C 76 47
                      SOMA = 869
27128 7E CB 41 20 01 AF CB 49
                      SOMA = 878
27136 20 01 A8 77 C9 CD F4 65
                      SOMA = 1071
27144 79 E6 40 20 0F 3E 7E 0F
                      SOMA = 665
27152 10 FD 47 CD D8 69 47 CD
                      SOMA = 1142
27160 F1 64 70 C9 7D E6 DE 6F
                      SOMA = 1342
27168 CD F4 69 23 77 11 20 00
                      SOMA = 757
27176 19 77 2D 77 C9 CD F4 65
                      SOMA = 1059
27184 79 E6 40 20 13 3E 1E 0F
                      SOMA = 573
27192 10 FD 47 CD D8 69 06 03
                      SOMA = 875
27200 4F CD F1 64 71 10 FA C9
                      SOMA = 1205
27208 7D E6 9C 6F CD F4 69 01
                      SOMA = 1177
27216 03 00 5D 54 13 3E 04 E5
                      SOMA = 494
27224 ED B0 D1 21 20 00 19 EB
                      SOMA = 947
27232 D5 01 04 00 3D 20 F1 D1
                      SOMA = 761
27240 C9 DD 21 98 68 CD 56 68
                      SOMA = 1106
27248 21 0D 6B E5 A7 C8 3D 28
                      SOMA = 850
27256 06 3D 28 11 CD 6D 6A 21
                      SOMA = 609
27264 00 E0 11 01 E0 01 FF 17
                      SOMA = 745
27272 36 00 ED B0 C9 21 00 F8
                      SOMA = 949
27280 11 01 F8 3A 3C 76 77 01
                      SOMA = 622
27288 FF 02 ED B0 3A F6 76 E6
                      SOMA = 1322
27296 FE 32 F6 76 C9 DD 21 FE
                      SOMA = 1361
```



```
27304 6A CD 56 68 21 0D 6B E5
                      SOMA = 883
27312 A7 C8 3D 28 06 3D 28 13
                      SOMA = 594
27320 CD CB 6A 21 00 E0 01 00
                      SOMA = 772
27328 18 7E 2F 77 23 08 78 B1
                      SOMA = 659
27336 20 F7 C9 CD A6 67 21 00
                      SOMA = 987
27344 F8 01 00 03 7E 5F 0F 0F
                      SOMA = 503
27352 0F AB E6 07 AB 57 7B 07
                      SOMA = 811
27360 07 07 AA E6 38 AA 77 23
                      SOMA = 794
27368 0B 78 B1 20 E7 C9 0A 08
                      SOMA = 790
27376 49 4E 56 45 52 54 27 4E
                      SOMA = 589
27384 41 44 41 27 54 45 4C 41
                      SOMA = 531
27392 20 20 27 41 54 52 49 27
                      SOMA = 446
27400 54 55 44 4F 27 3A 3B 75
                      SOMA = 590
27408 E6 DF 32 3B 76 47 E6 04
                      SOMA = 985
27416 28 14 78 21 00 40 ED 5B
                      SOMA = 605
27424 2E 77 ED 4B 38 76 E6 08
                      SOMA = 889
27432 CA C0 6C C3 36 6C 21 00
                      SOMA = 892
27440 E0 11 00 40 01 00 1B ED
                      SOMA = 570
27448 B0 C9 3A 3B 76 E6 14 FE
                      SOMA = 1116
27456 10 C0 2A 2E 77 CD E7 64
                      SOMA = 951
27464 E5 3A 37 76 77 3A 3B 76
                      SOMA = 814
27472 E6 08 01 0F 00 28 03 01
                      SOMA = 298
27480 07 00 5D 54 13 ED B0 E1
                      SOMA = 841
27488 3A 3B 76 E6 08 01 10 05
                      SOMA = 501
27496 28 03 01 08 05 C5 E5 11
                      SOMA = 500
27504 20 00 EB 19 EB 05 00 ED
                      SOMA = 770
27512 B0 E1 11 20 00 19 C1 10
                      SOMA = 684
27520 EC 3A 37 76 4F C6 03 A9
                      SOMA = 916
27528 E6 07 A9 D6 08 E6 3F 32
                      SOMA = 971
```

```
27536 37 76 AF C9 3A 3B 76 E6
                        SOMA = 1014
27544 08 01 10 60 28 03 01 08
                        SOMA = 173
27552 30 ED 43 38 76 ED 5B 3E
                        SOMA = 900
27560 77 7B E6 1F 3D 81 FE 20
                        SOMA = 979
27568 38 07 7B E6 E0 C6 20 91
                        SOMA = 1015
27576 5F CD C0 64 80 3D FE C0
                        SOMA = 1227
27584 38 06 3E C0 90 CD D0 64
                        SOMA = 973
27592 ED 53 2E 77 C3 0D 6B CD
                        SOMA = 1005
27600 0F 6C CD F6 6B CD 1C 6C
                        SOMA = 1022
27608 C8 CD F6 6B 3A 20 77 57
                        SOMA = 1054
27616 E6 FD 32 20 77 AA 5F 3A
                        SOMA = 1007
27624 27 77 57 E6 FE 32 27 77
                        SOMA = 937
27632 AA B3 32 28 77 C9 CD 93
                        SOMA = 1111
27640 69 3A 2D 77 A7 C8 5F 3A
                        SOMA = 663
27648 F6 76 E6 04 28 03 2A 2C
                        SOMA = 727
27656 77 10 FE 2D 20 F8 C9 21
                        SOMA = 951
27664 20 77 11 21 77 01 07 00
                        SOMA = 328
27672 36 1F ED B0 21 20 77 01
                        SOMA = 683
27680 FE 7F 1E 00 ED 78 2F A6
                        SOMA = 981
27688 77 B3 5F 23 CB 08 38 F4
                        SOMA = 939
27696 7B 32 28 77 A7 C9 E5 D5
                        SOMA = 1142
27704 C5 C5 D5 E5 41 C5 D5 1A
                        SOMA = 1337
27712 4F 06 04 CB 11 9F 5F CB
                        SOMA = 766
27720 11 9F AB E6 0F AB 77 23
                        SOMA = 917
27728 10 F1 D1 13 C1 10 E6 E1
                        SOMA = 1149
27736 06 03 C5 E5 5D 54 14 06
                        SOMA = 638
27744 00 CB 21 CB 21 ED B0 E1
                        SOMA = 1110
27752 24 C1 10 EE CD F1 64 EB
                        SOMA = 1054
27760 E1 CD F1 64 EB C1 10 C1
                        SOMA = 1408
```



```
27768 C1 E1 CD E7 64 C6 A0 57
                        SOMA = 1399
27776 5D E1 CD E7 64 CB 38 CB
                        SOMA = 1316
27784 38 CB 38 CB 21 C8 21 C5
                        SOMA = 984
27792 D5 E5 41 1A 77 23 77 23
                        SOMA = 841
27800 77 23 77 23 13 10 F4 06
                        SOMA = 593
27808 03 E1 C5 11 20 00 EB 19
                        SOMA = 734
27816 E5 EB 06 00 ED B0 CB 38
                        SOMA = 1142
27824 E1 C1 10 EE 11 20 00 19
                        SOMA = 746
27832 EB C1 09 EB C1 10 D0 C9
                        SOMA = 1290
27840 E5 D5 C5 C5 E5 D5 E5 41
                        SOMA = 1572
27848 D5 C5 1A 4F 06 04 AF CB
                        SOMA = 903
27856 01 17 17 10 FA 5F 0F B3
                        SOMA = 602
27864 77 23 06 04 AF CB 01 17
                        SOMA = 566
27872 17 10 FA 5F 0F B3 77 23
                        SOMA = 732
27880 C1 D1 1C 10 DB E1 5D 54
                        SOMA = 1067
27888 14 CB 21 ED B0 E1 CD F1
                        SOMA = 1340
27896 64 EB E1 24 CD F1 64 C1
                        SOMA = 1335
27904 10 C1 C1 E1 CD E7 64 C6
                        SOMA = 1361
27912 A0 57 5D E1 CD E7 64 78
                        SOMA = 1221
27920 0F 0F 0F E6 1F 47 C5 E5
                        SOMA = 803
27928 D5 41 C5 01 20 00 1A 77
                        SOMA = 653
27936 09 77 2C 77 ED 42 77 2C
                        SOMA = 757
27944 1C C1 10 EE E1 0E 20 09
                        SOMA = 755
27952 EB E1 0E 40 09 C1 10 DE
                        SOMA = 978
27960 C9 2A 2E 77 CD 21 65 7D
                        SOMA = 872
27968 07 07 07 E6 F8 5F 3A 3B
                        SOMA = 711
27976 76 21 20 18 E6 08 20 01
                        SOMA = 478
27984 29 19 ED 4B F8 76 22 F8
                        SOMA = 1026
27992 76 3A 3B 76 E6 20 C8 C3
                        SOMA = 1010
```

```
28000 97 69 C9 ED 4B F8 76 3A
                      SOMA = 1193
28008 3B 76 E6 08 28 14 79 D6
                      SOMA = 810
28016 1F 30 01 AF E6 F8 4F 78
                      SOMA = 932
28024 D6 17 30 01 AF E6 F8 47
                      SOMA = 1010
28032 18 12 79 D6 3F 30 01 AF
                      SOMA = 664
28040 E6 F8 4F 78 D6 2F 30 01
                      SOMA = 987
28048 AF E6 F8 47 CD 2D 66 22
                      SOMA = 1110
28056 2E 77 C3 94 6B 11 FF FF
                      SOMA = 1142
28064 18 21 11 00 FF 18 1C 11
                      SOMA = 398
28072 01 FF 18 17 11 FF 00 18
                      SOMA = 599
28080 12 11 01 00 18 0D 11 FF
                      SOMA = 345
28088 01 18 08 11 00 01 18 03
                      SOMA = 78
28096 11 01 01 21 0D 8B E3 D5
                      SOMA = 512
28104 3A 3B 76 E6 40 28 72 7A
                      SOMA = 805
28112 A7 28 3B F2 F3 6D 21 00
                      SOMA = 893
28120 F8 11 00 77 01 20 00 ED
                      SOMA = 554
28128 B0 11 00 F8 01 E0 02 ED
                      SOMA = 905
28136 B0 21 00 77 01 20 00 ED
                      SOMA = 598
28144 B0 18 1B 21 FF FA 11 1F
                      SOMA = 813
28152 77 01 20 00 ED B8 11 FF
                      SOMA = 845
28160 FA 01 E0 02 ED B8 21 1F
                      SOMA = 952
28168 77 01 20 00 ED B8 D1 7B
                      SOMA = 905
28176 A7 C8 F2 2B 6E 21 01 F8
                      SOMA = 1044
28184 11 00 F8 06 18 C5 1A 01
                      SOMA = 519
28192 1F 00 ED B0 12 23 13 C1
                      SOMA = 709
28200 10 F3 C9 21 FE FA 11 FF
                      SOMA = 1269
28208 FA 06 18 C5 1A 01 1F 00
                      SOMA = 535
28216 ED B8 12 1B 2B C1 10 F3
                      SOMA = 951
28224 C9 7A A7 28 53 F2 71 6E
                      SOMA = 1078
```



```
28232 21 00 E0 11 00 77 01 20
                      SOMA = 426
28240 00 ED B0 21 00 E1 11 00
                      SOMA = 688
28248 E0 E5 01 20 00 ED B0 D1
                      SOMA = 1108
28256 6B 62 CD F1 64 20 F2 21
                      SOMA = 1058
28264 00 77 01 20 00 ED B0 18
                      SOMA = 589
28272 27 21 FF F7 11 1F 77 01
                      SOMA = 742
28280 20 00 ED B8 11 FF F7 21
                      SOMA = 1005
28288 FF F6 E5 01 20 00 ED B8
                      SOMA = 1184
28296 E1 5D 54 CD 0D 65 20 F2
                      SOMA = 995
28304 21 1F 77 01 20 00 ED B8
                      SOMA = 637
28312 D1 7B A7 C8 F2 B9 6E 21
                      SOMA = 1269
28320 1F E0 06 C0 C5 E5 06 20
                      SOMA = 917
28328 CB 16 2B 10 FB E1 CB 1E
                      SOMA = 993
28336 CB 06 CD F1 64 C1 10 EC
                      SOMA = 1200
28344 C9 21 00 E0 06 C0 C5 E5
                      SOMA = 1082
28352 06 20 CB 1E 23 10 FB E1
                      SOMA = 798
28360 CB 16 CB 0E CD F1 64 C1
                      SOMA = 1181
28368 10 EC C9 78 B6 77 7C F6
                      SOMA = 1244
28376 E0 67 78 A3 B6 77 7C E6
                      SOMA = 1265
28384 5F 67 C9 78 A6 C9 3A FF
                      SOMA = 1199
28392 BF FE FF C0 79 0F 0F 0F
                      SOMA = 1058
28400 AC E6 E0 AC 08 7D D9 6F
                      SOMA = 1259
28408 08 67 E3 E5 D9 C9 21 00
                      SOMA = 1018
28416 3C 18 18 11 03 4E 55 4D
                      SOMA = 368
28424 45 52 4F 20 20 50 41 44
                      SOMA = 507
28432 52 41 4F 3F 27 CD 32 70
                      SOMA = 695
28440 DA 0D 6B 22 FC 76 D9 E5
                      SOMA = 1188
28448 CD F4 66 21 FF FF 22 FE
                      SOMA = 1382
28456 BF E5 AF 32 3B 75 CD 0B
                      SOMA = 1038
```

```
28464 70 ED 4B F8 76 CD F4 65
                        SOMA = 1340
28472 48 0D 3E 01 0F 10 FD 47
                        SOMA = 503
28480 7C E6 07 ED 5B FC 76 83
                        SOMA = 1190
28488 5F 7A CE 00 57 1A 5F 16
                        SOMA = 653
28496 00 CD 43 65 38 06 CD E3
                        SOMA = 869
28504 6E 28 F6 CD 32 65 CD D3
                        SOMA = 1168
28512 6E CD F1 64 28 10 CD E3
                        SOMA = 1144
28520 6E 28 04 CB 8A 18 07 CB
                        SOMA = 729
28528 4A CC E6 6E CB CA 2A 2A
                        SOMA = 1107
28536 77 CD 0D 65 28 10 CD E3
                        SOMA = 926
28544 6E 28 04 CB 82 18 07 CB
                        SOMA = 721
28552 42 CC E6 6E CB C2 2A 2A
                        SOMA = 1091
28560 77 CD 32 65 38 05 CD E3
                        SOMA = 958
28568 6E 28 C3 E1 7C E6 E0 07
                        SOMA = 1155
28576 07 07 4F 3C 47 7C E6 1F
                        SOMA = 609
28584 F6 40 67 FE 58 38 8B CD
                        SOMA = 1155
28592 0A 67 E1 D9 C3 0D 6B 13
                        SOMA = 889
28600 06 53 41 4C 56 41 2F 52
                        SOMA = 510
28608 45 43 55 50 45 52 41 27
                        SOMA = 556
28616 54 45 4D 50 4F 52 41 52
                        SOMA = 618
28624 49 41 4D 45 4E 54 45 27
                        SOMA = 554
28632 50 52 4F 56 49 53 4F 52
                        SOMA = 644
28640 49 41 4D 45 4E 54 45 27
                        SOMA = 554
28648 DD 21 B7 6F CD 55 68 2F
                        SOMA = 989
28656 00 0F 0F 0F C6 80 67 C9
                        SOMA = 675
28664 CD E8 6F EB 18 10 CD E8
                        SOMA = 1260
28672 6F 3A F5 76 E6 FE 32 F5
                        SOMA = 1313
28680 76 18 1C 11 00 80 21 00
                        SOMA = 348
28688 E0 01 00 18 ED B0 3A F5
                        SOMA = 956
```



```
28696 76 E6 01 01 00 03 28 0D
                        SOMA = 406
28704 28 FB 18 09 21 00 40 11
                        SOMA = 436
28712 00 E0 01 00 1B ED B0 C3
                        SOMA = 860
28720 0D 6B DD 21 03 6F CD B6
                        SOMA = 875
28728 68 CD 62 68 CD 74 67 87
                        SOMA = 1070
28736 87 87 6F 26 00 3A 28 77
                        SOMA = 636
28744 FE 02 3F D8 A7 11 4D 76
                        SOMA = 914
28752 28 03 11 9D 76 19 C9 CD
                        SOMA = 766
28760 32 70 DA 0D 6B E5 ED 4B
                        SOMA = 1041
28768 F8 76 CD 2D 66 7C E6 F8
                        SOMA = 1320
28776 67 D1 06 08 7E 12 13 24
                        SOMA = 525
28784 10 FA C3 0D 6B DD 21 A0
                        SOMA = 995
28792 70 CD B6 68 CD 62 68 CD
                        SOMA = 1215
28800 2E 67 CD 3D 67 CD A7 70
                        SOMA = 1002
28808 E5 DF D8 41 38 F4 FE 15
                        SOMA = 1307
28816 D2 0D 6B 26 00 87 87 87
                        SOMA = 773
28824 6F ED 5B 7B 5C 19 18 BD
                        SOMA = 892
28832 06 03 55 44 47 3F 27 11
                        SOMA = 352
28840 A8 61 3A 28 77 A7 28 09
                        SOMA = 698
28848 11 D0 61 3D 28 03 11 F8
                        SOMA = 691
28856 61 21 20 77 08 05 0E 08
                        SOMA = 314
28864 CB 1E 38 0B 13 10 F9 23
                        SOMA = 619
28872 0D 06 05 20 F3 AF C9 1A
                        SOMA = 701
28880 C9 3A F6 76 E6 FB 32 F6
                        SOMA = 1400
28888 76 3A 3B 76 E6 20 C8 3A
                        SOMA = 873
28896 3B 76 EE 20 32 3B 76 E6
                        SOMA = 904
28904 20 DD 21 00 A0 28 0F 3A
                        SOMA = 559
28912 F6 76 E6 02 DD 21 3D 76
                        SOMA = 1029
28920 28 04 DD 21 45 76 3A 3B
                        SOMA = 602
```

```
28928 7E ED 4B F8 76 E6 04 20
                        SOMA = 1062
28936 09 CD F4 65 DD E5 D1 C3
                        SOMA = 1413
28944 8F 68 CD 23 69 D8 28 35
                        SOMA = 902
28952 CD F4 65 06 08 C5 DD 6E
                        SOMA = 1076
28960 00 05 04 C5 CB 03 9F 4F
                        SOMA = 651
28968 CB 03 9F A9 E6 0F A9 06
                        SOMA = 954
28976 04 77 24 10 FC 7C D6 04
                        SOMA = 769
28984 67 C1 2C 10 E6 C1 DD 23
                        SOMA = 1035
28992 7D D6 04 6F 7C C6 04 67
                        SOMA = 883
29000 CD F6 64 10 D0 C9 CD F4
                        SOMA = 1425
29008 65 06 08 C5 DD 4E 00 DD
                        SOMA = 832
29016 23 05 04 AF CB 11 17 17
                        SOMA = 486
29024 10 FA 47 0F B0 77 06 04
                        SOMA = 857
29032 AF CB 11 17 17 10 FA 47
                        SOMA = 778
29040 0F B0 4E 2C 77 24 77 2D
                        SOMA = 632
29048 71 CD F1 64 C1 10 D4 C9
                        SOMA = 1281
29056 3A 3B 76 07 07 D8 07 DC
                        SOMA = 892
29064 93 69 3A F6 76 F6 04 32
                        SOMA = 974
29072 F6 76 ED 4B F8 76 78 E6
                        SOMA = 1392
29080 F8 47 79 E6 F8 4F ED 43
                        SOMA = 1301
29088 F8 76 CD 20 72 CD 2E 67
                        SOMA = 1071
29096 3E 10 32 2C 77 CD CF 6B
                        SOMA = 810
29104 CD A7 70 FE FF CA D1 70
                        SOMA = 1516
29112 A7 28 ED CD D1 71 3A 2C
                        SOMA = 1073
29120 77 D6 18 38 08 FE 20 30
                        SOMA = 755
29128 E1 3E 20 18 DD 3E 50 18
                        SOMA = 730
29136 D9 F5 CD D9 70 F1 FE 20
                        SOMA = 1523
29144 38 56 6F 26 00 3A F6 76
                        SOMA = 713
29152 E6 02 ED 5B 3B 5C 28 12
                        SOMA = 764
```



```
29160 7D FE 20 28 0D FE 41 D8
                        SOMA = 999
29168 FE 56 D0 D6 41 6F ED 5B
                        SOMA = 1266
29176 7B 5C 29 29 29 19 E5 ED
                        SOMA = 829
29184 4B F8 76 CD 2D 66 D1 D5
                        SOMA = 1215
29192 CD 6F 68 DD E1 ED 4B F8
                        SOMA = 1458
29200 76 79 C6 08 4F 78 47 C5
                        SOMA = 912
29208 CD FE 70 C1 ED 43 F8 76
                        SOMA = 1434
29216 CD 2D 66 11 00 A0 01 08
                        SOMA = 538
29224 00 7E 12 24 13 10 FA C9
                        SOMA = 666
29232 DD 21 00 A0 FE 09 20 0B
                        SOMA = 720
29240 3A F6 76 EE 02 32 F6 76
                        SOMA = 1076
29248 C3 2E 67 FE 05 D8 ED 4B
                        SOMA = 1131
29256 F8 76 20 05 79 D6 08 18
                        SOMA = 770
29264 C3 FE 08 28 BC FE 07 78
                        SOMA = 1066
29272 20 08 D6 08 30 B8 D6 40
                        SOMA = 772
29280 18 B4 C6 08 FE C0 38 AE
                        SOMA = 1086
29288 D6 C0 18 AA C5 3A 08 77
                        SOMA = 982
29296 90 ED 43 07 77 32 07 77
                        SOMA = 750
29304 CD 84 72 C1 79 48 47 91
                        SOMA = 1053
29312 C8 32 07 77 11 00 00 C5
                        SOMA = 590
29320 CD 95 72 C1 78 A7 C8 11
                        SOMA = 1165
29328 00 FF ED 44 47 3A 29 77
                        SOMA = 849
29336 A7 20 2A C5 D5 CD A9 72
                        SOMA = 1139
29344 D1 C1 79 A7 C8 ED 44 4F
                        SOMA = 1274
29352 1D 2A FA 76 7C 80 47 9F
                        SOMA = 921
29360 AA C0 78 FE C0 D0 7D 81
                        SOMA = 1390
29368 4F 9F AB C0 C5 CD 08 69
                        SOMA = 1116
29376 C1 CD E6 69 C9 3A 07 77
                        SOMA = 1118
29384 A7 C8 79 A7 28 D8 C5 D5
                        SOMA = 1324
```

```
29392 CD A9 72 D1 C1 C5 D5 1D
                      SOMA = 1329
29400 79 ED 44 4F CD A9 72 D1
                      SOMA = 1202
29408 C1 0D 18 E1 3E FF 18 01
                      SOMA = 797
29416 AF 32 29 77 CD 0B 70 CD
                      SOMA = 918
29424 F4 66 ED 4B F8 76 2A FA
                      SOMA = 1316
29432 76 78 94 30 02 ED 44 F5
                      SOMA = 986
29440 79 95 30 02 ED 44 CD 60
                      SOMA = 926
29448 73 F1 E5 CD 60 73 D1 19
                      SOMA = 1235
29456 CD 74 73 7C A7 20 46 7A
                      SOMA = 951
29464 B3 28 01 2C DD 21 00 77
                      SOMA = 637
29472 7D DD 77 00 A7 20 08 01
                      SOMA = 673
29480 00 00 CD A9 72 18 2E 11
                      SOMA = 575
29488 B5 00 CD 63 73 7C DD 77
                      SOMA = 1064
29496 01 DD 7E 01 CD 60 73 E5
                      SOMA = 994
29504 DD 7E 00 CD 60 73 C1 A7
                      SOMA = 1123
29512 ED 42 CD 74 73 45 DD 4E
                      SOMA = 1107
29520 01 CD 6C 72 DD 7E 01 DD
                      SOMA = 997
29528 35 01 A7 20 DC C3 0A 67
                      SOMA = 781
29536 5F 16 00 21 00 00 06 08
                      SOMA = 164
29544 CB 3F 30 01 19 CB 23 CB
                      SOMA = 781
29552 12 10 F5 C9 11 01 00 A7
                      SOMA = 665
29560 ED 52 38 04 13 13 18 F7
                      SOMA = 688
29568 19 CB 3A CB 1B EB C9 3A
                      SOMA = 1010
29576 3A 76 18 02 3E FF 6F D9
                      SOMA = 847
29584 E5 CD 0B 70 CD F4 66 2A
                      SOMA = 1150
29592 FA 76 ED 4B F8 76 11 01
                      SOMA = 1064
29600 01 78 94 28 05 30 04 ED
                      SOMA = 603
29608 44 15 15 47 79 95 28 05
                      SOMA = 496
29616 30 04 ED 44 1D 1D 4F B0
                      SOMA = 670
```



```
29624 28 42 79 B8 D5 38 06 16
                      SOMA = 708
29632 00 68 41 18 03 1E 00 69
                      SOMA = 331
29640 60 D9 ED 4B FA 76 D9 78
                      SOMA = 1330
29648 CB 3F B5 38 03 BC 38 07
                      SOMA = 709
29656 94 4F D9 D1 D5 18 04 4F
                      SOMA = 973
29664 D5 D9 D1 7A B0 47 79 83
                      SOMA = 1212
29672 4F CB 0D 30 0A E5 C5 CD
                      SOMA = 984
29680 08 69 C1 CD E6 59 E1 D9
                      SOMA = 1288
29688 79 10 D7 D1 CD 0A 67 E1
                      SOMA = 1104
29696 D9 C9 00 40 2A 02 74 3A
                      SOMA = 700
29704 3B 76 E6 04 28 03 2A 97
                      SOMA = 647
29712 65 CD E7 64 7C F6 E0 67
                      SOMA = 1334
29720 7E E6 20 3E 07 20 02 3E
                      SOMA = 553
29728 38 21 53 75 11 00 9B 01
                      SOMA = 462
29736 1C 00 ED B0 EB 11 1D 9B
                      SOMA = 877
29744 77 01 1B 00 ED B0 6F 0F
                      SOMA = 686
29752 0F 0F AD E6 07 AD 6F 67
                      SOMA = 827
29760 22 35 9B 22 28 9B EE 06
                      SOMA = 715
29768 32 30 9B 21 00 9B ED 4B
                      SOMA = 753
29776 F8 76 C5 79 0E 03 CD 7F
                      SOMA = 1033
29784 75 23 C1 3E AF 90 0E 03
                      SOMA = 743
29792 30 06 ED 44 36 2D 23 0D
                      SOMA = 506
29800 CD 7F 75 3A F6 76 E6 04
                      SOMA = 1105
29808 20 05 3A 3B 76 E6 03 07
                      SOMA = 512
29816 07 EB 6F 26 00 01 3F 75
                      SOMA = 572
29824 09 01 04 00 ED B0 11 0E
                      SOMA = 458
29832 9B 3A F6 76 E6 06 FE 04
                      SOMA = 1071
29840 30 08 3A 3B 76 E6 40 07
                      SOMA = 592
29848 07 07 07 6F 26 00 01 6F
                      SOMA = 292
```

```
29856 75 09 01 04 00 ED B0 3A
                      SOMA = 602
29864 3B 76 07 38 0B 2A 28 9B
                      SOMA = 488
29872 7C EE 05 6F 67 22 28 9B
                      SOMA = 810
29880 3A 3C 76 47 E6 07 C6 30
                      SOMA = 790
29888 32 17 9B E6 07 FE 04 17
                      SOMA = 746
29896 0F 07 17 0F 07 17 32 33
                      SOMA = 191
29904 9B 78 E6 38 FE 20 30 02
                      SOMA = 897
29912 F6 07 32 31 9B 0F 0F 0F
                      SOMA = 552
29920 E6 07 C6 30 32 15 9B C8
                      SOMA = 912
29928 00 30 08 3A 36 9B EE 87
                      SOMA = 696
29936 32 36 9B CB 00 30 08 3A
                      SOMA = 576
29944 35 9B EE 47 32 35 9B 3A
                      SOMA = 833
29952 3B 76 2E 32 E6 0C FE 08
                      SOMA = 777
29960 38 02 2E 34 E6 04 20 02
                      SOMA = 424
29968 2E 31 7D 32 14 9B 3A F6
                      SOMA = 749
29976 76 E6 10 C0 2A 02 74 DD
                      SOMA = 937
29984 21 00 9B 01 04 07 E5 C5
                      SOMA = 626
29992 DD 7E 1C 08 DD 7E 00 CD
                      SOMA = 935
30000 72 68 DD 23 10 F2 C1 E1
                      SOMA = 1150
30008 CD 59 65 0D 20 E8 C9 53
                      SOMA = 956
30016 45 54 20 58 4F 52 20 52
                      SOMA = 548
30024 45 53 20 53 41 4C 54 54
                      SOMA = 576
30032 45 58 54 20 20 20 2C 20
                      SOMA = 413
30040 20 20 20 20 20 20 20 53
                      SOMA = 307
30048 41 20 20 20 20 20 4D 20
                      SOMA = 334
30056 20 50 20 49 42 46 20 54
                      SOMA = 469
30064 45 4C 41 41 54 52 49 20
                      SOMA = 546
30072 20 20 20 47 52 41 46 E5
                      SOMA = 613
30080 06 00 21 95 75 09 5E E1
                      SOMA = 633
```



```
30088 06 2F 93 04 30 FC 83 70
                      SOMA = 747
30096 23 0D 20 EB C9 00 01 0A
                      SOMA = 527
30104 64 2A 02 74 7C FE 40 20
                      SOMA = 734
30112 05 21 80 50 18 03 21 00
                      SOMA = 306
30120 40 22 02 74 CD 0D 6B C3
                      SOMA = 736
30128 16 75 3A F6 76 EE 10 32
                      SOMA = 865
30136 F6 76 18 F0 0D 07 54 49
                      SOMA = 805
30144 50 4F 20 4C 49 4E 2F 27
                      SOMA = 504
30152 42 49 54 20 52 45 53 45
                      SOMA = 558
30160 54 27 42 49 54 20 53 45
                      SOMA = 530
30168 54 27 27 20 2A 2A 2A 2A
                      SOMA = 362
30176 2A 2A 2A 2A 27 27 DD 21
                      SOMA = 500
30184 BC 75 CD 31 68 21 C2 40
                      SOMA = 954
30192 E5 CD 65 68 3E 87 32 C3
                      SOMA = 1081
30200 58 E1 2C 06 08 C5 E5 CD
                      SOMA = 1002
30208 2E 67 CD 74 67 FE 02 30
                      SOMA = 877
30216 F6 6F 07 07 F6 43 08 3E
                      SOMA = 754
30224 30 85 E1 CD 72 68 E5 CD
                      SOMA = 1263
30232 E7 64 36 87 E1 C1 10 DD
                      SOMA = 1175
30240 05 08 21 C3 58 7E 23 E6
                      SOMA = 721
30248 04 0F 0F 0F CB 11 10 F5
                      SOMA = 530
30256 79 32 3A 76 C3 0D 6B 03
                      SOMA = 665
30264 08 28 AA 00 07 FF 81 81
                      SOMA = 738
30272 99 99 81 81 FF 81 7E 66
                      SOMA = 1176
30280 5A 5A 66 7E 81 55 AA 55
                      SOMA = 877
30288 AA 55 AA 55 AA AA 00 AA
                      SOMA = 1020
30296 00 AA 00 AA 00 FF 00 FF
                      SOMA = 850
30304 00 FF 00 FF 00 88 11 22
                      SOMA = 697
30312 44 88 10 20 40 21 90 48
                      SOMA = 565
```

```
30320 24 12 09 84 42 AA AA 44
                    SOMA = 569
30328 44 AA AA 11 11 EE 22 33
                    SOMA = 765
30336 99 EE 22 33 99 80 80 80
                    SOMA = 1013
30344 FF 08 08 08 FF 88 44 22
                    SOMA = 772
30352 22 22 44 88 88 22 50 88
                    SOMA = 658
30360 05 22 50 88 05 08 04 7C
                    SOMA = 396
30368 A4 25 3E 20 10 43 A5 5A
                    SOMA = 633
30376 34 2C 5A A5 C2 55 55 55
                    SOMA = 800
30384 55 55 55 55 55 CC CC 33
                    SOMA = 884
30392 33 CC CC 33 33 38 70 E0
                    SOMA = 953
30400 C1 83 07 0E 1C 22 FA 8A
                    SOMA = 795
30408 AA AA AA A8 AF E5 07 38
                    SOMA = 1145
30416 AB BA 83 E0 A7 86 55 22
                    SOMA = 1134
30424 11 44 22 55 88 08 1C 3E
                    SOMA = 438
30432 7F FF 7F 3E 1C A0 BE 82
                    SOMA = 1079
30440 FA 0A EB 28 AF 00 01 08
                    SOMA = 719
30448 10 18 20 30 40 50 00 00
                    SOMA = 264
```


LISTAGEM "DISASSEMBLADA" DOS CÓDIGOS HEXADECIMAIS DO PROGRAMA

```
>Disassemble £25000
61A8 2000       jr nz,61AA
61AA 4D         ld c,l
61AB 4E         ld c,(hl)
61AC 42         ld b,d
61AD 0D         dec c
61AE 4C         ld c,h
61AF 4B         ld c,e
61B0 4A         ld c,d
61B1 48         ld c,b
61B2 50         ld d,b
61B3 4F         ld c,a
61B4 49         ld c,c
61B5 55         ld d,l
61B6 59         ld e,c
61B7 3039       jr nc,61F2
61B9 3837       jr c,61F2
61BB 3631       ld (hl),31
61BD 323334     ld (3433),a
61C0 35         dec (hl)
61C1 51         ld d,c
61C2 57         ld d,a
61C3 45         ld b,l
61C4 52         ld d,d
61C5 54         ld d,h
61C6 41         ld b,c
61C7 53         ld d,e
61C8 44         ld b,h
61C9 46         ld b,(hl)
61CA 47         ld b,a
61CB 015A58     ld bc,585A
61CE 43         ld b,e
61CF 56         ld d,(hl)
61D0 FF         rst 38
61D1 00         nop
61D2 6D         ld l,l
61D3 6E         ld l,(hl)
61D4 62         ld h,d
61D5 0D         dec c
61D6 6C         ld l,h
61D7 6B         ld l,e
61D8 6A         ld l,d
61D9 68         ld l,b
61DA 70         ld (hl),b
61DB 6F         ld l,a
61DC 69         ld l,c
61DD 75         ld (hl),l
61DE 79         ld a,c
61DF 00         nop
61E0 09         add hl,bc
61E1 08         ex af,af'
61E2 07         rlca
61E3 0600       ld b,00
```

```
61E5 00         nop
61E6 00         nop
61E7 00         nop
61E8 05         dec b
61E9 71         ld (hl),c
61EA 77         ld (hl),a
61EB 6F         ld h,l
61EC 72         ld (hl),d
61ED 74         ld (hl),h
61EE 61         ld h,c
61EF 73         ld (hl),e
61F0 64         ld h,h
61F1 66         ld h,(hl)
61F2 67         ld h,a
61F3 00         nop
61F4 7A         ld a,d
61F5 78         ld a,b
61F6 63         ld h,e
61F7 76         halt
61F8 20A000     jr nz,61FA
61FA 2E2C       ld l,2C
61FC 2A0D3D     ld hl,(3D0D)
61FF 2B         dec hl
6200 2D         dec l
6201 5E         ld e,(hl)
6202 22B37F     ld (7F3B),hl
6205 5D         ld e,l
6206 5B         ld e,e
6207 5F         ld e,a
6208 29         add hl,hl
6209 282B       jr z,6236
620B 2621       ld h,21
620D 40         ld b,b
620E 23         inc hl
620F 24         inc h
6210 25         dec h
6211 20D0       jr nz,61E3
6213 20D3       jr nz,61E8
6215 3E7E       ld a,7E
6217 7C         ld a,h
6218 5C         ld e,h
6219 7B         ld a,e
621A 7D         ld a,l
621B 00         nop
621C 3A603F     ld a,(3F60)
621F 2F         cpl
6220 00         nop
6221 00         nop
6222 00         nop
6223 00         nop
6224 DE66       sbc a,66
6226 59         ld e,c
6227 75         ld (hl),l
6228 54         ld d,h
6229 67         ld h,a
622A 68         ld l,b
622B 65         ld h,l
622C D366       out (66),a
```



```
622E 00         nop
622F 00         nop
6230 00         nop
6231 00         nop
6232 00         nop
6233 00         nop
6234 BA         cp d
6235 66         ld h,(hl)
6236 C2E6BE     jp nz,BEE6
6239 66         ld h,(hl)
623A C666       add a,66
623C 00         nop
623D 00         nop
623E 7E         ld a,(hl)
623F 66         ld h,(hl)
6240 7E         ld a,(hl)
6241 66         ld h,(hl)
6242 7E         ld a,(hl)
6243 66         ld h,(hl)
6244 7E         ld a,(hl)
6245 66         ld h,(hl)
6246 7E         ld a,(hl)
6247 66         ld h,(hl)
6248 7E         ld a,(hl)
6249 66         ld h,(hl)
624A 7E         ld a,(hl)
624B 66         ld h,(hl)
624C 7E         ld a,(hl)
624D 66         ld h,(hl)
624E 7E         ld a,(hl)
624F 66         ld h,(hl)
6250 7E         ld a,(hl)
6251 66         ld h,(hl)
6252 85         add a,l
6253 65         ld h,l
6254 92         sub d
6255 65         ld h,l
6256 97         sub a
6257 65         ld h,l
6258 00         nop
6259 00         nop
625A 80         add a,b
625B 71         ld (hl),c
625C 9C         sbc a,h
625D 65         ld h,l
625E 63         ld h,e
625F 6D         ld l,l
6260 A4         and c
6261 65         ld h,l
6262 00         nop
6263 00         nop
6264 BC         cp h
6265 67         ld h,a
6266 00         nop
6267 00         nop
6268 A6         and (hl)
6269 65         ld h,l
626A AB         xor e
```

```
625B 65          ld h,l
625C B0          or b
625D 65          ld h,l
625E 00          nop
625F 00          nop
6270 00          nop
6271 00          nop
6272 00          nop
6273 00          nop
6274 E2566B2     jp po,B266
6277 75          ld (hl),l
6278 54          ld d,c
6279 67          ld h,a
627A 6F          ld l,a
627B 65          ld h,l
627C D366        out (66),a
627E E675        and 75
6280 8C          adc a,h
6281 73          ld (hl),e
6282 E8          ret pe
6283 72          ld (hl),d
6284 57          ld d,a
6285 70          ld (hl),b
6286 75          ld (hl),l
6287 70          ld (hl),b
6288 F8          ret m
6289 6F          ld l,a
628A FE6F        cp 6F
628C 00          nop
628D 00          nop
628E 7A          ld a,d
628F 66          ld h,(hl)
6290 7A          ld a,d
6291 66          ld h,(hl)
6292 7A          ld a,d
6293 66          ld h,(hl)
6294 7A          ld a,d
6295 66          ld h,(hl)
6296 7A          ld a,d
6297 66          ld h,(hl)
6298 7A          ld a,d
6299 66          ld h,(hl)
629A 7A          ld a,d
629B 66          ld h,(hl)
629C 7A          ld a,d
629D 66          ld h,(hl)
629E 7A          ld a,d
629F 66          ld h,(hl)
62A0 7A          ld a,d
62A1 66          ld h,(hl)
62A2 5F          ld e,a
62A3 64          ld h,h
62A4 61          ld h,c
62A5 64          ld h,h
62A6 64          ld h,(hl)
62A7 64          ld h,h
62A8 00          nop
62A9 00          nop
```



```
62AA 00          nop
62AB 00          nop
62AC 6B          ld l,e
62AD 64          ld h,h
62AE 39          add hl,sp
62AF 6D          ld l,l
62B0 70          ld (hl),b
62B1 64          ld h,h
62B2 FE6E        cp 6E
62B4 C0          ret nz
62B5 67          ld h,a
62B6 00          nop
62B7 00          nop
62B8 75          ld (hl),l
62B9 54          ld d,h
62BA 7A          ld a,d
62BB 64          ld h,h
62BC 7A          ld a,a
62BD 64          ld h,h
62BE 6E          ld l,c
62BF 6A          ld l,d
62C0 00          nop
62C1 00          nop
62C2 00          nop
62C3 00          nop
62C4 E666        and 66
62C6 B2          or d
62C7 75          ld (hl),l
62C8 54          ld d,c
62C9 67          ld h,a
62CA 70          ld (hl),b
62CB 65          ld h,l
62CC D366        out (66),a
62CE 00          nop
62CF 00          nop
62D0 87          add a,a
62D1 73          ld (hl),e
62D2 F472BA      call po,BA72
62D5 66          ld h,(hl)
62D6 C2666E      jp nz,6E66
62D9 66          ld h,(hl)
62DA C666        add a,66
62DC 00          nop
62DD 00          nop
62DE AB          xor e
62DF 66          ld h,(hl)
62E0 AB          xor e
62E1 66          ld h,(hl)
62E2 AB          xor e
62E3 66          ld h,(hl)
62E4 AB          xor e
62E5 66          ld h,(hl)
62E6 AB          xor e
62E7 66          ld h,(hl)
62E8 AB          xor e
62E9 66          ld h,(hl)
62EA AB          xor e
62EB 66          ld h,(hl)
```

```
62EC  AB        xor e
62ED  66        ld h,(hl)
62EE  AB        xor e
62EF  66        ld h,(hl)
62F0  AB        xor e
62F1  66        ld h,(hl)
62F2  9D        sbc a,l
62F3  6D        ld l,l
62F4  A2        and d
62F5  6D        ld l,l
62F6  A7        and a
62F7  6D        ld l,l
62F8  00        nop
62F9  00        nop
62FA  00        nop
62FB  00        nop
62FC  AC        xor h
62FD  6D        ld l,l
62FE  DA66B1    jp c,B166
6301  6D        ld l,l
6302  15        dec d
6303  6F        ld l,a
6304  C0        ret nz
6305  67        ld h,a
6306  00        nop
6307  00        nop
6308  B6        or (hl)
6309  6D        ld l,l
630A  BB        cp e
630B  6D        ld l,l
630C  C0        ret nz
630D  6D        ld l,l
630E  A5        and l
630F  6A        ld l,d
6310  1100E0    ld de,E000
6313  ED532E77  ld (772E),de
6317  CD0B70    call 700B
631A  AF        xor a
631B  323B75    ld (753B),a
631E  32F675    ld (75F6),a
6321  3E18      ld a,18
6323  322D77    ld (772D),a
6326  C9        ret
6327  ED73FE75  ld (75FE),sp
632B  3100E0    ld sp,E000
632E  CD5A63    call 635A
6331  ED7BFE75  ld sp,(75FE)
6335  C9        ret
6336  3AF775    ld a,(75F7)
6339  D3FE      out (FE),a
633B  CD0D6B    call 6B0D
633E  CD2E67    call 672E
6341  CD3D67    call 673D
6344  CD2664    call 6426
6347  CD0474    call 7404
634A  3A2577    ld a,(7725)
634D  E608      and 08
634F  2009      jr nz,635A
```



```
6351  3A2D77    ld a,(772D)
6354  A7        and a
6355  CC2E67    call z,672E
6358  18E7      jr 6341
635A  CDA667    call 67A6
635D  DD218E63  ld ix,638E
6361  CD5668    call 6856
6364  A7        and a
6365  28CF      jr z,6336
6367  FE05      cp 05
6369  01FF00    ld bc,00FF
636C  C8        ret z
636D  F5        push af
636E  CD0D6B    call 6B0D
6371  DD219463  ld ix,6394
6375  CD5668    call 6856
6378  C1        pop bc
6379  4F        ld c,a
637A  C5        push bc
637B  CD0D6B    call 6B0D
637E  DD218C63  ld ix,638C
6382  CDB568    call 68B6
6385  CD6268    call 6862
6388  C1        pop bc
6389  C9        ret
638A  4F        ld c,a
638B  C9        ret
638C  07        rlca
638D  03        inc bc
638E  4E        ld c,(hl)
638F  4F        ld c,a
6390  4D        ld c,l
6391  45        ld b,l
6392  3F        ccf
6393  27        daa
6394  110753    ld de,5307
6397  41        ld b,c
6398  56        ld d,(hl)
6399  45        ld b,l
639A  2F        cpl
639B  4C        ld c,h
639C  4F        ld c,a
639D  41        ld b,c
639E  44        ld b,h
639F  27        daa
63A0  41        ld b,c
63A1  52        ld d,d
63A2  45        ld b,l
63A3  41        ld b,c
63A4  2055      jr nz,63FB
63A6  44        ld b,h
63A7  47        ld b,a
63A8  27        daa
63A9  50        ld d,b
63AA  49        ld c,c
63AB  4E        ld c,(hl)
63AC  54        ld d,h
63AD  41        ld b,c
```

```
63AE 52          ld  d,d
63AF 2050        jr  nz,6401
63B1 41          ld  b,c
63B2 44          ld  b,h
63B3 52          ld  d,d
63B4 41          ld  b,c
63B5 4F          ld  c,e
63B6 27          daa
63B7 54          ld  d,h
63B8 45          ld  b,l
63B9 4C          ld  c,h
63BA 41          ld  b,c
63BB 2020        jr  nz,63DD
63BD 27          daa
63BE 180A        jr  63CA
63C0 4F          ld  c,a
63C1 50          ld  d,b
63C2 43          ld  b,e
63C3 4F          ld  c,a
63C4 45          ld  b,l
63C5 53          ld  d,e
63C6 2027        jr  nz,63EF
63C8 45          ld  b,l
63C9 44          ld  b,h
63CA 49          ld  c,c
63CB 54          ld  d,h
63CC 41          ld  b,c
63CD 52          ld  d,d
63CE 2054        jr  nz,6424
63D0 45          ld  b,l
63D1 4C          ld  c,h
63D2 41          ld  b,c
63D3 2027        jr  nz,63FC
63D5 53          ld  d,e
63D6 41          ld  b,c
63D7 56          ld  d,(hl)
63D8 45          ld  b,l
63D9 2050        jr  nz,642B
63DB 2F          cpl
63DC 2045        jr  nz,6424
63DE 49          ld  c,c
63DF 54          ld  d,h
63E0 41          ld  b,c
63E1 27          daa
63E2 4C          ld  c,h
63E3 4F          ld  c,a
63E4 41          ld  b,c
63E5 44          ld  b,h
63E6 2020        jr  nz,6408
63E8 44          ld  b,h
63E9 41          ld  b,c
63EA 2020        jr  nz,640C
63EC 46          ld  b,(hl)
63ED 49          ld  c,c
63EE 54          ld  d,h
63EF 41          ld  b,c
63F0 27          daa
63F1 53          ld  d,e
```



```
63F2 41          ld  b,c
63F3 55          ld  d,(hl)
63F4 45          ld  b,l
63F5 2050        jr  nz,6447
63F7 2F          cpl
63F8 2047        jr  nz,6447
63FA 49          ld  c,c
63FB 44          ld  b,e
63FC 52          ld  d,d
63FD 4F          ld  c,a
63FE 44          ld  b,h
63FF 52          ld  d,d
6400 49          ld  c,c
6401 56          ld  d,(hl)
6402 45          ld  b,l
6403 27          daa
6404 4C          ld  c,h
6405 4F          ld  c,a
6406 41          ld  b,c
6407 44          ld  b,h
6408 2020        jr  nz,642A
640A 44          ld  b,h
640B 45          ld  b,l
640C 2020        jr  nz,642E
640E 4D          ld  c,l
640F 49          ld  c,c
6410 43          ld  b,e
6411 52          ld  d,d
6412 4F          ld  c,a
6413 44          ld  b,h
6414 52          ld  d,d
6415 49          ld  c,c
6416 56          ld  d,(hl)
6417 45          ld  b,l
6418 27          daa
6419 53          ld  d,e
641A 41          ld  b,c
641B 49          ld  c,c
641C 2044        jr  nz,6462
641E 4F          ld  c,a
641F 2050        jr  nz,6471
6421 52          ld  d,d
6422 4F          ld  c,a
6423 47          ld  b,a
6424 2E27        ld  l,27
6426 2112062     ld  hl,6220
6429 3A2877      ld  a,(7728)
642C A7          and a
642D 2809        jr  z,6438
642F 217062      ld  hl,6270
6432 3D          dec a
6433 2803        jr  z,6438
6435 21C062      ld  hl,62C0
6438 DD212077    ld  ix,7720
643C 0E08        ld  c,08
643E DD7E00      ld  a,(ix+00)
6441 0605        ld  b,05
6443 1F          rra
```

```
6444 380A       jr c,6450
6446 23         inc hl
6447 23         inc hl
6448 10F9       djnz 6443
644A DD23       inc ix
644C 0D         dec c
644D 20EF       jr nz,643E
644F C9         ret
6450 5E         ld e,(hl)
6451 23         inc hl
6452 56         ld d,(hl)
6453 7A         ld a,d
6454 B3         or e
6455 C8         ret z
6456 212E67     ld hl,672E
6459 E5         push hl
645A EB         ex de,hl
645B E9         jp (hl)
645C 11FFFF     ld de,FFFF
645F 1821       jr 6482
6461 1100FF     ld de,FF00
6464 1816       jr 6482
6466 1101FF     ld de,FF01
6469 1817       jr 6482
646B 11FF00     ld de,00FF
646E 1812       jr 6482
6470 110100     ld de,0001
6473 180D       jr 6482
6475 11FF01     ld de,01FF
6478 1808       jr 6482
647A 110001     ld de,0100
647D 1803       jr 6482
647F 110101     ld de,0101
6482 219488     ld hl,8894
6485 E3         ex (sp),hl
6486 ED4B2E77   ld bc,(772E)
648A 79         ld a,c
648B E61F       and 1F
648D 83         add a,e
648E FE20       cp 20
6490 D0         ret nc
6491 7B         ld a,e
6492 81         add a,c
6493 4F         ld c,a
6494 7A         ld a,d
6495 A7         and a
6496 2811       jr z,64A9
6498 FE01       cp 01
649A 2812       jr z,64AE
649C 79         ld a,c
649D D620       sub 20
649F 4F         ld c,a
64A0 3007       jr nc,64A9
64A2 78         ld a,b
64A3 D608       sub 08
64A5 47         ld b,a
64A6 FEE0       cp E0
64A8 D8         ret c
```



```
64A9 ED432E77   ld (772E),bc
64AD C9         ret
64AE 08         ex af,af'
64AF 79         ld a,c
64B0 C620       add a,20
64B2 4F         ld c,a
64B3 30F4       jr nc,64A9
64B5 78         ld a,b
64B6 C608       add a,08
64B8 47         ld b,a
64B9 E618       and 18
64BB FE18       cp 18
64BD 20EA       jr nz,64A9
64BF C9         ret
64C0 7B         ld a,e
64C1 07         rlca
64C2 07         rlca
64C3 07         rlca
64C4 AA         xor d
64C5 E607       and 07
64C7 AA         xor d
64C8 07         rlca
64C9 07         rlca
64CA 07         rlca
64CB AA         xor d
64CC E6F8       and F8
64CE AA         xor d
64CF C9         ret
64D0 57         ld d,a
64D1 07         rlca
64D2 07         rlca
64D3 AB         xor e
64D4 E6E0       and E0
64D6 AB         xor e
64D7 5F         ld e,a
64D8 7A         ld a,d
64D9 37         scf
64DA 1F         rra
64DB 37         scf
64DC 1F         rra
64DD 37         scf
64DE 1F         rra
64DF AA         xor d
64E0 E6F8       and F8
64E2 AA         xor d
64E3 E6F8       and F8
64E5 57         ld d,a
64E6 C9         ret
64E7 7C         ld a,h
64E8 0F         rrca
64E9 0F         rrca
64EA 0F         rrca
64EB E603       and 03
64ED C658       add a,58
64EF 67         ld h,a
64F0 C9         ret
64F1 222A77     ld (772A),hl
64F4 24         inc h
```

```
64F5 7C        ld a,h
64F6 E607      and 07
64F8 C0        ret nz
64F9 7D        ld a,l
64FA C620      add a,20
64FC 6F        ld l,a
64FD 3805      jr c,6504
64FF 7C        ld a,h
6500 D608      sub 08
6502 67        ld h,a
6503 C9        ret
6504 7C        ld a,h
6505 2F        cpl
6506 E618      and 18
6508 C0        ret nz
6509 2A2A77    ld hl,(772A)
650C C9        ret
650D 222A77    ld (772A),hl
6510 25        dec h
6511 7C        ld a,h
6512 2F        cpl
6513 E607      and 07
6515 C0        ret nz
6516 7D        ld a,l
6517 D620      sub 20
6519 6F        ld l,a
651A 38E8      jr c,6504
651C 7C        ld a,h
651D C608      add a,08
651F 67        ld h,a
6520 C9        ret
6521 7D        ld a,l
6522 07        rlca
6523 07        rlca
6524 07        rlca
6525 AC        xor h
6526 E607      and 07
6528 AC        xor h
6529 07        rlca
652A 07        rlca
652B 07        rlca
652C AC        xor h
652D E6F8      and F8
652F AC        xor h
6530 57        ld d,a
6531 C9        ret
6532 0C        inc c
6533 CB08      rrc b
6535 D0        ret nc
6536 7D        ld a,l
6537 3C        inc a
6538 E61F      and 1F
653A 2802      jr z,653E
653C 2C        inc l
653D C9        ret
653E 0D        dec c
653F CB00      rlc b
6541 37        scf
```



```
6542 C9        ret
6543 0D        dec c
6544 CB00      rlc b
6546 D0        ret nc
6547 7D        ld a,l
6548 3D        dec a
6549 2F        cpl
654A E61F      and 1F
654C 2802      jr z,6550
654E 2D        dec l
654F C9        ret
6550 0C        inc c
6551 CB08      rrc b
6553 37        scf
6554 C9        ret
6555 7D        ld a,l
6556 E6E0      and E0
6558 6F        ld l,a
6559 7D        ld a,l
655A C620      add a,20
655C 6F        ld l,a
655D D0        ret nc
655E 7C        ld a,h
655F C608      add a,08
6561 67        ld h,a
6562 2F        cpl
6563 E618      and 18
6565 C0        ret nz
6566 2640      ld h,40
6568 C9        ret
6569 2AF876    ld hl,(76F8)
656C 22FA76    ld (76FA),hl
656F C9        ret
6570 CDF466    call 66F4
6573 ED4BFA76  ld bc,(76FA)
6577 CD9769    call 6997
657A CD9389    call 8993
657D CD1867    call 6718
6580 20F1      jr nz,6573
6582 CD0A67    call 670A
6585 C30D6B    jp 6B0D
6588 110000    ld de,0000
658B 1826      jr 65B3
658D 11FFFF    ld de,FFFF
6590 1821      jr 65B3
6592 1100FF    ld de,FF00
6595 181C      jr 65B3
6597 1101FF    ld de,FF01
659A 1817      jr 65B3
659C 11FF00    ld de,00FF
659F 1812      jr 65B3
65A1 110100    ld de,0001
65A4 180D      jr 65B3
65A6 11FF01    ld de,01FF
65A9 1808      jr 65B3
65AB 110001    ld de,0100
65AE 1803      jr 65B3
65B0 110101    ld de,0101
```

```
65B3 C1          pop bc
65B4 ED4BF875    ld bc,(75F8)
65B8 3A3B75      ld a,(753B)
65BB E640        and 40
65BD 280E        jr z,65CD
65BF 7B          ld a,e
65C0 E61F        and 1F
65C2 07          rlca
65C3 07          rlca
65C4 07          rlca
65C5 5F          ld e,a
65C6 7A          ld a,d
65C7 E61F        and 1F
65C9 07          rlca
65CA 07          rlca
65CB 07          rlca
65CC 57          ld d,a
65CD 78          ld a,b
65CE 82          add a,d
65CF FEC0        cp C0
65D1 D0          ret nc
65D2 47          ld b,a
65D3 7B          ld a,e
65D4 07          rlca
65D5 CB12        rl d
65D7 0F          rrca
65D8 81          add a,c
65D9 CB12        rl d
65DB 08          ex af,af'
65DC CB1A        rr d
65DE 17          rla
65DF AA          xor d
65E0 E601        and 01
65E2 C0          ret nz
65E3 08          ex af,af'
65E4 4F          ld c,a
65E5 2AF875      ld hl,(75F8)
65E8 ED43F875    ld (75F8),bc
65EC 4D          ld c,l
65ED 44          ld b,h
65EE CDE669      call 69E6
65F1 C30069      jp 6900
65F4 79          ld a,c
65F5 07          rlca
65F6 07          rlca
65F7 07          rlca
65F8 A8          xor b
65F9 E6C7        and C7
65FB A8          xor b
65FC 07          rlca
65FD 07          rlca
65FE 6F          ld l,a
65FF 3A3B75      ld a,(753B)
6602 E640        and 40
6604 2016        jr nz,661C
6606 78          ld a,b
6607 A7          and a
6608 1F          rra
```



```
6609 37          scf
660A 1F          rra
660B A7          and a
660C 1F          rra
660D A8          xor b
660E E6F8        and F8
6610 A8          xor b
6611 67          ld h,a
6612 79          ld a,c
6613 E607        and 07
6615 3C          inc a
6616 ED4B3B75    ld bc,(753B)
661A 47          ld b,a
661B C9          ret
661C 78          ld a,b
661D 07          rlca
661E 07          rlca
661F EE58        xor 58
6621 E603        and 03
6623 EE58        xor 58
6625 67          ld h,a
6626 ED4B3B75    ld bc,(753B)
662A 0600        ld b,00
662C C9          ret
662D 3A3B75      ld a,(753B)
6630 F5          push af
6631 E6BF        and BF
6633 323B75      ld (753B),a
6636 CD3E66      call 663E
6639 F1          pop af
663A 323B75      ld (753B),a
663D C9          ret
663E 79          ld a,c
663F 07          rlca
6640 07          rlca
6641 07          rlca
6642 A8          xor b
6643 E6C7        and C7
6645 A8          xor b
6646 07          rlca
6647 07          rlca
6648 6F          ld l,a
6649 3A3B75      ld a,(753B)
664C E640        and 40
664E 2018        jr nz,6668
6650 78          ld a,b
6651 F607        or 07
6653 0F          rrca
6654 0F          rrca
6655 0F          rrca
6656 A8          xor b
6657 E6F8        and F8
6659 A8          xor b
665A 67          ld h,a
665B 79          ld a,c
665C E607        and 07
665E 3C          inc a
665F 47          ld b,a
```

```
6660 3EFE       ld a,FE
6662 0F         rrca
6663 10FD       djnz 6662
6665 ED4B3B76   ld bc,(763B)
6669 47         ld b,a
666A C9         ret
666B 78         ld a,b
666C 07         rlca
666D 07         rlca
666E EEF8       xor F8
6670 E603       and 03
6672 EEF8       xor F8
6674 67         ld h,a
6675 ED4B3B76   ld bc,(763B)
6679 C9         ret
667A 0E01       ld c,01
667C 1802       jr 6680
667E 0E00       ld c,00
6680 C5         push bc
6681 CD7F67     call 677F
6684 C1         pop bc
6685 47         ld b,a
6686 FE08       cp 08
6688 380C       jr c,6696
668A D609       sub 09
668C 2F         cpl
668D 0D         dec c
668E 067F       ld b,7F
6690 280E       jr z,66A0
6692 CB08       rrc b
6694 180A       jr 66A0
6696 06F8       ld b,F8
6698 0D         dec c
6699 2005       jr nz,66A0
669B 07         rlca
669C 07         rlca
669D 07         rlca
669E 06C7       ld b,C7
66A0 4F         ld c,a
66A1 3A3C76     ld a,(763C)
66A4 A9         xor c
66A5 A0         and b
66A6 A9         xor c
66A7 323C76     ld (763C),a
66AA C9         ret
66AB CD7F67     call 677F
66AE 6F         ld l,a
66AF 2600       ld h,00
66B1 11ED76     ld de,76ED
66B4 19         add hl,de
66B5 7E         ld a,(hl)
66B6 322D77     ld (772D),a
66B9 C9         ret
66BA 0600       ld b,00
66BC 180A       jr 66C8
66BE 0601       ld b,01
66C0 1805       jr 66C8
66C2 0602       ld b,02
```



```
66C4 1802       jr 66C8
66C6 0603       ld b,03
66C8 3A3B76     ld a,(763B)
66CB A8         xor b
66CC E6FC       and FC
66CE A8         xor b
66CF 323B76     ld (763B),a
66D2 C9         ret
66D3 CDA667     call 67A6
66D6 0640       ld b,40
66D8 180E       jr 66E8
66DA 0680       ld b,80
66DC 180A       jr 66E8
66DE 0604       ld b,04
66E0 1806       jr 66E8
66E2 0610       ld b,10
66E4 1802       jr 66E8
66E6 0608       ld b,08
66E8 3A3B76     ld a,(763B)
66EB A8         xor b
66EC 323B76     ld (763B),a
66EF C39468     jp 6894
66F2 00         nop
66F3 00         nop
66F4 2A3B76     ld hl,(763B)
66F7 3AF676     ld a,(76F6)
66FA 67         ld h,a
66FB 22F266     ld (66F2),hl
66FE F608       or 08
6700 32F676     ld (76F6),a
6703 7D         ld a,l
6704 F680       or 80
6706 323B76     ld (763B),a
6709 C9         ret
670A 2AF266     ld hl,(66F2)
670D 7D         ld a,l
670E 323B76     ld (763B),a
6711 7C         ld a,h
6712 32F676     ld (76F6),a
6715 C9         ret
6716 3EFE       ld a,FE
6718 DBFE       in a,(FE)
671A F6E1       or E1
671C 47         ld b,a
671D 3E7F       ld a,7F
671F DBFE       in a,(FE)
6721 F6E2       or E2
6723 A0         and b
6724 47         ld b,a
6725 3E01       ld a,01
6727 DBFE       in a,(FE)
6729 F6E0       or E0
672B A0         and b
672C 3C         inc a
672D C9         ret
672E CD3A6B     call 6B3A
6731 CD0474     call 7404
6734 CD9369     call 6993
```

```
6737 CD1667    call 6716
673A C8        ret z
673B 18F1      jr 672E
673D CD3A6B    call 6B3A
6740 CD0474    call 7404
6743 CD9369    call 6993
6746 CD1667    call 6716
6749 28F2      jr z,673D
674B CDF66B    call 6BF6
674E C3CF6B    jp 6BCF
6751 DD216A67  ld ix,676A
6755 CDB668    call 68B6
6758 CD6268    call 6862
675B CD7467    call 6774
675E FE08      cp 08
6760 30F9      jr nc,675B
6762 32F776    ld (76F7),a
6765 D3FE      out (FE),a
6767 C30D6B    jp 6B0D
676A 09        add hl,bc
676B 03        inc bc
676C 42        ld b,d
676D 4F        ld c,a
676E 52        ld d,d
676F 44        ld b,h
6770 41        ld b,c
6771 203F      jr nz,67B2
6773 27        daa
6774 CDCF6B    call 6BCF
6777 CD7F67    call 677F
677A FEFF      cp FF
677C 28F6      jr z,6774
677E C9        ret
677F ED4B2377  ld bc,(7723)
6783 78        ld a,b
6784 F620      or 20
6786 0600      ld b,00
6788 0F        rrca
6789 04        inc b
678A 30FC      jr nc,6788
678C 78        ld a,b
678D FE06      cp 06
678F D8        ret c
6790 060B      ld b,0B
6792 79        ld a,c
6793 F620      or 20
6795 0F        rrca
6796 05        dec b
6797 30FC      jr nc,6795
6799 78        ld a,b
679A FE06      cp 06
679C 3003      jr nc,67A1
679E 3EFF      ld a,FF
67A0 C9        ret
67A1 FE0A      cp 0A
67A3 D8        ret c
67A4 AF        xor a
67A5 C9        ret
```



```
67A6 3AF676    ld a,(76F6)
67A9 0F        rrca
67AA D0        ret nc
67AB 3F        ccf
67AC 17        rla
67AD 32F676    ld (76F6),a
67B0 2100F8    ld hl,F800
67B3 1100F8    ld de,F800
67B6 010003    ld bc,0300
67B9 EDB0      ldir
67BB C9        ret
67BC 0601      ld b,01
67BE 1802      jr 67C2
67C0 0602      ld b,02
67C2 3A3B76    ld a,(763B)
67C5 E640      and 40
67C7 C0        ret nz
67C8 3AF676    ld a,(76F6)
67CB 210D6B    ld hl,6B0D
67CE E5        push hl
67CF E601      and 01
67D1 20D3      jr nz,67A6
67D3 C5        push bc
67D4 2100F8    ld hl,F800
67D7 1100FB    ld de,FB00
67DA 010003    ld bc,0300
67DD EDB0      ldir
67DF 3AF676    ld a,(76F6)
67E2 F601      or 01
67E4 32F676    ld (76F6),a
67E7 C1        pop bc
67E8 05        dec b
67E9 2026      jr nz,6811
67EB 2101F8    ld hl,F801
67EE 1102F8    ld de,F802
67F1 3638      ld (hl),38
67F3 2B        dec hl
67F4 3678      ld (hl),78
67F6 011E00    ld bc,001E
67F9 EDB0      ldir
67FB 23        inc hl
67FC EDA0      ldi
67FE 2E00      ld l,00
6800 011F00    ld bc,001F
6803 EDB0      ldir
6805 2100F8    ld hl,F800
6808 1140F8    ld de,F840
680B 01C002    ld bc,02C0
680E EDB0      ldir
6810 C9        ret
6811 2100F8    ld hl,F800
6814 110040    ld de,4000
6817 011810    ld bc,1018
681A 7E        ld a,(hl)
681B E6BF      and BF
681D B2        or d
681E 77        ld (hl),a
681F 23        inc hl
```

```
6820 7E        ld a,(hl)
6821 E6BF      and BF
6823 B3        or e
6824 77        ld (hl),a
6825 23        inc hl
6826 10F2      djnz 681A
6828 0610      ld b,10
682A 7A        ld a,d
682B 53        ld d,e
682C 5F        ld e,a
682D 0D        dec c
682E 20EA      jr nz,681A
6830 C9        ret
6831 CDB668    call 68B6
6834 DD7E00    ld a,(ix+00)
6837 D604      sub 04
6839 47        ld b,a
683A 48        ld c,b
683B E5        push hl
683C CD6268    call 6862
683F E1        pop hl
6840 CD5965    call 6559
6843 CD5965    call 6559
6846 E5        push hl
6847 79        ld a,c
6848 90        sub b
6849 C630      add a,30
684B CD7268    call 6872
684E 2C        inc l
684F CD6268    call 6862
6852 E1        pop hl
6853 10EE      djnz 6843
6855 C9        ret
6856 CD3168    call 6831
6859 C5        push bc
685A CD7467    call 6774
685D C1        pop bc
685E B9        cp c
685F D8        ret c
6860 18F7      jr 6859
6862 3E47      ld a,47
6864 08        ex af,af'
6865 DD23      inc ix
6867 DD7E00    ld a,(ix+00)
686A FE27      cp 27
686C C8        ret z
686D CD7268    call 6872
6870 18F3      jr 6865
6872 C5        push bc
6873 E5        push hl
6874 6F        ld l,a
6875 2600      ld h,00
6877 29        add hl,hl
6878 29        add hl,hl
6879 29        add hl,hl
687A 110030    ld de,3000
687D 19        add hl,de
687E EB        ex de,hl
```



```
687F E1        pop hl
6880 E5        push hl
6881 CD8F68    call 688F
6884 25        dec h
6885 CDE764    call 64E7
6888 08        ex af,af'
6889 77        ld (hl),a
688A 08        ex af,af'
688B E1        pop hl
688C 2C        inc l
688D C1        pop bc
688E C9        ret
688F 0608      ld b,08
6891 1A        ld a,(de)
6892 77        ld (hl),a
6893 24        inc h
6894 13        inc de
6895 10FA      djnz 6891
6897 C9        ret
6898 0A        ld a,(bc)
6899 08        ex af,af'
689A 4C        ld c,h
689B 49        ld c,c
689C 4D        ld c,l
689D 50        ld d,b
689E 41        ld b,c
689F 27        daa
68A0 4E        ld c,(hl)
68A1 41        ld b,c
68A2 44        ld b,h
68A3 41        ld b,c
68A4 27        daa
68A5 54        ld d,h
68A6 45        ld b,l
68A7 4C        ld c,h
68A8 41        ld b,c
68A9 2020      jr nz,68CB
68AB 27        daa
68AC 41        ld b,c
68AD 54        ld d,h
68AE 52        ld d,d
68AF 49        ld c,c
68B0 27        daa
68B1 54        ld d,h
68B2 55        ld d,l
68B3 44        ld b,h
68B4 4F        ld c,a
68B5 27        daa
68B6 DD4600    ld b,(ix+00)
68B9 DD4E01    ld c,(ix+01)
68BC 04        inc b
68BD 04        inc b
68BE 214040    ld hl,4040
68C1 3E70      ld a,70
68C3 08        ex af,af'
68C4 3E23      ld a,23
68C6 CD7268    call 6872
68C9 10F9      djnz 68C4
```

```
68CB CD5565    call 6555
68CE 3E23      ld a,23
68D0 CD7268    call 6872
68D3 3E47      ld a,47
68D5 08        ex af,af'
68D6 DD4600    ld b,(ix+00)
68D9 3E20      ld a,20
68DB CD7268    call 6872
68DE 10F9      djnz 68D9
68E0 3E70      ld a,70
68E2 08        ex af,af'
68E3 3E23      ld a,23
68E5 CD7268    call 6872
68E8 CD5565    call 6555
68EB 0D        dec c
68EC 20E0      jr nz,68CE
68EE DD4600    ld b,(ix+00)
68F1 04        inc b
68F2 04        inc b
68F3 3E23      ld a,23
68F5 CD7268    call 6872
68F8 10F9      djnz 68F3
68FA 214240    ld hl,4042
68FD DD23      inc ix
68FF C9        ret
6900 3A3B76    ld a,(763B)
6903 E620      and 20
6905 C49769    call nz,6997
6908 3A3B76    ld a,(763B)
690B E604      and 04
690D CAC669    jp z,69C6
6910 CD2359    call 5923
6913 D8        ret c
6914 CD5769    call 6957
6917 D8        ret c
6918 3A3B76    ld a,(763B)
691B E608      and 08
691D CA056A    jp z,6A05
6920 C32D6A    jp 6A2D
6923 2A2E77    ld hl,(772E)
6926 CD2165    call 6521
6929 7D        ld a,l
692A 07        rlca
692B 07        rlca
692C 07        rlca
692D E6F8      and F8
692F 5F        ld e,a
6930 79        ld a,c
6931 93        sub e
6932 3850      jr c,6984
6934 07        rlca
6935 4F        ld c,a
6936 0F        rrca
6937 0F        rrca
6938 0F        rrca
6939 0F        rrca
693A E61F      and 1F
693C 2A3876    ld hl,(7638)
```



```
693F BD        cp l
6940 3042      jr nc,6984
6942 78        ld a,b
6943 92        sub d
6944 383E      jr c,6984
6946 BC        cp h
6947 303B      jr nc,6984
6949 07        rlca
694A 47        ld b,a
694B 3A3B76    ld a,(763B)
694E E608      and 08
6950 C8        ret z
6951 CB00      rlc b
6953 CB01      rlc c
6955 A7        and a
6956 C9        ret
6957 3AF676    ld a',(76F6)
695A E608      and 08
695C C0        ret nz
695D C5        push bc
695E 2A0274    ld hl,(7402)
6961 CD2165    call 6521
6964 78        ld a,b
6965 92        sub d
6966 3819      jr c,6981
6968 FE20      cp 20
696A 3015      jr nc,6981
696C 7D        ld a,l
696D E61F      and 1F
696F 07        rlca
6970 07        rlca
6971 07        rlca
6972 6F        ld l,a
6973 79        ld a,c
6974 95        sub l
6975 38E0      jr c,6957
6977 FE38      cp 38
6979 3006      jr nc,6981
697B CD9975    call 7599
697E C1        pop bc
697F 37        scf
6980 C9        ret
6981 C1        pop bc
6982 A7        and a
6983 C9        ret
6984 3A3B76    ld a,(763B)
6987 E680      and 80
6989 37        scf
698A C0        ret nz
698B CD636D    call 6D63
698E CD1675    call 7516
6991 37        scf
6992 C9        ret
6993 ED4BF876  ld bc,(76F8)
6997 ED5B3B76  ld de,(763B)
699B 3AF676    ld a,(76F6)
699E E604      and 04
69A0 C2DF70    jp nz,70DF
```

```
69A3 78          ld a,e
69A4 EE20        xor 20
69A6 E6FC        and FC
69A8 F601        or 01
69AA 323B76      ld (763B),a
69AD D5          push de
69AE 3E3F        ld a,3F
69B0 323C76      ld (763C),a
69B3 C5          push bc
69B4 CD0869      call 6908
69B7 C1          pop bc
69B8 D1          pop de
69B9 3A3B76      ld a,(763B)
69BC AB          xor e
69BD E6FC        and FC
69BF AB          xor e
69C0 5F          ld e,a
69C1 ED533B76    ld (763B),de
69C5 C9          ret
69C6 CD5769      call 6957
69C9 D8          ret c
69CA CDF465      call 65F4
69CD 79          ld a,c
69CE E640        and 40
69D0 2022        jr nz,69F4
69D2 3EFE        ld a,FE
69D4 0F          rrca
69D5 10FD        djnz 69D4
69D7 47          ld b,a
69D8 7E          ld a,(hl)
69D9 CB41        bit 0,c
69DB 2001        jr nz,69DE
69DD A0          and b
69DE CB49        bit 1,c
69E0 2002        jr nz,69E4
69E2 A8          xor b
69E3 2F          cpl
69E4 77          ld (hl),a
69E5 C9          ret
69E6 C5          push bc
69E7 CD3E66      call 663E
69EA CDEF69      call 69EF
69ED C1          pop bc
69EE C9          ret
69EF 79          ld a,c
69F0 E640        and 40
69F2 28E4        jr z,69D8
69F4 3A3C76      ld a,(763C)
69F7 47          ld b,a
69F8 7E          ld a,(hl)
69F9 CB41        bit 0,c
69FB 2001        jr nz,69FE
69FD AF          xor a
69FE CB49        bit 1,c
6A00 2001        jr nz,6A03
6A02 A8          xor b
6A03 77          ld (hl),a
6A04 C9          ret
```



```
6A05 CDF465      call 65F4
6A08 79          ld a,c
6A09 E640        and 40
6A0B 200F        jr nz,6A1C
6A0D 3E7E        ld a,7E
6A0F 0F          rrca
6A10 10FD        djnz 6A0F
6A12 47          ld b,a
6A13 CDD869      call 69D8
6A16 47          ld b,a
6A17 CDF164      call 64F1
6A1A 70          ld (hl),b
6A1B C9          ret
6A1C 7D          ld a,l
6A1D E6DE        and DE
6A1F 6F          ld l,a
6A20 CDF469      call 69F4
6A23 23          inc hl
6A24 77          ld (hl),a
6A25 112000      ld de,0020
6A28 19          add hl,de
6A29 77          ld (hl),a
6A2A 2D          dec l
6A2B 77          ld (hl),a
6A2C C9          ret
6A2D CDF465      call 65F4
6A30 79          ld a,c
6A31 E640        and 40
6A33 2013        jr nz,6A48
6A35 3E1E        ld a,1E
6A37 0F          rrca
6A38 10FD        djnz 6A37
6A3A 47          ld b,a
6A3B CDD869      call 69D8
6A3E 0603        ld b,03
6A40 4F          ld c,a
6A41 CDF164      call 64F1
6A44 71          ld (hl),c
6A45 10FA        djnz 6A41
6A47 C9          ret
6A48 7D          ld a,l
6A49 E69C        and 9C
6A4B 6F          ld l,a
6A4C CDF469      call 69F4
6A4F 010300      ld bc,0003
6A52 5D          ld e,l
6A53 54          ld d,h
6A54 13          inc de
6A55 3E04        ld a,04
6A57 E5          push hl
6A58 EDB0        ldir
6A5A D1          pop de
6A5B 212000      ld hl,0020
6A5E 19          add hl,de
6A5F EB          ex de,hl
6A60 D5          push de
6A61 010400      ld bc,0004
6A64 3D          dec a
```

```
6A65 20F1       jr nz,6A58
6A67 D1         pop de
6A68 C9         ret
6A69 DD219868   ld ix,6898
6A6D CD5568     call 6855
6A70 210D6B     ld hl,6B0D
6A73 E5         push hl
6A74 A7         and a
6A75 C8         ret z
6A76 3D         dec a
6A77 2806       jr z,6A7F
6A79 3D         dec a
6A7A 2811       jr z,6A8D
6A7C CD8D6A     call 6A8D
6A7F 2100E0     ld hl,E000
6A82 1101E0     ld de,E001
6A85 01FF17     ld bc,17FF
6A88 3600       ld (hl),00
6A8A EDB0       ldir
6A8C C9         ret
6A8D 2100F8     ld hl,F800
6A90 1101F8     ld de,F801
6A93 3A3C76     ld a,(763C)
6A96 77         ld (hl),a
6A97 01FF02     ld bc,02FF
6A9A EDB0       ldir
6A9C 3AF676     ld a,(76F6)
6A9F E6FE       and FE
6AA1 32F676     ld (76F6),a
6AA4 C9         ret
6AA5 DD21EE6A   ld ix,6AEE
6AA9 CD5568     call 6855
6AAC 210D6B     ld hl,6B0D
6AAF E5         push hl
6AB0 A7         and a
6AB1 C8         ret z
6AB2 3D         dec a
6AB3 2806       jr z,6ABB
6AB5 3D         dec a
6AB6 2813       jr z,6ACB
6AB8 CDCB6A     call 6ACB
6ABB 2100E0     ld hl,E000
6ABE 010018     ld bc,1800
6AC1 7E         ld a,(hl)
6AC2 2F         cpl
6AC3 77         ld (hl),a
6AC4 23         inc hl
6AC5 0B         dec bc
6AC6 78         ld a,b
6AC7 B1         or c
6AC8 20F7       jr nz,6AC1
6ACA C9         ret
6ACB CDA667     call 67A6
6ACE 2100F8     ld hl,F800
6AD1 010003     ld bc,0300
6AD4 7E         ld a,(hl)
6AD5 5F         ld e,a
6AD6 0F         rrca
```



```
6AD7 0F         rrca
6AD8 0F         rrca
6AD9 AB         xor e
6ADA E607       and 07
6ADC AB         xor e
6ADD 57         ld d,a
6ADE 7B         ld a,e
6ADF 07         rlca
6AE0 07         rlca
6AE1 07         rlca
6AE2 AA         xor d
6AE3 E638       and 38
6AE5 AA         xor d
6AE6 77         ld (hl),a
6AE7 23         inc hl
6AE8 0B         dec bc
6AE9 78         ld a,b
6AEA B1         or c
6AEB 20E7       jr nz,6AD4
6AED C9         ret
6AEE 0A         ld a,(bc)
6AEF 08         ex af,af'
6AF0 49         ld c,c
6AF1 4E         ld c,(hl)
6AF2 56         ld d,(hl)
6AF3 45         ld b,l
6AF4 52         ld d,d
6AF5 54         ld d,h
6AF6 27         daa
6AF7 4E         ld c,(hl)
6AF8 41         ld b,c
6AF9 44         ld b,h
6AFA 41         ld b,c
6AFB 27         daa
6AFC 54         ld d,h
6AFD 45         ld b,l
6AFE 4C         ld c,h
6AFF 41         ld b,c
6B00 2020       jr nz,6B22
6B02 27         daa
6B03 41         ld b,c
6B04 54         ld d,h
6B05 52         ld d,d
6B06 49         ld c,c
6B07 27         daa
6B08 54         ld d,h
6B09 55         ld d,l
6B0A 44         ld b,h
6B0B 4F         ld c,a
6B0C 27         daa
6B0D 3A3B76     ld a,(763B)
6B10 E6DF       and DF
6B12 323B76     ld (763B),a
6B15 47         ld b,a
6B16 E604       and 04
6B18 2814       jr z,6B2E
6B1A 78         ld a,b
6B1B 210040     ld hl,4000
```

```
6B1E ED5B2E77  ld de,(772E)
6B22 ED4B3876  ld bc,(7638)
6B26 E608      and 08
6B28 CAC06C    jp z,6CC0
6B2B C3366C    jp 6C36
6B2E 2100E0    ld hl,E000
6B31 110040    ld de,4000
6B34 01001B    ld bc,1B00
6B37 EDB0      ldir
6B39 C9        ret
6B3A 3A3B76    ld a,(763B)
6B3D E614      and 14
6B3F FE10      cp 10
6B41 C0        ret nz
6B42 2A2E77    ld hl,(772E)
6B45 CDE764    call 64E7
6B48 E5        push hl
6B49 3A3776    ld a,(7637)
6B4C 77        ld (hl),a
6B4D 3A3B76    ld a,(763B)
6B50 E608      and 08
6B52 010F00    ld bc,000F
6B55 2803      jr z,6B5A
6B57 010700    ld bc,0007
6B5A 5D        ld e,l
6B5B 54        ld d,h
6B5C 13        inc de
6B5D EDB0      ldir
6B5F E1        pop hl
6B60 3A3B76    ld a,(763B)
6B63 E608      and 08
6B65 011008    ld bc,0810
6B68 2803      jr z,6B6D
6B6A 010805    ld bc,0508
6B6D C5        push bc
6B6E E5        push hl
6B6F 112000    ld de,0020
6B72 EB        ex de,hl
6B73 19        add hl,de
6B74 EB        ex de,hl
6B75 0600      ld b,00
6B77 EDB0      ldir
6B79 E1        pop hl
6B7A 112000    ld de,0020
6B7D 19        add hl,de
6B7E C1        pop bc
6B7F 10EC      djnz 6B6D
6B81 3A3776    ld a,(7637)
6B84 4F        ld c,a
6B85 C603      add a,03
6B87 A9        xor c
6B88 E607      and 07
6B8A A9        xor c
6B8B D608      sub 08
6B8D E63F      and 3F
6B8F 323776    ld (7637),a
6B92 AF        xor a
6B93 C9        ret
```



```
6B94 3A3B76    ld a,(763B)
6B97 E608      and 08
6B99 011060    ld bc,6010
6B9C 2803      jr z,6BA1
6B9E 010830    ld bc,3008
6BA1 ED4338 76 ld (7638),bc
6BA5 ED5B2E77  ld de,(772E)
6BA9 7B        ld a,e
6BAA E61F      and 1F
6BAC 3D        dec a
6BAD 81        add a,c
6BAE FE20      cp 20
6BB0 3807      jr c,6BB9
6BB2 7B        ld a,e
6BB3 E6E0      and E0
6BB5 C620      add a,20
6BB7 91        sub c
6BB8 5F        ld e,a
6BB9 CDC064    call 64C0
6BBC 80        add a,b
6BBD 3D        dec a
6BBE FEC0      cp C0
6BC0 3806      jr c,6BC8
6BC2 3EC0      ld a,C0
6BC4 90        sub b
6BC5 CDD064    call 64D0
6BC8 ED532E77  ld (772E),de
6BCC C30D6B    jp 6B0D
6BCF CD0F6C    call 6C0F
6BD2 CDF56B    call 6BF5
6BD5 CD1C6C    call 6C1C
6BD8 C8        ret z
6BD9 CDF56B    call 6BF5
6BDC 3A2077    ld a,(7720)
6BDF 57        ld d,a
6BE0 E6FD      and FD
6BE2 322077    ld (7720),a
6BE5 AA        xor d
6BE6 5F        ld e,a
6BE7 3A2777    ld a,(7727)
6BEA 57        ld d,a
6BEB E6FE      and FE
6BED 322777    ld (7727),a
6BF0 AA        xor d
6BF1 B3        or e
6BF2 322877    ld (7728),a
6BF5 C9        ret
6BF6 CD9369    call 6993
6BF9 3A2D77    ld a,(772D)
6BFC A7        and a
6BFD C8        ret z
6BFE 6F        ld l,a
6BFF 3AF576    ld a,(76F5)
6C02 E604      and 04
6C04 2803      jr z,6C09
6C06 2A2C77    ld hl,(772C)
6C09 10FE      djnz 6C09
6C0B 2D        dec l
```

```
6C0C 20F8      jr nz,6C09
6C0E C9        ret
6C0F 212077    ld hl,7720
6C12 112177    ld de,7721
6C15 010700    ld bc,0007
6C18 361F      ld (hl),1F
6C1A EDB0      ldir
6C1C 212077    ld hl,7720
6C1F 01FE7F    ld bc,7FFE
6C22 1E00      ld e,00
6C24 ED78      in a,(c)
6C26 2F        cpl
6C27 A6        and (hl)
6C28 77        ld (hl),a
6C29 B3        or e
6C2A 5F        ld e,a
6C2B 23        inc hl
6C2C CB08      rrc b
6C2E 38F4      jr c,6C24
6C30 78        ld a,e
6C31 322877    ld (7728),a
6C34 A7        and a
6C35 C9        ret
6C36 E5        push hl
6C37 D5        push de
6C38 C5        push bc
6C39 C5        push bc
6C3A D5        push de
6C3B E5        push hl
6C3C 41        ld b,c
6C3D C5        push bc
6C3E D5        push de
6C3F 1A        ld a,(de)
6C40 4F        ld c,a
6C41 0604      ld b,04
6C43 CB11      rl c
6C45 9F        sbc a,a
6C46 5F        ld e,a
6C47 CB11      rl c
6C49 9F        sbc a,a
6C4A AB        xor e
6C4B E60F      and 0F
6C4D AB        xor e
6C4E 77        ld (hl),a
6C4F 23        inc hl
6C50 10F1      djnz 6C43
6C52 D1        pop de
6C53 13        inc de
6C54 C1        pop bc
6C55 10E5      djnz 6C3D
6C57 E1        pop hl
6C58 0603      ld b,03
6C5A C5        push bc
6C5B E5        push hl
6C5C 5D        ld e,l
6C5D 54        ld d,h
6C5E 14        inc d
6C5F 0600      ld b,00
```



```
6C61 CB21      sla c
6C63 CB21      sla c
6C65 EDB0      ldir
6C67 E1        pop hl
6C68 24        inc h
6C69 C1        pop bc
6C6A 10EE      djnz 6C5A
6C6C CDF164    call 64F1
6C6F EB        ex de,hl
6C70 E1        pop hl
6C71 CDF164    call 64F1
6C74 EB        ex de,hl
6C75 C1        pop bc
6C76 10C1      djnz 6C39
6C78 C1        pop bc
6C79 E1        pop hl
6C7A CDE764    call 64E7
6C7D C6A0      add a,A0
6C7F 57        ld d,a
6C80 5D        ld e,l
6C81 E1        pop hl
6C82 CDE764    call 64E7
6C85 CB38      srl b
6C87 CB38      srl b
6C89 CB38      srl b
6C8B CB21      sla c
6C8D CB21      sla c
6C8F C5        push bc
6C90 D5        push de
6C91 E5        push hl
6C92 41        ld b,c
6C93 1A        ld a,(de)
6C94 77        ld (hl),a
6C95 23        inc hl
6C96 77        ld (hl),a
6C97 23        inc hl
6C98 77        ld (hl),a
6C99 23        inc hl
6C9A 77        ld (hl),a
6C9B 23        inc hl
6C9C 13        inc de
6C9D 10F4      djnz 6C93
6C9F 0603      ld b,03
6CA1 E1        pop hl
6CA2 C5        push bc
6CA3 112000    ld de,0020
6CA6 EB        ex de,hl
6CA7 19        add hl,de
6CA8 E5        push hl
6CA9 EB        ex de,hl
6CAA 0600      ld b,00
6CAC EDB0      ldir
6CAE CB38      srl b
6CB0 E1        pop hl
6CB1 C1        pop bc
6CB2 10EE      djnz 6CA2
6CB4 112000    ld de,0020
6CB7 19        add hl,de
```

```
6CB6 EB        ex de,hl
6CB9 C1        pop bc
6CBA 09        add hl,bc
6CBB EB        ex de,hl
6CBC C1        pop bc
6CBD 10D0      djnz 6C8F
6CBF C9        ret
6CC0 E5        push hl
6CC1 D5        push de
6CC2 C5        push bc
6CC3 C5        push bc
6CC4 E5        push hl
6CC5 D5        push de
6CC6 E5        push hl
6CC7 41        ld b,c
6CC8 D5        push de
6CC9 C5        push bc
6CCA 1A        ld a,(de)
6CCB 4F        ld c,a
6CCC 0604      ld b,04
6CCE AF        xor a
6CCF CB01      rlc c
6CD1 17        rla
6CD2 17        rla
6CD3 10FA      djnz 6CCF
6CD5 5F        ld e,a
6CD6 0F        rrca
6CD7 B3        or e
6CD8 77        ld (hl),a
6CD9 23        inc hl
6CDA 0604      ld b,04
6CDC AF        xor a
6CDD CB01      rlc c
6CDF 17        rla
6CE0 17        rla
6CE1 10FA      djnz 6CDD
6CE3 5F        ld e,a
6CE4 0F        rrca
6CE5 B3        or e
6CE6 77        ld (hl),a
6CE7 23        inc hl
6CE8 C1        pop bc
6CE9 D1        pop de
6CEA 1C        inc e
6CEB 10DB      djnz 6CC8
6CED E1        pop hl
6CEE 6D        ld e,l
6CEF 54        ld d,h
6CF0 14        inc d
6CF1 CB21      sla c
6CF3 EDB0      ldir
6CF5 E1        pop hl
6CF6 CDF164    call 64F1
6CF9 EB        ex de,hl
6CFA E1        pop hl
6CFB 24        inc h
6CFC CDF164    call 64F1
6CFF C1        pop bc
```



```
6D00 10C1      djnz 6CC3
6D02 C1        pop bc
6D03 E1        pop hl
6D04 CDE764    call 64E7
6D07 C6A0      add a,A0
6D09 57        ld d,a
6D0A 5D        ld e,l
6D0B E1        pop hl
6D0C CDE764    call 64E7
6D0F 78        ld a,b
6D10 0F        rrca
6D11 0F        rrca
6D12 0F        rrca
6D13 E61F      and 1F
6D15 47        ld b,a
6D16 C5        push bc
6D17 E5        push hl
6D18 D5        push de
6D19 41        ld b,c
6D1A C5        push bc
6D1B 012000    ld bc,0020
6D1E 1A        ld a,(de)
6D1F 77        ld (hl),a
6D20 09        add hl,bc
6D21 77        ld (hl),a
6D22 2C        inc l
6D23 77        ld (hl),a
6D24 ED42      sbc hl,bc
6D26 77        ld (hl),a
6D27 2C        inc l
6D28 1C        inc e
6D29 C1        pop bc
6D2A 10EE      djnz 6D1A
6D2C E1        pop hl
6D2D 0E20      ld c,20
6D2F 09        add hl,bc
6D30 EB        ex de,hl
6D31 E1        pop hl
6D32 0E40      ld c,40
6D34 09        add hl,bc
6D35 C1        pop bc
6D36 10DE      djnz 6D16
6D38 C9        ret
6D39 2A2E77    ld hl,(772E)
6D3C CD2165    call 6521
6D3F 7D        ld a,l
6D40 07        rlca
6D41 07        rlca
6D42 07        rlca
6D43 E6F8      and F8
6D45 5F        ld e,a
6D46 3A3B75    ld a,(753B)
6D49 212018    ld hl,1820
6D4C E608      and 08
6D4E 2001      jr nz,6D51
6D50 29        add hl,hl
6D51 19        add hl,de
6D52 ED4BF875  ld bc,(75F8)
```

```
6D56 22F876     ld (76F8),hl
6D59 3A3B76     ld a,(763B)
6D5C E620       and 20
6D5E C8         ret z
6D5F C39769     jp 6997
6D62 C9         ret
6D63 ED4BF876   ld bc,(76F8)
6D67 3A3B76     ld a,(763B)
6D6A E608       and 08
6D6C 2814       jr z,6D82
6D6E 7A         ld a,c
6D6F D61F       sub 1F
6D71 3001       jr nc,6D74
6D73 AF         xor a
6D74 E6F8       and F8
6D76 4F         ld c,a
6D77 78         ld a,b
6D78 D617       sub 17
6D7A 3001       jr nc,6D7D
6D7C AF         xor a
6D7D E6F8       and F8
6D7F 47         ld b,a
6D80 1812       jr 6D94
6D82 79         ld a,c
6D83 D63F       sub 3F
6D85 3001       jr nc,6D88
6D87 AF         xor a
6D88 E6F8       and F8
6D8A 4F         ld c,a
6D8B 78         ld a,b
6D8C D62F       sub 2F
6D8E 3001       jr nc,6D91
6D90 AF         xor a
6D91 E6F8       and F8
6D93 47         ld b,a
6D94 CD2D66     call 662D
6D97 222E77     ld (772E),hl
6D9A C39466     jp 6694
6D9D 11FFFF     ld de,FFFF
6DA0 1821       jr 6DC3
6DA2 1100FF     ld de,FF00
6DA5 181C       jr 6DC3
6DA7 1101FF     ld de,FF01
6DAA 1817       jr 6DC3
6DAC 11FF00     ld de,00FF
6DAF 1812       jr 6DC3
6DB1 110100     ld de,0001
6DB4 180D       jr 6DC3
6DB6 11FF01     ld de,01FF
6DB9 1808       jr 6DC3
6DBB 110001     ld de,0100
6DBE 1803       jr 6DC3
6DC0 110101     ld de,0101
6DC3 210D6B     ld hl,6B0D
6DC6 E3         ex (sp),hl
6DC7 D5         push de
6DC8 3A3B76     ld a,(763B)
6DCB E640       and 40
```



```
6DCD 2872       jr z,6E41
6DCF 7A         ld a,d
6DD0 A7         and a
6DD1 2838       jr z,6E0E
6DD3 F2F36D     jp p,6DF3
6DD6 2100F8     ld hl,F800
6DD9 110077     ld de,7700
6DDC 012000     ld bc,0020
6DDF EDB0       ldir
6DE1 1100F8     ld de,F800
6DE4 01E002     ld bc,02E0
6DE7 EDB0       ldir
6DE9 210077     ld hl,7700
6DEC 012000     ld bc,0020
6DEF EDB0       ldir
6DF1 181B       jr 6E0E
6DF3 21FFFA     ld hl,FAFF
6DF6 111F77     ld de,771F
6DF9 012000     ld bc,0020
6DFC EDB8       lddr
6DFE 11FFFA     ld de,FAFF
6E01 01E002     ld bc,02E0
6E04 EDB8       lddr
6E06 211F77     ld hl,771F
6E09 012000     ld bc,0020
6E0C EDB8       lddr
6E0E D1         pop de
6E0F 7B         ld a,e
6E10 A7         and a
6E11 C8         ret z
6E12 F22B6E     jp p,6E2B
6E15 2101F8     ld hl,F801
6E18 1100F8     ld de,F800
6E1B 0618       ld b,18
6E1D C5         push bc
6E1E 1A         ld a,(de)
6E1F 011F00     ld bc,001F
6E22 EDB0       ldir
6E24 12         ld (de),a
6E25 23         inc hl
6E26 13         inc de
6E27 C1         pop bc
6E28 10F3       djnz 6E1D
6E2A C9         ret
6E2B 21FEFA     ld hl,FAFE
6E2E 11FFFA     ld de,FAFF
6E31 0618       ld b,18
6E33 C5         push bc
6E34 1A         ld a,(de)
6E35 011F00     ld bc,001F
6E38 EDB8       lddr
6E3A 12         ld (de),a
6E3B 1B         dec de
6E3C 2B         dec hl
6E3D C1         pop bc
6E3E 10F3       djnz 6E33
6E40 C9         ret
6E41 7A         ld a,d
```

```
6E42 A7        and a
6E43 2853      jr z,6E98
6E45 F2716E    jp p,6E71
6E48 2100E0    ld hl,E000
6E4B 110077    ld de,7700
6E4E 012000    ld bc,0020
6E51 EDB0      ldir
6E53 2100E1    ld hl,E100
6E56 1100E0    ld de,E000
6E59 E5        push hl
6E5A 012000    ld bc,0020
6E5D EDB0      ldir
6E5F D1        pop de
6E60 6B        ld l,e
6E61 62        ld h,d
6E62 CDF164    call 64F1
6E65 20F2      jr nz,6E59
6E67 2100077   ld hl,7700
6E6A 012000    ld bc,0020
6E6D EDB0      ldir
6E6F 1827      jr 6E98
6E71 21FFF7    ld hl,F7FF
6E74 111F77    ld de,771F
6E77 012000    ld bc,0020
6E7A EDB8      lddr
6E7C 11FFF7    ld de,F7FF
6E7F 21FFF6    ld hl,F6FF
6E82 E5        push hl
6E83 012000    ld bc,0020
6E86 EDB8      lddr
6E88 E1        pop hl
6E89 5D        ld e,l
6E8A 54        ld d,h
6E8B CD0D65    call 650D
6E8E 20F2      jr nz,6E82
6E90 211F77    ld hl,771F
6E93 012000    ld bc,0020
6E96 EDB8      lddr
6E98 D1        pop de
6E99 7B        ld a,e
6E9A A7        and a
6E9B C8        ret z
6E9C F2B96E    jp p,6EB9
6E9F 21FFE0    ld hl,E01F
6EA2 06C0      ld b,C0
6EA4 C5        push bc
6EA5 E5        push hl
6EA6 0620      ld b,20
6EA8 CB16      rl (hl)
6EAA 2B        dec hl
6EAB 10FB      djnz 6EA8
6EAD E1        pop hl
6EAE CB1E      rr (hl)
6EB0 CB06      rlc (hl)
6EB2 CDF164    call 64F1
6EB5 C1        pop bc
6EB6 10EC      djnz 6EA4
6EB8 C9        ret
```



```
6EB9 2100E0    ld hl,E000
6EBC 06C0      ld b,C0
6EBE C5        push bc
6EBF E5        push hl
6EC0 0620      ld b,20
6EC2 CB1E      rr (hl)
6EC4 23        inc hl
6EC5 10FB      djnz 6EC2
6EC7 E1        pop hl
6EC8 CB16      rl (hl)
6ECA CB0E      rrc (hl)
6ECC CDF164    call 64F1
6ECF C1        pop bc
6ED0 10EC      djnz 6EBE
6ED2 C9        ret
6ED3 78        ld a,b
6ED4 B6        or (hl)
6ED5 77        ld (hl),a
6ED6 7C        ld a,h
6ED7 F6E0      or E0
6ED9 67        ld h,a
6EDA 78        ld a,b
6EDB A3        and e
6EDC B6        or (hl)
6EDD 77        ld (hl),a
6EDE 7C        ld a,h
6EDF E65F      and 5F
6EE1 67        ld h,a
6EE2 C9        ret
6EE3 78        ld a,b
6EE4 A6        and (hl)
6EE5 C9        ret
6EE6 3AFFBF    ld a,(BFFF)
6EE9 FEFF      cp FF
6EEB C0        ret nz
6EEC 79        ld a,c
6EED 0F        rrca
6EEE 0F        rrca
6EEF 0F        rrca
6EF0 AC        xor h
6EF1 E6E0      and E0
6EF3 AC        xor h
6EF4 08        ex af,af'
6EF5 7D        ld a,l
6EF6 D9        exx
6EF7 6F        ld l,a
6EF8 08        ex af,af'
6EF9 67        ld h,a
6EFA E3        ex (sp),hl
6EFB E5        push hl
6EFC D9        exx
6EFD C9        ret
6EFE 21003C    ld hl,3C00
6F01 1818      jr 6F1B
6F03 11034E    ld de,4E03
6F06 55        ld d,l
6F07 4D        ld c,l
6F08 45        ld b,l
```

```
5F09 52          ld d,d
5F0A 4F          ld c,a
5F0B 2020        jr nz,5F2D
5F0D 50          ld d,b
5F0E 41          ld b,c
5F0F 44          ld b,h
5F10 52          ld d,d
5F11 41          ld b,c
5F12 4F          ld c,a
5F13 3F          ccf
5F14 27          daa
5F15 CD3270      call 7032
5F18 DA0D68      jp c,680D
5F1B 22FC76      ld (76FC),hl
5F1E D9          exx
5F1F E5          push hl
5F20 CDF466      call 66F4
5F23 21FFFF      ld hl,FFFF
5F26 22FEBF      ld (BFFE),hl
5F29 E5          push hl
5F2A AF          xor a
5F2B 323B76      ld (763B),a
5F2E CD0B70      call 700B
5F31 ED4BF876    ld bc,(76F8)
5F35 CDF465      call 65F4
5F38 48          ld c,b
5F39 0D          dec c
5F3A 3E01        ld a,01
5F3C 0F          rrca
5F3D 10FD        djnz 5F3C
5F3F 47          ld b,a
5F40 7C          ld a,h
5F41 E607        and 07
5F43 ED55FC76    ld (de),(76FC)
5F47 83          add a,e
5F48 5F          ld e,a
5F49 7A          ld a,d
5F4A CE00        adc a,00
5F4C 57          ld d,a
5F4D 1A          ld a,(de)
5F4E 5F          ld e,a
5F4F 1600        ld d,00
5F51 CD4365      call 6543
5F54 3808        jr c,5F5E
5F56 CDE36E      call 6EE3
5F59 28F6        jr z,5F51
5F5B CD3265      call 6532
5F5E CDD36E      call 6ED3
5F61 CDF164      call 64F1
5F64 2810        jr z,5F76
5F66 CDE36E      call 6EE3
5F69 2804        jr z,5F6F
5F6B CB6A        res 1,d
5F6D 1807        jr 5F76
5F6F CB4A        bit 1,d
5F71 CCE66E      call z,6EE6
5F74 CBCA        set 1,d
5F76 2A2A77      ld hl,(772A)
```



```
5F79 CD0D68      call 680D
5F7C 2810        jr z,5F8E
5F7E CDE36E      call 6EE3
5F81 2804        jr z,5F87
5F83 CB82        res 0,d
5F85 1807        jr 5F8E
5F87 CB42        bit 0,d
5F89 CCE66E      call z,6EE6
5F8C CBC2        set 0,d
5F8E 2A2A77      ld hl,(772A)
5F91 CD3265      call 6532
5F94 3805        jr c,5F9B
5F96 CDE36E      call 6EE3
5F99 28C3        jr z,5F5E
5F9B E1          pop hl
5F9C 7C          ld a,h
5F9D E6E0        and E0
5F9F 07          rlca
5FA0 07          rlca
5FA1 07          rlca
5FA2 4F          ld c,a
5FA3 3C          inc a
5FA4 47          ld b,a
5FA5 7C          ld a,h
5FA6 E61F        and 1F
5FA8 F640        or 40
5FAA 67          ld h,a
5FAB FE58        cp 58
5FAD 3808        jr c,5F3A
5FAF CD0A67      call 670A
5FB2 E1          pop hl
5FB3 D9          exx
5FB4 C30D68      jp 680D
5FB7 13          inc de
5FB8 0653        ld b,53
5FBA 41          ld b,c
5FBB 4C          ld c,h
5FBC 56          ld d,(hl)
5FBD 41          ld b,c
5FBE 2F          cpl
5FBF 52          ld d,d
5FC0 45          ld b,l
5FC1 43          ld b,e
5FC2 55          ld d,l
5FC3 50          ld d,b
5FC4 45          ld b,l
5FC5 52          ld d,d
5FC6 41          ld b,c
5FC7 27          daa
5FC8 54          ld d,h
5FC9 45          ld b,l
5FCA 4D          ld c,l
5FCB 50          ld d,b
5FCC 4F          ld c,a
5FCD 52          ld d,d
5FCE 41          ld b,c
5FCF 52          ld d,d
5FD0 49          ld c,c
```

```
6FD1 41          ld b,c
6FD2 4D          ld c,l
6FD3 45          ld b,l
6FD4 4E          ld c,(hl)
6FD5 54          ld d,h
6FD6 45          ld b,l
6FD7 27          daa
6FD8 50          ld d,b
6FD9 52          ld d,d
6FDA 47          ld c,a
6FDB 56          ld d,(hl)
6FDC 49          ld c,c
6FDD 53          ld d,e
6FDE 47          ld c,a
6FDF 52          ld d,d
6FE0 49          ld c,c
6FE1 41          ld b,c
6FE2 4D          ld c,l
6FE3 45          ld b,l
6FE4 4E          ld c,(hl)
6FE5 54          ld d,h
6FE6 45          ld b,l
6FE7 27          daa
6FE8 DD21B76F    ld ix,6FB7
6FEC CD6668      call 6866
6FEF 2E00        ld l,00
6FF1 0F          rrca
6FF2 0F          rrca
6FF3 0F          rrca
6FF4 C680        add a,80
6FF6 67          ld h,a
6FF7 C9          ret
6FF8 CDE86F      call 6FE8
6FFB EB          ex de,hl
6FFC 1810        jr 700E
6FFE CDE86F      call 6FE8
7001 3AF676      ld a,(76F6)
7004 E6FE        and FE
7006 32F676      ld (76F6),a
7009 181C        jr 7027
700B 110080      ld de,8000
700E 2100E0      ld hl,E000
7011 010018      ld bc,1800
7014 EDB0        ldir
7016 3AF676      ld a,(76F6)
7019 E601        and 01
701B 010003      ld bc,0300
701E 280D        jr z,702D
7020 26FB        ld h,FB
7022 1809        jr 702D
7024 210040      ld hl,4000
7027 1100E0      ld de,E000
702A 01001B      ld bc,1B00
702D EDB0        ldir
702F C30D6B      jp 6B0D
7032 DD21035F    ld ix,5F03
7036 CD8668      call 6886
7039 CD6268      call 6862
```



```
703C CD7467      call 6774
703F 87          add a,a
7040 87          add a,a
7041 87          add a,a
7042 6F          ld l,a
7043 2600        ld h,00
7045 3A2877      ld a,(7728)
7048 FE02        cp 02
704A 3F          ccf
704B D8          ret c
704C A7          and a
704D 114D76      ld de,764D
7050 2803        jr z,7055
7052 119D76      ld de,769D
7055 19          add hl,de
7056 C9          ret
7057 CD3270      call 7032
705A DA0D6B      jp c,6B0D
705D E5          push hl
705E ED4BF876    ld bc,(76F8)
7062 CD2D66      call 662D
7065 7C          ld a,h
7066 E6F8        and F8
7068 67          ld h,a
7069 D1          pop de
706A 0608        ld b,08
706C 7E          ld a,(hl)
706D 12          ld (de),a
706E 13          inc de
706F 24          inc h
7070 10FA        djnz 706C
7072 C30D6B      jp 6B0D
7075 DD21A070    ld ix,70A0
7079 CDB566      call 66B5
707C CD6266      call 6662
707F CD2E67      call 672E
7082 CD3D67      call 673D
7085 CDA770      call 70A7
7088 E6DF        and DF
708A D641        sub 41
708C 38F4        jr c,7082
708E FE15        cp 15
7090 D20D6B      jp nc,6B0D
7093 2600        ld h,00
7095 87          add a,a
7096 87          add a,a
7097 87          add a,a
7098 6F          ld l,a
7099 ED5B7B5C    ld de,(5C7B)
709D 19          add hl,de
709E 18BD        jr 705D
70A0 0603        ld b,03
70A2 55          ld d,l
70A3 44          ld b,h
70A4 47          ld b,a
70A5 3F          ccf
70A6 27          daa
70A7 11A861      ld de,61A8
```

```
70AA 3A2877    ld a,(7728)
70AD A7        and a
70AE 2809      jr z,70B9
70B0 11D061    ld de,61D0
70B3 3D        dec a
70B4 2803      jr z,70B9
70B6 11F861    ld de,61F8
70B9 212077    ld hl,7720
70BC 0605      ld b,05
70BE 0E08      ld c,08
70C0 CB1E      rr (hl)
70C2 380B      jr c,70CF
70C4 13        inc de
70C5 10F9      djnz 70C0
70C7 23        inc hl
70C8 0D        dec c
70C9 0605      ld b,05
70CB 20F3      jr nz,70C0
70CD AF        xor a
70CE C9        ret
70CF 1A        ld a,(de)
70D0 C9        ret
70D1 3AF676    ld a,(76F6)
70D4 E6FB      and FB
70D6 32F676    ld (76F6),a
70D9 3A3B76    ld a,(763B)
70DC E620      and 20
70DE C8        ret z
70DF 3A3B76    ld a,(763B)
70E2 EE20      xor 20
70E4 323B76    ld (763B),a
70E7 E620      and 20
70E9 DD2100A0  ld ix,A000
70ED 280F      jr z,70FE
70EF 3AF676    ld a,(76F6)
70F2 E602      and 02
70F4 DD213D76  ld ix,763D
70F8 2804      jr z,70FE
70FA DD214576  ld ix,7645
70FE 3A3B76    ld a,(763B)
7101 ED4BF876  ld bc,(76F8)
7105 E604      and 04
7107 2009      jr nz,7112
7109 CDF465    call 65F4
710C DDE5      push ix
710E D1        pop de
710F C38F68    jp 688F
7112 CD2359    call 5923
7115 D8        ret c
7116 2836      jr z,714E
7118 CDF465    call 65F4
711B 0608      ld b,08
711D C5        push bc
711E DD5E00    ld e,(ix+00)
7121 0604      ld b,04
7123 C5        push bc
7124 CB03      rlc e
7126 9F        sbc a,a
```



```
7127 4F        ld c,a
7128 CB03      rlc e
712A 9F        sbc a,a
712B A9        xor c
712C E60F      and 0F
712E A9        xor c
712F 0604      ld b,04
7131 77        ld (hl),a
7132 24        inc h
7133 10FC      djnz 7131
7135 7C        ld a,h
7136 D604      sub 04
7138 67        ld h,a
7139 C1        pop bc
713A 2C        inc l
713B 10E6      djnz 7123
713D C1        pop bc
713E DD23      inc ix
7140 7D        ld a,l
7141 D604      sub 04
7143 6F        ld l,a
7144 7C        ld a,h
7145 C604      add a,04
7147 67        ld h,a
7148 CDF664    call 64F6
714B 10D0      djnz 711D
714D C9        ret
714E CDF465    call 65F4
7151 0608      ld b,08
7153 C5        push bc
7154 DD4E00    ld c,(ix+00)
7157 DD23      inc ix
7159 0604      ld b,04
715B AF        xor a
715C CB11      rl c
715E 17        rla
715F 17        rla
7160 10FA      djnz 715C
7162 47        ld b,a
7163 0F        rrca
7164 B0        or b
7165 77        ld (hl),a
7166 0604      ld b,04
7168 AF        xor a
7169 CB11      rl c
716B 17        rla
716C 17        rla
716D 10FA      djnz 7169
716F 47        ld b,a
7170 0F        rrca
7171 B0        or b
7172 4E        ld c,(hl)
7173 2C        inc l
7174 77        ld (hl),a
7175 24        inc h
7176 77        ld (hl),a
7177 2D        dec l
7178 71        ld (hl),c
```

```
7179 CDF184    call 84F1
717C C1        pop bc
717D 10D4      djnz 7153
717F C9        ret
7180 3A3B76    ld a,(763B)
7183 07        rlca
7184 07        rlca
7185 D8        ret c
7186 07        rlca
7187 DC9369    call c,6993
718A 3AF676    ld a,(76F6)
718D F604      or 04
718F 32F676    ld (76F6),a
7192 ED4BF876  ld bc,(76F8)
7196 78        ld a,b
7197 E6F8      and F8
7199 47        ld b,a
719A 79        ld a,c
719B E6F8      and F8
719D 4F        ld c,a
719E ED43F876  ld (76F8),bc
71A2 CDE072    call 72E0
71A5 CD2E67    call 672E
71A8 3E10      ld a,10
71AA 322C77    ld (772C),a
71AD CDCF5B    call 5BCF
71B0 CDA770    call 70A7
71B3 FEFF      cp FF
71B5 CAD170    jp z,70D1
71B8 A7        and a
71B9 28ED      jr z,71A8
71BB CDD171    call 71D1
71BE 3A2C77    ld a,(772C)
71C1 D618      sub 18
71C3 3808      jr c,71CD
71C5 FE20      cp 20
71C7 30E1      jr nc,71AA
71C9 3E20      ld a,20
71CB 18DD      jr 71AA
71CD 3E50      ld a,50
71CF 18D9      jr 71AA
71D1 F5        push af
71D2 CDD970    call 70D9
71D5 F1        pop af
71D6 FE20      cp 20
71D8 3856      jr c,7230
71DA 5F        ld l,a
71DB 2600      ld h,00
71DD 3AF676    ld a,(76F6)
71E0 E602      and 02
71E2 ED5B3B5C  ld de,(5C3B)
71E6 2812      jr z,71FA
71E8 7D        ld a,l
71E9 FE20      cp 20
71EB 280D      jr z,71FA
71ED FE41      cp 41
71EF D8        ret c
71F0 FE56      cp 56
```



```
71F2 D0        ret nc
71F3 D641      sub 41
71F5 6F        ld l,a
71F6 ED5B7B5C  ld de,(5C7B)
71FA 29        add hl,hl
71FB 29        add hl,hl
71FC 29        add hl,hl
71FD 19        add hl,de
71FE E5        push hl
71FF ED4BF876  ld bc,(76F8)
7203 CD2D66    call 662D
7206 D1        pop de
7207 D5        push de
7208 CD8F68    call 688F
720B DDE1      pop ix
720D ED4BF876  ld bc,(76F8)
7211 79        ld a,c
7212 C608      add a,08
7214 4F        ld c,a
7215 78        ld a,b
7216 47        ld b,a
7217 C5        push bc
7218 CDFE70    call 70FE
721B C1        pop bc
721C ED43F876  ld (76F8),bc
7220 CD2D66    call 662D
7223 1100A0    ld de,A000
7226 010800    ld bc,0008
7229 7E        ld a,(hl)
722A 12        ld (de),a
722B 24        inc h
722C 13        inc de
722D 10FA      djnz 7229
722F C9        ret
7230 DD2100A0  ld ix,A000
7234 FE09      cp 09
7236 2008      jr nz,7243
7238 3AF676    ld a,(76F6)
723B EE02      xor 02
723D 32F676    ld (76F6),a
7240 C32E67    jp 672E
7243 FE05      cp 05
7245 D8        ret c
7246 ED4BF876  ld bc,(76F8)
724A 2005      jr nz,7251
724C 79        ld a,c
724D D608      sub 08
724F 18C3      jr 7214
7251 FE08      cp 08
7253 28BC      jr z,7211
7255 FE07      cp 07
7257 78        ld a,b
7258 2008      jr nz,7262
725A D608      sub 08
725C 30B8      jr nc,7216
725E D640      sub 40
7260 18B4      jr 7216
7262 C608      add a,08
```

```
7264 FEC0       cp C0
7266 38AE       jr c,7216
7268 D6C0       sub C0
726A 18AA       jr 7216
726C C5         push bc
726D 3A0877     ld a,(7708)
7270 90         sub b
7271 ED430777   ld (7707),bc
7275 320777     ld (7707),a
7278 CD8472     call 7284
727B C1         pop bc
727C 79         ld a,c
727D 48         ld c,b
727E 47         ld b,a
727F 91         sub c
7280 C8         ret z
7281 320777     ld (7707),a
7284 110000     ld de,0000
7287 C5         push bc
7288 CD9572     call 7295
728B C1         pop bc
728C 78         ld a,b
728D A7         and a
728E C8         ret z
728F 1100FF     ld de,FF00
7292 ED44       neg
7294 47         ld b,a
7295 3A2977     ld a,(7729)
7298 A7         and a
7299 202A       jr nz,72C5
729B C5         push bc
729C D5         push de
729D CDA972     call 72A9
72A0 D1         pop de
72A1 C1         pop bc
72A2 79         ld a,c
72A3 A7         and a
72A4 C8         ret z
72A5 ED44       neg
72A7 4F         ld c,a
72A8 1D         dec e
72A9 2AFA75     ld hl,(75FA)
72AC 7C         ld a,h
72AD 80         add a,b
72AE 47         ld b,a
72AF 9F         sbc a,a
72B0 AA         xor d
72B1 C0         ret nz
72B2 78         ld a,b
72B3 FEC0       cp C0
72B5 D0         ret nc
72B6 7D         ld a,l
72B7 81         add a,c
72B8 4F         ld c,a
72B9 9F         sbc a,a
72BA AB         xor e
72BB C0         ret nz
72BC C5         push bc
```



```
72BD CD0869     call 6908
72C0 C1         pop bc
72C1 CDE569     call 69E5
72C4 C9         ret
72C5 3A0777     ld a,(7707)
72C8 A7         and a
72C9 C8         ret z
72CA 79         ld a,c
72CB A7         and a
72CC 28DB       jr z,72A9
72CE C5         push bc
72CF D5         push de
72D0 CDA972     call 72A9
72D3 D1         pop de
72D4 C1         pop bc
72D5 C5         push bc
72D6 D5         push de
72D7 1D         dec e
72D8 79         ld a,c
72D9 ED44       neg
72DB 4F         ld c,a
72DC CDA972     call 72A9
72DF D1         pop de
72E0 C1         pop bc
72E1 0D         dec c
72E2 18E1       jr 72C5
72E4 3EFF       ld a,FF
72E6 1801       jr 72E9
72E8 AF         xor a
72E9 322977     ld (7729),a
72EC CD0870     call 7008
72EF CDF466     call 66F4
72F2 ED4BF875   ld bc,(75F8)
72F6 2AFA75     ld hl,(75FA)
72F9 78         ld a,b
72FA 94         sub h
72FB 3002       jr nc,72FF
72FD ED44       neg
72FF F5         push af
7300 79         ld a,c
7301 95         sub l
7302 3002       jr nc,7306
7304 ED44       neg
7306 CD6073     call 7360
7309 F1         pop af
730A E5         push hl
730B CD6073     call 7360
730E D1         pop de
730F 19         add hl,de
7310 CD7473     call 7374
7313 7C         ld a,h
7314 A7         and a
7315 2048       jr nz,735D
7317 7A         ld a,d
7318 B3         or e
7319 2801       jr z,731C
731B 2C         inc l
731C DD210077   ld ix,7700
```

```
7320 7D          ld a,l
7321 DD7700      ld (ix+00),a
7324 A7          and a
7325 2008        jr nz,732F
7327 010000      ld bc,0000
732A CDA972      call 72A9
732D 182E        jr 735D
732F 1185B5      ld de,00B5
7332 CD6373      call 7363
7335 7C          ld a,h
7336 DD7701      ld (ix+01),a
7339 DD7E01      ld a,(ix+01)
733C CD6073      call 7360
733F E5          push hl
7340 DD7E00      ld a,(ix+00)
7343 CD6073      call 7360
7346 C1          pop bc
7347 A7          and a
7348 ED42        sbc hl,bc
734A CD7473      call 7374
734D 45          ld b,l
734E DD4E01      ld c,(ix+01)
7351 CD6C72      call 726C
7354 DD7E01      ld a,(ix+01)
7357 DD3501      dec (ix+01)
735A A7          and a
735B 20DC        jr nz,7339
735D C30A67      jp 670A
7360 5F          ld e,a
7361 1600        ld d,00
7363 210000      ld hl,0000
7366 0608        ld b,08
7368 CB3F        srl a
736A 3001        jr nc,736D
736C 19          add hl,de
736D CB23        sla e
736F CB12        rl d
7371 10F5        djnz 7368
7373 C9          ret
7374 110100      ld de,0001
7377 A7          and a
7378 ED52        sbc hl,de
737A 3804        jr c,7380
737C 13          inc de
737D 13          inc de
737E 18F7        jr 7377
7380 19          add hl,de
7381 CB3A        srl d
7383 CB1B        rr e
7385 EB          ex de,hl
7386 C9          ret
7387 3A3A76      ld a,(763A)
738A 1802        jr 738E
738C 3EFF        ld a,FF
738E 6F          ld l,a
738F D9          exx
7390 E5          push hl
7391 CD0B70      call 700B
```



```
7394 CDF455      call 55F4
7397 2AFA76      ld hl,(76FA)
739A ED4BF876    ld bc,(76F8)
739E 110101      ld de,0101
73A1 78          ld a,b
73A2 94          sub h
73A3 2805        jr z,73AA
73A5 3004        jr nc,73AB
73A7 ED44        neg
73A9 15          dec d
73AA 15          dec d
73AB 47          ld b,a
73AC 79          ld a,c
73AD 95          sub l
73AE 2805        jr z,73B5
73B0 3004        jr nc,73B6
73B2 ED44        neg
73B4 1D          dec e
73B5 1D          dec e
73B6 4F          ld c,a
73B7 B0          or b
73B8 2842        jr z,73FC
73BA 79          ld a,c
73BB B8          cp b
73BC D5          push de
73BD 3805        jr c,73C5
73BF 1600        ld d,00
73C1 68          ld l,b
73C2 41          ld b,c
73C3 1803        jr 73C8
73C5 1E00        ld e,00
73C7 69          ld l,c
73C8 60          ld h,b
73C9 D9          exx
73CA ED4BFA76    ld bc,(76FA)
73CE D9          exx
73CF 78          ld a,b
73D0 CB3F        srl a
73D2 85          add a,l
73D3 3803        jr c,73D8
73D5 BC          cp h
73D6 3807        jr c,73DF
73D8 94          sub h
73D9 4F          ld c,a
73DA D9          exx
73DB D1          pop de
73DC D5          push de
73DD 1804        jr 73E3
73DF 4F          ld c,a
73E0 D5          push de
73E1 D9          exx
73E2 D1          pop de
73E3 7A          ld a,d
73E4 80          add a,b
73E5 47          ld b,a
73E6 79          ld a,c
73E7 83          add a,e
73E8 4F          ld c,a
```

```
73E9 CB0D       rrc l
73EB 300A       jr nc,73F7
73ED E5         push hl
73EE C5         push bc
73EF CD0869     call 6908
73F2 C1         pop bc
73F3 CDE569     call 69E5
73F6 E1         pop hl
73F7 D9         exx
73F8 79         ld a,c
73F9 10D7       djnz 73D2
73FB D1         pop de
73FC CD0A67     call 670A
73FF E1         pop hl
7400 D9         exx
7401 C9         ret
7402 00         nop
7403 40         ld b,b
7404 2A0274     ld hl,(7402)
7407 3A3B76     ld a,(763B)
740A E604       and 04
740C 2803       jr z,7411
740E 2A9765     ld hl,(6597)
7411 CDE764     call 64E7
7414 7C         ld a,h
7415 F6E0       or E0
7417 67         ld h,a
7418 7E         ld a,(hl)
7419 E620       and 20
741B 3E07       ld a,07
741D 2002       jr nz,7421
741F 3E38       ld a,38
7421 215375     ld hl,7553
7424 11009B     ld de,9B00
7427 011C00     ld bc,001C
742A EDB0       ldir
742C EB         ex de,hl
742D 111D9B     ld de,9B1D
7430 77         ld (hl),a
7431 011B00     ld bc,001B
7434 EDB0       ldir
7436 6F         ld l,a
7437 0F         rrca
7438 0F         rrca
7439 0F         rrca
743A AD         Xor l
743B E607       and 07
743D AD         Xor l
743E 6F         ld l,a
743F 67         ld h,a
7440 22359B     ld (9B35),hl
7443 22289B     ld (9B28),hl
7446 EE06       Xor 06
7448 32309B     ld (9B30),a
744B 21009B     ld hl,9B00
744E ED4BF876   ld bc,(76F8)
7452 C5         Push bc
7453 79         ld a,c
```



```
7454 0E03       ld c,03
7456 CD7F75     call 757F
7459 23         inc hl
745A C1         pop bc
745B 3EAF       ld a,AF
745D 90         sub b
745E 0E03       ld c,03
7460 3006       jr nc,7468
7462 ED44       neg
7464 362D       ld (hl),2D
7466 23         inc hl
7467 0D         dec c
7468 CD7F75     call 757F
746B 3AF676     ld a,(76F6)
746E E604       and 04
7470 2005       jr nz,7477
7472 3A3B76     ld a,(763B)
7475 E603       and 03
7477 07         rlca
7478 07         rlca
7479 EB         ex de,hl
747A 6F         ld l,a
747B 2600       ld h,00
747D 013F75     ld bc,753F
7480 09         add hl,bc
7481 010400     ld bc,0004
7484 EDB0       ldir
7486 110E9B     ld de,9B0E
7489 3AF676     ld a,(76F6)
748C E606       and 06
748E FE04       cp 04
7490 3008       jr nc,749A
7492 3A3B76     ld a,(763B)
7495 E640       and 40
7497 07         rlca
7498 07         rlca
7499 07         rlca
749A 07         rlca
749B 6F         ld l,a
749C 2600       ld h,00
749E 016F75     ld bc,756F
74A1 09         add hl,bc
74A2 010400     ld bc,0004
74A5 EDB0       ldir
74A7 3A3B76     ld a,(763B)
74AA 07         rlca
74AB 3808       jr c,74B8
74AD 2A289B     ld hl,(9B28)
74B0 7C         ld a,h
74B1 EE05       Xor 05
74B3 6F         ld l,a
74B4 67         ld h,a
74B5 22289B     ld (9B28),hl
74B8 3A3C76     ld a,(763C)
74BB 47         ld b,a
74BC E607       and 07
74BE C630       add a,30
7400 32179B     ld (9B17),a
```

```
74C3 E607       and 07
74C5 FE04       cp 04
74C7 17         rla
74C8 0F         rrca
74C9 07         rlca
74CA 17         rla
74CB 0F         rrca
74CC 07         rlca
74CD 17         rla
74CE 323398     ld (9833),a
74D1 78         ld a,b
74D2 E638       and 38
74D4 FE20       cp 20
74D6 3002       jr nc,74DA
74D8 F607       or 07
74DA 323198     ld (9831),a
74DD 0F         rrca
74DE 0F         rrca
74DF 0F         rrca
74E0 E607       and 07
74E2 C630       add a,30
74E4 321598     ld (9815),a
74E7 CB00       rlc b
74E9 3008       jr nc,74F3
74EB 3A3698     ld a,(9836)
74EE EE87       xor 87
74F0 323698     ld (9836),a
74F3 CB00       rlc b
74F5 3008       jr nc,74FF
74F7 3A3598     ld a,(9835)
74FA EE47       xor 47
74FC 323598     ld (9835),a
74FF 3A3B76     ld a,(763B)
7502 2E32       ld l,32
7504 E60C       and 0C
7506 FE08       cp 08
7508 3802       jr c,750C
750A 2E34       ld l,34
750C E604       and 04
750E 2002       jr nz,7512
7510 2E31       ld l,31
7512 7D         ld a,l
7513 321498     ld (9814),a
7516 3AF676     ld a,(76F6)
7519 E610       and 10
751B C0         ret nz
751C 2A0274     ld hl,(7402)
751F DD2100DB   ld ix,DB00
7523 010407     ld bc,0704
7526 E5         push hl
7527 C5         push bc
7528 DD7E1C     ld a,(ix+1C)
752B 08         ex af,af'
752C DD7E00     ld a,(ix+00)
752F CD7268     call 6872
7532 DD23       inc ix
7534 10F2       djnz 7528
7536 C1         pop bc
```



```
7537 E1         pop hl
7538 CD5965     call 6559
753B 0D         dec c
753C 20E8       jr nz,7526
753E C9         ret
753F 53         ld d,e
7540 45         ld b,l
7541 54         ld d,h
7542 2058       jr nz,759C
7544 4F         ld c,a
7545 52         ld d,d
7546 2052       jr nz,759A
7548 45         ld b,l
7549 53         ld d,e
754A 2053       jr nz,759F
754C 41         ld b,c
754D 4C         ld c,h
754E 54         ld d,h
754F 54         ld d,h
7550 45         ld b,l
7551 50         ld e,b
7552 54         ld d,h
7553 2020       jr nz,7575
7555 20CC       jr nz,7523
7557 2020       jr nz,7579
7559 2020       jr nz,757B
755B 2020       jr nz,757D
755D 2020       jr nz,757F
755F 53         ld d,e
7560 41         ld b,c
7561 2020       jr nz,7583
7563 20CD       jr nz,7585
7565 204D       jr nz,75B4
7567 2020       jr nz,7589
7569 50         ld d,b
756A 2049       jr nz,75B5
756C 42         ld b,d
756D 46         ld b,(hl)
756E 2054       jr nz,75C4
7570 45         ld b,l
7571 4C         ld c,h
7572 41         ld b,c
7573 41         ld b,c
7574 54         ld d,h
7575 52         ld d,d
7576 49         ld c,c
7577 2020       jr nz,7599
7579 2020       jr nz,759B
757B 47         ld b,a
757C 52         ld d,d
757D 41         ld b,c
757E 46         ld b,(hl)
757F E5         push hl
7580 0600       ld b,00
7582 219575     ld hl,7595
7585 09         add hl,bc
7586 5E         ld e,(hl)
7587 E1         pop hl
```

```
7588 06 2F       ld b,2F
758A 93          sub e
758B 04          inc b
758C 30 FC       jr nc,758A
758E 83          add a,e
758F 70          ld (hl),b
7590 23          inc hl
7591 0D          dec c
7592 20 EB       jr nz,757F
7594 00          nop
7595 00          nop
7596 01 0A 64    ld bc,640A
7599 2A 02 74    ld hl,(7402)
759C 7C          ld a,h
759D FE 40       cp 40
759F 20 05       jr nz,75A6
75A1 11 80 50    ld hl,5080
75A4 18 03       jr 75A9
75A6 11 00 40    ld hl,4000
75A9 22 02 74    ld (7402),hl
75AC 21 80 5D    ld hl,5D80
75AF 18 76       jr 7627
75B1 7E          ld a,(hl)
75B2 10 FD       djnz 75B1
75B4 32 F6 75    ld (75F6),a
75B7 18 F0       jr 75A9
75B9 0D          dec c
75BA 79          ld a,c
75BB 94          sub h
75BC 20 40       jr nz,7610
75BE 4D          ld c,l
75BF 7E          ld a,(hl)
75C0 23          inc hl
75C1 46          ld b,(hl)
75C2 23          inc hl
75C3 5E          ld e,(hl)
75C4 23          inc hl
75C5 56          ld d,(hl)
75C6 23          inc hl
75C7 EB          ex de,hl
75C8 71          ld (hl),c
75C9 23          inc hl
75CA 70          ld (hl),b
75CB 23          inc hl
75CC 73          ld (hl),e
75CD 23          inc hl
75CE 72          ld (hl),d
75CF 23          inc hl
75D0 EB          ex de,hl
75D1 79          ld a,c
75D2 94          sub h
75D3 20 53       jr nz,7628
75D5 45          ld b,l
75D6 7C          ld a,h
75D7 93          sub e
75D8 28 02       jr z,75DC
75DA 18 F2       jr 75CE
75DC 3A 02 74    ld a,(7402)
75DF 3C          inc a
75E0 32 02 74    ld (7402),a
75E3 FE 28       cp 28
75E5 20 E8       jr nz,75CF
75E7 21 00 58    ld hl,5800
75EA 22 02 74    ld (7402),hl
75ED 18 E0       jr 75CF
75EF 06 2F       ld b,2F
75F1 93          sub e
75F2 04          inc b
75F3 30 FC       jr nc,75F1
75F5 83          add a,e
75F6 70          ld (hl),b
75F7 23          inc hl
75F8 0D          dec c
75F9 20 EB       jr nz,75E6
75FB 00          nop
75FC 00          nop
75FD 01 0A 64    ld bc,640A
7600 2A 02 74    ld hl,(7402)
7603 7C          ld a,h
7604 FE 40       cp 40
7606 20 05       jr nz,760D
7608 11 80 50    ld hl,5080
760B 18 03       jr 7610
760D 11 00 40    ld hl,4000
7610 22 02 74    ld (7402),hl
7613 21 80 5D    ld hl,5D80
7616 18 76       jr 768E
7618 7E          ld a,(hl)
7619 10 FD       djnz 7618
761B 32 F6 75    ld (75F6),a
761E 18 F0       jr 7610
7620 0D          dec c
7621 79          ld a,c
7622 94          sub h
7623 20 40       jr nz,7669
7625 4D          ld c,l
7626 7E          ld a,(hl)
7627 23          inc hl
7628 46          ld b,(hl)
7629 23          inc hl
762A 5E          ld e,(hl)
762B 23          inc hl
762C 56          ld d,(hl)
762D 23          inc hl
762E EB          ex de,hl
762F 71          ld (hl),c
7630 23          inc hl
7631 70          ld (hl),b
7632 23          inc hl
7633 73          ld (hl),e
7634 23          inc hl
7635 72          ld (hl),d
7636 23          inc hl
7637 EB          ex de,hl
7638 79          ld a,c
7639 94          sub h
763A 20 53       jr nz,768F
763C 45          ld b,l
763D 7C          ld a,h
763E 93          sub e
763F 28 02       jr z,7643
7641 18 F2       jr 7635
7643 3A 02 74    ld a,(7402)
7646 3C          inc a
7647 32 02 74    ld (7402),a
764A FE 28       cp 28
764C 20 E8       jr nz,7636
764E 21 00 58    ld hl,5800
7651 22 02 74    ld (7402),hl
7654 18 E0       jr 7636
7656 06 2F       ld b,2F
7658 93          sub e
7659 04          inc b
765A 30 FC       jr nc,7658
765C 83          add a,e
765D 70          ld (hl),b
765E 23          inc hl
765F 0D          dec c
7660 20 EB       jr nz,764D
7662 00          nop
7663 00          nop
7664 01 0A 64    ld bc,640A
7667 2A 02 74    ld hl,(7402)
766A 7C          ld a,h
766B FE 40       cp 40
766D 20 05       jr nz,7674
766F 11 80 50    ld hl,5080
7672 18 03       jr 7677
7674 11 00 40    ld hl,4000
7677 22 02 74    ld (7402),hl
767A 21 80 5D    ld hl,5D80
767D 18 76       jr 76F5
767F 7E          ld a,(hl)
7680 10 FD       djnz 767F
7682 32 F6 75    ld (75F6),a
7685 18 F0       jr 7677
7687 0D          dec c
7688 79          ld a,c
7689 94          sub h
768A 20 40       jr nz,76CC
768C 4D          ld c,l
768D 7E          ld a,(hl)
768E 23          inc hl
768F 46          ld b,(hl)
7690 23          inc hl
7691 5E          ld e,(hl)
7692 23          inc hl
7693 56          ld d,(hl)
7694 23          inc hl
7695 EB          ex de,hl
7696 71          ld (hl),c
7697 23          inc hl
7698 70          ld (hl),b
7699 23          inc hl
769A 73          ld (hl),e
769B 23          inc hl
769C 72          ld (hl),d
769D 23          inc hl
769E EB          ex de,hl
769F 79          ld a,c
76A0 94          sub h
76A1 20 53       jr nz,76F6
76A3 45          ld b,l
76A4 7C          ld a,h
76A5 93          sub e
76A6 28 02       jr z,76AA
76A8 18 F2       jr 769C
76AA 3A 02 74    ld a,(7402)
76AD 3C          inc a
76AE 32 02 74    ld (7402),a
76B1 FE 28       cp 28
76B3 20 E8       jr nz,769D
76B5 21 00 58    ld hl,5800
76B8 22 02 74    ld (7402),hl
76BB 18 E0       jr 769D
```



```
75E6 DD 21 BC 75  ld ix,75BC
75EA CD 31 68     call 6831
75ED 21 C0 40     ld hl,40C0
75F0 E5           push hl
75F1 CD 65 68     call 6865
75F4 3E 87        ld a,87
75F6 32 C0 58     ld (58C0),a
75F9 E1           pop hl
75FA 2C           inc l
75FB 06 08        ld b,08
75FD C5           push bc
75FE E5           push hl
75FF CD 2E 67     call 672E
7602 CD 74 67     call 6774
7605 FE 02        cp 02
7607 30 F6        jr nc,75FF
7609 6F           ld l,a
760A 07           rlca
760B 07           rlca
760C F6 43        or 43
760E 08           ex af,af'
760F 3E 30        ld a,30
7611 85           add a,l
7612 E1           pop hl
7613 CD 72 68     call 6872
7616 E5           push hl
7617 CD E7 64     call 64E7
761A 36 87        ld (hl),87
761C E1           pop hl
761D C1           pop bc
761E 10 DD        djnz 75FD
7620 06 08        ld b,08
7622 21 C0 58     ld hl,58C0
7625 7E           ld a,(hl)
7626 23           inc hl
7627 E6 04        and 04
7629 0F           rrca
762A 0F           rrca
762B 0F           rrca
762C CB 11        rl c
762E 10 F5        djnz 7625
7630 79           ld a,c
7631 32 3A 76     ld (763A),a
7634 C3 0D 5B     jp 5B0D
7637 03           inc bc
7638 08           ex af,af'
7639 28 AA        jr z,75E5
763B 00           nop
763C 07           rlca
763D FF           rst 38
763E 81           add a,c
763F 81           add a,c
7640 99           sbc a,c
7641 99           sbc a,c
7642 81           add a,c
7643 81           add a,c
7644 FF           rst 38
7645 81           add a,c
```

```
7646 7E          ld a,(hl)
7647 56          ld h,(hl)
7648 5A          ld e,d
7649 5A          ld e,d
764A 66          ld h,(hl)
764B 7E          ld a,(hl)
764C 81          add a,c
764D 55          ld d,l
764E AA          xor d
764F 55          ld d,l
7650 AA          xor d
7651 55          ld d,l
7652 AA          xor d
7653 55          ld d,l
7654 AA          xor d
7655 AA          xor d
7656 00          nop
7657 AA          xor d
7658 00          nop
7659 AA          xor d
765A 00          nop
765B AA          xor d
765C 00          nop
765D FF          rst 38
765E 00          nop
765F FF          rst 38
7660 00          nop
7661 FF          rst 38
7662 00          nop
7663 FF          rst 38
7664 00          nop
7665 88          adc a,b
7666 112244      ld de,4422
7669 88          adc a,b
766A 1020        djnz 768C
766C 40          ld b,b
766D 219048      ld hl,4890
7670 24          inc h
7671 16          ld (de),a
7672 09          add hl,bc
7673 84          add a,h
7674 42          ld b,d
7675 AA          xor d
7676 AA          xor d
7677 44          ld b,h
7678 44          ld b,h
7679 AA          xor d
767A AA          xor d
767B 1111EE      ld de,EE11
767E 223399      ld (9933),hl
7681 EEEE        xor EE
7683 33          inc sp
7684 99          sbc a,c
7685 80          add a,b
7686 80          add a,b
7687 80          add a,b
7688 FF          rst 38
7689 08          ex af,af'
```



```
768A 08          ex af,af'
768B 08          ex af,af'
768C FF          rst 38
768D 88          adc a,b
768E 44          ld b,h
768F 222222      ld (2222),hl
7692 44          ld b,h
7693 88          adc a,b
7694 88          adc a,b
7695 225088      ld (8850),hl
7698 05          dec b
7699 225088      ld (8850),hl
769C 05          dec b
769D 08          ex af,af'
769E 04          inc b
769F 7C          ld a,h
76A0 A4          and h
76A1 25          dec h
76A2 3E20        ld a,20
76A4 1043        djnz 76E9
76A6 A5          and l
76A7 53          ld e,d
76A8 34          inc (hl)
76A9 AC          inc l
76AA 53          ld e,d
76AB A5          and l
76AC CC25555     jp nz,5555
76AF 55          ld d,l
76B0 55          ld d,l
76B1 55          ld d,l
76B2 55          ld d,l
76B3 55          ld d,l
76B4 55          ld d,l
76B5 CCCCC33     call z,3300
76B8 33          inc sp
76B9 CCCCC33     call z,3300
76BC 33          inc sp
76BD 3870        jr c,772F
76BF E0          ret po
76C0 C1          pop bc
76C1 83          add a,e
76C2 07          rlca
76C3 10E10       ld c,1C
76C5 22FA8A      ld (8AFA),hl
76C8 AA          xor d
76C9 AA          xor d
76CA AA          xor d
76CB A8          xor b
76CC AA          xor d
76CD E5          push hl
76CE 07          rlca
76CF 38AB        jr c,767C
76D1 BA          cp d
76D2 83          add a,e
76D3 E0          ret po
76D4 A7          and a
76D5 88          adc a,b
76D6 55          ld d,l
```

```
76D7 221144     ld (4411),hl
76DA 225588     ld (8855),hl
76DD 08         ex af,af'
76DE 1C         inc e
76DF 3E7F       ld a,7F
76E1 FF         rst 38
76E2 7F         ld a,a
76E3 3E1C       ld a,1C
76E5 A0         and b
76E6 BE         cp (hl)
76E7 82         add a,d
76E8 FA0AEB     jp m,EB0A
76EB 28AF       jr z,769C
76ED 00         nop
76EE 010810     ld bc,1008
76F1 1820       jr 7713
76F3 3040       jr nc,7735
76F5 50         ld d,b
76F6 00         nop
76F7 00         nop
76F8 80         add a,b
76F9 60         ld h,b
```

APÊNDICE A

CONVERSÕES DE VALORES DECIMAIS, BINÁRIOS E HEXADECIMAIS

Hexadecimal	*Decimal*	*Binário*
00	0	00000000
01	1	00000001
02	2	00000010
03	3	00000011
04	4	00000100
05	5	00000101
06	6	00000110
07	7	00000111
08	8	00001000
09	9	00001001
0A	10	00001010
0B	11	00001011
0C	12	00001100
0D	13	00001101
0E	14	00001110
0F	15	00001111
10	16	00010000
11	17	00010001
12	18	00010010
13	19	00010011
14	20	00010100
15	21	00010101
16	22	00010110

Hexadecimal	Decimal	Binário
17	23	00010111
18	24	00011000
19	25	00011001
1A	26	00011010
1B	27	00011011
1C	28	00011100
1D	29	00011101
1E	30	00011110
1F	31	00011111
20	32	00100000
21	33	00100001
22	34	00100010
23	35	00100011
24	36	00100100
25	37	00100101
26	38	00100110
27	39	00100111
28	40	00101000
29	41	00101001
2A	42	00101010
2B	43	00101011
2C	44	00101100
2D	45	00101101
2E	46	00101110
2F	47	00101111
30	48	00110000
31	49	00110001
32	50	00110010
33	51	00110011
34	52	00110100
35	53	00110101
36	54	00110110
37	55	00110111
38	56	00111000
39	57	00111001
3A	58	00111010
3B	59	00111011
3C	60	00111100
3D	61	00111101
3E	62	00111110
3F	63	00111111

Hexadecimal	Decimal	Binário
40	64	01000000
41	65	01000001
42	66	01000010
43	67	01000011
44	68	01000100
45	69	01000101
46	70	01000110
47	71	01000111
48	72	01001000
49	73	01001001
4A	74	01001010
4B	75	01001011
4C	76	01001100
4D	77	01001101
4E	78	01001110
4F	79	01001111
50	80	01010000
51	81	01010001
52	82	01010010
53	83	01010011
54	84	01010100
55	85	01010101
56	86	01010110
57	87	01010111
58	88	01011000
59	89	01011001
5A	90	01011010
5B	91	01011011
5C	92	01011100
5D	93	01011101
5E	94	01011110
5F	95	01011111
60	96	01100000
61	97	01100001
62	98	01100010
63	99	01100011
64	100	01100100
65	101	01100101
66	102	01100110
67	103	01100111
68	104	01101000

Hexadecimal	Decimal	Binário
69	105	01101001
6A	106	01101010
6B	107	01101011
6C	108	01101100
6D	109	01101101
6E	110	01101110
6F	111	01101111
70	112	01110000
71	113	01110001
72	114	01110010
73	115	01110011
74	116	01110100
75	117	01110101
76	118	01110110
77	119	01110111
78	120	01111000
79	121	01111001
7A	122	01111010
7B	123	01111011
7C	124	01111100
7D	125	01111101
7E	126	01111110
7F	127	01111111
80	128	10000000
81	129	10000001
82	130	10000010
83	131	10000011
84	132	10000100
85	133	10000101
86	134	10000110
87	135	10000111
88	136	10001000
89	137	10001001
8A	138	10001010
8B	139	10001011
8C	140	10001100
8D	141	10001101
8E	142	10001110
8F	143	10001111
90	144	10010000
91	145	10010001

Hexadecimal	Decimal	Binário
92	146	10010010
93	147	10010011
94	148	10010100
95	149	10010101
96	150	10010110
97	151	10010111
98	152	10011000
99	153	10011001
9A	154	10011010
9B	155	10011011
9C	156	10011100
9D	157	10011101
9E	158	10011110
9F	159	10011111
A0	160	10100000
A1	161	10100001
A2	162	10100010
A3	163	10100011
A4	164	10100100
A5	165	10100101
A6	166	10100110
A7	167	10100111
A8	168	10101000
A9	169	10101000
AA	170	10101010
AB	171	10101011
AC	172	10101100
AD	173	10101101
AE	174	10101110
AF	175	10101111
B0	176	10110000
B1	177	10110001
B2	178	10110010
B3	179	10110011
B4	180	10110100
B5	181	10110101
B6	182	10110110
B7	183	10110111
B8	184	10111000
B9	185	10111001
BA	186	10111010

Hexadecimal	*Decimal*	*Binário*
BB	187	10111011
BC	188	10111100
BD	189	10111101
BE	190	10111110
BF	191	10111111
C0	192	11000000
C1	193	11000001
C2	194	11000010
C3	195	11000011
C4	196	11000100
C5	197	11000101
C6	198	11000110
C7	199	11000111
C8	200	11001000
C9	201	11001001
CA	202	11001010
CB	203	11001011
CC	204	11001100
CD	205	11001101
CE	206	11001110
CF	207	11001111
D0	208	11010000
D1	209	11010001
D2	210	11010010
D3	211	11010011
D4	212	11010100
D5	213	11010101
D6	214	11010110
D7	215	11010111
D8	216	11011000
D9	217	11011001
DA	218	11011010
DB	219	11011011
DC	220	11011100
DD	221	11011101
DE	222	11011110
DF	223	11011111
E0	224	11100000
E1	225	11100001
E2	226	11100010
E3	227	11100011

E4	228	11100100
E5	229	11100101
E6	230	11100110
E7	231	11100111
E8	232	11101000
E9	233	11101001
EA	234	11101010
EB	235	11101011
EC	236	11101100
ED	237	11101101
EE	238	11101110
EF	239	11101111
F0	240	11110000
F1	241	11110001
F2	242	11110010
F3	243	11110011
F4	244	11110100
F5	245	11110101
F6	246	11110110
F7	247	11110111
F8	248	11111000
F9	249	11111001
FA	250	11111010
FB	251	11111011
FC	252	11111100
FD	253	11111101
FE	254	11111110
FF	255	11111111

APÊNDICE B

CÓDIGOS DE OPERAÇÃO DO Z80 ORDENADOS POR MNEMÔNICAS

Código de Operação	*Códigos Hexadecimais*	*Códigos Decimais*
ADC A, (HL)	8E	142
ADC A, (IX + d)	DD8Edd	221, 142, xx
ADC A, (IY + d)	FD8Edd	253, 142, xx
ADC A, A	8F	143
ADC A, B	88	136
ADC A, C	89	137
ADC A, D	8A	138
ADC A, E	8B	139
ADC A, H	8C	140
ADC A, L	8D	141
ADC A, xx	CExx	206, xx
ADC HL, BC	ED4A	237, 74
ADC HL, DE	ED5A	237, 90
ADC HL, HL	ED6A	237, 106
ADC HL, SP	ED7A	237, 122
ADD A, (HL)	86	134
ADD A, (IX + d)	DD86dd	221, 134, xx
ADD A, (IY + d)	FD86dd	235, 134, xx
ADD A, A	87	135
ADD A, B	80	128
ADD A, C	81	129
ADD A, D	82	130

Código de Operação	*Códigos Hexadecimais*	*Códigos Decimais*
ADD A, E	83	131
ADD A, H	84	132
ADD A, L	85	133
ADD A, xx	C6xx	198, xx
ADD HL, BC	09	9
ADD HL, DE	19	25
ADD HL, HL	29	41
ADD HL, SP	39	57
ADD IX, BC	DD09	221, 9
ADD IX, DE	DD19	221, 25
ADD IX, IX	DD29	221, 41
ADD IX, SP	DD39	221, 57
ADD IY, BC	FD09	253, 9
ADD IY, DE	FD19	253, 25
ADD IY, IY	FD29	253, 41
ADD IY, SP	FD39	253, 57
AND (HL)	A6	166
AND (IX + d)	DDA6dd	221, 166, xx
AND (IY + d)	FDA6dd	253, 166, xx
AND A	A7	167
AND B	A0	160
AND C	A1	161
AND D	A2	162
AND E	A3	163
AND H	A4	164
AND L	A5	165
AND xx	E6xx	230, xx
BIT 0, (HL)	CB46	203, 70
BIT 0, (IX + d)	DDCBdd46	221, 203, xx, 70
BIT 0, (IY + d)	FDCBdd46	253, 203, xx, 70
BIT 0, A	CB47	203, 71
BIT 0, B	CB40	203, 64
BIT 0, C	CB41	203, 65
BIT 0, D	CB42	203, 66
BIT 0, E	CB43	203, 67
BIT 0, H	CB44	203, 68
BIT 0, L	CB45	203, 69

Código de Operação	Códigos Hexadecimais	Códigos Decimais
BIT 1, (HL)	CB4E	203, 70
BIT 1, (IX + d)	DDCBdd4E	221, 203, xx, 70
BIT 1, (IY + d)	FDCBdd4E	253, 203, xx, 70
BIT 1, A	CB4F	203, 79
BIT 1, B	CB48	203, 72
BIT 1, C	CB49	203, 73
BIT 1, D	CB4A	203, 74
BIT 1, E	CB4B	203, 75
BIT 1, H	CB4C	203, 76
BIT 1, L	CB4D	203, 77
BIT 2, (HL)	CB56	203, 86
BIT 2, (IX + d)	DDCBdd56	221, 203, xx, 86
BIT 2, (IY + d)	FDCBdd56	253, 203, xx, 86
BIT 2, A	CB57	203, 87
BIT 2, B	CB50	203, 80
BIT 2, C	CB51	203, 81
BIT 2, D	CB52	203, 82
BIT 2, E	CB53	203, 83
BIT 2, H	CB54	203, 84
BIT 2, L	CB55	203, 85
BIT 3, (HL)	CB5E	203, 94
BIT 3, (IX + d)	DDCBdd5E	221, 203, xx, 94
BIT 3, (IY + d)	FDCBdd5E	253, 203, xx, 94
BIT 3, A	CB5F	203, 95
BIT 3, B	CB58	203, 88
BIT 3, C	CB59	203, 89
BIT 3, D	CB5A	203, 90
BIT 3, E	CB5B	203, 91
BIT 3, H	CB5C	203, 92
BIT 3, L	CB5D	203, 93
BIT 4, (HL)	CB66	203, 102
BIT 4, (IX + d)	DDCBdd66	221, 203, xx, 102
BIT 4, (IY + d)	FDCBdd66	253, 203, xx, 102
BIT 4, A	CB67	203, 103
BIT 4, B	CB60	203, 96
BIT 4, C	CB61	203, 97
BIT 4, D	CB62	203, 98
BIT 4, E	CB63	203, 99

Código de Operação	Códigos Hexadecimais	Códigos Decimais
BIT 4, H	CB64	203, 100
BIT 4, L	CB65	202, 101
BIT 5, (HL)	CB6E	203, 110
BIT 5, (IX + d)	DDCBdd6E	221, 203, xx, 110
BIT 5, (IY + d)	FDCBdd6E	253, 203, xx, 110
BIT 5, A	CB6F	203, 111
BIT 5, B	CB68	203, 104
BIT 5, C	CB69	203, 105
BIT 5, D	CB6A	203, 106
BIT 5, E	CB6B	203, 107
BIT 5, H	CB6C	203, 108
BIT 5, L	CB6D	203, 109
BIT 6, (HL)	CB76	203, 118
BIT 6, (IX + d)	DDCBdd76	221, 203, xx, 118
BIT 6, (IY + d)	FDCBdd76	253, 203, xx, 118
BIT 6, A	CB77	203, 119
BIT 6, B	CB70	203, 112
BIT 6, C	CB71	203, 113
BIT 6, D	CB72	203, 114
BIT 6, E	CB73	203, 115
BIT 6, H	CB74	203, 116
BIT 6, L	CB75	203, 117
BIT 7, (HL)	CB7E	203, 116
BIT 7, (IX+ d)	DDCBdd7E	221, 203, xx, 116
BIT 7, (IY + d)	FDCBdd7E	253, 203, xx, 116
BIT 7, A	CB7F	203, 127
BIT 7, B	CB78	203, 120
BIT 7, C	CB79	203, 121
BIT 7, D	CB7A	203, 122
BIT 7, E	CB7B	203, 123
BIT 7, H	CB7C	203, 124
BIT 7, L	CB7D	203, 125
CALL C, xxxx	DCxxxx	220, xx, xx
CALL M, xxxx	FCxxxx	252, xx, xx
CALL NC, xxxx	D4xxxx	212, xx, xx
CALL NZ, xxxx	C4xxxx	196, xx, xx
CALL P, xxxx	F4xxxx	244, xx, xx

Código de Operação	Códigos Hexadecimais	Códigos Decimais
CALL PE, xxxx	ECxxxx	236, xx, xx
CALL PO, xxxx	E4xxxx	228, xx, xx
CALL Z, xxxx	CCxxxx	204, xx, xx
CP (HL)	BE	190
CP (IX + d)	DDBEdd	221, 190, xx
CP (IY + d)	FDBEdd	253, 190, xx
CP A	BF	191
CP B	B8	184
CP C	B9	185
CP D	BA	186
CP E	BB	187
CP H	BC	188
CP L	BD	189
CP xx	FExx	254, xx
CPD	EDA9	237, 169
CPDR	EDB9	237, 185
CPI	EDA1	237, 161
CPIR	EDB1	237, 177
CPL	2F	47
DAA	27	39
DEC (HL)	35	53
DEC (IX + d)	DD35dd	221, 53, xx
DEC (IY + d)	FD35dd	253, 53, xx
DEC A	3D	61
DEC B	05	5
DEC BC	0b	11
DEC C	0D	13
DEC D	15	21
DEC DE	1B	27
DEC E	1D	29
DEC H	25	37
DEC HL	2B	43
DEC IX	DD2B	221, 43
DEC IY	FD2B	253, 43
DEC L	2D	45
DEC SP	3B	59
DI	F3	243

Código de Operação	Códigos Hexadecimais	Códigos Decimais
DJNZ xx	10xx	16, xx
EI	FB	251
EX (SP), HL	E3	227
EX (SP), IX	DDE3	221, 227
EX (SP), IY	FDE3	253, 227
EX AF, AF′	08	8
EX DE, HL	EB	235
EXX	D9	217
HALT	76	118
IM 0	ED46	237, 70
IM 1	ED56	237, 86
IM 2	ED5E	237, 94
IN A, (C)	ED 78	237, 120
IN A, (xx)	DBxx	219, xx
IN B, (C)	ED40	237, 64
IN C, (C)	ED48	237, 72
IN D, (C)	ED50	237, 80
IN E, (C)	ED58	237, 88
IN H, (C)	ED60	237, 96
IN L, (C)	ED68	237, 104
INC (HL)	34	52
INC (IX + d)	DD34dd	221, 52, xx
INC (IY + d)	FD34dd	253, 52, xx
INC A	3C	60
INC B	04	4
INC BC	03	3
INC C	0C	12
INC D	14	20
INC DE	13	19
INC E	1C	28
INC H	24	36
INC HL	23	35
INC IX	DD23	221, 35
INC IY	FD23	253, 35
INC L	2C	44

Código de Operação	Códigos Hexadecimais	Códigos Decimais
INC SP	33	51
IND	EDAA	237, 170
INDR	EDBA	237, 186
INI	EDA2	237, 162
INIR	EDB2	237, 178
JP (HL)	E9	233
JP (IX)	DDE9	221, 233
JP (IY)	FDE9	253, 233
JP C, xxxx	DAxxxx	218, xx, xx
JP M, xxxx	FAxxxx	250, xx, xx
JP NC, xxxx	D2xxxx	210, xx, xx
JP NZ, xxxx	C2xxxx	194, xx, xx
JP P, xxxx	F2xxxx	242, xx, xx
JP PE, xxxx	EAxxxx	234, xx, xx
JP PO, xxxx	E2xxxx	226, xx, xx
JP xxxx	C3xxxx	195, xx, xx
JP Z, xxxx	CAxxxx	202, xx, xx
JR C, xx	38xx	56, xx
JR NC, xx	30xx	48, xx
JR NZ, xx	20xx	32, xx
JR xx	18xx	24, xx
JR Z, xx	28xx	40, xx
LD (BC), A	02	2
LD (DE), A	12	18
LD HL, (xxxx)	2Axxxx	42, xx, xx
LD (HL), A	77	119
LD (HL), B	70	112
LD (HL), C	71	113
LD (HL), D	72	114
LD (HL), E	73	115
LD (HL), H	74	116
LD (HL), L	75	117
LD (HL), xx	36xx	54, xx
LD (IX + d), A	DD77dd	221, 119 xx
LD (IX + d), B	DD70dd	221, 112, xx
LD (IX + d), C	DD71dd	221, 113, xx
LD (IX + d), D	DD72dd	221, 114, xx
LD (IX + d), E	DD73dd	221, 115, xx
LD (IX + d), H	DD74dd	221, 116, xx

Código de Operação	Códigos Hexadecimais	Códigos Decimais
LD (IX + d), L	DD75dd	221, 117, xx
LD (IX + d), xx	DD36ddxx	221, 54, xx, xx
LD (IY + d), A	FD77dd	253, 119, xx
LD (IY + d), B	FD70dd	253, 112, xx
LD (IY + d), C	FD71dd	253, 113, xx
LD (IY + d), D	FD72dd	253, 114, xx
LD (IY + d), E	FD73dd	253, 115, xx
LD (IY + d), H	FD74dd	253, 116, xx
LD (IY + d), L	FD75dd	253, 117, xx
LD (IY + d), xx	FD36ddxx	253, 54, xx, xx
LD (xxxx), A	32xxxx	50, xx, xx
LD (xxxx), BC	ED43xxxx	237, 67, xx, xx
LD (xxxx), DE	ED53xxxx	237, 83, xx, xx
LD (xxxx), HL	22xxxx	34, xx, xx
LD (xxxx), IX	DD22xxxx	221, 34, xx, xx
LD (xxxx), IY	Fd22xxxx	253, 34, xx, xx
LD (xxxx), SP	ED73xxxx	237, 115, xx, xx
LD A, (BC)	0A	10
LD A, (de)	1A	26
LD A, (HL)	7E	126
LD A, (IX + d)	DD7Edd	221, 126, xx
LD A, (IY + d)	FD7Edd	253, 126, xx
LD A, (xxxx)	3Axxxx	58, xx, xx
LD A, A	7F	127
LD A, B	78	120
LD A, C	79	121
LD A, D	7A	122
LD A, E	7B	123
LD A, H	7C	124
LD A, I	ED57	237, 87
LD A, L	7D	125
LD A, R	ED5F	237, 85
LD A, xx	3Exx	62, xx
LD B, (HL)	46	70
LD B, (IX + d)	DD46dd	221, 70, xx
LD B, (IY + d)	FD46dd	253, 70, xx
LD B, A	47	71
LD B, B	40	64

Código de Operação	Códigos Hexadecimais	Códigos Decimais
LD B, C	41	65
LD B, D	42	66
LD B, E	43	67
LD B, H	44	68
LD B, L	45	69
LD B, xx	06xx	6, xx
LD BC, (xxxx)	ED4Bxxxx	237, 75, xx, xx
LD BC, xxxx	01xxxx	1, xx, xx
LD C, (HL)	4E	78
LD C, (IX + d)	DD4Edd	221, 78, xx
LD C, (IY + d)	FD4Edd	253, 78, xx
LD C, A	4F	79
LD C, B	48	72
LD C, C	49	73
LD C, D	4A	74
LD C, E	4B	75
LD C, H	4C	76
LD C, L	4D	77
LD C, xx	0Exx	14, xx
LD D, (HL)	56	86
LD D, (IX + d)	DD56dd	221, 86, xx
LD D, (IY + d)	FD56dd	253, 86, xx
LD D, A	57	87
LD D, B	50	80
LD D, C	51	81
LD D, D	52	82
LD D, E	53	83
LD D, H	54	84
LD D, L	55	85
LD D, xx	16xx	22, xx
LD DE, (xxxx)	ED5Bxxxx	237, 91, xx, xx
LD DE, xxxx	11xxxx	17, xx, xx
LD E, (HL)	5E	94
LD E, (IX + d)	DD5Edd	221, 94, xx
LD E, (IY + d)	FD5Edd	253, 94, xx
LD E, A	5F	95
LD E, B	58	88
LD E, C	59	89
LD E, D	5A	90
LD E, E	5B	91
LD E, H	5C	92
LD E, L	5D	93
LD E, xx	1Exx	30, xx
LD H, (HL)	66	102
LD H, (IX + d)	DD66dd	221, 102, xx
LD H, (IY + d)	FD66dd	253, 102, xx
LD H, A	67	103
LD H, B	60	96
LD H, B	61	97
LD H, C	62	98
LD H, D	63	99
LD H, E	64	100
LD H, L	65	101
LD H, xx	26xx	38, xx
LD HL, xxxx	21xxxx	33, xx, xx
LD I, A	ED47	237, 71
LD IX, xxxx	DD21xxxx	221, 33, xx, xx
LD IX (xxxx)	DD2Axxxx	221, 42, xx, xx
LD IY, xxxx	FD21xxxx	253, 33, xx, xx
LD IY, (xxxx)	FD2Axxxx	253, 42, xx, xx
LD L, (HL)	6E	110
LD L, (IX + d)	DD6Edd	221, 110, xx
LD L, (IY + d)	FD6Edd	253, 110, xx
LD L, A	6F	111
LD L, B	68	104
LD L, C	69	105
LD L, D	6A	106
LD L, E	6B	107
LD L, H	6C	108
LD L, L	6D	109
LD L, xx	2Exx	46, xx
LD R, A	ED4F	237, 79
LD SP, (xxxx)	ED7Bxxxx	237, 123, xx, xx
LD SP, HL	F9	249
LD SP, IX	DDF9	221, 249

Código de Operação	*Códigos Hexadecimais*	*Códigos Decimais*
LD SP, IY	FDF9	253, 249
LD SP, xxxx	31xxxx	49, xx, xx
LDD	EDA8	237, 168
LDDR	EDB8	237, 184
LDI	EDA0	237, 160
LDIR	EDB0	237, 176
NEG	ED44	237, 68
NOP	00	0
OR (HL)	B6	182
OR (IX + d)	DDB6dd	221, 182, xx
OR (IY + d)	FDB6dd	253, 182, xx
OR A	B7	183
OR B	B0	176
OR C	B1	177
OR D	B2	178
OR E	B3	179
OR H	B4	180
OR L	B5	181
OR xx	F6xx	246, xx
OTDR	EDBB	237, 187
OTIR	EDB3	237, 179
OUT (C), A	ED79	237, 121
OUT (C), B	ED41	237, 65
OUT (C), C	ED49	237, 73
OUT (C), D	ED51	237, 81
OUT (C), E	ED59	237, 89
OUT (C), H	ED61	237, 97
OUT (C), L	ED69	237, 105
OUT (xx), A	D3xx	211, xx
OUTD	EDAB	237, 171
OUTI	EDA3	237, 163
POP AF	F1	241
POP BC	C1	193
POP DE	D1	209

POP HL	E1	225
POP IX	DDE1	221, 225
POP IY	FDE1	253, 225
PUSH AF	F5	245
PUSH BC	C5	197
PUSH DE	D5	213
PUSH HL	E5	229
PUSH IX	DDE5	221, 229
PUSH IY	FDE5	253, 229
RES 0, (HL)	CB86	203, 134
RES 0, (IX + d)	DDCBdd86	221, 203, xx, 134
RES 0, (IY + d)	FDCBdd86	253, 203, xx, 134
RES 0, A	CB87	203, 135
RES 0, B	CB80	203, 128
RES 0, C	CB81	203, 129
RES 0, D	CB82	203, 130
RES 0, E	CB83	203, 131
RES 0, H	CB84	203, 132
RES 0, L	CB85	203, 133
RES 1, (HL)	CB8E	203, 142
RES 1, (IX + d)	DDCBdd8E	221, 203, xx, 142
RES 1, (IY + d)	FDCBdd8E	253, 203, xx, 142
RES 1, A	CB8F	203, 143
RES 1, B	CB88	203, 136
RES 1, C	CB89	203, 137
RES 1, D	CB8A	203, 138
RES 1, E	CB8B	203, 139
RES 1, H	CB8C	203, 140
RES 1, L	CB8D	203, 141
RES 2, (HL)	CB96	203, 150
RES 2, (IX + d)	DDCBdd96	221, 203, xx, 150
RES 2, (IY + d)	FDCBdd96	253, 203, xx, 150
RES 2, A	CB97	203, 151
RES 2, B	CB90	203, 144
RES 2, C	CB91	203, 145
RES 2, D	CB92	203, 146
RES 2, E	CB93	203, 147
RES 2, H	CB94	203, 148

RES 2, L	CB95	203, 149
RES 3, (HL)	CB9E	203, 158
RES 3, (IX + d)	DDCBdd9E	221, 203, xx, 158
RES 3, (IY + d)	FDCBdd9E	253, 203, xx, 158
RES 3, A	CB9F	203, 159
RES 3, B	CB98	203, 152
RES 3, C	CB99	203, 153
RES 3, D	CB9A	203, 154
RES 3, E	CB9B	203, 155
RES 3, H	CB9C	203, 156
RES 3, L	CB9D	203, 157
RES 4, (HL)	CBA6	203, 166
RES 4, (X + d)	DDCBddA6	221, 203, xx, 166
RES 4, (IY + d)	FDCBddA6	253, 203, xx, 166
RES 4, A	CBA7	203, 167
RES 4, B	CBA0	203, 160
RES 4, C	CBA1	203, 161
RES 4, D	CBA2	203, 162
RES 4, E	CBA3	203, 163
RES 4, H	CBA4	203, 164
RES 4, L	CBA5	203, 165
RES 5, (HL)	CBAE	203, 174
RES 5, (IX + d)	DDCBddAE	221, 203, xx, 174
RES 5, (IY + d)	FDCBddAE	253, 203, xx, 174
RES 5, A	CBAF	203, 175
RES 5, B	CBA8	203, 168
RES 5, C	CBA9	203, 169
RES 5, D	CBAA	203, 170
RES 5, E	CBAB	203, 171
RES 5, H	CBAC	203, 172
RES 5, L	CBAD	203, 173
RES 6, (HL)	CBB6	203, 182
RES 6, (IX + d)	DDCBddB6	221, 203, xx, 182
RES 6, (IY + d)	FDCBddB6	253, 203, xx, 182
RES 6, A	CBB7	203, 183
RES 6, B	CBB0	203, 176
RES 6, C	CBB1	203, 177

RES 6, D	CBB2	203, 178
RES 6, E	CBB3	203, 179
RES 6, H	CBB4	203, 180
RES 6, L	CBB5	203, 181
RES 7, (HL)	CBBE	203, 190
RES 7, (IX + d)	DDCBddBE	221, 203, xx, 190
RES 7, (IY + d)	FDCBddBE	253, 203, xx, 190
RES 7, A	CBBF	203, 191
RES 7, B	CBB8	203, 184
RES 7, C	CBB9	203, 185
RES 7, D	CBBA	203, 186
RES 7, E	CBBB	203 187
RES 7, H	CBBC	203, 188
RES 7, L	CBBD	203, 189
RET	C9	201
RET C	D8	216
RET M	F8	248
RET NC	D0	208
RET NZ	C0	192
RET P	F0	240
RET PE	E8	232
RET PO	E0	224
RET Z	C8	200
RETI	ED4D	237, 77
RETN	ED45	237, 69
RL (HL)	CB16	203, 22
RL (IX + d)	DDCBdd16	221, 203, xx, 22
RL (IY + d)	FDCBdd16	253, 203, xx, 22
RL A	CB17	203, 23
RL B	CB10	203, 16
RL C	CB11	203, 17
RL D	CB12	203, 18
RL E	CB13	203, 19
RL H	CB14	203, 20
RL L	CB15	203, 21
RLA	17	23
RLC (HL)	CB06	203, 6

RLC (IX + d)	DDCBdd06	221, 203, xx, 6
RLD (IY + d)	FDCBdd06	253, 203, xx, 6
RLC A	CB07	203, 7
RLC B	CB00	203, 0
RLC C	CB01	203, 1
RLC D	CB02	203, 2
RLC E	CB03	203, 3
RLC H	CB04	203, 4
RLC L	CB05	203, 5
RLCA	07	7
RLD	ED6F	237, 111
RR (HL)	CB1E	203, 30
RR (IX + d)	DDCBdd1E	221, 203, xx, 1E
RR (IY + d)	FDCBdd1E	253, 203, xx, 1E
RR A	CB1F	203, 31
RR B	CB18	203, 24
RR C	CB 19	203, 25
RR D	CB1A	203, 26
RR E	CB1B	203, 27
RR H	CB1C	203, 27
RR L	CB1D	203, 28
RRA	1F	31
RRC (HL)	CB0E	203, 14
RRC (IX + d)	DDCBdd0E	221, 203, xx, 14
RRC (IY + d)	FDCBdd0E	253, 203, xx, 14
RRC A	CB0F	203, 15
RRC B	CB08	203, 8
RRC C	CB09	203, 9
RRC D	CB0A	203, 10
RRC E	CB0B	203, 11
RRC H	CB0C	203, 12
RRC L	CB0D	203, 13
RRCA	0F	15
RRD	ED67	237, 103
RST 0	C7	199
RST 10h	D7	215
RST 18h	DF	223

RST 20h	E7	231
RST 28h	EF	239
RST 30h	F7	247
RST 38h	FF	255
RST 8	CF	207
SBC A, (HL)	9E	158
SBC A, (IX + d)	DD9Edd	221, 158, xx
SBC A, (IY + d)	FD9Edd	253, 158, xx
SBC A, A	9F	159
SBC A, B	98	152
SBC A, C	99	153
SBC A, D	9A	154
SBC A, E	9B	155
SBC A, H	9C	156
SBC A, L	9D	157
SBC A, xx	DExx	222, xx
SBC HL, BC	ED42	237, 66
SBC HL, DE	ED52	237, 82
SBC HL, HL	ED62	237, 98
SBC HL, SP	ED72	237, 114
SCF	37	55
SET 0, (HL)	CBC6	203, 198
SET 0, (IX + d)	DDCBddC6	221, 203, xx, 198
SET 0, (IY + d)	FDCBddC6	253, 203, xx, 198
SET 0, A	CBC7	203, 199
SET 0, B	CBC0	203, 192
SET 0, C	CBC1	203, 193
SET 0, D	CBC2	203, 194
SET 0, E	CBC3	203, 195
SET 0, H	CBC4	203, 196
SET 0, L	CBC5	203, 197
SET 1, (HL)	CBCE	203, 206
SET 1, (IX + d)	DDCBddCE	221, 203, xx, 206
SET 1, (IY + d)	FDCBddCE	253, 203, xx, 206
SET 1, A	CBCF	203, 207
SET 1, B	CBC8	203, 200

SET 1, C	CBC9	203, 201
SET 1, D	CBCA	203, 202
SET 1, E	CBCB	203, 203
SET 1, H	CBCC	203, 204
SET 1, L	CBCD	203, 205
SET 2, (HL)	CBD6	203, 214
SET 2, (IX + d)	DDCBddD6	221, 203, xx, 214
SET 2, (IY + d)	FDCBddD6	253, 203, xx, 214
SET 2, A	CBD7	203, 215
SET 2, B	CBD0	203, 208
SET 2, C	CBD1	203, 209
SET 2, D	CBD2	203, 210
SET 2, E	CBD3	203, 211
SET 2, H	CBD4	203, 212
SET 2, L	CBD5	203, 213
SET 3, (HL)	CBDE	203, 222
SET 3, (IX + d)	DDCBddDE	221, 203, xx, 222
SET 3, (IY + d)	FDCBddDE	253, 203, xx, 222
SET 3, A	CBDF	203, 223
SET 3, B	CBD8	203, 216
SET 3, C	CBD9	203, 217
SET 3, D	CBDA	203, 218
SET 3, E	CBDB	203, 219
SET 3, H	CBDC	203, 220
SET 3, L	CBDD	203, 221
SET 4, (HL)	CBE6	203, 230
SET 4, (IX + d)	DDCBddE6	221, 203, xx, 230
SET 4, (IY + d)	FDCBddE6	253, 203, xx, 230
SET 4, A	CBE7	203, 231
SET 4, B	CBE0	203, 224
SET 4, C	CBE1	203, 225
SET 4, D	CBE2	203, 226
SET 4, E	CBE3	203, 227
SET 4, H	CBE4	203, 228
SET 4, L	CBE5	203, 229
SET 5, (HL)	CBEE	203, 238
SET 5, (IX + d)	DDCBddEE	221, 203, xx, 238

SET 5, A	CBEF	203, 239
SET 5, B	CBE8	203, 232
SET 5, C	CBE9	203, 233
SET 5, D	CBEA	203, 234
SET 5, E	CBEB	203, 235
SET 5, H	CBFC	203, 236
SET 5, L	CBED	203, 237
SET 6, (HL)	CBF6	203, 246
SET 6, (IX + d)	DDCBddF6	221, 203, xx, 246
SET 6, (IY + d)	FDCBddF6	253, 203, xx, 246
SET 6, A	CBF7	203, 247
SET 6, B	CBF0	203, 240
SET 6, C	CBF1	203, 241
SET 6, D	CBF2	203 242
SET 6, E	CBF3	203, 243
SET 6, H	CBF4	203, 244
SET 6, L	CBF5	203, 245
SET 7, (HL)	CBFE	203 254
SET 7, (IX + d)	DDCBddFE	221, 203, xx, 254
SET 7, (IY + d)	FDCBddFE	253, 203, xx, 254
SET 7, A	CBFF	203, 255
SET 7, B	CBF8	203, 248
SET 7, C	CBF9	203, 249
SET 7, D	CBFA	203, 250
SET 7, E	CBFB	203, 251
SET 7, H	CBFC	203, 252
SET 7, L	CBFD	203, 253
SLA (HL)	CB26	203, 38
SLA (IX + d)	DDCBdd26	221, 203, xx, 38
SLA (IY + d)	FDCBdd26	253, 203, xx, 38
SLA A	CB27	203, 39
SLA B	CB20	203, 32
SLA C	CB21	203, 33
SLA D	CB22	203, 34
SLA E	CB23	203, 35
SLA H	CB24	203, 36
SLA L	CB25	203, 37

SRA (HL)	CB2E	203, 46
SRA (IX + d)	DDCBdd2E	221, 203, xx, 46
SRA (IY + d)	FDCBdd2E	253, 203, xx, 46
SRA A	CB2F	203, 47
SRA B	CB28	203, 40
SRA C	CB29	203, 41
SRA D	CB2A	203, 42
SRA E	CB2B	203, 43
SRA H	CB2C	203, 44
SRA L	CB2D	203, 45
SRL (HL)	CB3E	203, 62
SRL (IX + d)	DDCBdd3E	221, 203, xx, 62
SRL (IY + d)	FDCBdd3E	253, 203, xx, 62
SRL A	CB3F	203, 63
SRL B	CB38	203, 56
SRL C	CB39	203, 57
SRL D	CB3A	203, 58
SRL E	CB3B	203, 59
SRL H	CB3C	203, 60
SRL L	CB3D	203, 61
SUB (HL)	96	150
SUB (IX + d)	DD96dd	221, 150, xx
SUB (IY + d)	FD96dd	253, 150, xx
SUB A	97	151
SUB B	90	144
SUB C	91	145
SUB D	92	146
SUB E	93	147
SUB H	94	148
SUB L	95	149
XOR (HL)	AE	174
XOR (IX + d)	DDAEdd	221, 174, xx
XOR (IY + d)	FDAEdd	253, 174, xx
XOR A	AF	175
XOR B	A8	168
XOR C	A9	169
XOR D	AA	170

XOR E	AB	171
XOR H	AC	172
XOR L	AD	173
XOR xx	EEx	238, xx

Gráfica Palas Athena
Associação "Palas Athena" do Brasil
Rua José Bento, 384
Fone: 279-6288 – CEP 01523
Cambuci – São Paulo